Efficient and Privacy-Preserving Compressive Learning

Antoine Chatalic

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Thèse de doctorat de

L'UNIVERSITÉ DE RENNES 1
Comue Université Bretagne Loire

École Doctorale n° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité: *Signal, Image, Vision*

Par
Antoine CHATALIC

Efficient and Privacy-Preserving Compressive Learning

Thèse présentée et soutenue à Rennes, le 19 novembre 2020
Unité de recherche: IRISA

Rapporteurs avant soutenance:

Francis Bach	Directeur de recherche – Inria Paris
Lorenzo Rosasco	Professeur – Université de Gênes

Composition du Jury:

Présidente:	Magalie Fromont Renoir	Professeure – Université de Rennes 2
Examinateurs:	Francis Bach	Directeur de recherche – Inria Paris
	Lorenzo Rosasco	Professeur – Université de Gênes
	Jamal Atif	Professeur – Université Paris-Dauphine
	Mike Davies	Professeur – Université d'Édimbourg
	Marc Tommasi	Professeur – Université de Lille
Directeur de thèse:	Rémi Gribonval	Directeur de recherche – Inria

Remerciements

« Cher lecteur, c'est peut-être de la haine que tu veux que j'invoque dans le commencement de cet ouvrage ! ». Que nenni, précisons d'emblée qu'une thèse peut se dérouler dans d'excellentes conditions. C'est pourquoi quelques remerciements s'imposent avant d'entrer dans le vif du sujet.

En premier lieu, j'adresse tout naturellement mais néanmoins sincèrement mes remerciements à Rémi Gribonval pour la qualité de son encadrement tout au long de la thèse et sa disponibilité. Ses commentaires avisés ainsi que ses relectures détaillées furent d'une aide précieuse, cela est certain ; j'ai toutefois également été impressionné par sa patience et son optimisme indéfectible ; j'ai par ailleurs apprécié sa capacité à prendre rapidement du recul là où j'en manquais cruellement — ce qui a certainement permis d'éviter quelques écueils.

Je souhaite remercier tous les membres du jury de m'avoir fait l'honneur d'accepter d'évaluer mon travail, et en particulier Francis Bach et Lorenzo Rosasco qui, en tant que rapporteurs, ont dû lire avec une attention particulière ce manuscrit.

Que Patrick Pérez et Frédéric Bimbot soient également remerciés pour avoir, au sein du comité de suivi individuel doctoral, partagé leur avis sur l'avancement des travaux et apporté un regard critique complémentaire et bienvenu.

La partie III de ce manuscrit est le résultat d'une collaboration avec Phil Schniter et Evan Byrne (Ohio State University) ; je les remercie tous les deux de m'avoir accueilli pour quelques mois à Columbus, et d'avoir pris le temps de répondre à toutes mes questions. Cette mobilité a par ailleurs été possible grâce au soutien financier de l'OSU et du GdR MIA.

La partie IV est, elle aussi, le fruit d'une collaboration. J'adresse par conséquent également des remerciements à Vincent Schellekens, avec qui la discussion fut initiée à Peyresq, ainsi qu'à Florimond Houssiau et Laurent Jacques avec qui ces travaux ont été réalisés. Ce fut un plaisir d'avancer sur ce sujet en votre compagnie !

À Rennes, j'ai une pensée pour tous les panaméens qui ont su entretenir la vie d'équipe, que ce soit d'un point de vue scientifique ou simplement en contribuant à la bonne humeur générale. À cet égard, nos voisins situés un peu plus loin dans le couloir ne sauraient être oubliés ! C'est sans aucun doute à la cafétéria que la science se diffuse le plus, et il convient donc de souligner l'importance des sempiternelles discussions sur la nature du chocolat, les tenseurs de lapins, la beauté

des monades et la redynamisation de l'eau par ondes cosmiques. Un merci également à tous ceux qui ont donné du temps et de l'énergie pour organiser la journée science et musique, car il y avait dans cette initiative, il me semble, également beaucoup de bonne volonté et de bonne humeur.

Bien entendu, je n'oublie aucunement toutes les personnes avec qui j'ai partagé de très bons moments hors du labo pendant toutes ces années, que ce soit en soirée à Rennes, à l'escalade, autour de jeux de société, lors de concerts et de festivals en tous genres, ou encore d'escapades improvisées aux confins de la Bretagne et au delà. Je n'égrènerai pas vos noms ici, mais ne vous fourvoyez pas à croire que je vous oublie car il n'en est rien, bien au contraire !

Je remercie ma famille pour le soutien constant qu'elle m'a apporté, et pour avoir toujours mis toutes les chances de mon côté pour que je puisse parvenir jusque là ; il semblerait que cela n'ait pas si mal fonctionné.

Enfin, j'adresse un remerciement teinté de tendresse et d'admiration à Katharina, avec qui j'ai partagé de nombreux moments ces dernières années, et qui m'a soutenu patiemment lors du confinement tandis que je reformulais un nombre exagéré de fois les phrases des chapitres à venir.

Résumé des travaux en français

> **Note:** This part contains a summary in french of the manuscript. The rest of the document is written in english, and the introduction can be found in Chapter 1.

Le domaine de l'apprentissage automatique, qui s'intéresse aux techniques permettant d'apprendre un modèle mathématique à partir d'une collection de données, a connu des avancées fulgurantes ces dernières décennies. Parmi les multiples facteurs ayant concouru au développement de ces méthodes, l'augmentation drastique des volumes de données produites, collectées et utilisées à des fins d'apprentissage figure en bonne position. L'invention du microprocesseur au début de la décennie 1970 et le processus de miniaturisation qui s'en est suivi ont en effet permis une augmentation exponentielle de la puissance de calcul disponible, mais également du nombre de capteurs[1] permettant d'enregistrer des données et des capacités de stockage. Le développement d'internet et de la toile en particulier ont, à l'évidence, également contribué à accroître les volumes de données disponibles, et donc la pertinence statistique des modèles mathématiques qui en sont issus.

[1] Au sens large : instruments industriels et scientifiques comme le LHC, ou encore terminaux mobiles multifonctions et appareils photos numériques, dont la démocratisation a aidé à perfectionner les techniques de vision par ordinateur.

Toutefois, et en dépit des progrès matériels[2], l'apprentissage à partir de vastes volumes de données reste une opération coûteuse en temps comme en énergie, et qui requiert des investissements importants. Ce manuscrit s'intéresse à l'apprentissage compressif, une technique introduite il y a quelques années[3] qui consiste à compresser l'ensemble des données avant apprentissage. Plus précisément, le jeu de données utilisé, que nous représentons ici comme une matrice $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ dont les colonnes correspondent à n observations numériques en dimension d, est compressé en une unique *empreinte* de la forme

[2] Augmentation des capacités de calcul à coût énergétique ou monétaire constant.

[3] L'obtention de garanties d'apprentissage est récente, bien que des techniques plus anciennes telles que la méthode des moments généralisés [1] continssent déjà des idées semblables.

$$\tilde{\mathbf{s}} = \frac{1}{n} \sum_{i=1}^{n} \Phi(\mathbf{x}_i), \tag{1}$$

où la fonction Φ est adaptée à la tâche d'apprentissage considérée, mais typiquement non linéaire et aléatoire. Un intérêt particulier est porté aux fonctions de la forme $\Phi : \mathbf{x} \mapsto \rho(\boldsymbol{\Omega}^T \mathbf{x}) \in \mathbb{R}^m$ ou $\mathbb{C}^m$, où $\boldsymbol{\Omega} \in \mathbb{R}^{d \times m}$ est une matrice aléatoire[4], et la fonction ρ est scalaire, déterministe et appliquée point à point. Le vecteur $\tilde{\mathbf{s}}$ peut alors être vu comme une collection de moments généralisés et aléatoires des données. Un exemple important consiste à choisir $\rho = \exp(\imath \cdot)$ (où $\imath = \sqrt{-1}$), auquel cas l'empreinte correspond à des échantillons aléatoires de la fonction caractéristique empirique des données[5].

[4] La matrice $\boldsymbol{\Omega}$ est tirée aléatoirement, mais une unique fois et avant compression, de sorte à ce que tous les échantillons $\mathbf{x}_i$ soient compressés de la même manière.

[5] Et donc de la transformée de Fourier (inverse) de la densité de probabilité, lorsque celle-ci existe.

En apprentissage statistique, une tâche d'apprentissage est représentée par une fonction de risque $\mathcal{R}(h, \pi)$, qui mesure l'inadéquation du modèle mathématique h vis-à-vis de la distribution de probabi-

lité π pour la tâche en question. Résoudre le problème d'apprentissage revient donc à trouver un minimiseur du risque, le plus souvent au sein d'une famille de modèles $\mathcal{H}$, i.e. on cherche à trouver $h^* \in \arg\min_{h\in\mathcal{H}} \mathcal{R}(h, \pi)$. La distribution sous-jacente π des données est en général inconnue, mais il est en revanche possible d'utiliser la distribution empirique $\pi_{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n} \delta_{\mathbf{x}_i}$ associée aux données mesurées, c'est à dire d'essayer de résoudre le problème d'optimisation $\min_{h\in\mathcal{H}} \mathcal{R}(h, \pi_{\mathbf{X}})$; on parle alors de minimisation du risque empirique.

Toutefois, évaluer $\mathcal{R}(h, \pi_{\mathbf{X}})$, même pour un unique modèle $h \in \mathcal{H}$, nécessite de parcourir les données dans leur intégralité, ce qui est coûteux et limite fortement l'utilité de cette méthode en pratique. Avec l'approche compressive, l'apprentissage est effectué en substituant à la fonction de risque $\mathcal{R}(\cdot, \pi_{\mathbf{X}})$ un succédané $f(\cdot, \tilde{\mathbf{s}})$ dans lequel les données n'interviennent qu'exclusivement via le vecteur $\tilde{\mathbf{s}}$. Dès lors, il est seulement nécessaire de calculer l'empreinte $\tilde{\mathbf{s}}$, suite à quoi les données peuvent être oubliées. D'autre part, la forme simpliste du sketch (1) en rend le calcul hautement parallélisable, et compatible avec des données déjà distribuées ou même en flux[6].

Cette approche a déjà été utilisée avec succès pour le problème de partitionnement type k-moyennes [2], pour la modélisation de densité avec modèle de mélange gaussien [3, 4], ainsi que pour l'analyse en composantes principales. Des garanties d'apprentissage statistique[7] ont été établies pour ces trois problèmes [5, 6].

Cette thèse propose d'étendre le cadre de l'apprentissage compressif dans plusieurs directions : l'étude du choix de la matrice aléatoire $\boldsymbol{\Omega}$, et notamment la proposition d'utiliser des matrices structurées afin d'accélérer la complexité du mécanisme dans son ensemble ; la proposition d'un nouvel algorithme pour l'apprentissage à partir de l'empreinte pour le problème de partitionnement ; et enfin l'introduction d'un mécanisme de compression légèrement modifié pour lequel des garanties de confidentialité peuvent être obtenues. Nous proposons de résumer ces contributions principales en suivant l'organisation du manuscrit.

[6] C'est-à-dire, on peut très bien calculer l'empreinte d'un flux de données en compressant les éléments au fur et à mesure qu'ils arrivent.

[7] Ces garanties portent sur le contrôle du risque, en supposant que le problème d'optimisation peut être résolu. Ce dernier étant généralement non convexe, diverses heuristiques sont utilisées pour le résoudre de manière approchée, et peu de garanties existent sur ces heuristiques.

Partie I La première partie de ce manuscrit propose un aperçu de la littérature existante sur le sujet de l'apprentissage à grande échelle.

- Le chapitre 2 introduit le domaine de l'apprentissage compressif, en partant de la technique d'acquisition comprimée qui en fournit l'inspiration. L'empreinte d'un jeu de données étant constituée d'un ensemble de moments, elle peut s'exprimer comme l'application de l'opérateur linéaire $\mathcal{A} : \pi \mapsto \mathbf{E}_{\mathbf{x}\sim\pi}\Phi(\mathbf{x})$ à la distribution empirique $\pi_{\mathbf{X}}$ du jeu de données. La tâche d'apprentissage peut alors être formulée comme un problème linéaire inverse sur un espace de distributions de probabilités, pour lequel des garanties théoriques peuvent être obtenus lorsque $\mathcal{A}$ satisfait une inégalité d'isométrie restreinte. Quelques outils théoriques liés aux espaces de Hilbert à noyaux reproduisants sont également introduits afin d'éclairer la construction de la fonction Φ.
- Dans le chapitre 3, nous proposons une vue d'ensemble des autres

types d'approches pour l'apprentissage automatique efficace à grande échelle, en grande dimension et/ou sur des données en flux. Plusieurs exemples d'empreintes classiques pour l'estimation de fréquences sur de grands volumes de données sont évoqués, en partant de méthodes proposées au siècle dernier dans la communauté des bases de données. Les méthodes de réduction de dimensionalité sont ensuite introduites, avec un intérêt particulier porté sur les méthodes linéaires stochastiques et agnostiques vis-à-vis des données, de type Johnson-Lindenstrauss. La notion de *coreset*[8] est définie, et l'utilisation du sous-échantillonnage pour la production de *coresets* est discutée. Enfin, plusieurs algorithmes stochastiques pour l'algèbre linéaire et en particulier l'approximation de faible rang sont présentés.

[8] Un *coreset* est un sous-ensemble du jeu de données pour lequel l'erreur d'apprentissage reste proche de l'erreur mesurée sur le jeu de données entier.

Partie II Dans la seconde partie, nous étudions le rôle de la distribution de la matrice aléatoire $\boldsymbol{\Omega}$, qui était dans les travaux précédents toujours tirée avec des entrées indépendantes et identiquement distribuées (i.i.d.) gaussiennes. Si les colonnes $(\boldsymbol{\omega}_i)_{1\leq i\leq m}$ de $\boldsymbol{\Omega}$ sont décomposées de la forme $\boldsymbol{\omega}_i = R_i\boldsymbol{\varphi}_i$ pour tout $i \in \{1, ..., m\}$, où les $(R_i)_{1\leq i\leq m}$ sont des rayons et les $(\boldsymbol{\varphi}_i)_{1\leq i\leq m}$ des vecteurs sur la sphère unité, nous proposons deux contributions distinctes, l'une relative à la distribution radiale des $(R_i)_{1\leq i\leq m}$ dans le cas du problème de partitionnement, et l'autre relative à la distribution directionnelle des $(\boldsymbol{\varphi}_i)_{1\leq i\leq m}$ et qui peut s'appliquer à diverses tâches d'apprentissage.

- Le chapitre 4 s'intéresse au problème de partitionnement compressif uniquement, pour lequel $\rho = \exp(\iota\cdot)$, et souligne expérimentalement l'importance de bien choisir la distribution radiale de $R_1, ..., R_m$. Les contributions sont de nature empirique, et mettent en évidence le lien existant entre le choix de l'échelle de la distribution radiale et la séparation (c.-à-d. la distance minimale) entre les différents groupes à identifier.
- Le chapitre 5 à l'inverse, suppose qu'une bonne distribution radiale est connue, et s'intéresse au choix de la distribution des vecteurs $\boldsymbol{\varphi}_1, ..., \boldsymbol{\varphi}_m$, c'est-à-dire à la distribution directionnelle. En particulier, il est suggéré que ces vecteurs peuvent être générés de manière corrélée par blocs afin de réduire la complexité algorithmique du processus de compression – et donc d'apprentissage puisque le succédané $f(\cdot, \tilde{\mathbf{s}})$ requiert également d'évaluer l'opérateur $\mathcal{A}$. Cette réduction du coût est obtenue en construisant $\boldsymbol{\Omega}$ comme une juxtaposition de blocs carrés faisant intervenir des matrices structurées de type Walsh-Hadamard, pour lesquelles des algorithmes de multiplication rapide existent. Une validation expérimentale de la méthode est proposée, et les problèmes restants en vue de l'obtention de garanties théoriques sont identifiés et discutés.

Partie III Constituée d'un unique chapitre, la troisième partie se concentre encore une fois sur le problème de partitionnement de type

k-moyennes, où l'on cherche à apprendre la position spatiale de k points correspondant aux centres des k groupes – chaque donnée $\mathbf{x}_i$ appartenant alors implicitement au groupe lié à celui des k points dont elle est le plus proche.

- Le Chapitre 6 introduit un algorithme permettant de retrouver ces k points à partir de l'empreinte $\tilde{\mathbf{s}}$. Cette dernière est calculée via la même fonction Φ que dans les contributions précédentes [2], mais l'algorithme diffère des précédentes approches car il repose sur des méthodes de propagation de convictions. Cette famille de méthodes et notamment les algorithmes par passage de messages sont introduits, puis nous montrons comment le problème de partitionnement compressif peut se réécrire[9] sous la forme d'un problème d'inférence bayésienne, qui peut être abordé avec de tels algorithmes moyennant quelques approximations. Quelques paramètres du modèle doivent être réglés, et nous montrons comment cela peut être effectué simultanément[10] au déroulement de l'algorithme de passage de messages. Nous évoquons les problèmes d'approximation numérique qui peuvent survenir, et proposons une validation expérimentale de la méthode, à la fois sur des données synthétiques et réelles. Ce chapitre est le fruit d'une collaboration, et ma contribution réside principalement dans la mise en œuvre expérimentale de la méthode.

[9] Sous l'hypothèse de données générées suivant un modèle de mélange gaussien.

[10] Plus précisément, de manière entrelacée puisque nous alternons les itérations des deux méthodes.

Partie IV La dernière partie du manuscrit étudie l'intérêt de l'approche compressive pour l'apprentissage avec garanties de confidentialité, ce qui s'avère être un critère essentiel lorsque l'on souhaite apprendre un modèle à partir de données à caractère personnel.

- Le chapitre 7 introduit le formalisme de la confidentialité différentielle, ainsi que les méthodes classiques permettant d'adapter un algorithme préexistant au moyen d'une perturbation aléatoire additive pour qu'il satisfasse cette définition. Une vue d'ensemble des autres techniques permettant de satisfaire la propriété de confidentialité différentielle est également proposée pour les tâches de partitionnement et d'analyse en composantes principales.
- Dans le chapitre 8, nous introduisons un mécanisme de compression bruité, et nous nous intéressons au niveau de bruit minimum à ajouter permettant d'obtenir un niveau de confidentialité (différentielle) donné. Cette étude est effectuée pour deux types d'empreintes en particulier, et donne des garanties pour les problèmes de partitionnement et d'analyse en composantes principales. Une variante de cet algorithme adjoignant au bruit additif un mécanisme de sous-échantillonnage des contributions[11] liées aux différents échantillons du jeu de données est également étudiée, permettant ainsi de contrôler plus finement le compromis entre confidentialité, efficacité et qualité de l'apprentissage.
- Enfin, nous montrons expérimentalement dans le chapitre 9 que la qualité d'apprentissage à partir d'une empreinte bruitée est fortement corrélée au rapport signal sur bruit[12]. Nous proposons donc

[11] i.e. seulement quelques entrées de $\Phi(\mathbf{x}_i)$ sont calculées pour chaque $\mathbf{x}_i$, mais tous les échantillons $(\mathbf{x}_i)_{1\leq i\leq n}$ entrent en compte dans le calcul de l'empreinte.

[12] C'est à dire le rapport entre l'énergie du bruit ajouté pour obtenir la garantie de confidentialité, et l'énergie de l'empreinte non bruitée.

d'utiliser cette quantité, pour laquelle une expression analytique est fournie, comme un critère permettant de guider le choix des différents paramètres du modèle. Les performances des méthodes introduites sont également mesurées expérimentalement pour les problèmes de partitionnement et d'analyse en composantes principales, et comparées à d'autres méthodes de l'état de l'art.

Certaines définitions et preuves sont omises des chapitres et produites en annexes A.1, B et C. Quelques considérations sur l'implémentation des diverses méthodes sont également proposées en Annexe D.

Contents

Notations

We summarize here the notations used throughout the paper.

Mathematical notations

Notation	Meaning
$\mathcal{O}(\cdot)$, $\Theta(\cdot)$, $\Omega(\cdot)$	Landau asymptotic notations
$[\![n, m]\!]$	Set of integers between n and m (included)
$\triangleq$	Variable definition
$\overset{\triangle}{\propto}$	Definition of a probability density function up to normalization
$\lesssim$, $\gtrsim$	Inequalities up to a constant factor
S^{d-1}	Unit sphere in $\mathbb{R}^d$ (for the l_2-norm)
$\mathbf{A}^T$	Transpose of $\mathbf{A}$
$\mathbf{A}^*$	Hermitian (conjugate) transpose of $\mathbf{A}$
$\Re$, $\Im$	Real and imaginary parts
$\mathbf{I}_d$	Identity matrix of size $d \times d$
$\otimes_k$, $\otimes$	Kronecker product, Outer product
$O(d)$	Real orthogonal group
i.i.d.	identically and independently distributed
w.h.p.	with high probability
w.l.o.g.	without loss of generality
psd	positive semi-definite
$\|\cdot\|_0$	l_0 vector pseudo-norm (number of nonzero entries)
$\|\cdot\|_1, \|\cdot\|_2$	l_1 and l_2 vector norms
$\langle\cdot,\cdot\rangle_F, \|\cdot\|_F$	Frobenius (matrix) inner-product and norm
$\|\cdot\|$	Spectral (matrix) norm
$\|\cdot\|_{p\to q}$	$l_p \to l_q$ operator (matrix) norm
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{L}(b)$	Centered Laplace distribution
$\mathcal{L}^{\mathbb{C}}(b)$	Centered complex Laplace distribution (Definition 7.7)
Bern (p)	Bernouilli distribution with parameter p
$\mathcal{U}(S)$	Uniform distribution on S (when properly defined)

Conventions regarding variable names

Notation	Meaning
d	Dimension of the data samples
n	Dataset size (number of samples)
k	Number of components in a mixture, number of clusters for k-means
m	Sketch size
Σ_s	Set of s-sparse vectors
Φ	Feature map
Φ^{RFF}	Random Fourier feature map (cf. Definition 2.5)
Φ^{RQF}	Random quadratic feature map (cf. Definition 2.6)
$\mathcal{A}$	Sketching operator (cf. (2.10))

Chapter 1
Introduction

Contents

The amount of digital data which is produced, transmitted, stored and processed each day has never stopped growing in the last centuries. The early days of computing are often dated circa 1837, with Charles Babbage's analytical engine which is the first design of a generic Turing-complete[1] computer, and Ada Lovelace's notes which contain what is considered as the first computer program [7]. But at the time, and even in the following century when first computers were successfully built – the analytical engine was never finalized –, both speed and memory were quite limited by today's standards. For instance, the Z3 of Konrad Zuse, completed in 1941, had only 64 memory words of 22 bits each; program memory was made of punched tape, following Babbage's ideas, and a multiplication took in average three seconds. These computers, albeit being programmable, were dedicated to specific engineering tasks or cryptography.

Although the computer science discipline was created and developed in the following decades, with already many significant theoretical contributions, the biggest game changer was certainly the invention of metal-oxide-silicon field-effect transistors (MOSFET), which led to the development of microprocessors and memory chips, opening the way for mass production of personal computers, embedded devices, and smartphones. This multiplication of the number of devices able to capture and process external data, together with the exponential[2] growing of computational power and the increasing availability of permanent storage (hard drives), led us quickly where we stand now: massive amounts of digital data are produced every second – not only by individuals, but also for instance by sensor networks or scientific devices –, transmitted via telecommunication infrastructures, stored in gigantic datacenters across the world, and processed, possibly using massively parallel supercomputers.

The domain of artificial intelligence (AI), which encompasses all endeavors towards designing machines or programs able to mimic, to some extent, cognitive functions of human beings such as perception, reasoning or learning, flourished in the last decades partly because of this evolution. Despite significant early developments in the mid-fifties, the discipline did not grow as fast as initially envisioned by some researchers[3]. But with larger and larger data collections available over the years and growing calculation capacities, it became easier to build and train new powerful models, leading to many consecutive

[1] i.e. which is expressive enough to simulate a Turing machine.

[2] According to Moore's law [8], the transistor density (for which the production cost per transistor is minimum) used to double every two years at the time. Dennard scaling law [9] moreover states that each step in this miniaturization process came with roughly a 40% increase of circuits' frequency, and a constant power consumption per surface unit. It is worth noting, however, that production costs have also been growing exponentially (Moore's second law).

[3] e.g. Marvin Minsky, considered as one of the fathers of AI, in 1970: "In from three to eight years we will have a machine with the general intelligence of an average human being." (Life Magazine).

successes, first in the nineties – for instance in 1997, with the victory of the Deep Blue AI over Garry Kasparov at chess –, and in the last decade with deep and convolutional neural networks[4] – that improved drastically machine performance on multiple learning tasks, especially in computer vision and natural language processing. All these methods, which "learn" the parameters of a mathematical model using large collections of empirical data, form the research area of "machine learning", which can be seen as a sub-field of artificial intelligence. Needless to say, applications go way beyond the examples given above as, by definition, such techniques can be used in pretty much any domain where data is available.

[4] These are classes of mathematical models, combining parametric linear operations and simple non-linear (most often deterministic) transformations. These models are highly expressive, but also difficult to train due to the very large number of parameters they induce.

However, building efficient and lightweight machine learning algorithms remains a challenge. The size of datasets used to train the models has grown at the same rate, if not faster, than computational capacities, and most traditional learning techniques simply cannot scale to the largest existing collections. Most of the numerous successes witnessed so far, and especially the ones based on deep architectures in the last few years, require tremendous resources: powerful computers or supercomputers[5], specialized graphical or tensor processing units, large storage capacities and, naturally, substantial engineering and research capacities as well. Beyond deeper problems coming with these constraints, such as the growing difficulty for academic or smaller actors to compete with a few major players, a maybe more fundamental observation is that these learning approaches are very expensive, especially energetically.

[5] Supercomputers are computers which can perform much more floating-point operations per seconds (FLOPS) than a "standard" computer, and are widely use in computational science.

A simple way to address this problem is to somehow compress the learning collections. By doing so, we reduce the amount of stored data, make it easier – or at least faster – to process the remaining information, and make it possible to tackle with limited resources problems on large datasets that simply are untractable otherwise. This thesis focuses on one such approach, called compressive (or sketched) learning, which maps a whole training collection to a single vector summarizing some useful statistics of the data distribution.

In this introduction, we propose to formalize the key concepts of machine learning, and explain further why traditional approaches are often not helpful in the large-scale setting (Section 1.1). A quick overview of compressive learning is provided in Section 1.2, and the main contributions of the thesis are summarized in Section 1.3.1.

1.1 Problem overview

In order to compare different learning techniques, we need to define more precisely the issue to be solved. Machine learning actually covers many different problems, that are also called learning tasks. Three examples of such tasks are represented (in dimension 2) in Figure 1.1, where the blue points represent the data samples, and the learned models are depicted in purplish red. Without going too much in the details, principal component analysis (PCA) aims at finding the subspace which somehow best "fits" the data; k-means clustering consists in

finding a set of k points which cover the different clusters of the dataset; the goal of Gaussian modeling is to find, in the model of Gaussian mixtures, the distribution whose density (represented with isolines on Figure 1.1) best fits the observed data. For each of these examples, one wants to identify in a predefined family of acceptable models, the one that best explains or summarizes the observed data. Statistical learning is a convenient framework to tackle all these tasks and more under a common formalism. Its basic concepts are introduced in Section 1.1.1, and we discuss the different challenges to take into account in Section 1.1.2.

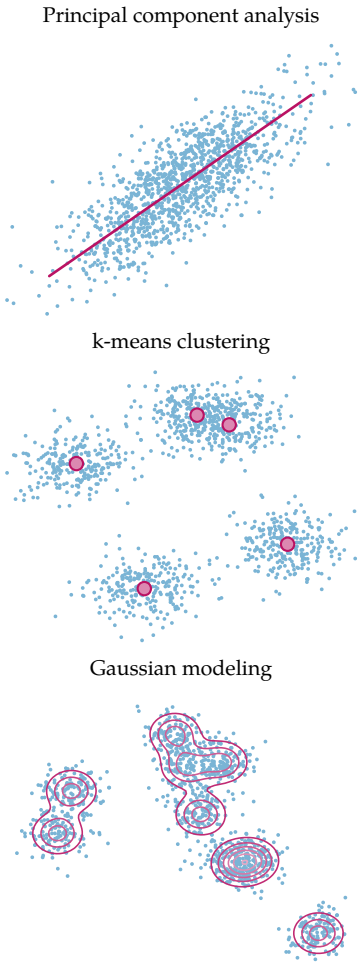


Figure 1.1: Examples of learning tasks: PCA (top), k-means clustering (middle), and density estimation with a Gaussian mixture model (bottom). Data points are represented in blue, and the learned model (the hypothesis) in purplish red.

1.1.1 A statistical approach to machine learning

Throughout this thesis, we always work with numerical features, i.e. the datasets we consider are collections of vectors in $\mathcal{X} = \mathbb{R}^d$. Unless otherwise specified, we always denote d the dimension of the data samples and n the number of samples in the collection. Any dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$ can be represented by its associated empirical distribution, that we denote $\pi_{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n} \delta_{\mathbf{x}_i}$, where $\delta_{\mathbf{x}}$ is the Dirac measure located at $\mathbf{x}$. We refer the reader to Appendix A.1 for some standard definitions related to measure theory.

The concepts of "learning task" and "model" used above can be formalized in the statistical learning framework. In this context, a model is referred to as an hypothesis, and we denote $\mathcal{H}$ the hypothesis space, i.e. the class of all considered models. It will typically be a parametrized family. A learning task is defined by a loss function $l : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$. Intuitively, the quantity $l(h, \mathbf{x})$ characterizes how "bad" the hypothesis h is with respect to the data sample $\mathbf{x}$ for this learning task. Any loss function naturally extends[6] to probability distributions via its associated risk function $\mathcal{R} : \mathcal{H} \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$, defined for any probability distribution π as

$$\mathcal{R}(h, \pi) \triangleq \mathbf{E}_{\mathbf{x}\sim\pi} l(h, \mathbf{x}). \tag{1.1}$$

where $\mathcal{P}(\mathcal{X})$ denotes the set of probability distributions over $\mathcal{X}$.

[6] We assume for conciseness that the chosen loss functions are always integrable, i.e. that (1.1) is always defined

Some classical learning tasks Learning problems of very different natures can be written using the formalism of loss and risk functions. The most iconic learning task is maybe classification[7], where one has a collection of data samples in $\mathcal{X}$ and associated labels, and wants to learn to predict labels for new data samples. We discuss this problem just below, but we first introduce the loss functions corresponding to the three tasks represented in Figure 1.1. These tasks will play an important role in this thesis for the simple reason that it is known that they can be addressed using compressive methods.

[7] To be more precise, supervised classification, as explained below.

Let us first consider the example of principal component analysis, depicted on the top of Figure 1.1. The hypothesis one wants to recover is in this case a linear subspace of $\mathcal{X}$ (we assume centered data for simplicity).

Definition 1.1 (PCA): Let $k \in \mathbb{N}$. Principal component analysis consists in finding a linear k-dimensional subspace h of $\mathcal{X}$ such that the orthogonal projection Π_h on h minimizes the risk induced by the loss function

$$l_{\text{PCA}}(h, \mathbf{x}) \triangleq \|\mathbf{x} - \Pi_h \mathbf{x}\|_2^2. \tag{1.2}$$

The loss induced by a sample that does not belong to h is thus simply its squared distance to h (e.g. in Figure 1.1, the squared distance to the purple line), and only the risk of probability distributions that are supported on a k-dimensional subspace vanishes for some subspace h. The two other problems that will be discussed extensively are k-means clustering and Gaussian modeling.

Definition 1.2 (Clustering): k-means clustering consist in finding a set h of $k \in \mathbb{N}$ points $h = \{\mathbf{c}_1, ..., \mathbf{c}_k\} \subseteq \mathcal{X}$ minimizing the risk induced by the loss function

$$l_{\text{KM}}(h, \mathbf{x}) \triangleq \min_{1 \leq i \leq k} \|\mathbf{x} - \mathbf{c}_i\|_2^2. \tag{1.3}$$

Definition 1.3 (Gaussian modeling): Gaussian mixture modeling aims at finding the parameters $h = \{(\alpha_i)_{1 \leq i \leq k}, (\mathbf{c}_i)_{1 \leq i \leq k}, (\mathbf{\Sigma}_i)_{1 \leq i \leq k}\}$ of the Gaussian mixture model with density $p_h(\mathbf{x}) = \sum_{i \leq i \leq k} \alpha_i \mathcal{N}(\mathbf{x}; \mathbf{c}_i, \mathbf{\Sigma}_i)$ minimizing the risk induced by the loss

$$l_{\text{KM}}(h, \mathbf{x}) \triangleq -\log p_h(\mathbf{x}). \tag{1.4}$$

The k-means clustering problem is NP-hard[8] [10], and heuristics such as Lloyd's algorithm [11] are traditionally used when possible. Many different variants and extensions exist [12], as clustering is a core component of many learning frameworks. Similarly, Gaussian mixtures are ubiquitous models; the de-facto approach to learn the parameters is the EM algorithm [13].

[8] i.e. at least as difficult as any problem in the NP (non-deterministic polynomial-time) complexity class.

Supervised learning The three tasks presented above belong to the group of unsupervised learning problems, by opposition to supervised problems where the data points in the training collection come with additional observations or labels. A standard example is supervised classification, where each sample comes with a category membership indication, and one can take these indications into account to better train the considered model. In that case, the loss function is of the form $l : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$, where $\mathcal{Y}$ is the finite set of possible labels. The relevance of a classifier $f : \mathcal{X} \to \mathcal{Y} = \{1, ..., c\}$ can for instance be measured with the 0-1 loss function

$$l_{0-1}(f, (\mathbf{x}, y)) \triangleq \mathbb{1}_{f(\mathbf{x}) = y} \tag{1.5}$$

where $\mathbb{1}_b$ takes the value 1 when the boolean expression b holds, and 0 otherwise. The function f plays here the role of the hypothesis, and

should be selected in a well-chosen class – i.e., expressive enough but not too large.

Another standard supervised task is regression, where $\mathcal{Y}$ is continuous (typically $\mathcal{Y} = \mathbb{R}$), and one wants to learn a function f which predicts the quantity y from the observation $\mathbf{x}$. Many variants of the problem exist, but if we restrict f to the class of linear functions, and measure the error with the squared loss, we get the well known linear least squared problem[9].

[9] A regularization term on the vector $\mathbf{h}$ from Definition 1.4 is often added in practice in the optimization objective, but we focus for now solely on the terms related to the data.

Definition 1.4: Linear least squares aims at finding a vector $\mathbf{h} \in \mathbb{R}^d$ minimizing the risk induced by the loss

$$l_{\text{LLS}}(\mathbf{h}, (\mathbf{x}, y)) = (\mathbf{x}^T\mathbf{h} - y)^2. \tag{1.6}$$

We will come back to these different tasks in the following chapters, and in particular to the three unsupervised tasks. For now, we simply assume having a task defined by an explicit loss function, and consider solving it using a given dataset.

Risk minimization In most situations, the samples of $\mathbf{X}$ are assumed to be independent and identically distributed (i.i.d.) according to some distribution π. Solving a learning task defined by a risk function $\mathcal{R}$ hence amounts to find

$$h^* \in \arg\min_{h \in \mathcal{H}} \mathcal{R}(h, \pi). \tag{1.7}$$

However, the distribution π is unknown in practical applications and one cannot access directly $\mathcal{R}(\cdot, \pi)$. A more realistic goal is thus to learn from $\mathbf{X}$ an hypothesis $\hat{h}$ for which the excess risk

$$\Delta\mathcal{R}(\hat{h}, \pi) \triangleq \mathcal{R}(\hat{h}, \pi) - \mathcal{R}(h^*, \pi) \tag{1.8}$$

is small, as shown in Figure 1.2. A natural approach to do so is to directly solve

$$\min_{h \in \mathcal{H}} \mathcal{R}(h, \pi_{\mathbf{X}}), \tag{1.9}$$

which is known as empirical risk minimization. This generic formulation of the problem can naturally call for very different optimization tools. For instance, the risk minimizer for PCA has a closed form (the subspace spanned by the first eigenvectors of the covariance matrix), whereas minimizing the risk for k-means clustering is NP-hard, and iterative heuristics such as the k-means algorithm are widely used [14]. As a consequence, we do not focus for now on how to solve (1.9) in practice, but simply note that the quantity $\mathcal{R}(\cdot, \pi_{\mathbf{X}})$ can be exactly evaluated using the data samples.

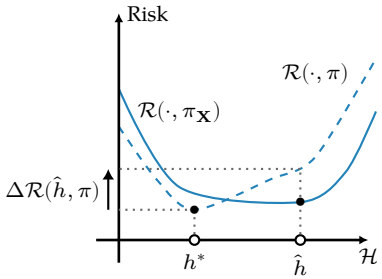


Figure 1.2: Schematic representation of the true and empirical risk functions. One ideally wants to find h^*, but can only access $\mathcal{R}(\cdot, \pi_{\mathbf{X}})$. Empirical risk minimization produces the hypothesis $\hat{h}$, whose excess risk is $\Delta\mathcal{R}(\hat{h}, \pi)$.

1.1.2 Challenges of modern data collections

As we have just seen in the previous section, any learning task formalized using a risk function implicitly defines an optimization problem, and can be addressed via empirical risk minimization (ERM). But even assuming that the problem is "nice" from an optimization perspective

(e.g. convex), the viability of this approach is often limited in practice by the nature of the data. We discuss here a few common characteristics of the datasets collected nowadays.

1. Most often, the number n of samples in the collection is very large[10]. This means that it takes time to go through the whole collection, but also that the dataset is unlikely to fit in core memory. When performing ERM, computing even a single gradient of the risk function requires going through the whole collection, and becomes expensive. Although this can naturally be mitigated using various techniques[11], one should still expect to load multiple times each sample in memory, and I/O operations can easily account for a substantial part of the overall learning time. Even assuming that sufficient memory is available, this makes any algorithm scaling more than linearly with respect to n practically useless.
2. In many applications, the dimension d of the data samples will also be large. This calls, similarly to the previous point, for algorithms with computational complexities that scale at most linearly with d. But it also changes drastically the geometric properties of the data and raises new questions; this is often referred to as the *curse of dimensionality*[12].
3. The collection might not be stored in one place, but rather distributed across multiple devices or data holders. Centralization might not be technically possible, or simply not desirable for various reasons, and algorithms need to adapt to this setting. In the extreme case, one can imagine applications where all the data samples are produced by different users, and must be processed locally.
4. Data might not be entirely known in advance, but take the form of a data stream. It should then be possible to process incoming data on the fly, and to update learned models gradually. Not any algorithm can be modified for this purpose, and it is often easier to come with new dedicated methods. This is often achieved via intermediate representation such as linear sketches (cf. Chapter 3) which are designed precisely to support sequential updates.
5. The data might be produced in a different location than where it is used. As a consequence, the network traffic grows together with the amount of collected data, which requires appropriate infrastructures and increases energy consumption. Compression algorithms can naturally be used to alleviate this problem, but why not designing efficient sensing methods, i.e. methods able to directly capture only the relevant parts or statistics of the data?
6. We mentioned the energy consumption induced by network infrastructures, but this is a larger problem which also applies to storage and algorithms. Datacenters, servers and supercomputers are known to require huge amounts of energy[13], and the global worldwide consumption has only been growing with the advent of cloud computing. Hence building large-scale efficient systems has become a crucial challenge [18]. Reducing the amount of stored

[10] For instance, Google translate uses a training collection comprising more than 10^{18} samples [15]. In terms of volume, many companies report processing more than 100 petabytes of data per day, although most of this information is never stored. The large hadron collider (LHC) collects (and stores permanently), after filtering, in average one petabyte (10^{15} bytes) of data per day [16].

[11] e.g. stochastic or block gradient methods.

[12] In particular, volume grows exponentially with the dimension, and the number of data points (sample complexity) required for accurately learning standard models also grows exponentially.

[13] It is not easy to give accurate estimations here, especially given that the increasing costs are balanced by significant efficiency gains, but multiple sources claim that data centers account for 1% of electricity consumption worldwide. See for instance [17] for up-to-date perspectives.

data by directly compressing it at acquisition time, while keeping its important characteristics for downstream learning applications, can thus be considered as a way to minimize the problem. But designing learning algorithms with this concern in mind is also necessary; although energy consumption is most often directly correlated with the computational complexity, the connection might at times be more tricky.

7. Sometimes, the data samples are considered to be of a sensitive nature. This does not mean that learning is not possible or desirable, but suggests that alternative algorithms must be designed with this concern in mind. A good example would be the medical domain, where combining data coming from many different patients could be valuable to learn accurate global models which could benefit to all; however, this should for obvious reasons be done without revealing individual medical records, and simple anonymization techniques are known to be insufficient for this matter [19]. This also suggests, from a user perspective, that the data might not be publicly available, but only accessible via a restricted interface, or in a different form, e.g. as rough statistics that do not reveal much about the individual samples.
8. Datasets might contain missing or corrupted data. We will not focus on this problem in this thesis, but this is something to take into account. This can be due to the failure of sensing devices, but also applies to labels in the context of supervised learning: for instance in computer vision, collecting large amounts of images became straightforward with the proliferation of compact cameras and smartphones, but labels are produced by human beings, which is much more time-consuming[14] and raises many technical as well as ethical questions. Developing semi-supervised methods (i.e. using partially annotated collections) or unsupervised methods able to produce themselves pseudo-labels has also become a crucial challenge.
9. Finally, data might not come in a numerical form as we assumed it in Section 1.1.1, but rather be categorical or structured. This is for instance the case of data measured across networks, where the geometry defined by the network's edges is often a valuable information which can be used jointly with the samples to improve learning performance. This calls for dedicated algorithms; this will not be discussed in this thesis, and we assume when necessary that structure can be leveraged into a preprocessing step to produce numerical data. But extending the methods presented in this thesis to integrate directly structural information would naturally be interesting[15].

[14] It thus comes at no surprise that mechanisms such as reCAPTCHA were introduced: by asking the user to assign categories to images in order to detect bots, this program also records the answers of valid users, which can be used later as categorization labels. This program was displayed 100 million times per day at the beginning of the last decade [20], before being replaced more recently by less intrusive mechanisms.

[15] In particular, one might benefit from any extension of the kernel methods presented in Section 2.3 to other domains.

Trade-offs In light of these constraints, it becomes clear that standard learning methods, such as for instance the generic ERM approach, cannot simply be adapted to fulfill all these requirements. One must come with tailored learning algorithms and frameworks, and integrate

the considerations discussed above into the design process.

These methods should naturally be theoretically grounded. In the case of statistical learning, it means that bounds on the excess risk[16] $\Delta\mathcal{R}(\hat{h}, \pi)$ associated to the hypothesis $\hat{h}$ learned by the algorithm should be provided. Note however that satisfying the above concerns will most often come at the cost of reduced performance, i.e. weaker and possibly probabilistic bounds.

[16] cf. (1.8).

This connects with the notion of probably approximately correct (PAC) learning [21, Chapter 3]. An hypothesis class $\mathcal{H}$ is said to be agnostic PAC learnable if there exists an algorithm which, for any distribution π on $\mathcal{X}$, any $\varepsilon > 0, \delta \in]0, 1[$ and any dataset $\mathbf{X}$ made of at least $n(\varepsilon, \delta)$ samples drawn i.i.d. from π, returns an hypothesis $\hat{h}$ whose excess risk satisfies $\Delta\mathcal{R}(\hat{h}, \pi) \leq \varepsilon$. We use here the notation $n(\varepsilon, \delta)$ to denote a function of ε, δ, which gives the smallest number of samples required for ε-approximate learning with probability $1 - \delta$; this is called the sample complexity. The results which appear later are not explicitly casted into this PAC framework, but their nature is sometimes very close: one always wants to control the excess risk with high probability using the smallest number of samples as possible. In our setting, the size of the intermediate compressed representation will often play a role as well, and guarantees will depend on it.

In this thesis, we focus on a method called compressive learning, which can cope at least with points 1-4 of the above list, and certainly help with other considerations (depending on the practical setting). The next section provides a rough overview of this method and why it is interesting in this regard. We provide in Chapter 3 a comprehensive overview of alternative approaches from the literature which can cope as well with the problems stated earlier.

1.2 The compressive learning approach

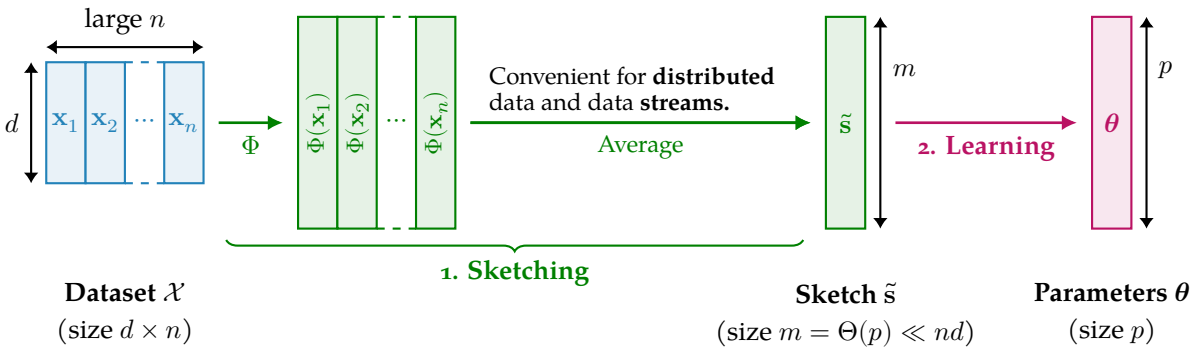


Figure 1.3: General Framework. The dataset is sketched into a single vector of generalized moments, from which the parameters of interests are then estimated.

In the compressive learning framework, which is depicted in Figure 1.3, the dataset is compressed into a single vector, called the sketch[17] of the data. The sketch $\tilde{\mathbf{s}}$ of a dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$ is a vector, and is simply defined as

$$\tilde{\mathbf{s}} = \frac{1}{n} \sum_{i=1}^{n} \Phi(\mathbf{x}_i), \tag{1.10}$$

[17] The term "sketch" has multiple meanings, which are further addressed in Chapter 3.

where $\Phi : \mathcal{X} \to \mathcal{Z}$ is a wisely chosen feature map, which is typically nonlinear. In the following, we will most often consider $\mathcal{Z} = \mathbb{R}^m$ or $\mathbb{C}^m$ for some sketch size m.

The empirical sketch $\tilde{\mathbf{s}}$ is thus just the collection of the generalized[18] empirical moments induced by Φ. As will be explained later in Chapter 2, we will consider in particular feature maps Φ of the form $\Phi(\mathbf{x}) = \rho(\mathbf{\Omega}^T\mathbf{x})$, where $\mathbf{\Omega} \in \mathbb{R}^{d\times m}$ is a randomly-drawn matrix and $\rho : \mathbb{R} \to \mathbb{R}$ a nonlinear function applied pointwise. We will see that in particular clustering and Gaussian modeling tasks can be performed using $\rho : x \mapsto \exp(\imath x)$ (in the whole document, $\imath$ denotes the imaginary number), and principal component analysis using the square function $\rho : x \mapsto x^2$.

[18] By opposition to the "standard" i-th order moments.

Advantages Although we postpone to Chapter 2 the details and intuition behind this design of the sketching mechanism, one can already see the computational benefits of working with such sketches.

Indeed, not only the feature map Φ has a simple expression and can be evaluated efficiently, but most importantly the average operation in Equation (1.10) makes the computation embarrassingly parallelizable[19] as shown in Figure 1.4. When the data is split across multiple data holders, each of them can sketch the local collection and only share the resulting sketch – as well as the number of samples used to compute it, so that correct weights can be computed in subsequent averaging operations. When the dataset is located at one single location, the sketching process can still be performed by batches on different cores in order to speed up the process. Streams can easily be sketched "on the fly" as well, simply by sketching the data samples one by one and averaging the sketches progressively. In the following, we also refer to this method as "online sketching".

[19] Samples can be sketched all independently in parallel or by batches. In a scenario where the data is already distributed (no data transfer cost), one should expect the speedup to be linear with the number of cores, hence beating Amdahl's law.

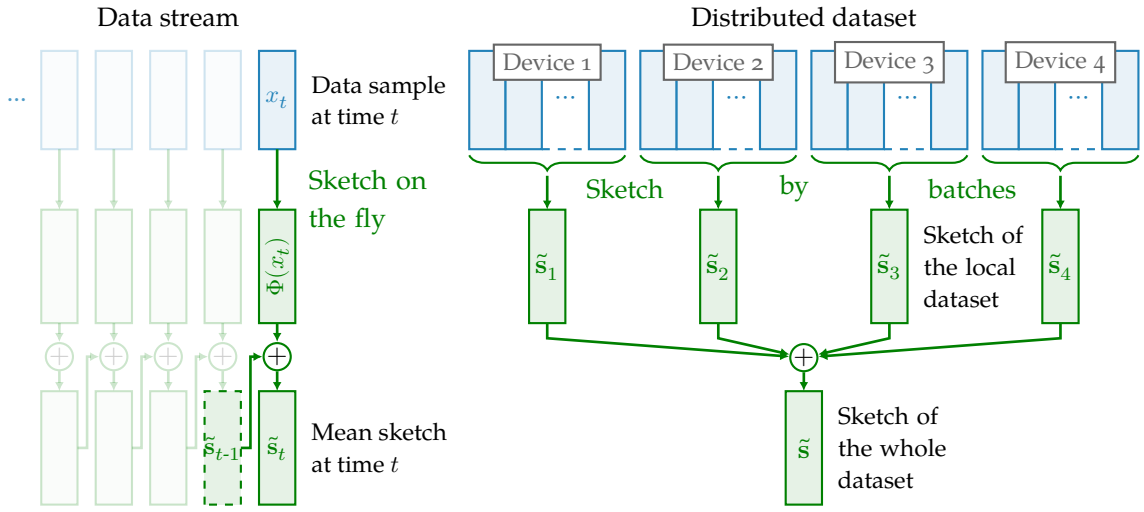


Figure 1.4: Left: A streaming scenario, where the data samples are sketched one by one, and the mean sketch is continuously updated. Right: A distributed scenario, where each device computes a local sketch, and a centralized entity further averages these local sketches.

But computational efficiency should not be the only reason to consider compressive approaches. Sketching drastically reduces the amount of information and thus is a natural candidate for privacy-aware learning. We will show in Part IV that guarantees can indeed be obtained

in this direction. Another interest of this compressive approach is its generic nature. The choice of the function ρ depends on the nature of the task to solve, but different learning tasks can still be tackled under the same formalism, and the algorithms used to learn from the sketch can be adapted for various problems.

Open Challenges Naturally, this approach has some limits. Beyond the successful examples mentioned above, it remains a challenge to find which tasks can or cannot be approximately solved with such a compressive approach. Adapting the mechanism to supervised learning tasks is also not straightforward. More details will be provided in Chapter 2, and some considerations for extension to broader families of learning tasks are discussed in Chapter 10.

Related approaches Although moments have been used in statistics for similar use cases, the compressive learning framework is rather rooted in signal processing, and especially relies on compressive sensing techniques and kernel methods (see Chapter 2). Many other approaches for learning with reduced resources exist. They often bear similarities with compressive learning in the common tools they rely on, such as the multiplication by random matrices, but these ideas can be used in many different ways, and analyzed with different theoretical tools. We discuss related ideas in Chapter 3, such as dimensionality reduction techniques and coresets.

Seen as the succession of random linear operation and a pointwise non-linearity, compressive learning can also be analyzed as the first layer of a random neural network. The main difference lies in the averaging operation; although such operations are sometimes performed in neural networks[20], they never apply to all the features at once. Note that shallow random networks are known to capture useful information for classification [22], and some random convolutional networks have more recently been observed to be invertible [23], so it should come at no surprise that the sketch defined at (1.10) can capture enough information to solve specific tasks when Φ is wisely chosen.

[20] Usually referred to as "pooling" operations.

1.3 Contributions

We summarize our contributions, and propose below a list of the publications which are directly related to the thesis.

1.3.1 Layout and summary of contributions

The rest of the thesis is structured in four distinct parts. The fist part mainly presents existing works, while the three other contain the contributions of the thesis. These contributions extend the compressive learning framework in different directions. We propose here a summary of these contributions, which follows the layout of the manuscript.

Part I reviews existing approaches for large-scale learning.

- Chapter 2 provides an introduction to compressive learning. It explains how the whole framework is grounded in the field of compressive sensing, and how the design of the sketching operator is related to kernel methods. This chapter also details the different learning tasks which have been addressed using compressive learning.
- Chapter 3 discusses other randomized techniques for large-scale learning, such as random dimensionality-reduction methods or coresets. This chapter does not aim at being exhaustive, as it encompasses a very large part of the literature, but rather tries to present the most common ways to leverage random projection and subsampling techniques.

Part II suggests two directions for improving the distribution from which the columns of the random matrix $\boldsymbol{\Omega}$ are drawn. In the following, it will be useful to decouple[21] the radial and directional distributions of these columns.

[21] We will only consider distributions for which this separability holds throughout the manuscript.

- Chapter 4 focuses on the particular setting of compressive clustering, where $\rho = \exp(\imath\cdot)$, and highlights the importance of choosing well the scale of the columns of $\boldsymbol{\Omega}$ – which can be interpreted as frequency vectors in this setting. The contributions of this chapter are empirical, and provide new insights on the connection between the "optimal" scale[22] of the frequencies and the separation (i.e. the minimum distance) between the clusters to retrieve.
- Chapter 5 assumes on the opposite that an appropriate radial distribution is known, and focuses on the directional distribution. In particular, it considers drawing the matrix $\boldsymbol{\Omega}$ by stacking structured blocks with the aim of reducing the overall computational complexity of the framework. We show empirically that the chosen construction does not degrade the learning performance, while indeed allowing significant time and memory savings. We also discuss how existing learning guarantees can be adapted to this new setting for compressive clustering.

[22] i.e. the scale providing empirically the lowest clustering error (assuming it is unique).

Part III consists of a single chapter, and solely focuses on the k-means clustering task, where one wants to recover k cluster centers. Solving the clustering problem with a compressive approach has already been proposed in the literature, using a sketch of random Fourier features and an algorithm inspired from generalized orthogonal matching pursuit to recover the cluster centers from the sketch.

- Chapter 6 introduces a new algorithm to address this second task, while still relying on a sketch of random Fourier features. The method is based on the SHyGAMP algorithm, which itself belongs to the family of approximate message passing methods. We provide a broad introduction to loopy belief propagation, and detail how our compressive learning task can be casted into a standard Bayesian formalism under the assumption of a Gaussian mixture generation model. We detail how the dependence of the entries

of the sketch in the cluster centers allows us to use approximate message passing techniques, and explain why further approximations are required. We discuss numerical issues arising when implementing the method, and provide an experimental validation on both synthetic and real data. A method is also proposed to tune the model hyperparameters. This chapter is the result of a collaboration; my personal contribution mainly consisted in experimental aspects of the work, and especially in exploring various approximation strategies to estimate posterior quantities required in the algorithm.

Part IV explores the potential of compressive learning for applications where privacy preservation is required.

- Chapter 7 provides a broad introduction to privacy-preserving machine learning, and in particular to the differential privacy formalism, that we use to define and quantify the level of privacy of an algorithm. It also reviews standard approaches for large-scale privacy-aware learning.
- Chapter 8 shows how a slight perturbation of the sketching mechanism is sufficient to obtain formal differential privacy guarantees. We provide results for the problems of clustering, Gaussian modeling and PCA with sharp privacy bounds. A subsampling mechanism is also introduced to reduce the computational complexity of the sketching operation.
- Chapter 9 suggests that the utility (for subsequent learning) is closely related to a signal-to-noise ratio, and uses this criterion to optimize miscellaneous parameters of the framework. Experimental results are also provided for the different learning tasks considered, and show that our method is competitive with state-of-the-art approaches.

This part is also the fruit of a collaboration, and my personal contribution consisted mainly in establishing the sharpness of the different bounds, and running experiments for the clustering and PCA applications.

1.3.2 List of publications

Here is the list of the publications related to the thesis. Some of these works are still under review.

- Efficient compressive learning (Part II)
  - Antoine Chatalic, Rémi Gribonval, and Nicolas Keriven. "Large-Scale High-Dimensional Clustering with Fast Sketching." In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018
  - Antoine Chatalic and Rémi Gribonval. "Learning to Sketch for Compressive Clustering." In: International Traveling Workshop on Interactions between Low-Complexity Data Models and Sensing Techniques (iTWIST). June 2020

- Message-passing (Part III)
  - Evan Byrne, Antoine Chatalic, Rémi Gribonval, and Philip Schniter. "Sketched Clustering via Hybrid Approximate Message Passing." In: *IEEE Transactions on Signal Processing* 67.17 (Sept. 2019)
- Privacy-preserving compressive learning (Part IV)
  - Antoine Chatalic, Vincent Schellekens, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques, and Rémi Gribonval. "Compressive Learning with Privacy Guarantees." Submitted to Information and Inference (under review). Submitted to Information and Inference (under review). Mar. 3, 2020
  - Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques, and Rémi Gribonval. "Compressive K-Means with Differential Privacy." In: SPARS Workshop. July 1, 2019
  - Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Yves-Alexandre De Montjoye, Laurent Jacques, and Rémi Gribonval. "Differentially Private Compressive K-Means." In: *44th International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*). Brighton, United Kingdom, May 2019
- Other contributions
  - Antoine Chatalic, Nicolas Keriven, and Rémi Gribonval. "Projections aléatoires pour l'apprentissage compressif." In: Gretsi. Aug. 26, 2019
  - Rémi Gribonval, Antoine Chatalic, Nicolas Keriven, Vincent Schellekens, Laurent Jacques, and Philip Schniter. "Sketching Datasets for Large-Scale Learning (Long Version)." Aug. 4, 2020 (under review)

Part I

Existing tools for Large-Scale Learning

This part reviews the different existing approaches for efficient large-scale learning. The compressive learning framework is presented in details in Chapter 2, and we provide in Chapter 3 an overview of other common tools used in the domain.

Chapter 2
A Guided Tour of Compressive Learning

This chapter provides an overview of the ideas, algorithms and tools used in compressive learning. It shows how the connections with the setting of compressive sensing, established in Section 2.1, can be leveraged to get generic bounds on the learning error (Section 2.2). This requires some assumptions on the sketching operator, which have been proved for several applications using tools from kernel methods, as suggested in Section 2.3. Reconstruction algorithms are discussed in Section 2.4, and a general summary of the different tasks and guarantees is provided in Section 2.5.

Contents

2.1 A learning framework rooted in compressive sensing

A fundamental result in signal processing is the Shannon[1] sampling theorem [32], which states that a bandlimited signal having no frequency higher than f_m hertz can be sampled regularly[2] without loss if the sampling frequency f_s satisfies $f_s > 2f_m$, i.e. is at least twice the highest frequency contained in the signal. This theorem however only provides a sufficient condition, and holds uniformly for all bandlimited signals. Many lossless compression schemes however exist, allowing to reduce further after acquisition the number of samples. Although such schemes usually rely on nonlinear transformations, they make clear that many signals are somehow much more "compressible" than what could be expected from the Shannon theorem.

Compressive sensing[3] is a field of signal processing which emerged in the early 2000s around the idea that sparse signals, i.e. signals that can be described in a domain using only a few nonzero coefficients, can be sampled without loss using a much smaller number of measurements that what the Shannon theorem would require. The sampling scheme considered is not regular anymore, but still linear, and the number of samples required is mainly related to the sparsity of the signal.

We only provide in this part a brief introduction to the domain, and refer the interested reader to the book *A Mathematical Introduction to Compressive Sensing* [33] for a comprehensive summary, or to the paper

[1] The theorem often takes the name of Shannon for his comprehensive paper [32] providing a concise proof, but the result was according to his own words "common knowledge in the communication art" and similar formulations had been earlier and independently proposed by Nyquist, Whittaker and Kotelnikov.

[2] i.e. using equally-spaced samples

[3] Also called compressed sensing, and sometimes compressive sampling

"A Survey of Compressed Sensing" [34] for a more concise overview of classical results.

2.1.1 Compressive sampling strategies

We introduce the notation $\|\mathbf{x}\|_0$ to denote the number of nonzero entries of a vector $\mathbf{x}$. We define the support of a vector $\mathbf{x} \in \mathbb{C}^d$ as

$$\text{supp}(\mathbf{x}) = \{i \in [\![1, d]\!] \mid x_i \neq 0\}, \text{ where } [\![1, d]\!] \triangleq \{1, ..., d\},$$

so that $\|\mathbf{x}\|_0 = |\text{supp}(\mathbf{x})|$. Note that although $\|\cdot\|_0$ is often referred to as the l_0-norm, it is actually neither a norm[4], nor a quasi-norm, nor a semi-norm. We refer to it as a pseudo-norm, but without any specific mathematical meaning behind this denomination. We say that a vector $\mathbf{x} \in \mathbb{C}^d$ is s-sparse when $\|\mathbf{x}\|_0 \leq s$, i.e. when $\mathbf{x}$ has at most s nonzero entries. We also say more qualitatively that $\mathbf{x}$ is sparse when $\|\mathbf{x}\|_0 \ll d$, and define Σ_s as the set of of all s-sparse signals. For any $p > 0$ and $\mathbf{x}$, we use the notation

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p}, \tag{2.1}$$

and recall that $\|\cdot\|_p$ is a proper norm (called the l_p norm) whenever $p \geq 1$.

In the standard compressive sensing setting, one has access to linear observations $\mathbf{y} \in \mathbb{C}^m$ of a signal of interest $\mathbf{x} \in \mathbb{C}^d$, obtained through a linear measurement operator $\mathbf{A}$ and possibly corrupted by noise $\mathbf{e}$:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}. \tag{2.2}$$

One would like to recover the original signal $\mathbf{x}$ from its observations, which is often referred to as an inverse problem. The number of observations is typically smaller than the ambient dimension, i.e. $m < d$, so that even in the noiseless scenario ($\mathbf{e} = 0$), the system (2.2) is undetermined and has possibly an infinite number of solutions.

Compressive sensing revolves around the idea that, if $\mathbf{x}$ is known to be sparse[5] as represented in Figure 2.1, then it is possible to design a measurement operator $\mathbf{A}$ such that $\mathbf{x}$ can be recovered from its measurements and, more generally, such that uniform recovery of all signals with a given sparsity level is possible.[6]

If a measurement operator $\mathbf{A}$ allows identifiability of all s-sparse signals, in the sense that any $\mathbf{x} \in \Sigma_s$ is the only solution to the problem $\mathbf{A}\mathbf{z} = y$ over Σ_s, then one can theoretically recover in the noiseless case ($\mathbf{e} = 0$) any $\mathbf{x} \in \Sigma_s$ by solving

$$\min_{\mathbf{z} \in \mathbb{C}^d} \|\mathbf{z}\|_0 \text{ subject to } \mathbf{A}\mathbf{z} = \mathbf{y}. \tag{2.3}$$

However, besides being a nonconvex optimization problem, Equation (2.3) is well-known to be NP-hard [33, p.55], and is therefore of limited practical utility.

The key challenge of compressive sensing is thus to design not only sensing matrices showing adequate properties while keeping the number m of observations as small as possible, but also practical recovery

[4] It does not satisfy the homogeneity property.

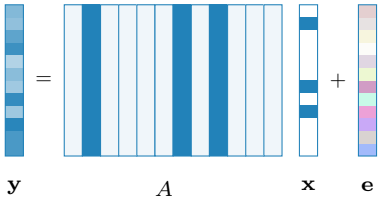


Figure 2.1: Measurement process: when $\mathbf{x}$ is a sparse vector, $\mathbf{y}$ is close (up to noise $\mathbf{e}$) to a linear combination of a few columns of $\mathbf{A}$. For this example, the non-zero entries of $\mathbf{x}$ and corresponding columns of $\mathbf{A}$ are filled with dark blue.

[5] We consider here signals which are sparse in he canonical basis for simplicity, but sparsity in another basis could naturally be used.

[6] Recovery of one fixed vector has been studied and nonuniform guarantees exist, but we do not cover this setting here.

algorithms with provable guarantees when used with suitable measurement matrices. The seminal papers in compressive sensing come from Candès, Romberg and Tao [35], and Donoho [36], who combined two main ideas to answer these questions. Firstly, it is possible to generate randomly measurement matrices that will be adequate with high probability; then, l_1-norm minimization can be used in place of (2.3), yielding a convex optimization problem known as basis pursuit.

Note that, in practical applications, observations are usually corrupted by noise ($\mathbf{e} > 0$ in (2.3)). Moreover, the signal of interest $\mathbf{x}$ might not be exactly sparse, but only compressible, i.e. close to the set of sparse signals. It is thus fundamental to derive guarantees that are robust to noise, and stable with respect to compressible signals.

We first review the most common recovery algorithms, and then detail in Section 2.1.3 under which conditions on the measurement matrices recovery guarantees can be obtained for these algorithms.

2.1.2 Algorithms for the inverse problem

Note that when $\mathbf{e} = 0$ and $\mathbf{x}$ is sparse, $\mathbf{y}$ is a linear combination of a few columns of $\mathbf{A}$. Recovering $\mathbf{x}$ from $\mathbf{y}$ thus amounts to choosing the few columns of $\mathbf{A}$ which best express the signal $\mathbf{x}$. As a consequence, we will sometimes refer in the following to $\mathbf{A}$ as the dictionary, and to its columns as the atoms of the dictionary.

We mentioned just earlier that using l_1-norm minimization for recovery had been one of the key ideas in the early days of compressive sensing. Many other algorithms have been since developed. They can be mainly classified into three different groups : convex approaches, which include l_1-minimization, greedy approaches, and thresholding-based techniques.

Convex relaxations Basis pursuit, which consists in replacing $\|\cdot\|_0$ by the $\|\cdot\|_1$ norm in (2.3), was one of the first approaches considered to solve the inverse problem. Using the l_1 norm induces sparse solutions similarly to the original problem, as suggested by Figure 2.2, but has the huge advantage of inducing a convex optimization problem. In applications where corruption by additive noise should be taken into account, a straightforward generalization is basis pursuit denoising:

$$\min_{\mathbf{z}\in\mathbb{C}^d} \tfrac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{z}\|_1, \tag{2.4}$$

where $\lambda > 0$ is a regularization parameter controlling the desired tradeoff between sparsity and fidelity to the measurements. There are other closely related formulations of the problem though, such as quadratically-constrained l_1-minimization, or the LASSO[7] [37] in the statistical community. Many different algorithms exist to solve these problems; the interested reader can refer for instance to [33, ch.15] for a first overview.

Greedy methods We regroup under this name the methods which start from the sparsest possible vector $\mathbf{z} = 0$ and iteratively increase

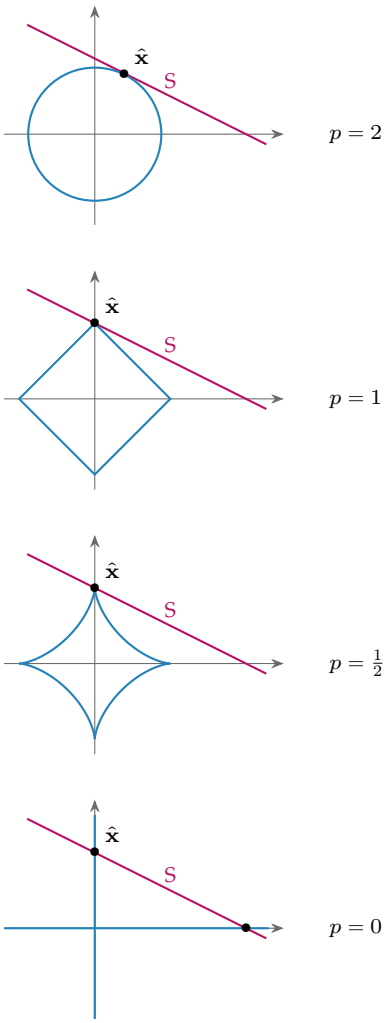


Figure 2.2: Standard l_p (pseudo-)norms in dimension 2. Here S represents the linear constraint induced by the observations. The blue curves show for each $p > 0$ the first isoline of the l_p (pseudo-)norm which intersects with S, giving the solution $\hat{\mathbf{x}}$ of smallest l_p norm which satisfies the linear constraint. For $p = 0$, the blue set is Σ_1. All l_p pseudo-norms for $p \leq 1$ somehow induce sparsity, but only l_1 yields a convex optimization problem.

[7] LASSO stands for "least absolute shrinkage and selection operator".

the support of $\mathbf{z}$ using greedy selection rules to better fit observations. Algorithms such as matching pursuit [38] and its generalization orthogonal matching pursuit [39], which had initially been designed for sparse approximation in various dictionaries, belong to this category and can be used for reconstruction from random measurements [40]. In this case, the support of the solution is increased by one index at each iteration. If $\mathbf{z}^{(k)}$ denotes the current approximation at iteration k, and assuming that $\mathbf{A}$ has normalized columns, the next index to be added to the support is $\arg\min_{1\leq i\leq d} |\mathbf{A}^*(\mathbf{y} - \mathbf{A}\mathbf{z}^{(k)})|$ (here $\mathbf{A}^*$ denotes the Hermitian transpose), i.e. the one corresponding to the column of $\mathbf{A}$ which has the highest correlation with the residual $\mathbf{y} - \mathbf{A}\mathbf{z}^{(k)}$.

The support can only grow with this algorithm, which can sometimes be a burden when an error is made during the first iterations. One way to avoid such problems is to simply add extra steps in which previously selected atoms can be replaced [41]. The CoSaMP algorithm [42] is slightly different as it greedily selects multiple atoms at each iteration, but also allows to replace previously selected atoms using an extra thresholding step.

Thresholding-based techniques Iterative hard thresholding [43] is the most common method in this category. Hard thresholding, is the operation which consists in keeping only the largest entries of a vector. If H_s denotes such an operator keeping the s largest entries, then the algorithm is simply the succession of steps of the form

$$\mathbf{x}^{(k+1)} = H_s(\mathbf{x}^{(k)} + \mathbf{A}^*(\mathbf{y} - \mathbf{A}\mathbf{x}^{(k)})). \tag{2.5}$$

Note that the quantity $\mathbf{A}^*(\mathbf{y} - \mathbf{A}\mathbf{x})$ can be interpreted as the gradient of the cost function $\mathbf{x} \mapsto \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2$, and thus the algorithm seen as a succession of gradient descent and thresholding steps.

We now give an idea of standard properties used to provide theoretical guarantees for these different families of reconstruction algorithms.

2.1.3 Measurement operators design

Multiple complementary properties have been introduced in order to characterize "good" sensing matrices. In practice, exact recovery of s-sparse vectors via (2.3) is possible using $m = 2s$ deterministic measurements [33, Theorem 2.14]. However, l_0-minimization is not practical as discussed above, and such guarantees are not robust to noise at all.

Basis pursuit[8] can be shown to successfully recover all sparse vectors if and only if the measurement operator satisfies a specific condition, known as the null-space property; a variant can be derived for robust reconstruction. Checking if a given matrix meets one of these properties is however not straightforward, de facto limiting their usability.

If $(\mathbf{a}_i)_{1\leq i\leq d}$ denote the columns of $\mathbf{A}$, and if we assume that $\mathbf{A}$ is normalized such that $\|\mathbf{a}_i\|_2 = 1$ for each $i \in [\![1, d]\!]$, then the coherence of $\mathbf{A}$, defined as $\mu(\mathbf{A}) \triangleq \max_{i\neq j} |\langle \mathbf{a}_i, \mathbf{a}_j \rangle|$, can also be used to state sufficient recovery conditions for various algorithms, while being easily

[8] i.e. recovery using the l_1-norm minimization.

computable. However such conditions can only be satisfied using a number of measurements m scaling quadratically with the sparsity, which is pessimistic as will be explained below.

The restricted isometry property (RIP) is a stronger[9] but more convenient tool to get around these issues. It has been introduced by Candès and Tao [44, 45].

[9] It implies in particular the null-space property.

Definition 2.1 (Restricted isometry property): A sensing matrix $\mathbf{A}$ satisfies the restricted isometry property on a set S with constant ε if

$$\forall \mathbf{x} \in S,\ (1-\varepsilon)\|\mathbf{x}\|_2^2 \leq \|\mathbf{A}\mathbf{x}\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}\|_2^2. \qquad (2.6)$$

In practice, it will be interesting to establish this property for sets of the form $S = \Sigma_s - \Sigma_s = \Sigma_{2s}$ as represented in Figure 2.3 – here and in the following, we denote $S_1 - S_2 \triangleq \{x - y \,|\, x \in S_1, y \in S_2\}$ for any two sets S_1, S_2. One can then think of an operator which satisfies the RIP on $\Sigma_s - \Sigma_s$ as an operator which preserves the geometry of Σ_s.

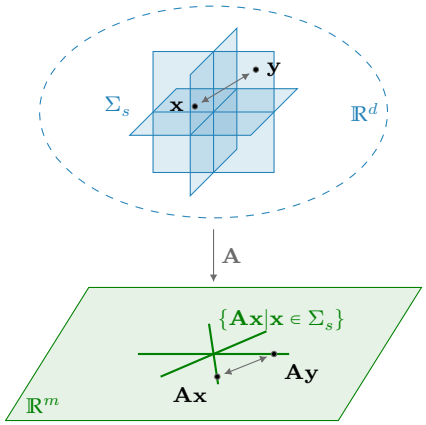


Figure 2.3: Restricted isometry property on $\Sigma_s - \Sigma_s$: the distances between the points in Σ_s are approximately preserved by $\mathbf{A}$.

When the set S indeed takes the form $S = \Sigma_s$ for some s, we denote ε_s the associated constant. We say qualitatively that $\mathbf{A}$ "satisfies the restricted isometry property" when (2.6) holds on Σ_s for a large enough s with a small constant ε_s. The RIP is a sufficient condition for successful uniform recovery of sparse vectors. Many guarantees have been proposed in the literature, for the different algorithms discussed in Section 2.1.2, based on conditions of the type $\varepsilon_{ks} < t$, where k is a small integer and t some threshold lower than 1. Results for convex approaches were for instance obtained first, both in the noiseless [45] and robust settings [46]. Many papers successively improved these bounds.

Role of randomness One of the reasons that contributed to the success of the RIP is that some random families of matrices can be shown to satisfy it with high probability, and using a number of measurements m smaller than what would be necessary to enforce for instance coherence-based conditions. Indeed, one can show that if $\mathbf{A}$ is a random $m \times d$ matrix with independent mean-zero sub-Gaussian entries[10] with variance 1, then there is a constant C such that for any $\varepsilon > 0$, $\frac{1}{\sqrt{m}}\mathbf{A}$ satisfies the RIP on Σ_s with a constant smaller than ε with probability $1 - 2\exp(-\varepsilon^2 m/(2C))$ provided that $m \geq 2C\varepsilon^{-2} s \log(ed/s)$ [33, Theorem 9.2]. Here e simply denotes the constant $e = \exp(1)$. This result is close to the paper of Baraniuk et al. [47], which itself relies on concentration inequalities which are similar to the ones used to prove the Johnson-Lindenstrauss lemma. The latter will only be introduced in the next chapter, but bears strong similarity with the restricted isometry property. Common examples of matrices satisfying the above condition are Gaussian matrices, which have i.i.d. $\mathcal{N}(0,1)$ entries, and Bernoulli matrices which have i.i.d. Rademacher-distributed entries, i.e. taking values ± 1 with probability $\frac{1}{2}$. Constructing deterministic matrices satisfying the RIP is actually a more challenging problem.

[10] A random variable X is called subgaussian if there exists $C, \alpha > 0$ such that for any $t > 0$, $\mathbb{P}[|X| > t] \leq Ce^{-\alpha t^2}$.

Note that the recovery guarantees obtained with the RIP when

sensing using random matrices are obtained w.h.p. for a number of measurements $m \gtrsim s \log(d/s)$ (where $\gtrsim$ means "up to a controlled multiplicative factor"), whereas coherence-based guarantees require $m \gtrsim s^2$. When the RIP holds, then robust (to noise) and stable (to approximately sparse signals) recovery guarantees exist for various reconstruction algorithms as mentioned just above.

We now explain how the key ideas of compressive sensing can be generalized to other models.

2.1.4 Generalization to low-rank matrix recovery

The tools presented above have been generalized to many other classes of signals than sparse vectors. We will explain in Section 2.1.5 how compressive learning can itself be interpreted as one of these extensions, but we first present here the problem of low-rank matrix recovery. It has many practical applications such as matrix completion (e.g. for recommender systems) [48] or phase retrieval [49], but is also a key building block for compressive PCA[11], which motivates this overview.

[11] The connection will be made clear later in Section 2.2.2.

The signal we consider is now a matrix $\mathbf{X} \in \mathbb{C}^{d_1 \times d_2}$, which is observed via a linear operator $\mathcal{M} : \mathbb{C}^{d_1 \times d_2} \to \mathbb{C}^m$, i.e. one has access only to the measurements

$$\mathbf{y} = \mathcal{M}(\mathbf{X}) \in \mathbb{C}^m. \tag{2.7}$$

The task consists here again in recovering $\mathbf{X}$ from its measurements. However, the problem will most often only be interesting when we consider a small value for m, i.e. when the problem is underdetermined. Similarly to the case of vectors recovery, additional assumptions are needed for unambiguous recovery. A natural (and meaningful in terms of applications) model to consider is the one of low-rank matrices, i.e. of matrices whose vector of singular values is sparse. We thus formulate the problem as

$$\min_{\mathbf{Z}} \operatorname{rank}(\mathbf{Z}) \text{ subject to } \mathcal{A}(\mathbf{Z}) = \mathbf{y}. \tag{2.8}$$

This problem, known as affine rank minimization, is non-convex and NP-hard[12]. However approximate solutions can be found by minimizing the nuclear norm of $\mathbf{Z}$ instead of its rank [50], which is a convex optimization problem. Note that the nuclear norm of $\mathbf{Z}$ is simply the l_1 norm of its vector of singular values, which explains the connection with the vectorial setting. Other classical approaches presented in Section 2.1.2 for vector recovery also have natural extensions, such as AdMIRA [51] for greedy methods (inspired from CoSaMP), and singular value projection [52] or thresholding [53].

[12] The problem (2.3) (for which a reference was provided regarding its NP-hardness), can be reduced to the subproblem of affine rank minimization where $\mathbf{Z}$ is assumed to be diagonal.

Because of the strong connection between the vector and matrix settings, it comes as no surprise that for any rank r and $m \gtrsim r \max(d_1, d_2)$ (up to a constant), one can design random operators $\mathcal{M} : \mathbb{C}^{d_1 \times d_2} \to \mathbb{C}^m$ which will satisfy a restricted isometry property[13] on the set of rank r matrices with high probability. This bound on m is optimal.

[13] i.e. a property similar to (2.6), but for matrices, e.g. with respect to Frobenius and l_2 norms.

Compressive sensing has been extended to many other models, such as block-sparse [54] or cosparse [55] vectors, and signals living

in union of subspaces [56] or low-dimensional manifolds [57]. More recently, generative models have successfully been used in place of the hand-crafted ones [58], opening the door for nonlinear measurement operators that are themselves learned using deep networks [59]. We of course cannot provide an extensive overview of all these developments here, but simply remark that these successive steps led to generic results which can potentially hold for any abstract model of low dimensionality [60]; compressive learning is a straight continuation of these works.

2.1.5 Sketches as linear distribution embeddings

We now make clear the connection between the sketching process introduced in Section 1.2 and compressive sensing. Recall that we defined in (1.10) the empirical sketch associated to a dataset $\mathbf{X}$ with empirical distribution $\pi_{\mathbf{X}}$ as

$$\tilde{\mathbf{s}} = \frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i). \tag{2.9}$$

Let $\mathcal{P}(\mathcal{X})$ denote the set of probability distributions over $\mathcal{X}$ and, for a given feature map Φ, let $\mathcal{D}_\Phi \subseteq \mathcal{P}(\mathcal{X})$ denote the set of probability distributions over $\mathcal{X}$ with respect to which Φ is integrable. We define the operator $\mathcal{A} : \mathcal{D}_\Phi \to \mathcal{Z}$ as

$$\mathcal{A}(\pi) \triangleq \mathbf{E}_{\mathbf{x}\sim\pi}\Phi(\mathbf{x}), \tag{2.10}$$

which implicitly depends on the chosen feature map Φ. The definition extends naturally[14] to finite signed measures. With this definition, the empirical sketch of $\mathbf{X}$ can simply be written $\tilde{\mathbf{s}} = \mathcal{A}(\pi_{\mathbf{X}})$. If the $(\mathbf{x}_i)_{1\leq i\leq n}$ are drawn i.i.d. according to π, then when n is large the quantity $\mathbf{e} \triangleq \mathcal{A}(\pi_{\mathbf{X}} - \pi)$ should be small[15]. It is thus meaningful to interpret it as noise, so that the sketch can be rewritten

$$\tilde{\mathbf{s}} = \mathcal{A}(\pi) + \mathbf{e}, \tag{2.11}$$

i.e. the empirical sketch is simply a noisy observation of the "signal" of interest π.

It is important to notice that, although the map Φ used for sketching is a nonlinear function, the operator $\mathcal{A}$ is linear with respect to probability distributions[16]. Hence compressive learning can itself be seen as a generalization of compressive sensing: the signal of interest is the distribution π, from which we measure a small number of noisy linear observations. Moreover, and as will be made clear in the following sections, the sketching operator will most often be randomized.

Learning from the sketch Compared to the compressive sensing setting were the measurement operator typically reduces drastically the dimension, we here map signals from the infinite-dimensional set of probability distributions[17] on $\mathcal{X}$ to a finite number of observations. There is of course no hope to build $\mathcal{A}$ so that uniform recovery of all distributions is possible, but one can still imagine being able to recover

[14] If $\mu = \mu_+ - \mu_-$ is the Jordan decomposition (cf. Definition A.5) of μ, and $\mu_+ = \alpha_+\pi_+, \mu_- = \alpha_-\pi_-$ for some $\alpha_+, \alpha_- \geq 0$ and probability distributions π_+, π_- (by normalizing), then $\mathcal{A}(\mu) \triangleq \langle \mu, \Phi\rangle = \alpha_+\mathcal{A}(\pi_+) - \alpha_-\mathcal{A}(\pi_-)$.

[15] It at least converges to 0 almost surely as $n \to \infty$ by the law of large numbers, but most often with good concentration properties.

[16] i.e. $\mathcal{A}(\alpha\pi_1 + (1-\alpha)\pi_2) = \alpha\mathcal{A}(\pi_1) + (1-\alpha)\mathcal{A}(\pi_2)$ for any $\pi_1, \pi_2 \in \mathcal{P}(\mathcal{X})$ and $\alpha \in [0,1]$, and $\mathcal{A}$ is linear on the vector space of finite signed measures.

[17] Or, more generally, from the vector space of signed measures of $\mathcal{X}$, see Appendix A.1.

an approximation of π if it belongs to – or can be well approximated by – some low-dimensional model. This can be interpreted as a form of generalized sparsity: for instance in the case of Gaussian modeling, one will approximate π with a Gaussian mixture, which can be explicitly parametrized with a small number of parameters. Besides, depending on the learning task to solve, one might not be interested in recovering exactly π, but rather only some related statistics (cf. Section 2.2.2).

Most often (see next section), learning from the sketch takes the form of the optimization problem

$$\min_{\pi \in \mathfrak{S}} \|\mathcal{A}(\pi) - \tilde{\mathbf{s}}\|_2, \tag{2.12}$$

where $\mathfrak{S}$ denotes the chosen low-dimensional model of interest for the task to solve. This could for example be the set of Gaussian mixtures with a fixed number k of components for Gaussian modeling.

We recall that statistical learning problems are often modeled[18] via a risk function $\mathcal{R}(h, \pi)$ that one wants to minimize over $h \in \mathcal{H}$. Intuitively, the function $\mathbf{s} \mapsto \|\mathbf{s} - \tilde{\mathbf{s}}\|_2$ plays the role of a proxy for this risk function, and the model set $\mathfrak{S}$ plays the role of the hypothesis space. The approximation quality of this proxy will typically increase with the sketch size[19].

[18] See Section 1.1.1.

[19] At least for some families of sketches. See the connection with kernel methods in Section 2.3.

Note that this problem is quite similar to the well known generalized method of moments (GeMM) in the statistical literature [1]. The latter differs by the fact that the chosen statistics are rarely randomized – by opposition to the constructions of Φ discussed below –, and often chosen so that (2.12) can be solved in closed form, whereas compressive learning often leads to non-convex optimization problems for which more complex algorithms are needed (see Section 2.4).

2.2 Recovery guarantees using ideal decoders

The parallel between the compressive sensing and compressive learning having been established in Section 2.1, we can now expose how ideas from linear inverse problems have been leveraged in order to formulate learning guarantees. We first present in Section 2.2.1 how solving (2.12) can produce a solution (hypothesis) with controlled excess risk, provided that the sketching operator satisfies a specific kind of restricted isometry property. In Section 2.2.2, we discuss the specific setting of semi-parametric models such as PCA, where one does not need to reconstruct a distribution, but only a statistic in a finite-dimensional intermediate space – e.g. the covariance matrix for principal component analysis.

2.2.1 Instance-optimal decoders and control of the excess risk

We will use here the notations and concepts introduced in Section 1.1.1: one has access to a dataset $\mathbf{X}$ (with empirical distribution $\pi_{\mathbf{X}}$) made of samples drawn i.i.d. according to some distribution π, and one wishes

to recover an hypothesis $\hat{h} \in \mathcal{H}$ for which the excess risk

$$\Delta\mathcal{R}(\hat{h}, \pi) = \mathcal{R}(\hat{h}, \pi) - \mathcal{R}(h^*, \pi) \tag{2.13}$$

with respect to some optimal hypothesis $h^* \in \arg\min_{h\in\mathcal{H}} \mathcal{R}(h, \pi)$ is controlled.

Gribonval et al. suggested [5] a procedure to learn from the sketch with statistical guarantees in two steps. Firstly, a probability distribution $\hat{\pi} \triangleq \Delta(\tilde{\mathbf{s}})$ is recovered from the sketch using a function Δ whose choice is discussed below, called the decoder. Then, the hypothesis is recovered from $\hat{\pi}$ simply as

$$\hat{h} \in \arg\min_{h\in\mathcal{H}} \mathcal{R}(h, \hat{\pi}). \tag{2.14}$$

In order to measure the proximity between probability distributions with respect to the considered learning task, we introduce the notations $\mathcal{L}(\mathcal{H}) \triangleq \{l(h, \cdot) | h \in \mathcal{H}\}$ and[20]

$$\|\pi - \pi'\|_{\mathcal{L}(\mathcal{H})} \triangleq \sup_{h\in\mathcal{H}} |\mathcal{R}(h, \pi) - \mathcal{R}(h, \pi')|. \tag{2.15}$$

Although we skip some technical details here, $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ can be shown to be a semi-norm on the space of finite signed measures for which $l(h, \cdot)$ is integrable for any h (cf. Appendix A.2). Observe that if $\hat{\pi} \triangleq \Delta(\tilde{\mathbf{s}})$ is chosen so that its risk uniformly approximates the risk of π for all hypotheses, i.e. if $\|\pi - \hat{\pi}\|_{\mathcal{L}(\mathcal{H})} \leq \frac{1}{2}\eta$ for some constant η, then the excess risk (2.13) is bounded[21] by η. As a consequence, we now focus on building a decoder Δ for which this bound η can be controlled.

Learning as an inverse problem As stated in Section 2.1.5, the sketch can be written

$$\tilde{\mathbf{s}} = \mathcal{A}(\pi) + \mathbf{e}, \tag{2.16}$$

where $\mathbf{e} = \mathcal{A}(\pi_{\mathbf{X}} - \pi)$ can be interpreted as noise. Thus the role of the abstract decoder Δ introduced earlier is to recover π from the noisy linear measurements $\tilde{\mathbf{s}}$, the reconstruction error being measured according to $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$.

As it has been observed in Section 2.1, recovering signals from underdetermined linear systems requires some regularity assumption. We introduce for this purpose a general model $\mathfrak{S}$ of probability distributions of interest. Some concrete choices of $\mathfrak{S}$ are summarized in Section 2.5 for the different learning tasks considered, but this model should qualitatively contain the probability distributions that can be well approximated for the considered learning task, i.e. for which there exists an hypothesis $h \in \mathcal{H}$ for which the risk is low. This model plays a similar role to the set Σ_s of sparse vectors in compressive sensing. Note that we need the decoder Δ to be both robust to noise, and stable with respect to probability distributions that are only close to $\mathfrak{S}$. This can be written as

$$\forall \pi \in \mathcal{P}(\mathcal{X}), \forall \mathbf{e} \in \mathcal{Z}, \ \|\pi - \Delta(\mathcal{A}(\pi) + \mathbf{e})\|_{\mathcal{L}(\mathcal{H})} \lesssim d(\pi, \mathfrak{S}) + \|\mathbf{e}\|_2 \tag{2.17}$$

for some distance $d(\cdot, \mathfrak{S})$ to the model, and where $\lesssim$ denotes inequality up to a constant factor (see Figure 2.4).

[20] We use this definition for readability, as it is sufficient for this chapter. More generally, for any class $\mathcal{F}$ of measurable functions on $\mathcal{X}$ one can define the semi-norm

$$\|\mu\|_{\mathcal{F}} \triangleq \sup_{f\in\mathcal{F}} |\langle \mu, f \rangle|,$$

where $\langle \pi, f \rangle \triangleq \mathbf{E}_{\mathbf{x}\sim\pi} f(\mathbf{x})$ for a distribution π, and can be extended to finite signed measures via the Jordan decomposition. We refer the reader to Appendix A.2 for more details.

[21] Indeed, we simply have in this case

$$\begin{aligned}
&\mathcal{R}(\hat{h}, \pi) - \mathcal{R}(h^*, \pi) \\
&= \mathcal{R}(\hat{h}, \pi) - \mathcal{R}(\hat{h}, \hat{\pi}) + \mathcal{R}(\hat{h}, \hat{\pi}) - \mathcal{R}(h^*, \pi) \\
&\leq \mathcal{R}(\hat{h}, \pi) - \mathcal{R}(\hat{h}, \hat{\pi}) + \mathcal{R}(h^*, \hat{\pi}) - \mathcal{R}(h^*, \pi) \\
&\leq 2 \sup_{h\in\mathcal{H}} |\mathcal{R}(h, \hat{\pi}) - \mathcal{R}(h, \pi)|,
\end{aligned}$$

where $\mathcal{R}(\hat{h}, \pi_{\mathbf{X}}) \leq \mathcal{R}(h^*, \pi_{\mathbf{X}})$ holds by definition of $\hat{h}$.

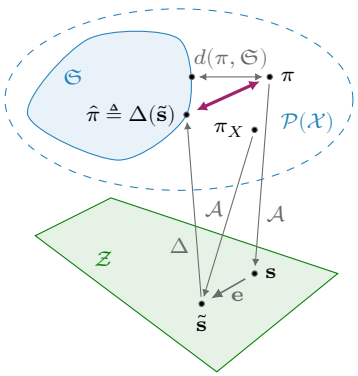


Figure 2.4: Instance optimality property: for any distribution π and noise $\mathbf{e}$, the distance symbolized by the thick magenta arrow can be controlled (in our setting, for the $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ semi-norm).

A decoder satisfying this property is called an instance optimal decoder, and it has been shown [60, Theorem 3 with $h \in \Sigma - \Sigma$] that the mere existence of an instance optimal decoder implies a lower[22] restricted isometry property: there exists a constant $C < \infty$ such that

$$\forall \tau, \tau' \in \mathfrak{S},\ \|\tau - \tau'\|_{\mathcal{L}(\mathcal{H})} \leq C \|\mathcal{A}(\tau) - \mathcal{A}(\tau')\|_2. \tag{2.18}$$

[22] By opposition to Equation (2.6) where we have both lower and upper inequalities.

But an interesting fact is that in the other direction, if (2.18) holds then the decoder

$$\Delta(s) = \arg\min_{\tau \in \mathfrak{S}} \|\mathcal{A}(\tau) - s\|_2 \tag{2.19}$$

is instance optimal [60, Theorems 7 and 4] for some $d(\cdot, \mathfrak{S})$ whose definition is omitted here. Proving (2.18) is thus the key idea to derive theoretical guarantees, and Section 2.3 focuses on this point.

These results have also been extended to nonlinear operators [61]. Also, we naturally assumed $\mathbf{X}$ to be made of i.i.d. samples drawn according to π, but the guarantees obtained with this strategy hold even when this is not the case, although controlling $\|\mathbf{e}\|_2$ might be more difficult.

A note on the loss metric Although the lower RIP (LRIP) (2.18) with respect to $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ is sufficient to derive statistical learning guarantees, it can be interesting as well to work with a variant of $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$, namely

$$\|\pi - \pi'\|_{\Delta\mathcal{L}(\mathcal{H})} \triangleq \sup_{h_1, h_2 \in \mathcal{H}} (\mathcal{R}(h_1, \pi) - \mathcal{R}(h_1, \pi')) - (\mathcal{R}(h_2, \pi) - \mathcal{R}(h_2, \pi')). \tag{2.20}$$

The function $\|\cdot\|_{\Delta\mathcal{L}(\mathcal{H})}$ can be extended to finite signed measures and checked to be a semi-norm[23] as well, we refer again the reader to Appendix A.2 for more details. The reason behind this choice is that more generic guarantees have been derived [5, Theorem 2.5] by establishing a LRIP with respect to $\|\cdot\|_{\Delta\mathcal{L}(\mathcal{H})}$ rather than to $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$, i.e. by proving

[23] On the subspace of measures with respect to which every $f \in \Delta\mathcal{L}(\mathcal{H}) = \{l(h_1, \cdot) - l(h_2, \cdot) | h_1, h_2 \in \mathcal{H}\}$ is integrable.

$$\forall \tau, \tau' \in \mathfrak{S},\ \|\tau - \tau'\|_{\Delta\mathcal{L}(\mathcal{H})} \leq C \|\mathcal{A}(\tau) - \mathcal{A}(\tau')\|_2 \tag{2.21}$$

for some $C > 0$. We stress that $\|\cdot\|_{\Delta\mathcal{L}(\mathcal{H})} \leq 2\|\cdot\|_{\mathcal{L}(\mathcal{H})}$, hence if the LRIP (2.18) holds (with respect to $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$) with constant C', then (2.21) holds as well with $C = 2C'$.

2.2.2 Generalization to semi-parametric models

The approach described in the previous section makes sense as soon as a meaningful parametric[24] model set $\mathfrak{S}$ can be associated to the task to solve. Provided that this model is "small" enough to allow the existence of a sketching operator $\mathcal{A}$ satisfying a lower RIP (2.18), then one can directly use its explicit parametrization to solve the inverse problem in $\mathcal{P}(\mathcal{X})$. However some problems do not induce such parametric models.

[24] i.e. $\mathfrak{S} = \{\pi_\theta \,|\, \theta \in \Theta\}$ where $\Theta \subseteq \mathbb{R}^p$ is finite-dimensional.

The case of PCA A typical example is principal component analysis[25], where the solution $h^* \in \arg\min_h \mathcal{R}_{\text{PCA}}(h, \pi)$ of the problem for a given (centered) distribution $\pi \in \mathcal{P}(\mathcal{X})$ and any fixed dimension k is well known to be entirely determined by the k first eigenvectors of the

[25] Cf. (1.1).

covariance matrix $\mathrm{Cov}(\pi)$. As a consequence, two distributions π_1, π_2 that have the same covariance matrix induce the same solution, and actually satisfy $\|\pi_1 - \pi_2\|_{\mathcal{L}_{\mathrm{PCA}}(\mathcal{H})} = 0$. Although it is technically possible to use as a model the set of distributions supported on k-dimensional subspaces, i.e. $\mathfrak{S}_k = \{\pi | \operatorname{rank}(\mathrm{Cov}(\pi)) \leq k\}$ (which are distributions for which the risk vanishes for some $h \in \mathcal{H}$), solving the inverse problem over $\mathfrak{S}$ would be inefficient because of the lack of a simple parametric formulation, but also useless given that only the support of the recovered distribution would be of practical interest. Such a model is said to be semi-parametric, in the sense that the supports of the distributions in the model can be parametrized, but the whole model is still infinite-dimensional.

Computing the covariance[26] matrix can already be interpreted as a compressive approach using the feature map $\Phi : \mathbf{x} \mapsto \mathbf{x}\mathbf{x}^T$, assuming the data is centered. However, in order to solve the PCA task one only needs as stated above to recover the k first eigenvectors of this matrix, i.e. its best rank-k approximation. As a consequence, further compression is possible using using tools from low-rank matrix recovery presented in Section 2.1.4. If $\mathcal{M} : \mathbb{R}^{d \times d} \to \mathbb{R}^m$ is an operator which satisfies an LRIP on the set of rank-$2k$ matrices[27], then one can show that the operator $\mathcal{A} : \pi \mapsto \mathcal{M}(\mathrm{Cov}(\pi))$ satisfies an LRIP[28] akin to (2.18) on $\mathfrak{S}_k$. We refer in the following to this approach as compressive PCA [5]. Reconstruction is then performed by reconstructing a rank-k estimate of the covariance matrix, but without estimating any probability density.

[26] We actually compute the autocorrelation matrix here rather than the covariance, but the approach will mostly be useful when considering centered data, so we make this assumption in the following and always refer to the covariance matrix.

[27] Typically a randomly designed operator, as discussed later in Section 2.5.

[28] See [5, Appendix E] for the precise constants

Other semi-parametric models This setting was later extended by Sheehan et al. [62], who cast the task of independent component analysis [63] into the same framework, but using as an intermediate statistic the fourth-order cumulant tensor rather than the covariance matrix. Learning unions of subspaces was also considered with a similar approach [62, Section 5.2], but for now still requires large sketch sizes.

2.3 Sketching operator design using kernel methods

We presented how compressive learning could be seen as a linear inverse problem, and explained that recovery guarantees exist as soon as a lower restricted isometry property (LRIP) (2.18) holds between the metric $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ induced by the loss function, and the distance between sketches $\|\mathcal{A}(\cdot)\|_2$. Apart from the semi-parametric settings where an explicit reduction to a finite-dimensional space is induced by the loss function, we however have not discussed so far how to prove this property, nor have we explained how to choose the sketching operator $\mathcal{A}$ in practice.

As detailed in Section 2.1.3, building linear operators between finite-dimensional spaces satisfying a RIP is well known to be possible using randomness. However, this becomes more tricky when working with infinite-dimensional spaces, such as the space $\mathcal{P}(\mathcal{X})$ of interest for us.

Some works provided useful tools for generalization to generic Hilbert spaces [64, 65], but without giving explicit or practical constructions of the measurement operators.

In this section, we give an overview of how kernel methods can be leveraged for this purpose. We will see that a kernel function κ (defined just below) implicitly defines under some conditions a metric $\|\cdot\|_\kappa$ between probability distributions (Section 2.3.3), which most often can be well approximated via an explicit finite-dimensional mapping (Section 2.3.2). It will thus often be easier to choose a kernel κ for which the LRIP between $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ and $\|\cdot\|_\kappa$ is satisfied (Section 2.3.4), and then choose Φ so that $\|\mathcal{A}(\cdot)\|_2 \approx \|\cdot\|_\kappa$. We start by introducing some concepts relative to kernels in Section 2.3.1.

2.3.1 Measuring similarity with kernels

Most standard machine learning algorithms, such as support vector machines[29] or principal component analysis, use the data samples $(\mathbf{x}_i)_{1\leq i\leq n}$ by computing inner products of the form $\langle \mathbf{x}_i, \mathbf{x}_j\rangle$. Kernel functions have originally been used in the learning community to extend such methods to non-linear settings. Let φ be a function (often referred to as the feature map) taking values in a space endowed with an inner product, that we also denote $\langle\cdot,\cdot\rangle$ for conciseness. One can define $\kappa(\mathbf{x},\mathbf{y}) \triangleq \langle\varphi(\mathbf{x}),\varphi(\mathbf{y})\rangle$ and replace all the inner products performed in an algorithm by this new similarity measure. This can be very interesting, especially when φ is chosen to be non-linear, to capture more subtle geometric properties of the dataset. In the other direction, one can wonder if a given similarity function $\kappa : \mathcal{X}\times\mathcal{X}\to\mathbb{K}$ (where $\mathbb{K}=\mathbb{R}$ or $\mathbb{C}$), which typically could be known to have a meaningful interpretation for the learning task to solve, can be expressed as a linear inner-product via a feature map; a class of particular interest for this purpose is the class of positive definite kernels.

[29] A support vector machine, or SVM, is a model for supervised classification or related tasks. In the standard binary (i.e. two classes) setting, it mainly consists in finding an hyperplane which separates the two classes with the largest possible margin, i.e. so that the samples of both classes are as far as possible from this hyperplane.

Definition 2.2: A function $\kappa : \mathcal{X}\times\mathcal{X}\to\mathbb{K}$ (where $\mathbb{K}=\mathbb{R}$ or $\mathbb{C}$) is called a positive definite kernel if for any $n\in\mathbb{N}$ and any $\mathbf{x}_1,...,\mathbf{x}_n\in\mathcal{X}$, the $n\times n$ matrix G with entries $G_{i,j}=\kappa(\mathbf{x}_i,\mathbf{x}_j)$ is symmetric (when $\mathbb{K}=\mathbb{R}$) or Hermitian (when $\mathbb{K}=\mathbb{C}$) and positive definite.

According to the Moore-Aronszajn theorem [66], any positive definite kernel defines a unique reproducible kernel Hilbert space (RKHS), i.e. a unique Hilbert space $\mathcal{H}$ of functions $f:\mathcal{X}\to\mathbb{K}$ endowed with a dot product $\langle\cdot,\cdot\rangle$, containing the $\{\kappa(\mathbf{x},\cdot)\}_{\mathbf{x}\in\mathcal{X}}$ and satisfying [67, Definition 2.9][30]:

1. for any $f\in\mathcal{H}$, and $\mathbf{x}\in\mathcal{X}$, $f(\mathbf{x})=\langle f,\kappa(\mathbf{x},\cdot)\rangle$;
2. $\mathcal{H}=\overline{\text{span}\{\kappa(\mathbf{x},\cdot)|\mathbf{x}\in\mathcal{X}\}}$.

The first property, which is known as the reproducing property, implies in particular that for any $\mathbf{x},\mathbf{y}\in\mathcal{X}$, $\kappa(\mathbf{x},\mathbf{y})=\langle\kappa(\mathbf{x},\cdot),\kappa(\mathbf{y},\cdot)\rangle$. We thus have a way to represent any positive definite kernel by an inner product via the map $\mathbf{x}\mapsto\kappa(\mathbf{x},\cdot)$. Although this property is interesting from a

[30] Here is another characterization: a RKHS is a Hilbert space of functions $f:\mathcal{X}\to\mathbb{K}$ for which the evaluation functional $\mathbf{x}\mapsto f(\mathbf{x})$ is continuous for any $\mathbf{x}\in\mathcal{X}$. The existence of a unique kernel satisfying the reproducing property can then be derived from the Riesz representation theorem.

theoretical perspective, working with the features $\kappa(\mathbf{x}, \cdot)$, which will typically be high- or infinite-dimensional, will often not be of practical interest.

Kernel trick The idea of replacing in a linear algorithm the inner-product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ by the quantity $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ for some kernel κ (and for each $\mathbf{x}_i, \mathbf{x}_j$) is known in the literature as the kernel trick, and allows to efficiently generalize existing algorithms to new similarity measures. Although evaluating $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is tantamount to computing an inner product in a high-dimensional space, the associated features $\varphi(\mathbf{x}_i)$ are most often never computed in practice – and φ is sometimes referred to as an "implicit embedding".

This concept was applied to many problems such as support vector machines [68], principal component analysis [69, 70] or unsupervised classification [71]. Numerous kernels have also been developed, to account for different geometric properties of the collections but also to deal with non-numerical or structured data.

However, most of the methods mentioned above still require for a dataset $\mathbf{x}_1, ..., \mathbf{x}_n$ to build the $n \times n$ kernel matrix K with entries $K_{ij} \triangleq \kappa(\mathbf{x}_j, \mathbf{x}_j)$. Although directly evaluating the kernel κ was initially perceived as beneficial compared to working with the high- or infinite-dimensional features $\kappa(\mathbf{x}_i, \cdot)$, computing and storing the kernel matrix can become prohibitive when working with large collections. As a consequence, some authors considered instead using kernels that can be computed or well approximated using finite low-dimensional feature maps. If φ is such a map, then one can simply run any algorithm on the features $\varphi(\mathbf{x}_1), ..., \varphi(\mathbf{x}_n)$ instead of the original data $\mathbf{x}_1, ..., \mathbf{x}_n$.

It is also important to notice that, even though it is often meaningful to start from a well-known kernel and to look for a finite-dimensional approximation, as it will be done in the next section, any given finite-dimensional feature map does in return define a positive definite kernel. As a consequence, any choice of a sketching operator Φ can be associated to a kernel function, and analyzed using the tools presented in the following sections.

2.3.2 Finite-dimensional feature maps for kernel approximation

In the following and unless otherwise specified, the term kernel will always refer to a positive definite kernel. Finite dimensional feature maps are of practical interest for compressive learning, as we will see that the feature map Φ used for sketching (whose dimension m should ideally be as small as possible) is always derived from a kernel.

Some kernels have known finite-dimensional maps. For instance, in dimension $d = 2$ the function $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$ can be computed as $\kappa(\mathbf{x}, \mathbf{y}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle$ via the mapping $\varphi : \mathbf{x} \mapsto [\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2]^T$. In some contexts, it can also be interesting to first define a feature map φ, and then to look at the induced kernel $\kappa(\cdot, \cdot) = \langle \varphi(\cdot), \varphi(\cdot) \rangle$. As mentioned above, most kernels that are of practical utility, e.g. because they

have nice regularity properties, do not come with finite-dimensional features. However, approximation using finite-dimensional features is still possible, i.e. it is interesting for a given kernel κ to find a map $\varphi : \mathcal{X} \to \mathbb{R}^m$ such that $\kappa(\mathbf{x}, \mathbf{y}) \approx \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle$ for all $\mathbf{x}, \mathbf{y}$.

Mercer's theorem [67, Theorem 2.10] provides a first step in this direction. Although we do not state the whole theorem here, one of its consequences is that a (positive definite) kernel κ can be approximated almost everywhere on $\mathcal{X}$ by an explicit finite-dimensional feature map, i.e. for any $\varepsilon > 0$, one can build a feature map $\varphi_\varepsilon : \mathcal{X} \to \mathbb{R}^m$ with $m \in \mathbb{N}$ such that $|\kappa(\mathbf{x}, \mathbf{y}) - \langle \varphi_\varepsilon(\mathbf{x}), \varphi_\varepsilon(\mathbf{y}) \rangle| \leq \varepsilon$ for almost every $\mathbf{x}, \mathbf{y} \in \mathcal{X}$.

An alternative approach was suggested by Rahimi and Recht [72] for the class of translation-invariant kernels, i.e. for kernels that can be written as $\kappa(\mathbf{x}, \mathbf{y}) = K(\mathbf{x} - \mathbf{y})$ for some function $K : \mathcal{X} \to \mathbb{R}$. The space $\mathcal{X}$ is here assumed to be a vector space. The approximation is based on the following result.

Theorem 2.3 (Bochner's theorem [73])

A bounded complex-valued and translation-invariant kernel $\kappa(\mathbf{x}, \mathbf{y}) = K(\mathbf{x} - \mathbf{y})$ defined on $\mathbb{R}^d$ is positive definite if and only if K is the Fourier transform of a non-negative Borel measure[31] Λ on $\mathbb{R}^d$.

31. i.e. any measure defined on the Borel σ-algebra.

Note in particular that, using the definition of the Fourier transform, the measure Λ is a probability distribution when $K(0) = 1$; we always make this assumption in the following. If κ is moreover real-valued, then we directly have for any $\mathbf{x}, \mathbf{y}$

$$\begin{aligned} \kappa(\mathbf{x}, \mathbf{y}) = K(\mathbf{x} - \mathbf{y}) &= \int_{\mathbb{R}^d} \cos(\boldsymbol{\omega}^T(\mathbf{x} - \mathbf{y})) d\Lambda(\boldsymbol{\omega}) \\ &= \int_{\mathbb{R}^d} \big(\cos(\boldsymbol{\omega}^T\mathbf{x})\cos(\boldsymbol{\omega}^T\mathbf{y}) + \sin(\boldsymbol{\omega}^T\mathbf{x})\sin(\boldsymbol{\omega}^T\mathbf{y})\big) d\Lambda(\boldsymbol{\omega}). \end{aligned} \tag{2.22}$$

Rahimi and Recht hence proposed to approximate any translation-invariant kernel κ with associated distribution Λ using the feature map

$$\varphi : \mathbf{x} \mapsto [\cos(\boldsymbol{\omega}_1^T\mathbf{x}), \sin(\boldsymbol{\omega}_1^T\mathbf{x}), ..., \cos(\boldsymbol{\omega}_m^T\mathbf{x}), \sin(\boldsymbol{\omega}_m^T\mathbf{x})]^T \in \mathbb{R}^{2m}, \tag{2.23}$$

where the $\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m$ are random vectors drawn i.i.d. according to Λ, and which can be interpreted as frequency vectors in light of (2.22). The features induced by φ are known in the literature as random Fourier features. We have $\mathbf{E}_{\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m} m^{-1} \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y})$ via Bochner's theorem, and more precise concentration results depending on the number of features m can naturally be derived. The original paper [72] provides uniform approximation guarantees for compact subsets of $\mathbb{R}^d$, and these results have been later refined [74]. We will consider in this thesis translation-invariant kernels, but approximation results also exist in broader settings, see e.g. [75].

Extensions Another common variant of (2.23) is $\mathbf{x} \mapsto [\cos(\boldsymbol{\omega}_1^T\mathbf{x} + u_1), ..., \cos(\boldsymbol{\omega}_m^T\mathbf{x} + u_m)]$ where $u_i \overset{i.i.d.}{\sim} \mathcal{U}([0, 2\pi[)^m$ and still $\boldsymbol{\omega}_i \overset{i.i.d.}{\sim} \Lambda$,

which also yields an unbiased estimator of the kernel, but often with poorer guarantees in comparison[32] [76].

Random Fourier features have been used to speed-up many different learning tasks. Multiple works suggested using structured matrices [77, 78] or optical processing units [79] to speed-up computation[33]. Alternative sampling schemes such as leverage scores [80] (for kernel ridge regression) or quasi Monte-Carlo sampling [81] have been considered to improve the concentration quality, or even data-dependent schemes [82]. Some works have considered the more generic task of kernel learning [83, 84], where the features themselves can be learned, or partially learned [85].

[32] At least for the Gaussian kernel (2.24), the variance is uniformly higher compared to (2.23). Uniform error bounds are provided, but also with larger constants.

[33] More details on acceleration in Chapter 5

The case of the Gaussian Kernel It is worth noting that the class of translation-invariant kernels includes already many common kernels, and should not be seen as too much of a limitation. It includes for instance the Gaussian kernel which will be of practical interest for us. It is defined as

$$\kappa_\sigma(\mathbf{x},\mathbf{y}) \triangleq K_\sigma(\mathbf{x}-\mathbf{y}) \triangleq \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}\right) \tag{2.24}$$

for some $\sigma > 0$. It turns out that this kernel induces a distribution of frequencies Λ which is itself a Gaussian distribution. Indeed, more generally for any positive definite matrix $\mathbf{\Sigma}$, using $\Lambda = \mathcal{N}(0, \mathbf{\Sigma})$ we have for any $\mathbf{z}$:

$$\begin{aligned}\mathbf{E}_{\boldsymbol{\omega}} \exp\big(\imath\boldsymbol{\omega}^T\mathbf{z}\big) &= \int_{\mathbb{R}^d} \exp\big(\imath\boldsymbol{\omega}^T\mathbf{z}\big)\frac{1}{\sqrt{2\pi\det(\mathbf{\Sigma})}}\exp\big(-\tfrac{1}{2}\boldsymbol{\omega}^T\mathbf{\Sigma}^{-1}\boldsymbol{\omega}\big)d\boldsymbol{\omega} \\ &= \int_{\mathbb{R}^d} \frac{1}{\sqrt{2\pi\det(\mathbf{\Sigma})}}\exp\big(-\tfrac{1}{2}(\boldsymbol{\omega}-\mathbf{\Sigma}i\mathbf{z})^T\mathbf{\Sigma}^{-1}(\boldsymbol{\omega}-\mathbf{\Sigma}i\mathbf{z}) - \tfrac{1}{2}\mathbf{z}^T\mathbf{\Sigma}\mathbf{z}\big)d\boldsymbol{\omega} \\ &= \exp\big(-\tfrac{1}{2}\mathbf{z}^T\mathbf{\Sigma}\mathbf{z}\big). \end{aligned} \tag{2.25}$$

Using $\mathbf{\Sigma} = \frac{1}{\sigma^2}I$, we conclude that the kernel κ_σ can be obtained via the frequency distribution $\Lambda = \mathcal{N}(0, \frac{1}{\sigma^2}I)$, i.e. the variance of the kernel κ_σ in the spatial domain is the inverse of the variance of the frequency distribution. The Gaussian kernel also belongs to the smaller class of radial basis functions (RBFs) kernels, which contains kernels that can be written as $\kappa(\mathbf{x},\mathbf{y}) = \phi(\|\mathbf{x}-\mathbf{y}\|_2)$ for some positive definite function ϕ.

2.3.3 Extension to distributions and mean embeddings

In this section, we explain how a given kernel implicitly defines a metric between probability distributions, and how finite-dimensional features introduced in Section 2.3.2 can be directly leveraged to design the sketching operator. We only give here a brief overview of useful concepts, and refer the reader to the review article of Muandet et al. [86] for more details on distributions embeddings.

Maximum mean discrepancy A common way to define a pseudo-metric[34] between distributions on $\mathcal{X}$ is to consider

[34] We recall that a pseudo-metric on a space E is an application $d : E \times E \to \mathbb{R}$ being positive, symmetric and subadditive. It differs from a metric in the fact that $d(p,q) = 0$ can hold for $p \neq q$.

$$d(p,q) \triangleq \|p-q\|_{\mathcal{F}} \triangleq \sup_{f\in\mathcal{F}} \left|\mathbf{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] - \mathbf{E}_{\mathbf{x}\sim p}[f(\mathbf{y})]\right| \tag{2.26}$$

for some class $\mathcal{F}$ of real-valued measurable functions on $\mathcal{X}$. Different pseudo-metric can be described in that way, such as the total variation or Wasserstein distances. The semi-norm $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ induced by the loss[35] also follows this definition, and we refer the reader to Appendix A.2 for more details. An interesting choice in our context is to take $\mathcal{F}$ to be the unit ball in some RKHS $\mathcal{H}$ according to $\|\cdot\|_{\mathcal{H}}$ (naturally induced by the inner-product), as it has been suggested by Gretton et al. [87]. If κ is the reproducing kernel of a RKHS, we denote this pseudo-metric $\|\cdot\|_\kappa$; it is known as the maximum mean discrepancy (MMD). Kernels for which $\|\cdot\|_\kappa$ is a true metric are called characteristic kernels and can be characterized precisely [88]. In particular, a translation-invariant kernel is characteristic if and only if its associated Fourier transform Λ (i.e. as defined in Theorem 2.3) satisfies $\operatorname{supp}(\Lambda) = \mathbb{R}^d$.

[35] Cf. Section 2.2.1.

Mean kernel and mean embedding We provide some other characterizations that are of interest here. Any kernel κ can naturally be extended to distributions as

$$\kappa(\pi_1,\pi_2) \triangleq \mathbf{E}_{\mathbf{x}\sim\pi_1}\mathbf{E}_{\mathbf{y}\sim\pi_2}\kappa(\mathbf{x},\mathbf{y}), \tag{2.27}$$

which is known as the mean kernel but denoted κ as well for simplicity. The definition naturally extends to finite signed measures via the Jordan decomposition[36]. The maximum mean discrepancy then simply derives from the mean kernel, i.e. for any finite signed measure μ we have

$$\|\mu\|_\kappa = \sqrt{\kappa(\mu,\mu)}. \tag{2.28}$$

[36] cf. Definition A.5 and remarks of Section 2.1.5.

Another close concept is the notion of mean embedding, which extends the feature map $\kappa(\mathbf{x},\cdot)$ by associating to every probability distribution a mean element in the reproducing kernel Hilbert space.

Definition 2.4 (Kernel mean embedding [89]): The kernel mean embedding of $\mathcal{P}(\mathcal{X})$ into a reproducing kernel Hilbert space $\mathcal{H}$ with reproducing kernel κ is the mapping

$$\mathrm{KME} : \pi \mapsto \int_{\mathcal{X}} \kappa(\mathbf{x},\cdot)d\pi(\mathbf{x}). \tag{2.29}$$

A sufficient condition for $\mathrm{KME}(\pi)$ to exist (as defined in (2.29)) and to belong to $\mathcal{H}$ is $\mathbf{E}_{\mathbf{x}\sim\pi}[\kappa(\mathbf{x},\mathbf{x})] < \infty$. For any $f \in \mathcal{H}$ and $\pi \in \mathcal{P}(\mathcal{X})$, the reproducing property implies that $\mathbf{E}_{\mathbf{x}\sim\pi} f(\mathbf{x}) = \langle f, \mathrm{KME}(\pi)\rangle$. This implies in particular that the mean kernel between two probability distributions π_1,π_2 can be written $\kappa(\pi_1,\pi_2) = \langle \mathrm{KME}(\pi_1), \mathrm{KME}(\pi_2)\rangle$. Mean embeddings and the MMD have been used for many applications, including (not exhaustively) density estimation [90], clustering [91], or posterior design for Bayesian estimation [92].

Connection with random features The connection with the random features introduced in the previous section now becomes clear. Let κ

be a kernel which can be written[37] $\kappa(\mathbf{x},\mathbf{y}) = \mathbf{E}_{\boldsymbol{\omega}\sim\Lambda}\langle\varphi_{\boldsymbol{\omega}}(\mathbf{x}),\varphi_{\boldsymbol{\omega}}(\mathbf{y})\rangle$ for some feature function $\varphi_{\boldsymbol{\omega}} : \mathcal{X} \to \mathbb{C}$ and distribution Λ. Let Φ be the feature map defined as

$$\Phi(\mathbf{x}) = [\varphi_{\boldsymbol{\omega}_1}(\mathbf{x}),...,\varphi_{\boldsymbol{\omega}_m}(\mathbf{x})]^T, \tag{2.30}$$

where $m \in \mathbb{N}$ and the $(\boldsymbol{\omega}_i)_{1\leq i\leq m}$ are drawn i.i.d. with respect to Λ. We recall that we defined earlier the sketching operator in (2.10) as $\mathcal{A}(\pi) \triangleq \mathbf{E}_{\mathbf{x}\sim\pi}\Phi(\mathbf{x})$. Hence the normalized inner product between the sketches of any two distributions π_1, π_2 is, in expectation over the draw of Φ:

$$\begin{aligned}\mathbf{E}\tfrac{1}{m}\langle\mathcal{A}(\pi_1),\mathcal{A}(\pi_2)\rangle &= \tfrac{1}{m}\mathbf{E}_{\boldsymbol{\omega}_1,...,\boldsymbol{\omega}_m \overset{i.i.d.}{\sim}\Lambda}\left\langle\mathbf{E}_{\mathbf{x}\sim\pi_1}\Phi(\mathbf{x}),\mathbf{E}_{\mathbf{y}\sim\pi_2}\Phi(\mathbf{y})\right\rangle \\ &= \mathbf{E}_{\mathbf{x}\sim\pi_1}\mathbf{E}_{\mathbf{y}\sim\pi_2}\mathbf{E}_{\boldsymbol{\omega}\sim\Lambda}\langle\varphi_{\boldsymbol{\omega}}(\mathbf{x}),\varphi_{\boldsymbol{\omega}}(\mathbf{y})\rangle \\ &= \kappa(\pi_1,\pi_2),\end{aligned}$$

i.e. the mean kernel between the distributions. The number m of features used in Φ controls the concentration and, of course, needs to be chosen wisely. An important consequence is that the pseudo-metric $\|\mathcal{A}(\pi_1-\pi_2)\|_2$ can be interpreted as an empirical estimate of the maximum mean discrepancy $\|\pi_1-\pi_2\|_\kappa$, which gives a whole new meaning to the geometry of the feature space $\mathcal{Z}$ to which the computed sketches belong.

We recall that our original motivation was to establish a restricted isometry property of the form (2.18) with respect to $\|\mathcal{A}(\cdot)\|_2$. Due to the regularity properties of $\|\cdot\|_\kappa$, it might be easier to find or design a kernel for which the LRIP (2.18) is satisfied with respect to $\|\cdot\|_\kappa$ with high probability. When this is possible, one can in a second time look at the concentration properties, and choose m so that the LRIP will be satisfied with respect to $\|\mathcal{A}(\cdot)\|_2$ with high probability.

Different kinds of concentration results can be considered. We refer the reader to [5, 6] for results that can be applied for the settings of clustering and Gaussian modeling using random Fourier features. The proof techniques usually rely on a pointwise concentration result, i.e. controlling for $t > 0$ and every τ, τ' in the model the probability $P(\left|\|\mathcal{A}(\tau-\tau')\|/\|\tau-\tau'\|_\kappa - 1\right| \geq t)$. Such a result can then be extended to a uniform result over the whole model by taking into account its size, measured in terms of its covering number, i.e. the minimum number of elementary balls it takes to cover it entirely with respect to a well chosen norm.

2.3.4 Tools for the lower restricted isometry property

We recall that the strategy sketched in Section 2.2.1 to derive recovery guarantees relies on a lower RIP of the form (2.18), i.e. between the pseudo-metrics $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ and $\|\mathcal{A}(\cdot)\|_2$. It was shown in the previous sections that kernel functions naturally induce (pseudo-)norms, which in many cases can be well approximated with explicit and randomized finite-dimensional feature maps. Assuming that the concentration of such features can be well controlled independently of the kernel

[37] For translation-invariant kernels, this construction can be derived from Bochner's theorem Theorem 2.3.

(which implicitly assumes that the considered model of distributions is somehow "small" enough), it still remains to choose a kernel for which the lower RIP between $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ and $\|\cdot\|_\kappa$ (i.e. the MMD) holds.

We do not provide more details on this matter, but refer again the interested reader to [6] for a general methodology which applies to models of mixtures of some base parametric distribution family (e.g. normal distributions, Diracs). These results use some common tools from compressive sensing, such as a generalized notion of coherence between atomic elements of the model.

2.4 Algorithms to learn from the sketch

The strategy exposed above to learn with theoretical guarantees was based on the ideal decoder (2.19). In practice however, the optimization problem (2.12) is not-convex and difficult to solve. We present here the main heuristics that have been proposed in the literature. Although presented here in a generic manner and probably usable in other contexts, these heuristics have mainly been used when sketching with random Fourier features (2.23).

Thresholding Bourrier et al. first proposed an algorithm presented as a generalization of iterative hard thresholding [3, 93], with application to density fitting using Gaussian mixtures. Considering a family of distributions (p_θ) parametrized[38] by $\theta \in \mathbb{R}^d$, the algorithm consists in initializing and updating a support $\{\theta_1, ..., \theta_k\}$ and associated weights $(\alpha_i)_{1\leq i\leq k}$ representing a "sparse" mixture $p = \sum_{i=1}^k \alpha_k p_{\theta_k}$. The support is updated by iteratively choosing some new atoms $\left\{\tilde{\theta}_1, ..., \tilde{\theta}_l\right\}$ having a high correlation with the residual $\mathcal{A}(p) - \tilde{\mathbf{s}}$, which are added to it, and then reducing (thresholding) the support again down to the k atoms that best fit the empirical sketch $\tilde{\mathbf{s}}$; this last step is performed via the non-negative least square problem

[38] p_θ could for instance be the multivariate normal distribution with mean θ and identity covariance

$$\min_{\alpha\in\mathbb{R}_+^{k+l}} \|\tilde{\mathbf{s}} - [\mathcal{A}(p_{\theta_1}), ..., \mathcal{A}(p_{\theta_k}), \mathcal{A}(p_{\tilde{\theta}_1}), ..., \mathcal{A}(p_{\tilde{\theta}_l})]\alpha\|^2$$

and keeping the atoms associated to the highest weights. Note that although the support is extended and reduced at each step, the mixture is still somehow always sparse by nature, limiting the connexion with thresholding algorithms used in compressive sensing. This method also bears similitudes with CoSaMP [42] by the way it selects multiple atoms at each iteration, although initialization differs.

Greedy approaches A generalization of orthogonal matching pursuit was proposed by Keriven et al., with applications to both Gaussian modeling [94, 4] and k-means clustering CKM [2].

We give a generic formulation of this algorithm in Algorithm 2.1 below; using the same notations as above, it starts from an empty support and iteratively adds a new atom to the support at line 3. This operation being a nonconvex optimization problem, a local optimum is chosen. At line 4, the weights are computed via a non-negative

least square problem – they will be used to update the residual. Then, a final optimization step is performed at line 5. This optimization is performed by starting from the current support and weights; it thus allows to correct slightly the previously chosen atoms, and has been shown to be useful in practice due to the lack of incoherence of the dictionary, but this step should not drastically change the current solution.

This process is repeated k times, yielding a mixture of k atoms. A variant with replacement steps has also been proposed [94], following the idea orthogonal matching pursuit with replacement [41].

Input: Sketch $\tilde{\mathbf{s}}$, sketching operator $\mathcal{A}$, size of mixture k.

1 $\hat{r} \leftarrow \tilde{\mathbf{s}}$, $\Theta \leftarrow \emptyset$ // Init

2 **for** $i \leftarrow 1$ **to** k **do**

3 $\quad \Theta \leftarrow \Theta \cup \left\{ \arg\max_{\theta} \ \mathfrak{R}\left\langle \frac{\mathcal{A}(p_\theta)}{\|\mathcal{A}(p_\theta)\|}, \hat{r} \right\rangle \right\}$ // Pick new atom

4 $\quad \alpha \leftarrow \arg\min_{\alpha \geq 0} \left\| \tilde{\mathbf{s}} - \sum_{j=1}^{|\Theta|} \alpha_j \mathcal{A}(p_{\theta_j}) \right\|$ // Find weights

5 $\quad \Theta, \alpha \leftarrow \arg\min_{\Theta, \alpha \geq 0} \left\| \tilde{\mathbf{s}} - \sum_{j=1}^{|\Theta|} \alpha_j \mathcal{A}(p_{\theta_j}) \right\|$ // Adjust atoms

6 $\quad \hat{r} \leftarrow \tilde{\mathbf{s}} - \sum_{j=1}^{|\Theta|} \alpha_j \mathcal{A}(p_{\theta_j})$ // Update residual

7 **return** the support Θ, the weights α.

Algorithm 2.1: CL-OMP. Here $\mathfrak{R}$ denotes the function extracting the real part of a complex number.

In the following, we refer to this approach as compressive learning orthogonal matching pursuit (CL-OMP). The variant with replacement is denoted CL-OMPR. Note that the algorithm is quite versatile in the sense that it can easily be adapted to new parametric families p_θ, as soon as one can compute $\mathcal{A}(p_\theta)$ and the gradients needed for the optimization steps. Some reconstruction guarantees can be obtained for some specific families of kernels [95, 96] when assuming that line 3 can be solved.

Note that this algorithm can be adapted to the PCA setting[39], where one does not reconstruct a distribution but directly a low-rank estimate of the covariance matrix. ADMiRA [51] is another greedy algorithm for low-rank reconstruction, but differs in the fact that it adds multiple new atoms at each iteration, similarly to CoSaMP (see Section 2.1.2).

[39] One would then build a $d \times k$ matrix, where k is the desired rank, by starting from one vector and iteratively adding new columns.

Beurling LASSO Although greedy methods have the advantage of being generic, we should not occult the fact that application-specific algorithms exist as well. The problem of compressive clustering, where one aims at retrieving a mixture of Diracs (sometimes also referred to as "spikes"), is indeed very close to the Beurling LASSO problem, which could in our setting be written as

$$\min_{\mu \in \mathcal{M}(\mathcal{X})} \|\mathcal{A}(\mu) - \tilde{\mathbf{s}}\|_2^2 + \lambda |\mu|(\mathcal{X}), \tag{2.31}$$

where $\mathcal{M}(\mathcal{X})$ is the set of finite signed measures on $\mathcal{X}$, λ is a regularization parameter and $|\mu|(\mathcal{X}) \triangleq \sup_{\text{partitions } (E_i)_{1 \leq i \leq \infty} \text{ of } \mathcal{X}} \sum_{i=1}^{\infty} |\mu(E_i)|$ denotes the total variation of μ. The latter can be interpreted as a continuous generalization of the l_1 norm, and favors sparse measures, i.e.

mixtures of Diracs. Methods such as the Frank-Wolfe algorithm and variants can be leveraged [97, 98, 99, 100] to address this problem, and bear connections with CL-OMP.

2.5 Learning tasks with known compressive approaches

Previous sections introduced the key ideas behind compressive learning. We now enumerate the different learning tasks for which compressive learning techniques have been successfully applied in the literature. The three tasks introduced in the introduction (density fitting with Gaussian mixture models, k-means clustering and PCA) are first discussed, and we then mention a few other applications for which empirical successes have been obtained. Tables summarizing these different compressive learning approaches are provided at the end of the section.

Density fitting Using Gaussian mixture models to fit the empirical density[40] was one of the first applications of compressive learning. In this context, the sketch is computed using random Fourier features (2.23) with Gaussian i.i.d. vectors $\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m$.

[40] Cf. Definition 1.3.

It is interesting to note that when using such features, the maximum mean discrepancy between any two distribution π_1, π_2, defined at (2.28), and which can be approximated by the distance between the finite-dimensional sketches, can be rewritten according to [88, Corollary 4.(iii)]

$$\|\pi_1 - \pi_2\|_\kappa = \|\kappa \star \pi_1 - \kappa \star \pi_2\|_{L^2(\mathbb{R}^d)} \approx \|\mathcal{A}(\pi_1) - \mathcal{A}(\pi_2)\|_2 \qquad (2.32)$$

where $\star$ denotes the convolution operator, and κ the Gaussian kernel associated to the chosen distribution of frequencies; hence sketching can be interpreted as smoothing the input distribution. From a statistical perspective, the sketch can also be seen as a collection of random samples – at frequencies $\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m$ – of the empirical characteristic function.

Reconstruction is performed by solving (2.12), using as a model the set of mixtures of k Gaussian distributions. Empirical success were first obtained by Bourrier et al. [3, 101] using the thresholding algorithm described in Section 2.4. CL-OMPR was then proposed as an alternative algorithm [4].

Statistical learning guarantees have later been obtained [5, 6] for the optimal solution of the inverse problem[41] using the strategy described in Sections 2.2.1 and 2.3, for some refined distribution of the frequencies and with additional separation assumptions between the centers of the Gaussians, provided $m \gtrsim k^2 d$ (up to logarithmic terms; note that the bound also depends on the scale of the kernel defining the sketching operator).

[41] We mean here that these guarantees hold for the distribution recovered by the optimal decoder (2.19), but we still lack guarantees on the heuristics such as CL-OMPR used in practice.

Density fitting using α-stable distributions has also been considered empirically for blind source separation [102]; such distributions enjoy

desirable modeling properties, but the associated likelihood has no closed form, and rewriting the task as a moment-fitting problem solves this problem.

Clustering There are very different ways to cluster a dataset[42], however the k-means formulation given in Definition 1.2 can be solved using again Fourier features. Reconstruction with the CL-OMPR algorithm has been proposed [2], and is performed via solving (2.12), but this time on the set of mixtures of k Diracs. Guarantees have been obtained with the same formalism [6], albeit the distribution of the frequencies differs slightly compared to Gaussian modeling. These guarantees hold provided that the clusters are sufficiently separated[43], and as soon as the sketch size satisfies (up to log factors) $m \gtrsim k^2 d$, which might still be pessimistic as only $p = kd$ parameters are learned. Note that the k-medians problem (where the hypothesis is still a set of k points $h = \{\mathbf{c}_1, ..., \mathbf{c}_k\}$, but the error is measured with the euclidean distance, i.e. the loss becomes $l(h, \mathbf{x}) = \min_{1 \leq i \leq k} \|\mathbf{x} - \mathbf{c}_i\|$ without a square) can also be addressed with the same formalism.

[42] For k-means, we try to recover optimal centroids centers, but we could also optimize the boundaries between clusters, or produce hierarchical clusterings, etc.

[43] We will see in Chapter 10 that such a separation is actually necessary to establish a LRIP.

The usage of random Fourier features follows early works in compressive sensing [35], where it is known that recovery of sparse signals from few random samples in the Fourier domain is possible. In a continuous setting, recovering a mixture of Diracs from samples of its Fourier transform is known as super-resolution or "off the grid" compressive sensing [103], and bears strong similarities compressive clustering. Samples are however in that case most often deterministically chosen, which brings slightly different theoretical guarantees.

Schellekens et al. suggested a quantized variant of the sketching mechanism [104], which uses Fourier features with quantized real and imaginary parts, i.e. in $\{-1, 1\} + \imath\{-1, 1\}$. Although quantization is "lost" in the average operation, this still reduces the sketching computational cost, and storage in the case where the individual sketches should be sent over a network. This quantization requires adding a random "dithering" $\mathbf{u} \in [0, 2\pi[^m$ before applying the non-linearity.

As we will make extensive use of random Fourier features throughout the thesis, and because guarantees for quantized features will also be provided in Part IV, we give now a unique definition and notation for these features, which generalizes (2.23).

Definition 2.5 (Random Fourier features): For any matrix of frequencies $\mathbf{\Omega} = [\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_m] \in \mathbb{R}^{d \times m}$ and dithering vector $\mathbf{u} \in [0, 2\pi[^m$, the random Fourier feature map is defined by

$$\Phi^{\text{RFF}}(\mathbf{x}) \triangleq \rho\left(\mathbf{\Omega}^T\mathbf{x} + \mathbf{u}\right) + \imath\rho\left(\mathbf{\Omega}^T\mathbf{x} + \mathbf{u} - \frac{\pi}{2}\right) \in \mathbb{C}^m, \qquad (2.33)$$

where ρ is applied pointwise and with the particular cases

- $\mathbf{u} = \mathbf{0}$ and $\rho = \cos$ for unquantized features[44];
- $\mathbf{u} \in [0, 2\pi[^m$ and $\rho = 2^{-1/2}\,\text{sign} \circ \cos$ for quantized features.

44. Then $\Phi^{\text{RFF}}(\mathbf{x}) \triangleq \exp(\imath\mathbf{\Omega}^T\mathbf{x})$.

In the rest of the manuscript, and unless specified otherwise, the

frequency vectors $\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m$ will be drawn i.i.d. according to a multivariate normal Gaussian distribution, so that the inner product between the induced features approximates a Gaussian kernel as shown in Section 2.3.2. We however dissociate on purpose the frequency distribution from the feature map, as we will consider in Chapter 5 other frequency distributions which may still approximate the same Gaussian kernel.

Semi-parametric models We discussed in Section 2.2.2 the case of PCA[45], and how it fits in the compressive learning formalism using tools from low-rank matrix recovery. Combining the reduction from a distribution to its covariance and the subsequent compression using a random linear measurement operator on the set of matrices, we obtain the following features[46].

[45] Cf. Definition 1.1.

> **Definition 2.6 (Random quadratic features):** For any matrix $\boldsymbol{\Omega} = [\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_m] \in \mathbb{R}^{d \times m}$, the associated random quadratic feature map is defined for any $\mathbf{x} \in \mathcal{X}$ as
>
> $$\Phi^{\text{RQF}}(\mathbf{x}) \triangleq [(\boldsymbol{\omega}_1^T \mathbf{x})^2, \dots, (\boldsymbol{\omega}_m^T \mathbf{x})^2]^T,$$
>
> i.e. corresponds to using the nonlinearity $f(\cdot) = (\cdot)^2$.

[46] In practice, we have $\mathcal{A}(\pi) \triangleq \mathcal{M}(\text{Cov}(\pi))$, where we only need $\mathcal{M}$ to satisfy a LRIP for low-rank matrices. There are multiple ways to do that, and the suggested construction, which can be interpreted as a collection of rank-1 measurements, i.e. $\mathcal{M} : \mathbf{C} \mapsto [\langle \boldsymbol{\omega}_1 \boldsymbol{\omega}_1^T, \mathbf{C} \rangle_F, ..., \langle \boldsymbol{\omega}_m \boldsymbol{\omega}_m^T, \mathbf{C} \rangle_F]$, is only one possibility.

Unless otherwise specified, we consider that the $(\boldsymbol{\omega}_i)_{1 \leq i \leq m}$ are i.i.d. Gaussian. We will see that the structured operators from Chapter 5 can also be used for such features.

The model considered when fitting a k-dimensional subspace is the set of distributions with a rank-k covariance matrix, and multiple algorithms exist in the literature for reconstruction. Reconstruction is possible as soon as m is of the order of $k(n-k)$, which is the dimension of the Grassmanian $\text{Gr}(k, \mathbb{R}^d)$, i.e. of the manifold of k-dimensional linear subspaces in $\mathbb{R}^d$.

Independent component analysis (ICA) has been performed in a similar manner [63]. Approaches using the fourth-order cumulant tensor were known in the literature, yielding a finite-dimensional representation with $\Theta(d^4)$ parameters, which can be further reduced via a randomized operator down to $\Theta(d^2)$ components.

Learning unions of subspaces (UoS) has also been considered under this formalism [105]. This approach relies on generalized PCA [106], and aims at recovering a low-rank approximation of the autocorrelation matrix of the so-called "Veronese" embeddings[47].

[47] These are made of all the monomials of degree N of the data, where N denotes the number of subspaces to recover

Supervised learning Knowing whether and how the sketching mechanism can be modified to take into account labels in order to solve supervised learning tasks is still an open question. However, for tasks such as classification or class membership verification (i.e. verify if a given sample belongs to a given class), it is at least possible when the number of classes is small to compute a sketch for each class. Then any new sample can be sketched and compared to the sketches of the different classes. This approach was initially proposed for the task

of speaker verification[48] [107, Section 5.4.6], i.e. checking if an audio sample comes from one specific user, and a similar idea was used for classification [108]. This approach actually emulates Parzen windows [109], but using random Features in order to approximate the chosen kernel.

High-dimensional regression has also been considered using the GLLiM model [110], which allows to recast regression as a Gaussian modeling problem with additional structure constraints on the covariance matrices of the different components. Sketching is performed jointly on both components, i.e. regression from $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d_2}$ is done by learning a GMM in dimension $d_1 + d_2$, and each sample $(\mathbf{x}_i^1 \in \mathbb{R}^{d_1}, \mathbf{x}_i^2 \in \mathbb{R}^{d_2})$ is sketched as one single vector $[(\mathbf{x}_i^1)^T, (\mathbf{x}_i^2)^T]^T$. Some elements have been written down[49], but never published as the method was numerically quite unstable.

We summarize in Table 2.1 the distinctive features of the three tasks for which theoretical guarantees have been derived, and in Table 2.2 the other tasks which have been addressed experimentally with various compressive approaches.

[48] To be exact, this work considered using both a sketch per class and an additional "global" sketch.

[49] See [111, Section 4.2] for an internship report, following discussions with N. Keriven, A. Deleforge and R. Gribonval.

Task	Hypothesis	Loss $l(h, \mathbf{x})$	Features	Algorithm	Guarantees
k-means/ medians	Set of points $h = \{\mathbf{c}_1, ..., \mathbf{c}_k\}$	$\min_{1\leq i\leq k} \|\mathbf{x} - \mathbf{c}_i\|_2^p$ $p = 2$ for k-means $p = 1$ for k-medians (Def. 1.2)	RFF (Def. 2.5)	CL-OMPR [2], CL-AMP (Chap. 6)	For $m \gtrsim k^2 d \times$ log factors [6], assuming $\forall i \|\mathbf{c}_i\|_2 \leq R$, $\min \|\mathbf{c}_i - \mathbf{c}_j\|_2^2 \gtrsim \sigma_\kappa^2 \log(ek)$, weighted features.
GMM fitting	GMM with means $(\mathbf{c}_i)_{1\leq i\leq k}$, fixed covariances $(\mathbf{\Sigma}_i)_{1\leq i\leq k}$ s.t. $\forall i \mathbf{\Sigma}_i = \mathbf{\Sigma}$, weights	$-\log(p_h(\mathbf{x}))$ (Def. 1.3)	RFF (Def. 2.5)	CL-OMPR [3, 107]	For $m \gtrsim k^2 d \times$ log factors [6] assuming $\max \|\mathbf{c}_i\|_{\mathbf{\Sigma}} \leq R$ $\min \|\mathbf{c}_i - \mathbf{c}_j\|_{\mathbf{\Sigma}}^2 \gtrsim (2 + \sigma_\kappa^2) \log(ek)$
PCA	Subspace $h \in \mathrm{Gr}(k, \mathbb{R}^d)$	$\|\mathbf{x} - \Pi_h \mathbf{x}\|_2^2$ (Def. 1.1)	RQF (Def. 2.6)	Discussed in Sec. 9.3	For $m \gtrsim kd$ [5], via a matrix RIP.

Table 2.1: Summary of the learning tasks which have been addressed using compressive methods and for which theoretical guarantees have been derived. For GMMs, we use the notation $\|\mathbf{x}\|_{\mathbf{\Sigma}} = \mathbf{x}\mathbf{\Sigma}^{-1}\mathbf{x}$, and p_h denotes the density of the mixture. For PCA, Π_h denotes the orthogonal projector onto h.

Task	Features	Approach
ICA	Random quartic features	Sketching of the data 4-th order moment. Recovery of a diagonalizable (via an orthogonal matrix) 4-th order tensor [63]. Inverse problem solved via iterative projected gradient [63].
UoS	Quadratic random measurements of a polynomial embedding	Recovery of a low-rank approximation of the autocorrelation matrix of the polynomial embeddings [105].
Regression	RFF (Def. 2.5)	Fitting a Gaussian mixture model on the joint data [111, Section 4.2]. Reconstruction via CL-OMPR.
Classification	RFF (Def. 2.5)	Density fitting for each class independently (one sketch by class) [108].
Class membership	RFF (Def. 2.5)	Density fitting for each class (one sketch by class) + 1 "universal" sketch [107, Section 5.4.6].

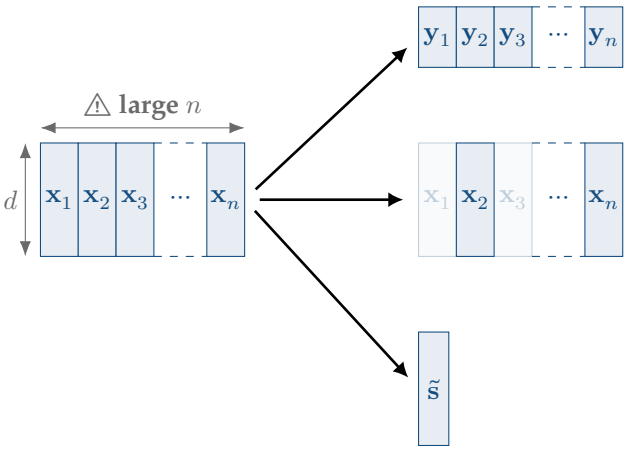
Table 2.2: Summary of other tasks which have been empirically addressed using compressive methods. ICA stands for "independent component analysis" and UoS for "union of subspaces" learning. For ICA, the sketch is made of random linear observations of the 4-th order cumulant tensor of the data (hence quartic w.r.t. the original data).

Chapter 3
Large-Scale Learning: Related Work

Contents

We presented in Chapter 2 the compressive learning framework, and stressed that it was very well suited for large scale learning and could by design be used in distributed and streaming scenarios. There exist however many other ways to solve learning tasks on large data collections. Going into the detail of each of these techniques is out of the scope of this chapter, but we propose to briefly present the different families of methods coexisting in the literature.

These methods can broadly be classified into three categories as depicted in Figure 3.1: computation of global statistics from the whole collection – including but not limited to the sketches of Chapter 2 –, reduction of the dimensionality, and reduction of the number of samples. All of these methods are sometimes called sketching techniques in the literature, as they all rely on a *sketch*, i.e. a much smaller representation of the dataset. Note however that the nature of the sketch will vary a lot between the different approaches.



▷ **Dimensionality Reduction**
Data-dependent or independent.
Cf. Section 3.2.

▷ **Coresets**
Subsampling, geometric decompositions.
Cf. Section 3.3.

▷ **Sketching**
Cf. Chapter 2 for sketches of moments, and Section 3.1 for other "linear" sketches.

Figure 3.1: Different approaches to learn efficiently from large data collections.

We will introduce in Section 3.1 some simple sketches from the database literature to answer basic queries about data streams, and cover in Sections 3.2 and 3.3 different methods to reduce the dimensionality and the number of samples in a collection, focusing mainly on randomized methods. We also briefly explain in Section 3.4 how these tools can be generalized or combined to perform some linear algebra

tasks in a randomized manner.

This chapter is not exhaustive as many different techniques have been introduced for large-scale learning. For instance, although some of the presented methods do extend naturally to the distributed setting, we reason in the centralized setting – where one data curator has access to the data matrix (or stream) $\mathbf{X}$, and we do not cover techniques such as federated learning [112], where data remains decentralized while learning. We do also not detail online and stochastic optimization techniques, and refer the reader for instance to the work of Slavakis et al. [113] on that matter.

Most of the approaches presented below are somehow generic and can be applied to various learning tasks. Application-specific surveys exist in the literature, see for instance [114] for the clustering setting.

3.1 Approximate query processing over data streams

We already stressed that algorithms scaling more than linearly with the dimension d or the number of samples n quickly become unusable for large collections. Although this might seem obvious for complex learning methods, it turns out that even answering basic queries or computing simple statistics can be costly owing to the multiple characteristics of data collections listed in Section 1.1.2, and in particular in the streaming model when data cannot be stored.

For many applications however, computing only an approximation of the desired quantity, or providing an answer which holds with high probability in the case of a binary query, is sufficient. A wide range of "sketching" algorithms, which rely on the computation of a small[1] synopsis of the dataset – often geared towards specific kinds of queries/statistics –, have been introduced for this purpose. In the database literature, these methods are known as approximate query processing techniques.

[1] Sublinear with respect to the dataset dimensions.

3.1.1 Approximate estimation

Approximate query processing methods can take different forms, but always share some common ideas. Firstly, they rely on stochastic quantities, i.e. the sketch computation involves randomness or hash functions[2]. Then, they provide only approximate answers: in the case of the estimation of a numerical quantity[3], a sketching method will typically provide an approximation which is, with probability $1-\delta$, within a factor $[1-\varepsilon, 1+\varepsilon]$ of the quantity to be estimated, where $\delta, \varepsilon > 0$ are ideally both as small as possible. We will use these notations in the next paragraphs. Finally, the size of the sketch is often a parameter, and theoretical guarantees typically provide a lower bound on it to obtain ε-approximations with probability $1-\delta$ for ε, δ given.

The synopsis, i.e. the sketch[4], is often designed such that it can be updated when new data samples arrive. Most often, the data samples are all processed in the same manner, i.e. in a way which is independent

[2] In the following, we call a hash function a function which maps its entries to a finite-dimensional space (typically a string of bits) while minimizing the risk of collisions, i.e. the fact that two different inputs will produce the same hash. A hash function can furthermore be efficiently evaluated.

[3] For a binary query, one would simply bound the probability of returning a wrong answer.

[4] The word "sketch" takes here a broader meaning compared to Chapter 2. It simply refers to a small summary of the data, but is not necessarily an empirical average of some feature function.

of previous samples. Most importantly, it is useful in practice to design sketches that can be merged after being computed, so that the sketch of a collection can be computed by defining a partition of the samples, sketching each group of samples independently, and then merging the sketches obtained that way.

The celebrated Bloom filter [115], proposed in 1970, satisfies this property. This compact data structure allows, by combining multiple hash functions, to approximately keep a track of elements that have already been seen in a stream. It will never produce false negatives[5], and the probability of having false positives can be carefully controlled.

[5] The bloom filter is always correct when it reports that an element has never been seen.

3.1.2 Linear sketches for frequency moments

For several applications, it makes sense to represent the data stream as a single vector $\mathbf{c}$ (never explicitly stored) initialized at zero, each update then consisting in increasing one entry of this vector. This is particularly useful to count occurrences of categorical quantities in a domain of known cardinality[6], the vector being then just a collection of counters, and each update adding $+1$ to the relevant counter. Linear sketches [116, Section 5], which are computed from such a vector via a linear transformation, can be merged by a simple addition. Since their introduction, they have been used for a wide range of applications, and production-quality libraries exist[7].

[6] Otherwise, a first hashing operation can be used to get back to a finite-dimensional domain of controlled cardinality.

[7] e.g. `https://datasketches.apache.org/`.

A seminal example is the Flajolet-Martin sketch [117] which can be used to count the number of distinct elements in the stream (i.e. compute $\|\mathbf{c}\|_0$), and which has been later refined [118, 119]. It relies on a binary array B of b bits, all initialized to zero, and on a hash function f taking values in $[0, 2^b - 1]$. For each sample element in the collection, the algorithm computes the position (between 1 and b) of the least significant bit of the hash of the element which takes the value 1, and sets to 1 the corresponding bit of B. Hence if f is uniform for the data at hand, B should contain almost only ones on the lowest bits, and almost only zeros on the highest bits, with a phase transition located around $\log(\|\mathbf{c}\|_0)$.

Another well-known example is the AMS[8] sketch [120], which is obtained by multiplying $\mathbf{c}$ by a random matrix with i.i.d. Rademacher[9] entries. Initially introduced to approximate moments of the vector $\mathbf{c}$, i.e. quantities of the form $F_k = \sum_{i=1}^{l} c_i^k$ for some $k \in \mathbb{N}$ (and where l denotes the dimension of $\mathbf{c}$), this sketch has proved to be very useful to solve approximately various related queries[10]. The size of this matrix can be chosen to yield a sketch of size $\Theta(\varepsilon^{-2} \log(1/\delta))$.

[8] Following the name of the authors, N. Alon, Y. Matias and M. Szegedy.

[9] i.e. taking values ± 1 with probability $\frac{1}{2}$ each.

[10] Such as estimating join sizes in databases, i.e. inner products between vector of counts [121].

Count-min sketch [122] has later been introduced to estimate various quantities on $\mathbf{c}$, but especially provide accurate estimations for the largest entries, i.e. allow to detect the elements which appear a lot in a stream, known as the heavy hitters. The sketch size here also scales with ε in $\Theta(\varepsilon^{-2})$.

Other constructions and variants have been proposed. For instance, sketches which make use of random matrices most of the time have equivalents relying on structured transforms to reduce the space com-

plexity. We refer to the work of Cormode [116, 123] for a more extensive overview.

3.2 Dimensionality reduction techniques

Dimensionality reduction, i.e. the fact of mapping all the samples $(\mathbf{x}_i)_{1\leq i\leq n}$ to a subspace of smaller dimension, has proved to be another valuable tool to efficiently run standard learning framework on large collections. The applicability of this method naturally depends on the task at hand, but in many contexts using a dimensionality-reducing operator which e.g. approximately preserves distances between data points might be sufficient to solve the learning task.

It can be used alone when the dimension d is large but the number of samples n is reasonable, as an alternative to streaming algorithms or methods which explicitly try to reduce the number of samples which will be presented in Section 3.3. However, when both d and n are large, it could be meaningful to reduce the dimensionality as a preprocessing step, and then still reduce the number of samples or rely on a streaming algorithm.

We stress that applications of dimensionality-reduction techniques are not limited to learning and also include for instance exploration, denoising or visualization [124].

3.2.1 Data-agnostic approaches

A seminal result is the Johnson-Lindenstrauss (JL) lemma, which states that it is possible to embed a set of n points into a lower-dimensional space of dimension scaling in $\log(n)$ – and independent of the initial dimension – while approximately preserving its geometry, as depicted in Figure 3.2.

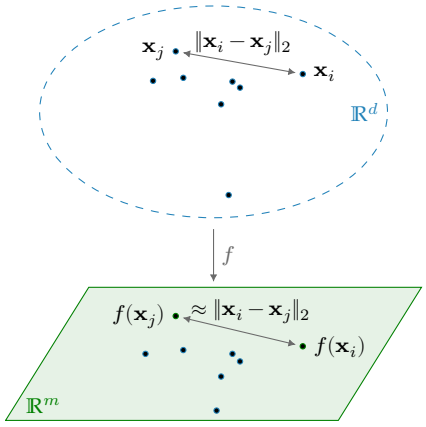


Figure 3.2: For m large enough, random embeddings behave like approximate isometries: distances between points are approximately preserved.

Lemma 3.1 (Johnson-Lindenstrauss [125, Lemma 1])

For any collection $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subseteq \mathbb{R}^d$ of n points (here seen as a set rather than a matrix) and $\varepsilon \in]0, \frac{1}{2}[$, there exists a map $f : \mathbf{X} \to \mathbb{R}^m$ such that f satisfies a restricted isometry property[11] on $\mathbf{X} - \mathbf{X}$ with constant ε provided $m \gtrsim \varepsilon^{-2} \log(n)$.

Although the lemma is often simply formulated as an existence result, it turns out that f can in practice be chosen as a linear random operator. Multiple proofs of the result exist, with explicit and different constructions of f. Dasgupta and Gupta addressed the case where f is the application of Gaussian matrix [126], while the original proof of Johnson and Lindestrauss uses a (scaled) orthogonal projection onto a random m-dimensional subspace[12] [125]. In the later case, we also refer to [127, Section 5.3] for a concise proof relying on the concentration of Lipschitz functions on the sphere and rotation invariance arguments. The results holds more generally for matrices with sub-Gaussian entries [128, 129] or structured constructions [130], which have the benefit of computational efficiency [130]. We will discuss

11. Cf. Definition 2.1, which we implicitly extend to non-linear operators. Stated otherwise, for any $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$, $(1-\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$.

12. We mean here a subspace uniformly drawn in the Grassmanian $\mathrm{Gr}(m, \mathbb{R}^d)$.

more extensively the design of random structured matrices in Chapter 5. The size $m \gtrsim \varepsilon^{-2}\log(n)$ was shown to be optimal, first for the setting where f is further required to be linear [131] and more recently for arbitrary transformations [132].

The idea of using data-agnostic dimensionality reduction to learn efficiently from the reduced embeddings has been successfully used for many tasks, such as SVM [133] or learning union of subspaces [134, 135]. We also note that any matrix distribution producing efficient Johnson-Lindenstrauss embeddings will also generate matrices satisfying the restricted isometry property for sparse vectors [47]. In the other direction, a matrix satisfying a restricted isometry property defines optimal (in dimension) Johnson-Lindenstrauss embeddings when randomly flipping the signs of its columns [136].

Hashing We focused so far on the JL result because it is fundamental tool, but other approaches have been considered as well. Some works achieve for instance an approximate isometry with respect to a kernel-norm rather than in l_2-norm. Random Fourier features (cf. Section 2.3.2) can naturally be used to approximate kernels, but the resulting features can be further reduced to small binary codes while approximately preserving the geometry[13] between all pairs of points [137]. Note that the quantized sketches of Schellekens et al. mentioned in Chapter 2 are closely[14] related to this approach [104], although the features are then further averaged.

Locality-sensitive hashing [138] and related hashing techniques have also been extensively studied as a way to provide compact embeddings for large-scale similarity search. The hashing functions are however chosen in this case to favor collisions (i.e. produce close embeddings) for data samples which are neighbors, but do not necessarily come with guarantees for points which are distant from each other. Product quantization techniques [139] are also a standard alternative to produce compact codes, and tend to scale better with the dimension compared to hashing-based techniques.

All these methods relying on quantization differ from Johnson-Lindenstrauss type embeddings by the fact that they are not linear, but they most often can still be computed efficiently and are thus used extensively for applications going beyond nearest neighbors search.

[13] In terms of Hamming distance for the binary codes.

[14] Apart from the averaging operation, the work of Raginsky et al. differs from [104] in the fact that quantizations are performed according to randomly chosen thresholds.

3.2.2 Adaptive approaches

Another way to perform dimensionality reduction is to use the data itself in order to decide how to reduce it; such methods are said to be adaptive, or data-dependent.

The most standard example is certainly principal component analysis (PCA) [140]. We introduced PCA as a learning task on its own via its loss function in Definition 1.1, but naturally the optimal k-dimensional subspace (for any $k > 0$) for this loss can be used to define k-dimensional features, which can in turn be used to solve various learning tasks with a reduced computational complexity. In this

regard, PCA will return the subspace maximizing the variance of the (projected) data. Hence when data is centered, and provided that all the components have similar scales, the associated features are a natural choice in many contexts. Multiple extensions of the method exist, such as probabilistic PCA [141] which accounts for more generic probabilistic models, robust PCA [142], or sparse PCA [143] where the subspace basis is further required to be sparse. Performing PCA in a feature space associated to a kernel is known as kernel PCA [144] and is also widely used.

Other standard approaches include canonical correlations analysis (CCA) [145], to linearly reduce two sets of variables while maximizing the correlation between the produced features, or linear discriminant analysis when measurements come with labels and are used to maximize separation between classes. Independent component analysis [146] is also commonly used to maximize statistical independence between the produced features, and in some contexts can be meaningful for dimensionality reduction. We refer to the work of Cunningham et al. for a more comprehensive survey of linear[15] methods [147].

[15] Here and in the following sections, we mean by linear dimensionality reduction that the new features are obtained from the original data via a linear transformation.

Feature selection The methods previously mentioned aim at producing $m \ll d$ new features, but another way to proceed would be to select m of the existing features. Uniform (non-adaptive) sampling can be used but will often be inefficient. Various adaptive sampling strategies have thus been developed to keep the most "informative" features. The idea of adaptive sampling can also be applied to sample the data points rather than the features – i.e. sampling columns rather than rows of the data matrix –, and we thus postpone to Section 3.3.2 the presentation of the method.

Many other deterministic approaches exist to select the optimal features for the task at hand. We refer the interested reader to the survey of Chandrashekar and Sahin [148]. Note that albeit selecting the chosen features is a linear operation, choosing which features to keep can require more complex and non-linear operations.

Non-linear techniques We mainly focused on linear dimensionality reduction techniques so far, but non-linear adaptive methods have been developed as well. In particular, many techniques have been proposed to learn (nonlinear) hashing functions from the dataset. We refer to the recent survey of Wang et al. for this category of methods [149].

Autoencoders have also emerged as a natural way to perform non-linear adaptive dimensionality reduction [150]. They are deep neural networks which have input and output layers of similar dimensions, and intermediate layers of smaller dimensions, with typically one "bottleneck" layer which is much smaller than the input dimension. By training such a network to approximate the identity function, one can use after training the features corresponding to the bottleneck layer as compact features of reduced dimensionality. These models are much more expressive compared to linear methods, but incur a large training cost and require large training collections.

3.3 Reduction of the number of samples

We discussed so far sketching approaches, which compress the whole dataset into a small summary, and dimensionality-reduction techniques which only make it easier to work with high-dimensional data but leave the number of samples in the collection intact. In this section we focus on coresets, which are small (weighted) subsets of the data collection which approximate the risk of the whole dataset.

Coresets are often simple to implement and to work with, but their design relies on various mathematical tools, and going through all the different approaches is out of the scope of this chapter. Moreover, many different definitions of a coreset coexist in the literature, which can be confusing. We focus here on a simple introduction and definition in Section 3.3.1, and briefly discuss in Section 3.3.2 how subsampling the collection can produce proper coresets.

We refer the interested reader to the recent works of Feldman [151], Phillips [152], and Bachem et al. [153] for more comprehensive overviews, and also to the work of Munteanu and Schwiegelshohn [154] for a presentation of the different tools used for coreset design. Another didactic introduction can be found in [155].

3.3.1 Coresets

Many variants exist for the definition of a coreset. If we focus on the context of statistical learning, where we recall[16] that a learning task is defined by a risk function $\mathcal{R} : \mathcal{H} \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$, a coreset of a collection $\mathbf{X}$ is broadly speaking a subset of the data which uniformly approximates the risk associated to $\mathbf{X}$ over $\mathcal{H}$.

[16] Cf. Section 1.1.1. We recall that $\mathcal{H}$ denotes the hypothesis space.

Definition 3.2 (Coreset): Let $\mathbf{X}$ be a dataset with associated empirical distribution $\pi_{\mathbf{X}}$. A (strong) (r, ε)-coreset of $\mathbf{X}$ for the risk function $\mathcal{R}$ is a subset $\mathbf{S} = \{\mathbf{s}_1, ..., \mathbf{s}_r\}$ of r points of $\mathbf{X}$ and weights $(\alpha_1, ..., \alpha_r)$ in the probability simplex satisfying

$$\exists C > 0, \forall h \in \mathcal{H},\ (1-\varepsilon)\mathcal{R}(h, \pi_{\mathbf{X}}) \leq C\mathcal{R}(h, \pi_{\mathbf{S}}) \leq (1+\varepsilon)\mathcal{R}(h, \pi_{\mathbf{X}}), \tag{3.1}$$

where $\pi_{\mathbf{S}} \triangleq \sum_{i=1}^{r} \alpha_i \delta_{\mathbf{s}_i}$.

Note that the weights $\alpha_1, ..., \alpha_r$ are constrained in the probability simplex only in order to define a probability distribution and hence to fit with our definition of the risk, but the presence of the normalization constant C actually means that arbitrary non-negative weights can be used – although one will typically have $C = 1$ for homogeneity reasons.

We list here the most common variants of this definition that can be found in the literature.

1. The weights $\alpha_1, ..., \alpha_r$ are sometimes imposed to be $\alpha_1 = ... = \alpha_r = \frac{1}{r}$.
2. The above definition is strong in the sense that approximation must be uniform over $\mathcal{H}$. A construction which only satisfies the

inequalities in (3.1) on a subset of $\mathcal{H}$ (for instance around the risk minimizer) is called a *weak coreset*.

3. In some papers, $\mathbf{S}$ must simply be a set of r points, but these should not necessarily belong to $\mathbf{X}$. In the following, we refer to such constructions as *generalized*[17] *coresets*.

[17] No particular naming convention exists for this setting in the literature.

A desirable property for coresets is to be composable, which means that if $\mathbf{S}_1, \mathbf{S}_2$ are respectively coresets of two datasets $\mathbf{X}_1, \mathbf{X}_2$, then $\mathbf{S}_1 \cup \mathbf{S}_2$ is also a coreset for $\mathbf{X}_1 \cup \mathbf{X}_2$. This often means that it is possible to start from $\mathbf{S}_1 \cup \mathbf{S}_2$ to produce a further reduced coreset $\mathbf{S}_3 \subseteq \mathbf{S}_1 \cup \mathbf{S}_2$. See for instance [156] for an application on various diversity-maximization tasks, i.e. tasks where a given metric between points in the coreset is maximized.

Note that this is highly reminiscent of the need to produce sketches that can be merged after computation discussed in Section 3.1. This should come at no surprise as this property is the key to handle data streams and for distributed learning.

Cardinality of the coreset It should be noted that some authors also use the word coresets to denote collections of dimensionality-reduced features, even when the number of samples is the same as the initial collection[18] [151, Section 4.1]. We prefer to make a clear distinction between the two approaches when possible, hence r is always assumed to grow sub-linearly in n in this section.

[18] In this case, Definition 3.2 must be adapted as the risk is defined on $\mathcal{P}(\mathcal{X})$.

A natural goal is to find coresets which are as small as possible. The size r will typically grow polynomially with $1/\varepsilon$. Some constructions are probabilistic and hold only with probability $1 - \delta$, in which case r usually also scales polynomially with $\log(1/\delta)$.

Construction Definition 3.2 is very generic, and multiple methods can be used to derive coresets in practice. Randomly subsampling the collection is the simplest and most common way to reduce the number of samples, but uniform sampling will not always yield a proper coreset in terms of approximation and more subtle sampling schemes are required; we discuss this approach in Section 3.3.2. Linear sketching approaches – i.e. multiplying $\mathbf{X}$ by a dense random matrix on the right – have been considered and compared to subsampling schemes, see e.g. for least-squares regression the work of Raskutti and Moheney [157]. They are computationally efficient, but however can at best yield generalized coresets by definition.

Histograms and other covering methods of the input space have been used extensively – each point of the coreset representing for instance one histogram bin –, but we do not review such approaches here as they scale poorly[19] with the dimension and hence are of limited utility for high-dimensional learning. Note however that most of the early papers on coresets relied on such constructions, see e.g. [158].

[19] i.e. exponentially.

Deterministic constructions also exist. In some settings, for instance in computational geometry for the computation of the minimum enclosing ball or other closely related problems, coresets can be derived

from convex optimization techniques [154, Section 3.2]. Points are then added to the coreset one by one. Connections of such greedy approaches with the Frank-Wolfe algorithm are discussed in [159]. Kernel herding approaches [160] for kernel density estimation can also be interpreted in this context [161, 162].

For high-dimensional clustering, efficient streaming deterministic algorithms have been proposed [163, 164], but often combine ideas from deterministic coresets literature with dimensionality reduction methods – e.g., the intermediate representation which is produced is not in the input space, and hence does not satisfy Definition 3.2 although it can be used to approximate the risk. When data is sparse, preserving high-dimensional but sparse vectors in the input space might be more desirable than using dimensionality-reduction techniques which will produce dense features. Dedicated deterministic solutions have been proposed for this setting (assuming $d \geq n$), for instance for k-means [165] and PCA [166].

Limitations It should be mentioned that Definition 3.2 is a broad definition, and coresets of small sizes may simply not exist for some problems. Impossibility results have for instance been provided for the 2-slab[20] problem [167], or in another context for Poisson dependency networks [168]. Even when existence can be proved, explicit constructions are sometimes too expensive to be useful in practice.

[20] A slab refers in this context to the portion of the space comprised between two parallel hyperplanes. The 2-slab problems consists in covering the data points by a union of two slabs of minimum width.

3.3.2 Subsampling

Randomly subsampling a dataset is the most intuitive way to reduce the number of samples in it. In some contexts, subsampling can produce sets of points which are proper coresets according to Definition 3.2.

Uniform sampling Let $\mathbf{S}$ be a set of r points drawn i.i.d. uniformly in the collection $\mathbf{X}$, with weights $\alpha_1 = ... = \alpha_r = 1/r$, and let $\pi_{\mathbf{S}} \triangleq \sum_{i=1}^{r} \alpha_i \delta_{\mathbf{s}_i}$ denote the associated distribution. Thus for any $h \in \mathcal{H}$, $\mathcal{R}(h, \pi_{\mathbf{S}})$ is an unbiased estimator of $\mathcal{R}(h, \pi_{\mathbf{X}})$ and its variance can be roughly bounded as follows

$$\mathrm{Var}(\mathcal{R}(h, \pi_{\mathbf{S}})) = \frac{1}{r}\mathrm{Var}_{\mathbf{x}\sim\mathcal{U}(\mathbf{X})}(l(h, \mathbf{x})) \leq \frac{1}{r}\mathbf{E}(l(h, \mathbf{x})^2) \leq \frac{n}{r}\mathcal{R}(h, \mathbf{X})^2.$$

where l denotes the loss associated to the risk function $\mathcal{R}$. To derive a sufficient condition on the cardinality of $\mathbf{S}$ in order to obtain a coreset, we can apply Chebyshev's inequality as follows

$$\begin{aligned}
&\mathbb{P}[|\mathcal{R}(h, \pi_{\mathbf{S}}) - \mathcal{R}(h, \pi_{\mathbf{X}})| > \varepsilon \mathcal{R}(h, \pi_{\mathbf{X}})] \\
&\leq \mathbb{P}\left[|\mathcal{R}(h, \pi_{\mathbf{S}}) - \mathcal{R}(h, \pi_{\mathbf{X}})| > \varepsilon \sqrt{\frac{r}{n}} \sqrt{\mathrm{Var}(\mathcal{R}(h, \pi_{\mathbf{S}}))}\right] \leq \frac{n}{r\varepsilon^2}.
\end{aligned}$$

With this approach, it is thus sufficient to choose $r \gtrsim n/(\varepsilon^2\delta)$ to obtain a valid coreset with probability $1 - \delta$, which is of limited utility as we recall that our goal is to obtain a coreset size which is sub-linear in n (and ideally independent of n). Although this analysis might look

simple, one can actually show for tasks such as k-means that having r on the order of n is indeed necessary to obtain a proper coreset via uniform subsampling[21]. This is typically the case when a single point is responsible for the largest part of the risk for some $h \in \mathcal{H}$. Taking into account the contribution of each point to the risk thus helps to reduce the required coreset size, yielding the so-called "importance sampling" strategy.

[21] At least without additional assumptions on the data distribution, one can manually build counter-examples. See for instance [153].

Importance sampling The idea of using non-uniform random subsampling in coreset design was introduced by Chen [169] for k-means and k-medians problems, yielding a coreset size polynomial in d. However, the key idea to reduce the coreset size is the notion of sensitivity introduced by Langberg and Schulman [170]. In this context[22], and for statistical learning applications, the sensitivity of a point $\mathbf{x} \in \mathbf{X}$ for a dataset $\mathbf{X}$ with empirical distribution $\pi_{\mathbf{X}}$ can be defined as

$$s(\mathbf{x}) \triangleq \sup_{h \in \mathcal{H}} \frac{\frac{1}{n} l(h, \mathbf{x})}{\mathcal{R}(h, \pi_{\mathbf{X}})}. \tag{3.2}$$

[22] We will see in Part IV that the word "sensitivity" has a different (albeit related) meaning in the context of privacy preservation.

It captures the proportion of the risk which is due to $\mathbf{x}$ (in the worst case on $h \in \mathcal{H}$), and thus indicates how important it is to keep $\mathbf{x}$ if one wants to preserve the overall risk when subsampling. The idea of Langberg and Schulman is hence to favor the samples which have a high sensitivity when subsampling. Naturally, computing exactly the sensitivities of all the samples can already be expensive. We now introduce the notion of importance sampling, and we will see just below that only computing an upper bound of the sensitivities can already be helpful.

Importance sampling is a generic method which is commonly used in order to reduce the variance of a Monte-Carlo estimator. For a given probability distribution p over a domain $\mathcal{D}$ and function f defined on the same domain, the classical Monte-Carlo method suggests that the integral $F = \int_{\mathcal{D}} f(x)p(x)\,\mathrm{d}x$ can be estimated by $\frac{1}{n}\sum_{i=1}^{n} f(x_i)$, where $x_1, ..., x_n \overset{i.i.d.}{\sim} p$. Importance sampling relies on the observation that, if $x_1, ..., x_n$ are samples drawn i.i.d. according to a different probability distribution q, the quantity $\frac{1}{n}\sum_{i=1}^{n} w(x_i)f(x_i)$ where $w(x_i) = \frac{p(x_i)}{q(x_i)}$ is still an unbiased estimator of F. This tweak can be leveraged when one has access to $f(x_1), ..., f(x_n)$ without being able to control the distribution of the samples $x_1, ..., x_n$. However, even when one is free to sample f as desired, it can be useful to choose on purpose a distribution $q \neq p$ to modify the properties of the obtained estimator, and in particular to reduce its variance.

In our context, one can show that for any fixed h, sampling each point $\mathbf{x}_i$ with a probability $q(\mathbf{x}_i) \overset{\triangle}{\propto} \frac{\frac{1}{n} l(h,\mathbf{x})}{\mathcal{R}(h,\pi_{\mathbf{X}})}$ (and properly reweighting the samples) indeed minimizes the variance of $\mathcal{R}(h, \pi_{\mathbf{S}})$. However, as we want (3.1) to hold uniformly, a workaround is to use the worst case on $h \in \mathcal{H}$, which yields precisely the definition of sensitivity s given in (3.2). It can then be shown [171][23] that, for any upper bound $\bar{s}$ on the sensitivity (i.e. such that $\forall \mathbf{x} \in \mathbf{X}\ \bar{s}(\mathbf{x}) \geq s(\mathbf{x})$), defining $\bar{S} \triangleq$

[23] See also [153, Section 2.5] for a didactic derivation.

$\sum_{\mathbf{x}\in\mathbf{X}} \bar{s}(\mathbf{x})$, the weighted subset obtained when performing importance sampling with probability $q(\mathbf{x}_i) \triangleq \bar{s}(\mathbf{x}_i)/\bar{S}$ yields for $\delta > 0$ a proper (r, ε)-coreset with probability $1 - \delta$ provided that

$$r \gtrsim \frac{\bar{S}^2}{\varepsilon^2}\left(D + \log\left(\frac{2}{\delta}\right)\right). \tag{3.3}$$

where D is the pseudo-dimension of the class of weighted losses $\{v(h)l(h,\cdot)/q(\cdot)|h \in \mathcal{H}\}$ for some weighting function v, i.e. a measure of the complexity of this class of functions. We refer the reader for instance to [172, Chapter 4] for a precise definition of the pseudo-dimension, and its connection with the Vapnik-Chervonenkis (VC) dimension, that it generalizes. Note that, this result being established, finding a good upper-bound for the sensitivity is often the most difficult step.

Often, obtaining a good sampling strategy already requires to have a reasonable estimation of the solution or of the data density. This chicken-and-egg problem can be addressed via an intermediate subset construction[24], which is larger in size and weaker in guarantees compared to a proper coreset, but which is sufficient for importance sampling.

With this approach, a coreset of size $r \gtrsim dk^2/\varepsilon^2$ was for instance obtained for k-means [170]. The notion of sensitivity was later extended by Feldman and Langberg [173] for various tasks such as k-medians or subspace clustering. Other applications of this method include dictionary learning [174], Gaussian modeling [175] and l_p regression [176][25].

Note that although some of these constructions can be adapted to the streaming scenario, the generalization is not obvious. Coreset sizes often vary depending on the considered coreset definition and setting (e.g. coreset / generalized coreset / streaming), and providing a summary of the different existing contributions and coreset sizes is out of the scope of this chapter. We refer for instance to [151, Table 1] for a list of coresets for the k-means clustering problem.

[24] A good example of that is (α, β)-bicriteria approximation, where one produces a subset which is β times larger than desired, and gives a risk for the optimal hypothesis which is in a factor α of the true optimal risk.

[25] This work is also based on sampling but does not rely on the sensitivity.

Leverage scores We introduced above a sampling distribution which directly depends on the loss function, but other variants exist in the literature. For instance, Tremblay et al. showed that coresets can be obtained by subsampling the data points using determinantal point processes [177], which promote diversity in the set of selected points. We do not dwell on this idea here, but rather introduce statistical leverage scores, which are another standard tool which has been used extensively for adaptive sampling. Assuming $\mathbf{X}\mathbf{X}^T$ is invertible, the leverage scores are defined as the diagonal entries of the matrix $\mathbf{H} \triangleq \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}$. This matrix has a nice interpretation for the linear least-squares regression problem (cf. Definition 1.4), where one aims at minimizing $\|\mathbf{X}^T\mathbf{h} - \mathbf{y}\|_2$ over $\mathbf{h} \in \mathbb{R}^d$ for fixed $\mathbf{X} \in \mathbb{R}^{d\times n}$ and $\mathbf{y} \in \mathbb{R}^n$. Indeed, the solution of this problem is $\mathbf{h}^* = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}$, and thus the predicted labels $\mathbf{y}^* \triangleq \mathbf{X}^T\mathbf{h}^*$ satisfy $\mathbf{y}^* = \mathbf{H}\mathbf{y}$. Hence the entry (i, j) of $\mathbf{H}$ characterizes the impact of $\mathbf{y}_j$ on the prediction $\mathbf{y}_i^*$. The role of leverage scores to detect useful data points – and potentially outliers –,

in particular for least squares regression, is well known and has been identified a long time ago [178].

As $\mathbf{H}$ corresponds to the projection matrix on the row space of $\mathbf{X}$, it turns out that the leverage scores can be computed from any basis of the row space of $\mathbf{X}$. For instance, if $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is a SVD decomposition of $\mathbf{X}$, then $\mathbf{H} = \mathbf{V}\mathbf{V}^T$ and the leverage scores correspond to the squared l_2-norms of the rows of $\mathbf{V}$.

Computing the leverage scores can be a challenge on its own, as extracting a basis of the row space already costs an SVD or QR decomposition. We will see in Section 3.4 that randomized approximate methods exist for these decompositions; such ideas have been used to reduce the computational complexity of estimating the leverage scores [179] from $\Theta(nd^2)$ to $\Theta(nd\log(d))$ – assuming here $n \gg d$.

Note that, although we used the linear regression problem to provide some intuition, the leverage scores are independent of $\mathbf{y}$ whereas the sensitivity defined above depends explicitly on the loss function, which itself is a function of $\mathbf{y}$.

It is also interesting to note that multiplication of $\mathbf{X}^T$ (on the left) by an orthogonal matrix[26], tends to make the leverage scores of the resulting matrix more uniform. This can be used as a preprocessing step, and one can then subsample the columns of the resulting matrix uniformly, without computing any leverage score [180]. We refer the interested reader to [181] for more insights on leverage scores in general, and in particular to [181, Section 4.4.1] for this idea.

[26] For instance with a structured $\mathbf{HD}$ block, where $\mathbf{H}$ is a Hadamard matrix (see Chapter 5) and D a diagonal matrix with uniform i.i.d. ± 1 entries.

3.4 Randomized linear algebra

We tried so far to make a clear distinction between dimensionality-reduction techniques (Section 3.2) and coreset methods (Section 3.3), which come with different interpretations. In practice however, the tools on which they rely are similar as both approaches compute a "sketch" which can be expressed for some matrix $\mathbf{S}$ as $\mathbf{SX}$ in the first case, and $\mathbf{XS}$ for coreset methods. Furthermore, in both cases $\mathbf{S}$ can be either dense (e.g. with normal entries), or correspond to a subsampling operator.

In this section, we explain how these sketching techniques can be combined to perform standard linear algebra tasks. Indeed, many tasks in linear algebra are ubiquitous and are used as building blocks to solve other problems. We focus on the low-rank approximation problem which has many useful applications – such as the approximation of kernel matrices –, but which can also be interpreted as a learning task itself.

3.4.1 Randomized low-rank factorization

A fundamental problem in linear algebra is to compute the singular value decomposition (SVD) of a matrix. This decomposition is widely used, and can for instance be leveraged to solve the PCA problem (Definition 1.1) – assuming centered data. In the last two decades,

randomization has proved to be useful in this context as well. We provide in this section some considerations on this particular problem, and refer the reader to the works of Martinsson [182] and Kannan and Vempala [183] for up-to-date overviews of other standard tools and problems[27] in randomized linear algebra.

[27] For instance, regression or matrix multiplication are also standard tasks which can be randomized.

Randomized SVD We denote again $\mathbf{X}$ the $d \times n$ data matrix, which can here contain complex entries (we thus use $\cdot^*$ to denote the conjugate transpose). Halko et al. proved in a seminal work [184] that a sketch of the form $\mathbf{Y} = \mathbf{X}\mathbf{\Omega}$, where $\mathbf{\Omega}$ has for instance i.i.d. normal entries, can be sufficient to obtain a good approximation of the action of $\mathbf{X}$. Here the matrix $\mathbf{\Omega}$ is chosen of size $n \times l$, with l being much smaller than n. That is, if we recover a matrix $\mathbf{Q}$ whose columns form an orthonormal basis for the range of $\mathbf{Y}$ – this can be done via a QR factorization of $\mathbf{Y}$ –, and choose the number l of columns of $\mathbf{\Omega}$ to be slightly larger than a target rank k, then with high probability

$$\|\mathbf{X} - \mathbf{Q}\mathbf{Q}^*\mathbf{X}\| \approx \min_{\mathbf{Z} \text{ s.t. } \operatorname{rank}(\mathbf{Z})=k} \|\mathbf{X} - \mathbf{Z}\|. \tag{3.4}$$

This fact is related to the observations made on random approximate isometries in Section 3.2.1, but is used here in a slightly different matter. The error in (3.4) is written in spectral norm but could also be measured in Frobenius norm, in which case the quantity $\|\mathbf{X} - \mathbf{Q}\mathbf{Q}^*\mathbf{X}\|_F^2$ matches the PCA risk $\mathcal{R}(\text{range}(\mathbf{Q}), \mathbf{X})$, where $\text{range}(\mathbf{Q})$ denotes the subspace spanned by the columns of $\mathbf{Q}$. Although results formulated with respect to the spectral norm tend to be more meaningful[28], both kind of bounds appear in the literature.

[28] In this regard, see [182, Remark 2.1].

An approximate singular value decomposition $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ of $\mathbf{X}$ can then be recovered from $\mathbf{Q}$, for instance by computing the SVD $\tilde{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^*$ of $\mathbf{Q}^*\mathbf{X}$, and then defining $\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}$. Given that this procedure is entirely deterministic – once $\mathbf{Q}$ is computed –, the approximation error satisfies

$$\|\mathbf{X} - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*\| = \|\mathbf{X} - \mathbf{Q}\tilde{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^*\| = \|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{X}\|,$$

i.e. only depends on the estimation of the range of $\mathbf{X}$, and thus on how close the approximation in (3.4) holds. Halko et al. provide precise bounds according to the optimal error[29], first in a deterministic setting [184, Theorem 9.1] and then for a Gaussian matrix $\mathbf{\Omega}$. In the latter case, it is shown that

$$\|(\mathbf{I}_d - \mathbf{Q}\mathbf{Q}^*)\mathbf{X}\| \leq C_1(l,k)\sigma_{k+1} + C_2(l,k)(\sum_{i>k} \sigma_i^2)^{1/2}$$

with a failure probability decaying exponentially with the oversampling parameter $l - k$, where C_1, C_2 are functions of only l and k, and where $C_2(l,k)$ also decays quickly with $l - k$. Hence using a sketch size of the kind $l \approx 2k$ will already ensure an error which is within a small constant factor of the optimal error.

[29] If $\sigma_1 \geq \sigma_2 \geq \ldots$ denote the singular value of $\mathbf{X}$, then the optimal error for a rank-k approximation is σ_{k+1} in spectral norm and $\sum_{i>k} \sigma_i^2$ in squared Frobenius norm according to the Eckart-Young-Mirsky theorem [185].

This yields an algorithm of complexity $\Theta(dnl + l^2(d+n))$ instead of $\Theta(\min(dn^2, d^2n))$ for a deterministic SVD. Structured operators can

naturally be used here as well to reduce the computational complexity of the linear operation – the exact complexity then depends of the chosen construction, but this will typically reduce the cost of the linear operation from nld down to $nl\log(d)$.

Single-pass approaches This randomized approach to SVD computation can be decomposed in two steps: first finding an appropriate basis $\mathbf{Q}$, and then using $\mathbf{Q}^*$ to reduce the dimension and solve a smaller problem. It could then fit in the category of dimensionality-reduction techniques, via a sketch of the column space. When n is large, computing the sketch $\mathbf{Y} = \mathbf{X}\mathbf{\Omega}$ is reasonable, but accessing a second time the data to compute the matrix $\mathbf{Q}^*\mathbf{X}$ might be too expensive, or simply not possible when $\mathbf{X}$ is provided in a streaming setting[30]. This is however necessary in the general case[31] as the sketch $\mathbf{Y}$ only contains information about the column range.

Note that if one only wants to solve the PCA problem as defined in Definition 1.1 – i.e., find the optimal subspace and not the weights associated to each sample – the sketch $\mathbf{Y}$ already contains the useful information to approximately solve the problem. Assuming a full decomposition is required, one way to avoid accessing the data multiple times is to sketch both rows and columns spaces simultaneously, i.e. compute two sketches $\mathbf{Y} = \mathbf{X}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{\Psi}\mathbf{X}$ where both $\mathbf{\Omega}$ and $\mathbf{\Psi}$ are random. This approach is already discussed in [184, Section 5.5], and an in-depth analysis has later been proposed by Tropp et al. [186].

Boutsidis et al. also considered combining the quantities $\mathbf{Y}, \mathbf{Z}$ with a third sketch of the form $\mathbf{\Xi}\mathbf{X}\mathbf{\Gamma}$ where both $\mathbf{\Xi}, \mathbf{\Gamma}$ are random [187], yielding algorithms with an optimal space complexity – in regard of the bounds provided by Clarkson and Woodruff [188].

Sampling It may sometimes be more desirable to use instead of $\mathbf{Q}$ a combination of a few columns of $\mathbf{X}$, for instance to preserve some properties of the data such as sparsity. Coreset constructions based on subsampling presented in Section 3.3.2 can be leveraged in this case [166]. Subsampling both columns and rows[32] yields the family of interpolatory and CUR decompositions [189] (see also e.g. [190, Section 10 and 11]), which we do not detail here. We discuss below the case of Nyström approximation with subsampled rows and columns of a positive semi-definite matrix.

[30] We only mentioned so far the case where the sample $(\mathbf{x}_i)_{1\leq i\leq n}$ are streamed, but we can more generally consider here the stream $\mathbf{X} = \mathbf{X}_1 + \mathbf{X}_2 + \dots$ where the $\mathbf{X}_i$ are all $d \times n$ (possibly sparse) matrices. The model where each $\mathbf{X}_i$ has furthermore only one non-zero entry is referred to as the turnstile model in the literature.

[31] The setting of symmetric or positive semi-definite matrices is different as the row and column ranges then coincide, see below.

[32] Typically using leverage scores.

3.4.2 The case of positive semi-definite matrices

The ideas presented above can be adapted to preserve specific structural properties of the considered matrix, such as positive semi-definiteness. We considered so far factorizing the data matrix $\mathbf{X}$, but the method can be used in various settings, and is particularly useful to approximate large kernel matrices.

Indeed as explained in Chapter 2, most learning approaches relying on kernel functions require computing the $n \times n$ kernel matrix $\mathbf{K}$ with entries $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, where κ denotes the kernel function used.

Random features can naturally be used to approximate $\mathbf{K}$ – i.e., one can approximate $\mathbf{K}$ by a sum of rank-one random estimators –, but low-rank approximation strategies can also be leveraged, as an alternative to random features.

Positive semi-definiteness is also a natural constraint when working with covariance matrices, or Hessian matrices in optimization methods. Depending on the setting, different approximation schemes can be leveraged.

The Nyström method One way to approximate a positive semi-definite (psd) matrix $\mathbf{M}$ is to compute a Nyström approximation

$$\tilde{\mathbf{M}} = (\mathbf{M}\mathbf{\Psi})(\mathbf{\Psi}^*\mathbf{M}\mathbf{\Psi})^\dagger(\mathbf{M}\mathbf{\Psi})^* \tag{3.5}$$

where $\cdot^\dagger$ denotes the Moore-Penrose pseudoinverse, and $\mathbf{\Psi}$ an $n \times r$ matrix. The quality of the approximation can be evaluated for different choices of $\mathbf{\Psi}$, but random matrices will typically be used.

When $\mathbf{\Psi}$ has normal entries, we get a method which is similar to the procedure for randomized SVD presented in Section 3.4.1, with the difference that the row and column ranges of $\mathbf{M}$ are here identical and thus both captured by $\mathbf{Y} = \mathbf{M}\mathbf{\Psi}$ at the same time. Approximation guarantees for this scheme are closely related to the ones discussed in Section 3.4.1 [191, 192]. This approach requires having access to the full matrix $\mathbf{M}$, but can be useful for instance when working with a covariance matrix which appears as a stream of rank-one updates. It has also been used for kernel ridge regression[33] [193], and more generally when applied on a kernel matrix its effect can be interpreted in the related RKHS [194]. Variants have also been proposed for psd approximation [195] – i.e. the output matrix is psd low-rank, but the input matrix must not necessarily be exactly psd.

[33] See below for a definition.

Nyström with partial evaluation When working with a kernel matrix $\mathbf{M} = \mathbf{K}$, one usually wants to avoid computing and storing the full matrix. It is then judicious to choose the columns of $\mathbf{\Psi}$ in the canonical basis, so that (3.5) can be rewritten

$$\tilde{\mathbf{K}} = \mathbf{K}_I\mathbf{K}_{I,I}^\dagger\mathbf{K}_I^* \tag{3.6}$$

for some set of indexes I of size l, where $\mathbf{K}_I$ is the $n \times l$ matrix obtained from $\mathbf{K}$ by subsampling the columns of indexes in I, and $\mathbf{K}_{I,I}$ is the $l \times l$ matrix obtained by further subsampling the rows of $\mathbf{K}_I$. The goal of this factorization is that only $\mathbf{K}_I$ and $\mathbf{K}_{I,I}$ need be stored, and not the whole approximation $\tilde{\mathbf{K}}$. Thus computing the decomposition requires nl kernel evaluations – or approximation with random features –, the $\Theta(n^2)$ space cost is avoided and the factorization can after computation be used for efficient linear algebra operations.

This approach was initially suggested[34] by Williams and Seeger [197], and then used extensively in the kernel literature. Following previous discussions regarding subsampling strategies, the set of indexes I can here again be sampled uniformly or using leverage scores[35]. Uniform

[34] We refer here to the approximation of kernel matrices, but the Nyström method is much more general and takes it name from a seminal paper of E.J. Nyström [196] (in german).

[35] Greedy methods have also been considered, but often incur higher computational costs.

sampling has initially been used [197], and sufficient sketch size l to obtain a near-optimal risk levels have been provided, e.g. for kernel k-means [198, 199] or kernel ridge regression [200]. We recall that these problems can be seen as generalizations of theirs linear equivalents (cf. Definitions 1.2 and 1.4), where the input data is pre-processed via the feature map associated to a chosen kernel. This is why the kernel matrix appears in the solution – at least for kernel ridge regression, where we have a closed form.

Ridge leverage scores The standard approach for adaptive sampling in the context of kernel methods is to compute the ridge leverage scores,

$$s(\mathbf{K}, \lambda) = \mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1}, \tag{3.7}$$

which are often expressed like here with a regularization parameter $\lambda > 0$. These scores can be seen as a natural generalization of the "standard" leverage scores introduced in Section 3.3.2, which had a natural interpretation with respect to the linear least squares problem, where one wants to minimize $\|\mathbf{X}^T\mathbf{a} - \mathbf{b}\|_2^2$ over $\mathbf{a}$. The ridge leverage scores appear when adding a ridge normalization term $\lambda\|\mathbf{a}\|_2^2$ to this objective, and when replacing the data $\mathbf{X}$ by the matrix of features $\mathbf{\Phi} = [\Phi(\mathbf{x}_1), ..., \Phi(\mathbf{x}_n)]$, where Φ denotes the feature map corresponding to the kernel κ used to compute the kernel matrix $\mathbf{K}$. Indeed, in that case the objective to minimize becomes $\|\mathbf{\Phi}^T\mathbf{a} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{a}\|_2^2$, and the solution expresses $\mathbf{a}^* = (\mathbf{\Phi}\mathbf{\Phi}^T + \lambda\mathbf{I})^{-1}\mathbf{\Phi}\mathbf{b}$, which can be rewritten via a special case of the Woodbury identity [201, eq. (158)] as $\mathbf{a}^* = \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{b}$, hence $\mathbf{b}^* = \mathbf{\Phi}^T\mathbf{a}^* = \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{b}$.

Once the ridge leverage scores have been computed, one can sample the column i with probability $s(\mathbf{K}, \lambda)_{ii} / \operatorname{tr}(s(\mathbf{K}, \lambda))$. Learning guarantees can often be obtained using ridge leverage scores using a smaller number of samples compared to uniform sampling for tasks such as kernel ridge regression [202] or kernel PCA [203].

Just as with plain leverage scores, computing the ridge leverage scores can be a challenge on its own, but efficient methods have been proposed – similarly to what has been discussed in Section 3.3.2 –, typically relying on recursive approximations [203, 204].

Note that efficient and practical frameworks often combine together multiple tools. For instance, low-rank approximation of kernel matrices can efficiently be computed by combining the Nyström method (using subsampling) with the linear sketches from Section 3.4.1 [205] to reduce the computational complexity. In another context, Rudi et al. proposed for kernel ridge regression to combine the Nyström approach with pre-conditioning techniques, which allow to reduce the number of iterations of iterative optimization algorithms [206].

We focused here on the Nyström method, but naturally random features introduced in Chapter 2 are another standard way to approximate the kernel matrix $\mathbf{K}$, and leverage scores can also be used for this purpose [207, 208].

Although we focused on approximation of kernel matrices, which play a very important role in the learning literature, it should be noted that similar techniques can be used to approximate Hessian matrices in order to speed-up second-order optimization techniques, see e.g. [209].

3.5 Conclusion

In this chapter, we presented some standard tools to alleviate the computational cost of learning from large collections. We mentioned a few deterministic techniques, but most of the considered methods belong to the category of "sketching" algorithms as they rely on computing a small sketch of the data matrix obtained by multiplication with a random matrix. We provide in Table 3.1 a categorization of these different methods (which does not include sketches from Chapter 2 nor Section 3.1).

Sketching type \ operator	**Dense S** (e.g. with i.i.d. normal entries, or structured variants)	**Subsampling S** (i.e. with columns (or rows) in the canonical basis)	
		Uniform sampling	**Importance sampling**
	← data-agnostic →		← adaptive →
Rows/features space (Sketch of the form $\mathbf{SX}$)	Random projections, JL-type embeddings cf. Section 3.2.1	Feature selection methods cf. Section 3.2.1	
Columns/samples space (Sketch of the form $\mathbf{XS}$)	Generalized coresets, cf. Section 3.4.1	Uniform subsampling, cf. Section 3.3.2	Sensitivity, Leverage scores cf. Section 3.3.2
Both	cf. single-pass approaches in Section 3.4.1		CUR decompositions Section 3.4.1
Both (Symmetric/PSD case)	Nyström as in (3.5), cf. Section 3.4.2	Nyström "with partial evaluation" cf. Section 3.4.2	Nyström with ridge leverage scores cf. Section 3.4.2

Table 3.1: Summary of the most common randomized "sketching"/subsampling techniques existing in the literature for efficient large-scale learning.

This table is not exhaustive, but shows the fact that random projections have been used in a variety of different settings in the literature. We stress that the structured random matrices which will be introduced in Chapter 5 can be (and have been) used for most methods of the first column of the table, i.e. pretty much every time dense random matrices are used. We will discuss in Part IV how these different methods compare in terms of privacy preservation, but we can already say that coresets and sketching methods have been used successfully for privacy-preserving learning; data subsampling techniques are also known to improve privacy in some contexts.

The compressive approach presented in Chapter 2 is quite different from these approaches, but still computes a small summary of the dataset. Hence in the following chapters, the word "sketch" will refer to the empirical sketch defined in (1.10) unless otherwise specified.

Part II

Efficient Compressive Learning

This part focuses on the design of the random matrix Ω which appears in the expression of the feature map. In Chapter 4, we study empirically the impact of the kernel scale (and hence the scaling of Ω) when performing compressive clustering with a Gaussian kernel. Chapter 5 shows how structured linear matrices can be leveraged to speed-up the complexity of the framework.

Chapter 4
Learning to Sketch with a Gaussian Kernel

Note Some ideas discussed in this chapter have been presented to the iTWIST workshop [25].

Contents

As explained in Chapter 2, random Fourier features[1] used for compressive clustering implicitly define, together with a frequency distribution Λ, a kernel in the data domain. For instance, drawing the frequency vectors according to the multivariate normal distribution $\Lambda = \mathcal{N}(\mathbf{0}, \frac{1}{\sigma_\kappa^2}\mathbf{I})$ for some $\sigma_\kappa^2 > 0$ yields a Gaussian kernel $\kappa(\mathbf{x}, \mathbf{y}) \triangleq \exp(-\|\mathbf{x}-\mathbf{y}\|_2^2/(2\sigma_\kappa^2))$ in the data domain as detailed in Section 2.3.2. However, we did not discuss so far how this variance σ_κ^2 should be chosen. This parameter being critical for many experiments in the thesis, we propose in this chapter to investigate the problem in detail via numerical simulations.

In the rest of the chapter, all the sketches are computed using $\Phi = \Phi^{\text{RFF}}$, and we denote $\mathcal{A}$ the induced sketching operator on probability distributions as defined in (2.10). We explain in Section 4.1 why choosing σ_κ^2 carefully is important, and discuss existing heuristics to do so. Simulations on synthetic data are performed in Section 4.2 to pin down the optimal[2] scale as a function of the characteristics of the dataset, and directions for estimation of this optimal scale from the data using sketches are discussed in Section 4.3.

[1] cf. Definition 2.5

[2] Because of the experimental nature of this chapter, "optimal scale" refers in the following to the scale which seems empirically to provide the best results.

4.1 Role of the kernel scale

The choice of the kernel scale has multiple impacts. If it is poorly chosen, then the resulting distance to the empirical sketch, which is used as the optimization criterion, will not be a good proxy for the risk (cf. Section 1.1.1). However even when the optimization problem is coherent with the risk function (e.g. has the same global minimizers), a poor choice of the scale might still induce many undesirable local minima[3] and therefore make optimization impossible in practice. We discuss these considerations in sections 4.1.1 and 4.1.2, and expose in Section 4.1.3 the heuristics that have been proposed so far to estimate a reasonable kernel scale.

[3] We will illustrate this point in Figure 4.1.

4.1.1 Theoretical insights

From now on, we denote σ_κ^2 the kernel variance. Unless otherwise specified, we compute the sketch using frequency vectors drawn according to the distribution $\Lambda_{\sigma_\kappa^2} \triangleq \mathcal{N}(\mathbf{0}, \frac{1}{\sigma_\kappa^2}\mathbf{I})$, i.e. implicitly defining a Gaussian kernel $\kappa(\mathbf{x}, \mathbf{y}) \triangleq \exp(-\|\mathbf{x}-\mathbf{y}\|_2^2/(2\sigma_\kappa^2))$ in the data domain.

If $\mathbf{c}_1, ..., \mathbf{c}_k$ are the cluster centers to recover, we define

$$\varepsilon \triangleq \min_{1 \leq i \neq j \leq k} \|\mathbf{c}_i - \mathbf{c}_j\|_2 \tag{4.1}$$

the separation between clusters. We will see that the optimum kernel variance is closely connected to this separation. We also recall[4] that when working with random Fourier features and this distribution of frequencies, for any π_1, π_2 we have (using the notation $\kappa(\mathbf{u}) \triangleq \kappa(\mathbf{0}, \mathbf{u})$) $\|\pi_1 - \pi_2\|_\kappa = \|\kappa \star \pi_1 - \kappa \star \pi_2\|_{L^2(\mathbb{R}^d)}$. This interpretation of the maximum mean discrepancy in terms of a low-pass filtering operation with κ tends to suggest intuitively that the optimal kernel variance should be no larger than ε^2, as locality information regarding clusters separated by a distance ε might otherwise be lost.

From a theoretical perspective, statistical learning guarantees have been obtained [210] for compressive clustering with weighted[5] random Fourier features. A separation assumption $\varepsilon > 0$ has been shown to be necessary[6] for a lower restricted isometry property to hold. Vice-versa, learning guarantees have been established when $\sigma_\kappa^2 \lesssim \varepsilon^2 / \log k$, by establishing a lower restricted isometry property on the model of bounded and ε-separated mixtures of Diracs[7]. Yet, when σ_κ^2 is too small, reconstruction algorithms can get stuck in local minima and theoretical error bounds become vacuous. We provide an empirical illustration of this fact just below.

It is worth noting that the weights used in [210] might simply be a proof artifact, and we consider in the following frequency vectors which are truly drawn according to $\Lambda_{\sigma_\kappa^2}$, and the corresponding features are not reweighted.

[4] Cf. Section 2.3.3.

[5] i.e. using a frequency distribution differing slightly from $\Lambda_{\sigma_\kappa^2}$, but still inducing a Gaussian kernel.

[6] We will explain this in Chapter 10.

[7] We say that a mixture of Diracs is "ε-separated" if all the Diracs that compose it are separated by at least ε in l_2-norm.

4.1.2 An illustration with CL-OMPR

In order to better understand the role that the kernel variance σ_κ^2 plays in the compressive clustering approach, we propose to take a closer look at the CL-OMP algorithm used to address the inverse problem. Its main steps are recalled in Algorithm 4.1. Note that we use in practical applications the CL-OMPR algorithm, which adds "replacement" steps after the k loop iterations of CL-OMP as described in Chapter 2, but we describe here only CL-OMP for simplicity.

In Figure 4.1, we show the evolution over iterations of the cost function appearing at line 3 of Algorithm 4.1 for three different values of the kernel variance σ_κ^2 on a two-dimensional dataset made of four separated clusters. At iteration $i+1$, if atoms $(\theta_l)_{1 \leq l \leq i}$ and weights $(\alpha_l)_{1 \leq l \leq i}$ have been previously selected, denoting $r = \tilde{\mathbf{s}} - \sum_{1 \leq l \leq i} \alpha_l \Phi(\boldsymbol{\theta}_l)$ the residual, this cost function reads $\theta \mapsto \mathfrak{R}\left(\left\langle \frac{\mathcal{A}(\delta_\theta)}{\|\mathcal{A}(\delta_\theta)\|_2}, r \right\rangle\right)$ where $\mathfrak{R}$ denotes the real part.

Input: Sketch $\tilde{\mathbf{s}}$, sketching operator $\mathcal{A}$, size of mixture k.

1 $\hat{r} \leftarrow \tilde{\mathbf{s}}, \Theta \leftarrow \emptyset$ `// Init`

2 **for** $i \leftarrow 1$ **to** k **do**

`/* Find a new atom */`

3 $\Theta \leftarrow \Theta \cup \left\{ \arg\max_\theta \ \mathfrak{R}\left\langle \frac{\mathcal{A}(p_\theta)}{\|\mathcal{A}(p_\theta)\|}, \hat{r} \right\rangle \right\}$

`/* Project to find weights */`

4 $\alpha \leftarrow \arg\min_{\alpha \geq 0} \left\| \tilde{\mathbf{s}} - \sum_{j=1}^{|\Theta|} \alpha_j \mathcal{A}(p_{\theta_j}) \right\|$

`/* Adjust centroids locations */`

5 $\Theta, \alpha \leftarrow \arg\min_{\Theta, \alpha \geq 0} \left\| \tilde{\mathbf{s}} - \sum_{j=1}^{|\Theta|} \alpha_j \mathcal{A}(p_{\theta_j}) \right\|$

`/* Update residual */`

6 $\hat{r} \leftarrow \tilde{\mathbf{s}} - \sum_{j=1}^{|\Theta|} \alpha_j \mathcal{A}(p_{\theta_j})$

7 **return** the support Θ, the weights α.

Algorithm 4.1: CL-OMP (Compressive Learning – Orthogonal Matching Pursuit)

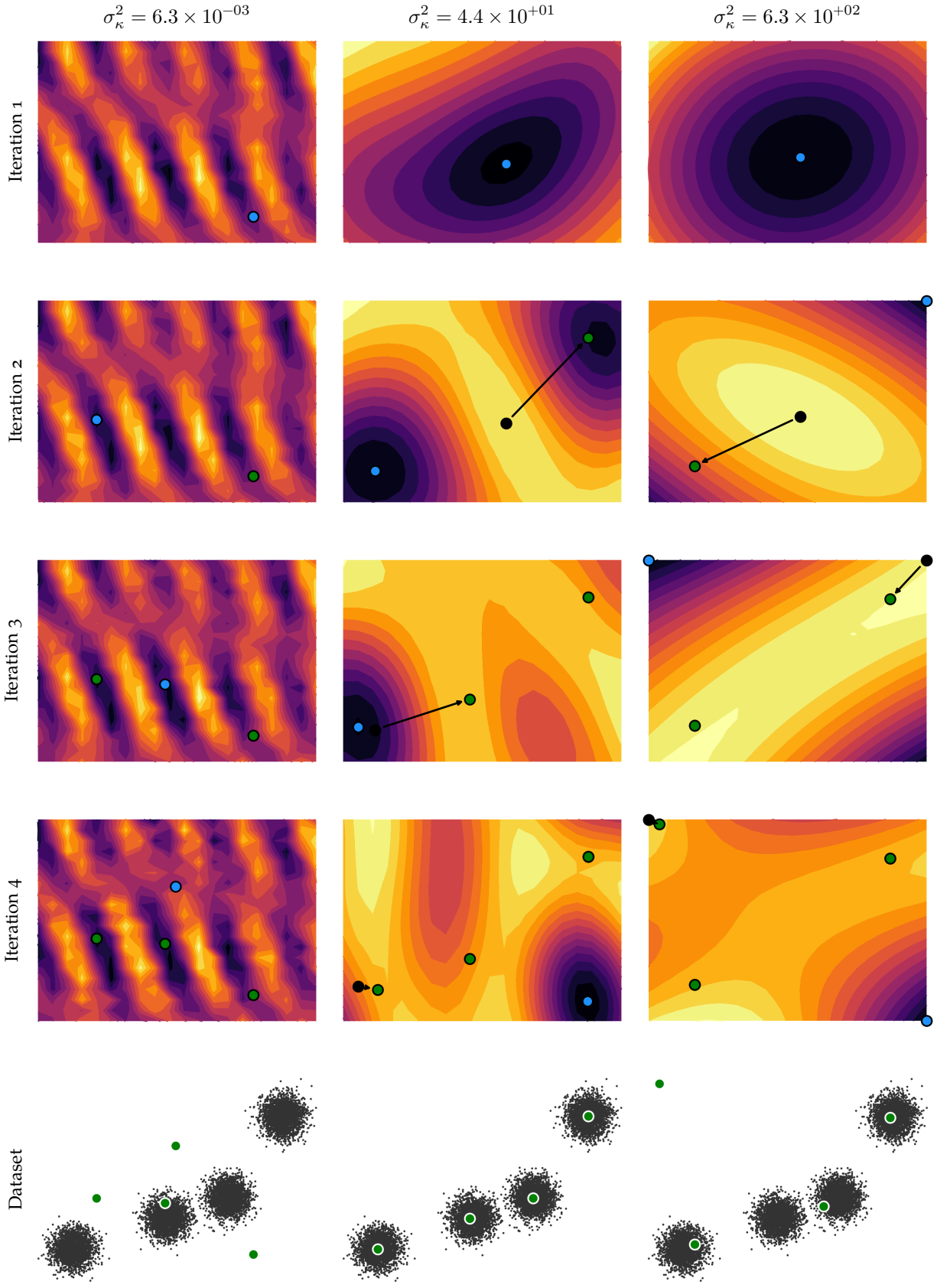


Figure 4.1: Impact of the kernel scale on the CL-OMP algorithm for three different kernel variances (left: too small, middle: optimal, right: too large). Synthetic data, $k = 4$, $n = 1000$, $d = 2$. The top rows represent the cost function (line 3 of Algorithm 4.1, darker = higher = better) used to pick the new atom for the four iterations of the algorithm (top to bottom), together with the selected atom (in blue) and atoms from previous iterations (in green). Black arrows correspond to the movement of atoms during the "global" optimization phases (Line 5 of Algorithm 4.1). The last row depicts the dataset in grey, with the recovered atoms superimposed in green.

As expected, the algorithm fails to recover the cluster centers when the chosen kernel variance is too large (rightmost column). Intuitively, and following the low-pass filtering interpretation, the whole dataset is smoothed to a single cluster and the individual groups cannot be recovered. However when σ_κ^2 is too small, the cost function used to pick the new atom has many spurious local minima (leftmost column), and the solution provided by the algorithm is meaningless.

Note that the alternative CL-AMP algorithm which will be presented in Chapter 6, despite being quite differently designed, suffers from numerical issues as well when the kernel scale is not properly chosen.

4.1.3 Existing heuristics

As shown above, it is crucial to choose the kernel scale wisely when performing compressive clustering in practice. Furthermore, one usually wants to estimate this scale quickly and using a small subset of the dataset, as adopting a compressive approach looses its interest when heavy calculations are required beforehand. Some empirical works suggested to tune σ_κ^2 using an estimate of the intra-cluster variance [4]. In certain scenarios we also reported that the second moment of the data, which we define as

$$\sigma_X^2 = \frac{1}{dn}\sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}\|_2^2 \quad \text{where} \quad \boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i, \tag{4.2}$$

can yield lower empirical error[8] than the intra-cluster variance [26].

However, these heuristics have been used because they empirically work well for some specific datasets, but no in-depth analysis can justify these observations. As shown in Figure 4.2, they do not generalize well to other datasets. In particular, the intra-cluster variance, which can seem intuitive in terms of "low pass filtering" interpretation, is highly inaccurate.

[8] This observation was made using a non-Gaussian frequency distribution (see Section 4.2.6), but remains valid in our context.

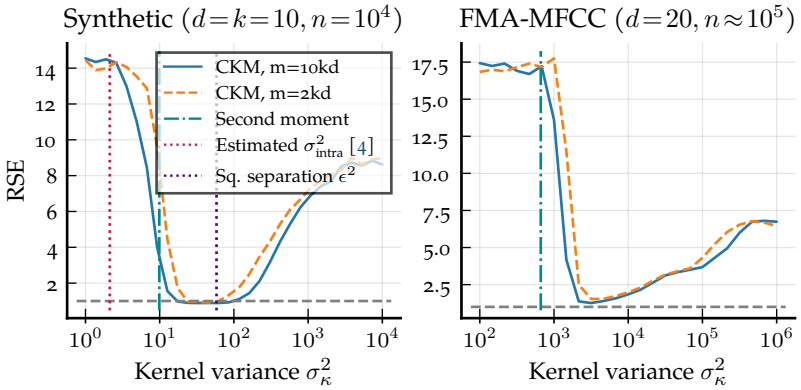


Figure 4.2: Clustering error versus kernel variance for synthetic and real data. FMA-MFCC consists in the MFCC features of the free music archive dataset [211] (cf. Appendix D for reproducibility). Means over 100 trials. Estimated σ^2_{intra} is out of the figure range for FMA (below 10^2). Synthetic data generated according to the generative model described in (4.5) just below.

In this figure (and in the following), CKM stands for "compressive k-means", and m denotes the sketch size. The clustering quality of centroids $\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_k]$ for the dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$ is measured using the following mean squared error

$$\text{MSE}(\mathbf{X}, \mathbf{C}) \triangleq \frac{1}{n}\sum_{i=1}^{n} \arg\min_{j \in [\![1,k]\!]} \|\mathbf{x}_i - \mathbf{c}_j\|_2^2, \tag{4.3}$$

which is just another notation for the clustering risk[9] $\mathcal{R}_{\text{KM}}(\mathbf{C}, \mathbf{X}) = \mathbf{E}_{\mathbf{x}\sim\pi_{\mathbf{X}}} l_{\text{KM}}(\mathbf{C}, \mathbf{x})$ where $\pi_{\mathbf{X}}$ is the empirical distribution of $\mathbf{X}$. We also introduce the relative squared error

$$\text{RSE}(\mathbf{X}, \mathbf{C}) \triangleq \frac{\text{MSE}(\mathbf{X}, \mathbf{C})}{\text{MSE}(\mathbf{X}, \mathbf{C}_{\text{k-means}})}, \tag{4.4}$$

where $\mathbf{C}_{\text{k-means}}$ is obtained by running Lloyd's k-means algorithm[10], so that relative errors which are close to one correspond to successful clusterings.

[9] cf. Definition 1.2

[10] We use in practice k-means++ with 3 trials. Our goal is not to get the optimal clustering with high precision (the problem is anyway NP-hard), but simply to get a reasonable estimation which can serve as a reference.

4.2 Experimenting with synthetic datasets

In order to better understand how the optimal kernel variance relates to the dataset, we propose to generate synthetic data according to a parametric gaussian mixture model and to study the impact of each parameter. Throughout this section, data is thus always generated according to a mixture π of k normal distributions defined as

$$\pi \triangleq \sum_{i=1}^{k} \alpha_i \mathcal{N}(\mathbf{c}_i, \sigma^2_{\text{intra}} \mathbf{I}_d), \tag{4.5}$$

$$\text{where } \mathbf{c}_i \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \sigma^2_{\text{inter}} \mathbf{I}_d), \qquad \sigma_{\text{intra}} = s k^{1/d} \tag{4.6}$$

$$\text{and } \sum_{i=1}^{k} \alpha_i = 1.$$

The weights α_i are the weights of the different clusters, and unless otherwise specified we choose $\alpha_i = 1/k$ for each $i \in [\![1, k]\!]$. The intra- and inter-cluster variances σ^2_{intra} and σ^2_{inter} control the separation and spread of the components of the mixture. The quantity s is used to parametrize the separation between clusters in a way which is coherent[11] across dimensions [4, Section 5.1]. Depending on the context, we might specify the separation parameter s or directly σ^2_{intra}, but both quantities are directly related to each other. We also define $\rho^2 \triangleq \sigma^2_{\text{inter}} / \sigma^2_{\text{intra}}$ the variance ratio; a large value of ρ thus corresponds to well-separated clusters. Two datasets in dimension $d = 2$ for two different values of ρ (obtained using the same σ^2_{inter} and two different values of s, i.e. two different values of σ^2_{intra}) are depicted in Figure 4.3. Note that synthetic data in Figure 4.2 was already generated according to this model.

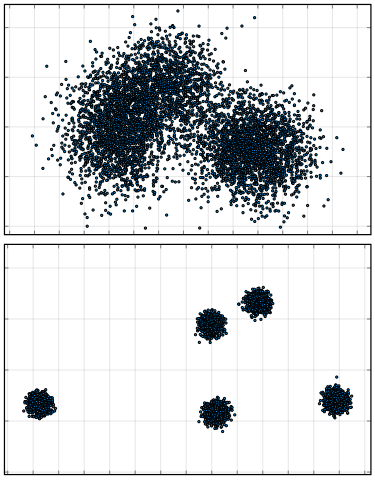


Figure 4.3: Examples of randomly drawn datasets for parameters $k = 5, d = 2$ with separation parameter $s = 1.5$ (top) and $s = 8$ (bottom).

[11] More precisely, when s equals one the volume of a sphere of radius σ_{inter} fits k spheres of radius σ_{intra}.

We now look at the impact of the different parameters k, d, m, σ^2_{intra} and σ^2_{inter}.

4.2.1 Impact of the sketch size

As could already be observed on the left part of Figure 4.2, increasing the sketch size slightly increases the range of variances for which near optimal clustering performance is obtained. We plot the error in log-scale in Figure 4.4 to better illustrate this fact. Because there are kd parameters to estimate here, and following previous works [2], we choose $m = Ckd$ for different values of $C > 1$.

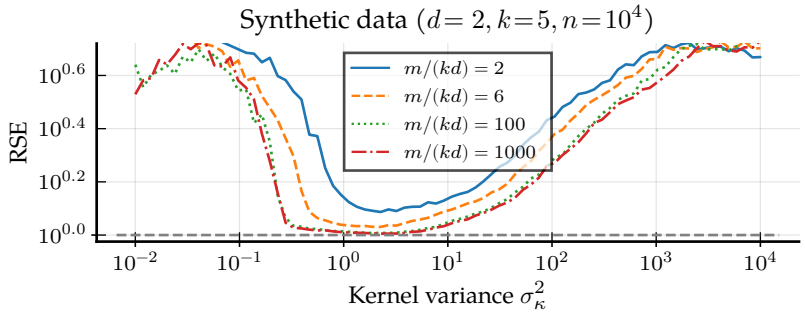


Figure 4.4: Clustering error versus kernel variance for different sketch sizes. Medians over 100 trials. Synthetic data generated according to (4.5), $\sigma^2_{\text{intra}} = 1.0, \rho^2 = 10.0$.

The data dimensionality d is reduced for this figure compared to Figure 4.2, which explains that the curves differ slightly for smaller values of m. However the same observations can be made. One can clearly see that the value of the optimal kernel scale only mildly depends on m and that the range of kernel variances σ_κ^2 for which near optimal performance is achieved gets larger as the sketch size grows. Overall there is a tradeoff between the sketch size m and the precision at which the variance needs to be tuned. To achieve high compression ratios, i.e. small $m/(kd)$ with good performance it seems important to have an accurate estimate of the optimal kernel variance. It is however difficult to be more explicit regarding the precision at which the kernel variance must be estimated, as the "width" of the range of variances for which near optimal performance is achieved itself depends on the various parameters in a way which is not straightforward to model. We thus focus in a first time on the location of the optimum only.

4.2.2 Scale-invariance

The problem presents intuitively some scale invariance. Indeed, if $\mathbf{C}^* = [\mathbf{c}_1, ..., \mathbf{c}_n]$ denotes some optimal centroids (in terms of MSE) for a dataset $\mathbf{X}$, then for any constant $\beta > 0$, $\beta\mathbf{C}^* = [\beta\mathbf{c}_1, ..., \beta\mathbf{c}_n]$ are also optimal centroids for the scaled dataset $\beta\mathbf{X}$ – although the MSE will be scaled by β^2.

However, when working with the compressive approach, the kernel scale must be adapted to the scale of the dataset. In particular, if a given kernel variance σ_κ^2 is optimal for some dataset $\mathbf{X}$, we expect the variance $\beta^2\sigma_\kappa^2$ to be optimal for $\beta\mathbf{X}$. Indeed, when the feature map Φ is built using the (scalar) feature function $\phi_{\boldsymbol{\omega}}(\mathbf{x}) = e^{-i\boldsymbol{\omega}^T\mathbf{x}}$ (with in practice $\boldsymbol{\omega} \in \{\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m\}$), we have for any $\beta > 0$: $\phi_{\boldsymbol{\omega}}(\mathbf{x}) = \phi_{\frac{1}{\beta}\boldsymbol{\omega}}(\beta\mathbf{x})$, i.e. the effect of scaling the dataset by a constant β can be balanced by a factor $1/\beta$ on the frequencies, and scaling the frequency variance by $1/\beta^2$ means scaling the kernel variance by β^2 in the data domain.

The quantity $\rho^2 = \sigma^2_{\text{inter}}/\sigma^2_{\text{intra}}$ is interesting to this regard, as it is invariant to data scaling – i.e. by definition, it does not vary when scaling at the same time σ^2_{inter} and σ^2_{intra} by the same factor. Figure 4.5 shows, for three different variance ratios ρ^2, each being obtained with two different combinations of $(\sigma^2_{\text{inter}}, \sigma^2_{\text{intra}})$, how the RSE behaves as a function of $\sigma_\kappa^2/\sigma^2_{\text{inter}}$.

In each setting, and as expected, the same results are obtained in-

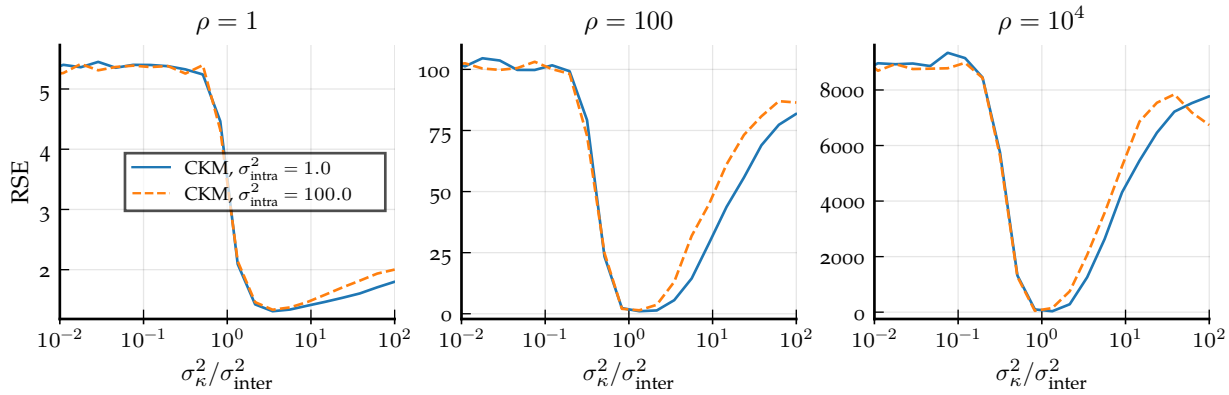


Figure 4.5: Clustering relative error vs. kernel variance for multiple variance ratios ρ. Here $k = d = 10$.

dependently of the combination of $(\sigma^2_{\text{inter}}, \sigma^2_{\text{intra}})$ used to obtain the considered ratio ρ. More interestingly, the optimal variance seems empirically to be always close to $\sigma^2_\kappa \approx \sigma^2_{\text{inter}}$ for the three values of ρ considered.

4.2.3 Impact of the separation between clusters

The effect of rescaling having been established, we propose to further investigate the role of the ratio ρ^2. Figure 4.6 shows the relative clustering error as a function of both ρ^2 and σ^2_κ for two different pairs (k, d), using the sketch size $m = 4kd$. The yellow areas correspond to successful runs of the algorithm (RSE close to 1).

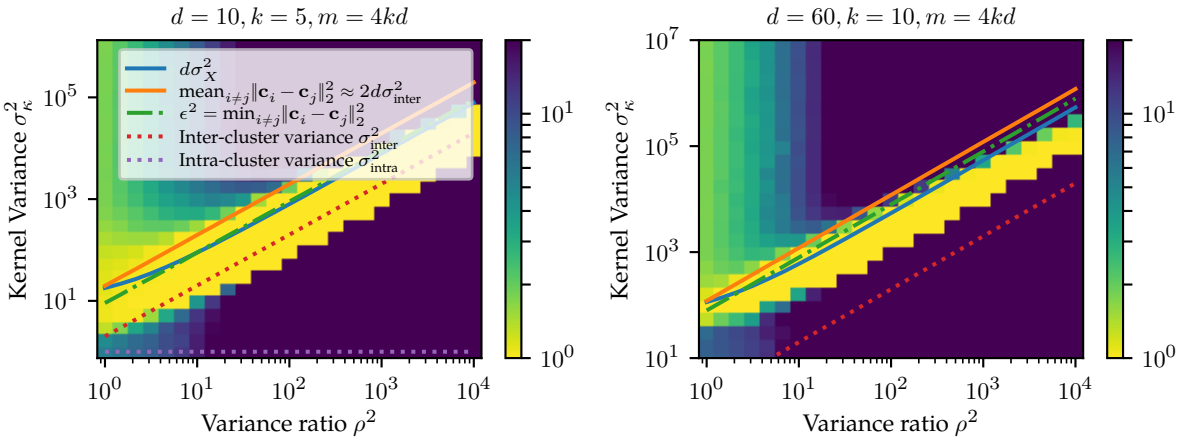


Figure 4.6: Impact of the cluster separation on the optimal kernel variance. The plotted quantity (color) is the RSE (medians over 100 trials). The intra-cluster variance is here fixed to $\sigma^2_{\text{intra}} = 1.0$.

We draw on top of the colormap several quantities. Since both the dataset generation and the sketching procedure are randomized, all the curves corresponding to random quantities are average curves. In accordance with the heuristic mentioned in Section 4.1.3, we draw the curve corresponding to $d\sigma^2_X$, where σ^2_X is defined in (4.2). It is important to note here that, if $\mathbf{c}_i \neq \mathbf{c}_j$ are cluster centers drawn according to our model, then $E[\|\mathbf{c}_i - \mathbf{c}_j\|_2^2] = 2\sigma^2_{\text{inter}} d$ as $\mathbf{c}_i - \mathbf{c}_j \sim \mathcal{N}(0, 2\sigma^2_{\text{inter}}\mathbf{I}_d)$. We also show the inter- and intra- cluster variances, as well as the (expected) squared separation $\varepsilon^2 = \min_{1\leq i\neq j\leq k}\|\mathbf{c}_i - \mathbf{c}_j\|_2^2$.

The inter-cluster variance matches only the yellow area for $d = 10$,

which should be expected as the dimension plays a role here. All quantities $\varepsilon^2, E\big[\|\mathbf{c}_i - \mathbf{c}_j\|_2^2\big]$ and $d\sigma_X^2$ however seem to scale with ρ^2 asymptotically similarly to the optimal kernel variance. The curves appear to be slightly shifted compared to the yellow area, but one should keep in mind that other parameters such as the dimension d and the number of clusters k also play a role as discussed below. This suggests at least for now that the optimal kernel variance scales linearly with $\sigma^2_{\text{inter}}/\sigma^2_{\text{intra}}$ when other quantities are fixed.

Connection with the second moment From the two mentioned heuristics, the second moment (scaled by the dimension) seems so far to be the most promising one. Hence is is worth noting that we have $\mathbf{E}[\sigma_X^2] \approx \sigma^2_{\text{inter}} + \sigma^2_{\text{intra}}$ when $\mathbf{X}$ is drawn according to (4.5).

4.2.4 Impact of the dimension

We now consider ρ to be fixed[12], and see how the dimension impacts the optimal kernel variance in Figure 4.7.

[12] More precisely, we also fix $\sigma^2_{\text{intra}} = 1.0$, so that we avoid an implicit dependence in d or k which would appear if we were using the separation parameter s.

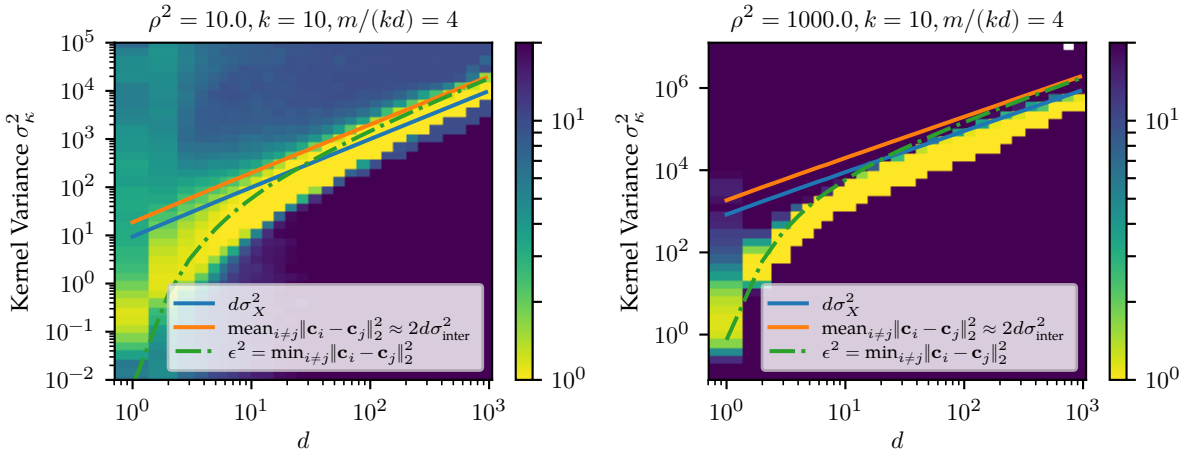


Figure 4.7: Impact of the dimension on the optimal kernel variance. Medians over multiple trials (100 for small dimensions, less for higher dimensions because of too long runtimes).

This figure shows clearly that, although the three curves superimposed on the plot grow asymptotically in a similar manner, only the squared separation ε^2 matches in low dimension the observations, while both $d\sigma_X^2$ and the mean distance between cluster centers are irrelevant when $d < 10$.

4.2.5 Impact of k

We now fix the dimension d, the sketch size and the separation between clusters[13], and look at the impact of the number of clusters k. Results are given in Figure 4.8 for both $\rho^2 = 10$ and $\rho^2 = 10^3$.

[13] Here again by setting σ^2_{intra} to a fixed value in order to avoid an implicit dependence in k or d.

This figure suggests like the previous one that the separation is clearly the important quantity to estimate here. The observed correlation is perfect for $\rho^2 = 10$ (left), while the second moment and the mean inter-cluster distance appear to be irrelevant here. When $\rho^2 = 10^3$ (right), the correspondence is not perfect but the same

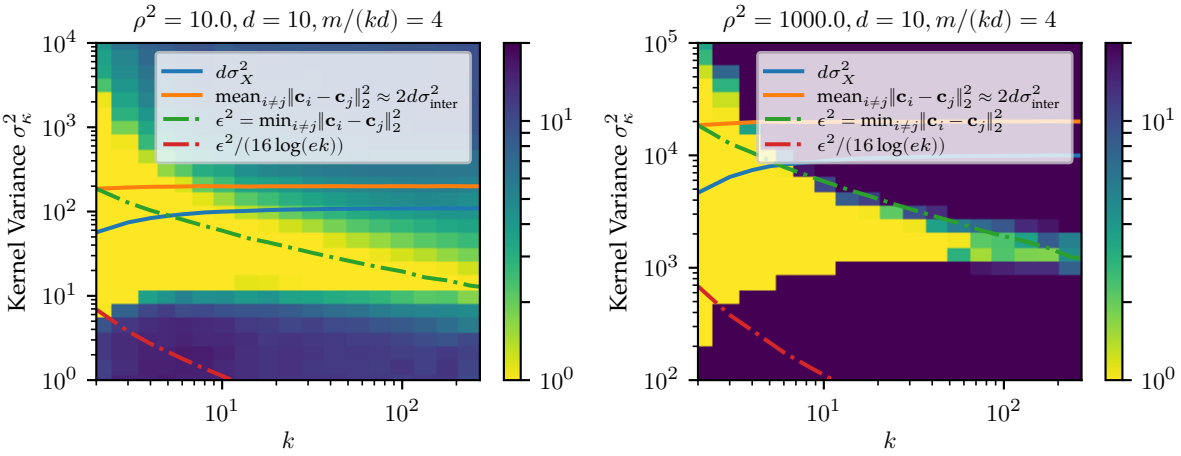


Figure 4.8: RSE as a function of both k and σ^2_κ. Here $\sigma^2_{\mathrm{intra}} = 1.0$, and errors are clipped to the interval $[1, 20]$ for coherence with other figures.

conclusion can be drawn. The red dot-dashed curve corresponds to $\sigma^2_\kappa = \varepsilon^2/(16\log(ek))$, which is the sufficient upper bound used to derive guarantees on the model of mixtures of ε-separated Diracs [6, Theorem 3.1]. It does not fit well our observations here, even when adjusting the multiplicative constant. This tend to suggest that the dependence in $\log(k)^{-1}$ might be a proof artifact rather than a necessity.

Estimation of the separation We now give a rough approximation of the expected separation. Deriving a closed form seems to be out of reach, but we rely on asymptotic results when $k \to \infty$. Naturally, the experiments presented so far in the chapter are relevant only for data drawn according to a Gaussian mixture model with balanced clusters, and thus are in practice of limited utility. This approximation should therefore not in any way be considered as a "generic" heuristic. It can however help when data is generated according to (4.5) with known parameters, which we will do in the following chapters. Furthermore, it can serve as a first rough estimation when dealing with other datasets.

Let $(\mathbf{c}_i)_{1\leq i\leq \infty}$ be a sequence of points drawn i.i.d. according to (4.6). We denote $p_{\mathbf{c}}$ the corresponding probability density function, i.e. $p_{\mathbf{c}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma^2_{\mathrm{intra}}\mathbf{I}_d)$. We denote $\varepsilon_k = \min_{1\leq i\neq j\leq k}\|\mathbf{c}_i - \mathbf{c}_j\|_2$ the minimum distance between the k first points. When $k \to \infty$, $k^2\varepsilon_k^d$ follows an exponential low. More precisely, for any $t > 0$ we have [212]

$$\lim_{k\to\infty} P[k^2\varepsilon_k^d > t] = e^{-ct}, \tag{4.7}$$

$$\text{where } c = \tfrac{1}{2}V_d\,\|p_{\mathbf{c}}\|_2^2 \tag{4.8}$$

$$\text{with } V_d = \frac{\pi^{d/2}}{\Gamma(1+d/2)} \text{ the volume of the unit ball} \tag{4.9}$$

$$\text{and } \|p_{\mathbf{c}}\|_2^2 = \int_{\mathbb{R}^d} p_{\mathbf{c}}^2(\mathbf{x})\,\mathrm{d}\mathbf{x} = (2\sigma_{\mathrm{inter}}\sqrt{\pi})^{-d}, \tag{4.10}$$

$$\text{i.e. } c = \frac{2^{-d-1}}{\Gamma(1+d/2)}\sigma_{\mathrm{inter}}^{-d}. \tag{4.11}$$

Hence, when k is large enough, we can use the following approxima-

tions

$$\forall t > 0\, P[\varepsilon_k > t^{1/d} k^{-2/d}] \approx \exp(-ct), \tag{4.12}$$
$$\forall u > 0\, P[\varepsilon_k \leq u] \overset{\text{(i)}}{\approx} 1 - \exp(-cu^d k^2).$$

(i) Using $u = (t/k^2)^{1/d}$, i.e. $t = u^d k^2$.

As a consequence, we can in this asymptotic regime approximate the density of ε_k by the density p_{ε_k} defined on $\mathbb{R}$ as

$$p_{\varepsilon_k}(u) \triangleq ck^2 d u^{d-1} \exp(-cu^d k^2). \tag{4.13}$$

With this approximation,

$$\begin{aligned}
\mathbf{E}[\varepsilon_k] &= ck^2 d \int_0^\infty u^d \exp(-cu^d k^2)\, \mathrm{d}u \\
&\overset{\text{(i)}}{=} ck^2 d \frac{\Gamma(1+1/d)}{d(ck^2)^{1+1/d}} \\
&= c^{-1/d} k^{-2/d} \Gamma(1+1/d) \\
&= 2^{1+1/d} \Gamma(1+d/2)^{1/d} \Gamma(1+1/d) k^{-2/d} \sigma_{\text{inter}}.
\end{aligned} \tag{4.14}$$

(i) cf. [213, Section 3.326, eq. 2.[10] p.337]

where the last equality follows from the definition of c in (4.11). Again, this derivation is only an approximation as it relies on the limit behavior when $k \to \infty$.

Empirical simulations (cf. Figure 4.9) suggest that this heuristic is usable as a rough approximation (relative approximation error inferior to 0.1) as soon as $k \geq 20$. It is naturally only an expectation, and data-dependent estimation procedures should be more relevant when working on fixed dataset instances.

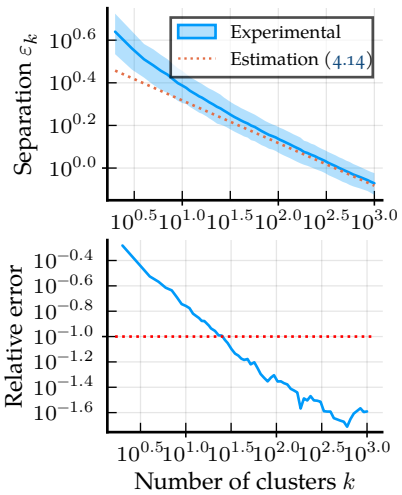


Figure 4.9: (Top) Mean (dark blue) and standard deviation (blue ribbon) of the minimal separation ε over 300 trials, as well as the approximation provided by (4.14). (Bottom) Corresponding relative error.

4.2.6 Impact of the frequency distribution

So far, we only considered drawing the frequency vectors according to a normal distribution. Note that drawing $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\sigma_\kappa^2}\mathbf{I}_d)$ is equivalent to the model $\boldsymbol{\omega} = R\sigma_\kappa^{-1}\boldsymbol{\varphi}$ with $\boldsymbol{\varphi} \sim \mathcal{U}(S^{d-1})$ and $R \sim \chi_d$, where $\mathcal{U}(S^{d-1})$ denotes the uniform distribution on the unit sphere S^{d-1} and χ_d the Chi distribution with d degrees of freedom. As the mass of the Chi probability distribution drifts away from the origin when the dimension increases, using another probability distribution producing more low frequencies could be beneficial. Previous works suggested for instance to use the following "adapted radius" probability density function [4, Section 3.3.1], initially in the context of Gaussian modeling

$$p_{\text{AR}}(R) \overset{\triangle}{\propto} \left(R^2 + \frac{1}{4}R^4\right)^{1/2} \exp\left(-\tfrac{1}{2}R^2\right). \tag{4.15}$$

This can be interpreted as a form of importance sampling. Our goal here is not to look for an "optimal" frequency distribution, as this is a complex subject on its own. We rather propose to quickly compare empirically the results obtained with the Gaussian kernel and the kernel induced when using random Fourier features with the radius distribution (4.15) – which we call the adapted radius kernel in the sequel. We do so in order to emphasize that the conclusions drawn earlier stay meaningful in this context.

Note that when using the "adapted radius" distribution exactly as proposed in [4], i.e. with $\boldsymbol{\omega} = R\sigma_\kappa^{-1}\boldsymbol{\varphi}$, we have $\mathbf{E}_{R\sim p_{\mathrm{AR}},\boldsymbol{\varphi}}\|R\sigma_\kappa^{-1}\boldsymbol{\varphi}\|^2 = \mathbf{E}_{R\sim p_{\mathrm{AR}}}[R^2]/\sigma_\kappa^2$ which differs from what would have been obtained with the Gaussian kernel as $\mathbf{E}\|\boldsymbol{\omega}\|^2 = d/\sigma_\kappa^2$ when $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\sigma_\kappa^2}\mathbf{I}_d)$. Hence when using the "adapted radius" distribution, we normalize the frequency to obtain comparable variances, i.e. we use

$$\boldsymbol{\omega} = \frac{R\sqrt{d}}{\sigma_\kappa \mathbf{E}_{R\sim p_{\mathrm{AR}}}[R^2]}\boldsymbol{\varphi} \tag{4.16}$$

with still $\boldsymbol{\varphi} \sim \mathcal{U}(S^{d-1})$, and R drawn with probability density function p_{AR}. We denote this distribution of $\boldsymbol{\omega}$ as $\mathrm{AR}(\sigma_\kappa^2)$, and $\mathbf{E}_{\boldsymbol{\omega}\sim\mathrm{AR}(\sigma_\kappa^2)}\|\boldsymbol{\omega}\|_2^2 = d/\sigma_\kappa^2$.

We now reproduce in Figure 4.10 the same results as in Figure 4.7, but using both Gaussian (on the left) and "adapted radius" (right) kernels.

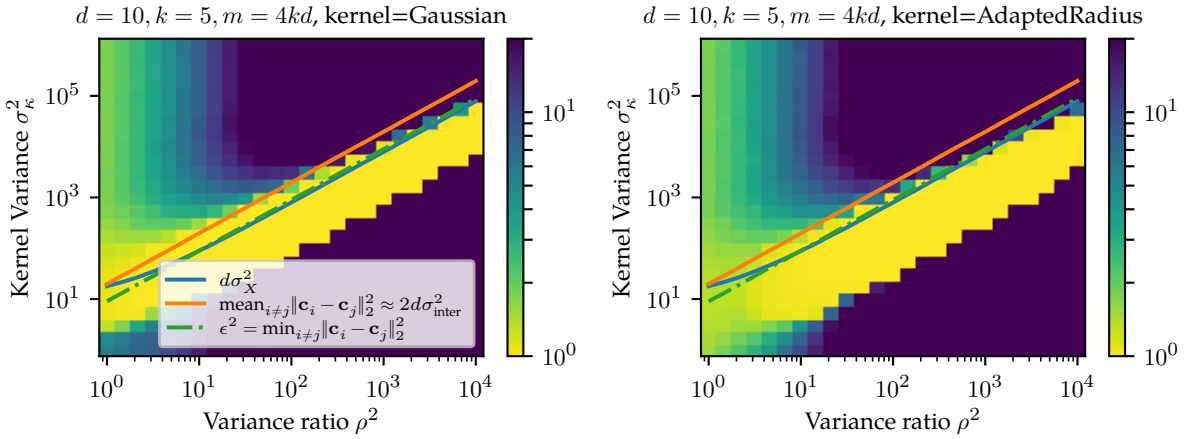


Figure 4.10: RSE as a function of σ_κ^2 and ρ for two different frequency distributions. Medians over 100 trials.

As can be seen on the figure, using this alternative distribution for the radius does increase (for this model at least) the range of kernel variance in which near optimal performance is achieved, especially on the lower side of the spectrum. This comes however at the cost of slightly worse results for smaller values of ρ. We provide a two-dimensional cut for comparison in Figure 4.11.

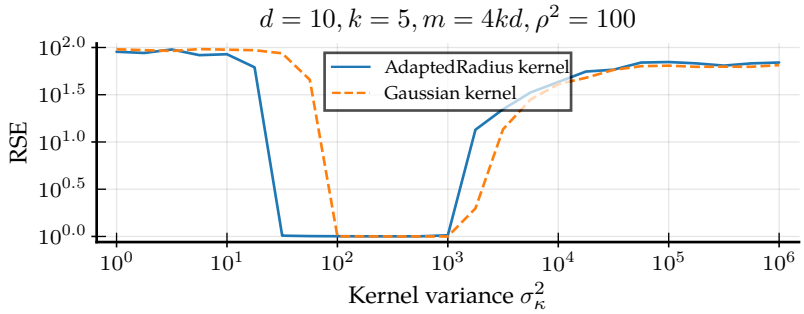


Figure 4.11: RSE as a function of σ_κ^2 for a fixed ρ and two different types of kernels. Medians over 100 trials.

To summarize, although looking for an "optimal" frequency distribution is not our goal here, we assume in the following chapters that the same considerations regarding the choice of the kernel variance

can be made for Gaussian and adapted radius kernels, provided that proper normalization is used.

4.3 Towards empirical estimation of the separation

In light of the simulations performed in the previous section, setting the kernel standard deviation to match the separation (i.e. $\sigma_\kappa \approx \varepsilon$) appears to be a reasonable choice in order to achieve good clustering results. It is thus of practical interest to be able to estimate the separation[14] ε of any given dataset. Furthermore, one usually wants to estimate this scale quickly and using a small subset of the data collection, as adopting a compressive approach looses its interest when heavy calculations are required beforehand. A simple method would be to run the k-means algorithm on a randomly chosen subset of the data, but this would not scale in high dimension and might perform poorly in the presence of unbalanced clusters.

[14] When no ground truth is known, we define the separation as the minimum distance between points of an optimal solution in terms of clustering risk.

To conclude this chapter, we thus briefly describe a proof of concept using sketches of random Fourier features. If the separation can indeed be estimated from a sketch, we could imagine computing a first smaller sketch – possibly on a small subset of the data –, and then use the estimated separation to draw new frequencies in order to compute the main sketch. To better understand which information regarding the separation can be inferred from such a sketch, we consider in a first time data drawn according to a pure and balanced mixture of Diracs $\pi_{\mathbf{C}} = \frac{1}{k}\sum_{1\leq i\leq k}\delta_{\mathbf{c}_i}$. In the following, we denote $\varphi(\boldsymbol{\omega})$ the characteristic function of $\pi_{\mathbf{C}}$ at $\boldsymbol{\omega}$, and define $\boldsymbol{\mu}_{ij} \triangleq \frac{1}{2}(\mathbf{c}_i + \mathbf{c}_j), \mathbf{d}_{ij} \triangleq \frac{1}{2}(\mathbf{c}_i - \mathbf{c}_j)$ for any i, j (so that $\varepsilon = 2\min_{i\neq j}\|\mathbf{d}_{ij}\|_2$). We start with the very simple setting of a mixture of $k = 2$ clusters, and see how our conclusions generalize to $k > 2$ below.

Case $\mathbf{k = 2}$. Estimating the separation in this settings is of limited utility, not only because clustering tasks most often involve larger numbers of clusters, but also because the minimum distance between cluster centers becomes the mean and maximum distance between clusters as well. However, as an illustration it is still interesting to note that

$$\varphi(\boldsymbol{\omega}) \triangleq \tfrac{1}{2}(e^{i\boldsymbol{\omega}^\top\mathbf{c}_1} + e^{i\boldsymbol{\omega}^\top\mathbf{c}_2}) = e^{i\boldsymbol{\omega}^\top\boldsymbol{\mu}_{12}}\cos(\boldsymbol{\omega}^\top\mathbf{d}_{12}).$$

Hence if we define

$$f(\boldsymbol{\omega}) \triangleq 1 - |\varphi(\boldsymbol{\omega})|^2 = \sin^2(\boldsymbol{\omega}^\top\mathbf{d}_{12}),$$

and if we draw $\boldsymbol{\omega} \sim \mathcal{N}(0, \sigma_\omega^2\mathbf{I}_d)$ with $\sigma_\omega \ll 1/\|\mathbf{d}_{12}\|$, then we have with high probability $f(\boldsymbol{\omega}) \approx |\boldsymbol{\omega}^\top\mathbf{d}_{12}|^2$. Bounds on the data itself can be used to bound $\|\mathbf{d}_{12}\|$ from above and choose an appropriate variance σ_ω^2. Multiple frequencies can be combined to improve the concentration, i.e. one can use $2\sigma_\omega^{-1}(\sum_{1\leq i\leq m} f(\omega_i)/m)^{1/2}$ as an estimator of ε.

Interestingly, using a sketch computed from a Gaussian mixture with centers $\mathbf{C}$ instead of $\mathbf{C}$ is not problematic as we are sampling only

low frequencies here. Estimations of a near-optimal σ_κ^2 obtained with this method are shown in Figure 4.12 as a function of the dimension.

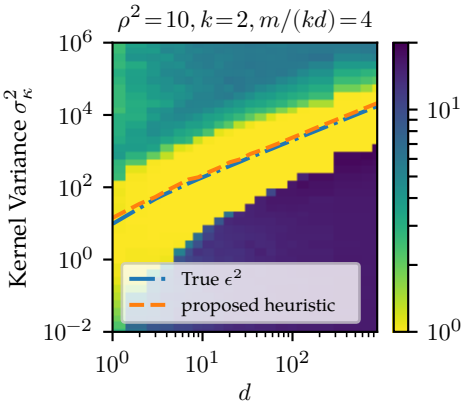


Figure 4.12: Estimated optimal kernel variance for $k = 2$. Data generated according to (4.5), medians over 100 trials.

Case k > 2. When the mixture involves more than two clusters, estimation is less straightforward. Defining $f_{ij} \triangleq \frac{1}{\pi}|\boldsymbol{\omega}^T\mathbf{d}_{ij}|$ we have in a similar manner

$$g_{\boldsymbol{\omega}}(t) \triangleq \tfrac{1}{2}(k^2|\varphi(t\boldsymbol{\omega})|^2 - k) = \sum_{i<j} \cos(2\pi f_{ij} t).$$

Proof 4.1: Using again the similar notations $\boldsymbol{\mu}_{ij} \triangleq \frac{1}{2}(\mathbf{c}_i + \mathbf{c}_j)$ and $\mathbf{d}_{ij} \triangleq \frac{1}{2}(\mathbf{c}_i - \mathbf{c}_j)$, we have $\mathbf{c}_i = \boldsymbol{\mu}_{ij} + \mathbf{d}_{ij}$ and $\mathbf{c}_j = \boldsymbol{\mu}_{ij} - \mathbf{d}_{ij}$. Let $p = \frac{1}{2}k(k-1)$ denote the number of pairs (without order) of centers. Every $\mathbf{c}_i$ appears in $k-1$ pairs, thus we have

$$\begin{aligned}
\varphi(\boldsymbol{\omega}) &= \frac{1}{k}\sum_{i=1}^{k} e^{i\boldsymbol{\omega}^T\mathbf{c}_i} = \frac{1}{k}\frac{1}{2k}\sum_{1\le i,j\le k} e^{i\boldsymbol{\omega}^T\mathbf{c}_i} + e^{i\boldsymbol{\omega}^T\mathbf{c}_j} \\
&= \frac{1}{k^2}\sum_{i,j} e^{i\boldsymbol{\omega}^T\boldsymbol{\mu}_{ij}}\cos(\boldsymbol{\omega}^T\mathbf{d}_{ij}) \\
k^4|\varphi(\omega)|^2 &= \left(\sum_{i,j} e^{i\boldsymbol{\omega}^T\boldsymbol{\mu}_{ij}}\cos(\boldsymbol{\omega}^T\mathbf{d}_{ij})\right)\left(\sum_{q,r} e^{-i\boldsymbol{\omega}^T\boldsymbol{\mu}_{qr}}\cos(\boldsymbol{\omega}^T\mathbf{d}_{qr})\right) \\
&= \sum_{i,j,q,r} \exp(\imath\boldsymbol{\omega}^T(\boldsymbol{\mu}_{ij} - \boldsymbol{\mu}_{qr}))\cos(\boldsymbol{\omega}^T\mathbf{d}_{ij})\cos(\boldsymbol{\omega}^T\mathbf{d}_{qr})
\end{aligned}$$

As $|\varphi(\omega)|^2$ is a real quantity, we have

$$k^4|\varphi(\omega)|^2 = \sum_{i,j,q,r} Q(i,j,q,r) \qquad \text{with}$$

$$\begin{aligned}
&Q(i,j,q,r) \\
&\triangleq \cos(\boldsymbol{\omega}^T(\boldsymbol{\mu}_{ij} - \boldsymbol{\mu}_{qr}))\cos(\boldsymbol{\omega}^T\mathbf{d}_{ij})\cos(\boldsymbol{\omega}^T\mathbf{d}_{qr}) \\
&= \tfrac{1}{2}\cos(\boldsymbol{\omega}^T(\mathbf{d}_{iq} + \mathbf{d}_{jr}))\left[\cos(\boldsymbol{\omega}^T(\mathbf{d}_{ij} + \mathbf{d}_{qr})) + \cos(\boldsymbol{\omega}^T(\mathbf{d}_{ij} - \mathbf{d}_{qr}))\right] \\
&= \tfrac{1}{4}\Big[\cos(\boldsymbol{\omega}^T(\mathbf{d}_{iq}+\mathbf{d}_{jr}+\mathbf{d}_{ij}+\mathbf{d}_{qr})) + \cos(\boldsymbol{\omega}^T(\mathbf{d}_{iq}+\mathbf{d}_{jr}-\mathbf{d}_{ij}-\mathbf{d}_{qr})) \\
&+\cos(\boldsymbol{\omega}^T(\mathbf{d}_{iq}+\mathbf{d}_{jr}+\mathbf{d}_{ij}-\mathbf{d}_{qr})) + \cos(\boldsymbol{\omega}^T(\mathbf{d}_{iq}+\mathbf{d}_{jr}-\mathbf{d}_{ij}+\mathbf{d}_{qr}))\Big] \\
&= \tfrac{1}{4}[\cos(2\boldsymbol{\omega}^T\mathbf{d}_{ir}) + \cos(2\boldsymbol{\omega}^T\mathbf{d}_{jq}) + \cos(2\boldsymbol{\omega}^T\mathbf{d}_{iq}) + \cos(2\boldsymbol{\omega}^T\mathbf{d}_{jr})]
\end{aligned}$$

Hence we have by symmetry

$$\begin{aligned}
k^4|\varphi(\omega)|^2 &= \sum_{i,j} k^2\cos(2\boldsymbol{\omega}^T\mathbf{d}_{ij}) \\
&= k^2\left(k + 2\sum_{i<j}\cos(2\boldsymbol{\omega}^T\mathbf{d}_{ij})\right) \\
\text{and} \quad |\varphi(\omega)|^2 &= \frac{1}{k^2}\left(k + 2\sum_{i<j}\cos(2\boldsymbol{\omega}^T\mathbf{d}_{ij})\right)
\end{aligned}$$

This means that the $(f_{ij})_{1\le i\ne j\le k}$ might be recovered, provided k is not too large, from the Fourier transform of $g_{\boldsymbol{\omega}}$ for some $\boldsymbol{\omega}$ as illustrated in Figure 4.13 (bottom). Recovering ε from the $(f_{ij})_{1\le i\ne j\le k}$ is straightforward in dimension $d = 1$. Yet, finding a way to estimate ε for $d > 1$ by combining estimations obtained in multiple directions $\boldsymbol{\omega}$ is still a

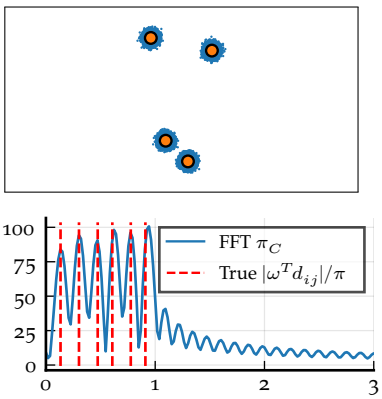


Figure 4.13: Dataset and mixture of cluster centers (top) and corresponding fast Fourier transform of $g_{\boldsymbol{\omega}}$ for one random direction $\boldsymbol{\omega}$ (bottom). $k = 4$ (6 pairs).

challenge. Sparse FFT [214] might be leveraged here to use as few samples of the characteristic function as possible.

If all the clusters are normally distributed and have similar scales (Figure 4.13 top), we can model the data distribution π as the convolution of a mixture of Diracs $\pi_{\mathbf{C}}$ (located at the cluster centers) and a Gaussian multivariate distribution $\pi_{\text{intra}} = \mathcal{N}(0, \sigma^2_{\text{intra}}\mathbf{I}_d)$. Keriven et al. have proposed [4] a way to estimate the intra-cluster variance σ^2_{intra} from a small sketch, using the fact that $|\varphi(\boldsymbol{\omega})|$ decreases approximately in $\exp(-\frac{1}{2}\|\boldsymbol{\omega}\|_2^2\sigma^2_{\text{intra}})$. Hence any empirical sketch $\tilde{\mathbf{s}}$ measured w.r.t. π can be deconvolved by dividing it pointwise by the (analytical) sketch of π_{intra} in order to estimate the sketch of the mixture of Diracs $\pi_{\mathbf{C}}$. The impact of such a deconvolution on the noise level of the signal could restrict severely the usability of this approach, but it remains nonetheless a straightforward way to extend the result to broader distributions.

4.4 Perspectives

This chapter clarifies on several aspects the role of the kernel variance when performing compressive clustering in practice. Section 4.2.6 suggests that when the variance ratio ρ^2 is not too small, using the "adapted radius" distribution rather than a pure Gaussian kernel helps as it increases the range of scales for which near optimal performance can be obtained, and thus reduces the risks of suffering from a poor choice of the kernel scale. The other experiments conducted in Section 4.2 allowed us to highlight that choosing $\sigma_\kappa \approx \varepsilon$ is a good heuristic, but learning ε directly from the dataset remains a challenge. Some ideas regarding the possibility to use sketches for this purpose have been discussed in Section 4.3, but this preliminary work is by no way close to providing a robust estimation procedure, which is left for future work. Furthermore, it would be interesting to look at the impact of unbalanced clusters and/or clusters with different covariance matrices in the generation model. Alternative importance sampling strategies should also be investigated in future work, as they can help significantly to alleviate the negative impact of a possibly poorly estimated kernel scale.

Chapter 5
High-Dimensional Sketching with Structured Linear Operators

Note The work presented in this chapter has been partially published to the ICASSP 2018 conference [24]. We provide here more extensive experiments, and discuss how existing theoretical guarantees can be adapted to this new setting.

Contents

The sketching operators considered so far, such as the one induced by random Fourier features (cf. Definition 2.5), are built as the succession of a linear and a non-linear step. Hence in order to compute the sketch of a dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$, one must first compute the matrix product $\boldsymbol{\Omega}^T\mathbf{X}$, then apply pointwise the nonlinear scalar function, which we denote ρ, and eventually average the resulting matrix across columns. The overall complexity of the sketching operation is thus driven[1] by the linear step, which scales in $\Theta(mdn)$. Furthermore, the sheer cost (in space) of storing the matrix $\boldsymbol{\Omega}$ is $\Theta(md)$, which can also be prohibitive. The sketch size m depends on the application considered, but will typically be at least of the order of d, yielding time and space complexities growing quadratically with the data dimension.

Many well-known linear digital signal processing operations, such as the discrete Fourier, cosine and wavelet transforms, can be performed using fast algorithms which can be interpreted as successions of multiplications by sparse matrices with particular structures[2]. This chapter considers using a matrix $\boldsymbol{\Omega}$ for which such a sparse factorization exist, rather than relying on a dense matrix with normally distributed i.i.d. entries, thus reducing the time and space complexities in the high-dimensional regime.

Summary of the contributions

- We review in Section 5.1 the different kinds of structured linear operators proposed in the literature.
- We leverage these results to design in Section 5.2 a structured transform for efficient compressive learning.
- We provide extensive simulations to study the computational gains in Section 5.3.

[1] This hypothesis, which seems reasonable in a first approach, is discussed further in Section 5.3.1 (paragraph "nonlinearity").

[2] This includes in particular many "divide-and-conquer" algorithms, such as the Cooley-Tukey method for the Fourier transform.

- We show in Section 5.4 that the feature map Φ^{RFF} induces a proper Gaussian kernel when computed with some of the proposed structured blocks, and discuss how existing statistical learning guarantees can be adapted in this case.

5.1 Literature on structured transforms

We discuss in Section 5.1.1 an approach consisting in factorizing a given dense operator into a product of sparse terms for subsequent efficient usage, while Section 5.1.2 details how to directly draw random structured matrices with a prescribed behavior.

5.1.1 Factorization approaches

Le Magoarou et al. introduced [215] under the name Faμst (Fast Approximate Multi-layer Sparse Transforms) a general family of linear operators and an associated factorization algorithm. More precisely, a matrix $\mathbf{A}$ is said to be Faμst if it can be written as a product (hence "multi-layer") of few sparse matrices, i.e. $\mathbf{A} = \prod_{j=1}^{J} \mathbf{S}_j$, where the $(\mathbf{S}_j)_{1\leq j\leq J}$ are individually sparse, and ideally the total number s of nonzero coefficients in all the factors $(\mathbf{S}_j)_{1\leq j\leq J}$ is much lower than the dimension of the dense matrix $\mathbf{A}$. The storage cost and the computational cost of the multiplication, which both scale in $\mathcal{O}(s)$, are therefore likewise reduced.

A hierarchical factorization algorithm is introduced to get a Faμst approximation of a given operator $\mathbf{A}$ by looking for

$$(\mathbf{S}_j)_{1\leq j\leq J} \in \underset{(S_j)_{1\leq j\leq J}}{\arg\min} \left\| \mathbf{A} - \prod_{j=1}^{J} \mathbf{S}_j \right\|^2 + \sum_{j=1}^{J} R(\mathbf{S}_j),$$

where R is a regularizer favoring sparse factors. Applications to dictionary learning and acceleration of linear inverse problems are provided. Similar decompositions have also been used to speed-up k-means algorithm [216], by factorizing the matrix of cluster centers. A slightly different factorization approach based on Givens rotation has been developed [217] and applied to principal components analysis.

Owing to the nature (random with i.i.d. entries) of the linear operators used so far for compressive learning, generating a dense matrix and then approximating it by a product of structured factors would not necessarily yield good approximations, and be mostly inefficient in terms of memory as the dense matrix would still have to be generated and temporarily stored. We thus take a closer look at approaches which directly produce structured matrices.

5.1.2 Families of linear operators based on known structured blocks

In contrast to the previous works focusing on the approximation of one given dense matrix, multiple authors have introduced families of random linear operators with a prescribed structure, from which one

can directly sample from. This approach is well suited to our setting, where the operator $\mathbf{\Omega}$ is anyway randomly generated.

In order to produce operators that can be efficiently evaluated, the number r of random quantities – or degrees of freedom – in these models is kept small, typically linear with the dimension. That is, a structured $d \times d$ matrix will be built from at most $r = \mathcal{O}(d)$ random entries. Examples of known square matrices parametrized by a linear number of parameters include Toeplitz (i.e. with constant diagonals), circulant (special case of Toeplitz), Hankel (i.e. having constant skew-diagonals) and Vandermonde matrices. In most applications however, these constructions are not useful per se; a structured family is thus usually designed to combine a small number of random independent quantities together with other structured deterministic operators, yielding in the end matrices that are not only structured, but that also display desirable properties with respect to the application at hand.

The fast Johnson-Lindenstrauss transform was introduced [130] as an efficient way to perform geometry-preserving dimensionality reduction (cf. Chapter 3). It takes the form $\mathbf{A} = \mathbf{PHD}$, where $\mathbf{P}$ is a random sparse matrix drawn according to a specific distribution, $\mathbf{H}$ is a Walsh-Hadamard matrix, and $\mathbf{D}$ is a diagonal matrix with i.i.d. Rademacher[3] entries on the diagonal. Walsh-Hadamard matrices play a key role in many families presented here. They have mutually orthogonal rows and columns, allow to perform matrix-vector products in quasilinear time, and can be obtained recursively by the following definition:

[3] i.e. uniform in $\{-1, +1\}$.

$$H_1 = [1] \quad \text{and} \quad H_{2d} = \begin{bmatrix} H_d & H_d \\ H_d & -H_d \end{bmatrix} = H_2 \otimes_k H_d \tag{5.1}$$

where $\otimes_k$ denotes the Kronecker product. Examples are provided in Figure 5.1.

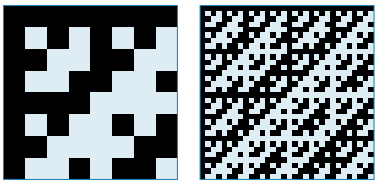


Figure 5.1: Naturally ordered Walsh-Hadamard matrices for $d = 2^4$ (left) and $d = 2^6$ (right). Black and blue entries represent respectively the values -1 and 1.

Although this construction yields only square matrices whose sizes are powers of 2, padding can be performed to cover all usecases. We discuss this matter in Section 5.2.3, but consider for now only square matrices for conciseness.

Many fast transforms families have been introduced in order to speed-up the computation of random features of the form

$$\mathbf{x} \mapsto \rho(\mathbf{Ax}) \tag{5.2}$$

for kernel approximation, where ρ is again a nonlinearity applied pointwise. We recall that a Gaussian kernel can be approximated by an inner product between features of the form (5.2) with $\mathbf{A}$ having i.i.d. normally distributed entries and $\rho = \exp(\imath \cdot)$ – we refer the reader to Section 2.3.2.

Le et al. introduced the Fastfood family [77] of matrices for Gaussian kernel approximation. A fastfood matrix can be factorized as $\mathbf{A}_{\text{ff}} = \mathbf{SHD}_G\mathbf{\Pi HD}_R$, where $\mathbf{D}_R, \mathbf{G}_G$ are diagonal with respectively Rademacher and Gaussian i.i.d. entries, $\mathbf{S}$ a diagonal random scaling matrix and $\mathbf{\Pi} \in \{0, 1\}^{d \times d}$ is a random permutation. The induced kernel

$$\kappa_{\text{ff}}(\mathbf{x}, \mathbf{y}) = \frac{1}{m} \exp(\iota \mathbf{A}_{\text{ff}} \mathbf{x})^* \exp(\iota \mathbf{A}_{\text{ff}} \mathbf{y}) \tag{5.3}$$

is shown to be an unbiased estimator of the Gaussian kernel and pointwise concentration results are provided.

Choromanski and Sindhwani generalized this work with their generic $\mathcal{P}$-model [218], whose design makes a clear separation between randomness and structure, and for which approximation guarantees are provided for the Gaussian and some arc-cosine kernels. Different metrics are introduced to quantify the structure of a given $\mathcal{P}$-model and its ability to "recycle" the random quantities, i.e. metrics measuring the level of dependence between the rows of the produced matrix.

Yu et al. observed empirically [219] that kernel estimators based on orthogonal operators[4] display lower approximation error, and further introduce the following construction, which is both structured and orthogonal:

$$\mathbf{A} = c\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1 \tag{5.4}$$

where $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$ are again diagonal with Rademacher i.i.d. entries and c is a constant. One can recognize here the association of three independent $\mathbf{HD}$ blocks. This association of a Hadamard matrix with a diagonal Rademacher matrix was already used by Ailon and Chazelle for fast Johnson-Lindenstrauss, and appears in the fastfood construction as well. Similar designs have been used by Bojarski et al., with guarantees oriented towards locality-sensitive hashing applications [220].

Further works provided theoretical evidence regarding the role of orthogonality and the importance of choosing an odd number of $\mathbf{HD}$ blocks in (5.4) [221]. Choromanski et al. later provided a more detailed analysis of regimes in which orthogonality yields better approximation performance for radial-basis function kernels [78], i.e. kernels of the form $\kappa(\mathbf{x}, \mathbf{y}) = \varphi(\|\mathbf{x} - \mathbf{y}\|)$ for some function φ.

Multiple works have also considered replacing the random quantities in these models by tunable parameters, yielding low-complexity families of linear operators that are especially useful for deep learning pipelines. Indeed, replacing the dense matrices typically used in fully-connected layers by such structured models allows to speed-up the runtimes while reducing the total number of parameters to learn in the model, and still providing good approximation capacities [222, 223, 224].

[4] We consider here computing features of the kind (5.2) where A has orthogonal rows.

5.2 Construction

We recall that our goal is to design a structured operator which can be used to replace the matrix of frequencies $\mathbf{\Omega}$ of size $d \times m$ used in the feature map $\Phi(\mathbf{x}) = \rho(\mathbf{\Omega}^T \mathbf{x})$, in order to compute sketches of size m as defined in (1.10). We address scaling concerns in Section 5.2.1, explain how to build one square structured block in Section 5.2.2 under specific hypotheses, and then detail how to extend this construction to arbitrary dimensions in Section 5.2.3.

5.2.1 Decoupling radial and directional distributions

As detailed in sections 2.3.2 and 4.2.6, a Gaussian kernel with spatial variance σ_κ^2 can be approximated using random Fourier features, using dense frequency[5] vectors drawn according to $\mathcal{N}(0, \frac{1}{\sigma_\kappa^2}\mathbf{I}_d)$. One way to generate such vectors is to use the generative model $\boldsymbol{\omega} = (R'/\sigma_\kappa)\boldsymbol{\varphi}$, where R' is a radius drawn as $R' \sim \chi_d$, and $\boldsymbol{\varphi}$ is a directional vector drawn uniformly (and independently from R) on the unit sphere S^{d-1}. This means that the probability distribution $\mathcal{N}(0, \frac{1}{\sigma_\kappa^2}\mathbf{I}_d)$ is also isotropic: its probability density function can be written $p(\boldsymbol{\omega}) = p_R(\|\boldsymbol{\omega}\|_2)$.

[5] In light of Chapter 2, and in particular given that the sketch is a collection of random samples of the characteristic function, we refer to the columns of $\boldsymbol{\Omega}$ in the following as frequency vectors.

In the following, we solely work with frequency distributions which have this separability property[6], i.e. for which the radial component can be drawn independently from the direction. However we will not restrict ourselves to isotropic distributions anymore.

[6] If both radial and directional probability distributions admit probability density functions p_R and $p_{\boldsymbol{\varphi}}$, and if $p_{\boldsymbol{\omega}}$ denotes the density of $\boldsymbol{\omega} = R\boldsymbol{\varphi}$, then we have the decomposition $p_{\boldsymbol{\omega}}(\boldsymbol{\omega}) = p_R(R)p_{\boldsymbol{\varphi}}(\boldsymbol{\varphi})$.

From the perspective of the design of the matrix $\boldsymbol{\Omega}$, the separability assumption implies that we can write $\boldsymbol{\Omega} = \mathbf{MS}$, where $\mathbf{M}$ is a matrix with l_2-normalized columns, and $\mathbf{S}$ is a diagonal matrix of radiuses with i.i.d. entries[7] drawn according to the desired radial distribution p_R. From a computational perspective, $\mathbf{S}$ is a diagonal matrix and thus does not impact the overall complexity of a matrix-vector product $\boldsymbol{\Omega}^T\mathbf{X}$, which is dominated by the multiplications by $\mathbf{M}^T$.

[7] The entries of $\mathbf{S}$ are mutually independent, but also independent with respect to $\mathbf{M}$.

In the following sections, we thus propose to keep this matrix $\mathbf{S}$ untouched and to modify only $\mathbf{M}$. More precisely, we will replace the "dense" generation scheme, where the columns of $\mathbf{M}$ are drawn independently from each other, by a structured scheme where $\mathbf{M}$ is built from i.i.d. square blocks $\mathbf{M}_1, ..., \mathbf{M}_b$. We can thus rewrite

$$\boldsymbol{\Omega} = [\mathbf{B}_1, ..., \mathbf{B}_b] = [\mathbf{M}_1\mathbf{S}_1, ..., \mathbf{M}_b\mathbf{S}_b]$$

where for each $i \in [\![1, b]\!]$, $\mathbf{S}_i$ corresponds to a square diagonal block[8] of the scaling matrix $\mathbf{S}$, and $\mathbf{M}_i$ is generated in a way such that it has l_2-normalized columns.

[8] Which implies that each $\mathbf{S}_i$ is a diagonal matrix as well.

5.2.2 Construction of a square block

We consider here building a square matrix $\mathbf{B}$, in the setting where $d = 2^q$ for some integer q. Note that this setting is similar to choosing a sketch size $m = d$, in which case one could simply use $\boldsymbol{\Omega}^T = \mathbf{B}$.

We use the construction

$$\mathbf{B} \triangleq \mathbf{M}_\mathbf{B}\mathbf{S}_\mathbf{B} \tag{5.5}$$

where $\mathbf{S}_\mathbf{B}$ is a diagonal scaling matrix with i.i.d. entries drawn according to the desired radial distribution p_R (see Section 5.2.1), and $\mathbf{M}_\mathbf{B}$ is a structured random block with l_2-normalized columns.

We will mainly focus on the setting $\mathbf{M}_\mathbf{B} = \mathbf{M}_{R^3}$, where $\mathbf{M}_{R^3}$ is a "triple-Rademacher" which we now define. Here and in the following, we work directly with the transpose of the square block $\mathbf{M}_\mathbf{B}^T$ for convenience – we recall that that the features are computed from the product $\boldsymbol{\Omega}^T\mathbf{X}$, and hence we are interested in the action of $\mathbf{M}_\mathbf{B}^T$ rather than $\mathbf{M}_\mathbf{B}$.

Definition 5.1: The triple-Rademacher (abbreviated "R^3") structured block is defined as

$$\mathbf{M}_{R^3}^T = \frac{1}{d^{3/2}}\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1, \tag{5.6}$$

where $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$ are diagonal matrices with i.i.d. Rademacher entries.

Here and in the following, $\mathbf{H}$ always denotes the Walsh-Hadamard matrix introduced in (5.1). Note that the normalization factor $d^{-3/2}$ is chosen such that $\mathbf{M}_{R^3}^T$ is an orthonormal matrix[9]. A representation of the construction (5.5) when $\mathbf{M}_\mathbf{B} = \mathbf{M}_{R^3}$ for $m = d = 2^4$ is given in Figure 5.2.

[9] As will be made clear in Lemma 5.4.

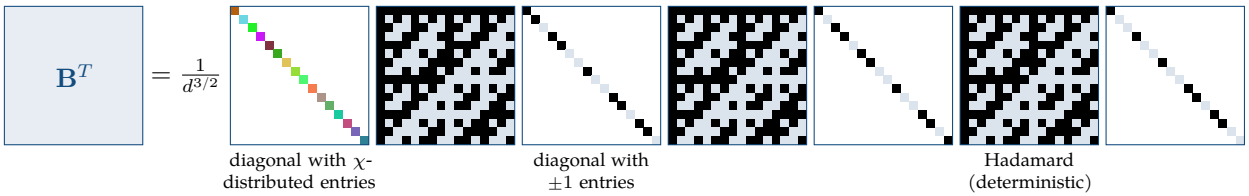


Figure 5.2: Design of a structured square matrix of frequencies. Walsh-Hadamard are made of ± 1 entries, and white parts are zeros. The constant factors are implicitly merged into the scaling matrix here.

The construction of $\mathbf{M}_{R^3}$ is based on the designs of Yu et al. [219] and Bojarski et al. [220] presented in Section 5.1.2, which themselves rely on the $\mathbf{HD}$ block initially introduced for dimensionality reduction. In the latter case, this block can be interpreted as a preprocessing step whose goal is to smooth the energy distribution of input data vectors [130].

Alternative constructions Although we will mostly use the construction presented above, we define here two other types of square blocks which can be used as alternatives for $\mathbf{M}_{R^3}^T$ in (5.5) (the scaling matrix $\mathbf{S}_\mathbf{B}$ staying the same).

Definition 5.2: The Gaussian - double Rademacher (abbreviated "GR^2") block is defined as

$$\mathbf{M}_{GR^2}^T = \frac{1}{d\|\mathbf{D}_G\|_F}\mathbf{H}\mathbf{D}_G\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1, \tag{5.7}$$

where $\mathbf{D}_G$ is diagonal with i.i.d. standard Gaussian entries, and $\mathbf{D}_1, \mathbf{D}_2$ are diagonal with i.i.d. Rademacher entries.

Definition 5.3 (Fastfood): The fastfood block (introduced in Section 5.1.2 and abbreviated "ff") is defined as

$$\mathbf{M}_{\text{ff}}^T = \frac{1}{\sqrt{d}\|\mathbf{D}_G\|_F}\mathbf{H}\mathbf{D}_G\mathbf{\Pi}\mathbf{H}\mathbf{D}_R. \tag{5.8}$$

where $\mathbf{D}_G$ is diagonal with i.i.d. Gaussian entries, $\mathbf{D}_R$ diagonal with i.i.d. Rademacher entries, and $\mathbf{\Pi}$ is a random permutation.

Both $\mathbf{M}_{R^3}^T$ and $\mathbf{M}_{\text{ff}}^T$ blocks have l_2-normalized rows, as will be proved

in Lemma 5.4. They are however not orthogonal. The $\mathbf{M}^T_{R^3}$ construction is preferred from a practical perspective as it can be stored more efficiently (see Section 5.2.4), but we will see that only $\mathbf{M}^T_{GR^2}$ and $\mathbf{M}^T_{\text{ff}}$ induce in expectation a Gaussian kernel. We discuss these aspects in Section 5.4.

5.2.3 Extension to arbitrary dimensions

We assumed previously that d was a power of 2, and explained how to build one square block in this case. However when performing compressive learning, empirical observations and theoretical results (cf. Section 2.5) suggest that the targeted sketch size m should be chosen to be of the order of the number p of parameters to learn. For instance with compressive k-means, recovering k d-dimensional centroids requires a sketch size $m \gtrsim kd$. Hence our construction needs to be extended to arbitrary $d \times m$ matrices with $m > d$, including the contexts where the dimension d is not a power of 2.

We denote $q = \lceil \log_2(d) \rceil$, $d_p = 2^q$, $r = \lceil m/d_p \rceil$ and $m_p = rd_p$ (where the letter "p" stands for "padding"). We use for sketching a $d_p \times m_p$ matrix $\mathbf{\Omega}$, whose transpose $\mathbf{\Omega}$ is built by horizontally stacking square blocks of size $2^q \times 2^q$ that are drawn independently according to (5.5) as depicted in Figure 5.3. When $d_p > d$, the distribution of the scaling matrix $\mathbf{S_B}$ in (5.5) must be adapted for homogeneity: for instance in order to approximate a Gaussian kernel, the radiuses will be drawn according to a χ_{d_p} probability distribution (instead of χ_d used otherwise).

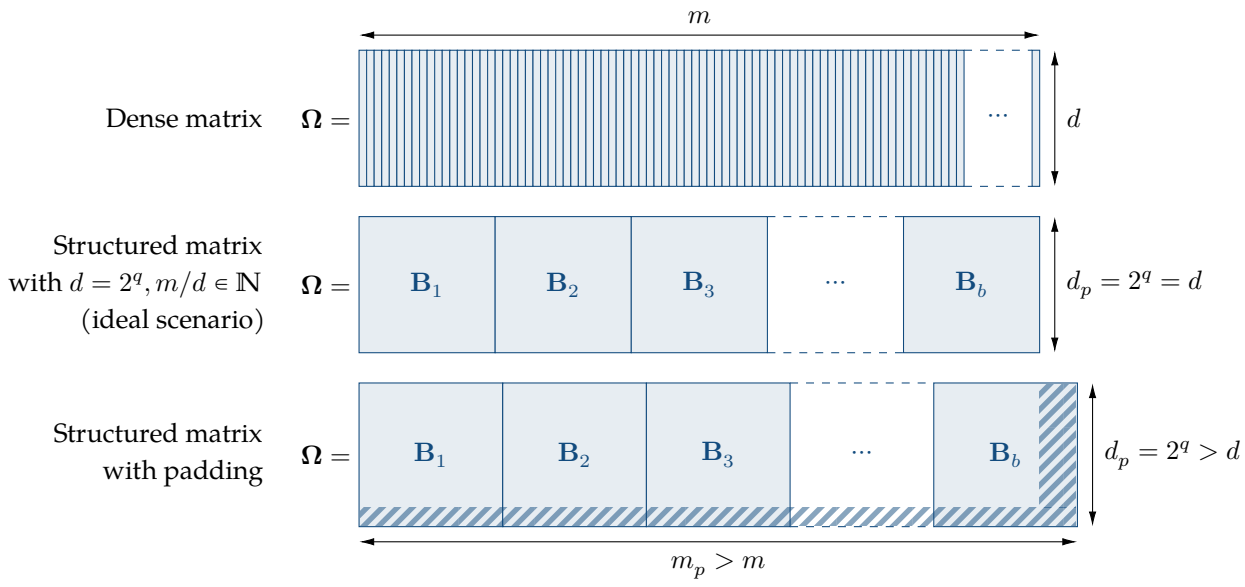


Figure 5.3: Construction of the whole matrix of frequencies, with the dense approach (top), with structured blocks when d is a power of 2 (middle), and with structured blocks and padding (bottom). The hatched area represents the "unused" part of the construction when using padding.

When sketching, we use zero-padding on the data $\mathbf{X}$ to get a matrix $\mathbf{X}_{pad}$ of matching dimensions, and keep only the m first rows when computing the product $\mathbf{\Omega}^T\mathbf{X}_{pad}$.

We now take closer look at the benefits of such a construction in comparison with a dense matrix. We recall that the matrix $\mathbf{\Omega}$ appears in the definition of the feature function $\Phi : \mathbf{x} \mapsto \rho(\mathbf{\Omega}^T\mathbf{x})$, which in turns

defines the sketching operator $\mathcal{A}$. This operator is used to compute the sketch, but also to address the optimization problem used to learn from the sketch (2.12).

5.2.4 Comparison of the costs

We motivated our interest for fast transforms by the benefits it would bring in terms of both the computational complexity of the matrix product, and the storage cost. As mentioned in the introduction, both matrix-vector complexity and the storage cost scale in $\Theta(md)$ for a dense operator.

With the proposed construction, the matrix-vector product for a square block of size d_p takes $\Theta(d\log(d))$ thanks to the fast Walsh-Hadamard transform. Multiplication by the whole matrix $\mathbf{\Omega}^T$ thus boils down to $\Theta(\lceil m/d_p \rceil d_p \log(d_p)) = \Theta(m\log(d))$ operations. From a storage perspective, the Walsh-Hadamard matrix being a deterministic operation, only the random diagonal entries need to be stored for a total memory cost of $\Theta(\lceil m/d_p \rceil d_p) = \Theta(m)$.

Learning from the sketch can take different forms depending on the learning task considered, but always requires evaluating the sketching operator. The usage of fast transform is thus likely to decrease the learning cost as well. Although the overall complexity depends on the considered application, we give complexities for the compressive k-means task in Table 5.1 just below, both for CL-OMPR, an alternative divide-to-conquer variant [2] referred as "Hierarchical", and the CL-AMP algorithm which will be introduced in Chapter 6.

	CKM	**FCKM**	**KM**
Time	$\mathbf{kd^2(n+k^2)}$	$\mathbf{kd\ln(d)(n+k\ln(k))}$	$\mathbf{ndkI}$
Sketching	nkd^2	$nkd\ln(d)$	–
Learning			
CL-OMP(R)	k^3d^2	$k^3d\ln(d)$	–
CL-AMP	k^2d^2I	$k^2d\ln(d)I$	–
Hierarchical	$k^2\ln(k)d^2$	$k^2\ln(k)d\ln(d)$	–
Space	$\mathbf{kd(d+n_b)}$	$\mathbf{kdn_b}$	$\mathbf{nd}$
Sketch	kd	kd	–
$\mathbf{\Omega}$	kd^2	kd	–
$\mathbf{\Omega}^T\mathbf{X}$ (by batch)	kdn_b	kdn_b	–

Table 5.1: Time and space (storage) complexities for k-means (KM), compressive k-means with a dense matrix (CKM) and fast compressive k-means (FCKM), i.e. CKM with fast transforms, for $m \approx kd$. Here n_b denotes the batch size used for sketching, k the number of clusters, I the number of iterations (for k-means and CL-AMP). All complexities should be read as $\Theta(\cdot)$. The space complexity for $\mathbf{\Omega}^T\mathbf{X}$ refers to the cost of storing the intermediate quantities $\mathbf{\Omega}^T\mathbf{X}_i$, where $\mathbf{X}_i$ is a batch of n_b columns of $\mathbf{X}$.

Note that for the compressive Gaussian mixture modeling task, which is performed with the same features, the sketching time and space costs are exactly the same as the one given in Table 5.1. However, the function to optimize during the learning phase involves the element-wise squared $\mathbf{\Omega}$, for which no simple form exists for the given construction. Depending on the available resources, it might be more efficient when possible to still use a fast transform and cache this matrix at the same time, which increases the storage cost but makes it possible to still benefit from the speedup on linear operations.

Similar speed-ups and memory gains could be achieved for other compressive learning tasks, such as compressive PCA, but are not reported here for conciseness.

5.3 Experimental validation

We now provide some empirical insights on the effectiveness of the proposed structured transform, focusing mainly on the compressive k-means learning task. We report runtime speedups in Section 5.3.1, and clustering performance for both synthetic and real data in sections 5.3.2 and 5.3.3. Unless otherwise specified, experiments involving structured matrices use $\mathbf{M}_{R^3}$ blocks. We will see later in the thesis in Section 9.3.2 a usecase of fast transforms for PCA in dimension $d = 32768$, where dense features can simply not be used. In this section, we however focus on smaller dimensions, where both methods can be run for comparison.

The proposed transform was implemented in Julia (cf. Appendix D), using the fast Hadamard transform (FFHT) library [225] as a core component. Other implementations of the fast Walsh-Hadamard transform exist, see e.g. McKernel [226] or the older Spiral project[10]. The FFHT library seemed to be the most efficient as it relies on SSE4 and AVX primitives[11], and comes under the MIT license.

[10] see https://www.spiral.net/software/wht.html

[11] These are instruction sets to perform vectorized operations.

5.3.1 Runtime speedup

We measure the runtimes on the compressive k-means task, with and without fast transforms. Details regarding data generation are discussed below as they do not impact the runtimes too much.

Note that, although the sketching process can be very easily parallelized, adopting a consistent parallelization scheme which would be efficient in all possible scenarios is not straightforward at all and would require more engineering. After a few trials, we thus decided to stay with a single-thread execution, and measure all the runtimes using a single core. For a fair comparison with the dense matrix product, we thus set the BLAS[12] number of threads to 1. We compiled the FFHT library with both AVX enabled and disabled, using SSE4 instructions in the latter case. The matrix-vector and matrix-matrix products which are used when relying on a dense linear operator are the BLAS[13] ones and most certainly rely on vectorization as well, although we do not mention it in the legends for conciseness.

[12] The Basic Linear Algebra Subprograms are routines for linear algebra operations.

[13] Openblas in our case.

Sketching times Results are presented in Figure 5.6 for different sketching batch sizes n_b (i.e., data is sketched by blocks of n_b samples at once) and discussed just below.

As we do not expect any computational gain for small dimensions, we consider values of d in the range $[32, 4096]$. The difference between runtimes obtained with structured transforms compared to using dense matrices are significant for high-dimensional data, and especially when sketching the dataset sample by sample (i.e. $n_b = 1$). In the following, we refer to this sketching scenario as the "streaming" case. Note

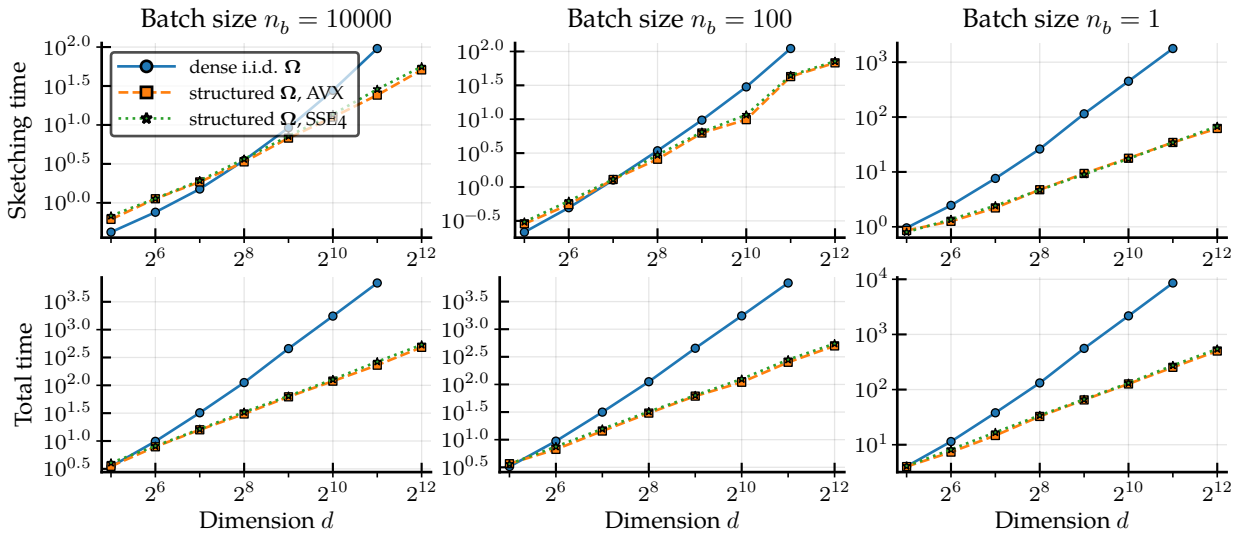


Figure 5.4: Sketching and total (sketching + learning) runtimes for multiple sketching batch sizes using dense or structured frequency matrices. Synthetic data, $k = 10, n = 10^4$. Median times over 100 iterations, obtained on an Intel(R) Xeon(R) CPU E5-2450 @ 2.10GHz.

however that doing so is useful not only for streaming data but also when memory is limited, as the sketching memory cost scales at least in $\Omega(mn_b)$[14]. For larger batch sizes, using fast transforms might not be useful for small dimensions (the BLAS matrix-matrix product used to compute $\mathbf{\Omega}^T\mathbf{B}$ for every batch $\mathbf{B}$ being too optimized to compete with it), but still allows us to deal with high-dimensional datasets.

We now give in Figure 5.5 the results in terms of speedups, that is as ratios of runtimes without and with fast transforms, and for two different architectures.

[14] This is the cost of just storing $\mathbf{\Omega}^T\mathbf{B}$, where $\mathbf{B}$ is a batch of data.

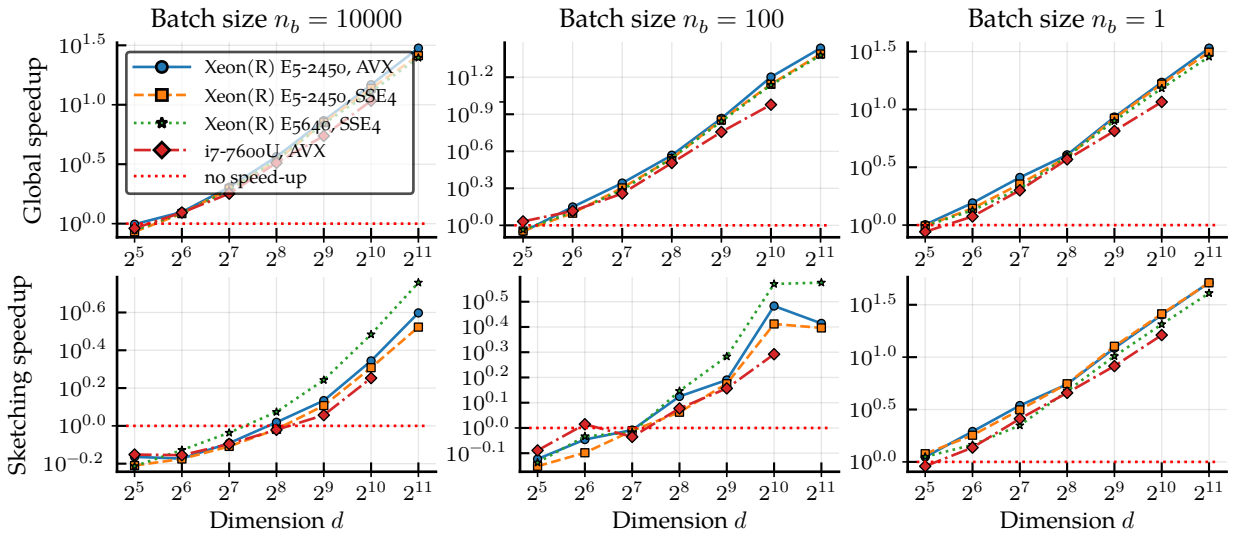


Figure 5.5: Sketching and global (sketching + learning) speedups (ratios of median runtimes over 100 trials) for multiple batch sizes on different architectures. Synthetic data, $k = 10, n = 10^4$.

In the streaming scenario and for the considered architecture, we get speedups of one order of magnitude as soon as $d = 512$. Note that the AVX instruction set brings only a very relative improvement over SSE4 on the considered architectures.

Learning time The learning phase benefits similarly from using fast transforms as shown in Figure 5.6. Results are obtained with the CL-

OMPR algorithm. Note that the learning time does not depend on the batch size.

The fact that speedups are always greater than one, even in small dimensions, is explained by the iterative structure of the CL-OMPR algorithm. We recall that atoms are added one by one (cf. Algorithm 2.1), and if $\mathbf{C}$ denotes the matrix whose columns are the selected atoms at the i-th iteration (thus having i columns), the algorithm only performs products of the kind $\mathbf{\Omega}^T\mathbf{C}$, for which no specific gains should be expected when using the matrix-matrix BLAS multiplication because of the small size of $\mathbf{C}$ – similarly to the online sketching scenario.

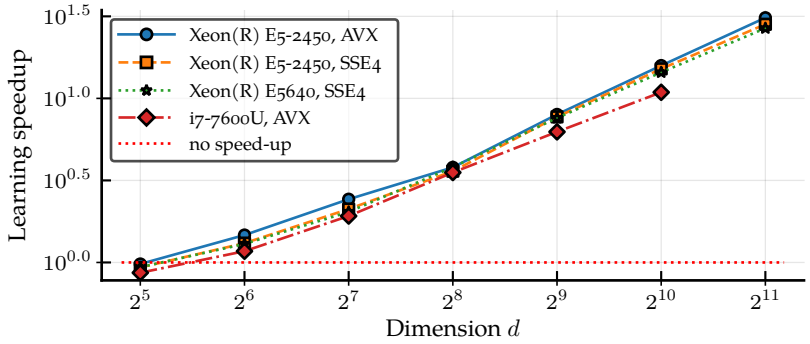


Figure 5.6: Learning speedups (ratios of median runtimes over 100 trials) using structured over dense linear operators. Same experimental setup as in Figure 5.5.

Different behavior might be observed for large values of k, but studying this matter would not be very relevant as CL-OMPR scales in $\mathcal{O}(k^3)$. We refer the reader to Chapter 6 for the alternative CL-AMP algorithm scaling with the number of clusters k in $\Theta(k^2)$.

Nonlinearity In order to fully understand the benefit of reducing the runtime of the linear operation, one must also take in consideration the runtime induced by the nonlinear step. The computational complexity of applying the nonlinear operations to the whole n samples is linear in d, while we recall that applying the linear operation scales in $\Theta(d \log(d))$ with fast transforms or $\Theta(d^2)$ without. We measured the runtimes of both – linear and non-linear – steps and report them in Figure 5.7. We recall that random Fourier features (RFF) rely on the nonlinear function $\mathrm{cis}(x) \triangleq \exp(ix)$. For reference, we also consider using quantized Fourier features (qRFF in the legends) as proposed in [104].

We use the standard `cis` function from the Julia math library, and our own implementation of its quantized version using four `floor` operations. Note that both functions could here again certainly be optimized. As can be seen on the left sub-figure, for small dimensions the runtime induced by the linear operation with a dense matrix is not only smaller that the one obtained with a structured operator (as discussed above, this is expected in low dimension), but also much smaller that the time spent on the nonlinear computations. With fast transforms, the linear and non-linear operations roughly take the same time for all dimensions. We show on the right subfigure the proportion of time spent on computing the nonlinearity, which confirms this. We get for

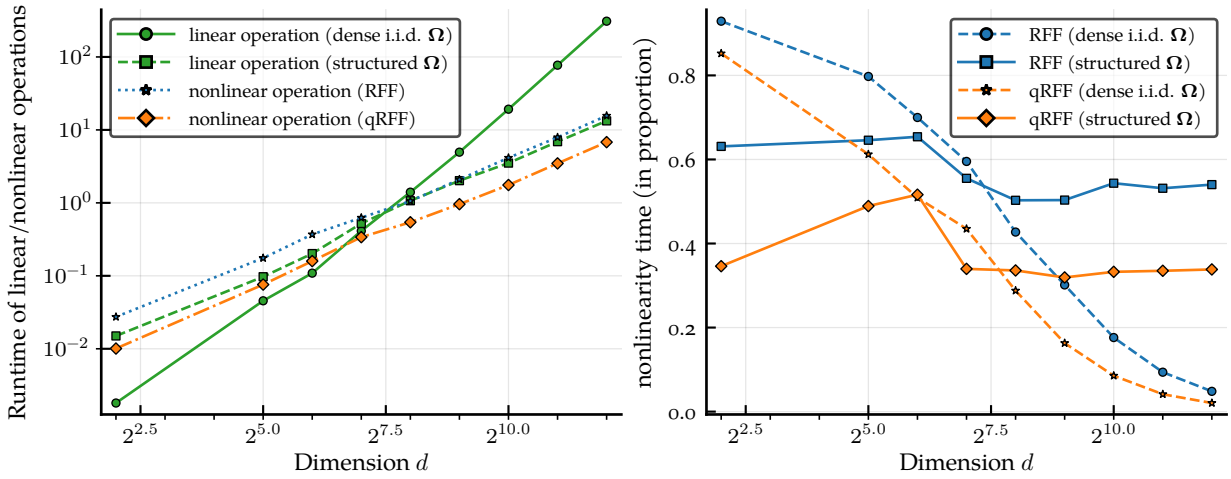


Figure 5.7: Runtimes (left) and relative runtimes (right) of linear and non-linear operations used for sketching as a function of the dimension d, using a sketch size $m = 10d$ and $n = 10^4$. The relative runtimes (right) are ratios of nonlinearity and total runtime (e.g. 0.6 means that 60% of the total runtime was spent on computing the nonlinearity).

higher dimensions proportions of the order 35% and 55% with/without quantization. Optimizing – or changing, if possible – the nonlinear function therefore seems to be essential for further improvements.

5.3.2 Clustering performance

We now take a closer look at the clustering quality obtained when using fast transforms. We consider both synthetic and real data, and error is measured with the mean squared error (MSE) introduced in Chapter 4, i.e. the clustering risk. From now on, we denote "CKM" the standard compressive k-means approach with dense matrices, and "FCKM" stands for fast compressive k-means, i.e. compressive k-means using fast transforms ($\mathbf{M}_{R^3}$ unless otherwise specified). In both case, the reconstruction is performed using the CL-OMPR algorithm. K-means results are all obtained using the Julia "Clustering" package[15]. The k-means++ initialization strategy is always used, although we omit the "++" in the legends.

[15] code available at https://github.com/JuliaStats/Clustering.jl

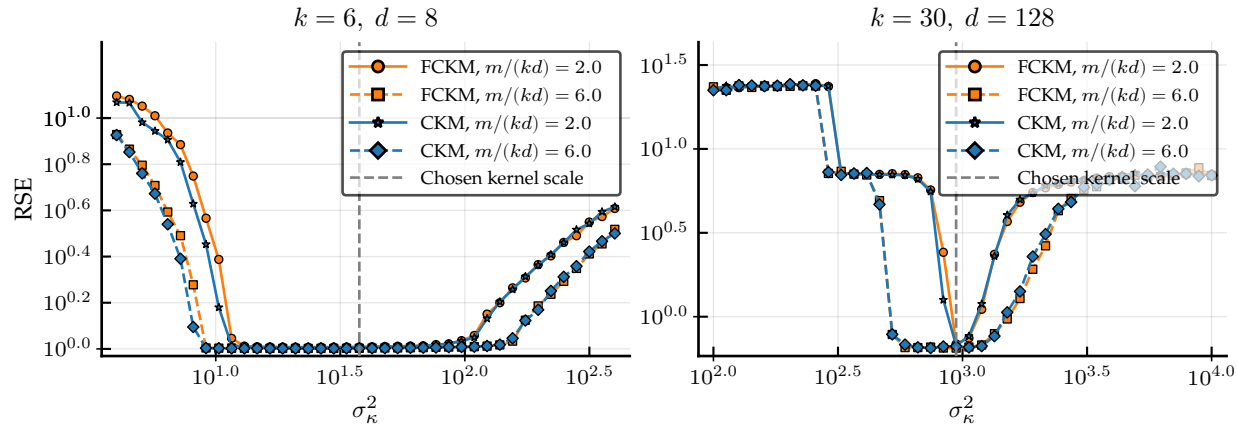


Figure 5.8: Clustering MSE using dense (CKM) and structured (FCKM) matrix $\mathbf{\Omega}$. Synthetic data with $\rho^2 = 10$, $n = 10^4$. Medians over 300 trials.

Synthetic data We first provide experiments on synthetic data, generated according to (4.5). We consider two different settings – namely, $(k = 6, d = 8)$ and $(k = 30, d = 128)$. Errors are reported in Figure 5.8

as functions of the kernel scale. We observe that the mean squared errors obtained with structured transforms follow very closely the ones obtained with dense matrices across the range of considered kernel scales on these datasets.

We now choose a single kernel scale in the range where optimal results are obtained (grey dashed lines in Figure 5.8) and give a violin plot[16] of the MSE corresponding to this scale only in Figure 5.9.

[16] Such plots provide a kernel density estimation of the probability density function of the data.

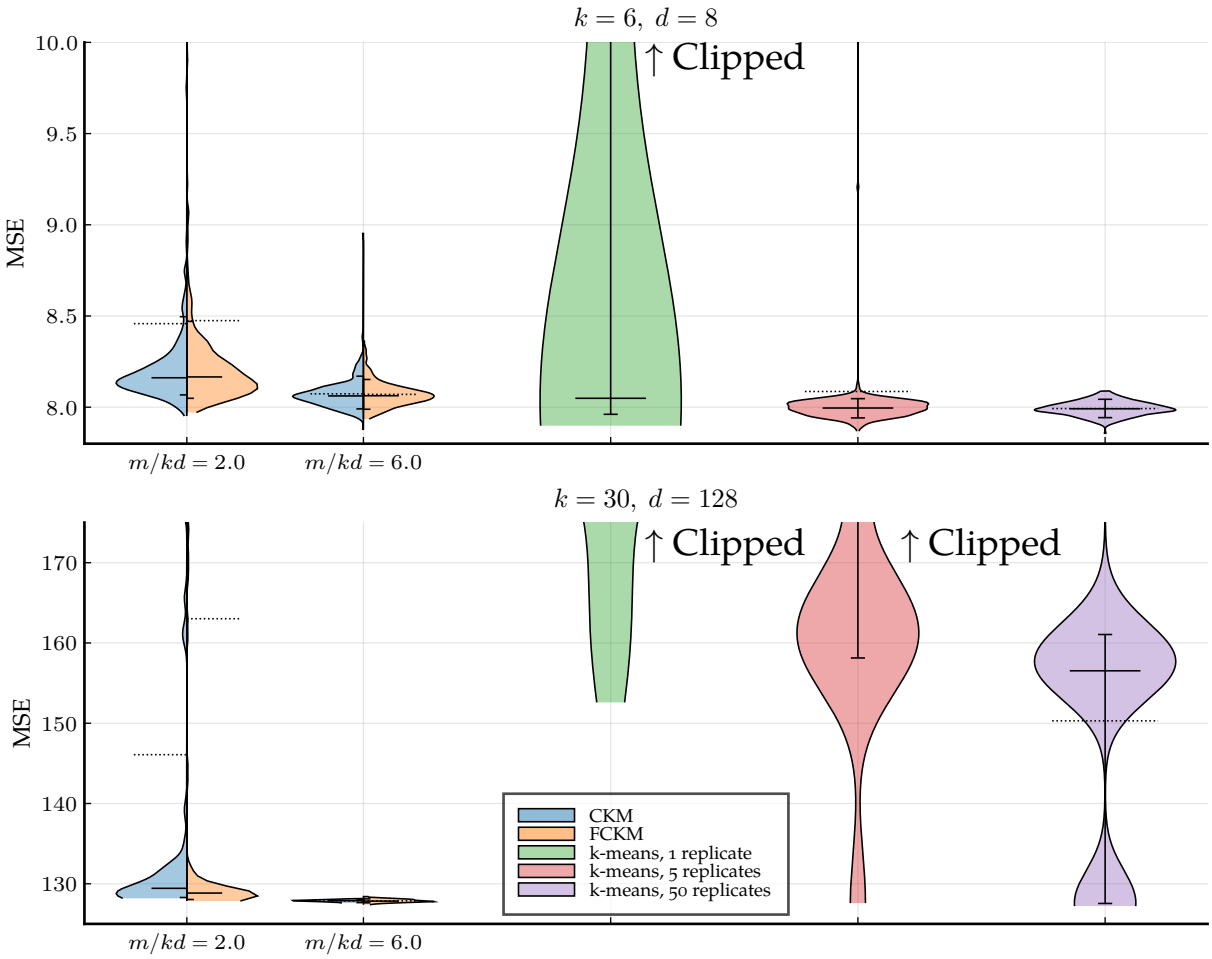


Figure 5.9: Violin plot with means (dotted black lines), medians (long solid black lines) and first/last deciles (short solid black lines) of clustering MSE using CKM, FCKM, and k-means++ with 1 and 5 replicates. Same experimental setting as in Figure 5.8, using the optimal kernel variance. 300 trials per group.

For k-means, "x replicates" in the legend means that the algorithm is run x times and the best solution in terms of MSE is kept[17]. For the setting $k = 6, d = 8$, the MSE obtained with CKM and FCKM follow very similar distributions, and no significant difference can be observed. On the bottom figure ($k = 30, d = 128$), a few more outlier appear when using FCKM compared to CKM with $m = 2kd$ as can be seen by looking at the means (last deciles are outside of the plotted range), but the difference is again shallow and the median values are the same for both methods. More importantly, both methods produce very well concentrated results for $m = 6kd$, whereas k-means is less stable and suffers from local optima, even using 50 replicates.

[17] Which is computationally expensive, and only possible if one can access the data multiple times.

MNIST features We now propose to cluster the MNIST dataset [227] of handwritten digits, which has $k = 10$ classes (one for each digit).

The original dataset contains $n = 70000$ pictures. Distorted variants of these images have been generated using infiMNIST [228], so that two other datasets of sizes $n = 3.10^5$ and $n = 1.10^6$ have been used for evaluation as well; these results are omitted here as the conclusions were similar.

As clustering in the image domain would be pointless, we used spectral features [229] computed as follows. Dense SIFT descriptors [230] are extracted for each image, and concatenated to form a single descriptor. The weighted k-nearest neighbours adjacency matrix is computed using FLANN[18]. The k eigenvectors associated to the k smallest positive eigenvalues of the Laplacian matrix[19] are used as features. Note that computing this matrix is very expensive for large collections[20]: we use this approach here as we focus on the clustering performance of fast transforms, and because the collection is still of a moderate size, but this framework would not be suitable for large-scale applications. We get n vectors of features in dimension $d = k = 10$, that can be clustered using a standard k-means algorithm. Note that with this small dimension, one should not expect to get a speed-up on the execution time using structured matrices, and we perform this experiment solely for an analysis of clustering results. Results are presented in Figure 5.10.

[18] see https://github.com/mariusmuja/flann

[19] The Laplacian matrix of a graph is a standard tool for graph signal processing. We refer the interested reader to [231] for a precise definition.

[20] And more expensive that the compressive clustering step.

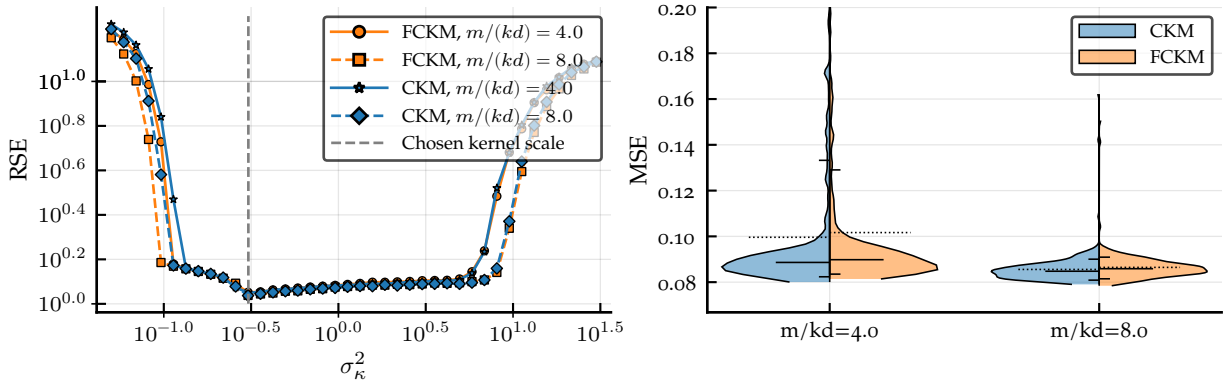


Figure 5.10: Clustering MSE using CKM and FCKM, as a function of the kernel variance (left) and as a violin plot for a fixed scale (right). The chosen kernel variance for the violin plot is the one corresponding to the dashed grey line on the left figure. Spectral features of dimension $d = 10$ computed on the MNIST dataset, $k = 10, n = 7 \times 10^4$.

We observe almost identical results using dense and structured operators. On the violin plot of the right figure, on can see that mean and median MSEs are slightly lower using a dense transform, but the difference is minor. On the other hand, on the left figure fast transforms seem to yield acceptable results over a range of kernel scales that is very slightly larger than when using dense matrices; here again, the difference is quite small and both method achieve overall similar clustering quality.

FMA dataset We also report clustering performance and runtimes on the FMA dataset[21] of audio features in Figure 5.11.

The data is centered and normalized as it contains features of very different natures. As in previous experiments, almost no difference can be seen in terms of clustering error between CKM and FCKM. Runtimes are measured using a single core for a fair comparison as explained in Section 5.3.1. The gain obtained when using structured features is almost of one order of magnitude for the learning phase,

[21] See https://github.com/mdeff/fma.

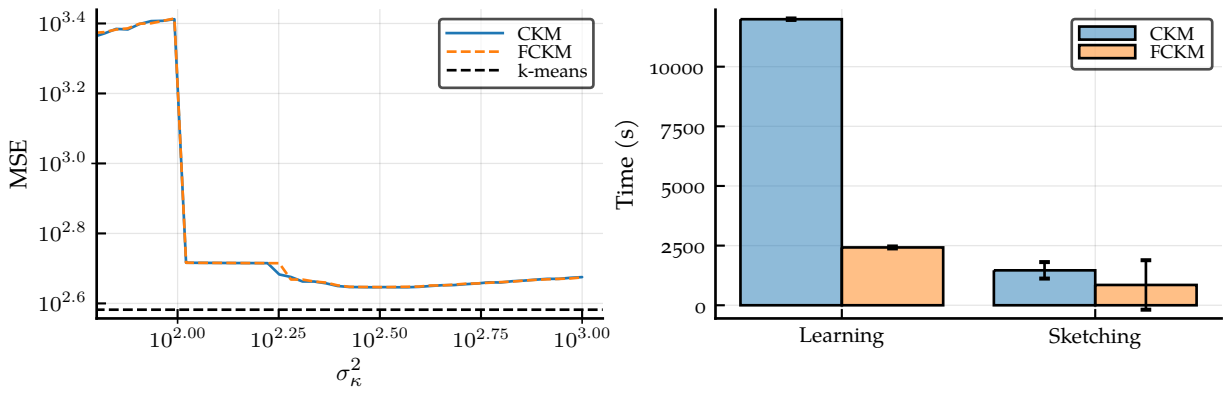


Figure 5.11: Clustering performance (left, medians over 10 trials) and runtimes (right, 10 trials) on the (centered and standardized) FMA dataset. $d = 518$, $k = 20$, $m = 10kd$.

and more moderate on the sketching phase where large batches are used (as observed on synthetic data in Section 5.3.1).

Comparison of the different structures We report in Figure 5.12 the clustering results obtained with the different types of squared blocks mentioned in Section 5.2.2. It appears that the three considered constructions yield highly similar results. The block $\mathbf{M}_{R^3}^T$ seems to be perform slightly worse for small values of σ_κ^2 in comparison to the two other in small dimension, but the difference is very subtle.

On the violin plots (right side of the figure), the $\mathbf{M}_{R^3}^T$ block seems on the contrary to yield slightly better results compared to the other structured blocks, and also in comparison with the dense design (if we look at means and medians at least, as the dense design seems to reduce the number of outliers in comparison with structured designs). However, the difference is here again too small to be significant, and overall the three structured designs can be considered to perform equally well on this dataset.

5.3.3 Hierarchical clustering on a co-purchasing graph

We propose to test our method on a graph spectral clustering task. This consists in leveraging the structure of the graph to produce numerical features for all the nodes, which can then be clustered. Here again, we use FCKM solely to assess the clustering performance and stability, but the features themselves are chosen of moderate size, so that no particular speedup should be expected with FCKM.

Computing spectral features involves computing the eigendecomposition of the Laplacian matrix of the graph, which is usually expensive. Tremblay et al. proposed to bypass this step using as features $\Theta(\log k)$ filtered random graph signals on the graph associated to the Laplacian matrix [231]. Standard k-means is then applied on a random subset (for efficiency) of these features, and interpolated to the whole collection. We propose to combine these fast random features with the FCKM framework, which allows to avoid the subsampling step.

We work on the Amazon co-purchasing network [232], which is a graph comprising $n = 334863$ nodes and $E = 925872$ edges. As there is no ground truth for this dataset, we used $k = 64, 128$ and

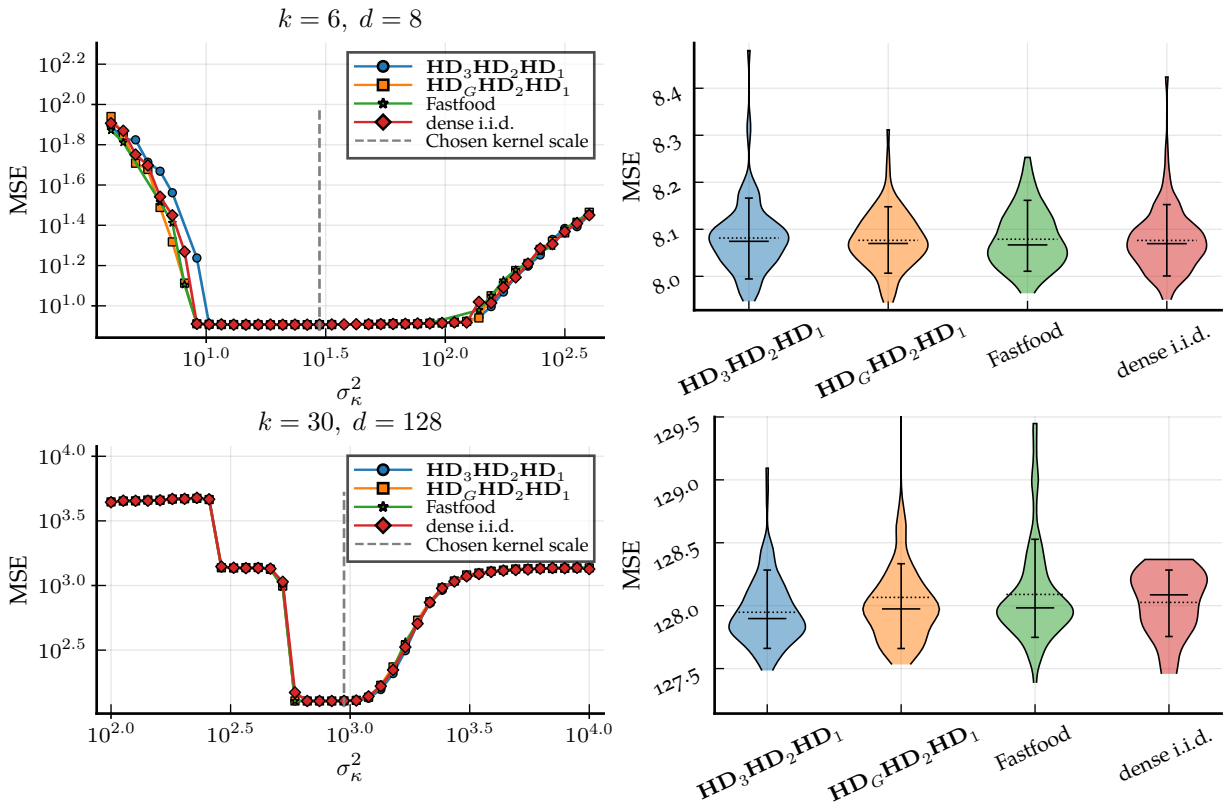
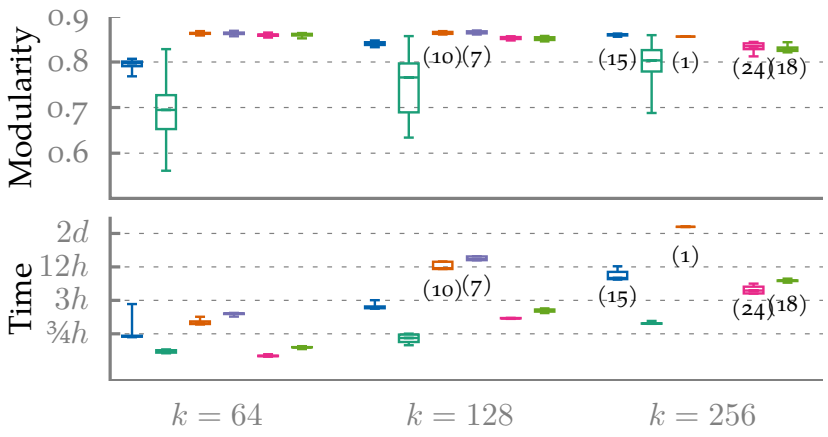


Figure 5.12: Clustering error for the different structured blocks, as a function of the kernel scale on the left (medians over 100 trials), and for a (manually) fixed scale on the right (violin plots from 100 trials, with first/last deciles and median in plain line, mean in dotted line). Using $m = 4kd$, synthetic data following (4.5) with $\rho^2 = 10$. Dashed lines on the left indicate the kernel scales used for the violin plots.

256. We compare the original spectral clustering (SC), compressive spectral clustering (CSC) [231], and 4 methods using sketching on the same random features: we combine the two types of matrices (dense/structured) with the two learning procedures – CL-OMPR, given in Algorithm 2.1, and the hierarchical approach that we do not detail here [4, Algorithm 2]. Results are provided in Figure 5.13, with a summary of the different methods in the table below. We measure the clustering quality with the modularity metric [233], which leverages the structure of the graph to assess the quality of the produced cluster (the higher the better, the maximum possible value is 1 and the expectation of the modularity of a random clustering is 0).

Standard k-means is launched with 2 replicates. Using compressive spectral clustering, the k-means step is performed only on a subset of the features and is therefore much faster; we used 20 replicates for a fair comparison. We used $m = 10kd$ for the sketch size when using CKM. All initializations were performed uniformly. As regards the elapsed times, CL-OMPR is not competitive for large values of k[22] but satisfying results are obtained with the hierarchical algorithm. Similar modularities are obtained with and without structured matrices; the results are slightly lower when using the hierarchical algorithm, but in both cases they are highly concentrated, whereas standard CSC yields a high variance. This likely results from the subsampling step, which is the only difference between CSC and the other approaches. We can imagine that CSC sometimes subsamples too harshly the data and thus misses some of the clusters.

[22] which is expected due to its $\Theta(k^3)$ complexity



	Method	Features	Subs.	Sk. matrix	Clustering
—	SC	spectral	No	n/a	KM
—	CSC	random	Yes	n/a	KM
—	S2C	random	No	Dense	CL-OMPR
—	FS2C	random	No	Structured	CL-OMPR
—	HS2C	random	No	Dense	Hierarchical
—	HFS2C	random	No	Structured	Hierarchical

Figure 5.13: Boxplots of modularity (the higher the better) and clustering time for $k = 64, 128, 256$. Only the learning times are displayed for sketching methods; sketching times are much smaller, even on a single core. All experiments were run on Intel(R) Xeon(R) CPUs E5640 and repeated 30 times (or less for experiments that were too long; the number of iterations is indicated below in these cases, and FS2C does not appear for $k = 256$). Owing to the long runtimes of the different methods, our goal here is not to get statistically significant results, but rather and idea of how the different methods compare. The table is a summary of the different methods (in the order of the boxplots, from left to right). This experiment was produced with an older code base relying on the sketchML Matlab toolbox [234], and was not ran again because of the long runtimes.

5.4 Towards theoretical guarantees

Experiments conducted in the previous sections suggest that using structured matrices does not degrade the learning performance, at least for the clustering task on which we focused for conciseness[23]. In this section, we shall try to investigate how existing statistical learning guarantees mentioned in Chapter 2, such as the ones for compressive clustering, could be adapted when using structured matrices.

We first review in Section 5.4.1 the structure of existing guarantees, and prove in Section 5.4.2 that some of the blocks induce a Gaussian kernel, which allow to reuse a substantial part of existing results. We then discuss in Section 5.4.3 how concentration results can be adapted when using structured matrices.

[23] We will see in Part IV that similar observations can be made for PCA. We expect that the same holds for density fitting, although we did not conduct the experiments. Moreover, in this case the structure of the blocks cannot be fully leveraged during the learning phase.

5.4.1 Adapting existing results

We recall[24] that existing guarantees for compressive learning methods have been obtained by establishing a lower restricted isometry property (LRIP) of the kind

$$\forall \tau, \tau' \in \mathfrak{S},\ \|\tau - \tau'\|_{\Delta\mathcal{L}(\mathcal{H})} \leq C_{\kappa_\Phi} \|\mathcal{A}(\tau) - \mathcal{A}(\tau')\|_2 \qquad (5.9)$$

for a well-chosen model set $\mathfrak{S}$. In the following, we denote κ_Φ the kernel associated to a specific feature map Φ and κ the associated mean kernel, i.e.

$$\kappa_\Phi(\mathbf{x}, \mathbf{y}) = \frac{1}{m}\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle \quad \text{and} \quad \kappa(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{E}_\Phi \kappa_\Phi(\mathbf{x}, \mathbf{y}), \qquad (5.10)$$

where the expectation on Φ denotes the expectation on the randomness of Φ which includes the draw of $\boldsymbol{\Omega}$, should it be using i.i.d. or structured features. The same notations are used for the implicit extensions of these kernels to distributions (cf. (2.27)).

[24] Cf. Section 2.2 for the general methodology.

We also denote

$$\|\pi_1 - \pi_2\|^2_{\kappa_\Phi} \triangleq \frac{1}{m}\|\mathcal{A}(\pi_1) - \mathcal{A}(\pi_2)\|_2^2 \tag{5.11}$$

$$= \frac{1}{m}\sum_{j=1}^{m} |\langle \pi_1, \phi_{\boldsymbol{\omega}_j}\rangle - \langle \pi_2, \phi_{\boldsymbol{\omega}_j}\rangle|^2, \tag{5.12}$$

where $\phi_{\boldsymbol{\omega}_j}$ denotes the feature function associated to the frequency $\boldsymbol{\omega}_j$, and using the inner-product notation (cf. Appendix A.2). As explained in Section 2.3.3, this quantity can be interpreted as the empirical version of the (squared) MMD, and extended to the set of finite signed measures.

Structure of the proof As briefly discussed in Section 2.3.4, one way to establish the relation (5.9) is to first prove that a LRIP holds w.r.t. the pseudo-norm $\|\cdot\|_\kappa$ associated to the mean kernel, i.e.

$$\forall \tau, \tau' \in \mathfrak{S},\ \|\tau - \tau'\|_{\Delta\mathcal{L}(\mathcal{H})} \leq C_\kappa \|\tau - \tau'\|_\kappa \tag{5.13}$$

for some $C_\kappa < \infty$, and then to prove that for any $0 < \delta < 1$,

$$\forall \tau, \tau' \in \mathfrak{S},\ 1 - \delta \leq \frac{\|\tau - \tau'\|^2_{\kappa_\Phi}}{\|\tau - \tau'\|^2_\kappa} \leq 1 + \delta \tag{5.14}$$

holds with high probability on the draw of Φ provided m is large enough. Indeed, when both Equation (5.13) and (5.14) hold (with respectively constants C_κ and δ), then (5.9) holds as well with constant $C_{\kappa_\Phi} = C_\kappa(1-\delta)^{-1/2}$. These two parts of the proof call for different tools.

LRIP for the mean kernel In sight of (5.13), it is useful to introduce the *normalized secant set* $\mathcal{S}_\kappa(\mathfrak{S})$ of $\mathfrak{S}$, defined as

$$\mathcal{S}_\kappa(\mathfrak{S}) \triangleq \left\{ \frac{\tau - \tau'}{\|\tau - \tau'\|_\kappa} | \tau, \tau' \in \mathfrak{S}, \|\tau - \tau'\|_\kappa > 0 \right\}. \tag{5.15}$$

Using the implicit extension of the semi-norm $\|\cdot\|_{\Delta\mathcal{L}(\mathcal{H})}$ to finite signed measures[25], the lowest constant C_κ such that (5.13) holds can be written $C_\kappa = \sup_{\mu \in \mathcal{S}_\kappa(\mathfrak{S})} \|\mu\|_{\Delta\mathcal{L}(\mathcal{H})}$.

This quantity can be upper-bounded under some assumptions[26] on the kernel. Bounds have in particular been computed for the Gaussian kernel w.r.t. the models of mixtures of separated[27] and bounded Diracs, and mixtures of Gaussian distributions with fixed covariances and separated and bounded centers. We refer to [6, Theorem 5.10], and to appendix D.2 of the same paper. We do not go deeper into the elements of the proof, but stress that (5.13) only depends on the mean kernel, and thus existing results can be reused for any construction of the feature map which yields a Gaussian kernel in expectation. We will see in Section 5.4.2 that this holds in particular for random Fourier features obtained using structured $\mathbf{M}_{\text{ff}}$ or $\mathbf{M}_{GR^2}$ blocks.

[25] cf. Appendix A.2 for more details

[26] Especially a "coherence" assumption.

[27] We will see in Chapter 10 that such a minimum separation hypothesis is necessary for the LRIP to hold.

Concentration The concentration property (5.14), on the other side, depends explicitly on the nature of the features. Using again the definition of the secant set, it is equivalent to showing (for m large enough) that with high probability on the draw of Φ,

$$\forall \mu \in \mathcal{S}_\kappa, \left| \|\mu\|^2_{\kappa_\Phi} - 1 \right| \leq \delta. \tag{5.16}$$

This result can be established by first proving a pointwise concentration result for any fixed $\mu \in \mathcal{S}_\kappa(\mathfrak{S})$, and then generalized to a uniform result by controlling the covering number[28] of the normalized secant set for some well-chosen metric. Here again, the covering numbers only depend on the chosen metric and on the mean kernel (via $\mathcal{S}_\kappa(\mathfrak{S})$). In particular, existing results have been obtained for the metric

$$d(\pi_1, \pi_2) = \sup_{\boldsymbol{\omega}} \left| |\langle \pi_1, \phi_{\boldsymbol{\omega}} \rangle|^2 - |\langle \pi_2, \phi_{\boldsymbol{\omega}} \rangle|^2 \right|,$$

where the supremum over $\boldsymbol{\omega}$ is implicitly taken over the support of the chosen distribution, which is $\mathbb{R}^d$ both for dense frequencies and when using $\mathbf{M}_{GR^2}$ and $\mathbf{M}_{\text{ff}}$ blocks. Hence existing results can be leveraged for features maps which yield a Gaussian mean kernel; in this case, only a pointwise concentration result needs to be established. In the case of $\mathbf{M}_{R^3}$ blocks, the support of the marginal frequency distribution is not $\mathbb{R}^d$ and computing covering numbers precisely might be more challenging, however the distance obtained when taking the supremum over $\mathbb{R}^d$ can still be used to obtain an upper-bound on the desired covering number.

[28] The covering number of a set is the minimum number of elementary balls needed to cover entirely this set.

A note on existing results on structured features As the structured blocks used in this chapter have been used in other contexts in the literature, some theoretical properties have been established. However, these results cannot directly be leveraged in our setting. In particular, most concentration results hold for the kernel itself, but are not particularly useful in order to establish (5.16). In the case of $\mathbf{M}_{R^3}$ blocks, specific results have been developed for cross-polytope LSH [220], but rely on specific assumptions which are not satisfied in our case.

5.4.2 Induced kernels

We first justify the normalization factors used in the definition of the square blocks, and then prove in Lemma 5.6 that the mean kernels associated to random Fourier features when using the $\mathbf{M}_{GR^2}$ and $\mathbf{M}_{\text{ff}}$ blocks are Gaussian.

Lemma 5.4

The blocks $\mathbf{M}_{\text{ff}}^T$, $\mathbf{M}_{GR^2}^T$ and $\mathbf{M}_{R^3}^T$ always have normalized rows.

Proof: We deal with the three settings separately.

- For $\mathbf{M}_{R^3}^T$, as $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$ contain Rademacher entries on their diagonals, we have $\forall i \in \{1, 2, 3\}\ \mathbf{D}_i \mathbf{D}_i^T = \mathbf{I}_d$. Furthermore, the

Walsh-Hadamard matrix satisfies $\mathbf{H}\mathbf{H}^T = d\mathbf{I}_d$, hence

$$(\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1)(\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1)^T = d^3\mathbf{I}_d$$

which justifies the normalization by $d^{3/2}$.

- In the case of $\mathbf{M}_{\text{ff}}^T$, we observe[29] for any draw of $\mathbf{D}_R, \mathbf{\Pi}, \mathbf{D}_G$ that

$$(\mathbf{H}\mathbf{D}_G\mathbf{\Pi}\mathbf{H}\mathbf{D}_R)(\mathbf{H}\mathbf{D}_G\mathbf{\Pi}\mathbf{H}\mathbf{D}_R)^T = d\mathbf{H}\mathbf{D}_G^2\mathbf{H}^T,$$

  whose diagonal elements all take the value $d\|\mathbf{D}_G\|_F^2$ as every entry H_{ij} of $\mathbf{H}$ satisfies $H_{ij}^2 = 1$. Hence we normalize each block by $d^{1/2}\|\mathbf{D}_G\|_F$.
- For $\mathbf{M}_{GR^2}^T$, we have for the same reasons

$$(\mathbf{H}\mathbf{D}_G\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1)(\mathbf{H}\mathbf{D}_G\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1)^T = d^2\mathbf{H}\mathbf{D}_G^2\mathbf{H}^T,$$

  for any draw of $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_G$, and thus the normalization factor yielding unit rows is $d\|\mathbf{D}_G\|_F$.

29. This result was already in [77], although the normalization factor $d^{1/2}\|\mathbf{D}_G\|_F^{1/2}$ given at the end of the proof seems to be a misprint.

Lemma 5.5

The marginal distribution of any row of a block $\mathbf{M}_{\text{ff}}^T$ or $\mathbf{M}_{GR^2}^T$ is uniform on the unit sphere.

Proof: By Lemma 5.4, the rows of the blocks are on the unit sphere. To prove uniformity, we will show that all rows of the unnormalized[30] blocks $\mathbf{U}_1 = \mathbf{H}\mathbf{D}_G(d^{-1/2}\mathbf{\Pi}\mathbf{H}\mathbf{D}_R)$ or $\mathbf{U}_2 \triangleq \mathbf{H}\mathbf{D}_G(d^{-1}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1)$ have i.i.d. normal entries. Indeed, if a vector $\mathbf{x}$ is a standard normal random vector, i.e. $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, then $\mathbf{y} = \mathbf{x}/\|\mathbf{x}\|_2$ is on the unit sphere and invariant by rotation[31], and thus uniformly distributed on the unit sphere.

Let $\mathbf{h}_i$ denote the i-th column of $\mathbf{H}$. For any $i \in [\![1, d]\!]$, $\mathbf{v}_i^T \triangleq \mathbf{h}_i^T\mathbf{D}_G$ is a vector of i.i.d. standard normal random variables (as $\mathbf{D}_G$ itself has i.i.d. standard normal entries).

Note that the i-th row $\mathbf{u}_i^T$ of both $\mathbf{U} = \mathbf{U}_1$ and $\mathbf{U} = \mathbf{U}_2$ can be written $\mathbf{u}_i^T = \mathbf{v}_i^T\mathbf{O}$ with $\mathbf{O} \in O(d)$ independent from $\mathbf{v}_i$, where $O(d)$ denotes the orthogonal group. Hence $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{O}\mathbf{O}^T = \mathbf{I}_d)$ as a linear transformation of $\mathbf{v}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. This proves that all the entries of $\mathbf{u}_i$ are normal and identically distributed (but this does not prove independence).

The fact that $\mathbf{O}$ is orthogonal and independent of $\mathbf{v}_i$ implies that the entries of $\mathbf{u}_i$ are independent using a result from Lukacs and King[32] [235] (which can be applied here given that all the moments of all the entries of $\mathbf{v}_i$ exist), which concludes the proof.

30. More precisely, we keep the constant normalization factor, but omit the $\|\mathbf{D}_G\|_F^{-1}$ term.

31. For any orthogonal matrix $\mathbf{O} \in O(d)$, where $O(d)$ denotes the orthogonal group, $\mathbf{O}\mathbf{x}/\|\mathbf{x}\|_2 = \mathbf{O}\mathbf{x}/\|\mathbf{O}\mathbf{x}\|_2$, and $\mathbf{O}\mathbf{x}$ and $\mathbf{x}$ are identically distributed by invariance to rotation of the multivariate normal distribution.

32. Note that orthogonality of $\mathbf{O}$ ensures that the entries of $\mathbf{u}_i$ are not correlated, but this does not imply independence contrarily to what is stated in [77, Section 3.2], even using the fact that $\mathbf{v}_i$ has normal entries (counter-example: $X \sim \mathcal{N}(0, 1)$ and $Y = RX$ with R following a Rademacher distribution).

It is important to note that $\mathbf{M}_{R^3}^T$ does not satisfy this property. Indeed, as all entries of $\mathbf{H}, \mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$ are in $\{-1, 0, +1\}$, the distribution of the rows of an $\mathbf{M}_{R^3}^T$ block is discrete. This phenomenon can be observed on Figure 5.14 which shows the distribution of $\boldsymbol{\omega}^T\mathbf{u}$ for a fixed vector $\mathbf{u}$ when $\boldsymbol{\omega}$ is either uniform on the unit sphere, or follows the

marginal column distribution induced by the block $\mathbf{M}_{R^3}$.

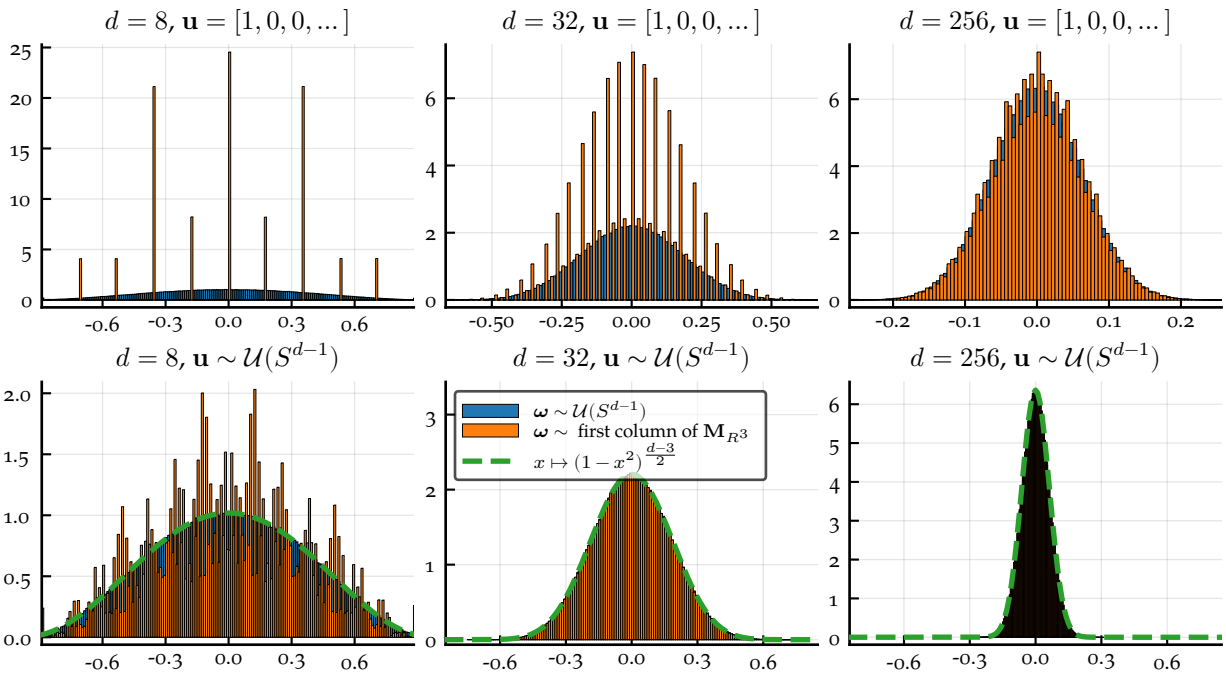


Figure 5.14: Distribution of $\boldsymbol{\omega}^T\mathbf{u}$ for both $\mathbf{u} = [1, 0, 0, ...]$ and one draw of $\mathbf{u} \sim \mathcal{U}(S^{d-1})$, when $\boldsymbol{\omega} \sim \mathcal{U}(S^{d-1})$ and $\boldsymbol{\omega}$ follows the marginal distribution of the first column of a $\mathbf{M}_{R^3}$ block. Histograms over $n = 10^6$ draws of $\boldsymbol{\omega}$ ($\mathbf{u}$ is fixed). The green dashed curve corresponds to the true probability density when $\boldsymbol{\omega} \sim \mathcal{U}(S^{d-1})$.

We can see that when $\mathbf{u}$ is the first vector of the canonical basis, then the distribution of $\boldsymbol{\omega}^T\mathbf{u}$ obtained when using a structured block is not only discrete (which is expected as $\boldsymbol{\omega}^T\mathbf{u}$ is the top left element of $\mathbf{M}_{R^3}$ in this case), but also quite different from what is obtained in the uniform setting. This is not necessarily detrimental for learning (as observed on clustering experiments), but makes the analysis of this setting more difficult.

A consequence of Lemma 5.5 is that the kernel induced by a feature map of random Fourier features when $\boldsymbol{\Omega}$ is made of $\mathbf{M}_{GR^2}$ or $\mathbf{M}_{\text{ff}}$ blocks is Gaussian. This is a relatively straightforward consequence of Bochner's theorem[33] when using full blocks[34]; we propose here a formulation of the result which takes into account padding operations.

[33] Cf. Theorem 2.3

[34] i.e. when d is a power of 2

Lemma 5.6

The kernel induced by the random Fourier feature map[35] $\mathbf{x} \mapsto \Phi^{\text{RFF}}(\tilde{\mathbf{x}})$, where $\tilde{\mathbf{x}} = [\mathbf{x}^T; \mathbf{0}]^T \in \mathbb{R}^{d_p}$ is the padded[36] version of $\mathbf{x}$ and when using a frequency matrix $\boldsymbol{\Omega} = \mathbf{MS} \in \mathbb{R}^{d_p \times m_p}$ where

- $\mathbf{S} \in \mathbb{R}^{m_p \times m_p}$ is a diagonal scaling matrix[37] with i.i.d. diagonal entries $S_{ii} = R_i/\sigma_\kappa$ where $R_i \overset{i.i.d.}{\sim} \chi_{d_p}$;
- $\mathbf{M} \in \mathbb{R}^{d_p \times m_p}$ is the concatenation of square i.i.d. $\mathbf{M}_{\text{ff}}$ or $\mathbf{M}_{GR^2}$ blocks (cf. definitions 5.2 and 5.3),

is an unbiased estimator of the Gaussian kernel, i.e.

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, \mathbf{E}_{\boldsymbol{\Omega}}\left[\frac{1}{m}\overline{\Phi^{\text{RFF}}(\tilde{\mathbf{x}})}^T\Phi^{\text{RFF}}(\tilde{\mathbf{y}})\right] = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma_\kappa^2}\right). \quad (5.17)$$

35. Cf. Definition 2.5.

36. We recall that $d_p \triangleq 2^{\lceil \log_2(d) \rceil}$.

37. As described in Section 5.2.1

Proof: Let $\mathbf{M} = [\mathbf{M}_1, ..., \mathbf{M}_b]$ be the decomposition in blocks of $\mathbf{M}$, where b denotes the number of blocks used in the construction. Let $\mathbf{S}_1, ..., \mathbf{S}_b$ denote the $d_p \times d_p$ square blocks of the diagonal of $\mathbf{S}$. Let $d_p = 2^{\lceil \log_2(d) \rceil}$ be the dimension of the square blocks[38]. Let

$$\kappa : \mathbf{x}, \mathbf{y} \mapsto \frac{1}{m} \overline{\Phi^{\text{RFF}}(\tilde{\mathbf{x}})}^T \Phi^{\text{RFF}}(\tilde{\mathbf{y}})$$

denote the kernel induced by the feature map, where $\tilde{\mathbf{x}}, \tilde{\mathbf{y}} \in \mathbb{R}^{d_p}$ are the zero-padded[39] versions of $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

We now assume for conciseness $m = m_p$, however note that subsampling the last block $\mathbf{M}_b$ as described in Section 5.2.3 does not change the result as all the rows of a random block $\mathbf{M}_b$ have the same marginal distribution[40]. By independence of the blocks, and by Lemma 5.5, we have

$$\begin{aligned}
\mathbf{E}_{\mathbf{\Omega}}[\kappa(\mathbf{y}, \mathbf{y})] &= \frac{1}{m} \mathbf{E}_{\mathbf{S}_1, ..., \mathbf{S}_b, \mathbf{M}_1, ..., \mathbf{M}_b} \sum_{i=1}^{b} \exp\big(\iota (\mathbf{M}_i \mathbf{S}_i)^T (\mathbf{x} - \mathbf{y})\big) \\
&= \frac{1}{d_p} \mathbf{E}_{\mathbf{S}_1, \mathbf{M}_1} \exp\big(\iota (\mathbf{M}_1 \mathbf{S}_1)^T (\tilde{\mathbf{x}} - \tilde{\mathbf{y}})\big) \\
&\overset{\text{(i)}}{=} \mathbf{E}_{\tilde{\boldsymbol{\varphi}} \sim \mathcal{U}(S^{d_p - 1}), R \sim \chi_{d_p}} \exp\left(\iota (\frac{R}{\sigma_\kappa} \tilde{\boldsymbol{\varphi}})^T (\tilde{\mathbf{x}} - \tilde{\mathbf{y}})\right) \\
&= \mathbf{E}_{\tilde{\boldsymbol{\omega}} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\sigma_\kappa^2} \mathbf{I}_{d_p})} \exp\big(\iota \tilde{\boldsymbol{\omega}}^T (\tilde{\mathbf{x}} - \tilde{\mathbf{y}})\big)
\end{aligned}$$

Let $\boldsymbol{\omega}$ denote the first d coordinates of $\tilde{\boldsymbol{\omega}}$. When $\tilde{\boldsymbol{\omega}} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\sigma_\kappa^2} \mathbf{I}_{d_p})$, we have $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\sigma_\kappa^2} \mathbf{I}_d)$, and hence

$$\begin{aligned}
\mathbf{E}_{\mathbf{\Omega}}[\kappa(\mathbf{x}, \mathbf{y})] &= \mathbf{E}_{\boldsymbol{\omega} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{\sigma_\kappa^2} \mathbf{I}_d\right)} \exp(\iota \boldsymbol{\omega}^T (\mathbf{x} - \mathbf{y})) \\
&\overset{\text{(ii)}}{=} \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{2\sigma_\kappa^2}\right).
\end{aligned}$$

38. We recall that this is required as the Walsh-Hadamard matrix exists only for powers of 2.

39. i.e. $\tilde{\mathbf{x}} = [\mathbf{x}^T, 0, ..., 0]^T$ with $d_p - d$ zeros.

40. By Lemma 5.5.

(i) By Lemma 5.5, which holds for each of the rows of the block.

(ii) By (2.25), which comes from Bochner's theorem (Theorem 2.3).

5.4.3 Concentration

The previous subsection established that random Fourier features computed using $\mathbf{M}_{GR^2}$ or $\mathbf{M}_{\text{ff}}$ blocks both induce a mean Gaussian kernel. In light of Section 5.4.1, the only missing element in order to control the excess risk is the pointwise concentration of $\|\cdot\|_{\kappa_\Phi}^2$ on the normalized secant set. That is, we need to show that for any $\delta > 0$ and $\mu \in \mathcal{S}_\kappa(\mathfrak{S})$,

$$\left| \|\mu\|_{\kappa_\Phi}^2 - 1 \right| \leq \delta. \tag{5.18}$$

holds with high probability on the draw of the feature map Φ. Indeed, if such a result holds for each $\mu \in \mathcal{S}_\kappa(\mathfrak{S})$, it can then be extended to the whole secant set via its covering number as discussed in Section 5.4.1. We thus discuss the strategies that could be used to establish (5.18) in this section.

In the following, we assume for simplicity that $m = bd$, that d is a power of 2, and that $\mathbf{\Omega}$ is drawn by concatenating b square blocks $(\mathbf{B}_i)_{1 \leq i \leq b}$ of d frequencies $(\boldsymbol{\omega}_j^i)_{1 \leq j \leq d}$ each. Using the inner-product

notation[41] and the characterization of the MMD[42], we have for any $\mu \in \mathcal{S}_\kappa(\mathfrak{S})$

$$\|\mu\|^2_{\kappa_\Phi} = \frac{1}{m}\sum_{i=1}^{b}\sum_{j=1}^{d}\left|\left\langle \mu, \phi_{\boldsymbol{\omega}_j^i}\right\rangle\right|^2 \tag{5.19}$$

$$= \frac{1}{b}\sum_{i=1}^{b} W(\mathbf{B}_i, \mu) \tag{5.20}$$

$$\text{where } W(\mathbf{B}_i, \mu) \triangleq \frac{1}{d}\sum_{j=1}^{m} Y(\boldsymbol{\omega}_j^i, \mu) \tag{5.21}$$

$$\text{and } Y(\boldsymbol{\omega}, \mu) \triangleq |\langle \mu, \phi_{\boldsymbol{\omega}}\rangle|^2. \tag{5.22}$$

Note in particular that the $W(\mathbf{B}_i, \mu)$ are independent, whereas the components which appear in (5.21) are not. This is in contrast with the setting where the frequency vectors $\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m$ are drawn independently, in which case

$$\|\mu\|^2_{\kappa_\Phi} = \frac{1}{m}\sum_{j=1}^{m} Y(\boldsymbol{\omega}_j, \mu). \tag{5.23}$$

with independent $(Y(\boldsymbol{\omega}_j, \mu))_{1\leq j\leq m}$. Previous works obtained concentration results in this setting and for the Gaussian kernel by leveraging this independence, either via the Bernstein theorem (thus bounding $Y(\boldsymbol{\omega}, \mu)$ for $\mu \in \mathcal{S}_\kappa(\mathfrak{S})$ and its variance) [5, Lemma 5.5], or by bounding higher moments of $Y(\boldsymbol{\omega}, \mu)$ [6, Lemma 5.11]. This yields in both settings a concentration of the form

$$P\Big[\Big|\|\mu\|^2_{\kappa_\Phi} - 1\Big| > t\Big] \leq 2\exp\left(-\frac{mt^2}{C_Y(1+t/2)}\right) \tag{5.24}$$

for some constant C_Y, which is naturally different in the two settings but always independent of the dimension d. For instance to apply the Bernstein theorem, C_Y is a bound on both $D_Y \triangleq \sup_{\boldsymbol{\omega}} Y(\boldsymbol{\omega}, \mu) - \inf_{\boldsymbol{\omega}} Y(\boldsymbol{\omega}, \mu)$ and on $\mathrm{Var}[Y(\boldsymbol{\omega}, \mu)]$.

In our case, similar concentration results can be applied on the variables $(W(\mathbf{B}_i, \mu))_{1\leq i\leq b}$ provided that the relevant quantities can be bounded, however the concentration will then increase proportionally to b only, and not proportionally to m as in (5.24). This means that, for instance if we rely on the Bernstein theorem, in order to obtain a similar concentration result compared to the independent setting[43], one needs to bound $D_W = \sup_{\mathbf{B}} W(\mathbf{B}, \mu) - \inf_{\mathbf{B}} W(\mathbf{B}, \mu)$ and $\mathrm{Var}[W(\mathbf{B}, \mu)]$ by a constant C_W which must satisfy $C_W \leq \frac{1}{d}C_Y$. Although obtaining a factor $1/d$ on D_W compared to D_Y seems very unlikely, we expect that the structural properties of the random blocks can at least be leveraged to derive a tighter bound on the variance, which would at least provide finer concentration results in the low-deviation regime (i.e. when t is small in (5.24)).

Yet, deriving a precise upper-bound on the variance of $W(\mathbf{B}, \mu)$ is not straightforward due to the expression of $W(\mathbf{B}, \mu)$, and our tentatives in this direction remain unsuccessful so far.

It should be noted that the bounds derived in the independent setting can still be reused[44] albeit being pessimistic in our case, which

[41] Cf. Appendix A.2.

[42] Cf. Section 2.3.3.

[43] Which can be expected given empirical observations.

[44] Indeed $D_W \leq D_Y$, and the bounds proposed on $\mathrm{Var}(Y(\boldsymbol{\omega}, \mu))$ in [5, Lemma 5.5] can also easily be checked to be valid upper-bounds for $\mathrm{Var}(W(\mathbf{B}, \mu))$.

is better than no result at all. However, the control of the excess risk will only hold with an additional factor d on the sketch size, i.e. with high probability provided that $m \gtrsim k^2 d^2$ (up to log factors) for clustering, while only $m \gtrsim k^2 d$ is proved to be sufficient with independent frequencies [5, Lemma 5.7] (and kd seems sufficient empirically).

5.5 Perspectives

In this chapter, we gave an overview of the diversity of structured randomized transforms existing in the literature, and proposed one way to adapt these ideas to our context. Through extensive experiments for the clustering task, we showed that using fast transforms indeed reduces greatly both sketching and learning times when the dimension is large enough, without degrading the clustering quality. This opens the way for new kinds of applications on high-dimensional data. This could also reveal to be very useful if the dimension is increased on purpose: we considered here the standard k-means task, but could simply perform kernel k-means[45] by converting each data point to an intermediate feature space when sketching, in which case the input dimension (for clustering) would be artificially increased. The biggest algorithmic constraint for larger experiments is maybe now the complexity with respect to k, as CL-OMPR scales in $\Theta(k^3)$.

[45] i.e. k-means using a custom kernel to measure similarity between data points in place of the euclidean inner-product.

Naturally, deriving statistical learning guarantees for these structured features is a challenge as discussed at the end of the chapter. We expect that strong concentration properties can be obtained for both $\mathbf{M}_{GR^2}$ and $\mathbf{M}_{\text{ff}}$ blocks, which would lead to a control of the excess risk with a smaller sketch size. Obtaining guarantees for the block $\mathbf{M}_{R^3}$ is certainly more complicated, and our tentatives in this direction have been unsuccessful so far; we leave this extension for future work.

To conclude, we also note that the usage of optical processing units has been considered in the literature [236, 79], and could be used as an alternative to structured matrices for high-dimensional applications. This technology seems promising as it could allow to perform multiplications by random matrices in constant time.

Part III

Compressive Clustering with Message Passing

Chapter 6

Compressive Clustering with Approximate Message Passing

Note The contributions presented in this chapter, which have been published in *IEEE Transactions on Signal Processing* [26], are a joint work with Evan Byrne and Philip Schniter, following a 3-month mobility performed during the PhD at the Ohio state university. During this collaboration, we improved on multiple points their initial work published at the Asilomar conference [237], which already contained some of the core ideas that will be exposed in Sections 6.2 and 6.3.

Contents

We introduced in Chapter 2 the compressive learning framework, and exposed how it has been used for clustering [2]. This has been done using the random Fourier feature map given in (2.5), and using CL-OMPR for reconstruction (cf. Algorithm 2.1). In this chapter, we introduce an alternative algorithm to recover the cluster centers from the sketch, named CL-AMP (compressive learning via approximate message passing), which is in most settings faster – it especially scales with the number of clusters k in $\mathcal{O}(k^2)$ rather than $\mathcal{O}(k^3)$ – and more accurate, especially for smaller sketch sizes.

An introduction to the family of message passing algorithms is provided in Section 6.1, and the problem of recovering the cluster centers from the sketch is translated into this formalism in Section 6.2. Section 6.3 details how the existing SHyGAMP algorithm can be adapted to our setting, and an experimental validation is proposed in Section 6.4.

6.1 An introduction to Approximate Message Passing

Message passing algorithms, also known as belief propagation algorithms, form a family of methods for inference in probabilistic models. We propose in this section to expose the key ideas of these techniques and related tools (Section 6.1.1), and explain how to get from the standard sum-product algorithm (Section 6.1.2) to the SHyGAMP algorithm (Section 6.1.3) that will be used in Section 6.3. We only provide a brief introduction to message passing techniques, and refer the reader

for instance to the book *Information, Physics, and Computation* [238] of Montanari and Mézard for more details.

6.1.1 Probabilistic model and factor graph for linear regression

In order to understand how such techniques are relevant for our problem, we will consider again the underdetermined linear inverse problem introduced in Section 2.1, i.e. recovering a signal $\mathbf{x}$, that we will assume in $\mathbb{R}^d$ for simplicity[1], from noisy linear measurements

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}. \tag{6.1}$$

[1] The algorithms can be adapted to the complex case, however this is not relevant in this chapter.

where $\mathbf{A}$ is known and $\mathbf{e}$ is noise. We mentioned that this problem can be addressed when $\mathbf{x}$ is assumed to be sparse via basis pursuit denoising, i.e. via the optimization problem[2]

$$\min_{\mathbf{x}\in\mathbb{C}^d} \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 + \lambda R(\mathbf{x}) \quad \text{with} \quad R(\mathbf{x}) = \|\mathbf{x}\|_1, \tag{6.2}$$

[2] For simplicity, we use here the same notation $\mathbf{x}$ for the true signal and the optimization variable.

where $\lambda > 0$ controls the tradeoff between data-fidelity and regularization. In the case of compressive sensing, the regularizer is simply $R = \|\cdot\|_1$ and promotes sparsity as $\mathbf{x}$ is assumed to be sparse, but other regularizers might be used for other applications.

Bayesian formulation To understand belief propagation, we need to reformulate this in a bayesian setting. Using the squared l_2-norm in order to measure the difference between observations $\mathbf{y}$ and $\mathbf{A}\mathbf{x}$ can be implicitly interpreted as modeling the noise as a Gaussian random variable. In this case, if we define $z \triangleq \mathbf{A}\mathbf{x}$, the probability of observing $\mathbf{y}$ knowing $\mathbf{z}$ is $p(\mathbf{y}|\mathbf{z}) \propto \exp(-\frac{1}{2}\|\mathbf{y}-\mathbf{z}\|_2^2)$, where the variance is arbitrary chosen to 1 – controlling this variance is equivalent to rescaling λ.

Note that we will in this chapter seldom give explicit normalization constants: unless otherwise specified, we assume normalization factors can either be computed by integration, or simply need not be known. Similarly, the regularizer R can be interpreted as a prior density $p(\mathbf{x}) \propto \exp(-\lambda R(\mathbf{x}))$ on the variable $\mathbf{x}$, so that Equation (6.2) becomes equivalent to maximum a posteriori (MAP) estimation of $\mathbf{x}$ under the posterior density $p(\mathbf{x}|y) \propto p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$:

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{C}^d} \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 + \lambda R(\mathbf{x}) &= \min_{\mathbf{x}\in\mathbb{C}^d} -(\log p(\mathbf{x}) + \log p(\mathbf{y}|\mathbf{x})) \\ &= \max_{\mathbf{x}\in\mathbb{C}^d} \log p(\mathbf{x}|\mathbf{y}).\end{aligned}$$

Instead of computing the MAP estimate, i.e. looking for the principal mode of the posterior density, we could rather compute $\mathbf{E}[\mathbf{x}|\mathbf{y}]$ (where the expectation is taken with respect to $p(\mathbf{x}|\mathbf{y})$), which is known as the minimum mean square error (MMSE) estimator[3]. MAP and MMSE estimation are the two main inference tasks for which message passing algorithms have been designed and can be used – although they might, by doing so, provide additional information such as estimates of the marginal distributions.

[3] Indeed, denoting $\mathbf{E}[\cdot|y]$ the conditional expectation, the estimator $\hat{\mathbf{x}}$ of $\mathbf{x}$ with minimum mean squared error $E[\|\hat{\mathbf{x}}-\mathbf{x}\|_2^2|y]$ satisfies

$$\frac{d}{d\hat{\mathbf{x}}}\mathbf{E}[\|\hat{\mathbf{x}}-\mathbf{x}\|_2^2|y] = 0,$$
$$\text{hence} \int_{\mathbf{x}} p(\mathbf{x}|y)[\hat{\mathbf{x}}-\mathbf{x}] = 0$$
$$\text{and } \hat{\mathbf{x}} = \mathbf{E}[\mathbf{x}|y].$$

Factor graphs We used previously separable likelihood and priors, in the sense that they can both be factorized as products of scalar functions. This derives from the fact that both $\|\cdot\|_2^2$ and $\|\cdot\|_1$ are (additively) separable functions. If we now generalize our model by using abstract likelihood and prior functions, while still requiring that they are both separable, i.e. that they can be written $p(\mathbf{y}|\mathbf{z}) = \prod_{j=1}^m p(y_j|z_j)$ and $p(\mathbf{x}) = \prod_{i=1}^d p(x_i)$, we end up with the generic probabilistic model

$$p(\mathbf{x}|\mathbf{y}) \propto \prod_{j=1}^{m} p(y_j|z_j) \prod_{i=1}^{d} p(x_i). \tag{6.3}$$

We represent this model in Figure 6.1 using a factor graph, i.e. a bipartite graph where all square nodes represent the factors of the model (6.3), circular nodes represent variables, and edges correspond to statistical dependence relations.

Message passing algorithms proceed by sending beliefs (the messages) along the edges of the factor graph, i.e. between variables and factors. We will explain the ideas behind the sum-product algorithm (SPA), which is the standard algorithm for MMSE estimation, and describe how it has been generalized via multiple successive improvements. A very similar algorithm can be derived for MAP estimation (the max-sum algorithm, or MSA), but we omit it here, as our application will be recasted into an MMSE inference task.

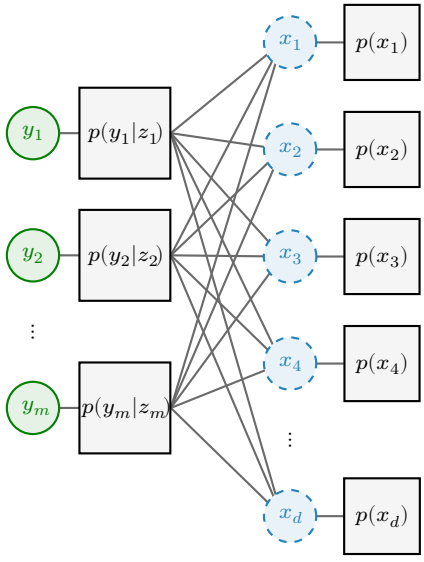


Figure 6.1: Factor graph of the model defined at Equation (6.3). Square nodes correspond to the factors. Plain green circles correspond to observed variables, and the dashed blue ones to the variables to infer.

6.1.2 The sum-product algorithm

The sum-product algorithm was introduced by Pearl [239]. After initialization, the different nodes (variables and factors) communicate via messages that are all sent simultaneously, i.e. at time t each node sends messages to its neighbors using the information received at time $t-1$. These messages, which are sent along the edges of the factor graph, take the form of probability densities and should be interpreted as beliefs about the marginal distributions of the variables $(x_i)_{1\leq i\leq d}$, and we will as a consequence denote them as functions of these variables. Each factor $p(y_j|z_j)$ communicates to each variable node x_i a belief[4] denoted $M_{j\to i}$, which is computed using the information (beliefs) coming from all the other[5] nodes $(x_l)_{l\in[\![1,d]\!]\setminus i}$ and the value y_j (deterministic scalar), and reversely each node x_i sends to each factor $p(y_j|z_j)$ the (log-)marginal distribution $M_{j\leftarrow i}$ that it estimates using its own prior and the messages coming from all other factors $(p(y_l|z_l))_{l\in[\![1,m]\!]\setminus j}$. These messages are defined (up to normalization) as

$$\exp(M_{j\to i}(x_i)) \propto \int_{(x_l)_{l\in[\![1,d]\!]\setminus i}} p(y_j|z_j = \mathbf{a}_j^T\mathbf{x}) \prod_{l\neq i} M_{j\leftarrow l}(x_l) \tag{6.4}$$

$$\exp(M_{j\leftarrow i}(x_i)) \propto p(x_i) \exp\left(\sum_{l\neq j} M_{l\to i}(x_i)\right), \tag{6.5}$$

where $\mathbf{a}_j^T$ is the j-th row of $\mathbf{A}$ and where the messages on the right-hand side are the ones received at the previous iteration (we avoid parametrization with a time variable for simplicity). It should be noticed that no messages will be sent from the variables $(y_j)_{1\leq j\leq m}$

[4] Hence, depending on the context, x_i can refer both to the node corresponding to the variable x_i, or to a free variable when it appears in a probability density.

[5] In the following, we use the backslash to denote exclusion of one element, e.g. $[\![1,d]\!]\setminus i$ is used as a shorter notation for $[\![1,d]\!]\setminus\{i\} = \{1,...,i-1,i+1,...,d\}$.

(which are observed, and only have one neighbor), and messages sent from the prior factors will be constant over time as, here again, each prior factor has only one neighbor in the graph. Although the sums in eqs. (6.4) and (6.5) always exclude one term[6], it is possible at any time to estimate the marginal probability of x_i using a variant of (6.5) where all the messages are used:

$$p(x_i|\mathbf{y}) \propto p(x_i) \exp\left(\sum_{j=1}^{m} M_{j\to i}(x_i)\right). \tag{6.6}$$

[6] So that information is not "sent back" to where it comes from.

The algorithm, although presented here in the specific case of the model (6.3), can naturally be used in any factor graph. It has initially been introduced for inference in factor graphs that contain no loops (i.e. trees), and in this case it estimates exact posterior marginals after two iterations – assuming one starts propagating the messages from the leaves of the tree. When the factor graph does contain loops as in our example, the algorithm can still be used, but convergence is not granted, and convergence to poor estimates of the marginals is possible [240]. This setting is referred to as loopy belief propagation.

6.1.3 Approximate message passing

The sum-product algorithm, when applied to the linear model described above, requires sending $2dm$ messages at each iteration, and thus becomes quickly too expensive in high dimension. Under some specific assumption on the matrix $\mathbf{A}$, and in the large-scale limit when $d, m \to \infty$ at constant ratio, it is possible to use the central limit theorem and Taylor developments to approximate the expressions of the message of the sum-product algorithm [241, 242]. We omit these derivations here for conciseness, but notice that they allow to approximate the messages by Gaussian densities, so that only mean and variances need to be computed. Moreover, by exploiting similarities between messages, the total number of messages exchanged at each iteration can be reduced to $m + d$ instead of $2md$. This idea was first proposed in the particular case of a separable likelihood $p(\mathbf{y}|\mathbf{z}) = \prod_{j=1}^{m} p(y_j|z_j)$ where each $p(y_j|z_j)$ is Gaussian [241], and we refer to the resulting algorithm as approximate message passing (AMP). This setting was generalized by Rangan [242] to the case where only separability of the likelihood and prior are assumed, which is much more generic and covers many different models. The resulting algorithms[7] are refered as GAMP (Generalized AMP). We consider this setting from now on.

[7] There are at least two versions of the algorithm for MAP and MMSE estimation, and other small variants have been proposed, thus GAMP should rather be considered as a family of algorithms.

An intuitive way to understand the GAMP algorithm is that it computes at each iteration and for every $j \in [\![1, m]\!]$ a pseudo-prior $\mathcal{N}(\hat{p}_j, q_j^p)$ of z_j, which can be combined with the likelihood $p(y_j|z_j)$ to compute posterior estimates of the mean and covariance of z_j. Similarly, it provides at each iteration and for each $i \in [\![1, d]\!]$ a pseudo-measurement r_i which follows the model $r_i = x_i + v_i$ where $v_i \sim \mathcal{N}(0, q_i^r)$ for some q_i^r, and which can be combined with the prior on x_i to produce posterior estimates of the mean and covariance of x_i. Detailed steps are provided in Algorithm 6.1, where the expectations and variances at lines 8, 9, 15

and 16 are computed with respect to the following densities (defined up to normalization)

$$p(x_i \,|\, \hat{r}_i; q^r) \propto p(x_i)\mathcal{N}(x_i; \hat{r}_i, q_i^r) \tag{6.7}$$
$$p(z_j \,|\, y_j, \hat{p}_j; q^p) \propto p(y_j|z_j)\mathcal{N}(z_j; \hat{p}_j, q_j^p). \tag{6.8}$$

Note that in all the chapter, quantities with hats denote estimators while variables starting with the letter "q" denote variances (and later covariance matrices/vectors).

Input: Prior $p(\mathbf{x})$ and likelihood $p(\mathbf{y}|\mathbf{z})$, which are used to define the densities $p(x_i \,|\, \hat{r}_i; q^r)$ and $p(z_j \,|\, y_j, \hat{p}_j; q^p)$ in eqs. (6.7) and (6.8).

1 $\hat{\mathbf{x}} \leftarrow \int \mathbf{x} p(\mathbf{x}) d\mathbf{x}$ `// Use prior for initialization`
2 $q_i^x \leftarrow \int |x_i - \hat{x}_i|^2 p(x_i) dx_i$ **for each** $i \in [\![1, d]\!]$
3 $\hat{\mathbf{s}} \leftarrow 0$
4 **repeat**
  `// Output nodes`
5 $q^p \leftarrow \frac{1}{m}\|\mathbf{A}\|_F^2 (\frac{1}{d}\sum_{i=1}^d q_i^x)$
6 $\hat{\mathbf{p}} \leftarrow \mathbf{A}\hat{\mathbf{x}} - \hat{\mathbf{s}}\, q^p$ `// size` m
7 **for** $j \in [\![1, m]\!]$ **do**
8 $q_j^z \leftarrow \mathrm{Var}(z_j|y_j, \hat{p}_j; q^p)$ `// Using (6.8)`
9 $\hat{z}_j \leftarrow \mathbf{E}[z_j|y_j, \hat{p}_j; q^p]$ `// Using (6.8)`
10 $q^s \leftarrow (1 - (\frac{1}{m}\sum_{j=1}^m q_j^z)/q^p)/q^p$
11 $\hat{\mathbf{s}} \leftarrow (\hat{\mathbf{z}} - \hat{\mathbf{p}})/q^p$ `// size` m
  `// Input nodes`
12 $q^r \leftarrow (\frac{1}{d}\|\mathbf{A}\|_F^2 q^s)^{-1}$
13 $\hat{r} \leftarrow \hat{x} + q^r \mathbf{A}^T \hat{s}$ `// size` d
14 **for** $i \in [\![1, d]\!]$ **do**
15 $q_i^x \leftarrow \mathrm{Var}(x_i|\hat{r}_i; q^r)$ `// Using (6.7)`
16 $\hat{x}_i \leftarrow \mathbf{E}[x_i|\hat{r}_i; q^r]$ `// Using (6.7)`
17 **until** *convergence*
18 **return** $\hat{\mathbf{x}}$

Algorithm 6.1: MMSE-GAMP, scalar version.

Note that multiple variants of this algorithm exist – different approximations can be combined, providing a tradeoff between accuracy and efficiency. The version given here is the "scalar" version of the algorithm [242, Algorithm 2, MMSE setting], which uses scalar variances q^p, q^s, q^r, whereas the standard version of the algorithm would use separate variances $(q_j^p)_{1\leq j\leq m}, (q_j^s)_{1\leq j\leq m}, (q_i^r)_{1\leq i\leq d}$. This modification can be justified when the entries of $\mathbf{A}$ all have roughly the same magnitude, or alternatively by the law of large numbers when $\mathbf{A}$ has i.i.d. components and the dimensions d and m are large. With this simplification, the dependence on $\mathbf{A}$ appears only[8] at lines 6 and 13 as multiplications by $\mathbf{A}$ and $\mathbf{A}^T$, whereas two more matrix multiplications would otherwise appear[9].

AMP and GAMP methods have been used for various applications, including the LASSO problem (cf. Chapter 2) where they usually perform better than other classical methods such as FISTA [243] provided

[8] Which is why it is sometimes referred to as a "first-order" algorithm.

[9] These two multiplications do not actually directly involve $\mathbf{A}$, but rather its elementwise squared version (and its transpose). Apart from the cost of the multiplication, this can also make the use of fast transforms problematic if this operation cannot be performed efficiently.

that $\mathbf{A}$ is well conditioned – and in particular when $\mathbf{A}$ has i.i.d. normal entries. When this is not the case, damping is often used [244, 245]: at each iteration, each variable is updated to a weighted combination of the new value computed according to Algorithm 6.1, and the value obtained at previous iteration. We use this terminology in the following, but note that many "inertial" methods in the optimization literature rely on similar ideas.

Analysis Both AMP and GAMP approximate well the sum-product algorithm when $\mathbf{A}$ is large and has i.i.d. sub-Gaussian entries, and can be analyzed using the state-evolution framework [246, 247] which proves that they converge to MMSE estimates in certain regimes. The fixed points of GAMP can also be shown to correspond in some settings to critical points of a constrained optimization problem on the posterior density (for MAP estimation) or a closely related free energy function (MMSE estimation) [248].

Complexity The cost of the linear operations performed by GAMP when using a dense matrix $\mathbf{A}$ is $\Theta(md)$ per iteration – we recall that the matrix $\mathbf{A}$ has size $m \times d$ –, and is dominated by the two matrix products involving $\mathbf{A}$ and its transpose. The complexity of lines 8, 9, 15 and 16 depends on the chosen prior and likelihood, however it is typically smaller than the cost of linear steps. As mentioned earlier, using the scalar version of the algorithm reduces the number of matrix multiplications, and does reduce a bit the total amount of memory to allocate, but it does not change the overall complexity. Structured matrices (e.g. based on the fast Fourier transform) can be used to reduce the computational cost and have empirically shown to perform well[10].

[10] We are not aware of papers in the literature using constructions similar to the ones used in Chapter 5 for message passing, but our experiments suggest that such matrices perform almost as well as Gaussian i.i.d. ones, at least for this compressive clustering application.

Further extensions We presented in this section the basic ideas of approximate message passing methods. Multiple further extensions exist. In particular, for compressive clustering we will use the SHyGAMP (Simplified Hybrid GAMP) algorithm. We discuss below how this algorithm differs from Algorithm 6.1, and how it can be used in our setting. We will also see that prior and likelihood functions sometimes depend on hyperparameters, that can be tuned simultaneously using the expectation-maximization (EM) algorithm as initially proposed by Vila and Schniter [249].

6.2 Compressive clustering as a high-dimensional inference problem

We introduced in the previous section the family of message passing algorithms, and focused on the case where measurements $\mathbf{y}$ are considered to depend – via an abstract separable likelihood function – on the linear measurements $\mathbf{z} = \mathbf{A}\mathbf{x}$, which is known as a generalized linear model. In this section, we detail how the task of learning the

cluster centers from the sketch can be modeled in a similar way: we will see that the sketch can be seen as a collection of generalized linear measurements of the centroids via the matrix of frequency vectors. We first express the expected sketch under a Gaussian mixture data model in Section 6.2.1, and use this model to formulate the learning problem as a Bayesian inference task in Section 6.2.2.

6.2.1 Model of the sketch

The CL-AMP algorithm relies on the modeling assumption that the data is drawn i.i.d. according to the Gaussian mixture model

$$\pi \triangleq \sum_{i=1}^{k} \alpha_i \mathcal{N}(\mathbf{c}_i, \mathbf{\Sigma}_i) \quad \text{with} \quad \sum_{i=1}^{k} \alpha_i = 1, \tag{6.9}$$

where $\mathbf{c}_i$ are the centroids to recover and $\mathbf{\Sigma}_i$ are unknown covariances – which will not be estimated directly, as discussed below. We denote $\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_k] \in \mathbb{R}^{d \times k}$ the true matrix of centroids.

We recall that for clustering, the sketch is computed using random Fourier features according to Definition 2.5 (i.e. $\Phi = \Phi^{\text{RFF}}$), so that the sketch of the centroids $\mathbf{C}$ with weights $(\alpha_i)_{1 \leq i \leq k}$ (that we define as the sketch of $\pi_{\mathbf{C}} \triangleq \sum_{i=1}^{k} \alpha_i \delta_{\mathbf{c}_i}$) is

$$\mathbf{s}_{\mathbf{C}} = \sum_{i=1}^{k} \alpha_i \exp(\imath \mathbf{\Omega}^T \mathbf{c}_i) \tag{6.10}$$

where the exponential is applied pointwise, and $\mathbf{\Omega} = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m]$ denotes the frequency vectors. The sketch is not scaled by $1/\sqrt{m}$. We define for convenience and for each $i \in [\![1, m]\!]$ the quantities

$$g_i \triangleq \|\boldsymbol{\omega}_i\|_2, \qquad \mathbf{a}_i = \boldsymbol{\omega}_i / g_i, \quad \text{and } \mathbf{z}_i \triangleq \mathbf{C}^T \mathbf{a}_i. \tag{6.11}$$

For a fixed $\boldsymbol{\omega}$ and $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, the quantity $\boldsymbol{\omega}^T \mathbf{x}$ follows[11] the distribution $\mathcal{N}(\boldsymbol{\omega}^T \boldsymbol{\mu}, \boldsymbol{\omega}^T \mathbf{\Sigma} \boldsymbol{\omega})$. As a consequence, for any $j \in [\![1, m]\!]$ and $\mathbf{x} \sim \pi$ we have $\boldsymbol{\omega}_j^T \mathbf{x} \sim \Sigma_{i=1}^{k} \alpha_i \mathcal{N}(\boldsymbol{\omega}_j^T \mathbf{c}_i, \boldsymbol{\omega}_j^T \mathbf{\Sigma}_i \boldsymbol{\omega}_j)$. Thus, using the fact that $\mathbf{E}_{x \sim \mathcal{N}(\mu, \sigma^2)} \exp(\imath x) = \exp(\imath \mu - \frac{1}{2}\sigma^2)$, the components of the sketch of the Gaussian mixture[12] π can be computed for each $i \in [\![1, m]\!]$ as

$$\begin{aligned} s_j &\triangleq \mathbf{E}_{\mathbf{x} \sim \pi}\left[\exp(\imath \boldsymbol{\omega}_j^T \mathbf{x})\right] \\ &= \sum_{i=1}^{k} \alpha_i \exp(\imath \boldsymbol{\omega}_j^T \mathbf{c}_i - \tfrac{1}{2} \boldsymbol{\omega}_j^T \mathbf{\Sigma}_i \boldsymbol{\omega}_j) \\ &= \sum_{i=1}^{k} \alpha_i \exp(\imath g_j \underbrace{\mathbf{a}_j^T \mathbf{c}_i}_{\triangleq z_{ji}} - \tfrac{1}{2} g_j^2 \underbrace{\mathbf{a}_j^T \mathbf{\Sigma}_i \mathbf{a}_j}_{\triangleq \tau_{ji}}) \end{aligned}$$

The vectors $(\mathbf{a}_j)_{1 \leq j \leq m}$ being drawn (uniformly) on the unit sphere when drawing the $(\boldsymbol{\omega}_j)_{1 \leq j \leq m}$ according to a multivariate normal distribution as previously assumed, for any $i \in [\![1, k]\!]$ the quantities $(\tau_{ji})_{1 \leq j \leq m}$ are with high probability well concentrated around their expectation [250, Theorem 2.1], that we denote τ_i and which depends on $\text{tr}(\mathbf{\Sigma}_i)$. Concentration would actually also hold using e.g. structured orthogonal frequencies[13].

[11] As an affine transformation of a multivariate normal random variable.

[12] We denote $\mathbf{s}$ the "true" sketch, i.e. computed in expectation over the considered data distribution, and $\tilde{\mathbf{s}}$ the expected sketch measured on n data samples.

[13] This result can be seen as a Hanson-Wright type inequality (a concentration inequality for a quadratic form in a sub-Gaussian random variable), however the general formulation [127, Theorem 6.2.1] requires the considered random vector X to have independent coordinates, which holds for $X = \boldsymbol{\omega}$ but not for $X = \mathbf{a}$ in our case. The result in [250] relies on the weaker assumption that $\langle \varphi, X \rangle$ is uniformly subgaussian for $\varphi \in S^{d-1}$, which holds because $\mathbf{a}$ is bounded. This formulation of the result uses (by opposition to the "standard" Hanson-Wright result) an additional hypothesis which is the positive-definiteness of $\mathbf{\Sigma}_i$, which holds in our setting.

As a consequence, when the number n of samples is large, we have

$$\tilde{\mathbf{s}}_j \approx \sum_{i=1}^{k} \alpha_i \exp(\imath g_j z_{ji} - \tfrac{1}{2} g_j^2 \tau_i), \tag{6.12}$$

which is quite convenient as the dependence to the covariance matrices $(\mathbf{\Sigma}_i)_{1\leq i\leq k}$ now only appears via the quantity $\boldsymbol{\tau} \triangleq [\tau_1, ..., \tau_k] \in \mathbb{R}^k$, which is considered as an hyperparameter, together with the vector of weights $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_k]$ (in the probability simplex).

6.2.2 Bayesian formalism

Now that we have an approximation of the sketch of the Gaussian mixture model (6.9), we recast the learning task into a bayesian inference problem. We denote $p(\mathbf{C})$ the chosen prior density on $\mathbf{C}$ (see below), and $p(\mathbf{s}|\mathbf{C})$ the likelihood function of $\mathbf{C}$ for an ideal sketch $\mathbf{s} \in \mathbb{C}^m$, so that the posterior density $p(\mathbf{C}|\mathbf{s})$ of C satisfies

$$p(\mathbf{C}|\mathbf{s}) \propto p(\mathbf{s}|\mathbf{C}) p(\mathbf{C}). \tag{6.13}$$

We want to return a minimum mean square error estimate (MMSE) denoted $\hat{\mathbf{C}}$, i.e.

$$\hat{\mathbf{C}} \triangleq \mathbf{E}[\mathbf{C}|\mathbf{s}]. \tag{6.14}$$

where the expectation is taken with respect to the posterior density (6.13).

Likelihood Using the previous derivations and especially (6.12), the likelihood is only separable "by blocks" and takes the form

$$p(\mathbf{s}|\mathbf{C}) = \prod_{i=1}^{m} p_{s|\mathbf{z}}(s_j|\mathbf{z}_j), \text{ where} \tag{6.15}$$

$$p_{s|\mathbf{z}}(s_j|\mathbf{z}_j; \boldsymbol{\alpha}, \boldsymbol{\tau}) = \delta\left(s_j - \sum_{i=1}^{k} \alpha_i \exp(\imath g_j z_{ji} - \tfrac{1}{2} g_j^2 \tau_i) \right). \tag{6.16}$$

This can be interpreted as a generalized linear model [251].

Prior Due to the nature of the GAMP algorithm used below, and the way pseudo-measurements are produced at each iteration[14], we impose here a prior which is factorized on the rows of $\mathbf{C}$, that we denote $(\mathbf{c}^i)_{1\leq i\leq d}$ by opposition to the centers themselves, which are columns of $\mathbf{C}$ and denoted $(\mathbf{c}_i)_{1\leq i\leq d}$ above. We write it

$$p(\mathbf{C}) \triangleq \prod_{i=1}^{d} p_{\mathbf{c}}(\mathbf{c}^i). \tag{6.17}$$

We will now detail how belief propagation can be used for approximate inference of (6.14).

[14] Indeed, the scalar $(\hat{r}_i)_{1\leq i\leq d}$ in Algorithm 6.1, which become k-dimensional vectors $(\hat{\mathbf{r}}_i)_{1\leq i\leq d}$ below, should be interpreted as pseudo-measurements of the rows of $\mathbf{C}$, hence a prior factorized e.g. on the columns would not be suitable.

6.3 Inference of the centers using GAMP

Although we made clear how our problem could be casted into an inference task, Algorithm 6.1 cannot be used directly to solve it and some

details must be provided. We discuss in Section 6.3.1 these adaptations, and detail in Section 6.3.2 the different approximations performed to estimate posterior means and variances appearing in the algorithm. Tuning of the hyperparameters $\boldsymbol{\alpha}, \boldsymbol{\tau}$ is discussed in Section 6.3.3, and the whole algorithm (including initialization and tuning) is given Section 6.3.4.

6.3.1 From GAMP to SHyGAMP

Firstly, in order to be coherent with previous notations, we will use as a linear mixing matrix $\mathbf{A} = [\mathbf{a}_1, ..., \mathbf{a}_m]^T$, whose each row $\mathbf{a}_j^T$ corresponds to an l_2-normalized column of $\boldsymbol{\Omega}$. Then, the observations we have access to are the entries of the empirical sketch $\tilde{\mathbf{s}} \in \mathbb{C}^m$, thus $\tilde{\mathbf{s}}$ takes the role of the observation vector $\mathbf{y}$ used earlier. Similarly, the signal to recover is now the $d \times k$ matrix $\mathbf{C}$, which takes the role of the d-dimensional vector $\mathbf{x}$ above[15].

As a consequence, the biggest difference with the setting presented in Section 6.1 is that the $(\mathbf{z}_j)_{1 \leq j \leq m}$ are k-dimensional vectors, by opposition to scalar quantities $(z_j)_{1 \leq j \leq m}$ above. Thus our likelihood, although it can be factorized by blocks, is not separable as assumed by GAMP, i.e. $p(\mathbf{s}|\mathbf{C}) = \prod_{j=1}^m p_{s|\mathbf{z}}(s_j|\mathbf{z}_j)$ but cannot be written $\prod_{j=1}^m \prod_{l=1}^k p(s_j|(\mathbf{z}_j)_l)$. The Hybrid GAMP algorithm [252] was developed to handle such structured models and can be applied here; its formulation would be highly similar to Algorithm 6.1, but with k-dimensional variables and observations $(\mathbf{z}_j)_{1 \leq j \leq m}$. However, this implies that the variances which were scalars before are now replaced by covariance matrices. From a computational perspective, HyGAMP would require inverting $m + d$ different k-dimensional covariance matrices at each iteration. We will hence rather use the simplified HyGAMP (SHyGAMP) algorithm [253], which diagonalizes all the covariance matrices, and thus further reduces the complexity of the method.

Algorithm 6.2 details this approach for our setting. We denote, by analogy with Algorithm 6.1, $\hat{\mathbf{R}} = [\hat{\mathbf{r}}_1, ..., \hat{\mathbf{r}}_d]^T$ (pseudo-measurements), $\hat{\mathbf{P}} = [\hat{\mathbf{p}}_1, ..., \hat{\mathbf{p}}_m]^T$ (pseudo-priors) and $\hat{\mathbf{Z}} = [\hat{\mathbf{z}}_1, ..., \hat{\mathbf{z}}_m]^T$. We also denote diag the operator which extracts the diagonal of a matrix (as a vector), and Diag the operator which does the opposite, i.e. creates a squared diagonal matrix with the given diagonal.

Note that our normalization implies $\|\mathbf{A}\|_F = d$ compared to Algorithm 6.1. When running the algorithm, the quantities $\hat{\mathbf{C}}$ and $\hat{\mathbf{Z}}$ will ideally converge towards approximations of the MMSE estimates $\mathbf{E}[\mathbf{C}|\tilde{\mathbf{s}}]$ and $\mathbf{E}[\mathbf{Z}|\tilde{\mathbf{s}}]$. Expectations and covariances of the $\hat{\mathbf{z}}_j$ and $\hat{\mathbf{c}}_i$ at lines 9, 10, 16 and 17 are computed with respect to the densities (up to normalization)

$$p_{\mathbf{z}|s,\mathbf{p}}(\mathbf{z}_j \,|\, s_j, \hat{\mathbf{p}}_j; \mathbf{q}^{\mathbf{p}}, \boldsymbol{\alpha}, \boldsymbol{\tau}) \propto p_{s|\mathbf{z}}(s_j|\mathbf{z}_j; \boldsymbol{\alpha}, \boldsymbol{\tau}) \mathcal{N}(\mathbf{z}_j; \hat{\mathbf{p}}_j, \mathbf{Q}^{\mathbf{p}}), \tag{6.18}$$

$$p_{\mathbf{c}|\mathbf{r}}(\mathbf{c}^i \,|\, \hat{\mathbf{r}}_i; \mathbf{q}^{\mathbf{r}}) \propto p_{\mathbf{c}}(\mathbf{c}^i) \mathcal{N}(\mathbf{c}^i; \hat{r}_i, \mathrm{Diag}(\mathbf{q}^{\mathbf{r}})). \tag{6.19}$$

where $\mathbf{Q}^{\mathbf{p}} \triangleq \mathrm{Diag}(\mathbf{q}^{\mathbf{p}})$. We will discuss in Section 6.3.3 how to tune the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ by alternating with EM iterations. For now, we assume they are fixed[16] and focus on the algorithm itself.

[15] We used the notation $\mathbf{x}$ for the signal in Section 6.1 to be coherent with previous discussions on the linear model and the literature, and to make clear that $\mathbf{x}$ was a vector while we are now dealing with a matrix.

[16] and thus write $p(s_j|\mathbf{z}_j)$ rather than $p(s_j|\mathbf{z}_j; \boldsymbol{\alpha}, \boldsymbol{\tau})$ in the next section

1 **function** SHyGAMP ($\tilde{\mathbf{s}}$, $\hat{\mathbf{C}}_\mathbf{0}$, $\mathbf{q}_0^\mathbf{p}$, $\boldsymbol{\alpha}$, $\boldsymbol{\tau}$)
2 $\mathbf{q}_i^\mathbf{c} \leftarrow \mathbf{q}_0^\mathbf{p}$ **for each** $i \in [\![1, d]\!]$ // For the init. of $\mathbf{q}^\mathbf{p}$
3 $\hat{\mathbf{S}} \leftarrow 0$
4 $\hat{\mathbf{C}} \leftarrow \hat{\mathbf{C}}_\mathbf{0}$ // See Section 6.3.4 for initialization
5 **repeat**
  // Output nodes
6 $\mathbf{q}^\mathbf{p} \leftarrow \frac{1}{d}\sum_{i=1}^{d} \mathbf{q}_i^\mathbf{c}$ // size k
7 $\hat{\mathbf{P}} \leftarrow \mathbf{A}\hat{\mathbf{C}} - \hat{\mathbf{S}}\,\mathrm{Diag}(\mathbf{q}^\mathbf{p})$ // size $m \times k$
8 **for** $j \in [\![1, m]\!]$ **do**
9 $\mathbf{q}_j^\mathbf{z} \leftarrow \mathrm{diag}(\mathrm{Cov}(\mathbf{z}_j | \tilde{s}_j, \hat{\mathbf{p}}_j; \mathrm{Diag}(\mathbf{q}^\mathbf{p}), \boldsymbol{\alpha}, \boldsymbol{\tau}))$ // size k
10 $\hat{\mathbf{z}}_j \leftarrow \mathbf{E}[\mathbf{z}_j | \tilde{s}_j, \hat{\mathbf{p}}_j; \mathrm{Diag}(\mathbf{q}^\mathbf{p}), \boldsymbol{\alpha}, \boldsymbol{\tau}]$ // size k
11 $\mathbf{q}^\mathbf{s} \leftarrow \mathbf{1} \oslash \mathbf{q}^\mathbf{p} - (\frac{1}{m}\sum_{i=1}^{m} \mathbf{q}_i^\mathbf{z}) \oslash (\mathbf{q}^\mathbf{p} \otimes \mathbf{q}^\mathbf{p})$ // size k
12 $\hat{\mathbf{S}} \leftarrow (\hat{\mathbf{Z}} - \hat{\mathbf{P}})\,\mathrm{Diag}(\mathbf{q}^\mathbf{p})^{-1}$ // size $m \times k$
  // Input nodes
13 $\mathbf{q}^\mathbf{r} \leftarrow \frac{d}{m}\mathbf{1} \oslash \mathbf{q}^\mathbf{s}$ // size k
14 $\hat{\mathbf{R}} \leftarrow \hat{\mathbf{C}} + \mathbf{A}^T\hat{\mathbf{S}}\,\mathrm{Diag}(\mathbf{q}^\mathbf{r})$ // size $d \times k$
15 **for** $i \in [\![1, d]\!]$ **do**
16 $\mathbf{q}_i^\mathbf{c} \leftarrow \mathrm{diag}(\mathrm{Cov}(\mathbf{c}_i | \hat{\mathbf{r}}_i; \mathrm{Diag}(\mathbf{q}^\mathbf{r})))$ // size k
17 $\hat{\mathbf{c}}_i \leftarrow \mathbf{E}[\mathbf{c}_i | \hat{\mathbf{r}}_i; \mathrm{Diag}(\mathbf{q}^\mathbf{r})]$ // size k
18 **until** *convergence*
19 **return** $(\hat{\mathbf{C}}, \hat{\mathbf{Z}}, (\mathbf{q}_j^\mathbf{z})_{1 \le j \le m})$

Algorithm 6.2: MMSE-SHyGAMP for compressive clustering. Here lines 9 and 10 are computed using the posterior density (6.18), and lines 16 and 17 using (6.19). Quantities $\mathbf{q}^\mathbf{p}, \mathbf{q}^\mathbf{s}, \mathbf{q}^\mathbf{r}$ are vectors because of the structure of the factor graph, but do not depend on indexes $i \in [\![1, d]\!], j \in [\![1, m]\!]$ because we use the "scalar" version of the algorithm.

Complexity The overall algorithm has a computational complexity of $\Theta(mdkI)$ when using a dense matrix of frequencies, where T denotes the number of iterations. SHyGAMP reduces the initial inference problem of dimension kd to $(m + d)$ k-dimensional inference problems (lines 9, 10, 16 and 17). In comparison, CL-OMPR has complexity $\Theta(mdk^2)$. Although it might not be straightforward to compare both methods without knowing the number of iterations of CL-AMP (which will typically be larger than k), CL-OMPR is in practice much slower due to the numerous optimization problems which are solved at each step. See for instance lines 3 and 5 of Algorithm 2.1, which are both non-convex optimization problems and are typically solved using L-BFGS-B (limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm with box constraints).

Solving the intermediate inference problems in our application of SHyGAMP is however still not straightforward either, and we now focus on this issue.

6.3.2 Solving the remaining inference tasks

There are four remaining inference tasks which require attention in the SHyGAMP method described in Algorithm 6.2. Operations performed at lines 16 and 17 naturally depend on the prior used. Simple closed forms exist for instance when using a Gaussian prior. It turns out that the algorithm is still well defined when using a non-informative prior (i.e. $p_\mathbf{c}(\mathbf{c}^i) \propto 1$ for every $i \in [\![1, d]\!]$), and lines 16 and 17 simply

become in this case $\mathbf{q}_i^{\mathbf{c}} \leftarrow \mathbf{q}^{\mathbf{r}}$ and $\hat{\mathbf{c}}_i \leftarrow \hat{\mathbf{r}}_i$ for every $i \in [\![1, d]\!]$. We use this approach in our experiments below as this already works well in practice.

The difficulties come rather from the estimation of the posterior mean and covariance of $\mathbf{z}$, computed at lines 9 and 10 using the density (6.18). In the following, we denote $\mathbf{Q}^{\mathbf{p}} \triangleq \mathrm{Diag}(\mathbf{q}^{\mathbf{p}})$. We need to compute

$$\hat{z}_{jl} \triangleq \frac{1}{C_j} \int_{\mathbb{R}^k} z_{jl} p(\tilde{s}_j|\mathbf{z}_j) \mathcal{N}(\mathbf{z}_j; \hat{\mathbf{p}}_j, \mathbf{Q}^{\mathbf{p}}) \, \mathrm{d}\mathbf{z}_j \tag{6.20}$$

$$q_{jl}^{\mathbf{z}} \triangleq \frac{1}{C_j} \int_{\mathbb{R}^k} |z_{jl} - \hat{z}_{jl}|^2 p(\tilde{s}_j|\mathbf{z}_j) \mathcal{N}(\mathbf{z}_j; \hat{\mathbf{p}}_j, \mathbf{Q}^{\mathbf{p}}) \, \mathrm{d}\mathbf{z}_j \tag{6.21}$$

for each $j \in [\![1, m]\!]$ and $l \in [\![1, k]\!]$, where $\hat{z}_{jl}$, z_{jl} and $q_{jl}^{\mathbf{z}}$ are the l-th elements of respectively $\hat{\mathbf{z}}_j$, $\mathbf{z}_j$ and $\mathbf{q}_j^{\mathbf{z}}$, and C_j is the normalization constant of the posterior density, i.e.

$$C_j = \int_{\mathbb{R}^k} p(\tilde{s}_j|\mathbf{z}_j) \mathcal{N}(\mathbf{z}_j; \hat{\mathbf{p}}_j, \mathbf{Q}^{\mathbf{p}}) \, \mathrm{d}\mathbf{z}_j. \tag{6.22}$$

The nature of $p_{s|\mathbf{z}}$ (cf. (6.16)) makes it impossible to derive closed-form expressions for $\hat{z}_{ji}$ and $q_{ji}^{\mathbf{z}}$. Following the idea of approximate message passing, we will make some additional approximations. For a given $t \in [\![1, k]\!]$, we rewrite (6.12) as

$$\tilde{s}_j \approx \alpha_t \exp(\imath g_j z_{jt} - \tfrac{1}{2} g_j^2 \tau_t) + \sum_{1 \le l \ne t \le k} \alpha_l \exp(\imath g_j z_{jl} - \tfrac{1}{2} g_j^2 \tau_l). \tag{6.23}$$

In order to simplify the following derivations, we omit until the end of this section (and in the appendices) the index j in all variables. We rewrite (6.23) as

$$\tilde{s}_j \approx \beta_t \exp(\imath \theta_t) + \sum_{1 \le l \ne t \le k} \beta_l \exp(\imath g(z_l + n_l)), \tag{6.24}$$

$$\text{where } \beta_l \triangleq \alpha_l \exp(-\tfrac{1}{2} g^2 \tau_l) \; \forall l \in [\![1, k]\!], \tag{6.25}$$

$$\theta_t \triangleq g(z_t + n_t), \tag{6.26}$$

$$\text{and } n_l \sim \mathcal{N}(0, q^n). \tag{6.27}$$

We introduced in this expression the i.i.d. random variables n_l to simplify the derivations, but will eventually consider $q^n \to 0$ so that (6.24) matches (6.23). We now rewrite $\hat{z}_t$ and $q_t^{\mathbf{z}}$ (the two quantities to estimate) with respect to the posterior mean $\hat{\theta}_t$ and variance $\mathrm{Var}(\theta_t|s)$ of θ_t. Derivations can be found in Appendix B.1, and yield the following expressions

$$\hat{z}_t = \frac{\hat{p}_t}{1 + \frac{q_t^{\mathbf{p}}}{q^n}} + \frac{\hat{\theta}_t}{g(1 + \frac{q^n}{q_t^{\mathbf{p}}})}$$

$$q_t^{\mathbf{z}} = \frac{q^n}{\frac{q^n}{q_t^{\mathbf{p}}} + 1} + \frac{1}{g^2(1 + \frac{q^n}{q_t^{\mathbf{p}}})^2} \mathrm{Var}(\theta_t|s).$$

Both the marginal posterior mean and variance of θ_t are still difficult to compute due to the form of $p(\theta_t|s)$ (provided in appendix, see (B.5)). We thus approximate the second term (the sum) in (6.24) as Gaussian

to estimate these quantities. Derivations can be found in Appendix B.2. The density $p(s|\theta_t)$ becomes with these approximations a generalized von Mises density [254] and we obtain the following:

$$p(\theta_t|s) \propto \exp\left(\kappa_t \cos(\theta_t - \zeta_t) + \overline{\kappa}_t \cos(2(\theta_t - \overline{\zeta}_t)) - \frac{|\theta_t - g\hat{p}_t|^2}{2g^2 q_t^{\mathbf{P}}}\right). \tag{6.28}$$

where the expressions of $\kappa_t, \overline{\kappa}_t, \zeta_t, \overline{\zeta}_t$ are provided in Appendix B.2.

Hence $\hat{z}_t, q_t^{\mathbf{z}}$ can be estimated from $\hat{\theta}_t, \mathrm{Var}(\theta_t|s)$, which can themselves be estimated numerically using this density. By doing the same approximations for every $t \in [\![1, k]\!]$ and[17] $j \in [\![1, m]\!]$, we obtain estimates of all the desired quantities.

[17] We recall that the indexes on j have been omitted for readability only, but all the involved quantities here do depend on j implicitly.

Numerical estimation under the approximated posterior We now need to estimate $\mathbf{E}[\theta_t|s]$ and $\mathrm{Var}(\theta_t|s)$ under (6.28). It seems difficult to obtain a closed form, hence we need again here to use further approximations, or simply to compute these integrals numerically.

Laplace's method [255, Section 4.4], which is designed to approximate integrals of the form $\int \exp(f(x))\,\mathrm{d}x$ where f is a twice differentiable function which takes large values and has a unique maximum, might be relevant due to the form of the posterior. In that case we would compute the maximum a posterior estimate of θ_t, i.e. use $\hat{\theta}_t \approx \hat{\theta}_{t,\mathrm{MAP}} = \arg\max p(\theta_t|s)$, and estimate via a second-order Taylor expansion $\mathrm{Var}(\theta_t|s) \approx -\frac{d^2}{d\theta_t^2} \log p(\theta_t|s)\big|_{\theta_t=\hat{\theta}_{t,\mathrm{MAP}}}$. Evaluating $\hat{\theta}_{t,\mathrm{MAP}}$ is still not straightforward[18], but this can be estimated by numerical integration. We used a uniform grid of $n_{pts}n_{per} + 1$ points centered at $g\hat{p}_t$ with width $2\pi n_{per}$, where $n_{per} = \lceil n_{std}/\pi\sqrt{g^2 q_t^{\mathbf{P}}}\rceil$. The parameter n_{std} correspond to the number of standard deviations of the prior covered by the grid. Practical experiments suggest that a good tradeoff between computational performance and estimation quality could be obtained with $n_{std} = 4, n_{pts} = 7$.

[18] Especially because the distribution might be multimodal.

Yet, although extensive simulations have been performed using the Laplace method, it turns out that estimation of $\mathbf{E}[\theta_t|s]$ and $\mathrm{Var}(\theta_t|s)$ directly via numerical integration is often more robust. We used in that case again regularly-spaced grids centered on $g\hat{p}_t$, with a grid width and resolution depending on the order of magnitude of $g^2 q_t^{\mathbf{P}}$. We omit the technical details for brevity, and refer the interested reader directly to the implementation for more details (cf. Appendix D). Note however that implementing all that in an efficient and stable manner is not straightforward. Experiments suggest that the Von-Mises density does sometimes takes values of the order of 10^3 in the log-domain. Moreover, the distribution can be both unimodal or bimodal depending on the values of the parameters. Hence, and in spite of our multiple attemps to get a robust estimation procedure, it might happen that the algorithm "misses" the principal mode, or that only the principal mode is found but that all other values measured on the integration grid are so low that the variance is estimated to zero. Hence, after estimation of

the $(\mathbf{q}_j^{\mathbf{z}})_{1\leq j\leq m}$ we add a "clipping" step on these quantities to ensure that $\mathbf{q}^{\mathbf{s}}$ will be positive after averaging and avoid numerical issues[19]. We also use damping on $\hat{\mathbf{S}}, \mathbf{q}^{\mathbf{s}}, \hat{\mathbf{C}}$ and $\mathbf{q}^{\mathbf{p}}$ to mitigate numerical issues and prevent instability.

[19] Such tweaks are commonly used in GAMP implementations.

6.3.3 Hyperparameters learning

We recall that the likelihood (6.16) depends on parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$, which we for now assumed to be known. We propose to use the expectation-maximization (EM) algorithm alternately with SHyGAMP to estimate these parameters, i.e. we will alternate between a few SHyGAMP iterations and a few EM iterations. This approach has already been suggested and used previously for other tasks [249, 253]. For initialization, we chose $\alpha_l = 1/k$ and $\tau_l = 0$ for each $l \in [\![1, k]\!]$.

Derivation of EM steps From [249, eq. (23)], and denoting $\mathbf{Q}_j^{\mathbf{z}} \triangleq \mathrm{Diag}(\mathbf{q}_j^{\mathbf{z}})$ for each $j \in [\![1, m]\!]$, the EM update can be written

$$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\tau}}) = \underset{\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{1}=1, \boldsymbol{\tau}>0}{\arg\max} \sum_{j=1}^{m} \int_{\mathbb{R}^k} \mathcal{N}(\mathbf{z}_j; \hat{\mathbf{z}}_j, \mathbf{Q}_j^{\mathbf{z}}) \log p_{s|\mathbf{z}}(\tilde{s}_j|\mathbf{z}_j; \boldsymbol{\alpha}, \boldsymbol{\tau}) \, \mathrm{d}\mathbf{z}_j, \tag{6.29}$$

where $\hat{\mathbf{z}}_j$ and $\mathbf{q}_j^{\mathbf{z}}$ are obtained by running SHyGAMP to convergence under $(\boldsymbol{\alpha}, \boldsymbol{\tau})$. We approximate the Dirac in (6.16) by a circular Gaussian distribution with small variance ε, i.e.

$$\log p_{s|\mathbf{z}}(\tilde{s}_j|\mathbf{z}_j; \boldsymbol{\alpha}, \boldsymbol{\tau}) = -\frac{1}{\varepsilon}\left|\tilde{s}_j - \sum_{l=1}^{k} \alpha_l q_{jl} v_{jl}\right|^2 + c.$$

where c is a constant, and where we used the notations

$$q_{jl} \triangleq \exp(-\tfrac{1}{2} g_j^2 \tau_l),$$
$$v_{jl} \triangleq \exp(\iota g_j z_{jl}).$$

Using this approximation, and because optimization is independent of ε and c, we have from (6.29)

$$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\tau}}) = \underset{\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{1}=1, \boldsymbol{\tau}>0}{\arg\min} \sum_{j=1}^{m} \int_{\mathbb{R}^k} \mathcal{N}(\mathbf{z}_j; \hat{\mathbf{z}}_j, \mathbf{Q}_j^{\mathbf{z}}) \left|\tilde{s}_j - \sum_{l=1}^{k} \alpha_l q_{jl} v_{jl}\right|^2 \mathrm{d}\mathbf{z}_j \tag{6.30}$$

We propose to solve this optimization problem alternately on $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ by gradient projection [256]. Using the notation $\rho_{jl} \triangleq \exp(\iota g_j \hat{z}_{jl} - \frac{1}{2} g_j^2 q_{jl}^{\mathbf{z}})$, we have for each index $j \in [\![1, m]\!]$

$$\begin{aligned} f_j(\boldsymbol{\alpha}, \boldsymbol{\tau}) &\triangleq \int_{\mathbb{R}^k} \mathcal{N}(\mathbf{z}_j; \hat{\mathbf{z}}_j, \mathbf{Q}_j^{\mathbf{z}}) \left|\tilde{s}_j - \sum_{l=1}^{k} \alpha_l q_{jl} v_{jl}\right|^2 \mathrm{d}\mathbf{z}_j \\ &\overset{\text{(i)}}{=} |\tilde{s}_j|^2 - 2\sum_{l=1}^{k} \alpha_l q_{jl} \Re(\tilde{s}_j^* \rho_{jl}) \\ &\quad + \sum_{l=1}^{k} \alpha_l q_{jl} \rho_{jl}^* \sum_{1\leq t\neq l\leq k} \alpha_t q_{jt} \rho_{jt} + \sum_{l=1}^{k} \alpha_l^2 q_{jl}^2 \end{aligned}$$

(i) Using $\int_{\mathbb{R}} \mathcal{N}(z_{jl}; \hat{z}_{jl}, q_{jl}^{\mathbf{z}}) v_{jl} \, \mathrm{d}z_{jl} = \rho_{jl}$.

As a consequence, the gradients with respect to $(\alpha_l)_{1\leq l\leq k}$ and $(\tau_l)_{1\leq l\leq k}$ can be written

$$\frac{\partial}{\partial \alpha_l}\sum_{j=1}^{m} f_j(\boldsymbol{\alpha},\boldsymbol{\tau}) = -2\sum_{j=1}^{m} q_{jl}\gamma_{jl} \tag{6.31}$$

$$\frac{\partial}{\partial \tau_l}\sum_{j=1}^{m} f_j(\boldsymbol{\alpha},\boldsymbol{\tau}) = \alpha_l\sum_{j=1}^{m} g_j^2 q_{jl}\gamma_{jl} \tag{6.32}$$

where

$$\gamma_{jl} \triangleq \Re(\widetilde{s}_j^* \rho_{jl}) - \alpha_l q_{jl} - \sum_{1\leq t\neq l\leq k} \alpha_t q_{jt}\Re(\rho_{jl}^*\rho_{jt}) \tag{6.33}$$

Experimentally, computing the sums in eqs. (6.31) and (6.32) using only on a small subset of the indexes j improves performance without degrading too much accuracy.

We provide the main algorithm combining SHyGAMP with these EM steps in the next section (cf. Algorithm 6.4), after describing the initialization procedure. Note that the Bethe free energy could also be used as an approximation of the required log-densities (see for instance [257, Appendix C]). However, our work in this direction resulted in expressions which are not easier to work with.

6.3.4 Initialization and main algorithm

Let σ_κ^2 denote the variance of the kernel, which can be assumed to be known or estimated from the dataset[20]. Experiments suggest that a good initialization procedure for SHyGAMP (cf. Algorithm 6.2) is to draw $\hat{\mathbf{C}}_\mathbf{0}$ with i.i.d. entries from $\mathcal{N}(0,\sigma_\kappa^2)$ and $\mathbf{q}_0^\mathbf{P} = \sigma_\kappa^2\mathbf{1}$.

[20] We explained in our paper [26] using $\sigma_\kappa^2 = \|X\|_F^2/(nd)$, which indeed works well in many settings, but this might still sometimes be inaccurate. We refer the reader to Chapter 4 for this matter.

However, we observed that trying $R > 1$ different initializations sometimes helps to avoid spurious solutions. For each of them, i.e. for each $r \in [\![1, R]\!]$, we can run a few iterations of SHyGAMP to recover an estimated matrix of centroid $\hat{\mathbf{C}}^{(r)} = \left[\hat{\mathbf{c}}_1^{(r)}, ..., \hat{\mathbf{c}}_k^{(r)}\right]$ and then compute the estimated sketch $\hat{\mathbf{s}}^{(r)}$ defined as

$$\hat{s}_r^{(r)} = \sum_{l=1}^{k} \exp(\imath g_j \mathbf{a}_j^T \hat{\mathbf{c}}_l^{(r)} - g_j^2\tau_l), \tag{6.34}$$

whose expression follows from eqs. (6.11) and (6.12). Note that we use the notation $\hat{\mathbf{s}}$ to denote an estimator of the sketch, but this quantity should not be confused with the variable $\hat{\mathbf{S}}$ in Algorithm 6.2. The distance to the empirical sketch can then be used as a selection metric to keep only one of the sets of centroids, and proceed with the rest of the algorithm. We summarize this procedure in Algorithm 6.3.

We provide the main CL-AMP procedure (with initialization and EM steps) in Algorithm 6.4. Different criteria could be used to detect the convergence of this algorithm and of the SHyGAMP procedure, but the simplest one is maybe to measure the distance between the recovered and empirical sketches as proposed above for initialization. SHyGAMP provides indeed all the quantities required for this estimation, and computation is not too expensive. For internal calls to SHyGAMP, it will be convenient in practice to combine this criterion with a manually chosen maximum number of iterations.

Algorithm 6.3: Initialization of MMSE-SHyGAMP

function Init $(\tilde{\mathbf{s}}, \sigma_\kappa^2, \boldsymbol{\alpha}_0, \boldsymbol{\tau}_0)$
  $\mathbf{q}_0^{\mathbf{p}} \leftarrow \sigma_\kappa^2 \mathbf{1}$ // size k
  for $r \in [\![1, R]\!]$ **do**
    $\hat{\mathbf{C}}_{\mathbf{0}}^{(r)} \leftarrow d \times k$ matrix with i.i.d. entries drawn from $\mathcal{N}(0, \sigma_\kappa^2)$
    $(\hat{\mathbf{C}}^{(r)}, \hat{\mathbf{Z}}^{(r)}, ((\mathbf{q}_j^{\mathbf{z}})^{(r)})_{1 \le j \le m}) \leftarrow$ SHyGAMP$(\hat{\mathbf{C}}_{\mathbf{0}}^{(r)}, \mathbf{q}_0^{\mathbf{p}}, \boldsymbol{\alpha}_0, \boldsymbol{\tau}_0)$
    Compute $\hat{\mathbf{s}}^{(r)}$ according to (6.34)
  $r^* \leftarrow \arg\min_{1 \le r \le R} \|\hat{\mathbf{s}}^{(r)} - \tilde{\mathbf{s}}\|_2$
  return $(\hat{\mathbf{C}}^{(r^*)}, \hat{\mathbf{Z}}^{(r^*)}, ((\mathbf{q}_j^{\mathbf{z}})^{(r^*)})_{1 \le j \le m})$

Algorithm 6.4: Main CL-AMP algorithm

function CL-AMP $(\tilde{\mathbf{s}}, k, \sigma_\kappa^2)$
  $\boldsymbol{\alpha} \leftarrow \frac{1}{k}\mathbf{1}$ // k-dimensional
  $\boldsymbol{\tau} \leftarrow \mathbf{0}$ // k-dimensional
  $(\hat{\mathbf{C}}, \hat{\mathbf{Z}}, (\mathbf{q}_j^{\mathbf{z}})_{1 \le j \le m}) \leftarrow$ Init $(\tilde{\mathbf{s}}, \sigma_\kappa^2, \boldsymbol{\alpha}, \boldsymbol{\tau})$
  repeat
    Update $(\boldsymbol{\alpha}, \boldsymbol{\tau})$ using eqs. (6.30) to (6.32), $\hat{\mathbf{Z}}$ and $(\mathbf{q}_j^{\mathbf{z}})_{1 \le j \le m}$
    $(\hat{\mathbf{C}}, \hat{\mathbf{Z}}, (\mathbf{q}_j^{\mathbf{z}})_{1 \le j \le m}) \leftarrow$ SHyGAMP$(\tilde{\mathbf{s}}, \hat{\mathbf{C}}, \mathbf{q}_0^{\mathbf{p}}, \boldsymbol{\alpha}, \boldsymbol{\tau})$
  until *convergence*
  return estimated centroids $\hat{\mathbf{C}}$

6.4 Experimental results

We perform extensive empirical simulations, first on synthetic data in Section 6.4.1, and on the MNIST dataset in Section 6.4.2.

Despite all the modeling possibilities for the prior covered by (6.17), we will use in the experiments the non-informative prior $p_{\mathbf{c}}(\mathbf{c}^i) \propto 1$ for every $i \in [\![1, d]\!]$, which already gives good results. Although we justified the derivation above using a true prior distribution, the obtained algorithm can easily be adapted to the case of a non-informative prior.

6.4.1 Synthetic data

In this subsection, we generate data according to the Gaussian mixture model described at Equation (4.5), where the intra-variance is computed using a separation factor fixed to $s = 2.5$.

Frequency distribution In the experiments below, we use the "adapted radius" distribution $\mathrm{AR}(\sigma_\kappa^2)$ suggested in [4] and briefly discussed in Section 4.2.6. It indeed gives slightly better results in comparison with a Gaussian frequency distribution, and helps with numerical issues. The kernel variance[21] σ_κ^2 for the figures below is chosen following the heuristic provided in (4.14).

[21] Compared to [4], we normalized the distribution so that $\mathbf{E}_{\varphi \sim \mathrm{AR}(\sigma_\kappa^2)}(\|\varphi\|_2^2) = 1/\sigma_\kappa^2$.

Sketch size We fix the kernel scale, the dimension d and number of clusters k, and look at the impact of the sketch size on clustering performance. We recall that recovery is expected to succeed for $m = \Omega(kd)$, hence we use $m/(kd)$ as parametrization of the horizontal axis.

Results for $(k = 10, d = 10)$ and $(k = 20, d = 60)$ are presented in Figure 6.2.

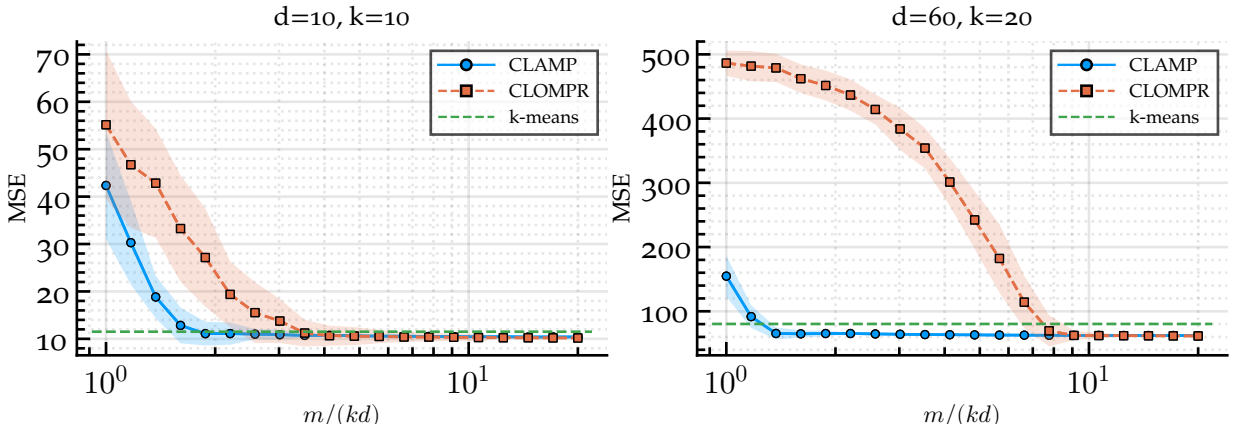


Figure 6.2: Clustering error versus sketch size. Frequencies drawn according to the "adapted radius" distribution, $n = 10^4$ points drawn according to the Gaussian mixture model (4.5) with separation $s = 2.5$. Medians and standard deviation (ribbon) over 100 trials.

We observe that CL-AMP is uniformly better than CL-OMPR. As previously observed [2], CL-OMPR succeeds provided that roughly $m \geq 5kd$ or $m \geq 10kd$ depending on the dimension, while CL-AMP works very well down to $m = 2kd$, and even a bit below. Note that this behavior is coherent with the observations on AMP and OMP in the traditional compressive sensing setting, see for instance [249, Fig. 8-10]. We also note that both methods perform better than k-means for a large enough sketch size. The performance of k-means could certainly be improved by increasing the number of repetitions and maximum number of iterations[22], but still suggests that the algorithm suffers from spurious local minima and that using a large number of random initializations is needed to obtain good performances, whereas compressive clustering is much more stable in comparison.

[22] Here and everywhere below unless otherwise specified, we use k-means++ with 3 trials.

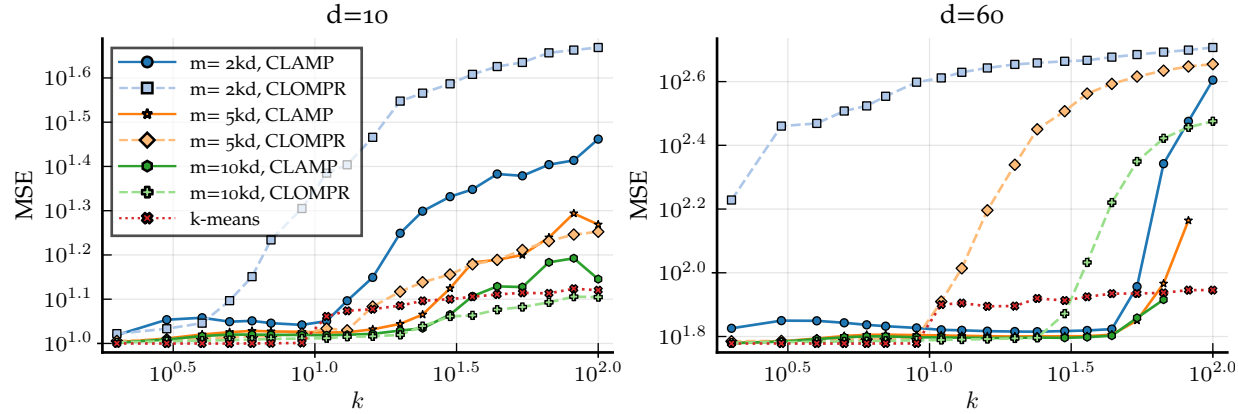


Figure 6.3: Clustering error versus number of clusters k. Frequencies drawn according to the "adapted radius" distribution, $n = 10^4$ points drawn according to the Gaussian mixture model (4.5) with separation $s = 2.5$. Medians over 100 trials for smaller values of k, and over a few trials for larger values of k (recall CL-OMPR scales in $\mathcal{O}(k^3)$, which makes it impractical for extensive simulations).

Number of clusters We now consider three different sketch sizes (namely $m = 2kd, 5kd, 10kd$), and look at the error as a function of the number of clusters k for $d = 10$ and $d = 60$.

Both compressive methods have increasing errors when k grows, which probably comes from the lack of separation between clusters. Figure 6.3 shows that CL-AMP yields in almost all settings lower errors

than CL-OMPR. This holds especially for $d = 60$, where CL-OMPR performs poorly, and/or when $m = 2kd$, confirming previous observations regarding the sketch size. It should be noted that the error of k-means does not increase too much with k in comparison with both CL-OMPR and CL-AMP. This suggest that the estimation of the kernel variance could also be problematic for values of k close to 100. The range of kernel variances in which compressive clustering is expected to perform well indeed becomes much thinner in this regime as observed in Section 4.2.5.

6.4.2 Real datasets

We now provide some empirical results on the MNIST dataset of handwritten digits [227], using the same spectral features as described in Section 5.3.2, and on the FMA dataset [211].

MNIST As discussed in Chapter 4, learning the optimal kernel variance from the dataset is necessary for practical applications. Yet, our goal here is mostly to compare CL-OMPR and CL-AMP algorithms, and both methods suffer from this problem. As a consequence, and in order to avoid drawing wrong conclusions because of a possibly inaccurate estimation of the optimal kernel variance, we will show the errors as functions of the kernel variance. Results are provided in Figure 6.4 for $m = 2kd$ and $m = 10kd$ in terms of both clustering error and adjusted rand index (ARI). The latter measures the similarity of the recovered clustering with the ground truth classes, and is computed so that a random clustering would have and ARI of zero in expectation. We consider the standard dataset comprising $n = 7 \times 10^4$ images, as well as a larger collection of $n = 10^6$ samples including distorted variants of these images generated using infiMNIST [228].

Interestingly, for the original dataset ($n = 7 \times 10^4$) CL-OMPR seems here to perform well on a larger range of kernel variances than CL-AMP, contrarily to what has been observed on synthetic data. CL-AMP however has slightly lower error at the optimal kernel variance, and almost always a smaller variance. In terms of adjusted rand index, CL-AMP is much better than CL-OMPR for both sketch sizes, with a clear gap between the two methods. Similar conclusions can be drawn for the extended dataset in terms of MSE, however both methods perform well in this case in terms of ARI, with a slightly reduced variance in favor of CL-AMP.

FMA dataset We also report in Figure 6.5 some results obtained on the "FMA" dataset[23] of audio features [211].

Dimension is $d = 518$ for this dataset, and the number of clusters as been arbitrarily chosen to $k = 20$. Data is centered and standardized, as features of very different natures are present in the collection. No ground truth classification can be used here, so we simply report the mean squared error as a function of the kernel scale for both CL-AMP and CL-OMPR. Both methods reach similar error levels, however none

[23] cf. https://github.com/mdeff/fma

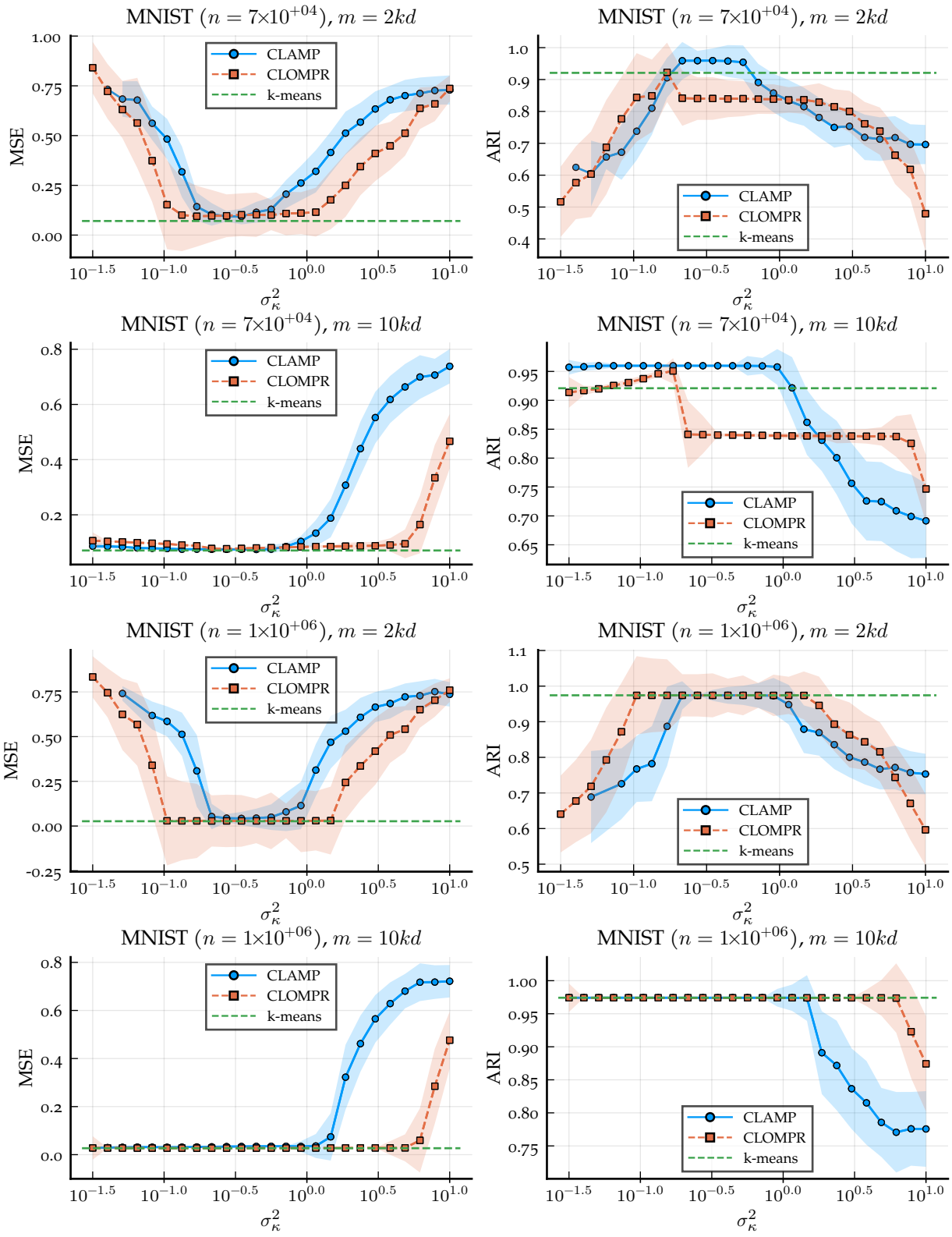


Figure 6.4: Clustering error and adjusted rand index (ARI) versus kernel scale σ_κ^2. Frequencies drawn according to the "adapted radius" distribution. Medians and standard deviation (ribbon) over 100 trials. k-means is run with three trials and kmeans++ initialization.

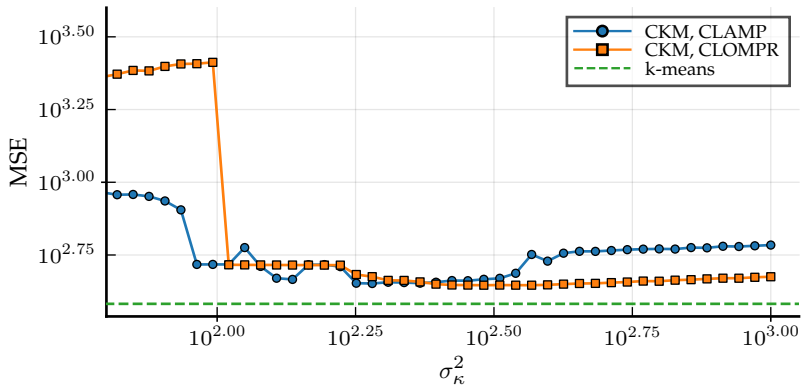


Figure 6.5: Clustering error (medians over 10 trials) versus kernel scale σ^2_κ on the (centered and standardized) FMA dataset. Using $m = 10kd$, dense Gaussian frequencies.

of the two matches k-means results. CL-OMPR seems however to performs well over a larger range of kernel variances in comparison with CL-AMP. This tends to confirm that CL-AMP is more sensitive to the tuning of the different parameters.

6.5 Perspectives

We introduced in this chapter a new method to recover cluster centers from a sketch of averaged random Fourier features. Our algorithm is based on the generic simplified hybrid generalized approximate message passing algorithm (SHyGAMP), and derived under a Gaussian mixture data generation model. It also relies on several approximations of the mean and covariance posteriors which are specific to our model.

The proposed method compares favourably to the continuous version of orthogonal matching pursuit (CL-OMPR), both on synthetic data and spectral features computed on the MNIST dataset, and has a lower computational complexity. We thus advise using CL-AMP rather than CL-OMPR in general, especially in settings where the number of clusters k to recover is large.

It should be noted, however, that CL-OMPR is more generic — it can for instance be used for Gaussian modeling using the same sketch, which would be much more difficult with the proposed method —, and still sometimes yields lower errors than CL-AMP on datasets whose distributions differ largely from a Gaussian mixture model. Furthermore, despite the promising performance which have been reported, we stress that tuning CL-AMP is not straightforward due to all the numerical estimations discussed in section 6.3.2, and thus this method might need more adjustments in different settings.

We do not report precise runtimes as it is difficult to fairly compare both methods, and one would need more engineering on the implementations of both algorithm to get meaningful results. With the settings that we have chosen, CL-AMP seems in general to be more efficient than CL-OMPR, but it should be noted that the maximum number of iterations in CL-AMP is arbitrarily chosen and has a significant impact on the complexity. In particular, when $m \approx kd$ CL-OMPR scales with k in $\Theta(k^3)$ while CL-AMP only in $\Theta(k^2)$, but the former has a fixed number of iterations while it is not easy to characterize how

the maximum number of iterations in CL-AMP should grow with k – and measuring numerically the convergence alone is not sufficient as a stopping criteria.

Also, we quickly tried with success using the fast transforms from Chapter 5 with CL-AMP. We do not compare the two methods on this criterion as they both benefit from the speedup in the same manner.

Part IV

Compressive Learning with Privacy Guarantees

We provide an introduction to differential privacy in Chapter 7, introduce the private sketching mechanisms in Chapter 8, and assess their performance in terms of learning quality in Chapter 9.

Chapter 7
Introduction to Differential Privacy and Related Work

Contents

In addition to its efficient use of computational resources, sketching is a promising tool for privacy-preserving machine learning. In numerous applications, such as when working with medical records, online surveys, or measurements coming from personal devices, data samples contain sensitive personal information and data providers ask that individuals' contributions to the dataset remain private, i.e. not publicly discoverable. Learning from such data collections while protecting the privacy of individual contributors has become a crucial challenge. Indeed, even releasing intermediate quantities computed from a collection of people's records – e.g. a machine learning model or aggregate statistics – rather than the raw data can already compromise the privacy of these users, even when aggregation is performed over millions of data providers [258].

We will explain in Chapter 8 how the sketching mechanism (1.10) can be adapted to ensure privacy, however we need for that to define formally what is meant by "privacy preservation". Differential privacy (DP) was proposed as a strong definition of privacy by Dwork et al. [259], and has since been studied and used extensively in research and industry [260, 261].

The goal of this chapter is thus to introduce this notion, and to explain the standard techniques which can be used to make an existing algorithm differentially private.

We give in Section 7.2 a brief introduction to DP, and also detail the assumptions made on the attacker (the attack model), which have a direct impact on the kind of guarantees that can be achieved. We then introduce in Section 7.3 the Laplace and Gaussian mechanisms, which are two standard methods to obtain differential privacy by adding carefully calibrated noise on the quantity to release. The differentially private sketching mechanism which will be introduced later in Chapter 8 relies on these ideas. A few methods achieving differential privacy via other tools are presented in Section 7.4, with a focus on the k-means, Gaussian modeling and PCA problems, as we will only tackle these three tasks with our compressive approach.

7.1 Attack model

Before providing the definition of differential privacy, it is important to specify an attack model. We consider a trusted curator model, where a trusted entity has access to the data, and publishes a noisy summary – in our case, a sketch of the data. The adversary is non-interactive, in that they have full access to the sketch of the dataset, or to sketches of disjoint subsets of the dataset if the latter is distributed across multiple devices (Figure 7.1), but cannot query the curator(s) for more data. On the figure, $\mathbf{z}(\mathbf{X})$ denotes the "private" sketch of the data, whose definition is postponed to later.

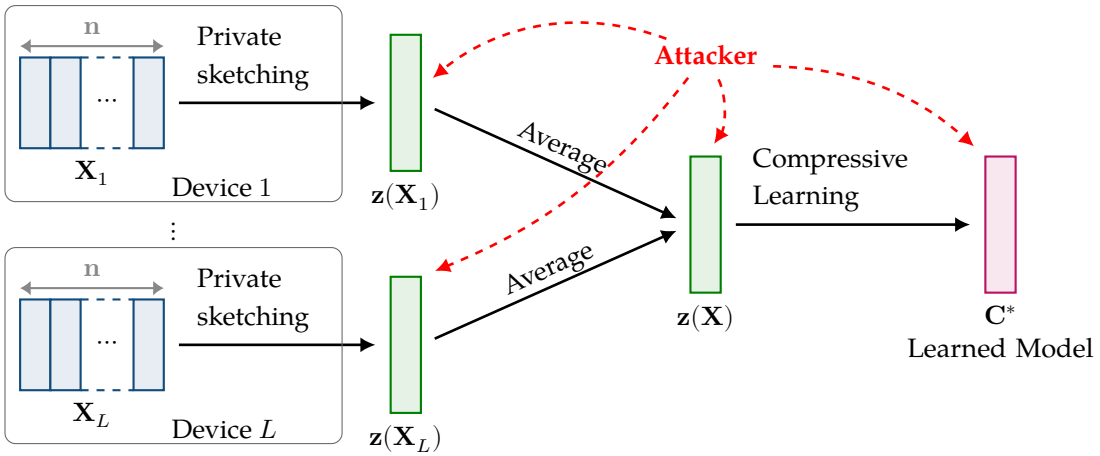


Figure 7.1: Attack model. The dataset is distributed between L devices, each computing and releasing publicly a subsampled sketch $\mathbf{z}(\mathbf{X}_i)$.

The feature map Φ and the matrix of frequencies $\boldsymbol{\Omega}$ used for sketching are assumed to be publicly known, in contrast to some approaches where random projection matrices are used as encryption keys [262]. This is essential for analysts, who need to know the feature map in order to learn from the sketch. The model also covers the case where analysts may be adversaries. We assume that each user contributes exactly one record to the total dataset, albeit our results can be extended to allow for multiple records per user. We do not make any assumptions on the background knowledge available to the adversary, nor on the operations that they are able to make. Hence, our privacy guarantees are robust to extreme cases where the adversary knows the entire database apart from one user, and has infinite computing power.

7.2 Definition and properties

We denote $\mathcal{D}_n \triangleq \mathcal{X}^n$ the set of (ordered) collections of n learning examples in the data domain $\mathcal{X}$, and $\mathcal{D} \triangleq \cup_{n\in\mathbb{N}}\mathcal{D}_n$. Hence datasets are seen depending on the context[1] either as matrices (as done in the previous chapters) or tuples of data samples – which we denote e.g. $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ –, and $|\mathbf{X}|$ denotes the size of the tuple, i.e. the number of samples in the dataset. We consider for now $\mathcal{X} = \mathbb{R}^d$, but we will restrict ourselves to the unit ball for the PCA application below. Note that we work with ordered datasets for technical reasons, but this order

[1] This is just for convenience, and there is naturally a bijection between the two spaces.

does not matter from a learning perspective.

Randomness is an old tool for introducing uncertainty[2] when using sensitive information, e.g. implemented as randomized response surveys [263]. Differential privacy [259] provides a formal definition of the privacy guarantees offered by a randomized data release mechanism. Intuitively, a mechanism R provides differential privacy if its output does not depend significantly on the presence of any one user in the database, hence hiding this presence from an adversary.

[2] Sometimes referred to as "privacy by plausible deniability".

Definition 7.1 (Differential privacy [259]): The randomized mechanism $R : \mathcal{D} \to \mathcal{Z}$ achieves ε-differential privacy (noted ε-DP) iff for any measurable set[3] S of $\mathcal{Z}$ and $\mathbf{X}, \mathbf{Y} \in \mathcal{D}$ s.t. $\mathbf{X} \sim \mathbf{Y}$ for some neighboring relation $\sim$ (see below):

$$\mathbb{P}[R(\mathbf{X}) \in S] \leq \exp(\varepsilon)\, \mathbb{P}[R(\mathbf{Y}) \in S]. \tag{7.1}$$

The parameter $\varepsilon > 0$ is called the privacy budget.

3. The codomain $\mathcal{Z}$ is implicitly endowed with a σ-algebra. Note that the choice of the σ-algebra does have an impact on the privacy definition. In the following, our sketching mechanisms take values in $\mathcal{Z} = \mathbb{R}^m$ or $\mathcal{Z} = \mathcal{C}^m$, and we always use the Borel σ-algebra to get strong privacy guarantees.

The smaller ε, the closer the output distributions for two neighboring datasets are, and the stronger the privacy guarantee. Equivalently, differential privacy can be defined through the notion of *privacy loss* of a randomized mechanism. This is particularly useful when proving that a mechanism is differentially private.

Definition 7.2 (Privacy loss [264]): Let R be a randomized algorithm taking values in $\mathcal{Z}$. If R admits a density $p_{R(\mathbf{X})}$ over $\mathcal{Z}$ for each input $\mathbf{X}$, the privacy loss *function* is defined by

$$L_R(\mathbf{z}, \mathbf{X}, \mathbf{Y}) \triangleq \log\left(\frac{p_{R(\mathbf{X})}(\mathbf{z})}{p_{R(\mathbf{Y})}(\mathbf{z})}\right).$$

The random mechanism R achieves ε-differential privacy iff

$$\sup_{\substack{\mathbf{z}\in\mathcal{Z} \\ \mathbf{X},\mathbf{Y}\in\mathcal{D}:\, \mathbf{X}\sim\mathbf{Y}}} L_R(\mathbf{z}, \mathbf{X}, \mathbf{Y}) \leq \varepsilon.$$

Intuitively, small values of the privacy loss L_R of R for some pair $\mathbf{X}, \mathbf{Y}$ characterize regions of the co-domain where output random variables $R(\mathbf{X})$ and $R(\mathbf{Y})$ have "close" distributions.

7.2.1 Neighboring relation

The neighboring relation $\sim$ in Definition 7.1 defines the practical guarantees that differential privacy offers. A common definition, called unbounded differential privacy (UDP), states that two datasets are neighbors if they differ by the addition or deletion of exactly one sample. From definition 7.1, this implies that the output of an algorithm that satisfies unbounded DP does not significantly depend on the presence of any one user in the dataset. An alternative is bounded DP (BDP), which defines two datasets as neighbors if and only if they differ by exactly one record by replacement.

We recall that $[\![1, n]\!] = \{1, ..., n\}$, and we also denote $\mathcal{S}_n$ the permutation group of $\{1, \dots, n\}$ and $\sigma(\mathbf{X})$ a permuted collection: $\sigma((\mathbf{x}_1, ..., \mathbf{x}_n)) \triangleq (\mathbf{x}_{\sigma(1)}, ..., \mathbf{x}_{\sigma(n)})$ for any $\sigma \in \mathcal{S}_n$.

Definition 7.3: An algorithm provides ε-unbounded DP (UDP) iff it provides ε-DP for the "removal" neighborhood relation $\overset{\text{U}}{\sim}$, defined as

$$\mathbf{X} \overset{\text{U}}{\sim} \mathbf{Y} \Leftrightarrow \begin{cases} ||\mathbf{X}| - |\mathbf{Y}|| = 1 \\ \exists \sigma \in \mathcal{S}_{|\mathbf{X}|} \text{ s.t. } \sigma(\mathbf{X}) \overset{\text{U}}{\approx} \mathbf{Y}, \end{cases}$$

where, assuming w.l.o.g. $|\mathbf{X}| = |\mathbf{Y}| + 1 \triangleq n \geq 2$, we define $((\mathbf{x}_1, ..., \mathbf{x}_n) \overset{\text{U}}{\approx} (\mathbf{y}_1, ..., \mathbf{y}_{n-1})) \Leftrightarrow ((\forall i \in [\![1, n-1]\!], \mathbf{x}_i = \mathbf{y}_i)$ and $\mathbf{x}_n$ is arbitrary).

Definition 7.4: An algorithm provides ε-bounded DP (BDP) iff it provides ε-DP for the "replacement" neighborhood relation $\overset{\text{B}}{\sim}$:

$$\mathbf{X} \overset{\text{B}}{\sim} \mathbf{Y} \Leftrightarrow |\mathbf{X}| = |\mathbf{Y}| \text{ and } \exists \sigma_1, \sigma_2 \in \mathcal{S}_{|\mathbf{X}|} \text{ s.t. } \sigma_1(\mathbf{X}) \overset{\text{B}}{\approx} \sigma_2(\mathbf{Y}),$$

where $((\mathbf{x}_1, ..., \mathbf{x}_n) \overset{\text{B}}{\approx} (\mathbf{y}_1, ..., \mathbf{y}_n)) \Leftrightarrow (\forall i \in [\![1, n-1]\!], \mathbf{x}_i = \mathbf{y}_i$ and $\mathbf{x}_n, \mathbf{y}_n$ are arbitrary).

We assumed $|\mathbf{X}| = |\mathbf{Y}| + 1$ in the definition for succinctness only, but the relation $\overset{\text{U}}{\sim}$ is symmetric. The key practical difference between the two definitions is that BDP assumes that the size of the dataset is not a sensitive value and can be published freely[4].

Unbounded differential privacy is a stronger definition, as an ε-UDP algorithm is necessarily[5] 2ε-BDP, while the reverse is not necessarily true. This 2ε bound might however not be tight. In the following, we will mainly consider the unbounded DP settings, which is sometimes more tricky. Most results will however also be provided for bounded DP.

[4] Indeed, $\mathbf{X} \overset{\text{B}}{\sim} \mathbf{Y}$ implies $|\mathbf{X}| = |\mathbf{Y}|$. We will come back on this matter a bit later.

[5] Using the composition lemmas presented below. Indeed not that if $\mathbf{X} \overset{\text{B}}{\sim} \mathbf{Y}$, then $\mathbf{X}$ can be obtained from $\mathbf{Y}$ by removing an element and adding a new one.

7.2.2 Composition properties

An important property of differential privacy is composition: using several differentially private algorithms on the same dataset results in similar guarantees, but with a total privacy budget equal to the sum of the budgets of the individual algorithms. Hence, one can design a complex DP algorithm by splitting its privacy budget ε between different simpler routines.

Lemma 7.5 (Sequential composition [265, Theorem 3])

Let $(R_i)_{1 \leq i \leq r}$ be a collection of DP mechanisms on the same domain with respective privacy budgets $(\varepsilon_i)_{1 \leq i \leq r}$. Then

$$R : \mathbf{X} \mapsto (R_1(\mathbf{X}), \dots, R_r(\mathbf{X}))$$

provides $(\sum_{i=1}^{r} \varepsilon_i)$-DP.

This holds for both bounded and unbounded DP. Parallel composition can also be performed; the following lemma however holds only in the unbounded case.

Lemma 7.6 (Parallel composition [265, Theorem 4])

Let $(R_i)_{1\leq i\leq r}$ be a collection of independent ε-UDP algorithms on the same domain $\mathcal{D}$, and $\mathcal{D}_i$ be disjoint subsets of $\mathcal{D}$. Then

$$R : \mathbf{X} \mapsto (R_1(\mathbf{X} \cap \mathcal{D}_1), \dots, R_r(\mathbf{X} \cap \mathcal{D}_r))$$

provides ε-UDP, where $(\mathbf{x}_1, \dots, \mathbf{x}_n) \cap \mathcal{D}_j$ denotes the subtuple with original ordering of the samples $(\mathbf{x}_i)_{1\leq i\leq n}$ that are in $\mathcal{D}_j$.

These lemmas hold only when the R_i are differentially private according to the same neighboring relation between datasets. Note also that privacy is robust to post-processing: if a mechanism R is ε-DP, then $f(R(\cdot))$ is also ε-DP for any function f. Thus Lemma 7.6 implies in particular that in a distributed setting, each data holder can compute and release an ε-DP synopsis of its local data (e.g. a noisy sketch), and merging these quantities will lead to a global synopsis which is also ε-DP with respect to the whole dataset.

7.2.3 Alternative privacy definitions

Many alternative definitions of privacy have been proposed in the literature [266]. Traditional statistical disclosure control metrics, such as k-anonymity [267], define anonymity as a property of the data, e.g. requiring that each user is indistinguishable from $k-1$ others. However, anonymizing large-scale high-dimensional data (such as, e.g., mobility datasets) was shown to be hard, due to the high uniqueness of users in such datasets [268]. Researchers have proposed to make privacy a property of the algorithm, enforcing for instance that the mutual information leakage is bounded [269]. Differential privacy is the most popular of such definitions, as it considers a worst-case adversary, and is hence "future-proof": no future release of auxiliary information can break the privacy guarantees. Connections between differential privacy and other information-theoretic definitions have also been investigated [270].

7.3 Standard perturbation mechanisms for differential privacy

7.3.1 The Laplace mechanism

In this section, we describe the Laplace mechanism [259], a very common and simple mechanism to release privately a function f computed over sensitive values. This mechanism adds Laplace noise to the function's output, whose scale ensures differential privacy. In the following, $\mathcal{L}(b)$ denotes the centered Laplace distribution[6] of parameter b.

[6] We recall that the density of the centered Laplace probability distribution of parameter b is $x \mapsto \frac{1}{2b}\exp\left(-\frac{|x|}{b}\right)$.

Definition 7.7 (Complex Laplace distribution): A random variable z follows a centered complex Laplace distribution of parameter b (denoted $z \sim \mathcal{L}^{\mathbb{C}}(b)$) iff its real and imaginary parts follow independently a real Laplace distribution of parameter b. In that case, z admits a density $p_z(z) \propto \exp(-(|\Re z| + |\Im z|)/b)$ and has variance $\sigma_z^2 = \mathbf{E}\big[|z|^2\big] = 4b^2$.

Definition 7.8 (Laplace mechanism): For any function $f : \mathcal{D} \to \mathbb{R}^m$ (resp. $\mathbb{C}^m$), we define the Laplace mechanism with parameter $b \in \mathbb{R}$ as the random mechanism $\mathbf{X} \mapsto f(\mathbf{X}) + \boldsymbol{\xi}$ where $(\xi_i)_{1 \leq i \leq m} \overset{\text{iid}}{\sim} \mathcal{L}(b)$ (resp. $\mathcal{L}^{\mathbb{C}}(b)$).

The Laplace mechanism provides differential privacy if the scale b of the noise is chosen carefully. This scale depends on the notion of sensitivity, which measures the maximum variation of a function between two neighboring datasets.

Definition 7.9 (l_1-sensitivity): The l_1-sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^m$ for a neighborhood relation $\sim$ is defined as

$$\Delta_1(f) \triangleq \sup_{\mathbf{X},\mathbf{Y} \in \mathcal{D} : \mathbf{X} \sim \mathbf{Y}} \|f(\mathbf{X}) - f(\mathbf{Y})\|_1. \tag{7.2}$$

This definition extends to complex-valued functions using the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$.

Throughout the paper, we will use superscripts Δ_1^{U} and Δ_1^{B} to denote sensitivities computed respectively w.r.t. the UDP and BDP neighboring relations. Dwork et. al [271] proved that the Laplace mechanism provides ε-differential privacy for the noise level $b = \Delta_1(f)/\varepsilon$. We propose below a straightforward extension of this result for the complex setting. Although only an upper bound on the sensitivity is required in order to prove that a mechanism is differentially private, we will also provide sharp bounds when possible, hence the notion of "optimal privacy level".

Theorem 7.10

Let $f : \mathcal{D} \to \mathbb{R}^m$ or $\mathbb{C}^m$. The Laplace mechanism applied on f is differentially private with optimal privacy budget $\varepsilon^* = \Delta_1(f)/b$. Alternatively for $\varepsilon > 0$, the lowest noise level yielding ε-differential privacy is given by $b^* = \Delta_1(f)/\varepsilon$. This holds for both bounded and unbounded DP, provided that the sensitivities are computed according to the relevant neighborhood relation.

Proof: Let $\mathbf{X}, \mathbf{Y} \in \mathcal{D}$ be such that $\mathbf{X} \sim \mathbf{Y}$. Let $p_{\mathbf{X}}$ and $p_{\mathbf{Y}}$ denote the probability densities of the Laplace mechanism applied on f for datasets $\mathbf{X}$ and $\mathbf{Y}$. In the real case, the privacy loss function

takes the form

$$L_f(\mathbf{z}, \mathbf{X}, \mathbf{Y}) = \log\left(\frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})}\right) = \frac{1}{b}(\|f(\mathbf{Y}) - \mathbf{z}\|_1 - \|f(\mathbf{X}) - \mathbf{z}\|_1)$$

Hence:

$$\sup_{\substack{\mathbf{z}\in\mathbb{R}^m \\ \mathbf{X},\mathbf{Y}\in\mathcal{D} \\ \text{s.t. } \mathbf{X}\sim\mathbf{Y}}} L_f(\mathbf{z}, \mathbf{X}, \mathbf{Y}) = b^{-1} \sup_{\substack{\mathbf{X},\mathbf{Y}\in\mathcal{D} \\ \text{s.t. } \mathbf{X}\sim\mathbf{Y}}} \sum_{j=1}^{m} \sup_{s_j\in\mathbb{R}} |f(\mathbf{Y})_j - \mathbf{z}_j| - |f(\mathbf{X})_j - \mathbf{z}_j|$$

$$\overset{(*)}{=} \sup_{\substack{\mathbf{X},\mathbf{Y}\in\mathcal{D} \\ \text{s.t. } \mathbf{X}\sim\mathbf{Y}}} \frac{\|f(\mathbf{X}) - f(\mathbf{Y})\|_1}{b} = \frac{\Delta_1(f)}{b}.$$

(*) The inequality $\leq$ follows from the triangle inequality, and $\mathbf{z}_j = f(\mathbf{Y})_j$ shows the equality.

In the complex case, the proof is similar but using the density of a complex Laplace variable (Definition 7.7), and the definition of l_1-sensitivity in the complex case.

Note that the function $f : \mathbf{X} \mapsto |\mathbf{X}|$ has UDP/BDP sensitivities $\Delta_1^{\text{U}}(f) = 1$ and $\Delta_1^{\text{B}}(f) = 0$, as all neighboring datasets have the same size for BDP. Releasing n publicly is therefore ε-BDP for any value of ε, but this is not the case with UDP. This confirms the intuition that UDP treats the dataset size as sensitive, while BDP does not.

7.3.2 Approximate Differential Privacy and the Gaussian Mechanism

Differential privacy is a very strong guarantee, and for many real-world tasks it can lead to severe degradations of the algorithms performance (utility) for small privacy budgets. For this reason, many relaxations of DP have been introduced, the most prominent of which is approximate differential privacy, also commonly called (ε, δ)-DP [271].

Definition 7.11 (Approximate differential privacy [271]): The randomized mechanism $R : \mathcal{X} \to \mathcal{Z}$ achieves (ε, δ)-approximate differential privacy (noted (ε, δ)-DP) for $\varepsilon > 0$, $\delta \geq 0$ iff for any measurable set S of $\mathcal{Z}$, and any $\mathbf{X}, \mathbf{Y} \in \mathcal{D}$ s.t. $\mathbf{X} \sim \mathbf{Y}$ for some neighboring relation:

$$\mathbb{P}[R(\mathbf{X}) \in S] \leq \exp(\varepsilon) \cdot \mathbb{P}[R(\mathbf{Y}) \in S] + \delta. \tag{7.3}$$

The most common mechanism to achieve (ε, δ)-DP is the Gaussian mechanism, adding Gaussian noise to the output of a function. As for the Laplace mechanism, we here consider potentially complex-valued outputs, and denote $z \sim \mathcal{N}^{\mathbb{C}}(0, \sigma^2)$ a random variable whose real and imaginary component are independently identically distributed as $\Re z, \Im z \sim \mathcal{N}(0, \sigma^2)$ (note that the variance of z then reads $\sigma_z^2 = 2\sigma^2$).

Definition 7.12 (Gaussian Mechanism): For any function $f : \mathcal{D} \to \mathbb{R}^m$ (resp. $\mathbb{C}^m$), we define the Gaussian mechanism with parameter σ as the random mechanism $\mathbf{X} \mapsto f(\mathbf{X}) + \boldsymbol{\xi}$ where

$(\xi_j)_{1\leq j\leq m} \overset{\text{iid}}{\sim} \mathcal{N}(0,\sigma^2)$ (resp. $\mathcal{N}^{\mathbb{C}}(0,\sigma^2)$).

The advantage of this DP relaxation is that the noise standard deviation needed for (ε,δ)-DP scales not with the l_1 but with the l_2 sensitivity of f, defined just below, which can be significantly smaller for many functions, including our sketching operator.

Definition 7.13 (l_2-sensitivity): The l_2-sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^m$ for a neighborhood relation $\sim$ is defined as $\Delta_2(f) \triangleq \sup_{\mathbf{X},\mathbf{Y}\in\mathcal{D}:\,\mathbf{X}\sim\mathbf{Y}} \|f(\mathbf{X}) - f(\mathbf{Y})\|_2$. This definition extends to complex-valued functions using the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$.

The "classical" noise calibration for the (real) Gaussian mechanism comes from [264, Appendix A], which shows that, assuming $\varepsilon < 1$, a standard deviation $\sigma > (2\ln(1.25/\delta))^{0.5}\Delta_2(f)/\varepsilon$ is sufficient to guarantee (ε,δ)-DP. This bound is commonly used but suboptimal, especially in the high privacy regime (i.e. small ε), and restricted to $\varepsilon < 1$. The calibration of the required noise parameter σ has recently been carefully tightened by Balle et al. [272], which is the mechanism we will use in this work[7].

[7] An implementation can be found at https://github.com/BorjaBalle/analytic-gaussian-mechanism.

Theorem 7.14 (Analytical Gaussian mechanism [272, Th. 9])

For each $\varepsilon,\delta > 0$, the lowest noise level σ^* such that the (real) Gaussian mechanism provides (ε,δ)-DP is given by $\sigma^* = \alpha(\varepsilon,\delta)\frac{\Delta_2(f)}{\sqrt{2\varepsilon}}$, where $\alpha(\varepsilon,\delta)$ is described in [272] and can be computed with a numerical algorithmic procedure.

The result holds for complex-valued feature maps as well using the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$, as applying the complex Gaussian mechanism on a complex-valued $\Phi(\cdot)$ is equivalent to applying the real Gaussian mechanism to $[\Re\Phi(\cdot);\Im\Phi(\cdot)]$, given that $\|[\Re\Phi(\cdot);\Im\Phi(\cdot)]\|_2 = \|\Phi(\cdot)\|_2$.

Note that simple composition theorems also exist for approximate differential privacy similarly to Lemma 7.5. We provide here only a result on sequential composition for succinctness, but results on parallel composition can be found in the literature as well.

Lemma 7.15 (Sequential composition [264, Theorem 3.16])

Let $(R_i)_{1\leq i\leq r}$ be a collection of (ε_i,δ_i)-DP mechanisms on the same domain. Then

$$R : \mathbf{X} \mapsto (R_1(\mathbf{X}), \dots, R_r(\mathbf{X}))$$

provides $(\sum_{i=1}^r \varepsilon_i, \sum_{i=1}^r \delta_i)$-DP.

We now explain how the privacy definitions introduced in this section can be satisfied with the sketching framework.

7.4 Private methods for the considered learning tasks: related work

We propose in this section to briefly review the most common methods[8] that have been introduced in the literature to achieve differential privacy. We focus on the three learning tasks which will be addressed using private sketches in the next chapter, which are Gaussian mixture modeling (GMM), PCA and k-means clustering. The two latter have already received a lot of attention in the differential privacy literature, while the former has been less studied. This overview is not exhaustive, and we refer the reader to [273] for an up-to-date survey of existing methods for private machine learning.

[8] We refer here to methods which either make multiple applications of the Laplace and/or Gaussian mechanisms, or rely on different tools.

Addition of noise is the most common way to achieve differential privacy, whether it is on the intermediate steps of an iterative algorithm or directly on the output. Private variants of standard iterative methods include DPLloyd for k-means [274], and variants with improved convergence guarantees [275]. The popular k-means++ seeding method has also been generalized to a private framework [276]. For Gaussian modeling, DP-GMM [277] and DP-EM [278] have been proposed. Note that for iterative algorithms, the privacy budget needs to be split between iterations, de facto limiting the total number of iterates, which becomes a hyper-parameter. The approach presented in the next chapter, which makes use of the Laplace and Gaussian mechanisms to release noisy sketches, does not suffer from this drawback since the sketch is released at once. Moreover, the same sketch can be used[9] to run the learning algorithm multiple times with e.g. different initializations.

[9] To some extent, as will be discussed in Chapter 9.

Releasing a private synopsis of the data (similarly to our sketch) rather than directly a noisy solution has already been studied as well. EUGkM [279, 280] suggests for instance to use noisy histograms for clustering (but this method is by nature limited to small dimensions), and private coresets[10] have been investigated by Feldman et al. [281, 282]. For PCA, noise can be added directly on the covariance matrix [283].

[10] See Section 3.3.1 for an introduction to corsets.

The exponential mechanism is another standard noise-additive approach for privacy. A random perturbation is drawn according to a distribution calibrated using a user-defined quality measure, and added to the output. It has been used with success for PCA, perturbing either the covariance [284, 285, 286] or directly the eigenvectors of the covariance [287, 288], and with genetic algorithms for k-means [289]. Such algorithms depend strongly on the quality measure of the output, which must be chosen carefully. Our sketch-based approach is in contrast more generic: the same sketch allows to solve different tasks such as clustering and GMM fitting, and it can easily be extended to new sketches in the future. Alternatively, our mechanism can be seen as a straightforward instantiation of the exponential mechanism, where the output (the sketch) is carefully designed so that is makes sense to simply use the l_1 or l_2 norms as quality measures.

Our sketching mechanism makes use of random projections, which

have proven to be very useful to solve efficiently large-scale problems, and induce as well a controlled loss of information which can be leveraged to derive privacy guarantees. Balcan et al. investigated the large-scale high-dimensional clustering setting with an approach based on Johnson-Lindenstrauss dimensionality reduction [290]. Many other embeddings based on random projections have been proposed, see e.g. [291]. Linear compression of the number of samples (rather than reducing the dimension) has been considered [292] but is less scalable. Random algorithms have also been used for PCA and, more generally, for low-rank factorization [293, 294, 295]. Note however that the features resulting from the random projection undergo in our setting a nonlinear transformation and are averaged; they thus differ a lot from what is done in these works, although they share this common idea. Sketches based on LSH[11] kernels have also recently been adapted to the private setting [296], and applied to various learning tasks such as classification or linear regression.

[11] cf. Chapter 3.

Private empirical risk minimization [297, 298] has emerged as a generic way to design private learning algorithms, but it relies on specific assumptions[12] on the loss function which defines the learning task, and still relies on multiple passes over the whole dataset.

[12] e.g. convexity, which does not hold for PCA, GMM modeling and k-means.

Closer to our work, Balog et al. recently proposed to release kernel mean embeddings [299], either as sets of synthetic data points in the input space or using feature maps, similarly to our method. However, to the best of our knowledge, the impact of privacy on the quality of learning in such methods has not been studied in the literature. Harder et al. also considered using kernel mean embeddings, however in the perspective of generating data [300].

Chapter 8

Differentially private sketching

Note A first version of this work with privacy upper-bounds and experimental results was originally published [29] and extended to the quantized setting [28]. Sharp bounds and results on the subsampling mechanism have been submitted to *Information and Inference* and are still under review [27].

Contents

We introduced in Chapter 7 the definition of differential privacy, as well as the standard mechanisms which can be used to satisfy this property. In this chapter, we explain how to leverage such mechanisms to produce differentially private sketches.

We will see that proving differential privacy, i.e. upper-bounding the privacy loss, can easily be done as the considered feature maps are bounded on the chosen domains[1] (Section 8.1). However, proving that these bounds are sharp is slightly more technical and calls for different mathematical tools. Then, we introduce in Section 8.2 an alternative private sketching mechanism which, besides using noise addition to ensure privacy, also subsamples the features of the individual samples. This yields a mechanism which, in the extreme case and when using quantized Fourier features[2], only measures one bit of data from each data sample in the dataset.

[1] Cf. Definitions 2.5 and 2.6.

[2] We refer here to the quantization of the nonlinear function, as explained in Definition 2.5.

Summary of contributions The contributions of this chapter are as follows:

- We build on compressive learning and define a **noisy sketching mechanism**.
- We derive **sharp differential privacy guarantees** for this mechanism for three unsupervised learning tasks: k-means clustering, Gaussian mixture modeling and principal components analysis.
- We extend our framework to **subsampled sketches**, giving the same privacy guarantees for a lower computational cost.

All the privacy results are summarized at the end of the chapter in tables 8.1 and 8.2.

8.1 Privacy with noisy sketches

Sketching, as proposed in (1.10), is not sufficient per se to ensure the differential privacy of user contributions, despite the fact that the sketch

itself (which is just at most $m \ll nd$ real or complex numbers) cannot contain much information about each of the n samples $\mathbf{x}_i \in \mathbb{R}^d$. In particular, although the vectors $(\boldsymbol{\omega}_j)_{j=1}^m$ are randomly drawn, the sketching mechanism induced by a given set of such vectors is deterministic. We construct a noisy sketch, based on the Laplacian (resp. Gaussian) mechanism, that guarantees ε-differential privacy (resp. (ε, δ)-differential privacy).

The clean sketch $\tilde{\mathbf{s}}$ from (1.10) can be written $\tilde{\mathbf{s}} = \boldsymbol{\Sigma}(\mathbf{X})/|\mathbf{X}|$, where $\boldsymbol{\Sigma}(\mathbf{X}) \triangleq \sum_{i=1}^n \Phi(\mathbf{x}_i)$ denotes the sum of features and $|\mathbf{X}|$ the number of records. Our mechanism adds noise to the numerator and denominator separately, i.e. releases $(\boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi}, |\mathbf{X}| + \zeta)$ where both $\boldsymbol{\xi}$ and ζ are random. Both quantities are thus made private provided that the noise levels are properly chosen, as discussed in the following subsections. This also means that we can further average sketches after computation in a distributed setting.

The sketch $\tilde{\mathbf{s}}$ can then be estimated from these two quantities, e.g. using $\mathbf{z}(\mathbf{X}) \triangleq (\boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi})/(|\mathbf{X}| + \zeta)$, which is private by composition properties of differential privacy. Note that DP is also robust to post-processing, so one could for instance replace $|\mathbf{X}| + \zeta$ by $\max(|\mathbf{X}| + \zeta, 1)$ to avoid dividing by a null or negative quantity. The noise $\boldsymbol{\xi}$ added to $\boldsymbol{\Sigma}$ can be either Laplacian or Gaussian, and we provide guarantees for both cases respectively in Section 8.1.1 and Section 8.1.2, each time for both random Fourier features and PCA. In the following, we use the notations $\boldsymbol{\Sigma}^{\mathrm{RFF}}$ and $\boldsymbol{\Sigma}^{\mathrm{RQF}}$ when $\boldsymbol{\Sigma}$ is computed using respectively $\Phi = \Phi^{\mathrm{RFF}}$ and $\Phi = \Phi^{\mathrm{RQF}}$.

8.1.1 Private Sketching with the Laplace Mechanism

We introduce formally the noisy sum of features.

Definition 8.1: The noisy sum of features $\boldsymbol{\Sigma}_{\mathcal{L}}$ of a dataset $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathcal{D}_n$ with noise parameters b is the random variable

$$\boldsymbol{\Sigma}_{\mathcal{L}}(\mathbf{X}) = \boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi},$$

where $\boldsymbol{\Sigma}(\mathbf{X}) \triangleq \sum_{i=1}^n \Phi(\mathbf{x}_i)$ and

$$\forall j \in [\![1, m]\!],\ \xi_j \overset{\text{iid}}{\sim} \begin{cases} \mathcal{L}^{\mathbb{C}}(b) & \text{if } \Phi \text{ is complex-valued} \\ \mathcal{L}(b) & \text{if } \Phi \text{ is real-valued} \end{cases}. \tag{8.1}$$

The scale of the noise will depend on the feature map used.

Remember that we need in practice an estimate of the sketch, and not only of the sum of features. In the BDP setting, we can simply divide the private sum of features by n, which is considered to be public; however we cannot proceed like that in the UDP case. We introduce a generic lemma for this purpose.

Lemma 8.2

For any privacy parameter $\varepsilon > 0$ and any choice of $\varepsilon_1, \varepsilon_2 > 0$ such that $\varepsilon_1 + \varepsilon_2 = \varepsilon$, if $\boldsymbol{\Sigma}$ has a finite sensitivity $\Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma})$, then any mechanism to estimate $\tilde{\mathbf{s}}$ using $\boldsymbol{\Sigma}_{\mathcal{L}}(\mathbf{X})$ instantiated with a noise level $b = \Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma})/\varepsilon_1$ and $|\mathbf{X}| + \zeta$, where $\zeta \sim \mathcal{L}(\varepsilon_2^{-1})$, is ε-UDP.

Proof: The Laplace mechanism applied on $\boldsymbol{\Sigma}$ with $b = \Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma})/\varepsilon_1$ is ε_1-UDP according to Theorem 7.10. Releasing $|\mathbf{X}|$ has sensitivity 1, and thus releasing $|\mathbf{X}| + \zeta$ with $\zeta \sim \mathcal{L}(\varepsilon_2^{-1})$ is ε_2-UDP. The result comes from the sequential composition Lemma 7.5.

To prove differential privacy of the sketching mechanism, we thus only need to compute the sensitivity $\Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma})$ of the sum-of-features function. We will see in Section 8.1.2 that a similar result can be stated for the Gaussian mechanism using the l_2 sensitivity. We introduce in the following lemma a common expression to deal with the different settings[3].

[3] i.e. Laplacian and Gaussian mechanisms, real- and complex-valued feature maps

Lemma 8.3

Let $\boldsymbol{\Sigma} : (\mathbf{x}_1, ..., \mathbf{x}_n) \mapsto \sum_{1 \leq i \leq |\mathbf{X}|} \Phi(\mathbf{x}_i)$ where Φ is any feature map taking values in $\mathbb{R}^m$ or $\mathbb{C}^m$. For $p = 1, 2$, the l_p sensitivity of Φ for datasets on a domain $\mathcal{X}$ and for the unbounded neighboring relation $\overset{\mathrm{U}}{\sim}$ is

$$\Delta_p^{\mathrm{U}}(\boldsymbol{\Sigma}) = \sup_{\mathbf{x} \in \mathcal{X}} Q_p^{\mathrm{U}}(\mathbf{x})$$

where $Q_p^{\mathrm{U}}(\mathbf{x}) = \|\Phi(\mathbf{x})\|_p$ for real-valued features maps, and extends to complex-valued feature maps using the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$.

Note that in particular, $Q_1^{\mathrm{U}}(\mathbf{x}) = \|\mathcal{R}(\Phi(\mathbf{x}))\|_1 + \|\mathcal{I}(\Phi(\mathbf{x}))\|_1$ for a complex-valued feature map Φ.

Proof: Remember that $\mathcal{D} = \bigcup_{n \in \mathbb{N}} \mathcal{X}^n$. For a real-valued feature map Φ, we have by Definitions 7.9 and 7.13

$$\begin{aligned} \Delta_p^{\mathrm{U}}(\boldsymbol{\Sigma}) &= \sup_{\mathbf{X}, \mathbf{Y} \in \mathcal{D} : \mathbf{X} \overset{\mathrm{U}}{\sim} \mathbf{Y}} \|\boldsymbol{\Sigma}(\mathbf{X}) - \boldsymbol{\Sigma}(\mathbf{Y})\|_p \\ &= \sup_{\substack{\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathcal{D}, \\ \mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_{n-1}) \in \mathcal{D}, \\ \text{such that } \mathbf{X} \overset{\mathrm{U}}{\sim} \mathbf{Y}}} \left\| \sum_{i=1}^{n} \Phi(\mathbf{x}_i) - \sum_{i=1}^{n-1} \Phi(\mathbf{y}_i) \right\|_p \end{aligned} \tag{8.2}$$

By Definition 7.3 of the unbounded neighboring relation, $\mathbf{X} \overset{\mathrm{U}}{\sim} \mathbf{Y}$ implies that all the samples $(\mathbf{y}_i)_{1 \leq i \leq n-1}$ appear in $\mathbf{X}$ as well, which means all but one of the terms cancel between the two sums in the right hand side of (8.2). Hence

$$\Delta_p^{\mathrm{U}}(\boldsymbol{\Sigma}) = \sup_{\mathbf{x} \in \mathcal{X}} \|\Phi(\mathbf{x})\|_p \tag{8.3}$$

The result extends to the complex case using the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$, with $\|[\mathcal{R}(\Phi(\mathbf{x})); \mathcal{I}(\Phi(\mathbf{x}))]\|_2 =$

$\|\Phi(\mathbf{x})\|_2$ and $\|[\mathcal{R}(\Phi(\mathbf{x}));\mathcal{I}(\Phi(\mathbf{x}))]\|_1 = \|\mathcal{R}(\Phi(\mathbf{x}))\|_1 + \|\mathcal{I}(\Phi(\mathbf{x}))\|_1$.

In the following, we use the notations $Q_p^{\text{RFF}}, Q_p^{\text{RQF}}$ when the feature maps $\Phi^{\text{RFF}}, \Phi^{\text{RQF}}$ are used. Note however that Lemma 8.3 is generic, and could be applied to any new feature map in the future. We will compute both $\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RFF}})$ and $\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RQF}})$ using Lemma 8.3 below.

We will compute the sensitivities using the expression of the feature maps given in Definitions 2.5 and 2.6, but any constant factor c_Φ could be used in these expressions provided that the inverse problem is solved using the same scaling[4]; this would yield similar privacy guarantees, but the sensitivity and thus the noise level b would be multiplied by the same factor.

[4] One could for instance use $c_\Phi = 1/\sqrt{m}$ to get normalized features.

We discuss how to optimality split the privacy budget between ε_1 and ε_2 in Section 9.2.3.

Random Fourier Features We compute the l_1-sensitivity of $\boldsymbol{\Sigma}^{\text{RFF}}$ in Lemma 8.6. We first introduce a lemma on diophantine approximation, that will be needed to prove the sharpness of our bound.

Definition 8.4: The scalars $(\omega_j)_{1\leq j\leq m} \in \mathbb{R}$ are called nonresonant frequencies [301] if they are linearly independent over the rationals. The vectors $(\boldsymbol{\omega}_j)_{1\leq j\leq m} \in \mathbb{R}^d$ are called nonresonant frequency vectors if there exists a vector $\mathbf{v} \in \mathbb{R}^d$ such that the scalars $(\boldsymbol{\omega}_j^T\mathbf{v})_{1\leq j\leq m}$ are nonresonant frequencies.

Lemma 8.5

Let $(\varphi_j)_{1\leq j\leq m}$ be real numbers, $(\boldsymbol{\omega}_j)_{1\leq j\leq m} \in \mathbb{R}^d$ nonresonant frequencies, and f a 2π-periodic function such that there exists z at which f is continuous and reaches its maximum. Then $\sup_{\mathbf{x}\in\mathbb{R}^d} \inf_{j\in[\![1;m]\!]} f(\boldsymbol{\omega}_j^T\mathbf{x} - \varphi_j) = \sup_{x\in\mathbb{R}} f(x)$.

The proof of this result can be found in Appendix C.1. We can now compute the desired sensitivity.

Lemma 8.6

The function $\boldsymbol{\Sigma}^{\text{RFF}}$ built using m frequencies has sensitivity $\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RFF}}) \leq m\sqrt{2}$ for both quantized and unquantized cases. If the frequencies are non resonant then $\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RFF}}) = m\sqrt{2}$.

Proof: We write[5] $\Phi^{\text{RFF}}(\mathbf{x}) = \left(\rho(\boldsymbol{\Omega}^T\mathbf{x} + \mathbf{u}) + \imath\rho(\boldsymbol{\Omega}^T\mathbf{x} + \mathbf{u} - \frac{\pi}{2})\right)$ in order to deal with both unquantized ($\rho = \cos$, $\mathbf{u} = 0$) and quantized ($\rho = 2^{-1/2}\,\text{sign}\circ\cos$, $\mathbf{u} \in [0, 2\pi[^m$) mechanisms with the same formalism. Using the definition of Q_p from Lemma 8.3 in

[5] Cf. Definition 2.5.

the Laplace real case, we have

$$\begin{aligned} Q_1^{\text{RFF}}(\mathbf{x}) &\triangleq \|\Re(\Phi^{\text{RFF}}(\mathbf{x}))\|_1 + \|\Im(\Phi^{\text{RFF}}(\mathbf{x}))\|_1 \\ &= \sum_{j=1}^{m} |\rho(\boldsymbol{\omega}_j^T\mathbf{x} + u_j)| + |\rho(\boldsymbol{\omega}_j^T\mathbf{x} + u_j - \frac{\pi}{2})| \end{aligned}$$

Denoting $f(\cdot) \triangleq \rho(\cdot) + \rho(\cdot - \pi/2)$ we show that $|\rho(\cdot)| + |\rho(\cdot - \pi/2)| = \sup_{\varphi\in\{0,\pi/2,\pi,3\pi/2\}} f(\cdot - \varphi)$. Indeed, both $\rho = \cos$ and $\rho = 2^{-1/2}\,\text{sign}\circ\cos$ satisfy the property $\forall t : \rho(t) = -\rho(t-\pi)$, hence for each $t \in \mathbb{R}$:

$$\begin{aligned} +\rho(t) + \rho(t-\pi/2) &= f(t) \\ +\rho(t) - \rho(t-\pi/2) &= \rho(t) + \rho(t+\pi/2) = f(t+\pi/2) \\ -\rho(t) - \rho(t-\pi/2) &= -f(t) = f(t+\pi) \\ -\rho(t) + \rho(t-\pi/2) &= -f(t+\pi/2) = f(t+3\pi/2). \end{aligned}$$

As a result, denoting $f_{\mathbf{p}}(\mathbf{x}) \triangleq \sum_{j=1}^{m} f(\boldsymbol{\omega}_j^T\mathbf{x} - \varphi_j)$ for each $\mathbf{p} \in \mathbb{R}^m$, we obtain

$$\begin{aligned} Q_1^{\text{RFF}}(\mathbf{x}) &= \sum_{j=1}^{m} \sup_{\varphi_j\in\{0,\pi/2,\pi,3\pi/2\}} f(\boldsymbol{\omega}_j^T\mathbf{x} + u_j - \varphi_j) \\ &= \sup_{\mathbf{p}\in\{0,\pi/2,\pi,3\pi/2\}^m} f_{\mathbf{p}-\mathbf{u}}(\mathbf{x}). \end{aligned} \tag{8.4}$$

In the complex exponential case, we have $\rho = \cos$ and $f : x \mapsto \sqrt{2}\cos(x - \pi/4)$. In the quantized case as $\rho = 2^{-1/2}\,\text{sign}\circ\cos$, f is a piecewise constant function taking values $0, \sqrt{2}, 0, -\sqrt{2}$. Thus in both cases we have $\sup_{x\in\mathbb{R}} f(x) = \sqrt{2}$. We obtain $\sup_{\mathbf{x}\in\mathbb{R}^d} f_{\mathbf{p}-\mathbf{u}}(\mathbf{x}) \leq m\sqrt{2}$ for any $\mathbf{p}, \mathbf{u}$ hence, by Lemma 8.3, we get that $\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RFF}}) \leq m\sqrt{2}$ as claimed.

When the frequencies $(\boldsymbol{\omega}_j)_{1\leq j\leq m}$ are nonresonant, f being 2π periodic and admitting (in both quantized/unquantized cases) a point $z \in \mathbb{R}$ at which it reaches its maximum and is continuous, we apply Lemma 8.5 and get according to Lemma 8.3:

$$\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RFF}}) = \sup_{\mathbf{x}\in\mathbb{R}^d} Q_1^{\text{RFF}}(\mathbf{x}) = \sup_{\mathbf{p}\in\{0,\pi/2,\pi,3\pi/2\}^m} \sup_{\mathbf{x}\in\mathbb{R}^d} f_{\mathbf{p}-\mathbf{u}}(\mathbf{x}) = m\sqrt{2}, \tag{8.5}$$

where the supremum is independent of the choice of $\mathbf{p}$.

This result holds only for $\mathcal{X} = \mathbb{R}^d$. If the domain is restricted to e.g. $\mathcal{X} = \mathcal{B}_2 = \{\mathbf{x} : \|\mathbf{x}\|_2 \leq 1\}$ the upper bound may not be reached, even with nonresonant frequencies, so an improved privacy may be possible.

For this result to be applicable, we still need to prove that the frequencies are nonresonant in practice.

Lemma 8.7

Frequency vectors drawn i.i.d. according to a distribution which is absolutely continuous w.r.t. the Lebesgue measure are almost surely nonresonant.

Proof: The set of resonant frequencies has a zero Lebesgue measure. The reader can refer to [301, Corollary 9.3 p. 166] for a proof relying on strong incommensurability.

Random quadratic features To deal with random quadratic features, we restrict ourselves[6] to datasets whose elements are bounded by 1 in l_2-norm. The domain is thus $\mathcal{X} = \mathcal{B}_2 \triangleq \left\{x \in \mathbb{R}^d : \|x\|_2 \leq 1\right\}$, and we still use the notations $\mathcal{D}_n \triangleq \mathcal{X}^n$ and $\mathcal{D} \triangleq \cup_{n\in\mathbb{N}}\mathcal{D}_n$.

We recall that the random quadratic features are introduced in Definition 2.6. We will consider in the following two different sampling schemes[7] for the matrix $\boldsymbol{\Omega}$.

- **Gaussian:** the $(\boldsymbol{\omega}_i)_{1\leq i\leq m}$ are drawn as $\boldsymbol{\omega}_i \sim \mathcal{N}(0, d^{-1}I_d)$. Note that with this variance, we have $\mathbf{E}_{\boldsymbol{\omega}\sim\mathcal{N}(0,d^{-1}I_d)}\|\boldsymbol{\omega}\|_2^2 = 1$ for coherence with the next sampling scheme.
- **Union of orthonormal bases:** when m/d is an integer, we consider $\boldsymbol{\Omega} = [\mathbf{B}_1, ..., \mathbf{B}_{m/d}]$ where the $(\mathbf{B}_i)_{1\leq i\leq m/d}$ are $d \times d$ blocs whose columns form orthonormal bases of $\mathbb{R}^d$. This setup is useful for two reasons. It makes it possible to use structured blocs $\mathbf{B}_i$ for which the matrix-vector product can be computed efficiently using fast transforms (cf. Chapter 5), but it also yields sharp privacy guarantees (cf. discussion just below after Lemma 8.8).

We first provide a result which holds for a deterministic $\boldsymbol{\Omega}$.

Lemma 8.8

The function $\boldsymbol{\Sigma}^{\text{RQF}}$ built using a matrix of frequencies $\boldsymbol{\Omega} = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m]$, has sensitivity $\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RQF}}) = \|\boldsymbol{\Omega}\|_2^2$ where $\|\cdot\|_2$ denotes the spectral norm.

Proof: Let $\lambda_{\max}$ denote the largest eigenvalue function. We have according to Lemma 8.3

$$\begin{aligned}\Delta_1^{\text{U}}(\boldsymbol{\Sigma}^{\text{RQF}}) &= \sup_{\mathbf{x}\in\mathcal{B}_2} Q_1^{\text{RQF}}(\mathbf{x}) = \sup_{\mathbf{x}\in\mathcal{B}_2} \|\Phi^{\text{RQF}}(\mathbf{x})\|_1 \\ &= \sup_{\mathbf{x}\in\mathcal{B}_2} \sum_{j=1}^{m} (\omega_j^T\mathbf{x})^2 = \sup_{\mathbf{x}:\|\mathbf{x}\|\leq 1} \mathbf{x}^T(\sum_{j=1}^{m}\boldsymbol{\omega}_j\boldsymbol{\omega}_j^T)\mathbf{x} \\ &= \lambda_{\max}(\boldsymbol{\Omega}\boldsymbol{\Omega}^T) = \|\boldsymbol{\Omega}\|_2^2.\end{aligned}$$

The quantity $\|\boldsymbol{\Omega}\|_2^2$ can be computed numerically for a given $\boldsymbol{\Omega}$. When m is a multiple of d and $\boldsymbol{\Omega}$ is a concatenation of m/d orthonormal bases as detailed above, we have $\boldsymbol{\Omega}\boldsymbol{\Omega}^T = m/d\mathbf{I}_d$ and thus

$$\|\boldsymbol{\Omega}\|_2^2 = m/d. \tag{8.6}$$

When $\boldsymbol{\Omega}$ has i.i.d. $\mathcal{N}(0, 1/d)$ entries, $\|\boldsymbol{\Omega}\|_2^2$ is of the same order with

[6] This assumption is required to obtain privacy guarantees as Φ^{RQF} is unbounded on $\mathbb{R}^d$, but is a common assumption in the literature for differentially private PCA.

[7] Note that both dense and structured matrices can also be used for random Fourier features (provided that the radial distribution is wisely chosen). Structured designs however do not yield sharp bounds in this case, contrarily to the RQF setting, which is why we make this distinction here.

high probability.

8.1.2 Approximate Differential Privacy with the Gaussian Mechanism

In practice, in order to increase the utility of private mechanisms relying on additive perturbations, ε-DP is often relaxed to approximate (ε, δ)-DP. In this section we provide an (ε, δ)-DP sketching mechanism based on the Gaussian mechanism.

Definition 8.9: The Gaussian noisy sum of features $\boldsymbol{\Sigma}_{\mathcal{G}}(\mathbf{X})$ of a dataset $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ with noise parameters σ is the random variable

$$\boldsymbol{\Sigma}_{\mathcal{G}}(\mathbf{X}) = \boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi}$$

where $\boldsymbol{\Sigma}(\mathbf{X}) \triangleq \sum_{i=1}^n \Phi(\mathbf{x}_i)$ and

$$\forall j \in [\![1, m]\!],\ \xi_j \overset{\text{iid}}{\sim} \begin{cases} \mathcal{N}^{\mathbb{C}}(0, \sigma^2) \text{ if } \Phi \text{ is complex-valued} \\ \mathcal{N}(0, \sigma^2) \text{ if } \Phi \text{ is real-valued} \end{cases}.$$

The only difference with Definition 8.1 is that the noise added on the sum of features $\boldsymbol{\Sigma}(\mathbf{X})$ is Gaussian.

We now introduce an equivalent of the composition lemma 8.2 for the Gaussian case.

Lemma 8.10

For any privacy parameter $\varepsilon > 0$ and choice of $\varepsilon_1, \varepsilon_2 > 0$ such that $\varepsilon_1 + \varepsilon_2 = \varepsilon$, if $\boldsymbol{\Sigma}_{\mathcal{G}}$ has finite l_2 sensitivity $\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma})$, then any mechanism to estimate $\tilde{\mathbf{s}}$ using $\boldsymbol{\Sigma}_{\mathcal{G}}(\mathbf{X})$ with noise level[8] $\sigma = \alpha(\varepsilon_1, \delta) \cdot \Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma})/\sqrt{2\varepsilon_1}$, and $|\mathbf{X}| + \zeta$ where $\zeta \sim \mathcal{L}(\varepsilon_2^{-1})$, is ε-UDP.

8. Here α refers to Theorem 7.14.

Proof: The Gaussian mechanism applied on $\boldsymbol{\Sigma}$ with $\sigma = \alpha(\varepsilon_1, \delta)\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma})/\sqrt{2\varepsilon_1}$ is (ε_1, δ)-UDP according to Theorem 7.14. As in lemma 8.2, releasing $|\mathbf{X}| + \zeta$ with $\zeta \sim \mathcal{L}(\varepsilon_2^{-1})$ is $(\varepsilon_2, 0)$-UDP. The result comes from Lemma 7.15 on sequential composition of approximate differential privacy.

Note that we add Laplacian noise on the dataset size; if Gaussian noise was added we would have to split not only ε but also δ between the sum of features and the dataset size. As there is no difference between $\Delta_1^{\mathrm{U}}(|\cdot|)$ and $\Delta_2^{\mathrm{U}}(|\cdot|)$, allocating a part of δ to the denominator would not bring any substantial gain compared to putting all the budget on the numerator.

We now compute the sensitivities $\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RFF}})$ and $\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RQF}})$, defined respectively in Section 8.1.2 and Section 8.1.2. Here again, in case the feature maps are multiplied by a constant factor, the l_2 sensitivity and the noise level σ need to be multiplied by the same factor.

Random Fourier features For random Fourier features, computing the l_2 sensitivity is much more straightforward than the l_1 sensitivity,

as each component of the feature map has a constant modulus. We get the following result.

Lemma 8.11

The function $\mathbf{\Sigma}^{\text{RFF}}$ has sensitivity $\Delta_2^{\text{U}}(\mathbf{\Sigma}^{\text{RFF}}) = \sqrt{m}$ for both quantized and unquantized cases.

Proof: Using the fact that $|\Phi(\mathbf{x})_j| = 1$ for any j and $\mathbf{x}$, we have by Lemma 8.3

$$\Delta_2^{\text{U}}(\mathbf{\Sigma}^{\text{RFF}}) = \sup_{\mathbf{x}\in\mathbb{R}^d} Q_2^{\text{RFF}}(\mathbf{x}) = \sup_{\mathbf{x}\in\mathbb{R}^d} \sqrt{\sum_{j=1}^{m} |\Phi_j(\mathbf{x})|^2} = \sqrt{m}.$$

As expected, the standard deviation of the Gaussian noise is smaller than the standard deviation of the Laplacian noise that one would need to add in order to reach the same privacy level with the Laplace mechanism. Indeed, the l_2 sensitivity only scales with $\sqrt{m}$, where the l_1 sensitivity was scaling linearly with m.

For bounded differential privacy, we have the following result.

Lemma 8.12

The function $\mathbf{\Sigma}^{\text{RFF}}$ computed with nonresonant features has sensitivity $\Delta_2^{\text{B}}(\mathbf{\Sigma}^{\text{RFF}}) = 2\sqrt{m}$ for both quantized and unquantized cases.

The proof of this result can be found in Appendix C.2.

Random quadratic features Here again, we consider datasets whose elements are bounded by 1 in l_2-norm, i.e. $\mathcal{X} = \mathcal{B}_2$.

Lemma 8.13

The function $\mathbf{\Sigma}^{\text{RQF}}$ built using a matrix of frequencies $\mathbf{\Omega} = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m]$, has l_2 sensitivity $\Delta_2^{\text{U}}(\mathbf{\Sigma}^{\text{RQF}}) = S_4(\mathbf{\Omega})$, where

$$S_4(\mathbf{\Omega}) \triangleq \left(\sup_{\mathbf{x}:\|\mathbf{x}\|_2\leq 1} \sum_{j=1}^{m} (\omega_j^T \mathbf{x})^4 \right)^{1/2}. \tag{8.7}$$

Proof: We have by Lemma 8.3

$$\begin{aligned}\Delta_2^{\text{U}}(\mathbf{\Sigma}^{\text{RQF}}) &= \sup_{\mathbf{x}\in\mathcal{B}_2} Q_2^{\text{RQF}}(\mathbf{x}) = \sup_{\mathbf{x}\in\mathcal{B}_2} \|\Phi^{\text{RQF}}(\mathbf{x})\|_2 \\ &= \left(\sup_{\mathbf{x}:\|\mathbf{x}\|_2\leq 1} \sum_{j=1}^{m} (\omega_j^T \mathbf{x})^4 \right)^{1/2} = S_4(\mathbf{\Omega}).\end{aligned}$$

We now discuss how the quantity $S_4(\mathbf{\Omega})$ can be estimated numerically, as this is not straightforward.

8.1.3 Computation of the bounds for quadratic features

According to the bounds from lemmas 8.8 and 8.13, running our algorithm for differentially private compressive PCA requires to compute

somehow the quantity $\lambda_{\max}(\mathbf{\Omega}\mathbf{\Omega}^T)$ (Laplacian case) or $S^4(\mathbf{\Omega})$ (Gaussian case), where $\lambda_{\max}$ denotes the largest eigenvalue function. When $\mathbf{\Omega}$ is a union of orthonormal bases, one can directly[9] use the closed form given in (8.6) for the Laplacian setting.

In the general case, computing $\lambda_{\max}(\mathbf{\Omega}\mathbf{\Omega}^T)$ can be done numerically[10] without building the $d \times d$ matrix, which is likely to be too expensive in terms of memory requirements if one relies on a sketching method in the first place.

Computing $S_4(\mathbf{\Omega})$, however, is much more tricky. We recall that

$$S_4(\mathbf{\Omega}) \triangleq \left(\sup_{\mathbf{x}:\|\mathbf{x}\|_2 \leq 1} \sum_{j=1}^{m} (\omega_j^T \mathbf{x})^4 \right)^{1/2}, \tag{8.8}$$

which amount to *maximizing* a convex function on the unit ball. Note that this supremum is reached[11], and can be reached on the unit sphere S^{d-1} by homogeneity. Computing (8.8) can be seen either as the estimation of an operator norm, or as a tensor approximation problem.

Rewriting S_4 as an operator norm is straightforward, as indeed we have

$$S_4(\mathbf{\Omega})^2 = \sup_{\mathbf{x}:\|\mathbf{x}\|_2 \leq 1} \sum_{j=1}^{m} (\omega_j^T \mathbf{x})^4 = \sup_{\mathbf{x}:\|\mathbf{x}\|_2 \leq 1} \|\mathbf{\Omega}^T \mathbf{x}\|_4^4 = \|\mathbf{\Omega}^T\|_{2\to 4}^4,$$

using the notation $\|\mathbf{A}\|_{p\to q} \triangleq \sup_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{A}\mathbf{x}\|_q / \|\mathbf{x}\|_p$ for the $l_p \to l_q$ operator norm.

Concentration results can be used to get a probabilistic upper-bound on S_4 (see for instance [302, Proposition 3.2] for a bound, which however only applies to square matrices). However, using a probabilistic bound would weaken our privacy guarantees, as the overall mechanism would only satisfy probabilistic differential privacy[12].

Approximation of $\|\cdot\|_{p\to q}$ norms when $p < q$, also known as hypercontractive norms, is known to be NP-hard in the general case [304], and in particular when $p = 2, q = 4$ [305]. Steinberg derived [306] upper-bounds for the $\|\cdot\|_{p\to q}$ operator norm when $1 \leq q \leq 2 \leq p \leq \infty$, and then proposed to interpolate these results for any value of p, q (including our case of interest $p = 2, q = 4$). However, in our setting this bound translates to the relation

$$\|\mathbf{\Omega}^T\|_{2\to 4}^4 \leq \|\mathbf{\Omega}^T\|_{2\to 2}^2 \|\mathbf{\Omega}^T\|_{2\to\infty}^2 = \|\mathbf{\Omega}^T\|_{2\to 2}^2 \sup_{1\leq j\leq m} \|\boldsymbol{\omega}_j\|_2^2, \tag{8.9}$$

which is (more directly) a consequence of the Riesz-Thorin theorem[13]. Barak et al. also proposed [305] a relaxation of the problem which provides an upper bound on the $\|\cdot\|_{2\to 4}$ operator norm[14], and enjoys good properties for random matrices in some specific settings.

The computation of (8.8) can also be considered as a low-rank tensor approximation problem. We denote $\otimes$ the outer product, and $\mathbf{x}^{\otimes 4} \triangleq \mathbf{x} \otimes \mathbf{x} \otimes \mathbf{x} \otimes \mathbf{x}$ for any $\mathbf{x} \in \mathbb{R}^d$ (which is a tensor of order 4). Defining $\mathcal{C} = \sum_{j=1}^{m} \boldsymbol{\omega}_j^{\otimes 4}$, we can rewrite

$$S_4(\mathbf{\Omega})^2 = \sup_{\mathbf{x}\in S^{d-1}} \left\langle \mathbf{x}^{\otimes 4}, \mathcal{C} \right\rangle_F \tag{8.10}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner-product[15]. Naturally, for any

[9] Note that the zero-padding operation performed when d is not a power of 2 (cf. Chapter 5) does not alter the bound.

[10] Using e.g. the power method on the matrix $\mathbf{\Omega}$.

[11] The function is continuous, takes finite values on the unit ball, and the unit ball is closed.

[12] "Probabilistic" differential privacy refers to the case where ε-DP holds with probability $1 - \delta$ for some ε, δ. It is known to be weaker than approximate (ε, δ)-DP [303].

[13] In our finite-dimensional context, the theorem states that for any r, p, q, θ satisfying $1/r = \theta/p + (1-\theta)/q$, we have $\|A\|_r \leq \|A\|_p^{\theta} \|A\|_q^{1-\theta}$. This is a generalization of the Hölder theorem.

[14] However, their optimization problem is formulated in a slightly too abstruse manner, which makes it difficult to implement the method in practice.

[15] Here for tensors of order 4, i.e.

$$\langle \mathcal{S}, \mathcal{T} \rangle_F \triangleq \sum_{\substack{1\leq i\leq d_1 \\ 1\leq j\leq d_2 \\ 1\leq k\leq d_3 \\ 1\leq l\leq d_4}} \mathcal{S}_{ijkl} \mathcal{T}_{ijkl}$$

for any $\mathcal{S}, \mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3 \times d_4}$.

$\mathbf{x} \in S^{d-1}$ we have $\|\mathbf{x}^{\otimes 4}\|_F = 1$ and thus

$$\|\mathbf{x}^{\otimes 4} - \mathcal{C}\|_F^2 = 1 + \|\mathcal{C}\|_F^2 - 2\langle \mathbf{x}^{\otimes 4}, \mathcal{C}\rangle_F, \tag{8.11}$$

hence finding an upper-bound on (8.10) is equivalent to finding a lower-bound on

$$\inf_{\mathbf{x}:\|\mathbf{x}\|_2 \leq 1} \|\mathbf{x}^{\otimes 4} - \mathcal{C}\|_F. \tag{8.12}$$

Our problem can therefore be seen as a constrained rank-one symmetric tensor approximation problem. Also, for any $\mathbf{x} \in S^{d-1}$, we have

$$\begin{aligned}
\inf_{\lambda \in \mathbb{R}} \|\lambda \mathbf{x}^{\otimes 4} - \mathcal{C}\|_F^2 &\overset{\text{(i)}}{=} \|\langle \mathbf{x}^{\otimes 4}, \mathcal{C}\rangle_F \mathbf{x}^{\otimes 4} - \mathcal{C}\|_F^2 \\
&= \|\mathcal{C}\|_F^2 - 2\langle \mathbf{x}^{\otimes 4}, \mathcal{C}\rangle_F^2 + \langle \mathbf{x}^{\otimes 4}, \mathcal{C}\rangle_F^2 \|\mathbf{x}^{\otimes 4}\|_F^2 \\
&\overset{\text{(ii)}}{=} \|\mathcal{C}\|_F^2 - \langle \mathbf{x}^{\otimes 4}, \mathcal{C}\rangle_F^2.
\end{aligned}$$

(i) By a simple derivation.

(ii) Using $\|\mathbf{x}^{\otimes 4}\|_F = \|\mathbf{x}\|_2^4 = 1$.

As a consequence

$$S_4(\mathbf{\Omega})^2 = \|\mathcal{C}\|_F^2 - \inf_{\lambda \in \mathbb{R}, \mathbf{x} \in S^{d-1}} \|\lambda \mathbf{x}^{\otimes 4} - \mathcal{C}\|_F^2.$$

Moreover, because $\mathcal{C}$ itself is a symmetric tensor, we have

$$\inf_{\mathbf{z} \in S^{d-1}} \inf_{\lambda \in \mathbb{R}} \|\lambda \mathbf{z}^{\otimes 4} - \mathcal{C}\|_F^2 \overset{\text{(iii)}}{=} \inf_{\lambda \in \mathbb{R}, \mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z} \in S^{d-1}} \|\lambda \mathbf{w} \otimes \mathbf{x} \otimes \mathbf{y} \otimes \mathbf{z} - \mathcal{C}\|_F^2. \tag{8.13}$$

(iii) See [307, Theorem 2.1] or [308].

Hence any lower bound B on the (unconstrained) rank-1 approximation problem (8.13) translates to the upper bound $(\|\mathcal{C}\|_F^2 - B)^{1/2}$ for $S_4(\mathbf{\Omega})$ itself.

Various algorithms exist to address (8.12) and (8.13) (most often producing local optimas), such as the high order value decomposition (HOSVD) and truncated variants [309, 310], higher-order power methods (HOPM) [311], alternating least squares (ALS) techniques [312], and quasi-Newton methods [313]. Some of these methods have variants for the symmetric setting, i.e. when the tensor to approximate is itself symmetric [314, 315, 316], which is the case here.

The HOSVD comes with approximation guarantees[16] [310], and can thus be used to derive an upper bound on $S_4(\mathbf{\Omega})$. It however requires to compute the SVD of unfolded representations of the tensor, which can be expensive (in space) depending on the considered application.

The methods for which no guarantees are known can still help to derive lower-bounds on the sensitivity, and can be useful when used for instance in combination with the upper bounds provided by (8.9).

[16] i.e. we can derive a rank-1 approximation $\tilde{\mathcal{C}}$ of $\mathcal{C}$ satisfying for the order-4 setting: $\|\mathcal{C} - \tilde{\mathcal{C}}\|_F \leq 2 \inf_{\mathbf{y} \in \mathbb{R}^d} \|\mathbf{y} \otimes \mathbf{y} \otimes \mathbf{y} \otimes \mathbf{y} - \mathcal{C}\|$ We refer the reader to [317, Theorem 10.2] (in the general case, the factor 2 is replaced by the square root of the tensors' order).

8.2 A FASTER MECHANISM WITH FREQUENCY SUBSAMPLING

We now introduce a sketching mechanism that subsamples the features, and then build on top of it a noisy sketch that guarantees ε-differential privacy. We will derive sharp guarantees in terms of pure differential privacy only, and give a generic upper bound that applies to approximate differential privacy.

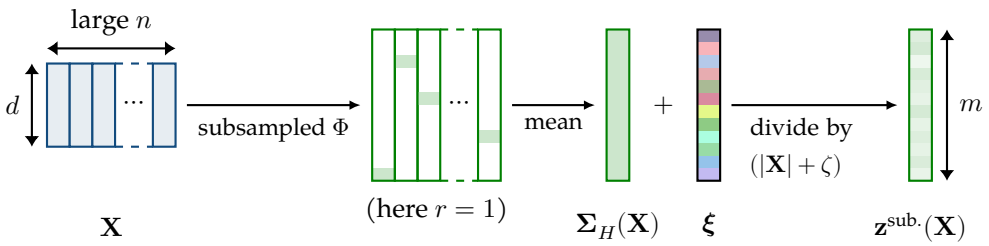


Figure 8.1: Overview of the sketching mechanism with subsampling.

This subsampling mechanism, which consists in computing only some of the m entries of $\Phi(\mathbf{x}_i)$ for each data sample $\mathbf{x}_i$ as shown in Figure 8.1, is introduced in order to reduce the computational complexity of the sketching operation. This complexity is dominated by the computation of all the $(\boldsymbol{\omega}_j^T \mathbf{x}_i)_{1 \leq i \leq n, 1 \leq j \leq m}$, i.e. by the matrix product $\mathbf{\Omega}^T \mathbf{X}$, which costs $\Theta(mdn)$ when using a dense matrix $\mathbf{\Omega}$. As shown below in Lemma 8.17, subsampling does not bring any advantage in enforcing differential privacy, i.e. the noise level required to get privacy is at least the same as without subsampling.We will prove however that in some settings, the guarantees obtained with and without subsampling are exactly the same. Moreover, it will be shown in the next chapter[17] that subsampling can be performed for large collections without damaging the utility of the sketch, which motivates our approach.

[17] See Section 9.2.2

Subsampling does also reduce the amount of information that is released about each sample, although this has no impact on differential privacy guarantees. Indeed, in the extreme case we will measure only one floating point[18] number per data sample. When using quantized sketch, this is further reduced to one bit (or two) of information per sample. For instance, if we only have the quantized random Fourier measurement of a sample $\mathbf{x}$ associated to the frequency $\boldsymbol{\omega}_j$, we can can only infer that $\mathbf{x}$ belongs to a union of "slices" of the ambient space delimited by affine hyperplanes orthogonal to $\boldsymbol{\omega}_j$. However in practice these features are further averaged over the samples[19], the subsampling is performed randomly – so that we don't know which entry of the sketch a given sample contributed to –, and in the differential privacy scenario noise can still be added to the obtained sketch.

[18] Or one complex number.

[19] Such individual sketches are computed by the data holder but not released publicly

Hence, although we solely focus on differential privacy in this thesis, we expect that this variant of the framework would be beneficial when working with alternative privacy definitions that rely on average information-theoretic quantities, such as mutual information [318].

Subsampling schemes We define $\mathcal{Q} \triangleq \{0,1\}^m$ the set of binary masks $\mathbf{q}$ and $\mathcal{Q}^n$ the set of all possible tuples $(\mathbf{q}_1, \dots, \mathbf{q}_n)$ of n such masks. Pointwise multiplication is denoted $\odot$. In the following, we consider an integer $r \geq 1$ and denote $\mathcal{P}_r$ the set of probability distributions $p_{\mathbf{q}}$ on $\mathcal{Q}$ satisfying $\forall j \in [\![1, m]\!]\, \mathbf{E}_{\mathbf{q} \sim p_{\mathbf{q}}} \mathbf{q}_j = r/m$. Particular examples of probability distributions belonging to $\mathcal{P}_r$ include

- **Uniform:** the uniform distribution over the set

$$\mathcal{Q}_r \triangleq \{\mathbf{q} \in \mathcal{Q} | \sum_{i=1}^{m} h_i = r\};$$

- **Bernoulli:** the distribution $(\text{Bern}\,(r/m))^m$, corresponding to the masks whose m entries are drawn i.i.d. according to a Bernoulli distribution with parameter r/m;
- **Block-uniform:** when m/d is an integer and r a multiple of d, the distribution $\mathcal{U}(\mathcal{Q}_r^{\text{struct.}})$ where $\mathcal{Q}_r^{\text{struct.}}$ is the subset of $\mathcal{Q}_r$ containing only the vectors which are structured by blocs of size d, i.e. $\mathcal{Q}_r^{\text{struct.}} \triangleq \{\mathbf{q} = [h_1, ..., h_m] \in \mathcal{Q}_r | \forall i \in [\![1, m/d]\!], h_{(i-1)d+1} = h_{(i-1)d+2} = ... = h_{id}\}$. This scheme will be useful when $\mathbf{\Omega}$ is a structured transform, as explain in the next paragraph.

A note on the sketching complexity When computing r features per input sample rather than computing the whole matrix product $\mathbf{\Omega}^T X$, the sketching complexity[20] goes down from $\Theta(mdn)$ to $\Theta(rdn)$. In the high-dimensional setting, we have seen in Chapter 5 that structured matrices $\mathbf{\Omega}$ made of $\lceil m/d \rceil$ square $d \times d$ blocks could be used to speed-up computations[21]. In that case, the matrix-vector multiplication for each square block is performed at once using the corresponding fast transform with complexity $\Theta(d \log(d))$. We can thus rely on block-uniform subsampling mechanism introduced above using $r = d$, so that for each data sample $\mathbf{x}_i$ we compute the d measurements associated to a randomly chosen block. The sketching cost is then $\Theta(d \log(d) n)$, while computing the same number $r = d$ of measurements with a dense matrix $\mathbf{\Omega}$ would have scaled in $\Theta(d^2 n)$.

[20] This is of course only a theoretical complexity. One might expect this subsampling strategy to be slightly slower than subsampling the samples for a same theoretical complexity due to the various caching mechanisms at play and I/O operations performed.

[21] We recall that such constructions require a dimension which is a power of 2, otherwise padding must be used. We omit these details here.

Sketching with subsampling We first define how features are subsampled using a fixed tuple of masks, and then define the sketching mechanism using random masks.

Definition 8.14: The sum of subsampled features of a dataset $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, using a fixed set of binary masks $\mathbf{Q} = (\mathbf{q}_1, ..., \mathbf{q}_n) \in \mathcal{Q}^n$ drawn[22] according to some distribution in $\mathcal{P}_r$ is defined as

$$\mathbf{\Sigma}_{\mathbf{Q}}(\mathbf{X}) \triangleq \frac{m}{r} \sum_{i=1}^{n} \Phi(\mathbf{x}_i) \odot \mathbf{q}_i.$$

[22] The mechanism is deterministic and can naturally be defined for any set of masks, but we will always make this assumption.

The constant m/r in Definition 8.14 is used to ensure that we always have $\mathbf{E}_{\mathbf{Q}} \frac{1}{|\mathbf{X}|} \mathbf{\Sigma}_{\mathbf{Q}}(\mathbf{X}) = \tilde{\mathbf{s}}$ when $\mathbf{Q}$ is drawn according to $p_{\mathbf{q}}^n$ for some $p_{\mathbf{q}} \in \mathcal{P}_r$. We now introduce the whole mechanism, where the masks themselves are drawn randomly.

Definition 8.15: The Laplacian subsampled sum of features $\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X})$ of a dataset $\mathbf{X} \in \mathcal{D}_n$ using a mask distribution $p_{\mathbf{q}} \in \mathcal{P}_r$ and a noise parameter b is the random vector

$$\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X}) = \mathbf{\Sigma}_{\mathbf{Q}}(\mathbf{X}) + \boldsymbol{\xi}, \tag{8.14}$$

where

- $\forall j \in [\![1, m]\!]$, $\xi_j \overset{\text{iid}}{\sim} \begin{cases} \mathcal{L}^{\mathbb{C}}(b) & \text{if } \Phi \text{ is complex-valued} \\ \mathcal{L}(b) & \text{if } \Phi \text{ is real-valued} \end{cases}$
- $\mathbf{Q} = (\mathbf{q}_1, ..., \mathbf{q}_n)$ with $\mathbf{q}_i \overset{\text{iid}}{\sim} p_{\mathbf{q}}$.

For a deterministic set of masks $\mathbf{Q}$, we denote $\mathbf{\Sigma}_{\mathcal{L},\mathbf{Q}}(\mathbf{X}) = \mathbf{\Sigma}_{\mathbf{Q}}(\mathbf{X}) + \boldsymbol{\xi}$ the sum that is randomized only on $\boldsymbol{\xi}$. Compared to the deterministic sum of features $\mathbf{\Sigma}$, the subsampled sum of features $\overline{\mathbf{\Sigma}}_{\mathcal{L}}$ thus picks randomly some values from each feature vector $\Phi(\mathbf{x}_i)$ according to a random mask $\mathbf{q}_i$ and then adds Laplacian noise $\boldsymbol{\xi}$ on the sum of these contributions. Note that in this mechanism, both[23] $\boldsymbol{\xi}$ and $\mathbf{Q}$ are random quantities.

[23] By opposition to $\mathbf{\Sigma}_{\mathcal{L},\mathbf{Q}}$.

In order to formulate our results, we define a quantity Q_1^{U} which is similar to the quantity from Lemma 8.3 but takes into account a mask $\mathbf{q} \in \mathcal{Q}$. Although we only consider Q_1^{U} for the moment, we also introduce the quantities $Q_1^{\mathrm{B}}, Q_2^{\mathrm{U}}, Q_2^{\mathrm{B}}$ which will be used in Section 8.2.3 for generalizing some results to the BDP and/or approximate DP settings. For a real-valued feature map and $p \in \{1, 2\}$, we define

$$Q_p^{\mathrm{U}}(\mathbf{x}, \mathbf{q}) \triangleq \frac{m}{r} \|\Phi(\mathbf{x}) \odot \mathbf{q}\|_p \tag{8.15}$$

$$Q_p^{\mathrm{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) \triangleq \frac{m}{r} \|(\Phi(\mathbf{x}) - \Phi(\mathbf{y})) \odot \mathbf{q}\|_p. \tag{8.16}$$

The definition extends to complex-valued feature maps using the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$, but using the same mask $\mathbf{q}$ for both real and imaginary parts.

Similarly to Section 8.1 where the privacy was directly driven by the quantity $\Delta_1^{\mathrm{U}}(\mathbf{\Sigma})$, itself equal to $\sup_{\mathbf{x}\in\mathcal{X}} Q_1^{\mathrm{U}}(\mathbf{x})$, the following lemma gives a generalization taking the masks into account.

Lemma 8.16

The subsampled sum $\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X})$ from Definition 8.15 with noise level b is UDP with optimal privacy parameter ε^*, defined as

$$\exp(\varepsilon^*) = \sup_{\mathbf{x}\in\mathcal{X}} \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b} Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q})\right).$$

See Proof C.6 in appendix. Note that although we assumed for simplicity in Definition 8.15 the mask $\mathbf{Q}$ to be defined as $\mathbf{Q} = (\mathbf{q}_1, ..., \mathbf{q}_n)$ with i.i.d. $(\mathbf{q}_i)_{1\leq i\leq n}$, we actually only need the distribution of $\mathbf{Q}$ to be exchangeable, i.e. to be invariant by permutation of the samples.

Lemma 8.16 allows us to show that subsampling cannot improve differential privacy guarantees.

Lemma 8.17

If the subsampled sum $\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X})$ is ε-UDP, then the noisy sum $\mathbf{\Sigma}_{\mathcal{L}}(\mathbf{X})$ computed with the same feature map and the same noise parameter (but without subsampling) is ε-UDP as well.

Before proving Lemma 8.17, let us just mention that for specific feature maps discussed later, the subsampled sum $\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X})$ is in fact just as differentially private as the sum $\mathbf{\Sigma}_{\mathcal{L}}(\mathbf{X})$ computed without subsampling, while offering flexible tradeoffs between computational complexity and utility.

Proof: Recall the definitions of $Q_1^{\mathrm{U}}(\mathbf{x})$ and $Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q})$ given respectively in Lemma 8.3 and Equation (8.15). Using Jensen's inequality and the fact that the masks are drawn according to some $p_{\mathbf{q}} \in \mathcal{P}_r$, we have for any $\mathbf{x}$ the lower bound

$$\mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b} Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q})\right) \geq \exp\left(\frac{1}{b} \mathbf{E}_{\mathbf{q}} Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q})\right) = \exp\left(\frac{1}{b} Q_1^{\mathrm{U}}(\mathbf{x})\right)$$

According to Lemma 8.16, taking the supremum on $\mathbf{x}$, we get

$$\begin{aligned}
\sup_{\mathbf{X}, \mathbf{Y} \in \mathcal{D} : \mathbf{X} \overset{\mathrm{U}}{\sim} \mathbf{Y}} \sup_{\mathbf{z} \in \mathcal{Z}} \frac{p_{\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X})}(\mathbf{z})}{p_{\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{Y})}(\mathbf{z})} &= \sup_{\mathbf{x} \in \mathcal{X}} \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b} Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q})\right) \\
&\geq \sup_{\mathbf{x} \in \mathcal{X}} \exp\left(\frac{1}{b} Q_1^{\mathrm{U}}(\mathbf{x})\right) \\
&= \exp\left(\frac{1}{b} \sup_{\mathbf{x} \in \mathcal{X}} Q_1^{\mathrm{U}}(\mathbf{x})\right) = \exp\left(\frac{1}{b} \Delta_1^{\mathrm{U}}(\mathbf{\Sigma})\right)
\end{aligned}$$

where the last equality comes from Lemma 8.3. If $\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X})$ is ε-DP, we thus have

$$\exp(\frac{1}{b} \Delta_1^{\mathrm{U}}(\mathbf{\Sigma})) \leq \sup_{\mathbf{X}, \mathbf{Y} \in \mathcal{D} : \mathbf{X} \overset{\mathrm{U}}{\sim} \mathbf{Y}} \sup_{\mathbf{z} \in \mathcal{Z}} \frac{p_{\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{X})}(\mathbf{z})}{p_{\overline{\mathbf{\Sigma}}_{\mathcal{L}}(\mathbf{Y})}(\mathbf{z})} \leq \exp(\varepsilon)$$

which means $b \geq \Delta_1^{\mathrm{U}}(\mathbf{\Sigma})/\varepsilon$, hence by Theorem 7.10, $\mathbf{\Sigma}_{\mathcal{L}}(\mathbf{X})$ is ε-DP.

In the following, we denote $\overline{\mathbf{\Sigma}}_{\mathbf{Q}}^{\mathrm{RFF}}$ and $\overline{\mathbf{\Sigma}}_{\mathbf{Q}}^{\mathrm{RQF}}$ the sums of subsampled features when using respectively $\Phi = \Phi^{\mathrm{RFF}}$ or $\Phi = \Phi^{\mathrm{RQF}}$ as a feature map. We now provide specific results for these two feature maps.

8.2.1 Random Fourier Features

The following lemma generalizes the notion of sensitivity to the subsampled case. We include the BDP case which will be used in Section 8.2.3.

Lemma 8.18

Consider Φ^{RFF} built using nonresonant frequencies, and $r \in [\![1, m]\!]$. For each $\mathbf{q} \in \mathcal{Q}_r$ we have

$$\begin{aligned}
\sup_{\mathbf{x} \in \mathbb{R}^d} Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q}) &= \sup_{\mathbf{x} \in \mathbb{R}^d} \inf_{\mathbf{q}' \in \mathcal{Q}_r} Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q}') = \sqrt{2} m. \\
\sup_{\mathbf{x}, \mathbf{y} \in \mathbb{R}^d} Q_1^{\mathrm{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) &= \sup_{\mathbf{x}, \mathbf{y} \in \mathbb{R}^d} \inf_{\mathbf{q}' \in \mathcal{Q}_r} Q_1^{\mathrm{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}') = 2\sqrt{2} m.
\end{aligned}$$

Moreover $Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q}) \leq \sqrt{2} m$ and $Q_1^{\mathrm{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) \leq 2\sqrt{2} m$ always hold, even for resonant frequencies.

The proof is quite similar to the proof of Lemma 8.6, and can be found in Appendix C.3. We can now state the main result for random Fourier features.

Lemma 8.19

Consider $r \in [\![1, m]\!]$ and a probability distribution $p_{\mathbf{q}} \in \mathcal{P}_r$ such that $\mathbf{q} \in \mathcal{Q}_r$ almost surely. Then for any $\varepsilon > 0$, $\overline{\mathbf{\Sigma}}^{\text{RFF}}_{\mathcal{L}}(\mathbf{X})$ from Definition 8.15 with noise level $b = \sqrt{2}m/\varepsilon$ and mask distribution $p_{\mathbf{q}}$ is ε-UDP. The bound is sharp if Φ^{RFF} is built using nonresonant frequencies.

Proof: By Lemma 8.16 and Lemma 8.18 we have

$$\begin{aligned}\sup_{\mathbf{X},\mathbf{Y}\in\mathcal{D}:\mathbf{X}\overset{\text{U}}{\sim}\mathbf{Y}} \sup_{\mathbf{z}\in\mathcal{Z}} \frac{p_{\overline{\mathbf{\Sigma}}^{\text{RFF}}_{\mathcal{L}}(\mathbf{X})}(\mathbf{z})}{p_{\overline{\mathbf{\Sigma}}^{\text{RFF}}_{\mathcal{L}}(\mathbf{Y})}(\mathbf{z})} &= \sup_{\mathbf{x}\in\mathcal{X}} \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b} Q_1^{\text{U}}(\mathbf{x}, \mathbf{q})\right) \\ &= \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b} \sup_{\mathbf{x}\in\mathcal{X}} Q_1^{\text{U}}(\mathbf{x}, \mathbf{q})\right) \\ &= \exp\left(\frac{1}{b} m\sqrt{2}\right) = \exp(\varepsilon).\end{aligned}$$

The second and third equalities are consequences of Lemma 8.18, and hold because $\mathbf{q}$ belongs to $\mathcal{Q}_r$ almost surely.

8.2.2 Compressive principal component analysis

We recall that for PCA $\mathcal{X} = \mathcal{B}_2$. We give a generic upper bound in Lemma 8.21, and below in Lemma 8.23 a sharp bound when $\mathbf{\Omega}$ is a union of orthonormal bases. We first provide a simple lemma used in both results. For any mask $\mathbf{q} \in \mathcal{Q}_r$ with r non-zero entries at indexes $i_1, ..., i_r$, and any matrix of frequencies $\mathbf{\Omega} = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m] \in \mathbb{R}^{d\times m}$, we denote $\mathbf{\Omega}_{\mathbf{q}} = [\boldsymbol{\omega}_{i_1}, ..., \boldsymbol{\omega}_{i_r}]$ the matrix obtained from $\mathbf{\Omega}$ by keeping only the columns corresponding to nonzero indexes of $\mathbf{q}$.

Lemma 8.20

Consider the functions $Q_1^{\text{U}}, Q_1^{\text{B}}$ associated to the feature map Φ^{RQF}. For each $\mathbf{q} \in \mathcal{Q}$

$$\sup_{\mathbf{x}\in\mathcal{X}} Q_1^{\text{U}}(\mathbf{x}, \mathbf{q}) = \sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} Q_1^{\text{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) = \frac{m}{r}\|\mathbf{\Omega}_{\mathbf{q}}\|_2^2.$$

Proof:

$$\begin{aligned}\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} Q_1^{\text{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) &= \sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \frac{m}{r} \sum_{j\in\text{supp}(\mathbf{q})} |(\boldsymbol{\omega}_j^T x)^2 - (\boldsymbol{\omega}_j^T y)^2| \\ &= \sup_{\mathbf{x}\in\mathcal{X}} \frac{m}{r} \sum_{j\in\text{supp}(\mathbf{q})} (\boldsymbol{\omega}_j^T x)^2 = \sup_{\mathbf{x}\in\mathcal{X}} Q_1^{\text{U}}(\mathbf{x}, \mathbf{q}) \\ &= \sup_{\mathbf{x}\in\mathcal{X}} \frac{m}{r} \mathbf{x}^T \left(\sum_{j\in\text{supp}(\mathbf{q})} \boldsymbol{\omega}_j \boldsymbol{\omega}_j^T\right) \mathbf{x} \\ &= \frac{m}{r} \lambda_{\max}\left(\sum_{j\in\text{supp}(\mathbf{q})} \boldsymbol{\omega}_j \boldsymbol{\omega}_j^T\right) = \frac{m}{r}\|\mathbf{\Omega}_{\mathbf{q}}\|_2^2.\end{aligned}$$

For any $p_{\mathbf{q}} \in \mathcal{P}_r$, we denote $\text{supp}(p_{\mathbf{q}})$ the support of $p_{\mathbf{q}}$, that is the set of possible outcomes of $\mathbf{q} \sim p_{\mathbf{q}}$.

Lemma 8.21

Let $p_{\mathbf{q}} \in \mathcal{P}_r$. For any $\varepsilon > 0$, releasing $\overline{\mathbf{\Sigma}}^{\text{RQF}}_{\mathcal{L}}(\mathbf{X})$ from Definition 8.15 with noise parameter $b = \frac{m}{r\varepsilon} \sup_{\mathbf{q} \in \text{supp}(p_{\mathbf{q}})} \|\mathbf{\Omega}_{\mathbf{q}}\|_2^2$ and mask distribution $p_{\mathbf{q}}$ is ε-UDP.

Proof: By Lemma 8.16, with $\mathcal{X} = \mathcal{B}_2$

$$\begin{aligned}\sup_{\mathbf{X},\mathbf{Y} \in \mathcal{D}:\mathbf{X} \overset{\text{U}}{\approx} \mathbf{Y}} \sup_{\mathbf{s} \in \mathcal{Z}} \frac{p_{\overline{\mathbf{\Sigma}}^{\text{RQF}}_{\mathcal{L}}(\mathbf{X})}(\mathbf{s})}{p_{\overline{\mathbf{\Sigma}}^{\text{RQF}}_{\mathcal{L}}(\mathbf{Y})}(\mathbf{s})} &= \sup_{\mathbf{x} \in \mathcal{X}} \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b} Q_1^{\text{U}}(\mathbf{x}, \mathbf{q})\right) \\ &\leq \sup_{\mathbf{x} \in \mathcal{X}} \sup_{\mathbf{q} \in \text{supp}(p_{\mathbf{q}})} \exp\left(\frac{1}{b} Q_1^{\text{U}}(\mathbf{x}, \mathbf{q})\right) \\ &= \exp\left(\frac{1}{b}\frac{m}{r} \sup_{\mathbf{q} \in \text{supp}(p_{\mathbf{q}})} \|\mathbf{\Omega}_{\mathbf{q}}\|_2^2\right),\end{aligned}$$

by Lemma 8.20, which concludes the proof.

Note that the finer bound $\mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b}\frac{m}{r}\|\mathbf{\Omega}_{\mathbf{q}}\|_2^2\right)$ holds, but does not yield explicit guarantees.

Lemma 8.22

Consider m a multiple of d and $\mathbf{\Omega}$ a concatenation of m/d orthonormal bases as described in Section 8.1.1. Let r be a multiple of d, and $\mathbf{q} \in \mathcal{Q}_r^{\text{struct.}}$. Then for any $\mathbf{x}$ such that $\|\mathbf{x}\|_2 = 1$, we have

$$Q_1^{\text{U}}(\mathbf{x}, \mathbf{q}) = \sup_{\mathbf{x} \in \mathcal{X}} Q_1^{\text{U}}(\mathbf{x}, \mathbf{q}) = \sup_{\mathbf{x}, \mathbf{y} \in \mathcal{X}} Q_1^{\text{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) = \frac{m}{d}.$$

Proof: Let us rewrite $\mathbf{\Omega} = [\mathbf{B}_1, ..., \mathbf{B}_{m/d}]$ where the $(\mathbf{B}_i)_{1 \leq i \leq m/d}$ are $d \times d$ blocs corresponding to orthonormal bases. We have $\mathbf{\Omega}\mathbf{\Omega}^T = \sum_{i=1}^{m/d} \mathbf{B}_i \mathbf{B}_i^T = m/d\, \mathbf{I}_d$. As $\mathbf{q} \in \mathcal{Q}_r^{\text{struct.}}$, we have for the same reason $\mathbf{\Omega}_{\mathbf{q}}\mathbf{\Omega}_{\mathbf{q}}^T = (r/d)\, \mathbf{I}_d$. As a result, for any $\mathbf{x} \in \mathcal{X}$ we have $Q_1^{\text{U}}(\mathbf{x}, \mathbf{q}) = \frac{m}{r}(r/d)\|\mathbf{x}\|_2^2 = (m/d)\|\mathbf{x}\|_2^2$ and the result follows from $\mathcal{X} = \mathcal{B}_2$. Furthermore

$$\sup_{\mathbf{x}, \mathbf{y} \in \mathcal{X}} Q_1^{\text{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) = \sup_{\mathbf{x}, \mathbf{y} \in \mathcal{X}} \|\Phi^{\text{RFF}}(\mathbf{x}) - \Phi^{\text{RFF}}(\mathbf{y})\|_1 \tag{8.17}$$

$$\overset{\text{(i)}}{=} \sup_{\mathbf{x} \in \mathcal{X}} \|\Phi^{\text{RFF}}(\mathbf{x})\|_1 = Q_1^{\text{U}}(\mathbf{x}, \mathbf{q}). \tag{8.18}$$

(i) Given that Φ^{RFF} takes only positive values and vanishes in 0.

Lemma 8.23

Consider m a multiple of d and $\mathbf{\Omega}$ a concatenation of m/d orthonormal bases as described in Section 8.1.1. Let r be a multiple of d, and $p_{\mathbf{q}} = \mathcal{U}(\mathcal{Q}_r^{\text{struct.}})$ be the block-uniform distribution. For any $\varepsilon > 0$, releasing $\overline{\mathbf{\Sigma}}^{\text{RQF}}_{\mathcal{L}}(\mathbf{X})$ with mask distribution $p_{\mathbf{q}}$ and noise parameter $b = m/(d\varepsilon)$ is ε-UDP, and the bound is sharp.

Proof: By Lemma 8.16 and Lemma 8.22, it follows that

$$\begin{aligned}\sup_{\mathbf{X},\mathbf{Y}\in\mathcal{D}:\mathbf{X}\overset{\mathrm{U}}{\sim}\mathbf{Y}}\sup_{\mathbf{z}\in\mathcal{Z}}\frac{p_{\overline{\mathbf{\Sigma}}^{\mathrm{RQF}}_{\mathcal{L}}(\mathbf{X})}(\mathbf{z})}{p_{\overline{\mathbf{\Sigma}}^{\mathrm{RQF}}_{\mathcal{L}}(\mathbf{Y})}(\mathbf{z})} &= \sup_{\mathbf{x}\in\mathcal{X}}\mathbf{E}_{\mathbf{q}}\exp\left(\frac{1}{b}Q_1^{\mathrm{U}}(\mathbf{x},\mathbf{q})\right)\\ &= \sup_{\mathbf{x}\in\mathcal{X}}\exp\left(\frac{1}{b}\frac{m}{d}\|\mathbf{x}\|_2^2\right)\\ &= \exp\left(\frac{m}{db}\right) = \exp(\varepsilon),\end{aligned}$$

where the second equality comes from Lemma 8.22 as any $\mathbf{x}$ for which $\|\mathbf{x}\|^2 = 2$ reaches the supremum for all $\mathbf{q} \in \mathcal{Q}_r^{\text{struct.}}$ simultaneously.

Note that the noise level required to get differential privacy when $\mathbf{\Omega}$ is a union of orthonormal bases is independent of r and is the same as when $r = m$, i.e. without subsampling.

8.2.3 An Upper Bound for Approximate and Bounded Differential Privacy

Similarly to Definition 8.15, we define the Gaussian subsampled sum of features.

Definition 8.24: The Gaussian subsampled sum of features $\overline{\mathbf{\Sigma}}_{\mathcal{G}}(\mathbf{X})$ of a dataset $\mathbf{X} \in \mathcal{D}_n$ using a mask distribution $p_{\mathbf{q}} \in \mathcal{P}_r$ and a noise parameter σ is the random variable

$$\overline{\mathbf{\Sigma}}_{\mathcal{G}}(\mathbf{X}) = \mathbf{\Sigma}_{\mathbf{Q}}(\mathbf{X}) + \boldsymbol{\xi}, \tag{8.19}$$

where

- $\forall j \in [\![1, m]\!]$, $\xi_j \overset{\text{iid}}{\sim} \begin{cases}\mathcal{N}^{\mathbb{C}}(0,\sigma^2) \text{ if } \Phi \text{ is complex-valued}\\ \mathcal{N}(0,\sigma^2) \text{ if } \Phi \text{ is real-valued}\end{cases}$
- $\mathbf{Q} = (\mathbf{q}_1, ..., \mathbf{q}_n)$ with $\mathbf{q}_i \overset{\text{iid}}{\sim} p_{\mathbf{q}}$.

Although we do not have an equivalent of Lemma 8.16 for approximate DP, we provide in Lemma 8.25 a generic upper bound, which holds for both pure and approximate DP, bounded and unbounded DP. In order to do so, we introduce the following definitions for $p \in \{1, 2\}$ and $\mathbf{q} \in \mathcal{Q}$

$$\begin{aligned}Q_p^{\mathrm{U}}(\mathbf{q}) &\triangleq \sup_{\mathbf{x}\in\mathcal{X}} Q_p^{\mathrm{U}}(\mathbf{x},\mathbf{q})\\ Q_p^{\mathrm{B}}(\mathbf{q}) &\triangleq \sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} Q_p^{\mathrm{U}}(\mathbf{x},\mathbf{y},\mathbf{q}).\end{aligned}$$

Lemma 8.25

Let $p_{\mathbf{q}} \in \mathcal{P}_r$ be a mask distribution.

- For any $\varepsilon > 0$, the mechanism $\overline{\mathbf{\Sigma}}_{\mathcal{L}}$ from Definition 8.15 with mask distribution $p_{\mathbf{q}}$ and noise level $b \geq \max_{\mathbf{q}\in\text{supp}(p_{\mathbf{q}})} Q_1(\mathbf{q})/\varepsilon$ is ε-DP.
- For any $\varepsilon, \delta > 0$, the mechanism $\overline{\mathbf{\Sigma}}_{\mathcal{G}}$ from Defini-

tion 8.24 with mask distribution $p_{\mathbf{q}}$ and noise level $\sigma \geq \alpha(\varepsilon,\delta)\max_{\mathbf{q}\in\mathrm{supp}(p_{\mathbf{q}})} Q_2(\mathbf{q})/(2\varepsilon)^{1/2}$ (where $\alpha(\varepsilon,\delta)$ refers to Theorem 7.14) is (ε,δ)-DP.

These hold for both BDP and UDP, with $Q_p(\mathbf{q})$ defined accordingly as $Q_p^{\mathrm{B}}(\mathbf{q})$ or $Q_p^{\mathrm{U}}(\mathbf{q})$.

Proof: Let $\varepsilon > 0$, $R \in \{\overline{\mathbf{\Sigma}}_{\mathcal{L}}, \overline{\mathbf{\Sigma}}_{\mathcal{G}}\}$ be one of the two random mechanisms, and $R_{\mathbf{Q}}$ for any $\mathbf{Q}$ be the associated mechanism that uses the fixed masks $\mathbf{Q}$ but is randomized on $\boldsymbol{\xi}$. Let $\sim \in \{\overset{\mathrm{U}}{\sim}, \overset{\mathrm{B}}{\sim}\}$ denote the considered neighborhood relation, and δ be such that $\delta = 0$ for pure DP, $\delta > 0$ for approximate DP. We need to show that

$$\forall \mathbf{X} \sim \mathbf{Y} \in \mathcal{D}, \mathbf{z} \in \mathcal{Z} : p_{R(\mathbf{X})}(\mathbf{z}) \leq \exp(\varepsilon) p_{R(\mathbf{Y})}(\mathbf{z}) + \delta$$

Fix $n > 0$ and an arbitrary set of masks $\mathbf{Q} = (\mathbf{q}_1, ..., \mathbf{q}_n) \in \mathcal{Q}^n$, and consider the mechanism $\mathbf{\Sigma}_{\mathbf{Q}}$ on $\mathcal{D}' \triangleq \mathcal{D}_n$ (BDP case) or $\mathcal{D}' \triangleq \mathcal{D}_n \cup \mathcal{D}_{n-1}$ (UDP case[24]) given in Definition 8.14. For a neighboring relation $\approx$, let $\Delta_{p,\approx}$ denote the l_p sensitivity computed according to $\approx$. For any ordered neighboring relation $\approx \in \{\overset{\mathrm{U}}{\approx}, \overset{\mathrm{B}}{\approx}\}$, according to Theorem 7.10 for pure DP and Theorem 7.14 for ADP applied on $\mathcal{D}'$ and w.r.t. $\approx$, if the noise level of $\boldsymbol{\xi}$ in $R_{\mathbf{Q}}$ is chosen as $b \geq b^*_{\mathbf{Q}} \triangleq \Delta_{1,\approx}(\mathbf{\Sigma}_{\mathbf{Q}})/\varepsilon$ or $\sigma \geq \sigma^*_{\mathbf{Q}} \triangleq \alpha(\varepsilon,\delta)\Delta_{2,\approx}(\mathbf{\Sigma}_{\mathbf{Q}})/(2\varepsilon)^{1/2}$, then we have for any $\mathbf{X}, \mathbf{Y} \in \mathcal{D}'$ such that $\mathbf{X} \approx \mathbf{Y}$

$$\forall \mathbf{z} \in \mathcal{Z} : p_{R_{\mathbf{Q}}(\mathbf{X})}(\mathbf{z}) \leq \exp(\varepsilon) p_{R_{\mathbf{Q}}(\mathbf{Y})}(\mathbf{z}) + \delta, \tag{8.20}$$

Note that the sensitivities depend on the neighboring relation used (UDP/BDP), but are always computed for an ordered relation, thus for $p \in \{1, 2\}$, we have $\Delta_{p,\approx}(\mathbf{\Sigma}_{\mathbf{Q}}) = Q_p(\mathbf{q}_n)$ if $\mathbf{Q} = (\mathbf{q}_1, ..., \mathbf{q}_n)$. The result follows by taking the expectation of these inequalities, which hold simultaneously for all $\mathbf{Q}$ provided that

$$b \geq \max_{\mathbf{Q}\in\mathrm{supp}(p_{\mathbf{Q}})} b^*_{\mathbf{Q}} = \max_{\mathbf{q}\in\mathrm{supp}(p_{\mathbf{q}})} Q_1(\mathbf{q})/\varepsilon \tag{8.21}$$

(resp. that

$$\sigma \geq \max_{\mathbf{Q}\in\mathrm{supp}(p_{\mathbf{Q}})} \sigma^*_{\mathbf{Q}} = \alpha(\varepsilon,\delta) \max_{\mathbf{q}\in\mathrm{supp}(p_{\mathbf{q}})} Q_2(\mathbf{q})/(2\varepsilon)^{1/2} \tag{8.22}$$

).

The masks are drawn i.i.d. according to $p_{\mathbf{q}}$, and for any pair $(\mathbf{X}, \mathbf{Q})$ we have $\mathbf{\Sigma}_{\mathbf{Q}}(\sigma(\mathbf{X})) = \mathbf{\Sigma}_{\sigma^{-1}(\mathbf{Q})}(\mathbf{X})$, thus for any dataset $\mathbf{X}$ and permutation $\sigma \in \mathcal{S}_{|\mathbf{X}|}$ we have

$$\begin{aligned} p_{R(\mathbf{X})}(\mathbf{z}) &= \mathbf{E}_{\mathbf{Q}}\left[p_{R_{\mathbf{Q}}(\mathbf{X})}(\mathbf{z})\right] \\ &= \mathbf{E}_{\mathbf{Q}}\left[p_{R_{\sigma^{-1}(\mathbf{Q})}(\mathbf{X})}(\mathbf{z})\right] = \mathbf{E}_{\mathbf{Q}}\left[p_{R_{\mathbf{Q}}(\sigma(\mathbf{X}))}(\mathbf{z})\right] \end{aligned}$$

If $\mathbf{X}$ and $\mathbf{Y}$ are two datasets such that $\mathbf{X} \sim \mathbf{Y}$ (we assume for now $|\mathbf{X}| \geq |\mathbf{Y}|$), then there are two permutations $\sigma_1 \in \mathcal{S}_{|\mathbf{X}|}, \sigma_2 \in$

24. Note that the expression of $\mathbf{\Sigma}_{\mathbf{Q}}(\mathbf{X})$ does not involve the last mask $\mathbf{q}_n$ when $|\mathbf{X}| = n - 1$ in this case.

$\mathcal{S}_{|\mathbf{Y}|}$ such that $\sigma_1(\mathbf{X}) \approx \sigma_2(\mathbf{Y})$ for the related ordered relation (it follows from the definition for BDP, and one can take one permutation to be the identity for UDP).

Thus using the appropriate noise level according to Equation (8.20) we have

$$\begin{aligned}
&\forall \mathbf{X} \sim \mathbf{Y}, \mathbf{z} \in \mathcal{Z} : \mathbf{E}_{\mathbf{Q}}[p_{R_{\mathbf{Q}}(\sigma_1(\mathbf{X}))}(\mathbf{z})] \\
&\qquad\qquad \leq \exp(\varepsilon)\mathbf{E}_{\mathbf{Q}}[p_{R_{\mathbf{Q}}(\sigma_2(\mathbf{Y}))}(\mathbf{z})] + \delta \\
&\text{i.e. } \forall \mathbf{X} \sim \mathbf{Y}, \mathbf{z} \in \mathcal{Z} : p_{R(\sigma_1(\mathbf{X}))}(\mathbf{z}) \leq \exp(\varepsilon) p_{R(\sigma_2(\mathbf{Y}))}(\mathbf{z}) + \delta \\
&\text{i.e. } \forall \mathbf{X} \sim \mathbf{Y}, \mathbf{z} \in \mathcal{Z} : p_{R(\mathbf{X})}(\mathbf{z}) \leq \exp(\varepsilon) p_{R(\mathbf{Y})}(\mathbf{z}) + \delta,
\end{aligned}$$

which is the desired result.

Note that $\overset{\cup}{\approx}$ is not a symmetric relation, but in the UDP case with $|\mathbf{Y}| = |\mathbf{X}| + 1$, we can still find σ_1, σ_2 such that $\sigma_1(\mathbf{Y}) \overset{\cup}{\approx} \sigma_2(\mathbf{X})$. We hence obtain the desired result by deriving an equivalent of Equation (8.20) for the relation $\overset{\cup}{\approx}_s$, defined as $\mathbf{X} \overset{\cup}{\approx}_s \mathbf{Y} \Leftrightarrow \mathbf{Y} \overset{\cup}{\approx} \mathbf{X}$. As for any $\mathbf{Q}$, we have $\Delta_{p,\overset{\cup}{\approx}}(\mathbf{\Sigma}_{\mathbf{Q}}) = \Delta_{p,\overset{\cup}{\approx}_s}(\mathbf{\Sigma}_{\mathbf{Q}})$ on $\mathcal{D}'$, we get the same result.

Whether or nor the bounds from Lemma 8.25 are sharp for certain scenarios is a question left open for future work.

From Lemma 8.25, one can get guarantees for (ε, δ)-DP with the two simple following results.

Lemma 8.26

Let $\mathbf{q} \in \mathcal{Q}_r$. Then for RFF we have $\sup_{\mathbf{x}} Q_2^{\mathrm{U}}(\mathbf{x}, \mathbf{q}) = \frac{m}{\sqrt{r}}$.

Proof:

$$\sup_{\mathbf{x}} Q_2^{\mathrm{U}}(\mathbf{x}, \mathbf{q}) = \sup_{\mathbf{x} \in \mathcal{X}} \frac{m}{r} \|\Phi^{\mathrm{RFF}}(\mathbf{x})\|_2 = \frac{m}{r}\sqrt{r} = \frac{m}{\sqrt{r}}.$$

Lemma 8.27

Let $\mathbf{q} \in \mathcal{Q}_r$. Then for RQF we have $\sup_{\mathbf{x},\mathbf{y} \in \mathcal{X}} Q_2^{\mathrm{B}}(\mathbf{x}, \mathbf{y}, \mathbf{q}) = \sup_{\mathbf{x} \in \mathcal{X}} Q_2^{\mathrm{U}}(\mathbf{x}, \mathbf{q}) = \frac{m}{r} S_4(\mathbf{\Omega}_{\mathbf{q}})$.

Proof:

$$\begin{aligned}
\sup_{\mathbf{x}} Q_2^{\mathrm{U}}(\mathbf{x}, \mathbf{q}) &= \sup_{\mathbf{x} \in \mathcal{X}} \frac{m}{r} \|\Phi^{\mathrm{RQF}}(\mathbf{x})\|_2 \\
&= \sup_{\mathbf{x} \in \mathcal{X}} \frac{m}{r} \Big(\sum_{i \in \mathrm{supp}(\mathbf{q})} (\boldsymbol{\omega}_i^T \mathbf{x})^4 \Big)^{1/2} = \frac{m}{r} S_4(\mathbf{\Omega}_{\mathbf{q}})
\end{aligned}$$

As Φ^{RQF} takes positive values and vanishes in $\mathbf{0}$, which belongs to $\mathcal{X}$, the same bound holds for BDP.

Note that in these two cases, subsampling increases the bounds and might have a negative impact on the utility (for subsequent learning) of the mechanism.

Summary We summarize the results obtained in this paper in the following tables, where $\alpha = \alpha(\varepsilon, \delta)$ refers to Theorem 7.14.

	Pure ε-DP		**Approximate (ε, δ)-DP**	
	$\boldsymbol{\Sigma}_{\mathcal{L}}(\mathbf{X}) = \boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi}$ with $\xi_j \sim \mathcal{L}(b)$, $b \geq b^*$		$\boldsymbol{\Sigma}_{\mathcal{G}}(\mathbf{X}) = \boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi}$ with $\xi_j \sim \mathcal{N}(0, \sigma^2)$, $\sigma \geq \sigma^*$	
	UDP	**BDP**	**UDP**	**BDP**
Generic	Theorem 7.10: $b^* = \Delta_1(\boldsymbol{\Sigma})/\varepsilon$		Theorem 7.14: $\sigma^* = \frac{\alpha}{\sqrt{2\varepsilon}}\Delta_2(\boldsymbol{\Sigma})$	
	$\Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma}) = \sup_{\mathbf{x}} \lVert\Phi(\mathbf{x})\rVert_1$	$\Delta_1^{\mathrm{B}}(\boldsymbol{\Sigma}) = \sup_{\mathbf{x},\mathbf{y}} \lVert\Phi(\mathbf{x}) - \Phi(\mathbf{y})\rVert_1$	$\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma}) = \sup_{\mathbf{x}} \lVert\Phi(\mathbf{x})\rVert_2$	$\Delta_2^{\mathrm{B}}(\boldsymbol{\Sigma}) = \sup_{\mathbf{x},\mathbf{y}} \lVert\Phi(\mathbf{x}) - \Phi(\mathbf{y})\rVert_2$
RFF	Lemma 8.6: $\Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) \leq \sqrt{2}m$	Lemma 8.18:[1] $\Delta_1^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) \leq 2\sqrt{2}m$	Lemma 8.11: $\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) = \sqrt{m}$	Lemma 8.11:[2] $\Delta_2^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) \leq 2\sqrt{m}$
+ $\boldsymbol{\Omega}$ nonresonant	$\Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) = \sqrt{2}m$	$\Delta_1^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) = 2\sqrt{2}m$	$\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) = \sqrt{m}$	Lemma 8.12: $\Delta_2^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RFF}}) = 2\sqrt{m}$
RQF	Lemma 8.8: $\Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RQF}}) = \Delta_1^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RQF}}) = \lVert\boldsymbol{\Omega}\rVert_2^2$		Lemma 8.13: $\Delta_2^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RQF}}) = \Delta_2^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RQF}}) = S_4(\boldsymbol{\Omega})$	
+ $\boldsymbol{\Omega}$ union of orthogonal bases.	Lemma 8.8: $\Delta_1^{\mathrm{U}}(\boldsymbol{\Sigma}^{\mathrm{RQF}}) = \Delta_1^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RQF}}) = m/d$		No particular closed form.	

Table 8.1: Summary of privacy results **without** subsampling (Section 8.1) and for the sum of features only. For each type of privacy guarantee (column) and for each sketch feature function (row), we provide a potentially loose ($\leq$) or sharp ($=$) bound on the relevant sensitivity Δ, which can be plugged into the associated privacy-preserving sum of features mechanism (top row). [1] With $h = \mathbf{1}$, i.e. $\Delta_1(\boldsymbol{\Sigma}^{\mathrm{RFF}}) = \sup_{\mathbf{x}} Q_1^{\mathrm{B}}(\mathbf{x}, \mathbf{1})$ where Q_1^{B} is computed with $r = m$. [2] Using a simple triangle inequality.

	Pure ε-DP		**Approximate (ε, δ)-DP**	
	$\overline{\boldsymbol{\Sigma}}_{\mathcal{L}}(\mathbf{X}) = \boldsymbol{\Sigma}_{\mathbf{Q}}(\mathbf{X}) + \boldsymbol{\xi}$ with $\xi_j \sim \mathcal{L}(b), \mathbf{Q} \sim p_{\mathbf{q}}^n$, $b \geq b^*$		$\overline{\boldsymbol{\Sigma}}_{\mathcal{G}}(\mathbf{X}) = \boldsymbol{\Sigma}_{\mathbf{Q}}(\mathbf{X}) + \boldsymbol{\xi}$ with $\xi_j \sim \mathcal{N}(0, \sigma^2), \mathbf{Q} \sim p_{\mathbf{q}}^n$, $\sigma \geq \sigma^*$	
	UDP	**BDP**	**UDP**	**BDP**
Generic	Lemma 8.16: $e^{\varepsilon^*} = \sup_{\mathbf{x}} \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b^*} Q_1^{\mathrm{U}}(\mathbf{x}, \mathbf{q})\right)$	Lemma 8.25: $b^* \leq \sup_{\mathbf{Q}} \Delta_1^{\mathrm{B}}(\boldsymbol{\Sigma}_{\mathbf{Q}})/\varepsilon$	Lemma 8.25: $\sigma^* \leq \frac{\alpha}{\sqrt{2\varepsilon}} \sup_{\mathbf{q}} Q_2^{\mathrm{U}}(\mathbf{q})$	Lemma 8.25: $\sigma^* \leq \frac{\alpha}{\sqrt{2\varepsilon}} \sup_{\mathbf{q}} Q_2^{\mathrm{B}}(\mathbf{q})$
RFF	Lemma 8.19: $b^* \leq \sqrt{2}m/\varepsilon$	Lemmas 8.18 and 8.25: $b^* \leq 2\sqrt{2}m/\varepsilon$	Lemmas 8.25 and 8.26: $\sigma^* \leq \frac{\alpha}{\sqrt{2\varepsilon}} \frac{m}{\sqrt{r}}$	Lemmas 8.25 and 8.26:[1] $\sigma^* \leq \frac{\alpha}{\sqrt{2\varepsilon}} \frac{2m}{\sqrt{r}}$
+ $\boldsymbol{\Omega}$ nonresonant	$b^* = \sqrt{2}m/\varepsilon$ (same lemma)	not covered	not covered	not covered
RQF	Lemma 8.21: $b^* \leq \frac{m}{r\varepsilon} \sup_{\mathbf{q}} \lVert\boldsymbol{\Omega}_{\mathbf{q}}\rVert_2^2$	Lemmas 8.20 and 8.25: $b^* \leq \frac{m}{r\varepsilon} \sup_{\mathbf{q}} \lVert\boldsymbol{\Omega}_{\mathbf{q}}\rVert_2^2$	Lemmas 8.25 and 8.27: $\sigma^* \leq \frac{\alpha}{\sqrt{2\varepsilon}} \frac{m}{r} \sup_{\mathbf{q}} S_4(\boldsymbol{\Omega}_{\mathbf{q}})$	
+ $\boldsymbol{\Omega}$ union of orthogonal bases and $\mathbf{q} \sim \mathcal{U}(\mathcal{Q}_r^{\mathrm{struct.}})$.	Lemma 8.23: $b^* = m/(d\varepsilon)$	Lemmas 8.22 and 8.25: $b^* \leq m/(d\varepsilon)$	No particular closed form	

Table 8.2: Summary of privacy results **with** subsampling (Section 8.2). For each type of privacy guarantee (column) and for each sketch feature function (row), we provide a potentially loose ($\leq$) or sharp ($=$) bound on the optimal required additive noise levels (b^* or σ^*). [1] Using a simple triangle inequality.

Chapter 9
Utility Guarantees under Differential Privacy

Note All the work related to the NSR is contained in our submission to *Information and Inference* [27]. Experimental results from Section 9.3 are new and are likely to be submitted in the next months together with other empirical works on broader applications of the private sketches.

Contents

Having established the differential privacy properties of noisy sketching mechanisms, we conclude this part of the thesis with an investigation of the impact of this added noise on the utility of the sketches for learning. We will see that this utility – i.e. the quality of the models learned from noisy sketches, as measured by the metrics introduced in definitions 1.1 to 1.3 – is directly related to the noise level. This suggests to use a noise-to-signal ratio (NSR) as a practical proxy for the utility (Section 9.1.1). The NSR is then computed analytically (Section 9.1.2) and used to tune some of the parameters of our method: the subsampling strategy (Sections 9.2.1 and 9.2.2), the splitting of the privacy budget (Section 9.2.3), and the sketch size (Section 9.2.4). We then provide in Section 9.3 experiments for clustering and PCA, and compare our method to other standard approaches from the literature in terms of privacy-utility tradeoff.

Summary of contributions The contributions of this chapter are the following:

- We show that the utility of a noisy sketch, i.e. its quality for subsequent learning, can be measured by a **signal-to-noise ratio**.
- We use this quantity for **tuning** some parameters of the method.
- We also **measure empirically** the utility of our methods (in terms of risk), and compare them to other approaches from the literature.

9.1 Assessing utility with the noise-to-signal ratio

We first define the noise-to-signal ratio and show how it is useful in Section 9.1.1, and then derive its analytical expression in Section 9.1.2.

9.1.1 The role of the noise-to-signal ratio

We recall[1] that a learning task is defined by a risk function $\mathcal{R}$ and an hypothesis space $\mathcal{H}$, and the optimal parameters one would like to learn are

$$h^* \in \arg\min_{h \in \mathcal{H}} \mathcal{R}(\pi, h),$$

where π is the true (unknown) distribution of the data. The goal is to estimate[2] a set of parameters $\hat{h}$ such that the excess risk $\Delta\mathcal{R}(\hat{h}, \pi) = \mathcal{R}(\hat{h}, \pi) - \mathcal{R}(h^*, \pi)$ can be controlled. As discussed in Chapter 2, previous works [210] showed that such a control can be achieved using proof techniques that leverage the analogy between sketching and compressive sensing.

In practice, we observe a noisy version $\mathbf{z}(\mathbf{X})$ of the empirical sketch, which can for instance[3] be computed as the ratio $\mathbf{z}(\mathbf{X}) = (\boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi})/(|\mathbf{X}| + \zeta)$ with $\boldsymbol{\xi}$ being either Laplacian or Gaussian according to Definitions 8.1 and 8.9. As shown[4] in [210], for k-means clustering, Gaussian mixture modeling and PCA, learning from the noisy sketch $\mathbf{z}(\mathbf{X})$ can be expressed as solving a linear inverse problem on a certain parametric set of probability distributions.

Under some assumptions on the sketching function Φ and the learning task, the excess risk can be bounded by a quantity that involves a measure of noise level $\|\mathbf{e}\|_2$, with $\mathbf{e} \triangleq \mathbf{z}(\mathbf{X}) - \mathbf{s}$. As a proxy for the utility of a noisy sketch, we thus propose the noise-to-signal ratio (NSR), defined as

$$\mathrm{NSR} \triangleq \frac{\|\mathbf{z}(\mathbf{X}) - \mathbf{z}\|_2^2}{\|\mathbf{z}\|^2}$$

w.r.t. some reference sketch $\mathbf{z}$, that will typically be the clean empirical sketch of $\mathbf{X}$, i.e. $\mathbf{z} = \tilde{\mathbf{s}}$, or the sketch $\mathbf{z} = \mathbf{s}$ of the assumed underlying distribution π.

[1] Cf. Section 1.1.1.

[2] In our case, from the noisy sketch only.

[3] Remember we could use any other estimator relying on the quantities $\boldsymbol{\Sigma}(\mathbf{X}) + \boldsymbol{\xi}$ and $|\mathbf{X}| + \zeta$, e.g. to avoid numerical issues.

[4] And discussed in Chapter 2.

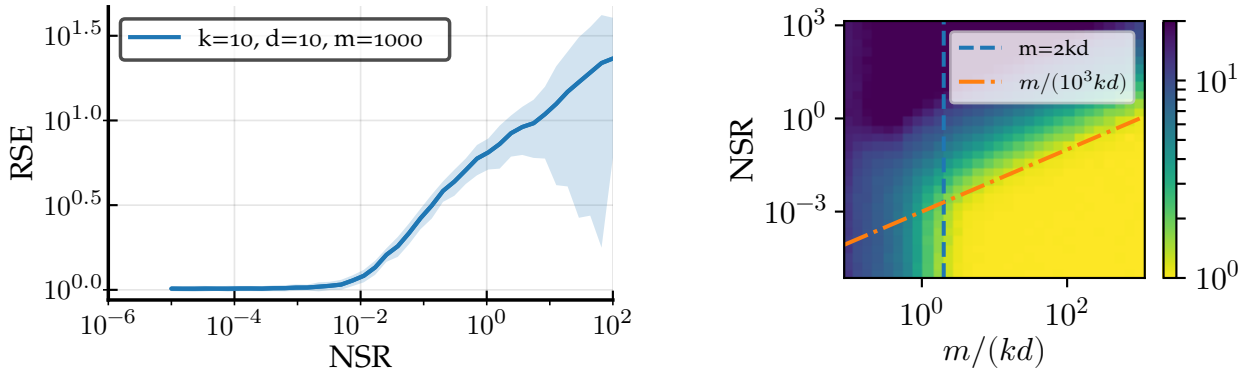


Figure 9.1: (**Left**) Correlation between RSE and NSR (computed numerically, with respect to $\tilde{\mathbf{s}}$). Medians of 100 trials and variance. (**Right**) RSE as a function of m/kd and NSR, using $n = 10^4$, $k = d = 10$, medians of 100 trials.

Empirically, the NSR appears to be a good proxy to estimate the utility of a sketching mechanism for the task of clustering where performance is measured with the MSE (cf. (4.3)), i.e. with the risk $\mathcal{R}_{\mathrm{KM}}$ from Definition 1.2. Figure 9.1 provides an overview of this correlation. On the left side, we plot the RSE[5] for data generated according to Gaussian mixtures with parameters $k = d = 10$ and $m = 10kd$. The desired NSR is obtained by adding isotropic noise of controlled

[5] We recall that the RSE is the ratio between the MSE obtained with centroids determined from a sketch and the MSE obtained with centroids computed using Lloyd's algorithm, cf. (4.4).

magnitude on the clean sketch computed without subsampling. In Figure 9.1 (right side), we plot the RSE for $n = 10^4$ using different sketch sizes and NSRs, again obtained with isotropic noise and without subsampling. The red dashed line corresponds to $m = 2kd$, and as expected [2] the reconstruction fails below this line[6]. From this plot, we derive that when $m \geq 2kd$, one can consider that the reconstruction is successful provided that NSR $\leq m/(10^3 kd) \triangleq \mathrm{NSR}_{\max}(m)$ (yellow area). We thus propose to use NSR-minimization as a criterion to tune the parameters of our method.

[6] We use CL-OMPR for reconstruction here.

9.1.2 Analytical estimation of the noise level

We now compute in this section the expected noise level and NSR induced by the mechanisms introduced in the previous sections and possibly combined with a data subsampling mechanism introduced just below.

Let $\mathbf{X}$ be a fixed dataset. The noise level can be measured with respect to the "true" sketch $\mathbf{s}$ of the assumed underlying distribution π, or with respect to the clean empirical sketch $\tilde{\mathbf{s}}$. In the first case, which is relevant to take into account the statistical significance due to the size n of the dataset, we define $\mathbf{e} \triangleq \mathbf{z}(\mathbf{X}) - \mathbf{s}$, and the noise level as $\mathbf{E}\|\mathbf{e}\|_2^2$, where the expectation is taken on both the randomness of the mechanism and on the draw of $\mathbf{X}$. We define the noise-to-signal ratio (NSR) as the noise level normalized by the signal energy, i.e. $\mathrm{NSR} = \mathbf{E}\|\mathbf{e}\|_2^2 / \|\mathbf{s}\|_2^2$. When $\tilde{\mathbf{s}}$ is chosen as the reference signal rather than $\mathbf{s}$, we have $\mathrm{NSR} = \mathbf{E}\|\mathbf{z}(\mathbf{X}) - \tilde{\mathbf{s}}\|_2^2 / \|\tilde{\mathbf{s}}\|_2^2$, and the expectation is taken w.r.t. the randomness of the mechanism only.

Subsampling the dataset. Although we focused in Section 8.2 on a mechanism which subsamples the individual features $\Phi(\mathbf{x}_i)$, another straightforward way to reduce the computational complexity is to subsample the dataset, i.e. to use only a number $n' < n$ of samples in order to compute the sketch. The sum of features[7] combining both subsampling strategies is given as

$$\widetilde{\boldsymbol{\Sigma}}(\mathbf{X}) \triangleq \frac{1}{\beta}\frac{m}{r}\sum_{i=1}^{n} g_i(\Phi(\mathbf{x}_i) \odot \mathbf{q}_i) + \boldsymbol{\xi}, \tag{9.1}$$

where the $(g_i)_{1\leq i\leq n}$ are in $\{0,1\}$ and randomly drawn (i.i.d. Bernouilli with parameter β or without replacement), and the additive noise $\boldsymbol{\xi}$ (Laplacian or Gaussian) and the masks $(\mathbf{q}_i)_{1\leq i\leq n}$ are drawn as previously[8].

[7] The tilde in the notation $\widetilde{\boldsymbol{\Sigma}}$ is not related in any way to the choice of the notation $\tilde{\mathbf{s}}$ for the empirical sketch.

[8] Cf. Section 8.2.

Privacy amplification through sampling has already been discussed in the literature [319]. In particular, it was shown by Balle et al. that if a random mechanism $\boldsymbol{\Sigma}$ is ε-UDP, then the mechanism $\mathbf{X} \mapsto \boldsymbol{\Sigma}(\mathcal{S}(\mathbf{X}))$ where $\mathcal{S}$ denotes dataset Bernoulli subsampling with parameter β, is $\log(1 + \beta(\exp(\varepsilon) - 1))$-UDP [320, Table 1]. Mechanisms for which this bound is sharp can be exhibited (same reference, Section 5). The same holds for BDP when sampling $n' = \beta n$ samples without replacement. Put differently, for $\varepsilon > 0$ fixed, our sketching mechanism composed

with dataset subsampling will be ε-DP provided that we tune the noise level σ_ξ^2 with respect to the privacy parameter $\varepsilon' = \log(1 + (\exp(\varepsilon) - 1)/\beta) > \varepsilon$. This will decrease the required variance σ_ξ^2 of the noise added to achieve the targeted privacy, but will in turn increase the variance of the sketch, precisely because of the data subsampling. The impact of subsampling the dataset is discussed in Section 9.2.1. From now on, we consider an estimator $\mathbf{z}(\mathbf{X})$ of $\tilde{\mathbf{s}}$ as a function of the sum of features $\widetilde{\boldsymbol{\Sigma}}(\mathbf{X})$ introduced in Equation (9.1), which by an adequate choice of the parameters encompasses all the mechanisms previously defined.

Noise-to-signal ratio when n is public. When the dataset size n is assumed to be public[9], we can use the estimator $\mathbf{z}(\mathbf{X}) \triangleq \widetilde{\boldsymbol{\Sigma}}(\mathbf{X})/n$. The following result is proved in Appendix C.4 (Proof C.10).

[9] Recall that this is always the case in the BDP setting.

Lemma 9.1

The noise-to-signal ratio of the mechanism $\mathbf{z}(\mathbf{X}) = \widetilde{\boldsymbol{\Sigma}}(\mathbf{X})/n$ with additive noise of variance σ_ξ^2, features subsampling with parameter $\alpha \triangleq r/m$ and i.i.d. Bernoulli subsampling of the dataset samples with parameter $0 \leq \beta \leq 1$ is

$$\text{w.r.t. } \mathbf{s}: \quad \mathrm{NSR}_{\mathbf{z}} = \frac{1}{n}\left(\frac{1}{\alpha\beta}\frac{\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|^2}{\|\mathbf{s}\|^2} - 1\right) + \frac{m}{n^2\beta^2}\frac{\sigma_\xi^2}{\|\mathbf{s}\|^2}$$

$$\text{w.r.t. } \tilde{\mathbf{s}}: \quad \mathrm{NSR}_{\tilde{\mathbf{s}}} = \frac{1}{n}\left(\frac{1}{\alpha\beta} - 1\right)\left(\frac{1}{n}\sum_{i=1}^{n}\|\Phi(\mathbf{x}_i)\|^2\right)\frac{1}{\|\tilde{\mathbf{s}}\|^2} + \frac{m}{n^2\beta^2}\frac{\sigma_\xi^2}{\|\tilde{\mathbf{s}}\|^2}.$$

The expressions for sampling without replacement differ slightly and are given in the proof in Appendix C.4.

Noise-to-signal ratio when n is sensitive. When the dataset size is considered sensitive, noise ζ must be added on n for privacy as discussed earlier. Our estimator of the sketch can then be written $\mathbf{z}(\mathbf{X}) = \widetilde{\boldsymbol{\Sigma}}(\mathbf{X})f(|\mathbf{X}| + \zeta)$, where $f(|\mathbf{X}| + \zeta)$ is an estimator of $1/|\mathbf{X}|$. The noise-to-signal ratio is now defined as

$$\mathrm{NSR} \triangleq \frac{\mathbf{E}\|\widetilde{\boldsymbol{\Sigma}}(\mathbf{X})f(|\mathbf{X}| + \zeta) - \mathbf{z}\|_2^2}{\|\mathbf{z}\|_2^2},$$

where $\mathbf{z}$ stands for the reference signal, which can again be either $\mathbf{s}$ or $\tilde{\mathbf{s}}$. An analytic expression of this NSR is given in Appendix C.4 and involves the bias and variance of the estimator of $1/|\mathbf{X}|$ defined by f. Considering an unbiased estimator, a Cramer-Rao lower bound leads to the following result which is proved in Appendix C.4.

Lemma 9.2

When using an estimator f of $1/n$ computed from the quantity $n + \zeta$, where $\zeta \sim \mathcal{L}(0, \sigma_\zeta/\sqrt{2})$, a Cramer-Rao bound on the noise-

to-signal ratio of the sketching mechanism is

$$\mathrm{NSR}_\zeta \geq \left(1 + \frac{\sigma_\zeta^2}{2n^2}\right)(\mathrm{NSR} + 1) - 1,$$

where NSR refers to the noise-to-signal ratio obtained without f (i.e. when $\zeta = 0$) as computed in Lemma 9.1, and can be computed with respect to either[10] $\mathbf{s}$ or $\tilde{\mathbf{s}}$.

10. We mean here that the quantity NSR_ζ implicitly uses the same reference sketch as NSR.

9.2 Hyperparameters tuning using the SNR

9.2.1 Comparison of the two subsampling strategies

For a given dataset size n, the sketching cost scales in $\Theta(n\alpha\beta)$ when subsampling the sketches with $r = \alpha m$ observations (with $\alpha \leq 1$) and subsampling the dataset by using only $n' = \beta n$ samples. Hence for a given n, a constant product $\alpha\beta$ means a constant computational complexity[11]. We now use Lemma 9.1 to show that, for an equivalent computational complexity and privacy, subsampling the sketches leads to a better NSR, and hence likely a better utility, than subsampling the dataset.

11. Or more precisely, a constant theoretical computational complexity. Caching mechanisms may play in favor of data subsampling, although we do not try to measure this phenomenon.

For Bernoulli sampling, the only term of the NSR (as expressed in Lemma 9.1) that varies with (α, β) at constant complexity $\alpha\beta$ is the term coming from the additive noise:

$$\mathrm{NSR}_\xi \triangleq \frac{m}{n^2\beta^2}\frac{\sigma_\xi^2}{\|\mathbf{s}\|^2}. \tag{9.2}$$

Note that this holds as well when working with the Cramer-Rao bound from Lemma 9.2, as the term σ_ζ does not depend on α, β at all. To investigate how this term varies we need to take into account that for a fixed target privacy ε, the variance σ_ξ^2 also depends on β.

Let us consider the ε-DP setting with random Fourier features as an illustration[12]. According to [320], to achieve a privacy level ε after random Bernoulli subsampling of the collection with probability β, it is sufficient to start with an initial mechanism ensuring privacy level $\varepsilon' = \log(1 + (\exp(\varepsilon) - 1)/\beta)$. Note that although we can always build mechanisms for which this generic result is sharp, it does not prove that it is sharp for our mechanism. In the absence of more detailed results in this direction, we still use this bound for comparing the two subsampling mechanisms. According to Table 8.1, the desired privacy can be achieved using a noise variance $\sigma_\xi^2 \approx (m/\varepsilon')^2$ up to a small multiplicative constant. The resulting NSR term is then of the order of

12. A similar reasoning holds for random quadratic features.

$$\mathrm{NSR}_\xi \propto \frac{m^3}{n^2\|\mathbf{s}\|^2}\frac{1}{\beta^2(\varepsilon')^2} = \frac{m^3}{n^2\|\mathbf{s}\|^2}\frac{1}{\beta^2\log^2(1 + (\exp(\varepsilon) - 1)/\beta)}. \tag{9.3}$$

In particular when $\beta = 1$ we have

$$\mathrm{NSR}_\xi \propto \frac{m^3}{n^2\|\mathbf{s}\|^2}\frac{1}{\varepsilon^2}.$$

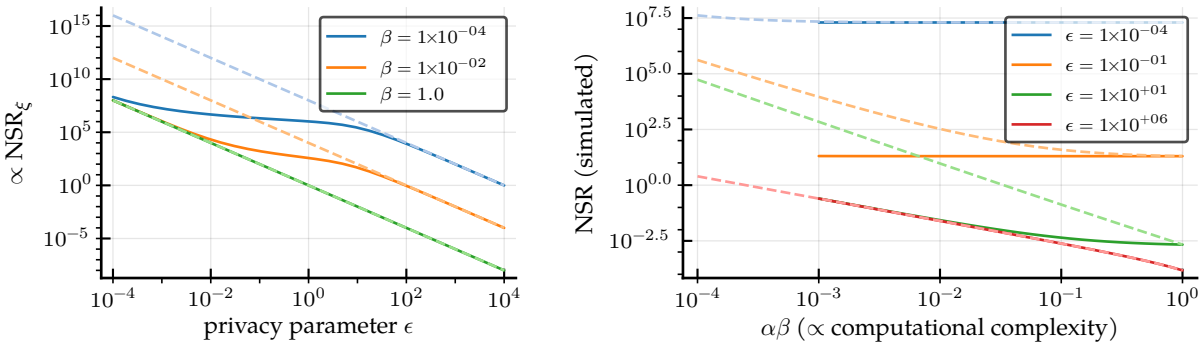


Figure 9.2: **(Left)** Variation of $1/(\beta\varepsilon')^2$. Dashed curves correspond to $1/(\beta\varepsilon)^2$, which are the asymptotes for $\varepsilon \gg 1$. **(Right)** Total NSR (analytic expressions) for $m = 10^3, n = 10^4$ when subsampling the features only ($\alpha \in [1/m, 1], \beta = 1$, plain curves) and the samples only ($\alpha = 1, \beta \in [1/n, 1]$, dashed curves, Bernoulli sampling).

When $\varepsilon \ll \beta$, we have $\varepsilon' \approx \varepsilon/\beta$ and the difference between the two subsampling schemes is negligible, but on the other side if $\varepsilon \gg 1$ then $\varepsilon' \approx \varepsilon$ and NSR_ξ is thus increased by a factor $1/\beta^2$ when subsampling on n, which might be damageable in terms of utility. This behavior is illustrated in Figure 9.2 (left) where we see that for each privacy parameter the minimum value of $1/(\beta\varepsilon')^2$ – the quantity which drives the order of magnitude of NSR_ξ – is achieved at $\beta = 1$. The effect on the total NSR for the two sampling scenarios is shown using the analytic expressions in Figure 9.2 (right), which confirms that subsampling the features rather than the samples yields substantial NSR gains for moderate ε. For large ε, the noise level is very low anyway, and no difference appears between the two scenarios. Additional experiments (not shown here) show that when sampling the dataset without replacement rather than with i.i.d. Bernoulli sampling, and measuring the NSR with $\tilde{\mathbf{s}}$ as a reference signal, subsampling the features can become slightly disadvantaging for large values of ε, but the difference is very small.

9.2.2 Regimes combining privacy and utility

In this section, we try to highlight the regimes in which the sketches produced by our mechanism are useful from a learning perspective (i.e. not too much degraded). We do so by comparing the different contributions to the NSR.

In light of the results of Section 9.2.1, we focus on subsampling the features only (i.e. $\beta = 1$). In this setting, and when working with random Fourier features, since $\|\Phi^{\mathrm{RFF}}(\mathbf{x})\|^2 = m$ for every $\mathbf{x}$, the different contributions to the NSR (computed with $\mathbf{s}$ as reference) are of the order[13] of

$$\mathrm{NSR}_{\mathbf{X}} \approx \frac{1}{n}(C_0 - 1), \qquad \mathrm{NSR}_\xi \approx \frac{C_0 m^2}{n^2\varepsilon^2}, \qquad \mathrm{NSR}_{\mathbf{Q}} \approx \frac{C_0}{n}\left(\frac{1}{\alpha} - 1\right),$$

where $C_0 \triangleq m/\|\mathbf{s}\|^2$. Using the interpretation of $\|\mathbf{s}\|^2/m$ as an expected value [210] the quantity C_0 is essentially independent of m and satisfies $C_0 > 1$. In practice, empirical simulations on very different datasets suggest that one can safely assume $C_0 < 10$.

[13] We refer the reader to Lemma 9.1 and Proof C.10 to see where the different contributions come from.

Acceptable noise level without subsampling. The total noise is acceptable when the sum of these contributions to the NSR is smaller than some threshold $\mathrm{NSR}_{\max}$, which depends on the sketch size m as seen on Figure 9.1 (right). Necessary conditions read:

$$\mathrm{NSR}_{\mathbf{X}} \leq \mathrm{NSR}_{\max} \Leftrightarrow n \gtrsim \frac{1}{\mathrm{NSR}_{\max}} \tag{9.4}$$

$$\mathrm{NSR}_{\xi} \leq \mathrm{NSR}_{\max} \Leftrightarrow n \gtrsim \frac{m}{\varepsilon\sqrt{\mathrm{NSR}_{\max}}} \tag{9.5}$$

Thus utility is preserved (and privacy achieved) when

$$n \gtrsim \max\left(\frac{1}{\mathrm{NSR}_{\max}}, \frac{m}{\varepsilon\sqrt{\mathrm{NSR}_{\max}}}\right). \tag{9.6}$$

Acceptable noise level with subsampling. The noise induced by feature subsampling is acceptable when $\mathrm{NSR}_{\mathbf{Q}} \lesssim \mathrm{NSR}_{\max}$, i.e., for subsampling with $\alpha = 1/m$, when $n \gtrsim m/\mathrm{NSR}_{\max}$. Combining this with the two conditions from the previous paragraph, we conclude that

$$n \gtrsim m \max\left(\frac{1}{\mathrm{NSR}_{\max}}, \frac{1}{\varepsilon\sqrt{\mathrm{NSR}_{\max}}}\right) \tag{9.7}$$

allows drastic feature subsampling while preserving utility.

Regime where feature subsampling adds insignificant noise. When

$$\mathrm{NSR}_{\mathbf{Q}} \ll \max(\mathrm{NSR}_{\mathbf{X}}, \mathrm{NSR}_{\xi}) \quad \Leftrightarrow \quad \frac{1}{\alpha} \ll 1 + \max(1, \frac{m^2}{n\varepsilon^2}), \tag{9.8}$$

feature subsampling adds insignificant noise compared to the other noises. When subsampling with parameter $\alpha = 1/m$, this is equivalent to $n \ll \frac{m}{\varepsilon^2}$. In light of (9.5), one can check that the regime where subsampling noise is insignificant while the total noise is acceptable corresponds to

$$\frac{m}{\varepsilon\sqrt{\mathrm{NSR}_{\max}}} \lesssim n \ll \frac{m}{\varepsilon^2}, \tag{9.9}$$

which is only feasible when the target privacy satisfies $\varepsilon \ll \sqrt{\mathrm{NSR}_{\max}}$.

Example (Compressive clustering with RFF) When performing compressive clustering using random Fourier Features, we observed empirically on Figure 9.1 (right) that $\mathrm{NSR}_{\max} \approx 10^{-3}m/(kd)$. Thus the condition (9.7) for having an acceptable noise level when subsampling can be rewritten

$$n \gtrsim \max\left(10^3 kd, \frac{\sqrt{10^3 kdm}}{\varepsilon}\right). \tag{9.10}$$

Similarly, subsampling with $\alpha = 1/m$ will induce an insignificant noise level only when $\varepsilon \ll \sqrt{10^{-3}m/(kd)}$ according to Equation (9.9). This confirms that sketching is compatible with drastic features subsampling for private compressive clustering when working with large collections, but also that subsampling can be performed without any impact on the NSR for high privacy levels.

9.2.3 A Heuristic for Privacy Budget Splitting (Laplacian Noise)

When using unbounded differential privacy, one needs to split the total privacy budget ε between a budget $\varepsilon_\xi \triangleq \gamma\varepsilon$ (where $\gamma \in]0,1[$) used for releasing the sum of sketches $\widetilde{\Sigma}$, and a budget $\varepsilon_\zeta \triangleq (1-\gamma)\varepsilon$ used for releasing the dataset size. We recall that for BDP, the dataset size is not considered as sensitive and can be used directly.

We build a heuristic for the ε-DP setting which consists in choosing $\gamma^* \in]0,1[$ minimizing the NSR. In light of Section 9.2.1, we consider for simplicity $\beta = 1$, i.e. subsampling is only performed on the features but not on the samples. We further focus on random Fourier features, where $\|\Phi^{\text{RFF}}(\mathbf{x})\|_2^2 = m$ does not depend on $\mathbf{x}$, leading to a simplified expression of the Cramer-Rao bound on the NSR from Lemma 9.2:

$$\text{NSR}_*^{\text{RFF}} = \left(1 + \frac{\sigma_\zeta^2}{2n^2}\right)\left(1 - \frac{1}{n} + \frac{m}{n\|\mathbf{s}\|^2}\left(\frac{1}{\alpha} + \frac{1}{n}\sigma_\xi^2\right)\right) - 1,$$

with $\mathbf{s}$ as the reference signal. By injecting for σ_ζ^2 and σ_ξ^2 the values obtained previously for the UDP Laplacian setting, see Table 8.1, we get an expression of $\text{NSR}_*^{\text{RFF}}$ as a function of γ, which can be minimized w.r.t. the parameter γ.

Lemma 9.3

For random Fourier features, an expression of the parameter γ^* minimizing $\text{NSR}_*^{\text{RFF}}$ is given in Appendix C.5 as a function of ε and n. The following approximations can be derived

$$\text{when } n \ll 1/\varepsilon, \gamma^*(n,\varepsilon) \approx 1/2$$
$$\text{when } n \gg 1/\varepsilon, \gamma^*(n,\varepsilon) \approx 1 - (n\varepsilon)^{-2/3}.$$

In practice, it is important to choose γ independently of n in order for the whole mechanism to stay private. Given that the NSR only decreases with n, we have for any $\varepsilon > 0$ and any n_0 that

$$\arg\min_\gamma \max_{n \geq n_0} \text{NSR}(\gamma, n) = \gamma^*(n_0, \varepsilon),$$

yielding a simple rule to choose γ. In light of Section 9.2.2, in the regime of acceptable noise levels we have

$$n \gtrsim n_0(m,\varepsilon) \triangleq m \max\left(\frac{1}{\text{NSR}_{\text{max}}}, \frac{1}{\varepsilon\sqrt{\text{NSR}_{\text{max}}}}\right) \gg 1/\varepsilon$$

hence a possible heuristic is to choose

$$\gamma(m,\varepsilon) \triangleq \gamma^\star(n_0(m,\varepsilon),\varepsilon) \approx 1 - (n_0\varepsilon)^{-2/3}.$$

Note that this is only a heuristic, allowing to choose γ independently of n but optimized for the worst-case scenario with acceptable utility; even if $n < n_0$ the mechanism will be guaranteed to be private (although with limited utility).

9.2.4 Choice of the Sketch Size

Because the noise level depends on the sketch size m, the design of a sketching procedure becomes delicate since overestimating m decreases the performance, unlike in the non-private case where increasing m usually only helps. As an illustration of this fact, consider the numerical experiment represented Figure 9.3 (top row), where we estimate the RSE[14] achieved by compressive k-means (CKM) from the ε-DP sketch as a function of its size m. As expected, in the non-private setting the SSE decreases monotonically with m. However, when $\varepsilon < \infty$ and the number of samples n is moderate, increasing m (and thus the noise, which is proportional to m according to Lemma 8.6) results in a worse RSE at some point. This phenomenon is more pronounced when the privacy constraints are higher, i.e. a smaller ε induces a smaller range of "optimal" values for the sketch size. There is thus a trade-off to make between revealing enough information for CKM to succeed (m large enough) and not revealing too much information, such that the noise needed to ensure the privacy guarantee is not too penalizing, this trade-off being more difficult in the high privacy regime.

[14] The reference used to compute the relative error is still the standard k-means algorithm with 3 trials, and still run on a non-private sketch.

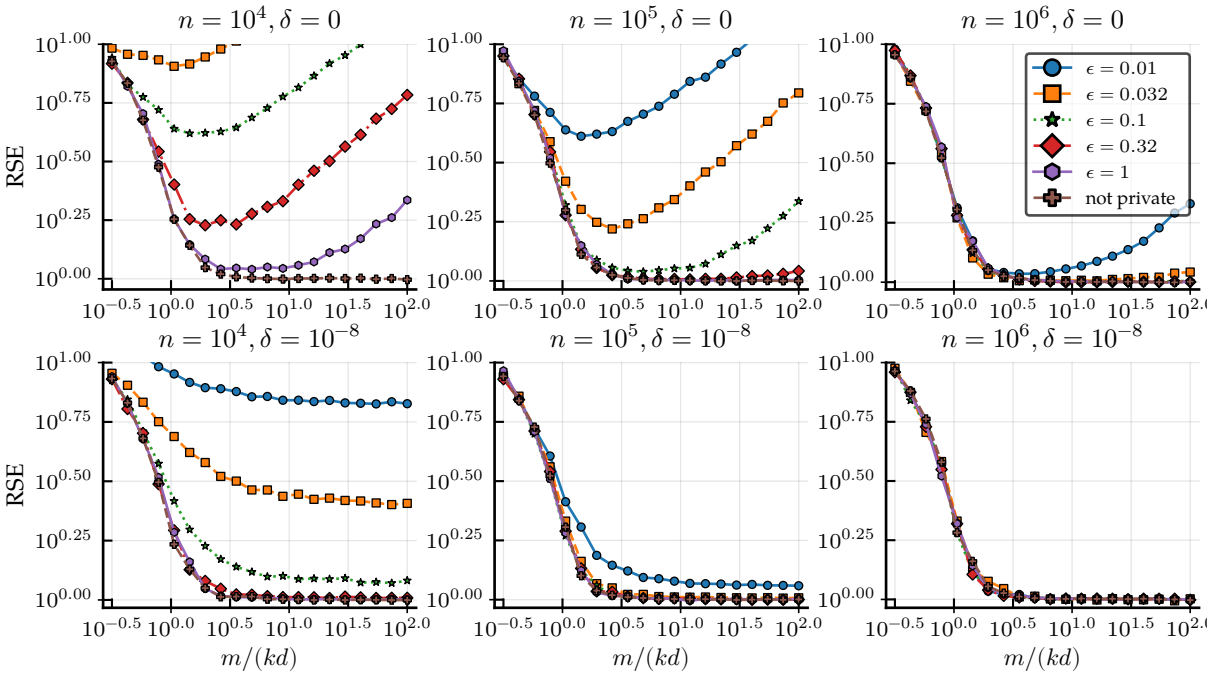


Figure 9.3: Performance of differentially private compressive k-means as a function of m for $\delta = 0$ (top) and $\delta = 10^{-8}$ (bottom), $n = 10^4, 10^5, 10^6$ and different values of ε. Medians over 200 trials. Synthetic data, $k = 4, d = 8$.

This behavior can be explained by the observations of Section 9.2.2 (paragraph "acceptable noise level") relative to the NSR. We consider for conciseness here that no subsampling is used (i.e. $\mathrm{NSR}_{\mathbf{Q}} = 0$) and ε-DP ($\delta = 0$). Given that utility is measured w.r.t. the RSE – which is relative to the optimal error for the given dataset, but agnostic to the true data distribution –, we take $\tilde{\mathbf{s}}$ as the reference signal to compute the NSR, i.e. we have $\mathrm{NSR}_{\mathbf{X}} = 0$. Utility is then preserved provided that $\mathrm{NSR}_{\xi} \leq \mathrm{NSR}_{\max}$, which according to (9.10) translates to the condition $n \geq \sqrt{1000kdm}/\varepsilon$. Recall that we also need $m \geq 2kd$ as shown in Section 9.1.1. These conditions can be

rewritten $2 \leq m/(kd) \leq n^2\varepsilon^2/(10^3(kd)^2)$, which is possible only when $n \geq \sqrt{2 \times 10^3}kd/\varepsilon$. In Figure 9.3 we have $k = 4, d = 8$, thus this requirement translates respectively for $n = 10^4, 10^5, 10^6$ to the conditions $\varepsilon \geq 0.14, \varepsilon \geq 0.014, \varepsilon \geq 0.0014$, which correspond quite well to what is observed (top row).

As shown on Figure 9.3 (bottom), relaxing the privacy constraint to allow $\delta > 0$ mitigates the impact of m on the noise to add (recall from theorem 8.11 that the noise is then proportional to $\sqrt{m}$ instead of m), and that even for smaller values of n. This relaxation has the clear advantage of improving the utility for similar values of n and ε even for small δ, and also facilitates the choice of m, as good utilities can be reached on a wider range of sketch sizes.

9.3 Experimental validation

We now evaluate experimentally the privacy-utility trade-off of our method for clustering and principal component analysis[15]. We use the acronyms CKM and CPCA for our compressive methods. Details regarding the implementation and datasets can be found in Appendix D.

[15] Experiments for density fitting with Gaussian mixture models have been performed as well, but are not reported here as they have mainly been carried out by Vincent Schellekens.

9.3.1 Clustering

A note on the frequency distribution For random Fourier features, the entries of the matrix of frequencies $\mathbf{\Omega}$ are Gaussian with a variance σ_κ^2, which relates to the spatial characteristics of the dataset and needs to be wisely chosen as explained in Chapter 4. Estimating the optimal kernel variance from the data can be a tricky problem, but is not our focus here and we thus pick the kernel variance manually to dissociate this issue from privacy concerns.

As σ_κ^2 is only a single scalar, and can reasonably be bounded (assuming a bounded data domain), we expect that any heuristic to estimate a good value for σ_κ^2 (e.g. performing one more pass on the data or using a small subset of the collection) can be adapted to the private setting, using only a small portion of the privacy budget and without degrading utility too much.

Synthetic data We first consider synthetic data drawn according to a mixture of k multivariate normal distributions as described in (4.5), with the centroids drawn according to (4.6) with separation parameter $s = 2.5$. The frequencies are drawn according to an "adapted radius" kernel, with the kernel variance chosen using the heuristic provided in Chapter 4.

Figure 9.4 shows the curves of the privacy-utility tradeoff on such a synthetic dataset, using $d = 10$, $k = 10$ (both for generating the data and for performing k-means), and $n = 10^7$. The first plot (starting from the top) illustrates the role of the sketch size for both CL-OMPR and CL-AMP decoders. We observe that the best tradeoff in the high privacy regime (small ε) is obtained with smaller values of the sketch size m, i.e. $m = 4kd$ or even[16] $m = 2kd$, which is coherent with

[16] Only using CL-OMPR in this case, which is coherent with what has been observed in Section 6.4.

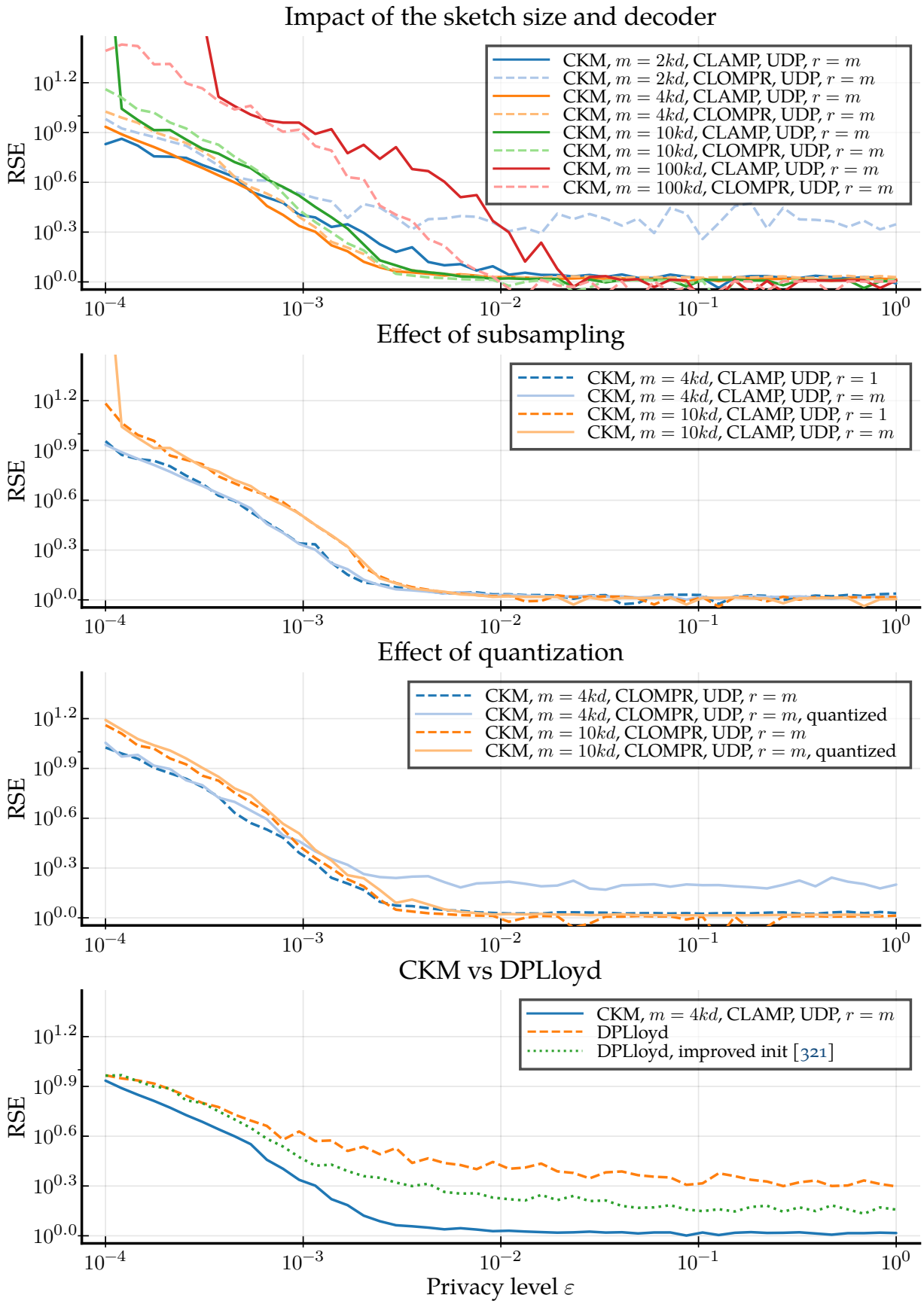


Figure 9.4: Privacy-utility tradeoff on synthetic data for ε-UDP. Parameters: $k = d = 10$, $n = 10^7$. Data generated according to (4.5) with separation factor $s = 2.5$. Medians over 50 trials (except for $m = 100kd$ where fewer trials are used).

our observations from Section 9.2.4. As expected, in the low-privacy regime (ε large), we observe the inverse phenomenon and using very large sketch sizes such as $m = 100kd$ only improves performance. We also note that CL-AMP performs poorly when $m = 100kd$ in the high-privacy regime, which could be justified by the fact that the modeling assumptions for this method do not incorporate noise[17].

[17] As the noise level itself is not private, we could imagine taking it into account in the model.

The second plot (starting from the top on Figure 9.4) shows that subsampling the data using only one feature per data sample does not degrade the performance in this setting. This confirms that subsampling can sometimes be used without degrading utility, despite not yielding better privacy guarantees (and hopefully have a beneficial impact on the runtimes).

The third plot illustrates the impact of quantization. We do obtain higher errors here with quantization when $m = 4kd$, however note that this degradation is solely due to quantization, and not to the additive perturbation (as can be seen when ε is close to 1). Interestingly, in the high privacy regime (ε small) the impact of the additive noise becomes stronger and the degradation of utility due to quantization hence becomes negligible in comparison. This suggests that quantization might be used more broadly when strong privacy guarantees are required. When $m = 10kd$, the results obtained with quantization are almost identical to the ones obtained with unquantized features.

Finally, the last plot compares our method to the differentially private version of Lloyd's k-means algorithm (DPLloyd) [274], and to a variant of DPLloyd with improved initialization [321]. These iterative methods suffer from their fixed number of iterations, even for large values of ε, whereas CKM does not add any noise in this setting, and thus yields better results in this setting.

Real data We provide clustering results for two real datasets, Gowalla[18] which consists in $n = 6,442,892$ locations in dimension $d = 2$, and FMA[19] [322], which is a dataset for music analysis. In the latter case, we only consider the MFCC attributes, yielding $n = 106,574$ features in dimension $d = 20$. In addition to DPLloyd, we compare our results with EUGkM [321] on Gowalla, as this methods relies on histograms and is only appropriate in small dimension[20], and with PrivGene [289] but only for FMA as this method does not scale[21] well with n.

[18] See https://snap.stanford.edu/data/loc-gowalla.html.

[19] See https://github.com/mdeff/fma.

[20] Which is also why we did not use this method in Figure 9.4.

[21] Same remark.

Results are presented in Figure 9.5. We use for CKM unquantized features without subsampling, and the random frequencies are drawn with manually chosen[22] kernel variances $\sigma^2 = 1/310$ for Gowalla and $\sigma^2 = 1/2500$ for FMA. We report the results for both Gaussian and adapted radius kernels, as none of the two is uniformly better than the other. With a sketch size of $m = 4kd$, we obtain errors which are always better than DPLloyd and EUGkM on both datasets. PrivGene yields slightly better results for small values of ε, but produces very degraded centroids even for higher values of ε, and scales poorly with the number of samples. Interestingly, on the FMA dataset (right), better compressive clustering results are obtain with a Gaussian kernel when the noise-to-signal ratio grows (i.e. for lower values of ε). Note

[22] Following observations of Chapter 4, this choice might be tricky.

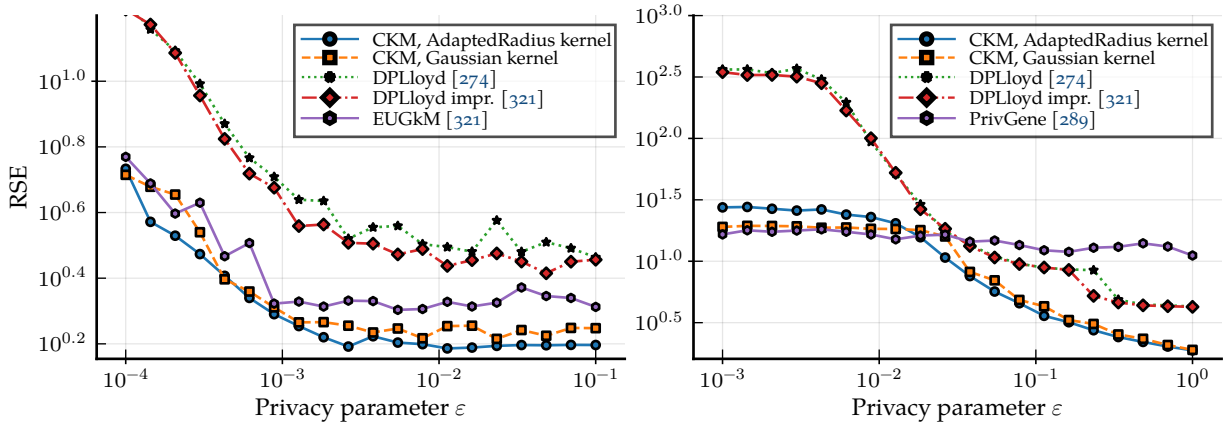


Figure 9.5: Privacy-utility tradeoff on Gowalla (left) and FMA (right, using MFCC features only) datasets. Medians over 100 trials. Results for CKM have been obtained using $m = 4kd$.

that the EUGkM algorithm is well adapted to the Gowalla dataset here where $d = 2$, but would not scale at all to larger dimensions as it relies on spatial histograms.

9.3.2 Principal component analysis

For principal component analysis, we first provide pure DP results on two real datasets of moderate size, for which the $d \times d$ covariance matrix can perfectly be computed, and thus for which methods scaling in $\Theta(d^2)$ can be run as well for comparison. We then provide results on synthetic high-dimensional data, for both pure and approximate differential privacy.

Real data We present the utility-privacy tradeoff for PCA on two different real datasets: kddcup99[23], a collection of network metadata in dimension $d = 107$ after converting categorical values to binary features, and FMA which has been previously introduced, here using all the features ($d = 518$). Data is centered and rescaled in order to lie in the l_2 unit ball.

We use a dense matrix $\mathbf{\Omega}$ for simplicity here, as the dimensions involved are still moderate, but fast transforms from Chapter 5 can be used[24] and work similarly well. We will use fast transforms below, as they are really required in high dimension. Indeed, as a sketch size $m \gtrsim kd$ is required in practice for successful recovery, the cost of storing the dense matrix $\mathbf{\Omega}$ of size $d \times m$ scales quadratically with the dimension d. This means that whenever the covariance cannot be computed in terms of memory, fast transforms are required.

For principal component analysis, the inverse problem consists in recovering a low-rank approximation of the covariance matrix from the sketch, as discussed in Section 2.2.2, and could thus be written

$$\hat{\mathbf{C}} \in \underset{\mathbf{C} \succeq 0, \operatorname{rank}(\mathbf{C}) \leq k}{\arg\min} \|\mathcal{M}(\mathbf{C}) - \mathbf{z}(\mathbf{X})\|_2, \tag{9.11}$$

where $\mathcal{M} : \mathbf{A} \mapsto [\boldsymbol{\omega}_1^T \mathbf{A} \boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m^T \mathbf{A} \boldsymbol{\omega}_m]$, and $\mathbf{z}(\mathbf{X})$ denotes any[25] of the private sketches from Chapter 8 built on random quadratic features. We do not attempt to solve (9.11), as it would incur a $\Theta(d^2)$ memory

[23] See `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[24] For PCA the columns of $\mathbf{\Omega}$ can be chosen to have a unit l_2-norm. As a consequence, only the structured part of the fast transforms is required (radiuses can be fixed to 1), and we get a closed form of the privacy bound as detailed in Equation (8.6).

[25] With Laplacian or Gaussian noise, BDP or UDP.

overhead when optimizing on $\mathbf{C} \in \mathbb{R}^{d \times d}$. We rely instead on a Burer-Monteiro factorization [323], i.e. we solve

$$\min_{\mathbf{U} \in \mathbb{R}^{d \times k}} \|\mathcal{M}(\mathbf{U}\mathbf{U}^T) - \mathbf{z}(\mathbf{X})\|_2^2. \tag{9.12}$$

Although dedicated methods have been proposed to solve this problem using inertial methods and carefully tuned spectral initializations [324, 325], we observed in our noisy setting that optimizing directly the relaxed objective (9.12) with a random initialization yielded better results in practice. We use the L-BFGS[26] algorithm, starting from an initialization $\mathbf{U}_0$ with normal standard i.i.d. entries.

We implement a simple baseline (called LaplacePCA) which consists in computing a private version of the covariance matrix with the Laplace mechanism. For this, we first compute the matrix $\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T$, then add symmetric Laplacian noise with scale $\frac{2d}{n\varepsilon}$. This mechanism is known to be private as a straightforward application of the Laplace mechanism, see for instance [285, Theorem 3]. We also provide results for the Wishart mechanism [285, 286], which perturbs the full covariance matrix by a $d \times d$ noise matrix drawn according to the Wishart distribution – and hence preserves positive semi-definiteness.

[26] The limited memory Broyden-Fletcher-Goldfarb-Shanno.

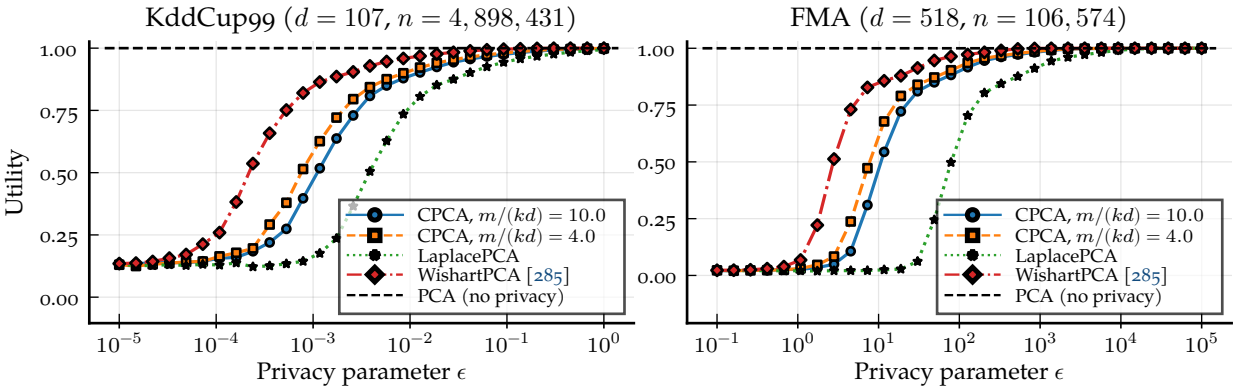


Figure 9.6: Privacy-utility tradeoff on kddcup99 and FMA datasets. Medians over 100 trials.

Privacy-utility curves are presented in Figure 9.6, and for ε-DP only. Denoting $\mathbf{C}$ the covariance matrix and $\lambda_1 \geq \ldots \geq \lambda_d$ its eigenvalues, we measure the quality of an orthogonal basis $\mathbf{U} \in \mathbb{R}^{d \times k}$ of a k-dimensional subspace with the quantity $\mathrm{tr}(\mathbf{U}^T\mathbf{C}\mathbf{U})/(\sum_{1 \leq i \leq k} \lambda_i)$ which is always in $[0, 1]$, and reaches 1 when $\mathbf{U}$ corresponds to the eigenvectors of $\mathbf{C}$ associated to the k largest eigenvalues. Our method performs better than Laplace PCA, but slightly worse than the Wishart mechanism. However, we emphasize that these two methods need to compute the $d \times d$ covariance matrix explicitly, which can be prohibitive or even impossible for large dimensions in a memory-limited context, whereas our sketch has a memory complexity which is in $\Theta(kd)$. The next experiment uses precisely a setting where methods scaling in $\Theta(d^2)$ cannot be used for comparison.

Large-scale setting To investigate this limited-memory setting, we generate synthetic data in dimension $d = 2^{15} = 32768$, approximately lying on a random k-dimensional subspace. More precisely, we generated a $d \times k$ matrix B with normal i.i.d. entries, and draw $\mathbf{y}_i \sim$

$B\mathbf{c} + \nu k\boldsymbol{\eta}$ where $(\mathbf{c})_{1\leq i\leq k}, (\boldsymbol{\eta})_{1\leq i\leq d}$ are also i.i.d. normal with unit variance. We then renormalize the data such that it fits in the unit ball, i.e. $\mathbf{x}_i \triangleq \mathbf{y}_i / \max_{1\leq i\leq n}(\|\mathbf{y}_i\|_2)$. We used in practice $\nu = 0.05$. We choose $\delta = 1/(100n)$ and $n = 10^4$. It would naturally be interesting to check that similar results are obtained when $n > d$, but this moderate choice of n allows us to explicitly compute the singular values of the dataset and thus to normalize the error, why one would otherwise need to rely on approximate methods.

Figure 9.7 shows the obtained results, where private-covariance-lrf refers to an algorithm proposed by Upadhyay [294, p.23 of the supplementary material], which relies on linear (w.r.t. the samples, not the distribution) sketching of the data matrix and can thus be adapted to high-dimensional and streaming settings as well.

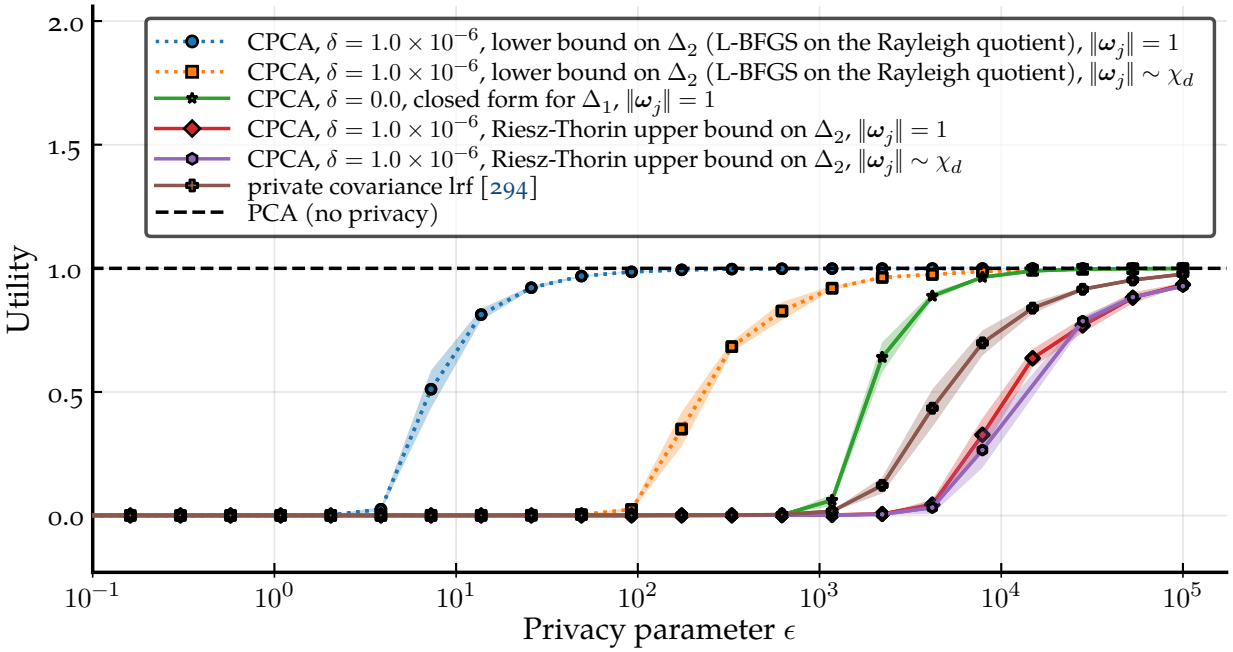


Figure 9.7: Privacy-utility tradeoff of CPCA. Medians (curves) and standard deviations (ribbon) over 10 trials. Synthetic data, $d = 2^{15}, k = 20, n = 10^4$. Dotted curves correspond to lower-bounds and are thus not private strictly speaking.

We slightly modify the algorithm from Upadhyay in two ways. First, we replace the standard Gaussian mechanism by the improved mechanism from Balle [272], so that the method remains private for values of ε greater than 1. This can only improve the initial algorithm, and makes the comparison with our method more fair as we also rely on this improved Gaussian mechanism as detailed in Section 7.3.2. Then, we modify the noise level by a factor 2 in order to yield comparable[27] privacy definitions. The method of Upadhyay also only provides (ε, δ)-DP, but we include for our method both ε-DP and (ε, δ)-DP curves on the same figure for reference, although they cannot be compared directly in terms of privacy. We do not include the methods presented in the previous section, as computing the $d \times d$ covariance matrix is prohibitive in this setting.

For our compressive method, we recall that the sensitivity needs to be computed numerically as detailed in Section 8.1.3. For the pure DP setting, we rely on the closed form (8.6) as we use fast transforms.

[27] Upadhyay defines two datasets $\mathbf{X}, \mathbf{Y}$ (seen as matrices here) as neighbors when $\|\mathbf{X} - \mathbf{Y}\|_F \leq 1$. We slightly increase the noise level, so that the algorithm remains private for $\mathbf{X}, \mathbf{Y}$ satisfying $\|\mathbf{X} - \mathbf{Y}\|_F \leq \sqrt{2}$. Note that, using our BDP neighboring relation (Definition 7.4), and assuming data in the unit ball, $\sup_{\mathbf{X}\sim\mathbf{Y}} \|\mathbf{X} - \mathbf{Y}\|_F^2 = \sup_{\mathbf{x},\mathbf{y}\in\mathcal{B}_2} \|\mathbf{x} - \mathbf{y}\|_2^2 = 2$, so our definition is, with our modification, strictly stronger than the other one.

For the ADP setting, we show both the Riesz-Thorin upper-bound (cf. Section 8.1.3), which yields a truly private (but pessimistic) method, and a lower bound obtained by maximizing the generalized Rayleigh quotient $\langle \mathcal{C}, \mathbf{x}^{\otimes 4} \rangle_F / \|\mathbf{x}\|^4$ (where $\mathcal{C} = \sum_{j=1}^m \boldsymbol{\omega}_j^{\otimes 4}$) using L-BFGS-B, starting from a uniform initialization on the sphere (the maximum over 10 initializations is reported). We also show the lower bound obtained using the same method, but drawing the norms of the $(\boldsymbol{\omega}_j)_{1 \leq j \leq m}$ vectors as $\|\boldsymbol{\omega}_j\| \overset{i.i.d.}{\sim} \chi_d$. Note that the scale of the $(\boldsymbol{\omega}_j)_{1 \leq j \leq m}$ does not matter from a learning perspective, and simply changes the expression of the sensitivity.

As can be seen on Figure 9.7, the curve corresponding to our method using the Riesz-Thorin upper-bound on the sensitivity is already quite close to the results obtained with private-covariance-lrf. There is still a large gap between these methods and the curves obtained via lower-bound on the sensitivity, but this suggests that our method is at least competitive with the other approach. Interestingly with our method, choosing the radiuses of the vectors $(\boldsymbol{\omega}_i)_{1 \leq i \leq m}$ set to 1 rather than drawn according to a χ_d distribution does improve significantly the overall performance; this could be an intrinsic gain, or also a change of behavior of the optimization landscape leading to weaker local optima. Investigating how the NSR varies with the choice of the distribution of the $(\boldsymbol{\omega}_i)_{1 \leq i \leq m}$ is left for future work, but could certainly help to get a better understanding of this behavior.

Note that our mechanism does not work better under ε-DP on this example, but this comes from the fact that the good utility is obtained on this dataset for large values of ε; hence (ε, δ)-DP should not be expected to yield significantly better results, as the noise level must still scale in $\Omega(\varepsilon^{-1/2})$ when $\varepsilon \to \infty$ [272, Section 2.3]. It remains however useful to satisfy both definitions, as (ε, δ)-DP could perfectly well improve the obtained utility on other datasets.

Note that these experimental results hold for data which is truly approximately low-rank, i.e. k needs to be chosen so that $\lambda_1, ..., \lambda_k$ concentrate a large part of the energy. When this condition does not hold, solving Equation (9.12) becomes difficult and algorithms do not converge well in practice.

Here again, building provable algorithms to learn from the sketch is a challenge in itself, especially when using the Burer-Monteiro which makes the problem non-convex [323].

9.4 Discussion and perspectives

In this chapter and the previous one, we highlighted the potential of the compressive approach to learn from potentially massive datasets using limited computational resources, while ensuring the differential privacy of the data providers at the same time. Beside being promising as privacy-inducing mechanism in terms of privacy, our framework has several key interesting features compared to other methods from the literature, that are discussed here together with main limitations

and perspectives.

Efficient and distributed learning Firstly, the computational advantages of non-private sketching remain valid after our addition of a privacy layer. In particular, learning from the sketch can be done with time and space complexities which do not depend on the number of samples n in the dataset. Moreover, the sketching process is embarrassingly parallel due to the averaging operation as highlighted in Chapter 1. Sketches coming from several separate data holders can thus be aggregated again after sketching, providing distributed differential privacy for free, without any need for a trusted central party.

Versatility Another advantage is that the sketch, acting as a surrogate for the whole dataset, contains more information than just the output of one specialized algorithm, and can thus be used multiple times. This can be leveraged to solve different learning tasks from a same sketch without breaking privacy, assuming that those tasks can be solved using the same sketching operator. This is what we already observed for random Fourier features, which can be used for both k-means clustering and fitting a Gaussian mixture model, two different but related estimation problems.

This versatility of the sketch also allows to run the learning algorithm with different initializations and parameters, producing multiple solutions; the distance to the empirical sketch can be used as a metric to pick the best of these solutions. This is in contrast with usual (e.g. iterative) differentially private methods that can be highly sensitive to the choice of such parameters (which have to be selected *a priori*, as accessing the data for parameter tuning breaks the privacy guarantee).

Open challenges Although the sketch serves as a general-purpose synopsis of the dataset, at least some a priori knowledge about the data distribution and/or the target task is required when designing the sketch feature map $\Phi : \mathbf{x} \mapsto \rho(\Omega^T \mathbf{x})$. Naturally the nonlinearity ρ must be selected according to the desired task, but we postpone to Chapter 10 the discussion on the generalization to other tasks. However even for a given non-linearity, we explained in Section 9.2.4 that the choice of the sketch size m could be seen as a trade-off between performance and privacy. Going for approximate DP mitigates this difficulty, and otherwise the NSR can help to choose this parameter as explained above.

Another crucial point is the choice of the radial distribution of the frequencies for Fourier features, and in particular the estimation of the "scale" of this probability distribution as explained in Chapter 4. This might be a limitation to using sketching in practice but, on the other side, any heuristic that could be developed in the future to estimate a good value of the kernel scale σ_κ^2 could potentially be easy to make private as it releases a single scalar value, which is bounded as soon as the data is assumed to be bounded.

Perspectives All the guarantees which have been formulated are differential privacy guarantees. Although this framework is certainly the most standard in the literature, other definitions of privacy exist as discussed in Section 7.2.3. The subsampling mechanism introduced in Section 8.2, albeit allowing to control more precisely the tradeoff between privacy and computational complexity, does not improve at all the differential privacy guarantees. It is however somehow interesting for privacy preservation, at least intuitively, as it uses less information from the dataset: in the extreme case where only one quantized Fourier feature is computed per data sample, only one bit of information is measured from each data sample[28]. Revisiting the obtained privacy guarantees with information-theoretic tools could thus lead to guarantees of a different nature, but which could potentially be improved when using subsampling.

Finally, we expect that compressive learning will be extended to more learning tasks in future works (see Part V). The private sketching framework presented here would be directly transferable to those new algorithms, although the sketch sensitivity would have to be recomputed for novel feature functions.

[28] And later averaged with other sketches.

Part V

Perspectives

Chapter 10

Towards handling broader families of compression-type tasks

Contents

Finding which learning tasks can be addressed using sketches of the form (1.10) remains one of the biggest challenges for future work. We recall that in the compressive approach, the data empirical distribution is mapped to a sketch $\tilde{\mathbf{s}}$ using a feature map Φ, and then an hypothesis $\hat{h} \in \mathfrak{S}$ is recovered from this sketch, where $\mathfrak{S}$ denotes the chosen model set.

As explained in Section 2.2, our goal in order to generalize the compressive framework to a new learning task would be to find a feature map Φ and a procedure $\hat{h}(\cdot)$ to recover an hypothesis from a sketch such that the excess risk $\Delta\mathcal{R}(\hat{h}(\tilde{\mathbf{s}}), \pi)$ (where $\tilde{\mathbf{s}} = \frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i)$ denotes the empirical sketch) is bounded with high probability when the samples $(\mathbf{x}_i)_{1 \leq i \leq n}$ are drawn i.i.d. from π. The bounds derived so far on the excess risk have been obtained [5, 6] by leveraging results from linear inverse problem, and in particular by establishing a lower restricted isometry property (LRIP) of the kind

$$\forall \tau, \tau' \in \mathfrak{S}, \ \|\tau - \tau'\|_{\Delta\mathcal{L}(\mathcal{H})} \leq C_{\kappa_\Phi} \|\mathcal{A}(\tau) - \mathcal{A}(\tau')\|_2, \tag{10.1}$$

which implies that the optimal decoder $\Delta(s) = \arg\min_{\tau \in \mathfrak{S}} \|\mathcal{A}(\tau) - s\|_2$ is instance optimal[1] for the pseudo-norm $\|\cdot\|_{\Delta\mathcal{L}(\mathcal{H})}$, i.e. robust to both noise and modeling error. This, in turn, implies that the excess risk can be controlled[2]. We also recall that in the other direction, the existence of instance optimal decoders implies that (10.1) holds for some $C_{\kappa_\Phi} > 0$. These implications are summarized on Figure 10.1.

In this chapter, we propose to show for specific families of learning tasks how simple regularity arguments can be leveraged to prove that the LRIP *cannot hold* for some model sets. Naturally, this does not imply that the excess risk cannot be controlled in another way. However, we expect that such examples could help us to derive necessary conditions that the model set (and/or the feature map) must satisfy in order for a LRIP to hold.

Our strategy will be to consider distributions of the kind[3] $\tau = \delta_{\mathbf{c}}$ and $\tau_\varepsilon = \frac{1}{2}(\delta_{\mathbf{c}+\varepsilon\mathbf{u}} + \delta_{\mathbf{c}+\mathbf{u}})$, and to show that for some judicious choices of $\mathbf{c}, \mathbf{u} \in \mathbb{R}^d$ and a $\mathcal{C}^2$ feature map Φ, the quantity $\|\mathcal{A}(\tau_\varepsilon) - \mathcal{A}(\tau)\|_2$ is dominated by $\|\tau_\varepsilon - \tau\|_{\Delta\mathcal{L}(\mathcal{H})}$ when $\varepsilon \to 0$.

We introduce in Section 10.1 the considered family of loss functions and study their directional derivatives, which allows us to derive ex-

[1] In the sense of (2.17).

[2] Cf. Section 2.2.1.

[3] We will need to choose $\mathbf{c}, \mathbf{u}$ such that the distributions belong to the model set.

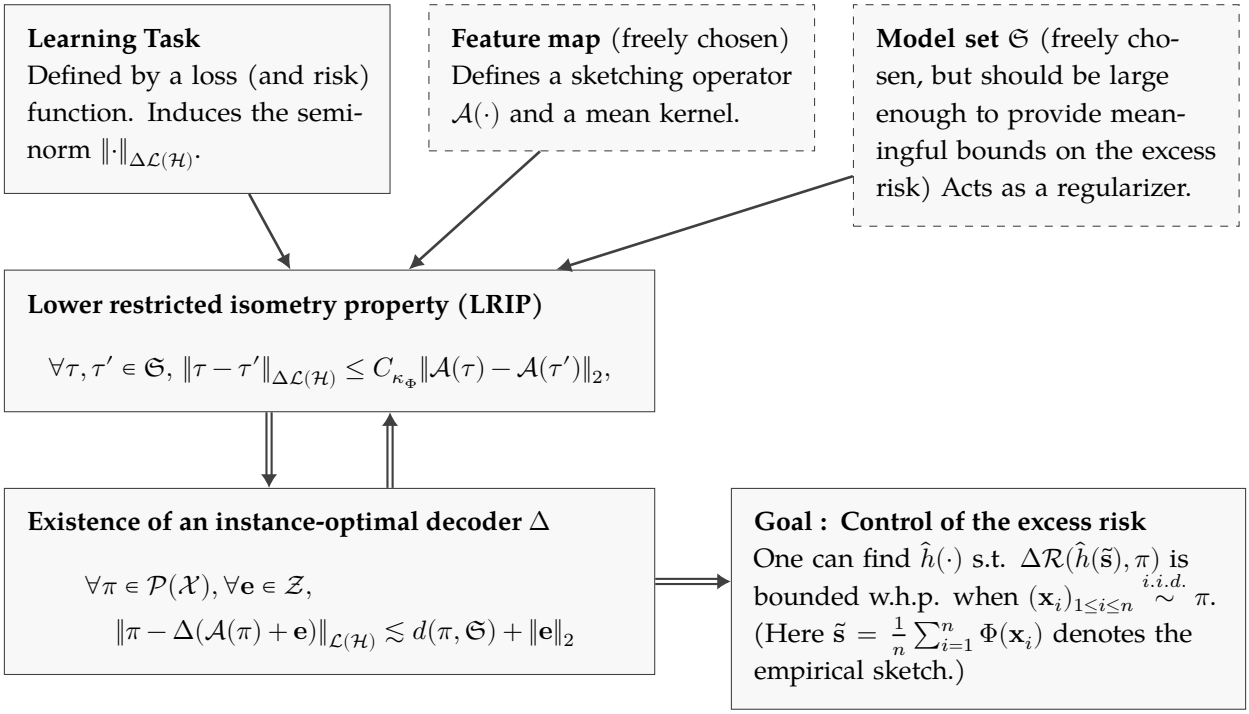


Figure 10.1: Summary of the properties used to control the excess risk. Double arrows denote implications, and simple arrows denote dependence relations. Both the model set and the feature maps can be chosen, while the loss function is fixed as it defines the learning task.

pressions for the difference between risks of such simple mixtures of diracs using geometric arguments. We will see in Section 10.2 that these results will in turn translate to lower bounds on the semi-norm $\|\cdot\|_{\Delta\mathcal{L}(\mathcal{H})}$ induced by the loss. From these observations, we derive a few impossibility results when the feature map Φ is $\mathcal{C}^2$.

We will see in particular that the LRIP cannot hold for the task of subspace clustering for the model of all unions of k subspaces, and for k-means using the model of all unions of k points[4]. Our results are summarized at the end of the chapter in Table 10.1.

[4] Which is not contradictory with existing results, which hold only for a restricted model set (a separation condition between the Diracs is added).

10.1 Working with projective losses

In all this chapter, we restrict ourselves to a particular family of learning tasks, which we define here.

Definition 10.1 (Compression-type task): A compression-type[5] task is a learning task for which the hypothesis space satisfies $\mathcal{H} \subset \mathcal{P}(\mathcal{X})$ (i.e. each $h \in \mathcal{H}$ is a subset of the data space) and the loss function can be written

$$l_p(h, \mathbf{x}) = \inf_{\mathbf{y}\in h} \|\mathbf{x} - \mathbf{y}\|_2^p \tag{10.2}$$

where $p > 0$.

[5] We follow the terminology proposed in [5], but restrict ourselves to Euclidean spaces. The definition could be generalized to arbitrary metric spaces.

In the following, the risk associated to $l_p(h, \cdot)$ is denoted $\mathcal{R}_p(h, \cdot)$. We will focus in particular on the setting $p = 1$ and $p = 2$. Note that some of the learning tasks introduced so far are already compression-type tasks, for instance:

- k-means clustering (h is a set of at most k points, and $p = 2$);
- k-medians clustering (h is a set of at most k points, and $p = 1$);
- PCA (h is an at most k-dimensional subspace, $p = 2$).

We will see below that a few other standard learning problems can be written using this formalism.

10.1.1 Projection onto closed convex sets

We first focus on the setting where h is a closed convex set[6]. In this case, the minimizer in (10.2) is unique according to the Hilbert projection theorem [326, Theorem 2 p.51]. We define the projection operator as

$$\Pi_h : \mathbf{x} \mapsto \arg\min_{\mathbf{y} \in h} \|\mathbf{x} - \mathbf{y}\|_2,$$

and we have $l_p(\mathbf{x}, h) = \|\mathbf{x} - \Pi_h(\mathbf{x})\|_2^p$.

We now provide expressions of the directional derivatives of l_2 and l_1 in this context.

Lemma 10.2

Let h be a nonempty closed convex set. Then $l_2(h, \cdot)$ is differentiable everywhere and for every $\mathbf{c}, \mathbf{u} \in \mathbb{R}^d$ we have when $\varepsilon \to 0$:

$$l_2(h, \mathbf{c} + \varepsilon\mathbf{u}) - l_2(h, \mathbf{c}) = \varepsilon 2(\mathbf{c} - \Pi_h(\mathbf{c}))^T\mathbf{u} + o(\varepsilon).$$

Proof: The function $l_2(h, \cdot)$ is differentiable with gradient $\nabla l_2(h, \mathbf{c}) = 2(\mathbf{x} - \Pi_h(\mathbf{x}))$, see e.g. [327]. This proves the existence of directional derivatives in all directions everywhere, and their expression is derived from the gradient.

Unless otherwise specified, all the asymptotic notations used from now on in the chapter hold, as in Lemma 10.2, when $\varepsilon \to 0$. In order to provide a result similar to Lemma 10.2 for the l_1 loss, we first define the notion of tangent cone which will be useful.

Definition 10.3 (Tangent direction): Let $S \subset \mathbb{R}^d$ and $\mathbf{x} \in S$. A direction $\mathbf{d} \in \mathbb{R}^d$ is called tangent to S at $\mathbf{x}$ if there exists sequences $(\mathbf{x}_i)_{i \in \mathbb{N}}$ in S and $(t_i)_{i \in \mathbb{N}}$ in $\mathbb{R}_+$ such that

$$\lim_{i \to \infty} \mathbf{x}_i = \mathbf{x}, \qquad \lim_{i \to \infty} t_i = 0, \qquad \text{and} \quad \lim_{i \to \infty} \frac{\mathbf{x}_i - \mathbf{x}}{t_i} = \mathbf{d}.$$

Definition 10.4 (Tangent cone): We define the solid tangent cone[7] of S at $\mathbf{x} \in S$, denoted $T_S(x)$, as the set of all tangent directions of S at $\mathbf{x}$.

We use this definition which is the standard one and quite generic, however we will mostly work in the following with closed convex sets, in which case the tangent cone of S at $\mathbf{x}$ can alternatively [328, Proposition 5.2.1] be defined as the topological closure of $S - \{\mathbf{x}\} \triangleq \{\mathbf{s} - \mathbf{x} | \mathbf{s} \in S\}$. In particular, in this case $T_h(\mathbf{c})$ is closed and convex and thus $\Pi_{T_h(\mathbf{c})}(\cdot)$ is well defined.

[6] In the whole chapter, all the considered sets are implicitly non-empty.

7. Also known as Bouligand's cone, or the contingent cone [328, Definition 5.1.1].

In the following, we also denote ∂S the boundary of S, and $\operatorname{int} S$ its interior. We use $D_{\mathbf{u}} f$ to denote the directional derivative of the function f in the direction $\mathbf{u}$.

Lemma 10.5

Let h be a nonempty closed convex set, and let $\mathbf{c}, \mathbf{u} \in \mathbb{R}^d$.

- If $\mathbf{c} \in \operatorname{int} h$, then there exists $B > 0$ such that for any $\varepsilon \in]0, B[$ we have

$$l_1(h, \mathbf{c} + \varepsilon \mathbf{u}) - l_1(h, \mathbf{c}) = 0.$$

- If $c \notin h$, then for any $\varepsilon > 0$:

$$l_1(h, \mathbf{c} + \varepsilon \mathbf{u}) - l_1(h, \mathbf{c}) = \varepsilon \frac{(\mathbf{c} - \Pi_h(\mathbf{c}))^T}{\|\mathbf{c} - \Pi_h(\mathbf{c})\|_2} \mathbf{u} + o(\varepsilon)$$

- Otherwise, i.e. when $c \in \partial h$, for any $\varepsilon > 0$:

$$l_1(h, \mathbf{c} + \varepsilon \mathbf{u}) - l_1(h, \mathbf{c}) = \varepsilon \|\mathbf{u} - \Pi_{T_h(\mathbf{c})}(\mathbf{u})\|_2 + o(\varepsilon).$$

Proof:

- On $\operatorname{int} h$, we have $l_1(\cdot, h) = 0$ everywhere. Hence for any $\mathbf{x} \in \operatorname{int} h$, one can find a neighborhood $N(\mathbf{x}) \subset \operatorname{int} h$ on which $l_1(\cdot, h) = 0$.
- Note that $l_1(\mathbf{c}, h) = l_2(\mathbf{c}, h)^{1/2}$, and thus at any $\mathbf{c} \notin h$, the loss $l_1(\cdot, h)$ is differentiable by composition. Using the chain rule, we have

$$\nabla l_1(\mathbf{c}, h) = \frac{1}{2\|\mathbf{c} - \Pi_h(\mathbf{c})\|_2} 2(\mathbf{c} - \Pi_h(\mathbf{c})),$$

  which yields the desired result.
- For $\mathbf{c} \in h$, we have $D_{\mathbf{u}} \Pi_h(\mathbf{c}) = \Pi_{T_h(\mathbf{c})}(\mathbf{u})$ according to [328, Proposition III.5.3.5]. Hence in particular for any $\mathbf{c} \in \partial h$, and for any direction $\mathbf{u} \in \mathbb{R}^d$:

$$\begin{aligned} D_{\mathbf{u}} l_1(\cdot, h) &= \lim_{\lambda \to 0^+} \frac{l_1(\mathbf{c} + \lambda \mathbf{u}, h) - l_1(\mathbf{c}, h)}{\lambda} \\ &= \lim_{\lambda \to 0^+} \frac{\|\mathbf{c} + \lambda \mathbf{u} - \Pi_h(\mathbf{c} + \lambda \mathbf{u})\|_2}{\lambda} \\ &= \lim_{\lambda \to 0^+} \left\| \mathbf{u} - \frac{\Pi_h(\mathbf{c} + \lambda \mathbf{u}) - \mathbf{c}}{\lambda} \right\|_2 \\ &= \left\| \mathbf{u} - \Pi_{T_h(c)}(\mathbf{u}) \right\|_2 \end{aligned}$$

  by composition and continuity of $\|\mathbf{u} - \cdot\|_2$.

We will now use these expressions of the directional derivatives of the l_1 and l_2 losses to derive results on the risk of very simple distributions. This will, in turn, allow us to lower-bound the semi-norms induced by these losses in Section 10.2.

Regularity of the associated risk In the following, we will consider distributions of the form

$$\tau = \delta_{\mathbf{c}} \text{ and } \forall \varepsilon > 0 \, \tau_\varepsilon = \tfrac{1}{2}(\delta_{\mathbf{c}+\varepsilon\mathbf{u}} + \delta_{\mathbf{c}-\varepsilon\mathbf{u}}), \tag{10.3}$$

where $\mathbf{c}, \mathbf{u} \in \mathbb{R}^d$, and look at the difference of their risks when $\varepsilon \to 0$.

Lemma 10.6

Let h be a closed convex set. Let $\mathbf{c}, \mathbf{u} \in \mathbb{R}^d$, and τ, τ_ε defined as in (10.3). Then $\mathcal{R}_2(h, \tau_\varepsilon) - \mathcal{R}_2(h, \tau) = o(\varepsilon)$.

Proof: We have by definition

$$\begin{aligned}
&\mathcal{R}_2(h, \tau_\varepsilon) - \mathcal{R}_2(h, \tau) \\
&= \tfrac{1}{2}(l_2(h, \mathbf{c} + \varepsilon\mathbf{u}) - l_2(h, \mathbf{c})) + \tfrac{1}{2}(l_2(h, \mathbf{c} - \varepsilon\mathbf{u}) - l_2(h, \mathbf{c})) \\
&\overset{\text{(i)}}{=} \tfrac{1}{2}(\varepsilon 2(\mathbf{c} - \Pi_h(\mathbf{c}))^T\mathbf{u}) + \tfrac{1}{2}(\varepsilon 2(\mathbf{c} - \Pi_h(\mathbf{c}))^T(-\mathbf{u}) + o(\varepsilon)) \\
&= o(\varepsilon).
\end{aligned}$$

(i) By Lemma 10.2.

Lemma 10.7

Let h be a closed convex set, $\mathbf{c} \in \partial h$ and $\mathbf{u} \in \mathbb{R}^d$ be chosen such that[8] $\|\mathbf{u} - \Pi_{T_h(\mathbf{c})}(\mathbf{u})\|_2 > 0$. Let τ, τ_ε be defined as in (10.3). Then $\mathcal{R}_1(h, \tau_\varepsilon) - \mathcal{R}_1(h, \tau) = \Theta(\varepsilon)$.

8. We will see when using Lemma 10.7 that this is always possible for a closed convex (e.g. by choosing $\mathbf{u}$ to be the normal vector to a supporting hyperplane for h at $\mathbf{c}$ oriented such that the ray $\mathbf{c} + \mathbb{R}_+\mathbf{u}$ does not intersect h).

Proof: By two direct applications of Lemma 10.5:

$$\begin{aligned}
\mathcal{R}_1(h, \tau_\varepsilon) - \mathcal{R}_1(h, \tau) &= \tfrac{1}{2}(l_1(h, \mathbf{c} + \varepsilon\mathbf{u}) - l_1(h, \mathbf{c})) \\
&\quad + \tfrac{1}{2}(l_1(h, \mathbf{c} + \varepsilon(-\mathbf{u})) - l_1(h, \mathbf{c})) \\
&= \varepsilon\Big(\|\mathbf{u} - \Pi_{T_h(\mathbf{c})}(\mathbf{u})\|_2 + \|\mathbf{u} - \Pi_{T_h(\mathbf{c})}(-\mathbf{u})\|_2\Big) \\
&\quad + o(\varepsilon) \\
&= \Theta(\varepsilon).
\end{aligned}$$

Note that h can perfectly have an empty interior here. We will for instance use this lemma below in the setting where h is a linear subspace, and $\mathbf{u}$ a normal vector to the hyperplane.

10.1.2 Handling unions of convex sets

We now consider the setting where the hypothesis h is chosen in a family of unions of convex sets, i.e. $h = \bigcup_{i=1}^{l} h_i$ with each h_i convex. In that case, the loss can be written $l_p(h, \mathbf{x}) = \min_{1 \le i \le l} l_p(h_i, \mathbf{x})$, and is most often not differentiable even for $p = 2$.

We introduce some useful geometric definitions, and then derive properties of the loss similarly to what has been done before.

Definition 10.8: Let $h = \bigcup_{i=1}^{l} h_i$ be a set of convex sets and d a distance. We define

- For every i, the territory[9] of h_i is

$$\text{terr}(h_i) \triangleq \{\mathbf{x} \in \mathbb{R}^d | \forall j \neq i, d(\mathbf{x}, h_i) < d(\mathbf{x}, h_j)\}.$$

- For every $i \neq j$, the conflict set[10] between h_i and h_j is

$$\text{confl}(h_i, h_j) \triangleq \{\mathbf{x} \in \mathbb{R}^d | d(\mathbf{x}, h_i) = d(\mathbf{x}, h_j)\}.$$

- The conflict set of h is

$$\text{confl}\{(h_1, ..., h_l)\} \triangleq \mathbb{R}^d \setminus \bigcup_{i=1}^{l} \text{terr}(h_i).$$

By an abuse of notation, we write in the following $\text{confl}(h)$ instead of $\text{confl}((h_1, ..., h_l))$, i.e. h refers to both the union or the tuple of the $(h_i)_{1 \leq i \leq l}$ depending on the context. Also, we will alway use the previous definition with the euclidean distance $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$ in the following.

Note that $\text{confl}(h)$ is a (most often strict) subset of $\cup_{i \neq j} \text{confl}(h_i, h_j)$. It corresponds to all the points at which the loss is not differentiable. Let us now illustrate the three concepts from Definition 10.8 on two settings of interest.

Examples:

- When all the $(h_i)_{1 \leq i \leq l}$ are singletons, the territories are the associated (open) Voronoi cells, and $\text{confl}(h)$ the boundary around these Voronoi cells, as illustrated in Figure 10.2.
- If h is a union of two intersecting lines in the two-dimensional plane, then their conflict set is the union of their two bissectors.

Naturally, when h is more complicated, the geometry of the conflict itself can become more complex as well, and is an object of study of its own [329]. Note also that, if the $(h_i)_{1 \leq i \leq l}$ intersect, then their intersection[11] belongs to the conflict set.

Definition 10.9: Let $\mathbf{c} \in \text{confl}(h_1, h_2)$ be such that $\Pi_{h_1}(\mathbf{c}) \neq \Pi_{h_2}(\mathbf{c})$. We say that $\mathbf{u} \in S^{d-1}$ is the direction of conflict at $\mathbf{c}$ from h_1 to h_2 if

$$\mathbf{u} = \frac{\Pi_{h_2}(\mathbf{c}) - \Pi_{h_1}(\mathbf{c})}{\|\Pi_{h_2}(\mathbf{c}) - \Pi_{h_1}(\mathbf{c})\|_2}.$$

A schematic representation is provided in Figure 10.3. Note that the orientation (sign) of $\mathbf{u}$ is imposed here to make the definition unique, but will not play a role in the following given the usage that will be made of the definition. Indeed, we will use directions of conflict to define symmetric distributions; if $\mathbf{c} \in \text{confl}(h_1, h_2)$, then the distribution τ_ε defined as in (10.3) is invariant to the sign of $\mathbf{u}$, and thus using for $\mathbf{u}$ the direction of conflict from h_1 to h_2 or from h_2 to h_1 is equivalent. In the following, $\mathbf{c} + \mathbb{R}_+ \mathbf{u} \triangleq \{\mathbf{c} + \lambda \mathbf{u} | \lambda \in \mathbb{R}_+\}$ denotes the ray starting from $\mathbf{c}$ in the direction $\mathbf{u}$.

We now introduce a geometric construction which will help us to

9. Note that the territories and conflict sets implicitly depend on all the $(h_i)_{1 \leq i \leq l}$ and on the distance d, although we do not reflect it in our notations for conciseness.

10. Sometimes also called "ties".

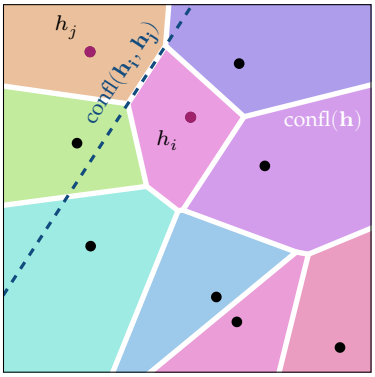


Figure 10.2: Illustration of Definition 10.8 when h is a union of k points. We obtain a Voronoi diagram, whose colored cells correspond to the territories associated to the points $(h_l)_{1 \leq l \leq k}$. The blue dashed line corresponds to the conflict set between h_i and h_j (points in purple), and $\text{confl}(h)$ is represented in white.

11. And all the intersections between pairs (h_i, h_j) with $i \neq j$.

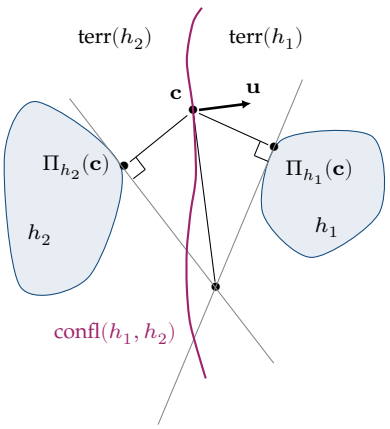


Figure 10.3: Construction of the direction of conflict (here from h_2 to h_1). The purple line is the conflict set. In dimension 2, when h_1, h_2 are closed disjoint convex sets, the direction of conflict $\mathbf{u}$ is a normal of the conflict set at $\mathbf{c}$ [330, Theorem 1].

build counter-examples.

Definition 10.10: We say that a pair $(\mathbf{c}, \mathbf{u}) \in (\mathbb{R}^d)^2$ is contentious for a union of closed convex sets $h = \bigcup_{i=1}^{l} h_i$ if there exists indexes $1 \leq i \neq j \leq l$ and a neighborhood N of $\mathbf{c}$ such that

(i) $\mathbf{c} \in \text{confl}(h) \cap \text{confl}(h_i, h_j)$;

(ii) $\Pi_{h_i}(\mathbf{c}) \neq \Pi_{h_j}(\mathbf{c})$ (which implies $\mathbf{c} \notin h$);

(iii) $\mathbf{u}$ is a direction of conflict at $\mathbf{c}$ (w.l.o.g. from h_j to h_i);

(iv) $(\mathbf{c} + \mathbb{R}_+\mathbf{u}) \cap N \subseteq \overline{\text{terr}(h_i)}$ and $(\mathbf{c} - \mathbb{R}_+\mathbf{u}) \cap N \subseteq \overline{\text{terr}(h_j)}$.

We do not aim at characterizing exactly when such contentious pairs exist, but we will show below that it is always possible to build one for unions of disjoint compact convex sets, as well as for union of subspaces.

Although the formulation of Definition 10.10 might seem a bit cumbersome[12], we expect that contentious pairs exist for most settings of practical interest. For instance in the k-means setting, h is a set of k points, and the conflict set $\text{confl}(h)$ corresponds to the boundary of the k Voronoi cells (cf. Figure 10.2); thus any $\mathbf{c}$ which is in $\text{confl}(h)$ and at the same time in the conflict set of exactly two distinct points[13] can be used to build a contentious pair.

We will briefly discuss later (see two paragraphs below) why the four conditions in Definition 10.10 are required for our needs, but we first review what these conditions imply for the loss when a contentious pair exist, and show how to build a contentious pair in two practical settings.

[12] There might be a simpler reformulation suiting our needs.

[13] We will see below in Figure 10.5 that the situation may be more complex if $\mathbf{c}$ is in multiple conflict sets at the same time.

Implications of the existence of contentious pairs for the loss We have seen in Section 10.1.1 that the loss $l_2(h, \cdot)$ is $\mathcal{C}^1$ for a convex set h, and that we always have $\mathcal{R}_2(h, \tau_\varepsilon) - \mathcal{R}_2(h, \tau) = o(\varepsilon)$ as a consequence (cf. Lemma 10.6). When considering unions of convex sets, the loss is still $\mathcal{C}^1$ in each "territory", however we can find points on the conflict set where the loss is not differentiable when there exists $(\mathbf{c}, \mathbf{u})$ satisfying Definition 10.10. The same holds for the l_1 loss (albeit we already established in this case that $l_1(h, \cdot)$ is non differentiable on ∂h for a single convex set h).

Lemma 10.11

Let h be a union of $l \geq 2$ convex sets admitting a contentious pair $(\mathbf{c}, \mathbf{u})$, and τ, τ_ε be defined as in (10.3). Then $\mathcal{R}(h, \tau_\varepsilon) - \mathcal{R}(h, \tau) = \Theta(\varepsilon)$ for both l_1 and l_2.

Proof: By definition, h can be written $h = \bigcup_{i=1}^{l} h_i$. By Definition 10.10, there are indexes $i \neq j$ such that $\mathbf{u}$ is the direction of conflict at $\mathbf{c}$ from j to i, and a neighborhood N of $\mathbf{c}$ such that $(\mathbf{c} + \mathbb{R}_+\mathbf{u}) \cap N \subseteq \text{terr}\, h_i$ and $(\mathbf{c} - \mathbb{R}_+\mathbf{u}) \cap N \subseteq \text{terr}\, h_j$. Hence for $\varepsilon > 0$ small enough $l(h, \mathbf{c} + \varepsilon\mathbf{u}) = l(h_i, \mathbf{c} + \varepsilon\mathbf{u})$ and

$l(h, \mathbf{c} - \varepsilon\mathbf{u}) = l(h_j, \mathbf{c} - \varepsilon\mathbf{u})$. Moreover, by continuity of the loss $l(h, \mathbf{c}) = l(h_i, \mathbf{c})$ as $\mathbf{c} \in \overline{\text{terr}(h_i)}$, and the same holds for j.

Thus we have (for both $l = l_1$ and $l = l_2$):

$$\mathcal{R}(h, \tau_\varepsilon) - \mathcal{R}(h, \tau) = \tfrac{1}{2}(l(h_i, \mathbf{c} + \varepsilon\mathbf{u}) - l(h_i, \mathbf{c})) + \tfrac{1}{2}(l(h_j, \mathbf{c} + \varepsilon(-\mathbf{u})) - l(h_j, \mathbf{c})).$$

This translates, first for the l_1 loss by Lemma 10.5 to:

$$\mathcal{R}_1(h, \tau_\varepsilon) - \mathcal{R}_1(h, \tau) = \tfrac{1}{2}\varepsilon \left(\frac{\mathbf{c} - \Pi_{h_i}(\mathbf{c})}{\|\mathbf{c} - \Pi_{h_i}(\mathbf{c})\|_2} - \frac{\mathbf{c} - \Pi_{h_j}(\mathbf{c})}{\|\mathbf{c} - \Pi_{h_j}(\mathbf{c})\|_2} \right)^T \mathbf{u} + o(\varepsilon).$$

And for the l_2 loss by Lemma 10.2:

$$\mathcal{R}_2(h, \tau_\varepsilon) - \mathcal{R}_2(h, \tau) = \varepsilon \big(\Pi_{h_j}(\mathbf{c}) - \Pi_{h_i}(\mathbf{c}) \big)^T \mathbf{u} + o(\varepsilon).$$

By definition of the direction of conflict, and because $\|\mathbf{c} - \Pi_{h_i}(\mathbf{c})\|_2 = \|\mathbf{c} - \Pi_{h_j}(\mathbf{c})\|_2$, we have both $\big(\Pi_{h_j}(\mathbf{c}) - \Pi_{h_i}(\mathbf{c}) \big)^T \mathbf{u} \neq 0$ and $\left(\frac{\mathbf{c} - \Pi_{h_i}(\mathbf{c})}{\|\mathbf{c} - \Pi_{h_i}(\mathbf{c})\|_2} - \frac{\mathbf{c} - \Pi_{h_j}(\mathbf{c})}{\|\mathbf{c} - \Pi_{h_j}(\mathbf{c})\|_2} \right)^T \mathbf{u} \neq 0$.

Proving the existence of contentious situations In dimension 2, the existence of a tuple $(\mathbf{c}, \mathbf{u})$ satisfying Definition 10.10 can be shown easily in some contexts using the fact that the conflict set of two distinct convex sets h_1, h_2 is differentiable, and that the direction of conflict $\mathbf{u} = (\Pi_{h_1}(\mathbf{c}) - \Pi_{h_2}(\mathbf{c}))/\|\Pi_{h_1}(\mathbf{c}) - \Pi_{h_2}(\mathbf{c})\|_2$ is precisely the normal of the conflict set at $\mathbf{c}$ [330, Theorem 1][14] (cf. Figure 10.3).

We provide two lemmas for the settings where h is a set of points, or a set of subspaces, which hold in any dimension.

Lemma 10.12

If the $(h_i)_{1 \leq i \leq l}$ are pairwise disjoint compact convex sets, then $h = \bigcup_{1 \leq i \leq l} h_i$ admits a contentious pair.

Proof: Let $D = \min_{p \neq q} \inf_{\mathbf{x} \in h_p, \mathbf{y} \in h_q} \|\mathbf{x} - \mathbf{y}\|_2$. Note that $D > 0$ as the $(h_p)_{1 \leq p \leq l}$ are pairwise disjoint. One can find $\mathbf{x} \in h_i, \mathbf{y} \in h_j$ such that $\|\mathbf{x} - \mathbf{y}\|_2 = D$, as D is the minimizer of a continuous function over $\bigcup_{p \neq q} h_p - h_q$, which is compact as a finite union of differences of compact sets (convexity is not needed here). Let i, j be the (unique by disjointness) indexes satisfying $\mathbf{x} \in h_i$, $\mathbf{y} \in h_j$.

Define $\mathbf{c} = (\mathbf{x} + \mathbf{y})/2$. We have $\Pi_{h_i}(\mathbf{c}) = \mathbf{x}$ and $\Pi_{h_j}(\mathbf{c}) = \mathbf{y}$ (otherwise if $\Pi_{h_i}(\mathbf{c}) = \mathbf{w} \neq \mathbf{x}$, we would have $\|\mathbf{w} - \mathbf{c}\|_2 < \|\mathbf{x} - \mathbf{c}\|_2 = \|(\mathbf{x} - \mathbf{y})/2\|_2 = D/2$, which would imply $\|\mathbf{w} - \mathbf{y}\|_2 \leq \|\mathbf{w} - \mathbf{c}\|_2 + \|\mathbf{c} - \mathbf{y}\|_2 < D$ and is in contradiction with the definition of $\mathbf{x}$ and $\mathbf{y}$). As a consequence, $\|\mathbf{c} - \Pi_{h_i}(\mathbf{c})\|_2 = \|\mathbf{c} - \Pi_{h_j}(\mathbf{c})\|_2 = D/2$ and $\mathbf{c} \in \text{confl}(h_i, h_j)$. Furthermore we have $\mathbf{c} \in \text{confl}(h)$, otherwise there would be $k \in [\![1, l]\!]$ satisfying $k \neq i, k \neq j$ such

[14] We expect that this result can be useful as well in higher dimension by considering the 2d plane containing $\mathbf{c}, \Pi_{h_1}(\mathbf{c}), \Pi_{h_2}(\mathbf{c})$. However we could not find any generalized result regarding the regularity of the conflict set in this case.

that $\mathbf{c} \in \text{terr}(h_k)$, which would contradict the definition of D.

Now if we define $\mathbf{u} \triangleq (\mathbf{y}-\mathbf{x})/\|\mathbf{y}-\mathbf{x}\|_2$ then $(\mathbf{c},\mathbf{u})$ satisfies Definition 10.10. Indeed, we have $\alpha\mathbf{c}+(1-\alpha)\mathbf{x} \in \text{terr}(h_i)$ for any $0 \leq \alpha < 1$ using again the definition of D (and the same holds for $\alpha\mathbf{z}+(1-\alpha)\mathbf{y}$ and $\text{terr}(h_j)$ by symmetry). Hence one can choose for instance the neighborhood N of $\mathbf{c}$ appearing in Definition 10.10 to be the ball of radius $R = \|(\mathbf{x}-\mathbf{y})/2\|_2$ centered in $\mathbf{c}$.

Lemma 10.13

Let $h = \bigcup_{i=1}^{l} h_i$, where the $(h_i)_{1\leq i\leq l}$ are distinct linear subspaces of $\mathbb{R}^d$. Then h admits a contentious pair.

Proof: The proof is similar to the one of Lemma 10.12, however we need to rely on principal angles between subspaces to measure the "closest" subspaces. For any $p \neq q \in [\![1,l]\!]$, defining $d_p = \dim(h_p), d_q \triangleq \dim(h_q)$ and assuming $d_p \leq d_q$ w.l.o.g., we denote $0 \leq \theta_1^{pq} \leq \ldots \leq \theta_{d_p}^{pq} \leq \pi/2$ the principal angles associated to h_p and h_q. As the subspaces are assumed to be distinct, there is for each $p \neq q$ a lowest index i_{pq} such that $\theta_{\min}^{pq} \triangleq \theta_{i_{pq}}^{pq} > 0$.

Let i,j be chosen such that $\theta_{\min}^{ij} = \min_{p\neq q} \theta_{\min}^{pq}$. Then one can find $\mathbf{x} \in h_p \cap S^{d-1}$ and $\mathbf{y} \in h_q \cap S^{d-1}$ such that $\theta_{\min}^{ij} = \arccos(|\langle \mathbf{x},\mathbf{y}\rangle|)$. Defining $\mathbf{c} = \cos(\theta_{\min}^{ij}/2)^{-1}(\mathbf{x}+\mathbf{y})/\|\mathbf{x}+\mathbf{y}\|$ and $\mathbf{u} = (\mathbf{x}-\mathbf{y})/\|\mathbf{x}-\mathbf{y}\|_2$, the pair $(\mathbf{c},\mathbf{u})$ satisfies Definition 10.10. Indeed:

- $\Pi_{h_i}(\mathbf{c}) = \mathbf{x}$ ($\mathbf{x}$ is the orthogonal projection of $\mathbf{c}$ onto the line containing $\mathbf{0}$ and $\mathbf{x}$ by construction of $\mathbf{c}$; it is however also the orthogonal projection onto h_i, as there would otherwise be contradiction with the minimality of $\theta_{\min}^{ij}$) and $\Pi_{h_j}(\mathbf{c}) = \mathbf{y}$ (by symmetry);
- this implies $\mathbf{c} \in \text{confl}(h_i,h_j)$, as $\|\mathbf{c}-\mathbf{x}\|_2 = \|\mathbf{c}-\mathbf{y}\|_2$;
- we also have $\mathbf{c} \in \text{confl}(h)$ (if this does not hold, then it means that $\mathbf{c}$ is in the territory of some h_k, which would contradict the minimality of $\theta_{\min}^{ij}$);
- still using the minimality of $\theta_{\min}^{ij}$, we have that for every $0 < \varepsilon < \|\mathbf{c}\|_2 \tan(\theta_{\min}^{ij}/2)$, $\mathbf{x}+\varepsilon\mathbf{u} \in \text{terr}(h_i)$, and a similar property holds for $\text{terr}(h_j)$ by symmetry.

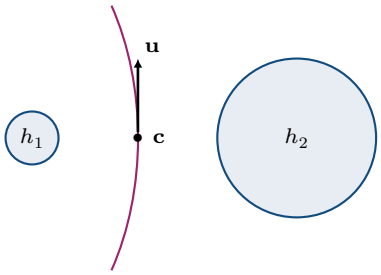


Figure 10.4: Here $\mathbf{c} \in \text{confl}(h_1,h_2)$, however $\mathbf{u}$ is not a direction of conflict and $\mathbf{c}+\mathbb{R}_+\mathbf{u} \subseteq \overline{\text{terr}(h_2)}$.

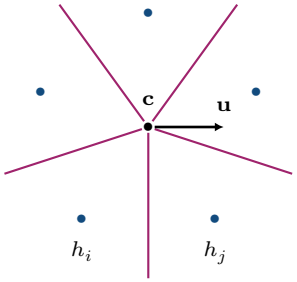


Figure 10.5: Here $\mathbf{c} \in \text{confl}(h) \cap \text{confl}(h_1,h_2)$ and is a direction of conflict, but does not satisfy point (iv) of Definition 10.10 (at least not for the choice of i,j represented). Note that on this example, it is however possible to build a contentious pair.

Intuition behind Definition 10.10 We provide in figs. 10.4 and 10.5 two examples illustrating why the four conditions of Definition 10.10 are required in order to design distributions τ, τ_ε for which $\mathcal{R}(h,\tau_\varepsilon) - \mathcal{R}(h,\tau) = \Theta(\varepsilon)$ (as we did in Lemma 10.11). For instance, the vector $\mathbf{u}$ depicted in Figure 10.4 will not allow us to derive a risk difference of the order of $\Theta(\varepsilon)$: indeed $\mathbf{c}+\mathbb{R}\mathbf{u}$ is a subset of $\overline{\text{terr}(h_2)}$ and thus the loss is differentiable on this line. On Figure 10.5, it is possible to build a contentious pair, however $\mathbf{c}$ and $\mathbf{u}$ must be wisely chosen: we depict an example which is *not* a contentious pair, as $\mathbf{c}$ and $\mathbf{u}$ satisfy properties

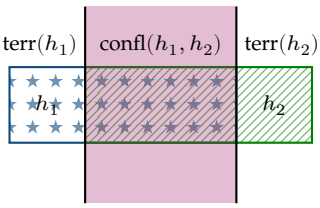


Figure 10.6: Here the conflict set has a non-empty interior as h_1 and h_2 are overlapping, and there exists no contentious pair.

(i)-(iii) of Definition 10.10 for h_i, h_j defined on the picture but $\mathbf{c} + \mathbb{R}_+\mathbf{u}$ is neither contained in the territory of h_i, nor in the territory of h_j. Note that, even for simple convex sets, existence of contentious pairs is not trivial when allowing overlapping sets as shown in Figure 10.6: in this example, the boundaries of the two territories have no intersection and the conflict set has a non-empty interior.

10.2 Implications for the lower restricted isometry property

The results provided so far in Section 10.1 focused on the difference of the risks for distributions of the kind τ_ε, τ.

We will now focus on the other semi-norm $\|\mathcal{A}(\cdot)\|_2$ (associated to the sketching operator) which appears in the expression of the LRIP, and build explicit counter-examples $\tau_\varepsilon, \tau \in \mathfrak{S}$ for which $\|\tau_\varepsilon - \tau\|_{\Delta\mathcal{L}(\mathcal{H})} = \Theta(\varepsilon)$ and $\|\mathcal{A}(\tau_\varepsilon - \tau)\|_2 = o(\varepsilon)$, thus proving that the LRIP (10.1) cannot hold for the considered models.

10.2.1 Regularity of the sketching operator

Let $\mathcal{C}^2(\mathbb{R}^d, \mathbb{R}^m)$ denote the class of twice differentiable functions from $\mathbb{R}^d$ to $\mathbb{R}^m$ with continuous first and second derivatives.

Lemma 10.14

Let $\Phi \in \mathcal{C}^2(\mathbb{R}^d, \mathbb{R}^m)$, and $\mathcal{A} : \pi \mapsto \mathbf{E}_{\mathbf{x}\sim\pi}\Phi(\mathbf{x})$ the associated sketching operator. Let $\mathbf{c}, \mathbf{u} \in \mathbb{R}^d$, and τ, τ_ε defined as in (10.3).

Then $\|\mathcal{A}(\tau_\varepsilon - \tau)\|_2 = o(\varepsilon)$ when $\varepsilon \to 0$.

Proof: If Φ is $\mathcal{C}^2$, then for any $i \in [\![1, m]\!]$, denoting $\nabla\Phi_j$ and H_{Φ_j} the gradient and the hessian of Φ_j we have

$$\begin{aligned}
\mathcal{A}(\tau_\varepsilon)_j - \mathcal{A}(\tau)_j &= \tfrac{1}{2}\left[\Phi(\mathbf{c}+\varepsilon\mathbf{u})_j + \Phi(\mathbf{c}-\varepsilon\mathbf{u})_j\right] - \Phi(\mathbf{c})_j \\
&= \tfrac{1}{2}\Big[(\Phi(\mathbf{c})_j + \varepsilon\nabla\Phi_j(\mathbf{c})^T\mathbf{u} + \tfrac{1}{2}\varepsilon^2\mathbf{u}^T H_{\Phi_j}(\mathbf{c})\mathbf{u}) \\
&\qquad + (\Phi(\mathbf{c})_j - \varepsilon\nabla\Phi_j(\mathbf{c})^T\mathbf{u} + \tfrac{1}{2}\varepsilon^2\mathbf{u}^T H_{\Phi_j}(\mathbf{c})\mathbf{u})\Big] \\
&\qquad - \Phi(\mathbf{c})_j + o(\varepsilon^2) \\
&= \tfrac{1}{2}\varepsilon^2\mathbf{u}^T H_{\Phi_j}(\mathbf{c})\mathbf{u} + o(\varepsilon^2)
\end{aligned}$$

Hence when $\varepsilon \to 0$, $\mathcal{A}(\tau_\varepsilon)_j - \mathcal{A}(\tau)_j = o(\varepsilon)$, and more precisely $\mathcal{A}(\tau_\varepsilon)_j - \mathcal{A}(\tau)_j = \Theta(\varepsilon^2)$ whenever $\mathbf{u}^T H_{\Phi_j}(\mathbf{c})\mathbf{u}$ is not null. This gives the claimed result for $\|\mathcal{A}(\tau_\varepsilon) - \mathcal{A}(\tau)\|_2$.

10.2.2 Implications for the LRIP

We now try to see the implications of the previous results for the LRIP, assuming a $\mathcal{C}^2$ feature map. In the following, we denote

$$\mathfrak{S}_0(\mathcal{H}) = \{\pi | \exists h \in \mathcal{H} \text{ s.t. } \mathcal{R}(h, \pi) = 0\},$$

which is a natural choice of model set for compression-type tasks.
Examples:

- For k-means and k-medians, if $\mathcal{H}$ is the set of all sets of at most k points, then $\mathfrak{S}_0(\mathcal{H})$ is the set of mixtures of at most k Diracs.
- For PCA, if $\mathcal{H}$ is the set of at most k-dimensional linear subspaces, then $\mathfrak{S}_0(\mathcal{H})$ is the set of distributions supported on k-dimensional linear subspaces, i.e. centered distributions with covariance matrices of rank lower or equal to k.

Compression-type tasks In addition to the k-means, k-medians and PCA problems, we will consider the non-negative matrix (NMF) factorization task and subspace clustering.

Definition 10.15 (NMF): Non-negative matrix factorization with rank k of a dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$ consists in finding $\mathbf{W} \in \mathbb{R}_+^{d\times k}, \mathbf{Y} \in \mathbb{R}_+^{k\times n}$ minimizing $\|\mathbf{X} - \mathbf{W}\mathbf{Y}\|_F$, i.e. in finding a positive cone $h(\mathbf{W}) \triangleq \{\mathbf{W}\mathbf{y} | \mathbf{y} > 0\}$ for $\mathbf{W} \in \mathbb{R}_+^{d\times k}$ minimizing the risk induced by the loss[15] $l_2(h(\mathbf{W}), \mathbf{x})$ for the empirical distribution of $\mathbf{X}$.

15. Cf. (10.2).

We refer to the problem obtained by replacing $l_2(h, \pi_{\mathbf{X}})$ by $l_1(h, \pi_{\mathbf{X}})$ in Definition 10.15 as the robust NMF (RNMF) problem. Similarly, the PCA loss from (1.2) can be modified to $l(h, \mathbf{x}) \triangleq \|\mathbf{x} - \Pi_h \mathbf{x}\|_2$ (where h is a subspace), and we call the problem induced by this variant of the loss the robust PCA (RPCA) problem. Note that other definitions of "robustness" exist; we use this terminology by convenience here, but the reader should keep in mind that this name can refer to different problems in the literature.

We also provide the following definition for unions of subspaces.

Definition 10.16 (Subspace clustering): Subspace clustering (SC) (resp. robust subspace clustering, RSC) of a dataset $\mathbf{X}$ with empirical probability distribution $\pi_{\mathbf{X}}$ consists in finding a union h of k linear subspaces minimizing the risk $\mathcal{R}_2(h, \pi_{\mathbf{X}})$ (resp. $\mathcal{R}_1(h, \pi_{\mathbf{X}})$).

Note that we restrict ourselves to linear subspaces here for conciseness, but subspace clustering can also be performed using unions of affine subspaces in the literature. Restrictions on the dimensions of the different subspaces can also be considered.

Projection onto a single convex set For PCA, NMF and their robust variants (RPCA, RNMF), the loss takes the form (10.2), with h being a convex set – a subspace for (R)PCA, a cone for (R)NMF.

As a consequence, for any construction of τ_ε, τ we have $\|\tau_\varepsilon - \tau\|_{\Delta\mathcal{L}(\mathcal{H})} = o(\varepsilon)$ by Lemma 10.6 and $\|\mathcal{A}(\tau_\varepsilon) - \mathcal{A}(\tau)\|_2 = o(\varepsilon)$ by Lemma 10.14 for a $\mathcal{C}^2$ feature map, which does not allow to build a counter-example to the LRIP (10.1). This is naturally not surprising for PCA, as the LRIP is known to hold in this case [5]. However, this is reasonably encouraging for future work on NMF.

Regarding RPCA and RNMF, the non-differentiability of the loss on the boundary ∂h of the hypothesis[16] allows us to build concrete counter-examples for the LRIP.

[16] Cf Lemma 10.7.

Lemma 10.17

Let $k \in [\![1, d]\!]$, and $\mathcal{H}$ be the set of all subspaces of dimension k in $\mathbb{R}^d$. Then the LRIP (10.1) cannot be satisfied for the l_1 loss on $\mathfrak{S} = \mathfrak{S}_0(\mathcal{H})$ for any $\Phi \in \mathcal{C}^2(\mathbb{R}^d, \mathbb{R}^m)$.

Lemma 10.18

Let $k > 1$. Let $\mathcal{H}$ be the set of all positive cones spanned by k points in $\mathbb{R}^d_+$. Then the LRIP (10.1) cannot be satisfied for the l_1 loss on $\mathfrak{S} = \mathfrak{S}_0(\mathcal{H})$ for any $\Phi \in \mathcal{C}^2(\mathbb{R}^d, \mathbb{R}^m)$.

Proof (Proof of lemmas 10.17 and 10.18): Let $h \in \mathcal{H}$, and $\mathbf{c} \in \partial h$. Note that $\tau \in \mathfrak{S}$. Let P be a supporting hyperplane[17] for h at $\mathbf{c}$ (which exists by convexity of h), and take $\mathbf{u}$ to be normal to P, and oriented such that $\mathbf{u} \notin T_h(\mathbf{c})$ (which is possible by definition of a supporting hyperplane). By Lemma 10.7, we have $\mathcal{R}_1(h, \tau_\varepsilon) - \mathcal{R}_1(h, \tau) = \Theta(\varepsilon)$. Let h_2 be any element from $\mathcal{H}$ such that $\mathbf{c} \notin \partial h_2$ (which is always possible for the considered $\mathcal{H}$). Then $\mathcal{R}_1(h, \tau_\varepsilon) - \mathcal{R}_1(h, \tau) = o(\varepsilon)$ by applying Lemma 10.5 (case 1 or 2).

As a consequence, $\|\tau_\varepsilon - \tau\|_{\Delta\mathcal{L}(\mathcal{H})} \geq \Theta(\varepsilon)$ as soon as $\tau_\varepsilon \in \mathfrak{S}$, which holds as soon as $k > 1$ for the model of positive cones, and always holds for the model of subspaces (when $k > 2$ there is always a 2d linear subspace containing $\mathbf{c} + \varepsilon\mathbf{u}$ and $\mathbf{c} - \varepsilon\mathbf{u}$, but even for $k = 1$ we can choose w.l.o.g. $\mathbf{c} = \mathbf{0}$ above).

We have $\|\mathcal{A}(\tau_\varepsilon - \tau)\|_2 = o(\varepsilon)$ by Lemma 10.14 using the fact that Φ is $\mathcal{C}^2$, which concludes the proof by taking $\varepsilon \to 0$.

[17] A hyperplane P is a supporting hyperplane for h if h is entirely contained in one of the two (closed) half-spaces defined by P, and $P \cap \partial h \neq \emptyset$.

Clustering We now consider the setting of k-means and k-medians clustering. These two problems are compression-type tasks, with h being a union of points in $\mathbb{R}^d$, and $p = 2$ for k-means, $p = 1$ for k-medians.

Lemma 10.19

Let $k > 1$. Let $\mathcal{H}$ be the set of all sets of k points in $\mathbb{R}^d$. Then the LRIP (10.1) cannot be satisfied for both l_1 and l_2 losses on $\mathfrak{S} = \mathfrak{S}_0(\mathcal{H})$ for any $\Phi \in \mathcal{C}^2(\mathbb{R}^d, \mathbb{R}^m)$.

Proof: Let $h \in \mathcal{H}$ be a set of k points. Let $(\mathbf{c}, \mathbf{u})$ be a contentious pair for h, which exists by Lemma 10.12, and τ, τ_ε defined as in (10.3). We have $\mathcal{R}(h, \tau_\varepsilon) - \mathcal{R}(h, \tau) = \Theta(\varepsilon)$ by Lemma 10.11 for both l_1 and l_2. Furthermore, one can easily construct another $h_2 \in \mathcal{H}$ such that $\mathbf{c} \notin \mathrm{confl}(h_2)$, and hence for which $\mathcal{R}(h, \tau_\varepsilon) - \mathcal{R}(h, \tau) = o(\varepsilon)$ by Lemma 10.5. As $k > 1$, both τ_ε, τ belong to $\mathcal{H}$

and thus $\|\tau_\varepsilon - \tau\|_{\Delta\mathcal{L}(\mathcal{H})} \geq \Theta(\varepsilon)$. This yield the desired result by Lemma 10.14 as Φ is assumed to be $\mathcal{C}^2$.

This result was already observed in [6, Lemma 3.4]. It is not contradictory with existing guarantees, which established the LRIP for a *restricted* version of the model set. In particular, a separation assumption between the different points is introduced, which makes the counter-example used in Lemma 10.19 impossible as τ_ε does not belong to the model anymore for ε small enough.

Unions of subspaces We consider now the setting where h is a union of linear subspaces. We provide a first impossibility result for an unrestricted model set, and then a second one one for the specific case of union of lines with additional restrictions (minimal angular separation).

Lemma 10.20

Let $\mathcal{H}$ be a model which contains at least the set of all unions of $l = 2$ distinct subspaces of dimension $d' < d$ in $\mathbb{R}^d$. Then the LRIP (10.1) cannot be satisfied on $\mathfrak{S} = \mathfrak{S}_0(\mathcal{H})$ for any $\Phi \in \mathcal{C}^2(\mathbb{R}^d, \mathbb{R}^m)$. This holds for both l_1 and l_2 losses.

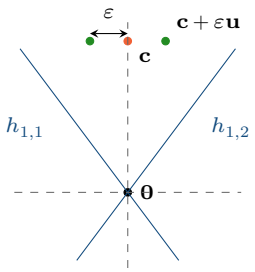


Figure 10.7: The hypothesis h is in blue, and the conflict set in dashed grey. The plane h_2 is the plane of the picture.

Proof: Let $h_1 \in \mathcal{H}$ be any union of two subspaces $h_{1,1}$ and $h_{1,2}$. By definition of $\mathfrak{S}_0(\mathcal{H})$, $h_{1,1}$ and $h_{1,2}$ are distinct. Hence by Lemma 10.13, h admits a contentious pair $(\mathbf{c}, \mathbf{u})$. Let τ, τ_ε defined as in (10.3) (and as depicted in Figure 10.7). Both τ and τ_ε are in $\mathfrak{S}_0(\mathcal{H})$, as $\mathbf{c}$ belongs to a line (and any broader subspace containing $\mathrm{span}(\{\mathbf{c}\})$; there is at least one such subspace in $\mathfrak{S}$ by definition), and $\{\mathbf{c}+\varepsilon\mathbf{u}\} \cup \{\mathbf{c}-\varepsilon\mathbf{u}\}$ to a union of two lines (and any subspace containing $\mathrm{span}(\{\mathbf{c}+\varepsilon\mathbf{u}\}) \cup \mathrm{span}(\{\mathbf{c}-\varepsilon\mathbf{u}\})$, same remark).

By Lemma 10.11 we have $\mathcal{R}(h_1, \tau_\varepsilon) - \mathcal{R}(h_1, \tau) = \Theta(\varepsilon)$. Let $h_2 \in \mathcal{H}$ be the two-dimensional subspace containing $\mathbf{0}, \mathbf{c}, \mathbf{c}+\mathbf{u}$ (and $\mathbf{c}-\mathbf{u}$). Then $\mathcal{R}(h_2, \tau) = \mathcal{R}(h_2, \tau_\varepsilon) = 0$ for any $\varepsilon > 0$. Hence

$$\begin{aligned} \|\tau_\varepsilon - \tau\|_{\Delta\mathcal{L}(\mathcal{H})} &\geq (\mathcal{R}(h_1, \tau_\varepsilon) - \mathcal{R}(h_1, \tau)) - (\mathcal{R}(h_2, \tau_\varepsilon) - \mathcal{R}(h_2, \tau)) \\ &= \Theta(\varepsilon) \end{aligned} \tag{10.4}$$

when $\varepsilon \to 0$. This concludes the proof using $\|\mathcal{A}(\tau_\varepsilon - \tau)\|_2 = o(\varepsilon)$ by Lemma 10.14.

In the proof of Lemma 10.20, τ_ε belongs to the model set $\mathfrak{S} = \mathfrak{S}_0(\mathcal{H})$ for any $\varepsilon > 0$ as it is contained in a union of two lines, or alternatively in the two-dimensional space h_2.

Hence, one way to get a model for which such counter-examples cannot be built is to restrict the model to the set of unions of lines (and not subspaces) which are separated by a minimum angle. Interestingly, this is not sufficient for the l_1 loss as shown in the next lemma.

Lemma 10.21

Let $\mathcal{H}$ be the set of all unions of $l = 2$ lines which are separated by an angle greater than 2α (with $0 < \alpha < \pi/2$). Then the LRIP (10.1) cannot be satisfied on $\mathfrak{S} = \mathfrak{S}_0(\mathcal{H})$ when using the l_1 loss for any $\Phi \in \mathcal{C}^2(\mathbb{R}^d, \mathbb{R}^m)$.

Proof: Let $h_1 = h_{1,1} \cup h_{1,2}$ be an union of two lines separated by an angle 2β with $\pi > 2\beta > 2\alpha$, and P be the 2d plane containing h_1. Let h_2 be the bissector of h_1. Let h_3 be a union of two lines in P with also have h_2 as bissector, but separated by an angle angle 2γ with $\gamma \in]\alpha, \beta[$, as depicted in Figure 10.8. Let $\mathbf{u}$ be the normal vector to the bissector in P, let $\mathbf{c} = \delta u_b$ where $\mathbf{v} \in h_2$ has unit norm, and let $\varepsilon(\delta) = \delta \tan(\gamma)$ so that $\mathbf{c} - \varepsilon\mathbf{u}, \mathbf{c} + \varepsilon\mathbf{u} \in h_3$. Let τ, τ_ε defined as in (10.3) (with now $\mathbf{c}$ being a function of δ), which are thus in the model set for any value of δ.

Note that $\|\mathbf{c} \pm \varepsilon(\delta)\mathbf{u}\|_2 = \delta/\cos(\gamma)$, and we have:

$$\begin{aligned}
\mathcal{R}(h_1, \tau_\varepsilon) &= \frac{\delta}{\cos(\gamma)} \sin(\beta - \gamma) = \delta(\sin(\beta) - \cos(\beta)\tan(\gamma)) \\
\mathcal{R}(h_1, \tau) &= (\delta \sin(\beta)) \\
\mathcal{R}(h_2, \tau_\varepsilon) &= (\varepsilon(\delta)) = (\delta \tan(\gamma)) \\
\mathcal{R}(h_2, \tau) &= 0
\end{aligned}$$

and thus $\|\tau_\varepsilon - \tau\|_{\Delta\mathcal{L}(\mathcal{H})} \geq \Theta(\delta)$

Assuming the feature map is C^2, we have using a variant of Lemma 10.14:

$$\begin{aligned}
\mathcal{A}(\tau_\varepsilon)_j - \mathcal{A}(\tau)_j &= \tfrac{1}{2}(\Phi(\delta\mathbf{v} + \varepsilon(\delta)\mathbf{u})_j + \Phi(\delta\mathbf{v} - \varepsilon(\delta)\mathbf{u})_j) - \Phi(\delta\mathbf{v})_j \\
&= \tfrac{1}{2}((\Phi(\mathbf{0})_j + \nabla\Phi_j(\mathbf{0})^T(\delta\mathbf{v} + \varepsilon(\delta)\mathbf{u}) \\
&\quad + \tfrac{1}{2}(\delta\mathbf{v} + \varepsilon(\delta)\mathbf{u})^T H_{\Phi_j}(\mathbf{0})(\delta\mathbf{v} + \varepsilon(\delta)\mathbf{u}) \\
&\quad + (\Phi(\mathbf{0})_j + \nabla\Phi_j(\mathbf{0})^T(\delta\mathbf{v} - \varepsilon(\delta)\mathbf{u}) \\
&\quad + \tfrac{1}{2}(\delta\mathbf{v} - \varepsilon(\delta)\mathbf{u})^T H_{\Phi_j}(\mathbf{0})(\delta\mathbf{v} - \varepsilon(\delta)\mathbf{u})) \\
&\quad - (\Phi(\mathbf{0}) + \nabla\Phi_j(\mathbf{0})^T(\delta\mathbf{v}) + (\delta\mathbf{v})^T H_{\Phi_j}(\mathbf{0})(\delta\mathbf{v})) \\
&\quad + o(\delta^2) \\
&= \tfrac{1}{2}(\delta\tan(\gamma))^2 \mathbf{u}^T H_{\Phi_j}(\mathbf{0})\mathbf{u} + o(\delta^2) \\
&= O(\delta^2)
\end{aligned}$$

This concludes the proof, as for any vector $\mathbf{u}$, one can choose at the beginning of the proof a union h_1 such that $\mathbf{u}$ is a normal vector to its bissector (as $\mathfrak{S}$ contains all unions of separated lines by definition).

This suggest that we need to exclude a neighborhood of $\mathbf{0}$ as well in the considered model in order to avoid such problems.

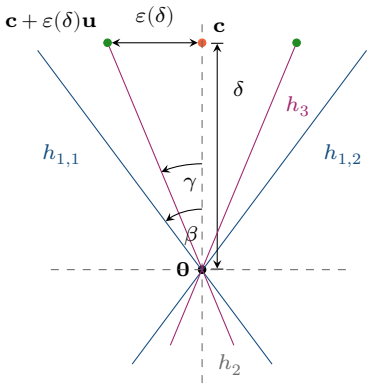


Figure 10.8: Construction of the counter-example used in Lemma 10.21. h_1 in blue, its conflict set is in dashed grey (and contains h_2), and h_3 is in purple.

10.3 Summary

Task	Nature of h	Loss (and its properties as a function of $\mathbf{x}$)			Notes
		Expression	Convex?	$\mathcal{C}^1$?	
PCA	Subspace (parametrized by $\mathbf{U}$)	$\|\mathbf{x}-\mathbf{U}\mathbf{U}^T\mathbf{x}\|^2$	✓	✓	The LRIP holds [5].
NMF	Cone h (spanned by the columns of $\mathbf{W}>0$)	$\min_{\mathbf{y}>0}\|\mathbf{x}-\mathbf{W}\mathbf{y}\|^2$	✓	✓	No counter-example.
RPCA	Subspace (parametrized by $\mathbf{U}$)	$\|\mathbf{x}-\mathbf{U}\mathbf{U}^T\mathbf{x}\|_2$	✓	✗	No LRIP by Lemma 10.17
RNMF	Cone h (spanned by the columns of $\mathbf{W}>0$)	$\min_{\mathbf{y}>0}\|\mathbf{x}-\mathbf{W}\mathbf{y}\|$	✓	✗	No LRIP by Lemma 10.18 provided $k>1$
k-means	Set of points $\{\mathbf{c}_1,...,\mathbf{c}_k\}$	$\min_{1\leq j\leq k}\|\mathbf{x}-\mathbf{c}_j\|_2^2$	✗	✗	No LRIP by Lemma 10.19 assuming $k>1$ (LRIP with separation assumption [5]).
k-medians	Set of points $\{\mathbf{c}_1,...,\mathbf{c}_k\}$	$\min_{1\leq j\leq k}\|\mathbf{x}-\mathbf{c}_j\|_2^2$	✗	✗	No LRIP by Lemma 10.19 assuming $k>1$ (LRIP with separation assumption [5]).
SC ($l>1$)	Union of subspaces paramerized by $\{\mathbf{U}_1,...,\mathbf{U}_l\}$	$\min_{1\leq i\leq l}\|\mathbf{x}-\mathbf{U}_i\mathbf{U}_i^T\mathbf{x}\|_2^2$	✗	✗	No LRIP by Lemma 10.20. A separation assumption might help.
RSC ($l>1$)	Union of subspaces paramerized by $\{\mathbf{U}_1,...,\mathbf{U}_l\}$	$\min_{1\leq i\leq l}\|\mathbf{x}-\mathbf{U}_i\mathbf{U}_i^T\mathbf{x}\|_2^2$	✗	✗	No LRIP by Lemma 10.20. Same for separated lines by Lemma 10.21.

Table 10.1: Summary of our results for different compression-type tasks. We implictly consider the "unrestricted" models unless specified otherwise, i.e. the union of all k-dimensional subspaces for PCA, the union of all sets of k points for k-means/k-medians, etc.

We provide in Table 10.1 a summary of our results for the different tasks considered.

Although we do not provide any extension of the framework, we expect that these few impossibility results might help to find necessary conditions on the model set for future work. In particular, we stress that the counter-example which has been built for the (non-robust) subspace clustering problem does not hold anymore when we restrict the model to separated (in angle) linear subspaces. The considered constructions also cannot be applied to NMF, for which the loss is $\mathcal{C}^2$.

A natural problem for future work would be to know if the control of the excess risk implies somehow that the LRIP (10.1) must hold for some constant (cf. Figure 10.1). If this is the case, then our results would have strong implications as they would not only allow us to derive necessary conditions that the model set must satisfy in order to obtain statistical learning guarantees, but also induce negative results in some settings. If the excess risk can be controlled without any LRIP holding, then only partial conclusions can be drawn; the LRIP might however still be provable for some of the refined models, and one could try to do so in future work.

Finally, the tools used to derive the results on union of convex sets are very generic, and could certainly be applied to other compression-type tasks in the future.

Chapter 11
Conclusion

Contents

The contributions of this thesis have been grouped in three different parts. We started with considerations on the design of the random matrix involved in the sketching process and the usage of structured random operators, then introduced a novel algorithm for compressive clustering, and eventually demonstrated the relevance of the compressive learning framework to learn from large collections with privacy guarantees.

To conclude this thesis, we summarize in detail the contributions of these three parts (Section 11.1), and propose some perspectives for future work (Section 11.2).

11.1 Summary of the contributions

We summarize here the main contributions of parts II to IV of the thesis. All these ideas build on the existing compressive learning framework introduced in Part I. Details related to the implementation are deferred to Appendix D, but should naturally be considered as a contribution of the thesis as well.

We recall that data is compressed using a feature map of the form $\Phi : \mathbf{x} \mapsto \rho(\boldsymbol{\Omega}^T\mathbf{x})$, where ρ is applied pointwise and $\boldsymbol{\Omega}$ is a random matrix. The feature map Φ implicitly defines a mean kernel in the data space, which depends on ρ but also on the distribution of $\boldsymbol{\Omega}$.

Distribution of $\boldsymbol{\Omega}$ We developed in Part II some considerations related to the distribution of the random matrix $\boldsymbol{\Omega}$ used in the feature map. Chapter 4 highlighted empirically the importance for compressive clustering of choosing the kernel variance to be close to the minimum squared distance ε^2 between clusters. Even if this conclusion has only be drawn for data generated according to a Gaussian mixture model, we expect that our observations should be useful in the future to design robust heuristics to estimate automatically a good kernel scale.

In Chapter 5, we proposed to replace the dense i.i.d. sampling scheme of $\boldsymbol{\Omega}$ used so far by a new design structured by blocks, such that blocks are independent from each other, but the columns of a same block[1] are now correlated. By doing so, we reduce drastically the computational complexity of both sketching and learning operations, as well as the amount of memory required to store the matrix $\boldsymbol{\Omega}$. Extensive empirical simulations for the clustering task show that structured

[1] And hence the associated features, i.e. the associated entries of the sketch.

operators can be used without damaging the quality of the learned models. We also discuss how existing guarantees could be adapted to this new setting.

Compressive clustering with message passing In Part III, we introduced the CL-AMP algorithm to solve the inverse problem arising in compressive clustering. This algorithm comes as an alternative to the existing CL-OMPR method, which is based on orthogonal matching pursuit. We stress that the sketching operator is still the same as in previous works, i.e. made of random Fourier features; the proposed algorithm only tackles the task of recovering the locations of the cluster centers from the empirical sketch of the data.

Our method is based on approximate message passing techniques, and more precisely on the SHyGAMP algorithm. The derivation relies on the modeling assumption that the data is distributed according to a Gaussian mixture model. Deriving a detailed and robust algorithm from the core idea requires many approximations and numerical considerations, which are carefully detailed. We also explained how the method's hyperparameters can be tuned jointly with the message passing iterations.

This new algorithm compares favourably to CL-OMPR on a number of experiments. Successful recovery of the cluster centers has in particular been empirically observed with a sketch size of the order $m = 2kd$, whereas $m = 10kd$ is usually necessary with CL-OMPR to achieve similar results.

Privacy-aware learning In Part IV, we demonstrated the potential of compressive learning for learning applications where privacy is required. After introducing the tools required to define and quantify the notion of privacy, we proved that the sketching mechanism can quite simply be modified to satisfy differential privacy by noise addition. The main contribution consisted in computing sharp bounds on the required noise levels for both random Fourier and quadratic sketches, and the introduction of the subsampled sketching mechanism which enjoys the same privacy guarantees. The latter however intuitively releases less information[2], and allows to better control the privacy-utility-complexity tradeoff.

The noise-to-signal ratio (NSR) of the noisy sketch was then introduced, and shown empirically to be a good indicator[3] of the utility of the sketch for subsequent learning. We thus provided analytical expression of this NSR, and used these expressions for tuning several parameters.

Extensive simulations on both synthetic and real datasets have also been provided to assess the performance of the private sketching mechanism. Some behaviors predicted by the analysis of the NSR have been confirmed empirically, and our method overall compares favorably with other methods from the literature.

[2] From an information-theoretic perspective.

[3] Provided the sketch size m is large enough to obtain meaningful results.

11.2 Future work

Although some short-term perspectives have also been proposed at the end of the preceding chapters, we provide here more general directions for future work.

11.2.1 Short-term perspectives

Information-theoretic privacy guarantees All the guarantees formulated in Part IV are differential privacy guarantees. Although this framework is certainly the most standard in the literature, other definitions of privacy exist as discussed in Section 7.2.3. As suggested in the conclusion of Chapter 9, revisiting the obtained privacy results using alternative privacy metrics, and for instance information-theoretic tools, would be highly interesting. Furthermore, this would be a way to better understand the role that the feature subsampling mechanism could play for privacy preservation.

Stabilizing CL-AMP Despite our efforts on the implementation of CL-AMP, the algorithm is still numerically unstable in some contexts. We expect that at least some parts of these issues could be avoided by spending more time on the implementation, adding more safety checks and maybe adding more clipping steps for the critical parts when needed. Running the code with floating point numbers of arbitrary precision might also help to make the difference between overflows resulting from coding choices (which could possibly be avoided with more engineering), and real diverging behaviors of the algorithm which call for specific mechanisms such as clipping or damping.

Better guarantees for structured operators Despite the good empirical results observed when using structured operators, obtaining statistical learning guarantees in this setting is not straightforward. In particular, the matrices obtained using $\mathbf{M}_{R^3}$ blocks induce a non-isotropic kernel, which is more difficult to work with, compared to the $\mathbf{M}_{\text{ff}}$ or $\mathbf{M}_{GR^2}$ blocks which induce a Gaussian kernel. Indeed, in the latter case existing results for clustering and GMM fitting can be mostly reused, and only the concentration of the features differs and needs to be controlled. We forecast that concentration results could be obtained in a first time, while obtaining results on the kernel induced by $\mathbf{M}_{R^3}$ might be more challeging[4], and should be considered as a middle-term perspective.

[4] We started during the thesis by tackling this problem, but were unsuccessful so far.

11.2.2 Research directions for future work

Handling a broader variety of learning tasks Extending the compressive learning framework to broader families of learning tasks remains one of the core challenges for future work. We reviewed in Section 2.5 the different tasks that have been addressed using compressive methods, and highlighted in Chapter 10 how regularity arguments

could be invoked to derive necessary conditions on the model set and/or the feature map. This short chapter can be seen as first step towards extension to other compressive-type learning tasks, but its conclusions are limited. Going further in this direction would certainly be beneficial, and could help to derive restrictions on the model set to use for simple projection-type tasks such as subspace clustering.

Tackling new tasks, even from an empirical perspective, remains a challenge as it requires to find a sketching operator adapted to the task at hand. Here, studying the different invariances of the considered problem might help to derive invariance properties that the kernel itself should satisfy, and thus help to reduce the class of "compatible" kernels.

Supervised compressive learning Although extending the framework to unsupervised tasks is already a challenge per se, another natural direction for future work would be to address supervised problems. We mentioned already a few works in this direction, using e.g. one sketch per class[5] for classification problems [107, 108], or a single sketch for regression computed by sketching jointly the predictor and outcome variables [111]. There may however be other ways to integrate the data corresponding to the labels (or observed variables), for instance by using specific kernels targeted for the considered tasks.

[5] Plus one sketch for the whole dataset corresponding to a "generic" model in [107].

In the case of two-classes classification problems for instance, the data samples $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$ can be mapped to $\mathbf{z}_i = y_i[\mathbf{x}_i^T; 1]^T \in \mathbb{R}^{d+1}$ without loss of information. It is well known that this formulation allows to easily rewrite some tasks, such as support vector machine classification[6].

[6] In this case, the loss is a function of the quantity $\alpha = y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b)$, where $(\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R})$ defines an affine hyperplane. This quantity can be rewritten $\alpha = \langle \mathbf{z}_i, [\mathbf{w}^T; b]^T \rangle$.

Extension to other types of data We solely considered data in $\mathbb{R}^d$ in our work. Although this is was a natural choice for the considered tasks, one can imagine in the future dealing with different learning tasks for which the data lies in different spaces, or satisfies particular structural properties. A simple extension would be to consider categorical data, but we can also imagine working with collections of images or graphs.

It is possible to convert such data samples into numerical features[7], but often domain-specific kernels exist in the literature[8]. We could thus avoid mapping the data to an euclidean space, and directly work with the features associated to such kernels. Doing so could open the way for applications on very different types of datasets, and contribute to the versatility of the compressive approach.

[7] We used for instance features computed from a graph via the Lagrangian in Chapter 5.

[8] See for instance [331] for references on graph kernels.

Provably good algorithms Obtaining statistical learning guarantees on the whole compressive learning framework remains a challenge.

We recall that the existing guarantees for clustering, Gaussian modeling and PCA [5, 6], only hold for an "ideal" decoder, i.e. assuming that a global optimum can be found for the inverse problem (2.12). Some results have been obtained for CL-OMPR [95], but only for an idealized version of the algorithm and for restricted families of kernels.

The CL-AMP algorithm for clustering introduced in Chapter 6 also comes without any guarantee. As it relies on many approximations, it seems difficult to provide results on the implementation itself. However, we can imagine using tools from the message passing literature to derive some properties[9] of a simplified version of the algorithm, possibly in the large-scale limit in a first time.

[9] Such as a state-evolution rule.

Kernel design and kernel learning We focused in Chapter 4 in finding the variance of the Gaussian kernel yielding optimal clustering performance. We briefly compared in Section 4.2.6 different radial distributions, which amounts to considering kernels which are not Gaussians, and we have seen that the considered distributions could lead to better or more robust clustering results compared to a Gaussian kernel. We have seen that the structured operators from Chapter 5, introduced for the purpose of computational efficiency, can induce new kernels as well[10].

[10] Think for instance of RFF features with blocks of the type $\mathbf{SHD}_3\mathbf{HD}_2\mathbf{HD}_1$, and by opposition to RFF with blocks of the form $\mathbf{SHD}_G\mathbf{HD}_2\mathbf{HD}_1$ with $\mathbf{D}_G$ diagonal Gaussian, which in expectation still induces a Gaussian kernel.

It would be interesting to further investigate the impact of the kernel choice. For instance, it was highlighted empirically in Section 4.2.6 that the adapted radius kernel yields good performance over a wider range of kernel variances (in comparison with a Gaussian kernel). The construction of this kernel has been justified by the will to reduce the use of low frequencies, still in comparison with the Gaussian kernel. Understanding this phenomenon theoretically could help to further optimize the kernel design, at least for clustering in a first time.

In addition to designing kernels which work well for the chosen learning task uniformly for all datasets, we could also try to learn (or at least adapt slightly) the kernel using a small subset of the data before sketching.

For the clustering problem, the estimation of the inter-cluster separation remains a challenge as well. We expect that the results from Chapter 4 will help to design better heuristics to estimate this quantity in a near future.

Kernel compressive learning We mainly focused in this manuscript on the tasks of k-means and principal component analysis. It is sometimes interesting[11] to first map the original data from $\mathbb{R}^d$ to an intermediate reproducing kernel Hilbert space associated to a kernel κ_1, and then perform the desired task (e.g. clustering or PCA) in this space, as it potentially allows to detect non-linear structures in the data. Equivalently, any algorithm which operates on the data samples exclusively via an inner product $\langle \cdot, \cdot \rangle$ can be modified by replacing all the instances of this inner product by the desired kernel κ_1.

[11] As discussed in Section 2.3.

When this kernel is associated to (or well approximated by) a finite-dimensional feature map $\varphi : \mathbb{R}^d \to \mathbb{R}^D$, i.e. $\forall \mathbf{x}, \mathbf{y}, \kappa_1(\mathbf{x}, \mathbf{y}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle$, we can in practice simply run the "linear" algorithm on the transformed version $[\varphi(\mathbf{x}_1), ..., \varphi(\mathbf{x}_n)]$ of the original dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$. This approach is convenient as it does not require to modify the algorithm, but is nonetheless likely to incur an extra memory cost as the feature map φ usually maps the data to a higher-

dimensional space (i.e. $D > d$).

The sketching framework can, from an algorithmic perspective at least, simply be modified to compute generalized moments in the intermediate RKHS. If Φ denotes a feature map to solve the desired task in $\mathbb{R}^D$, one indeed simply needs to compute the sketch using the feature map $\Phi \circ \varphi$, where $\circ$ denotes the composition. We stress that the features $(\varphi(\mathbf{x}_i))_{1 \leq i \leq n}$ can be computed one by one when sketching, but do not need to be stored. Furthermore, fast structured random operators might here be very helpful: firstly, to compute φ when it is a itself a random feature map, but more importantly to compute Φ in settings where D is large.

This strategy can be applied in a straightforward manner for kernel PCA. For kernel k-means, finding a meaningful value of the kernel variance σ_κ^2 used to design the operator Φ might become more tricky, as all the concerns exposed in Chapter 4 remain valid, but any estimation procedure will need to be carried out in the RKHS.

Part VI

Appendix

Chapter A

Generic tools

A.1 Reminder on measures

We recall here the definitions of finite and signed measures.

Definition A.1 (Positive measure): Let $(\mathcal{X}, \Sigma)$ be a measurable space. A function $\mu : \Sigma \to [0, \infty[$ is a measure (i.e. a positive measure) if it satisfies

- $\mu(\emptyset) = 0$;
- for any countable collection $(E_i)_{1 \leq i \leq \infty}$ of pairwise disjoint sets in Σ, $\mu(\bigcup_{i=1}^{\infty} E_i = \sum_{i=1}^{\infty} \mu(E_i)$.

Definition A.2 (Signed measure): A function $\mu : \Sigma \to]-\infty, +\infty[$ that satisfies the two properties of Definition A.1 is called a signed measure.

Definition A.3 (Finite measure): A measure μ on a measurable space $(\mathcal{X}, \Sigma)$ is said finite if $\mu(\mathcal{X}) < \infty$.

Definition A.4 (Probability measure): A finite (positive) measure μ on the measurable space $(\mathcal{X}, \Sigma)$ is called a probability measure if $\mu(\mathcal{X}) = 1$.

We consider only finite measures in this manuscript. We recall that for any measurable space $(\mathcal{X}, \Sigma)$, the set of finite signed measures on $(\mathcal{X}, \Sigma)$ is a real vector space, the set of positive signed measures on $(\mathcal{X}, \Sigma)$ is a convex cone, and the set $\mathcal{P}(\mathcal{X})$ of probability distributions on $(\mathcal{X}, \Sigma)$ is a convex subset of the former.

The operator $\mathcal{A}$ is defined for distributions in (2.10), but can actually easily be extended to the space of finite measures for which Φ is integrable. We used for that the Jordan decomposition.

Definition A.5 (Jordan decomposition [332, Paragraph 6.6]): Let $(\mathcal{X}, \Sigma)$ be a measurable space. Any measure μ defined on Σ admits a unique decomposition $\mu = \mu_+ - \mu_-$, where μ_+, μ_- are positive measures, at least one of which being finite.

A.2 Semi-norms on the space of finite signed measures

We defined in (2.15) the metric $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ as

$$\|\pi - \pi'\|_{\mathcal{L}(\mathcal{H})} \triangleq \sup_{h \in \mathcal{H}} |\mathcal{R}(h, \pi) - \mathcal{R}(h, \pi')| \tag{A.1}$$

in order to avoid introducing too many notations. This formulation was sufficient for our needs in Chapter 2 given that the formulation of the LRIP (2.18) and of the instance optimality property (2.17) only make use of $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ for measures of the form $\pi - \pi'$.

The definition is however more general. For any class $\mathcal{F}$ of measurable functions on the data space $\mathcal{X}$, we define

$$\|\mu\|_{\mathcal{F}} \triangleq \sup_{f \in \mathcal{F}} |\langle \mu, f \rangle|, \tag{A.2}$$

where $\langle \cdot, f \rangle$ is defined as:

- $\langle \pi, f \rangle \triangleq \mathbf{E}_{\mathbf{x} \sim \pi} f(\mathbf{x})$ for a distribution π;
- $\langle \mu, f \rangle \triangleq \alpha_+ \langle \pi_+, f \rangle - \alpha_- \langle \pi_-, f \rangle$ for any finite signed measure μ with Jordan decomposition $\mu = \alpha_+ \pi_+ - \alpha_- \pi_-$ (cf Definition A.5).

We implicitly assume with such notations that f is integrable with respect to π (or π_+, π_- in the second case).

Note that $\|\cdot\|_{\mathcal{F}}$, as defined in (A.2), is a pseudo-norm on the subspace of finite signed measures for which every $f \in \mathcal{F}$ is integrable. Furthermore, $\|\cdot\|_{\mathcal{L}(\mathcal{H})}$ can simply be seen as an instantiation of (A.2) with $\mathcal{F} = \mathcal{L}(\mathcal{H}) \triangleq \{l(h, \cdot) | h \in \mathcal{H}\}$.

Chapter B

Derivations for CL-AMP

We provide here the missing derivations for Section 6.3.2. We refer the reader to this section for the definition of the notations used; this appendix should not be considered as a standalone proof.

B.1 Posterior mean and covariance of $\mathbf{z}$

Marginal posterior $p(z_t|s)$ We first derive an expression for the marginal posterior $p(z_t|s)$ under the pseudo-prior $z_l \sim \mathcal{N}(\hat{p}_l, q_l^{\mathbf{p}})\ \forall l \in [\![1,k]\!]$. We have

$$\begin{aligned} p(z_t|s) &= \int_{\mathbb{R}^k} p(\mathbf{z},\theta_t|s)\,\mathrm{d}\theta_t\,\mathrm{d}\mathbf{z}_{\setminus t} \\ &= \frac{1}{p(s)}\int_{\mathbb{R}^k} p(s|\mathbf{z},\theta_t)p(\theta_t|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\theta_t\,\mathrm{d}\mathbf{z}_{\setminus t} \\ &= \frac{1}{p(s)}\int_{\mathbb{R}^k} p(s|\mathbf{z}_{\setminus t},\theta_t)\mathcal{N}(\theta_t; gz_t, g^2q^n)\prod_{l=1}^{k}\mathcal{N}(z_l;\hat{p}_l,q_l^{\mathbf{p}})\,\mathrm{d}\theta_t\,\mathrm{d}\mathbf{z}_{\setminus t} \end{aligned} \tag{B.1}$$

We now introduce $\tilde{z}_l \triangleq z_l - \hat{p}_l$ for all $l \neq t$, and separate the term with index t from the rest of the product, so that

$$p(z_t|s) = \frac{\mathcal{N}(z_t;\hat{p}_t,q_t^{\mathbf{p}})}{p(s)}\int_{\mathbb{R}}\mathcal{N}(\theta_t;gz_t,g^2q^n) \left[\int_{\mathbb{R}^{k-1}} p(s|\tilde{\mathbf{z}}_{\setminus t},\theta_t)\prod_{1\le l\neq t\le k}\mathcal{N}(\tilde{z}_l;0,q_l^{\mathbf{p}})\,\mathrm{d}\tilde{\mathbf{z}}_{\setminus t}\right]\mathrm{d}\theta_t,$$

where $p(s|\tilde{\mathbf{z}}_{\setminus t},\theta_t)$ is associated to the model

$$s = \beta_t\exp(\imath\theta_t) + \sum_{1\le l\neq t\le k}\beta_l\exp(\imath g(\hat{p}_l+\tilde{z}_l+n_l)) \tag{B.2}$$

with i.i.d. $n_l \sim \mathcal{N}(0,q^n)$. We use the mutual independence of $\hat{z}_l$ and n_l to define $\tilde{n}_l \triangleq \tilde{z}_l + n_l \sim \mathcal{N}(0, q_l^{\mathbf{p}} + q^n)$, so that if we denote $p(s|\theta_t)$ the conditional density associated with the model

$$s = \beta_t\exp(\imath\theta_t) + \sum_{1\le l\neq t\le k}\beta_l\exp(\imath g(\hat{p}_l+\tilde{n}_l)), \tag{B.3}$$

with i.i.d. $\tilde{n}_l \sim \mathcal{N}(0, q_l^{\mathbf{p}} + q^n)$, then we can rewrite (B.1) as

$$\boxed{p(z_t|s) = \frac{\mathcal{N}(z_t;\hat{p}_t,q_t^{\mathbf{p}})}{p(s)}\int_{\mathbb{R}}\mathcal{N}(\theta_t;gz_t,g^2q^n)p(s|\theta_t)\,\mathrm{d}\theta_t.} \tag{B.4}$$

Expression of $\hat{z}_t$ and $q_t^{\mathbf{z}}$ w.r.t. the marginal posterior $p(z_t|s)$ By definition of $\hat{z}_t$, we have

$$\begin{aligned}
\hat{z}_t &\triangleq \int_{\mathbb{R}} z_t p(z_t|s)\,\mathrm{d}z_t \\
&\overset{\text{(i)}}{=} \frac{1}{p(s)} \int_{\mathbb{R}} \left[\int_{\mathbb{R}} z_t \mathcal{N}(z_t; \hat{p}_t, q_t^{\mathbf{p}}) \mathcal{N}(z_t; \theta_t/g, q^n)\,\mathrm{d}z_t \right] p(s|\theta_t)\,\mathrm{d}\theta_t \\
&\overset{\text{(ii)}}{=} \int_{\mathbb{R}} \left[\int_{\mathbb{R}} z_t \mathcal{N}\left(z_t; \frac{\frac{\hat{p}_t}{q_t^{\mathbf{p}}} + \frac{\theta_t/g}{q^n}}{\frac{1}{q_t^{\mathbf{p}}} + \frac{1}{q^n}}, \frac{1}{\frac{1}{q_t^{\mathbf{p}}} + \frac{1}{q^n}} \right) \mathrm{d}z_t \right] p(\theta_t|s)\,\mathrm{d}\theta_t
\end{aligned}$$

(i) Using (B.4) and rewriting.

(ii) Using the gaussian multiplication rule, i.e. $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = \mathcal{N}(\boldsymbol{\mu}_1; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)\mathcal{N}(\mathbf{x}; (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2), (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1})$ [201, Section 8.1.8].

where

$$p(\theta_t|s) \triangleq \frac{\mathcal{N}\big(\theta_t; g\hat{p}_t, g^2(q_t^{\mathbf{p}} + q^n)\big) p(s|\theta_t)}{p(s)}. \tag{B.5}$$

We denote $\hat{\theta}_t \triangleq \int_{\mathbb{R}} \theta_t p(\theta_t|s)\,\mathrm{d}\theta_t$ the posterior mean of θ_t, so that we can rewrite

$$\begin{aligned}
\hat{z}_t &\overset{\text{(i)}}{=} \int_{\mathbb{R}} \left(\frac{\hat{p}_t}{q_t^{\mathbf{p}}} + \frac{\theta_t/g}{q^n} \right) \left(\frac{1}{q_t^{\mathbf{p}}} + \frac{1}{q^n} \right)^{-1} p(\theta_t|s)\,\mathrm{d}\theta_t \\
&= \left(\frac{\hat{p}_t}{q_t^{\mathbf{p}}} + \frac{\hat{\theta}_t/g}{q^n} \right) \left(\frac{1}{q_t^{\mathbf{p}}} + \frac{1}{q^n} \right)^{-1} \\
&\boxed{\hat{z}_t = \frac{\hat{p}_t}{1 + \frac{q_t^{\mathbf{p}}}{q^n}} + \frac{\hat{\theta}_t}{g(1 + \frac{q^n}{q_t^{\mathbf{p}}})}}
\end{aligned} \tag{B.6}$$

(i) By integrating z_t out.

We now proceed similarly for the variance. We have

$$\begin{aligned}
q_t^{\mathbf{z}} &\triangleq \mathrm{Var}(z_t|s) = \int_{\mathbb{R}} |z_t - \hat{z}_t|^2 p(z_t|s)\,\mathrm{d}z_t \\
&= \int_{\mathbb{R}} \left[\int_{\mathbb{R}} |z_t - \hat{z}_t|^2 \mathcal{N}(z_t; \hat{p}_t, q_t^{\mathbf{p}}) \mathcal{N}(\theta_t; g z_t, g^2 q^n)\,\mathrm{d}z_t \right] \frac{p(s|\theta_t)}{p(s)}\,\mathrm{d}\theta_t \\
&\overset{\text{(i)}}{=} \int_{\mathbb{R}} \left[\int_{\mathbb{R}} |z_t - \hat{z}_t|^2 \mathcal{N}\left(z_t; \frac{\frac{\hat{p}_t}{q_t^{\mathbf{p}}} + \frac{\theta_t/g}{q^n}}{\frac{1}{q_t^{\mathbf{p}}} + \frac{1}{q^n}}, \frac{1}{\frac{1}{q_t^{\mathbf{p}}} + \frac{1}{q^n}} \right) \mathrm{d}z_t \right] p(\theta_t|s)\,\mathrm{d}\theta_t
\end{aligned}$$

(i) Using the same Gaussian multiplication rule as above.

Changing the variable z_t to $\tilde{z}_t \triangleq z_t - \hat{z}_t$, we get

$$\begin{aligned}
q_t^{\mathbf{z}} &\overset{\text{(ii)}}{=} \int_{\mathbb{R}} \left[\int_{\mathbb{R}} \tilde{z}_t^2 \mathcal{N}\left(\tilde{z}_t; \frac{\theta_t - \hat{\theta}_t}{g(1 + \frac{q^n}{q_t^{\mathbf{p}}})}, \frac{1}{\frac{1}{q_t^{\mathbf{p}}} + \frac{1}{q^n}} \right) \mathrm{d}z_t \right] p(\theta_t|s)\,\mathrm{d}\theta_t \\
&= \int_{\mathbb{R}} \left[\left(\frac{\theta_t - \hat{\theta}_t}{g(1 + \frac{q^n}{q_t^{\mathbf{p}}})} \right)^2 + \frac{q^n}{\frac{q^n}{q_t^{\mathbf{p}}} + 1} \right] p(\theta_t|s)\,\mathrm{d}\theta_t \\
&\boxed{q_t^{\mathbf{z}} = \frac{q^n}{\frac{q^n}{q_t^{\mathbf{p}}} + 1} + \frac{1}{g^2(1 + \frac{q^n}{q_t^{\mathbf{p}}})^2} \mathrm{Var}(\theta_t|s)}
\end{aligned}$$

(ii) using (B.6)

In the following, we denote $q_t^{\theta} = \mathrm{Var}(\theta_t|s)$ – which is computed with respect to $p(\theta_t|s)$.

B.2 Approximation of the sum

We derived expressions of $\hat{z}_t$ and $q_t^{\mathbf{z}}$ in terms of marginal posterior mean and variance of θ_t, but these quantities are still difficult to compute due to the form of $p(\theta_t|s)$ in (B.5). We thus approximate the sum in (B.3) as Gaussian. The $(\tilde{n})_{1\leq l\neq t\leq k}$ being mutually independent, we can sum the mean and covariances of each term of the sum. We define $v_l = \exp(jg(\hat{p}_l + \tilde{n}_l))$, so that the model (B.3) can then be rewritten

$$\begin{bmatrix} \Re(s) \\ \Im(s) \end{bmatrix} \sim \mathcal{N}\Bigg(\beta_t \begin{bmatrix} \cos(\theta_t) \\ \sin(\theta_t) \end{bmatrix} + \sum_{1\leq l\neq t\leq k} \beta_l \mathbf{E}\Bigg(\begin{bmatrix} \Re(v_l) \\ \Im(v_l) \end{bmatrix}\Bigg), \sum_{1\leq l\neq t\leq k} \beta_l^2 \operatorname{Cov}\Bigg(\begin{bmatrix} \Re(v_l) \\ \Im(v_l) \end{bmatrix}\Bigg)\Bigg). \tag{B.7}$$

We have, using $q^n \to 0$,

$$\mathbf{E}(v_l) = \int_{\mathbb{R}} \mathcal{N}(z_l; \hat{p}_l, q_l^{\mathbf{p}}) \exp(\iota g z_l)\, \mathrm{d}z_l \tag{B.8}$$

$$= \exp(\iota g \hat{p}_l - \tfrac{1}{2} g^2 q_l^{\mathbf{p}}) \tag{B.9}$$

$$\mathbf{E}\begin{bmatrix} \Re(v_l) \\ \Im(v_l) \end{bmatrix} = \exp(-\tfrac{1}{2} g^2 q_l^{\mathbf{p}}) \begin{bmatrix} \cos(g\hat{p}_l) \\ \sin(g\hat{p}_l) \end{bmatrix}, \tag{B.10}$$

$$2\mathbf{E}\big[\Re(v_l)^2\big] = 1 + \exp(-g^2 q_l^{\mathbf{p}}) \cos(2g\hat{p}_l), \tag{B.11}$$

$$2\mathbf{E}\big[\Im(v_l)^2\big] = 1 - \exp(-g^2 q_l^{\mathbf{p}}) \cos(2g\hat{p}_l), \tag{B.12}$$

$$2\mathbf{E}[\Re(v_l)\Im(v_l)] = \exp(-g^2 q_l^{\mathbf{p}}) \sin(2g\hat{p}_l). \tag{B.13}$$

Hence (B.7) can be written

$$p\left(\begin{bmatrix} \Re(s) \\ \Im(s) \end{bmatrix} \middle| \theta_t\right) \approx \mathcal{N}\left(\begin{bmatrix} \Re(s) \\ \Im(s) \end{bmatrix}; \beta_t \begin{bmatrix} \cos(\theta_t) \\ \sin(\theta_t) \end{bmatrix} + \mathbf{m}_t, \mathbf{\Gamma}_t\right) \tag{B.14}$$

where

$$\mathbf{m}_t = \sum_{1\leq l\neq t\leq k} \alpha_l \exp(-\tfrac{1}{2} g^2 (\tau_l + q_l^{\mathbf{p}})) \begin{bmatrix} \cos(g\hat{p}_l) \\ \sin(g\hat{p}_l) \end{bmatrix}$$

$$\mathbf{\Gamma}_t = \tfrac{1}{2} \sum_{1\leq l\neq t\leq k} \beta_l^2 (1 - \exp(-g^2 q_l^{\mathbf{p}}) \times \left(\mathbf{I} - \exp(-g^2 q_l^{\mathbf{p}}) \begin{bmatrix} \cos(2g\hat{p}_l) & \sin(2g\hat{p}_l) \\ \sin(2g\hat{p}_l) & -\cos(2g\hat{p}_l) \end{bmatrix}\right)$$

We scale (B.14) by β_t^{-1}, i.e. we rewrite

$$p\left(\beta_t^{-1}\begin{bmatrix} \Re(s) \\ \Im(s) \end{bmatrix} \middle| \theta_t\right) \approx \mathcal{N}\left(\begin{bmatrix} \cos(\theta_t) \\ \sin(\theta_t) \end{bmatrix}; \beta_t^{-1} \begin{bmatrix} \Re(s) \\ \Im(s) \end{bmatrix} + \beta_t^{-1}\mathbf{m}_t, \beta_t^{-2}\mathbf{\Gamma}_t\right) \tag{B.15}$$

so that the right hand side becomes proportional to a generalized von Mises density [254] over $\theta_t \in [0, 2\pi[$. That is,

$$p(s|\theta_t) \propto \exp(\kappa_t \cos(\theta_t - \zeta_t) + \overline{\kappa}_t \cos(2(\theta_t - \overline{\zeta}_t))) \tag{B.16}$$

where the parameters $\kappa_t, \overline{\kappa}_t > 0$ ad $\zeta_t, \overline{\zeta}_t \in [0, 2\pi[$ are defined from $\beta_t^{-1}s$, $\beta_t^{-1}\mathbf{m}_t$ and $\beta_t^{-2}\mathbf{\Gamma}_t$ as follows

$$\kappa_t \cos(\zeta_t) = -\frac{1}{1-\rho_t^2}\left(\frac{\rho_t \overline{\nu}_t}{\sigma_t \overline{\sigma}_t} - \frac{\nu_t}{\sigma_t^2}\right) \tag{B.17}$$

$$\kappa_t \sin(\zeta_t) = -\frac{1}{1-\rho_t^2}\left(\frac{\rho_t \nu_t}{\sigma_t \overline{\sigma}_t} - \frac{\overline{\nu}_t}{\overline{\sigma}_t^2}\right) \tag{B.18}$$

$$\overline{\kappa}_t \cos(2\overline{\zeta}_t) = -\frac{1}{4(1-\rho_t^2)}\left(\frac{1}{\sigma_t^2} - \frac{1}{\overline{\sigma}_t^2}\right) \tag{B.19}$$

$$\overline{\kappa}_t \sin(2\overline{\zeta}_t) = \frac{\rho_t}{2(1-\rho_t^2)\sigma_t \overline{\sigma}_t}, \tag{B.20}$$

where the intermediate quantities are defined by

$$\begin{bmatrix} \nu_t \\ \overline{\nu}_t \end{bmatrix} \triangleq \beta_t^{-1}\left(\begin{bmatrix} \Re(s) \\ \Im(s) \end{bmatrix} - \mathbf{m}_t\right)$$

$$\begin{bmatrix} \sigma_t^2 & \rho_t \sigma_t \overline{\sigma}_t, \\ \rho_t \sigma_t \overline{\sigma}_t & \overline{\sigma}_t^2 \end{bmatrix} \triangleq \beta_t^{-2}\mathbf{\Gamma}_t.$$

The posterior distribution of θ_t under the pseudo-prior $z_t \sim \mathcal{N}(\hat{p}_t, q_t^{\mathbf{P}})$ takes the form

$$\begin{aligned} p(\theta_t|s) &\overset{\text{(i)}}{\propto} \mathcal{N}(\theta_t; g\hat{p}_t, g^2 q_t^{\mathbf{P}}) p(s|\theta_t) \\ &\propto \exp\left(\kappa_t \cos(\theta_t - \zeta_t) + \overline{\kappa}_t \cos(2(\theta_t - \overline{\zeta}_t)) - \frac{|\theta_t - g\hat{p}_t|^2}{2g^2 q_t^{\mathbf{P}}}\right) \end{aligned}$$

(i) By definition, $\theta_t = gz_t$ once $q^n \to 0$.

Chapter C

Privacy Proofs

C.1 Results on Nonresonant Frequencies

In order to prove the sharpness of the sensitivity computed in Lemma 8.6, we rely on some results from diophantine approximation theory. We recall the definition of nonresonant frequencies.

Definition C.1: The vectors $(\omega_j)_{1\leq j\leq m} \in \mathbb{R}$ are called nonresonant frequencies if they are linearly independent over the rationals. The vectors $(\boldsymbol{\omega}_j)_{1\leq j\leq m} \in \mathbb{R}^d$ are called nonresonant frequency vectors if there exists a vector $\mathbf{v} \in \mathbb{R}^d$ such that the scalars $(\boldsymbol{\omega}_j^T\mathbf{v})_{1\leq j\leq m}$ are nonresonant frequencies.

Before proving Lemma 8.5, we introduce a variant of the result in dimension 1.

Lemma C.2

Let $(\omega_j, \varphi_j)_{1\leq j\leq m}$ be real numbers, and f a 2π-periodic function such that there exists z at which f is continuous and reaches its maximum. If the $(\omega_j)_{1\leq j\leq m}$ are linearly independent over the rationals, then $\sup_{x\in\mathbb{R}} \inf_{j\in[\![1;m]\!]} f(\omega_j x - \varphi_j) = \sup_{x\in\mathbb{R}} f(x)$.

Proof C.3: Let $z \in [0, 2\pi[$ be a point at which f reaches its maximum, i.e., $z \in \arg\max_{[0,2\pi[} f(z)$, and at which f is continuous. Using this continuity assumption, the result amounts to saying that one can find $t \in \mathbb{R}$ such that the $(\omega_j t - \varphi_j - z)_{1\leq j\leq m}$ are simultaneously arbitrary close to $2\pi\mathbb{Z}$. Denoting $d(x, S) = \inf\{|x - s| : s \in S\}$, this is equivalent to saying that for any $\varepsilon > 0$, we can find a real t such that we simultaneously have $d((\omega_j t - \varphi_j - z)/(2\pi), \mathbb{Z}) < \varepsilon$ for all $j \in [\![1, m]\!]$. This derives directly from Kronecker's theorem [333] on diophantine approximation, given that the $\omega_j/(2\pi)$ are linearly independent over the rationals.

Proof C.4 (Proof of Lemma 8.5): We propose to convert the problem to its one-dimensional counterpart.

$$\sup_{\mathbf{x}\in\mathbb{R}^d} \inf_{j\in[\![1;m]\!]} f(\boldsymbol{\omega}_j^T\mathbf{x} - \varphi_j) = \sup_{\mathbf{v}\in\mathbb{R}^d:\|\mathbf{v}\|=1} \sup_{x\in\mathbb{R}} \inf_{j\in[\![1;m]\!]} f(x\boldsymbol{\omega}_j^T\mathbf{v} - \varphi_j) \tag{C.1}$$

Let $\mathbf{v}$ be such that the scalars $a_j \triangleq \boldsymbol{\omega}_j^T \mathbf{v}$ (for $1 \leq j \leq m$) are nonresonant, which exists because the vectors $(\boldsymbol{\omega}_j)_{1 \leq j \leq m}$ are themselves nonresonant. The quantity (C.1) is upper-bounded by $\sup_{x \in \mathbb{R}} f(x) = f(z)$, and can be lower-bounded by

$$\sup_{x \in \mathbb{R}} \inf_{j \in [\![1;m]\!]} f(x a_j - \varphi_j) = \sup_{x \in \mathbb{R}} f(x)$$

where the last equality comes from Lemma C.2, the a_j being nonresonant.

C.2 Results without subsampling

Proof C.5 (Proof of Lemma 8.12): We have

$$\begin{aligned}
\Delta_2^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RFF}})^2 &= \sup_{\mathbf{x},\mathbf{y} \in \mathbb{R}^d} \|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\|_2^2 = \sup_{\mathbf{x},\mathbf{y} \in \mathbb{R}^d} \sum_{j=1}^m |\Phi(\mathbf{x})_j - \Phi(\mathbf{y})_j|^2 \\
&= \sup_{\mathbf{x},\mathbf{y} \in \mathbb{R}^d} \sum_{j=1}^m |(\rho(\boldsymbol{\omega}_j^T \mathbf{x} + u_j) + i\rho(\boldsymbol{\omega}_j^T \mathbf{x} + u_j - \tfrac{\pi}{2})) \\
&\qquad - (\rho(\boldsymbol{\omega}_j^T \mathbf{y} + u_j) + i\rho(\boldsymbol{\omega}_j^T \mathbf{y} + u_j - \tfrac{\pi}{2}))|^2 \\
&= \sup_{\mathbf{x},\mathbf{y} \in \mathbb{R}^d} \sum_{j=1}^m (\rho(\boldsymbol{\omega}_j^T \mathbf{x} + u_j) - \rho(\boldsymbol{\omega}_j^T \mathbf{y} + u_j))^2 \\
&\qquad + (\rho(\boldsymbol{\omega}_j^T \mathbf{x} + u_j - \tfrac{\pi}{2}) - \rho(\boldsymbol{\omega}_j^T \mathbf{y} + u_j - \tfrac{\pi}{2}))^2 \\
&= \sup_{\mathbf{x},\mathbf{y} \in \mathbb{R}^d} \sum_{j=1}^m 2(1 - (\rho(\boldsymbol{\omega}_j^T \mathbf{x} + u_j)\rho(\boldsymbol{\omega}_j^T \mathbf{y} + u_j) \\
&\qquad + \rho(\boldsymbol{\omega}_j^T \mathbf{x} + u_j - \tfrac{\pi}{2})\rho(\boldsymbol{\omega}_j^T \mathbf{y} + u_j - \tfrac{\pi}{2}))) \qquad \text{(C.2)}
\end{aligned}$$

- For **unquantized features**, we have $\rho = \cos$ and $\rho(\cdot - \frac{\pi}{2}) = \sin$, hence

$$\begin{aligned}
\Delta_2^{\mathrm{B}}(\boldsymbol{\Sigma}^{\mathrm{RFF}})^2 &= \sup_{\mathbf{x},\mathbf{y} \in \mathbb{R}^d} \sum_{j=1}^m 2(1 - (\cos(\boldsymbol{\omega}_j^T \mathbf{x} + u_j)\cos(\boldsymbol{\omega}_j^T \mathbf{y} + u_j) \\
&\qquad + \sin(\boldsymbol{\omega}_j^T \mathbf{x} + u_j)\sin(\boldsymbol{\omega}_j^T \mathbf{y} + u_j))) \\
&= \sup_{\mathbf{x},\mathbf{y} \in \mathbb{R}^d} \sum_{j=1}^m 2(1 + \cos(\boldsymbol{\omega}_j^T(\mathbf{x} - \mathbf{y}) - \pi)) \\
&= 2\left(m + \sup_{\mathbf{z} \in \mathbb{R}^d} \sum_{j=1}^m \cos(\boldsymbol{\omega}_j^T \mathbf{z} - \pi)\right) \\
&= 4m
\end{aligned}$$

  by Lemma 8.5 using the nonresonant property of the frequencies.

- For **quantized features**, we reuse the quantities defined in the proof of Lemma 8.6, i.e. we denote $f(\cdot) \triangleq \rho(\cdot) + \rho(\cdot - \frac{\pi}{2})$ and, for any $\boldsymbol{\varphi} = [\varphi_1, ..., \varphi_m]$, define $f_{\boldsymbol{\varphi}}(\mathbf{x}) = \sum_{j=1}^m f(\boldsymbol{\omega}_j^T \mathbf{x} -$

φ_j). Starting from the generic expression (C.2) we get

$$\Delta_2^{\text{B}}(\boldsymbol{\Sigma}^{\text{RFF}})^2 = 2\Big(m - \inf_{\mathbf{x}\in\mathbb{R}^d}\inf_{\mathbf{y}\in\mathbb{R}^d}\Big[\sum_{j=1}^m \rho(\boldsymbol{\omega}_j^T\mathbf{x}+u_j)\rho(\boldsymbol{\omega}_j^T\mathbf{y}+u_j) + \rho(\boldsymbol{\omega}_j^T\mathbf{x}+u_j-\tfrac{\pi}{2})\rho(\boldsymbol{\omega}_j^T\mathbf{y}+u_j-\tfrac{\pi}{2})\Big]\Big)$$

For any fixed $\mathbf{x}\in\mathbb{R}^d$, we have $\rho(\boldsymbol{\omega}_j^T\mathbf{x}+u_j) = \pm 2^{-1/2}$ and $\rho(\boldsymbol{\omega}_j^T\mathbf{x}+u_j-\frac{\pi}{2}) = \pm 2^{-1/2}$, thus using the same arguments developed in the proof of Lemma 8.6, there are some $\varphi_j \in \{0,\pi/2,\pi,3\pi/2\}$ such that

$$\begin{aligned}
&\inf_{\mathbf{y}\in\mathbb{R}^d}\Big(\sum_{j=1}^m(\rho(\boldsymbol{\omega}_j^T\mathbf{x}+u_j)\rho(\boldsymbol{\omega}_j^T\mathbf{y}+u_j) \\
&\qquad + \rho(\boldsymbol{\omega}_j^T\mathbf{x}+u_j-\tfrac{\pi}{2})\rho(\boldsymbol{\omega}_j^T\mathbf{y}+u_j-\tfrac{\pi}{2}))\Big) \\
&= 2^{-1/2}\inf_{\mathbf{y}\in\mathbb{R}^d}\sum_{j=1}^m \pm\rho(\boldsymbol{\omega}_j^T\mathbf{y}+u_j) \pm \rho(\boldsymbol{\omega}_j^T\mathbf{y}+u_j-\tfrac{\pi}{2}) \\
&= 2^{-1/2}\inf_{\mathbf{y}\in\mathbb{R}^d}\sum_{j=1}^m f(\boldsymbol{\omega}_j^T\mathbf{y}+u_j+\varphi_j) \\
&= -2^{-1/2}\sup_{\mathbf{y}\in\mathbb{R}^d}\sum_{j=1}^m f_{\pi-\boldsymbol{\varphi}-\mathbf{u}}(\boldsymbol{\omega}_j^T\mathbf{y}) \\
&= -m
\end{aligned}$$

which is independent of the choice of $\mathbf{x}$ and concludes the proof.

C.3 Proofs on Sketching with Subsampling

C.3.1 General results

Proof C.6 (Proof of Lemma 8.16): We define the permutation of a set of masks as $\sigma((\mathbf{q}_1,...,\mathbf{q}_n)) = (\mathbf{q}_{\sigma(1)},...,\mathbf{q}_{\sigma(n)})$ for $\sigma\in\mathcal{S}_n$. For any set of masks $\mathbf{Q}\in\mathcal{Q}^n$, and any dataset $\mathbf{X}$ such that $|\mathbf{X}| = n$, we denote $p_{\mathbf{X}}(\cdot|\mathbf{Q}) = p_{\boldsymbol{\Sigma}_{\mathcal{L},\mathbf{Q}}(\mathbf{X})}(\cdot)$ the density of $\boldsymbol{\Sigma}_{\mathcal{L},\mathbf{Q}}(\mathbf{X})$. Unless otherwise specified, $p_{\mathbf{X}}$ denotes the density of $\overline{\boldsymbol{\Sigma}}_{\mathcal{L}}(\mathbf{X})$.

We prove the result for a real-valued feature map Φ, and discuss the complex case at the end of the proof. We will prove that

$$\sup_{\mathbf{X},\mathbf{Y}\in\mathcal{D}:\mathbf{X}\overset{\cup}{\sim}\mathbf{Y}}\sup_{\mathbf{z}\in\mathcal{Z}}\frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})} = \exp(\varepsilon^*),$$

which is equivalent to the lemma statement. If $\mathbf{Q}_{n-1} = (\mathbf{q}_1,...,\mathbf{q}_{n-1})$ is a set of masks and $\mathbf{q}_n$ a single mask, defining $\mathbf{Q} = (\mathbf{q}_1,...,\mathbf{q}_n)$ we use the notations $\overline{\boldsymbol{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\cdot) \triangleq \overline{\boldsymbol{\Sigma}}_{\mathbf{Q}}(\cdot)$ and $p(\mathbf{z}|\mathbf{Q}_{n-1},\mathbf{q}_n) \triangleq p(\mathbf{z}|\mathbf{Q})$. In the following $\mathbf{q}_n$, $\mathbf{Q}_{n-1}$ and $\mathbf{Q}$ are implicitly drawn (independently) from respectively $p_{\mathbf{q}}$, $p_{\mathbf{q}}^{n-1}$ and $p_{\mathbf{q}}^n$, where $p_{\mathbf{q}}$ is the probability distribution of the masks from Definition 8.15. Considering $\mathbf{X},\mathbf{Y}\in\mathcal{D}$ such that $\mathbf{X}\overset{\cup}{\sim}\mathbf{Y}$ we distinguish

two cases, depending whether $|\mathbf{X}| = |\mathbf{Y}| + 1$ or $|\mathbf{X}| = |\mathbf{Y}| - 1$.

Case $|\mathbf{X}| = |\mathbf{Y}| + 1$ For any $\mathbf{X} \overset{\cup}{\sim} \mathbf{Y}$, denoting $n = |\mathbf{X}|$ and assuming for now that $|\mathbf{X}| = |\mathbf{Y}| + 1$, there is by Definition 7.3 a permutation $\sigma \in \mathcal{S}_n$ such that $\sigma(\mathbf{X})$ satisfies $\sigma(\mathbf{X}) \overset{\cup}{\approx} \mathbf{Y}$. We have $\overline{\mathbf{\Sigma}}_{\mathbf{Q}}(\sigma(\mathbf{X})) = \overline{\mathbf{\Sigma}}_{\sigma^{-1}(\mathbf{Q})}(\mathbf{X})$, and as the masks are drawn i.i.d. according to $p_{\mathbf{q}}$, we obtain

$$\begin{aligned}
p_{\mathbf{X}}(\mathbf{z}) &= \mathbf{E}_{\mathbf{Q}}[p_{\mathbf{X}}(\mathbf{z}|\mathbf{Q})] = \mathbf{E}_{\mathbf{Q}}\left[p_{\mathbf{X}}(\mathbf{z}|\sigma^{-1}(\mathbf{Q}))\right] = \mathbf{E}_{\mathbf{Q}}\left[p_{\sigma(\mathbf{X})}(\mathbf{z}|\mathbf{Q})\right] \\
&= \mathbf{E}_{\mathbf{Q}_{n-1},\mathbf{q}_n}\left[p_{\sigma(\mathbf{X})}(\mathbf{z}|\mathbf{Q}_{n-1},\mathbf{q}_n)\right] \\
&= \mathbf{E}_{\mathbf{q}_n}\mathbf{E}_{\mathbf{Q}_{n-1}}\left[p_{\sigma(\mathbf{X})}(\mathbf{z}|\mathbf{Q}_{n-1},\mathbf{q}_n)\right] \\
p_{\mathbf{Y}}(\mathbf{z}) &= \mathbf{E}_{\mathbf{Q}_{n-1}}[p_{\mathbf{Y}}(\mathbf{z}|\mathbf{Q}_{n-1})]
\end{aligned}$$

As a consequence we have

$$\begin{aligned}
\frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})} &= \frac{\mathbf{E}_{\mathbf{q}_n}\mathbf{E}_{\mathbf{Q}_{n-1}}\left[p_{\sigma(\mathbf{X})}(\mathbf{z}|\mathbf{Q}_{n-1},\mathbf{q}_n)\right]}{\mathbf{E}_{\mathbf{Q}_{n-1}}[p_{\mathbf{Y}}(\mathbf{z}|\mathbf{Q}_{n-1})]} \\
&= \frac{\mathbf{E}_{\mathbf{q}_n}\mathbf{E}_{\mathbf{Q}_{n-1}}\exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{X}))\|_1)}{\mathbf{E}_{\mathbf{Q}_{n-1}}\exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{Y})\|_1)} \qquad \text{(C.3)}
\end{aligned}$$

Note that for any $\mathbf{Q}_{n-1}, \mathbf{q}_n$ we have $\overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{X})) = \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{Y}) + \frac{m}{r}\Phi(\mathbf{x}_n) \odot \mathbf{q}_n$ by definition of σ and thus for any $\mathbf{Q}_{n-1}, \mathbf{q}_n, \mathbf{z}$ we have

$$\begin{aligned}
-\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{X}))\|_1 &= -\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{Y}) - \frac{m}{r}\Phi(\mathbf{x}_n) \odot \mathbf{q}_n\|_1 \\
&\leq -\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{Y})\|_1 + \|\frac{m}{r}\Phi(\mathbf{x}_n) \odot \mathbf{q}_n\|_1. \qquad \text{(C.4)}
\end{aligned}$$

Equality holds iff for all $j \in [\![1, m]\!]$, $(\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{X})))_j$ and $(\Phi(\mathbf{x}_n) \odot \mathbf{q}_n)_j$ have the same sign or any of the two terms is null. Define $c \triangleq \max_{1 \leq i \leq n} \|\Phi(\mathbf{x}_i)\|_\infty$. For any choice of binary masks $\mathbf{Q}_{n-1}$, we have $\|\overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{X}))\|_\infty \leq nc\frac{m}{r} \triangleq M$. In particular, if we define $\tilde{\mathbf{z}} \triangleq M\,\mathrm{sign}(\Phi(\mathbf{x}_n))$, where sign is applied pointwise, $\tilde{\mathbf{z}}$ yields equality in Equation (C.4) for all $\mathbf{Q}_{n-1}, \mathbf{q}_n$ simultaneously. Using Equation (C.4) in Equation (C.3) and taking the supremum over $\mathbf{z}$, we get

$$\begin{aligned}
&\sup_{\mathbf{z} \in \tilde{\mathcal{Z}}} \frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})} \\
&\leq \sup_{\mathbf{z} \in \tilde{\mathcal{Z}}} \frac{\mathbf{E}_{\mathbf{q}_n}\mathbf{E}_{\mathbf{Q}_{n-1}}\exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{Y})\|_1 + \frac{1}{b}\|\frac{m}{r}\Phi(\mathbf{x}_n) \odot \mathbf{q}_n\|_1)}{\mathbf{E}_{\mathbf{Q}_{n-1}}\exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{Y})\|_1)} \\
&= \mathbf{E}_{\mathbf{q}_n}\exp\left(\frac{1}{b}\frac{m}{r}\|\Phi(\mathbf{x}_n) \odot \mathbf{q}_n\|_1\right)
\end{aligned}$$

but we also have

$$\sup_{\mathbf{z} \in \tilde{\mathcal{Z}}} \frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})} \geq \frac{p_{\mathbf{X}}(\tilde{\mathbf{z}})}{p_{\mathbf{Y}}(\tilde{\mathbf{z}})} = \mathbf{E}_{\mathbf{q}_n}\exp\left(\frac{1}{b}\frac{m}{r}\|\Phi(\mathbf{x}_n) \odot \mathbf{q}_n\|_1\right),$$

therefore equality holds.

Case $|\mathbf{X}| = |\mathbf{Y}| - 1$ We assumed so far $|\mathbf{X}| = |\mathbf{Y}| + 1$, but now if $|\mathbf{X}| + 1 = |\mathbf{Y}| = n$, there is σ such that $\sigma(\mathbf{Y}) \overset{\cup}{\approx} \mathbf{X}$ and we have $\overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{X}) = \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{Y})) - \frac{m}{r}\Phi(\mathbf{y}_n) \odot \mathbf{q}_n$. Another triangular inequality yields

$$\begin{aligned} -\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{X})\|_1 &= -\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{Y})) + \frac{m}{r}\Phi(\mathbf{y}_n) \odot \mathbf{q}_n\|_1 \\ &\leq -\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{X}))\|_1 \\ &\quad + \|-\frac{m}{r}\Phi(\mathbf{y}_n) \odot \mathbf{q}_n\|_1. \end{aligned} \tag{C.5}$$

Using Jensen's inequality[1] we get

$$\begin{aligned} &\frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})} = \frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\sigma(\mathbf{Y})}(\mathbf{z})} \\ &= \frac{\mathbf{E}_{\mathbf{Q}_{n-1}} \exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{X}))\|_1}{\mathbf{E}_{\mathbf{q}_n}\mathbf{E}_{\mathbf{Q}_{n-1}} \exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{Y}))\|_1)} \\ &\leq \mathbf{E}_{\mathbf{q}_n} \frac{\mathbf{E}_{\mathbf{Q}_{n-1}} \exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1}}(\mathbf{X})\|_1)}{\mathbf{E}_{\mathbf{Q}_{n-1}} \exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{Y}))\|_1)} \\ &\leq \mathbf{E}_{\mathbf{q}_n} \frac{\mathbf{E}_{\mathbf{Q}_{n-1}} \exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\mathbf{Y})\|_1) \exp(\frac{1}{b}\|\frac{m}{r}\Phi(\mathbf{y}_n) \odot \mathbf{q}_n\|_1)}{\mathbf{E}_{\mathbf{Q}_{n-1}} \exp(-\frac{1}{b}\|\mathbf{z} - \overline{\mathbf{\Sigma}}_{\mathbf{Q}_{n-1},\mathbf{q}_n}(\sigma(\mathbf{Y}))\|_1)} \\ &= \mathbf{E}_{\mathbf{q}_n} \exp(\frac{1}{b}\frac{m}{r}\|\Phi(\mathbf{y}_n) \odot \mathbf{q}_n\|_1). \end{aligned}$$

1. All quantities are positive and $x \mapsto 1/x$ is convex on $\mathbb{R}_+$.

Conclusion Previous results hold for any dataset size $|\mathbf{X}| \in \mathbb{N}$. We now take the supremum over $\mathbf{X}, \mathbf{Y}$, which includes both cases $|\mathbf{X}| = |\mathbf{Y}| + 1$ and $|\mathbf{Y}| = |\mathbf{X}| + 1$; the supremum is the same in both cases, and we have the equality from the first case. Thus

$$\begin{aligned} \sup_{\mathbf{X},\mathbf{Y}\in\mathcal{D}:\mathbf{X}\overset{\cup}{\sim}\mathbf{Y}} \sup_{\mathbf{z}\in\mathcal{Z}} \frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})} &= \sup_{\mathbf{x}\in\mathcal{X}} \mathbf{E}_{\mathbf{q}_n} \exp\left(\frac{1}{b}\frac{m}{r}\|\Phi(\mathbf{x}) \odot \mathbf{q}_n\|_1\right) \\ &= \sup_{\mathbf{x}\in\mathcal{X}} \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b}Q_1^{\cup}(\mathbf{x}, \mathbf{q})\right), \end{aligned}$$

which concludes the proof.

Complex case If Φ is complex, the same proof holds using the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$. Indeed, an equivalent of (C.3) can be established using Definition 7.7 of a complex Laplace random variable. The triangle inequality (C.4) holds in a similar manner by considering complex and real parts independently, and $\tilde{\mathbf{z}}$ can be defined as $\tilde{\mathbf{z}} = M(\text{sign}(\Re\Phi(\mathbf{x}_n)) + i\,\text{sign}(\Im\Phi(\mathbf{x}_n)))$. We get

$$\begin{aligned} &\sup_{\mathbf{X},\mathbf{Y}\in\mathcal{D}:\mathbf{X}\overset{\cup}{\sim}\mathbf{Y}} \sup_{\mathbf{z}\in\mathcal{Z}} \frac{p_{\mathbf{X}}(\mathbf{z})}{p_{\mathbf{Y}}(\mathbf{z})} \\ &= \sup_{\mathbf{x}\in\mathcal{X}} \mathbf{E}_{\mathbf{q}_n} \exp\left(\frac{1}{b}\frac{m}{r}(\|\Re\Phi(\mathbf{x}) \odot \mathbf{q}_n\|_1 + \|\Im\Phi(\mathbf{x}) \odot \mathbf{q}_n\|_1)\right) \\ &= \sup_{\mathbf{x}\in\mathcal{X}} \mathbf{E}_{\mathbf{q}} \exp\left(\frac{1}{b}Q_1^{\cup}(\mathbf{x}, \mathbf{q})\right), \end{aligned}$$

which concludes the proof.

C.3.2 Random Fourier Features

Proof C.7 (Proof of Lemma 8.18): This proof bears strong similarities with the proof of Lemma 8.6, and we therefore use the same notations and tools. In particular, we recall that $f(\cdot) \triangleq \rho(\cdot) + \rho(\cdot - \pi/2)$, and that $\sup_{x\in\mathbb{R}} f(x) = \sqrt{2}$ for both complex exponential case and one-bit quantization. We also denote $\mathrm{supp}(\mathbf{q}) = \{j \in [\![1, m]\!] \,|\, h_j \neq 0\}$ the support of $\mathbf{q}$.

By analogy with Equations (8.4) and (8.5), but summing only on the frequencies that appear in the mask $\mathbf{q}$, denoting $f_{\mathbf{p},\mathbf{q}}(\mathbf{x}) \triangleq \sum_{j\in\mathrm{supp}(\mathbf{q})} f(\boldsymbol{\omega}_j^T\mathbf{x} - \varphi_j)$, the quantities $Q_1^{\mathrm{U}}(\mathbf{x},\mathbf{q})$ and $Q_1^{\mathrm{B}}(\mathbf{x},\mathbf{y},\mathbf{q})$ can be expressed as

$$\begin{aligned}
Q_1^{\mathrm{U}}(\mathbf{x},\mathbf{q}) &= \frac{m}{r}\sum_{j\in\mathrm{supp}(\mathbf{q})} |\rho(\boldsymbol{\omega}_j^T\mathbf{x} + u_j)| + |\rho(\boldsymbol{\omega}_j^T\mathbf{x} + u_j - \frac{\pi}{2})| \\
&= \frac{m}{r}\sup_{\mathbf{p}\in\{0,\pi/2,\pi,3\pi/2\}^m} f_{\mathbf{p}-\mathbf{u},\mathbf{q}}(\mathbf{x}). \\
Q_1^{\mathrm{B}}(\mathbf{x},\mathbf{y},\mathbf{q}) &= \frac{m}{r}\sum_{j\in\mathrm{supp}(\mathbf{q})} |\rho(\boldsymbol{\omega}_j^T\mathbf{x} + u_j) - \rho(\boldsymbol{\omega}_j^T\mathbf{y} + u_j)| \\
&\quad + |\rho(\boldsymbol{\omega}_j^T\mathbf{x} + u_j - \frac{\pi}{2}) - \rho(\boldsymbol{\omega}_j^T\mathbf{y} + u_j - \frac{\pi}{2})| \\
&= \frac{m}{r}\sup_{\mathbf{p}\in\{0,\pi/2,\pi,3\pi/2\}^m} f_{\mathbf{p}-\mathbf{u},\mathbf{q}}(\mathbf{x}) - f_{\mathbf{p}-\mathbf{u},\mathbf{q}}(\mathbf{y}) \\
&= \frac{m}{r}\sup_{\mathbf{p}\in\{0,\pi/2,\pi,3\pi/2\}^m} f_{\mathbf{p}-\mathbf{u},\mathbf{q}}(\mathbf{x}) + f_{\mathbf{p}-\mathbf{v},\mathbf{q}}(\mathbf{y}).
\end{aligned}$$

where $v_j = u_j + \pi$ for $1 \leq j \leq m$. The frequencies being nonresonant, a direct consequence of Lemma 8.5 is that for each $\mathbf{p} \in \mathbb{R}^m$, we have

$$\begin{aligned}
\sup_{\mathbf{x}\in\mathbb{R}^d}\inf_{\mathbf{q}\in\mathcal{Q}_r} f_{\mathbf{p},\mathbf{q}}(\mathbf{x}) &= \sup_{\mathbf{x}\in\mathbb{R}^d}\inf_{\mathbf{q}\in\mathcal{Q}_r}\sum_{j\in\mathrm{supp}(\mathbf{q})} f(\boldsymbol{\omega}_j^T\mathbf{x} - \varphi_j) \\
&= r\sup_{x\in\mathbb{R}} f(x) = r\sqrt{2}.
\end{aligned}$$

The supremum being independent of $\mathbf{p}$ this yields

$$\begin{aligned}
&\sup_{\mathbf{x}\in\mathbb{R}^d}\inf_{\mathbf{q}\in\mathcal{Q}_r}\sup_{\mathbf{p}\in\{0,\pi/2,\pi,3\pi/2\}^m} f_{\mathbf{p}-\mathbf{u},\mathbf{q}}(\mathbf{x}) \\
&\qquad \geq \sup_{\mathbf{p}\in\{0,\pi/2,\pi,3\pi/2\}^m}\sup_{\mathbf{x}\in\mathbb{R}^d}\inf_{\mathbf{q}\in\mathcal{Q}_r} f_{\mathbf{p}-\mathbf{u},\mathbf{q}}(\mathbf{x}) = r\sqrt{2}.
\end{aligned}$$

As we also have (even for resonant frequencies) the upper bound

$$\sup_{\mathbf{x}\in\mathbb{R}^d}\inf_{\mathbf{q}\in\mathcal{Q}_r}\sup_{\mathbf{p}\in\mathbb{R}^m} f_{\mathbf{p},\mathbf{q}}(\mathbf{x}) \leq \sup_{\mathbf{x}\in\mathbb{R}^d}\sup_{\mathbf{q}\in\mathcal{Q}_r}\sup_{\mathbf{p}\in\mathbb{R}^m} f_{\mathbf{p},\mathbf{q}}(\mathbf{x}) \leq r\sqrt{2}$$

we get for each $\mathbf{q} \in \mathcal{Q}_r$

$$\begin{aligned}
m\sqrt{2} &\leq \sup_{\mathbf{x}\in\mathbb{R}^d}\inf_{\mathbf{q}'\in\mathcal{Q}_r} Q_1^{\mathrm{U}}(\mathbf{x},\mathbf{q}') \leq \sup_{\mathbf{x}\in\mathbb{R}^d} Q_1^{\mathrm{U}}(\mathbf{x},\mathbf{q}) \\
&\leq \sup_{\mathbf{x}\in\mathbb{R}^d}\sup_{\mathbf{q}'\in\mathcal{Q}_r} Q_1^{\mathrm{U}}(\mathbf{x},\mathbf{q}') = m\sqrt{2}.
\end{aligned}$$

In the BDP setting, the supremum is taken independently on $\mathbf{x}$ and $\mathbf{y}$, thus for any $\mathbf{q}$ we have $\sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^d} Q_1^{\mathrm{B}}(\mathbf{x},\mathbf{y},\mathbf{q}) =$

$2\sup_{\mathbf{x}\in\mathbb{R}^d} Q_1^{\mathrm{U}}(\mathbf{x},\mathbf{q})$ and

$$2m\sqrt{2} = \sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^d} \inf_{\mathbf{q}'\in\mathcal{Q}_r} Q_1^{\mathrm{B}}(\mathbf{x},\mathbf{y},\mathbf{q}') \leq \sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^d} Q_1^{\mathrm{B}}(\mathbf{x},\mathbf{y},\mathbf{q}) \leq \sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^d} \sup_{\mathbf{q}'\in\mathcal{Q}_r} Q_1^{\mathrm{B}}(\mathbf{x},\mathbf{y},\mathbf{q}') = 2m\sqrt{2}.$$

C.4 Derivation of the noise-signal ratio

Lemma C.8

Let X denote the mean of n' samples taken without replacement from a collection $x_1,...,x_n$. Let $\sigma^2 = \frac{1}{n}\sum_{i=1}^n |x_i-\mu|^2$, then we have

$$\mathrm{Var}(X) = \frac{\sigma^2}{n'}\frac{n-n'}{n-1}.$$

Proof C.9: Denote $X = \frac{1}{n}\sum_{i=1}^n g_i x_i$, with $g_i = 1$ if x_i is selected, and 0 otherwise. Note that as a consequence, $\sum_{i=1}^n g_i = n'$. For any $1 \leq i < j \leq n$, the marginal of g_i is uniform and $\mathbf{E}(g_i g_j) = P[g_i g_j = 1] = P[g_i = 1 \text{ and } g_j = 1] = P[z = 2]$ for z a random variable having an hypergeometric law of parameters $(n, 2/n, n')$.

$$\begin{aligned}
\mathrm{Var}(g_i) &= \mathbf{E}|g_i|^2 - |\mathbf{E}g_i|^2 = \frac{n'}{n}(1-\frac{n'}{n}) = \frac{n'(n-n')}{n^2} \\
\mathrm{Cov}(g_i,g_j) &= \mathbf{E}(g_i g_j) - \mathbf{E}(g_i)\mathbf{E}(g_j) \\
&= \frac{n'(n'-1)}{n(n-1)} - \frac{(n')^2}{n^2} \text{ (hypergeometric law)} \\
&= \frac{n'}{n}\left(\frac{n'-1}{n-1} - \frac{n'}{n}\right) = \frac{n'(n'-n)}{n^2(n-1)}.
\end{aligned}$$

We also have

$$\begin{aligned}
&\mathrm{Var}\left(\frac{1}{n'}\sum_{i=1}^n g_i x_i\right) \\
&= \frac{1}{(n')^2}\left(\sum_{i=1}^n \mathrm{Var}_{\mathbf{g}}(x_i g_i) + 2\sum_{1\leq i<j\leq n}\mathrm{Cov}(x_i g_i, x_j g_j)\right) \\
&= \frac{1}{(n')^2}\left(\frac{n'(n-n')}{n^2}\sum_{i=1}^n x_i^2 + 2\frac{n'(n'-n)}{n^2(n-1)}\sum_{1\leq i<j\leq n} x_i x_j\right) \\
&= \frac{1}{n'}\frac{n-n'}{n^2}\left(\sum_{i=1}^n x_i^2 - 2\frac{1}{n-1}\sum_{1\leq i<j\leq n} x_i x_j\right).
\end{aligned}$$

Let $\mu = \frac{1}{n}\sum_{i=1}^n x_i$, and $\sigma^2 = \frac{1}{n}\sum_{i=1}^n (x_i - \mu)^2$. Note that

$$\begin{aligned} n\sigma^2 &= \sum_{i=1}^n (x_i - \mu)^2 \\ &= \sum_{i=1}^n x_i^2 - n\mu^2 \\ &= \sum_{i=1}^n x_i^2 - \frac{1}{n}(\sum x_i^2 + 2\sum_{i<j} x_i x_j) \\ &= \frac{n-1}{n}\left(\sum x_i^2 - 2\frac{1}{n-1}\sum_{i<j} x_i x_j\right). \end{aligned}$$

As a consequence

$$\begin{aligned} \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^n g_i x_i\right) &= \frac{1}{n'}\frac{n-n'}{n^2}\frac{n^2}{n-1}\sigma^2 \\ &= \frac{\sigma^2}{n'}\frac{n-n'}{n-1}. \end{aligned}$$

We can now give the proof.

Proof C.10 (Proof of Lemma 9.1):

We define the error as $\mathbf{e} \triangleq \mathbf{z}(\mathbf{X}) - \mathbf{z}$ for some reference signal $\mathbf{z}$, which can be either $\tilde{\mathbf{s}}$ or the true sketch $\mathbf{s}$. The noise level is $\mathbf{E}\|\mathbf{e}\|_2^2$, and the noise-to-signal ratio is defined as $\mathrm{NSR} = \mathbf{E}\|\mathbf{e}\|_2^2/\|\mathbf{z}\|_2^2$. In these expressions, the expectations are taken w.r.t. the randomness of the sketching mechanism when $\tilde{\mathbf{s}}$ is chosen as the reference signal, and w.r.t. both the randomness of the mechanism and the draw of $\mathbf{X}$ when $\mathbf{s}$ is the reference signal. We denote $\boldsymbol{\Sigma}$ the clean sum of features, $\boldsymbol{\Sigma}_{n'}$ the sum of features computed on a random subset of the collection, $\boldsymbol{\Sigma}_{\mathbf{Q},n'}$ the mechanism combining both types of subsampling, i.e.

$$\begin{aligned} \boldsymbol{\Sigma}(\mathbf{X}) &= \sum_{i=1}^n \Phi(\mathbf{x}_i) \\ \boldsymbol{\Sigma}_{n'}(\mathbf{X}) &= \sum_{i=1}^n g_i \Phi(\mathbf{x}_i) \\ \boldsymbol{\Sigma}_{\mathbf{Q},n'}(\mathbf{X}) &= \frac{m}{r}\sum_{i=1}^n g_i(\Phi(\mathbf{x}_i) \odot \mathbf{q}_i) \\ \mathbf{z}(\mathbf{X}) &\overset{\text{(i)}}{=} \frac{1}{n'}(\boldsymbol{\Sigma}_{\mathbf{Q},n'}(\mathbf{X}) + \boldsymbol{\xi}). \end{aligned}$$

(i) We recall here that n is assumed public, so no noise is added on the denominator.

Thus the error can be decomposed as

$$\begin{aligned} \mathbf{e} &= \frac{1}{n'}\widetilde{\boldsymbol{\Sigma}}(\mathbf{X}) - \mathbf{z} \\ &= \underbrace{\frac{1}{n}\Sigma(\mathbf{X}) - \mathbf{z}}_{\mathbf{e}_{\mathbf{X}}} + \underbrace{\tfrac{1}{n'}\boldsymbol{\Sigma}_{n'}(\mathbf{X}) - \tfrac{1}{n}\Sigma(\mathbf{X})}_{\mathbf{e}_{n'}} \\ &\quad + \underbrace{\tfrac{1}{n'}\big(\boldsymbol{\Sigma}_{\mathbf{Q},n'}(\mathbf{X}) - \boldsymbol{\Sigma}_{n'}(\mathbf{X})\big)}_{\mathbf{e}_{\mathbf{Q}}} + \underbrace{\tfrac{1}{n'}\boldsymbol{\xi}}_{\mathbf{e}_{\xi}}. \end{aligned}$$

We now estimate the noise level of each of these components

separately.

Without noise nor subsampling. When no noise is added ($\boldsymbol{\xi} = \zeta = 0$), and all features of all samples are used ($r = m$, no subsampling), then $\mathbf{z}(\mathbf{X}) = \tilde{\mathbf{s}} = \mathcal{A}(\pi_{\mathbf{X}}) = \frac{\boldsymbol{\Sigma}(\mathbf{X})}{|\mathbf{X}|}$. When the true sketch is chosen as the reference signal, we have:

$$\begin{aligned}
\mathbf{e}_{\mathbf{X}} &= \frac{1}{n}\sum \Phi(\mathbf{x}_i) - \mathbf{s} \\
\mathbf{E}_{\mathbf{X}}\, \mathbf{e}_{\mathbf{X}} &= 0 \\
\|\mathbf{e}_{\mathbf{X}}\|_2^2 &= \|\tilde{\mathbf{s}} - \mathbf{s}\|_2^2 \\
\mathbf{E}_{\mathbf{X}}\, \|\mathbf{e}_{\mathbf{X}}\|_2^2 &= \sum_{j=1}^{m} \mathrm{Var}_{\mathbf{X}}\left(\frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i)_j\right) = \frac{1}{n}\sum_{j=1}^{m} \mathrm{Var}_{\mathbf{x}}(\Phi(\mathbf{x})_j) \\
\boxed{\mathbf{E}_{\mathbf{X}}\, \|\mathbf{e}_{\mathbf{X}}\|_2^2 = \frac{1}{n}\left(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|^2 - \|\mathbf{s}\|^2\right)}
\end{aligned}$$

If $\tilde{\mathbf{s}}$ is chosen as the reference signal, then $\mathbf{e}_{\mathbf{X}} = 0$, $\boxed{\mathbf{E}\|\mathbf{e}_{\mathbf{X}}\|_2^2 = 0}$.

Additive noise (for privacy). The noise contribution due to the additive noise is $\mathbf{e}_\xi = \boldsymbol{\xi}/n'$, thus

$$\begin{aligned}
\mathbf{E}_\xi \mathbf{e}_\xi &= 0 \\
\mathbf{E}_\xi \|\mathbf{e}_\xi\|_2^2 &= \frac{1}{(n')^2} m \mathbf{E}\left[\xi_i^2\right] \\
\boxed{\mathbf{E}_\xi \|\mathbf{e}_\xi\|_2^2 = \frac{m}{(n')^2}\sigma_\xi^2}
\end{aligned}$$

and is independent from the reference signal. Here σ_ξ^2 is the noise level such that the whole mechanism[2] is ε-DP. It is thus computed using a privacy level $\varepsilon' = \log(1 + (\exp(\varepsilon) - 1)/\beta)$.

2. Including the sampling step.

Samples subsampling We consider here the noise contribution due to the dataset subsampling operation. We have

$$\begin{aligned}
\mathbf{e}_{n'} &= \tfrac{1}{n'}\boldsymbol{\Sigma}_{n'}(\mathbf{X}) - \tfrac{1}{n}\boldsymbol{\Sigma}(\mathbf{X}) \\
\mathbf{E}_{\mathbf{g}} \mathbf{e}_{n'} &= 0
\end{aligned}$$

The noise level here depends on the subsampling strategy. We consider two cases

- sampling of n' samples out of n without replacement (denoted

$\mathrm{WOR}(n, n')$):

$$\begin{aligned}\mathbf{E}_{\mathbf{g}\sim\mathrm{WOR}(n,n')}\|\mathbf{e}_{n'}\|^2 &= \sum_{j=1}^{m}\mathrm{Var}_{\mathbf{g}}\left(\frac{1}{n'}\sum_{i=1}^{n} g_i\Phi(\mathbf{x}_i)_j\right)\\ &\overset{\text{(i)}}{=} \sum_{j=1}^{m}\left(\frac{\sum_{i=1}^{n}|\Phi(\mathbf{x}_i)_j-(\mathbf{z}_{\mathbf{X}})_j|^2}{nn'}\frac{n-n'}{n-1}\right)\\ &= \frac{1}{nn'}\frac{n-n'}{n-1}\sum_{i=1}^{n}\|\Phi(\mathbf{x}_i)-\mathbf{z}_{\mathbf{X}}\|_2^2\end{aligned}$$

(i) By Lemma C.8.

$$\boxed{\mathbf{E}_{\mathbf{g}\sim\mathrm{WOR}(n,n')}\|\mathbf{e}_{n'}\|^2 = \frac{1}{n-1}\left(\frac{n}{n'}-1\right)\left(\frac{1}{n}\sum_{i=1}^{n}\|\Phi(\mathbf{x}_i)\|_2^2-\|\mathbf{z}_{\mathbf{X}}\|_2^2\right)}$$

Taking the expectation with respect to the draw of $\mathbf{X}$ as well we obtain

$$\begin{aligned}&\mathbf{E}_{\mathbf{X}}\mathbf{E}_{\mathbf{g}\sim\mathrm{WOR}(n,n')}\|\mathbf{e}_{n'}\|^2\\ &\quad= \frac{1}{n-1}\left(\frac{n}{n'}-1\right)(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|_2^2-\mathbf{E}_{\mathbf{X}}\|\mathbf{z}_{\mathbf{X}}\|_2^2)\\ &\quad= \frac{1}{n-1}\left(\frac{n}{n'}-1\right)\left(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|_2^2-(\|\mathbf{s}\|_2^2+\sum_{j=1}^{m}\mathrm{Var}((\mathbf{z}_{\mathbf{X}})_j))\right)\\ &\quad= \frac{1}{n-1}\left(\frac{n}{n'}-1\right)\left(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|_2^2-(\|\mathbf{s}\|_2^2+\frac{1}{n}(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|^2-\|\mathbf{s}\|^2))\right)\\ &\quad= \frac{1}{n-1}\left(\frac{n}{n'}-1\right)(1-\frac{1}{n})(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|_2^2-\|\mathbf{s}\|_2^2)\end{aligned}$$

$$\boxed{\mathbf{E}_{\mathbf{X}}\mathbf{E}_{\mathbf{g}\sim\mathrm{WOR}(n,n')}\|\mathbf{e}_{n'}\|^2 \quad= \frac{1}{n}\left(\frac{n}{n'}-1\right)(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|_2^2-\|\mathbf{s}\|_2^2)}$$

- i.i.d. Bernoulli sampling with parameter β:

$$\begin{aligned}\mathbf{E}_{\mathbf{g}\sim\mathrm{Bern}(\beta)^n}\|\mathbf{e}_{n'}\|^2 &= \sum_{j=1}^{m}\mathrm{Var}_{\mathbf{g}}\left(\frac{1}{\beta n}\sum_{i=1}^{n} g_i\Phi(\mathbf{x}_i)_j\right)\\ &= \frac{1}{\beta^2 n^2}\sum_{j=1}^{m}\sum_{i=1}^{n}|\Phi(\mathbf{x}_i)_j|^2\mathrm{Var}_{\mathbf{g}}(g_i)\end{aligned}$$

$$\boxed{\mathbf{E}_{\mathbf{g}\sim\mathrm{Bern}(\beta)^n}\|\mathbf{e}_{n'}\|^2 = \frac{1}{n}\left(\frac{1}{\beta}-1\right)\left(\frac{1}{n}\sum_{i=1}^{n}\|\Phi(\mathbf{x}_i)\|^2\right)}$$

Taking the expectation with respect to the draw of $\mathbf{X}$ as well we obtain

$$\boxed{\mathbf{E}_{\mathbf{X}}\mathbf{E}_{\mathbf{g}\sim\mathrm{Bern}(\beta)^n}\|\mathbf{e}_{n'}\|^2 = \frac{1}{n}\left(\frac{1}{\beta}-1\right)\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|_2^2}$$

Frequencies subsampling. We define the noise contribution due to frequency subsampling as

$$\begin{aligned}
\mathbf{e}_{\mathbf{Q}} &= \frac{1}{n'}\left(\mathbf{\Sigma}_{\mathbf{Q},n'}(\mathbf{X}) - \mathbf{\Sigma}_{n'}(\mathbf{X})\right) \\
\text{where:} \quad \mathbf{\Sigma}_{\mathbf{Q},n'} &= \frac{m}{r}\sum_{i=1}^{n} g_i(\Phi(\mathbf{x}_i)\odot\mathbf{q}_i) \\
\mathbf{\Sigma}_{n'} &= \sum_{i=1}^{n} g_i\Phi(\mathbf{x}_i) \\
\mathbf{E}_{\mathbf{Q}}\|\mathbf{e}_{\mathbf{Q}}\|_2^2 &= \frac{1}{n'}\Big(\frac{m}{r}-1\Big)\frac{1}{n'}\sum_{i=1}^{n} g_i\|\Phi(\mathbf{x}_i)\|^2
\end{aligned}$$

Recall that the masks entries are in $\{0,1\}$, thus $\forall i,j\,(\mathbf{q}_i)_j^2 = (\mathbf{q}_i)_j$, but also $\forall j\,\mathbf{E}_{\mathbf{q}\sim p_{\mathbf{q}}}\,\mathbf{q}_j = r/m$ because $p_{\mathbf{q}} \in \mathcal{P}_r$. Therefore we have

$$\begin{aligned}
\mathrm{Var}(\mathbf{q}_i)_j &= \mathbf{E}\,((\mathbf{q}_i)_j)^2 - (\mathbf{E}\,(\mathbf{q}_i)_j)^2 \\
&= \mathbf{E}\,(\mathbf{q}_i)_j - \Big(\frac{r}{m}\Big)^2 = \frac{r}{m}(1-\frac{r}{m})
\end{aligned}$$

As a result

$$\begin{aligned}
|n'|^2\mathbf{E}_{\mathbf{Q}}\|\mathbf{e}_{\mathbf{Q}}\|_2^2 &= \mathbf{E}_{\mathbf{Q}}\left\|\frac{m}{r}\sum_{i=1}^{n} g_i(\Phi(\mathbf{x}_i)\odot\mathbf{q}_i) - \sum_{i=1}^{n} g_i\Phi(\mathbf{x}_i)\right\|_2^2 \\
&= \sum_{j=1}^{m}\mathrm{Var}_{\mathbf{Q}}\left(\frac{m}{r}\sum_{i=1}^{n} g_i\Phi(\mathbf{x}_i)_j(\mathbf{q}_i)_j\right) \\
&= \Big(\frac{m}{r}\Big)^2\sum_{j=1}^{m}\sum_{i=1}^{n} g_i|\Phi(\mathbf{x}_i)_j|^2\mathrm{Var}\big((\mathbf{q}_i)_j\big) \\
\mathbf{E}_{\mathbf{Q}}\|\mathbf{e}_{\mathbf{Q}}\|_2^2 &= \frac{1}{n'}\Big(\frac{m}{r}-1\Big)\frac{1}{n'}\sum_{i=1}^{n} g_i\|\Phi(\mathbf{x}_i)\|^2 \\
\boxed{\mathbf{E}_{\mathbf{g}}\mathbf{E}_{\mathbf{Q}}\|\mathbf{e}_{\mathbf{Q}}\|_2^2 &= \frac{1}{n'}\Big(\frac{m}{r}-1\Big)\frac{1}{n}\sum_{i=1}^{n}\|\Phi(\mathbf{x}_i)\|^2.}
\end{aligned}$$

Taking the expectation w.r.t. the dataset, we get

$$\boxed{\mathbf{E}_{\mathbf{X}}\mathbf{E}_{\mathbf{g}}\mathbf{E}_{\mathbf{Q}}\|\mathbf{e}_{\mathbf{Q}}\|_2^2 = \frac{1}{n'}\Big(\frac{m}{r}-1\Big)\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|^2.}$$

Total noise level For conciseness, we use the notation $\beta = n'/n$ when sampling n' samples without replacement, and $\alpha = r/m$. The total noise level for Bernoulli sampling is, first with respect to $\mathbf{s}$:

$$\begin{aligned}
\mathbf{E}_{\mathbf{X},\mathbf{g},\mathbf{Q},\boldsymbol{\xi}}\|\mathbf{e}\|_2^2 &= \mathbf{E}_{\mathbf{X},\mathbf{g},\mathbf{Q},\boldsymbol{\xi}}\left(\|\mathbf{e}_{\mathbf{X}}\|_2^2 + \|\mathbf{e}_{\boldsymbol{\xi}}\|_2^2 + \|\mathbf{e}_{n'}\|_2^2 + \|\mathbf{e}_{\mathbf{Q}}\|_2^2\right) \\
&= \frac{1}{n}\big(\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|^2 - \|\mathbf{s}\|^2\big) + \frac{m}{(n')^2}\sigma_\xi^2 \\
&\quad + \frac{1}{n}\Big(\frac{1}{\beta}-1\Big)\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|_2^2 + \frac{1}{n'}\Big(\frac{1}{\alpha}-1\Big)\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|^2 \\
&= \frac{1}{\beta n}\frac{1}{\alpha}\mathbf{E}_{\mathbf{x}}\|\Phi(\mathbf{x})\|^2 - \frac{1}{n}\|\mathbf{s}\|^2 + \frac{m}{n^2\beta^2}\sigma_\xi^2.
\end{aligned}$$

With respect to $\tilde{\mathbf{s}}$, we obtain:

$$\begin{aligned}\mathbf{E}_{\mathbf{g},\mathbf{Q},\boldsymbol{\xi}} \|\mathbf{e}\|_2^2 &= \mathbf{E}_{\mathbf{g},\mathbf{Q},\boldsymbol{\xi}} \|\mathbf{e}_\xi\|_2^2 + \|\mathbf{e}_{n'}\|_2^2 + \|\mathbf{e}_\mathbf{Q}\|_2^2 \\ &= \frac{m}{n^2\beta^2}\sigma_\xi^2 + \frac{1}{n}\left(\frac{1}{\beta}\frac{1}{\alpha} - 1\right)\left(\frac{1}{n}\sum_{i=1}^n \|\Phi(\mathbf{x}_i)\|^2\right).\end{aligned}$$

For WOR sampling, we get with respect to $\mathbf{s}$:

$$\begin{aligned}\mathbf{E}_{\mathbf{X},\mathbf{g},\mathbf{Q},\boldsymbol{\xi}} \|\mathbf{e}\|_2^2 &= \mathbf{E}_{\mathbf{X},\mathbf{g},\mathbf{Q},\boldsymbol{\xi}} \left(\|\mathbf{e}_\mathbf{X}\|_2^2 + \|\mathbf{e}_\xi\|_2^2 + \|\mathbf{e}_{n'}\|_2^2 + \|\mathbf{e}_\mathbf{Q}\|_2^2\right) \\ &= \frac{1}{n'}\frac{1}{\alpha}\mathbf{E}_\mathbf{x}\|\Phi(\mathbf{x})\|^2 - \frac{1}{n'}\|\mathbf{s}\|^2 + \frac{m}{(n')^2}\sigma_\xi^2.\end{aligned}$$

And with respect to $\tilde{\mathbf{s}}$:

$$\begin{aligned}\mathbf{E}_{\mathbf{g},\mathbf{Q},\boldsymbol{\xi}} \|\mathbf{e}\|_2^2 &= \mathbf{E}_{\mathbf{g},\mathbf{Q},\boldsymbol{\xi}} \|\mathbf{e}_\xi\|_2^2 + \|\mathbf{e}_{n'}\|_2^2 + \|\mathbf{e}_\mathbf{Q}\|_2^2 \\ &= \frac{m}{(n')^2}\sigma_\xi^2 + \frac{1}{n-1}\left(\frac{1}{\beta} - 1\right)\left(\frac{1}{n}\sum_{i=1}^n \|\Phi(\mathbf{x}_i)\|_2^2 - \|\mathbf{z}_\mathbf{X}\|_2^2\right) \\ &\quad + \frac{1}{n'}\left(\frac{1}{\alpha} - 1\right)\frac{1}{n}\sum_{i=1}^n \|\Phi(\mathbf{x}_i)\|^2 \\ &= \frac{m}{n^2\beta^2}\sigma_\xi^2 + \left(\frac{1}{n-1}\left(\frac{1}{\beta} - 1\right)\right. \\ &\quad \left. + \frac{1}{\beta n}\left(\frac{1}{\alpha} - 1\right)\right)\left(\frac{1}{n}\sum_{i=1}^n \|\Phi(\mathbf{x}_i)\|_2^2\right) \\ &\quad - \frac{1}{n-1}\left(\frac{1}{\beta} - 1\right)\|\mathbf{z}_\mathbf{X}\|_2^2.\end{aligned}$$

Proof C.11 (Proof of Lemma 9.2):

We rewrite $\mathbf{z}(\mathbf{X}) = f(|\mathbf{X}| + \zeta)\overline{\boldsymbol{\Sigma}}(\mathbf{X})$, where $f(|\mathbf{X}| + \zeta)$ is an estimator of $1/|\mathbf{X}|$. We define the reference signal as $\mathbf{z} = \mathbf{s}$ or $\tilde{\mathbf{s}}$, and the noise as $\mathbf{e} = f(|\mathbf{X}| + \zeta)\overline{\boldsymbol{\Sigma}}(\mathbf{X}) - \mathbf{z}$. In the following, the expectations are taken w.r.t. the randomness of the sketching mechanism when $\tilde{\mathbf{s}}$ is chosen as the reference signal, and w.r.t. both the randomness of the mechanism and the draw of $\mathbf{X}$ when $\mathbf{s}$ is the reference signal. We have $\mathbf{E}\mathbf{e} = 0$.

$$\begin{aligned}\mathbf{E}\,\|\mathbf{e}\|_2^2 &= \sum_{j=1}^m \mathrm{Var}(\mathbf{e}_j) = \sum_{j=1}^m \mathrm{Var}\Big(f(|\mathbf{X}| + \zeta)\overline{\boldsymbol{\Sigma}}(\mathbf{X})_j\Big) \\ &= \sum_{j=1}^m \left[\mathbf{E}(f^2)\mathrm{Var}\Big(\overline{\boldsymbol{\Sigma}}(\mathbf{X})_j\Big) + \mathrm{Var}(f)\big|\mathbf{z}_j\big|^2 n^2\right] \\ &= \mathbf{E}(f^2)n^2\mathbf{E}\|\widetilde{\boldsymbol{\Sigma}}(\mathbf{X})/n - \mathbf{z}\|_2^2 + \mathrm{Var}(f)\|\mathbf{z}\|^2 n^2\end{aligned}$$

Thus the noise-to-signal ratio NSR_ζ of the whole mechanism (including noise ζ) can be written as a function of the noise-to-signal ratio of $\widetilde{\boldsymbol{\Sigma}}(\mathbf{X})/n$ as computed in Lemma 9.1 (i.e. using the same parameters but without ζ), which we denote simply NSR in the

rest of the proof.

$$\begin{aligned}\mathrm{NSR}_\zeta &= \mathbf{E}(f^2)n^2\mathrm{NSR} + \mathrm{Var}(f)n^2 \\ &= ((\mathbf{E}f)^2 + \mathrm{Var}(f))n^2\mathrm{NSR} + (\mathbf{E}f)^2 n^2 \frac{\mathrm{Var}(f)}{(\mathbf{E}f)^2} \\ &= (\mathbf{E}f)^2 n^2 \left[\left(1 + \frac{\mathrm{Var}(f)}{(\mathbf{E}f)^2}\right)(\mathrm{NSR}+1) - 1\right]. \qquad \text{(C.6)}\end{aligned}$$

For an unbiased estimator f (if there exists any), we have $(\mathbf{E}f)^2 = 1/n^2$ and the variance can be bounded via a Cramer-Rao bound.

A bound on the variance of f. Remember that ζ is drawn as $\zeta \sim \mathcal{L}(0, b)$. We want to estimate $\theta = 1/n$ from an observation x drawn with probability density (and log-density)

$$\begin{aligned}p_\theta(x) &= \frac{1}{2b_\zeta} e^{-\frac{|x-\frac{1}{\theta}|}{b_\zeta}} \\ \log(p_\theta(x)) &= -\log(2b_\zeta) - \frac{1}{b_\zeta}|x - \frac{1}{\theta}|.\end{aligned}$$

Using the Cramer-Rao bound for an unbiased estimator f, we have

$$\begin{aligned}\mathrm{Var}(f) &\geq \mathbf{E}\left[\left(\frac{d(\log p_\theta(x))}{d\theta}\right)^2\right]^{-1} \\ &= \left[\int_x \left(\frac{\mathrm{sign}(\frac{1}{\theta} - x)}{b_\zeta \theta^2}\right)^2 p_\theta(x)dx\right]^{-1} \\ &= \left[\int_x \left(\frac{1}{b_\zeta \theta^2}\right)^2 p_\theta(x)dx\right]^{-1} \\ &= b_\zeta^2\theta^4 = b_\zeta^2/n^4 = \sigma_\zeta^2/(2n^4) = (\mathbf{E}f)^2\sigma_\zeta^2/(2n^2).\end{aligned}$$

Conclusion Combining this bound with Equation (C.6) yields for an unbiased estimator of minimal variance (if there exists any)

$$\mathrm{NSR}_\zeta \geq \left(1 + \frac{\sigma_\zeta^2}{2n^2}\right)(\mathrm{NSR}+1) - 1.$$

C.5 Heuristic for Splitting the Privacy Budget

Proof C.12 (Proof of Lemma 9.3): The noise level for ζ is $b_\zeta = 1/\varepsilon_\zeta = 1/((1-\gamma)\varepsilon)$ for Laplacian noise according to Lemma 8.2. In the Laplacian-UDP setting, the lowest noise level yielding ε-DP is $\sigma_\xi = 2b = 2\sqrt{2}m/(\gamma\varepsilon)$ (complex Laplace distribution). We

then have

$$\mathrm{NSR}_*^{\mathrm{RFF}} = \left(1 + \frac{1}{n^2(1-\gamma)^2\varepsilon^2}\right)\left(1 - \frac{1}{n} + \frac{m}{n\|\mathbf{s}\|^2}\left(\frac{1}{\alpha} + \frac{1}{n}\frac{8m^2}{\gamma^2\varepsilon^2}\right)\right) - 1,$$

For succinctness in the derivation, denote $A = 1/(n^2\varepsilon^2)$, $B = 1 - 1/n + m^2/(nr\|\mathbf{s}\|^2)$ and $C = \frac{1}{n^2\|\mathbf{s}\|^2}\frac{8m^3}{\varepsilon^2}$, so that we try to minimize

$$\mathrm{NSR}_*^{\mathrm{RFF}} = \left(1 + \frac{A}{(1-\gamma)^2}\right)\left(B + \frac{C}{\gamma^2}\right) - 1$$

Note that $\mathrm{NSR}_*^{\mathrm{RFF}}$ diverges to $+\infty$ when $\gamma \to 0_+$ or $\gamma \to 1_-$, but is continuous on $]0, 1[$. Any minimizer on $]0, 1[$ must cancel the quantity

$$\begin{aligned}\frac{1}{2C}\gamma^3(1-\gamma)^3\frac{d\mathrm{NSR}_*^{\mathrm{RFF}}}{d\gamma}(\gamma) &= \frac{1}{C}A\gamma^3\left(B + \frac{C}{\gamma^2}\right) \\ &\quad - \frac{1}{C}C(1-\gamma)^3\left(1 + \frac{A}{(1-\gamma)^2}\right) \\ &= AB/C\gamma^3 + A\gamma - (1-\gamma)^3 - A(1-\gamma) \\ &= (AB/C + 1)\gamma^3 - 3\gamma^2 \\ &\quad + (2A+3)\gamma - (A+1)\end{aligned}$$

where $AB/C = \left(1 - \frac{1}{n} + \frac{m^2}{nr\|\mathbf{s}\|^2}\right)\frac{\|\mathbf{s}\|^2}{m}\frac{1}{8m^2} \ll 1$. Note that, if we start from the expression of the NSR which takes $\tilde{\mathbf{s}}$ as a reference signal, we would get $B = 1 - 1/n + m^2/(nr\|\mathbf{s}\|^2)$, but the same approximation would still hold. The only real root of $\gamma^3 - 3\gamma^2 + (2A+3)\gamma - (A+1)$ can be computed as $\gamma^* = 1 - \frac{1}{3}E + \frac{2A}{E} = 1 + \frac{6A - E^2}{3E}$, where

$$E = \frac{1}{2^{1/3}}\left(27A + 3\sqrt{81A^2 + 96A^3}\right)^{1/3}$$

In this setting where $\varepsilon \ll 1/n$, $A \gg 1$ and we can use the following approximation.

$$\gamma^* = 1 + \frac{6A - E^2}{3E} \approx 1 + \frac{6A - 6A\left(1 + \frac{27A}{3\sqrt{96}A^{3/2}}\right)^{2/3}}{\sqrt{6}A^{1/2}} \approx \frac{1}{2}.$$

On the other side, if $A \ll 1$, we get

$$E \approx 3A^{1/3} \text{ and } \gamma^* = 1 + \frac{6A - E^2}{3E} \approx 1 + \frac{6A - 9A^{2/3}}{9A^{1/3}} \approx 1 - A^{1/3}.$$

Chapter D

Implementation details

A significant amount of time[1] was allocated during the thesis to the implementation of the different methods presented in this manuscript. We briefly summarize here these different contributions, which have all been coded in the Julia language[2]. Some Matlab code has been written at the beginning of thesis as well (based on the SketchMLbox toolbox[3]), but we do not discuss the Matlab implementation here.

[1] It certainly took more time than expected, as could have been expected from Hofstadter's Law.

[2] See `https://julialang.org/`.

[3] http://sketchml.gforge.inria.fr/

D.1 The CompressiveLearning package

All the code related to the compressive learning framework has been packaged together. The implemented features are the following:

- Sketching operators for random Fourier and quadratic features;
- Main wrapper methods for compressive clustering (`CKM`), Gaussian mixture modeling (`CGMM`) and compressive PCA (`CPCA`);
- CL-OMPR decoder for `CKM` and `CGMM`;
- CL-AMP decoder for `CKM`;
- Various decoders for `CPCA`;
- Support of structured linear operators (with blocks of the form $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$, $\mathbf{HD}_G\mathbf{HD}_2\mathbf{HD}_1$ and fastfood), with padding;
- Support of quantized random Fourier features, and decoding in CLOMPR with quantization;
- UDP/BDP private sketching mechanisms;
- Sketching mechanism with features subsampling.

D.2 The FatDatasets package

The FatDatasets package does not contain any data, but defines data dependencies[4] which can be used to download and preprocess the data. A summary of the datasets available via the package is provided in Table D.1. The following methods are also provided for synthetic data generation:

[4] Using the DataDeps package.

- `GMMDataset` can be used to generate data according to (4.5);
- `lowrank_dataset` can be used to generate (approximately) low-rank data (cf. privacy experiments);
- `gmm_stream` and `lowrank_stream` generate data according to

the same models as respectively `GMMDataset` and `lowrank_dataset`, but the data is never stored and only generated when requested. Randomization seeds are stored to allow multiple iterations over the same data stream.

Name	n	d	Description and url
fma	$106,574$	518	Miscellaneous aggregated audio features. `https://github.com/mdeff/fma`
fma_mfcc	$106,574$	20	Contains mean MFCC features from FMA dataset. `https://github.com/mdeff/fma`
kddcup99	$4,898,431$	107	Network data. Contains categorical features (converted to numerical features via binary encoding). `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`
gowalla	$6,442,892$	2	Localization (GPS) data. `https://snap.stanford.edu/data/loc-gowalla.html`
fasttext_LG	$2,000,000$	300	Dataset of textual features. LG should be one of: "en" (english), "fr" (french), "de" (german), "eo" (esperanto). `https://fasttext.cc/docs/en/crawl-vectors.html`
lfw	13233	62500	Raw vectorized 250×250 images of celebrities faces from the "Labeled Faces in the Wild" dataset. `http://vis-www.cs.umass.edu/lfw/`
celeba	202599	38804	Raw vectorized 178×218 images of celebrities faces from CelebA dataset. `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`

Table D.1: Summary of the main datasets accessible via the FatDatasets package

D.3 The PrivateClustering package

This short packages contains a few methods for private clustering, including:

- an implementation of the DPLloyd algorithm [274], and its modified version with improved initialization [321];
- wrapper code to run the EUGkM algorithm [321] (using a modified version of the author's python implementation);
- wrapper code to run the privgene algorithm [289] (using the author's python implementation).

D.4 The PrivatePCA package

This short packages contains a few methods for private PCA, including:

- an implementation of the Laplace PCA baseline;
- an implementation of the Wishart PCA method [285];
- an implementation of the method of Upadhyay [294, p.23 of the supplementary material];
- an implementation of the method of Arora et al. [295].

D.5 The BatchIterators package

BatchIterators is a very small package, which provides a constructor `BatchIterator` to iterate over a (possibly out of core) dataset or data stream by batches. The package also provides the helper function `choose_batchsize` to select an appropriate batch size (i.e. so that the overall memory consumption does not exceed the total amount of available memory, or a user-chosen threshold).

D.6 The ExperimentsManager package

Many experiments in the thesis, and in particular experiments from Chapter 4, required to run the same algorithm over wide parameter spaces (e.g. performing compressive clustering with different kernel variance σ_κ^2, different sketch sizes and different types of structured operators). The ExperimentsManager package aims at providing a convenient way to run simple workflows across whole parameter spaces.

Any workflow provided to the package (e.g. running multiple clustering algorithms on a dataset, each with different sets of parameters) will be expanded into a collection of atomic workflows (i.e. running one algorithm with a specific set of parameters), which will by default be run locally, but can be dispatched (e.g. on a computing grid) thanks to an abstraction layer.

D.7 The Igrida package

Igrida is the name of a computing grid. The Igrida package provides a function to execute Julia code[5] asynchronously on this grid, as well as helper function to install/update software and data dependencies and fetch results.

[5] Represented in Julia via the `Expr` type, however metaprogramming is only marginally used and any (not only Julia) code could technically be executed with this package.

Bibliography

[1] Alastair R. Hall. *Generalized Method of Moments*. Oxford university press, 2005.

[2] Nicolas Keriven, Nicolas Tremblay, Yann Traonmilin, and Rémi Gribonval. "Compressive K-Means." In: International Conference on Acoustics, Speech and Signal Processing (ICASSP). Mar. 5, 2017. URL: https://hal.inria.fr/hal-01386077/document.

[3] Anthony Bourrier, Rémi Gribonval, and Patrick Pérez. "Compressive Gaussian Mixture Estimation." In: ICASSP-38th International Conference on Acoustics, Speech, and Signal Processing. 2013, pp. 6024–6028.

[4] Nicolas Keriven, Anthony Bourrier, Rémi Gribonval, and Patrick Pérez. "Sketching for Large-Scale Learning of Mixture Models." In: *Information and Inference: A Journal of the IMA* 7.3 (2017), pp. 447–508.

[5] Rémi Gribonval, Gilles Blanchard, Nicolas Keriven, and Yann Traonmilin. *Compressive Statistical Learning with Random Feature Moments*. Apr. 16, 2020. URL: https://hal.inria.fr/hal-01544609.

[6] Rémi Gribonval, Gilles Blanchard, Nicolas Keriven, and Yann Traonmilin. *Statistical Learning Guarantees for Compressive Clustering and Compressive Mixture Modeling*. Apr. 16, 2020. URL: https://hal.inria.fr/hal-02536818.

[7] L. F. Menabrea and Ada Lovelace. "Sketch of the Analytical Engine Invented by Charles Babbage." In: (1843), p. 63.

[8] Gordon E. Moore. "Cramming More Components onto Integrated Circuits." In: *Proceedings of the IEEE* 86.1 (1998), pp. 82–85.

[9] Robert H. Dennard, Fritz H. Gaensslen, V. Leo Rideout, Ernest Bassous, and Andre R. LeBlanc. "Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions." In: *IEEE Journal of Solid-State Circuits* 9.5 (1974), pp. 256–268.

[10] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. "NP-Hardness of Euclidean Sum-of-Squares Clustering." In: *Machine learning* 75.2 (2009), pp. 245–248.

[11] Stuart Lloyd. "Least Squares Quantization in PCM." In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[12] Anil K. Jain. "Data Clustering: 50 Years beyond K-Meansq." In: *Pattern Recognition Letters* 31 (2010), pp. 651–666.

[13] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[14] Stuart Lloyd. "Least squares quantization in PCM." In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[15] URL: https://web.archive.org/web/20200102030204/https://developers.google.com/machine-learning/data-prep/construct/collect/data-size-quality.

[16] URL: https://web.archive.org/web/20200411023503/https://home.cern/news/news/computing/cern-data-centre-passes-200-petabyte-milestone.

[17] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. "Recalibrating Global Data Center Energy-Use Estimates." In: *Science* 367.6481 (2020), pp. 984–986.

[18] Pierson and Jean-Marc. *Large-Scale Distributed Systems and Energy Efficiency : A Holistic View*. 1st ed. Wiley Series on Parallel and Distributed Computing. Wiley, 2015. ISBN: 978-1-118-86463-0.

[19] Arvind Narayanan and Vitaly Shmatikov. *How To Break Anonymity of the Netflix Prize Dataset*. 2006. arXiv: cs/0610105 [cs.CR].

[20] URL: https://web.archive.org/web/20130206171809/http://www.google.com/recaptcha/faq.

[21] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press, 2014.

[22] Ali Rahimi and Benjamin Recht. "Weighted Sums of Random Kitchen Sinks: Replacing Minimization with Randomization in Learning." In: *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Curran Associates, Inc., 2009, pp. 1313–1320. URL: http://papers.nips.cc/paper/3495-weighted-sums-of-random-kitchen-sinks-replacing-minimization-with-randomization-in-learning.pdf.

[23] Anna C. Gilbert, Yi Zhang, Kibok Lee, Yuting Zhang, and Honglak Lee. "Towards Understanding the Invertibility of Convolutional Neural Networks." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI'17. Melbourne, Australia: AAAI Press, Aug. 19, 2017, pp. 1703–1710. ISBN: 978-0-9992411-0-3.

[24] Antoine Chatalic, Rémi Gribonval, and Nicolas Keriven. "Large-Scale High-Dimensional Clustering with Fast Sketching." In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018. URL: https://hal.inria.fr/hal-01701121/document.

[25] Antoine Chatalic and Rémi Gribonval. "Learning to Sketch for Compressive Clustering." In: International Traveling Workshop on Interactions between Low-Complexity Data Models and Sensing Techniques (iTWIST). June 2020.

[26] Evan Byrne, Antoine Chatalic, Rémi Gribonval, and Philip Schniter. "Sketched Clustering via Hybrid Approximate Message Passing." In: *IEEE Transactions on Signal Processing* 67.17 (Sept. 2019), pp. 4556–4569. DOI: 10.1109/TSP.2019.2924585.

[27] Antoine Chatalic, Vincent Schellekens, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques, and Rémi Gribonval. "Compressive Learning with Privacy Guarantees." Submitted to Information and Inference (under review). Submitted to Information and Inference (under review). Mar. 3, 2020. URL: https://hal.inria.fr/hal-02496896.

[28] Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques, and Rémi Gribonval. "Compressive K-Means with Differential Privacy." In: SPARS Workshop. July 1, 2019. URL: https://hal.inria.fr/hal-02154820.

[29] Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Yves-Alexandre De Montjoye, Laurent Jacques, and Rémi Gribonval. "Differentially Private Compressive K-Means." In: *44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Brighton, United Kingdom, May 2019, pp. 1–5. URL: https://hal.inria.fr/hal-02060208.

[30] Antoine Chatalic, Nicolas Keriven, and Rémi Gribonval. "Projections aléatoires pour l'apprentissage compressif." In: Gretsi. Aug. 26, 2019. URL: https://hal.inria.fr/hal-02154803.

[31] Rémi Gribonval, Antoine Chatalic, Nicolas Keriven, Vincent Schellekens, Laurent Jacques, and Philip Schniter. "Sketching Datasets for Large-Scale Learning (Long Version)." Aug. 4, 2020. URL: https://hal.inria.fr/hal-02909766.

[32] Claude Elwood Shannon. "Communication in the Presence of Noise." In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[33] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Vol. 1. 3. Birkhäuser Basel, 2013. URL: http://www.ams.org/bull/2017-54-01/S0273-0979-2016-01546-1/.

[34] Holger Boche, Robert Calderbank, Gitta Kutyniok, and Jan Vybíral. "A Survey of Compressed Sensing." In: *Compressed Sensing and Its Applications*. Ed. by Holger Boche, Robert Calderbank, Gitta Kutyniok, and Jan Vybíral. Applied and Numerical Harmonic Analysis. Cham: Springer International Publishing, 2015, pp. 1–39. ISBN: 978-3-319-16041-2 978-3-319-16042-9. DOI: 10.1007/978-3-319-16042-9_1. URL: http://link.springer.com/10.1007/978-3-319-16042-9_1.

[35] Emmanuel J. Candès, Justin Romberg, and Terence Tao. "Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information." In: *IEEE Transactions on information theory* 52.2 (2006), pp. 489–509.

[36] David L. Donoho. "Compressed Sensing." In: *IEEE Transactions on information theory* 52.4 (2006), pp. 1289–1306.

[37] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[38] S.G. Mallat and Zhifeng Zhang. "Matching Pursuits with Time-Frequency Dictionaries." In: *IEEE Transactions on Signal Processing* 41.12 (Dec./1993), pp. 3397–3415. ISSN: 1053587X. DOI: 10.1109/78.258082. URL: http://ieeexplore.ieee.org/document/258082/.

[39] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition." In: *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*. IEEE. 1993, pp. 40–44.

[40] Joel A. Tropp and Anna C. Gilbert. "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit." In: *IEEE Transactions on Information Theory* 53.12 (2007), p. 4655.

[41] Prateek Jain, Ambuj Tewari, and Inderjit S. Dhillon. "Orthogonal Matching Pursuit with Replacement." In: *Advances in Neural Information Processing Systems*. 2011, pp. 1215–1223.

[42] Deanna Needell and Joel A. Tropp. "CoSaMP: Iterative Signal Recovery from Incomplete and Inaccurate Samples." In: *Applied and computational harmonic analysis* 26.3 (2009), pp. 301–321.

[43] Thomas Blumensath and Mike E. Davies. "Iterative Thresholding for Sparse Approximations." In: *Journal of Fourier Analysis and Applications* 14.5-6 (Dec. 2008), pp. 629–654. ISSN: 1069-5869, 1531-5851. DOI: 10.1007/s00041-008-9035-z. URL: http://link.springer.com/10.1007/s00041-008-9035-z.

[44] Emmanuel J. Candes and Terence Tao. "Near-Optimal Signal Recovery from Random Projections: Universal Encoding Strategies?" In: *IEEE transactions on information theory* 52.12 (2006), pp. 5406–5425.

[45] Emmanuel J. Candes and Terence Tao. "Decoding by Linear Programming." In: *IEEE transactions on information theory* 51.12 (2005), pp. 4203–4215.

[46] Emmanuel J. Candès, Justin Romberg, and Terence Tao. "Stable Signal Recovery from Incomplete and Inaccurate Measurements." In: *Communications on Pure and Applied Mathematics* 59 (2006), pp. 1207–1223.

[47] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. "A Simple Proof of the Restricted Isometry Property for Random Matrices." In: *Constructive Approximation* 28.3 (Dec. 2008), pp. 253–263. ISSN: 0176-4276, 1432-0940. DOI: 10.1007/s00365-007-9003-x. URL: http://link.springer.com/10.1007/s00365-007-9003-x.

[48] Emmanuel J. Candès and Benjamin Recht. "Exact Matrix Completion via Convex Optimization." In: *Foundations of Computational Mathematics* 9.6 (Dec. 2009), pp. 717–772. ISSN: 1615-3375, 1615-3383. DOI: 10.1007/s10208-009-9045-5. URL: http://link.springer.com/10.1007/s10208-009-9045-5.

[49] Emmanuel J. Candes, Yonina C. Eldar, Thomas Strohmer, and Vladislav Voroninski. "Phase Retrieval via Matrix Completion." In: *SIAM review* 57.2 (2015), pp. 225–251.

[50] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization." In: *SIAM Review* 52.3 (Jan. 2010), pp. 471–501. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/070697835. URL: http://epubs.siam.org/doi/10.1137/070697835.

[51] Kiryung Lee and Yoram Bresler. "ADMiRA: Atomic Decomposition for Minimum Rank Approximation." In: *IEEE Transactions on Information Theory* 56.9 (Sept. 2010), pp. 4402–4416. ISSN: 1557-9654. DOI: 10.1109/TIT.2010.2054251.

[52] Prateek Jain, Raghu Meka, and Inderjit S. Dhillon. "Guaranteed Rank Minimization via Singular Value Projection." In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 937–945. URL: http://papers.nips.cc/paper/3904-guaranteed-rank-minimization-via-singular-value-projection.pdf.

[53] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. "A Singular Value Thresholding Algorithm for Matrix Completion." In: *SIAM Journal on Optimization* 20.4 (Jan. 2010), pp. 1956–1982. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/080738970. URL: http://epubs.siam.org/doi/10.1137/080738970.

[54] Richard G. Baraniuk, Volkan Cevher, Marco F. Duarte, and Chinmay Hegde. "Model-Based Compressive Sensing." In: *IEEE Transactions on Information Theory* 56.4 (Apr. 2010), pp. 1982–2001. ISSN: 0018-9448, 1557-9654. DOI: 10.1109/TIT.2010.2040894. arXiv: 0808.3572. URL: http://arxiv.org/abs/0808.3572.

[55] Sangnam Nam, Mike E. Davies, Michael Elad, and Rémi Gribonval. "The Cosparse Analysis Model and Algorithms." In: *Applied and Computational Harmonic Analysis* 34.1 (2013), pp. 30–56.

[56] Yue M. Lu and Minh N. Do. "A Theory for Sampling Signals from a Union of Subspaces." In: *IEEE transactions on signal processing* 56.6 (2008), pp. 2334–2345.

[57] Richard G. Baraniuk and Michael B. Wakin. "Random Projections of Smooth Manifolds." In: *Foundations of computational mathematics* 9.1 (2009), pp. 51–77.

[58] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. "Compressed Sensing Using Generative Models." In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 537–546.

[59] Yan Wu, Mihaela Rosca, and Timothy Lillicrap. *Deep Compressed Sensing*. May 16, 2019. arXiv: 1905.06723 [cs, eess, stat]. URL: http://arxiv.org/abs/1905.06723.

[60] A. Bourrier, M. E. Davies, T. Peleg, P. Pérez, and R. Gribonval. "Fundamental Performance Limits for Ideal Decoders in High-Dimensional Linear Inverse Problems." In: *IEEE Transactions on Information Theory* 60.12 (Dec. 2014), pp. 7928–7946. ISSN: 0018-9448. DOI: 10.1109/TIT.2014.2364403.

[61] Nicolas Keriven and Rémi Gribonval. "Instance Optimal Decoding and the Restricted Isometry Property." In: *Journal of Physics: Conference Series*. Vol. 1131. IOP Publishing, 2018, p. 012002.

[62] Michael P. Sheehan, Antoine Gonon, and Mike E. Davies. *Compressive Learning for Semi-Parametric Models*. Oct. 22, 2019. arXiv: 1910.10024 [cs, stat]. URL: http://arxiv.org/abs/1910.10024.

[63] Michael Sheehan, Madeleine Kotzagiannidis, and Mike Davies. "Compressive Independent Component Analysis." In: 27th European Signal Processing Conference (EUSIPCO). 2019.

[64] G. Puy, M. E. Davies, and R. Gribonval. "Recipes for Stable Linear Embeddings From Hilbert Spaces to Rm." In: *IEEE Transactions on Information Theory* 63.4 (Apr. 2017), pp. 2171–2187. ISSN: 0018-9448. DOI: 10.1109/TIT.2017.2664858.

[65] Sjoerd Dirksen. "Dimensionality Reduction with Subgaussian Matrices: A Unified Theory." In: *Foundations of Computational Mathematics* 16.5 (Oct. 1, 2016), pp. 1367–1396. ISSN: 1615-3383. DOI: 10.1007/s10208-015-9280-x. URL: https://doi.org/10.1007/s10208-015-9280-x.

[66] Nachman Aronszajn. "Theory of Reproducing Kernels." In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.

[67] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. 1st. Adaptive Computation and Machine Learning. The MIT Press, 2001. ISBN: 978-0-262-19475-4.

[68] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers." In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. 1992, pp. 144–152.

[69] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Kernel Principal Component Analysis." In: *Artificial Neural Networks — ICANN'97*. Ed. by Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1997, pp. 583–588. ISBN: 978-3-540-69620-9. DOI: 10.1007/BFb0020217.

[70] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem." In: *Neural computation* 10.5 (1998), pp. 1299–1319.

[71] Mark Girolami. "Mercer Kernel-Based Clustering in Feature Space." In: *IEEE Transactions on Neural Networks* 13.3 (2002), pp. 780–784.

[72] Ali Rahimi and Benjamin Recht. "Random Features for Large-Scale Kernel Machines." In: *Advances in Neural Information Processing Systems*. 2008, pp. 1177–1184. URL: http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

[73] S Bochner. "Monotone Funktionen, Stieltjessche Integrale und harmonische Analyse." In: *Collected Papers of Salomon Bochner* 2 (1992), p. 87.

[74] Bharath Sriperumbudur and Zoltán Szabó. "Optimal Rates for Random Fourier Features." In: *Advances in Neural Information Processing Systems*. 2015, pp. 1144–1152.

[75] Francis Bach. "On the Equivalence between Kernel Quadrature Rules and Random Feature Expansions." In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 714–751.

[76] Dougal J. Sutherland and Jeff Schneider. "On the Error of Random Fourier Features." In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. UAI'15. Arlington, Virginia, USA: AUAI Press, July 12, 2015, pp. 862–871. ISBN: 978-0-9966431-0-8.

[77] Quoc Le, Tamás Sarlós, and Alex Smola. "Fastfood: Approximating Kernel Expansions in Loglinear Time." In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, 2013, pp. III-244-III–252. URL: http://www.jmlr.org/proceedings/papers/v28/le13-supp.pdf.

[78] Krzysztof Choromanski, Mark Rowland, Tamas Sarlos, Vikas Sindhwani, Richard Turner, and Adrian Weller. "The Geometry of Random Features." In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. Playa Blanca, Lanzarote, Canary Islands: PMLR, Apr. 9–11, 2018, pp. 1–9. URL: http://proceedings.mlr.press/v84/choromanski18a.html.

[79] Ruben Ohana, Jonas Wacker, Jonathan Dong, Sébastien Marmin, Florent Krzakala, Maurizio Filippone, and Laurent Daudet. *Kernel Computations from Large-Scale Random Features Obtained by Optical Processing Units*. Dec. 2, 2019. arXiv: 1910.09880 [cs]. URL: http://arxiv.org/abs/1910.09880.

[80] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. *Random Fourier Features for Kernel Ridge Regression: Approximation Bounds and Statistical Guarantees*. Apr. 26, 2018. arXiv: 1804.09893 [cs, math, stat]. URL: http://arxiv.org/abs/1804.09893.

[81] Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W. Mahoney. "Quasi-Monte Carlo Feature Maps for Shift-Invariant Kernels." In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 4096–4133.

[82] Yanjun Li, Kai Zhang, Jun Wang, and Sanjiv Kumar. "Learning Adaptive Random Features." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 4229–4236.

[83] Zichao Yang, Andrew Wilson, Alex Smola, and Le Song. "A La Carte–Learning Fast Kernels." In: *Artificial Intelligence and Statistics*. 2015, pp. 1098–1106.

[84] Aman Sinha and John C. Duchi. "Learning Kernels with Random Features." In: *Advances in Neural Information Processing Systems*. 2016, pp. 1298–1306.

[85] Kenji Kawaguchi, Bo Xie, and Le Song. "Deep Semi-Random Features for Nonlinear Function Approximation." In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[86] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. "Kernel Mean Embedding of Distributions: A Review and Beyond." In: *Foundations and Trends® in Machine Learning* 10.1-2 (2017), pp. 1–141.

[87] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J. Smola. "A Kernel Method for the Two-Sample-Problem." In: *Advances in Neural Information Processing Systems*. 2007, pp. 513–520.

[88] Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet. "Hilbert Space Embeddings and Metrics on Probability Measures." In: *Journal of Machine Learning Research* 11 (Apr 2010), pp. 1517–1561. ISSN: ISSN 1533-7928. URL: http://www.jmlr.org/papers/v11/sriperumbudur10a.html.

[89] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. "A Hilbert Space Embedding for Distributions." In: *Algorithmic Learning Theory*. Ed. by Marcus Hutter, Rocco A. Servedio, and Eiji Takimoto. Vol. 4754. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 13–31. ISBN: 978-3-540-75224-0 978-3-540-75225-7. DOI: 10.1007/978-3-540-75225-7_5. URL: http://link.springer.com/10.1007/978-3-540-75225-7_5.

[90] Le Song, Xinhua Zhang, Alex Smola, Arthur Gretton, and Bernhard Schölkopf. "Tailoring Density Estimation via Reproducing Kernel Moment Matching." In: *Proceedings of the 25th International Conference on Machine Learning - ICML '08*. The 25th International Conference. Helsinki, Finland: ACM Press, 2008, pp. 992–999. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390281. URL: http://portal.acm.org/citation.cfm?doid=1390156.1390281.

[91] Stefanie Jegelka, Arthur Gretton, Bernhard Schölkopf, Bharath K. Sriperumbudur, and Ulrike von Luxburg. "Generalized Clustering via Kernel Embeddings." In: *KI 2009: Advances in Artificial Intelligence*. Ed. by Bärbel Mertsching, Marcus Hund, and Zaheer Aziz. Vol. 5803. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 144–152. ISBN: 978-3-642-04616-2 978-3-642-04617-9. DOI: 10.1007/978-3-642-04617-9_19. URL: http://link.springer.com/10.1007/978-3-642-04617-9_19.

[92] Badr-Eddine Chérief-Abdellatif and Pierre Alquier. *MMD-Bayes: Robust Bayesian Estimation via Maximum Mean Discrepancy*. Dec. 11, 2019. arXiv: 1909.13339 [cs, math, stat]. URL: http://arxiv.org/abs/1909.13339.

[93] Anthony Bourrier, Rémi Gribonval, and Patrick Pérez. "Compressive Gaussian Mixture Estimation." In: *Compressed Sensing and Its Applications*. Ed. by Holger Boche, Robert Calderbank, Gitta Kutyniok, and Jan Vybíral. Applied and Numerical Harmonic Analysis. Springer International Publishing, 2015, pp. 239–258. ISBN: 978-3-319-16041-2 978-3-319-16042-9. DOI: 10.1007/978-3-319-16042-9_8. URL: http://link.springer.com/chapter/10.1007/978-3-319-16042-9_8.

[94] N. Keriven, A. Bourrier, R. Gribonval, and P. Pérez. "Sketching for Large-Scale Learning of Mixture Models." In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2016, pp. 6190–6194. DOI: 10.1109/ICASSP.2016.7472867.

[95] Clément Elvira, Rémi Gribonval, Charles Soussen, and Cedric Herzet. "Omp and Continuous Dictionaries: Is k-Step Recovery Possible?" In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5546–5550.

[96] Clément Elvira, Rémi Gribonval, Charles Soussen, and Cédric Herzet. *When Does OMP Achieve Support Recovery with Continuous Dictionaries?* Apr. 17, 2019. arXiv: 1904.06311 [physics]. URL: http://arxiv.org/abs/1904.06311.

[97] Yohann de Castro and Fabrice Gamboa. "Exact Reconstruction Using Beurling Minimal Extrapolation." In: *J. Math. Anal. Appl* 395 (2012), pp. 336–354.

[98] Kristian Bredies and Hanna Katriina Pikkarainen. "Inverse Problems in Spaces of Measures." In: *ESAIM: Control, Optimisation and Calculus of Variations* 19.1 (Jan. 2013), pp. 190–218. ISSN: 1292-8119, 1262-3377. DOI: 10.1051/cocv/2011205. URL: http://www.esaim-cocv.org/10.1051/cocv/2011205.

[99] Gongguo Tang, Badri Narayan Bhaskar, Parikshit Shah, and Benjamin Recht. "Compressed Sensing off the Grid." In: *IEEE transactions on information theory* 59.11 (2013), pp. 7465–7490.

[100] Quentin Denoyelle, Vincent Duval, Gabriel Peyré, and Emmanuel Soubies. "The Sliding Frank–Wolfe Algorithm and Its Application to Super-Resolution Microscopy." In: *Inverse Problems* 36.1 (2019), p. 014001.

[101] Anthony Bourrier. "Échantillonnage Compressé et Réduction de Dimension Pour l'apprentissage Non Supervisé." PhD thesis. Université Rennes 1, May 13, 2014. URL: https://tel.archives-ouvertes.fr/tel-01023030/document.

[102] Nicolas Keriven, Antoine Deleforge, and Antoine Liutkus. "Blind Source Separation Using Mixtures of Alpha-Stable Distributions." In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 771–775.

[103] Gongguo Tang, Badri Narayan Bhaskar, Parikshit Shah, and Benjamin Recht. "Compressive Sensing off the Grid." In: *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2012.

[104] V. Schellekens and L. Jacques. "Quantized Compressive K-Means." In: *IEEE Signal Processing Letters* 25.8 (Aug. 2018), pp. 1211–1215. ISSN: 1070-9908. DOI: 10.1109/LSP.2018.2847908.

[105] Michael P. Sheehan, Antoine Gonon, and Mike E. Davies. "Compressive Learning for Semi-Parametric Models." Oct. 22, 2019. URL: http://arxiv.org/abs/1910.10024.

[106] Rene Vidal, Yi Ma, and Shankar Sastry. "Generalized Principal Component Analysis (GPCA)." In: *IEEE transactions on pattern analysis and machine intelligence* 27.12 (2005), pp. 1945–1959.

[107] Nicolas Keriven. "Sketching for Large-Scale Learning of Mixture Models." PhD thesis. Université Rennes 1, Oct. 12, 2017. URL: https://tel.archives-ouvertes.fr/tel-01620815/document.

[108] Vincent Schellekens and Laurent Jacques. "Compressive Classification." In: ITWIST. Marseille, 2018.

[109] Zhi-Wei Pan, Dao-Hong Xiang, Quan-Wu Xiao, and Ding-Xuan Zhou. "Parzen Windows for Multi-Class Classification." In: *Journal of Complexity* 24.5 (Oct. 1, 2008), pp. 606–618. ISSN: 0885-064X. DOI: 10.1016/j.jco.2008.07.001. URL: http://www.sciencedirect.com/science/article/pii/S0885064X08000526.

[110] Antoine Deleforge, Florence Forbes, and Radu Horaud. "High-Dimensional Regression with Gaussian Mixtures and Partially-Latent Response Variables." In: *Statistics and Computing* 25.5 (Sept. 1, 2015), pp. 893–911. ISSN: 0960-3174, 1573-1375. DOI: 10.1007/s11222-014-9461-5. URL: https://link.springer.com/article/10.1007/s11222-014-9461-5.

[111] Antoine Chatalic. *Towards Scalable Sketched Learning*. Master 2 internship report. Rennes: Université de Rennes 1, June 2017.

[112] Peter Kairouz et al. *Advances and Open Problems in Federated Learning*. Dec. 10, 2019. arXiv: 1912.04977 [cs, stat]. URL: http://arxiv.org/abs/1912.04977.

[113] Konstantinos Slavakis, Seung-Jun Kim, Gonzalo Mateos, and Georgios B. Giannakis. "Stochastic Approximation Vis-a-Vis Online Learning for Big Data Analytics [Lecture Notes]." In: *IEEE Signal Processing Magazine* 31.6 (Nov. 2014), pp. 124–129. ISSN: 1053-5888. DOI: 10.1109/MSP.2014.2345536. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6923526.

[114] Dongkuan Xu and Yingjie Tian. "A Comprehensive Survey of Clustering Algorithms." In: *Annals of Data Science* 2.2 (June 1, 2015), pp. 165–193. ISSN: 2198-5812. DOI: 10.1007/s40745-015-0040-1. URL: https://doi.org/10.1007/s40745-015-0040-1.

[115] Burton H. Bloom. "Space/Time Trade-Offs in Hash Coding with Allowable Errors." In: *Communications of the ACM* 13.7 (1970), pp. 422–426.

[116] Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine. "Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches." In: *Databases* 4.1-3 (2011), pp. 1–294.

[117] Philippe Flajolet and G. Nigel Martin. "Probabilistic Counting Algorithms for Database Applications." In: *Journal of Computer and System Sciences* 31 (1985), pp. 182–209.

[118] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. "Hyperloglog: The Analysis of a near-Optimal Cardinality Estimation Algorithm." In: 2007.

[119] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. "An Optimal Algorithm for the Distinct Elements Problem." In: *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2010, pp. 41–52.

[120] Noga Alon, Yossi Matias, and Mario Szegedy. "The Space Complexity of Approximating the Frequency Moments." In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. 1996, pp. 20–29.

[121] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. "Tracking Join and Self-Join Sizes in Limited Storage." In: *Journal of Computer and System Sciences* 64.3 (2002), pp. 719–747.

[122] Graham Cormode and Shan Muthukrishnan. "An Improved Data Stream Summary: The Count-Min Sketch and Its Applications." In: *Journal of Algorithms* 55.1 (2005), pp. 58–75.

[123] Graham Cormode. "Data Sketching." In: *Communications of the ACM* 60.9 (2017), pp. 48–55.

[124] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data Using T-SNE." In: *Journal of machine learning research* 9 (Nov 2008), pp. 2579–2605.

[125] William B. Johnson and Joram Lindenstrauss. "Extensions of Lipschitz Mappings into a Hilbert Space." In: *Contemporary mathematics* 26.189-206 (1984), p. 1.

[126] Sanjoy Dasgupta and Anupam Gupta. "An Elementary Proof of a Theorem of Johnson and Lindenstrauss." In: *Random Structures & Algorithms* 22.1 (2003), pp. 60–65. ISSN: 1098-2418. DOI: 10.1002/rsa.10073. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.10073.

[127] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Vol. 47. Cambridge University Press, 2018.

[128] Dimitris Achlioptas. "Database-Friendly Random Projections." In: *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2001, pp. 274–281.

[129] Jiří Matoušek. "On Variants of the Johnson–Lindenstrauss Lemma." In: *Random Structures & Algorithms* 33.2 (2008), pp. 142–156.

[130] Nir Ailon and Bernard Chazelle. "The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors." In: *SIAM Journal on Computing* 39.1 (2009), pp. 302–322. URL: http://epubs.siam.org/doi/abs/10.1137/060673096.

[131] Kasper Green Larsen and Jelani Nelson. "The Johnson-Lindenstrauss Lemma Is Optimal for Linear Dimensionality Reduction." In: *Proceedings of the 43rd International Colloquium on Authomata, Languages and Programming (ICALP)*. 2016.

[132] Kasper Green Larsen and Jelani Nelson. "Optimality of the Johnson-Lindenstrauss Lemma." In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2017, pp. 633–638.

[133] Robert Calderbank, Sina Jafarpour, and Robert Schapire. *Compressed Learning: Universal Sparse Dimensionality Reduction and Learning in the Measurement Domain*. 2009, p. 10.

[134] Xingyu Xu, Gen Li, and Yuantao Gu. *Johnson-Lindenstrauss Property Implies Subspace Restricted Isometry Property*. Sept. 28, 2019. arXiv: 1905.09608 [cs, math]. URL: http://arxiv.org/abs/1905.09608.

[135] Yuchen Jiao, Gen Li, and Yuantao Gu. *Compressed Subspace Learning Based on Canonical Angle Preserving Property*. July 24, 2019. arXiv: 1907.06166 [cs, math, stat]. URL: http://arxiv.org/abs/1907.06166.

[136] Felix Krahmer and Rachel Ward. "New and Improved Johnson–Lindenstrauss Embeddings via the Restricted Isometry Property." In: *SIAM Journal on Mathematical Analysis* 43.3 (2011), pp. 1269–1281.

[137] Maxim Raginsky and Svetlana Lazebnik. "Locality-Sensitive Binary Codes from Shift-Invariant Kernels." In: *Advances in Neural Information Processing Systems*. 2009, pp. 1509–1517.

[138] Piotr Indyk and Rajeev Motwani. "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality." In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. 1998, pp. 604–613.

[139] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. "A Survey of Product Quantization." In: *ITE Transactions on Media Technology and Applications* 6.1 (2018), pp. 2–10.

[140] Karl Pearson. "On Lines and Planes of Closest Fit to Systems of Points in Space." In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[141] Michael E. Tipping and Christopher M. Bishop. "Probabilistic Principal Component Analysis." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622.

[142] Jacqueline S. Galpin and Douglas M. Hawkins. "Methods of L1 Estimation of a Covariance Matrix." In: *Computational Statistics & Data Analysis* 5.4 (1987), pp. 305–319.

[143] Hui Zou, Trevor Hastie, and Robert Tibshirani. "Sparse Principal Component Analysis." In: *Journal of computational and graphical statistics* 15.2 (2006), pp. 265–286.

[144] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Kernel Principal Component Analysis." In: *International Conference on Artificial Neural Networks*. Springer, 1997, pp. 583–588.

[145] Harold Hotelling. "Relations between Two Sets of Variates." In: *Breakthroughs in Statistics*. Springer, 1992, pp. 162–190.

[146] Aapo Hyvärinen and Erkki Oja. "Independent Component Analysis: Algorithms and Applications." In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[147] John P. Cunningham and Zoubin Ghahramani. "Linear Dimensionality Reduction: Survey, Insights, and Generalizations." In: *Journal of Machine Learning Research* 16 (2015), pp. 2859–2900.

[148] Girish Chandrashekar and Ferat Sahin. "A Survey on Feature Selection Methods." In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[149] Jingdong Wang, Ting Zhang, Nicu Sebe, and Heng Tao Shen. "A Survey on Learning to Hash." In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 769–790.

[150] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks." In: *science* 313.5786 (2006), pp. 504–507.

[151] Dan Feldman. "Core-Sets: An Updated Survey." In: *WIREs Data Mining and Knowledge Discovery* 10.1 (2020), e1335. ISSN: 1942-4795. DOI: 10.1002/widm.1335. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1335.

[152] Jeff M Phillips. "Coresets and Sketches." In: *Handbook of Discrete and Computational Geometry*. Ed. by Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth. 3rd edition. CRC press LCC, 2017, p. 20.

[153] Olivier Bachem, Mario Lucic, and Andreas Krause. *Practical Coreset Constructions for Machine Learning*. 2017. arXiv: 1703.06476.

[154] Alexander Munteanu and Chris Schwiegelshohn. "Coresets-Methods and History: A Theoreticians Design Pattern for Approximation and Streaming Algorithms." In: *KI - Künstliche Intelligenz* 32.1 (Feb. 1, 2018), pp. 37–53. ISSN: 1610-1987. DOI: 10.1007/s13218-017-0519-3. URL: https://doi.org/10.1007/s13218-017-0519-3.

[155] Ibrahim Jubran, Alaa Maalouf, and Dan Feldman. *Introduction to Coresets: Accurate Coresets*. Version 1. Oct. 19, 2019. arXiv: 1910.08707 [cs, stat]. URL: http://arxiv.org/abs/1910.08707.

[156] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. "Composable Coresets for Diversity and Coverage Maximization." In: *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2014, pp. 100–108.

[157] Garvesh Raskutti and Michael W. Mahoney. "A Statistical Perspective on Randomized Sketching for Ordinary Least-Squares." In: *The Journal of Machine Learning Research* 17.1 (Jan. 1, 2016), pp. 7508–7538. ISSN: 1532-4435.

[158] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. "Geometric Approximation via Coresets." In: *Combinatorial and computational geometry* 52 (2005), pp. 1–30.

[159] Kenneth L. Clarkson. "Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm." In: *ACM Transactions on Algorithms (TALG)* 6.4 (2010), pp. 1–30.

[160] Yutian Chen, Max Welling, and Alex Smola. "Super-Samples from Kernel Herding." In: *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. UAI'10. Arlington, Virginia, USA: AUAI Press, July 8, 2010, pp. 109–116. ISBN: 978-0-9749039-6-5.

[161] Francis Bach, Simon Lacoste-Julien, and Guillaume Obozinski. "On the Equivalence between Herding and Conditional Gradient Algorithms." In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Madison, WI, USA: Omnipress, June 26, 2012, pp. 1355–1362. ISBN: 978-1-4503-1285-1.

[162] Jeff M. Phillips and Wai Ming Tai. "Improved Coresets for Kernel Density Estimates." In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2018, pp. 2718–2727.

[163] Dan Feldman, Melanie Schmidt, and Christian Sohler. "Turning Big Data into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering." In: *SIAM Journal on Computing* 49.3 (2020), pp. 601–657.

[164] Jan-Philipp W. Kappmeier, Daniel R. Schmidt, and Melanie Schmidt. "Solving K-Means on High-Dimensional Big Data." In: *International Symposium on Experimental Algorithms*. Springer, 2015, pp. 259–270.

[165] Artem Barger and Dan Feldman. "K-Means for Streaming and Distributed Big Sparse Data." In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 342–350.

[166] Dan Feldman, Mikhail Volkov, and Daniela Rus. "Dimensionality Reduction of Massive Sparse Datasets Using Coresets." In: *Advances in Neural Information Processing Systems*. 2016, pp. 2766–2774.

[167] Sariel Har-Peled. "No, Coreset, No Cry." In: *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 2004, pp. 324–335.

[168] Alejandro Molina, Alexander Munteanu, and Kristian Kersting. "Core Dependency Networks." In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[169] Ke Chen. "On Coresets for K-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications." In: *SIAM Journal on Computing* 39.3 (2009), pp. 923–947.

[170] Michael Langberg and Leonard J. Schulman. "Universal ε-Approximators for Integrals." In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2010, pp. 598–607.

[171] Yi Li, Philip M. Long, and Aravind Srinivasan. "Improved Bounds on the Sample Complexity of Learning." In: *Journal of Computer and System Sciences* 62.3 (2001), pp. 516–527.

[172] M. Vidyasagar. *Learning and Generalisation: With Applications to Neural Networks*. 2nd ed. Communications and Control Engineering. Springer-Verlag London, 2003. ISBN: 978-1-84996-867-6.

[173] Dan Feldman and Michael Langberg. "A Unified Framework for Approximating and Clustering Data." In: *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. 2011, pp. 569–578.

[174] Dan Feldman, Micha Feigin, and Nir Sochen. "Learning Big (Image) Data via Coresets for Dictionaries." In: *Journal of mathematical imaging and vision* 46.3 (2013), pp. 276–291.

[175] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. "Training Gaussian Mixture Models at Scale via Coresets." In: *The Journal of Machine Learning Research* 18.1 (Jan. 1, 2017), pp. 5885–5909. ISSN: 1532-4435.

[176] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. "Sampling Algorithms and Coresets for L_p Regression." In: *SIAM Journal on Computing* 38.5 (2009), pp. 2060–2078.

[177] Nicolas Tremblay, Simon Barthelmé, and Pierre-Olivier Amblard. "Determinantal Point Processes for Coresets." In: *Journal of Machine Learning Research* 20.168 (2019), pp. 1–70.

[178] David C. Hoaglin and Roy E. Welsch. "The Hat Matrix in Regression and ANOVA." In: *The American Statistician* 32.1 (1978), pp. 17–22.

[179] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. "Fast Approximation of Matrix Coherence and Statistical Leverage." In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 3475–3506.

[180] Tamas Sarlos. "Improved Approximation Algorithms for Large Matrices via Random Projections." In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06). Oct. 2006, pp. 143–152. DOI: 10.1109/FOCS.2006.37.

[181] Michael W. Mahoney. "Randomized Algorithms for Matrices and Data." In: *Foundations and Trends® in Machine Learning* 3.2 (Feb. 1, 2011), pp. 123–224. ISSN: 1935-8237. DOI: 10.1561/2200000035. URL: https://doi.org/10.1561/2200000035.

[182] Per-Gunnar Martinsson and Joel Tropp. *Randomized Numerical Linear Algebra: Foundations & Algorithms*. Feb. 4, 2020. arXiv: 2002.01387 [cs, math]. URL: http://arxiv.org/abs/2002.01387.

[183] Ravindran Kannan and Santosh Vempala. "Randomized Algorithms in Numerical Linear Algebra." In: *Acta Numerica* 26 (2017), p. 95.

[184] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions." In: *SIAM review* 53.2 (2011), pp. 217–288.

[185] Carl Eckart and Gale Young. "The Approximation of One Matrix by Another of Lower Rank." In: *Psychometrika* 1.3 (1936), pp. 211–218.

[186] Joel A. Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. "Practical Sketching Algorithms for Low-Rank Matrix Approximation." In: *SIAM Journal on Matrix Analysis and Applications* 38.4 (2017), pp. 1454–1485.

[187] Christos Boutsidis, David P. Woodruff, and Peilin Zhong. "Optimal Principal Component Analysis in Distributed and Streaming Models." In: *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. 2016, pp. 236–249.

[188] Kenneth L. Clarkson and David P. Woodruff. "Numerical Linear Algebra in the Streaming Model." In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. 2009, pp. 205–214.

[189] Michael W. Mahoney and Petros Drineas. "CUR Matrix Decompositions for Improved Data Analysis." In: *Proceedings of the National Academy of Sciences* 106.3 (2009), pp. 697–702.

[190] Per-Gunnar Martinsson. "Randomized Methods for Matrix Computations." In: *The Mathematics of Data* 25 (2019), pp. 187–231.

[191] Alex A. Gittens. "Topics in Randomized Numerical Linear Algebra." PhD thesis. California Institute of Technology, 2013. URL: http://resolver.caltech.edu/CaltechTHESIS:06102013-100609092.

[192] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. "Fixed-Rank Approximation of a Positive-Semidefinite Matrix from Streaming Data." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 1225–1234. URL: http://papers.nips.cc/paper/6722-fixed-rank-approximation-of-a-positive-semidefinite-matrix-from-streaming-data.pdf.

[193] Yun Yang, Mert Pilanci, and Martin J. Wainwright. "Randomized Sketches for Kernels: Fast and Optimal Nonparametric Regression." In: *The Annals of Statistics* 45.3 (2017), pp. 991–1023. URL: http://arxiv.org/abs/1501.06195.

[194] Samory Kpotufe and Bharath K. Sriperumbudur. *Kernel Sketching Yields Kernel JL*. 2019. arXiv: 1908.05818.

[195] Kenneth L. Clarkson and David P. Woodruff. "Low-Rank PSD Approximation in Input-Sparsity Time." In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2017, pp. 2061–2072.

[196] E. J. Nyström. "Über Die Praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben." In: *Acta Mathematica* 54 (1930), pp. 185–204. ISSN: 0001-5962, 1871-2509. DOI: 10.1007/BF02547521. URL: https://projecteuclid.org/euclid.acta/1485887849.

[197] Christopher KI Williams and Matthias Seeger. "Using the Nyström Method to Speed up Kernel Machines." In: *Advances in Neural Information Processing Systems*. 2001, pp. 682–688.

[198] Shusen Wang, Alex Gittens, and Michael W. Mahoney. "Scalable Kernel K-Means Clustering with Nyström Approximation: Relative-Error Bounds." In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 431–479.

[199] Daniele Calandriello and Lorenzo Rosasco. *Statistical and Computational Trade-Offs in Kernel K-Means*. Aug. 27, 2019. arXiv: 1908.10284 [cs, stat]. URL: http://arxiv.org/abs/1908.10284.

[200] Francis Bach. "Sharp Analysis of Low-Rank Kernel Matrix Approximations." In: *Conference on Learning Theory*. 2013, pp. 185–209.

[201] Kaare Brandt Petersen and Michael Syskind Pedersen. "The Matrix Cookbook." In: (2012).

[202] Ahmed Alaoui and Michael W. Mahoney. "Fast Randomized Kernel Ridge Regression with Statistical Guarantees." In: *Advances in Neural Information Processing Systems*. 2015, pp. 775–783.

[203] Cameron Musco and Christopher Musco. "Recursive Sampling for the Nystrom Method." In: *Advances in Neural Information Processing Systems*. 2017, pp. 3833–3845.

[204] Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco. "On Fast Leverage Score Sampling and Optimal Learning." In: *Advances in Neural Information Processing Systems*. 2018, pp. 5672–5682.

[205] Mu Li, Wei Bi, James T. Kwok, and Bao-Liang Lu. "Large-Scale Nyström Kernel Matrix Approximation Using Randomized SVD." In: *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS* 26.1 (2015).

[206] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. "Falkon: An Optimal Large Scale Kernel Method." In: *Advances in Neural Information Processing Systems*. 2017, pp. 3888–3898.

[207] Shahin Shahrampour and Soheil Kolouri. *On Sampling Random Features From Empirical Leverage Scores: Implementation and Theoretical Guarantees*. Mar. 19, 2019. arXiv: 1903.08329 [cs, stat]. URL: http://arxiv.org/abs/1903.08329.

[208] Alessandro Rudi and Lorenzo Rosasco. "Generalization Properties of Learning with Random Features." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 3215–3225. URL: http://papers.nips.cc/paper/6914-generalization-properties-of-learning-with-random-features.pdf.

[209] Graham Cormode and Charlie Dickens. *Iterative Hessian Sketch in Input Sparsity Time*. Version 1. Oct. 30, 2019. arXiv: 1910.14166 [cs, stat]. URL: http://arxiv.org/abs/1910.14166.

[210] Rémi Gribonval, Gilles Blanchard, Nicolas Keriven, and Yann Traonmilin. *Compressive Statistical Learning with Random Feature Moments*. 2017. arXiv: 1706.07180.

[211] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. "FMA: A Dataset for Music Analysis." In: *ISMIR*. 2016. arXiv: 1612.01840.

[212] Takuji Onoyama, Masaaki Sibuya, and Hiroshi Tanaka. "Limit Distribution of the Minimum Distance between Independent and Identically Distributed D-Dimensional Random Variables." In: *Statistical Extremes and Applications*. Ed. by J. Tiago Oliveira. Dordrecht: Springer Netherlands, 1984, pp. 549–562. ISBN: 978-90-481-8401-9 978-94-017-3069-3. DOI: 10.1007/978-94-017-3069-3_42. URL: http://link.springer.com/10.1007/978-94-017-3069-3_42.

[213] I. S. Gradshteĭn, I. M. Ryzhik, and Alan Jeffrey. *Table of Integrals, Series, and Products*. 7th ed. Amsterdam ; Boston: Academic Press, 2007. 1171 pp. ISBN: 978-0-12-373637-6.

[214] Anna C Gilbert, Piotr Indyk, Mark Iwen, and Ludwig Schmidt. "Recent Developments in the Sparse Fourier Transform - A compressed Fourier transform for big data." In: *IEEE Signal Process. Mag.* 31.5 (2014), pp. 91–100.

[215] L. Le Magoarou and R. Gribonval. "Flexible Multilayer Sparse Approximations of Matrices and Applications." In: *IEEE Journal of Selected Topics in Signal Processing* 10.4 (June 2016), pp. 688–700. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2016.2543461.

[216] Luc Giffon, Valentin Emiya, Liva Ralaivola, and Hachem Kadri. "QuicK-Means: Acceleration of K-Means by Learning a Fast Transform." Nov. 2019. URL: https://hal.archives-ouvertes.fr/hal-02174845.

[217] Cristian Rusu and Lorenzo Rosasco. *Fast Approximation of Orthogonal Matrices and Application to PCA*. 2019. arXiv: 1907.08697.

[218] Krzysztof Choromanski and Vikas Sindhwani. *Recycling Randomness with Structure for Sublinear Time Kernel Expansions*. May 29, 2016. arXiv: 1605.09049 [cs, stat]. URL: http://arxiv.org/abs/1605.09049.

[219] Felix X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. "Orthogonal Random Features." In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 1975–1983. URL: http://papers.nips.cc/paper/6246-orthogonal-random-features.pdf.

[220] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Francois Fagan, Cedric Gouy-Pailler, Anne Morvan, Nouri Sakr, Tamas Sarlos, and Jamal Atif. "Structured Adaptive and Random Spinners for Fast Machine Learning Computations." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 20–22, 2017, pp. 1020–1029. URL: http://proceedings.mlr.press/v54/bojarski17a.html.

[221] Krzysztof M Choromanski, Mark Rowland, and Adrian Weller. "The Unreasonable Effectiveness of Structured Random Orthogonal Embeddings." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 219–228. URL: http://papers.nips.cc/paper/6626-the-unreasonable-effectiveness-of-structured-random-orthogonal-embeddings.pdf.

[222] Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. "Deep Fried Convnets." In: Proceedings of the IEEE International Conference on Computer Vision. 2015, pp. 1476–1483. URL: http://openaccess.thecvf.com/content_iccv_2015/html/Yang_Deep_Fried_Convnets_ICCV_2015_paper.html.

[223] Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. "Structured Transforms for Small-Footprint Deep Learning." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 3088–3096. URL: http://papers.nips.cc/paper/5869-structured-transforms-for-small-footprint-deep-learning.pdf.

[224] Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. *ACDC: A Structured Efficient Linear Layer*. Mar. 19, 2016. arXiv: 1511.05946 [cs]. URL: http://arxiv.org/abs/1511.05946.

[225] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. "Practical and Optimal LSH for Angular Distance." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 1225–1233. URL: http://papers.nips.cc/paper/5893-practical-and-optimal-lsh-for-angular-distance.pdf.

[226] J. D. Curtó, I. C. Zarza, F. Yang, A. Smola, F. Torre, C. W. Ngo, and L. Gool. *McKernel: A Library for Approximate Kernel Expansions in Log-Linear Time*. Dec. 31, 2019. arXiv: 1702.08159 [cs, stat]. URL: http://arxiv.org/abs/1702.08159.

[227] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. URL: http://ieeexplore.ieee.org/abstract/document/726791/.

[228] *InfMNIST webpage and code*. URL: http://leon.bottou.org/projects/infimnist.

[229] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm." In: *Advances in neural information processing systems*. 2002, pp. 849–856. URL: http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf.

[230] David G Lowe. "Distinctive image features from scale-invariant keypoints." In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[231] Nicolas Tremblay, Gilles Puy, Rémi Gribonval, and Pierre Vandergheynst. "Compressive Spectral Clustering." In: *Machine Learning, Proceedings of the Thirty-Third International Conference (ICML 2016), June*. 2016, pp. 20–22. URL: http://www.jmlr.org/proceedings/papers/v48/tremblay16.pdf.

[232] Jaewon Yang and Jure Leskovec. "Defining and evaluating network communities based on ground-truth." In: *Knowledge and Information Systems* 42.1 (2015), pp. 181–213.

[233] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks." In: *Physical review E* 69.2 (2004), p. 026113.

[234] Nicolas Keriven, Nicolas Tremblay, and Rémi Gribonval. "SketchMLbox: A MATLAB Toolbox for large-scale mixture learning." In: 2016. URL: http://sketchml.gforge.inria.fr/.

[235] Eugene Lukacs and Edgar P. King. "A Property of the Normal Distribution." In: *The Annals of Mathematical Statistics* 25.2 (1954), pp. 389–394.

[236] Alaa Saade, Francesco Caltagirone, Igor Carron, Laurent Daudet, Angélique Drémeau, Sylvain Gigan, and Florent Krzakala. "Random Projections through Multiple Optical Scattering: Approximating Kernels at the Speed of Light." In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference On*. IEEE, 2016, pp. 6215–6219.

[237] Evan Byrne, Rémi Gribonval, and Philip Schniter. "Sketched Clustering via Hybrid Approximate Message Passing." In: *Asilomar Conference on Signals, Systems, and Computers*. 2017.

[238] Andrea Montanari and Marc Mézard. *Information, Physics, and Computation*. Oxford Graduate Text. Oxford university press, 2009.

[239] Judea Pearl. *Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science ..., 1982.

[240] Joris M. Mooij and Hilbert J. Kappen. "Sufficient Conditions for Convergence of the Sum-Product Algorithm." In: *IEEE Transactions on Information Theory* 53.12 (Dec. 2007), pp. 4422–4437. ISSN: 0018-9448. DOI: 10.1109/TIT.2007.909166. arXiv: cs/0504030. URL: http://arxiv.org/abs/cs/0504030.

[241] David L. Donoho, Arian Maleki, and Andrea Montanari. "Message Passing Algorithms for Compressed Sensing." In: *Proceedings of the National Academy of Sciences* 106.45 (Nov. 10, 2009), pp. 18914–18919. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0909892106. arXiv: 0907.3574. URL: http://arxiv.org/abs/0907.3574.

[242] Sundeep Rangan. "Generalized approximate message passing for estimation with random linear mixing." In: *2011 IEEE International Symposium on Information Theory Proceedings, ISIT 2011*. 2011 IEEE International Symposium on Information Theory Proceedings, ISIT 2011. 2011, pp. 2168–2172. DOI: 10.1109/ISIT.2011.6033942. URL: https://nyuscholars.nyu.edu/en/publications/generalized-approximate-message-passing-for-estimation-with-rando.

[243] Amir Beck and Marc Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems." In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[244] Sundeep Rangan, Philip Schniter, Alyson K. Fletcher, and Subrata Sarkar. "On the Convergence of Approximate Message Passing with Arbitrary Matrices." In: *IEEE Transactions on Information Theory* 65.9 (2019), pp. 5339–5351.

[245] Jeremy Vila, Philip Schniter, Sundeep Rangan, Florent Krzakala, and Lenka Zdeborová. "Adaptive Damping and Mean Removal for the Generalized Approximate Message Passing Algorithm." In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 2021–2025.

[246] Mohsen Bayati and Andrea Montanari. "The Dynamics of Message Passing on Dense Graphs, with Applications to Compressed Sensing." In: *IEEE Transactions on Information Theory* 57.2 (2011), pp. 764–785.

[247] Adel Javanmard and Andrea Montanari. "State Evolution for General Approximate Message Passing Algorithms, with Applications to Spatial Coupling." In: *Information and Inference: A Journal of the IMA* 2.2 (2013), pp. 115–144.

[248] Sundeep Rangan, Philip Schniter, Erwin Riegler, Alyson K. Fletcher, and Volkan Cevher. "Fixed Points of Generalized Approximate Message Passing with Arbitrary Matrices." In: *IEEE Transactions on Information Theory* 62.12 (2016), pp. 7464–7474.

[249] J. P. Vila and P. Schniter. "Expectation-Maximization Gaussian-Mixture Approximate Message Passing." In: *IEEE Transactions on Signal Processing* 61.19 (Oct. 2013), pp. 4658–4672. ISSN: 1053-587X. DOI: 10.1109/TSP.2013.2272287.

[250] Daniel Hsu, Sham Kakade, and Tong Zhang. "A Tail Inequality for Quadratic Forms of Subgaussian Random Vectors." In: *Electronic Communications in Probability* 17 (2012).

[251] P. McCullagh and John A. Nelder. *Generalized Linear Models, Second Edition (Chapman & Hall CRC Monographs on Statistics & Applied Probability)*. 2nd ed. Chapman and Hall\/CRC, 1989. ISBN: 978-0-412-31760-6.

[252] Sundeep Rangan, Alyson K. Fletcher, Vivek K. Goyal, and Philip Schniter. "Hybrid Generalized Approximate Message Passing with Applications to Structured Sparsity." In: *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium On*. IEEE, 2012, pp. 1236–1240.

[253] Evan Byrne and Philip Schniter. "Sparse Multinomial Logistic Regression via Approximate Message Passing." In: *IEEE Transactions on Signal Processing* 64.21 (2016), pp. 5485–5498.

[254] Riccardo Gatto and Sreenivasa Rao Jammalamadaka. "The generalized von Mises distribution." In: *Statistical Methodology* 4.3 (2007), pp. 341–353.

[255] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. springer, 2006.

[256] Dimitri P. Bertsekas. "Nonlinear Programming." In: *Athena Scientific* (1999).

[257] Philip Schniter and Sundeep Rangan. "Compressive Phase Retrieval via Generalized Approximate Message Passing." In: *IEEE Transactions on Signal Processing* 63.4 (2014), pp. 1043–1055.

[258] Irit Dinur and Kobbi Nissim. "Revealing information while preserving privacy." In: *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2003, pp. 202–210.

[259] Cynthia Dwork. "Differential Privacy: A Survey of Results." In: *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[260] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "Rappor: Randomized Aggregatable Privacy-Preserving Ordinal Response." In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, pp. 1054–1067.

[261] Apple Privacy Team. "Learning with Privacy at Scale." In: *Apple Mach. Learn. J* 1.9 (2017).

[262] Matteo Testa, Diego Valsesia, Tiziano Bianchi, and Enrico Magli. *Compressed Sensing for Privacy-Preserving Data Processing*. Springer, 2019.

[263] Stanley L. Warner. "Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias." In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69.

[264] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy." In: *Theoretical Computer Science* 9.3-4 (2014), pp. 211–407.

[265] Frank D. McSherry. "Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis." In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, 2009, pp. 19–30.

[266] Isabel Wagner and David Eckhoff. "Technical Privacy Metrics: A Systematic Survey." In: *ACM Computing Surveys (CSUR)* 51.3 (2018), pp. 1–38.

[267] Latanya Sweeney. "K-Anonymity: A Model for Protecting Privacy." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[268] Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. "Unique in the Crowd: The Privacy Bounds of Human Mobility." In: *SCIENTIFIC REPORTS* 3.1376 (2013), p. 1.

[269] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. "Privacy Aware Learning." In: *Advances in Neural Information Processing Systems*. 2012, pp. 1430–1438.

[270] W. Wang, L. Ying, and J. Zhang. "On the Relation Between Identifiability, Differential Privacy, and Mutual-Information Privacy." In: *IEEE Transactions on Information Theory* 62.9 (Sept. 2016), pp. 5018–5029. ISSN: 0018-9448. DOI: 10.1109/TIT.2016.2584610.

[271] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis." In: *Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[272] Borja Balle and Yu-Xiang Wang. "Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising." In: *International Conference on Machine Learning*. International Conference on Machine Learning. July 3, 2018, pp. 394–403. URL: http://proceedings.mlr.press/v80/balle18a.html.

[273] Maoguo Gong, Yu Xie, Ke Pan, Kaiyuan Feng, and A. K. Qin. "A Survey on Differentially Private Machine Learning." In: *IEEE Computational Intelligence Magazine* 15.2 (2020), pp. 49–64.

[274] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. "Practical Privacy: The SuLQ Framework." In: *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, 2005, pp. 128–138.

[275] Zhigang Lu and Hong Shen. "A Convergent Differentially Private K-Means Clustering Algorithm." In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Qiang Yang, Zhi-Hua Zhou, Zhiguo Gong, Min-Ling Zhang, and Sheng-Jun Huang. Vol. 11439. Cham: Springer International Publishing, 2019, pp. 612–624. ISBN: 978-3-030-16147-7 978-3-030-16148-4. DOI: 10.1007/978-3-030-16148-4_47. URL: http://link.springer.com/10.1007/978-3-030-16148-4_47.

[276] Richard Nock, Raphaël Canyasse, Roksana Boreli, and Frank Nielsen. "K-Variates++: More Pluses in the k-Means++." In: *International Conference on Machine Learning*. 2016, pp. 145–154.

[277] Yuncheng Wu, Yao Wu, Hui Peng, Juru Zeng, Hong Chen, and Cuiping Li. "Differentially private density estimation via Gaussian mixtures model." In: *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE. 2016, pp. 1–6.

[278] Mijung Park, James Foulds, Kamalika Choudhary, and Max Welling. "DP-EM: Differentially Private Expectation Maximization." In: *Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics. Apr. 10, 2017, pp. 896–904. URL: http://proceedings.mlr.press/v54/park17c.html.

[279] Wahbeh Qardaji, Weining Yang, and Ninghui Li. "Differentially Private Grids for Geospatial Data." In: *Data Engineering (ICDE), 2013 IEEE 29th International Conference On*. IEEE, 2013, pp. 757–768.

[280] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. "Differentially Private K-Means Clustering." In: *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. ACM, 2016, pp. 26–37.

[281] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. "Private Coresets." In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC '09. New York, NY, USA: ACM, 2009, pp. 361–370. ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536465. URL: http://doi.acm.org/10.1145/1536414.1536465.

[282] Dan Feldman, Chongyuan Xiang, Ruihao Zhu, and Daniela Rus. "Coresets for Differentially Private K-Means Clustering and Applications to Privacy in Mobile Sensor Networks." In: *Information Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE International Conference On*. IEEE, 2017, pp. 3–16.

[283] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. "Analyze Gauss: Optimal Bounds for Privacy-Preserving Principal Component Analysis." In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing - STOC '14*. The 46th Annual ACM Symposium. New York, New York: ACM Press, 2014, pp. 11–20. ISBN: 978-1-4503-2710-7. DOI: 10.1145/2591796.2591883. URL: http://dl.acm.org/citation.cfm?doid=2591796.2591883.

[284] Kamalika Chaudhuri, Anand Sarwate, and Kaushik Sinha. "Near-Optimal Differentially Private Principal Components." In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 989–997. URL: http://papers.nips.cc/paper/4565-near-optimal-differentially-private-principal-components.pdf.

[285] Wuxuan Jiang, Cong Xie, and Zhihua Zhang. "Wishart Mechanism for Differentially Private Principal Components Analysis." In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[286] H. Imtiaz and A. D. Sarwate. "Symmetric Matrix Perturbation for Differentially-Private Principal Component Analysis." In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Mar. 2016, pp. 2339–2343. DOI: 10.1109/ICASSP.2016.7472095.

[287] Michael Kapralov and Kunal Talwar. "On Differentially Private Low Rank Approximation." In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2013, pp. 1395–1414.

[288] Kareem Amin, Travis Dick, Alex Kulesza, Andres Munoz, and Sergei Vassilvitskii. "Differentially Private Covariance Estimation." In: *Advances in Neural Information Processing Systems*. 2019, pp. 14190–14199.

[289] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. "PrivGene: Differentially Private Model Fitting Using Genetic Algorithms." In: *Proceedings of the 2013 International Conference on Management of Data - SIGMOD '13*. The 2013 International Conference. New York, New York, USA: ACM Press, 2013, p. 665. ISBN: 978-1-4503-2037-5. DOI: 10.1145/2463676.2465330. URL: http://dl.acm.org/citation.cfm?doid=2463676.2465330.

[290] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. "Differentially Private Clustering in High-Dimensional Euclidean Spaces." In: *International Conference on Machine Learning*. 2017, pp. 322–331.

[291] Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. "Privacy via the Johnson-Lindenstrauss Transform." In: *Journal of Privacy and Confidentiality* 5.1 (2013), pp. 39–71.

[292] S. Zhou, K. Ligett, and L. Wasserman. "Differential Privacy with Compression." In: *2009 IEEE International Symposium on Information Theory*. 2009 IEEE International Symposium on Information Theory. June 2009, pp. 2718–2722. DOI: 10.1109/ISIT.2009.5205863.

[293] Moritz Hardt and Aaron Roth. "Beating Randomized Response on Incoherent Matrices." In: *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '12. New York, NY, USA: Association for Computing Machinery, May 19, 2012, pp. 1255–1268. ISBN: 978-1-4503-1245-5. DOI: 10.1145/2213977.2214088. URL: https://doi.org/10.1145/2213977.2214088.

[294] Jalaj Upadhyay. "The Price of Privacy for Low-Rank Factorization." In: *Advances in Neural Information Processing Systems*. 2018, pp. 4176–4187.

[295] Raman Arora, Vladimir braverman, and Jalaj Upadhyay. "Differentially Private Robust Low-Rank Approximation." In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 4137–4145. URL: http://papers.nips.cc/paper/7668-differentially-private-robust-low-rank-approximation.pdf.

[296] Benjamin Coleman and Anshumali Shrivastava. *A One-Pass Private Sketch for Most Machine Learning Tasks*. June 16, 2020. arXiv: 2006.09352 [cs, stat]. URL: http://arxiv.org/abs/2006.09352.

[297] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. "Differentially Private Empirical Risk Minimization." In: *Journal of Machine Learning Research* 12 (Mar 2011), pp. 1069–1109.

[298] Di Wang, Minwei Ye, and Jinhui Xu. "Differentially Private Empirical Risk Minimization Revisited: Faster and More General." In: *Advances in Neural Information Processing Systems*. 2017, pp. 2722–2731.

[299] Matej Balog, Ilya Tolstikhin, and Bernhard Schölkopf. "Differentially Private Database Release via Kernel Mean Embeddings." In: *International Conference on Machine Learning*. International Conference on Machine Learning. July 3, 2018, pp. 414–422. URL: http://proceedings.mlr.press/v80/balog18a.html.

[300] Frederik Harder, Kamil Adamczewski, and Mijung Park. *Differentially Private Mean Embeddings with Random Features (DP-MERF) for Simple & Practical Synthetic Data Generation*. Mar. 10, 2020. arXiv: 2002.11603 [cs, stat]. URL: http://arxiv.org/abs/2002.11603.

[301] Dmitry Treschev and Oleg Zubelevich. *Introduction to the perturbation theory of Hamiltonian systems*. Springer Science & Business Media, 2009.

[302] Eric Benhamou, Jamal Atif, and Rida Laraki. *Operator Norm Upper Bound for Sub-Gaussian Tailed Random Matrices*. Jan. 19, 2019. arXiv: 1812.09618 [math]. URL: http://arxiv.org/abs/1812.09618.

[303] Sebastian Meiser. "Approximate and Probabilistic Differential Privacy Definitions." In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 277.

[304] Vijay Bhattiprolu, Mrinalkanti Ghosh, Venkatesan Guruswami, Euiwoong Lee, and Madhur Tulsiani. "Approximability of p → q Matrix Norms: Generalized Krivine Rounding and Hypercontractive Hardness." In: *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms*. 0 vols. Proceedings. Society for Industrial and Applied Mathematics, Jan. 1, 2019, pp. 1358–1368. DOI: 10.1137/1.9781611975482.83. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611975482.83.

[305] Boaz Barak, Fernando GSL Brandao, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. "Hypercontractivity, Sum-of-Squares Proofs, and Their Applications." In: *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*. 2012, pp. 307–326.

[306] Daureen Steinberg. "Computation of Matrix Norms with Applications to Robust Optimization." In: *Research thesis, Technion-Israel University of Technology* 2 (2005).

[307] Xinzhen Zhang, Chen Ling, and Liqun Qi. "The Best Rank-1 Approximation of a Symmetric Tensor and Related Spherical Optimization Problems." In: *SIAM Journal on Matrix Analysis and Applications* 33.3 (2012), pp. 806–821.

[308] Shmuel Friedland. "Best Rank One Approximation of Real Symmetric Tensors Can Be Chosen Symmetric." In: *Frontiers of Mathematics in China* 8.1 (Feb. 2013), pp. 19–40. ISSN: 1673-3452, 1673-3576. DOI: 10.1007/s11464-012-0262-x. URL: http://link.springer.com/10.1007/s11464-012-0262-x.

[309] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "A Multilinear Singular Value Decomposition." In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278.

[310] Nick Vannieuwenhoven, Raf Vandebril, and Karl Meerbergen. "A New Truncation Strategy for the Higher-Order Singular Value Decomposition." In: *SIAM Journal on Scientific Computing* 34.2 (Jan. 2012), A1027–A1052. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/110836067. URL: http://epubs.siam.org/doi/10.1137/110836067.

[311] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "On the Best Rank-1 and Rank-(r 1, r 2,..., Rn) Approximation of Higher-Order Tensors." In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1324–1342.

[312] Liqi Wang and Moody T. Chu. "On the Global Convergence of the Alternating Least Squares Method for Rank-One Approximation to Generic Tensors." In: *SIAM Journal on Matrix Analysis and Applications* 35.3 (2014), pp. 1058–1072.

[313] Berkant Savas and Lek-Heng Lim. "Quasi-Newton Methods on Grassmannians and Multilinear Approximations of Tensors." In: *SIAM Journal on Scientific Computing* 32.6 (2010), pp. 3352–3393.

[314] Yu Guan, Moody T. Chu, and Delin Chu. "SVD-Based Algorithms for the Best Rank-1 Approximation of a Symmetric Tensor." In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (Jan. 2018), pp. 1095–1115. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/17M1136699. URL: https://epubs.siam.org/doi/10.1137/17M1136699.

[315] Eleftherios Kofidis and Phillip A. Regalia. "On the Best Rank-1 Approximation of Higher-Order Supersymmetric Tensors." In: *SIAM Journal on Matrix Analysis and Applications* 23.3 (2002), pp. 863–884.

[316] Tamara G. Kolda and Jackson R. Mayo. "Shifted Power Method for Computing Tensor Eigenpairs." In: *SIAM Journal on Matrix Analysis and Applications* 32.4 (Oct. 1, 2011), pp. 1095–1124. ISSN: 0895-4798. DOI: 10.1137/100801482. URL: https://epubs.siam.org/doi/abs/10.1137/100801482.

[317] Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. 2nd ed. 2019. Springer Series in Computational Mathematics 56. Springer International Publishing, 2019. ISBN: 978-3-030-35553-1.

[318] W. Wang, L. Ying, and J. Zhang. "On the Relation Between Identifiability, Differential Privacy, and Mutual-Information Privacy." In: *IEEE Transactions on Information Theory* 62.9 (Sept. 2016), pp. 5018–5029. ISSN: 0018-9448. DOI: 10.1109/TIT.2016.2584610.

[319] Raef Bassily, Adam Smith, and Abhradeep Thakurta. "Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds." In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. 2014.

[320] Borja Balle, Gilles Barthe, and Marco Gaboardi. "Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences." In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 6277–6287. URL: http://papers.nips.cc/paper/7865-privacy-amplification-by-subsampling-tight-analyses-via-couplings-and-divergences.pdf.

[321] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, Min Lyu, and Hongxia Jin. "Differentially Private K-Means Clustering and a Hybrid Approach to Private Optimization." In: *ACM Trans. Priv. Secur.* 20.4 (Oct. 2017), 16:1–16:33. ISSN: 2471-2566. DOI: 10.1145/3133201. URL: http://doi.acm.org/10.1145/3133201.

[322] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. "FMA: A Dataset for Music Analysis." In: *ISMIR*. 2016. arXiv: 1612.01840.

[323] Irène Waldspurger and Alden Waters. *Rank Optimality for the Burer-Monteiro Factorization*. Dec. 7, 2018. arXiv: 1812.03046 [math]. URL: http://arxiv.org/abs/1812.03046.

[324] Meng Huang and Zhiqiang Xu. *Solving Systems of Quadratic Equations via Exponential-Type Gradient Descent Algorithm*. 2018. arXiv: 1806.00904.

[325] Kaihui LIU. "Structured Signal Recovery from Quadratic Measurements." In: SPARS Workshop. Toulouse, 2019.

[326] David G. Luenberger. *Optimization by Vector Space Methods* (*Wiley Professional*). 1969th ed. Wiley-Interscience, 1969. ISBN: 978-0-471-18117-0.

[327] Lucas Rencker, Francis Bach, Wenwu Wang, and Mark D. Plumbley. "Sparse Recovery and Dictionary Learning From Nonlinear Compressive Measurements." In: *IEEE Transactions on Signal Processing* 67.21 (Nov. 2019), pp. 5659–5670. ISSN: 1941-0476. DOI: 10.1109/TSP.2019.2941070.

[328] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal (auth.) *Convex Analysis and Minimization Algorithms I: Fundamentals*. 1st ed. Grundlehren Der Mathematischen Wissenschaften 305. Springer-Verlag Berlin Heidelberg, 1993. ISBN: 978-3-642-08161-3.

[329] Martijn van Manen. "The Geometry of Conflict Sets." S.l.: s.n.], 2003. ISBN: 9789039334164.

[330] Dirk Siersma. "Properties of Conflict Sets in the Plane." In: *Banach Center Publications* 50.1 (1999), pp. 267–276. ISSN: 0137-6934, 1730-6299. DOI: 10.4064/-50-1-267-276. URL: http://www.impan.pl/get/doi/10.4064/-50-1-267-276.

[331] Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. "A Survey on Graph Kernels." In: *Applied Network Science* 5.1 (Dec. 2020), p. 6. ISSN: 2364-8228. DOI: 10.1007/s41109-019-0195-3. arXiv: 1903.11835. URL: http://arxiv.org/abs/1903.11835.

[332] Walter Rudin. *Real and Complex Analysis*. 3°. McGraw-Hill, 1987. ISBN: 978-0-07-054234-1.

[333] Leopold Kronecker. *Näherungsweise ganzzahlige auflösung linearer gleichungen*. 1884.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Méthodes efficaces pour l'apprentissage compressif avec garanties de confidentialité

Mots-clés : apprentissage compressif, apprentissage à grande échelle, parcimonie, matrices structurées, confidentialité différentielle, propagation de convictions.

Résumé : Ce travail de thèse, qui se situe à l'interface entre traitement du signal, informatique et statistiques, vise à l'élaboration de méthodes d'apprentissage automatique à grande échelle et de garanties théoriques associées. Il s'intéresse en particulier à l'apprentissage compressif, un paradigme dans lequel le jeu de données est compressé en un unique vecteur de moments généralisés aléatoires, appelé le sketch et contenant l'information nécessaire pour résoudre de manière approchée la tâche d'apprentissage considérée. Le schéma de compression utilisé permet de tirer profit d'une architecture distribuée ou de traiter des données en flux, et a déjà été utilisé avec succès sur plusieurs tâches d'apprentissage non-supervisé: partitionnement type k-moyennes, modélisation de densité avec modèle de mélange gaussien, analyse en composantes principales.

Les contributions de la thèse s'intègrent dans ce cadre de plusieurs manières. D'une part, il est montré qu'en bruitant le sketch, des garanties de confidentialité (différentielle) peuvent être obtenues; des bornes exactes sur le niveau de bruit requis sont données, et une comparaison expérimentale permet d'établir que l'approche proposée est compétitive vis-à-vis d'autres méthodes récentes. Ensuite, le schéma de compression est adapté pour utiliser des matrices aléatoires structurées, qui permettent de réduire significativement les coûts de calcul et rendent possible l'utilisation de méthodes compressives sur des données de grande dimension. Enfin, un nouvel algorithme basé sur la propagation de convictions est proposé pour résoudre la phase d'apprentissage (à partir du sketch) pour le problème de partitionnement type k-moyennes.

Title: Efficient and Privacy-Preserving Compressive Learning

Keywords: compressive learning, large-scale clustering, sparsity, structured matrices, differential privacy, belief propagation.

Abstract: The topic of this Ph.D. thesis lies on the borderline between signal processing, statistics and computer science. It mainly focuses on compressive learning, a paradigm for large-scale machine learning in which the whole dataset is compressed down to a single vector of randomized generalized moments, called the sketch. An approximate solution of the learning task at hand is then estimated from this sketch, without using the initial data. This framework is by nature suited for learning from distributed collections or data streams, and has already been instantiated with success on several unsupervised learning tasks such as k-means clustering, density fitting using Gaussian mixture models, or principal component analysis.

We improve this framework in multiple directions. First, it is shown that perturbing the sketch with additive noise is sufficient to derive (differential) privacy guarantees. Sharp bounds on the noise level required to obtain a given privacy level are provided, and the proposed method is shown empirically to compare favourably with state-of-the-art techniques. Then, the compression scheme is modified to leverage structured random matrices, which reduce the computational cost of the framework and make it possible to learn on high-dimensional data. Lastly, we introduce a new algorithm based on message passing techniques to learn from the sketch for the k-means clustering problem. These contributions open the way for a broader application of the framework.