Compressive Clustering with an Optical Processing Unit

Luc Giffon, Rémi Gribonval

Compressive Clustering with an Optical Processing Unit

Luc GIFFON[1], Rémi GRIBONVAL[1]

[1]Univ Lyon, Inria, CNRS, ENS de Lyon, UCB Lyon 1, LIP UMR 5668, F-69342, Lyon, France
luc.giffon@ens-lyon.fr, remi.gribonval@ens-lyon.fr

Résumé – Nous proposons une procédure de *sketching* pour le *clustering* compressif utilisant des processeurs optiques (OPU) pour calculer les composantes de Fourier alatoires. Cette procédure fait intervenir une stratégie novatrice qui permet de choisir efficacement l'échelle du sketch.

Abstract – We explore the use of Optical Processing Units (OPU) to compute random Fourier features for sketching, and adapt the overall compressive clustering pipeline to this setting. We also propose some tools to help tuning a critical hyper-parameter of compressive clustering.

1 Introduction

A popular tool to address high-dimensional machine learning problems is the use of Random Fourier Features (RFF), which provide nonlinear feature maps exploiting random projections [14]. This paper focuses on the use of RFF to build sketches for compressive K-means clustering [9].

A sketch is the expected value of a feature map $\Phi(\cdot)$ on samples of a data distribution. Clusters can be learnt from a sketch built using properly chosen RFF [9], and *sketching* and *learning from the sketch* can happen on different machines. This gives rise [16] to an asynchronous framework in which the (possibly private [3]) sketch is constructed on a low-resource *sketching device* and sent to a central server where the clusters can be recovered from it. Yet, sketching for asynchronous compressive clustering is currently not a turnkey procedure.

A first major obstacle is to choose the distribution of the random matrix involved in RFF. Despite existing heuristics [1, 9], tuning this distribution is still something of an art essentially boiling down to trying several distributions for the random matrix, to build as many sketches and learn from them before picking the best. This procedure is expensive, and all the more unsatisfying as it contradicts the very philosophy of sketched learning which is to make only one pass on the dataset.

The second issue with sketching is its reliance on a very large matrix to compute RFF in high dimension. Chatalic et al. in [2] alleviate this burden for compressive clustering by re-using a classical method from the kernel community, based on replacing a dense random Gaussian matrix with products of structured and sparse random matrices associated to fast linear transforms. Here we investigate an alternate approache exploiting a novel co-processor called Optical Processing Unit (OPU), which is dedicated to perform random matrix multiplications by leveraging the physical properties of light traveling through a scattering medium. In theory, the OPUs random multiplications have a linear time and space complexity with respect to the sum of the input and output dimensions [11]. An OPU also has a constant power consumption of 30W.

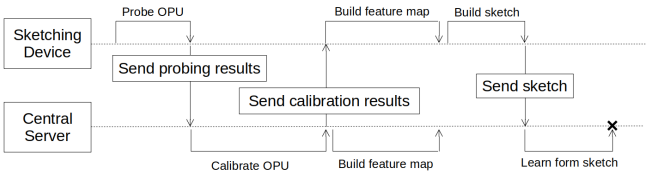

FIGURE 1 – Sequence diagram of the proposed pipeline.

OPUs can speed up RFF for kernel machines [11] and it seems natural to apply them to compressive clustering. However, each OPU is associated to its unique draw of a random matrix, so it is not possible to have the same OPU on the sketching device and on the central server. More importantly, some steps of the main algorithms to learn from a sketch require explicit access to the underlying random matrix, while an OPU only implements multiplication by this matrix. Fortunately, the matrix associated to an OPU can be estimated via calibration, which leads us to propose a new asynchronous compressive clustering pipeline. Its main steps, and accordingly the contributions of this paper, are the following (see also Figure 1) :

1. Feature maps are built on the sketching device and the central server by leveraging calibration (Section 2) ;
2. We propose a trick to simultaneously compute sketches for different RFF distributions on the sketching device (Section 3.1) ;
3. We propose a criterion to choose the best distribution and send the corresponding sketch to the central server (Section 3.2) ;
4. The server learns the clusters from the sketch (Experimental results in Section 4).

1.1 Sketching for compressive clustering

A training collection made of N observations in D dimensions $\{\mathbf{x}_i\}_{i=1}^N$ is represented by its empirical data distribution $\mathcal{P}_{\hat{\mathcal{X}}} := \frac{1}{N}\sum_{i=1}^N \delta_{\mathbf{x}_i}$. The RFF map $\Phi : \mathbb{R}^D \to \mathbb{C}^M$ is defined [1] as $\Phi(\mathbf{x}) := \exp(-i\mathbf{W}\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{M\times D}$ is a random matrix yet to be specified. The sketch $\hat{\mathbf{z}} \in \mathbb{C}^M$ is then defined as

1. The exponential $\exp : \mathbb{C} \to \mathbb{C}$ is applied entry-wise on its input.

the empirical average of this feature map over the training set :

$$\hat{\mathbf{z}} := \frac{1}{N}\sum_{i=1}^{N}\Phi(\mathbf{x}_i) = \mathbb{E}_{\mathbf{x}\sim\mathcal{P}_{\hat{\mathcal{X}}}}[\Phi(\mathbf{x})]. \quad (1)$$

For compressive K-means clustering, M is usually in the order of $\mathcal{O}(KD)$, with K the prescribed number of clusters. Each line $\mathbf{w}_j$ of $\mathbf{W}$ (so called *frequencies* due to the link with Fourier analysis) is typically a unit-norm direction $\mathbf{u}_j \in \mathbb{R}^D$ sampled uniformly and independently on the unit-sphere, multiplied by a radius $\rho_j \in \mathbb{R}^+$ sampled from a distribution akin to a folded Gaussian [9] with variance 1, and finally rescaled by $\frac{1}{\sigma}$. Stacked together, these frequencies give the following definition for the matrix $\mathbf{W}$:

$$\mathbf{W} := \frac{1}{\sigma}\mathbf{\Delta}\mathbf{U}, \quad (2)$$

with $\mathbf{\Delta} := \texttt{diag}(\rho_1, \ldots, \rho_M)$, $\mathbf{U} := [\mathbf{u}_1, \ldots, \mathbf{u}_M]^T \in \mathbb{R}^{M\times D}$. The distribution of $\mathbf{W}$ is parameterized by the value of the sketching scale σ, which must be tuned with respect to the dataset.

1.2 Learning centroids from a RFF sketch

K-means clustering seeks K centroids $\mathbf{\Theta} := \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ as close as possible, in the sense of the Euclidean distance, to the observed data points. This corresponds to minimizing

$$\hat{\mathcal{R}}(\mathbf{\Theta}) := \sum_{i=1}^{N}\|\mathbf{x}_i - \mathbf{\Theta}(\mathbf{x}_i)\|^2 \quad (3)$$

where $\mathbf{\Theta}(\cdot)$ maps its input to the closest centroid. Alternatively, K-means clustering can be interpreted as minimizing the Wasserstein-2 distance between the empirical data distribution $\mathcal{P}_{\hat{\mathcal{X}}}$ and a mixture of K Diracs [20] $\mathcal{P}_{\mathbf{\Theta},\boldsymbol{\alpha}} := \sum_{k=1}^{K}\alpha_k\delta_{\mathbf{c}_k}$ with $\sum_{k=1}^{K}\alpha_k = 1$. Similarly in **compressive K-means clustering**, the centroids are chosen by minimizing the distance $\left\|\mathcal{A}(\mathcal{P}_{\mathbf{\Theta},\boldsymbol{\alpha}}) - \mathcal{A}(\mathcal{P}_{\hat{\mathcal{X}}})\right\|_2^2$ between the *sketches* of these distributions. where the *sketching operator* $\mathcal{A} : \mathcal{P} \to \mathbb{C}^M$ gives a vector representation, the sketch, of its input distribution. The data sketch (1) is $\hat{\mathbf{z}} = \mathcal{A}(\mathcal{P}_{\hat{\mathcal{X}}})$, while $\mathcal{A}(\mathcal{P}_{\mathbf{\Theta},\boldsymbol{\alpha}}) = \sum_{k=1}^{K}\alpha_k\Phi(\mathbf{c}_k)$.

Compressive Learning Orthogonal Matching Pursuit with Replacement (CLOMP-R) [9] is a heuristic algorithm to minimize $\left\|\mathcal{A}(\mathcal{P}_{\mathbf{\Theta},\boldsymbol{\alpha}}) - \mathcal{A}(\mathcal{P}_{\hat{\mathcal{X}}})\right\|_2^2$. Given a feature map $\Phi(\cdot)$, the only input to CLOMP-R is the empirical sketch $\hat{\mathbf{z}}$ (and *not* the training dataset). CLOMP-R involves gradient descent exploiting the derivative of the feature map $\Phi(\cdot)$. This is a challenging requirement when implementing sketches with an OPU.

1.3 Optical Processing Units

An "**Optical Processing Unit (OPU)**" [15] is a hardware implementing a function $A(\cdot)$ to approximately perform a random matrix product with an input vector $\mathbf{x} \in \mathbb{R}^D$. It leverages the diffusion properties of light traveling through a scattering medium, and can be modeled as :

$$A(\mathbf{x}) = D(\mathbf{A}E(\mathbf{x}) + \boldsymbol{\epsilon}). \quad (4)$$

Functions $E(\cdot)$ and $D(\cdot)$ are respectively bit plane encoding and decoding functions needed because the physical, low level, OPU only takes as input binary vectors (vectors in $\{0,1\}^D$); vector $\boldsymbol{\epsilon} \in \mathbb{R}^M$ is a Gaussian noise sampled for each input $\mathbf{x}$ and caused by the analogic implementation of the function $A(\cdot)$; finally, and more importantly, the *transmission matrix* $\mathbf{A} \in \mathbb{R}^{M\times D}$ is sampled from a centered Gaussian distribution, and can be considered as fixed once the OPU is built.

Calibration An estimate $\hat{A}$ of the transmission matrix $\mathbf{A}$ can be obtained via calibration : similarly to [6], the linearity of the OPU can be leveraged to create a set of linear equations allowing to recover the coefficients in $\mathbf{A}$. This is done by applying the transmission matrix $\mathbf{A}$ to the columns of the Hadamard matrix $\mathbf{H}$ in dimension D (up to zero padding to the first greater power of 2) [7] and solving the M resulting linear equations.

2 Building OPU-based feature maps

In the proposed pipeline (see Figure 1), the sketching device benefits the low-power property of the OPU to sketch the data with an energy-efficient feature map $\Phi_{\texttt{OPU}}(\cdot)$. The central server, which has no resource constraint, builds a sort of *digital twin feature map* $\hat{\Phi}_{\texttt{OPU}}(\cdot) \approx \Phi_{\texttt{OPU}}(\cdot)$ to learn from the sketch.

The sketching device feature map $\Phi_{\texttt{OPU}}(\cdot)$ The core idea of using the OPU for sketching is to take advantage of the OPU function $A(\mathbf{x})$ to perform the random matrix multiplication in $\Phi(\mathbf{x})$ in place of the $\mathbf{W}\mathbf{x}$ product. However, the transmission matrix $\mathbf{A}$ of the OPU has random Gaussian entries while $\mathbf{W}$ is different, see (2). Hence, the output of $A(\cdot)$ must be rescaled by a diagonal matrix in order to fit the definition of $\mathbf{W}$.

Denote $\hat{\mathbf{a}}_i$ the rows of the calibrated transmission matrix $\hat{\mathbf{A}}$ (see below) and let $\mathbf{N}$ be the diagonal with entries $\mathbf{N}_{i,i} := \frac{1}{\|\hat{\mathbf{a}}_i\|_2}$. We propose to re-scale the output of $A(\cdot)$ with $\mathbf{N}$ such that $\mathbf{N}A(\mathbf{x})$ can be identified with $\mathbf{U}\mathbf{x}$. Scaling again with $\mathbf{\Delta}$ and $\frac{1}{\sigma}$ to mimic (2) gives the following expression for $\Phi_{\texttt{OPU}}$:

$$\Phi_{\texttt{OPU}}(\mathbf{x}) := \exp\left(-i\frac{1}{\sigma}\mathbf{\Delta}\mathbf{N}A(\mathbf{x})\right). \quad (5)$$

The calibration step To determine $\mathbf{N}$, the central server estimates $\hat{\mathbf{A}}$ (see Section 1.3) by exploiting probed vectors sent from the sketching device to the server. Such vectors are obtained from the application of the OPU to the columns of the Hadamard matrix $\mathbf{H}$. From the calibrated matrix, the server can send back the (diagonal of) the matrix $\mathbf{\Delta}\mathbf{N}$ to the sketching device, as it is needed to compute $\Phi_{\texttt{OPU}}(\cdot)$ according to (5) and to compute a sketch $\hat{\mathbf{z}}_{\texttt{OPU}} := \frac{1}{N}\sum_{i=1}^{N}\Phi_{\texttt{OPU}}(\mathbf{x}_i)$ of the empirical data distribution like in Equation (1). In the process, the sketching device never stores the calibrated transmission matrix $\hat{\mathbf{A}}$ nor anything requiring more than $\mathcal{O}(M)$ bytes of memory.

The digital twin feature map $\hat{\Phi}_{\texttt{OPU}}(\cdot)$ The matrix $\mathbf{\Delta}\mathbf{N}$ is also used on the server to define a mirroring feature map $\hat{\Phi}_{\texttt{OPU}}(\mathbf{x}) := \exp\left(-i\frac{1}{\sigma}\mathbf{\Delta}\mathbf{N}\hat{\mathbf{A}}\mathbf{x}\right)$. This feature map has a known derivative so it can be used to run CLOMP-R and learn the centroids from the sketch $\hat{\mathbf{z}}_{\texttt{OPU}}$ received from the sketching device.

3 Choosing the sketching scale

To achieve good clustering performance one needs to tune the scale σ used for the random projection matrix (2) [3, 5, 9].

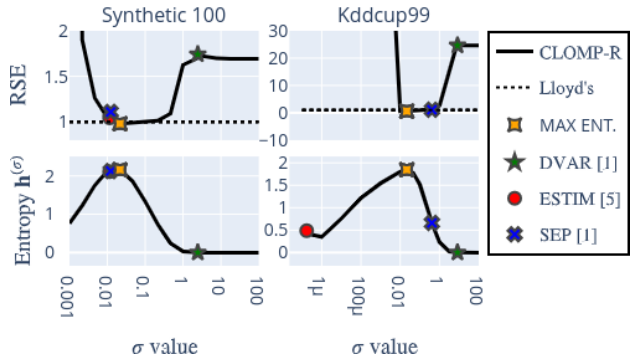

FIGURE 2 – Entropy of sketch VS clustering quality on a synthetic dataset (see Section 4) and KDDCUP99 dataset [19].

In this section, we propose a two-fold procedure to efficiently and accurately select a good scale σ for compressive clustering.

3.1 Efficient grid-search for sketching scale

An experiment on a single dataset needs a total of S different sketching scales $\sigma_1, \ldots, \sigma_S$ (Equation (2)) corresponding to build as many feature maps $\{\Phi^{(1)}, \ldots, \Phi^{(S)}\}$ and to produce as many sketches. Computing these sketches can be prohibitively expensive if done naively one sketch after another. Fortunately, the bulk of the computation can be shared between the S feature maps and producing the sketches can be much more efficient if they are all computed simultaneously. With the method we propose, the total complexity (in flops) of computing these S sketches on silicium (CPU or GPU) drops from $NS\mathcal{O}(MD)$ to $N(\mathcal{O}(SM) + \mathcal{O}(MD))$. For large D, this divides the complexity by S. This speed is enabled by the fact that for each observation $\mathbf{x}$ in the dataset the product $\mathbf{\Delta U x}$ can be computed only once (with a cost $\mathcal{O}(MD)$), then its Kronecker product with the vector $\mathbf{s} := \left[\frac{1}{\sigma^{(1)}}, \ldots, \frac{1}{\sigma^{(S)}}\right]^T$ is computed in SM flops, and the pointwise exponential $\exp(-i\mathbf{s} \otimes (\mathbf{\Delta U x}))$ returns all RFF feature maps at once with $\mathcal{O}(SM)$ flops ($\otimes$ denotes the Kronecker product). This yields the claimed gain in flops, with a small overhead in space complexity of MS (instead of M) corresponding to storing in memory all the S sketches at once, each of dimension M.

Does this trick also reduce the time cost of computing S sketches with an OPU ? Suppose that the time cost of applying $A(\cdot)$ is equivalent to $C_{\texttt{OPU}}(M + D)$ flops and that $M \gg D$ (in compressive K-means [1] $M = 100KD$ is typical). One can check that the trick multiplies the time cost by $1/C_{\texttt{OPU}} + 1/S$. This leads to a "win-win" situation : if the OPU is moderately efficient –associated to somewhat large constant $C_{\texttt{OPU}} \geq S$– then the proposed grid-search still pays off ; otherwise $C_{\texttt{OPU}} \ll S$ leads to a very fast implementation of $\Phi_{\texttt{OPU}}$ even though grid-search cannot benefit from further acceleration.

In practice, we observed a $\times 2$ speed up on GPU when $S = 10$, $D = 200$ and $M = 500D$. Using a real OPU, we evaluated $C_{\texttt{OPU}} = (\texttt{Time of}\, A(\mathbf{x}))/\left((\texttt{Time of one flop}) \times (M + D)\right) \approx 6000$. This big constant $C_{\texttt{OPU}}$ led to an observed $\times 12$ speed up in the same setting as for the GPU. This was of crucial importance to run the experiments of Section 4.

3.2 Entropy as a proxy for sketch quality

Sketching an empirical distribution (1) is equivalent to sampling its characteristic function on the locations defined by the frequencies in $\mathbf{W}$ (see e.g. [9]). In the high and low frequency regimes (large / small norm $\|\mathbf{w}_j\|_2$), the characteristic function has values close to 0 or 1 respectively. Since this holds whatever the distribution [9], sampling the characteristic function of any distribution with too high or too low frequencies does not allow to distinguish between distributions.

It is thus natural to assume that a well chosen scale σ should ensure that the magnitude of the sketch coefficients are well distributed between 0 and 1. To promote this property, we propose to use an entropy measure. Concretely, the interval $[0, 1]$ is split into B non overlapping bins of size $\frac{1}{B}$. Then, the sketch coefficients are assigned a bin with respect to their magnitude. Hence, a sketch coefficient defines a discrete random variable $\mathcal{B} \in [\![B]\!]$. From one sketch obtained with a particular σ, we can make an estimation of the probability $P_b^{(\sigma)}$ of any sketch coefficient to fall inside the b^{th} bin and then apply the classical Shanon formula for entropy estimation [18], $\mathbf{h}^{(\sigma)}(\mathcal{B}) = -\sum_{b=1}^{B} P_b^{(\sigma)} \log\left(P_b^{(\sigma)}\right)$. We expect this entropy to be smallest when σ leads to magnitudes either all close to 0 or all close to 1. Vice-versa, the highest entropy should lead to a "best" σ.

On Figure 3, the top plots display the Relative Squared Error (RSE) (see Metrics in Section 4) of CLOMP-R with respect to σ. We observe that the range of σ for which the performance of CLOMP-R is equivalent to the Lloyd algorithm (RSE close to one) is relatively narrow. From the observation of the bottom plot –displaying the entropy of the sketch with respect to σ– it appears clearly that the best performance is indeed reached when the entropy of the sketch is at its peak (MAX ENT. marker). Comparing results on both dataset, we remark that other existing methods to select σ give unpredictable performance depending on the datasets : DVAR is the global data variance [1] and doesn't perform good on any dataset ; ESTIM is an iterative method proposed in [9] and perform well on the synthetic dataset but very bad on KDDCUP99 ; SEP is an expectation of the separation between clusters[1] and perform relatively well in both dataset but still worse than the entropy based estimator.

4 Assessing the quality of centroids

The proposed pipeline involving the OPU and its calibration is evaluated by comparing the quality of the obtained centroids compared with that of the vanilla CLOMP-R on two synthetic Gaussian mixture datasets. Before proceeding to the actual results, we provide some experimental details.

Data We use a synthetic data generation routine implemented in the `pycle` toolbox [17]. It samples observations from a mixture of $K = 5$ isotropic Gaussians in dimension $D = 10$ or 100 with a inter-cluster/intra-cluster variance ratio equal to 10.

Code We adapted `pycle` [17] to support `pytorch` [12] and `lightonml` [10] . Our version is available online [4].

Hyper-parameters We choose the scale σ among a log-range of $S = 10$ possible values in $[10^{-3}, 1]$ by applying the en-

Method	Synthetic 10			Synthetic 100		
	RSE	AMI	W-Dist.	RSE	AMI	W-Dist.
CLOMP-M	1.02(± 0.0)	0.84(± 0.1)	0.03(± 0.1)	0.98(± 0.1)	0.96(± 0.1)	0.07(± 0.1)
CLOMP-SO	1.02(± 0.0)	0.84(± 0.1)	0.03(± 0.1)	0.98(± 0.1)	0.96(± 0.1)	0.08(± 0.1)
CLOMP-RO	1.04(± 0.0)	0.80(± 0.1)	0.04(± 0.0)	1.24(± 0.1)	0.85(± 0.1)	0.17(± 0.1)
RAND-CLS	1.53(± 0.1)	0.43(± 0.1)	0.22(± 0.1)	1.88(± 0.2)	0.88(± 0.1)	0.31(± 0.1)
RAND-DATA	1.69(± 0.2)	0.37(± 0.1)	0.25(± 0.1)	2.13(± 0.3)	0.63(± 0.1)	0.39(± 0.1)

TABLE 1 – Results : Averages and standard deviations.

tropy criterion (Section 3.2). We repeat the experiments for 10 samples of the radius matrix $\mathbf{\Delta}$ from the distribution proposed in [9]. Centroids in the CLOMP-R algorithm as well as those of the Lloyd algorithm are initialized randomly in the hyper-cube containing the dataset. We use a sketch size $M = 100KD$.

Metrics & Competing methods We use three metrics computed with respect to centroids returned by the Lloyd's algorithm which serve as an ideal clustering method : (i) the Relative Squared Error (RSE) e.g. the ratio $\frac{\hat{\mathcal{R}}(\mathbf{\Theta}_{\texttt{GPU}})}{\hat{\mathcal{R}}(\mathbf{\Theta}_{\texttt{Lloyd}})}$ [1] (see Equation (3)), the Adjusted Mutual Information(AMI) [21] and the Wasserstein distance (W-Dist.)[8]. Three variants of the pipeline are compared : CLOMP-M ("Matrix"), where a GPU is used to sketch by computing RFF and the vanilla CLOMP-R algorithm is used to learn from the resulting sketch ; CLOMP-SO (Simulated OPU) using a simulated OPU that should behave like an ideal OPU, however with no noise ; CLOMP-RO (Real OPU) using an actual OPU to compute the sketches.

We compare the results with two baselines that randomly sample : (i) each centroid uniformly from all observations in the dataset (RAND-DATA) ; or (ii) the k-th centroid uniformly from all observations *from the k-th cluster* (RAND-CLS ; this highly informed baseline is feasible since the clusters have been synthetically generated). These controls serve as a reference points to assess the quality of clustering with CLOMP-R.

Results Table 1 shows that, on the Synthetic 10 dataset, CLOMP-RO has results similar to the classical CLOMP-M method. On the Synthetic 100 dataset however, while CLOMP-SO behaves similarly to CLOMP-M, the performance of CLOMP-RO is degraded suggesting that it suffers from the analogic noise of the real OPU (4). Yet, the performance of CLOMP-RO remains noticeably better than the baseline RAND-CLS. This confirms that *some* clustering information is caught by the CLOMP-RO pipeline as RAND-CLS is highly informed, contrary to RAND-DATA whose results are clearly worse.

5 Conclusion and perspectives

We propose a compressive clustering pipeline exploiting the energy-efficiency of OPUs to sketch and using the power of CPU/GPU to learn from sketches. While current OPU hardware is sized to fit in a server rack, the obtained results prompt for the development of a minified and embeddable OPU to be deployed on a low-resource sketching device. Further challenges include the investigation and possible correction of the performance degradation observed in higher dimension, as well as the validation of the approach on diverse real world datasets.

Acknowledgement This project was supported in part by the AllegroAssai ANR project ANR-19-CHIA-0009. We acknowledge LightOn™for giving us access to the OPU and the support of the Centre Blaise Pascal's IT test platform at ENS de Lyon. The platform operates the SIDUS solution [13].

Références

[1] A. CHATALIC. "Efficient and privacy-preserving compressive learning". Thèse de doct. Univ. Rennes 1, 2020.

[2] A. CHATALIC, R. GRIBONVAL et N. KERIVEN. "Large-scale high-dimensional clustering with fast sketching". In : *IEEE ICASSP*. IEEE. 2018, p. 4714-4718.

[3] A. CHATALIC et al. "Compressive learning with privacy guarantees". In : *Information and Inference* (2021).

[4] L. GIFFON. *The pycle toolbox*. `https://www.github.com/lucgiffon/pycle`. 2020.

[5] R. GRIBONVAL et al. "Sketching data sets for large-scale learning : Keeping only what you need". In : *IEEE Signal Processing Magazine* 38.5 (2021), p. 12-36.

[6] S. GUPTA et al. "Fast optical system identification by numerical interferometry". In : *IEEE ICASSP*. 2020.

[7] P. P. KANJILAL. *Adaptive prediction and predictive control (p.210)*. 52. IET, 1995.

[8] L. KANTOROVICH. "Mathematical methods of organizing and planning production". In : *Management science* 6.4 (1960), p. 366-422.

[9] N. KERIVEN et al. "Sketching for large-scale learning of mixture models". In : *Information and Inference : A Journal of the IMA* 7.3 (2018), p. 447-508.

[10] LIGHTONAI. *The LightOnML library*. `https://www.github.com/lightonai/lightonml`. 2020.

[11] R. OHANA et al. "Kernel computations from large-scale random features obtained by optical processing units". In : *IEEE ICASSP*. IEEE. 2020, p. 9294-9298.

[12] A. PASZKE et al. "PyTorch : An Imperative Style, High-Performance Deep Learning Library". In : (2019). Sous la dir. de H. WALLACH et al., p. 8024-8035.

[13] Emmanuel QUEMENER et Marianne CORVELLEC. "SIDUS—the solution for extreme deduplication of an operating system". In : *Linux Journal* 2013.235 (2013), p. 3.

[14] A. RAHIMI et B. RECHT. "Random features for large-scale kernel machines". In : *NIPS* 20 (2007).

[15] A. SAADE et al. "Random projections through multiple optical scattering : Approximating kernels at the speed of light". In : *IEEE ICASSP*. IEEE. 2016, p. 6215-6219.

[16] V. SCHELLEKENS. "Extending the Compressive Statistical Learning Framework : Quantization, Privacy, and Beyond". Thèse de doct. Univ. of Edinburgh, U.K., 2021.

[17] V. SCHELLEKENS. *The pycle toolbox*. `https://www.github.com/schellekensv/pycle`. 2020.

[18] C-E. SHANNON. "A mathematical theory of communication". In : *ACM SIGMOBILE mobile computing and communications review* 5.1 (2001), p. 3-55.

[19] M. TAVALLAEE et al. "A detailed analysis of the KDD CUP 99 data set". In : *IEEE CISDA*. Ieee. 2009, p. 1-6.

[20] T. VAYER et R. GRIBONVAL. "Controlling Wasserstein distances by Kernel norms with application to Compressive Statistical Learning". In : *arXiv :2112.00423* (2021).

[21] N. X. VINH, J. EPPS et J. BAILEY. "Information theoretic measures for clusterings comparison : Variants, properties, normalization and correction for chance". In : *JMLR* 11 (2010), p. 2837-2854.