HAL
open science

Nonsmooth Optimization for Statistical Learning with Structured Matrix Regularization

Federico Pierucci

To cite this version:

Federico Pierucci. Nonsmooth Optimization for Statistical Learning with Structured Matrix Regularization. Machine Learning [stat.ML]. Université Grenoble-Alpes, 2017. English. NNT : . tel-01572186v1

HAL Id: tel-01572186

https://inria.hal.science/tel-01572186v1

Submitted on 4 Aug 2017 (v1), last revised 12 Jan 2018 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Mathématiques et Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Federico Pierucci

Thèse dirigée par **Pr. Anatoli Juditsky**
et codirigée par **Dr. Jérôme Malick, Pr. Zaid Harchaoui**

préparée au sein **Laboratoire Jean Kuntzmann**
et de **École Doctorale Mathmatiques, Sciences et technologies de l'information, Informatique**

Nonsmooth Optimization for Statistical Learning with Structured Matrix Regularization

Thèse soutenue publiquement le **23 juin 2017**,
devant le jury composé de :

Pr. Massih-Reza Amini
Université Grenoble Alpes, Grenoble, France, Président
Pr. Alexander Nazin
Institute of Control Sciences RAS, Moscow, Russia, Rapporteur
Pr. Stéphane Chrétien
National Physical Laboratory, Teddington, Middlesex, UK, Rapporteur
Dr. Nelly Pustelnik
CNRS, ENS Lyon, Lyon, France, Examinatrice
Pr. Joseph Salmon
TELECOM ParisTech, Paris, France, Examinateur
Pr. Anatoli Juditsky
Université Grenoble Alpes, Grenoble, France, Directeur de thèse
Dr. Jérôme Malick
CNRS, Université Grenoble Alpes, Grenoble, France, Co-Directeur de thèse
Pr. Zaid Harchaoui
University of Washington, WA, USA, Co-Directeur de thèse

Contents

1 First-order optimization for machine learning: context and contributions 12

1.1 Elements of statistical learning . . . 12
1.1.1 Predictor functions . . . 13
1.1.2 Find the model in theory . . . 14
1.1.3 Find the model in practice . . . 14
1.1.4 Regularized problems . . . 15
1.1.5 Parameter tuning . . . 15
1.2 Elements of convex optimization . . . 16
1.2.1 Basic elements for convex analysis . . . 16
1.2.2 Convex optimization problems . . . 18
1.2.3 Gauges and atomic norms . . . 20
1.2.4 Oracle . . . 20
1.2.5 Optimization algorithms . . . 24
1.3 First order algorithms for convex optimization - unconstrained case . . . 29
1.3.1 Gradient descent algorithm . . . 31
1.3.2 Proximal gradient algorithm . . . 32
1.3.3 Accelerated proximal gradient algorithm . . . 34
1.3.4 Accelerated gradient descent algorithm . . . 36
1.3.5 Composite conditional gradient algorithm . . . 36
1.3.6 Lagrangian matching pursuit algorithm . . . 38
1.4 First order algorithms for convex optimization - constrained case . . . 39
1.4.1 Projected subgradient algorithm . . . 39
1.4.2 Projected gradient algorithm . . . 40
1.4.3 Conditional gradient algorithm . . . 42
1.4.4 Matching pursuit algorithm . . . 44
1.4.5 Orthogonal matching pursuit algorithm . . . 45
1.4.6 Randomized incremental algorithm . . . 45
1.5 Machine learning applications . . . 47
1.5.1 Collaborative filtering for movie advertising . . . 47
1.5.2 Multiclass image classification . . . 48
1.6 Contributions in this context . . . 52
1.6.1 Group Schatten norm . . . 52
1.6.2 Smoothing techniques for learning with first-order optimization . . . 53
1.6.3 Conditional gradient algorithms for doubly non-smooth learning . . . 54

2 Group Schatten norm **57**
2.1 Introduction . . . 57
2.2 Notation . . . 58
2.3 Group Schatten norm: Definition and examples . . . 58
2.4 Group nuclear norm as a convex surrogate . . . 60
2.5 Algorithms for learning with group p-Schatten norm . . . 65
2.5.1 Group p-Schatten norm as regularization penalty . . . 65
2.5.2 (Accelerated) proximal-gradient algorithm . . . 67
2.5.3 Composite conditional gradient . . . 68
2.6 Illustrations . . . 70
2.7 Proposed applications: Initial steps . . . 71
2.7.1 Multiclass classification . . . 71
2.7.2 Collaborative filtering with attributes . . . 74
2.7.3 Compression of a structured database . . . 75
2.7.4 Feature concatenation . . . 76
2.7.5 Combinations . . . 76
2.7.6 Object cosegmentation . . . 77
2.7.7 Unsupervised learning . . . 77
2.7.8 Issue of groups that are unions of other groups . . . 77

3 Smoothing techniques for first-order optimization **78**
3.1 Introduction . . . 78
3.1.1 Smoothing in optimization . . . 78
3.1.2 Contributions and outline of this chapter . . . 79
3.1.3 Recalls in convex analysis . . . 80
3.2 Smoothing by infimal convolution . . . 80
3.2.1 General construction and special cases . . . 81
3.2.2 A simple example of smoothing by saddle-point representation: absolute value . 84
3.2.3 An advanced example of smoothing by saddle-point representation: the top-k function . . . 85
3.3 Smoothing by product convolution . . . 92
3.3.1 General construction . . . 92
3.3.2 Simple examples in $\mathbb{R}$ and $\mathbb{R}^n$ . . . 95
3.4 Smoothing-based first-order methods for doubly nonsmooth learning problems . . . 98
3.4.1 Composite conditional gradient . . . 101
3.4.2 Accelerated proximal gradient algorithm . . . 103
3.4.3 Incremental gradient . . . 105
3.5 Algebraic calculus . . . 106
3.5.1 Fenchel-type approximation . . . 106
3.5.2 Product convolution approximation . . . 108
3.6 Smoothing of SVM with reject option . . . 109
3.6.1 Smoothing of piecewise affine convex functions . . . 110
3.6.2 Smoothing the SVM with reject . . . 113

4 Conditional gradient algorithms for doubly non-smooth learning **117**
4.1 Introduction . . . 117
4.2 Smooth optimization with atomic-decomposition regularization . . . 118
4.2.1 Learning with atomic-decomposition norms . . . 118
4.2.2 Conditional gradient for smooth risk . . . 119

4.2.3 Extension to non-smooth empirical risk . . . 121
4.3 Motivating examples . . . 122
4.3.1 Collaborative filtering . . . 122
4.3.2 Multiclass learning . . . 122
4.4 Smoothed Conditional Gradient algorithms . . . 123
4.4.1 Smoothed Conditional Gradient algorithm . . . 124
4.4.2 Smoothed Composite Conditional Gradient Algorithm . . . 127
4.4.3 Smoothing the empirical risk - Application to the motivating examples . . . 130
4.4.4 Collaborative filtering . . . 131
4.4.5 Multiclass learning . . . 131
4.5 Experiments . . . 133
4.5.1 Implementation details . . . 134
4.5.2 Collaborative filtering . . . 136
4.5.3 Multi-class classification . . . 136
4.5.4 Competing approaches . . . 138
4.6 Conclusion . . . 141
4.7 Proofs . . . 144
4.8 Additional results . . . 148

5 Conclusion **153**
5.1 Summary of contributions . . . 153
5.2 Potential future research topics . . . 154

A Useful results **169**
A.1 Computing the top pair of singular vectors . . . 169
A.2 Projection on a norm-ball . . . 174
A.2.1 Examples of norm-balls . . . 174
A.2.2 Proximal operator of quadratic function . . . 175
A.3 Proofs of chapter 1 . . . 177
A.4 Computation of a support function without projection . . . 179

Résumé

La phase d'apprentissage des méthodes d'apprentissage statistique automatique correspond à la résolution d'un problème d'optimisation mathématique dont la fonction objectif se décompose en deux parties: a) le risque empirique, construit à partir d'une fonction de perte, dont la forme est déterminée par la métrique de performance et les hypothèses sur le bruit sur les données; b) la pénalité de régularisation, construite à partir d'une norme ou fonction jauge, dont la structure est déterminée par l'information à priori disponible sur le problème à résoudre.

Les fonctions de perte usuelles, comme la fonction de perte charnière pour la classification supervisée binaire, ainsi que les fonctions de perte plus avancées comme celle pour la classification supervisée avec possibilité d'abstention, sont non-différentiables. Les pénalités de régularisation comme la norme ℓ_1 (vectorielle), ainsi que la norme nucléaire (matricielle), sont également non-différentiables. Le but de cette thèse est d'étudier les problèmes d'apprentissage doublement non-différentiables (perte non-différentiable et régularisation non-différentiable), ainsi que les algorithmes d'optimisation numérique qui sont en mesure de bénéficier de cette structure composite.

Dans le premier chapitre, nous présentons une nouvelle famille de pénalités de régularisation, les normes de Schatten par blocs, qui généralisent les normes de Schatten classiques. Nous démontrons les principales propriétés des normes de Schatten par blocs en faisant appel à des outils d'analyse convexe et d'algèbre linéaire; nous retrouvons en particulier des propriétés caractérisant les normes proposées en termes d'enveloppes convexes. Nous discutons plusieurs applications potentielles de la norme nucléaire par blocs, pour le filtrage collaboratif, la compression de bases de données, et l'annotation multi-étiquettes d'images.

Dans le deuxième chapitre, nous présentons une synthèse de différentes techniques de lissage qui permettent d'utiliser pour le problème doublement non-lisse des algorithmes de premier ordre initialement adaptés aux objectifs composites avec fonction de perte différentiable. Nous montrons comment le lissage peut être utilisé pour lisser la fonction de perte correspondant à la précision au rang k, populaire pour

le classement et la classification supervises d'images. Nous décrivons dans les grandes lignes plusieurs familles d'algorithmes de premier ordre qui peuvent bénéficier du lissage: i) les algorithmes de gradient conditionnel; ii) les algorithmes de gradient proximal; iii) les algorithmes de gradient incrémental.

Dans le troisième chapitre, nous étudions plus en profondeur les algorithmes de gradient conditionnel pour les problèmes d'optimisation non-différentiables d'apprentissage statistique automatique. Nous montrons qu'une stratégie de lissage adaptatif associée à un algorithme de gradient conditionnel donne lieu à de nouveaux algorithmes de gradient conditionnel qui satisfont des garanties de convergence théoriques. Nous présentons des résultats expérimentaux prometteurs des problèmes de filtrage collaboratif pour la recommandation de films et de catégorisation d'images.

Mots clés : méthodes de prémier ordre, gradient conditionnel, lissage, norme nucléaire, apprentissage automatique, optimisation mathématique

Abstract

Training machine learning methods boils down to solving optimization problems whose objective functions often decomposes into two parts: a) the empirical risk, built upon the loss function, whose shape is determined by the performance metric and the noise assumptions; b) the regularization penalty, built upon a norm, or a gauge function, whose structure is determined by the prior information available for the problem at hand.

Common loss functions, such as the hinge loss for binary classification, or more advanced loss functions, such as the one arising in classification with reject option, are non-smooth. Sparse regularization penalties such as the (vector) ℓ_1-penalty, or the (matrix) nuclear-norm penalty, are also non-smooth. The goal of this thesis is to study doubly non-smooth learning problems (with non-smooth loss functions and non-smooth regularization penalties) and first-order optimization algorithms that leverage the composite structure of non-smooth objectives.

In the first chapter, we introduce new regularization penalties, called the group Schatten norms, to generalize the standard Schatten norms to block-structured matrices. We establish the main properties of the group Schatten norms using tools from convex analysis and linear algebra; we retrieve in particular some convex envelope properties. We discuss several potential applications of the group nuclear-norm, in collaborative filtering, database compression, multi-label image tagging.

In the second chapter, we present a survey of smoothing techniques that allow us to use first-order optimization algorithms originally designed for learning problems with nonsmooth loss. We also show how smoothing can be used on the loss function corresponding to the top-k accuracy, used for ranking and multi-class classification problems. We outline some first-order algorithms that can be used in combination with the smoothing technique: i) conditional gradient algorithms; ii) proximal gradient algorithms; iii) incremental gradient algorithms.

In the third chapter, we study further conditional gradient algorithms for solving doubly non-smooth

optimization problems. We show that an adaptive smoothing combined with the standard conditional gradient algorithm gives birth to new conditional gradient algorithms having the expected theoretical convergence guarantees. We present promising experimental results in collaborative filtering for movie recommendation and image categorization.

Keywords: first-order optimization, conditional gradient, smoothing, nuclear-norm, machine learning, mathematical optimization

Acknowledgments

The success of this work is due also to the people who shared their time with me during the last years.

A special thank goes to my supervisors for their attention and for the challenging research subject proposed to me, my scientific results are also the fruit of the exchanges with them; to my parents for moral support and good advises; to my friends around the world for the extra activities that we did together; to my colleagues for their patient explanations and challenging discussions; to all the other cool people that I met everywhere for their interesting exchanges.

Introduction

"Those who are enamored of practice without theory are like a pilot who goes into a ship without rudder or compass and never has any certainty where he is going. Practice should always be based on a sound knowledge of theory." - Leonardo da Vinci

This work is in the intersection of optimization and machine learning, introduced in the next two sections. The two fields have often different terminologies to indicate the same mathematical objects. The simplicity of the objective functions we use in our applications is justified by this way to approximate and re-define the real word applications.

In this work, we deal with convex objective functions, possibly nondifferentiable, and iterative algorithms with complexity bounds. We focus here on convex optimization algorithms. Indeed, we seek worst-case and finite-time theoretical guarantees of convergence in terms of objective evaluations. In convex optimization, these theoretical guarantees can be derived under verifiable conditions of the objective such as smoothness. In contrast, unless one makes stringent and often difficult to verify assumptions on the objective, worst-case theoretical guarantees non-convex optimization give at best a rate of convergence to a stationary point of the objective.

In the first chapter we introduce the state of art of convex optimization, statistical learning and first order algorithms to make this work self contained. We then present briefly our three contributions in this context. The next chapters are dedicated to each contribution. A conclusion section discusses some perspectives for future research. Finally the appendix contains a table of the notation we use and several additional results useful to understand the subject and the proofs. We chose to use the language of optimization, but to not to lose a part of the audience we indicate regularly the corresponding terms used in machine learning.

Notation

f, F	convex functions of the variables to optimize $\mathbf{x}$ and $\mathbf{y}$
L	Lipschitz constant of ∇f
μ	constant of strongly convexity
E	space where $\mathbf{x}$ is defined, we consider $\mathbb{R}^n$ or $\mathbb{R}^{d\times k}$
n	dimension of the space E. For matrices we use $n = dk$
$\mathbf{x}_t \in E$	iterates generated by an iterative algorithm,
	or in general, when the index is subscript, elements of a sequence
$\mathbf{x}^i \in \mathbb{R}$	the i-th entry of $\mathbf{x} \in E$, with $i = 1 \dots n$ also for matrices
$\mathbf{x}_\star$	optimal solution
f^*	Fenchel conjugate of f
$\mathcal{Q} \subset E$	subset for constrained optimization (often closed convex)
$i_{\mathcal{Q}}$	indicator function on the set $\mathcal{Q}$
τ_ε	number of iterations needed to satisfy a termination criterion
$\pi_{\mathcal{Q}}(\mathbf{x})$	projection of $\mathbf{x}$ onto a set $\mathcal{Q} \subset E$
$\mathrm{prox}_{\gamma g}(\mathbf{x})$	Moreau-type proximal operator of function g
$\|\mathbf{x}\|_p$	p-norm $\left(\sum_{i=1}^n (\mathbf{x}^i)^p\right)^{\frac{1}{p}}$, with $p \geq 1$
$\|\mathbf{x}\|_{\sigma,1}$	nuclear norm, i.e. sum of singular values of $\mathbf{x}$
$\|\mathbf{x}\|$	euclidean norm, if not specified differently
$\sigma_{\mathcal{Q}}$	support function of set $\mathcal{Q}$
$\mathbf{d} \in \mathcal{D}$	atoms
$c^i \in \mathbb{R}$	weights of atomic decomposition
$\sum_{i\in\mathcal{I}} c^i \mathbf{d}_i \in E$	atomic decomposition of a variable
$\|\cdot\|_{\mathcal{D}}$	atomic norm
$\mathbf{y} \mapsto \bar{\mathbf{x}}(\mathbf{y})$	linear minimization operator on $\mathcal{D}$
B_∞	ball of infinity norm $\{\mathbf{x} \in E \mid \|\mathbf{x}\|_\infty \leq 1\}$
B_p	ball of p-norm $\left\{\mathbf{x} \in E \,\middle|\, \|\mathbf{x}\|_p \leq 1\right\}$, with $p \geq 1$
$B_{\sigma,p}$	ball of Schatten p-norm, i.e of matrices whose singular values
	have bounded p-norm $\left\{\mathbf{x} \in \mathbb{R}^{d\times k} \,\middle|\, \|s(\mathbf{x})\|_p \leq 1\right\}$
$\mathbf{W}^{:i}$	column i of a matrix
$\mathbf{W}^{i:}$	row i of a matrix

Table 1: Summary table of notation, that is defined step by step in the text.

Chapter 1

First-order optimization for machine learning: context and contributions

Many branches of both pure and applied mathematics are in great need of computing instruments to break the present stalemate created by the failure of the purely analytical approach to nonlinear problems

John von Neumann

With this chapter we present an overview on concepts, theorems, algorithms, and examples, that are fundamental milestones to develop our thesis. Most of the definitions are spread in the text and are placed where they are used for the first time. The main references for this chapter that can be considered as milestones are Nesterov (2004) Bertsekas (2004) Hiriart-Urruty and Lemarechal (1996) Hiriart-Urruty and Lemarechal (1993) for the field of convex optimization, and Hastie et al. (2008); Huber (1981); Vapnik (2005) for the part of statistical learning.

Once the context of this thesis is thus settled, we finish this chapter with a summary of our main contributions, developed in the following chapters.

1.1 Elements of statistical learning

"A problem of finding the desired dependence using a limited number of observations". Vapnik (2005), about learning.

The principle of statistical learning is that there is a phenomenon of the real world that we want to

learn and generalize by studying a set of observations.

A learning algorithm takes as input a huge sample of observations and returns a model that describes the phenomenon and that is generalized also for non observed data. This process is completely automatic and defined by optimization algorithms. Let us take as instance one of our application: statistical image classification. The dataset is based on a collection of pictures and associated labels that describe their content. A vector $\mathbf{x}$ is extracted from one picture and represents some of its important characteristics, called features. The response $\mathbf{y}$ is the label associated to that picture. The challenge is to observe a lot of pictures for which the label is known and learn a function that predicts the associated label of any new picture. We need then to define another function, the empirical risk, that is big when a lot of prediction are wrong. The optimization algorithm is then used to minimize the empirical risk and obtain good predictions. The aim is to build a model capable of accurate predictions.

1.1.1 Predictor functions

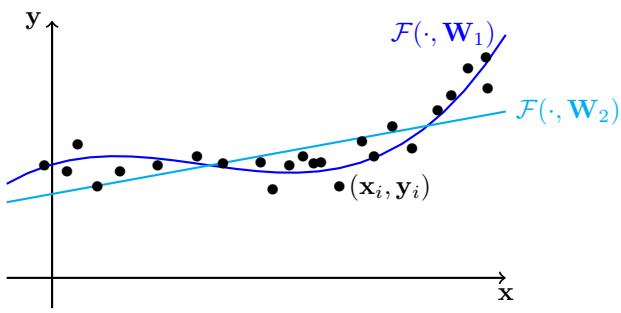

Figure 1.1: Two predictor functions $\mathcal{F}(\cdot, \mathbf{W}_1)$ and $\mathcal{F}(\cdot, \mathbf{W}_2)$ based on data $(\mathbf{x}_i, \mathbf{y}_i)$.

More formally, a feature vector $\mathbf{x}$ has distribution $F(\mathbf{x})$, and a response $\mathbf{y}$ has conditional distribution $F(\mathbf{y}|\mathbf{x})$. A *learning machine* implements a class of *predictor functions*

$$\mathcal{F} = \{\mathbf{x} \mapsto \mathcal{F}(\mathbf{x}, \mathbf{W}) \,|\, \mathbf{W} \in E\}\,, \tag{1.1}$$

where E is a set of parameters. When a particular $\mathbf{W}$ is chosen, the predicted response associated to each $\mathbf{x}$ is $\hat{\mathbf{y}} = \mathcal{F}(\mathbf{x}, \mathbf{W})$.

1.1.2 Find the model in theory

Again, the aim is to find the best function $\mathcal{F}$, i.e. the best $\mathbf{W}$, that can give good predictions on observations and generalizes also on non-observed data. The training task consists in selecting the best function, which is based on a *train set* of independent identically distributed (i.i.d.) observations

$$p_1 = (\mathbf{x}_1, \mathbf{y}_1), \ldots, p_N = (\mathbf{x}_N, \mathbf{y}_N) \tag{1.2}$$

drawn with the distribution $F(\mathbf{x}, \mathbf{y}) = F(\mathbf{x})F(\mathbf{y}|\mathbf{x})$. The only information available is in the train set, being the distribution $F(\mathbf{x}, \mathbf{y})$ unknown. To select the best function one define a loss $\ell(\mathbf{y}, \hat{\mathbf{y}})$. The worst the predictor function, the bigger the loss. We will see examples of loss later when we use them, starting at section 1.5.

Once the loss and the set of functions are chosen, the theoretical risk

$$R(\mathbf{W}) := \int \ell(\mathbf{y}, \mathcal{F}(\mathbf{x}, \mathbf{W}))dF(\mathbf{x}, \mathbf{y}), \tag{1.3}$$

has minimum at $\mathbf{W}_\star$, which defines the best function $\mathcal{F}(\,\cdot\,, \mathbf{W}_\star)$.

1.1.3 Find the model in practice

The previous result is only theoretical; we see here how to estimate the model using data.

To minimize the risk (1.3) on the basis of empirical data (1.2) one has to substitute the risk with the *empirical risk*

$$R_{\text{emp}}(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{y}_i, \mathcal{F}(\mathbf{x}_i, \mathbf{W})) \tag{1.4}$$

We minimize (1.4) rather than the abstract problem, obtaining a solution approximating $\mathbf{W}_\star$. This technique is called empirical risk minimization (ERM) inductive principle. We note that the loss function is referred to only one example and the empirical risk is the average of the loss on all the data.

Clearly the current ERM problem

$$\min_{\mathbf{W} \in E} R_{\text{emp}}(\mathbf{W}) \tag{1.5}$$

is *ill-posed*, i.e. small changes in the data could result in a big change of the solution, and then in the predictor function. More precisely, the optimization on new data $\mathbf{x}_i^{(\delta)}$ close to the previous one, where $\sum_{i=1}^{N} \left\|\mathbf{x}_i - \mathbf{x}_i^{(\delta)}\right\| \leq \delta$ is arbitrarily small, can cause a large deviation in the solutions $\mathbf{W}$ and $\mathbf{W}^{(\delta)}$, i.e. $\left\|\mathbf{W} - \mathbf{W}^{(\delta)}\right\|$ is big. By consequence there is a big difference also on the two predictors $\mathcal{F}(\cdot, \mathbf{W})$ and

$\mathcal{F}(\cdot, \mathbf{W}^{(\delta)})$.

1.1.4 Regularized problems

Now we want show how to avoid the issues of ill-posedness of the problem (1.5).

The structural risk minimization consists in the minimization of the empirical risk together with another term called regularization Vapnik and Chervonenkis (1974). This leads to look for solutions in a more restricted space.

Let us see in detail the two ways to restrict the solutions: the regularized problem

$$\min_{\mathbf{W} \in E} \Phi_\lambda(\mathbf{W}) := R_{\text{emp}}(\mathbf{W}) + \lambda \Omega(\mathbf{W}), \tag{1.6}$$

where Ω is called *regularizer*, and the constrained problem

$$\min_{\Omega(\mathbf{W}) \leq r} R_{\text{emp}}(\mathbf{W}), \tag{1.7}$$

for any λ, $r > 0$. A consequence of solving the regularized problem (1.6) and the constrained problem (1.7) is that the admissible predictor functions are more "simple" and have a particular structure. For instance, in our applications that we present at section 1.5, we are interested in predictor functions composed with linear applications defined by a low rank matrix.

The restriction to simple predictor functions can be done by taking for Ω an atomic norm, defined at section 1.2.3. Then the solution of (1.6) or (1.7) tends to be written as linear combination of atoms. For example, the optimization where Ω is the nuclear norm has tendency to give low rank solutions.

1.1.5 Parameter tuning

The capacity of a predictor function to generalize to non-observed data is related to the choice of λ and r. Each choice of λ or r defines a different optimization problem and return a different optimal solution, so they can be considered hyperparameters.

A technique to learn the best $\lambda_\star$ is to observe the empirical risk on an independent set called *validation set*. An algorithm is run on the train set on a grid of values for λ, then λ is chosen so that it minimizes the empirical risk on validation set. If the experiment is run correctly, the empirical risk on validation set decreases and then increase again, for increasing λ. The smallest λ corresponds to a model *overfitted* on the known data, where we can say also that there is no learning. Bigger λ refers to a better model, but when λ gets too large, the empirical risk gets bigger again, because the learned function is too simple,

e.g. it can be constant zero, and is unsuitable for prediction. On the other hand, when there is overfitting, the empirical risk on the train set will be small when λ is close to zero. A third independent *test set* can be used to observe the empirical risk related to $\lambda_\star$, but here no more tuning is allowed. The test set is used only to have a final check.

To learn $r_\star$, in the constraint case (1.7), there is the same behaviors and the same way to tune on validation set.

1.2 Elements of convex optimization

In the previous section we have seen why we minimize specific functions and what is their meaning from the statistical point of view. Now the interest is to see how to optimize these problems in an efficient way for large scale problems.

We start this section by introducing briefly several definitions, terms of language, theorems and algorithms that are a ground knowledge for this thesis. We introduce then a class of algorithms called first order algorithms, and we clarify the motivations of our study.

1.2.1 Basic elements for convex analysis

Our working space is called E, and is $\mathbb{R}$, $\mathbb{R}^n$ or $\mathbb{R}^{d\times k}$. Most of the results are valid for the tree sets; we will specify when we need one specific space, e.g. some matrix norms apply only to $E = \mathbb{R}^{d\times k}$.

A *closed convex function* f is such that (i) for all $t \in [0, 1]$ and for all $\mathbf{x}$, $\mathbf{y} \in \mathrm{dom}(f)$

$$f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$$

and (ii) f has a closed epigraph $\{(\mathbf{x}, r) \in E \times \mathbb{R} \,|\, r \geq f(\mathbf{x}),\, r \in \mathbb{R},\, \mathbf{x} \in \mathrm{dom}(f)\}$. The second condition is the closeness and avoids pathological behavior of f at the boundary of its domain. If not differently specified, in this chapter any function is closed convex. We highlight that we use the term "*convex function*" meaning "closed convex function".

In this context, a function is said L-*smooth* when it is differentiable and its gradient ∇f is Lipschitz with constant L with respect to a norm $\|\cdot\|$, i.e. for all $\mathbf{x}$, $\mathbf{y}$ in the domain of f

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_* \leq L\,\|\mathbf{x} - \mathbf{y}\|$$

where and $\|\cdot\|_*$ is the dual norm. When not specified $\|\cdot\|$ is the euclidean or the Frobenius norm. A

subgradient of f at $\mathbf{x}$ is a vector $\mathbf{s} \in E$ s.t. for all $\mathbf{y}$ in the domain of f

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{s}, \mathbf{y} - \mathbf{x} \rangle.$$

A function is said *strongly convex* with parameter $\mu > 0$ when

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{s}, \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|^2 \tag{1.8}$$

for any subgradient $\mathbf{s}$ of f at $\mathbf{x}$.

We assume a function $f\colon E \to \mathbb{R}$ minorized by an affine function. Then the *Fenchel conjugate* of a function f, also called convex conjugate, is

$$f^*(\mathbf{y}) := \max_{\mathbf{x} \in \mathrm{dom}(f)} \langle \mathbf{x}, \mathbf{y} \rangle - f(\mathbf{x}). \tag{1.9}$$

The obtained function f^* is convex. If the domain of f is nonempty, then f^* never takes the value $-\infty$. One important consequence of this definition is the Fenchel duality theorem: if the relative interiors of the domains of f and g have nonempty intersection, i.e. $\mathrm{ri}(\mathrm{dom}\, f) \cap \mathrm{ri}(\mathrm{dom}\, g) \neq \varnothing$, then

$$\inf_{\mathbf{x} \in \mathrm{dom}(f) \cap \mathrm{dom}(g)} f(\mathbf{x}) + g(\mathbf{x}) = -\min_{\mathbf{y} \in E} f^*(\mathbf{y}) + g^*(-\mathbf{y}), \tag{1.10}$$

supposing that the $\inf$ of the first part is finite. The Fenchel duality theorem is useful to prove the convergence of some algorithms, in particular the certificate, introduced in section 1.2.5.

The Fenchel conjugate solves also the problem of finding the inverse operator of the derivation. Suppose that the mapping $\mathbf{x} \mapsto \nabla f(\mathbf{x})$ is differentiable. Formally, the question is whether, given $\mathbf{s} \in E$, it is possible to find an $\mathbf{x} \in E$ such that $\mathbf{s} = \nabla f(\mathbf{x})$. The answer is that the map

$$\mathbf{s} \mapsto \mathbf{x} = \nabla h(\mathbf{s})$$

is the inverse operator of ∇f, where $h(\mathbf{s}) := f^*(\mathbf{s})$ is the Fenchel conjugate of f.

Let $\mathcal{Q}$ be a nonempty set in $\mathbb{R}^n$. The function $\sigma_{\mathcal{Q}}\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ defined by

$$\sigma_{\mathcal{Q}}(\mathbf{x}) := \sup_{\mathbf{y} \in \mathcal{Q}} \langle \mathbf{y}, \mathbf{x} \rangle$$

is called *support function* of $\mathcal{Q}$. A support function is finite everywhere iff $\mathcal{Q}$ is bounded.

Details on this argument can be found at (Hiriart-Urruty and Lemarechal, 1993, Chap. E).

1.2.2 Convex optimization problems

In this section we define optimization problems and underline that is worthless to look for exact solutions that need a huge computational effort.

We are interested in convex problems because they have the outstanding property of "local to global phenomenon". In fact (i) the subdifferential at one point contains global information of a linear lower bound on the whole function, and (ii) a local minimum is also global. In addition, (iii) a concept called lower bound, that we will introduce in section 1.2.5, is known only for convex problems. From here the functions we optimize are called F, f, and g.

Let us take the problem of minimizing a convex function F on (a subset of) E, written as

$$\mathcal{P}: \quad \min_{\mathbf{x}} F(\mathbf{x}), \tag{1.11}$$

were F is called the *objective*. When the optimization is on the whole space E, we call $\mathcal{P}$ an *unconstrained optimization problem*

$$\min_{\mathbf{x}\in E} F(\mathbf{x}). \tag{1.12}$$

When optimizing on a subset $\mathcal{Q} \subset E$, we call $\mathcal{P}$ a *constrained optimization problem*.

$$\min_{\mathbf{x}\in \mathcal{Q}} f(\mathbf{x}), \tag{1.13}$$

were f is convex.

Any constrained problem (1.13) can be rewritten in form of unconstrained optimization using the *indicator function*

$$i_{\mathcal{Q}}(\mathbf{x}) := \begin{cases} 0 & \mathbf{x} \in \mathcal{Q} \\ +\infty & \mathbf{x} \notin \mathcal{Q} \end{cases} \tag{1.14}$$

of the constraint set $\mathcal{Q}$. As the indicator function is an extended value function, $i_{\mathcal{Q}} : E \to \mathbb{R} \cup \{+\infty\}$, the new formulation

$$\min_{\mathbf{x}\in \mathcal{Q}} f(\mathbf{x}) = \min_{\mathbf{x}\in E} f(\mathbf{x}) + i_{\mathcal{Q}}(\mathbf{x})$$

of problem (1.13) is useful only for theoretical approaches. The new objective could lose some interesting properties after this transformation. For example, given f differentiable, the resulting $f + i_{\mathcal{Q}}$ is no more differentiable on the whole space. Therefore, it is worth to keep the constrained problem (1.13), as we see in Section 1.4.

Let us introduce other definitions.

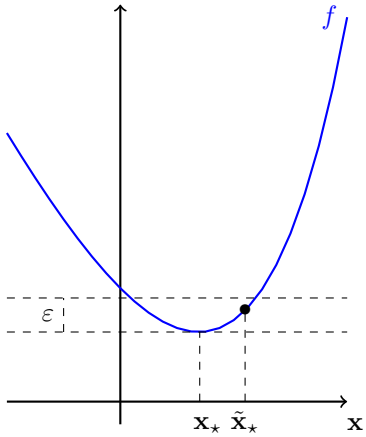

Figure 1.2: The optimal solution $\mathbf{x}_\star$ and an ε-optimal solution $\tilde{\mathbf{x}}_\star$ for the optimization of f.

Optimal Solution An *optimal solution* $\mathbf{x}_\star \in E$ for the problem (1.11) is a point of minimum of F, i.e. for all $\mathbf{x}$ in E (or in $\mathcal{Q}$ for constrained optimization)

$$F(\mathbf{x}_\star) \leq F(\mathbf{x}).$$

A necessary and sufficient optimality condition for $\mathbf{x}_\star$ to be an optimal solution of (1.11) is that zero is a subgradient of F at $\mathbf{x}_\star$

$$\mathbf{0} \in \partial F(\mathbf{x}_\star). \tag{1.15}$$

This implies that:

(i) When F is differentiable a necessary and sufficient optimality condition for the unconstrained problem (1.12) is $\nabla F(\mathbf{x}_\star) = \mathbf{0}$.

(ii) For the constraint optimization (1.13), a necessary and sufficient optimality condition is

$$\langle \mathbf{s}, \mathbf{x} - \mathbf{x}_\star \rangle \geq 0$$

for all $\mathbf{x}$ in $\mathcal{Q}$, for all subgradients $\mathbf{s}$ in $\partial f(\mathbf{x}_\star)$. The condition (1.15) is only a necessary condition if F is not convex.

ε-optimal solution As the joke at the beginning of this section suggests, to look for an optimal solution is worthless, computationally expensive, and often not possible to compute in practice, especially in learning tasks. We define then an approximate solution that is close to the optimal solution, but allows an

accuracy $\varepsilon > 0$ on the minimum. We note that in this context the solution $\mathbf{x}_\star$ is better when the accuracy ε is smaller. An ε*-optimal solution* is $\mathbf{x}_\star \in E$ such that, for all $\mathbf{x}$ in E

$$F(\mathbf{x}_\star) \leq F(\mathbf{x}) + \varepsilon.$$

For constrained optimization, the latter is for $\mathbf{x}$, $\mathbf{x}_\star \in \mathcal{Q}$.

The expression "*to solve a problem*" we mean to find an approximate ε-optimal solution.

1.2.3 Gauges and atomic norms

In this section we define an extension of the norm: the gauge, also called *Minkowski functional*. It will be used in some problems to define the objective.

Let $\mathcal{Q} \subset E$ be a set that contains the origin. The *gauge function* of $\mathcal{Q}$ is defined

$$\Omega_{\mathcal{Q}}(\mathbf{x}) := \inf \{t > 0 \,|\, \mathbf{x} \in t\mathcal{Q}\},$$

If no t satisfies the inequality, then $\Omega_{\mathcal{Q}}(\mathbf{x}) = +\infty$. Any norm is a gauge, but a gauge is not always a norm. In fact there is a counterexample when $\Omega_{\mathcal{Q}}(\mathbf{x}) \neq \Omega_{\mathcal{Q}}(-\mathbf{x})$, which happens in case $\mathcal{Q}$ is not symmetric to the origin.

Given a gauge Ω, a *dictionary*

$$\mathcal{D} = \{\mathbf{d}_i\}_{i \in \mathcal{I}}$$

is a family of elements in a Hilbert space $\mathcal{H}$, of unit gauge $\Omega(\mathbf{d}_i) = 1$, such that the linear span of $\mathcal{D}$ is dense in $\mathcal{H}$.

When $\mathcal{Q}$ is the convex envelope of a dictionary $\mathcal{D}$, then the gauge

$$\|\mathbf{x}\|_{\mathcal{D}} := \inf \{t > 0 \,|\, \mathbf{x} \in t\,\mathrm{co}(\mathcal{D})\} \tag{1.16}$$

is called *atomic norm* of the atom set $\mathcal{D}$, and the points $\mathbf{d}_i$ are called *atoms*, with $\|\mathbf{d}_i\|_{\mathcal{D}} = 1$.

We can think the atoms as the simplest elements in $\mathcal{H}$; e.g. the matrices of rank one, when $\|\cdot\|_{\mathcal{D}}$ is the nuclear norm, or the matrices with an entry equal to one and the others equal to zero, when $\|\cdot\|_{\mathcal{D}} = \|\cdot\|_1$.

1.2.4 Oracle

We approach now to the algorithmic part of this presentation and describe an object that is used at each iteration. In a general definition, an *oracle* is a unit that answers the questions of the algorithm. The

answer depends only on the information collected during the previous iterations.

Oracles can be classified by the order of derivatives. A zero order oracle returns the value of $F(\mathbf{x})$ given $\mathbf{x}$. A first order oracle returns the value of $F(\mathbf{x})$ and a subgradient $\mathbf{s} \in \partial F(\mathbf{x})$; if F is differentiable, we have also the gradient $\nabla F(\mathbf{x})$. If the function is twice differentiable a second order oracle is defined and it returns $F(\mathbf{x}), \nabla F(\mathbf{x})$ and also the hessian $\nabla^2 F(\mathbf{x})$. When dealing with large scale problems, a second order oracle is usually inefficient, because it can be expensive to compute and memory demanding. On the other hand, a first order oracle gives too few information and leads to a slow convergence.

We see now three operators that are used in combination with oracles inside the algorithms described in the following sections.

Projection operator We define the *projection operator* of a closed convex set $\mathcal{Q}$, which is just the projection of a point $\mathbf{x}$ onto $\mathcal{Q}$

$$\pi_{\mathcal{Q}}(\mathbf{x}) := \underset{\mathbf{y}\in\mathcal{Q}}{\operatorname{argmin}} \|\mathbf{y}-\mathbf{x}\|^2 . \tag{1.17}$$

It consist in finding the point $\mathbf{y}_\star$ in $\mathcal{Q}$ closest to $\mathbf{x}$. By the convexity of $\mathcal{Q}$ this point is unique.

Moreau-type proximal operator The *Moreau-type proximal operator*, or just prox, is a natural extension of the notion of projection operator. The prox operator of a convex function g with parameter γ is defined as

$$\operatorname{prox}_{\gamma g}(\mathbf{x}) := \underset{\mathbf{y}\in E}{\operatorname{argmin}}\ g(\mathbf{y}) + \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{y}\|^2 . \tag{1.18}$$

Let us see several examples of proximal operators for different choice of g.

Example 1. (i) Indicator function. $g(\mathbf{x}) = i_{\mathcal{Q}}(\mathbf{x})$, where $\mathcal{Q}$ is a closed convex set. Then the prox is the projection on $\mathcal{Q}$

$$\operatorname{prox}_{\gamma i_{\mathcal{Q}}}(\mathbf{x}) = \pi_{\mathcal{Q}}(\mathbf{x}). \tag{1.19}$$

Next examples lead to extensions of projection operator.

(ii) ℓ_1 norm. We have $g(\mathbf{x}) = \lambda \|\mathbf{x}\|_1$. Then the prox is the so-called the soft thresholding $\operatorname{prox}_{\gamma\|\cdot\|_1}(\mathbf{x}) = p^\gamma(\mathbf{x})$, where each entry i is

$$(p^\gamma(\mathbf{x}))^i = \begin{cases} x^i + \gamma & x^i \le -\gamma \\ 0 & -\gamma < x^i \le \gamma \\ x^i - \gamma & \gamma < x^i. \end{cases} \tag{1.20}$$

We plot this prox on Figure 1.3.

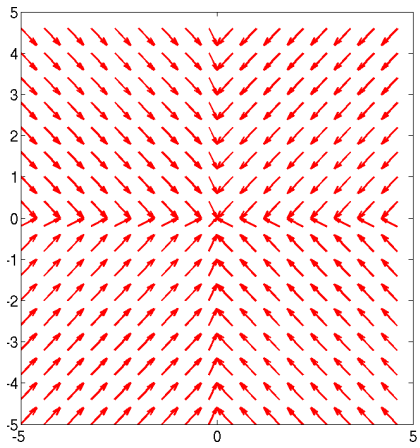

Figure 1.3: Prox of $g(\mathbf{x}) = \lambda \|\mathbf{x}\|_1$. Axis represent x^1, x^2. Each arrow represents $\mathrm{prox}_{\gamma g}(\mathbf{x})$ and is placed at $\mathbf{x}$.

(iii) Nuclear-norm regularization. $g(\mathbf{x}) = \|\mathbf{x}\|_{\sigma,1}$, with $\mathbf{x} \in \mathbb{R}^{d\times k}$. Using a singular value decomposition we rewrite $\mathbf{x} = \mathbf{U}\mathbf{s}\mathbf{V}^\top$. Then $\mathrm{prox}_{\gamma\|\cdot\|_{\sigma,1}}(\mathbf{x}) = \mathbf{U}p^\gamma(\mathbf{s})\mathbf{V}^\top$, where $p^\gamma(\mathbf{s})$ is the prox found in the previous example.

(iv) Zero. When $g = 0$ the prox is the identity function

$$\mathrm{prox}_0(\mathbf{x}) = \mathbf{x} \tag{1.21}$$

and the proximal gradient algorithm reduces to gradient descent, Algorithm 1. □

Proofs of these examples are in the appendix A.2.2. For a complete study of the properties of the proximal operator, we refer the interested reader to Combettes and Pesquet (2011).

Linear minimization operator The *Linear Minimization Operator* is a function that solves a linear sub-problem

$$\mathbf{y} \mapsto \bar{\mathbf{x}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{D}}{\mathrm{argmax}}\langle \mathbf{y}, \mathbf{x}\rangle. \tag{1.22}$$

and returns an atom of the dictionary.

Let us see several examples of linear minimization operator, by defining $\mathcal{Q}$ as the convex hull of $\mathcal{D}$.

Example 2. (i) The cone $\mathcal{Q} = \{\mathbf{x} \in \mathbb{R}^{n\times n} \,|\, \mathbf{x} \succeq \mathbf{0},\ \mathrm{tr}(\mathbf{x}) = 1\}$ of positive semidefinite matrices with trace equals one. This convex set, also called spectrahedron, is a natural generalization of the simplex Hazan (2008).

Then the linear minimization operator is the matrix of rank one which is given by the eigenvector

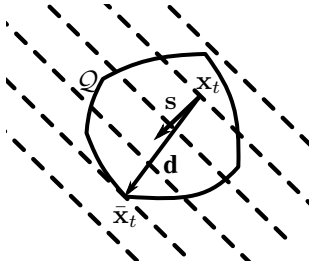

Figure 1.4: Linear minimization operator on a closed convex set $\mathcal{Q}$ with respect to the direction $\mathbf{s}$. $\mathbf{d} = LMO(\mathbf{x}_t)$. The dashed lines are orthogonal to $\mathbf{s}$. It is possible to see that the descent direction $\mathbf{d}$ is not aligned with $\mathbf{s}$, but is directed to a corner of $\mathcal{Q}$. This corner corresponds to the element of the dictionary that will be added to the sequence of atoms that compose the next iterate $\mathbf{x}_{t+1}$.

related to the largest eigenvalue $(\lambda, \mathbf{v}) = \text{eigs}_{\max}(\mathbf{s})$, then

$$\bar{\mathbf{x}}(\mathbf{s}) = \mathbf{v}\mathbf{v}^\top. \tag{1.23}$$

This linear minimization operator can be computed with Lanczos algorithm (Algorithm 16) presented at section A.1.

(ii) Maximization on the ball of nuclear norm $\mathcal{Q} = \left\{\mathbf{x} \in \mathbb{R}^{d\times k} \,\middle|\, \|\mathbf{x}\|_{\sigma,1} \leq 1\right\}$. Then the linear minimization operator is the matrix of rank one which is the given by the two singular vectors related to the largest singular value $(\mathbf{u}, d, \mathbf{v}) = \text{svd}_{\max}(\mathbf{s})$,

$$\bar{\mathbf{x}}(\mathbf{s}) = \mathbf{u}\mathbf{v}^\top. \tag{1.24}$$

We show how to approximate svd_{max} in section A.1. We obtained this result by adapting Lanczos algorithm Lanczos (1961) Cullum et al. (1983) that finds the largest eigenvalue and related eigenvector.

(iii) The ball of norm one $\mathcal{Q} = \left\{\mathbf{x} \in \mathbb{R}^n \,\middle|\, \sum_i \left|\mathbf{x}^i\right| \leq 1\right\}$. Then the linear minimization operator is $\bar{\mathbf{x}}(\mathbf{s}) = -\operatorname{sign}(\mathbf{s}_{i_\star})\mathbf{e}_{i_\star}$, where $i_\star = \operatorname{argmax}_{i=1\ldots n} |\mathbf{s}_i|$ and $\mathbf{e}_i$ are vectors of the standard base of $\mathbb{R}^n$. This works as well when $\mathbf{s}$ is a matrix.

(iv) The probability simplex $\mathcal{Q} = \left\{\mathbf{x} \in \mathbb{R}^n \,\middle|\, \mathbf{x}^i \geq 0, \sum_i \mathbf{x}^i = 1\right\}$. Then the linear minimization operator is

$$\bar{\mathbf{x}}(\mathbf{g}) = \mathbf{e}_{i_\star} \tag{1.25}$$

with

$$i_\star = \operatorname*{argmax}_{i=1\ldots n} \mathbf{x}^i$$

The computation of this operator has complexity $O(n)$. $\square$

Proof. (Of equation (1.25)) Here the set $\mathcal{Q}$ is the convex envelope of the set $\mathcal{D} = \{\mathbf{e}_1 \dots \mathbf{e}_n\}$ of the vectors of the standard base in $\mathbb{R}^n$. Then

$$\max_{\mathbf{y}\in\mathcal{Q}} \langle \mathbf{x}, \mathbf{y} \rangle = \max_{\mathbf{y}\in\mathcal{D}} \langle \mathbf{x}, \mathbf{y} \rangle = \max_{i=1\dots n} \langle \mathbf{x}, \mathbf{e}_i \rangle = \max_{i=1\dots n} \mathbf{x}^i.$$

We call $i_\star$ the index where the maximum of the last expression is attained, i.e.

$$i_\star = \operatorname*{argmax}_{i=1\dots n} \mathbf{x}^i$$

then the first expression has maximum at $\mathbf{e}_{i_\star}$.

$\square$

Sparsity The optimization with atomic norms leads to sparse representation of iterates, that are built with linear combinations on atoms. Sparsity does not mean only matrices or vectors with few nonzero entries. A variable is sparse when the atoms needed to form it are few. For example, if we consider the nuclear norm, a sparse matrix is just one that have rank much smaller than the maximum possible for a matrix of the same size, i.e. a sparse matrix is linear combination of few rank-one matrices.

To build sparse representation of complex signals we need dictionaries of vectors. These dictionary are bigger than bases, or better, they are overcomplete bases. Sparse representation in redundant dictionaries can improve pattern recognition, compression, and noise reduction. Processing is faster and simpler, and could be less memory demanding, in a sparse representation when few coefficients reveal the information we are looking for, i.e. the solution of an optimization problem Mallat (2009).

For instance, we will see that working with low rank matrices, instead to consider a full matrix $\mathbf{W}$, we work directly with its factorization $\mathbf{USV}^\top$.The matrix $\mathbf{S}$ is diagonal, $\mathbf{U}$ and $\mathbf{V}$ have few columns. The space needed to work with the decomposition is then orders of magnitude smaller than the full size of $\mathbf{W}$.

1.2.5 Optimization algorithms

We introduce in this section notions related to convergence of optimization algorithms, useful to compare algorithms and related "efficiency".

The *performance* of a numerical algorithm M on a problem P is the total amount of computational efforts required by M to solve P, where to solve a problem means to find an ε-optimal solution.

The notion of *complexity* helps to measure the performance and the computational effort of an algorithm. The *arithmetical complexity* measures the total number of arithmetical operations needed to reach a solution with accuracy ε. The *analytical complexity* counts just the calls of oracle to reach an accuracy ε. The study of analytical complexity is easier and enough informative to estimate the arithmetical complexity of the algorithm. One just needs to multiply the number of oracle calls to the cost of one iteration, which is the sum of the analytical complexity of the oracle and the cost of the other operations. Oracles and other operations, e.g. the projection onto a set and the minimization of a sub-problem, are interesting only when they are *easy to compute*. This also is an intuitive definition, and basically means that an algorithm is not usable in practice when inside each iteration there is an operation as expensive as the solution of the whole problem.

Convergence

We consider the problem 1.12. We say that the *sequence of iterates* $\{\mathbf{x}_t\}_{t\geq 1}$ generated by an algorithm, starting at $\mathbf{x}_0$, *converges* to a point $\mathbf{x}_\star$ when for any accuracy $\varepsilon > 0$ exists a number T of iterations s.t.

$$|F(\mathbf{x}_t) - F(\mathbf{x}_\star)| \leq \varepsilon$$

for all $t \geq T$. We define the *class* $\mathcal{F}$ of functions that share some properties, e.g. the class of L-smooth functions and the class of μ-strongly convex functions.

Upper bounds

Given a class $\mathcal{F}$ of problems, an oracle $\mathcal{O}$, and an algorithm that generates a sequence of iterates $\mathbf{x}_t$, the *upper bound* is a sequence $\{u_t\}_t$ such that for all objectives $F \in \mathcal{F}$ it can be proved that

$$\left(\min_{i=1\ldots t} F(\mathbf{x}_i)\right) - F(\mathbf{x}_\star) \leq u_t,$$

where the comparison is between the optimal $F(\mathbf{x}_\star)$ and the smallest objective in the previous iterations. It could be used just $F(\mathbf{x}_t)$ if we consider descent algorithms, but this definition is also valid for those algorithms that sometimes can ascend, e.g. projected gradient descent. The sequence u_t is also called *rate of convergence* of an algorithm for a given class of functions.

Lower bounds

The upper bound is informative and can be used to compare the performance of two algorithms, for example, we say that an algorithm with upper bound of K/t^2 is faster and more efficient than another one with rate of H/t, with H, $K > 0$. Nevertheless, there are two big issues:

(i) The upper bound is related to a chosen algorithm;

(ii) The proof of the upper bound could be too rough.

In the first case (i), a better convergence for that class could be reached just changing the algorithm; in the second case (ii), it could exist a proof of convergence, of the same algorithm, that gives a faster upper bound, e.g. one has proved that for one algorithm u_t is $O(1/t)$, but there exists a proof showing that u_t is $O(1/t^2)$.

Therefore we need to define a new bound that describes the difficulty of a class of functions, whatever the algorithm is.

Given a class $\mathcal{F}$, the *lower bound* is a sequence ℓ_t such that there exist a function $F \in \mathcal{F}$, the "worst" function in the class, that gives

$$\ell_t \leq \min_{i=1\ldots t} F(\mathbf{x}_i) - F(\mathbf{x}_\star),$$

for any first order oracle that gives answers based on iterates $\mathbf{x}_0, \ldots, \mathbf{x}_t$ and (sub)gradients $\mathbf{s}_0, \ldots \mathbf{s}_t$ collected until time t. The concept of lower bound appeared first in Nemirovski and Yudin (1983). Also compare Nesterov (2004). The lower bounds are independent from the oracle or from algorithm and are studied to know if an algorithm can be improved for a given problem.

A *first order black-box algorithm* is a mapping from the previous iterates $\mathbf{x}_0, \ldots, \mathbf{x}_t$ and relative (sub)gradients $\mathbf{s}_0, \ldots, \mathbf{s}_t$ to the new iterate $\mathbf{x}_{t+1}$.

We see now lower bounds for problems that we will solve with black-box algorithms in the next sections. We consider the problem 1.12 and we suppose that

$$\mathbf{x}_0 = 0 \tag{1.26}$$

$$\mathbf{x}_{t+1} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_t\}, \text{ for any } t \geq 0 \tag{1.27}$$

Then

- there exists a Lipschitz objective F s.t. $\ell_t = O(1/\sqrt{t})$;
- there exists an L-smooth objective F, s.t. $\ell_t = O(1/t^2)$;

- there exists an L-smooth and μ-strongly convex objective F, s.t. $\ell_t = O(k^{-t})$,

where in the third case

$$k = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$

Nemirovski and Yudin (1983)Nesterov (2004). This means that no first order algorithm that respects hypothesis (1.26) and (1.27) can optimize the worst objective of $\mathcal{F}$ faster than ℓ_t.

Given an objective function the lower bound is obtained theoretically, without the need to propose an algorithm. In fact, the lower bound can be found directly by defining the most difficult objective in the class, or using a resisting oracle, i.e. an oracle that creates the worst problem for each concrete algorithm Nesterov (2005). It is needed to argue that if the number of queries to the oracle is too small, then we do not have enough information about the function to identify an ε-optimal solution.

If the lower bound is proportional to the upper bound of an algorithm, then that algorithm is called *optimal*.

Certificate

We have seen at sections 1.2.5 and 1.2.5 the complexity bounds as theoretical information that can be used to compare algorithms, but complexity bounds are not observable, i.e. they depend on some constant that can be determined only with the knowledge of the optimal solution. A typical example of these constants is the distance between initial iterate and optimal solution. So, even if lower or upper bound are known for an algorithm, these bounds remain theoretical and cannot be used to implement algorithms. In addition, as complexity bounds are related to the hardest problem in the class, it can happen that the objective function is simpler and there is no need to run the algorithm for many iterations to obtain an ε-optimal solution. We introduce now a more practical concept that is used to decide if more iterations are needed.

We see now a third bound that is explicit and can be used in practice in the implementation: the certificate. A *certificate* of an algorithm is a sequence G_t that bounds from above the approximation quality, i.e.

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq G_t, \tag{1.28}$$

where G_t depends only on the previous iterates $\mathbf{x}_0, \ldots, \mathbf{x}_t$. The certificate should not be confused with the upper bound, the difference between them is strong: here the last iterate $\mathbf{x}_t$ is no more compared theoretically with the optimal solution, but can be evaluated using only the iterates $\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}$.

A certificate is interesting when it is proportional to the upper bound. In this case the approximation quality can be controlled by the certificate. Otherwise, if the certificate converges slower than the upper

bound, it is not worth to use it. Of course, if a certificate for an algorithm is faster than the proved upper bound, then the certificate becomes a more efficient new upper bound.

The certificate is an observable criteria, and it has be easy to compute in order to be used as termination criterion, as it must be checked at each iteration. We will see in section 1.3.5 an algorithm that includes the certificate as "free" product of its iterations. This means that the only operations used to find a solution are enough to make available a certificate, without additional computation.

Complexity bounds

Instead of focusing on the accuracy as function of iterations, we can reverse the point of view and focus on the number of iterations as a function of the accuracy. The *complexity bound* is the number of iterations needed to have an accuracy of ε for the worst objective in the class. If we show that we need at most $\bar{T}$ iterations of the algorithm to achieve an accuracy ε, then $\bar{T}$ is called *upper complexity bound*. If we show we need at least $\underline{T}$ iterations then $\underline{T}$ is called *lower complexity bound*. For example, in one algorithm with convergence rate

$$\frac{c_1}{t} \leq F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{c_2}{\sqrt{t}},$$

upper complexity bound is $\bar{t} = (c_2/\varepsilon)^2$ (since $c_2/\sqrt{t} \leq \varepsilon$ implies $(c_2/\varepsilon)^2 \leq t$), and the lower complexity bound is $\underline{T} = c_1/\varepsilon$.

Scalability

The next definition is intuitive. We say that an algorithm is more *scalable* than another one, with respect to the size n of the problem, when the cost of its oracle and the rate of convergence depends in a minor way on n. For example, an algorithm with convergence rate $O(n/t^2)$ is more scalable than another one with rate $O(n^3/t^2)$. The best would be to have algorithm and oracle which do not depend at all on the size n. We are then interested in iterative algorithms that use first order oracles. These algorithms make a trade off between high scalability with respect to n and efficient rate of convergence with respect to the number of iterations t.

Termination criterion

We have seen at Section 1.2.5 the complexity bounds as theoretical information that can be used to compare algorithms, but complexity bounds are not observable, i.e. they depend on some constant that can be determined only with the knowledge of the optimal solution. In addition, as complexity bounds are related to the hardest problem in the class, but it can happen that the objective function is more simple

and there is no need to run the algorithm for many iterations to obtain an ε-optimal solution. We introduce now a more practical concept that is used to decide if more iterations are needed.

A *termination criterion* is a decision rule used to determine when to stop the iterative algorithm. The choice of a good termination criterion is important, because if the criterion is too weak the solution obtained may be useless; if the criterion is strict, it may lead to an algorithm that never stops or that may have a huge computational cost. Examples of simple termination criteria used in practice are (i) to fix a maximum number of iterations or (ii) to fix a maximum amount of computing time. These criteria are simple and give no information about the quality of the solution. We present now a criterion that is more reliable: (iii) the certificate.

We highlight that using a stopping criterion there is no more theoretical interest in the worst function of the class; an algorithm stops when the criterion is satisfied for the current objective function.

A termination criterion based on the certificate (1.28) takes into account that when the next condition is satisfied

$$G_t \leq \varepsilon$$

the algorithm has reached an ε-optimal solution $\mathbf{x}_t$.

An important aspect in this thesis is that the algorithms are validated by upper bounds, lower bounds and convergence proofs, rather than excellent experimental tests run on machines. We run also tests, which are necessary to give evidence for the statistical learning tasks, that are based on observations of data. But these tests are not meant to validate the optimization.

1.3 First order algorithms for convex optimization - unconstrained case

"Nothing is more practical than a good algorithm."

(Good old principle)

We have seen in the previous sections a general description of what is optimization and how to obtain problems to solve which are meaningful from the statistical point of view. In these two sections we make an overview of several fundamental algorithms, that solve convex problems, and that can handle in particular the large scale dimension of a problem. We start with the algorithms that minimize a function on the whole space E, then we add some constraints and present the corresponding algorithms.

Let us suppose we want to optimize an objective F that belongs to a class $\mathcal{F}$ of functions. A *first order algorithm* that optimizes an objective F on a closed convex $\mathcal{Q}$, or on the whole space E, is an algorithm

that knows in advance $\mathcal{Q}$ and the class $\mathcal{F}$, but does not know what exactly F is. A first order algorithm discovers F step by step through the first order oracle and finds the iterates $\mathbf{x}_t$ with a rule. The output of the rule at the step t is based only on the structure of $\mathcal{Q}$ and on the first order information related to previous iterates $\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}$ provided by the oracle. So the rule is said to be *non-anticipating* Juditsky and Nemirovski (2010).

In addition, also the termination criterion is non-anticipating and depends only on the information collected during the previous iterations.

The increasing interest for *large scale problems* makes first order algorithms more interesting than second order algorithms. The latter are more efficient in terms of (few) calls to the oracle, but each call to the oracle can become suddenly expensive when the size of the problem increases, both in terms of computations and in terms of memory. For example, if the optimization is in $\mathbb{R}^n$, the gradients of the objective have size n, while the Hessian has dimension n^2. Taking a quite small $n = 10^6$, the gradient needs 4 Mega byte of memory and the Hessian 4 Tera byte. Sure, may be there is a way to avoid to store all this matrix, but second order algorithms need at least to compute them.

Let us summarize some good properties of first order algorithms with hypothesis 1.26 and 1.27, applied to large scale problems:

i) Cheap iterations, in general, when the constraint are simple, but this depends also on the cost of the oracle. One iteration costs $O(n)$, where n is the dimension of the space.

ii) The rate of convergence is (almost) independent on the dimension of the problem, when the geometry of the problem is favorable.

iii) Adapt to find medium accuracy solutions for large scale convex problems.

But there are also bad news:

I) The rate of convergence is sublinear, can be linear only with strongly convexity of the objective.

II) The performance of a first order algorithm is strictly related to the Lipschitz constant L and the size of $\mathcal{Q}$, e.g. its diameter.

III) first order algorithms cannot find high accuracy solutions in a reasonable time.

In the next sections we present the main types of first order algorithm, each type is shown through a precise algorithm.

1.3.1 Gradient descent algorithm

In unconstrained minimization the optimization is done in the whole space E. The simplest scheme for unconstrained minimization is the gradient algorithm, also called steepest descent algorithm. Given a differentiable F, Algorithm 1 solves the problem

$$\min_{\mathbf{x}\in E} F(\mathbf{x}).$$

A *descent direction* at $\mathbf{x}$ is a vector $\mathbf{d}$ such that $\langle \nabla F(\mathbf{x}), \mathbf{d}\rangle < 0$. In this algorithm the descent direction is the direction of the fastest local decrease of F at $\bar{\mathbf{x}}$ is the antigradient, i.e. the direction $\mathbf{d} = -\nabla F(\bar{\mathbf{x}})$ opposite to the gradient.

Algorithm 1 Gradient algorithm

Choose $\mathbf{x}_0 \in E$
for $t = 0, 1, 2 \ldots$ **do**
  $\mathbf{x}_{t+1} = \mathbf{x}_t - h_t \nabla F(\mathbf{x}_t)$
end for

The sequence h_t is called *step size*, (or learning rate, in statistical learning) and can be chosen in different ways. The step size can be chosen in advance, constant or with a decreasing sequence; or it can be found during the iterations through a *line search*. The line search in a direction $\mathbf{d}$ from a (fixed) point $\mathbf{x}_t$ consist in finding an $h > 0$ such that $F(\mathbf{x}_t + h\mathbf{d})$ has a "sufficient" decrease. The exact line search that minimizes $F(\mathbf{x}_t + h\mathbf{d})$ is purely theoretical and can be computed in finite time only when the structure of the objective makes possible to find an exact formula.

An inexact line search solves a sub problem using a termination criterion, as for instance, the Wolfe rule, or the Goldstein-Armijo rule, when the gradient is too expensive to compute. This strategy is used in the majority of the practical algorithms. The gradient descent can be used for any differentiable function, but without any stronger assumption we can only say that it converges to a stationary point. We do not know even if it is a local minimum. If the function is convex the results are more interesting.

For an L-smooth function F and a fixed stepsize $h = 1/L$, the upper bound is

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{2L \left\|\mathbf{x}_0 - \mathbf{x}_\star\right\|^2}{t+4} \tag{1.29}$$

The (1.29) is called *sublinear rate* of convergence because it is slower than linear rate, i.e. when the upper bound is exponential. In terms of complexity using gradient descent, each new right digit of the solution demands an amount of computations comparable with the total amount of the previous work.

In addition this algorithm is not optimal for smooth functions, in fact the lower bound for the class of smooth functions is $O(1/t^2)$, as seen in sec 1.2.5.

Gradient descent is much more efficient when F is strongly convex, and the upper bound is

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{L}{2}\left(\frac{L-\mu}{L+\mu}\right)^{2t} \|\mathbf{x}_0 - \mathbf{x}_\star\|^2 \tag{1.30}$$

with the optimal stepsize $h = 2/(\mu + L)$. The (1.30) is called *linear rate* of convergence and is fast: each new right digit of the solution demands a constant amount of computation. Nevertheless, this amount of computation is related to the cost of the oracle, and it is worth to use gradient descend combined with a cheap and scalable oracle. More details and proofs at chapter 4 of Nesterov (2004).

In addition we notice that in both cases the convergence of gradient descent depends on the distance from the solution of the start point, but not on the dimension of the problem. Then gradient descent is scalable with respect to the size of the problem.

1.3.2 Proximal gradient algorithm

The previous algorithm is used to optimize a function considered as a black box. Now let us start to consider also the structure of the objective, that restricts the class of functions to optimize, but allows to develop better algorithms.

Suppose we want to optimize a convex function that is a sum or a composition of more simple functions. To prove that it is convex we need to analyze its components, it is impossible to prove convexity numerically. Then the objective is no more a black box, but we can say something about its structure and use it for an algorithm.

The problem to solve is

$$(\mathcal{P}) \quad \min_{\mathbf{x}\in E} F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) \tag{1.31}$$

where f is L-smooth and g is convex, possibly non differentiable.

This decomposition is motivated for applications in statistical learning, where f is the empirical risk and g is a regularization term, and for applications in signal processing for compression, denoising and pattern recognition. W describe the problem at section 1.3.6.

Proximal gradient is a first order algorithm, in which the most expensive operation in each iteration is the evaluation of the proximal operator (1.18). Then the scalability of the algorithm depends on the type of the function g. When g has a cheap proximal operator, proximal gradient algorithms are scalable.

As instance of proximal gradient, let us see the iterative shrinkage-thresholding algorithm (ISTA) of

Beck and Teboulle (2009).

Algorithm 2 ISTA

Choose $\mathbf{x}_0 \in E$
for $t = 1, 2, \dots$ **do**
  $\mathbf{x}_t = \operatorname{prox}_{\frac{1}{L}g}(\mathbf{x}_{t-1} - \frac{1}{L}\nabla f(\mathbf{x}_{t-1}))$
end for

The upper bound for ISTA is

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{L \left\|\mathbf{x}_0 - \mathbf{x}_\star\right\|^2}{2t}. \tag{1.32}$$

For the proof of convergence see Theorem 3.1 in Beck and Teboulle (2009). This algorithm is not optimal. In fact, the lower bound of the composite problem (1.31) is $O(1/t^2)$, as shown in Nemirovski and Yudin (1983). We will see in section 1.3.3 an optimal algorithm for the problem (1.31).

For more about this algorithm, that belongs to the forward-backward iterative scheme, we refer to Passty (1979) Bruck (1977) Combettes and Wajs (2005).

Certificate for composite problems Now we study a certificate for composite problems of type (1.31). The dual problem of $(\mathcal{P})$ is

$$(\mathcal{D}) \quad \max_{\mathbf{y} \in E} -f^*(\mathbf{y}) - g^*(\mathbf{y}),$$

where f^* and g^* are the Fenchel conjugates of respectively f and g, defined at (1.9).

A certificate for the problem $(\mathcal{P})$ is the sequence

$$G_t = \langle \mathbf{x}_t, \nabla f(\mathbf{x}_t) \rangle + g(\mathbf{x}_t) + g^*(-\nabla f(\mathbf{x}_t)). \tag{1.33}$$

This certificate is computationally useful when g and its conjugate g^* are easy to compute, and does not need to compute f^*, that in practice could be expensive. So the gap G_t is a certificate because we can calculate it with the information collected until the step t, we just need as hypothesis that we know an explicit form of g^*. For example, if g is a norm, then g^* is the indicator function of the polar of the unit ball of g.

Proof. (Of equation (1.33)) By Fenchel duality theorem (1.10) the minimum of $(\mathcal{P})$ is equal to the maxi-

mum of $(\mathcal{D})$. With this result we bound the optimal value $F(\mathbf{x}_\star) = f(\mathbf{x}_\star) + g(\mathbf{x}_\star)$

$$
\begin{aligned}
& -f^*(\mathbf{y}_t) - g^*(-\mathbf{y}_t) \leq f(\mathbf{x}_\star) + g(\mathbf{x}_\star) \leq f(\mathbf{x}_t) + g(\mathbf{x}_t) \\
\iff & f^*(\mathbf{y}_t) + g^*(-\mathbf{y}_t) \geq -f(\mathbf{x}_\star) - g(\mathbf{x}_\star) \geq -f(\mathbf{x}_t) - g(\mathbf{x}_t) \\
\iff & f(\mathbf{x}_t) + g(\mathbf{x}_t) + f^*(\mathbf{y}_t) + g^*(-\mathbf{y}_t) \geq +f(\mathbf{x}_t) + g(\mathbf{x}_t) - f(\mathbf{x}_\star) - g(\mathbf{x}_\star) \geq 0 \\
\iff & f(\mathbf{x}_t) + f^*(\mathbf{y}_t) + g(\mathbf{x}_t) + g^*(-\mathbf{y}_t) \geq F(\mathbf{x}_t) - F(\mathbf{x}_\star) \geq 0.
\end{aligned}
$$

As this result is valid for all $\mathbf{x}_t, \mathbf{y}_t$, if we choose $\mathbf{y}_t = \nabla f(\mathbf{x}_t)$, then by Fenchel equality

$$f(\mathbf{x}_t) + f^*(\mathbf{y}_t) = \langle \mathbf{x}_t, \nabla f(\mathbf{x}_t) \rangle$$

we can eliminate f^* and deduce

$$\langle \mathbf{x}_t, \nabla f(\mathbf{x}_t) \rangle + g(\mathbf{x}_t) + g^*(-\nabla f(\mathbf{x}_t)) \geq F(\mathbf{x}_t) - F(\mathbf{x}_\star) \geq 0. \qquad \square$$

For instance, for the problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + \|\mathbf{x}\|_1 ,$$

the certificate found in this way is

$$G_t = \langle \mathbf{x}_t, \nabla f(\mathbf{x}_t) \rangle + \|\mathbf{x}_t\|_1 + i_{\{\mathbf{y} \,|\, \max_i |\mathbf{y}_i| \leq 1\}}(\nabla f(\mathbf{x}_t))$$

We observe that this certificate is not useful because it could be infinity even if the found solution is ε-optimal.

1.3.3 Accelerated proximal gradient algorithm

This algorithm is a faster version of proximal gradient, presented first in Nesterov (1983), and then in two works Nesterov (2007a) and Beck and Teboulle (2009). Known also as Nesterov's accelerated proximal gradient, it can be seen as a general algorithm of which the accelerated projected gradient and accelerated gradient algorithms are particular cases.

First we define a *fast iterative algorithm* as in definition 3.1 of Beck and Teboulle (2012). Given the problem

$$\min_{\mathbf{x} \in E} F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}),$$

where f is L-smooth and g is subdifferentiable convex, an iterative algorithm is "fast" with constant $0 < k < \infty$, if it generates a sequence $\{\mathbf{x}_t\}$ s.t.

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{Lk}{t^2}.$$

The constant k possibly depends on $\mathbf{x}_0$ and $\mathbf{x}_\star$.

As instance, we show the fast iterative shrinkage-thresholding algorithm (FISTA), described by Beck and Teboulle (2009).

Algorithm 3 FISTA

Choose $\mathbf{x}_0 \in E$
$\mathbf{y}_1 = \mathbf{x}_0$
$\alpha_1 = 1$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{x}_t = \mathrm{prox}_{\frac{1}{L}g}(\mathbf{y} - \frac{1}{L}\nabla f(\mathbf{y}))$
  $\alpha_{t+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4\alpha_t^2}\right)$
  $\mathbf{y}_{t+1} = \mathbf{x}_t + \left(\frac{\alpha_t - 1}{\alpha_{t+1}}\right)(\mathbf{x}_t - \mathbf{x}_{t-1})$
end for

The convergence of FISTA is

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{2L\left\|\mathbf{x}_0 - \mathbf{x}_\star\right\|^2}{(t+1)^2}. \tag{1.34}$$

Note the difference between (1.32), which is $O(1/t)$, and (1.34), which is $O(1/t^2)$. In fact the latter is much more efficient from the computational point of view.

Recent works There are several recent works on (accelerated) proximal gradient. The evaluation of the proximity operator may generate approximation errors. In Schmidt et al. (2011) it is proposed an analysis of the convergence of (accelerated) proximal gradient algorithms that includes the error that is generated. As main result the convergence rate with errors is shown equal to the exact version, i.e. when the proximal operator is evaluated analytically. An overview of (accelerated) gradient algorithms is given in Nesterov (2013), where the class of problems is restricted to the minimization of a smooth function plus a nonsmooth convex regularizer. In fact, for general nonsmooth and nonconvex objectives, it is proved that it is NP-hard to know whether it exists a descent direction given a point. This article shows that convex and nonconvex cases can be solved with composite gradient mapping and have the same complexity of minimizing only the differentiable function. Efficient line search methods are shown for all algorithms.

1.3.4 Accelerated gradient descent algorithm

This algorithm is an accelerated version of gradient descent at section 1.3.1, i.e. its rate of convergence is $O(1/t^2)$ instead of $O(1/t)$.

The problem to solve is

$$\min_{\mathbf{x}\in E} F(\mathbf{x})$$

where F is L-smooth. We see that this problem is a particular case of (1.36).

If the problem is rewritten as

$$\min_{\mathbf{x}\in E} F(\mathbf{x}) + 0,$$

we see that the assumptions to use accelerated proximal gradient at section 1.3.3 are verified. In particular, the proximal operator of $g = 0$ is the identity function, as seen at equation (1.21).

Then the accelerated gradient algorithm is a particular case of accelerated proximal gradient algorithms. The convergence of accelerated gradient is

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{2L \left\|\mathbf{x}_0 - \mathbf{x}_\star\right\|^2}{(t+1)^2}.$$

Despite accelerated proximal gradient and accelerated gradient share the same convergence rate, the latter has a much cheaper oracle. On the other hand, accelerated proximal gradient could be more useful for statistical learning because it allows to solve problems with a regularization g that possibly is not differentiable.

1.3.5 Composite conditional gradient algorithm

The composite conditional gradient algorithm is a regularized version of the conditional gradient algorithm Dudik et al. (2012); Harchaoui et al. (2012b) that we will present in Section 1.4.3.

The regularized problem to solve is

$$\min_{\mathbf{x}\in E} F(\mathbf{x}) := f(\mathbf{x}) + \lambda \left\|\mathbf{x}\right\|_{\mathcal{D}}, \tag{1.35}$$

where f is L-smooth, $\|\cdot\|_{\mathcal{D}}$ is the atomic norm defined at (1.16) on the set of atoms $\mathcal{D}$, the dictionary. Despite this is the same problem as (1.31), here we highlight that the regularization is an atomic norm. In fact, composite conditional gradient uses the structure of $\mathcal{D}$ to retrieve a sparse solution. As this algorithm finds at each iteration a new atom in $\mathcal{D}$ through a step, called *greedy step* Temlyakov (2012), composite conditional gradient is a *greedy algorithm*. A greedy step maximizes a function determined by

the information from the previous steps of the algorithm.

For composite conditional gradient the greedy step consists in the computation of the linear minimization operator (1.22) that returns atoms $\mathbf{d}_t$. A linear combination of them will form the solution of the problem. To have scalability of composite conditional gradient we assume that the linear minimization operator can be solved in time polynomial in n. We will discuss this point at the end of this section.

Algorithm 4 Composite Conditional Gradient

Choose $\mathbf{x}_0 \in E$
for $t = 1, 2 \ldots$ **do**
  $\bar{\mathbf{x}}_t = \operatorname{argmin}_{\mathbf{x} \in \mathcal{Q}} \langle \nabla f(\mathbf{x}_{t-1}), \mathbf{x} - \mathbf{x}_{t-1} \rangle$
  $\mathbf{x}_t = \bar{\mathbf{x}}_t$
  $\mathbf{c}_\star = \operatorname{argmin}_{\mathbf{c} \in \mathbb{R}^t,\, \mathbf{c} \geq \mathbf{0}} f(\sum_{\tau=1}^t c^\tau \mathbf{x}_\tau) + \lambda \sum_{\tau=1}^t c^\tau$
  Update $\mathbf{x}_t = \sum_{\tau=1}^t c_\star^\tau \mathbf{x}_\tau$
end for

The last line of the algorithm that updates the iterate $\mathbf{x}_t$ is optional and just shows a formula to get $\mathbf{x}_t$. In practice and normally, it is possible and suitable to code the composite conditional gradient using all the time the atomic decomposition of the iterate.

From Theorem 3 at Harchaoui et al. (2012b), we report the upper bound of composite conditional gradient

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{8LD^2}{t+14} \qquad t \geq 2,$$

where D is the problem-domain parameter in the norm $\|\cdot\|_{\mathcal{D}}$, defined at Harchaoui et al. (2012b).

We see then that the composite conditional gradient algorithm is not optimal in the usual information-based complexity framework.

Discussion When the set $\mathcal{Q} = \mathrm{co}(\mathcal{D})$ is a polyhedron, conditional gradient is slow, because the direction of descent is often towards a vertex, and by consequence can be orthogonal to the direction to reach the optimal solution Bertsekas (2004). Conditional gradient is recommended when the accuracy of the solution is not very relevant. In fact, for large scale problems, we are interested in a large accuracy; conditional gradient is enough to get a bit closer to the solution in a reasonable time.

The most important property of conditional gradient is that it produces sparse iterates. The greedy step returns atoms, that are outputs of the linear minimization operator composed with a first order oracle. The iterates are linear combinations of these atoms and the solution also. Sparse variables are more tractable from the computational point of view because they are controlled by few coefficients, they need less memory. Then the computational effort is reduced. It is strongly recommended to use algorithms that handle sparse representation for large scale optimization.

To solve problem (1.35) one could choose proximal gradient algorithm, but in some cases composite conditional gradient is more useful. We compare the two algorithms from the practical point of view.

We saw that proximal gradient has analytical complexity of $O(1/t^2)$, whether composite conditional gradient has only $O(1/t)$. Then proximal gradient seems to be much more efficient. But if we think in this way, we forget that the analytical complexity just measures the calls to an oracle, without taking into account whether the oracle is cheap or not. To compare the two algorithms more fairly, one should estimate also the arithmetical complexity, and take into account the cost of the oracle.

A key property of composite conditional gradient from a computational perspective is the replacement of the proximal greedy step present in (accelerated) proximal algorithm, described at section 1.3.4, with a linear optimization on $\mathcal{Q}$, the linear minimization operator. For certain types of atomic norms this greedy step can be a much simpler and scalable subproblem.

In fact, when the prox of the atomic norm is expensive to compute, e.g. the prox of nuclear norm, it could be better to use another type of first order oracle: the linear minimization operator (1.22). The point of view changes: an oracle could become a computational bottleneck of an algorithm. Another problem is scalability of the oracle. As instance we take again the nuclear norm: For small size problems prox operator and linear minimization operator are comparable, and this makes accelerated prox gradient more suitable; But sliding to large scale, the prox becomes very expensive, as it needs a full singular values decomposition, whether the linear minimization operator still remains usable in practice, as it needs just the maximum vector pair of the singular value decomposition.

Another strength in favor of conditional gradient with respect to proximal gradient is that conditional gradient is valid for an arbitrary norm. In fact it is requested to find a norm with respect to that the gradient of f is Lipschitz Bubeck (2014). On the other hand, proximal gradient requires ∇f to be Lipschitz with respect to the Euclidean norm.

We conclude that for some atomic norms and problem scale, the prox operator is cheaper than the linear minimization operator, ans then the best algorithm is accelerated prox. For other atomic norms and different scale, the linear minimization operator is cheaper and more scalable than prox operator, and then may be it is worth to use composite conditional gradient.

1.3.6 Lagrangian matching pursuit algorithm

We conclude this section with an algorithm more related to signal processing. The problem solved with lagrangian matching pursuit can be seen as a particular case of the ones described in the previous sections.The main difference is that some variables, instead of being vectors, are sequences or functions of real variable. Proximal methods can also solve this problem.

Suppose it is given a signal $\mathbf{x} \in E$ that want to decompose as a weighted sum of more simple signals. This process of finding these more simple signals and related coefficients is called *signal compression*.

A deep and accurate description can be found in Mallat (2009).

Lagrangian matching pursuit computes a sparse approximation

$$\mathbf{x}_\star = \sum_{i \in \mathcal{I}} c^i \mathbf{d}_i$$

of $\mathbf{x}$ by solving the regularized problem

$$\min_{\{c^i\}_{i \in \mathcal{I}}} \frac{1}{2} \left\| \theta - \sum_{i \in \mathcal{I}} c^i \mathbf{d}_i \right\|^2 + \lambda \sum_{i \in \mathcal{I}} \left| c^i \right| . \tag{1.36}$$

1.4 First order algorithms for convex optimization - constrained case

In the previous section we have seen algorithms to optimize unconstrained problems. Now let us see by symmetry the respective algorithms for problems where the objective is minimized on a closed convex set $\mathcal{Q}$.

1.4.1 Projected subgradient algorithm

We start from an algorithm that considers the objective as a black box with unknown structure.

The projected subgradient optimization algorithm solves the problem

$$\min_{\mathbf{x} \in \mathcal{Q}} f(\mathbf{x}), \tag{1.37}$$

where $\mathcal{Q}$ is a convex set and the function f is convex, possibly nondifferentiable and Lipschitz continuous with constant L.

Projected subgradient ignores the structure of the objective function, that is why it is called black box algorithm. The shape of the constraint $\mathcal{Q}$ is hidden and managed by the projection (1.17), whose efficiency determines the practical (in)utility of this algorithm, i.e. the algorithm is useful when the projection operator is cheap.

We notice that the subgradient is normalized, as its norm is not informative.

The stepsize h_t must be chosen to decrease in a particular way, because the subgradient is normalized.

Algorithm 5 Projected Subgradient Optimization

Choose $\mathbf{x}_0 \in \mathrm{dom}(f)$ and a sequence of step lengths $\{h_t\}_{t>0}$ s.t. $h_t > 0$, $h_t \to 0$, $\sum_{t=0}^{\infty} h_t = \infty$
for $t = 0, 1, 2, \ldots$ **do**
  Choose $\mathbf{s}_t \in \partial f(\mathbf{x}_t)$
  if $\mathbf{s}_t = \mathbf{0}$ **then**
    Exit
  end if
  $\mathbf{x}_{t+1} = \pi_{\mathcal{Q}}(\mathbf{x}_t - h_t(\mathbf{s}_t / \|\mathbf{s}_t\|))$
end for

If by chance a subgradient is zero, then the algorithm stops and the current $\mathbf{x}_t$ is the optimal solution.

The upper bound is

$$\min_{k=0\ldots t} f(\mathbf{x}_k) - f(\mathbf{x}_\star) \leq \frac{LR}{\sqrt{t+1}},$$

where R is the radius of a ball that contains $\mathcal{Q}$ and L the Lipschitz constant of f. This algorithm is slow, but as the lower bound for Lipschitz nondifferentiable functions is $O(1/\sqrt{t})$, see section 1.2.5, it is also optimal and cannot be improved.

1.4.2 Projected gradient algorithm

Projected gradient is similar to the projected subgradient, it has the same problem with the projection operator, which could be expensive. Nevertheless there is a difference: here the norm of the gradient gets smaller when approaching the optimal solution, and is used to adapt the stepsize.

The projected gradient descent algorithm solves the problem

$$\min_{\mathbf{x} \in \mathcal{Q}} f(\mathbf{x}), \tag{1.38}$$

where $\mathcal{Q}$ is a convex set and f is a convex differentiable function.

We see now that this algorithm generates a sequence of so-called feasible points by searching along the descent directions. A point $\mathbf{x}$ is *feasible* if it satisfies all the constraints, i.e. $\mathbf{x}$ is feasible iff $\mathbf{x} \in \mathcal{Q}$. A *feasible direction* at $\mathbf{x} \in \mathcal{Q}$ is a vector $\mathbf{d}$ such that for all $\alpha > 0$ sufficiently small $\mathbf{x} + \alpha\mathbf{d} \in \mathcal{Q}$. Feasible directions algorithms are well described in Bertsekas (2004).

Projected gradient finds the descent direction using the same principle as gradient descent, i.e. use the steepest descent to find a descent direction. Here as there is an additional constraint, which is taken into account by adding a projection operator to the algorithm.

The discussion related to the cost of oracles that we made in section 1.3.5 applies also to projections. To project on a convex ball could be expensive and not usable in practice for large scale problems,

depending on the shape of $\mathcal{Q}$. So, if we want to implement and run the gradient descent in "human time", let us suppose also that the projection onto $\mathcal{Q}$ is easy to compute and scalable with respect to the size of the space n.

Algorithm 6 Projected Gradient Descent

Choose $\mathbf{x}_0 \in \mathrm{dom}(f)$
for $t = 0, 1, 2, \ldots$ **do**
  $\bar{\mathbf{x}}_t = \pi_{\mathcal{Q}}(\mathbf{x}_t - h_t \nabla f(\mathbf{x}_t))$
  $\mathbf{d} = \bar{\mathbf{x}}_t - \mathbf{x}_t$
  $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t \mathbf{d}$
end for

At each iteration t, $\alpha_t \in (0, 1]$ is the stepsize that can be chosen with line search, $h_t > 0$ is a scalar that can be constant for all iterations, differently from subgradient algorithms, because here the gradient gets smaller when approaching to the solution. Here the feasible direction $\mathbf{d} = \bar{\mathbf{x}}_t - \mathbf{x}_t$.

The rate of convergence of projected gradient is the same as that of gradient descent algorithm

$$O\left(\frac{1}{t}\right).$$

This algorithm is faster than projected subgradient, because here the norm of the gradient is informative and decreases while $\mathbf{x}_t$ approaches to the solution, but it is not optimal. In fact, for a smooth function the lower bound is $O(1/t^2)$, as seen in sec 1.2.5. The next algorithm is a faster and optimal version of projected gradient.

Accelerated projected gradient algorithm A way to accelerate the projected gradient consists in reformulate the objective function and then apply accelerated proximal gradient algorithm, described at Section 1.3.3.

The problem to optimize is (1.38), with f L-smooth convex. Using the indicator function of $\mathcal{Q}$, an equivalent formulation is

$$\min_{\mathbf{x} \in E} f(\mathbf{x}) + i_{\mathcal{Q}}(\mathbf{x}).$$

Then the proximal operator of $i_{\mathcal{Q}}$ returns the projection on $\mathcal{Q}$, as seen in 1.19.

The accelerated projected gradient is a fast algorithm and has complexity $O(1/t^2)$, but it needs to compute a projection at each iteration. The cost of the projection operator can be expensive and depends on shape number of dimensions of $\mathcal{Q}$. In that case it could be interesting to chose an algorithm that is slower, but that has an oracle easier to compute. We will see one in the next section.

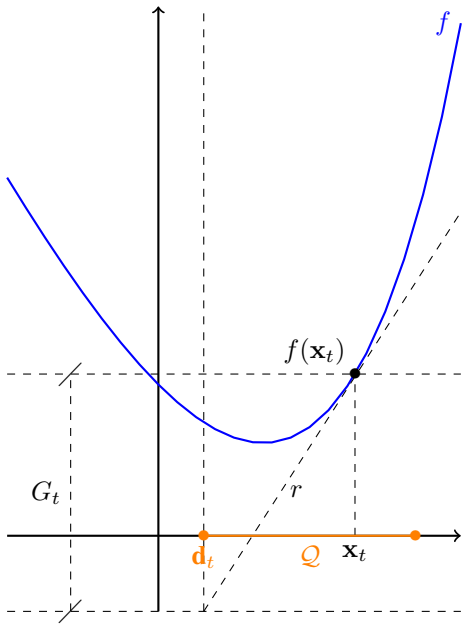

Figure 1.5: Conditional gradient. At $\mathbf{x}_t$ the algorithm finds the gradient, then builds the tangent r. The minimum of r over $\mathcal{Q}$ is $\mathbf{d}_t$, the output of the linear minimization operator. The certificate at iteration t is G_t .

1.4.3 Conditional gradient algorithm

In the previous section 1.4.2, we saw a fast algorithm for constraint optimization based on the proximal operator, accelerated projected gradient. Here we present another algorithm that solves to same problem (1.38) of minimizing an L-smooth function f over a convex $\mathcal{Q}$, conditional gradient Frank and Wolfe (1956) Demyanov and Rubinov (1970), for which we saw a regularized version at section 1.3.5. Another well known name for it is Frank-Wolfe algorithm.

The central operation in conditional gradient is also used

There are two algorithms used for signal compression, denoising and pattern recognition that are based on the scheme of conditional gradient: matching pursuit algorithm and orthogonal matching pursuit. They are described at sections 1.4.4 and 1.4.5.

The upper bound of conditional gradient is

$$f(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \frac{2L}{t+1} \qquad t \geq 2,$$

while the lower bound for smooth functions, as seen at 1.2.5, is $O(1/t^2)$.

Algorithm 7 Conditional Gradient

Choose $\mathbf{x}_0 \in \mathrm{dom}(f)$
for $t = 0, 1, 2 \ldots$ **do**
  $\bar{\mathbf{x}}_t = \mathrm{argmin}_{\mathbf{x} \in \mathcal{Q}} \langle \nabla f(\mathbf{x}_t), \mathbf{x} - \mathbf{x}_t \rangle$
  $\mathbf{d}_t = \bar{\mathbf{x}}_t - \mathbf{x}_t$
  $\mathbf{x}_{t+1} = \mathbf{x}_t + h\mathbf{d}_t$
end for

Despite conditional gradient is not optimal and has a slower rate of convergence than accelerated projected gradient, it could be more suitable depending on the shape of $\mathcal{Q}$. In fact, if the projection on $\mathcal{Q}$ is expensive and the linear minimization operator (1.22) of $\mathcal{Q}$ is cheaper, then it is worth to use conditional gradient.

At each iteration this algorithm calls the Linear Minimization Operator, computes a descent direction $\mathbf{d}$, and finds the next point through line search with $h \in [0, 1]$. We notice that if $\mathcal{Q}$ is the convex envelope of $\mathcal{D}$, this oracle is the same as the one used in composite conditional gradient, equation 1.22.

At the first line of the iteration we can recognize the linear minimization operator, that we defined at (1.22). At each iteration a new atom is generated by this line, called greedy step.

The sequence

$$G_t := \max_{\mathbf{x} \in \mathcal{Q}} \langle \nabla f(\mathbf{x}_{t-1}), \mathbf{x}_{t-1} - \mathbf{x} \rangle = \langle \nabla f(\mathbf{x}_{t-1}), \mathbf{x}_{t-1} - \bar{\mathbf{x}}_t \rangle \tag{1.39}$$

is the accuracy certificate of conditional gradient.

The quantity

$$\langle \nabla f(\mathbf{x}_{t-1}), \mathbf{x}_{t-1} - \bar{\mathbf{x}}_t \rangle \leq \varepsilon,$$

which implies

$$F(\mathbf{x}_t) - F(\mathbf{x}_\star) \leq \varepsilon$$

is a stopping criterion; see Harchaoui et al. (2014); Jaggi (2013) for details and discussion.

This observable stopping criterion is cheap because it is “automatically" computed as by-product of every iteration of composite conditional gradient. In fact, the greedy step at the beginning of each iteration gives already the solution of the maximization problem that defines G_t.

This certificate is important for two reasons: It demands no additional cost in the algorithm, as the information is already generated by the greedy step; It has the same complexity $O(1/t)$ of the upper bound, so it is worth to control the algorithm using the certificate.

The conditional gradient algorithm belongs to the class of feasible directions algorithms for constraint

optimization Bertsekas (2004).

1.4.4 Matching pursuit algorithm

The aim of matching pursuit is to select T vectors in a redundant dictionary $\mathcal{D} = \{\mathbf{d}_i\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is a set of indices, and compute an optimal approximation of a signal $\mathbf{x} \in E$. This problem is NP-hard Mallat (2009). Matching pursuits are iterative algorithms that select vectors from the dictionary one by one to build an ε-optimal approximation $\mathbf{x}_T$. Possible applications are compression, denoising and pattern recognition.

The problem to solve is

$$\begin{cases} \min_{\{c^i\}_{i\in\mathcal{I}}} \frac{1}{2} \left\| \mathbf{x} - \sum_{i\in\mathcal{I}} c^i \mathbf{d}_i \right\|^2 \\ \text{s.t. } \left|\text{supp}\left(\mathbf{c}\right)\right| \leq T \ , \end{cases} \tag{1.40}$$

where the *support* of $\mathbf{c}$ contains the indices i for which the coefficient c^i are different from zero,

$$\text{supp}\left(\mathbf{c}\right) := \left\{ i \in \mathcal{I} \,\middle|\, c^i \neq 0 \right\}.$$

This problem can be formulated as linear program, while the Lagrangian pursuit at section 1.3.6 can not.

Algorithm 8 Matching pursuit

Input signal $\mathbf{x}$, relaxation factor $\alpha \in (0, 1]$
$\mathbf{R}_0 = \mathbf{x}$
for $t = 0, 1, 2 \ldots T$ **do**
  $\mathbf{d}_{i_t}$ s.t. $|\langle \mathbf{R}_t, \mathbf{d}_{i_t} \rangle| \geq \alpha \sup_{i\in\mathcal{I}} |\langle \mathbf{x}, \mathbf{d}_i \rangle|$
  $\mathbf{R}_{t+1} = \mathbf{R}_t - \langle \mathbf{R}_t, \mathbf{d}_{i_t} \rangle \mathbf{d}_{i_t}$
  $c^t_\star = \langle \mathbf{R}_t, \mathbf{d}_{i_t} \rangle$
end for
Return $\mathbf{x}_T = \sum_{t=0}^{T-1} c^t_\star \mathbf{d}_{i_t}$

Algorithm 8 describes matching pursuit, where the values $\mathbf{R}_t$ are called residues and the coefficients of the decomposition are c^t and the atoms are $\mathbf{d}_{i_t}$. The matching pursuit returns a solution

$$\mathbf{x}_T = \sum_{t=0}^{T-1} c^t_\star \mathbf{d}_{i_t}$$

and has linear convergence

$$\left\| \mathbf{x} - \mathbf{x}_T \right\|^2 = O(k^T),$$

with a constant $0 < k < 1$.

1.4.5 Orthogonal matching pursuit algorithm

This is a version of matching pursuit that improves the approximation by orthogonalizing the directions of projection with a Gram-Schmidt procedure. At each iteration the vector $\mathbf{d}_{i_t}$ that is found is orthogonalized with respect to all the previous vectors and the vector $\mathbf{u}_t$ is obtained.

$$\begin{cases} \min_{\{c^i\}_{i\in\mathcal{I}}} \frac{1}{2} \left\| \mathbf{x} - \sum_{i\in\mathcal{I}} c^i \mathbf{d}_i \right\|^2 \\ \text{s.t. } |\text{supp}\,(\mathbf{c})| \leq T \\ \qquad \{\mathbf{d}_i\}_{i\in\text{supp}(\mathbf{c})} \text{ are orthogonal} \end{cases} \tag{1.41}$$

Algorithm 9 Orthogonal matching pursuit

Input signal $\mathbf{x}$, $\alpha \in (0, 1]$
$\mathbf{R}_0 = \mathbf{x}$
for $t = 0, 1, 2 \ldots T$ **do**
  $\mathbf{d}_{i_t}$ s.t. $|\langle \mathbf{R}_t, \mathbf{d}_{i_t} \rangle| \geq \alpha \sup_{i\in\mathcal{I}} |\langle \mathbf{x}, \mathbf{d}_i \rangle|$
  $\mathbf{u}_t = \mathbf{d}_{i_t} - \sum_{i=0}^{t-1} \frac{\langle \mathbf{d}_{i_t}, \mathbf{u}_i \rangle}{\|\mathbf{u}_i\|^2} \mathbf{u}_i$
  $\mathbf{R}_{t+1} = \mathbf{R}_t - \frac{\langle \mathbf{R}_t, \mathbf{u}_t \rangle}{\|\mathbf{u}_t\|^2} \mathbf{u}_t$
  $c_\star^t = \frac{\langle \mathbf{R}_t, \mathbf{u}_t \rangle}{\|\mathbf{u}_t\|^2}$
end for
Return $\mathbf{x}_T = \sum_{t=0}^{T-1} c_\star^t \mathbf{u}_t$

The coefficients are c^t and the atoms are the normalized vectors $\mathbf{u}_t$.

Orthogonal matching pursuit has linear convergence, but for large T the Grahm-Schmidt orthogonalization, the search for $\mathbf{u}_t$ increases significantly the arithmetical complexity of each iteration.

The main reference for (orthogonal) matching pursuit is Mallat (2009).

1.4.6 Randomized incremental algorithm

We focus here on a general family of randomized incremental algorithms based on the principle of majorization-minimization. The principle consists in iteratively minimizing a majorizing surrogate of the objective function. We see an instance of randomized incremental algorithm in which the objective is decomposed in sum of functions. We present the minimization by incremental surrogate optimization (MISO), proposed in Mairal (2013).

Here the optimization problem is

$$\min_{\mathbf{x}\in\mathcal{Q}} f(\mathbf{x}) = \frac{1}{J}\sum_{j=1}^{J} f^j(\mathbf{x}),$$

where $\mathcal{Q}$ is a convex subset of $\mathbb{R}^n$, f^j are a convex functions.

Algorithm At each iteration t, MISO finds surrogates g_t^j of the original functions f^j. It is assumed that the surrogates are majorant of the original function and the difference $g_t^j - f^j$ is L-smooth.

Algorithm 10 Minimization by Incremental Surrogate Optimization

Choose $\mathbf{x}_0 \in \mathcal{Q}$
For all $j = 1 \dots J$ **choose** surrogates g_0^j of f^j near $\mathbf{x}_0$
for $t = 1, 2, \dots$ **do**
  Randomly pick up an index $\hat{j}_t \in 1, \dots, J$
  Choose a surrogate $g_t^{\hat{j}_t}$ of $f^{\hat{j}_t}$ near $\mathbf{x}_{t-1}$.
  $g_t^j = g_{t-1}^j$, for $t \neq \hat{j}_t$
  Compute a surrogate function g_t of f near $\mathbf{x}_{t-1}$
  Update $\mathbf{x}_t \in \operatorname{argmin}_{\mathbf{x}\in\mathcal{Q}} \frac{1}{J}\sum_{j=1}^{J} g_t^j(\mathbf{x})$
end for

Convergence If the surrogates g_t^j are L-smooth and ρ-strongly convex, with $\rho \geq L$, then the convergence for MISO is

$$\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}_\star)] \leq \frac{LJ\left\|\mathbf{x}_\star - \mathbf{x}_0\right\|^2}{2t}.$$

In addition, when f is also μ-strongly convex, the upper bound is linear

$$\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}_\star)] \leq \tfrac{1}{2}\left\|\mathbf{x}_\star - \mathbf{x}_0\right\|^2 \left((1 - \tfrac{1}{J}) + \tfrac{1}{J}\tfrac{L}{\rho+\mu}\right)^{t-1}$$

An overview of first-order convex optimization algorithms and their iteration-complexity is given in Table 1.1.

Unconstrained opt. algorithm	Constrained opt. algorithm	Analytical complexity
Gradient descent	Projected gradient	$O(1/t)$
Proximal gradient		$O(1/t)$
	Randomized incremental	$O(1/t)$
Composite conditional gradient	Frank-Wolfe	$O(1/t)$
Accelerated proximal gradient		$O(1/t^2)$
Accelerated gradient	Accel. projected gradient	$O(1/t^2)$

Table 1.1: **First column** first order algorithms for unconstrained optimization, **second column** first order algorithms for constrained optimization, **third column** magnitude of analytical complexity.

1.5 Machine learning applications

In our studies we apply our theoretical results to statistical image classification and collaborative filtering. Let us start by defining some general notation used in machine learning, that we will use in the next sections to present our applications.

$i = 1, \dots, N$	indices of data
$\mathbf{y}_i \in \mathbb{R}$	response
$\mathbf{x}_i$	feature vector
$(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)$	observations of the random variables $\mathbf{x}$ and $\mathbf{y}$
$S \subset E$	set of parameters (scalars, vectors, or matrices; constraint or the whole space)
$\mathbf{W} \in S$	the model, i.e object containing all the parameters of a predictor function
$\mathbf{W}^{i,j} \in \mathbb{R}$	indicates the entry of row i and column j
$\mathbf{W}^{:,j} \in \mathbb{R}^d$	indicates the j-th column of $\mathbf{W}$
$\mathbf{x} \mapsto \mathcal{F}(\mathbf{W}, \mathbf{x})$	predictor function for a fixed $\mathbf{W}$
$\hat{\mathbf{y}} = \mathcal{F}(\mathbf{W}, \mathbf{x})$	prediction for $\mathbf{x}$ with model $\mathbf{W}$
$\ell(\mathbf{W}, \mathbf{x}, \mathbf{y})$	is a loss function
$R(\mathbf{W}) := \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{W}, \mathbf{x}_i, \mathbf{y}_i)$	is the empirical risk
$\ell^{01}(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \hat{\mathbf{y}} = \mathbf{y} \\ 1, & \hat{\mathbf{y}} \neq \mathbf{y} \end{cases}$	gives 0 for a correct prediction and 1 otherwise
$\lambda > 0$	regularization parameter

Table 1.2: Summary of the notation used in the sections related to applications. We remind that here $\mathbf{x}$ and $\mathbf{y}$ are related to the dataset and have a completely different meaning of the $\mathbf{x}$ and $\mathbf{y}$ used in the sections about optimization. We do not minimize with respect to $\mathbf{x}$ or $\mathbf{y}$, but we minimize with respect to $\mathbf{W}$.

1.5.1 Collaborative filtering for movie advertising

Collaborative filtering, or matrix completion, consists in the generation of a low-rank matrix from few known approximate entries. This entries are ratings of users to movies they have seen. The aim of this task is to predict, i.e. to guess, what rating someone would give to a movie he hasn't yet seen. This kind of prediction could be used, for instance, to advise to watch the movies for which we predict high rating. Here the feature vector $\mathbf{x}_i$ belongs to $\mathbb{R}^2$.

We use the loss $\ell(\mathbf{w}, \mathbf{x}) = |\mathbf{w} - \mathbf{x}|$, based on ℓ_1 norm Huber (1981), that ensures robustness to outliers.

The feature vector $\mathbf{x}_i = (\mathbf{x}_i^1, \mathbf{x}_i^2) = (row, column) \in \mathbb{N}^2$ contains just two integers that represent (1) the index for an user and (2) the index for a movie. So $\{\mathbf{x}_i\}_{i=1}^N$ is the list of observed entries. A response

$\mathbf{y}_i$ is the rating of the user number $\mathbf{x}_i^1$ for the movie number $\mathbf{x}_i^2$ and is stored in the matrix $\mathbf{Y}$ at row $\mathbf{x}_i^1$ and column $\mathbf{x}_i^2$.

We have the regularized problem

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times k}} \quad \frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{W}^{\mathbf{x}_i^1\mathbf{x}_i^2}-\mathbf{y}_i\right| + \lambda\left\|\mathbf{W}\right\|_{\mathcal{D}}$$

and the constraint problem

$$\begin{cases} \min\limits_{\mathbf{W}\in\mathbb{R}^{d\times k}} & \frac{1}{N}\sum\limits_{i=1}^{N}\left|\mathbf{W}^{\mathbf{x}_i^1\mathbf{x}_i^2}-\mathbf{y}_i\right| \\ \text{s.t.} & \left\|\mathbf{W}\right\|_{\mathcal{D}} \leq r \end{cases}$$

The predicted value is just an entry of $\mathbf{W}$:

$$\hat{\mathbf{y}} = \mathcal{F}(\mathbf{W},\mathbf{x}) = \mathbf{W}^{\mathbf{x}^1\mathbf{x}^2}.$$

MovieLens dataset We test our approach on the MovieLens dataset for collaborative filtering, described in Miller et al. (2003) and available at `http://grouplens.org`. This dataset contains evaluations of movies made by customers, represented by the sparse matrix $\mathbf{Y} \in \mathbb{R}^{d\times k}$. As every customer evaluated only a small number the movies, $\mathbf{Y}$ is sparse. Here completing $\mathbf{Y}$ means predicting how a customer would evaluate a movie which he hasn't seen. Entries of $\mathbf{Y}$ are normalized dividing by the max entry of $\mathbf{Y}$ which is 5. So $\mathbf{Y}$ has entries in $[0,1]$.

71 567	users
10 681	movies
10 000 054	ratings

then the sparsity of the resulting matrix $\mathbf{Y}$ is 1.3 Some real data from MovieLens are represented at Fig.1.6.

1.5.2 Multiclass image classification

Let us introduce two types of classification: the classical multiclass classification, called top-1 classification, and an extension that is often used in machine learning, especially in computer vision, the top-k multiclass classification. The fundamental distinction between them is how to define if an example has been correctly classified.

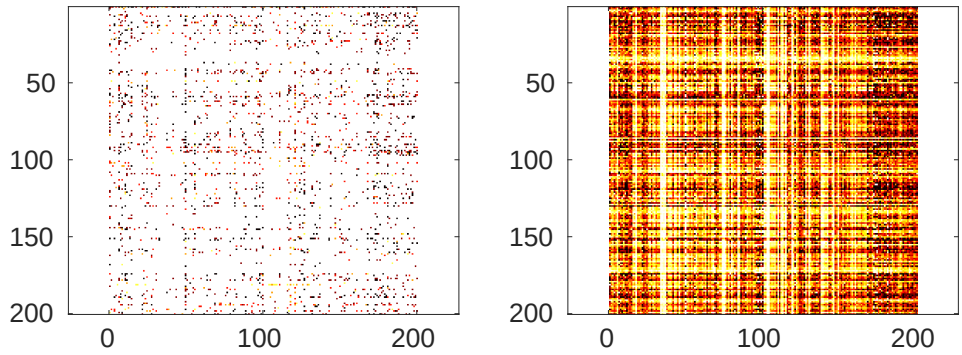

Figure 1.6: View of first 200 rows and columns of Movielens data. **(Left)** the input matrix of observations $\mathbf{Y}$, where the non observed ratings are drawn in white, **(right)** the low rank solution $\mathbf{W}$ obtained running SCCG. The highest ratings are black, then red. The more poor ratings are and yellow and white.

Top-1 classification

Here the feature vector $\mathbf{x}_i$ belongs to $\mathbb{R}^d$. Let $p_1 = (\mathbf{x}_1, \mathbf{y}_1), \ldots, p_N = (\mathbf{x}_N, \mathbf{y}_N)$ be labeled training data examples, where $\mathbf{x}_i \in \mathbb{R}^d$ are feature vectors, k is the number of classes and the integers $\mathbf{y}_i \in [1 \ldots k]$ are the associated class labels. A linear classifier is specified by a separate weight vector $\mathbf{W}^{:\mathbf{y}} \in \mathbb{R}^d$. For a given test example $\mathbf{x} \in \mathbb{R}^d$, the predicted class is

$$\hat{\mathbf{y}} = \mathcal{F}(\mathbf{x}, \mathbf{W}) := \operatorname*{argmax}_{j=1\ldots k} \mathbf{W}^{:j\top}\mathbf{x}. \tag{1.42}$$

An example p_i is said to be correctly classified if the prediction and the true label coincide, as defined by the 0-1-loss

$$\ell^{01}_{\text{top},1}(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{y} = \operatorname*{argmax}_{j=1\ldots k} \mathbf{W}^{:j\top}\mathbf{x} \qquad \text{(correct classification)} \\ 1 & \text{if } \mathbf{y} \neq \operatorname*{argmax}_{j=1\ldots k} \mathbf{W}^{:j\top}\mathbf{x} \qquad \text{(misclassification).} \end{cases} \tag{1.43}$$

The optimization task consists in finding a model $\mathbf{W}$ to minimize the top-1 misclassification error

$$R^{01}_{\text{top},1}(\mathbf{W}) := \frac{1}{N}\sum_{i=1}^{N} \ell^{01}_{\text{top},1}(\mathbf{W}, \mathbf{x}_i, \mathbf{y}_i). \tag{1.44}$$

and the top-1 accuracy is defined as

$$\text{acc}_{\text{top},q}(\mathbf{W}) := 1 - R^{01}_{\text{top},1}(\mathbf{W}). \tag{1.45}$$

But $R^{01}_{\text{top},1}$ is not convex and hard to optimize. Let us present a convex upper bound of it.

We first define the affine map $\mathcal{A}_{\mathbf{x},\mathbf{y}} : \mathbb{R}^{d\times k} \to \mathbb{R}^k$ for classification related to the example $p = (\mathbf{x}, \mathbf{y})$

$$\mathcal{A}_{\mathbf{x},\mathbf{y}}\mathbf{W} := \{1 - \delta(j, \mathbf{y}) + (\mathbf{W}^{:j} - \mathbf{W}^{:\mathbf{y}})^\top \mathbf{x}\}_{j=1}^k. \tag{1.46}$$

where $\delta(j, i) := 0,$ if $j \neq i,$ 1 otherwise. The hinge function for top-1 multiclass is defined as

$$\ell^H_{\text{top},1}(\mathbf{W}, \mathbf{x}, \mathbf{y}) := \max_{j=1\ldots k} (\mathcal{A}_{\mathbf{x},\mathbf{y}}\mathbf{W})_j$$

and is an upper bound of (1.43), i.e. for all $\mathbf{W} \in \mathbb{R}^{d\times k}$

$$\ell^H_{\text{top},1}(\mathbf{W}, \mathbf{x}, \mathbf{y}) \geq \ell^{01}_{\text{top},1}(\mathbf{W}, \mathbf{x}, \mathbf{y}),$$

and this is because $\mathcal{A}_{\mathbf{x},\mathbf{y}}\mathbf{W}$ has always one nonnegative entry so that

$$\max_{j=1\ldots k} (\mathcal{A}_{\mathbf{x},\mathbf{y}}\mathbf{W})_j \geq 0.$$

A convex upper bound of the misclassification error is

$$R^H_{\text{top},1}(\mathbf{W}) := \frac{1}{N}\sum_{i=1}^N \ell^H_{\text{top},1}(\mathbf{W}, (\mathbf{x}_i, \mathbf{y}_i)).$$

So now we have a convex function $R^H_{\text{top},1}(\mathbf{W})$ to optimize. It is still an ill-posed problem, so we are interested to optimize a regularized (doubly) nonsmooth version

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times k}} R^H_{\text{top},1}(\mathbf{W}) + \lambda \|\mathbf{W}\|_{\mathcal{D}} = \frac{1}{N}\sum_{i=1}^N \max_{j=1\ldots k} (\mathcal{A}_{\mathbf{x}_i,\mathbf{y}_i}\mathbf{W})^j + \lambda \|\mathbf{W}\|_{\mathcal{D}}\,, \tag{1.47}$$

and a nonsmooth constrained version

$$\min_{\|\mathbf{W}\|_{\mathcal{D}}\leq r} R^H_{\text{top},1}(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^N \max_{j=1\ldots k} (\mathcal{A}_{\mathbf{x}_i,\mathbf{y}_i}\mathbf{W})^j \tag{1.48}$$

The problems (1.48) and (1.48) are now convex and good to optimize. The only issue with them is the nondifferentiability: state of art algorithms treat this condition in a bad way, especially (1.47), which is doubly nonsmooth.

Top-q classification

Let us see another more general type of multiclass classification, of which the one we introduced above is a particular case. Here we are interested in predicting q classes instead of only one. We assume that: (i) Any image contains more than only one object. Even if there is a central big one, several other small objects appear. (ii) Speaking of a dataset, the ground truth, i.e. the true label of each image, is subjective and depends on the personal choice of the human. The function we learn associates to each image a list of q labels ordered by relevancy, instead of having only the best one as in top-1 classification. Then the q predicted classes by the model $\mathbf{W}$ for the example x are given by

$$(\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(q)}) = \mathcal{F}(\mathbf{W}, \mathbf{x}) := \underset{j=1\ldots k}{\text{q-argmax}} \ \mathbf{W}^{:j\top}\mathbf{x}, \tag{1.49}$$

where q-argmax returns a set of q values instead of only one as in the traditional argmax operator.

One motivation to use the prediction function (1.49) is when two objects A and B are similar, for example two types of trees. It is acceptable that (1.49) predicts both of them among the first q results and does not take care too much of the subjective choice of the ground truth. On the other hand, the top-1 can predict A when the true is B, and we would get the same error as predicting any other object. An example of subjective labeling is an image with a bee on a flower. It could have the label `bee`, but the classifier gives the top score to the class `flower` and the second top score to `bee`. While the top-1 accuracy would accept only `flower` and consider this a misclassification, a top-q accuracy (here $q \geq 2$) would consider it as a good classification.

Another motivation is to use top-q inside a sequence of computer programs. For example a first program could find the best q "candidate" classes for the given image. After that, a second program could apply object detection to identify bounding boxes of different objects and a third program could apply filters to find the most relevant class. For large scale number of classes k it is considered good to have the true class predicted among the first q of highest score. We assume that q is much smaller than k, for example $q = 5$ or $q = 10$. Top-q accuracy is commonly used to evaluate the performance of a classifier. Let us define first the misclassification top-q loss

$$\ell^{01}_{\text{top},q}(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{y} \in \underset{j=1\ldots k}{\text{q-argmax}} \ \mathbf{W}^{:j\top}\mathbf{x} \qquad \text{correct classification} \\ 1 & \text{if } \mathbf{y} \notin \underset{j=1\ldots k}{\text{q-argmax}} \ \mathbf{W}^{:j\top}\mathbf{x} \qquad \text{misclassification.} \end{cases} \tag{1.50}$$

The top-q error, that is the empirical risk, is defined as the rate of true labels $\mathbf{y}_i$ not present among the

first q predicted

$$R^{01}_{\text{top},q}(\mathbf{W}) := \frac{1}{N}\sum_{i=1}^{N} \ell^{01}_{\text{top},q}(\mathbf{W}, \mathbf{x}_i, \mathbf{y}_i) \tag{1.51}$$

and the top-q accuracy is defined as

$$\text{acc}_{\text{top},q}(\mathbf{W}) := 1 - R^{01}_{\text{top},q}(\mathbf{W}). \tag{1.52}$$

We shall generalize to top-q the convex hinge presented for the top-1 in this section.

1.6 Contributions in this context

In this section we introduce our contributions presented with details in the next chapters. We define and study a new norm for matrices that are decomposed into groups. We describe first some techniques for the construction of surrogates that approximate nonsmooth functions. Then we create new algorithms based on conditional gradient to solve nonsmooth optimization problems using smoothing as a tool. Each contribution is presented in its main lines together with some recent related work. More details and further references are given in corresponding chapters.

1.6.1 Group Schatten norm

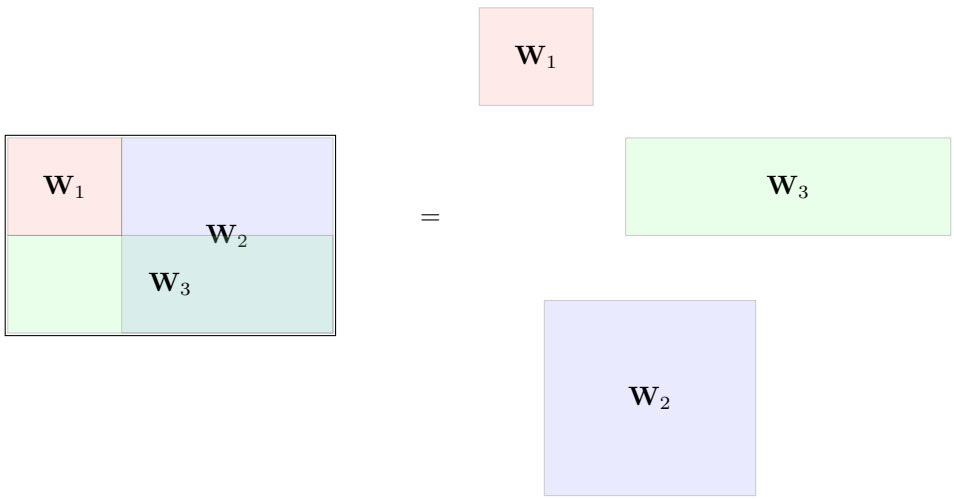

Figure 1.7: The matrix on the left is composed by the matrices on the right, that we call ‘groups’.

Contribution The popular nuclear norm, i.e. the sum of singular values of a matrix, is the convex hull of the rank, properly restricted to a ball of its domain. The nuclear norm used as regularizer in optimization problems has the useful property of generating low rank solutions, which improves robustness of models in statistical learning tasks. The Schatten norm is a generalization of matrix norms as the Frobenius norm, the ℓ^1 norm and the ℓ^∞ norm.

Here we introduce a new mathematical object: the group Schatten norm, as generalization of Schatten norm for matrices that are decomposed into groups. We call groups the submatrices obtained by eliminating rows and columns, possibly non-contiguous. The name group Schatten norm comes then from the groups in which the full matrix is decomposed, and the Schatten norm, that is applied to each group.

This contribution is two-fold: in the first part we study the properties of the group Schatten norm from the point of view of convex analysis and we consider it as the convex envelope of the reweighted group rank function; In the second part we consider one particular instance of group Schatten norm: the group nuclear norm. We apply the group nuclear norm as regularizer for optimization problems, which has the interesting property of forcing the solution to have low rank in each group. In addition we illustrate an application for matrix completion, where we reconstruct a matrix from a small sample of its entries.

We discuss potential applications of the group nuclear norm, divided into tree categories: collaborative filtering, database compression, multi attribute image classification. In the latter, the strength point of the group nuclear norm is that it can take into account a tree hierarchy of the examples and at the same time it can handle attributes independently from the hierarchical structure.

Related works The recent Tomioka and Suzuki (2013) independently proposed a norm similar to our group Schatten norm, the 'latent' norm for tensors, that can be used to regularize convex problems and obtain an equivalent of low-rank solution, where the rank is extended to tensors. From both theoretical and empirical approach the latent norm performs better than the other 'overlapped' norms for tensors.

1.6.2 Smoothing techniques for learning with first-order optimization

Contribution Nonsmooth problems are key tools for applications in machine learning because they lead to more robust predictive models. The main issue with these problems is that state of the art algorithms, as subgradient or bundle-type algorithms, are not scalable with the size of the space when optimizing some particular nonsmooth function, as instance the nuclear norm. On the other hand, to optimize a smooth objective is in general easier and faster, but the key property of model robustness could vanish. Therefore, the practical advantages of nonsmooth optimization suggests to study algorithms that can handle nonsmooth functions. We are interested in surrogate functions and scalable algorithms to

approximate the nonsmooth objectives and to solve nonsmooth problems.

This contribution is divided in two: in the first part we describe the construction of smooth surrogates approximating nonsmooth functions; in the second part we propose new algorithms to solve nonsmooth optimization problems. The action of building a surrogate to approximate a nonsmooth function is denoted “smoothing". We present a simple framework of smoothing that can be summarized into two main classes of smoothing techniques: i) The smooth surrogate is obtained by infimal convolution of the nonsmooth function with a smooth convex function Beck and Teboulle (2012). To this class belong the Moreau-Yosida transform, the Fenchel-type approximation, the Nesterov smoothing and the asymptotic function smoothing. ii) The smooth surrogate is obtained by convolution of the nonsmooth function with a measurable function Duchi et al. (2012).

We develop also a new way to smooth the top-k error, used in ranking and classification. We show also how to apply the smoothing to collaborative filtering for movie advertising and multiclass image classification.

In the second part we propose three new algorithms for nonsmooth problems, obtained by combining these smoothing techniques with state of art algorithms: composite conditional gradient, FISTA, and MISO. We take advantage of the smoothing to partially smooth the objective and plug the surrogate into the algorithm.

Related works Fast gradient methods as FISTA suffer from error accumulation when the subproblem for the oracle returns an approximate solution. This issue is tackled in Devolder et al. (2014), where the notion of inexact first-order oracle is introduced and better convergence results are obtained for smoothed max-representable functions. An example of inexact oracle is given by the approximation by a smooth function of the objective.

An extension of Nesterov method and FISTA, called proximal iterative smoothing algorithm, is presented in Orabona et al. (2012). This algorithm optimizes an objective function that is the sum of three convex parts: of which one smooth, and one Lipschitz. The proximal operators of the nonsmooth parts are supposed to be easy to compute. Applications are robust PCA, sparse inverse covariance selection, and collaborative filtering and clustering regularized with the max-norm. An advantage of this algorithm is that it has an optimal stepsize without knowing in advance the number of iterations.

1.6.3 Conditional gradient algorithms for doubly non-smooth learning

Contribution The problems we want to solve here are doubly nonsmooth: the loss is nonsmooth and the regularization too. For example, combining the nuclear norm as regularization with a nonsmooth loss,

we get a doubly nonsmooth problem for wich the existing algorithms are not scalable with the size of the problem. On the other hand, (composite) conditional gradient is scalable for nonsmooth functions that are "difficult", as the nuclear norm, but a smooth loss is needed. When the loss is nonsmooth, (composite) conditional gradient gets stuck at some iterations and does not converge to the optimum.

Our contribution is the creation of two new algorithms based on (composite) conditional gradient. The first algorithm is adaptive conditional gradient, which optimizes a nonsmooth function with a constraint represented by the ball of a nonsmooth function. The second algorithm is adaptive composite conditional gradient, which optimizes the sum of a nonsmooth loss and a nonsmooth regularization.

The main idea is that we use as tool the smoothing techniques: adaptive (composite) conditional gradient is a combination of adaptive smoothing and (composite) conditional gradient. The convergence of adaptive (composite) conditional gradient toward the solution of the initial nonsmooth problem is proved with guaranties.

We run computational experiences on applications to collaborative filtering for movie advertising and multiclass classification of images.

Related works An extension of conditional gradient, called block-coordinate Frank-Wolfe optimization for structural SVM, is presented in Lacoste-Julien et al. (2013). This algorithm is adapted to problems whose constraints are separable into cartesian product. While keeping the same convergence rate of conditional gradient, the oracle cost at each iteration is cheaper.

Conditional gradient can be seen as an extension of herding algorithm Bach et al. (2012a). The two algorithms are the same when applied to the problem of estimating the mean of a convex polytope.

In Lacoste-Julien et al. (2015) conditional gradient is used to approximate the integral of functions, useful in case of expensive estimation of the likelihood. When applied to particle filtering problems, it gives better performance of random sampling or quasi-Monte Carlo sampling,

Several variants of conditional gradient called 'away-steps', 'pairwise' and 'fully-corrective', are presented in Lacoste-Julien and Jaggi (2015). These variants keep linear convergence for strongly convex objectives, avoid the zigzagging problem when the solution is on the boundary of the constraint and in practice the convergence is improved.

New convergence rates for offline smooth and strongly convex optimization, new online algorithms and stochastic algorithms for convex optimization are provided in Garber and Hazan (2015), considering convex problems constrained by polytope. Here the minimization of a linear function on a polytope is supposed to be easy to compute, e.g. for the matroid polytope, the flow polytope, and the set of rotations.In these examples the projection over the polytope is not scalable with respect to the size of

the space, but the proposed algorithms avoid the projection. Instead, the proposed conditional gradient requires only one call to the linear optimization oracle per iteration and per game.

Chapter 2

Group Schatten norm

2.1 Introduction

The rank function, when properly restricted to a subset of its domain, has a well-known convex envelope: the nuclear norm. We introduce a new norm that can be derived similarly from a rank function of submatrices.

Regularization is a popular approach in statistics as it leads to sparse models, which have good interpretation in applications such as computer vision, biology and social sciences.

A popular example of regularization is the nuclear norm, which leads to low-rank models, but it does not allow inclusion of any *a priori* information about the groups of covariates. We introduce a new norm for the setting when groups of covariates are given over which the matrix should be of low rank. Our norm can be viewed as a matrix generalization of the popular group lasso norm (Jacob et al., 2009; Yuan and Lin, 2006; Zhao et al., 2009).

Used as a penalty function in optimization, the group nuclear norm leads to sparse matrix solutions, where the sparsity here corresponds to a low rank when restricted to some submatrices. Optimization problems that enforce low-rank solutions are building blocks of several important applications in machine learning, including recommender systems and multiclass classification algorithms for a large number of classes.

The chapter is organized as follows. In Section 2.3 we describe the decomposition of a matrix into groups and define the group Schatten norm. In Section 2.4 we analyze the properties of the group Schatten norm from the point of view of convex analysis, and consider it as the convex envelope of the reweighted group rank function. In Section 2.5 we show how to solve optimization problems regularized by the group

nuclear norm, using two popular algorithms: FISTA and composite conditional gradient. In Section 2.6 we illustrate how to recover a matrix by observing some random entries.

2.2 Notation

In a vector space $\mathbb{R}^d$, we use notation $\|\cdot\|_p$, $p \geq 1$ for a general ℓ_p-norm. For a matrix $\mathbf{W}$ in $\mathbb{R}^{d\times k}$, we write $\sigma(\mathbf{W})$ for the spectrum viewed as a vector of singular values of $\mathbf{W}$. We define the so-called Schatten p-norm as

$$\|\mathbf{W}\|_{\sigma,p} := \|\sigma(\mathbf{W})\|_p \ .$$

We obtain the spectral norm by choosing $p = \infty$, Frobenius norm by choosing $p = 2$, and the nuclear norm of the matrix by choosing $p = 1$. The latter is also called the trace norm, because for positive definite semidefinite matrices it is equal to the trace. It is popular in machine learning, where it is the convex surrogate of the rank optimization.

2.3 Group Schatten norm: Definition and examples

Motivated by applications in machine learning, we refer to the rows of matrix $\mathbf{W}$ as covariates and columns as classes. To set up our group norm, we need to introduce the notions of groups of classes and covariates. Let $\mathcal{G}$ denote a finite set of indices, where each index $g \in \mathcal{G}$ is associated with a set of classes $\mathcal{Y}_g \subseteq \mathcal{Y} := \{1, \ldots, k\}$ and a set of covariates $\mathcal{D}_g \subseteq \mathcal{D} := \{1, \ldots, d\}$. A *group* is a set of index pairs $\mathcal{D}_g \times \mathcal{Y}_g$. The groups may overlap, we just assume that the groups cover all the covariates and classes:

$$\bigcup_{g\in\mathcal{G}} \mathcal{D}_g \times \mathcal{Y}_g = \mathcal{D} \times \mathcal{Y} \ .$$

The groups are known and fixed in advance.

Next, consider the vector space $E_{\mathcal{G}}$ defined as the direct sum of spaces $\mathbb{R}^{|\mathcal{D}_g|\times|\mathcal{Y}_g|}$ across all g,

$$E_{\mathcal{G}} := \bigoplus_{g\in\mathcal{G}} \mathbb{R}^{|\mathcal{D}_g|\times|\mathcal{Y}_g|} \ .$$

Thus, $E_{\mathcal{G}}$ is a vector space whose elements, $(\mathbf{W}_g)_g$, are tuples of $|\mathcal{G}|$ matrices.

In addition to groups $\mathcal{G}$, we fix the positive weights α_g, $g \in \mathcal{G}$. In $E_{\mathcal{G}}$, we consider a norm which is a

weighted sum of p-Schatten norms over the elements of the tuple,

$$\Omega_{\oplus}((\mathbf{W}_g)_g) := \sum_{g \in \mathcal{G}} \alpha_g \|\mathbf{W}_g\|_{\sigma,p} \ . \tag{2.1}$$

Each group g is associated with the following two maps:

- $\Pi_g \colon \mathbb{R}^{d \times k} \to \mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}$ is the projection such that $\Pi_g(\mathbf{W})$ zeroes out the entries of $\mathbf{W}$ not indexed by $\mathcal{D}_g$ and $\mathcal{Y}_g$.
- $i_g \colon \mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|} \to \mathbb{R}^{d \times k}$ is the natural (injective) embedding of $\mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}$ into $\mathbb{R}^{d \times k}$, such that $i_g \circ \Pi_g$ is the identity function on $\mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}$. Thus, Π_g is the adjoint of i_g, *i.e.* for all $\mathbf{Z} \in \mathbb{R}^{d \times k}$ and all $\mathbf{W}_g \in \mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}$ we have $\langle \mathbf{Z}, i_g(\mathbf{W}_g) \rangle_{\mathbb{R}^{d \times k}} = \langle \Pi_g(\mathbf{Z}), \mathbf{W}_g \rangle_{\mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}}$.

The mappings i_g and Π_g allow us to identify the space $\mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}$ with its image in $\mathbb{R}^{d \times k}$.

Consider also the linear operator $\mathcal{A}_{\mathcal{G}} \colon E_{\mathcal{G}} \to \mathbb{R}^{d \times k}$ defined by

$$\mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g) := \sum_{g \in \mathcal{G}} i_g(\mathbf{W}_g).$$

The group p-Schatten norm associated with a grouping $\mathcal{G}$ is defined as

$$\Omega_{\mathcal{G}}(\mathbf{W}) := \min_{\substack{(\mathbf{V}_g)_g \in E_{\mathcal{G}} \\ s.t.\ \mathbf{W} = \mathcal{A}_{\mathcal{G}}((\mathbf{V}_g)_g)}} \sum_{g \in \mathcal{G}} \alpha_g \|\mathbf{V}_g\|_{\sigma,p}. \tag{2.2}$$

The fact that $\Omega_{\mathcal{G}}$ as defined above is indeed a norm will follow easily from Lemma 1. Many basic properties of $\Omega_{\mathcal{G}}$ come from its interpretation as a so-called "image-function", a standard notion of convex analysis Hiriart-Urruty and Lemarechal (1996).

For the special case when the groups are disjoint, the decomposition of $\mathbf{W}$ is unique with $\mathbf{W}_g = \Pi_g(\mathbf{W})$, *i.e.*

$$\Omega_{\mathcal{G}}(\mathbf{W}) = \sum_{g \in \mathcal{G}} \alpha_g \|\Pi_g(\mathbf{W})\|_{\sigma,p}, \tag{2.3}$$

and the minimum in (2.2) is attained at a single element of $E_{\mathcal{G}}$.

Some of the examples of the group Schatten norm with $p = 1$, a.k.a. the group nuclear norm, are the following:

1. We get the nuclear norm $\Omega_{\text{trace}}(\mathbf{W}) = \|\mathbf{W}\|_{\sigma,1}$ by defining one group that fully covers the matrix, $\mathcal{G} = \{1\}$, $\mathcal{D}_1 = \{1, \ldots, d\}$, and $\mathcal{Y}_1 = \{1, \ldots, k\}$.

2. We get the matrix ℓ_1 norm $\Omega_{\text{lasso}}(\mathbf{W}) = \sum_{j=1}^{d} \sum_{y \in \mathcal{Y}} |W_{jy}|$ by defining $d \times k$ groups, each covering one matrix entry.

3. We get the group lasso $\Omega_{\text{gr-lasso}}(\mathbf{W}) = \sum_{j=1}^{d} \|\mathbf{W}_j\|_2$, where $\mathbf{W}_j$ denotes the jth column of $\mathbf{W}$, by defining k groups $\{1, \ldots, d\} \times \{y\}$ for any $y \in \mathcal{Y}$, since the nuclear norm of the column $\mathbf{W}_j$ is just $\|\mathbf{W}_j\|_{\sigma,1} = \|\mathbf{W}_j\|_2$.

Lemma 1 (Norm). The group Schatten norm $\Omega_{\mathcal{G}}$ is a norm on $\mathbb{R}^{d \times k}$ and the minimum in (2.2) is always attained on a nonempty compact convex subset of $E_{\mathcal{G}}$.

Proof. (Of Lemma 1) We can write the group Schatten norm as

$$\Omega_{\mathcal{G}}(\mathbf{W}) = \inf_{\substack{(\mathbf{V}_g)_g \in E_{\mathcal{G}} \\ \text{s.t. } \mathbf{W} = \mathcal{A}_{\mathcal{G}}((\mathbf{V}_g)_g)}} \Omega_{\oplus}((\mathbf{V}_g)_g) \tag{2.4}$$

We recognize now that $\Omega_{\mathcal{G}}$ is the image-function on the affine space

$$\mathcal{W}_{\Omega_{\mathcal{G}}} = \{(\mathbf{V}_g)_g \in E_{\mathcal{G}} \,|\, \mathbf{W} = \mathcal{A}_{\mathcal{G}}((\mathbf{V}_g)_g)\} \ ,$$

see (Hiriart-Urruty and Lemarechal, 1996, B.2.4.1). From this expression, we deduce first that $\Omega_{\mathcal{G}}$ is convex (by (Hiriart-Urruty and Lemarechal, 1996, B.2.4.2)), and we also see that $\Omega_{\mathcal{G}}$ is indeed a norm on $\mathbb{R}^{d \times k}$, since so is $\Omega_{\oplus}$. Finally, since $\Omega_{\oplus}$ is closed convex, the set of minimizers in (2.4) must also be closed and convex. Since $\Omega_{\oplus}$ is moreover a norm, it has bounded sublevel sets and therefore the set of minimizers in (2.4) is also bounded and thus compact. □

2.4 Group nuclear norm as a convex surrogate

A standard nuclear norm acts as a convex surrogate for rank function in various machine learning applications. Here we show that the group Schatten norm can be similarly viewed as a convex surrogate that combines the notion of sparsity among different group sparsity and small rank within each group.

The next theorem shows that the group Schatten norm is in fact the convex surrogate of two functions involving submatrices, defined on $\mathbb{R}^{d \times k}$: the “reweighted group rank" function

$$\Omega_{\mathcal{G}}^{\text{rank}}(\mathbf{W}) := \inf_{\substack{(\mathbf{W}_g)_g \in E_{\mathcal{G}} \\ \mathbf{W} = \mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g)}} \sum_{g \in \mathcal{G}} \alpha_g \operatorname{rank}(\mathbf{W}_g), \tag{2.5}$$

and the "reweighted rank" function

$$\Omega^{\text{rank}}(\mathbf{W}) := \min_{g \in \mathcal{G}} \quad \alpha_g \operatorname{rank}(\Pi_g(\mathbf{W})) + \delta_g(\mathbf{W}), \tag{2.6}$$

where δ_g is the indicator function of the group g:

$$\delta_g(\mathbf{W}) := \begin{cases} 0 & \text{if the nonzero entries of } \mathbf{W} \text{ are all included in the } g\text{-th group} \\ +\infty & \text{otherwise.} \end{cases}$$

These functions, when used as regularizers in optimization problems, enforce solutions that are low rank on some groups and zero outside.

Theorem 1 (Convex hull). The group nuclear norm $\Omega_{\mathcal{G}}$ is the closed convex hull of the two functions (2.6) and (2.5), when restricted to the convex set

$$C := \left\{ \mathbf{W} \in \mathbb{R}^{d \times k} \,\middle|\, (\mathbf{W}_g)_g \in E_{\mathcal{G}},\, \sum_{g=1}^{|\mathcal{G}|} \sigma_{\max}(\mathbf{W}_g) \leq 1,\, \mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g) = \mathbf{W} \right\}.$$

Proof. (Of Theorem 1) Note that C is convex by definition, since $\sigma_{\max}$ is a convex function and $\mathcal{A}_{\mathcal{G}}$ is a linear operator.

Let us prove now that, for all $\mathbf{W}$ in C, we can bound

$$\Omega^{\text{rank}}(\mathbf{W}) \geq \Omega^{\text{rank}}_{\mathcal{G}}(\mathbf{W}) \geq \Omega_{\mathcal{G}}(\mathbf{W}).$$

That $\Omega^{\text{rank}} \geq \Omega$ is evident from the the results we introduced in the previous section.

We start to show that $\Omega^{\text{rank}}_{\mathcal{G}} \leq \Omega^{\text{rank}}$ restricted to C. Let us call $\hat{g}$ the minimizer of the right part of inequality. Then, if not always infinity, $\Omega^{\text{rank}}(\mathbf{W}) = \alpha_{\hat{g}} \operatorname{rank}(\Pi_{\hat{g}}(\mathbf{W}))$. For the left hand side we can take a $(\mathbf{W}_g)_g$ such that $\mathbf{W}_{\hat{g}} = \mathbf{W}$ and $\mathbf{W}_g$ equal the zero matrix for $g \neq \hat{g}$. We observe that this verifies $\mathbf{W} = \mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g)$. Then the left argument not bigger than $\alpha_{\hat{g}} \operatorname{rank}(\Pi_{\hat{g}}(\mathbf{W}))$.

Now we show that $\Omega_{\mathcal{G}}$ is the convex hull of Ω^{rank} by showing that $\Omega_{\mathcal{G}}$ is equal to the double Fenchel conjugate $(\Omega^{\text{rank}})^{**}$. The first inequality is easy to get by using Ω^{rank} as the minimum of many functions: $\Omega^{\text{rank}}(\mathbf{W}) = \min_g \psi_g(\mathbf{W})$, with

$$\psi_g(\mathbf{W}) := \begin{cases} \alpha_g \operatorname{rank}(\mathbf{W}) + \delta_g(\mathbf{W}) & \sigma_{\max}(\mathbf{W}) \leq 1 \\ +\infty & \text{otherwise.} \end{cases}$$

We get also that

$$(\Omega^{\text{rank}})^*(\mathbf{V}) = \max_g \psi_g^*(\mathbf{V}) \tag{2.7}$$

for any $\mathbf{V} \in \mathbb{R}^{d\times k}$.

$$\begin{aligned}
(\Omega^{\text{rank}})^*(\mathbf{V}) &= \sup_{\mathbf{Z}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{Z}\rangle - \Omega^{\text{rank}}(\mathbf{Z}) \\
&= \sup_{\mathbf{Z}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{Z}\rangle - \min_{g\in\mathcal{G}} \quad \alpha_g \operatorname{rank}(\Pi_g(\mathbf{W})) + \delta_g(\mathbf{W}) \\
&= \sup_{\mathbf{Z}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{Z}\rangle + \max_{g\in\mathcal{G}} -\alpha_g \operatorname{rank}(\Pi_g(\mathbf{W})) - \delta_g(\mathbf{W}) \\
&= \max_{g\in\mathcal{G}} \sup_{\mathbf{Z}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{Z}\rangle - \alpha_g \operatorname{rank}(\Pi_g(\mathbf{W})) - \delta_g(\mathbf{W}) \\
&= \max_{g\in\mathcal{G}} \psi_g^*(\mathbf{V}).
\end{aligned}$$

Now we conclude by showing that $\Omega_{\mathcal{G}}$ is Ω^{rank} conjugated twice.

$$\begin{aligned}
(\Omega^{\text{rank}})^{**}(\mathbf{W}) &= \sup_{\mathbf{V}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{W}\rangle - \max_{g=1\ldots\mathcal{G}} \psi_g^*(\mathbf{V}) \\
&= \sup_{\mathbf{V}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{W}\rangle - \max_{\substack{\beta_g\geq 0 \\ \sum_g \beta_g = 1}} \sum_g \beta_g \psi^*(\mathbf{V}) \\
&= \sup_{\mathbf{V}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{W}\rangle + \min_{\substack{\beta_g\geq 0 \\ \sum_g \beta_g = 1}} \left(-\sum_g \beta_g \psi^*(\mathbf{V}) \right) \\
&= \sup_{\mathbf{V}\in\mathbb{R}^{d\times k}} \min_{\substack{\beta_g\geq 0 \\ \sum_g \beta_g = 1}} \langle \mathbf{V}, \mathbf{W}\rangle - \sum_g \beta_g \psi^*(\mathbf{V}) \\
&= \min_{\substack{\beta_g\geq 0 \\ \sum_g \beta_g = 1}} \sup_{\mathbf{V}\in\mathbb{R}^{d\times k}} \langle \mathbf{V}, \mathbf{W}\rangle - \sum_g \beta_g \psi^*(\mathbf{V}) \\
&= \min_{\substack{\beta_g\geq 0 \\ \sum_g \beta_g = 1}} \left(\sum_g \beta_g \psi_g^* \right)^* (\mathbf{W})
\end{aligned}$$

Now we use the fact that the conjugate of a sum is the infimal convolution of conjugates Hiriart-Urruty and Lemarechal (1993).

For any $\mathbf{W}$ in C we have

$$(\Omega^{\text{rank}})^{**}(\mathbf{W}) = \min_{\substack{\beta_g\geq 0 \\ \sum_g \beta_g = 1}} \quad \inf_{\substack{\mathbf{H}_g\in\mathbb{R}^{d\times k} \\ \sum_g \mathbf{H}_g = \mathbf{W}}} \sum_g (\beta_g \psi_g^*)^*(\mathbf{H}_g) \tag{2.8}$$

$$= \min_{\substack{\beta_g \geq 0 \\ \sum_g \beta_g = 1}} \inf_{\substack{\mathbf{H}_g \in \mathbb{R}^{d \times k} \\ \sum_g \mathbf{H}_g = \mathbf{W}}} \sum_g \beta_g \psi_g^{**} \left(\frac{\mathbf{H}_g}{\beta_g} \right) \tag{2.9}$$

$$= \min_{\substack{\beta_g \geq 0 \\ \sum_g \beta_g = 1}} \inf_{\substack{\sum_g \mathbf{H}_g = \mathbf{W} \\ \sigma_{\max}(\mathbf{H}_g/\beta_g) \leq 1}} \sum_g \beta_g \alpha_g \| \frac{\mathbf{H}_g}{\beta_g} \|_{\sigma,1} + \delta_g \left(\frac{\mathbf{H}_g}{\beta_g} \right) \tag{2.10}$$

$$= \min_{\substack{\beta_g \geq 0 \\ \sum_g \beta_g = 1}} \inf_{\substack{\sum_g \mathbf{H}_g = \mathbf{W} \\ \sigma_{\max}(\mathbf{H}_g) \leq \beta_g}} \sum_g \alpha_g \| \mathbf{H}_g \|_{\sigma,1} + \delta_g(\mathbf{H}_g) \tag{2.11}$$

$$= \inf_{\substack{\sum_g \mathbf{H}_g = \mathbf{W} \\ \sum_g \sigma_{\max}(\mathbf{H}_g) \leq 1}} \sum_g \alpha_g \| \mathbf{H}_g \|_{\sigma,1} + \delta_g(\mathbf{H}_g) \tag{2.12}$$

$$= \inf_{\substack{(\mathbf{W}_g)_g \in E_{\mathcal{G}} \\ \sum_g i_g(\mathbf{W}_g) = \mathbf{W} \\ \sum_g \sigma_{\max}(i_g(\mathbf{W}_g)) \leq 1}} \sum_g \alpha_g \| i_g(\mathbf{W}_g) \|_{\sigma,1} \tag{2.13}$$

$$= \Omega_{\mathcal{G}}(\mathbf{W}). \tag{2.14}$$

The equality (2.9) is based on a property of Fenchel conjugate. □

In this section we provide a complete convex analysis of the group Schatten norm: computation of subgradient, dual norm, polar norm, and convex conjugate, by using standard results of convex analysis Rockafellar (1970), Hiriart-Urruty and Lemarechal (1993).

Let us define the unit-ball of the group p-Schatten norm $\mathcal{B} = \{\mathbf{W} : \Omega_{\mathcal{G}}(\mathbf{W}) \leq 1\}$. By construction, the norm is the gauge of its unit-ball

$$\Omega_{\mathcal{G}}(\mathbf{W}) = \inf\{\lambda \geq 0 : \mathbf{W} \in \lambda \mathcal{B}\}. \tag{2.15}$$

The dual norm of $\Omega_{\mathcal{G}}$ is defined as

$$\Omega_{\mathcal{G}}^{\circ}(\mathbf{V}) := \sup_{\Omega_{\mathcal{G}}(\mathbf{W}) \leq 1} \langle \mathbf{W}, \mathbf{V} \rangle. \tag{2.16}$$

This is the best norm satisfying the Cauchy-Schwarz inequality $\langle \mathbf{W}, \mathbf{V} \rangle \leq \Omega_{\mathcal{G}}(\mathbf{W}) \Omega_{\mathcal{G}}^{\circ}(\mathbf{V})$. The dual norm of p-Schatten norm is $(\| \cdot \|_{\sigma,p})^{\circ} = \| \cdot \|_{\sigma,q}$, for any $p \geq 1$, and q s.t. $1/p + 1/q = 1$.

Proposition 2. (Dual norm) The dual norm $\Omega_{\mathcal{G}}^{\circ}$, defined by (2.16), satisfies

$$\Omega_{\mathcal{G}}^{\circ}(\mathbf{V}) = \max_{g \in \mathcal{G}} \frac{1}{\alpha_g} \| \Pi_g(\mathbf{V}) \|_{\sigma,q}.$$

Proof. of proposition 2. The unit ball of $\Omega_{\mathcal{G}}$ can be written as

$$\mathcal{B} = \left\{ \mathbf{W} \in \mathbb{R}^{d\times k} \,\middle|\, \Omega_{\mathcal{G}}(\mathbf{W}) \leq 1 \right\} = \left\{ \mathcal{A}_g((\mathbf{V}_g)_g) \,\middle|\, (\mathbf{V}_g)_g \in E_{\mathcal{G}},\, \sum_g \alpha_g \|\mathbf{V}_g\|_{\sigma,p} \leq 1 \right\} \tag{2.17}$$

We have

$$\begin{aligned}
\Omega^{\circ}_{\mathcal{G}}(\mathbf{V}) &= \sup_{\substack{(\mathbf{W}_g)_g \in E_{\mathcal{G}} \\ \mathbf{W} = \mathcal{A}((\mathbf{W}_g)_g) \\ \sum_g \alpha_g \|\mathbf{W}_g\|_{\sigma,p} \leq 1}} \langle \mathbf{W}, \mathbf{V} \rangle \\
&= \sup_{\substack{(\mathbf{W}_g)_g \in E_{\mathcal{G}} \\ \sum_g \alpha_g \|\mathbf{W}_g\|_{\sigma,p} \leq 1}} \langle \mathcal{A}((\mathbf{W}_g)_g), \mathbf{V} \rangle \\
&= \sup_{\substack{(\mathbf{W}_g)_g \in E_{\mathcal{G}} \\ \sum_g \alpha_g \|\mathbf{W}_g\|_{\sigma,p} \leq 1}} \sum_g \langle i_g(\mathbf{W}_g), \mathbf{V} \rangle \\
&= \sup_{\substack{(\mathbf{W}_g)_g \in E_{\mathcal{G}} \\ \sum_g \alpha_g \|\mathbf{W}_g\|_{\sigma,p} \leq 1}} \sum_g \langle \mathbf{W}_g, \Pi_g(\mathbf{V}) \rangle \\
&= \sup_{\substack{\sum_g \beta_g \leq 1 \\ \beta_g \geq 0}} \sum_g \sup_{\substack{\mathbf{W}_g \in \mathbb{R}^{|\mathcal{P}_g| \times |\mathcal{Y}_g|} \\ \|\mathbf{W}_g\|_{\sigma,p} \leq \beta_g/\alpha_g}} \langle \mathbf{W}_g, \Pi_g(\mathbf{V}) \rangle.
\end{aligned}$$

We recognize that the sup over $\mathbf{W}_g$ in the previous expression is the dual norm of $\|\cdot\|_{\sigma,p}$, which is $\|\cdot\|_{\sigma,q}$. Thus we have

$$\sup_{\substack{\mathbf{W}_g \in \mathbb{R}^{|\mathcal{P}_g| \times |\mathcal{Y}_g|} \\ \|\mathbf{W}_g\|_{\sigma,p} \leq \boldsymbol{\beta}_g/\alpha_g}} \langle \mathbf{W}_g, \Pi_g(\mathbf{V}) \rangle = \frac{\boldsymbol{\beta}_g}{\alpha_g} \|\Pi_g(\mathbf{V}_g)\|_{\sigma,q}.$$

This results in

$$\Omega^{\circ}_{\mathcal{G}}(\mathbf{V}) = \sup_{\substack{\sum_g \beta_g \leq 1 \\ \beta_g \geq 0}} \sum_g \frac{\boldsymbol{\beta}_g}{\alpha_g} \|\Pi_g(\mathbf{V}_g)\|_{\sigma,q} = \max_g \frac{1}{\alpha_g} \|\Pi_g(\mathbf{V}_g)\|_{\sigma,q}$$

which is the desired expression. □

The duality of norms is perfect: the dual norm of the dual norm is the primal norm itself. This gives the following variational formulation of the group p-Schatten norm.

Proposition 3. (Group p-Schatten norm as a support function) The group p-Schatten norm $\Omega_{\mathcal{G}}$ satisfies

$$\Omega_{\mathcal{G}}(\mathbf{W}) = \max_{\|\Pi_g(\mathbf{V})\|_{\sigma,q} \leq \alpha_g,\ g \in \mathcal{G}} \langle \mathbf{V}, \mathbf{W} \rangle. \tag{2.18}$$

We also have that the argmax of (2.18) coincide with $\partial\Omega_{\mathcal{G}}(\mathbf{W})$ by standard subdifferential calculus rules.

Proof. of proposition 3. The dual norm of $\Omega_{\mathcal{G}}^{\circ}$ is $\Omega_{\mathcal{G}}$ ((Rockafellar, 1970, Ch.15), (Hiriart-Urruty and Lemarechal, 1996, C.3.2)). Inverting the role of $\Omega_{\mathcal{G}}^{\circ}$ and $\Omega_{\mathcal{G}}$ in (2.16) gives the desired expression with the help of Proposition 2. □

The conjugate function of $\Omega_{\mathcal{G}}$ is defined as

$$\Omega_{\mathcal{G}}^{*}(\mathbf{V}) := \sup_{\mathbf{W}\in\mathbb{R}^{d\times k}} \langle \mathbf{W}, \mathbf{V}\rangle - \Omega_{\mathcal{G}}(\mathbf{W}) \tag{2.19}$$

This is the best convex function satisfying $\langle \mathbf{W}, \mathbf{V}\rangle \leq \Omega_{\mathcal{G}}(\mathbf{W}) + \Omega_{\mathcal{G}}^{*}(\mathbf{V})$. The conjugate of the p-Schatten norm is the indicator function of the dual ball, *i.e.* $(\|\cdot\|_{\sigma,p})^{*} = \delta_{\{X:\, \|X\|_{\sigma,q}\}\leq 1}$. We generalize this result in the next proposition.

Proposition 4 (Conjugate function). The conjugate $\Omega_{\mathcal{G}}^{*}$ satisfies

$$\Omega_{\mathcal{G}}^{*}(\mathbf{V}) = \begin{cases} 0 & \|\sigma(P_g(\mathbf{V}))\|_q \leq \alpha_g \text{ for all } g \in \mathcal{G} \\ +\infty & \text{otherwise} \end{cases}$$

Proof. (*Of proposition 4.*) This proposition can be proved directly; here we get it as is a direct corollary of Proposition 3. In general, it is easy to see on the definition that the conjugate of the indicator function of a closed convex set B is the support function of B. We deduce that the conjugate of support function of B is the indicator-function of B. Observe that Proposition 3 reads that $\Omega_{\mathcal{G}}$ is the support-function of the set of $= \left\{\mathbf{V} \in \mathbb{R}^{d\times k} \,\middle|\, \|\Pi_g(\mathbf{V})\|_{\sigma,q} \leq \alpha_g \text{ for all } g \in \mathcal{G}\right\}$. Its conjugate is then the indicator of B. □

2.5 Algorithms for learning with group p-Schatten norm

2.5.1 Group p-Schatten norm as regularization penalty

Though defined implicitly through a minimization problem, the group Schatten norm can be used efficiently as regularizer for learning optimization problems.

Let us consider a general matrix optimization problem

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times k}} \quad F(\mathbf{W}) + \lambda\,\Omega_{\mathcal{G}}(\mathbf{W}), \tag{2.20}$$

where F is convex and smooth and $\Omega_{\mathcal{G}}$ is the group Schatten norm. Many classical learning problems can be formulated in this way.

Applying directly standard optimization algorithms on (2.20), such as proximal gradient Nesterov (2004) or conditional gradient Nemirovski and Yudin (1983), would require computing operators related to $\Omega_{\mathcal{G}}$, such as proximal operator (Hiriart-Urruty and Lemarechal, 1996, Section XV.4) or linear minimization oracle Cox et al. (2014).

When dealing with overlapping groups, instead of solving directly (2.20), we avoid the question of computing the proximal operator or linear minimization oracle of $\Omega_{\mathcal{G}}$ by redefining the problem on the space $E_{\mathcal{G}}$.

Let us add some structure. The vector space $E_{\mathcal{G}}$ is naturally equipped with the standard scalar product

$$\langle (\mathbf{W}_g)_g, (\mathbf{V}_g)_g \rangle_{E_{\mathcal{G}}} := \sum_{g \in \mathcal{G}} \langle \mathbf{W}_g, \mathbf{V}_g \rangle_{\mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}}$$

and with the induced euclidean norm

$$\|(\mathbf{W}_g)_g\|_2 := \sqrt{\sum_{g \in \mathcal{G}} \|\mathbf{W}_g\|_2^2},$$

where under the square root there is the euclidean norm defined on $\mathbb{R}^{|\mathcal{D}_g| \times |\mathcal{Y}_g|}$. When the meaning is evident, we write $\langle \cdot, \cdot \rangle$ for any scalar product. $E_{\mathcal{G}}$ is also equipped with the norm $\Omega_{\oplus}$ defined by Eq. (2.1).

Proposition 5. The problem (2.20) can be rewritten with the $(\mathbf{W}_g)_g \in E_{\mathcal{G}}$ as

$$\min_{(\mathbf{W}_g)_g \in E_{\mathcal{G}}} \quad F(\mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g)) + \lambda \sum_{g \in \mathcal{G}} \alpha_g \|\mathbf{W}_g\|_{\sigma,p}. \tag{2.21}$$

Any solution $(\mathbf{W}_g^\star)_g$ of (2.21) yields a solution $\mathbf{W}^\star = \mathcal{A}_{\mathcal{G}}((\mathbf{W}_g^\star)_g)$ of (2.20). Also, any solution of (2.20) yields a solution of (2.21) via any decomposition optimizing (2.2).

Proof. (Of Proposition 5) To prove this we observe that the constraint $\mathbf{W} = \mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g)$ disappears because the minimization is also with respect to $\mathbf{W}$:

$$\begin{aligned}
&\min_{\mathbf{W} \in \mathbb{R}^{d \times k}} \quad F(\mathbf{W}) + \lambda\, \Omega_{\mathcal{G}}(\mathbf{W}) \\
&= \min_{\mathbf{W} \in \mathbb{R}^{d \times k}} \quad F(\mathbf{W}) + \lambda \inf_{\substack{(\mathbf{W}_g)_g \in E_{\mathcal{G}} \\ \text{s.t. } \mathbf{W} = \mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g)}} \Omega_{\oplus}((\mathbf{W}_g)_g) \\
&= \min_{(\mathbf{W}_g)_g \in E_{\mathcal{G}}} \quad F(\mathcal{A}_{\mathcal{G}}((\mathbf{W}_g)_g)) + \lambda \Omega_{\oplus}((\mathbf{W}_g)_g)
\end{aligned}$$

□

2.5.2 (Accelerated) proximal-gradient algorithm

Proximal-gradient is an efficient state-of-the-art algorithm to solve the problem (2.21). The key computation at each iteration t is

$$\text{prox}_{\beta^{(t)}\Omega_\oplus}\Big((\mathbf{W}_g^{(t)})_g - \delta^{(t)}\nabla(F\circ\mathcal{A}_\mathcal{G})\big((\mathbf{W}_g^{(t)})_g\big)\Big).$$

where $\delta^{(t)}$ is the stepsize chosen according to some rules and the prox operator, for a generic vector space E, a function $\Omega\colon E\to\mathbb{R}$ and parameter $\beta>0$, is defined as

$$\text{prox}_{\beta\Omega}(\mathbf{W}) \coloneqq \underset{\mathbf{Z}\in E}{\text{argmin}}\ \Omega(\mathbf{Z}) + \frac{1}{2\beta}\|\mathbf{W}-\mathbf{Z}\|_2^2\ . \tag{2.22}$$

For nuclear norm $\|\cdot\|_{\sigma,1}$, we can derive the explicit expression of the prox operator. Given a matrix $\mathbf{W}\in\mathbb{R}^{d\times k}$ with the SVD decomposition

$$\mathbf{W}=\mathbf{U}\mathbf{S}\mathbf{V}^\top,\quad \text{where } \mathbf{S}=\text{diag}(\{s_i\}_{1\le i\le r})\ ,$$

we have

$$\text{prox}_{\beta\|\cdot\|_{\sigma,1}}(\mathbf{W}) = \mathbf{U}D_\beta(\mathbf{S})\mathbf{V}^\top, \tag{2.23}$$

where D_β is the soft thresholding operator

$$D_\beta(\mathbf{S}) = \text{diag}(\{\max\{s_i-\beta,0\}\}_{1\le i\le r}).$$

So, on the space $E_\mathcal{G}$, the prox operator for the norm $\Omega_\oplus$, defined as

$$\text{prox}_{\beta\Omega_\oplus}((\mathbf{W}_g)_g) \coloneqq \underset{(\mathbf{V}_g)_g\in E_\mathcal{G}}{\text{argmin}}\quad \Omega_\oplus((\mathbf{V}_g)_g) + \frac{1}{2\beta}\|(\mathbf{W}_g)_g-(\mathbf{V}_g)_g\|_2^2\ ,$$

can be explicitly computed using the SVD decomposition of each group:

$$\text{prox}_{\beta\Omega_\oplus}((\mathbf{W}_g)_g) = \Big(\mathbf{U}_g D_\beta(\mathbf{S}_g){\mathbf{V}_g}^\top\Big)_{g\in\mathcal{G},} \tag{2.24}$$

where $\mathbf{U}_g$, $\mathbf{S}_g$, and $\mathbf{V}_g$ are the SVD decomposition matrices of $\mathbf{W}_g$.

Proof. (Of equation (2.24)) Recall that $\text{prox}_{\beta\Omega_\oplus}$ is defined on $E_\mathcal{G}$ and returns values in $E_\mathcal{G}$. From the

definition, we have

$$
\begin{aligned}
\operatorname{prox}_{\beta\Omega_\oplus}((\mathbf{W}_g)_g) &= \underset{(\mathbf{Z}_g)_g \in E_{\mathcal{G}}}{\operatorname{argmin}} \quad \Omega_\oplus((\mathbf{Z}_g)_g) + \frac{1}{2\beta}\|(\mathbf{W}_g)_g - (\mathbf{Z}_g)_g\|_2^2 \\
&= \underset{(\mathbf{Z}_g)_g \in E_{\mathcal{G}}}{\operatorname{argmin}} \quad \sum_{g\in\mathcal{G}} \alpha_g \|\mathbf{Z}_g\|_{\sigma,p} + \frac{1}{2\beta}\sum_{g\in\mathcal{G}} \|\mathbf{W}_g - \mathbf{Z}_g\|_2^2 \\
&= \underset{(\mathbf{Z}_g)_g \in E_{\mathcal{G}}}{\operatorname{argmin}} \quad \sum_{g\in\mathcal{G}} \left(\alpha_g \|\mathbf{Z}_g\|_{\sigma,p} + \frac{1}{2\beta}\|\mathbf{W}_g - \mathbf{Z}_g\|_2^2\right) \\
&= \left(\underset{\mathbf{Z}_g \in \mathbb{R}^{|\mathcal{D}_g|\times|\mathcal{Y}_g|}}{\operatorname{argmin}} \quad \alpha_g \|\mathbf{Z}_g\|_{\sigma,p} + \frac{1}{2\beta}\|\mathbf{W}_g - \mathbf{Z}_g\|_2^2\right)_{g\in\mathcal{G}} \\
&= \left(\operatorname{prox}_{\beta\alpha_g\|\cdot\|_{\sigma,p}}(\mathbf{W}_g)\right)_{g\in\mathcal{G}}.
\end{aligned}
$$

The result now follows by Eq. (2.23). □

2.5.3 Composite conditional gradient

Composite conditional gradient Dudik et al. (2012) is an efficient state-of-the-art algorithm to solve the problem (2.20), which relies on a linear minimization operator. The linear minimization operator (LMO) for a generic vector space E and a norm $\|\cdot\|$ is defined as

$$
\operatorname{LMO}_{\|\cdot\|}(\mathbf{W}) := \underset{\mathbf{Z}\in E \mid \|\mathbf{Z}\|\leq 1}{\operatorname{argmax}} \langle \mathbf{Z}, \mathbf{W}\rangle,
$$

where $\mathbf{W}$ is in a vector space E with scalar product $\langle\cdot,\cdot\rangle$.

For nuclear norm $\|\cdot\|_{\sigma,1}$, we know the explicit expression of the LMO, described by Dudik et al. (2012)

$$
\underset{\substack{\mathbf{Z}\in\mathbb{R}^{d\times k} \\ \|\mathbf{Z}\|_{\sigma,1}\leq 1}}{\operatorname{argmax}} \langle \mathbf{Z},\, \mathbf{W}\rangle = \mathbf{u}\mathbf{v}^\top,
$$

where $\mathbf{u}$ and $\mathbf{v}$ are the top singular vectors of the matrix $\mathbf{W}$.

The key computation, to solve (2.21), at each iteration of composite conditional gradient is

$$
\operatorname{LMO}_{\Omega_{\mathcal{G}}}\Big(-\nabla(F\circ\mathcal{A}_{\mathcal{G}})\big((\mathbf{W}_g^{(t)})_g\big)\Big).
$$

So, for all $(\mathbf{W}_g)_g \in E_{\mathcal{G}}$, the LMO

$$\mathrm{LMO}_{\Omega_\oplus}((\mathbf{W}_g)_g) := \underset{\substack{(\mathbf{Z}_g)_g \in E_{\mathcal{G}} \\ \Omega_\oplus((\mathbf{Z}_g)_g) \leq 1}}{\operatorname{argmax}} \langle (\mathbf{Z}_g)_g, (\mathbf{W}_g)_g \rangle$$

for the norm $\Omega_\oplus$ can be explicitly computed by finding the maximum singular vectors of the matrix of each group. $\mathrm{LMO}_{\Omega_\oplus}$ returns a tuple of zero matrices, except for the group with index $\hat{g}$,

$$\mathrm{LMO}_{\Omega_\oplus}((\mathbf{W}_g)_g) = \left(\mathbf{0}, \ldots, \mathbf{0}, \tfrac{1}{\alpha_{\hat{g}}} \mathbf{u}_{\hat{g}} \mathbf{v}_{\hat{g}}^\top, \mathbf{0}, \ldots, \mathbf{0}\right), \tag{2.25}$$

where, for all $g \in \mathcal{G}$, $\mathbf{u}_g$ and $\mathbf{v}_g^\top$ are the top singular vectors of the matrix $\mathbf{W}_g$, and

$$\hat{g} = \underset{g \in \mathcal{G}}{\operatorname{argmax}} \quad \tfrac{1}{\alpha_g} \left(\mathbf{u}_g^\top \mathbf{W}_g \mathbf{v}_g\right).$$

We observe also that $\mathrm{LMO}_{\Omega_\oplus}((\mathbf{W}_g)_g)$ is the point where the dual norm (2.16) of $\Omega_\oplus$ takes the maximum.

Proof. (Of equation (2.25)) We derive the LMO for the norm $\Omega_\oplus$ on the space $E_{\mathcal{G}}$. Recall that $\alpha_g > 0$. First, we note that

$$\max_{\substack{\mathbf{z} \in \mathbb{R}^{|\mathcal{G}|} \\ \sum_g \alpha_g |z_g| \leq 1}} \sum_g z_g w_g = \max_g \left| \frac{w_g}{\alpha_g} \right|,$$

because

$$\max_{\substack{\mathbf{z} \in \mathbb{R}^{|\mathcal{G}|} \\ \sum_g \alpha_g |z_g| \leq 1}} \sum_g z_g w_g = \max_{\substack{\mathbf{y} \in \mathbb{R}^{|\mathcal{G}|} \\ \sum_g |y_g| \leq 1}} \sum_g \frac{y_g}{\alpha_g} w_g = \max_{\|\mathbf{y}\|_1 \leq 1} \sum_g y_g \frac{w_g}{\alpha_g}.$$

The maximum of the last equation is attained for an $\mathbf{y} \in \{\pm \mathbf{e}_g\}$ in the canonical overcomplete base of $\mathbb{R}^{|\mathcal{G}|}$. Therefore,

$$\begin{aligned}
\mathrm{LMO}_{\Omega_\oplus}((\mathbf{W}_g)_g) &= \max_{\substack{(\mathbf{Z}_g)_g \in E_{\mathcal{G}} \\ \Omega_\oplus((\mathbf{Z}_g)_g) \leq 1}} \langle (\mathbf{Z}_g)_g, (\mathbf{W}_g)_g \rangle \\
&= \max_{\sum_g \alpha_g \|\mathbf{Z}_g\|_{\sigma,1} \leq 1} \langle (\mathbf{Z}_g)_g, (\mathbf{W}_g)_g \rangle \\
&= \max_{\sum_g \alpha_g \|\mathbf{Z}_g\|_{\sigma,1} \leq 1} \sum_g \langle \mathbf{Z}_g, \mathbf{W}_g \rangle \\
&= \max_{\sum_g \|\mathbf{Y}_g\|_{\sigma,1} \leq 1} \sum_g \langle \mathbf{Y}_g, \tfrac{1}{\alpha_g} \mathbf{W}_g \rangle
\end{aligned}$$

$$
\begin{aligned}
&= \max_{\sum_g \beta_g \leq 1} \sum_g \max_{\|\mathbf{Y}_g\|_{\sigma,1} \leq \beta_g} \langle \mathbf{Y}_g, \tfrac{1}{\alpha_g} \mathbf{W}_g \rangle \\
&= \max_{\sum_g \beta_g \leq 1} \sum_g \tfrac{\beta_g}{\alpha_g} \left(\mathbf{u}_g^\top \mathbf{W}_g \mathbf{v}_g \right) \\
&= \max_g \tfrac{1}{\alpha_g} \left(\mathbf{u}_g^\top \mathbf{W}_g \mathbf{v}_g \right),
\end{aligned}
$$

so the maximum is attained at an index $\hat{g}$, and the matrices $\mathbf{Z}_g$ corresponding to other indices are zero. □

2.6 Illustrations

In this section we illustrate the recovery of a matrix $\mathbf{L} \in \mathbb{R}^{d \times k}$ given its noisy version $\mathbf{A}_\sigma = \mathbf{L} + \varepsilon_\sigma$. Each entry of the matrix ε_σ is sampled independently from a Gaussian with mean 0 and standard deviation σ. In our experiments we consider several noise levels σ. $\mathbf{L}$ is the sum of matrices of rank 2 over each of the groups. The matrix $\mathbf{L}$ is normalized to have mean of 0 and standard deviation of 1. This choice of normalization is suitable to replicate the experience on true datasets, possibly with unknown structure.

As visible in Fig. 2.1, we chose the overlapping groups $\{1...1575\} \times \{1...1500\}$, $\{1576...2100\} \times \{1...2000\}$, $\{1...2100\} \times \{1501...2000\}$, $\{1156...2100\} \times \{1101...2000\}$, $\{772...1021\} \times \{1290...1499\}$, and added other small sparse groups $\{1...250\} \times \{542...751\}$, $\{272...521\} \times \{166...375\}$, $\{345...594\} \times \{619...828\}$, $\{735...984\} \times \{965...1174\}$, where each group is indicated with $\{$row indices$\} \times \{$column indices$\}$. We added also a “grid shaped" group, visible at the left part of the matrix. On this group we generated a sinus function.

In a simple generalization of the group lasso norm to matrices, the split would divide only columns into groups, and keep all the rows together. Our division allows more flexibility in splitting matrix entries into groups, including the group structure of our example. Thus, our setup allows more general applications, which we will discuss in Section 2.7.

To recover $\mathbf{L}$, we solve the regularized problem

$$
\min_{\mathbf{W} \in \mathbb{R}^{d \times k}} \frac{1}{2dk} \|M_{\mathcal{I}}(\mathbf{W} - \mathbf{A}_\sigma)\|^2 + \lambda \Omega_{\mathcal{G}}(\mathbf{W}),
$$

where $\mathcal{I}$ is a set of indices that covers 10% of the matrix and $M_{\mathcal{I}}$ is a linear operator that selects the entries of the set $\mathcal{I}$.

The coefficients for the group Schatten norm $\Omega_{\mathcal{G}}$ were set to $\alpha_g = 1$. We also experimented with choosing the coefficients α_g based on the size of each corresponding group, but did not observe consistent improvements in the matrix recovery, so we leave adaptive choices of α_g open for future study.

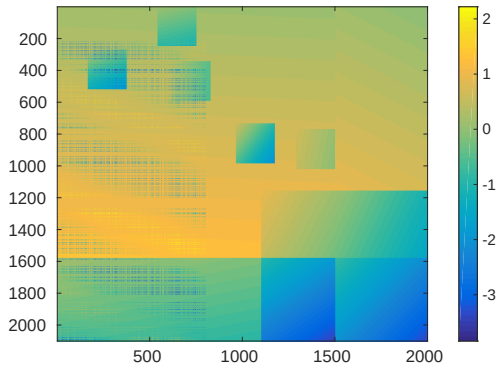

Figure 2.1: Initial matrix $\mathbf{L}$ to recover. It is generated as the sum of matrices of rank two, each defined over its respective group of indices. The groups overlap.

For each value of σ, we tuned the regularization parameter λ to minimize the error and report the best performance. The quality of the resulting solution $\widehat{\mathbf{W}}$ is evaluated according to the recovery error $\frac{1}{dk}\|\widehat{\mathbf{W}} - \mathbf{L}\|^2$. In Fig. 2.2 we plot the noisy matrix $\mathbf{A}_\sigma$ with $\sigma = 0.2$, where only 10% of its entries are observed, as well as the solution $\widehat{\mathbf{W}}$. The all-zeros solution would achieve the recovery error of 1. Our solution recovers $\mathbf{A}_{0.2}$ with an error of 0.0051.

2.7 Proposed applications: Initial steps

We have seen that the regularization with group nuclear norm forces the solution of an optimization problem to be (1) low rank on each group and (2) nonzero on only few groups. In this section, we overview possible applications that benefit from these properties, including robust models for collaborative filtering, image classification, feature aggregation and database compression.

We focus on the group nuclear norm instead of the more general group Schatten norm, because we are interested in low-rank solutions. We will highlight the characteristics that make the group nuclear norm useful in applications when the groups can overlap.

2.7.1 Multiclass classification

We describe now in short the linear model for classification with k classes. We call "model" a matrix $\mathbf{W} \in \mathbb{R}^{d\times k}$, "example" the vector of features $\mathbf{x} \in \mathbb{R}^d$ and "response" an index $y \in \{1 \dots k\}$ that indicates the class. The dataset to learn $\mathbf{W}$ consists of N pairs $(\mathbf{x}^{(j)}, y^{(j)})$, $j = 1, \dots, N$. We define the

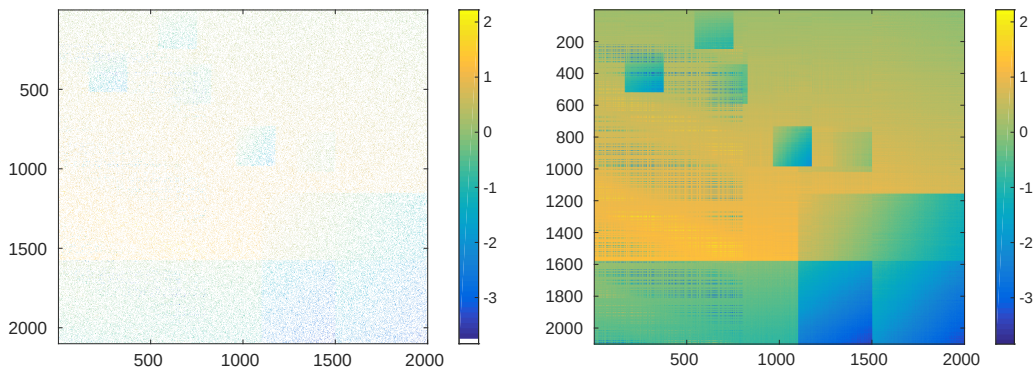

Figure 2.2: **(Left)** Noisy matrix $\mathbf{A}_{0.2}$, where only 10% of matrix entries are observed. Unknown entries are white. **(Right)** Solution matrix $\widehat{\mathbf{W}}$ recovered with $\lambda = 3.6 \cdot 10^{-6}$. This gives an error of 0.0067. We see that both the big overlapping groups and the small ones are recovered. The group on the left side, which consists of a grid with different spacing in rows and columns, and that has sinusoidal values, is also well recovered.

classifier

$$\hat{y}(\mathbf{x}) := \underset{i=1\ldots k}{\operatorname{argmax}} (\mathbf{W}^\top \mathbf{x})_i \ . \tag{2.26}$$

The predicted class of a generic $\mathbf{x}$ is then $\hat{y}(\mathbf{x})$.

The nuclear norm regularization for support vector machines has already been proposed for classification Amit et al. (2007). The small rank of $\mathbf{W}$ is justified by the fact that the examples $\mathbf{x}^{(j)}$ are, in practice, embedded to a low dimension subspace of $\mathbb{R}^d$.

A toy example of a hierarchically structured dataset We build a toy example to illustrate how to define the groups for a hierarchical dataset. In addition, we will show how to use our groupings to represent attributes of classes, which are not easily expressed in a hierarchy. Our examples are motivated by the dataset of images ImageNet Deng et al. (2009), where the image classes are within a tree-structured taxonomy. Apart from its position in the taxonomy, each class can have zero, one or more attributes that describe it, and the number of attributes may differ from class to class.

To make this more concrete, consider the taxonomy in Fig. 2.3, defined over four ground classes. We begin by defining the singleton groups of classes corresponding to the four ground classes: $\mathcal{Y}_1 = \{1\}$ for `catamaran`, $\mathcal{Y}_2 = \{2\}$ for `trimaran`, $\mathcal{Y}_3 = \{3\}$ for `catboat`, $\mathcal{Y}_4 = \{4\}$, for `galleon`. We then create additional groups corresponding to the inner nodes of the taxonomy tree: $\mathcal{Y}_5 = \{1, 2, 3\}$ for `sailboat` and $\mathcal{Y}_6 = \{1, 2, 3, 4\}$, for `sailing vessel`. And finally, we add a group for the attribute `made of wood` $\mathcal{Y}_7 = \{3, 4\}$, present in the classes `catboat` and for `galleon`. These groups split

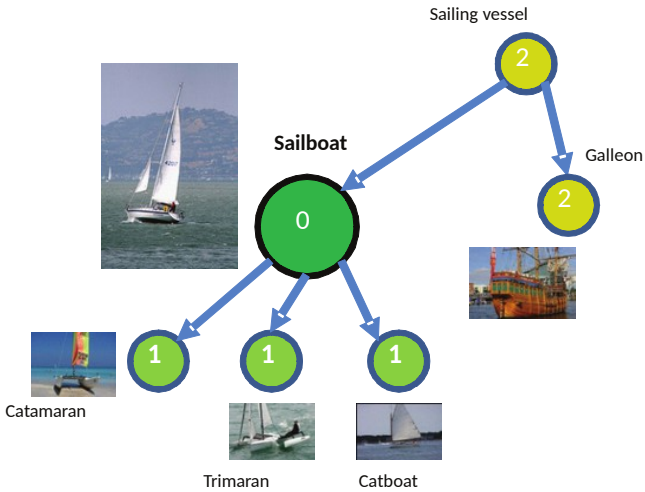

Figure 2.3: Hierarchical organisation of images in ImageNet dataset. Figure adapted from Deng et al. (2010). We see that the classes `catamaran`, `trimaran`, `catboat`, `sailboat`, `galleon`, `sailing vessel` are organized in a tree graph.

only the columns of $\mathbf{W}$, *i.e.* $\mathcal{D}_1 = \mathcal{D}_2 = \ldots \mathcal{D}_9 = \{1, \ldots, d\}$. We notice that the group 7 is defined across the hierarchy: potentially the same attribute could be added to any class, independently of the tree structure.

Local features, constellation models In some applications of image classification, the pictures are divided into "sectors." We can imagine that a physical object has a different meaning if its position is in the center of the picture or below or on the top.

Each sector of the image has then its features and the object to recognize is placed only into one of the sectors of the image.

For our toy example, we split the image into 4 sectors: top-left, top-right, bottom-left, bottom-right. The feature vectors $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, each of size a, are extracted from each sector and then concatenated into a single vector $\mathbf{x} = [\mathbf{x}_1\, \mathbf{x}_2\, \mathbf{x}_3\, \mathbf{x}_4] \in \mathbb{R}^d$. We then build 4 groups: $\mathcal{D}_1 = \{1, \ldots, a\}$, $\mathcal{D}_2 = \{a + 1, \ldots, 2a\}$, $\mathcal{D}_3 = \{2a + 1, \ldots, 3a\}$, $\mathcal{D}_4 = \{3a + 1, \ldots, 4a\}$ and $\mathcal{Y}_1 = \mathcal{Y}_2 = \mathcal{Y}_3 = \mathcal{Y}_4 = \{1, \ldots, k\}$.

As we defined a group for each sector of image, the resulting model favors sparsity over sectors and a low-rank model in each sector. So, for instance, if one of the sectors always contains the relevant object, our model will be non-zero only on the parameters related to that sector and within that group will be of low rank.

Some interesting data for this application can be found in Rogez et al. (2014), where the images are

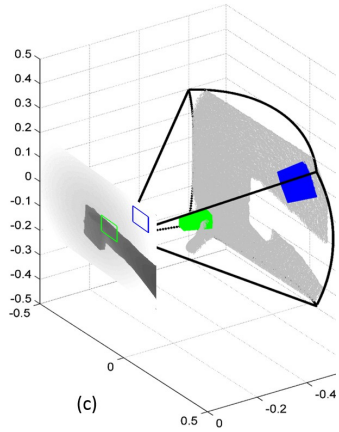

Figure 2.4: We see the observable volume that is projected into the depth image, which is dark gray for close objects, *e.g.* a hand grasping something, and light grey for far objects. Picture from Rogez et al. (2014)

taken from the point of view of an egocentric camera placed at the chest of the person that observes. These data are used to classify the pose of hands while they interact with objects. The data images include also depth maps, see Figure 2.7.1, that can be used to create more groups and enhance the classification. In fact, the distance from the camera is related to the activity of the hand. So, we propose to discretize the distance into three layers, and to build four groups per layer. Thus, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$ are the features extracted in the four sectors, if they contain an object at short distance, $\mathbf{x}_5$, $\mathbf{x}_6$, $\mathbf{x}_7$, $\mathbf{x}_8$ for the medium distance, and $\mathbf{x}_9$, $\mathbf{x}_{10}$, $\mathbf{x}_{11}$, $\mathbf{x}_{12}$ when the object is far from the camera. With this definition, many features will be zero. For example, if the object(s) in the top-left sector are far from the camera, then $\mathbf{x}_1 = \mathbf{x}_5 = \mathbf{0}$ and $\mathbf{x}_9$ will be the only top-sector layer with nonzero features.

A similar approach can be used to build groups for constellation models. For instance, consider image recognition for faces. Here the sectors of the image are the mouth, the nose, one eye, etc., each of them giving a separate vector of features that are concatenated similarly as in the previous example.

2.7.2 Collaborative filtering with attributes

Collaborative filtering, when modeled as matrix completion, can be implemented using algorithms that rely on the nuclear norm regularization Candès and Plan (2010); Srebro et al. (2004); Lafond et al. (2014); Hummel et al. (2007). Here we show how it can benefit from the group nuclear norm.

We consider the regularized matrix completion problem of the form

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times k}} \quad \frac{1}{N}\sum_{(i,j)\in\mathcal{I}}(\mathbf{W}_{ij}-\mathbf{X}_{ij})^2 \quad + \quad \lambda\Omega_{\mathcal{G}}(\mathbf{W}),$$

where the data are represented with a sparse matrix $\mathbf{X}$, each column representing the item j, each row a customer i, and the entries (i,j) corresponding to the rating of the item j by the customer i. The set of observed ratings is $\mathcal{I}$. The aim of collaborative filtering is to predict any unknown entry $\mathbf{X}_{ij}$, with $(i,j)\notin\mathcal{I}$.

In real-world applications, the items are naturally grouped into types, possibly hierarchically, *i.e.* we can assign items to the leafs of a tree hierarchy and each item can have one or more attribute, similar to our treatment of classes in Section 2.7.1.

One possible group design is as follows. First, we introduce a single group for the entire matrix, to obtain the low-rank matrix factorization behavior of state of art models, with $\mathcal{D}_1=\{1,\dots,d\}$ and $\mathcal{Y}_1=\{1,\dots,k\}$. Then, for $g=2\dots T+1$, where T is the number of attributes, we define one group consisting of all the items that hshare a given attribute, *i.e.*

$$\mathcal{Y}_g=\{i\in\{1,\dots,k\}\,|\,i\text{-th item has the }g\text{-th attribute}\}$$

and $\mathcal{D}_g=\{1,\dots,d\}$.

2.7.3 Compression of a structured database

We next consider a lossy compression of a huge dataset, by using composite conditional gradient Dudik et al. (2012). The input is a full matrix $\mathbf{X}\in\mathbb{R}^{d\times k}$ containing the dataset, where each column is the feature vector $\mathbf{x}_j\in\mathbb{R}^d$ of the example j.

For instance, consider the data described in Section 2.7.1 and Fig. 2.3. Suppose that there are n_c images for `catamaran`, n_t for `trimaran`, n_b for `catboat` and n_g for `galleon`. In total we have $k=n_c+n_t+n_b+n_g$ examples with d features.

We model the problem as

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times k}} \tfrac{1}{2}\|\mathbf{W}-\mathbf{X}\|^2+\lambda\Omega_{\mathcal{G}}(\mathbf{W}). \tag{2.27}$$

Here we could define one group per each node: $\mathcal{Y}_1=\{1,\dots,n_c\}$ for the `catamaran` node, $\mathcal{Y}_2=\{n_c+1,\dots,n_t\}$ for the `trimaran` node, $\mathcal{Y}_3=\{n_t+1,\dots,n_b\}$ for the `catboat` node, $\mathcal{Y}_4=\{n_b+1,\dots,n_g\}$ for the `galleon` node. We add also one group $\mathcal{Y}_4=\{1,\dots,n_c+n_t+n_b\}$ for all

`sailboat` and one for `sailing vessel` $\mathcal{Y}_5 = \{1, \ldots, k\}$. $\mathcal{D}_g = \{1, \ldots, d\}$.

For each block g we can choose a (small) rank $r_g > 0$, possibly identical for all groups. The composite conditional gradient algorithm takes advantage of the groups and produces a sequence of factorized matrices $\mathbf{M} := \{\mathbf{U}_g \mathbf{V}_g^\top\}_{g \in \mathcal{G}}$, where $\mathbf{U}_g \in \mathbb{R}^{|\mathcal{D}_g| \times r_g}$ and $\mathbf{V}_g \in \mathbb{R}^{|\mathcal{Y}_g| \times r_g}$

The compressed dataset $\mathbf{M}$ will then only require the storage of

$$\sum_g r_g(|\mathcal{D}_g| + |\mathcal{Y}_g|)$$

floating point numbers.

2.7.4 Feature concatenation

Suppose we have a signal and several sets of features (*e.g.* Fourier transform of the signal, wavelet transform, etc.), and suppose we want to find a linear (and low rank) model for each feature set, *i.e.* $\hat{\mathbf{y}} = \mathbf{W}_{\mathcal{D}_g}^\top \mathbf{x}_{\mathcal{D}_g}$. To train all the models together, we could concatenate the feature vectors into a single one, concatenate the matrices of the model parameters, and define groups associated with individual feature sets. In this example only the rows of the matrix $\mathbf{W}$ are split into different groups.

This construction is particularly effective when the feature vectors are concatenations of different types of features. The sparsity-promoting properties of the group nuclear norm can then select the group(s) corresponding to most important feature sets, and zero out the less informative feature sets. More formally, each example is represented by its feature vector $\mathbf{x} \in \mathbb{R}^n$. The feature indices $i \in \mathcal{D}$ are divided into groups $\mathcal{D}_g$ according to the type of feature.

In this way the linear model $\mathbf{W}$ is forced to be of small rank with respect to each feature group, tries to select one group and keep zeroes on the coefficients of all the other groups.

2.7.5 Combinations

While we chose to described various approaches to the group construction separately, all the previous examples of groupings are mutually compatible and can be applied simultaneously. Thus the set $\mathcal{G}$ can contain groups that are based on the tree hierarchy, attributes, as well as features concatenation, as illustrated in Fig. 2.2, where we defined several different types of groups.

2.7.6 Object cosegmentation

The cosegmentation problem Kim et al. (2011) Rother et al. (2006) Vicente et al. (2011) is the task of finding the regions of images or videos that contain the same object. Here the group Schatten norm can impose the constraint that regions with the same object in different images should have similar visual feature distributions. The images can be segmented into parts, possibly of different shapes, that define the groups for the group Schatten norm.

2.7.7 Unsupervised learning

All of our applications of the group Schatten norm assume that the groups are known ahead of time. Nevertheless, one could also embed the optimization with group Schatten norm in a bigger algorithm. For example, we could imagine an algorithm that, in each iteration, finds groups with some criteria and then optimizes with respect to the current choice of groups.

2.7.8 Issue of groups that are unions of other groups

Let $\mathbf{A}_1$ be the concatenation of two matrices, *i.e.* $\mathbf{A}_1 = [\mathbf{A}_2\ \mathbf{A}_3]$, and we introduce three groups for each matrix $\mathbf{A}_g$.

As $\text{rank}(\mathbf{A}_1) \leq \text{rank}(\mathbf{A}_2) + \text{rank}(\mathbf{A}_3)$, the two smaller groups ($g = 2, 3$) might have no effect on the optimization problem. If this happens while running numerical experiments, one could force the regularization $\Omega_{\mathcal{G}}$ to consider the small groups by choosing the corresponding weights α_2 and α_3 sufficiently larger than α_1.

Chapter 3

Smoothing techniques for first-order optimization

3.1 Introduction

3.1.1 Smoothing in optimization

A general methodology for solving nonsmooth optimization problems is to solve instead a sequence of smooth problems approaching the original nonsmooth optimization problem. The main idea behind this methodology is each smooth problem can be solved efficiently. In contrast to smooth optimization algorithms, nonsmooth optimization algorithms such as subgradient optimization or bundle methods typically converge slowly to the solution in worst-case theoretical analysis Hiriart-Urruty and Lemarechal (1996).

Various smoothing techniques have been proposed in the mathematical optimization literature. Some are problem-dependent, others are generic for analytic classes of functions; see *e.g.* the pioneering papers Moreau (1965) and Bertsekas (1973). We build in this chapter upon the general approach of Beck and Teboulle (2012). The central concept is a *smoothable function*, which allows to combine generic smoothing approximations with fast first-order algorithms for solving nonsmooth convex optimization problems.

As far as optimization is concerned, the existence of a gradient and a local smoothing around the minimum would be enough. However, in order to use smooth approximations for the first-order oracles called by first-order algorithms, a stronger smoothness and a uniform approximations are required to control the convergence of iterates produced by the algorithms. More precisely, we say in this chapter

that f is a smooth approximation of g if

i) f is differentiable with Lipschitz continuous gradient;

ii) f approximates uniformly g.

The Lipschitz constant of the gradient and the uniform approximation constant will have special roles in the speed of convergence of the methods.

3.1.2 Contributions and outline of this chapter

In this chapter we aim at unifying the design and analysis of smoothing techniques combined with first order optimization algorithms. The chapter can be viewed as a natural companion to Beck and Teboulle (2012). A preliminary version of this work was presented in a conference Pierucci et al. (2014), focusing on a problem where smoothing is particularly useful: machine learning problems leading to "doubly" nonsmooth objectives, namely robust collaborative filtering or and multiclass classification.Our contributions are the following:

1. We present a simple and general framework to construct smooth approximations by inf-convolution (the classic smoothing in optimization) and by product convolution (the classical smoothing in analysis). We follow the construction of Beck and Teboulle (2012) and complements it with the one of Bertsekas (1973); Duchi et al. (2012). We make them as accessible as possible by simplify the general framework of Beck and Teboulle (2012), adopting a uniform presentation of the two smoothing techniques, and illustrate them by simple examples.

2. We develop examples of smoothing approximations for the top-k function, a nonsmooth function appearing machine learning applications Harchaoui et al. (2012a); Yager (1988). This is our main technical contribution.

3. We show how to combine smooth with optimization algorithm, extending the approach for proximal algorithms of Nesterov (2007b) and Beck and Teboulle (2012) to other popular algorithms in machine learning (conditional gradient algorithms and minimization-majorization algorithms).

We refer the reader to the above references for more details, analysis, and applications on smoothing approaches, especially for numerical illustrations.

The outline of this chapter is then as follows. After finishing the introduction by recalling some basic notions and notation, Section 3.2 presents the generic smoothing by inf-convolution and illustrates it on a simple example (the ℓ_1-norm) and on a sophisticated example (the top-k function). Section 3.3

follows the same structure and presents the generic smoothing by convolution and illustrates it on the ℓ_1-norm. Finally Section 3.4 develops the combination of smoothing with first-order methods for doubly nonsmooth optimization learning problems.

3.1.3 Recalls in convex analysis

In this chapter we make a basic but constant use of standard definitions and concepts of convex analysis: we briefly recall them here for the convenience of a non-expert reader, and we refer the classical textbooks Rockafellar (1970) and Hiriart-Urruty and Lemarechal (1993) for a comprehensive exposition of these concepts.

We recall that the conjugate of the indicator function of a set $\mathcal{Z}$ is called the support function of $\mathcal{Z}$

$$g(\mathbf{x}) = (i_{\mathcal{Z}})^*(\mathbf{x}) = \max_{\mathbf{z}\in\mathcal{Z}} \langle \mathbf{x}, \mathbf{z} \rangle \ . \tag{3.1}$$

In particular, positive homogeneous functions, that is, satisfying

$$f(t\mathbf{x}) = tf(\mathbf{x}) \qquad \text{for all } \mathbf{x} \in E, \text{ and all } t > 0 \ , \tag{3.2}$$

can be naturally written as support functions. Recall also that, in general, we have that $(f^*)^* = f$ for a closed and convex function f Hiriart-Urruty and Lemarechal (1993). Finally, we recall the definition of infimal convolution (inf-convolution) of two convex functions

$$(f \Box h)(\mathbf{x}) := \inf_{\mathbf{z}\in E} (f(\mathbf{x} - \mathbf{z}) + h(\mathbf{z})) \tag{3.3}$$

and the property regarding its interplay with Fenchel conjugation

$$(f \Box g)^* = f^* + g^* \ . \tag{3.4}$$

3.2 Smoothing by infimal convolution

In this section, we briefly review the ideas of the smoothing by inf-convolution, insisting on the special case of the so-called smoothing by saddle-point representation. We refer mostly on the general treatment of Beck and Teboulle (2012) which encompasses many previous techniques. We also develop an original application of the smoothing by saddle-point representation to a function used in machine learning applications.

3.2.1 General construction and special cases

A classical smoothing technique in optimization is based on inf-convolution operations (3.3). The essence of this technique can be traced back to Moreau (1965). The chapter showed how to inter-twin this technique with first-order methods Nesterov (2007b) to get optimal rates of convergence.

Let g be a convex function; we can construct a smooth approximation of g with convolution with smooth convex function ω. More precisely, we have the following definition from Beck and Teboulle (2012). Note that, in contrast to Beck and Teboulle (2012), we do not look for generality here, as our main goal is to illustrate the technique for machine learning problems. Therefore we slightly simplify the framework of Beck and Teboulle (2012) by restricting to the convex function in Euclidean space to keep the assumptions and the developments as simple as possible.

Definition 6. For a given L-smooth closed convex function ω, we define the "inf-conv γ-approximation" of the function g by the infimal convolution of g with ω rescaled by a parameter $\gamma > 0$

$$g_\gamma^{ic}(\mathbf{x}) := \left(g \square \gamma\omega\left(\frac{\cdot}{\gamma}\right)\right)(\mathbf{x}) = \inf_{\mathbf{z}\in E} g(\mathbf{x}-\mathbf{z}) + \gamma\omega\left(\frac{\mathbf{z}}{\gamma}\right) \tag{3.5}$$

The function g_γ^{ic} is thus defined implicitly by the result of a minimization problem parameterized by $\gamma > 0$. Note that if g is positively homogeneous (3.2), the role of the parameter γ is transparent, since with a mere change of variable

$$g_\gamma^{ic}(\mathbf{x}) = \gamma \inf_{\mathbf{z}\in E} \frac{1}{\gamma} g(\mathbf{x}-\mathbf{z}) + \omega\left(\tfrac{\mathbf{z}}{\gamma}\right) = \gamma \inf_{\mathbf{z}\in E} g\left(\tfrac{\mathbf{x}}{\gamma} - \mathbf{z}\right) + \omega(\mathbf{z}) = \gamma g_1^{ic}\left(\frac{\mathbf{x}}{\gamma}\right) , \tag{3.6}$$

so that g_γ^{ic} can simply expressed from g_1^{ic}.

The next theorem states that g_γ^{ic} is indeed a smooth approximation of g, under a mild assumption on the sub-differential of ω. The proof of this result, based on standard convex analysis properties Hiriart-Urruty and Lemarechal (1993), follows directly from results stated in Beck and Teboulle (2012). A refined version of it is given in (Beck and Teboulle, 2012, Corollary 4.1).

Theorem 3.2.1. *For a given L-smooth convex function ω, the function g_γ^{ic} defined by* (3.5) *satisfies the properties of smooth approximation of g:*

i) The function g_γ^{ic} is L/γ-smooth and its gradient can be expressed as

$$\nabla g_\gamma^{ic}(\mathbf{x}) = \nabla\omega\left(\frac{\mathbf{x}-\mathbf{z}_\star(\mathbf{x})}{\gamma}\right) , \tag{3.7}$$

where $\mathbf{z}_\star(\mathbf{x})$ minimize (3.5).

ii) If there exists a constant M such that $\omega^*(\mathbf{s}) \leq M$ for all $\mathbf{x} \in E$ and all $\mathbf{s} \in \partial g(\mathbf{x})$, then we have the uniform bound

$$\left|g(\mathbf{x}) - g_\gamma^{ic}(\mathbf{x})\right| \leq \gamma m, \qquad \textit{for all } \mathbf{x} \in E \tag{3.8}$$

with $m := \max\{-M, \omega(\mathbf{0})\}$.

Proof. The differentiability of g_γ^{ic} follows from the basic properties of inf-convolution (Beck and Teboulle, 2012, Theorem 4.1). The same holds for the expression of ∇g_γ^{ic} and its Lipschitz continuity. Now (Beck and Teboulle, 2012, Lemma 4.2) gives that

$$g(\mathbf{x}) - \gamma\omega^*(\mathbf{s}) \leq g_\gamma^{ic}(\mathbf{x}) \leq g(\mathbf{x}) + \gamma\omega(\mathbf{0}), \qquad \text{for all } \mathbf{s} \in \partial g(\mathbf{x}) \ .$$

This gives property (ii) with the help of the boundedness assumption. □

It is worth noting that the Lipschitz constant of g_γ^{ic} depends only on ω and γ, which can be both appropriately chosen for our purpose. This explains the interest of combining such smoothing with first-order optimization methods since the theoretical rate of convergence typically depends on the Lipschitz constant of the function to be minimized (see forthcoming Section 3.4 and Beck and Teboulle (2012)). We also observe that the practical computational cost to evaluate the value of the function (3.5) and its gradient (3.7) is the same and boils down to computing a solution of (3.5).

The above construction of smooth approximation is generic enough to recover many well-known classical approximations: Moreau-Yosida smoothing Moreau (1965), asymptotic smoothing Ben-Tal and Teboulle (1989), and Nesterov or smoothing by saddle-point representation Nesterov (2007b).

Example 7 (Moreau-Yosida smoothing). Let g be any nonsmooth convex function. Recall that the Moreau-Yosida approximation of g (Hiriart-Urruty and Lemarechal, 1993, Section E.2) is defined by

$$g_\gamma^{px}(\mathbf{x}) := \inf_{\mathbf{z} \in E} g(\mathbf{z}) + \tfrac{1}{2\gamma} \left\|\mathbf{z} - \mathbf{x}\right\|^2 .$$

By taking $\omega = \frac{1}{2} \left\|\cdot\right\|^2$ in (3.5), we see easily that the Moreau-Yosida approximation is a special instance of inf-conv approximation (3.5).

Example 8 (Asymptotic smooth approximation). Let g be an asymptotic function Ben-Tal and Teboulle (1989), defined by the pointwise limit of ω rescaled by γ, more precisely

$$g(\mathbf{x}) = \lim_{\gamma \to 0^+} \gamma\omega\left(\frac{\mathbf{x}}{\gamma}\right) \qquad \text{for all } \mathbf{x}. \tag{3.9}$$

In this case, $g_\gamma^{as} := \gamma\omega\left(\frac{\cdot}{\gamma}\right)$ itself is a smooth approximation of g, see Ben-Tal and Teboulle (1989) and (Beck and Teboulle, 2012, Theorem 4.2 and Lemma 4.3). By taking (3.9) in (3.5), we see that g_γ^{as} can also be interpreted as a special instance of inf-conv smooth γ-approximation, because there also holds

$$\left(g \,\square\, \gamma\omega\left(\frac{\cdot}{\gamma}\right)\right) = g_\gamma^{as}. \tag{3.10}$$

To prove the above equation, use first (Beck and Teboulle, 2012, Lemma 4.3) states that $g^* = \delta_{\text{cl dom}\,\omega^*}$ in the case (3.9), and then write using (3.4) and the closedness of g_γ^{as}:

$$g\square g_\gamma^{as} = (g^* + (g_\gamma^{as})^*)^* = (\delta_{\text{cl dom}\,\omega^*} + (g_\gamma^{as})^*)^* = (g_\gamma^{as*})^* = g_\gamma^{as}$$

Thus the inf-conv smoothing is not used here to produce smooth approximation, but this natural smoothing in this context can be a posteriori interpreted as inf-conv smoothing.

Example 9 (Saddle-point representation). Let g be the Fenchel conjugate of the sum of a closed convex function ϕ and the indicator function of compact convex set $\mathcal{Z} \subset E$

$$g = (\phi + i_{\mathcal{Z}})^*. \tag{3.11}$$

Recall that the saddle-point representation of g Cox et al. (2014), which underlies the so-called *Nesterov smoothing technique* Nesterov (2005), is defined by

$$g_\gamma^{ft}(\mathbf{x}) := \max_{\mathbf{z}\in\mathcal{Z}} \langle A\mathbf{x}, \mathbf{z}\rangle - \phi(\mathbf{z}) - \gamma d(\mathbf{z}) \ , \tag{3.12}$$

where A is an affine function and d is a strongly convex function (known in literature as the proximity function, or the distance generating function Nesterov (2005)).

By taking $\omega = d^*$ in (3.5), we see that (3.12) is in fact constructed from an inf-conv smooth γ-approximation, as follows. For all $\mathbf{x}$, we first observe that by simple algebra

$$\gamma\omega\left(\frac{\cdot}{\gamma}\right) = (\gamma d)^* \tag{3.13}$$

and then we write

$$
\begin{aligned}
& \left(g \,\square\, \gamma\omega\left(\tfrac{\cdot}{\gamma}\right)\right)(A\mathbf{x}) & \\
= \;& \left((\phi + i_{\mathcal{Z}})^* \,\square\, (\gamma d)^*\right)(A\mathbf{x}) & \text{[by (3.11) and (3.13)]} \\
= \;& (\phi + i_{\mathcal{Z}} + \gamma d)^*(A\mathbf{x}) & \text{[by (3.4)]} \\
= \;& \max_{\mathbf{z}\in E}\langle A\mathbf{x}, \mathbf{z}\rangle - \phi(\mathbf{z}) - i_{\mathcal{Z}}(\mathbf{z}) - \gamma d(\mathbf{z}) & \text{[by definition of the conjugate]} \\
= \;& g_\gamma^{ft}(\mathbf{x}) & \text{[by definition of } g_\gamma\text{]}
\end{aligned}
$$

Thus g_γ^{ft} is a inf-conv approximation composed with A.

Example 10 (smoothing by saddle-point representation of support function with squared norm). Let us instantiate the previous example with $\phi \equiv 0$ and the squared function as proximity function

$$\omega(\cdot) = d(\cdot) = \frac{1}{2}\left\|\cdot\right\|_2^2.$$

The smooth approximation then simply writes as follows

$$g_\gamma^{ft}(\mathbf{x}) = \left\langle \mathbf{x}, \pi_{\mathcal{Z}}\left(\frac{\mathbf{x}}{\gamma}\right)\right\rangle - \frac{\gamma}{2}\left\|\pi_{\mathcal{Z}}\left(\frac{\mathbf{x}}{\gamma}\right)\right\|^2 \tag{3.14}$$

where $\pi_{\mathcal{Z}}$ is projector operator onto the convex set $\mathcal{Z}$. For $\gamma = 1$ this comes from

$$\operatorname*{argmax}_{\mathbf{z}\in\mathcal{Z}} \langle \mathbf{x}, \mathbf{z}\rangle - \tfrac{1}{2}\left\|\mathbf{z}\right\|^2 = \operatorname*{argmin}_{\mathbf{z}\in\mathcal{Z}} \left\|\mathbf{x} - \mathbf{z}\right\|^2 = \pi_{\mathcal{Z}}(\mathbf{x}),$$

This yields (3.14) with the help of (3.6).

3.2.2 A simple example of smoothing by saddle-point representation: absolute value

Several interesting examples are developed in (Beck and Teboulle, 2012, Section 4.6). Here we illustrate the versatility of smoothing by saddle-point representation on a simple one-dimensional example: the absolute value function. We write the absolute value function as the support function of $\mathcal{Z} = [-1, 1] \subset \mathbb{R}$

$$g(x) = |x| = \max_{z\in[-1,1]} zx \tag{3.15}$$

The Fenchel-type approximation of this function at Example 9 have the following expressions

- using the squared norm $d(z) = \frac{1}{2}z^2$, we have

$$g_\gamma(x) = \begin{cases} \frac{1}{2\gamma}x^2 & \text{if } |x| \leq \gamma \\ |x| - \frac{\gamma}{2} & \text{if } |x| > \gamma \end{cases} \tag{3.16}$$

- using the symmetric entropy

$$d(z) = \begin{cases} (1-|z|)\ln(1-|z|) + |z| & \text{if } |z| < 1 \\ 1 & \text{if } |z| = 1, \end{cases}$$

  we have

$$g_\gamma(x) = \gamma e^{-\left|\frac{x}{\gamma}\right|} + |x| - \gamma \tag{3.17}$$

- using $d(y) = 1 - \sqrt{1-y^2}$, we have

$$g_\gamma(x) = \sqrt{x^2+\gamma^2} - \gamma \tag{3.18}$$

The proof (3.16) is a straightforward application of (3.14). For (3.17), we observe first that the symmetric entropy function is strongly convex (since the second derivative $d''(z) \geq 1$), and second we compute (for $\gamma = 1$)

$$z_\star(x) = \max_{z\in[-1,1]} zx - d(z) = \begin{cases} 1 - e^{-x} & \text{if } x \geq 0 \\ e^x - 1 & \text{if } x < 0 \end{cases}$$

Plugging this expression in the definition and using (3.6) gives easily (3.17). Similar calculus gives (3.18).

3.2.3 An advanced example of smoothing by saddle-point representation: the top-k function

In this subsection, we illustrate the smoothing by saddle-point representation on an example coming from machine learning and computer vision. The top-k accuracy is a popular measure of performance for applications involving multi-class classification Harchaoui et al. (2012a) or ranking Usunier et al. (2009); Weimer et al. (2007); Yager (1988). The convex top-k error function is the support function

$$g(\mathbf{x}) = \max_{\mathbf{z}\in\mathcal{Z}^{tk}} \langle \mathbf{x}, \mathbf{z} \rangle \tag{3.19}$$

of the “reduced simplex" in $\mathbb{R}^n$

$$\mathcal{Z}^{tk} := \left\{ \mathbf{z} \in \mathbb{R}^n : \ 0 \leq z_i \leq \frac{1}{k}, \sum_{i=1}^{n} z_i \leq 1 \right\}.$$

When $k = 1$, the top-1 error is well-known as the multi-dimension hinge function, written as

$$g(\mathbf{x}) = \max_{\sum_i z_i \leq 1, z_i \geq 0} \langle \mathbf{x}, \mathbf{z} \rangle = \max_{i=1,\ldots,n} \{x_i, 0\} \ . \tag{3.20}$$

Except for this case $k = 1$, the non-smoothness of top-k error function for any k makes its use non-straightforward when using gradient-based optimization algorithms for training machine learning models.

We develop here tractable smooth approximations of it, which makes its use easier for these machine learning problems. The next theorem gives the general smoothing by saddle-point representation of top-k error function. The following example describes further a special case. The third result finally shows that in the case $k = 1$, we retrieve a smooth surrogate widely used in machine learning.

Theorem 3.2.2. *Consider a strongly convex function $d(\mathbf{z}) := \sum_{i=1}^{n} d^{(i)}(z_i)$ defined on $[0, 1/k]^n$. Then the Fenchel-type smooth approximation of the top-k function* (3.19) *can be written as*

$$g_\gamma(\mathbf{x}) = \max_{\mathbf{z} \in \mathcal{Z}^{tk}} \langle \mathbf{z}, \mathbf{x} \rangle - \gamma d(\mathbf{z}) = -\lambda_\star(\mathbf{x}, \gamma) + \sum_{i=1}^{n} H_\gamma^{(i)}(x_i + \lambda_\star(\mathbf{x}, \gamma)). \tag{3.21}$$

In the above equation, $H_\gamma^{(i)}$ is the following smooth approximation of $d^{(i)}$

$$H_\gamma^{(i)}(t) := \max_{0 \leq r \leq 1/k} \langle t, r \rangle - d^{(i)}(r) \qquad \textit{for any } t \in \mathbb{R} \ , \tag{3.22}$$

and $\lambda_\star(\mathbf{x}, \gamma) \in \mathbb{R}^+$ is an optimal solution of the minimization of

$$\Theta(\lambda) = -\lambda + \sum_{i=1}^{n} H_\gamma^{(i)}(x_i + \lambda) \ .$$

We plot an illustration of (3.21) in Figures 3.1 and 3.2.

Proof. Let us dualize the linking constraint $\sum_{i=1}^{n} z_i \leq 1$ in the definition of the smooth approximation (3.12). We define the associated Lagrange function for $z \in [0, \frac{1}{k}]$ and $\lambda \geq 0$

$$L(\lambda, \mathbf{z}) := \langle \mathbf{x}, \mathbf{z} \rangle - \gamma d(\mathbf{z}) + \lambda \left(\sum_{i=1}^{n} z_i - 1 \right) = -\lambda + \sum_{i=1}^{n} \left(z_i x_i - \gamma d^{(i)}(z_i) + \lambda z_i \right).$$

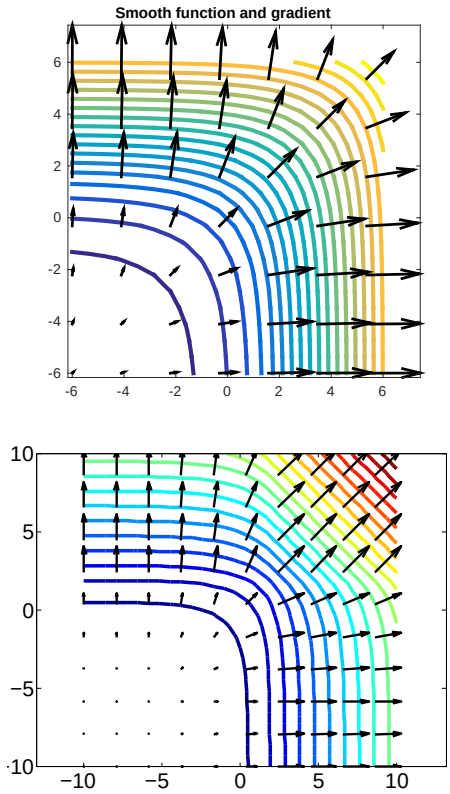

Figure 3.1: Smoothed $g_\gamma(\mathbf{x})$ of equation (3.21) and its gradient, $\mathbf{x} \in \mathbb{R}^2$ for top-k error. **Top** with $k = 1$, **bottom** with $k = 2$. The color lines represent level sets and the arrows are the gradients of the smooth surrogate.

and the associated convex dual problem

$$\Theta(\lambda) := \max_{\forall i,\ z_i \in [0, \frac{1}{k}]} L(\lambda, \mathbf{z}).$$

For simplicity the dependence of Θ and L on $\mathbf{x}$ and γ is implicit. Developing the expression of Θ, we get

$$\Theta(\lambda) = -\lambda + \sum_{i=1}^{n} \max_{z_i \in [0, \frac{1}{k}]} \Big(z_i(x_i + \lambda) - \gamma d^{(i)}(z_i) \Big) = -\lambda + \sum_{i=1}^{n} H_\gamma^{(i)}(x_i + \lambda).$$

Since they are smooth approximations (recall Example 9), the functions $H_\gamma^{(i)}$ are differentiable, and so is Θ. Its derivative is moreover

$$\Theta'(\lambda) = -1 + \sum_{i=1}^{n} \nabla H_\gamma^{(i)}(x_i + \lambda). \tag{3.23}$$

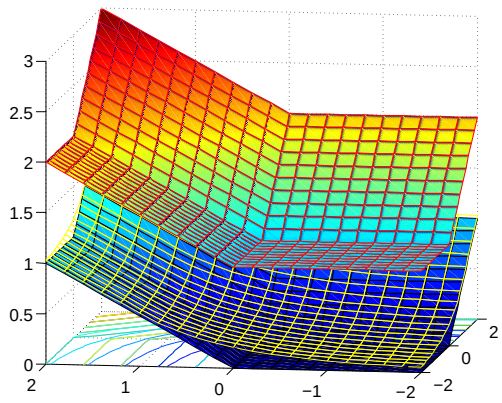

Figure 3.2: Smoothed $g_\gamma(\mathbf{x})$ of equation (3.21) for $k = 1$, between the nonsmooth bounds g (below) and $g + \gamma m$ (above). Compare with equation 3.8.

From (3.22), we see that $\nabla H_\gamma^{(i)}$ takes all the values in $]0, \frac{1}{k}[$. Since $\nabla H_\gamma^{(i)}$ is increasing and $n > k$, we get that $\Theta'(\lambda) = 0$ has always a solution, that we denote $\tilde{\lambda}$. Therefore we have that the optimal dual solution is $\lambda_\star(\mathbf{x}, \gamma) = \max\{0, \tilde{\lambda}\}$. By convexity and primal compactness, there is no duality gap (Hiriart-Urruty and Lemarechal, 1996, Chap.XII), $g_\gamma(\mathbf{x}) = \Theta(\lambda_\star(\mathbf{x}, \gamma))$, which gives the desired expression. $\square$

In practice, minimizing Θ over $\mathbb{R}^+$ amounts to compute a solution of $\Theta'(\lambda) = 0$, which can be made by a mere bisection algorithm, for example. For special cases, we have refined procedures as in the next example, or even explicit expressions as in forthcoming Corollary 3.2.3.

Practical computation of top-k smooth surrogate

Squared norm By choosing $d^{(i)}(z_i) = \frac{1}{2} z_i^2$, we have more specific expressions of the objects of Theorem 3.2.2, as follows. First we obtain directly $H_\gamma^{(i)}(t) = \pi_{[0, \frac{1}{k}]}(t)$ and then

$$\Theta'(\lambda) = 1 - \sum_i p_i(\lambda) \quad \text{with} \quad p_i(\lambda) := \pi_{[0, \frac{1}{k}]}(x_i + \lambda).$$

Then we introduce, for a given $\mathbf{x}$, the set

$$P := \left\{ x_i, x_i - \frac{1}{k} : \ i = 1 \dots n \right\}$$

and a as the largest element of P such that $\Theta'(\lambda) \leq 0$, and similarly b as the smallest such that $\Theta'(\lambda) \geq 0$. Note that each p_i is constant (equal to 0) when λ is lower than a point $\hat{x}_i$ in P, constant also (equal to $1/k$)

when λ is larger than another point $\tilde{x}_i$ in P, and affine in-between. Thus Θ' is increasing and piecewise affine with kinks on P. By definition of a and b in P, Θ' has its zero $\tilde{\lambda}$ on $[a, b]$. Since Θ' is affine in $[a, b]$, its slope is given by $(\Theta'(b) - \Theta'(a))/(b - a)$, and then its zero can be expressed by $\tilde{\lambda} = a - \frac{\Theta'(a)(b-a)}{\Theta'(b)-\Theta'(a)}$. This yields

$$\lambda_\star(\mathbf{x}, \gamma) = \max\left\{0, a - \frac{\Theta'(a)(b-a)}{\Theta'(b) - \Theta'(a)}\right\},$$

Thus the smooth approximation of the top-k function is given by (3.21) with the above expressions of $H_\gamma^{(i)}(t)$ and $\lambda_\star(\mathbf{x}, \gamma)$. The main computation consists in finding a and b by dichotomy in time $O(\log(n))$ by evaluating the sign of $\Theta'(x_i)$.

Entropy and k = 1 When $k = 1$ and $d^{(i)}(z_i) = z_i \ln(z_i) - z_i$, we obtain the following smooth approximation of (3.20)

$$g_\gamma(\mathbf{x}) = \begin{cases} \gamma\left(1 + \ln \sum_{i=1}^n e^{\frac{x_i}{\gamma}}\right) & \text{if} \quad \sum_{i=1}^n e^{\frac{x_i}{\gamma}} > 1, \\ \gamma \sum_{i=1}^n e^{\frac{x_i}{\gamma}} & \text{if} \quad \sum_{i=1}^n e^{\frac{x_i}{\gamma}} \leq 1. \end{cases} \tag{3.24}$$

A common application in machine learning for this function is in support vector machines (SVM), used for instance for multi-class classification. Surprisingly, we see that choosing $\gamma = 1$, we obtain the multinomial logistic loss, which is usually motivated from the generalized linear models framework Hastie et al. (2008).

To prove the above formula, we start by the smooth approximation (3.22) which follows from calculations as

$$H_\gamma^{(i)}(t) = \begin{cases} \gamma e^{\frac{t}{\gamma}} & \text{if} \quad t < 0, \\ t + \gamma & \text{if} \quad t \geq 0 \end{cases}$$

Following from (3.23) we find

$$\Theta'(\lambda) = -1 + \sum_{i=1}^n \begin{cases} e^{\frac{x_i+\lambda}{\gamma}} & \text{if} \quad \lambda < -x_i, \\ 1 & \text{if} \quad \lambda \geq -x_i. \end{cases}$$

We notice that to have $\Theta'(\lambda) = 0$, it must be $\lambda < -x_i$ for all indices i, otherwise it would be $\Theta'(\lambda) > 0$. We emphasize that this argument rely on $k = 1$. Then

$$\Theta'(\lambda) = 0 \iff -1 + \sum_{i=1}^n e^{\frac{x_i+\lambda}{\gamma}} = 0 \iff \lambda = -\gamma \ln\left(\sum_{i=1}^n e^{\frac{x_i}{\gamma}}\right)$$

We obtain the optimal nonnegative

$$\lambda_\star(\mathbf{x}) = \begin{cases} -\gamma \ln\left(\sum_{i=1}^n e^{\frac{x_i}{\gamma}}\right) & \text{if } \sum_{i=1}^n e^{\frac{x_i}{\gamma}} < 1, \\ 0 & \text{if } \sum_{i=1}^n e^{\frac{x_i}{\gamma}} \geq 1. \end{cases}$$

We substitute $b := \sum_{j=1}^n e^{\frac{x_j}{\gamma}}$ and obtain after simplifications

$$\Theta(\lambda_\star(\mathbf{x})) = -\lambda_\star(\mathbf{x}) + \sum_{i=1}^n H_\gamma(x_i + \lambda_\star(x_i)) = \begin{cases} \gamma \ln(b) + \gamma & \text{if } b > 1, \\ \gamma b & \text{if } b \leq 1. \end{cases}$$

This yields (3.24).

Entropy and k $\geq$ 1 When k is larger than 1, the tricks of the previous example cannot be applied, and we need to proceed by dichotomy to have an exact solution. This technique is suitable also for $k = 1$.

Let the proximity function be defined as $\omega(\mathbf{z}) = \sum_{i=1}^n z_i \ln(kz_i)$. Then the dual objective is

$$\Theta(\lambda) = -\lambda + \sum_{i=1}^n h_\gamma(x_i + \lambda), \tag{3.25}$$

where $h(t) := t \ln(kt)$, and its gradient is

$$\Theta'(\lambda) = -1 + \sum_{i=1}^n \nabla h_\gamma(x_i + \lambda) = -1 + \sum_{\lambda \leq \gamma - x_i} \frac{1}{k} e^{\frac{x_i+\lambda}{\gamma} - 1} + \sum_{\lambda > \gamma - x_i} \frac{1}{k}\ , \tag{3.26}$$

that we illustrated in Figure 3.3.

We use here proposition 3.2.3. We look for $\lambda_\star$ such that $\Theta'(\lambda_\star) = 0$. First we sort $\mathbf{x}$ in decreasing order. From now on we have $\mathbf{x}_0 \geq \mathbf{x}_1 \geq \cdots \geq \mathbf{x}_n$. The nondifferentiable points of Θ' are $\gamma - x_i$. By dichotomy we find an index $i_\star$ such that $\Theta'(\gamma - \mathbf{x}_{i_\star}) < 0$ and $\Theta'(\gamma - \mathbf{x}_{i_\star+1}) \geq 0$.

Then now the indices of the sums don't depend on $\lambda \in [\gamma - \mathbf{x}_{i_\star}, \gamma - \mathbf{x}_{i_\star+1}]$

$$\begin{aligned} \Theta'(\lambda) &= -1 + \sum_{i=i_\star+1}^n \frac{1}{k} e^{\frac{x_i+\lambda}{\gamma} - 1} + \sum_{i=1}^{i_\star} \frac{1}{k} \\ &= -1 + e^{\frac{\lambda}{\gamma}} \frac{1}{k} \sum_{i=i_\star+1}^n e^{\frac{x_i}{\gamma} - 1} + i_\star \frac{1}{k} \\ &= -1 + De^{\frac{\lambda}{\gamma}} + E \end{aligned}$$

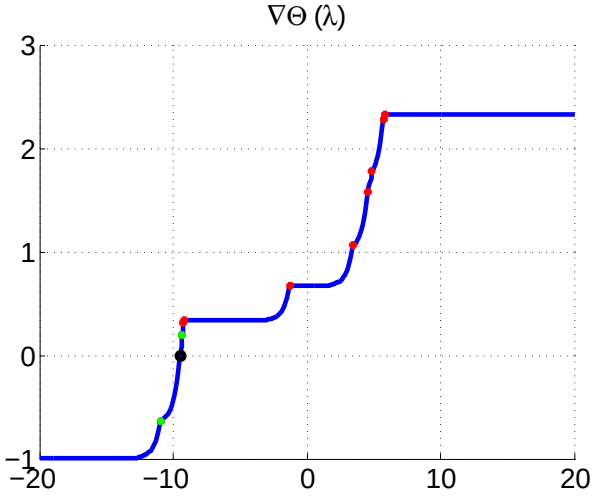

Figure 3.3: Illustration of the derivative of the dual function Θ, for $\gamma = 1/2$, smooth surrogate of top-k error. Red: points without second derivative; black: point of minimum of Θ; green: bounds of the 'segment' (with second derivative) that contains the minimum of Θ. The x_i are randomly sampled.

where $D := \sum_{i=i_\star+1}^{n} \frac{1}{k} e^{\frac{x_i}{\gamma}-1} = \sum_{i=i_\star+1}^{n} \nabla g_\gamma(x_i)$ and $E := i_\star \frac{1}{k} = \sum_{i=1}^{i_\star} \nabla g_\gamma(x_i)$, if $i_\star = n$ we consider the second sum equal to zero. We conclude $-1 + De^{\frac{\lambda_\star}{\gamma}} + E = 0 \Leftrightarrow e^{\frac{\lambda_\star}{\gamma}} = \frac{1-E}{D} \Leftrightarrow \lambda_\star = \gamma \ln(\frac{1-E}{D})$.

We obtained an explicit formula to compute the surrogate, but computationally this is not accurate, due to exponential of big quantities or logarithms of small quantities, in particular when γ approaches 0.

The dichotomy has complexity $O(\ln(n))$, to sort $\mathbf{x}$ takes $O(n\ln(n))$. Then the evaluation of the surrogate is $O(n\ln(n))$.

Proposition 3.2.3. *Let us define on $\mathbb{R}$ the function $h(t) := t\ln(nt)$, where $n > 0$ is any real number. Then its conjugate over the segment $[0, \tau]$ is*

$$h_\gamma(t) = \begin{cases} \frac{\gamma}{n} e^{\frac{t}{\gamma}-1} & \text{if } \frac{t}{\gamma} < \ln(n\tau) + 1 \\ \tau t - \gamma\tau \ln(n\tau) & \text{if } \frac{t}{\gamma} \geq \ln(n\tau) + 1. \end{cases} \tag{3.27}$$

and the gradient of conjugate

$$\nabla h(t, \gamma) = \begin{cases} \frac{1}{n} e^{\frac{t}{\gamma}-1} & \text{if } \frac{t}{\gamma} < \ln(n\tau) + 1 \\ \tau & \text{if } \frac{t}{\gamma} \geq \ln(n\tau) + 1 \end{cases}$$

Proof. with $\gamma = 1$ we have $h_1(t) = \max_{y\in[0,\tau]} ty - y\ln(ny)$. The derivative is $\frac{d}{dy}(ty - y\ln(ny)) = t - \ln(ny) - 1 = 0 \Leftrightarrow \ln(ny) = t - 1 \Leftrightarrow y = \frac{1}{n}e^{t-1}$. Then the point of maximum is

$$y_\star(t) = \begin{cases} 0 & \text{if } \frac{1}{n}e^{t-1} < 0 \\ \frac{1}{n}e^{t-1} & \text{if } \frac{1}{n}e^{t-1} \in [0,\tau] \\ \tau & \text{if } \frac{1}{n}e^{t-1} > \tau, \end{cases}$$

i.e.

$$y_\star(t) = \begin{cases} \frac{1}{n}e^{t-1} & \text{if } t < \ln(n\tau) + 1 \\ \tau & \text{if } t \geq \ln(n\tau) + 1 \end{cases}.$$

Then

$$\nabla h_1(t) = \begin{cases} \frac{1}{n}e^{t-1} & \text{if } t < \ln(n\tau) + 1 \\ \tau & \text{if } t \geq \ln(n\tau) + 1 \end{cases}$$

and

$$h_1(t) = \begin{cases} \frac{1}{n}e^{t-1} & \text{if } t < \ln(n\tau) + 1 \\ \tau t - \tau\ln(n\tau) & \text{if } t \geq \ln(n\tau) + 1. \end{cases}$$

To conclude we use $\nabla h_\gamma(t) = \nabla h_1(\frac{t}{\gamma})$ and $h_\gamma(t) = \gamma h_1(\frac{t}{\gamma})$ □

3.3 Smoothing by product convolution

3.3.1 General construction

A classical smoothing technique in analysis in a broad sense is based on convolution with probability density. In the context of optimization, this smoothing technique can be traced back to Bertsekas (1973). We recall the general definition and we particularize it for two densities that give birth to smooth approximations as in the previous section on inf-convolution.

Definition 11. For a given probability density function μ on $\mathbb{R}^n$, we define the "convoluted smooth γ-approximation" of the convex function g by the convolution of g with μ rescaled by a concentration parameter γ:

$$g_\gamma^c(\mathbf{x}) := \int_{\mathbb{R}^n} g(\mathbf{x}-\mathbf{z})\frac{1}{\gamma}\mu\left(\frac{\mathbf{z}}{\gamma}\right)\mathrm{d}\mathbf{z} = \int_{\mathbb{R}^n} g(\mathbf{x}-\gamma\mathbf{z})\mu(\mathbf{z})\mathrm{d}\mathbf{z}. \tag{3.28}$$

It is the same definition as in (3.5) replacing ω by μ and the inf-convolution operation by the convolution operation. As previously, we also notice that the parameter $\gamma > 0$, controlling the concentration of

the distribution, has a simple role when g is positive homogeneous:

$$g_\gamma^c(\mathbf{x}) = \int_{\mathbb{R}^n} \gamma g(\tfrac{\mathbf{x}}{\gamma} - \mathbf{y})\mu(\mathbf{y})\mathrm{d}\mathbf{y} = \gamma\, g_1^c\left(\frac{\mathbf{x}}{\gamma}\right). \tag{3.29}$$

The convexity of the integral within (3.28) comes easily by definition; differentiability and expression of the gradient is obtained with a well-known result Bertsekas (1973) with a simple assumption on the measure. Restricting to μ_∞ the uniform distribution on the ℓ_∞ unit ball or μ_2 the Gaussian distribution (centered and with the identity as covariance matrix)

$$\mu_\infty(\mathbf{z}) = \frac{1}{2^n}\delta_{\{\|\cdot\|_\infty \leq 1\}}(\mathbf{z}) \qquad \text{and} \qquad \mu_2(\mathbf{z}) = \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{\|\mathbf{z}\|^2}{2}},$$

we obtain moreover the Lipschitz-continuity of the gradient, as formalized in the next theorem. In the proof of this theorem, we use technical results presented in Duchi et al. (2012).

Theorem 3.3.1. *Let $g\colon \mathbb{R}^n \to \mathbb{R}$ be a convex and Lipschitz continuous function. For the distributions μ_∞ and μ_2, the function g_γ^c defined by* (3.28) *satisfies the properties of smooth approximation of g:*

i) The function g_γ^c is differentiable and its gradient is

$$\nabla g_\gamma^c(\mathbf{x}) = \int_{\mathbb{R}^n} s(\mathbf{x} - \gamma\mathbf{z})\mu(\mathbf{z})\mathrm{d}\mathbf{z},$$

where $s(\mathbf{x})$ is a subgradient of g at $\mathbf{x}$. In addition ∇g_γ^c is Lipschitz continuous with constant $L_\gamma = L/\gamma$, where L depends on the dimension of the space and the Lipschitz constant of g.

ii) We have moreover

$$\left|g(\mathbf{x}) - g_\gamma^c(\mathbf{x})\right| \leq \gamma L_0 K, \tag{3.30}$$

where K depends on the distribution μ.

Proof. By (Bertsekas, 1973, Proposition 2.4), we know that g_γ^c is differentiable if the subset of non-differentiability of the nonsmooth function g has measure zero for μ. In our situation, the convexity of g implies that the subset where g is not differentiable has zero measure for the Lebesgue measure, and then for μ_∞ and μ_2 as well. The equation for ∇g_γ^c is described in (Bertsekas, 1973, Proposition 2.2). We obtain the Lipschitz continuity of the gradient we apply a couple of technical lemmas from Duchi et al. (2012). First, we introduce L_0 the Lipschitz constant of g and we use (Duchi et al., 2012, Lemma 11) to

obtain

$$\begin{aligned}\left\|\nabla g_\gamma^c(\mathbf{x}) - \nabla g_\gamma^c(\mathbf{y})\right\| \leq & L_0 \int \left|\tfrac{1}{\gamma}\mu(\tfrac{\mathbf{z}-\mathbf{x}}{\gamma}) - \tfrac{1}{\gamma}\mu_\gamma(\tfrac{\mathbf{z}-\mathbf{y}}{\gamma})\right| \mathrm{d}\mathbf{z} \\ = & L_0 \int \left|\mu(\mathbf{z} - \tfrac{\mathbf{x}}{\gamma}) - \mu_\gamma(\mathbf{z} - \tfrac{\mathbf{y}}{\gamma})\right| \mathrm{d}\mathbf{z},\end{aligned}$$

Second, (Duchi et al., 2012, Lemma 12) yields

$$\int \left|\mu(\mathbf{z} - \tfrac{\mathbf{x}}{\gamma}) - \mu(\mathbf{z} - \tfrac{\mathbf{y}}{\gamma})\right| \mathrm{d}\mathbf{z} \leq \tfrac{1}{\gamma}\left\|\mathbf{x}-\mathbf{y}\right\|_1 \leq \tfrac{\sqrt{n}}{\gamma}\left\|\mathbf{x}-\mathbf{y}\right\|.$$

for the uniform measure. Combining this with the first inequality we obtain that the gradient is Lipschitz-continuous with constant $L_0\sqrt{n}/\gamma$.

We follow similar arguments for the Gaussian measure to obtain

$$\Delta_{\mathbf{z},\mathbf{x},\mathbf{y}} := \int \left|\mu(\mathbf{z} - \tfrac{\mathbf{x}}{\gamma}) - \mu(\mathbf{z} - \tfrac{\mathbf{y}}{\gamma})\right| \mathrm{d}\mathbf{z} \leq \frac{1}{\gamma}\left\|\mathbf{x}-\mathbf{y}\right\|_2 \tag{3.31}$$

and then a Lipschitz constant of L_0/γ. This comes from results found in Duchi et al. (2012) (more precisely see the end of proof of lemma 11 and equation (40) of Duchi et al. (2012)) given there without proofs or details. For completeness, we provide here a quick proof of the inequality. First we need the following inequality, for $a \geq 0$ and $\sigma > 0$,

$$\int_{-\infty}^{\infty} \left|\tfrac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(t-a)^2}{2\sigma^2}} - \tfrac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(t+a)^2}{2\sigma^2}}\right| \mathrm{d}t \leq \frac{4a}{\sqrt{2\pi}\sigma} \leq \frac{2a}{\sigma}. \tag{3.32}$$

This inequality comes from easy calculus as follows:

$$\begin{aligned}\frac{1}{2}\int_{-\infty}^{\infty} \left|e^{-\frac{(t-a)^2}{2\sigma^2}} - e^{-\frac{(t+a)^2}{2\sigma^2}}\right| \mathrm{d}t = & \int_0^{\infty} e^{-\frac{(t-a)^2}{2\sigma^2}} - e^{-\frac{(t+a)^2}{2\sigma^2}} \mathrm{d}t \\ = & \int_{-a}^{\infty} e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t - \int_a^{\infty} e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t \\ = & \int_{-\infty}^{a} e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t - \left(\frac{1}{\sqrt{2\pi}\sigma} - \int_{-\infty}^{a} e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t\right) \\ = & 2\int_{-\infty}^{0} e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t + 2\int_0^a \tfrac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t - \frac{1}{\sqrt{2\pi}\sigma} \\ = & 2\int_0^a e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t \;\leq\; 2\int_0^a 1 \mathrm{d}t \;=\; 2a.\end{aligned}$$

Second we rewrite the left-hand side of 3.31 by two changes of variables: the translation $\mathbf{z} \to \mathbf{z} - \frac{\mathbf{x}+\mathbf{y}}{2\gamma}$

followed by $\mathbf{z} = t\mathbf{v} + \mathbf{w}$, where $t \in \mathbb{R}$, $\mathbf{v} := \frac{\mathbf{y}-\mathbf{x}}{2}$ and $\mathbf{w}$ is in the vector space orthogonal to $\mathbf{v}$. We obtain

$$\Delta_{\mathbf{z},\mathbf{x},\mathbf{y}} = \int_{\mathbb{R}^n} \frac{1}{(\sqrt{2\pi})^n} \left| e^{-\frac{\|\mathbf{v}(t-1)+\mathbf{w}\|^2}{2}} - e^{-\frac{\|\mathbf{v}(t+1)+\mathbf{w}\|^2}{2}} \right| d\mathbf{w} \, \|\mathbf{v}\| \, \mathrm{d}t$$

We can then develop and bound with the help of (3.32) (with $a = 1$ and $\sigma = 1$) as follows

$$\begin{aligned}
\Delta_{\mathbf{z},\mathbf{x},\mathbf{y}} &= \int_{\mathbb{R}^n} \frac{1}{(\sqrt{2\pi})^n} \left| e^{-\frac{\|\mathbf{v}\|^2(t-1)^2+\|\mathbf{w}\|^2}{2}} - e^{-\frac{\|\mathbf{v}\|^2(t+1)^2+\|\mathbf{w}\|^2}{2}} \right| d\mathbf{w} \, \|\mathbf{v}\| \, \mathrm{d}t \\
&= e^{-\frac{\|\mathbf{v}\|^2}{2}} \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}} \left| e^{-\frac{(t-1)^2}{2}} - e^{-\frac{(t+1)^2}{2}} \right| \left(\int_{\mathbb{R}^{n-1}} \frac{1}{(\sqrt{2\pi})^{n-1}} e^{-\frac{\|\mathbf{w}\|^2}{2}} d\mathbf{w} \right) \|\mathbf{v}\| \, \mathrm{d}t \\
&\leq e^{-\frac{\|\mathbf{v}\|^2}{2}} 2 \, \|\mathbf{v}\| \;\leq\; 2 \, \|\mathbf{v}\| \;=\; 2 \left\| \frac{\mathbf{y}-\mathbf{x}}{2\gamma} \right\| \;=\; \tfrac{1}{\gamma} \, \|\mathbf{y}-\mathbf{x}\|
\end{aligned}$$

We finally turn to (ii). Using the fact that $\int \mu(\mathbf{z})\mathrm{d}\mathbf{z} = 1$ and the Lipschitz continuity of g, we write

$$\begin{aligned}
\left| g(\mathbf{x}) - g_\gamma^c(\mathbf{x}) \right| &= \left| \int g(\mathbf{x})\mu(\mathbf{z})\mathrm{d}\mathbf{z} - \int g(\mathbf{x}-\gamma\mathbf{z})\mu(\mathbf{z})\mathrm{d}\mathbf{z} \right| \\
&\leq \int \left| g(\mathbf{x}) - g(\mathbf{x}-\gamma\mathbf{z}) \right| \mu(\mathbf{z})\mathrm{d}\mathbf{z} \;\leq\; \gamma L_0 \int \|\mathbf{z}\| \, \mu(\mathbf{z})\mathrm{d}\mathbf{z}.
\end{aligned}$$

In our two cases μ_2 and μ_∞, the last integral is well-defined and finite, which ends the proof. □

The above result has the same form as Theorem 3.2.1. The assumptions giving the uniform approximation are also similar: a bounded assumption of the subgradients of g for Theorem 3.2.1 vs a Lipschitz assumption of g for Theorem 3.3.1. The main difference between the two results is that the Lipschitz constant of g_γ depends on the one of g here. We notice that the Lipschitz constant also depends on the dimension n of the space.

3.3.2 Simple examples in $\mathbb{R}$ and $\mathbb{R}^n$

As a first example in $\mathbb{R}$, we pursue section 3.2.2 on absolute value. We derive here the smooth approximations of $g(x) = |x|$ by convolution with Gaussian and uniform distributions:

- For the Gaussian density, we get

$$g_\gamma(x) = -xF(-\tfrac{x}{\gamma}) - \tfrac{\sqrt{2}}{\sqrt{\pi}}\gamma e^{-\frac{x^2}{2\gamma^2}} + xF(\tfrac{x}{\gamma}), \tag{3.33}$$

where F is the cumulative distribution function of the Gaussian distribution

$$F(x) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} \mathrm{d}t;$$

- For the uniform density, we get the so-called huberized ℓ_1

$$g_\gamma(x) = \begin{cases} \frac{1}{2\gamma}x^2 + \frac{\gamma}{2} & \text{if } |x| \leq \gamma \\ |x| & \text{if } |x| > \gamma. \end{cases} \tag{3.34}$$

We observe that the eq. (3.34) is a generalzation of the Huber function

$$h(t) = \begin{cases} \frac{1}{2}t^2 + \frac{1}{2} & \text{if } |t| \leq 1 \\ |t| & \text{if } |t| > 1. \end{cases} \tag{3.35}$$

In view of (3.29), we just prove these two expressions for $\gamma = 1$. For (3.33), we split the integral and find

$$\begin{aligned} g_1(x) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} |x - z| \, e^{-\frac{z^2}{2}} \mathrm{d}z \\ =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} (x - z) e^{-\frac{z^2}{2}} \mathrm{d}z - \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} (x - z) e^{-\frac{z^2}{2}} \mathrm{d}z \\ =& \, xF(x) - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} z e^{-\frac{z^2}{2}} \mathrm{d}z - xF(-x) + \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} z e^{-\frac{z^2}{2}} \mathrm{d}z \\ =& xF(x) - \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} - xF(-x) - \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}. \end{aligned}$$

For (3.35), we distinguish 3 cases ($x \leq -1$, $-1 < x < 1$, $x \geq 1$). For example when $-1 < x < 1$, we write

$$\frac{1}{2} \int_{-1}^{1} |x - t| \, \mathrm{d}t = \int_{-1}^{x} (x - t) \, \mathrm{d}t \; + \int_{x}^{1} (-x + t) \, \mathrm{d}t = \left[-\frac{1}{2}t^2 + xt \right]_{-1}^{x} + \left[\frac{1}{2}t^2 - xt \right]_{x}^{1} = x^2 + 1;$$

the two other cases are similar.

In $\mathbb{R}^n$, it is usually more complicated to come up with explicit expressions of the integrals. In general, numerical integration or Monte Carlo sampling could be envisioned, but numerical stability would then be an issue in view of integrating the smoothing techniques within algorithms (see section 3.4). Some special cases of integration are tractable tough, as the case of decomposable function smoothed by uniform

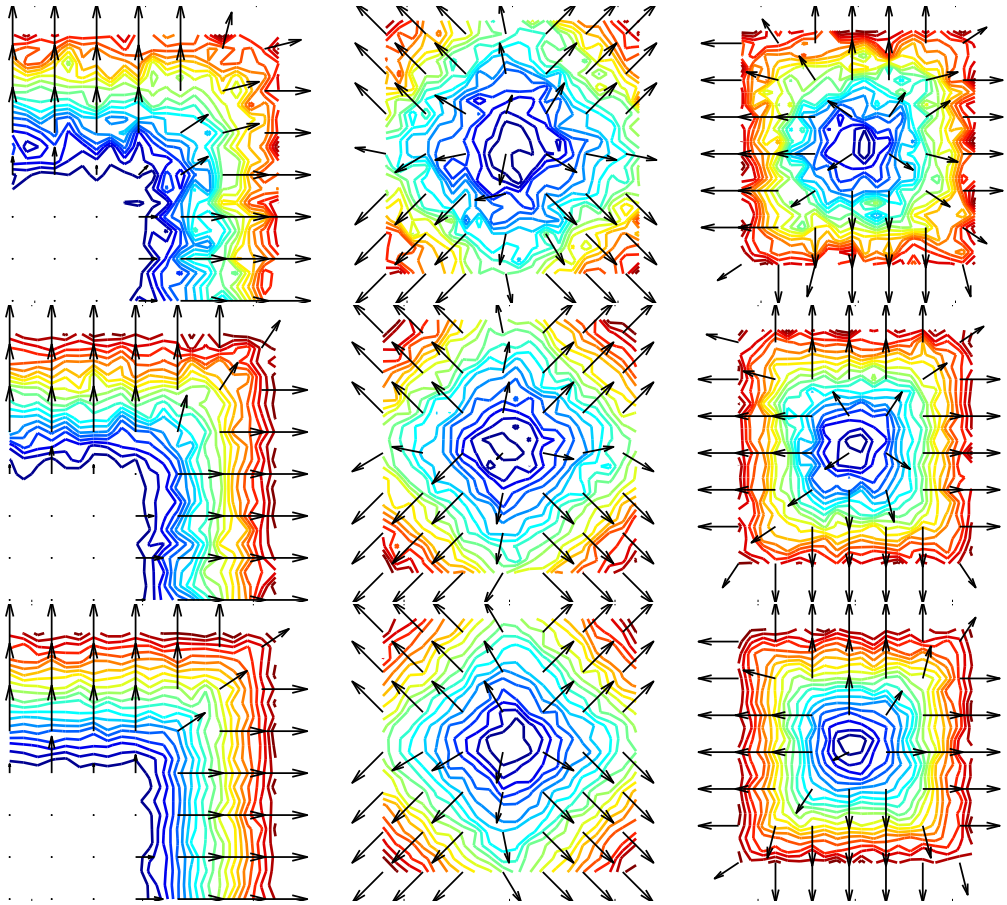

Figure 3.4: Level sets of smooth surrogates obtained with product convolution technique, through uniform random sampling, instead of using an exact formula. Gradients are represented by arrows. **First column** Smoothing of hinge; **second column** Smoothing of $\|\cdot\|_1$; **third column** Smoothing of $\|\cdot\|_\infty$. **First row** Gaussian distribution with mean 0 and standard deviation 1; **second row** uniform distribution on norm ∞ ball, with radius $r = 1$; **third row** uniform distribution on norm 1 ball, with radius $r = 1$.

distribution that boils down to an integration in one dimension.

Proposition 3.3.2. *Let a nonsmooth function be decomposed as*

$$g(\mathbf{x}) = \sum_{i=1}^{n} g^{(i)}(x_i)\,; \tag{3.36}$$

then its smooth approximation for the uniform distribution μ_∞ is

$$g_\gamma(\mathbf{x}) = \sum_{i=1}^{n} g_\gamma^{(i)}(x_i) \tag{3.37}$$

where $g_\gamma^{(i)}$ is the smoothing of $g^{(i)}$ with the uniform distribution on $[-1, 1]$.

Proof. We derive the expression for the case $\gamma = 1$ as follows

$$\begin{aligned} g_1(\mathbf{x}) &= \frac{1}{2^n}\int_{\{\|\mathbf{z}\|_\infty \le 1\}} \sum_{i=1}^{n} g^{(i)}(x_i - z_i)\mathrm{d}\mathbf{z} = \frac{1}{2^n}\sum_{i=1}^{n}\int_{\{\|\mathbf{z}\|_\infty \le 1\}} g^{(i)}(x_i - z_i)\mathrm{d}\mathbf{z} \\ &= \frac{1}{2^n}\sum_{i=1}^{n} 2^{n-1}\int_{\{|z|_i \le 1\}} g^{(i)}(x_i - z_i)\mathrm{d}z_i = \sum_{i=1}^{n}\frac{1}{2}\int_{\{|z|_i \le 1\}} g^{(i)}(x_i - z_i)\mathrm{d}z_i \end{aligned}$$

which gives (3.37) for $\gamma = 1$. The general case follows easily. □

We apply the above result to get the smooth approximation by μ_∞ of the ℓ_1-norm $g(\mathbf{x}) = \|\mathbf{x}\|_1$, where h is defined by (3.35)

$$g_\gamma(\mathbf{x}) = \gamma \sum_{i=1}^{k} h\left(\frac{x_i}{\gamma}\right). \tag{3.38}$$

3.4 Smoothing-based first-order methods for doubly nonsmooth learning problems

In this section, we consider “doubly nonsmooth" optimization problems, that is, composite optimization problems of the form

$$\min_{\mathbf{x}\in E} \quad F(\mathbf{x}) := R(\mathbf{x}) + \lambda\Omega(\mathbf{x}), \tag{3.39}$$

where both R and Ω are convex and nonsmooth. More specifically, we consider the case of learning optimization problems where Ω is a nonsmooth regularizer enforcing “low-complexity" structure, and R

is an empirical risk function featuring a nonsmooth loss function g

$$R(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} g(A_i\mathbf{x} + b_i). \tag{3.40}$$

This framework covers many generic optimization models used in image processing and machine learning. Let us briefly illustrate this with two models in image classification and collaborative filtering, involving nuclear norm as regularization. We recall that the nuclear norm denoted $\|\cdot\|_{\sigma,1}$ (or trace-norm) is defined by the sum of the singular values of a matrix. It is a popular regularizer in machine learning enforcing low-rank solutions and suitable for large-scale applications (by allowing efficient storage of matrices); see e.g. Bach (2008) and Candès and Recht (2009).

Example 12 (Collaborative filtering). Collaborative filtering is a basic problem of machine learning (see e.g. Ekstrand et al. (2011)) and it can be formulated in various ways, for example as a matrix completion problem: Observing only some entries with index $(j,k) \in \mathcal{I}$ of a matrix $\mathbf{X}$, the aim is to guess any unknown entry $\mathbf{X}_{jk}$, with $(j,k) \notin \mathcal{I}$. Some problems require modeling with nuclear norm regularization and absolute value as loss function, so that the collaborative filtering problem is written as

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times k}} \quad \frac{1}{N}\sum_{(j,k)\in\mathcal{I}} |\mathbf{W}_{jk} - \mathbf{X}_{jk}| \;+\; \lambda\,\|\mathbf{W}\|_{\sigma,1}\,,$$

where the optimizing variable is $\mathbf{W}$. By choosing

$$A_i\mathbf{W} + b_i := \mathbf{W}_{j_ik_i} - \mathbf{X}_{j_ik_i}\,, \quad g(s) = |s|\,,$$

this problem is a particular case of our general problem (3.39).

Example 13 (Multiclass classification). Multiclass classification is a second basic learning problems. The observed data are pairs $(\mathbf{x}_i,\, \mathbf{y}_i) \in \mathbb{R}^d \times \mathbb{R}^k$ (objects, labels) and the aim is to predict the class $\mathbf{y}$ of a new example $\mathbf{x}$. One way to model this problem is to use a linear support vector machine, which leads to the optimization problem

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times k}} \quad \frac{1}{N}\sum_{i=1}^{N} \max\{0,\, 1 + \max_{\mathbf{r}\text{ s.t. }\mathbf{r}\neq\mathbf{y}_i}\{\mathbf{W}_{\mathbf{r}}^\top\mathbf{x}_i - \mathbf{W}_{\mathbf{y}_i}^\top\mathbf{x}_i\} + \lambda\,\|\mathbf{W}\|_{\sigma,1}\,,$$

where the j-th column of $\mathbf{W}$ here is indicated with $\mathbf{W}_j \in \mathbb{R}^d$. By choosing

$$A_i\mathbf{W} + b_i := \mathbf{W}^\top\mathbf{x}_i - \mathbf{y}_i + \mathbf{1} - \langle \mathbf{y}_i, \mathbf{W}^\top\mathbf{x}_i\rangle, \quad g(\mathbf{s}) = \max\{0, \max_r\{\mathbf{s}_r\}\}.$$

this problem is a particular case of our general problem (3.39).

When g is smooth, there exist efficient first-order methods to solve learning optimization problem (3.39). For our situation, where g is nonsmooth, an approach is to apply these first-order methods on a "partially smoothed version" of the problem, smoothing the least possible to preserve the desired structural properties on the solutions. With a smooth γ-approximation g_γ of the nonsmooth loss function g, we define the partially smoothed problem

$$\min_{\mathbf{x}\in E} \quad F_\gamma(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} g_\gamma(A_i\mathbf{x}+b_i) + \lambda\Omega(\mathbf{x}), \tag{3.41}$$

that we solve with a first-order method. The question is now if this approach can give approximate solutions of the initial problem (3.39).

The celebrated result of (Nesterov, 2005, Th X) has exactly this form answering positively for the accelerated proximal gradient algorithm and the smoothing by saddle-point representation. This result is generalized in (Beck and Teboulle, 2012, Th 3.1) for any fast first-order methods for specific structured problems and for generic smoothing techniques. We derive here similar results in the context of this chapter for three first-order methods popular in the machine learning community. The theorems show that we can control γ and the number of iterations of the algorithm to obtain ε-solution to the initial doubly nonsmooth problem. This control depends on the following constants:

- the uniform approximation bound m which depends on the smoothing technique (see (3.8) for inf-convolution and (3.30) for convolution), and may depend also on the dimension of the space;
- the Lipschitz constant L depending on the chosen smoothing auxiliary function (ω or μ); by Theorems 3.2.1 and 3.3.1) we have that ∇g_γ is L/γ-smooth;
- the distance from the initial iterate to the optimal solution $D := \|\mathbf{x}_0 - \mathbf{x}_\star\|_2$;
- the average

$$A := \frac{1}{N}\sum_i \|A_i\|\,\|A_i^\dagger\| \tag{3.42}$$

  of the operator norm of matrices A_i and its adjoint $A_i^\dagger$. We recall that the operator norm of A_i defined through the euclidean norm is

$$\|A_i\| := \sup_{\mathbf{x}} \frac{\|A_i\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$$

  and correspond to the square-root of the largest singular value of A_i.

Each of the following three subsections is devoted to an algorithm and follows the same presentation pattern. The common key calculation in the proofs of convergence results is formalized in the following lemma.

Lemma 3.4.1. *Let $a > 0$, $c \geq 0$ and $p > 0$. For b small enough, the minimum*

$$t(\gamma) = \sqrt[p]{\frac{a}{\gamma(b-\gamma)}} - c.$$

over all positive $t(\gamma)$ is $\sqrt[p]{4a/b^2} - c$ obtained for $\gamma = b/2$.

Proof. Minimizing $t(\gamma)$ boils down to maximizing $\gamma(b-\gamma)$ subject to the constraint $t(\gamma) > 0$. Thus the solution is $b/2$ as soon as $t(b/2) > 0$ which reads $4a/b^2 > c^p$. This last equation is satisfied when b is small enough. □

3.4.1 Composite conditional gradient

We consider the case when the regularization Ω admits an easy-to-compute linear minimization oracle, that is, when minimizing a linear function over the "unit ball" associated to Ω is not expensive

$$\operatorname*{argmin}_{\Omega(\mathbf{y})\leq 1} \langle \mathbf{y}, \mathbf{x} \rangle \ . \tag{3.43}$$

For example, linear minimization oracle of the nuclear norm regularization for a matrix $\mathbf{W}$ consists in computing the largest singular value of $\mathbf{W}$ Dudik et al. (2012); Harchaoui et al. (2014).

In this situation, the composite conditional gradient algorithm is a method of choice, especially in large-scale settings. The key computation at each iteration of this algorithm is

$$\operatorname*{argmin}_{\Omega(\mathbf{y})\leq 1} \langle \mathbf{y}, \nabla R(\mathbf{x}_t) \rangle.$$

See Harchaoui et al. (2014); Pierucci et al. (2014) for a complete description of the algorithm.

However, the algorithm cannot be used directly for our doubly-nonsmooth machine learning problem (3.39), as it requires differentiability of R. See Pierucci et al. (2014) for a counter-example. We show here that applied to a smooth surrogate of R, the algorithm does bring a solution of (3.39).

Theorem 3.4.2. *For any $\varepsilon > 0$; the composite conditional gradient algorithm applied to the partially smoothed problem* (3.41) *with $\gamma = \frac{\varepsilon}{4m}$ produces an ε-solution of the initial nonsmooth problem* (3.39)

after

$$\frac{64mLAD^2}{\varepsilon^2} \quad \textit{iterations.}$$

Proof. (Of Theorem 3.4.2) The proof is based on the following splitting into 3 parts:

$$F(\mathbf{x}_t) - \min_{\mathbf{x}} F(\mathbf{x}) \leq |F(\mathbf{x}_t) - F_\gamma(\mathbf{x}_t)| \tag{3.44}$$

$$+ \left| F_\gamma(\mathbf{x}_t) - \min_{\mathbf{x}} F_\gamma(\mathbf{x}) \right| \tag{3.45}$$

$$+ \left| \min_{\mathbf{x}} F_\gamma(\mathbf{x}) - \min_{\mathbf{x}} F(\mathbf{x}) \right| \tag{3.46}$$

Let us bound each of these three terms in our case. The first term is bounded with the help of the uniform approximation of the smoothing function, thanks to (3.8), as follows:

$$|F(\mathbf{x}_t) - F_\gamma(\mathbf{x}_t)| = |R(\mathbf{x}_t) + \lambda\Omega(\mathbf{x}_t) - R_\gamma(\mathbf{x}_t) - \lambda\Omega(\mathbf{x}_t)| = |R(\mathbf{x}_t) - R_\gamma(\mathbf{x}_t)|$$
$$\leq \frac{1}{N}\sum_{i=1}^{N} |g(A_i\mathbf{x}_t + b_i) - g_\gamma(A_i\mathbf{x}_t + b_i)| \leq \gamma m \ .$$

The second term (3.45) is the difference between the current iteration to the minimum for the smooth problem; this difference is controlled by the convergence result for composite conditional gradient of (Harchaoui et al., 2014, Theorem 3)

$$F_\gamma(\mathbf{x}_t) - \min_{\mathbf{x}} F_\gamma(\mathbf{x}) \leq \frac{8L_{R_\gamma}D^2}{t+14},$$

where L_{R_γ} is the Lipschitz constant of ∇R_γ. We have

$$\nabla R_\gamma(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} A_i^\dagger \nabla g_\gamma(A_i\mathbf{x} + b_i) \ .$$

We obtain

$$L_{R_\gamma} = LA/\gamma \ ,$$

where A is defined at (3.42).

Finally using again the uniform approximation of smoothed function, (3.8), we get a bound for (3.46), as follows. For all $\mathbf{x} \in E$ and all $i \in 1, \ldots, N$, we have

$$|g_\gamma(A_i\mathbf{x} + b_i) - g(A_i\mathbf{x} + b_i)| \leq \gamma m \ .$$

Summing over i, we get

$$g(A_i\mathbf{x} + b_i) - \gamma m \leq g_\gamma(A_i\mathbf{x} + b_i) \leq g(A_i\mathbf{x} + b_i) + \gamma m$$

and then

$$R_\gamma(\mathbf{x}) + \lambda\Omega(\mathbf{x}) - \gamma m \leq R_\gamma(\mathbf{x}) + \lambda\Omega(\mathbf{x}) \leq R_\gamma(\mathbf{x}) + \lambda\Omega(\mathbf{x}) + \gamma m \ .$$

Minimizing over $\mathbf{x}$, we therefore get

$$\min_{\mathbf{x}} F(\mathbf{x}) - \gamma m \leq \min_{\mathbf{x}} F_\gamma(\mathbf{x}) \leq \min_{\mathbf{x}} F(\mathbf{x}) + \gamma m \ .$$

Finally, putting the three bounds together, we end up with

$$F(\mathbf{x}_t) - \min_{\mathbf{x}} F(\mathbf{x}) \leq 2\gamma m + \frac{8LAD^2}{\gamma(t+14)} \ .$$

The iterate $\mathbf{x}_t$ is thus guaranteed to be a ε-solution when

$$2\gamma m + \frac{8LAD^2}{\gamma(t+14)} \leq \varepsilon \quad \text{i.e.} \quad \gamma + \frac{\frac{4LAD^2}{m}}{\gamma(t+14)} \leq \frac{\varepsilon}{2m} \ . \tag{3.47}$$

Apply now Lemma 3.4.1 with $a = \frac{4LAD^2}{m}$, $b = \varepsilon/2m$, $p = 1$, $c = 14$ and to get the optimal $\gamma_\star = \varepsilon/4m$. Substituting back in (3.47), we obtain

$$2\tfrac{\varepsilon}{4m}m + \frac{8LAD^2}{\frac{\varepsilon}{4m}(N+14)} \leq \varepsilon \iff \frac{64mLAD^2}{\varepsilon^2} - 14 \leq N \ .$$

which ends the proof. □

3.4.2 Accelerated proximal gradient algorithm

We now consider the case when the regularization Ω admits a proximal setup, which means that it is possible to compute in an efficient way the proximal operator

$$\mathrm{prox}_{k\Omega}(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{z}} \left\{ \Omega(\mathbf{z}) + \frac{1}{2k} \|\mathbf{z} - \mathbf{x}\|^2 \right\} \tag{3.48}$$

related to Ω. For example, proximal point of (block) ℓ_1 regularization can be computed in closed form Bach et al. (2012b).

In this situation, the (accelerated) proximal gradient algorithms are the method of choice; among them

we consider the celebrated Fast Iterative Thresholding Algorithm (FISTA) algorithm Beck and Teboulle (2012). The key computation at each iteration of this algorithm is

$$\mathrm{prox}_{\delta_t \Omega}(\mathbf{x}_t - \delta_t \nabla R(\mathbf{x}_t)),$$

where δ_t is the stepsize chosen according to some rules and depends on a constant $\alpha \geq 1$, which is 1 or the backtracking constant of FISTA (Beck and Teboulle, 2012, Algorithms at Section 4); we refer to this chapter for a complete description of the algorithm. We show here that applied to a smooth surrogate of R, the algorithm does bring a solution of (3.39).

Theorem 3.4.3. *For $\varepsilon > 0$ small enough; the accelerated proximal gradient algorithm applied the partially smoothed problem* (3.41) *with $\gamma = \frac{\varepsilon}{4m}$ produces an ε-solution of the initial nonsmooth problem* (3.39) *after*

$$\frac{4D\sqrt{m\alpha LA}}{\varepsilon} - 1 \quad \textit{iterations.}$$

Proof. (Of Theorem 3.4.3) As for Theorem 3.4.2, the proof is based on the splitting (3.44)-(3.45)-(3.46), and the bounds for (3.44) (3.46) depending on the smoothing are the same. The bound for (3.45) depends on the algorithm and we get here:

$$F_\gamma(\mathbf{x}_t) - \min_{\mathbf{x}} F_\gamma(\mathbf{x}) \leq \frac{2\alpha LAD^2}{\gamma(t+1)^2},$$

by applying the convergence result of FISTA (Beck and Teboulle, 2009, Theorem 4.4) for the smooth R_γ. Thus we have

$$F(\mathbf{x}_t) - \min_{\mathbf{x}} F(\mathbf{x}) \leq 2\gamma m + \frac{2\alpha LAD^2}{\gamma(t+1)^2}.$$

The iterate $\mathbf{x}_t$ is thus guaranteed to be an ε-solution of (3.39) when

$$2\gamma m + \frac{2\alpha L_\gamma D^2}{(t+1)^2} \leq \varepsilon \quad \text{i.e.} \quad \gamma + \frac{\frac{\alpha LAD^2}{m}}{\gamma(t+1)^2} \leq \frac{\varepsilon}{2m}. \tag{3.49}$$

Apply now Lemma 3.4.1 with $a = \frac{\alpha LAD^2}{m}$, $b = \varepsilon/2m$, $p = 2$, $c = 1$ to get the optimal $\gamma = \varepsilon/4m$ when ε is small enough so that $\frac{4\alpha LAD^2}{m} > \frac{\varepsilon^2}{4m^2}$. Substituting back in (3.49), we obtain

$$2\frac{\varepsilon}{4m}m + \frac{8m\alpha LAD^2}{\varepsilon(N+1)^2} \leq \varepsilon \iff \frac{4D\sqrt{m\alpha LA}}{\varepsilon} - 1 \leq N.$$

which ends the proof. □

3.4.3 Incremental gradient

In this section, we consider an incremental gradient algorithm assuming a finite-sum structure of R. The algorithm called MISO (for Minimization by Incremental Surrogate Optimization) (Mairal, 2013, Algorithm 5) is an algorithm of this kind, tailored for minimizing large finite sums of convex objectives. This algorithm uses strongly convex surrogates for the terms of the sum of (3.40). We explain here how we can combine them with smoothing.

After an initialization phase, the iterations of the MISO algorithm has two steps. First it picks up randomly one index j, it chooses a strongly convex surrogate h_t^j near $\mathbf{x}_{t-1}$, by keeping unchanged all the other surrogates $h_t^i := h_{t-1}^i$ for $i \neq j$. The next iterate is computed by solving a subproblem, where the key computation is

$$\operatorname*{argmin}_{\mathbf{x}\in E} \frac{1}{N}\sum_{i=1}^{N} h_t^i(\mathbf{x}). \tag{3.50}$$

We refer to (Mairal, 2013, Algorithm 5) for a more precise description of the algorithm.

In our situation with a nonsmooth loss function, we combine the construction of the surrogate of MISO with smoothing to be able to apply this algorithm. Hence we build a local surrogate h_t^i of $f^i(\mathbf{x}) := g_\gamma(A_i\mathbf{x}+b_i)+\lambda\Omega(\mathbf{x})$ satisfying the two required properties: first, the surrogates h_t^i are (local) majorants near the iterate $\mathbf{x}_{t-1}$

$$h^i(\mathbf{x}) \geq g_\gamma(A_i\mathbf{x}+b_i) \quad \text{for } \mathbf{x} \text{ near } \bar{\mathbf{x}},$$

and second the function $h_t^i - f^i$ is smooth and strongly convex.

We define for all $\mathbf{x}$ near $\bar{\mathbf{x}}$

$$h^i(\mathbf{x}) := g_\gamma(A_i\bar{\mathbf{x}}+b_i) + \langle A_i^\dagger \nabla g_\gamma(A_i\bar{\mathbf{x}}+b_i), \mathbf{x}-\bar{\mathbf{x}}\rangle + \frac{LB}{2\gamma}\|\mathbf{x}-\bar{\mathbf{x}}\|_2^2 + \lambda\Omega(\mathbf{x}), \tag{3.51}$$

where $B := \max_{i=1\ldots N}\|A_i\|^2$ is a bound on the Lipschitz constants of all the gradients $\nabla g_\gamma(A_i\mathbf{x}+b_i)$. We observe that h^i satisfies the two properties above (see also (Mairal, 2013, First paragraph of Section 2.2)). Moreover all the surrogates h_i have the same Lipschitz constant as their gradients, that is also requested for convergence result of (Mairal, 2013, Proposition 6.2). We now apply the MISO algorithm to get a solution for the initial nonsmooth problem. The following convergence result is similar to the two previous ones up to an additional expectation, since MISO is a random algorithm.

Theorem 3.4.4. *For any $\varepsilon > 0$; the MISO algorithm applied the partially smoothed problem* (3.41) *with*

$\gamma = \frac{\varepsilon}{4m}$ *is expected to produces an solution of the initial nonsmooth problem* (3.39) *such that*

$$\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}_\star)] \leq \varepsilon \tag{3.52}$$

after

$$\frac{4mBLMD^2}{\varepsilon^2} \quad \textit{iterations.}$$

Proof. (Of Theorem 3.4.4) We repeat the same splitting as in the proof of Theorem 3.4.3, with addition expectation. The difference is in the second term which is controlled by the convergence result of (Mairal, 2013, Proposition 6.2):

$$\mathbb{E}[g_\gamma(\mathbf{x}_t) - \min_{\mathbf{x}} g_\gamma(\mathbf{x})] \leq \frac{BLMD^2}{2t}$$

which gives

$$\mathbb{E}\left[F(\mathbf{x}_t) - \min_{\mathbf{x}} F(\mathbf{x})\right] \leq 2\gamma m + \frac{BLND^2}{2\gamma t}.$$

The iterate $\mathbf{x}_t$ is thus guaranteed to satisfy (3.52) when

$$2\gamma m + \frac{BLMD^2}{2\gamma t} \leq \varepsilon \quad \text{i.e.} \quad \gamma + \frac{\frac{BLMD^2}{4m}}{\gamma t} \leq \frac{\varepsilon}{2m}.$$

Apply now Lemma 3.4.1 with $a = \frac{BLMD^2}{4m}$, $b = \varepsilon/2m$, $p = 1$, $c = 0$ we get $\gamma_\star = \varepsilon/4m$. Substituting back we obtain

$$2\frac{\varepsilon}{4m}m + \frac{4mBLMD^2}{2\varepsilon N} \leq \varepsilon \iff \frac{4mBLMD^2}{\varepsilon^2} \leq N.$$

which ends the proof. □

3.5 Algebraic calculus

In the next two sections we introduce some useful rules that can be used as tools to combine and generate new smooth surrogates.

3.5.1 Fenchel-type approximation

It is useful to have some calculus rules to construct the approximations g_γ. In this section we deal with the operator $\mathcal{T}(\cdot)$ that generates a Fenchel-type γ-approximation:

$$\mathcal{T}(d)(\mathbf{x}) := \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{x}, \mathbf{z} \rangle - d(\mathbf{z})$$

with a fixed $\mathcal{Z}$ and d a convex function. With this definition we get again the smooth approximation

$$g_\gamma(\mathbf{x}) = \mathcal{T}\left(\gamma d\right)(\mathbf{x}).$$

For all $\gamma > 0$, $b \in \mathbb{R}$ and $\mathbf{k} \in \mathbb{R}^n$, $\alpha \in [0,1]$, we get the next rules for scaling, translation, minimum and two properties of the operator $\mathcal{T}\left(\cdot\right)$:

$$\mathcal{T}\left(\gamma d\right)(\mathbf{x}) = \gamma \mathcal{T}\left(d\right)\left(\frac{\mathbf{x}}{\gamma}\right) \tag{3.53}$$

$$\mathcal{T}\left(d + \langle \cdot, \mathbf{k}\rangle + b\right)(\mathbf{x}) = \mathcal{T}\left(d\right)(\mathbf{x} - \mathbf{k}) - b. \tag{3.54}$$

$$\min_{\mathbf{z}\in\mathcal{Z}} d(\mathbf{z}) = -\mathcal{T}\left(d\right)(0) \tag{3.55}$$

$$f > g \text{ on } \mathcal{Z} \Longrightarrow \mathcal{T}\left(\gamma f\right) < \mathcal{T}\left(\gamma g\right) \text{ on } \mathbb{R}^n \quad \text{anti-monotonicity} \tag{3.56}$$

$$\mathcal{T}\left(\alpha f + (1-\alpha) g\right) \leq \alpha \mathcal{T}\left(f\right) + (1-\alpha)\mathcal{T}\left(g\right) \quad \text{convexity} \tag{3.57}$$

By deriving the (3.53), we obtain the gradient

$$\nabla g_\gamma(\mathbf{x}) = \nabla g_1\left(\tfrac{\mathbf{x}}{\gamma}\right). \tag{3.58}$$

Proof. (Of equation (3.53)) We just develop from the definitions:

$$\mathcal{T}\left(\gamma d\right)(\mathbf{x}) = \max_{\mathbf{z}\in\mathcal{Z}} \langle \mathbf{x}, \mathbf{z}\rangle - \gamma d(\mathbf{z}) = \gamma \max_{\mathbf{z}\in\mathcal{Z}} \tfrac{1}{\gamma}\langle \mathbf{x}, \mathbf{z}\rangle - d(\mathbf{z}) = \gamma \mathcal{T}\left(d\right)\left(\frac{\mathbf{x}}{\gamma}\right)$$

□

Proof. (Of equation (3.54)) Addition of an affine map to the function d.

$$\begin{aligned}
\mathcal{T}\left(d + \langle \cdot, \mathbf{k}\rangle + b\right)(\mathbf{x}) &= \max_{\mathbf{z}\in\mathcal{Z}} \langle \mathbf{x}, \mathbf{z}\rangle - d(\mathbf{z}) - \langle \mathbf{z}, \mathbf{k}\rangle - b \\
&= \max_{\mathbf{z}\in\mathcal{Z}} \langle \mathbf{x} - \mathbf{k}, \mathbf{z}\rangle - d(\mathbf{z}) - b \\
&= \mathcal{T}\left(d\right)(\mathbf{x} - \mathbf{k}) - b.
\end{aligned}$$

□

Proof. (*Of equation* (3.55))

$$m := \min_{\mathbf{z}\in\mathcal{Z}} d(\mathbf{z}) = -\max_{\mathbf{z}\in\mathcal{Z}} -d(\mathbf{z}) = -\max_{\mathbf{z}\in\mathcal{Z}} \langle 0, \mathbf{z}\rangle - d(\mathbf{z}) = -\mathcal{T}\left(d\right)(0)$$

□

Proof. (*Of equation* (3.56))

$$\begin{aligned}
f(\mathbf{z}) > g(\mathbf{z}) \Rightarrow \quad & \gamma f(\mathbf{z}) > \gamma g(\mathbf{z}) \\
\Rightarrow & -\langle \mathbf{x}, \mathbf{z} \rangle + \gamma f(\mathbf{z}) > -\langle \mathbf{x}, \mathbf{z} \rangle + \gamma g(\mathbf{z}) \\
\Rightarrow \quad & \langle \mathbf{x}, \mathbf{z} \rangle - \gamma f(\mathbf{z}) < \langle \mathbf{x}, \mathbf{z} \rangle - \gamma g(\mathbf{z}) \\
\Rightarrow \quad & \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{x}, \mathbf{z} \rangle - \gamma f(\mathbf{z}) < \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{x}, \mathbf{z} \rangle - \gamma g(\mathbf{z}) \\
\Rightarrow \quad & \mathcal{T}(\gamma f)(\mathbf{x}) < \mathcal{T}(\gamma g)(\mathbf{x})
\end{aligned}$$

□

Proof. (*Of equation* (3.57)) The results come easily from the definitions:

$$\begin{aligned}
\mathcal{T}(\alpha f + (1-\alpha) g) &= \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, \mathbf{x} \rangle - \alpha f(\mathbf{z}) - (1-\alpha) g(\mathbf{z}) \\
&= \max_{\mathbf{z}, \mathbf{y} \in \mathcal{Z}, \mathbf{z} = \mathbf{y}} \alpha(\langle \mathbf{z}, \mathbf{x} \rangle - f(\mathbf{z})) + (1-\alpha)(\langle \mathbf{y}, \mathbf{x} \rangle - g(\mathbf{y})) \\
&\leq \max_{\mathbf{z}, \mathbf{y} \in \mathcal{Z}} \alpha(\langle \mathbf{z}, \mathbf{x} \rangle - f(\mathbf{z})) + (1-\alpha)(\langle \mathbf{y}, \mathbf{x} \rangle - g(\mathbf{y})) \\
&= \alpha \max_{\mathbf{z} \in \mathcal{Z}} \{\langle \mathbf{z}, \mathbf{x} \rangle - f(\mathbf{z})\} + (1-\alpha) \max_{\mathbf{y} \in \mathcal{Z}} \{\langle \mathbf{y}, \mathbf{x} \rangle - g(\mathbf{y})\} \\
&= \alpha \mathcal{T}(f) + (1-\alpha) \mathcal{T}(g)
\end{aligned}$$

□

3.5.2 Product convolution approximation

Here we consider the generation of a smooth surrogate of g as a transform:

$$\mathcal{S}(g)(\mathbf{x}) := \int g(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z},$$

which is the same definition of (3.28), but with a more comfortable notation to see this properties.

Then we have

$$\mathcal{S}(g + \tau) = \mathcal{S}(g) + \mathcal{S}(\tau) \tag{3.59}$$

$$\mathcal{S}(kg) = k\mathcal{S}(g) \qquad \text{for } k > 0 \tag{3.60}$$

$$\mathcal{S}(\langle \cdot, \mathbf{k} \rangle + b) = \langle \cdot, \mathbf{k} \rangle + b \tag{3.61}$$

$$\mathcal{S}(\max\{g_1, g_2\}) = \mathcal{S}(g_1 i_{U_1}) + \mathcal{S}(g_2 i_{U_2}) \tag{3.62}$$

where $U_r := \left\{ \mathbf{x} \in \mathbb{R}^n \,\middle|\, r = \operatorname{argmax}_j \{g_j(\mathbf{x})\} \right\}$ and Eq. (3.61) is the invariance on affine functions (symmetric μ). To prove (3.59) and (3.60) is immediate, we show here only the next proofs.

Proof. (Of equation (3.61)) Under the hypothesis of symmetry

$$\forall \mathbf{k} \in \mathbb{R}^n \quad \int_{\mathbb{R}^n} \langle \mathbf{z}, \mathbf{k} \rangle \mu(\mathbf{z}) \mathrm{d}\mathbf{z} = 0$$

we get

$$\begin{aligned}
\mathcal{S}\left(\langle \cdot, \mathbf{k} \rangle + b\right) &= \int \left(\langle \mathbf{x} - \mathbf{z}, \mathbf{k} \rangle + b\right) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int \langle \mathbf{x} - \mathbf{z}, \mathbf{k} \rangle \mu(\mathbf{z}) \mathrm{d}\mathbf{z} + b \int \mu(\mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int \langle \mathbf{x}, \mathbf{k} \rangle \mu(\mathbf{z}) \mathrm{d}\mathbf{z} - \int \langle \mathbf{z}, \mathbf{k} \rangle \mu(\mathbf{z}) \mathrm{d}\mathbf{z} + b \\
&= \langle \mathbf{x}, \mathbf{k} \rangle \int \mu(\mathbf{z}) \mathrm{d}\mathbf{z} + b \\
&= \langle \mathbf{x}, \mathbf{k} \rangle + b
\end{aligned}$$

□

Proof. (Of equation (3.62)) We just separate the integral into the two subsets where $\max\{g_1, g_2\}$ is maximized.

$$\begin{aligned}
&\mathcal{S}\left(\max\{g_1, g_2\}\right)(\mathbf{x}) = \int_{\mathbb{R}^n} \max\{g_1(\mathbf{x} - \mathbf{z}), g_2(\mathbf{x} - \mathbf{z})\} \mu(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int_{\mathbf{x} - \mathbf{z} \in U_1} \max\{g_1, g_2\}(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} + \int_{\mathbf{x} - \mathbf{z} \in U_2} \max\{g_1, g_2\}(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int_{\mathbf{x} - \mathbf{z} \in U_1} g_1(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} + \int_{\mathbf{x} - \mathbf{z} \in U_2} g_2(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int_{\mathbf{x} - \mathbf{z} \in U_1} g_1(\mathbf{x} - \mathbf{z}) i_{U_1}(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} + \int_{\mathbf{x} - \mathbf{z} \in U_2} g_2(\mathbf{x} - \mathbf{z}) i_{U_2}(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int_{\mathbb{R}^n} g_1(\mathbf{x} - \mathbf{z}) i_{U_1}(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} + \int_{\mathbb{R}^n} g_2(\mathbf{x} - \mathbf{z}) i_{U_2}(\mathbf{x} - \mathbf{z}) \mu(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \mathcal{S}\left(g_1 i_{U_1}\right) + \mathcal{S}\left(g_2 i_{U_2}\right)
\end{aligned}$$

□

3.6 Smoothing of SVM with reject option

This section is motivated by a version of support vector machine for binary classification. This type of SVM is particular because, when the classification is ambiguous, the prediction is rejected, instead of returning one of the two classes. The management of ambiguity is motivated when a wrong prediction leads to dangerous consequences. For example, suppose that in a clinical test it is ambiguous if the patient is ill. A classifier that rejects to classify and that advises the doctor to make more detailed analysis is then safer for the patient.

We notice that this is not a 3-class classification, in fact the label `reject` is not an observation. In this section we present a way to find a smooth γ-approximation of the loss for SVM with reject option Grandvalet et al. (2009). This loss is a piecewise affine convex function (PAC), therefore we are going to show in the next sections how to write a PAC as combination of support functions and how to smooth it.

3.6.1 Smoothing of piecewise affine convex functions

The aim of these sections is to find smooth γ-approximations of any piecewise affine convex function (PAC).

We define now a general form of a PAC. Let us take the real points

$$\{t_{-m}, \dots, t_{-1}, t_1, \dots, r_n\}$$

in $\mathbb{R}$, where $t_{-1} \leq t_{+1}$ and for the other indices $i < j$ we have $t_i < t_j$; possibly $t_{-1} = t_1$.

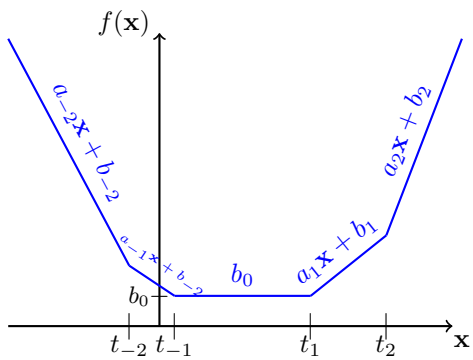

Figure 3.5: Piecewise affine function

Any PAC $f : \mathbb{R} \to \mathbb{R}$ has a general form

$$f(\mathbf{x}) := \begin{cases} a_{-m}\mathbf{x} + b_{-m} & \mathbf{x} < t_{-m} \\ a_i\mathbf{x} + b_i & t_{i-1} \leq \mathbf{x} < t_i \quad i = (-m+1), \dots, -1 \\ b_0 & t_{-1} \leq \mathbf{x} \leq t_1 \\ a_i\mathbf{x} + b_i & t_i < \mathbf{x} \leq t_i + 1 \quad i = 1, \dots, (n-1) \\ a_n\mathbf{x} + b_n & t_n < \mathbf{x}, \end{cases} \tag{3.63}$$

where we suppose that all the coefficients a_i and b_i are such that f is continuous.

When $a_{-m} \leq \cdots \leq a_i \leq \cdots \leq a_n$ this function f is also convex. When $t_{-1} = t_1$ then f is strictly convex.

We observe that even if $t_{-1} = t_1$, there are $N = m + n$ pieces with nonzero steepness, and the minimum of f is b_0. Represented in Figure 3.5 an example with $n = m = 2$.

In the next we see two ways to smooth a PAC: the first decomposing it as max of affine functions and

using the inf-conv smoothing technique, second way by decomposing it as sum of hinge functions and using Fenchel-type smooth γ-approximation .

A way to write a convex PAC (3.63) is as the maximum of affine functions

$$f(\mathbf{x}) = \max\{a_{-m}\mathbf{x} + b_{-m}, \ldots, b_0, \ldots, a_n\mathbf{x} + b_n\} \tag{3.64}$$

with $a_0 = 0$. The proof of (3.64) is evident because f is convex and defined on $\mathbb{R}$.

Smoothing of a PAC via infimal convolution (IC)

Let a convex PAC f be written as (3.64), then

$$f^{\gamma}(\mathbf{x}) = \gamma \log\left(\sum_{i=-m}^{n} e^{\frac{1}{\gamma}(a_i\mathbf{x}+b_i)}\right) \tag{3.65}$$

is a Fenchel-type L^{γ}-smooth γ-approximation of g, with Lipschitz constant

$$L^{\gamma} = \tfrac{1}{\gamma} \max_{i=-m\ldots n} |a_i| \,. \tag{3.66}$$

The proof of (3.65) follows the route of Sec. 3.2.3 extended for $\mathbf{x} \in \mathbb{R}^n$. For the value of L one can see Example 4.5 in Beck and Teboulle (2012).

We saw that a way to smooth a PAC is to write it as max of affine functions. Despite this result is general, it hides the nondifferentiable points t_i. Writing a PAC in a form where the points t_i are explicit is useful to be applied for SVM with reject. As explained in Grandvalet et al. (2009), the values of t_i are used to tune the objective function. We propose an additional decomposition for a PAC that leads to a smoothing and keeps explicit the values of t_i.

Proposition 3.6.1. *Let the PAC defined at* (3.63) *be convex. Then it can be written as a sum of supports functions compound by affine functions*

$$f(x) = b_0 + \sum_{i=-1}^{-m} \sigma((a_i - a_{i+1})(x - t_i)) + \sum_{i=1}^{n} \sigma((a_i - a_{i-1})(x - t_i)) \tag{3.67}$$

where σ is the support function of the interval $[0, 1]$ *which is the hinge function*

$$\sigma(x) := \max_{y\in[0,1]} yx = \begin{cases} 0 & x \leq 0 \\ x & x > 0. \end{cases} \tag{3.68}$$

With the new notation we rewrite (3.67) *as*

$$f(x) = b_0 + \sum_{i=-m}^{n} f_i(x),$$

where we defined

$$f_i(x) := \sigma(\mathcal{A}_i x) := \sigma((a_i - a_{i-\mathrm{sign}(i)})(x - t_i)).$$

An immediate consequence is that the f_i have smooth γ-approximation f_i^γ with properties

(i) the Lipschitz constant of ∇f_i^γ is

$$L_i^\gamma := \tfrac{1}{\gamma} \left| a_i - a_{i-\mathrm{sign}(i)} \right|, \tag{3.69}$$

(ii) the bounds for f_i^γ are

$$\gamma m_i \leq f_i(x) - f_i^\gamma(x) \leq \gamma M_i,$$

where m_i and M_i are finite.

The next proposition shows how it is possible to find a smoothing f^γ of a convex PAC f using the Fenchel-type smooth γ-approximation of each sub-unit f_i.

Proposition 3.6.2. *The function*

$$f^\gamma(x) := b_0 + \sum_{i=-m}^{n} f_i^\gamma(x) \tag{3.70}$$

is a L^γ-smooth γ-approximation of f, with Lipschitz constant

$$L^\gamma = \sum_{i=-m}^{n} L_i^\gamma = \frac{|a_{-m}| + |a_n|}{\gamma}. \tag{3.71}$$

Proof. (Of Proposition 3.6.2) For each f_i we have

$$\gamma m_i \leq f_i(x) - f_i^\gamma(x) \leq \gamma M_i.$$

By summing up we obtain

$$\sum_i \gamma m_i \leq \sum_i f_i(x) - \sum_i f_i^\gamma(x) \leq \sum_i \gamma M_i$$

$$\gamma \sum_i m_i \leq b_0 + \sum_i f_i(x) - b_0 - \sum_i f_i^\gamma(x) \leq \gamma \sum_i M_i.$$

We define $\bar{m} := \sum_i m_i$ and $\bar{M} := \sum_i M_i$ and then

$$\gamma\bar{m} \leq f(x) - f^\gamma(x) \leq \gamma\bar{M}.$$

In addition f^γ is differentiable by construction. We find now the Lipschitz constant (3.71).

$$\begin{aligned}
\|\nabla f^\gamma(x) - \nabla f^\gamma(y)\| &= \left\| \sum_i \nabla f_i^\gamma(x) - \sum_i \nabla f_i^\gamma(y) \right\| \\
&\leq \sum_i \|\nabla f_i^\gamma(x) - \nabla f_i^\gamma(y)\| \\
&\leq \sum_i L_i^\gamma \|x - y\| .
\end{aligned}$$

We get that $L^\gamma = \sum_{i=-m}^n L_i^\gamma$. Now we use (3.69) and consider that $a_i - a_{i-\mathrm{sign}(i)}$ is positive for $i > 0$ and negative for $i < 0$. Then

$$\begin{aligned}
\gamma \sum_{i=-m}^n L_i^\gamma &= \sum_{i=-m}^n \left| a_i - a_{i-\mathrm{sign}(i)} \right| \\
&= \sum_{i=-m}^0 -a_i + a_{i+1} + \sum_{i=0}^n a_i - a_{i-1} \\
&= -a_{-m} + a_n \\
&= |a_{-m}| + |a_n| .
\end{aligned}$$

□

3.6.2 Smoothing the SVM with reject

In the previous section we have seen how to find two smooth γ-approximation of a piecewise affine function. In this section we apply this results to the loss used for SVM with reject.

The binary SVM with reject is introduced to take care of the points that are close to the decision boundary between positive examples and negative examples. Instead of predicting always $+1$ or -1, the classifier may abstain to classify ambiguous observations and alert the user. To reject some examples can be useful to avoid expensive misclassifications. "For instance, in clinical trials it is important be able to reject a tumor diagnostic classification since the consequences of misdiagnosis are severe and scientific expertise is required to make reliable determination" Bartlett and Wegkamp (2008).

It has been proposed a double hinge loss Bartlett and Wegkamp (2008) and a generalization to arbi-

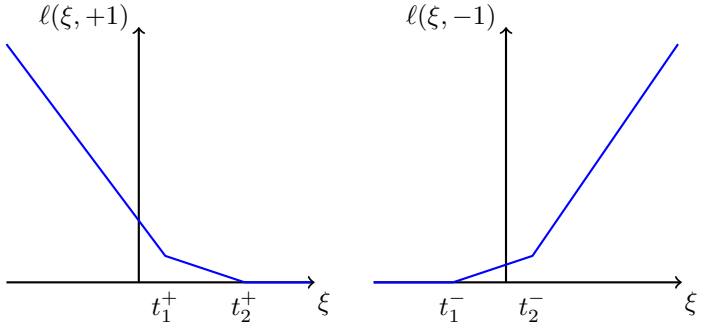

Figure 3.6: SVM with reject loss. **Left** for label $+1$; **Right** for label -1. We notice that the positions of t_1 and t_2 are different in the 2 cases. In addition $t_1^+ = t_2^-$.

trary asymmetric misclassification and reject costs Grandvalet et al. (2009).

Piecewise affine loss For examples with label -1

$$\ell(\xi, -1) = \begin{cases} 0 & \xi \leq t_1^- \\ p_-\xi + H(p_-) & t_1^- < \xi \leq t_2^- \\ p_+\xi + H(p_+) & t_2^- < \xi, \end{cases}$$

where $H(p) = -p\log(p) - (1-p)\log(1-p)$.

For examples with label $+1$

$$\ell(\xi, +1) = \begin{cases} (p_+ - 1)\xi + H(p_+) & \xi \leq t_1^+ \\ (p_- - 1)\xi + H(p_-) & t_1^+ < \xi \leq t_2^+ \\ 0 & t_2^+ < \xi \end{cases}$$

We use here the same notation of Grandvalet et al. (2009) When the predicted labels are $+1$ or -1, the costs of misclassification are $c_- > 0$ and $c_2 > 0$. When on the other hand the classifier decide to abstain and reject the example, the costs $r_+ > 0$ and $r_- > 0$. This problem become interesting where the costs c_+ and c_- are expensive, such as in medical decision making. In these processes it might be better to alert the user and abstain from prediction Grandvalet et al. (2009).

The costs of reject r_+ and r_- are supposed to be smaller than the misclassification costs c_+ and c_-; this is described in the assumption $c_-r_+ + c_+r_- < c_-c_+$.

With these definitions

$$p_+ = \frac{c_- - r_-}{c_- - r_- + r_+}$$

$$p_- = \frac{r_-}{c_+ - r_+ + r_-}$$

$$f_+ = \log(p_+) - \log(1 - p_+)$$

$$f_- = \log(p_-) - \log(1 - p_-)$$

we have

$$t_1^- = \frac{-H(p_-)}{p_-} \tag{3.72}$$

$$t_2^- = \frac{H(p_+) - H(p_-)}{p_- - p_+} \tag{3.73}$$

$$t_1^+ = \frac{H(p_+) - H(p_-)}{p_- - p_+} \tag{3.74}$$

$$t_2^+ = \frac{H(p_-)}{1 - p_-}\ , \tag{3.75}$$

where t_1^- and t_2^- are for label=-1, and t_1^+ and t_2^+ are for label=$+1$.

Proof. (Of equation (3.72))

$$p_-\xi + H(p_-) = 0 \iff p_-\xi = -H(p_-) \iff \xi = \frac{-H(p_-)}{p_-}.$$ □

Proof. (Of equation (3.73))

$$\begin{aligned} p_-\xi + H(p_-) = p_+\xi + H(p_+) &\iff (p_- - p_+)\xi = H(p_+) - H(p_-) \\ &\iff \xi = \frac{H(p_+) - H(p_-)}{p_- - p_+}. \end{aligned}$$ □

Proof. (Of equation (3.74))

$$\begin{aligned} &(p_+ - 1)\xi + H(p_+) = (p_- - 1)\xi + H(p_-) \\ \iff &((p_+ - 1) - (p_- - 1))\xi = H(p_-) - H(p_+) \\ \iff &\xi = \frac{H(p_-) - H(p_+)}{p_+ - p_-}. \end{aligned}$$ □

Proof. (Of equation (3.75))

$$(p_- - 1)\xi + H(p_-) = 0 \iff \xi = \frac{-H(p_-)}{p_- - 1}.$$ □

Decomposition through support functions For label=-1.

$$\ell(\xi, -1) = \sigma(p_-(\xi - t_1^-)) + \sigma((p_+ - p_-)(\xi - t_2^-))$$

For label =$+1$

$$\ell(\xi, +1) = \sigma((p_- - p_+)(\xi - t_1^+)) + \sigma((p_- - 1)(\xi - t_2^+))$$

Smoothed loss Now just take a smooth γ-approximation σ^γ of σ.

Chapter 4

Conditional gradient algorithms for doubly non-smooth learning

4.1 Introduction

The conditional gradient algorithm, *a.k.a.* Frank-Wolfe, perform smooth optimization over a compact convex set and only requires i) a first-order oracle and ii) a linear minimization oracle over that compact convex set. This historical algorithm and its recent extensions to different optimization formulations Harchaoui et al. (2014); Hazan and Kale (2012); Jaggi and Sulovský (2010); Lacoste-Julien et al. (2013); Zhang et al. (2012) are increasingly popular due to their relevance for large-scale applications. Applications include collaborative filtering on the Netflix dataset or image categorization on the ImageNet dataset Harchaoui et al. (2012a); Jaggi and Sulovský (2010); Shalev-Shwartz et al. (2011). Related works also include greedy or forward selection algorithms Shalev-Shwartz et al. (2011), which can be considered as cousins to conditional gradient algorithms.

Indeed, conditional gradient algorithms stands in contrast to proximal algorithms for first-order optimization. For composite smooth optimization, proximal algorithms Bach et al. (2012b) require a first-order oracle that returns objective and gradient evaluations (for the smooth part), and a proximal operator oracle associated with the nonsmooth part of the objective. Such algorithms are particularly attractive when the proximal operator is cheap to compute, as *e.g.* for the vector ℓ_1-norm, and they enjoy an $O(1/t^2)$ convergence rate for their accelerated versions Juditsky and Nemirovski (2010). However, they could turn out to be prohibitive when the proximal operator is expensive if not impossible to compute, *e.g.* for the nuclear-norm of matrices when these matrices are high-dimensional, as it arises in the large-scale

applications mentioned above. On the other hand, in place of the proximal operator oracle, conditional gradient algorithms (CGAs) require instead a linear minimization oracle (LMO), which is much cheaper to compute, *e.g.* for the nuclear-norm of matrices (maximal pair of singular vectors, in place of full SVD for the proximal operator).

Composite conditional gradient algorithms, that is first-order optimization algorithms for composite objectives that decompose into a smooth part and a nonsmooth part for which a LMO is available, have been proposed Dudik et al. (2012); Harchaoui et al. (2014); Zhang et al. (2012). Composite objectives correspond to learning problems with smooth loss functions and nonsmooth *regularization penalty*. Convergence rates with rate $O(1/t)$ were recently proven for such algorithms Harchaoui et al. (2014). However, in a machine learning context, these algorithms assume smooth loss functions, whereas for several applications nonsmooth loss functions would be preferable Amit et al. (2007); Weimer et al. (2007). Smoothing strategies were recently proposed for nonsmooth counterparts of the "historical" conditional gradient algorithm, that is for nonsmooth objectives (instead of smooth in the original Jaggi (2013)) with a compact convex constraint Garber and Hazan (2013); Lan (2013).

We propose here a smoothed version of the composite conditional gradient algorithm, using the smoothing technique from Nesterov (2005). We give a detailed study of smoothing of nonsmooth loss functions in a machine learning context, and give theoretical grounding for several popular smoothed counterpart of nonsmooth loss functions. We prove a theoretical guarantee on the accuracy of the solution given by our algorithm and present promising experimental results on collaborative filtering.

4.2 Smooth optimization with atomic-decomposition regularization

In this section, we recall the main properties of atomic-decomposition norms, and then describe composite conditional gradient algorithms Dudik et al. (2012); Harchaoui et al. (2014); Zhang et al. (2012), which are tailored for learning problems with these norms, as regularizers.

4.2.1 Learning with atomic-decomposition norms

Consider a sequence of *i.i.d.* examples $u_1, \ldots, u_N$, and a loss function $\ell(W, u)$. Denote the corresponding empirical risk

$$R(W) := \frac{1}{N} \sum_{i=1}^{N} \ell(W, u_i) \ .$$

In this chapter, we consider regularized learning problems that write as

$$\min_{W} \; g(W) := \lambda \|W\|_{\mathcal{A}} + R(W) \tag{4.1}$$

where $\|\cdot\|_{\mathcal{A}}$ is a so-called atomic-decomposition norm Chandrasekaran et al. (2012); Dudik et al. (2012). Atomic-decomposition norms (or atomic norm, in short) can be defined by the following simple variational description with respect to a compact set $\mathcal{A}$ (the "atoms"). Assume that the elements of $\mathcal{A}$ are the extreme points of conv($\mathcal{A}$) (the convex hull of $\mathcal{A}$), we have

$$\|W\|_{\mathcal{A}} = \inf \left\{ \sum_{i \in I} \theta_i \; : \; \theta_i > 0, \, W = \sum_{i \in I} \theta_i a_i \right\} \tag{4.2}$$

where I is an index set spanning the elements of $\mathcal{A}$, and where $(a_i)_{i \in I} \in \mathcal{A}$. Such characterization leverages the property that norms belong to the larger family of "gauges", that are convex and positively homogeneous functions, centered in the origin. The support function of the collection of atoms $\mathcal{A}$ writes as

$$\|W\|_{*\mathcal{A}} = \sup_{a \in \mathcal{A}} \langle W, a \rangle. \tag{4.3}$$

We can recognize that $\|\cdot\|_{*\mathcal{A}}$ is the dual (or polar) norm associated with $\|\cdot\|_{\mathcal{A}}$.

Many useful atomic norms enjoy collections of atoms $\mathcal{A}$ that are simple to describe, and whose support functions are *computationally easy to compute*. Examples include the ℓ_1-norm in $\mathbb{R}^d$, where $\mathcal{A}$ is the canonical basis of $\mathbb{R}^d$, and the trace-norm (or nuclear-norm) in the space of rectangular matrices $\mathbb{R}^{d \times m}$, where $\mathcal{A} = \{uv^\top, \, \|u\|_2 = \|v\|_2 = 1\}$. We refer to Jaggi (2013) for a review of popular atomic norms.

Conditional gradient algorithms, which we shall describe in the next paragraph, take advantage of this attractive feature: they make progress using an (approximated) optimal solution of (4.3).

4.2.2 Conditional gradient for smooth risk

Assume that the empirical risk $R(\cdot)$ is a convex function with Lipschitz continuous gradient with Lipschitz constant L. Under suitable assumptions Harchaoui et al. (2014), the composite conditional gradient algorithm with infinite memory enjoys the following theoretical guarantee

$$g(W_t) - \min g \leq O\left(\frac{1}{t}\right).$$

The composite conditional gradient algorithm works by making calls to a *first-order oracle*, that returns $R(W)$ and $\nabla R(W)$ for any W, and to a *linear minimization oracle*, that is a subroutine that returns for any W

$$\mathbf{LMO}(W) := \operatorname*{argmin}_{Z \in \mathcal{A}} \langle Z - W, \nabla R(W) \rangle \ . \tag{4.4}$$

This is in contrast to proximal algorithms, which make progress by making calls to a *proximal operator oracle*. Proximal operators are computationally expensive to compute in several large-scale learning problems. Typical examples are matrix completion with noise, or multi-class classification with nuclear-norm penalty, where the proximal operator associated with the nuclear-norm corresponds to a full singular value decomposition of the current iterate, which is prohibitive in large-scale applications. Moreover, recent results from Guzman and Nemirovski (2013) show that conditional gradient algorithm is almost optimal (up to a log factor) for large-scale optimization problems.

The composite conditional gradient algorithm with conic-hull acceleration, is summarized below (see Algo. 1).

Algorithm 11 Composite Conditional Gradient, with conic-hull acceleration

Inputs: λ, ϵ
Initialize $W_0 = \mathbf{0}$, $t = 1$
for $k = 0 \ldots K$ **do**
  Call the linear minimization oracle: $a_i \leftarrow \mathbf{LMO}(W_k)$
  Compute

$$\min_{\theta_1, \ldots, \theta_t \geq 0} \quad \lambda \sum_{i=1}^{t} \theta_i + R\left(\sum_{i=1}^{t} \theta_i a_i \right)$$

  Increment $t \leftarrow t + 1$
end for
Return $W = \sum_i \theta_i a_i$

Algorithm 12 Conditional gradient algorithm: Frank-Wolfe

Input
Initialize $W_0 = \mathbf{0}$, $t = 1$
for $k = 0 \ldots K$ **do**
  Call linear minimization oracle $a_k \leftarrow \mathbf{LMO}(W_t)$
  Set step-size $\alpha_k = 2/(2 + k)$
  Update $W_{k+1} \leftarrow (1 - \alpha_k) W_k + \alpha_k a_k$
end for
Return W_K

4.2.3 Extension to non-smooth empirical risk

Composite conditional gradient assumes that the empirical risk in the objective function g is smooth. Indeed, at each iteration, the algorithm requires to compute $\nabla R(W)$. Should we consider nonsmooth loss functions, such as the ℓ_1-loss or the hinge-loss, the convergence of the algorithm is unclear if we replace the gradient by *a subgradient* in $\partial R(W)$. In fact, we can produce a simple counterexample showing that the corresponding algorithm can get stuck in a suboptimal point.

Let us describe a counterexample in two dimensions (generalization to higher dimension is straightforward). We consider the ℓ_1-norm with its four atoms $\{(1,0),(0,1),(-1,0),(0,-1)\}$ and a convex function of the type of a translated weighted ℓ_1-norm

$$f(w_1, w_2) = |w_1 + w_2 - 3/2| + 4\,|w_2 - w_1|\,.$$

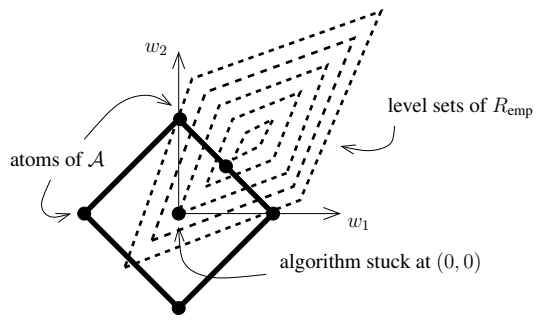

Figure 4.1: Drawing of a situation where the algorithm using a subgradient of a nonsmooth empirical risk does not converge.

We observe that the four directions given by the atoms go from $(0,0)$ towards level-sets of R with larger values. This yields that, for small λ, the minimization of the objective function on these directions returns $(0,0)$. Thus, if we simply replace $\nabla R(W)$ by any subgradient in $\partial R(W)$ in Algorithm 1, then the algorithm would get stuck to the initial iterate $(0,0)$, while the optimal solution is $(1/2, 1/2)$.

We are interested in the smoothing of the only empirical risk because smoothing the regularizer leads to loose its fundamental properties.

4.3 Motivating examples

We present here motivating examples for designing a composite conditional gradient algorithm for matrix learning problems with nonsmooth loss functions. In all applications, the nonsmooth regularization is the nuclear-norm, the sum of singular values of the matrix, which has an atomic-decomposition form $\|\cdot\|_{\mathcal{A}}$ with

$$\mathcal{A} = \left\{ uv^\top \in \mathbb{R}^{d\times m} \,\middle|\, \|u\|_2 = \|v\|_2 = 1 \right\}.$$

4.3.1 Collaborative filtering

Collaborative filtering, or matrix completion, consists in the generation of a low-rank matrix from few known approximate entries. The loss $\ell(w, x) = |w - x|$, based on ℓ_1 norm Huber (1981) ensures robustness to outliers. We have (4.1)

$$\min_{W\in\mathbb{R}^{d\times m}} \quad \frac{1}{N} \sum_{(i,j)\in\Omega} |W_{ij} - X_{ij}| + \lambda \|W\|_{\mathcal{A}} \tag{4.5}$$

where Ω is the subset of $\{1, \ldots, d\} \times \{1, \ldots, m\}$ denoting pairs of ratings. The size of Ω is N and $\{x_{ij}\}_{(i,j)\in\Omega}$ are the known entries.

4.3.2 Multiclass learning

Let $p_1 = (x_1, y_1), \ldots, p_N = (x_N, y_N)$ be labeled training data examples, where $x_i \in \mathbb{R}^d$ are feature vectors, m is the number of classes and $y_i \in \mathbb{R}^m$ are the associated class labels. A (linear) classifier is specified by a separate weight vector $W_y \in \mathbb{R}^d$: for a given test example $x \in \mathbb{R}^d$, the predicted class is $\hat{y} = \operatorname{argmax}_{r=1\ldots m} W_r^\top x$. The weight vectors are kept in columns in a matrix $W = [W_1, \ldots, W_m] \in \mathbb{R}^{d\times m}$.

Here we are interested in predicting k classes instead of only one. This could seem surprising, but the main assumptions are that: 1) Any image contains more than only one object: even if there is a central big one, several other small objects appear; 2) Speaking of a dataset, the ground truth, i.e. the true label of each image, is subjective and depends on the personal choice of the human. The function we learn associates to each image a list of k labels ordered by relevancy, instead of having only the best one as in top-1 classification.

One motivation to use the prediction function (1.49) is when two objects A and B are very similar, for example two types of trees. It is acceptable that (1.49) predicts both of them among the first k results and does not take care too much of the subjective choice of the ground truth. On the other hand, the top-1 can

predict A when the true is B, and we would get the same error as predicting any other object. An example of subjective labeling is an image with a bee on a flower. It could have the label `bee`, but the classifier gives the top score to the class `flower` and the second top score to `bee`. While the top-1 accuracy would accept only `flower` and consider this a misclassification, a top-k accuracy (here $k \geq 2$) would consider it as a good classification.

Another motivation is to use top-k inside a flow of actions. For example we could find the best k "candidate" classes for the given image, then we apply object detection to identify bounding boxes of different objects and hereafter apply filters to find the most relevant class. For large scale number of classes m it is considered good to have the true class predicted among the first k of highest score. We assume that k is much smaller than m, for example $k = 5$ or $k = 10$. Top-k accuracy is commonly used to evaluate the performance of a classifier. The top-k error is defined as

$$W \mapsto \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 0 & \text{if } y_i \in \text{top } k \text{ among} \{W_r^\top x_i\}_{r=1\ldots m} \\ 1 & \text{if } y_i \notin \text{top } k \text{ among} \{W_r^\top x_i\}_{r=1\ldots m} \end{cases} \tag{4.6}$$

and corresponds to 1 minus top-k accuracy. This gives an error of 0 when the good class y is among the first k predicted and 1 otherwise.

We want to solve the nonsmooth classification problem

$$\min_{W \in \mathbb{R}^{d \times m}} \frac{1}{N} \sum_{i=1}^{N} \ell(W, (x_i, y_i)) + \lambda \|W\|_{\mathcal{A}}, \tag{4.7}$$

where the loss ℓ is a convex nonsmooth upper bound of the top-k error. We will define it later, now we just observe that for $k = 1$ the loss ℓ is the hinge loss.

4.4 Smoothed Conditional Gradient algorithms

We present Smoothed Conditional Gradient algorithms to minimize large-scale doubly nonsmooth optimization problems. We consider a family of smooth convex functions $f(\cdot, \gamma)$ parameterized by the smoothing parameter γ. More precisely, $f(\cdot, \gamma)$ is differentiable with Lipschitz continuous gradient, with Lipschitz constant $L_\gamma = 1/(c\gamma)$ for $c > 0$. We assume that $f(\cdot, \gamma)$ approximates f up a global constant linear in γ:

$$f(W, \gamma) \leq f(W) \leq f(W, \gamma) + \gamma M. \tag{4.8}$$

As we explained in the previous section, a direct application of conditional gradient methods for

nonsmooth loss functions would not converge to an optimal solution. Inspired by smoothing approaches in other contexts (see in particular Nesterov (2005)), we propose in this chapter to apply, in an original way, conditional gradient algorithms to an adaptive smoothing approximation of f.

We could apply the conditional gradient method to minimize the smooth $f(W, \gamma)$: the algorithm would generate a sequence (W_t) such that

$$f(W_t, \gamma) - f(W^\star, \gamma) \leq O(1/\sqrt{t})$$

where $f(W^\star, \gamma) = \min_{W \in X} f(W, \gamma)$. Even if $f(W^\star, \gamma)$ tends to $f(W^\star)$ as γ vanishes (by (4.8)), we do not have any guarantee on the speed of convergence, as the Lipschitz constant tends to infinity. So our proposed algorithm depends on a sequence of positive smoothing parameters $(\gamma_t)_t$, that we will specialize to get guarantees. These algorithms are generic with respect to the family $f(\cdot, \gamma)$ the choice of γ_t, and the way we compute W_{t+1}.

4.4.1 Smoothed Conditional Gradient algorithm

Algorithm 13 (Generic) Smoothed Conditional Gradient Algorithm

Initialize $W = \mathbf{0}$, $t = 1$
for $t = 1, \ldots$ **do**
  Set $\alpha_t = 2/(t+1)$ and compute $f(W_t, \gamma_t)$
  Call the linear minimization oracle:
    $W_t^+ = \operatorname{argmin}_{W \in X} \langle \nabla f(W_t, \gamma_t), W \rangle$
  Compute W_{t+1} such that $f(W_{t+1}, \gamma_t) \leq f(\widetilde{W}_{t+1}, \gamma_t)$ where $\widetilde{W}_{t+1} = W_t + \alpha_t(W_t^+ - W_t)$
end for

At each iteration t, the algorithm produces the linearization of the convex $f(\cdot, \gamma_t)$ given by the gradient $W \mapsto f(W_t, \gamma_t) + \langle \nabla f(W_t, \gamma_t), W - W_t \rangle$. This linearization gives in turn a lower bound on $f(W^\star, \gamma_t)$

$$f_t^{\text{low}} = \min_{W \in X} \; f(W_t, \gamma_t) + \langle \nabla f(W_t, \gamma_t), W - W_t \rangle \tag{4.9}$$

$$= f(W_t, \gamma_t) + \left\langle \nabla f(W_t, \gamma_t), W_t^+ - W_t \right\rangle. \tag{4.10}$$

Since γ_t is non-increasing, the previous f_k^{low} for $k \leq t$ are also lower bounds for $f(W^\star, \gamma_t)$. At iteration t, we record the best of the lower bounds found so far

$$f_t^{\text{rec}} = \max_{1 \leq k \leq t} f_k^{\text{low}} \leq \min_{W \in X} f(W, \gamma_t) \leq \min_{W \in X} f(W). \tag{4.11}$$

Notice that we consider here the maximum of the minimum of the linearizations whereas cutting-plane based algorithms as bundle methods uses the minimum of the maximum of the linearizations. The reason is that the lower bound (4.11) is a direct output of the algorithm while the other may be expensive to compute. The lower bound allows us to defined the current observed criteria

$$\Delta_t^{\text{obs}} = f(W_t, \gamma_t) - f_t^{\text{rec}}.$$

Convergence of Δ_t^{obs} controls the quality of the current iterate W_t, as formalize in the next lemma.

Lemma 4.4.1 (Gap by observed criteria)**.** *At iteration t, the smoothed conditional gradient algorithm returns W_t and Δ_t^{obs} such that*

$$f(W_t) - \min_{W \in X} f(W) \leq \gamma_t M + \Delta_t^{obs}. \tag{4.12}$$

We can choose a vanishing smoothing parameter γ_t, but this lemma says that we should moreover pay attention to the resulting behavior of Δ_t^{obs}. The following lemma is the key technical lemma in our derivations.

Lemma 4.4.2 (Decrease of Δ_t^{obs})**.** *For the smoothed conditional gradient algorithm, we have the following bound on Δ_t^{obs}, for all $t \geq 2$,*

$$\Delta_t^{obs} \leq \frac{1}{(t-1)t} \sum_{k=1}^{t-1} k(k+1)\Big((\gamma_k - \gamma_{k+1})M + \frac{2D^2}{c\gamma_k(k+1)^2}\Big) \tag{4.13}$$

where D is the diameter of X defined by

$$D = \max_{W, W' \in X} \|W - W'\|_2 .$$

For example, if $X = \Big\{W \in \mathbb{R}^{d \times m} \,\Big|\, \|W\|_{\sigma,1} \leq r\Big\}$, then $D = 2r$.

Proof. Let us find a recurrence between $\Delta_{t+1}^{\text{obs}}$ and Δ_t^{obs}. Since the lower-bound f_t^{rec} is non-decreasing by construction, we have

$$\begin{aligned} \Delta_{t+1}^{\text{obs}} &= f(W_{t+1}, \gamma_{t+1}) - f_{t+1}^{\text{ref}} \leq f(W_{t+1}, \gamma_{t+1}) - f_t^{\text{rec}} \\ &\leq f(W_{t+1}, \gamma_t) + (\gamma_t - \gamma_{t+1})M - f_t^{\text{rec}}. \end{aligned} \tag{4.14}$$

The usual inequality for differentiable function with Lipschitz gradient for $f(\cdot, \gamma_t)$ (with Lipschitz con-

stant $L_{\gamma_t} = 1/(\gamma_t c)$) gives

$$f(W_{t+1}, \gamma_t) \leq f(W_t, \gamma_t) + \alpha_t \langle \nabla f(W_t, \gamma_t), W_t^+ - W_t \rangle + \alpha_t^2 \left\| W_t^+ - W_t \right\|^2 / (2\gamma_t c). \tag{4.15}$$

We notice that $f_t^{\text{rec}} \geq f_t^{\text{low}}$ gives

$$\langle \nabla f(W_t, \gamma_t), W_t^+ - W_t \rangle \leq f_t^{\text{rec}} - f(W_t, \gamma_t)$$

Thus, (4.15) can be rewritten

$$f(W_{t+1}, \gamma_t) \leq f(W_t, \gamma_t) + \alpha_t (f_t^{\text{rec}} - f(W_t, \gamma_t)) + \alpha_t^2 D^2 / (2\gamma_t c). \tag{4.16}$$

Plugging this in (4.14) gives

$$\Delta_{t+1}^{\text{obs}} \leq \Delta_t^{\text{obs}} (1 - \alpha_t) + (\gamma_t - \gamma_{t+1}) M + \alpha_t^2 D^2 / (2\gamma_t c).$$

By induction, we end up with

$$\Delta_{t+1}^{\text{obs}} \leq \Delta_1^{\text{obs}} \prod_{k=1}^{t} (1 - \alpha_k) \tag{4.17}$$

$$+ \sum_{k=1}^{t} \left((\gamma_k - \gamma_{k+1}) M + \frac{\alpha_k^2 D^2}{2\gamma_k c} \right) \prod_{i=k+1}^{t} (1 - \alpha_i) \tag{4.18}$$

We use now the choice of stepsize $\alpha_k = 2/(k+1)$ which zeroes the first term and allows us to explicit the second as follows

$$\prod_{i=k+1}^{t} (1 - \alpha_i) = k(k+1)/(t(t+1)).$$

This yields (4.13). □

We can study the consequence of the above bound on the decrease of Δ_t^{obs} in two cases: (1) for γ_t fixed and (2) for γ_t of order of $1/\sqrt{t+1}$. This results in the following theorems on the convergence of the algorithm.

Theorem 4.4.3 (Case 1: fixed smoothing parameter)**.** *For the smooth generic conditional gradient algorithm with a constant smoothing parameter $\gamma_t = \gamma_0$, we have for all $t \geq 2$*

$$f(W_t) - f(W^\star) \leq \gamma_0 M + \frac{2}{\gamma_0} \frac{1}{t+1}. \tag{4.19}$$

The above proposition can be interpreted as follows. Given a target accuracy ϵ, the optimal amount of smoothing $\gamma(\epsilon)$ can be computed so that after some number of iterations $T(\epsilon)$ an ϵ-optimal minimum of the objective function of interest is reached.

Up to our knowledge, the parameter γ is often considered fixed as above in theory. In practice, it is either chosen empirically from first numerical experiments, or reduced gradually for better computational performances by continuation techniques Becker et al. (2011).

One might also consider a sequence of decreasing γ_t to get theoretical guarantees which do not depend on a priori choice for the total number of iterations, as in the previous theorem.

Theorem 4.4.4 (Case 2: smoothing parameter in $1/\sqrt{t+1}$)**.** *For the smooth generic conditional gradient algorithm where*

$$\gamma_t = \frac{\gamma_0}{\sqrt{t+1}} \qquad \textit{with } \gamma_0 = D/\sqrt{cM},$$

we have for all $t \geq 2$ *and* $C = 2\sqrt{M/c}(1+\sqrt{2}D)/3$

$$f(W_t) - f(W^\star) \leq \frac{C}{\sqrt{t-1}} \tag{4.20}$$

4.4.2 Smoothed Composite Conditional Gradient Algorithm

We now turn to a smoothed version of the composite conditional gradient algorithm (CCG). We start by stating a simpler version of the composite conditional gradient algorithm for solving

$$\min_W \; f(W) + \lambda \|W\|$$

where f is convex, Lipschitz-continuous with constant L and λ is non-negative.

The generic conditional gradient algorithm naturally generates a sequence of lower bounds f_t^{rec} along the iterations which act as a certificate on the theoretical convergence of the algorithm. In order to equip the composite conditional gradient algorithm with a similar property, we make the following assumption.

We assume that we have an upper-bound D of the norm at the minimum

$$\|W^\star\| \leq D \; .$$

and therefore restrict X to this ball. We are then solving the equivalent optimization problem

$$\min_{W \in X} \; f(W) + \lambda \|W\| \; . \tag{4.21}$$

We use the construction outlined in Sec. 5 of Harchaoui et al. (2014). Introducing the change of variable $V := [W, r]$ and the set $Z := X \times [0, D]$, and rewriting Eq. 4.21 in epigraph form (Boyd and Vandenberghe, 2004), we now have

$$\min_{V \in Z} F(V) := \{f(W) + \lambda r\} \ . \tag{4.22}$$

The composite conditional gradient algorithm generates a sequence of iterates (V_t), where $V_t := [W_t, r_t]$, which satisfy

$$F(V_t) - \min_{V \in Z} F(V) \leq \frac{8LD^2}{t+14} \ , \quad \text{for all } t = 2, 3, \ldots$$

We summarize this generic composite conditional algorithm below.

Algorithm 14 Generic Composite Conditional gradient algorithm

Input
Initialize $V_0 = [\mathbf{0}, D]$, $t = 1$
for $t = 0 \ldots T$ **do**
  Call linear minimization oracle $V_t^+ = [W_t^+, D]$ where $W_t^+ = \operatorname{argmin}_{W \in X} \langle \nabla f(W_t, \gamma_t), W \rangle$
  $V_{t+1} = \underset{\beta \geq 0,\, \delta \geq 0,\, \beta + \delta \leq 1}{\operatorname{argmin}} F(\beta V_t + \delta V_t^+)$
end for
Get $[W_t, r_t] = V_t$ and return W_t.

At each iteration t, the algorithm produces a *partial linearization* of the convex function $F(\cdot, \gamma_t)$, that is performs a linearization of $f(\cdot, \gamma_t)$. The partial linearization gives the lower bound on $F(V^\star, \gamma_t)$

$$F_t^{\text{low}} = \min_{V \in Z} F(V_t, \gamma_t) + \langle [\nabla f(W_t, \gamma_t), \lambda], V - V_t \rangle \tag{4.23}$$

$$= F(V_t, \gamma_t) + \left\langle [\nabla f(W_t, \gamma_t), \lambda], V_t^+ - V_t \right\rangle \tag{4.24}$$

where $V_t^+ = [W_t^+, D]$ and $W_t^+ = \operatorname{argmin}_{W \in X} \langle \nabla f(W_t, \gamma_t), W \rangle$. Note that, owing to the a priori upper-bound D, we have $\|W\| \leq D$ and $0 \leq r \leq D$, hence the lower-bound is well-defined.

At iteration t, we record the best of the lower bounds found so far

$$F_t^{\text{rec}} = \max_{1 \leq k \leq t} F_k^{\text{low}} \leq \min_{V \in Z} F(V, \gamma_t) \leq \min_{V \in Z} F(V) \ , \tag{4.25}$$

yielding the observable criterion

$$\Delta_t^{\text{obs}} = F(V_t, \gamma_t) - F_t^{\text{rec}} \ .$$

Lemma 4.4.5 (Gap by observed criteria)**.** *At iteration t, the smoothed composite conditional gradient*

algorithm returns V_t and Δ_t^{obs} such that

$$F(V_t) - \min_{V \in Z} F(V) \leq \gamma_t M + \Delta_t^{obs}. \tag{4.26}$$

The smoothed composite conditional gradient algorithm satisfies the following theoretical guarantee when the smoothing sequence is fixed and constant.

Theorem 4.4.6 (Case 1: fixed smoothing parameter)**.** *For the smooth generic composite conditional gradient algorithm with a constant smoothing parameter $\gamma_t = \gamma_0$, we have for all $t \geq 2$*

$$f(W_t) - f(W^\star) \leq \gamma_0 M + \frac{2}{\gamma_0} \frac{1}{t+14}. \tag{4.27}$$

We may also consider a sequence of varying γ_t which do not depend on an apriori choice for the total number of iterations. We have

$$\Delta_{t+1}^{\text{obs}} \quad =\leq \quad F(W_{t+1}, \gamma_t) + (\gamma_t - \gamma_{t+1})M - F_t^{\text{rec}} \ . \tag{4.28}$$

Invoking the Lipschitz-continuity of $f(\cdot, \gamma_t)$, whose Lipschitz constant is $L_{\gamma_t} = 1/(\gamma_t c)$, we get the partial linearization upper-bound on $F(\cdot, \gamma_t)$ for all $0 \leq \alpha \leq 1$

$$F(V_{t+1}) \leq F((1-\alpha)V_t + \alpha V_t^+, \gamma_t) \leq F(V_t, \gamma_t) + \alpha \big\langle [\nabla f(W_t, \gamma_t), \lambda], V_t^+ - V_t \big\rangle + \alpha^2 \frac{\left\| W_t^+ - W_t \right\|^2}{2\gamma_t c} \ .$$

Denote

$$\alpha_t := \operatorname*{argmin}_{0 \leq \alpha \leq 1} \ F(V_t, \gamma_t) + \alpha \big\langle [\nabla f(W_t, \gamma_t), \lambda], V_t^+ - V_t \big\rangle + \alpha^2 \frac{\left\| W_t^+ - W_t \right\|^2}{2\gamma_t c} \ . \tag{4.29}$$

We can see that the appropriate amount of smoothing γ_t is proportional to $\sqrt{\alpha_t}$.

When $\alpha_t = 2/(t+1)$ for all $t \geq 1$, then

$$\Delta_t^{\text{obs}} \leq \frac{1}{(t-1)t} \sum_{k=3}^{t-1} k(k+1) \Big((\gamma_k - \gamma_{k+1})M + \frac{2D^2}{c\gamma_k (k+1)^2} \Big) \tag{4.30}$$

Theorem 4.4.7 (Case 2: smoothing parameter in $1/(t{+}1)$)**.** *For the smooth generic composite conditional gradient algorithm, when $\alpha_t = 2/(t+1)$ and*

$$\gamma_t = \frac{\gamma_0}{\sqrt{t+1}} \qquad \textit{with } \gamma_0 = D/\sqrt{cM},$$

we have for all $t \geq 3$ *and* $C = 4\sqrt{M/c}D$

$$f(W_t) - f(W^\star) \leq \frac{C}{\sqrt{t+16}} \tag{4.31}$$

In practice, we shall also explore variants of the smoothed composite conditional gradient, where the smoothing parameter is varying according to $\gamma_t := \gamma_0/(t+1)^p$ with $p \in \{0; 0.5; 1; 1.5; 2\}$; see Sec. 4.5.

4.4.3 Smoothing the empirical risk - Application to the motivating examples

We showed in Chapter 3 that the Nesterov smoothing technique Nesterov (2005) consists in performing an infimal convolution on a saddle-point representation. We illustrate here on several examples of interest how this smoothing technique can be successfully applied.

In all the examples we consider, the empirical risk $R(W)$ is an empirical average over all the examples of some *nonsmooth* loss function:

$$R(W) = \frac{1}{N}\sum_{i=1}^{N} \ell(W, p_i)$$

where we wrote the loss function in a compact abstract form that can be instantiated in our motivating examples.

In our notation the loss for each example p is the composition of a support function

$$\sigma(\xi) := \max_{z \in \mathcal{Z}} \langle \xi, z \rangle$$

and an affine map $W \mapsto A_p W + b_p$: $\ell(W, p) := \sigma(A_p W + b_p)$. The support function we chose is related to the type of error we intend minimize and the map function is related to the type of data.

Thanks to the smoothing technique, we can now design a *smoothed* version of the empirical risk: $R(W, \gamma)$, parametrized by a smoothing parameter γ that controls the amount of smoothing

$$R(W, \gamma) = \frac{1}{N}\sum_{i=1}^{N} \ell^\gamma(W, p_i)$$

$R(\cdot, \gamma)$ is differentiable with Lipschitz continuous gradient.

Then, in our notation, we have surrogates $\ell^\gamma(W, p) := \sigma(A_p W + b_p, \gamma)$ and $\sigma(\xi, \gamma) := \max_{z \in \mathcal{Z}} \langle \xi, z \rangle - \gamma\omega(z)$. We will see later the properties for the function ω and how to find smooth surrogates of the nonsmooth empirical losses for the motivating examples.

4.4.4 Collaborative filtering

For collaborative filtering with noise, we approximate the absolute value in the empirical risk of problem (4.5), with two different smoothing functions ω.

The size of Ω is N. The empirical risk of this problem is now smooth and we have a explicit expression of its gradient by (4.32). For any $(i, j) \in \Omega$

$$(\nabla R^{\gamma}(W))_{ij} = \nabla_{W_{ij}} \ell^{\gamma}(W_{ij}, X_{ij}). \tag{4.32}$$

We observe that we need only the gradient corresponding to the observations when we run the algorithm.

4.4.5 Multiclass learning

The aim of this section is to present the smoothing of the top-k misclassification error. We start with an upper bound called Ordered Weighted Averaging, and then we show how we find smooth surrogates of it.

We are interested in a particular nonsmooth convex function

$$\text{owa}(\xi) := \frac{1}{k} \sum_{j=1}^{k} \xi_{\beta(j)}, \tag{4.33}$$

where β is the permutation that sorts ξ in decreasing order, that is one of the Ordered Weighted Averaging loss functions, introduced in Yager (1988) and defined in Usunier et al. (2009) for multiclass classification. In applications to classification the values of the vector ξ correspond to the positive part of the linear operator, i.e they minimize $W \mapsto \text{owa}(\max\{0, AW + b\})$. The affine mapping for classification related to the example $p = (x, y)$ is

$$A_{x,y}W + b := \{1 - \delta(r, y) + (W_r - W_y)^{\top} x\}_{r=1}^{m},$$

where $\delta(a, b) := 0, \text{ if } a \neq b, \ 1 \text{ otherwise.}$

Convex upper bound for the top-q misclassification We show that the top-k empirical risk which we will smooth is an upper bound of the top-k misclassification error (4.6). We start showing that the Ordered Weightd Average (4.33) is an upper bound of the top-k misclassification.

Proof. We have $W \in \mathbb{R}^{d \times m}$, y is the true class associated to the feature vector $x \in \mathbb{R}^{d}$, $r \in \{1 \dots m\}$ is

a class index. Let us define a permutation β on the indices such that $W_{\beta(1)}^\top x \geq \cdots \geq W_{\beta(m)}^\top x$. Then

$$\begin{aligned} owa(AW+b)_+ = \sum_{r=1}^{k}(AW+b)_+ &\geq \frac{1}{k}\sum_{r=1}^{k}\max\{0, 1-\Delta(\beta(y),\beta(r)) - W_{\beta(y)}^\top x + W_{\beta(r)}^\top\} \\ &\geq \frac{1}{k}\sum_{r=1,\, r\neq y}^{k}\max\{0, 1 - W_{\beta(y)}^\top x + W_{\beta(r)}^\top\} \\ &\geq \frac{1}{k}\sum_{r=1,\, r\neq y}^{k}\mathbf{1}(W_{\beta(y)}^\top x \leq W_{\beta(r)}^\top x) \\ &\geq \mathbf{1}(\beta(y) \geq k+1) \end{aligned}$$

The latter is 1 when the score of the true class is smaller than the scores of at least other k classes and corresponds to the top-k error for the example (x, y). □

Writing owa as support function In order to apply the smoothing technique, we show that owa can be expressed as support function.

We define three sets: the discrete set

$$\mathcal{Z}_d := \left\{ z \in \mathbb{R}^n \,\middle|\, z_i \in \left\{0, \tfrac{1}{k}\right\}, \sum_{i=1}^{n} z_i = 1 \right\} = \{Pe_k \in \mathbb{R}^n \mid P \in \mathcal{P}\},$$

where $\mathcal{P}$ is the set of permutation matrices, the "flat" ball

$$\mathcal{Z}_f := \left\{ z \in \mathbb{R}^n \,\middle|\, 0 \leq z_i \leq \frac{1}{k}, \sum_{i=1}^{n} z_i = 1 \right\} \tag{4.34}$$

and the "full" ball

$$\mathcal{Z} := \left\{ z \in \mathbb{R}^n \,\middle|\, 0 \leq z_i \leq \tfrac{1}{k}, \sum_{i=1}^{n} z_i \leq 1 \right\}, \tag{4.35}$$

The equivalence for $\mathcal{Z}_d$ is evident because Pe_k is just a reordering of the vector e_k. We also notice that the cardinality of $\mathcal{Z}_d$ corresponds to the combinations of k elements from n: $\#\mathcal{Z}_d = C_n^k = \binom{n}{k}$. We observe that the convex hull of $\mathcal{Z}_d$ is $\mathcal{Z}_f$, because the extremal points of $\mathcal{Z}_f$ are elements of $\mathcal{Z}_d$ Hiriart-Urruty and Lemarechal (1993).

We want to show that owa and $\sigma_{\text{top},k}(\cdot)$ are the same when applied to classification, where we want to optimize the nonsmooth empirical risk $W \mapsto \sigma_{\text{top},k}(AW+b)$.

Proposition 4.4.8. *The function owa of* (4.33) *corresponds to the support function of the set $\mathcal{Z}$ defined*

at (4.35)*, which we call* $\sigma_{\text{top},k}(\cdot)$*:*

$$\text{owa}(\xi_+) = \sigma_{\text{top},k}(\xi) := \max_{z \in \mathcal{Z}} \langle z, \xi \rangle$$

Proof. We start by claim that owa can be expressed as support function of $\mathcal{Z}_f$:

$$\text{owa}(\xi) = \max_{z \in \mathcal{Z}_f} \langle z, \xi \rangle.$$

Suppose ξ sorted in decreasing order. Then $\sup_{z \in \mathcal{Z}_f} \langle z, \xi \rangle = \sup_{z \in \mathcal{Z}_f} \sum_{i=1}^n z_i \xi_i = \sup_{z \in \mathcal{Z}_f} (\sum_{i=1}^k z_i \xi_i + \sum_{i=k+1}^n z_i \xi_i) = \frac{1}{k} \sum_{i=1}^k \xi_i$. The z that maximizes in this case is e_k. For any ξ we deduce $\sup_{z \in \mathcal{Z}_f} \langle z, \xi \rangle = \frac{1}{k} \sum_{i=1}^k \xi_{\sigma(i)} = \text{owa}(\xi)$. Using the previous claim we can conclude $\text{owa}(\xi_+) = \text{owa}(\max\{0, \xi\}) = \max_{z \in \mathcal{Z}_f} \langle z, \max\{0, \xi\} \rangle = \max_{z \in \mathcal{Z}} \langle z, \xi \rangle = \sigma_{\text{top},k}(\xi)$, where the third equality is proven at appendix A.4.1. □

4.5 Experiments

The general form of the *smoothed generic conditional gradient algorithm* (SCCG) is summarized in Algo. 2. The SCCG algorithm works by making calls to a first-order oracle, that returns $R(W, \gamma_t)$ and $\nabla R(W, \gamma_t)$ for any W, for the smoothing parameter γ_t, and to a linear minimization oracle, that is a subroutine that returns for any W.

Algorithm 15 Smoothed Composite Conditional Gradient (general smoothing sequence (γ_t))

Inputs: λ
Initialize $W = \mathbf{0}$, $t = 1$
for $t = 1, 2, \ldots$ **do**
  Call the linear minimization oracle:
  $a_t = \text{argmin}_{a \in \mathcal{A}} \langle a, \nabla R(W_t, \gamma_t) \rangle$
  Compute
$$\min_{\theta_1, \ldots, \theta_t \geq 0} \quad \lambda \sum_{i=1}^t \theta_i + R^{\gamma_t}\left(\sum_{i=1}^t \theta_i a_i\right)$$
end for
Return $W = \sum_i \theta_i a_i$

We now present the experimental results of the proposed composite conditional gradient algorithm for learning problems with nonsmooth loss functions. We consider two problems: i) collaborative filtering with noise, on the MovieLens datasets; ii) multi-class learning, on ImageNet datasets, with nuclear-norm penalty. The experiences are launched on two disjoint sets for train and validation.

We run SCCG with smoothing based on squared euclidean norm proximity function. We remind that

at each iteration the algorithm SCCG optimize the family of surrogates of empirical risk R^{γ_t}. We chose as smoothing parameter

$$\gamma_t := \gamma_0/(t+1)^p \quad \text{with } p \in \{0; 0.5; 1; 1.5; 2\}$$

and with γ_0 ranging between 0.01 and 1000, and with λ ranging between 10^{-2} and 10^{-12}.

For the experiment with SCCG, the possible combinations of parameters to define γ_t is quite large. To represent the results of convergence in plots we chose just some values of γ_0 and p. On the other hand we plot all the experiments at a fixed iteration to compare the objective function for all the used parameters γ_0 and p.

4.5.1 Implementation details

To deal with large scale data we need to keep small the RAM memory used during the computations. Here we show some techniques for our experiments, but the main one is to keep the iterate W_t decomposed as a weighted combination of matrices of rank 1.

Conic hull acceleration using quasi-Newton optimization We implement our algorithm SCCG in Matlab. We use the quasi-Newton solver L-BFGS-B Byrd et al. (1995) (via a Matlab interface) to perform, at each iteration of our algorithm, the minimization over the fixed set of t atoms. Where t can be smaller than the number of iterations due to the elimination of atoms with zero coefficient θ_i. In the particular case of image classification, where the computing time to compute the objective function and its gradient is significant since there are large number of examples, we choose then to sample 10% each time we run the subspace optimization. At each iteration, we find the new descent direction using all the dataset, then we sample to optimize the parameters θ_i. The classes in the sample have the same proportion as in the whole dataset.

Efficient computation of top-k A great improvement for the computation of top-k is the vectorization of operations instead of `for` loops. This take advantage of Matlab efficient matrix multiplication. We have N nondecreasing functions $f_j : \mathbb{R} \to \mathbb{R}$ and N vectors $d_i \in \mathbb{R}^n$ with nondecreasing entries, i.e $\forall j = 1 \dots N$, $d_j^{(1)} \leq d_j^{(2)} \leq \cdots \leq d_j^{(n)}$. For each j we want to find an index i such that $f_j(d_j^{(i)}) \leq 0 \leq f_j(d_j^{(i)})$. The simplest way to implement is with a loop on i, but we want to avoid it. Instead we put all this vectors in columns of the matrix $D \in \mathbb{R}^{n \times N}$. $D_{ij} := d_j^{(i)}$ and define an indexing with the array of indices $I \in \mathbb{N}^N$, which takes from each column of D one entry. $D_I := (D_{I_1,1}, \dots, D_{I_N,N})$, We

define a function $F: \mathbb{R}^N \to \mathbb{R}^N$ such that $F = (f_1, \ldots, f_N)$. Then we look for row indices I such that $F(D_I) \leq \mathbf{0} \leq F(D_{I+\mathbf{1}})$ In practice we do dichotomic search on each column of D, but all the columns are treated at the same time. We start with $I_{\text{min}} = \mathbf{1}$ and $I_{\text{max}} = \mathbf{n}$. The new candidate vector of indices is $I_c \leftarrow \text{floor}(1/2(I + J))$. On columns k where the sign of $F(D_{I_C})$ is positive we assign $I^i_{\text{max}} \leftarrow I^i_c$, otherwise we assign $I^i_{\text{min}} \leftarrow I^i_c$. Until $I_{\text{max}} - I_{\text{min}} = \mathbf{1}$. The length of the loop on rows is $\log_2(n)$. In our dataset we need only $\log_2(4096) = 12$ computation of the function F which is parallelized by Matlab.

Memory storage of iterates during optimization In all applications, we manage the size of the memory and we avoid to store all the iterations W_t generated by the algorithm. Each W_t is represented as a pair of matrices U, V containing the vectors u_t, v_t on columns and a vector θ of coefficients. Nevertheless we want to keep some partial iterations to keep partial models and plot whatever without relaunching everything. We decided to save power of 2 iterations number $0,\ 1,\ 2,\ 4,\ \ldots,\ T$. To save an iteration t the needed RAM is less than $8(dt + mt + 1)$ Bytes. Summing up we need $16(dT + mT + \log_2(T))$ Bytes for T iterations. Besides, to increase the sparsity, we applied hard thresholding to u_j and v_j with constant $\varepsilon = 10^{-6}$.

In the particular case of collaborative filtering, even though each W_t is a dense matrix of dimension $d \times m$, when we optimize we are interested only in its observed entries. So, W_t is never created as matrix object, but we keep only a representation of it with a vector of entries and a vector of indices of length n. So we use only n doubles instead of dm. The only time we need a matrix of size $d \times m$ is to compute the descent direction, but this matrix corresponds to the gradient of the loss and is sparse. So also here the memory needed is $O(n)$. We have a trade-off between computing time and memory. We decided for each past iteration i to keep in memory also the values of observed entries in $u_i v_i^\top$. So for T iterations we need memory of size $O(T)$. The time to compute this is constant. For instance, in the MovieLens dataset, for $T = 256$ iterations we need $10^7 * 256 * 8B \simeq 20.5GB$ to store iterations. The alternative is to recompute all the $u_i v_i^\top$ at each iteration, we need to add a time of size $O(T)$. In this case we estimate the computing time for 256 iterations in 1000 days, while we used just 4 days for a grid with 4 values of λ, 3 of p and 5 of γ_0.

In the particular case of image classification, the use of RAM memory is concentrated in the dataset, which takes 5.9GB, and the iterations decomposed always into U, θ, v. This takes $t * 34KB$ where t is the iteration number. This is quite good because the memory needed increases very slowly with iterations.

4.5.2 Collaborative filtering

First iteration The chosen initial iteration corresponds to a constant matrix that contains the average value of ratings. We observed that this choice is better than to start with an all-zero matrix. With all-zero matrix the initial empirical risk is higher and the first iterations are wasted to find a model W that predicts worse than the constant average.

MovieLens dataset We test our approach on the MovieLens[1] dataset for collaborative filtering, described in Miller et al. (2003). This dataset contains evaluations of movies made by customers, represented by the sparse matrix $X \in \mathbb{R}^{d \times m}$. As every customer evaluated only a small number the movies, X is sparse. Here to complete X means predict how a customer would evaluate a movie which he hasn't seen. Entries of X are normalized dividing by the max entry of X which is 5. We split the dataset into a training, validation and test sets with respectively 60%, 20% and 20% of the entries. In the MovieLens dataset there are

$$\begin{aligned} 71\,567 &\quad \text{users,} \\ 10\,681 &\quad \text{movies,} \\ 10\,000\,054 &\quad \text{ratings,} \end{aligned}$$

then the sparsity of the resulting matrix is 1.3%. All users selected had rated at least 20 movies.

4.5.3 Multi-class classification

We considered multi-class classification with large number of classes and nuclear-norm penalty, as in Harchaoui et al. (2012a). We perform experiments on a subset of ImageNet.

ImageNet dataset

The dataset is a group of all leaf nodes that descend from the parent node "fungus" in the ImageNet hierarchy, described in Deng et al. (2010). We have 134 classes and 100 images per class, i.e. $13\ 400$ image examples. Each example is a vector of 4096 features and has a unique associated class. The model is learned on the train set. The aim is to predict classes of images present in the test set. We split the dataset in train (37.5%), validation (12.5%), test (50%). The image features are Fisher vectors, each one is normalized with ℓ^2 norm. We observe also that "fungus" is considered a 'difficult' dataset because the classes are very similar. It is possible to compare our results with Akata et al. (2014) and Harchaoui et al. (2012a).

We call j the current number of atoms. In the end we delete empty atoms, i.e. when $\theta_i = 0$ we delete it and the corresponding columns of U and V.

[1] `http://grouplens.org`

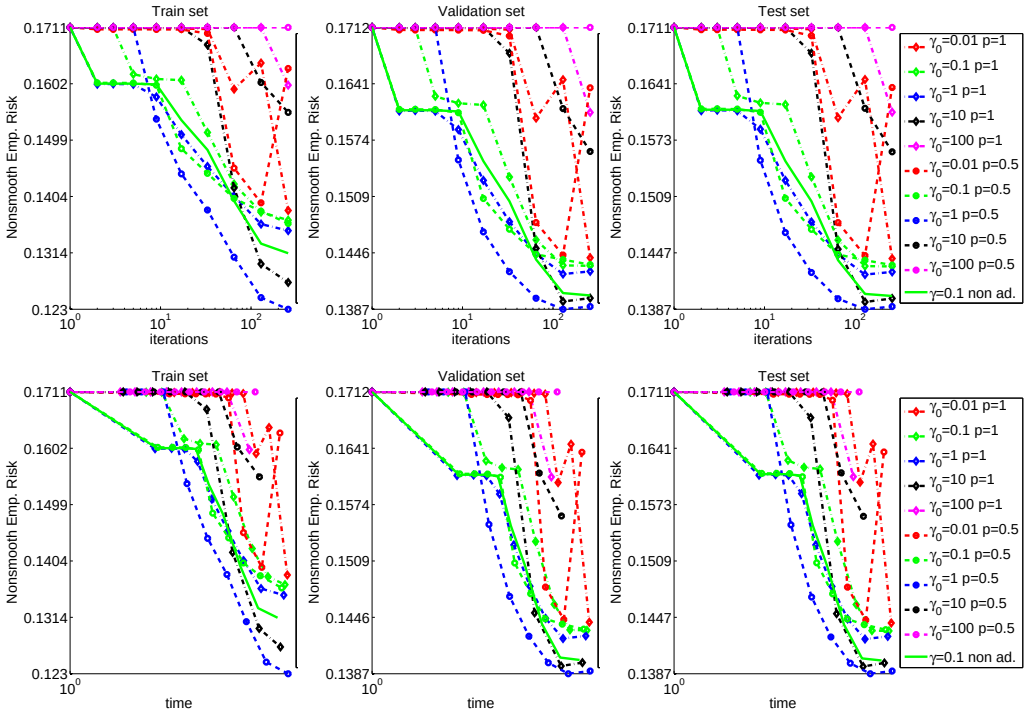

Figure 4.2: Nonsmooth empirical risk, $\lambda = 10^{-6}$. (left) Train set, (center) validation set, (right) test set. (Top) plot versus iteration number, (bottom) plots versus time. We see that some launches with adaptive smoothing perform better than the best non-adaptive algorithm, represented by the continuous line. Collaborative filtering.

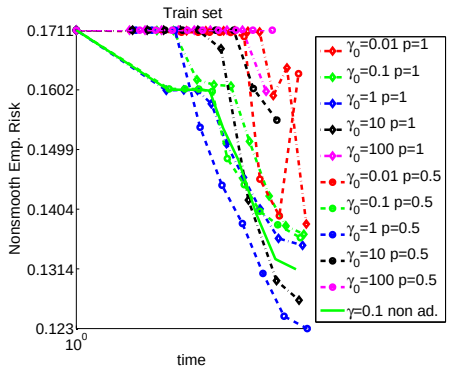

Figure 4.3: Nonsmooth empirical risk versus time, for $\lambda = 10^{-6}$ on train set. Collaborative filtering.

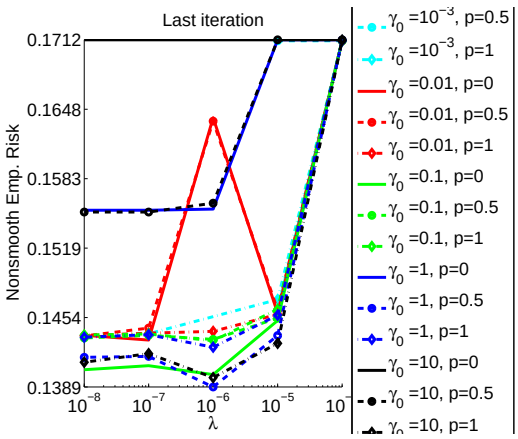

Figure 4.4: **choice of** λ - Nonsmooth empirical risk at last iteration on validation set. Here we chose the best lambda for regularization and the best γ_0 non adaptative, i.e. with $p = 0$. Collaborative filtering.

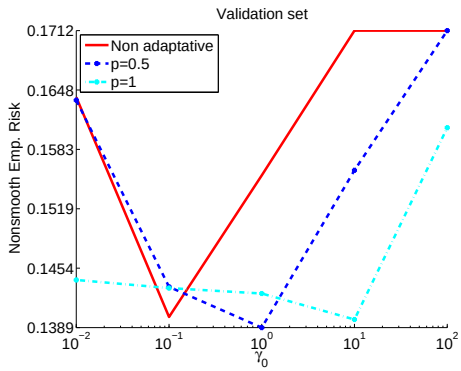

Figure 4.5: Nonsmooth empirical risk at last iteration for $\lambda = 10^{-6}$ on train set. We see on the train set that the best optimization if for $p = 0.5$. Collaborative filtering.

4.5.4 Competing approaches

A direct approach to solve out problem (4.1) would be to use standard nonsmooth optimization algorithms, namely bundle-like methods (see Hiriart-Urruty and Lemarechal (1993)) or subgradient-like methods (see Nesterov (2004), including proximal methods interpreted as implicit subgradient methods). Each iteration of these methods requires the knowledge of a subgradient of the entire objective function g (or at least an approximation of a subgradient). For many standard empirical losses, as the ones used in this chapter, a subgradient is readily available. There also exists an explicit expression of the subdifferen-

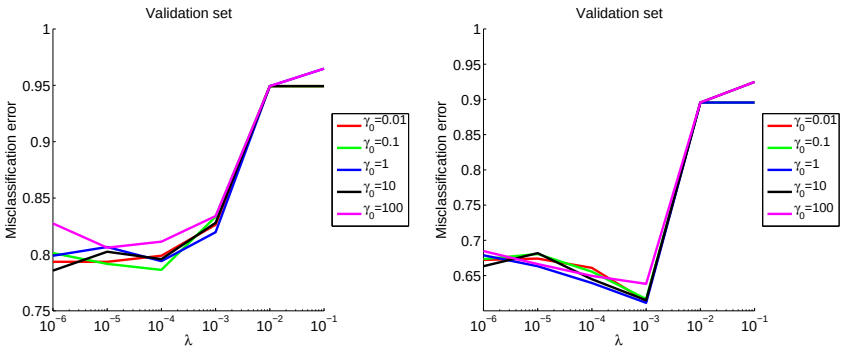

Figure 4.6: Top-k misclassification error on validation set for non adaptive algorithm. Here we choose the best λ and γ. Imagenet dataset. $\omega =$ squared norm. (left) top-5, (right) top-10.

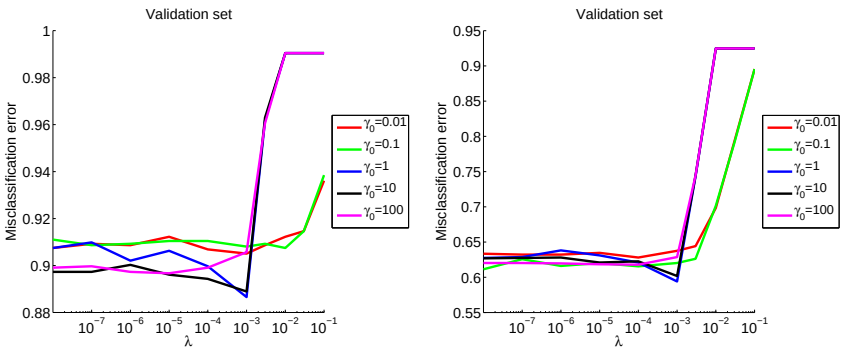

Figure 4.7: Top-kmisclassification error on validation set for non adaptive algorithm. Here we choose the best λ and γ. Imagenet dataset. $\omega =$ entropy. (left) top-5, (right) top-10.

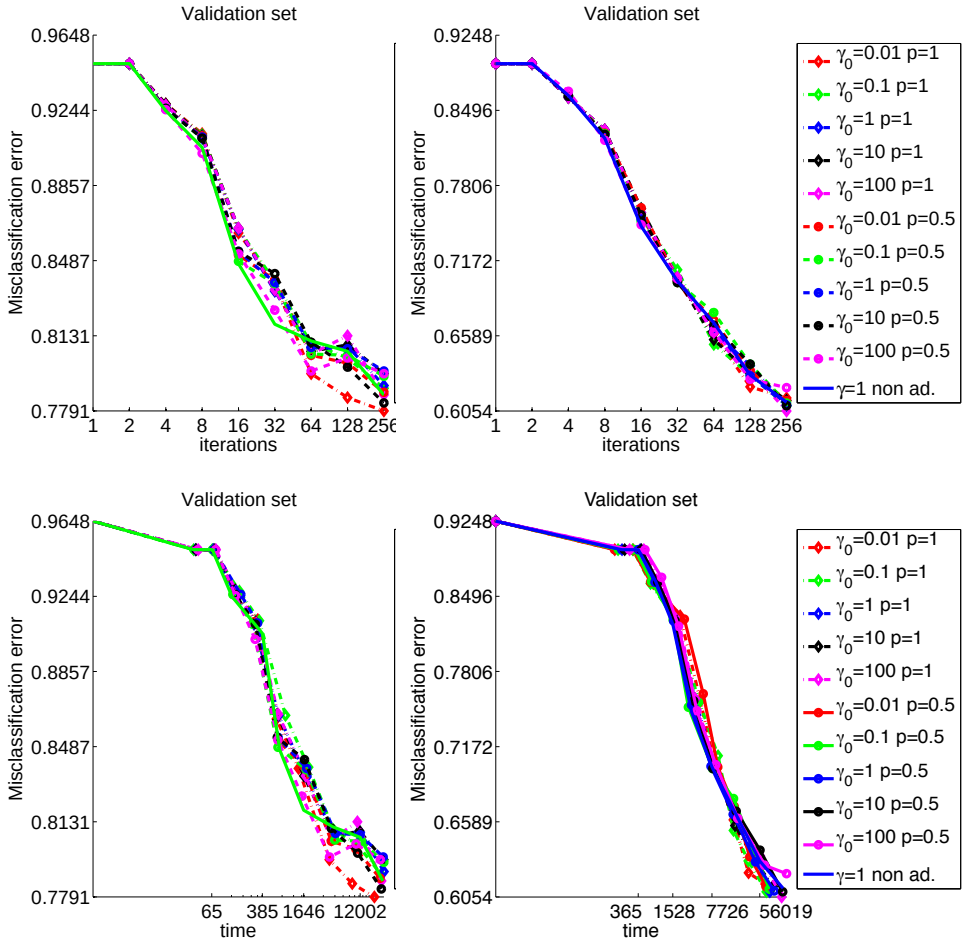

Figure 4.8: Top-k misclassification error on validation set versus iterations (top) and versus time (bottom). (left) top-5, (right) top-10. Time is in seconds. We observe that all the adaptive launches (dot lines) perform better or in a comparable way than the best non-adaptive launch (the blue line at the bottom of the legend). So it is better to use the adaptive algorithm because we don't need to launch several times to find the best initial smoothing parameter γ_0.

tial of the trace-norm: we get a subgradient of the trace-norm at W from an SVD decomposition of W, see Lewis (1999). This is a bottleneck in scaling such approach to large dimension: for the large-scale learning problems we consider, even computing a single SVD (then a single iteration of a nonsmooth optimization algorithm) is out-of-reach in a reasonable amount of time.

To illustrate this fact on collaborative filtering problems, we compare the SCCG algorithm with fixed γ with a tailored basic nonsmooth optimization: truncated subgradient optimization. An iteration of this algorithm iteration writes $W_{k+1} = W_k + t_k G_k$ with G_k approximates a subgradient in $\partial g(W_k)$, and we start from zero. We compute only the 100 largest singular values to construct G_k to save computing time.

Being the stepsize $t_k = \frac{\delta}{t+1}$, the subgradient algorithm is sensitive to the initial stepsize δ.

The comparison of the decrease of the nonsmooth empirical risk is plotted in Figure 4.11. We see that for the medium dataset the decrease of the subgradient method is better with respect of iterations and time, but that the situation is reversed for the large-scale problems. This confirms the discussion above about the prohibitive cost of computing a (even a poorly approximate of a) subgradient.

Again, more efficient algorithm as bundle methods would suffer from the same drawback: even though the algorithms are well-performing, they use information given by an oracle which over-costly for the problems we consider.

Comparison with non-adaptive We compare with fixed smoothing $\gamma_t = \gamma$. As shown in Pierucci et al. (2014), the optimal smoothing parameter γ has to be chosen accordingly to the accuracy ε.

Results on classification on Imagenet are at Figures 4.5.4 and 4.9. We can compare with Figure 4.8, where the choice of initial step does not appear relevant.

Other results on collaborative filtering, appeared on Pierucci et al. (2014), are at Figures 4.8 and 4.8.

Comparison with subgradient optimization algorithm We compared SCCG with subgradient optimization. In Fig. 4.11 we see that the subgradient has a better performance on the medium Movielens dataset. But when the size of the problem increases, as with the large Movielens dataset, the computational time of the singular value decomposition used for the subgradient of trace norm makes the algorithm non scalable from the point of view of the problem size.

4.6 Conclusion

We proposed a composite conditional gradient algorithm that is suitable for regularized learning problems with nonsmooth loss functions, and showed promising experimental results. The framework we used al-

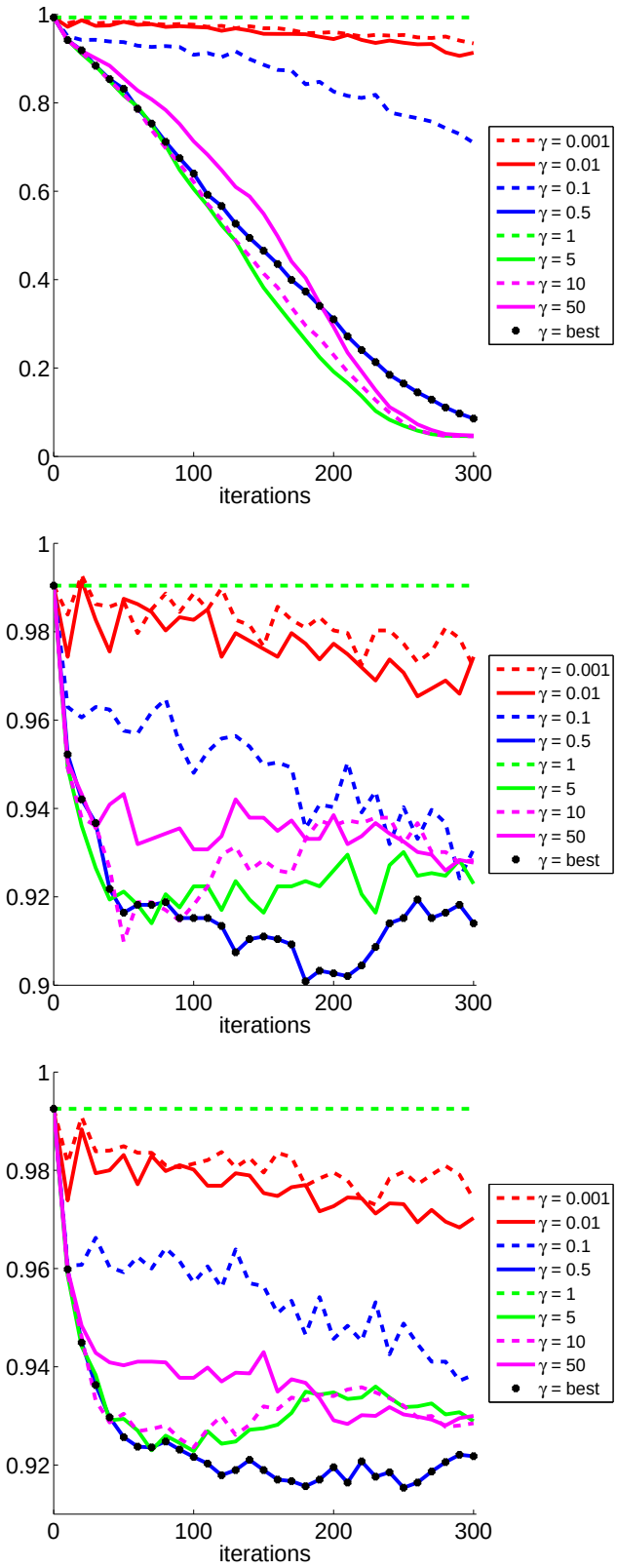

Figure 4.9: Imagenet data - fixed smoothing parameter $\gamma_t = \gamma$ - Misclassification error versus iterations. The best smoothing γ is chosen as the one that minimizes the misclassification error on the validation set. We see that a too small γ correspond to slower convergence, due to a large Lipschitz constant; a too large γ correspond to slower convergence, due to bad approximation of the loss.

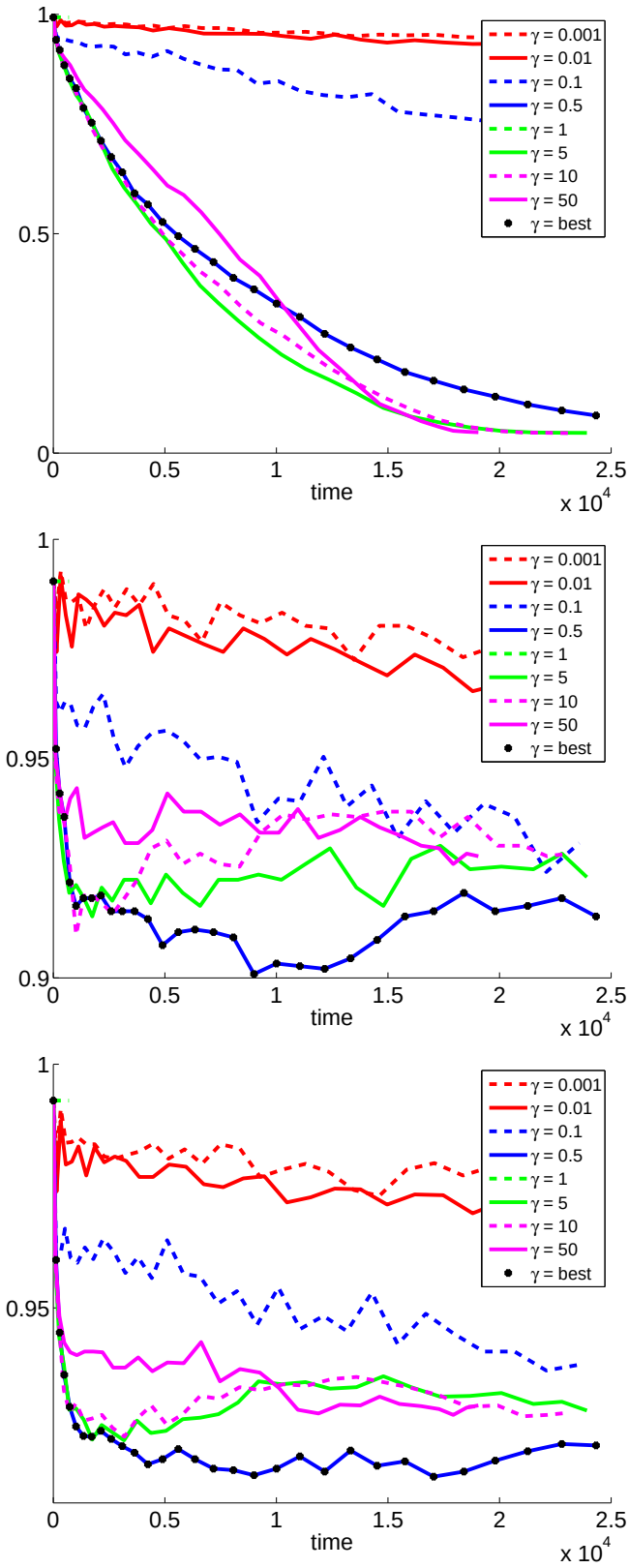

Figure 4.10: Imagenet data - fixed smoothing parameter $\gamma_t = \gamma$ - Misclassification error versus time (in seconds). The best smoothing γ is chosen as the one that minimizes the misclassification error on the validation set. We see that a too small γ correspond to slower convergence, due to a large Lipschitz constant; a too large γ correspond to slower convergence, due to bad approximation of the loss.

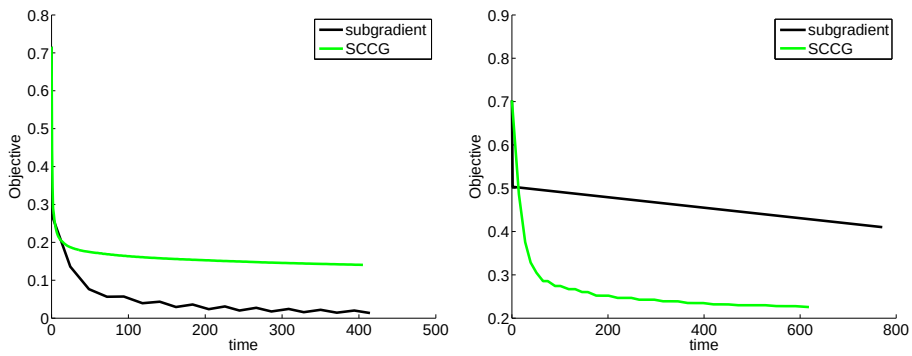

Figure 4.11: Comparison of nonsmooth objective vs time, in seconds. $\gamma_t = 1/\sqrt{t+1}$, $\lambda = 10^{-6}$, on MovieLens datasets: (left) the medium dataset with 10^6 ratings, (right) the large dataset with 10^7 ratings, for the two on the right-hand side. The results for subgradient are dependent on the initial step size $\delta = 500$.

lows to build smoothed counterparts of nonsmooth loss functions in a principled manner, with theoretical guarantees on the accuracy with respect to the original *doubly non-smooth* objective.

4.7 Proofs

In this section, we give the proofs of the results stated in the chapter.

Proof. (of lemma 4.4.1)

By (4.11), we have that $f(W_t) - f(W^\star) \leq f(W_t) - f_t^{\text{rec}}$. We have by construction for all W

$$0 \leq f(W) - f(W, \gamma) \leq \gamma M. \tag{4.36}$$

where M is an upper bound of ω on the (bounded) set $\operatorname{dom} \phi$:

$$0 \leq \omega(z) \leq M \qquad \text{for all } z \in \operatorname{dom} \phi.$$

Splitting the right-hand side with $f(W_t, \gamma_t)$, we get

$$f(W_t) - f(W^\star) \leq f(W_t) - f(W_t, \gamma_t) + \Delta_t^{\text{obs}}$$

which yields (4.12) by (4.36). □

Proof. (of theorem 4.4.3)

For constant $\gamma_t = \gamma_0$, the first term depending of γ_t in (4.13) collapses and the whole formula simplifies by factorizing out γ_0. Then we get (4.19) by combining with Lemma 4.12. □

Proof. (of theorem 4.4.4)

We can bound in (4.13) the two terms depending on γ_t as follows: First,

$$(\gamma_k - \gamma_{k+1})M \leq 2D\sqrt{\frac{M}{c}}\frac{1}{\sqrt{k+1}-\sqrt{k+2}} \leq D\sqrt{\frac{M}{c}}\frac{1}{(k+1)\sqrt{(k+2)}} \leq D\sqrt{\frac{M}{c}}\frac{1}{(k+1)^{3/2}}$$

and second,

$$\frac{2D^2}{c\gamma_k(k+1)^2} \leq D\sqrt{\frac{M}{c}}\frac{1}{(k+1)^{3/2}}$$

Thus, (4.13) yields

$$\Delta_t^{\text{obs}} p \leq 2\sqrt{\frac{M}{c}}\frac{1}{(t+1)t}\sum_{k=1}^{t}\frac{k}{\sqrt{k+1}}.$$

Observe that since $\frac{\sqrt{t+1}}{t} \leq \frac{1}{\sqrt{t-1}}$ we have

$$\sum_{k=1}^{t}\frac{k}{\sqrt{k+1}} \leq \frac{2}{3}(t+1)\sqrt{t+1}.$$

We get finally

$$\Delta_t^{\text{obs}} p \leq \frac{4D}{3}\sqrt{\frac{M}{c}}\frac{1}{\sqrt{t-1}} \tag{4.37}$$

To conclude we combine the above bound with Lemma 4.12. □

Lemma 4.7.1 (Example: dual smoothing by the squared norm)**.** *Let $\gamma > 0$ and $\mathcal{Z}$ convex compact set in $\mathbb{R}^{d\times m}$.*

Consider a function f admitting a (non-unique) Fenchel-type representation defined through the convex conjugate of ϕ (see (Hiriart-Urruty and Lemarechal, 1993, Chap.E)) as

$$f(W) = \phi^*(AW+b) = \sup_{z\in \operatorname{dom}\phi} \langle AW+b, z\rangle - \phi(z), \tag{4.38}$$

where ϕ is a continuous convex function such that its (closed) domain $\operatorname{dom}\phi$ is bounded and $\operatorname{dom}\phi \subset \operatorname{dom}\omega$. Then this structure gives an easy, constructive and controllable way to approximate f by smooth functions.

Nonsmooth $\sigma(\xi)$	Ball $\mathcal{Z}$	Proximity $\omega(z)$	Smooth surrogate $\sigma(\xi,\gamma)$
$\lvert\xi\rvert$	$[-1,1]$	$\frac{1}{2}\lvert\cdot\rvert^2$	$\begin{cases}\frac{1}{2\gamma}\xi^2 & \text{if } \lvert\xi\rvert\le\gamma\\ \lvert\xi\rvert-\frac{\gamma}{2} & \text{if } \lvert\xi\rvert>\gamma\end{cases}$
$\lvert\xi\rvert$	$[-1,1]$	$(1-\lvert z\rvert)\ln(1-\lvert z\rvert)+\lvert z\rvert$	$f(\xi,\gamma)=\gamma e^{-\left\lvert\frac{\xi}{\gamma}\right\rvert}+\lvert\xi\rvert-\gamma$
$\max_i\{\xi_i,0\}$	$\mathrm{co}(\Delta_n\cup\{\mathbf{0}\})$	$\frac{1}{2}\lVert\cdot\rVert^2$	$\left\langle\xi,\pi_{\mathcal{Z}}\left(\frac{\xi}{\gamma}\right)\right\rangle-\frac{\gamma}{2}\left\lVert\pi_{\mathcal{Z}}\left(\frac{\xi}{\gamma}\right)\right\rVert^2$
$\max_i\{\xi_i,0\}$	$\mathrm{co}(\Delta_n\cup\{\mathbf{0}\})$	$1+\sum_{i=1}^n z_i\log(z_i)-z_i$	$\begin{cases}\gamma\left(-1+\sum_{i=1}^n\exp\left(\xi_i/\gamma\right)\right) & \text{if } \frac{\xi}{\gamma}\in C\\ \gamma\log\left(\sum_{i=1}^n\exp\left(\xi_i/\gamma\right)\right) & \text{if } \frac{\xi}{\gamma}\in B\end{cases}$
$\frac{1}{k}\sum_{i=1}^k\xi_{\alpha(i)}$	$\left\{z\,\middle\vert\,\sum z_i\le 1;\ z_i\in\left[0,\frac{1}{k}\right]\right\}$	$\frac{1}{2}\lVert\cdot\rVert^2$	$\left\langle\xi,\pi_{\mathcal{Z}}\left(\frac{\xi}{\gamma}\right)\right\rangle-\frac{\gamma}{2}\left\lVert\pi_{\mathcal{Z}}\left(\frac{\xi}{\gamma}\right)\right\rVert^2$
$\frac{1}{k}\sum_{i=1}^k\xi_{\alpha(i)}$	$\left\{z\,\middle\vert\,\sum z_i\le 1;\ z_i\in\left[0,\frac{1}{k}\right]\right\}$	$\sum_{i=1}^n z_i\ln(nz_i)$	$\Theta(\lambda_\star(\xi,\gamma))$ (solve dual problem)

Table 4.1: On the first line we obtain the Huber function, third and fourth lines we have the smoothing of the multiclass hinge, 5th and 6th line: smoothing of the top-k error. $C:=\{s\in\mathbb{R}^n\mid\sum_{i=1}^n\exp(s_i)\le 1\}$ and $B:=\{s\in\mathbb{R}^n\mid\sum_{i=1}^n\exp(s_i)>1\}$. We assume that $0\log 0=1$. α is the permutation that orders in decreasing order: $x_{\alpha(1)}=\max_i x_i$.

Indeed, consider the support function of $\mathcal{Z}$, where $\phi=i_{\mathcal{Z}}$ is the indicator function of $\mathcal{Z}$. If $\omega(\cdot)=\frac{1}{2}\lVert\cdot\rVert_2^2$ and the projection $\pi_{\mathcal{Z}}(s)=\mathrm{argmin}_{v\in\mathcal{Z}}\lVert v-s\rVert$, then

$$f(W,\gamma)=\langle\pi_{\mathcal{Z}}\left((AW+b)/\gamma\right),(AW+b)\rangle-\frac{\gamma}{2}\lVert\pi_{\mathcal{Z}}((AW+b)/\gamma)\rVert_2^2.$$

The gradient $\nabla_W f(W,\gamma)=A^\dagger\pi_{\mathcal{Z}}\left((AW+b)/\gamma\right)$ has Lipschitz constant $L=1/\gamma$. In practice, such a function is interesting only if the projection $\pi_{\mathcal{Z}}$ is fast to compute.

Proof. We consider all the cases that share in common the use of squared norm as proximity function.

We start with $\gamma=1$:

$$\begin{aligned}\nabla\sigma(\xi,1)&=\operatorname*{argmax}_{z\in\mathcal{Z}}\langle z,\xi\rangle-\omega(z)=\operatorname*{argmin}_{z\in\mathcal{Z}}\tfrac{1}{2}\lVert z\rVert^2-\langle\xi,z\rangle\\&=\operatorname*{argmin}_{z\in\mathcal{Z}}\lVert\xi-z\rVert^2-\tfrac{1}{2}\lVert\xi\rVert^2=\operatorname*{argmin}_{z\in\mathcal{Z}}\lVert\xi-z\rVert^2=\pi_{\mathcal{Z}}(\xi),\end{aligned}$$

from which we get $\sigma(\xi,1)=\langle z(\xi,1),\xi\rangle-\frac{1}{2}\lVert z(\xi,1)\rVert^2=\langle\pi_{\mathcal{Z}}(\xi),x\rangle-\frac{1}{2}\lVert\pi_{\mathcal{Z}}(\xi)\rVert^2$.

Obviously $\omega=\frac{1}{2}\lVert\cdot\rVert_2^2$ is strongly convex with modulus 1. The gradient of $f(\cdot,1)$ at ξ coincides with $z(\xi)$, and is Lipschitz continuous with Lipschitz constant is $L=1$. We can also this property directly from the above expression of the gradient and the properties of the projection (which is 1-Lipschitz).

□

Lemma 4.7.2. *Let A be the function defined on $[a,b]$ compound of two segments such that $Aa=Ab=0$*

and $A(\frac{a+b}{2}) = 1$. *Let* $h(t) := At(\ln(At) - 1)$. *Then* h *is strongly convex in* $[a, b]$ *with constant* $\alpha = \frac{4}{(b-a)^2}$.

Proof. We define $t^\star := (a+b)/2$. For $t \neq t^\star$ we define the derivative $|A't| =: v$ and observe that $v = \frac{2}{b-a}$. We claim that h is twice differentiable. For $t \neq t^\star$ we compute the derivative $h'(t) = A't(\ln At - 1) + A'(t) = A't \ln At$. $\lim_{t\to t^\star} A't \ln At = 0$. Then h is differentiable in $]a, b[$ and $h'(t) = A't \ln At$. For $t \neq t^\star$ we compute the second derivative $h''(t) = \frac{A't}{At} A't$. $\lim_{t\to t^\star} \frac{(A't)^2}{At} = \lim_{t\to t^\star} \frac{v^2}{At} = \frac{v^2}{At^\star}$. Then h is twice differentiable in $]a, b[$ and $h''(t) = \frac{v^2}{At}$. We claim h is strongly convex. We have $At \leq 1 \Rightarrow \frac{1}{At} \geq 1 \Rightarrow \frac{v^2}{At} \geq v^2$. Then $h''(t) \geq v^2$. Then h is strongly convex with constant $\alpha = v^2$ in $]a, b[$. Because h is bounded this property is preserved to the limits, and then h is strongly convex also in $[a, b]$. □

Lemma 4.7.3. *Let* $h(t) := \begin{cases} (1 - |2t-1|)\ln(1 - |2t-1|) + |2t-1| & \textit{if } t \in]0,1[\\ 1 & \textit{if } t \in \{0,1\}. \end{cases}$

Then the derivative is

$$h'(t) = \begin{cases} 2\ln 2t & \textit{if } t \in]0, \frac{1}{2}[\\ -2\ln(2-2t) & \textit{if } t \in [\frac{1}{2}, 1[. \end{cases}$$

and $h'' \geq 4$ *in* $]0, 1[$.

Proof.

$$h(t) := \begin{cases} (1 - |2t-1|)\ln(1 - |2t-1|) + |2t-1| & \text{if } t \in]0,1[\\ 1 & \text{if } t \in \{0,1\} \end{cases}$$

$$= \begin{cases} 2t\ln(2t) + 1 - 2t & \text{if } t \in]0, \frac{1}{2}[\\ (2-2t)\ln(2-2t) + 2t - 1 & \text{if } t \in [\frac{1}{2}, 1[\\ 1 & \text{if } t \in \{0,1\} \end{cases}$$

$$h'(t) = \begin{cases} 2\ln 2t & \text{if } t \in]0, \frac{1}{2}[\\ -2\ln(2-2t) & \text{if } t \in [\frac{1}{2}, 1[. \end{cases}$$

$$h''(t) = \begin{cases} \frac{2}{t} & \text{if } t \in]0, \frac{1}{2}[\\ \frac{2}{1-t} & \text{if } t \in [\frac{1}{2}, 1[. \end{cases}$$

In addition h'' has minimum in $\frac{1}{2}$. Then $h''(t) \geq h''(\frac{1}{2}) = 4$.

□

Lemma 4.7.4. *The function $\omega(z) = \sum_{i=1}^{n} h(z_i)$, $z \in \mathbb{R}^n$, where we define the entropy-like function*

$$h(t) := \begin{cases} (1-|2t-1|)\ln(1-|2t-1|) + |2t-1| & \text{if } t \in]0,1[\\ 1 & \text{if } t \in \{0,1\} \end{cases}$$

is strongly convex with constant $\alpha = 4$.

Proof. We use Lem. 4.7.3 on each component of the sum. □

4.8 Additional results

Proposition 4.8.1. *(If the proximity function is not differentiable on the boundary of $\mathcal{B}$, i.e.) If the norm of all subgradients of ω have limit to infinity for y approaching the boundary, then the minimizer $y_\star$ is in the interior of the ball, i.e. if this condition is valid:*

$$\forall \bar{y} \in fr(\mathcal{B})\ \forall g_y \in \partial\omega(y) \quad \lim_{y \to \bar{y}} \|g_y\| = \infty$$

(euclidean norm in $\mathbb{R}^n$)

then the maximizer is in the interior of the ball:

$$y_\star \in \overset{\circ}{\mathcal{B}}$$

Proof. Let as reduce to absurd and suppose that $y_\star$ is in the boundary of the ball. For an optimal $\bar{y}$ we have $0 \in \partial_y(\langle x, \cdot\rangle - \omega)(\bar{y}) \Leftrightarrow 0 \in x - \partial\omega(\bar{y}) \Leftrightarrow x \in \partial\omega(\bar{y})$. As $x \in \mathbb{R}^n$ and $x \in \partial\omega(y_\star)$ we deduce that $\partial\omega(y_\star)$ is bounded. Absurd. □

Entropy on polytopes We want to show here how to define a proximity function based on the entropy for any polytope. The given polytope P is defined as the convex envelope of the set $\{a_i\}_{i\in\mathcal{I}}$ in $\mathbb{R}^m$.

We can express $z \in \mathbb{R}^m$ with barycentric coordinates in a "lifted" space

$$z = \sum_{i\in\tilde{\mathcal{I}}} \theta_i a_i$$

where $\tilde{\mathcal{I}} \subset \mathcal{I}$ is the finite set of indices corresponding to nozero θ_i, and the vector containing θ_i is on a simplex of dimension $\left|\tilde{\mathcal{I}}\right|$, i.e. $\sum_{i\in\tilde{\mathcal{I}}} \theta_i = 1$ and $\theta_i \geq 0$. This decomposition is not unique. In practice

we can build a linear continuous map from the simplex to the polytope P

$$z_\theta := \psi(\theta) := \sum_{i \in \mathcal{I}} \theta_i a_i \tag{4.39}$$

See Dudik et al. (2012) for more details. We define the proximity function

$$\omega(z_\theta) := \sum_{i \in \mathcal{I}} (1 - \theta_i) \ln(1 - \theta_i) + \theta_i \tag{4.40}$$

We show now the convexity of ω using the Fenchel conjugate, showing $\omega^{**} = \omega$. We suppose here P has a finite number of corners.

$$\omega(\theta) := \sum_i \theta_i \log(K\theta_i) \tag{4.41}$$

If we define $h(\theta_i) := \xi_i \theta_i - \theta_i \log(K\theta_i)$, then the derivative $h'(\theta_i) = \xi_i - \log(K\theta_i) - 1$ and the optimality conditions correspond to $h'(\theta_i) = 0 \Leftrightarrow \theta_{i\star} = \frac{1}{K} e^{\xi_i - 1} > 0$.

$$\begin{aligned}
\omega^*(\xi) &= \max_{\theta_i \geq 0} \langle \xi, \theta \rangle - \omega(\theta) && (4.42)\\
&= \max_{\theta_i \geq 0} \langle \xi, \theta \rangle - \sum_i \theta_i \log(K\theta_i) && (4.43)\\
&= \max_{\theta_i \geq 0} \sum_i \xi_i \theta_i - \theta_i \log(K\theta_i) && (4.44)\\
&= \sum_i \max_{\theta_i \geq 0} \xi_i \theta_i - \theta_i \log(K\theta_i) && (4.45)\\
&= \sum_i \xi_i \theta_{i\star} - \theta_{i\star} \log(K\theta_{i\star}) && (4.46)\\
&= \sum_i \xi_i \tfrac{1}{K} e^{\xi_i - 1} - \tfrac{1}{K} e^{\xi_i - 1}(\xi_i - 1) && (4.47)\\
&= \frac{1}{K} \sum_i e^{\xi_i - 1} && (4.48)\\
& && (4.49)
\end{aligned}$$

Now we find the double conjugate.

If we define $g(\xi_i) := \xi_i \theta_i - \frac{1}{K} e^{\xi_i - 1}$, then the derivative is $g'(\xi_i) = \theta_i - \frac{1}{K} e^{\xi_i - 1}$ and the optimality conditions correspond to $g'(\xi_i) = 0 \Leftrightarrow \xi_{i\star} = \log(K\theta_i) + 1$.

$$\omega^{**}(\theta) = \max_{\xi \in \mathbb{R}^n} \langle \theta, \xi \rangle - \omega^*(\xi) \tag{4.50}$$

$$= \max_{\xi \in \mathbb{R}^n} \langle \theta, \xi \rangle - \frac{1}{K} \sum_i e^{\xi_i - 1} \tag{4.51}$$

$$= \max_{\xi \in \mathbb{R}^n} \sum_i \xi_i \theta_i - \frac{1}{K} e^{\xi_i - 1} \tag{4.52}$$

$$= \sum_i \max_{\xi_i \in \mathbb{R}} \xi_i \theta_i - \frac{1}{K} e^{\xi_i - 1} \tag{4.53}$$

$$= \sum_i \xi_{i_\star} \theta_i - \frac{1}{K} e^{\xi_{i_\star} - 1} \tag{4.54}$$

$$= \sum_i (\log(K\theta_i) + 1)\theta_i - \theta_i \tag{4.55}$$

$$= \sum_i \theta_i \log(K\theta_i) \tag{4.56}$$

$$= \omega(\theta) \tag{4.57}$$

We have also $\min_\theta \omega(\theta) = -\max_{\theta_i \geq 0} \langle 0, \theta \rangle - \omega(\theta) = -\omega^*(0) = -\frac{1}{K} \sum_{i=1}^n e^{-1} = -\frac{n}{Ke}$

Strong convexity.

We show that (4.41) is strongly convex with constant $c = 1$. The function $t \mapsto t \log(Kt)$, for $t \in [0, 1]$ is strongly convex with constant $c = 1$. In fact $(t \log(Kt))' = \log(Kt) + 1$ and $(t \log(Kt))'' = \frac{1}{t}$, which has minimum 1. This is equivalent to $s \log(Ks) \geq t \log(Kt) + (\log(Kt) + 1)(s - t) + \frac{1}{2}(s - t)^2$.

We sum up and obtain

$$\begin{aligned}
&\forall i \quad y_i \log(Ky_i) \geq x_i \log(Kx_i) + (\log(Ky_i) + 1)(y_i - x_i) + \frac{1}{2}(y_i - x_i)^2 \\
\Rightarrow \quad & \sum_i y_i \log(Ky_i) \geq \sum_i x_i \log(Kx_i) + \sum_i (\log(Ky_i) + 1)(y_i - x_i) + \frac{1}{2} \sum_i (y_i - x_i)^2 \\
\Rightarrow \quad & \omega(y) \geq \omega(x) + \langle \nabla \omega(x), y - x \rangle + \frac{1}{2} \|y - x\|^2
\end{aligned}$$

Then also ω is strongly convex with constant $c = 1$.

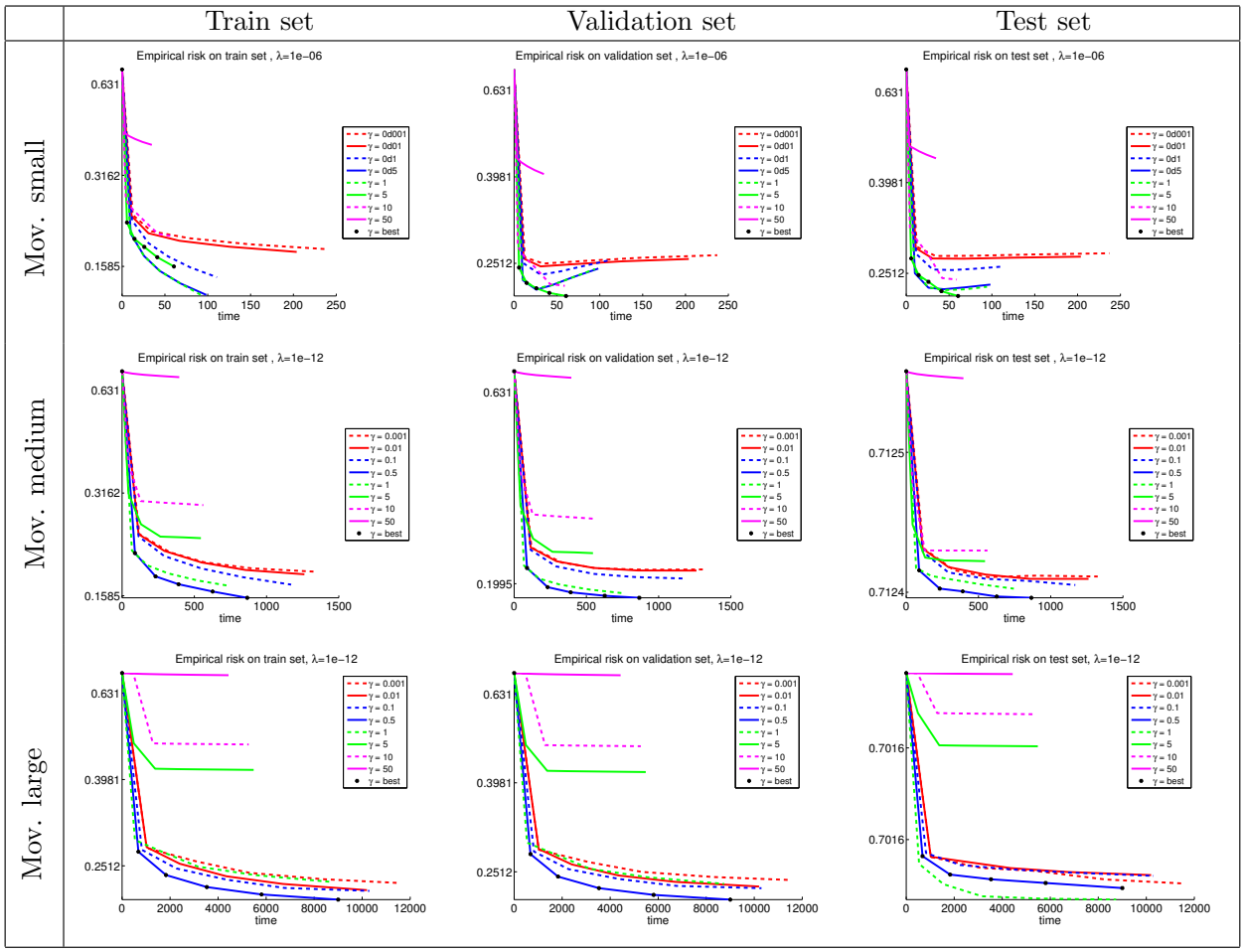

Figure 4.12: Movielens data - non-adaptive algorithm - Empirical risk versus time. Related to all γ for the best choice of λ.

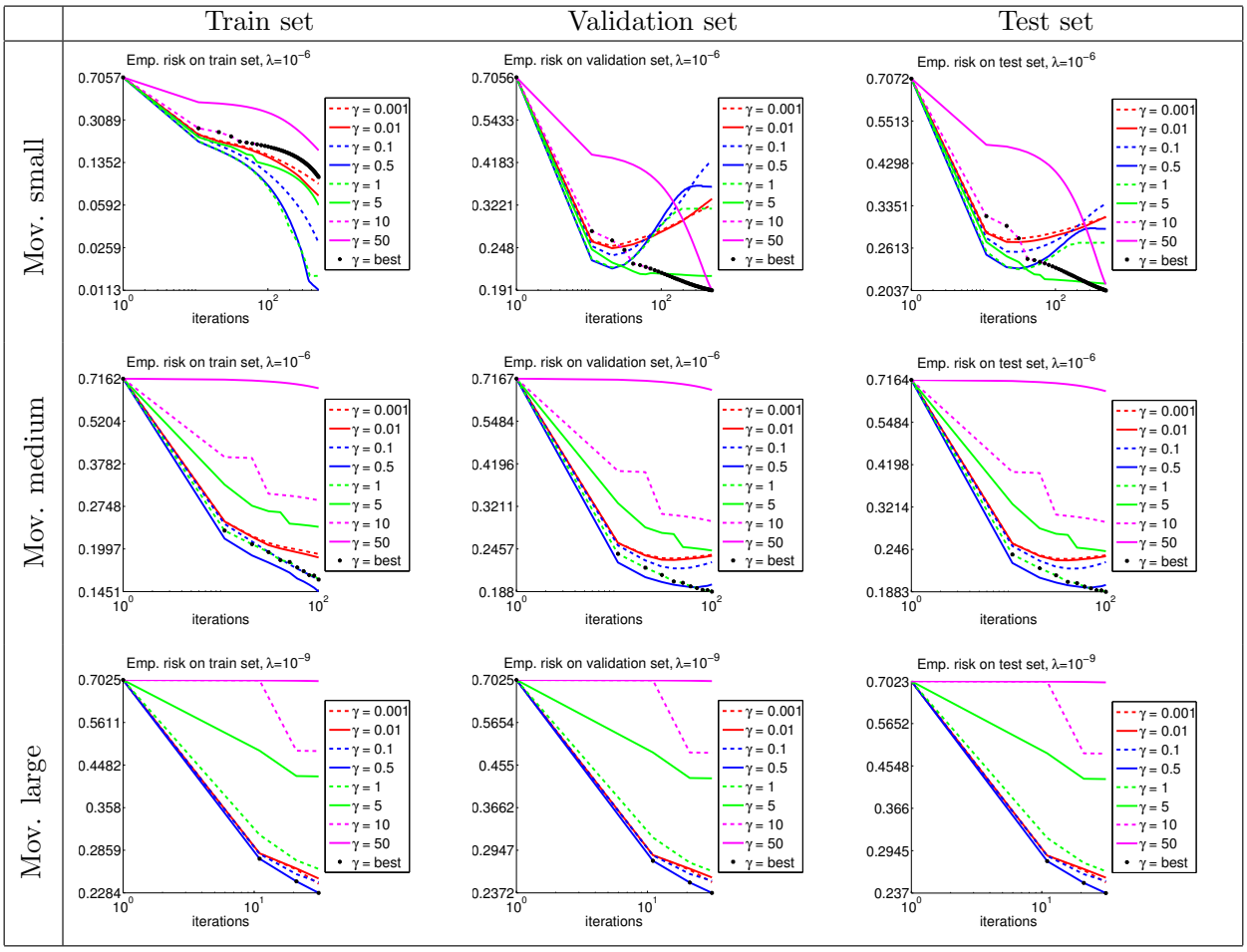

Figure 4.13: Movielens data - non-adaptive algorithm - Empirical risk versus iterations.

Chapter 5

Conclusion

In our work we have explored three interconnected parts, that are aspects of nonsmooth optimization, statistical learning, and structured matrix regularization. We have reviewed existing approaches and presented our contributions. In this section we summarize our contributions and propose possible future directions of research.

5.1 Summary of contributions

Group Schatten norm As first contribution, we introduced a new mathematical object, the group Schatten norm. This is an extension of the Schatten norm to matrices that are composed into blocks. As a central result, we proved that this norm is the convex envelope of the re-weighted group rank function. We studied then its properties from the point of view of convex analysis and we analyzed one particular instance, called *group nuclear norm*. This new norm used as regularizer in an optimization problem has the potential to aggregate into one object several low rank models. The potential applications of the group nuclear norm include collaborative filtering, database compression, and multi attribute image classification.

Smoothing techniques for first-order optimization In our second contribution, we described several *smoothing techniques* building surrogates to approximate non-smooth functions. We applied these techniques to several machine learning models resp. for collaborative filtering for movie recommendation and multi-class image classification. We also derived a smoothed counterpart of the top-k misclassification error, popular in ranking and multi-class classification. We outlined three algorithms for non-smooth

problems, by combining state of art algorithms - composite conditional gradient, FISTA, and MISO - with these smoothing techniques.

Adaptive smoothing conditional gradient algorithms As our third contribution, we proposed two new algorithms for learning problems where the non-smoothness lies both in the convex empirical risk and in the regularization penalty. The corresponding *doubly non-smooth* optimization problems prevents the use of efficient first-order optimization algorithms. The main idea for our algorithms is that we combine the smoothing techniques with the conditional gradient algorithm. We proved the convergence of the proposed adaptive conditional gradient algorithms toward the solution of the initial non-smooth problem. We presented promising experimental results on applications to collaborative filtering for movie recommendation and multi-class classification of images.

5.2 Potential future research topics

Quantitative finance We have developed non-smooth optimization algorithms with structured matrix regularization, with applications to multi-class classification and collaborative filtering. Another potential application for which our algorithm could be used is quantitative finance. Portfolio management, based on Markowitz portfolio theory Markowitz (1952), boils down to modeling the covariance matrix between the assets by solving an optimization problem over the set of positive semi-definite matrices with constraints encoding the structure, see e.g. Brodie et al. (2009). The group nuclear-norm could be used to derive convex relaxation counterparts of portfolio optimization problems where the assets can be divided into groups of assets, in which a dependence is suspected. Group nuclear-norm could here lead to finer models than a "flat" (without group-structure) nuclear-norm constraint.

Learning with hierarchies and feature concatenation Applications in computer vision, as instance object classification and object identification, can take advantage of rich high-dimensional feature vectors. A common way to deal the issue of the size of features is to first reduce the feature dimension with a PCA for example, and then to run the learning algorithm. A drawback is that the dimension reduction deletes part of information present in the examples and, by consequence, the accuracy of the obtained model is reduced.

The group Schatten norm could be used here to regularize learning problems for multiclass classification and feature concatenation with various learning tasks and huge feature vectors. Two cases for multi-class classification could be considered : (i) the classes are organized in a hierarchical way, with

possibly more than one hierarchy at the same time, and (ii) there are attributes assigned the examples, independently from the hierarchy. When concatenating features, the group Schatten norm could keep the features divided into groups, and we could use this information to build more robust models. For instance, in action recognition, local features in an image can be combined to obtain a model that localizes automatically the part of the image with the action.

Real-time systems The computational power and memory of current devices, as instance the smartphones, are small and need light learning tools. By 'embedded' we mean that the optimization algorithm is part of a larger, fully automated system, that executes automatically with newly arriving data or changing conditions, and without any human intervention or action. By 'real-time' we mean that the optimization algorithm executes much faster than a typical or generic method with a human in the loop, in times measured in milliseconds or microseconds for small and medium size problems, and (a few) seconds for larger problems Mattingley and Boyd (2009, 2010). The optimization algorithms we proposed produce sequence of iterates that are sparse, hence allowing low-memory (e.g. RAM) requirements to run the algorithms. Furthermore, the number of floating point operations at test time is reduced. Therefore, the algorithms could potentially be run efficiently on CPU of common devices. Conditional gradient algorithms could potentially be run real-time in embedded systems, for instance for recommendation and classification.

Index

Accuracy, 19
Atom, 20
Atomic norm, 20

Black-box algorithm, 26

Certificate, 27
Class of functions, 25
Complexity, 24
Complexity bound, 28
Compression, 39
Conditional gradient, 42
Constrained optimization problem, 18
Convergence, 25
    rate of, 25
Convex
    conjugate, 17
    function, 16
Convex function, 16

Descent direction, 31
Dictionary, 20

Easy to compute, 24
Empirical risk, 14

Fast iterative algorithm, 34
Feasible
    direction, 40
    point, 40
Fenchel conjugate, 17
First order algorithm, 30
Frank-Wolfe, 42

Gauge function, 20
Greedy
    algorithm, 36
    step, 36

Hinge function, 111

Ill-posed optimization problem, 14
Indicator function, 18
Iterate, 25

Learning machine, 13
Line search, 31
Linear rate, 32
Lower bound, 26

Minkowski functional, 20

Non-anticipating rule, 30

Objective, 18
Operator
    Linear minimization operator , 22
    Moreau-type proximal operator, 21
Optimal

algorithm, 27
solution, 19
Optimal algorithm, 27
Optimality conditions, 19
Oracle, 20
Overfitting, 15

Performance, 24
Predictor function, 13
Projection operator, 21

Regularization, 15

Scalable algorithm, 28
Smooth function, 16
Solve a problem, 20
Step size, 31
Strongly convex function, 17
Subgradient, 16
Sublinear rate, 31
Support, 44
Support function, 17

Termination criterion, 29
Test set, 16
Train set, 14

Unconstrained optimization problem, 18
Upper bound, 25

Validation set, 15

List of Figures

1.1 Two predictor functions $\mathcal{F}(\cdot, \mathbf{W}_1)$ and $\mathcal{F}(\cdot, \mathbf{W}_2)$ based on data $(\mathbf{x}_i, \mathbf{y}_i)$. 13

1.2 The optimal solution $\mathbf{x}_\star$ and an ε-optimal solution $\tilde{\mathbf{x}}_\star$ for the optimization of f. 19

1.3 Prox of $g(\mathbf{x}) = \lambda \|\mathbf{x}\|_1$. Axis represent x^1, x^2 . Each arrow represents $\mathrm{prox}_{\gamma g}(\mathbf{x})$ and is placed at $\mathbf{x}$. 22

1.4 Linear minimization operator on a closed convex set $\mathcal{Q}$ with respect to the direction $\mathbf{s}$. $\mathbf{d} = LMO(\mathbf{x}_t)$. The dashed lines are orthogonal to $\mathbf{s}$. It is possible to see that the descent direction $\mathbf{d}$ is not aligned with $\mathbf{s}$, but is directed to a corner of $\mathcal{Q}$. This corner corresponds to the element of the dictionary that will be added to the sequence of atoms that compose the next iterate $\mathbf{x}_{t+1}$. 23

1.5 Conditional gradient. At $\mathbf{x}_t$ the algorithm finds the gradient, then builds the tangent r. The minimum of r over $\mathcal{Q}$ is $\mathbf{d}_t$, the output of the linear minimization operator. The certificate at iteration t is G_t 42

1.6 View of first 200 rows and columns of Movielens data. **(Left)** the input matrix of observations $\mathbf{Y}$, where the non observed ratings are drawn in white, **(right)** the low rank solution $\mathbf{W}$ obtained running SCCG. The highest ratings are black, then red. The more poor ratings are and yellow and white. 49

1.7 The matrix on the left is composed by the matrices on the right, that we call 'groups'. . . 52

2.1 Initial matrix $\mathbf{L}$ to recover. It is generated as the sum of matrices of rank two, each defined over its respective group of indices. The groups overlap. 71

2.2 **(Left)** Noisy matrix $\mathbf{A}_{0.2}$, where only 10% of matrix entries are observed. Unknown entries are white. **(Right)** Solution matrix $\widehat{\mathbf{W}}$ recovered with $\lambda = 3.6 \cdot 10^{-6}$. This gives an error of 0.0067. We see that both the big overlapping groups and the small ones are recovered. The group on the left side, which consists of a grid with different spacing in rows and columns, and that has sinusoidal values, is also well recovered. 72

2.3 Hierarchical organisation of images in ImageNet dataset. Figure adapted from Deng et al. (2010). We see that the classes `catamaran`, `trimaran`, `catboat`, `sailboat`, `galleon`, `sailing vessel` are organized in a tree graph. 73
2.4 We see the observable volume that is projected into the depth image, which is dark gray for close objects, *e.g.* a hand grasping something, and light grey for far objects. Picture from Rogez et al. (2014) 74

3.1 Smoothed $g_\gamma(\mathbf{x})$ of equation (3.21) and its gradient, $\mathbf{x} \in \mathbb{R}^2$ for top-k error. **Top** with $k = 1$, **bottom** with $k = 2$. The color lines represent level sets and the arrows are the gradients of the smooth surrogate. 87
3.2 Smoothed $g_\gamma(\mathbf{x})$ of equation (3.21) for $k = 1$, between the nonsmooth bounds g (below) and $g + \gamma m$ (above). Compare with equation 3.8. 88
3.3 Illustration of the derivative of the dual function Θ, for $\gamma = 1/2$, smooth surrogate of top-k error. Red: points without second derivative; black: point of minimum of Θ; green: bounds of the 'segment' (with second derivative) that contains the minimum of Θ. The x_i are randomly sampled. 91
3.4 Level sets of smooth surrogates obtained with product convolution technique, through uniform random sampling, instead of using an exact formula. Gradients are represented by arrows. **First column** Smoothing of hinge; **second column** Smoothing of $\|\cdot\|_1$; **third column** Smoothing of $\|\cdot\|_\infty$. **First row** Gaussian distribution with mean 0 and standard deviation 1; **second row** uniform distribution on norm ∞ ball, with radius $r = 1$; **third row** uniform distribution on norm 1 ball, with radius $r = 1$. 97
3.5 Piecewise affine function 110
3.6 SVM with reject loss. **Left** for label $+1$; **Right** for label -1. We notice that the positions of t_1 and t_2 are different in the 2 cases. In addition $t_1^+ = t_2^-$. 114

4.1 Drawing of a situation where the algorithm using a subgradient of a nonsmooth empirical risk does not converge. 121
4.2 Nonsmooth empirical risk, $\lambda = 10^{-6}$. (left) Train set, (center) validation set, (right) test set. (Top) plot versus iteration number, (bottom) plots versus time. We see that some launches with adaptive smoothing perform better than the best non-adaptive algorithm, represented by the continuous line. Collaborative filtering. 137
4.3 Nonsmooth empirical risk versus time, for $\lambda = 10^{-6}$ on train set. Collaborative filtering. 137

4.4 **choice of** λ - Nonsmooth empirical risk at last iteration on validation set. Here we chose the best lambda for regularization and the best γ_0 non adaptative, i.e. with $p = 0$. Collaborative filtering. 138
4.5 Nonsmooth empirical risk at last iteration for $\lambda = 10^{-6}$ on train set. We see on the train set that the best optimization if for $p = 0.5$. Collaborative filtering. 138
4.6 Top-k misclassification error on validation set for non adaptive algorithm. Here we choose the best λ and γ. Imagenet dataset. ω = squared norm. (left) top-5, (right) top-10. 139
4.7 Top-kmisclassification error on validation set for non adaptive algorithm. Here we choose the best λ and γ. Imagenet dataset. ω = entropy. (left) top-5, (right) top-10. 139
4.8 Top-k misclassification error on validation set versus iterations (top) and versus time (bottom). (left) top-5, (right) top-10. Time is in seconds. We observe that all the adaptive launches (dot lines) perform better or in a comparable way than the best non-adaptive launch (the blue line at the bottom of the legend). So it is better to use the adaptive algorithm because we don't need to launch several times to find the best initial smoothing parameter γ_0. 140
4.9 Imagenet data - fixed smoothing parameter $\gamma_t = \gamma$ - Misclassification error versus iterations. The best smoothing γ is chosen as the one that minimizes the misclassification error on the validation set. We see that a too small γ correspond to slower convergence, due to a large Lipschitz constant; a too large γ correspond to slower convergence, due to bad approximation of the loss. 142
4.10 Imagenet data - fixed smoothing parameter $\gamma_t = \gamma$ - Misclassification error versus time (in seconds). The best smoothing γ is chosen as the one that minimizes the misclassification error on the validation set. We see that a too small γ correspond to slower convergence, due to a large Lipschitz constant; a too large γ correspond to slower convergence, due to bad approximation of the loss. 143
4.11 Comparison of nonsmooth objective vs time, in seconds. $\gamma_t = 1/\sqrt{t+1}$, $\lambda = 10^{-6}$, on MovieLens datasets: (left) the medium dataset with 10^6 ratings, (right) the large dataset with 10^7 ratings, for the two on the right-hand side. The results for subgradient are dependent on the initial step size $\delta = 500$. 144
4.12 Movielens data - non-adaptive algorithm - Empirical risk versus time. Related to all γ for the best choice of λ. 151
4.13 Movielens data - non-adaptive algorithm - Empirical risk versus iterations. 152

List of Tables

1 Summary table of notation, that is defined step by step in the text. 11

1.1 **First column** first order algorithms for unconstrained optimization, **second column** first order algorithms for constrained optimization, **third column** magnitude of analytical complexity. 46

1.2 Summary of the notation used in the sections related to applications. We remind that here $\mathbf{x}$ and $\mathbf{y}$ are related to the dataset and have a completely different meaning of the $\mathbf{x}$ and $\mathbf{y}$ used in the sections about optimization. We do not minimize with respect to $\mathbf{x}$ or $\mathbf{y}$, but we minimize with respect to $\mathbf{W}$. 47

4.1 On the first line we obtain the Huber function, third and fourth lines we have the smoothing of the multiclass hinge, 5th and 6th line: smoothing of the top-k error. $C := \{s \in \mathbb{R}^n \mid \sum_{i=1}^n \exp(s_i) \leq 1\}$ and $B := \{s \in \mathbb{R}^n \mid \sum_{i=1}^n \exp(s_i) > 1\}$. We assume that $0 \log 0 = 1$. α is the permutation that orders in decreasing order: $x_{\alpha(1)} = \max_i x_i$. . 146

Bibliography

Akata, Z., Perronnin, F., Harchaoui, Z., and Schmid, C. (2014). Good practice in large-scale learning for image classification. *IEEE*.

Amit, Y., Fink, M., Srebro, N., and Ullman, S. (2007). Uncovering shared structures in multiclass classification. In *ICML*.

Bach, F. (2008). Consistency of trace norm minimization. *The Journal of Machine Learning Research*.

Bach, F., Lacoste-Julien, S., and Obozinski, G. (2012a). On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12.

Bach, F. R., Jenatton, R., Mairal, J., and Obozinski, G. (2012b). Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106.

Bartlett, P. and Wegkamp, M. (2008). Classification with a reject option using a hinge loss. *The Journal of Machine Learning Research*.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*.

Beck, A. and Teboulle, M. (2012). Smoothing and first order methods: a unified framework. *SIAM Journal on Optimization*.

Becker, S., Bobin, J., and Candès, E. J. (2011). Nesta: a fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*.

Ben-Tal, A. and Teboulle, M. (1989). A smoothing technique for nondifferentiable optimization problems. In *Optimization*, pages 1–11. Springer.

Bertsekas, D. (1973). Stochastic optimization problems with nondifferentiable cost functionals. *Journal of Optimization Theory and Applications*, 12(2):218–231.

Bertsekas, D. (2004). *Nonlinear Programming (2nd ed.)*. Athena Scientific.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge Univ Pr.

Brodie, J., Daubechies, I., De Mol, C., Giannone, D., and Loris, I. (2009). Sparse and stable markowitz portfolios. *Proceedings of the National Academy of Sciences*, 106(30):12267–12272.

Bruck, R. (1977). On the week convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in hilbert space. *J. math. Anal. appl.*

Bubeck, S. (2014). Theory of convex optimization for machine learning. *arXiv preprint arXiv:1405.4980*.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Candès, E. J. and Plan, Y. (2010). Matrix completion with noise. *Proceedings of the IEEE*.

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational mathematics*.

Chandrasekaran, V., Recht, B., Parrilo, P. A., and Willsky, A. S. (2012). The convex geometry of linear inverse problems. *FOCM*, 12(6):805–849.

Combettes, P. and Pesquet, J. (2011). Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer.

Combettes, P. and Wajs, V. (2005). Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200.

Cox, B., Juditsky, A., and Nemirovski, A. (2014). Dual subgradient algorithms for large-scale nonsmooth learning problems. *Mathematical Programming*.

Cullum, J., Willoughby, R., and Lake, M. (1983). A lanczos algorithm for computing singular values and vectors of large matrices. *SIAM Journal on Scientific and Statistical Computing*.

Demyanov, V. and Rubinov, A. (1970). *Approximate methods in optimization problems*, volume 32. Elsevier Publishing Company.

Deng, J., Berg, A. C., Li, K., and Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? In *Computer Vision*. ECCV 2010.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Devolder, O., Glineur, F., and Nesterov, Y. (2014). First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM.

Duchi, J. C., Bartlett, P., and Wainwright, M. J. (2012). Randomized smoothing for stochastic optimization. *ArXiv*.

Dudik, M., Harchaoui, Z., and Malick, J. (2012). Lifted coordinate descent for learning with trace-norm regularization. *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval research logistics quarterly*.

Garber, D. and Hazan, E. (2013). A Linearly Convergent Conditional Gradient Algorithm with Applications to Online and Stochastic Optimization. *ArXiv e-prints*, 1301.4666.

Garber, D. and Hazan, E. (2015). Faster rates for the frank-wolfe method over strongly-convex sets. In *Proceedings of the 32nd International Conference on Machine Learning*.

Grandvalet, Y., Rakotomamonjy, A., Keshet, J., and Canu, S. (2009). Support vector machines with a reject option. In *Advances in neural information processing systems*, pages 537–544.

Guzman, C. and Nemirovski, A. (2013). On Lower Complexity Bounds for Large-Scale Smooth Convex Optimization. *ArXiv e-prints*, 1307.5001.

Harchaoui, Z., Douze, M., Paulin, M., Dudik, M., and Malick, J. (2012a). Large-scale image classification with trace-norm regularization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*.

Harchaoui, Z., Juditsky, A., and Nemirovski, A. (2012b). Conditional gradient algorithms for machine learning. In *NIPS Workshop on Optimization for ML*.

Harchaoui, Z., Juditsky, A., and Nemirovski, A. (2014). Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming, Series A*, pages 1–30.

Hastie, T., Tibshirani, R., and Friedman, J. (2008). *The Elements of Statistical Learning (2nd Ed.)*. Springer Series in Statistics. Springer.

Hazan, E. (2008). Sparse approximate solutions to semidefinite programs. In *LATIN 2008: Theoretical Informatics*, Lecture Notes in Computer Science, pages 306–316. Springer Berlin Heidelberg.

Hazan, E. and Kale, S. (2012). Projection-free online learning. In *ICML*.

Hiriart-Urruty, J. and Lemarechal, C. (1993). *Fundamentals of Convex Analysis*. Springer Verlag.

Hiriart-Urruty, J. and Lemarechal, C. (1996). *Convex Analysis and Minimization Algorithms II (Grundlehren Der Mathematischen Wissenschaften)*. Springer Verlag.

Huber, P. (1981). *Robust statistics*. Wiley Series in Probability and Mathematical Statistics. J. Wiley.

Hummel, H. G., Van Den Berg, B., Berlanga, A. J., Drachsler, H., Janssen, J., Nadolski, R., and Koper, R. (2007). Combining social-based and information-based approaches for personalised recommendation on sequencing learning activities. *International Journal of Learning Technology*.

Jacob, L., Obozinski, G., and Vert, J. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM.

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, pages 427–435.

Jaggi, M. and Sulovský, M. (2010). A Simple Algorithm for Nuclear Norm Regularized Problems. *ICML 2010: Proceedings of the 27th international conference on Machine learning*.

Juditsky, A. and Nemirovski, A. (2010). First order methods for nonsmooth convex large-scale optimization. *Optimization for Machine Learning,(Sra, Nowozin, Wright, Eds), MIT Press, 2012*.

Kim, G., Xing, E. P., Fei-Fei, L., and Kanade, T. (2011). Distributed cosegmentation via submodular optimization on anisotropic diffusion. In *2011 International Conference on Computer Vision*, pages 169–176. IEEE.

Kuczynski, J. and Wozniakowski, H. (1992). Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications*.

Lacoste-Julien, S. and Jaggi, M. (2015). On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*.

Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. (2013). Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *ICML 2013*.

Lacoste-Julien, S., Lindsten, F., and Bach, F. R. (2015). Sequential kernel herding: Frank-wolfe optimization for particle filtering. In *AISTATS*.

Lafond, J., Klopp, O., Moulines, E., and Salmon, J. (2014). Probabilistic low-rank matrix completion on finite alphabets. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc.

Lan, G. (2013). The Complexity of Large-scale Convex Programming under a Linear Optimization Oracle. *ArXiv e-prints*, 1302.2325.

Lanczos, C. (1961). *Linear differential operators*. SIAM.

Lewis, A. (1999). Nonsmooth analysis of eigenvalues. *Mathematical Programming*, 84(1):1–24.

Mairal, J. (2013). Optimization with first-order surrogate functions. In *Proceedings of The 30th International Conference on Machine Learning*, pages 783–791.

Mallat, S. (2009). *A wavelet tour of signal processing: the sparse way*. Academic press, 3rd edition.

Markowitz, H. (1952). Portfolio selection. *The journal of finance*.

Mattingley, J. and Boyd, S. (2009). Automatic code generation for real-time convex optimization. *Convex optimization in signal processing and communications*.

Mattingley, J. and Boyd, S. (2010). Real-time convex optimization in signal processing. *IEEE Signal processing magazine*.

Miller, B., Albert, I., Lam, S. K., Konstan, J., and Riedl, J. (2003). Movielens unplugged: Experiences with a recommender system on four mobile devices. In *ACM SIGCHI Conference on Human Factors in Computing Systems*.

Moreau, J. (1965). Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93:273–299.

Nemirovski, A. and Yudin, D. (1983). Information-based complexity of mathematical programming. *Izvestia AN SSSR, Ser. Tekhnicheskaya Kibernetika (the journal is translated to English as Engineering Cybernetics. Soviet J. Computer & Systems Sci.)*, 1.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376.

Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*. Springer.

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, 103.

Nesterov, Y. (2007a). Gradient methods for minimizing composite objective function. *CORE DISUSSION PAPER 2007/76*.

Nesterov, Y. (2007b). Smoothing technique and its applications in semidefinite optimization. *Math. Program.*, 110(2):245–259.

Nesterov, Y. (2013). Gradient methods for minimizing composite functions. *Mathematical Programming*.

Orabona, F., Argyriou, A., and Srebro, N. (2012). Prisma: Proximal iterative smoothing algorithm.

Passty, G. (1979). Ergodic convergence to a zero of the sum of monotone operatorsin hilbert space. *J. Math. Anal. Appl*.

Pierucci, F., Harchaoui, Z., and Malick, J. (2014). A smoothing approach for composite conditional gradient with nonsmooth loss. *CAP Conference d'Apprentissage Automatique*.

Rockafellar, R. T. (1970). Convex analysis (princeton mathematical series). *Princeton University Press*.

Rogez, G., Khademi, M., Supančič III, J., Montiel, J., and Ramanan, D. (2014). 3d hand pose detection in egocentric rgb-d images. In *Computer Vision-ECCV 2014 Workshops*. Springer.

Rother, C., Minka, T., Blake, A., and Kolmogorov, V. (2006). Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE.

Schmidt, M., Roux, N. L., and Bach, F. R. (2011). Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*.

Shalev-Shwartz, S., Gonen, A., and Shamir, O. (2011). Large-Scale Convex Minimization with a Low-Rank Constraint. In *ICML*.

Srebro, N., Rennie, J., and Jaakkola, T. S. (2004). Maximum-margin matrix factorization. In *Advances in neural information processing systems*. NIPS.

Temlyakov, V. (2012). Greedy approximation in convex optimization. *Constructive Approximation*.

Tomioka, R. and Suzuki, T. (2013). Convex tensor decomposition via structured schatten norm regularization. In *Advances in neural information processing systems*.

Usunier, N., Buffoni, D., and Gallinari, P. (2009). Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML 2009.

Vapnik, V. (2005). *The nature of statistical learning theory*. Springer Science & Business Media.

Vapnik, V. N. and Chervonenkis, A. J. (1974). Theory of pattern recognition.

Vicente, S., Rother, C., and Kolmogorov, V. (2011). Object cosegmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE.

Weimer, M., Karatzoglou, A., Le, Q. V., and Smola, A. J. (2007). Cofi rank - maximum margin matrix factorization for collaborative ranking. In *NIPS*.

Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *Systems, Man and Cybernetics, IEEE Transactions on*.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zhang, X., Yu, Y., and Schuurmans, D. (2012). Accelerated training for matrix-norm regularization: A boosting approach. In *NIPS*.

Zhao, P., Rocha, G., and Yu, B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497.

Appendix A

Useful results

A.1 Computing the top pair of singular vectors

The maximum singular value and the corresponding pair of singular vectors can be computed using the Lanczos algorithm (Algorithm 16) Kuczynski and Wozniakowski (1992).

We show that it is possible to find the maximum singular value and singular vectors of a matrix A by computing eigenvectors with Lanczos algorithm. The idea to build Algorithm 17 is from Lanczos (1961) andCullum et al. (1983).

Algorithm 16 Lanczos algorithm $\lambda_{\texttt{Lan}}(\mathbf{B}, \mathbf{b}, t)$

Input $\mathbf{B} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$, iterations number is t
$(\lambda_\star, \mathbf{z}_\star) = \lambda_{\texttt{Lan}}(\mathbf{B}, \mathbf{b}, t) := \max \left\{ \frac{\langle \mathbf{Bz}, \mathbf{z} \rangle}{\langle \mathbf{z}, \mathbf{z} \rangle} \middle| \mathbf{z} \in \text{span}\{\mathbf{b}, \mathbf{Bb}, \mathbf{B}^2\mathbf{b}, \ldots, \mathbf{B}^{t-1}\mathbf{b}\}, \mathbf{z} \neq 0 \right\}$ maximized at $\mathbf{z}_\star$
output $\lambda_\star$, $\mathbf{z}_\star$

Algorithm 17 Lanczos algorithm - adaptation for singular values $\sigma_{\texttt{LS}}(\mathbf{B}, \mathbf{b}, t)$

Input $\mathbf{A} \in \mathbb{R}^{d \times k}$, $\mathbf{b} \in \mathbb{R}^{d+k}$, iterations number is t
$\mathbf{B} = \begin{pmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^\top & \mathbf{0} \end{pmatrix}$
$(\sigma_\star, \mathbf{z}) = \sigma_{\texttt{LS}}(\mathbf{A}, \mathbf{b}, t) := \lambda_{\texttt{Lan}}(\mathbf{B}, \mathbf{b}, t)$
$\mathbf{u}_\star = [z^1, \ldots, z^d]$
$\mathbf{v}_\star = [z^{d+1}, \ldots, z^{d+k}]$
output $\sigma_\star$, $\mathbf{u}_\star, \mathbf{v}_\star$

Let suppose that $\mathbf{A} \in \mathbb{R}^{d \times k}$ is decomposed with two orthonormal matrices $\mathbf{U} \in \mathbb{R}^{d \times r}$, $\mathbf{V} \in \mathbb{R}^{k \times r}$

and a diagonal matrix $\mathbf{S} \in \mathbb{R}^{r \times r}$ that contains on its diagonal the singular values s_i

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top. \tag{A.1}$$

We claim that the decomposition (A.1) implies

$$\mathbf{B} = \mathbf{Z}\Lambda\mathbf{Z}^\top, \tag{A.2}$$

where

$$\mathbf{B} = \begin{pmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^\top & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{n \times n},$$

with $n = d + k$,

$$\Lambda := \begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix},$$

$$\mathbf{Z} := \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{U} & \mathbf{U} \\ \mathbf{V} & -\mathbf{V} \end{pmatrix}.$$

The decomposition (A.2) means that an eigenvector base for $\mathbf{B}$ is $\mathbf{Z}$ and the eigenvalues of $\mathbf{B}$ are $s_1, \ldots, s_r, -s_1, \ldots, -s_r$. It is then possible to find an approximation of the largest singular value s_1 of $\mathbf{A}$ by looking for the largest eigenvalue $\Lambda^{1,1}$ of $\mathbf{B}$ with Algorithm 16, where $\mathbf{z}_\star$ is the first column of $\mathbf{Z}$, with Algorithm 17.

Proof. (Of (A.1) $\Longrightarrow$ (A.2)) With few calculus is evident that $\mathbf{Z}$ is orthonormal because $\mathbf{U}$ and $\mathbf{V}$ are orthonormal (just assert $\mathbf{Z}^\top\mathbf{Z} = \mathbf{Z}\mathbf{Z}^\top = \mathbf{I}_n$). We compute

$$\mathbf{Z}\Lambda = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{U} & \mathbf{U} \\ \mathbf{V} & -\mathbf{V} \end{pmatrix} \begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{U}\mathbf{S} & -\mathbf{U}\mathbf{S} \\ \mathbf{V}\mathbf{S} & \mathbf{V}\mathbf{S} \end{pmatrix};$$

$$\mathbf{B}\mathbf{Z} = \begin{pmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^\top & \mathbf{0} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{U} & \mathbf{U} \\ \mathbf{V} & -\mathbf{V} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{A}\mathbf{V} & -\mathbf{A}\mathbf{V} \\ \mathbf{A}^\top\mathbf{U} & \mathbf{A}^\top\mathbf{U} \end{pmatrix}.$$

Then

$$\mathbf{Z}\Lambda = \mathbf{B}\mathbf{Z} \iff \begin{cases} \mathbf{U}\mathbf{S} = \mathbf{A}\mathbf{V} \\ \mathbf{V}\mathbf{S} = \mathbf{A}^\top\mathbf{U}. \end{cases}$$

The equation on the left implies (A.2) and the one on the right is equivalent to to (A.1). $\square$

We define the relative error

$$E = \frac{|\sigma_{\mathtt{LS}}(\mathbf{A}, \mathbf{b}, t) - s_1(\mathbf{A})|}{s_1(\mathbf{A})}.$$

Algorithm 16 does not converge when $\mathbf{b}$ is orthogonal to the maximum eigenvector of $\mathbf{B}$ Kuczynski and Wozniakowski (1992), i.e it is possible that $E > \varepsilon$ even with a lot of iterations, but this happens only when $\mathbf{b}$ is on a subset of E of (Lebesgue) measure zero. We are going to define another type of error that considers $\mathbf{b}$ a random variable with uniform distribution on the unit sphere. We define the average relative error

$$E_{\mathtt{av}}(\sigma, \mathbf{A}, t) := \int_{\|\mathbf{b}\|=1} \left| \frac{\sigma(\mathbf{A}, \mathbf{b}, t) - s_1(\mathbf{A})}{s_1(\mathbf{A})} \right| \mu(d\mathbf{b}),$$

where μ is the Lebesgue measure and σ is Algorithm 16 or 17.

Theorem A.1.1. *Let $\sigma_{\mathtt{LS}}(\mathbf{A}, \mathbf{b}, t)$ obtained using the extensionded Algorithm 17. Then for any $\mathbf{A} \in \mathbb{R}^{d \times k}$*

$$\begin{aligned} E_{\mathtt{av}}(\sigma_{\mathtt{LS}}, \mathbf{A}, t) &= 0 && t \geq 2m; \\ E_{\mathtt{av}}(\sigma_{\mathtt{LS}}, \mathbf{A}, t) &\leq O\left(\frac{\log(d+k)}{t-1}\right) && t \in [4, 2m-1]. \end{aligned}$$

where m is the number of distinct singular values of $\mathbf{A}$. In addition the complexity in space is $O(dk)$, for a general case, and can shrink to $O((d+k)t)$ if $\mathbf{A}$ is sparse.

Proof. (Of Theorem A.1.1)

Power algorithm has accuracy $O(\log(n)/(t-1))$. Lanczos algorithm has accuracy upper bound $O((\log(n)/(t-1))^2)$ for $t \in [4, n-1]$, and the relative error is zero when $t > n$. For large sale problems the exact solution is normally not attained because the number of iterations is much smaller than the dimension, which implies that conditional gradient algorithms have cheaper iterations than proximal methods, when the regularizer is the nuclear norm.

In addition we notice that to compute only the biggest singular vectors, or just an approximation of them, is much more cheap than find the whole singular value decomposition.

The complexity in space, i.e the memory needed to run the algorithm, is $O(dk)$. In fact $\mathbf{B}$ needs a memory of $2dk$, $\mathbf{b}$ and z need $d+k$. To find the base $\{\mathbf{b}, \mathbf{B}\mathbf{b}, \mathbf{B}^2\mathbf{b}, \ldots, \mathbf{B}^{t-1}\mathbf{b}\}$ just start from $\mathbf{b}$ and then multiply each time by $\mathbf{B}$ to obtain the next vector. Then the subspace needs just $(d+k) \cdot t$, where the iteration number t is much smaller than $d+k$. Then we get $O(dk)$. Nevertheless it could be possible to have $O((d+k) \cdot t)$, which cannot be further improved, if for some reason $\mathbf{B}$ is sparse. □

Proof. (Of equation (1.24)) We need to solve the problem

$$\max_{\|\mathbf{x}\|_{\sigma,1}\leq 1} \langle \mathbf{y}, \mathbf{x} \rangle.$$

The maximum singular value and related vectors of the matrix $\mathbf{y}$ are

$$(\mathbf{u}_1, s_1, \mathbf{v}_1) := \text{svd}_{max}(\mathbf{y}).$$

First we show that $\langle \mathbf{y}, \mathbf{x} \rangle \leq s_1$. Any $\mathbf{x} \in \mathcal{Q}$ can be written as $\mathbf{x} = \sum_{h=1}^{r} c_h \bar{\mathbf{u}}_h \bar{\mathbf{v}}_h^\top$ with $c_h \geq 0$ and $\sum_{h=1}^{r} c_h \leq 1$.

$$\begin{aligned}
\langle \mathbf{y}, \mathbf{x} \rangle &= \sum_i \sum_j \mathbf{y}^{ij} \mathbf{x}^{ij} \\
&= \sum_i \sum_j \mathbf{y}^{ij} \sum_h c_h \bar{\mathbf{u}}_h^i \bar{\mathbf{v}}_h^j \\
&= \sum_h c_h \sum_i \sum_j \bar{\mathbf{u}}_h^i \mathbf{y}^{ij} \bar{\mathbf{v}}_h^j \\
&= \sum_h c_h \bar{\mathbf{u}}_h \mathbf{y} \bar{\mathbf{v}}_h^\top \\
&\leq \sum_h c_h s_1 \|\bar{\mathbf{u}}_h\| \|\bar{\mathbf{v}}_h\| \\
&= s_1 \sum_h c_h \\
&\leq s_1
\end{aligned}$$

We used that a bilinear form $\mathbf{x}\mathbf{A}\mathbf{y}^\top$ is not larger than the maximum singular value of $\mathbf{A}$, with $\mathbf{x}$ and $\mathbf{y}$ of norm 1.

Now we show that the maximum singular value is attained at $\mathbf{x}_\star = \mathbf{u}_1 \mathbf{v}_1^\top$. The singular values decomposition is $\mathbf{y} = \sum_{k=1}^{r} s_k \mathbf{u}_k \mathbf{v}_k^\top$, with

$$\begin{aligned}
\langle \mathbf{y}, \mathbf{x}_\star \rangle &= \langle \mathbf{y}, \mathbf{u}_1 \mathbf{v}_1^\top \rangle \\
&= \sum_i \sum_j \mathbf{y}^{ij} \mathbf{u}_1^i \mathbf{v}_1^j \\
&= \sum_i \sum_j \left(\sum_k s_k \mathbf{u}_k^i \mathbf{v}_k^j \right) \mathbf{u}_1^i \mathbf{v}_1^j
\end{aligned}$$

$$
\begin{aligned}
&= \sum_k s_k \sum_i \sum_j \mathbf{u}_k^i \mathbf{v}_k^j \mathbf{u}_1^i \mathbf{v}_1^j \\
&= \sum_k s_k \sum_i \sum_j \mathbf{u}_k^i \mathbf{u}_1^i \mathbf{v}_k^j \mathbf{v}_1^j \\
&= \sum_k s_k \mathbf{u}_k^\top \mathbf{u}_1 \mathbf{v}_k^\top \mathbf{v}_1 \\
&= \sum_k s_k \delta_{k1} \delta_{k1} \\
&= s_1.
\end{aligned}
$$

The last part is because $\{\mathbf{u}_k\}$ and $\{\mathbf{v}_k\}$ are two orthonormal bases. □

Proof. (Of equation (1.23)) The proof follows the same scheme used to prove (1.24). We need to solve the problem

$$
\max_{\mathbf{x} \succeq \mathbf{0},\, \mathrm{tr}(\mathbf{x})=1} \langle \mathbf{y}, \mathbf{x} \rangle.
$$

The maximum eigenvalue and related vector of the matrix $\mathbf{y}$ are $(\lambda_1, \mathbf{v}_1) := \mathrm{eigs}_{max}(\mathbf{y})$.

First we show that $\langle \mathbf{y}, \mathbf{x} \rangle \leq \lambda_1$. Any $\mathbf{x} \in \mathcal{Q}$ can be written as $\mathbf{x} = \sum_{h=1}^r c_h \bar{\mathbf{v}}_h \bar{\mathbf{v}}_h^\top$ with $c_h \geq 0$ and $\sum_{h=1}^r c_h = 1$.

$$
\begin{aligned}
\langle \mathbf{y}, \mathbf{x} \rangle &= \sum_i \sum_j \mathbf{y}^{ij} \mathbf{x}^{ij} \\
&= \sum_i \sum_j \mathbf{y}^{ij} \sum_h c_h \bar{\mathbf{v}}_h^i \bar{\mathbf{v}}_h^j \\
&= \sum_h c_h \sum_i \sum_j \bar{\mathbf{v}}_h^i \mathbf{y}^{ij} \bar{\mathbf{v}}_h^j \\
&= \sum_h c_h \bar{v}_h \mathbf{y} \bar{\mathbf{v}}_h^\top \\
&\leq \sum_h c_h \lambda_1 \left\| \bar{\mathbf{v}}_h \right\| \left\| \bar{\mathbf{v}}_h \right\| \\
&= \lambda_1 \sum_h c_h \\
&= \lambda_1
\end{aligned}
$$

Now we show that the maximum λ is attained at $\mathbf{x}_\star = \mathbf{v}_1 \mathbf{v}_1^\top$. The decomposition into eigenvalues is $\mathbf{y} = \sum_{k=1}^r \lambda_k \mathbf{v}_k \mathbf{v}_k^\top$, with

$$
\langle \mathbf{y}, \mathbf{x}_\star \rangle = \langle \mathbf{y}, \mathbf{v}_1 \mathbf{v}_1^\top \rangle
$$

$$
\begin{aligned}
&= \sum_i \sum_j \mathbf{y}^{ij} \mathbf{v}_1^i \mathbf{v}_1^j \\
&= \sum_i \sum_j \left(\sum_k \lambda_k \mathbf{v}_k^i \mathbf{v}_k^j \right) \mathbf{v}_1^i \mathbf{v}_1^j \\
&= \sum_k \lambda_k \sum_i \sum_j \mathbf{v}_k^i \mathbf{v}_k^j \mathbf{v}_1^i \mathbf{v}_1^j \\
&= \sum_k \lambda_k \sum_i \sum_j \mathbf{v}_k^i \mathbf{v}_1^i \mathbf{v}_k^j \mathbf{v}_1^j \\
&= \sum_k \lambda_k \mathbf{v}_k^\top \mathbf{v}_1 \mathbf{v}_k^\top \mathbf{v}_1 \\
&= \sum_k \lambda_k \delta_{k1} \delta_{k1} \\
&= \lambda_1 .
\end{aligned}
$$

□

A.2 Projection on a norm-ball

A.2.1 Examples of norm-balls

Given a closed convex ball $\mathcal{Q} \subset E$, the problem of projecting onto $\mathcal{Q}$ a point $\mathbf{x} \in E$ is defined as

$$
\pi_{\mathcal{Q}}(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Q}} \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|^2 .
$$

We see several algorithms and their complexity for different choices of $\mathcal{Q}$

Euclidean ball

$$
\left\{ \mathbf{x} \in E \,\middle|\, \sum_{i=1}^n (\mathbf{x}^i)^2 \leq 1 \right\} \tag{A.3}
$$

Then the projection on (A.3) is

$$
\pi_{\mathcal{Q}}(\mathbf{x}) = \begin{cases} \mathbf{x} & \|\mathbf{x}\| \leq 1 \\ \frac{\mathbf{x}}{\|\mathbf{x}\|} & \|\mathbf{x}\| > 1 \end{cases} \tag{A.4}
$$

The complexity is $O(n)$, where n is he dimension of E. This result is valid also when $\mathbf{x}$ is a matrix.

Proof. (Of equation (A.4) Let us change variable: $\mathbf{y} = r\mathbf{u}$ and $\mathbf{x} = t\mathbf{v}$, with $r, t \geq 0$, $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$. This notation implies that $\mathbf{v} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$. Let us notice that $\langle \mathbf{u}, \mathbf{v} \rangle$ is maximized when $\mathbf{u}_\star = \mathbf{v}$ and

$\max_{\|\mathbf{u}\|=1}\langle \mathbf{u}, \mathbf{v}\rangle = 1$.

$$\begin{aligned}
\min_{\mathbf{y}\in B} \|\mathbf{x}-\mathbf{y}\|^2 &= \min_{0\le r\le 1, \|\mathbf{u}\|=1} \|t\mathbf{v} - r\mathbf{u}\|^2 \\
&= \min_{0\le r\le 1, \|\mathbf{u}\|=1} \|t\mathbf{v}\|^2 - 2\langle r\mathbf{v}\mathbf{u}, t\mathbf{v}\rangle + \|r\mathbf{u}\|^2 \\
&= \min_{0\le r\le 1, \|\mathbf{u}\|=1} t^2 - 2rt\langle \mathbf{u}, \mathbf{v}\rangle + r^2 \\
&= \min_{0\le r\le 1} t^2 - 2rt + r^2 \\
&= \min_{0\le r\le 1} (t-r)^2.
\end{aligned}$$

Then the optimal is $r_\star = \begin{cases} r & t \le 1 \\ 1 & t > 1 \end{cases}$ and considering $\pi_{\mathcal{Q}}(\mathbf{x}) = \mathbf{y}_\star = r_\star \mathbf{u}_\star$ we conclude the proof. □

Positive simplex and $\|\cdot\|_1$ ball

$$\left\{ \mathbf{x} \in E \,\middle|\, \sum_{i=1}^n \mathbf{x}^i \le 1,\ \mathbf{x}^i \ge 0 \right\} \qquad \text{(Positive simplex)} \tag{A.5}$$

$$\left\{ \mathbf{x} \in E \,\middle|\, \sum_{i=1}^n \left|\mathbf{x}^i\right| \le 1 \right\} \qquad \text{(L1 ball)} \tag{A.6}$$

Efficient algorithms for the projection on (A.5) and (A.6) are described at Duchi et al. (2008). A Python implementation is at `https://gist.github.com/daien/1272551` As these algorithms involve the sorting of $\mathbf{x}$, their complexity is $O(n \log n)$.

Nuclear norm ball

$$\{\mathbf{x} \in E \mid \|\mathbf{x}\|_* \le 1\} \tag{A.7}$$

where $\|\mathbf{x}\|_*$ is the sum of singular values of $\mathbf{x}$.

A.2.2 Proximal operator of quadratic function

We suppose to know how to compute the proximal operator

$$p(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^n} \|\mathbf{y}\|^2 + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2$$

of the squared norm. In the next proposition we show how to compute the proximal operator

$$p_{\gamma\mathbf{Q}}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^n}{\operatorname{argmin}}\ \mathbf{y}^\top\mathbf{Q}\mathbf{y} + \frac{1}{2\gamma}\left\|\mathbf{x}-\mathbf{y}\right\|^2$$

of a quadratic function $\mathbf{y}^\top\mathbf{Q}\mathbf{y}$.

Proposition A.2.1. *Let $\mathbf{Q}\in\mathbb{R}^{n\times n}$ a positive definite matrix, $\gamma>0$. This implies that also $\frac{1}{3}(2\gamma\mathbf{Q}+\mathbf{I})$ is positive definite. We define $\mathbf{M}$ and Λ through the eigenvalue decomposition $\mathbf{M}^\top\Lambda\mathbf{M}=\frac{1}{3}(2\gamma\mathbf{Q}+\mathbf{I})$. Then*

$$p_{\gamma\mathbf{Q}}(\mathbf{x}) = \mathbf{A}p\left(\mathbf{A}\mathbf{x}\right) \tag{A.8}$$

where $\mathbf{A} := (\sqrt{\Lambda})^{-1}\mathbf{M}^\top$.

We verify that if $\mathbf{Q}$ is the identity and $\gamma=1$, then $\mathbf{A}=\mathbf{I}$ and the two operators coincide

$$p_{\mathbf{I}}(\mathbf{x}) = p(\mathbf{x}).$$

In fact $\frac{1}{3}(2\gamma\mathbf{Q}+\mathbf{I}) = \frac{1}{3}(2\mathbf{I}+\mathbf{I}) = \mathbf{I}$, $\mathbf{M}=\mathbf{I}$, $\Lambda=\mathbf{I}$.

Proof. (Of equation (A.8)) We make a constructive proof to find the transformation $\mathbf{A}$. We remind that $\mathbf{M}$ is an orthogonal operator, and $\mathbf{M}^{-1}=\mathbf{M}^\top$. The matrix Λ has diagonal elements λ^i, $\sqrt{\Lambda}$ has diagonal elements $\sqrt{\lambda^i}$, and $(\sqrt{\Lambda})^{-1}$ has diagonal elements $1/\sqrt{\lambda^i}$.

$$\begin{aligned}
p_{\gamma\mathbf{Q}}(\mathbf{x}) &= \underset{\mathbf{y}}{\operatorname{argmin}}\ \mathbf{y}^\top\mathbf{Q}\mathbf{y} + \frac{1}{2\gamma}\left\|\mathbf{x}-\mathbf{y}\right\|^2 \\
&= \underset{\mathbf{y}}{\operatorname{argmin}}\ 2\gamma\mathbf{y}^\top\mathbf{Q}\mathbf{y} + \left\|\mathbf{x}-\mathbf{y}\right\|^2 \\
&= \underset{\mathbf{y}}{\operatorname{argmin}}\ 2\gamma\mathbf{y}^\top\mathbf{Q}\mathbf{y} + \left\|\mathbf{y}\right\|^2 - 2\mathbf{y}^\top\mathbf{x} + \left\|\mathbf{x}\right\|^2 \\
&= \underset{\mathbf{y}}{\operatorname{argmin}}\ 2\gamma\mathbf{y}^\top\mathbf{Q}\mathbf{y} + y^\top\mathbf{I}\mathbf{y} - 2\mathbf{y}^\top\mathbf{x} \\
&= \underset{\mathbf{y}}{\operatorname{argmin}}\ \mathbf{y}^\top(2\gamma\mathbf{Q}+\mathbf{I})\mathbf{y} - 2\mathbf{y}^\top\mathbf{x} \\
&= \underset{\mathbf{y}}{\operatorname{argmin}}\ \mathbf{y}^\top(\tfrac{2}{3}(2\gamma\mathbf{Q}+\mathbf{I}))\mathbf{y} + \mathbf{y}^\top(\tfrac{1}{3}(2\gamma\mathbf{Q}+\mathbf{I}))\mathbf{y} - 2\mathbf{y}^\top\mathbf{x} \\
&= \underset{\mathbf{y}}{\operatorname{argmin}}\ 2\mathbf{y}^\top\mathbf{M}^\top\Lambda\mathbf{M}\mathbf{y} + 2\mathbf{y}^\top\mathbf{M}^\top\Lambda\mathbf{M}\mathbf{y} - 2\mathbf{y}^\top\mathbf{x} \\
&= \underset{\mathbf{y}}{\operatorname{argmin}}\ 2(\sqrt{\Lambda}\mathbf{M}\mathbf{y})^\top(\sqrt{\Lambda}\mathbf{M}\mathbf{y}) + \left\|(\sqrt{\Lambda}\mathbf{M}\mathbf{y})\right\|^2 - 2\mathbf{y}^\top\mathbf{x}.
\end{aligned}$$

We look for a quadratic expression of the right part. We need to find a and b to have

$$\left\|(\sqrt{\Lambda}\mathbf{M}\mathbf{y})\right\|^2 - 2\mathbf{y}^\top\mathbf{x} = \left\|\sqrt{\Lambda}\mathbf{M}\mathbf{y} + a\mathbf{x}\right\|^2 + b$$

where a is a constant and b may depend on $\mathbf{x}$. This is equivalent to

$$\left\|(\sqrt{\Lambda}\mathbf{M}\mathbf{y})\right\|^2 - 2\mathbf{y}^\top\mathbf{x} = \left\|(\sqrt{\Lambda}\mathbf{M}\mathbf{y})\right\|^2 + 2\langle\sqrt{\Lambda}\mathbf{M}\mathbf{y}, a\mathbf{x}\rangle + \|a\mathbf{x}\|^2 + b.$$

We need $b = -\|a\mathbf{x}\|^2$ and a must verify

$$-2\mathbf{y}^\top\mathbf{x} = 2\langle\sqrt{\Lambda}\mathbf{M}\mathbf{y}, a\mathbf{x}\rangle$$

for any $\mathbf{y} \in \mathbb{R}^n$. Then $a = (\sqrt{\Lambda})^{-1}\mathbf{M}$. We get

$$\begin{aligned} p_{\gamma\mathbf{Q}}(\mathbf{x}) &= \operatorname*{argmin}_{\mathbf{y}}\ 2(\sqrt{\Lambda}\mathbf{M}\mathbf{y})^\top(\sqrt{\Lambda}\mathbf{M}\mathbf{y}) + \left\|\sqrt{\Lambda}\mathbf{M}\mathbf{y} + a\mathbf{x}\right\|^2 + b \\ &= \operatorname*{argmin}_{\mathbf{y}}\ 2(\sqrt{\Lambda}\mathbf{M}\mathbf{y})^\top(\sqrt{\Lambda}\mathbf{M}\mathbf{y}) + \left\|\sqrt{\Lambda}\mathbf{M}\mathbf{y} + (\sqrt{\Lambda})^{-1}\mathbf{M}\mathbf{x}\right\|^2. \end{aligned}$$

Let us change variable with $\bar{\mathbf{y}} := \sqrt{\Lambda}\mathbf{M}\mathbf{y}$. The inverse is $\mathbf{y} = (\sqrt{\Lambda})^{-1}\mathbf{M}^\top\bar{\mathbf{y}} =: \mathbf{A}\mathbf{y}$. Then we substitute $\bar{\mathbf{y}}$ and obtain

$$\begin{aligned} p_{\gamma\mathbf{Q}}(\mathbf{x}) &= \mathbf{A}(\operatorname*{argmin}_{\bar{\mathbf{y}}}\ 2\bar{\mathbf{y}}^\top\bar{\mathbf{y}} + \|\bar{\mathbf{y}} + \mathbf{A}\mathbf{x}\|^2) \\ &= \mathbf{A}(\operatorname*{argmin}_{\bar{\mathbf{y}}}\ \|\bar{\mathbf{y}}\|^2 + \frac{1}{2}\|\bar{\mathbf{y}} + \mathbf{A}\mathbf{x}\|^2) \\ &= \mathbf{A}p(\mathbf{A}\mathbf{x}). \end{aligned}$$

□

A.3 Proofs of chapter 1

Proof. (Of equation (1.19)) It is straight forward from the definitions of prox and projection

$$\operatorname{prox}_{\gamma i_{\mathcal{Q}}}(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in E}\ i_{\mathcal{Q}}(\mathbf{y}) + \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{y}\|^2 = \operatorname*{argmin}_{\mathbf{y}\in\mathcal{Q}}\ \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{y}\|^2 = \pi_{\mathcal{Q}}(\mathbf{y})$$

□

Proof. (Of equation (1.20)) We separate the coordinates, which are independent and

$$\begin{aligned}\min_{\mathbf{y}\in E} \|\mathbf{y}\|_1 + \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{y}\|^2 &= \min_{\mathbf{y}} \sum_i |y^i| + \frac{1}{2\gamma}(x^i - y^i)^2 \\ &= \sum_i \min_{y^i} |y^i| + \frac{1}{2\gamma}(x^i - y^i)^2.\end{aligned}$$

Then, for all $i = 1 \dots n$,

$$\min_{y^i\in\mathbb{R}} |y^i| + \frac{1}{2\gamma}(x^i - y^i)^2 \tag{A.9}$$

$$= \min\left\{\min_{y^i<0} -y^i + \frac{1}{2\gamma}(x^i - y^i)^2,\ \frac{1}{2\gamma}(x^i)^2,\ \min_{y^i>0} y^i + \frac{1}{2\gamma}(x^i - y^i)^2\right\} \tag{A.10}$$

$$= \min\left\{|x^i + \gamma| + \frac{1}{2}\gamma,\ \frac{1}{2\gamma}(x^i)^2,\ |x^i - \gamma| + \frac{1}{2}\gamma\right\} \tag{A.11}$$

$$= \begin{cases} |x^i + \gamma| + \frac{1}{2\gamma} & x^i \le -\gamma \\ \frac{1}{2\gamma}(x^i)^2 & -\gamma < x^i \le \gamma \\ |x^i - \gamma| + \frac{1}{2\gamma} & \gamma < x^i \end{cases} \tag{A.12}$$

$$= \begin{cases} -x^i - \frac{1}{2\gamma} & x^i \le -\gamma \\ \frac{1}{2\gamma}(x^i)^2 & -\gamma < x^i \le \gamma \\ x^i - \frac{1}{2\gamma} & \gamma < x^i. \end{cases} \tag{A.13}$$

The points of maximum for the three components at equation (A.11) are

$$y^i_\star = \operatorname*{argmin}_{y^i\in\mathbb{R}} |y^i| + \frac{1}{2\gamma}(x^i - y^i)^2 = \begin{cases} x^i + \gamma & x^i \le -\gamma \\ 0 & -\gamma < x^i \le \gamma \\ x^i - \gamma & \gamma < x^i. \end{cases}$$

□

Proof. (Of equation (1.19))

$$\operatorname{prox}_{\gamma i_{\mathcal{Q}}}(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in E} i_{\mathcal{Q}}(\mathbf{y}) + \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{y}\|^2 \tag{A.14}$$

$$= \operatorname*{argmin}_{\mathbf{y}\in\mathcal{Q}} \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{y}\|^2 \tag{A.15}$$

$$= \pi_{\mathcal{Q}}(\mathbf{x}) \tag{A.16}$$

□

A.4 Computation of a support function without projection

In the next lemma we show that adding the zero on the ball we can compute the support function without the need to use the projection on the positive orthant.

Lemma A.4.1. *Let $\mathcal{Z}_f$ be a compact set in the positive orthant*

$$\mathbb{R}^{+^n} := \left\{ \mathbf{z} \in \mathbb{R}^n \,\middle|\, \mathbf{z}^i \geq 0 \right\}$$

and $\mathcal{Z}$ be the set of points between $\mathbf{0}$ and $\mathcal{Z}_f$, with respect to the lexicographic order:

$$\mathcal{Z} := \left\{ \xi = k \odot \mathbf{z} \,\middle|\, \mathbf{z} \in \mathcal{Z}_f,\, 0 \leq k^i \leq 1 \right\} = \bigcup_{\mathbf{z} \in \mathcal{Z}_f} \left\{ x \in \mathbb{R}^n \,\middle|\, \forall i \, 0 \leq \xi^i \leq \mathbf{z}^i \right\}.$$

Then

$$\max_{\mathbf{z} \in \mathcal{Z}_f} \langle \mathbf{z}, \max\{0, \xi\} \rangle = \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, \xi \rangle,$$

Proof. (*of lemma A.4.1*) Let the function P be the projection on $\mathbb{R}^{+^n}$:

$$P\xi := \max\{0, \xi\}$$

and P^o be the projection on the polar cone of $\mathbb{R}^{+^n}$:

$$P^o\xi = \min\{0, \xi\}.$$

To prove the lemma we show first $\max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, \xi \rangle \leq \max_{\mathbf{z} \in \mathcal{Z}_f} \langle \mathbf{z}, P\xi \rangle$ and then $\max_{\mathbf{z} \in \mathcal{Z}_f} \langle \mathbf{z}, P\xi \rangle \leq \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, \xi \rangle$. **First**:

$$\begin{aligned}
\max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, \xi \rangle &= \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, P\xi \rangle + \langle \mathbf{z}, P^o\xi \rangle \leq \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, P\xi \rangle + \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, P^o\xi \rangle \\
&= \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, P\xi \rangle + 0 = \max_{\mathbf{z} \in \mathcal{Z}_f, t \in [0,1]} \langle t\mathbf{z}, P\xi \rangle \\
&= \max_{t \in [0,1]} t \max_{\mathbf{z} \in \mathcal{Z}_f} \langle \mathbf{z}, P\xi \rangle = \max_{\mathbf{z} \in \mathcal{Z}_f} \langle \mathbf{z}, P\xi \rangle.
\end{aligned}$$

The second equality is because $\mathbf{z}$ and $P^o\xi$ belong to two polar cones. Then the scalar product is non

positive. But the max is 0 because $\mathbf{0} \in \mathcal{Z}$. **Second**: We have

$$\langle \mathbf{z}, P\xi \rangle = \sum_i \mathbf{z}^i \begin{cases} \xi^i, & \xi^i \leq 0 \\ 0, & \xi_i > 0 \end{cases} = \sum_i \mathbf{z}^i \xi^i \begin{cases} 1, & \xi^i \leq 0 \\ 0, & \xi^i > 0 \end{cases}$$

$$= \sum_i \xi^i \begin{cases} \mathbf{z}^i, & \xi^i \leq 0 \\ 0, & \xi^i > 0 \end{cases} = \langle Q_\xi \mathbf{z}, \xi \rangle,$$

where the (non surjective) function $Q_\xi : \mathcal{Z} \to \mathcal{Z}$ depends on ξ, which is a fixed parameter for all the proof. Then

$$\max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, P\xi \rangle = \max_{\mathbf{z} \in \mathcal{Z}} \langle Q_\xi \mathbf{z}, \xi \rangle \leq \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, \xi \rangle.$$

We have the inclusion $\mathcal{Z}_f \subset \mathcal{Z}$, then

$$\max_{\mathbf{z} \in \mathcal{Z}_f} \langle \mathbf{z}, P\mathbf{x} \rangle \leq \max_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z}, P\xi \rangle$$

and we conclude the second part of the proof. □

é