HAL
open science

Coping with the Computational and Statistical Bipolar Nature of Machine Learning

Pierre Machart

To cite this version:

Pierre Machart. Coping with the Computational and Statistical Bipolar Nature of Machine Learning. Machine Learning [cs.LG]. Aix-Marseille Université, 2012. English. NNT : . tel-00771718

HAL Id: tel-00771718

https://theses.hal.science/tel-00771718v1

Submitted on 9 Jan 2013

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.

UNIVERSITÉ D'AIX-MARSEILLE

ÉCOLE DOCTORALE MATHÉMATIQUES ET INFORMATIQUE (ED 184)

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ D'AIX-MARSEILLE

Coping with the Computational and Statistical Bipolar Nature of Machine Learning

Pierre MACHART

Thèse effectuée au LIF sous la direction de Liva Ralaivola et au LSIS sous la direction d'Hervé Glotin

Abstract

Machine Learning is known to have its roots in a broad spectrum of fields including Artificial Intelligence, Pattern Recognition, Statistics or Optimisation. From the earliest stages of Machine Learning, both *computational* issues and *generalisation* properties have been identified as central to the field. While the former address the question of computability, complexity (from a fundamental perspective) or computational efficiency (on a more practical standpoint) of learning systems, the latter aim at understanding and characterising how well the solutions they provide perform on new, unseen data.

Those last years, the emergence of *large-scale* datasets in Machine Learning has been deeply reshaping the principles of Learning Theory. Taking into account possible constraints on the training time, one has to deal with more complex trade-offs than the ones classically addressed by Statistics. As a direct consequence, designing new efficient algorithms (both in theory and practice), able to handle large-scale datasets, imposes to jointly deal with the statistical and computational aspects of Learning.

The present thesis aims at unravelling, analysing and exploiting some of the connections that naturally exist between the statistical and computational aspects of Learning. More precisely, in a first part, we extend the stability analysis, which relates some algorithmic properties to the generalisation abilities of learning algorithms, to a novel (and fine-grain) performance measure, namely the confusion matrix. In a second part, we present a novel approach to learn a kernel-based regression function, that serves the learning task at hand and exploits the structure of the problem so that the optimisation procedure is made inexpensive. Finally, we investigate the trade-off between convergence rate and computational cost when minimising a composite functional with inexact proximal-gradient methods. In that setting, we identify optimisation strategies that provably are computationally optimal.

Résumé

L'Apprentissage Automatique tire ses racines d'un large champ disciplinaire qui inclut l'Intelligence Artificielle, la Reconnaissance de Formes, les Statistiques ou l'Optimisation. Dès les origines de l'Apprentissage, les questions computationelles et les propriétés en généralisation ont toutes deux été identifiées comme centrales pour la discipline. Tandis que les premières concernent les questions de calculabilité ou de complexité (sur un plan fondamental) ou d'efficacité computationelle (d'un point de vue plus pratique) des systèmes d'apprentissage, les secondes visent à comprendre et caractériser comment les solutions qu'elles fournissent vont se comporter sur de nouvelles données non encore vues.

Ces dernières années, l'émergence de jeux de données à grande échelle en Apprentissage Automatique a profondément remanié les principes de la Théorie de l'Apprentissage. En prenant en compte de potentielles contraintes sur le temps d'entrainement, il faut faire face à un compromis plus complexe que ceux qui sont classiquement traités par les Statistiques. Une conséquence directe tient en ce que la mise en place d'algorithmes efficaces (autant en théorie qu'en pratique) capables de tourner sur des jeux de données à grande échelle doivent impérativement prendre en compte les aspects statistiques et computationels de l'Apprentissage de façon conjointe.

Cette thèse a pour but de mettre à jour, analyser et exploiter certaines des connections qui existent naturellement entre les aspects statistiques et computationels de l'Apprentissage. Plus précisément, dans une première partie, nous étendons l'analyse en stabilité, qui relie certaines propriétes algorithmiques aux capacités de généralisation des algorithmes d'apprentissage, à la matrice de confrusion, que nous suggérons comme nouvelle mesure de performance (fine). Dans une seconde partie, nous présentons un nouvelle approche pour apprendre une fonction de régression basée sur les noyaux, où le noyau appris sert directement la tâche de regression, et qui exploite la structure du problème pour offrir une procédure d'optimisation peu coûteuse. Finalement, nous étudions le compromis entre vitesse de convergence et coût computationel lorsque l'on minimise une fonction composite avec des méthodes par gradient-proximal inexact. Dans ce contexte, nous identifions des stratégies d'optimisation qui sont computationellement optimales.

Acknowledgements

Après trois années de thèse, j'ai furieusement envie de remercier la Terre entière, le Cosmos, les Dieux de la Pétanque (qui ne m'ont pourtant pas fait de cadeaux) et bien d'autres encore. Je vais donc m'efforcer de faire des remerciements aussi déstructurés et spontannés que possible, en n'oubliant bien sûr pas d'oublier une bonne poignée de gens pourtant méritants (désolé les mecs/meufs).

C'est bien parce que je tiens à ne pas mettre d'ordre particulier dans ces remerciements que je vais commencer par Liva. Je crois que je t'en dois bien une. Je n'oublierai pas mes errances passées et la Main ENTJique qui m'aura remis dans l'unique droit chemin : celui de la science qui ne marche pas mais qui a la classe. Je n'oublierai pas non plus les discussions d'avion, les rendez-vous boulot au milieu de la nuit et les moments où on a quand même fini par réussir à bosser un peu ensemble. Je sais que je peux compter sur l'équipe pour continuer à te recadrer après mon départ, mais je garderai quand même un oeil lointain mais bienveillant sur tout ça. Je remercie également Hervé, dont le soutien constant et inconditionnel aura été précieux. Je regretterai probablement de ne t'avoir jamais demandé de m'apprendre à construire des igloos mais en même temps, c'est pas à Rennes que ça m'aurait trop servi et la vie est encore longue. Je remercie bien évidemment Cécile, celle qui m'aura élevé avec amour avant le rapt ENTJique. Je crois que je n'oublierai jamais ton record du monde de cyclisme sur piste, même si ça n'était qu'un rêve. Il y a bien une dernière mère d'adoption à remercier ici. Sandrine, je sais ce que je te dois aussi. Tu n'auras pas vacillé, même dans les moments les plus terribles. Ces moments où on se rend compte qu'on a réinitialisé une variable à chaque itération. Ces moments où tu avais oublié ce qu'était le sommeil, mais où je venais quand même te faire part de mes découvertes, avec toute l'absence de structure qui me caractérise. Sois assurée que je continuerai à piller les stands en conférence, pour te ramener des stylos.

Les figures paternelles et maternelles étant expédiées, je vais pouvoir passer au reste de la famille QARMA, l'équipe de choc. Vous avez vraiment contribué à faire de cette thèse une expérience si riche. Comment ne pas aimer la vie avec autant d'intensité après avoir partagé autant de tartes Tatin et de Cabernet Boisé avec Valentin ? Après avoir découvert que contre toute attente, les méthodes spectrales, ça envoie du pâté, grâce à François ? Après avoir passé une semaine complète à se peler en Baie de Somme avec Stéphane ? Après avoir été touché par

la Grâce suite aux prêches enragés d'Hachem ? Après avoir grimpé dans les hauteurs de Grenade avec François-Xavier, resté incognito ?

L'ascenseur social arrive maintenant à la cave (quoi qu'en dise Valentin) : les doctorants (ou ex-). Quand je parle d'intensité, je crois que vous savez de quoi je parle. D'ailleurs, pour un certain nombre d'entre vous, je n'ai pas attendu d'avoir bouclé cette thèse pour vous remercier (et vous saurez que ça ne s'arrêtera pas là non plus), j'en garde donc sous le coude. Grâce à EmilieLG, j'aurai découvert des facettes magiques et inattendues de la garcitude. Avant de te rencontrer, je n'aurais jamais pu imaginer que les moqueries incessantes, les coiffures de madame et même de simples étirements puissent apporter la félicité. Raphaël, tu resteras un maître incontesté de l'irrévérence et donc une source d'inspiration inépuisable quand il s'agit de faire preuve de mauvaise foi. Un grand merci à toi, Guillaume, le roi de l'évènementiel marseillais. Les matchs au Vélodrome, les terrasses, les frites belges, la contrée, les films que moi seul aurait aimé au ciné... Je n'ose imaginer le nombre de choses à côté desquelles je serais passé sans toi. Merci à toi, Anaïk, l'amie du soir-espoir. Tu m'auras fait découvrir qu'on pouvait planer bien plus que moi, et que c'était cool ! Thomas, pendant 2 ans, tu auras partagé mon intimité. Malgré les ronflements, malgré les passantes et les passants, tu seras resté à mes côtés. Sois-en remercié comme il se doit. Merci aussi à la guerrière parmis les guerriers, EmilieM. Ta maîtrise de la gueule de bois aura permis si souvent de m'abreuver de cachets lorsque j'ai un peu trop déconné en conf'. On aura aussi partagé la confusion (et j'associe ici Sokol), et fait face ensemble à la terrible ingratitude des chefs. Merci de m'avoir accompagné dans ces épreuves. Je pense que Sylvain aura singulièrement réduit mon espérance de vie en me traînant à Sainte-Sophie, mais c'était pour la bonne cause. Je crois que je te dois aussi une partie de la patience de ma mère adoptive. Merci pour tout a ! Comment bien sûr ne pas remercier Juliette ? Contre mon gré, tu m'auras fait découvrir les contrées les plus bobinesques d'Aix-en-Provence, porter des gilets vert sapin, goûter des demi-coco. Personne n'aurait pu réussir un tel exploit et je t'en remercie. Pour continuer sur les chemins inattendus, qu'on a un peu peur d'explorer, je peux également remercier Harold. Je n'aurai pas rencontré Jacques personnellement, je ne me serai pas servi du téléphone arabe, mais leur présence aura été très inspirante. Mattias, vite arrivé, vite intégré. Si peu de temps, et tellement de petits douzièmes, de petits plats mijotés, de rires, de larmes. Nous en vivrons bien d'autres. Antoine, Ugo, nous aurons eu encore moins de temps. Pourtant, je sais que je peux compter sur vous. Il y aura bien d'autres châlets à partager.

Malgré nos efforts de conversion, le Couloir n'est pas peuplé que d'apprenticiens. Malgré vos défauts évidents, vous aurez définitivement pesé très positivement dans ma thèse. Mes sincères remerciements à Bruno, Marie-Christine, aux Frédéric, Thomas W., Benjamin, Sangnam, Sebastiano, Alexandre, Alain. Un merci spécial Clothilde pour les discussions sur la procrastination, le cinéma et le sens de la vie, à Claire sans qui cette th'ese n'aurait pas été bipolaire et à Pierre P. grâce à qui j'ai pu boire tant et tant d'excellents cafés.

Je n'oublie bien sûr pas celles qui sont un peu nos mamans à tous et qui m'auront chouchouté pendant 3 ans. Martine, Sylvie, Nadine, vous aurez été magiques ! Et pourtant, mes dossiers de mission pourraient à elles seules remplir une étagère entière. Merci infiniement pour votre gentillesse, votre réactivité, votre humour et les soirées à Porquerolles (qui resteront à Porquerolles).

And now, some love for the foreigners (or almost) ! This PhD was filled with amazing meetings. Those meetings have drastically undermined my productivity during poster sessions but also have fed an unbounded enthusiasm for the "scientific" conferences. Special thanks to Luca, with whom it has been a real pleasure to collaborate and party. A collective (you guys know exactly whom I'm thanking here) big up for the Stéphanois crew, the Quebecquois freaks, the Greek mob, the Israeli crowd, those who get naked in pools (but not in our hot tub!), the Lillois pack and many others I've met at conferences, summer schools or who knows where.

I wanted to keep those thanks "professional" but I really want to thank Michela too! What a crazy meeting. You have followed the whole PhD very closely and supported me all the way. More than any other, you helped me become the happy person I am now. I think you know how grateful I am for that. I just wanted to make it even more clear!

Quand les remerciements deviennent plus longs que les chapitres de thèse, c'est probablement qu'il est temps d'arrêter. Pour autant, soyez tous assurés que cette partie de ma thèse a pour moi un sens beaucoup plus fort que n'importe quelle autre.

"Most of my life I went to parties and heard a little groan when people heard what I did, now they're all excited to meet me."

Robert Tibshirani

Contents

1 Introduction **1**

1.1 A (Partial) History of Principles in Machine Learning 1

1.2 Retracing History through a Success Story: Support Vector Machines 5

1.3 Outline of the thesis 11

2 Optimisation for Machine Learning: Basic Definitions, Methods and Properties **13**

2.1 Convexity 13

2.2 First-Order Methods 15

2.3 Rates of Convergence 16

2.4 A word on non-smooth convex optimisation 18

3 Confusion Matrix Stability Bounds for Multiclass Classification **21**

3.1 Introduction 22

3.2 Confusion Loss 23

3.2.1 Notation 23

3.2.2 Confusion Matrix versus Misclassification Rate 24

3.3 Deriving Stability Bounds on the Confusion Matrix 27

3.3.1 Stability 27

3.3.2 Noncommutative McDiarmid's Bounded Difference Inequality 28

3.3.3 Stability Bound 29

3.4 Analysis of existing algorithms 31

3.4.1 Hilbert Space Regularised Algorithms 31

3.4.2 Lee, Lin and Wahba model 32

3.4.3 Weston and Watkins model 33

3.5 Discussion and Conclusion 34

4 Stochastic Low-Rank Kernel Learning for Regression **37**

4.1 Introduction 38

4.2 Proposed Model . . . 39
4.2.1 Data-parameterised Kernels . . . 39
4.2.2 Kernel Ridge Regression . . . 40
4.2.3 A Convex Optimisation Problem . . . 41
4.3 Solving the problem . . . 43
4.3.1 A Second-Order Stochastic Coordinate Descent . . . 43
4.3.2 Iterative Updates . . . 45
4.4 Analysis . . . 47
4.4.1 Pivotal Hyperparameter $\lambda\nu$ . . . 48
4.4.2 Runtime Complexity and Memory Usage . . . 50
4.4.3 Related Work . . . 50
4.5 Numerical Simulations . . . 52
4.5.1 Setup . . . 52
4.5.2 Influence of the parameters . . . 53
4.5.3 Comparison to other methods . . . 54
4.5.4 Larger-scale dataset . . . 55
4.6 Conclusion and future work . . . 56

5 Optimal Computational Trade-Off of Inexact Proximal Methods 59
5.1 Introduction . . . 60
5.2 Setting . . . 63
5.2.1 Inexact Proximal Methods . . . 63
5.2.2 Convergence Rates . . . 64
5.2.3 Approximation Trade-off . . . 65
5.3 Defining the Problem . . . 65
5.3.1 The Computational Cost of Inexact Proximal Methods . . . 66
5.3.2 Parameterising the Error . . . 67
5.3.3 Parameterised Bounds . . . 67
5.3.4 Towards a Computationally Optimal Tradeoff . . . 68
5.4 Results . . . 68
5.4.1 Optimal Strategies . . . 68

5.4.2 Comments and Interpretation of the Results 71
5.4.3 On the Usability of the Optimal Strategies 73
5.5 Numerical Simulations 75
5.5.1 TV-regularisation for image deblurring 75
5.5.2 Graph prediction 76
5.5.3 Why the "computationally optimal" strategies are good but not that optimal 77
5.6 Conclusion and future work 80

6 Conclusion **81**
6.1 Summary of Contributions 81
6.2 Broader Prospects 83

7 Appendices **87**
7.1 Proof of Theorem 5 87
7.2 Matrix Inversion Formulas 91
7.3 Proof of Proposition 7 91
7.4 Proof of Proposition 8 95
7.5 Proof of Proposition 9 96
7.6 Proof of Proposition 10 99

Bibliography **103**

Chapter 1

Introduction

Contents

1.1 A (Partial) History of Principles in Machine Learning 1

1.2 Retracing History through a Success Story: Support Vector Machines 5

1.3 Outline of the thesis 11

1.1 A (Partial) History of Principles in Machine Learning

Machine Learning is known to have its roots in a broad spectrum of fields including Artificial Intelligence, Pattern Recognition,Statistics or Optimisation. From the earliest stages of Machine Learning, both *computational* issues and *generalisation* properties have been identified as central to the field. While the former address the question of computability, complexity (from a fundamental perspective) or computational efficiency (on a more practical standpoint) of learning systems, the latter aim at understanding and characterising how well the solutions they provide perform on new, unseen data. But a lack of formalism and of frameworks, generalising the different existing methods, has long made it hard to study, beyond the mere acknowledgement of their paramount importance.

However, empirical observations (e.g. [Clark and Farley, 1955]) have highlighted the relevance of a statistical view of *predicting from data*. In the supervised setting (which the present thesis will focus on), considering observed data (input-output pairs) as realisations of a couple of random variables

$$(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y},$$

with an underlying (yet unknown) distribution D, an abundant literature in statistics ([Vapnik, 1995] and the references within), especially from the field of *empirical process theory* and *stochastic approximation* ([Boucheron et al., 2005] and the references within) have allowed to

quantitatively characterise the *generalisation abilities* of learning systems, that is to relate the empirical behaviour of learning systems (i.e. how they perform on the data used for learning) to their expected behaviour (i.e. how they generalise to unseen data).

More precisely, given some *loss* functional

$$\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$$

measuring how a prediction $h(\boldsymbol{x})$, with $h \in \mathcal{H}$, fits the actual target y, and a training set $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, this connection with Statistics made it possible to understand, for instance, under what conditions, and at what rates, the empirical risk

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, \boldsymbol{x}_i, y_i),$$

converges to its expectation, the true risk

$$R(h) = \mathbb{E}_D \ell(h, \boldsymbol{x}, y),$$

when the number of examples n grows. When the *consistency* property

$$\lim_{n \to \infty} R_n(h) - R(h) = 0$$

is (almost surely) ensured, one can safely assume that the learnt predictors will have good generalisation properties and will perform nearly as well on unseen data as they do on the training data, as long as one provides enough data in the learning phase.

However, these results, long known in Statistics, mainly focused on the asymptotics of the convergence of means to their expectations.As a consequence, they gave only little information on how to successfully learn, when the amount of data is finite. Addressing this weakness, the *Vapnik-Chervonenkis* (VC) theory [Vapnik, 1982] has been a huge breakthrough in Machine Learning.

In a nutshell, one of the main results from the VC theory states that the generalisation abilities of a learning algorithm critically depend on the empirical risk and some measure of capacity (namely the *VC dimension*) of the class of hypotheses among which the algorithm searches. One direct consequence is the theoretical soundness of the so-called *Structural Risk Minimisation* (SRM) principle (i.e. minimising the prediction errors made on a set of training examples while controlling the complexity of the solution). Those seminal results have been

crucial for Statistical Machine Learning as they led to great theoretical and practical results, and drew a lot of attention on the field.

Following this evolution, it became a natural trend to cast learning problems as optimisation problems such as:

$$\min_{h \in \mathcal{H}} R_n(h) + \lambda\, \Omega(h),$$

where $\lambda\, \Omega(h)$ is a *regularising* term aiming at controlling the complexity of h. Fortunately, a vast numerical optimisation literature already existed and boosted the development of learning procedures that were both theoretically-founded and computationally efficient. Even though adding a regularising can give rise to difficult problems, the large amount of already existing methods, have eventually made it possible to provably solve many learning problems with an arbitrary high precision.

If the advent of Statistics and Optimisation in Machine Learning share common historical roots, as we just depicted, their development and use, however, were disjoint. To get a very rough picture: Statistics points out which are the problems to solve, and Optimisation teaches how to solve them. For years, even though the success of learning algorithms equally depended on both aspects, they have been addressed and handled separately. Looking back on the rough picture: statisticians worked on problems designed to give satisfactory guarantees on the generalisation abilities, without regards to the way to solve them, while optimisers assessed their difficulty from a computational perspective and developed (when possible) algorithms to solve these problems.

However, over the last decade, the size of datasets in Machine Learning problems has grown critically. In Computer Vision or Natural Language Processing, for instance, the exponential growth of websites like Flickr, Youtube, Google books or online newspapers, to name a few, have allowed the access to masses of freely available pictures, videos or textual data. In Genomics, the development of inexpensive DNA chips have also had the collection of new samples reached unprecedented levels. With the extremely fast development of online social networks, claiming billions of users, new problems have risen for machine learners, implying the handling of huge-scale datasets.

This explosion of data has been deeply reshaping Machine Learning those last few years. The effects of this change of scale need not to be neglected or underestimated. The growth rate of the amount of data has been exceeding that of processing power to such an extent that it is hopeless

to only rely on faster optimisation methods to handle this tremendous amount of data at hand. Fortunately, while it becomes harder to accurately solve optimisation problems, the formerly described statistical estimation problem becomes less prominent.

Bottou and Bousquet [2007] has quantitatively explored this issue, by re-exploring the *excess error* $\mathcal{E}$ as the difference between the risk of the hypothesis learnt by a given learning algorithm, and that of the best possible predictor: Let

$$h^* := \operatorname*{argmin}_{h \in \mathcal{Y}^{\mathcal{X}}} R(h)$$

be the best predictor possible,

$$h^*_{\mathcal{H}} := \operatorname*{argmin}_{h \in \mathcal{H}} R(h)$$

be the best predictor in the hypothesis class $\mathcal{H}$ through which our learning algorithm searches,

$$h_n := \operatorname*{argmin}_{h \in \mathcal{H}} R_n(h)$$

be the predictor, in $\mathcal{H}$, minimising the empirical risk, and $\tilde{h}_n$ be the approximation of h_n that our learning algorithm finds. The *excess error* $\mathcal{E}$ can be decomposed as follows:

$$\begin{aligned}
\mathcal{E} :&= R(\tilde{h}_n) - R(h^*) \\
&= \big(R(h^*_{\mathcal{H}}) - R(h^*)\big) + \big(R(h_n) - R(h^*_{\mathcal{H}})\big) + \big(R(\tilde{h}_n) - R(h_n)\big) \\
&=: \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}},
\end{aligned}$$

where $\mathcal{E}_{\text{app}}$ is the *approximation error*, quantifying the excess error due to the choice of the hypothesis class $\mathcal{H}$, $\mathcal{E}_{\text{est}}$ is the *estimation error*, quantifying the excess error due to the minimisation of the empirical risk instead of the real risk and $\mathcal{E}_{\text{opt}}$ is the *optimisation error*, quantifying the excess error due to the imprecision solving the optimisation problem. The essential novelty in [Bottou and Bousquet, 2007] is to take into account this *optimisation error* $\mathcal{E}_{\text{opt}}$ and weigh its impact on the generalisation ability.

In the historical paradigm (called *small-scale* in Bottou and Bousquet [2007]), where the number of data is the bottleneck for generalisation, it is commonly admitted that the third term $\mathcal{E}_{\text{opt}}$ can be made arbitrarily small, i.e. it is possible to push the optimisation procedure to death. When neglected, the two first terms remain and describe the classical approximation-estimation trade-off that is addressed by the *Structural Risk Minimisation* principle described

earlier. However, when one tackles *large-scale* problems where constraints on the training time and memory usage may have to be considered, the optimisation error $\mathcal{E}_{\text{opt}}$ cannot be neglected any longer. For instance, when dealing with larger-scale problems, one could consider using only a fraction of the available data for training, in order to fall into the well-known *small-scale* paradigm. But then, being able to have $\mathcal{E}_{\text{opt}}$ virtually tend to zero comes at the cost of potentially having $\mathcal{E}_{\text{est}}$ to explode. In fact, [Bottou and Bousquet, 2007] suggests that one may make a better use of the available data by balancing the three types of excess error. To do so, one has to deal with a complex trade-off between those three terms.

Anyhow, this seminal analysis suggests that the emergence of *large-scale* datasets in Machine Learning is deeply reshaping the principles of Learning Theory. Taking into account possible constraints on the training time, one has to deal with more complex trade-offs than the ones classically addressed by Statistics. As a direct consequence, designing new efficient algorithms (both in theory and practice), able to handle large-scale datasets, imposes to jointly deal with the statistical and computational aspects of Learning.

1.2 Retracing History through a Success Story: Support Vector Machines

Illustrating the first part of this introduction, we will now revisit this history through a particular angle. Within the last 20 years, the *Support Vector Machines* (SVM) [Cortes and Vapnik, 1995] have become one of the most prominent symbols of Machine Learning and have drawn unprecedented attention from both users and researchers. Taking their roots in the VC theory, they have consistently been at the heart of the preoccupations of a large part of the machine community since then. As a consequence, the development of SVM has closely followed the different trends described earlier in this introduction.

In their primary form, Support Vector Machines aim at finding a linear function f for binary classification problems (i.e. $\mathcal{Y} = \{-1, +1\}$):

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}\,,\, \boldsymbol{x} \rangle + b,$$

where $\boldsymbol{w}$ is the normal of the separating hyperplane, b the bias term and whose sign is used to classify data. From this perspective, one may track some of the origins of SVM as far as the

end of the fifties with the *Perceptron* algorithm [Rosenblatt, 1958]. It consists in an iterative algorithm where, at each step, the current hypothesis is updated if it fails at predicting the class of a training example, so that the algorithm builds a classifier that gradually learns from his mistakes. Interestingly, [Novikoff, 1962] proves that if the training set is separable, that is if there exists a constant $\gamma > 0$ (called margin) such that:

$$\forall i \in [\![1, n]\!], f(\boldsymbol{x}_i)\, y_i \geq \gamma, \tag{1.1}$$

the perceptron algorithm converges in a finite number of iterations which depends on the margin γ (i.e. the larger the margin, the faster the convergence).

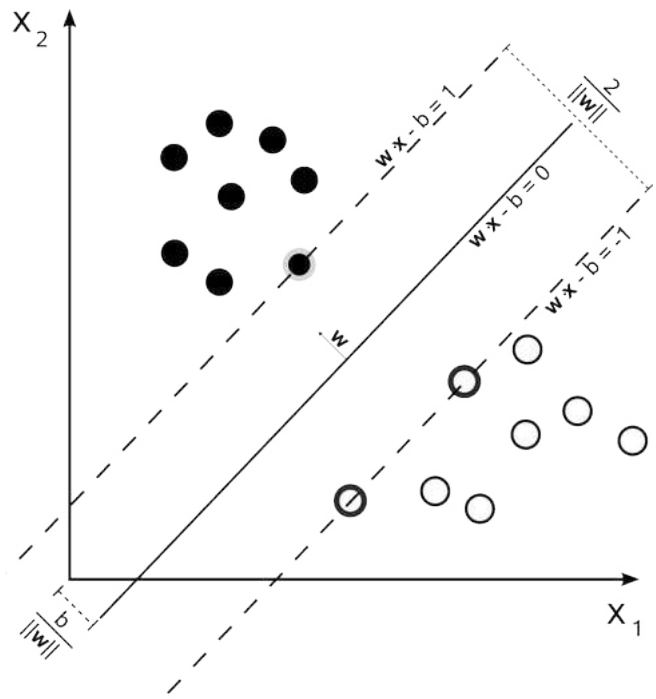

Figure 1.1: An example of a linear separator on a toy problem in $\mathbb{R}^2$

Perceptrons are known to have good generalisation abilities. This property is a direct consequence of the use of separating hyperplanes, which have a low VC dimension as long as the dimension of $\mathcal{X}$ is not too large. More formally, if the input space is such that $\mathcal{X} = \mathbb{R}^d$, the VC-dimension of the class of hyperplanes is known to be $d + 1$. This has long led machine learners to investigate the use of those hyperplanes, preferably in low-dimensional input spaces, in order to find predictors with good generalisation abilities.

Let us now consider the hypothesis class $\mathcal{H}_\gamma$ of separating hyperplanes *with a margin* γ (as defined in Eq. (1.1)). One may show that if you can bound the norm of the data points at hand by some constant R:

$$\forall i \in [\![1, n]\!], \|\boldsymbol{x}_i\| \leq R,$$

the VC-dimension of $\mathcal{H}_\gamma$ can be bounded as follows:

$$\mathrm{VC}(\mathcal{H}_\gamma) \leq \min\left(d, \left\lceil \frac{4R^2}{\gamma^2} \right\rceil\right) + 1.$$

This result has the following crucial consequence: when using linear separators with a bounded margin as predictors, the larger the margin is, the better the generalisation abilities will be ([Cristianini and Shawe-Taylor, 2000]). Using geometric arguments, Vapnik and Chervonenkis even get to a very counter-intuitive conclusion: mapping the data to a high-dimensional space can make it possible to find separating hyperplanes with larger margin, so that the VC-dimension actually remains controlled. Those two elements are at the core of the design of Support Vector Machines.

Support Vector Machines aim at finding the linear separator with *maximal margin*, i.e. the separator that is "the farthest" from the data points, as depicted on Fig.1.1, in a high-dimensional space, with respect to the training set $\mathcal{S}$. Among all the possible linear separators, the one (if exists) found with the SVM provably has good generalisation abilities. More formally, the SVM problem can be cast as follows:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{1.2}$$
$$\text{s.t.}\, \forall i, y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1$$

The constraints of problem (1.2) impose that each example $(\boldsymbol{x}_i, y_i)$ is correctly classified with some margin. At the optimum $\boldsymbol{w}^*$, one can show that this margin will exactly be:

$$\gamma = \frac{1}{\|\boldsymbol{w}^*\|}.$$

As $\boldsymbol{w}^*$ has minimal norm, the solution of the SVM problem (1.2) precisely is the linear separator with maximal margin. With this formulation, a solution exists only if the training set is separable. However, a slightly different version, called *soft-margin SVM*, also described in [Cortes and Vapnik, 1995], allows one to handle non-separable problems.

The main idea of soft-margin SVM consists in finding a separator satisfying two criteria: minimising the classification error and maximising the margin, instead of finding a separator that exactly classifies each data point. This can be achieved by introducting the so-called *slack variables* $\{\xi_i\}_{i=1}^n$. Given some separating hyperplane parameterised by $(\boldsymbol{w}, b)$, one may define, for each training data $(\boldsymbol{x}_i, y_i)$, an associated slack variable ξ_i such that:

$$\xi_i = \begin{cases} 0 & \text{if } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 \\ 1 - y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) & \text{otherwise,} \end{cases}$$

that measures the discrepancy between the actual class y_i and the prediction $\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b$ of a given linear separator.

The trade-off between the classification error (measured through the sum of the slack variables $\sum_{i=1}^n \xi_i$) and the margin can be controlled through a cost hyper-parameter C. The soft-margin SVM problem then writes:

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t. } \forall i, y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i \\ \xi_i \geq 0. \end{aligned} \tag{1.3}$$

This formulation of the problem is called *primal* as one optimises over the variables $\boldsymbol{w}$ and b, in the *primal space*, under some constraints. However, another equivalent formulation, called *dual* may be expressed. It allows to relax the constraints of the primal formulation, casting them as penalties (when the constraints are violated). The dual problem hence approximates the primal but can be simpler to solve as one does not need to directly handle the constraints in the optimisation procedure. Moreover, under some conditions on the constraints of the primal problem (e.g. Slater's conditions for instance), the *strong duality* is ensured. The problem then has the nice following property: the optimal of the dual problem exactly meets the optimal of the primal and the two formulations (primal and dual) are equivalent. For more details about duality in optimisation, one can refer to [Boyd and Vandenberghe, 2004] for instance.

Computing the Lagrangian of the soft-margin SVM primal formulation, (1.3), where the α_i's are the Lagrange multipliers (i.e. the dual variables), one can obtain the following equivalent

(dual) problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{1.4}$$
$$\text{s.t.} \sum_{i=1}^{n} y_i \alpha_i = 0$$
$$\forall i, 0 \leq \alpha_i \leq C.$$

This dual formulations is a standard *Quadratic Program* (QP). As a consequence, the historical implementations of soft-margin SVM solvers have benefited from standard black-box QP solvers (*Interior Point Methods* (IPM)for instance). When applied in the most straightforward manner, those methods have a runtime that scales as $O(n^3)$ and require the storage of the scalar product between all the pairs of points $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ for $i, j = 1..n$ (i.e. $O(n^2)$ elements). Both these aspects make the standard black-box approaches unable to handle large-scale datasets. On a side note, it is worth mentioning that when one wants to obtain an ϵ-accurate solution of the SVM problem, IPM have a training time that scales with ϵ as $O(\log(\log(\frac{1}{\epsilon}))$. As a consequence, IPM are excellent candidates when one needs to solve small SVM problems with a very high precision.

Taking into account the specificity of the SVM problems, more efficient implementations for larger-scale problems were designed. For instance, building on the fact that the solution of a SVM problem only depends on few points, namely the *support vectors* (whose number m is usually such that $m < n$), several studies ([Platt, 1998; Joachims, 1999] for instance) have developed *active sets* strategies to lower the runtime to $O(nm^2)$ and memory usage to $O(m^2)$. This improvement can be extremely beneficial, especially when dealing with separable dataset as the number of support vectors becomes much smaller than the number of data points ($m << n$). However, when the dataset is not separable the improvement may not be important enough to handle large-scale problems, as suggested by [Woodsend, 2009].

A more recent trend consists in tackling the SVM problem in the primal. For instance, [Joachims, 2006; Chapelle, 2007] proposes to approximatively solve (up to a precision ϵ) an equivalent primal formulation of (1.3) with only one global slack variable ξ but exponentially many constraints. The proposed algorithm, SVM^{perf} iteratively builds a set of active constraints and uses a simple *Cutting Plane* algorithm for solving the sequence of problems. The global algorithm has a running time that scales as $O(sn/\epsilon^2)$ where $s \leq d$ is the sparsity level of the data, i.e. the average number of non-zero features in the training set.

Shortly after SVM^{perf}, *PEGASOS* (Primal Estimated sub-GrAdient SOlver for Svm) [Shalev-Schwartz et al., 2007], based on the use of *stochastic gradient descent* with projection steps, has improved the state-of-the-art with a runtime that scales as $O(s/\epsilon)$. Besides a better dependency on the precision ϵ, the use of stochastic gradient descent makes it possible to reach a runtime that does *not* directly depend on the size n of the training set, which makes it extremely efficient for dealing with problems where the number of data n is huge. Interestingly, this breakthrough with respect to the computational efficiency of the algorithm contrasts with the simplicity of the stochastic gradient descent algorithm. It is worth mentioning that this last remark had already been pointed out by [Bottou and Le Cun, 2003] and further analysed in [Bottou and Bousquet, 2007].

Taking the aforementioned Trade-Offs of Large-Scale Learning directly into account, Hazan et al. [2011] has higlighted that in order to reach some given *generalisation level* δ (i.e. ensuring that the true risk of the learnt predictor is no larger than this of a reference hypothesis plus a fixed $\delta > 0$), the required training time of PEGASOS scaled linearly with the number of data n. In the process, they develop a bi-stochastic primal-dual approach, that further improves that dependency. In a nutshell, the bi-stochastic approach, that is to say stochastic in both the primal and the dual, only requires to access one (random) feature of one (random) example, at each iteration of the proposed algorithm. Under some favourable conditions (on the high-dimensionality and on the level of noise of the problem), this new algorithm provably has a runtime that is "smaller" than the size of the dataset required for reaching the desired level of generalisation.

Going into further details about those approaches falls out of the scope of that introduction. However, this short overview highlights that breakthroughs in the computational efficiency of learning algorithms can be achieved by a deep joint understanding of: a) the essence of large-scale learning, b) the structure of the learning problem at hand and c) how this structure can be exploited in the optimisation procedure.

Naturally, there is no reason why this trend could not be explored further in order to derive even faster SVM algorithms. However, as stated earlier in this section, SVM have consistently benefited from an unprecedented attention in the machine learning community. In that sense, there is still room for a lot of work so that the efforts made for SVM can carry over to other machine learning problems or to more complex learning tasks than binary classification. In

essence, this is what the present thesis is about.

1.3 Outline of the thesis

Throughout this thesis, we will use tools and elements from the field of Numerical Optimisation. For the sake of self-containedness and to make the thesis easier to read, in Chapter 2, we first introduce some elements of Optimisation that will be further used in the subsequent chapters. More specifically we introduce some definitions and properties that are especially relevant when dealing with Machine Learning problems.

The VC theory gave rise to generalisation bounds, connecting the capacity of the hypothesis class with the generalisation abilities of the learnt predictors. Two decades later, Bousquet and Elisseeff [2002] has given new insights on generalisation, focusing on algorithmic properties of learning procedures. Defining the notion of *algorithmic stability*, the authors made it possible to draw a connection between properties that were seen as purely algorithmic and the statistical notion of generalisation abilities. However, this early work suffers from an exclusive focus on regression and binary classification problems, using the classical definition of risk. To overcome this limitation, Chapter 3 proposes an extension to the multi-class setting, suggesting the use of the *Confusion Matrix*, as a finer-grain measure than the classical risk. To the best of our knowledge, this work is the first that focuses on Confusion Matrices from a theoretical point of view and derives generalisation bounds.

In Chapter 4, we explore another kind of connection between statistics and optimisation. The use of kernel machines obviously makes it hard to deal with large-scale problems as one has to deal with *Gram* matrices, whose size directly scales with the number of data. To address this issue, one may make use of low-rank approximations of those matrices, so that one can deal with more compact representation of the data, scaling better with large-scale problems. From that point, two natural questions arise: how to find a good low-rank representation of the data? How to efficiently leverage this low-rank structure for more computational efficiency and lower memory usage? Chapter 4 tries to tackle both problems at the same time by designing an iterative algorithm that: a) tailors a representation for the learning task at hand and b) efficiently makes use of the structure of the problem to provide inexpensive update formulas.

As described earlier, the analysis conducted by Bottou and Bousquet [2007] urges machine

learners to consider solving optimisation problems with a lower precision and pay closer attention to the computational cost of the optimisation procedures. Chapter 5 precisely aims at providing an analysis of *inexact proximal-gradient* algorithms that takes their computational costs into account. Building on that analysis, we derive a strategy that directly minimises the computational cost of the algorithm, under the constraint that problem (5.1) is approximated with some desired accuracy. We show that this new strategy is fundamentally different from those that achieve optimal convergence rates

Chapter 2

Optimisation for Machine Learning: Basic Definitions, Methods and Properties

Contents

2.1 Convexity 13
2.2 First-Order Methods 15
2.3 Rates of Convergence 16
2.4 A word on non-smooth convex optimisation 18

Some basic elements of Optimisation will be used throughout this thesis. In order to make the reading of the following chapters easier, we introduce some of these elements here, focusing on the definitions, results or algorithms, along with some elements of discussion explaining why they are relevant in Machine Learning.

2.1 Convexity

Many learning algorithms aim at finding the best predictor (in an "optimisation" sense), usually parameterised by some vector $\boldsymbol{w}$. For instance, in the context of binary classification, let $\mathcal{X}$ denote the input space and $\mathcal{Y} := \{-1; +1\}$ the target space. One may consider using the hypothesis class $\mathcal{H}$ formed by linear predictors such that any hypothesis $h_{\boldsymbol{w}} \in \mathcal{H}$ has the following form:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle).$$

Inspired by the *Structural Risk Minimisation* principle described in the introduction, many algorithms consist in, given some training set $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, solving the following generic

optimisation problem:

$$\boldsymbol{w}^* \in \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{w}, \boldsymbol{x}_i, y_i) + \lambda\, \Omega(\boldsymbol{w}), \tag{2.1}$$

where ℓ is a loss function that usually measures the discrepancy between the prediction $h_{\boldsymbol{w}}(\boldsymbol{x}_i)$ and the actual label y_i, and $\Omega(\boldsymbol{w})$ is a regularising term aiming at controlling the complexity of the solution. The choice of both ℓ and Ω leaves a lot of options and will have an important impact on both statistical and optimisation properties of the algorithm at hand.

While the choice of the 0-1 loss:

$$\ell(\boldsymbol{w}, \boldsymbol{x}_i, y_i) = \mathbb{I}_{\{h_{\boldsymbol{w}}(\boldsymbol{x}_i) \neq y_i\}}$$

seems like an obvious choice for binary classification, it actually makes it very hard to solve (2.1). As a result, other choices of loss functions, sometimes referred as *surrogate losses*, are usually preferred.

Let us introduce now some important properties that make optimisation problems easier to handle. First of all, a lot of optimisation problems cast as *convex* ones in Machine Learning.

Definition 1 (Convex Set)**.** *A set $\mathcal{X} \in \mathbb{R}^d$ is said to be* convex *if,*

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \forall t \in [0,1], \quad t\,\boldsymbol{x} + (1-t)\,\boldsymbol{x}'$$

belongs to $\mathcal{X}$

We can define a *convex* function as:

Definition 2 (Convex Function)**.** *A real-valued function $f : \mathcal{X} \to \mathbb{R}$ defined on a convex set $\mathcal{X}$ is called* convex *if for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ and any $t \in [0,1]$, the following holds:*

$$f(t\,\boldsymbol{x} + (1-t)\,\boldsymbol{x}') \leq t\, f(\boldsymbol{x}) + (1-t)\, f(\boldsymbol{x}')$$

It is said to be strictly convex *if the following holds, for any $t \in]0,1[$:*

$$f(t\,\boldsymbol{x} + (1-t)\,\boldsymbol{x}') < t\, f(\boldsymbol{x}) + (1-t)\, f(\boldsymbol{x}')$$

A stronger property (implying strict convexity), namely *strong convexity* can also be defined:

Definition 3 (Stongly-Convex Function)**.** *A real-valued function $f : \mathcal{X} \to \mathbb{R}$ defined on a convex set $\mathcal{X}$ is said to be* strongly convex *with parameter $\mu > 0$ if, for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ and any $t \in [0,1]$, the following holds:*

$$f(t\,\boldsymbol{x} + (1-t)\,\boldsymbol{x}') \leq t\, f(\boldsymbol{x}) + (1-t)\, f(\boldsymbol{x}') - \frac{1}{2}\,\mu\, t\,(1-t)\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2.$$

Convex functions have the following simple property:

Theorem 1. *A local minimum of a convex function on a convex set is always a global minimum. If the function is strictly convex, this minimum is unique.*

As a direct consequence, whenever possible, one may prefer choosing a loss function ℓ and a regulariser Ω such that the functional (2.1) to minimise is convex. This motivates the use, in most machine learning algorithms, of *convex surrogate* losses to minimise in place of the 0-1 loss. The choice of a specific loss function has many statistical consequences but falls out of the scope of this thesis (for a deeper analysis on the topic, one may refer to [Ben-David et al., 2012]). From now on, we will limit the scope of this thesis to *convex optimisation*.

A vast literature provides us with a large number of methods to minimise convex functions. For a broader overview of the field, one may refer to [Boyd and Vandenberghe, 2004; Nocedal and Wright, 1999]. We now focus on a certain class, namely the First-Order Methods.

2.2 First-Order Methods

Given some functional $f : \mathcal{X} \to \mathbb{R}$ to minimise, descent algorithms aim at producing a sequence $\{\boldsymbol{x}_k\}_k$ of the following form:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + t_k \, \boldsymbol{\Delta}\mathbf{x}_k,$$

that converges to a minimiser $\boldsymbol{x}^*$ of f. One may notice that an iterate $\boldsymbol{x}_k$, will depend on its predecessor $\boldsymbol{x}_{k-1}$, $\boldsymbol{\Delta}\mathbf{x}_k$, which defines the direction along which one will descend, and $t_k > 0$, the step-size, which determines how far we will descend.

The so-called *First-Order Methods* have been devised for the situation where f is differentiable. They have been extensively used because they are provably efficient and easy to implement. As might be expected, they rely on the use of the first-order information on f, i.e. the gradient, to define the descent direction:

$$\boldsymbol{\Delta}\mathbf{x}_k := -\nabla\mathbf{f}(\boldsymbol{x}_\mathbf{k}),$$

where $\nabla\mathbf{f}$ is the gradient of f.

Regardless of the choice of the step-size, using $-\nabla\mathbf{f}(\boldsymbol{x}_\mathbf{k})$ as a descent direction gives rise to a class of well-known iterative optimisation algorithms called *gradient descent* depicted in

Algorithm 1. The typical behaviour of the iterates obtained with a gradient descent algorithm is illustrated in Figure 2.1[1].

Algorithm 1 The gradient descent algorithm

Require: An initial point $\boldsymbol{x}_0$

repeat

$\boldsymbol{\Delta}\mathbf{x}_k := -\nabla\mathbf{f}(\boldsymbol{x}_\mathbf{k})$

Choose step size t_k

$\boldsymbol{x}_k := \boldsymbol{x}_{k-1} + t_k\,\boldsymbol{\Delta}\mathbf{x}_k$

until a stopping criterion is met

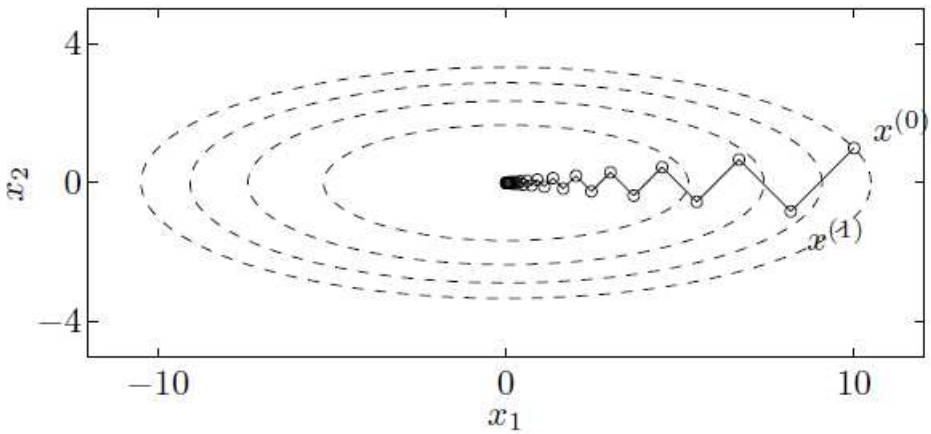

Figure 2.1: Some contour lines of the function $f(\boldsymbol{x}) = (1/2)(x_1^2 + 10x_2^2)$. The figure shows the iterates of the gradient method with exact line search, started at $\boldsymbol{x}^{(0)} = (10, 1)$.

2.3 Rates of Convergence

In order to analyse the convergence of the gradient descent algorithm, let us define one new desirable property for the function f to minimise:

Definition 4 (Convex Function with L-Lipschitz Continuous Gradient)**.** *A real-valued convex and differentiable function $f : \mathcal{X} \to \mathbb{R}$ defined on a convex set $\mathcal{X}$ has L-Lipschitz Continuous*

[1]This figure is courtesy of Stephen Boyd and Lieven Vandenberghe

Gradient if for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, the following holds:

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{x}')\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{x}'\|_2,$$

or, equivalently:

$$\forall t \in [0,1], \quad t\, f(\boldsymbol{x}) + (1-t)\, f(\boldsymbol{x}') \leq f(t\,\boldsymbol{x} + (1-t)\,\boldsymbol{x}') + \frac{L}{2}\, t\,(1-t)\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2.$$

Now, we can derive interesting results on the rates of convergence of the Gradient Descent algorithm described earlier. Those results (and their proofs) can be found in [Nesterov, 2004].

Theorem 2 (Rate of Convergence of Gradient Descent for Convex Function with L-Lipschitz Continuous Gradient)**.** *Let f be a convex function with L-Lipschitz continuous gradient . Then the gradient method (with constant step-size $h = \frac{1}{L}$) generates a sequence $\{\boldsymbol{x}_k\}$ which converges as follows:*

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \leq \frac{2\,L\,\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2}{k+4}.$$

First of all, this result shows that the Gradient Descent algorithm produces a sequence $\{\boldsymbol{x}_k\}$ that converges to a minimiser $\boldsymbol{x}^*$ of f. Moreover, it does so with an asymptotic rate of $O(1/k)$ which is often referred as *sublinear*.

With stronger assumptions on f, one can even prove that the Gradient Descent algorithm converges "faster" than $O(1/k)$.

Theorem 3 (Rate of Convergence of Gradient Descent for Strongly Convex Function with L-Lipschitz Continuous Gradient)**.** *Let f be a μ strongly-convex function with L-Lipschitz continuous gradient. Then the gradient method (with constant step-size $h = \frac{2}{\mu+L}$) generates a sequence $\{\boldsymbol{x}_k\}$ which converges as follows:*

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \leq \frac{L}{2}\left(\frac{Q_f - 1}{Q_f + 1}\right)^{2k} \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2,$$

where $Q_f = L/\mu$.

Under the strong convexity assumption, Gradient Descent can provably reach a so-called "*linear* rate" of convergence.

Those scarce results allowed us to introduce the notion of *rates of convergence* of optimisation algorithms. It is worth mentioning that this is a limited overview of the abundant field of

convex optimisation. Plenty of other algorithms exist. For many of them, similar results on the convergence have been derived. Of course, the rates of convergence of those methods vary a lot and it is worth paying a closer attention to them.

Another important conclusion is that the typical behaviour of one specific algorithm will greatly depend on the properties of the function to be optimised. As a consequence, the design of a learning problem (i.e. the design of the function f) will have a tremendous impact on the performance of the optimisation procedure. In order to produce computationally efficient Machine Learning algorithms, one should always keep in mind that, in the light of this discussion, the design of the problem to be solved is of utmost importance.

2.4 A word on non-smooth convex optimisation

Gradient Descent algorithms are widely used for solving smooth convex optimisation problems and rely on the gradient of the function f to minimise. However, some problems of the form of (2.1) impose to deal with non-smooth functions. For instance, SVM rest on the use of the *hinge loss* function:

$$\ell(\boldsymbol{w}, \boldsymbol{x}, y) = \max(0, 1 - y \langle \boldsymbol{w}, \boldsymbol{x} \rangle),$$

which is not differentiable when $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = 1/y$. Another well-known example is the use of a one-norm regularising term (as in the Lasso [Tibshirani, 1996], for instance):

$$\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_1,$$

which is not differentiable whenever any component of $\boldsymbol{w}$ is 0.

Fortunately, *subgradient* generalises the concept of gradient to convex but non-smooth functions:

Definition 5. *Let f be a convex function. A vector $\boldsymbol{g}$ is called a subgradient of f at point $\boldsymbol{x}_0$ if for any $\boldsymbol{x}$, we have:*

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \langle \boldsymbol{g}, \boldsymbol{x} - \boldsymbol{x}_0 \rangle.$$

The set of all subgradients of f at $\boldsymbol{x}_0$, $\partial f(\boldsymbol{x}_0)$, is called the subdifferential *of f at $\boldsymbol{x}_0$.*

Remark 1. If f is differentiable at $\boldsymbol{x}_0$, the usual gradient $\nabla f(\boldsymbol{x}_0)$ is the only subgradient of f at $\boldsymbol{x}_0$ (i.e. $\partial f(\boldsymbol{x}_0) = \{\nabla f(\boldsymbol{x}_0)\}$). As a consequence, the usual concept of gradient is a particular case of the more general concept of subgradient.

As a direct extension of Gradient Descent, one can derive the so-called *Subgradient Descent* algorithms (see [Nesterov, 2004], for instance), using one subgradient in ∂f as a descent direction. We will not develop here this point much further. However, one important issue with those algorithms lies in the computation and non-trivial choice of a specific subgradient in order to define the descent direction. Addressing this point, the *proximal-gradient* methods ([Combettes and Wajs, 2005]) have been meeting an increasing success in the Machine Learning community. More formally, they make it possible to handle the minimisation of *composite* functions of the form:

$$\min_x f(x) := g(x) + h(x), \tag{2.2}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ is convex and smooth with an L-Lipschitz continuous gradient and $h : \mathbb{R}^n \to \mathbb{R}$ is *lower semi-continuous proper convex.*

Definition 6 (Lower Semi-Continuity)**.** *A function f is* lower semi-continuous at x_0 *if, for every $\epsilon > 0$, there exists a neighbourhood U of x_0 such that $f(x) \geq f(x_0) - \epsilon$ for all x in U. A function f is simply called* lower semi-continuous *if it is lower semi-continuous at each point of its domain.*

Definition 7 (Proper Convexity)**.** *A convex function f is* proper convex *it takes values in the extended real number line $\mathbb{R} \cup \{+\infty\}$ and is such that $f(x) < +\infty$ for at least one x.*

Essentially, lower semi-continuous proper convex, which combine both of those properties, cover continuous non-smooth convex functions and indicators on convex set. We defer the presentation of those methods to Chapter 5 where they will be further studied.

Chapter 3

Confusion Matrix Stability Bounds for Multiclass Classification

Contents

3.1 Introduction **22**

3.2 Confusion Loss **23**

3.2.1 Notation 23

3.2.2 Confusion Matrix versus Misclassification Rate 24

3.3 Deriving Stability Bounds on the Confusion Matrix **27**

3.3.1 Stability 27

3.3.2 Noncommutative McDiarmid's Bounded Difference Inequality 28

3.3.3 Stability Bound 29

3.4 Analysis of existing algorithms **31**

3.4.1 Hilbert Space Regularised Algorithms 31

3.4.2 Lee, Lin and Wahba model 32

3.4.3 Weston and Watkins model 33

3.5 Discussion and Conclusion **34**

Abstract

In this chapter, we provide new theoretical results on the generalisation properties of learning algorithms for multiclass classification problems. The originality of our work is that we propose to use the *confusion matrix* of a classifier as a measure of its quality; our contribution is in the line of work which attempts to set up and study the statistical properties of new evaluation measures

such as, e.g. ROC curves. In the confusion-based learning framework we propose, we claim that a targeted objective is to minimise the size of the confusion matrix $\mathcal{C}$, measured through its *operator norm* $\|\mathcal{C}\|$. We derive generalisation bounds on the (size of the) confusion matrix in an extended framework of uniform stability, adapted to the case of matrix valued loss. Pivotal to our study is a very recent matrix concentration inequality that generalises McDiarmid's inequality. As an illustration of the relevance of our theoretical results, we show how two SVM learning procedures can be proved to be confusion-friendly. To the best of our knowledge, the present chapter is the first that focuses on the confusion matrix from a theoretical point of view.

3.1 Introduction

Multiclass classification is an important problem of machine learning. The issue of having at hand statistically relevant procedures to learn reliable predictors is of particular interest, given the need of such predictors in information retrieval, web mining, bioinformatics or neuroscience (one may for example think of document categorisation, gene classification, fMRI image classification).

Yet, the literature on multiclass learning is not as voluminous as that of binary classification, while this multiclass prediction raises questions from the algorithmic, theoretical and practical points of view. One of the prominent questions is that of the measure to use in order to assess the quality of a multiclass predictor. Here, we develop our results with the idea that the *confusion matrix* is a performance measure that deserves to be studied as it provides a finer information on the properties of a classifier than the mere misclassification rate. We do want to emphasise that we provide theoretical results on the confusion matrix itself and that misclassification rate *is not* our primary concern —as we shall see, though, getting bounds on the confusion matrix entails, as a byproduct, bounds on the misclassification rate.

Building on matrix-based concentration inequalities [Recht, 2011; Tropp, 2011; Gosh et al., 2011; Rudelson and Vershynin, 2007; Chaudhuri et al., 2009], also referred to as noncommutative concentration inequalities, we establish a stability framework for confusion-based learning algorithm. In particular, we prove a generalisation bound for *confusion stable* learning algorithms and show that there exist such algorithms in the literature. In a sense, our framework and our results extend those of [Bousquet and Elisseeff, 2002], which are designed for scalar loss functions. To the best of our knowledge, this is the first work that establishes generalisation bounds based

on confusion matrices.

The chapter is organised as follows. Section 3.2 describes the setting we are interested in and motivates the use of the confusion matrix as a performance measure. Section 3.3 introduces the new notion of *stability* that will prove essential to our study; the main theorem of this chapter, is provided. Please refer to 7.1 for the proof. Section 3.4 is devoted to the analysis of two SVM procedures in the light of our new framework. A discussion on the merits and possible extensions of our approach concludes the chapter (Section 3.5).

3.2 Confusion Loss

3.2.1 Notation

As said earlier, we focus on the problem of multiclass classification. The input space is denoted by $\mathcal{X}$ and the target space is

$$\mathcal{Y} = \{1, \dots, Q\}.$$

The training sequence

$$\mathbf{Z} = \{Z_i = (X_i, Y_i)\}_{i=1}^m$$

is made of m identically and independently random pairs $Z_i = (X_i, Y_i)$ distributed according to some unknown (but fixed) distribution D over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. The sequence of input data will be referred to as $\boldsymbol{X} = \{X_i\}_{i=1}^m$ and the sequence of corresponding labels $\boldsymbol{Y} = \{Y_i\}_{i=1}^m$, we may write $\mathbf{Z} = \{\boldsymbol{X}, \boldsymbol{Y}\}$. The realisation of $Z_i = (X_i, Y_i)$ is $z_i = (x_i, y_i)$ and $\boldsymbol{z}$, $\boldsymbol{x}$ and $\boldsymbol{y}$ refer to the realisations of the corresponding sequences of random variables. For a sequence $\boldsymbol{y} = \{y_1, \cdots, y_m\}$ of m labels, $m_q(\boldsymbol{y})$, or simply m_q when clear from context, denotes the number of labels from $\boldsymbol{y}$ that are equal to q; $\boldsymbol{s}(\boldsymbol{y})$ it the binary sequence $\{s_1(\boldsymbol{y}), \dots, s_Q(\boldsymbol{y})\}$ of size Q such that:

$$s_q(\boldsymbol{y}) = \begin{cases} 1 \text{ if } q \in \boldsymbol{y} \\ 0 \text{ otherwise.} \end{cases}$$

We will use $D_{X|y}$ for the conditional distribution of X given that $Y = y$; therefore, for a given sequence $\boldsymbol{y} = \{y_1, \dots, y_m\} \in \mathcal{Y}^m$,

$$D_{\boldsymbol{X}|\boldsymbol{y}} = \otimes_{i=1}^m D_{X|y_i}$$

is the distribution of the random sample $\boldsymbol{X} = \{X_1, \ldots, X_m\}$ over $\mathcal{X}^m$ such that X_i is distributed according to $D_{X|y_i}$; for $q \in \mathcal{Y}$, and $\boldsymbol{X}$ distributed according to $D_{\boldsymbol{X}|\boldsymbol{y}}$, $\boldsymbol{X}_q = \{X_{i_1}, \ldots, X_{i_{m_q}}\}$ denotes the random sequence of variables such that X_{i_k} is distributed according to $D_{X|q}$. $\mathbb{E}[\cdot]$ and $\mathbb{E}_{X|y}[\cdot]$ denote the expectations with respect to D and $D_{X|y}$, respectively.

For a training sequence $\mathbf{Z}$, $\mathbf{Z}^i$ denotes the sequence

$$\mathbf{Z}^i = \{Z_1, \ldots Z_{i-1}, Z_i', Z_{i+1}, \ldots, Z_m\}$$

where Z_i' is distributed as Z_i; $\mathbf{Z}^{\setminus i}$ is the sequence

$$\mathbf{Z}^{\setminus i} = \{Z_1, \ldots Z_{i-1}, Z_{i+1}, \ldots, Z_m\}.$$

These definitions directly carry over when conditioned on a sequence of labels $\boldsymbol{y}$ (with, henceforth, $y_i' = y_i$).

We will consider a family $\mathcal{H}$ of predictors such that

$$\mathcal{H} \subseteq \left\{h : h(x) \in \mathbb{R}^Q,\ \forall x \in \mathcal{X}\right\}.$$

For $h \in \mathcal{H}$, $h_q \in \mathbb{R}^{\mathcal{X}}$ denotes its qth coordinate. Also,

$$\boldsymbol{\ell} = (\ell_q)_{1 \leq q \leq Q}$$

is a set of loss functions such that:

$$\ell_q : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+.$$

Finally, for a given algorithm $\mathcal{A} : \cup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H}$, $\mathcal{A}_{\mathbf{Z}}$ will denote the hypothesis learned by $\mathcal{A}$ when trained on $\mathbf{Z}$.

3.2.2 Confusion Matrix versus Misclassification Rate

We here provide a discussion as to why minding the *confusion matrix* or *confusion loss* (terms that we will use interchangeably) is crucial in multiclass classification. We also introduce the reason why we may see the confusion matrix as an operator and, therefore, motivate the recourse to the *operator norm* to measure the 'size' of the confusion matrix.

In many situations, e.g. class-imbalanced datasets, it is important not to measure the quality of a predictor h on its classification error $\mathbb{P}_{XY}(h(X) \neq Y)$ only, as this may lead to erroneous

conclusions regarding the quality of h. Indeed, if, for instance, some class q is predominantly present in the data at hand, say $\mathbb{P}(Y = q) = 1 - \varepsilon$, for some small $\varepsilon > 0$, then the predictor h_{maj} that always outputs $h_{\text{maj}}(x) = q$ regardless of x has a classification error lower than ε. Yet, it might be important not to classify an instance of some class p in class q: take the example of classifying mushrooms according to the categories $\{\texttt{hallucinogen}, \texttt{poisonous}, \texttt{innocuous}\}$, it might not be benign to predict `innocuous` (the majority class) instead of `hallucinogen` or `poisonous`. The framework we consider allows us, among other things, to be immune to situations where class-imbalance may occur.

We do claim that a more relevant object to consider is the *confusion matrix* which, given a binary sequence $\boldsymbol{s} = \{s_1 \cdots s_Q\} \in \{0,1\}^Q$, is defined as

$$\mathcal{C}_{\boldsymbol{s}}(h) := \sum_{q:s_q=1} \mathbb{E}_{X|q} L(h, X, q),$$

where, given an hypothesis $h \in \mathcal{H}$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$, $L(h, x, y) = (l_{ij})_{1 \le i,j \le Q} \in \mathbb{R}^{Q \times Q}$ is the *loss matrix* such that:

$$l_{ij} := \begin{cases} \ell_j(h, x, y) & \text{if } i = y \text{ and } i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

Note that this matrix has at most one nonzero row, namely its ith row.

For a sequence $\boldsymbol{y} \in \mathcal{Y}^m$ of m labels and a random sequence $\boldsymbol{X}$ distributed according to $D_{\boldsymbol{X}|\boldsymbol{y}}$, the conditional empirical confusion matrix $\widehat{\mathcal{C}}_{\boldsymbol{y}}(h, \boldsymbol{X})$ is

$$\widehat{\mathcal{C}}_{\boldsymbol{y}}(h, \boldsymbol{X}) := \sum_{i=1}^{m} \frac{1}{m_{y_i}} L(h, X_i, y_i) = \sum_{q \in \boldsymbol{y}} \frac{1}{m_q} \sum_{i:y_i=q} L(h, X, q) = \sum_{q \in \boldsymbol{y}} L_q(h, \boldsymbol{X}, \boldsymbol{y}),$$

where

$$L_q(h, \boldsymbol{X}, \boldsymbol{y}) := \frac{1}{m_q} \sum_{i:y_i=q} L(h, X_i, q).$$

For a random sequence $\boldsymbol{Z} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ distributed according to D^m, the (unconditional) empirical confusion matrix is given by

$$\mathbb{E}_{\boldsymbol{X}|\boldsymbol{Y}} \widehat{\mathcal{C}}_{\boldsymbol{Y}}(h, \boldsymbol{X}) = \mathcal{C}_{\boldsymbol{s}(\boldsymbol{Y})}(h),$$

which is a random variable, as it depends on the random sequence $\boldsymbol{Y}$. For exposition purposes it will often be more convenient to consider a fixed sequence $\boldsymbol{y}$ of labels and state results on $\widehat{\mathcal{C}}_{\boldsymbol{y}}(h, \boldsymbol{X})$, noting that

$$\mathbb{E}_{\boldsymbol{X}|\boldsymbol{y}} \widehat{\mathcal{C}}_{\boldsymbol{y}}(h, \boldsymbol{X}) = \mathcal{C}_{\boldsymbol{s}(\boldsymbol{y})}(h).$$

The slight differences between our definitions of (conditional) confusion matrices and the usual definition of a confusion matrix is that the diagonal elements are all zero and that they can accomodate any family of loss functions (and not just the 0-1 loss).

A natural objective that may be pursued in multiclass classification is to learn a classifier h with 'small' confusion matrix, where 'small' might be defined with respect to (some) matrix norm of $\mathcal{C}_{\boldsymbol{s}}(h)$. The norm that we retain is the *operator norm* that we denote $\|\cdot\|$ from now on: recall that, for a matrix M, $\|M\|$ is computed as

$$\|M\| = \max_{\boldsymbol{v}\neq\boldsymbol{0}} \frac{\|M\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2},$$

where $\|\cdot\|_2$ is the Euclidean norm; $\|M\|$ is merely the largest singular value of M —note that $\|M^\top\| = \|M\|$.

Not only is the operator norm a 'natural' norm on matrices but an important reason for working with it is that $\mathcal{C}_{\boldsymbol{s}}(h)$ is often precisely used as an *operator* acting on the vector of prior distributions

$$\boldsymbol{\pi} = [\mathbb{P}(Y=1)\cdots\mathbb{P}(Y=Q)]^\top.$$

Indeed, a quantity of interest is for instance the $\boldsymbol{\ell}$*-risk* $R_{\boldsymbol{\ell}}(h)$ of h, with

$$\begin{aligned} R_{\boldsymbol{\ell}}(h) &:= \mathbb{E}_{XY}\left\{\sum_{q=1}^{Q}\ell_q(h,X,Y)\right\} = \mathbb{E}_Y\left\{\sum_{q=1}^{Q}\mathbb{E}_{X|Y}\ell_q(h,X,Y)\right\} \\ &= \sum_{p,q=1}^{Q}\mathbb{E}_{X|p}\ell_q(h,X,p)\pi_p = \|\boldsymbol{\pi}^\top\mathcal{C}_{\mathbf{1}}(h)\|_1. \end{aligned}$$

It is interesting to observe that, $\forall h, \forall\boldsymbol{\pi}\in\Lambda := \{\boldsymbol{\lambda}\in\mathbb{R}^Q : \lambda_q \geq 0, \sum_q \lambda_q = 1\}$:

$$\begin{aligned} 0 \leq R_{\boldsymbol{\ell}}(h) &= \|\boldsymbol{\pi}\mathcal{C}_{\mathbf{1}}(h)\|_1 = \boldsymbol{\pi}^\top\mathcal{C}_{\mathbf{1}}(h)\mathbf{1} \\ &\leq \sqrt{Q}\left\|\boldsymbol{\pi}^\top\mathcal{C}_{\mathbf{1}}(h)\right\|_2 = \sqrt{Q}\left\|\mathcal{C}_{\mathbf{1}}^\top(h)\boldsymbol{\pi}\right\|_2 \\ &\leq \sqrt{Q}\left\|\mathcal{C}_{\mathbf{1}}^\top(h)\right\|\|\boldsymbol{\pi}\|_2 \leq \sqrt{Q}\left\|\mathcal{C}_{\mathbf{1}}^\top(h)\right\| = \sqrt{Q}\,\|\mathcal{C}_{\mathbf{1}}(h)\|, \end{aligned}$$

where we have used Cauchy-Schwarz inequalty in the second line, the definition of the operator norm on the third line and the fact that $\|\boldsymbol{\pi}\|_2 \leq 1$ for any $\boldsymbol{\pi}$ in Λ; $\mathbf{1}$ is the Q-dimensional vector where each entry is 1. Recollecting things, we just established the following proposition.

Proposition 1. $\forall h\in\mathcal{H},\ R_{\boldsymbol{\ell}}(h) = \|\boldsymbol{\pi}^\top\mathcal{C}_{\mathbf{1}}(h)\|_1 \leq \sqrt{Q}\,\|\mathcal{C}_{\mathbf{1}}(h)\|.$

This precisely says that the operator norm of the confusion matrix (according to our definition) provides a bound on the risk. As a consequence, bounding $\|\mathcal{C}_{\mathbf{1}}(h)\|$ is a relevant way to bound the risk in a way that is independent from the class priors (since the $\mathcal{C}_{\mathbf{1}}(h)$ is independent form these prior distributions as well). This is essential in class-imbalanced problems and also critical if sampling (prior) distributions are different for training and test data.

Again, we would like to insist on the fact that the confusion matrix *is the subject of our study* for its ability to provide fine-grain information on the prediction errors made by classifiers; as mentioned in the introduction, there are application domains where confusion matrices *indeed are* the measure of performance that is looked at. If needed, the norm of the confusion matrix allows us to summarise the characteristics of the classifiers in one scalar value (the larger, the worse), and it provides, as a (beneficial) "side effect", a bound on $R_{\boldsymbol{\ell}}(h)$.

3.3 Deriving Stability Bounds on the Confusion Matrix

One of the most prominent issues in *learning theory* is to estimate the real performance of a learning system. The usual approach consists in studying how empirical measures converge to their expectation. In the traditional settings, it often boils down to providing bounds describing how the empirical risk relates to the expected one. In this work, we show that one can use similar techniques to provide bounds on (the operator norm of) the confusion loss.

3.3.1 Stability

Following the early work of [Vapnik, 1982], the risk has traditionally been estimated through its empirical measure and a measure of the complexity of the hypothesis class such as the Vapnik-Chervonenkis dimension, the fat-shattering dimension or the Rademacher complexity. During the last decade, a new and successful approach based on *algorithmic stability* to provide some new bounds has emerged. One of the highlights of this approach is the focus on properties of the learning algorithm at hand, instead of the richness of hypothesis class. In essence, algorithmic stability results aim at taking advantage from the way a given algorithm actually explores the hypothesis space, which may lead to tight bounds. The main results of [Bousquet and Elisseeff, 2002] were obtained using the definition of *uniform stability*.

Definition 8 (Uniform stability [Bousquet and Elisseeff, 2002])**.** *An algorithm $\mathcal{A}$ has* uniform

stability β *with respect to loss function* ℓ *if the following holds:*

$$\forall \mathbf{Z} \in \mathcal{Z}^m, \forall i \in \{1, \dots, m\}, \|\ell(\mathcal{A}_{\mathbf{Z}}, .) - \ell(\mathcal{A}_{\mathbf{Z}^{\setminus i}}, .)\|_\infty \leq \beta.$$

In the present chapter, we now focus on the generalisation of stability-based results to confusion loss. We introduce the definition of *confusion stability.*

Definition 9 (Confusion stability)**.** *An algorithm* $\mathcal{A}$ *is* confusion stable *with respect to the set of loss functions* $\boldsymbol{\ell}$ if *there exists a constant* $B > 0$ *such that* $\forall i \in \{1, \dots, m\}, \forall \boldsymbol{z} \in \mathcal{Z}^m$*, whenever* $m_q \geq 2, \forall q \in \mathcal{Y}$,

$$\sup_{x \in \mathcal{X}} \|L(\mathcal{A}_{\boldsymbol{z}}, x, y_i) - L(\mathcal{A}_{\boldsymbol{z}^{\setminus i}}, x, y_i)\| \leq \frac{B}{m_{y_i}}.$$

From here on, q^**,* m^* *and* β^* *will stand for*

$$q^* := \operatorname*{argmin}_q m_q, \ m^* := m_{q^*}, \ \text{and } \beta^* := B/m^*.$$

3.3.2 Noncommutative McDiarmid's Bounded Difference Inequality

Central to the result of [Bousquet and Elisseeff, 2002] is a variation of Azuma's concentration inequality, due to [McDiarmid, 1989]. It describes how a scalar function of independent random variables (the elements of our training set) concentrates around its mean, given how changing one of the random variables impacts the value of the function.

Recently there has been an extension of McDiarmid's inequality to the matrix setting [Tropp, 2011] . For the sake of self-containedness, we recall this noncommutative bound.

Theorem 4 (Matrix bounded difference ([Tropp, 2011], corollary 7.5))**.** *Let* H *be a function that maps* m *variables from some space* $\mathcal{Z}$ *to a self-adjoint matrix of dimension* $2Q$*. Consider a sequence* $\{A_i\}$ *of fixed self-adjoint matrices that satisfy*

$$\big(H(z_1, \dots, z_i, \dots, z_m) - H(z_1, \dots, z_i', \dots, z_m)\big)^2 \preccurlyeq A_i^2, \tag{3.1}$$

for $z_i, z_i' \in \mathcal{Z}$ *and for* $i = 1, \dots, m$*, where* $\preccurlyeq$ *is the (partial) order on self-adjoint matrices. Then, if* $\mathbf{Z}$ *is a random sequence of independent variables over* $\mathcal{Z}$*:*

$$\forall t \geq 0, \ \mathbb{P}\{\|H(\mathbf{Z}) - \mathbb{E}_{\mathbf{Z}} H(\mathbf{Z})\| \geq t\} \leq 2Qe^{-t^2/8\sigma^2},$$

where $\sigma^2 := \|\sum_i A_i^2\|$.

The confusion matrices we deal with are not necessarily self-adjoint, as is required by the theorem. To make use of the theorem, we rely on the dilation $\mathfrak{D}(A)$ of A, with

$$\mathfrak{D}(A) := \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix},$$

where A^* is the adjoint of A (note that $\mathfrak{D}(A)$ is self-adjoint) and on the result (see [Tropp, 2011])

$$\|\mathfrak{D}(A)\| = \|A\|.$$

3.3.3 Stability Bound

The following theorem is the main result of the chapter. It says that the empirical confusion is close to the expected confusion whenever the learning algorithm at hand exhibits confusion-stability properties. This is a new flavor of the results of [Bousquet and Elisseeff, 2002] for the case of matrix-based loss.

Theorem 5 (Confusion bound)**.** *Let $\mathcal{A}$ be a learning algorithm. Assume that all the loss functions under consideration take values in the range $[0; M]$. Let $\boldsymbol{y} \in \mathcal{Y}^m$ be a fixed sequence of labels.*

If $\mathcal{A}$ is a confusion stable as defined in Definition 9, then, $\forall m \geq 1$, $\forall \delta \in (0,1)$, the following holds, with prob. $1-\delta$ over the random draw of $\boldsymbol{X} \sim D_{\boldsymbol{X}|\boldsymbol{y}}$,

$$\left\|\widehat{\mathcal{C}}_{\boldsymbol{y}}(\mathcal{A}, \boldsymbol{X}) - \mathcal{C}_{\boldsymbol{s}(\boldsymbol{y})}(\mathcal{A})\right\| \leq 2B \sum_q \frac{1}{m_q} + Q\sqrt{8 \ln\left(\frac{Q^2}{\delta}\right)} \left(4\sqrt{m^*}\beta^* + M\sqrt{\frac{Q}{m^*}}\right).$$

As a consequence, with probability $1-\delta$ over the random draw of $\mathbf{Z} \sim D^m$,

$$\left\|\widehat{\mathcal{C}}_{\boldsymbol{Y}}(\mathcal{A}, \boldsymbol{X}) - \mathcal{C}_{\boldsymbol{s}(\boldsymbol{Y})}(\mathcal{A})\right\| \leq 2B \sum_q \frac{1}{m_q} + Q\sqrt{8 \ln\left(\frac{Q^2}{\delta}\right)} \left(4\sqrt{m^*}\beta^* + M\sqrt{\frac{Q}{m^*}}\right).$$

Sketch. The complete proof can be found in the appendices. We here provide the skeleton of the proof. We proceed in 3 steps to get the first bound.

1. **Triangle inequality.** To start with, we know by the triangle inequality

$$\begin{aligned} \|\widehat{\mathcal{C}}_{\boldsymbol{y}}(\mathcal{A}, \boldsymbol{X}) - \mathcal{C}_{\boldsymbol{s}(\boldsymbol{y})}(\mathcal{A})\| &= \left\|\sum_{q \in \boldsymbol{y}} (L_q(\mathcal{A}_{\boldsymbol{Z}}, \boldsymbol{Z}) - \mathbb{E}_{\boldsymbol{X}} L_q(\mathcal{A}_{\boldsymbol{Z}}, \boldsymbol{Z}))\right\| \\ &\leq \sum_{q \in \boldsymbol{y}} \|L_q(\mathcal{A}_{\boldsymbol{Z}}, \boldsymbol{Z}) - \mathbb{E}_{\boldsymbol{X}} L_q(\mathcal{A}_{\boldsymbol{Z}}, \boldsymbol{Z})\|. \end{aligned} \tag{3.2}$$

Using uniform stability arguments, we bound each summand with probability $1 - \delta/Q$.

2. **Union Bound.** Then, using the union bound we get a bound on $\|\widehat{\mathcal{C}}(\mathcal{A}, \boldsymbol{X}) - \mathcal{C}_{\boldsymbol{s}(\boldsymbol{y})}(\mathcal{A})\|$ that holds with probability at least $1 - \delta$.

3. **Wrap up.** Finally, recoursing to a simple argument, we express the obtained bound solely with respect to m^*.

Among the three steps, the first one is the more involved and much part of the proof is devoted to address it.

To get the bound with the unconditional confusion matrix $\mathcal{C}_{\boldsymbol{s}(\boldsymbol{Y})}(\mathcal{A})$ it suffices to observe that for any event $\mathcal{E}(\boldsymbol{X}, \boldsymbol{Y})$ that depends on $\boldsymbol{X}$ and $\boldsymbol{Y}$, such that for all sequences $\boldsymbol{y}$, $\mathbb{P}_{\boldsymbol{X}|\boldsymbol{y}}\{\mathcal{E}(\boldsymbol{X}, \boldsymbol{y})\} \leq \delta$, the following holds:

$$\begin{aligned}
\mathbb{P}_{XY}(\mathcal{E}(\boldsymbol{X}, \boldsymbol{Y})) &= \mathbb{E}_{XY}\left\{\mathbb{I}_{\{\mathcal{E}(\boldsymbol{X},\boldsymbol{Y})\}}\right\} = \mathbb{E}_{\boldsymbol{Y}}\left\{\mathbb{E}_{\boldsymbol{X}|\boldsymbol{Y}}\mathbb{I}_{\{\mathcal{E}(\boldsymbol{X},\boldsymbol{Y})\}}\right\} \\
&= \sum_{\boldsymbol{y}} \mathbb{E}_{\boldsymbol{X}|\boldsymbol{Y}}\mathbb{I}_{\{\mathcal{E}(\boldsymbol{X},\boldsymbol{Y})\}}\mathbb{P}_{\boldsymbol{Y}}(\boldsymbol{Y} = \boldsymbol{y}) = \sum_{\boldsymbol{y}} \mathbb{P}_{\boldsymbol{X}|\boldsymbol{y}}\{\mathcal{E}(\boldsymbol{X}, \boldsymbol{y})\}\mathbb{P}_{\boldsymbol{Y}}(\boldsymbol{Y} = \boldsymbol{y}) \\
&\leq \sum_{\boldsymbol{y}} \delta\mathbb{P}_{\boldsymbol{Y}}(\boldsymbol{Y} = \boldsymbol{y}) = \delta,
\end{aligned}$$

which gives the desired result. □

□

Remark 2. If needed, it is straightforward to bound $\|\mathcal{C}_{\boldsymbol{s}(\boldsymbol{y})}(\mathcal{A})\|$ and $\|\mathcal{C}_{\boldsymbol{s}(\boldsymbol{Y})}(\mathcal{A})\|$ by using the triangle inequality $|\|A\| - \|B\|| \leq \|A - B\|$ on the stated bounds.

Remark 3. A few comments may help understand the meaning of our main theorem. First, it is expected to get a bound expressed in terms of $1/\sqrt{m^*}$, since a) $1/\sqrt{m}$ is a typical rate encountered in bounds based on m data and b) the bound cannot be better than a bound devoted to the least informed class (that would be in $1/\sqrt{m^*}$) —resampling procedures may be a strategy to consider to overcome this limit. Second, this theorem says that it is a relevant idea to try and minimise the empirical confusion matrix of a multiclass predictor provided the algorithm used is stable —as will be the case of the algorithms analysed in the following section. Designing algorithm that minimise the norm of the confusion matrix is therefore an enticing challenge. Finally, when $Q = 2$, that is we are in a binary classification framework, Theorem 5 gives a bound on the maximum of the false-positive rate and the false-negative rate, since this the operator norm of the confusion matrix precisely corresponds to this maximum value.

3.4 Analysis of existing algorithms

Now that the main result on stability bound has been established, we will investigate how existing multiclass algorithms exhibit stability properties and thus fall in the scope of our analysis. More precisely, we will analyse two well-known models for multiclass support vector machines and we will show that they may promote small confusion error. But first, we will study the more general stability of multiclass algorithms using regularisation in Reproducing Kernel Hilbert Spaces (RKHS).

3.4.1 Hilbert Space Regularised Algorithms

Many well-known and widely-used algorithms feature a minimisation of a regularised objective functions [Tikhonov and Arsenin, 1977]. In the context of multiclass kernel machines [Crammer and Singer, 2001; Cristianini and Shawe-Taylor, 2000], this regulariser $\Omega(h)$ may take the following form:

$$\Omega(h) = \sum_q \|h_q\|_k^2.$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denotes the kernel associated to the RKHS $\mathcal{H}$.

In order to study the stability properties of algorithms, minimising a data-fitting term, penalised by such regularisers, in our multi-class setting, we need to introduce a minor definition that is an addition to the following existing definition:

Definition 10 (definition 19 in [Bousquet and Elisseeff, 2002])**.** *A loss function ℓ defined on $\mathcal{H} \times \mathcal{Y}$ is σ-admissible with respect to $\mathcal{H}$ if the associated cost function c is convex with respect to its first argument and the following condition holds:*

$$\forall y_1, y_2 \in D, \forall y' \in \mathcal{Y}, |c(y_1, y') - c(y_2, y')| \leq \sigma |y_1 - y_2|,$$

where $D = \{y : \exists h \in \mathcal{H}, \exists \boldsymbol{x} \in \mathcal{X}, h(\boldsymbol{x}) = y\}$ is the domain of the first argument of c.

Definition 11. *A loss function ℓ defined on $\mathcal{H}^Q \times \mathcal{Y}$ is σ-multi-admissible if ℓ is σ-admissible with respect to any of his Q first arguments.*

This allows us to come up with the following theorem.

Theorem 6. *Let $\mathcal{H}$ be a reproducing kernel Hilbert space (with kernel k) such that $\forall X \in \mathcal{X}, k(X,X) \leq \kappa^2 < +\infty$. Let L be a loss matrix, such that $\forall q \in \mathcal{Y}$, ℓ_q is σ_q-multi-admissible. And let $\mathcal{A}$ be an algorithm such that*

$$\begin{aligned}\mathcal{A}_{\mathcal{S}} &= \operatorname*{argmin}_{h \in \mathcal{H}^Q} \sum_q \sum_{n:y_n=q} \frac{1}{m_q} \ell_q(h, x_n, q) + \lambda \sum_q \|h_q\|_k^2. \\ &:= \operatorname*{argmin}_{h \in \mathcal{H}^Q} J(h).\end{aligned}$$

Then $\mathcal{A}$ is confusion stable with respect to the set of loss functions $\boldsymbol{\ell}$. Moreover, a B value defining the stability is

$$B = \max_q \frac{\sigma_q^2 Q \kappa^2}{2\lambda},$$

where κ is such that $k(X,X) \leq \kappa^2 < +\infty$

Sketch of proof. In essence the idea is to exploit Definition 11 in order to apply Theorem 22 of [Bousquet and Elisseeff, 2002] for each loss ℓ_q. Moreover our regulariser is a sum (over q) of RKHS norms, hence the additional Q in the value of B. □ □

3.4.2 Lee, Lin and Wahba model

One of the most well-known and well-studied model for multi-class classification, in the context of SVM, was proposed by [Lee et al., 2004]. In this work, the authors suggest the use of the following loss function.

$$\ell(h, x, y) = \sum_{q \neq y} \left(h_q(x) + \frac{1}{Q-1} \right)_+$$

Their algorithm, denoted $\mathcal{A}^{\text{LLW}}$, then consists in minimising the following (penalised) functional,

$$J(h) = \frac{1}{m} \sum_{k=1}^{m} \sum_{q \neq y_k} \left(h_q(x_k) + \frac{1}{Q-1} \right)_+ + \lambda \sum_{q=1}^{Q} \|h_q\|^2,$$

with the constraint $\sum_q h_q = 0$.

We can trivially rewrite $J(h)$ as

$$J(h) = \sum_q \sum_{n:y_n=q} \frac{1}{m_q} \ell_q(h, x_n, q) + \lambda \sum_{q=1}^{Q} \|h_q\|^2,$$

with

$$\ell_q(h, x_n, q) = \sum_{p \neq q} \left(h_p(x_k) + \frac{1}{Q-1} \right)_+ .$$

It is straightforward that for any q, ℓ_q is 1-multi-admissible. We thus can apply theorem 6 and get $B = Q\kappa^2/2\lambda$.

Lemma 1. *Let h^* denote the solution found by $\mathcal{A}^{LLW}$. $\forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, \forall q$, we have*

$$\ell_q(h^*, x, y) \leq \frac{Q\kappa}{\sqrt{\lambda}} + 1.$$

Proof. As h^* is a minimiser of J, we have

$$J(h^*) \leq J(0) = \sum_q \sum_{n:y_n=q} \frac{1}{m_q} \ell_q(0, x_n, q) = \sum_q \sum_{n:y_n=q} \frac{1}{(Q-1)m_q} = 1.$$

As the data fitting term is non-negative, we also have

$$J(h^*) \geq \lambda \sum_q \|h_q^*\|_k^2.$$

Given that $h^* \in \mathcal{H}$, Cauchy-Schwarz inequality gives

$$\forall x \in \mathcal{X}, \|h_q^*\|_k \geq \frac{|h_q^*(x)|}{\kappa}.$$

Collecting things, we have

$$\forall x \in \mathcal{X}, |h_q^*(x)| \leq \frac{\kappa}{\sqrt{\lambda}}.$$

Going back to the definition of ℓ_q, we get the result. □ □

Using theorem 5, it follows that, with probability $1 - \delta$,

$$\left\| \widehat{\mathcal{C}}_{\boldsymbol{Y}}(\mathcal{A}^{\mathrm{LLW}}, \boldsymbol{X}) - \mathcal{C}_{\boldsymbol{s}(\boldsymbol{Y})}(\mathcal{A}^{\mathrm{LLW}}) \right\| \leq \sum_q \frac{Q\kappa^2}{\lambda m_q} + \frac{\sqrt{8 \ln\left(\frac{Q^2}{\delta}\right)} \left(\frac{2Q^2\kappa^2}{\lambda} + \left(\frac{Q\kappa}{\sqrt{\lambda}} + 1 \right) Q\sqrt{Q} \right)}{\sqrt{m^*}}.$$

3.4.3 Weston and Watkins model

Another multiclass mode is due to [Weston and Watkins, 1998]. They consider the following loss functions.

$$\ell(h, x, y) = \sum_{q \neq y} (1 - h_y(x) + h_q(x))_+$$

The algorithm $\mathcal{A}^{\mathrm{WW}}$ minimises the following functional

$$J(h) = \frac{1}{m}\sum_{k=1}^{m}\sum_{q\neq y_k}(1 - h_y(x) + h_q(x))_+ + \lambda\sum_{q<p=1}^{Q}\|h_q - h_p\|^2,$$

This time, for $1 \leq p, q \leq Q$, we will introduce the functions $h_{pq} = h_p - h_q$. We can then rewrite $J(h)$ as

$$J(h) = \sum_q \sum_{n:y_n=q}\frac{1}{m_q}\ell_q(h, x_n, q) + \lambda\sum_{p=1}^{Q}\sum_{q=1}^{p-1}\|h_{pq}\|^2,$$

with

$$\ell_q(h, x_n, q) = \sum_{p\neq q}(1 - h_{pq}(x_n))_+ .$$

It still is straightforward that for any q, ℓ_q is 1-multi-admissible. However, this time, our regulariser consists in the sum of $\frac{Q(Q-1)}{2} < \frac{Q^2}{2}$ norms. Applying Theorem 6 therefore gives $B = Q^2\kappa^2/4\lambda$.

Lemma 2. *Let h^* denote the solution found by $\mathcal{A}^{WW}$. $\forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, \forall q$, we have $\ell_q(h^*, x, y) \leq Q\left(1 + \kappa\sqrt{\frac{Q}{\lambda}}\right)$.*

This lemma can be proven following exactly the same techniques and reasoning as Lemma 1.

Using theorem 5, it follows that, with probability $1 - \delta$,

$$\left\|\widehat{\mathcal{C}}_{\boldsymbol{Y}}(\mathcal{A}^{\mathrm{WW}}, \boldsymbol{X}) - \mathcal{C}_{\boldsymbol{s}(\boldsymbol{Y})}(\mathcal{A}^{\mathrm{WW}})\right\| \leq \sum_q \frac{Q^2\kappa^2}{2\lambda m_q} + \frac{\sqrt{8\ln\left(\frac{Q^2}{\delta}\right)}\left(\frac{Q^3\kappa^2}{\lambda} + Q^2\left(\sqrt{Q} + \kappa\frac{Q}{\sqrt{\lambda}}\right)\right)}{\sqrt{m^*}}.$$

3.5 Discussion and Conclusion

In this chapter, we have proposed a new framework, namely the algorithmic *confusion stability*, together with new bounds to characterise the generalisation properties of multiclass learning algorithms. The crux of our study is to envision the confusion matrix as a performance measure, which differs from commonly encountered approaches that investigate generalisation properties of scalar-valued performances.

A few questions that are raised by the present work are the following. Is it possible to derive confusion stable algorithms that precisely aim at controlling the norm of their confusion matrix?

Are there other algorithms than those analysed here that may be studied in our new framework? On a broader perspective: how can noncommutative concentration inequalities be of help to analyse complex settings encountered in machine learning (such as, e.g., structured prediction, operator learning)?

Chapter 4

Stochastic Low-Rank Kernel Learning for Regression

Contents

4.1 Introduction **38**

4.2 Proposed Model **39**

4.2.1 Data-parameterised Kernels 39

4.2.2 Kernel Ridge Regression 40

4.2.3 A Convex Optimisation Problem 41

4.3 Solving the problem **43**

4.3.1 A Second-Order Stochastic Coordinate Descent 43

4.3.2 Iterative Updates 45

4.4 Analysis **47**

4.4.1 Pivotal Hyperparameter $\lambda\nu$ 48

4.4.2 Runtime Complexity and Memory Usage 50

4.4.3 Related Work 50

4.5 Numerical Simulations **52**

4.5.1 Setup 52

4.5.2 Influence of the parameters 53

4.5.2.1 Evolution of the objective 53

4.5.2.2 Zero-norm of $\boldsymbol{\mu}$ 53

4.5.3 Comparison to other methods 54

4.5.4 Larger-scale dataset 55

4.6 Conclusion and future work **56**

Abstract

We present a novel approach to learn a kernel-based regression function. It is based on the use of conical combinations of data-based parameterised kernels and on a new stochastic convex optimisation procedure of which we establish convergence guarantees. The overall learning procedure has the nice properties that a) the learned conical combination is automatically designed to perform the regression task at hand and b) the updates implicated by the optimisation procedure are inexpensive. In order to shed light on the appositeness of our learning strategy, we present empirical results from experiments conducted on various benchmark datasets.

4.1 Introduction

Our goal is to learn a kernel-based regression function, tackling at once two problems that commonly arise with kernel methods: working with a kernel tailored to the task at hand *and* efficiently handling problems whose size prevents the Gram matrix from being stored in memory. Though the present work focuses on regression, the material presented here might as well apply to classification.

Compared with similar methods, we introduce two novelties. Firstly, we build conical combinations of rank-1 Nyström approximations, whose weights are chosen so as to serve the regression task – this makes our approach different from [Kumar et al., 2009] and [Suykens et al., 2002], which focus on approximating the full Gram matrix with no concern for any specific learning task. Secondly, to solve the convex optimisation problem entailed by our modeling choice, we provide an original stochastic optimisation procedure based on [Nesterov, 2010]. It has the following characteristics: i) the computations of the updates are inexpensive (thanks to the designing choice of using rank-1 approximations) and ii) the convergence is guaranteed. To assess the practicality and effectiveness of our learning procedure, we conduct a few experiments on benchmark datasets, which allow us to draw positive conclusions on the relevance of our approach.

The chapter is organised as follows. Section 4.2 introduces some notation and our learning setting; in particular the optimisation problem we are interested in and the rank-1 parameterisation of the kernel our approach builds upon. Section 4.3 describes our new stochastic optimisation

procedure, establishes guarantees of convergence and details the computations to be implemented. Section 4.4 discusses the hyperparameters inherent to our modeling as well as the complexity of the proposed algorithm. Section 4.5 reports results from numerical simulations on benchmark datasets.

4.2 Proposed Model

Notation $\mathcal{X}$ is the input space, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denotes the (positive) kernel function we have at hand and $\phi : \mathcal{X} \to \mathcal{H}$ refers to the mapping $\phi(\boldsymbol{x}) := k(\boldsymbol{x}, \cdot)$ from $\mathcal{X}$ to the reproducing kernel Hilbert space $\mathcal{H}$ associated with k. Hence, $k(\boldsymbol{x}, \boldsymbol{x}')=\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle$, with $\langle \cdot, \cdot \rangle$ the inner product of $\mathcal{H}$.

The training set is $\mathcal{L} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathbb{R})^n$, where y_i is the target value associated to $\boldsymbol{x}_i$. $K = (k(\boldsymbol{x}_i, \boldsymbol{x}_j))_{1\leq i,j\leq n} \in \mathbb{R}^{n\times n}$ is the Gram matrix of k with respect to $\mathcal{L}$. For $m = 1, \ldots, n$, $\boldsymbol{c}_m \in \mathbb{R}^n$ is defined as:

$$\boldsymbol{c}_m := \frac{1}{\sqrt{k(\boldsymbol{x}_m, \boldsymbol{x}_m)}}[k(\boldsymbol{x}_1, \boldsymbol{x}_m), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x}_m)]^\top.$$

It is the m-th column of K rescaled by:

$$\sqrt{k(\boldsymbol{x}_m, \boldsymbol{x}_m)} = \|\phi(\boldsymbol{x}_m)\|.$$

4.2.1 Data-parameterised Kernels

For $m = 1, \ldots, n$, let $\tilde{\phi}_m : \mathcal{X} \to \tilde{\mathcal{H}}_m$ be the mapping:

$$\tilde{\phi}_m(\boldsymbol{x}) := \frac{\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}_m)\rangle}{k(\boldsymbol{x}_m, \boldsymbol{x}_m)}\phi(\boldsymbol{x}_m). \tag{4.1}$$

It directly follows that $\tilde{k}_m$ defined as, $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$,

$$\tilde{k}_m(\boldsymbol{x}, \boldsymbol{x}') := \langle \tilde{\phi}_m(\boldsymbol{x}), \tilde{\phi}_m(\boldsymbol{x}')\rangle = \frac{k(\boldsymbol{x}, \boldsymbol{x}_m)k(\boldsymbol{x}', \boldsymbol{x}_m)}{k(\boldsymbol{x}_m, \boldsymbol{x}_m)},$$

is indeed a positive kernel. Therefore, these parameterised kernels $\tilde{k}_m$ give rise to a family $(\tilde{K}_m)_{1\leq m\leq n}$ of Gram matrices of the following form:

$$\tilde{K}_m = (\tilde{k}_m(\boldsymbol{x}_i, \boldsymbol{x}_j))_{1\leq i,j\leq n} = \boldsymbol{c}_m\boldsymbol{c}_m^T, \tag{4.2}$$

which can be seen as rank-1 Nyström approximations of the full Gram matrix K [Drineas and Mahoney, 2005; Williams and Seeger, 2001].

As studied in Kumar et al. [2009], it is sensible to consider convex combinations of the $\tilde{K}_m$ to perform classification or regression if they are of very low rank. Building on this idea, we will investigate the use of a parameterised Gram matrix of the form:

$$\tilde{K}(\boldsymbol{\mu}) = \sum_{m \in \mathcal{S}} \mu_m \tilde{K}_m \quad \text{with} \quad \mu_m \geq 0, \tag{4.3}$$

where $\mathcal{S}$ is a set of indices corresponding to the specific rank-one approximations used. Note that since we consider *conical* combinations of the $\tilde{K}_m$, which are all positive semi-definite, $\tilde{K}(\boldsymbol{\mu})$ is positive semi-definite as well.

Using (4.1), one can show that the kernel $\tilde{k}_{\boldsymbol{\mu}}$, associated to our parameterised Gram matrix $\tilde{K}(\boldsymbol{\mu})$, is such that:

$$\tilde{k}_{\boldsymbol{\mu}}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_A = \phi(\boldsymbol{x})^\top A \phi(\boldsymbol{x}), \tag{4.4}$$

with

$$A := \sum_{m \in \mathcal{S}} \mu_m \frac{\phi(\boldsymbol{x}_m)\phi(\boldsymbol{x}_m)^\top}{k(\boldsymbol{x}_m, \boldsymbol{x}_m)}.$$

In other words, our parameterisation induces a modified metric in the feature space $\mathcal{H}$ associated to k. On a side note, remark that when $\mathcal{S} = \{1 \ldots, n\}$ (i.e. all the columns are picked) and we have uniform weights $\boldsymbol{\mu}$, then $\tilde{K}(\boldsymbol{\mu}) = KK^\top$, which is a matrix encountered when working with the so-called empirical kernel map [Schölkopf et al., 1999].

From now on, M denotes the size of $\mathcal{S}$ and m_0 refers to the number of non-zero components of $\boldsymbol{\mu}$ (i.e. it is the 0-pseudo-norm of $\boldsymbol{\mu}$).

4.2.2 Kernel Ridge Regression

Kernel Ridge regression (KRR) is the kernelised version of the popular ridge regression [Hoerl and Kennard, 1970] method. The associated optimisation problem reads:

$$\min_{\boldsymbol{w}} \left\{ \lambda \|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} \left(y_i - \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle \right)^2 \right\}, \tag{4.5}$$

where $\lambda > 0$ is a regularisation parameter.

Using I for the identity matrix, the following dual formulation may be considered:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \left\{ F_{KRR}(\boldsymbol{\alpha}) := \boldsymbol{y}^T\boldsymbol{\alpha} - \frac{1}{4\lambda}\boldsymbol{\alpha}^T(\lambda I + K)\boldsymbol{\alpha} \right\}. \tag{4.6}$$

The solution $\boldsymbol{\alpha}^*$ of the concave problem (4.6) and the optimal solution $\boldsymbol{w}^*$ of (4.5) are connected through the equality

$$\boldsymbol{w}^* = \frac{1}{2\lambda}\sum_{i=1}^{n} \alpha_i^* \phi(\boldsymbol{x}_i),$$

and $\boldsymbol{\alpha}^*$ can be found by setting the gradient of F_{KRR} to zero, to give

$$\boldsymbol{\alpha}^* = 2(I + \tfrac{1}{\lambda}K)^{-1}\boldsymbol{y}. \tag{4.7}$$

The value of the objective function at $\boldsymbol{\alpha}^*$ is then:

$$F_{KRR}(\boldsymbol{\alpha}^*) = \boldsymbol{y}^T(I + \tfrac{1}{\lambda}K)^{-1}\boldsymbol{y}, \tag{4.8}$$

and the resulting regression function is given by:

$$f(\boldsymbol{x}) = \frac{1}{2\lambda}\sum_{i=1}^{n} \alpha_i^* k(\boldsymbol{x}_i, \boldsymbol{x}). \tag{4.9}$$

4.2.3 A Convex Optimisation Problem

At a cost of $O(n^3)$ operations, KRR may be solved by solving the following linear system:

$$(I + \tfrac{K}{\lambda})\boldsymbol{\alpha} = 2\boldsymbol{y}.$$

Solving this linear system might be prohibitive for large n, even more so if the matrix $I + \frac{K}{\lambda}$ does not fit into memory. To cope with this issue problem, we work with $\tilde{K}(\boldsymbol{\mu})$ (4.3) instead of the Gram matrix K. As we shall see, this not only makes it possible to avoid memory issues but it also allows us to set up a learning problem where both $\boldsymbol{\mu}$ and a regression function are sought for at once. This is very similar to the Multiple Kernel Learning paradigm [Rakotomamonjy et al., 2008] where one learns an optimal kernel along with the target function.

To set up the optimisation problem we are interested in, we proceed in a way similar to [Rakotomamonjy et al., 2008]. For $m = 1, \ldots, n$, define the Hilbert space $\tilde{\mathcal{H}}'_m$ as:

$$\tilde{\mathcal{H}}'_m := \left\{ f \in \tilde{\mathcal{H}}_m \,\middle|\, \frac{\|f\|_{\tilde{\mathcal{H}}_m}}{\mu_m} < \infty \right\}. \tag{4.10}$$

One can prove [Aronszajn, 1950] that $\tilde{\mathcal{H}} = \bigoplus \tilde{\mathcal{H}}'_m$ is the RKHS associated to $\tilde{k} = \sum \mu_m \tilde{k}_m$. Mimicking the reasoning of [Rakotomamonjy et al., 2008], our primal optimisation problem reads:

$$\min_{\{f_m\},\boldsymbol{\mu}} \left\{ \lambda \sum_{m\in\mathcal{S}} \frac{1}{\mu_m} \|f_m\|^2_{\tilde{\mathcal{H}}'_m} + \sum_{i=1}^{n} (y_i - \sum_{m\in\mathcal{S}} f_m(\boldsymbol{x}_i))^2 \right\},$$
$$\text{s.t.} \sum_{m\in\mathcal{S}} \mu_m \leq n_1 \ , \quad \mu_m \geq 0, \tag{4.11}$$

where n_1 is a parameter controlling the 1-norm of $\boldsymbol{\mu}$. As this problem is also convex in $\boldsymbol{\mu}$, using the earlier results on the KRR problem, (4.11) is equivalent to:

$$\min_{\boldsymbol{\mu}} \left\{ \max_{\boldsymbol{\alpha}} \ \boldsymbol{y}^T \boldsymbol{\alpha} - \frac{1}{4\lambda} \boldsymbol{\alpha}^T (\lambda I + \tilde{K}(\boldsymbol{\mu})) \boldsymbol{\alpha} \right\}$$
$$\text{s.t.} \ \mu_i \geq 0, \ \forall i,$$

or,

$$\min_{\boldsymbol{\mu}} \left\{ \boldsymbol{y}^T (I + \tfrac{1}{\lambda} \tilde{K}(\boldsymbol{\mu}))^{-1} \boldsymbol{y} \right\} \tag{4.12}$$

$$\text{s.t.} \begin{cases} \sum_{m\in\mathcal{S}} \mu_m \leq n_1 \\ \mu_i \geq 0, \ \forall i \end{cases} \tag{4.13}$$

Finally, using the equivalence between Tikhonov and Ivanov regularisation formulations [Vasin, 1970], we obtain the following *convex* and *smooth* optimisation problem:

$$\min_{\boldsymbol{\mu}\geq 0} \left\{ F(\boldsymbol{\mu}) := \boldsymbol{y}^T (I + \tfrac{1}{\lambda} \tilde{K}(\boldsymbol{\mu}))^{-1} \boldsymbol{y} + \nu \sum_m \mu_m \right\}. \tag{4.14}$$

This is the problem we focus on.

The regression function $\tilde{f}$ is derived using (4.1), a minimiser $\boldsymbol{\mu}^*$ of the latter problem and the accompanying weight vector $\boldsymbol{\alpha}^*$ such that

$$\boldsymbol{\alpha}^* = 2 \left(I + \tfrac{1}{\lambda} \tilde{K}(\boldsymbol{\mu}^*) \right)^{-1} \boldsymbol{y}, \tag{4.15}$$

(obtained adapting (4.7) to the case $K = K(\boldsymbol{\mu}^*)$). We have:

$$\begin{aligned} \tilde{f}(\boldsymbol{x}) &= \frac{1}{2\lambda} \sum_{i=1}^{n} \alpha_i^* \tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}) \\ &= \frac{1}{2\lambda} \sum_{m\in\mathcal{S}} \mu_m^* \sum_{i=1}^{n} \alpha_i^* \tilde{k}_m(\boldsymbol{x}_i, \boldsymbol{x}) \\ &= \frac{1}{2\lambda} \sum_{m\in\mathcal{S}} \tilde{\alpha}_m^* k(\boldsymbol{x}_m, \boldsymbol{x}), \end{aligned} \tag{4.16}$$

where

$$\tilde{\alpha}_m^* := \mu_m^* \frac{\boldsymbol{c}_m^\top \boldsymbol{\alpha}^*}{\sqrt{k(\boldsymbol{x}_m, \boldsymbol{x}_m)}}. \tag{4.17}$$

4.3 Solving the problem

We now introduce a new stochastic optimisation procedure to solve (4.14). It implements a coordinate descent strategy with step sizes that use second-order information.

4.3.1 A Second-Order Stochastic Coordinate Descent

Problem (4.14) is a constrained minimisation based on the differentiable and convex objective function F. Usual convex optimisation methods (such as projected gradient descent, proximal methods [Beck and Teboulle, 2009b]) may be employed to solve this problem, but they may be too computationally expensive if n is very large, which is essentially due to a suboptimal exploitation of the parameterisation of the problem. Instead, the optimisation strategy we propose is specifically tailored to take advantage of the parameterisation of $\tilde{K}(\boldsymbol{\mu})$. For instance, as suggested by [Hsieh et al., 2008] or [Shalev-Schwartz and Tewari, 2009], coordinate descent methods have proven to be particularly efficient when dealing with high-dimensional problems.

Algorithm 2 depicts our stochastic descent method, inspired by [Nesterov, 2010]. At each iteration, a randomly chosen coordinate of $\boldsymbol{\mu}$ is updated via a Newton step. This method has two essential features: i) using coordinate-wise updates of $\boldsymbol{\mu}$ involves only partial derivatives which can be easily computed and ii) the stochastic approach ensures a reduced memory cost while still guaranteeing convergence.

Algorithm 2 Stochastic Coordinate Newton Descent

Input: $\boldsymbol{\mu}^0$ random.

repeat

Choose coordinate m_k uniformly at random in $\mathcal{S}$.

Update :

$$\mu_m^{k+1} = \mu_m^k \text{ if } m \neq m_k \text{ and}$$

$$\mu_{m_k}^{k+1} = \underset{v \geq 0}{\operatorname{argmin}} \frac{\partial F(\boldsymbol{\mu}^k)}{\partial \mu_{m_k}}(v - \mu_{m_k}^k) + \frac{1}{2}\frac{\partial^2 F(\boldsymbol{\mu}^k)}{\partial \mu_{m_k}^2}(v - \mu_{m_k}^k)^2, \tag{4.18}$$

until $F(\boldsymbol{\mu}^k) - F(\boldsymbol{\mu}^{k-M}) < \epsilon F(\boldsymbol{\mu}^{k-M})$

Notice that the Stochastic Coordinate Newton Descent (SCND) is similar to the algorithm proposed in [Nesterov, 2010], except that we replace the Lipschitz constants by the second-order partial derivatives $\frac{\partial^2 F(\mu^k)}{\partial \mu^2_{m_k}}$. Thus, we replace a constant step-size gradient descent by a Newton-step in (4.18), which allows us to make larger steps.

We show that for the function F in (4.14), SCND does provably converge to a minimiser of Problem (4.14). First, we rewrite (4.18) as a Newton step and compute the partial derivatives:

Proposition 2. *Eq.* (4.18) *is equivalent to*

$$\mu^{k+1}_{m_k} = \begin{cases} \left(\mu^k_{m_k} - \frac{\partial F(\mu^k)}{\partial \mu_{m_k}} / \frac{\partial^2 F(\boldsymbol{\mu}^k)}{\partial \mu^2_{m_k}}\right)_+ & \text{if } \frac{\partial^2 F(\boldsymbol{\mu}^k)}{\partial \mu^2_{m_k}} \neq 0 \\ 0 \text{ otherwise.} \end{cases} \tag{4.19}$$

Proof. (4.19) gives the optimality conditions for (4.18). □

Proposition 3. *The partial derivatives* $\frac{\partial^p F(\boldsymbol{\mu})}{\partial \mu^p_m}$ *are:*

$$\frac{\partial F(\boldsymbol{\mu})}{\partial \mu_m} = -\lambda(\boldsymbol{y}^\top \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}} \boldsymbol{c}_m)^2 + \nu, \tag{4.20}$$

$$\frac{\partial^p F(\boldsymbol{\mu})}{\partial \mu^p_m} = (-1)^p p! \lambda (\boldsymbol{y}^\top \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}} \boldsymbol{c}_m)^2 (\boldsymbol{c}_m^\top \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}} \boldsymbol{c}_m)^{p-1},$$

$$\text{with } p \geq 2 \text{ and } \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}} := (\lambda I + \tilde{K}(\boldsymbol{\mu}))^{-1}. \tag{4.21}$$

Proof. Easy but tedious calculations give the results. □

Theorem 7 (Convergence)**.** *For any sequence* $\{m_k\}_k$*, the sequence* $\{F(\boldsymbol{\mu}^k)\}_k$ *verifies:*

(a) $\lim_{k\to\infty} F(\boldsymbol{\mu}^k) = \min_{\boldsymbol{\mu}\geq 0} F(\boldsymbol{\mu})$.

Moreover, if there exists a minimiser $\boldsymbol{\mu}^*$ *of* F *such that the Hessian* $\nabla^2 F(\boldsymbol{\mu}^*)$ *is positive definite then:*

(b) $\boldsymbol{\mu}^*$ *is the unique minimiser of* F*. The sequence* $\{\boldsymbol{\mu}^k\}$ *converges to* $\boldsymbol{\mu}^*$*:* $||\boldsymbol{\mu}^k - \boldsymbol{\mu}^*|| \to 0$.

Sketch of proof. In order to establish a proof of the theorem it is handy to introduce the mapping $q_{\boldsymbol{\mu},m} : \mathbb{R} \to \mathbb{R}$ of one variable defined as:

$$q_{\boldsymbol{\mu},m}(v) := F(\boldsymbol{\mu}^{\setminus m} + v\boldsymbol{e}_m),$$

where $\boldsymbol{\mu}^{\setminus m}$ is equal to $\boldsymbol{\mu}$ except for its m-th component that is set to zero and $\boldsymbol{e}_m \in \mathbb{R}^n$ is the canonical basis n-dimensional vector that has a 1 in its m-th component (and zero on the others).

The Taylor series expansion of q might be computed to any order, because F can be differentiated to any order. In particular, we have that:

$$\begin{aligned} q_{\boldsymbol{\mu},m}(v) = q_{\boldsymbol{\mu},m}(\mu_m) &+ q'_{\boldsymbol{\mu},m}(\mu_m)(v-\mu_m) \\ &+ \frac{1}{2} q''_{\boldsymbol{\mu},m}(\mu_m)(v-\mu_m)^2 \\ &+ \frac{1}{2}\int_{\mu_m}^{v} (t-\mu_m)^2 q^{(3)}_{\boldsymbol{\mu},m}(t)dt, \end{aligned} \tag{4.22}$$

where, obviously,

$$q^{(r)}_{\boldsymbol{\mu},m}(t) = \frac{\partial^r F(\boldsymbol{\mu}^{\setminus m} + t\boldsymbol{e}_m)}{\partial \mu_m^r}$$

(a) We may truncate the Taylor expansion of $q_{\boldsymbol{\mu},m}$ to the second order, to get the following quadratic form

$$Q_{\boldsymbol{\mu},m}(v) := F(\boldsymbol{\mu}) + \tfrac{\partial F(\boldsymbol{\mu})}{\partial \mu_m}(v-\mu_m) + \tfrac{1}{2}\tfrac{\partial^2 F(\boldsymbol{\mu})}{\partial \mu_m^2}(v-\mu_m)^2,$$

which matches F and ∇F at $\boldsymbol{\mu}$ (for any fixed m and $\boldsymbol{\mu}$). Using that $\frac{\partial^3 F(\boldsymbol{\mu})}{\partial \mu_m^3} \leq 0$ (see (4.20)) and taking into account the form of the integral remainder in (4.22), it comes that

$$\begin{cases} Q_{\boldsymbol{\mu},m}(v) \leq F(\boldsymbol{\mu}^{\setminus m} + v\boldsymbol{e}_m) & \text{if } 0 \leq v \leq \mu_m, \\ Q_{\boldsymbol{\mu},m}(v) \geq F(\boldsymbol{\mu}^{\setminus m} + v\boldsymbol{e}_m) & \text{if } v \geq \mu_m. \end{cases}$$

This means that this quadratic function is an upper bound of F for $v \geq \mu_m$ and a lower bound for $v \leq \mu_m$. Thus F may be handled as a function with an L-Lipschitz gradient when $\frac{\partial F(\boldsymbol{\mu})}{\partial \mu_m} < 0$ and as a strongly convex function when $\frac{\partial F(\boldsymbol{\mu})}{\partial \mu_m} > 0$. Combining the bounds obtained in both cases (see e.g. Nesterov [2004]) ensures the convergence of the algorithm.

(b) is standard in convex optimisation.

□

4.3.2 Iterative Updates

One may notice from (4.20) that the computations of the derivatives, as well as the computation of $\boldsymbol{\alpha}^*$, depend on $\tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}}$. Moreover, the dependency in $\boldsymbol{\mu}$, for all those quantities, only appears in $\tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}}$. Thus, special care need be taken on how $\tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}}$ is stored and updated throughout.

Let $\mathcal{S}^+_{\boldsymbol{\mu}} = \{m \in \mathcal{S} | \mu_m > 0\}$ and $m_0 = \|\boldsymbol{\mu}\|_0 = |\mathcal{S}^+_{\boldsymbol{\mu}}|$. Let $C = [\boldsymbol{c}_{i_1} \cdots \boldsymbol{c}_{i_{m_0}}] \in \mathbb{R}^{n \times m_0}$ be the concatenation of the $\boldsymbol{c}_{i_j}$'s, for $i_j \in \mathcal{S}^+_{\boldsymbol{\mu}}$ and D the diagonal matrix with diagonal elements μ_{i_j}, for

$i_j \in \mathcal{S}_{\boldsymbol{\mu}}^+$. Remark that throughout the iterations the sizes of C and D may vary. Given (4.21) and using Woodbury formula (Theorem 8, Appendix), we have:

$$\tilde{K}_{\lambda,\boldsymbol{\mu}}^{-1} = \left(\lambda I + CDC^\top\right)^{-1} = \frac{1}{\lambda} I - \frac{1}{\lambda^2} CGC^\top, \tag{4.23}$$

with

$$G := \left(D^{-1} + \frac{1}{\lambda} C^\top C\right)^{-1}. \tag{4.24}$$

Note that G is a square matrix of order m_0 and that an update on $\boldsymbol{\mu}$ will require an update on G. Even though updating $D^{-1} + \frac{1}{\lambda} C^\top C = G^{-1}$, is trivial, we may prefer to directly store and update G. This is what we describe now.

At each iteration, only one coordinate of $\boldsymbol{\mu}$ is updated. Let p be the index of the updated coordinate, $\boldsymbol{\mu}_{\text{old}}$, C_{old}, D_{old} and G_{old}, the vectors and matrices before the update and $\boldsymbol{\mu}_{\text{new}}$, C_{new}, D_{new} and G_{new} the updated matrices/vectors. Let also $\boldsymbol{e}_p$ be the vector whose pth coordinate is 1 while other coordinates are 0. We encounter four different cases.

Case 1: $\mu_p^{\text{old}} = 0$ and $\mu_p^{\text{new}} = 0$. No update needed:

$$\begin{aligned} G_{\text{new}} &= G_{\text{old}}, \\ C_{\text{new}} &= C_{\text{old}}. \end{aligned} \tag{4.25}$$

Case 2: $\mu_p^{\text{old}} \neq 0$ and $\mu_p^{\text{new}} \neq 0$.

$$C_{\text{new}} = C_{\text{old}},$$

and

$$D_{\text{new}}^{-1} = D_{\text{old}}^{-1} + \Delta_p \boldsymbol{e}_p \boldsymbol{e}_p^\top,$$

where,

$$\Delta_p := \frac{1}{\mu_p^{\text{new}}} - \frac{1}{\mu_p^{\text{old}}}.$$

Then, using Woodbury formula, we have:

$$G_{\text{new}} = \left(G_{\text{old}}^{-1} + \Delta_p \boldsymbol{e}_p \boldsymbol{e}_p^\top\right)^{-1} = G_{\text{old}} - \frac{\Delta_p}{1 + \Delta_p g_{pp}} \boldsymbol{g}_p \boldsymbol{g}_p^\top, \tag{4.26}$$

with g_{pp} the (p,p)th entry of G_{old} and $\boldsymbol{g}_p$ its pth column.

Case 3: $\mu_p^{\mathbf{old}} \neq 0$ and $\mu_p^{\mathbf{new}} = 0$. Here, $\mathcal{S}^+_{\boldsymbol{\mu}_{\text{new}}} = \mathcal{S}^+_{\boldsymbol{\mu}_{\text{old}}} \setminus \{p\}$. It follows that we have to remove $\boldsymbol{c}_p$ from C_{old} to get C_{new}. To compute G_{new}, we may consider the previous update formula when $\mu_p^{\text{new}} \to 0$ (that is, when $\Delta_p \to +\infty$). Note that we can use the previous formula because $\mu_p \mapsto \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}}$ is well-defined and continuous at 0. Thus, as

$$\lim_{\mu_p^{\text{new}} \to 0} \frac{\Delta_p}{1 + \Delta_p g_{pp}} = \frac{1}{g_{pp}},$$

we have:

$$G_{\text{new}} = \left(G_{\text{old}} - \frac{1}{g_{pp}} \boldsymbol{g}_p \boldsymbol{g}_p^\top \right)_{\setminus\{p\}}, \tag{4.27}$$

where $A_{\setminus\{p\}}$ denotes the matrix A from which the pth column and pth row have been removed.

Case 4: $\mu_p^{\mathbf{old}} = 0$ and $\mu_p^{\mathbf{new}} \neq 0$. We have $C_{\text{new}} = [C_{\text{old}}\ \boldsymbol{c}_p]$. Using (4.24), it follows that

$$\begin{aligned}
G_{\text{new}} &= \begin{pmatrix} D_{\text{old}}^{-1} + \frac{1}{\lambda} C_{\text{old}}^\top C_{\text{old}} & \frac{1}{\lambda} C_{\text{old}}^\top \boldsymbol{c}_p \\ \frac{1}{\lambda} \boldsymbol{c}_p^\top C_{\text{old}} & \frac{1}{\mu_p^{\text{new}}} + \frac{1}{\lambda} \boldsymbol{c}_p^\top \boldsymbol{c}_p \end{pmatrix}^{-1} \\
&= \begin{pmatrix} G_{\text{old}}^{-1} & \frac{1}{\lambda} C_{\text{old}}^\top \boldsymbol{c}_p \\ \frac{1}{\lambda} \boldsymbol{c}_p^\top C_{\text{old}} & \frac{1}{\mu_p^{\text{new}}} + \frac{1}{\lambda} \boldsymbol{c}_p^\top \boldsymbol{c}_p \end{pmatrix}^{-1} \\
&= \begin{pmatrix} A & \boldsymbol{v} \\ \boldsymbol{v}^\top & s \end{pmatrix},
\end{aligned}$$

where, using the block-matrix inversion formula of Theorem 9 (Appendix), we have:

$$\begin{aligned}
s &= \left(\frac{1}{\mu_p^{\text{new}}} + \frac{1}{\lambda} \boldsymbol{c}_p^\top \boldsymbol{c}_p - \frac{1}{\lambda^2} \boldsymbol{c}_p^\top C_{\text{old}} G_{\text{old}} C_{\text{old}}^\top \boldsymbol{c}_p \right)^{-1} \\
\boldsymbol{v} &= -\frac{s}{\lambda} G_{\text{old}} C_{\text{old}}^\top \boldsymbol{c}_p \\
\boldsymbol{A} &= G_{\text{old}} + \frac{1}{s} \boldsymbol{v}\boldsymbol{v}^\top.
\end{aligned} \tag{4.28}$$

Complete learning algorithm. Algorithm 3 depicts the full Stochastic Low-Rank Kernel Learning algorithm (SLKL), which recollects all the pieces just described.

4.4 Analysis

Here, we discuss the relation between λ and ν and we argue that there is no need to keep both hyperparameters. In addition, we provide a short analysis on the runtime complexity of our learning procedure.

Algorithm 3 SLKL: Stochastic Low-Rank Kernel Learning

inputs: $\mathcal{L} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, $\nu > 0$, $M > 0$, $\epsilon > 0$.

outputs: $\boldsymbol{\mu}$, G and C (yield $(\lambda I + K(\mu))^{-1}$ from (4.23)).

initialisation: $\boldsymbol{\mu}^{(0)} = \mathbf{0}$.

repeat

Choose coordinate m_k uniformly at random in $\mathcal{S}$.

Update $\boldsymbol{\mu}^{(k)}$ according to (4.19), by changing only the m_k-th coordinate $\mu^k_{m_k}$ of $\boldsymbol{\mu}^{(k)}$:

- compute the second order derivative

$$h = \lambda(\boldsymbol{y}^\top \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}} \boldsymbol{c}_{m_k})^2 (\boldsymbol{c}^\top_{m_k} \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}} \boldsymbol{c}_{m_k}) \ ;$$

- **if** $h > 0$ **then**

$$\mu^{(k+1)}_{m_k} = \max\left(0, \mu^{(k)}_{m_k} + \frac{\lambda(\boldsymbol{y}^\top \tilde{K}^{-1}_{\lambda,\boldsymbol{\mu}} \boldsymbol{c}_{m_k})^2 - \nu}{h}\right);$$

else $\mu^{(k+1)}_{m_k} = 0$.

Update $G^{(k)}$ and $C^{(k)}$ according to (4.25)-(4.28).

until $F(\boldsymbol{\mu}^k) - F(\boldsymbol{\mu}^{k-M}) < \epsilon F(\boldsymbol{\mu}^{k-M})$

4.4.1 Pivotal Hyperparameter $\lambda\nu$

First recall that we are interested in the minimiser $\boldsymbol{\mu}^*_{\lambda,\nu}$ of constrained optimisation problem (4.14), i.e.:

$$\boldsymbol{\mu}^*_{\lambda,\nu} = \operatorname*{argmin}_{\boldsymbol{\mu}\geq 0} F_{\lambda,\nu}(\boldsymbol{\mu}), \tag{4.29}$$

where, for the sake of clarity, we purposely show the dependence on λ and ν of the objective function $F_{\lambda,\nu}$

$$F_{\lambda,\nu}(\boldsymbol{\mu}) = \boldsymbol{y}^\top \left(I + \tilde{K}\left(\tfrac{\boldsymbol{\mu}}{\lambda}\right)\right)^{-1} \boldsymbol{y} + \lambda\nu \sum_m \tfrac{\mu_m}{\lambda}, \tag{4.30}$$

We may call $\boldsymbol{\alpha}^*_{\lambda,\nu}$, $\tilde{\boldsymbol{\alpha}}^*_{\lambda,\nu}$ the weight vectors associated with $\boldsymbol{\mu}^*_{\lambda,\nu}$ (see (4.15) and (4.17)). We have the following:

Proposition 4. *Let $\lambda, \nu, \lambda', \nu'$ be strictly positive real numbers. If $\lambda\nu = \lambda'\nu'$ then*

$$\boldsymbol{\mu}^*_{\lambda',\nu'} = \tfrac{\lambda'}{\lambda} \boldsymbol{\mu}^*_{\lambda,\nu},$$

and

$$\tilde{f}_{\lambda,\nu} = \tilde{f}_{\lambda',\nu'}.$$

As a direct consequence:

$$\forall \lambda, \nu \geq 0,\ \tilde{f}_{\lambda,\nu} = \tilde{f}_{1,\lambda\nu}.$$

Proof. Suppose that we know $\boldsymbol{\mu}^*_{\lambda,\nu}$. Given the definition (4.30) of $F_{\lambda,\nu}$ and using $\lambda\nu = \lambda'\nu'$, we have

$$F_{\lambda,\nu}(\boldsymbol{\mu}) = F_{\lambda',\nu'}\left(\tfrac{\lambda'}{\lambda}\boldsymbol{\mu}\right)$$

Since the only constraint of problem (4.29) is the nonnegativity of the components of $\boldsymbol{\mu}$, it directly follows that $\lambda'\boldsymbol{\mu}^*_{\lambda,\nu}/\lambda$ is a minimiser of $F_{\lambda',\nu'}$ (under these constraints), hence $\boldsymbol{\mu}^*_{\lambda',\nu'} = \lambda'\boldsymbol{\mu}^*_{\lambda,\nu}/\lambda$.

To show $\tilde{f}_{\lambda,\nu} = \tilde{f}_{\lambda',\nu'}$, it suffices to observe that, according to the way $\boldsymbol{\alpha}^*_{\lambda,\nu}$ is defined (cf. (4.15)),

$$\begin{aligned}\boldsymbol{\alpha}^*_{\lambda',\nu'} &= 2\left(I + K\left(\tfrac{\boldsymbol{\mu}^*_{\lambda',\nu'}}{\lambda'}\right)\right)^{-1}\boldsymbol{y}\\ &= 2\left(I + K\left(\tfrac{\lambda'}{\lambda}\tfrac{\boldsymbol{\mu}^*_{\lambda,\nu}}{\lambda'}\right)\right)^{-1}\boldsymbol{y} = \boldsymbol{\alpha}^*_{\lambda,\nu},\end{aligned}$$

and, thus, $\tilde{\boldsymbol{\alpha}}^*_{\lambda',\nu'} = \lambda'\tilde{\boldsymbol{\alpha}}^*_{\lambda,\nu}/\lambda$. The definition (4.16) of $\tilde{f}_{\lambda,\nu}$ then gives $\tilde{f}_{\lambda,\nu} = \tilde{f}_{\lambda',\nu'}$, which entails $\tilde{f}_{\lambda,\nu} = \tilde{f}_{1,\lambda\nu}$. □

This proposition has two nice consequences.

- First, it says that the pivotal hyperparameter is actually the product $\lambda\nu$: this *is* the quantity that parameterises the learning problem (not λ or ν, seen independently). Thus, the set of regression functions, defined by the λ and ν hyperparameter space, can be described by exploring the set of vectors $(\boldsymbol{\mu}^*_{1,\nu})_{\nu>0}$, which only depends on a single parameter.

- Second, considering $(\boldsymbol{\mu}^*_{1,\nu})_{\nu>0}$ allows us to work with the family of objective functions $(F_{1,\nu})_{\nu>0}$, which are well-conditioned numerically as the hyperparameter λ is set to 1. This is especially beneficial as the use of the Woodbury formula is known to induce numerical instability in the algorithms, when applied on ill-conditioned matrices.

4.4.2 Runtime Complexity and Memory Usage

For the present analysis, let us assume that we pre-compute the M (randomly) selected columns $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M$. If a is the cost of computing a column $\boldsymbol{c}_m$, the pre-computation has a cost of $O(Ma)$ and has a memory usage of $O(nM)$, as each column is of size n, the number of data.

At each iteration, we have to compute the first and second-order derivatives of the objective function, as well as its value and the weight vector $\boldsymbol{\alpha}$. Using (4.23), (4.20), (4.14) and (4.15), one can show that those operations have a complexity of $O(nm_0)$ if m_0 is the zero-norm of $\boldsymbol{\mu}$.

Besides, in addition to C, we need to store G for a memory cost of $O(m_0^2)$. Overall, if we denote the number of iterations by k, the algorithm has a memory cost of $O(nM + m_0^2)$ and a runtime complexity of $O(knm_0 + Ma)$.

If memory is a critical issue, one may prefer to compute the columns $\boldsymbol{c}_m$ on-the-fly and m_0 columns need to be stored instead of M (this might be a substantial saving in terms of memory as can be seen in the next section). This improvement in term of memory usage implies an additive cost in the runtime complexity. In the worst case, we have to compute a new column $\boldsymbol{c}$ at each iteration. The resulting memory requirement scales as $O(nm_0 + m_0^2)$ and the runtime complexity varies as $O(k(nm_0 + a))$.

4.4.3 Related Work

Lying at the intersection between (Multiple) Kernel Learning, Low-Rank Approximation and Sparse Modelling more generally, our work has connections with many existing related works. Of course, we will not be able to provide an extensive review of all of them, but we may focus on the specificity of this study and highlight the main differences with the most closely related approaches.

First of all, when it comes to learning a kernel, one may distinguish two types of approaches. The first one consists in separating the kernel learning phase from the machine learning task at hand. Those methods are sometimes referred as "two-stage" procedures (e.g. [Cortes et al., 2010]) or, in the more general setting of feature selection, as "filter methods". A prominent line of work in this family is based on "kernel-target alignment" [Cristianini et al., 2001] where one is trying to align the designed kernel to an ideal one [Crammer et al., 2002], taking into account the label of the input data. As stated in the introduction, our parameterisation of the

kernel is a particular instance of the one used in [Kumar et al., 2009], where they consider the combination of low-rank (but not necessarily rank-one) kernels. But the main difference lies in the fact that instead of considering the learning task at hand, their method focuses on providing a good low-rank approximation of a base (higher-rank) kernel. It is also worth mentioning a recent work [Zhang et al., 2012] where the Nyström approach is used to design a low-rank kernel, along with a term penalising low values of the kernel-target alignment in order to make use of the label information and ensure that the learnt kernel will be well-aligned with the "ideal" one.

The other family of methods, on the other hand, is referred to as "one-stage" procedures, or "embedded methods". Instead of separating the learning of the kernel (or data representation) and the actual machine learning task, those methods try to tackle the two problems at once. Along with most Multiple Kernel Learning algorithms [Rakotomamonjy et al., 2008; Kloft et al., 2010], our work falls into this category. Though the effectiveness of these approaches is often discussed [Cortes et al., 2010], the main idea is to learn an optimal kernel with respect to the task at hand.

From algorithmic point of view, many related work would be worth mentioning. Our problem (Eq. 4.14) is a convex, smooth, and constrained problem. Dropping the positivity constraint, it could be cast as a well-known non-smooth but unconstrained problem as follows:

$$\min_{\boldsymbol{\mu}} \boldsymbol{y}^T (I + \tfrac{1}{\lambda}\tilde{K}(\boldsymbol{\mu}))^{-1}\boldsymbol{y} + \nu\|\boldsymbol{\mu}\|_1. \tag{4.31}$$

Due to the popularity of the Lasso [Tibshirani, 1996], involving such an L_1-norm regularisation term, many algorithms have been specifically designed. Among them, the Least Angle Regression (LARS) algorithm [Efron et al., 2004] could have been considered to be adapted to our problem. However, as mentionned earlier about proximal-gradient based approaches, such an algorithm would require the storage of all the columns of the Gram matrix at a time, which can make it prohibitive for large-scale problems. Moreover, dropping the positivity constraint could result in learning a non-positive semi-definite kernel. If so, the representer theorem does not hold anymore and the solution of our algorithm is not guaranteed to be the optimal of the original primal problem (Eq. 4.11).

Finally, it has been reported [Cortes et al., 2009] that using L_2 regularisation instead of L_1 could improve the performance of Multiple Kernel Learning algorithms. However, this possible improvement comes at the cost of losing the sparsity of the solution. In our problem, the

relevance of our algorithm specifically relies on this sparsity property in order not to have to store a potentially huge number of columns of the Gram matrix. Although the design of good regularisers for solving ill-posed problems still remains a complex issue, the only options we could consider are the ones inducing sparsity of the solution.

4.5 Numerical Simulations

We now present results from various numerical experiments, for which we describe the datasets and the protocol used. We study the influence of the different parameters of our learning approach on the results and compare the performance of our algorithm to that of related methods.

4.5.1 Setup

First, we use a toy dataset (denoted by *sinc*) to better understand the role and influence of the parameters. It consists in regressing the cardinal sine of the two-norm (i.e. $\boldsymbol{x} \mapsto \sin(\|\boldsymbol{x}\|)/\|\boldsymbol{x}\|$) of random two-dimensional points, each drawn uniformly between -5 and $+5$. In order to have a better idea on how the solutions may or may not over-fit the training data, we add some white Gaussian noise on the target variable of the randomly picked 1000 training points (with a 10 dB signal-to-noise ratio). The test set is made of 1000 non-noisy independent instance/target pairs.

We then assess our method on two UCI datasets: Abalone (*abalone*) and Boston Housing (*boston*), using the same normalisations, Gaussian kernel parameters (σ denotes the kernel width) and data partition as in [Smola and Schölkopf, 2000]. The United States Postal Service (*USPS*) dataset is used with the same setting as in [Williams and Seeger, 2001]. Finally, the Modified National Institute of Standards and Technology (*MNIST*) dataset is used with the same pre-processing as in [Maji and Malik, 2009]. Table 4.1 summarises the characteristics of all the datasets we used.

As displayed in Algorithm 2, at each iteration $k > M$, we check if $F(\boldsymbol{\mu}^k) - F(\boldsymbol{\mu}^{k-M}) < \epsilon F(\boldsymbol{\mu}^{k-M})$ holds. If so, we stop the optimisation process. ϵ thus controls our stopping criterion. In the experiments, we set $\epsilon = 10^{-4}$ unless otherwise stated and we set λ to 1 for all the experiments and we run simulations for various values of ν and M. In order to assess the variability incurred by the stochastic nature of our learning algorithm, we run each experiment 20 times.

Table 4.1: Datasets used for the experiments.

dataset	**#features**	**#train** (n)	**#test**	$\boldsymbol{\sigma}^2$
sinc	2	1000	1000	1
abalone	10	3000	1177	2.5
boston	13	350	156	3.25
USPS	256	7291	2007	64
MNIST	2172	60000	10000	4

4.5.2 Influence of the parameters

4.5.2.1 Evolution of the objective

We have established (Section 4.3) the convergence of our optimisation procedure, under mild conditions. A question that we have not tackled yet is to evaluate its convergence rate. Figure 4.1 plots the evolution of the objective function on the *sinc* dataset. We observe that the evolutions of the objective function are impressively similar among the different runs. This empirically tends to assert that it is relevant to look for theoretical results on the convergence rate.

A question left for future work is the impact of the random selection of the set of columns $\mathcal{S}$ on the reached solution.

4.5.2.2 Zero-norm of $\boldsymbol{\mu}$

As shown in Section 4.4.2, both memory usage and the complexity of the algorithm depend on m_0. Thus, it is interesting to take a closer look at how this quantity evolves. Figure 4.2 and 4.3 experimentally point out two things. On the one hand, the number of active components $m_0 = \|\boldsymbol{\mu}\|_0$ remains significantly smaller than M. In other words, as long as the regularisation parameter is well-chosen, we never have to store all of the $\boldsymbol{c}_m$ at the same time. On the other hand, the solution $\boldsymbol{\mu}^*$ is sparse and $\|\boldsymbol{\mu}^*\|_0$ grows with M and diminishes with ν. A theoretical study on the dependence of $\boldsymbol{\mu}^*$ and m_0 in M and ν, left for future work, would be all the more interesting since sparsity is the cornerstone on which the scalability of our algorithm depends.

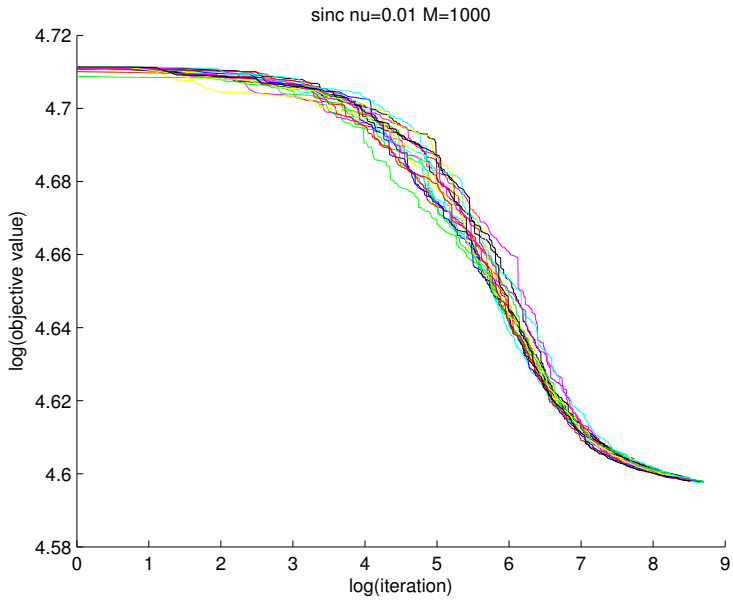

Figure 4.1: Evolution of the objective during the optimisation process for the *sinc* dataset with $\nu = 0.01$, $M = 1000$ (for 20 runs).

4.5.3 Comparison to other methods

This section aims at giving a hint on how our method performs on regression tasks. To do so, we compare the Mean Square Error (over the test set). In addition to our Stochastic Low-Rank Kernel Learning method (*SLKL*), we solve the problem with the standard Kernel Ridge Regression method, using the n training data (*KRRn*) and using only M training data (*KRRM*). We also evaluate the performance of the KRR method, using the kernel obtained with uniform weights on the M rank-1 approximations selected for *SLKL* (*Unif*). The results are displayed in Table 4.2, where the bold font indicates the best low-rank method (*KRRM, Unif* or *SLKL*) for each experiment.

Table 4.2 confirms that optimising the weight vector $\boldsymbol{\mu}$ is decisive as our results dramatically outperform those of *Unif*. As long as $M < n$, our method also outperforms *KRRM*. The explanation probably lies in the fact that our approximations keep information about similarities between the M selected points and the $n - M$ others. Furthermore, our method *SLKL* achieves comparable performances (or even better on *abalone*) than *KRRn*, while finding sparse solutions. Compared to the approach from [Smola and Schölkopf, 2000], we seem to achieve lower test error

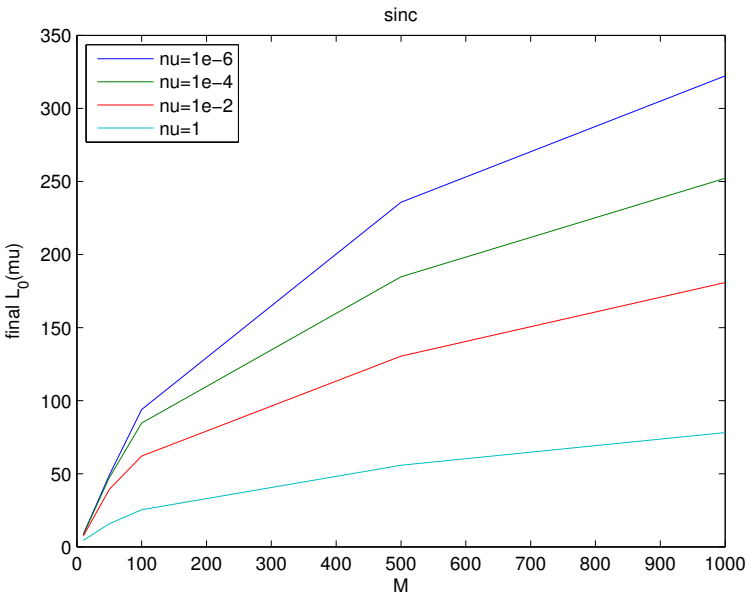

Figure 4.2: Zero-norm of the optimal $\boldsymbol{\mu}^*$ as a function of M for different values of ν for the *sinc* dataset (averaged on 20 runs).

on the *boston* dataset even for $M = 128$. On the *abalone* dataset, this method outperforms ours for every M we tried.

Finally, we also compare the results we obtain on the *USPS* dataset with the ones obtained in [Williams and Seeger, 2001] (*Nyst*). As it consists in a classification task, we actually perform a regression on the labels to adapt our method, which is known to be equivalent to solving Fisher Discriminant Analysis [Duda and Hart, 1973]. The performance achieved by *Nyst* outperforms ours. However, one may argue that the performance have a same order of magnitude and note that the *Nyst* approach focuses on the classification task, while ours was designed for regression.

4.5.4 Larger-scale dataset

To assess the scalability of our method, we ran experiments on the larger handwritten digits *MNIST* dataset, whose training set is made of 60000 examples. We used a Gaussian kernel computed over histograms of oriented gradients as in [Maji and Malik, 2009], in a "one versus all" setting. For $M = 1000$, we obtained classification error rates around 2% over the test set, which do not compete with state-of-the-art results but achieve reasonable performance, considering

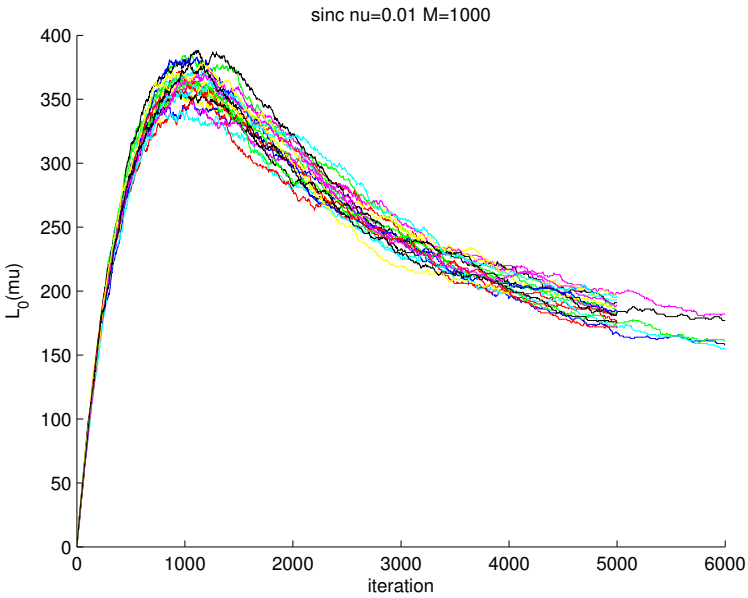

Figure 4.3: Evolution of the zero-norm of $\boldsymbol{\mu}$ (m_0) with the iterations for the *sinc* dataset with $\nu = 0.01$, $M = 1000$ (20 runs).

that we use only a small part of the data (cf. the size of M) and that our method was designed for regression.

Although our method overcomes memory usage issues for such large-scale problems, it still is computationally intensive. In fact, a large number of iterations is spent picking coordinates whose associated weight remains at 0. Though those iterations do not induce any update, they do require computing the associated Gram matrix column (which is not stored as it does not weigh in the conic combination) as well as the derivatives of the objective function.

4.6 Conclusion and future work

We have presented an original kernel-based learning procedure for regression. The main features of our contribution are the use of a conical combination of data-based kernels and the derivation of a stochastic convex optimisation procedure, that acts coordinate-wise and makes use of second-order information. We provide theoretical convergence guarantees for this optimisation procedure, we depict the behavior of our learning procedure and illustrate its effectiveness through a number of numerical experiments carried out on several benchmark datasets.

Table 4.2: Mean square error with standard deviation measured on three regression tasks.

	sinc			*boston*			*abalone*		
M	256	512	1000	128	256	350	512	1024	3000
KRRn	0.009 ± 09			10.17 ± 0			6.91 ± 0		
KRRM	0.0146 $\pm 1e^{-3}$	0.0124 $\pm 7e^{-4}$	**0.0099** ± 0	33.27 ± 7.8	16.89 ± 3.27	**10.17** ± 0	6.14 ± 0.25	5.51 ± 0.09	5.25 ± 0
Unif	0.0124 $\pm 1e^{-4}$	0.0124 $\pm 3e^{-5}$	0.0124 ± 0	149.7 ± 5.57	147.84 ± 2.24	147.72 ± 0	10.04 ± 0.17	9.96 ± 0.06	9.99 ± 0
SLKL	**0.0106** $\pm 4e^{-4}$	**0.0103** $\pm 2e^{-4}$	0.0104 $\pm 1e^{-4}$	**20.17** ± 2.3	**13.1** ± 0.87	11.43 ± 0.06	**5.04** ± 0.08	**4.94** ± 0.03	**4.95** ± 0.004
m_0	83	108	139	108	161	184	159	191	253

Table 4.3: Number of errors and standard deviation on the test set (2007 examples) of the USPS dataset.

M	64	256	1024
Nyst	101.3 ± 22.9	34.5 ± 3.0	35.9 ± 2.0
SLKL	76.3 ± 9.9	47.6 ± 3.1	41.5 ± 3.9
m_0	61	210	515

The present work naturally raises several questions and perspectives for future work. For better computational performances, several strategies could be considered. For instance, as we last pointed out, many iterations are wasted optimising on coordinates that are "stuck" on the constraint (i.e. coordinates whose value is zero). To avoid those computations, using techniques such as shrinkage [Hsieh et al., 2008], discarding those coordinates from the set among which we draw at random at the beginning of each iteration, has proven, in a different context, to achieve drastic improvements computation-wise.

On a slightly broader perspective, we had discarded full-gradient-based approaches because of their high memory usage in the context of large-scale problems. To overcome this problem, active-set strategies, such as [Kowalski et al., 2011], could also be considered, taking into account only as many coordinates at a time as available memory allows.

Considering other strategies than our purely stochastic one would also be worth the effort. For

instance, greedy approaches could also be investigated in order to optimise, at each step, along the "best" coordinate instead of a random one. Naturally, with such approaches, the efficiency of the optimisation comes at the cost of a potentially computationally intensive selection of the coordinate at each iteration. This being said, digging into the matching pursuit [Mallat and Zhang, 1993] literature, one can find many intermediate strategies, see [Peel et al., 2012] for instance, exploring the trade-off between a cheap (but inefficient) and optimal (but costly) selection. A deep study of this trade-off from both practical and theoretical point of views would be of great interest, far beyond the scope of our work.

Building on the work developped in [Kumar et al., 2009], we used a Nyström decomposition to build our low-rank kernel. On the other hand, there exist other techniques where our algorithmic setting may also be used. For instance, the use of the so-called *CUR* decomposition [Drineas et al., 2008] could also be investigated.

In the mean time, establishing precise rate of convergence for the stochastic optimisation procedure and generalising our approach to the use of several kernels, establishing data-dependent generalisation bounds taking advantage of either the one-norm constraint on $\boldsymbol{\mu}$ or the size M of the kernel combination is of primary importance to us. The connection established between the one-norm hyperparameter ν and the ridge parameter λ, in section 4.4, seems interesting and may be witnessed in [Rakotomamonjy et al., 2008]. Although not been mentioned so far, there might be connections between our modeling strategy and boosting/leveraging-based optimisation procedures. Finally, we plan on generalising our approach to other kernel methods, noting that rank-1 update formulas as those proposed here can possibly be exhibited even for problems with no closed-form solution.

Chapter 5

Optimal Computational Trade-Off of Inexact Proximal Methods

Contents

5.1 Introduction **60**

5.2 Setting **63**

5.2.1 Inexact Proximal Methods 63

5.2.2 Convergence Rates 64

5.2.3 Approximation Trade-off 65

5.3 Defining the Problem **65**

5.3.1 The Computational Cost of Inexact Proximal Methods 66

5.3.2 Parameterising the Error 67

5.3.3 Parameterised Bounds 67

5.3.4 Towards a Computationally Optimal Tradeoff 68

5.4 Results **68**

5.4.1 Optimal Strategies 68

5.4.2 Comments and Interpretation of the Results 71

5.4.3 On the Usability of the Optimal Strategies 73

5.5 Numerical Simulations **75**

5.5.1 TV-regularisation for image deblurring 75

5.5.2 Graph prediction 76

5.5.3 Why the “computationally optimal” strategies are good but not that optimal 77

5.6 Conclusion and future work **80**

Abstract

In this chapter, we investigate the trade-off between convergence rate and computational cost when minimising a composite functional with proximal-gradient methods, which are optimisation tools that are popular in machine learning. We consider the case when the *proximity operator* is computed via an iterative procedure, which provides an approximation of the exact proximity operator. In that case, we obtain algorithms with two nested loops. We show that the strategy that minimises the computational cost to reach a solution with a desired accuracy in finite time is to set the number of inner iterations to a constant, which differs from the strategy indicated by a convergence rate analysis. In the process, we also present a new procedure called SIP (that is Speedy Inexact Proximal-gradient algorithm) that is both computationally efficient and easy to implement. Our numerical experiments confirm the theoretical findings and suggest that SIP can be a very competitive alternative to the standard procedure.

5.1 Introduction

Recent advances in machine learning and signal processing have led to more involved optimisation problems, while abundance of data calls for more efficient optimisation algorithms. First-order methods are now extensively employed to tackle these issues and, among them, proximal-gradient algorithms [Combettes and Wajs, 2005; Nesterov, 2007; Beck and Teboulle, 2009b] are becoming increasingly popular. They make it possible to solve very general convex non-smooth problems of the following form:

$$\min_x f(x) := g(x) + h(x), \tag{5.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ is convex and smooth with an L-Lipschitz continuous gradient and $h : \mathbb{R}^n \to \mathbb{R}$ is lower semi-continuous proper convex (see Definition 6 and 7), with remarkably simple, while effective, iterative algorithms which are guaranteed [Beck and Teboulle, 2009b] to achieve the optimal convergence rate of $O(1/k^2)$, for a first order method, in the sense of [Nemirovsky and Yudin, 1983]. They have been applied to a wide range of problems, from supervised learning with sparsity-inducing norm [Bach et al., 2011; Chen et al., 2011; Baldassarre et al., 2012; Mosci et al., 2010], imaging problems [Chambolle, 2004; Beck and Teboulle, 2009a; Fadili and Peyré,

2011], matrix completion [Cai et al., 2010; Lin et al., 2009], sparse coding [Jenatton et al., 2011] and multi-task learning [Chen et al., 2009].

The heart of these procedures is the *proximity operator*. In the favorable cases, analytical forms exist. However, there are many problems, such as Total Variation (TV) denoising and deblurring [Chambolle and Pock, 2011], non-linear variable selection [Mosci et al., 2010], structured sparsity [Jenatton et al., 2011; Baldassarre et al., 2012], trace norm minimisation [Cai et al., 2010; Lin et al., 2009], matrix factorisation problems such as the one described in [Schmidt et al., 2011], where the proximity operator can only be computed numerically, giving rise to what can be referred to as *inexact proximal-gradient* algorithms [Schmidt et al., 2011; Villa et al., 2011].

Both theory and experiments show that the precision of those numerical approximations has a fundamental impact on the performance of the algorithm. A simple simulation, experimenting different strategies for setting this precision, on a classical Total Variation image deblurring problem (see Section 5.5 for more details) highlights two aspects of this impact. Fig. 5.1 depicts the evolution of the objective value (hence precision) versus the computational cost (i.e. running time, see Section 5.3 for a more formal definition). The different curves are obtained by solving the exact same problem of the form (5.1), using, along the optimisation process, either a constant precision (for different constant values) for the computation of the proximity operator, or an increasing precision (that is computing the proximity operator more and more precisely along the

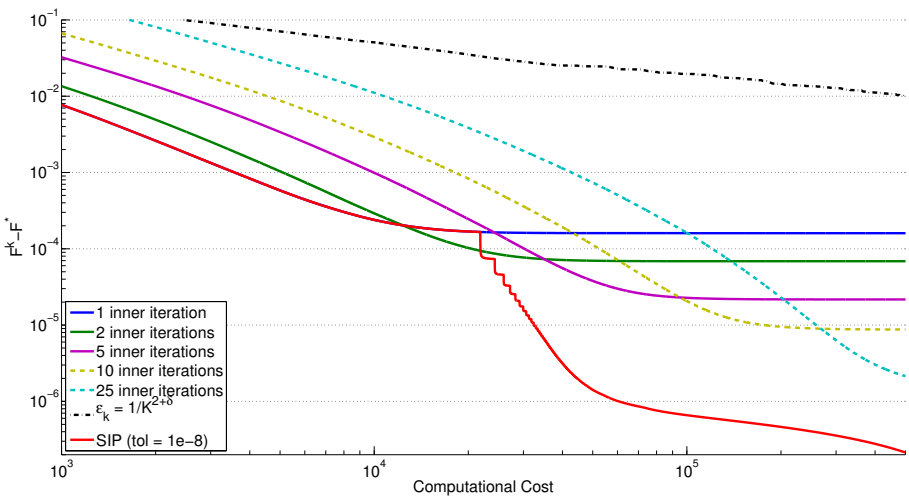

Figure 5.1: TV-regularisation : Computational cost vs. objective value for different strategies

process). It shows that the computation cost required to reach a fixed value of the objective value varies greatly between the different curves (i.e. strategies). That means that the computational performance of the procedure will dramatically depend on the chosen strategy. Moreover, we can see that the curves do not reach the same plateaus, meaning that the different strategies cause the algorithm to converge to different solutions. As discussed in the introduction of this thesis, the analysis of [Bottou and Bousquet, 2007] has highlighted the fact that solving optimisation problems with limited precision can be theoretically founded with an analysis of the trade-offs of large scale learning. This crucial point motivates the study of strategies that lead to an approximate solution, at a smaller computational cost, as the figure depicts.

However, the choice of the strategy that determines the precision of the numerical approximations seems to be often overlooked in practice. Yet, in the light of what we have discussed in that introduction, we think it is pivotal. In several studies, the precision is set so that the global algorithm converges to an optimum of the functional [Chaux et al., 2009], by studying sufficient conditions for such a convergence. In many others, it is only considered as a mere implementation detail. A quick review of the literature shows that many application-centered papers seem to neglect this aspect and fail at providing any detail regarding this point (e.g. [Anthoine et al., 2012]).

Recently, some papers have addressed this question from a more theoretical point of view. For instance, [Schmidt et al., 2011; Villa et al., 2011] give conditions on the approximations of the proximity operator so that the optimal convergence rate is still guaranteed. However, rate analysis is not concerned by the complexity of computing the proximity operator. As a consequence, the algorithms yielding the fastest rates of convergence are not necessarily the computationally lightest ones, hence not the ones yielding the shortest computation time. In fact, no attempts have yet been made to assess the global computational cost of those inexact proximal-gradient algorithms. It is worth mentioning that for some specific cases, other types of proximal-gradient algorithms have been proposed that allow to avoid computing complex proximity operator [Loris and Verhoeven, 2011; Chambolle and Pock, 2011].

In Section 5.2, we start from the results in [Schmidt et al., 2011] that link the overall accuracy of the iterates of inexact proximal-gradient methods with the errors in the approximations of the proximity operator. We consider iterative methods for computing the proximity operator and, in Section 5.3, we show that if one is interested in minimising the computational cost (defined

in Section 5.3.4) for achieving a desired accuracy, other strategies than the ones proposed in [Schmidt et al., 2011] and [Villa et al., 2011] might lead to significant computational savings.

The main contribution of our work is showing, in Section 5.4, that for both accelerated and non-accelerated proximal-gradient methods, the strategy minimising the global cost to achieve a desired accuracy is to keep the number of internal iterations constant. This constant depends on the desired accuracy and the convergence rate of the algorithm used to compute the proximity operator. Coincidentally, those theoretical strategies meet those of actual implementations and widely-used packages and help us understand both their efficiency and limitations. After a discussion on the applicability of those strategies, we also propose a more practical one, namely the Speedy Inexact Proximal-gradient (SIP) strategy, motivated by our analysis.

In Section 5.5, we numerically assess different strategies (i.e. constant numbers of inner iterations, SIP, the strategy yielding optimal convergence rates) on two problems, illustrating the theoretical analysis and suggesting that our new strategy SIP can be very effective. This leads to a final discussion about the relevance and potential limits of our approach along with some hints on how to overcome them.

5.2 Setting

5.2.1 Inexact Proximal Methods

To solve problem (5.1), one may use the so-called *proximal*-gradient methods [Nesterov, 2007]. Those iterative methods consist in generating a sequence $\{x_k\}$, where

$$x_k = \operatorname{prox}_{h/L}\left[y_{k-1} - \tfrac{1}{L}\nabla g(y_{k-1})\right],$$

$$\text{with } y_k = x_k + \beta_k(x_k - x_{k-1}),$$

and the choice of β_k gives rise to two schemes: $\beta_k = 0$ for the *basic scheme*, or some well-chosen sequence (see [Nesterov, 2007; Tseng, 2008; Beck and Teboulle, 2009b] for instance) for an *accelerated scheme*. The *proximity operator* $\operatorname{prox}_{h/L}$ is defined as:

$$\operatorname{prox}_{h/L}(z) = \operatorname*{argmin}_x \tfrac{L}{2}\|x - z\|^2 + h(x). \tag{5.2}$$

In the most classical setting, the proximity operator is computed exactly. The sequence $\{x_k\}$ then converges to the solution of problem (5.1). However, in many situations no closed-form

solution of (5.2) is known and one can only provide an approximation of the proximal point. From now on, let us denote by ϵ_k an upper bound on the error induced in the proximal objective function by this approximation, at the k-th iteration:

$$\frac{L}{2}\|x_k - z\|^2 + h(x_k) \leq \epsilon_k + \min_x \left\{ \frac{L}{2}\|x - z\|^2 + h(x) \right\}. \tag{5.3}$$

For the basic scheme, the convergence of $\{x_k\}$ to the optimum of Problem (5.1) has been studied in [Combettes and Wajs, 2005] and is ensured under fairly mild conditions on the sequence $\{\epsilon_k\}$.

5.2.2 Convergence Rates

The authors of [Schmidt et al., 2011] go beyond the study on the convergence of inexact proximal methods: they establish their rates of convergence. (This is actually done in the more general case where the gradient of g is also approximated. In the present study, we restrict ourselves to error in the proximal part.)

Let us denote by x^* the solution of problem (5.1). The convergence rates of the basic (non-accelerated) proximal method (e.g. $y_k = x_k$) thus reads:

Proposition 5 (Basic proximal-gradient method (Proposition 1 in [Schmidt et al., 2011])). *For all $k \geq 1$,*

$$f(x_k) - f(x^*) \leq \frac{L}{2k}\left(\|x_0 - x^*\| + 2\sum_{i=1}^{k}\sqrt{\frac{2\epsilon_i}{L}} + \sqrt{\sum_{i=1}^{k}\frac{2\epsilon_i}{L}} \right)^2. \tag{5.4}$$

Remark 4. In [Schmidt et al., 2011], this bound actually holds on the average of the iterates x_i, i.e.

$$f\left(\frac{1}{k}\sum_{i=1}^{k} x_i\right) - f(x^*) \leq \frac{L}{2k}\left(\|x_0 - x^*\| + 2\sum_{i=1}^{k}\sqrt{\frac{2\epsilon_i}{L}} + \sqrt{\sum_{i=1}^{k}\frac{2\epsilon_i}{L}} \right)^2.$$

(5.4) thus holds for the iterate that achieve the lowest function value. It also trivially holds all the time for algorithms with which the objective is non-increasing.

The convergence rate of accelerated schemes (e.g. $y_k = x_k + \frac{k-1}{k+2}x_{k-1}$) reads:

Proposition 6 (Accelerated proximal-gradient method (Proposition 2 in [Schmidt et al., 2011])).

For all $k \geq 1$,

$$f(x_k) - f(x^*) \leq \frac{2L}{(k+1)^2} \left(\|x_0 - x^*\| + 2\sum_{i=1}^{k} i\sqrt{\frac{2\epsilon_i}{L}} + \sqrt{\sum_{i=1}^{k} \frac{2i^2\epsilon_i}{L}} \right)^2 . \tag{5.5}$$

Bounds with faster rates (Proposition 3 and 4 in [Schmidt et al., 2011]) can be obtained if the objective is strongly convex (see Definition 3). Some results will be briefly mentioned in Section 5.4 in this case. However, we will not detail them as much as in the more general setting.

5.2.3 Approximation Trade-off

The inexactitude in the computation of the proximity operator imposes two additional terms in each bound, for instance in (5.4): :

$$2\sum_{i=1}^{k} \sqrt{\frac{2\epsilon_i}{L}} \text{ and } \sqrt{\sum_{i=1}^{k} \frac{2\epsilon_i}{L}}.$$

When the ϵ_i's are set to 0 (i.e. the proximity operator is computed exacted), one obtains the usual bounds of the exact proximal methods. These additional terms (in (5.4) and (5.5) resp.) are summable if $\{\epsilon_k\}$ converges at least as fast as $O\left(\frac{1}{k^{(2+\delta)}}\right)$ (resp. $O\left(\frac{1}{k^{(4+\delta)}}\right)$), for any $\delta > 0$. One direct consequence of these bounds (in the basic and accelerated schemes respectively) is that the optimal convergence rates in the error-free setting are still achievable, with such conditions on the $\{\epsilon_k\}$'s. Improving the convergence rate of $\{\epsilon_k\}$ further causes the additional terms to sum to smaller constants, hence inducing a faster convergence of the algorithm without improving the rate. However, [Schmidt et al., 2011] empirically notices that imposing too fast a decrease rate on $\{\epsilon_k\}$ is computationally counter-productive, as the precision required on the proximal approximation becomes computationally demanding. In other words, there is a subtle trade-off between the number of iterations needed to reach a certain solution and the cost of those iterations. This is the object of study of the present chapter.

5.3 Defining the Problem

The main contribution of this chapter is to define a *computationally optimal* way of setting the trade-off between the number of iterations and their cost, in various situations. We consider the case where the proximity operator is approximated *via* an iterative procedure. The global

algorithm thus consists in an iterative proximal method, where at each (outer-)iteration, one performs (inner-)iterations, as Algorithm 4 depicts.

Algorithm 4 A general framework for Inexact Proximal Algorithms

Require: initial point x_0

for $i = 1$ to k **do**

$x_{i-\frac{1}{2}} := x_{i-1} - \frac{1}{L}\nabla g(x_{i-1})$ "gradient descent" step

while precision is lower than ϵ_i **do**

Increase the precision of $\text{prox}_{h/L}(x_{i-\frac{1}{2}})$

end while

$x_i := \text{prox}_{h/L}(x_{i-\frac{1}{2}})$

end for

With that setting, it is possible to define (hence optimise) the global computational cost of the algorithm. If the convergence rate of the procedure used in the inner-loops is known, the main result of this study provides a strategy to set the number of inner iterations that minimises the cost of the algorithm, under some constraint upper-bounding the precision of the solution (as defined in (5.8)).

5.3.1 The Computational Cost of Inexact Proximal Methods

As stated earlier, our goal is to take into account the complexity of the global cost of inexact proximal methods. Using iterative procedures to estimate the proximity operator at each step, it is possible to formally express this cost. Let us assume that each inner-iteration has a constant computational cost C_{in} and that, in addition to the cost induced by the inner-iterations, each outer-iteration has a constant computational cost C_{out}. It immediately follows that the global cost of the algorithm is:

$$C_{\text{glob}}(k, \{l_i\}_{i=1}^k) = C_{\text{in}} \sum_{i=1}^{k} l_i + kC_{\text{out}}, \tag{5.6}$$

and the question we are interested in is to minimise this cost. In order to formulate our problem as a minimisation of this cost, subject to some guarantees on the global precision of the solution, we now need to relate the number of inner iterations to the precision of the proximal point estimation. This issue is addressed in the following subsections.

5.3.2 Parameterising the Error

Classical methods to approximate the proximity operator achieve either *sublinear rates* of the form $O\left(\frac{1}{k^\alpha}\right)$ ($\alpha = \frac{1}{2}$ for sub-gradient or stochastic gradient descent in the general case; $\alpha = 1$ for gradient and proximal descent or $\alpha = 2$ for accelerated descent/proximal schemes) or *linear rates* $O\left((1-\gamma)^k\right)$ (for strongly convex objectives or second-order methods). Let l_i denote the number of inner iterations performed at the i-th iteration of the outer-loop. We thus consider two types of upper bounds on the error defined in (5.3):

$$\epsilon_i = \frac{A_i}{l_i^\alpha} \quad \text{(sublinear rate)} \qquad \text{or} \qquad \epsilon_i = A_i(1-\gamma)^{l_i} \quad \text{(linear rate)}, \tag{5.7}$$

for some positive A_i's.

5.3.3 Parameterised Bounds

Plugging (5.7) into (5.4) or (5.5), we can get four different global bounds:

$$f(x_k) - f(x^*) \leq B_j(k, \{l_i\}_{i=1}^k), \; j = 1, .., 4,$$

depending on whether we are using a basic or accelerated scheme on the one hand, and on whether we have sub-linear or linear convergence rate in the inner-loops on the other hand. More precisely, we have the following four cases:

1. basic out, sub-linear in:

$$B_1(k, \{l_i\}_{i=1}^k) = \frac{L}{2k}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k \sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2$$

2. basic out, linear in:

$$B_2(k, \{l_i\}_{i=1}^k) = \frac{L}{2k}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k \sqrt{\frac{2A_i(1-\gamma)^{l_i}}{L}}\right)^2$$

3. accelerated out, sub-linear in:

$$B_3(k, \{l_i\}_{i=1}^k) = \frac{2L}{(k+1)^2}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k i\sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2$$

4. accelerated out, linear in:

$$B_4(k, \{l_i\}_{i=1}^k) = \frac{2L}{(k+1)^2}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k i\sqrt{\frac{2A_i(1-\gamma)^{l_i}}{L}}\right)^2$$

5.3.4 Towards a Computationally Optimal Tradeoff

Those bounds highlight the aforementioned trade-off. To achieve some fixed global error

$$\rho = f(x_k) - f(x^*)$$

, there is a natural trade-off that need to be set by the user, between the number k of outer-iterations and the numbers of inner-iterations $\{l_i\}_{i=1}^k$, which can be seen as hyper-parameters of the global algorithms. As mentioned earlier, and witnessed in [Schmidt et al., 2011] the choice of those parameters will have a crucial impact on the computational efficiency (see equation (5.6)) of the algorithm.

Our aim to "optimally" set the hyper-parameters (k and $\{l_i\}_{i=1}^k$) may be conveyed by the following optimisation problem. For some fixed accuracy ρ, we want to minimise the global cost C_{glob} of the algorithm, under the constraint that our bound on the error B is smaller than ρ:

$$\min_{k\in\mathbb{N},\{l_i\}_{i=1}^k\in\mathbb{N}^k} C_{\text{in}} \sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } B(k,\{l_i\}_{i=1}^k) \leq \rho. \tag{5.8}$$

This optimisation problem the rest of this chapter will rest upon.

5.4 Results

Problem (5.8) is an integer optimisation problem as the variables of interest are numbers of (inner and outer) iterations. As such, this is a complex (NP-hard) problem and one cannot find a closed form for the integer solution, but if we relax our problem in l_i to a continuous one:

$$\min_{k\in\mathbb{N},\{l_i\}_{i=1}^k\in[1,\infty)^k} C_{\text{in}} \sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } B(k,\{l_i\}_{i=1}^k) \leq \rho, \tag{5.9}$$

it actually is possible to find an *analytic expression* of the optimal $\{l_i\}_{i=1}^k$ and to numerically find the optimal k.

5.4.1 Optimal Strategies

The next four propositions describe the solution of the relaxed version (5.9) of Problem (5.8) in the four different scenarios defined in Section 5.3.3 and for a constant value $A_i = A$.

Scenarios 1 and 2: basic out

Let

$$C(k) = \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{2k\rho}{L}} - \|x_0 - x^*\|\right).$$

Solving the continuous relaxation of problem (5.8) with the bounds B_1 and B_2 leads to the following propositions:

Proposition 7 (Basic out, sub-linear in)**.** *If $\rho < 6\sqrt{2LA}\|x_0 - x^*\|$, the solution of problem (5.8) for $B = B_1$ is:*

$$\forall\ i, l_i^* = \left(\frac{C(k^*)}{k^*}\right)^{-\frac{2}{\alpha}},\ \textit{with } k^* = \underset{k\in\mathbb{N}^*}{\operatorname{argmin}}\, kC_{in}\left(\frac{C(k)}{k}\right)^{-\frac{2}{\alpha}} + kC_{out}. \tag{5.10}$$

Proposition 8 (Basic out, linear in)**.** *If $\rho < 6\sqrt{2LA(1-\gamma)}\|x_0 - x^*\|$, the solution of problem (5.8) for $B = B_2$ is:*

$$\forall\ i, l_i^* = \frac{2\ln\frac{C(k^*)}{k^*}}{\ln(1-\gamma)},\ \textit{with } k^* = \underset{k\in\mathbb{N}^*}{\operatorname{argmin}}\, \frac{2kC_{in}}{\ln(1-\gamma)}\ln\left(\frac{C(k)}{k}\right) + kC_{out}. \tag{5.11}$$

Scenarios 3 and 4: accelerated out

Let

$$D(k) = \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{\rho}{2L}}(k+1) - \|x_0 - x^*\|\right).$$

Solving the continuous relaxation (5.9) of problem (5.8) with the bound B_3 leads to the following proposition:

Proposition 9 (Accelerated out, sub-linear in)**.** *If $\rho < \left(\sqrt{12\sqrt{2LA}\|x_0 - x^*\|} - 3\sqrt{A}\right)^2$, the solution of problem (5.8) for $B = B_3$ is:*

$$\forall i, l_i^* = \left(\frac{2D(k^*)}{k^*(k^*+1)}\right)^{-\frac{2}{\alpha}},\ \textit{with } k^* = \underset{k\in\mathbb{N}^*}{\operatorname{argmin}}\, kC_{in}\left(\frac{2D(k)}{k(k+1)}\right)^{-\frac{2}{\alpha}} + kC_{out}. \tag{5.12}$$

A similar result holds for the last scenario: $B = B_4$. However in this case, the optimal l_i are equal to 1 up to $n(k^*)$ ($1 \le n(k^*) < k^*$) and then increase with i:

Proposition 10 (Accelerated out, linear in)**.** *If $\rho < \left(\sqrt{12\sqrt{2LA(1-\gamma)}\|x_0 - x^*\|} - 3\sqrt{A}\right)^2$, the solution of problem (6) for $B = B_4$ is:*

$$l_i^* = \begin{cases} 1 & \textit{for } 1 \le i \le n(k^*) - 1 \\ \frac{2}{\ln(1-\gamma)}\left(\ln\left(\frac{D(k) - \frac{n(k)(n(k)-1)}{2}\sqrt{1-\gamma}}{k+1-n(k)}\right)\right) & \textit{for } n(k^*) \le i \le k^* \end{cases}$$

$$\textit{with } k^* = \underset{k\in\mathbb{N}^*}{\operatorname{argmin}}\Bigg\{kC_{out} + C_{in}(n(k)-1) - \tfrac{2C_{in}}{\ln(1-\gamma)}\ln\left(\tfrac{k!}{n(k)!}\right) - \tfrac{2C_{in}(k-n(k)+1)}{\ln(1-\gamma)}\ln\left(\frac{k+1-n(k)}{D(k)-\frac{n(k)(n(k)-1)}{2}\sqrt{1-\gamma}}\right)\Bigg\}, \tag{5.13}$$

and $n(k)$ is defined as the only integer such that:

$$\big(n(k)-1\big)\big(2k+2-n(k)\big)\sqrt{1-\gamma} \leq 2D(k) < n(k)\big(2k+1-n(k)\big)\sqrt{1-\gamma}.$$

Sketch of proof (For a complete proof, please see appendix 7.2.) First note that:

$$\min_{k,\{l_i\}_{i=1}^k} C_{\text{in}}\sum_{i=1}^k l_i + kC_{\text{out}} = \min_k \min_{\{l_i\}_{i=1}^k} C_{\text{in}}\sum_{i=1}^k l_i + kC_{\text{out}}.$$

We can solve problem (5.8) by first solving, for any k, the minimisation problem over $\{l_i\}_{i=1}^k$. This is done using the standard Karush-Kuhn-Tucker approach [Kuhn and Tucker, 1951]. Plugging the analytic expression of those optimal $\{l_i^*\}_{i=1}^k$ into our functional, we get our problem in k. □

Remark 5. Notice that the propositions hold for ρ smaller than a threshold. If not, the analysis and results are different. Since we focus on a high accuracy, we here develop the results for small values of ρ. We defer the results for the larger values of ρ to appendix 7.2.

Remark 6. In none of the scenarios can we provide an analytical expression of k^*. However, the expressions given in the propositions allow us to exactly retrieve the solution. The functions of k to minimise are monotonically decreasing then increasing. As a consequence, it is possible to numerically find the minimiser in $\mathbb{R}$, for instance in the first scenario:

$$\hat{k} = \underset{k\in\mathbb{R}}{\operatorname{argmin}}\, kC_{\text{in}}\left(\frac{C(k)}{k}\right)^{-\frac{2}{\alpha}} + kC_{\text{out}},$$

with an arbitrarily high precision, using for instance a First-Order Method. It follows that the integer solution k^* is exactly either the flooring or ceiling of $\hat{k}$. Evaluating the objective for the two possible roundings gives the solution.

Finally, as briefly mentioned, bounds with faster rates can be obtained when the objective is known to be strongly convex. In that case, regardless of the use of basic or accelerated schemes and of sub-linear or linear rates in the inner loops, the analysis leads to results similar to those reported in Proposition 10 (e.g. using 1 inner iteration for the first rounds and an increasing number then). Due to the lack of usability and interpretability of these results, we will not report them here.

5.4.2 Comments and Interpretation of the Results

Constant number of inner iterations

Our theoretical results urge to use a constant number of inner iterations in 3 scenarios. Coincidentally, many actual efficient implementations of such two nested algorithms, in [Anthoine et al., 2012] or in packages like SLEP[1] or PQN[2], use these constant number schemes. However, the theoretical grounds for such an implementation choice were not explicited. Our results can give some deeper understanding on why and how those practical implementations perform well. They also help acknowledging that the computation gain comes at the cost of an intrinsic limitation to the precision of the obtained solution.

An Integer Optimisation Problem

The impact of the continuous relaxation of the problem in $\{l_i\}_{i=1}^{k^*}$ is subtle. In practice, we need to set the constant number on inner iterations l_i to an integer number. Setting, $\forall i \in [1, k^*], l_i = \lceil l_i^* \rceil$ ensures that the final error is smaller than ρ. This provides us with an approximate (but feasible) solution to the integer problem.

One may want to refine this solution by sequentially setting l_i to $\lfloor l_i^* \rfloor$ (hence reducing the computational cost), starting from $i = 1$, while the constraint is met, i.e. the final error remains smaller than ρ. Refer to Algorithm 5 for an algorithmic description of the procedure.

Algorithm 5 A finer grain procedure to obtain an integer solution for the l_i's

Require: $\{l_i^*\}_{i=1}^{k^*}$

$\forall i \in [1, k^*], l_i \leftarrow \lceil l_i^* \rceil$

$i \leftarrow 1$

repeat

$l_i \leftarrow \lfloor l_i^* \rfloor$

$i \leftarrow i + 1$

until $B(k^*, \{l_i\}_{i=1}^{k^*}) > \rho$

[1] http://www.public.asu.edu/ jye02/Software/SLEP/index.htm

[2] http://www.di.ens.fr/ mschmidt/Software/PQN.html

Computationally-Optimal vs. Optimal Convergence Rates Strategies

The original motivation of this study is to show how, in the inexact proximal methods setting, optimisation strategies that are the most computationally efficient, given some desired accuracy ρ, are *fundamentally different* from those that achieve optimal convergence rates. The following discussion motivates why minding this gap is of great interest for machine learners while an analysis of the main results of this work highlights it.

When one wants to obtain a solution with an arbitrarily high precision, optimal rate methods are of great interest: regardless of the constants in the bounds, there always exists a (very high) precision ρ beyond which methods with optimal rates will be faster than methods with suboptimal convergence rates. However, when dealing with real large-scale problems, reaching those levels of precision is not computationally realistic. When taking into account budget constraints on the computation time, and as suggested by [Bottou and Bousquet, 2007], generalisation properties of the learnt function will depend on both statistical and computational properties.

At the levels of precision intrinsically imposed by the budget constraints, taking other elements than the convergence rates becomes crucial for designing efficient procedures as our study shows. Other examples of that phenomenon have been witnessed, for instance, when using Robbins-Monro algorithm (Stochastic Gradient Descent). It has been long known (see [Polyak and Juditsky, 1992] for instance) that that the use of a step-size proportional to the inverse of the number of iterations allows to reach the optimal convergence rates (namely $1/k$) .

On the other hand, using a non-asymptotic analysis [Bach and Moulines, 2011], one can prove (and observe in practice) that such a strategy can also lead to catastrophic results when k is small (i.e. possibly a large increase of the objective value) and undermines the computational efficiency of the whole procedure.

Back to our study, for the first three scenarios (Propositions 7, 8 and 9), the computationally-optimal strategy imposes constant number of inner iterations. Given our parameterisation, Eq. (5.7), this also means that the errors ϵ_i on the proximal computation remains constant. On the opposite, the optimal convergence rates can only be achieved for sequences of ϵ_i decreasing strictly faster than $1/i^2$ for the basic schemes and $1/i^4$ for the accelerated schemes. Obviously, the optimal convergence rates strategies also yield a bound on the minimal number of outer iterations needed to reach precision ρ by inverting the bounds (5.4) or (5.5). However, this

strategy is provably less efficient (computationally-wise) than the optimal one we have derived.

In fact, the pivotal difference between "optimal convergence rates" and "computationally optimal" strategies lies in the fact that the former ones arise from an asymptotic analysis while the latter arise from a finite-time analysis. While the former ensures that the optimisation procedure will converge to the optimum of the problem (with optimal rates in the worst case), the latter only ensures that after k^* iterations, the solution found by the algorithm is not further than ρ from the optimum.

Do not *optimise further*

To highlight this decisive point in our context, let us fix some arbitrary precision ρ. Propositions 7 to 9 give us the optimal values k^* and $\{l_i^*\}_{i=1}^{k^*}$ depending on the inner and outer algorithms we use. Now, if one wanted to *further optimise* by continuing the same strategy for $k' > k^*$ iterations (i.e. still running l_i^* inner iterations), we would have the following bound:

$$B(k', \{l_i^*\}_{i=1}^{k'}) > B(k^*, \{l_i^*\}_{i=1}^{k^*}) = \rho.$$

In other words, if one runs more than k^* iterations of our optimal strategy, with the same l_i, we can not guarantee that the error still decreases. In a nutshell, our strategy is precisely computationally optimal because it does not ensure more than what we ask for.

5.4.3 On the Usability of the Optimal Strategies

Designing computationally efficient algorithms or optimisation strategies is motivated by practical considerations. The strategies we proposed are provably the best to ensure a desired precision. Yet, in a setting that covers a very broad range of problems, their usability can be compromised. We point out those limitations and propose a solution to overcome them.

First, these strategies require the desired (absolute) precision to be known. In most situations, it is actually difficult, if not impossible, to know in advance which precision will ensure that the solution found has desired properties (e.g. reaching some specific SNR ratio for image deblurring). More critically, if it turned out that the user-defined precision was not sufficient, we showed that "optimising further" with the same number of inner iterations does not guarantee to improve the solution. For a sharper precision, one would technically have to compute the new optimal strategy and run it all over again.

Although it is numerically possible, evaluating the optimal number of iterations k^* still requires to solve an optimisation problem. More importantly, the optimal values for the numbers of inner and outer iterations depend on quantities like $\|x_0 - x^*\|$ which are unknown and very difficult to estimate. Those remarks undermine the direct use of the presented computationally optimal strategies.

To overcome these problems, we propose a new strategy called *Speedy Inexact Proximal-gradient algorithm* (*SIP*), described in Algorithm 6, which is motivated by our theoretical study and very simple to implement. In a nutshell, it starts using only one inner iteration. When the outer objective stops decreasing fast enough, the algorithm increases the number of internal iterations used for computing the subsequent proximal steps, until the objective starts decreasing fast enough again.

Algorithm 6 Speedy Inexact Proximal-gradient strategy (*SIP*)

Require: An initial point x_0, an update rule $\mathcal{A}_{\text{out}}$, an iterative algorithm $\mathcal{A}_{\text{in}}$ for computing the proximity operator, a tolerance tol > 0, a stopping criterion STOP.

$x \leftarrow x_0$, $l \leftarrow 1$

repeat

  $\hat{x} = x - \frac{1}{L}\nabla g(x)$ *Gradient Step*

  $z^0 \leftarrow 0$

  for $i = 1$ to l **do**

    $z^i = \mathcal{A}_{\text{in}}(\hat{x}, z^{i-1})$ *Proximal Step*

  end for

  $\hat{x} = z^l$

  if $f(x) - f(\hat{x}) < \text{tol} f(x)$ **then**

    $l \leftarrow l + 1$ *Increase proximal iterations*

  end if

  $x = \mathcal{A}_{\text{out}}(x, \hat{x})$ *Basic or accelerated update*

until STOP is met

Beyond the simplicity of the algorithm (no parameter except for the tolerance, no need to set a global accuracy in advance), *SIP* leverages the observation that a constant number of inner iterations l only allows to reach some underlying accuracy. As long as this accuracy has not

been reached, it is not necessary to put more efforts into estimating the proximity operator. The rough idea is that far from the minimum of a convex function, moving along a rough estimation of the steepest direction will be very likely to have the function decrease fast enough, hence the low precision required for the proximal point estimation. On the other hand, when close to the minimum, a much higher precision is required, hence the need for using more inner iterations. This point of view meets the one developed in [Boyles et al., 2011] in the context of stochastic optimisation, where the authors suggest to use increasing batch sizes (along the optimisation procedure) for the stochastic estimation of the gradient of functional to minimise, in order to achieve computational efficiency.

5.5 Numerical Simulations

The objective of this section is to empirically investigate the behaviour of proximal-gradient methods when the proximity operator is estimated via a fixed number of iterations. We also assess the performance of the proposed SIP algorithm. Our expectation is that a strategy with just one internal iteration will be computationally optimal only up to a certain accuracy, after which using two internal iterations will be more efficient and so on. We consider an image deblurring problem with total variation regularisation and a semi-supervised learning problem using two sublinear methods for computing the proximity operator.

5.5.1 TV-regularisation for image deblurring

The problem of denoising or deblurring an images is often tackled via Total Variation regularisation [Rudin et al., 1992; Chambolle, 2004; Beck and Teboulle, 2009a]. The total variation regulariser allows one to preserve sharp edges and is defined as

$$g(x) = \lambda \sum_{i,j=1}^{N} \|(\nabla x)_{i,j}\|_2$$

where $\lambda > 0$ is a regularisation parameter and ∇ is the discrete gradient operator [Chambolle, 2004]. We use the smooth quadratic data fit term $f(x) = \|Ax - y\|_2^2$, where A is a linear blurring operator and y is the image to be deblurred. This leads to the following problem:

$$\min_x \|Ax - y\|_2^2 + \lambda \sum_{i,j=1}^{N} \|(\nabla x)_{i,j}\|_2.$$

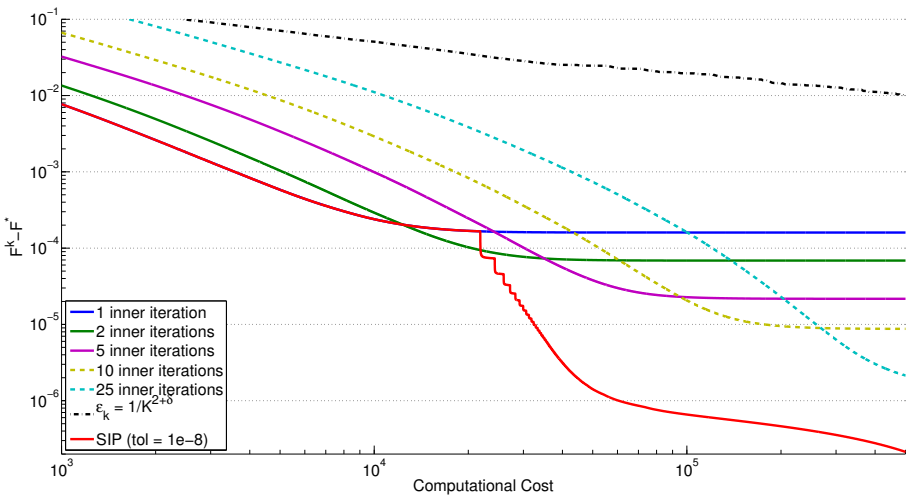

Figure 5.2: Deblurring with Total Variation - Basic method

Our experimental setup follows the one in [Villa et al., 2011], where it was used for an asymptotic analysis. We start with the famous Lena test image, scaled to 256×256 pixels. A 9×9 Gaussian filter with standard deviation 4 is used to blur the image. Normal noise with zero mean and standard deviation 10^{-3} is also added. The regularisation parameter λ was set to 10^{-4}. We run the basic proximal-gradient method up to a total computational cost of $C = 10^6$ (where we set $C_{\text{in}} = C_{\text{out}} = 1$) and the accelerated method up to a cost of 5×10^4. We computed the proximity operator using the algorithm of [Beck and Teboulle, 2009a], which is a basic proximal-gradient method applied to the dual of the proximity operator problem. We used a fixed number of iterations and compared with the convergent strategy proposed in [Schmidt et al., 2011] and the SIP algorithm with tolerance 10^{-8}. As a reference for the optimal value of the objective function, we used the minimum value achieved by any method (i.e. the SIP algorithm in all cases) and reported the results in Fig. 5.2 and 5.3.

As the figures display a similar behaviour for the different problems we ran our simulations on, we defer the analysis of the results to 5.5.3.

5.5.2 Graph prediction

The second simulation is on the graph prediction setting of [Herbster and Lever, 2009]. It consists in a sequential prediction of boolean labels on the vertices of a graph, the learner's goal being

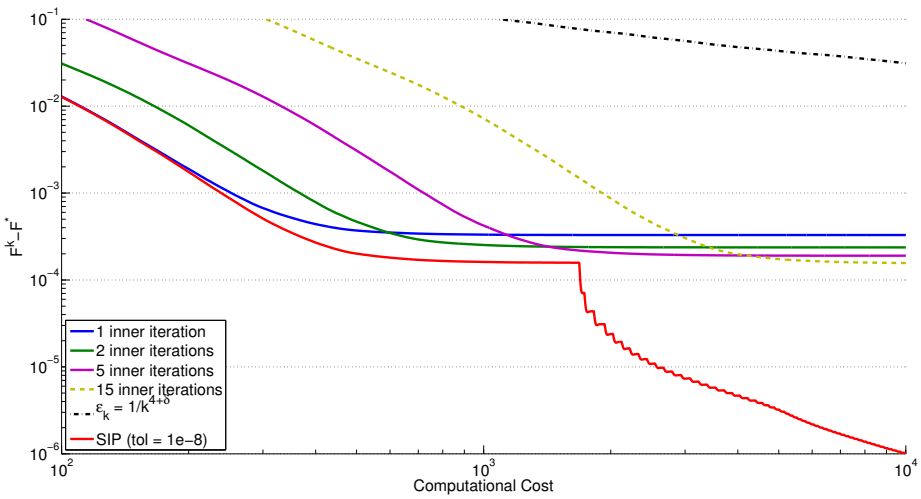

Figure 5.3: Deblurring with Total Variation - Accelerated method

the minimisation of the number of mistakes. More specifically, we consider a 1-seminorm on the space of graph labellings, which corresponds to the minimisation of the following problem (composite ℓ_1 norm)

$$\min_x \|Ax - y\|^2 + \lambda \|Bx\|_1,$$

where A is a linear operator that selects only the vertices for which we have labels y, B is the edge map of the graph and $\lambda > 0$ is a regularisation parameter (set to 10^{-4}). We constructed a synthetic graph of $d = 100$ vertices, with two clusters of equal size. The edges in each cluster were selected from a uniform draw with probability $\frac{1}{2}$ and we explicitly connected $d/25$ pairs of vertices between the clusters. The labelled data y were the cluster labels ($+1$ or -1) of $s = 10$ randomly drawn vertices. We compute the proximity operator of $\lambda\|Bx\|_1$ via the method proposed in [Combettes et al., 2010], which essentially is a basic proximal method on the dual of the proximity operator problem. We follow the same experimental protocol as in the total variation problem and report the results in Fig. 5.4 and 5.5.

5.5.3 Why the "computationally optimal" strategies are good but not that optimal

On all the displayed results (Fig. 5.2, 5.3, 5.4 and 5.5), and as the theory predicted, we can see that for almost any given accuracy ρ (i.e. $F^k - F^*$ on the figures), there exists some constant

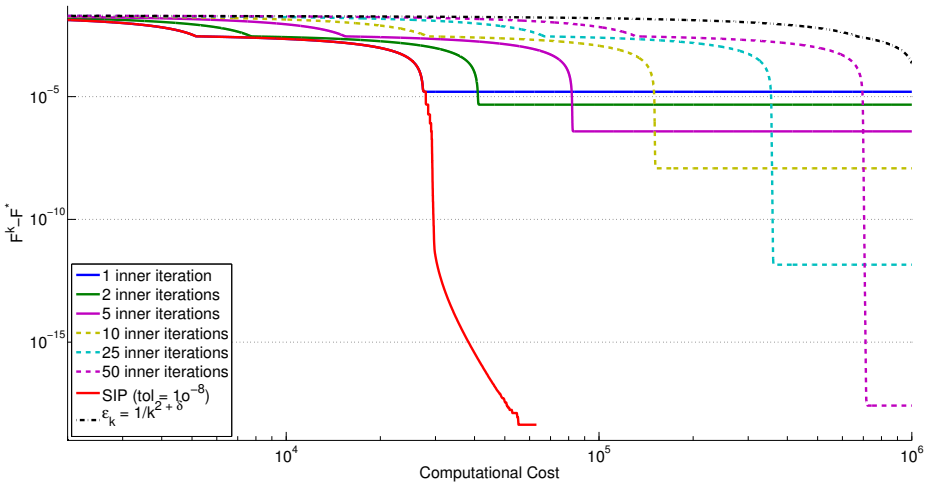

Figure 5.4: Graph learning - Basic method

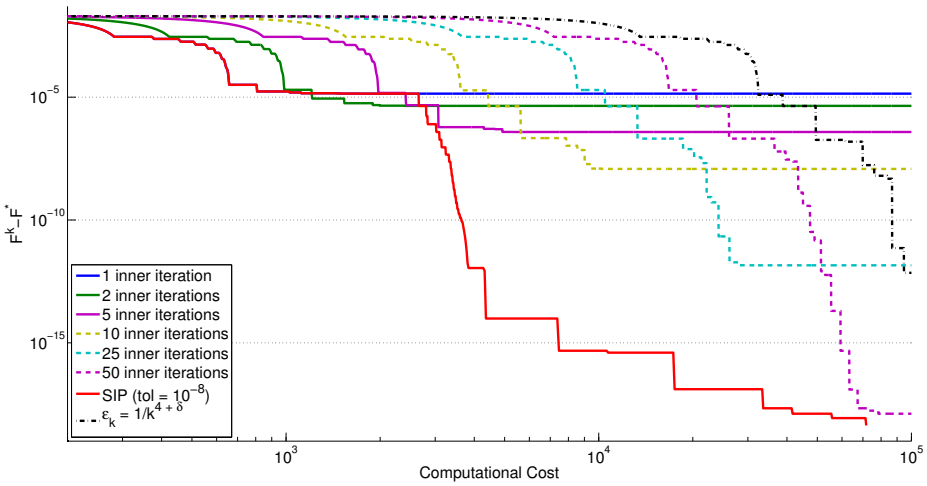

Figure 5.5: Graph learning - Accelerated method

value for l_i that yields a strategy that is potentially orders of magnitude more efficient than the strategy that ensures the fastest global convergence rate. On any of the figures, comparing the curves obtained with 1 and 2 inner iterations, one may notice that the former first increases the precision faster than the latter. Meanwhile, the former eventually converges to a higher plateau than the latter. This observation remains as the number of constant iterations increases. This highlights the fact that smaller constant values of l_i lead to faster algorithms at the cost of a worse global precision. On the other hand, the *SIP* strategy seems to almost always be the fastest strategy to reach any desired precision. That makes it the most computationally efficient strategy as the figures show. This may look surprising as the constant l_i's strategies are supposed to be optimal for a specific precision and obviously are not.

In fact, there is no contradiction with the theory: keeping l_i constant leads to the optimal strategies for minimising a bound on the real error, which can be significantly different than directly minimising the error.

This remark raises crucial issues. If the bound we use for the error was a perfect description of the real error, the strategies with constant l_i would be the best also in practice. Intuitively, the tighter the bounds, the closest our theoretical optimal strategy will be from the actual optimal one. This intuition is corroborated by our numerical experiments. In our parameterisation of ϵ_i, in a first approximation, we decided to consider constant A_i (see equation (5.7)). When not using warm restarts between two consecutive outer iterations, our model of ϵ_i does describe the actual behaviour much more accurately and our theoretical optimal strategy seems much closer to the real optimal one. To take warm starts into account, one would need to consider decreasing sequences of A_i's. Doing so, one can notice that in the first 3 scenarios, the optimal strategies would not consist in using constant number of inner iterations any longer, but only constant ϵ_i's, hence maintaining the same gap between optimal rates and computationally optimal strategies.

These ideas urge for a finer understanding on how optimisation algorithms behave in practice. Our claim is that one pivotal key to design practically efficient algorithms is to have new tools such as warm-start analysis and, perhaps more importantly, convergence bounds that are tighter for specific problems (i.e. "specific-case" analysis rather than the usual "worst-case" ones).

5.6 Conclusion and future work

We analysed the computational cost of proximal-gradient methods when the proximity operator is computed numerically. Building upon the results in [Schmidt et al., 2011], we proved that the optimisation strategies, using a constant number of inner iterations, can have very significant impacts on computational efficiency, at the cost of obtaining only a suboptimal solution. Our numerical experiments showed that these strategies do exist in practice, albeit it might be difficult to access them. Coincidentally, those theoretical strategies meet those of actual implementations and widely-used packages and help us understanding both their efficiency and limitations. We also proposed a novel optimisation strategy, the SIP algorithm, that can bring large computational savings in practice and whose theoretical analysis needs to be further developed in future studies. Throughout the paper, we highlighted the fact that finite-time analysis, such as ours, urges for a better understanding of (even standard) optimisation procedures. There is a need for sharper and problem-dependent error bounds, as well as a better theoretical analysis of warm-restart, for instance.

Finally, although we focused on inexact proximal-gradient methods, the present work was inspired by the paper "The Trade-offs of Large-Scale Learning" [Bottou and Bousquet, 2007]. Bottou and Bousquet studied the trade-offs between computational accuracy and statistical performance of machine learning methods and advocate for sacrificing the rate of convergence of optimisation algorithms in favour of lighter computational costs. At a higher-level, future work naturally includes finding other situations where such trade-offs appear and analyse them using a similar methodology.

Chapter 6

Conclusion

Contents

6.1 Summary of Contributions 81
6.2 Broader Prospects 83

In this chapter, we first briefly summarise our main contributions. This opens an opportunity to re-examine the discussion that was conducted in the introduction and to highlight, for each chapter, the prospect that seems the most relevant to us. This allows us to expose, in a second part, the extent to which our initial questions have been answered. Finally, we take time to pinpoint some potential future directions and we draw connections between our contributions and seemingly unrelated work. Note that since the more technical short-term directions have already been exposed in the conclusion of each chapter, we here voluntarily focus on broader prospects.

6.1 Summary of Contributions

Chapter 3 In Chapter 3, we investigated (and extended) the use of *Algorithmic Stability* to derive generalisation bounds, which allow one to relate algorithmic properties of learning algorithms to their generalisation abilities. In the literature, along with other related analyses such as those based on the Vapnik-Chervonenkis dimension or on Rademacher averages, the analysis based on algorithmic stability has essentially focused on relating *scalar-based* empirical performance measures (e.g. the empirical risk, as defined in the introduction) to their expectations (e.g. the true risk). Our contribution is a first step towards going beyond the use of scalar-based performance measures. Indeed, in the context of multiclass prediction, we extended the traditional framework to a finer-grain quantity of interest, namely the *Confusion Matrix* and gave elements to derive algorithms that have good generalisation abilities with respect to this performance

measure, of widespread empirical use, but scarcely analysed from a statistical perspective.

As the purpose of our work was to explore the connections between the computational and statistical aspects of Machine Learning, the Algorithmic Stability was an obvious framework to consider. By making use of non-commutative concentration inequalities, we showed that the existing results based on scalar risks, carried over to the more complex setting we analysed. Naturally, we can now wonder if we could extend the framework of the Algorithmic Stability even further. For instance, over the last few years, more complex tasks like *Structured Output Prediction* have drawn more and more attention. For these tasks, it would be worth investigating if tools such as the ones we have developed could be relevant to derive new efficient learning algorithms and establish their generalisation guarantees. The questions underlying this line of research would be, e.g., the definition of new performance measures and/or the derivation of dimension-free concentration inequalities.

Chapter 4 In Chapter 4, we derived a novel learning algorithm, in the specific context of regression, that aims at tackling two non-trivial issues related to the use of kernel methods: a) learning the kernel function, i.e. a representation of the data that directly serves the learning task at hand, along with the predictor function itself, and b) deriving a learning algorithm that leverages the structure of the learning problem for a better computational efficiency. Building on the well-studied general idea that a low-rank structure in the representation of the data could make it possible to use algorithmic tricks that lower the runtime complexity of learning procedures, we designed a learning problem that explicitly enforced this low-rank structure in the learnt kernel.

Among others, this work has raised a question whose interest goes beyond the strict scope of learning a kernel for regression tasks. At each iteration, our *stochastic* algorithm updates a randomly-picked coordinate of the current iterate. As described in the chapter, this procedure performs each iteration at a very low cost. However, it turns out that in practice, many iterations are useless because the selected coordinate does not make it possible to get closer to the minimiser of the functional at hand. On the other hand, a greedy approach could make it possible to pick a more suitable coordinate, but would induce some additional computations at each coordinate, so that a natural trade-off arises between the number of iterations needed to reach a solution, and the cost of these iterations. Determining how to set this trade-off in order to increase the

computational efficiency of the optimisation procedure is not trivial but is of primary importance. Studying this trade-off more formally, in a more general setting than this of Low-Rank Kernel Learning for Regression, is therefore a direction of choice for future work and which has been partially explored in Chapter 5.

Chapter 5 Chapter 5 builds on an important message, given by Bottou and Bousquet [2007], about large-scale problems: one should consider solving optimisation problems with a limited precision and pay a closer attention to the computational cost of the algorithm at hand, for a given generalisation error. To address these issues, we describe how to explicitly take into account and minimise the computational cost of a class of optimisation algorithms for non-smooth convex problems, namely the inexact proximal methods, under the constraint that the distance between the computed solution and the exact minimizer of the problem is no larger than a given precision.

The results obtained in the chapter show that the strategies leading to optimisation algorithms with the faster convergence rates may be fundamentally different from those leading to algorithms with lower computational cost to achieve a given precision – note the aforementioned opposition between rates and actual cost. Once again, those results, highlighting the gap between classical asymptotic analyses and our novel finite-time analysis, were obtained in the specific context of inexact proximal method. How those results could extend to different optimisation settings remains an open question. An important future direction is to study further if this type of finite-time analyses could help designing computationally efficient algorithms to solve more general large-scale learning problems.

6.2 Broader Prospects

The different contributions of this thesis have addressed a few issues depicted in the introduction. On different levels, each chapter has dug up existing connections between the computational and statistical aspects of Machine Learning and, to some extent, we were capable of leveraging those connections to derive and analyse efficient algorithms. However, among the most important prospects of each of those chapters, the question of understanding how our results could generalise to broader classes of problems (e.g. structured output prediction) or algorithms (e.g. general First-Order Methods in Optimisation) remains open.

Moreover, the diversity of the studied settings and of the natures of the tools that were used also suggests that our initial questions can be addressed from many fundamentally different angles. In fact, some of them have already been explored in the literature. A recent line of work has raised interesting issues regarding connections between computational and statistical aspects of learning procedures. In Orecchia and Mahoney [2011], the connections between the notion of *approximate computation* and *implicit regularisation* are explored. More precisely, a very specific leaning problem is studied, namely the computation of the smallest non-trivial eigenvector of a graph Laplacian. For this problem, several existing algorithms that compute an approximate solution are described. For each of these, the authors are able to express the implicitly regularised problem that the algorithm solves exactly. The question of the generalisation of those results in more diverse settings is also left open, though it is partially addressed in [Mahoney, 2012], where a more general discussion is conducted about approximate computation and statistical regularisation.

The numerical simulations conducted in Chapter 5 empirically suggest, in two different settings, that using a constant number of inner iterations with inexact proximal methods, cause the optimisation algorithm to converge to approximate solutions of the exact problem. Mimicking the discussion conducted in Orecchia and Mahoney [2011], it would be interesting to study the regularity properties of those approximate solutions. If relevant, determining if those approximate solutions are the exact solutions of an implicitly regularised problem would be of great interest.

When it comes to relating the computational and statistical aspects of Machine Learning, the most significant and pioneer example certainly is the *Probably Approximatively Correct (PAC)-Learnability* framework [Valiant, 1984]. Developing a *Theory of the Learnable*, this seminal work characterises the class of concepts that can be *approximatively correctly* learnt, with *high probability* (over the sampling of the training data). In that theory, the importance of the runtime complexity is paramount. Building on the idea that humans implicitly can learn concepts "in polynomial time", Leslie Valiant developed this formal discrimination between what *can* or *cannot* be learnt with an algorithm that runs in time that is *polynomial* in the number of data n (and other critical quantities), to achieve a given generalisation error. Interestingly, the separation criterion he proposed relies on a runtime analysis and is essentially computational (i.e. not statistical). It has allowed a much deeper understanding of the intrinsic limits of Machine Learning.

However, this pioneering work suffers from the binary separation between algorithms that run in polynomial versus those that run in exponential time. The amount of data available in the recent large-scale problems prohibits the use of "polynomial time" algorithms if the degree of the polynomial is too high. It is now understandable that a runtime with a dependence higher than linear on the number of data is not reasonable for handling such datasets. As a consequence, it would be extremely useful to develop a more detailed theory of the learnable that would characterise the concepts that can be learnt, given some degree in the polynomial runtime. For instance, characterising the class of concepts that can be learnt with a runtime that is linear with the number of data would certainly help understanding the intrinsic limits of *Large-Scale* Learning.

Even though such a precise characterisation sounds like a very distant objective, it is worth mentioning that some recent work in this line already exist (see [Agarwal, 2012] and the references therein, for instance). Also, Shalev-Schwartz et al. [2012] have tried to answer the closely related following question. It is known that the more data you have for training your predictor, the higher accuracy you can expect. Now, does it mean that, to obtain a predictor with a fixed accuracy, using more data may actually can decrease the required runtime? Though the question remains partially open, the authors of Shalev-Schwartz et al. [2012] analyse different learning settings where *more data* actually means *less work* and give elements towards a positive answer, which urge for further explorations.

The seminal analysis of Bottou and Bousquet [2007] has summoned Machine Learners to unravel the connections between the Statistical and Computational aspects of Learning. However, we have seen that such connections could be observed at many different levels. We hope that the present thesis contributes to that trend and sheds light on some of these levels.

Chapter 7

Appendices

Contents

7.1 Proof of Theorem 5 87
7.2 Matrix Inversion Formulas 91
7.3 Proof of Proposition 7 91
7.4 Proof of Proposition 8 95
7.5 Proof of Proposition 9 96
7.6 Proof of Proposition 10 99

7.1 Proof of Theorem 5

To ease the readability, we introduce additional notation:

$$\mathcal{L}_q := \mathbb{E}_{X|q} L(\mathcal{A}_{\mathbf{Z}}, X, q), \quad \hat{\mathcal{L}}_q := L_q(\mathcal{A}_{\mathbf{Z}}, \boldsymbol{X}, \boldsymbol{y}),$$
$$\mathcal{L}_q^i := \mathbb{E}_{X|q} L(\mathcal{A}_{\mathbf{Z}^i}, X, q), \quad \hat{\mathcal{L}}_q^i := L_q(\mathcal{A}_{\mathbf{Z}^i}, \boldsymbol{X}^i, \boldsymbol{y}^i),$$
$$\mathcal{L}_q^{\setminus i} := \mathbb{E}_{X|q} L(\mathcal{A}_{\mathbf{Z}^{\setminus i}}, X, q), \quad \hat{\mathcal{L}}_q^{\setminus i} := L_q(\mathcal{A}_{\mathbf{Z}^{\setminus i}}, \boldsymbol{X}^{\setminus i}, \boldsymbol{y}^{\setminus i}).$$

After using the triangle inequality in (3.2), we need to provide a bound on each summand. To get the result, we will, for each q, fix the X_k such that $y_k \neq q$ and work with functions of m_q variables. Then, we will apply Theorem 4 for each

$$H_q(\boldsymbol{X}_q, \boldsymbol{y}_q) := \mathfrak{D}(\mathcal{L}_q) - \mathfrak{D}(\hat{\mathcal{L}}_q).$$

To do so, we prove the following lemma

Lemma 3. $\forall q, \forall i, y_i = q$

$$(H_q(\boldsymbol{Z}_q) - H_q(\boldsymbol{Z}_q^i))^2 \preccurlyeq \left(\frac{4B}{m_q} + \frac{\sqrt{Q}M}{m_q}\right)^2 I.$$

Proof. This is a proof that works in 2 steps.

Note that

$$\begin{aligned}\|H_q(\boldsymbol{X}_q, \boldsymbol{y}_q) - H_q(\boldsymbol{X}_q^i, \boldsymbol{y}_q^i)\| &= \|\mathfrak{D}(\mathcal{L}_q) - \mathfrak{D}(\hat{\mathcal{L}}_q) - \mathfrak{D}(\mathcal{L}_q^i) + \mathfrak{D}(\hat{\mathcal{L}}_q^i)\| \\ &= \|\mathcal{L}_q - \hat{\mathcal{L}}_q - \mathcal{L}_q^i + \hat{\mathcal{L}}_q^i\| \leq \|\mathcal{L}_q - \mathcal{L}_q^i\| + \|\hat{\mathcal{L}}_q - \hat{\mathcal{L}}_q^i\|.\end{aligned}$$

Step 1: bounding $\|\mathcal{L}_q - \mathcal{L}_q^i\|$. We can trivially write:

$$\|\mathcal{L}_q - \mathcal{L}_q^i\| \leq \|\mathcal{L}_q - \mathcal{L}_q^{\setminus i}\| + \|\mathcal{L}_q^i - \mathcal{L}_q^{\setminus i}\|$$

Taking advantage of the stability of $\mathcal{A}$:

$$\begin{aligned}\|\mathcal{L}_q - \mathcal{L}_q^{\setminus i}\| &= \left\|\mathbb{E}_{X|q}\left[L(\mathcal{A}_{\boldsymbol{Z}}, X, q) - L(\mathcal{A}_{\boldsymbol{Z}^{\setminus i}}, X, q)\right]\right\| \\ &\leq \mathbb{E}_{X|q}\left\|L(\mathcal{A}_{\boldsymbol{Z}}, X, q) - L(\mathcal{A}_{\boldsymbol{Z}^{\setminus i}}, X, q)\right\| \\ &\leq \frac{B}{m_q},\end{aligned}$$

and the same holds for $\|\mathcal{L}_q^i - \mathcal{L}_q^{\setminus i}\|$, i.e. $\|\mathcal{L}_q^i - \mathcal{L}_q^{\setminus i}\| \leq B/m_q$. Thus, we have:

$$\|\mathcal{L}_q - \mathcal{L}_q^i\| \leq \frac{2B}{m_q}. \tag{7.1}$$

Step 2: bounding $\|\hat{\mathcal{L}}_q - \hat{\mathcal{L}}_q^i\|$. This is a little trickier than the first step.

$$\begin{aligned}\|\hat{\mathcal{L}}_q - \hat{\mathcal{L}}_q^i\| &= \left\|L_q(\mathcal{A}_{\mathbf{Z}}, \boldsymbol{Z}) - L_q(\mathcal{A}_{\mathbf{Z}^i}, \boldsymbol{Z}^i)\right\| \\ &= \frac{1}{m_q}\Big\| \sum_{k:k\neq i, y_k=q} \Big(L(\mathcal{A}_{\mathbf{Z}}, X_k, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_k, q)\Big) \\ &\quad + L(\mathcal{A}_{\mathbf{Z}}, X_i, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q)\Big\| \\ &\leq \frac{1}{m_q}\Big\| \sum_{k:k\neq i, y_k=q} \Big(L(\mathcal{A}_{\mathbf{Z}^i}, X_k, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_k, q)\Big)\Big\| \\ &\quad + \frac{1}{m_q}\Big\|L(\mathcal{A}_{\mathbf{Z}}, X_i, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q)\Big\|\end{aligned}$$

Using the stability argument as before, we have:

$$\begin{aligned}&\Big\| \sum_{k:k\neq i, y_k=q} \Big(L(\mathcal{A}_{\mathbf{Z}}, X_k, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_k, q)\Big)\Big\| \\ &\qquad \leq \sum_{k:k\neq i, y_k=q} \|L(\mathcal{A}_{\mathbf{Z}}, X_k, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_k, q)\| \leq \sum_{k:k\neq i, y_k=q} 2\frac{B}{m_q} \leq 2B.\end{aligned}$$

On the other hand, we observe that

$$\left\| L(\mathcal{A}_{\mathbf{Z}}, X_i, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q) \right\| \leq \sqrt{Q}M.$$

Indeed, the matrix $\Delta := L(\mathcal{A}_{\mathbf{Z}}, X_i, q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q)$ is a matrix that is zero except for (possibly) its qth row, that we may call $\boldsymbol{\delta}_q$. Thus:

$$\|\Delta\| = \sup_{\boldsymbol{v}:\|\boldsymbol{v}\|_2 \leq 1} \|\Delta \boldsymbol{v}\|_2 = \sup_{\boldsymbol{v}:\|\boldsymbol{v}\|_2 \leq 1} \|\boldsymbol{\delta}_q \cdot \boldsymbol{v}\| = \|\boldsymbol{\delta}_q\|_2,$$

where $\boldsymbol{v}$ is a vector of dimension Q. Since each of the Q elements of $\boldsymbol{\delta}_q$ is in the range $[-M; M]$, we get that $\|\boldsymbol{\delta}_q\|_2 \leq \sqrt{Q}M$.

This allows us to conclude that

$$\|\hat{\mathcal{L}}_q - \hat{\mathcal{L}}_q^i\| \leq \frac{2B}{m_q} + \frac{\sqrt{Q}M}{m_q} \tag{7.2}$$

Combining (7.1) and (7.2) we just proved that, for all i such that $y_i = q$

$$(H_q(\boldsymbol{Z}_q) - H_q(\boldsymbol{Z}_q^i))^2 \preccurlyeq \left(\frac{4B}{m_q} + \frac{\sqrt{Q}M}{m_q} \right)^2 I.$$

□ □

We then establish the following Lemma

Lemma 4. $\forall q$,

$$\mathbb{P}_{\boldsymbol{X}|\boldsymbol{y}} \left\{ \|\mathcal{L}_q - \hat{\mathcal{L}}_q\| \geq t + \|\mathbb{E}_{\boldsymbol{X}|\boldsymbol{y}}[\mathcal{L}_q - \hat{\mathcal{L}}_q]\| \right\} \leq 2Q \exp\left\{ -\frac{t^2}{8\left(\frac{4B}{\sqrt{m_q}} + \frac{\sqrt{Q}M}{\sqrt{m_q}}\right)^2} \right\}.$$

Proof. Given the previous Lemma, Theorem 4, when applied on $H_q(\boldsymbol{X}_q, y_q) = \mathfrak{D}(\mathcal{L}_q - \hat{\mathcal{L}}_q)$ gives

$$\sigma_q^2 = \left(\frac{4B}{m_q} + \frac{\sqrt{Q}M}{\sqrt{m_q}} \right)^2$$

to give, for $t > 0$:

$$\mathbb{P}_{\boldsymbol{X}|\boldsymbol{y}} \left\{ \|\mathcal{L}_q - \hat{\mathcal{L}}_q - \mathbb{E}[\mathcal{L}_q - \hat{\mathcal{L}}_q]\| \geq t \right\} \leq 2Q \exp\left\{ -\frac{t^2}{8\left(\frac{4B}{m_q} + \frac{\sqrt{Q}M}{\sqrt{m_q}}\right)^2} \right\},$$

which, using the triangle inequality

$$|\|A\| - \|B\|| \leq \|A - B\|,$$

gives the result. □ □

Finally, we observe

Lemma 5. *$\forall q$,*

$$\mathbb{P}_{\boldsymbol{X}|\boldsymbol{y}}\left\{\|\mathcal{L}_q - \hat{\mathcal{L}}_q\| \geq t + \frac{2B}{m_q}\right\} \leq 2Q \exp\left\{-\frac{t^2}{8\left(\frac{4B}{\sqrt{m_q}} + \frac{\sqrt{Q}M}{\sqrt{m_q}}\right)^2}\right\}.$$

Proof. It suffices to show that

$$\left\|\mathbb{E}[\mathcal{L}_q - \hat{\mathcal{L}}_q]\right\| \leq \frac{2B}{m_q},$$

and to make use of the previous Lemma. We note that for any i such that $y_i = q$, and for X_i' distributed according to $D_{X|q}$:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{X}|\boldsymbol{y}}\hat{\mathcal{L}}_q = \mathbb{E}_{\boldsymbol{X}|\boldsymbol{y}} L_q(\mathcal{A}_{\mathbf{Z}}, \boldsymbol{X}, \boldsymbol{y}) &= \frac{1}{m_q}\sum_{j:y_j=q} \mathbb{E}_{\boldsymbol{X}|\boldsymbol{y}} L(\mathcal{A}_{\mathbf{Z}}, X_j, q) \\ &= \frac{1}{m_q}\sum_{j:y_j=q} \mathbb{E}_{\boldsymbol{X},X_i'|\boldsymbol{y}} L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q) = \mathbb{E}_{\boldsymbol{X},X_i'|\boldsymbol{y}} L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q).\end{aligned}$$

Hence, using the stability argument,

$$\begin{aligned}\|\mathbb{E}[\mathcal{L}_q - \hat{\mathcal{L}}_q]\| &= \left\|\mathbb{E}_{\boldsymbol{X},X_i'|\boldsymbol{y}}\left[L(\mathcal{A}_{\mathbf{Z}}, X_i', q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q)\right]\right\| \\ &\leq \mathbb{E}_{\boldsymbol{X},X_i'|\boldsymbol{y}}\left\|L(\mathcal{A}_{\mathbf{Z}}, X_i', q) - L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q)\right\| \\ &\leq \mathbb{E}_{\boldsymbol{X},X_i'|\boldsymbol{y}}\left\|L(\mathcal{A}_{\mathbf{Z}}, X_i', q) - L(\mathcal{A}_{\mathbf{Z}^{\setminus i}}, X_i', q)\right\| \\ &\quad + \mathbb{E}_{\boldsymbol{X},X_i'|\boldsymbol{y}}\left\|L(\mathcal{A}_{\mathbf{Z}^i}, X_i', q) - L(\mathcal{A}_{\mathbf{Z}^{\setminus i}}, X_i', q)\right\| \\ &\leq \frac{2B}{m_q}.\end{aligned}$$

This inequality in combination with the previous lemma provides the result. □ □

We are now set to make use of a union bound argument:

$$\begin{aligned}\mathbb{P}\left\{\exists q : \|\mathcal{L}_q - \hat{\mathcal{L}}_q\| \geq t + \frac{2B}{m_q}\right\} &\leq \sum_{q\in\mathcal{Y}} \mathbb{P}\left\{\exists q : \|\mathcal{L}_q - \hat{\mathcal{L}}_q\| \geq t + \frac{2B}{m_q}\right\} \\ \leq 2Q\sum_q \exp\left\{-\frac{t^2}{8\left(\frac{4B}{\sqrt{m_q}} + \frac{\sqrt{Q}M}{\sqrt{m_q}}\right)^2}\right\} &\leq 2Q^2 \max_q \exp\left\{-\frac{t^2}{8\left(\frac{4B}{\sqrt{m_q}} + \frac{\sqrt{Q}M}{\sqrt{m_q}}\right)^2}\right\}\end{aligned}$$

According to our definition m^*, we get

$$\mathbb{P}\left\{\exists q : \|\mathcal{L}_q - \hat{\mathcal{L}}_q\| \geq t + \frac{2B}{m_q}\right\} \leq 2Q^2 \exp\left\{-\frac{t^2}{8\left(\frac{4B}{\sqrt{m^*}} + \frac{\sqrt{Q}M}{\sqrt{m^*}}\right)^2}\right\}.$$

Setting the right hand side to δ, gives the result of Theorem 5.

7.2 Matrix Inversion Formulas

Theorem 8. (Woodbury matrix inversion formula [Woodbury, 1950]) *Let n and m be positive integers, $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{m\times m}$ be non-singular matrices and let $U \in \mathbb{R}^{n\times m}$ and $V \in \mathbb{R}^{m\times n}$ be two matrices. If $C^{-1}+VA^{-1}U$ is non-singular then so is $A+UCV$ and:*

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}.$$

Theorem 9. (Matrix inversion with added column) *Given m, integer and $M \in \mathbb{R}^{(n+1)\times(n+1)}$ partitioned as:*

$$M = \begin{pmatrix} A & \boldsymbol{b} \\ \boldsymbol{b}^\top & c \end{pmatrix}, \quad \text{where } A \in \mathbb{R}^{n\times n}, \boldsymbol{b} \in \mathbb{R}^n \text{ and } c \in \mathbb{R}.$$

If A is non-singular and $c - \boldsymbol{b}^\top A^{-1}\boldsymbol{b} \neq 0$, then M is non-singular and the inverse of M is given by

$$M^{-1} = \begin{pmatrix} A^{-1} + \frac{1}{k}A^{-1}\boldsymbol{b}\boldsymbol{b}^\top A^{-1} & -\frac{1}{k}A^{-1}\boldsymbol{b} \\ -\frac{1}{k}\boldsymbol{b}^\top A^{-1} & \frac{1}{k} \end{pmatrix}, \tag{7.3}$$

where $k = c - \boldsymbol{b}^\top A^{-1}\boldsymbol{b}$.

7.3 Proof of Proposition 7

In this scenario, we use non-accelerated outer iterations and sublinear inner iterations. Our optimisation problem thus reads:

$$\min_k \min_{\{l_i\}_{i=1}^k} C_{\text{in}} \sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } \frac{L}{2k}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k \sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2 \leq \rho.$$

Let us first examine the constraint.

$$\begin{aligned} &\frac{L}{2k}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k \sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2 \leq \rho \\ \Leftrightarrow &\|x_0 - x^*\| + 3\sum_{i=1}^k \sqrt{\frac{2A_i}{Ll_i^\alpha}} \leq \sqrt{\frac{2k\rho}{L}} \\ \Leftrightarrow &\sum_{i=1}^k \sqrt{\frac{A_i}{l_i^\alpha}} \leq \frac{\sqrt{L}}{3\sqrt{2}}\left(\sqrt{\frac{2k\rho}{L}} - \|x_0 - x^*\|\right) \end{aligned}$$

As a first remark, this constraint can be satisfied only if

$$k \geq \frac{L}{2\rho}\|x_0 - x^*\|^2.$$

However this always holds as this only implies that the number of outer iterations k is larger than the amount we would need if the proximity operator could be computed exactly.

Let us recall that for any i, A_i is such that $\epsilon_i \leq A_i/l_i^\alpha$. For most iterative optimisation methods, the tightest bounds (of this form) on the error are obtained for constants A_i depending on: a) properties of the objective function at hand, b) the initialisation. To mention an example we have already introduced, for basic proximal methods, one can choose

$$A_i = \frac{L}{2l_i}\|(x_k)_0 - x_k^*\|,$$

where $(x_k)_0$ is the initialisation for our inner-problem at outer-iteration k and x_k^* the optimal of this problem. As the problem seems intractable in the most general case, we will first assume that $\forall i, A_i = A$. This only implies that we don't introduce any prior knowledge on $\|(x_k)_0 - x_k^*\|$ at each iteration. This is reasonable if, at each outer-iteration, we randomly initialise $(x_k)_0$ but may lead to looser bounds if we use wiser strategies such as warm starts.

With that new assumption on A_i, one can state that the former constraint will hold if and only if:

$$\sum_{i=1}^{k}\sqrt{\frac{1}{l_i^\alpha}} \leq \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{2k\rho}{L}} - \|x_0 - x^*\|\right).$$

Let us first solve the problem of finding the $\{l_i\}_{i=1}^k$ for some fixed k. We need to solve:

$$\underset{\{l_i\}_{i=1}^k \in \mathbb{N}^{*k}}{\operatorname{argmin}} C_{\text{in}}\sum_{i=1}^{k} l_i + kC_{\text{out}} \qquad \text{s.t.} \quad \sum_{i=1}^{k}\sqrt{\frac{1}{l_i^\alpha}} \leq \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{2k\rho}{L}} - \|x_0 - x^*\|\right) := C_k,$$

which is equivalent to solving:

$$\underset{\{l_i\}_{i=1}^k \in \mathbb{N}^{*k}}{\operatorname{argmin}} \sum_{i=1}^{k} l_i \qquad \text{s.t.} \quad \sum_{i=1}^{k}\sqrt{\frac{1}{l_i^\alpha}} \leq C_k.$$

Remark 7. $l_i \in \mathbb{N}^{*k} \Rightarrow \sqrt{\frac{1}{l_i^\alpha}} \in]0,1] \Rightarrow \sum_{i=1}^k \sqrt{\frac{1}{l_i^\alpha}} \leq k$. So, if $C_k \geq k$, then the solution of the constrained problem is the solution of the unconstrained problem. In that case, the trivial solution is $l_i = 1, \forall i$. Moreover, if $l_i = 1, \forall i$ is the solution of the constrained problem, then $\sum_{i=1}^k \sqrt{\frac{1}{l_i^\alpha}} = k \leq C_k$. As a consequence, the solution of the unconstrained problem is the solution of the constrained problem *if and only if* $C_k \geq k$.

We then have two cases to consider:

Case 1: $C_k \geq k$ As stated before, the optimum will be trivially reached for $l_i = 1, \forall i$. Now, we need to find the optimal over k. It consists in finding:

$$\min_{k \in \mathbb{N}^*} k(C_{\text{in}} + C_{\text{out}}) \qquad \text{s.t. } C_k \geq k.$$

Let us have a look at the constraint.

$$\begin{aligned} C_k \geq k &\Leftrightarrow \frac{\sqrt{L}}{3\sqrt{2A}} \left(\sqrt{\frac{2k\rho}{L}} - \|x_0 - x^*\| \right) \geq k \\ &\Leftrightarrow \sqrt{\frac{2k\rho}{L}} \geq \frac{3\sqrt{2A}}{\sqrt{L}} k + \|x_0 - x^*\| \\ &\Leftrightarrow \left(\sqrt{k} - \frac{\sqrt{\rho}}{6\sqrt{A}} \right)^2 \leq \frac{\rho}{36A} - \frac{\sqrt{L}\|x_0 - x^*\|}{3\sqrt{2A}} \end{aligned}$$

Then:

- if $\frac{\rho}{36A} < \frac{\sqrt{L}\|x_0 - x^*\|}{3\sqrt{2A}}$ then there is no solution (i.e. $C_k < k, \forall k$).
- if $\frac{\rho}{36A} \geq \frac{\sqrt{L}\|x_0 - x^*\|}{3\sqrt{2A}}$ then, the constraint holds for

$$k \in \left[\left(\frac{\sqrt{\rho}}{6\sqrt{A}} - \sqrt{\frac{\rho}{36A} - \frac{\sqrt{L}\|x_0 - x^*\|}{3\sqrt{2A}}} \right)^2, \left(\frac{\sqrt{\rho}}{6\sqrt{A}} + \sqrt{\frac{\rho}{36A} - \frac{\sqrt{L}\|x_0 - x^*\|}{3\sqrt{2A}}} \right)^2 \right]$$

. The optimum will then be achieved for the smallest integer (if exists) larger than $\left(\frac{\sqrt{\rho}}{6\sqrt{A}} - \sqrt{\frac{\rho}{36A} - \frac{\sqrt{L}\|x_0 - x^*\|}{3\sqrt{2A}}} \right)^2$ and smaller than $\left(\frac{\sqrt{\rho}}{6\sqrt{A}} + \sqrt{\frac{\rho}{36A} - \frac{\sqrt{L}\|x_0 - x^*\|}{3\sqrt{2A}}} \right)^2$.

Case 2: $C_k \leq k$ As remark 7 shows, the solution of the constrained problem is different from the unconstrained one. The solution of this *integer* optimisation problem is hard to compute. In a first step, we may relax the problem and solve it as if $\{l_i\}_{i=1}^k$ were continuous variables taking values into $[1, +\infty[^k$. Because both our objective function and the constraints are continuous with respect to $\{l_i\}_{i=1}^k$, the optimal (over $\{l_i\}_{i=1}^k$) of our problem will precisely lie on the constraint. Our problem now is:

$$\operatorname*{argmin}_{\{l_i\}_{i=1}^k \in [1, +\infty[^k} \sum_{i=1}^k l_i \qquad \text{s.t. } \sum_{i=1}^k l_i^{-\frac{\alpha}{2}} = C_k.$$

For any $i \in [1, k]$, let $n_i := l_i^{-\frac{\alpha}{2}}$. Our problem becomes:

$$\operatorname*{argmin}_{\{n_i\}_{i=1}^k \in]0,1]^k} \sum_{i=1}^k n_i^{-\frac{2}{\alpha}} \qquad \text{s.t. } \sum_{i=1}^k n_i = C_k.$$

Introducing the Lagrange multiplier $\lambda \in \mathbb{R}$, the Lagrangian of this problem writes:

$$L(\{n_i\}_{i=1}^k, \lambda) := \sum_{i=1}^k n_i^{-\frac{2}{\alpha}} + \lambda \left(\sum_{i=1}^k n_i - C_k \right).$$

And it follows that, $\forall i \in [1, k]$, when the optimum $\{n_i^*\}_{i=1}^k$ is reached:

$$\frac{\partial L}{\partial n_i} = 0 \Leftrightarrow n_i^* = \left(\frac{\alpha\lambda}{2} \right)^{\frac{1}{-\frac{2}{\alpha}-1}}$$

And now, plugging into our constraint:

$$\sum_{i=1}^k n_i^* = C_k \Rightarrow \lambda = \frac{2}{\alpha} \left(\frac{C_k}{k} \right)^{-\frac{2}{\alpha}-1}.$$

Hence, for any $i \in [1, k]$, $n_i^* = \frac{C_k}{k}$.

As $C_k \leq k$, it is clear that $\forall p, n_p^* \in]0, 1]$ and we have, $\forall i, l_i^* = \left(\frac{C_k}{k} \right)^{-\frac{2}{\alpha}}$.

We can now plug the optimal l_i^* in our first problem and we now need to find the optimal k^* such that:

$$\begin{aligned}
k^* &= \operatorname*{argmin}_{k \in \mathbb{N}^*} C_{\text{glob}}(k, \{l_i^*\}_{i=1}^k) \\
&= \operatorname*{argmin}_{k \in \mathbb{N}^*} C_{\text{in}} \sum_{i=1}^k l_i^* + kC_{\text{out}} \\
&= \operatorname*{argmin}_{k \in \mathbb{N}^*} C_{\text{in}} \sum_{i=1}^k \left(\frac{C_k}{k} \right)^{-\frac{2}{\alpha}} + kC_{\text{out}} \\
&= \operatorname*{argmin}_{k \in \mathbb{N}^*} k \left(C_{\text{in}} \left(\frac{C_k}{k} \right)^{-\frac{2}{\alpha}} + C_{\text{out}} \right).
\end{aligned}$$

Once again, we can relax this integer optimisation problem into a continuous one, assuming $k \in \mathbb{R}^+$. It directly follows that the solution of that relaxed problem is reached when the derivative (w.r.t. k) of $C_{\text{glob}}(k, \{l_i^*\}_{i=1}^k)$ equals 0. The derivative can be easily computed:

$$\frac{\partial C_{\text{glob}}(k, \{l_i^*\}_{i=1}^k)}{\partial k} = C_{\text{in}} \left(\left(\frac{2}{\alpha} + 1 \right) k^{\frac{2}{\alpha}} C_k^{-\frac{2}{\alpha}} - \frac{2}{\alpha} C_k' C_k^{-\frac{2}{\alpha}-1} k^{\frac{2}{\alpha}+1} \right) + C_{\text{out}},$$

where C_k' is the derivative of C_k w.r.t. k:

$$C_k' = \frac{\sqrt{\rho}}{3\sqrt{A}} k^{-\frac{1}{2}}.$$

However, giving an analytic form of that zero is difficult. But using any numeric solver, it is very easy to find a very good approximation of k^*. As described in Remark 6, this allows us to exactly retrieve the exact integer minimiser.

7.4 Proof of Proposition 8

In this scenario, we use non-accelerated outer iterations and linear inner iterations. Our optimisation problem thus reads:

$$\min_k \min_{\{l_i\}_{i=1}^k} C_{\text{in}} \sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } \frac{L}{2k}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k \sqrt{\frac{2A_i(1-\gamma)^{l_i}}{L}}\right)^2 \leq \rho.$$

We consider $A_i = A$. The error in the ith inner iteration reads:

$$\epsilon_i = A(1-\gamma)^{l_i}. \tag{7.4}$$

Hence the corresponding bound on the error:

$$\rho_k \leq \frac{L}{2k}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k \sqrt{\frac{2A(1-\gamma)^{l_i}}{L}}\right)^2. \tag{7.5}$$

Problem in $\{l_i\}$ boils down to:

$$\operatorname*{argmin}_{\{l_i\}_{i=1}^k \in \mathbb{N}^{*k}} \sum_{i=1}^k l_i \qquad \text{s.t. } \sum_{i=1}^k (1-\gamma)^{\frac{l_i}{2}} \leq C_k,$$

still with $C_k = \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{2k\rho}{L}} - \|x_0 - x^*\|\right)$.

Case 1: $C_k \geq k\sqrt{1-\gamma}$ identical except for the threshold, which will also impact the interval for k^*.

Case 2: $C_k \leq k\sqrt{1-\gamma}$ For any $i \in [1,k]$, let $n_i := (1-\gamma)^{\frac{l_i}{2}}$. Our problem becomes:

$$\operatorname*{argmin}_{\{n_i\}_{i=1}^k \in]0,\sqrt{1-\gamma}]^k} -\sum_{i=1}^k \ln n_i \qquad \text{s.t. } \sum_{i=1}^k n_i = C_k.$$

Writing again the Lagrangian of this new problem, we obtain the same result: for any $i \in [1,k]$,

$$n_i^* = \frac{C_k}{k}.$$

This leads to

$$l_i^* = \frac{2\ln\left(\frac{C_k}{k}\right)}{\ln(1-\gamma)}.$$

Following the same reasoning, we now plug this analytic solution of the first optimisation problem into the second one. This leads to:

$$k^* = \underset{k\in\mathbb{N}^*}{\operatorname{argmin}}\, k\left(\frac{2C_{\text{in}}}{\ln(1-\gamma)}\ln\left(\frac{C_k}{k}\right)+C_{\text{out}}\right)$$

This time, the derivative of the continuous relaxation writes:

$$\frac{\partial C_{\text{glob}}(k,\{l_i^*\}_{i=1}^k)}{\partial k} = \frac{2C_{\text{in}}}{\ln(1-\gamma)}\left(\ln\frac{C_k}{k}+\frac{kC_k'}{C_k}-1\right)+C_{\text{out}},$$

where C_k' is the derivative of C_k w.r.t. k:

$$C_k' = \frac{\sqrt{\rho}}{3\sqrt{A}}k^{-\frac{1}{2}}.$$

The optimum k^* of our problem is the (unique) zero of that derivative.

7.5 Proof of Proposition 9

In this scenario, we use accelerated outer iterations and sublinear inner iterations. Our optimisation problem thus reads:

$$\min_k \min_{\{l_i\}_{i=1}^k} C_{\text{in}}\sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } \frac{L}{(k+1)^2}\left(\|x_0-x^*\|+3\sum_{i=1}^k i\sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2 \le \rho.$$

We consider $A_i = A$. The error in the ith inner iteration reads:

$$\epsilon_i = \frac{A}{l_i^\alpha}. \tag{7.6}$$

Similarly, for the accelerated case, we have:

$$\rho_k \le \frac{2L}{(k+1)^2}\left(\|x_0-x^*\|+3\sum_{i=1}^k i\sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2. \tag{7.7}$$

Those problems can naturally be extended with the use of accelerated schemes and we get this "error-oriented" problem:

$$\min_{k,\{l_i\}_{i=1}^k} C_{\text{in}}\sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } \frac{2L}{(k+1)^2}\left(\|x_0-x^*\|+3\sum_{i=1}^k i\sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2 \le \rho.$$

We will follow the same reasoning as for the non-accelerated case. We will consider this optimisation problem:

$$\min_k \min_{\{l_i\}_{i=1}^k} C_{\text{in}} \sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } \frac{2L}{(k+1)^2}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k i\sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2 \leq \rho.$$

Let us first have a look at the constraint.

$$\begin{aligned}
&\frac{2L}{(k+1)^2}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k i\sqrt{\frac{2A_i}{Ll_i^\alpha}}\right)^2 \leq \rho \\
\Leftrightarrow &\|x_0 - x^*\| + 3\sum_{i=1}^k i\sqrt{\frac{2A_i}{Ll_i^\alpha}} \leq \sqrt{\frac{\rho}{2L}}(k+1) \\
\Leftrightarrow &\sum_{i=1}^k i\sqrt{\frac{A_i}{l_i^\alpha}} \leq \frac{\sqrt{L}}{3\sqrt{2}}\left(\sqrt{\frac{\rho}{2L}}(k+1) - \|x_0 - x^*\|\right)
\end{aligned}$$

As in the former case, this can only hold if $(k+1) \geq \sqrt{\frac{2L}{\rho}}\|x_0 - x^*\|$ which is trivial.

We will now assume again that $A_i = A$ for any i. As earlier, we first solve the following problem in $\{l_i\}_{i=1}^k$:

$$\operatorname*{argmin}_{\{l_i\}_{i=1}^k \in \mathbb{N}^{*k}} \sum_{i=1}^k l_i \qquad \text{s.t. } \sum_{i=1}^k i\sqrt{\frac{1}{l_i^\alpha}} \leq \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{\rho}{2L}}(k+1) - \|x_0 - x^*\|\right) =: D_k.$$

Remark 8. $l_i \in \mathbb{N}^{*k} \Rightarrow \sqrt{\frac{1}{l_i^\alpha}} \in]0,1] \Rightarrow \sum_{i=1}^k i\sqrt{\frac{1}{l_i^\alpha}} \leq \frac{k(k+1)}{2}$. So, if $D_k \geq \frac{k(k+1)}{2}$, then the solution of the constrained problem is the solution of the unconstrained problem. In that case, the trivial solution is $l_i = 1, \forall i$. Moreover, if $l_i = 1, \forall i$ is the solution of the constrained problem, then $\sum_{i=1}^k i\sqrt{\frac{1}{l_i^\alpha}} = \frac{k(k+1)}{2} \leq D_k$. As a consequence, the solution of the unconstrained problem is the solution of the constrained problem *if and only if* $D_k \geq \frac{k(k+1)}{2}$.

Case 1: $D_k \geq \frac{k(k+1)}{2}$ As stated before, the optimum will be trivially reached for $l_i = 1, \forall i$. Now, we need to find the optimal over k. It consists in finding:

$$\min_{k \in \mathbb{N}^*} k(C_{\text{in}} + C_{\text{out}}) \qquad \text{s.t. } D_k \geq \frac{k(k+1)}{2}.$$

Let us have a look at this constraint.

$$\begin{aligned}
D_k \geq \frac{k(k+1)}{2} &\Leftrightarrow \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{\rho}{2L}}(k+1) - \|x_0 - x^*\|\right) \geq \frac{k(k+1)}{2} \\
&\Leftrightarrow k^2 + k\left(1 - \frac{\sqrt{\rho}}{3\sqrt{A}}\right) \leq \frac{\sqrt{\rho}}{3\sqrt{A}} - \frac{\sqrt{2L}}{3\sqrt{A}}\|x_0 - x^*\| \\
&\Leftrightarrow \left(k + \frac{1}{2}\left(1 - \frac{\sqrt{\rho}}{3\sqrt{A}}\right)\right)^2 \leq -\frac{\sqrt{2L}}{3\sqrt{A}}\|x_0 - x^*\| + \frac{1}{4}\left(1 + \frac{\sqrt{\rho}}{3\sqrt{A}}\right)^2 =: K.
\end{aligned}$$

Then:

- if $K < 0$ then there is no solution (i.e. $D_k < \frac{k(k+1)}{2}, \forall k$).
- if $K \geq 0$ then, the constraint holds for $k \in \left[\frac{1}{2}\left(\frac{\sqrt{\rho}}{3\sqrt{A}} - 1\right) - \sqrt{K}, \frac{1}{2}\left(\frac{\sqrt{\rho}}{3\sqrt{A}} - 1\right) + \sqrt{K}\right]$. The optimum will then be achieved for the smallest integer (if exists) larger than $\frac{1}{2}\left(\frac{\sqrt{\rho}}{3\sqrt{A}} - 1\right) - \sqrt{K}$ and smaller than $\frac{1}{2}\left(\frac{\sqrt{\rho}}{3\sqrt{A}} - 1\right) + \sqrt{K}$.

Case 2: $D_k \leq \frac{k(k+1)}{2}$ Once again, we fall in the same scenario as in the non-accelerated case. The solution of our problem is different from the unconstrained one and we can relax our discrete optimisation problem to a continuous one. The optimal then precisely lies again on the constraint. We now have:

$$\min_{\{l_i\}_{i=1}^k} \sum_{i=1}^k l_i \qquad \text{s.t.} \sum_{i=1}^k i\sqrt{\frac{1}{l_i^\alpha}} = D_k.$$

For any $i \in [1, k]$, let $n_i := i l_i^{-\frac{\alpha}{2}}$. Our problem becomes:

$$\min_{\{n_i\}_{i=1}^k} \sum_{i=1}^k \left(\frac{n_i}{i}\right)^{-\frac{2}{\alpha}} \qquad \text{s.t.} \sum_{i=1}^k n_i = D_k.$$

The Lagrangian writes:

$$L(\{n_i\}_{i=1}^k, \lambda) := \sum_{i=1}^k \left(\frac{n_i}{i}\right)^{-\frac{2}{\alpha}} + \lambda\left(\sum_{i=1}^k n_i - D_k\right).$$

And it follows that, $\forall i \in [1, k]$, when the optimum $\{n_i^*\}_{i=1}^k$ is reached:

$$\frac{\partial L}{\partial n_i} = 0 \Leftrightarrow n_i^* = i\left(\frac{\alpha\lambda}{2}\right)^{\frac{1}{-\frac{2}{\alpha}-1}}$$

And now, plugging into our constraint:

$$\sum_{i=1}^k n_i^* = D_k \Rightarrow \lambda = \frac{2}{\alpha}\left(\frac{2D_k}{k(k+1)}\right)^{-\frac{2}{\alpha}-1}.$$

Hence, for any $i \in [1, k]$, $n_i^* = \frac{2D_k}{k(k+1)} i$, giving the corresponding $l_i^* = \left(\frac{2D_k}{k(k+1)}\right)^{-\frac{2}{\alpha}}$.

We can now plug the optimal l_i^* in our first problem and we now need to find the optimal k^* such that:

$$\begin{aligned} k^* &= \operatorname*{argmin}_{k \in \mathbb{N}^*} C_{\text{glob}}(k, \{l_i^*\}_{i=1}^k). \\ &= \operatorname*{argmin}_{k \in \mathbb{N}^*} k\left(C_{\text{in}}\left(\frac{2D_k}{k(k+1)}\right)^{-\frac{2}{\alpha}} + C_{\text{out}}\right). \end{aligned}$$

Once again, we can relax this integer optimisation problem into a continuous one, assuming $k \in \mathbb{R}^+$.It directly follows that the solution of that relaxed problem is reached when the derivative (w.r.t. k) of $C_{\text{glob}}(k, \{l_i^*\}_{i=1}^k)$ equals 0.

7.6 Proof of Proposition 10

In this scenario, we use accelerated outer iterations and linear inner iterations. Our optimisation problem thus reads:

$$\min_k \min_{\{l_i\}_{i=1}^k} C_{\text{in}} \sum_{i=1}^k l_i + kC_{\text{out}} \qquad \text{s.t. } \frac{L}{(k+1)^2}\left(\|x_0 - x^*\| + 3\sum_{i=1}^k i\sqrt{\frac{2A_i(1-\gamma)^{l_i}}{L}}\right)^2 \leq \rho.$$

We consider $A_i = A$. The error in the ith inner iteration reads:

$$\epsilon_i = A(1-\gamma)_i^l. \tag{7.8}$$

The problem in $\{l_i\}$ boils down to:

$$\underset{\{l_i\}_{i=1}^k \in \mathbb{N}^{*k}}{\text{argmin}} \sum_{i=1}^k l_i \qquad \text{s.t. } \sum_{i=1}^k i(1-\gamma)^{\frac{l_i}{2}} \leq D_k, \tag{7.9}$$

with $D_k = \frac{\sqrt{L}}{3\sqrt{2A}}\left(\sqrt{\frac{\rho}{2L}}(k+1) - \|x_0 - x^*\|\right)$.

Case 1: $D_k \geq \frac{k(k+1)}{2}\sqrt{1-\gamma}$ identical except for the threshold, which will also impact the interval for k^*.

Case 2: $D_k \leq \frac{k(k+1)}{2}\sqrt{1-\gamma}$

Relaxing Problem (7.9) to real numbers, we want to solve:

$$\underset{\{l_i\}_{i=1}^k \in \mathbb{R}^{+k}}{\text{argmin}} \sum_{i=1}^k l_i \qquad \text{s.t. } \sum_{i=1}^k i(1-\gamma)^{\frac{l_i}{2}} - D_k \leq 0 \tag{7.10}$$

$$1 - l_i \leq 0, \forall i. \tag{7.11}$$

According to the KKT conditions, there exist $\{\mu_i\},\ i = 1, .., k$ and λ, such that the optimum

$\{l_i^*\}$ verify:

$$\text{(stationarity)}\ 1 + \lambda i(1-\gamma)^{\frac{l_i^*}{2}} \ln(\sqrt{1-\gamma}) - \mu_i = 0, \quad \forall i = 1,..,k \tag{7.12}$$

$$\text{(primal feasibility)}\ \sum_{i=1}^{k} i(1-\gamma)^{\frac{l_i^*}{2}} - D_k \leq 0, \tag{7.13}$$

$$1 - l_i^* \leq 0, \quad \forall i = 1,..,k, \tag{7.14}$$

$$\text{(dual feasibility)}\ \lambda \geq 0, \tag{7.15}$$

$$\mu_i \geq 0, \quad \forall i = 1,..,k, \tag{7.16}$$

$$\text{(complementary slackness)}\ \lambda(\sum_{i=1}^{k} i(1-\gamma)^{\frac{l_i^*}{2}} - D_k) = 0, \tag{7.17}$$

$$\mu_i(1 - l_i^*) = 0, \quad \forall i = 1,..,k. \tag{7.18}$$

Eq. (7.15) yields two cases: $\lambda = 0$ or $\lambda > 0$.

$\lambda = 0$ Then Eq. (7.12) yields $\mu_i = 1, \forall i$ thus Eq.(7.18) implies $l_i^* = 1$. All the KKT conditions are thus fulfilled if Eq.(7.13) is, i.e. if

$$D_k \geq \frac{k(k+1)}{2}\sqrt{1-\gamma}.$$

We work here in the case where $D_k \leq \frac{k(k+1)}{2}\sqrt{1-\gamma}$ thus this solution is valid if and only if $D_k = \frac{k(k+1)}{2}\sqrt{1-\gamma}$.

$\lambda > 0$ Again, Eq. (7.15) yields two cases: $\mu_i = 0$ or $\mu_i > 0$.

Subcase 1: $\mu_i > 0$

Then by Eq. (7.18), we have $l_i^* = 1$ and by (7.12) $\mu_i = 1 + \lambda i\sqrt{1-\gamma}\ln(\sqrt{1-\gamma})$. Then $\mu_i > 0$ implies:

$$i < \frac{1}{\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}.$$

Subcase 2: $\mu_i = 0$

Then by Eq. (7.12) we have $1 + \lambda i(1-\gamma)^{\frac{l_i^*}{2}} \ln(\sqrt{1-\gamma}) = 0$, i.e:

$$l_i^* = \frac{\ln\left(i\lambda\ln(\sqrt{\frac{1}{1-\gamma}})\right)}{\ln(\sqrt{\frac{1}{1-\gamma}})}.$$

Since Eq. (7.14) enforces $l_i^* \leq 1$, we have:

$$i \geq \frac{1}{\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}.$$

Conclusion: For $\lambda > 0$, Eq. (7.12), (7.14), (7.15), (7.16) and (7.18) are fullfilled all at once if we set:

$$\begin{aligned}
&\text{For } i = 1..\lceil\frac{1}{\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}\rceil - 1: \quad l_i = 1 && \mu_i = 1 + \lambda i\sqrt{1-\gamma}\ln(\sqrt{1-\gamma}) \\
&\text{For } i = \lceil\frac{1}{\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}\rceil, .., k: \quad l_i = \frac{\ln\left(i\lambda\ln(\sqrt{\frac{1}{1-\gamma}})\right)}{\ln(\sqrt{\frac{1}{1-\gamma}})} && \mu_i = 0.
\end{aligned} \tag{7.19}$$

With these values set for μ_i and l_i^*, let us now find the value of λ.

Computing λ

We need to fulfill Eq. (7.13) and (7.17).

Let us define $M(\lambda) = \lceil\frac{1}{\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}\rceil$.

Note that for $\lambda > \frac{1}{(k+1)\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}$, we have: $0 < M(\lambda) \leq k+1$, and:

- $M(\lambda) = 1 \Leftrightarrow \lambda \geq \frac{1}{\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}$
- $M(\lambda) = n \Leftrightarrow \frac{1}{n\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})} < \lambda < \frac{1}{(n-1)\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}$ for $n = 2, .., k+1$.

Eq. (7.13) and (7.17) are true if and only if

$$\begin{aligned}
D_k &= \sum_{i=1}^{k} i(1-\gamma)^{\frac{l_i^*}{2}} \\
D_k &= \frac{M(\lambda)(M(\lambda)-1)}{2}\sqrt{1-\gamma} + \frac{k - M(\lambda) + 1}{\lambda\ln(\sqrt{\frac{1}{1-\gamma}})}.
\end{aligned}$$

We define $F : \mathbb{R}^{+*} \to \mathbb{R}$ by $F(\lambda) = \frac{M(\lambda)(M(\lambda)-1)}{2}\sqrt{1-\gamma} + \frac{k-M(\lambda)+1}{\lambda\ln(\sqrt{\frac{1}{1-\gamma}})}$.

Examining F on each interval where M is constant, it is easy to see that F is continuous and non-increasing. Moreover F decreases strictly on $[\frac{1}{k\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}, \infty)$, $\lim_{\lambda\to\infty} F = 0$ and F reaches its highest value $\max F = \frac{k(k+1)}{2}\sqrt{1-\gamma}$ on $[\frac{1}{(k+1)\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}, \frac{1}{k\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}]$.

We thus have for all D_k such that $0 < D_k < \frac{k(k+1)}{2}\sqrt{1-\gamma}$, there exists a unique λ such that $F(\lambda) = D_k$ and thus all KKT conditions are fullfilled.

To find this value of λ as a function of D_k, we first find $M(\lambda)$ from D_k. Notice that

$$F\left(\frac{1}{n\lambda\sqrt{1-\gamma}\ln(\sqrt{\frac{1}{1-\gamma}})}\right) = \frac{n(2k+1-n)}{2}\sqrt{1-\gamma}.$$

As $D_k < \frac{k(k+1)}{2}\sqrt{1-\gamma}$, there exists a unique integer n in $1,..,k$ such that

$$\frac{(n-1)(2k+2-n)}{2}\sqrt{1-\gamma} \leq D_k < \frac{n(2k+1-n)}{2}\sqrt{1-\gamma}. \tag{7.20}$$

Then $M(\lambda) = n$ and the KKT conditions are all fulfilled for:

$$\lambda = \frac{k+1-n}{(D_k - \frac{n(n-1)}{2}\sqrt{1-\gamma})\ln(\sqrt{\frac{1}{1-\gamma}})}.$$

In particular:

$$\begin{aligned} &\text{For } i = 1,..,n-1: \quad l_i = 1. \\ &\text{For } i = n,..,k: \qquad l_i = \frac{\ln\left(\frac{k+1-n}{D_k - \frac{n(n-1)}{2}\sqrt{1-\gamma}}\right)}{\ln(\sqrt{\frac{1}{1-\gamma}})} \end{aligned} \tag{7.21}$$

Back to the global problem We now seek to find the the value k^* that minimises the global problem. Outisde of the interval defined in *Case 1*, the global cost is defined by the following. Let us define $n(k)$ as the integer verifying Eq. (7.20). Then

$$\begin{aligned} C_{glob}(k) = kC_{out} + C_{in}(n(k)-1) + \frac{C_{in}(k-n(k)+1)}{\ln(\sqrt{\frac{1}{1-\gamma}})}\ln\left(\frac{k+1-n(k)}{D_k - \frac{n(k)(n(k)-1)}{2}\sqrt{1-\gamma}}\right) \\ + \frac{C_{in}}{\ln(\sqrt{\frac{1}{1-\gamma}})}\ln\left(\frac{k!}{n(k)!}\right). \end{aligned}$$

Bibliography

Agarwal, A. (2012). *Computational Trade-offs in Statistical Learning*. PhD thesis, Department of Computer Science, UC Berkeley. 85

Anthoine, S., Aujol, J.-F., Mlot, C., and Boursier, Y. (2012). Some proximal methods for cbct and pet tomography. inverse problems in imaging. Accepted to Inverse Problems and Imaging. 62, 71

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404. 42

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2011). *Optimization for Machine Learning*, chapter Convex optimization with sparsity-inducing norms, pages 19–54. MIT Press. 60

Bach, F. and Moulines, E. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems (NIPS)*. 72

Baldassarre, L., Morales, J., Argyriou, A., and Pontil, M. (2012). A general framework for structured sparsity via proximal optimization. In *AISTATS*. 60, 61

Beck, A. and Teboulle, M. (2009a). Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Trans. on Im. Proc.*, 18(11):2419–2434. 60, 75, 76

Beck, A. and Teboulle, M. (2009b). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202. 43, 60, 63

Ben-David, S., Loker, D., Srebro, N., and Sridharan, K. (2012). Minimizing the misclassification error rate using a surrogate convex loss. In *Proceedings of the 29th international conference on Machine learning (ICML)*. 15

Bottou, L. and Bousquet, O. (2007). The trade-offs of large scale learning. In *Adv. in Neural Information Processing Systems (NIPS)*. 4, 5, 10, 11, 62, 72, 80, 83, 85

Bottou, L. and Le Cun, Y. (2003). Large scale online learning. In *Advances in Neural Information Processing Systems 16*. 10

Boucheron, S., Bousquet, O., and Lugosi, G. (2005). Theory of classification: a survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375. 1

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *JMLR*, 2:499–526. 11, 22, 27, 28, 29, 31, 32

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. 8, 15

Boyles, L., Korattikara, A., Ramanan, D., and Welling, M. (2011). Statistical tests for optimization efficiency. In *Advances in Neural Information Processing Systems (NIPS)*. 75

Cai, J., Candès, E., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20:1956. 61

Chambolle, A. (2004). An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97. 60, 75

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40:120–145. 61, 62

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178. 9

Chaudhuri, K., Kakade, S., Livescu, K., and Sridharan, K. (2009). Multi-view clustering via canonical correlation analysis. In *Proc. of the 26th Int. Conf. on Machine Learning – ICML 09*, ICML '09, pages 129–136, New York, NY, USA. ACM. 22

Chaux, C., Pesquet, J.-C., and Pustelnik, N. (2009). Nested iterative algorithms for convex constrained image recovery problems. *SIAM Journal on Imaging Sciences*, 2:730–762. 62

Chen, X., Lin, Q., Kim, S., Carbonell, J., and Xing, E. P. (2011). Smoothing proximal gradient method for general structured sparse learning. In *UAI'11*, pages 105–114. 60

Chen, X., Pan, W., Kwok, J., and Carbonell, J. (2009). Accelerated gradient method for multi-task sparse learning problem. In *Ninth IEEE Intern. Conf. on Data Mining (ICDM '09)*., pages 746 –751. 61

Clark, W. A. and Farley, B. G. (1955). Generalization of pattern recognition in a self-organizing system. In *Proceedings of the March 1-3, 1955, western joint computer conference*, AFIPS '55 (Western), pages 86–91, New York, NY, USA. ACM. 1

Combettes, P., Dũng, D., and Vũ, B. (2010). Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, pages 1–32. 77

Combettes, P. and Wajs, V. (2005). Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200. 19, 60, 64

Cortes, C., Mohri, M., and Rostamizadeh, A. (2009). L2 regularization for learning kernels. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*. 51

Cortes, C., Mohri, M., and Rostamizadeh, A. (2010). Two-stage learning kernel algorithms. In *Proceedings of the 27 th International Conference on Machine Learning (ICML)*. 50, 51

Cortes, C. and Vapnik, V. N. (1995). Support vector networks. *Machine Learning*, 20:273–297. 5, 7

Crammer, K., Keshet, J., and Singer, Y. (2002). Kernel design using boosting. In *Proceedings of the Fifteenth Annual Conference on Neural Information Processing Systems (NIPS)*. 50

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:2001. 31

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press. 7, 31

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. (2001). On kernel-target alignment. In *Adv. in Neural Information Processing Systems (NIPS)*. 50

Drineas, P. and Mahoney, M. W. (2005). On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175. 40

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. (2008). Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:957–987. 58

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley and Sons. 55

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annels of Statistics*, 32:407–499. 51

Fadili, J. and Peyré, G. (2011). Total variation projection with first order schemes. *Image Processing, IEEE Transactions on*, 20(3):657–669. 60

Gosh, A., Kale, S., and McAfee, P. (2011). Who moderates the moderators?: crowdsourcing abuse detection in user-generated content. In *Proc. of the 12th ACM conference on Electronic commerce, EC 11*, pages 167–176. 22

Hazan, E., Koren, T., and Srebro, N. (2011). Beating sgd: Learning svms in sublinear time. In *25th Annual Conference on Neural Information Processing Systems (NIPS)*. 10

Herbster, M. and Lever, G. (2009). Predicting the labelling of a graph via minimum p-seminorm interpolation. In *Proc. of the 22nd Conference on Learning Theory*. 76

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: applications to nonorthogonal problems. *Technometrics*, 12(1):69–82. 40

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning*, pages 408–415. 43, 57

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. (2011). Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Reasearch*, 12:2297–2334. 61

Joachims, T. (1999). *Advances in Kernel Methods - Support Vector Learning*, chapter Making Large-Scale SVM Learning Practical. MIT Press. 9

Joachims, T. (2006). Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 9

Kloft, M., Brefeld, U., Sonnenburg, S., and Zien, A. (2010). Non-sparse regularization and efficient training with multiple kernels. Technical report, University of California, Berkeley. 51

Kowalski, M., Weiss, P., Gramfort, A., and Anthoine, S. (2011). Accelerating ista with an active set strategy. In *OPT 2011: 4th International Workshop on Optimization for Machine Learning*. 57

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. In *Proc. of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492. University of California Press. 70

Kumar, S., Mohri, M., and Talwalkar, A. (2009). Ensemble nyström method. In *Advances in Neural Information Processing Systems*. 38, 40, 51, 58

Lee, Y., Lin, Y., and Wahba, G. (2004). Multicategory support vector machines. *J. of the American Statistical Association*, 99:67–81. 32

Lin, Z., Ganesh, A., Wright, J., Wu, L., Chen, M., and Ma, Y. (2009). Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. 61

Loris, I. and Verhoeven, C. (2011). On a generalization of the iterative soft-thresholding algorithm for the case of non-separable penalty. *Inverse Problems*, 27:125007. 62

Mahoney, M. (2012). Approximate computation and implicit regularization for very large-scale data analysis. In *Proceedings of the 2012 ACM Symposium on Principles of Database Systems (PODS)*. 84

Maji, S. and Malik, J. (2009). Fast and accurate digit classification. Technical report, EECS Department, UC Berkeley. 52, 55

Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41:3397–3415. 58

McDiarmid, C. (1989). On the method of bounded differences. In *Surveys in Combinatorics*, pages 148–188. 28

Mosci, S., Rosasco, L., Santoro, M., Verri, A., and Villa, S. (2010). Solving structured sparsity regularization with proximal methods. In *Machine Learning and Knowledge Discovery in Databases*, volume 6322 of *Lecture Notes in Computer Science*, pages 418–433. Springer. 60, 61

Nemirovsky, A. and Yudin, D. (1983). *Problem complexity and method efficiency in optimization.* Wiley-Interscience Series in Discrete Mathematics. Johon Wiley & Sons, New York. 60

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization - A Basic Course.* Kluwer Academic Publishers. 17, 19, 45

Nesterov, Y. (2007). Gradient methods for minimizing composite objective function. Technical report, CORE Discussion Papers. 60, 63

Nesterov, Y. (2010). Efficiency of coordinate descent methods on huge-scale optimization problems. Core discussion papers, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain. 38, 43, 44

Nocedal, J. and Wright, S. (1999). *Numerical Optimization.* Springer-Verlag. 15

Novikoff, A. B. (1962). On convergence proofs on perceptrons. *Symposium on the Mathematical Theory of Automata*, 12:615–622. 6

Orecchia, L. and Mahoney, M. (2011). Implementing regularization implicitly via approximate eigenvector computation. In *Proceedings of the 28th International Conference on Machine Learning (ICML).* 84

Peel, T., Emiya, V., Ralaivola, L., and Anthoine, S. (2012). Matching pursuit with stochastic selection. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO).* 58

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research. 9

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838–855. 72

Rakotomamonjy, A., Bach, F. R., Canu, S., and Grandvalet, Y. (2008). Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521. 41, 42, 51, 58

Recht, B. (2011). A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12:3413–3430. 22

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408. 6

Rudelson, M. and Vershynin, R. (2007). Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54(4). 22

Rudin, L., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268. 75

Schmidt, M., Le Roux, N., and Bach, F. (2011). Convergence rates of inexact proximal-gradient methods for convex optimization. In *Adv. in Neural Information Processing Systems (NIPS)*. 61, 62, 63, 64, 65, 68, 76, 80

Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Müller, K.-R., Rätsch, G., and Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017. 40

Shalev-Schwartz, S., Shamir, O., and Tromer, E. (2012). Using more data to speed-up training time. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 85

Shalev-Schwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning (ICML)*. 10

Shalev-Schwartz, S. and Tewari, A. (2009). Stochastic methods for l_1 regularized loss minimization. In *Proceedings of the 26th International Conference on Machine Learning*. 43

Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *International Conference on Machine Learning*, pages 911–918. 52, 54

Suykens, J. A. K., Gestel, T. V., Brabanter, J. D., Moor, B. D., and Vandewalle, J. (2002). *Least Squares Support Vector Machines*, chapter 6, pages 173–182. World Scientific. 38

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistics Society Series B.*, 58:267–288. 18, 51

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of Ill-Posed Problems*. Winston. 31

Tropp, J. A. (2011). User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*. 22, 28, 29

Tseng, P. (2008). On accelerated proximal gradient methods for convex-concave optimization. Submitted to SIAM Journals on Optimization. 63

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142. 84

Vapnik, V. N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag. 2, 27

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer Verlag. 1

Vasin, V. V. (1970). Relationship of several variational methods for the approximate solution of ill-posed problems. *Mathematical Notes*, 7(3):161–166. 42

Villa, S., Salzo, S., Baldassarre, L., and Verri, A. (2011). Accelerated and inexact forward-backward algorithms. Technical report, Optimization Online. 61, 62, 63, 76

Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Technical report, Royal Holloway, University of London. 33

Williams, C. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press. 40, 52, 55

Woodbury, M. A. (1950). Inverting modified matrices. Technical report, Statistical Research Group, Princeton University. 91

Woodsend, K. (2009). *Using Interior Point Methods for Large-scale Support Vector Machine training*. PhD thesis, University of Edinburgh. 9

Zhang, K., Lan, L., Liu, J., Rauber, A., and Moerchen, F. (2012). Inductive kernel low-rank decomposition with priors. In *Proceedings of the 29 th International Conference on Machine Learning*. 51