Habilitation à diriger des recherches
Université Grenoble-Alpes

Spécialité : Mathématiques Appliquées et Informatique

Julien Mairal

Large-Scale Machine Learning and Applications

Soutenue le 4 octobre 2017

Rapporteurs :

Monsieur Léon Bottou	Professeur, New York University & Chercheur, Facebook
Monsieur Yves Grandvalet	Directeur de recherche, CNRS
Monsieur Klaus-Robert Müller	Professeur, Technische Universität Berlin

Membres du jury :

Monsieur Mário Figueiredo	Professeur, Instituto Superior Técnico, Lisbonne
Monsieur Anatoli Judistky	Professeur, Université Grenoble-Alpes
Monsieur Florent Perronnin	Chercheur, Naver Labs
Madame Cordelia Schmid	Directeur de recherche, Inria

Summary

This thesis presents my main research activities in statistical machine learning after my PhD, starting from my post-doc at UC Berkeley to my present research position at Inria Grenoble. The first chapter introduces the context and a summary of my scientific contributions and emphasizes the importance of pluri-disciplinary research. For instance, mathematical optimization has become central in machine learning and the interplay between signal processing, statistics, bioinformatics, and computer vision is stronger than ever. With many scientific and industrial fields producing massive amounts of data, the impact of machine learning is potentially huge and diverse. However, dealing with massive data raises also many challenges. In this context, the manuscript presents different contributions, which are organized in three main topics.

Chapter 2 is devoted to large-scale optimization in machine learning with a focus on algorithmic methods. We start with majorization-minimization algorithms for structured problems, including block-coordinate, incremental, and stochastic variants. These algorithms are analyzed in terms of convergence rates for convex problems and in terms of convergence to stationary points for non-convex ones. We also introduce fast schemes for minimizing large sums of convex functions and principles to accelerate gradient-based approaches, based on Nesterov's acceleration and on Quasi-Newton approaches.

Chapter 3 presents the paradigm of deep kernel machine, which is an alliance between kernel methods and multilayer neural networks. In the context of visual recognition, we introduce a new invariant image model called convolutional kernel networks, which is a new type of convolutional neural network with a reproducing kernel interpretation. The network comes with simple and effective principles to do unsupervised learning, and is compatible with supervised learning via backpropagation rules.

Chapter 4 is devoted to sparse estimation—that is, the automatic selection of model variables for explaining observed data; in particular, this chapter presents the result of pluri-disciplinary collaborations in bioinformatics and neuroscience where the sparsity principle is a key to build intepretable predictive models.

Finally, the last chapter concludes the manuscript and suggests future perspectives.

Résumé

Ce mémoire présente mes activités de recherche en apprentissage statistique après ma thèse de doctorat, dans une période allant de mon post-doctorat à UC Berkeley jusqu'à mon activité actuelle de chercheur chez Inria. Le premier chapitre fournit un contexte scientifique dans lequel s'inscrivent mes travaux et un résumé de mes contributions, en mettant l'accent sur l'importance de la recherche pluri-disciplinaire. L'optimisation mathématique est ainsi devenue un outil central en apprentissage statistique et les interactions avec les communautés de vision artificielle, traitement du signal et bio-informatique n'ont jamais été aussi fortes. De nombreux domaines scientifiques et industriels produisent des données massives, mais les traiter efficacement nécessite de lever de nombreux verrous scientifiques. Dans ce contexte, ce mémoire présente différentes contributions, qui sont organisées en trois thématiques.

Le chapitre 2 est dédié à l'optimisation à large échelle en apprentissage statistique. Dans un premier lieu, nous étudions plusieurs variantes d'algorithmes de majoration/minimisation pour des problèmes structurés, telles que des variantes par bloc de variables, incrémentales, et stochastiques. Chaque algorithme est analysé en terme de taux de convergence lorsque le problème est convexe, et nous montrons la convergence de ceux-ci vers des points stationnaires dans le cas contraire. Des méthodes de minimisation rapides pour traiter le cas de sommes finies de fonctions sont aussi introduites, ainsi que des algorithmes d'accélération pour les techniques d'optimisation de premier ordre.

Le chapitre 3 présente le paradigme des méthodes à noyaux profonds, que l'on peut interpréter comme un mariage entre les méthodes à noyaux classiques et les techniques d'apprentissage profond. Dans le contexte de la reconnaissance visuelle, ce chapitre introduit un nouveau modèle d'image invariant appelé réseau convolutionnel à noyaux, qui est un nouveau type de réseau de neurones convolutionnel avec une interprétation en termes de noyaux reproduisants. Le réseau peut être appris simplement sans supervision grâce à des techniques classiques d'approximation de noyaux, mais est aussi compatible avec l'apprentissage supervisé grâce à des règles de backpropagation.

Le chapitre 4 est dédié à l'estimation parcimonieuse, c'est à dire, à la séléction automatique de variables permettant d'expliquer des données observées. En particulier, ce chapitre décrit des collaborations pluri-disciplinaires en bioinformatique et neuroscience, où le principe de parcimonie est crucial pour obtenir des modèles prédictifs interprétables.

Enfin, le dernier chapitre conclut ce mémoire et présente des perspectives futures.

Contents

Chapter 1

Introduction

Machine learning is a relatively recent scientific field at the crossing of statistics, computer science, and applied mathematics. It may be seen as the science of data analysis and potentially impacts every domain where data has become a potential source of knowledge or economic activity. Machine learning is thus pluri-disciplinary by nature, and its importance is growing. For instance, it may be used in bioinformatics for personalized medicine, in neuroscience to better understand the brain activity, in physics for detecting the Higgs boson, in image processing and computer vision to manipulate content acquired by smartphone cameras, or even to design autonomous cars. Its field of application is therefore highly diverse, and it is thus tempting to replace in the context of machine learning the following quote from John Tukey, who once said that "the best thing about being a statistician is that you get to play in everyone's backyard".

In light of John Tukey's remark, this chapter presents the recent scientific context of my research directions and a summary of my contributions organized in three different topics: large-scale machine learning, deep kernel machines for visual recognition, and pluri-disciplinary collaborations in bioinformatics and neuroscience.

1.1 Scientific context

Recently, machine learning has seen significant changes due to the increased size, diversity, and complexity of datasets, along with the availability of more powerful computational devices to process them. With such changes, many paradigms are shifted and new scientific challenges arise, which demand new research.

Big data is shifting classical paradigms. We have entered an era of massive data acquisition, leading to the revival of an old scientific utopia: it should be possible to better understand the world by converting data into knowledge. It is also leading to a new economic paradigm, where data is a valuable asset and a source of activity. Therefore, developing scalable technology to make sense of massive data has become a strategic issue. Machine learning has already started to adapt to these changes. We have indeed devoted a significant effort to scaling up simple techniques such as empirical risk minimization

with linear models (Schmidt et al., 2016; Shalev-Shwartz and Zhang, 2013, 2016; Xiao and Zhang, 2014; Defazio et al., 2014b; Mairal, 2015; Lin et al., 2015), but large parts that have been highly effective for medium-sized problems have been left aside.

In particular, kernel methods (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) are often considered unsuitable to large-scale settings despite recent advances using random sampling (Rahimi and Recht, 2007; Le et al., 2013) or other approximations for shift-invariant kernels (Vedaldi and Zisserman, 2012). Traditional kernel approaches require indeed computing a Gram matrix of size $n \times n$, where n is the number of training samples. When n is large, these techniques become impractical from memory and computational point of views. To avoid this quadratic complexity, various schemes have been proposed for approximating a non-linear kernel with a linear one (Smola and Schölkopf, 2000; Fine and Scheinberg, 2001; Williams and Seeger, 2001; Bach and Jordan, 2005; Rahimi and Recht, 2007; Vedaldi and Zisserman, 2012; Le et al., 2013); yet, improving the scalability of kernel methods remain an important open problem today.

Large-scale optimization algorithms are crucial in the big data context. To process large amounts of data, one source of speed-up in machine learning is coming from stochastic optimization (Bousquet and Bottou, 2008; Shalev-Shwartz et al., 2011; Schmidt et al., 2016; Defazio et al., 2014b; Shalev-Shwartz and Zhang, 2016; Xiao and Zhang, 2014; Mairal, 2013a, 2015). Specifically, these techniques have shown to be well adapted *to the minimization of a regularized empirical risk*—that is, the minimization of a large sum of n (possibly convex) functions. Typically, classical batch optimization algorithms, such as gradient descent, load the entire dataset at each iteration, which has recently become cumbersome for problems where n is large, *e.g.*, when n is greater than a million. Stochastic and incremental algorithms, on the other hand, exploit the specific structure of the cost function and observe a single training point at a time or a mini-batch. Their cost per iteration is thus independent of n, but more importantly, they achieve a significant speed-up over batch algorithms in the big data regime.

Interestingly, stochastic optimization has been present since the beginning of machine learning. For instance, the roots of the stochastic gradient descent method lie in the algorithm of Robbins and Monro (1951) from the 50's, and was implemented in the first machine learning devices like the Adaline at the beginning of the 60's (Widrow and Hoff, 1960) for solving least-square problems. Since then, it has been heavily used to train neural networks (see LeCun et al., 1998). Despite these old origins, stochastic optimization remains a very active research topic in machine learning, and was recently the subject of important improvements. These include faster algorithms than the stochastic gradient descent method when the training set is large but finite (Schmidt et al., 2016; Shalev-Shwartz and Zhang, 2013, 2016; Xiao and Zhang, 2014; Mairal, 2015; Lin et al., 2015), or fast dedicated algorithms for specific problems such as matrix factorization with sparsity-inducing regularization (Mairal et al., 2010a; Mairal, 2013a; Mensch et al., 2016).

Deep neural networks are highly successful nowadays. This change is closely related to the advent of big data. On the one hand, deep neural networks involve a huge number of parameters and are rich enough to represent well complex objects such as

natural images or text corpora. On the other hand, they are prone to overfitting (fitting too closely to training data without being able to generalize to new unseen data); to work well on difficult tasks, they require a large amount of labeled data that has been available only recently. Other cues may explain their success: the deep learning community has made significant engineering efforts, making it possible to learn complicated model on GPUs in a matter of hours (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015), and it has accumulated enough empirical experience to find good hyper-parameters for its networks. [1] Therefore, there is a growing excitement about deep learning in machine learning, and it is clear that deep learning has revolutionized the practice of the computer vision community. Yet, these approaches still suffer from several limitations.

We are still lacking principled methodology and theory for deep learning. Deep models are sometimes seen as engineering black boxes that have been designed with a lot of insight collected since they were introduced. For example, the first ideas that led to convolutional neural networks were originally introduced at the end of the 70's (Fukushima, 1979), and were significantly developed in the 90's (LeCun et al., 1998) and recently with more complex (deeper) architectures (Krizhevsky et al., 2012; Ciresan et al., 2012; Simonyan and Zisserman, 2015). Yet, we can enumerate several ever-lasting important open problems regarding this class of models.

(i) *How to regularize them?* The main popular approaches to regularize deep neural networks are based on early-stopping the optimization procedure, model averaging (Wan et al., 2013; Srivastava et al., 2014), and data augmentation (Simard et al., 1992; Ciresan et al., 2012). However, model averaging techniques such as Dropout are poorly understood from a theoretical point of view (Wager et al., 2014) and seem to provide mitigated benefits (Wan et al., 2013). On the other hand, data augmentation is a powerful approach that consists of increasing the size of a training set by artificially generating virtual examples, thus reducing the importance of regularization. Unfortunately, generating virtual examples for data augmentation requires domain-knowledge, which is not always available, and also requires selecting manually several hyper-parameters.

(ii) *How to learn with less supervision?* Deep models have been highly successful for prediction tasks given labeled training data. However, unsupervised deep learning approaches such as autoencoders (see, *e.g.*, Hinton et al., 2006) or restricted Boltzmann machines (Smolensky, 1986) have had mixed success in the sense that other unsupervised techniques such as sparse coding (Olshausen and Field, 1996; Aharon et al., 2006; Mairal et al., 2010a, 2014a), independent component analysis (Bell and Sejnowski, 1997; Hyvärinen et al., 2009), or non-negative matrix factorization (Paatero and Tapper, 1994), are still the state of the art in various scientific fields. The question of learning with less supervision is related to the question of regularization, but it is also related to the difficulty of learning deep models with no supervision at all.

(iii) *What are their functional spaces and what are their properties?* Understanding the geometry of functional spaces corresponding to deep models is a fundamental question

1. For instance, the multilayer network used by Krizhevsky et al. (2012) for winning the image classification challenge ImageNet in 2012 has 60 million parameters and about 50 hyperparameters that are manually chosen.

since it can solve the issue of regularization, by providing ways to control the variations of prediction functions in a principled manner. The scattering transform (Bruna and Mallat, 2013) is a recent attempt to fill in this gap by combining convolutional neural networks (LeCun et al., 1998) with wavelets. The theory provides elegant insight about invariant properties of signal representations produced by deep networks. Nevertheless, scattering networks do not involve "learning" in the classical sense since the filters of the networks are pre-defined, and they differ significantly from the most used ones.

(iv) *How to optimize them?* Deep neural networks are often formulated as the minimization of a non-convex cost function. Because of non-convexity, the global optimum cannot be found in general, making the approach difficult to analyze. Only very recently, a few theoretical results have been obtained suggesting that very large networks may be tractable under some assumptions (Choromanska et al., 2015; Livni et al., 2014). Even though these fresh results are significant from an optimization perspective, the gap between theory and practice is still present, and the theory does not explain yet how to design successful deep architectures.

Data is more and more complex and structured. With the development of new acquisition devices and storage capacity, data complexity has been continuously increasing. To illustrate this trend, we may draw an example from neuroscience, where it is classical to display a visual stimulus to a subject while recording and analyzing its cortical activity. Such experiments were first carried out with random noise (unstructured data), then successively with artificial stimuli (gratings, bars), sequences of natural images, and finally color movies (Nishimoto et al., 2011). On the data acquisition side, electrophysiological recordings of individual cells have been used before arrays of electrodes and fMRI of the full cortex. The trend is thus to go to more complex data, which have also more structure (time in movies, spatial location of voxels in fMRI data).

Unfortunately, there is a lack of scalable technology that is adaptive to various data structures. In particular, the room for improvement is substantial for deep learning approaches, which are often dedicated and manually tuned to specific applications. For instance, convolutional neural networks are designed for images and there is no consensus yet on how to even apply them to videos (Karpathy et al., 2014; Simonyan and Zisserman, 2014; Tran et al., 2015; Wang et al., 2015a). Kernels, on the other hand, have a long tradition of working with structured data (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004). For instance, kernels have been defined for protein sequences in computational biology (Leslie et al., 2002; Cuturi and Vert, 2005), trees and graphs (Borgwardt and Kriegel, 2005), or time series for audio data (Cuturi et al., 2007).

The need for scalable deep kernel methods. Kernel methods are appealing since they seem to address most of the open questions described above concerning deep learning. They provide sound principles to regularize the models, lead to well-studied functional spaces (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004), and produce often convex optimization problems that can be solved exactly.

Making an alliance between kernels and deep learning is however a difficult problem, and only a few links between kernels and deep networks have been drawn so far.

Early relations appear for instance in a multilayer variant of kernel principal component analysis (Schölkopf et al., 1998) or in the PhD thesis of Neal (1994), who has shown that particular neural networks with infinite number of hidden units were equivalent to Gaussian processes. This type of infinitely large networks was then revisited later numerous times by Le Roux and Bengio (2007); Cho and Saul (2010); Bach (2017); Hazan and Jaakkola (2015), showing that—among other properties—they could approximate positive definite kernels defined as an integral in various cases. In the last few years, other works have also used kernels as purely theoretical tools to gain some insight about classical multilayer neural networks (Montavon et al., 2011; Anselmi et al., 2015; Daniely et al., 2016). Remarkably, Smale et al. (2010) also proposes a conceptually interesting multilayer construction involving kernels, but, in fine, the corresponding learning system is not a deep neural network in the classical sense, meaning they do not perform sequences of linear operations interleaved with nonlinearities; it also has not shown yet competitive performance on classification benchmarks. In chapter 3, we will show a new construction with reasonable performance for image classification, which we call convolutional kernel networks (Mairal et al., 2014c; Mairal, 2016).

The need of scalable sparse models in data science. The sparsity principle is a fundamental heuristic of science, consisting of representing, or explaining, observed data by as few variables as possible. It was introduced in a famous epistemological paper by Wrinch and Jeffreys (1921), who discussed in modern terms the preference of statistical models with few parameters as a natural heuristic. Since then, this principle has emerged many times in various scientific communities, independently at different times.

For instance, a significant body of work about sparsity was produced in statistics in the 60's and 70's. Mallows (1964, 1966) introduced the C_p-statistics for model selection, later generalized by Akaike (1973) with the AIC criterion and then by Schwarz (1978) with the Bayesian information criterion (BIC). Around the same times, greedy algorithms for least square problems were developed in the context of best subset selection (Efroymson, 1960; Hocking, 1976). Sparsity is a also a key principle from Markowitz (1952)'s portfolio selection in finance, and from the theory of wavelets in signal processing, which was developed in large parts in the 80's and 90's (see Daubechies, 1988; Mallat, 1989; Simoncelli et al., 1992). The ℓ_1-norm was introduced as a sparsity-inducing regularization in geophysics (Claerbout and Muir, 1973) before being popularized in statistics (Tibshirani, 1996) and signal processing (Chen et al., 1999).

From an application point of view, the sparsity principle has had a huge impact in various fields, notably in image processing (Elad and Aharon, 2006) and computer vision (see Mairal et al., 2014b, for a review). Today, it remains an important principle for interpreting a predictive model learned on experimental data. This is for instance the case in neuroscience (Varoquaux et al., 2011) or in genetics (Carvalho et al., 2008). With such experimental fields producing more and more high-dimensional data, where prediction is equally important as interpretation, developing scalable sparse estimation tools is therefore still an acute problem today.

1.2 Research activity

The scientific context described above raises many challenges, including difficult ones that are still open today. Of course, we do not address all of them in this thesis, but we present a few contributions about large-scale optimization, deep kernel methods, and sparse estimation involving pluri-disciplinary collaborations. In the rest of this section, I will briefly introduce these contributions and the research context in which they were made, but also provide a short description of my earlier PhD work.

1.2.1 Research conducted before and during the PhD

I started conducting research before my PhD in 2006, during a master internship at the Electrical Engineering and Computer Science department (EECS) of the University of Minnesota, under the supervision of Guillermo Sapiro. At that time, I also had the chance to work remotely with Michael Elad from the Technion. During this fruitful collaboration, we developed several image processing techniques (Mairal et al., 2008c,e) based on dictionary learning, a matrix factorization formulation with sparse factors introduced by Olshausen and Field (1996), which we used for representing image patches.

After this first contact with research, I started my PhD in 2007 under the supervision of Jean Ponce and Francis Bach. Following the recent success obtained during my master thesis for image reconstruction tasks, we developed discriminative local appearance models for images and extended for that purpose the original unsupervised dictionary learning framework (Mairal et al., 2008a,d). Ultimately, this research direction converged to supervised learning models (Mairal et al., 2008b, 2012) that combine ideas from neural networks (backpropagation) with dictionary learning.

In the second year of my PhD, I decided to work again on image processing and developed a restoration method (Mairal et al., 2009b) that uses both sparse representations and image self-similarities—which is the main principle used by the non-local means approach (Buades et al., 2005); several years after it was published, this method remains a top-competitive one in denoising and demosaicking benchmarks. Then, I shifted my research focus towards optimization for machine learning and signal processing. Inspired by a talk by Léon Bottou at the NIPS conference about stochastic optimization in 2008, I decided to attack the problem of dictionary learning from a large-scale perspective, where input signals come from a very large, or possibly infinite, data stream. The resulting algorithm (Mairal et al., 2009a, 2010a) turned out to be highly scalable and generic enough to be applied to many matrix factorization problems such as non-negative matrix factorization (Paatero and Tapper, 1994) or sparse principal component analysis.

Finally, during the third and last year of my PhD, I had the chance to start a fruitful collaboration with Rodolphe Jenatton, another student from Francis Bach, who was working on structured sparsity (Jenatton et al., 2011a), which consists of extending classical sparsity-inducing regularization functions such as the ℓ_1-norm to take into account a given structure. Together, we developed fast algorithms to compute the proximal operator of hierarchical sparsity-inducing norms using dynamic programming (Jenatton et al.,

2010, 2011b), and then network-flow optimization techniques for more general structured norms (Mairal et al., 2010b, 2011). This was the last contribution of my PhD.

The common line in this PhD work is the use of sparse representations of signals or data. Since then, I explored other research directions that are unrelated to the sparsity principle, but I kept an applied perspective on it. To draw a conclusion to this line of research, I invested recently a significant amount of time writing a 200-pages monograph (Mairal et al., 2014a). Thus, only few contributions presented in this HdR thesis are related to the sparsity principle, as shown in the next section.

1.2.2 Contributions of the HdR

This manuscript is organized into three parts presenting independent contributions about large-scale optimization, deep kernel machines, and sparse estimation in a pluridisciplinary applied context.

Contributions in large-scale optimization. Chapter 2 is devoted to large-scale optimization; in particular, we focus on the minimization of a large sum of functions. This is a key setting in (Frequentist) machine learning formulations, where the goal is to find high-dimensional parameters that best fit a model, while taking into account a possibly non-smooth regularization function. The optimization problem may be non-convex, as in matrix factorization, but it may also be a convex one such as support vector machines or logistic regression. I got interested in optimization relatively early during my PhD from a practical and methodological point of view for solving sparse estimation problems. Later in 2012, I started studying optimization from a more general and theoretical perspective, essentially by reading Nesterov's book (Nesterov, 2004).

My first object of research was the principle of majorization-minimization, which is a simple (naive) mechanism consisting of driving the objective function of a given problem downhill by iteratively minimizing tight upper bounds of the objective. I introduced several variants for structured problems, where either the model parameters can be decomposed into blocks of variables, or when the objective function itself is separable (Mairal, 2013b, 2015). For each variant, I studied convergence to stationary points for non-convex problems, and their convergence rates for convex ones. I also extended this work to stochastic optimization problems (Mairal, 2013a), where the objective is not a finite sum anymore, but an expectation. This stochastic majorization-minimization framework was then further developed for huge-scale matrix factorization by Arthur Mensch, joint PhD student with Gaël Varoquaux and Bertrand Thirion (see Mensch et al., 2016), who needed to factorize large matrices of several terabytes on a single workstation.

In (Mairal, 2015), I also introduced an algorithm for minimizing finite sums of strongly-convex functions. The algorithm is also based on "surrogate optimization"—that is, the minimization at each iteration of a model of the objective—but the surrogates are quadratic lower bounds (based on strong convexity) instead of upper bounds as in traditional majorization-minimization schemes. The algorithm, called MISO, converges much faster (both in theory and in practice) than the majorization-minimization variants mentioned above and falls into the class of stochastic gradient descent algorithms with vari-

ance reduction. It can be interpreted as a variant of the stochastic dual coordinate ascent (SDCA) technique of Shalev-Shwartz and Zhang (2013), without the duality point of view (see also Shalev-Shwartz, 2016). It offers also similar guarantees as recent alternatives such as SAG (Schmidt et al., 2016) or SAGA (Defazio et al., 2014b). Originally designed for smooth problems, we extended MISO to the proximal setting with my PhD student Hongzhou Lin, who is co-advised by Zaid Harchaoui (see Lin et al., 2015). In the context of minimizing large sums of functions, the method achieves near-optimal convergence rates for first-order oracles (Agarwal and Bottou, 2015; Lan and Zhou, 2017).

Finally, the last part of the chapter presents the work conducted by my student Hongzhou Lin about accelerating gradient-based methods in a generic fashion, either by generalizing a principle introduced by Nesterov in the 80's (see Lin et al., 2015), or by using limited-memory Quasi-Newton principles (Lin et al., 2017).

Towards deep kernel machines. Chapter 3 presents several contributions about deep kernel machines. I originally got interested in this topic in 2011, while I was a post-doc at UC Berkeley. At that time, Stéphane Mallat was presenting at various occasions an impressive work about the scattering transform (Bruna and Mallat, 2013), which inspired me to develop image representations with similar invariance properties based on kernels. I quickly realized that a large part of what I had in mind was already present in the hierarchical kernel descriptors of Bo et al. (2011). After leaving the topic aside for a while, the project restarted in 2013, leading to the convolutional kernel network model (Mairal et al., 2014c): local image neighborhoods are represented as points in a reproducing kernel Hilbert space via the kernel trick. Then, hierarchical representations are built via kernel compositions, producing a sequence of "feature maps" akin to convolutional neural networks, but of infinite dimension. To make the image model computationally tractable, convolutional kernel networks provide an approximation scheme that can be interpreted as a particular type of convolutional neural network learned without supervision.

Then, I simplified and extended the model to deal with supervised data in (Mairal, 2016). To perform end-to-end learning given labeled data, I use a simple but effective principle consisting of learning discriminative subspaces in RKHSs, where we project data. This idea is implemented in the context of convolutional kernel networks, where linear subspaces, one per layer, are jointly optimized by minimizing a supervised loss function. The formulation turns out to be a new type of convolutional neural network with a geometric interpretation. This approach was evaluated successfully on deep learning classification benchmarks and on image super-resolution.

In the meantime, the original unsupervised convolutional kernel network model (Mairal et al., 2014c) was adapted for the image retrieval task by Mattis Paulin, a PhD student I started working with in 2015 (see Paulin et al., 2015, 2016). When we published this work, a large number of research groups were trying to adapt convolutional neural networks to the task of image retrieval, obtaining good, but mitigated results compared to other classical approaches. Our model managed to match their performance with no supervision, using only a fraction of their training time. Only very recently, classical convolutional neural networks managed to achieve outstanding results for this task (Gordo et al., 2016; Radenović et al., 2016).

Sparse estimation and pluri-disciplinary research. After my PhD, I had the chance to be mentored by Bin Yu at UC Berkeley during my post-doc. One important advice she gave me was to spend a significant amount of time on topics that are significantly different from my current background, and to enjoy that time. Yet, I continued working a bit on the sparsity topic for a while, resulting in one theoretical (Mairal and Yu, 2012) contribution and in a methodological one (Mairal and Yu, 2013). Then, I had several interesting opportunities to start pluri-disciplinary collaborations where the sparsity principle is important, either for computational or for interpretability reasons. This was the case in bioinformatics with Laurent Jacob and Jean-Philippe Vert (Bernard et al., 2014, 2015), in natural language processing with Guillaume Bouchard and Cédric Archambeau from Xerox (XRCE) (see Nelakanti et al., 2013), and also in neuroscience on a project initiated by Bin Yu and Jack Gallant (Mairal et al., 2013).

Chapter 4 presents some of these contributions. The first theoretical one from (Mairal and Yu, 2012) is about the regularization path of the Lasso—that is, the set of all possible solutions of the Lasso when varying the regularization parameter. This path can be shown to be piecewise linear, making it possible to "follow" and explicitly compute the entire path. This strategy known as "homotopy" (Osborne et al., 2000), or as a variant of the LARS algorithm (Efron et al., 2004) is in fact probably the most popular one for solving the Lasso. Here, we prove that its worst case complexity is exponential in the number of variables and we construct an explicit pathological example that suffers from this complexity. The story is then reminiscent of the simplex algorithm, an effective approach in practice for solving linear program despite its exponential worst-case complexity.

In (Mairal and Yu, 2013), we introduce structured sparsity-inducing norms, where each variable can be mapped to the nodes of a directed acyclic graph, and propose efficient optimization techniques to deal with such norms in the context of empirical risk minimization. At that time, the method was lacking convincing realistic applications. Shortly after publishing this paper, I discovered the problem of isoform discovery in RNA-Seq data by attending the NIPS workshop on computational biology. Interestingly, it seemed to me that the optimization tools we introduced could be applied there, since the formulation was involving a sparse optimization problem with an exponential number of variables, each one being mapped to a path of a directed acyclic graph. After discussing this matter with Jean-Philippe Vert and Laurent Jacob, Jean-Philippe proposed to start a collaboration with Elsa Bernard, one of his PhD students, leading to two publications (Bernard et al., 2014, 2015) and an open-source software package called FlipFlop.

Finally, the last chapter also presents a collaboration initiated by Bin Yu and Jack Gallant from UC Berkeley when I was a post-doctoral researcher. This work involves notably Yuval Benjamini, who was a PhD student of Bin Yu at that time and contributed as much as I did on this project. The main idea of this work is to use dictionary learning to build an interpretable predictive model of the neural activity in the V4 area of the mammalian visual cortex. In terms of prediction, the proposed model outperforms state-of-the-art ones based on Gabor features, while being easily interpretable.

Acknowledgements

I am grateful to Léon Bottou, Yves Grandvalet and Klaus-Robert Müller for accepting to review this thesis. Time has become one of the most expensive resource for today's researchers, and I greatly appreciate that they sacrificed part of their research time to participate to this evaluation. For the same reason, I would also like to thank Mario Figueiredo, Anatoli Judistky, Florent Perronnin, and Cordelia Schmid for accepting to be part of the jury.

Then, I would like to thank my past mentors, with whom I had the chance to interact. In chronological order, this includes Guillermo Sapiro and Michael Elad—when I was a master intern in Minnesota—,then my two PhD advisors Francis Bach and Jean Ponce, and later my post-doc superviser Bin Yu from UC Berkeley.

I would also like to thank my current and former colleagues from the Lear/Thoth team at Inria Grenoble, notably, in alphabetical order, Karteek Alahari, Matthijs Douze, Zaid Harchaoui, Gregory Rogez, Cordelia Schmid, and Jakob Verbeek.

I also had the chance to build fruitful pluri-disciplinary collaborations, with many researchers outside of my field. I shall mention Michael Blum, Laurent Jacob and Jean-Philippe Vert in bioinformatics; Bertrand Thirion and Gael Varoquaux in neuroimaging, and ex-XRCE colleagues Cédric Archambeau and Guillaume Bouchard in natural language processing. I thank in particular their patience for introducing me to their respective fields, and share their expertise.

Most importantly, I also had the chance to work with many students, and in fact a significant part of the work presented in this dissertation was conducted by them. The non-exhaustive list includes notably Elsa Bernard, Mattis Paulin, Thomas Dias Alves, Hongzhou Lin, Daan Wynen, Arthur Mensch, Alberto Bietti, Mikita Dvornik, and very recently Andrei Kulunchakov and Dexiong Chen.

On a more personal note, I am also very grateful to Marine to her support during all these years, and I hope that Maia will enjoy the HdR defense as well as she enjoyed her mother's PhD presentation.

Chapter 2

Large-Scale Optimization for Machine Learning

This chapter is based on the following publications and preprints.

J. Mairal. Optimization with first-order surrogate functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

H. Lin, J. Mairal, and Z. Harchaoui. Quickening: A generic Quasi-Newton algorithm for faster gradient-based optimization. *preprint arXiv:1610.00960*, 2017.

A. Bietti and J. Mairal. Stochastic Optimization with Variance Reduction for Infinite Datasets with Finite-Sum Structure. *preprint arXiv:1610.00970*, to appear at NIPS 2017.

Continuous and discrete optimization has been central to machine learning for a long time (see Bottou et al., 2016). In this chapter, we are interested in continuous problems with a particular "large-scale" structure that prevents us from using generic optimization toolboxes such as CVX (Grant and Boyd, 2014), Mosek [1], CPLEX [2], or simply vanilla first- or second-order gradient descent methods. In such a context, all of these tools suffer indeed either from too large complexity per iteration, or too slow convergence, or both, which has motivated the machine learning community to develop dedicated algorithms. Formally, we consider the general minimization problem

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}),$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a continuous function and $\mathcal{C}$ is a convex subset of a high-dimensional Euclidean space $\mathbb{R}^p$, *e.g.*, $p \geq 10\,000$. We will now present typical problem structures in machine learning that require a specific treatment.

Large sums of functions and expected costs. Many machine learning formulations can be written as an empirical risk minimization problem:

$$\min_{\mathbf{x}\in\mathcal{C}} \left\{ F(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right\}, \tag{2.1}$$

where the functions $f_i : \mathbb{R}^p \to \mathbb{R}$ are continuous and $\mathcal{C}$ is a convex subset of $\mathbb{R}^p$. When F is non-convex, exactly solving (2.1) is intractable in general, and when F is also non-smooth, simply finding a stationary point can be difficult. In the empirical risk minimization framework, the solution $\mathbf{x}$ represents model parameters and each function f_i measures the fit of $\mathbf{x}$ to an observed data point indexed by i. In this context, minimizing F amounts to finding parameters $\mathbf{x}$ that explain well observed data. In the most classical setting, the functions f_i measure the fit of a linear model and involve a squared norm regularization to prevent overfitting:

$$f_i(\mathbf{x}) := \ell(y_i, \mathbf{x}^\top \boldsymbol{\xi}_i) + \frac{\mu}{2}\|\mathbf{x}\|_2^2, \tag{2.2}$$

where $(y_i, \boldsymbol{\xi}_i)$ is an example-label pair randomly drawn from the unknown data distribution, ℓ is a convex loss function, *e.g.*, logistic, square, or hinge loss, and μ is a regularization parameter that provides strong convexity to the global objective. In many interesting cases, the situation is however more involved. For instance, the functions f_i may be non-convex and may also require solving optimization sub-problems:

$$f_i(\mathbf{x}) := \min_{\boldsymbol{\beta}\in\mathcal{C}'} \tilde{f}_i(\mathbf{x}, \boldsymbol{\beta}), \tag{2.3}$$

where $\boldsymbol{\beta}$ is a latent variable and $\mathcal{C}'$ a convex set of another Euclidean space where $\boldsymbol{\beta}$ lives. This is notably the case of the regularized matrix factorization problem, where one looks for a matrix $\mathbf{X}$ in $\mathbb{R}^{m\times k}$ that minimizes the cost

$$\min_{\mathbf{X}\in\mathcal{C}} \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{X}) \quad \text{where} \quad f_i(\mathbf{X}) := \min_{\boldsymbol{\beta}\in\mathbb{R}^k} \frac{1}{2}\|\boldsymbol{\xi}_i - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\Omega(\boldsymbol{\beta}).$$

1. `http://www.mosek.com`.
2. `https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`.

There, the vectors $\boldsymbol{\xi}_i$ in $\mathbb{R}^m$ represent the columns of an input matrix we wish to factorize, and $\mathcal{C}$ and Ω provide regularization to the factors $\mathbf{X}$ and $\boldsymbol{\beta}$, respectively. When Ω is a sparsity-inducing penalty such as the ℓ_1-norm, the problem is known as dictionary learning (Olshausen and Field, 1996; Mairal et al., 2014b); when instead one considers non-negativity constraints, we obtain the non-negative matrix factorization formulation (Paatero and Tapper, 1994). In both cases, the overall problem is non-convex.

Note that (2.1) assumes that the training set is finite. But, as pointed out for instance by Bousquet and Bottou (2008), one is usually not interested in the minimization of an empirical cost on a finite training set, but instead in minimizing the expected cost

$$\min_{\mathbf{x}\in\mathcal{C}}\{F(\mathbf{x}) := \mathbb{E}_{\boldsymbol{\xi}}[f(\mathbf{x},\boldsymbol{\xi})]\}, \tag{2.4}$$

where $\boldsymbol{\xi}$ is a vector representing a data point, which is drawn according to some unknown distribution from which we know how to sample, and f is a continuous loss function. As often done in the literature, we assume here that the expectations are always well defined and finite valued. Minimizing (2.4) is feasible in many cases, when one has access to an infinite stream of i.i.d. data examples $(\boldsymbol{\xi}_t)_{t\geq 1}$.

Optimization for minimizing expectations. In the last few years, stochastic optimization techniques to solve (2.4) or (2.1) when n is huge have become very popular in machine learning (see, *e.g.*, Bousquet and Bottou, 2008; Duchi and Singer, 2009; Xiao, 2010). The simplest of all is the stochastic gradient descent method (SGD) for smooth or convex problems of the form (2.4) with no constraint; this approach consists of drawing at iteration t a training point $\boldsymbol{\xi}_t$, computing the gradient $\mathbf{g}_t = \nabla_{\mathbf{x}} f(\mathbf{x}_{t-1}, \boldsymbol{\xi}_t)$ or a subgradient if the function is nonsmooth and convex, and performing the update

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \eta_t \mathbf{g}_t,$$

where η_t is a positive scalar often called "learning rate" in machine learning. The algorithm is also appropriate for minimizing the finite sum (2.1), which is a particular case of (2.4) when the data distribution is discrete. Note that when F is smooth, we have $\mathbb{E}[\mathbf{g}_t] = \nabla F(\mathbf{x}_{t-1})$ and the method performs gradient descent steps "on average".

The convergence of stochastic optimization algorithms in the convex case is well understood. Even though these approaches have sublinear convergence rates in such settings (Lan, 2012; Nemirovski et al., 2009), they typically have a cheap computational cost per iteration, enabling them to efficiently find an approximate solution. In a nutshell, combined with an averaging strategy of the iterates (see Nemirovski et al., 2009; Nesterov and Vial, 2008), their expected complexity for obtaining a solution with accuracy ε—that is, such that $\mathbb{E}[F(\mathbf{x}_t) - F^\star] \leq \varepsilon$, where $F^\star$ is the optimal value function—is of the order $O(1/\varepsilon)$ for strongly convex problems and $O(1/\varepsilon^2)$ for convex ones.[3] It is also known that these complexities are optimal (Nemirovski et al., 2009), up to a constant factor, for solving (2.4) for stochastic oracles—that is, for optimization algorithms that only access

3. Note that all convergence rates or complexities presented in this chapter introduction are non-asymptotic. For better readability, they are written with the $O(.)$ notation that hides constants of the problem.

a stochastic estimate $\mathbf{g}_t$ of the true gradient at iteration t. Note that to make it easy to translate convergence rates (the speed of convergence of the sequence $(F(\mathbf{x}_t) - F^\star)_{t\geq 0}$) into complexities, we provide correspondences in Table 2.1 at the end of this section.

For non-convex smooth problems, convergence to a stationary point is also guaranteed under a few assumptions (Bottou, 1998). In Section 2.1, we present a class of stochastic algorithms that also enjoy similar properties, but also with convergence results to stationary points for some non-smooth non-convex problems (Mairal, 2013a).

Optimization for large finite sums. Recently, randomized incremental algorithms have been proposed for minimizing finite sums of functions such as SAG (Schmidt et al., 2016), SAGA (Defazio et al., 2014b), SDCA (Shalev-Shwartz and Zhang, 2013), SVRG (Xiao and Zhang, 2014), or MISO/Finito (Mairal, 2015; Lin et al., 2015; Defazio et al., 2014a), which will be presented in Section 2.2. At the price of a higher memory cost than purely stochastic algorithms such as SGD, these incremental methods enjoy faster convergence rates, while also having a cheap per-iteration computational cost. For instance, the method SAG for smooth unconstrained problems is an extension of the incremental gradient method of Blatt et al. (2007); it uses the following update:

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \frac{\alpha}{nL_f}\sum_{i=1}^n \mathbf{z}_t^i \quad \text{with} \quad \mathbf{z}_t^i = \begin{cases} \nabla f_i(\mathbf{x}_{t-1}) & \text{if } \ i = i_t \\ \mathbf{z}_{t-1}^i & \text{otherwise} \end{cases},$$

after randomly selecting an index i_t at iteration t. The constant L_f is an upper bound on the Lipschitz constant of the gradients ∇f_i, assuming them to be Lipschitz continuous. Storing the auxiliary vectors $\mathbf{z}_t^i$ is potentially costly in general, but in the context of linear models (2.2), one may notice that SAG only requires storing one scalar per data point since $\nabla f_{i_t}(\mathbf{x}_{t-1}) = \ell'(y_{i_t}, \mathbf{x}_{t-1}^\top \boldsymbol{\xi}_{i_t})\boldsymbol{\xi}_{i_t}$; thus, the memory overhead reduces to $O(n)$. When the objective is μ-strongly convex, the method is guaranteed to provide a solution optimal up to ε in expectation after evaluating

$$O\left(\max\left(n, \frac{L_f}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$$

gradients of the functions f_i. In comparison, the gradient descent method applied to F achieves the same guarantee (but in a deterministic sense) in $O(n(L_F/\mu)\log(1/\varepsilon))$ gradients evaluations, and the complexity becomes $O\big(n\sqrt{L_F/\mu}\log(1/\varepsilon)\big)$ for the accelerated gradient descent method of Nesterov (1983). There, L_F is the Lipschitz constant of ∇F, which is smaller than L_f (individual Lipschitz constant of the gradients ∇f_i). Therefore, unless there is a big mismatch between L_F and L_f, SAG significantly outperforms optimal gradient descent methods when n is large enough. While doing better than an "optimal" method may seem surprising, we may remark that SAG is tailored to the finite-sum setting, whereas the optimal rates of Nesterov (2004) apply to the more general case; indeed, the rate of convergence of SAG is in fact slower than the accelerated gradient descent method; by exploiting cleverly the finite-sum structure, only *its cost per iteration* is significantly lower, resulting in lower overall complexity. At first sight, SAG seems

also to outperform the optimal stochastic complexity $O(1/\varepsilon)$ for μ-strongly convex objectives (Nemirovski et al., 2009). In fact, SAG addresses a deterministic objective (2.1) and it thus not limited by stochastic lower bounds on optimization algorithms for solving (2.4). On the other hand, it is affected by specific optimal complexity bounds for finite-sum objectives (Agarwal and Bottou, 2015; Lan and Zhou, 2017), which we will discuss in Section 2.3, where we will show how to accelerate SAG to make it "optimal".

Other incremental methods were then invented with similar complexity guarantees such as SDCA, SAGA, SVRG, or Finito/MISO, which will be presented in Section 2.2. All of them were found to be very efficient for fitting a linear model, showing that the theoretical gain observed in terms of complexity could lead to practical benefits in many cases. Their deployment for solving difficult non-convex problems, such as fitting a multilayer neural network, is however still an open research topic today.

Composite problems. Composite optimization arises in many scientific fields such as image and signal processing or machine learning. It consists of minimizing a real-valued function composed of two terms:

$$\min_{\mathbf{x}\in\mathbb{R}^p}\{F(\mathbf{x}) := f(\mathbf{x}) + \psi(\mathbf{x})\}, \tag{2.5}$$

where f is smooth with Lipschitz continuous derivatives, and ψ is a regularization function which is not necessarily differentiable. For instance, the ℓ_1-norm $\psi(\mathbf{x}) = \|\mathbf{x}\|_1$ is very popular in image processing (Elad, 2010; Mairal et al., 2014a) for encouraging sparse solutions; composite minimization also encompasses constrained minimization when considering extended-valued indicator functions ψ that may take the value $+\infty$ outside of a convex set $\mathcal{C}$ and 0 inside (see Hiriart-Urruty and Lemaréchal, 1996).

General non-smooth convex problems are harder to solve than smooth ones in terms of optimal complexity (Juditsky and Nemirovski, 2011). Yet, the class of composite problems is particularly interesting since instances from this class can often be solved as fast as if they were smooth ones. In general, algorithms that are dedicated to composite optimization only require to be able to compute efficiently the proximal operator of ψ:

$$\mathrm{Prox}_{\psi}[\mathbf{y}] := \arg\min_{\mathbf{x}\in\mathbb{R}^p}\Big\{\psi(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2\Big\}.$$

Note that when ψ is an indicator function, the proximal operator corresponds to the simple Euclidean projection. It turns out that the proximal operator admits closed form solutions in many interesting cases (see, *e.g.*, Bach et al., 2012, for a review).

The most classical approach for solving (2.5) is the proximal gradient method, known in the signal processing community as the iterative soft-thresholding approach when ψ is the ℓ_1-norm (Figueiredo and Nowak, 2003; Daubechies et al., 2004; Wright et al., 2009; Beck and Teboulle, 2009; Nesterov, 2013). The algorithm obeys the recursion

$$\mathbf{x}_t \leftarrow \mathrm{Prox}_{\frac{1}{L}\psi}\Big[\mathbf{x}_{t-1} - \frac{1}{L}\nabla f(\mathbf{x}_{t-1})\Big], \tag{2.6}$$

where L is the Lipschitz constant of ∇f, or an upper bound. When ψ is an indicator function, the relation (2.6) corresponds to the projected gradient descent technique. It

can be shown that the algorithm has the same convergence guarantee as the gradient descent method in the smooth case. When f is non-convex, it asymptotically provides a stationary point (see, *e.g.*, Mairal, 2013b), and in the convex case, it achieves ε-expected-optimality in $O(L/\varepsilon)$ iterations. When the objective is μ-strongly convex, the complexity becomes $O((L/\mu)\log(1/\varepsilon))$. Interestingly, it was also shown by Beck and Teboulle (2009) and Nesterov (2013) that the optimal gradient descent algorithm of Nesterov (1983) could be extended to the proximal setting. For instance, FISTA can be written as

$$\mathbf{x}_t \leftarrow \mathrm{Prox}_{\frac{1}{L}\psi}\Big[\mathbf{y}_{t-1} - \frac{1}{L}\nabla f(\mathbf{y}_{t-1})\Big] \quad \text{and} \quad \mathbf{y}_t \leftarrow \mathbf{x}_{t-1} + \beta_t(\mathbf{x}_t - \mathbf{x}_{t-1}),$$

where β_t in $(0, 1)$ is an extrapolation parameter that obeys a particular recursion (see Beck and Teboulle, 2009). Then, the computational complexity in the convex and μ-strongly-convex cases becomes the optimal ones for smooth functions, respectively $O(\sqrt{L/\varepsilon})$ and $O(\sqrt{L/\mu}\log(1/\varepsilon))$ iterations. In the latter case, the speed-up is of the order $\sqrt{L/\mu}$ which may be huge if the problem is badly conditioned, *e.g.*, when $L/\mu \geq 10\,000$.

When the function f is a convex expectation (2.4), proximal variants of the stochastic gradient descent algorithm may also be used such as FOBOS (Duchi and Singer, 2009), regularized dual averaging (RDA) (Xiao, 2010), or the stochastic mirror descent algorithm (see Juditsky and Nemirovski, 2011). In a nutshell, these methods have similar convergence guarantees as the stochastic gradient descent method applied to a smooth function. This also holds true for large finite sums (2.1). For instance, the methods SAGA, SVRG, SDCA, MISO are compatible with composite optimization. This will be also the case of all methods introduced subsequently in this chapter.

Separable constraints and/or regularization. Finally, we consider the case where the constraint set $\mathcal{C}$ or the function ψ have a separable structure:

$$\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \ldots \mathcal{C}_k \quad \text{and/or} \quad \psi(\mathbf{x}) = \sum_{j=1}^{k} \psi\Big(\mathbf{x}_{b(j)}\Big), \tag{2.7}$$

where the $b(j)$'s for $j = 1, \ldots, k$ are sets of indices forming a partition of $\{1, \ldots, p\}$. Then, $\mathbf{x}_{b(j)}$ represents the vector of size $|b(j)|$ corresponding of the entries of $\mathbf{x}$ associated to the indices $b(j)$. A particular case of interest is the Group Lasso penalty (Grandvalet and Canu, 1999; Turlach et al., 2005; Yuan and Lin, 2006):

$$\psi(\mathbf{x}) := \sum_{j=1}^{k} \|\mathbf{x}_{b(j)}\|_2, \tag{2.8}$$

which will be discussed in more details in Chapter 4. Such penalties are typically well suited to block coordinate descent algorithms, whose complexity per iteration is typically of the order $O(|b(j)|)$ instead of $O(p)$. Such methods for dealing with non-smooth penalties such as (2.8) were analyzed first by Tseng (2001). This chapter does not focus particularly on the separable case, which has been well explored by others (see also Nesterov, 2012; Richtárik and Takáč, 2014); nevertheless, one algorithm introduced in Section 2.1 is devoted to this setting.

Terminology	Convergence rate	Complexity
sub-linear	$O(1/\sqrt{t})$	$O(1/\varepsilon^2)$
sub-linear	$O(1/t)$	$O(1/\varepsilon)$
sub-linear	$O(1/t^2)$	$O(1/\sqrt{\varepsilon})$
linear	$O((1-\rho)^t)$	$O((1/\rho)\log(1/\varepsilon)$
super-linear	$O(e^{-O(\alpha^t)})$ with $\alpha > 1$	$O((1/\log(\alpha))\log\log(1/\varepsilon))$

Table 2.1 – We compare convergence rates—that is, the speed of convergence of the sequence $(f(\mathbf{x}_t) - f^\star)_{t\geq 0}$—and the corresponding complexities, which are the maximum number of iterations required to achieve $f(\mathbf{x}_t) - f^\star \leq \varepsilon$. Note that for better readability, we present all convergence rates in this chapter as asymptotic ones with the $O(.)$ notation. Non-asymptotic rates are available in the corresponding publications.

Discussions and organization of this chapter. We have presented so far a few problem structures that have motivated the contributions presented in this chapter.

In Section 2.1, we will introduce several variants of majorization-minimization algorithms to tackle the settings we have described previously. In most cases, we present algorithms with convergence rates and known worst-case complexities for convex and strongly-convex objective functions, and we show convergence to stationary points for non-convex ones. These methods often come as alternatives choices to known algorithms in the convex case, but their main asset is probably their ability to deal with some "exotic" difficult problems, for which no generic optimization method exists, such as the minimization of non-smooth non-convex stochastic optimization objectives.

Section 2.2 is devoted to algorithms dedicated to large sums of strongly-convex functions, which achieve typically lower complexity than the majorization-minimization algorithms presented in the previous section. Finally, Section 2.3 introduces two acceleration schemes for convex optimization, which are able to lower the complexity of a large class of gradient-based methods, either by generalizing Nesterov's acceleration (Nesterov, 1983), or by using Quasi-Newton principles (see Nocedal, 1980).

2.1 Majorization-minimization algorithms for structured problems

The principle of successively minimizing upper bounds of the objective function is often called *majorization-minimization* (Lange et al., 2000) or *successive upper-bound minimization* (Razaviyayn et al., 2016). Each upper bound is locally tight at the current estimate, and each minimization step decreases the value of the objective function. Even though this principle does not provide any theoretical guarantee about the quality of the returned solution, it has been very popular and widely used because of its simplicity. Many existing approaches can indeed be interpreted from the majorization-minimization point of view. For instance, this is the case of gradient-based or proximal methods (Wright et al., 2009), expectation-maximization (EM) algorithms in statistics (Dempster et al.,

1977; Neal and Hinton, 1998), difference-of-convex (DC) programming (Horst and Thoai, 1999), boosting (Collins et al., 2002; Della Pietra et al., 2001), some variational Bayes techniques used in machine learning (Wainwright and Jordan, 2008). Majorizing surrogates have also been used successfully in the signal processing literature about sparse optimization (Candès et al., 2008; Daubechies et al., 2004; Gasso et al., 2009), linear inverse problems in image processing (Ahn et al., 2006; Erdogan and Fessler, 1999), and matrix factorization (Lee and Seung, 2001; Mairal et al., 2010a).

The basic majorization-minimization principle for minimizing a function f over a convex set $\mathcal{C}$ of $\mathbb{R}^p$ is described in Algorithm 1 and illustrated in Figure 2.1. At iteration t, the estimate $\mathbf{x}_t$ is obtained by minimizing a surrogate function g_t of f. When g_t uniformly majorizes f and when $g_t(\mathbf{x}_{t-1}) = f(\mathbf{x}_{t-1})$, it is clear that the objective function value monotonically decreases.

Algorithm 1 Basic majorization-minimization scheme.

input $\mathbf{x}_0 \in \mathcal{C}$ (initial estimate); T (number of iterations).
1: **for** $t = 1, \ldots, T$ **do**
2: Compute a surrogate function g_t of f near $\mathbf{x}_{t-1}$;
3: Minimize the surrogate and update the solution: $\mathbf{x}_t \in \arg\min_{\theta \in \mathcal{C}} g_t(\mathbf{x})$.
4: **end for**
output $\mathbf{x}_T$ (final estimate);

For this approach to be effective, intuition tells us that we need functions g_t that are easy to minimize and that approximate well the objective f. Therefore, we measure the quality of the approximation through the smoothness of the error $h_t := g_t - f$, which is a key quantity arising in the convergence analysis. Specifically, we require h_t to be L-smooth for some constant $L > 0$ in the following sense:

Definition 1 (**L-smooth functions).** *A function $f : \mathbb{R}^p \to \mathbb{R}$ is called L-smooth when it is differentiable and when its gradient ∇f is L-Lipschitz continuous.*

With this definition in hand, we now introduce the class of "first-order surrogate functions", which will be shown to have good enough properties for the convergence analysis. First-order surrogates are interesting because their approximation error—the difference between the surrogate and the objective—can be easily controlled.

Definition 2 (**First-order surrogate functions).** *A function $g : \mathbb{R}^p \to \mathbb{R}$ is a first-order surrogate of f near $\mathbf{y}$ in $\mathcal{C}$ when*

- *$g(\mathbf{x}) \geq f(\mathbf{x})$ for all minimizers $\mathbf{x}$ of g over $\mathcal{C}$. When the more general condition $g \geq f$ holds, we say that g is **majorizing** f;*
- *the approximation error $h := g - f$ is L-smooth, $h(\mathbf{y}) = 0$, and $\nabla h(\mathbf{y}) = 0$.*

We denote by $\mathcal{S}_L(f, \mathbf{y})$ the set of such surrogates, and by $\mathcal{S}_{L,\rho}(f, \mathbf{y})$ the subset of ρ-strongly convex surrogates. [4]

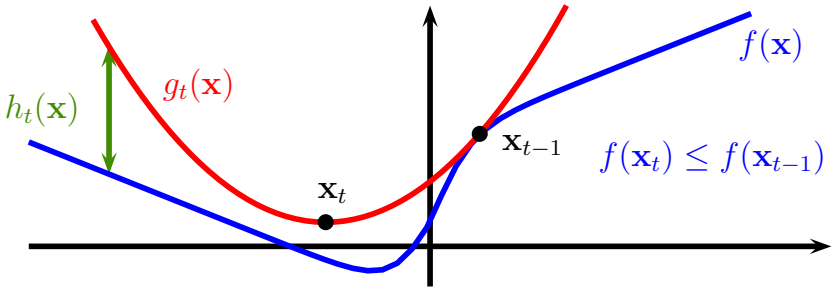


Figure 2.1 – Illustration of the basic majorization-minimization principle. We choose a surrogate g_t of f near the current estimate $\mathbf{x}_{t-1}$. The new estimate $\mathbf{x}_t$ is a minimizer of g_t. The function $h_t = g_t - f$ represents the approximation error.

In (Mairal, 2015), we present simple convergence results regarding Algorithm 1 with first-order surrogate functions g_t. More precisely, we show that, under simple assumptions, the sequence of iterates asymptotically satisfies a stationary point condition (see, *e.g.*, Bertsekas, 1999). To do so, we make the following mild assumption when f is non-convex:

(A) f is bounded below and for all $\mathbf{x}, \mathbf{y}$ in $\mathcal{C}$, the directional derivative $\nabla f(\mathbf{x}, \mathbf{y} - \mathbf{x})$ of f at $\mathbf{x}$ in the direction $\mathbf{y} - \mathbf{x}$ exists.

A necessary condition for $\mathbf{x}$ to be a local minimum of f is to have $\nabla f(\mathbf{x}, \mathbf{y} - \mathbf{x}) \geq 0$ for all $\mathbf{y}$ in Θ (see, *e.g.*, Borwein and Lewis, 2006). In other words, there is no feasible descent direction $\mathbf{y} - \mathbf{x}$ and $\mathbf{x}$ is a stationary point. Thus, we consider the following condition for assessing the quality of a sequence $(\mathbf{x}_t)_{t \geq 0}$ for non-convex problems:

Definition 3 (Asymptotic stationary point). *Under assumption **(A)**, a sequence $(\mathbf{x}_t)_{t \geq 0}$ satisfies the asymptotic stationary point condition if*

$$\liminf_{t \to +\infty} \inf_{\mathbf{y} \in \mathcal{C}} \frac{\nabla f(\mathbf{x}_t, \mathbf{y} - \mathbf{x}_t)}{\|\mathbf{y} - \mathbf{x}_t\|_2} \geq 0. \tag{2.9}$$

Note that if f is differentiable on $\mathbb{R}^p$ and $\mathcal{C} = \mathbb{R}^p$, $\nabla f(\mathbf{x}_t, \mathbf{y} - \mathbf{x}_t) = \nabla f(\mathbf{x}_t)^\top (\mathbf{y} - \mathbf{x}_t)$, and the condition (2.9) implies that the sequence $(\nabla f(\mathbf{x}_t))_{t \geq 0}$ converges to 0.

After showing that the condition (2.9) is satisfied by Algorithm 1 as soon as as the surrogates g_t are in $\mathcal{S}_L(f, \mathbf{x}_{t-1})$ and are majorizing or strongly-convex, we present a similar result with relaxed assumptions on g_t when f is a composition of two functions (see Mairal, 2015, Section 2.1). Finally, we present non-asymptotic convergence rates when f is convex. By adapting convergence proofs of proximal gradient methods (Nesterov, 2004) to our more general setting, we recover classical non-asymptotic sublinear convergence rates and linear ones for strongly convex problems (see Mairal, 2015, Section 2.2).

4. Unless otherwise specified, strong convexity is always defined with respect to the Euclidean norm in this thesis.

2.1.1 Examples of first-order surrogate functions.

We now introduce practical first-order surrogate functions and draw links between Algorithm 1 and existing approaches. Even though the previous generic convergence analysis does not always bring new results for each specific case, it provides a unique theoretical treatment to all of them.

Lipschitz gradient surrogates. When f is L-smooth, we naturally consider:

$$g : \mathbf{x} \mapsto f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2.$$

The function g is an upper bound of f, which is a classical result (Nesterov, 2004). It is then easy to see that g is L-strongly convex and L-smooth. As a consequence, the difference $g - f$ is $2L$-smooth (as a sum of two L-smooth functions), and thus g is in $\mathcal{S}_{2L,L}(f, \mathbf{y})$. When f is convex, it is also possible to show that g is in fact in $\mathcal{S}_{L,L}(f, \mathbf{y})$, and when f is μ-strongly convex, g is in $\mathcal{S}_{L-\mu,L}(f, \mathbf{y})$. We remark that minimizing g amounts to performing a gradient descent step: $\mathbf{x}' \leftarrow \mathbf{y} - \frac{1}{L}\nabla f(\mathbf{y})$.

Proximal gradient surrogates. Let us now consider the composite optimization problem (2.5), where f is L-smooth. Then, a surrogate of $F = f + \psi$ is the following function:

$$g : \mathbf{x} \mapsto f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + \psi(\mathbf{x}).$$

The function g is an upper bound of F and the approximation error $g - F$ is the same as for Lipschitz gradient surrogates in the smooth case. Minimizing g amounts to performing a proximal gradient descent step (2.6).

Linearizing concave functions and DC programming. Assume that F is a sum of two functions $f + \psi$, where ψ is concave and L-smooth. Then, the following function g is a majorizing surrogate in $\mathcal{S}_L(F, \mathbf{y})$:

$$g : \mathbf{x} \mapsto f(\mathbf{x}) + \psi(\mathbf{y}) + \nabla \psi(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}).$$

Such a surrogate appears in DC (difference of convex) programming (Horst and Thoai, 1999). When f is convex, F is indeed the difference of two convex functions. It is also used in sparse estimation for dealing with some non-convex penalties.

Variational surrogates. We now consider a real-valued function $\tilde{f}$ defined on $\mathbb{R}^p \times \mathbb{R}^{p'}$ and two convex sets $\mathcal{C} \subseteq \mathbb{R}^p$ and $\mathcal{C}' \subseteq \mathbb{R}^{p'}$. Minimizing $\tilde{f}$ over $\mathcal{C} \times \mathcal{C}'$ is equivalent to minimizing the function f over $\mathcal{C}$ defined as $f : \mathbf{x} \mapsto \min_{\mathbf{x}' \in \mathcal{C}'} \tilde{f}(\mathbf{x}, \mathbf{x}')$. Assume now that

- $\mathbf{x}' \mapsto \tilde{f}(\mathbf{x}, \mathbf{x}')$ is μ-strongly convex for all $\mathbf{x}$ in $\mathbb{R}^p$;
- $\mathbf{x} \mapsto \tilde{f}(\mathbf{x}, \mathbf{x}')$ is L-smooth for all $\mathbf{x}'$;
- $\nabla_{\mathbf{x}} \tilde{f}(\mathbf{x}, \mathbf{x}')$ is L'-Lipschitz with respect to $\mathbf{x}'$.

Let us fix $\mathbf{y}$ in $\mathcal{C}$. Then, the following function is a majorizing surrogate in $\mathbf{S}_{L''}(f, \mathbf{y})$:

$$g : \mathbf{x} \mapsto f(\mathbf{x}, \mathbf{y}'^{\star}) \text{ with } \mathbf{y}'^{\star} := \operatorname*{arg\,min}_{\mathbf{y}' \in \mathcal{C}'} f(\mathbf{y}, \mathbf{y}'),$$

with $L'' = 2L + L'^2/\mu$. Minimizing the surrogate g leads to an alternate minimization algorithm; it is then interesting to note that the convergence results we provide in the convex case provide similar rates as the recent analysis of Beck and Tetruashvili (2013), which makes slightly different assumptions on the function f. Variational surrogates may be used for matrix factorization and for sparse estimation, where the ℓ_1-norm and other variants admit smoothed variational forms (see Bach et al., 2012).

Jensen surrogates. Jensen's inequality also provides a natural mechanism to build surrogates for convex functions. Following the presentation of Lange et al. (2000), we consider a convex function $\tilde{f} : \mathbb{R} \mapsto \mathbb{R}$, a vector $\boldsymbol{\xi}$ in $\mathbb{R}^p$, and we define $\tilde{f} : \mathbb{R}^p \to \mathbb{R}$ as $f(\mathbf{x}) := \tilde{f}(\boldsymbol{\xi}^\top \mathbf{x})$. We also consider $\mathbf{w}$ a weight vector in $\mathbb{R}^p_+$ such that $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w}[j] \neq 0$ whenever $\boldsymbol{\xi}[j] \neq 0$. Then, we define for any $\mathbf{y}$ in $\mathbb{R}^p$:

$$g : \theta \mapsto \sum_{j=1}^{p} \mathbf{w}[j] \tilde{f}\left(\frac{\boldsymbol{\xi}[j]}{\mathbf{w}[j]}(\mathbf{x}[j] - \mathbf{y}[j]) + \boldsymbol{\xi}^\top \mathbf{y}\right).$$

When f is L-smooth, and when $\mathbf{w}[j] := |\mathbf{x}[j]|^\nu / \|\boldsymbol{\xi}\|_\nu^\nu$, g is in $\mathbf{S}_{L'}(f, \mathbf{y})$ with $L' = L\|\boldsymbol{\xi}\|_\infty^2 \|\boldsymbol{\xi}\|_0$ for $\nu = 0$; $L' = L\|\boldsymbol{\xi}\|_\infty \|\boldsymbol{\xi}\|_1$ for $\nu = 1$; and $L' = L\|\boldsymbol{\xi}\|_2^2$ for $\nu = 2$. Jensen surrogates are in fact relatively uncommon but they appear in a few occasions. In addition to the few examples given by Lange et al. (2000), they are used for instance in machine learning by Della Pietra et al. (2001) for interpreting boosting procedures.

Quadratic surrogates. Finally, when f is twice differentiable and admits a global upper-bound matrix $\mathbf{H}$ on the Hessian $\nabla^2 f$—that is, such that $\mathbf{H} - \nabla^2 f$ is always positive definite—the following function is a first-order majorizing surrogate:

$$g : \mathbf{x} \mapsto f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{1}{2}(\mathbf{x} - \mathbf{y})^\top \mathbf{H} (\mathbf{x} - \mathbf{y}).$$

The Lipschitz constant of $\nabla(g - f)$ is the largest eigenvalue of $\mathbf{H} - \nabla^2 f(\mathbf{x})$ over $\mathcal{C}$. Such surrogates appear frequently in the statistics and machine learning literature (Böhning and Lindsay, 1988; Jebara and Choromanska, 2012; Khan et al., 2010).

2.1.2 Stochastic optimization.

After presenting the basic majorization-minimization scheme and various first-order surrogate functions, we are now interested in the stochastic optimization problem (2.4), which was tackled in (Mairal, 2013a). The resulting scheme consists of iteratively building a surrogate of the expected cost when only a single data point is observed at each iteration; this data point is used to update the surrogate, which in turn is minimized to obtain a new estimate. The scheme is presented in Algorithm 2. Some previous works are

closely related to this scheme: the online EM algorithm for latent data models (Neal and Hinton, 1998; Cappé and Moulines, 2009) and the online matrix factorization technique of Mairal et al. (2010a) involve for instance surrogate functions updated in a similar fashion. Compared to these two approaches, the stochastic MM method is targeted to more general optimization problems.

Algorithm 2 Stochastic Majorization-Minimization Scheme

input $\mathbf{x}_0 \in \mathcal{C}$ (initial estimate); T (number of iterations); $(w_t)_{t\geq 1}$, weights in $(0, 1]$;

1: initialize the approximate surrogate: $\bar{g}_0 : \mathbf{x} \mapsto \frac{\rho}{2}\|\mathbf{x} - \mathbf{x}_0\|_2^2$; $\bar{\mathbf{x}}_0 = \mathbf{x}_0$; $\hat{\mathbf{x}}_0 = \mathbf{x}_0$;
2: **for** $t = 1, \ldots, T$ **do**
3: draw a training point $\boldsymbol{\xi}_t$; define $f_t : \mathbf{x} \mapsto f(\boldsymbol{\xi}_t, \mathbf{x})$;
4: choose a surrogate function g_t in $\mathcal{S}_{L,\rho}(f_t, \mathbf{x}_{t-1})$;
5: update the approximate surrogate: $\bar{g}_t = (1 - w_t)\bar{g}_{t-1} + w_t g_t$;
6: update the current estimate:

$$\mathbf{x}_t \in \arg\min_{\mathbf{x}\in\mathcal{C}} \bar{g}_t(\mathbf{x});$$

7: **end for**

output $\mathbf{x}_T$ (current estimate) or a weighted average of the past estimates;

We remark that Algorithm 2 is only practical when the aggregated surrogates $\bar{g}_t$ can be parameterized with a small number of variables, and when they can be easily minimized over $\mathcal{C}$. This is notably the case for Lipschitz proximal gradient surrogates, and also for some variational surrogates (see Mairal et al., 2010a).

Theoretical analysis. We study the convergence of the algorithm when using strongly convex first-order surrogate functions. For convex objectives, proximal gradient surrogates lead to expected convergence rates that are asymptotically optimal, or close to optimal (Nemirovski et al., 2009). More precisely, the convergence rate is of order $O(1/\sqrt{t})$ in a finite horizon setting, and $O(1/t)$ for a strongly convex objective in an infinite horizon setting (see Mairal, 2013a, Sections 3.1 and 3.2), by using particular averaging schemes of the iterates, which are not presented in Algorithm 2 for simplicity. For *non-convex* problems and more general first-order surrogates, the method almost surely converges to a set of stationary points under suitable assumptions. We believe that this result is equally valuable as convergence rates for convex optimization. To the best of our knowledge, the literature on stochastic non-convex non-smooth optimization is rather scarce, and we are only aware of convergence results in more restricted settings than ours—see for instance (Bottou, 1998) for the stochastic gradient descent algorithm, (Cappé and Moulines, 2009) for online EM, (Mairal et al., 2010a) for online matrix factorization, or (Ghadimi and Lan, 2013) for unconstrained smooth problems.

Instances of the algorithm. The first instance is a new stochastic proximal gradient method for composite or constrained optimization. This algorithm is related to a long series of work in the convex optimization literature (Duchi and Singer, 2009; Hazan et al.,

2007; Lan, 2012; Langford et al., 2009; Nemirovski et al., 2009; Xiao, 2010). Specifically, let us consider the composite stochastic objective

$$\min_{\mathbf{x}\in\mathcal{C}} \mathbb{E}_{\boldsymbol{\xi}}[f(\boldsymbol{\xi},\mathbf{x})] + \psi(\mathbf{x}),$$

where ψ is a convex deterministic regularization function, and the functions $\mathbf{x} \mapsto f(\boldsymbol{\xi},\mathbf{x})$ are L-smooth for all $\boldsymbol{\xi}$. Let us now choose a weight sequence $(w_t)_{t\geq 1}$ such that $w_1 = 1$. By defining some other weights w_t^i recursively as $w_t^i := (1-w_t)w_t^{i-1}$ for $i<t$ and $w_t^t := w_t$, we can use the proximal gradient surrogate, leading to the update rule

$$\mathbf{x}_t \leftarrow \arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{t} w_t^i \Big[\nabla f_i(\mathbf{x}_{i-1})^\top \mathbf{x} + \frac{L}{2}\|\mathbf{x}-\mathbf{x}_{i-1}\|_2^2 + \psi(\mathbf{x}) \Big]. \tag{SMM}$$

This algorithm is related to FOBOS (Duchi and Singer, 2009) and to the truncated gradient method (Langford et al., 2009), which use the update rule

$$\mathbf{x}_t \leftarrow \arg\min_{\mathbf{x}\in\mathcal{C}} \nabla f_t(\mathbf{x}_{t-1})^\top \mathbf{x} + \frac{1}{2\eta_t}\|\mathbf{x}-\mathbf{x}_{t-1}\|_2^2 + \psi(\mathbf{x}). \tag{FOBOS}$$

Another related scheme is the regularized dual averaging (RDA) of Xiao (2010):

$$\mathbf{x}_t \leftarrow \arg\min_{\mathbf{x}\in\mathcal{C}} \frac{1}{t}\sum_{i=1}^{t} \nabla f_i(\mathbf{x}_{i-1})^\top \mathbf{x} + \frac{1}{2\eta_t}\|\mathbf{x}\|_2^2 + \psi(\mathbf{x}). \tag{RDA}$$

Compared to these approaches, the stochastic majorization-minimization scheme (SMM) includes a weighted average of previously seen gradients, and a weighted average of the past iterates. Some links can also be drawn with approaches such as the "approximate follow the leader" algorithm of Hazan et al. (2007).

Another application of Algorithm 2 is a stochastic DC programming technique, which has been demonstrated to be better than batch alternatives for large-scale non-convex sparse estimation problems. Finally, the last application introduced in (Mairal, 2013a) is an online scheme for factorizing large-scale structured matrices with sparse regularization.

2.1.3 Algorithm for large (non-convex) finite sums

In a different context, incremental EM algorithms have been proposed by Neal and Hinton (1998), where upper bounds of a non-convex negative log-likelihood function are incrementally updated. By using similar ideas, we introduce the scheme MISO in Algorithm 3 for minimizing a large sum of functions (2.1). At every iteration, a single function is observed, and an approximate surrogate of f is updated.

The convergence analysis of the algorithm is presented in Section 3.1 of Mairal (2015). The main conclusion for non-convex problems is that the method achieves similar guarantees as the basic majorization-minimization scheme with probability one, even though it could be potentially much faster in practice when n is large. On the other hand, the conclusion for convex optimization is a bit mitigated. When using proximal gradient surrogates, the expected complexity of the algorithm is the same as the basic

Algorithm 3 Incremental scheme MISO-MM.

input $\mathbf{x}_0 \in \mathcal{C}$ (initial estimate); T (number of iterations).

1: Initialization: choose some surrogates g_0^i of f^i near $\mathbf{x}_0$ for all i;
2: **for** $t = 1, \ldots, T$ **do**
3: Randomly pick up one index $\hat{\imath}_t$ and choose a surrogate $g_t^{\hat{\imath}_t}$ of $f^{\hat{\imath}_t}$ near $\mathbf{x}_{t-1}$; set $g_t^i := g_{t-1}^i$ for all $i \neq \hat{\imath}_t$.
4: Update the solution: $\mathbf{x}_t \in \arg\min_{\mathbf{x}\in\mathcal{C}} \frac{1}{n} \sum_{i=1}^n g_t^i(\mathbf{x})$.
5: **end for**

output $\mathbf{x}_T$ (final estimate);

majorization-minimization scheme that does not exploit the problem structure. We will see in Section 2.2 that in fact the majorization-minimization principle is too conservative for convex optimization. Significantly faster convergence rates can indeed be obtained with an Algorithm similar to 3 by using quadratic lower bounds instead of upper bounds.

2.1.4 A block-coordinate majorization-minimization algorithm

A block-coordinate variant of the majorization-minimization principle has also been proposed in (Mairal, 2013b). Assuming that the constraint set (or the regularization function ψ in the composite setting) is separable as in (2.7), it is natural to consider first-order surrogate functions with the same property

$$g_t(\mathbf{x}) = \sum_{j=1}^k g_t^j(\mathbf{x}_{s(j)}) \quad \text{for} \quad \mathbf{x} = (\mathbf{x}_{s(1)}, \ldots, \mathbf{x}_{s(k)}) \in \mathcal{C}.$$

Then, we present a block-coordinate variant of the majorization-minimization scheme in Algorithm 4.

Algorithm 4 Block Coordinate Descent Scheme BCD-MM

input $\mathbf{x}_0 \in \mathcal{C} = (\mathcal{C}_1 \times \ldots \times \mathcal{C}_k)$; T (number of iterations).

1: **for** $t = 1, \ldots, T$ **do**
2: Choose a separable surrogate g_t of f near $\mathbf{x}_{t-1}$;
3: Randomly pick up one block $\hat{\imath}_t$ and update $\mathbf{x}_{t,s(\hat{\imath}_t)}$:

$$\mathbf{x}_{t,s(\hat{\imath}_t)} \in \arg\min_{\mathbf{x}\in\mathcal{C}_{i_t}} g_t^{\hat{\imath}_t}(\mathbf{x}).$$

4: **end for**

output $\mathbf{x}_T$ (final estimate);

As always, we study its convergence properties (see Mairal, 2013a, Section 3). For non-convex problems, the same guarantees as the basic majorization-minimization algorithm hold with probability one, and for convex problems, we recover the classical non-asymptotic sublinear $O(1/t)$ rates for convex problems and linear ones for strongly-convex ones. These results are general since they hold for all first-order surrogate functions.

2.1.5 Frank-Wolfe variant

When I was working on majorization-minimization schemes in 2013, the Frank-Wolfe algorithm started to gain a lot of popularity in machine learning (Jaggi, 2013; Harchaoui et al., 2015). This principle has nothing to do with majorization-minimization since it relies on lower bounds of the objective. Yet, lower bounds of f can be obtained from first-order upper bounds g in $\mathcal{S}_L(f, \mathbf{y})$ as $\mathbf{x} \mapsto g(\mathbf{x}) - \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2$. It is also possible to show that proof techniques used in the analysis of the previous majorization-minimization algorithms could be used in the context of Frank-Wolfe, which motivated me to propose a Frank-Wolfe variant for composite optimization with convergence rate $O(1/t)$ (see Mairal, 2013a, Section 4). This approach is presented in Algorithm 5. When f is L-smooth and the gradient Lipschitz surrogates are used, the function minimized in (2.10) is linear and we recover the classical Frank-Wolfe method. [5] Our point of view is slightly more general since it allows to use proximal gradient surrogates for composite optimization.

Algorithm 5 Frank-Wolfe Scheme

input $\mathbf{x}_0 \in \mathcal{C}$; T (number of iterations).

1: **for** $t = 1, \ldots, T$ **do**

2: Let g_t be a majorizing surrogate in $\mathcal{S}_{L,L}(f, \mathbf{x}_{t-1})$.

3: Compute a search direction:

$$\boldsymbol{\nu}_t \in \arg\min_{\mathbf{x} \in \mathcal{C}} \Big[g_t(\mathbf{x}) - \frac{L}{2}\|\mathbf{x} - \mathbf{x}_{t-1}\|_2^2 \Big]. \tag{2.10}$$

4: Line search: $\alpha^\star := \arg\min_{\alpha \in [0,1]} g_t(\alpha \boldsymbol{\nu}_t + (1-\alpha)\mathbf{x}_{t-1})$.

5: Update solution: $\mathbf{x}_t := \alpha^\star \boldsymbol{\nu}_t + (1-\alpha^\star)\mathbf{x}_{t-1}$.

6: **end for**

output $\mathbf{x}_T$ (final estimate);

2.1.6 Second-order surrogates and cubic regularization

Finally, it is natural to ask if one may define second-order surrogate functions. This can be done in fact via the concept of cubic regularization (Nesterov and Polyak, 2006). Consider indeed the basic majorization-minimization scheme and consider first-order surrogate functions g_t that satisfy the following additional properties: the error functions $h_t := g_t - f$ are twice differentiable, their Hessians $\nabla^2 h_t$ are M-Lipschitz continuous, and $\nabla^2 h_t(\mathbf{x}_{t-1}) = 0$ for all t. Then, the conference rate of the sequence $f(\mathbf{x}_t) - f^\star$ is in $O(1/t^2)$ for convex problems, and superlinear with order $3/2$ for strongly convex ones (see Mairal, 2013a, Proposition 2.4). When using gradient Lipschitz surrogate functions, we simply recover the results of Nesterov and Polyak (2006). Note that such second-order surrogate functions are typically nonconvex and difficult to minimize beyond the case studied by Nesterov and Polyak (2006). It is thus not clear that they are useful in practice, even though they are conceptually interesting since they provide fast convergence rates.

5. Note that the classical Frank-Wolfe algorithm performs the line search over f and not over g_t.

2.2 Variance-reduced stochastic optimization for convex optimization

We may now move to another class of algorithms dedicated to strongly-convex objective functions, which instead of using upper bounds as in the previous section, leverage lower bounds provided by the strong convexity inequality:

$$f(\mathbf{x}) \geq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|_2^2,$$

which holds for any pair $\mathbf{x}, \mathbf{y}$ in $\mathbb{R}^p$ and any μ-strongly convex function f. Note that the inequality also holds for non-smooth functions, by replacing $\nabla f(\mathbf{y})$ by any subgradient. Using lower bounds to build a model of an objective is a classical principle in convex optimization. For instance, this is a key ingredient of Kelley's cutting plane and bundle methods (Kelley, 1960; Hiriart-Urruty and Lemaréchal, 1996), and also Nesterov's estimate sequences used in the accelerated gradient method (Nesterov, 2004).

2.2.1 The MISO algorithm for strongly convex functions

The MISO algorithm for minimizing a sum of strongly convex smooth functions was proposed in (Mairal, 2015); it was then extended by my student Hongzhou Lin to the composite setting (Lin et al., 2015), which we consider now:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \Big\{ F(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{x}) + \psi(\mathbf{x}) \Big\},$$

where the f_i's are L-smooth and μ-strongly convex, and ψ is convex. The MISO approach for this problem can be described as a variant of Algorithm 3, where instead of first-order majorizing surrogates g_t, lower bounds d_t are used instead:

$$d_t^{\hat{\imath}_t}(\mathbf{x}) = (1-\delta) d_t^{i-1}(\mathbf{x}) + \delta\Big(f_{\hat{\imath}_t}(\mathbf{x}_{t-1}) + \nabla f_{\hat{\imath}_t}(\mathbf{x}_{t-1})^\top (\mathbf{x} - \mathbf{x}_{t-1}) + \frac{\mu}{2}\|\mathbf{x}_{t-1} - \mathbf{x}\|_2^2 + \psi(\mathbf{x}) \Big). \tag{2.11}$$

and $d_t^i = d_{t-1}^i$ for $i \neq \hat{\imath}_t$. There, $\delta = \min(1, \mu n/2(L-\mu))$ in $(0,1]$. Under the "big data" condition $n \geq 2(L-\mu)/\mu \approx 2L/\mu$, we have $\delta = 1$ and thus the update is particularly simple. The current iterate $\mathbf{x}_t$ is then obtained by minimizing the lower bound of F:

$$\mathbf{x}_t = \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^p} \Big\{ D_t(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^n d_t^i(\mathbf{x}) \Big\}.$$

By parametrizing the surrogates d_t^i as

$$d_t^i(\mathbf{x}) = c_t^i - \mu\langle \mathbf{x}, \mathbf{z}_t^i \rangle + \frac{\mu}{2}\|\mathbf{x}\|^2 + \psi(\mathbf{x}),$$

we obtain that the update (2.11) can be implemented as

$$\mathbf{z}_t^i = \begin{cases} (1-\delta)\mathbf{z}_{t-1}^i + \delta\Big(\mathbf{x}_{t-1} - \frac{1}{\mu}\nabla f_i(\mathbf{x}_{t-1})\Big), & \text{if } i = \hat{\imath}_t \\ \mathbf{z}_{t-1}^i, & \text{otherwise.} \end{cases}$$

and $\mathbf{x}_t$ can be obtained by computing the proximal operator

$$\mathbf{x}_t = \operatorname{Prox}_{\frac{\psi}{\mu}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathbf{z}_t^i\right].$$

The resulting algorithm can be interpreted as a variant of proximal SDCA (Shalev-Shwartz and Zhang, 2013), which uses the same update with a slightly different step size δ. The difference with MISO lies in fact in two aspects: first, MISO provides a more practical optimality certificate, which will be presented in the sequel. Specifically, checking the optimality condition does not require evaluating a dual objective. Second, the construction is indeed purely *primal*. Neither the proof of convergence nor the algorithm use duality, while SDCA is originally a dual ascent technique.

More precisely, since D_t is a lower bound of F we also have $D_t(\mathbf{x}_t) \leq F^\star$, and thus the quantity $F(\mathbf{x}_t) - D_t(\mathbf{x}_t)$ can be used as an optimality certificate that upper-bounds $F(\mathbf{x}_t) - F^\star$. Furthermore, this certificate was shown to converge to zero with a rate similar to SAG/SDCA/SVRG/SAGA, as stated in the next theorem.

Theorem 1 (Convergence of MISO-Prox). *The sequence $(\mathbf{x}_t)_{t\geq 0}$ produced by MISO-Prox satisfies*

$$\mathbb{E}[F(\mathbf{x}_t)] - F^* \leqslant \frac{1}{\tau}(1-\tau)^{t+1}(F(\mathbf{x}_0) - D_0(\mathbf{x}_0)) \quad \textit{with} \quad \tau \geqslant \min\left\{\frac{\mu}{4L}, \frac{1}{2n}\right\}. \tag{2.12}$$

Furthermore, we also have fast convergence in expectation of the certificate

$$\mathbb{E}[F(\mathbf{x}_t) - D_t(\mathbf{x}_t)] \leqslant \frac{1}{\tau}(1-\tau)^{t}(F^* - D_0(\mathbf{x}_0)).$$

The speed of convergence is similar to other incremental approaches such as SAG, SDCA, SAGA, and SVRG. In fact, like SVRG, the MISO update in the smooth case can be interpreted as a stochastic gradient descent step with an unbiased estimate of the gradient, but with a smaller variance than SGD.

2.2.2 The stochastic MISO algorithm

Finally, we consider in (Bietti and Mairal, 2017b) a hybrid case between purely stochastic and finite-sum settings, motivated by the following observations. In many situations, augmenting a finite training set with well-chosen random perturbations of each example can lead to smaller test error in theory (Wager et al., 2014) and in practice (Simard et al., 1998). Examples of such procedures include random transformations of images in classification problems (*e.g.*, Simard et al., 1998), and Dropout (Srivastava et al., 2014). The objective describing these scenarios, which we consider in this section, is the following:

$$\min_{\mathbf{x}\in\mathbb{R}^p}\left\{F(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) + \psi(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\boldsymbol{\rho}}[\tilde{f}_i(\mathbf{x},\boldsymbol{\rho})] + \psi(\mathbf{x})\right\}, \tag{2.13}$$

where $\boldsymbol{\rho}$ parametrizes the random perturbation, $\tilde{f}_i(\cdot, \boldsymbol{\rho})$ is always a convex L-smooth function, and ψ is a convex regularization. We also assume that F is μ-strongly convex. Because each function f_i is an expectation, computing a single gradient ∇f_i is intractable in general, and standard variance reduction methods such as SAG, SVRG, or SDCA cannot be used. A straightforward way to optimize this objective is to use SGD (or a proximal variant if $\psi \neq 0$) by choosing an index $\hat{\imath}_t$ randomly in $\{1, \dots, n\}$ at iteration t, sampling a perturbation $\boldsymbol{\rho}_t$, and performing the update $\mathbf{x}_t = \mathbf{x}_{t-1} - \eta_t \nabla \tilde{f}_{\hat{\imath}_t}(\mathbf{x}_{t-1}, \boldsymbol{\rho}_t)$, where η_t is a step-size. Note that this approach ignores the finite-sum structure in the objective and thus leads to gradient estimates with high variance. To address this issue, we introduce in (Bietti and Mairal, 2017b) an algorithm, called *stochastic MISO*, which can exploit the problem structure using variance reduction. Specifically, the updates are similar to (2.11) but δ is replaced by a value α_t at iteration t, with $(\alpha_t)_{t \geq 1}$ decreasing at a specific rate, and the gradient $\nabla f_{\hat{\imath}_t}(\mathbf{x}_{t-1})$ in (2.11) is replaced by $\nabla \tilde{f}_{\hat{\imath}_t}(\mathbf{x}_{t-1}, \boldsymbol{\rho}_t)$ where $\boldsymbol{\rho}_t$ is a random perturbation sampled at time t.

Convergence analysis. The method achieves a $O(1/t)$ convergence rate like SGD, but with a constant factor that is much smaller in typical settings, only depending on the variance of the gradient estimates due to the random perturbations on a single example. Specifically, we make the following low-variance assumption at the optimum:

$$\mathbb{E}_{\boldsymbol{\rho}}\Big[\|\nabla \tilde{f}_i(\mathbf{x}^\star, \boldsymbol{\rho}) - \nabla f_i(\mathbf{x}^\star)\|_2^2\Big] \leq \sigma^2,$$

for all i, where $\mathbf{x}^\star$ is the (unique) minimizer of F. In contrast, a standard assumption for the SGD algorithm on the objective (2.13) would take the form $\mathbb{E}_{i,\boldsymbol{\rho}}[\|\nabla \tilde{f}_i(\mathbf{x}, \boldsymbol{\rho})\|_2^2] \leq M^2$ for all $\mathbf{x}$. The quantity M^2 takes into account the noise induced by the random index i in addition to $\boldsymbol{\rho}$, and can thus be much larger than σ^2, particularly if the perturbations on input data are small. We show in (Bietti and Mairal, 2017b) that after an initial linearly convergent phase, and under appropriate choice of step-sizes $(\alpha_t)_{t \geq 1}$, the stochastic MISO algorithm will satisfy $\mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}^*)] \leq \epsilon$ after

$$O\left(\frac{L\sigma^2}{\mu^2 \epsilon}\right)$$

iterations and $O(\sigma^2/(\mu\epsilon))$ with a particular averaging scheme in the smooth case. This complexity is similar to that of SGD (Bottou et al., 2016; Nemirovski et al., 2009), but with σ^2 replacing the quantity M^2, leading to a much faster rate than SGD if $\sigma^2 \ll M^2$, something which we observed in experiments (Bietti and Mairal, 2017b).

2.3 Generic acceleration by smoothing

The last contribution of this chapter is a generic method for accelerating a large class of gradient-based optimization algorithms by using a smoothing technique, which may seem counter-intuitive since smoothing and acceleration are apparently unrelated concepts. Yet, we show in Section 2.3.1 that it is possible to provide Nesterov's acceleration

to originally unaccelerated algorithms with this idea, leading to faster theoretical and practical convergence rates (Lin et al., 2015). Then, we extend this approach by using limited-memory Quasi-Newton principles (Lin et al., 2017) in Section 2.3.2, which provides significant speed-up in practice for solving ill-conditioned problems.

The Moreau-Yosida smoothing. Specifically, we consider the Moreau-Yosida regularization of a function f, defined as the infimal convolution

$$F(\mathbf{x}) := \min_{\mathbf{z}\in\mathbb{R}^p} \left\{ f(\mathbf{z}) + \frac{\kappa}{2}\|\mathbf{z}-\mathbf{x}\|^2 \right\}, \tag{2.14}$$

where κ is a positive scalar. Note that the solution of the optimization problem (2.14) is the proximal operator $\mathbf{z}^\star = \text{Prox}_{f/\kappa}[\mathbf{x}]$. The Moreau-Yosida regularization admits classical properties, which we present below (see Lemaréchal and Sagastizábal, 1997).

Proposition 1 (**Basic properties of the Moreau-Yosida regularization**). *The Moreau-Yosida regularization F of the convex function f has the following properties:*

1. *F is convex and minimizing f and F is equivalent in the sense that*

$$\min_{\mathbf{x}\in\mathbb{R}^p} F(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}),$$

   and the solution set of the two above problems coincide with each other.

2. *F is κ-smooth even when f is not differentiable and*

$$\nabla F(\mathbf{x}) = \kappa(\mathbf{x} - \text{Prox}_{f/\kappa}[\mathbf{x}]). \tag{2.15}$$

3. *When f is μ-strongly convex, F is μ_F-strongly convex with constant $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$.*

These observations yield a simple strategy for minimizing any convex function f, by simply minimizing F with an algorithm that is able to handle smooth functions. Such an approach is appealing but it raises several difficulties. In particular, the exact gradient of F depends on the proximal operator of f—equivalently the exact solution of (2.14)—for which no closed-form is available in general. It is thus necessary to use an approximate solution, which implies defining an inexactness criterion that is easy to check and to control the accuracy of the gradient approximation to ensure (fast) convergence.

The generic recipe for acceleration. It is interesting to see that this strategy can be used not only for *smoothing* a non-smooth objective, but in fact for *accelerating* first-order optimization techniques. That may be achieved with the following recipe:

1. Choose a fast generic gradient-based method $\mathcal{A}$ for minimizing the smooth function F, *e.g.*, Nesterov's accelerated gradient method, or a Quasi-Newton approach.
2. Choose a dedicated first-order optimization method $\mathcal{M}$ that is able to exploit the structure of f (*e.g.*, finite sum, composite), and which we wish to accelerate.
3. Apply $\mathcal{A}$ on F with inexact gradients: at each iteration of $\mathcal{A}$, use the method $\mathcal{M}$ to solve approximately the sub-problems (2.14).

The success of the approach relies on two ingredients. First, we need to choose the right value κ: increasing κ improves the conditioning of the sub-problems (2.14), but it also degrades the conditioning of the function F; an optimal trade-off is then required. Second, we need to choose a good inexactness criterion and the right accuracy for solving the sub-problems. The theory provides an answer to the two points above and the procedure can minimize the function f with lower complexity than if $\mathcal{M}$ was used directly on f, both in theory and in practice, as we will now show.

2.3.1 Catalyst

Catalyst is a meta-algorithm that provides Nesterov's acceleration to a given first-order method $\mathcal{M}$. We assume that the objective f is either L-smooth, or has a composite structure, which $\mathcal{M}$ can handle. It was inspired both by Güler (1992) and by the accelerated stochastic dual coordinate ascent algorithm of Shalev-Shwartz and Zhang (2016).

Relation between Moreau-Yosida and the proximal point algorithm. Consider the classical gradient descent algorithm with step size $1/\kappa$ to minimize F:

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \frac{1}{\kappa}\nabla F(\mathbf{x}_{t-1}).$$

By rewriting the gradient $\nabla F(\mathbf{x}_{t-1})$ as $\kappa(\mathbf{x}_{t-1} - \mathrm{Prox}_{f/\kappa}(\mathbf{x}_{t-1}))$, we obtain

$$\mathbf{x}_t = \mathrm{Prox}_{f/\kappa}[\mathbf{x}_{t-1}] = \operatorname*{arg\,min}_{z\in\mathbb{R}^p}\left\{ f(\mathbf{z}) + \frac{\kappa}{2}\|\mathbf{z} - \mathbf{x}_{t-1}\|_2^2 \right\}. \tag{2.16}$$

This is exactly the proximal point algorithm (Rockafellar, 1976).

Acceleration with Catalyst. Since gradient descent on F yields the proximal point algorithm, it is also natural to use an accelerated first-order method to get faster convergence. To that effect, Nesterov's algorithm (Nesterov, 1983) uses the two-stage update:

$$\mathbf{x}_t = \mathbf{y}_{t-1} - \frac{1}{\kappa}\nabla F(\mathbf{y}_{t-1}) \quad \text{and} \quad \mathbf{y}_t = \mathbf{x}_t + \beta_t(\mathbf{x}_t - \mathbf{x}_{t-1}),$$

where β_t is a specific extrapolation parameter (see Nesterov, 2004). We may now rewrite the update using the value of ∇F given in (2.15), which gives:

$$\mathbf{x}_t = \mathrm{Prox}_{f/\kappa}[\mathbf{y}_{t-1}] \quad \text{and} \quad \mathbf{y}_t = \mathbf{x}_t + \beta_t(\mathbf{x}_t - \mathbf{x}_{t-1}). \tag{2.17}$$

This is known as the accelerated proximal point algorithm introduced by Güler (1992), which has been studied with inexact gradients—equivalently, approximate values of the proximal operator $\mathrm{Prox}_{f/\kappa}[\mathbf{y}_{t-1}]$—several times (Güler, 1992; Salzo and Villa, 2012; Devolder et al., 2014). The Catalyst algorithm developed in (Lin et al., 2015) is an extension of this approach with a practical inexactness criterion and a global complexity analysis that provides the right parameter κ and sequence $(\varepsilon_t)_{t\geq 1}$, leading to acceleration.

Computing the inexact gradient. Specifically, we compute an approximate solution at iteration t:

$$\mathbf{z}_t \approx \arg\min_{\mathbf{z}\in\mathbb{R}^p} \Big\{ G_t(\mathbf{z}) := f(\mathbf{z}) + \frac{\kappa}{2}\|\mathbf{z} - \mathbf{y}_{t-1}\|_2^2 \Big\}, \tag{2.18}$$

with $G_t(\mathbf{z}_t) - G_t^\star \leq \varepsilon_t$, where $G_t^\star = F(\mathbf{y}_{t-1})$ is the minimum value of the function G_t and ε_t is the accuracy parameter at iteration t. Then, the update (2.17) of $\mathbf{x}_t$ is simply replaced by $\mathbf{x}_t = \mathbf{z}_t$. Equivalently, it corresponds to performing the inexact gradient descent step $\mathbf{x}_t = \mathbf{y}_{t-1} - \frac{1}{\kappa}\mathbf{g}_t$, with the inexact gradient $\mathbf{g}_t = \kappa(\mathbf{y}_{t-1} - \mathbf{z}_t)$.

Complexity analysis. We now summarize the main results, which rely on the assumption that the method $\mathcal{M}$ has a linear convergence rate for solving (2.18). Linear convergence is a typical property of many gradient-based approaches for minimizing smooth or composite strongly-convex functions and the results also hold for randomized algorithms when the convergence rate of $\mathcal{M}$ is given in expectation. Then, we consider two cases:

- **μ-strongly convex objective**: by choosing a sequence $(\varepsilon_t)_{t\geq 1} = O((1-\rho)^t)$ with $\rho < \sqrt{\mu/(\mu+\kappa)}$, we show that $f(\mathbf{x}_t) - f^\star$ converges at the same linear rate (Lin et al., 2015, Theorem 3.1). Furthermore, with an appropriate restart strategy, $\mathbf{z}_t$ in (2.18) can be obtained in $\tilde{O}(Q_{\mathcal{M}}(\kappa))$ iterations of $\mathcal{M}$ (Lin et al., 2015, Proposition 3.2), where the constant $Q_{\mathcal{M}}(\kappa)$ depends on κ;[6] indeed, κ changes the condition number of G_t: the larger κ, the easier it is to solve the sub-problems. From these observations, we can derive the global iteration complexity of the method:

  $$\tilde{O}\left(\frac{Q_{\mathcal{M}}(\kappa)\sqrt{\mu+\kappa}}{\sqrt{\mu}} \log\Big(\frac{1}{\varepsilon}\Big) \right). \tag{2.19}$$

  When $\mathcal{M}$ is the gradient descent method or ISTA, we can show that $Q_{\mathcal{M}}(\kappa) = (L+\kappa)/(\mu+\kappa)$; minimizing (2.19) yields $\kappa = L - 2\mu$, and the complexity becomes $\tilde{O}(\sqrt{L/\mu}\log(1/\varepsilon))$—that is, near the optimal rate of Nesterov (1983).
- **convex but not strongly convex objective**: With a sequence ε_t decreasing with a rate slightly faster than $O(1/t^4)$, the sequence $f(\mathbf{x}_t) - f^\star$ converges with the optimal rate $O(1/t^2)$ (Lin et al., 2015, Theorem 3.3). After taking into account the cost of solving the sub-problems (2.18), we pay a logarithmic price in the global complexity analysis (Lin et al., 2015, Proposition 3.4).

To illustrate these results, we present in Table 2.2 various complexities when f is an average of n convex L-smooth functions with a possible composite regularization. We call L_g the global Lipschitz constant of the objective, which is smaller than L (individual Lipschitz constants, see discussion in the paragraph "Optimization for large finite sums" in the introduction of this chapter).

2.3.2 QuickeNing

We remark that our work on Catalyst (Lin et al., 2015) did not originally mention the Moreau-Yosida regularization and presented Catalyst from the proximal point algorithm

6. The $\tilde{O}(.)$ notation hides logarithmic terms.

	Comp. $\mu > 0$	Comp. $\mu = 0$	Catalyst $\mu > 0$	Catalyst $\mu = 0$
FG	$O\big(n\big(\frac{L_g}{\mu}\big)\log\big(\frac{1}{\varepsilon}\big)\big)$	$O\big(n\frac{L_g}{\varepsilon}\big)$	$\tilde{O}\big(n\sqrt{\frac{L_g}{\mu}}\log\big(\frac{1}{\varepsilon}\big)\big)$	$\tilde{O}\Big(n\sqrt{\frac{L_g}{\varepsilon}}\Big)$
SAG	$O\big(\frac{L}{\mu}\log\big(\frac{1}{\varepsilon}\big)\big)$	$O\big(n\frac{L}{\varepsilon}\big)$	$\tilde{O}\big(\sqrt{\frac{nL}{\mu}}\log\big(\frac{1}{\varepsilon}\big)\big)$	$\tilde{O}\big(\sqrt{\frac{nL}{\varepsilon}}\big)$
SAGA				
MISO-Prox		not avail.		
SDCA				
SVRG				
Acc-FG	$O\big(n\sqrt{\frac{L_g}{\mu}}\log\big(\frac{1}{\varepsilon}\big)\big)$	$O\Big(n\sqrt{\frac{L_g}{\varepsilon}}\Big)$	no acceleration	
Acc-SDCA	$\tilde{O}\big(\sqrt{\frac{nL}{\mu}}\log\big(\frac{1}{\varepsilon}\big)\big)$	not avail.		

Table 2.2 – Comparison of complexities before and after Catalyst for minimizing a composite sum of n functions. We only present the case where $n \leq L/\mu$ for $\mu > 0$. For all incremental algorithms, there is indeed no acceleration otherwise since their complexity is $O(n\log(1/\varepsilon))$, which is independent of the condition number and already optimal. Note that L_g denotes the global Lipschitz constant of the objective which is smaller than L (see main text).

point of view. We realized later that introducing Moreau-Yosida in the equation could open several doors. In particular, any method designed for smooth optimization could be applied to the smoothed objective F. Quasi-Newton was then a natural principle to consider. In particular, how to exploit the problem structure (finite sum, composite) is well known for first-order methods, but these methods are usually unable to exploit the local curvature of the objective. On the other hand, limited-memory Quasi-Newton approaches such as L-BFGS (Liu and Nocedal, 1989; Nocedal, 1980) are able to model curvature, and are one of the greatest practical success of smooth optimization, but it is not clear how to efficiently adapt them to composite and structured problems.

For instance, there have been several attempts to develop a proximal Quasi-Newton method (Byrd et al., 2015; Lee et al., 2012; Scheinberg and Tang, 2014; Yu et al., 2008), but they typically require computing many times the proximal operator of the composite regularization with respect to a variable metric, which may be as computationally demanding as solving the original problem. More related, L-BFGS is combined with SVRG for minimizing smooth finite sums by Gower et al. (2016). The goal of QuickeNing [7] is more general since it is not limited to SVRG, but, like Catalyst, it can be applied to a large-class of first-order techniques for composite optimization.

The method. QuickeNing applies to the same objectives as Catalyst—that is L-smooth or composite functions; it relies on L-BFGS on F and uses a similar mechanism to obtain inexact gradients—that is, it requires solving sub-problems such as (2.18) with a first-order method $\mathcal{M}$ that is able to exploit the problem structure. The first difficulty consists of dealing with inexact gradients in L-BFGS, which is possible by adding skipping steps in the approximate Hessian updates (Friedlander and Schmidt, 2012). Then,

7. Note that the letters "Q" and "N" in QuickeNing stand for Quasi-Newton.

we also introduce a specific restart strategy that does not require using a line search algorithm, which is sometimes computationally expensive—especially in the context of Moreau-Yosida, since evaluating $F(\mathbf{x})$ requires solving a sub-problem. Then, we show that it admits a worst-case linear convergence rate for strongly convex problems, which is typically the best guarantees obtained for L-BFGS schemes in the literature.

The idea of combining second-order or quasi-Newton methods with Moreau-Yosida regularization is in fact relatively old. It may be traced back to variable metric proximal bundle methods (Chen and Fukushima, 1999; Fukushima and Qi, 1996; Mifflin, 1996). QuickeNing revisits this principle with a limited-memory variant (to deal with large dimension p), with an alternative strategy to line search schemes—which is useful when f is a large sum of n functions—and with *a global complexity analysis* that is more relevant than convergence rates that do not take into account the cost per iteration (see Lin et al., 2017, Section 4). By being able to exploit both the problem structure and local curvature, we show experimentally in (Lin et al., 2017, Section 5) that significant speed-up can be obtained over Catalyst for solving difficult ill-conditioned optimization problems.

Chapter 3

Towards Deep Kernel Machines

This chapter is in large parts based on the NIPS'16 publication:

J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

This work is a follow-up on the NIPS'14 paper below, which introduced a first proof of concept for building hierarchical kernels for images. The new model from the NIPS'16 paper corrects conceptual drawbacks of the earlier one. Its performance for unsupervised learning is significantly better, and it is also compatible with supervised learning.

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

In this thesis, we also briefly mention applications to image retrieval, corresponding to the following two publications, which use the model from the NIPS'14 paper. These applications are not detailed in this chapter.

M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid. Local convolutional features with unsupervised training for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, and C. Schmid. Convolutional patch representations for image retrieval: an unsupervised approach. *International Journal of Computer Vision (IJCV)*, 2016.

In the past years, deep neural networks such as convolutional or recurrent ones have become highly popular for solving various prediction problems, notably in computer vision and natural language processing. Conceptually close to approaches that were developed several decades ago (see LeCun et al., 1998), they greatly benefit from the large amounts of labeled data that have been available recently, allowing to learn huge numbers of model parameters without worrying too much about overfitting. Among other reasons explaining their success, the engineering effort of the deep learning community and various methodological improvements have made it possible to learn in a day on a GPU complex models that would have required weeks of computations on a traditional CPU (see Ciresan et al., 2012; Krizhevsky et al., 2012; Simonyan and Zisserman, 2015).

Before the resurgence of neural networks, non-parametric models based on positive definite kernels were one of the most dominant topics in machine learning (Schölkopf and Smola, 2002). These approaches are still widely used today because of several attractive features. Kernel methods are indeed versatile; as long as a positive definite kernel is specified for the type of data considered—*e.g.*, vectors, sequences, graphs, or sets—a large class of machine learning algorithms originally defined for linear models may be used. These include supervised formulations such as support vector machines and unsupervised ones such as principal or canonical component analysis, or K-means and spectral clustering. The problem of data representation is thus decoupled from that of learning theory and algorithms. Kernel methods also admit natural mechanisms to control the learning capacity and reduce overfitting (see Schölkopf and Smola, 2002).

On the other hand, traditional kernel methods suffer from several drawbacks. The first one is their computational complexity, which grows quadratically with the sample size due to the computation of the Gram matrix. Fortunately, significant progress has been achieved to solve the scalability issue, either by exploiting low-rank approximations of the kernel matrix (Smola and Schölkopf, 2000; Williams and Seeger, 2001; Zhang and Kwok, 2010), or with random sampling techniques for shift-invariant kernels (Rahimi and Recht, 2007; Le et al., 2013). The second disadvantage is more critical; by decoupling learning and data representation, kernel methods seem by nature incompatible with end-to-end learning—that is, the representation of data adapted to the task at hand, which is the cornerstone of deep neural networks and one of the main reason of their success. The main objective of this chapter is precisely to tackle this issue for image data.

Before introducing the main model, called convolutional kernel networks (Mairal et al., 2014c; Mairal, 2016), we will briefly introduce some methodological foundations. Note that the notation in this chapter significantly differs from the previous one since it targets a machine learning audience instead of the optimization community.

Regularization and empirical risk minimization. Let us consider a prediction task with n training data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ in some input space $\mathcal{X}$, which are associated, one-by-one, to observed scalars $y_1, y_2, \ldots, y_n$ in $\mathbb{R}$. The goal is to learn from the training data a prediction function $f : \mathcal{X} \to \mathbb{R}$ that is able to map a new point $\mathbf{x}$ in $\mathcal{X}$ onto the correct label y in $\mathbb{R}$. Typically, the problem is formulated as the minimization

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) + \lambda \Omega(f), \tag{3.1}$$

where $\mathcal{F}$ is a functional space on which the optimization is performed, and Ω is a function operating on $\mathcal{F}$. As we have seen in the previous chapter in a finite-dimensional setting, the first part of the objective, called "empirical risk", involves a loss function $\ell : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ that measures how far the prediction $f(\mathbf{x}_i)$ for the i-th data point is to the corresponding label y_i. The second part Ω, called "regularization", encodes a priori knowledge about the problem solution; when it has appropriate properties (see below for kernel methods), it may for instance encourage solutions f with slow variations, hence smooth ones, or solutions with a particular structure such as sparsity (see Chapter 4).

To turn (3.1) into a concrete formulation, it is necessary to choose a functional space $\mathcal{F}$ along with a regularization Ω. The simplest setting is the set of linear functions when $\mathcal{X}$ is the Euclidean space $\mathbb{R}^p$. In that case, $\mathcal{F} := \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^p\}$ with $f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{x}^\top\mathbf{w}$ and the regularization is the squared norm: $\Omega(f_{\mathbf{w}}) = \|\mathbf{w}\|_2^2$ for all $\mathbf{w}$ in $\mathbb{R}^p$. By choosing different loss functions ℓ, one recovers classical machine learning formulations such as linear support vector machines or logistic regression (Vapnik, 2000; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004). In Chapter 2, we addressed such formulations from an optimization point of view. Going beyond such simple linear models on Euclidean spaces brings us to two successful paradigms: kernel methods and deep learning.

Kernel methods. When $\mathcal{X}$ is not a Euclidean space or when the functions in $\mathcal{F}$ are non-linear, kernel methods offer a framework where (3.1) is turned into an optimization problem in $\mathbb{R}^n$. The main idea is to exploit a positive definite kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is a symmetric function such that all combinations $\sum_{i,j=1}^n \alpha_i\alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$ are non-negative for all positive integer n, points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathcal{X}$, and scalars α_i in $\mathbb{R}$. In other words, the symmetric $n \times n$ Gram matrix defined as $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ has non-negative eigenvalues. A classical result states that a positive definite kernel implicitly defines a Hilbert space $\mathcal{H}$ of functions from $\mathcal{X}$ to $\mathbb{R}$, called reproducing kernel Hilbert space (RKHS), along with a mapping $\varphi : \mathcal{X} \to \mathcal{H}$. Moreover, this embedding is such that the kernel value $K(\mathbf{x}, \mathbf{x}')$ and the function evaluation $f(\mathbf{x})$ are respectively the inner products $\langle\varphi(\mathbf{x}), \varphi(\mathbf{x}')\rangle_{\mathcal{H}}$ and $\langle f, \varphi(\mathbf{x})\rangle_{\mathcal{H}}$ for all $\mathbf{x}, \mathbf{x}'$ in $\mathcal{X}$ and f in $\mathcal{H}$. RKHSs have been well studied in functional and harmonic analysis, and in machine learning (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002). Specifically, it is possible to choose the space $\mathcal{F} = \mathcal{H}$ associated to a positive definite kernel K, and then

- the norm $\|.\|_{\mathcal{H}}$ is a natural regularization function that controls the variations over $\mathcal{X}$ of all f in $\mathcal{H}$ according to the geometry induced by the kernel. For all points $\mathbf{x}, \mathbf{x}'$ in $\mathcal{X}$, we indeed have the following Lipschitz inequality $|f(\mathbf{x}) - f(\mathbf{x}')| \leq \|f\|_{\mathcal{H}}\|\varphi(\mathbf{x}) - \varphi(\mathbf{x}')\|_{\mathcal{H}}$, where the term $\|\varphi(\mathbf{x}) - \varphi(\mathbf{x}')\|_{\mathcal{H}}$ defines a distance between the points $\mathbf{x}$ and $\mathbf{x}'$ from $\mathcal{X}$. Hence, a small norm $\|f\|_{\mathcal{H}}$ implies small variations.
- the solution $f^\star$ of (3.1) lies in a finite-dimensional subspace of dimension at most n, and can be obtained by solving an optimization problem in $\mathbb{R}^n$. In fact, there exists a set of scalars $\alpha_1, \ldots, \alpha_n$ such that $f^\star = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, .)$.

Apparently, kernel methods enjoy several properties: it is possible to work with structured sets $\mathcal{X}$, and the data representations $\varphi(\mathbf{x})$ may be in a space of infinite dimension $\mathcal{H}$. These assets are unfortunately mitigated by a lack of scalability to big data problems.

The complexity of kernel methods is typically quadratic $O(n^2)$ since, in general, they require computing the $n \times n$ kernel matrix, which is intractable when n is large.

To cope with this issue, the most classical strategy is to approximate the non-linear kernel K by a linear one in a Euclidean space. More precisely, a linear kernel is the simplest example of positive definite kernel when $\mathcal{X} = \mathbb{R}^p$. It is defined for all $\mathbf{x}, \mathbf{x}'$ in $\mathbb{R}^p$ as the inner product $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$, and the corresponding RKHS $\mathcal{H}$ is simply the space of linear functions on $\mathbb{R}^p$. Then, the approximation scheme for a nonlinear kernel K consists in looking for a mapping $\psi : \mathcal{X} \to \mathbb{R}^p$ such that

$$K(\mathbf{x}, \mathbf{x}') \approx \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle, \quad \text{for all } \mathbf{x} \text{ and } \mathbf{x}' \text{ in } \mathcal{X}. \tag{3.2}$$

Assuming that we are given a good explicit mapping ψ, the formulation (3.1) with regularization $\Omega = \|.\|_{\mathcal{H}}^2$ may be approximated by

$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell\Big(y_i, \mathbf{w}^\top \psi(\mathbf{x}_i)\Big) + \lambda \|\mathbf{w}\|_2^2, \tag{3.3}$$

which we now how to minimize efficiently (see Chapter 2). Among the different strategies to find ψ, the Nyström method (Smola and Schölkopf, 2000; Fine and Scheinberg, 2001; Williams and Seeger, 2001) consists of projecting the data points $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_n)$ onto a subspace of $\mathcal{H}$ with finite dimension p. Such a strategy is able to leverage the intrinsic low dimensionality of the data in $\mathcal{H}$, but finding efficiently the optimal subspace for large-scale problems and for a specific task is still an open issue in general.

Deep learning. A significantly different approach to kernels consists of defining $\mathcal{F}$ as a set of parametrized nonlinear functions $\mathcal{F} := \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^p\}$ and optimizing (3.1) directly over the parameters $\mathbf{w}$ in $\mathbb{R}^p$. This is the strategy chosen in deep neural networks, where the functions $f_{\mathbf{w}}$ perform successive "simple" non-linear operations organized in several layers—say k. More precisely, each layer performs a linear operation followed by a pointwise non-linearity. Formally, it is possible to write $f_{\mathbf{w}}(\mathbf{x})$ for some vector $\mathbf{x}$ as

$$f_{\mathbf{w}}(\mathbf{x}) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \ldots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 \mathbf{x})) \ldots)), \tag{3.4}$$

where the quantities $\mathbf{A}_i$ are matrices corresponding to linear operations and the σ_i's are pointwise non-linear functions, *e.g.*, sigmoids or rectified linear units. Then, the entries of the matrices $\mathbf{A}_i$ are learned (*i.e.*, they form the entries of $\mathbf{w}$). Solving (3.1) is however difficult because of the non-convexity of the objective and the potential huge number of parameters p. Yet, approximate solutions returned by stochastic gradient descent algorithms have shown empirically to provide outstanding prediction performance for various tasks. One reason for this success seem to be the good quality of local minima of the objective when the number of parameters is huge (Choromanska et al., 2015).

In the context of images, the most popular models are probably convolutional neural networks (CNNs) (LeCun et al., 1998), whose architecture is a variant of (3.4) adapted to two-dimensional images; linear operations are constrained to be local convolutions, and some layers perform a downsampling operation called "feature pooling". As a result,

CNNs are able to model the local stationarity in natural images at multiple scales, and seem to be able to learn how to combine low-level features into high-level interpretable ones (Zeiler and Fergus, 2014). Other successful deep learning approaches are recurrent neural networks (Williams and Zipser, 1989; Hochreiter and Schmidhuber, 1997), which fall beyond the scope of this chapter.

3.1 Basic principles of deep kernel machines

In the rest of the chapter, we will focus on deep kernel machines, which combine principles from kernel methods and multilayer neural networks. Below, we review some of their principles, before introducing convolutional kernel networks in the next section. [1]

Composition of feature spaces. Consider a positive definite (p.d.) kernel K_1 and its RKHS mapping $\varphi_1 : \mathcal{X} \to \mathcal{H}_1$. Consider also $K_2 : \mathcal{H}_1 \times \mathcal{H}_1 \to \mathbb{R}$ another p.d. kernel, with mapping $\varphi_2 : \mathcal{H}_1 \to \mathcal{H}_2$; then, it is well known that both kernels can be combined into a new one (see Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002), as suggested for instance by Schölkopf et al. (1998) for a multilayer variant of their kernel principal component analysis formulation. The composition of kernels may be written as

$$K_3(\mathbf{x}, \mathbf{x}') = K_2(\varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}')) \quad \text{and its RKHS mapping is} \quad \varphi_3 = \varphi_2 \circ \varphi_1, \tag{3.5}$$

where $\circ$ represents the composition operator. For example, one may consider the Gaussian or radial basis function (RBF) kernel operating on $\mathcal{H}_1$

$$K_2(\varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}')) = e^{-\frac{1}{2\sigma^2}\|\varphi_1(\mathbf{x}) - \varphi_1(\mathbf{x}')\|^2_{\mathcal{H}_1}}, \tag{3.6}$$

or the polynomial one

$$K_2(\varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}')) = \langle \varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}') \rangle^2_{\mathcal{H}_1} = K_1(\mathbf{x}, \mathbf{x}')^2. \tag{3.7}$$

It is then possible to iterate several times this construction to build multilayer kernels. Yet, as discussed by Cho and Saul (2010), it is not clear that such a multilayer mechanism is useful, especially when "simple" kernels—*e.g.*, RBF or polynomial ones—are involved. For instance, by composing several polynomial kernels, one only gets another polynomial kernel of higher degree. More generally, by composing several dot-product kernels of the form $K_l(\mathbf{x}, \mathbf{x}') = \kappa_l(\langle \mathbf{x}, \mathbf{x}' \rangle)$, with $l = 1, \ldots, m$, one will simply get another dot-product kernel $K(\mathbf{x}, \mathbf{x}') = (\kappa_m \circ \kappa_{m-1} \circ \ldots \circ \kappa_1)(\langle \mathbf{x}, \mathbf{x}' \rangle)$. Similarly, composing the RBF kernel (3.6) with the polynomial one (3.7) yields another RBF kernel with a bandwidth reduced by $\sqrt{2}$.

The work we present in this chapter does not contradict the fact that the basic multilayer kernels (3.5) may have limited practical benefits. The kernel K_3 in (3.5) operates indeed on the same type of object as K_1, and there is no clear reason why a kernel build

1. Note that the terminology "deep kernel machine" seems to be originally due to Yger et al. (2011), who have proposed an extension of the multilayer kernel machine of Cho and Saul (2010).

by composition may be more appropriate for learning than another one. Instead, we will show later that the principle of feature space composition, when used in a slightly different manner than (3.5), may be useful for building a sequence of kernels that operate on more and more complex objects. In the context of convolutional kernel networks that we will introduce, it means building a sequence of kernels that operate on larger and larger image neighborhoods (aka, receptive fields), with more and more invariance properties.

Note also that feature space composition was also used recently by Steinwart et al. (2016) to build hierarchical Gaussian kernels; the kernels at each layer are defined as a weighted sums of Gaussian kernels that operate on subsets of the previous layer's RKHS. Thus, the composition principle is used in more complicated manner than in (3.5) and leads to non-trivial and potentially interesting feature spaces.

Kernels and shallow infinite random neural nets. Besides the composition of feature spaces, explicit links can be drawn between kernels and neural networks with one layer and an infinite number of neurons with random weights. Such a relation was discovered by Neal (1994) in the context of Gaussian processes, before being exploited and generalized by Le Roux and Bengio (2007); Cho and Saul (2010) and Bach (2017).

The main observation of interest for us is that many positive definite kernels $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ admit an integral form involving a linear operation on each entry $\mathbf{x}$ and $\mathbf{x}'$ followed by a pointwise non-linearity, denoted below by s:

$$K(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w}}\Big[s\Big(\mathbf{w}^\top \mathbf{x}\Big) s\Big(\mathbf{w}^\top \mathbf{x}'\Big)\Big], \tag{3.8}$$

where $\mathbf{w}$ in $\mathbb{R}^m$ is a random vector. Then, each random realization $s(\mathbf{w}^\top \mathbf{x})$ can be interpreted as the output of a neuron with random weights $\mathbf{w}$. For instance, for any vectors $\mathbf{x}$ and $\mathbf{x}'$ on the sphere $\mathbb{S}^{m-1}$, it is relatively simple to show that the RBF kernel is equal to

$$e^{-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}'\|_2^2} \propto \mathbb{E}_{\mathbf{w}}\Big[e^{\frac{2}{\sigma^2}\mathbf{w}^\top \mathbf{x}} e^{\frac{2}{\sigma^2}\mathbf{w}^\top \mathbf{x}'}\Big] \quad \text{with} \quad \mathbf{w} \sim \mathcal{N}(0, (\sigma^2/4)\mathbf{I}).$$

Interestingly, the relation (3.8) can also be analyzed when using more classical non-linearities of neural networks. For example, rectified linear units have been shown to yield a closed-form for (3.8) called arc-cosine kernel (see Le Roux and Bengio, 2007; Cho and Saul, 2010; Bach, 2017). Specifically, let us consider

$$K_\alpha(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w}}\Big[\max\Big(\mathbf{w}^\top \mathbf{x}, 0\Big)^\alpha \max\Big(\mathbf{w}^\top \mathbf{x}', 0\Big)^\alpha\Big] \quad \text{with} \quad \mathbf{w} \sim \mathcal{N}(0, \mathbf{I}),$$

where α is a non-negative integer. Then, K_α admits a closed form that depends on the angle between $\mathbf{x}, \mathbf{x}'$, hence the name "arc-cosine kernel". When $\mathbf{x}$ and $\mathbf{x}'$ is on the sphere, we obtain a particular dot-product kernel. Finally, Rahimi and Recht (2007) have proposed an approximation scheme for shift-invariant kernels, based on the relation

$$\kappa(\mathbf{x} - \mathbf{x}') = \mathbb{E}_{\mathbf{w},b}\Big[\cos\Big(\mathbf{w}^\top \mathbf{x} + b\Big) \cos\Big(\mathbf{w}^\top \mathbf{x}' + b\Big)\Big] \;\text{ with }\; b \sim \mathcal{U}[0, 2\pi] \text{ and } \mathbf{w} \sim \frac{1}{(2\pi)^m}\hat{\kappa}(\mathbf{w}),$$

where we assume, with the right assumptions such that the quantities above are well defined, that $\kappa(0) = 1$ and $\hat{\kappa}$ is the Fourier transform of κ. The principle of infinite neural

networks was then revisited to gain some insight about the function classes underlying deep neural networks (Hazan and Jaakkola, 2015; Daniely et al., 2016).

In (Mairal et al., 2014c), we also adopt this point of view to develop a data-driven approximation scheme for kernels of the form (3.8), for which a closed-form $K(\mathbf{x}, \mathbf{x}')$ is available; we applied this technique to the RBF kernel, which is a basic component of the convolutional kernel networks that will be presented shortly. Given a set of training points $\mathbf{x}_1, \dots, \mathbf{x}_n$ in $\mathbb{R}^m$, we consider the following non-convex objective

$$\min_{\mathbf{W} \in \mathbb{R}^{m \times p}, \boldsymbol{\eta} \in \mathbb{R}^p_+} \frac{1}{n^2} \sum_{i,j=1}^{n} \left(K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{l=1}^{p} \eta_l s\Big(\mathbf{w}_l^\top \mathbf{x}_i\Big) s\Big(\mathbf{w}_l^\top \mathbf{x}_j\Big) \right), \tag{3.9}$$

where $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_p]$ and $\boldsymbol{\eta} = [\eta_1, \dots, \eta_p]$. The formulation can be interpreted as an importance sampling strategy to approximate the expectation (3.8), where the samples $\mathbf{w}_l$ and weights η_l, $l = 1, \dots, p$ are not random, but learned with no supervision. In (Paulin et al., 2015, 2016), we use a stochastic gradient descent algorithm to approximately minimize (3.9). Then, the kernel $K(\mathbf{x}, \mathbf{x}')$ can be approximated by the mapping $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$, where $\psi(\mathbf{x}) = [\sqrt{\eta_l} s(\mathbf{w}_l^\top \mathbf{x})]_{l=1,\dots,p}$ can be interpreted as the output of a one-layer neural network with p neurons.

This link is appealing since it provides a new principle for learning a neural network in an unsupervised manner by approximating a positive definite kernel. Yet, a drawback of this approach is that the mappings $\psi(\mathbf{x}_1), \psi(\mathbf{x}_2), \dots$ are designed to approximate the inner-product $K(.,.)$, but they do not admit an interpretation in terms of points in the RKHS $\mathcal{H}$. Next, we will present another link between kernel methods and neural network that does not suffer from this conceptual issue, by using instead the Nyström method (Williams and Seeger, 2001). In the context of convolutional kernel networks, this is the approach that we have adopted in (Mairal, 2016).

Dot-product kernels and subspace learning in RKHSs. Consider a dot-product kernel $K(\mathbf{x}, \mathbf{x}') = \kappa(\langle \mathbf{x}, \mathbf{x}' \rangle) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}$, where $\mathbf{x}$ and $\mathbf{x}'$ are vectors in $\mathbb{R}^m$. The Nyström method consists of projecting the points $\varphi(\mathbf{x})$ in $\mathcal{H}$ onto a p-dimensional subspace $\mathcal{F}$ of $\mathcal{H}$, and then parametrizing the projected points in $\mathcal{F}$ by a vector of dimension p. Such a principle is general and can be used with any positive definite kernel, but it turns out that the approximation scheme can be interpreted as a neural network when applied to dot-product kernels on the sphere. It is usually known as the Nyström method (Williams and Seeger, 2001), but seems to have been proposed simultaneously several times from different point of views (Smola and Schölkopf, 2000; Fine and Scheinberg, 2001).

Specifically, the subspace $\mathcal{F}$ is parametrized by p anchor points

$$\mathcal{F} := \mathrm{Span}(\varphi(\mathbf{z}_1), \dots, \varphi(\mathbf{z}_p)),$$

where the $\mathbf{z}_i$'s are in $\mathbb{R}^m$ (since we consider a kernel operating on $\mathbb{R}^m$). The Euclidean projection on $\mathcal{F}$ of an input point $\mathbf{x}$ can be formulated as

$$f_{\mathbf{x}} := \arg\min_{f \in \mathcal{F}} \|\varphi(\mathbf{x}) - f\|_{\mathcal{H}}^2,$$

which is equivalent to

$$f_{\mathbf{x}} := \sum_{j=1}^{p} \alpha_j^\star \varphi(\mathbf{z}_j) \quad \text{with} \quad \boldsymbol{\alpha}^\star \in \operatorname*{arg\,min}_{\boldsymbol{\alpha} \in \mathbb{R}^p} \left\| \varphi(\mathbf{x}) - \sum_{j=1}^{p} \alpha_j \varphi(\mathbf{z}_j) \right\|_{\mathcal{H}}^2 .$$

After short classical calculations (see Schölkopf and Smola, 2002, Chapter 18, in the context of reduced set methods), we obtain

$$f_{\mathbf{x}} = \sum_{j=1}^{p} \alpha_j^\star \varphi(\mathbf{z}_j) \quad \text{with} \quad \boldsymbol{\alpha}^\star \in \operatorname*{arg\,min}_{\boldsymbol{\alpha} \in \mathbb{R}^p} \Big[\kappa\Big(\mathbf{x}^\top \mathbf{x}\Big) - 2\boldsymbol{\alpha}^\top \kappa\Big(\mathbf{Z}^\top \mathbf{x}\Big) + \boldsymbol{\alpha}^\top \kappa\Big(\mathbf{Z}^\top \mathbf{Z}\Big) \boldsymbol{\alpha} \Big],$$

where, with an abuse of notation, the function κ is applied pointwise to its arguments. Assuming $\mathbf{x}$ to be normalized in $\mathcal{H}$—that is, $\kappa(\mathbf{x}^\top \mathbf{x}) = 1$, and $\kappa(\mathbf{Z}^\top \mathbf{Z})$ is invertible, we are left with the unique solution $\boldsymbol{\alpha}^\star = \kappa(\mathbf{Z}^\top \mathbf{Z})^{-1} \kappa(\mathbf{Z}^\top \mathbf{x})$. Then, we define the mapping $\psi : \mathbb{R}^m \to \mathbb{R}^p$ such that $\langle f_{\mathbf{x}}, f_{\mathbf{x}'} \rangle_{\mathcal{H}} = \boldsymbol{\alpha}^{\star\top} \kappa(\mathbf{Z}^\top \mathbf{Z}) \boldsymbol{\alpha}'^\star = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$, defined as

$$\psi(\mathbf{x}) := \kappa\Big(\mathbf{Z}^\top \mathbf{Z}\Big)^{-1/2} \kappa\Big(\mathbf{Z}^\top \mathbf{x}\Big). \tag{3.10}$$

Here, this operation can be seen as a neural network since the set of operations performed by ψ is a sequence of a linear operation (multiplication by $\mathbf{Z}^\top$), pointwise non-linearity (kernel value κ), and again a linear operation (multiplication by $\kappa(\mathbf{Z}^\top \mathbf{Z})^{-1/2}$). This classical mapping provides a kernel approximation in the sense that

$$K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}} \approx \langle f_{\mathbf{x}}, f_{\mathbf{x}'} \rangle_{\mathcal{H}} = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle.$$

Besides, the vector $\psi(\mathbf{x})$ in $\mathbb{R}^p$ can be interpreted as coordinates in the linear subspace $\mathcal{F} \subseteq \mathcal{H}$. Therefore, we may define several mathematical operations in $\mathcal{H}$, which involve manipulating the finite-dimensional mappings $\psi(\mathbf{x}_1), \psi(\mathbf{x}_2), \ldots$ For example, linear combinations $\sum_{i=1}^n \beta_i \psi(\mathbf{x}_i)$ in $\mathbb{R}^p$ can be seen as coordinates of the point $\sum_{i=1}^n \beta_i f_{\mathbf{x}_i}$ in $\mathcal{F} \subseteq \mathcal{H}$. In the next section, we will use this principle to define valid pooling operations in $\mathcal{F}$ that do not break the kernel interpretation.

How to choose the anchor points $\mathbf{z}_1, \ldots, \mathbf{z}_p$—equivalently, learning the subspace $\mathcal{F}$ of $\mathcal{H}$—optimally is still an open question. Different approaches are selecting these points from training data, either randomly (Williams and Seeger, 2001), or in a greedy manner (Smola and Schölkopf, 2000; Fine and Scheinberg, 2001; Bach and Jordan, 2005). For vectorial data, which is the case we consider here, another approach consists of learning the anchor points $\mathbf{z}_1, \ldots, \mathbf{z}_p$ from training data. For that purpose, the clustered Nyström method (Zhang and Kwok, 2010) consists of assigning the vectors $\mathbf{z}_j$ to the centroids obtained by the K-means algorithm. This is the kernel approximation we have found the most effective in the context of convolutional kernel networks (see next section).

Other models. Multiple kernel learning (Sonnenburg et al., 2006) is also related to deep kernel machines since is it is a notable attempt to introduce supervision in the kernel design. It provides techniques to select a combination of kernels from a predefined collection, and typically requires to have already "good" kernels in the collection

to perform well. It was notably extended to learn a structured combinations of kernels by Bach (2008b) and Szafranski et al. (2010). Also related to deep kernel machines, the backpropagation algorithm for the Fisher kernel introduced by Sydorov et al. (2014) learns the parameters of a Gaussian mixture model with supervision. In comparison, the model we will introduce in the next section does not require a probabilistic model and learns parameters at several layers. Finally, we remark that a concurrent effort is conducted in the Bayesian community with deep Gaussian processes (Damianou and Lawrence, 2013), complementing the Frequentist approach that we follow in this thesis.

Recently, other techniques combining deep neural networks and kernels have been also introduced. For example, Montavon et al. (2011); Anselmi et al. (2015); Daniely et al. (2016) introduce different models based on kernels to gain some theoretical insight about classical multilayer neural networks, while Zhang et al. (2016) use kernels and RKHSs to convexify a specific class of two-layer neural networks. Kernels that enjoy invariance properties for visual recognition were also proposed by Smale et al. (2010). Such kernels are built with a parameterized "neural response" function, which consists in computing the maximal response of a base kernel over a local neighborhood. Multiple layers are then built by iteratively renormalizing the response kernels and pooling using neural response functions. Learning is performed by plugging the obtained kernel in a SVM.

3.2 Convolutional kernel networks

In this section, we now present deep kernel machines called convolutional kernel networks, which we have introduced in (Mairal et al., 2014c; Mairal, 2016). This approach is in fact a combination of three elements:

1. **a multilayer convolutional kernel for representing images.** Specifically, we build a hierarchy of positive definite kernels for local image neighborhoods (aka, receptive fields). When going up in the hierarchy, these kernels have more invariant properties and apply to larger and larger neighborhoods. Unfortunately, these kernels are computationally extremely demanding and cannot be used directly in a learning algorithm; hence, they require modifications and/or approximation.

2. **an unsupervised learning scheme.** Between layers, we use a kernel approximation technique to make the model scalable. In (Mairal et al., 2014c), we use the method presented in (3.9), which exploit the integral expansion of the RBF kernel. In (Mairal, 2016), we use the clustered Nyström approximation, presented in (3.10), which provides better empirical performance and faster training. Equivalently, it means learning finite-dimensional subspaces in the RKHSs where we project data.

3. **a supervised learning algorithm.** In (Mairal, 2016), we extend the model to handle labeled data and perform end-to-end learning. For that, we jointly optimize subspaces of the RKHSs at each layer by minimizing a supervised loss function.

We will now detail these three components.

3.2.1 The multilayer convolutional kernel for images

The convolutional multilayer kernel is a generalization of the hierarchical kernel descriptors introduced in computer vision by Bo et al. (2011), which produce a sequence of image representations that are built on top of each other. It also generalizes a single-layer kernel for images proposed by Schölkopf (1997) in his PhD thesis. Each layer can be interpreted as a non-linear transformation of the previous one with additional spatial invariance. We call these layers *image feature maps*[2], and formally define them as follows:

Definition 4. *An image feature map is a function $I : \Omega \to \mathcal{H}$, where Ω is a two-dimensional grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.*

Given $\boldsymbol{\omega}$ in Ω, the point $I(\boldsymbol{\omega})$ represents some characteristics of the image at location $\boldsymbol{\omega}$, or in a neighborhood of $\boldsymbol{\omega}$. For instance, a color image of size $m \times m$ with three channels, red, green, and blue, may be represented by a feature map $I_0 : \Omega_0 \to \mathcal{H}_0$, where Ω_0 is an $m \times m$ regular grid, $\mathcal{H}_0$ is the Euclidean space $\mathbb{R}^3$, and I_0 provides the RGB pixel values in $\mathbb{R}^3$. Next, we will show how to build a sequence of image feature maps, organized in a multilayer fashion, which encode more complex image characteristics.

Use the kernel trick to represent local image neighborhoods in a RKHS. Consider an initial image $I_0 : \Omega_0 \to \mathcal{H}_0$. Image patches of size $e_0 \times e_0$ can naturally be defined as elements of the Cartesian product $\mathcal{P}_0 := \mathcal{H}_0^{e_0 \times e_0}$ endowed with its natural inner-product. Given two of such patches $\mathbf{x}, \mathbf{x}'$ in $\mathcal{P}_0$, we may define several types of kernels that only depend on inner-products in $\mathcal{P}_0$. For instance, this is the case of shift-invariant kernels of the form

$$K_1(\mathbf{x}, \mathbf{x}') = \kappa_1\Big(\|\mathbf{x} - \mathbf{x}'\|_{\mathcal{P}_0}^2\Big), \tag{3.11}$$

or a homogeneous variant of a dot-product kernel, which we have found useful in practice:

$$K_1(\mathbf{x}, \mathbf{x}') = \|\mathbf{x}\|_{\mathcal{P}_0} \|\mathbf{x}'\|_{\mathcal{P}_0} \kappa_1\left(\frac{\langle \mathbf{x}, \mathbf{x}' \rangle_{\mathcal{P}_0}}{\|\mathbf{x}\|_{\mathcal{P}_0} \|\mathbf{x}'\|_{\mathcal{P}_0}}\right) \text{ if } \mathbf{x}, \mathbf{x}' \neq 0 \quad \text{and 0 otherwise.} \tag{3.12}$$

Note that κ_1 should be smooth and its Taylor expansion coefficients should be non-negative to ensure positive definiteness of (3.12), see Schölkopf and Smola (2002). For example, the arc-cosine kernel of Cho and Saul (2010), various polynomial kernels, or the Gaussian (RBF) kernels may be used: given two vectors $\mathbf{y}, \mathbf{y}'$ on the sphere of $\mathcal{P}_0$, choose indeed

$$\kappa_1(\langle \mathbf{y}, \mathbf{y}' \rangle_{\mathcal{P}_0}) = e^{\alpha_1\left(\langle \mathbf{y}, \mathbf{y}' \rangle_{\mathcal{P}_0} - 1\right)} = e^{-\frac{\alpha_1}{2}\|\mathbf{y} - \mathbf{y}'\|_{\mathcal{P}_0}^2}. \tag{3.13}$$

Then, we have implicitly defined the RKHS $\mathcal{H}_1$ associated to K_1 and a mapping $\varphi_1 : \mathcal{P}_0 \to \mathcal{H}_1$. Subsequently, this also allow us to build an image feature map $I_{0.5} : \Omega_0 \to \mathcal{H}_1$ such that $I_{0.5}(\boldsymbol{\omega})$ is the mapping $\varphi_1(P_\omega)$ of the patch P_ω of I_0 centered at location $\boldsymbol{\omega}$.[3] This

2. In the kernel literature, "feature map" denotes the mapping between data points and their representation in a reproducing kernel Hilbert space (RKHS). Here, feature maps refer to spatial maps representing local image characteristics at every location, as usual in the neural network literature.

3. To simplify, we use zero-padding when patches are close to the image boundaries, but this is an optional step.

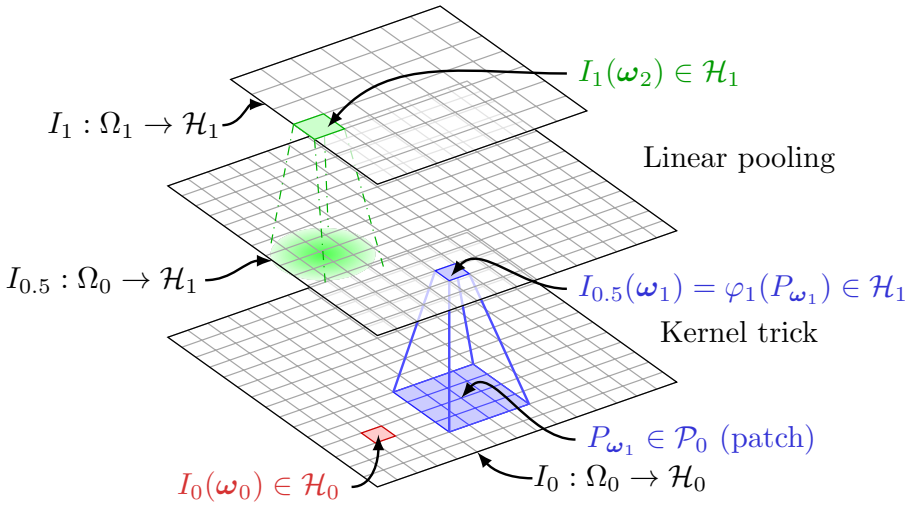


Figure 3.1 – Illustration of the image feature map construction. The intermediate map $I_{0.5}$ is obtained by mapping every patch from I_0 to the RKHS $\mathcal{H}_1$. Then, the resolution is reduced by linear pooling to produce I_1. The multilayer construction consists of repeating the above mechanism many times, in order to produce the maps $I_0, I_1, \ldots, I_k$. This construction is abstract in the sense that the quantities represented in the figure have possibly infinite dimension. The concrete approximation is presented in Figure 3.2.

mechanism is illustrated at the bottom of Figure 3.1. Next, we show how to downsample the map $I_{0.5}$ to build the map I_1 with smaller resolution.

Downsampling the maps with linear pooling in $\mathcal{H}_1$. The previous step transforms an image $I_0 : \Omega_0 \to \mathcal{H}_0$ into a map $I_{0.5} : \Omega_0 \to \mathcal{H}_1$, where each point $I_{0.5}(\boldsymbol{\omega})$ carries information of a local image neighborhood centered at location $\boldsymbol{\omega}$ in Ω_0. Then, the multilayer kernel construction involves a pooling step to gain invariance to small shifts, leading to the map $I_1 : \Omega_1 \to \mathcal{H}_1$ with smaller resolution:

$$I_1(\boldsymbol{\omega}) = \sum_{\boldsymbol{\omega}' \in \Omega_0} I_{0.5}(\boldsymbol{\omega}') e^{-\beta_1 \|\boldsymbol{\omega}' - \boldsymbol{\omega}\|_2^2}. \tag{3.14}$$

The Gaussian weights can be interpreted as an anti-aliasing filter for downsampling the map $I_{0.5}$ and β_1 is set according to the desired subsampling factor (see Mairal et al., 2014c), which does not need to be integer. Note that the linear pooling step was originally motivated in (Mairal et al., 2014c) as an approximation scheme for a match kernel that compares two maps $I_{0.5}$ and $I'_{0.5}$:

$$\begin{aligned}\sum_{\boldsymbol{\omega},\boldsymbol{\omega}' \in \Omega_0} I_{0.5}(\boldsymbol{\omega}) I'_{0.5}(\boldsymbol{\omega}') e^{-\frac{\beta_1}{2}\|\boldsymbol{\omega}' - \boldsymbol{\omega}\|_2^2} &\propto \sum_{\boldsymbol{\omega},\boldsymbol{\omega}' \in \Omega_0} \langle I_{0.5}(\boldsymbol{\omega}) I'_{0.5}(\boldsymbol{\omega}') \rangle_{\mathcal{H}_1} \int_{\boldsymbol{\omega}'' \in \mathbb{R}^2} e^{-\beta_1 \|\boldsymbol{\omega}' - \boldsymbol{\omega}''\|_2^2} e^{-\beta_1 \|\boldsymbol{\omega} - \boldsymbol{\omega}''\|_2^2} d\boldsymbol{\omega}'' \\ &= \int_{\boldsymbol{\omega}'' \in \mathbb{R}^2} \langle I_1(\boldsymbol{\omega}''), I'_1(\boldsymbol{\omega}'') \rangle_{\mathcal{H}_1} d\boldsymbol{\omega}'',\end{aligned} \tag{3.15}$$

where, with an abuse of notation, the map I_1 from (3.14) is defined on $\mathbb{R}^2$ instead of Ω_1. The point of view of the match kernel (3.15) is not critical to describe convolutional kernel networks, but it provides a useful interpretation. The kernel (3.15) consists of a sum of pairwise comparisons between image features $I_{0.5}(\boldsymbol{\omega})$ and $I'_{0.5}(\boldsymbol{\omega}')$ computed at all spatial locations. To be significant in the sum, a comparison needs the corresponding $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$ to be close in Ω_0, and the local image features to be close in the feature space $\mathcal{H}_0$. The parameters β_1 and α_1 respectively control these two definitions of "closeness". Indeed, when β_1 is large, the match kernel is invariant to the positions $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$ but when β_1 is small, only features placed at the same location $\boldsymbol{\omega} = \boldsymbol{\omega}'$ are compared to each other. Therefore, β_1 controls by how much the kernel is locally shift-invariant.

Stack the compositions to build the multilayer convolutional kernel. We may now iterate the previous construction to build a sequence of maps $I_0, I_1, \ldots, I_k$. At layer l, we define a kernel K_l that operates on patches from the previous layer—that is, on local image neighborhoods. When going up in the hierarchy, the size of these receptive fields increases and the kernel gains some invariance. Finally, we define a kernel on two images I_0, I'_0 represented by the maps I_k, I'_k as follows

$$K(I_0, I'_0) = \sum_{\boldsymbol{\omega} \in \Omega_k} \langle I_k(\boldsymbol{\omega}), I'_k(\boldsymbol{\omega}) \rangle_{\mathcal{H}_k}, \tag{3.16}$$

which we call multilayer convolutional kernel. Due to its high computational complexity, it requires approximations to be used, which is the purpose of the next sections.

3.2.2 Unsupervised subspace learning in RKHSs

To make the multilayer convolutional kernel model tractable, we use the Nyström approximation at each layer. This amounts to adding projection steps on parametrized finite-dimensional subspaces of the RKHS's layers. Concretely, we want to manipulate finite-dimensional image feature maps instead of abstract infinite-dimensional ones. We illustrate this mechanism in Figure 3.2.

Project onto finite-dimensional subspaces of the RKHSs Assuming that we initially have such a finite-dimensional map $M_0 = I_0 : \Omega_0 \to \mathbb{R}^{p_0}$ (meaning $\mathcal{H}_0 = \mathbb{R}^{p_0}$), we map a $e_0 \times e_0$ patch of M_0 to the RKHS $\mathcal{H}_1$, as before, but then we project it onto a finite-dimensional subspace $\mathcal{F}_1 \subseteq \mathcal{H}_1$, parametrized as

$$\mathcal{F}_1 := \mathrm{Span}(\varphi(\mathbf{z}_{1,1}), \ldots, \varphi(\mathbf{z}_{1,p_1})),$$

where the $\mathbf{z}_{1,j}$'s are anchor points in $\mathbb{R}^{p_0 e_0^2}$. Then, the projected points can be represented as vector in $\mathbb{R}^{p_1}$. Specifically, consider a patch $\mathbf{x}$ in $\mathbb{R}^{p_0 e_0^2}$ of I_0, which is mapped to $\varphi_1(\mathbf{x})$ in $\mathcal{H}_1$. The projection on $\mathcal{F}_1$ of this point can be represented by a vector $\psi_1(\mathbf{x})$ in $\mathbb{R}^{p_1}$, according to (3.10). Specifically, if we choose the kernel K_1 defined in (3.12), we define

$$\psi_1(\mathbf{x}) := \|\mathbf{x}\| \kappa_1(\mathbf{Z}_1^\top \mathbf{Z}_1)^{-1/2} \kappa_1\left(\mathbf{Z}_1^\top \frac{\mathbf{x}}{\|\mathbf{x}\|}\right) \text{ if } \mathbf{x} \neq 0 \text{ and } 0 \text{ otherwise,} \tag{3.17}$$

where $\mathbf{Z}_1 = [\mathbf{z}_{1,1}, \ldots, \mathbf{z}_{1,p_1}]$ parametrizes $\mathcal{F}_1$ and $\|.\|$ is the Euclidean norm. Subsequently, the inner-product in $\mathcal{F}_1$ becomes a Euclidean one under such a mapping. This allows us to define the map $M_{0.5} : \Omega_0 \to \mathbb{R}^{p_1}$ that carries the finite-dimensional patch representations at every location in Ω_0. The spatial map $M_{0.5}$ can be obtained by (i) computing the quantities $\mathbf{Z}_1^\top \mathbf{x}$ for all patches $\mathbf{x}$ of the image M_0 (spatial convolution after mirroring the filters $\mathbf{z}_j$); (ii) contrast-normalization involving the norm $\|\mathbf{x}\|$; (iii) applying the pointwise non-linear function κ_1; (iv) applying the linear transform $\kappa_1(\mathbf{Z}_1^\top \mathbf{Z}_1)^{-1/2}$ at every pixel location (which may be seen as 1×1 spatial convolution); (v) multiplying by the norm $\|\mathbf{x}\|$ making ψ homogeneous. In other words, we obtain a particular convolutional neural network, with non-standard parametrization.

For learning the anchor points, we have adapted the algorithm of Zhang and Kwok (2010), which is fast and effective. When using dot-product kernels on the sphere such as (3.12), we constrain the columns of the $\mathbf{Z}_j$'s to be on the sphere as well. For that purpose, we build a database of n patches $\mathbf{x}_1, \ldots, \mathbf{x}_n$ randomly extracted from various images and normalized to have unit ℓ_2-norm, and perform a spherical K-means algorithm to obtain p_1 centroids $\mathbf{z}_{1,1}, \ldots, \mathbf{z}_{1,p_1}$ with unit ℓ_2-norm as well.

Finally, it is worth noting that the encoding function ψ_1 is reminiscent of radial basis function networks (RBFNs) (Broomhead and Lowe, 1988), whose hidden layer resembles (3.17) without the matrix $\kappa_1(\mathbf{Z}_1^\top \mathbf{Z}_1)^{-1/2}$ and with no normalization. The differences between RBFNs and our model is however significant. The RKHS mapping, which is absent from RBFNs, is indeed a key to the multilayer construction: a network layer takes points from the RKHS's previous layer as input and use the RKHS inner-product. To the best of our knowledge, there is no similar multilayer and/or convolutional construction in the radial basis function network literature.

Perform linear pooling in $\mathcal{F}_1$. The next step consists of performing linear pooling on the map $M_{0.5}$ to obtain another map $M_1 : \Omega_1 \to \mathbb{R}^{p_1}$, following (3.14):

$$M_1(\boldsymbol{\omega}) = \sum_{\boldsymbol{\omega}' \in \Omega_0} M_{0.5}(\boldsymbol{\omega}') e^{-\beta_1 \|\boldsymbol{\omega}' - \boldsymbol{\omega}\|_2^2}.$$

There, the pooling operation on these finite-dimensional maps does not break the kernel interpretation: The vectors $M_{0.5}(\boldsymbol{\omega}')$ can be seen as coordinates of points in $\mathcal{F}_1 \subseteq \mathcal{H}_1$ and thus the linear combination $M_1(\boldsymbol{\omega})$ of such points also admit the same interpretation.

Build a multilayer image representation. Finally, we can iterate the previous steps to build a sequence of finite-dimensional maps $M_0, \ldots, M_k$, which implicitly represent points from the finite-dimensional subspaces $\mathcal{F}_0, \ldots, \mathcal{F}_k$ of the respective possibly-infinite-dimensional RKHSs $\mathcal{H}_0, \ldots, \mathcal{H}_k$. Finally, the modified multilayer convolutional kernel (3.16) on two images I_0, I_0', which now takes into account projections steps on the linear subspaces $\mathcal{F}_1, \ldots, \mathcal{F}_k$, becomes

$$K_{\mathcal{Z}}(I_0, I_0') = \sum_{\boldsymbol{\omega} \in \Omega_k} \langle M_k(\boldsymbol{\omega}), M_k'(\boldsymbol{\omega}) \rangle, \tag{3.18}$$

where $\mathcal{Z} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_k\}$ is a set of anchor points that we need to learn.

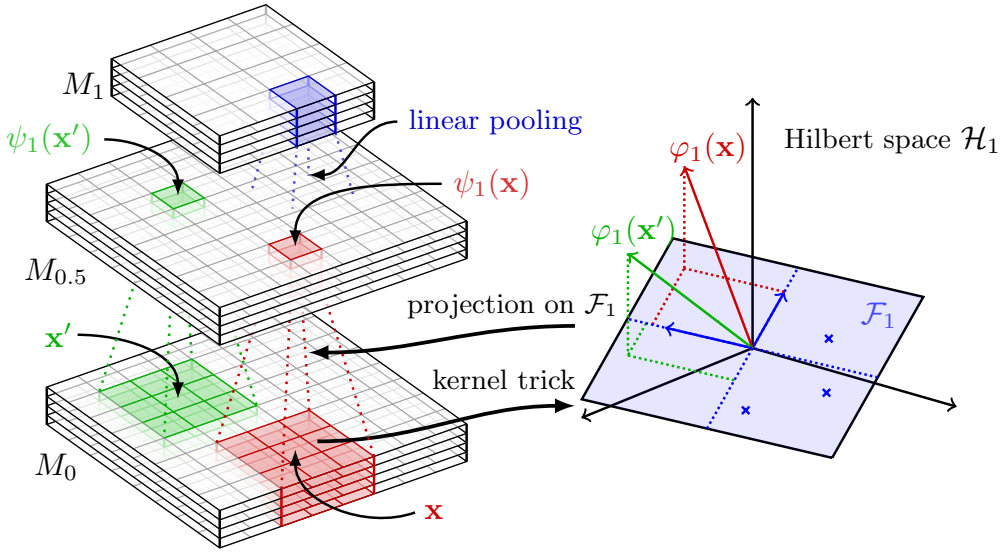


Figure 3.2 – Subspace learning variant of convolutional kernel networks, illustrated between layers 0 and 1. Local patches are mapped to the RKHS $\mathcal{H}_1$ via the kernel trick and then projected to the finite-dimensional subspace $\mathcal{F}_1 = \mathrm{Span}(\varphi(\mathbf{z}_{1,1}), \ldots, \varphi(\mathbf{z}_{1,p_1}))$. The small blue crosses on the right represent the points $\varphi(\mathbf{z}_{1,1}), \ldots, \varphi(\mathbf{z}_{1,p_1})$. With no supervision, optimizing $\mathcal{F}_1$ consists of minimizing projection residuals. With supervision, the subspace is optimized via back-propagation. Going from layer k to layer $k+1$ is achieved by stacking the model described here and shifting indices.

3.2.3 Supervised subspace learning with backpropagation

To perform end-to-end learning given labeled data, *we use a simple but effective principle consisting of learning discriminative subspaces in RKHSs, where we project data.* We implement this idea by jointly optimizing the finite-dimensional subspaces $\mathcal{F}_1, \ldots, \mathcal{F}_k$ by minimizing a supervised loss function. The formulation turns out to be a new type of convolutional neural network with a non-standard parametrization.

Specifically, we now consider a prediction task, where we are given a training set of images $I_0^1, I_0^2, \ldots, I_0^n$ with respective scalar labels $y_1, \ldots, y_n$ living either in $\{-1; +1\}$ for binary classification and $\mathbb{R}$ for regression. For simplicity, we only present these two settings here, but extensions to multiclass classification and multivariate regression are straightforward. We also assume that we are given a smooth convex loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that measures the fit of a prediction to the true label y.

Given the kernel (3.18), the classical empirical risk minimization formulation consists of finding a prediction function in the corresponding RKHS $\mathcal{H}_\mathcal{Z}$ by minimizing

$$\min_{f \in \mathcal{H}_\mathcal{Z}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(I_0^i)) + \frac{\lambda}{2} \|f\|_{\mathcal{H}_\mathcal{Z}}^2. \tag{3.19}$$

Note that the kernel is indexed by $\mathcal{Z}$, which represents the network parameters—that is, the subspaces $\mathcal{F}_1, \ldots, \mathcal{F}_k$, or equivalently the set of filters/anchor points $\mathbf{Z}_1, \ldots, \mathbf{Z}_k$.

Then, formulation (3.19) becomes equivalent to

$$\min_{\mathbf{W}\in\mathbb{R}^{p_k\times|\Omega_k|}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \langle \mathbf{W}, M_k^i\rangle_{\mathrm{F}}) + \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{3.20}$$

where $\|.\|_{\mathrm{F}}$ is the Frobenius norm that extends the Euclidean norm to matrices, and, with an abuse of notation, the maps M_k^i are seen as matrices in $\mathbb{R}^{p_k\times|\Omega_k|}$. Eventually, *the supervised convolutional kernel network formulation consists of jointly minimizing (3.20) with respect to* $\mathbf{W}$ *in* $\mathbb{R}^{p_k\times|\Omega_k|}$ *and with respect to the set of filters* $\mathbf{Z}_1, \ldots, \mathbf{Z}_k$. Note that when using dot-product kernels on the sphere (3.12), we still constrain the columns of the $\mathbf{Z}_j$'s to have unit norm, which provides a natural way to regularize them.

Since we consider a smooth loss function ℓ, *e.g.*, logistic, squared hinge, or square loss, optimizing (3.20) with respect to $\mathbf{W}$ while fixing the filters can be achieved with any gradient-based method. Moreover, when ℓ is convex, we may also use fast dedicated solvers, (see the second chapter of this thesis). Optimizing with respect to the filters $\mathbf{Z}_j$, $j = 1, \ldots, k$ is more involved because of the lack of convexity. Yet, the objective function is differentiable, and there is hope to find a "good" stationary point by using classical stochastic optimization techniques that have been successful for training deep networks.

For that, we need to compute the gradient by using the chain rule—also called "back-propagation" (LeCun et al., 1998). The details are provided in (Mairal, 2016).

3.3 Applications to image-related tasks

To demonstrate the effectiveness of the convolutional kernel networks in various contexts, we consider image classification benchmarks such as CIFAR-10 (Krizhevsky et al., 2012) and Street View House Number (SVHN) (Netzer et al., 2011), which are often used to evaluate deep neural networks without going through the engineering burden of processing the ImageNet dataset; then, we adapt our model to perform super-resolution from a single image, which is a challenging inverse problem. On the SVHN and CIFAR-10 datasets, our classification accuracy is competitive, with about 2% and 10% error rates, respectively, without model averaging or data augmentation. We consider also image up-scaling where classical convolutional neural networks have obtained outstanding results (Dong et al., 2014, 2016). The results we obtain outperform the best published results at the time of the submission of Mairal (2016). [4] All experiments have been conduced on CPUs, but we are planning to release soon a GPU implementation.

3.3.1 Image Classification on "Deep Learning" Benchmarks

We consider the datasets CIFAR-10 (Krizhevsky et al., 2012) and SVHN (Netzer et al., 2011), which contain 32×32 images from 10 classes. CIFAR-10 is medium-sized with 50 000 training samples and 10 000 test ones. SVHN is larger with 604 388

4. More precisely, better results than ours with very deep architectures and residual learning have been subsequently published at CVPR'16 (Kim et al., 2016).

training examples and 26 032 test ones. We evaluate the performance of a 9-layer network, designed with few hyper-parameters: for each layer, we learn 512 filters and choose the RBF kernels κ_j defined in (3.13) with initial parameters $\alpha_j = 4$. Layers $1, 3, 5, 7, 9$ use 3×3 patches and a subsampling pooling factor of $\sqrt{2}$ except for layer 9 where the factor is 3; Layers $2, 4, 6, 8$ use simply 1×1 patches and no subsampling. For CIFAR-10, the parameters α_j are kept fixed during training, and for SVHN, they are updated in the same way as the filters. We use the squared hinge loss in a one vs all setting to perform multi-class classification (with shared filters $\mathcal{Z}$ between classes). The input of the network is pre-processed with the local whitening procedure described in (Paulin et al., 2015). We use the optimization heuristics from the previous section, notably the automatic learning rate scheme, and a gradient momentum with parameter 0.9, following Krizhevsky et al. (2012). The regularization parameter λ and the number of epochs are set by first running the algorithm on a 80/20 validation split of the training set. λ is chosen near the canonical parameter $\lambda = 1/n$, in the range $2^i/n$, with $i = -4, \ldots, 4$, and the number of epochs is at most 100. The initial learning rate is 10 with a minibatch size of 128.

We present our results in Table 3.1 along with the performance achieved by a few recent methods without data augmentation or model voting/averaging. In this context, the best published results are obtained by the generalized pooling scheme of Lee et al. (2016). We achieve about 2% test error on SVHN and about 10% on CIFAR-10, which positions our method as a reasonably "competitive" one, in the same ballpark as the deeply supervised nets of Lee et al. (2015) or network in network of Lin et al. (2013).

	Stoch P	MaxOut	NiN	DSN	Gen P.	SCKN (Ours)
CIFAR-10	15.13	11.68	10.41	9.69	**7.62**	10.20
SVHN	2.80	2.47	2.35	1.92	**1.69**	2.04

Table 3.1 – Test error in percents reported by a few recent publications on the CIFAR-10 and SVHN datasets without data augmentation or model voting/averaging. Stoch P., MaxOut, NiN, DSN, Gen P. are the methods of Zeiler and Fergus (2013); Goodfellow et al. (2013); Lin et al. (2013); Lee et al. (2015), respectively.

Note that the results reported here only include a single supervised model. Preliminary experiments with no supervision show however that one may obtain competitive accuracy with wide shallow architectures. For instance, a two-layer network with (1024-16384) filters achieves 14.2% error on CIFAR-10. Note also that our unsupervised model outperforms original CKNs (Mairal et al., 2014c). The best single model from Mairal et al. (2014c) gives indeed 21.7%. Training the same architecture with our approach is two orders of magnitude faster and gives 19.3%. Another aspect we did not study is model complexity. Here as well, preliminary experiments are encouraging. Reducing the number of filters to 128 per layer yields indeed 11.95% error on CIFAR-10 and 2.15% on SVHN. A more precise comparison with no supervision and with various network complexities will be presented in another venue.

3.3.2 Image Super-Resolution from a Single Image

Image up-scaling is a challenging problem, where convolutional neural networks have obtained significant success (Dong et al., 2014, 2016; Wang et al., 2015b). Here, we follow Dong et al. (2016) and replace traditional convolutional neural networks by our supervised kernel machine. Specifically, RGB images are converted to the YCbCr color space and the upscaling method is applied to the luminance channel only to make the comparison possible with these previous works. Then, the problem is formulated as a multivariate regression one. We build a database of 200 000 patches of size 32×32 randomly extracted from the BSD500 dataset after removing image 302003.jpg, which overlaps with one of the test images. 16×16 versions of the patches are build using the Matlab function imresize, and upscaled back to 32×32 by using bicubic interpolation; then, the goal is to predict high-resolution images from blurry bicubic interpolations.

The blurry estimates are processed by a 9-layer network, with 3×3 patches and 128 filters at every layer without linear pooling and zero-padding. Pixel values are predicted with a linear model applied to the 128-dimensional vectors present at every pixel location of the last layer, and we use the square loss to measure the fit. The optimization procedure and the kernels κ_j are identical to the ones used for processing the SVHN dataset in the classification task. The pipeline also includes a pre-processing step, where we remove from input images a local mean component obtained by convolving the images with a 5×5 averaging box filter; the mean component is added back after up-scaling.

For the evaluation, we consider three datasets: Set5 and Set14 are standard for super-resolution; Kodim is the Kodak Image database, available at `http://r0k.us/graphics/kodak/`, which contains high-quality images with no compression or demoisaicing artefacts. The evaluation procedure follows Dong et al. (2014, 2016); Timofte et al. (2013); Wang et al. (2015b) by using the code from the author's web page. We present quantitative results in Tables 3.2. For x3 upscaling, we simply used twice our model learned for x2 upscaling, followed by a 3/4 downsampling. This is clearly suboptimal since our model is not trained to up-scale by a factor 3, but this naive approach still outperforms other baselines (Dong et al., 2014, 2016; Wang et al., 2015b) that are trained end-to-end. Note that Wang et al. (2015b) also proposes a data augmentation scheme at test time that slightly improves their results. Visual results are presented in Figure 3.3.

Fact.	Dataset	Bicubic	SC	ANR	A+	CNN1	CNN2	CSCN	SCKN
x2	Set5	33.66	35.78	35.83	36.54	36.34	36.66	36.93	**37.07**
	Set14	30.23	31.80	31.79	32.28	32.18	32.45	32.56	**32.76**
	Kodim	30.84	32.19	32.23	32.71	32.62	32.80	32.94	**33.21**
x3	Set5	30.39	31.90	31.92	32.58	32.39	32.75	**33.10**	33.08
	Set14	27.54	28.67	28.65	29.13	29.00	29.29	29.41	**29.50**
	Kodim	28.43	29.21	29.21	29.57	29.42	29.64	29.76	**29.88**

Table 3.2 – Reconstruction accuracy for super-resolution in PSNR (the higher, the better). All CNN approaches are without data augmentation at test time. SC, (ANR/A+), CNN1, CNN2, CSCN are the methods of Zeyde et al. (2010); Timofte et al. (2013); Dong et al. (2014, 2016); Wang et al. (2015b), respectively.

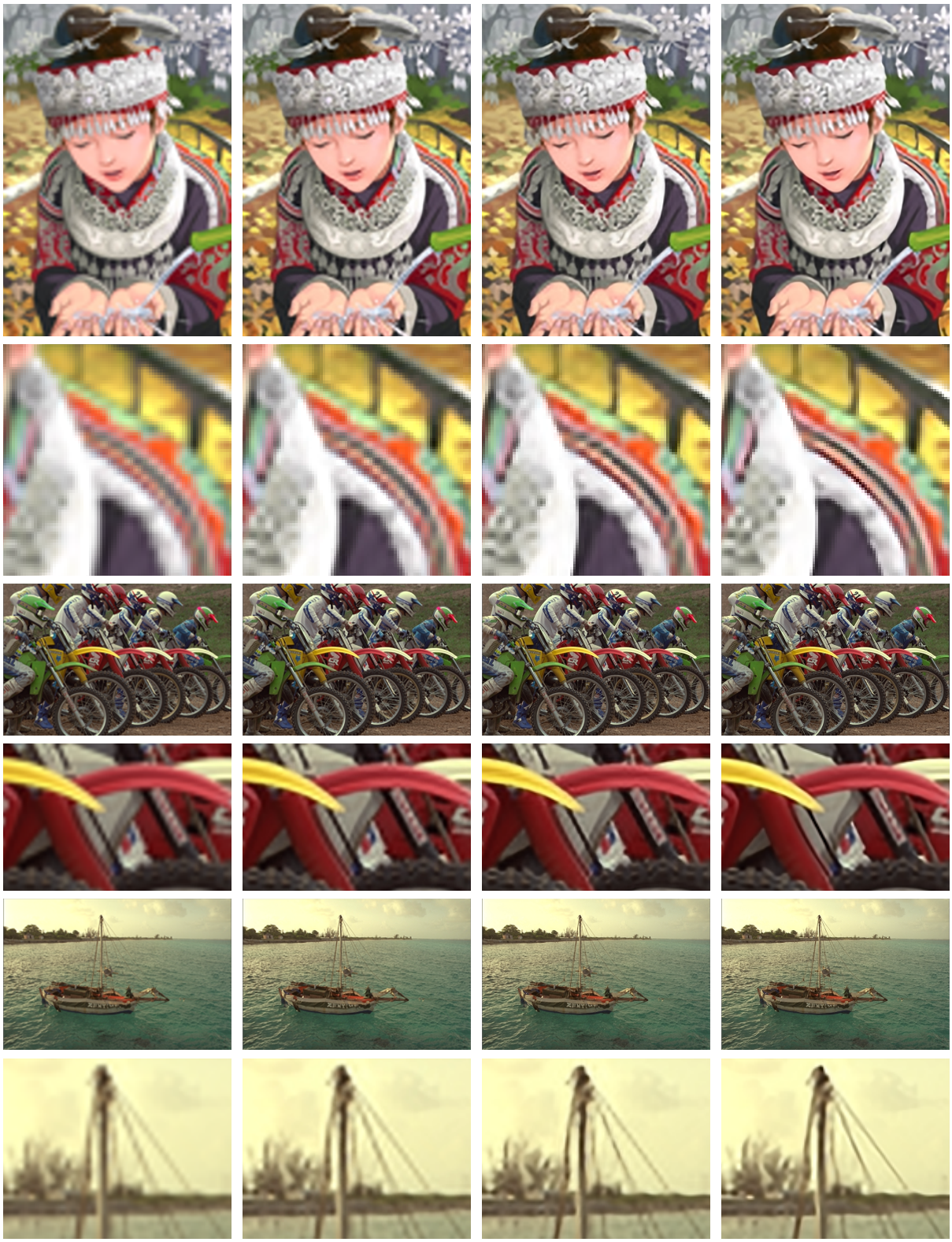


Figure 3.3 – Visual comparison for x3 image up-scaling. SC stands for Zeyde et al. (2010) and CNN2 for Dong et al. (2016).

Chapter 4

Sparse Estimation and Pluri-Disciplinary Research

This chapter is based on the following publications and on one unpublished manuscript.

J. Mairal and B. Yu. Complexity analysis of the Lasso regularization path. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

J. Mairal and B. Yu. Supervised feature selection in graphs with path coding penalties and network flows. *Journal of Machine Learning Research*, 14(1):2449–2485, 2013.

E. Bernard, L. Jacob, J. Mairal, and J.-P. Vert. Efficient RNA isoform identification and quantification from RNA-Seq data with network flows. *Bioinformatics*, 30(17): 2447–2455, 2014.

I am very grateful to Yuval Benjamini and Bin Yu, who have allowed me to use some material that we produced together for the following unpublished manuscript, which we hope will be submitted in 2017, after being in preparation for a long time.

J. Mairal, Y. Benjamini, B. D. Willmore, M. Oliver, J. L. Gallant and B. Yu. Modeling V4 under naturalistic conditions with invariant image representations. *unpublished*. 2013.

This chapter is devoted to recent pluri-disciplinary work related to the sparsity or parcimony principle, which was originally introduced by Wrinch and Jeffreys (1921) as a fundamental heuristic of science to select natural laws that seem equally plausible given observations of the world. In such a case, the preference should go to the simplest one, involving for instance the smallest number of parameters. In modern terms, the heuristic arises in statistics, machine learning, or signal processing due to the need of selecting models learned from input data (see, *e.g.* Friedman et al., 2001). Why model selection is an important issue may be explained by our limited ability to observe and understand the world. Data may be noisy, imperfect, and models considered may not be the right ones without good a priori knowledge about the problem we want to tackle.

On the one hand, the quality of data is continually improving, with technological development such as new acquisition devices (*e.g.*, better tools to see the infinitesimally small world, or better telescopes to understand early moments of the universe). On the other hand, non-technological issues may also affect the quality of our observations. To explain a given phenomenon or build a predictive system, we may simply not know what are the *a priori* right quantities to measure and where to look. For instance, automatically identifying the materials from which an object is made from visual data may seem a difficult task. Yet, it may become a trivial one if we know in advance that with the appropriate hyperspectral sensor, a single pixel may carry enough information to solve the problem with high accuracy. In this example, the right technology may exist, but a-priori knowlege is the key that tells us which one to use.

Besides, observations are intrinsically limited, regardless of technological choices: they typically consist of a finite set of numbers representing physical quantities (*e.g.*, the number of photons that hit a sensor), measured during particular time windows in particular spatial volumes; as large as the set of observations may be, this set does not represent *globally* the phenomenon, but only particular instances. The philosophical question of wether or not one may infer general laws from a limited set of observations is of course beyond the scope of this thesis—it was eventually solved by Popper (1934)—but it affects deeply the scientific foundations of machine learning, as discussed for example by Vapnik (2000). Given a collection of predictive models learned from the same data, selecting which one is "best" can potentially be achieved by testing them, and by measuring their generalization performance given new (or held-out) data.

Yet, estimating the generalization capabilities of various models, when it is possible, is not always sufficient to perform the right selection: several models may have the same or statistically indistinguishable performance; even when one model performs slightly better than another one, it is not always clear that the practitioner should prefer the former if the latter is much simpler and easier to interpret. To some extent, the sparsity heuristic provides a partial answer to the above issues. It allows us to automatically make choices among different models with similar generalization capabilities; with lack of a priori knowledge, it also allows us to use several technologies or models and automatically choose which one is relevant a posteriori. From a mathematical point of view, the principle is often implemented as a regularization function for optimization problems, which encourages the solution to have many zeroes. This is notably the case of the ℓ_1-norm, which was first introduced in geophysics (Claerbout and Muir, 1973), before being

popularized in statistics and signal processing (Chen et al., 1999; Tibshirani, 1996). For instance, the ℓ_1-regularization in empirical risk minimization formulations

$$\min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 \tag{4.1}$$

leads to sparse solutions $\mathbf{w}^\star$ (see Mairal et al., 2014a, for many interpretations).

After my PhD, my research interests shifted away from sparse estimation, but I kept an applied perspective on the topic; to conclude this line of my research, I nevertheless invested a large amount of time writing a monograph (Mairal et al., 2014a). After one last theoretical contribution that will be presented in Section 4.1, my main interest became to apply the sparsity principle to various scientific problems, essentially through external collaborations. After presenting one theoretical contribution in Section 4.1, two applied work will be presented in Section 4.2 and 4.3, in bioinformatics and neuroscience, respectively. Other collaborations, not presented here, also occured in natural language processing (Nelakanti et al., 2013), neuroimaging with massive data (Mensch et al., 2016, 2017), and signal processing for phase retrieval (Tillmann et al., 2016).

4.1 Complexity of the Lasso regularization path

In empirical risk minimization formulations such as (4.1), controlling the regularization requires to tune the parameter λ. In a few cases, the regularization path—that is, the set of solutions for all values of the regularization parameter, can be shown to be piecewise linear (Rosset and Zhu, 2007). This property is exploited in homotopy methods, which consist of following the piecewise linear path by computing the direction of the current linear segment and the points where the direction changes (also known as kinks). Piecewise linearity of regularization paths was discovered by Markowitz (1952) for portfolio selection; it was similarly exploited by Osborne et al. (2000) and Efron et al. (2004) for the Lasso, and by Hastie et al. (2004) for the support vector machine (SVM). As observed by Gärtner et al. (2012), all of these examples are in fact particular instances of *parametric quadratic programming* formulations, for which path-following algorithms appear early in the optimization literature (Ritter, 1962).

In (Mairal and Yu, 2012), we study the number of linear segments of the regularization path of the Lasso problem, defined as

$$\min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1, \tag{4.2}$$

where $\mathbf{y}$ is a vector in $\mathbb{R}^n$ and $\mathbf{X}$ is a design matrix in $\mathbb{R}^{n\times p}$. Even though experience with data suggests that the number of segments is linear in the problem size (Rosset and Zhu, 2007), it is known that discrepancies can be observed between worst-case and empirical complexities. This is notably the case for the simplex algorithm (Dantzig, 1951), which performs empirically well for solving linear programs even though it suffers from exponential worst-case complexity (Klee and Minty, 1972). Similarly, by using geometrical tools originally developed to analyze the simplex algorithm, Gärtner et al. (2012) have shown

that the complexity of the SVM regularization path can be exponential. However, none of these results do apply to the Lasso regularization path, whose theoretical complexity remained unknown, before we addressed the question in (Mairal and Yu, 2012). The goal of this work was indeed to fill in this gap and clarify the situation. More precisely, the main contribution is the following characterization:

Theorem 2 (Worst-case complexity of the Lasso regularization path). *In the worst case, the regularization path of the Lasso (4.2) with p variables has exactly $(3^p + 1)/2$ linear segments.*

We remark that our proof is constructive and significantly different than the ones proposed by Klee and Minty (1972) for the simplex algorithm and by Gärtner et al. (2012) for SVMs. Our approach does not rely on geometry but on an adversarial scheme. Given a Lasso problem with p variables, we show how to build a new problem with $p+1$ variables that increases the complexity of the path by a multiplicative factor. It results in explicit pathological examples that are surprisingly simple, unlike pathological examples for the simplex algorithm or SVMs. Note that these examples require to have at least as many observations n than variables p, and the question whether or not the result can be extended to the underdetermined case $n < p$ is still open.

Worst-case complexity analysis is by nature pessimistic. Our second contribution on approximate regularization paths is more optimistic. In fact, we show that an approximate path for the Lasso with at most $O(1/\sqrt{\varepsilon})$ segments can always be obtained, where every point on the path is guaranteed to be optimal up to a relative ε-duality gap. We follow in part the methodology of Giesen et al. (2010) and Jaggi (2011), who have presented weaker results but in a more general setting for parameterized convex optimization problems. Our analysis builds upon approximate optimality conditions, which we maintain along the path, leading to a practical approximate homotopy algorithm. In Figure 4.1, we present a pathological example with $p = 6$ variables.

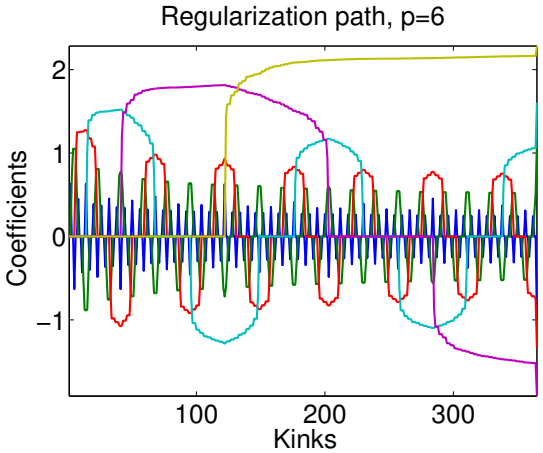


Figure 4.1 – Pathological regularization path with $p = 6$ variables and $(3^6+1)/2 = 365$ kinks. The curves represent the coefficients of the solution at every kink of the path. For visibility purposes, we use a non-linear scale $w \mapsto \text{sign}(w)|w|^{0.1}$.

4.2 Path selection in graphs and RNA-Seq

When sparse estimation was one of the most active topic in machine learning, many regularization functions that lead to *sparse and structured* solutions were introduced such as variants of the Group Lasso penalty (Grandvalet and Canu, 1999; Turlach et al., 2005; Yuan and Lin, 2006):

$$\psi(\mathbf{w}) := \sum_{g \in \mathcal{G}} \|\mathbf{w}_g\|_2, \tag{4.3}$$

where $\mathcal{G} = \{g_1, \ldots, g_k\}$ is a set of groups of indices g_j, $j = 1, \ldots, k$, which form a partition of $\{1, \ldots, p\}$, and $\mathbf{w}_g$ is the "sub-vector" of $\mathbf{w}$ that carries the entries of $\mathbf{w}$ corresponding to the indices in g. Extensions where then considered by Szafranski et al. (2007) and Zhao et al. (2009) when the groups overlap but form a tree-structure (meaning two groups overlap if and only if one is included in the other), and then the most general case of overlapping groups was studied by Jenatton et al. (2011a). The effect of the resulting penalty is to encourage many sub-vectors $\mathbf{w}_g$ to be equal to zero, hence providing a solution whose support is in the *intersection* of a few groups of variables.

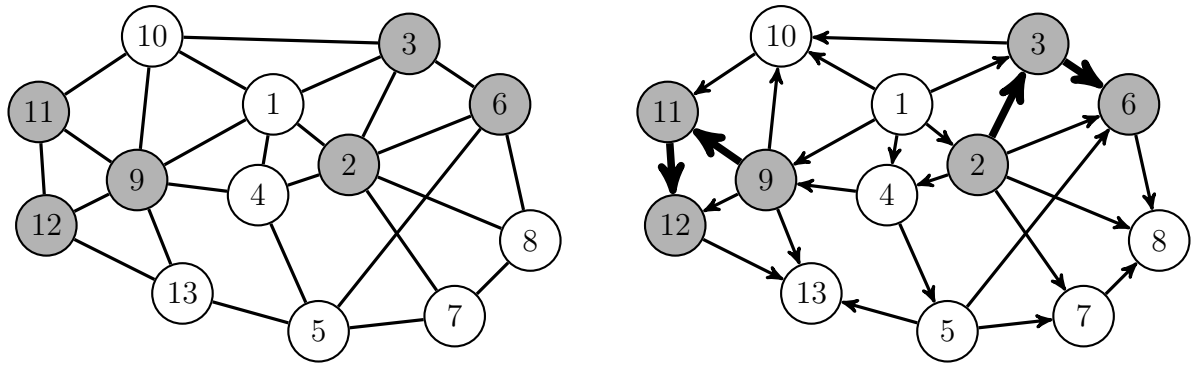
At the same time, another class of convex penalties was proposed by Jacob et al. (2009) to induce the opposite effect—that is, encouraging solutions whose support is in the *union* of a few groups—which have the following variational form

$$\psi(\mathbf{w}) := \min_{(\mathbf{z}^g \in \mathbb{R}^p)_{g \in \mathcal{G}}} \left\{ \sum_{g \in \mathcal{G}} \eta_g \|\mathbf{z}^g\|_2 \;\text{ s.t. }\; \mathbf{w} = \sum_{g \in \mathcal{G}} \mathbf{z}^g \quad \text{and} \quad z^g_j = 0 \;\; \forall\, j \notin g \right\}, \tag{4.4}$$

where $\mathbf{z}^g = [z^g_1, \ldots, z^g_p]^\top$ for all g in $\mathcal{G}$, and the weights η_g are non-negative. By construction, the solution $\mathbf{w}$ is a sum of vectors $\mathbf{z}^g$, where each $\mathbf{z}^g$'s support is included in the group g. The sum of ℓ_2-norms can be interpreted as a Group-Lasso penalty, which encourages many of the vectors $\mathbf{z}^g$ to be exactly zero. As a result, the support of a solution $\mathbf{w}$ will be in the union of a few groups g, which is the desired effect.

A natural question that arises from these constructions is whether or not there are interesting group structures $\mathcal{G}$, *e.g.*, with an exponential number of groups, that lead to tractable formulations. In (Nelakanti et al., 2013), we found such a formulation for an application in natural language processing, where we considered the penalty (4.3) for infinite vectors where the groups have a tree-structure. Next, we will summarize another case, where the goal is to select paths in a directed acyclic graph.

Path coding penalties for directed acyclic graphs. We now consider the variational penalty (4.4) where the entries of the parameter vector $\mathbf{w}$ can be mapped to the nodes of a directed acyclic graph $G = (V, E)$, where V is a set of p vertices and $E \subset V \times V$ is an arc set. This setting, introduced in (Mairal and Yu, 2013), was motivated by a sequence of works that tried to perform variable selection in graphs (Jacob et al., 2009; Huang et al., 2009)—that is, encouraging the selection of variables forming few connected components. Typically, the situation occurs with gene expressions organized in a gene network. In this context, it may be interesting to take into account the graph structure in the regularization, for instance, to automatically identify a subgraph with a few connected components such as groups of genes involved in a disease. There are two equally



(a) Sparsity pattern in an undirected graph.

(b) Selected paths in a DAG.

Figure 4.2 – Left: an undirected graph with 12 nodes. A sparsity pattern forming a subgraph with two connected components is represented by gray nodes. Right: when the graph is a DAG, the sparsity pattern is covered by two paths $(2, 3, 6)$ and $(9, 11, 12)$ represented by bold arrows.

important reasons for this: either connectivity of the solution is good prior information which might improve the prediction performance, or connected components may be easier to interpret than isolated variables would be.

When we attacked this problem, the limitations of existing approaches were twofold: either penalties in the literature would not model long-range interactions in the graph (Jacob et al., 2009), or they would lead to intractable NP-hard optimization problems (Huang et al., 2009). When the graph G is *directed and acyclic*, we propose in (Mairal and Yu, 2013) a solution that builds upon two ideas. First, we use in the penalty framework of Jacob et al. (2009) a novel group structure $\mathcal{G}_p$ which contains *all the paths* in G, where a path is defined as a sequence of vertices $(v_1, \ldots, v_k)$ such that for all $1 \leq i < k$, we have $(v_i, v_{i+1}) \in E$. The second idea is to use appropriate costs η_g for each path, which allows us to leverage network flow optimization. We call the resulting regularization functions "path coding" penalties. They go beyond pairwise interactions between vertices and model long-range interactions between the variables in the graph. They encourage sparsity patterns forming subgraphs that can be covered by a small number of paths, therefore promoting connectivity of the solution. We illustrate the "path coding" concept for DAGs in Figure 4.2. Even though the number of paths in a DAG is exponential in the graph size, we map the *path selection* problems our penalties involve to network flow formulations (see Ahuja et al., 1993; Bertsekas, 1998), which can be solved in polynomial time. This allows us to efficiently compute the penalties and their proximal operators, and use them in regularized empirical risk minimization formulations.

As a result, we design a new link between structured graph penalties in graphs and network flow optimization. The development of network flow optimization techniques has been very active from the 60's to the 90's (see Ford and Fulkerson, 1956; Goldberg and Tarjan, 1986; Ahuja et al., 1993; Goldberg, 1997; Bertsekas, 1998). They have attracted a lot of attention during the last decade in the computer vision community for their

ability to solve large-scale combinatorial problems typically arising in image segmentation tasks (Boykov et al., 2001). Concretely, by mapping a problem at hand to a network flow formulation, one can possibly obtain fast algorithms to solve the original problem. Of course, such a mapping does not always exist or can be difficult to find. This is made possible in the context of our path coding penalties thanks to decomposability properties of the path costs, which we make explicit in (Mairal and Yu, 2013).

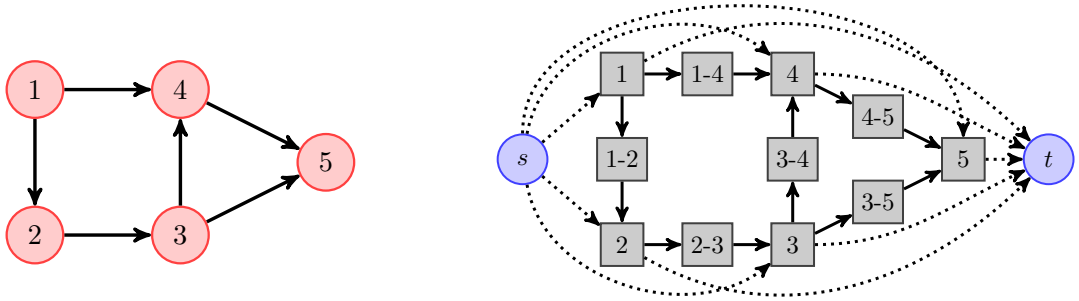
Yet, when we published this work, a convincing application on real data was missing. The opportunity to use similar optimization techniques came later as a collaboration in bioinformatics with Jean-Philippe Vert, Laurent Jacob and Elsa Bernard.

Isoform discovery in RNA Seq data. In (Bernard et al., 2014), we consider the problem of isoform discovery and quantification in RNA-Seq data, which consists of identifying which proteins encoded by a specific gene are produced in a cell and in which abundance. The process of a single gene encoding multiple proteins is known as *alternative splicing* (see Pan et al., 2008) and is believed to have a key role in many diseases. Specifically, genes in eukaryote cells contain a succession of DNA sequences called exons and introns. Transcription results in a pre-mRNA molecule from which most introns are removed and some exons are retained during the processing step called RNA splicing: the set of selected exons can vary, resulting for the same gene in different versions of the mRNA, referred to as transcripts or isoforms. Identification and quantification of isoforms present in a sample is of outmost interest because different isoforms can later be translated as different proteins.

Unfortunately, most sequencing technologies are not able to read directly long RNA fragments contained in a cell. They start by cutting them into small pieces, which are called *reads*. Then, the problem of isoform discovery and quantification consists in recovering long RNA fragments from the set of small reads, which are mixed together. Even though it is possible to align the reads on the genome and to know on which exons each read is located, recovering the long RNA fragments is a difficult ill-posed inverse problem. Next, we will show how to reformulate it as a path selection problem in a directed acyclic graph, making the problem compatible with network flow optimization.

First, we consider the observed reads, and place them into different bins, which represent the set of exons they overlap. For instance, consider a gene with 5 exons, as in Figure 4.3. We place in bin $(3-5)$, all reads that start in exon 3 and end in 5 without overlapping with 4—reads from this bin necessarily come from an isoform produced without exon 4, *e.g.*, isoform $(1-2-3-5)$ or $(2-3-5)$. Naturally, the bins can be ordered to form a directed acyclic graph $G = (V, E)$, as shown in Figure 4.3b, which we call "FlipFlop graph", where FlipFlop is the name of the open-source software that was produced in this work. The graph extends the concept of *splicing graph* introduced by Heber et al. (2002). After building the graph from the set of observed reads, there is a one-to-one mapping between the paths in G (starting from the source node s and ending at the sink node t) and the candidate isoforms.

To solve the isoform detection problem, we use an extension of the classical statistical model of Jiang and Wong (2009), which models read counts in bins using a Poisson distribution. The model involves physical parameters such as the length of each bin since



(a) Splicing graph with 5 exons.

(b) A possible FlipFlop graph.

Figure 4.3 – Left: the splicing graph of a gene with 5 exons. There is a one-to-one mapping between the possible isoforms and the paths in this directed acyclic graph. Right: the Flip-flop graph constructed from the set of observed reads. Each node corresponds to a read type, *e.g.*, $3-4$ denotes a read that starts in exon 3 and ends in exon 4. For simplicity, we do not consider the scenario where a read overlaps with more than two exons in this figure (which may occur when an exon length is smaller than the read length); the graph can be extended to deal with such cases (see Bernard et al., 2014).

a bin corresponding to a long exon is likely to receive more reads than a bin related to a shorter exon. Formally, let us call m the number of candidate isoforms (which is exponential in the graph size), and q the number of bins (nodes in the graph). We denote by $\mathbf{U}$ the $m \times q$ binary matrix defined as $\mathbf{U}_{ji} = 1$ if bin i is associated to isoform j and 0 otherwise—meaning the node/bin i is along the isoform/path j— and by $\theta_j \in \mathbb{R}_+$ the expression of isoform j (the expected number of reads per base in isoform j).

Thus, $\sum_{j=1}^{m} \mathbf{U}_{ji}\theta_j$ represents the sum of expressions of all isoforms involving bin i. We expect the observed count for bin i to be distributed around this value times the length of the bin l_i, and therefore, we model the read count y_i as a Poisson random variable with parameter $\delta_i = l_i \sum_{j=1}^{m} \mathbf{U}_{ji}\theta_j$. For a vector $\boldsymbol{\theta} = [\theta_j]_{j=1}^{m}$ in $\mathbb{R}_+^m$, this yields the log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{q} [-\delta_i + y_i \log \delta_i - \log(y_i!)] \quad \text{with} \quad \delta_i = l_i \sum_{j=1}^{m} \mathbf{U}_{ji}\theta_j. \tag{4.5}$$

Since minimizing the log-likelihood is a highly ill-posed problem involving an exponential number of variables, we consider a sparse regularization to encourage the selection of a few isoforms only. More precisely, we want to minimize

$$\min_{\boldsymbol{\theta} \in \mathbb{R}_+^m} \mathcal{L}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1, \tag{4.6}$$

which looks infeasible at first sight, since m is exponentially large, unless the gene involves very few exons—say 10. Yet, we will show next that an exact solution can be obtained by using network flow optimization. The first step consists of rewriting the objective in

terms of quantities associated to nodes $v \in V$ of the graph, and paths $p \in \mathcal{P}$, where $\mathcal{P}$ represents the set of all paths from G that start from the source s and end at the sink t.

Based on the one-to-one mapping between isoforms and paths, we can reformulate the penalized maximum likelihood problem (4.6) as follows: we want to find nonnegative weights θ_p for each path $p \in \mathcal{P}$ which minimize:

$$\sum_{v \in V} [\delta_v - y_v \log \delta_v] + \lambda \sum_{p \in \mathcal{P}} \theta_p \quad \text{with} \quad \delta_v = \Big(l_v \sum_{p \in \mathcal{P} : p \ni v} \theta_p \Big), \tag{4.7}$$

where the sum $\sum_{p \in \mathcal{P}} \theta_p$ is equal to the ℓ_1-norm $\|\boldsymbol{\theta}\|_1$ since the entries of $\boldsymbol{\theta}$ are non-negative. Note that we have removed the constant term $\log(y_v!)$ from the log likelihood since it does not play a role in the optimization. This reformulation is therefore a path selection (finding which θ_p are non-zero) and quantification problem over G. Next, we will show how (4.7) can further be written as a flow problem, *i.e.*, technically a constrained optimization problem over the edges of the graph rather than the set of paths in $\mathcal{P}$.

A *flow* f on G is defined as a non-negative function on arcs $[f_{uv}]_{(u,v) \in E}$ that satisfies conservation constraints: the sum of incoming flow at a vertex is equal to the sum of outgoing flow except for the source s and the sink t. Such a conservation property leads to a physical interpretation about flows as quantities circulating in the network, for instance, water in a pipe network or electrons in a circuit board. The source node s injects into the network some units of flow, which move along the arcs before reaching the sink t.

For example, given a path $p \in \mathcal{P}$ and a non-negative number θ_p, we can make a flow by setting $f_{uv} = \theta_p$ when u and v are two consecutive vertices along the path p, and $f_{uv} = 0$ otherwise. This construction corresponds to sending θ_p units of flows from s to t along the path p. Such simple flows are called (s,t)-*path flows*. More interestingly, if we have a set of non-negative weights $\boldsymbol{\theta} \in \mathbb{R}_+^{|\mathcal{P}|}$ associated to all paths in $\mathcal{P}$, then we can form a more complex flow by superimposing all (s,t)-path flows according to

$$f_{uv} = \sum_{p \in \mathcal{P} : p \ni (u,v)} \theta_p, \tag{4.8}$$

where $(u,v) \in p$ means that u and v are consecutive nodes on p.

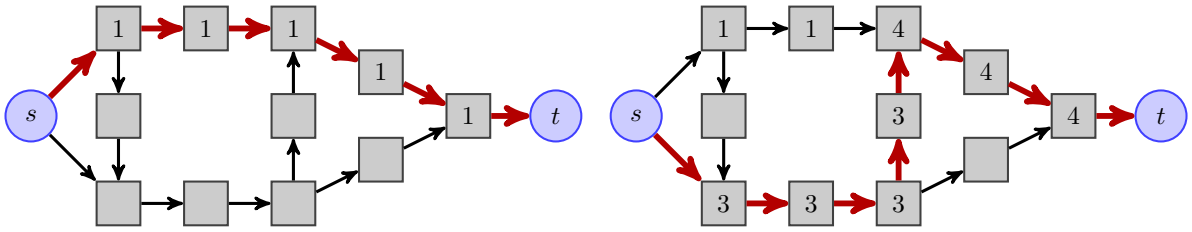
While (4.8) shows how to make a complex flow from simple ones, a converse exists, known as the *flow decomposition theorem* (see, *e.g.*, Ahuja et al., 1993). It says that for any DAG, every flow vector can always be decomposed into a sum of (s,t)-path flows. In other words, given a flow $[f_{uv}]_{(u,v) \in E}$, there exists a vector $\boldsymbol{\theta}$ in $\mathbb{R}_+^{|\mathcal{P}|}$ such that (4.8) holds. Moreover, there exists linear-time algorithms to perform this decomposition (Ahuja et al., 1993). As illustrated in Figure 4.4, this leads to a flow interpretation for isoforms.

We now have all the tools in hand to turn (4.7) into a flow problem by following Mairal and Yu (2013). Given a flow $f = [f_{uv}]_{(u,v) \in E}$, we define the amount of flow incoming to a node v in V as $f_v := \sum_{u \in V : (u,v) \in E} f_{uv}$. Given a vector $\boldsymbol{\theta} \in \mathbb{R}_+^{|\mathcal{P}|}$ associated to f by the flow decomposition theorem, *i.e.*, such that (4.8) holds, we remark that $f_v = \sum_{p \in \mathcal{P} : p \ni v} \theta_p$ and that $f_t = \sum_{p \in \mathcal{P}} \theta_p$. Therefore, problem (4.7) can be equivalently rewritten as:

$$\min_{f \in \mathcal{F}} \sum_{v \in V} [\delta_v - y_v \log \delta_v] + \lambda f_t \quad \text{with} \quad \delta_v = l_v f_v \,. \tag{4.9}$$

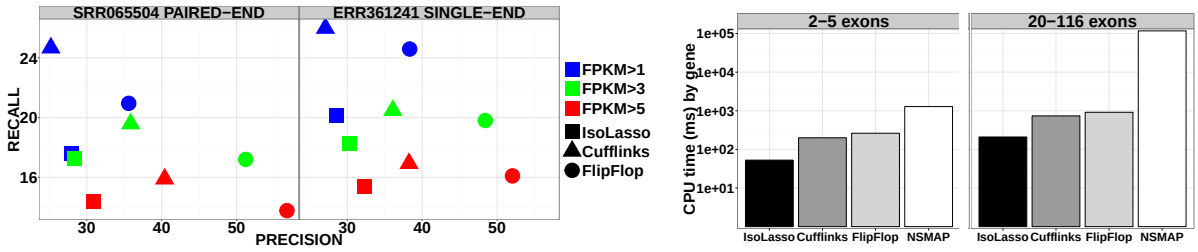
where $\mathcal{F}$ denotes the set of possible flows. Once a solution $f^\star$ of (4.9) is found, a solution $\boldsymbol{\theta}^\star$ of (4.7) can be recovered by decomposing $f^\star$ into (s,t)-path flows. The use of network flows has two consequences. First, (4.9) involves a polynomial number of variables, as many as arcs in the graph, whereas this number was exponential in (4.7). Second, problem (4.9) falls into the class of *convex cost flow* problems (Ahuja et al., 1993), for which efficient algorithms exist. In our experiments, we implemented a variant of the scaling push-relabel algorithm (Goldberg and Tarjan, 1986), which also appears under the name of ε-relaxation method (Bertsekas, 1998).

This work resulting in an open-source software package called FlipFlop [1], which achieves competitive performance in terms of speed and quality for the recovery of isoforms (see the experimental section of Bernard et al., 2014) and Figure 4.5. Later, an extension was developed to deal with multiple samples at the same time (Bernard et al., 2015)—that is, to process data from several experiments at the same time, where it is assumed that all experiments involve a common superset of isoforms to recover.



(a) One unit of flow along a path. (b) Three units of flow along another path.

Figure 4.4 – Flow interpretation of isoforms using the same graph as in Figure 4.3. For the sake of clarity, some edges connecting s and t to internal nodes are not represented, and the bins are assumed to be of equal length. On the left, one unit of flow is carried along the path in red, corresponding to an isoform with abundance 1. On the right, another isoform with abundance 3 is added, yielding additional read counts at every node.



(a) Precision-recall comparison. (b) Speed comparison in CPU time.

Figure 4.5 – We compare the precision-recall of various methods and their speed in CPU time. The evaluation is performed on real human RNA-Seq data (see Bernard et al., 2014, for more details). NSMAP, Cufflinks and IsoLasso are respectively from Xia et al. (2011); Trapnell et al. (2010) and Li et al. (2011).

1. FlipFlop for R is available in the Bioconductor repository `http://www.bioconductor.org`.

4.3 A computational model for area V4

In this section, we summarize the results of a collaboration in neuroscience, where we use the sparsity principle to build an interpretable predictive model of area V4 from the mammalian visual cortex. This is a joint work involving statisticians (Yuval Benjamini and Bin Yu) and neuroscientists from UC Berkeley (Jack Gallant, Michael Oliver, Ben Willmore). Most of the material that follows here is based on an unpublished manuscript (Mairal et al., 2013), and also appears in the PhD thesis of Yuval Benjamini.

Specifically, visual cortex area V4 is located in the mammalian ventral visual pathway. It is believed to play an important role in the recognition of shapes and objects and in visual attention (Roe et al., 2012), but its complexity makes it hard to analyze. The primary role of V4 neurons remains an open question, and its functional organization is not well understood. Moreover, models of V1 and V2 are typically not performing well in terms of prediction for neuronal responses to natural images (David and Gallant, 2005; Willmore et al., 2010). In this context, the goal of our work was twofold: *build a model with good prediction performance, and provide a clear interpretation.*

Task and data collection. The data consists in electrophysiological recordings measured at 70 well-isolated neurons in area V4 from two awake, behaving, macaques. For each neuron, the acquisition is performed while displaying a sequence of 4 000 to 12 000 natural images to the subjects at 30Hz, centered on the central receptive field (CRF), while the animals fixated. The number of spikes for each neuron was measured at 60Hz, a similar rate as in recent studies about V1 and V2 (Willmore et al., 2010), resulting in two measurements per image. This data was used to learn the parameters of our model. To measure the prediction accuracy, a second test data with higher signal to noise ratio was acquired, using a sequence of 300 images that were kept aside from the training set.

Low-rank regularization for learning nearly space-time-separable models. Formally, for each neuron, we are given a sequence of images $I_1, \dots, I_T$, along with a sequence of spiking rates $y_1, \dots, y_T$. Neural responses are not instantaneous with respect to a visual stimuli and neurons in V4 have typically a long memory; thus, we predict the responses y_t at time t given visual stimuli shown at previous times $t-9(-166.7ms), \dots, t-1(-16.7ms)$, as illustrated in Figure 4.6. The time window of 9 lags, was selected here by cross validation (we decided to use the same time window for all neurons). Specifically, each image I_t is encoded into a high-dimensional vector $\psi(I_t)$ in $\mathbb{R}^p$ using a nonlinear encoding function ψ that will be described next. Then, we consider the linear model

$$y_t \approx \sum_{j=1}^{\tau} \psi(I_{t-j})^\top \mathbf{w}_j, \tag{4.10}$$

where the parameters to learn form a matrix $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_\tau]$ in $\mathbb{R}^{p \times \tau}$ (with $\tau = 9$).

Before we discuss the choice of image model ψ, two questions naturally arise: which loss function may be used to learn $\mathbf{W}$ and with which regularization? Regarding the first point, a natural statistical model for spiking rates is a Poisson distribution. Yet, when learning the model (4.10), we observed that using the square loss (after centering the

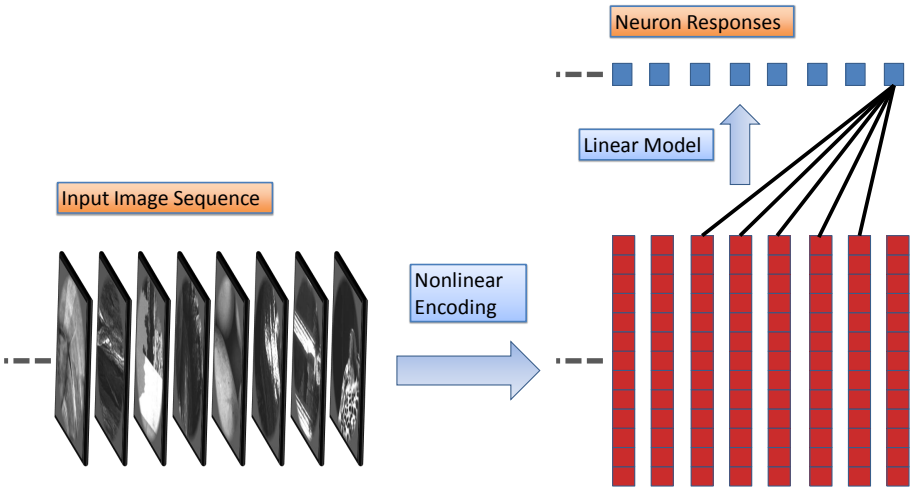


Figure 4.6 – Illustration of the prediction pipeline. Frames from time $t-1,\ldots,t-10$ are used to predict the neuron activity at time t.

responses as pre-processing) gives similar prediction performance as a negative Poisson log-likelihood, while being much simpler to minimize. Then, we encourage the matrix $\mathbf{W}$ to be low-rank by using the trace norm regularization (sum of singular values):

$$\min_{\mathbf{W}\in\mathbb{R}^{p\times\tau}} \frac{1}{T-\tau}\sum_{t=\tau+1}^{T}\frac{1}{2}\left(y_t-\sum_{j=1}^{\tau}\psi(I_{t-j})^\top\mathbf{w}_j\right)^2+\lambda\|\mathbf{W}\|_*.$$

The motivation for encouraging low rank is to obtain solutions that are nearly space-time separable—that is, a neuron response should be close to the product of a time function and a spatial pattern function, which is exactly achieved when $\mathbf{W}$ is rank one. This is in contrast to non-separable models, where responses to spatial patterns can arbitrarily change over time. For simple cells, it was found that the space-time separability assumption was approximately correct for some neurons but not all of them (Mazer et al., 2002). For instance, we report a typical time response of a neuron (learned by our model) in Figure 4.8a: after an excitatory image is shown to the subject at time $t=0$, the neuron is excited at the lag $t=2$, before going to an inhibitory state where no stimuli will excite him. In practice, the low-rank assumption allows us to find the right trade-off. By choosing λ by cross-validation, we observed that for 68 out 70 neurons, the ranks were spread between 3 and 6, which seems to confirm that interpolating between separable and non-separable models is an appropriate strategy for modeling V4 neurons.

An unsupervised sparse coding model to learn interpretable features. We briefly describe here the image model ψ, inspired by sparse coding approaches that were the state of the art for image classification when we developed the predictive model in 2011 (Yang et al., 2009). The pipeline is illustrated in Figure 4.7 and consists of three layers. Each layer is connected to the output of the preceding one and produces several spatial maps. Fhe first layer directly processes an input image and produces 8 orientation

maps. This is achieved by keeping the positive response to the image convolution with 8 local oriented filters. Then the orientation maps are smoothed and down-sampled by a factor 8, introducing invariance for local shifts and deformations.

The second layer is connected to the orientation maps and produces feature maps. They differ from orientation maps in their selectivity to higher-level features tuned to natural images (*e.g.*, features representing edges, bars, corners, curves, and texture), and are invariant to local contrast changes. These features are obtained by sparsely encoding patches from the previous orientation maps on a learned dictionary. Specifically, we learn offline the dictionary by extracting a large number—say 1 000 000— patches $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^m$ from orientation maps. We normalize them for contrast, and learn the dictionary in $\mathbb{R}^{m \times p}$ according to the classical formulation (see Mairal et al., 2014a):

$$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A} \in \mathbb{R}_+^{p \times n}} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1,$$

where $\mathcal{C}$ is the set of matrices whose columns are in the unit ℓ_2-ball. As in computer vision applications (Yang et al., 2009), we choose $p = 2\,048$ dictionary elements and set $\lambda = 0.1$, which leads to about 10 features selected per patch on average. Once the dictionary is learned, we may now produce the 2 048 feature maps of an image, by encoding all overlapping patches from the orientation maps using $\mathbf{D}$.

The third layer makes the feature maps of the second layer shift invariant inside some pre-defined pooling regions. Each feature map is partitioned into one central and four peripheral areas (see Figure 4.6), and the pixels in a feature map falling into a pooling region are summarized by their ℓ_2-norm. The output of this hierarchical image processing pipeline is a high-dimensional vector (5×2048) representation $\psi(I)$ for each image.

Prediction performance. We present in Figure 4.8 the performance of our model versus a state-of-the-art model based on Gabor features, which was developed in the Gallant's lab. The scatter plot in Figure 4.8c displays a clear advantage over the Gabor model in terms of correlation between the true and predicted responses evaluated on a test set (average correlation 0.457 vs 0.398). Next, we will present some tools to intepret the activity of each neuron, which is the most challenging part of this work.

Annotation of dictionary elements. A natural question is what kind of shape or surface property each of the learned dictionary element represents? This cannot be answered automatically and requires visualization, interpretation, and manual annotation. Thus, we developed visualization tools to study and categorize these dictionary elements. Since the dictionary elements were not learned on raw image-patches but on orientation maps, they cannot be directly displayed. We overcame this difficulty by forming two million pairs of patches from orientation maps and their corresponding raw image patches. To visualize a feature or a dictionary element, we found the closest orientation patches to this element and displayed their corresponding raw image patches. In Figure 4.9, we visualize 12 dictionary elements (rows). For each of them, we display the natural image patches from our database whose orientation maps are the most correlated with the element (right), and their mean (left).

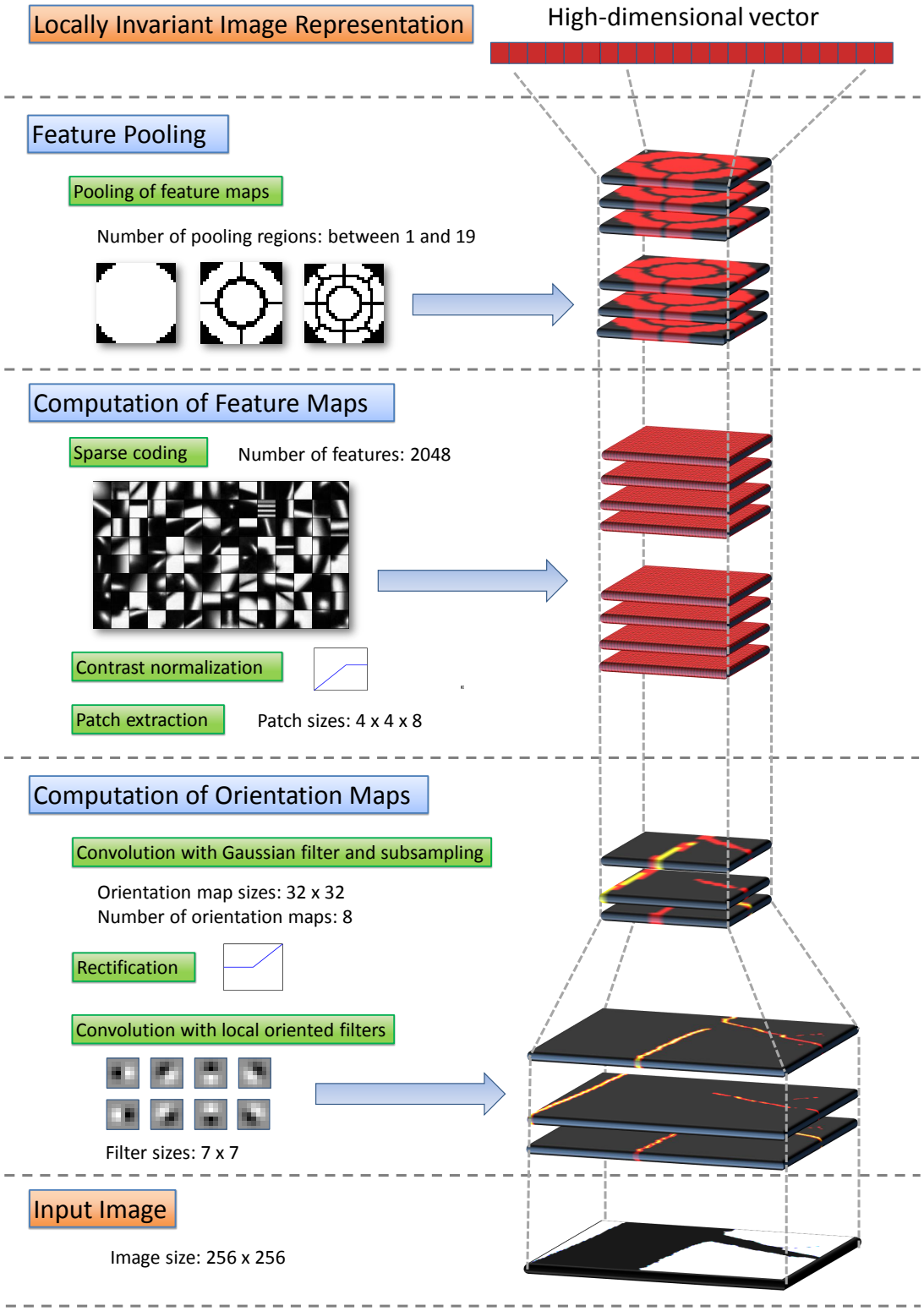
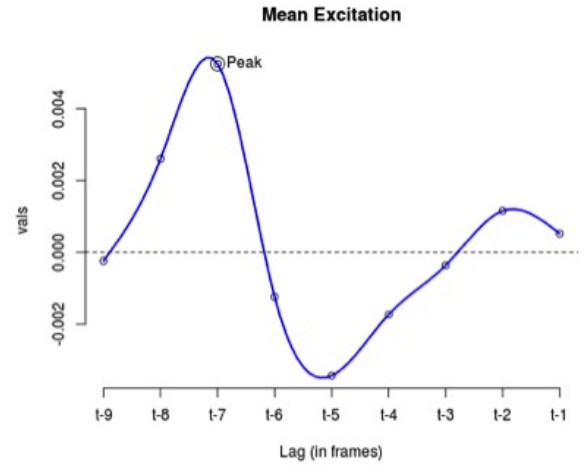
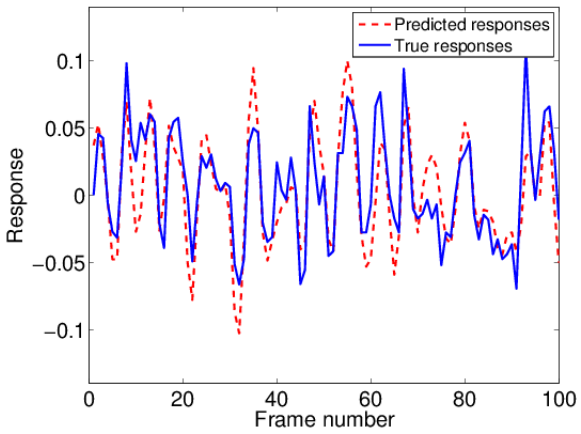


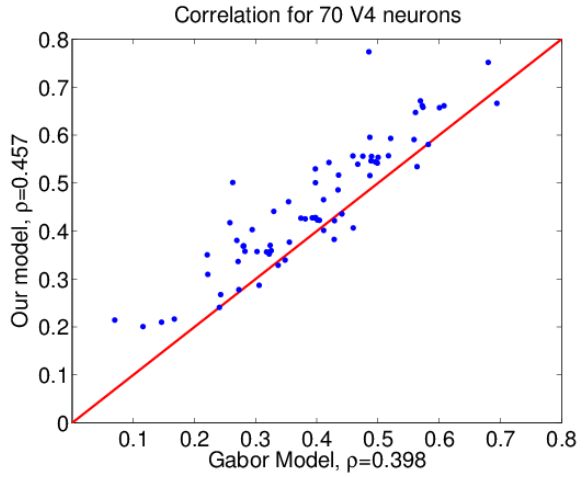
Figure 4.7 – Illustration of the image model, based on unsupervised learning.



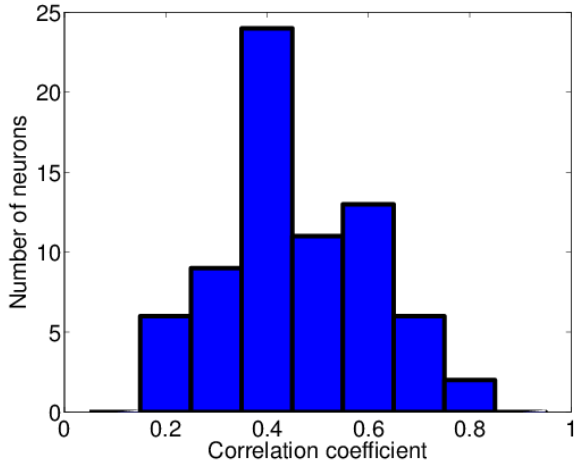
(a) Time response of a neuron.



(b) Predicted and true responses.



(c) Comparison between a Gabor model and ours.



(d) Distribution of correlation coefficients.

Figure 4.8 – Various results obtained by our predictive model.

With a visualization tool in hand, we manually annotated the dictionary. We observed that many dictionary elements were associated to image patches with straight edges between bright and dark surfaces. Other elements were associated to patches with curved edges of varying complexity, as well as corners. The full taxonomy we obtained is presented in Figure 4.9; it includes additional shape features such as bars, blobs, and surface features such as stripes and non-oriented texture. For downstream analysis, we consolidated the dictionary elements into eight categories. Image patches correspond to a feature or element shared not only a shape, but also other image attributes. Different features preferred textured versus un-textured surfaces. Furthermore, shapes varied in their predominant orientation and in their size. Obtaining such a rich annotated set is particularly interesting for interpretation purposes. Neuron selectivity profiles that one can discover by interpreting a model are often restricted by the image feature set the model is built on. In contrast, the diversity of our features translates into large diversity of selectivity profiles our model can characterize.

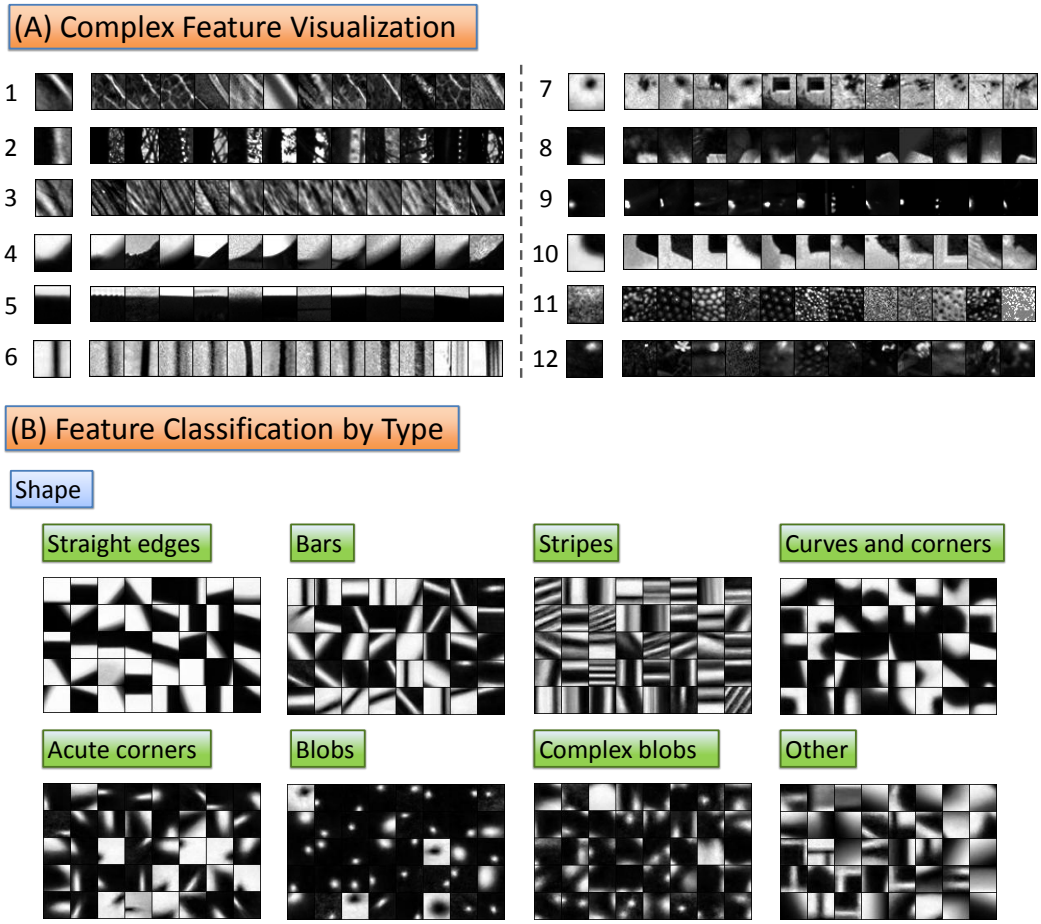


Figure 4.9 – Classification and visualization of local image features based on sparse coding.

Visualizing the activity of single neurons. We developed two independent ways of interpreting the model, which we hope to be consistent with each other when we try to understand the activity of a specific neuron. The first way is based on the annotation of the dictionary elements and the other one consists of displaying the most "excitatory images" for each neuron. We will now sketch these interpretation procedures, and will provide more details in the manuscript (Mairal et al., 2013), once it is published.

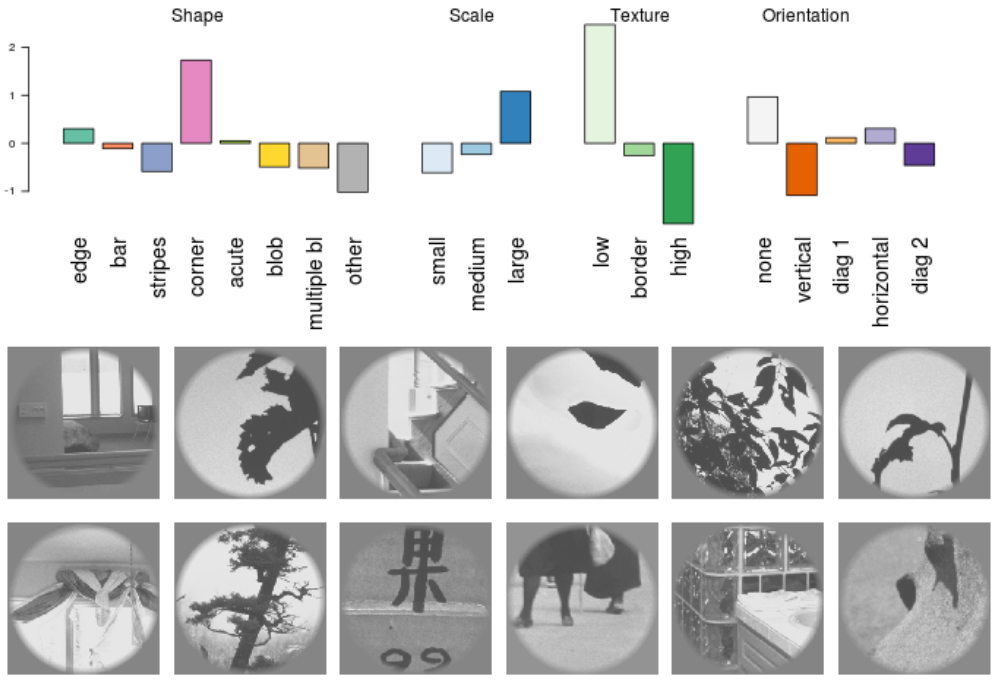
Both approaches require a common step, consisting of estimating the peak response lag of the neuron, as illustrated in Figure 4.8a. This is achieved by computing the average contribution of each image to a specific lag in our model. In other words, we compute for each lag $j = 1, \ldots, \tau$, the contribution $\frac{1}{n}\sum_{t=1}^{n} \psi(I_t)^\top \mathbf{w}_j$. Typically, a shape similar to Figure 4.8a can be obtained, and the peak excitatory lag $j^\star$ may be estimated. Then, it is possible to display the most excitatory images that maximize the product $\psi(I_t)^\top \mathbf{w}_{j^\star}$.

The second procedure uses the annotation of the dictionary to compute "impact values" for each neuron, which consists of turning the model weights into values measuring the direction and impact of each category of features on the response. The impact of a feature category measures (i) whether the presence of features from this category excites

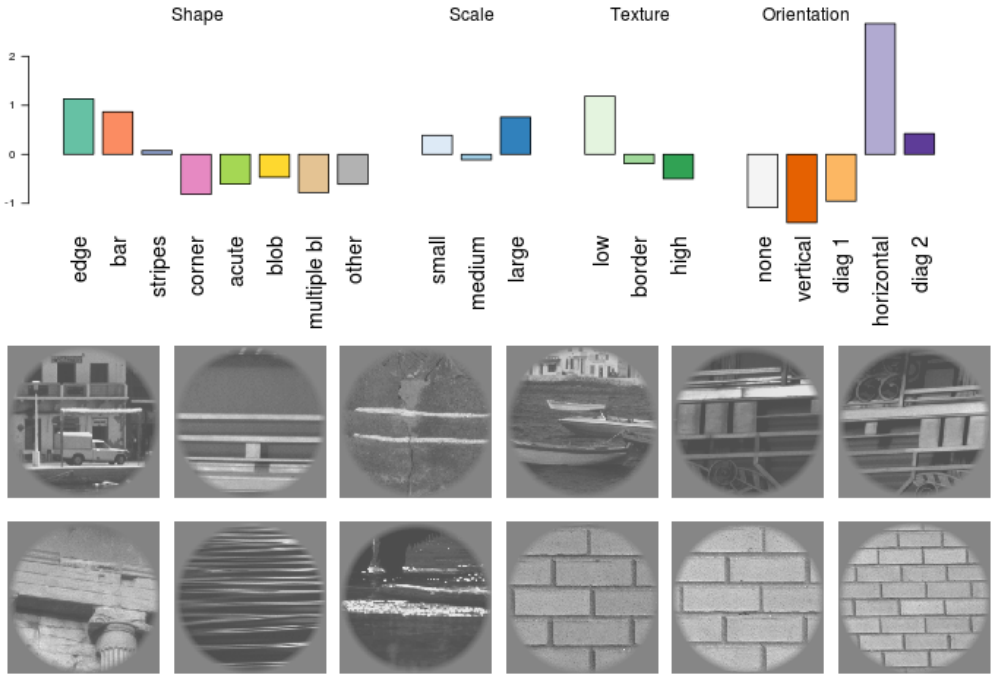
or inhibits the response (directionality) and (ii) the variation in responses due to features from this category (magnitude). Large impact values were given to categories whose features are large and have a dominant directionality, either positive or negative. Impact values were computed for the four classifications schemes described above (8 shapes, 5 orientations, 3 scales and 3 textures) resulting in 19 impact values total. More details will be provided in (Mairal et al., 2013). In Figures 4.10 and 4.11, we present our interpretations for four neurons whose activity is well predicted by our model (correlation greater than 0.6). Two of them respond to non-oriented features, respectively corners and texture, while two of them respond to oriented features (horizontal lines, and bars).

Population analysis with sparse principal component analysis. Finally, we perform a population analysis of the 70 neurons, by using the impact values computed for each neuron. We use the sparse principal component analysis formulation of Zou et al. (2006), which provides easier interpretation of the principal components. In Figure 4.12a, we present the five first principal components learned by our model and their interpretation. On the left are the loadings of each component to the feature categories. On the right, a visualization of these contrasts in the stimulus space: the center-right images would respond negatively to the contrast, and far-right images respond positively. The first component differentiates texture features (positive) from edges and contours. The second component is a contrast of orientation: vertical vs. horizontal. A similar orientation contrast (diag 1 vs. diag 2) can be seen in component 5. The final two shape components, 3 and 4, describe variation between neurons belonging to the larger non-texture cluster. Component 3 seems to separate by scale, whereas component 4 tells apart bars from corners and curves

Then, the projection of the data onto the first two principal components is presented in Figure 4.12b. The size of a point reflects the accuracy of our model for the neuron in terms of correlation on the test set, and the top 20% neurons are identified. The neurons cluster roughly into two main groups: those selective for texture (right) and those selective for non-texture. Texture neurons display a strong positive projection on the first component, and a weak projection on the second component, implying they do not have a horizontal/vertical preference. We visualize the high-scoring neurons found in both clusters by showing one excitatory image for each neuron.

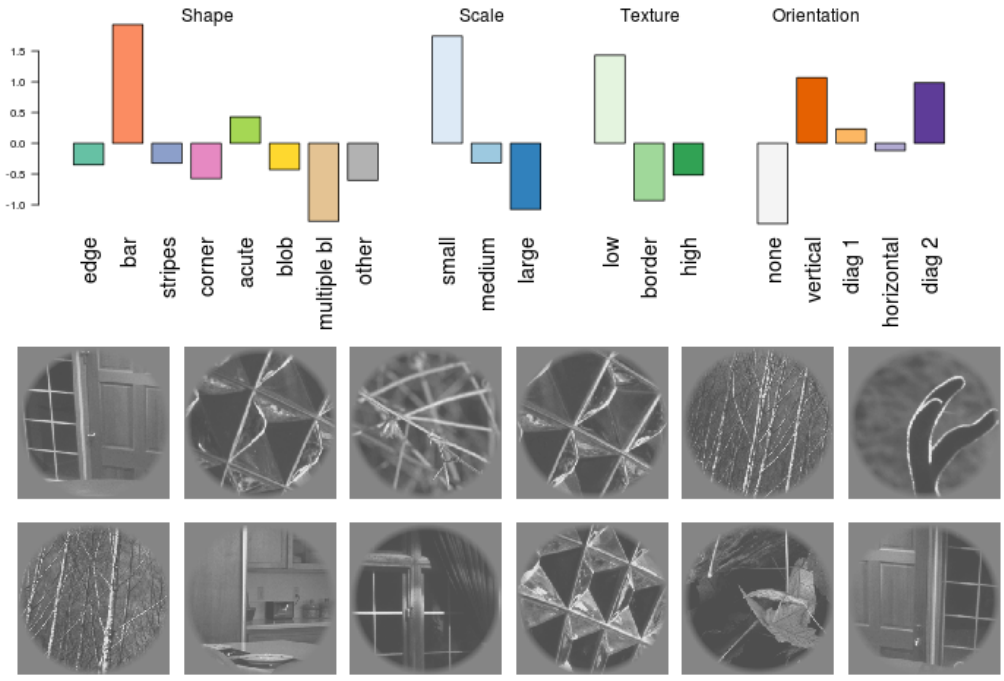


(a) Visualization of a neuron responding to corners.

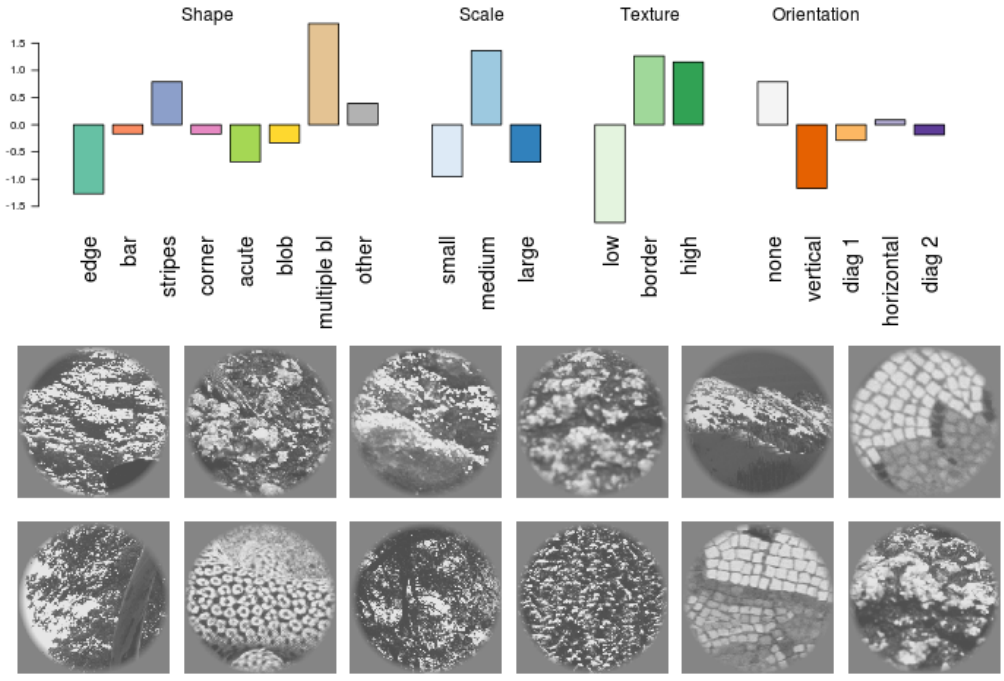


(b) Visualization of a neuron responding to horizontal features.

Figure 4.10 – Visualization of two neurons. For each one, the top part shows the impact values in terms of shape, scale, texture, and orientation. Below, the most excitatory images are displayed.

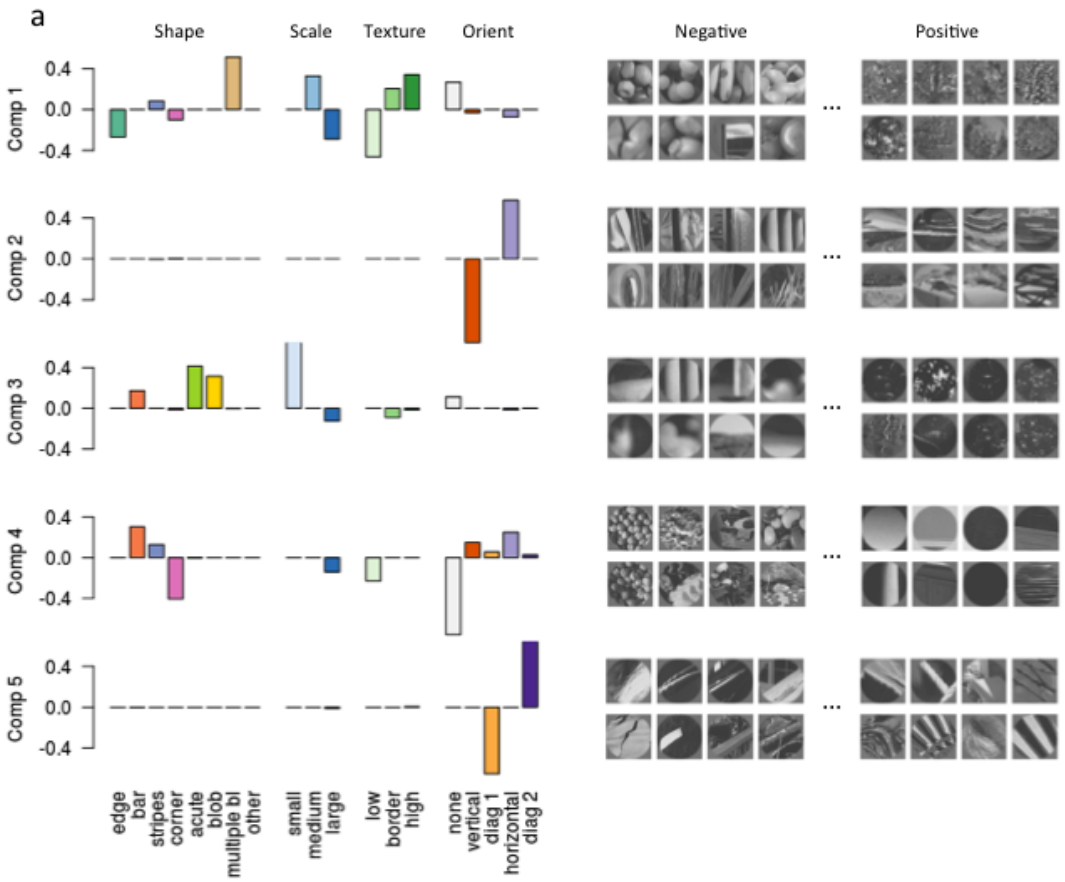


(a) Visualization of one neuron responding to bars.

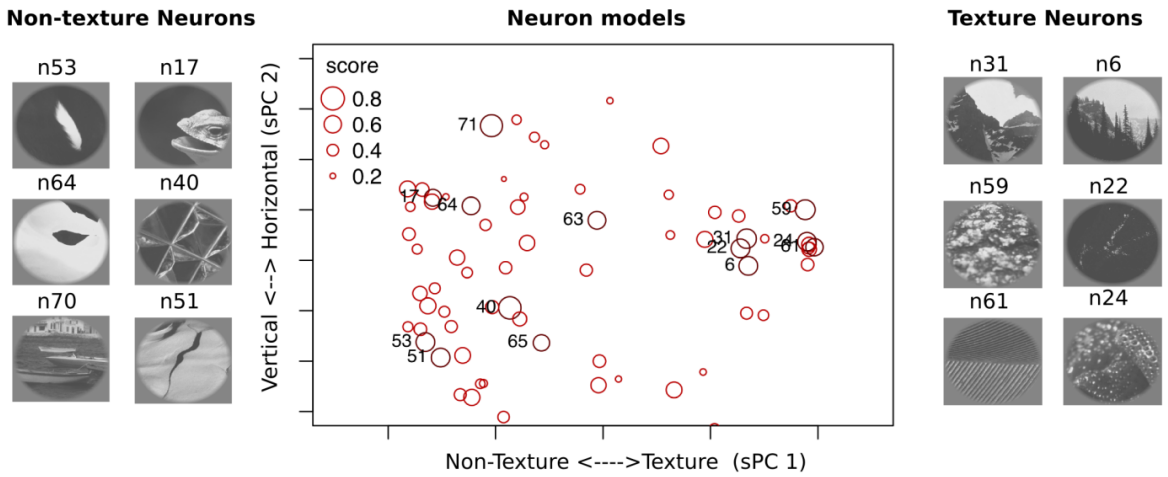


(b) Visualization of one neuron responding to textures.

Figure 4.11 – Visualization of two neurons. For each one, the top part shows the impact values in terms of shape, scale, texture, and orientation. Below, the most excitatory images are displayed.



(a) Visualization of the first five sparse principal components.



(b) Population projected on the two first principal components. The size of the circles is proportional to the prediction accuracy on the test set, in terms of correlation.

Figure 4.12 – Population analysis of the neuron using sparse principal component analysis.

Chapter 5

Discussions and Concluding Remarks

5.1 Discussions

There are many approaches to machine learning or schools of thought. Several of them are present in this dissertation, whereas some others are not considered, such as Bayesian statistics. Below, we briefly discuss three classical antagonisms between different point of views, which are respectively related to the three main chapters of this thesis.

Frequentists vs. Bayes. In this thesis, we do not consider Bayesian inference, which consists of modeling every variable of a given problem with a probability distribution. Even though it is often possible to give a probabilistic interpretation to the minimization problems addressed in Chapter 2—as maximum a posteriori estimation—Bayesian inference goes beyond finding a "true" prediction function $f^\star$ that maps a label y to a given observation $\mathbf{x}$. Instead, it directly models the conditional probability distribution $p(y|\mathbf{x})$, which may be used to do prediction, but also to quantify the uncertainty of the prediction, whereas uncertainty estimation is often performed a posteriori in frequentists contexts.

The goal of Bayesian inference seems thus more ambitious than the frequentist point of view we adopt. Nevertheless, this ambitious goal is also much more difficult to achieve, and today the hundred-year-old debate between frequentists and Bayesian statisticians is still unsettled. Instead, these two approaches should probably not be seen as antagonists. Bayesian inference is often intractable and requires approximations, for which efficient optimization techniques originally designed in a frequentist context, such as those presented in Chapter 2, may prove useful. Conversely, Bayesian models have shown to be useful for learning hyper-parameters of deep (frequentist) networks (Snoek et al., 2012).

Neural networks vs. kernels methods. Chapter 3 may be seen as an interpolation between kernel methods and deep neural networks. Specifically, convolutional kernel networks can be used in two different ways. The first one consists of defining a positive definite kernel for images and approximate it using a simple unsupervised subspace learning procedure. This approach is in the same spirit as classical kernel methods, where data representation and supervised learning are decoupled, and the RKHS norm is used

as a natural regularization function. Then, when subspaces of the RKHSs are learned with supervision, the approach resembles classical convolutional kernel networks, where data representation (meaning here subspace learning) is coupled with the learning task. Continuously interpolating between the two regimes, *e.g.*, for semi-supervised learning, is feasible in principle, even though we have not tested this idea yet.

It is then natural to ask what properties of kernels methods are useful for neural networks and vice versa. Chapter 3 does not provide a detailed answer to this question, but it opens several perspectives (see also next section). On the one hand, principles from convolutional neural networks have proven there to be useful for designing a kernel that models the local stationarity and invariance of natural images at several scales. Conversely, the kernel point of view provides a new type of convolutional neural network with a geometric interpretation, where each layer performs a projection on a linear subspace of an RKHS. We argue that the geometric interpretation is important. First, it allows to treat the unsupervised and supervised regimes in the same fashion, by aligning subspaces with data in the former case, and via backpropagation in the latter; second, each quantity computed by kernel networks can be mapped to points of functional spaces that we should be able to subsequently study; finally, it may allow us to explore many new variants, *e.g.*, involving a structured collection of subspaces at every layer as in sparse coding models, which are not natural from the classical neural network point of view.

Laplace vs. Gauss and Hilbert. Chapter 4 is devoted to sparse estimation involving the ℓ_1-regularization (which may be interpreted as a Laplace prior) in a pluri-disciplinary context. This is in contrast to other regularization functions used in the previous chapters, which were essentially the squared-ℓ_2-norm (Gaussian prior), or an Hilbertian norm as in Chapter 3. [1] Preferring one regularization instead of another is essentially a question of a priori knowledge we want to encode in a problem solution. The ℓ_2-norm leads to stable and smooth solutions, and adds strong convexity to the objective function, which makes the resulting optimization problems easier to solve. One the other hand, the ℓ_1-norm is able to find a sparse solution, which may be a key to obtain simple interpretable models that will please experimental scientists. Yet, sparse estimation may also lead to unstable solutions, whose sparsity pattern may change after small perturbations of the input data (Bach, 2008a; Meinshausen and Bühlmann, 2010).

5.2 New perspectives and future research.

We now conclude with a few future research directions.

Perspectives in optimization for machine learning. The most recent and promising direction is related to the acceleration of optimization techniques, which we have started investigating with the Catalyst (Lin et al., 2015) and QuickeNing (Lin et al.,

1. Of course, the contemporary question of which regularization function should be used for statistical estimation has nothing to do with the mathematical problems studied by Laplace, Gauss, and Hilbert.

2017) approaches. In both cases, we have studied acceleration for convex optimization problems, using Nesterov's and Quasi-Newton principles, respectively.

A first effort along this line of research was started recently during a collaboration between Inria (Hongzhou Lin, myself) and the University of Washington (Courtney Paquette, Dimitry Drusvyatskiy and Zaid Harchaoui), to adapt the Catalyst acceleration to non-convex optimization problems (see Paquette et al., 2017). In addition to new theoretical results allowing to apply Nesterov's acceleration in a non-convex context, the algorithm we introduce shows promising results for practical non-convex problems such as sparse matrix factorization and simple neural networks.

A second effort will be conducted to extend the acceleration schemes to stochastic objectives, since the original methods require deterministic problems. This is a necessary step to deploy this successful idea to deep neural networks, which are typically trained with stochastic gradient descent. Finally, we believe that these two-loop algorithms are also promising for distributed computing, where at every iteration of the outer loop, a collection of sub-problems will be solved simultaneously on different computing nodes.

Perspectives with deep kernel machines. There are two aspects, a theoretical one and a practical one, which we would like to develop regarding deep kernel machines.

From the theoretical side, we would like to study the convolutional kernel for images, which we present in Chapter 3. Notably, we would like to characterize the invariant properties of the kernel. This could be achieved by looking for two results about convolutional multilayer kernels: we want first the RKHS mapping to be invertible—that is, no information should be lost by using the kernel representation. Second, we want the representation to be robust to local deformations. It is known that such invariant properties can be obtained using signal processing tools (Bruna and Mallat, 2013), and whether or not similar results can be achieved using kernel methods remain an open question today. [2]

Then, we will focus on developing deep kernel machines for structured data such as graph and sequences, which may be useful for example in bioinformatics for representing proteins (Borgwardt and Kriegel, 2005), in chemoinformatics for molecules, or in computer vision for representing videos.

Perspectives in sparse estimation. Even though my research interests have shifted away from sparse estimation, an important challenge I would still like to address is that of stable feature selection, which is typically addressed via bootstrap (Bach, 2008a) or resampling (Meinshausen and Bühlmann, 2010). While these approaches are satisfactory in terms of model selection consistency, they require solving a large number of optimization problems for estimating reliably the support of a single solution, and are intrinsically computationally expensive. Instead, a promising direction consists of solving a single stochastic optimization problem involving data perturbations for each training point. The idea is to estimate a solution whose sparsity pattern is robust (and thus stable) to these perturbations. It turns out that we now how to solve such problems efficiently

2. At the time of the defense of this HdR (several month after this paragraph was written), various results have been obtained to characterize the invariance of stability of deep kernel representations in (Bietti and Mairal, 2017a), providing a positive answer to the question raised above.

(Bietti and Mairal, 2017b), and it would be of utmost interest to perform the missing statistical analysis. We believe that such an approach could potentially make these stable feature selection approaches affordable from a computational point of view.

Bibliography

A. Agarwal and L. Bottou. A lower bound for the optimization of finite sums. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11): 4311–4322, 2006.

S. Ahn, J. A. Fessler, D. Blatt, and A. O. Hero. Convergent incremental optimization transfer algorithms: Application to tomography. *IEEE Transactions on Medical Imaging*, 25(3):283–296, 2006.

R. K. Ahuja, T. L. Magnanti, and J. Orlin. *Network Flows*. Prentice Hall, 1993.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, pages 267–281, 1973.

F. Anselmi, L. Rosasco, C. Tan, and T. Poggio. Deep convolutional networks are hierarchical kernel machines. *preprint arXiv:1508.01084*, 2015.

F. Bach. Bolasso: model consistent lasso estimation through the bootstrap. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008a.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008b.

F. Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research (JMLR)*, 18(19):1–53, 2017.

F. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

E. Bernard, L. Jacob, J. Mairal, and J.-P. Vert. Efficient RNA isoform identification and quantification from RNA-Seq data with network flows. *Bioinformatics*, 30(17): 2447–2455, 2014.

E. Bernard, L. Jacob, J. Mairal, E. Viara, and J.-P. Vert. A convex formulation for joint rna isoform detection and quantification from multiple rna-seq samples. *BMC Bioinformatics*, 16(1):262, 2015.

D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999. 2nd edition.

A. Bietti and J. Mairal. Group invariance and stability to deformations of deep convolutional representations. *preprint arXiv:1706.03078*, 2017a.

A. Bietti and J. Mairal. Stochastic optimization with variance reduction for infinite datasets with finite-sum structure. *preprint arXiv:1610.00970*, 2017b.

D. Blatt, A. O. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

D. Böhning and B. G. Lindsay. Monotonicity of quadratic-approximation algorithms. *Annals of the Institute of Statistical Mathematics*, 40(4):641–663, 1988.

K. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *IEEE International Conference on Data Mining (ICDM)*, 2005.

J. M. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: Theory and examples*. Springer, 2006.

L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning and Neural Networks*. 1998.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *arXiv:1606:04838*, 2016.

O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(12):1222–1239, 2001.

D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *SIAM Journal on Multiscale Modeling and Simulation*, 4(2):490–530, 2005.

R. H. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for L-1 regularized optimization. *Mathematical Programming*, 157(2):375–396, 2015.

E. J. Candès, M. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

O. Cappé and E. Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B*, 71(3):593–613, 2009.

C. M. Carvalho, J. Chang, J. E. Lucas, J. R. Nevins, Q. Wang, and M. West. High-dimensional sparse factor modeling: Applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456, 2008.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1999.

X. Chen and M. Fukushima. Proximal quasi-Newton methods for nondifferentiable convex optimization. *Mathematical Programming*, 85(2):313–334, 1999.

Y. Cho and L. K. Saul. Large-margin classification in infinite neural networks. *Neural Computation*, 22(10):2678–2697, 2010.

A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surface of multilayer networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

J. F. Claerbout and F. Muir. Robust modeling with erratic data. *Geophysics*, 38(5): 826–844, 1973.

M. Collins, R. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1):253–285, 2002.

M. Cuturi and J.-P. Vert. The context-tree kernel for strings. *Neural Networks*, 18(8): 1111–1123, 2005.

M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

A. Damianou and N. Lawrence. Deep Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

A. Daniely, R. Frostig, and Y. Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. Wiley, New York, 1951.

I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

S. V. David and J. L. Gallant. Predicting neuronal responses during natural vision. *Network: Computation in Neural Systems*, 16(2-3):239–260, 2005.

A. J. Defazio, T. S. Caetano, and J. Domke. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014a.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014b.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Duality and auxiliary functions for Bregman distances. Technical report, CMU-CS-01-109, 2001.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1): 1–38, 1977.

O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.

C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(2):295–307, 2016.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research (JMLR)*, 10:2899–2934, 2009.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

M. A. Efroymson. Multiple regression analysis. *Mathematical methods for digital computers*, 9(1):191–203, 1960.

M. Elad. *Sparse and Redundant Representations*. Springer, 2010.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

H. Erdogan and J. A. Fessler. Ordered subsets algorithms for transmission tomography. *Phys. Med. Biol.*, 44(11):2835–2851, 1999.

M. A. T. Figueiredo and R. D. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research (JMLR)*, 2:243–264, 2001.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer, 2001.

K. Fukushima. Neural network model for a mechanism of pattern recognition unaffected by shift in position - neocognitron. *Transactions IECE*, J62-A(10):658–665, 1979.

M. Fukushima and L. Qi. A globally and superlinearly convergent algorithm for nonsmooth convex minimization. *SIAM Journal on Optimization*, 6(4):1106–1120, 1996.

B. Gärtner, M. Jaggi, and C. Maria. An exponential lower bound on the complexity of regularization paths. *Journal of Computational Geometry (JoCG)*, 3(1):168–195, 2012.

G. Gasso, A. Rakotomamonjy, and S. Canu. Recovering sparse signals with non-convex penalties and DC programming. *IEEE Transactions on Signal Processing*, 57(12): 4686–4698, 2009.

S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. Technical report, 2013.

J. Giesen, M. Jaggi, and S. Laue. Approximating parameterized convex optimization problems. In *Algorithms - ESA*, Lectures Notes Comp. Sci. 2010.

A. V. Goldberg. An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In *Proc. of ACM Symposium on Theory of Computing*, pages 136–146, 1986.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

A. Gordo, J. Almazan, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

R. M. Gower, D. Goldfarb, and P. Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Y. Grandvalet and S. Canu. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, 2014.

O. Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

Z. Harchaoui, A. Juditsky, and A. Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1-2):75–112, 2015.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research (JMLR)*, 5:1391–1415, 2004.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

T. Hazan and T. Jaakkola. Steps toward deep kernel methods from infinite neural networks. *preprint arXiv:1508.05133*, 2015.

S. Heber et al. Splicing graphs and EST assembly problem. *Bioinformatics*, 18(suppl 1): S181–S188, 2002.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I.* Springer, 1996.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

R. R. Hocking. A Biometrics invited paper. The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49, 1976.

R. Horst and N. V. Thoai. DC programming: overview. *Journal of Optimization Theory and Applications*, 103(1):1–43, 1999.

J. Huang, Z. Zhang, and D. Metaxas. Learning with structured sparsity. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

A. Hyvärinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.* Springer, 2009.

L. Jacob, G. Obozinski, and J.-P. Vert. Group Lasso with overlap and graph Lasso. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

M. Jaggi. *Sparse Convex Optimization Methods for Machine Learning.* PhD thesis, ETH Zürich, 2011.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

T. Jebara and A. Choromanska. Majorization for CRFs and latent likelihoods. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research (JMLR)*, 12:2777–2824, 2011a.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research (JMLR)*, 12:2297–2334, 2011b.

H. Jiang and W. H. Wong. Statistical inferences for isoform expression in RNA-Seq. *Bioinformatics*, 25(8):1026–1032, 2009.

A. Juditsky and A. Nemirovski. First order methods for nonsmooth convex large-scale optimization. In *Optimization for Machine Learning*. MIT Press, 2011.

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

E. Khan, B. Marlin, G. Bouchard, and K. Murphy. Variational bounds for mixed-data factor analysis. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, 1972.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397, 2012.

G. Lan and Y. Zhou. An optimal randomized incremental gradient method. *Mathematical Programming, Serie A*, 2017.

K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics*, 9(1):1–20, 2000.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research (JMLR)*, 10:777–801, 2009.

Q. Le, T. Sarlós, and A. Smola. Fastfood–approximating kernel expansions in loglinear time. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

N. Le Roux and Y. Bengio. Continuous neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. 1998.

C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

C.-Y. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

J. Lee, Y. Sun, and M. Saunders. Proximal Newton-type methods for convex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 836–844, 2012.

C. Lemaréchal and C. Sagastizábal. Practical aspects of the Moreau–Yosida regularization: Theoretical preliminaries. *SIAM Journal on Optimization*, 7(2):367–385, 1997.

C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific symposium on biocomputing*, volume 7, pages 566–575, 2002.

W. Li et al. IsoLasso: a LASSO regression approach to RNA-Seq based transcriptome assembly. *Journal of Computational Biology*, 18:1693–1707, 2011.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

H. Lin, J. Mairal, and Z. Harchaoui. Quickening: A generic Quasi-Newton algorithm for faster gradient-based optimization. *preprint arXiv:1610.00960*, 2017.

M. Lin, Q. Chen, and S. Yan. Network in network. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2013a.

J. Mairal. Optimization with first-order surrogate functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013b.

J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

J. Mairal and B. Yu. Complexity analysis of the Lasso regularization path. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008a.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008b.

J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008c.

J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008d.

J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Modeling and Simulation*, 7(1):214–241, 2008e.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009a.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009b.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research (JMLR)*, 11:19–60, 2010a.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *Advances in Neural Information Processing Systems (NIPS)*, 2010b.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Convex and network flow optimization for structured sparsity. *Journal of Machine Learning Research (JMLR)*, 12:2681–2720, 2011.

J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(4):791–804, 2012.

J. Mairal, Y. Benjamini, B. D. Willmore, M. Oliver, J. L. Gallant, and B. Yu. Modeling V4 under naturalistic conditions with invariant image representations, 2013. unpublished.

J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Vision and Graphics*, 2014a.

J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

J. Mairal and B. Yu. Supervised feature selection in graphs with path coding penalties and network flows. *Journal of Machine Learning Research (JMLR)*, 14(1):2449–2485, 2013.

J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2–3):85–283, 2014b.

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014c.

S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 11 (7):674–693, 1989.

C. L. Mallows. Choosing variables in a linear regression: A graphical aid. unpublished paper presented at the Central Regional Meeting of the Institute of Mathematical Statistics, Manhattan, Kansas, 1964.

C. L. Mallows. Choosing a subset regression. unpublished paper presented at the Joint Statistical Meeting, Los Angeles, California, 1966.

H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

J. A. Mazer, W. Vinje, J. McDermott, P. Schiller, and J. Gallant. Spatial frequency and orientation tuning dynamics in area v1. *Proceedings of the National Academy of Sciences USA*, pages 1645–1650, 2002.

N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux. Dictionary learning for massive matrix factorization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux. Stochastic subsampling for factorizing huge matrices. *to appear in IEEE Transactions on Signal Processing*, 2017.

R. Mifflin. A quasi-second-order proximal bundle algorithm. *Mathematical Programming*, 73(1):51–72, 1996.

G. Montavon, M. L. Braun, and K.-R. Müller. Kernel analysis of deep networks. *Journal of Machine Learning Research (JMLR)*, 12:2563–2581, 2011.

R. M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, Dept. of Computer Science, University of Toronto, 1994.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 89, 1998.

A. Nelakanti, C. Archambeau, J. Mairal, F. Bach, and G. Bouchard. Structured penalties for log-linear language models. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

Y. Nesterov and B. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Y. Nesterov and J.-P. Vial. Confidence level solutions for stochastic programming. *Automatica*, 44(6):1559–1568, 2008.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

S. Nishimoto, A. T. Vu, T. Naselaris, Y. Benjamini, B. Yu, and J. L. Gallant. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 21(19):1641–1646, 2011.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

M. R. Osborne, B. Presnell, and B. A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.

P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

Q. Pan, O. Shai, L. J. Lee, B. J. Frey, and B. J. Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, 40(12):1413–1415, 2008.

C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui. 4wd-catalyst acceleration for gradient-based non-convex optimization. 2017. submitted.

M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid. Local convolutional features with unsupervised training for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, and C. Schmid. Convolutional patch representations for image retrieval: an unsupervised approach. *International Journal of Computer Vision (IJCV)*, 2016.

K. Popper. *Logik der Forschung. Zur Erkenntnistheorie der modernen Naturwissenschaft*. Payot, 1934. translated in French under the title "La logique de la découverte scientifique" in 1973.

F. Radenović, G. Tolias, and O. Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

M. Razaviyayn, M. Sanjabi, and Z.-Q. Luo. A stochastic successive minimization method for nonsmooth nonconvex optimization with applications to transceiver design in wireless communication networks. *Mathematical Programming*, 157(2):515–545, 2016.

P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2): 1–38, 2014.

K. Ritter. Ein verfahren zur lösung parameterabhängiger, nichtlinearer maximumprobleme. *Math. Method Oper. Res.*, 6(4):149–166, 1962.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

A. W. Roe, L. Chelazzi, C. E. Connor, B. R. Conway, I. Fujita, J. L. Gallant, H. Lu, and W. Vanduffel. Toward a unified theory of visual area V4. *Neuron*, 74(1):12–29, 2012.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *Annals of Statistics*, 35(3):1012–1030, 2007.

S. Salzo and S. Villa. Inexact and accelerated proximal point algorithms. *Journal of Convex Analysis*, 19(4):1167–1192, 2012.

K. Scheinberg and X. Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. *Mathematical Programming*, pages 1–35, 2014.

M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2016.

B. Schölkopf. *Support Vector Learning*. PhD thesis, Technischen Universität Berlin, 1997.

B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research (JMLR)*, 14:567–599, 2013.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1):105–145, 2016.

S. Shalev-Shwartz. SDCA without Duality, Regularization, and Individual Convexity. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

J. Shawe-Taylor and N. Cristianini. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2004.

P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent prop–a formalism for specifying selected invariances in an adaptive network. In *Advances in Neural Information Processing Systems (NIPS)*, 1992.

P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri. Transformation Invariance in Pattern Recognition — Tangent Distance and Tangent Propagation. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, number 1524 in Lecture Notes in Computer Science, pages 239–274. Springer Berlin Heidelberg, 1998.

E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multiscale transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, 1992.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

S. Smale, L. Rosasco, J. Bouvrie, A. Caponnetto, and T. Poggio. Mathematics of the neural response. *Foundations of Computational Mathematics*, 10(1):67–91, 2010.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

P. Smolensky. Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chapter information processing in dynamical systems: foundations of harmony theory. In *MIT Press*, pages 194–281. 1986.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research (JMLR)*, 7:1531–1565, 2006.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

I. Steinwart, P. Thomann, and N. Schmid. Learning with hierarchical gaussian kernels. *preprint arXiv:1612.00824*, 2016.

V. Sydorov, M. Sakurada, and C. Lampert. Deep Fisher kernels — end to end learning of the Fisher kernel GMM parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

M. Szafranski, Y. Grandvalet, and P. Morizet-Mahoudeaux. Hierarchical penalization. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

M. Szafranski, Y. Grandvalet, and A. Rakotomamonjy. Composite kernel learning. *Machine learning*, 79(1-2):73–103, 2010.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

A. M. Tillmann, Y. C. Eldar, and J. Mairal. Dolphin—dictionary learning for phase retrieval. *IEEE Transactions on Signal Processing*, 64(24):6485–6500, 2016.

R. Timofte, V. Smet, and L. van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

C. Trapnell et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

B. A. Turlach, W. N. Venables, and S. J. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

V. Vapnik. *The nature of statistical learning theory*. Springer, 2000. second edition.

G. Varoquaux, A. Gramfort, F. Pedregosa, V. Michel, and B. Thirion. Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. In *Biennial International Conference on Information Processing in Medical Imaging*, 2011.

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(3):480–492, 2012.

S. Wager, W. Fithian, S. Wang, and P. Liang. Altitude Training: Strong Bounds for Single-layer Dropout. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015a.

Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015b.

B. Widrow and M. E. Hoff. Adaptive switching circuits. In *RE WESCON Convention Record*, volume 4, pages 96–104, 1960.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

B. D. Willmore, R. J. Prenger, and J. L. Gallant. Neural representation of natural images in visual area V2. *Journal of Neuroscience*, 30(6):2102–2114, 2010.

S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

D. Wrinch and H. Jeffreys. XLII. On certain fundamental principles of scientific inquiry. *Philosophical Magazine Series 6*, 42(249):369–390, 1921.

Z. Xia et al. NSMAP: a method for spliced isoforms identification and quantification from RNA-Seq. *BMC Bioinformatics*, 12:162, 2011.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research (JMLR)*, 11:2543–2596, 2010.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

F. Yger, M. Berar, G. Gasso, and A. Rakotomamonjy. A supervised strategy for deep kernel machine. In *Proceedings of ESANN*, 2011.

J. Yu, S. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to non-smooth convex optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68:49–67, 2006.

M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. 2010.

K. Zhang and J. T. Kwok. Clustered nyström method for large scale manifold learning and dimension reduction. *IEEE Transactions on Neural Networks*, 21(10):1576–1587, 2010.

Y. Zhang, P. Liang, and M. J. Wainwright. Convexified convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, 37(6A):3468–3497, 2009.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.