Learning to Integrate Data from Different Sources and Tasks

Andreas Argyriou

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London.

Department of Computer Science

University College London

October 2007

I, Andreas Argyriou, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Supervised learning aims at developing models with good generalization properties using input/output empirical data. Methods which use linear functions and especially kernel methods, such as ridge regression, support vector machines and logistic regression, have been extensively applied for this purpose. The first question we study deals with selecting kernels appropriate for a specific supervised task. To this end we formulate a methodology for learning combinations of prescribed basic kernels, which can be applied to a variety of kernel methods. Unlike previous approaches, it can address cases in which the set of basic kernels is infinite and even uncountable, like the set of all Gaussian kernels. We also propose an algorithm which is conceptually simple and is based on existing kernel methods. Secondly, we address the problem of learning common feature representations across multiple tasks. It has been empirically and theoretically shown that, when different tasks are related, it is possible to exploit task relatedness in order to improve prediction on each task – as opposed to treating each task in isolation. We propose a framework which is based on learning a common set of features jointly for a number of tasks. This framework favors sparse solutions, in the sense that only a small number of features are involved. We show that the problem can be reformulated as a convex one and can be solved with a conceptually simple alternating algorithm, which is guaranteed to converge to an optimal solution. Moreover, the formulation and algorithm we propose can be phrased in terms of kernels and hence can incorporate nonlinear feature maps. Finally, we connect the two main questions explored in this thesis by demonstrating the analogy between learning combinations of kernels and learning common feature representations across multiple tasks.

Acknowledgements

The following thesis would not have been a reality without my collaboration with several people, the feedback they have provided and their support. During its preparation, I have benefited greatly from interaction with my thesis supervisor, Massimiliano Pontil, from having collaborated with him and from his expertise in kernel learning methods. He has an eye for problems that are exciting and important and an appreciation for mathematical rigor. Equally importantly, he has always been accessible and willing to discuss even the least promising ideas and his critique, positive or negative, has been extremely valuable. Last but not least, he has cultivated an environment of top-rate collaborators who are an inexhaustible source of research ideas. Among these, I have been very fortunate to have worked with a mathematician of the stature of Charles A. Micchelli and an all-around scientist like Theodoros Evgeniou. I would like to thank them for their support in this and many other ways. I would also like to thank Raphael Hauser, Mark Herbster, Andreas Maurer and Yiming Ying, with whom it has been a joy to work and be friends.

Many thanks go to Mark Girolami and John Shawe-Taylor who have read the thesis thoroughly and provided excellent comments and suggestions. Also to Zoubin Ghahramani who has given feedback for early versions of my work. In addition, I have benefited from discussions with several people in the wider machine learning community and especially from the fruitful and exciting intellectual environment of UCL Computer Science and the Gatsby Institute. Finally, I would like to thank my family and my friends, in London and abroad, for providing a supportive environment but also for the intellectual exchange with them, in science and many other areas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A key goal in machine learning is to develop algorithms for estimating models with good generalization properties, based on empirical data. In particular, a large number of machine learning problems are *supervised* problems. Supervised learning is characterized by the availability of input/output data $\{(x_1, y_1), \ldots, (x_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$. The data is usually assumed to be drawn i.i.d. from a probability distribution on the input/output space $\mathcal{X} \times \mathcal{Y}$. For example, the inputs may be vectors in $\mathbb{R}^d$, such as those obtained from images, text, biological data etc. and the outputs may describe positive or negative $(+1, -1)$, an integer rating or a continuous quantity. The goal of a supervised learning algorithm is to select a prediction function $f : \mathcal{X} \to \mathcal{Y}$ from a function class $\mathcal{F}$ so that the expected error of predictions $f(x)$ over the whole input space is small. Moreover, depending on the type of output, supervised learning includes *classification*, *regression* and *ranking*. This distinction, however, will not be important in the treatment we follow in this thesis.

Our focus will be on algorithms for supervised tasks that use linear classes of functions. That is, the inputs are transformed through a *feature map* $\Phi : \mathcal{X} \to \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space, and the outputs are real numbers. The function learned should be of the form $f(x) = \langle w, \Phi(x) \rangle$, for some $w \in \mathcal{H}$. These methods allow for treating various forms of data as mathematically manageable quantities. More importantly, the use of a feature map which is appropriate for a particular task can lead to a predictor with greatly improved statistical accuracy, as opposed to a simple linear predictor. In this sense, the feature map can be viewed as a representation for the data. The availability of a good representation is clearly important for the success of the learning algorithm.

It is often the case, though, that the best feature maps are high or infinite-dimensional. Fortunately, the associated computational issues can be surpassed if the inner products of the form $\langle \Phi(x), \Phi(x') \rangle$, $x, x' \in \mathcal{X}$, are known – even without explicitly knowing the feature map. Learning algorithms that only involve such inner products have been studied extensively in recent years and are known as *kernel methods*. Such methods include ridge regression, support vector machines, logistic regression etc.

In this thesis, we are going to study a family of kernel methods in two different contexts. First, we will consider the question of how to select feature maps appropriate for a specific task. One approach is to *jointly select a number of feature maps* from a prescribed set. The second question is how to jointly learn a number of features common across *multiple tasks*. Moreover, we will connect these two questions towards the end of the thesis, where our proposed methods will be shown to be equivalent.

Throughout our discussion there is a common theme of learning by using multiple feature maps, either for single-task or for multi-task learning. Combining feature maps boosts performance in many cases, especially when each feature map expresses only some part of the useful representation. Moreover, an optimal feature map may be unknown or hard to obtain. Thus, frequently a set of feature maps is assumed a priori and the goal is to select the ones which together obtain the best results for the task at hand.

For example, in many biological applications such as protein classification, it is expensive to obtain certain types of information and the information available may refer to different characteristics of proteins. This makes it imperative to combine disparate types of information. Another example comes from computer vision, where one objective is to discriminate between images of two objects or characters. It is often hard to describe such images completely with a single feature map, even a complex one. Moreover, little may be known about the nature of a good representation so that we have to start with a large set of candidates.

In practice, combining feature maps is done implicitly and corresponds to combining kernels. In prior work it has been shown that concatenating feature maps can be phrased equivalently as combining the corresponding kernels in a convex way. This further favors a small number of feature maps in the solution through penalization with an L_1 type norm. As a consequence, in this thesis we will consider the problem of learning a supervised task using convex combinations of kernels.

This topic is addressed in Chapter 3, where we formulate a methodology for learning combinations of kernels that relates to recent research. Our framework has the advantage that it can be applied to a variety of learning methods with kernels. Unlike previous approaches, it can address cases in which the number of prescribed kernels is not finite. In this way, it is particularly suitable for learning *parameterized kernels*. We study the associated optimization problem and, among other results, show that the number of kernels in the solution is limited by the size of the data. Our theoretical study motivates an algorithm that can learn combinations of infinitely many parameterized kernels. This algorithm is conceptually simple and based on existing kernel methods such as SVMs. However, as we shall demonstrate, the problem is a hard one in many cases.

In Chapter 4, we describe our method for learning common features across multiple tasks. Multi-task learning has been motivated from situations in which only a relatively small amount of data is available for each of a number of tasks. The main insight is that, if the tasks are related in some way, then it may be possible to *exploit task relatedness* in order to improve performance on each task – as opposed to treating each task in isolation. Other terms, like “inductive transfer”, “transfer learning” or “learning to learn”, are also sometimes used to describe similar learning problems.

In nature, it is often the case that the same processes underly similar learning tasks. At an early age, humans learn how to recognize such analogies and then transfer knowledge from one task they have learned to a new one. For example, recognizing alphanumeric characters is essentially learned once and every time a new symbol is created or seen it is learned with no effort. Clearly, this happens because some general aspects of the character-generating process are learned by the mind, not just the specifics of recognizing a particular character. In other situations, the mind may develop a concept by learning a specialized task and then generalizing to more abstract concepts as more related tasks are encountered. Instances of this can be seen in the discovery of scientific laws and the educational or training processes. Similarly, learning some tasks benefits greatly from (or could be impossible without) pooling them together, as in recognizing an object from images taken under different conditions, pose, views etc.

Our working assumption will be that task relatedness is due to the existence of a *common representation* shared among the tasks. Clearly this assumption is more applicable in certain situations than others. We claim, however, that it leads naturally to a method which recognizes and exploits task similarities of this nature (if any exist). Our formulation is based on *learning* a common set of simple feature maps for all the tasks. We further wish to have *sparse* solutions in the sense that only few of the feature maps are involved in the solution. Sparsity conforms with the insight that we should find shared similarities only where they exist, in other words we should not overfit the data with a complicated representation.

The formulation and associated algorithm we propose are phrased in terms of kernels and hence can be extended to incorporate nonlinear feature maps. The main questions that will occupy us relate to the optimization and computational properties of our method. We shall show that the problem can be reformulated as a *convex* one and, consequently, does not suffer from problems of local optima. Moreover, it can be solved with a conceptually simple alternating algorithm which is guaranteed to converge to the optimal solution. Finally, experiments with synthetic and real data demonstrate that this algorithm indeed finds good common features and exploits them to obtain better statistical performance.

Chapter 2

Convex Analysis and Regularization in Hilbert Spaces

During the last decade, the use of regularization methods has become one of the main tools in machine learning research. The regularization framework covers a variety of popular methods in machine learning, such as support vector machines, ridge regression and logistic regression, as well as several methods in statistics, signal processing and other fields. In this thesis, we shall build on the regularization approach for addressing problems such as combining kernels and multi-task learning.

To this end, in this chapter we review some of the fundamentals of convex analysis, which will be a valuable mathematical tool throughout this thesis. These will allow posing and studying a number of optimization problems, as well as the algorithms for their solution. Moreover, we review some of the theory of L_2 regularization with reproducing kernels. In particular, we present the main results about optimal solutions of regularization problems and some fundamental theorems from the theory of reproducing kernels. We will also be interested in another category of regularization problems, which involve the spectra of matrices and will make their appearance in the context of multi-task learning. To this end, we review some results from matrix analysis.

2.1 Convex Sets and Functions

In this section we present the basics of convex analysis. The presentation is mainly based on [Borwein and Lewis, 2005] and [Rockafellar and Wets, 1998]. We use $\mathbb{R}^m$ to denote the set of vectors with m real components and $\mathbb{N}_m$ the set of integers $\{1, \dots\ m\}$, for every natural number m. We also use $\langle \cdot, \cdot \rangle$ to denote the standard inner product in $\mathbb{R}^m$ and $\mathbb{R}^{m \times s}$ to denote the set of $m \times s$ matrices with real elements.

2.1.1 Basic Definitions

A set is called *convex* if it includes every linear segment whose endpoints belong to the set.

Definition 1 *A subset $\mathcal{C}$ of $\mathbb{R}^m$ is convex if and only if*

$$\lambda c + (1 - \lambda) c' \in \mathcal{C} \qquad \forall \lambda \in [0, 1], c, c' \in \mathcal{C}.$$

The expression $\lambda c + (1-\lambda)c'$, whenever $\lambda \in [0,1]$, is called a *convex combination* of c and c'. More generally, $\sum_{i\in\mathbb{N}_n} \lambda_i c_i$ is called a convex combination of $c_1, \ldots, c_n$ if $\lambda_i \geq 0, \forall i \in \mathbb{N}_n$, and $\sum_{i\in\mathbb{N}_n} \lambda_i = 1$.

Definition 2 *The convex hull of a subset S of $\mathbb{R}^m$ is the smallest convex set that includes S. It is denoted by* $\operatorname{conv} S$.

Theorem 3 *The convex hull* $\operatorname{conv} S$ *of a set $S \subseteq \mathbb{R}^m$ consists of all the convex combinations of elements of S.*

Similarly, an *affine combination* is of the form $\tau c + (1-\tau)c'$ with no restriction on τ. Sets closed under affine combinations are called *affine sets*. Equivalently, an affine set is the intersection of a finite family of hyperplanes. Also, the *affine hull* $\operatorname{aff}S$ of a set S is the smallest affine set that includes S.

An important class of convex sets are *polyhedral sets*.

Definition 4 *A set $P \subseteq \mathbb{R}^m$ is called polyhedral if it is the intersection of a finite family of closed half-spaces or hyperplanes.*

Definition 5 *A set $P \subseteq \mathbb{R}^m$ is called a polytope if it is the convex hull of a finite subset of $\mathbb{R}^m$.*

In fact, bounded polyhedral sets and polytopes coincide. Another important class of convex sets are *convex cones*.

Definition 6 *A set $K \subseteq \mathbb{R}^m$ is called a cone if $0 \in K$ and $\lambda c \in K$ for all $\lambda \geq 0$ and $c \in K$.*

In other words, cones are unions of rays. Examples of convex cones are half-spaces of the form $\{\langle a, x\rangle \leq 0 : x \in \mathbb{R}^m\}$, the orthant $\mathbb{R}^m_+$ of vectors in $\mathbb{R}^m$ with nonnegative elements and the set $\mathbf{S}^m_+$ of $m \times m$ positive semidefinite matrices.

We now introduce *convex functions* taking values on the extended real line $\mathbb{R}\cup\{+\infty\}$. This convention is often followed in the optimization literature since it facilitates the inclusion of convex constraints. On the other hand it prohibits certain operations, such as subtraction.

Definition 7 *A function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is called convex if*

$$f(\lambda c + (1-\lambda)c') \leq \lambda f(c) + (1-\lambda) f(c') \qquad \forall \lambda \in [0,1], c, c' \in \mathbb{R}^m. \tag{2.1.1}$$

In particular, f is called *strictly convex* if the above inequality is strict whenever $c \neq c'$, $\lambda \in (0,1)$ and $f(c), f(c') \in \mathbb{R}$.

Definition 8 *A function $f : \mathbb{R}^m \to \mathbb{R} \cup \{-\infty\}$ is called* concave *if $-f$ is convex. It is called* strictly concave *if $-f$ is strictly convex.*

Well known examples of convex functions include the quadratic forms $f(c) = \langle c, Ac\rangle$ whenever A is an $m \times m$ positive semidefinite matrix (which are strictly convex for A positive definite). Affine functions $f(c) = Ac + b, A \in \mathbb{R}^{s\times m}, b \in \mathbb{R}^s$, are the only finite functions on $\mathbb{R}^m$ which are both convex and concave. The exponential function on the reals and the function $f(c) = \begin{cases} -\log c & c > 0 \\ +\infty & c \leq 0 \end{cases}$ are convex. Also, any *norm* on $\mathbb{R}^m$ is convex.

We call the set in which f takes finite values the *domain* of f.

Definition 9 *The domain of a convex function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, denoted by* $\operatorname{dom} f$*, is the set*

$$\{c \in \mathbb{R}^m : f(c) \in \mathbb{R}\} .$$

Clearly, the domain of a convex function f is a convex set. Thus, any real-valued function, defined on a convex subset $\mathcal{C}$ of $\mathbb{R}^m$ and satisfying inequalities (2.1.1), can be thought of as a convex function on the whole of $\mathbb{R}^m$ with domain equal to $\mathcal{C}$, by setting its value to $+\infty$ outside $\mathcal{C}$. However, we will also call such functions *convex on* $\mathcal{C}$, for brevity.

Not only the domain but all the level sets of a convex function are convex.

Theorem 10 *If $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is convex then all sets of the type*

$$\{c \in \mathbb{R}^m : f(c) \leq a\} \quad \textit{or} \quad \{c \in \mathbb{R}^m : f(c) < a\} ,$$

with $a \in \mathbb{R}$, are convex.

Convexity is preserved under some common operations.

Theorem 11

(a) The intersection of a family of convex sets is convex.

(b) The supremum of a family of convex functions is convex.

(c) The supremum of a finite *family of strictly convex functions is strictly convex.*

Theorem 12 *Let $f_i : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be convex functions for all $i \in \mathbb{N}_n$. Then for any real coefficients $\alpha_i \geq 0, i \in \mathbb{N}_n$, the function $\sum_{i \in \mathbb{N}_n} \alpha_i f_i$ is convex. Moreover, it is strictly convex if there exists $i \in \mathbb{N}_n$ such that $\alpha_i > 0$ and f_i strictly convex.*

Theorem 13 (Separability) *Let $f_i : \mathbb{R}^{m_i} \to \mathbb{R} \cup \{+\infty\}$ be convex functions for all $i \in \mathbb{N}_n$ and let $m = \sum_{i \in \mathbb{N}_n} m_i$. Then the function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ defined as $f(c) = \sum_{i \in \mathbb{N}_n} f_i(c_i)$, where $c_i \in \mathbb{R}^{m_i}$ and c is the concatenation of the c_i, is convex. If all the f_i are strictly convex then f is strictly convex as well.*

Theorem 14 (Composition with an affine map) *Let $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be a convex function and $A \in \mathbb{R}^{m \times s}, a \in \mathbb{R}^m$. Then the function $f : \mathbb{R}^s \to \mathbb{R} \cup \{+\infty\}$ defined as $f(c) = g(Ac + a), \forall c \in \mathbb{R}^s$, is convex.*

Theorem 15 (Partial infimum) *Let $g : \mathbb{R}^m \times \mathbb{R}^s \to \mathbb{R} \cup \{+\infty\}$ be convex. Then the function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ defined as $f(c) = \inf\{g(c, u) : u \in \mathbb{R}^s\}, \forall c \in \mathbb{R}^m$, is convex.*

Finally, there is a simple test for convexity when the function is twice differentiable.

Theorem 16 *Assume that $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is twice differentiable on an open convex set $\mathcal{O} \subseteq \mathbb{R}^m$. Then f is convex on $\mathcal{O}$ if and only if its Hessian matrix is positive semidefinite. Moreover, positive definiteness is sufficient (but not necessary) for strict convexity.*

2.1.2 Closure and Continuity

A concept appearing as a technical detail in many statements in convex analysis is that of *closure of a function*.

Definition 17 *The function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is lower semicontinuous at $c \in \mathbb{R}^m$ if*

$$\liminf_{c' \to c} f(c') = f(c).$$

Equivalently, the level sets $\{c \in \mathbb{R}^m : f(c) \leq a\}$ are closed for every $a \in \mathbb{R}^m$.

For every function, there is a greatest minorizing function which is lower semicontinuous.

Definition 18 *The* lower closure *of a function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, denoted by $\operatorname{cl} f$, is defined as*

$$(\operatorname{cl} f)(c) = \liminf_{c' \to c} f(c') \qquad\qquad \forall c \in \mathbb{R}^m.$$

A few useful facts about closures are the following.

Theorem 19

(a) For any function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, $\operatorname{cl} f \leq f$.

(b) For any functions $f_1, f_2 : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, $f_1 \leq f_2$ (uniformly) implies $\operatorname{cl} f_1 \leq \operatorname{cl} f_2$.

Theorem 20 *Let $\mathcal{T}$ be a set and $f_t : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be convex functions for every $t \in \mathcal{T}$. Then*

$$\operatorname{cl}(\sup\{f_t : t \in \mathcal{T}\}) = \sup\{\operatorname{cl} f_t : t \in \mathcal{T}\}\,.$$

Theorem 21 *The lower closure of a convex function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is convex.*

In general, convex functions are not lower semicontinuous. However, inside the interior of their domain they are continuous.

Theorem 22 *A convex function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is continuous on $\operatorname{int}(\operatorname{dom} f)$.*

In fact, f is locally Lipschitz inside $\operatorname{int}(\operatorname{dom} f)$.

2.1.3 Separation and Representations

Another fundamental property of convex sets which underlies much of convex analysis is *separation*. There are several separation theorems but here we state one special case which we will use later.

Theorem 23 (Strict separation) *Let $\mathcal{C} \subseteq \mathbb{R}^m$ be a* closed *convex set and $z \in \mathbb{R}^m$ be a point that does not belong to $\mathcal{C}$. Then z can be* strictly *separated from $\mathcal{C}$, that is, there exist $a \in \mathbb{R}^m, b \in \mathbb{R}$ such that $\langle z, a\rangle + b > 0$ and $\langle c, a\rangle + b \leq 0\,,\ \forall c \in \mathcal{C}$.*

An important basic fact that will be useful in Section 3.1 is that any point in a convex set has a representation of limited size.

Theorem 24 (Carathéodory) *Let $S \subseteq \mathbb{R}^m$ be a nonempty set. Then every point in $\operatorname{conv} S$ can be expressed as a convex combination of at most $m+1$ points of S.*

A consequence is that compact convex sets can be represented using only extreme points.

Theorem 25 *The convex hull of a compact set $S \subseteq \mathbb{R}^m$ is compact. In particular, the convex hull of a finite set is compact.*

An *extreme point* of a convex set $\mathcal{C} \subseteq \mathbb{R}^m$ is a point $c \in \mathcal{C}$ whose complement $\mathcal{C} \setminus \{c\}$ is convex.

Theorem 26 (Minkowski) *Any compact convex set $\mathcal{C} \subseteq \mathbb{R}^m$ is the convex hull of its extreme points.*

2.1.4 Directional Derivatives and Subgradients

It is often the case that we have to deal with convex functions which are not everywhere differentiable, regardless of whether they are finite or not. This will be true for some important convex functions appearing in Section 3.1. On the other hand, we need first order conditions that characterize the solutions of minimization problems. This has led to several generalizations of derivatives for nondifferentiable functions. Here we follow one common such approach, based on the right directional derivative.

Definition 27 *The* directional derivative *of a convex function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ at $c \in \operatorname{dom} f$ in a direction $\delta \in \mathbb{R}^m$ is defined as*

$$f'(c;\delta) = \lim_{t \to 0^+} \frac{f(c+t\delta) - f(c)}{t}\,.$$

Theorem 28 *Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be a convex function and let $c \in \operatorname{dom} f$. The directional derivative $f'(c;\cdot)$ exists everywhere and takes values in $\mathbb{R} \cup \{-\infty, +\infty\}$. Moreover, it is positively homogeneous.*

Theorem 29 *Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be a convex function and let $c \in \operatorname{int}(\operatorname{dom} f)$. The directional derivative $f'(c;\cdot)$ is everywhere finite and sublinear (and hence convex). In particular, f is differentiable if and only if $f'(c;\cdot)$ is linear.*

A sublinear function is one that is both positively homogeneous, $f(\alpha c) = \alpha f(c), \forall c \in \mathbb{R}^m, \alpha \geq 0$, and subadditive, $f(c + c') \leq f(c) + f(c'), \forall c, c' \in \mathbb{R}^m$.

Now, let us generalize the concept of gradient by allowing more than one *subgradients* of the function at a point.

Definition 30 *A subgradient of a convex function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ at $c \in \mathbb{R}^m$ is any vector $g \in \mathbb{R}^m$ satisfying the inequalities*

$$\langle g, c' - c \rangle \leq f(c') - f(c) \qquad \forall c' \in \mathbb{R}^m.$$

Definition 31 *The* subdifferential *of a convex function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ at $c \in \mathbb{R}^m$, denoted by $\partial f(c)$, is the set of subgradients of f at c. In particular, if $c \notin \operatorname{dom} f$ then $\partial f(c) = \emptyset$.*

Geometrically, subgradients correspond to hyperplanes that pass through point c but leave the graph of the function in the same half-space. It is easy to see that the subdifferential is always a closed convex set. Moreover, at points $c \in \operatorname{int}(\operatorname{dom} f)$ where f is differentiable, it consists of just one element, the gradient $\nabla f(c)$.

It is easy to obtain an equivalent characterization of subgradients, using directional derivatives.

Theorem 32 *Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be a convex function and let $c \in \operatorname{dom} f$. Then $g \in \partial f(c)$ if and only if*

$$\langle g, \delta \rangle \leq f'(c;\delta) \qquad \forall \delta \in \mathbb{R}^m .$$

These inequalities are sharp in the following sense.

Theorem 33 (Max formula) *Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be a convex function and let $c \in \operatorname{dom} f$. Then*

$$\operatorname{cl}(f'(c;\cdot)) = \sup\{\langle g, \cdot \rangle : g \in \partial f(c)\} .$$

Note that we use the convention $\sup \emptyset = -\infty$.

The most useful property of subgradients is the first order condition for optimality.

Theorem 34 *Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be a convex function and let $c \in \operatorname{dom} f$. Then c is a minimizer of f if and only if $0 \in \partial f(c)$. Equivalently this condition can be written as*

$$f'(c;\delta) \geq 0 \qquad \forall \delta \in \mathbb{R}^m .$$

2.1.5 Max-Functions

We have already mentioned the convexity of a supremum of convex functions. This type of functions occurs very frequently in optimization problems, for example in minimax problems. There are important rules for directional derivatives and the subdifferentials of such functions and they appear in different variants. Here, we take special care to allow for infinite families of convex functions.

The first fact concerns differentiation of a max-function. It is a slight variation of a result from [Micchelli and Pontil, 2005]. See also [Rockafellar and Wets, 1998, 8.31,10.31], [Borwein and Lewis, 2005, Sec. 2.3] for smooth versions.

Theorem 35 *Let $\mathcal{T}$ be a compact set, $\mathcal{C}$ a convex subset of $\mathbb{R}^m$ and $f_t : \mathcal{C} \to \mathbb{R}$ a convex function for every $t \in \mathcal{T}$. Assume also that, for every $c \in \mathcal{C}$, the function $t \mapsto f_t(c)$ is continuous on $\mathcal{T}$. We define the convex function $f : \mathcal{C} \to \mathbb{R}$ as*

$$f(c) := \max\{f_t(c) : t \in \mathcal{T}\} \qquad \forall c \in \mathcal{C}$$

and the sets

$$M(c) := \{t \in \mathcal{T} : f_t(c) = f(c)\} \qquad \forall c \in \mathcal{C}.$$

Then the right derivative of f in the direction $\delta \in \mathbb{R}^m$ is given by

$$f'(c;\delta) = \sup\{f'_t(c;\delta) : t \in M(c)\} \qquad \forall c \in \mathcal{C}.$$

Inside the above supremum we allow $+\infty$ and assume that the resulting value in such cases equals $+\infty$.

The next theorem gives a formula for the subdifferential of a max-function. See also [Hiriart-Urruty and Lemaréchal, 1996, Sec. VI.4.4] for other versions.

Theorem 36 *Under the assumptions of Theorem 35, the subdifferential of f at any point $c \in \mathcal{C}$ is given by*

$$\partial f(c) = \mathrm{cl}\left(\mathrm{conv}\left(\bigcup_{t\in M(c)} \partial f_t(c)\right)\right).$$

PROOF. Let $\mathcal{M} := \mathrm{cl}\left(\mathrm{conv}\left(\bigcup_{t\in M(c)} \partial f_t(c)\right)\right)$. First assume that $z \in \partial f(c)$. If $z \notin \mathcal{M}$ then there exists a hyperplane $\{v : v \in \mathbb{R}^m, \langle w, v\rangle + \alpha = 0\}$, $\alpha \in \mathbb{R}$, $w \in \mathbb{R}^m$, which strictly separates z from $\mathcal{M}$, that is, $\langle w, z\rangle + \alpha > 0$ and $\langle w, g\rangle + \alpha \le 0$, $\forall g \in \mathcal{M}$ (see Theorem 23). Subtracting, we conclude that

$$\sup\{\langle w, g\rangle : g \in \mathcal{M}\} < \langle w, z\rangle\,. \tag{2.1.2}$$

On the other hand, using Theorem 35 we obtain that $f'(c;\delta) = \sup\{{f_t}'(c;\delta) : t \in M(c)\}$ for every $\delta \in \mathbb{R}^m$. Thus, $\mathrm{cl}\, f'(c;\cdot) = \mathrm{cl}(\sup\{{f_t}'(c;\cdot) : t \in M(c)\}) = \sup\{\mathrm{cl}({f_t}'(c;\cdot)) : t \in M(c)\}$, by Theorem 20. Applying the "max formula" (Theorem 33), we get

$$\sup\{\langle w, g\rangle : g \in \partial f(c)\} = \sup\{\sup\{\langle w, g\rangle : g \in \partial f_t(c)\} : t \in M(c)\}\,,$$

which contradicts (2.1.2). Thus $z \in \mathcal{M}$.

Conversely, assuming that $z \in \mathcal{M}$, it has to be the limit of a sequence of vectors $\{z_n \in \mathrm{conv}\big(\bigcup_{t\in M(c)} \partial f_t(c)\big) : n \in \mathbb{N}\}$. From the definition of subgradients, for each z_n and every $c' \in \mathcal{C}$,

$$\begin{aligned}\langle z_n, c' - c\rangle &\le \sup\{\langle g_t, c' - c\rangle : g_t \in \partial f_t(c), t \in M(c)\}\\ &\le \sup\{f_t(c') - f(c) : t \in M(c)\} \le f(c') - f(c)\,.\end{aligned}$$

Taking the limits, we obtain that $z \in \partial f(c)$ and the result follows. ■

2.1.6 The Convex Conjugate

An important part of optimization theory is concerned with primal-dual pairs of problems. Underlying such duality correspondences there are several types of dualities, one of which arises from *Fenchel conjugation.*

Definition 37 *Consider an arbitrary function $f : \mathbb{R}^m \to \mathbb{R} \cup \{-\infty, +\infty\}$. The* conjugate *of f is the function $f^* : \mathbb{R}^m \to \mathbb{R} \cup \{-\infty, +\infty\}$ defined as*

$$f^*(z) = \sup\{\langle z, c\rangle - f(c) : c \in \mathbb{R}^m\} \qquad \forall z \in \mathbb{R}^m.$$

The conjugate function f^* is always convex. If $\operatorname{dom} f$ is nonempty then f^* never takes the value $-\infty$.

Theorem 38 *If $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is convex and its domain is nonempty, then f^* is lower semicontinuous and $f^{**} = \operatorname{cl} f$. Thus, conjugacy induces a one-to-one correspondence in the class of lower semicontinuous convex functions with nonempty domain.*

Another interesting duality correspondence connects conjugacy and subgradients.

Theorem 39 *Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ a lower semicontinuous convex function with nonempty domain. Then for all $c, z \in \mathbb{R}^m$,*

$$f(c) + f^*(z) \geq \langle c, z \rangle \, .$$

Equality holds if and only if $z \in \partial f(c)$, which is also equivalent to $c \in \partial f^*(z)$.

Two examples that relate to popular learning methods are the following. The conjugate of the square loss $f(c) = \|c - y\|^2$ equals $f^*(z) = \frac{1}{4}\|z\|^2 + \langle z, y \rangle$. The conjugate of the hinge loss $f(c) = \max(1 - yc, 0)$ equals $f^*(z) = \begin{cases} \dfrac{z}{y} & \text{if } \min(0, -y) \leq z \leq \max(0, -y) \\ +\infty & \text{otherwise} \end{cases}$.

One important use of the convex conjugate is in obtaining a dual problem equivalent to a given optimization problem. We shall apply such a technique in Lemma 55 of Section 3.2.3, which shows the convexity of the optimization problem for learning kernels. Another well known approach to obtaining dual problems is *Fenchel's duality theorem* (see for example [Borwein and Lewis, 2005]).

2.1.7 Optimization

One recurring theme of this thesis is to treat several learning problems under the light of optimization theory. In particular, we will be mainly interested in *convex optimization* problems.

A minimization problem has the form

$$\inf\{f(c) : c \in \mathbb{R}^m\} \, , \tag{2.1.3}$$

for some function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$. The first question that should be answered about a minimization problem is whether the minimum is actually attained. That is, whether there exists a (global) minimizer $\hat{c} \in \mathbb{R}^m$ of f such that $f(\hat{c}) = \inf\{f(c) : c \in \mathbb{R}^m\}$. A local minimizer is a minimizer of the problem further restricted in a neighborhood around $\hat{c}$. The set of minimizers of f is denoted by $\operatorname{argmin} f$.

Theorem 40 *Let $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be a lower semicontinuous function with nonempty domain. If all level sets of the form $\{c \in \mathbb{R}^m : f(c) \leq a\}, a \in \mathbb{R}$, are bounded then f has a global minimizer. This condition is equivalent to $\liminf\limits_{\|c\| \to \infty} f(c) = +\infty$.*

A convex problem is a problem of the form

$$\inf\{\, f(c) \; : \; c \in \mathbb{R}^m, \; f_i(c) \leq 0 \;\; \forall i \in \mathbb{N}_k, \; \langle a_j, c \rangle = 0 \;\; \forall j \in \mathbb{N}_\ell \} \, ,$$

where $f, f_1, \ldots, f_k : \mathbb{R}^m \to \mathbb{R}$ are convex functions. The inequalities $f_i(c) \leq 0, i \in \mathbb{N}_k$, express a finite set of *convex constraints* and the equalities $\langle a_j, c \rangle = 0, j \in \mathbb{N}_\ell$, a finite set of *linear constraints*. Clearly, we can rewrite any such problem in the form (2.1.3) using some convex function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ whose domain is specified by the constraints, so that the convex analytic tools of the previous sections can be applied.

Theorem 41 *Assume that $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is convex. Then*

(a) Any local minimizer of f is a global minimizer.

(b) The set of all minimizers, $\operatorname{argmin} f$*, is convex.*

(c) If in addition f is strictly convex *then the minimizer is* unique.

2.1.8 Minimax Problems

Another category of optimization problems is minimax problems. They are of the form

$$\inf\{\sup\{f(c,z) : z \in \mathcal{Z}\} : c \in \mathcal{C}\}$$

or of the form

$$\sup\{\inf\{f(c,z) : c \in \mathcal{C}\} : z \in \mathcal{Z}\}$$

where $f : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ and $\mathcal{C}, \mathcal{Z}$ are nonempty sets. It is easy to show that the values of the two problems are always ordered:

$$\sup\{\inf\{f(c,z) : c \in \mathcal{C}\} : z \in \mathcal{Z}\} \le \inf\{\sup\{f(c,z) : z \in \mathcal{Z}\} : c \in \mathcal{C}\}.$$

Under certain conditions, this inequality becomes an equality, that is, the infimum and the supremum can be interchanged. One case when this happens is when there exists a *saddle point*.

Definition 42 *A point $(\hat{c}, \hat{z}) \in \mathcal{C} \times \mathcal{Z}$ is called a* saddle point *of f if it satisfies*

$$f(\hat{c}, z) \le f(\hat{c}, \hat{z}) \le f(c, \hat{z}) \qquad \forall c \in \mathcal{C}, z \in \mathcal{Z}.$$

There are several results relating to minimax problems in the literature. Here, we record a version of the classical von Neumann minimax theorem that can be found in [Aubin, 1982, Ch. 7].

Theorem 43 *Let $f : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}$ where $\mathcal{C}$ is a closed convex subset of a Hausdorff topological vector space $\mathscr{C}$ and $\mathcal{Z}$ is a convex subset of a vector space $\mathscr{Z}$. If the function $c \mapsto f(c,z)$ is convex and lower semicontinuous for every $z \in \mathcal{Z}$, the function $z \mapsto f(c,z)$ is concave for every $c \in \mathcal{C}$ and there exists $z_0 \in \mathcal{Z}$ such that for all $a \in \mathbb{R}$ the set $\{c : c \in \mathcal{C},\ f(c, z_0) \le a\}$ is compact, then there exists $\hat{c} \in \mathcal{C}$ such that*

$$\sup\{f(\hat{c}, z) : z \in \mathcal{Z}\} = \sup\{\inf\{f(c,z) : c \in \mathcal{C}\} : z \in \mathcal{Z}\}.$$

In particular, we have that

$$\min\{\sup\{f(c,z) : z \in \mathcal{Z}\} : c \in \mathcal{C}\} = \sup\{\inf\{f(c,z) : c \in \mathcal{C}\} : z \in \mathcal{Z}\}.$$

Note that the above theorem does not show the existence of a saddle point, since the supremum over z may not be attained in general.

2.2 Regularization with Reproducing Kernels

We now review a family of learning methods that has been used extensively in recent years. These methods are based on regularization problems which are posed over Hilbert spaces of a certain type. The discussion below is based on [Aronszajn, 1950, Cucker and Smale, 2001, Evgeniou et al., 2000, Gohberg et al., 2003, Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004, Wahba, 1990].

2.2.1 Reproducing Kernel Hilbert Spaces

An *inner product* on a vector space E is a real valued function $\langle \cdot, \cdot \rangle$ defined on $E \times E$ with the properties:

(a) $\langle u + u', v\rangle = \langle u, v\rangle + \langle u', v\rangle$

(b) $\langle au, v\rangle = a\langle u, v\rangle$

(c) $\langle u, v\rangle = \langle v, u\rangle$

(d) $\langle u, u\rangle \geq 0$

(e) $\langle u, u\rangle = 0$ if and only if $u = 0$

for all $u, u', v \in E, a \in \mathbb{R}$. E together with an inner product $\langle \cdot, \cdot \rangle$ is called an *inner product space*. The associated norm is defined as $\|u\| = \sqrt{\langle u, u\rangle}$ for every $u \in E$.

A sequence $\{u_n : n \in \mathbb{N}\}$ in an inner product space $\mathcal{H}$ is said to *converge* to a point $u \in \mathcal{H}$, denoted by $u_n \to u$, if $\|u_n - u\| \to 0$. A *Cauchy sequence* is a sequence $\{u_n : n \in \mathbb{N}\}$ such that $\|u_n - u_m\| \to 0$ as $n, m \to \infty$. The space $\mathcal{H}$ is called *complete* if any Cauchy sequence converges to an element of $\mathcal{H}$.

Definition 44 *A* Hilbert space *is an inner product space that is also complete.*

The concept of Hilbert space is a natural generalization of $\mathbb{R}^m$ with the standard inner product.

In machine learning, we are interested in a particular type of Hilbert space. It relates closely to the concept of *reproducing kernel*. Consider a set $\mathcal{X}$.

Definition 45 *A* reproducing kernel *is a symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that, for every finite set of inputs $\{x_j : j \in \mathbb{N}_m\} \subseteq \mathcal{X}$ and every $m \in \mathbb{N}$, the $m \times m$ matrix $(K(x_i, x_j) : i, j \in \mathbb{N}_m)$ is positive semidefinite.*

It turns out that every kernel is associated with an (essentially) *unique* Hilbert space $\mathcal{H}_K$ of functions $f : \mathcal{X} \to \mathbb{R}$. Consider a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle$. Then K is a reproducing kernel associated with $\mathcal{H}$ if and only if

(a) for every $x \in \mathcal{X}$, $K_x \in \mathcal{H}$

(b) for every $f \in \mathcal{H}$ and $x \in \mathcal{X}$, $\langle f, K_x\rangle = f(x)$,

where $K_x(\cdot) := K(x, \cdot)$. Property (b) is the “reproducing property”. It implies that the inner product in $\mathcal{H}$ can be defined from the bilinear form $\sum_{i,j \in \mathbb{N}_m} \alpha_i \beta_j K(x_i, x_j)$, which can be shown to induce a norm $\langle f, f\rangle$.

Another important property is that a Hilbert space $\mathcal{H}$ admits a reproducing kernel if and only if, for every $x \in \mathcal{X}$, the point evaluation functional $L_x(f) := f(x),\ \forall f \in \mathcal{H}$, is continuous on $\mathcal{H}$. Finally, if $\mathcal{H}$ admits a reproducing kernel, this kernel is *unique*.

A Hilbert space that admits a reproducing kernel is called a *reproducing kernel Hilbert space* (RKHS). From now on, we will be using the term kernel instead of reproducing kernel, for brevity.

Some operations that give rise to kernels are the following.

Theorem 46

(a) If K_1 and K_2 are kernels then $K_1 + K_2$ is a kernel.

(b) If K is a kernel then aK is also a kernel, for every $a \geq 0$.

(c) If K_1 and K_2 are kernels then $K_1 K_2$ is a kernel.

(d) Let a function $g : \mathcal{X} \to \mathbb{R}^k$. Then the function defined by $K(x, x') = \langle g(x), g(x') \rangle, \forall x, x' \in \mathcal{X}$, is a kernel.

Common examples of kernels on $\mathcal{X} = \mathbb{R}^m$ include

(a) *linear kernels*, $K(x, x') = \langle x, Ax' \rangle$, where A is an $m \times m$ positive semidefinite matrix

(b) *polynomial kernels*, $K(x, x') = (\langle x, x' \rangle + a)^k$, where $a \geq 0, k \in \mathbb{N}$

(c) *Gaussian kernels*, $K(x, x') = e^{-\omega \|x - x'\|^2}$, where $\omega > 0$.

2.2.2 Regularization in an RKHS

Let $\mathcal{X}$ be a set from which inputs are drawn and $\mathbf{x} := \{(x_j, y_j) : j \in \mathbb{N}_m\} \subseteq \mathcal{X} \times \mathbb{R}$ be prescribed data. Supervised learning methods learn a function $f : \mathcal{X} \to \mathbb{R}$ that fits well the data $\mathbf{x}$ while having small generalization error on the whole space $\mathcal{X}$. The outputs $\{y_j : j \in \mathbb{N}_m\}$ may take values in $\mathbb{R}$, which is the case of *regression*, in $\{-1, +1\}$ (*classification*), or in $\mathbb{N}$ (*ranking*).

One well studied approach for supervised learning is to develop methods that, based on $\mathbf{x}$, select a suitable function f from an RKHS $\mathcal{H}$. The main motivation is that learning in a linear class requires simpler algorithms. Moreover, using kernels allows us to take advantage of some appealing computational properties, as we shall see below. Thus, one can learn in an RKHS that includes functions of arbitrary complexity or predictive ability, as long as the associated kernel is known.

Another motivation is that such methods can be interpreted as learning by using *feature maps*. A feature map is a map $\Phi : \mathcal{X} \to \mathcal{W}$, where $\mathcal{W}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{W}}$. It can be viewed as a transformation or as a representation of the inputs. For example, when the feature space $\mathcal{W}$ is finite dimensional, the inputs $x \in \mathcal{X}$ are mapped to a finite number of features $\Phi_1(x), \ldots, \Phi_k(x) \in \mathbb{R}$. The kernel K is obtained from the feature map simply from inner products, $K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{W}}, \forall x, x' \in \mathcal{X}$. However, feature maps are not uniquely determined from kernels.

Let us use the notation $K_{\mathbf{x}} := (K(x_i, x_j) : i, j \in \mathbb{N}_m)$ for the *kernel matrix* or *Gram matrix* of kernel values on the data. Assume also a prescribed convex nonnegative error function $Q : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+$. The most usual choice is $Q(y, w) = \sum_{j \in \mathbb{N}_m} L(y_j, w_j)$, for $w = (w_j : j \in N_m)$, where L is a nonnegative loss function, convex in the second argument. Assume that a kernel K is given and let $\mathcal{H}_K$ be its associated RKHS. Let also $\langle \cdot, \cdot \rangle_K$, $\| \cdot \|_K$ be the inner product and norm respectively in $\mathcal{H}_K$.

For every $f \in \mathcal{H}_K$, the standard L_2 *regularization functional* is defined as

$$\mathcal{E}_\gamma(f, K) := Q(y, I_{\mathbf{x}}(f)) + \gamma \|f\|_K^2 \tag{2.2.1}$$

where the operator $I_{\mathbf{x}}$ is defined as $I_{\mathbf{x}}(f) := (f(x_j) : j \in \mathbb{N}_m)$. The constant $\gamma > 0$ is called the regularization parameter. L_2 regularization methods solve the variational problem

$$E_\gamma(K) := \inf\{\mathcal{E}_\gamma(f, K) : f \in \mathcal{H}_K\}\,. \tag{2.2.2}$$

Minimization of the functional $\mathcal{E}_\gamma$ balances the effect of two terms. The error term, $Q(y, I_{\mathbf{x}}(f))$, favors functions that fit well the available training data $\mathbf{x}$. In contrast, the smoothness term or *regularizer*, $\|f\|_K^2$, is the square of an L_2 type norm in $\mathcal{H}_K$ and hence favors functions that have a more uniform representation. Thus, the regularization functional avoids *overfitting* with a highly complex function that fits "too well" the data, by favoring simple solutions through the regularizer. To see what happens at the extremes, when $\gamma \to 0$, the minimizing $\hat{f}$ approaches the function with the smallest norm that fits exactly the data (*minimal norm interpolation*). When $\gamma \to +\infty$, the solution approaches the identically zero function. Moreover, results bounding the generalization error of the solutions of such problems can be derived [Shawe-Taylor and Cristianini, 2004].

We have implicitly assumed that problem (2.2.2) admits a minimizer and this is indeed true. Since $Q : \mathbb{R}^m \to \mathbb{R}_+$ is continuous and γ is a positive number, the infimum in (2.2.2) is attained because the unit ball in $\mathcal{H}_K$ is *weakly* compact. In addition, since Q is convex the minimizer is unique because the right hand side of equation (2.2.1) is a *strictly convex* functional of f.

It is feasible to solve problem (2.2.2) regardless of the dimensionality of the Hilbert space, because of the so called Representer Theorem [Wahba, 1990].

Theorem 47 *If $\hat{f}$ is a solution to problem (2.2.2) then it has the form*

$$\hat{f}(x) = \sum_{j \in \mathbb{N}_m} \alpha_j K(x_j, x) \qquad \forall x \in \mathcal{X} \tag{2.2.3}$$

for some real vector $\alpha = (\alpha_j : j \in \mathbb{N}_m)$.

Although it is simple to prove, this result is remarkable as it makes the variational problem (2.2.2) amenable to computations. By replacing f with the right hand side of equation (2.2.3) in equation (2.2.1) and then optimizing with respect to the vector α, we obtain the equivalent finite dimensional problem

$$E_\gamma(K) := \min\{Q(K_{\mathbf{x}}\alpha) + \gamma\langle \alpha, K_{\mathbf{x}}\alpha \rangle : \alpha \in \mathbb{R}^m\} \tag{2.2.4}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product on $\mathbb{R}^m$.

The regularization framework presented above subsumes many well known learning methods. For example, when Q is the square loss, $Q(y,w) = \|w-y\|^2$, the objective function in (2.2.4) is quadratic in the vector α and its minimizer is obtained by solving a linear system of equations. When Q is the hinge loss, $Q(y,w) = \sum_{j\in\mathbb{N}_m} \max(1-y_j w_j, 0)$, we obtain the standard support vector machine (SVM) problem – see, for example, [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004]. A variety of algorithmic methods can be applied to the solution of such regularization problems. In the SVM case, for instance, either off-the-shelf quadratic programming methods or recently developed decomposition methods [Joachims, 1998, Platt, 1998] can be used.

Finally, regarding the issue of determining the regularization parameter γ, the most common approach has been *cross validation* [Wahba, 1990]. In this, the training data $\mathbf{x}$ is split into a small number k of subsets. Then, the optimization problem (2.2.4) is solved k times, each time replacing $\mathbf{x}$ with the data in the $k-1$ of the data sets and measuring the test error of the solution on the k-th data set. In this way, an average measure of the error can be obtained for any fixed value of γ. This technique has good statistical properties but has significant computational cost because the learning algorithm has to be repeated for a number of values covering the range of γ. There are also a few other, more recent, methods for determining the regularization parameter, such as computation of the regularization path [Hastie et al., 2004].

2.2.3 Semi-Supervised Learning with Graph Laplacians

Apart from supervised learning, regularization in an RKHS has been applied to other problems as well. Among these are *semi-supervised learning* problems, which are classification problems with only a few of the output labels y_j available. They have received significant attention in recent years, see, for example, [Belkin and Niyogi, 2004, Blum and Chawla, 2001, Joachims, 2003, Kondor and Lafferty, 2002, Zhou et al., 2004, Zhu et al., 2003, 2005] and references therein. The main insight of semi-supervised methods is that unlabeled data may be used to improve on the performance of learners that are only based on the labeled data. Several semi-supervised learning methods build on the assumption that the data is situated on a low dimensional manifold within the ambient space of the data and that this manifold can be approximated by a weighted discrete graph whose vertices are identified with the empirical (labeled and unlabeled) data.

Let $\mathscr{G}$ be an undirected graph with m vertices and an $m \times m$ adjacency matrix $\mathbf{A}$ such that $A_{ij} = 1$ if there is an edge connecting vertices i and j and 0 otherwise[1]. The graph Laplacian $\mathbf{L}$ is the $m \times m$ matrix defined as $\mathbf{L} := \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \operatorname{diag}(d_i : i \in \mathbb{N}_m)$ and d_i is the degree of vertex i, that is $d_i = \sum_{j\in\mathbb{N}_m} A_{ij}$.

We can identify the linear space of real-valued functions defined on the graph with $\mathbb{R}^m$ and introduce on it the semi-inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle := \mathbf{u}^\top \mathbf{L} \mathbf{v} \qquad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^m.$$

The induced seminorm is $\|\mathbf{v}\| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$, $\forall \mathbf{v} \in \mathbb{R}^m$. It is a seminorm since $\|\mathbf{v}\| = 0$ if $\mathbf{v}$ is a constant

[1] The ideas we discuss below naturally extend to weighted graphs.

vector, as can be verified by noting that $\|\mathbf{v}\|^2 = \frac{1}{2}\sum_{i,j\in\mathbb{N}_m}(v_i - v_j)^2 A_{ij}$.

It is known that $\mathscr{G}$ has r connected components if and only if $\mathbf{L}$ has r eigenvectors with zero eigenvalues. These eigenvectors are piecewise constant on the connected components of the graph. In particular, $\mathscr{G}$ is connected if and only if the constant vector is the only eigenvector of $\mathbf{L}$ with zero eigenvalue [Chung, 1997]. Therefore, we consider the linear subspace $\mathcal{H}(\mathscr{G})$ of $\mathbb{R}^m$

$$\mathcal{H}(\mathscr{G}) := \mathrm{range}(\mathbf{L}).$$

We wish to learn a function $\mathbf{v} \in \mathcal{H}(\mathscr{G})$ based on a set of labeled vertices. Without loss of generality we assume that the first $\ell \le m$ vertices are labeled and let $y_1, ..., y_\ell \in \{-1, 1\}$ be the corresponding labels. Following [Belkin and Niyogi, 2004] we prescribe a loss function Q and compute the function $\mathbf{v}$ by solving the optimization problem

$$\min\left\{Q(y, \bar{\mathbf{v}}) + \gamma\|\mathbf{v}\|^2 : \mathbf{v} \in \mathcal{H}(\mathscr{G})\right\}\ , \tag{2.2.5}$$

where $\bar{\mathbf{v}}$ is the labeled part of $\mathbf{v}$. A similar approach is presented in [Zhu et al., 2003] where $\mathbf{v}$ is (essentially) obtained as the minimal norm interpolant in $\mathcal{H}(\mathscr{G})$ to the labeled vertices. The functional (2.2.5) balances the error on the labeled points with a smoothness term measuring the complexity of $\mathbf{v}$ on the graph. Note that this last term contains the information of both the labeled and unlabeled vertices via the graph Laplacian.

Method (2.2.5) is a special case of problem (2.2.2). Indeed, the restriction of the semi-norm $\|\cdot\|$ on $\mathcal{H}(G)$ is a norm. Moreover, the pseudoinverse of the Laplacian, $\mathbf{L}^+$, is the reproducing kernel of $\mathcal{H}(\mathscr{G})$ – see, for example, [Herbster et al., 2005] for a proof. This means that for every $\mathbf{v} \in \mathcal{H}(\mathscr{G})$ and $i \in \mathbb{N}_m$ there holds the reproducing kernel property $v_i = \langle \mathbf{L}_i^+, \mathbf{v}\rangle$, where $\mathbf{L}_i^+$ is the i-th column of $\mathbf{L}^+$. Hence, by setting $\mathcal{X} = \mathbb{N}_m$, $f(i) = v_i$ and $K(i,j) = L_{ij}^+, \forall i,j \in \mathbb{N}_m$, we see that $\mathcal{H}_K = \mathcal{H}(\mathscr{G})$.

2.3 Spectral Functions

In Chapter 4, which is concerned with multi-task learning, we will be interested in regularization problems that involve matrices instead of vectors. Let us use $\mathbf{S}^d_{++}$ to denote the set of $d \times d$ positive semidefinite matrices, $\mathbf{O}^d$ the set of $d \times d$ orthogonal matrices and $\mathrm{Diag}(\lambda_i)_{i\in\mathbb{N}_d}$ the diagonal matrix with diagonal entries $\{\lambda_i : i \in \mathbb{N}_d\}$. We will use regularizers of the form

$$\mathrm{trace}(W^\top F(D) W) = \sum_{t\in\mathbb{N}_T} w_t^\top F(D) w_t\,, \tag{2.3.1}$$

where $W \in \mathbb{R}^{d\times T}, D \in \mathbf{S}^d_{++}$ and $F : \mathbf{S}^d_{++} \to \mathbf{S}^d_{++}$ is a *spectral* matrix function. Therefore, a question of interest is when such regularizers are convex (jointly in W, D).

Definition 48 *We call a matrix function $F : \mathbf{S}^d_{++} \to \mathbf{S}^d_{++}$* spectral *if it is induced by applying a real function $f : (0,\infty) \to (0,\infty)$ to the eigenvalues of its argument. That is, for every $D \in \mathbf{S}^d_{++}$ we write $D = U\mathit{Diag}(\lambda_i)_{i\in\mathbb{N}_d}U^\top$, where $U \in \mathbf{O}^d$, and define*

$$F(D) = U\mathit{Diag}(f(\lambda_i))_{i\in\mathbb{N}_d}U^\top\,. \tag{2.3.2}$$

We will adhere to the convention that capital letters denote a spectral matrix function and small letters the associated real function as above.

Addressing the issue of convexity of the regularizer (2.3.1) requires the matrix analytic concept of concavity (see, for example, [Bhatia, 1997]).

Definition 49 *We say that the function $g : (0,\infty) \to \mathbb{R}$ is* matrix concave of order d *if*

$$\lambda G(A) + (1-\lambda)G(B) \preceq G(\lambda A + (1-\lambda)B) \qquad \forall A, B \in \mathbf{S}^d_{++}, \lambda \in [0,1]\,,$$

where G is defined as in (2.3.2).

The notation $\preceq$ denotes the *Loewner partial order* on $\mathbf{S}^d$: $A \preceq B$ if and only if $B - A$ is positive semidefinite. If g is a matrix concave function of order d for any $d \in \mathbb{N}$, we simply say that g is *matrix concave*. We also say that g is *matrix convex* (of order d) if $-g$ is matrix concave (of order d).

Definition 50 *We say that the function $g : (0,\infty) \to \mathbb{R}$ is* matrix monotone of order d *if*

$$A \preceq B \implies G(A) \preceq G(B) \qquad \forall A, B \in \mathbf{S}^d_{++}\,,$$

where G is defined as in (2.3.2).

Theorem 51 *A function $g : (0,\infty) \to \mathbb{R}$ is matrix monotone (of any order) if and only if it is matrix concave.*

Examples of matrix concave/monotone functions on $(0,\infty)$ are $\log x$ and the function x^s for $s \in [0,1]$ – see [Bhatia, 1997, Horn and Johnson, 1991] for other examples and theoretical results.

Now we can characterize the class of spectral functions F for which the term $w^\top F(D) w$ is jointly convex in (w, D), with $w \in \mathbb{R}^d, D \in \mathbf{S}^d_{++}$.

Theorem 52 *Let $F : \mathbf{S}^d_{++} \to \mathbf{S}^d_{++}$ be a spectral function. Then the function $\rho : \mathbb{R}^d \times \mathbf{S}^d_{++} \to [0,\infty)$ defined as $\rho(w,D) = w^\top F(D) w$ is jointly convex if and only if $\frac{1}{f}$ is matrix concave of order d.*

PROOF. By definition, ρ is convex if and only if, for any $w_1, w_2 \in \mathbb{R}^d, D_1, D_2 \in \mathbf{S}^d_{++}$ and $\lambda \in (0,1)$, it holds that

$$\rho(\lambda w_1 + (1-\lambda)w_2, \lambda D_1 + (1-\lambda)D_2) \le \lambda \rho(w_1, D_1) + (1-\lambda)\rho(w_2, D_2).$$

Let $C := F(\lambda D_1 + (1-\lambda)D_2), A := F(D_1)/\lambda, B := F(D_2)/(1-\lambda)$, $w := \lambda w_1 + (1-\lambda)w_2$ and $z := \lambda w_1$. Using this notation, the above inequality can be rewritten as

$$w^\top C w \le z^\top A z + (w-z)^\top B (w-z) \qquad \forall\, w, z \in \mathbb{R}^d. \tag{2.3.3}$$

The right hand side in (2.3.3) is minimized for $z = (A+B)^{-1} B w$ and hence (2.3.3) is equivalent to

$$w^\top C w \le w^\top \left[B(A+B)^{-1} A (A+B)^{-1} B + \left(I - (A+B)^{-1}B\right)^\top B \left(I - (A+B)^{-1}B\right)\right] w\,,$$

$\forall\, w \in \mathbb{R}^d$, or to

$$\begin{aligned}
C &\preceq B(A+B)^{-1}A(A+B)^{-1}B + \big(I-(A+B)^{-1}B\big)^{\top} B \big(I-(A+B)^{-1}B\big) \\
&= B(A+B)^{-1}A(A+B)^{-1}B + B - 2B(A+B)^{-1}B + B(A+B)^{-1}B(A+B)^{-1}B \\
&= B - B(A+B)^{-1}B = (A^{-1}+B^{-1})^{-1}\,,
\end{aligned}$$

where the last equality follows from the matrix inversion lemma [Horn and Johnson, 1985, Sec. 0.7]. The above inequality is identical to (see e.g. [Horn and Johnson, 1985, Sec. 7.7])

$$A^{-1}+B^{-1} \preceq C^{-1}\,,$$

or, using the initial notation,

$$\lambda\big(F(D_1)\big)^{-1} + (1-\lambda)\big(F(D_2)\big)^{-1} \preceq \big(F(\lambda D_1 + (1-\lambda)D_2)\big)^{-1}\,.$$

By definition, this inequality holds for any $D_1, D_2 \in \mathbf{S}^d_{++}, \lambda \in (0,1)$ if and only if $\frac{1}{f}$ is matrix concave of order d. ■

Finally, we state a basic inequality due to von Neumann which underlies many facts about spectral functions. It can be found, for example, in [Horn and Johnson, 1991, ex. 3.3.10]. We use $\sigma(W), \bar{\sigma}(W)$ to denote the vector of singular values of matrix $W \in \mathbb{R}^{d\times T}$ in *nonincreasing* and *nondecreasing* order respectively.

Lemma 53 *For any $W, C \in \mathbb{R}^{d\times T}$, we have that*

$$\langle \bar{\sigma}(W), \sigma(C)\rangle \leq \langle W, C\rangle \leq \langle \sigma(W), \sigma(C)\rangle\,.$$

The upper equality holds if and only if there are $U \in \mathbf{O}_d$ and $V \in \mathbf{O}_T$ such that $W = U\mathit{Diag}(\sigma(W))V^\top$ and $C = U\mathit{Diag}(\sigma(C))V^\top$. Similarly for the lower equality.

We emphasize that equality holds when not only are the row and column spaces for W and C equal, but also the basis vectors match *if ordered according to the ordering of the singular values*. It is also worth noting that this inequality is stronger than the Cauchy-Schwarz inequality for the Frobenius norm, $\langle W, C\rangle \leq \|W\|_2\|C\|_2$.

Chapter 3

Learning from Multiple Sources and Parameterizations

On a daily basis, humans make decisions by using heterogeneous sources of information about the same task. In fact, there are many situations in which combining different sources helps us learn significantly better than by using a single source. The importance of this trait has been appreciated in the machine learning community as well. A substantial amount of recent work attempts to exploit heterogeneous features which might encode different types of information about the same task or different representations of the data. The resulting algorithms exhibit improved statistical performance in many applications, such as biological ones, in which it is important to integrate different sources of knowledge.

We begin this chapter by reviewing relevant approaches from the supervised learning literature. Among the various approaches we discuss, we emphasize those which are based on regularization in reproducing kernel Hilbert spaces. The main idea behind them is that heterogeneous sources can give rise to different feature maps and hence to different kernels, which can then be appropriately combined. One case widely considered in the literature is that of a *finite* number of prescribed kernels. In this thesis, however, we do not limit ourselves to a finite number of kernels, nor is anything known about them other than their parameterized functional form.

We proceed to study the problem of combining kernels, by following the steps below.

- We define a general framework which formalizes the approach of learning combinations of parameterized kernels
- show that regularization in this framework leads to a saddle point optimization problem
- show that the "simplest" solution to this problem can be represented as a combination of a *bounded number of kernels* and exhibits some interesting properties
- specialize these results to the better studied case of combining a finite number of kernels and show that it leads to a convex optimization problem.

Our discussion then naturally leads to

- a general-purpose algorithm for combining heterogeneous parameterized kernels.

- an investigation of its convergence properties and
- application of techniques from optimization theory for accelerating it in computationally demanding situations.

We also discuss the computational complexity of the saddle point optimization problem and show that, depending on the parameterization of the kernel, it can give rise to both *convex and nonconvex problems*.

Finally, we present a variation of our algorithm which can be used in different classification settings, when only few labels are available (semi–supervised learning).

3.1 Learning by Integrating Sources

There are many practical situations in which it is advantageous to combine heterogeneous sources or data representations. One example is protein function prediction (see [Lanckriet et al., 2004]) where different types of information about the same protein may come from amino-acid sequences, protein-protein interactions, gene expression data etc. Another biological example is combining co-occurrences of oligomers between two DNA sequences (see [Sonnenburg et al., 2006]), which gives rise to a *weighted degree* kernel as it is called. In vision problems, one may want to combine different image features or different parts of the image, as in [Argyriou et al., 2006a]. Also, different representations can be obtained from a graph or different graphs, especially in a semi-supervised setting (see [Argyriou et al., 2006b, Dai and Yeung, 2007, Sindhwani et al., 2005, Tsuda et al., 2005, Zhu et al., 2005]).

Several recently proposed approaches [Bach et al., 2004, Bennett et al., 2002, Bi et al., 2004, Crammer et al., 2003, Girolami and Rogers, 2005, Girolami and Zhong, 2007, Kondor and Jebara, 2007, Lanckriet et al., 2004, Lin and Zhang, 2003, Micchelli and Pontil, 2005, Ong et al., 2005, Parrado-Hernández et al., 2003, Pelckmans, 2005] tackle such situations by learning with *combinations of kernels* (or *multiple kernels*, to use another common term). First, a kernel is defined for each source or representation, through inner products of features or in some other way. Thus a set of *basic kernels* is obtained. Then a regularization-based optimization problem is solved over the set of linear or convex combinations of these kernels. The goal is to learn a combined kernel and a corresponding regression/classification function which predicts better than any function trained with one of the basic kernels. Moreover, the weights in this combination should give a measure of how relevant each kernel is to the task at hand.

The main motivation behind this methodology has been computational. A principled approach for selecting optimal kernels from a set of candidates (or a set of combinations for that matter) is *cross validation* – see [Wahba, 1990] for a discussion. In this, the training data is split in k subsets and the learning method is run k times, each time using one of the subsets as test data and the rest as training data. In this way, an estimate of the expected error of the method for a specific kernel can be computed. One drawback is that cross validation can be very inefficient, since the space of candidate kernels has to be properly covered with a grid of kernels, which can incur exponential cost. A further drawback is that the size of the training data is effectively reduced and this leads to deterioration of performance. Thus, the aforementioned works on combinations of kernels attempt to avoid these drawbacks by employing appropriate optimization problems, which are more efficient to solve. On the other hand, there are better statistical guarantees for the results obtained with cross validation, although some guarantees also exist for the methods of kernel combinations (see Section 3.7).

In these methods, the set of basic kernels can be constructed using prior information, but they can also be general-purpose ones such as the linear, polynomial or Gaussian kernels. That is, even in the absence of any knowledge about possible "good" features or representations, one can attempt to combine different types of generic kernels. Feature maps with high approximating power, such as the ones corresponding to Gaussian kernels, can be useful in many tasks and sometimes even more so than a priori known feature maps.

In this thesis, we shall make the general assumption that the basic kernels belong to a *parameterized family*. This subsumes both of the situations described above and departs from most of the literature in that the number of basic kernels *need not be finite*. In order to learn combinations of such basic kernels, we adopt the framework of [Micchelli and Pontil, 2005], which we study in more detail. One notable example that falls under this framework is isotropic Gaussian basic kernels whose variance parameter can take any nonnegative real value. We will also consider anisotropic Gaussian kernels, although we shall see that the optimization problems to be solved become less tractable as the number of kernel parameters increases. A finite set of basic kernels is another example, as are unions of sets of basic kernels.

3.1.1 Finite Combinations of Kernels

In the context of supervised learning there has been a great deal of interest in learning combinations of a finite number of kernels. Some characteristic early work on this topic can be found in [Lanckriet et al., 2002, 2004], which address this problem as a semidefinite program. The main formulation of the papers is applicable to supervised learning with SVMs, even though the setting of these papers is *transduction*, which means predicting the missing labels $y_{s+1}, \ldots, y_m$ of a partially labeled sample $\mathbf{x} = \{(x_1, y_1), \ldots, (x_s, y_s), x_{s+1}, \ldots, x_m\}$ drawn from an input space $\mathcal{X}$. An extension to inductive supervised learning is also presented but, since it is based on transduction, it is less intuitive and efficient. The goal is to learn a combination of n kernels which gives good generalization performance on the task at hand by optimizing over a set of linear combinations of these kernels.

Assume that n kernels $B_1, \ldots, B_n$ are given and let $B_{\mathbf{x},i}, i \in \mathbb{N}_n$, denote the kernel matrix on the data $\mathbf{x}$ corresponding to kernel B_i. The optimization problem is considered over a convex set $\mathcal{K}$ of kernel matrices which can be either

$$\mathcal{K} = \left\{ K_{\mathbf{x}} = \sum_{i \in \mathbb{N}_n} \mu_i B_{\mathbf{x},i} : K_{\mathbf{x}} \in \mathbf{S}^m_{++}, \mathrm{trace} K_{\mathbf{x}} = c \right\} \tag{3.1.1}$$

or

$$\mathcal{K} = \left\{ K_{\mathbf{x}} = \sum_{i \in \mathbb{N}_n} \mu_i B_{\mathbf{x},i} : \mu_i \geq 0,\ i \in \mathbb{N}_n, \mathrm{trace} K_{\mathbf{x}} = c \right\} \tag{3.1.2}$$

The constraint normalizing the trace of the kernel matrix is necessary to avoid overfitting of the data (see Section 3.7). In [Lanckriet et al., 2004] and most related work, the regularization problems considered are hard-margin or soft-margin support vector machine (SVM) problems. In [Lanckriet et al., 2004], the formulation involves the dual optimization problem of an SVM, that is,

$$\max\{E(\alpha, K) \;:\; \alpha \in \mathbb{R}^m : 0 \leq \alpha \leq C, \langle \alpha, y \rangle = 0\},$$

where $C \geq 0$ is the margin parameter and $E(\alpha, K)$ is a function quadratic in α and linear in K (see Section 2.2.2). The above maximal value then serves as the criterion whose minimization yields the optimal K:

$$\min\{\max\{E(\alpha, K) \;:\; \alpha \in \mathbb{R}^m : 0 \leq \alpha \leq C, \langle \alpha, y \rangle = 0\} : K \in \mathcal{K}\}. \tag{3.1.3}$$

The authors show how this problem can be rewritten as a semidefinite program – case (3.1.1) – or a quadratically constrained quadratic program – case (3.1.2). These types of problems can be solved with general-purpose optimization packages which use interior-point methods.

However, such methods are practical only for moderately large values of m, n. As a result, more efficient variations have been proposed for large data sets. One such variation using sequential minimal optimization (SMO) techniques is the main topic in [Bach et al., 2004]. Applying SMO techniques to problem (3.1.3) is achieved by deriving Karush-Kuhn-Tucker conditions for optimality and regularizing the nonsmooth convex objective function with a technique called Moreau-Yosida regularization. This paper also offers an interesting insight about the relation of problem (3.1.3) to a feature selection problem: decomposing the data vectors into n blocks and doing L_1 regularization. We shall return to this connection in Section 3.6.

Apart from this method, other approaches that exploit decomposition of regularization problems into small subproblems have been proposed, for example the semi-infinite programming approach of [Sonnenburg et al., 2006] – see also Section 3.2.5.

The formulation of combining a finite number of kernels has also appeared, under a different guise, in the statistical community. An example is the related approach called COSSO [Lin and Zhang, 2003]. The authors consider an RKHS $\mathcal{F}$ that can be decomposed into p orthogonal subspaces as follows:

$$\mathcal{F} = \{1\} \oplus \bigoplus_{i=1}^{p} \mathcal{F}^i.$$

The COSSO estimate minimizes

$$\min\left\{ \frac{1}{m} \sum_{i \in \mathbb{N}_m} (y_i - f(x_i))^2 + \tau_m^2 \sum_{i \in \mathbb{N}_p} \|P^i f\| : f \in \mathcal{F} \right\} \tag{3.1.4}$$

where $P^i f$ is the orthogonal projection of f onto $\mathcal{F}^i$ and τ_m a regularization parameter. Problem (3.1.4) can be reduced to minimizing an equivalent form

$$\min\left\{ \frac{1}{m} \sum_{i \in \mathbb{N}_m} (y_i - f(x_i))^2 + \gamma_0 \sum_{i \in \mathbb{N}_p} \theta_i^{-1} \|P^i f\|^2 + \gamma \sum_{i \in \mathbb{N}_p} \theta_i \right. \\ \left. : \; f \in \mathcal{F}, \theta_i \geq 0 \;\; \forall i \in \mathbb{N}_p \right\} \tag{3.1.5}$$

where γ_0 is a fixed positive constant and γ a regularization parameter. This formulation is similar to the standard smoothing spline method, except for the term $\sum_{i \in \mathbb{N}_p} \theta_i$. This term encourages zero components in the solution. Problem (3.1.5) is also similar to the feature space version of learning convex combinations of kernels – see Section 3.6 – where instead there is a constraint $\sum_{i\in\mathbb{N}_p}\theta_i = 1$. Another similar method to COSSO also originates from the statistical literature and is known as the *group lasso* – see [Bakin, 1999, Yuan and Lin, 2006].

In machine learning, another way to formulate model selection in a convex set of kernels has been proposed in [Ong et al., 2003, 2005]. In this work, the authors define the concept of a *hyper reproducing kernel Hilbert space* as a kind of reproducing kernel Hilbert space of functions $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, where $\mathcal{X}$ is the input space. This space is generated by a hyperkernel $\bar{K} : \mathcal{X}^2 \times \mathcal{X}^2 \mapsto \mathbb{R}$ with the property that for every $\bar{x} \in \mathcal{X}^2$, $\bar{K}(\bar{x}, (\cdot, \cdot))$ is a kernel. The learning algorithm is based on an optimization problem involving the values of $\bar{K}$ on the input data. Since the number of these values is $O(m^4)$, the computational task can be much harder than standard regularization and low-rank approximations often have to be employed.

3.1.2 Model Selection and Kernel Parameter Learning

Apart from the problem of combining kernels, another important problem is how to select the parameter(s) of the kernel in an optimal way. In general, the question of model selection is a fundamental one in any learning method, since there is always the assumption of a model that depends on a number of hyperparameters. As we have already mentioned, cross validation is a good but inefficient way to do model selection. However, as in the kernel combinations' literature, the problem has been recently viewed in the light of optimization.

Specifically, [Chapelle et al., 2002] have proposed optimizing a variety of bounds on the true error, thus obtaining a number of possible algorithms based on gradient descent. Other points of view are to optimize some approximation of maximum likelihood [Gold and Sollich, 2003, Law and Kwok, 2001]; a method inspired by cross validation [Schittkowski, 2005]; maximizing a measure of alignment between kernel matrices [Cristianini et al., 2001]. And in Gaussian processes, a variational approximation has been proposed for optimizing over the hyperparameters of the kernel [Seeger, 2000].

Even though the objectives of learning combined kernels and learning kernel parameters are related, we would like to point out some differences. In some situations, the prescribed basic kernels capture different aspects of the problem at hand, so that *several kernels can be relevant* and they may *complement each other*. Then it is natural to learn combinations of these basic kernels.[1] In other situations, it may be expected that a single basic kernel achieves optimal results and then it is more natural to learn the optimal kernel parameters. Besides this dilemma, it may not even be obvious what the set of basic kernels should be. Clearly, the choice of basic kernels reflects, to a small or large extent, prior knowledge and beliefs about the task or practical considerations and influences the resolution of the above dilemma.

Finally, we briefly mention some of the works that attempt to learn a distance for the input space via learning a kernel matrix. The methodology is different from the works mentioned above but the objective is similar. One approach is to learn a Mahalanobis distance metric, which can be used in a classification method such as k-nearest neighbors [Bar-Hillel et al., 2003, Goldberger et al., 2005, Weinberger et al., 2006, Xing et al., 2003]. To this end, the authors optimize measures combining clustering with label

[1]We shall see several such examples in the experimental section.

information. In fact, in some cases, the labeled information may be given in the form of constraints between data points. Another approach is to learn a kernel matrix using some type of information-theoretic entity subject to linear constraints [Davis et al., 2007, Kulis et al., 2006, Tsuda and Noble, 2004].

3.2 Parameterized Convex Combinations of Kernels

We now present the framework we shall use for learning convex combinations of parameterized basic kernels. It has initially been proposed and studied for the case of the square loss function in [Micchelli and Pontil, 2005].

3.2.1 Notation

For the reader's convenience, we display here the notational conventions used in this chapter.

$\mathbb{R}_+$	set of nonnegative real numbers
$\mathbb{N}_m$	set $\{1,\ldots,m\}$
$\mathbf{S}_+^m$	set of $m \times m$ symmetric positive semidefinite matrices
$\mathbf{S}_{++}^m$	set of $m \times m$ symmetric positive definite matrices
$\mathcal{X}$	set from which inputs are drawn
$\mathbf{x}$	set of inputs $\{x_j : j \in \mathbb{N}_m\} \subseteq \mathcal{X}$
y	vector of outputs $(y_j : j \in \mathbb{N}_m)$
Q	error function $\mathbb{R}^m \to \mathbb{R}_+$
L	loss function $\mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$
K	kernel function
$\mathcal{H}_K$	Hilbert space associated with kernel K
f	function belonging to a Hilbert space
$\mathcal{A}_+(\mathcal{X})$	set of kernels on $\mathcal{X}$
$\mathcal{A}_{++}(\mathcal{X})$	set of positive definite kernels on $\mathcal{X}$
$\langle\cdot,\cdot\rangle_K$, $\|\cdot\|_K$	inner product, norm in $\mathcal{H}_K$
$\langle\cdot,\cdot\rangle$, $\|\cdot\|$	standard inner product, norm in $\mathbb{R}^m$
$\mathcal{E}_\gamma$	regularization functional
E_γ	minimum value of the regularization functional
γ	regularization parameter
$K_\mathbf{x}$	kernel matrix on input data $\mathbf{x}$
$\operatorname{dom} f$	domain of convex function f

$\mathrm{cl}\, S$	closure of set S
$\mathrm{conv}\, C$	convex hull of set C
$\mathcal{B}$	set of basic kernels
B_i	basic kernels in $\mathcal{B}$
$\mathcal{K}$	set of convex combinations of basic kernels, $\mathrm{conv}(\mathcal{B})$
Q^*	convex conjugate of function Q
c	vector of m variables in the dual of the regularization problem
R	minus objective of the dual problem
$f'(x;\delta)$	right directional derivative of function f at x in the direction δ
$\partial f(x)$	subdifferential of function f at x
$(\hat{c}, \hat{K})$	saddle point of the minimax problem
Ω	space of kernel parameters
G	continuous mapping of parameters to kernels $\Omega \to \mathcal{A}_+(\mathcal{X})$
$\mathcal{P}(\Omega)$	set of probability measures on Ω
$\mathcal{K}(G)$	convex set of kernels generated by mapping G
$C(\Omega)$	set of continuous real valued functions on Ω
Σ	covariance matrix parameter of Gaussian kernel
σ	variance parameter of isotropic Gaussian kernel
Φ	feature map $\mathbb{R}^m \to \mathbb{R}^s$
$V(P)$	vertex set of polytope P
$\mathscr{G}$	graph
d_i	degree of vertex i
L	graph Laplacian
$\mathcal{H}(\mathscr{G})$	RKHS on graph $\mathscr{G}$
L^+	pseudoinverse of matrix L

3.2.2 Framework for Learning Combinations of Kernels

Let $\mathcal{X}$ be a set, from which inputs are drawn. Recall from Section 2.2 the notation $K_{\mathbf{x}} := (K(x_i, x_j) : i, j \in \mathbb{N}_m)$, for every finite set of inputs $\mathbf{x} = \{x_j : j \in \mathbb{N}_m\} \subseteq \mathcal{X}$ and every $m \in \mathbb{N}$. Also, let $\mathcal{A}_+(\mathcal{X})$ denote the set of all kernels on the set $\mathcal{X}$ and $\mathcal{A}_{++}(\mathcal{X})$ the set of kernels K such that, for each $m \in \mathbb{N}$ and each choice of $\mathbf{x}$, $K_{\mathbf{x}} \in \mathbf{S}^m_{++}$.

Let $\{(x_j, y_j) : j \in \mathbb{N}_m\} \subseteq \mathcal{X} \times \mathbb{R}$ be prescribed data and y be the vector $(y_j : j \in \mathbb{N}_m)$. Assume also a prescribed convex nonnegative error function $Q : \mathbb{R}^m \to \mathbb{R}_+$. For example, we could have, for $w = (w_j : j \in N_m)$, that $Q(w) = \sum_{j \in \mathbb{N}_m} L(y_j, w_j)$, where L is a loss function. Note that we suppress the dependence of Q on y as the latter is fixed throughout our discussion. For every kernel K with associated Hilbert space $\mathcal{H}_K$ and every $f \in \mathcal{H}_K$, the standard L_2 *regularization functional* is defined as

$$\mathcal{E}_\gamma(f, K) := Q(I_{\mathbf{x}}(f)) + \gamma \|f\|_K^2 \tag{3.2.1}$$

where $\|f\|_K^2 := \langle f, f \rangle_K$ and γ is a positive constant. Recall that the operator $I_{\mathbf{x}}$ is defined as $I_{\mathbf{x}}(f) := (f(x_j) : j \in \mathbb{N}_m)$.

Associated with the functional $\mathcal{E}_\gamma$ and the kernel K is the variational problem

$$E_\gamma(K) := \min\{\mathcal{E}_\gamma(f, K) : f \in \mathcal{H}_K\} \tag{3.2.2}$$

which defines a functional $E_\gamma : \mathcal{A}_+(\mathcal{X}) \to \mathbb{R}_+$. We remark, in passing, that all of what we say about problem (3.2.2) applies to functions Q on $\mathbb{R}^m$ which are bounded from below as we can merely adjust the expression (3.2.1) by a constant independent of f and K. Note that the minimum in (3.2.2) is attained, as explained in Section 2.2.2. Moreover, if f is a solution to problem (3.2.2) then it has the form

$$f(x) = \sum_{j \in \mathbb{N}_m} \alpha_j K(x_j, x) \qquad \forall x \in \mathcal{X}, \tag{3.2.3}$$

for some real vector $\alpha = (\alpha_j : j \in \mathbb{N}_m)$. This result is known as the *Representer Theorem* – see Section 2.2.2. In particular, if Q is convex, the unique minimizer of problem (3.2.2) can be found by replacing f by the right hand side of equation (3.2.3) in equation (3.2.1) and then optimizing with respect to the vector α. That is, we have the finite dimensional variational problem

$$E_\gamma(K) := \min\{Q(K_{\mathbf{x}}\alpha) + \gamma\langle \alpha, K_{\mathbf{x}}\alpha\rangle : \alpha \in \mathbb{R}^m\} \tag{3.2.4}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product on $\mathbb{R}^m$. For example, when Q is the square loss defined for $w = (w_j : j \in \mathbb{N}_m) \in \mathbb{R}^m$ as $Q(w) = \|w - y\|^2 := \sum_{j \in \mathbb{N}_m} (w_j - y_j)^2$, the objective function in (3.2.4) is quadratic in the vector α and its minimizer is obtained by solving a linear system of equations.

As we have seen in Sections 3.1.1 and 3.1.2, a common approach to selecting good kernels or good combinations of kernels is to use the functional E_γ in (3.2.2) as a design criterion. That is, one can perform an optimization over *both the kernel K and the regression coefficients in vector α.* Theoretical justifications that the functional E_γ is a good quantity to optimize have appeared in the literature and will be briefly presented in Section 3.7. Here, we just remark that this optimization is justified provided that the kernel matrix $K_{\mathbf{x}}$ is bounded, in order to avoid overfitting.

Therefore, we may assume that a convex subset $\mathcal{K}$ of $\mathcal{A}_+(\mathcal{X})$ is given and study the problem

$$E_\gamma(\mathcal{K}) := \inf\{E_\gamma(K) : K \in \mathcal{K}\}\,. \tag{3.2.5}$$

We can view the set $\mathcal{K}$ as the convex hull of a set of *basic kernels* $\mathcal{B} \subseteq \mathcal{A}_+(\mathcal{X})$,

$$\mathcal{K} = \mathrm{conv}(\mathcal{B})\,.$$

This general formulation covers most of the proposed optimization problems in learning convex combinations of kernels, as well as hyperkernels, linear combinations of basic kernels (3.1.1), (3.1.2) and several other methods (see Section 3.2.6). An advantage of this formulation is that it can accommodate for infinite numbers of basic kernels (either countable or uncountable), for basic kernels defined through convex constraints etc. As for the necessary constraint on the size of the kernel, it is usually a bound on the trace of the kernel matrix, but other possibilities are allowed.

Every input set $\mathbf{x}$ and convex set $\mathcal{K}$ of kernels determines a convex set of matrices in $\mathbf{S}_+^m$, namely $\mathcal{K}(\mathbf{x}) := \{K_\mathbf{x} : K \in \mathcal{K}\}$. Obviously, it is this set of matrices that affects the variational problem (3.2.5). For this reason, we say that the set of kernels $\mathcal{K}$ is compact (convex) provided that for all $\mathbf{x}$ the set of matrices $\mathcal{K}(\mathbf{x})$ is compact (convex). The following result is taken directly from [Micchelli and Pontil, 2005] with a minor modification.

Lemma 54 *If $\mathcal{K}$ is a compact and convex subset of $\mathcal{A}_+(\mathcal{X})$ such that $K_\mathbf{x} \in \mathbf{S}_{++}^m$ for every $K \in \mathcal{K}$ and $Q : \mathbb{R}^m \to \mathbb{R}$ is continuous then the minimum of (3.2.5) exists.*

3.2.3 Convexity of Learning the Kernel

Next, we establish that if the loss function $Q : \mathbb{R}^m \to \mathbb{R}$ is convex then the functional $E_\gamma : \mathcal{A}_+(\mathcal{X}) \to \mathbb{R}_+$ is convex as well, that is, the variational problem (3.2.5) is a *convex problem with respect to K*. In this section we present a proof of this fact which expresses the functional as the maximum of functions convex in the kernel. This expression leads to a dual formulation of the problem and to the theoretical and algorithmic results of the following sections. An alternative proof, which is simpler but does not give an expression for E_γ, will be presented in Section 4.9. The significance of that proof lies in that it reveals connections between learning the kernel and multi-task learning.

To obtain the dual expression, we use the conjugate function of Q. Recall from Section 2.1.6 that the Fenchel conjugate $Q^* : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is defined, for every $v \in \mathbb{R}^m$, as

$$Q^*(v) = \sup\{\langle w, v\rangle - Q(w) : w \in \mathbb{R}^m\} \tag{3.2.6}$$

and it follows, for every $w \in \mathbb{R}^m$, that

$$Q(w) = \sup\{\langle w, v\rangle - Q^*(v) : v \in \mathbb{R}^m\}\,. \tag{3.2.7}$$

Note that Q^* is lower semicontinuous and convex. Also note that $Q^*(0) = -\inf\{Q(w) : w \in \mathbb{R}^m\} < +\infty$ since Q is bounded from below. This observation is used in the proof of the following lemma.

Lemma 55 *If $K \in \mathcal{A}_+(\mathcal{X})$, $K_\mathbf{x} \in \mathbf{S}_{++}^m$ and $Q : \mathbb{R}^m \to \mathbb{R}$ is a convex function then there holds the formula*

$$E_\gamma(K) = \max\left\{-\frac{1}{4\gamma}\langle c, K_\mathbf{x} c\rangle - Q^*(c) : c \in \mathbb{R}^m\right\}. \tag{3.2.8}$$

The fact that the maximum above is attained follows from the hypothesis that $K_\mathbf{x} \in \mathbf{S}_{++}^m$ and the fact that $Q^*(v) \geq -Q(0)$ for all $v \in \mathbb{R}^m$, which follows from equation (3.2.6). The proof of this lemma is based on a version of von Neumann's minimax theorem (see Appendix A).

We note that Q^* maps to the extended real line and hence equation (3.2.8) can encapsulate any type of convex constraints on c. For example, in the case of SVMs, the "box constraints" of the SVM dual appear as the domain in which Q^* takes real values.

Equation (3.2.8) expresses $E_\gamma(K)$ as the maximum of linear functions in the kernel K. Thus, the above lemma implies convexity in K.

Corollary 56 *Let $\mathcal{A}_{++}(\mathcal{X}, \mathbf{x}) := \{K : K \in \mathcal{A}_+(\mathcal{X}), K_\mathbf{x} \in \mathbf{S}^m_{++}\}$. Then the functional $E_\gamma : \mathcal{A}_{++}(\mathcal{X}, \mathbf{x}) \to \mathbb{R}_+$ is convex.*

3.2.4 Minimax Formulation and its Properties

Gathering the results of the previous section, we see that problem (3.2.5) reduces to the minimax problem

$$E_\gamma(\mathcal{K}) = -\max\{\min\{R(c, K) : c \in \mathbb{R}^m\} : K \in \mathcal{K}\} \tag{3.2.9}$$

where the function R is defined as

$$R(c, K) = \frac{1}{4\gamma}\langle c, K_\mathbf{x} c\rangle + Q^*(c)\,, \qquad \forall c \in \mathbb{R}^m,\ K \in \mathcal{K}. \tag{3.2.10}$$

We now proceed to show that problem (3.2.9) admits a saddle point and hence that the minimum and maximum in (3.2.9) can be interchanged and describe the properties of this saddle point. We consider this problem in the general case that $\mathcal{B}$ is any *compact* set of basic kernels. That is, what we say applies to an infinite, countable or not, set of basic kernels. Thereafter, we specialize our main result to the well-studied case of a finite set of basic kernels. As an example, we also provide a corollary for the case of isotropic Gaussian basic kernels whose parameter takes values in $\mathbb{R}_+$.

The main theorem states that the minimum and maximum in problem (3.2.9) can be interchanged and that the "simplest" solution can be represented with at most $m + 1$ basic kernels. It is an extension of the theorem in [Argyriou et al., 2005] to more general classes of loss functions. It also extends the result in [Micchelli and Pontil, 2005] where only the square loss function was studied in detail. Recall from Section 2.1 the notation $f'(c; \delta),\ c, \delta \in \mathbb{R}^m$, for the right directional derivative of function f at c along direction δ. Let us also define

$$\mathcal{C} := \operatorname{dom} Q^*\,,$$

the set where Q^* is finite.

Theorem 57 *Let $\mathcal{B} \subseteq \mathcal{A}_+(\mathcal{X})$ be a given compact set of basic kernels, such that $B_\mathbf{x} \in \mathbf{S}^m_{++}$ for every $B \in \mathcal{B}$, and let $\mathcal{K} = \operatorname{conv}(\mathcal{B})$. Then there exist $\mu \le m + 1$ kernels $\{B_i, i \in \mathbb{N}_\mu\} \subseteq \mathcal{B}$ and coefficients $\{\lambda_i, i \in \mathbb{N}_\mu\} \subseteq (0, 1]$, $\sum_{i\in\mathbb{N}_\mu} \lambda_i = 1$, such that*

$$R(\hat{c}, B_i) = \max\{R(\hat{c}, B) : B \in \mathcal{B}\} \qquad \forall i \in \mathbb{N}_\mu\,, \tag{3.2.11}$$

where $\hat{c} \in \mathcal{C}$ is the unique solution of the inequalities

$$\frac{1}{2\gamma}\langle \hat{K}_\mathbf{x}\hat{c}, \delta\rangle + {Q^*}'(\hat{c}; \delta) \ge 0 \qquad \forall \delta \in \mathbb{R}^m \tag{3.2.12}$$

and

$$\hat{K} = \sum_{i\in\mathbb{N}_\mu} \lambda_i B_i\,. \tag{3.2.13}$$

Moreover, $(\hat{c}, \hat{K})$ is a saddle point, that is, for every $c \in \mathbb{R}^m$ and $K \in \mathcal{K}$,

$$R(\hat{c}, K) \leq R(\hat{c}, \hat{K}) \leq R(c, \hat{K})\,. \tag{3.2.14}$$

PROOF. Let us first comment on the nonlinear inequalities (3.2.12). For *any* kernel $K \in \mathcal{K}$ the extremal problem

$$\min\{R(c,K) : c \in \mathcal{C}\}$$

has a *unique* solution, since the function $c \mapsto R(c,K)$ is lower semicontinuous and strictly convex and $\liminf\limits_{\|c\|\to\infty} R(c,K) = +\infty$. Moreover, if we let $c_K \in \mathcal{C}$ be this minimizer, it satisfies the inequalities

$$\frac{1}{2\gamma}\langle K_{\mathbf{x}} c_K, \delta\rangle + {Q^*}'\langle c_K; \delta\rangle \geq 0 \qquad\qquad \forall \delta \in \mathbb{R}^m.$$

This condition is necessary and sufficient and thus equation (3.2.12) states that $\hat{c}$ minimizes $R(\cdot, \hat{K})$.

Now let us turn to the existence of the kernel $\hat{K}$. First, we define the functions $\varphi_B : \mathcal{C} \to \mathbb{R}$ as

$$\varphi_B(c) := R(c, B) \qquad\qquad \forall c \in \mathcal{C}, B \in \mathcal{B}$$

and the function $\varphi : \mathcal{C} \to \mathbb{R}$ as

$$\varphi(c) := \max\{\varphi_B(c) : B \in \mathcal{B}\} \qquad\qquad \forall c \in \mathcal{C}.$$

Observe that $\varphi(c) = \max\{R(c,K) : K \in \mathcal{K}\}$, for every $c \in \mathcal{C}$, because R is linear in the kernel. From the definition it is clear that φ is lower semicontinuous and strictly convex and $\liminf\limits_{\|c\|\to\infty} \varphi(c) = +\infty$. Hence, φ has a *unique* minimizer, which we denote with $\hat{c}$.

This minimizer is characterized by the fact that 0 is a subgradient of φ at $\hat{c}$. Let us define the set $\mathcal{B}^*$ as

$$\mathcal{B}^* := \{B : B \in \mathcal{B},\ \varphi_B(\hat{c}) = \varphi(\hat{c})\}\,.$$

Applying Theorem 36 we obtain a first order condition involving the subdifferentials of the functions φ_B with $B \in \mathcal{B}^*$, namely that

$$0 \in \operatorname{cl}\mathcal{G}\,,$$

where

$$\mathcal{G} := \operatorname{conv}\left(\bigcup_{B\in\mathcal{B}^*} \partial\varphi_B(\hat{c})\right)\,.$$

For the definition of the subdifferential of a convex function see Section 2.1. Consequently, there is a sequence

$$\{g_n \in \mathcal{G} : n \in \mathbb{N}\}$$

such that

$$g_n \to 0\,. \tag{3.2.15}$$

By Carathéodory's theorem (Theorem 24), every vector in $\mathcal{G}$ can be expressed as a convex combination of at most $m+1$ of the elements of $\mathcal{G}$. In particular, we have that

$$g_n = \sum_{i\in\mathbb{N}_{\mu_n}} \lambda_{n,i}\, g_{n,i} \qquad \forall n \in \mathbb{N}$$

for some subgradients $g_{n,i} \in \partial\varphi_{B_{n,i}}(\hat{c})$, (not necessarily distinct) kernels $B_{n,i} \in \mathcal{B}^*$, $\lambda_{n,i} \in [0,1]$, $\forall i \in \mathbb{N}_{\mu_n}$, with $\sum_{i\in\mathbb{N}_{\mu_n}} \lambda_{n,i} = 1$ and $\mu_n \leq m+1, \forall n \in \mathbb{N}$. Applying Theorem 32, we obtain that

$${\varphi_{B_{n,i}}}'(\hat{c};\delta) \geq \langle g_{n,i}, \delta\rangle \qquad \forall \delta \in \mathbb{R}^m, n \in \mathbb{N}, i \in \mathbb{N}_{\mu_n}$$

and hence

$$\sum_{i\in\mathbb{N}_{\mu_n}} \lambda_{n,i}\, {\varphi_{B_{n,i}}}'(\hat{c};\delta) \geq \langle g_n, \delta\rangle \qquad \forall \delta \in \mathbb{R}^m, n \in \mathbb{N}\,.$$

Now we set

$$K_n := \sum_{i\in\mathbb{N}_{\mu_n}} \lambda_{n,i} B_{n,i} \qquad \forall n \in \mathbb{N} \tag{3.2.16}$$

and, because the directional derivative of R with respect to c is linear in the kernel, we obtain that

$$\frac{1}{2\gamma}\langle (K_n)_{\mathbf{x}}\hat{c}, \delta\rangle + {Q^*}'(\hat{c};\delta) \geq \langle g_n, \delta\rangle \qquad \forall \delta \in \mathbb{R}^m, n \in \mathbb{N}\,. \tag{3.2.17}$$

Since $\mathcal{B}$ is compact, the sets $\mathcal{B}^*$ and $\operatorname{conv}\mathcal{B}$ are compact as well, by Theorem 25. Thus, all the sequences

$$\{K_n : n \in \mathbb{N}\},\ \{B_{n,i} : n \in \mathbb{N}\},\ \{\lambda_{n,i} : n \in \mathbb{N}\}, \qquad i \in \mathbb{N}_{m+1}$$

(where we have added arbitrary basic kernels with zero coefficients wherever necessary) have convergent subsequences. We can extract such subsequences successively to obtain convergent subsequences

$$\{K_{n_\ell} : \ell \in \mathbb{N}\},\ \{B_{n_\ell,i} : \ell \in \mathbb{N}\},\ \{\lambda_{n_\ell,i} : \ell \in \mathbb{N}\}, \qquad i \in \mathbb{N}_{m+1}\,.$$

We are going to show that

$$\hat{K} := \lim_{\ell\to\infty} K_{n_\ell}$$

is an optimal kernel with the wanted properties. First, taking the limits as $\ell \to \infty$ in (3.2.16) and using the fact that $\mathcal{B}^*$ is closed, we see that $\hat{K}$ can be written as the convex combination of $m+1$ kernels in $\mathcal{B}^*$,

$$\hat{K} = \sum_{i\in\mathbb{N}_\mu} \lambda_i B_i,$$

for some $B_i \in \mathcal{B}^*, \lambda_i \in [0,1]$, $i \in \mathbb{N}_\mu$, $\sum_{i\in\mathbb{N}_\mu} \lambda_i = 1, \mu \leq m+1$. By definition of $\mathcal{B}^*$, the kernels B_i also satisfy equation (3.2.11). Next, taking the limits in (3.2.17) as $\ell \to \infty$ and using (3.2.15), it follows that

$$\frac{1}{2\gamma}\langle \hat{K}_{\mathbf{x}}\hat{c}, \delta\rangle + {Q^*}'(\hat{c};\delta) \geq 0 \qquad \forall \delta \in \mathbb{R}^m.$$

This establishes equation (3.2.12) and implies that

$$\min\{R(c,\hat{K}) : c \in \mathcal{C}\} = R(\hat{c},\hat{K})\,,$$

which is the upper inequality in (3.2.14). For the lower inequality we observe that

$$\max\{R(\hat{c}, K) : K \in \mathcal{K}\} = \max\{R(\hat{c}, B) : B \in \mathcal{B}\}$$

and the right hand side equals

$$\varphi(\hat{c}) = R(\hat{c}, B_i) \qquad \forall i \in \mathbb{N}_\mu\,.$$

Since R is linear in the kernel,

$$R(\hat{c}, \hat{K}) = \sum_{i \in \mathbb{N}_\mu} \lambda_i R(\hat{c}, B_i)$$

and (3.2.14) follows. ■

In the special case that Q^* is finite on $\mathbb{R}^m$ and differentiable (for example, when Q is the square loss), the inequalities (3.2.12) become just an equality condition

$$\frac{1}{2\gamma} \hat{K}_{\mathbf{x}} \hat{c} + \nabla Q^*(\hat{c}) = 0\,. \tag{3.2.18}$$

Unfortunately, for many interesting parameterizations, this condition yields a nonlinear equation in terms of the parameters of the basic kernels and may be hard to solve.

To summarize, the above theorem states two facts. First, that the minimization and maximization of the objective R in (3.2.9) can be interchanged and that there is a saddle point. Secondly, the theorem states that there is an optimal kernel which has a representation of at most $m + 1$ basic kernels. This is a consequence of Carathéodory's Theorem 24, which is a basic fact about convex sets.

We see that the kernel that solves problem (3.2.5) may be a convex combination of complementary kernels, each giving a higher value of E_γ. Thus, a convex combination of kernels may yield better results on the data than each of the basic kernels does separately. Note also that while $\hat{c}$ solves the regularization problem (SVM, ridge regression etc.) for $\hat{K}$, the solutions of the problems corresponding to each basic kernel are, in general, different.

The proof of Theorem 57 we have presented is not the only possible one. An alternative proof using the Fenchel duality theorem has been proposed in [Rifkin and Lippert, 2007]. Probably the most concise proof would be one using a minimax theorem like Theorem 43 to show the existence of a saddle point and a simple argument based on Carathéodory's Theorem for the representation result. However, we have opted for a proof that is constructive in order to better illustrate what the set of saddle points is and how they can be obtained.

Regarding this question, there may exist optimal kernels which can be represented as $\hat{K} = \sum_{i \in \mathbb{N}_\mu} \lambda_i B_i$, with $m + 1 < \mu \leq \dfrac{m(m+1)}{2} + 1$. The upper bound of $\dfrac{m(m+1)}{2} + 1$ comes from the dimensionality of the set of kernel matrices $\{K_{\mathbf{x}} \in \mathbf{S}^m_{++} : K \in \mathcal{K}\}$. Theorem 57 merely ensures that there is *at least one* kernel with $\mu \leq m + 1$. At the same time, the proof of this theorem gives a way to construct any optimal kernel matrix. With this caveat in mind, we now show that the two properties stated in Theorem 57 are necessary and sufficient.

Theorem 58 *Let the assumptions of Theorem 57 hold. Then, a point* $(\hat{c}, \hat{K}) \in \mathbb{R}^m \times \mathcal{K}$ *is a solution of problem* (3.2.9) *if and only if there exist* $\{B_i : i \in \mathbb{N}_\mu\} \subseteq \mathcal{B}$ *such that* $\mu \leq \frac{m(m+1)}{2} + 1$, $\hat{K} = \sum_{i \in \mathbb{N}_\mu} \lambda_i B_i$ *and conditions* (3.2.11) *and* (3.2.12) *are satisfied. Moreover, the set of solutions of* (3.2.9) *is obtained from the* unique $\hat{c}$ *and the convex set of* $\hat{K}$ *that solve* (3.2.12).

PROOF. If $(\hat{c}, \hat{K})$ is a saddle point of (3.2.9) then $\hat{c}$ is a minimizer of the function $R(\cdot, \hat{K})$ and hence satisfies (3.2.12). Moreover, we have that $\sum_{i \in \mathbb{N}_\mu} \lambda_i R(\hat{c}, B_i) = R(\hat{c}, \hat{K}) = \max\{R(\hat{c}, K) : K \in \mathcal{K}\}$ implying equation (3.2.11). On the other hand, if $\hat{c}$ satisfies the inequalities (3.2.12) we obtain the upper inequality in (3.2.14), whereas equation (3.2.11) brings the lower inequality. The uniqueness of $\hat{c}$ can be seen from its construction in the proof of Theorem 57. ■

In general, we are interested in finding concise solutions of the minimax problem (3.2.9) in the sense of representability with a small number of basic kernels. However, solutions with larger representations are also valid and could be plausible for the learning task of interest.

We now specialize Theorem 57 to some important cases. The first of them is when $\mathcal{B}$ is a finite set of prescribed basic kernels, $\mathcal{B} = \{B_\ell : \ell \in \mathbb{N}_n\}$. Below, we use the notation $K_{\mathbf{x},\ell}$ for the matrix $(K_\ell)_{\mathbf{x}}$.

Corollary 59 *Assume that* $\mathcal{B} = \{B_\ell : \ell \in \mathbb{N}_n\} \subseteq \mathcal{A}_+(\mathcal{X})$ *and* $B_{\mathbf{x},\ell} \in \mathbf{S}^m_{++}$ *for every* $\ell \in \mathbb{N}_n$*. Then there exist* $I \subseteq \mathbb{N}_n$*, a kernel* $\hat{K} = \sum_{i \in I} \lambda_i B_i$*, where* $\lambda_i \in (0, 1]\, \forall i \in I, \sum_{i \in I} \lambda_i = 1$*, and* $\hat{c} \in \mathbb{R}^m$*, such that card*$(I) \leq \min(m + 1, n)$*,*

$$R(\hat{c}, B_i) = \max\{R(\hat{c}, B_\ell) : \ell \in \mathbb{N}_n\} \qquad \forall i \in I$$

and

$$\frac{1}{2\gamma} \langle \hat{K}_{\mathbf{x}} \hat{c}, \delta \rangle + Q^{*'}(\hat{c}; \delta) \geq 0 \qquad \forall \delta \in \mathbb{R}^m .$$

Moreover, for every $c \in \mathbb{R}^m$ *and* $K \in \operatorname{conv}(\mathcal{B})$ *we have that*

$$R(\hat{c}, K) \leq R(\hat{c}, \hat{K}) \leq R(c, \hat{K}).$$

Another important case is when each kernel in $\mathcal{B}$ is a Gaussian kernel, $B(x, z) = e^{-\omega \|x - z\|^2}, x, z \in \mathbb{R}^d$, and $\omega \in [\omega_1, \omega_2]$ where $0 < \omega_1 < \omega_2$. Theorem 57 establishes that a mixture of at most $m + 1$ Gaussian kernels provides an optimal kernel. What happens if we consider all possible Gaussians, that is, allow $\omega \in \mathbb{R}_+$? This question is important because Gaussians generate the *whole class of radial kernels*. Indeed, we recall a beautiful result by I.J. Schoenberg [Schoenberg, 1938].

Theorem 60 *Let* h *be a real-valued function defined on* $\mathbb{R}_+$ *such that* $h(0) = 1$*. We form a kernel* K *on* $\mathbb{R}^d \times \mathbb{R}^d$ *by setting, for each* $x, z \in \mathbb{R}^d$*,* $K(x, z) = h(\|x - z\|^2)$*. Then* K *is positive definite for* any d *if and only if there is a probability measure* p *on* $\mathbb{R}_+$ *such that*

$$K(x, z) = \int_{\mathbb{R}_+} e^{-\omega \|x - z\|^2} dp(\omega) \qquad \forall x, z \in \mathbb{R}^d.$$

Note that the set $\mathbb{R}_+$ is *not* compact and the identically one kernel ($\omega = 0$) is not positive definite on any data $\mathbf{x}$. Therefore, on both accounts Theorem 57 does not apply in this circumstance. In general, we may overcome this difficulty by a limiting process which can handle *locally compact* sets of kernels. This will lead us to an extension of Theorem 57 where $\mathcal{B}$ is locally compact. However, we only describe our approach in detail for the Gaussian case and $\Omega = \mathbb{R}_+$. An important ingredient in the discussion presented below is that the limit of the Gaussian kernel as $\omega \to +\infty$ is Δ, the delta kernel. In other words, we need to include the delta kernel in the set of basic kernels for the theorem to work.

Theorem 61 *Let $\mathcal{B} = \{\Gamma_\omega : \omega \in \mathbb{R}_+\} \cup \{\Delta\}$ where $\Gamma_\omega \in \mathcal{A}_+(\mathbb{R}^d)$ are defined as*

$$\Gamma_\omega(x, z) = e^{-\omega\|x-z\|^2} \qquad \forall x, z \in \mathbb{R}^d,\ \omega \in \mathbb{R}_+.$$

Then there exist $\mu \leq m+1$ kernels $\{B_i : i \in \mathbb{N}_\mu\} \subseteq \mathcal{B}$ and coefficients $\{\lambda_i : i \in \mathbb{N}_\mu\} \subseteq (0, 1]$, $\sum_{i\in\mathbb{N}_\mu} \lambda_i = 1$, such that

$$R(\hat{c}, B_i) = \max\{R(\hat{c}, B) : B \in \mathcal{B}\} \qquad \forall i \in \mathbb{N}_\mu, \tag{3.2.19}$$

for some $\hat{c} \in \mathbb{R}^m$ satisfying the inequalities

$$\frac{1}{2\gamma}\langle \hat{K}_{\mathbf{x}}\hat{c}, \delta\rangle + Q^{*'}(\hat{c}; \delta) \geq 0 \qquad \forall \delta \in \mathbb{R}^m \tag{3.2.20}$$

and

$$\hat{K} = \sum_{i\in\mathbb{N}_\mu} \lambda_i B_i\,.$$

Moreover, $(\hat{c}, \hat{K})$ is a saddle point, that is, for every $c \in \mathbb{R}^m$ and $K \in \mathcal{K}$,

$$R(\hat{c}, K) \leq R(\hat{c}, \hat{K}) \leq R(c, \hat{K})\,. \tag{3.2.21}$$

PROOF. For every $\ell \in \mathbb{N}$ we consider the Gaussian kernels on the interval $\Omega_\ell := [\ell^{-1}, \ell]$ and appeal to Theorem 57 to produce a sequence of kernels $\hat{K}_\ell = \sum_{i\in\mathbb{N}_{m+1}} \lambda_{\ell,i}\Gamma_{\omega_{\ell,i}}$ and $\hat{c}_\ell \in \mathbb{R}^m$ with the properties described there. Let us examine what happens as ℓ tends towards infinity. All the kernel matrices on the data $\mathbf{x}$ are bounded since they correspond to Gaussian kernels. Thus, each of the kernel sequences $\{\Gamma_{\omega_{\ell,i}} : \ell \in \mathbb{N}\}$ as well as their corresponding weights have subsequences which converge. Some kernel subsequences may converge to Γ_0 while others to Δ. In any case, we can extract convergent subsequences $\{\lambda_{n_\ell,i} : \ell \in \mathbb{N}\}$ of coefficients and $\{\Gamma_{\omega_{n_\ell,i}} : \ell \in \mathbb{N}\}$ of kernels, such that $\lim_{\ell\to\infty} \lambda_{n_\ell,i} = \hat{\lambda}_i$, $\lim_{\ell\to\infty} \hat{K}_{n_\ell} = \hat{K}$, $\lim_{\ell\to\infty} \Gamma_{\omega_{n_\ell,i}} = \hat{B}_i$ and $\hat{K} = \sum_{i\in\mathbb{N}_{m+1}} \hat{\lambda}_i\hat{B}_i$, with the provision that $\hat{B}_i$ may take the values Γ_0 or Δ.

To establish that $\hat{K}$ is an optimal kernel, we turn our attention to the sequence of vectors $\hat{c}_{n_\ell}$. We claim that this sequence also has a convergent subsequence. Indeed, from inequality (3.2.14), for every $K \in \mathcal{K}$ we have that

$$R(\hat{c}_{n_\ell}, K) \leq R(0, \hat{K}_{n_\ell}) = Q^*(0) < +\infty.$$

Using the fact that the function Q^* is bounded from below (see our comments after the proof of Lemma 55) we see that the sequence $\hat{c}_{n_\ell}$ has bounded Euclidean norm independently of ℓ. Hence, it has a convergent subsequence whose limit we call $\hat{c}$. Taking the limits in (3.2.14) as $\ell \to \infty$, we obtain inequality (3.2.21), from which (3.2.19) and (3.2.20) also follow. ■

3.2.5 Special Cases of Interest

We now consider some possibilities for the set $\mathcal{B}$ of basic kernels which give rise to tractable problems.

Finite Number of Basic Kernels

Assume that $\mathcal{B} = \{B_\ell : \ell \in \mathbb{N}_n\}$. Then from Theorems 57 and 58, we observe that problem (3.2.9) is equivalent to

$$E_\gamma(\mathcal{K}) = -\min\{\max\{R(c, B_\ell) : \ell \in \mathbb{N}_n\} : c \in \mathbb{R}^m\}$$

or

$$E_\gamma(\mathcal{K}) = -\min\{\theta : \theta \in \mathbb{R}, c \in \mathbb{R}^m, \theta \geq R(c, B_\ell) \ \forall \ell \in \mathbb{N}_n\}. \tag{3.2.22}$$

Each of the functions $R(\cdot, B_\ell)$ is convex on $\mathbb{R}^m$ and hence there is a finite number of convex constraints in (3.2.22). Thus, we have the following.

Corollary 62 *If $\mathcal{B}$ is finite then* (3.2.5) *is a convex optimization problem.*

For hinge-like loss functions, formulation (3.2.22) is essentially the same as that derived in [Lanckriet et al., 2004, Thms. 17 etc.]. In these cases, the optimization problem becomes a quadratic program which is a type of *semidefinite program* (SDP). In fact, it is clear from the above that the problem can be written as an SDP for any semidefinite representable function Q^*.

An approach to solving problem (3.2.22) is by using standard methods for SDPs, quadratic programs or general constrained convex programs. Such a method gives the optimal values for c and t. Subsequently, the coefficients λ_i can be obtained from the primal problem (3.2.4) or from condition (3.2.12), which require solving a linear system of equations in the smooth case or a linear program in the case of SVMs.

Another approach, followed in [Sonnenburg et al., 2006], is to use methods (such as column generation) for *semi-infinite programming*. This is a direct consequence of the minimax nature of the problem:

$$E_\gamma(\mathcal{K}) = -\max\left\{\min\left\{R\left(c, \sum_{\ell \in \mathbb{N}_n} \lambda_\ell B_\ell\right) : c \in \mathbb{R}^m\right\} : \lambda_\ell \in [0,1], \sum_{i \in \mathbb{N}_n} \lambda_\ell = 1\right\}$$

or

$$E_\gamma(\mathcal{K}) = -\max\left\{\theta \ : \ \sum_{\ell \in \mathbb{N}_n} \lambda_\ell R(c, B_\ell) \geq \theta \ \forall c \in \mathbb{R}^m, \sum_{i \in \mathbb{N}_n} \lambda_\ell = 1, \lambda_\ell \in [0,1], \theta \in \mathbb{R}\right\}.$$

There is an infinite number of linear constraints above and thus this problem is a semi-infinite linear program. The algorithm proposed in the aforementioned work is to iteratively select a subset of values for c, solve for λ_ℓ and θ, then, with fixed λ_ℓ, minimize $R(c, B_\ell)$ over c and so on.

In Section 3.3, we present another algorithm for solving the problem in the finite setting, which is a specialization of an algorithm for general families of basic kernels.

There are some cases which appear to be infinite at first sight, but reduce to some finite case. This happens when the set $\mathcal{B}$ is *finitely generated* by a set of μ basic kernels $\{B_i : i \in \mathbb{N}_\mu\} \subseteq \mathcal{B}$. Note that here μ can be larger than the bound $\frac{m(m+1)}{2} + 1$ from Carathéodory's Theorem. In such cases, the corresponding optimization problems (3.2.9) are again convex. Such an example is the set of *polynomial* basic kernels of bounded degree, $B_\alpha(x, z) = \sum_{i \in \mathbb{N}_n} \alpha_i \langle x, z \rangle^i$, $\forall x, z \in \mathcal{X}$, where $\alpha_i \geq 0 \ \forall i \in \mathbb{N}_n$ and the parameter space of the α's is a polytope.

Infinite Number of Basic Kernels

In the more general case when $\operatorname{conv}(\mathcal{B})$ has infinitely many extreme points, standard methods for convex programming are not applicable. Indeed, in such a case there are infinitely many constraints in (3.2.22). One could conceive a semi-infinite type of iterative algorithm, such as one that selects a finite subset of the constraints at each iteration. However, a finite-number-of-kernels minimax problem would have to be solved at each iteration. In the alternate step, the objective R would be maximized over $\mathcal{B}$, which in many cases is a hard problem. Moreover, after the end of the iterative process, a linear system or program would be needed to recover the λ_i in (3.2.13).

There is a more efficient approach we can follow, which we shall present in Section 3.3. Our algorithm constructs the optimal convex combination of kernels iteratively, by adding one kernel at a time. One advantage is that the optimal kernel is available after the last iteration and no further optimization is needed. More importantly, it turns out that a 2-kernels minimax problem has to be solved at each iteration and this is an one-dimensional convex program. Still, the maximization step over $\mathcal{B}$ remains and this is where most of the difficulty of the problem lies. This step is not convex, in general, although it may be tractable in certain cases, especially if the number of kernel parameters is small.

Finally, there are some classes of basic kernels with infinite cardinality which lead to tractable optimization problems. We show a few such examples in Section 3.4.1.

3.2.6 Related Problems

We briefly note that a variety of problems which optimize a functional of the kernel can be treated using the methodology of the previous sections. For instance, the *hyperkernels* approach (see Section 3.1.1) optimizes a functional of kernels belonging to a hyper-reproducing kernel Hilbert space and penalizes their norms in this space. Essentially, this is an optimization problem similar to (3.2.5) with $\mathcal{K}$ being a compact and convex subset of the cone generated by the kernels $\{\bar{K}(\bar{x}, (\cdot, \cdot)) : \bar{x} \in \mathcal{X} \times \mathcal{X}\}$. Of course, some type of continuity assumption on $\bar{K}$ needs to be imposed, to ensure closedness of $\mathcal{B}$.

A possible variation on our optimization framework could be to allow for loss functions Q whose domain is a closed convex subset of $\mathbb{R}^m$. That is, we could constrain the values that the functions $f \in \mathcal{H}_K$ in (3.2.2) take on the data. The results of Sections 3.2.3 and 3.2.4 should hold, under the assumption that Q is lower semicontinuous (see Section 2.1.2).

Our framework of Section 3.2.2 can also be applied as is, or with small modifications, to other formulations for learning problems. These include, for example, one-class SVM [Tax and Duin, 2004];

kernel PCA [Schölkopf and Smola, 2002]; maximum entropy discrimination [Jaakkola et al., 1999].

Finally, we note that [Kim et al., 2006] have recently established the convexity of selecting a kernel from a convex set in kernel Fisher discriminant analysis (this situation does not fall under our framework).

3.2.7 Examples of Kernel Parameterizations

Before we describe and study our algorithm for solving problem (3.2.9), it is useful to briefly discuss what the set of basic kernels $\mathcal{B}$ may be in some cases of interest. As we have already mentioned, we are often interested in sets denser than finite sets of basic kernels.

Frequently, the class of kernels defining our hypothesis space is defined by a parameterization, that is, a functional form which depends on one or more parameters, along with a range for the values of these parameters. Formally, we may assume a (locally) compact Hausdorff space of parameters Ω (see, for example, [Royden, 1988]). The basic kernels are given by a continuous mapping $G : \Omega \to \mathcal{A}_+(\mathcal{X})$, that is,

$$\mathcal{B} = \{G(\omega) : \omega \in \Omega\}\,.$$

We also need to assume that the function $\omega \mapsto G(\omega)(x, z)$ is continuous on Ω for each $x, z \in \mathcal{X}$. Under these assumptions the requirements of Theorems 57 and 61 are satisfied.

Note that the (local) compactness of $\mathcal{B}$ is a consequence of the weak*-compactness of the unit ball in the dual space of $C(\Omega)$, the set of all continuous real-valued functions g on Ω with norm $\|g\|_\Omega := \max\{|g(\omega)| : \omega \in \Omega\}$ – see [Royden, 1988]. We can also use $\mathcal{K}(G)$ to denote $\mathcal{K} = \mathrm{conv}(\mathcal{B})$.

For example, the choice of $\Omega = \mathbb{N}_n$ corresponds to the special case of a finite number of basic kernels. As other examples, we may choose $\Omega = [\omega_1, \omega_2]$, where $0 < \omega_1 < \omega_2$ and $G(\omega)(x, z) = e^{-\omega\|x-z\|^2}$, $\forall x, z \in \mathbb{R}^d, \omega \in \Omega$, to obtain *radial kernels*, or $G(\omega)(x, z) = e^{\omega\langle x,z\rangle}$ to obtain *dot product kernels*. The choice of the mapping G is a matter of a priori knowledge or guesswork that depends on the learning task to be solved. General-purpose kernels such as the above can be used and these have a standard form and parameterization. Ad hoc kernels appropriate for the learning task can also be used. These may be parameterized simply by an index but they may also involve additional parameters that, for example, balance different effects. Thus, one could even consider infinite numbers of ad hoc basic kernels.

Finally, let us make a few observations about Gaussian basic kernels with more than one parameters. Let $\Sigma \in \mathbf{S}^d_{++}$ be a $d \times d$ positive definite matrix. A basic kernel can be built from the Gaussian kernel

$$G(\Sigma)(x, z) = e^{-\langle (x-z), \Sigma^{-1}(x-z)\rangle} \qquad \forall x, z \in \mathbb{R}^d. \tag{3.2.23}$$

The case $\Sigma = \sigma I, \sigma \in \mathbb{R}_{++}$, has already been mentioned above. Clearly, Σ needs to lie in a compact set and moreover, to ensure that Σ plays the role of a covariance, we must bound its determinant away from zero. A special case is provided by a block diagonal covariance

$$\Sigma = \mathrm{diag}(\Sigma_1, \ldots, \Sigma_\ell),$$

where each Σ_i is a $d_i \times d_i$ matrix, $i \in \mathbb{N}_\ell$, and $\sum_{i \in \mathbb{N}_\ell} d_i = d$. We write x as $x = (x^i, i \in \mathbb{N}_\ell)$, where $x^i \in \mathbb{R}^{d_i}$. Thus, we obtain basic kernels of the form

$$G(\Sigma)(x,z) = \prod_{i \in \mathbb{N}_\ell} e^{-\langle (x^i - z^i), \Sigma_i^{-1}(x^i - z^i)\rangle} \qquad \forall x, z \in \mathbb{R}^d.$$

Another example is the case of a full two-dimensional covariance. For this purpose, we write $\Sigma^{-1} = UMU^\top$ where $M = \left(\begin{smallmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{smallmatrix}\right)$, $0 < \mu_2 \le \mu_1$ and U is unitary, i.e. $U_{11} = U_{22} = \cos\theta$, $U_{12} = -U_{21} = \sin\theta$. A direct computation gives that

$$\langle x, \Sigma^{-1} x\rangle = \left(\frac{\mu_1 + \mu_2}{2} + \frac{\mu_2 - \mu_1}{2}\cos(2\theta - \zeta)\right)\|x\|^2 \qquad \forall x \in \mathbb{R}^2,$$

where ζ depends only on x. Using this formula and integrating (3.2.23) over the parameters θ, μ_1, μ_2 with the measure $\frac{1}{4} W_1(\frac{\mu_1+\mu_2}{2}) W_2(\frac{\mu_2-\mu_1}{2}) d\theta d\mu_1 d\mu_2$ and appropriate W_1, W_2, we obtain that kernels in $\mathcal{K}(G)$ are of the form

$$K(x,z) = H_1(\|x-z\|^2) H_2(\|x-z\|^2) \qquad \forall x, z \in \mathbb{R}^2,$$

where

$$H_1(t) = \int_0^\infty e^{-tu} W_1(u) du, \quad H_2(t) = \pi \int_0^\infty I_0(tv) W_2(v) dv$$

and $I_0(t) = \frac{1}{\pi}\int_{-\pi}^{\pi} e^{-t\cos\theta} d\theta$, the modified Bessel function of order zero.

3.3 Greedy Algorithm for Learning Convex Combinations of Kernels

The analysis in Section 3.2.4 establishes necessary and sufficient conditions for a pair $(\hat{c}, \hat{K}) \in \mathbb{R}^m \times \mathcal{K}$ to be a saddle point of the problem

$$-E_\gamma(\mathcal{K}) := \max\left\{\min\left\{R(c,K) : c \in \mathbb{R}^m\right\} : K \in \mathcal{K}\right\}.$$

The main part of this problem is to compute the optimal kernel $\hat{K}$. Indeed, once $\hat{K}$ has been computed, $\hat{c}$ is obtained as the unique solution of the inequalities (3.2.12).

With this observation in mind, in this section we focus on a computational method for the problem

$$E_\gamma(\mathcal{K}) = \min\{E_\gamma(K) : K \in \mathcal{K}\}$$

where recall that

$$E_\gamma(K) := -\min\left\{R(c,K) : c \in \mathbb{R}^m\right\} \qquad \forall K \in \mathcal{A}_+(\mathcal{X}).$$

The method we propose is a greedy algorithm that iteratively builds a convex combination of basic kernels. The algorithm starts with an initial kernel $K^{(1)} \in \mathcal{K}$ and computes iteratively a sequence of kernels $K^{(k)} \in \mathcal{K}$ and their corresponding vectors $c^{(k)}$ that minimize $R(\cdot, K^{(k)})$. At each iteration k, the algorithm searches for a basic kernel $\hat{B} \in \mathcal{B}$, if any, such that

$$\langle c^{(k)}, \hat{B}_{\mathbf{x}} c^{(k)}\rangle \tag{3.3.1}$$

Algorithm 1 Algorithm to compute an optimal convex combination of basic kernels.

```
Initialization: Choose $K^{(1)} \in \mathcal{K}$
Set k = 1
while $\|E_\gamma(K^{(k)}) - E_\gamma(K^{(k-1)})\| < tol$ do
    1. Compute $c^{(k)} = \mathrm{argmin}\{R(c, K^{(k)}) : c \in \mathbb{R}^m\}$
    2. Select $\hat{B} \in \mathrm{argmax}\{\langle c^{(k)}, B_{\mathbf{x}} c^{(k)}\rangle : B \in \mathcal{B}\}$.
       If $\|\langle c^{(k)}, \hat{B}_{\mathbf{x}} c^{(k)}\rangle - \langle c^{(k)}, K_{\mathbf{x}}^{(k)} c^{(k)}\rangle\| < tol$, terminate.
    3. Compute $\hat{\lambda} = \mathrm{argmin}\{E_\gamma(\lambda \hat{B} + (1-\lambda)K^{(k)}) : \lambda \in (0,1]\}$
    4. Set $K^{(k+1)} = \hat{\lambda}\hat{B} + (1-\hat{\lambda})K^{(k)}$
       Set $k = k + 1$
end while
```

is maximized. If such $\hat{B}$ is found then a new kernel $K^{(k+1)}$ is computed to be the optimal convex combination of the kernels $\hat{B}$ and $K^{(k)}$, that is,

$$E_\gamma(K^{(k+1)}) = \min\left\{E_\gamma(\lambda\hat{B} + (1-\lambda)K^{(k)}) : \lambda \in [0,1]\right\}.$$

If no $\hat{B} \in \mathcal{B}$ maximizing (3.3.1) can be found, the algorithm terminates.

We shall see in the next section that, in this way, the values of the objective function E_γ decrease, as . Moreover, we shall show that the iterates $c^{(k)}, K^{(k)}$ concentrate around saddle points of problem (3.2.9).

Computationally, the hardest part of the algorithm is step 2, which maximizes the quadratic form (3.3.1). This form depends on the parameterization G of the basic kernels and will not be concave, in general. Different methods can be applied to this maximization problem, which may have few or many local maxima (see Section 3.4). Step 3 is a simple minimization problem which can be solved, for example, using Newton's method, since the function $E_\gamma(\lambda\hat{B} + (1-\lambda)K^{(k)})$ is convex in λ and its derivative can be computed by applying Theorem 35. We also use tolerance parameters to stop if there is small change in the quadratic form (3.3.1) or in the value of the objective $E_\gamma(K^{(k)})$. A version of the algorithm for the case of finite $\mathcal{B}$ can also be implemented (below we shall refer to this version as the "finite algorithm"). It only differs from the continuous version at Step 2, in that (3.3.1) needs to be computed on all the basic kernels of $\mathcal{B}$.

3.3.1 Convergence Properties

It can be shown that the greedy algorithm we have described approaches the optimal value of problem (3.2.9) at every iteration. This is a consequence of the following lemma.

Lemma 63 *Let $K_1, K_2 \in \mathcal{A}_+(\mathcal{X})$ such that $(K_1)_{\mathbf{x}}, (K_2)_{\mathbf{x}} \in \mathbf{S}^m_{++}$. Then $\lambda = 0$ is* not *a solution to the problem*

$$\min\{E_\gamma(\lambda K_1 + (1-\lambda)K_2) : \lambda \in [0,1]\}$$

if and only if $R(c_{K_2}, K_1) > R(c_{K_2}, K_2)$.

PROOF. Follows immediately from Theorem 58. ■

Applying this lemma to the case that $K_1 = \hat{B}$ and $K_2 = K^{(k)}$ we conclude that $E_\gamma(K^{(k+1)}) < E_\gamma(K^{(k)})$ if and only if $\hat{B}$ satisfies the inequality $R(c^{(k)}, \hat{B}) > R(c^{(k)}, K^{(k)})$, which is established by step 2. Thus after each iteration, either the objective function E_γ decreases or the algorithm terminates:

$$E_\gamma(K^{(1)}) > E_\gamma(K^{(2)}) > \cdots > E_\gamma(K^{(k_{max})}) \,. \tag{3.3.2}$$

Not only do the values of the objective converge but they also converge to the optimal value of (3.2.9). Moreover, the iterates of c and K concentrate near saddle points of the problem.

Theorem 64 *There exists a limit point of Algorithm 1 and any limit point* $(\bar{c}, \bar{K})$ *is a saddle point of problem* (3.2.9).

PROOF. See Appendix B. The proof is a consequence of the continuity of E_γ. ■

The assumption that the quadratic form is maximized at step 2 is crucial for optimality to hold. An implementation that simply finds a larger value of $\langle c^{(k)}, K_{\mathbf{x}}^{(k)} c^{(k)} \rangle$ will approach the optimal value of the problem at every iteration but is not guaranteed to converge to it. Indeed, one could conceive of an example with, say, 3 basic kernels, B_1, B_2, B_3, and a sequence of steps which converges (without attainment) to the saddle point of the subproblem defined by B_1 and B_2. That is, at no iteration is B_3 selected for inclusion in the combination of kernels. If B_3 is such that $(B_3)_{\mathbf{x}} - (B_1)_{\mathbf{x}}, (B_3)_{\mathbf{x}} - (B_2)_{\mathbf{x}} \in \mathbf{S}^m_{++}$ then this process will always remain far from the optimal kernel, which is B_3.

3.3.2 Implementation Issues

Step 1 in Algorithm 1 is a standard regularization problem in a dual form. In the case that Q^* is everywhere finite and differentiable, equation (3.2.18) can be used for computing $c^{(k)}$. In general, a constrained optimization problem may need to be solved instead. For example, in the case of SVMs a standard SVM problem has to be solved. Moreover, step 3 is an one-dimensional maximization problem that can be solved in a small number of iterations with Newton's or Brent's method. However, at each iteration a regularization problem, such as an SVM, needs to be solved. Thus, the total computational cost of learning the optimal kernel with an SVM is proportional to the cost of SVM learning. The experimental results in Section 3.5 indicate that k_{max} is small (usually less than 20). To improve efficiency one could use heuristics for speeding up each SVM problem. One idea is to use the result of an SVM run as the starting point for the next one. This should result in large improvements, at step 3 in particular. Another heuristic implementation could be similar to SVMlight, caching kernel matrix entries (see [Joachims, 1998]), but also indices of the sets selected by SMO and possibly other information, across consecutive SVM runs.

In the experiments of Section 3.5, we have verified that the most costly part of the algorithm is step 2, especially with multiple kernel parameters. To lower this computational cost a heuristic could be to reuse estimates, under the assumption that the form of the function does not change much between consecutive iterations.

3.4 Computational Issues

3.4.1 Tractable Cases

As we pointed out above, a key step in our algorithm is to maximize the objective function

$$q(\omega) = \langle c, G_{\mathbf{x}}(\omega) c \rangle \tag{3.4.1}$$

over $\omega \in \Omega$, where Ω is the parameter space and c is fixed. There are some cases when this optimization problem is tractable. Clearly, if q is a concave function of the parameters ω and Ω is a convex set, then the maximization is a convex program.

This situation can indeed arise from some kernel parameterizations. One example is the family of basic kernels

$$K(x, z) = \langle x, F(\Psi) z \rangle \qquad \forall x, z \in \mathbb{R}^d,$$

where $\Psi \in \mathbf{S}^d_{++}$ is the parameter matrix (corresponding to ω) and $F : \mathbf{S}^d_{++} \to \mathbf{S}^d_{++}$ is a *spectral* matrix function (see Section 2.3). In this case, $q(\omega)$ is equal to $\langle Xc, F(\Psi) Xc \rangle$, where $X = (x_1, \cdots, x_m)$ is the data matrix. It can easily be seen that q is concave (for every $c \in \mathbb{R}^m$) if and only if F is induced by a *matrix concave* function. For example, *linear kernels* $\langle x, \Psi z \rangle$, kernels such as $\langle x, \Psi^{\frac{1}{2}} z \rangle$, $\langle x, \log(\Psi + I) z \rangle$ etc. give rise to convex programs. Other similar cases can be kernels corresponding to the matrices $F(X^\top \Psi X), F(X^\top X \Psi)$ etc.

In fact, the larger optimization problem (3.2.5) is then *convex*, as a consequence of Theorem 52 and the variable transformation $\beta = K_{\mathbf{x}} \alpha$ in the primal (3.2.4). Convex optimization packages can be used for solving this problem, provided that $\operatorname{conv}(\mathcal{B})$ can easily be expressed as a finite set of convex constraints. However, in many situations this is not possible and we need to resort to Algorithm 1 as an efficient alternative.

3.4.2 Minimizing Sums of Exponentials

Now we present some insights into the case that the basic kernels are exponential functions, namely

$$G(\omega)(x, z) = e^{-\omega d(x,z)} \qquad \forall x, z \in \mathcal{X}$$

for some function $d : \mathcal{X}^2 \to \mathbb{R}_+$ and $\Omega \subseteq \mathbb{R}_+$. It is well known that $G(\omega)$ is a kernel for all $\omega \in \mathbb{R}_+$ if and only if the matrix $D = (d(x_i, x_j) : i, j \in \mathbb{N}_m)$ is conditionally negative definite for all data $\{x_i : i \in \mathbb{N}_m\}, m \in \mathbb{N}$, that is, $\langle c, Dc \rangle \leq 0$ whenever $\sum_{i \in \mathbb{N}_m} c_i = 0$. For example, in the Gaussian case, $d(x_i, x_j) = \|x_i - x_j\|^2$ has this property – see, for example, [Schölkopf and Smola, 2002].

We wish to bound the number of local extrema on $\mathbb{R}_+$ of the function

$$q(\omega) = \sum_{i \in \mathbb{N}_m} c_i^2 + 2 \sum_{i<j} c_i c_j e^{-\omega d(x_i, x_j)}.$$

For this purpose, let us define the univariate function $g(x) = \sum_{i \in \mathbb{N}_n} a_i e^{b_i x}$, $x \in \mathbb{R}$, where $a_i \in \mathbb{R}, b_1 < \cdots < b_n$. We recall that Laguerre's rule of signs states that the number of nonnegative zeros of g (counting multiplicities) does not exceed the number of sign changes in the sequence $a_1, \cdots, a_n$, which is at most $n - 1$. Moreover, this result is sharp, see for example [Steinig, 1986].

In our experiments in Section 3.5 we have observed between two to five local maxima of q with one-parameter Gaussian kernels. This fact is confirmed by Laguerre's rule of signs. Indeed, in our simulations the inputs x_i clustered well in two groups, that is, $d(x_i, x_j)$ is small when $y_i = y_j$ and larger when $y_i \neq y_j$. Moreover, each c_i usually has the same sign as y_i (this would always be the case for support vector machines, see, for example, [Shawe-Taylor and Cristianini, 2004]). Hence, when we order the $d(x_i, x_j)$ in a non-decreasing fashion the corresponding ordering of the coordinates of the vector $(c_i c_j : i, j \in \mathbb{N}_m)$ exhibits only a few sign changes.

3.4.3 A DC-Programming Approach

As we have already noted, the objective function $R(c, G(\cdot))$ is *not* convex, in general. This makes Step 2 of Algorithm 1 a challenging task. In fact, for a wide class of basic kernels, the function $R(c, G(\cdot))$ belongs to the class of *DC functions*. There are available iterative algorithms for optimizing such functions, although their theoretical complexity is not polynomial. We first review a few necessary definitions and results from the theory of DC functions, as presented in [Horst and Thoai, 1999].

Let Ω be a closed convex subset of $\mathbb{R}^D$. A function $f : \Omega \to \mathbb{R}$ is called DC on Ω if there exist two *convex* functions g and h such that

$$f(\omega) = g(\omega) - h(\omega) \qquad \forall \omega \in \Omega.$$

A remarkable result by [Hartman, 1959] states that *locally DC functions* are DC. It also implies that every twice continuously differentiable function on Ω is DC and every continuous function on Ω is the limit of a sequence of DC functions that converges uniformly on Ω. Moreover, the class of DC functions is linear and closed under multiplication and the *finite* min/max operations. Optimization problems of the type

$$\inf\{f(\omega) : \omega \in \Omega, f_i(\omega) \leq 0,\ i \in \mathbb{N}_n\}, \tag{3.4.2}$$

where f and f_i, $i \in \mathbb{N}_n$, are DC, are called *DC programs*.

We now derive a DC-programming formulation for the problem of maximizing function (3.4.1). To this end, we note that, for every $c \in \mathbb{R}^m$, the function $R(c, G(\cdot))$ is the limit of DC functions since, for every $x, z \in \mathcal{X}$, we have assumed continuity of $G(\cdot)(x, z)$. In addition, if $G(\cdot)(x, t)$ is twice continuously differentiable, maximizing (3.4.1) is a DC program.

Therefore, for most interesting continuous parameterizations, $G_{\mathbf{x}}(\omega)$ and hence $f(\omega)$ in (3.4.1) are DC functions. If, furthermore, the DC decomposition of $G(\cdot)(x, z)$ is available, we obtain an optimization problem of the form

$$\hat{\varphi} = \min\{f(\omega) = g(\omega) - h(\omega) :\ \omega \in \Omega\}, \tag{3.4.3}$$

where g, h are convex.

In particular, in the case of Gaussian kernels as in (3.2.23), $D = d(d+1)/2$, ω consists of the lower triangular elements of Σ^{-1} and the DC decomposition is given by

$$f(\omega) = -\sum_{\{i,j: c_i c_j > 0\}} c_i c_j e^{-\langle b_{ij}, \omega \rangle} \quad - \sum_{\{i,j: c_i c_j < 0\}} c_i c_j e^{-\langle b_{ij}, \omega \rangle},$$

where the indices $i, j \in \mathbb{N}_m$ and, for every i, j, $b_{ij} := ((2-\delta(k,\ell))(x_{ik}-x_{jk})\,(x_{i\ell}-x_{j\ell}) : d \geq k \geq \ell \geq 1)$. Here, g is the second term in the right hand side of the above equation and h is minus the first term.

A necessary and sufficient condition for $\hat{\omega}$ to solve problem (3.4.3) is that

$$\min\{-h(\omega)+t : \omega \in \Omega, t \in \mathbb{R}, g(\omega)-t \leq g(\hat{\omega})-h(\hat{\omega})\}$$

equals zero, see, for example, [Horst and Thoai, 1999, Proposition 4.4]. This observation has motivated the cutting plane algorithm proposed by Horst and Thoai, a variant of which we have implemented [Argyriou et al., 2006a]. The details appear in Algorithm 2. The algorithm works by constructing outer polytopes $P^{k+1} \subseteq P^k$ which contain the optimal solution $(\hat{\omega}, \hat{t} = h(\hat{\omega}))$. Subsequent polytopes are defined by cutting out the current vertex (ω^k, t^k) while keeping the solution inside. In other words, a hyperplane $\ell^k : \Omega \times \mathbb{R} \to \mathbb{R}$ is constructed so that

$$\begin{aligned} &\ell^k(\omega^k, t^k) > 0, \\ &\ell^k(\omega, t) \leq 0, \qquad \text{for all } (\omega, t) \in \mathbb{R}^{D+1} \text{ such that } \omega \in \Omega \text{ and } g(\omega) - t \leq \varphi^{k+1}. \end{aligned}$$

One can show that the sequence of function values converges to the optimal one from above, that is, $\hat{\varphi} \leq \varphi^{k+1} \leq \varphi^k$ – see [Horst and Thoai, 1999] for a proof and a more detailed explanation of the algorithm. Although the algorithm is guaranteed to converge in finite time, the worst case convergence rate is exponential. However, there are practical implementations of this and other DC programming algorithms which can tackle several hundreds or thousands of variables.

Finally, we should mention that the algorithm of [Horst and Thoai, 1999] is just one possible algorithm for DC problems. There have been several approaches for such types of problems, such as those in [An and Tao, 1998, Mangasarian, 1997]. Also, see [Yuille and Rangarajan, 2003] for another method, called concave-convex, which has been applied to several machine learning problems by the authors.

3.5 Experimental Validation

3.5.1 Experiments with the Greedy Algorithm

We have tested Algorithm 1 on eight handwritten digit recognition tasks of varying difficulty from the MNIST data set[2]. The data are 28×28 images with pixel values ranging between 0 and 255. We used Gaussian kernels as the basic kernels, that is, $G(\sigma)(x,z) = \exp(-\|x-z\|^2/\sigma^2)$, $\sigma \in [\sigma_1, \sigma_2]$. In all the experiments, the test error rates were measured over 1000 points from the MNIST test set.

The continuous and finite algorithms were trained using the square loss and compared to an SVM[3]. In all experiments, the training set consisted of 500 points and the test set of 1000 points. For the finite case, we chose ten Gaussian kernels with σ's equally spaced in an interval $[\sigma_1, \sigma_2]$. For both versions of our algorithm, the starting value of the kernel was the average of these ten kernels. The performance of the SVM was obtained as the best among the results for the above ten kernels. This strategy slightly favors the SVM but compensates for the fact that the loss functions are different. The

[2]Available at: `http://yann.lecun.com/exdb/mnist/index.html`

[3]Trained using SVM-light, see: `http://www.cs.cornell.edu/People/tj/svm_light`

Algorithm 2 Cutting plane algorithm for DC programming.

```
Inputs: A point $\omega^0$ in the interior of $\Omega$; a simplex $S^0 \supseteq \Omega$ with vertex set $V(S^0)$; a convex function
$\alpha : \mathbb{R}^D \to \mathbb{R}$ such that $\Omega = \{\omega \in \mathbb{R}^D : \alpha(\omega) \leq 0\}$.
  Initialization:
    Set $\varphi^0 = g(\omega^0) - h(\omega^0)$.
    Compute a subgradient $s \in \partial g(\omega^0)$.
    Choose $\bar{t} > \max\{g(\omega) : \omega \in V(S^0)\} - \bar{\varphi}$, where
        $\bar{\varphi} = \min\{g(\omega) : \omega \in V(S^0)\} - \max\{h(\omega) : \omega \in V(S^0)\}$.
    Construct a polytope $P^0$ from $S^0, \bar{t}$ and $\omega^0$.
    Set $k = 0$.
  while $\|\varphi^{k+1} - \varphi^k\| < tol$ do
    Compute an optimal solution $(\omega^*, t^*)$ of the problem $\min\{-h(\omega) + t : (\omega, t) \in V(P^k)\}$.
    if $-h(\omega^*) + t^* = 0$ then
      Stop. $\omega^k$ is an optimal solution to (3.4.3) with optimal value $\varphi^k$.
    else $\{-h(\omega^*) + t^* < 0\}$
      Case 1: $\omega^* \in \Omega$
        Compute $s^k \in \partial g(\omega^*)$.
        Case 1a: $g(\omega^*) - h(\omega^*) < \varphi^k$
          Set $\omega^{k+1} = \omega^*$.
        Case 1b: $g(\omega^*) - h(\omega^*) \geq \varphi^k$
          Set $\omega^{k+1} = \omega^k$.
      Case 2: $\omega^* \notin \Omega$
        Compute $s^k \in \partial \beta^k(\omega^*, t^*)$, where $\beta^k(\omega, t) := \max\{\alpha(\omega), g(\omega) - t - \varphi^k\}$.
        Compute the zero $(\zeta^*, \theta^*)$ of $\beta^k(x, t)$ on the line segment joining $(\omega^*, t^*)$ and $(\omega^0, \bar{t})$.
        Case 2a: $g(\zeta^*) - h(\zeta^*) < \varphi^k$
          Set $\omega^{k+1} = \zeta^*$.
        Case 2b: $g(\zeta^*) - h(\zeta^*) \geq \varphi^k$
          Set $\omega^{k+1} = \omega^k$.
      Construct the cutting plane (affine function)
        $\ell^k(x,t) = \begin{cases} \langle \omega - \omega^*, s^k \rangle + g(\omega^*) - \varphi^{k+1} - t & \text{if } \omega^* \in \Omega \\ \langle (\omega, t) - (\zeta^*, \theta^*), s^k \rangle + \beta^k(\zeta^*, \theta^*) & \text{if } \omega^* \notin \Omega. \end{cases}$
      Set $P^{k+1} = P^k \cap \{(\omega, t) : \ell^k(\omega, t) \leq 0\}$.
      Set $k = k + 1$.
    end if
  end while
```

regularization parameter was selected with 5-fold cross validation in all cases. The tolerance parameters of our algorithm were set to be equal to 10^{-3}. Finally, the optimization technique used for step 2 of the

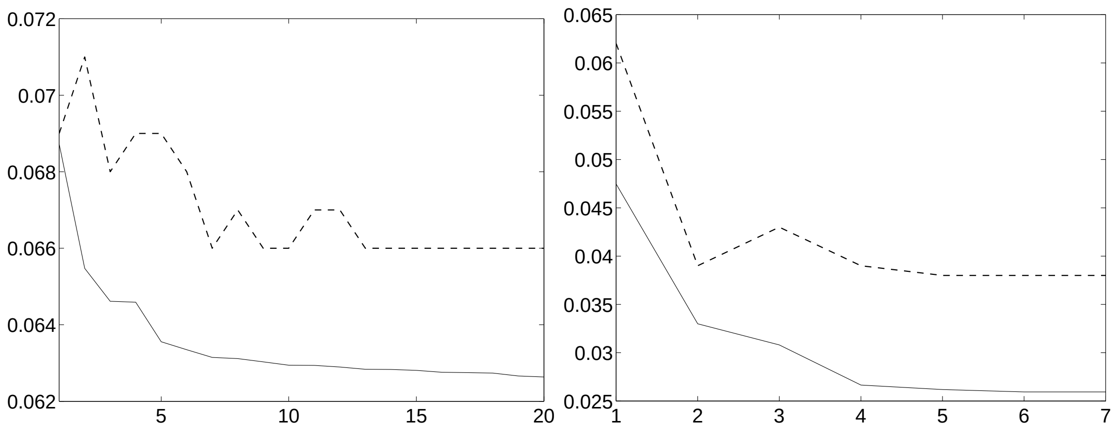


Figure 3.1: Functional E_γ (*solid line*) and misclassification error (*dotted line*) after the first iteration of Algorithm 1 for even vs. odd (*left*) and 3 vs. 8 (*right*).

continuous algorithm consisted of local searches covering the whole range of σ.

Table 3.1 shows the results obtained. The range of σ is $[75, 25000]$ in columns 2–4, $[100, 10000]$ in columns 5–7 and $[500, 5000]$ in columns 8–10. Note that, in most cases, the continuous algorithm finds a better combination of kernels than the finite version. In general, the continuous algorithm performs better than the SVM, whereas most of the time the finite algorithm is worse than the SVM. Moreover, the results indicate that the continuous algorithm *is not affected by the range of* σ, unlike the other two methods.

Typically, the continuous algorithm requires less than 20 iterations to terminate whereas the finite algorithm may require as much as 100 iterations. Figure 3.1 depicts the convergence behavior of the continuous algorithm on two different tasks. In both cases $\sigma \in [100, 10000]$. The actual values of E_γ are six orders of magnitude smaller, but they were rescaled to fit the plot. Note that, in agreement with inequality (3.3.2), E_γ decreases and eventually converges. The misclassification error also converges to a lower value, indicating that E_γ provides a good learning criterion.

3.5.2 Experiments with the DC Algorithm

We also performed a series of experiments on the MNIST data set using the greedy algorithm (Algorithm 1) combined with the DC method (Algorithm 2). In Table 3.2, we present the results obtained using the DC algorithm alongside the previous greedy algorithm, the greedy algorithm for a finite number of basic kernels and SVM-light. It is clear from this table that the DC approach is as good as the greedy algorithm for selecting a single variance.

Next, we performed experiments with multiple parameters. In such cases, the standard exhaustive search methods suffer from the curse of dimensionality, whereas DC approaches are able to tackle optimization problems in several dimensions. In Table 3.3, we show the performance of the DC algorithm simultaneously learning two isotropic variances, one corresponding to the left and one to the right part of the image. The performance with two variances significantly improves on that with one variance and the algorithm remains robust over different ranges of σ_1 and σ_2, which is evidence that the optimization

Table 3.1: Misclassification error percentage for the continuous and finite versions of the algorithm and the SVM on different handwritten digit recognition tasks.

Task	Method								
	Cont.	Finite	SVM	Cont.	Finite	SVM	Cont.	Finite	SVM
	$\sigma \in [75, 25000]$			$\sigma \in [100, 10000]$			$\sigma \in [500, 5000]$		
odd vs. even	6.6	18.0	11.8	6.6	10.9	8.6	6.5	6.7	6.9
3 vs. 8	3.8	6.9	6.0	3.8	4.9	5.1	3.8	3.7	3.8
4 vs. 7	2.5	4.2	2.8	2.5	2.7	2.6	2.5	2.6	2.3
1 vs. 7	1.8	3.9	1.8	1.8	1.8	1.8	1.8	1.9	1.8
2 vs. 3	1.6	3.9	3.1	1.6	2.8	2.3	1.6	1.7	1.6
0 vs. 6	1.6	2.2	1.7	1.6	1.7	1.5	1.6	1.6	1.5
2 vs. 9	1.5	3.2	1.9	1.5	1.9	1.8	1.5	1.4	1.4
0 vs. 9	0.9	1.2	1.1	0.9	1.0	1.0	0.9	0.9	1.0

subproblem was successfully solved. In the same table, we present the performance of the finite kernels method mentioned above using grids of 5×5 and 10×10 kernels with equally spaced σ's. This method succeeds only in the smallest range of σ's and with the 10×10 grid. But in the absence of information about σ, even with the finer grid the finite kernels method is not competitive. We also performed experiments with four isotropic variances (corresponding to the four quadrants of the image), which did not further improve the results.

As regards the computational cost, our method using DC programming compares favorably to the finite method. We implemented both of them in Matlab and performed the experiments on a 1GHz dual-processor machine running Linux. For the local optimization of $-h(x) + t$ in Algorithm 2, we used Matlab's fmincon() routine. We observed that the finite method is much worse in time cost than the DC algorithm (about 1 hour vs. 5 minutes with 2 parameters). The main computational cost of our algorithm is incurred by the aforementioned local optimization, whereas the finite method scales polynomially with the grid size. Still, the running time of our algorithm deteriorates fast with the number of parameters. For example, it takes 1-2 minutes for learning one parameter, about 5 minutes for 2 parameters and about 1 hour for 4 parameters. We speculate that faster local search exploiting linear programming and the special nature of the function can lead to further improvements in efficiency. With respect to memory requirements, our algorithm is clearly more efficient because it only needs to store the linear constraints, whereas the grid method requires all the kernels to be in memory.

We also observed that the greedy algorithm usually required less than 20 iterations to terminate,

Table 3.2: Misclassification error percentage for learning one kernel parameter on the MNIST tasks.

Task	Method											
	greedy		finite	SVM	greedy		finite	SVM	greedy		finite	SVM
	DC	local			DC	local			DC	local		
	$\sigma \in [75, 25000]$				$\sigma \in [100, 10000]$				$\sigma \in [500, 5000]$			
odd vs. even	6.5	6.6	18.0	11.8	6.5	6.6	10.9	8.6	6.5	6.5	6.7	6.9
3 vs. 8	3.7	3.8	6.9	6.0	3.9	3.8	4.9	5.1	3.6	3.8	3.7	3.8
4 vs. 7	2.7	2.5	4.2	2.8	2.4	2.5	2.7	2.6	2.3	2.5	2.6	2.3

Table 3.3: Misclassification error percentage of DC algorithm vs. finite grid for 2 parameters on the MNIST tasks.

Task	Number of kernel parameters								
	greedy-DC	5×5	10×10	greedy-DC	5×5	10×10	greedy-DC	5×5	10×10
	$\sigma \in [75, 25000]$			$\sigma \in [100, 10000]$			$\sigma \in [500, 5000]$		
odd vs. even	5.8	15.8	11.2	5.8	10.1	6.2	5.8	6.8	5.8
3 vs. 8	2.7	6.5	5.1	2.5	4.6	2.5	2.6	3.5	2.5
4 vs. 7	1.8	3.9	2.9	1.7	2.7	2.0	1.8	2.0	1.8

which is evidence in favor of the DC approach. The cutting plane method usually required less than 100 iterations to converge. Thus, the learned convex combination has a small (usually less than 10) number of kernels.

Finally, in Figure 3.2 we present the learned kernel coefficients for two isotropic variances (left and right image) in the range $[100, 10000]$. These indicate that for the odd vs. even task it is better to combine several complementary kernels focused on different parts of the images than use a single Gaussian kernel. However, for the 3-8, 4-7 tasks there is a clear winner among the kernels. This conforms with the intuition that odd vs. even is a more complex task than the binary ones. Moreover, augmenting the parameter class for odd vs. even results in a more complex (and more effective) representation for the solution. In order to gain more insight into the nature of this solution, we have plotted the corresponding variances for odd vs. even in Figure 3.3. It is clear that the learned kernels are either focused *exclusively* on each half of the images or operate on the image as a whole.

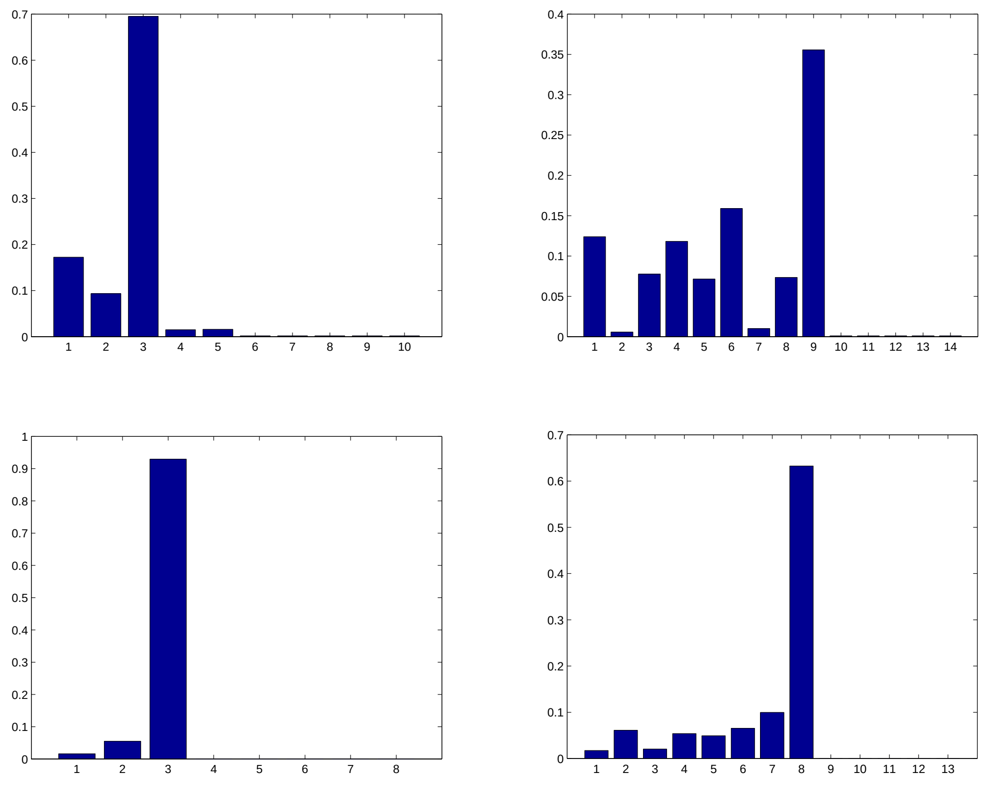


Figure 3.2: Learned kernel coefficients for different classification tasks and kernel parameterizations. Top plots are for odd vs. even (the dimensionality is 1 on the left and 2 on the right). Bottom plots are for the 3-8 task (left) and the 4-7 task (right), with dimensionality 2.

3.6 Interpretation of Learning the Kernel in the Space of Features

An alternate point of view for learning the kernel focuses on the feature space representation of kernels. The idea here is to reformulate the variational problem (3.2.5), which has been the focus of our discussion, in the space associated with the features of basic kernels. This issue is investigated in generality in [Micchelli and Pontil, 2007]. Here, we wish to highlight some of its main observations. To keep the discussion accessible we restrict ourselves to the case that the parameter set Ω is finite, that is, $\Omega = \mathbb{N}_n$ and each of the basic kernels is determined by a finite number of features. Hence, for each $\ell \in \mathbb{N}_n, x, z \in \mathcal{X}$ we write $G_\ell(x,z) = \langle \Phi_\ell(x), \Phi_\ell(z) \rangle$, where $\Phi_\ell(x) \in \mathbb{R}^s$ and s is the number of features. With this representation of the basic kernels, we follow [Micchelli and Pontil, 2007] and consider the variational problem

$$Q\left(\sum_{\ell \in \mathbb{N}_n} w_\ell^\top \Phi_\ell(\mathbf{x})\right) + \gamma \left(\sum_{\ell \in \mathbb{N}_n} \|w_\ell\|\right)^2, \tag{3.6.1}$$

where $\Phi_\ell(\mathbf{x}) = (\Phi_\ell(x_1), \ldots, \Phi_\ell(x_m))$. The minimizer $(\hat{w}_\ell : \ell \in \mathbb{N}_n)$ of this variational problem provides an optimal kernel based on the features defined above. In particular, the optimal kernel is given by

$$\hat{K} = \sum_{\ell \in \mathbb{N}_n} \frac{\|w_\ell\|}{\sum_{r \in \mathbb{N}_n} \|w_r\|} G_\ell. \tag{3.6.2}$$

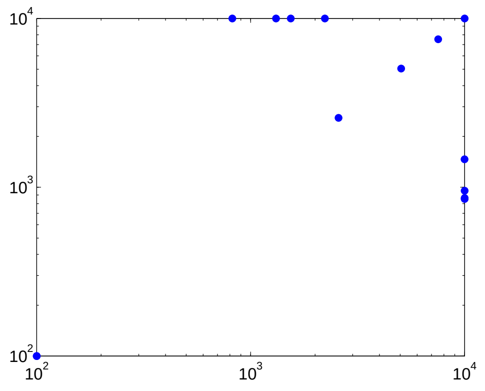


Figure 3.3: Learned $[\sigma_1, \sigma_2]$ parameters of the kernels in the optimal convex combination, for the odd vs. even task. The parameter range was [100,10000].

A detailed explanation for this fact and its extensions to the continuous case can be found in [Micchelli and Pontil, 2007, Thm. 2.1, Cor. 3.1, Eq. (3.4)]. As pointed out in that paper, when there is one feature, namely $s = 1$, equation (3.6.1) reduces to the L_1 regularization problem. There has been renewed interest in this problem because a minimizing solution will often have few nonzero coefficients, in other words will be sparse. Furthermore, the formula (3.6.2) will appear again in Section 4.3.3, in the context of learning features for multiple tasks.

3.7 Bounds for Learning the Kernel

In the theoretical study of the optimization problem of learning with convex sets of kernels, we have postponed the issue of the generalization properties of this formulation. First, the possibility of overfitting the data in problem (3.2.5) should be ruled out. This is clearly the case when the set of kernels $\mathcal{K}$ is so rich that the term $\langle \beta, K_{\mathbf{x}}^{-1} \beta \rangle$ can become zero for any $\beta \in \mathbb{R}^m$, where $\beta = K_{\mathbf{x}} \alpha$ in (3.2.4). Hence the requirement for a constraint that bounds the kernel matrix. The question that arises is what this constraint may be. The answer given in [Micchelli et al., 2005] is that the kernels in class $\mathcal{K}$ should be uniformly bounded, that is,

$$\sup\{K(x,x) : x \in \mathcal{X}, K \in \mathcal{K}\} < +\infty \,,$$

and continuous. Then, for a broad class of loss functions, the minimum of the regularization functional, $E_\gamma(\mathcal{K})$, is bounded from below by a positive constant that depends on the value of the above supremum.

A second question is how the difference between the empirical error on the data $\{\mathbf{x}, y\}$ and the optimal expected error over $\mathcal{X}$ is bounded. An upper bound on this difference has been shown in [Srebro and Ben-David, 2006, Thm. 2] using covering numbers. This bound involves the *pseudodimension* d_ϕ of the set $\mathcal{K}$, which is a measure of the set's complexity. It appears as $\sqrt{\frac{O\left(d_\phi + \frac{1}{\eta^2}\right)}{m}}$, where η is the margin of the classifier. In the case of a finite number n of basic kernels, the pseudodimension d_ϕ is at most n. Also, in the case of Gaussian basic kernels, it is bounded in terms of the size of the covariance as $d_\phi \leq \frac{d(d+1)}{2}$. Finally, with isotropic Gaussian kernels, $d_\phi \leq d$ and with rank-k

covariance, $d_\phi \leq kd\log_2(8ekd)$. Thus, as intuitively expected, kernel learning becomes harder the richer the covariance parameter space.

3.8 Convex Combinations of Graph Kernels

In this section, we show how optimization over convex sets of kernels can be applied to a special setting. This is the case of *semi-supervised learning*, which addresses classification problems with few labeled data (see Section 2.2.3). The semi-supervised learning methods we have mentioned exploit the structure of a prescribed graph for separating the data.

The construction of this graph usually consists of two stages, first selection of a distance function and then application of it to determine the graph's edges (or weights). For example, below we shall consider distances between images based on the Euclidean distance, Euclidean distance combined with image transformations, and the related tangent distance [Hastie and Simard, 1998]; we shall determine the edge set of the graph with k-nearest neighbors. Another common choice is to weight edges by a decreasing function of the distance $d(v_i, v_j)$ between vertices v_i, v_j such as $e^{-\beta d(v_i,v_j)^2}$.

Although a surplus of unlabeled data may improve the quality of the empirical approximation of the manifold (via the graph) leading to improved performances, practical experience with these methods indicates that their performance significantly depends on how the graph is constructed. Hence, it is necessary to address the model selection problem, that is, selection of the distance function and the parameters k or β used in the graph building process described above. A diversity of methods have been proposed for graph construction. In contrast, the method we propose *combines* a number of graphs via their Laplacians and the corresponding kernels. For a given data set each combination of distance functions and edge set specifications from the distance will lead to a specific graph. Each of these graphs may then be associated with a kernel and we will learn the optimal convex combination of these kernels. This approach is further motivated by the fact that, unlike in the supervised case, cross validation is not an option here. Indeed, all labeled vertices must be conserved for training since their number is small.

Figure 3.5 in Section 3.8.2 illustrates our algorithm on a simple example. There, three different distances for 400 images of the digits 'six' and 'nine' are depicted, namely, the Euclidean distance, a distance invariant under small centered image rotations from $[-10°, 10°]$ and a distance invariant under rotations from $[-180°, 180°]$. Clearly, the last distance is problematic as sixes become similar to nines. The performance of the graph regularization algorithm with these distances is reported below each plot; as expected, this performance is much lower in the case that the third distance is used. Moreover, the algorithm which we shall describe in Section 3.8.1 performs optimally and combines only the two "good" graphs.

3.8.1 Combining Graph Laplacians

We now describe our framework for learning with multiple graph Laplacians. We assume that we are given n graphs $\mathscr{G}_\ell$, $\ell \in \mathbb{N}_n$, all having m vertices, with corresponding Laplacians $\mathbf{L}_\ell$, kernel matrices $\mathbf{B}_\ell = (\mathbf{L}_\ell)^+$, Hilbert spaces $\mathcal{H}_\ell := \mathcal{H}(\mathscr{G}_\ell)$ and norms $\|\mathbf{v}\|_\ell^2 := \mathbf{v}^\top \mathbf{L}_\ell \mathbf{v}$, $\mathbf{v} \in \mathcal{H}_\ell$. We propose to learn an optimal convex combination of graph kernels, that is, we solve the optimization problem

$$\min\left\{Q(\bar{\mathbf{v}}) + \gamma\|\mathbf{v}\|^2_{\mathbf{K}(\lambda)} : \lambda \in \Lambda,\ \mathbf{v} \in \mathcal{H}_{\mathbf{K}(\lambda)}\right\}. \tag{3.8.1}$$

Here, we have defined $\bar{\mathbf{v}}$ to be the labeled part of $\mathbf{v}$, the set $\Lambda := \{\lambda \in [0,1]^n : \sum_{\ell\in\mathbb{N}_n} \lambda_\ell = 1\}$ and, for each $\lambda \in \Lambda$, the kernel matrix $\mathbf{K}(\lambda) := \sum_{\ell\in\mathbb{N}_n} \lambda_\ell \mathbf{B}_\ell$ and the RKHS $\mathcal{H}_{\mathbf{K}(\lambda)}$ to be the range of $\mathbf{K}(\lambda)$ with inner product $\langle\cdot,\cdot\rangle_{\mathbf{K}(\lambda)}$, norm $\|\cdot\|_{\mathbf{K}(\lambda)}$ induced by the corresponding Laplacian.

Solving problem (3.8.1) for fixed λ in the cases of square loss regularization [Belkin and Niyogi, 2004] and minimal norm interpolation [Zhu et al., 2003] requires solving a linear system of m and $m-\ell$ equations respectively. Instead, we choose to use the representer theorem to express $\mathbf{v}$ as

$$\mathbf{v} = \left(\sum_{j\in\mathbb{N}_\ell} L^+_{ij}\alpha_j : i \in \mathbb{N}_m\right).$$

This approach is advantageous if $\mathbf{L}^+$ can be computed off-line because, typically, $\ell \ll m$. A further advantage of this approach is that multiple problems may be solved with the same Laplacian kernel. The coefficients α_j are obtained by solving problem (3.8.1) with $\mathbf{K} = (L^+_{ij})_{i,j\in\mathbb{N}_\ell}$. For example, for square loss regularization the computation of the parameter vector $\alpha = (\alpha_j : j \in \mathbb{N}_\ell)$ involves solving a linear system of ℓ equations, namely

$$(\mathbf{K} + \gamma I)\alpha = y\,. \tag{3.8.2}$$

Problem (3.8.1) is clearly a special case of (3.2.5). Using (3.2.8), we can rewrite problem (3.8.1) as

$$\max\left\{\min\left\{\frac{1}{4\gamma}c^\top\mathbf{K}(\lambda)c + Q^*(c) : c \in \mathbb{R}^\ell\right\} : \lambda \in \Lambda\right\}. \tag{3.8.3}$$

As we have already discussed, this problem is simpler to solve than the original problem (3.8.1) since its objective function is linear in λ. Thus, Algorithm 1 of Section 3.3 can be used for computing a saddle point $(\hat{c},\hat{\lambda}) \in \mathbb{R}^\ell \times \Lambda$.

When solving problem (3.8.1) it is important to require that the kernel matrices $\mathbf{B}_\ell$ satisfy a normalization condition such as that they all have the same trace or the same Frobenius norm (see Section 3.7). We also note that the above analysis naturally extends to the case that $\mathbf{L}$ is replaced by any positive semidefinite matrix. In particular, in our experiments below we will use the normalized Laplacian matrix given by $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is the diagonal matrix of the degrees of the vertices.

3.8.2 Experiments

We have performed experiments with our algorithm for combining graphs on optical character recognition. In these, we observed the following. First, the optimal convex combination of graph kernel matrices computed by our algorithm is competitive with the best basic kernel matrices. Second, by observing the 'weights' of the convex combination we can distinguish the strong from the weak candidate graph kernels and hence the weight matrices as well. We proceed by discussing the details of the experimental design interleaved with our results.

We used the USPS dataset[4] of 16×16 images of handwritten digits with pixel values ranging between -1 and 1. We present the results for 5 pairwise classification tasks of varying difficulty and for odd

[4] Available at: *http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html*

	Euclidean (10 kernels)			Transformation (10 kernels)			Tangent distance (10 kernels)			All (30 kernels)		
Labels %	1%	2%	3%	1%	2%	3%	1%	2%	3%	1%	2%	3%
Task												
1 vs. 7	1.55	1.53	1.50	1.45	1.45	1.38	1.01	1.00	1.00	1.28	1.24	1.20
	0.08	0.05	0.15	0.10	0.11	0.12	0.00	0.09	0.11	0.28	0.27	0.22
2 vs. 3	3.08	3.34	3.38	0.80	0.85	0.82	0.73	0.19	0.03	0.79	0.25	0.10
	0.85	1.21	1.29	0.40	0.38	0.32	0.93	0.51	0.09	0.93	0.61	0.21
2 vs. 7	4.46	4.04	3.56	3.27	2.92	2.96	2.95	2.30	2.14	3.51	2.54	2.41
	1.17	1.21	0.82	1.16	1.26	1.08	1.79	0.76	0.53	1.92	0.97	0.89
3 vs. 8	7.33	7.30	7.03	6.98	6.87	6.50	4.43	4.22	3.96	4.80	4.32	4.20
	1.67	1.49	1.43	1.57	1.77	1.78	1.21	1.36	1.25	1.57	1.46	1.53
4 vs. 7	2.90	2.64	2.25	1.81	1.82	1.69	0.88	0.90	0.90	1.04	1.14	1.13
	0.77	0.78	0.77	0.26	0.42	0.45	0.17	0.20	0.20	0.37	0.42	0.39
Labels	10	20	30	10	20	30	10	20	30	10	20	30
odd vs. even	18.6	15.5	13.4	15.7	11.7	8.52	14.66	10.50	8.38	17.07	10.98	8.74
	3.98	2.40	2.67	4.40	3.14	1.32	4.37	2.30	1.90	4.38	2.61	2.39

Table 3.4: Misclassification error percentage (*top*) and standard deviation (*bottom*) for the best convex combination of kernels on different handwritten digit recognition tasks, using different distances. See text for description.

vs. even digit classification. For pairwise classification, the training set consisted of the first 200 images for each digit in the USPS training set and the number of labeled points was chosen to be 4, 8 or 12 (with equal numbers for each digit). For odd vs. even digit classification, the training set consisted of the first 80 images per digit in the USPS training set and the number of labeled points was 10, 20 or 30, with equal numbers for each digit. Performance was averaged over 30 random selections, each with the same number of labeled points.

In each experiment, we constructed $n = 30$ graphs $\mathscr{G}_\ell$ ($\ell \in \mathbb{N}_n$) by combining k-nearest neighbors ($k \in \mathbb{N}_{10}$) with three different distances (described below). Then, n corresponding Laplacians were computed together with their associated kernels. We chose as the loss function Q the square loss. Since kernels obtained from different types of graphs can vary widely, it was necessary to renormalize them. Hence, we chose to normalize each kernel during the training process by the Frobenius norm of its submatrix corresponding to the labeled data. We also observed that similar results were obtained when normalizing with the trace of this submatrix. The regularization parameter was set to 10^{-5} in all algorithms. As the starting kernel in Algorithm 1 we always used the average of the n kernels. We observed that the number of iterations needed was usually about 30.

Table 3.4 shows the results obtained using three distances combined with k-NN ($k \in \mathbb{N}_{10}$). The first distance is the *Euclidean* distance between images. The second method is *transformation*, where the distance between two images is given by the smallest Euclidean distance between any pair of transformed images as determined by applying a number of affine transformations and a thickness transformation[5], see [Hastie and Simard, 1998] for more information. The third distance is *tangent distance*, as described in [Hastie and Simard, 1998], which is a first-order approximation to the above transformations. For the first three columns in the table the Euclidean distance was used, for columns 4–6 the image transformation distance was used, for columns 7–9 the tangent distance was used. Finally, in the last three columns all three methods were jointly compared.

As the results indicate, when combining different types of kernels, the algorithm tends to select the most effective ones (in this case the tangent distance kernels and to a lesser degree the transformation distance kernels, which did not work very well because of the Matlab optimization routine we used). We also observed that within each of the methods the performance of the convex combination is comparable to that of the best kernels. Figure 3.4 reports the weight of each individual kernel learned by our algorithm when 2% labels are used in the pairwise tasks and 20 labels are used for odd vs. even. With the exception of the easy 1 vs. 7 task, the large weights are associated with the graphs/kernels built with the tangent distance.

The effectiveness of our algorithm in selecting the good graphs/kernels is better demonstrated in Figure 3.5, where the Euclidean and the transformation kernels are combined with a "low-quality" kernel. This "low-quality" kernel is induced by considering distances invariant over rotation in the range $[-180°, 180°]$, so that the image of a 6 can easily have a small distance from an image of a 9. That is, if $\mathbf{x}$ and $\mathbf{t}$ are two images and $T_\theta(\mathbf{x})$ is the image obtained by rotating $\mathbf{x}$ by θ degrees, we set

$$d(\mathbf{x}, \mathbf{t}) = \min\{\|T_\theta(\mathbf{x}) - T_{\theta'}(\mathbf{t})\| : \theta, \theta' \in [-180°, 180°]\}.$$

The figure shows the distance matrix on the set of labeled and unlabeled data for the Euclidean, transformation and "low-quality distance" respectively. The best error among 15 different values of k within each distance, the error of the learned convex combination and the total learned weights for each distance are shown below each plot. It is clear that the solution of the algorithm is dominated by the good kernels and is not influenced by the ones with low performance. As a result, the error of the convex combination is comparable to that of the Euclidean and transformation distances. The final experiment (see Figure 3.6) demonstrates that availability of unlabeled data improves the performance of our method.

[5]This distance was approximated using Matlab's constrained minimization function.

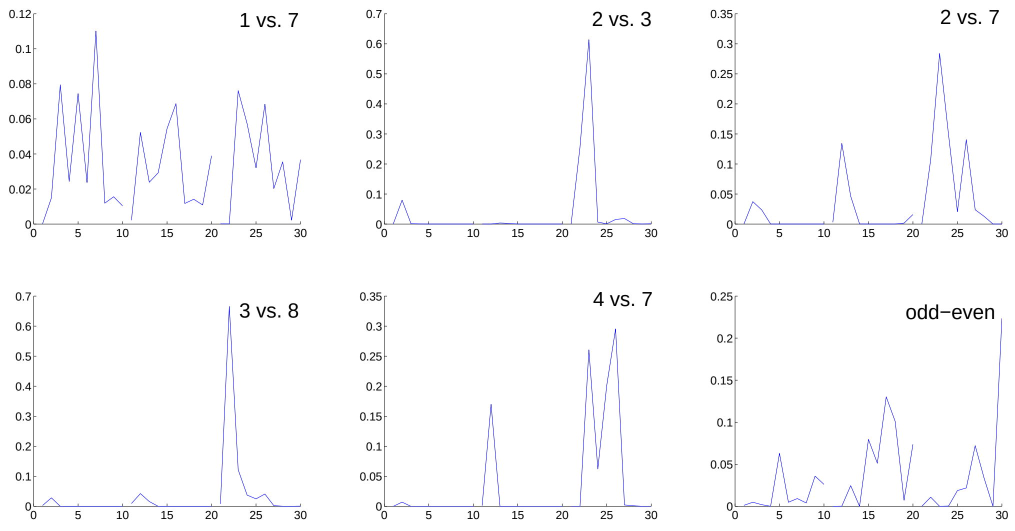
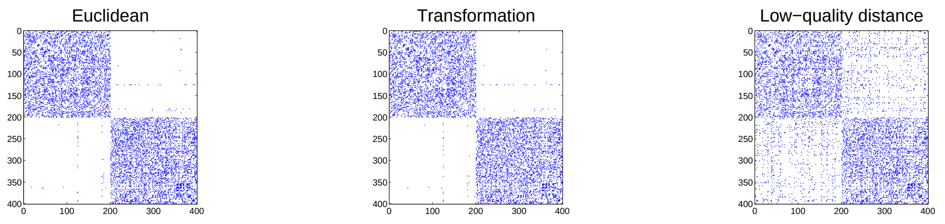


Figure 3.4: Kernel weights for Euclidean (first 10), Transformation (middle 10) and Tangent (last 10).



error = 0.24% error = 0.24% error = 17.47%

$\sum_{i=1}^{15} \lambda_i = 0.553$ $\sum_{i=16}^{30} \lambda_i = 0.406$ $\sum_{i=31}^{45} \lambda_i = 0.041$

convex combination error = 0.26%

Figure 3.5: Similarity matrices and corresponding learned coefficients of the convex combination for the 6 vs. 9 task.

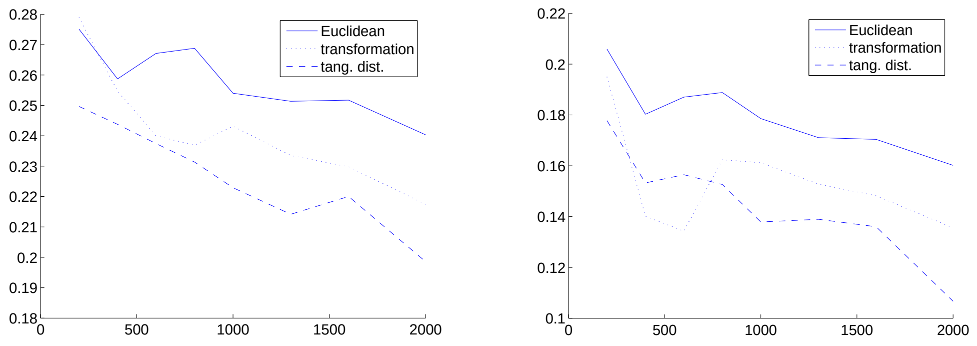


Figure 3.6: Misclassification error vs. number of training points for odd vs. even classification. The number of labeled points is 10 on the left and 20 on the right.

Chapter 4

Multi-Task Feature Learning

This chapter addresses some fundamental issues on how to simultaneously learn multiple tasks. Multi–task learning has recently been recognized as an important objective in machine learning, one reason being that it is directly motivated by human intelligence. From an early age, infants learn how to recognize similarities among related tasks and subsequently apply any previously learned knowledge to new tasks. Similarly, our aim is to develop algorithms which exploit the similarities among given tasks. This contrasts with the traditional approach in machine learning, which treats different tasks in isolation and ignores any connections, structural or of any other kind.

We take a first step towards this goal by viewing the multi–task question as one of comparing feature representations. Given a set of tasks which are somehow related, is there a way to *learn* features which are *common to all the tasks*? An answer can be provided in the form of an optimization problem which extends standard L_2 regularization using a suitably defined *mixed norm*. Solving this and similar problems is the main object of this chapter.

Our approach can also be viewed as a generalization of previous work on feature selection and basis pursuit. We briefly review some of this work, as well as related multi–task learning work, and discuss the connections to our own work. We then show that our optimization problem can be reduced to a *convex* one. This implies that the optimal solution can be efficiently determined by a simple algorithm with an interesting interpretation. It alternately performs a supervised and an unsupervised step, where in the latter step we learn common-across-tasks representations and in the former step we learn task-specific functions using these representations. As a special case, the algorithm can also be used for *selecting* (not learning) features among multiple tasks.

Since in many cases the features of interest can be nonlinear or high-dimensional, we also develop an algorithm which only depends on knowledge of a kernel function. This algorithm is motivated by a representer theorem but makes further improvements in efficiency.

Finally, in the remainder of the chapter we discuss how multi–task feature learning relates to learning from different sources, especially in the context of learning convex combinations of kernels. Both problems can be viewed as generalizations of feature selection and in fact it can be shown that multi–task feature learning is a special case of learning the kernel.

4.1 Prior Work

The problem of learning *common data representations* across *multiple related tasks* appears in many research areas. For example, in computer vision or multimedia search, images (or video, audio, text, etc.) of different objects may share a common underlying representation [Heisele et al., 2002, Serre et al., 2005] – much like in human vision common features (e.g. wavelet like) are used to represent and recognize different objects. In this case, recognizing a particular object is considered to be a single task. Moreover, a number of recent works in computer vision have successfully exploited similarities in object recognition tasks to learn new related tasks, sometimes even from a single example [Bart and Ullman, 2005, Ferencz et al., 2005, Fink, 2005, Miller et al., 2000, Torralba et al., 2004].

In modeling users/consumers' preferences (conjoint analysis) [Evgeniou et al., 2007, Kim et al., 2004, Lenk et al., 1996], there are common products of a particular type (such as books, music, web-pages, consumer electronics etc.) with standard product attributes (such as size, color, price). Usually, there are features that are considered to be important by a number of people and often combine different attributes. That is, modeling an individual's preferences corresponds to a task. Also, in reinforcement learning, exploiting similarities between different tasks or environments has been recognized to enhance learning performance [Madden and Howley, 2004, Marthi et al., 2005].

The main insights about learning multiple tasks were set out in [Baxter, 2000] (and in [Baxter, 1997] from a Bayesian/information theoretic perspective). The distinction was made therein between *simultaneous learning* of a number T of tasks versus *learning by transfer* of knowledge from T tasks to a new one. The term *multi-task learning* usually refers to the former, whereas the latter is also known by the names *learning to learn* and *inductive transfer*.

Baxter views multi-task learning as a manifestation of *inductive bias*. A learning task can be thought of as a probability distribution on the input/output space $\mathcal{X} \times \mathcal{Y}$. For the purpose of learning multiple tasks, we may assume that they are obtained by a probability distribution which favors tasks related in some sense. On the algorithmic side, the learning algorithms for the T tasks are biased by the choice of a *common hypothesis space* (a space of functions from which to select a good solution for each task). Therefore, algorithms for multi-task learning can be considered to *learn the bias*, in other words, the hypothesis space is not fixed but belongs to a family $\mathbb{H}$ of hypothesis spaces.

Within this model, it was shown that the generalization error can be bounded in terms of the empirical error over a certain hypothesis space, if the number of tasks T and the number of examples per task m are sufficiently large. As the number of tasks T increases the required number of examples m decreases, which means that less data per task are needed to learn more tasks. In particular, the upper bound on m/T is better for multiple tasks than for learning a single task. Another result from [Baxter, 2000] is that it is possible to transfer bias to a new task after having learned a hypothesis space $\mathcal{H}$ based on T tasks. Indeed, the number of examples needed for the novel task depends on the capacity of $\mathcal{H}$ and not on that of the whole family $\mathbb{H}$. A necessary caveat is that $\mathcal{H}$ should be sufficiently related to the new task so that the empirical error over $\mathcal{H}$ is small. According to an empirical observation by [Caruana, 1997], pooling unrelated tasks may hurt performance – however we shall provide some experimental evidence to the contrary in Section 4.7.

An example of bias on $\mathcal{H}$ is to assume that *tasks share a common set of features*. Low-dimensional feature representations have long been assumed to underly many learning problems [Vapnik, 2000]. In the multi-task case, a common set of features corresponds to a hypothesis space, that is, it induces relations across the tasks. The goal is then to *learn the common features* instead of a priori assuming a given set. This intuition, that inductive transfer may take place through a shared representation, is also advanced by [Caruana, 1997]. Moreover, sample complexity is related to the complexity of the space of features [Baxter, 2000] or to data compressibility [Juba, 2006]. In particular, [Maurer, 2006a,b] showed that, in the case that these feature maps are bounded linear operators, the sample complexity depends on their Hilbert-Schmidt norm. In general, common features may not be shared equally across tasks but their relevance may vary depending on groupings or hierarchical structures of tasks – see, for example, [Bakker and Heskes, 2003, Torralba et al., 2004].

A number of multi-task algorithms that have been proposed use neural networks. As already mentioned, [Baxter, 2000, Caruana, 1997, Silver and Mercer, 1996] impose a small number of common features to be learned jointly for all the tasks. This is ensured through a hidden layer with few nodes and through a set of network weights "shared" by all the tasks.

Another popular approach involves hierarchical Bayesian models [Bakker and Heskes, 2003, Bonilla et al., 2007, Dominici et al., 1997, Kim et al., 2004, Mallick and Walker, 1997, Raina et al., 2006, Xue et al., 2007, Yu et al., 2005, Zhang et al., 2006]. In general, these models enforce task relatedness through a common prior probability distribution on the tasks' parameters. The prior is learned as part of the training process and different variations using Gaussian mixtures, Gaussian processes, Independent Component Analysis, Dirichlet process priors etc. have been applied. Some of this work has been developed in statistics, where the problem of *meta-analysis* has long been considered [Glass, 1976]. Meta-analysis is concerned with combining data from different experiments that address related problems.

Recently, an approach within the framework of regularization has been proposed [Evgeniou et al., 2005, Micchelli and Pontil, 2005]. It allows for "coupling"of the tasks through the regularizer, which is a function of all the task parameters. Equivalently, this can be viewed as regularization in the joint input/task space with a *multi-task kernel*. Also, some specific regularizers that penalize deviation from the mean of the task parameters or cluster the tasks are suggested.

Another recent algorithm for finding common structures shared across tasks alternates between regression and singular value decomposition of the tasks' parameters [Ando and Zhang, 2005]. The motivation of this algorithm is that shared structure can be encoded as a structural matrix parameter, essentially corresponding to a linear feature map, which can be learned from the data.

We mention in passing a few other of the many works on the topic, such as estimating gradients within multi-task regularization [Guinney et al., 2007]; multi-task feature selection with support vector machines [Jebara, 2004]; co-regularization for semi-supervised learning [Rosenberg and Bartlett, 2007]; learning a nearest neighbor distance metric while clustering tasks [Thrun and O'Sullivan, 1996]; a boosting algorithm combined with a hierarchical structure of features [Torralba et al., 2004]. In [Ben-David and Schuller, 2003] a specific multi-task paradigm, in which task relatedness is due to transformations of the input, is analyzed and improved multi-task bounds are obtained for some cases. In the aforementioned and other works, empirical studies typically indicate that simultaneously learning multiple related tasks significantly improves performance relative to learning each task independently.

The multi-task learning problem is also related to *collaborative filtering*. Collaborative filtering seeks to predict a consumer's preference to products, such as films or books, based on the choices of other consumers, especially when preferences are only partially known. As in multi-task learning, there is common structure that can be exploited since many consumers share opinions about certain types of products. One way to rephrase this question is *matrix factorization* – see, for example, [Abernethy et al., 2006, Ding et al., 2006, Lee and Seung, 2001, Srebro et al., 2005]. In matrix factorization, the objective is to compute two factors of a given target matrix which have a fixed (or low) rank. There is a clear analogy to multi-task learning without attributes, where the rows of the target matrix correspond to tasks and its columns to input coordinates. In addition, the low rank assumption translates to that of a small feature representation, which we have already discussed. Later, in Section 4.5, we will discuss the connection of some matrix factorization methods with the method we propose for multi-task learning.

In statistics, an approach that is often applied to problems with multiple related tasks is multilevel modeling [Gelman and Hill, 2007, Goldstein, 1991, Kreft and Leeuw, 1998]. It is a hierarchical type of method, in that it models both the data and the regression parameters. Other related statistical approaches include multivariate linear models such as reduced rank regression [Izenman, 1975], partial least squares [Wold et al., 1984] and canonical correlation analysis [Breiman and Friedman, 1997, Hotelling, 1936]. These methods are based on generalized eigenvalue problems – see, for example, [Borga, 1998, Chapter 4] for a nice review. They have also been extended to an RKHS setting – see, for example, [Bennett and Embrechts, 2003, Hardoon et al., 2004] and references therein.

4.2 Feature Selection and Learning

The problem of selecting or learning sparse representations has been extensively studied, either for single supervised tasks or for unsupervised learning. We briefly review two well studied methods which are related to our presentation in the following sections. Several other methods, such as Independent Component Analysis, dimensionality reduction methods etc., exist but they are outside the scope of this thesis.

Supervised feature selection is often performed with L_1 regularization. Like L_2 regularization, it balances an error term that fits the data and a smoothness term that favors simpler solutions:

$$\min \left\{ Q(X^\top v) + \gamma \|v\|_1^2 \;:\; v \in \mathbb{R}^d \right\} , \tag{4.2.1}$$

where Q is a convex, nonnegative loss function, $X \in \mathbb{R}^{d \times m}$ is the data matrix and $\gamma > 0$. The regularizer $\|v\|_1 = \sum_{i \in \mathbb{N}_d} |v_i|$ is the L_1 norm of v. Unlike the L_2 norm, this regularizer favors *sparse* solutions, that is, vectors v for which "many" components v_i are zero. In fact, problem (4.2.1) is a very common *convex relaxation* of L_0 regularization,

$$\min \left\{ Q(X^\top v) + \gamma \|v\|_0^2 \;:\; v \in \mathbb{R}^d \right\} , \tag{4.2.2}$$

where $\|v\|_0$ equals the cardinality of vector v, that is, the number of nonzero components.

There have been some theoretical results which confirm the above intuitions about sparsity. According to [Donoho, 2004], for losses like the square loss, there is high probability that the solution of (4.2.1) approximates well that of (4.2.2) whenever the L_0 norm of the latter is small (compared to the size of the data). Moreover, in most cases the L_0 norm is a nonincreasing function of the regularization parameter γ [Micchelli and Pinkus, 1994]. It has also been confirmed from experimental work that optimal L_1 norm solutions approximate well optimal L_0 solutions.

Among other fields, L_1 regularization has been studied in the statistics community under the name of “lasso” [Hastie et al., 2001], in signal processing as “basis pursuit” [Chen et al., 2001], in feature selection with support vector machines [Fung and Mangasarian, 2004] and in function approximation [Poggio and Girosi, 1998]. Such problems are usually solved with linear programs obtained by a suitable change of variables.

Regarding learning features from unsupervised data, a popular method with a long history is *principal component analysis* (PCA) – see, for example, [Duda and Hart, 1973, Schölkopf and Smola, 2002]. PCA finds those k orthogonal directions (principal components) whose subspace best aligns with the data. The number k is prescribed in advance or determined ad hoc. The algorithm is simple and first centers the data, then computes the covariance matrix $C = XX^\top$ and performs an eigendecomposition of C. The principal components are the eigenvectors of C corresponding to the k highest eigenvalues. There is also a version with kernels (kernel PCA) that involves only inner products of the data. We will observe some similarities of the method we propose in the following sections to PCA, since our method learns orthogonal features as well and involves an eigendecomposition of a covariance matrix.

4.3 Multi-Task Feature Learning

4.3.1 Overview

As mentioned in the previous section, the problem of learning (or selecting) sparse representations has been extensively studied either for single-task supervised learning (for example, using L_1 regularization) or for unsupervised learning – for example, using principal component analysis (PCA) or independent component analysis (ICA). In contrast, there has been only limited work [Ando and Zhang, 2005, Baxter, 2000, Jebara, 2004, Zhang et al., 2006] in the multi-task supervised learning setting.

In the following sections, we present a novel method for learning sparse representations common across many supervised learning tasks. In particular, we develop a novel multi-task generalization of the L_1 regularization, which is known to provide sparse variable *selection* in the single-task case. Our goal is to *learn* a few features common across the tasks using a regularizer which both *couples the tasks and enforces sparsity*. These features are orthogonal functions in a prescribed reproducing kernel Hilbert space. The number of common features learned is controlled, as we empirically show, by a regularization parameter – much like sparsity is controlled in the case of single-task L_1 regularization. Moreover, the method can be used, as a special case, for *variable selection*. We call "learning features" to be the estimation of new features which are functions of the input variables, like the features learned in the unsupervised setting by methods such as PCA. We call "selecting variables" to be simply the selection of some of the input variables (or prescribed features).

The objective of sparsity is a desirable one because of the intuition that related tasks should share representations (or parts thereof) and these representations should be as simple as possible. Thus, there is an "Ockham's razor" assumption of few features and an assumption that task relatedness is due to the sharing of these features among the tasks. Of course, not all situations in which we wish to learn multiple tasks fit in this framework, but we claim that it can be useful in a large number of cases. Other multi-task assumptions could be made and have been proposed in the literature (Section 4.1), and may be suitable for different situations.

Although the novel regularized problem we shall propose is not convex, we are going to show that it is equivalent to another optimization problem which is convex. To solve the latter we use an iterative algorithm which simultaneously learns *both* the features and the task functions through two alternating steps. The first step consists in independently learning the parameters of the tasks' regression or classification functions. The second step consists in learning, in an unsupervised way, a low-dimensional representation for these task parameters. This alternating algorithm will be shown to converge to an optimal solution of the convex and the (equivalent) original non-convex problem. Finally, we shall develop a nonlinear generalization of the proposed method using kernels.

The discussion of the following sections has appeared in [Argyriou et al., 2007a,b,c].

4.3.2 Learning Sparse Multi-Task Representations

Notation

We follow the same notational conventions as in Section 3.2 with the few additions below.

$\mathbb{R}^{\mu\times\nu}$	set of real $\mu \times \nu$ matrices
$\|w\|_p\,,\ p \geq 1$	L_p norm of w : $\left(\sum_{i\in\mathbb{N}_d} \|w_i\|^p\right)^{\frac{1}{p}}$
a^i	i-th row of matrix A
a_t	t-th column of matrix A
$\|A\|_{r,p}\,,\ r,p \geq 1$	$\left(\sum_{i\in\mathbb{N}_d} \|a^i\|_r^p\right)^{\frac{1}{p}}$
$\mathrm{trace}(X)$	trace of matrix X
$\mathrm{Diag}(w)$	the diagonal matrix having the components of vector w on the diagonal
$\mathrm{Diag}\,(w_i)_{i\in\mathbb{N}_\mu}$	the diagonal matrix having $\{w_i : i \in \mathbb{N}_\mu\}$ on the diagonal
$\mathrm{range}(X)\,,\ X \in \mathbb{R}^{\mu\times\nu}$	range of matrix X : $\{x \in \mathbb{R}^\mu : x = Xz$ for some $z \in \mathbb{R}^\nu\}$
$\mathrm{null}(X)\,,\ X \in \mathbb{R}^{\mu\times\nu}$	null space of matrix X : $\{x \in \mathbb{R}^\nu : Xx = 0\}$

$\mathbf{O}^{\mu}$	set of $\mu \times \mu$ orthogonal matrices

We shall call $\|A\|_{r,p}$ the (r,p)-norm of A.

Problem Formulation

Assume that we are given T supervised learning tasks. For every $t \in \mathbb{N}_T$, the corresponding task is identified by a function $f_t : \mathbb{R}^d \to \mathbb{R}$ (for example, a regressor or margin classifier). For each task, we are given a data set of m input/output data examples $\{(x_{ti}, y_{ti}) : i \in \mathbb{N}_m\} \subseteq \mathbb{R}^d \times \mathbb{R}$. For simplicity, we assume that each data set contains the same number of examples; however, our treatment in the following sections applies equally to the case that the number of data per task varies. Moreover, the data may be common to all (or some of) the tasks, or not. Also, it is possible that the y data are partially known, as in collaborative filtering.

We wish to design an algorithm which, based on the data above, computes all the functions f_t, $t \in \mathbb{N}_T$. We would also like such an algorithm to be able to uncover particular relationships across the tasks. Specifically, we study the case that the tasks are related in the sense that they *all share a small set of features*. Formally, our hypothesis is that the functions f_t can be represented as

$$f_t(x) = \sum_{i \in \mathbb{N}_I} a_{it} h_i(x) \qquad \forall t \in \mathbb{N}_T, x \in \mathbb{R}^d,$$

where $h_i : \mathbb{R}^d \to \mathbb{R},\ i \in \mathbb{N}_I$, are the I features and $a_{it} \in \mathbb{R}$ the regression parameters.

Our goal is to learn the features h_i, the parameters a_{it} and the number of features I from the data. For simplicity, we first consider the case that the features are linear homogeneous functions, that is, they are of the form $h_i(x) = \langle u_i, x \rangle$, where $u_i \in \mathbb{R}^d$. In Section 4.6, we will extend our formulation to the case that the h_i are elements of a reproducing kernel Hilbert space, hence in general nonlinear functions.

We make only one assumption about the features, namely that the vectors u_i are *orthogonal*. Thus, we may consider only up to d of those vectors for the linear case. This assumption, which is similar in spirit to that of *unsupervised* methods such as PCA, will enable us to develop a convex learning method in the next section. We leave extensions to other cases for future research.

Thus, if we denote by $U \in \mathbf{O}^d$ the matrix whose columns are the vectors u_i, the task functions can be written as

$$f_t(x) = \sum_{i \in \mathbb{N}_d} a_{it} \langle u_i, x \rangle = \langle a_t, U^\top x \rangle \qquad \forall t \in \mathbb{N}_T, x \in \mathbb{R}^d.$$

We use A to denote the matrix $(a_{it} : i \in \mathbb{N}_d, t \in \mathbb{N}_t)$, $a_t, t \in \mathbb{N}_T$, its columns and $a^i, i \in \mathbb{N}_d$, its rows. Our assumption that the tasks share a "small" set of features $I \leq d$ means that the matrix A has "many" rows which are identically equal to zero and for these rows the corresponding features (columns of matrix U) will not be used by any task. Rather than learning the number of features I directly, we introduce a regularization which favors a small number of nonzero rows in the matrix A.

Specifically, we introduce the regularization error function

$$\mathcal{E}(A, U) = \sum_{t \in \mathbb{N}_T} \sum_{i \in \mathbb{N}_m} L(y_{ti}, \langle a_t, U^\top x_{ti} \rangle) + \gamma \|A\|_{2,1}^2 \qquad \forall A \in \mathbb{R}^{d \times T}, U \in \mathbf{O}^d, \qquad (4.3.1)$$

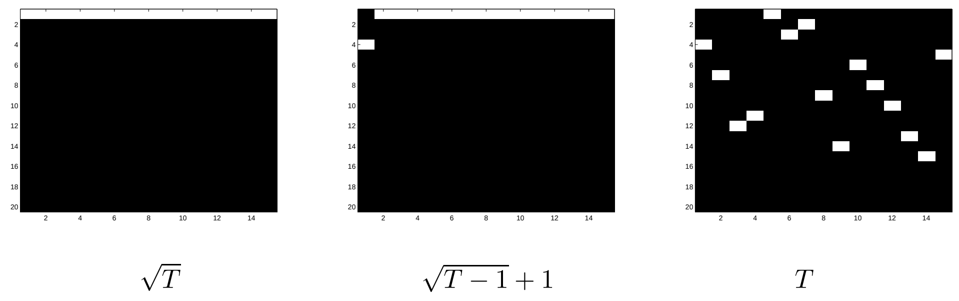


Figure 4.1: Values of the $(2,1)$-norm of a matrix containing T nonzero entries, equal to 1. When the norm increases, the level of sparsity along the rows decreases.

where $\gamma > 0$ is a regularization parameter. The first term in (4.3.1) is the average of the error across the tasks, measured according to a prescribed loss function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ which is convex in the second argument.[1] The second term is a regularization term which penalizes the $(2,1)$-norm of matrix A,

$$\|A\|_{2,1} := \sum_{i \in \mathbb{N}_d} \|a^i\|_2 \,.$$

It is obtained by first computing the 2-norms of the (across the tasks) rows a^i (corresponding to feature i) and then the 1-norm of the vector $b(A) = (\|a^1\|_2, \ldots, \|a^d\|_2)$. The magnitudes of the components of the vector $b(A)$ indicate how important each feature is.

The $(2,1)$-norm favors a small number of nonzero rows in the matrix A, in a way similar to the L_1 norm, thereby ensuring that common features will be selected across the tasks. This point is further illustrated in Figure 4.1, where we consider the case that the entries of matrix A take binary values and that there are exactly T entries which equal 1. The minimum value of the $(2,1)$-norm equals $\sqrt{T}$ and is obtained when the "1" entries are all aligned along one row. Instead, the maximum value equals T and is obtained when each "1" entry is placed in a different row (we assume here that $d \geq T$). This example also illustrates why plain L_1 or L_2 regularizers on the entries of matrix A are inappropriate for our purposes. Such regularizers cannot distinguish between the three matrices depicted in the figure. The standard L_1 norm, in particular, favors sparsity over the matrix but does not induce any structure aligning similar values on the same row.

When the feature matrix U is prescribed and $\hat{A}$ minimizes the convex function $\mathcal{E}(\cdot, U)$ the number of nonzero components of the vector $b(\hat{A})$ will typically be nonincreasing with γ. This sparsity property can be better understood by considering the case that there is only one task, say task t. In this case, function (4.3.1) is given by

$$\mathcal{E}(a_t, U) = \sum_{i \in \mathbb{N}_m} L(y_{ti}, \langle a_t, U^\top x_{ti} \rangle) + \gamma \|a_t\|_1^2 \,. \tag{4.3.2}$$

It is well known that using the L_1 norm leads to sparse solutions, that is, many components of the learned vector a_t are zero (see Section 4.2). Moreover, the number of nonzero components of a solution of problem (4.3.2) is typically a nonincreasing function of γ.

[1] We remark in passing that any convex function of inner products could be used in the error term. However, when the error term separates across tasks, like in (4.3.1), the computational task is easier (see Algorithm 3).

Since we do not simply want to select the features but also learn them, we further minimize the function $\mathcal{E}$ over U. Therefore, our approach for multi-task feature learning is to solve the optimization problem

$$\min\left\{\mathcal{E}(A,U) : U \in \mathbf{O}^d, A \in \mathbb{R}^{d\times T}\right\}. \tag{4.3.3}$$

This method learns a low-dimensional representation which is shared across the tasks. As in the single-task case, the number of features learned will be typically nonincreasing with the regularization parameter γ – we will present experimental evidence of this in Section 4.7.

We note that solving problem (4.3.3) is challenging for two main reasons. First, it is a non-convex problem, although it is separately convex in each of the variables A and U. Secondly, the regularizer $\|A\|_{2,1}^2$ is not smooth, which makes the optimization problem more difficult. In the next section, we shall show how to find a global optimal solution of this problem through solving an equivalent convex optimization problem. From this point on we assume that $A = 0$ does not minimize problem (4.3.3), which would clearly be a case of no practical interest.

We conclude by noting that when matrix U is not learned and we set $U = I_{d\times d}$, problem (4.3.3) selects a "small" set of *variables*, common across the tasks. In this case, we have the following *convex* optimization problem

$$\min\left\{\sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle a_t, x_{ti}\rangle) + \gamma\|A\|_{2,1}^2 \ : \ A \in \mathbb{R}^{d\times T}\right\}. \tag{4.3.4}$$

We are interested in this problem as well and will specialize our results and algorithms to it, during our discussion.[2]

4.3.3 Equivalent Convex Optimization Problem

In this section, we show that the non-convex problem (4.3.3) can be transformed into an equivalent convex problem. To this end, for every $W \in \mathbb{R}^{d\times T}$ with columns w_t and $D \in \mathbf{S}_+^d$, we define the function

$$\mathcal{R}(W,D) = \sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma\sum_{t\in\mathbb{N}_T}\langle w_t, D^+ w_t\rangle. \tag{4.3.5}$$

Under certain constraints, this objective function gives rise to a convex optimization problem, as we will see. Furthermore, even though the regularizer in $\mathcal{R}$ is still nonsmooth, in Appendix C we show that partial minimization with respect to D has a closed-form solution. This fact leads naturally to a globally convergent optimization algorithm that is discussed in Section 4.4.

Theorem 65 *Problem (4.3.3) is equivalent to the problem*

$$\begin{aligned}\min\big\{\mathcal{R}(W,D) \ : \ W \in \mathbb{R}^{d\times T},\ D \in \mathbf{S}_+^d,\ \operatorname{trace}(D) \le 1,\\ \operatorname{range}(W) \subseteq \operatorname{range}(D)\big\}.\end{aligned} \tag{4.3.6}$$

In particular, if $(\hat{A}, \hat{U})$ is an optimal solution of (4.3.3) then

$$(\hat{W}, \hat{D}) = \left(\hat{U}\hat{A}\,,\ \hat{U}\,\mathrm{Diag}\left(\frac{\|\hat{a}^i\|_2}{\|\hat{A}\|_{2,1}}\right)_{i\in\mathbb{N}_d}\hat{U}^\top\right)$$

[2]A similar regularization function was also independently developed by [Obozinski et al., 2006] for the purpose of multi-task feature selection.

is an optimal solution of problem (4.3.6); conversely, if $(\hat{W}, \hat{D})$ is an optimal solution of problem (4.3.6) then any $(\hat{A}, \hat{U})$, such that the columns of $\hat{U}$ form an orthonormal basis of eigenvectors of $\hat{D}$ and $\hat{A} = \hat{U}^\top \hat{W}$, is an optimal solution of problem (4.3.3).

To prove the theorem, we first introduce the following lemma which will be useful in our analysis.

Lemma 66 *For any $b = (b_1, \dots, b_d)^\top \in \mathbb{R}^d$ such that $b_i \neq 0, \forall i \in \mathbb{N}_d$, we have that*

$$\min\left\{ \sum_{i \in \mathbb{N}_d} \frac{b_i^2}{\lambda_i} \ : \ \lambda_i > 0, \ \sum_{i \in \mathbb{N}_d} \lambda_i \leq 1 \right\} = \|b\|_1^2 \tag{4.3.7}$$

and the minimizer is $\hat{\lambda}_i = \frac{|b_i|}{\|b\|_1}$, $i \in \mathbb{N}_d$.

PROOF. From the Cauchy-Schwarz inequality we have that

$$\|b\|_1 = \sum_{i \in \mathbb{N}_d} \lambda_i^{\frac{1}{2}} \lambda_i^{-\frac{1}{2}} |b_i| \leq \left(\sum_{i \in \mathbb{N}_d} \lambda_i \right)^{\frac{1}{2}} \left(\sum_{i \in \mathbb{N}_d} \lambda_i^{-1} b_i^2 \right)^{\frac{1}{2}} \leq \left(\sum_{i \in \mathbb{N}_d} \lambda_i^{-1} b_i^2 \right)^{\frac{1}{2}}.$$

The minimum is attained if and only if $\frac{\lambda_i^{\frac{1}{2}}}{\lambda_i^{-\frac{1}{2}} |b_i|} = \frac{\lambda_j^{\frac{1}{2}}}{\lambda_j^{-\frac{1}{2}} |b_j|}$ for all $i, j \in \mathbb{N}_d$ and $\sum_{i \in \mathbb{N}_d} \lambda_i = 1$. Hence the minimizer satisfies $\lambda_i = \frac{|b_i|}{\|b\|_1}$. ■

We can now prove Theorem 65.

Proof of Theorem 65. First suppose that (A, U) belongs to the feasible set of problem (4.3.3). Let $W = UA$ and $D = U \, \mathrm{Diag}\left(\frac{\|a^i\|_2}{\|A\|_{2,1}} \right)_{i \in \mathbb{N}_d} U^\top$. Then

$$\begin{aligned}
\sum_{t \in \mathbb{N}_T} \langle w_t, D^+ w_t \rangle &= \mathrm{trace}(W^\top D^+ W) \\
&= \mathrm{trace}\left(A^\top U^\top U \, \mathrm{Diag}\left(\|A\|_{2,1} \, \|a^i\|_2^+ \right)_{i \in \mathbb{N}_d} U^\top U A \right) \\
&= \|A\|_{2,1} \, \mathrm{trace}\left(A^\top \mathrm{Diag}\left(\|a^i\|_2^+ \right)_{i \in \mathbb{N}_d} A \right) \\
&= \|A\|_{2,1} \sum_{i \in \mathbb{N}_d} \|a^i\|_2^+ \, \|a^i\|_2^2 \ = \ \|A\|_{2,1}^2.
\end{aligned}$$

Therefore, $\mathcal{R}(W, D) = \mathcal{E}(A, U)$. Moreover, notice that W is a matrix multiple of the submatrix of U which corresponds to the nonzero a^i and hence to the nonzero eigenvalues of D. Thus, we obtain the range constraint in problem (4.3.6). Therefore, the infimum (4.3.6) (we will show below that the infimum is attained) does not exceed the minimum (4.3.3). Conversely, suppose that (W, D) belongs to the feasible set of problem (4.3.6). Let $D = U \mathrm{Diag}\left(\lambda_i \right)_{i \in \mathbb{N}_d} U^\top$ be an eigendecomposition and $A = U^\top W$. Then

$$\sum_{t \in \mathbb{N}_T} \langle w_t, D^+ w_t \rangle = \mathrm{trace}\left(A^\top \mathrm{Diag}\left(\lambda_i^+ \right)_{i \in \mathbb{N}_d} A \right) = \sum_{i \in \mathbb{N}_d} \lambda_i^+ \|a^i\|_2^2.$$

If $\lambda_i = 0$ for some $i \in \mathbb{N}_d$, then $u_i \in \text{null}(D)$, thus using the range constraint and $W = UA$ we deduce that $a^i = 0$. Consequently,

$$\sum_{i \in \mathbb{N}_d} \lambda_i^+ \|a^i\|_2^2 = \sum_{a^i \neq 0} \frac{\|a^i\|_2^2}{\lambda_i} \geq \left(\sum_{a^i \neq 0} \|a^i\|_2 \right)^2 = \|A\|_{2,1}^2 ,$$

where we have used Lemma 66. Therefore, $\mathcal{E}(A, U) \leq \mathcal{R}(W, D)$ and the minimum (4.3.3) does not exceed the infimum (4.3.6). Because of the above application of Lemma 66, we see that the infimum (4.3.6) is attained. Finally, the condition for the minimizer in Lemma 66 yields the relationship between the optimal solutions of problems (4.3.3) and (4.3.6). ■

In problem (4.3.6) we have bounded the trace of matrix D from above, because otherwise the optimal solution would be to simply set $D = \infty$ and only minimize the empirical error term in the right hand side of equation (4.3.5). Similarly, we have imposed the range constraint to ensure that the penalty term is bounded from below and away from zero. Indeed, without this constraint, it may be possible that $DW = 0$ when W does not have full rank, in which case there is a matrix D for which $\sum_{t \in \mathbb{N}_T} \langle w_t, D^+ w_t \rangle = \text{trace}(W^\top D^+ W) = 0$.

In fact, the presence of the range constraint in problem (4.3.6) is due to the presence of the pseudoinverse in $\mathcal{R}$. As the following corollary shows, it is possible to eliminate this constraint and obtain the smooth regularizer $\langle w_t, D^{-1} w_t \rangle$ at the expense of not always attaining the minimum.

Corollary 67 *Problem (4.3.6) is equivalent to the problem*

$$\inf \left\{ \mathcal{R}(W, D) \ : \ W \in \mathbb{R}^{d \times T}, \ D \in \mathbf{S}_{++}^d, \ \text{trace}(D) \leq 1 \right\} . \tag{4.3.8}$$

In particular, any minimizing sequence of problem (4.3.8) converges to a minimizer of problem (4.3.6).

PROOF. The corollary follows immediately from Theorem 65 and the equality of the $\min$ and $\inf$ values in Appendix C. ■

Returning to the discussion of Section 4.3.2 on the $(2, 1)$-norm, we note that the rank of the optimal matrix D indicates how many common relevant features the tasks share. Indeed, it is clear from Theorem 65 that the rank of matrix $\hat{D}$ equals the number of nonzero rows of matrix $\hat{A}$.

We also note that problem (4.3.6) is similar to that in [Evgeniou et al., 2007], where the regularizer is $\sum_{t \in \mathbb{N}_T} \langle (w_t - w_0), D^+ (w_t - w_0) \rangle$ instead of $\sum_{t \in \mathbb{N}_T} \langle w_t, D^+ w_t \rangle$ – that is, in our formulation we do not penalize deviations from a common "mean" w_0. However, the regularization problem including the tasks' mean can easily be rephrased to a problem like (4.3.6) but with an error term that mixes the tasks.

The next proposition establishes that problem (4.3.6) is convex.

Proposition 68 *Problem (4.3.6) is a convex optimization problem.*

PROOF. Let us define the function $f : \mathbb{R}^d \times \mathbf{S}^d \to \mathbb{R} \cup \{+\infty\}$ as

$$f(w, D) := \begin{cases} w^\top D^+ w & \text{if } D \in \mathbf{S}^d_+ \text{ and } w \in \mathrm{range}(D) \\ +\infty & \text{otherwise} \end{cases} .$$

With this definition, problem (4.3.6) is identical to minimizing the sum of T such functions plus the error term in (4.3.5), subject to the trace constraint. This is indeed true because the constraint $\mathrm{range}(W) \subseteq \mathrm{range}(D)$ is equivalent to the T constraints $w_t \in \mathrm{range}(D), t \in \mathbb{N}_T$. Also, the trace constraint is linear and hence convex. Thus, to show that problem (4.3.6) is convex, it suffices to show that f is convex. We show this by expressing f as a supremum of convex functions, more specifically as

$$f(w, D) = \sup\{w^\top v + \mathrm{trace}(ED) : E \in \mathbf{S}^d, v \in \mathbb{R}^d, 4E + vv^\top \in \mathbf{S}^d_-\} \quad \forall w \in \mathbb{R}^d, D \in \mathbf{S}^d.$$

To prove this equation, we first consider the case $D \notin \mathbf{S}^d_+$. We let u be an eigenvector of D corresponding to a negative eigenvalue and set $E = auu^\top, a \leq 0, v = 0$ to obtain that the supremum on the right equals $+\infty$. Next, we consider the case that $w \notin \mathrm{range}(D)$. We can write $w = Dz + n$, where $z, n \in \mathbb{R}^d, n \neq 0$ and $n \in \mathrm{null}(D)$. Thus,

$$w^\top v + \mathrm{trace}(ED) = z^\top Dv + n^\top v + \mathrm{trace}(ED)$$

and setting $E = -\frac{1}{4}vv^\top, v = an, a \geq 0$ we obtain $+\infty$ as the supremum. Finally, we assume that $D \in \mathbf{S}^d_+$ and $w \in \mathrm{range}(D)$. Combining with $E + \frac{1}{4}vv^\top \in \mathbf{S}^d_-$ we get that $\mathrm{trace}((E + \frac{1}{4}vv^\top)D) \leq 0$. Therefore

$$w^\top v + \mathrm{trace}(ED) \leq w^\top v - \frac{1}{4}v^\top Dv$$

and the expression on the right is maximized for $w = \frac{1}{2}Dv$ and obtains the maximal value

$$\frac{1}{2}v^\top Dv - \frac{1}{4}v^\top Dv = \frac{1}{4}v^\top Dv = \frac{1}{4}v^\top DD^+Dv = w^\top D^+ w.$$

This completes the proof. ■

Alternatively, the proposition can be seen to be a direct consequence of Theorem 52. Thus, problem (4.3.6) can be readily generalized to a family of regularization problems involving spectral regularizers, which will be discussed in Section 4.8.

We conclude this section by noting that when matrix D in problem (4.3.6) is additionally constrained to be diagonal, we obtain a problem equivalent to the variable selection problem (4.3.4). Formally, we have the following corollary.

Corollary 69 *Problem (4.3.4) is equivalent to the problem*

$$\min\left\{ \mathcal{R}(W, \mathrm{Diag}(\lambda)) \;:\; W \in \mathbb{R}^{d\times T},\ \lambda \in \mathbb{R}^d_+,\ \sum_{i\in\mathbb{N}_d} \lambda_i \leq 1, \right.$$
$$\left. w_i = 0 \text{ whenever } \lambda^i = 0 \right\} \tag{4.3.9}$$

and the optimal $\hat{\lambda}$ is given by

$$\hat{\lambda}_i = \frac{\|\hat{w}^i\|_2}{\|\hat{W}\|_{2,1}} \qquad \forall i \in \mathbb{N}_d. \tag{4.3.10}$$

4.4 Alternating Minimization Algorithm

In this section, we propose an algorithm for solving the convex optimization problem (4.3.6) which, as we prove, converges to an optimal solution. By Theorem 65 above this algorithm also provides a solution for the multi-task feature learning problem (4.3.3).

The main idea is to alternately minimize function $\mathcal{R}$ with respect to D and W. However, the correctness of such a scheme is not guaranteed in theory, because the ranges of W and D remain equal and constant throughout the algorithm (see Algorithm 3). In the step of learning W, it is easy to see that each w_t should belong in the range of D, by the Representer Theorem (2.2.3). In the step of learning D, the closed form implies that the eigenvectors of D coincide with the left singular vectors of W. For instance, if we start with $D = I_d$, the identity matrix, then at the next step the learned w_t will be the solutions of T independent ridge regressions, SVMs or similar and the algorithm will converge inside the range of this W. Thus, the outcome of Algorithm 3 with $\varepsilon = 0$ depends on the choice of the initial value for D.

Consequently, we choose instead to minimize a perturbation of the objective function $\mathcal{R}$ with a small parameter $\varepsilon > 0$. The perturbed objective can be minimized with an alternating algorithm, which converges to the optimal solution as we show in the next section. This allows us to prove convergence to an optimal solution of problem (4.3.6) by letting $\varepsilon \to 0$. However, in practice we have observed that alternating minimization of the unperturbed objective function $\mathcal{R}$ converges to an optimal solution of (4.3.6).[3]

More specifically, Algorithm 3 minimizes the function $\mathcal{R}_\varepsilon : \mathbb{R}^{d\times T} \times \mathbf{S}^d_{++} \to \mathbb{R}$, given by

$$\mathcal{R}_\varepsilon(W,D) = \sum_{t\in\mathbb{N}_T} \sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma\, \mathrm{trace}(D^{-1}(WW^\top + \varepsilon I_d)),$$

which keeps D nonsingular. The regularizer in this function is smooth and strictly convex, hence $\mathcal{R}_\varepsilon$ has a *unique* minimizer.

We now describe the two steps of Algorithm 3 for minimizing $\mathcal{R}_\varepsilon$. In the first step (we call this the W-step), we keep D fixed and minimize over W, that is, we solve the problem

$$\min \left\{ \sum_{t\in\mathbb{N}_T} \sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma \sum_{t\in\mathbb{N}_T} \langle w_t, D^{-1} w_t\rangle \ : \ W \in \mathbb{R}^{d\times T} \right\},$$

where, recall, w_t are the columns of matrix W. This minimization can be carried out independently across the tasks since the regularizer decouples when D is fixed. More specifically, introducing new variables for $D^{-\frac{1}{2}} w_t$ yields a standard L_2 regularization problem for each task with the *same* kernel $K(x,z) = \langle x, Dz\rangle$, $\forall x, z \in \mathbb{R}^d$.

[3]This is probably due to round-off effects that act in a way similar to perturbation.

Algorithm 3 Multi-Task Feature Learning

```
Input: training sets $\{(x_{ti}, y_{ti}) : i \in \mathbb{N}_m\}, t \in \mathbb{N}_T$
Parameters: regularization parameter $\gamma$, tolerances $\varepsilon, tol$
Output: $d \times d$ matrix $D$, $d \times T$ regression matrix $W = (w_1, \dots, w_T)$
Initialization: set $D = \frac{I_d}{d}$
while $\|W - W_{prev}\| > tol$ do
    for $t = 1, \dots, T$ do
        compute $w_t = \text{argmin}\left\{\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle w, x_{ti}\rangle) + \gamma\langle w, D^{-1}w\rangle : w \in \mathbb{R}^d\right\}$
    end for
    set $D = \frac{(WW^\top + \varepsilon I_d)^{\frac{1}{2}}}{\text{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}}$
end while
```

In the second step (we call this the D-step), we keep matrix W fixed, and minimize $\mathcal{R}_\varepsilon$ with respect to D. To this end, we solve the problem

$$\min\left\{\sum_{t\in\mathbb{N}_T}\langle w_t, D^{-1}w_t\rangle + \varepsilon\,\text{trace}(D^{-1}) : D \in \mathbf{S}^d_{++}, \text{trace}(D) \leq 1\right\}. \tag{4.4.1}$$

The term $\text{trace}(D^{-1})$ keeps the D-iterates of the algorithm at a certain distance from the boundary of $\mathbf{S}^d_+$ and plays a role similar to that of the barrier used in interior-point methods. In Appendix C, we show that the optimal solution of problem (4.4.1) is given by

$$D_\varepsilon(W) = \frac{(WW^\top + \varepsilon I_d)^{\frac{1}{2}}}{\text{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}} \tag{4.4.2}$$

and the optimal value equals $\left(\text{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}\right)^2$. In the same appendix, we also show that for $\varepsilon = 0$, equation (4.4.2) gives the minimizer of the function $\mathcal{R}(W, \cdot)$ subject to the constraints in problem (4.3.6).

Algorithm 3 can be interpreted as alternately performing a supervised and an unsupervised step. In the supervised step we learn task-specific functions (namely the vectors w_t) using a common representation across the tasks. This is because D encapsulates the features u_i and thus the feature representation is kept fixed. In the unsupervised step, the regression functions are fixed and we learn the common representation. In effect, the $(2, 1)$-norm criterion favors the most concise representation which "models" the regression functions through $W = UA$. Moreover, the matrix D learned in this step is a function of the covariance of the task parameters, $WW^\top$.

We conclude this section with the case of variable selection, problem (4.3.4). Using Corollary 69, we can make a simple modification to Algorithm 3 so that it can be used to solve problem (4.3.9). Specifically, we modify the D-step to be $D = Diag(\lambda)$, where the vector $\lambda = (\lambda_1, \dots, \lambda_d)^\top$ is computed using equation (4.3.10).

4.4.1 Convergence

We now briefly mention some convergence properties of Algorithm 3. We state here only the main results and postpone their proofs to Appendix D. Let us denote the value of W at the n-th iteration by $W^{(n)}$. First, we observe that, by construction, the values of the objective are nonincreasing, that is,

$$\mathcal{R}_\varepsilon(W^{(n+1)}, D_\varepsilon(W^{(n+1)})) \leq \mathcal{R}_\varepsilon(W^{(n+1)}, D_\varepsilon(W^{(n)})) \leq \mathcal{R}_\varepsilon(W^{(n)}, D_\varepsilon(W^{(n)}))\,.$$

These values are also bounded, since L is bounded from below, and thus the iterates of the objective function converge. Moreover, the iterates $W^{(n)}$ also converge as stated in the following theorem.

Theorem 70 *If $\varepsilon > 0$ then the sequence $\{(W^{(n)}, D_\varepsilon(W^{(n)}) : n \in \mathbb{N}\}$ converges to the minimizer of $\mathcal{R}_\varepsilon$ subject to the constraints in (4.4.1).*

Algorithm 3 minimizes the perturbed objective $\mathcal{R}_\varepsilon$. In order to obtain a minimizer of the original objective $\mathcal{R}$, we can employ a modified algorithm in which ε is reduced towards zero whenever $W^{(n)}$ has stabilized near a value. Our next theorem shows that the limiting points of such an algorithm are optimal.

Theorem 71 *Consider a sequence $\{\varepsilon_\ell > 0 : \ell \in \mathbb{N}\}$ which converges to zero. Let $(W_\ell, D_{\varepsilon_\ell}(W_\ell))$ be the minimizer of $\mathcal{R}_{\varepsilon_\ell}$ subject to the constraints in (4.4.1), for every $\ell \in \mathbb{N}$. Then any limiting point of the sequence $\{(W_\ell, D_{\varepsilon_\ell}(W_\ell)) : \ell \in \mathbb{N}\}$ is an optimal solution to problem (4.3.6).*

4.5 Relation to Trace Norm Regularization

We proceed with a few remarks on an alternative formulation for problem (4.3.6). By substituting equation (4.4.2) with $\varepsilon = 0$ in the equation (4.3.5) for $\mathcal{R}$, we obtain a regularization problem in W only, which is given by

$$\min\left\{\sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma\|W\|_{\mathrm{tr}}^2 \;:\; W \in \mathbb{R}^{d\times T}\right\}. \tag{4.5.1}$$

where we have defined $\|W\|_{\mathrm{tr}} := \mathrm{trace}(WW^\top)^{\frac{1}{2}}$.

The expression $\|W\|_{\mathrm{tr}}$ in the regularizer is called the *trace norm*. It can also be expressed as the sum of the singular values of W. As shown in [Fazel et al., 2001], the trace norm is the convex envelope of $\mathrm{rank}(W)$ in the unit ball, which gives another interpretation of the relationship between the rank and γ in our experiments. We also note that a similar problem has been studied in [Srebro et al., 2005] for the particular case of an SVM loss function. It was shown there that the optimization problem in the SVM case can be solved through an equivalent semidefinite programming problem.

Thus, one possible approach is to solve problem (4.5.1) directly, using a matrix optimization method. This can be costly if dT is large and hard to implement because the trace norm is nonsmooth. The other strategy, which we have opted for in Algorithm 3, is to use singular value decompositions (D-step) and smaller-size vector problems (W-step). This approach also has the advantage of simplicity and a natural interpretation. A similar algorithm is to smoothen the trace norm and optimize using gradient descent. However, as demonstrated in the experiments of Section 4.8.3, gradient descent is significantly slower than alternating minimization.

4.6 Learning Nonlinear Features

In this section, we consider the case that the features are associated to a kernel and hence that they are in general nonlinear functions of the input variables. First, in Section 4.6.1 we use a representer theorem for any optimal solution of problem (4.3.6), in order to obtain an optimization problem of bounded dimensionality. Then, in Section 4.6.2 we show how to solve this problem using an algorithm which is a variation of Algorithm 3.

4.6.1 A Representer Theorem

We begin by restating our optimization problem when the functions learned belong to a *reproducing kernel Hilbert space* (see Section 2.2). Formally, we now wish to learn T regression functions $f_t, t \in \mathbb{N}_T$, of the form

$$f_t(x) = \langle a_t, U^\top \varphi(x) \rangle = \langle w_t, \varphi(x) \rangle \qquad \forall x \in \mathbb{R}^d,$$

where $\varphi : \mathbb{R}^d \to \mathbb{R}^M$ is a prescribed feature map. This map will, in general, be nonlinear and its dimensionality M may be large. In fact, the theoretical and algorithmic results which follow apply to the case of an infinite dimensionality as well. As typical, we assume that the kernel function $K(x, x') = \langle \varphi(x), \varphi(x') \rangle$ is given. As before, in the following we will use the subscript notation for the columns of a matrix, for example w_t denotes the t-th column of matrix W.

We begin by recalling that Appendix C applied to problem (4.3.6) leads to a problem in W with the trace norm as the regularizer. Modifying slightly to account for the feature map, we obtain the problem

$$\min \left\{ \sum_{t \in \mathbb{N}_T} \sum_{i \in \mathbb{N}_m} L(y_{ti}, \langle w_t, \varphi(x_{ti}) \rangle) + \gamma \|W\|_{\mathrm{tr}}^2 \ : \ W \in \mathbb{R}^{d \times T} \right\}. \tag{4.6.1}$$

This problem can be viewed as a generalization of the standard L_2 regularization problem. Indeed, in the case $t = 1$ the trace norm $\|W\|_{\mathrm{tr}}$ is simply equal to $\|w_1\|_2$. In this case, it is well known that an optimal solution $\hat{w} \in \mathbb{R}^d$ of such a problem is in the span of the training data, that is

$$\hat{w} = \sum_{i \in \mathbb{N}_m} c_i \, \varphi(x_i)$$

for some $\{c_i \in \mathbb{R} : i \in \mathbb{N}_m\}$. This result is known as the Representer Theorem – see Section 2.2.2. We now extend this result to the more general form (4.6.1). [4]

Theorem 72 *If $\hat{W}$ is an optimal solution of problem (4.6.1) then for every $t \in \mathbb{N}_T$ there exists a vector $c_t \in \mathbb{R}^{mT}$ such that*

$$\hat{w}_t = \sum_{s \in \mathbb{N}_T} \sum_{i \in \mathbb{N}_m} (c_t)_{si} \varphi(x_{si}). \tag{4.6.2}$$

PROOF. Let $\mathcal{L} = \mathrm{span}\{\varphi(x_{si}) : s \in \mathbb{N}_T, i \in \mathbb{N}_m\}$. We can write $w_t = p_t + n_t$, $t \in \mathbb{N}_T$, where $p_t \in \mathcal{L}$ and $n_t \in \mathcal{L}^\perp$. Hence $W = P + N$, where P is the matrix with columns p_t and N the matrix with columns n_t. Moreover we have that $P^\top N = 0$. From Lemma 79 in Appendix E, we obtain that

[4] A representer theorem for a similar problem has been proven in [Abernethy et al., 2006]. Here, we give an alternative proof connected to the theory of matrix monotone functions. We also note that this theorem can be extended to a general family of spectral norms [Argyriou et al., 2007d].

$\|W\|_{\mathrm{tr}} \geq \|P\|_{\mathrm{tr}}$. We also have that $\langle w_t, \varphi(x_{ti})\rangle = \langle p_t, \varphi(x_{ti})\rangle$. Thus, we conclude that whenever W is optimal, N must be zero. ■

An alternative way to write (4.6.2), using matrix notation, is to express $\hat{W}$ as a multiple of the input matrix. The latter is the matrix $\varPhi \in \mathbb{R}^{M\times mT}$ whose (t,i)-th column is the vector $\varphi(x_{ti}) \in \mathbb{R}^M$, $t \in \mathbb{N}_T, i \in \mathbb{N}_m$. Hence, denoting with $C \in \mathbb{R}^{mT\times T}$ the matrix with columns c_t, equation (4.6.2) becomes

$$\hat{W} = \varPhi C. \tag{4.6.3}$$

We now apply Theorem 72 to problem (4.6.1) in order to obtain an equivalent optimization problem in a number of variables independent of M. This theorem implies that we can restrict the feasible set of (4.6.1) only to matrices $W \in \mathbb{R}^{d\times T}$ satisfying (4.6.3) for some $C \in \mathbb{R}^{mT\times T}$.

Let $\mathcal{L} = \mathrm{span}\{\varphi(x_{ti}) : t \in \mathbb{N}_T, i \in \mathbb{N}_m\}$ as above and let δ its dimensionality. In order to exploit the *orthogonal invariance* of the trace norm, we consider a matrix $V \in \mathbb{R}^{M\times\delta}$ whose columns form an orthogonal basis of $\mathcal{L}$. Equation (4.6.3) implies that there is a matrix $\varTheta \in \mathbb{R}^{\delta\times T}$, whose columns we denote by ϑ_t, $t \in \mathbb{N}_T$, such that

$$W = V\varTheta\,. \tag{4.6.4}$$

Substituting equation (4.6.4) in the objective of (4.6.1) yields the objective function

$$\begin{aligned}\sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle V\vartheta_t, \varphi(x_{ti})\rangle) + \gamma\Big(\mathrm{trace}(V\varTheta\varTheta^\top V^\top)^{\frac{1}{2}}\Big)^2 &= \\ \sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle \vartheta_t, V^\top\varphi(x_{ti})\rangle) + \gamma\Big(\mathrm{trace}(\varTheta\varTheta^\top)^{\frac{1}{2}}\Big)^2 &= \\ \sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle \vartheta_t, V^\top\varphi(x_{ti})\rangle) + \gamma\|\varTheta\|_{\mathrm{tr}}^2\,. &\end{aligned}$$

Thus, we obtain the following proposition.

Proposition 73 *Problem (4.6.1) is equivalent to*

$$\min\left\{\sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, \langle\vartheta_t, z_{ti}\rangle) + \gamma\|\varTheta\|_{\mathrm{tr}}^2 \;:\; \varTheta \in \mathbb{R}^{\delta\times T}\right\}, \tag{4.6.5}$$

where

$$z_{ti} = V^\top\varphi(x_{ti}) \qquad\qquad \forall t \in \mathbb{N}_T, i \in \mathbb{N}_m. \tag{4.6.6}$$

Moreover, there is an one-to-one correspondence between optimal solutions of (4.6.1) and those of (4.6.5), given by $\hat{W} = V\hat{\varTheta}$.

Problem (4.6.5) is a problem in δT variables, where $\delta T \leq mT^2$, and hence it can be tractable regardless of the dimensionality M of the original feature map.

Algorithm 4 Multi-Task Feature Learning with Kernels

```
**Input:** training sets $\{(x_{ti}, y_{ti}) : i \in \mathbb{N}_m\}, t \in \mathbb{N}_T$
**Parameters:** regularization parameter $\gamma$, tolerances $\varepsilon, tol$
**Output:** $\delta \times T$ coefficient matrix $B = (b_1, \dots, b_T)$, indices $\{(t_\nu, i_\nu) : \nu \in \mathbb{N}_\delta\} \subseteq \mathbb{N}_T \times \mathbb{N}_m$
**Initialization:**
    • using only the kernel values, find a matrix $R \in \mathbb{R}^{\delta \times \delta}$ and indices $\{(t_\nu, i_\nu) : \nu \in \mathbb{N}_\delta\}$ such that $\left\{\sum_{\nu \in \mathbb{N}_\delta} \varphi(x_{t_\nu i_\nu}) r_{\nu\mu} : \mu \in \mathbb{N}_\delta\right\}$ form an orthogonal basis for the space generated by the feature map applied on the training data
    • compute the modified inputs $z_{ti} = R^\top \Big(K(x_{t_\nu i_\nu}, x_{ti}) : \nu \in \mathbb{N}_\delta\Big), \forall t \in \mathbb{N}_T, i \in \mathbb{N}_m$
    • set $\Delta = \frac{I_\delta}{\delta}$
**while** $\|\Theta - \Theta_{prev}\| > tol$ **do**
    **for** $t = 1, \dots, T$ **do**
        compute $\vartheta_t = \text{argmin}\left\{\sum_{i \in \mathbb{N}_m} L(y_{ti}, \langle \vartheta, z_{ti}\rangle) + \gamma \langle \vartheta, \Delta^{-1}\vartheta\rangle : \vartheta \in \mathbb{R}^\delta\right\}$
    **end for**
    set $\Delta = \frac{(\Theta\Theta^\top + \varepsilon I_\delta)^{\frac{1}{2}}}{\text{trace}(\Theta\Theta^\top + \varepsilon I_\delta)^{\frac{1}{2}}}$
**end while**
return $B = R\Theta$ and $\{(t_\nu, i_\nu) : \nu \in \mathbb{N}_\delta\}$
```

4.6.2 An Alternating Algorithm for Nonlinear Features

We now address how to solve problem (4.6.5) by applying the same strategy as in Algorithm 3. It is clear from the discussion in Section 4.4 that (4.6.5) can be solved with an alternating minimization algorithm, which we present as Algorithm 4.

In the initialization step, Algorithm 4 computes a matrix $R \in \mathbb{R}^{\delta \times \delta}$ which relates the orthogonal basis V of $\mathcal{L}$ with a basis $\{\varphi(x_{t_\nu i_\nu}) : \nu \in \mathbb{N}_\delta, t_\nu \in \mathbb{N}_T, i_\nu \in \mathbb{N}_m\}$ from the inputs. We can write this relation as

$$V = \widetilde{\Phi} R \tag{4.6.7}$$

where $\widetilde{\Phi} \in \mathbb{R}^{M \times \delta}$ is the matrix whose ν-th column is the vector $\varphi(x_{t_\nu i_\nu})$.

To compute R using only Gram matrix entries, one approach is Gram-Schmidt orthogonalization. At each step, we consider an input x_{ti} and determine whether it enlarges the current subspace or not by computing kernel values with the inputs forming the subspace. However, Gram-Schmidt orthogonalization is sensitive to round-off errors, which can affect the accuracy of the solution (see [Golub and van Loan, 1996, Sec. 5.2.8]). A more stable but computationally less appealing approach is to compute an eigendecomposition of the $mT \times mT$ Gram matrix $\Phi^\top \Phi$. A middle strategy may be preferable, namely, randomly select a reasonably large number of inputs and compute an eigendecomposition of their Gram matrix; obtain the basis coefficients; complete the vector space with a Gram-Schmidt procedure.

After the computation of R, the algorithm computes the inputs in (4.6.6), which by (4.6.7) equal $z_{ti} = V^\top \varphi(x_{ti}) = R^\top \widetilde{\Phi}^\top \varphi(x_{ti}) = R^\top \widetilde{K}(x_{ti})$, where $\widetilde{K}(x)$ denotes the δ-vector with entries

$K(x_{t_\nu i_\nu}, x)$, $\nu \in \mathbb{N}_\delta$. In the main loop, there are two steps. The first one (Θ-step) solves T independent regularization problems using the Gram entries $z_{ti}^\top \Delta z_{tj}$, $i, j \in \mathbb{N}_m, t \in \mathbb{N}_T$. The second one ($\Delta$-step) is the computation of a $\delta \times \delta$ matrix square root.

Finally, the output of the algorithm, matrix B, satisfies that

$$W = \widetilde{\Phi} B \tag{4.6.8}$$

by combining equations (4.6.4) and (4.6.7). Thus, a prediction on a new input $x \in \mathbb{R}^d$ is computed as

$$f_t(x) = \langle w_t, \varphi(x) \rangle = \langle b_t, \widetilde{K}(x) \rangle \qquad \forall t \in \mathbb{N}_T.$$

One can also express the learned features in terms of the input basis $\{\varphi(x_{t_\nu i_\nu}) : \nu \in \mathbb{N}_\delta\}$. To do this, we need to compute an eigendecomposition of $B \widetilde{\Phi}^\top \widetilde{\Phi} B$. Indeed, we know that

$$W = \bar{U} \Sigma Q^\top , \tag{4.6.9}$$

where $\bar{U} \in \mathbb{R}^{M \times \delta'}$ is orthogonal, $\Sigma \in \mathbf{S}_{++}^{\delta'}$ diagonal, $Q \in \mathbb{R}^{T \times \delta'}$ orthogonal, $\delta' \leq \delta$, and the columns of $\bar{U}$ are the significant features learned. From this and (4.6.8) we obtain that $\bar{U} = \widetilde{\Phi} B Q \Sigma^{-1}$ and Σ, Q can be computed from

$$Q \Sigma^2 Q^\top = W^\top W = B^\top \widetilde{\Phi}^\top \widetilde{\Phi} B \,.$$

Finally, the coefficient matrix[5] A can be computed from $W = UA$ (Theorem 65) and (4.6.9), yielding

$$A = \begin{pmatrix} \Sigma Q^\top \\ 0 \end{pmatrix} .$$

The computational cost of Algorithm 4 depends mainly on the dimensionality δ. Note that kernel evaluations using K appear only in the initialization step. There are $O(\delta m T)$ kernel computations during the orthogonalization process and $O(\delta^2 m T)$ additional operations for computing the vectors z_{ti}. However, these costs are incurred only once. Within each iteration, the cost of computing the Gram matrices in the Θ-step is $O(\delta^2 m T)$ and the cost of each learning problem depends on δ. The matrix square root computation in the Δ-step involves $O(\delta^3)$ operations. Thus, for most commonly used loss functions, the overall expected cost of the algorithm is $O(\delta^2 m T)$ operations. In particular, in several cases of interest, such as when all tasks share the same training inputs, δ can be small and Algorithm 4 can be particularly efficient. We would also like to note here that experimental trials, which are reported in Section 4.7, showed that usually between 20 and 100 iterations were sufficient for Algorithms 3 and 4 to converge.

As a final remark, we note that an algorithm similar to Algorithm 4 would not work for variable selection. This is true because Representer Theorem 72 does not apply to the optimization problem (4.3.9), where matrix D is constrained to be diagonal. Thus, multi-task variable selection – and in particular L_1 regularization – with kernels is not feasible. Nevertheless, this fact does not seem to be

[5]Recall that all simultaneous permutations of the columns of U and rows of A give valid solutions.

significant in the multi-task context of this thesis. As we will discuss in Section 4.7, variable selection was outperformed by feature learning in our experimental trials. However, variable selection could still be important in a different setting, when a set including some "good" features is a priori given and the question is how to select exactly these features.

4.7 Experiments

In this section, we present experiments with our methods, both the linear Algorithm 3 and the nonlinear Algorithm 4. We have used both synthetic (where we know what the underlying features used in all tasks are) and real datasets. The results show that, in agreement with previous work (Section 4.1), multi-task learning improves performance relative to single-task learning when the tasks are related. More importantly, the results confirm that when the tasks are related in the way we have defined our algorithm learns a small number of features which are common across the tasks. In all experiments, we used the square loss function and automatically tuned the regularization parameter γ by selecting among the values $\{10^r : r \in \{-6, \ldots, 3\}\}$ with 5-fold cross-validation.

4.7.1 Synthetic Data

We first used synthetic data to test the ability of the algorithms to learn the common across tasks features. This setting makes it possible to evaluate the quality of the features learned, as in this case we know what the common across tasks features are.

Linear Synthetic Data

We consider the case of regression and a number of up to $T = 200$ tasks. Each of the w_t parameters of these tasks was selected from a 5-dimensional Gaussian distribution with zero mean and covariance $Cov = \mathrm{Diag}(1, 0.64, 0.49, 0.36, 0.25)$. To these 5-dimensional w_t's we kept adding up to 10 irrelevant dimensions which are exactly zero. The training and test data were generated uniformly from $[0, 1]^d$ where d ranged from 5 to 15. The outputs y_{ti} were computed from the w_t and x_{ti} as $y_{ti} = \langle w_t, x_{ti} \rangle + n$, where n is zero-mean Gaussian noise with standard deviation equal to 0.1. Thus, the true features $\langle u_i, x \rangle$ we wish to learn were in this case just 5 of the input variables. However, we did not a priori assume this and we let our algorithm learn – not select – the features. That is, we used Algorithm 3 to learn the features, not its variant which performs variable selection (see our discussion at the end of Section 4.4). The desired result is a feature matrix U which is close to the identity matrix (on 5 columns) and a matrix D approximately proportional to the covariance Cov used to generate the task parameters (on a 5×5 principal submatrix). In this experiment, we did not use a bias term.

We generated 5 and 20 examples per task for training and testing, respectively. To test the effect of the number of jointly learned tasks on the test performance and (more importantly) on the quality of the features learned, we used our methods with $T = 10, 25, 100, 200$ tasks. For $T = 10, 25$ and 100, we averaged the performance metrics over randomly selected subsets of the 200 tasks, so that our estimates have comparable variance. We also estimated each of the 200 tasks independently using standard ridge regressions.

We present, in Figure 4.2, the impact of the number of simultaneously learned tasks on the test

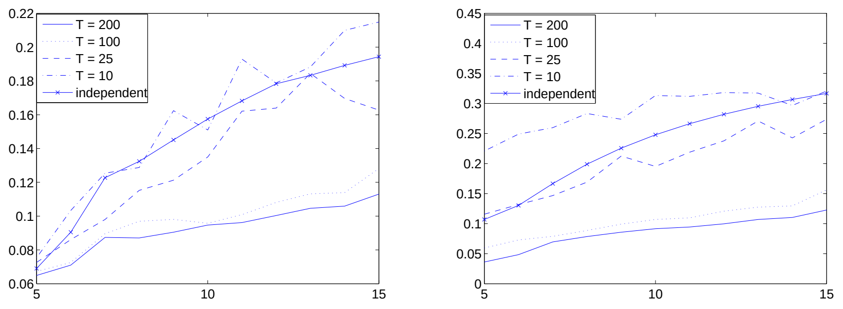


Figure 4.2: Linear synthetic data. Left: test error versus the number of irrelevant variables, as the number of tasks changes. Right: Frobenius norm of the difference of the learned and actual matrices D versus the number of irrelevant variables, as the number of tasks changes. This is a measure of the quality of the learned features.

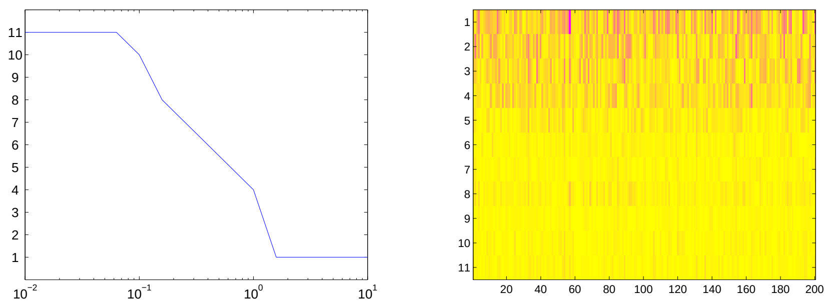


Figure 4.3: Linear synthetic data. Left: number of features learned versus the regularization parameter γ for 6 irrelevant variables. Right: matrix A learned, indicating the importance of the learned features – the first 5 rows correspond to the true features (see text). The color scale ranges from yellow (low values) to purple (high values).

performance as well as the quality of the features learned, as the number of irrelevant variables increases. First, as the left plot shows, in agreement with past empirical and theoretical evidence – see Section 4.1 – learning multiple tasks together significantly improves on learning the tasks independently, as the tasks are indeed related in this case. Moreover, performance improves as the number of tasks increases. More important, this improvement increases with the number of irrelevant variables.

The plot on the right of Figure 4.2 is the most relevant one for our purposes. It shows the distance of the learned features from the actual ones used to generate the data. More specifically, we depict the Frobenius norm of the difference of the learned 5×5 principal submatrix of D and the actual Cov matrix (normalized to have trace 1). We observe that adding more tasks leads to better estimates of the underlying features. Moreover, as for the test performance, the relative (as the number of tasks increases) quality of the features learned increases with the number of irrelevant variables.

We also tested the effect of the regularization parameter γ on the number of learned features (as measured by $\operatorname{rank}(D)$) for 6 irrelevant variables. We show the results on the left plot of Figure 4.3. As

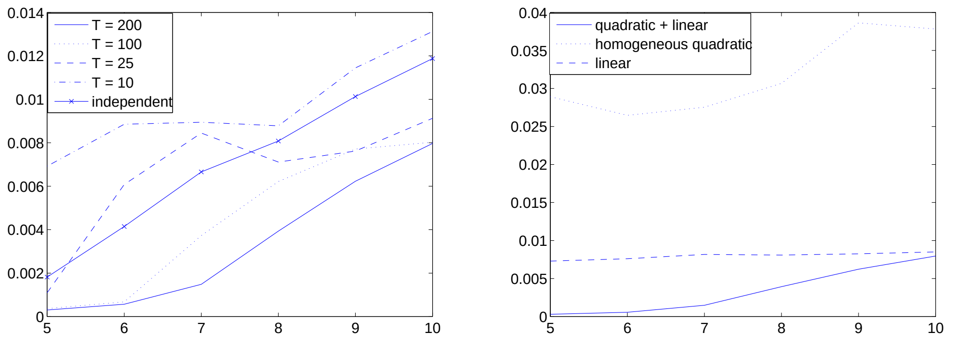


Figure 4.4: Nonlinear synthetic data. Left: test error versus number of variables as the number of simultaneously learned tasks changes, using a quadratic + linear kernel. Right: test error versus number of variables for 200 tasks, using three different kernels (see text).

expected, the number of features learned decreases with γ. Finally, the right plot in Figure 4.3 shows the absolute values of the elements of the matrix A learned using the parameter γ selected by cross-validation. This is the resulting matrix for 6 irrelevant variables and all 200 simultaneously learned tasks. This plot indicates that our algorithm learns a matrix A with the expected structure: there are only five important features. The (normalized) 2-norms of the corresponding rows are $0.31, 0.21, 0.12, 0.10$ and 0.09 respectively, while the true values (diagonal elements of Cov scaled to have trace 1) are $0.36, 0.23, 0.18, 0.13$ and 0.09 respectively.

Nonlinear Synthetic Data

Next, we tested whether our nonlinear method (Algorithm 4) can outperform the linear one when the true underlying features are nonlinear. For this purpose, we created a new synthetic data set in a similar way as before, but this time we used a feature map $\phi : \mathbb{R}^5 \to \mathbb{R}^7$. More specifically, we have 6 relevant linear and quadratic features and a bias term: $\varphi(x) = \big(x_1^2, x_4^2, x_1x_2, x_3x_5, x_2, x_4, 1\big)$. That is, the outputs were generated as $y_{ti} = \langle w_t, \varphi(x_{ti})\rangle + n$, with the task parameters w_t corresponding to the features above selected from a 7-dimensional Gaussian distribution with zero mean and covariance equal to $\mathrm{Diag}(0.5, 0.25, 0.1, 0.05, 0.15, 0.1, 0.15)$. All other components of each w_t were 0. The training and test sets were selected randomly from $[0,1]^d$ with d ranging from 5 to 10, and each contained 20 examples per task. Since there are more task parameters to learn than in the linear case, we used more data per task for training in this simulation. In the execution of our method, we did not augment the input with a bias term.

We report the results in Figure 4.4. As for the linear case, the left plot in the figure shows the test performance versus the number of simultaneously learned tasks, as the number of irrelevant variables increases. Note that the dimensionality of the feature map scales quadratically with the input dimensionality shown on the x-axis of the plot. The kernel used for this plot was $K_{ql}(x, x') := (x^\top x' + 1)^2$. This is a "good" kernel for this data set because the corresponding features include all of the monomials of φ. The results are qualitatively similar to those in the linear case. Learning multiple tasks together

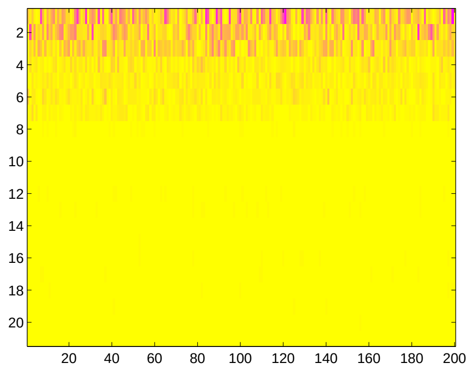


Figure 4.5: Matrix A learned in the nonlinear synthetic data experiment. The first 7 rows correspond to the true features (see text).

improves on learning the tasks independently. In this experiment, a certain number of tasks (greater than 10) is required for improvement over independent learning.

Next, we tested the effects of using the "wrong" kernel, as well as the difference between using a nonlinear kernel versus using a linear one. We used three different kernels. One is the sum of the quadratic and linear kernels defined above, the second is $K_q(x, x') := (x^\top x')^2$ and the third is $K_l(x, x') := x^\top x' + 1$. The results are shown on the right plot of Figure 4.4. First, notice that since the underlying feature map involves both quadratic and linear features, it would be expected that the first kernel gives the best results, and this is indeed true. Second, notice that using a linear kernel (and the linear Algorithm 3) leads to poorer test performance. Thus, our nonlinear Algorithm 4 can exploit the higher approximating power of the most complex kernel in order to obtain better performance.

Finally, Figure 4.5 contains the plot of the matrix A learned for this experiment using kernel K_{ql}, no irrelevant variables and all 200 tasks simultaneously, as we did in Figure 4.3 for the linear case. Similarly to the linear case, our method learns a matrix A with the desired structure: only the first 7 rows have large entries. Note that the first 7 rows correspond to the monomials of φ, while the remaining 14 rows correspond to the other monomial components of the feature map associated with the kernel.

4.7.2 Real Data

Conjoint Analysis Experiment

Next, we tested our algorithms using a real data set from [Lenk et al., 1996] about people's ratings of products.[6] The data was taken from a survey of 180 persons who rated the likelihood of purchasing one of 20 different personal computers. Here the persons correspond to tasks and the computer models to examples. The input is represented by the following 13 binary attributes: telephone hot line (TE), amount of memory (RAM), screen size (SC), CPU speed (CPU), hard disk (HD), CD-ROM/multimedia (CD), cache (CA), color (CO), availability (AV), warranty (WA), software (SW), guarantee (GU) and price (PR). We also added an input component accounting for the bias term. The output is an integer

[6]We would like to thank Peter Lenk for kindly sharing this data set with us.

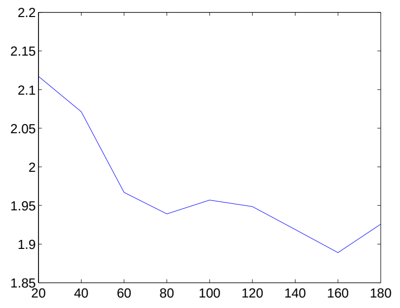


Figure 4.6: Conjoint experiment with computer survey data: average root mean square error vs. number of tasks.

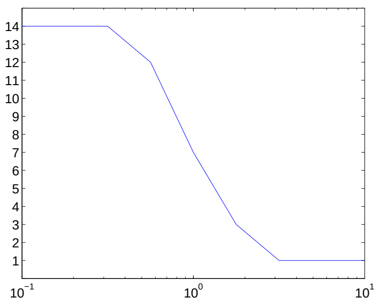


Figure 4.7: Conjoint experiment with computer survey data: number of features learned (with 180 tasks) versus the regularization parameter γ.

rating on the scale $0 - 10$. As in one of the cases in [Lenk et al., 1996], for this experiment we used the first 8 examples per task as the training data and the last 4 examples per task as the test data. We measured the root mean square error of the predicted from the actual ratings for the test data, averaged across the people.

We show results for the linear Algorithm 3 in Figure 4.6. In agreement with the simulations results above and past empirical and theoretical evidence – see Section 4.1 – the performance of Algorithm 3 improves as the number of tasks increases. It also performs better (for all 180 tasks) – test error is 1.93 – than independent ridge regressions, whose test error is equal to 3.88. Moreover, as shown in Figure 4.7, the number of features learned decreases as the regularization parameter γ increases, as expected.

This data has also been used in [Evgeniou et al., 2007]. One of the empirical findings of [Evgeniou et al., 2007, Lenk et al., 1996], a standard one regarding people's preferences, is that estimation improves when one also shrinks the individual w_t's towards a "mean of the tasks", for example the mean of all the w_t's. Hence, it may be more appropriate for this data set to use the regularization term $\sum_{t \in \mathbb{N}_T} \langle (w_t - w_0), D^+(w_t - w_0) \rangle$ as in [Evgeniou et al., 2007] instead of $\sum_{t \in \mathbb{N}_T} \langle w_t, D^+ w_t \rangle$ which we use here. Indeed, test performance is better with the former than the latter. The results are summarized in Table 4.1. We also note that the hierarchical Bayes method of [Lenk et al., 1996], similar to that of [Bakker and Heskes, 2003], also shrinks the w_t's towards a mean across the tasks. Algorithm 3 performs similarly

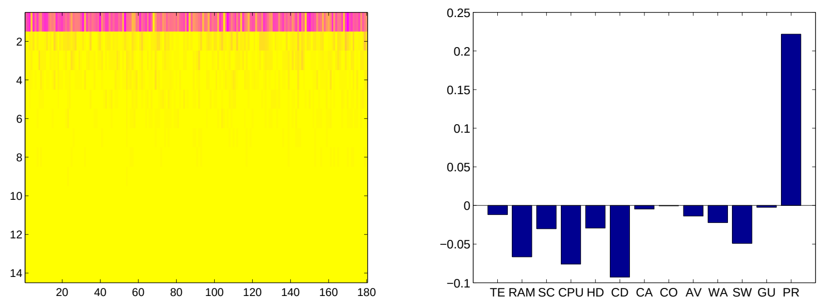


Figure 4.8: Conjoint experiment with computer survey data. Left: matrix A learned, indicating the importance of features learned for all 180 tasks simultaneously. Right: the most important feature learned, common across the 180 people/tasks.

to hierarchical Bayes (despite not shrinking towards a mean of the tasks) but worse than the method of [Evgeniou et al., 2007]. However, we are mainly interested here in learning the common across people/tasks features. We discuss this next.

We investigate which features are important to all consumers as well as how these features weight the 13 computer attributes. We demonstrate the results in the two adjacent plots of Figure 4.8, which were obtained by simultaneously learning all 180 tasks. The plot on the left shows the absolute values of matrix A of feature coefficients learned for this experiment. This matrix has only a few large rows, that is, only a few important features are learned. In addition, the coefficients in each of these rows do not vary significantly across tasks, which means that the learned feature representation is shared across the tasks. The plot on the right shows the weight of each input variable in the most important feature. This feature seems to weight the technical characteristics of a computer (RAM, CPU and CD-ROM) against its price. Note that this is different from selecting the most important variables. In particular, in this case the relative "weights" of each of the 4 variables used in this feature (RAM, CPU, CD-ROM and price) have similar values across all tasks/people.

We also tested our multi-task variable *selection* method, which constrains matrix D in Algorithm 3 to be diagonal. This method led to inferior performance. Specifically, for $T = 180$, multi-task variable selection had test error equal to 2.01, which is worse than the 1.93 error achieved with multi-task feature learning. This supports the argument that "good" features should combine multiple attributes in this problem. Finally, we tested Algorithm 4 with a Gaussian kernel, achieving a slight improvement in performance – see Table 4.1. By considering radial kernels of the form $K(x, x') = e^{-\omega\|x-x'\|^2}$ and selecting ω through cross-validation, we obtained a test error of 1.85 for all 180 tasks. However, interpreting the features learned is more complicated in this case, because of the infinite dimensionality of the feature map for the Gaussian kernel.

Finally, we present preliminary results on *transfer learning*. We trained our method on 150 randomly selected tasks and then used the learned structure matrix D for training 30 ridge regressions on the remaining tasks. We obtained an RMSE of 1.98 on these 30 "new" tasks, which is not much worse

Table 4.1: Comparison of different methods for the computer survey data. MTL-FEAT is the method developed in this thesis.

Method	RMSE
Independent	3.88
Hierarchical Bayes [Lenk et al., 1996]	1.90
RR-Het [Evgeniou et al., 2007]	1.79
MTL-FEAT (linear kernel)	1.93
MTL-FEAT (Gaussian kernel)	1.85
MTL-FEAT (variable selection)	2.01

than an RMSE of 1.88 on the 150 tasks. In comparison, when using the raw data ($D = \frac{I}{d}$) on the 30 tasks we obtained an RMSE of 3.83.

School Data

We have also tested our algorithms on the data from the Inner London Education Authority[7]. This data set has been used in previous work on multitask learning, for example in [Bakker and Heskes, 2003, Evgeniou et al., 2005, Goldstein, 1991]. It consists of examination scores of 15362 students from 139 secondary schools in London during the years 1985, 1986 and 1987. Thus, there are 139 tasks, corresponding to predicting student performance in each school. The input consists of the year of the examination (YR), 4 school-specific and 3 student-specific attributes. Attributes which are constant in each school in a certain year are: percentage of students eligible for free school meals, percentage of students in VR band one (highest band in a verbal reasoning test), school gender (S.GN.) and school denomination (S.DN.). Student-specific attributes are: gender (GEN), VR band (can take the values 1,2 or 3) and ethnic group (ETH). Following [Evgeniou et al., 2005], we replaced categorical attributes (that is, all attributes which are not percentages) with one binary variable for each possible attribute value. In total, we obtained 27 attributes. We also found that results were similar with and without a bias term.

We generated the training and test sets by 10 random splits of the data, so that 75% of the examples from each school (task) belong to the training set and 25% to the test set. We note that the number of examples (students) differs from task to task (school). On average, the training set includes about 80 students per school and the test set about 30 students per school. To account for different school populations, we computed the cross-validation error within each task and then normalized according to school population. The overall mean squared test error was computed by normalizing for each school in a similar way. In order to compare with previous work on this data set, we used the measure of

[7]Available at `http://www.mlwin.com/intro/datasets.html`.

Table 4.2: Comparison of different methods for the school data.

Method	Explained variance
Aggregate	$22.7 \pm 1.3\%$
Independent	$23.8 \pm 2.1\%$
MTL-FEAT (variable selection)	$24.8 \pm 2.0\%$
MTL-FEAT (linear kernel)	$26.7 \pm 2.0\%$
Bayesian MTL [Bakker and Heskes, 2003]	$29.5 \pm 0.4\%$

percentage explained variance from [Bakker and Heskes, 2003]. Explained variance is defined as one minus the mean squared test error over the total variance of the data (computed within each task) and indicates the percentage of variance explained by the prediction model.

The results for this experiment are shown in Table 4.2. The "independent" result is the one obtained by training 139 ridge regressions on each task separately (this also means learning the regularization parameters independently). The "aggregate" result is the one obtained by training one ridge regression on the whole data, as though all students belonged to the same school. A first observation is that training independently does at least as well as aggregate training. This is reinforced when computing the across-tasks standard deviation of explained variance, which is 30% for independent and 26% for aggregate learning. Therefore, there is high variance across the tasks and it seems that they are not concentrated around one prototype task.

Results on this data set have also been obtained in [Bakker and Heskes, 2003] using a hierarchical Bayesian multi-task method and in [Evgeniou et al., 2005] using a multi-task regularization method with $\langle w_t - \frac{1}{T}\sum_{s\in\mathbb{N}_T} w_s, w_t - \frac{1}{T}\sum_{s\in\mathbb{N}_T} w_s\rangle$ in the regularizer (unlike our method, no matrix D was included and estimated). It is difficult to compare with these results because of the more elaborate objective functions used there. Moreover, the data splits used in [Bakker and Heskes, 2003] are not available and may affect the result because of the high variance across the tasks. We report however the best result from [Bakker and Heskes, 2003] as an indication.

From the table we see that our MTL-FEAT algorithm improves upon both independent and aggregate single task learning. Our result is also comparable but not as good as that of [Bakker and Heskes, 2003], even though our regularizer is much simpler. We also found that variable selection performs worse than feature learning and not clearly better than independent learning.

This data set seems well-suited to the approach we have proposed, as one may expect the learning tasks to be very related without being the same – as also discussed in [Bakker and Heskes, 2003, Evgeniou et al., 2005] – in the sense assumed in this chapter. Indeed, one may expect that academic achievement should be influenced by the same variables across schools, if we exclude statistical variation of the

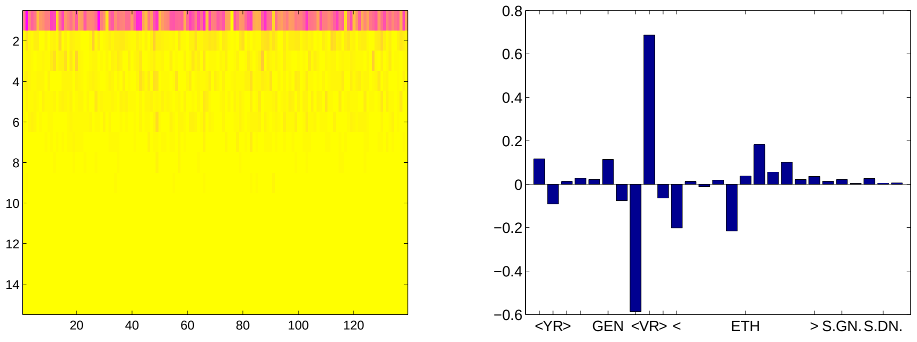


Figure 4.9: School data. Left: matrix A learned for the school data set using a linear kernel. For clarity, only the 15 most important learned features/rows are shown. Right: The most important feature learned, common across all 139 schools/tasks.

student population within each school. This is confirmed in Figure 4.9, where the learned coefficients and the most important feature are shown. As expected, the predicted examination score depends very strongly on the student's VR band. The other variables are much less significant. Ethnic background (primarily British-born, Carribean and Indian) and gender have the next largest influence. What is most striking perhaps is that none of the school-specific attributes has any noticeable significance.

The effects of the number of tasks on the test performance and of the regularization parameter γ on the number of features learned are similar to those for the conjoint and synthetic data: as the number of tasks increases, test performance improves and as γ increases sparsity increases. These plots are similar to Figures 4.6 and 4.7 and are not shown for brevity.

Finally, we performed a transfer learning experiment like that for the computer survey data. We first trained on a random subset of 110 schools and then transferred D to the remaining 29 schools. We obtained an explained variance of 19.2% on the new tasks. This was worse than the explained variance of 24.8% on the 110 tasks but still better than the explained variance of 13.9% with the raw representation.

Dermatology Data

Finally, we show a real-data experiment where it seems (as these are real data, we cannot know for sure whether this is the case indeed) that the tasks are unrelated, in the way that we view task relatedness through common features. In this case, our methods find features which are different across the tasks, and do not improve or decrease performance relative to learning each task independently.

We used the UCI dermatology data set[8] as in [Jebara, 2004]. The problem is a multi-class one, namely to diagnose one of six dermatological diseases based on 33 clinical and histopathological attributes (and an additional bias component). As in the aforementioned paper, we obtained a multi-task problem from the six binary classification tasks. We divided the data set into 10 random splits of 200 training and 166 testing points and measured the average test error across these splits.

We report the misclassification test error in Table 4.3. Algorithm 3 gives similar performance to

[8]Available at `http://www.ics.uci.edu/mlearn/MLSummary.html`.

Table 4.3: Performance of the algorithms for the dermatology data.

Method	Misclassifications
Independent (linear)	16.5 ± 4.0
MTL-FEAT (linear)	16.5 ± 2.6
Independent (Gaussian)	9.8 ± 3.1
MTL-FEAT (Gaussian)	9.5 ± 3.0

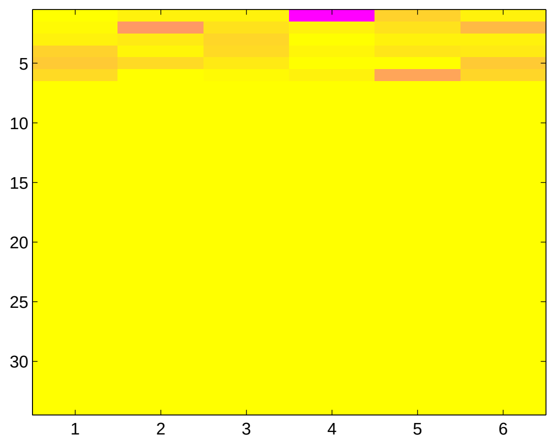


Figure 4.10: Dermatology data. Feature coefficients matrix A learned, using a linear kernel.

that obtained in [Jebara, 2004] with joint feature *selection* and linear SVM classifiers. However, similar performance is also obtained by training 6 independent classifiers. The test error decreased when we ran Algorithm 4 with a single-parameter Gaussian kernel, but it is again similar to that obtained by training 6 independent classifiers with a Gaussian kernel. Hence one may conjecture that these tasks are weakly related to each other or unrelated in the way we have defined.

To further explore this point, we show the matrix A learned by Algorithm 3 in Figure 4.10. This figure indicates that different tasks (diseases) are explained by different features. These results reinforce the conjecture that these tasks may be independent. They indicate that in such a case our methods do not "hurt" performance by simultaneously learning all tasks. In other words, in this problem our algorithms did learn a "sparse common representation" but did not – and probably should not – force each feature learned to be equally important across the tasks.

4.8 Generalization to Matrix Concave Functions

4.8.1 Modeling Tasks' Structure

Our goal in the previous sections has been to learn task parameters $w_1, \dots, w_T$, as well as common features/structure underlying the tasks, from data examples. This structure across tasks is summarized by a positive definite matrix D which is linked to the covariance matrix of the tasks, $WW^\top$, through equation (4.4.2). Thus, a natural generalization of this framework is to obtain D as some *spectral* function of the covariance.

For this purpose, we now consider the regularization functional $\mathrm{Reg} : \mathbb{R}^{d\times T} \times \mathbf{S}^d_{++} \to \mathbb{R}$,

$$\mathrm{Reg}(W, D) := \mathrm{Err}(W) + \gamma\, \mathrm{Penalty}(W, D)\,, \tag{4.8.1}$$

where the regularizer has the form

$$\mathrm{Penalty}(W, D) = \mathrm{trace}(W^\top F(D) W) = \sum_{t\in\mathbb{N}_T} w_t^\top F(D) w_t\,. \tag{4.8.2}$$

Here, $F : \mathbf{S}^d_{++} \to \mathbf{S}^d_{++}$ is a prescribed spectral matrix function (see Section 2.3). This is to say that F is induced by applying a real function $f : \mathbb{R}_{++} \to \mathbb{R}_{++}$ to the eigenvalues of its argument. Thus, for every $D \in \mathbf{S}^d_{++}$ we write $D = U\Lambda U^\top$, where $U \in \mathbf{O}^d$, $\Lambda = \mathrm{Diag}(\lambda_j)_{j\in\mathbb{N}_d}$, and define

$$F(D) = U\,\mathrm{Diag}\left(f(\lambda_j)\right)_{j\in\mathbb{N}_d} U^\top\,.$$

As before, the error term Err in (4.8.1) may be any bounded from below and convex function evaluated at the values $w_t^\top x_{ti}$, $t \in \mathbb{N}_T$, $i \in \mathbb{N}_m$. Typically, it will be the average error on the tasks, namely, $\mathrm{Err}(W) = \sum_{t\in\mathbb{N}_T}\sum_{i\in\mathbb{N}_m} L(y_{ti}, w_t^\top x_{ti})$ and L is a prescribed loss function (such as quadratic, SVM, logistic etc.).

Minimization of the function Reg allows us to learn the tasks and at the same time a good representation for them which is summarized by the eigenvectors and eigenvalues of the matrix D. Different choices of the function f reflect different properties which we would like the tasks to share. In the special case that f is a constant, the tasks are totally independent and the regularizer (4.8.2) is a sum of T independent L_2 regularizers. In the special case $f(\lambda) = \lambda^{-1}$, we obtain problem (4.3.6) which has been the subject of discussion in the previous sections. More generally, functions of the form $f(\lambda) = \lambda^{-\alpha}, \alpha \geq 0$, allow for combining shared features and task-specific features to some degree tuned by the exponent α.

Thus, we propose to solve the minimization problem

$$\inf\left\{\mathrm{Reg}(W, D) : W \in \mathbb{R}^{d\times T}, D \in \mathbf{S}^d_{++}, \mathrm{trace}\, D \leq 1\right\} \tag{4.8.3}$$

for appropriate prescribed functions f. By Theorem 52, this formulation is convex if and only if $\frac{1}{f}$ is *matrix concave* of order d. We may also consider non-convex regularizers, like the ones inducing Schatten L_p prenorms which we discuss in the next section.

Since the first term in (4.8.1) is independent of D, we can first optimize the second term with respect to D. That is, we can compute the infimum

$$\Omega_f(W) := \inf\left\{\mathrm{trace}(W^\top F(D) W) : D \in \mathbf{S}^d_{++}, \mathrm{trace}\, D \leq 1\right\}\,.$$

Moreover, this can be reduced to the vector problem

$$\Omega_f(W) = \inf \left\{ \sum_{i \in \mathbb{N}_d} f(\delta_i)\sigma_i^2 : \delta_i > 0 \ \forall i \in \mathbb{N}_d, \sum_{i \in \mathbb{N}_d} \delta_i \leq 1 \right\}, \tag{4.8.4}$$

where $\{\sigma_i : i \in \mathbb{N}_d\}$ are the singular values of matrix W, by using the following lemma.

Lemma 74 *Let $F : \mathbf{S}^d \to \mathbf{S}^d$ be a spectral function, $B \in \mathbf{S}^d$ and $\{\beta_i : i \in \mathbb{N}_d\}$ the eigenvalues of B. Then*

$$\inf\{\mathrm{trace}(F(D)B) : D \in \mathbf{S}^d_{++}, \mathrm{trace}\, D \leq 1\} = \inf \left\{ \sum_{i \in \mathbb{N}_d} f(\delta_i)\beta_i \ : \ \delta_i > 0 \ \forall i \in \mathbb{N}_d, \sum_{i \in \mathbb{N}_d} \delta_i \leq 1 \right\}.$$

Moreover, for the infimum on the left to be attained, $F(D)$ has to share a set of eigenvectors with B so that the corresponding eigenvalues are in the reverse order as the β_i.

PROOF. We use an inequality of Von Neumann (Lemma 53) to obtain, for all $X, Y \in \mathbf{S}^d$, that

$$\mathrm{trace}(XY) \geq \sum_{i \in \mathbb{N}_d} \lambda_i \mu_i$$

where $\{\lambda_i : i \in \mathbb{N}_d\}$ and $\{\mu_i : i \in \mathbb{N}_d\}$ are the eigenvalues of X and Y in nonincreasing and nondecreasing order, respectively. The equality is attained whenever $X = U\mathrm{Diag}(\lambda)U^\top, Y = U\mathrm{Diag}(\mu)U^\top$ for some $U \in \mathbf{O}^d$. Applying this inequality for $X = F(D), Y = B$ and denoting $f(\delta_i) = \lambda_i, \forall i \in \mathbb{N}_d$, the result follows. ∎

Thus, Ω_f is a function of the singular values of W only. A first observation is that, even though the infimum in (4.8.4) above is not attained in general, the optimization problem in W obtained from (4.8.3) after partial minimization over D admits a minimizer. The reason is that Ω_f is bounded from below and grows at infinity.

Secondly, the optimization problem (4.8.4) may or may not have a closed form solution, but in all cases it is a d-dimensional vector problem. Therefore, we can solve problem (4.8.3) by alternately minimizing over D and W with an algorithm similar to Algorithm 3. The only matrix operation required by such an algorithm is singular value decomposition and the rest are relatively easy vector problems (if the error term decomposes across the tasks). The algorithm can also be extended naturally to a reproducing kernel Hilbert space setting, like Algorithm 4.

In the case of $\mathrm{Penalty}(W, D) = \mathrm{trace}(W^\top D^{-1} W)$ studied in detail up to now, it was observed that the trace constraint prevents this quantity from being zero, which would lead to overfitting (see Section 4.3.3). This is also true in the general case, provided that f is bounded away from zero in the interval $(0, 1]$. In particular, in the case that $\frac{1}{f}$ is matrix concave (which corresponds to jointly convex regularizers) this condition is satisfied. It is also satisfied in the case of negative power functions, which give rise to Schatten L_p prenorms (Section 4.8.2). This allows one to be confident that overfitting does not occur with a broad class of choices for f. Finally, generalization error bounds have been derived for the case of Schatten L_p norm regularization by [Maurer, 2006a].

4.8.2 Regularization with Schatten L_p Prenorms

In this section, we focus on the family of negative power functions f and, using Lemma 74, we obtain that function Ω_f in (4.8.4) relates to the Schatten L_p prenorms. [9]

Proposition 75 *Let $B \in \mathbf{S}^d_+$ and $s \in (0,1]$. Then*

$$(\operatorname{trace} B^s)^{\frac{1}{s}} = \inf\left\{\operatorname{trace}(D^{\frac{s-1}{s}} B) : D \in \mathbf{S}^d_{++}, \operatorname{trace} D \leq 1\right\}.$$

Moreover, if $B \in \mathbf{S}^d_{++}$ the infimum is attained and the minimizer is given by $D = \dfrac{B^s}{\operatorname{trace} B^s}$.

PROOF. By Lemma 74, it suffices to show the analogous statement for vectors, namely that

$$\left(\sum_{i\in\mathbb{N}_d} \beta_i^s\right)^{\frac{1}{s}} = \inf\left\{\sum_{i\in\mathbb{N}_d} \delta_i^{\frac{s-1}{s}} \beta_i : \delta_i > 0, i \in \mathbb{N}_d, \sum_{i\in\mathbb{N}_d} \delta_i \leq 1\right\}$$

where $\beta_i \geq 0, \forall i \in \mathbb{N}_d$. To this end, we apply Hölder's inequality with $p = \frac{1}{s}$ and $q = \frac{1}{1-s}$:

$$\sum_{i\in\mathbb{N}_d} \beta_i^s = \sum_{i\in\mathbb{N}_d} \left(\delta_i^{\frac{s-1}{s}} \beta_i\right)^s \delta_i^{1-s} \leq \left(\sum_{i\in\mathbb{N}_d} \delta_i^{\frac{s-1}{s}} \beta_i\right)^s \left(\sum_{i\in\mathbb{N}_d} \delta_i\right)^{1-s} \leq \left(\sum_{i\in\mathbb{N}_d} \delta_i^{\frac{s-1}{s}} \beta_i\right)^s .$$

When $\beta_i > 0, \forall i \in \mathbb{N}_d$, the equality is attained for $\delta_i = \dfrac{\beta_i^s}{\sum_{j\in\mathbb{N}_d} \beta_j^s}$, $\forall i \in \mathbb{N}_d$. To show that the inequality is sharp in all other cases, we replace β_i by $\beta_{i,\varepsilon} := \beta_i + \varepsilon$, $\forall i \in \mathbb{N}_d, \varepsilon > 0$, define $\delta_{i,\varepsilon} := \dfrac{\beta_{i,\varepsilon}^s}{\sum_{j\in\mathbb{N}_d} \beta_{j,\varepsilon}^s}$ and take the limit of the right hand side of the inequality as $\varepsilon \to 0$. ■

The above result implies that the regularization problem (4.8.3) is conceptually equivalent to regularization with a Schatten L_p prenorm of W, when the coupling function f takes the form $f(x) = x^{1-\frac{2}{p}}$ with $p \in (0, 2]$ (obtained by letting $p = 2s$ in the proposition). The Schatten L_p prenorm is the L_p prenorm of the singular values of a matrix. Such regularizers are convex (strictly convex) if and only if $p \geq 1$ ($p > 1$). In particular, our formulation in Section 4.3 as well as trace norm regularization [Abernethy et al., 2006, Srebro et al., 2005] correspond to the case $p = 1$.

4.8.3 Experiments

In this section, we first report a comparison of the computational cost between the alternating minimization algorithm and the gradient descent algorithm. We then study how performance varies for different L_p regularizers on the *computer survey data* and the *school data set* of Section 4.7.

In the first experiment, we study the computational cost of the alternating minimization algorithm against the gradient descent algorithm, both implemented in Matlab, for the Schatten $L_{1.5}$ norm. The left plot in Figure 4.11 shows the value of the objective function (4.8.1) versus the number of iterations, on the computer survey data. The curves for different learning rates η are shown, whereas for rates greater than 0.05 gradient descent diverges. The curve for the alternating Algorithm 3 with $\varepsilon = 10^{-16}$ is also shown. We further note that for both data sets our algorithm typically needed less than 30 iterations

[9]A prenorm is a function satisfying the properties of a norm except the triangle inequality – see [Horn and Johnson, 1985, Sec. 5.4].

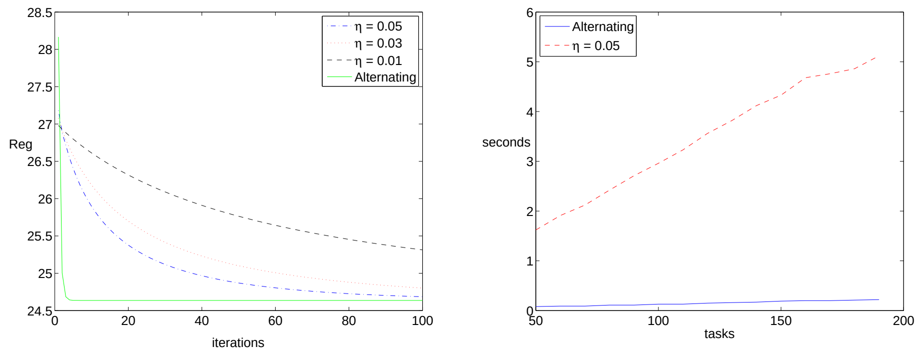


Figure 4.11: Comparison between the alternating algorithm and the gradient descent algorithm.

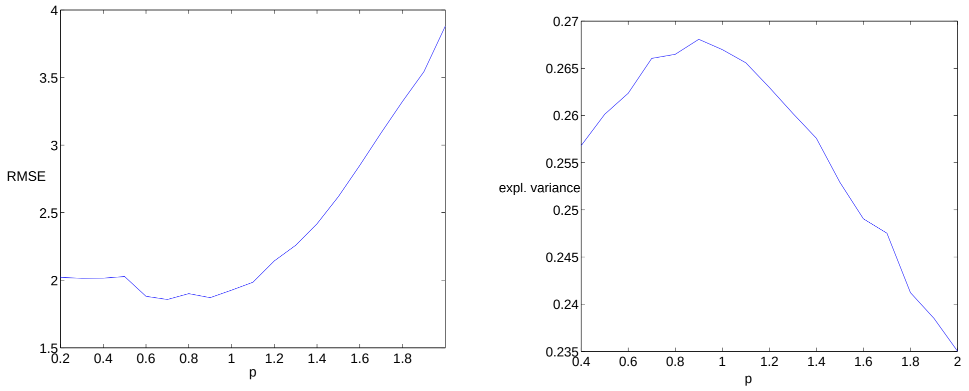


Figure 4.12: Performance versus p for the computer survey data (left) and the school data (right).

to converge. The right plot depicts the CPU time (in seconds) needed to reach a value of the objective function which is less than 10^{-5} away from the minimum, versus the number of tasks. It is clear that our algorithm is at least an order of magnitude faster than gradient descent with the optimal learning rate and scales better with the number of tasks. We note that the computational cost of our method is mainly due to the T ridge regressions in the supervised step (learning W) and the singular value decomposition in the unsupervised step (learning D). A singular value decomposition is also needed in gradient descent, for computing the gradient of the Schatten L_p norm. We have observed that the cost per iteration is smaller for gradient descent but the number of iterations is at least an order of magnitude larger, leading to the large difference in time cost.

In the second experiment we study the statistical performance of our method as the spectral function changes. Specifically, we choose functions giving rise to Schatten L_p prenorms, as discussed in Section 4.8.2. The results, shown in Figure 4.12, indicate that the trace norm is the best norm on these data sets. However, on the computer survey data a value of p less than one gives the best result overall. From this we speculate that our method can even approximate well the solutions of certain non-convex problems. In contrast, on the school data the trace norm gives the best result.

4.9 Connection between Multi-Task Learning and Learning the Kernel

In this section, we bring together the multi-task framework built in this chapter and the framework for learning the kernel from Chapter 3. To this end, we define the kernel $K_f(D)(x,z) = x^\top (F(D))^{-1} z$, $x, z \in \mathbb{R}^d$, the set of kernels $\mathcal{K}_f = \{K_f(D) : D \in \mathbf{S}^d_{++}, \operatorname{trace} D \leq 1\}$ and, for every kernel K, the task kernel matrices $K_t = (K(x_{ti}, x_{tj}) : i, j \in \mathbb{N}_m)$, $t \in \mathbb{N}_T$. It is easy to prove, using Weyl's monotonicity theorem [Horn and Johnson, 1985, Sec. 4.3] and [Bhatia, 1997, Thm. V.2.5], that the set $\mathcal{K}_f$ is convex if and only if $\frac{1}{f}$ is matrix concave. By the Representer Theorem (see Section 2.2.2), problem (4.8.3) is equivalent to

$$\min\left\{\sum_{t\in\mathbb{N}_T}\left(\sum_{i\in\mathbb{N}_m} L(y_{ti}, (K_t c_t)_i) + \gamma\, c_t^\top K_t c_t\right) : c_t \in \mathbb{R}^m, K \in \mathcal{K}_f\right\}$$

It is apparent that this objective function is not jointly convex in c_t and K. However, minimizing each t-term over the vector c_t gives a convex function of K.

Proposition 76 *Let $\mathscr{K}$ be the set of all reproducing kernels on $\mathbb{R}^d$. If $L(y,\cdot)$ is convex for any $y \in \mathbb{R}$ then the function $E_t : \mathscr{K} \to \mathbb{R}_+$ defined for every $K \in \mathscr{K}$ as*

$$E_t(K) = \min\left\{\sum_{i\in\mathbb{N}_m} L(y_{ti}, (K_t c)_i) + \gamma\, c^\top K_t c : c \in \mathbb{R}^m\right\}$$

is convex.

PROOF. Without loss of generality, we can assume that K_t is invertible for every $t \in \mathbb{N}_T$. For every $a \in \mathbb{R}^m$ and $K \in \mathscr{K}$, we define the function $G_t(a, K) = \sum_{i\in\mathbb{N}_m} L(y_{ti}, a_i) + \gamma\, a^\top K_t^{-1} a$, which is jointly convex by Theorem 52. Clearly, $E_t(K) = \min\{G_t(a,K) : a \in \mathbb{R}^m\}$. Recalling that the partial minimum of a jointly convex function is convex (Theorem 15), we obtain the convexity of E_t. ■

The fact that function E_t is convex has already been proven as Corollary 56 using a minimax theorem and the convex conjugate. Here, we were able to simplify the proof of this result by appealing to the joint convexity property stated in Theorem 52.

Some further insight about the connection between multi-task learning and learning the kernel in a convex set can be gained by comparing our formulation (4.3.1)-(4.3.3) to the feature map interpretation of (3.6.1). To be able to do this comparison, we need to determine the appropriate *multi-task kernels*, in the sense defined in [Evgeniou et al., 2005]. These act on the joint input-task space and, in this case, induce block diagonal Gram matrices:

$$K_i((x,t),(x',t')) = \delta_{t,t'}\langle h_i(x), h_i(x')\rangle \qquad \forall x, x' \in \mathbb{R}^d, t, t' \in \mathbb{N}_T, i \in \mathbb{N}_d.$$

The corresponding feature maps in the input-task space are $\Psi_i : \mathbb{R}^d \times \mathbb{N}_T \to \mathbb{R}^T$ with

$$\Psi_i(x,t) = \begin{array}{cc} \begin{pmatrix} 0 \\ \vdots \\ h_i(x) \\ \vdots \\ 0 \end{pmatrix} & \begin{array}{c} \\ \\ t \\ \\ \\ \end{array} \end{array} \qquad \forall x \in \mathbb{R}^d, t \in \mathbb{N}_T, i \in \mathbb{N}_d.$$

Thus, learning a number of tasks together is equivalent to learning a number of features together. The feature maps $\Psi_i, i \in \mathbb{N}_d$, in the multi-task setup correspond to the feature maps $\Phi_\ell, \ell \in \mathbb{N}_n$, in kernel learning. Visually, the analogy is as follows

$$\begin{array}{cc} & k \\ \ell & \begin{pmatrix} \ddots & \vdots & \iddots \\ \cdots & (w_\ell)_k & \cdots \\ \iddots & \vdots & \ddots \end{pmatrix} \end{array} \quad \longleftrightarrow \quad \begin{array}{cc} & t \\ i & \begin{pmatrix} \ddots & \vdots & \iddots \\ \cdots & a_{it} & \cdots \\ \iddots & \vdots & \ddots \end{pmatrix} \end{array},$$

where $\ell \in \mathbb{N}_n, k \in \mathbb{N}_s, i \in \mathbb{N}_d, t \in \mathbb{N}_T$ index the feature map Φ_ℓ, the feature within Φ_ℓ, the feature map Ψ_i and the task, respectively. The property that all tasks share a small set of feature maps corresponds to the property that a small set of kernels are involved in the optimal combination. These properties are induced by the appearance of a $(2,1)$-norm in both (4.3.1) and (3.6.1).

Another conclusion obtained is that the variable selection method described in this chapter can be seen as learning convex combinations of the multi-task kernels K_i. Regarding the multi-task feature learning method, which has been the main focus of this chapter, it corresponds to learning a single linear kernel (described by matrix D) whose trace is bounded. The orthogonality assumption for the features is automatically implied since any linear kernel can be written as a convex combination of linear kernels whose matrices have rank one and are mutually orthogonal.

Chapter 5

Conclusion

In Chapter 3, we have studied the problem of learning a kernel which minimizes a convex error functional over the convex hull of prescribed basic kernels. A substantial innovation of our approach is that it avoids deciding on a finite subset of basic kernels in advance. The main contribution of the chapter is a general analysis of this problem when the basic kernels are continuously parameterized by a compact set. In particular, we have shown that there always exists an optimal kernel which is a finite combination of the basic kernels and presented a greedy algorithm for learning it. This algorithm is simple to implement and is based on standard kernel methods such as SVMs. We have also proven the convergence of this algorithm to the optimal kernel. To tackle the main computational problem involved, we have used a recently developed technique for global optimization.

We have provided experimental results where the basic kernels are Gaussians with constrained diagonal covariance matrices. These computational results indicate the advantage of working with a continuous parameterization as opposed to an a priori choice of a finite number of basic kernels. The method is robust because the choice of the optimal kernel is insensitive to the range of the continuous parameters. These preliminary findings indicate that the algorithm typically converges in a small number of iterations to a kernel with a competitive statistical performance.

There remain several issues about this framework that need to be addressed in the future. One has to do with generalization error bounds for important classes of basic kernels and for these the methodology followed in [Srebro and Ben-David, 2006] could be used. Other future work could apply new optimization techniques to our greedy algorithm, especially to the DC problem involved. There are also implementation issues with large data sets which may be very important in the context of some applications. We believe that our approach is a good starting point for learning large data sets with combinations of parameterized kernels. In such practical scenarios, a variety of heuristics could be used to make implementations more efficient.

Another important question is under which conditions the optimization problem (3.3.1) is tractable. As already observed, this is true in the case of a finite number of basic kernels, linear and polynomial kernels, certain spectral functions of the Gram matrix (see Sections 3.2.5, 3.4.1), but in general it is a DC problem. More investigation is required to determine whether any other useful parameterizations lead to tractable problems.

Other future directions are towards extending the framework we have proposed by minimizing other criteria instead of the combination of training error and regularizer. Some candidates are the bounds on the generalization error mentioned in [Chapelle et al., 2002]. Another one is the cross-validation error whose optimization is studied, for example, in [Bennett et al., 2006]. One would expect that some of these measures are statistically more robust than the regularization functional. However, the convex property (Corollary 56) we have shown does not hold in general. Thus a more thorough theoretical and algorithmic study is needed, for which our treatment may provide a good starting basis. The performance of different functionals should be assessed using error bounds and empirical trials and should be balanced against the computational cost.

In the second part of this thesis (Chapter 4), we have presented an algorithm which learns common sparse representations across a pool of related tasks. These representations are assumed to be orthonormal functions in a reproducing kernel Hilbert space. Our method is based on a regularization problem with a novel type of regularizer, which is a mixed $(2, 1)$-norm.

We showed that this problem, which is non-convex, can be reformulated as a convex optimization problem. This result makes it possible to compute the optimal solutions using a simple alternating minimization algorithm, whose convergence we have proven. For the case of a high-dimensional feature map, we have developed a variation of the algorithm which uses kernel functions. We have also proposed a variation of the first algorithm for solving the problem of multi-task feature *selection* with a linear feature map.

We have reported experiments with our method on synthetic and real data. They indicate that our algorithms learn sparse feature representations common to all the tasks whenever this helps improve performance. In such cases, the performance obtained is better than that of training the tasks independently. Moreover, when applying our algorithm on a data set with weak task interdependence, performance does not deteriorate and the learned representation reflects the lack of task relatedness. As indicated in one such experiment, the estimated matrix A can be used to visualize task relatedness. Finally, our experiments have shown that *learning* orthogonal features improves on just *selecting* input variables.

To our knowledge, our approach provides the first convex optimization formulation for multi-task feature learning. Although convex optimization methods have been derived for the simpler problem of feature selection [Jebara, 2004], prior work on multi-task feature learning has been based on more complex optimization problems which are not convex [Ando and Zhang, 2005, Baxter, 2000, Caruana, 1997] and, so, these methods are not guaranteed to converge to a global optimum. In particular, in [Baxter, 2000, Caruana, 1997] different neural networks with one or more hidden layers are trained for each task and they all share the same hidden weights. These common hidden layers and weights act as an internal representation (like the features in our formulation) which is shared by all the tasks.

Our algorithm also shares some similarities with recent work in [Ando and Zhang, 2005] where they also alternately update the task parameters and the features. Two main differences are that their formulation is not convex and that, in our formulation, the number of learned features is not fixed in advance but is controlled by a regularization parameter. Other relevant work is [Raina et al., 2006], which also proposes an alternate update algorithm, inspired from a maximum a posteriori perspective.

As noted in Section 4.5, our work relates to [Abernethy et al., 2006, Srebro et al., 2005], which investigate regularization with the trace norm in the context of collaborative filtering. In fact, the sparsity assumption which we have made in our work, starting with the $(2, 1)$-norm, connects to the low rank assumption in those works. Hence, it may be possible that our alternating algorithm, or some variation of it, could be used to solve the optimization problems of [Abernethy et al., 2006, Srebro et al., 2005, etc.]. An additional benefit, as the experiments in Section 4.8.3 demonstrate, is that our alternating algorithm converges faster than gradient descent. Our work also relates to partial least squares methods (see, for example, [Bennett and Embrechts, 2003, Wold et al., 1984]). Although these methods have proved useful in practical applications, they require that the same input examples are shared by all the tasks. On the contrary, our approach does not rely on this assumption.

Our work may be extended in different directions. First, it would be interesting to carry out a learning theory analysis of the algorithms presented. Results in [Caponnetto and De Vito, 2006, Maurer, 2006b] may be useful for this purpose. Another interesting question is to study how the solution of our algorithms depends on the regularization parameter and investigate conditions which ensure that the number of features learned decreases with the degree of regularization, as we have experimentally observed in this thesis. Results in [Micchelli and Pinkus, 1994] may be useful for this purpose.

Some experimental and theoretical investigation could be devoted to formulations with spectral functions other than the L_p prenorms considered here. There are some candidates, like the relative entropy of W and D, which make intuitive sense and satisfy the matrix concavity condition of Theorem 52. The choice of a specific spectral regularizer should also depend on the tasks at hand, so that certain problems may call for assumptions other than L_1-type sparsity.

Another promising research direction is to explore whether different assumptions about the features (other than the orthogonality one which we have made) can still lead to different convex optimization methods. More specifically, it would be interesting to study whether non-convex models for learning structures across the tasks, like those in [Zhang et al., 2006] where ICA type features are learned, or hierarchical features models like in [Torralba et al., 2004], can be reformulated in a framework similar to ours. Yet another variation could be to introduce additional constraints which have some practical interpretation. For example, we would like to study the problem under the constraint that the task coefficients are nonnegative. This problem is motivated by many situations in which we want to *mix* features, not weigh them *against each other*. It also relates to previous work on nonnegative matrix factorization, such as [Lee and Seung, 2001].

It would also be interesting to explore whether our formulation can be extended to *transformations parameterized by the task* – which can be orthogonal, as in our presentation, or other. This assumption has been advanced by [Ben-David and Schuller, 2003] and covers a wider class of problems than the ones which we have studied. It is a realistic assumption in many situations where there exist underlying invariances and the data lies on a low-dimensional manifold.

An important practical issue in the development of any regularization-based algorithm is the choice of the regularization parameter γ. The regularization parameter plays an important role in our approach, since it affects the sparsity of the common-across-tasks feature representation. Thus, it should not be fixed a priori but should be determined from the data. The most standard technique, which we have also used in our experiments, is cross-validation. This technique is computationally taxing, however, and a few other approaches, such as the *regularization path* [Hastie et al., 2004], have been proposed. Therefore, a future theoretical question in multi–task feature learning would be to describe the statistical behavior of the obtained solution as a function of the regularization parameter.

Finally, at the end of Chapter 4, we have shown a connection between the two formulations for learning in a convex set of kernels and learning common features for multiple tasks. There is a mapping between the basic feature maps in the former and the multi-task feature maps in the latter. Clearly, this analogy may inspire one to transfer results and methods from one problem to the other. For example, it could lead to more complex regularization-based formulations for multi-task learning. Also, it may be possible to translate results about the generalization error from one problem to the other.

Appendix A

Proof of Lemma 55

PROOF. Theorem 43 applies. Indeed, we let $f(\alpha, z) = \langle K_{\mathbf{x}}\alpha, z\rangle - Q^*(z) + \gamma\langle \alpha, K_{\mathbf{x}}\alpha\rangle$, $\mathcal{C} = \mathbb{R}^m$, $\mathcal{Z} = \operatorname{dom} Q^*$ and $z_0 = 0$. Then, $\mathcal{Z}$ is convex and, for any $a \in \mathbb{R}$, the set $\{\alpha : \alpha \in \mathcal{C}, f(\alpha, z_0) \leq a\}$ is compact. Therefore, all the hypotheses of Theorem 43 hold. Consequently, using (3.2.7) in (3.2.4) we have that

$$E_\gamma(K) = \sup\{\min\{\langle K_{\mathbf{x}}\alpha, z\rangle - Q^*(z) + \gamma\langle \alpha, K_{\mathbf{x}}\alpha\rangle : \alpha \in \mathbb{R}^m\} : z \in \operatorname{dom} Q^*\}.$$

For each $z \in \operatorname{dom} Q^*$, the minimum over α satisfies the equation $K_{\mathbf{x}}z + 2\gamma K_{\mathbf{x}}\alpha = 0$, implying that

$$\min\{\langle K_{\mathbf{x}}\alpha, z\rangle - Q^*(z) + \gamma\langle \alpha, K_{\mathbf{x}}\alpha\rangle : \alpha \in \mathbb{R}^m\} = -\frac{\langle z, K_{\mathbf{x}}z\rangle}{4\gamma} - Q^*(z)$$

and the result follows. The attainment of the supremum follows from closedness of Q^* and the facts that $\liminf_{\|z\|\to\infty} \langle z, K_{\mathbf{x}}z\rangle \to +\infty$ and Q^* is bounded from below. ■

Appendix B

Convergence of Algorithm 1

PROOF. First, from (3.3.2) and the boundedness property $E_\gamma(K) \geq -R(0,K) = -Q^*(0) > -\infty$, it is clear that the sequence $\{E_\gamma(K^{(k)}) : k \in \mathbb{N}\}$ converges to a real number. Moreover, observe that the sequence $\{K_{\mathbf{x}}^{(k)} : k \in \mathbb{N}\}$ is bounded, that $R(c^{(k)}, K^{(k)}) \leq R(0, K^{(k)}) = Q^*(0) < +\infty$ and that Q^* is bounded from below by $-Q(0)$. Therefore, there are convergent subsequences $\{K_{\mathbf{x}}^{(k_\ell)} : \ell \in \mathbb{N}\}$, $\{\langle c^{(k_\ell)}, K_{\mathbf{x}}^{(k_\ell)} c^{(k_\ell)}\rangle : \ell \in \mathbb{N}\}$, $\left\{\frac{c^{(k_\ell)}}{\|c^{(k_\ell)}\|} : \ell \in \mathbb{N}\right\}$, or alternatively $\{c^{(k_\ell)} = 0 : \ell \in \mathbb{N}\}$. In the former case, the sequence $\|c^{(k_\ell)}\|^2 = \frac{\left\langle c^{(k_\ell)}, K_{\mathbf{x}}^{(k_\ell)} c^{(k_\ell)}\right\rangle}{\left\langle \frac{c^{(k_\ell)}}{\|c^{(k_\ell)}\|}, K_{\mathbf{x}}^{(k_\ell)} \frac{c^{(k_\ell)}}{\|c^{(k_\ell)}\|}\right\rangle}$ is also convergent. This shows the existence of a limit point.

Consider now any limiting subsequence $\{(c^{(k_\ell)}, K^{(k_\ell)}) : \ell \in \mathbb{N}\}$ and let $(\bar{c}, \bar{K})$ be its limit. First, we observe that the function

$$M \in \mathbf{S}_{++}^m \mapsto \min\left\{\frac{1}{4\gamma}\langle c, Mc\rangle + Q^*(c) : c \in \mathbb{R}^m\right\}$$

is concave as the minimum of concave functions and hence continuous (by Theorem 22). Thus,

$$R(c^{(k_\ell)}, K^{(k_\ell)}) = -E_\gamma(K^{(k_\ell)}) \to -E_\gamma(\bar{K}) \qquad \text{as } \ell \to \infty.$$

Therefore, $\bar{c}$ minimizes $R(\cdot, \bar{K})$ and the upper inequalities in (3.2.14) are satisfied. To show that $(\bar{c}, \bar{K})$ is a saddle point it suffices to show the lower inequalities. Let us assume the opposite and try to obtain a contradiction. That is, assume that

$$\langle \bar{c}, B_{\mathbf{x}}^* \bar{c}\rangle > \langle \bar{c}, \bar{K}_{\mathbf{x}} \bar{c}\rangle$$

for some $B^* \in \mathcal{B}$. Also, let $\hat{B}^{(k)}, \hat{\lambda}^{(k)}$ be the iterates computed in steps 2 and 3, respectively, of Algorithm 1. Since $\mathcal{B}$ is compact, the corresponding sequences have convergent subsequences. Without loss of generality, assume that

$$\{K_{\mathbf{x}}^{(k_\ell+1)} : \ell \in \mathbb{N}\}, \{\hat{B}_{\mathbf{x}}^{(k_\ell)} : \ell \in \mathbb{N}\}, \{\hat{\lambda}^{(k_\ell)} : \ell \in \mathbb{N}\}$$

are convergent (since there exist simultaneously convergent subsequences). Let $\bar{K}_{\mathbf{x}}^+, \bar{B}_{\mathbf{x}}, \bar{\lambda}$ be their limits, respectively. From continuity and step 4 we obtain that

$$E_\gamma(\bar{\lambda}\bar{B} + (1-\bar{\lambda})\bar{K}) = E_\gamma(\bar{K}^+)\,.$$

We have already observed that $\{E_\gamma(K^{(k)}) : k \in \mathbb{N}\}$ converges, thus

$$E_\gamma(\bar{\lambda}\bar{B} + (1-\bar{\lambda})\bar{K}) = E_\gamma(\bar{K})\,. \tag{B.0.1}$$

Because of the maximization in step 2, we have that

$$\langle c^{(k_\ell)}, \hat{B}_{\mathbf{x}}^{(k_\ell)} c^{(k_\ell)}\rangle \geq \langle c^{(k_\ell)}, B_{\mathbf{x}}^{*} c^{(k_\ell)}\rangle \qquad \forall \ell \in \mathbb{N}$$

and, taking the limits, that

$$\langle \bar{c}, \bar{B}_{\mathbf{x}}\bar{c}\rangle \geq \langle \bar{c}, B_{\mathbf{x}}^{*}\bar{c}\rangle$$

and hence

$$\langle \bar{c}, \bar{B}_{\mathbf{x}}\bar{c}\rangle > \langle \bar{c}, \bar{K}_{\mathbf{x}}\bar{c}\rangle\,.$$

By Lemma 63, there exists $\lambda \in (0,1]$ such that

$$E_\gamma(\lambda\bar{B} + (1-\lambda)\bar{K}) < E_\gamma(\bar{K})\,.$$

On the other hand, by step 3,

$$E_\gamma(\lambda^{(k_\ell)}\hat{B}^{(k_\ell)} + (1-\lambda^{(k_\ell)})K^{(k_\ell)}) \leq E_\gamma(\lambda\hat{B}^{(k_\ell)} + (1-\lambda)K^{(k_\ell)}) \qquad \forall \ell \in \mathbb{N}\,,$$

which yields a contradiction when the limits are taken and (B.0.1) is taken into account. ■

Appendix C

Proof of Equation (4.4.2)

PROOF. Consider a matrix $C \in \mathbf{S}^d_+$. We will compute $\inf\{\operatorname{trace}(D^{-1}C) \,:\, D \in \mathbf{S}^d_{++}, \operatorname{trace}(D) \leq 1\}$. From the Cauchy-Schwarz inequality for the Frobenius norm, we obtain

$$\begin{aligned}
\operatorname{trace}(D^{-1}C) &\geq \operatorname{trace}(D^{-1}C)\operatorname{trace}(D) \\
&= \operatorname{trace}\Big(\big(D^{-\frac{1}{2}}C^{\frac{1}{2}}\big)\big(C^{\frac{1}{2}}D^{-\frac{1}{2}}\big)\Big)\operatorname{trace}\big(D^{\frac{1}{2}}D^{\frac{1}{2}}\big) \\
&\geq \Big(\operatorname{trace}\Big(D^{\frac{1}{2}}\big(C^{\frac{1}{2}}D^{-\frac{1}{2}}\big)\Big)\Big)^2 = \Big(\operatorname{trace} C^{\frac{1}{2}}\Big)^2.
\end{aligned}$$

The equality is attained if and only if $\operatorname{trace}(D) = 1$ and $C^{\frac{1}{2}}D^{-\frac{1}{2}} = aD^{\frac{1}{2}}$ for some $a \in \mathbb{R}$, or equivalently for $D = \dfrac{C^{\frac{1}{2}}}{\operatorname{trace} C^{\frac{1}{2}}}$. ■

Using similar arguments as above, it can be shown that $\min\{\operatorname{trace}(D^{+}C) \,:\, D \in \mathbf{S}^d_+, \operatorname{trace}(D) \leq 1, \operatorname{range}(C) \subseteq \operatorname{range}(D)\}$ also equals $\big(\operatorname{trace} C^{\frac{1}{2}}\big)^2$.

Appendix D

Convergence of Algorithm 3

In this appendix, we present the proofs of Theorems 70 and 71. For this purpose, we substitute equation (4.4.2) in the definition of $\mathcal{R}_\varepsilon$ obtaining the objective function

$$\begin{aligned}\mathcal{S}_\varepsilon(W) &:= \mathcal{R}_\varepsilon(W, D_\varepsilon(W)) \\ &= \sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma \left(\operatorname{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}\right)^2 .\end{aligned}$$

Moreover, we define the following function which formalizes the supervised step of the algorithm,

$$g_\varepsilon(W) := \min\{\mathcal{R}_\varepsilon(V, D_\varepsilon(W)), : V \in \mathbb{R}^{d\times T}\} .$$

Since $\mathcal{S}_\varepsilon(W) = \mathcal{R}_\varepsilon(W, D_\varepsilon(W))$ and $D_\varepsilon(W)$ minimizes $\mathcal{R}_\varepsilon(W, \cdot)$, we obtain that

$$\mathcal{S}_\varepsilon(W^{(n+1)}) \leq g_\varepsilon(W^{(n)}) \leq \mathcal{S}_\varepsilon(W^{(n)}) . \tag{D.0.1}$$

We begin by observing that $\mathcal{S}_\varepsilon$ has a *unique* minimizer. This is a direct consequence of the following proposition.

Proposition 77 *The function $\mathcal{S}_\varepsilon$ is strictly convex for every $\varepsilon > 0$.*

PROOF. It suffices to show that the function

$$W \mapsto \left(\operatorname{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}\right)^2$$

is strictly convex. But this is simply a function of the singular values of W. By [Lewis, 1995, Sec. 3], strict convexity follows directly from strict convexity of the real function $\sigma \mapsto \left(\sum_i \sqrt{\sigma_i^2 + \varepsilon}\right)^2$. This function is strictly convex because it is the square of a positive strictly convex function. ∎

We note that when $\varepsilon = 0$, the function $\mathcal{S}_\varepsilon$ is regularized by the trace norm squared which is not a strictly convex function. Thus, in many cases of interest $\mathcal{S}_0$ may have multiple minimizers. For instance, this is true if the loss function L is not strictly convex, which is the case with SVMs.

Next, we show the following continuity property which underlies the convergence of Algorithm 3.

Lemma 78 *The function g_ε is continuous for every $\varepsilon > 0$.*

PROOF. We first show that the function $G_\varepsilon : \mathbf{S}^d_{++} \to \mathbb{R}$ defined as

$$G_\varepsilon(D) := \min\left\{\mathcal{R}_\varepsilon(V, D) : V \in \mathbb{R}^{d\times T}\right\}$$

is convex. Indeed, G_ε is the minimal value of T separable regularization problems with a common kernel function determined by D and Corollary 56 can be used. Since the domain of this function is open, G_ε is also continuous (see Theorem 22).

In addition, the matrix-valued function $W \mapsto (WW^\top + \varepsilon I_d)^{\frac{1}{2}}$ is continuous. To see this, we recall the fact that the matrix-valued function $Z \in \mathbf{S}^d_+ \mapsto Z^{\frac{1}{2}}$ is continuous.

Combining, we obtain that g_ε is continuous, as the composition of continuous functions. ■

Proof of Theorem 70. By inequality (D.0.1) the sequence $\{\mathcal{S}_\varepsilon(W^{(n)}) : n \in \mathbb{N}\}$ is nonincreasing and, since L is bounded from below, it is bounded. As a consequence, as $n \to \infty$, $\mathcal{S}_\varepsilon(W^{(n)})$ converges to a number, which we denote by $\widetilde{\mathcal{S}_\varepsilon}$. We also deduce that the sequence $\left\{\operatorname{trace}\left(W^{(n)}W^{(n)\top} + \varepsilon I_d\right)^{\frac{1}{2}} : n \in \mathbb{N}\right\}$ is bounded and hence so is the sequence $\{W^{(n)} : n \in \mathbb{N}\}$. Consequently there is a convergent subsequence $\{W^{(n_\ell)} : \ell \in \mathbb{N}\}$, whose limit we denote by $\widetilde{W}$.

Since $\mathcal{S}_\varepsilon(W^{(n_\ell+1)}) \le g_\varepsilon(W^{(n_\ell)}) \le \mathcal{S}_\varepsilon(W^{(n_\ell)})$, $g_\varepsilon(W^{(n_\ell)})$ converges to $\widetilde{\mathcal{S}_\varepsilon}$. Thus, by Lemma 78 and the continuity of $\mathcal{S}_\varepsilon$, $g_\varepsilon(\widetilde{W}) = \mathcal{S}_\varepsilon(\widetilde{W})$. This implies that $\widetilde{W}$ is a minimizer of $\mathcal{R}_\varepsilon(\cdot, D_\varepsilon(\widetilde{W}))$, because $\mathcal{R}_\varepsilon(\widetilde{W}, D_\varepsilon(\widetilde{W})) = \mathcal{S}_\varepsilon(\widetilde{W})$.

Moreover, recall that $D_\varepsilon(\widetilde{W})$ is the minimizer of $\mathcal{R}_\varepsilon(\widetilde{W}, \cdot)$ subject to the constraints in (4.4.1). Since the regularizer in $\mathcal{R}_\varepsilon$ is smooth, any directional derivative of $\mathcal{R}_\varepsilon$ is the sum of its directional derivatives with respect to W and D. Hence, $(\widetilde{W}, D_\varepsilon(\widetilde{W}))$ is the minimizer of $\mathcal{R}_\varepsilon$.

We have shown that any convergent subsequence of $\{W^{(n)} : n \in \mathbb{N}\}$ converges to the minimizer of $\mathcal{R}_\varepsilon$. Since the sequence $\{W^{(n)} : n \in \mathbb{N}\}$ is bounded it follows that it converges to the minimizer as a whole. ■

Proof of Theorem 71. Let $\left\{\left(W_{\ell_n}, D_{\varepsilon_{\ell_n}}(W_{\ell_n})\right) : n \in \mathbb{N}\right\}$ be a limiting subsequence of the minimizers of $\{\mathcal{R}_{\varepsilon_\ell} : \ell \in \mathbb{N}\}$ and let $(\widetilde{W}, \widetilde{D})$ be its limit as $n \to \infty$. From the definition of $\mathcal{S}_\varepsilon$ it is clear that $\min\{\mathcal{S}_\varepsilon(W) : W \in \mathbb{R}^{d\times T}\}$ is a decreasing function of ε and converges to $\bar{\mathcal{S}} = \min\{\mathcal{S}_0(W) : W \in \mathbb{R}^{d\times T}\}$ as $\varepsilon \to 0$. Thus, $\mathcal{S}_{\varepsilon_{\ell_n}}(W_{\ell_n}) \to \bar{\mathcal{S}}$. Since $\mathcal{S}_\varepsilon(W)$ is continuous in both ε and W (see proof of Lemma 78), we obtain that $\mathcal{S}_0(\widetilde{W}) = \bar{\mathcal{S}}$. ■

Appendix E

Proof of Lemma 79

Lemma 79 *Let $P, N \in \mathbb{R}^{d\times T}$ such that $P^\top N = 0$. Then $\|P+N\|_{\mathrm{tr}} \geq \|P\|_{\mathrm{tr}}$. The equality is attained if and only if $N = 0$.*

PROOF. We use the fact that, for matrices $A, B \in \mathbf{S}^n_+$, $A \succeq B$ implies that $\mathrm{trace} A^{\frac{1}{2}} \geq \mathrm{trace} B^{\frac{1}{2}}$. This is true because the square root function on the reals, $t \mapsto t^{\frac{1}{2}}$, is *matrix monotone* – see Section 2.3. We apply this fact to the matrices $P^\top P + N^\top N$ and $N^\top N$ to obtain that

$$\|P+N\|_{\mathrm{tr}} = \mathrm{trace}((P+N)^\top (P+N))^{\frac{1}{2}} = \mathrm{trace}(P^\top P + N^\top N)^{\frac{1}{2}} \geq \\ \mathrm{trace}(P^\top P)^{\frac{1}{2}} = \|P\|_{\mathrm{tr}}.$$

The equality is attained if and only if the spectra of $P^\top P + N^\top N$ and $P^\top P$ are equal, whence $\mathrm{trace}(N^\top N) = 0$, that is $N = 0$. ■

Bibliography

J. Abernethy, F. Bach, T. Evgeniou, and J-P. Vert. Low-rank matrix factorization with attributes. Technical Report 2006/68/TOM/DS, INSEAD, 2006. Working paper.

L. T. H. An and P. D. Tao. A branch-and-bound method via d.c. optimization algorithm and ellipsoidal technique for box constrained nonconvex quadratic programming problems. *Journal of Global Optimization*, 13:171–206, 1998.

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 2007a. In press.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, 2007b.

A. Argyriou, R. Hauser, C. A. Micchelli, and M. Pontil. A DC-programming algorithm for kernel selection. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006a.

A. Argyriou, M. Herbster, and M. Pontil. Combining graph Laplacians for semi–supervised learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 67–74. MIT Press, 2006b.

A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceedings of the Eighteenth Conference on Learning Theory*, pages 338–352, 2005.

A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems*, 2007c. To appear.

A. Argyriou, C.A. Micchelli, and M. Pontil. Regularizers which admit a linear representer theorem. Working paper, Dept. of Computer Science, University College London, 2007d.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 686: 337–404, 1950.

J. P. Aubin. *Mathematical Methods of Game and Economic Theory*, volume 7 of *Studies in Mathematics and its applications*. North-Holland, 1982.

F. R. Bach, G. R. G Lanckriet, and M. I. Jordan. Multiple kernels learning, conic duality, and the smo algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

S. Bakin. *Adaptive regression and model selection in data mining problems*. PhD thesis, Australian National University, Canberra, 1999.

B. Bakker and T. Heskes. Task clustering and gating for bayesian multi–task learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

E. Bart and S. Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *Conference on Computer Vision and Pattern Recognition*, 2005.

J. Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28:7–39, 1997.

J. Baxter. A model for inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

M. Belkin and P. Niyogi. Semi–supervised learning on Riemannian manifolds. *Machine Learning*, 56: 209–239, 2004.

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proceedings of the Sixteenth Annual Conference on Learning Theory*, volume 2777 of *LNCS*, pages 567–580. Springer, 2003.

K. P. Bennett and M. J. Embrechts. An optimization perspective on partial least squares. In J. A. K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Models and Applications*, volume 190 of *NATO Science Series III: Computer & Systems Sciences*, pages 227–250. IOS Press Amsterdam, 2003.

K. P. Bennett, J. Hu, G. Kunapuli, and J.-S. Pang. Model selection via bilevel optimization. In *International Joint Conference in Neural Networks*, Vancouver, 2006.

K. P. Bennett, M. Momma, and M. J. Embrechts. Mark: a boosting algorithm for heterogeneous kernel models. In *Proceedings of the 8th SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

R. Bhatia. *Matrix analysis*. Graduate texts in Mathematics. Springer, 1997.

J. Bi, T. Zhang, and K. P. Bennett. Column-generation boosting methods for mixture of kernels. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 521–526. ACM Press, 2004.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Learning Theory*, 2001.

E. V. Bonilla, F. Agakov, and C. Williams. Kernel multi-task learning using task-specific features. In *Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

M. Borga. *Learning multidimensional signal processing*. PhD thesis, Dept. of Electrical Engineering, Linköping University, Sweden, 1998.

J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS Books in Mathematics. Springer, 2005.

L. Breiman and J. H. Friedman. Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society, Series B*, 59(1):3–54, 1997.

A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, August 2006.

R. Caruana. Multi–task learning. *Machine Learning*, 28:41–75, 1997.

O. Chapelle, V. N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

F. R. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 537–544. MIT Press, 2003.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2001.

G. Dai and D.-Y. Yeung. Kernel selection for semi-supervised kernel machines. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, 2007.

J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, 2007.

C. Ding, T. Li, and W. Peng. Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence, chi-square statistic, and a hybrid method. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-06)*, 2006.

F. Dominici, G. Parmigiani, K. H. Reckhow, and R. L. Wolpert. Combining information from related regressions. *Journal of Agricultural, Biological and Environmental Statistics*, 2(3):313–332, 1997.

D. Donoho. For most large underdetermined systems of linear equations, the minimal l1-norm near-solution approximates the sparsest near-solution. Preprint, Dept. of Statistics, Stanford University, 2004.

R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. Wiley, 1973.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

T. Evgeniou, M. Pontil, and O. Toubia. A convex optimization approach to modeling consumer heterogeneity in conjoint estimation. Forthcoming at Marketing Science, 2007.

M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings, American Control Conference*, volume 6, pages 4734–4739, 2001.

A. Ferencz, E. G. Learned-Miller, and J. Malik. Building a classification cascade for visual identification from one example. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 286–293, 2005.

M. Fink. Object classification from a single example utilizing class relevance metrics. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 449–456. MIT Press, 2005.

G. M. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.

A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press, 2007.

M. Girolami and S. Rogers. Hierarchic Bayesian models for kernel learning. In *Proceedings of the Twenty-Second International Conference in Machine Learning*, 2005.

M. Girolami and M. Zhong. Data integration for classification problems employing Gaussian process priors. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 465–472. MIT Press, 2007.

G. V. Glass. Primary, secondary, and meta-analysis of research. *Educational Researcher*, 5(10):3–8, 1976.

I. Gohberg, S. Goldberg, and M. A. Kaashoek. *Basic Classes of Linear Operators*. Birkhäuser, 2003.

C. Gold and P. Sollich. Model selection for support vector machine classification. *Neurocomputing*, 55: 221–249, 2003.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 513–520. MIT Press, 2005.

H. Goldstein. Multilevel modelling of survey data. *The Statistician*, 40:235–244, 1991.

G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

J. Guinney, Q. Wu, and S. Mukherjee. Estimating variable structure and dependence in multi-task learning via gradients. Working paper, 2007.

D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

P. Hartman. On functions representable as a difference of convex functions. *Pacific Journal of Mathematics*, 9:707–713, 1959.

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

T. Hastie and P. Simard. Models and metrics for handwritten character recognition. *Statistical Science*, 13(1):54–65, 1998.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag Series in Statistics, 2001.

B. Heisele, T. Serre, M. Pontil, T. Vetter, and T. Poggio. Categorization by learning and combining object parts. In *Advances in Neural Information Processing Systems 14*, pages 1239–1245. MIT Press, 2002.

M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 2005.

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

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

R. Horst and N. V. Thoai. DC programming: overview. *Journal of Optimization Theory and Applications*, 103:1–41, 1999.

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

A. J. Izenman. Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, 5:248–264, 1975.

T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. Technical Report No. AITR-1668, MIT, Cambridge, MA, 1999.

T. Jebara. Multi-task feature and kernel selection for SVMs. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schlkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, USA, 1998.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the Twentieth International Conference on Learning Theory*, 2003.

B. Juba. Estimating relatedness via data compression. In *Proceedings of the Twenty-Third International Conference on Machine learning*, 2006.

J. G. Kim, U. Menzefricke, and F. M. Feinberg. Assessing heterogeneity in discrete choice models using a Dirichlet process prior. *Review of Marketing Science*, 2, 2004. Article 1.

S.-J. Kim, A. Magnani, and S. Boyd. Optimal kernel selection in kernel Fisher discriminant analysis. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

R. Kondor and T. Jebara. Gaussian and wishart hyperkernels. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

I. Kreft and J. Leeuw. *Introducing multilevel modeling*. Sage, London, 1998.

B. Kulis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

M. H. Law and J. T. Kwok. Bayesian support vector regression. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 239–244, 2001.

D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, volume 13, pages 556–562. MIT Press, 2001.

P. J. Lenk, W. S. DeSarbo, P. E. Green, and M. R. Young. Hierarchical Bayes conjoint analysis: recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15(2):173–191, 1996.

A. S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2 (1/2):173–183, 1995.

Y. Lin and H. H. Zhang. Component selection and smoothing in smoothing spline analysis of variance models – COSSO. Technical Report 2556, Institute of Statistics Mimeo Series, NCSU, 2003.

M. G. Madden and T. Howley. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, 21(3):375–398, 2004.

B. K. Mallick and S. G. Walker. Combining information from several experiments with nonparameter priors. *Biometrika*, 84(3):697–706, 1997.

O. L. Mangasarian. Solution of general linear complementarity problems via nondifferentiable concave minimization. *Acta Mathematica Vietnamica*, 22(1):199–205, 1997.

B. Marthi, S. Russell, D.Latham, and C. Guestrin. Concurrent hierarchical reinforcement learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2005.

A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006a.

A. Maurer. The Rademacher complexity of linear transformation classes. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT)*, volume 4005 of *LNAI*, pages 65–78. Springer, 2006b.

C. A. Micchelli and A. Pinkus. Variational problems arising from balancing several error criteria. *Rendiconti di Matematica, Serie VII*, 14:37–86, 1994.

C. A. Micchelli and M. Pontil. Kernels for multi-task learning. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 921–928. MIT Press, 2005.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

C. A. Micchelli and M. Pontil. Feature space perspectives for learning the kernel. *Machine Learning*, 66:297–319, 2007. See also: Technical Report RN/05/11, Dept. of Computer Science, UCL.

C. A. Micchelli, M. Pontil, Q. Wu, and D-X. Zhou. Error bounds for learning the kernel. Technical Report RN/05/09, Dept. of Computer Science, UCL, 2005.

E. Miller, N. Matsakis, and P. Viola. Learning from one example through shared densities on transforms. In *Conference on Computer Vision and Pattern Recognition*, 2000.

G. Obozinski, B. Taskar, and M.I. Jordan. Multi-task feature selection. Technical report, Dept. of Statistics, UC Berkeley, June 2006.

C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 478–485. MIT Press, 2003.

C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

E. Parrado-Hernández, J. Arenas-García, I. Mora-Jiménez, and A. Navia-Vázquez. On problem-oriented kernel refining. *Neurocomputing*, 55:135–150, 2003.

K. Pelckmans. *Primal-Dual Kernel Machines*. PhD thesis, Katholieke Universiteit Leuven, May 2005.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR98 -14, Microsoft Research, 1998.

T. Poggio and F. Girosi. A sparse representation for function approximation. *Neural Computation*, 10: 1445–1454, 1998.

R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

R. M. Rifkin and R. A. Lippert. Value regularization and fenchel duality. *Journal of Machine Learning Research*, 8:441–479, 2007.

R. T. Rockafellar and R. J-B. Wets. *Variational Analysis*. Springer, 1998.

D. Rosenberg and P. L. Bartlett. The Rademacher complexity of co-regularized kernel classes. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007. To appear.

H. L. Royden. *Real Analysis*. Macmillan, New York, 1988.

K. Schittkowski. Optimal parameter selection in support vector machines. *Journal of Industrial and Management Optimization*, 1(4):465–476, November 2005.

I. J. Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, 39:811–841, 1938.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, 2002.

M. Seeger. Bayesian model selection for support vector machines, gaussian processes and other kernel classifiers. In S. A. Solla, T. K. Leen, and K-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 603–609. MIT Press, 2000.

T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. Theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. AI Memo 2005-036, Massachusetts Institute of Technology, 2005.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

D. L. Silver and R. E. Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science*, 8(2):277–294, 1996.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. In *Proceedings of the Twenty-Second ICML Workshop on Learning with Multiple Views*, 2005.

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

N. Srebro and S. Ben-David. Learning bounds for support vector machines with learned kernels. In *Proceedings of the Nineteenth Conference on Learning Theory*, volume 4005 of *LNAI*, pages 169–183. Springer, 2006.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

J. Steinig. A rule of signs for real exponential polynomials. *Acta Mathematica Hungarica*, 47(1):187–190, 1986.

D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 762–769, 2004.

K. Tsuda and W. S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20:i326–i333, 2004.

K. Tsuda, H. Shin, and B. Schölkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21, Suppl. 2:ii59–ii65, 2005.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, 2006.

S. Wold, A. Ruhe, H. Wold, and W. J. Dunn III. The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. *SIAM Journal of Scientific Computing*, 3:735–743, 1984.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 2005.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 68(1):49–67, 2006.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems 18*, pages 1585–1592. MIT Press, 2006.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi–supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1641–1648. MIT Press, 2005.