University of Pavia

Ph.D. Thesis in Mathematics and Statistics

Learning functions with kernel methods

Author:
Francesco Dinuzzo

Supervisor:
Prof. Giuseppe De Nicolao

January 9, 2011

Contents

Acknowledgements

Many people have influenced my research and my life during the years of my Ph.D. Without them, this thesis would not have been possible.

First of all, I wish to thank my supervisor, Giuseppe De Nicolao, for his excellent guidance and valuable advice that helped me to make the first steps into the world of research. Second, I would like to thank Gianluigi Pillonetto for very productive collaboration and numerous insightful discussions.

During my Ph.D program, I had the opportunity to live and conduct my studies in different countries, interacting with great people all over the world. This would not have been possible without the support of the following institutions: University of Pavia, Istituto Universitario degli Studi Superiori (IUSS), Consorzio Italia-MIT, Unione Matematica Italiana (UMI), National Science Council of Taiwan (NSC), ETH Zürich. I would also like to thank my supervisor and the coordinators of my Ph.D program Giuseppe Savaré and Gianpietro Pirola for allowing me to conduct research on leave for a consistent part of my program.

I wish to thank Tomaso Poggio for hosting my visit at the Center of Biological and Computational Learning (CBCL), Massachusetts Institute of Technology. The visit has been interesting and enjoyable thanks to Sven Eberhardt, Sharat Chikkerur, Lorenzo Rosasco, Huei-Han Jhuang, Jake Bouvrie, Federico Girosi, Gadi Geiger, and Kathleen Sullivan.

I have been lucky enough to spend a very inspiring summer at the National Taiwan University. I would like to thank Chih-Jen Lin for hosting my visit, and all the members of the Machine Learning group for making it so pleasant. At the risk of forgetting somebody, I would like to thank Hsiang-Fu Yu, Kai-Wei Chang, Cho-Jui Hsieh, Peng-Jen Chen, Tien-Liang Huang, and Li-Jen Chu.

I would like to thank Joachim Buhmann for supporting my research stay at ETH Zürich. It has been a pleasure to work with Cheng Soon Ong, and I remember several interesting discussions with Alberto Giovanni Busetto.

I owe my gratitude to many people at University of Pavia, where I grew up as a student and learned from so many excellent teachers. I wish to thank all the professors and researchers of the Department of Mathematics for creating and keeping an environment of open exchange of ideas and knowledge. I could not have asked for better officemates than Luca Natile, Francesco Bastianelli, Claudio Durastanti, Gloria Della Noce, Emanuele Raviolo, and Emanuele Dolera. I would also like to thank the following members of the Identification and Control of Dynamic Systems Laboratory (ICDS): Riccardo Porreca, Davide Martino Raimondo, Luca Capisani, Matteo Rubagotti, Alberto Russu, Lalo Magni, Antonella Ferrara, Giancarlo Ferrari-Trecate.

Finally, I would like to thank my parents, my sister, and Stefania for continuous support and encouragement during these years.

1 Machine learning and kernel methods

Machine learning is a discipline whose goal is to create algorithms that allow machines to learn from experience. For a computer machine, experience takes the form of a finite sequence of symbols called data, which can be organized in structures representing abstract entities such as graphs, images, text documents, etc.

In recent years, the growing availability and ubiquity of computational resources has made machine learning not only a scientific framework for understanding intelligence, but also a source of technologies for data analysis. Indeed, modern machine learning algorithms are able to address problems involving very large datasets, which would be untractable for humans.

1.1 Learning functions from data

The goal of a machine learning algorithm is to synthesize functional relationships on the basis of the data. A *learning algorithm* is a rule that associates a dataset $\mathcal{D}$ with a function $g : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are called *input set* and *output set*, respectively. Depending on the structure of the dataset, learning problems can be classified as being supervised, unsupervised, or semi-supervised.

1.1.1 Supervised learning

The dataset for a supervised learning problem is also called *training set*, and is made of a finite-number of input-output pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$:

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}\,.$$

The framework of supervised learning describes a scenario in which a supervisor shows a set of examples to a learner. There's a *training phase* in which the learner infers a functional relationship between inputs and outputs using a *supervised learning algorithm*. The learned relationship can be used during a *test phase* to make predictions over new inputs, possibly not present in the training set.

If all the inputs x_i are distinct, it is always possible to find a functional relationship that correctly predicts the output for all the examples in the training set.

However, such relationship doesn't necessarily ensures good predictive performances on new examples. *Overfitting* occurs when an overly complex function is used to correctly predict the training outputs, regardless of predictions on the test set.

Depending on the structure of the output set $\mathcal{Y}$, one can distinguish between two types of supervised learning problems: *classification* and *regression*.

In a classification problem, the output set contains a finite number $d > 1$ of categorical elements called *classes*:

$$\mathcal{Y} = \{c_1, \ldots, c_d\}\,. \tag{1.1}$$

It is customary to distinguish between *binary classification*, where $d = 2$, and *multi-class classification*, where $d > 2$.

In a regression problem, the output set $\mathcal{Y}$ contains numerical vectors and is usually assumed to be a subset of $\mathbb{R}^m$:

$$\mathcal{Y} \subseteq \mathbb{R}^m, \tag{1.2}$$

although, in practice, machines only have a finite number of symbols (e.g. bits) to represent numbers. When $m > 1$, the problem is called *multiple regression*, or *multi-output regression*, or *vector-valued regression*.

1.1.2 Unsupervised learning

Differently from supervised learning problems, where samples of both inputs and outputs are given, datasets for unsupervised learning problems are made only of inputs $x_i \in \mathcal{X}$:

$$\mathcal{D} = \{x_1, \ldots, x_\ell\}\,.$$

The goal is to learn a functional relationship $g : \mathcal{X} \to \mathcal{Y}$, where even the structure of the output set $\mathcal{Y}$ may possibly be learned from the data. Examples of unsupervised learning problems are *clustering* and *dimensionality reduction*.

Clustering is the unsupervised counterpart of classification, since it involve assigning inputs to groups (classes). The output set for clustering is of the form (1.1), where d denotes the number of clusters, an important parameter that can be either fixed a-priori or learned from the data.

Dimensionality reduction aims at finding an appropriate transformation to compress input patterns into a representation which is more parsimonious in terms of information. Methods for unsupervised *feature extraction* or *feature selection* belong to this class of learning algorithm. A large class of methodologies aim at extracting a reduced number of numerical *features* from an input pattern, and thus can be seen as an unsupervised counterpart of the regression problem, where the output set is chosen as in (1.2). The number m of features to extract is a parameter that may be subject of learning from the data as well. A particular case of feature extraction is feature selection (or variable selection), that occurs when the input set $\mathcal{X}$ is also a subset of $\mathbb{R}^n$, and the map $g : \mathcal{X} \to \mathcal{Y}$ depends only on a subset of components of the input vector. It should be observed that

unsupervised feature extraction is different from *feature extraction for the purpose of prediction*, which is usually obtained as a by-product of training process for supervised problems. Indeed, while the goal of unsupervised feature selection is to capture regularities in the input data for the purpose of information compression, supervised feature selection aims at finding good transformations of the input data which are informative in order to predict the output with a given supervised algorithm.

1.1.3 Semi-supervised learning

Semi-supervised learning problems falls in between supervised and unsupervised learning problems, and are characterized by the presence of both labeled and unlabeled examples in the dataset, namely:

$$\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U, \quad \mathcal{D}_L = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}, \quad \mathcal{D}_U = \{x_{\ell+1}, \ldots, x_{\ell+\ell_U}\}.$$

Semi-supervised problems are motivated by situations in which obtaining output labels is costly, whereas unlabeled examples abound. Is this the case of many data analysis problems involving very large datasets in which labeling requires human intervention.

Similarly to the supervised case, $\mathcal{D}$ is called training set, and the goal is to solve classification or regression problems. Under certain conditions, the availability of unlabeled examples can bring considerable benefit to the learning performances. For instance, in many classification problems, it is reasonable to postulate a *cluster assumption*, according to which optimal decision boundaries are likely to lie in regions of low input's density. Similarly, the *manifold assumption* states that input examples associated with the same class should lie in a common manifold in the input space. See [Chapelle et al., 2006] for a review of recent approaches to semi-supervised learning.

1.2 Kernels

A kernel is a similarity function that can be used to compare two objects belonging to some input domain $\mathcal{X}$.

Definition 1 (Kernel)**.** Let $\mathcal{X}$ denote a non-empty set. A kernel over $\mathcal{X}$ is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is symmetric

$$K(x_1, x_2) = K(x_2, x_1).$$

By fixing one of the two arguments of a kernel, one obtain a real-valued function defined over $\mathcal{X}$ which will play a key role in the following.

Definition 2 (Kernel section)**.** Given a kernel K, a *kernel section* centered on $\bar{x} \in \mathcal{X}$ is a function $K_{\bar{x}} : \mathcal{X} \to \mathbb{R}$ defined as

$$K_{\bar{x}}(x) = K(\bar{x}, x).$$

The next definition introduces a specific class of similarity functions that can be interpreted as generalized inner products, as shown in section 1.4.

Definition 3 (Positive semidefinite kernel)**.** Let $\mathcal{X}$ denote a non-empty set. A kernel K over $\mathcal{X}$ is called *positive semidefinite kernel* if, for any finite integer ℓ, it holds

$$\sum_{i=1}^{\ell}\sum_{j=1}^{\ell} c_i c_j K(x_i, x_j) \geq 0, \qquad \forall (x_i, c_i) \in (\mathcal{X}, \mathbb{R}), \quad i = 1, \ldots, \ell.$$

Although most of the theory and applications of kernel methods have been developed under the assumption of positive semidefiniteness, sometimes it might be useful to consider indefinite kernels as well. In subsection 1.5.4, we review some theory and results about machine learning with indefinite kernels.

1.2.1 Reproducing Kernel Hilbert Spaces

Reproducing Kernel Hilbert Spaces (RKHS) provides a connection between the problem of choosing an appropriate hypothesis space for learning functional relationships, and the concept of positive semidefinite kernel. In this subsection, we briefly review the central results in the theory of RKHS, and refer to standard references [Aronszajn, 1950, Saitoh, 1988] for more details.

Recall that the goal of a learning algorithm is to synthesize a function to make predictions for every possible $x \in \mathcal{X}$. Hence, it is natural to search for the predictor into a set of functions which are at least well-defined everywhere on $\mathcal{X}$. We would also like to introduce a suitable norm to evaluate the "complexity" (the size) of a function. A possible way to define a norm is to require that "small functions" are associated with small function values (namely, predictions that are close to zero). Such choice also induces the desirable property that functions close to each other produce similar predictions. In order to keep the notation simple, in this section we will focus on the problem of synthesizing a real-valued function, that is $\mathcal{Y} = \mathbb{R}$. In section 1.5, some techniques to deal with more general output sets are presented.

Definition 4 (Reproducing Kernel Hilbert Space)**.** A Reproducing Kernel Hilbert Space of functions $g : \mathcal{X} \to \mathbb{R}$ is an Hilbert space $\mathcal{H}$ such that, for all $x \in \mathcal{X}$, there exists $C_x \in \mathbb{R}$ such that

$$|g(x)| \leq C_x \|g\|_{\mathcal{H}}, \qquad \forall g \in \mathcal{H}.$$

In other words, the definition of RKHS requires that all the point-wise evaluation functionals $L_x : \mathcal{H} \to \mathbb{R}$, defined as

$$L_x g = g(x),$$

are bounded (continuous) functionals. The following theorem [Aronszajn, 1950] provides a correspondence between positive semidefinite kernels and reproducing kernel Hilbert spaces.

Theorem 1 (Moore-Aronszajn)**.** *To every RKHS $\mathcal{H}$ there corresponds a unique positive semidefinite kernel K, called the reproducing kernel, such that the following* reproducing property *holds:*

$$g(x) = \langle g, K_x \rangle_{\mathcal{H}}, \qquad \forall (x, g) \in (\mathcal{X}, \mathcal{H}) \tag{1.3}$$

Conversely, given a positive semidefinite kernel K, there exists a unique RKHS of real valued functions defined over $\mathcal{X}$ whose reproducing kernel is K.

The following result characterizes the solution of a large class of optimization problems defined over an RKHS $\mathcal{H}_K$, featuring the minimization of a functional $J : \mathcal{H}_K \to \mathbb{R} \cup \{+\infty\}$ defined as

$$J(g) = Q\left(g(x_1), \ldots, g(x_\ell), \|g\|_{\mathcal{H}_K}\right), \tag{1.4}$$

where

$$Q : \mathbb{R}^\ell \times \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\}$$

is an extended-valued functional. Observe that $J(g)$ depends on g only through a finite number of point-wise evaluations and the norm $\|g\|_{\mathcal{H}_K}$. The structure of the objective functional encodes the idea that optimal solutions of a learning problem should depend only on the predictions over a finite set of available input data, and should not be "too complex", where complexity is measured by the RKHS norm.

Theorem 2 (Representer theorem)**.** *Let $\mathcal{X}$ denote a non-empty set, and $x_i \in \mathcal{X}$ $(i = 1, \ldots, \ell)$ a finite number of elements. Let $\mathcal{H}_K$ be an RKHS of real-valued functions defined over $\mathcal{X}$ with reproducing kernel K. If there exist minimizers of functional J defined as in (1.4), and Q is non-decreasing in the last argument, then there exists at least one minimizer g^* such that*

$$g^*(x) = \sum_{i=1}^{\ell} c_i K_{x_i}(x). \tag{1.5}$$

The importance of the representer theorem lies in the fact that the optimal solution of a broad range of optimization problems can be represented as a linear combination of a finite number of kernel sections centered on the input data, independently of the dimensionality of the search space. Philosophically, the representer theorem formalize the intuition that, having a finite amount of available data, it is not possible to learn arbitrarily complex functions. From a practical point of view, it makes possible to carry out inference in possibly infinite-dimensional spaces without introducing approximations, by solving finite-dimensional optimization problems.

Many different versions of the representer theorem have appeared in the literature, see [Schölkopf et al., 2001] and references therein. One of the earliest instances can be found in [Kimeldorf and Wahba, 1971]. Theorem 2 is a rather general version that encompasses many results. Since the functional J can assume the value $+\infty$, it is also possible to include hard constraints jointly on the predictions $g(x_i)$ and the norm $\|g\|_{\mathcal{H}_K}$. Observe that existence of minimizers of

functionals of the form (1.4) is taken as an hypothesis. If Q is *strictly increasing* in the last argument, it holds that *every* minimizer of J can be written as in (1.5). These facts are summarized in the following theorem, that also gives a sufficient condition for existence of minimizers.

Theorem 3. *If Q is lower semicontinuous and coercive in the last argument, there exist minimizers of J defined in (1.4). By the representer theorem, at least one of them can be written in the form (1.5). If Q is strictly increasing in the last argument, then all the minimizers are in the form (1.5).*

Once an optimal solution is expressed in the form (1.5), the problem reduces to the characterization of coefficients c_i of the linear combination. To address this issue, we introduce the following definition.

Definition 5 (Kernel matrix)**.** Given a kernel K over $\mathcal{X}$, the *kernel matrix* associated with K and a set of points $x_i \in \mathcal{X}$, $i = 1, \dots, \ell$, is the matrix $\mathbf{K} \in \mathbb{R}^{\ell \times \ell}$ whose entries k_{ij} are defined as

$$k_{ij} = K(x_i, x_j).$$

By plugging the representation (1.5) into (1.4) and using the reproducing property (1.3), the problem of minimizing J boils down to the problem of obtaining a vector of coefficients $c \in \mathbb{R}^{\ell}$, by minimizing a functional of the type

$$F(c) = Q\left(\mathbf{K}c, \sqrt{c^T \mathbf{K} c}\right),$$

where $\mathbf{K}$ is the kernel matrix associated with K and the set of data points $x_i \in \mathcal{X}$.

1.3 Regularization

A mathematical problem whose solution depends on data is said to be well-posed in the Hadamard sense if it admits a unique solution whose dependence on the data is continuous. Problems which do not satisfy this condition are called *ill-posed*.

The classical theory of regularization [Tikhonov and Arsenin, 1977] has been developed to solve ill-posed problems, by converting them to well-posed ones. The regularization approach has been very successful in many areas of science and engineering, such as system identification, computational vision, geophysics, remote sensing, and signal processing.

Regularization theory has been introduced to the machine learning literature in [Poggio and Girosi, 1990], where it is shown that a particular regularization algorithm is equivalent to the training of a radial basis function (RBF) network. The modern approach to regularization in machine learning uses the framework of RKHS, within which different regularization methods such as smoothing splines [Wahba, 1990], regularization networks [Girosi et al., 1995], Gaussian processes [Rasmussen and Williams, 2006], and support vector machines [Vapnik, 1998], can be seen to be related to each other and treated in a unified way.

In this thesis, we use the expression *regularized kernel methods* to denote those learning algorithms in which the solution is searched within an RKHS by minimizing a functional of the type (1.4), where

$$Q(z,t) = f(z) + \Omega(t), \tag{1.6}$$

$f : \mathbb{R}^\ell \to \mathbb{R}$ is a function (usually depending on data) called *empirical risk*, and $\Omega : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{+\infty\}$ is called *regularizer*, or *regularization term*. The following theorem summarizes important facts about existence, uniqueness, and characterization of solutions for a regularization problems. The first two statements are immediate corollaries of Theorems 2 and 3.

Theorem 4. *Let $\mathcal{H}^* \subset \mathcal{H}$ denote the set of minimizers of functional (1.4), where (1.6) holds, f and Ω are lower semi-continuous, and Ω is non-decreasing. The following statements hold:*

1. *If Ω is coercive, then $\mathcal{H}^*$ is non-empty and bounded.*
2. *If Ω is strictly increasing, then any $g^* \in \mathcal{H}^*$ satisfy (1.5).*

When f is convex, the following additional statements hold:

1. *If Ω is convex, then $\mathcal{H}^*$ is a convex set.*
2. *If Ω is strictly convex, then $\mathcal{H}^* = \{g^*\}$ is a singleton, and g^* satisfy (1.5).*

Common choices for Ω include the Tikhonov regularizer

$$\Omega(t) = \frac{t^2}{2},$$

or the Ivanov regularizer

$$\Omega(t) = \begin{cases} 0, & 0 \le t \le 1 \\ +\infty, & t > 1 \end{cases}.$$

Both the regularizers are lower-semicontinuous (extended-valued) convex, and coercive. In addition, the Tikhonov regularizer is strictly increasing and strictly convex.

1.4 Feature maps

In this section, it is shown that positive semidefinite kernels can be interpreted as generalized inner products. Such interpretation has been widely exploited to obtain non-linear versions of classical linear algorithms, a method sometimes referred to as the "kernel trick" [Schölkopf and Smola, 2001], which was used already in [Aizerman et al., 1964]. The following equivalent definition of positive semidefinite kernel is a corollary of the Moore-Aronszajn Theorem.

Corollary 1 (Feature space representation)**.** *Let $\mathcal{X}$ denote a non-empty set. A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semidefinite kernel if and only if there exist an Hilbert space $\mathcal{F}$ and a map $\Phi : \mathcal{X} \to \mathcal{H}$ called* feature map*, such that*

$$K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle_{\mathcal{F}}. \tag{1.7}$$

The space $\mathcal{F}$ and the feature map Φ are not unique.

On one hand, it is straightforward to verify that (1.7) implies positive semidefiniteness of K. On the other hand, the Moore-Aronszajn theorem gives a way to explicitly build at least one feature map for a given positive semidefinite kernel, namely the RKHS feature map.

1.4.1 RKHS feature map

Definition 6 (RKHS feature map)**.** Given a positive semidefinite kernel K, its *RKHS feature map* is a function Φ that associates any $x \in \mathcal{X}$ with the corresponding kernel section centered on x:

$$\Phi : X \to \mathcal{H}, \qquad \Phi(x) = K_x.$$

To see that the feature space representation (1.7) holds, it suffices to set $\mathcal{F} = \mathcal{H}$, and apply the reproducing property (1.3) to a pair of kernel sections:

$$K(x_1, x_2) = \langle K_{x_1}, K_{x_2} \rangle_{\mathcal{H}}.$$

The RKHS feature map is not the only possible one. In fact, other maps may give additional insights into the structure of an RKHS, as well as providing different interpretations of the kernel function.

1.4.2 Spectral feature maps

Consider the problem of *diagonalizing* a positive semidefinite kernel, namely obtaining a representation of the type

$$K(x_1, x_2) = \sum_{i \in \mathcal{I}} \lambda_i \phi_i(x_1) \phi_i(x_2), \tag{1.8}$$

where $\mathcal{I}$ is a set which is at most countable, and functions ϕ_i are orthonormal with respect to some inner product. When the input set has finite cardinality, the set of positive semidefinite kernels over $\mathcal{X}$ is isomorphic to the set of symmetric positive semidefinite matrices of finite order. Then, letting $\mathcal{I} = \mathcal{X}$, representation (1.8) follows from the spectral theorem, where $\lambda_i \geq 0$ are the eigenvalues, and ϕ_i are the orthonormal eigenvectors of the linear operator associated with the corresponding kernel matrix.

One possible generalization of this last result is the following. Let $(\mathcal{X}, M, \mu)$ be a σ-finite measure space. A square integrable kernel on $\mathcal{X}$ is a kernel $K \in$

$L^2_{\mu\otimes\mu}(\mathcal{X}\times\mathcal{X})$. Each square integrable kernels is associated with a bounded integral operator $T_K : L^2_\mu(\mathcal{X}) \to L^2_\mu(\mathcal{X})$ defined as

$$\langle T_K\varphi, \psi\rangle_{L^2} = \int_{\mathcal{X}\times\mathcal{X}} K(x_1,x_2)\varphi(x_1)\psi(x_2)d[\mu\otimes\mu](x_1,x_2),$$

which is a positive semidefinite compact Hilbert-Schmidt operator. By the spectral theorem for compact operators, T_K generates an orthonormal basis of $L^2_\mu(\mathcal{X})$ made of eigenvectors $\{\phi_i\}_{i\in\mathcal{I}}$ associated with strictly positive eigenvalues $\{\lambda_i\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is at most countable. It turns out that the eigenvalues are square summable, that is $\sum_{i\in\mathcal{I}}\lambda_i^2 < +\infty$, and the series on the right hand side of (1.8) converges to the quantity on left hand side in the $L^2_\mu(\mathcal{X})$ norm. The following theorem gives a condition under which the series converges uniformly, see [Mercer, 1909, Cucker and Smale, 2001].

Theorem 5 (Mercer)**.** *Let $(\mathcal{X}, M, \mu)$ denote a σ-finite measure space. Let K denote a positive semidefinite kernel over $\mathcal{X}$ which is square integrable. If $\mathcal{X}$ is a compact space with respect to some metric, and K is continuous, then (1.8) holds, and convergence of the series is absolute and uniform.*

The representation (1.8) gives another class of feature maps that highlights the spectral properties of a kernel function.

Definition 7 (Spectral feature maps)**.** Let K denote a positive definite kernel over $\mathcal{X}$ such that a representation of the type (1.8) holds, where $\sum_{i\in\mathcal{I}}\lambda_i^2 < +\infty$ and $\{\phi_i\}_{i\in\mathcal{I}}$ is a set of orthonormal functions with respect to some inner product. Then, a *spectral feature map* of K is defined as:

$$\Phi_\mu : \mathcal{X} \to \ell^2(\mathcal{I}), \qquad \Phi_\mu(x) = \left\{\sqrt{\lambda_i}\phi_i(x)\right\}_{i\in\mathcal{I}}.$$

Here, $\ell^2(\mathcal{I})$ is either a finite dimensional Euclidean space or the space of square summable sequences. Representation (1.7) follows by taking $\mathcal{F} = \ell^2(\mathcal{I})$, and applying (1.8). Observe that spectral maps are not unique, since eigenvalues and eigenvectors depend on the measure μ.

1.4.3 Stochastic process feature map

We now discuss the interpretation of positive semidefinite kernel functions as covariance functions of stochastic processes. Let $\mathcal{S}$ denote the Hilbert space of real-valued random variables whose inner product is given by the covariance

$$\langle \mathbf{A}_1, \mathbf{A}_2\rangle_{\mathcal{S}} := \operatorname{cov}(\mathbf{A}_1, \mathbf{A}_2).$$

Let $\mathbf{U}_x$ denote a real-valued zero mean stochastic process indexed by $x \in \mathcal{X}$ with covariance

$$K(x_1,x_2) = \operatorname{cov}(\mathbf{U}_{x_1}, \mathbf{U}_{x_2}).$$

Then, it is natural to define a new feature map as follows.

Definition 8 (Stochastic process feature map)**.** A stochastic process $\mathbf{U}_x$ defined over $\mathcal{X}$, whose covariance function is given by the positive semidefinite kernel K, induces the following feature map:

$$\Phi_{\mathbf{U}} : \mathcal{X} \to \mathcal{S}, \qquad \Phi_{\mathbf{U}}(x) = \mathbf{U}_x.$$

The equivalence between positive semidefinite kernel functions and covariance functions of stochastic processes is particularly insightful in the case of centered Gaussian Processes, which are uniquely identified by the covariance function.

1.4.4 Fourier map for translation invariant kernels

A kernel over $\mathcal{X} = \mathbb{R}^d$ is called *translation invariant* if there exists a real-valued function R such that

$$K(x_1, x_2) = R(x_1 - x_2).$$

In this subsection, we derive a feature map associated with continuous and translation invariant positive semi-definite kernels over $\mathbb{R}^d$. Recall the following result, that characterizes the Fourier transform of a positive measure, see [Rudin, 1994].

Theorem 6 (Bochner)**.** *A positive semidefinite kernel K over $\mathcal{X} = \mathbb{R}^d$ is continuous and translation invariant if and only if there exists a probability measure μ and a number $\alpha \geq 0$ such that:*

$$K(x_1, x_2) = \alpha \int_{\mathcal{X}} \cos\left(\langle \xi, x_1 - x_2 \rangle_2\right) d\mu(\xi).$$

Let $\mathcal{S}^d$ denote the Hilbert space of random vectors in $\mathbb{R}^d$, whose inner product is given by the covariance. Introduce a generic random vector $\mathbf{U} \in \mathcal{S}^d$ distributed according to μ, and let

$$\mathbf{Z}_{\mathbf{U}}(x) := \sqrt{\alpha} \begin{pmatrix} \cos(\langle \mathbf{U}, x \rangle_2) \\ \sin(\langle \mathbf{U}, x \rangle_2) \end{pmatrix} \in \mathcal{S}^2. \tag{1.9}$$

Using Bochner theorem and the sum rule of cosines, one can verify that

$$K(x_1, x_2) = \langle \mathbf{Z}_{\mathbf{U}}(x_1), \mathbf{Z}_{\mathbf{U}}(x_2) \rangle_{\mathcal{S}^2},$$

so that we obtain the following feature map.

Definition 9 (Fourier feature map)**.** Given any continuous translation invariant positive semidefinite kernel K over $\mathcal{X} = \mathbb{R}^d$, its Fourier feature map is defined as:

$$\Phi_F : \mathcal{X} \to \mathcal{S}^2, \qquad \Phi_F(x) = \mathbf{Z}_{\mathbf{U}}(x),$$

where $\mathbf{Z}_{\mathbf{U}}(x)$ is defined as in (1.9).

1.5 Extensions

In this section, some extensions of the standard theory of positive semidefinite kernels and RKHS are reviewed.

1.5.1 Learning with structured outputs

So far, we focussed on the case in which outputs are real numbers. For supervised learning problems, there's a simple technique, called *structured output learning* [Tsochantaridis, 2005, Bakir et al., 2007], that can be used to learn functional relationships with generic output set $g : \mathcal{X} \to \mathcal{Y}$. The main idea is to model the input-output relationship as

$$g(x) = \arg\max_{y \in \mathcal{Y}} h(x, y), \tag{1.10}$$

(max can be equivalently replaced by the min) where $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is to be searched within a standard RKHS of real valued functions. Hence, predictions are obtained by solving an optimization problem whose complexity depends on the structure of the output set $\mathcal{Y}$ and the function h. In order to obtain h, one can minimize a functional J of the standard form (1.4), where the inputs x_i are replaced by the pairs (x_i, y_i), namely

$$J(h) = Q(h(x_1, y_1), \ldots, h(x_\ell, y_\ell), \|h\|_{\mathcal{H}_K}),$$

so that the problem is basically reduced to the scalar case. Under the assumptions of the representer theorem, there exists an optimal h^* in the form:

$$h^*(x, y) = \sum_{i=1}^{\ell} c_i K_{(x_i, y_i)}(x, y).$$

The choice of the *joint kernel* K over $\mathcal{X} \times \mathcal{Y}$ is crucial, since one need to ensure that the optimization problem (1.10) is well-posed, and can be solved efficiently.

1.5.2 Learning vector-valued functions

In this subsection, the definition of kernel is properly generalized to deal with functions mapping a generic set $\mathcal{X}$ into a vector space $\mathcal{Y}$. A theory of reproducing kernel Hilbert spaces of functions with values in a general locally convex topologically spaces has been developed in [Schwartz, 1964]. The presentation here is focussed on the case in which $\mathcal{Y}$ is an Hilbert space, and follows [Micchelli and Pontil, 2005]. Let $\mathcal{L}(\mathcal{Y})$ denote the space of bounded linear operators from $\mathcal{Y}$ into itself, and consider the following definitions.

Definition 10 ($\mathcal{Y}$-kernel)**.** Let $\mathcal{X}$ denote a non-empty set and $\mathcal{Y}$ an Hilbert space. An $\mathcal{Y}$-kernel over $\mathcal{X}$ is a function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ which is symmetric

$$K(x_1, x_2) = K(x_2, x_1).$$

Definition 11 (Kernel section)**.** Given an $\mathcal{Y}$-kernel K, a *kernel section* centered on $\bar{x} \in \mathcal{X}$ is a map $K_{\bar{x}} : \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ defined as

$$K_{\bar{x}}(x) = K(\bar{x}, x).$$

Definition 12 (Positive semidefinite $\mathcal{Y}$-kernel)**.** Let $\mathcal{X}$ denote a non-empty set and $\mathcal{Y}$ an Hilbert space. An $\mathcal{Y}$-kernel K is called *positive semidefinite* if, for any finite integer ℓ, it holds

$$\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\langle y_i, K(x_i, x_j)y_j\rangle_{\mathcal{Y}} \geq 0, \qquad \forall (x_i, y_i) \in (\mathcal{X}, \mathcal{Y}), \quad i = 1, \ldots, \ell.$$

Positive semidefinite $\mathcal{Y}$-kernels are associated with suitable spaces of vector-valued functions.

Definition 13 (Reproducing Kernel Hilbert Space of $\mathcal{Y}$-valued functions)**.** A Reproducing Kernel Hilbert Space (RKHS) of $\mathcal{Y}$-valued functions $g : \mathcal{X} \to \mathcal{Y}$ is an Hilbert space $\mathcal{H}$ such that, for all $x \in \mathcal{X}$, there exists $C_x \in \mathbb{R}$ such that

$$\|g(x)\|_{\mathcal{Y}} \leq C_x \|g\|_{\mathcal{H}}, \qquad \forall g \in \mathcal{H}.$$

It turns out that Moore-Aronszajn theorem can be extended to the vector-valued case, with no substantial changes in the proof. Hence, it holds that positive semidefinite operator-valued kernels are in a one-to-one correspondence with reproducing kernel Hilbert spaces of vector-valued functions, where the reproducing property now reads

$$\langle g(x), y\rangle_{\mathcal{Y}} = \langle g, K_x y\rangle_{\mathcal{H}}, \qquad \forall (x, y, g) \in (\mathcal{X}, \mathcal{Y}, \mathcal{H}).$$

The representer theorem can be also extended to RKHS of vector-valued functions, so that the search for solution of optimization problems of the type (1.4) can be restricted to the set of finite combinations of kernel sections of the type:

$$g(x) = \sum_{i=1}^{\ell} K_{x_i}(x)c_i, \qquad c_i \in \mathcal{Y}, \qquad i = 1, \ldots, \ell.$$

Let's analyze further the case in which $\mathcal{Y}$ is a vector space of finite dimension m, and K is an $\mathcal{Y}$-kernel over $\mathcal{X}$. By fixing a basis $\{b_i\}_{i\in\mathcal{T}}$ for the output space, where $\mathcal{T} = \{1, \ldots, m\}$, one can uniquely define an associated (scalar-valued) kernel R over $\mathcal{X} \times \mathcal{T}$ such that

$$\langle b_i, K(x_1, x_2)b_j\rangle_{\mathcal{Y}} = R((x_1, i), (x_2, j)),$$

so that an $\mathcal{Y}$-kernel can be seen as a function that maps two inputs into the space of square matrices of order m. Similarly, by fixing any function $g : \mathcal{X} \to \mathcal{Y}$, one can uniquely define an associated function $h : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$ such that

$$g(x) = \sum_{i\in\mathcal{T}} h(x, i)b_i.$$

Consequently, a space of $\mathcal{Y}$-valued functions over $\mathcal{X}$ is isomorphic to a standard space of scalar-valued functions defined over the input set $\mathcal{X} \times \mathcal{T}$. In conclusion, by fixing a basis for the output space, the problem of learning vector-valued functions can be also reduced to the scalar case over an enlarged input set.

1.5.3 Learning the kernel

By choosing the kernel function, we have a rather flexible way to embed prior knowledge into a machine learning algorithm. However, the available prior knowledge might not be sufficient to uniquely determine the best kernel for a given problem among the set of all possible kernels on a given domain $\mathcal{X}$. This observation leads to the idea of learning the kernel function itself from the data, which is motivating a considerable amount of research in recent years.

First of all, observe that the set of all possible positive semidefinite kernels over $\mathcal{X}$ can be characterized as follows.

Lemma 1. *The set of positive semidefinite kernels over $\mathcal{X}$ is a cone $\mathcal{A}_+(\mathcal{X})$ which is convex, pointed, salient, and closed in the uniform metric.*

From Lemma 1, one can obtain a variety of rules to build new kernels from elementary components. Since $\mathcal{A}_+(\mathcal{X})$ is a pointed convex cone, it follows that a linear combination of basis positive semidefinite kernels with non-negative coefficients is still a positive semidefinite kernel. In particular, the null kernel $K = 0$ is a valid positive semidefinite kernel. Since the cone is salient, for any non-null positive semidefinite kernel K, its opposite $-K$ is not positive semidefinite. Finally, since the cone is closed in the uniform metric, the limit of a uniformly convergent sequence of positive semidefinite kernels is still a positive semidefinite kernel.

A possible framework to attack the problem of learning the kernel is the one based on minimization problems of the form

$$\min_{K \in \mathcal{K}} \min_{g \in \mathcal{H}_K} Q\left(g(x_1), \ldots, g(x_\ell), \|g\|_{\mathcal{H}_K}\right), \tag{1.11}$$

where $\mathcal{K} \subset \mathcal{A}_+(\mathcal{X})$ is a subset of the cone of positive semidefinite kernels. Under the assumptions of Theorem 2, as soon as the inner problem admit minimizers, there exists an optimal g^* in the form (1.8). It follows that (1.11) can be rewritten as

$$\min_{\mathbf{K} \in \mathcal{M}} \min_{c \in \mathbb{R}^\ell} Q\left(\mathbf{K}c, \sqrt{c^T \mathbf{K} c}\right), \tag{1.12}$$

where $\mathcal{M} \subset \mathbb{S}_+^\ell$ is the set of kernel matrices associated with all the kernel functions in $\mathcal{K}$ and the data points $x_1, \ldots, x_\ell$. The problem is then reduced to finding a symmetric positive semidefinite matrix and a vector of coefficients. A popular choice for $\mathcal{K}$ is the convex hull of a finite set of basis kernels:

$$\mathcal{K} = \operatorname{co}\left\{\widetilde{K}_1, \ldots, \widetilde{K}_m\right\}, \tag{1.13}$$

meaning that every $K \in \mathcal{K}$ satisfies

$$K = \sum_{i=1}^{m} d_i \widetilde{K}_i, \quad d_i \geq 0, \quad \sum_{i=1}^{m} d_i = 1.$$

Multiple kernel learning (MKL) is a family of optimization algorithms of the type (1.12) in which $\mathcal{K}$ is chosen as in (1.13).

More generally, when $\mathcal{K}$ is compact and convex, a theorem of Carathéodory's can be used to characterize the solution of the outer optimization problem, as shown in [Argyriou et al., 2005].

Theorem 7. *Under the assumptions of Theorem 2, if $\mathcal{K}$ is compact convex and there exist minimizers in (1.11), then there exist basis kernels $\widetilde{K}_i$ $(i = 1, \ldots, m)$, and an optimal kernel K^* such that*

$$K^* \in co\left\{\widetilde{K}_1, \ldots, \widetilde{K}_m\right\} \subseteq \mathcal{K}, \qquad m \leq \ell + 1.$$

Hence, whenever $\mathcal{K}$ is compact and convex, problem (1.11) is equivalent to a suitable MKL problem where the number of basis kernels does not exceed $\ell + 1$. Unfortunately, the theorem does not specify how to find the basis kernels $\widetilde{K}_i$, so that this property cannot be directly used to simplify the optimization problem.

Apart from methods that boils down to optimization problems of the form (1.12), there are at least two other methodologies that address the problem of learning the kernel. The first is based on the idea of searching the kernel function itself into another RKHS of functions defined over $\mathcal{X} \times \mathcal{X}$ whose kernel, called *hyper-kernel*, satisfying additional symmetry properties [Ong et al., 2005]. The second is based on the framework of *kernel machines with two layers* developed in this thesis. The two methodologies share the idea of introducing an additional regularization layer in the optimization problem, but lead to different optimization problems.

1.5.4 Indefinite kernels

Indefinite kernels have been sometimes employed in machine learning with some degree of success, so that a certain amount of theory has been produced to interpret optimization methods with indefinite kernel from a functional analytic point of view. Along this direction, it has been proposed to generalize Reproducing Kernel Hilbert Spaces, whose theory is heavily based on the assumption of positive semidefiniteness of the kernel, by introducing RKKS (Reproducing Kernel Krein Spaces) [Ong et al., 2004].

A Krein space can be obtained by the following construction. Consider the direct sum $\mathcal{H} = \mathcal{H}_+ \oplus \mathcal{H}_-$ of two Hilbert spaces, and observe that $\mathcal{H}$ is automatically endowed with the inner product:

$$\langle g_1, g_2 \rangle_{\mathcal{H}} = \langle g_1^+, g_2^+ \rangle_{\mathcal{H}_+} + \langle g_1^-, g_2^- \rangle_{\mathcal{H}_-},$$

with natural meaning of the notation. Now, $\mathcal{H}$ can be turned into a Krein space by endowing it with an additional symmetric bilinear form defined as

$$\langle g_1, g_2 \rangle_{\mathcal{H}}^- = \langle g_1^+, g_2^+ \rangle_{\mathcal{H}_+} - \langle g_1^-, g_2^- \rangle_{\mathcal{H}_-}, \tag{1.14}$$

whose associated quadratic form is not necessarily positive semidefinite.

Definition 14 (Reproducing Kernel Krein Space)**.** A Reproducing Kernel Krein Space (RKKS) of functions $g : \mathcal{X} \to \mathbb{R}$ is the direct sum of two RKHS $\mathcal{H} = \mathcal{H}_+ \oplus \mathcal{H}_-$ endowed with an additional symmetric bilinear form defined in (1.14).

Letting K_+ and K_- denote the reproducing kernels of $\mathcal{H}_+$ and $\mathcal{H}_-$, respectively, the RKKS $\mathcal{H}$ can be associated with two different kernels, namely the sum $K = K_+ + K_+$, which is positive semidefinite, and the difference $\underline{K} = K_+ - K_-$, which may be indefinite. Indeed, there's two different reproducing properties, namely the usual (1.3) and the new one:

$$g(x) = \langle g, \underline{K}_x \rangle_{\mathcal{H}}^{-}, \qquad \forall (x, g) \in (\mathcal{X}, \mathcal{H}) .$$

Conversely, it can be shown that a generic indefinite kernel $\underline{K}$ can be associated to an RKKS if and only if it can be expressed as the difference of two positive semidefinite kernels.

Now, the goal is to derive learning algorithms characterized by a representer theorem involving the (possibly) indefinite kernel $\underline{K}$. The key idea is to introduce functionals of the form

$$f\left(g(x_1), \ldots, g(x_\ell)\right) + \frac{\langle g, g \rangle_{\mathcal{H}}^{-}}{2},$$

and looking for their stationary points (rather than their minimizers). It can be shown that, under suitable conditions on f, there exist stationary points g^* which admit the expansion:

$$g^*(x) = \sum_{i=1}^{\ell} c_i \underline{K}_{x_i}(x).$$

The problem then reduces to finding a stationary point with respect to $c \in \mathbb{R}^\ell$ of the functional

$$f(\underline{\mathbf{K}} c) + \frac{c^T \underline{\mathbf{K}} c}{2},$$

where $\underline{\mathbf{K}}$ is the kernel matrix associated with $\underline{K}$ and the points x_i $(i = 1, \ldots, \ell)$, which may be indefinite.

In chapter 2 we will show that, by allowing indefinite kernel matrices, one can solve convex non-differentiable kernel regularization problems with positive semidefinite kernel by finding stationary points of a corresponding differentiable functional with indefinite kernel.

1.6 Techniques for large scale problems

Considerable effort has been devoted to make kernel methods feasible on large scale problems, see e.g. [Bottou et al., 2007]. In this section, we review some techniques that have been proposed in the literature to address this issue.

1.6.1 The kernel matrix may not fit into the memory

Computing weighted combination of kernel functions is a crucial operation in many kernel-based algorithms. To practically implement such operation, one

must typically face a trade-off between computational efficiency and memory occupation. The most memory efficient solution would be not storing any of the entries of the kernel matrix, thus re-computing them when necessary. However, this solution requires evaluating the kernel function (a computationally intensive operation) many times on the same data pairs, which might significantly reduce computation speed. On the other extreme, one can consider evaluating the kernel function once for all the data pairs and storing all the results into the memory. However, storing the whole kernel matrix into the memory might be unfeasible. For instance, when the kernel matrix is fully dense, the amount of memory required to store all the entries scales with $O(\ell^2)$. For a learning problem with $\ell = 10^6$ examples and single precision accuracy (4 bytes for each entry), storing a single row of the kernel matrix requires 4 MegaBytes of memory, while storing the whole matrix requires 4 TeraBytes (about the half by exploiting symmetry). Apparently, storing the kernel matrix is an approach that doesn't scale well to large problems.

1.6.2 Exact kernel factorizations

Even when storing the kernel matrix is possible, plain matrix-vector multiplication need not to be the most efficient way to compute the product and, indeed, there are many situations in which other solutions work better. A representative example is given by the case in which K is the linear kernel on $\mathbb{R}^n$ defined as

$$K(x_1, x_2) = \langle x_1, x_2 \rangle_2,$$

which imply the decomposition $\mathbf{K} = \mathbf{X}\mathbf{X}^T$, where $\mathbf{X} \in \mathbb{R}^{\ell \times n}$ is the matrix that contains the input data. In the linear case, the whole matrix-vector product $z = \mathbf{K}c$ can be computed in two steps without actually forming the kernel matrix:

$$\begin{aligned} w &= \mathbf{X}^T c, \\ z &= \mathbf{X} w. \end{aligned}$$

The computational cost is $O(n\ell)$, which may be much better than $O(\ell^2)$. When the matrix containing input data is sparse, the computational cost is further reduced. More generally, techniques that are suitable for the linear case also apply whenever the kernel matrix admits a compact factorization of the form $\mathbf{K} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$, where the factor $\widetilde{\mathbf{X}}$ has some desirable property such as being low-rank, sparse or both. When a closed-form expression for the feature map is available and involve a finite number of features, one can reduce exactly to the linear case, possibly at the expense of a high number of features. Is this the case of polynomial kernels

$$K(x_1, x_2) = (1 + \langle x_1, x_2 \rangle_2)^d,$$

for which is it possible to work out explicit feature maps involving a finite, though exponential in d, number of features. Taking advantage of sparsity in the original features which propagates to the derived features, the idea has been applied with success to low degree polynomial kernels [Chang et al., 2010].

1.6.3 Approximate kernel factorizations

In order to handle more general situations, one can compute approximate factorizations of the kernel matrix of the form

$$\mathbf{K} \approx \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T,$$

where the entries $\widetilde{x}_{ij}$ of the factor $\widetilde{\mathbf{X}}$ can be obtained using a variety of methods, including incomplete Cholesky factorization [Fine and Scheinberg, 2001, Bach and Jordan, 2002, 2005, Kulis et al., 2006], approximate eigendecomposition based on the Nyström method [Williams and Seeger, 2000, Drineas and Mahoney, 2005], and random selection of a subset of columns of the kernel matrix [Smola and Schölkopf, 2000].

For kernels with an infinite number of derived features, such as the Gaussian RBF, one can still take advantage of explicit feature maps. For example, one may think about approximating the kernel function itself by truncating the spectral representation

$$K(x_1, x_2) \approx \sum_{j=1}^{n} \lambda_j \phi_j(x_1)\phi_j(x_2).$$

This is equivalent to using an approximate factorization of the kernel matrix, where $\widetilde{x}_{ij} = \sqrt{\lambda_j}\phi_j(x_i)$. The approximation error as a function of n decreases at the same rate of the eigenvalues λ_i. A spectral representation for Gaussian RBF kernels can be found in [Steinwart et al., 2006].

Instead of approximating the spectral feature map (definition 7), one can also approximate the stochastic process feature map (definition 8). Indeed, the covariance can be empirically approximated by randomly drawing a finite number of realizations $\mathbf{U}^j$ $(j = 1, \ldots, n)$ of the associated zero-mean stochastic process $\mathbf{U}$, and then computing the empirical covariance:

$$K(x_1, x_2) = \operatorname{cov}\left(\mathbf{U}_{x_1}, \mathbf{U}_{x_2}\right) \approx \sum_{j=1}^{n} \frac{\mathbf{U}^j_{x_1}\mathbf{U}^j_{x_2}}{n},$$

which is equivalent to using an approximate factorization of the kernel matrix, where $\widetilde{x}_{ij} = \mathbf{U}^j_{x_i}/\sqrt{n}$.

For continuous and translation invariant positive semi-definite kernels on $\mathbb{R}^d$, one can also approximate the Fourier feature map (subsection 1.4.4) by drawing random samples $\mathbf{U}^j$ $(j = 1, \ldots, n)$ from the distribution μ given by the inverse Fourier transform of R/α, see [Rahimi and Recht, 2008]. Then, letting $\mathbf{Z}_{\mathbf{U}}(x)$ as in equation (1.9), the Fourier feature map can be approximated as follows:

$$K(x_1, x_2) = \operatorname{cov}\left(\mathbf{Z}_{\mathbf{U}}(x_1), \mathbf{Z}_{\mathbf{U}}(x_2)\right) \approx \sum_{j=1}^{n} \frac{\mathbf{Z}^T_{\mathbf{U}^j}(x_1)\mathbf{Z}_{\mathbf{U}^j}(x_2)}{n},$$

which amounts to approximately factorize the kernel matrix using

$$\widetilde{x}_{i(2j-1)} = \sqrt{\frac{\alpha}{n}}\cos(\langle \mathbf{U}, x\rangle_2), \quad \widetilde{x}_{i(2j)} = \sqrt{\frac{\alpha}{n}}\sin(\langle \mathbf{U}, x\rangle_2), \quad j = 1, \ldots, n.$$

1.6.4 Sparse kernels

Another possibility to improve computational performances when computing weighted kernel combination is to explicitly introduce sparsity in the kernel expansion by using kernels that take the value zero on a certain subset of the input pairs. The idea has been extensively developed in the case of radial basis function kernels on $\mathbb{R}^n$, see [Gneiting, 2002] and references therein. For instance, [Genton, 2001] reports that a general class of compactly supported RBF kernels can be obtained by multiplying a compactly supported RBF function of the form

$$\psi(x_1, x_2) = (1 - \|x_1 - x_2\|)_+^{\bar{\nu}}, \qquad \bar{\nu} \geq (n+1)/2,$$

by RBF kernels of the Matérn type [Matérn, 1960], defined as

$$K(x_1, x_2) = \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{2\sqrt{\nu}\|x_1 - x_2\|}{\theta} \right)^{\nu} H_\nu \left(\frac{2\sqrt{\nu}\|x_1 - x_2\|}{\theta} \right),$$

where Γ is the Gamma function, and H_ν is the modified Bessel function of the second kind of order ν. Observe that Matérn kernels reduces to the Gaussian kernel for $\nu + \infty$. Several other examples of compactly supported radial basis functions have been introduced in the literature, see e.g. [Schaback, 1995, Wendland, 1995].

1.6.5 Decomposition methods

Besides the two extreme approaches of *pre-computing* the kernel matrix and kernel computation *on-the-fly*, a whole spectrum of intermediate solutions have been proposed to practically address the trade-off between space and time. In order to overcome memory limitations in support vector machines training, researchers introduced a variety of *decomposition methods* [Osuna et al., 1997, Joachims, 1998, Keerthi et al., 2001], which involve iteratively choosing a subset of the training data, called *working set*, and solving the corresponding subproblem of reduced size for which the corresponding entries of the kernel matrix do fit into the memory. To avoid recomputing many times the same entries of the kernel matrix, one can maintain a cache of kernel values that are likely to be used in the next iterations. Convergence of decomposition methods has been studied in [Lin, 2001, 2002, List and Simon, 2004, 2007, Palagi and Sciandrone, 2005, Lucidi et al., 2007]. The popular SMO (Sequential Minimal Optimization) algorithm [Platt, 1998] is an instance of decomposition method for SVM training. The working set size for SMO is two, which is minimal for the original formulation of SVM, that included an constant bias term. Different working set selection rules for SMO have been introduced in [Keerthi et al., 2001, Hush and Scovel, 2003, Fan et al., 2005, Glasmachers and Igel, 2006]. Convergence of SMO has been studied in [Keerthi and Gilbert, 2002, Lin, 2002, Chen et al., 2006]. A major complication in the analysis of classical SVMs comes from the presence of an unregularized constant bias term, which corresponds to an additional equality constraint in the optimality conditions. Recently, it is becoming customary to formulate SVM without unregularized bias, see e.g. [Fan et al., 2008]. In this case, the minimal size of the working set is one, which leads to

coordinate descent algorithms. In this thesis, we present a general class of coordinate descent algorithms for regularized kernel methods with generic convex loss function.

1.7 Contribution of this thesis

In this thesis, we present some extensions of classical framework of kernel machines, and discuss a new variational approach to analyze optimization algorithms for large scale learning problems. The major contributions can be grouped as follows:

- Analysis of optimization algorithms for large scale kernel methods.
- Kernel machines with two layers.
- Kernel methods for multi-task learning.

1.7.1 Optimization for large scale kernel methods

We study two general classes of optimization algorithms for kernel methods with convex loss function and quadratic norm regularization, and analyze their convergence. The first approach, based on fixed-point iterations, is simple to implement and analyze, and can be easily parallelized. The second, based on coordinate descent, exploits the structure of additively separable loss functions to compute solutions of line searches in closed form. Instances of these classes of algorithms are already incorporated into state of the art machine learning software for large scale problems. We start from a solution characterization of the regularized problem, obtained using sub-differential calculus and resolvents of monotone operators, that holds for general convex loss functions regardless of differentiability. The two methodologies can be regarded as instances of nonlinear Jacobi and Gauss-Seidel algorithms, and are both well-suited to solve large scale problems.

1.7.2 Kernel machines with two layers

In recent years, a variety of techniques such as semi-definite programming, hyper-kernels, and multiple kernel learning (MKL), have been used to address the problem of learning the kernel from the data. Kernel machines with two layers extend the classical framework of regularization over Reproducing Kernel Hilbert Spaces (RKHS), by modeling the unknown relationship between input and outputs as the composition of two functional layers. A suitable representer theorem for kernel machines with two layers shows that a general class of regularization problems boils down to finite-dimensional optimization problems independently of the dimensionality of the search spaces. It is shown that MKL can be interpreted as a particular instance of kernel machine with two layers in which the second layer is a linear function. Finally, a simple and effective MKL method called RLS2 (regularized least squares with two layers) is introduced,

and his performances on several learning problems are extensively analyzed. An open source MATLAB toolbox to train and validate RLS2 models with a Graphic User Interface is available at http://www.mloss.org.

1.7.3 Kernel methods for multi-task learning

Multi-task learning is the simultaneous solution of multiple related learning problems by joint analysis of several datasets. There's theoretical and experimental evidence suggesting that simultaneous solution of multiple related learning tasks performs better than independent (single-task) learning. In this thesis, we present several results related to a general class of mixed-effect models where each functional relation is modeled as the sum of a common function plus an individual shift:

$$g_j(x) = \overline{g}(x) + \widetilde{g}_j(x).$$

Mixed-effect models already have a well-established role in the solution of PK-PD (pharmacokinetics and pharmacodynamics) data analysis problems, whose goal is to estimate responses of different subjects to a drug administration from multiple tiny sets of individual measurements. We describe a client-server architecture for on-line learning of multiple tasks based on the non-parametric mixed-effect model. Interestingly, it turns out that such client-server architecture also enjoys other desirable properties such as the ability to perform distributed computations, and preserve privacy of individual data.

Bibliography

A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In Peter Auer and Ron Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 338–352. Springer Berlin / Heidelberg, 2005.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd Annual international conference on Machine learning (ICML 2005)*, pages 33–40. ACM Press, 2005.

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors. *Large Scale Kernel Machines*. MIT Press, Cambridge, MA, USA, 2007.

Y-W. Chang, C-J. Hsieh, K-W. Chang, M. Ringgaard, and C-J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11:1471–1490, 2010.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, USA, 2006.

P-H Chen, R-E Fan, and C-J Lin. A study on SMO-type decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 17(4): 893–908, 2006.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39:1–49, 2001.

P. Drineas and M. W. Mahoney. On the Nystrom method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

R. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

R-E Fan, P-H Chen, and C-J Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6, 2005.

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

M. G. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

T. Glasmachers and C. Igel. Maximum-gain working set selection for SVMs. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

T. Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508, 2002.

D. Hush and C. Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51(1):51–71, 2003.

T. Joachims. *Advances in Kernel Methods: Support Vector Machines*, chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1998.

S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46(1–3):351–360, 2002.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

B. Kulis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. In *Proceedings of the 23rd Annual international conference on Machine learning (ICML 2006)*, pages 505–512, 2006.

C-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001.

C-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 13:1045–1052, 2002.

N. List and H. U. Simon. A general convergence theorem for the decomposition method. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 363–377. Springer Berlin / Heidelberg, 2004.

N. List and H. U. Simon. General polynomial time decomposition algorithms. *Journal of Machine Learning Research*, 8:303–321, 2007.

S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone. A convergent decomposition algorithm for support vector machines. *Computational Optimization and Applications*, 38:217–234, 2007.

B. Matérn. *Spatial Variation*. Springer, New York, NY, USA, 1960.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London*, 209:415–446, 1909.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005.

C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *Proceedings of the 21th Annual international conference on Machine learning (ICML 2004)*, page 81, New York, NY, USA, 2004. ACM.

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. Osuna, Freund. R., and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of Computer Vision and Pattern Recognition*, pages 130–136, 1997.

L. Palagi and M. Sciandrone. On the convergence of a modified version of the svmlight algorithm. *Optimization Methods and Software*, 20:315–332, 2005.

J. Platt. Fast training of support vector machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA, 1998.

T. Poggio and F. Girosi. Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. MIT Press, Cambridge, MA, USA, 2008.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

W. Rudin. *Fourier Analysis on Groups*. Wiley-Interscience, New York, NY, USA, 1994.

S. Saitoh. *Theory of Reproducing Kernels and its Applications*, volume 189 of *Pitman Research Notes in Mathematics Series*. Longman Scientific and Technical, Harlow, 1988.

R. Schaback. Creating surfaces from scattered data using radial basis functions. In T. Lyche M. Dhlen and L.L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design III*, pages 477–496. Vanderbilt Univ. Press, 1995.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. (Adaptive Computation and Machine Learning). MIT Press, 2001.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

L. Schwartz. Sous-espaces hilbertiens d'espaces vectoriels topologiques et noyaux associés (noyaux reproduisants). *J. Analyse Math.*, 13:115–256, 1964.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th Annual international conference on Machine learning (ICML 2000)*, pages 911–918, 2000.

I. Steinwart, D. R. Hush, and C. Scovel. An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. *IEEE Transactions on Information Theory*, 52(10):4635–4643, 2006.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D. C., 1977.

I. Tsochantaridis. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, USA, 1998.

G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, USA, 1990.

H. Wendland. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics*, pages 389–396, 1995.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Proceedings of the 13th Annual Conference on Neural Information Processing Systems*, pages 682–688, Whistler, BC, Canada, 2000.

2

Optimization for large scale regularized kernel methods

The development of optimization software for learning from large datasets is heavily influenced by memory hierarchies of computer storage. In presence of memory constraints, most of the high order optimization methods become unfeasible, whereas techniques such as coordinate descent or stochastic gradient descent may exploit the specific structure of learning functionals to scale well with the dataset size. One of the most important features of modern machine learning methodologies is the ability to leverage on sparsity in order to obtain scalability. Typically, learning methods that impose sparsity are based on the minimization of non-differentiable objective functionals. Is this the case of support vector machines or methods based on ℓ_1 regularization.

In this chapter, we analyze optimization algorithms for a general class of regularization functionals, using sub-differential calculus and resolvents of monotone operators [Rockafellar, 1970, Hiriart-Urruty and Lemaréchal, 2004] to manage non-differentiability. In particular, we study learning methods that can be interpreted as the minimization of a convex empirical risk term plus a squared norm regularization into a reproducing kernel Hilbert space [Aronszajn, 1950] $\mathcal{H}_K$ with non-null reproducing kernel K, namely

$$\min_{g \in \mathcal{H}_K} \left(f\left(g(x_1), \ldots, g(x_\ell)\right) + \frac{\|g\|^2_{\mathcal{H}_K}}{2} \right), \tag{2.1}$$

where $f : \mathbb{R}^\ell \to \mathbb{R}_+$ is a finite-valued bounded below convex function. Regularization problems of the form (2.1) admit a unique optimal solution which, in view of the representer theorem [Schölkopf et al., 2001], can be represented as a finite linear combination of kernel sections:

$$g(x) = \sum_{i=1}^{\ell} c_i K_{x_i}(x).$$

We characterize optimal coefficients c_i of the linear combination via a family of non-linear equations. Then, we introduce two general classes of optimization algorithms for large scale regularization methods that can be regarded as instances of non-linear Jacobi and Gauss-Seidel algorithms, and analyze their

convergence properties. Finally, we state a theorem that shows how to reformulate convex regularization problems, so as to trade off positive semidefiniteness of the kernel matrix for differentiability of the empirical risk.

2.1 Solution characterization

As a consequence of the representer theorem, an optimal solution of problem (2.1) can be obtained by solving finite-dimensional optimization problems of the form

$$\min_{c\in\mathbb{R}^\ell} F(c), \qquad F(c) = f(\mathbf{K}c) + \frac{c^T\mathbf{K}c}{2}, \tag{2.2}$$

where $\mathbf{K} \in \mathbb{R}^{\ell\times\ell}$ is a non-null symmetric positive semi-definite matrix called *kernel matrix*. The entries k_{ij} of the kernel matrix are given by

$$k_{ij} = K(x_i, x_j),$$

where $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semidefinite kernel function. Let k_i denote the columns of the kernel matrix ($i = 1, \ldots, \ell$). Particularly interesting is the case when function f is additively separable.

Definition 15 (Additively separable functional)**.** A functional $f : \mathbb{R}^\ell \to \mathbb{R}$ is called *additively separable* if

$$f(z) = \sum_{i=1}^{\ell} f_i(z_i). \tag{2.3}$$

Models with ℓ_2 (ridge) regularization correspond to the case in which inputs are n-dimensional numeric vectors ($\mathcal{X} = \mathbb{R}^n$) and the kernel matrix is chosen as $\mathbf{K} = \mathbf{X}\mathbf{X}^T$, where $\mathbf{X} \in \mathbb{R}^{\ell\times n}$ is a matrix whose rows are the input data x_i. Letting

$$w := \mathbf{X}^T c, \tag{2.4}$$

the following class of problems is obtained:

$$\min_{w\in\mathbb{R}^n} \left(f(\mathbf{X}w) + \frac{\|w\|_2^2}{2} \right). \tag{2.5}$$

Observe that one can optimize over the whole space $\mathbb{R}^n$, since the optimal weight vector will automatically be in the form (2.4). Parametric models with ℓ_2 regularization can be seen as specific instances of kernel methods in which K is the linear kernel:

$$K(x_1, x_2) = \langle x_1, x_2\rangle_2.$$

In the following, two key mathematical objects will be used to characterize optimal solutions of problems (2.2) and (2.5). The first is the subdifferential ∂f

of the empirical risk. The second is the resolvent J_α of the inverse subdifferential, defined as

$$J_\alpha := \left(\mathbf{I} + \alpha\,(\partial f)^{-1}\right)^{-1}. \tag{2.6}$$

See the appendix for more details about these objects. The following result characterizes optimal solutions of problem (2.2) via a non-linear equation involving J_α. The characterization also holds for non-differentiable loss functions, and is obtained without introducing constrained optimization problems. The proof of Theorem 8 is given into the appendix.

Theorem 8. *For any $\alpha > 0$, there exist optimal solutions of problem (2.2) such that*

$$c = -J_\alpha(\alpha \mathbf{K} c - c), \tag{2.7}$$

where J_α is the resolvent of the inverse sub-differential $(\partial f)^{-1}$, see (2.6).

The usefulness of condition (2.7) depends on the possibility of computing closed-form expressions for the resolvent, which may not be feasible for general convex functionals. Remarkably, for many learning methods one can typically exploit the specific structure of f to work out closed-form expressions. For instance, when f is additively separable as in (2.3), the sub-differential decouples with respect to the different components. In such a case, the computation of the resolvent reduces to the inversion of a function of a single variable, which can be often obtained in closed form. Indeed, additive separability holds for many supervised learning problems. Typically, we have $f_i(z_i) = CL(y_i, z_i)$, where $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is a loss function, and $C > 0$ is a complexity parameter. Table 2.1 reports the expression of the J_α in correspondence with commonly used loss functions. When f is additively separable, the characterization (2.7) can be generalized as follows.

Corollary 2. *Assume that (2.3) holds. Then, for any $\alpha_i > 0$, $i = 1, \ldots, \ell$, there exist optimal solutions of problem (2.2) such that*

$$c_i = -J^i_{\alpha_i}(\alpha_i k_i^T c - c_i), \qquad i = 1, \ldots, \ell, \tag{2.8}$$

where $J^i_{\alpha_i}$ are the resolvents of the inverse sub-differentials $(\partial f_i)^{-1}$, see (2.6).

In this chapter, we analyze two iterative approaches to compute optimal solutions of problem (2.2), based on the solution characterizations of Theorem 8 and Corollary 2. For both methods, we show that cluster points of the iteration sequence are optimal solutions, and we have

$$\min_{c \in \mathbb{R}^\ell} F(c) = \lim_{k \to +\infty} F(c^k), \tag{2.9}$$

where F denote the functional of problem (2.2). Section 2.2 describes a first approach, which involves simply iterating equation (2.7) according to the fixed-point method. The method can be also regarded as a non-linear Jacobi algorithm to solve equation (2.7). It is shown that α can be always chosen so as to make the iterations approximate an optimal solution to arbitrary precision.

Name	Loss $L(y_1, y_2)$	Operator $-J_\alpha(v)$
L1-SVM	$(1 - y_1 y_2)_+$	$y \odot \min\left(C, (\alpha e - y \odot v)_+\right)$
L2-SVM	$(1 - y_1 y_2)_+^2 / 2$	$y \odot (\alpha e - y \odot v)_+ / (1 + \alpha / C)$
RLS	$(y_1 - y_2)^2 / 2$	$(\alpha y - v) / (1 + \alpha / C)$
RLA	$\lvert y_1 - y_2 \rvert$	$\operatorname{sign}(y - v) \odot \min(C, \lvert y - v \rvert)$
SVR	$(\lvert y_1 - y_2 \rvert - \epsilon)_+$	$\operatorname{sign}(y - v) \odot \min\left(C, (\lvert y - v \rvert - \alpha\epsilon e)_+\right)$

Table 2.1: Operator $-J_\alpha$ for different methods. In the rightmost column, $\odot$ denotes the element-wise product, e denotes the vector of all ones, and all the functions min, max, sign, $|\cdot|$ are applied component-wise. The "positive part" function is defined as $(x)_+ = \max\{0, x\}$.

In section 2.3, we describe a second approach, that involves separately iterating the single components using the characterization of equation (2.8). For a suitable choice of α_i, the method boils down to coordinate descent, and optimality of cluster points holds whenever indices are picked according to an "essentially cyclical" rule. Equivalently, the method can be regarded as a non-linear Gauss-Seidel algorithm to solve (2.8).

2.2 Fixed-point algorithms

In this section, we suggest computing the optimal coefficient vector c of problem (2.2) by simply iterating equation (2.7), starting from any initial condition c^0:

$$c^{k+1} = -J_\alpha(\alpha \mathbf{K} c^k - c^k). \tag{2.10}$$

Such procedure is the well-known fixed point iteration (also known as Picard or Richardson iteration) method. Provided that α is properly chosen, the procedure can be used to solve problem (2.2) to any given accuracy. Before analyzing the convergence properties of method (2.10), let's study the computational complexity of a single iteration. To this end, one can decompose the iteration into three intermediate steps:

$$\begin{aligned} z^k &= \mathbf{K} c^k, && \textbf{step 1} \\ v^k &= \alpha z^k - c^k, && \textbf{step 2} \\ c^{k+1} &= -J_\alpha(v^k). && \textbf{step 3} \end{aligned}$$

The decomposition emphasize the separation between the role of the kernel (affecting only step 1) and the role of the function f (affecting only step 3).

Step 1

Step one is the only one that involves the kernel matrix. Generally, it is also the most computationally and memory demanding step. Since $z = \mathbf{K} c$ represents predictions on training inputs (or a quantity related to them), it holds that being able to perform fast predictions has a crucial impact also on the training

time. This is remarkable, since good prediction speed is a desirable goal on its own. Notice that an efficient implementation of the prediction step is beneficial for any learning method of the form (2.2), independently of f. Ideally, the computational cost of such matrix-vector multiplication is $O(\ell^2)$. However, the kernel matrix might not fit into the memory, so that the time needed to compute the product might also include special computations or additional I/O operations. Observe that, if many components of vector c are null, only a subset of the rows of the kernel matrix is necessary in order to compute the product. Hence, methods that impose sparsity in vector c may produce a significant speed-up in the prediction step. As an additional remark, observe that the matrix-vector product is an operation that can be easily parallelized.

In the linear case (2.5), the computation of z^k can be divided in two parts:

$$w^k = \mathbf{X}^T c^k,$$
$$z^k = \mathbf{X} w^k.$$

In order to compute the product, it is not even necessary to form the kernel matrix, which may yields a significant memory saving. The two intermediate products both need $O(n\ell)$ operations and the overall cost still scales with $O(n\ell)$. When the number of features is much lower than the number of examples ($n \ll \ell$), there's a significant improvement with respect to $O(\ell^2)$. Speed-up and memory saving are even more dramatic when $\mathbf{X}$ is sparse. In such a case, computing the product in two steps might be more convenient also when $n > \ell$.

Step 2

Step two is a simple subtraction between vectors, whose computational cost is $O(\ell)$. In section 2.4, it is shown that $v = \alpha \mathbf{K} c - c$ can be interpreted as the vector of predictions on the training inputs associated with another learning problem consisting in stabilizing a functional regularized whose empirical risk is always differentiable, and whose kernel is not necessarily positive.

Step 3

Step three is the only one that depends on function f. Hence, different algorithms can be implemented by simply choosing different resolvents J_α. Table 2.1 reports the loss function L and the corresponding resolvent for some common supervised learning methods. Some examples are given below. Consider problem (2.2) with the "hinge" loss function $L(y_1, y_2) = (1 - y_1 y_2)_+$, associated with the popular Support Vector Machine (SVM). For SVM, step three reads

$$c^{k+1} = y \odot \min\left(C, \left(1 - y \odot v^k\right)_+\right),$$

where $\odot$ denotes the element-wise product, and min is applied element-wise. As a second example, consider classic regularized least squares (RLS). In this case, step three reduces to

$$c^{k+1} = \frac{\alpha y - v^k}{1 + \alpha / C}.$$

Generally, the complexity of step three is $O(\ell)$ for any of these classical loss functions.

2.2.1 Convergence

The following result states that the sequence generated by the iterative procedure (2.10) can be used to approximately solve problem (2.2) to any precision, provided that α is suitably chosen.

Theorem 9. *If the sequence c^k is generated according to algorithm (2.10), and*

$$0 < \alpha < \frac{2}{\|\mathbf{K}\|_2}, \tag{2.11}$$

then (2.9) holds. Moreover, c^k is bounded, and any cluster point is a solution of problem (2.2).

A stronger convergence result holds when the kernel matrix is strictly positive or f is differentiable with Lipschitz continuous gradient. Under these conditions, it turns out that the whole sequence c^k converges at least linearly to an unique fixed point.

Theorem 10. *Suppose that the sequence c^k is generated according to algorithm (2.10), where α satisfy (2.11), and one of the following conditions holds:*

1. *The kernel matrix $\mathbf{K}$ is positive definite.*

2. *Function f is everywhere differentiable and ∇f is Lipschitz continuous,*

Then, there exists a unique solution c^* of equation (2.7), and c^k converges to c^* with the following rate

$$\|c^{k+1} - c^*\|_2 \leq \mu \|c^k - c^*\|_2, \qquad 0 \leq \mu < 1.$$

In practice, condition (2.11) can be equivalently satisfied by fixing $\alpha = 1$ and scaling the kernel matrix to have spectral norm between 0 and 2. In problems that involve a regularization parameter, this last choice will only affect its scale. A possible practical rule to choose the value of α is $\alpha = 1/\|\mathbf{K}\|_2$, which is equivalent to scale the kernel matrix to have spectral norm equal to one. However, in order to compute the scaling factor in this way, one generally needs all the entries of the kernel matrix. A cheaper alternative that uses only the diagonal entries of the kernel matrix is $\alpha = 1/\mathrm{tr}(\mathbf{K})$, which is equivalent to fix α to one and normalizing the kernel matrix to have trace one. To see that this last rule satisfy condition (2.11), observe that the trace of a positive semidefinite matrix is an upper bound for the spectral norm. In the linear case (2.5), one can directly compute α on the basis of the data matrix $\mathbf{X}$. In particular, we have $\|\mathbf{K}\|_2 = \|\mathbf{X}\|_2^2$, and $\mathrm{tr}(\mathbf{K}) = \|\mathbf{X}\|_F^2$, where $\|\cdot\|_F$ denotes the Frobenius norm.

2.3 Coordinate-wise iterative algorithms

In this section, we describe a second optimization approach that can be seen as a way to iteratively enforce optimality condition (2.8). Throughout the section, it is assumed that f is additively separable as in (2.3). In view of Corollary 2, the optimality condition can be rewritten for a single component as in (2.8). Consider the following general update algorithm:

$$c_i^{k+1} = -J_{\alpha_i}^i(\alpha_i k_i^T c^k - c_i^k), \qquad i = 1, \dots, \ell. \tag{2.12}$$

A serial implementation of algorithm (2.10) can be obtained by choosing $\alpha_i = \alpha$ and by cyclically computing the new components c_i^{k+1} according to equation (2.12). Observe that this approach requires to keep in memory both c^k and c^{k+1} at a certain time. In the next sub-section, we analyze a different choice of parameters α_i that leads to a class of coordinate descent algorithms, based on the principle of using new computed information as soon as it is available.

2.3.1 Coordinate descent methods

Algorithm 1 Coordinate descent for regularized kernel methods

while $\max_i |h_i| \geq \delta$ **do**
  Pick a coordinate index i according to some rule,
  $z_i^k = k_i^T c^k$,
  $v_i^k = z_i^k / k_{ii} - c_i^k$,
  $\text{tmp} = S_i(v_i^k)$,
  $h_i = \text{tmp} - c_i^k$,
  $c_i^{k+1} = \text{tmp}$,
end while

A coordinate descent algorithm updates a single variable at each iteration by solving a sub-problem of dimension one. During the last years, optimization via coordinate descent is becoming a popular approach in machine learning and statistics, since its implementation is straightforward and enjoys favorable computational properties [Friedman et al., 2007, Tseng and Yun, 2008, Wu and Lange, 2008, Chang et al., 2008, Hsieh et al., 2008, Yun and Toh, 2009, Huang et al., 2010, Friedman et al., 2010]. Although the method may require many iterations to converge, the specific structure of supervised learning objective functionals allows to solve the sub-problems with high efficiency. This makes the approach competitive especially for large-scale problems, in which memory limitations hinder the use of second order optimization algorithms. As a matter of fact, state of the art solvers for large scale supervised learning such as `glmnet` [Friedman et al., 2010] for generalized linear models, or `LIBLINEAR` [Fan et al., 2008] for SVMs are based on coordinate descent techniques.

The update for c_i^k in algorithm (2.12) also depends on components c_j^k with $j < i$ which have already been updated. Hence, one needs to keep in memory coefficients from two subsequent iterations c^{k+1} and c^k. In this sub-section, we describe a method that allows to take advantage of the computed information

as soon as it is available, by overwriting the coefficients with the new values. Assume that the diagonal elements of the kernel matrix are strictly positive, i.e. $k_{ii} > 0$. Notice that this last assumption can be made without any loss of generality. Indeed, if $k_{ii} = 0$ for some index i then, in view of the inequality $|k_{ij}| \leq \sqrt{k_{ii}k_{jj}}$, it follows that $k_{ij} = 0$ for all j. Hence, the whole i-th column (row) of the kernel matrix is zero, and can be removed without affecting optimization results for the other coefficients. By letting $\alpha_i = 1/k_{ii}$ and $S_i := -J^i_{(k_{ii})^{-1}}$ in equation (2.8), the i-th coefficient in the inner sum does cancel out, and we obtain

$$c_i = S_i \left(\sum_{j \neq i} \frac{k_{ij}}{k_{ii}} c_j \right). \tag{2.13}$$

The optimal i-th coefficient is thus expressed as a function of the others. Similar characterizations have been also derived in [Dinuzzo and De Nicolao, 2009] for several loss functions. Equation (2.13) is the starting point to obtain a variety of coordinate descent algorithms involving the iterative choice of a a coordinate index i followed by the optimization of c_i as a function of the other coefficients. A simple test on the residual of equation (2.13) can be used as a stopping condition. The approach can be also regarded as a non-linear Gauss-Seidel method [Ortega and Rheinboldt, 2000] for solving the equations (2.8). It is assumed that vector c is initialized to some initial c^0, and coefficients h_i are initialized to the residuals of equation (2.13) evaluated in correspondence with c^0. Remarkably, in order to implement the method for different loss functions, we simply need to modify the expression of functions S_i. Each update only involves a single row (column) of the kernel matrix. In the following, we will assume that indices are recursively picked according to a rule that satisfy the following condition, see [Tseng, 2001, Luenberger and Ye, 2008].

Essentially Cyclic Rule. There exists a constant integer $T > \ell$ such that every index $i \in \{1, \ldots, \ell\}$ is chosen at least once between the k-th iteration and the $(k + T - 1)$-th, for all k.

Iterations of coordinate descent algorithms that use an essentially cyclic rule can be grouped in *macro-iterations*, containing at most T updates of the form (2.13), within which all the indices are picked at least once. Below, we report some simple rules that satisfy the essentially cyclic condition and don't require to maintain any additional information (such as the gradient):

1. **Cyclic rule:** In each macro-iteration, each index is picked exactly once in the order $1, \ldots, \ell$. Hence, each macro-iteration consists exactly of ℓ iterations.

2. **Aitken double sweep rule:** Consists in alternating macro-iterations in which indices are chosen in the natural order $1, \ldots, \ell$ with macro-iterations in the reverse order, i.e. $(\ell - 1), \ldots, 1$.

3. **Randomized cyclic rule:** The same as the cyclic rule, except that indices are randomly permuted at each macro-iteration.

In the linear case (2.5), z_i^k can be computed as follows

$$w^k = \mathbf{X}c^k,$$
$$z_i^k = x_i^T w^k.$$

By exploiting the fact that only one component of vector c changes from an iteration to the next, the first equation can be further developed:

$$w^k = \mathbf{X}^T c^k = w^{k-1} + (\mathbf{X}^T e_p)h_p = w^{k-1} + x_p h_p$$

where p denotes the index chosen in the previous iteration, and h_p denotes the variation of coefficient c_p in the previous iteration. By introducing these new quantities, the coordinate descent algorithm can be rewritten as in Algorithm 2, where we have set $S_i := -J^i_{\|x_i\|_2^{-2}}$.

Algorithm 2 Coordinate descent (linear kernel)

while $\max_i |h_i| \geq \delta$ **do**
  Pick a coordinate index i according to some rule,
  if $h_p \neq 0$ **then**
    $w^k = w^{k-1} + x_p h_p$,
  end if
  $z_i^k = x_i^T w^k$,
  $v_i^k = z_i^k / \|x_i\|_2^2 - c_i^k$,
  $\text{tmp} = S_i(v_i^k)$,
  $h_i = \text{tmp} - c_i^k$,
  $c_i^{k+1} = \text{tmp}$,
  $p = i$
end while

The computational cost of a single iteration depends mainly on the updates for w and z_i, and scales linearly with the number of features, i.e. $O(n)$. When the loss function has linear traits, it is often the case that coefficient c_i does not change after the update, so that $h_i = 0$. When this happen, the next update of w can be skipped, obtaining a significant speed-up. Further, if the vectors x_i are sparse, the average computational cost of the second line may be much lower than $O(n)$. A technique of this kind has been proposed in [Hsieh et al., 2008] and implemented in the package `LIBLINEAR` [Fan et al., 2008] to improve speed of coordinate descent iterations for linear SVM training. Here, one can see that the same technique can be applied to any convex loss function, provided that an expression for the corresponding resolvent is available.

2.3.2 Convergence

The main convergence result for coordinate descent is stated below. It should be observed that the classical theory of convergence for coordinate descent is typically formulated for differentiable objective functionals. When the objective functional is not differentiable, there exist counterexamples showing that the method may get stuck in a non-stationary point [Auslender, 1976]. In the non-differentiable case, optimality of cluster points of coordinate descent iterations has been proven in [Tseng, 2001] (see also references therein), under the

additional assumption that the non-differentiable part is additively separable. Unfortunately, the result of [Tseng, 2001] cannot be directly applied to problem (2.2), since the (possibly) non-differential part $f(\mathbf{K}c)$ is not separable with respect to the optimization variables c_i, even when (2.3) holds. Notice also that, when the kernel matrix is not strictly positive, level sets of the objective functional are unbounded (see Lemma 12 in the appendix). Despite these facts, it still holds that cluster points of coordinate descent iterations are optimal, as stated by the next Theorem.

Theorem 11. *Suppose that the following conditions hold:*

1. *Function f is additively separable as in (2.3),*
2. *The diagonal entries of the kernel matrix satisfy $k_{ii} > 0$,*
3. *The sequence c^k is generated by the coordinate descent algorithm (Algorithm 1 or 2), where indices are recursively selected according to an essentially cyclic rule.*

Then, (2.9) holds, c^k is bounded, and any cluster point is a solution of (2.2).

2.4 A reformulation theorem

The following result shows that solutions of problem (2.2) satisfying equation (2.8) are also stationary points of a suitable family of differentiable functionals. See the appendix for the definition of the Moreau-Yosida regularization.

Theorem 12. *If c satisfy (2.7), then it is also a stationary point of the following functional:*

$$F_\alpha(c) = \alpha^{-1} f_\alpha(\mathbf{K}_\alpha c) + \frac{c^T \mathbf{K}_\alpha c}{2},$$

where f_α denotes the Moreau-Yosida regularization of f, and $\mathbf{K}_\alpha := \alpha\mathbf{K} - \mathbf{I}$.

Theorem 12 gives an insight into the role of parameter α, as well as providing an interesting link with machine learning with indefinite kernels. By the properties of the Moreau-Yosida regularization, f_α is differentiable with Lipschitz continuous gradient. It follows that F_α also have such property. Notice that lower values of α are associated with smoother functions f_α, while the gradient of $\alpha^{-1} f_\alpha$ is non-expansive. A lower value of α also implies a "less positive semidefinite" kernel, since the eigenvalues of $\mathbf{K}_\alpha$ are given by $(\alpha\alpha_i - 1)$, where α_i denote the eigenvalues of $\mathbf{K}$. Indeed, the kernel becomes non-positive as soon as $\alpha \min_i\{\alpha_i\} < 1$. Hence, the relaxation parameter α regulates a trade-off between smoothness of f_α and positivity of the kernel.

When f is additively separable as in (2.3), it follows that f_α is also additively separable:

$$f_\alpha(z) = \sum_{i=1}^{\ell} f_{i\alpha}(z_i),$$

and $f_{i\alpha}$ is the Moreau-Yosida regularization of f_i. The components can be often computed in closed form, so that an "equivalent differentiable loss function" can be derived for non-differentiable problems. For instance, when f_i is given by the hinge loss $f_i(z_i) = (1 - y_i z_i)_+$, letting $\alpha = 1$, we obtain

$$f_{i1}(z_i) = \begin{cases} 1/2 - y_i z_i, & y_i z_i \leq 0 \\ (1 - y_i z_i)_+^2/2, & y_i z_i > 0 \end{cases}$$

Observe that this last function is differentiable with Lipschitz continuous derivative. By Theorem 12, it follows that the SVM solution can be equivalently computed by searching the stationary points of a new regularization functional obtained by replacing the hinge loss with its equivalent differentiable loss function, and modifying the kernel matrix by subtracting the identity.

2.5 Conclusions

In this chapter, fixed-point and coordinate descent algorithms for regularized kernel methods with convex empirical risk and squared RKHS norm regularization have been analyzed. The two approaches can be regarded as instances of non-linear Jacobi and Gauss-Seidel algorithms to solve a suitable non-linear equation that characterizes optimal solutions. While the fixed-point algorithm has the advantage of being parallelizable, the coordinate descent algorithm is able to immediately exploit the information computed during the update of a single coefficient. Both classes of algorithms have the potential to scale well with the dataset size. Finally, it has been shown that minimizers of convex regularization functionals are also stationary points of a family of differentiable regularization functionals involving the Moreau-Yosida regularization of the empirical risk.

Bibliography

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

A. Auslender. *Optimisation Méthodes Numériques*. Masson, France, 1976.

K-W. Chang, C-J. Hsieh, and C-J. Lin. Coordinate descent method for large-scale L2-loss linear support vector machines. *Journal of Machine Learning Research*, 9:1369–1398, 2008.

F. Dinuzzo and G. De Nicolao. An algebraic characterization of the optimum of regularized kernel methods. *Machine Learning*, 74(3):315–345, 2009.

R. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

J. Friedman, T. Hastie, H. Hoefling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1): 1–22, 2010.

J-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer, 2 edition, 2004.

C. Hsieh, K.W. Chang, C.J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 408–415, Helsinki, Finland, 2008.

F-L Huang, C-J Hsieh, K-W Chang, and C-J Lin. Iterative scaling and coordinate descent methods for maximum entropy models. *Journal of Machine Learning Research*, 11:815–848, 2010.

D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. International series in operation research and management science. Springer, 2008.

J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Classics in Applied Mathematics. SIAM, 2000.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, USA, 1970.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3): 475–494, June 2001.

P. Tseng and S. Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, pages 1–28, 2008.

T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, 2(1):224–244, 2008.

S. Yun and K.-C. Toh. A coordinate gradient descent method for ℓ_1-regularized convex minimization. *Computational Optimization and Applications*, pages 1–35, 2009.

3

Kernel machines with two layers

In this chapter, we introduce the framework of *kernel machines with two layers*, that generalizes classical regularized kernel methods and provides also a flexible way to learn a kernel function from the data. Consider a generic architecture whose input-output behavior can be described as a function composition of two functions called *layers*

$$g = g_2 \circ g_1, \quad g_1 : \mathcal{X} \to \mathcal{Z}, \quad g_2 : \mathcal{Z} \to \mathcal{Y}, \tag{3.1}$$

where $\mathcal{X}$ is a generic set while $\mathcal{Z}$ and $\mathcal{Y}$ are two Hilbert spaces. We will formalize in a functional analytic setting the problem of learning simultaneously the two functions g_1 and g_2 from the data. Structures of the type (3.1) can be interpreted in different ways:

- In many data modeling problems, the outputs can be assumed to depend indirectly on the inputs through a set of unobserved factors. By using a structure with two layers of the form (3.1), one can embed prior knowledge about the dependence of both the factors on the inputs (function g_1), and the outputs on the factors (function g_2).
- Function g_1 can be interpreted as a transformation to pre-process the inputs in order to extract features which are useful for the purpose of predicting the output. Indeed, pre-processing of input examples is a customary step in many supervised learning problems. An architecture with two layers of the form (3.1) can be seen as a way to learn simultaneously a pre-processing function g_1, and the final output predictor g_2.

In section 3.1 we state a representer theorem for kernel machines with two layers: irrespectively of dimension of the hypothesis spaces for g_1 and g_2, finite linear combination of kernel functions on each layer are optimal computational architectures with respect to general regularization functionals. Remarkably, such representer theorem also imply that, upon training, architecture (3.1) can be equivalently regarded as a standard kernel machine in which the kernel function is learned from the data.

The problem of *learning the kernel* is receiving considerable attention in recent years, both from functional analytic point of view and from pure optimization

perspectives [Bach et al., 2004, Lanckriet et al., 2004, Ong et al., 2005, Micchelli and Pontil, 2005, Argyriou et al., 2005, Wu et al., 2007, Micchelli and Pontil, 2007]. Indeed, the difficulty of choosing a good hypothesis space with little available a-priori knowledge is significantly reduced when the kernel is also learned from the data. The flexibility of algorithms which learn the kernel makes also possible to address important issues such as feature selection, learning from heterogeneous sources of data, and multi-scale approximation.

After discussing the general result on the solution representation for two non-linear layers, the attention is focused on the case in which the second layer is linear. In section 3.2, we introduce a regularization framework that turns out to be equivalent to a general class of methods to learn simultaneously a predictor and the associated kernel as convex combination of basis kernels, sometimes called multiple kernel learning (MKL).

3.1 Kernel machines with two layers

A learning architecture with two layers can be formalized as a map $g : \mathcal{X} \to \mathcal{Y}$ expressed as a function composition as in equation (3.1). Introduce an RKHS $\mathcal{H}_1$ of $\mathcal{Z}$-valued functions over $\mathcal{X}$ and an RKHS $\mathcal{H}_2$ of $\mathcal{Y}$-valued functions over $\mathcal{Z}$, with kernel functions K^1 and K^2, respectively. Then, consider the following problem:

$$\min_{\substack{g_1 \in \mathcal{H}_1, \\ g_2 \in \mathcal{H}_2}} \left[f\left((g_2 \circ g_1)(x_1), \ldots, (g_2 \circ g_1)(x_\ell)\right) + \Omega_1(\|g_1\|_{\mathcal{H}_1}) + \Omega_2(\|g_2\|_{\mathcal{H}_2}) \right]. \quad (3.2)$$

Here, $f : \mathcal{Y}^\ell \to \mathbb{R}_+$ is a functions measuring the approximation of training data, while $\Omega_1, \Omega_2 : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{+\infty\}$ are two extended-valued proper non-decreasing functions that play the role of regularization terms. Problem 3.2 is outside the scope of standard representer theorems [Schölkopf et al., 2001] due to the presence of the composition $(g_2 \circ g_1)$. Nevertheless, it still holds that linear combinations of a finite number of kernel functions are optimal solutions, as soon as there exist minimizers.

Theorem 13. *If the functional of problem (3.2) admit minimizers, then there exist minimizers in the form*

$$g_1(x) = \sum_{i=1}^{\ell} K^1_{x_i}(x) c^1_i, \qquad g_2(z) = \sum_{i=1}^{\ell} K^2_{g_1(x_i)}(z) c^2_i.$$

Letting

$$K(x_1, x_2) := K^2(g_1(x_1), g_1(x_2)),$$

denote the equivalent input-output kernel*, there exists optimal learning architectures whose input-output map can be written as:*

$$g(x) = (g_2 \circ g_1)(x) = \sum_{i=1}^{\ell} K_{x_i}(x) c^2_i. \quad (3.3)$$

Theorem 13 is a *restriction* theorem: the search for solutions of problem (3.2) can be restricted to *kernel machines with two layers* involving a finite number of kernel functions, even when $\mathcal{H}_1$ and $\mathcal{H}_2$ are infinite dimensional spaces. Notice that Theorem 13 is not an existence theorem, since existence of minimizers is one of the hypotheses. As shown in the next sections, existence can be ensured under mild additional conditions on f_i, Ω_1, Ω_2. Under the general hypothesis of Theorem 13, uniqueness of minimizers is also not guaranteed, even when f is strictly convex. Notice also that Theorem 13 do admit the presence of optimal solutions not in the form of finite kernel expansions. However, if such solutions exist, then their projections over the finite dimensional span of kernel sections are optimal as well, so that one can restrict the attention to kernel machines with two layers also in this case. Finally, when Ω_1 and Ω_2 are strictly increasing, it holds that *every* minimizer can be expressed as a kernel machine with two layers.

3.2 MKL as a kernel machine with two layers

Theorem 13 shows that training an architecture with two layers is equivalent to training simultaneously a single-layer kernel network and the kernel function, see equation (3.3). In this section, it is shown that multiple kernel learning, consisting in simultaneous learning of a finite convex combination of kernels and the associated predictor, can be interpreted as a specific instance of kernel architecture with two layers. Introduce a set of m positive semidefinite kernels $\widetilde{K}_i$ over $\mathcal{X}$, called *basis kernels* and consider the following choice for $\mathcal{H}_1$ and $\mathcal{H}_2$.

- $\mathcal{H}_1$ is an RKHS of vector valued functions $g : \mathcal{X} \to \mathbb{R}^m$ associated with the matrix-valued kernel function K^1 such that

  $$K^1 = \operatorname{diag}\left\{\widetilde{K}_1, \ldots, \widetilde{K}_m\right\}.$$

- $\mathcal{H}_2$ is the RKHS of real valued functions $g : \mathbb{R}^m \to \mathbb{R}$ associated with the linear kernel

  $$K^2(z_1, z_2) = z_1^T \mathbf{S} z_2,$$

  where $\mathbf{S}$ is a diagonal scaling matrix:

  $$\mathbf{S} = \operatorname{diag}\{s_1, \ldots, s_m\} > 0.$$

For any $g \in \mathcal{H}_1$, let g^i, $(i = 1, \ldots, m)$ denote its components. Introduce the indicator function I of the interval $[0, 1]$ defined as

$$I(t) = \begin{cases} 0, & 0 \le t \le 1 \\ +\infty, & t > 1 \end{cases},$$

let $f : \mathbb{R}^\ell \to \mathbb{R}_+$ denote finite-valued convex empirical risk. In the following, we analyze the particular case of problem (3.2) in which Ω_1 is the Tikhonov regularizer and Ω_2 is the Ivanov regularizer:

$$\Omega_1(t) = \frac{t^2}{2}, \qquad \Omega_2(t) = I(t),$$

which are both convex functions:

$$\min_{\substack{g_1\in\mathcal{H}_1,\\ g_2\in\mathcal{H}_2}} \left[f\left((g_2\circ g_1)(x_1),\dots,(g_2\circ g_1)(x_\ell)\right) + \frac{\|g_1\|^2_{\mathcal{H}_1}}{2} + I\left(\|g_2\|_{\mathcal{H}_2}\right)\right]. \quad (3.4)$$

Since f is convex and g_2 is linear, the problem is separately convex in both g_1 and g_2. Apparently, the Ivanov regularization on g_2 is equivalent to imposing the constraint $\|g_2\|_{\mathcal{H}_2} \leq 1$. The next Theorem characterizes optimal solutions of problem (3.4).

Theorem 14. *There exist optimal solutions g_1 and g_2 of problem (3.4) in the form*

$$g_1^i(x) = s_i a_i \sum_{j=1}^{\ell} c_j \widetilde{K}_i(x_j, x), \qquad g_2(z) = z^T \mathbf{S} a.$$

Letting $d_i := s_i a_i^2$, optimal coefficients (c, d) solve the following problem

$$\min_{c\in\mathbb{R}^\ell, d\in\mathbb{R}^m} \left(f(\mathbf{K}c) + \frac{c^T\mathbf{K}c}{2}\right) \quad (3.5)$$

subject to

$$\mathbf{K}^k_{ij} = s_k \widetilde{K}_k(x_i, x_j), \qquad \mathbf{K} = \sum_{k=1}^{m} d_k \mathbf{K}^k, \qquad d_k \geq 0, \qquad \sum_{k=1}^{m} d_k \leq 1. \quad (3.6)$$

Finally, the solution of problem (3.4) can be written as in equation (3.3), where the kernel K satisfies

$$K(x,y) = \sum_{i=1}^{m} d_i K_i(x,y), \qquad K_i(x,y) = \sum_{j_1=1}^{\ell}\sum_{j_2=1}^{\ell} c_{j_1} c_{j_2} \widetilde{K}_i(x_{j_1}, x)\widetilde{K}_i(x_{j_2}, y). \quad (3.7)$$

Theorem 14 shows that the variational problem (3.4) for a two-layer kernel machine is equivalent to problem (3.5)-(3.6). The non-negativity constraints $d_k \geq 0$ produces a sparse selection of a subset of basis kernels. In standard formulations, multiple kernel learning problems feature the equality constraint $\sum_{k=1}^m d_k = 1$, instead of the inequality in (3.6). Nevertheless, Lemma 2 below shows that there always exist optimal solutions satisfying the equality, so that the two optimization problems are equivalent.

Lemma 2. *There exists an optimal vector d for Problem 3.5 satisfying the equality constraint*

$$\sum_{k=1}^{m} d_k = 1. \quad (3.8)$$

Lemma 2 completes the equivalence between the specific *kernel machines with two layers* obtained by solving (3.4) and MKL.

A few comments on certain degeneracies in problem (3.5)-(3.6) are in order. First of all, observe that the absolute value of optimal coefficients a_i characterizing the kernel machine with two layers is given by $|a_i| = \sqrt{d_i/s_i}$, but $\text{sign}(a_i)$ is undetermined. Then, without loss of generality, it is possible to choose $a_i = \sqrt{d_i/s_i}$. Second, observe that the objective functional in (3.5) depends on c through the product $z = \mathbf{K}c$. When $\mathbf{K}$ is singular, the optimal vector c is not unique (independently of f). In particular, if u belongs to the nullspace of $\mathbf{K}$, then $c + u$ achieves the same objective value of c. One possible way to solve the indetermination, again without any loss of generality, is to constrain c to belong to the range of the kernel matrix. With such additional constraint, there exists v such that $c = \mathbf{K}^{\dagger}v$, where $\dagger$ denote the Moore-Penrose pseudo-inverse (notice that, in general, v might be different from z).

Finally, the following Lemma gives another important insight into the structure of problem (3.5)-(3.6).

Lemma 3. *Letting $z = \mathbf{K}c$, problem (3.5)-(3.6) can be rewritten as*

$$\min_{z \in \mathbb{R}^{\ell}, d \in \mathbb{R}^{m}} \left(f(z) + h(z, \mathbf{K}) \right), \quad \textit{subject to} \quad (3.6), \tag{3.9}$$

where

$$h(z, \mathbf{K}) = \begin{cases} (z^T \mathbf{K}^{\dagger} z)/2, & z \in \textit{range}(\mathbf{K}) \\ +\infty, & \textit{otherwise} \end{cases}$$

Problem (3.9) is a convex optimization problem.

Observe that, under the hypothesis of Lemma 3, it follows that local minimizers in (3.5) are also global minimizers.

3.3 Conclusions

This chapter introduces the framework of kernel machines with two layers, a general class of computational architectures that generalize classical regularized kernel methods. These architectures can be also seen as a general way to learning a kernel function from the data. Algorithms that learn the kernel as a convex combination of basis kernels (multiple kernel learning) are shown to be a subclass of kernel machines with two layers.

Bibliography

A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In Peter Auer and Ron Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 338–352. Springer Berlin / Heidelberg, 2005.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21th Annual international conference on Machine learning (ICML 2004)*, page 6, New York, NY, USA, 2004. ACM Press.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

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.

C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

Q. Wu, Y. Ying, and D. Zhou. Multi-kernel regularized classifiers. *Journal of Complexity*, 23(1):108–134, 2007.

4

Regularized least squares with two layers

In this chapter, we study the problem of learning a convex combination of kernels based on regularization with the square loss function [Micchelli and Pontil, 2005]. Despite the simplicity and importance of the model, no efficient algorithmic implementations have been made available so far. Here, we show that the involved optimization problem can be efficiently solved using a two-step optimization procedure called RLS2 (regularized least squares with two layers), alternating between the solution of a linear system and a constrained least squares problem. We adopt a combination of conjugate gradient for the linear system and SMO (sequential minimal optimization) for the least square problem, with suitable variable shrinking techniques. In section 4.2, RLS2 is interpreted as the Bayesian MAP estimate of a suitable model with two layers. Such interpretation can be used to obtain confidence intervals for least square multiple kernel learning algorithms. It can be shown, see [Micchelli and Pontil, 2007], that regularized learning of a convex combination of linear kernels on each feature with the square loss function boils down to ℓ_1 regularized least squares, also known as the Lasso [Tibshirani, 1996]. In section 4.3, such interpretation is briefly reviewed, showing that RLS2 with linear basis kernels on each feature coincides with least squares regularized with a suitably scaled ℓ_1 norm. Important aspects related to the choice of the scaling of basis kernels in least squares multiple kernel learning and weighted ℓ_1 norms are investigated in subsection 4.4. The application of RLS2 on a variety of learning problems is analyzed in section 4.5. State of the art generalization performances are achieved on several datasets, including multi-class classification of genomic data. An open source toolbox to train and validate RLS2 models with a Graphic User Interface is available at `http://www.mloss.org`. All the proof of Theorems and Lemmas are given in the Appendix.

4.1 Regularized least squares with two layers

In the previous chapter, a general class of convex optimization problems to learn finite linear combinations of kernels is shown to be equivalent to a kernel machine with two layers. Different choices of loss functions f_i lead to a variety of learning algorithms. For instance, the version with two layers of standard Support Vector Machines with "hinge" loss functions $f_i(z) = (1 - y_i z)_+$

is equivalent to the SILP (Semi-Infinite Linear Programming) multiple kernel learning problem studied in [Sonnenburg et al., 2006], whose solution can be computed, for instance, by using gradient descent.

Herein, attention is focussed on square loss functions $f_i(z) = (y_i - z)^2/2$ and the associated kernel machine with two layers. As we shall show, coefficients c_j and d_j defining the architecture as well as the "equivalent input-output kernel" K can be computed by solving a very simple optimization problem. Such problem features the minimization of a functional in (c, d), that is separately quadratic in both c and d. It is worth noticing that the square loss function can be used to solve regression problems as well as classification ones. In this respect, it has been observed that test performances of regularized least squares classifiers are comparable to that of Support Vector Machines on many datasets, see [Rifkin et al., 2003, Fung and Mangasarian, 2005] and references therein.

Let $\lambda > 0$, and consider the following optimization problem:

$$\min_{c \in \mathbb{R}^\ell, d \in \mathbb{R}^m} \left(\frac{\|y - \mathbf{K}(d)c\|_2^2}{2\lambda} + \frac{c^T \mathbf{K}(d) c}{2} \right), \quad \text{subject to} \quad (3.6). \tag{4.1}$$

Also, let Δ_m denote the standard $(m-1)$-simplex in $\mathbb{R}^m$:

$$\Delta_m := \left\{ d \in \mathbb{R}^m : \quad d \geq 0, \quad \sum_{i=1}^m d_i = 1 \right\}.$$

For any fixed d, problem (4.1) is an unconstrained quadratic optimization problem with respect to c. It is then possible to solve for the optimal c^* in closed form as a function of d:

$$c^*(d) = \left(\sum_{i=1}^m d_i \mathbf{K}^i + \lambda \mathbf{I} \right)^{-1} y. \tag{4.2}$$

By eliminating c, problem (4.1) can be reduced to a problem in d only, as stated by Lemma 4 below.

Lemma 4. *If (4.2) holds and d^* is an optimal solution of the following problem:*

$$\min_{d \in \Delta_m} \left(\frac{y^T c^*(d)}{2} \right), \tag{4.3}$$

then the pair (c^*, d^*) is an optimal solution of (4.1).

Optimal coefficients can be computed using an iterative two-step procedure that alternates between kernel and predictor optimization. The specific structure of problem (4.1) allows for exact minimization in each of the two phases. Let

$$\mathbf{V} := \begin{pmatrix} v_1 & \cdots & v_m \end{pmatrix} = \begin{pmatrix} \mathbf{K}^1 c & \cdots & \mathbf{K}^m c \end{pmatrix}, \qquad u := \left(y - \frac{\lambda c}{2} \right).$$

For any fixed c, minimization with respect to d boils down to a simplex-constrained least squares problem, as stated by Lemma 5.

Lemma 5. *For any fixed c, the optimal coefficient vector d of (4.1) can be obtained as the solution of the following problem:*

$$\min_{d \in \Delta_m} \|\mathbf{V}d - u\|_2^2. \tag{4.4}$$

Algorithm 3 alternates between minimization with respect to c obtained through the solution of the linear system (4.2), and the solution of the simplex-constrained least squares problem (4.4) in d. The non-negativity constraint induces sparsity in the vector d, thus selecting a subset of basis kernels. To understand the initialization of coefficients c and d in Algorithm 3, consider the limiting solution of the optimization problem when the regularization parameter tends to infinity. Such solution is the most natural starting point for a regularization path, since optimal coefficients can be computed in closed form.

Lemma 6. *The limiting solution of problem (4.1) when $\lambda \to +\infty$ is given by*

$$(c_\infty, d_\infty) = (0, e_i)\,, \qquad i \in \arg\max_{k=1,\ldots,m} \left(y^T \mathbf{K}^k y\right).$$

As shown in section 4.4, the result of Lemma 6 can be also used to give an important insight into the choice of the scaling $\mathbf{S}$ in the second layer.

Algorithm 3 Regularized least squares with two layers

$i \leftarrow \arg\max_{k=1,\ldots,m} y^T \mathbf{K}^k y$
$d \leftarrow e_i$
$B \leftarrow \{i\}$
while (stopping criterion is not met) **do**
  $\mathbf{K} \leftarrow 0$
  for $j \in B$ **do**
    $\mathbf{K} \leftarrow \mathbf{K} + d_j \mathbf{K}^j$
  end for
  $c \leftarrow$ Solution of the linear system $(\mathbf{K} + \lambda \mathbf{I})\, c = y$
  $u \leftarrow \left(y - \frac{\lambda c}{2}\right)$
  for $i = 1, \ldots, m$ **do**
    $v_i \leftarrow \mathbf{K}^i c$
  end for
  $d \leftarrow$ Solution of (4.4).
  $B \leftarrow \{j : d_j \neq 0\}$
end while

4.2 A Bayesian MAP interpretation of RLS2

The equivalence between regularization problem (3.4) and multiple kernel learning optimization can be readily exploited to give a Bayesian MAP (maximum a posteriori) interpretation of RLS2. To specify the probabilistic model, we need to introduce a prior distribution over the set of real-valued functions defined over $\mathcal{X}$, and define the data generation model (likelihood). In the following, $N(\mu, \sigma^2)$ denote a real Gaussian distribution with mean μ and variance σ^2, $GM(g, K)$

a Gaussian measure on the set of functions from $\mathcal{X}$ into $\mathbb{R}^m$ with mean g and covariance function K, and $U(\Omega)$ the uniform distribution in $\mathbb{R}^m$ over a set Ω of positive finite measure. Let $g : \mathcal{X} \to \mathbb{R}$ be such that

$$g = a^T \mathbf{S} g_1,$$

where $g_1 : \mathcal{X} \to \mathbb{R}^m$ is distributed according to a Gaussian measure over $\mathcal{H}_1$ with zero mean and (matrix-valued) covariance function K^1, and a is a random vector independent of g_1, distributed according to an uniform distribution over the ellipsoid $E_S := \{a \in \mathbb{R}^m : a^T \mathbf{S} a \leq 1\}$:

$$g_1 \sim GM\left(0, K^1\right), \qquad a \sim U(E_{\mathbf{S}}).$$

Regarding the likelihood, assume that the dataset is generated by drawing pairs (x_i, y_i) independently and identically distributed according to an additive Gaussian noise model:

$$y_i|(x_i, g) \sim N(g(x_i), \sigma^2).$$

When $\mathcal{H}_1$ is a finite dimensional space, the MAP estimate of g is:

$$g^* = a^{*T} \mathbf{S} g_1^*,$$

where (g_1^*, a^*) maximize the posterior density:

$$p(g|\mathcal{D}) \propto p(\mathcal{D}|g) p(g_1) p(a).$$

Specifically, we have

$$\begin{aligned} p(\mathcal{D}|g) &\propto \prod_{i=1}^{\ell} \exp\left(-\frac{\left(y_i - a^T \mathbf{S} g_1(x_i)\right)^2}{2\sigma^2}\right), \\ p(g_1) &\propto \exp\left(-\frac{\|g_1\|_{\mathcal{H}_1}^2}{2}\right), \\ p(a) &\propto \begin{cases} 1, & x \in E_{\mathbf{S}} \\ 0, & \text{else} \end{cases}. \end{aligned}$$

It follows that $a^* \in E_{\mathbf{S}}$ and, by taking the negative logarithm of $p(g|\mathcal{D})$, that the MAP estimate coincides with the solution of the regularization problem (3.4) with square loss functions and $\lambda = \sigma^2$. When $\mathcal{H}_1$ is an infinite-dimensional function space, the Gaussian measure prior for g_1 do not admit a probability density. Nevertheless, (3.4) can be still recovered by interpreting the MAP estimate as a maximal point of the posterior probability measure, as described in [Hengland, 2007].

4.3 Linear regularized least squares with two layers

In applications of standard kernel methods involving high-dimensional input data, the linear kernel on $\mathbb{R}^n$

$$K(x_1, x_2) = \langle x_1, x_2 \rangle_2$$

plays an important role. Optimization algorithms for linear machines are being the subject of a renewed attention in the literature, due to some important experimental findings. First, it turns out that linear models are already enough flexible to achieve state of the art classification performances in application domains such as text document classification, word-sense disambiguation, and drug design, see e.g. [Joachims, 2006]. Second, linear machines can be trained using extremely efficient and scalable algorithms [Hsieh et al., 2008, Shalev-Shwartz et al., 2007, Fan et al., 2008]. Finally, linear methods can be also used to solve certain non-linear problems (by using non-linear feature maps), thus ensuring a good trade-off between flexibility and computational convenience.

Linear kernels are also meaningful in the context of multiple kernel learning methods. Indeed, when the input set $\mathcal{X}$ is a subset of $\mathbb{R}^n$, a possible choice for the set of basis kernels $\widetilde{K}_k$ is given by linear kernels on each component:

$$\widetilde{K}_k(x_1, x_2) = x_1^k x_2^k. \tag{4.5}$$

Such a choice makes the input output map (3.3) a linear function:

$$g(x) = \sum_{j=1}^{m} \left(d_j s_j \sum_{i=1}^{\ell} c_i x_i^j \right) x^j = w^T x, \tag{4.6}$$

where

$$w_j := d_j s_j z_j, \qquad z_j := \sum_{i=1}^{\ell} c_i x_i^j. \tag{4.7}$$

Here, an important benefit is sparsity in the vector of weights w, that follows immediately from sparsity of vector d. In this way, linear multiple kernel learning algorithms can simultaneously perform regularization and linear *feature selection*. Such property is apparently linked to the introduction of the additional layer in the architecture, since standard kernel machines with one layer are not able to perform any kind of automatic feature selection.

Recall that standard regularized least squares with the linear kernel boils down to finite-dimensional Tikhonov regularization [Tikhonov and Arsenin, 1977], also known as ridge regression [Hoerl and Kennard, 1970]:

$$\min_{w \in \mathbb{R}^m} \left(\frac{\|y - \mathbf{X}w\|_2^2}{2\lambda} + \frac{\|w\|_2^2}{2} \right),$$

where $\mathbf{X} \in \mathbb{R}^{\ell \times m}$ denote the matrix of inputs data.

Interestingly, when the input set $\mathcal{X}$ is a subset of $\mathbb{R}^m$ and basis kernels are chosen as in (4.5), problem (4.1) produces an algorithm whose regularization path coincide with that of a scaled version on ℓ_1-regularized least squares, as shown in [Micchelli and Pontil, 2007]. The output of the RLS2 algorithm can be also used to obtain the following matrix:

$$\mathbf{\Gamma}(d) := \operatorname{diag}\{s_1 d_1, \ldots, s_m d_m\}, \tag{4.8}$$

whose diagonal elements are proportional to the estimated precisions (inverse variances) of the weights w_i, under a suitable Bayesian model.

Lemma 7. *If the basis kernels are chosen as in (4.5), the optimal solution of (4.1) can be written as in (4.6)-(4.7), where w solves the following problem:*

$$\min_{w \in \mathbb{R}^m} \left(\frac{\|y - \mathbf{X}w\|_2^2}{2\lambda} + \frac{1}{2} \left(\sum_{i=1}^{m} \frac{|w_i|}{\sqrt{s_i}} \right)^2 \right). \tag{4.9}$$

Finally, Lemma 8 below states that problem (4.1) with linear basis kernels on each feature is also equivalent to a scaled ridge regression problem, in which the optimal scaling is estimated from the data. For any fixed d, let $n(d)$ be the number of non-zero coefficients d_i, $\widetilde{\mathbf{\Gamma}}(d) \in \mathbb{R}^{n(d) \times n(d)}$ denote the diagonal sub-matrix of $\mathbf{\Gamma}$ containing all the strictly positive coefficients. Moreover, let $\widetilde{\mathbf{X}}$ denote the scaled sub-matrix of selected features $\widetilde{\mathbf{X}} := \mathbf{X}\widetilde{\mathbf{\Gamma}}$.

Lemma 8. *If the basis kernels are chosen as in (4.5), the optimal solution of (4.1) can be written as in (4.6)-(4.7), where $\mathbf{X}w = \widetilde{\mathbf{X}}\widetilde{w}$, and $(\widetilde{w}, d)$ solve the following problem:*

$$\min_{\substack{\widetilde{w} \in \mathbb{R}^{n(d)}, \\ d \in \Delta_m}} \left(\frac{\|y - \widetilde{\mathbf{X}}\widetilde{w}\|_2^2}{2\lambda} + \frac{\|\widetilde{\mathbf{\Gamma}}^{1/2}(d)\widetilde{w}\|_2^2}{2} \right), \quad \textit{subject to} \quad (4.8). \tag{4.10}$$

When d is fixed to its optimal value in problem (4.10), the optimal $\widetilde{w}$ is given by the expression:

$$\widetilde{w} = \left(\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}} + \lambda \widetilde{\mathbf{\Gamma}} \right)^{-1} \widetilde{\mathbf{X}}^T y.$$

The result of Lemma 8 can be also used to give an interesting interpretation of linear RLS2. In fact, the coefficient $s_i d_i$ can be interpreted as a quantity proportional to the inverse variance of the i-th coefficient w_i. Problem (4.10) can be seen as a Bayesian MAP estimation with Gaussian residuals, Gaussian prior on the coefficients and uniform prior over a suitable simplex on the vector of coefficients's precisions.

It is useful to introduce a notion of "degrees of freedom", see e.g. [Efron, 2004, Hastie et al., 2008]. Degrees of freedom is an index more interpretable than the regularization parameter, and can be also used to choose the regularization parameter according to tuning criteria such as C_p [Mallows, 1973], AIC [Akaike, 1973], BIC [Schwarz, 1978], GCV [Craven and Wahba, 1979]. A general expression for the effective degrees of freedom of non-linear kernel regression methods, based on the SURE (Stein's Unbiased Risk Estimator) approximation [Stein, 1981] has been recently derived in [Dinuzzo and De Nicolao, 2009]. For linear RLS2, the following quantity is an appropriate approximation of the degrees of freedom:

$$\widehat{df}(\lambda) = \operatorname{tr}\left(\widetilde{\mathbf{X}} \left(\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}} + \lambda \widetilde{\mathbf{\Gamma}} \right)^{-1} \widetilde{\mathbf{X}}^T \right). \tag{4.11}$$

Expression (4.11) corresponds to the equivalent degrees of freedom of a linear regularized estimator with regressors fixed to $\widetilde{\mathbf{X}}$ and diagonal regularization $\lambda\widetilde{\mathbf{\Gamma}}$. Notice that (4.11) neglects the non-linear dependence of matrix $\widetilde{\mathbf{X}}$ on the output

data and does not coincide with the SURE estimator of the degrees of freedom. Nevertheless, the property $0 \leq \widehat{df}(\lambda) \leq m$ holds, so that $\widehat{df}$ can be conveniently used to interpret the complexity of the linear RLS2 model (see subsection 4.5.1 for an example).

4.4 Choice of the scaling and feature selection

The ability of RLS2 to select features or basis kernels is highly influenced by the scaling $\mathbf{S}$. In this subsection, we analyze certain scaling rules that are connected with popular statistical indices, often used as "filters" for feature selection, see [Guyon et al., 2006]. Since the issue of scaling still needs further investigation, it may happen that rules different from those mentioned in this subsection perform better on specific problems.

A key observation is the following: according to Lemma 6, RLS2 with heavy regularization favors basis kernels that maximize the quantity $A_k = y^T \mathbf{K}^k y$, that represents a kind of *alignment* between the kernel $\mathbf{K}^k$ and the outputs. This means that RLS2 tends to select kernels that are highly aligned with the outputs. Since each alignment A_k is proportional to the scaling factor s_k, an effective way to choose the scaling is one that makes the alignment a meaningful quantity to maximize. First of all, we discuss the choice of scaling for the linear RLS2 algorithm introduced in subsection 4.3. The generalization to the case of non-linear basis kernels easily follows by analyzing the associated feature maps.

In the linear case, we have $\mathbf{K}^k = s_k x^k x^{kT}$, where x^k is the k-th feature vector, so that

$$A_k = s_k (y^T x^k)^2.$$

By choosing

$$s_k = (\|y\|_2 \|x^k\|_2)^{-2},$$

the alignment becomes the squared cosine of the angle between the k-th feature vector and the output vector:

$$A_k = \left(\frac{y^T x^k}{\|y\|_2 \|x^k\|_2} \right)^2 = \cos^2 \theta_k.$$

In particular, when the outputs y and the features x^k are centered to have zero mean, A_k coincides with the squared Pearson correlation coefficient between the outputs and the k-th feature, also known as *coefficient of determination*. This means that RLS2 with heavy regularization selects the features that mostly correlate to the outputs. Since the term $\|y\|_2^2$ is common to all the factors, one can also use

$$s_k = \|x^k\|_2^{-2}, \tag{4.12}$$

without changing the profile of solutions along a regularization path (though, the scale of regularization parameters is shifted). Observe that rule (4.12) makes sense also when data are not centered or centered around values other than

the mean. In fact, for some datasets, performances are better without any centering (this is the case, for instance, of Experiment 1 in subsection 4.5.1). Notice also that (4.12) only uses training inputs whereas, in a possible variation, one can replace x^k with the vector containing values of the k-th feature for both training and test data (when available). The latter procedure sometimes works better than scaling using training inputs only, and will be referred to as *transductive scaling* in the following. For binary classification with labels ± 1, the choice (4.12) with or without centering still makes sense, but other rules are also possible. Let ℓ_+ and ℓ_- denote the number of samples in the positive and negative class, respectively, and m_+^k and m_-^k denote the within-class mean values of the k-th feature:

$$m_+^k = \frac{1}{\ell_+} \sum_{i:y_i=1} x_i^k, \qquad m_-^k = \frac{1}{\ell_-} \sum_{i:y_i=-1} x_i^k.$$

By choosing

$$s_k = \frac{1}{(\sigma_+^k)^2 + (\sigma_-^k)^2}, \tag{4.13}$$

where σ_+^k and σ_-^k denote the within class standard deviations of the k-th feature, one obtain

$$A_k = \frac{(\ell_+ m_+^k - \ell_- m_-^k)^2}{(\sigma_+^k)^2 + (\sigma_-^k)^2}.$$

When the two classes are balanced ($\ell_+ = \ell_- = \ell/2$), A_k boils down to a quantity proportional to the classical Fisher criterion (or signal-to-interference ratio):

$$A_k = \frac{\ell^2}{4} \frac{(m_+^k - m_-^k)^2}{(\sigma_+^k)^2 + (\sigma_-^k)^2}.$$

Rules (4.12) and (4.13) can be generalized to the case of non-linear basis kernels, by observing that non-linear kernels can be always seen as linear upon mapping the data in a suitable *feature space*. A rule that generalizes (4.12) is the following [Rakotomamonjy et al., 2008]:

$$s_k = \left(\sum_{i=1}^{\ell} \widetilde{K}_k(x_i, x_i) \right)^{-1}, \tag{4.14}$$

that amounts to scale each basis kernel by the trace of the kernel matrix, and reduces exactly to (4.12) in the linear case. Also (4.14) can be applied with or without centering. A typical centering is *normalization in feature space*, that amounts to subtract $1/\ell \sum_{i,j} \widetilde{K}_k(x_i, x_j)$ to the basis kernel $\widetilde{K}_k$, before computing (4.14). A transductive scaling rule can be obtained by extending the sum to both training and test inputs, namely computing the inverse trace of the overall kernel matrix, as in [Lanckriet et al., 2004]. Finally, a non-linear

generalization of (4.13) is obtained by letting:

$$(\sigma_+^k)^2 = \frac{1}{\ell_+} \sum_{i:y_i=1}^{\ell} \left(\widetilde{K}_k(x_i, x_i) - \frac{1}{\ell_+} \sum_{j:y_j=1} \widetilde{K}_k(x_i, x_j) \right)$$

$$(\sigma_-^k)^2 = \frac{1}{\ell_-} \sum_{i:y_i=-1}^{\ell} \left(\widetilde{K}_k(x_i, x_i) - \frac{1}{\ell_-} \sum_{j:y_j=-1} \widetilde{K}_k(x_i, x_j) \right).$$

4.5 Experiments

In this section, the behavior of linear and non-linear RLS2 on several learning problems is analyzed. In subsection 4.5.1, an illustrative analysis of linear RLS2 is proposed, whose goal is to study the feature selection capabilities and the dependence on the regularization parameter of the algorithm in simple experimental settings. Recall that linear RLS2 is input-output equivalent to a scaled version of the Lasso. Indeed, the following experiments show that the choice of the scaling crucially affect performances. In this respect, the hierarchical interpretation of RLS2 is very useful to choose the feature scaling in a principled way, see section 4.4. RLS2 with non-linear kernels is analyzed in subsection 4.5.2, where an extensive benchmark on several regression and classification problems from UCI repository is carried out. Finally, multi-class classification of microarray data is considered in subsection 4.5.3.

Computations are carried out in a Matlab environment and the sub-problem (4.4) is solved using an SMO-like (Sequential Minimal Optimization) algorithm [Platt, 1998]. The current implementation features conjugate gradient to solve linear systems and a sophisticated variable shrinking technique to reduce gradient computations. In all the experiments, the stopping criterion for Algorithm 3 is the following test on the normalized residual of linear system (4.2):

$$\| (\mathbf{K} + \lambda \mathbf{I})\, c - y \|_2 \leq \delta \|y\|_2.$$

The choice $\delta = 10^{-2}$ turns out to be sufficient to make all the coefficients stabilize to a good approximation of their final values. A full discussion of optimization details is outside the scope of this chapter. All the experiments have been run on a Core 2 Duo T7700 2.4 GHz, 800 MHz FSB, 4 MB L2 cache, 2 GB RAM.

4.5.1 Linear RLS2: illustrative experiments

In this subsection, we perform two experiments to analyze the behavior of linear RLS2. In the first experiment, a synthetic dataset is used to investigate the ability of linear RLS2 to perform feature selection. The dependence of generalization performances of RLS2 and other learning algorithms on the training set size is analyzed by means of learning curves. The goal of the second experiment is to illustrate the qualitative dependence of coefficients on the regularization parameter and give an idea of the predictive potentiality of the algorithm.

Experiment 1 (Binary strings data) In the first experiment, a synthetic BINARY STRINGS dataset has been generated: 250 random binary strings $x_i \in \{0,1\}^{100}$ are obtained by independently sampling each bit from a Bernoulli distribution with $p = 0.5$. Then, the outputs have been generated as

$$y_i = x_i^1 + x_i^2 + x_i^3 + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are small independent Gaussian noises with zero mean and standard deviation $\sigma = 0.01$. In this way, the outputs only depend on the first three bits of the input binary string. The dataset has been divided into a training set of 150 input output pairs and a test set containing the remaining 100 data pairs. We compare the RMSE (root mean squared error) learning curves obtained by varying the training set size using five different methods:

1. RLS (regularized least squares) with "ideal" kernel:

$$K(x_1, x_2) = x_1^1 x_2^1 + x_1^2 x_2^2 + x_1^3 x_2^3. \tag{4.15}$$

2. RLS with linear kernel (ridge regression).
3. RLS with Gaussian RBF kernel

$$K(x_1, x_2) = \exp\left(-0.005 \|x_1 - x_2\|_2^2\right).$$

4. RLS2 with linear basis kernels (4.5) and scaling (4.12).
5. Lasso regression.

The goal here is to assess the overall quality of regularization paths associated with different regularization algorithms, independently of model selection procedures. To this end, we compute the RMSE on the test data as a function of the training set size and evaluate the lower bounds of the learning curves with respect to variation of the regularization parameter. Results are shown in Figure 4.1, whose top plot reports the lower bounds of learning curves for all the five algorithms with training set sizes between 1 and 150. Notice that all the five methods are able to learn, asymptotically, the underlying "concept", up to the precision limit imposed by the noise, but methods that exploits coefficients sparsity are faster to reach the asymptotic error rate. Not surprisingly, the best method is RLS with the "ideal kernel" (4.15), which incorporates a strong prior knowledge: the dependence of the outputs on the first three bits only. Though knowing in advance the optimal features is not realistic, this method can be used as a reference. The slowest learning curve is that associated to RLS with Gaussian RBF kernel, which only incorporates a notion of smoothness. A good compromise is RLS with linear kernel, which uses the knowledge of linearity of the underlying function, and reaches a good approximation of the asymptotic error rate after seeing about 100 strings. The remaining two methods (Lasso and linear RLS2) incorporate the knowledge of both linearity and sparsity. They are able to learn the underlying concept after seeing only 12 examples, despite the presence of the noise. Since after the 12-th example, performances of Lasso and linear RLS2 are basically equivalent, is it interesting to see what happen for very small sample sizes. The bottom plot of Figure 4.1 is a zoomed version

Term	RLS2	Best subset	LS	Ridge	Lasso	PCR	PLS
INTERCEPT	2.452	2.477	2.465	2.452	2.468	2.497	2.452
LCAVOL	0.544	0.740	0.680	0.420	0.533	0.543	0.419
LWEIGHT	0.207	0.316	0.263	0.238	0.169	0.289	0.344
AGE			-0.141	-0.046		-0.152	-0.026
LBPH	0.104		0.210	0.162	0.002	0.214	0.220
SVI	0.170		0.305	0.227	0.094	0.315	0.243
LCP			-0.288	0.000		-0.051	0.079
GLEASON			-0.021	0.040		0.232	0.011
PGG45	0.064		0.267	0.133		-0.056	0.084
Test error	0.454	0.492	0.521	0.492	0.479	0.449	0.528
Std error	0.152	0.143	0.179	0.165	0.164	0.105	0.152

Table 4.1: PROSTATE CANCER data: comparison of RLS2 with other subset selection and shrinkage methods. Estimated coefficients, test error and their standard error are reported. Results for methods other than RLS2 are taken from [Hastie et al., 2008]. Blank entries corresponds to variables not selected.

of the top plot with training set sizes between 1 and 30, showing only the learning curves for the three methods that impose sparsity. Until the 8-th example, the Lasso learning curve stays lower than the RLS2 learning curve. After the 8-th example, the RLS2 learning curve stays uniformly lower than the Lasso, indicating an high efficiency in learning noisy sparse linear combinations. Since the multiple kernel learning interpretation of RLS2 suggests that the algorithm is being learning the "ideal" kernel (4.15) simultaneously with the predictor, it might be interesting to analyze the asymptotic values of kernel coefficients d_i. Indeed, after the first 12 training examples, RLS2 sets to zero all the coefficients d_i except the first three, which are approximately equal to $1/3$.

Experiment 2 (Prostate Cancer data) Linear RLS2 is applied to the PROSTATE CANCER dataset, a regression problem whose goal is to predict the level of prostate-specific antigen on the basis of a number of clinical measures in men who were about to receive a radical prostatectomy [Stamey et al., 1989]. These data are used in the textbook [Hastie et al., 2008] to compare different feature selection and shrinkage methods, and have been obtained from the web site `http://www-stat.stanford.edu/ElemStatLearn/`. Data have been preprocessed by normalizing all the inputs to zero mean and unit standard deviation. The dataset is divided into a training set of 67 examples and a test set of 30 examples. To choose the regularization parameter, the 10-fold cross-validation score has been computed for different values of λ in the interval $[10^{-4}, 10^{4}]$ on a logarithmic scale. The scaling coefficients s_i are chosen as in (4.12), thus normalizing each training feature to have unit norm. An intercept term equal to the average of training outputs has been subtracted to the outputs before estimating the other coefficients. For each of the dataset splits, the MSE (mean squared error) has been computed on the validation data. Figure 4.2 reports average and standard error bands for validation MSE along a regularization path. Following [Hastie et al., 2008], we pick the value of λ corresponding to the least complex model within one standard error of the best validation score.

Dataset	Feature standardization	Examples	Features	Kernels
AUTO-MPG	Yes	392	7	104
CPU	Yes	209	8	494
SERVO	No	167	4	156
HOUSING	Yes	506	13	182
HEART	Yes	270	13	182
LIVER	No	345	6	91
PIMA	Yes	768	8	117
IONOSPHERE	Yes	351	33	442
WPBC	Yes	194	34	455
SONAR	No	208	60	793

Table 4.2: Data sets used in the experiments. The first four are regression problems while the last six are classification problems.

Dataset	60/40	70/30
AUTO-MPG	2.79(0.209)	2.72(0.224)
CPU	21.8(11.3)	21.2(11.9)
SERVO	0.755(0.116)	0.696(0.152)
HOUSING	3.61(0.465)	3.49(0.558)
HEART	83.8(2.98)	84 (3.28)
LIVER	69(3.57)	69.8(3.79)
PIMA	76.7(1.92)	77.1(1.96)
IONOSPHERE	93.3(1.83)	93.5(1.93)
WPBC	76.7(3.71)	76.4(4.63)
SONAR	83.6(3.69)	86.1(4.52)

Table 4.3: RLS2 regression and classification: average and standard deviation of test performance (RMSE for regression, accuracy for classification) over 100 dataset splits. Results with two different training/test ratio are reported: 60/40 (first two columns), and 70/30 (last two columns).

In a second phase, the whole training set (67 examples) is used to compute the RLS2 solution with different values of λ. Figure 4.3 reports the profile of RLS2 coefficients w_j, see equation (4.7), along the whole regularization path as a function of the degrees of freedom defined as in (4.11). In correspondence with the value of λ chosen in the validation phase, RLS2 selects 5 input variables out of 8. Table 4.1 reports the value of coefficients estimated by RLS2 together with the test error and his standard error. For comparison, Table 4.1 also reports models and results taken from [Hastie et al., 2008] associated with LS (Least Squares), Best subset regression, Ridge Regression, Lasso regression, PCR (Principal Component Regression), PLS (Partial Least Squares). The best model on these data is PCR, but RLS2 achieves the second lowest test error by using only 5 variables.

4.5.2 RLS2: regression and classification benchmark

In this subsection, benchmark experiments on four regression and six classification problems from UCI repository are illustrated (Table 4.2). RLS2 has

Dataset	Split ratio	Kernels	Iterations	Time (s)
Auto-mpg	60/40	21.3(2.64)	78.8(2.16)	7.2(1.22)
	70/30	22.1(2.3)	80.6(2.17)	17.4(1.82)
Cpu	60/40	30.2(3.65)	46.6(2.42)	26.4(13)
	70/30	31.9(3.43)	47(1.97)	29(13.8)
Servo	60/40	11(1.37)	71(2.26)	1.95(0.214)
	70/30	11.4(1.8)	73.6(2.09)	2.37(0.162)
Housing	60/40	37.4(3.95)	90.6(2.73)	38.7(1.81)
	70/30	38.7(3.31)	92.4(2.43)	51.9(2.07)
Heart	60/40	13.7(2.03)	75.9(2.33)	4.22(0.234)
	70/30	14.3(2.12)	77.8(2.16)	5.83(0.411)
Liver	60/40	23.5(2.87)	54.6(2.53)	3.56(0.545)
	70/30	15.1(2.11)	55.1(2.43)	4.51(0.399)
Pima	60/40	14.6(2.17)	84.7(2.67)	53.6(2.23)
	70/30	15.6(2.18)	86.6(2.33)	70.8(4.14)
Ionosphere	60/40	61.6(6.03)	70.7(3.13)	16.7(0.96)
	70/30	67.6(6.92)	73.1(2.92)	24.2(2.45)
Wpbc	60/40	2.29(0.832)	60.5(3.03)	4.89(0.305)
	70/30	2.13(0.906)	60.6(2.42)	6.85(0.198)
Sonar	60/40	45.1(4.45)	60.5(2.18)	9.42(0.191)
	70/30	35.9(3.5)	61.6(1.9)	11.4(0.227)

Table 4.4: RLS2 regression and classification: number of selected kernels in correspondence with the optimal value of λ, number of iterations and training time in seconds to compute a regularization path.

been run on 100 random dataset splits with two different training/test ratios: 60/40 and 70/30. For each dataset split, an approximate regularization path with 30 values of λ on a logarithmic scale in the interval $\left[10^{-6}, 10^{6}\right]$ has been computed. To speed-up the regularization path computation, a warm-start technique is employed: the value of λ is iteratively decreased, while kernel-expansion coefficients d_i are initialized to their optimal values obtained with the previous value of λ. Performances are measured by accuracy for classification and RMSE (root mean squared error) for regression. For each dataset split and value of the regularization parameter, the following quantities are computed: prediction performance on the test set, number of selected kernels (number of non-zero d_i), training time in seconds and number of iterations to compute the whole regularization path. Datasets have been pre-processed by removing examples with missing features and converting categorical features to binary indicators. For some of the datasets (see Table 4.2) input features have been standardized to have zero mean and unitary standard deviation. For classification, output labels are ± 1 and predictions are given by the sign of the predictor. For regression, an intercept term equal to the mean of training outputs is subtracted to the training data. Basis kernel matrices are pre-computed and the scaling matrix $\mathbf{S}$ is chosen according to the rule (4.14) with transductive scaling.

To better compare the results to similar benchmarks for multiple kernel learning, see e.g. [Rakotomamonjy et al., 2008], the same set of basis kernels for all the datasets has been chosen. We remark that such agnostic approach is not representative of a realistic application of the algorithm, in which the choice of basis kernels $\widetilde{K}_k$ should reflect a-priori knowledge about the learning task to be solved. The set of basis kernels contains the following:

- Polynomial kernels

  $$\widetilde{K}_k(x_1, x_2) = (1 + \langle x_1, x_2 \rangle_2)^d$$

  with $d = 1, 2, 3$.

- Gaussian RBF kernels

  $$\widetilde{K}_k(x_1, x_2) = \exp\left(-\gamma \|x_1 - x_2\|_2^2\right),$$

  with 10 different values of γ chosen on a logarithmic scale between 10^{-3} and 10^{3}.

Kernels on each single feature and on all the features are considered, so that the number of basis kernels is an affine function of the number of features (recall that categorical features have been converted to binary indicators). More precisely, we have $m = 13(N + 1)$.

All the profiles of test prediction performance, number of kernels and number of iterations for the 70/30 dataset split in correspondence with different values of the regularization parameter are reported in Figures 4.4-4.8. From the top plots, it can be seen that test performances are relatively stable to variations of the regularization parameter around the optimal value λ^*, indicating that RLS2 is robust with respect to the use of different model selection procedures. For regression datasets such as CPU, SERVO, or HOUSING, optimal performances

seems to be reached in correspondence with the un-regularized solution $\lambda \to 0^+$. Lines in light color are associated with single dataset splits, while thick lines are the averages over different dataset splits. The vertical dotted line corresponds to the value of the regularization parameter with best average test performance. The average number of selected kernels vary quite smoothly with respect to the regularization parameter. For large values of λ, RLS2 chooses only one basis kernel. For small values of λ, the number of selected kernels grows and exhibits an higher variability. The bottom plots in Figures 4.4-4.8 give an idea of the computation burden required by alternate optimization for RLS2 in correspondence with different values of λ. In correspondence with high values of the regularization parameter, the algorithm converges in a single iteration. This occurs also for the very first value on the regularization path, meaning that the initialization rule is effective. With low values of λ, RLS2 also converges in a single iteration.

Test performances for regression and classification are summarized in Table 4.3, where the average and standard deviation with respect to the 100 dataset splits of either RMSE (regression) or accuracy (classification) in correspondence with to the best value of λ are reported. Performances of other kernel learning algorithms on some of these datasets can be found in [Lanckriet et al., 2004, Ong et al., 2005, Rakotomamonjy et al., 2008] and references therein. Another benchmark study that might be useful for comparison is [Meyer et al., 2003]. Observe that comparisons should be handled with care due to the use of different experimental procedures and optimization problems. For instance, [Lanckriet et al., 2004] uses an 80/20 dataset split ratio, [Ong et al., 2005] uses 60/40, while [Rakotomamonjy et al., 2008] uses 70/30. Also, these work use different numbers of dataset splits. Here, we perform experiment both with a 60/40 rule, and a 70/30 rule and obtain state of the art results on all the datasets (compare with the mentioned references). These experiments shows that RLS2 results are competitive and complexity of the model is well controlled by regularization. In particular, state of the art performances are reached on Servo, Housing, Hearth, Pima, Ionosphere. Finally, it should be observed that, although multiple kernel learning machines have been used as black box methods, the use of basis kernels on single features sometimes also selects a subset of relevant features. Such property is remarkable since standard kernel methods are not able to perform "embedded" feature selection.

Table 4.4 reports the average and standard deviation of number of selected kernels in correspondence with the optimal value of λ, number of iterations and training time needed to compute a regularization path for all the regression and classification datasets studied in this subsection. From the columns of selected kernels, it can be seen that a considerable fraction of the overall number of basis kernels is filtered out by the algorithm in correspondence with the optimal value of the regularization parameter. By looking at the number of iterations needed to compute the path with 30 values of the regularization parameter, one can see that the average number of iterations to compute the solution for a single value of λ is in between 1 and 3, indicating that the warm-start procedure is rather effective at exploiting the continuity of solutions with respect to the regularization parameter. As a matter of fact, most of the optimization work is spent in correspondence with a central interval of values of λ, as shown in the bottom plots of Figures 4.4-4.8. Finally, from the last column, reporting

Method	Test errors out of 54	Selected genes
Support Vector Classifier	14.0	16,063
L1-penalized multinomial	13.0	269
Lasso regression (OVA)	12.5	1,429
L2-penalized discriminant analysis	12.0	16,063
Elastic-net penalized multinomial	11.8	384
Linear RLS2 (OVA)	**9.8**	855

Table 4.5: 14 CANCERS data: average test error (see the text for details) and number of selected genes for different classification algorithms. For RLS2, the number of selected genes is relative to the least complex model maximizing the validation accuracy. Results for methods other than RLS2 are taken from [Hastie et al., 2008].

average and standard deviation of training times, it can be seen that, with the current implementation of RLS2, regularization paths for all the datasets in this subsection can be computed in less than one minute in the average (see the introduction of this section for experimental details).

4.5.3 RLS2: multi-class classification of microarray data

RLS2 can be applied to multi-class classification problems by solving several binary classification problems and combining their outcomes. A possible way to combine binary classifiers is the OVA (one versus all) approach, in which each class is compared to all the others and test labels are assigned to the class maximizing the confidence (the real-valued output) of the corresponding binary classifier.

Linear RLS2 with OVA has been applied to the 14 CANCERS dataset [Ramaswamy et al., 2001], a delicate multi-class classification problem whose goal is to discriminate between 14 different types of cancer, on the basis of microarray measurements of 16063 gene expressions. Gene measurements and type of cancer (labels) are available for 198 patients, the dataset being already divided in a training set of 144 patients, and a test set of 54 patients. Another important goal in this problem is to individuate a small subset of genes which is relevant to discriminate between the different kind of cancer. [Hastie et al., 2008] reports several results for these data using a variety of classification methods. Algorithms such as the Support Vector Classifier uses all the genes to compute the classification boundaries, while others such as Lasso or Elastic Net are also able to select a subset of relevant genes. Since the feature selection experiment in subsection 4.5.1 suggests that the algorithm may be very efficient at selecting relevant features from noisy examples, a microarray dataset seems to be an appropriate choice for testing the algorithm.

Gene expressions for each patient have been firstly standardized to have zero mean and variance one. For each binary classifier, coefficients s_i are chosen as $s_i = \left(\sigma_+^2 + \sigma_-^2\right)^{-1/2}$, where σ_+^2 and σ_-^2 are the within-class sample variances computed using all the training data. Such scaling gives more weight to genes whose expressions exhibits small within-class variability, and seems to slightly improve classification performances. A validation accuracy profile has been

computed using stratified 8-fold cross validation, where the folds are organized to preserve the class proportions. For the final model, we pick the highest value of λ maximizing the validation accuracy. Figure 4.9 reports the profiles of training accuracy, cross-validation accuracy with corresponding standard error bands, and test accuracy for 50 logarithmically spaced values of the regularization parameter. Table 4.5 reports the number of test errors and selected genes in correspondence with the value of λ chosen in the validation phase, for RLS2 and other methods from [Hastie et al., 2008]. Test errors in Table 4.5 are averages of test errors for different classifiers associated with all the different values of the regularization parameter that maximizes the cross-validation score (this explains the presence of non-integer values). For linear RLS2, such procedure yields a value of about 9.8. Although the test set size is too small to draw significative conclusions from this comparison, linear RLS2 seems to work rather well on this problem and achieve the best test performances. Such good performance confirm both the effectiveness of the OVA multi-class approach, and the importance of weight scaling in ℓ_1-regularized problems.

4.6 Conclusions

RLS2 is an optimization algorithm to learn convex combination of kernels based on regularization with the square loss function. Optimization is based on a combination of alternate minimization, linear conjugate gradient, and sequential minimal optimization (SMO) techniques. State of the art performances are achieved on several learning problems, including multi-class classification of microarray data. An open source set of MATLAB scripts with Graphic User Interface for RLS2 and linear RLS2 is available at `http://www.mloss.org`.

Bibliography

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csáki, editors, *Second International Symposium on Information Theory*. Académiai Kiadó, Budapest, 1973.

P. Craven and G. Wahba. Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403, 1979.

F. Dinuzzo and G. De Nicolao. An algebraic characterization of the optimum of regularized kernel methods. *Machine Learning*, 74(3):315–345, 2009.

B. Efron. The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(14):619–632, 2004.

R. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

G. M. Fung and O. L. Mangasarian. Multicategory proximal support vector machine classifiers. *Machine Learning*, 59(1-2):77–97, 2005.

I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, editors. *Feature Extraction: Foundations and Applications*. Studies in Fuzziness and Soft Computing. Springer-Verlag, Secaucus, NJ, USA, 2006.

T. J. Hastie, R. J. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Springer-Verlag, Canada, 2nd edition, 2008.

M. Hengland. Approximate maximum a posteriori with Gaussian process priors. *Constructive Approximation*, 26:205–224, 2007.

A. E. Hoerl and R. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

C. Hsieh, K.W. Chang, C.J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 408–415, Helsinki, Finland, 2008.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–226, Philadelphia, PA, USA, 2006.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

C. Mallows. Some comments on C_p. *Technometrics*, 15:661–675, 1973.

D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55(1-2):169–186, 2003.

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.

C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

J. Platt. Fast training of support vector machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA, 1998.

A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

S Ramaswamy, P Tamayo, R Rifkin, S Mukherjee, C H Yeang, M Angelo, C Ladd, M Reich, E Latulippe, J P Mesirov, T Poggio, W Gerald, M Loda, E S Lander, and T R Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences of the United States of America*, 98:15149–15154, 2001.

R. Rifkin, G. Yeo, and T. Poggio. Regularized least squares classification. In Suykens, Horvath, Basu, Micchelli, and Vandewalle, editors, *Advances in Learning Theory: Methods, Model and Applications*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*, chapter 7, pages 131–154. VIOS Press, Amsterdam, 2003.

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

S. Shalev-Shwartz, Y. Singer, and N. Srebro. PEGASOS: Primal Estimated sub-GrAdient SOlver for Svm. In *Proceedings of the 24th Annual international conference on Machine learning (ICML 2007)*, pages 807–814, New York, NY, USA, 2007. ACM.

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

T. Stamey, J. Kabalin, J. McNeal, I. Johnstone, F. Freiha, E. Redwine, and N. Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate ii radical prostatectomy treated patients. *Journal of Urology*, 16:1076–1083, 1989.

C. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9:1135–1151, 1981.

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

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D. C., 1977.

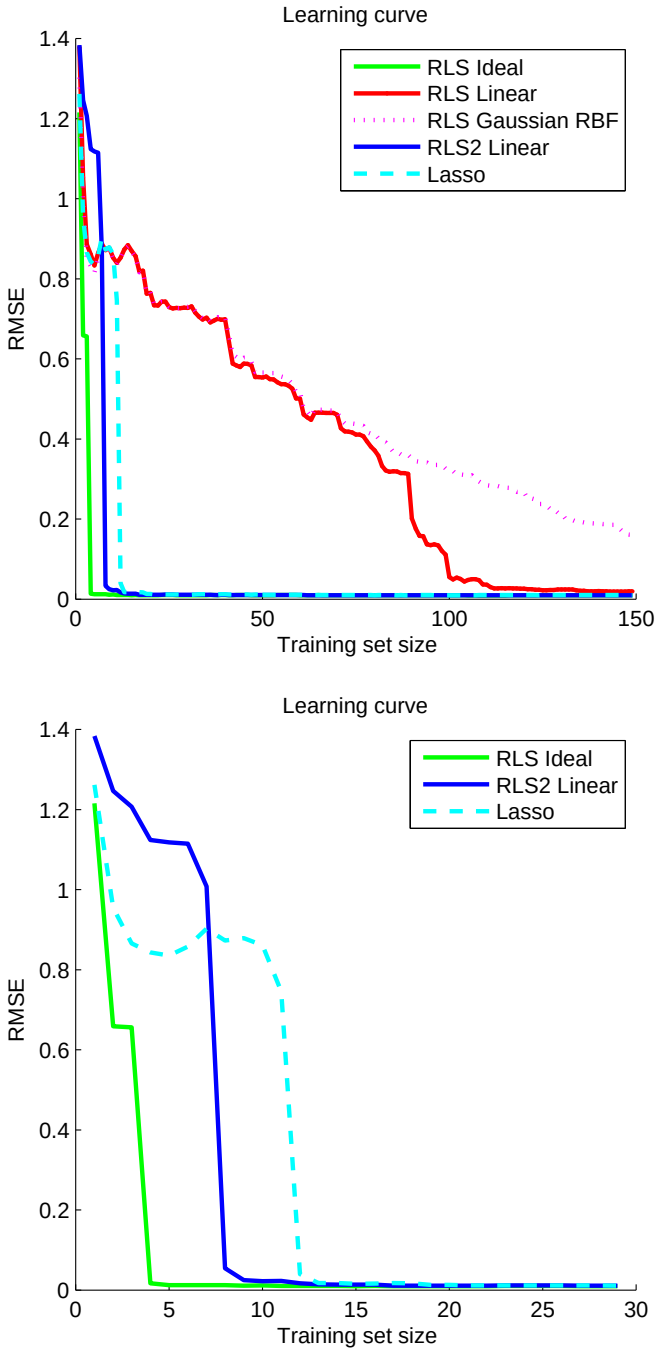

Figure 4.1: BINARY STRINGS data: lower bounds of RMSE learning curves. The top plot shows test RMSE for training set sizes between 1 and 150 with five different methods: RLS with ideal kernel (see details in the text), RLS with linear kernel (ridge regression), RLS with Gaussian RBF kernel, linear RLS2 and Lasso. The bottom plot is the "zoomed" version of the top plot for training set sizes between 1 and 30, for RLS with ideal kernel, linear RLS2 and Lasso.

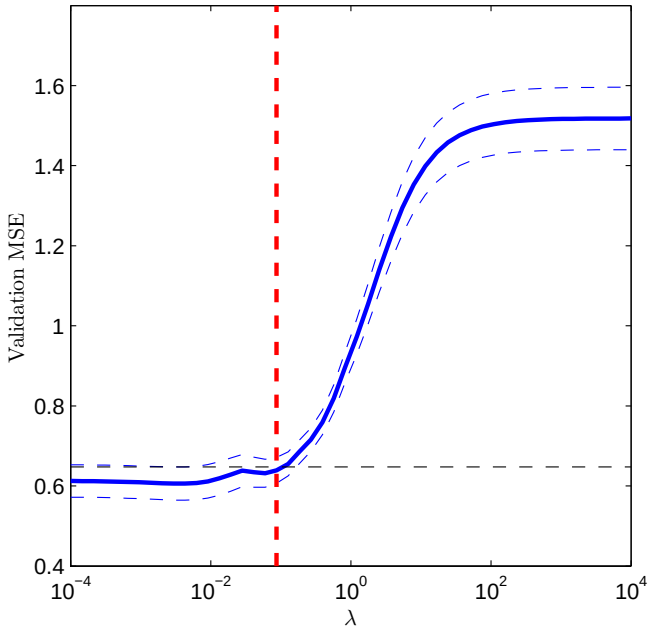

Figure 4.2: Prostate Cancer data: 10-fold cross-validation prediction error curves and their standard errors bands for linear RLS2. Model complexity increases from the right to the left. The vertical line corresponds to the least complex model within one standard error of the best.

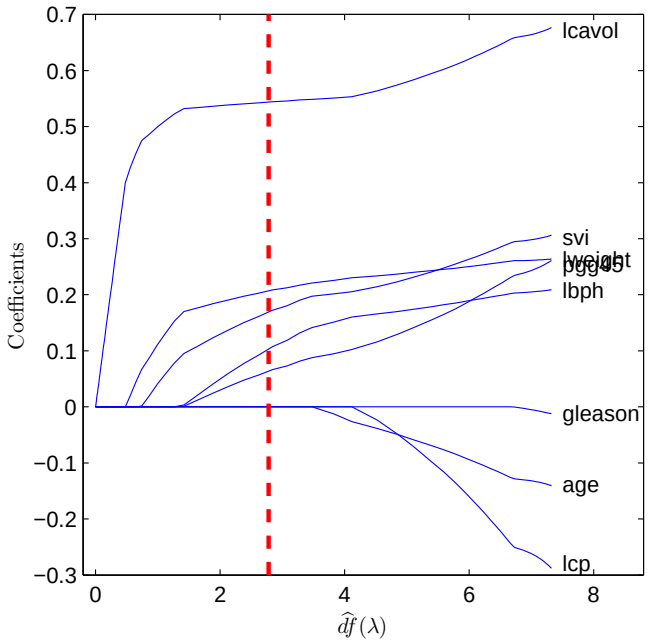

Figure 4.3: Prostate Cancer data: profiles of RLS2 coefficients with respect to a continuous variation of the regularization parameter. Coefficients are plotted versus $\widehat{df}(\lambda)$, the approximate degrees of freedom. The vertical line corresponds to the value of λ chosen in the validation phase.

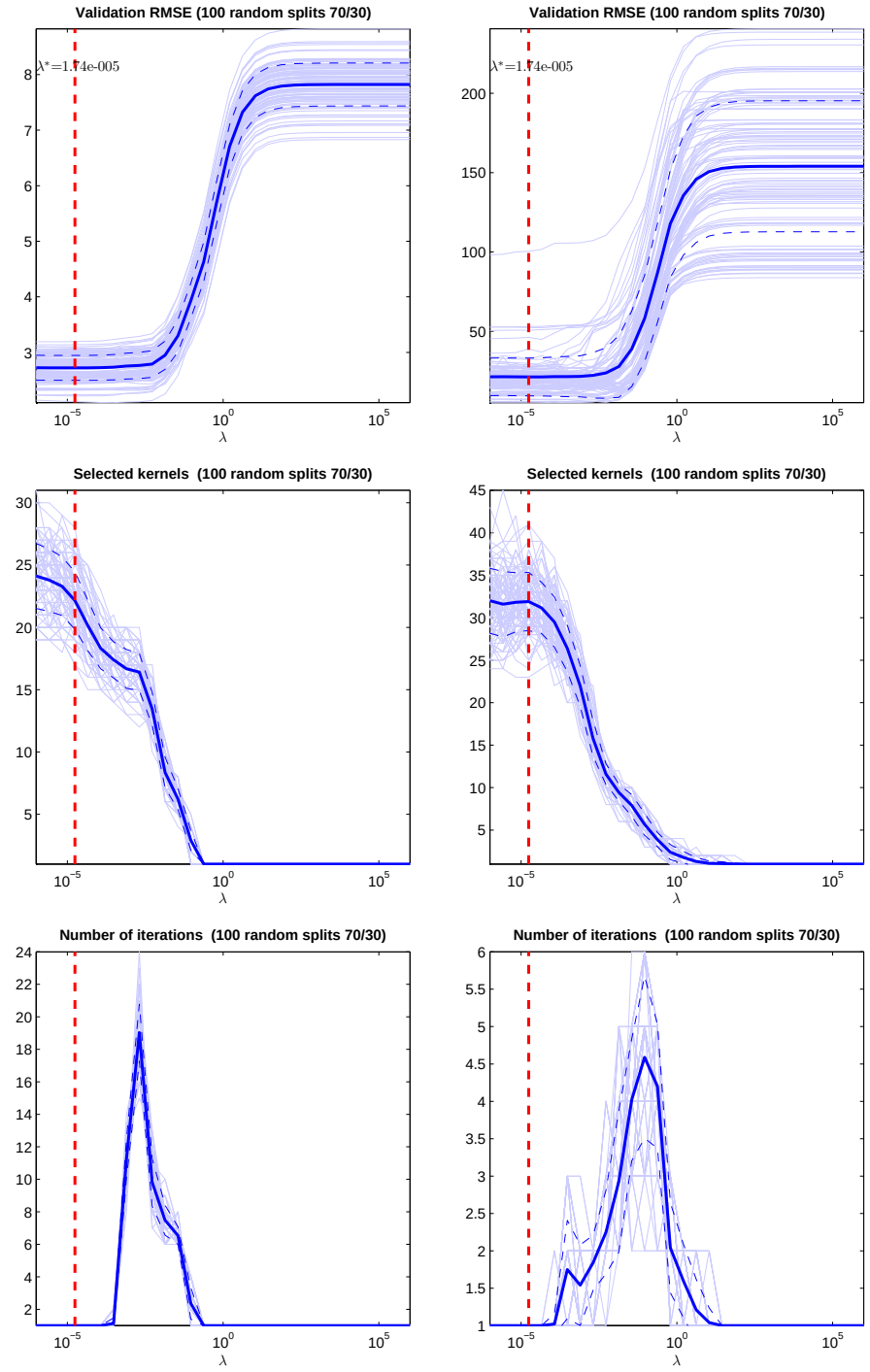

Figure 4.4: RLS2 on the Auto-mpg (left) and the Cpu (right) dataset: RMSE on the test data (top), number of selected kernels (center), and number of iterations (bottom) along a regularization path.

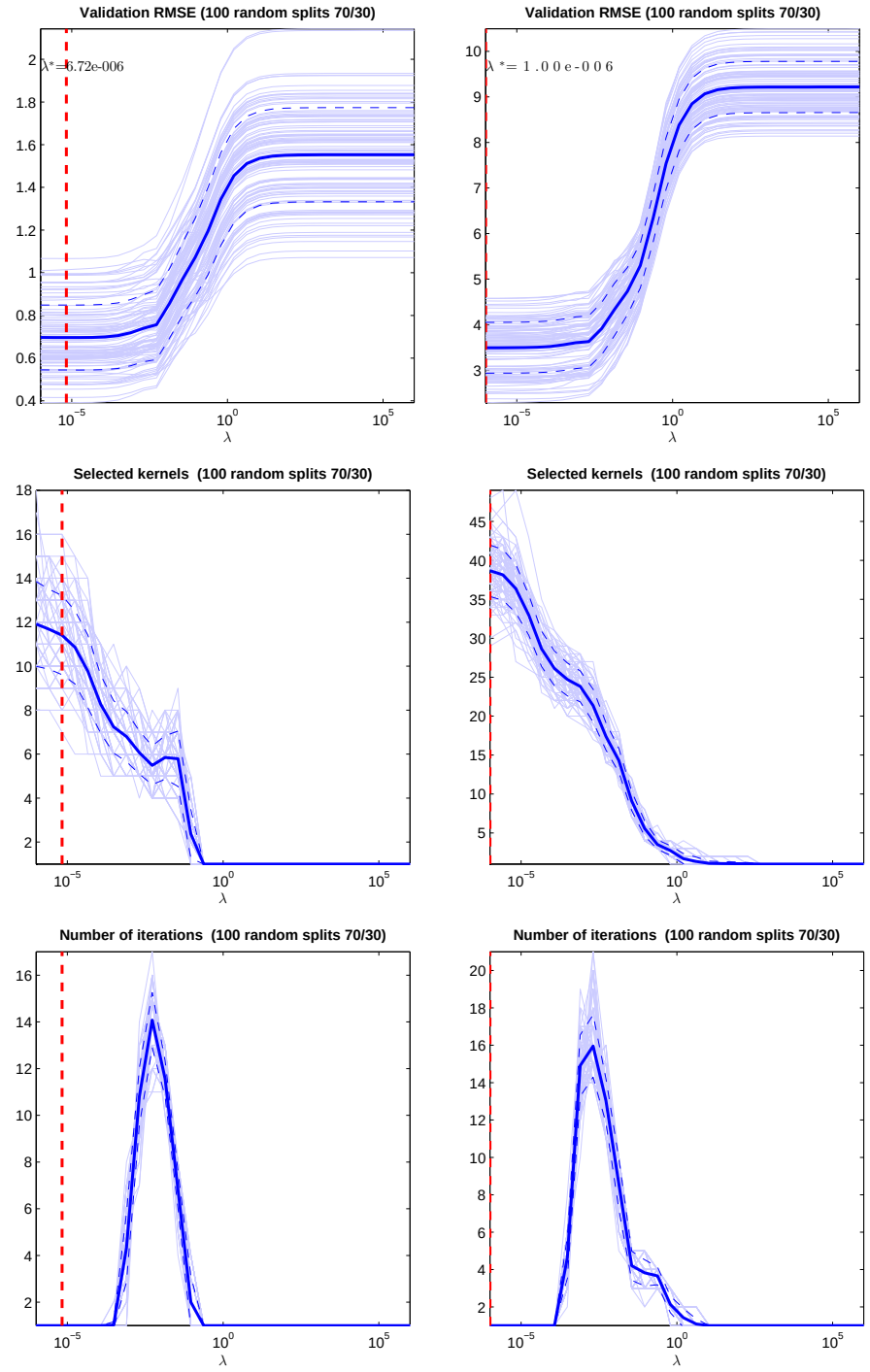

Figure 4.5: RLS2 on the SERVO (left) and the HOUSING (right) dataset: RMSE on the test data (top), number of selected kernels (center), and number of iterations (bottom) along a regularization path.

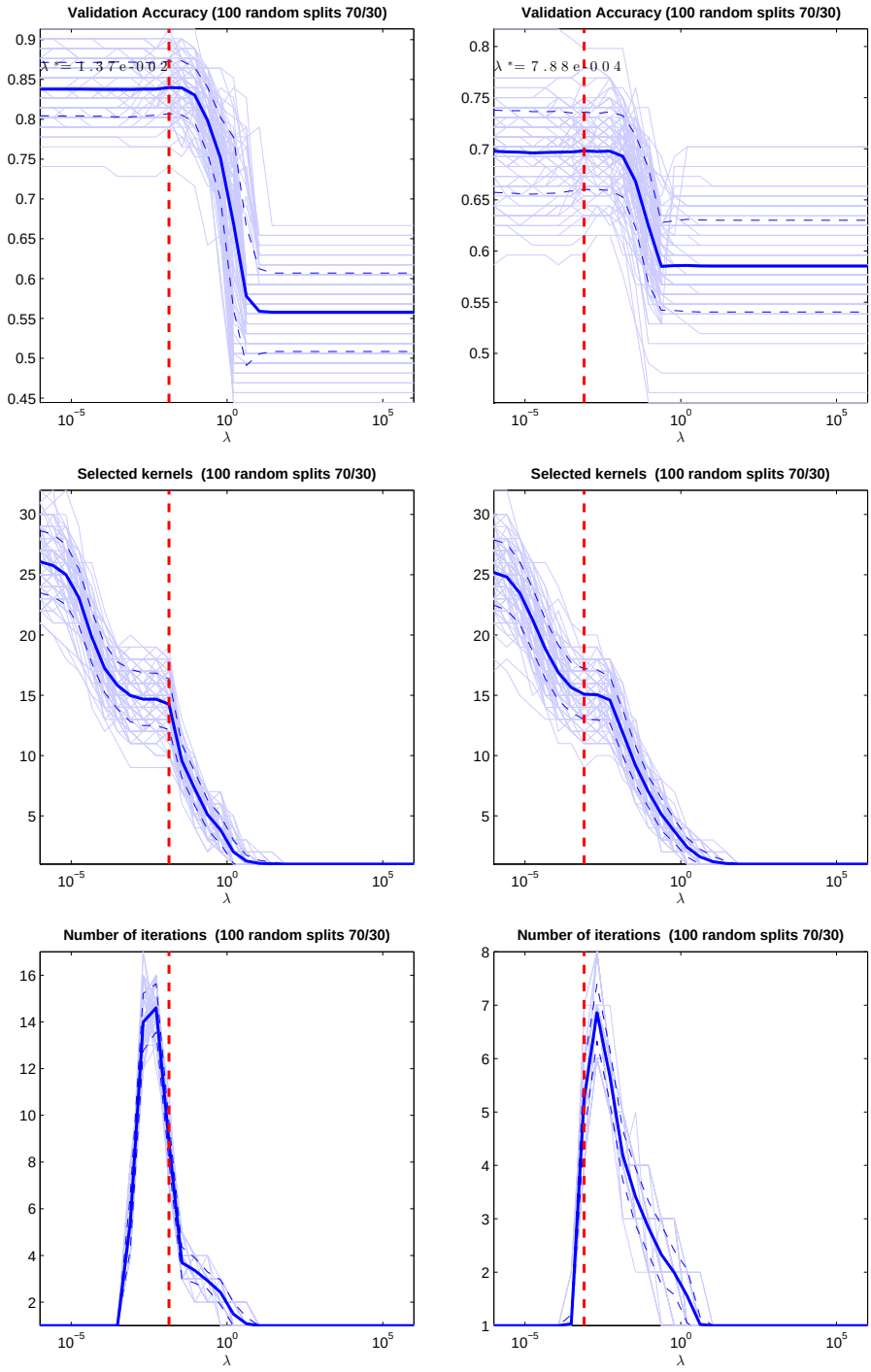

Figure 4.6: RLS2 on the HEART (left) and LIVER (right) dataset: classification accuracy on the test data (top), number of selected kernels (center), and number of iterations (bottom) along a regularization path.

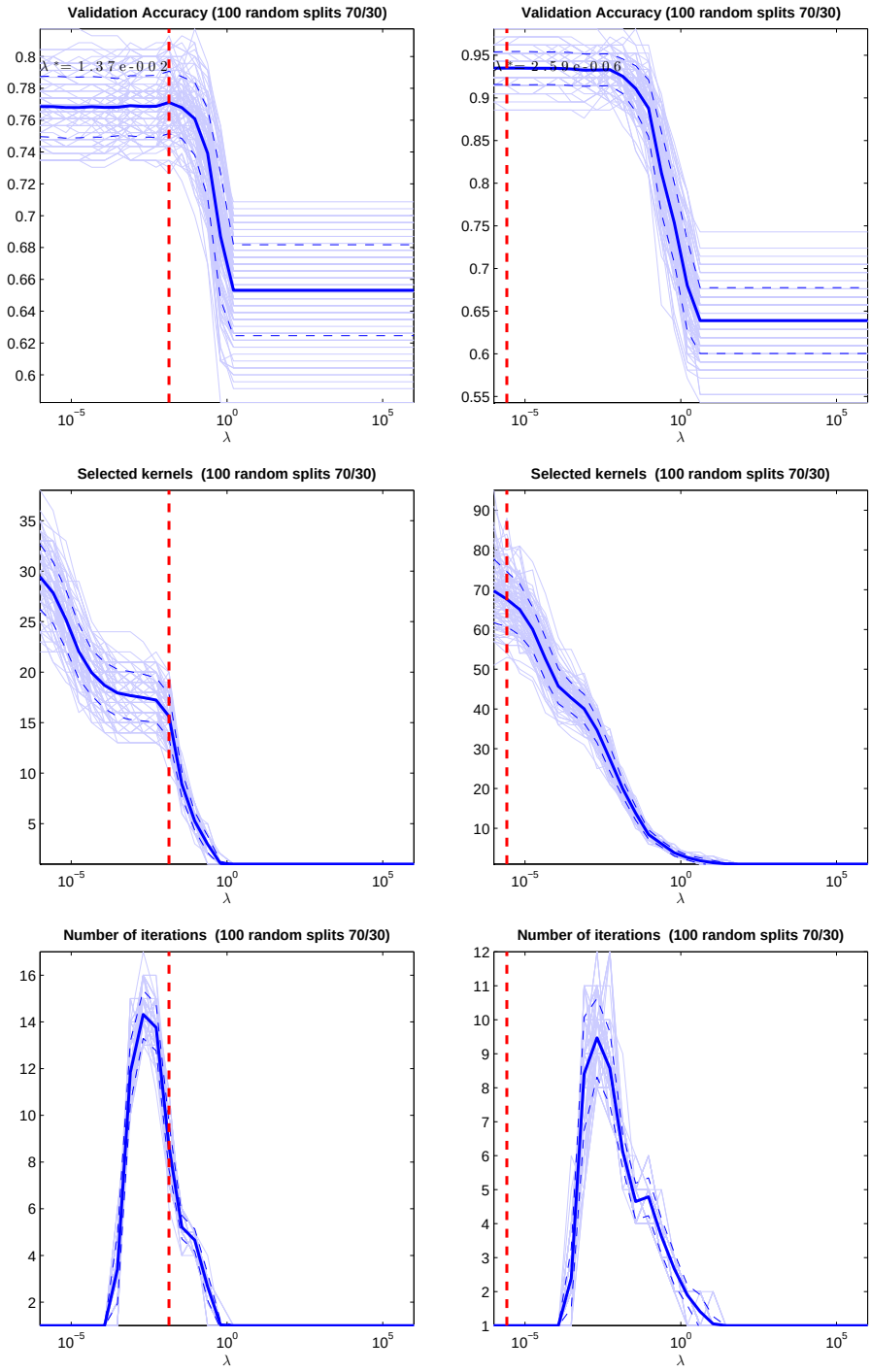

Figure 4.7: RLS2 on the PIMA (left) and IONOSPHERE (right) dataset: classification accuracy on the test data (top), number of selected kernels (center), and number of iterations (bottom) along a regularization path.

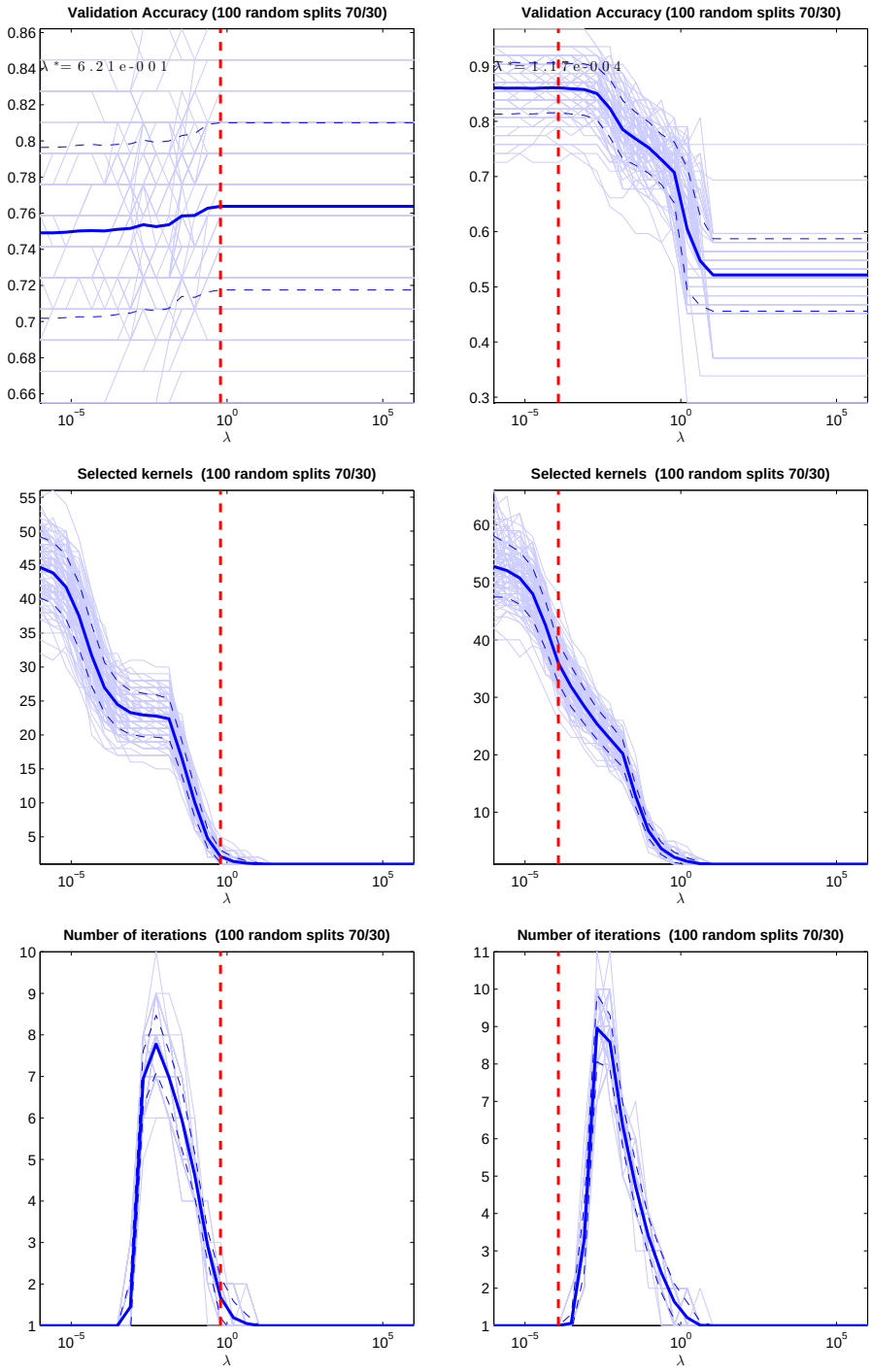

Figure 4.8: RLS2 on the WPBC (left) and SONAR (right) dataset: classification accuracy on the test data (top), number of selected kernels (center), and number of iterations (bottom) along a regularization path.

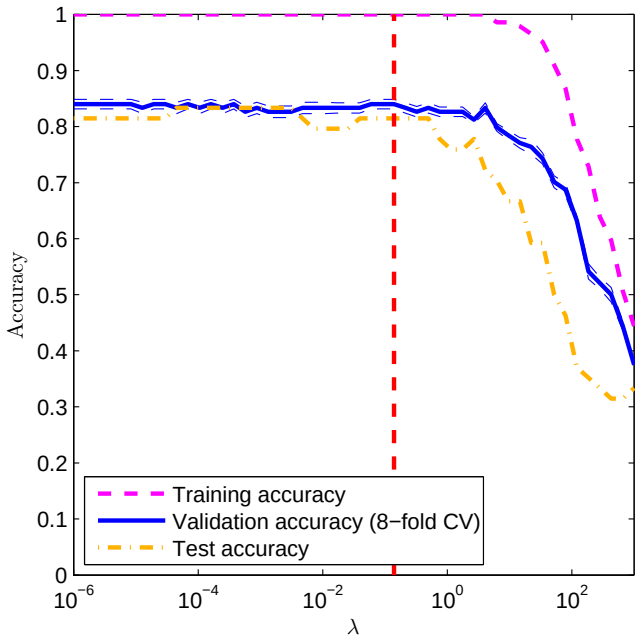

Figure 4.9: 14 CANCERS data: profiles of training error, 8-fold validation error and test error for different values of the regularization parameter. Standard error bands for the validation error are also shown. The vertical line corresponds to the least complex model maximizing the validation accuracy.

5

Client-server multi-task learning from distributed datasets

The solution of learning tasks by joint analysis of multiple datasets is receiving increasing attention in different fields and under various perspectives. Indeed, the information provided by data for a specific task may serve as a domain-specific inductive bias for the others. Combining datasets to solve multiple learning tasks is an approach known in the machine learning literature as *multi-task learning* or *learning to learn* [Thrun, 1996, Caruana, 1997, Thrun and Pratt, 1998, Baxter, 1997, Ben-David et al., 2002, Bakker and Heskes, 2003, Lawrence and Platt, 2004]. In this context, the analysis of the *inductive transfer* process and the investigation of general methodologies for the simultaneous learning of multiple tasks are important topics of research. Many theoretical and experimental results support the intuition that, when relationships exist between the tasks, simultaneous learning performs better than separate (single-task) learning [Schwaighofer et al., 2005, Yu et al., 2005, 2007, Xue et al., 2007, Bonilla et al., 2007, Argyriou et al., 2007, Bickel et al., 2008, Zhang et al., 2008, Qi et al., 2008, An et al., 2008]. Theoretical results include the extension to the multi-task setting of generalization bounds and the notion of VC-dimension [Baxter, 2000, Ben-David and Schuller, 2003, Maurer, 2006] as well as a methodology for learning multiple tasks exploiting unlabeled data in the semi-supervised setting [Ando and Zhang, 2005].

Information fusion from different but related datasets is widespread in econometrics and marketing analysis, where the goal is to learn user preferences by analyzing both user-specific information and information from related users, see e.g. [Srivastava and Dwivedi, 1971, Arora et al., 1998, Allenby and Rossi, 1999, Greene., 2002]. The so-called *conjoint analysis* aims to determine the features of a product that mostly influence customer's decisions. In the web, collaborative approaches to estimate user preferences have become standard methodologies in many commercial systems and social networks, under the name of *collaborative filtering* or *recommender systems*, see e.g. [Resnick and Varian, 1997]. Pioneering collaborative filtering systems include Tapestry [Goldberg et al., 1992], GroupLens [Resnick et al., 1994, Konstan et al., 1997], ReferralWeb [Kautz et al., 1997], PHOAKS [Terveen et al., 1997]. More recently, the collaborative filtering problem has been attacked with machine learning methodologies such as Bayesian networks [Breese et al., 1998], MCMC algorithms [Chen and

George, 1999], mixture models [Hofmann and Puzicha, 1999], dependency networks [Heckerman et al., 2000], maximum margin matrix factorization [Srebro et al., 2005].

Biomedicine is another field in which importance of combining datasets is especially evident. In pharmacological experiments, few training examples are typically available for a specific subject due to technological and ethical constraints [Carson et al., 1983, Jacquez, 1985]. To obviate this problem, the so-called *population method* has been studied and applied with success since the seventies in pharmacology [Sheiner et al., 1977, Beal and Sheiner, 1982, Yuh et al., 1994]. Population methods belong to the family of so-called *mixed-effect* statistical methods [Sheiner and Steimer, 2000], and are based on the knowledge that subjects, albeit different, belong to a population of similar individuals, so that data collected in one subject may be informative with respect to the others [Vozeh et al., 1996, Park et al., 1997]. Classical approaches postulate finite-dimensional nonlinear dynamical systems whose unknown parameters can be determined by means of optimization algorithms [Beal and Sheiner, 1992, Sheiner, 1994, Davidian and Giltinan, 1995, Aarons, 1999]. Other strategies include Bayesian estimation with stochastic simulation [Wakefield et al., 1994, Lunn et al., 2002, Gilks et al.] and nonparametric regression [Fattinger and Verotta, 1995, Magni et al., 2002, Neve et al., 2005, 2007, Pillonetto et al., 2009].

In the machine learning literature, much attention has been given in the last years to techniques based on regularization, such as kernel methods [Schölkopf and Smola, 2001] and Gaussian processes [Rasmussen and Williams, 2006]. The regularization approach is powerful and theoretically sound, having their mathematical foundation in the theory of inverse problems, statistical learning theory and Bayesian estimation [Aronszajn, 1950, Tikhonov and Arsenin, 1977, Poggio and Girosi, 1990, Wahba, 1990, Vapnik, 1998, Cucker and Smale, 2001]. The flexibility of kernel engineering allows for the estimation of functions defined on generic sets from arbitrary sources of data. The methodology has been recently extended to the multi-task setting. In [Evgeniou et al., 2005], a general framework to solve multi-task learning problems using kernel methods and regularization has been proposed, relying on the theory of reproducing kernel Hilbert spaces (RKHS) of vector-valued functions [Micchelli and Pontil, 2005].

In many applications (e-commerce, social network data processing, recommender systems), real-time processing of examples is required. On-line multi-task learning schemes find their natural application in data mining problems involving very large datasets, and are therefore required to scale well with the number of tasks and examples. In [Pillonetto et al., 2010], an on-line task-wise algorithm to solve multi-task regression problems has been proposed. The learning problem is formulated in the context of on-line Bayesian estimation, see e.g. [Opper, 1998, Csató and Opper, 2002], and Gaussian processes with suitable covariance functions are used to characterize a non-parametric mixed-effect model. One of the key features of the algorithm is the capability to exploit shared inputs between the tasks in order to reduce computational complexity. However, the algorithm in [Pillonetto et al., 2010] has a centralized structure in which tasks are sequentially analyzed, and is not able to address neither architectural issues regarding the flux of information nor privacy protection.

In this chapter, multi-task learning from distributed datasets is addressed using

a client-server architecture. In our scheme, clients are in a one-to-one correspondence with tasks and their individual database of examples. The role of the server is to collect examples from different clients in order to summarize their informative content. When a new example associated with any task becomes available, the server executes an on-line update algorithm. While in [Pillonetto et al., 2010] different tasks are sequentially analyzed, the architecture presented in this chapter can process examples coming in any order from different learning tasks. The summarized information is stored in a *disclosed database* whose content is available for download enabling each client to compute its own estimate exploiting the informative content of all the other datasets. Particular attention is paid to confidentiality issues, especially valuable in commercial and recommender systems, see e.g. [Ramakrishnan et al., 2001, Canny, 2002]. First, we require that each specific client cannot access other clients data. In addition, individual datasets cannot be reconstructed from the disclosed database. Two kind of clients are considered: *active* and *passive* ones. An active client sends its data to the server, thus contributing to the collaborative estimate. A passive client only downloads information from the disclosed database without sending its data. A regularization problem with a parametric bias term is considered in which a *mixed-effect kernel* is used to exploit relationships between the tasks. Albeit specific, the mixed-effect non-parametric model is quite flexible, and its usefulness has been demonstrated in several works [Neve et al., 2005, 2007, Lu et al., 2008, Pillonetto et al., 2010].

The chapter is organized as follows. Multi-task learning with regularized kernel methods is presented in section 5.1, where a class of mixed-effect kernels is also introduced. In section 5.2, an efficient centralized off-line algorithm for multi-task learning is described that solves the regularization problem of section 5.1. In section 5.3, a rather general client-server architecture is described, which is able to efficiently solve online multi-task learning from distributed datasets. The server-side algorithm is derived and discussed in subsection 5.3.1, while the client-side algorithm for both active and passive clients is derived in subsection 5.3.2. In section 5.4, a simulated music recommendation system is employed to test performances of our algorithm. The application to the analysis of real data from a multicentric clinical trial for an antidepressant drug is discussed in section 5.5. Conclusions (section 5.6) end the chapter.

Notational preliminaries

- For all $n \in \mathbb{N}$, let $[n] := \{1, 2, \ldots, n\}$.
- An (n, p) *index vector* is an object $k \in [n]^p$.
- Given $a \in \mathcal{X}^n$ and an (n, p) index vector k, let

$$a(k) := \begin{pmatrix} a_{k_1} & \cdots & a_{k_p} \end{pmatrix} \in \mathcal{X}^p.$$

- Given $\mathbf{A} \in \mathcal{X}^{n \times m}$ and two index vectors k^1 and k^2, that are (n, p_1) and

(m, p_2), respectively, let

$$\mathbf{A}(k^1, k^2) := \begin{pmatrix} a_{k_1^1 k_1^2} & \cdots & a_{k_1^1 k_{p_2}^2} \\ \vdots & \ddots & \vdots \\ a_{k_{p_1}^1 k_1^2} & \cdots & a_{k_{p_1}^1 k_{p_2}^2} \end{pmatrix} \in \mathcal{X}^{p_1 \times p_2}.$$

- Finally, let

$$\mathbf{A}(:, k^2) := \mathbf{A}([n], k^2), \quad \mathbf{A}(k^1, :) := \mathbf{A}(k^1, [m]).$$

5.1 Problem formulation

Let m denote the total number of tasks. For each task j, we have access to a set of ℓ_j input-output pairs

$$(x_{ij}, y_{ij}) \in (\mathcal{X} \times \mathbb{R}), \qquad i = 1, \dots \ell_j.$$

The aim is to learn m relationships $g_j : \mathcal{X} \to \mathbb{R}$. Task-labeling is a simple technique to reduce multi-task learning problems to single-task ones. Letting $\mathcal{T} = \{1, \dots, m\}$, a task label is an integer $t_i \in \mathcal{T}$ that identify the task to whom the i-th example belong. The overall available data can be viewed as a set of triples

$$(x_i, t_i, y_i) \in (\mathcal{X} \times \mathcal{T} \times \mathbb{R}), \qquad i = 1, \dots \ell,$$

where $\ell := \sum_{j=1}^{m} \ell_j$ denotes the overall number of examples. The correspondence between the individual datasets and the overall dataset can be expressed by using an (ℓ, ℓ_j) index vector k^j such that

$$t_{k_i^j} = j, \quad i = 1, \dots, \ell_j. \tag{5.1}$$

For instance, k_3^2 denote the position i in the overall dataset of the pair (x_{32}, y_{32}).

In this chapter, predictors are searched within suitable reproducing kernel Hilbert spaces (RKHS) of functions defined over $\mathcal{X} \times \mathcal{T}$, by solving a regularization problem of the form

$$g^* = \arg\min_{g \in \mathcal{H}} \left(\sum_{i=1}^{\ell} \frac{(y_i - g(x_i, t_i))^2}{2\lambda w_i} + \frac{\|g\|_{\mathcal{H}}^2}{2} \right),$$

where $\lambda > 0$ is a regularization parameter, and $w_i > 0, (i = 1, \dots, \ell)$.

We focus on a class of *mixed effect kernels* that can be expressed as a convex combination of a task-independent contribution and a task-specific one:

$$K((x_1, t_1), (x_2, t_2)) = \alpha \overline{K}(x_1, x_2) + (1 - \alpha) \sum_{j=1}^{m} K_T^j(t_1, t_2) \widetilde{K}^j(x_1, x_2), \tag{5.2}$$

$0 \leq \alpha \leq 1$. The rationale behind such structure is that predictors for a generic task j will result in the sum of a common task function and an individual shift,

see (5.3) below. Kernels $\overline{K}$ and $\widetilde{K}^j$ can possibly be all distinct. On the other hand, K_T^j are positive semi-definite "selector" kernels defined as

$$K_T^j(t_1, t_2) = \begin{cases} 1, & t_1 = t_2 = j; \\ 0, & \text{otherwhise} \end{cases}$$

The representer theorem gives the following expression for the optimum g^*:

$$g^*(x, t) = \sum_{i=1}^{\ell} c_i K_{(x_i, t_i)}(x, t).$$

The estimate $g_j^* := g^*(x, j)$ for the j-th task is defined to be the function obtained by plugging the corresponding task-label $t = j$ in the previous expression. As a consequence of the structure of K, the expression of g_j^* decouples into two parts:

$$g_j^*(x) = \bar{g}(x) + \widetilde{g}_j(x), \tag{5.3}$$

where

$$\bar{g}(x) = \alpha \sum_{i=1}^{\ell} c_i \overline{K}(x_i, x), \quad \widetilde{g}_j(x) = (1 - \alpha) \sum_{i \in k^j}^{\ell} c_i \widetilde{K}^j(x_i, x).$$

There exists an optimal coefficient vector c solve the linear system:

$$(\mathbf{K} + \lambda \mathbf{W})\, c = y, \tag{5.4}$$

where $\mathbf{W} = \mathrm{diag}\{w_1, \ldots, w_\ell\}$, and $\mathbf{K}$ is the kernel matrix associated with kernel K and all the input data. From (5.2), it follows that $\mathbf{K}$ has the following structure:

$$\mathbf{K} = \left(\alpha \overline{\mathbf{K}} + (1 - \alpha) \sum_{j=1}^{m} \mathbf{I}(:, k^j) \widetilde{\mathbf{K}}^j(k^j, k^j) \mathbf{I}(k^j, :) \right), \tag{5.5}$$

where $\overline{\mathbf{K}}$ and $\widetilde{\mathbf{K}}^j$ are the kernel matrices associated with the kernels $\overline{K}$ and $\widetilde{K}^j$ respectively.

Function $\bar{g}$ is independent of the task label and can be regarded as the estimate of an *average task*, whereas $\widetilde{g}_j$ is the estimate of an *individual shift*. The value α is related to the "shrinking" of the individual estimates toward the average task. When $\alpha = 1$, the same function is learned for all the tasks, as if all examples referred to an unique task (*pooled* approach). On the other hand, when $\alpha = 0$, all the tasks are learned independently (*separate* approach), as if tasks were not related at all.

5.2 Complexity reduction

In many applications of multi-task learning, some or all of the input data x_i are shared between the tasks so that the number of different basis functions appearing in the expansion (5.3) may be considerably less than ℓ. As explained below,

such feature can be exploited to derive efficient incremental online algorithms for multi-task learning. Let

$$c^j := c(k^j), \quad y^j := y(k^j), \quad w^j := w(k^j),$$

where index vectors k^j are defined in equation (5.1).

Introduce the set of *unique inputs*

$$\breve{\mathcal{X}} = \{\breve{x}_1, \ldots, \breve{x}_n\} \in \mathcal{X}^n, \quad \text{such that} \quad \breve{x}_i \neq \breve{x}_j, \quad \forall i \neq j,$$

where $n < \ell$ denote the number of unique inputs. For each task j, a new (n, ℓ_j) index vector h^j can be defined such that

$$x_{ij} = \breve{x}_{h_i^j}, \quad i = 1, \ldots, \ell_j.$$

The information contained in the index vectors h^j is equivalently expressed by a binary matrix $\mathbf{P} \in \{0,1\}^{\ell \times n}$, whose entries satisfy

$$p_{ij} = \begin{cases} 1, & x_i = \breve{x}_j \\ 0, & \text{otherwise.} \end{cases} \tag{5.6}$$

We have the following decompositions:

$$\overline{\mathbf{K}} = \mathbf{P}\breve{\mathbf{K}}\mathbf{P}^T, \quad \breve{\mathbf{K}} := \mathbf{L}\mathbf{D}\mathbf{L}^T, \tag{5.7}$$

where $\mathbf{L} \in \mathbb{R}^{n \times r}$, $\mathbf{D} \in \mathbb{R}^{r \times r}$ are suitable rank-r factors, $\mathbf{D}$ is diagonal, and $\breve{\mathbf{K}} \in \mathbb{R}^{n \times n}$ is the kernel matrix associated with $\overline{K}$ and the set of unique inputs. If $\overline{K}$ is strictly positive, then $\mathbf{L}$ can be taken as a full-rank lower triangular Cholesky factor, see e.g. [Golub and Van Loan, 1996]. Letting

$$\breve{c} := \mathbf{P}^T c, \tag{5.8}$$

the optimal estimates (5.3) can be rewritten in a compact form:

$$g_j^*(x) = \alpha \sum_{i=1}^{n} \breve{c}_i \overline{K}(\breve{x}_i, x) + (1-\alpha) \sum_{i=1}^{\ell_j} c_i^j \widetilde{K}^j(\breve{x}_{h_i^j}, x).$$

The following result shows that coefficient vectors c and $\breve{c}$ can be obtained by solving a system of linear equations involving only "small-sized" matrices so that complexity depends on the number of unique inputs rather than the overall number of examples.

Theorem 15. *Coefficient vectors c and $\breve{c}$, defined in (5.4) and (5.8), can be computed by Algorithm 4.*

Algorithm 4 is an off-line (centralized) procedure whose computational complexity scales with $O(n^3 m)$. In the next section, we derive a client-server on-line version of Algorithm 4 that preserves such complexity bound. Typically, this is much better than $O\left(\ell^3\right)$, the worst-case complexity of directly solving (5.4).

Algorithm 4 Centralized off-line algorithm.

1: **for** $j = 1 : m$ **do**
2: $\quad \mathbf{R}^j \leftarrow \left[(1-\alpha)\widetilde{\mathbf{K}}^j(k^j, k^j) + \lambda \mathbf{W}(k^j, k^j)\right]^{-1}$
3: **end for**
4: Compute factorization $\breve{\mathbf{K}} = \mathbf{L}\mathbf{D}\mathbf{L}^T$
5: $\breve{y} \leftarrow \sum_{j=1}^m \mathbf{L}^T(:, h^j)\mathbf{R}^j y^j$
6: $\mathbf{H} \leftarrow \left(\mathbf{D}^{-1} + \alpha \sum_{j=1}^m \mathbf{L}^T(:, h^j)\mathbf{R}^j \mathbf{L}(h^j, :)\right)^{-1}$
7: $z \leftarrow \mathbf{H}\breve{y}$
8: $\breve{c} \leftarrow$ Solution to $\left(\mathbf{D}\mathbf{L}^T\right)\breve{c} = z$.
9: **for** $j = 1 : m$ **do**
10: $\quad c^j \leftarrow \mathbf{R}^j \left[y^j - \alpha \mathbf{L}(h^j, :) z\right]$
11: **end for**

5.3 A client-server online algorithm

The main objective of the present section is to derive an incremental distributed version of Algorithm 4 having a client-server architecture. It is assumed that each client is associated with a different learning task, so that the terms "task" and "client" will be used interchangeably. The role of the server is twofold:

1. *Collecting* triples (x_i, y_i, w_i) (input-output-weight) from the clients and updating on-line all matrices and coefficients needed to compute estimates for all the tasks.
2. *Publishing* sufficient information so that any client (task) j can independently compute its estimate, possibly without sending data to the server.

On the other hand, each client j can perform two kind of operations:

1. *Sending* triples (x_i, y_i, w_i) to the server.
2. *Receiving* information from the server sufficient to compute its own estimate.

To preserve privacy, each client can neither access other clients data nor reconstruct their individual estimates. With reference to matrices $\breve{y}$, $\mathbf{H}$, $\mathbf{R}^j$, whose definition is given inside Algorithm 4, we have the following scheme:

- Undisclosed Information: $h^j, y^j, w^j, \mathbf{R}^j, j = 1, \ldots, m$.
- Disclosed Information: $\breve{\mathcal{X}}$, $\breve{y}$, $\mathbf{H}$.

The server update algorithm represents an incremental implementation of lines 1-6 of Algorithm 4, having access to all the information (disclosed and undisclosed). Server-side computations are described in subsection 5.3.1. The client-side algorithm represents the computation of lines 7-11 of Algorithm 4 distributed among the clients, where access to undisclosed information is denied. Client-side computations are described in subsection 5.3.2.

5.3.1 Server side

In order to formulate the algorithm in compact form, it is useful to introduce the functions "find" and "ker". Let

$$A(x) := \{i : x_i = x\}.$$

For any $p, q \in \mathbb{N}$, $x \in \mathcal{X}$, $X \in \mathcal{X}^p, Y \in \mathcal{X}^q$, let

$$\begin{array}{rcl} \text{find} : \mathcal{X} \times \mathcal{X}^p & \to & \{1, \ldots, p+1\} \\ \text{find}(x, X) & = & \begin{cases} p+1, & A(x) = \emptyset, \\ \min A(x), & A(x) \neq \emptyset. \end{cases} \\ \ker(\cdot, \cdot; K) : \mathcal{X}^p \times \mathcal{X}^q & \to & \mathbb{R}^{p \times q} \\ \ker(X, Y; K)_{ij} & = & K(x_i, y_j). \end{array}$$

The complete computational scheme is reported in Algorithm 5. The initialization is defined by resorting to empty matrices whose manipulation rules can be found in [Nett and Haddad, 1993]. In particular, all the matrices and vectors are all initialized to the empty matrix. It is assumed that function "ker" returns an empty matrix when applied to empty matrices.

Algorithm 5 is mainly based on the use of matrix factorizations and matrix manipulation lemmas in the Appendix. The rest of this subsection is an extensive proof devoted to show that the algorithm correctly updates all the relevant quantities when a new triple (x_i, y_i, w_i) becomes available from task j. Three cases are possible:

1. The input x_i is already among the inputs of task j.
2. The input x_i is not among the inputs of task j, but can be found in the common database $\breve{\mathcal{X}}$.
3. The input x_i is new.

Case 1: repetition within inputs of task j

The input x_i has been found in $\breve{\mathcal{X}}(h^j)$, so that it is also present in $\breve{\mathcal{X}}$. Thus, the flow of Algorithm 5 can be equivalently reorganized as in Algorithm 6. Let r denote the number of triples of the type (x, y_i, w_i) belonging to task j. These data can be replaced by a single triple (x, y, w) without changing the output of the algorithm. Letting

$$w := \left(\sum_{i=1}^{r} w_i^{-1} \right)^{-1}, \qquad y := w \sum_{i=1}^{r} y_i / w_i,$$

Algorithm 5 Server: receive (x_i, y_i, w_i) from client j and update the database.

$s = \text{find}\left(x_i, \breve{\mathcal{X}}\right)$
if $(s = n + 1)$ **then**
  $n \leftarrow n + 1$,
  $\breve{\mathcal{X}} \leftarrow \left\{\breve{\mathcal{X}}, x_i\right\}$,
  $\breve{y} \leftarrow \begin{pmatrix} \breve{y} \\ 0 \end{pmatrix}$,
  $\overline{k} \leftarrow \ker\left(x_i, \breve{\mathcal{X}}; \overline{K}\right)$,
  $r \leftarrow$ Solution to $\mathbf{L}\mathbf{D}r = \overline{k}([n-1])$,
  $\beta \leftarrow \overline{k}_n - r^T\mathbf{D}r$,
  $\mathbf{H} \leftarrow \text{diag}\{\mathbf{H}, \beta\}$
  $\mathbf{D} \leftarrow \text{diag}\{\mathbf{D}, \beta\}$
  $\mathbf{L} \leftarrow \begin{pmatrix} \mathbf{L} & 0 \\ r^T & 1 \end{pmatrix}$
end if
$p = \text{find}\left(x_i, \breve{\mathcal{X}}(h^j)\right)$
if $(p = \ell_j + 1)$ **then**
  $\ell_j \leftarrow \ell_j + 1$
  $h^j \leftarrow \begin{pmatrix} h^j & s \end{pmatrix}$
  $y^j \leftarrow \begin{pmatrix} y^j \\ y_i \end{pmatrix}$
  $w^j \leftarrow \begin{pmatrix} w^j \\ w_i \end{pmatrix}$
  $\widetilde{k} \leftarrow (1-\alpha) \cdot \ker\left(x_i, \breve{\mathcal{X}}(h^j); \widetilde{K}^j\right)$,
  $u \leftarrow \begin{pmatrix} \mathbf{R}^j\widetilde{k}([\ell_j - 1]) \\ -1 \end{pmatrix}$
  $\gamma \leftarrow 1/\left(\lambda w_i - u^T\widetilde{k}\right)$.
  $\mu \leftarrow \gamma u^T y^j$,
  $\mathbf{R}^j \leftarrow \text{diag}\left\{\mathbf{R}^j, 0\right\}$
else
  $u \leftarrow \mathbf{R}^j(:, p)$,
  $w_p^j \leftarrow w_p^j w_i / \left(w_p^j + w_i\right)$,
  $y_p^j \leftarrow y_p^j + w_p^j(y_i - y_p^j)/w_i$,
  $\gamma \leftarrow -\left[\lambda(w_p^j)^2 / \left(w_p^j - w_i\right) + r_{pp}^j\right]^{-1}$,
  $\mu \leftarrow w_p^j(y_i - y_p^j)/(w_i - w_p^j) + \gamma u^T y^j$,
end if
$\mathbf{R}^j \leftarrow \mathbf{R}^j + \gamma u u^T$
$v \leftarrow \mathbf{L}^T(:, h^j)u$
$\breve{y} \leftarrow \breve{y} + \mu v$,
$q \leftarrow \mathbf{H}v$,
$\mathbf{H} \leftarrow \mathbf{H} - (qq^T)/(\alpha\gamma)^{-1} + v^T q)$.

Algorithm 6 Server (Case 1)

1: $s = \text{find}\left(x_i, \breve{\mathcal{X}}\right)$
2: $p = \text{find}\left(x_i, \breve{\mathcal{X}}(h^j)\right)$
3: $u \leftarrow \mathbf{R}^j(:, p)$,
4: $w_p^j \leftarrow w_p^j w_i / \left(w_p^j + w_i\right)$,
5: $y_p^j \leftarrow y_p^j + w_p^j (y_i - y_p^j)/w_i$,
6: $\gamma \leftarrow -\left[\lambda (w_p^j)^2 / \left(w_p^j - w_i\right) + r_{pp}^j\right]^{-1}$,
7: $\mu \leftarrow w_p^j (y_i - y_p^j)/(w_i - w_p^j) + \gamma u^T y^j$,
8: $\mathbf{R}^j \leftarrow \mathbf{R}^j + \gamma u u^T$
9: $v \leftarrow \mathbf{L}^T(:, h^j) u$
10: $\breve{y} \leftarrow \breve{y} + \mu v$,
11: $q \leftarrow \mathbf{H} v$,
12: $\mathbf{H} \leftarrow \mathbf{H} - (q q^T)/((\alpha\gamma)^{-1} + v^T q)$.

the part of the empirical risk associated with these data can be rewritten as

$$\begin{aligned}\sum_{i=1}^{r} \frac{(y_i - g_j(x))^2}{2w_i} =& \frac{1}{2} \sum_{i=1}^{r} \left(\frac{y_i^2}{w_i} - 2 g_j(x) \frac{y_i}{w_i} + g_j(x)^2 \frac{1}{w_i} \right) \\ =& \frac{\left(g_j(x)^2 - 2 g_j(x) y\right)}{2w} + \sum_{i=1}^{r} \frac{y_i^2}{2w_i} \\ =& \frac{(y - g_j(x))^2}{2w} + A,\end{aligned}$$

where A is a constant independent of g. To recursively update w and y when a repetition is detected, notice that

$$w^{r+1} = \left(\sum_{i=1}^{r+1} \frac{1}{w_i} \right)^{-1} = \left(\frac{1}{w^r} + \frac{1}{w_{r+1}} \right)^{-1} = \frac{w^r w_{r+1}}{w^r + w_{r+1}} = w^r - \frac{(w^r)^2}{w^r + w_{r+1}}$$

$$y^{r+1} = w^{r+1} \sum_{i=1}^{r+1} \frac{y_i}{w_i} = \left(\frac{1}{w^r} + \frac{1}{w_{r+1}} \right)^{-1} \left(\frac{y^r}{w^r} + \frac{y_{r+1}}{w_{r+1}} \right) = y^r + \frac{w^r}{w_i} (y_i - y^r).$$

The variations can be expressed as:

$$\Delta w^{r+1} = -\frac{(w^r)^2}{w^r + w_{r+1}} = \frac{(w^{r+1})^2}{w^{r+1} - w_{r+1}},$$

$$\Delta y^{r+1} = \frac{w^r}{w_i} (y_i - y^r) = w^{r+1} \frac{y_{r+1} - y^{r+1}}{w_{r+1} - w^{r+1}}.$$

By applying these formulas to the p-th data of task j, lines 4, 5 of Algorithm 6 are obtained. In particular, the variation of coefficients w_p^j and y_p^j after the update can be written as follows:

$$\Delta w_p^j = \frac{(w_p^j)^2}{w_p^j - w_i}, \qquad \Delta y_p^j = w_p^j \frac{y_i - y_p^j}{w_i - w_p^j}.$$

At this point, we also need to modify matrix $\mathbf{R}^j$, since its definition (see Algorithm 4, line 2) depends on the weight coefficient w_p^j. To check that $\mathbf{R}^j$ is correctly updated by lines $3, 6, 8$ of Algorithm 6 observe that, applying Lemma 9, we have:

$$\mathbf{R}^j \leftarrow \left(\left(\mathbf{R}^j\right)^{-1} + \lambda e_p e_p^T \Delta w_p^j\right)^{-1} = \mathbf{R}^j - \frac{\mathbf{R}^j e_p e_p^T \mathbf{R}^j}{\lambda \Delta w_p^j + e_p^T \mathbf{R}^j e_p} = \mathbf{R}^j + \gamma u u^T.$$

Consider now the update of $\breve{y}$. Since y^j has already been modified, the previous y^j is given by $y^j - e_p \Delta y_p^j$. Recalling the definition of $\breve{y}$ in Algorithm 4, and the update for $\mathbf{R}^j$, we have

$$\breve{y} \leftarrow \sum_{k \neq j}^{m} \mathbf{L}^T(:, h^k)\mathbf{R}^k y^k + \mathbf{L}^T(:, h^j)\left(\mathbf{R}^j + \gamma u u^T\right) y^j.$$

Using the definition of μ and u in Algorithm 6, we have

$$\begin{aligned}\left(\mathbf{R}^j + \gamma u u^T\right) y^j &= \left(\mathbf{R}^j + \gamma u u^T\right)\left(y^j - \Delta y_p^j e_p + \Delta y_p^j e_p\right) \\ &= \mathbf{R}^j\left(y^j - \Delta y_p^j e_p\right) + \left(\Delta y_p^j + \gamma u^T y^j\right) u \\ &= \mathbf{R}^j\left(y^j - \Delta y_p^j e_p\right) + \mu u,\end{aligned}$$

so that

$$\breve{y} \leftarrow \breve{y} + \mu \mathbf{L}^T(:, h^j) u.$$

By defining v as in line 9 of Algorithm 6, the update of line 10 is obtained. Finally, we show that $\mathbf{H}$ is correctly updated. Recall, from Algorithm 4 that

$$\mathbf{H} = \left(\mathbf{D}^{-1} + \alpha \sum_{j=1}^{m} \mathbf{L}^T(:, h^j)\mathbf{R}^j \mathbf{L}(h^j, :)\right)^{-1}.$$

In view of lines 8 and 9 of Algorithm 6, we have

$$\mathbf{H} \leftarrow \left(\mathbf{H}^{-1} + \alpha\gamma v v^T\right)^{-1}.$$

Lines 11 and 12 follows by applying Lemma 9 to the last expression.

Case 2: repetition in $\breve{\mathcal{X}}$.

Since x_i belongs to $\breve{\mathcal{X}}$ but not to $\breve{\mathcal{X}}(h^j)$, we have

$$s \neq n+1, \qquad p = \ell_j + 1.$$

The flow of Algorithm 5 can be organized as in Algorithm 7.

First, vectors h^j, y^j and w^j must be properly enlarged as in lines 3-6. Then, recalling the definition of $\mathbf{R}^j$ in Algorithm 4, we have:

$$(\mathbf{R}^j)^{-1} \leftarrow \begin{pmatrix} (\mathbf{R}^j)^{-1} & \widetilde{k}([\ell_j - 1]) \\ \widetilde{k}([\ell_j - 1])^T & \widetilde{k}_{\ell_j} + \lambda w_i \end{pmatrix}$$

Algorithm 7 Server (Case 2)

1: $s = \text{find}\left(x_i, \breve{\mathcal{X}}\right)$
2: $p = \text{find}\left(x_i, \breve{\mathcal{X}}(h^j)\right)$
3: $\ell_j \leftarrow \ell_j + 1$
4: $h^j \leftarrow \begin{pmatrix} h^j & s \end{pmatrix}$
5: $y^j \leftarrow \begin{pmatrix} y^j \\ y_i \end{pmatrix}$
6: $w^j \leftarrow \begin{pmatrix} w^j \\ w_i \end{pmatrix}$
7: $\widetilde{k} \leftarrow (1-\alpha) \cdot \ker\left(x_i, \breve{\mathcal{X}}(h^j); \widetilde{K}^j\right)$,
8: $u \leftarrow \begin{pmatrix} \mathbf{R}^j \widetilde{k}([\ell_j - 1]) \\ -1 \end{pmatrix}$
9: $\gamma \leftarrow 1/\left(\lambda w_i - u^T \widetilde{k}\right)$.
10: $\mathbf{R}^j \leftarrow \text{diag}\left\{\mathbf{R}^j, 0\right\} + \gamma u u^T$
11: $v \leftarrow \mathbf{L}^T(:, h^j)u$
12: $\mu \leftarrow \gamma u^T y^j$,
13: $\breve{y} \leftarrow \breve{y} + \mu v$,
14: $q \leftarrow \mathbf{H} v$,
15: $\mathbf{H} \leftarrow \mathbf{H} - (qq^T)/((\alpha\gamma)^{-1} + v^T q)$.

The update for $\mathbf{R}^j$ in lines 7-10 is obtained by applying Lemma 10.

Consider now the update of $\breve{y}$. Recall that h^j and y^j have already been updated. By the definition of $\breve{y}$ and in view of line 10 of Algorithm 7, we have

$$\begin{aligned}\breve{y} &\leftarrow \sum_{k \neq j}^{m} \mathbf{L}^T(:, h^k)\mathbf{R}^k y^k + \mathbf{L}^T(:, h^j)\left[\text{diag}\left\{\mathbf{R}^j, 0\right\} + \gamma u u^T\right] y^j \\ &= \breve{y} + \gamma(u^T y^j)\mathbf{L}^T(:, h^j)u.\end{aligned}$$

The update in lines 11-13 immediately follows. Finally, the update in lines 14-15 for $\mathbf{H}$ follows by applying Lemma 10, as in Case 1.

Case 3: x_i is a new input.

Since x_i is a new input, we have

$$s = n + 1, \qquad p = \ell_j + 1.$$

The flow of Algorithm 5 can be reorganized as in Algorithm 8. The final part of Algorithm 8 coincides with Algorithm 7. In addition, the case of new input also requires updating factors $\mathbf{D}$ and $\mathbf{L}$. If $\overline{K}$ is strictly positive, then $\mathbf{D}$ is diagonal and $\mathbf{L}$ can be chosen as a lower triangular Cholesky factor. If $\overline{K}$ is not strictly positive, other kinds of decompositions can be used. In particular, for the linear kernel $\overline{K}(x_1, x_2) = \langle x_1, x_2 \rangle_2$ over $\mathbb{R}^r$, $\mathbf{D}$ and $\mathbf{L}$ can be taken, respectively, equal

Algorithm 8 Server (Case 3)

1: $n \leftarrow n+1$
2: $\breve{\mathcal{X}} \leftarrow \left\{\breve{\mathcal{X}}, x_i\right\}$.
3: $\overline{k} \leftarrow \ker\left(x_i, \breve{\mathcal{X}}; \overline{K}\right)$,
4: $r \leftarrow$ Solution to $\mathbf{L}\mathbf{D}r = \overline{k}([n-1])$,
5: $\beta \leftarrow \overline{k}_n - r^T\mathbf{D}r$,
6: $\mathbf{D} \leftarrow \operatorname{diag}\{\mathbf{D}, \beta\}$
7: $\mathbf{L} \leftarrow \begin{pmatrix} \mathbf{L} & 0 \\ r^T & 1 \end{pmatrix}$
8: $\breve{y} \leftarrow \begin{pmatrix} \breve{y} \\ 0 \end{pmatrix}$
9: $\mathbf{H} \leftarrow \operatorname{diag}\{\mathbf{H}, \beta\}$
10: Call Algorithm 7.

to the identity and $\breve{\mathcal{X}}$. Recalling equation (5.7), we have

$$\begin{aligned}\breve{\mathbf{K}} \leftarrow \begin{pmatrix} \breve{\mathbf{K}} & \overline{k}([n-1]) \\ \overline{k}([n-1])^T & \overline{k}_n \end{pmatrix} &= \begin{pmatrix} \mathbf{L}\mathbf{D}\mathbf{L}^T & \overline{k}([n-1]) \\ \overline{k}([n-1])^T & \overline{k}_n \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{L} & 0 \\ r^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{D} & 0 \\ 0 & \beta \end{pmatrix} \begin{pmatrix} \mathbf{L} & 0 \\ r^T & 1 \end{pmatrix}^T,\end{aligned}$$

with r and β as in lines 4-5. Finally, updates for $\breve{y}$ and $\mathbf{H}$ are similar to that of previous Case 2, once the enlargements in lines 8 and 9 are made.

5.3.2 Client side

It turns out that each client can compute its own estimate $\hat{f}_j$ by only accessing disclosed data $\breve{\mathcal{X}}$, $\breve{y}$, and $\mathbf{H}$. It is not even necessary to know the overall number m of tasks, nor their kernels $\widetilde{K}^j$. First of all, we show that vector $\breve{c}$ can be computed by knowing only disclosed data. From Algorithm 4, we have

$$\left(\mathbf{D}\mathbf{L}^T\right)\breve{c} = \mathbf{H}\breve{y}.$$

Apparently, the right-hand side can be computed by just knowing $\breve{y}$ and $\mathbf{H}$. Moreover, from equation (5.7) one can see that factors $\mathbf{L}$ and $\mathbf{D}$ can be computed by knowing matrix $\breve{\mathbf{K}}$ which, in turn, can be computed by just knowing vector $\breve{\mathcal{X}}$. In practice, factors $\mathbf{L}$ and $\mathbf{D}$ can be incrementally computed by lines 1-7 of Algorithm 9. Notice that $\breve{c}$ is only needed at prediction time. While it is possible to keep it updated on the server, this would require solving a linear system for each new example to compute quantities that are possibly overwritten before being used. To reduce computation load on the server, it is preferable that each client computes by itself these quantities, whenever it needs predictions.

As mentioned in the introduction, two kind of clients are considered.

- An *active* client j, which sends its own data to the server.

- A *passive* client j, which does not send its data.

Passive clients must run a local version of the update algorithm before obtaining coefficient vectors, since the information contained into the disclosed database doesn't take into account their own data. For such reason, they need to know matrix $\mathbf{H}$ and vector $\breve{y}$ separately.

Once the disclosed data and vector $\breve{c}$ have been obtained, each client still needs the individual coefficients vector c^j in order to perform predictions. In this respect, notice that line 10 of Algorithm 4 decouples with respect to the different tasks:

$$c^j \leftarrow \mathbf{R}^j \left(y^j - \alpha \mathbf{L}(h^j, :) z\right),$$

so that c^j can be computed by knowing only disclosed data together with private data of task j. This is the key feature that allows a passive client to perform predictions without disclosing its private data and exploiting the information contained in all the other datasets. Client side computations are summarized in Algorithm 9.

Algorithm 9 (Client j) Receive $\breve{x}$, $\breve{y}$ and $\mathbf{H}$ and evaluate $\breve{c}$ and c^j

1: **for** $i = 1 : n$ **do**
2: $\quad \overline{k} \leftarrow \ker\left(\breve{x}_i, \breve{\mathcal{X}}([i]); \overline{K}\right)$,
3: $\quad r \leftarrow$ Solution to $\mathbf{L}\mathbf{D}r = \overline{k}([i-1])$,
4: $\quad \beta \leftarrow \overline{k}_i - r^T \mathbf{D} r$,
5: $\quad \mathbf{D} \leftarrow \operatorname{diag}\{\mathbf{D}, \beta\}$,
6: $\quad \mathbf{L} \leftarrow \begin{pmatrix} \mathbf{L} & 0 \\ r^T & 1 \end{pmatrix}$,
7: **end for**
8: **if** Passive **then**
9: $\quad$ **for** $i = 1 : \ell_j$ **do**
10: $\quad\quad$ Run Algorithm 5 with (x_{ij}, y_{ij}, w_{ij}).
11: $\quad$ **end for**
12: **end if**
13: $z \leftarrow \mathbf{H}\breve{y}$
14: $\breve{c} \leftarrow$ Solution to $\left(\mathbf{D}\mathbf{L}^T\right)\breve{c} = z$.
15: $c^j \leftarrow \mathbf{R}^j \left(y^j - \alpha \mathbf{L}(h^j, :) z\right)$.

5.4 Illustrative example: music recommendation

In this section, we apply the algorithm presented in the chapter to a simulated music recommendation problem, which consists in predicting preferences of several virtual users for a set of artists. Artist data were obtained from the May 2005 AudioScrobbler Database dump (`http://www-etud.iro.umontreal.ca/~bergstrj/audioscrobbler_data.html`) which is the last dump released by AudioScrobbler/LastFM under Creative Commons license. LastFM is an internet radio that provides individualized broadcasts based on user preferences. The database dump includes users playcounts and artists names so that it is

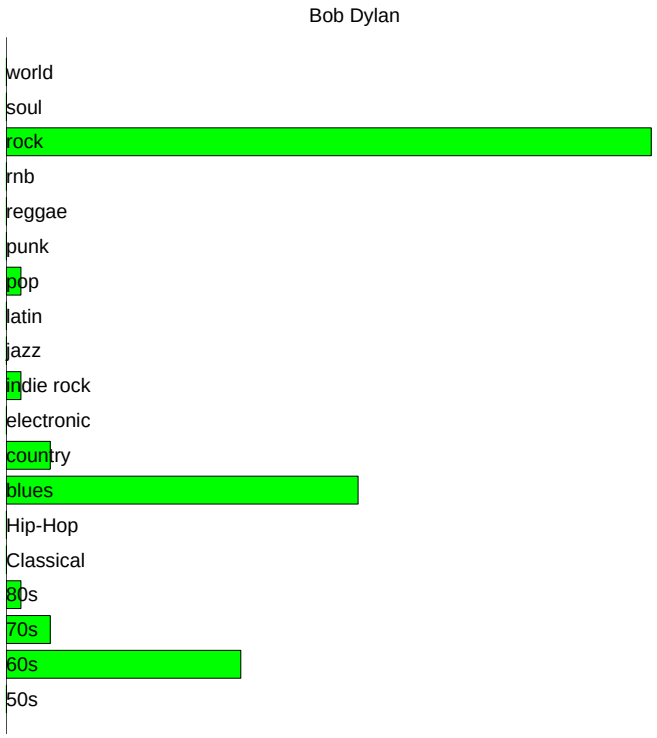

Figure 5.1: Example of artist tagging

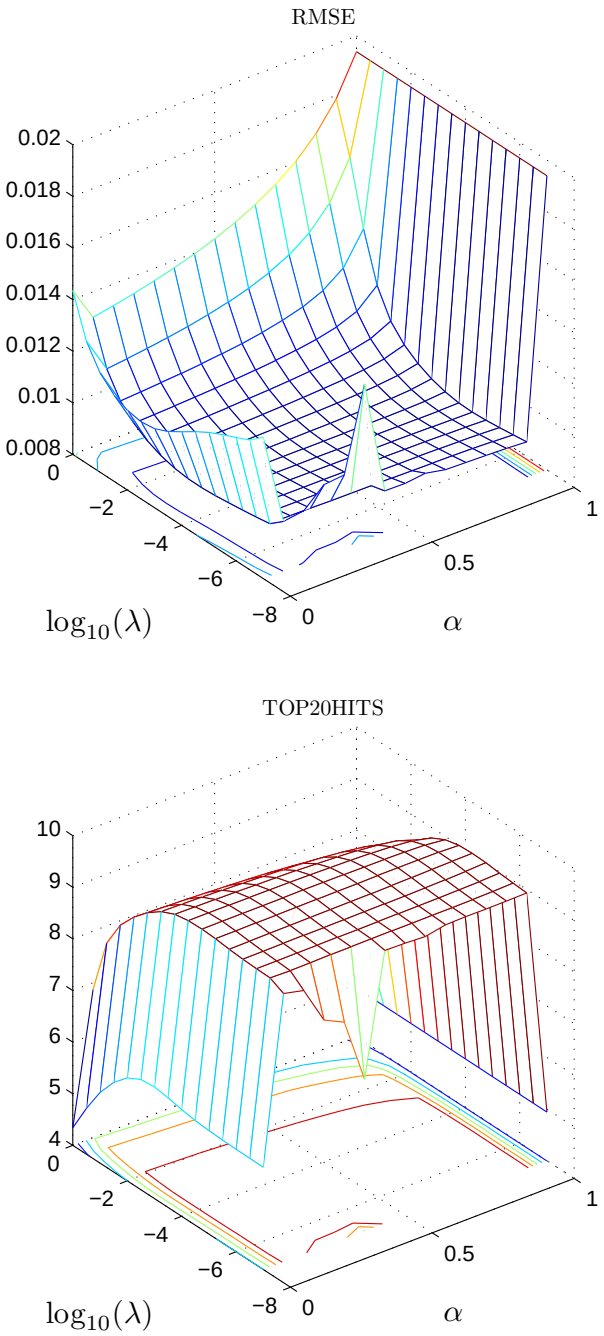

Figure 5.2: Average TOP20HITS and RMSE against α and λ.

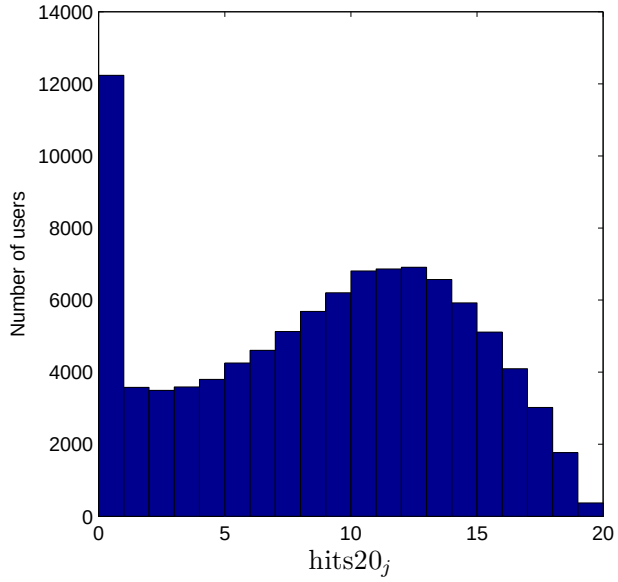

Figure 5.3: Distribution of hits20$_j$ in correspondence with α^* and λ^* achieving optimal $RMSE$.

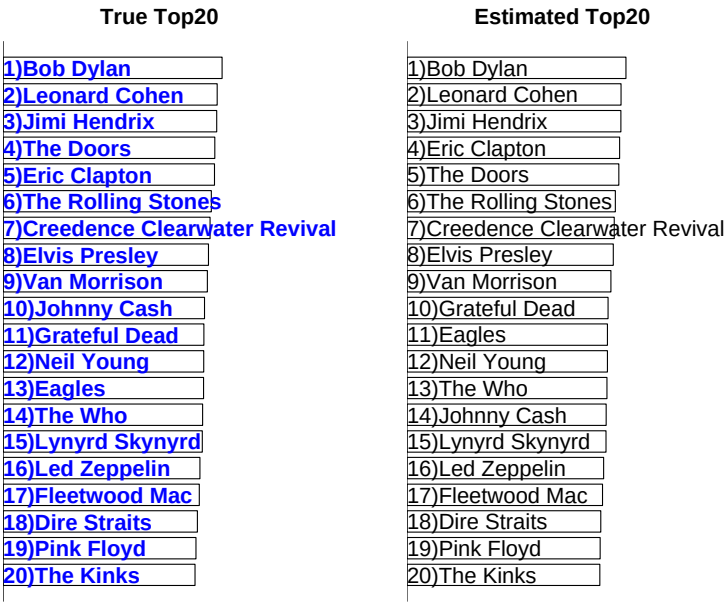

Figure 5.4: True and estimated Top20 for the "average user".

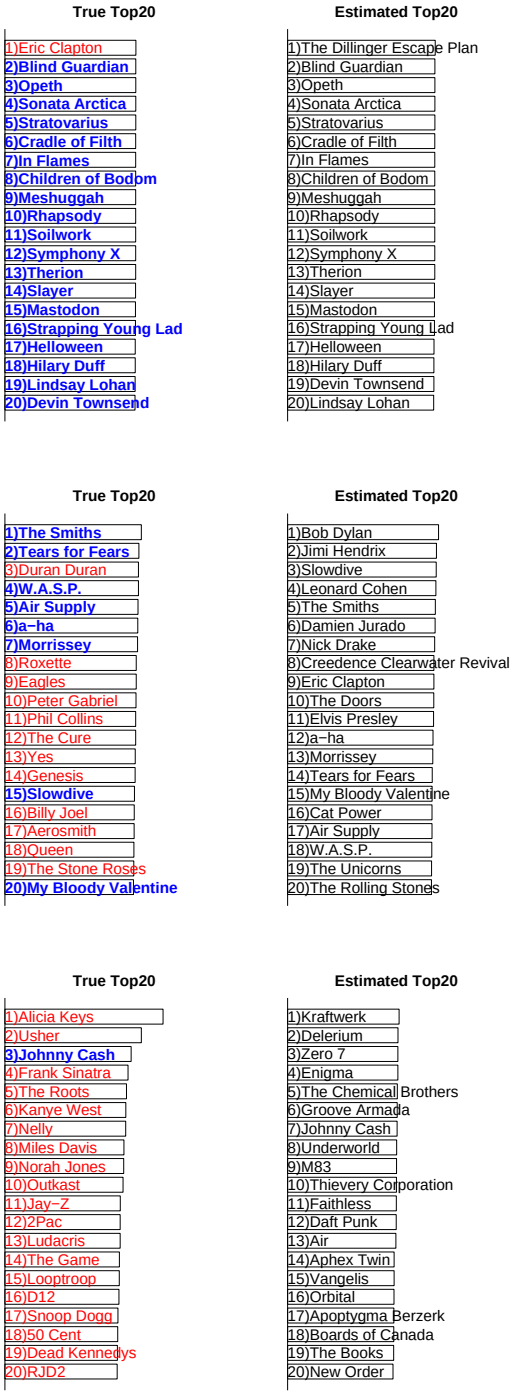

Figure 5.5: True and estimated Top20 for three representative virtual users.

possible to sort artists according to overall number of playcounts. After sorting according to decreasing playcounts, 489 top ranking artists were selected. The input set $\mathcal{X}$ is therefore a set of 489 items. The tasks are associated with user preference functions. More precisely, normalized preferences of user j over the entire set of artists are expressed by functions $g_j : \mathcal{X} \to \mathbb{R}$ that are to be learnt from data.

The simulated music recommendation system relies on music type classification expressed by means of tags (rock, pop, etc). The i-th artist is associated with a vector $x_i \in [0,1]^{19}$ of 19 *tags*, whose values were obtained by querying the LastFM web site (http://www.lastfm.com) on September 22, 2008. In Figure 5.1, the list of the tags considered in this experiment, together with an example of artist tagging are reported. Vectors x_i were normalized to lie on the unit hyper-sphere, i.e. $\|x_i\|_2 = 1$. Tag information is used to build a mixed-effect kernel over $\mathcal{X}$, where

$$\overline{K}(x_i, x_j) = e^{\langle x_i, x_j \rangle_2}, \qquad \widetilde{K}(x_i, x_j) = \langle x_i, x_j \rangle_2.$$

These kernels were also used to generate synthetic users. First, an "average user" was generated by drawing a function $\overline{h} : \mathcal{X} \to \mathbb{R}$ from a Gaussian process with zero mean and auto-covariance $\overline{K}$. Then, $m = 10^5$ virtual user's preferences were generated as

$$g_j = 0.25\overline{h} + 0.75\widetilde{h}_j,$$

where $\widetilde{h}_j$ were drawn from a Gaussian process with zero mean and auto-covariance $\widetilde{K}$. For the j-th virtual user, $\ell_j = 5$ artists x_{ij} were uniformly randomly sampled from the input set, and corresponding noisy outputs y_{ij} generated as

$$y_{ij} = g_j(x_{ij}) + \epsilon_{ij},$$

where ϵ_{ij} are i.i.d. Gaussian errors with zero mean and standard deviation $\sigma = 0.01$. Performance is evaluated by both the average root mean squared error

$$\mathrm{RMSE} = \frac{1}{m}\sum_{j=1}^{m}\sqrt{\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|}(g_j(x_i) - g_j^*(x_i))^2},$$

and the average number of hits within the top 20 ranked artists, defined as

$$\mathrm{TOP20HITS} = \frac{1}{m}\sum_{i=1}^{m}\mathrm{hits20}_j,$$
$$\mathrm{hits20}_j := \left|\mathrm{top20}(g_j) \cap \mathrm{top20}(g_j^*)\right|,$$

where $\mathrm{top20} : \mathcal{H} \to \mathcal{X}^{20}$ returns the sorted vector of 20 inputs with highest scores, measured by a function $g \in \mathcal{H}$.

Learning was performed for 15 values of the shrinking parameter α linearly spaced in $[0,1]$ and 15 values of the regularization parameter λ logarithmically spaced in the interval $\left[10^{-7}, 10^{0}\right]$, see Figure 5.2. The multi-task approach, i.e. $0 < \alpha < 1$ outperforms both the separate ($\alpha = 0$) and pooled ($\alpha =$

1) approaches. Interestingly, performance remains fairly stable for a range of values of α. Figure 5.3 shows the distribution of hits20_j over the users in correspondence with α^* and λ^* achieving the optimal RMSE. Although α^* and λ^* were selected so as to minimize the RMSE, good performances are obtained also with respect to the TOP20HITS score which is 9.4175 (meaning that, on the average, about 9 artists among the top-20 are correctly retrieved). Finally, true and estimated top-20 hits are reported for the average user (Figure 5.4) and three representative users chosen so as to give examples of good, average, and bad prediction performances (Figure 5.5). Artists of the true top-20 that are correctly retrieved in the estimated top-20 are reported in bold-face.

Concerning the computational burden, it is worth observing that, without exploiting the presence of repeated inputs and the mixed-effect structure of the kernel, the complexity of a naive approach would be of the order of the cube of the overall number of examples, that is $(5 \cdot 10^5)^3$. Conversely, the complexity of the approach proposed in this chapter scales with $n^3 m$, where n is the number of unique inputs and m the number of tasks (in our example, n is bounded by the cardinality of the input set $|\mathcal{X}| = 489$, and $m = 10^5$).

5.5 Multicentric pharmacological experiment

In this section, we consider data coming from a double-blind clinical trial (Study 810) for testing efficacy of paroxetine, an antidepressant drug [Merlo-Pich and Gomeni, 2008, Gomeni et al., 2009]. In this trial, patients suffering from major depressive disorder were either treated with placebo (placebo arm) or administered paroxetine at two different doses (treatment arms) for 10 weeks. Visits at weeks 0, 1, 2, 3, 4, 6, 8, were planned. During each visit, the patient was interviewed and a questionnaire filled. Based on the scores of the questionnaire items, a global score (the so-called HAMD score) which measures the degree of depression (the higher the score the more severe the depression) was assigned. In this way, each patient is associated with the time profile of her/his scores. An effective drug is expected to result in a suitable decrease of the HAMD score, measured by the difference between the score at the beginning (week 0) and that at the end of the study (week 10).

In psychiatric trials, dropouts are rather common, meaning that several patients may abandon the study before its completion. Since dropout occurrences are usually correlated with the response (e.g. patients whose HAMD score does not decrease are more likely to drop out), neglecting dropouts from the calculation of the average HAMD decrease would bias the result of the trial. Hence, it is necessary to reconstruct the missing observations (especially the final one at week 8) of dropouts so that they can be included in the global efficacy assessment. The conventional approach (LOCF-Last Observation Carried Forward) uses a heuristic imputation scheme that assigns the last available HAMD score to all visits following dropout occurrence. Since this solution is far from being satisfactory [Hu and Sale, 2003], more recent approaches [Gomeni et al., 2009] adopt a two-step procedure based on a parametric mathematical model of patient response: first, individual model parameters are estimated from the whole dataset and then they are used to compute the complete profile of each individual pa-

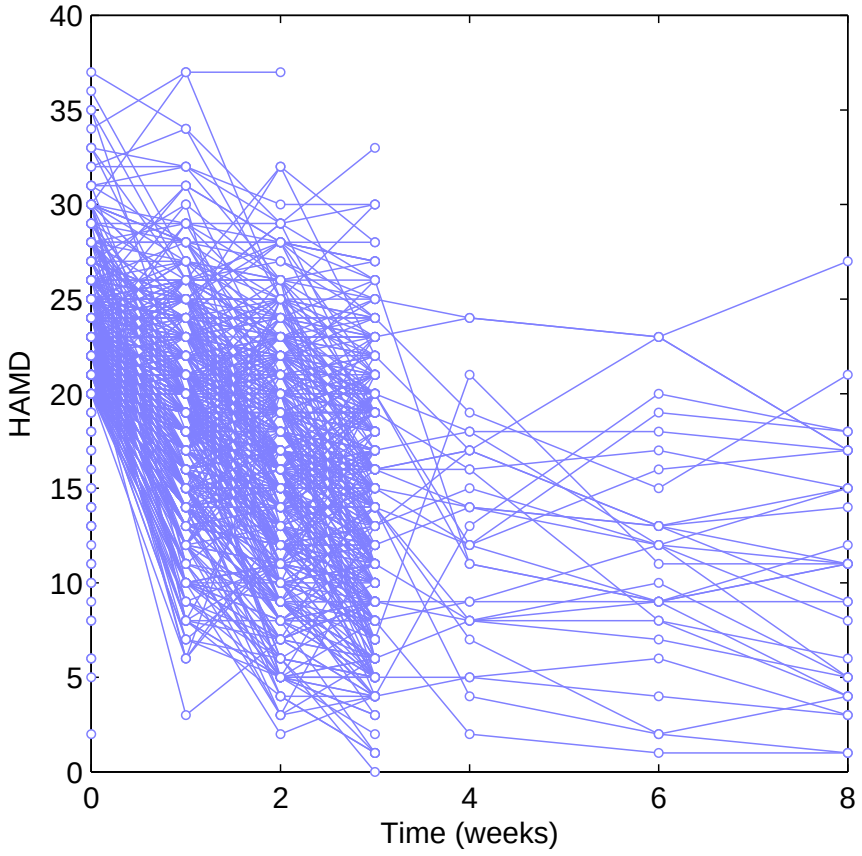

Figure 5.6: Pharmacological experiment: training data for 494 patients. Only a subset of 44 patients has been fully sampled, while for the others only measurements taken before the 4th week are available.

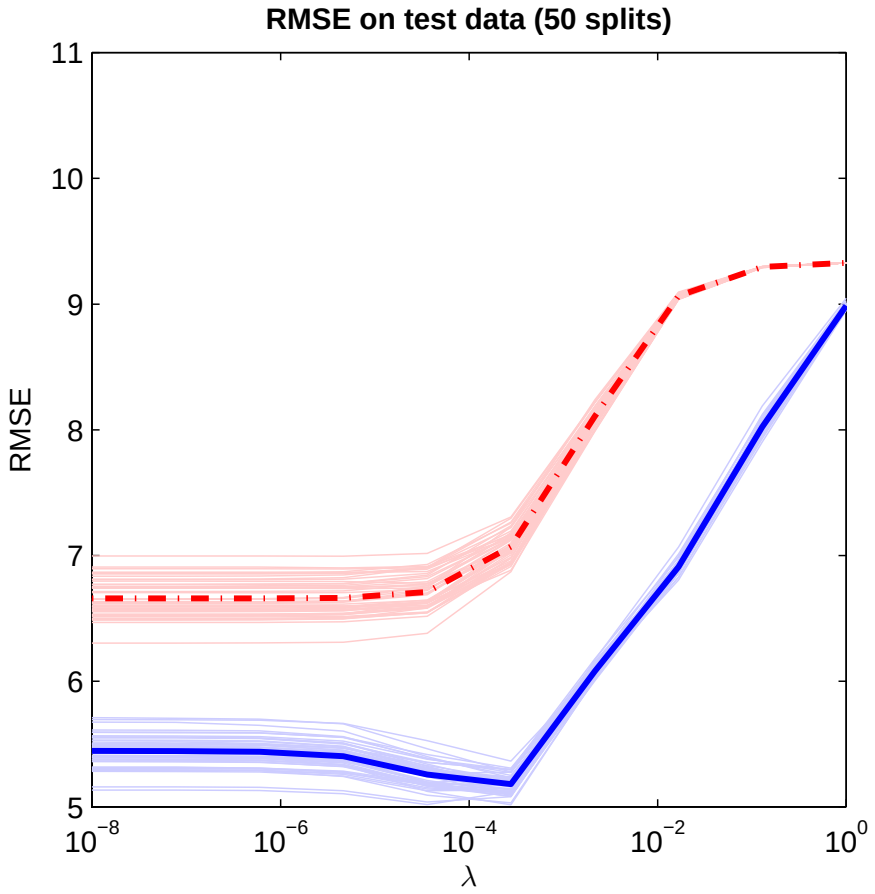

Figure 5.7: Pharmacological experiment: RMSE on the test data. The thick lines are averages over 50 randomly extracted training subsets, using a separate single-task method (dash-dotted thick line), and the multi-task learning method (continuous thick line).

tient. The main difficulty lies in the choice of the mathematical model of drug response, especially in the psychiatric field where a physiologically-grounded mechanistic description of drug effects is hardly possible.

The problem of reconstructing the missing scores of dropout patients is well suited to be reformulated as a multi-task learning problem, in which the different tasks correspond to the different patients enrolled in the study. A kernel-based approach offers two major advantages: it is flexible enough to accommodate a large variety of response shapes and there is no need to assume a particular structural model which could be difficult to justify. In the following, our multi-task learning method is applied to Study 810, in order to assess its ability to reconstruct responses of dropout patients compared to the use of a single-task method. This kind of clinical trials are multicentric, with patients enrolled by many different clinical centers. Therefore, the client-server architecture developed in the present chapter would allow any clinical center to complete the responses of its dropouts patients as if all the data were available, but without actually accessing confidential data of patients enrolled by the other centers.

The overall dataset contains 2855 measurements for 494 different patients. First of all, we extracted a test set containing 1012 examples which includes all the measurements taken after the 3th week for a subset of 450 randomly chosen patients. The remaining 1843 examples are plotted in Figure 5.6. To robustly assess performances, 50 additional random subsets have been extracted from the 1843 examples using a 30/70 splitting rule, namely 553 randomly chosen examples are removed, while the remaining 1290 examples are used for training. In order to reconstruct the individual response curves, we adopt the multi-task learning method proposed in this chapter with a mixed-effect kernel of the form (5.2), where both $\overline{K}$ and $\widetilde{K}^j$ are chosen as linear splines kernels of the type

$$\overline{K}(x_1, x_2) = \widetilde{K}^j(x_1, x_2) = 1 + \min\{x_1, x_2\},$$

and α has been fixed to 0.5. For each of the 50 splits, the solution is computed in correspondence with 10 different values of λ on a logarithmic scale in the interval $\left[10^{-8}, 1\right]$. Results are compared with independent learning of the response curves with a linear spline kernel, which corresponds to fixing $\alpha = 0$ in the mixed-effect kernel. Figure 5.7 reports the RMSE on the test data over the 50 dataset splits for both methods. The multi-task method outperforms the single-task approach for all the values of the regularization parameter.

5.6 Conclusions

Recent studies have highlighted the potentialities of kernel methods applied to multi-task learning, but their effective implementation involves the solution of architectural and complexity issues. In this chapter, emphasis is posed on the architecture with reference to learning from distributed datasets. For a general class of kernels with a "mixed-effect" structure it is shown that the optimal solution can be given a collaborative client-server architecture that enjoys favorable computational and confidentiality properties. By interacting with the server, each client can solve its own estimation task while taking advantage of all the data from the other clients without having any direct access to them. Client's

privacy is preserved, since both active and passive clients are allowed by the architecture. The former are those that agree to send their data to the server while the latter only exploit information from the server without disclosing their private data. The proposed architecture has several potential applications ranging from biomedical data analysis (where privacy issues are crucial) to web data mining. The simulated music recommendation system shows that, when the learning tasks are similar enough and the kernel is chosen well, combining estimates leads to good generalization performances even when a very small number of examples for each task is available. The use of kernels allows to take advantage of meta-data to improve estimates, and computational complexity scales favorably with the number of users.

Bibliography

L. Aarons. Software for population pharmacokinetics and pharmacodynamics. *Clinical Pharmacokinetics*, 36(4):255–264, 1999.

G. M. Allenby and P. E. Rossi. Marketing models of consumer heterogeneity. *Journal of Econometrics*, 89(1):57–78, 1999.

Q. An, C. Wang, I. Shterev, E. Wang, L. Carin, and D. Dunson. Hierarchical kernel stick-breaking process for multi-task image analysis. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 17–24. Omnipress, 2008.

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, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 25–32. MIT Press, Cambridge, MA, USA, 2007.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

N. Arora, G. M. Allenby, and J. Ginter. A hierarchical Bayes model of primary and secondary demand. *Marketing Science*, 17(1):29–44, 1998.

B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

S. Beal and L. Sheiner. *NONMEM User's Guide*. NONMEM Project Group, University of California, San Francisco, 1992.

S. L. Beal and L. B. Sheiner. Estimating population kinetics. *Critical Reviews in Biomedical Engineering*, 8(3):195–222, 1982.

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proceedings of Computational Learning Theory (COLT)*, 2003.

S. Ben-David, J. Gehrke, and R. Schuller. A theoretical framework for learning from a pool of disparate data sources. In *Proceedings of the The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 443–449, 2002.

S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer. Multi-task learning for HIV therapy screening. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 56–63. Omnipress, 2008.

E. V. Bonilla, F. V. Agakov, and C. K. I. Williams. Kernel multi-task learning using task-specific features. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, New York, NY, USA, 2002. ACM.

E. R. Carson, C. Cobelli, and L. Finkelstein. *The Mathematical Modeling of Metabolic and Endocrine Systems*. New York: Wiley, 1983.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

Y.-H. Chen and E. George. A bayesian model for collaborative filtering. In *Online Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, 1999.

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39:1–49, 2001.

M. Davidian and D. M. Giltinan. *Nonlinear Models for Repeated Measurement Data*. Chapman and Hall, 1995.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

K. E. Fattinger and D. Verotta. A nonparametric subject-specific population method for deconvolution: I. description, internal validation and real data examples. *Journal of Pharmacokinetics and Biopharmaceutics*, 23:581–610, 1995.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

G. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 1996.

R. Gomeni, A. Lavergne, and E. Merlo-Pich. Modelling placebo response in depression trials using a longitudinal model with informative dropout. *European Journal of Pharmaceutical Sciences*, 36(1):4–10, 2009.

W. Greene. *Econometric Analysis*. Prentice Hall, 5 edition, 2002.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

C. Hu and M. E. Sale. A joint model for nonlinear longitudinal data with informative dropout. *Journal of Pharmacokinetics and Pharmacodynamics*, 30(1):83–103, 2003.

J. A. Jacquez. *Compartmental analysis in biology and medicine*. University of Michigan Press, Ann Arbor, 1985.

H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In *Proceedings of the 21th International Conference in Machine Learning (ICML 2004)*, volume 69, page 65, 2004.

Z. Lu, T. Leen, Y. Huang, and D. Erdogmus. A reproducing kernel Hilbert space framework for pairwise time series distances. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 624–631. Omnipress, 2008.

D. J. Lunn, N. Best, A. Thomas, J. C. Wakefield, and D. Spiegelhalter. Bayesian analysis of population PK/PD models: general concepts and software. *Journal of Pharmacokinetics Pharmacodynamics*, 29(3):271–307, 2002.

P. Magni, R. Bellazzi, G. De Nicolao, I. Poggesi, and M. Rocchetti. Nonparametric AUC estimation in population studies with incomplete sampling: a Bayesian approach. *Journal of Pharmacokinetics Pharmacodynamics*, 29 (5/6):445–471, 2002.

A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006.

E. Merlo-Pich and R. Gomeni. Model-based approach and signal detection theory to evaluate the performance of recruitment centers in clinical trials with antidepressant drugs. *Clinical Pharmacology and Therapeutics*, 84:378–384, September 2008.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005.

C. N. Nett and W. M. Haddad. A system-theoretic appropriate realization of the empty matrix concept. *IEEE Transactions on automatic control*, 38(5): 771–775, 1993.

M. Neve, G. De Nicolao, and L. Marchesi. Nonparametric identification of pharmacokinetic population models via Gaussian processes. In *Proceedings of 16th IFAC World Congress*, Praha, Czech Republic, 2005.

M. Neve, G. De Nicolao, and L. Marchesi. Nonparametric identification of population models via Gaussian processes. *Automatica*, 43(7):1134–1144, 2007.

M. Opper. *Online Learning in Neural Networks*, chapter A Bayesian Approach to Online Learning. Cambridge University Press, 1998.

K. Park, D. Verotta, T. F. Blaschke, and L. B. Sheiner. A semiparametric method for describing noisy population pharmacokinetic data. *Journal of pharmacokinetics and biopharmaceutics*, 25(5):615–642, 1997.

G. Pillonetto, G. De Nicolao, M. Chierici, and C. Cobelli. Fast algorithms for nonparametric population modeling of large data sets. *Automatica*, 45(1): 173–179, 2009.

G. Pillonetto, F. Dinuzzo, and G. De Nicolao. Bayesian online multitask learning of Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):193–205, 2010.

T. Poggio and F. Girosi. Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990.

Y. Qi, D. Liu, D. Dunson, and L. Carin. Multi-task compressive sensing with Dirichlet process priors. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 768–775. Omnipress, 2008.

N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, 2001.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. (Adaptive Computation and Machine Learning). MIT Press, 2001.

A. Schwaighofer, V. Tresp, and K. Yu. Learning Gaussian process kernels via hierarchical Bayes. In *Advances in Neural Information Processing Systems*, volume 17, pages 1209–1216, 2005.

L. B. Sheiner. The population approach to pharmacokinetic data analysis: rationale and standard data analysis methods. *Drug Metabolism Reviews*, 15: 153–171, 1994.

L. B. Sheiner and J. L. Steimer. Pharmacokinetic/pharmacodynamic modeling in drug development. *Annual Review of Pharmacology and Toxicology*, 40: 67–95, 2000.

L. B. Sheiner, B. Rosenberg, and V. V. Marathe. Estimation of population characteristics of pharmacokinetic parameters from routine clinical data. *Journal of Pharmacokinetics and Biopharmaceutics*, 5(5):445–479, 1977.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*. MIT Press, 2005.

V. Srivastava and T. Dwivedi. Estimation of seemingly unrelated regression equations: A brief survey. *Journal of Econometrics*, 10:15–32, 1971.

L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: a system for sharing recommendations. *Communications of the ACM*, 40(3): 59–62, 1997.

S. Thrun. Is learning the n-th thing any easier than learning the first. In *Advances in Neural Information Processing Systems*, volume 8, pages 640–646. MIT Press, 1996.

S. Thrun and L. Y. Pratt, editors. *Learning To Learn*. Kluwer Academic Publishers, Boston, MA, 1998.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D. C., 1977.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, USA, 1998.

S. Vozeh, J. L. Steimer, M. Rowland, P. Morselli, F. Mentre, L. P. Balant, and L. Aarons. The use of population pharmacokinetics in drug development. *Clinical Pharmacokinetics*, 30(2):81–93, 1996.

G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, USA, 1990.

J. C. Wakefield, A. F. M. Smith, A. Racine-Poon, and A. E. Gelfand. Bayesian analysis of linear and non-linear population models by using the Gibbs sampler. *Applied Statistics*, 41:201–221, 1994.

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 22th Annual international conference on Machine learning (ICML 2005)*, pages 1012–1019, 2005.

S. Yu, V. Tresp, and K. Yu. Robust multi-task learning with t-processes. In *Proceedings of the 24th Annual international conference on Machine learning (ICML 2007)*, pages 1103–1110, New York, NY, USA, 2007. ACM.

L. Yuh, S. Beal, M. Davidian, F. Harrison, A. Hester, K. Kowalski, E. Vonesh, and R. Wolfinger. Population pharmacokinetic/pharmacodynamic methodology and applications: a bibliography. *Biometrics*, 50:566–575, 1994.

J. Zhang, Z. Ghahramani, and Y. Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

A

Appendix

A.1 Mathematical preliminaries

In this section, we review some concepts and theorems from analysis and linear algebra, which are used in the proofs. For more details, see [Rockafellar, 1970, Hiriart-Urruty and Lemaréchal, 2004]. Let $\mathbb{E}$ denote an Euclidean space endowed with the standard inner product $\langle x_1, x_2\rangle_2 = x_1^T x_2$ and the induced norm $\|x\|_2 = \sqrt{\langle x, x\rangle_2}$.

Set-valued maps

A set-valued map (or multifunction) $A : \mathbb{E} \to 2^{\mathbb{E}}$ is a rule that associate to each point $x \in \mathbb{E}$ a subset $A(x) \subseteq \mathbb{E}$. Notice that any map $A : \mathbb{E} \to \mathbb{E}$ can be seen as a specific instance of multifunction such that $A(x)$ is a singleton for all $x \in \mathbb{E}$. The multi-function A is called *monotone* whenever

$$\langle y_1 - y_2, x_1 - x_2\rangle_2 \geq 0, \qquad \forall x_1, x_2 \in \mathbb{E}, \qquad y_1 \in A(x_1), \quad y_2 \in A(x_2),$$

If there exists $L \geq 0$ such that

$$\|y_1 - y_2\|_2 \leq L\|x_1 - x_2\|_2, \qquad \forall x_1, x_2 \in \mathbb{E}, \qquad y_1 \in A(x_1), \quad y_2 \in A(x_2),$$

then A is single-valued, and is called *Lipschitz continuous function* with modulus L. A Lipschitz continuous function is called *nonexpansive* if $L = 1$, *contractive* if $L < 1$, and *firmly non-expansive* if

$$\|y_1 - y_2\|_2^2 \leq \langle y_1 - y_2, x_1 - x_2\rangle_2, \qquad \forall x_1, x_2 \in \mathbb{E}, \qquad y_1 \in A(x_1), \quad y_2 \in A(x_2).$$

In particular, firmly non-expansive maps are single-valued, monotone, and non-expansive. For any monotone multifunction A, its *resolvent* J_α^A is defined for any $\alpha > 0$ as $J_\alpha^A := (\mathbf{I} + \alpha A)^{-1}$, where $\mathbf{I}$ stands for the identity operator. Resolvents of monotone operators are known to be firmly non-expansive.

Finite-valued convex functions

A function $f : \mathbb{E} \to \mathbb{R}$ is called *finite-valued convex* if, for any $\alpha \in [0, 1]$ and any $x_1, x_2 \in \mathbb{E}$, it satisfy

$$-\infty < f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha) f(x_2) < +\infty$$

The subdifferential of a finite-valued convex function f is a multifunction $\partial f : \mathbb{E} \to 2^{\mathbb{E}}$ defined as

$$\partial f(x) = \{\xi \in \mathbb{E} : f(y) - f(x) \geq \langle \xi, y - x \rangle_2, \quad \forall y \in \mathbb{E}\}.$$

It can be shown that the following properties hold:

1. $\partial f(x)$ is a non-empty convex compact set for any $x \in \mathbb{E}$.
2. f is (Gâteaux) differentiable at x if and only if $\partial f(x) = \{\nabla f(x)\}$ is a singleton (whose unique element is the gradient).
3. ∂f is a monotone multifunction.
4. The point x^* is a (global) minimizer of f if and only if $0 \in \partial f(x^*)$.

For any finite-valued convex function f, its Moreau-Yosida regularization (or Moreau envelope, or quadratic min-convolution) is defined as

$$f_\alpha(x) := \min_{y \in \mathbb{E}} \left(f(y) + \frac{\alpha}{2} \|y - x\|_2^2 \right).$$

For any fixed x, the minimum in the definition of f_α is attained at $y = p_\alpha(x)$, where $p_\alpha := \left(\mathbf{I} + \alpha^{-1} \partial f\right)^{-1}$ denotes the so-called *proximal mapping*. It can be shown that the following remarkable properties hold:

1. f_α is convex differentiable, and the gradient ∇f_α is Lipschitz continuous with modulus $1/\alpha$.
2. $f_\alpha(x) = f(p_\alpha(x)) + \frac{\alpha}{2} \|p_\alpha(x) - x\|_2^2$.
3. f_α and f have the same set of minimizers for all α.
4. The gradient ∇f_α is called Moreau-Yosida regularization of ∂f, and satisfy

   $$\nabla f_\alpha(x) = \alpha \left(x - p_\alpha(x)\right) = \alpha J_\alpha(x),$$

   where J_α denote the resolvent of the inverse sub-differential defined as

   $$J_\alpha := \left(\mathbf{I} + \alpha \left(\partial f\right)^{-1}\right)^{-1}.$$

Convergence theorems

Theorem 16 (Contraction mapping theorem)**.** *Let $A : \mathbb{E} \to \mathbb{E}$ and suppose that, given c^0, the sequence c^k is generated as*

$$c^{k+1} = A(c^k).$$

If A is contractive with modulus μ, then there exists a unique fixed-point c^* such that $c^* = A(c^*)$, and the sequence c^k converges to c^* at linear rate:

$$\|c^{k+1} - c^*\|_2 \leq \mu \|c^k - c^*\|_2, \qquad 0 \leq \mu < 1.$$

The following result is know as Zangwill's convergence theorem [Zangwill, 1969], see also page 206 of [Luenberger and Ye, 2008].

Theorem 17 (Zangwill's convergence theorem)**.** *Let $A : \mathbb{E} \to 2^{\mathbb{E}}$ denote a multifunction, and suppose that, given c^0, the sequence c^k is generated as*

$$c^{k+1} \in A(c^k).$$

Let $\Gamma \subset \mathbb{E}$ called solution set*. If the following conditions hold:*

1. *The graph $G_A = \{(x, y) \in \mathbb{E} \times \mathbb{E} : y \in A(x)\}$ is a closed set,*
2. *There exists a* descent function *F such that*
   - *For all $x \in \Gamma$, $F(A(x)) \leq F(x)$,*
   - *For all $x \notin \Gamma$, $F(A(x)) < F(x)$,*
3. *The sequence c^k is bounded,*

then all the cluster points of c^k belongs to the solution set.

Some matrix manipulation lemmas

In this subsection, we recall some important matrix manipulation lemmas, all related to the concept of Shur complement, see [Zhang, 2005] for further details. Let $\mathbb{S}_+^m$ denotes the cone of symmetric positive semi-definite matrices of order m, and $\mathbf{A}^\dagger$ denote the Moore-Penrose pseudo-inverse of a square matrix $\mathbf{A}$.

Lemma 9. *Let $\mathbf{A}$ and $\mathbf{B}$ denote two nonsingular matrices. If $\mathbf{U}$ and $\mathbf{V}$ are such that $(\mathbf{A} + \mathbf{UBV})$ is nonsingular, then matrix*

$$\mathbf{E} = \mathbf{B}^{-1} + \mathbf{VA}^{-1}\mathbf{U}$$

is nonsingular, and

$$(\mathbf{A} + \mathbf{UBV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{UE}^{-1}\mathbf{VA}^{-1}.$$

Lemma 10 (Schur Complement)**.** *Let*

$$\mathbf{X} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix},$$

where $\mathbf{A}$ and $\mathbf{C}$ are symmetric matrices. Then, $\mathbf{X}$ is non-singular if and only if

1. *$\mathbf{A}$ is non-singular.*
2. *The Schur complement $\mathbf{D} = \mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ is non-singular.*

If $\mathbf{X}$ is non-singular, its inverse is given by

$$\mathbf{X}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{BD}^{-1}\mathbf{B}^T\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{BD}^{-1} \\ -\mathbf{D}^{-1}\mathbf{B}^T\mathbf{A}^{-1} & \mathbf{D}^{-1} \end{pmatrix}.$$

Finally, we recall the following generalization, due to [Albert, 1969].

Lemma 11 (Generalized Schur Complement)**.** *Let*

$$\mathbf{X} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix},$$

where $\mathbf{A}$ *and* $\mathbf{C}$ *are symmetric matrices. Then,* $\mathbf{X} \in \mathbb{S}_+^{n+m}$ *if and only if*

1. $\mathbf{A} \in \mathbb{S}_+^n$.
2. $\mathbf{C} - \mathbf{B}^T\mathbf{A}^\dagger\mathbf{B} \in \mathbb{S}_+^m$.
3. $\mathbf{A}\mathbf{A}^\dagger\mathbf{B} = \mathbf{B}$.

A.2 Proofs for fixed-point and coordinate descent algorithms

The following Lemma will prove useful in the subsequent proofs.

Lemma 12. *The functional* F *of problem (2.2) is such that* $F(c+u) = F(c)$*, for any vector* u *in the nullspace of the kernel matrix.*

Proof. Let u denote any vector in the nullspace of the kernel matrix. Then, we have

$$F(c+u) = f\left(\mathbf{K}(c+u)\right) + \frac{(c+u)^T\mathbf{K}(c+u)}{2} = f\left(\mathbf{K}c\right) + \frac{c^T\mathbf{K}c}{2} = F(c).$$

□

Proof of Theorem 8. Problem (2.2) is a convex optimization problem, where the functional F is continuous and bounded below. First of all, we show that there exist optimal solutions. Observe that minimization can be restricted to the range of the kernel matrix. Indeed, any vector $c \in \mathbb{E}$ can be uniquely decomposed as $c = u + v$, where u belongs to the nullspace of $\mathbf{K}$ and v belongs to the range. By Lemma 12, we have $F(c) = F(v)$. Since F is coercive on the range of the kernel matrix ($\lim_{\|v\|_2 \to +\infty} F(v) = +\infty$), it follows that there exist optimal solutions.

A necessary and sufficient condition for c^* to be optimal is

$$0 \in \partial F(c^*) = \mathbf{K}\left(\partial f\left(\mathbf{K}c^*\right) + c^*\right) = \mathbf{K}G(c^*), \qquad G(c^*) := \partial f\left(\mathbf{K}c^*\right) + c.$$

Consider the decomposition $G(c^*) = u_G + v_G$, where u_G belongs to the nullspace of the kernel matrix and v_G belongs to the range. Observe that

$$v_G = G(c^*) - u_G = G(c^* - u_G).$$

We have

$$0 \in \mathbf{K}G(c^*) = \mathbf{K}v_G \quad \Rightarrow \quad 0 \in G(c^* - u_G) = v_G,$$

so that, for any optimal c^*, there exists an optimal $c = c^* - u_G$ such that

$$0 \in \partial f\left(\mathbf{K}c\right) + c. \tag{A.1}$$

By introducing the inverse sub-differential, equation (A.1) can be rewritten as

$$\mathbf{K}c \in \left(\partial f\right)^{-1}(-c).$$

Multiplying by $\alpha > 0$ both sides and subtracting c, we obtain

$$\alpha\mathbf{K}c - c \in \alpha\left(\partial f\right)^{-1}(-c) - c.$$

Finally, introducing the resolvent J_α as in (2.6), we have

$$\alpha\mathbf{K}c - c \in \left(J_\alpha\right)^{-1}(-c)$$

Since J_α is single-valued, equation (2.7) follows. □

Proof of Corollary 2. Let's start from the sufficient condition for optimality (A.1). If (2.3) holds, then the subdifferential of f decouples with respect to the different components, so that there exist optimal coefficients c_i such that

$$0 \in \partial f_i\left(k_i^T c\right) + c_i, \qquad i = 1, \ldots, \ell.$$

Equivalently,

$$k_i^T c \in \left(\partial f_i\right)^{-1}(-c_i).$$

Multiplying by $\alpha_i > 0$ both sides and subtracting c_i, we have

$$\alpha_i k_i^T c - c_i \in \alpha_i\left(\partial f_i\right)^{-1}(-c_i) - c_i.$$

The thesis follows by introducing the resolvents $J^i_{\alpha_i}$ and solving for $-c_i$. □

Proof of Theorem 9. We show that a subsequence of c^k generated by algorithm (2.10) converges to an optimal solution of Problem (2.2). By Theorem 8, there exists optimal solutions c^* satisfying (2.7). We now observe that any other vector c such that $\mathbf{K}(c^* - c) = 0$ is also optimal. Indeed, we have $c = c^* + u$, where u belongs to the nullspace of the kernel matrix. By Lemma 12, it follows that $F(c) = F(c^*)$. To prove (2.9), it suffices to show that $\mathbf{K}r^k \to 0$, where $r^k := c^k - c^*$ can be uniquely decomposed as

$$r^k = u^k + v^k, \qquad \mathbf{K}u^k = 0, \qquad \langle u^k, v^k\rangle_2 = 0.$$

We need to prove that $\|v^k\|_2 \to 0$. Since J_α is nonexpansive, we have

$$\begin{aligned}
\gamma^{k+1} := \|r^{k+1}\|_2^2 &= \|c^{k+1} - c^*\|_2^2 \\
&= \|J_\alpha(\alpha\mathbf{K}c^k - c^k) - J_\alpha(\alpha\mathbf{K}c^* - c^*)\|_2^2 \\
&\leq \|\alpha\mathbf{K}r^k - r^k\|_2^2 \\
&= \|\alpha\mathbf{K}v^k - r^k\|_2^2.
\end{aligned}$$

Observing that v^k is orthogonal to the nullspace of the kernel matrix, we can further estimate as follows

$$\|\alpha\mathbf{K}v^k - r^k\|_2^2 = \gamma^k - v^{kT}\left(2\alpha\mathbf{K} - \alpha^2\mathbf{K}^2\right)v^j \leq \gamma^k - \beta\|v^k\|_2^2,$$

where

$$\beta := \min_{i:\alpha_i>0} \alpha\alpha_i(2 - \alpha\alpha_i).$$

and α_i denote the eigenvalues of the kernel matrix. Since the kernel matrix is positive semidefinite and condition (2.11) holds, we have

$$0 \leq \alpha\alpha_i < 2.$$

Since the kernel matrix is not null and have a finite number of eigenvalues, there's at least one eigenvalue with strictly positive distance from zero. It follows that $\beta > 0$. Since

$$0 \leq \gamma^{k+1} \leq \gamma^0 - \beta\sum_{j=1}^{k}\|v^j\|_2^2,$$

we have, necessarily, that $\|v^k\|_2 \to 0$. Finally, observe that c^k remains bounded

$$\|c^k\|_2 \leq \|r^k\|_2 + \|c^*\|_2 \leq \|r^0\|_2 + \|c^*\|_2,$$

so that there's a subsequence converging to an optimal solution. In fact, by (2.9) it follows that any cluster point of c^k is an optimal solution. □

Proof of Theorem 10. Algorithm (2.10) can be rewritten as

$$c^{k+1} = A(c^k),$$

where the map $A : \mathbb{E} \to \mathbb{E}$ is defined as

$$A(c) := -J_\alpha\left(\alpha\mathbf{K}c - c\right).$$

Under both conditions (1) and (2) of the theorem, we show that A is contractive. Uniqueness of the fixed-point, and convergence with linear rate will then follow from the contraction mapping theorem (Theorem 16). Let

$$\mu_1 := \|\alpha\mathbf{K} - \mathbf{I}\|_2 = \max_i |1 - \alpha_i\alpha|,$$

where α_i denote the eigenvalues of the kernel matrix. Since the kernel matrix is positive semidefinite, and condition (2.11) holds, we have

$$0 \leq \alpha_i\alpha < 2,$$

so that $\mu_1 \leq 1$. We now show that the following inequality holds:

$$\|J_\alpha(y_1) - J_\alpha(y_2)\|_2 \leq \mu_2\|y_1 - y_2\|_2, \tag{A.2}$$

where

$$\mu_2 := \left(1 + \frac{1}{L^2}\right)^{-1/2},$$

and L denotes the Lipschitz modulus of ∇f when f is differentiable with Lipschitz continuous gradient, and $L = +\infty$ otherwise. Since J_α is nonexpansive, it is easy to see that (A.2) holds when $L = +\infty$. Suppose now that f is differentiable and ∇f is Lipschitz continuous with modulus L. It follows that the inverse gradient satisfies

$$\|(\nabla f)^{-1}(x_1) - (\nabla f)^{-1}(x_2)\|_2 \geq \frac{1}{L}\|x_1 - x_2\|_2.$$

Since $(\nabla f)^{-1}$ is monotone, we have

$$\begin{aligned}\|J_\alpha^{-1}(x_1) - J_\alpha^{-1}(x_2)\|_2^2 &= \|x_1 - x_2 + (\nabla f)^{-1}(x_1) - (\nabla f)^{-1}(x_2)\|_2^2 \\ &\geq \|x_1 - x_2\|_2^2 + \|(\nabla f)^{-1}(x_1) - (\nabla f)^{-1}(x_2)\|_2^2 \\ &\geq \left(1 + \frac{1}{L^2}\right)\|x_1 - x_2\|_2^2.\end{aligned}$$

From this last inequality, we obtain (A.2). Finally, we have

$$\begin{aligned}\|A(c_1) - A(c_2)\|_2 &= \|J_\alpha\left(\alpha\mathbf{K}c_1 - c_1\right) - J_\alpha\left(\alpha\mathbf{K}c_2 - c_2\right)\|_2 \\ &\leq \mu_2\|(\alpha\mathbf{K} - \mathbf{I})(c_1 - c_2)\|_2 \\ &\leq \mu\|c_1 - c_2\|_2,\end{aligned}$$

where we have set $\mu := \mu_1\mu_2$. Consider the case in which $\mathbf{K}$ is strictly positive definite. Then, it holds that

$$0 < \alpha_i\alpha < 2,$$

so that $\mu_1 < 1$, and A is contractive. Finally, when f is differentiable and ∇f is Lipschitz continuous, we have $\mu_2 < 1$ and, again, it follows that A is contractive. By the contraction mapping theorem (Theorem 16), there exists a unique c^* satisfying (2.7), and the sequence c^k of Picard iterations converges to c^* at a linear rate. □

Proof of Theorem 11. We shall apply Theorem 17 to the coordinate descent macro-iterations, where the solution set Γ is given by

$$\Gamma := \{c \in \mathbb{E} : (2.8) \quad \text{holds}\}.$$

Let A denote the algorithmic map obtained after each macro-iteration of the coordinate descent algorithm. By the essentially cyclic rule, we have

$$c \in A(c) = \bigcup_{(i_1,\ldots,i_s)\in I} \{(A_{i_1} \circ \cdots \circ A_{i_s})(c)\},$$

where I is the set of strings of length at most $s = T$ on the alphabet $\{1, \ldots, \ell\}$ such that all the characters are picked at least once. Observing that the set

I has finite cardinality, it follows that the graph G_A is the union of a finite number of graphs of point-to-point maps:

$$G_A = \bigcup_{(i_1,\ldots,i_s)\in I} \{(x,y) \in \mathbb{E}\times\mathbb{E} : y = (A_{i_1} \circ \cdots \circ A_{i_s})(x)\}.$$

Now notice that each map A_i is of the form

$$A_i(c) = c + e_i t_i(c), \qquad t_i(c) := S_i\left(\sum_{j\neq i} \frac{k_{ij}}{k_{ii}} c_j\right) - c_i.$$

All the resolvents are Lipschitz continuous, so that functions A_i are also Lipschitz continuous. It follows that the composition of a finite number of such maps is continuous, and its graph is a closed set. Since the union of a finite number of closed sets is also closed, we obtain that G_A is closed.

Each map A_i yields the solution of an exact line search over the i-th coordinate direction for minimizing functional F of Problem (2.2). Hence, the function

$$\phi_i(t) = F(c + e_i t),$$

is minimized at $t_i(c)$, that is

$$0 \in \partial\phi_i(t_i(c)) = \langle e_i, \partial F(c + e_i t_i(c))\rangle_2 = \langle k_i, \partial f(\mathbf{K}A_i(c)) + A_i(c)\rangle_2.$$

Equivalently,

$$-\langle k_i, A_i(c)\rangle_2 \in \langle k_i, \partial f(\mathbf{K}A_i(c))\rangle_2. \tag{A.3}$$

By definition of subdifferential, we have

$$f(\mathbf{K}A_i(c)) - f(\mathbf{K}c) \leq t_i(c)\gamma, \qquad \forall\gamma \in \langle k_i, \partial f(\mathbf{K}A_i(c))\rangle_2.$$

In particular, in view of (A.3), we have

$$f(\mathbf{K}A_i(c)) - f(\mathbf{K}c) \leq -t_i(c)\langle k_i, A_i(c)\rangle_2.$$

Now, observe that

$$\begin{aligned}
F(A(c)) &\leq F(A_i(c)) = F(c + e_i t_i(c)) \\
&= F(c) + t_i^2(c)\frac{k_{ii}}{2} + t_i(c)\langle k_i, c\rangle_2 + f(\mathbf{K}A_i(c)) - f(\mathbf{K}c) \\
&\leq F(c) + t_i^2(c)\frac{k_{ii}}{2} + t_i(c)\langle k_i, c\rangle_2 - t_i(c)\langle k_i, A_i(c)\rangle_2 \\
&= F(c) + t_i^2(c)\frac{k_{ii}}{2} + t_i(c)\langle k_i, c - A_i(c)\rangle_2 \\
&= F(c) + t_i^2(c)\frac{k_{ii}}{2} - t_i^2(c)k_{ii} \\
&= F(c) - t_i^2(c)\frac{k_{ii}}{2}.
\end{aligned}$$

Since $k_{ii} > 0$, the following inequalities hold:

$$t_i^2(c) \leq \frac{2}{k_{ii}}\left(F(c) - F(A_i(c))\right) \leq \frac{2}{k_{ii}}\left(F(c) - F(A(c))\right). \tag{A.4}$$

We now show that F is a *descent function* for the map A associated with the solution set Γ. Indeed, if c satisfy (2.8), then the application of the map A doesn't change the position, so that

$$F(A(c)) = F(c).$$

On the other hand, if c does not satisfy (2.8), there's at least one index i such that $t_i(c) \neq 0$. Since all the components are chosen at least once, and in view of (A.4), we have

$$F(A(c)) < F(c).$$

Finally, we need to prove that the sequence of macro-iterations remains bounded. In fact, it turns out that the whole sequence c^k of iterations of the coordinate descent algorithm is bounded. From the first inequality in (A.4), the sequence $F(c^k)$ is non-increasing and bounded below, and thus it must converge to a number

$$F_\infty = \lim_{k \to +\infty} F(c^k) \leq F(c^0). \tag{A.5}$$

Again from (A.4), we obtain that the sequence of step sizes is square summable:

$$\sum_{k=0}^{+\infty} \left\| c^{k+1} - c^k \right\|_2^2 \leq \frac{2}{\min_j k_{jj}} \left(F(c^0) - F_\infty \right) < +\infty.$$

In particular, step-sizes are also uniformly bounded:

$$t_i^2(c^k) = \left\| c^{k+1} - c^k \right\|_2^2 \leq \frac{2}{\min_j k_{jj}} \left(F(c^0) - F_\infty \right) < +\infty. \tag{A.6}$$

Now, fix any coordinate i, and consider the sequence c_i^k. Let h_{ij} denote the subsequence of indices in which the i-th component is picked by the essentially cyclic rule and observe that

$$c_i^{h_{ij}} = S_i \left(\frac{k_i^T c^{h_{ij}-1}}{k_{ii}} - c_i^{h_{ij}-1} \right).$$

Recalling the definition of S_i, and after some algebra, the last equation can be rewritten as

$$c_i^{h_{ij}} \in -\partial f_i \left(k_i^T c^{h_{ij}-1} + k_{ii} t_i \left(c^{h_{ij}-1} \right) \right).$$

Since $\partial f_i(x)$ is a compact set for any $x \in \mathbb{R}$, it suffices to show that the argument of the subdifferential is bounded. For any k, let's decompose c^k as

$$c^k = u^k + v^k, \qquad \mathbf{K} u^k = 0, \qquad \langle u^k, v^k \rangle_2 = 0.$$

Letting $\alpha_1 > 0$ denote the smallest non-null eigenvalue of the kernel matrix, we have

$$\alpha_1 \|v^k\|_2^2 \leq v^{kT} \mathbf{K} v^k = c^{kT} \mathbf{K} c^k \leq 2F(c^k) \leq 2F(c^0).$$

By the triangular inequality, we have

$$\left|k_i^T c^k + k_{ii} t_i\left(c^k\right)\right| \leq M \left|\frac{k_i^T c^k}{k_{ii}} + t_i\left(c^k\right)\right| \leq M \left(\left|\frac{k_i^T c^k}{k_{ii}}\right| + \left|t_i\left(c^k\right)\right|\right),$$

where $M := \max_j |k_{jj}|$. The first term can be majorized as follows:

$$\left|\frac{k_i^T c^k}{k_{ii}}\right| = \left|\frac{k_i^T v^k}{k_{ii}}\right| \leq \left\|\frac{k_i}{k_{ii}}\right\|_2 \|v^k\|_2 \leq \left\|\frac{k_i}{k_{ii}}\right\|_2 \sqrt{\frac{2F(c^0)}{\alpha_1}} \leq \sqrt{\frac{2\ell F(c^0)}{\alpha_1}} < +\infty,$$

while the term $\left|t_i\left(c^k\right)\right|$ is bounded in view of (A.6). It follows that c_i^k is bounded independently of i, which implies that c^k is bounded. In particular, the subsequence consisting of the macro-iterations is bounded as well.

By Theorem 17, there's at least one subsequence of the sequence of macro-iterations converging to a limit c_∞ that satisfies (2.8), and thus minimizes F. By continuity of F, we have

$$F(c_\infty) = \min_{c \in \mathbb{R}^\ell} F(c).$$

Finally, in view of (A.5), we have $F_\infty = F(c_\infty)$, which proves (2.9) and shows that any cluster point of c^k is an optimal solution of Problem (2.2). □

Proof of Theorem 12. Equation (2.7) can be rewritten as

$$J_\alpha\left(\mathbf{K}_\alpha c\right) + c = 0.$$

Now, let f_α denote the Moreau-Yosida regularization of f. From the properties of f_α, we have

$$\nabla f_\alpha(\mathbf{K}_\alpha c) + \alpha c = 0.$$

Multiplying both sides of the previous equation by $\alpha^{-1}\mathbf{K}_\alpha$, we obtain

$$\alpha^{-1}\mathbf{K}_\alpha \nabla f_\alpha(\mathbf{K}_\alpha c) + \mathbf{K}_\alpha c = 0.$$

Finally, the last equation can be rewritten as

$$\nabla_c \left[\alpha^{-1} f_\alpha\left(\mathbf{K}_\alpha c\right) + \frac{c^T \mathbf{K}_\alpha c}{2}\right] = 0,$$

so that the thesis follows. □

A.3 Proofs for kernel machines with two layers

Proof of Theorem 13. By fixing any optimal g_1, and letting

$$z_i := g_1(x_i), \qquad i = 1, \ldots, \ell,$$

problem (3.2) can be rewritten as a function of only g_2:

$$\min_{g_2 \in \mathcal{H}_2} \left[f\left(g_2(z_1), \ldots, g_2(z_\ell)\right) + \Omega_2(\|g_2\|_{\mathcal{H}_2})\right].$$

By standard representer theorems for vector-valued functions (see [Micchelli and Pontil, 2005a] and the remark on monotonicity in [Schölkopf et al., 2001] after Theorem 1), there exists an optimal g_2 in the form

$$g_2(z) = \sum_{i=1}^{\ell} K^2_{z_i}(z) c_i^2.$$

Then, by fixing an optimal g_2 in this form, problem (3.2) can be written as a function of only g_1 as

$$\min_{g_1 \in \mathcal{H}_1} \left(\widetilde{f}\left(g_1(x_1), \ldots, g_1(x_\ell)\right) + \Omega_1(\|g_1\|_{\mathcal{H}_1}) \right),$$

where

$$\widetilde{f}(z) := f\left(g_2(z_1), \ldots, g_2(z_\ell)\right).$$

Notice that the new function $\widetilde{f}$ depends on g_2. Again, by the single-layer representer theorem the finite kernel expansion for g_1 follows. Finally, it is immediate to see that the overall input-output relation $g_2 \circ g_1$ can be written as in (3.3). □

Proof of Theorem 14. Problem (3.4) is a specific instance of problem (3.2). The functional to minimize is bounded below, lower semi-continuous and radially-unbounded with respect to (g_1, g_2). Existence of minimizers follows by weak-compactness of the unit ball in $\mathcal{H}_1$ and $\mathcal{H}_2$. By Theorem 13, there exists an optimal g_1 in the form

$$g_1(x) = \sum_{j=1}^{\ell} K^1_{x_j}(x) c_j^1 = \sum_{j=1}^{\ell} \operatorname{diag}\left\{ \widetilde{K}_1(x, x_j), \ldots, \widetilde{K}_m(x, x_j) \right\} c_j^1.$$

Introduce the matrix $\mathbf{C} \in \mathbb{R}^{m \times \ell}$ whose rows are denoted by $(c^i)^T$ and whose columns are c_j^1. Then, the i-th component of g_1 can be written as:

$$g_1^i(x) = \sum_{j=1}^{\ell} c_j^i \widetilde{K}_i(x, x_j).$$

By Theorem 13, there exists an optimal g_2 such that

$$g_2(z) = \sum_{j=1}^{\ell} c_j^2 K^2_{g_1(x_j)}(z) = \sum_{j=1}^{\ell} c_j^2 z^T \mathbf{S} g_1(x_j) = z^T \mathbf{S} \sum_{j=1}^{\ell} c_j^2 g_1(x_j) = z^T \mathbf{S} a,$$

where

$$a := \sum_{j=1}^{\ell} c_j^2 g_1(x_j).$$

Letting matrices $\mathbf{K}^k \in \mathbb{S}_+^\ell$ as in (3.6), problem (3.4) can be rewritten as

$$\min_{c^1, \ldots c^m \in \mathbb{R}^\ell, a \in \mathbb{R}^m} \left[f\left(\sum_{k=1}^{m} a_k \mathbf{K}^k c^k \right) + \sum_{k=1}^{m} \frac{c^{kT} \mathbf{K}^k c^k}{2 s_k} \right], \quad \text{s.t.} \quad a^T \mathbf{S} a \leq 1.$$

Vectors c^i are optimal if and only if

$$0 \in s_i a_i \mathbf{K}^i \partial f \left(\sum_{k=1}^{m} a_k \mathbf{K}^k c^k \right) + \mathbf{K}^i c^i,$$

where ∂ is the sub-differential of a convex function [Rockafellar, 1970]. Now, letting

$$c \in -\partial f \left(\sum_{k=1}^{m} a_k \mathbf{K}^k c^k \right),$$

we obtain the following sufficient condition for optimality:

$$c^i = s_i a_i c.$$

Letting $d_i := s_i a_i^2$ and $\mathbf{K} := \sum_{i=1}^{m} d_i \mathbf{K}^i$, problem (3.4) boils down to (3.5)-(3.6). By Theorem 13 again, the overall input-output relation can be written as in equation (3.3), where the kernel K satisfy

$$\begin{aligned} K(x_1, x_2) &= K^2(g_1(x_1), g_1(x_2)) = g_1(x_1)^T \mathbf{S} g_1(x_2) = \sum_{i=1}^{m} s_i g_1^i(x_1) g_1^i(x_2) \\ &= \sum_{i=1}^{m} s_i a_i^2 \sum_{j_1=1}^{\ell} \sum_{j_2=1}^{\ell} c_{j_1} c_{j_2} \widetilde{K}_i(x_{j_1}, x_1) \widetilde{K}_i(x_{j_2}, x_2) \\ &= \sum_{i=1}^{m} d_i K_i(x_1, x_2), \end{aligned}$$

and K_i are as in (3.7). □

Proof of Lemma 2. Assume that (c^*, d^*) is an optimal pair for problem (3.4). Without loss of generality, we can assume $d^* \neq 0$. Indeed, if there's an optimal solution with $d^* = 0$, then $c = 0, d \neq 0$ is optimal as well. Now, let $\gamma := \sum_{i=1}^{m} d_i^*$, and notice that $0 < \gamma \leq 1$. Introducing the new pair $(c, d) = (\gamma c^*, d^*/\gamma)$, the value of the objective functional in correspondence with (c, d) is

$$\begin{aligned} f(\mathbf{K}(d)c) + \frac{c^T \mathbf{K}(d) c}{2} &= f(\mathbf{K}(d^*)c^*) + \frac{\gamma}{2} (c^*)^T \mathbf{K}(d^*) c^* \\ &\leq f(\mathbf{K}(d^*)c^*) + \frac{c^{*T} \mathbf{K}(d^*) c^*}{2}, \end{aligned}$$

so that the new pair is optimal as well as satisfy the equality constraint (3.8). □

Proof of Lemma 3. By introducing $z = \mathbf{K}c$, we have $c = \mathbf{K}^\dagger z + u$, where $\mathbf{K}u = 0$, and problem 3.4 can be rewritten as

$$\min_{z \in \mathbb{R}^\ell, \mathbf{K} \in \mathbb{S}_+^m} \left(f(z) + \frac{z^T \mathbf{K}^\dagger z}{2} \right), \quad \text{subject to (3.6)}, \quad z \in \text{range}(\mathbf{K}).$$

Now, the range constraint can be incorporated into the objective functional by introducing function h as in the statement of the Lemma. We obtain the following problem:

$$\min_{z\in\mathbb{R}^\ell,\mathbf{K}\in\mathbb{S}_+^m} \left(f(z)+h(z,\mathbf{K})\right), \quad \text{subject to (3.6).} \tag{A.7}$$

The functional of problem (A.7) can be seen to be jointly convex in $(z,\mathbf{K})$. It suffices to prove that the functional is convex when restricted to the set of pairs $(z,\mathbf{K})$ such that $z\in\text{range}(\mathbf{K})$, where we have

$$\mathbf{K}\mathbf{K}^\dagger z = z.$$

By the Generalized Schur Complement (Lemma 11), letting $\mathbf{A}=\mathbf{K}$, $\mathbf{B}=z$, $\mathbf{C}=\alpha$, we have

$$z^T\mathbf{K}^\dagger z\le\alpha \quad\Leftrightarrow\quad \begin{pmatrix}\mathbf{K} & z\\ z^T & \alpha\end{pmatrix}\in\mathbb{S}_+^{m+1}.$$

It follows that the epigraph of $h(z,\mathbf{K})$ is a convex set. Since f is a convex function, the overall functional in (A.7) is convex. Since constraints (3.6) are linear, we have a convex optimization problem. Finally, since $\mathbf{K}(d)$ is a linear function of d, problem (3.9) is also convex. □

Proof of Lemma 4. By Lemma 2, minimization with respect to d can be restricted to the standard simplex Δ_m. For any fixed d, the functional of problem (4.1) is a convex quadratic function of c. If $c^*(d)$ satisfy equation (4.2), then the partial derivative of the objective functional with respect to c is zero at c^*, meaning that c^* is optimal. Dropping the dependence on d, equation (4.2) can be rewritten as

$$y-\mathbf{K}c^*=\lambda c^*.$$

In correspondence with such optimal c^*, we have

$$\frac{\|y-\mathbf{K}c^*\|_2^2}{2\lambda}+\frac{c^{*T}\mathbf{K}c^*}{2}=\frac{\lambda}{2}\|c^*\|_2^2+\frac{c^{*T}\left(y-\lambda c^*\right)}{2}=\frac{y^Tc^*}{2}.$$

□

Proof of Lemma 5. By Lemma 2, minimization with respect to d can be restricted to the standard simplex Δ_m. In addition, we have

$$\begin{aligned}\frac{\|y-\mathbf{K}c\|_2^2}{2\lambda}+\frac{c^T\mathbf{K}c}{2}&=\frac{1}{2\lambda}\left\|u-\mathbf{K}c+\frac{\lambda c}{2}\right\|_2^2+\frac{c^T\mathbf{K}c}{2}\\&=\frac{\|u-\mathbf{K}c\|_2^2}{2\lambda}+\frac{c^T\left(\lambda c/2+u\right)}{2}\\&=\frac{\|u-\mathbf{K}c\|_2^2}{2\lambda}+\frac{c^Ty}{2},\end{aligned}$$

where $c^T y/2$ does not depend on $\mathbf{K}$ (and thus does not depend on d). Now, recalling that

$$\mathbf{K}(d) = \sum_{i=1}^{m} d_i \mathbf{K}^i,$$

we have

$$\mathbf{K}c = \sum_{i=1}^{m} d_i \mathbf{K}^i c = \sum_{i=1}^{m} d_i v_i = \mathbf{V}d.$$

□

Proof of Lemma 6. From equation (4.2), we have

$$c_\infty = \lim_{\lambda \to +\infty} \left(\mathbf{K}(d) + \lambda \mathbf{I}\right)^{-1} y = 0.$$

Since $\mathbf{K}(d)$ is a continuous function of d defined over the compact set Δ_m, by fixing any matrix norm $\|\cdot\|$ there exists a sufficiently large $\underline{\lambda}$ such that

$$\max_{d \in \Delta_m} \|\mathbf{K}(d)\| < \underline{\lambda}.$$

For $\lambda > \underline{\lambda}$, the expansion

$$\left(\mathbf{K}(d)/\lambda + \mathbf{I}\right)^{-1} = \mathbf{I} - \mathbf{K}(d)/\lambda + o(1/\lambda^2),$$

holds. By Lemma 4, it follows that

$$\begin{aligned} d^*(\lambda) &= \arg\min_{d \in \Delta_m} y^T \left(\mathbf{K}(d)/\lambda + \mathbf{I}\right)^{-1} y \\ &= \arg\min_{d \in \Delta_m} \left[\lambda \|y\|_2^2/2 - (y^T \mathbf{K}(d) y) + o(1/\lambda)\right] \\ &= \arg\min_{d \in \Delta_m} \left[\|y\|_2^2/2 - (y^T \mathbf{K}(d) y)/\lambda + o(1/\lambda^2)\right] \\ &= \arg\max_{d \in \Delta_m} \left[y^T \mathbf{K}(d) y - o(1/\lambda)\right]. \end{aligned}$$

Hence, $d_\infty = \lim_{\lambda \to +\infty} d^*(\lambda)$ solves the following linear program

$$\max_{d \in \Delta_m} \sum_{i=1}^{m} d_i (y^T \mathbf{K}^i y).$$

Then, it is easy to see that $d = e_k, k \in \arg\max_{i=1,\ldots,m} y^T \mathbf{K}^i y$ is an optimal solution of the linear program, where k is any index maximizing the "kernel alignment" $y^T \mathbf{K}^i y$. □

Proof of Lemma 7. When basis kernel are chosen as in (4.5), $g(x)$ can be written as in (4.6)-(4.7), and we have

$$\mathbf{K}(d) = \mathbf{X}\mathbf{\Gamma}\mathbf{X}^T.$$

Letting $z := \mathbf{X}^T c$, and $w := \mathbf{\Gamma} z$, it follows that

$$\mathbf{K}c = \mathbf{X}\mathbf{\Gamma}z = \mathbf{X}w,$$

and

$$c^T\mathbf{K}c = c^T\mathbf{X}\mathbf{\Gamma}\mathbf{X}^T c = z^T\mathbf{\Gamma}z = \sum_{i=1}^{m} s_i d_i z_i^2 = \sum_{i:d_i\neq 0}^{m} s_i d_i z_i^2 = \sum_{i:d_i\neq 0}^{m} \frac{w_i^2}{s_i d_i}.$$

Hence, problem (4.1) can be rewritten as

$$\min_{\substack{w\in\mathbb{R}^n \\ d\in\Delta_m}} \left(\frac{\|y - \mathbf{X}w\|_2^2}{2} + \frac{\lambda}{2} \sum_{i:d_i\neq 0}^{m} \frac{w_i^2}{s_i d_i} \right), \qquad \text{s.t.} \qquad w \in \text{range}\left(\mathbf{\Gamma}(d)\right).$$

It turns out that optimal coefficients d_i can be obtained in closed form by adapting an argument of [Micchelli and Pontil, 2005b] (Lemma 26). Indeed, by the Cauchy-Schwartz inequality and the fact that $d \in \Delta_m$, we have

$$\sum_{i:d_i\neq 0}^{m} \frac{w_i^2}{s_i d_i} = \sum_{i:d_i\neq 0}^{m} \left(\frac{w_i}{\sqrt{s_i d_i}} \right)^2 \sum_{i:d_i\neq 0}^{m} (\sqrt{d_i})^2 \geq \left(\sum_{i:d_i\neq 0}^{m} \frac{|w_i|}{\sqrt{s_i}} \right)^2,$$

Now, since $w \in \text{range}\left(\mathbf{\Gamma}(d)\right)$, it holds that

$$\sum_{i:d_i\neq 0}^{m} \frac{|w_i|}{\sqrt{s_i}} = \sum_{i:w_i\neq 0}^{m} \frac{|w_i|}{\sqrt{s_i}} = \sum_{i=1}^{m} \frac{|w_i|}{\sqrt{s_i}}.$$

Now, letting

$$d_i = \frac{|w_i|}{\sqrt{s_i}} \left(\sum_{j=1}^{m} \frac{|w_j|}{\sqrt{s_j}} \right)^{-1}, \qquad i = 1, \ldots, m,$$

the equality

$$\sum_{i:d_i\neq 0}^{m} \frac{w_i^2}{s_i d_i} = \sum_{i=1}^{m} \frac{|w_i|}{\sqrt{s_i}}$$

is achieved, and all the constraints on both d_i and w_i are satisfied. Hence, coefficients d_i can be eliminated from the optimization problem. Since the objective functional doesn't depend on d anymore, the range constraint can be also removed, and problem (4.9) is obtained. □

Proof of Lemma 8. When basis kernel are chosen as in (4.5), $g(x)$ can be written as in (4.6)-(4.7), and we have

$$\mathbf{K}(d) = \mathbf{X}\mathbf{\Gamma}\mathbf{X}^T.$$

Letting $z := \mathbf{X}^T c$, it follows that

$$\mathbf{K}c = \mathbf{X}\mathbf{\Gamma}z = \widetilde{\mathbf{X}}\widetilde{w}$$

$$c^T\mathbf{K}c = c^T\mathbf{X}\mathbf{\Gamma}\mathbf{X}^T c = \|\mathbf{\Gamma}^{1/2}z\|_2^2 = \|\widetilde{\mathbf{\Gamma}}^{1/2}\widetilde{w}\|_2^2$$

Hence, problem (4.1) reduces to problem (4.10). □

A.4 Proofs for client-server multi-task learning

Proof of Theorem 15. Let

$$\mathbf{R} := \left((1-\alpha) \sum_{j=1}^{m} \mathbf{I}(:, k^j) \widetilde{\mathbf{K}}^j(k^j, k^j) \mathbf{I}(k^j, :) + \lambda \mathbf{W} \right)^{-1},$$

and observe that $\mathbf{R}$ is a sparse matrix such that

$$\mathbf{R}(k^i, k^j) = \begin{cases} \mathbf{R}^j, & i = j \\ \mathbf{0}, & i \neq j \end{cases},$$

where matrices $\mathbf{R}^j$ are defined as in lines 1-3 of Algorithm 4. Introduce the following matrices and vectors,

$$\mathbf{F} := \alpha \mathbf{L}^T \mathbf{P}^T \mathbf{R} \mathbf{P} \mathbf{L}, \quad \breve{y} := \mathbf{L}^T \mathbf{P}^T \mathbf{R} y, \quad \mathbf{H} := \left(\mathbf{D}^{-1} + \mathbf{F} \right)^{-1}, \quad z := \mathbf{H} \breve{y}.$$

By the definition of $\mathbf{H}$, we have

$$\mathbf{DFH} = \mathbf{HFD} = \mathbf{D} - \mathbf{H}.$$

By the definition of $\mathbf{P}$ in equation (5.6), $\mathbf{F}$ and $\breve{y}$ can be rewritten as

$$\mathbf{F} := \sum_{j=1}^{m} \mathbf{L}^T(:, h^j) \mathbf{R}^j \mathbf{L}(h^j, :),$$
$$\breve{y} := \sum_{j=1}^{m} \mathbf{L}^T(:, h^j) \mathbf{R}^j y^j.$$

From (5.5) and (5.7), we have

$$\mathbf{K} + \lambda \mathbf{W} = \alpha \overline{\mathbf{K}} + \mathbf{R}^{-1} = \alpha \mathbf{PLDL}^T \mathbf{P}^T + \mathbf{R}^{-1}.$$

By applying Lemma 9, we have:

$$(\mathbf{K} + \lambda \mathbf{W})^{-1} = \left(\alpha \mathbf{PLDL}^T \mathbf{P}^T + \mathbf{R}^{-1} \right)^{-1} = \mathbf{R} - \alpha \mathbf{RPLHL}^T \mathbf{P}^T \mathbf{R}.$$

From (5.4), (5.7), and (5.8), we have

$$\begin{aligned} \mathbf{DL}^T \check{c} =& \mathbf{DL}^T \mathbf{P}^T c = \mathbf{DL}^T \mathbf{P}^T (\mathbf{K} + \lambda \mathbf{W})^{-1} y \\ =& \mathbf{DL}^T \mathbf{P}^T \left(\mathbf{R} - \alpha \mathbf{RPLHL}^T \mathbf{P}^T \mathbf{R} \right) y = (\mathbf{D} - \mathbf{DFH}) \breve{y} \\ =& \mathbf{H} \breve{y} = z. \end{aligned}$$

Hence, $\check{c}$ solves the linear system $\mathbf{DL}^T \check{c} = z$. Again from (5.4), we have

$$y = (\mathbf{K} + \lambda \mathbf{W}) c = \alpha \overline{\mathbf{K}} c + \mathbf{R}^{-1} c,$$

so that

$$\begin{aligned} c &= \mathbf{R}\left[y - \alpha\overline{\mathbf{K}}c\right] = \mathbf{R}\left[y - \alpha\mathbf{PL}\left(\mathbf{DL}^T\mathbf{P}^T c\right)\right] \\ &= \mathbf{R}\left[y - \alpha\mathbf{PL}\left(\mathbf{DL}^T\breve{c}\right)\right] = \mathbf{R}\left[y - \alpha\mathbf{PL}z\right]. \end{aligned}$$

Finally,

$$\begin{aligned} c^j &= c(k^j) = \mathbf{R}(k^j,:)\left[y - \alpha\mathbf{PL}z\right] = \mathbf{R}(k^j,k^j)\left[y(k^j) - \alpha\mathbf{P}(k^j,:)\mathbf{L}z\right] \\ &= \mathbf{R}^j\left[y^j - \alpha\mathbf{L}(h^j,:)z\right]. \end{aligned}$$

□

Bibliography

A. Albert. Conditions for positive and nonnegative definiteness in terms of pseudoinverses. *SIAM Journal on Applied Mathematics*, 17(2):434–440, 1969.

J-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer, 2 edition, 2004.

D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. International series in operation research and management science. Springer, 2008.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005a.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005b.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, USA, 1970.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

W. Zangwill. *Non-linear Programming: A Unified Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1969.

F. Zhang, editor. *The Schur Complement and its applications*, volume 4 of *Numerical Methods and Algorithms*. Springer, 2005.

General bibliography

L. Aarons. Software for population pharmacokinetics and pharmacodynamics. *Clinical Pharmacokinetics*, 36(4):255–264, 1999.

A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csáki, editors, *Second International Symposium on Information Theory*. Académiai Kiadó, Budapest, 1973.

A. Albert. Conditions for positive and nonnegative definiteness in terms of pseudoinverses. *SIAM Journal on Applied Mathematics*, 17(2):434–440, 1969.

G. M. Allenby and P. E. Rossi. Marketing models of consumer heterogeneity. *Journal of Econometrics*, 89(1):57–78, 1999.

Q. An, C. Wang, I. Shterev, E. Wang, L. Carin, and D. Dunson. Hierarchical kernel stick-breaking process for multi-task image analysis. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 17–24. Omnipress, 2008.

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, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In Peter Auer and Ron Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 338–352. Springer Berlin / Heidelberg, 2005.

A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 25–32. MIT Press, Cambridge, MA, USA, 2007.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

N. Arora, G. M. Allenby, and J. Ginter. A hierarchical Bayes model of primary and secondary demand. *Marketing Science*, 17(1):29–44, 1998.

A. Auslender. *Optimisation Méthodes Numériques*. Masson, France, 1976.

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd Annual international conference on Machine learning (ICML 2005)*, pages 33–40. ACM Press, 2005.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21th Annual international conference on Machine learning (ICML 2004)*, page 6, New York, NY, USA, 2004. ACM Press.

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

S. Beal and L. Sheiner. *NONMEM User's Guide*. NONMEM Project Group, University of California, San Francisco, 1992.

S. L. Beal and L. B. Sheiner. Estimating population kinetics. *Critical Reviews in Biomedical Engineering*, 8(3):195–222, 1982.

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proceedings of Computational Learning Theory (COLT)*, 2003.

S. Ben-David, J. Gehrke, and R. Schuller. A theoretical framework for learning from a pool of disparate data sources. In *Proceedings of the The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 443–449, 2002.

S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer. Multi-task learning for HIV therapy screening. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 56–63. Omnipress, 2008.

E. V. Bonilla, F. V. Agakov, and C. K. I. Williams. Kernel multi-task learning using task-specific features. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors. *Large Scale Kernel Machines*. MIT Press, Cambridge, MA, USA, 2007.

J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, New York, NY, USA, 2002. ACM.

E. R. Carson, C. Cobelli, and L. Finkelstein. *The Mathematical Modeling of Metabolic and Endocrine Systems*. New York: Wiley, 1983.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

K-W. Chang, C-J. Hsieh, and C-J. Lin. Coordinate descent method for large-scale L2-loss linear support vector machines. *Journal of Machine Learning Research*, 9: 1369–1398, 2008.

Y-W. Chang, C-J. Hsieh, K-W. Chang, M. Ringgaard, and C-J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11:1471–1490, 2010.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, USA, 2006.

P-H Chen, R-E Fan, and C-J Lin. A study on SMO-type decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 17(4):893–908, 2006.

Y.-H. Chen and E. George. A bayesian model for collaborative filtering. In *Online Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, 1999.

P. Craven and G. Wahba. Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403, 1979.

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14 (3):641–668, 2002.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39:1–49, 2001.

M. Davidian and D. M. Giltinan. *Nonlinear Models for Repeated Measurement Data*. Chapman and Hall, 1995.

F. Dinuzzo and G. De Nicolao. An algebraic characterization of the optimum of regularized kernel methods. *Machine Learning*, 74(3):315–345, 2009.

P. Drineas and M. W. Mahoney. On the Nystrom method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

B. Efron. The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(14):619–632, 2004.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

R. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

R-E Fan, P-H Chen, and C-J Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6, 2005.

K. E. Fattinger and D. Verotta. A nonparametric subject-specific population method for deconvolution: I. description, internal validation and real data examples. *Journal of Pharmacokinetics and Biopharmaceutics*, 23:581–610, 1995.

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

J. Friedman, T. Hastie, H. Hoefling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

G. M. Fung and O. L. Mangasarian. Multicategory proximal support vector machine classifiers. *Machine Learning*, 59(1-2):77–97, 2005.

M. G. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

T. Glasmachers and C. Igel. Maximum-gain working set selection for SVMs. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

T. Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508, 2002.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

G. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 1996.

R. Gomeni, A. Lavergne, and E. Merlo-Pich. Modelling placebo response in depression trials using a longitudinal model with informative dropout. *European Journal of Pharmaceutical Sciences*, 36(1):4–10, 2009.

W. Greene. *Econometric Analysis*. Prentice Hall, 5 edition, 2002.

I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, editors. *Feature Extraction: Foundations and Applications*. Studies in Fuzziness and Soft Computing. Springer-Verlag, Secaucus, NJ, USA, 2006.

T. J. Hastie, R. J. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Springer-Verlag, Canada, 2nd edition, 2008.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

M. Hengland. Approximate maximum a posteriori with Gaussian process priors. *Constructive Approximation*, 26:205–224, 2007.

J-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer, 2 edition, 2004.

A. E. Hoerl and R. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

C. Hsieh, K.W. Chang, C.J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 408–415, Helsinki, Finland, 2008.

C. Hu and M. E. Sale. A joint model for nonlinear longitudinal data with informative dropout. *Journal of Pharmacokinetics and Pharmacodynamics*, 30(1):83–103, 2003.

F-L Huang, C-J Hsieh, K-W Chang, and C-J Lin. Iterative scaling and coordinate descent methods for maximum entropy models. *Journal of Machine Learning Research*, 11:815–848, 2010.

D. Hush and C. Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51(1):51–71, 2003.

J. A. Jacquez. *Compartmental analysis in biology and medicine*. University of Michigan Press, Ann Arbor, 1985.

T. Joachims. *Advances in Kernel Methods: Support Vector Machines*, chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1998.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–226, Philadelphia, PA, USA, 2006.

H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46(1–3):351–360, 2002.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

B. Kulis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. In *Proceedings of the 23rd Annual international conference on Machine learning (ICML 2006)*, pages 505–512, 2006.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In *Proceedings of the 21th International Conference in Machine Learning (ICML 2004)*, volume 69, page 65, 2004.

C-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001.

C-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 13:1045–1052, 2002.

N. List and H. U. Simon. A general convergence theorem for the decomposition method. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 363–377. Springer Berlin / Heidelberg, 2004.

N. List and H. U. Simon. General polynomial time decomposition algorithms. *Journal of Machine Learning Research*, 8:303–321, 2007.

Z. Lu, T. Leen, Y. Huang, and D. Erdogmus. A reproducing kernel Hilbert space framework for pairwise time series distances. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 624–631. Omnipress, 2008.

S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone. A convergent decomposition algorithm for support vector machines. *Computational Optimization and Applications*, 38:217–234, 2007.

D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. International series in operation research and management science. Springer, 2008.

D. J. Lunn, N. Best, A. Thomas, J. C. Wakefield, and D. Spiegelhalter. Bayesian analysis of population PK/PD models: general concepts and software. *Journal of Pharmacokinetics Pharmacodynamics*, 29(3):271–307, 2002.

P. Magni, R. Bellazzi, G. De Nicolao, I. Poggesi, and M. Rocchetti. Nonparametric AUC estimation in population studies with incomplete sampling: a Bayesian approach. *Journal of Pharmacokinetics Pharmacodynamics*, 29(5/6):445–471, 2002.

C. Mallows. Some comments on C_p. *Technometrics*, 15:661–675, 1973.

B. Matérn. *Spatial Variation*. Springer, New York, NY, USA, 1960.

A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London*, 209:415–446, 1909.

E. Merlo-Pich and R. Gomeni. Model-based approach and signal detection theory to evaluate the performance of recruitment centers in clinical trials with antidepressant drugs. *Clinical Pharmacology and Therapeutics*, 84:378–384, September 2008.

D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55(1-2):169–186, 2003.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005a.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005b.

C. A. Micchelli and M. Pontil. Feature space perspectives for learning the kernel. *Machine Learning*, 66:297–319, 2007.

C. N. Nett and W. M. Haddad. A system-theoretic appropriate realization of the empty matrix concept. *IEEE Transactions on automatic control*, 38(5):771–775, 1993.

M. Neve, G. De Nicolao, and L. Marchesi. Nonparametric identification of pharmacokinetic population models via Gaussian processes. In *Proceedings of 16th IFAC World Congress*, Praha, Czech Republic, 2005.

M. Neve, G. De Nicolao, and L. Marchesi. Nonparametric identification of population models via Gaussian processes. *Automatica*, 43(7):1134–1144, 2007.

C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *Proceedings of the 21th Annual international conference on Machine learning (ICML 2004)*, page 81, New York, NY, USA, 2004. ACM.

C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

M. Opper. *Online Learning in Neural Networks*, chapter A Bayesian Approach to Online Learning. Cambridge University Press, 1998.

J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Classics in Applied Mathematics. SIAM, 2000.

E. Osuna, Freund. R., and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of Computer Vision and Pattern Recognition*, pages 130–136, 1997.

L. Palagi and M. Sciandrone. On the convergence of a modified version of the svmlight algorithm. *Optimization Methods and Software*, 20:315–332, 2005.

K. Park, D. Verotta, T. F. Blaschke, and L. B. Sheiner. A semiparametric method for describing noisy population pharmacokinetic data. *Journal of pharmacokinetics and biopharmaceutics*, 25(5):615–642, 1997.

G. Pillonetto, G. De Nicolao, M. Chierici, and C. Cobelli. Fast algorithms for nonparametric population modeling of large data sets. *Automatica*, 45(1):173–179, 2009.

G. Pillonetto, F. Dinuzzo, and G. De Nicolao. Bayesian online multitask learning of Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):193–205, 2010.

J. Platt. Fast training of support vector machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA, 1998.

T. Poggio and F. Girosi. Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990.

Y. Qi, D. Liu, D. Dunson, and L. Carin. Multi-task compressive sensing with Dirichlet process priors. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 768–775. Omnipress, 2008.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. MIT Press, Cambridge, MA, USA, 2008.

A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, 2001.

S Ramaswamy, P Tamayo, R Rifkin, S Mukherjee, C H Yeang, M Angelo, C Ladd, M Reich, E Latulippe, J P Mesirov, T Poggio, W Gerald, M Loda, E S Lander, and T R Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences of the United States of America*, 98:15149–15154, 2001.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

R. Rifkin, G. Yeo, and T. Poggio. Regularized least squares classification. In Suykens, Horvath, Basu, Micchelli, and Vandewalle, editors, *Advances in Learning Theory: Methods, Model and Applications*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*, chapter 7, pages 131–154. VIOS Press, Amsterdam, 2003.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, USA, 1970.

W. Rudin. *Fourier Analysis on Groups*. Wiley-Interscience, New York, NY, USA, 1994.

S. Saitoh. *Theory of Reproducing Kernels and its Applications*, volume 189 of *Pitman Research Notes in Mathematics Series*. Longman Scientific and Technical, Harlow, 1988.

R. Schaback. Creating surfaces from scattered data using radial basis functions. In T. Lyche M. Dhlen and L.L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design III*, pages 477–496. Vanderbilt Univ. Press, 1995.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. (Adaptive Computation and Machine Learning). MIT Press, 2001.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

A. Schwaighofer, V. Tresp, and K. Yu. Learning Gaussian process kernels via hierarchical Bayes. In *Advances in Neural Information Processing Systems*, volume 17, pages 1209–1216, 2005.

L. Schwartz. Sous-espaces hilbertiens d'espaces vectoriels topologiques et noyaux associés (noyaux reproduisants). *J. Analyse Math.*, 13:115–256, 1964.

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

S. Shalev-Shwartz, Y. Singer, and N. Srebro. PEGASOS: Primal Estimated sub-GrAdient SOlver for Svm. In *Proceedings of the 24th Annual international conference on Machine learning (ICML 2007)*, pages 807–814, New York, NY, USA, 2007. ACM.

L. B. Sheiner. The population approach to pharmacokinetic data analysis: rationale and standard data analysis methods. *Drug Metabolism Reviews*, 15:153–171, 1994.

L. B. Sheiner and J. L. Steimer. Pharmacokinetic/pharmacodynamic modeling in drug development. *Annual Review of Pharmacology and Toxicology*, 40:67–95, 2000.

L. B. Sheiner, B. Rosenberg, and V. V. Marathe. Estimation of population characteristics of pharmacokinetic parameters from routine clinical data. *Journal of Pharmacokinetics and Biopharmaceutics*, 5(5):445–479, 1977.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th Annual international conference on Machine learning (ICML 2000)*, pages 911–918, 2000.

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, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*. MIT Press, 2005.

V. Srivastava and T. Dwivedi. Estimation of seemingly unrelated regression equations: A brief survey. *Journal of Econometrics*, 10:15–32, 1971.

T. Stamey, J. Kabalin, J. McNeal, I. Johnstone, F. Freiha, E. Redwine, and N. Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate ii radical prostatectomy treated patients. *Journal of Urology*, 16:1076–1083, 1989.

C. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9:1135–1151, 1981.

I. Steinwart, D. R. Hush, and C. Scovel. An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. *IEEE Transactions on Information Theory*, 52(10):4635–4643, 2006.

L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: a system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

S. Thrun. Is learning the n-th thing any easier than learning the first. In *Advances in Neural Information Processing Systems*, volume 8, pages 640–646. MIT Press, 1996.

S. Thrun and L. Y. Pratt, editors. *Learning To Learn*. Kluwer Academic Publishers, Boston, MA, 1998.

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

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D. C., 1977.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, June 2001.

P. Tseng and S. Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, pages 1–28, 2008.

I. Tsochantaridis. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, USA, 1998.

S. Vozeh, J. L. Steimer, M. Rowland, P. Morselli, F. Mentre, L. P. Balant, and L. Aarons. The use of population pharmacokinetics in drug development. *Clinical Pharmacokinetics*, 30(2):81–93, 1996.

G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, USA, 1990.

J. C. Wakefield, A. F. M. Smith, A. Racine-Poon, and A. E. Gelfand. Bayesian analysis of linear and non-linear population models by using the Gibbs sampler. *Applied Statistics*, 41:201–221, 1994.

H. Wendland. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics*, pages 389–396, 1995.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Proceedings of the 13th Annual Conference on Neural Information Processing Systems*, pages 682–688, Whistler, BC, Canada, 2000.

Q. Wu, Y. Ying, and D. Zhou. Multi-kernel regularized classifiers. *Journal of Complexity*, 23(1):108–134, 2007.

T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, 2(1):224–244, 2008.

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 22th Annual international conference on Machine learning (ICML 2005)*, pages 1012–1019, 2005.

S. Yu, V. Tresp, and K. Yu. Robust multi-task learning with t-processes. In *Proceedings of the 24th Annual international conference on Machine learning (ICML 2007)*, pages 1103–1110, New York, NY, USA, 2007. ACM.

L. Yuh, S. Beal, M. Davidian, F. Harrison, A. Hester, K. Kowalski, E. Vonesh, and R. Wolfinger. Population pharmacokinetic/pharmacodynamic methodology and applications: a bibliography. *Biometrics*, 50:566–575, 1994.

S. Yun and K.-C. Toh. A coordinate gradient descent method for ℓ_1-regularized convex minimization. *Computational Optimization and Applications*, pages 1–35, 2009.

W. Zangwill. *Non-linear Programming: A Unified Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1969.

F. Zhang, editor. *The Schur Complement and its applications*, volume 4 of *Numerical Methods and Algorithms*. Springer, 2005.

J. Zhang, Z. Ghahramani, and Y. Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.