Regularized Multi–Task Learning

Theodoros Evgeniou
Technology Management
INSEAD
Bd de Constance, 77300 Fontainebleau, France
theodoros.evgeniou@insead.edu

Massimiliano Pontil
Department of Computer Science
University College London
Gower St., London, WC1E 6BT UK
m.pontil@cs.ucl.ac.uk

ABSTRACT

Past empirical work has shown that learning multiple related tasks from data simultaneously can be advantageous in terms of predictive performance relative to learning these tasks independently. In this paper we present an approach to multi–task learning based on the minimization of regularization functionals similar to existing ones, such as the one for Support Vector Machines (SVMs), that have been successfully used in the past for single–task learning. Our approach allows to model the relation between tasks in terms of a novel kernel function that uses a task–coupling parameter. We implement an instance of the proposed approach similar to SVMs and test it empirically using simulated as well as real data. The experimental results show that the proposed method performs better than existing multi–task learning methods and largely outperforms single–task learning using SVMs.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning.

General Terms

Algorithms, Theory.

Keywords

Multi–Task Learning, Support Vector Machines, Regularization, Kernel Methods.

1. INTRODUCTION

In many practical situations a number of statistical models need to be estimated from data. For example multi–modal human computer interface requires the modeling of both, say, speech and vision; machine vision problems may themselves require the estimation of multiple models, for example one for detecting each object, i.e. a face, from a pool of similar objects; in finance forecasting models for predicting the value of many possibly related indicators simultaneously is often required; in marketing modeling the preferences of many individuals simultaneously is common practice [1, 2].

When there are relations between the tasks to learn, it can be advantageous to learn all tasks simultaneously instead of following the more traditional approach of learning each task independently of the others. There has been a lot of experimental work showing the benefits of such multi–task learning relative to individual task learning when tasks are related, see [4, 11, 15, 22]. There have also been various attempts to theoretically study multi–task learning, see [4, 5, 6, 7, 8, 15, 23].

In this paper we develop methods for multi–task learning that are natural extensions of existing kernel based learning methods for single task learning, such as Support Vector Machines (SVMs) [25]. To the best of our knowledge, this is the first generalization of regularization–based methods from single–task to multi–task learning. We test an instance of the proposed methods experimentally using both simulated and real data. The experiments show that the proposed method performs better than existing multi–task learning methods and largely outperforms single–task learning.

1.1 Related Work

A statistical learning theory based approach to multi–task learning has been developed in [5, 6] and [8]. In [6] the problem of bias learning is considered, where the goal is to choose an optimal hypothesis space from a family of hypothesis spaces. In [6] the notion of the "extended VC dimension" (for a family of hypothesis spaces) is defined and it is used to derive generalization bounds on the average error of T tasks learned which is shown to decrease at best as $\frac{1}{T}$. In [5] the same setup was used to answer the question "how much information is needed per task in order to learn T tasks" instead of "how many examples are needed for each task in order to learn T tasks", and the theory is developed using Bayesian and information theory arguments instead of VC dimension ones. In [8] the extended VC dimension was used to derive tighter bounds that hold for each task (not just the average error among tasks as considered in [6]) in the case that the learning tasks are related in a particular way defined.

The problem of multi–task learning has been also studied in the statistics literature. Breiman and Friedman [9] propose the curds&whey method, where the relations be-

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

tween the various tasks are modeled in a post–processing fashion. Brown and Zidek [10] consider the case of regression and propose an extension of the standard ridge regression to multivariate ridge regression. Finally, a number of approaches for learning multiple tasks or for learning to learn [22] are Bayesian, where a probability model capturing the relations between the different tasks is estimated simultaneously with the models' parameters for each of the individual tasks. In [1, 2] a hierarchical Bayes model is estimated. First, it is assumed that the parameters of the T functions to be learned are all sampled from an unknown Gaussian distribution. Then, an iterative Gibbs sampling based approach is used to simultaneously estimate both the individual functions and the parameters of the Gaussian distribution. In this model relatedness between the tasks is captured by this Gaussian distribution: the smaller the variance of the Gaussian the more related the tasks are. Finally, [4, 15] suggest a similar hierarchical model. In [4] a mixture of Gaussians for the "upper level" distribution instead of a single Gaussian is used. This leads to clustering the tasks, one cluster for each Gaussian in the mixture.

1.2 Notation and Setup

We consider the following setup. We have T learning tasks and we assume that all data for the tasks come from the same space $X \times Y$. For simplicity we assume that $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}$. For each task we have m data points

$$\{(\mathbf{x}_{1t}, y_{1t}), (\mathbf{x}_{2t}, y_{2t}) \ldots, (\mathbf{x}_{mt}, y_{mt})\}$$

sampled from a distribution P_t on $X \times Y$. So the total data available is:

$$\{\{(\mathbf{x}_{11}, y_{11}), \ldots, (\mathbf{x}_{m1}, y_{m1})\}, \ldots \{(\mathbf{x}_{1T}, y_{1T}), \ldots, (\mathbf{x}_{mT}, y_{mT})\}\}.$$

We assume that P_t is different for each task but that the P_t's are related – as, for example, considered in [8]. The goal is to learn T functions $f_1, f_2, \ldots, f_T$ such that $f_t(\mathbf{x}_{it}) \approx y_{it}$. The case $T = 1$ is the standard (single–task) learning problem.

There are various versions of this setup. A simpler version is when the same input data $\mathbf{x}_{it}$ are used for all the tasks. That is, for every $i \in \{1, \ldots, m\}$ the vector $\mathbf{x}_{it}$ is the same for all $t \in \{1, \ldots, T\}$, but the output values y_{it} differ for each t. This is for example the standard setup in marketing applications of preference modeling [1, 2] where the same choice panel questions (the same "$\mathbf{x}$'s") are given to many individual consumers, each individual provides his/her own preferences (the "y's"), and we assume that there is some commonality among the preferences of the individuals (the "f_t's"). We consider this preference modeling application in the experiments section below.

Clearly one can consider other scenarios, too. For example: a) the case of having the same output ("y's") and different inputs ("$\mathbf{x}$'s"), which corresponds to the problem of integrating information from heterogeneous databases [7]; or, b) the case of multi–modal learning or learning by components, where the $(\mathbf{x}, y)$ data for each of the tasks do not belong to the same space $X \times Y$ but data for task t come from a space $X_t \times Y_t$ – this is for example the machine vision case of learning to recognize a face by first learning to recognize parts of the face, such as eyes, mouth, and nose [14]. Each of these related tasks can be learned using images of different size (or different representations). The methods we develop below may be extended to handle such scenarios, for example through the appropriate choice of a matrix–valued kernel [20] discussed in section 2.2.

2. METHODS FOR MULTI–TASK LEARNING

For simplicity we first assume that function f_t for the tth task is a hyperplane, that is $f_t(\mathbf{x}) = \mathbf{w}_t \cdot \mathbf{x}$, where "$\cdot$" denotes the standard inner product in $\mathbb{R}^d$. The generalization to nonlinear models will then be done through the use of Reproducing Kernel Hilbert Spaces (RKHS), see for example [20, 25, 26]. In the case of classification each y_{it} takes the values ± 1, and f_t is the sign of $\mathbf{w}_t \cdot \mathbf{x}$. Below we consider this case – regression can be treated similarly.

All previously proposed frameworks and methods for multi–task learning (i.e. those discussed in the introduction) are based on some formal definition of the notion of relatedness of the tasks. This relatedness is then formalized through the design of a multi–task learning method. For example, hierarchical Bayesian methods [1, 2, 4, 15] assume that all functions $\mathbf{w}_t$ come from a particular probability distribution such as a Gaussian. This implies that all $\mathbf{w}_t$ are "close" to some mean function $\mathbf{w}_0$ (the mean of the Gaussian).

We follow the intuition of Hierarchical Bayes [1, 2, 15]. In particular we assume that all $\mathbf{w}_t$ can be written, for every $t \in \{1, \ldots, T\}$, as

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t \tag{1}$$

where the vectors $\mathbf{v}_t$ are "small" when the tasks are similar to each other. In other words we assume that the tasks are related in a way that the true models are all close to some model $\mathbf{w}_0$ (playing the role of the mean of the Gaussian used for Hierarchical Bayes [1, 2]). We then estimate all $\mathbf{v}_t$ as well as the (common) $\mathbf{w}_0$ simultaneously. To this end we solve the following optimization problem which is analogous to SVMs used for single task learning:

PROBLEM 2.1.

$$\min_{\mathbf{w}_0, \mathbf{v}_t, \xi_{it}} \Big\{ \; J(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}) :=$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_{it} + \frac{\lambda_1}{T} \sum_{t=1}^{T} \|\mathbf{v}_t\|^2 + \lambda_2 \|\mathbf{w}_0\|^2 \Big\} \tag{2}$$

subject, for all $i \in \{1, 2, \ldots, m\}$ and $t \in \{1, 2, \ldots, T\}$, to the constraints that

$$\begin{aligned} y_{it}(\mathbf{w}_0 + \mathbf{v}_t) \cdot \mathbf{x}_{it} &\geq 1 - \xi_{it} \\ \xi_{it} &\geq 0. \end{aligned} \tag{3}$$

In this problem, λ_1 and λ_2 are positive regularization parameters and the ξ_{it} are slack variables measuring the error that each of the final models $\mathbf{w}_t$ makes on the data. We therefore impose a regularization constraint on the "average" model $\mathbf{w}_0$ and control how much the solutions $\mathbf{w}_t$ differ from each other by controlling the size of the $\mathbf{v}_t$. Intuitively, for a fixed value of λ_2 a large value of the ratio $\frac{\lambda_1}{\lambda_2}$, say $\frac{\lambda_1}{\lambda_2} > 100$, will tend to make the models to be the same model (that is, the $\mathbf{v}_t$ are nearly equal to zero), while for a fixed value of λ_1 a small value of the ratio $\frac{\lambda_1}{\lambda_2}$, say $\frac{\lambda_1}{\lambda_2} < 0.01$, will tend to make all the tasks unrelated ($\mathbf{w}_0$ nearly equal to zero). In

particular, when λ_1 tends to infinity problem 2.1 reduces to solving one single–task learning problem (finding $\mathbf{w}_0$, having $\mathbf{v}_t = 0$ for every $t \in \{1, \ldots, T\}$). On the other hand when λ_2 tends to infinity problem 2.1 reduces to solving the T tasks independently (finding the $\mathbf{v}_t$, having $\mathbf{w}_0 = 0$).

Let $\mathbf{w}_0^*$ and $\mathbf{v}_t^*$ be the optimal solution of problem 2.1 and $\mathbf{w}_t^* := \mathbf{w}_0^* + \mathbf{v}_t^*$. Our next observation shows a relation between these quantities.

LEMMA 2.1. *The optimal solution to the multi–task optimization method (3) satisfies the equation*

$$\mathbf{w}_0^* = \frac{\lambda_1}{\lambda_2 + \lambda_1} \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t^*. \tag{4}$$

PROOF. This result follows by inspecting the Lagrangian function for problem 2.1. This is given by the formula

$$L(\mathbf{w}_0, \mathbf{v}_t, \alpha_{it}, \gamma_{it}) = J(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}) -$$

$$- \sum_{t=1}^{T} \sum_{i=1}^{m} \alpha_{it}(y_{it}(\mathbf{w}_0 + \mathbf{v}_t) \cdot \mathbf{x}_{it} - 1 + \xi_{it}) - \sum_{t=1}^{T} \sum_{i=1}^{m} \gamma_{it} \xi_{it} \tag{5}$$

where α_{it} and γ_{it} are nonnegative Lagrange multipliers. Setting the derivative of L with respect to $\mathbf{w}_0$ to zero gives the equation

$$\mathbf{w}_0^* = \frac{1}{2\lambda_2} \sum_{t=1}^{T} \sum_{i=1}^{m} \alpha_{it} y_{it} \mathbf{x}_{it}.$$

The same operation for $\mathbf{v}_t$ gives, for every $t \in \{1, \ldots, T\}$, the equation

$$\mathbf{v}_t^* = \frac{T}{2\lambda_1} \sum_{i=1}^{m} \alpha_{it} y_{it} \mathbf{x}_{it}.$$

By combining these equations we obtain that

$$\mathbf{w}_0^* = \frac{\lambda_1}{T\lambda_2} \sum_{t=1}^{T} \mathbf{v}_t^*.$$

The result now follows by this equation and equation (1). □

This lemma suggests that we can replace $\mathbf{w}_0$ in equation (3) with an expression of $\mathbf{v}_t$ and obtain an optimization problem which involves only the $\mathbf{v}_t$'s. Replacing $\mathbf{w}_t$'s for the $\mathbf{v}_t$'s and choosing appropriate regularization parameters instead, leads to the following lemma:

LEMMA 2.2. *The multi–task problem 2.1 is equivalent to solving the following optimization problem:*

PROBLEM *2.2.*

$$\min_{\mathbf{w}_t, \xi_{it}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_{it} + \right.$$

$$\left. + \rho_1 \sum_{t=1}^{T} \|\mathbf{w_t}\|^2 + \rho_2 \sum_{t=1}^{T} \left\| \mathbf{w}_t - \frac{1}{T} \sum_{s=1}^{T} \mathbf{w}_s \right\|^2 \right\} \tag{6}$$

subject, for all $i \in \{1, 2, \ldots, m\}, t \in \{1, 2, \ldots, T\}$, to the constraints that

$$\begin{aligned} y_{it} \mathbf{w}_t \cdot \mathbf{x}_{it} &\geq 1 - \xi_{it} \\ \xi_{it} &\geq 0 \end{aligned}$$

where the parameters ρ_1 and ρ_2 are related to λ_1 and λ_2 by the equations

$$\rho_1 = \frac{1}{T} \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} \tag{7}$$

and

$$\rho_2 = \frac{1}{T} \frac{\lambda_1^2}{\lambda_1 + \lambda_2}. \tag{8}$$

PROOF. Using equations (1) and (4), we rewrite the stabilizer in the objective function J in equation (2) as

$$\lambda_1 \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{v}_t\|^2 + \lambda_2 \|\mathbf{w}_0\|^2 =$$

$$= \lambda_1 \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{w}_t\|^2 - \frac{1}{T^2} \frac{\lambda_1^2}{\lambda_1 + \lambda_2} \left\| \sum_{t=1}^{T} \mathbf{w}_t \right\|^2. \tag{9}$$

On the other hand the stabilizer in equation (6),

$$\rho_1 \sum_{t=1}^{T} \|\mathbf{w_t}\|^2 + \rho_2 \sum_{t=1}^{T} \left\| \mathbf{w}_t - \frac{\sum_{s=1}^{T} \mathbf{w}_s}{T} \right\|^2 \tag{10}$$

can be rewritten as

$$(\rho_1 + \rho_2) \sum_{t=1}^{T} \|\mathbf{w}_t\|^2 +$$

$$+ \rho_2 \left(\frac{1}{T} \left\| \sum_{t=1}^{T} \mathbf{w}_t \right\|^2 - \frac{2}{T} \left(\sum_{t=1}^{T} \mathbf{w}_t \right) \cdot \left(\sum_{s=1}^{T} \mathbf{w}_s \right) \right) =$$

$$= (\rho_1 + \rho_2) \sum_{t=1}^{T} \|\mathbf{w}_t\|^2 - \rho_2 \frac{1}{T} \left\| \sum_{t=1}^{T} \mathbf{w}_t \right\|^2. \tag{11}$$

The result now follows by observing that equations (9) and (11) coincide provided that ρ_1 and ρ_2 satisfy equations (7) and (8). □

Thus our regularization method finds a trade off between small size parameter vectors for each model and closeness of these model parameters to the average of the model parameters. In SVMs language we are finding a trade off between each SVM having large margin (this quantity is defined as $1/\|w_t\|$) and having each SVM close to the average SVM.

2.1 Dual Optimization Problem

We now derive the dual of problem 2.1. To this end, one may follow the standard duality theory approach, see e.g. [19] to optimize the Lagrangian function (5). However, the following observation allows us to directly link the dual of problem 2.1 to the standard SVM dual problem, see e.g. [25].

The set of functions $f_t(\mathbf{x}) = \mathbf{w}_t \cdot \mathbf{x}$, $t = 1, \ldots, T$ can be identified by a real–valued function

$$F : X \times \{1, \ldots, T\} \to \mathbb{R}$$

defined as

$$F(\mathbf{x}, t) = f_t(\mathbf{x}). \tag{12}$$

Learning this function requires examples of the type $((\mathbf{x}, t), y)$, where $(\mathbf{x}, t) \in X \times \{1, \ldots, T\}$ and $y \in \{-1, 1\}$.

We assume that the reader is familiar with the notion of *feature map* and of *kernels*, see e.g. [25] for a discussion. We note that F can be represented by means of the feature map

$$\Phi((\mathbf{x}, t)) = (\frac{\mathbf{x}}{\sqrt{\mu}}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{t-1}, \mathbf{x}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{T-t}) \tag{13}$$

where we have denoted by $\mathbf{0}$ the vector in $\mathbb{R}^d$ whose coordinates are all zero, $\mu = \frac{T\lambda_2}{\lambda_1}$, and we are now estimating a vector

$$\mathbf{w} = (\sqrt{\mu}\mathbf{w}_0, \mathbf{v}_1, \ldots, \mathbf{v}_T).$$

By construction we have that

$$\mathbf{w} \cdot \Phi((\mathbf{x}, t)) = (\mathbf{w}_0 + \mathbf{v}_t) \cdot \mathbf{x}$$

and

$$\|\mathbf{w}\|^2 = \sum_{t=1}^{T} \|\mathbf{v}_t\|^2 + \mu \|\mathbf{w}_0\|^2.$$

It is then clear that solving the SVM multi–task problem (1) is equivalent to learning the function F in equation (12) with a standard SVM which uses the kernel associated to the feature map (13). Consequently, using the standard SVM dual problem, see e.g. [25], we have the following theorem:

THEOREM 2.1. *Let $C := \frac{T}{2\lambda_1}$, $\mu = \frac{T\lambda_2}{\lambda_1}$, and define the kernel*

$$K_{st}(\mathbf{x}, \mathbf{z}) := \left(\frac{1}{\mu} + \delta_{st}\right) \mathbf{x} \cdot \mathbf{z}, \quad s, t = 1, \ldots, T. \tag{14}$$

The dual problem of 2.1 is given by

PROBLEM *2.3.*

$$\max_{\alpha_{it}} \left\{ \sum_{i=1}^{m} \sum_{t=1}^{T} \alpha_{it} - \right.$$

$$\left. \frac{1}{2} \sum_{i=1}^{m} \sum_{s=1}^{T} \sum_{j=1}^{m} \sum_{t=1}^{T} \alpha_{is} y_{is} \alpha_{jt} y_{jt} K_{st}(\mathbf{x}_{is}, \mathbf{x}_{jt}) \right\} \tag{15}$$

subject, for all $i \in \{1, 2, \ldots, m\}$ and $t \in \{1, 2, \ldots, T\}$, to the constraints that the constraint that

$$0 \leq \alpha_{it} \leq C.$$

In addition, if α^*_{it} is a solution to the above problem, the solution to problem 2.1 is given by

$$f^*_t(\mathbf{x}) = \sum_{i=1}^{m} \sum_{s=1}^{T} \alpha^*_{is} K_{st}(\mathbf{x}_{is}, \mathbf{x}).$$

It is therefore required to select two parameters: the parameter C for the training error as in the standard SVM case, and the parameter μ that captures the similarity between the tasks. These two parameters can be selected for example using a validation set or using some form of cross-validation. For example one may select μ through a task-cross-validation process where instead of leaving training points out as done for the typical cross validation case, tasks are left out. We do not have any theoretical justification for doing this or any theory that can lead to some method for selecting parameter μ, and we leave this as an open question. In the experiments below we either report the performances for all the parameters we tested (to see their effects) or simply selected both C and μ among a small number of choices (less than 20 pairs (C, μ) in total) based on the actual test performance. As observed in [13] when the number of test data is large (for all experiments below we have more than 2800 test data in total for all the tasks) then model selection among only a small number of models leads to a choice that has actual test performance similar to the one observed on the test data used.

Moreover, notice that solving the optimization problem 2.3 requires solving a standard SVM problem with Tm training data. Assuming that a standard SVM training method is used which requires O([number of training data]^3]) time, this implies that to solve problem 2.3 we need $O(T^3m^3)$ time. Instead, if we were to solve each task independently we would only need $O(Tm^3)$ time. In the experiments below, since we had only relatively small Ts and ms, we did not have any significant differences in the training time of single versus multi-task SVM training. In many practical applications, however, this may be an issue. Optimizing the running time of the multi-task learning method we propose is therefore an important practical issue that we leave as an open problem. We conjecture that with an appropriate choice of kernels and parameters C for each task it may be possible to solve T independent SVMs that will lead to the same solutions as the ones found using the proposed multi-task learning method.

2.2 Non–linear multi–task learning

An important characteristic of SVM is that they can be used to estimate highly non-linear functions through the use of kernels [25]. We can clearly generalize the linear multi-task learning method outlined above to the non-linear case using kernels as is done for SVM. Morevoer, the above observation in theorem 2.1 can be generalized to include non–linear multi–task learning. We simply learn F by using a nonlinear feature map

$$\Phi : X \times \{1, \ldots, T\} \to \mathcal{H}$$

where $\mathcal{H}$ is a separable Hilbert space (the feature space). The kernel associated to Φ is

$$G((\mathbf{x}, t), (\mathbf{z}, s)) = \langle \Phi((\mathbf{x}, t)), \Phi((\mathbf{z}, s)) \rangle$$

where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathcal{H}$. In general we can also consider situations where each task is trained on different number of examples. That is, we sample our examples $((\mathbf{x}_i, t_i), y_i) \in (X \times \{1, \ldots, T\}) \times \{-1, 1\}$, $i = 1, \ldots, N$, and learn the coefficients β_i for the function

$$F(\mathbf{x}, t) = \sum_{1=1}^{N} \beta_i G((\mathbf{x}, t), (\mathbf{x}_i, t_i)) \tag{16}$$

by solving the standard SVM dual problem with kernel G, namely

PROBLEM 2.4.

$$\max_{\beta_i} \left\{ \sum_{i=1}^{N} \beta_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \beta_i y_i \beta_j y_j G((\mathbf{x}_i, t_i), (\mathbf{x}_j, t_j)) \right\} \tag{17}$$

subject to the constraint that $\beta_i \in [0, C]$ for every i.

In particular problem 2.3 reduces to problem 2.4 if we set $N = mT$, define, for $i \in \{1, \ldots, N\}$

$$\begin{aligned} \mathbf{x}_i &= \mathbf{x}_{(i \bmod T)(i \bmod m)} \\ y_i &= y_{(i \bmod T)(i \bmod m)} \end{aligned}$$

and use the kernel

$$G((\mathbf{x}_i, t_i), (\mathbf{x}_j, t_j)) = K_{t_i t_j}(\mathbf{x}_i, \mathbf{x}_j)$$

where K is given by equation (14) where we can replace the dot product $\mathbf{x} \cdot \mathbf{z}$ with a nonlinear kernel as is done for standard SVM. We note that the above ideas appear in greater generality in [20] where the notion of operator–valued kernels is derived.

3. EXPERIMENTS

We run two types of experiments. The first one is with simulated data in order to study the behavior of the proposed approach under varying conditions. We then tested the method on a real dataset.

3.1 Simulated Data

We tested the proposed method using data that capture the preferences of individuals (consumers) when they choose among products. This is the standard problem of conjoint analysis [1, 2] for preference modeling. It turns out [12] that this problem is equivalent to solving a classification problem, therefore the results we report below can be seen as results for a classification problem.

We followed the basic simulation design used by other researchers in the past. In particular we simply replicated the experimental setup of [3, 12, 24]. For completeness we briefly describe that setup.

We generated data describing products with 4 attributes (i.e. size, weight, functionality, and ease–of–use of a product), each attribute taking 4 values (i.e. very high value (1000), high value (0100), low value (0010), very low value (0001)). Therefore each product was represented by a $4 \times 4 = 16$ dimensional binary vector. Each question given to an individual consumer consists of 4 such (vectors) products to choose from. Each question was subsequently transformed into 6 data points (twice the number of comparisons of the "winner" product among the four and the remaining 3 "loser" products) of 16 dimensions each that were used for the classification learning problem corresponding to this preference modeling problem [12]. Therefore the training data for each task are 16-dimensonal vectors with elements that take values only $\{+1, -1, 0\}$ – the outcome of taking the difference between two binary vectors describing two products. The questions were generated randomly. We generated 16 questions per individual, hence the individual classification problems used $16 \times 6 = 96$ 16–dimensional training data. We simulated 100 or 30 individuals, hence we had a total of 100 (or 30) tasks: one task for each individual in order to estimate the "utility function" of that individual using his/her responses to the 16 questions given represented with the ± 1 labels (prefer or not prefer one product from another) of the 96 training data used for the classification problem for that individual.

The four utility function coefficients (corresponding to the four values an attribute can take) for each of the four attributes were generated randomly from a Gaussian with mean

$$\left(-\beta, -\frac{1}{3}\beta, \frac{1}{3}\beta, \beta\right).$$

We used the same Gaussian for each of the four attributes. The actual utility function $\mathbf{w}_t$ was therefore a vector

$$\mathbf{w}_t = (w_{t1}, ..., w_{t4}, w_{t5}, ..., w_{t8}, w_{t9}, ..., w_{t12}, w_{t12}, ..., w_{t16})$$

where $(w_{t1}, ..., w_{t4})$, $(w_{t5}, ..., w_{t8})$, $(w_{t9}, ..., w_{t12})$, and $(w_{t12}, ..., w_{t16})$ are four 4–dimensional vectors sampled from the aforementioned Gaussian. Notice that the functions $\mathbf{w}_t$ we are estimating are real-valued, while the training data are vectors with values only $\{+1, -1, 0\}$ as described above. It turns out that parameter β controls the noise of the data (i.e. the response accuracy of the individual consumers) which is modeled according to the assumptions of Hierarchial Bayes (HB), see [1, 2]. As in [3, 12, 24] we used $\beta = 3$ for low noise in the data and $\beta = 0.5$ for high noise in the data. We modeled the similarity among the 100 (or 30) individuals, hence the similarity among the 100 (or 30) tasks to be learned, by varying the variance σ^2 of the Gaussian from which the true utility functions $\mathbf{w}_t$ were generated. The covariance matrix of the Gaussian was a diagonal matrix with all diagonal elements being σ^2. We modeled low similarity among the tasks using $\sigma^2 = 3\beta$, and high similarity using $\sigma^2 = 0.5\beta$, like again in [3, 12, 24]. As discussed in [3, 12, 24] these parameters are chosen so that the range of average utility functions, noise, and similarities among the preferences of the individual consumers found in practice is covered.

Therefore in this experiment we tested all 2×2 scenarios in terms of: a) the amount of noise in the data (low versus high), and b) the similarity among the tasks to be learned (low versus high). All experiments were repeated five times – so a total of 500 (or 150) individual utility functions $\mathbf{w}_t$ were estimated – and the average performance is reported. We used two measures of performance: a) the Root Mean Square Error (RMSE) of the estimated utility functions relative to the true (supposedly unknown) utility functions – an error measure typically used for preference modeling [3, 12, 24]; b) the average hit errors (misclassification) of the estimated functions on a test set of 16 questions per individual – hence a total of 96 test data per individual for the corresponding classification problem solved leading to 2880 and 9600 test data for the 30 and 100 tasks cases, respectively. We note that the conclusions are qualitatively the same for both error measures, as also observed by [12, 24].

We compared our method with Hierarchical Bayes (HB) [1, 2] which is considered to be a state of the art method for preference modeling of a population of individual consumers. It is important to note that *in all cases we generated the data in a way that gives an advantage to HB* – that is, the data were generated, as described above, according to the probability distributions *assumed* by HB.

We tested various values (namely, 0.1, 0.5, 1, 2, 10, 1000) of the task–coupling parameter μ used in the definition of kernel (14) to examine its effects. We also tested the case of using one SVM for all tasks as if we assume that all data come from the same task – which corresponds to the limit $\mu \to 0$. We did these for $C = 0.1$ and $C = 1$. In Figures 1 and 2 we show the effects of μ for $C = 0.1$ and for 30 and 100 tasks, respectively – the plots are similar for $C = 1$. We report the results for RMSE – which are similar for the hit test error. The left-most point at each graph is for solving one SVM only for all tasks as if we assume that there is only

one task, corresponding to $\mu \to 0$ – clearly we do not log the $\mu = 0$ in the figure and we use instead a small μ to visualize the results.

Firstly, the results show the importance of the parameter μ, and that a wrong selection of μ may some times decrease performance relative to solving T SVMs. This is for example the case when the tasks have low similarity (the right side of Figures 1 and 2) and we use a very small μ – where, in the limit, using only one SVM for all the data as if there is only one task ($\mu = 0$) can hurt performance a lot. Moreover, the optimal μ can have a significantly better performance than, for example, a very large or a very small μ corresponding to having T SVMs or 1 SVM for all tasks, respectively. It is therefore important to select μ correctly, an issue for which we do not have currently a method as discussed in section 2.1.

The results also show clearly the benefits of solving all tasks simultaneously using the proposed method. The advantage of the proposed method relatively to learning each task independently (estimating one utility function per individual) is higher when there is more similarity among the tasks (similarity among the true utility functions of the individuals). Moreover, the proposed method performs similarly or better than HB when μ is selected appropriately. In Tables 1 and 2 we report the average performance of the T independent SVMs, the performance of having a single SVM for all tasks (as if we assume all data come from the same task), the performance of HB, and the performance of the proposed method for the best pair (C, μ) (from Figures 1 and 2) for both 30 and 100 tasks. Given that we have a large test set (2880 and 9600 test data for the 30 and 100 tasks, respectively) and we choose (C, μ) among only $2 \times 6 = 12$ pairs $(C, \mu) \in \{(0.1, 1) \times (0.1, 0.5, 1, 2, 10, 1000)\}$ (or just among 2 values of C $(0.1, 1)$ for the T independent SVMs and for the 1 SVM), the test performances we report are very close to the actual test performances, as we discussed in section 2.1. Notice also that when there are few tasks (30 in this case) the proposed method is relatively better than HB than when there are many tasks (100 in this case).

Noise	Similar	HB	$\mu = 0.1$	T SVMs	1 SVM
H	L	0.85 26.14%	0.81* 25.86%*	0.84 26.22%	1.32 38.2%
H	H	0.90 31.03%	**0.86** **30.58%**	0.97 31.60%	0.96 33.2%
L	L	0.60 14.34%	0.58* 14.12%*	0.65 16.00%	1.00 24.1%
L	H	0.48 13.42%	0.46* 13.19%*	0.68 17.11%	0.57 15.8%

Table 1: Comparison of methods using RMSE and hit error rates. There are 30 individuals. The C of both the T individual SVMs and the proposed method is 0.1. Bold indicates best or not significantly different than best at $p < 0.05$. A * indicates best or not significantly different than best at $p < 0.10$.

3.2 Using a Real Dataset

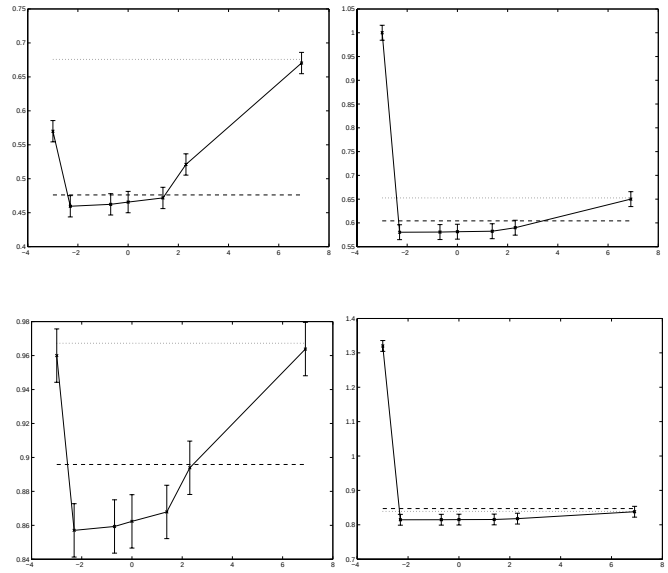

Figure 1: Horizontal axis is the $\log(\mu)$ ($\mu = 0.1, 0.5, 1, 2, 10, 1000$) of the proposed method. As in Table 1, the C of both the T individual SVMs and the proposed method is 0.1. The Vertical axis is the average RMSE of the estimated functions. Dashed straight line is the RMSE of HB. Dotted horizontal line is the average RMSE when we estimate one SVM per individual. The solid line is the average RMSE of the proposed method - the left-most point at each graph is for solving one SVM only for all tasks as if we assume that there is only one task - the limit of $\mu = 0$ (we don't log $\mu = 0$ clearly). There are 30 individuals. The C of both the individual SVMs and of the proposed method is 0.1. Top: Noise is low. Bottom: Noise is high. Left: Similarity of tasks is high. Right: Similarity of tasks is low.

We also tested the method using the "school data" from the Inner London Education Authority available at *multilevel.ioe.ac.uk/intro/datasets.html*.
We selected this dataset to compare our method directly with the work of [4] where a number of multi-task learning methods are compared using this dataset. This data consists of examination records of 15362 students from 139 secondary schools. The goal is to predict the exam scores of the students based on the following inputs: year of the exam, gender, VR band, ethnic group, percentage of students eligible for free school meals in the school, percentage of students in VR band one in the school, gender of the school (i.e. male, female, mixed), and school denomination. We represented the categorical variables using binary (dummy) variables, so the total number of inputs for each student in each of the schools was 27. Since the goal is to predict the exam scores of the students we run regression using the SVM ϵ–loss function [25] for the multi–task learning method proposed. We consider each school to be "one task". Therefore we had 139 tasks. We made 10 random splits of the data into training (75% of the data, hence around 70 students per school on average) and test (the remaining 25% of the data, hence around 40 students per school on average) data and we measured the generalization performance using the

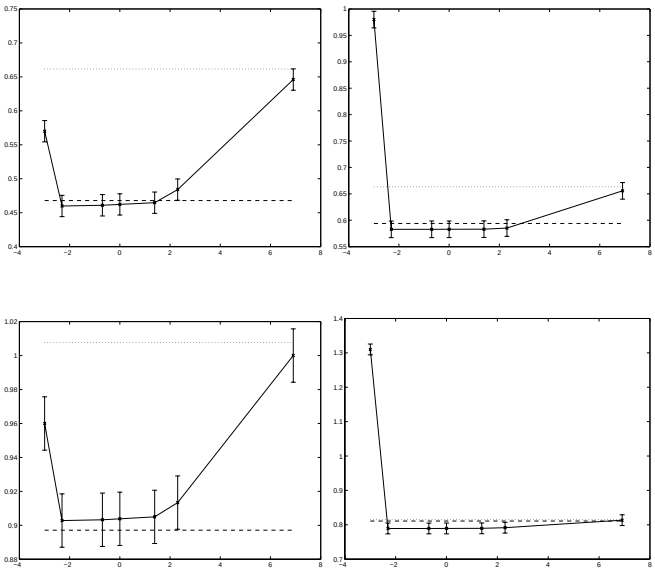

Figure 2: Horizontal axis is the $\log(\mu)$ ($\mu = 0.1, 0.5, 1, 2, 10, 1000$) of the proposed method. As in Table 1, the C of both the T individual SVMs and the proposed method is 0.1. The Vertical axis is the average RMSE of the estimated functions. Dashed straight line is the RMSE of HB. Dotted horizontal line is the average RMSE when we estimate one SVM per individual. The solid line is the average RMSE of the proposed method - the left-most point at each graph is for solving one SVM only for all tasks as if we assume that there is only one task - the limit of $\mu = 0$ (we don't log $\mu = 0$ clearly). There are 100 individuals. The C of both the individual SVMs and of the proposed method is 0.1. Top: Noise is low. Bottom: Noise is high. Left: Similarity of tasks is high. Right: Similarity of tasks is low.

explained variance of the test data as a measure in order to have a direct comparison with [4] where this error measure is used. The explained variance is defined in [4] to be the total variance of the data minus the sum–squared error on the test set as a percentage of the total data variance, which is a percentage version of the standard R^2 error measure for regression for the test data. Finally, we used a simple linear kernel for each of the tasks.

The results for this experiment are reported in Table 3. For comparison we also report the performance of the task clustering method reported in [4]. We show the results for all the parameters C and μ we tested, other than $\mu = 0$ corresponding to having 1 SVM for all tasks. We let the ratio $\mu = \frac{T\lambda_2}{\lambda_1}$ vary to see the effects. As with the previous experiments, when μ is large we get the performance of solving one SVM regression model per school (per task).

The results, although still preliminary, show the advantage of learning all tasks (for all schools) simultaneously instead of learning them one by one. Moreover, even the simple linear kernel (14) significantly outperforms the Bayesian method of [4], which is in turn better than other methods as compared in [4]. It turns out that for this dataset 1 SVM for all tasks performs the same as the best performance we report here for $\mu = 0.5$, hence it appears that the particular

Noise	Similar	HB	$\mu = 0.1$	T SVMs	1 SVM
H	L	0.81 24.65%	**0.79** **24.24%**	0.82 24.98%	1.31 38.9%
H	H	**0.90** **31.49%**	**0.90** **31.48%**	1.01 33.13%	0.96 33.0%
L	L	**0.59** **13.97%**	**0.58** **14.02%**	0.66 15.57%	0.98 23.9%
L	H	**0.47** **13.05%**	**0.46** **13.28%**	0.66 16.98%	0.57 15.8%

Table 2: Comparison of methods using RMSE and hit rates. There are 100 individuals. The C of both the T individual SVMs and the proposed method is 0.1. Bold indicates best or not significantly different than best at $p < 0.05$. A * indicates best or not significantly different than best at $p < 0.10$

dataset is close to being from a single task (despite this observation, we use this dataset for direct comparison with [4]) – this indicates that when the tasks are the same task, using the proposed multi-task learning method does not hurt as long as a small enough μ is chosen.

$\mu = 0.5$	34.30 ± 0.3	34.37 ± 0.4
$\mu = 1$	34.28 ± 0.4	34.37 ± 0.3
$\mu = 2$	34.26 ± 0.4	34.11 ± 0.4
$\mu = 10$	34.32 ± 0.3	29.71 ± 0.4
$\mu = 1000$	11.92 ± 0.5	4.83 ± 0.4
Bayesian	29.5 ± 0.4	29.5 ± 0.4

Table 3: School Data: first column shows the explained variance of the proposed method with $C = 0.1$ and second column with $C = 1$. The last row is the explained variance for the Bayesian task clustering method of [4].

4. DISCUSSION

We presented a new method for multi–task learning using a regularization approach. This is a natural extension of existing regularization based learning methods, such as SVMs, from single–task to multi–task learning. We tested the approach using both simulated and real data, and compared it with existing multi–task learning methods. The results a) show the strength of the proposed approach relative to other multi–task learning methods, and b) verify, in agreement with past experimental work [4, 11, 15], the advantage of multi–task learning relative to single task learning. The proposed method also reduces to standard single–task learning when we set the task coupling parameter μ appearing in the matrix–valued kernel (14) to be very large: hence there is no risk using this multi–task learning method even when the tasks are not related. It is a matter of choosing the appropriate parameter μ, which may be possible to do for example using some form of cross–validation or a validation set [25]. How to select μ is currently an open problem which we believe is also related to the general question of how to

model and measure the relatedness between tasks – e.g. how do we know a priori that tasks are related?

A number of extensions of the methods discussed above are possible. These concern for example the use of different loss functions (i.e. square loss) and, especially, the use of more general matrix–valued kernels. We discuss some of these below.

- The learning methods presented in this paper concern only SVMs with the Hinge loss function. These ideas can be applied as well to other loss functions and continuous multi–task learning problems. In general this problem is of the form

$$\min_{\mathbf{w}} \sum_{i=1}^{N} V(y_i, \langle \mathbf{w}, \Phi(\mathbf{x}_i, t_i)\rangle) + \lambda \langle \mathbf{w}, \mathbf{w}\rangle$$

  where V is the loss function and $\langle \cdot, \cdot \rangle$ the inner product in a feature space. For example, in the "school dataset" each function is a regression function and we used for V the ϵ–loss function of SVM regression [25]. In virtue of the representer theorem, see e.g. [20], the solution to this problem, $F^* = \langle \mathbf{w}^*, \Phi \rangle$ has the form in equation (16).

- We observed above that the regularizer in problem 2.2 (which is equivalent to problem 2.1) forces each model parameters to be close to the average of the model parameters. In many practical situations this assumption may be too restrictive. Instead, we may know that only tasks in some subgroups are similar to each others. If $\mathcal{I}_1, \ldots, \mathcal{I}_c \subseteq \{1, \ldots, T\}$ denotes the (possibly overlapping) index sets of these similar tasks, we modify the stabilizer in problem 2.2 to be

$$\rho_1 \sum_{t=1}^{T} \|\mathbf{w_t}\|^2 + \sum_{i=2}^{c+1} \rho_i \sum_{t \in \mathcal{I}_i} \left\| \mathbf{w}_t - \frac{1}{T_i} \sum_{s \in \mathcal{I}_i} \mathbf{w}_s \right\|^2$$

  where T_i is the cardinality of the set $\mathcal{I}_i$. We may even consider learning these index sets. For example, one way to do so would be to use the task clustering approach in [4].

- Equation (1) can be generalized in different ways by means of nonlinear kernels. One immediate possibility is to model f_t as

$$f_t = g + g_t$$

  where g is a function common to all tasks and g_t are individual functions for each task. If K_1 is the kernel used to model g and K_2 is the kernel used to model the g_t, the matrix–valued kernel to be used in problem 2.4 becomes

$$K_{st}(\mathbf{x}, \mathbf{z}) := \frac{1}{\mu} K_1(\mathbf{x}, \mathbf{z}) + \delta_{st} K_2(\mathbf{x}, \mathbf{z}), \quad s, t = 1, \ldots, T.$$

  For example one can model g by a low-degree polynomial and allow a higher degree polynomial kernel to model the g_t, for example $K_1(\mathbf{x}, \mathbf{z}) = \mathbf{x} \cdot \mathbf{z}$, $K_2(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$.

- Suppose each task/function uses different types of features, that is we have a collection of sets, $\mathcal{X}_t$, $t = 1, \ldots, T$, and for each set we wish to learn a target function $g_t : \mathcal{X}_t \to \mathbb{R}$. For example, we could have $\mathcal{X}_t = \mathbb{R}^{d_t}$, where d_t are some positive integers. This problem can be cast in the above framework by defining the input space

$$\mathcal{X} := \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \mathcal{X}_T$$

  and the vector–valued function $f : \mathcal{X} \to \mathbb{R}^n$ whose coordinates are given by $f_t(\mathbf{x}) = g_t(x_t)$, where $\mathbf{x} = (x_1, \ldots, x_T) \in \mathcal{X}$.

  For each $s, t \in \{1, \ldots, T\}$, define the functions $C_{st} : \mathcal{X}_s \times \mathcal{X}_t \to \mathbb{R}$ so that the matrix–valued kernel

$$K_{st}(\mathbf{x}, \mathbf{z}) = C_{st}(x_s, z_t), \quad s, t \in \{1, \ldots, T\}$$

  satisfies the properties in Proposition 1 of [20], where $\mathbf{x} = (x_1, \ldots, x_T)$ and $\mathbf{z} = (z_1, \ldots, z_T) \in \mathcal{X}$. In this case the function

$$G((\mathbf{x}, s), (\mathbf{z}, t)) = C_{st}(x_s, z_t)$$

  is a kernel (is it symmetric and positive definite) and can be used in problem 2.4 indeed.

We leave the exploration of matrix–valued kernels for different types of multi–task learning applications as part of future work. On the theoretical side another important problem will be to study generalization error bounds for the proposed methods. In particular, it may be possible to link the matrix–valued kernels to the notion of relatedness between tasks discussed in [8].

Acknowledgements

We wish to thank Sayan Mukherjee, Tomaso Poggio, and especially Charlie Micchelli for useful discussions.

5. REFERENCES

[1] G.M. Allenby and P.E. Rossi. Marketing Models of Consumer Heterogeneity. *Journal of Econometrics, 89, p. 57–78*, 1999.

[2] N. Arora G.M Allenby, and J. Ginter. A Hierarchical Bayes Model of Primary and Secondary Demand. Marketing Science, 17,1, p. 29–44, 1998

[3] N. Arora and J. Huber. Improving parameter estimates and model prediction by aggregate customization in choice experiments. Journal of Consumer Research, Vol. 28, September 2001.

[4] B. Bakker and T. Heskes. Task clustering and gating for Bayesian multi–task learning. Journal of Machine Learning Research, 4: 83–99, 2003.

[5] J. Baxter. A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling. *Machine Learning, 28, pp. 7–39*, 1997.

[6] J. Baxter. A Model for Inductive Bias Learning. *Journal of Artificial Intelligence Research, 12, p. 149–198*, 2000.

[7] S. Ben-David, J. Gehrke, and R. Schuller. A Theoretical Framework for Learning from a Pool of Disparate Data Sources. Proceedings of Knowledge Discovery and Datamining (KDD), 2002.

[8] S. Ben-David and R. Schuller. Exploiting Task Relatedness for Multiple Task Learning. Proceedings of Computational Learning Theory (COLT), 2003.

[9] L. Breiman and J.H Friedman. Predicting Multivariate Responses in Multiple Linear Regression. *Royal Statistical Society Series B*, 1998.

[10] P.J. Brown and J.V. Zidek. Adaptive Multivariate Ridge Regression. *The Annals of Statistics, Vol. 8, No. 1, p. 64–74*, 1980.

[11] R. Caruana. Multi–Task Learning. *Machine Learning, 28, p. 41–75*, 1997.

[12] T. Evgeniou, C. Boussios, and G. Zacharia. Generalized Robust Conjoint Estimation. INSEAD Working Paper, 2002.

[13] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[14] B. Heisele, T. Serre, M. Pontil, T. Vetter, and T. Poggio. Categorization by Learning and Combining Object Parts. In: Advances in Neural Information Processing Systems 14, Vancouver, Canada, Vol. 2, 1239–1245, 2002.

[15] T. Heskes. Empirical Bayes for learning to learn. Proceedings of ICML–2000, ed. Langley, P., pp. 367–374, 2000.

[16] M.I. Jordan and R.A. Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, 1993.

[17] J. Kim, G.M. Allenby, and P.E. Rossi. Modeling Consumer Demand for Variety. *Marketing Science, Vol. 21, No. 3, pp. 229–250*, Summer 2002.

[18] G.R.G. Lanckriet, T. De Bie, N. Cristianini, M.I. Jordan, and W.S. Noble. A framework for genomic data fusion and its application to membrane protein prediction. Technical Report CSD–03–1273, Division of Computer Science, University of California, Berkeley, 2003.

[19] O.L. Mangasarian. *Nonlinear Programming*. Classics in Applied Mathematics. SIAM, 1994.

[20] C.A. Micchelli and M. Pontil. On Learning Vector–Valued Functions. Research Note RN/03/08, Dept of Computer Science, UCL, July 2003.

[21] D.L. Silver and R.E Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science, 8, p. 277–294*, 1996.

[22] S. Thrun and L. Pratt. Learning to Learn. Kluwer Academic Publishers, November 1997.

[23] S. Thrun and J. O'Sullivan. Clustering Learning Tasks and the Selective Cross–Task Transfer of Knowledge. *In Learning To Learn, Editors S. Thrun and L.Y. Pratt, Kluwer Academic Publishers*, 1998.

[24] O. Toubia, D.I. Simester, J.R. Hauser, and E. Dahan. Fast Polyhedral Adaptive Conjoint Estimation. Working paper, MIT Sloan School of Management, 2001.

[25] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[26] G. Wahba. *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.