Journal of Machine Learning Research 6 (2005) 615–637 Submitted 2/05; Published 4/05

Learning Multiple Tasks with Kernel Methods

Theodoros Evgeniou THEODOROS.EVGENIOU@INSEAD.EDU
Technology Management
INSEAD
77300 Fontainebleau, France

Charles A. Micchelli CAM@MATH.ALBANY.EDU
Department of Mathematics and Statistics
State University of New York
The University at Albany
1400 Washington Avenue
Albany, NY 12222, USA

Massimiliano Pontil M.PONTIL@CS.UCL.AC.UK
Department of Computer Science
University College London
Gower Street, London WC1E, UK

Editor: John Shawe-Taylor

Abstract

We study the problem of learning many related tasks simultaneously using kernel methods and regularization. The standard single-task kernel methods, such as support vector machines and regularization networks, are extended to the case of multi-task learning. Our analysis shows that the problem of estimating many task functions with regularization can be cast as a single task learning problem if a family of multi-task kernel functions we define is used. These kernels model relations among the tasks and are derived from a novel form of regularizers. Specific kernels that can be used for multi-task learning are provided and experimentally tested on two real data sets. In agreement with past empirical work on multi-task learning, the experiments show that learning multiple related tasks simultaneously using the proposed approach can significantly outperform standard single-task learning particularly when there are many related tasks but few data per task.

Keywords: multi-task learning, kernels, vector-valued functions, regularization, learning algorithms

1. Introduction

Past empirical work has shown that, when there are multiple related learning tasks it is beneficial to learn them simultaneously instead of independently as typically done in practice (Bakker and Heskes, 2003; Caruana, 1997; Heskes, 2000; Thrun and Pratt, 1997). However, there has been insufficient research on the theory of multi-task learning and on developing multi-task learning methods. A key *goal* of this paper is to extend the single-task kernel learning methods which have been successfully used in recent years to multi-task learning. Our analysis establishes that the problem of estimating many task functions with regularization can be linked to a single task learning problem provided a family of multi-task kernel functions we define is used. For this purpose, we use kernels for vector-valued functions recently developed by Micchelli and Pontil (2005). We elaborate on these ideas within a practical context and present experiments of the proposed kernel-based multi-task learning methods on two real data sets.

Multi-task learning is important in a variety of practical situations. For example, in finance and economics forecasting predicting the value of many possibly related indicators simultaneously is often required (Greene, 2002); in marketing modeling the preferences of many individuals, for example with similar demographics, simultaneously is common practice (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998); in bioinformatics, we may want to study tumor prediction from multiple micro–array data sets or analyze data from mutliple related diseases.

It is therefore important to extend the existing kernel-based learning methods, such as SVM and RN, that have been widely used in recent years, to the case of multi-task learning. In this paper we shall demonstrate experimentally that the proposed multi-task kernel-based methods lead to significant performance gains.

The paper is organized as follows. In Section 2 we briefly review the standard framework for single-task learning using kernel methods. We then extend this framework to multi-task learning for the case of learning linear functions in Section 3. Within this framework we develop a general multi-task learning formulation, in the spirit of SVM and RN type methods, and propose some specific multi-task learning methods as special cases. We describe experiments comparing two of the proposed multi-task learning methods to their standard single-task counterparts in Section 4. Finally, in Section 5 we discuss extensions of the results of Section 3 to non-linear models for multi-task learning, summarize our findings, and suggest future research directions.

1.1 Past Related Work

The empirical evidence that multi-task learning can lead to significant performance improvement (Bakker and Heskes, 2003; Caruana, 1997; Heskes, 2000; Thrun and Pratt, 1997) suggests that this area of machine learning should receive more development. The simultaneous estimation of multiple statistical models was considered within the econometrics and statistics literature (Greene, 2002; Zellner, 1962; Srivastava and Dwivedi, 1971) prior to the interests in multi-task learning in the machine learning community.

Task relationships have been typically modeled through the assumption that the error terms (noise) for the regressions estimated simultaneously—often called “Seemingly Unrelated Regressions”—are correlated (Greene, 2002). Alternatively, extensions of regularization type methods, such as ridge regression, to the case of multi-task learning have also been proposed. For example, Brown and Zidek (1980) consider the case of regression and propose an extension of the standard ridge regression to multivariate ridge regression. Breiman and Friedman (1998) propose the curds&whey method, where the relations between the various tasks are modeled in a post–processing fashion.

The problem of multi-task learning has been considered within the statistical learning and machine learning communities under the name “learning to learn” (see Baxter, 1997; Caruana, 1997; Thrun and Pratt, 1997). An extension of the VC-dimension notion and of the basic generalization bounds of SLT for single-task learning (Vapnik, 1998) to the case of multi-task learning has been developed in (Baxter, 1997, 2000) and (Ben-David and Schuller, 2003). In (Baxter, 2000) the problem of bias learning is considered, where the goal is to choose an optimal hypothesis space from a family of hypothesis spaces. In (Baxter, 2000) 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 (Baxter, 1997) 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 (Ben-David and Schuller, 2003) 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 (Baxter, 2000)) in the case that the learning tasks are related in a particular way defined. More recent work considers learning multiple tasks in a semi-supervised setting (Ando and Zhang, 2004) and the problem of feature selection with SVM across the tasks (Jebara, 2004).

Finally, a number of approaches for learning multiple tasks 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 (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998) a hierarchical Bayes model is estimated. First, it is assumed a priori 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, (Bakker and Heskes, 2003; Heskes, 2000) suggest a similar hierarchical model. In (Bakker and Heskes, 2003) 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.

In this paper we will not follow a Bayesian or a statistical approach. Instead, our goal is to develop multi-task learning methods and theory as an extension of widely used kernel learning methods developed within SLT or Regularization Theory, such as SVM and RN. We show that using a particular type of kernels, the regularized multi-task learning method we propose is equivalent to a single-task learning one when such a multi-task kernel is used. The work here improves upon the ideas discussed in (Evgeniou and Pontil, 2004; Micchelli and Pontil, 2005b).

One of our aims is to show experimentally that the multi-task learning methods we develop here significantly improve upon their single-task counterpart, for example SVM. Therefore, to emphasize and clarify this point we only compare the standard (single-task) SVM with a proposed multi-task version of SVM. Our experiments show the benefits of multi-task learning and indicate that multi-task kernel learning methods are superior to their single-task counterpart. An exhaustive comparison of *any* single-task kernel methods with their multi-task version is beyond the scope of this work.

2. Background and Notation

In this section, we briefly review the basic setup for single-task learning using regularization in a reproducing kernel Hilbert space (RHKS) $\mathcal{H}_K$ with kernel K. For more detailed accounts (see Evgeniou, Pontil, and Poggio, 2000; Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002; Vapnik, 1998; Wahba, 1990) and references therein.

2.1 Single-Task Learning: A Brief Review

In the standard single-task learning setup we are given m examples $\{(x_i, y_i) : i \in \mathbb{N}_m\} \subset \mathcal{X} \times \mathcal{Y}$ (we use the notation $\mathbb{N}_m$ for the set $\{1, \ldots, m\}$) sampled *i.i.d.* from an unknown probability distribution P on $\mathcal{X} \times \mathcal{Y}$. The input space $\mathcal{X}$ is typically a subset of $\mathbb{R}^d$, the d dimensional Euclidean space, and the output space $\mathcal{Y}$ is a subset of $\mathbb{R}$. For example, in binary classification $\mathcal{Y}$ is chosen to be $\{-1, 1\}$.

The goal is to learn a function f with small expected error $E[L(y, f(x))]$, where the expectation is taken with respect to P and L is a prescribed loss function such as the square error $(y - f(x))^2$. To this end, a common approach within SLT and regularization theory is to learn f as the minimizer in $\mathcal{H}_K$ of the functional

$$\frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_j, f(x_j)) + \gamma \|f\|_K^2 \tag{1}$$

where $\|f\|_K^2$ is the norm of f in $\mathcal{H}_K$. When $\mathcal{H}_K$ consists of linear functions $f(x) = w'x$, with $w, x \in \mathbb{R}^d$, we minimize

$$\frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_j, w'x_j) + \gamma w'w \tag{2}$$

where all vectors are column vectors and we use the notation A' for the transpose of matrix A, and w is a $d \times 1$ matrix.

The positive constant γ is called the regularization parameter and controls the trade off between the error we make on the m examples (the training error) and the complexity (smoothness) of the solution as measured by the norm in the RKHS. Machines of this form have been motivated in the framework of statistical learning theory (Vapnik, 1998). Learning methods such as RN and SVM are particular cases of these machines for certain choices of the loss function L (Evgeniou, Pontil, and Poggio, 2000).

Under rather general conditions (Evgeniou, Pontil, and Poggio, 2000; Micchelli and Pontil, 2005b; Wahba, 1990) the solution of Equation (1) is of the form

$$f(x) = \sum_{j \in \mathbb{N}_m} c_j K(x_j, x) \tag{3}$$

where $\{c_j : j \in \mathbb{N}_m\}$ is a set of real parameters and K is a kernel such as an homogeneous polynomial kernel of degree r, $K(x,t) = (x't)^r$, $x, t \in \mathbb{R}^d$. The kernel K has the property that, for $x \in \mathcal{X}$, $K(x, \cdot) \in \mathcal{H}_K$ and, for $f \in \mathcal{H}_K$ $\langle f, K(x, \cdot)\rangle_K = f(x)$, where $\langle \cdot, \cdot \rangle_K$ is the inner product in $\mathcal{H}_K$ (Aronszajn, 1950). In particular, for $x, t \in \mathcal{X}$, $K(x,t) = \langle K(x, \cdot), K(t, \cdot)\rangle_K$ implying that the $m \times m$ matrix $(K(x_i, x_j) : i, j \in \mathbb{N}_m)$ is symmetric and positive semi-definite for *any* set of inputs $\{x_j : j \in \mathbb{N}_m\} \subseteq \mathcal{X}$.

The result in Equation (3) is known as the *representer theorem*. Although it is simple to prove, it is remarkable as it makes the variational problem (1) amenable for computations. In particular, if L is convex, the unique minimizer of functional (1) can be found by replacing f by the right hand side of Equation (3) in Equation (1) and then optimizing with respect to the parameters $\{c_j : j \in N_m\}$.

A popular way to define the space $\mathcal{H}_K$ is based on the notion of a *feature map* $\Phi : \mathcal{X} \to \mathcal{W}$, where $\mathcal{W}$ is a Hilbert space with inner product denoted by $\langle \cdot, \cdot \rangle_{\mathcal{W}}$. Such a feature map gives rise to the linear space of all functions $f : \mathcal{X} \to \mathbb{R}$ defined for $x \in \mathcal{X}$ and $w \in \mathcal{W}$ as $f(x) = \langle w, \Phi(x)\rangle_{\mathcal{W}}$ with norm $\langle w, w\rangle_{\mathcal{W}}$. It can be shown that this space is (modulo an isometry) the RKHS $\mathcal{H}_K$ with kernel K defined, for $x, t \in \mathcal{X}$, as $K(x,t) = \langle \Phi(x), \Phi(t)\rangle_{\mathcal{W}}$. Therefore, the regularization functional (1) becomes

$$\frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_j, \langle w, \Phi(x_j)\rangle_{\mathcal{W}}) + \gamma \langle w, w\rangle_{\mathcal{W}}. \tag{4}$$

Again, any minimizer of this functional has the form

$$w = \sum_{j \in \mathbb{N}_m} c_j \Phi(x_j) \tag{5}$$

which is consistent with Equation (3).

2.2 Multi-Task Learning: Notation

For multi-task learning we have n tasks and corresponding to the $\ell-$th task there are available m examples $\{(x_{i\ell}, y_{i\ell}) : i \in \mathbb{N}_m\}$ sampled from a distribution P_ℓ on $\mathcal{X}_\ell \times \mathcal{Y}_\ell$. Thus, the total data available is $\{(x_{i\ell}, y_{i\ell}) : i \in \mathbb{N}_m, \ell \in \mathbb{N}_n\}$. The goal it to learn all n functions $f_\ell : \mathcal{X}_\ell \to \mathcal{Y}_\ell$ from the available examples. In this paper we mainly discuss the case that the tasks have a common input space, that is $X_\ell = X$ for all ℓ and briefly comment on the more general case in Section 5.1.

There are various special cases of this setup which occur in practice. Typically, the input space $\mathcal{X}_\ell$ is independent of ℓ. Even more so, the input data $x_{i\ell}$ may be independent of ℓ for every sample i. This happens in marketing applications of preference modeling (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998) where the same choice panel questions are given to many individual consumers, each individual provides his/her own preferences, and we assume that there is some commonality among the preferences of the individuals. On the other hand, there are practical circumstances where the output data $y_{i\ell}$ is independent of ℓ. For example, this occurs in the problem of integrating information from heterogeneous databases (Ben-David, Gehrke, and Schuller, 2002).

In other cases one does not have either possibilities, that is, the spaces $\mathcal{X}_\ell \times \mathcal{Y}_\ell$ are different. 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 (Heisele et al., 2002). Each of these tasks can be learned using images of different size (or different representations).

We now turn to the extension of the theory and methods for single-task learning using the regularization based kernel methods briefly reviewed above to the case of multi-task learning. In the following section we will consider the case that functions f_ℓ are all linear functions and postpone the discussion of non-linear multi-task learning to Section 5.

3. A Framework for Multi-Task Learning: The Linear Case

Throughout this section we assume that $\mathcal{X}_\ell = \mathbb{R}^d$, $\mathcal{Y}_\ell = \mathbb{R}$ and that the functions f_ℓ are linear, that is, $f_\ell(x) = u_\ell' x$ with $u_\ell \in \mathbb{R}^d$. We propose to estimate the vector of parameters $u = (u_\ell : \ell \in \mathbb{N}_n) \in \mathbb{R}^{nd}$ as the minimizer of a regularization function

$$R(u) := \frac{1}{nm} \sum_{\ell \in \mathbb{N}_n} \sum_{j \in \mathbb{N}_m} L(y_{j\ell}, u_\ell' x_{j\ell}) + \gamma J(u) \tag{6}$$

where γ is a positive parameter, J is a homogeneous quadratic function of u, that is,

$$J(u) = u' E u \tag{7}$$

and E a $dn \times dn$ matrix which captures the relations between the tasks. From now on we assume that matrix E is symmetric and *positive definite*, unless otherwise stated. We briefly comment on the case that E is positive semidefinite below.

For a certain choice of J (or, equivalently, matrix E), the regularization function (6) learns the tasks independently using the regularization method (1). In particular, for $J(u) = \sum_{\ell \in \mathbb{N}_n} \|u_\ell\|^2$ the function (6) decouples, that is, $R(u) = \frac{1}{n} \sum_{\ell \in \mathbb{N}_n} r_\ell(u_\ell)$ where $r_\ell(u_\ell) = \frac{1}{m} \sum_{j \in \mathbb{N}_m} L(y_{j\ell}, u_\ell' x_{j\ell}) + \gamma \|u_\ell\|^2$, meaning that the task parameters are learned *independently*. On the other hand, if we choose $J(u) =$

$\sum_{\ell,q\in\mathbb{N}_n}\|u_\ell - u_q\|^2$, we can "force" the task parameters to be close to each other: task parameters u_ℓ are learned *jointly* by minimizing (6).

Note that function (6) depends on dn parameters whose number can be very large if the number of tasks n is large. Our analysis below establishes that the multi-task learning method (6) is equivalent to a single-task learning method as in (2) for an appropriate choice of a multi-task kernel in Equation (10) below. As we shall see, the input space of this kernel depends is the original $d-$dimensional space of the data plus an additional dimension which records the task the data belongs to. For this purpose, we take the feature space point of view and write all functions f_ℓ in terms of the *same* feature vector $w \in \mathbb{R}^p$ for some $p \in \mathbb{N}$, $p \geq dn$. That is, for each f_ℓ we write

$$f_\ell(x) = w'B_\ell x, \;\; x \in \mathbb{R}^d, \;\; \ell \in \mathbb{N}_n \tag{8}$$

or, equivalently,

$$u_\ell = B'_\ell w, \;\; \ell \in \mathbb{N}_n \tag{9}$$

for some $p \times d$ matrix B_ℓ yet to be specified. We also define the $p \times dn$ *feature matrix* $B := [B_\ell : \ell \in \mathbb{N}_n]$ formed by concatenating the n matrices B_ℓ, $\ell \in \mathbb{N}_n$.

Note that, since the vector u_ℓ in Equation (9) is arbitrary, to ensure that there exists a solution w to this equation it is necessary that the matrix B_ℓ is of full rank d for each $\ell \in \mathbb{N}_n$. Moreover, we assume that the feature matrix B is of full rank dn as well. If this is not the case, the functions f_ℓ are linearly related. For example, if we choose $B_\ell = B_0$ for every $\ell \in \mathbb{N}_n$, where B_0 is a prescribed $p \times d$ matrix, Equation (8) tells us that all tasks are the same task, that is, $f_1 = f_2 = \cdots = f_n$. In particular if $p = d$ and $B_0 = I_d$ the function (11) (see below) implements a single-task learning problem, as in Equation (2) with all the mn data from the n tasks as if they all come from the same task.

Said in other words, we view the vector-valued function $f = (f_\ell : \ell \in \mathbb{N}_n)$ as the real-valued function

$$(x,\ell) \mapsto w'B_\ell x$$

on the input space $\mathbb{R}^d \times \mathbb{N}_n$ whose squared norm is $w'w$. The Hilbert space of all such real-valued functions has the reproducing kernel given by the formula

$$K((x,\ell),(t,q)) = x'B'_\ell B_q t, \;\; x,t \in \mathbb{R}^d, \; \ell,q \in \mathbb{N}_n. \tag{10}$$

We call this kernel a *linear multi-task kernel* since it is bilinear in x and t for fixed ℓ and q.

Using this linear feature map representation, we wish to convert the regularization function (6) to a function of the form (2), namely,

$$S(w) := \frac{1}{nm}\sum_{\ell\in N_n}\sum_{j\in N_m} L(y_{j\ell}, w'B_\ell x_{j\ell}) + \gamma w'w, \;\; w \in \mathbb{R}^p. \tag{11}$$

This transformation relates matrix E defining the homogeneous quadratic function of u we used in (6), $J(u)$, and the feature matrix B. We describe this relationship in the proposition below.

Proposition 1 *If the feature matrix B is full rank and we define the matrix E in Equation (7) as to be $E = (B'B)^{-1}$ then we have that*

$$S(w) = R(B'w), \;\; w \in \mathbb{R}^d. \tag{12}$$

Conversely, if we choose a symmetric and positive definite matrix E in Equation (7) and T is a squared root of E then for the choice of $B = T'E^{-1}$ Equation (12) holds true.

PROOF. We first prove the first part of the proposition. Since Equation (9) requires that the feature vector w is *common* to all vectors u_ℓ and those are arbitrary, the feature matrix B must be of full rank dn and, so, the matrix E above is well defined. This matrix has the property that $BEB' = I_p$, this being the $p \times p$ identity matrix. Consequently, we have that

$$w'w = J(B'w), \;\; w \in \mathbb{R}^p \tag{13}$$

and Equation (12) follows.

On the other direction, we have to find a matrix B such that $BEB' = I_p$. To this end, we express E in the form

$$E = TT'$$

where T is a $dn \times p$ matrix, $p \geq dn$. This maybe done in various ways since E is positive definite. For example, with $p = dn$ we can find a $dn \times dn$ matrix T by using the eigenvalues and eigenvectors of E. With this representation of E we can choose our features to be

$$B = VT'E^{-1}$$

where V is an *arbitrary* $p \times p$ orthogonal matrix. This fact follows because $BEB' = I_p$. In particular, if we choose $V = I_p$ the result follows.

□

Note that this proposition requires that B is of full rank because E is positive definite. As an example, consider the case that B_ℓ is a $dn \times d$ matrix all of whose $d \times d$ blocks are zero except for the $\ell-$th block which is equal to I_d. This choice means that we are learning all tasks independently, that is, $J(u) = \sum_{\ell \in \mathbb{N}_n} \|u_\ell\|^2$ and proposition (1) confirms that $E = I_{dn}$.

We conjecture that if the matrix B is not full rank, the equivalence between function (11) and (6) stated in proposition 1 still holds true provided matrix E is given by the pseudoinverse of matrix $(B'B)$ *and* we minimize the latter function on the linear subspace $\mathcal{S}$ spanned by the eigenvectors of E which have a positive eigenvalue. For example, in the above case where $B_\ell = B_0$ for all $\ell \in \mathbb{N}_n$ we have that $\mathcal{S} = \{(u_\ell : \ell \in \mathbb{N}_n) : u_1 = u_2 = \cdots = u_n\}$. This observation would also extend to the circumstance where there are arbitrary linear relations amongst the task functions. Indeed, we can impose such linear relations on the features directly to achieve this relation amongst the task functions. We discuss a specific example of this set up in Section 3.1.3. However, we leave a complete analysis of the positive semidefinite case to a future occasion.

The main implication of proposition 1 is the equivalence between function (6) and (11) when E is positive definite. In particular, this proposition implies that when matrix B and E are linked as stated in the proposition, the unique minimizers w^* of (11) and u^* of (6) are related by the equations $u^* = B'w^*$.

Since functional (11) is like a single task regularization functional (2), by the representer theorem—see Equation (5)—its minimizer has the form

$$w^* = \sum_{j \in \mathbb{N}_m} \sum_{\ell \in \mathbb{N}_n} c_{j\ell} B_\ell x_{j\ell}.$$

This implies that the optimal task functions are

$$f_q^*(x) = \sum_{j \in \mathbb{N}_m} \sum_{\ell \in \mathbb{N}_n} c_{j\ell} K((x_{j\ell}, \ell), (x, q)), \;\; x \in \mathbb{R}^d, q \in \mathbb{N}_n \tag{14}$$

where the kernel is defined in Equation (10). Note that these equations hold for *any* choice of the matrices B_ℓ, $\ell \in \mathbb{N}_n$.

Having defined the kernel for (10), we can now use standard single-task learning methods to learn multiple tasks simultaneously (we only need to define the appropriate kernel for the input data (x,ℓ)). Specific choices of the loss function L in Equation (11) lead to different learning methods.

Example 1 In regularization networks (RN) we choose the square loss $L(y,z) = (y-z)^2$, $y,z \in \mathbb{R}$ (see, for example, Evgeniou, Pontil, and Poggio, 2000). In this case the parameters $c_{j\ell}$ in Equation (14) are obtained by solving the system of linear equations

$$\sum_{q\in\mathbb{N}_n} \sum_{j\in\mathbb{N}_m} K((x_{jq},q),(x_{i\ell},\ell))c_{jq} = y_{i\ell}, \quad i \in \mathbb{N}_m, \ell \in \mathbb{N}_n. \tag{15}$$

When the kernel K is defined by Equation (10) this is a form of multi-task ridge regression.

□

Example 2 In support vector machines (SVM) for binary classification (Vapnik, 1998) we choose the hinge loss, namely $L(y,z) = (1-yz)_+$ where $(x)_+ = \max(0,x)$ and $y \in \{-1,1\}$. In this case, the minimization of function (11) can be rewritten in the usual form

Problem 3.1

$$\min\left\{\sum_{\ell\in\mathbb{N}_n}\sum_{i\in\mathbb{N}_m}\xi_{i\ell} + \gamma\|w\|^2\right\} \tag{16}$$

subject, for all $i \in \mathbb{N}_m$ and $\ell \in \mathbb{N}_n$, to the constraints that

$$\begin{aligned} y_{i\ell} w' B_\ell x_{i\ell} &\geq 1 - \xi_{i\ell} \\ \xi_{i\ell} &\geq 0. \end{aligned} \tag{17}$$

Following the derivation in Vapnik (1998) the dual of this problem is given by

Problem 3.2

$$\max_{c_{i\in\ell}}\left\{\sum_{\ell\in\mathbb{N}_n}\sum_{i\in\mathbb{N}_m} c_{i\ell} - \frac{1}{2}\sum_{\ell,q\in\mathbb{N}_n}\sum_{i,j\in\mathbb{N}_m} c_{i\ell}y_{i\ell}c_{jq}y_{jq}K((x_{i\ell},\ell),(x_{jq},q))\right\} \tag{18}$$

subject, for all $i \in \mathbb{N}_n$ and $\ell \in \mathbb{N}_n$, to the constrains that

$$0 \leq c_{i\ell} \leq \frac{1}{2\gamma}.$$

□

We now study particular examples some of which we also test experimentally in Section 5.

3.1 Examples of Linear Multi-Task Kernels

We discuss some examples of the above framework which are valuable for applications. These cases arise from different choices of matrices B_ℓ that we used above to model task relatedness or, equivalently, by directly choosing the function J in Equation (6).

Notice that a particular case of the regularizer J in Equation (7) is given by

$$J(u) = \sum_{\ell,q\in\mathbb{N}_n} u'_\ell u_q G_{\ell q} \tag{19}$$

where $G = (G_{\ell,q} : \ell, q \in \mathbb{N}_n)$ is a positive definite matrix. Proposition (1) implies that the kernel has the form

$$K((x,\ell),(t,q)) = x't\, G^{-1}_{\ell q}. \tag{20}$$

Indeed, J can be written as (u, Eu) where E is the $n \times n$ block matrix whose ℓ, q block is the $d \times d$ matrix $G_{\ell q} I_d$ and the result follows. The examples we discuss are with kernels of the form (20).

3.1.1 A USEFUL EXAMPLE

In our first example we choose B_ℓ to be the $(n+1)d \times d$ matrix whose $d \times d$ blocks are all zero except for the $1-$st and $(\ell+1)-$th block which are equal to $\sqrt{1-\lambda} I_d$ and $\sqrt{\lambda n} I_d$ respectively, where $\lambda \in (0,1)$ and I_d is the $d-$dimensional identity matrix. That is,

$$B'_\ell = [\sqrt{1-\lambda} I_d, \underbrace{0,\ldots,0}_{\ell-1}, \sqrt{\lambda n} I_d, \underbrace{0,\ldots,0}_{n-\ell}] \tag{21}$$

where here 0 stands for the $d \times d$ matrix all of whose entries are zero. Using Equation (10) the kernel is given by

$$K((x,\ell),(t,q)) = (1 - \lambda + \lambda n \delta_{\ell q}) x't, \;\; \ell, q \in \mathbb{N}_n, \; x, t \in \mathbb{R}^n. \tag{22}$$

A direct computation shows that

$$E_{\ell q} = ((B'B)^{-1})_{\ell q} = \frac{1}{n}\left(\frac{\delta_{\ell q}}{\lambda} - \frac{1-\lambda}{n\lambda}\right) I_d$$

where $E_{\ell q}$ is the $(\ell, q)-$th $d \times d$ block of matrix E. By proposition 1 we have that

$$J(u) = \frac{1}{n}\left(\sum_{\ell\in\mathbb{N}_n} \|u_\ell\|^2 + \frac{1-\lambda}{\lambda} \sum_{\ell\in\mathbb{N}_n} \|u_\ell - \frac{1}{n}\sum_{q\in\mathbb{N}_n} u_q\|^2\right). \tag{23}$$

This regularizer enforces a trade–off between a desirable small size for per–task parameters and closeness of each of these parameters to their average. This trade-off is controlled by the coupling parameter λ. If λ is small the tasks parameters are *related* (closed to their average) whereas if $\lambda = 1$ the task are learned independently.

The model of minimizing (11) with the regularizer (24) was proposed by Evgeniou and Pontil (2004) in the context of support vector machines (SVM's). In this case the above regularizer trades off large margin of each per–task SVM with closeness of each SVM to the average SVM. In Section 4 we will present numerical experiments showing the good performance of this multi–task SVM compared to both independent per–task SVM's (that is, $\lambda = 1$ in Equation (22)) and previous multi–task learning methods.

We note in passing that an alternate form for the function J is

$$J(u) = \min \left\{ \frac{1}{\lambda n} \sum_{\ell \in \mathbb{N}_n} \|u_\ell - u_0\|^2 + \frac{1}{1-\lambda} \|u_0\|^2 : u_0 \in \mathbb{R}^d \right\}. \tag{24}$$

It was this formula which originated our interest in multi-task learning in the context of regularization, see (Evgeniou and Pontil, 2004) for a discussion. Moreover, if we replace the identity matrix I_d in Equation (21) by a (any) $d \times d$ matrix A we obtain the kernel

$$K((x,\ell),(t,q)) = (1 - \lambda + \lambda n \delta_{\ell q}) x' Q t, \;\; \ell, q \in \mathbb{N}_n, \; x, t \in \mathbb{R}^n \tag{25}$$

where $Q = A'A$. In this case the norm in Equation (23) and (24) is replaced by $\|\cdot\|_{Q^{-1}}$.

3.1.2 TASK CLUSTERING REGULARIZATION

The regularizer in Equation (24) implements the idea that the task parameters u_ℓ are all related to each other in the sense that each u_ℓ is close to an "average parameter" u_0. Our second example extends this idea to different groups of tasks, that is, we assume that the task parameters can be put together in different groups so that the parameters in the $k-$th group are all close to an average parameter u_{0k}. More precisely, we consider the regularizer

$$J(u) = \min \left\{ \sum_{k \in \mathbb{N}_c} \left(\sum_{\ell \in \mathbb{N}_n} \rho_k^{(\ell)} \|u_\ell - u_{0k}\|^2 + \rho \|u_{0k}\|^2 \right) : u_{0k} \in \mathbb{R}^d, k \in \mathbb{N}_c \right\} \tag{26}$$

where $\rho_k^{(\ell)} \geq 0$, $\rho > 0$, and c is the number of clusters. Our previous example corresponds to $c = 1$, $\rho = \frac{1}{1-\lambda}$ and $\rho_1^{(\ell)} = \frac{1}{\lambda n}$. A direct computation shows that

$$J(u) = \sum_{\ell, q \in \mathbb{N}_n} u_\ell' u_q G_{\ell q}$$

where the elements of the matrix $G = (G_{\ell q} : \ell, q \in \mathbb{N}_n)$ are given by

$$G_{\ell q} = \sum_{k \in \mathbb{N}_c} \left(\rho_k^{(\ell)} \delta_{\ell q} - \frac{\rho_k^{(\ell)} \rho_k^{(q)}}{\rho + \sum_{r \in \mathbb{N}_n} \rho_k^{(r)}} \right).$$

If $\rho_k^{(\ell)}$ has the property that given any ℓ there is a cluster k such that $\rho_k^{(\ell)} > 0$ then G is positive definite. Then J is positive definite and by Equation (20) the kernel is given by $K((x,\ell),(t,q)) = G_{\ell q}^{-1} x' t$. In particular, if $\rho_h^{(\ell)} = \delta_{hk(\ell)}$ with $k(\ell)$ the cluster task ℓ belongs to, matrix G is invertible and takes the simple form

$$G_{\ell q}^{-1} = \delta_{\ell q} + \frac{1}{\rho} \theta_{\ell q} \tag{27}$$

where $\theta_{\ell q} = 1$ if tasks ℓ and q belong to the same cluster and zero otherwise. In particular, if $c = 1$ and we set $\rho = \frac{1-\lambda}{\lambda n}$ the kernel $K((x,\ell),(t,q)) = (\delta_{\ell q} + \frac{1}{\rho}) x' t$ is the same (modulo a constant) as the kernel in Equation (22).

3.1.3 GRAPH REGULARIZATION

In our third example we choose an $n \times n$ symmetric matrix A all of whose entries are in the unit interval, and consider the regularizer

$$J(u) := \frac{1}{2} \sum_{\ell,q \in \mathbb{N}_n} \|u_\ell - u_q\|^2 A_{\ell q} = \sum_{\ell,q \in \mathbb{N}_n} u'_\ell u_q L_{\ell q} \tag{28}$$

where $L = D - A$ with $D_{\ell q} = \delta_{\ell q} \sum_{h \in \mathbb{N}_n} A_{\ell h}$. The matrix A could be the weight matrix of a graph with n vertices and L the graph Laplacian (Chung, 1997). The equation $A_{\ell q} = 0$ means that tasks ℓ and q are not related, whereas $A_{\ell q} = 1$ means strong relation.

The quadratic function (28) is only positive semidefinite since $J(u) = 0$ whenever all the components of u_ℓ are independent of ℓ. To identify those vectors u for which $J(u) = 0$ we express the Laplacian L in terms of its eigenvalues and eigenvectors. Thus, we have that

$$L_{\ell q} = \sum_{k \in \mathbb{N}_n} \sigma_k v_{k\ell} v_{kq} \tag{29}$$

where the matrix $V = (v_{k\ell})$ is orthogonal, $\sigma_1 = \cdots = \sigma_r < \sigma_{r+1} \leq \cdots \leq \sigma_n$ are the eigenvalues of L and $r \geq 1$ is the multiplicity of the zero eigenvalue. The number r can be expressed in terms of the number of connected components of the graph, see, for example, (Chung, 1997). Substituting the expression (29) for L in the right hand side of (28) we obtain that

$$J(u) = \sum_{k \in \mathbb{N}_n} \sigma_k \left\| \sum_{\ell \in \mathbb{N}_n} u_\ell v_{k\ell} \right\|^2 .$$

Therefore, we conclude that J is positive definite on the space

$$\mathcal{S} = \left\{ u : u \in \mathbb{R}^{dn}, \sum_{\ell \in \mathbb{N}_n} u_\ell v_{k\ell} = 0, k \in \mathbb{N}_r \right\}.$$

Clearly, the dimension of $\mathcal{S}$ is $d(n-r)$. $\mathcal{S}$ gives us a Hilbert space of vector-valued linear functions

$$\mathcal{H} = \left\{ f_u(x) = (u'_\ell x : \ell \in \mathbb{N}_n) : u \in \mathcal{S} \right\}$$

and the reproducing kernel of $\mathcal{H}$ is given by

$$K((x,\ell),(t,q)) = L^+_{\ell q} x' t. \tag{30}$$

where L^+ is the pseudoinverse of L, that is,

$$L^+_{\ell q} = \sum_{k=r+1}^{n} \sigma_k^{-1} v_{k\ell} v_{kq}.$$

The verification of these facts is straightforward and we do not elaborate on the details. We can use this observation to assert that on the space $\mathcal{S}$ the regularization function (6) corresponding to the Laplacian has a *unique* minimum and it is given in the form of a representer theorem for kernel (30).

4. Experiments

As discussed in the introduction, we conducted experiments to compare the (standard) single-task version of a kernel machine, in this case SVM, to a multi-task version developed above. We tested two multi-task versions of SVM: a) we considered the simple case that the matrix Q in Equation (25) is the identity matrix, that is, we use the multi-task kernel (22), and b) we estimate the matrix Q in (25) by running PCA on the previously learned task parameters. Specifically, we first initialize Q to be the identity matrix. We then iterate as follows:

1. We estimate parameters u_ℓ using (25) and the current estimate of matrix Q (which, for the first iteration is the identity matrix).

2. We run PCA on these estimates, and select only the top principal components (corresponding to the largest eigenvalues of the empirical correlation matrix of the estimated u_ℓ). In particular, we only select the eigenvectors so that the sum of the corresponding eigenvalues (total "energy" kept) is at least 90% of the sum of all the eigenvalues (not using the remaining eigenvalues once we reach this 90% threshold). We then use the covariance of these principal components as our estimate of matrix Q in (25) for the next iteration.

We can repeat steps (1) and (2) until all eigenvalues are needed to reach the 90% energy threshold – typically in 4-5 iterations for the experiments below. We can then pick the estimated u_ℓ after the iteration that lead to the best validation error. We emphasize, that this is simply a heuristic. We do not have a theoretical justification for this heuristic. Developing a theory as well as other methods for estimating matrix Q is an open question. Notice that instead of using PCA we could directly use for matrix Q simply the covariance of the estimated u_ℓ of the previous iteration. However doing so is sensitive to estimation errors of u_ℓ and leads (as we also observed experimentally – we don't show the results here for simplicity) to poorer performance.

One of the key questions we considered is: *how does multi-task learning perform relative to single-task as the number of data per task and as the number of tasks change?* This question is also motivated by a typical situation in practice, where it may be easy to have data from many related tasks, but it may be difficult to have many data per task. This could often be for example the case in analyzing customer data for marketing, where we may have data about many customers (tens of thousands) but only a few samples per customer (only tens) (Allenby and Rossi, 1999; Arora, Allenby, and Ginter, 1998). It can also be the case for biological data, where we may have data about many related diseases (for example, types of cancer), but only a few samples per disease (Rifkin et al., 2003). As noted by other researchers in (Baxter, 1997, 2000; Ben-David, Gehrke, and Schuller, 2002; Ben-David and Schuller, 2003), one should expect that multi-task learning helps more, relative to single task, when we have many tasks but only few data per task – while when we have many data per task then single-task learning may be as good.

We performed experiments with two real data sets. One was on customer choice data, and the other was on school exams used by (Bakker and Heskes, 2003; Heskes, 2000) which we use here also for comparison with (Bakker and Heskes, 2003; Heskes, 2000). We discuss these experiments next.

4.1 Customer Data Experiments

We tested the proposed methods using a real data set capturing choices among products made by many individuals.[1] The goal is to estimate a function for each individual modeling the preferences of the individual based on the choices he/she has made. This function is used in practice to predict what product each individual will choose among future choices. We modeled this problem as a classification one along the lines of (Evgeniou, Boussios, and Zacharia, 2002). Therefore, the goal is to estimate a classification function for each individual.

We have data from 200 individuals, and for each individual we have 120 data points. The data are three dimensional (the products were described using three attributes, such as color, price, size, etc.) each feature taking only discrete values (for example, the color can be only blue, or black, or red, etc.). To handle the discrete valued attributes, we transformed them into binary ones, having eventually 20-dimensional binary data. We consider each individual as a different "task". Therefore we have 200 classification tasks and 120 20-dimensional data points for each task – for a total of 24000 data points.

We consider a linear SVM classification for each task – trials with non-linear (polynomial of degree 2 and 3) SVM did not improve performance for this data set. To test how multi-task compares to single task as the number of data per task and/or the number of tasks changes, we ran experiments with varying numbers of data per task and number of tasks. In particular, we considered 50, 100, and 200 tasks, splitting the 200 tasks into 4 groups of 50 or 2 groups of 100 (or one group of 200), and then taking the average performance among the 4 groups, the 2 groups (and the 1 group). For each task we split the 120 points into 20, 30, 60, 90 training points, and 100, 90, 60, 30 test points respectively.

Given the limited number of data per task, we chose the regularization parameter γ for the single-task SVM among only a few values (0.1, 1, 10) using the actual test error.[2] On the other hand, the multi-task learning regularization parameter γ and parameter λ in (22) were chosen using a validation set consisting of one (training) data point per task which we then included back to the training data for the final training after the parameter selection. The parameters λ and γ used when we estimated matrix Q through PCA were the same as when we used the identity matrix as Q. We note that one of the advantages of multi-task learning is that, since the data are typically from many tasks, parameters such as regularization parameter γ can be practically chosen using only a few, proportionally to all the data available, validation data without practically "losing" many data for parameter selection – which may be a further important practical reason for multi-task learning. Parameter λ was chosen among values (0, 0.2, 0.4, 0.6, 0.8) – value 1 corresponding to training one SVM per task. Below we also record the results indicating how the test performance is affected by parameter λ.

We display all the results in Table 4.1. Notice that the performance of the single-task SVM does not change as the number of tasks increases – as expected. We also note that when we use one SVM for all the tasks—treating the data as if they come from the same task—we get a very poor performance: between 38 and 42 percent test error for the (data $\times$ tasks) cases considered.

From these results we draw the following conclusions:

1. The data are proprietary were provided to the authors by Research International Inc. and are available upon request.
2. This lead to some overfitting of the single task SVM, however it only gave our competitor an advantage over our approach.

Tasks	Data	One SVM	Indiv SVM	Identity	PCA
50	20	41.97	29.86	**28.72**	**29.16**
100	20	41.41	29.86	**28.30**	29.26
200	20	40.08	29.86	**27.79**	28.53
50	30	40.73	26.84	**25.53**	**25.65**
100	30	40.66	26.84	**25.25**	**24.79**
200	30	39.43	26.84	25.16	**24.13**
50	60	40.33	22.84	22.06	**21.08**
100	60	40.02	22.84	22.27	**20.79**
200	60	39.74	22.84	21.86	**20.00**
50	90	38.51	19.84	19.68	**18.45**
100	90	38.97	19.84	19.34	**18.08**
200	90	38.77	19.84	19.27	**17.53**

Table 1: Comparison of Methods as the number of data per task and the number of tasks changes. “One SVM” stands for training one SVM with all the data from all the task, “Indiv SVM” stands for training for each task independently, “Identity” stands for the multi-task SVM with the identity matrix, and “PCA” is the multi-task SVM using the PCA approach. Misclassification errors are reported. Best performance(s) at the 5% significance level is in bold.

- When there are few data per task (20, 30, or 60), both multi-task SVMs significantly outperform the single-task SVM.
- As the number of tasks increases the advantage of multi-task learning increases – for example for 20 data per task, the improvement in performance relative to single-task SVM is 1.14, 1.56, and 2.07 percent for the 50, 100, and 200 tasks respectively.
- When we have many data per task (90), the simple multi-task SVM does not provide any advantage relative to the single-task SVM. However, the PCA based multi-task SVM significantly outperforms the other two methods.
- When there are few data per task, the simple multi-task SVM performs better than the PCA multi-task SVM. It may be that in this case the PCA multi-task SVM overfits the data.

The last two observations indicate that it is important to have a good estimate of matrix Q in (25) for the multi-task learning method that uses matrix Q. Achieving this is currently an open questions that can be approached, for example, using convex optimization techniques, see, for example, (Lanckriet et al., 2004; Micchelli and Pontil, 2005b)

To explore the second point further, we show in Figure 1 the change in performance for the identity matrix based multi-task SVM relative to the single-task SVM in the case of 20 data per task. We use $\lambda = 0.6$ as before. We notice the following:

- When there are only a few tasks (for example, less than 20 in this case), multi-task can hurt the performance relative to single-task. Notice that this depends on the parameter λ used.

For example, setting λ close to 1 leads to using a single-task SVM. Hence our experimental findings indicate that *for few tasks one should use either a single-task SVM or a multi-task one with parameter λ selected near 1*.

- As the number of tasks increases, performance improves – surpassing the performance of the single-task SVM after 20 tasks in this case.

As discussed in (Baxter, 1997, 2000; Ben-David, Gehrke, and Schuller, 2002; Ben-David and Schuller, 2003), an important theoretical question is to study the effects of adding additional tasks on the generalization performance (Ben-David, Gehrke, and Schuller, 2002; Ben-David and Schuller, 2003). What our experiments show is that, for few tasks it may be inappropriate to follow a multi-task approach if a small λ is used, but as the number of tasks increases performance relative to single-task learning improves. Therefore one should choose parameter λ depending on the number of tasks, much like one should choose regularization parameter γ depending on the number of data.

We tested the effects of parameter λ in Equation (22) on the performance of the proposed approach. In Figure 2 we plot the test error for the simple multi-task learning method using the identity matrix (kernel (22)) for the case of 20 data per task when there are 200 tasks (third row in Table 4.1), or 10 tasks (for which single-task SVM outperforms multi-task SVM for $\lambda = 0.6$ as shown in Figure 1). Parameter λ varies from 0 (one SVM for all tasks) to 1 (one SVM per task). Notice that for the 200 tasks the error drops and then increases, having a flat minimum between $\lambda = 0.4$ and 0.6. Moreover, for any λ between 0.2 and 1 we get a better performance than the single-task SVM. The same behavior holds for the 10 tasks, except that now the space of λ's for which the multi-task approach outperforms the single-task one is smaller – only for λ between 0.7 and 1. Hence, *for a few tasks multi-task learning can still help if a large enough λ is used*. However, as we noted above, it is an open question as to how to choose parameter λ in practice – other than using a validation set.

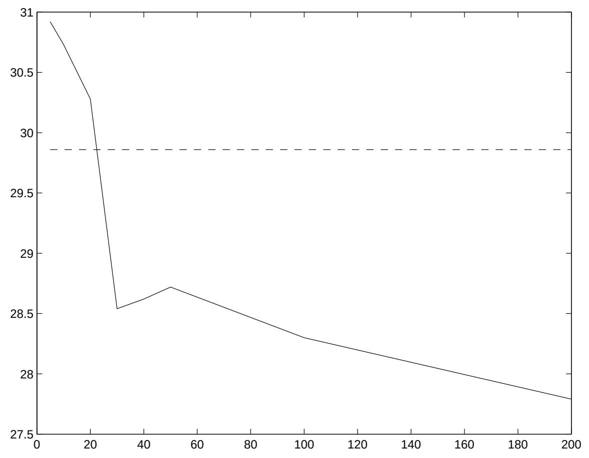

Figure 1: The horizontal axis is the number of tasks used. The vertical axis is the total test misclassification error among the tasks. There are 20 training points per task. We also show the performance of a single-task SVM (dashed line) which, of course, is not changing as the number of tasks increases.

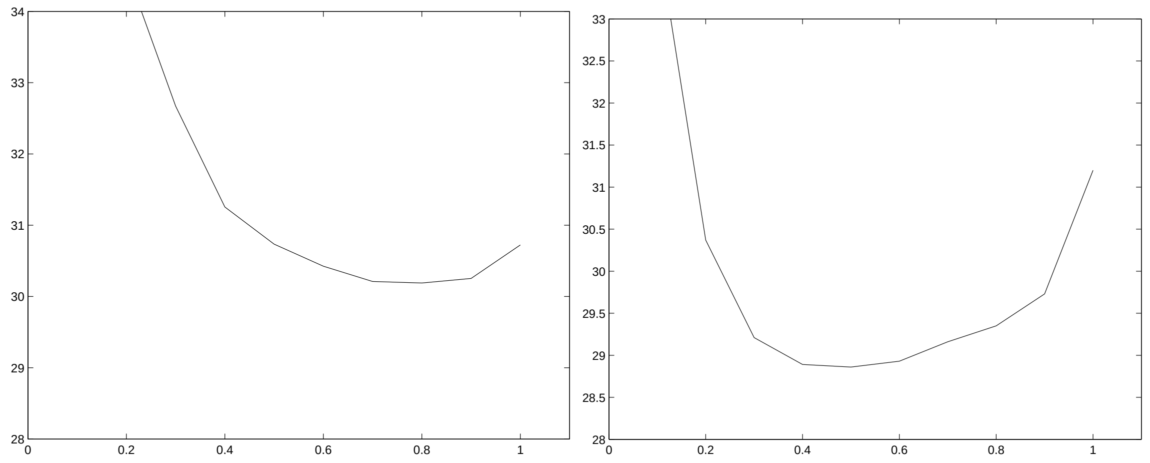

Figure 2: The horizontal axis is the parameter λ for the simple multi-task method with the identity matrix kernel (22). The vertical axis is the total test misclassification error among the tasks. There are 200 tasks with 20 training points and 100 test points per task. Left is for 10 tasks, and right is for 200 tasks.

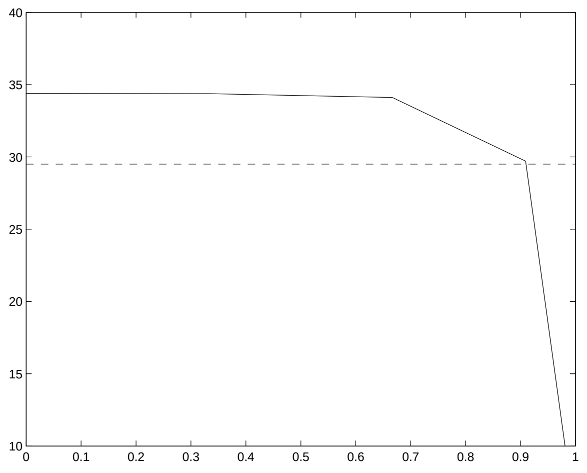

Figure 3: Performance on the school data. The horizontal axis is the parameter λ for the simple multi-task method with the identity matrix while the vertical is the explained variance (percentage) on the test data. The solid line is the performance of the proposed approach while the dashed line is the best performance reported in (Bakker and Heskes, 2003).

4.2 School Data Experiment

We also tested the proposed approach using the "school data" from the Inner London Education Authority available at *multilevel.ioe.ac.uk/intro/datasets.html*. This experiment is also discussed in (Evgeniou and Pontil, 2004) where some of the ideas of this paper were first presented. We selected this data set so that we can also compare our method directly with the work of Bakker and Heskes (2003) where a number of multi-task learning methods are applied to this data set. 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 ran regression using the SVM ε–loss function (Vapnik, 1998) for the multi–task learning method proposed. We considered each school to be "one task". Therefore, we had 139 tasks in total. 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 explained variance of the test data as a measure in order to have a direct comparison with (Bakker and Heskes, 2003) where this error measure is used. The explained variance is defined in (Bakker and Heskes, 2003) 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 shown in Figure 3. We set regularization parameter γ to be 1 and used a linear kernel for simplicity. We used the simple multi-task learning method proposed with the identity matrix. We let the parameter λ vary to see the effects. For comparison we also report on the performance of the task clustering method described in (Bakker and Heskes, 2003) – the dashed line in the figure.

The results show again the advantage of learning all tasks (for all schools) simultaneously instead of learning them one by one. Indeed, learning each task separately in this case hurts performance a lot. Moreover, even the simple identity matrix based approach significantly outperforms the Bayesian method of (Bakker and Heskes, 2003), which in turn in better than other methods as compared in (Bakker and Heskes, 2003). Note, however, that for this data set one SVM for all tasks performs the best, which is also similar to using a small enough λ (any λ between 0 and 0.7 in this case). Hence, it appears that the particular data set may come from a single task (despite this observation, we use this data set for direct comparison with (Bakker and Heskes, 2003)). This result also 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 λ is used. Notice that for this data set the performance does not change significantly for λ between 0 and 0.7, which shows that, as for the customer data above, the proposed method is not very sensitive to λ. A theoretical study of the sensitivity of our approach to the choice of the parameter λ is an open research direction which may also lead to a better understanding of the effects of increasing the number of tasks on the generalization performance as discussed in (Baxter, 1997, 2000; Ben-David and Schuller, 2003).

5. Discussion and Conclusions

In this final section we outline the extensions of the ideas presented above to non-linear functions, discuss some open problems on multi-tasks learning and draw our conclusions.

5.1 Nonlinear Multi-Task Kernels

We discuss a non-linear extension of the multi-task learning methods presented above. This gives us an opportunity to provide a wide variety of multi-task kernels which may be useful for applications. Our presentation builds upon earlier work on learning vector–valued functions (Micchelli and Pontil, 2005) which developed the theory of RKHS of functions whose range is a Hilbert space.

As in the linear case we view the vector-valued function $f = (f_\ell : \ell \in \mathbb{N}_n)$ as a real-valued function on the input space $\mathcal{X} \times \mathbb{N}_n$. We express f in terms of the feature maps $\Phi_\ell : \mathcal{X} \to \mathcal{W}$, $\ell \in \mathbb{N}_n$ where $\mathcal{W}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$. That is, we have that

$$f_\ell(x) = \langle w, \Phi_\ell(x) \rangle, \;\; x \in \mathcal{X}, \;\; \ell \in \mathbb{N}_n.$$

The vector w is computed by minimizing the single-task functional

$$S(w) := \frac{1}{nm} \sum_{\ell \in N_n} \sum_{j \in N_m} L(y_{j\ell}, \langle w, \Phi_\ell(x_{j\ell}) \rangle) + \gamma \langle w, w \rangle, \;\; w \in \mathcal{W}. \tag{31}$$

By the representer theorem, the minimizer of functional S has the form in Equation (14) where the multi-task kernel is given by the formula

$$K((x,\ell),(t,q)) = \langle \Phi_\ell(x), \Phi_q(t) \rangle \;\; x,t \in \mathcal{X}, \; \ell, q \in \mathbb{N}_n. \tag{32}$$

In Section 3 we have discussed this approach in the case that $\mathcal{W}$ is a finite dimensional Euclidean space and Φ_ℓ the linear map $\Phi_\ell(x) = B_\ell x$, thereby obtaining the linear multi-task kernel (10). In order to generalize this case it is useful to recall a result of Schur which states that the elementwise product of two positive semidefinite matrices is also positive semidefinite, (Aronszajn, 1950, p. 358). This implies that the elementwise product of two kernels is a kernel. Consequently, we conclude that, for any $r \in \mathbb{N}$,

$$K((x,\ell),(t,q)) = (x' B'_\ell B_q t)^r \tag{33}$$

is a polynomial multi-task kernel.

More generally we have the following lemma.

Lemma 2 *If G is a kernel on $\mathcal{T} \times \mathcal{T}$ and, for every $\ell \in \mathbb{N}_n$, there are prescribed mappings $z_\ell : \mathcal{X} \to \mathcal{T}$ such that*

$$K((x,\ell),(t,q)) = G(z_\ell(x), z_q(t)), \quad x,t \in \mathcal{X}, \; \ell, q \in \mathbb{N}_n \tag{34}$$

then K is a multi-task kernel.

PROOF. We note that for every $\{c_{i\ell} : i \in \mathbb{N}_m, \ell \in \mathbb{N}_n\} \subset \mathbb{R}$ and $\{x_{i\ell} : i \in \mathbb{N}_m, \ell \in \mathbb{N}_n\} \subset \mathcal{X}$ we have

$$\sum_{i,j \in \mathbb{N}_m} \sum_{\ell, q \in \mathbb{N}_n} c_{i\ell} c_{jq} G(z_\ell(x_{i\ell}), z_q(x_{jq})) = \sum_{i,\ell} \sum_{jq} c_{i\ell} c_{jq} G(\tilde{z}_{i\ell}, \tilde{z}_{jq}) \geq 0$$

where we have defined $\tilde{z}_{i\ell} = z_\ell(x_{i\ell})$ and the last inequality follows by the hypothesis that G is a kernel. □

For the special case that $\mathcal{T} = \mathbb{R}^p$, $z_\ell(x) = B_\ell x$ with B_ℓ a $p \times d$ matrix, $\ell \in \mathbb{N}_n$, and $G : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is the homogeneous polynomial kernel, $G(t,s) = (t's)^r$, the lemma confirms that the function (33) is a multi-task kernel. Similarly, when G is chosen to be a Gaussian kernel, we conclude that

$$K((x,\ell),(t,q)) = \exp(-\beta \|B_\ell x - B_q t\|^2)$$

is a multi-task kernel for every $\beta > 0$.

Lemma 2 also allows us to generalize multi-task learning to the case that each task function f_ℓ has a *different* input domain $\mathcal{X}_\ell$, a situation which is important in applications, see, for example, (Ben-David, Gehrke, and Schuller, 2002) for a discussion. To this end, we specify sets $\mathcal{X}_\ell$, $\ell \in \mathbb{N}_n$, functions $g_\ell : \mathcal{X}_\ell \to \mathbb{R}$, and note that multi–task learning can be placed in the above framework by defining the input space

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

We are interested in the functions $f_\ell(x) = g_\ell(P_\ell x)$, where $x = (x_1, \ldots, x_n)$ and $P_\ell : \mathcal{X} \to \mathcal{X}_\ell$ is defined, for every $x \in \mathcal{X}$ by $P_\ell(x) = x_\ell$. Let G be a kernel on $\mathcal{T} \times \mathcal{T}$ and choose $z_\ell(\cdot) = \phi_\ell(P_\ell(\cdot))$ where $\phi_\ell : \mathcal{X}_\ell \to \mathcal{T}$ are some prescribed functions. Then by lemma 2 the kernel defined by Equation (34) can be used to represent the functions g_ℓ. In particular, in the case of linear functions, we choose $\mathcal{X}_\ell = \mathbb{R}^{d_\ell}$, where $d_\ell \in \mathbb{N}$, $\mathcal{T} = \mathbb{R}^p$, $p \in \mathbb{N}$, $G(s,t) = s't$ and $z_\ell = D_\ell P_\ell$ where D_ℓ is a $p \times d_\ell$ matrix. In this case, the multi-task kernel is given by

$$K((x,\ell),(t,q)) = x'P_\ell'D_\ell'D_qP_qt$$

which is of the form in Equation (10) for $B_\ell = D_\ell P_\ell$, $\ell \in \mathbb{N}_n$.

We note that ideas related to those presented in this section appear in (Girosi, 2003).

5.2 Conclusion and Future Work

We developed a framework for multi-task learning in the context of regularization in reproducing kernel Hilbert spaces. This naturally extends standard single-task kernel learning methods, such as SVM and RN. The framework allows to model relationships between the tasks and to learn the task parameters simultaneously. For this purpose, we showed that multi-task learning can be seen as single-task learning if a particular family of kernels, that we called multi-task kernels, is used. We also characterized the non-linear multi-task kernels.

Within the proposed framework, we defined particular linear multi-task kernels that correspond to particular choices of regularizers which model relationships between the function parameters. For example, in the case of SVM, appropriate choices of this kernel/regularizer implemented a trade–off between large margin of each per–task individual SVM and closeness of each SVM to linear combinations of the individual SVMs such as their average.

We tested some of the proposed methods using real data. The experimental results show that the proposed multi-task learning methods can lead to significant performance improvements relative to the single-task learning methods, especially when many tasks with few data each are learned.

A number of research questions can be studied starting from the framework and methods we developed here. We close with commenting on some issues which stem out of the main theme of this paper.

- *Learning a multi-task kernel.* The kernel in Equation (22) is perhaps the simplest nontrivial example of a multi-task kernel. This kernel is a convex combination of two kernels, the first of which corresponds to learning independent tasks and the second one is a rank one kernel which corresponds to learning all tasks as the same task. Thus this kernel linearly combines two opposite models to form a more flexible one. Our experimental results above indicate the value of this approach provided the parameter λ is chosen for the application at hand. Recent work by Micchelli and Pontil (2004) shows that, under rather general conditions, the optimal convex combination of kernels can be learned by minimizing the functional in Equation (1) with respect to K and $f \in \mathcal{H}_K$, where K is a kernel in the convex set of kernels, see also (Lanckriet et al., 2004). Indeed, in our specific case we can show—along the lines in (Micchelli and Pontil, 2004)—that the regularizer (24) is convex in λ and u. This approach is rather general and can be adapted also for learning the matrix Q in the kernel in Equation (25) which in our experiment we estimated by our "ad hoc" PCA approach.

- *Bounds on the generalization error.* Yet another important question is how to bound the generalization error for multi-task learning. Recently developed bounds relying on the notion of algorithmic stability or Rademacher complexity should be easily applicable to our context. This should highlight the role played by the matrices B_ℓ in Equation (10). Intuitively, if $B_\ell = B_0$ we should have a simple (low-complexity) model whereas if the B_ℓ are orthogonal a more complex model. More specifically, this analysis should say how the generalization error, when using the kernel (22), depends on λ.

- *Computational considerations.* A drawback of our proposed multi-task kernel method is that its computational complexity time is $O(p(mn))$ which is worst than the complexity of solving n independent kernel methods, this being $nO(p(m))$. The function p depends on the loss function used and, typically, $p(m) = m^a$ with a a positive constant. For example for the square loss $a = 3$. Future work will focus on the study of efficient decomposition methods for solving the multi-task SVM or RN. This decomposition should exploit the structure provided by the matrices B_ℓ in the kernel (10). For example, if we use the kernel (22) and the tasks share the same input examples it is possible to show that the linear system of mn Equations (15) can be reduced to solving $n+1$ systems of m equations, which is essentially the same as solving n independent ridge regression problems.

- *Multi-task feature selection.* Continuing on the discussion above, we observe that if we restrict the matrix Q to be diagonal then learning Q corresponds to a form of feature selection across tasks. Other feature selection formulations where the tasks may share only some of their features should also be possible. See also the recent work by Jebara (2004) for related work on this direction.

- *Online multi-task learning.* An interesting problem deserving of investigation is the question of how to learn a set of tasks online where at each instance of time a set of examples for a *new task* is sampled. This problem is valuable in applications where an environment is explored and new data/tasks are provided during this exploration. For example, the environment could be a market of customers in our application above, or a set of scenes in computer vision which contains different objects we want to recognize.

- *Multi-task learning extensions.* Finally it would be interesting to extent the framework presented here to other learning problems beyond classification and regression. Two example which come to mind are kernel density estimation, see, for example, (Vapnik, 1998), or one-class SVM (Tax and Duin, 1999).

Acknowledgments

The authors would like to thank Phillip Cartwright and Simon Trusler from Research International for their help with this data set.

References

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

R. K. Ando and T. Zhang. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. Technical Report RC23462, IBM T.J. Watson Research Center, 2004.

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.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 686, pp. 337–404, 1950.

B. Bakker and T. Heskes. Task clustering and gating for Bayesian multi–task 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, pp. 7–39*, 1997.

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

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.

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

L. Breiman and J.H Friedman. Predicting multivariate responses in multiple linear regression. *Royal Statistical Society Series B*, 1998.

P. J. Brown and J. V. Zidek. Adaptive multivariate ridge regression. *The Annals of Statistics, Vol. 8, No. 1, p. 64–74*, 1980.

R. Caruana. Multi–task learning. *Machine Learning, 28, p. 41–75*, 1997.

F. R. K. Chung. *Spectral Graph Theory* CBMS Series, AMS, Providence, 1997.

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

T. Evgeniou, C. Boussios, and G. Zacharia. Generalized robust conjoint estimation. *Marketing Science*, 2005 (forthcoming).

T. Evgeniou and M. Pontil. Regularized multi-task learning. Proceedings of the 10^{th} Conference on '*Knowledge Discovery and Data Mining*, Seattle, WA, August 2004.

F. Girosi. *Demographic Forecasting*. PhD Thesis, Harvard University, 2003.

W. Greene. *Econometric Analysis*. Prentice Hall, fifth edition, 2002.

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.

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

T. Jebara. Multi-Task Feature and Kernel Selection for SVMs. International Conference on Machine Learning, ICML, July 2004.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 1993.

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

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.

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

C. A. Micchelli and M. Pontil. Learning the kernel via regularization. Research Note RN/04/11, Dept of Computer Science, UCL, September, 2004.

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

C. A. Micchelli and M. Pontil. Kernels for multi-task learning. Proc. of the 18–th Conf. on Neural Information Processing Systems, 2005.

R. Rifkin, S. Mukherjee, P. Tamayo, S. Ramaswamy, C. Yeang, M. Angelo, M. Reich, T. Poggio, T. Poggio, E. Lander, T. Golub, and J. Mesirov. An analytical method for multi-class molecular cancer classification *SIAM Review*, Vol. 45, No. 4, p. 706-723, 2003.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, USA, 2002.

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.

V. Srivastava and T. Dwivedi. Estimation of seemingly unrelated regression equations: A brief survey *Journal of Econometrics*, 10, p. 15–32, 1971.

D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20 (11-13), pp. 1191–1199, 1999.

S. Thrun and L. Pratt. *Learning to Learn*. Kluwer Academic Publishers, November 1997.

S. Thrun and J. O'Sullivan. Clustering learning tasks and the selective cross–task transfer of knowledge. *In Learning to Learn*, S. Thrun and L. Y. Pratt Eds., Kluwer Academic Publishers, 1998.

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

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

A. Zellner. An efficient method for estimating seemingly unrelated regression equations and tests for aggregation bias. *Journal of the American Statistical Association*, 57, p. 348–368, 1962.