Joint Kernel Maps

Jason Weston[1], Bernhard Schölkopf[2], and Olivier Bousquet[2]

[1] NEC Laboratories, America, Princeton, NJ, USA
[2] Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany

Abstract. We develop a methodology for solving high dimensional dependency estimation problems between pairs of data types, which is viable in the case where the output of interest has very high dimension, e.g., thousands of dimensions. This is achieved by mapping the objects into continuous or discrete spaces, using joint kernels. Known correlations between input and output can be defined by such kernels, some of which can maintain linearity in the outputs to provide simple (closed form) pre-images. We provide examples of such kernels and empirical results.

1 Introduction

We begin by providing some background in kernel methods.

Suppose we are given empirical data

$$(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \mathcal{Y}. \tag{1}$$

Here, the domain $\mathcal{X}$ is some nonempty set that the inputs x_i are taken from; the $y_i \in \mathcal{Y}$ are called *targets*. Here and below, $i, j = 1, \ldots, m$.

Note that we have not made any assumptions on the domain $\mathcal{X}$ other than it being a set. In order to study the problem of learning, we need additional structure. In learning, we want to be able to *generalize* to unseen data points. In the case of pattern recognition, given some new input $x \in \mathcal{X}$, we want to predict the corresponding $y \in \{\pm 1\}$. Loosely speaking, we want to choose y such that (x, y) is in some sense *similar* to the training examples. To this end, we need similarity measures in $\mathcal{X}$ and in $\{\pm 1\}$. The latter is easier, as two target values can only be identical or different. For the former, we require a similarity measure

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad (x, x') \mapsto k(x, x') \tag{2}$$

with the property that there exists a map Φ into a Hilbert space $\mathcal{H}$ such that for all $x, x' \in \mathcal{X}$,

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle . \tag{3}$$

Such a function k is called a *positive definite (pd) kernel* [1–3], $\mathcal{H}$ is the *reproducing kernel Hilbert space (RKHS)* associated with it, and Φ is called its *feature map*. A popular example, in the case where $\mathcal{X}$ is a normed space, is the Gaussian

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\,\sigma^2}\right), \tag{4}$$

where $\sigma > 0$.

The advantage of using a pd kernel as a similarity measure is that it allows us to construct algorithms in Hilbert spaces. For instance, consider the following simple classification algorithm, where $\mathcal{Y} = \{\pm 1\}$. The idea is to compute the means of the two classes in the RKHS, $\mathbf{c}_1 = \frac{1}{m_1} \sum_{\{i:y_i=+1\}} \Phi(x_i)$, and $\mathbf{c}_2 = \frac{1}{m_2} \sum_{\{i:y_i=-1\}} \Phi(x_i)$, where m_1 and m_2 are the number of examples with positive and negative target values, respectively. We then assign a new point $\Phi(x)$ to the class whose mean is closer to it. This leads to

$$y = \operatorname{sgn}\left(\langle \Phi(x), \mathbf{c}_1 \rangle - \langle \Phi(x), \mathbf{c}_2 \rangle + b\right) \tag{5}$$

with $b = \frac{1}{2}\left(\|\mathbf{c}_2\|^2 - \|\mathbf{c}_1\|^2\right)$. Rewritten in terms of k, this reads

$$y = \operatorname{sgn}\left(\frac{1}{m_1} \sum_{\{i:y_i=+1\}} k(x, x_i) - \frac{1}{m_2} \sum_{\{i:y_i=-1\}} k(x, x_i) + b\right) \tag{6}$$

and $b = \frac{1}{2}\left(\frac{1}{m_2^2} \sum_{\{(i,j):y_i=y_j=-1\}} k(x_i, x_j) - \frac{1}{m_1^2} \sum_{\{(i,j):y_i=y_j=+1\}} k(x_i, x_j)\right)$.

Let us consider one well-known special case of this type of classifier. Assume that the class means have the same distance to the origin (hence $b = 0$), and that k can be viewed as a density, i.e., it is positive and has integral 1, $\int_X k(x, x')dx = 1 \quad \text{for all } x' \in \mathcal{X}$. Then (6) corresponds to the Bayes decision boundary separating the two classes, subject to the assumption that the two classes are equally likely and were generated from two probability distributions that are correctly estimated by the Parzen windows estimators of the two classes,

$$p_1(x) := \frac{1}{m_1} \sum_{\{i:y_i=+1\}} k(x, x_i), \quad p_2(x) := \frac{1}{m_2} \sum_{\{i:y_i=-1\}} k(x, x_i). \tag{7}$$

The classifier (6) is quite close to the *Support Vector Machine (SVM)* that has recently attracted much attention [3–5]. It is linear in the RKHS (see (5)), while in the input domain, it is represented by a kernel expansion (6). It is example-based in the sense that the kernels are centered on the training examples, i.e., one of the two arguments of the kernels is always a training example. This is a general property of kernel methods, due to the Representer Theorem [5, 6]. The main point where SVMs deviate from (6) is in the selection of the examples that the kernels are centered on, and in the weight that is put on the individual kernels in the decision function. The SVM decision boundary takes the form

$$y = \operatorname{sgn}\left(\sum_{i=1}^{m} \lambda_i k(x, x_i) + b\right), \tag{8}$$

where the coefficients λ_i and b are computed by solving a convex quadratic programming problem such that the margin of separation of the classes in the RKHS is maximized. It turns out that for many problems this leads to sparse solutions, i.e., often many of the λ_i take the value 0. The x_i with nonzero λ_i are called *Support Vectors*.

Using methods from statistical learning theory [4], one can bound the generalization error of SVMs. In a nutshell, statistical learning theory shows that it is imperative that one uses a class of functions whose *capacity* (e.g., measure by the VC dimension) is matched to the size of the training set. In SVMs, the capacity measure used is the size of the margin.

The SV algorithm has been generalized to problems such as regression estimation [4], one-class problems and novelty detection [5], as well as to mappings between general sets of objects [7]. The latter uses two kernels, $k_\mathcal{X}$ and $k_\mathcal{Y}$, on $\mathcal{X}$ and $\mathcal{Y}$, respectively, and learns a linear map between the associated RKHS's $\mathcal{H}_\mathcal{X}$ and $\mathcal{H}_\mathcal{Y}$. The feature map of $k_\mathcal{X}$ analyzes the input by computing its feature representation in $\mathcal{H}_\mathcal{X}$, while the feature map of $k_\mathcal{Y}$ synthesizes the output Ψ of the linear map in $\mathcal{H}_\mathcal{Y}$. It can be thought of as inducing a generative model for the outputs; in the algorithm, it is usually employed to compute a pre-image of Ψ in $\mathcal{Y}$.

In the present paper, we will generalize this situation to the case where there is a kernel that jointly compares inputs and outputs.

To this end, first consider the problem of linear regression. Given a training set of paired objects $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$ identically and independently sampled from a distribution P over the product space $\mathcal{X} \times \mathcal{Y}$, we wish to find a function W that maps from $\mathcal{X}$ into $\mathcal{Y}$ such that:

$$\int_{\mathcal{X}\times\mathcal{Y}} \|\mathbf{y} - W\mathbf{x}\|^2_\mathcal{Y} \mathrm{dP}(\mathbf{x}, \mathbf{y})$$

is minimized.

This is a classical learning problem that has been widely studied when $\mathcal{Y} \subset \mathbb{R}^q$ has a small dimension. When the output dimension becomes very high, in order to generalize well one must take into account (i) correlation between output variables (ii) correlation between input variables $\mathcal{X} \subset \mathbb{R}^p$ and (iii) correlation between input *and* output variables.

If prior knowledge about such correlations exists, it can be encoded into a regularizer. For example, a minimization scheme could be adopted that minimizes

$$\frac{1}{m}\sum_{i=1}^{m} \|\mathbf{y}_i - \mathrm{W}\mathbf{x}_i\| + \sum_{i,j=1}^{\dim(\mathcal{X})} \sum_{s,t=1}^{\dim(\mathcal{Y})} W_{ij} W_{st} S_{ijst}.$$

Here, S_{ijst} encodes the correlation between inputs i,j with outputs s and t.

For example, suppose one is learning a mapping between two spaces of equal and large dimension, e.g. pairs of images or spectra. Then the most obvious prior knowledge one has is that, e.g., pixels in images that are close in the input are also close in the output. This knowledge can be encoded into S. The challenge is to rewrite such an optimization problem in the general case so that (i) it can be solved in a dual form to make it tractable for high dimension and (ii) it can be generalized with kernels to also solve nonlinear problems.

In this work we will show how to encode such prior knowledge by defining appropriate joint kernel functions and subsequent minimization in dual variables, building on work such as [7] and [8]. The subsequent algorithm will solve much more than linear regression: it will generalize nonlinear support vector machines for classification *and* regression, and will be also be able to deal with structured outputs such as strings, trees and graphs via kernels [8–10].

2 Linear Maps

We start by learning the linear map W such that a prediction on data is

$$\mathbf{y}(\mathbf{x}) = \text{argmin}_{\mathbf{y}\in\mathcal{Y}} ||\mathrm{W}\mathbf{x} - \mathbf{y}||^2 = \mathrm{W}\mathbf{x}.$$

Note that if the argmin is taken over a linear space, then $\mathbf{y}(\mathbf{x}) = \mathrm{W}\mathbf{x}$, but in more general settings, it will be necessary to compute it using other means. We consider an ε-insensitive loss approach, as in support vector regression [11]. We choose the W that minimizes

$$||\mathrm{W}||^2_{FRO} \tag{9}$$

using the Frobenius norm, subject to

$$||\mathrm{W}\mathbf{x}_i - \mathbf{y}||^2 \geq ||\mathrm{W}\mathbf{x}_i - \mathbf{y}_i||^2 + \varepsilon^2/2, \tag{10}$$

$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : ||\mathbf{y}_i - \mathbf{y}|| > \varepsilon\}.$$

We note that this generalizes support vector classification and Regression:

- For $\mathbf{y} \in \mathbb{R}$ one obtains support vector regression (SVR) [11] without threshold, and for $\mathbf{y} \in \mathbb{R}^q$ one obtains vector-valued ε-insensitive SVR [12]. We rewrite (10) as

  $$\min_{\mathbf{y}\in C_\epsilon(\mathbf{y_i})} ||\mathbf{W}\mathbf{x_i} - \mathbf{y}||^2 \geq ||\mathbf{W}\mathbf{x_i} - \mathbf{y_i}||^2 + \epsilon^2/2$$

  where $C_\epsilon(\mathbf{y_i})$ is the complement of the ball of radius ϵ centered at $\mathbf{y_i}$. If $\mathbf{W}\mathbf{x_i}$ is not in the latter ball, the value of this minimum is zero and the problem does not have any solution. On the other hand, if $\mathbf{W}\mathbf{x_i}$ is in the ball, then this minimum is not zero and can be computed directly. Its value is attained for the following $\mathbf{y}$:

  $$\mathbf{y} = \mathbf{y_i} + \frac{\mathbf{W}\mathbf{x_i} - \mathbf{y_i}}{||\mathbf{W}\mathbf{x_i} - \mathbf{y_i}||}\epsilon.$$

  The value of the minimum is then $(\epsilon - ||\mathbf{W}\mathbf{x_i} - \mathbf{y_i}||)^2$. We then have the constraint

  $$(\epsilon - ||\mathbf{W}\mathbf{x_i} - \mathbf{y_i}||)^2 \geq ||\mathbf{W}\mathbf{x_i} - \mathbf{y_i}||^2 + \epsilon^2/2,$$

  which gives, after some algebra, $||\mathbf{W}\mathbf{x_i} - \mathbf{y_i}|| \leq \epsilon/4$.
- For $y \in \{\pm 1\}$ and $0 \leq \varepsilon < 2$ we obtain two-class SVMs [11] (W is a $1 \times p$ matrix). Expanding the constraint (10) for each i gives

  $$-2y\mathrm{W}x_i + 2y_i\mathrm{W}x_i \geq \varepsilon^2/2.$$

  For $y, y_i \in \{\pm 1\}, ||y_i - y|| > \varepsilon$ only occurs for $y = -y_i$, in which case we have $y_i\mathrm{W}x_i \geq \varepsilon^2/8$, the usual SVM constraints, disregarding scaling and threshold b.

– Similarly, for $\mathbf{y} \in \{0,1\}^q$, where the c_i^{th} entry is 1 when example i is in class c_i, and 0 otherwise, and $0 \leq \varepsilon < \sqrt{2}$ we can obtain multiclass SVMs [13]. As $||\mathbf{y}|| = 1$ we have the constraints

$$\mathbf{y}_i^\top W\mathbf{x}_i - \mathbf{y}^\top \mathbf{W}\mathbf{x}_i \geq \varepsilon^2/4,$$

where the q rows of $\mathrm{W} = \begin{pmatrix} \mathbf{w}_1 \\ \dots \\ \mathbf{w}_q \end{pmatrix}$ correspond to the q hyperplanes of multi-class SVMs (W is a $q \times p$ matrix). Because only one constraint is switched on at one time due to the zeros in $\mathbf{y}$ we have to minimize $||W||^2_{FRO} = \sum_i ||\mathbf{w}_i||^2$ subject to $\forall i, \mathbf{w}_{c_i}\mathbf{x}_i - \mathbf{w}_j\mathbf{x}_i \geq \varepsilon^2/4, \forall j \in \{1,\dots,q\} \setminus c_i$ which is the same as in [13], again disregarding scaling and thresholds.

Generalizing to the non-separable case in the usual manner [8, 11] should be straightforward. Note that the constraints can also be written as:

$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : ||\mathbf{y}_i - \mathbf{y}|| > \varepsilon\}: \;\; 2(\mathbf{y}_i - \mathbf{y})Wx_i \geq \varepsilon^2/2 + ||\mathbf{y}_i||^2 - ||\mathbf{y}||^2. \tag{11}$$

Let us now restrict ourselves slightly to the situation where the outputs are normalized so $\forall \mathbf{y} \in \mathcal{Y} : ||\mathbf{y}|| = 1$. (Obviously this is only useful in the multi-dimensional case.) Hence, we rewrite our optimization problem as: minimize

$$||W||^2_{FRO} \tag{12}$$

subject to

$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : ||\mathbf{y}_i - \mathbf{y}|| > \varepsilon\}: \;\; \mathbf{y}_i^\top W\mathbf{x}_i - \mathbf{y}^\top W\mathbf{x}_i \geq \varepsilon^2/4. \tag{13}$$

We can regard $F(\mathbf{x}, \mathbf{y}) = \mathbf{y}^\top W\mathbf{x}$ as a function that returns the degree of fit between $\mathbf{x}$ and $\mathbf{y}$. The output on a test point can now be written

$$\begin{aligned} \mathbf{y}(\mathbf{x}) &= \text{argmin}_{\mathbf{y} \in \mathcal{Y}} ||W\mathbf{x} - \mathbf{y}||^2 \\ &= \text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{y}^\top W\mathbf{x} = \frac{W\mathbf{x}}{||W\mathbf{x}||}. \end{aligned} \tag{14}$$

because, by Cauchy-Schwarz, the function $\text{argmax}_{\mathbf{y}} \mathbf{y}^\top W\mathbf{x}$ is maximal if $\frac{\mathbf{y}}{||\mathbf{y}||}$ is parallel to $W\mathbf{x}^*$.

With this optimization problem for the case of discrete $\mathcal{Y}$ and $\varepsilon \to 0$, we obtain the support vector machine for interdependent and structured output spaces (SVM-ISOS) of [8]. In practice, one could relax the restriction upon the normalization of $\mathbf{y}$ during training because separability could still be obtained. However, if one is dealing with continuous outputs without this restriction then the preimage given by $\text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{y}^\top W\mathbf{x}$ would not be well defined. This is the reason why in the work of [8] normalization was not an issue, as only the discrete output case was considered[1].

We now show how to develop our method for joint kernels.

[1] In practice, in our experiments with joint kernels, we normalize the joint kernel itself, not the outputs, because the output in this case is not easily accessible.

3 Joint Kernel Maps

We can rewrite the last optimization problem by considering W as a vector $\mathbf{w}$ of dimension $\dim(\mathcal{X})\dim(\mathcal{Y})$, and choosing the feature map

$$\Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x},\mathbf{y}) = \langle(\mathbf{x}\mathbf{y}^\top)_{ij}\rangle_{\substack{i=1,\ldots,\dim(\mathcal{Y})\\ j=1,\ldots,\dim(\mathcal{X}).}}$$

The optimization problem then consists of minimizing[2]

$$\|\mathbf{w}\|^2 \tag{15}$$

subject to

$$\langle \mathbf{w}, \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y}_i) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y})\rangle \geq \varepsilon^2/2, \tag{16}$$
$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\}.$$

However, we are free to choose another mapping, as we shall see later (indeed, choosing a mapping which incorporates prior knowledge is the whole point of using this approach).
We call $\Phi_{\mathcal{X}\mathcal{Y}}$ the *joint kernel map* (JKM), and

$$J((\mathbf{x},\mathbf{y}),(\hat{\mathbf{x}},\hat{\mathbf{y}})) = \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x},\mathbf{y})^\top \Phi_{\mathcal{X}\mathcal{Y}}(\hat{\mathbf{x}},\hat{\mathbf{y}})$$

the *joint kernel*. This relates our method to the work of [14] and [15].

Constructing the corresponding dual problem we obtain: maximize[3]

$$\frac{\varepsilon^2}{4}\sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon} \alpha_{i\mathbf{y}} - \frac{1}{2}\sum_{\substack{i,\mathbf{y}\,:\,\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon\\ j,\hat{\mathbf{y}}\,:\,\|\mathbf{y}_i-\hat{\mathbf{y}}\|>\varepsilon}} \alpha_{i\mathbf{y}}\,\alpha_{j\hat{\mathbf{y}}}\langle \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y}_i) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y}), \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_j,\mathbf{y}_j) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_j,\hat{\mathbf{y}})\rangle$$

subject to

$$\alpha_{ij} \geq 0, \;\; i = 1,\ldots,m, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\}.$$

[2] Note that we could also simplify the optimization problem further by splitting the constraints: i.e. minimize $\|\mathbf{w}\|^2$ subject to

$$\forall i : \langle \mathbf{w}, \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y}_i)\rangle + b \geq \varepsilon^2/8$$
$$\{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| > \varepsilon\} : \;\; \langle \mathbf{w}, \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y})\rangle + b \leq -\varepsilon^2/8.$$

If this problem is linearly separable, then its solution $\mathbf{w}$ is also a feasible solution of (15)-(16).

[3] Note that with infinitely many constraints, standard duality does not apply for our optimization problem. However, for the purposes of the present paper, we are not concerned with this. For practical purposes, we may assume that for any $\epsilon > 0$, our data domain has a finite ϵ-cover (e.g., our domain could be a compact subset of $\mathbb{R}^n$). Since on a computer implementation, a constraint can only be enforced up to machine precision, we can thus imagine choosing a sufficiently small ϵ, which reduces our setting to one with a finite number of constraints. Furthermore, we find experimentally that the number of active constraints is small and scales sublinearly with the number of examples or output dimension (see Figure 1).

The objective can be rewritten with kernels:

$$\frac{\varepsilon^2}{4}\sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon}\alpha_{i\mathbf{y}} - (1/2)\sum_{\substack{i,\mathbf{y}\,:\,\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon\\ j,\hat{\mathbf{y}}\,:\,\|\mathbf{y}_i-\hat{\mathbf{y}}\|>\varepsilon}}\alpha_{i\mathbf{y}}\alpha_{j\hat{\mathbf{y}}}[J((\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}_j,\mathbf{y}_j))$$

$$-J((\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}_j,\hat{\mathbf{y}})) - J((\mathbf{x}_i,\mathbf{y}),(\mathbf{x}_j,\mathbf{y}_j))$$

$$+J((\mathbf{x}_i,\mathbf{y}),(\mathbf{x}_j,\hat{\mathbf{y}}))].$$

The standard linear map therefore requires $J((\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}_j,\mathbf{y}_j)) = \langle\mathbf{x}_i,\mathbf{x}_j\rangle\langle\mathbf{y}_i,\mathbf{y}_j\rangle = K(\mathbf{x}_i,\mathbf{x}_j)L(\mathbf{y}_i,\mathbf{y}_j)$, where $K(\mathbf{x}_i,\mathbf{x}_j) = \langle\mathbf{x}_i,\mathbf{x}_j\rangle$ and $L(\mathbf{y}_i,\mathbf{y}_j) = \langle\mathbf{y}_i,\mathbf{y}_j\rangle$ are kernel maps for input and output respectively.

Now

$$\mathbf{w} = \sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon}\alpha_{ij}[\Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y}_i) - \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}_i,\mathbf{y})].$$

For certain joint kernels (that are linear in the outputs) we can compute the matrix W explicitly to calculate the mapping. However, for general nonlinear mappings of the output (or input) we must solve the pre-image problem (cf. (14)):

$$\begin{aligned}\mathbf{y}(\mathbf{x}^*) &= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}}\langle W,\Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}^*,\mathbf{y})\rangle\\ &= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}}\sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon}\alpha_{ij}J((\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}^*,\mathbf{y}^*))\\ &\qquad - \alpha_{ij}J((\mathbf{x}_i,\mathbf{y}),(\mathbf{x}^*,\mathbf{y}^*)).\end{aligned}$$

In the next section we discuss joint kernels, and consider several examples that do not require one to solve the general pre-image problem. First, let us discuss related work, and practical implementation considerations.

Optimization Finding a solution to the above equations, which contain an infinite number of constraints, is feasible because in practice the solution tends to be very sparse. In fact, the solution can be found in polynomial time if the pre-image can be computed in polynomial time. An efficient method for the SVM for Interdependent and Structured Output Spaces was developed in [8] and can be analogously implemented for Joint Kernel Maps by using an iterative scheme: add the most violating example to the working set and reoptimize, repeating until completion. One can then show that on each iteration the objective function strictly improves and is guaranteed to terminate if the problem is separable. In practice, in our experiments we also start with ε large, and decrease it upon separability.

Related Algorithms The idea of learning maps by embedding both input and output spaces using kernels was first employed in the Kernel Dependency Estimation algorithm [7], where the kernels were defined separately. This allowed correlations to be encoded between output features, nonlinear loss functions to be defined, and for outputs to be structured objects such as strings and trees [8–10] (however, one must then solve an often difficult pre-image problem). The method first decorrelates the outputs via performing a kernel principal component analysis (kPCA). kPCA yields principal components $v_l \in \mathbb{R}^q, l = 1 \ldots n$ and corresponding variances λ_l. Henceforth the output labels $\{y_i\}_{i=1}^m$ are projected to the column vectors v_l to retrieve the m principal coordinates $z_i \in \mathbb{R}^n$. This projection results in the new estimation task

$$\arg\min_{W \in \mathbb{R}^{n \times p}} \sum_{i=1}^{m} \|z_i - Wx_i\|^2.$$

KDE for example performs a ridge regression on each component z_{ij}, $1 \leq j \leq n$ to overcome overfitting. Predictions for a new point $x^\star$ are made via predicting first the principal coordinates $z^\star = Wx^\star$, and then using the principal components.

$$y^\star = Vz^\star.$$

Here $V \in \mathbb{R}^{q \times n}$ consists of the n principal components v_l. In the case where $n = q$ the prediction performance will only depend on the basic regression used for estimating $z^\star$ since V acts as a basis transformation.

If one assumes that the main variation in the output are according to signal and the small variances according to noise, then it is reasonable to take the first n principal components corresponding to the largest variance λ_l. Alternatively, instead of cutting off it is also possible to *shrink* the directions according their variance.

Compared to the current work and work such as SVM-ISOS [8], KDE has the advantage during training of not requiring the computation of pre-images. On the other hand, it requires an expensive matrix inversion step, and does not give sparse solutions. The inability to use Joint Kernels in KDE means that prior knowledge cannot be so easily encoded into the algorithm. In our experiments (see Section 5) the difference between using this prior knowledge or not in real applications can be large, at least for small sample size.

The authors of [16] also provide a method of using kernels to deal with high-dimensional output regression problems using vector-valued kernel functions. One defines a prediction function as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{m} K(\mathbf{x}_i, \mathbf{x})\mathbf{c}_i$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is a q by q matrix which in position $K_{s,t}$ encodes the similarity between training points i and j with respect to outputs s and t. The weights c_i are hence q by 1 vectors. Although at first sight this approach seems very complicated in terms of defining kernels, there are some natural examples where known correlation across outputs can be encoded. However, simply minimizing $\sum_i \|\mathbf{y}_i - f(\mathbf{x}_i)\|^2$ yields a large, non-sparse optimization problem with qm variables.

Considering once again classification problems, the current work also turns out to have strong relations with the work of [15] who employed a ranking perceptron algorithm and a specific joint kernel on the natural language problem of parsing (outputting a parse tree). In this case, the difficult pre-image problem was avoided by only selecting among n pre-selected experts (parsing algorithms). The algorithm they used is thus similar to the one given in footnote 2, except in their case not all possible negative constraints are enforced, but only $n-1$ per example. Using the multi-class SVM formulation of [11, 13]:

$$f(\mathbf{x}_i, \mathbf{y}_i) > f(\mathbf{x}_i, \mathbf{y}), \ \ \forall \{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i\} \tag{17}$$

and considering $\mathcal{Y}$ as some large set, e.g. of structured objects, one arrives at the formulation of SVM-ISOS [8]. Essentially, this is a special case of our algorithm, where the output is structured (discrete $\mathcal{Y}$) and $\varepsilon = 0$[4]. The authors apply the algorithm to problems of label sequence learning, named entity recognition and others. Our work complements this last one in helping to understand the role of joint kernels in learning problems where one can supply prior knowledge by way of the similarity measure. The authors of [17] also provide a similar formulation to [8] but with a probabilistic interpretation.

Although in this paper we do not consider structured output problems, the algorithm we develop could indeed be applied to such problems. Let us consider one such problem, machine translation: translating a sentence into another language. The relation between regression and classification in this framework is an interesting one. On the one hand, one could argue that one desires separability, to return the correct pre-image (sentence) on the training set. This is the approach of [8]. On the other hand, to classify one sentence as correct, and all others as wrong as in the constraints of (17) could be dangerous because it ignores the distance measure in the output space (other sentences may also be plausible.) Thus even when the embedding is discrete, it may make sense to treat it as regression if outputs close in output space have the same "label". Although the authors of [8] try to fix this problem with an adaptive soft margin approach, the ε-insensitive approach of the current paper would preserve sparsity.

4 Joint Kernels

A joint kernel is a nonlinear similarity measure between input-output pairs, i.e., $J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$ where $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ are labeled training examples,[5]

$$J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, \mathbf{y}), \Phi_{\mathcal{X}\mathcal{Y}}(\mathbf{x}', \mathbf{y}') \rangle,$$

where $\Phi_{\mathcal{X}\mathcal{Y}}$ is a map into a dot product space. All functions $J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$ that take this form are positive definite, and all positive definite kernels $J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$ can be written in this form. This follows directly from the corresponding statements for kernels $k(\mathbf{x}, \mathbf{x}')$ (see, for example, [5]). The point of a joint kernel is to describe the similarity between input-output pairs by mapping pairs into a joint space. A joint kernel can encode more than just information about inputs or outputs independent of each other: it can also encode known dependencies/correlations between inputs and outputs. Joint Kernels have already begun to be studied ([14], [8]); however, so far only discrete

[4] Ignoring the normalization conditions on the output which come from our original derivation, as discussed previously.

[5] Note there is nothing stopping us considering not just pairs here but also kernels on n-tuples, e.g., of the form $(\mathbf{x}, \mathbf{y}, \mathbf{z})$.

output spaces and structured outputs (such as sequences) were considered. One of the problems with Joint Kernels is that only for a subset of possible kernels can one compute the pre-image easily. In [8] kernels on sequences are chosen that are amenable to dynamic programming. Although some methods for speeding up pre-image computations exist [18, 19], this remains a difficult problem. In the following we describe some kernels which avoid complex pre-image problems.

Tensor Product Kernels A kernel that does not encode any correlations can be obtained by using the product

$$\begin{aligned} J_{\text{LINEAR}}((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) &= K(\mathbf{x},\mathbf{x}')L(\mathbf{y},\mathbf{y}') \\ &= \langle \Phi_{\mathcal{X}}(\mathbf{x}),\Phi_{\mathcal{X}}(\mathbf{x}')\rangle\langle \Phi_{\mathcal{Y}}(\mathbf{y}),\Phi_{\mathcal{Y}}(\mathbf{y}')\rangle \end{aligned}$$

where K and L are respectively kernels on the inputs and outputs. If K and L are positive definite, then J will be, too; moreover, the associated feature space is known to be the tensor product of the individual feature spaces.

An interesting special case is when L is a linear kernel. In that case

$$\mathrm{W}_{\text{LINEAR}} = \sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon} \alpha_{ij}\Phi_{\mathcal{X}}(\mathbf{x}_i)\mathbf{y}_i^\top - \alpha_{ij}\Phi_{\mathcal{X}}(\mathbf{x}_i)\mathbf{y}^\top.$$

When $\dim(\mathcal{X})$ or $\dim(\mathcal{Y})$ are very large it can be more efficient to avoid the calculation of W and calculate a test prediction directly:

$$\mathrm{W}_{\text{LINEAR}}\mathbf{x} = \sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon} \alpha_{ij}K(\mathbf{x}_i,\mathbf{x})\mathbf{y}_i^\top - \alpha_{ij}K(\mathbf{x}_i,\mathbf{x})\mathbf{y}^\top.$$

Hence we avoid difficult pre-image problems in this case.

Diagonal Regularization Consider the case where $\dim(\mathcal{X}) = \dim(\mathcal{Y})$, and it is known that one is looking for a linear map where the true matrix W is close to the identity map. Slightly more generally, one may know that the n^{th} dimension of the input is correlated with the n^{th} dimension of the output. Instances of such problems include decoding mass spectrometry (mapping from observed to theoretical spectra) and image mapping problems (deblurring, morphing, etc.). This correlation can be directly encoded:

$$\begin{aligned} &J_{\text{DIAG}}((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = \\ &\qquad (1-\lambda)K(\mathbf{x},\mathbf{x}')\langle\mathbf{y},\mathbf{y}'\rangle + \lambda\Big[\sum_{k=1}^{q} x_k x'_k y_k y'_k\Big] \end{aligned} \tag{18}$$

where λ controls the amount of encoded correlation. If λ is large, then the n^{th} dimension in the input is presumed highly correlated with the n^{th} dimension in the output, and the similarity measure is dominated by these relationships. Algorithms that minimize the Frobenius norm choose these dimensions as relevant. Furthermore, the solution is still linear (does not require a pre-image) because we can write

$$\begin{aligned} \mathrm{W}_{\text{DIAG}}\mathbf{x} = {} &(1-\lambda)\mathrm{W}_{\text{LINEAR}}\mathbf{x} + \\ &\lambda \sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon} \alpha_{ij}[\mathrm{DIAG}(\mathbf{x}_i\mathbf{y}_i^\top) - \mathrm{DIAG}(\mathbf{x}_i\mathbf{y}^\top)]\mathbf{x}. \end{aligned}$$

where $D = \text{DIAG}(M)$ is a diagonal matrix with $D_{ii} = M_{ii}$.

Patch-Wise Correlation. The natural generalization of the previous kernel is when you know that the n^{th} dimension of the output is strongly correlated with a known set of dimensions in the input; e.g., for mappings between images, one could know that a region in the output image is strongly correlated with a region in the input image. This knowledge can be encoded with the kernel

$$\begin{aligned} J_{\text{PATCH}}((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) &= (1-\lambda)K(\mathbf{x},\mathbf{x}')\langle\mathbf{y},\mathbf{y}'\rangle \\ &+\lambda\sum_{k=1}^{|\mathcal{P}|}\Big[\sum_{p\in\mathcal{P}_k}\mathbf{x}_p\mathbf{x}'_p\sum_{p\in\mathcal{P}_k}\mathbf{y}_p\mathbf{y}'_p\Big] \end{aligned} \tag{19}$$

where $\mathcal{P}$ is the set of known correlated patches. This encodes patch correlation between dimensions in $\mathbf{x}$, between dimensions in $\mathbf{y}$, and correlation between input and output, i.e. between $\mathbf{x}$ and $\mathbf{y}$.[6] The evaluation on a test example can be expressed as:

$$\begin{aligned} &\mathrm{W}_{\text{PATCH}}\mathbf{x} = (1-\lambda)\mathrm{W}_{\text{LINEAR}}\mathbf{x} \\ &+\lambda\sum_{i,\mathbf{y}:\|\mathbf{y}_i-\mathbf{y}\|>\varepsilon}\alpha_{ij}[\sum_{k=1}^{|\mathcal{P}|}\mathrm{P}_k(\mathbf{x}_i\mathbf{y}_i^\top)-\sum_{k=1}^{|\mathcal{P}|}\mathrm{P}_k(\mathbf{x}_i\mathbf{y}^\top)]\mathbf{x} \end{aligned}$$

where $P = P_k(M)$ is a matrix such that $P_{ij} = M_{ij}$ if $i \in \mathcal{P}_k$ or $j \in \mathcal{P}_k$ (if i or j are in the k^{th} patch), or $P_{ij} = 0$, otherwise.

Image Reconstruction. Consider the problem of image reconstruction. For example, in a problem of digit reconstruction one should predict the bottom half of a digit given its top half. The authors of [7] solved such a problem with the KDE algorithm. The input and output kernels, K and L, used by that algorithm are separate and the algorithm is not given in advance prior knowledge that the two images are related, i.e. that their concatenation creates a single image. The kernels used were

$$\begin{aligned} K(\mathbf{x},\mathbf{x}') &= \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/(2\sigma^2)) \\ L(\mathbf{y},\mathbf{y}') &= \exp(-\|\mathbf{y}-\mathbf{y}'\|^2/(2(\sigma^*)^2)) \end{aligned} \tag{20}$$

In that work it was apparent that sometimes in the middle of the digit this approach can cause some "glitches" when the two halves are connected together. A simple joint kernel such as

$$J_{\text{RBF}}((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = \exp(-\|(\mathbf{x},\mathbf{y})-(\mathbf{x}',\mathbf{y}')\|^2/(2\sigma^2))$$

(i.e. concatenating the images together, and then taking the RBF kernel) could capture more of the problem than taking the product of the kernels in (20). The joint kernel given here would take into account nonlinearities between pixels of input and output

[6] One can introduce a weighting function over the patches, corresponding to the assumption that the closer the pixels are, the more reliable is their correlation, cf. [5, Eq. (13.21)].

dimensions. To improve this method further, invariances could also be encoded into the kernel, e.g. by concatenating the input and output images and then taking into account rotations, translations, etc. A local polynomial kernel [11] which takes encodes spatial information within the image would also help to encode the mapping between input and output; i.e., it would encode that the pixels at the very bottom of the input are highly correlated with the pixels on the top of the output, as before.

5 Experiments

As said before, JKM reduces to support vector classification and regression for particular $\mathcal{Y}$. We therefore only test our algorithm on regression problems of multiple outputs, and show how employing joint kernels can benefit in this case.

5.1 Artificial Problem : The Identity Map

We performed a first experiment on toy data to demonstrate the potential of the approach. We chose a very simple problem: the input are $x_i \in R^p$, each dimension drawn independently from a normal distribution of mean 0, standard deviation 1. The output is the same as the input, $y_i = x_i$, i.e. the task is to learn the identity map.

Table 1. Mean squared error for different joint kernels encoding the identity map (first three rows) compared to ridge regression (RR) and k-nearest neighbors. Incorporating prior knowledge in the joint kernel approach $(\lambda > 0)$ improves generalization performance.

$\dim(\mathcal{X}) = \dim(\mathcal{Y})$	20	30	50	75	100
JKM_{DIAG} $(\lambda = 1)$	0.00	0.00	0.01	0.02	0.02
JKM_{DIAG} $(\lambda = 0.5)$	0.03	0.14	0.34	0.50	0.62
JKM_{DIAG} $(\lambda = 0)$	0.06	0.40	0.78	1.00	1.14
RR (best γ)	0.06	0.43	0.82	1.07	1.21
k-NN (best k)	0.92	1.09	1.27	1.40	1.47

We compared k-nearest neighbor and ridge regression with our approach. For the former (k-NN and RR) we chose the best possible parameters, for the latter (JKM) we show the results for the identity-map regularizing joint kernel (18) for $\lambda = 0, \frac{1}{2}$ and 1, with $\varepsilon = \frac{0.5}{\sqrt{p}}$. For $\lambda = 0$ the set of possible linear maps is free; for $\lambda = 1$ only linear maps that are diagonal matrices are considered.

The mean squared error for $p = 20, \dots, 100$ features are given in Table 1, with 20 examples for training and 100 for testing, averaged over 20 runs. A Wilcoxon signed ranked test confirms that the two kernels with $\gamma > 0$ outperform the other techniques. Further experiments adding noise to the dataset (not shown) yielded similar conclusions. Figure 1 shows the number of active constraints (support vectors) for varying output dimensions with training size 20 (left) and varying training set sizes with output dimension 20 (right). The solutions are relatively sparse (consider that dual ridge regression [20] uses pm variables for p outputs and m examples). Note that larger values of λ (where the capacity of the set of functions is lower) have less active constraints.

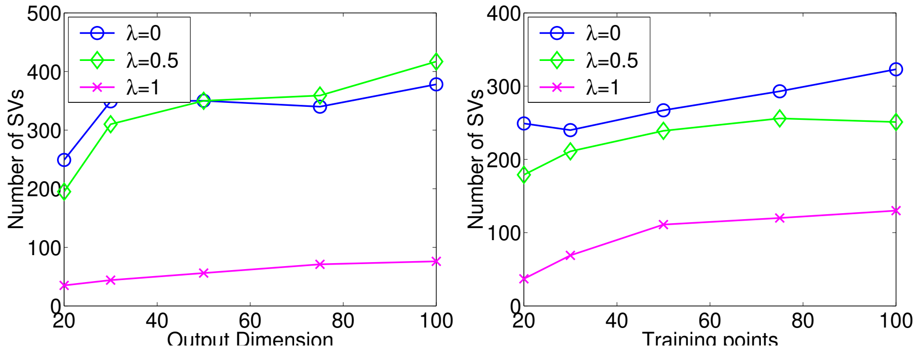

Fig. 1. Number of Active Constraints (Support Vectors) on Artificial data varying output dimension (left) and training set size (right).

5.2 Image Mapping: Learning to Smile

We consider the problem of mapping from the image of a face with a plain expression to an image of the same person smiling using images from the MPI face database [21,22]. We use 20 examples for training, and 50 for testing. The images are $156 \times 176 = 27456$ pixels. We selected a small number of training examples because in this setting the weakness of existing methods was further exposed.

We applied a joint kernel mapping using the tensor product (linear) kernel ($\epsilon = 0.05$) and the patch-wise kernel (19) with $\gamma = 0.95, \epsilon = 0.1$ and patches of size 10×10 which overlap by 5 pixels. Training took 344 and 525 steps of adding a single violating example for the linear and patch kernels, resulting in 150 and 162 support vectors, respectively. Again, we compared with conventional regression techniques, choosing their best possible hyperparameters. A naive employment of ridge regression on this task fails, outputting a kind of "average" face image, independent of the input, see Figure 2. The large dimensionality means there are many solutions with low empirical error, RR (after choosing the optimal regularization constant) selects one that uses many (irrelevant) inputs due to its regularizer. Similarly, k-NN cannot solve this problem well for small sample size. See Figure 2 for example images, and Table 2 for mean squared error rates comparing all these methods. By way of comparison, the baseline of simply predicting the input image as the output (the plain expression) gives a test error of 0.1823 ± 0.003.

5.3 Conclusions

In this work we presented a general method of supervised learning via joint kernel mappings, and showed how such kernels can encode certain regularization properties which reflect prior knowledge in mappings. While the experiments shown here used only simple types of joint kernels taking advantage of patch-wise information, these examples are only an instantiation of our approach, to show its validity and to bring insight into why and how joint kernels are useful. Joint kernels are mainly useful in cases

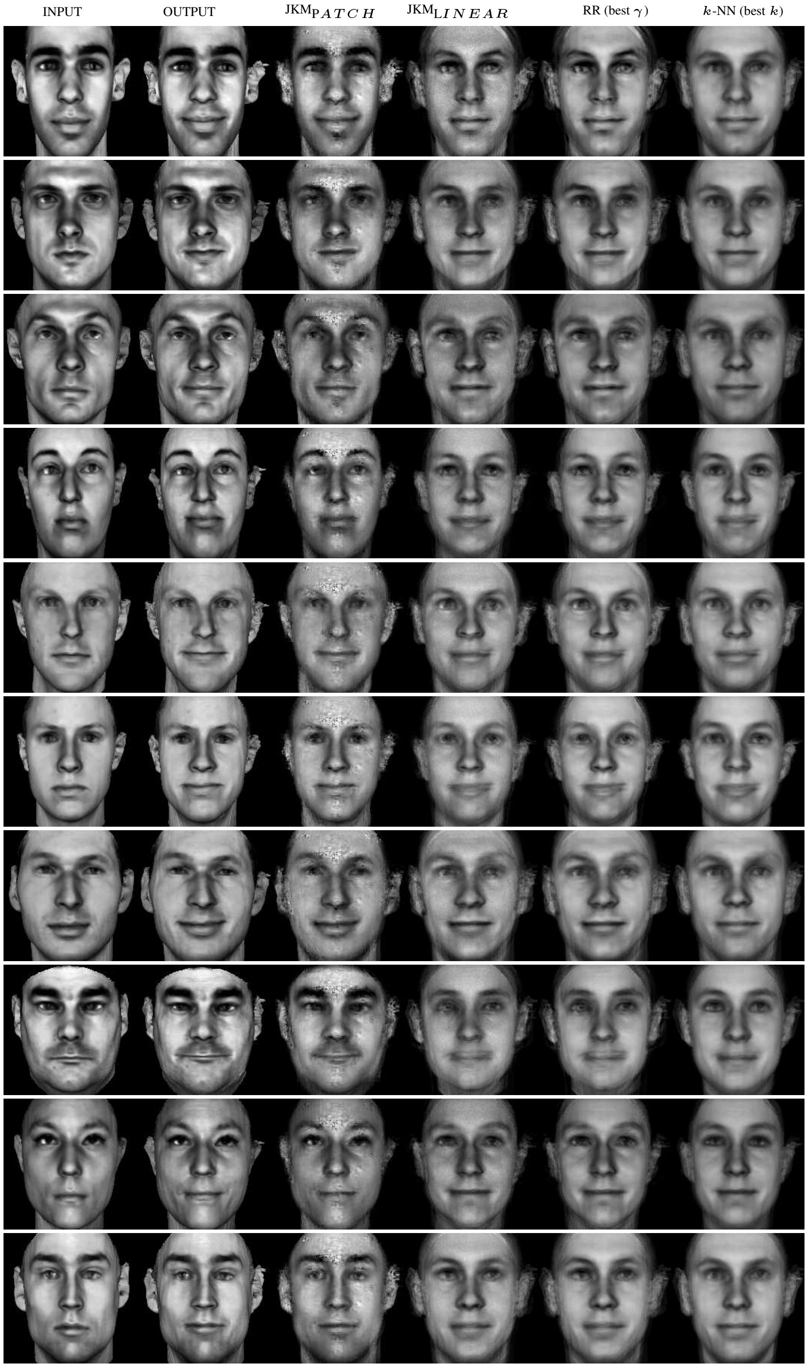

Fig. 2. Prediction of smiling face given plain expression by joint kernel maps (patch and linear) and ridge regression and k-NN. The large dimensionality means there are many solutions with low empirical error, RR (after choosing the optimal regularization constant) selects one that uses many (irrelevant) inputs due to its regularizer $\|w\|^2$ which favors non-sparse solutions. Only the Patch-Kernel Joint Kernel Map is successful, as the choice of (joint) kernel limits the possible choice of functions to ones which are close to the identity map.

Table 2. Test error on the smiling problem of the MPI face database.

	JKM−PATCH ($\varepsilon = 0.1$)	JKM−LINEAR ($\varepsilon = 0.05$)	RR (best γ)	k-NN (best k)
Test error	0.142	0.227	0.222	0.244
Test error	±0.002	±0.006	±0.006	±0.006

where their pre-image is easily computable, and are extendable to complex outputs such as strings, trees and graphs. Indeed, we believe the gain of joint kernel methods is in employing such complex structured outputs that go beyond standard classification and regression such as in parsing, machine translation and other applications. In those cases the difference between coding prior knowledge into a joint kernel and using two separate kernels for input and output could potentially be large, at least in the small sample size case. Although first studies in some of these areas have been completed [8, 15], no study that we know of has yet directly compared this benefit.

Future work should also address issues of efficiency (efficiency of training, pre-images for more complex nonlinear and structured kernels), and to more deeply explore applications of these results.

Acknowledgments We thank Christian Wallraven for providing the MPI face data. We thank André Elisseeff, Goekhan Bakır, Jan Eichorn, Olivier Chapelle, Arthur Gretton, Tobias Mann, William Stafford Noble, Massimiliano Pontil, Fabian Sinz and Thomas Hofmann for useful discussions.

References

1. J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415–446, 1909.
2. M. A. Aizerman, É. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
3. B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.
4. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
5. B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
6. G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.
7. J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. *Neural Processing Information Systems 15*, 2002.
8. I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. *ICML*, 2004.
9. D. Haussler. Convolution kernels on discrete structure. Technical report, UC Santa Cruz, 1999.

10. C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.
11. V. N. Vapnik. *Statistical Learning Theory*. Springer, 1998.
12. F. Pérez-Cruz, G. Camps, E. Soria, J Pérez, A.R. Figueiras-Vidal, and A. Artés-Rodrguez. Multi-dimensional function approximation and regression estimation. In *ICANN*, 2002.
13. J. Weston and C. Watkins. Multi-class support vector machines. *Royal Holloway Technical Report CSD-TR-98-04*, 1998.
14. T. Hofmann, I. Tsochantaridis, and Y. Altun. Learning over discrete output spaces via joint kernel functions. *Kernel Methods Workshop, Neural Processing Information Systems 15*, 2002.
15. M. Collins and N. Duffy. Convolution kernels for natural language. *Neural Processing Information Systems 14*, 2001.
16. C.A. Micchelli and M. Pontil. On learning vector–valued functions. *Research Note RN/03/08, Dept of Computer Science, UCL*, 2003.
17. C. Guestrin B. Taskar and D. Koller. Max-margin markov networks. *Neural Information Processing Systems 16*, 2003.
18. J. T. Kwok and I. W. Tsang. Finding the pre-images in kernel principal component analysis. *6th Annual Workshop On Kernel Machines, Whistler, Canada*, 2002.
19. G .H. Bakir, J. Weston, and B. Schölkopf. Learning to find pre-images. *Advances in Neural Information Processing Systems 16*, 2004.
20. Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann Publishers Inc., 1998.
21. Max Planck Institute Face Database. http://faces.kyb.tuebingen.mpg.de/.
22. V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. *SIGGRAPH'99*, pages 187–194, 1999.