# Predicting Structured Data

## Advances in Neural Information Processing Systems

**Published by Morgan-Kaufmann**

**NIPS-1**
*Advances in Neural Information Processing Systems 1: Proceedings of the 1988 Conference,*
David S. Touretzky, ed., 1989.

**NIPS-2**
*Advances in Neural Information Processing Systems 2: Proceedings of the 1989 Conference,*
David S. Touretzky, ed., 1990.

**NIPS-3**
*Advances in Neural Information Processing Systems 3: Proceedings of the 1990 Conference,*
Richard Lippmann, John E. Moody, and David S. Touretzky, eds., 1991.

**NIPS-4**
*Advances in Neural Information Processing Systems 4: Proceedings of the 1991 Conference,*
John E. Moody, Stephen J. Hanson, and Richard P. Lippmann, eds., 1992.

**NIPS-5**
*Advances in Neural Information Processing Systems 5: Proceedings of the 1992 Conference,*
Stephen J. Hanson, Jack D. Cowan, and C. Lee Giles, eds., 1993.

**NIPS-6**
*Advances in Neural Information Processing Systems 6: Proceedings of the 1993 Conference,*
Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, eds., 1994.

**Published by The MIT Press**

**NIPS-7**
*Advances in Neural Information Processing Systems 7: Proceedings of the 1994 Conference,*
Gerald Tesauro, David S. Touretzky, and Todd K. Leen, eds., 1995.

**NIPS-8**
*Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference,*
David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, eds., 1996.

**NIPS-9**
*Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference,*
Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, eds., 1997.

**NIPS-10**
*Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference,*
Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, eds., 1998.

**NIPS-11**
*Advances in Neural Information Processing Systems 11: Proceedings of the 1998 Conference,*
Michael J. Kearns, Sara A. Solla, and David A. Cohn, eds., 1999.

*Advances in Large Margin Classifiers,*
Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, eds., 2000

# Predicting Structured Data

edited by
Gökhan Bakır
Thomas Hofmann
Bernhard Schölkopf
Alexander J. Smola
Ben Taskar
S.V.N. Vishy Vishwanathan

The MIT Press
Cambridge, Massachusetts
London, England

Printed and bound in the United States of America

Library of Congress Cataloging-in-Publication Data

Predicting Structured Data/ edited by Gökhan Bakır . . . [et al.].
p. cm.
Includes bibliographical references and index.
ISBN 0-262-19448-1 (hc : alk. paper)
1. Machine learning. 2. Algorithms. 3. Kernel functions. I.

Q325.5.A34 2005
006.3'1--dc21

00-027641

# Contents

# 1 Joint Kernel Maps

***Jason Weston***
*NEC Labs America, Princeton, NJ, USA.*
*jasonw@nec-labs.com*

***Goekhan BakIr***
*Max Planck Institute for Biological Cybernetics*
*gb@tuebingen.mpg.de*

***Olivier Bousquet***
*Max Planck Institute for Biological Cybernetics*
*o.bousquet@pertinence.com*

***Bernhard Schoelkopf***
*Max Planck Institute for Biological Cybernetics*
*bs@tuebingen.mpg.de*

***Tobias Mann***
*Department of Genome Sciences, University of Washington, Seattle, WA*
*mann@gs.washington.edu*

***William Stafford Noble***
*Department of Genome Sciences, University of Washington, Seattle, WA*
*noble@gs.washington.edu*

We develop a methodology for solving high dimensional estimation problems between pairs of arbitrary data types as a regression problem. This is achieved by mapping the objects into a continuous or discrete space using joint kernels. The resulting algorithm is an extension of the large margin classification based structured output algorithms to the regression case, and includes all standard SVM-type optimization problems as special cases. Joint kernels allow us to explicitly specify a-priori known input-output and output-output correlations for each output dimension. We provide examples of such kernels and empirical results on mass spectrometry prediction and a problem of image transformation.

## 1.1 Introduction

A standard concern in predicting output data that is non-vectorial is to represent the structural dependencies between output variables in a form amenable for prediction. Consider for example predicting a binary sequence where the $n$-th bit depends on a subset of other output bits. A learning algorithm has to take into account the structure of the output while learning to yield an accurate prediction model. One approach to incorporate such constraints into the learning process is to pose each output variable as a single output dimension and to encode the dependency as an output-output correlation. This will be one of the main concerns of this chapter.

Our framework is based on a multivariate regression setting where we use kernels to map the data into a single *joint* feature space using *joint kernels* (1), a way of implicitly embedding data by only requiring the computation of dot products in the embedding space. Using such joint kernels on inputs and outputs simultanously we are able to encode any prior-knowledge on possible correlations into our learning algorithm.

This chapter is organized as follows. In the following section we start with the basic setting of linear regression and discuss how to incorporate correlation into the problem of learning linear maps. We will show that our formulation is very general, and is in fact a generalization of classification based structured output algorithms to the regression case. In fact it includes all standard SVM-type optimization problems as special cases: binary (2), multi-class (3) and regression (4).

Subsequently we extend our formulation to use joint kernels on both input and output variables simultaneously, which leads to the development of a structured output learning algorithm for the continuous output variable case. We show how this is related to the existing discrete structured output learning case (1). Finally we demonstrate our framework by predicting peptides given an observed mass spectrum which is posed as a prediction problem from a spectrum to a string and on an image transformation problem between related pairs of images where the output is very high dimensional, but local dependencies are known: from the image of a person with a plain expression to the image of them smiling.

## 1.2 Incorporating correlations into linear regression

We begin with the problem of linear regression. Given a training set of input-output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), .., (\mathbf{x}_m, \mathbf{y}_m)\}$ 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:

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

is minimized. Here, we assume $\mathcal{X}$ and $\mathcal{Y}$ are vector spaces; later we will consider them to be any objects, and use kernels to embed them in a vector space.

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

Known input correlations can be exploited with weight sharing (choosing a regularizer that treats these features similarly) or in nonlinear methods by lowering the amount of regularization on nonlinear features that are a function of correlated features, as has been exploited e.g. in digit recognition (7).

A classical approach for incorporating output correlations is to adopt the cost function

$$\frac{1}{m} \sum_{s,t}^{\dim(\mathcal{Y})} \sum_{i=1}^{m} [W\mathbf{x}_i - \mathbf{y}_i]_s [W\mathbf{x}_i - \mathbf{y}_i]_t \sigma_{st},$$

where $[\mathbf{z}]_s$ denotes the $s$-th component of vector $\mathbf{z}$ and $\Sigma = (\sigma_{ij}) \in \mathbb{R}^{\dim(\mathcal{Y}) \times \dim(\mathcal{Y})}$ is a reweighting of the error metric. That is, we replace the Euclidean distance by an arbitrary quadratic form, hence off-diagonal elements of $\Sigma$ incorporate the correlations.

In contrast, if prior knowledge exists on input-output correlations as well one could choose to introduce a regularization functional $\Omega : \mathcal{X}^{\mathcal{Y}} \to \mathbb{R}_+$ instead of altering the used error metric. Then, a classical approach to minimization scheme could be adopted that minimizes

$$\frac{1}{m} \sum_{i=1}^{m} ||W\mathbf{x}_i - \mathbf{y}_i||^2 + \Omega[W].$$

For instance, we propose

$$\Omega[W] = \sum_{i,j=1}^{\dim(\mathcal{X})} \sum_{s,t=1}^{\dim(\mathcal{Y})} W_{is} W_{jt} S_{ijst}, \tag{1.1}$$

where the tensor $S_{ijst}$ encodes the correlation between parameters $W_{is}$ and $W_{jt}$. For example, suppose one is learning a mapping between two images of equal and large dimension, where it is known that the top left corner of the input image is correlated to the top left hand corner of the output image. 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. We will see that regularization schemes based on (1.1) will allow us to achieve this goal.

We now show how to encode such prior knowledge by defining appropriate joint kernel functions and subsequent minimization in dual variables, building on work

such as **??** and (1). 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 (5; 6; 1). This will be an extension of previous work on structured outputs, as our method will be able to deal with both the discrete and continuous output variable cases, whereas only the discrete case was addressed before.

## 1.3 Linear Maps and Kernel Methods : Generalizing Support Vector Machines

In the following section we describe an algorithm which contains all standard support vector algorithms as special cases. We start by considering a linear map $W$. We will make predictions on data using the equation:

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

We consider an $\varepsilon$-insensitive loss approach, as in support vector regression (7). We choose the W that minimizes

$$\|\mathrm{W}\|_{FRO}^2 \tag{1.2}$$

using the Frobenius norm, subject to

$$\|\mathrm{W}\mathbf{x}_i - \mathbf{y}\|^2 \geq \|\mathrm{W}\mathbf{x}_i - \mathbf{y}_i\|^2 + \varepsilon^2/2, \quad \forall i, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| \geq \varepsilon\}. \tag{1.3}$$

Note that the constraints can also be written as: $\forall i\{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| \geq \varepsilon\} : \quad 2(\mathbf{y}_i - \mathbf{y})^\top W x_i \geq \varepsilon^2/2 + \|\mathbf{y}_i\|^2 - \|\mathbf{y}\|^2$. Generalizing to the non-separable case in the usual manner (7; 1) should be straightforward. We now show how this algorithm generalizes both support vector classification and regression:

**Support Vector Regression** For $\mathbf{y} \in \mathbb{R}$ one obtains support vector regression (SVR) (7) without threshold, and for $\mathbf{y} \in \mathbb{R}^q$ one obtains vector-valued $\varepsilon$-insensitive SVR (8). We rewrite (1.3) 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 open ball of radius $\epsilon$ 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, $\|\mathbf{W}\mathbf{x_i} - \mathbf{y_i}\| \leq \epsilon/4$. Disregarding the scaling, this is the same as the usual SVR constraints.

**Support Vector Classification** For $y \in \{\pm 1\}$ and $0 \leq \varepsilon < 2$ we obtain two-class SVMs (7) (W is a $1 \times p$ matrix). Expanding the constraint (1.3) for each $i$ gives $-2y\mathrm{W}x_i + 2y_i\mathrm{W}x_i \geq \varepsilon^2/2$. For $y, y_i \in \{\pm 1\}$, $\|y_i - y\| \geq \varepsilon$ 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 $b$.

**Multi-class Support Vector Machines** 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 (3). As $||\mathbf{y}|| = 1$ we have the constraints $\mathbf{y}_i^\top W\mathbf{x}_i - \mathbf{y}^\top \mathrm{W}\mathbf{x}_i \geq \varepsilon^2/4$ 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 $\|W\|_{\mathrm{FRO}} = \sum_i ||\mathbf{w}_i||^2$ 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 (3), again disregarding scaling and thresholds.

**Structured Output case** 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

$$\|W\|_{FRO}^2 \tag{1.4}$$

subject to

$$\forall i, \{\forall \mathbf{y} \in \mathcal{Y} : \|\mathbf{y}_i - \mathbf{y}\| \geq \varepsilon\} : \quad \mathbf{y}_i^\top W\mathbf{x}_i - \mathbf{y}^\top W\mathbf{x}_i \geq \varepsilon^2/4. \tag{1.5}$$

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

$$\begin{aligned} \mathbf{y}(\mathbf{x}) &= \mathrm{argmin}_{\mathbf{y} \in \mathcal{Y}} \|W\mathbf{x} - \mathbf{y}\|^2 \\ &= \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{y}^\top W\mathbf{x} = \frac{W\mathbf{x}}{\|W\mathbf{x}\|}. \end{aligned} \tag{1.6}$$

because, by Cauchy-Schwarz, the function $\mathrm{argmax}_{\mathbf{y}} \mathbf{y}^\top W\mathbf{x}$ 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 (1). 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 (1) 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.

## 1.4 Joint Kernel Maps

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

$$[\Phi_{\mathcal{XY}}(\mathbf{x},\mathbf{y})]_{ij} = (\mathbf{x}\mathbf{y}^\top)_{ij}, \quad i = 1,\ldots,\dim(\mathcal{Y}), \quad j = 1,\ldots,\dim(\mathcal{X}) \tag{1.7}$$

where the dimensions are indexed by $i$ and $j$. The optimization problem then consists of minimizing[2]

$$\|\mathbf{w}\|^2 \tag{1.8}$$

subject to

$$\langle \mathbf{w}, \Phi_{\mathcal{XY}}(\mathbf{x}_i,\mathbf{y}_i) - \Phi_{\mathcal{XY}}(\mathbf{x}_i,\mathbf{y})\rangle \geq \varepsilon^2/2, \tag{1.9}$$

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

However, we are free to choose another mapping other than the one given above in equation (1.7), as we shall see later. (Indeed, choosing a mapping which incorporates prior knowledge is the whole point of using this approach.) We call $\Phi_{\mathcal{XY}}$ the *joint kernel map* (JKM), and

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

the *joint kernel*. This relates our method to the work of (9) and (10).

---

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

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

If this problem is linearly separable, then its solution $\mathbf{w}$ is also a feasible solution of (1.8)-(1.9).

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

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

$$-\Phi_{\mathcal{XY}}(\mathbf{x}_i,\mathbf{y}),\Phi_{\mathcal{XY}}(\mathbf{x}_j,\mathbf{y}_j)-\Phi_{\mathcal{XY}}(\mathbf{x}_j,\hat{\mathbf{y}})\rangle$$

subject to

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

The objective can be rewritten with kernels:

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

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

The standard linear map therefore implies $J((\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}_j,\mathbf{y}_j))=\langle\mathbf{x}_i,\mathbf{x}_j\rangle\langle\mathbf{y}_i,\mathbf{y}_j\rangle = K(\mathbf{x}_i,\mathbf{x}_j)L(\mathbf{y}_i,\mathbf{y}_j)$, 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

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

For certain joint kernels (that are linear in the outputs) we can compute the matrix W explicitly to calculate the mapping (in the equation above we see the vectorized version of this matrix). However, for general nonlinear mappings of the output (or input) we must solve the pre-image problem (cf. (1.6)):

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

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 $\epsilon$-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 $\epsilon$, 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 1.1).

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 (see Figure 1.1). An efficient method for the SVM for Interdependent and Structured Output Spaces was developed in (1) 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. We used the following simple procedure: For each example $i$ compute $\mathbf{y} = \mathrm{W}\mathbf{x}_i$. We require $||\mathrm{W}\mathbf{x}_i - \mathbf{y}_i|| \leq \varepsilon/4$ (see Section 1.3). So, if $||\mathbf{y} - \mathbf{y}_i|| > \varepsilon/4$, one of the constraints from (1.9) is violated. We thus add $\mathbf{y}$ to our list of active constraints, and reoptimize. We repeat this until there are no more violations. In practice, in our experiments we also start with $\varepsilon$ large, and decrease it upon separability, similar to the procedure in (1).

**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 (11), 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 (5; 6; 1) (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 $\lambda_l$. 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

$$\operatorname*{arg\,min}_{W \in \mathbb{R}^{n \times p}} \sum_{i=1}^{m} \|z_i - W x_i\|^2.$$

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 $z^\star = W x^\star$, and then using the principal components.

$$y^\star = V z^\star.$$

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 is 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 $\lambda_l$. 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 (1), 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 1.6) the difference between using this prior knowledge or not in real applications can be large, at least for small sample size.

The authors of (12) 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 (10) 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 $n$ candidate parse trees. 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 (7; 3):

$$f(\mathbf{x}_i, \mathbf{y}_i) > f(\mathbf{x}_i, \mathbf{y}), \quad \forall\{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i\} \tag{1.10}$$

and considering $\mathcal{Y}$ as some large set, e.g. of structured objects, one arrives at the formulation of SVM-ISOS (1). 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 (13) also provide a similar formulation to (1) but with a probabilistic interpretation. Finally, both the authors of (1) and (**?** ) have generalized the constraints of type (1.10) to be able to quantify how good a

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

prediction is relative to the correct output. One way of doing this is by defining a loss function of choice on $L(\mathbf{y}, \mathbf{y}_i)$ and enforcing a margin on each constraint equal to this loss.

## 1.5 Joint Kernels

As discussed before, 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]

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

where $\Phi_{\mathcal{XY}}$ 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, (14)). 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 ((9),(1)); however, so far only discrete 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 (1) kernels on sequences are chosen that are amenable to dynamic programming. Although some methods for speeding up pre-image computations exist (15; 16), 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

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

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.

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

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

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

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:

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

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:

$$J_{\text{DIAG}}((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = (1-\lambda)K(\mathbf{x},\mathbf{x}')\langle\mathbf{y},\mathbf{y}'\rangle + \lambda\Big[\sum_{k=1}^{q} x_k x'_k y_k y'_k\Big] \tag{1.11}$$

where $\lambda$ controls the amount of encoded correlation. If $\lambda$ 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, because this regularizer gives these features larger weights. Furthermore, the solution is still linear (does not require a pre-image) because we can write

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

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

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

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:

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

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.

## 1.6 Experiments

As said before, the JKM algorithm 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.

### 1.6.1 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.

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

**Table 1.1** 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.

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)

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. (14, Eq. (13.21)).

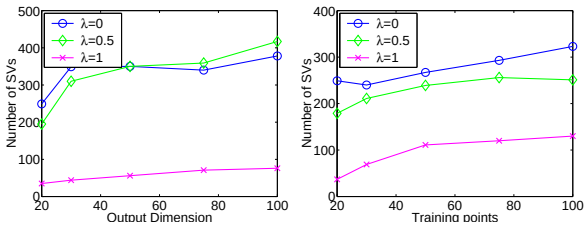


**Figure 1.1 Number of Active Constraints (Support Vectors) on Artificial data** varying output dimension (left) and training set size (right).

we show the results for the identity-map regularizing joint kernel (**??**) 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 $p = 20, \ldots, 100$ features are given in Table 1.1, 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 1.1 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 (17) uses $pm$ variables for $p$ outputs and $m$ examples). Note that larger values of $\lambda$ (where the capacity of the set of functions is lower) have less active constraints.

### 1.6.2 Mass Spectrometry : Prediction of Peptides

An important application of protein mass spectrometry (MS) is to identify proteins in a complex mixture, e.g. blood taken from a patient. In this technique, proteins are ionized and transferred to the gas phase. Their mass to charge ratio can be measured by directing them to an ion detector using an electric field, and this measurement can be used to infer protein identity. In practice, the protein is first dissolved into peptides using an enzyme. These peptides are of varying lengths up to about 20 amino acids. The peptides are run through an MS device, further fragmented, and subjected to a second MS analysis. The final result is one spectrum per peptide, in which the x-axis is the mass-to-charge ratio (m/z) and the y-axis reflects the abundance of subpeptides with the given m/z. This spectrum thus contains information about the peptide sequence, and can be used to identify the protein from which the peptide was cleaved.

The problem is, given such a spectrum, to infer the peptide that generated it. Hence the problem is to map from a spectrum to a string. We used a dataset taken from (18) with a training set of 290 spectra, and a test set of 1277 spectra.

As stated before, JKM generalizes to the case of non-vectorial outputs via the (joint) kernel trick, effectively defining an embedding space via the joint map.

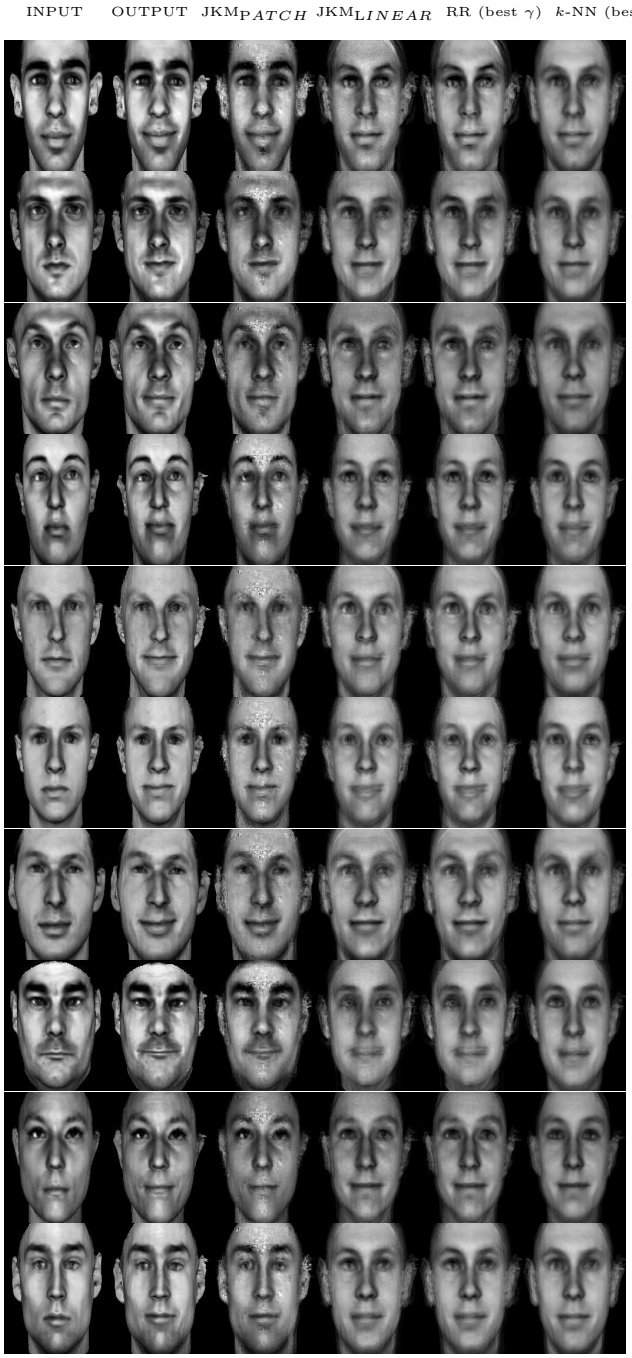


**Figure 1.2** Prediction of smiling face given plain expression by joint kernel maps (patch and linear) and ridge regression and $k$-NN. The large dimensionality means there are many solutions with low empirical error, RR (after choosing the optimal regularization constant) selects one that uses many (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.

| | JKM− PATCH ($\lambda = 0.95$) | JKM− LINEAR ($\lambda = 0$) | RR (best $\gamma$) | $k$-NN (best $k$) |
|---|---|---|---|---|
| Test error | 10.98 ±0.50 | 40.7 ±0.96 | 29.6 ±0.78 | 49.7 ±1.28 |

**Table 1.2** Test error (mean rank of true peptides) on the Mass Spectrometry Problem.

For each peptide in our database the peaks that could be observed in a mass spectrum are known, and are represented as 1000-dimensional vectors. Similarly, the input (the observed spectra) is a vector of the same length. We therefore use the diagonal regularization kernel (**??**) to encode the prior knowledge that the input vector is a noisy variant of the output vector. The quality of a given predictor is inversely proportional to the rank assigned to the true peptide in the ranked output. We use this rank as our performance metric. Here, $\mathcal{Y}$ is the set of known spectra in the database, $|\mathcal{Y}| = 1567$, and $\varepsilon = 0$. As shown in Table 1.2, the diagonal kernel outperforms conventional regression techniques (RR and $k$-NN) even when using their best choice of hyperparameters chosen using the testing set. This preliminary result gives us a hint at the improvement one can get from both encoding information about the known classes in the output space and via encoding knowledge about the map. Note that using existing kernels such as the string kernels used in (1) to represent the outputs would be unlikely to improve this result, because then the joint representation with the inputs would not be possible. We aim to more deeply explore this application in future work.

### 1.6.3 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 (19; 20). 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 with $\gamma = 0.95, \epsilon = 0.1$ and patches of size $10 \times 10$ 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 1.2. 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 1.2 for example images, and Table 1.3 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 $0.1823 \pm 0.003$. The complete test set can be viewed at the supplementary web site.

| | JKM– PATCH ($\varepsilon = 0.1$) | JKM– LINEAR ($\varepsilon = 0.05$) | RR (best $\gamma$) | $k$-NN (best $k$) |
|---|---|---|---|---|
| Test error | 0.142 | 0.227 | 0.222 | 0.244 |
| Test error | $\pm 0.002$ | $\pm 0.006$ | $\pm 0.006$ | $\pm 0.006$ |

**Table 1.3** Test error on the smiling problem of the MPI face database.

### 1.6.4 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 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 (10; 1), no study that we know of has yet directly compared this benefit.

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

***Acknowledgments***

We thank Christian Wallraven for providing the MPI face data. We thank André Elisseeff, Jan Eichorn, Olivier Chapelle, Arthur Gretton, Massimiliano Pontil and Thomas Hofmann for useful discussions.

# References


[1] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. *ICML*, 2004.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the Annual Conference on Computational Learning Theory*, pages 144 – 152, Pittsburgh, PA, July 1992. ACM Press.

[3] J. Weston and C. Watkins. Multi-class support vector machines. *Royal Holloway Technical Report CSD-TR-98-04*, 1998.

[4] H. Drucker, C. J. C. Burges, L. Kaufman, A.J. Smola, and V. Vapnik. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 155 – 161, Cambridge, MA, 1997. MIT Press.

[5] D. Haussler. Convolution kernels on discrete structure. Technical report, UC Santa Cruz, 1999.

[6] C. Watkins. Dynamic alignment kernels. Technical report, UL Royal Holloway, 1999.

[7] V. N. Vapnik. *Statistical Learning Theory*. Springer, 1998.

[8] 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.

[9] 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.

[10] M. Collins and N. Duffy. Convolution kernels for natural language. *Neural Processing Information Systems 14*, 2001.

[11] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. *Neural Processing Information Systems 15*, 2002.

[12] C.A. Micchelli and M. Pontil. On learning vector–valued functions. *Research Note RN/03/08, Dept of Computer Science, UCL*, 2003.

[13] C. Guestrin B. Taskar and D. Koller. Max-margin markov networks. *Neural Information Processing Systems 16*, 2003.

[14] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, 2002.

[15] 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.

[16] G .H. Bakir, J. Weston, and B. Schölkopf. Learning to find pre-images. *Advances in Neural Information Processing Systems 16*, 2004.

[17] 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.

[18] A. Keller, S. Purvine, A. I. Nezvizhskii, S. Stolyar, D. R. Goodlett, and E. Kolker. Experimental protein mixture for validating tandem mass spectral analysis. *OMICS*, 6(2):207–212, 2002.

[19] Max Planck Institute Face Database. http://faces.kyb.tuebingen.mpg.de/.

[20] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. *SIGGRAPH'99*, pages 187–194, 1999.