Als Manuskript gedruckt

Technische Universität Dresden
Institut für Numerische Mathematik

# Pairwise Kernels, Support Vector Machines, and the Application to Large Scale Problems

C. Brunner, A. Fischer, K. Luig, and T. Thies

MATH-NM-04-2011
Oktober 2011

# Pairwise Kernels, Support Vector Machines, and the Application to Large Scale Problems

**Carl Brunner and Andreas Fischer**

Institute of Numerical Mathematics, Department of Mathematics,
Technische Universität Dresden, 01062 Dresden, Germany
c.brunner@gmx.net; andreas.fischer@tu-dresden.de

**Klaus Luig and Thorsten Thies**

Cognitec Systems GmbH
Grossenhainer Str. 101, 01127 Dresden, Germany
luig@cognitec.com; thies@cognitec.com

October 12, 2011

**Abstract.** Support Vector Machines have been applied to multiclass problems in several ways. Here, the task of predicting whether the examples $u, v$ of a pair $(u, v)$ belong to the same class or to different classes is considered. This task can be tackled by means of kernels that take two pairs of examples as arguments. The article provides a systematic treatment of such pairwise kernels. This enables applications with large numbers of examples and classes and an increased classification performance. Moreover, we present a new methodology for parameter tuning. The strength of these contributions is confirmed by excellent results on the labeled faces in the wild benchmark.

**Keywords.** Pairwise Support Vector Machines, Interclass Classification, Pairwise Kernels, Model Selection, Large Scale Problems

**Mathematics Subject Classification 2010:** 68T10, 90C20, 90C06, 46E22

# 1 Introduction

A Support Vector Machine (SVM) is a binary classifier. To extend SVMs to multiclass classification several modifications have been suggested, for example the one against all technique, the one against one technique, directed acyclic graphs, or multiclass SVMs [Rifkin and Klautau, 2004]. A more recent approach for multiclass classification is the pairwiseclassification which relies on two input examples instead of one and predicts whether the two input examples belong to the same class or to different classes [Abernethy et al., 2009, Bar-Hillel et al., 2004a,b, Bar-Hillel and Weinshall, 2007, Ben-Hur and Noble, 2005, Gao and Koller, 2011, Phillips, 1999, Vert et al., 2007]. In this work, an SVM which is able to handle pairwise classification tasks is called pairwise SVM.

A natural requirement for a pairwise classifier is that the order of the two examples should not influence the classification result (symmetry). A common approach to enforce this kind of symmetry is the use of selected kernels. In this paper we discuss relations between such kernels and certain projections. For pairwise SVMs there is another approach to obtain symmetry. It is based on training sets with a special symmetric structure. We will prove a strong connection between both approaches.

A common pairwise classification task is face recognition. There, one is often interested in a good interclass generalization. Therefore, one demands that any person in the training set is not part of the test set. We will demonstrate that training sets with many classes (persons) are needed to obtain a good interclass generalization. The training on such sets is computational expensive. Additionally, most machine learning methods have a large number of parameters. Therefore, we introduce a new heuristic model selection technique based on tasks of increasing difficulty. Furthermore, we discuss the efficient implementation of pairwise SVMs.

This paper is structured as follows. In Section 2 we give a short introduction to pairwise classification. Additionally, we present an overview of pairwise kernels. Subsection 3.1 discusses the symmetry of a pairwise classifier, with an emphasis on pairwise SVMs in Subsection 3.2. Then, the new connection between the two approaches for obtaining symmetry is proved in Subsection 3.3. The new model selection technique is introduced in Section 4, while the efficient implementation of pairwise SVMs is discussed in Section 5. Finally, in Section 6 we provide several experiments on synthetic data and on the labeled faces in the wild (LFW) database. Those experiments confirm our theoretical results and demonstrate the benefits of the model selection technique. In particular, a performance is achieved which is superior to the current state of the art.

# 2 Pairwise Classification

Given are $m$ training examples $x_i \in \mathbb{R}^n$ with $i \in M := \{1, \ldots, m\}$. The class of a training example might be unknown, but we demand that we know for each pair $(x_i, x_j)$ of training examples whether its examples belong to the same class or to different classes. Accordingly, we define $y_{ij} := +1$ if the examples of the pair $(x_i, x_j)$ belong to the same class and call it *positive pair*. Otherwise, if the examples of the pair $(x_i, x_j)$ belong to different classes we define $y_{ij} := -1$ and call it *negative pair*.

In pairwise classification the aim is to decide whether the examples of a pair $(u, v) \in \mathbb{R}^n \times \mathbb{R}^n$ belong to the same class or not. In this paper, we will make use of pairwise decision functions

$f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Such a function predicts whether the examples $u, v$ of a pair $(u, v)$ belong to the same class ($f(u,v) > 0$) or not ($f(u,v) < 0$). Note that neither $u, v$ need belong to the set of training examples nor the classes of $u, v$ need belong to the classes of the training examples.

A natural and desirable property of any pairwise decision function is that it should by symmetric in the following sense:

$$f(u,v) = f(v,u) \quad \text{for all } u, v \in \mathbb{R}^n. \tag{1}$$

A common tool in machine learning are kernels $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Let $\mathscr{H}$ denote an arbitrary Hilbert space with scalar product $\langle \cdot, \cdot \rangle$ and $\phi : \mathbb{R}^n \to \mathscr{H}$, then

$$k(e,f) \coloneqq \langle \phi(e), \phi(f) \rangle$$

defines a *standard* kernel. The extension of those kernels to pairwise classification leads to *pairwise* kernels $K : (\mathbb{R}^n \times \mathbb{R}^n) \times (\mathbb{R}^n \times \mathbb{R}^n) \to \mathbb{R}$. For instance,

$$K^{lin}_{PS}((a,b),(c,d)) \coloneqq (\langle a,c \rangle + \langle b,d \rangle + r)^p$$
$$K^{lin}_{PT}((a,b),(c,d)) \coloneqq (\langle a,c \rangle \cdot \langle b,d \rangle + r)^p$$

with $p \in \mathbb{N}, r \in \mathbb{R}$. We call $K^{lin}_{PS}$ *(linear) polynomial direct sum pairwise kernel* and $K^{lin}_{PT}$ *(linear) polynomial tensor pairwise kernel*.

Let us assume that $I \subseteq M \times M$. Frequently, a pairwise decision function $f$ is given by

$$f(u,v) \coloneqq \sum_{(i,j) \in I} \gamma_{ij} K\left((x_i, x_j), (u,v)\right) + b \tag{2}$$

with $b \in \mathbb{R}$ and $\gamma_{ij} \in \mathbb{R}$ for all $(i,j) \in I$.

Obviously, a decision function of the form (2) which uses $K^{lin}_{PS}$ or $K^{lin}_{PT}$ will not be symmetric in general.

Now, let us define the following pairwise kernels.

$$K^{lin}_{SD}((a,b),(c,d)) \coloneqq \frac{1}{2}\left(\langle a,c \rangle + \langle a,d \rangle + \langle b,c \rangle + \langle b,d \rangle\right)$$
$$K^{lin}_{ML}((a,b),(c,d)) \coloneqq \frac{1}{4}\left(\langle a,c \rangle - \langle a,d \rangle - \langle b,c \rangle + \langle b,d \rangle\right)^2$$
$$K^{lin}_{TL}((a,b),(c,d)) \coloneqq \frac{1}{2}\left(\langle a,c \rangle \langle b,d \rangle + \langle a,d \rangle \langle b,c \rangle\right)$$
$$K^{lin}_{AT}((a,b),(c,d)) \coloneqq \frac{1}{4}\left(\langle a,c \rangle \langle b,d \rangle - \langle a,d \rangle \langle b,c \rangle\right)^2.$$

Obviously, any decision function of the form (2) which uses any of these kernels is symmetric. This motivates to call a kernel $K$ *symmetric* if

$$K((a,b),(c,d)) = K((a,b),(d,c)) \quad \text{for all } a,b,c,d \in \mathbb{R}^n.$$

It might be interesting to combine pairwise kernels with standard kernels [see Ben-Hur and Noble, 2005]. For instance, we could use a polynomial or Gaussian kernel as standard kernel $k$. Now, by slight abuse of notation, we calculate

$$\begin{aligned} K^{lin}_{PS}\left((\phi(a),\phi(b)),(\phi(c),\phi(d))\right) &= \left(\langle \phi(a),\phi(c) \rangle + \langle \phi(b),\phi(d) \rangle\right) + r)^p \\ &= (k(a,c) + k(b,d) + r)^p \end{aligned}$$

This motivates the definition

$$K_{PS}((a,b),(c,d)) := (k(a,c)+k(b,d)+r)^p\,. \tag{3a}$$

Analogously, we obtain

$$K_{PT}((a,b),(c,d)) := (k(a,c)\cdot k(b,d)+r)^p \tag{3b}$$

$$K_{SD}((a,b),(c,d)) := \frac{1}{2}\left(k(a,c)+k(a,d)+k(b,c)+k(b,d)\right) \tag{3c}$$

$$K_{ML}((a,b),(c,d)) := \frac{1}{4}\left(k(a,c)-k(a,d)-k(b,c)+k(b,d)\right)^2 \tag{3d}$$

$$K_{TL}((a,b),(c,d)) := \frac{1}{2}\left(k(a,c)k(b,d)+k(a,d)k(b,c)\right) \tag{3e}$$

$$K_{AT}((a,b),(c,d)) := \frac{1}{4}\left(k(a,c)k(b,d)-k(a,d)k(b,c)\right)^2. \tag{3f}$$

Additionally, for later use we define

$$K_{DS}((a,b,(c,d)) := K_{SD}((a,b),(c,d))+K_{ML}((a,b),(c,d)) \tag{3g}$$

$$K_{TM}((a,b,(c,d)) := K_{TL}((a,b),(c,d))+K_{ML}((a,b),(c,d)). \tag{3h}$$

Vert et al. [2007] call $K_{ML}$ *metric learning pairwise kernel* due to its close connection to the Euclidean metric and call $K_{TL}$ *tensor learning pairwise kernel*. We call $K_{SD}$ *symmetric direct sum pairwise kernel* and the new pairwise kernel $K_{AT}((a,b),(c,d))$ *asymmetric tensor pairwise kernel*. Moreover, we call $K_{DS}$ *direct sum pairwise kernel* and $K_{TM}$ *tensor metric learning pairwise kernel*.

**Remark 1.** *It is well known that if we add or multiply two kernels or multiply one kernel with a positive constant, then we obtain a new kernel. If those operations are applied to symmetric pairwise kernels then the resulting kernel is symmetric, too.*

# 3 Pairwise Symmetry

In Subsection 3.1 we show that the symmetric pairwise kernels presented in Section 2 can be rewritten by means of appropriate projections. In this way we can conclude that the use of some of these kernels may lead to a loss of information. Moreover, for pairwise SVMs with pairwise kernels based on the direct sum of two vector spaces, it has been claimed that any symmetric set of training pairs leads to a symmetric decision function [see Bar-Hillel et al., 2004a]. Note that a set of training pairs is called symmetric, if both $(a,b)$ and $(b,a)$ belong to this set. In Subsection 3.2 we prove these results in a more general context, which includes the tensor product of two vector spaces. Additionally, we show in Subsection 3.3 that using symmetric training sets leads to the same decision function as the use of selected pairwise kernels if we disregard the bias. Interestingly, the application of selected pairwise kernels leads to significantly shorter training times.

Throughout this section we need scalar products on the direct sum $\mathscr{H}\times\mathscr{H}$ and on the tensor product $\mathscr{H}\otimes\mathscr{H}$, where $\mathscr{H}$ denotes an arbitrary Hilbert space. To this end, let us assume that $a,b,c,d$ are elements of $\mathscr{H}$. Then, $\langle(a,b),(c,d)\rangle := \langle a,c\rangle+\langle b,d\rangle$ defines a scalar product on $\mathscr{H}\times\mathscr{H}$ and the bilinear continuation of $\langle(a\otimes b),(c\otimes d)\rangle := \langle a,c\rangle\cdot\langle b,d\rangle$ defines a scalar product on $\mathscr{H}\otimes\mathscr{H}$. Here, $a\otimes b$ denotes the tensor product of $a$ and $b$.

## 3.1 Pairwise Symmetric Kernels

There are several approaches to obtain pairwise symmetric kernels. Basically, those approaches make use of a direct sum of vector spaces or of a tensor product of vector spaces. In this subsection we will show a close relationship between symmetric pairwise kernels and their representation by means of selected projections.

At first, we discuss pairwise kernels based on the direct sum of two vector spaces. We start with $K_{SD}$ (3c). Obviously, this kernel has the desired pairwise symmetry. For a better understanding we only take a linear kernel as standard kernel into account in this case, for instance we use $K_{SD}^{lin}$.

Let us consider the projection $P_{SD}^{lin} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$ with

$$P_{SD}^{lin}(a,b) \coloneqq \frac{1}{2}(a+b, a+b).$$

If we have a closer look at the mapping $P_{SD}^{lin}$ we see that it is an orthogonal projection into the subspace

$$W_{SD}^{lin} \coloneqq \{(u,v) \in \mathbb{R}^n \times \mathbb{R}^n | u = v\}.$$

Additionally, we obtain $P_{SD}^{lin}(a,b) = P_{SD}^{lin}(b,a)$ and

$$\left\langle P_{SD}^{lin}(a,b), P_{SD}^{lin}(c,d) \right\rangle = \frac{1}{2}\left(\langle a,c\rangle + \langle a,d\rangle + \langle b,c\rangle + \langle b,d\rangle\right) = K_{SD}^{lin}((a,b),(c,d)).$$

Therefore, $K_{SD}^{lin}$ is the scalar product of projected pairs of examples. Obviously, by definition, $P_{SD}^{lin}(a,b)$ contains only information about the midpoint of $a$ and $b$. In other words,

$$P_{SD}^{lin}(a,b) = P_{SD}^{lin}(a+t, b-t) \tag{4}$$

holds for all $t \in \mathbb{R}^n$. We will show below that this property is a drawback.

Before, we discuss the Metric Learning Pairwise Kernel $K_{ML}^{lin}$ (3d). Again, this kernel has the desired pairwise symmetry. Let us consider the mapping $P_{ML}^{lin} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$ with

$$P_{ML}^{lin}(a,b) \coloneqq \frac{1}{2}(a-b, b-a).$$

Again, $P_{ML}$ is an orthogonal projection, but instead of projecting onto $W_{SD}^{lin}$ it projects onto

$$W_{ML}^{lin} \coloneqq \{(u,v) \in \mathbb{R}^n \times \mathbb{R}^n | u = -v\}.$$

Note that $K_{SD}^{lin}$ can be seen as counterpart of $K_{ML}^{lin}$ due to the following relationships

$$W_{SD}^{lin} \oplus W_{ML}^{lin} = \mathbb{R}^n \times \mathbb{R}^n \quad \text{and} \quad W_{SD}^{lin} \cap W_{ML}^{lin} = \{0\}.$$

Furthermore, we obtain $P_{ML}^{lin}(a,b) = -P_{ML}^{lin}(b,a)$ and

$$\left\langle P_{ML}^{lin}(a,b), P_{ML}^{lin}(c,d) \right\rangle^2 = \ldots = K_{\mathrm{ML}}^{lin}((a,b),(c,d)). \tag{5}$$

Thus, $P^{lin}_{ML}(a,b)$ contains only information about the relative position of $a$ and $b$, for instance

$$P^{lin}_{ML}(a,b) = P^{lin}_{ML}(a+t,b+t)$$

holds for all $t \in \mathbb{R}^n$. By (5) we see that $K^{lin}_{ML}$ enforces the pairwise symmetry by combining the projection $P^{lin}_{ML}$ with a homogeneous polynomial pairwise kernel of degree 2. So, the question arises, whether we can use other pairwise kernels and still obtain pairwise symmetry if the pairs are projected by $P^{lin}_{ML}$.

**Theorem 2.** *Let us assume that $K$ is a pairwise kernel on the direct sum $\mathbb{R}^n \oplus \mathbb{R}^n$ with*

$$K((a,b),(c,d)) = K((a,b),-(c,d)). \tag{6}$$

*Then, by*

$$\tilde{K}((a,b),(c,d)) \coloneqq K\left(P^{lin}_{ML}(a,b),P^{lin}_{ML}(c,d)\right) \tag{7}$$

*a symmetric pairwise kernel is defined.*

*Proof.*

$$\begin{aligned}\tilde{K}((a,b),(d,c)) =& K\left(P^{lin}_{ML}(a,b),P^{lin}_{ML}(d,c)\right) = K\left(P^{lin}_{ML}(a,b),-P^{lin}_{ML}(c,d)\right)\\ =& K\left(P^{lin}_{ML}(a,b),P^{lin}_{ML}(c,d)\right) = \tilde{K}((a,b),(c,d))\end{aligned}$$

□

Note that $K^{lin}_{PS}$ (3a) with $r = 0$ and even $p$ fulfills property (6). Furthermore, if we replace $P^{lin}_{ML}$ by $P^{lin}_{SD}$ in (7), then we do not need property (6) at all and still get a symmetric pairwise kernel.

Now, we want to discuss the application of $K^{lin}_{SD}$ and $K^{lin}_{ML}$ to the (synthetic) checker board task. In this task the input space is $\mathbb{R}^2$. Furthermore, the examples $a$ and $b$ belong to the same class if and only if $\lfloor a \rfloor = \lfloor b \rfloor$ where the floor operator $\lfloor \cdot \rfloor$ is applied elementwise. Now, let us assume that we use $K^{lin}_{SD}$. Then, due to (4) one would expect a very bad performance since the midpoint of $a$ and $b$ contains almost no relevant information about the classes. Now, let us assume that we use $K^{lin}_{ML}$. Obviously, we can obtain the Euclidean distance between $a$ and $b$ by $P^{lin}_{ML}(a,b)$. Hence, for large distances we know that the examples belong to different classes. For smaller distances this becomes more difficult. Nevertheless, we would expect to achieve a better performance by using $K^{lin}_{ML}$ than by using $K^{lin}_{SD}$. Empirical evidence is given in Figure 1b. Note that it is possible to reconstruct $(a,b)$ if $P^{lin}_{ML}(a,b)$ and $P^{lin}_{SD}(a,b)$ are known. Hence, it might be interesting to use the direct sum pairwise kernel $K_{DS} \coloneqq K_{SD} + K_{ML}$ (3g). However, in Section 6 it is shown that $K_{DS}$ does not lead to good results for the checker board task.

**Remark 3.** *Here, let us mention an approach of Bar-Hillel et al. [2004a]. There, it is proposed to use a representation of $(a,b)$ as*

$$\begin{pmatrix} a+b \\ sgn(a_1-b_1)(a-b) \end{pmatrix}.$$

*Here, $a+b$ is connected to $P^{lin}_{SD}$ while $sgn(a_1-b_1)(a-b)$ is connected to $P^{lin}_{ML}$. However, by multiplying $sgn(a_1-b_1)$ the symmetry is enforced and more general pairwise kernels can be used. For instance,*

$$\bar{K}((a,b),(c,d)) \coloneqq \left\langle \begin{pmatrix} a+b \\ sgn(a_1-b_1)(a-b) \end{pmatrix}, \begin{pmatrix} c+d \\ sgn(c_1-d_1)(c-d) \end{pmatrix} \right\rangle.$$

**Remark 4.** *Note that one can easily adapt the discussions presented above to arbitrary standard kernels instead of linear standard kernels.*

Now, we are going to deal with pairwise kernels which are based on the tensor product of two vector spaces instead of the direct sum of two vector spaces.

We begin this discussion with $K_{TL}$ (3e). Let us define $P_{TL} : \mathscr{H} \otimes \mathscr{H} \to \mathscr{H} \otimes \mathscr{H}$ with $P_{TL}(z) \coloneqq \frac{1}{2}(z + \bar{z})$. Here, $z \in \mathscr{H} \otimes \mathscr{H}$ and $\bar{z}$ denotes the adjoint of $z$. Especially, for $p, q \in \mathscr{H}$ it holds that $p \otimes q \in \mathscr{H} \otimes \mathscr{H}$ and $P_{TL}(p \otimes q) = \frac{1}{2}(p \otimes q + q \otimes p)$. One can easily check that $P_{TL}$ is an orthogonal projection onto the subspace

$$W_{TL} = \operatorname{span}\{p \otimes q + q \otimes p | p, q \in \mathscr{H}\}.$$

Analogously to $P_{ML}^{lin}$ it holds that

$$\langle P_{TL}(\phi(a) \otimes \phi(b)), P_{TL}(\phi(c) \otimes \phi(d)) \rangle = \ldots = K_{TL}((a,b),(c,d)).$$

Again, by using $K_{TL}$ an implicit projection is made and an information loss might occur.

Before discussing this information loss we look at $K_{AT}$ (3f). Let us define $P_{AT} : \mathscr{H} \otimes \mathscr{H} \to \mathscr{H} \otimes \mathscr{H}$ by $P_{AT}(z) \coloneqq \frac{1}{2}(z - \bar{z})$ with $z \in \mathscr{H} \otimes \mathscr{H}$. Especially, for $p, q \in \mathscr{H}$ it holds that $P_{AT}(p \otimes q) = \frac{1}{2}(p \otimes q - q \otimes p)$. Obviously, $P_{AT}(p \otimes q) = -P_{AT}(q \otimes p)$ and $P_{AT}$ is the orthogonal projection onto the subspace

$$W_{AT} = \operatorname{span}\{p \otimes q - q \otimes p | p, q \in \mathscr{H}\}.$$

We obtain

$$W_{TL} \oplus W_{AT} = \mathscr{H} \otimes \mathscr{H} \quad \text{and} \quad W_{TL} \cap W_{AT} = \{0\}.$$

Then, analogously to $P_{ML}^{lin}$

$$\langle P_{AT}(\phi(a) \otimes \phi(b)), P_{AT}(\phi(c) \otimes \phi(d)) \rangle^2 = \ldots = K_{AT}((a,b),(c,d)).$$

Note that Theorem 2 can be transferred easily to the case of tensor products by replacing $P_{ML}^{lin}$ with $P_{AT}$.

To analyze the loss of information of $P_{TL}$ for a linear standard kernel we start with the following lemma.

**Lemma 5.** *For some $x, y \in \mathbb{R}^n \setminus \{0\}$ and $u, v \in \mathbb{R}^n$ let $B \in \mathbb{R}^{n \times n}$ be defined as $B \coloneqq xy^\top$. Then, $uv^\top = B$ holds, if and only if there is some $\lambda \in \mathbb{R} \setminus \{0\}$ so that*

$$(u, v) = (\lambda x, \lambda^{-1} y).$$

*Proof.* Obviously, for $(u, v) = (\lambda x, \lambda^{-1} y)$ it holds that $uv^\top = B$. Thus, we need to prove that no other choice of $(u, v)$ exists.

As $y \neq 0$ there is $k \in \{1, \ldots, n\}$ with $y_k \neq 0$. Hence, the $k$-th column of $B$ is $y_k x$. Hence, $u$ and $x$ must be linearly dependent. Similar arguments show the linear dependence of $v$ and $y$. Thus, there are $\lambda_1, \lambda_2 \in \mathbb{R} \setminus \{0\}$ so that

$$uv^\top = \lambda_1 x (\lambda_2 y)^\top = B.$$

This is true if and only if $\lambda_1 = \lambda_2^{-1}$. □

**Remark 6.** *Let the examples $u, v \in \mathbb{R}^n$ be given. Lemma 5 shows that some information about the norm of the examples is lost if $uv^\top$ is given instead of $(u,v)$. For instance, if we want to decide whether*

$$\lfloor \|u\|_2 \rfloor = \lfloor \|v\|_2 \rfloor$$

*or not, then we cannot answer this question by means of $uv^\top$.*

*Note that the information loss on the norm can be reduced if we have the additional information that each example has the same norm.*

Now, we analyze the loss of information of $P_{TL}$ for a linear standard kernel.

**Theorem 7.** *For some $x, y \in \mathbb{R}^n \setminus \{0\}$ and $u, v \in \mathbb{R}^n$ let $A \in \mathbb{R}^{n \times n}$ be defined as $A \coloneqq xy^\top + yx^\top$. Then, $uv^\top + vu^\top = A$ holds, if and only if there is some $\lambda \in \mathbb{R} \setminus \{0\}$ so that*

$$(u,v) = (\lambda x, \lambda^{-1} y) \quad \text{or} \quad (u,v) = (\lambda y, \lambda^{-1} x).$$

*Proof.* Obviously, for $(u,v) = (\lambda x, \lambda^{-1} y)$ we have $uv^\top + vu^\top = A$ and for $(u,v) = (\lambda y, \lambda^{-1} x)$ we have $uv^\top + vu^\top = A$, too. Thus, we need to prove that no other choice of $(u,v)$ exists.

Firstly, let us assume that $y = \delta x$ for some $\delta \in \mathbb{R} \setminus \{0\}$. Then, $A$ has rank 1. Hence, the image $\mathrm{Im}(A)$ of $A$ has dimension 1. Due to $Ax = 2\delta \langle x, x \rangle x$ we have $\mathrm{Im}(A) = \mathrm{span}\{x\}$. Hence, $u$ and $v$ depend linearly on $x$ as otherwise $\mathrm{Im}\left(uv^\top + vu^\top\right) \neq \mathrm{Im}(A)$. Thus, $u = \lambda_1 x$ and $v = \lambda_2 x$ for some $\lambda_1, \lambda_2 \in \mathbb{R} \setminus \{0\}$. From

$$uv^\top + vu^\top = 2\lambda_1 \lambda_2 xx^\top = A = 2\delta xx^\top$$

we have $\lambda_1 = \delta / \lambda_2$. Then, we obtain $u = \lambda_1 x = (\delta/\lambda_2)x = \lambda_2^{-1} y$ and $v = \lambda_2 x$, or $u = \lambda_1 x$ and $v = \lambda_2 x = (\delta/\lambda_1)x = \lambda_1^{-1} y$.

Secondly, let us assume that $x$ does not linearly depend on $y$. Then, it is easy to verify that $A$ has rank 2. Hence, $\mathrm{Im}(A)$ has dimension 2. To obtain this image let us analyze whether

$$z_1 \coloneqq Ax = \langle x, y \rangle x + \langle x, x \rangle y \quad \text{and} \quad z_2 \coloneqq Ay = \langle y, y \rangle x + \langle x, y \rangle y$$

do linearly depend or not. Therefore, we consider $rz_1 + sz_2 = 0$ for unknown $r, s \in \mathbb{R}$. This yields

$$\begin{aligned} & r(\langle x, y \rangle x + \langle x, x \rangle y) + s(\langle y, y \rangle x + \langle x, y \rangle y) = 0 \\ \Leftrightarrow \quad & (r\langle x, y \rangle + s\langle y, y \rangle)x + (r\langle x, x \rangle + s\langle x, y \rangle)y = 0. \end{aligned}$$

By the linearly independence of $x$ and $y$ this is equivalent to

$$\begin{pmatrix} \langle x, y \rangle & \langle y, y \rangle \\ \langle x, x \rangle & \langle x, y \rangle \end{pmatrix} \begin{pmatrix} r \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{8}$$

For the determinant of the matrix in (8) we obtain by the Cauchy Schwarz inequality and the linear independence of $x$ and $y$

$$\langle x, y \rangle^2 - \langle x, x \rangle \langle y, y \rangle < 0.$$

Hence, $r = s = 0$ is the only solution of (8) and $z_1$ is linearly independent of $z_2$. As the dimension of $\mathrm{Im}(A)$ is 2 we conclude that $\mathrm{Im}(A) = \mathrm{span}\{z_1, z_2\} = \mathrm{span}\{x, y\}$. Therefore, $u, v$ must belong

to span$\{x,y\}$ since otherwise $\mathrm{Im}\left(uv^\top + vu^\top\right) \neq \mathrm{Im}(A)$. Thus, with $u = r_1x + s_1y$ and $v = r_2x + s_2y$ we obtain

$$\begin{aligned} A = xy^\top + yx^\top &= (r_1x + s_1y)(r_2x + s_2y)^\top + (r_2x + s_2y)(r_1x + s_1y)^\top \\ &= (2r_1r_2)xx^\top + (2s_1s_2)yy^\top + (r_1s_2 + r_2s_1)xy^\top + (r_2s_1 + r_1s_2)yx^\top. \end{aligned}$$

Note that the four occurring matrices $xx^\top, yy^\top, xy^\top$, and $yx^\top$ are linearly independent. Hence, by equating the coefficients we have the system

$$2r_1r_2 = 0 \quad r_1s_2 + r_2s_1 = 1 \quad 2s_1s_2 = 0$$

with the solution set

$$\{(r_1,s_1,r_2,s_2) = (\lambda,0,0,\lambda^{-1}), \lambda \in \mathbb{R}\setminus\{0\}\} \cup \{(r_1,s_1,r_2,s_2) = (0,\lambda,\lambda^{-1},0), \lambda \in \mathbb{R}\setminus\{0\}\}.$$

Hence, $(u,v) = (\lambda x, \lambda^{-1}y)$ or $(u,v) = (\lambda y, \lambda^{-1}x)$. □

The last theorem shows that if $P_{TL}(ab^\top)$ is known instead of $ab^\top$, then no information is lost except the ordering, in other words $ab^\top$ is regarded the same as $ba^\top$ (see also Remark 6).

**Remark 8.** *Lemma 5 and Theorem 7 can be extended to the use of arbitrary standard kernels. The extension to standard kernels based on finite dimensional Hilbert spaces is straightforward. However, to extend those results to standard kernels based on infinite dimensional Hilbert spaces, more technical details would be needed.*

Now, we show that some additional information might be lost when $P_{AT}(ab^\top)$ is used instead of $ab^\top$. To this end, let $a,b \in \mathbb{R}^2$ be given and let us define the rotation matrix

$$R(\theta) \coloneqq \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad \text{for some} \quad \theta \in [0,2\pi).$$

Then, we get $P_{AT}\left((R(\theta)a)(R(\theta)b)^\top\right) = P_{AT}(ab^\top)$ for all $\theta$. Hence, by $P_{AT}(ab^\top)$ we cannot answer whether $a,b$ belong to the same orthant, in other words whether $a_1b_1 > 0 \wedge a_2b_2 > 0$.

As conclusion of this subsection we propose that if a symmetric pairwise kernel should be selected, then $K_{DS}$ (3g) should be considered. If all examples have the same norm, then $K_{TL}$ (3e) should also be considered, too. Note that this is a general rule and that in particular cases other symmetric pairwise kernels could have a superior performance.

## 3.2 Pairwise Symmetry and Pairwise SVMs

In the last subsection we presented an approach to obtain a symmetric decision function by means of certain projections. For pairwise SVMs another approach for ensuring a symmetric decision function is known. It is not based on symmetric pairwise kernels but on specially structured training sets. Obviously, if a symmetric kernel is used, then we can exclude the pair $(b,a)$ from the training set if the pair $(a,b)$ is contained. Now, let us assume that we have symmetric training sets for the training of pairwise SVMs, that is if $(a,b)$ is a training pair then $(b,a)$ is a training pair, too. Then, we obtain a symmetric decision function [see Bar-Hillel et al., 2004a, Wei et al., 2006, and Lemma 9 below].

Let $x_i \in \mathbb{R}^n,\ i \in M \coloneqq \{1,\dots,m\}$ be given and let us assume that $I \subseteq M \times M$ with $(i,j) \in I \Rightarrow (j,i) \in I$. Obviously, $I$ leads to a symmetric training set. Additionally, let us define $I_R \subseteq I$ with $I_R \coloneqq \{(i,i)|(i,i) \in I\}$ and $I_N \coloneqq I \setminus I_R$. Furthermore, let us assume that $K$ is a pairwise kernel with

$$K((a,b),(c,d)) = K((b,a),(d,c)). \tag{9}$$

Not that (9) holds for any symmetric pairwise kernels, but also for other pairwise kernels. For instance, for $K = K_{PS}$ (3a) or $K = K_{PT}$ (3b). Now, let us consider the dual pairwise SVM

$$\begin{aligned} g(\alpha) \coloneqq \frac{1}{2} \sum_{(i,j),(k,l)\in I} \alpha_{ij}\alpha_{kl}y_{ij}y_{kl}K((x_i,x_j),(x_k,x_l)) - \sum_{(i,j)\in I} \alpha_{ij} &\longrightarrow \min_{\alpha} \\ \text{s.t. } 0 \le \alpha_{ij} \le C \quad &\text{for all } (i,j) \in I_N \\ 0 \le \alpha_{ii} \le 2C \quad &\text{for all } (i,i) \in I_R \\ \sum_{(i,j)\in I} y_{ij}\alpha_{ij} = 0. & \end{aligned} \tag{10}$$

**Lemma 9.** *Let us assume that* (9) *holds. Then, there is a solution* $\hat{\alpha}$ *of* (10) *with* $\hat{\alpha}_{ij} = \hat{\alpha}_{ji}$ *for all* $(i,j) \in I$. *We call such a solution symmetric.*

*Proof.* By the theorem of Weierstrass there is a solution $\alpha^*$ of (10). Let us define another feasible point $\tilde{\alpha}$ of (10) by

$$\tilde{\alpha}_{ij} \coloneqq \alpha^*_{ji} \quad \text{for all } (i,j) \in I.$$

For easier notation let us define $K_{ij,kl} \coloneqq K((x_i,x_j),(x_k,x_l))$. Then,

$$2g(\tilde{\alpha}) = \sum_{(i,j),(k,l)\in I} \alpha^*_{ji}\alpha^*_{lk}y_{ij}y_{kl}K_{ij,kl} - 2\sum_{(i,j)\in I} \alpha^*_{ji}.$$

Note that $y_{ij} = y_{ji}$ holds for all $(i,j) \in I$. By (9) we further obtain

$$\begin{aligned} 2g(\tilde{\alpha}) &= \sum_{(i,j),(k,l)\in I} \alpha^*_{ji}\alpha^*_{lk}y_{ji}y_{lk}K_{ji,lk} - 2\sum_{(i,j)\in I} \alpha^*_{ji} \\ &= 2g(\alpha^*) \end{aligned}$$

The last equality follows by the symmetry of the set $I$. Hence, $\tilde{\alpha}$ is also a solution of (10). From SVM theory it is known that problem (10) is convex. Therefore,

$$\alpha^\lambda \coloneqq \lambda\alpha^* + (1-\lambda)\tilde{\alpha}$$

solves (10) for any $\lambda \in (0,1)$. Thus, $\alpha^{1/2}$ is a symmetric solution. □

Note that Wei et al. [2006] present a similar result for regression. However, they claim that any solution has the described symmetry. This does not hold in general.

**Theorem 10.** *Each solution* $\alpha$ *of the optimization problem* (10) *leads to a symmetric decision function* $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. *In other words* $f(a,b) = f(b,a)$ *holds for any* $(a,b) \in \mathbb{R}^n \times \mathbb{R}^n$.

*Proof.* For any solution $\alpha$ let us define $g_\alpha : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ by

$$g_\alpha(a,b) \coloneqq \sum_{(i,j)\in I} \alpha_{ij} y_{ij} K((x_i,x_j),(a,b)).$$

Then, the obtained decision function can be written as

$$f_\alpha(a,b) = g_\alpha(a,b) + c$$

for some appropriate $c \in \mathbb{R}$. From SVM theory it is known that if $\alpha^1$ and $\alpha^2$ are solutions of (10) then $g_{\alpha^1} = g_{\alpha^2}$. According to Lemma 9 there is always a solution $\alpha^*$ of (10) with $\alpha^*_{ij} = \alpha^*_{ji}$ for all $(i,j) \in I$. Obviously, such a solution leads to symmetric functions $g_{\alpha^*}$ and $f_{\alpha^*}$. As $g_\alpha$ is the same function for all solutions $\alpha$ of (10) we obtain that $g_\alpha$ and therefore $f_\alpha$ are symmetric for all solutions. □

## 3.3 Connecting Pairwise Kernels and Symmetric Training Sets

To obtain a symmetric decision function we discussed in Subsection 3.1 that if we present projected pairs to a learning machine, then a loss of information may occur. Thereafter, in Subsection 3.2 we used symmetric training sets in pairwise SVMs to obtain a symmetric decision function and presented all the available information to the learning machine. Now, we show in Theorem 11 that the same decision function is obtained, regardless whether a symmetric training set is used or a certain projection is made to enforce symmetry. Hence, even if a symmetric training set is presented to a pairwise SVM and no projection is made the same information loss as in the case of projections occurs. It is known that the needed training time of SMO scales quadratically with the number of training points [see Platt, 1999]. For symmetric training sets the number of training pairs is nearly doubled compared to the number in the case of symmetric kernels. Simultaneously, the cost of symmetric kernels is computationally four times more expensive compared to the corresponding non symmetric kernel. Hence, we expected that both approaches need the same training time. However, Table 1 shows that the approach of using symmetric kernels is significantly faster. Therefore, for pairwise SVMs the approach of using certain projections supersedes the approach of using symmetric training sets. Note that to generate the results in Table 1 the technique of caching the standard kernel values as described in Section 5 is used for both approaches.

| Number of examples | Symmetric Training Set (t in hh:mm) | Symmetric Kernel |
|---:|---:|---:|
| 500 | 0:03 | 0:01 |
| 1000 | 0:46 | 0:17 |
| 1500 | 3:26 | 0:56 |
| 2000 | 9:44 | 2:58 |
| 2500 | 23:15 | 6:20 |

Table 1: Training Time of Symmetric Training Sets vs. Training Time of Symmetric Kernels. The technique described in Section 5 is also used for those measurements.

Let $J$ denote a subset of $I$ with maximal cardinality and with the property $(i,j) \in J \wedge j \neq i \Rightarrow (j,i) \notin J$. Furthermore, $J_R \coloneqq I_R$ and $J_N \coloneqq J \setminus J_R$. Let us consider the optimization problem

$$\begin{aligned} &\frac{1}{2} \sum_{(i,j),(k,l) \in J} \beta_{ij} \beta_{kl} y_{ij} y_{kl} \hat{K}_{ij,kl} - \sum_{(i,j) \in J} \beta_{ij} \longrightarrow \min_{\beta} \\ &\text{s.t. } 0 \leq \beta_{ij} \leq 2C \quad \text{for all} (i,j) \in J \\ &\sum_{(i,j) \in J} y_{ij} \beta_{ij} = 0 \end{aligned} \tag{11}$$

with $\hat{K}_{ij,kl} \coloneqq \frac{1}{2}\left(K_{ij,kl} + K_{ji,kl}\right)$, where $K$ is a pairwise kernel which fulfills (9). For instance, if $K = K_{PS}$ (3a) with $r = 0$ and $p = 1$ then $\hat{K} = K_{SD}$ (3c) or if $K = K_{PT}$ (3b) with $r = 0$ and $p = 1$ then $\hat{K} = K_{TL}$ (3e).

**Theorem 11.** *Let the functions $g_\alpha : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and $h_\beta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be defined by*

$$\begin{aligned} g_\alpha(a,b) &\coloneqq \sum_{(i,j) \in I} \alpha_{ij} y_{ij} K((x_i,x_j),(a,b)), \\ h_\beta(a,b) &\coloneqq \sum_{(i,j) \in J} \beta_{ij} y_{ij} \hat{K}((x_i,x_j),(a,b)), \end{aligned}$$

*where $\alpha$ is a feasible point of* (10) *and $\beta$ is a feasible point of* (11)*. Then, for any solution $\alpha^*$ of* (10) *and for any solution $\beta^*$ of* (11) *it holds that $g_{\alpha^*} = h_{\beta^*}$.*

*Proof.* Due to Lemma 9 and Theorem 10 we can assume that $\alpha^*$ is a symmetric solution of (10). Let us define

$$\bar{\alpha}_{ij} \coloneqq \begin{cases} \beta^*_{ij}/2 & \text{if } (i,j) \in J_N \\ \beta^*_{ii} & \text{if } (i,i) \in J_R \\ \beta^*_{ji}/2 & \text{else} \end{cases} \quad \text{and} \quad \bar{\beta}_{ij} \coloneqq \begin{cases} \alpha^*_{ij} + \alpha^*_{ji} & \text{if } (i,j) \in J_N \\ \alpha^*_{ii} & \text{if } (i,i) \in J_R \end{cases}.$$

Obviously, $\bar{\alpha}$ is a feasible point of (10) and $\bar{\beta}$ is a feasible point of (11). Then, by $\alpha^*_{ij} = \alpha^*_{ji}$ we obtain for

$$\begin{aligned} (i,j) \in J_N: \qquad \bar{\beta}_{ij} \hat{K}_{ij,kl} &= \frac{\bar{\beta}_{ij}}{2}(K_{ij,kl} + K_{ji,kl}) = \frac{\alpha^*_{ij} + \alpha^*_{ji}}{2}\left(K_{ij,kl} + K_{ji,kl}\right) \\ &= \alpha^*_{ij} K_{ij,kl} + \alpha^*_{ji} K_{ji,kl} \\ (i,i) \in J_R: \qquad \bar{\beta}_{ii} \hat{K}_{ii,kl} &= \frac{\bar{\beta}_{ii}}{2}(K_{ii,kl} + K_{ii,kl}) = \alpha^*_{ii} K_{ii,kl} \end{aligned}$$

This implies $g_{\alpha^*} = h_{\bar{\beta}}$. Additionally, we obtain for

$$\begin{aligned} (i,j) \in J_N: \quad \bar{\alpha}_{ij} K_{ij,kl} + \bar{\alpha}_{ji} K_{ji,kl} &= \frac{\beta^*_{ij}}{2}(K_{ij,kl} + K_{ji,kl}) = \beta^*_{ij} \hat{K}_{ij,kl} \\ (i,i) \in J_R: \quad \bar{\alpha}_{ii} K_{ii,kl} &= \frac{\beta^*_{ii}}{2}(K_{ii,kl} + K_{ii,kl}) = \beta^*_{ii} \hat{K}_{ii,kl}. \end{aligned}$$

Hence, $g_{\bar{\alpha}} = h_{\beta^*}$ follows.

In a second step we prove that $\bar{\alpha}$ and $\bar{\beta}$ are solutions of problem (10) and (11), respectively. To this end, note that for any solution of (10) or (11) a corresponding Karush-Kuhn-Tucker (KKT) point exists and, vice versa, that each KKT point corresponds to a solution. Therefore, let us compare the KKT systems of both problems. The KKT system of (10) is

$$\begin{aligned}
\sum_{(k,l)\in I} y_{ij}y_{kl}\alpha_{kl}K_{ij,kl} - 1 - u_{ij} + v_{ij} + wy_{ij} &= 0 \quad \text{for all } (i,j)\in I \\
u_{ij} &\geq 0 \quad \text{for all } (i,j)\in I \\
v_{ij} &\geq 0 \quad \text{for all } (i,j)\in I \\
u_{ij}\alpha_{ij} &= 0 \quad \text{for all } (i,j)\in I \\
v_{ij}(C-\alpha_{ij}) &= 0 \quad \text{for all } (i,j)\in I_N \\
v_{ii}(2C-\alpha_{ii}) &= 0 \quad \text{for all } (i,i)\in I_R \\
C \geq \alpha_{ij} &\geq 0 \quad \text{for all } (i,j)\in I_N \\
2C \geq \alpha_{ii} &\geq 0 \quad \text{for all } (i,i)\in I_R \\
\sum_{(i,j)\in I} y_{ij}\alpha_{ij} &= 0.
\end{aligned}$$

Note that by exchanging the summation index in the definition of $g_\alpha$ from $(i,j)$ to $(k,l)$ we can rewrite the first line by

$$y_{ij}g_\alpha(x_i,x_j) - 1 - u_{ij} + v_{ij} + wy_{ij} = 0 \quad \text{for all } (i,j)\in I.$$

Accordingly, the KKT system of problem (11) is

$$\begin{aligned}
y_{ij}h_\beta(x_i,x_j) - 1 - \lambda_{ij} + \mu_{ij} + \kappa y_{ij} &= 0 \quad \text{for all } (i,j)\in J \\
\lambda_{ij} &\geq 0 \quad \text{for all } (i,j)\in J \\
\mu_{ij} &\geq 0 \quad \text{for all } (i,j)\in J \\
\lambda_{ij}\beta_{ij} &= 0 \quad \text{for all } (i,j)\in J \\
\mu_{ij}(2C-\beta_{ij}) &= 0 \quad \text{for all } (i,j)\in J \\
2C \geq \beta_{ij} &\geq 0 \quad \text{for all } (i,j)\in J \\
\sum_{(i,j)\in J} y_{ij}\beta_{ij} &= 0.
\end{aligned}$$

Let $(\alpha^*,u^*,v^*,w^*)$ be a KKT point of problem (10) and $(\beta^*,\lambda^*,\mu^*,\kappa^*)$ be a KKT point of problem (11). Moreover, let us define

$$\bar{\lambda}_{ij} := u^*_{ij} \quad \text{for all } (i,j)\in J, \qquad \bar{\mu}_{ij} := v^*_{ij} \quad \text{for all } (i,j)\in J, \qquad \bar{\kappa} := w^*,$$

and

$$\bar{u}_{ij} := \begin{cases} \lambda^*_{ij} & \text{for all } (i,j)\in J, \\ \lambda^*_{ji} & \text{for all } (i,j)\in I\setminus J, \end{cases} \qquad \bar{v}_{ij} := \begin{cases} \mu^*_{ij} & \text{for all } (i,j)\in J, \\ \mu^*_{ji} & \text{for all } (i,j)\in I\setminus J, \end{cases} \qquad \bar{w} := \kappa^*.$$

Then, using $h_{\bar{\beta}} = g_{\alpha^*}$ it can be verified by lengthy calculations that $(\bar{\alpha},\bar{u},\bar{v},\bar{w})$ is a KKT point of (10). Similarly, it can be shown that $(\bar{\beta},\bar{\lambda},\bar{\mu},\bar{\kappa})$ is a KKT point of (11), too. Hence, $\bar{\alpha}$ is a solution of (10) and $\bar{\beta}$ is a solution of (11).

From SVM theory it is known that independently of the chosen solution $\alpha^*$ of (10) or $\beta^*$ of (11) we obtain the same $g_{\alpha^*}$ or $h_{\beta^*}$, respectively. This implies $g_{\alpha^*} = g_{\bar{\alpha}} = h_{\bar{\beta}} = h_{\beta^*}$. □

Note that the proof shows how to construct a solution of (10) by a solution of (11) and vice versa.

# 4 A Technique for Model Selection

Now, we discuss an approach for model selection for pairwise classification. To this end, let us introduce several sets. In Section 2 we did not demand to know the explicit classes of the training examples. Here, assume for model selection that the explicit classes are known. Let a *set of training examples* be given. Then, the set consisting of all classes of the training examples is called *set of training classes*. Furthermore, the *set of training pairs* is a chosen subset of the pairs of training examples. Unless otherwise noted, the set of training pairs consists of all possible pairs of training examples. Finally, the *set of test examples*, the *set of test classes*, and the *set of test pairs* are defined accordingly.

Let us now describe three tasks which can be used for model selection. In the *interclass task* the intersection of the set of training classes and the set of test classes is empty. For instance, let us consider face recognition. Then, the interclass task is to classify pairs of images of unknown persons. However, if we use the interclass task to measure the quality of a pairwise classifier, we cannot determine whether a bad result is caused by badly chosen SVM or kernel parameters, by a bad *example per class ratio* (EPCR) in the set of training examples, or by an undersized number of classes in the set of training examples. The EPCR denotes the average number of examples belonging to a single class in a given set of training or test examples. If the number of examples belonging to a given class is equal for all classes, then we call this *constant* EPCR.

In addition to the interclass task it will turn out that the next two tasks can be used for model selection, too. In the *interexample task* the intersection of the set of training examples and the set of test examples is empty while the set of training classes and the set of test classes are equal. Thus, in face recognition the interexample task is to classify pairs of unknown images of known persons. In the *pair task*, the set of training examples and the set of test examples are equal while the intersection of the set of training pairs and the set of test pairs is empty. Therefore, the set of training pairs is a real subset of all pairs of the training examples for this task. Furthermore, the union of the set of training pairs and the set of test pairs equals the set of all pairs of the training examples. Hence, in face recognition the pair task is to classify unknown pairs of known images.

Assume that a pairwise classification task is given. Then, the interclass task seems harder than the interexample task which again seems harder than the pair task. Thus, if we achieve a bad result on the pair task or interexample task, we cannot expect a good result on the interclass task. Therefore, we suggest using the pair task to find sufficiently good parameters of the learning machine like the used kernels or the used penalty parameter of a pairwise SVM, the interexample task to find a sufficient EPCR in the training set, and the interclass task to find a sufficiently large set of training classes. Empirical evidence for using the described technique is given in Section 6. Moreover, it seems important to specify the used task for any measurement of the quality of a pairwise classification task.

Detection error trade-off curves (DET curves) will be used in the following to measure the quality of a pairwise classifier. Such a curve shows for any false match rate (FMR) the corresponding false non match rate (FNMR). A special point of interest of such a curve is the (approximated) equal error rate (EER), that is the value for which FMR=FNMR holds [Gamassi

et al., 2004].

# 5 Implementation

Much effort has been put into solving SVMs efficiently. One of the most widely used techniques is the sequential minimal optimization (SMO) [Platt, 1999]. A well known implementation of this technique is LIBSVM [Chang and Lin, 2011]. Let us assume for the moment that we want to solve a pairwise SVM with kernel $K_{ML}^{lin}$ (3d). To create a training set which can be used by the LIBSVM we calculate $P_{ML}^{lin}(a,b)$ for all used training pairs $(a,b)$ explicitly and save those projected pairs in a file for training. However, this approach leads to superfluously large files as all examples are part of many pairs and therefore are saved repeatedly. For example, to store all pairs of 10,000 examples of dimension 1,000 in an ASCII file, one needed at least 5GB. This situation becomes even worse for other kernels. Therefore, we decided to modify the LIBSVM code.

In a first attempt the examples were stored in RAM and each standard kernel was calculated on demand. This modification suffered from a bad computational performance. One reason for this seems the empirically known fact that the SMO scales quadratically with the number of training points [Platt, 1999]. Note that in our case the training points are the training pairs. For the interclass task and the interexample task the number of training pairs grows quadratically with the number of training examples. If we use $n$ examples, then there are $n(n+1)/2$ pairs. Hence, the runtime of the LIBSVM scales at least quartically with the number $n$ of used training examples. Using 500 training examples already results in 125,250 training pairs and corresponding pairwise SVMs would need several hours to be solved. Therefore, we will present a technique to reduce the needed training time.

Kernel evaluations are crucial for the performance of LIBSVM. If we could cache the whole kernel matrix we would get a huge increase of speed. Today, this seems impossible for significantly more than 125,250 training pairs as storing the (symmetric) kernel matrix for this number of pairs in double precision needs approximately 59GB. However, we describe now how the costs of kernel evaluations can be drastically reduced. In Section 2 we have introduced several kernels. For example, let us select $K_{TL}$ (3e) and an arbitrary standard kernel. For a single evaluation of $K_{TL}$ the standard kernel has to be evaluated four times with vectors of $\mathbb{R}^n$. Afterwards, four arithmetic operations are needed.

It is easy to see that each standard kernel value is needed for many different evaluations of $K_{TL}$. In general, it is possible to cache the standard kernel values for all pairs of examples in the training set. For example, to cache the standard kernel values for 10,000 examples one needs 400MB. Thus, if all standard kernel values are cached, then each kernel evaluation of $K_{TL}$ costs four arithmetic operations. This does not depend on the chosen standard kernel. Using other pairwise kernels is at most slightly more expensive. Furthermore, we could free memory by deleting the examples after computing the standard kernel values as we do not need them anymore. Additionally, the dimension of the examples does not influence the costs of a single pairwise kernel evaluation in any case. Only the time needed to calculate all standard kernel values depends on the dimension $n$ of the examples. However, if a linear standard kernel is used, then in case of 10,000 examples of dimension 1,000, one needs about $10^{11}$ operations to calculate all such values. This can be done in less than a minute.

Tables 2a and 2b compare the training times with and without the described technique. For

this measurement examples from the Double Interval Task (cf. Subsection 6.2) are used with a constant EPCR of 5, $K_{TL}^{lin}$ as pairwise kernel, a cache size of 100MB, and all pairs are used for training. In each run of Table 2a 250 examples are used for different dimensions $n$. Table 2b shows results for different numbers of examples of dimension $n = 500$. The speedup factor by the described technique is up to 100.

| Dimension $n$ of examples | normal (t in mm:ss) | improved |
|---|---|---|
| 200 | 2:08 | 0:07 |
| 400 | 4:31 | 0:07 |
| 600 | 6:24 | 0:07 |
| 800 | 9:41 | 0:08 |
| 1000 | 11:27 | 0:09 |

(a) Different dimensions $n$ of examples

| Number of examples | normal (t in hh:mm) | improved |
|---|---|---|
| 200 | 0:04 | 0:00 |
| 400 | 1:05 | 0:01 |
| 600 | 4:17 | 0:02 |
| 800 | 12:40 | 0:06 |
| 1000 | 28:43 | 0:13 |

(b) Different numbers of examples

Table 2: Training time

# 6 Experiments

In this section we will present results of applying pairwise SVMs to two synthetic datasets and to one real world dataset. Before we come to those datasets we introduce the pairwise kernel $K_{TM}^{lin} := K_{TL}^{lin} + K_{ML}^{lin}$ (3). Furthermore, let $K_{SD}^{poly}$ denote $K_{SD}$ with homogenous polynomial standard kernel of degree two. In analogy to $K_{SD}^{poly}$ the kernels $K_{ML}^{poly}$, $K_{TL}^{poly}$, and $K_{TM}^{poly}$ are defined.

## 6.1 Checker Board Task

As described in Subsection 3.1 an example $x$ of this task has the following form:

$$x \in \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{with} \quad x_1, x_2 \in [0, m)$$

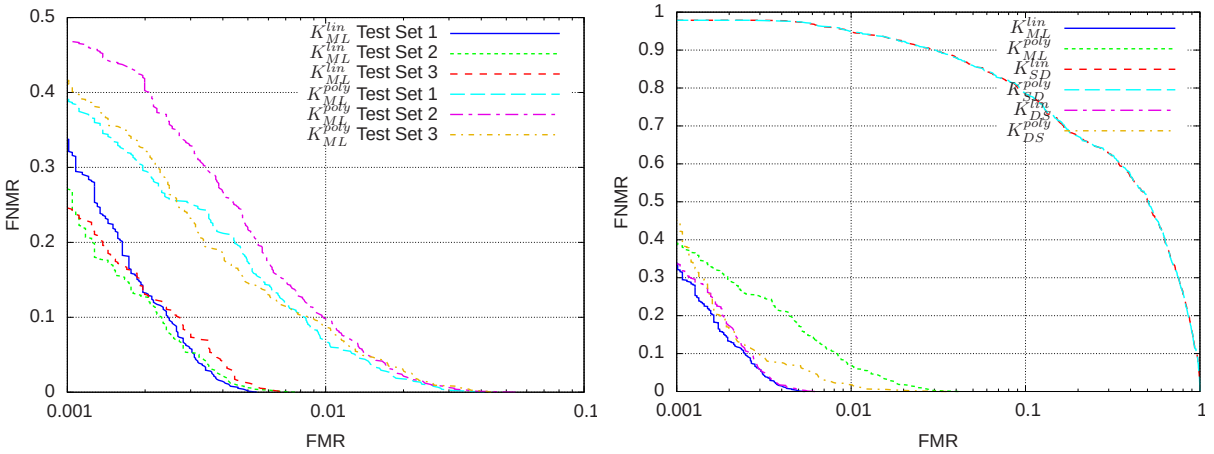
and $m \in \mathbb{N}$ arbitrary but fixed. The class $c$ of an example $x$ is determined by

$$c(x) := \text{floor}(x_1)m + \text{floor}(x_2).$$

In other words each square of the checker board is a single class of the $m^2$ classes.

For our measurements we selected $m = 25$ and set the penalty parameter $C = 1$. For model selection we used the technique described in Section 4. To keep this work short, we only present selected results. In the pair task we created a set consisting of 50 classes with a constant EPCR of 5. Using the pair task we excluded many kernels as they led to bad results in this task. As test set for the interexample and interclass task we used the whole set used in the pair task and call this set Test Set 1. In the interexample task we trained on newly generated training sets with different EPCRs (5,10,15,20,25). We observed that the EPCR does not significantly influence

the performance for the checker board task. Hence, we chose an EPCR of 5. Furthermore, we obtained by the interexample task that $K_{ML}^{lin}$ (3d) and $K_{ML}^{poly}$ have the best performance. Then, we tested different numbers $(50, 75, \ldots, 200)$ of training classes within the interclass task. Again, the results differed only slightly. Hence, we selected the two models with 50 classes. Figure 1a presents the performance of the models with $K_{ML}^{lin}$ and $K_{ML}^{poly}$ on Test Set 1. Additionally, we tested the performance on two newly generated test set (Test Set 2 and Test Set 3) with the same properties as Test Set 1 (interclass task, 50 classes, EPCR=5). This shows the robustness of the model selection technique. In Figure 1b we also present the performance for different kernels in the interclass task to complete the discussion of Section 3.




(a) Interclass task: robustness

(b) Interclass task: different kernels

Figure 1: DET curves for checker board task. In (b) $K_{SD}^{lin}$ and $K_{SD}^{poly}$ (3c) provide almost the same curve.

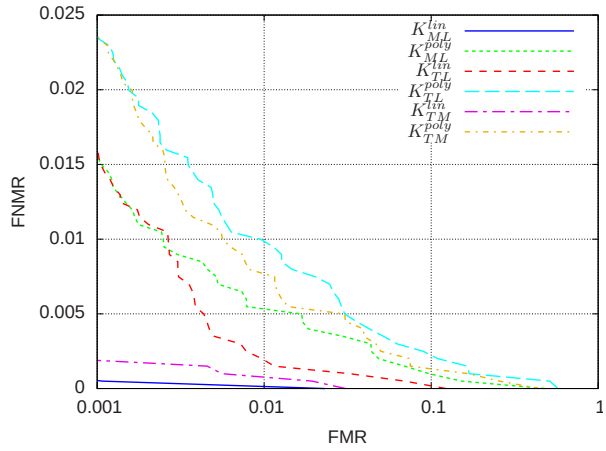
## 6.2 Double Interval Task

Let us define the *double interval task* of dimension $n$. To draw an example $x \in \{-1,1\}^n$ of the double interval task one draws $i, j, k, l \in \mathbb{N}$ so that $2 \leq i \leq j, j+2 \leq k \leq l \leq n$ and sets

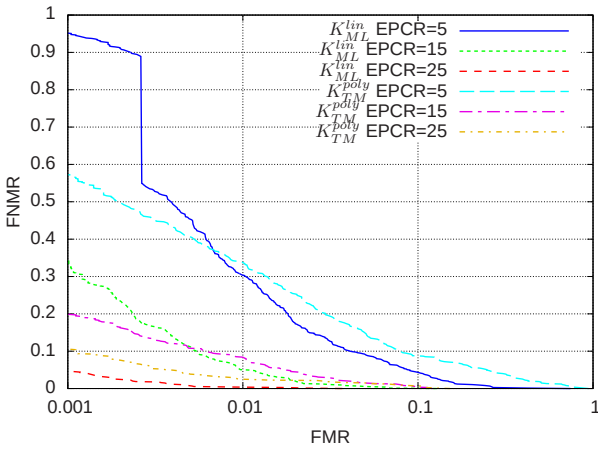
$$x_p := \begin{cases} 1 & p \in \{i, \ldots, j\} \cup \{k, \ldots, l\}, \\ -1 & \text{otherwise.} \end{cases}$$

The class $c$ of such an example is defined by $c(x) := (i,k)$. Note that the pair $(j,l)$ does not influence the class. Hence, there are $(n-3)(n-2)/2$ classes.

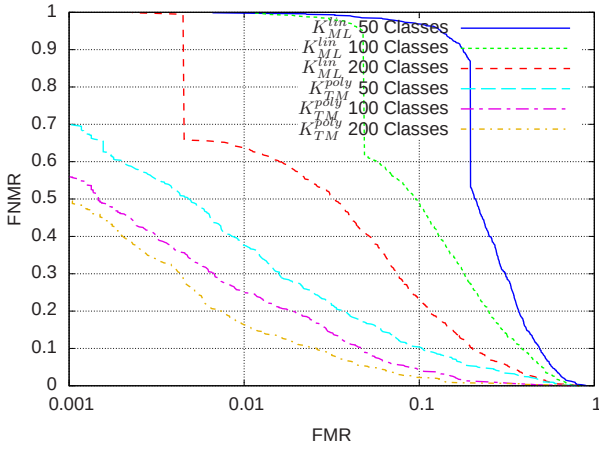
Obviously, all examples have the same Euclidean norm. For our measurements we selected $n = 500$. In the pair task we created an initial set consisting of 750 examples out of 50 classes with a constant EPCR of 15. Then, we used 75% of all pairs for training and tested on the remaining ones. By the pair task several parameters were selected. Firstly, it turned out that the penalty parameter $C$ should be set to 1,000 independently of the other parameters. Secondly, the kernels $K_{ML}^{lin}$, $K_{ML}^{poly}$, $K_{TL}^{lin}$, $K_{TL}^{poly}$, $K_{TM}^{lin}$, and $K_{TM}^{poly}$ (3) were selected due to their superior performance. In Figure 2a, we present the performance of those kernels in the pair task. Afterwards, the whole set of the pair task is used as test set for the interexample and interclass task and is called Test Set 1. In the interexample task, we tested different EPCRs (5,10,15,20,25). In



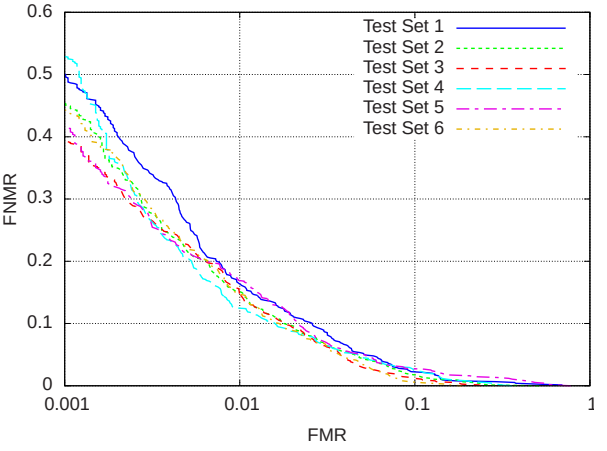

(a) Pair task: different kernels



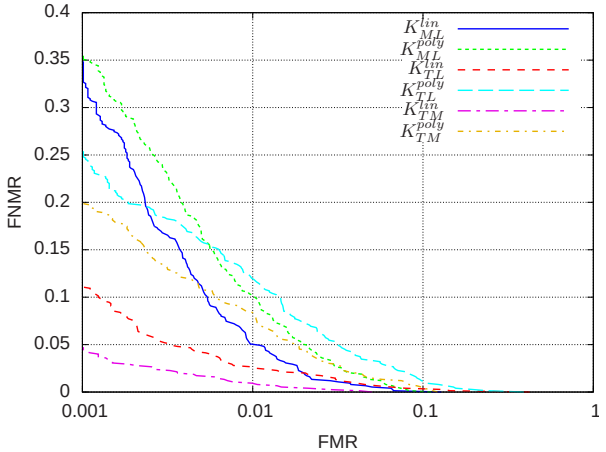

(b) Interexample task: different EPCRs



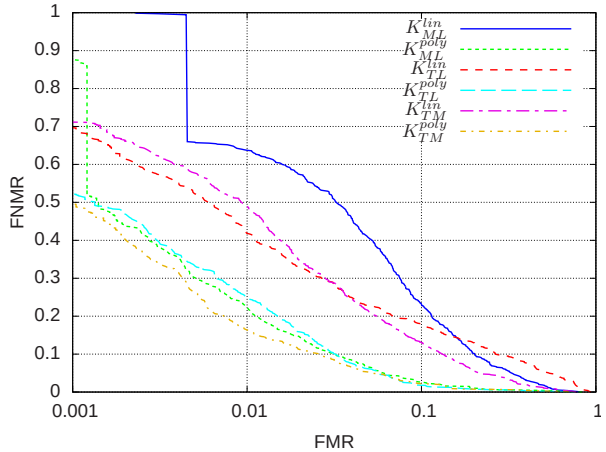

(c) Interclass task: different class numbers




(d) Interclass task: robustness




(e) Interexample task: different kernels




(f) Interclass task: different kernels

Figure 2: DET curves for double interval tasks

Figure 2b it is shown that increasing the EPCR leads to better results in the interexample task. This holds for all kernels selected. Due to space limitations we only present results for $K_{ML}^{lin}$ and $K_{TM}^{poly}$. Note, that we chose as trade-off between performance and needed training time an EPCR of 15. In Figure 2c we show that an increasing number of used classes increases the performance in the interclass task. Again, this holds for all kernels mentioned above but we do only present results for $K_{ML}^{lin}$ and $K_{TM}^{poly}$. Furthermore, using six different test sets we show in Figure 2d that the heuristic model selection technique led to robust results for the interclass task for kernel $K_{TM}^{poly}$.

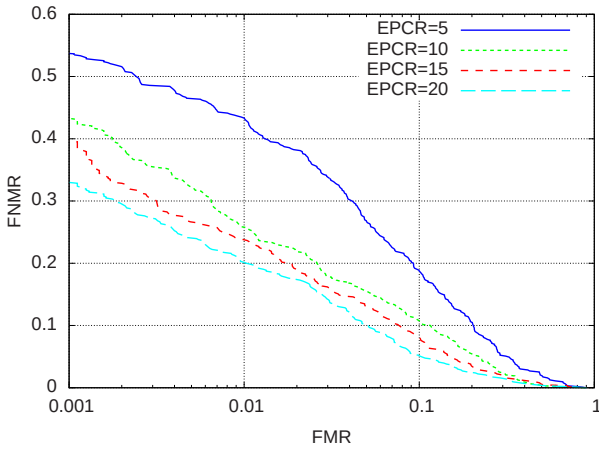
By Figures 2a, 2e, and 2f it can be seen for a fixed kernel that the DET curve of the pair task is below the DET curve of the interexample task, which again is below the DET curve of the interclass task. In the interexample task (Figure 2e) a training set consisting of 50 classes with a constant EPCR of 15 is used, while a training set consisting of 200 classes with a constant EPCR of 15 is used in the interclass task (Figure 2f). One obtains that $K_{ML}^{lin}$ is the best kernel in the pair task and interexample task. However, compared to the other selected kernels it leads to bad results in the interclass task. At the same time the performance of $K_{TM}^{poly}$ in the pair task and interexample task is worse than most of the other used kernels. However, $K_{TM}^{poly}$ leads to the best performance in the interclass task.

Note that the model selection technique is based on the assumption that a DET curve of the pair task is below a corresponding DET curve of the interexample task, which again is below a DET curve of the interclass task. This does neither mean that a good result in the pair task implies a good result in the interexample task nor that a good result in the interexample task implies a good result in the interclass task.

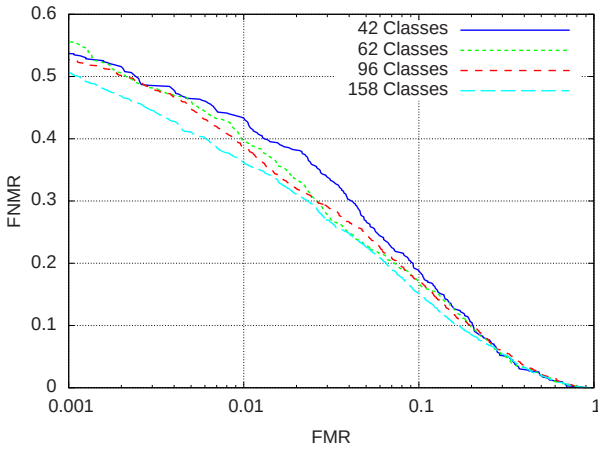
## 6.3 Labeled Faces in the Wild

The labeled faces in the wild (LFW) dataset [Huang et al., 2007] consists of 13,233 images of 5,749 persons. Several remarks on this dataset are in order. Firstly, the dataset is very inhomogeneous. There are only 1,680 persons with two or more images. Moreover, there are persons with up to 530 images. Secondly, Huang et al. [2007] suggest two standard test procedures for this dataset. Here, the unrestricted test procedure is used. This test procedure is a fixed tenfold cross validation in the interclass setting, where each test set consists of 300 positive pairs and 300 negative pairs. Thirdly, there are several feature vectors available for the LFW dataset. For the presented measurements we mainly followed Prince et al. [2011] and used the scale-invariant feature transform (SIFT)-based feature vectors for the funneled version [Guillaumin et al., 2009] of LFW. In addition, the aligned images [Wolf et al., 2009] are used as well. Again, following Prince et al. [2011], the aligned images are cropped to 80×150 pixels and are then normalized by passing them through a log function ($\log(x+1)$). Afterwards, the local binary patterns (LBP) [Ojala et al., 2002] and three-patch LBP (TPLBP) [Wolf et al., 2008] are extracted. In contrast to Prince et al. [2011] the pose is neither estimated nor swapped. Furthermore, no PCA is applied to the data. As the norm of the LBP feature vectors is not the same for all images we scaled those features to unit norm.

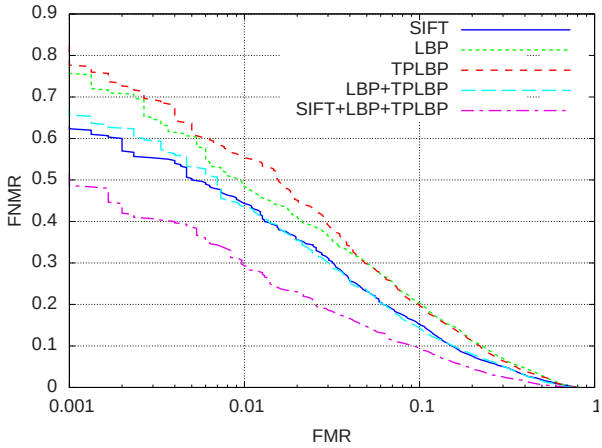
For models selection, the View 1 partition of the LFW database is recommended [Huang et al., 2007]. Using this partition, we obtained that $K_{TM}^{poly}$ works best independently of the chosen type of feature vector. Additionally, we applied our model selection technique to the LFW database. Due to the inhomogeneity of the dataset the model selection technique is only




(a) Interexample task: different EPCRs




(b) Interexample task: different class numbers




(c) Interclass task: different feature vectors

Figure 3: DET curves for LFW Dataset

used to select the pairwise kernel. We obtained by the pair task and the interexample task the same results as by the View 1 partition. It seems that $K_{TM}^{poly}$ will work best in this dataset independently of the chosen feature vector.

In addition, using the idea of the model selection technique in Section 4 we present an interesting analysis about the EPCR by means of the SIFT-based feature vectors. In Figure 3a 42 classes are used. There, it is shown that the performance in the interexample task increases with an increasing EPCR. Especially, we see that a constant EPCR of 5 seems too small. Hence, we conclude that this dataset suffers from a small EPCR (2.3 in average). Fortunately, using an EPCR of 5 and increasing the numbers of classes in the training set increases the performance in the interexample setting, too (see Figure 3b).

Now, we analyze the interclass task by the tenfold cross validation mentioned above. Due to the speed up technique presented in Section 5 we were able to train with large numbers of training pairs. However, if all pairs are used for training, then any training set would consist of approximately 50,000,000 pairs and the training would still need too much time. Hence, whereas in any training set all positive training pairs are used, the negative training pairs are randomly selected in such a way that any training set consist of 2,000,000 pairs. The training of such a model took less than 24 hours. In Figure 3c we present the average DET curves

for feature vectors based on SIFT, LBP, and TPLBP. Inspired by Prince et al. [2011] we added decision function values of these pairwise SVMs and obtained two further DET curves. This led to very good results (see Figure 3c). Furthermore, we concatenated the SIFT, LBP, and TPLBP feature vectors. Surprisingly, the training of some of those models needed longer than a week. Therefore, we do not present these results.

In Table 6.3 the mean equal error rate (EER) and standard error of the mean (SEM) for several types of feature vectors are provided. Note, that many of our results are state of the art or even better. The current state of the art can be found on the homepage of Huang et al. [2007] and in the publication of Prince et al. [2011]. If only SIFT-based feature vectors are used, then the best result is $0.125 \pm 0.0040$ (EER$\pm$SEM). Pairwise SVMs achieve the same EER but a higher SEM $0.1252 \pm 0.0062$. If we add the decision function values corresponding to the LBP and TPLBP feature vectors, then our result $0.1210 \pm 0.0046$ is slightly worse compared to $0.1050 \pm 0.0051$. One possible reason for this fact might be that we did not swap the pose. Finally, for the added decision function values, our performance $0.0947 \pm 0.0057$ is significantly better than $0.0993 \pm 0.0051$. Furthermore, it is worth noting that our standard errors of the mean are comparable to the other presented learning algorithms although most of them use a PCA to reduce noise and dimension of the feature vectors. Only for the SIFT based feature vectors our SEM is larger. Note that the commercial system uses outside training data.

| | | SIFT | LBP | TPLBP | L+T | S+L+T | CS |
|---|---|---|---|---|---|---|---|
| Pairwise | Mean | 0.1252 | 0.1497 | 0.1452 | 0.1210 | 0.0947 | - |
| SVM | SEM | 0.0062 | 0.0052 | 0.0060 | 0.0046 | 0.0057 | - |
| State of | Mean | 0.1250 | 0.1267 | 0.1630 | 0.1050 | 0.0993 | 0.0870 |
| the Art | SEM | 0.0040 | 0.0055 | 0.0070 | 0.0051 | 0.0051 | 0.0030 |

Table 3: EER and SEM for LFW Dataset. Abbreviations: S=SIFT, L=LBP, T=TPLBP, +=adding decision function values, CS=Commercial system `face.com r2011b`

# 7 Final Remarks

In this paper we suggested the SVM framework for handling pairwise classification problems. We analyzed two approaches to enforce the symmetry of the obtained classifiers. To the best of our knowledge, we give the first proof that symmetry is indeed achieved. Then, we prove that for each parameter set of one approach there is a corresponding parameter set of the other one such that both approaches lead to the same classifier. Additionally, we showed that the approach based on projections leads to shorter training times.

Although the number of parameters might be a pitfall of pairwise SVMs, its coincident flexibility might be a strength. To handle this pitfall a technique for model selection is proposed. We discussed details of the implementation of a pairwise SVM solver and presented numerical results. Those results show that pairwise SVMs are capable of successfully treating large scale pairwise classification problems and that the heuristic model selection technique leads to robust and promising results. Furthermore, we showed that pairwise SVMs compete very well for a real world dataset.

We would like to underline that some of the discussed techniques could be transferred to other approaches for solving pairwise classification problems. For example, it is easy to apply most of the results to One Class Support Vector Machines [Schölkopf et al., 1999, Tax and Duin, 2004]. Moreover, the presented model selection technique can be used for other pairwise learning algorithms.

# Acknowledgments

We would like to thank the unknown referees for their feedback on a related unpublished manuscript.

# References

Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, March 2009. ISSN 1532-4435.

Aharon Bar-Hillel and Daphna Weinshall. Learning distance function by coding similarity. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 65–72, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.

Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Boosting margin based distance functions for clustering. In *In Proceedings of the Twenty-First International Conference on Machine Learning*, pages 393–400, 2004a.

Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Learning distance functions for image retrieval. volume 2, pages II–570–II–577 Vol.2, June 2004b.

Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21:38–46, January 2005. ISSN 1367-4803.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–26, 2011. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm (August 2011)`.

Marco Gamassi, Massimo Lazzaroni, Mauro Misino, Vincenzo Piuri, Daniele Sana, and Fabio Scotti. *Accuracy and performance of biometric systems*, pages 510 – 515. Institute of electrical and electronics engineers, Piscataway, 2004. ISBN 078038248X.

Tianshi Gao and Daphne Koller. Multiclass boosting with hinge loss based on output coding. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *International Conference on Computer Vision*, pages 498–505, Sep 2009. URL `http://lear.inrialpes.fr/pubs/2009/GVS09 (August 2011)`.

Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. URL `http://vis-www.cs.umass.edu/lfw/ (August 2011)`.

Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:971–987, July 2002. ISSN 0162-8828. URL `http://www.cse.oulu.fi/MVG/Downloads/LBPMatlab (August 2011)`.

P. Jonathon Phillips. Support vector machines applied to face recognition. In *Advances in Neural Information Processing Systems 11*, pages 803–809. MIT Press, 1999.

John C. Platt. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19416-3.

Simon Prince, Peng Li, Yun Fu, Umar Mohammed, and James Elder. Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2011. ISSN 0162-8828.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. Technical report, November 1999.

David M. J. Tax and Robert P. W. Duin. Support vector data description. *Machine Learning*, 54:45–66, January 2004. ISSN 0885-6125.

Jean P. Vert, Jian Qiu, and William Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, 8(Suppl 10):S8, 2007. ISSN 1471-2105.

Liyang Wei, Yongyi Yang, Robert M. Nishikawa, and Miles N. Wernick. Learning of perceptual similarity from expert readers for mammogram retrieval. In *ISBI*, pages 1356–1359, 2006.

Lior Wolf, Tal Hassner, and Yaniv Taigman. Descriptor based methods in the wild. In *Real-Life Images workshop at the European Conference on Computer Vision (ECCV)*, October 2008. URL `http://www.openu.ac.il/home/hassner/projects/Patchlbp (August 2011)`.

Lior Wolf, Tal Hassner, and Yaniv Taigman. Similarity scores based on background samples. In *ACCV (2)*, pages 88–97, 2009.