Journal of Machine Learning Research 13 (2012) 2279-2292 Submitted 8/11; Revised 3/12; Published 8/12

Pairwise Support Vector Machines and their Application to Large Scale Problems

Carl Brunner C.BRUNNER@GMX.NET
Andreas Fischer ANDREAS.FISCHER@TU-DRESDEN.DE
Institute for Numerical Mathematics
Technische Universität Dresden
01062 Dresden, Germany

Klaus Luig LUIG@COGNITEC.COM
Thorsten Thies THIES@COGNITEC.COM
Cognitec Systems GmbH
Grossenhainer Str. 101
01127 Dresden, Germany

Editor: Corinna Cortes

Abstract

Pairwise classification is the task to predict whether the examples a, b of a pair (a, b) belong to the same class or to different classes. In particular, interclass generalization problems can be treated in this way. In pairwise classification, the order of the two input examples should not affect the classification result. To achieve this, particular kernels as well as the use of symmetric training sets in the framework of support vector machines were suggested. The paper discusses both approaches in a general way and establishes a strong connection between them. In addition, an efficient implementation is discussed which allows the training of several millions of pairs. The value of these contributions is confirmed by excellent results on the labeled faces in the wild benchmark.

Keywords: pairwise support vector machines, interclass generalization, pairwise kernels, large scale problems

1. Introduction

To extend binary classifiers to multiclass classification several modifications have been suggested, for example the one against all technique, the one against one technique, or directed acyclic graphs, see Duan and Keerthi (2005), Hill and Doucet (2007), Hsu and Lin (2002), and Rifkin and Klautau (2004) for further information, discussions, and comparisons. A more recent approach used in the field of multiclass and binary classification is pairwise classification (Abernethy et al., 2009; Bar-Hillel et al., 2004a,b; Bar-Hillel and Weinshall, 2007; Ben-Hur and Noble, 2005; Phillips, 1999; Vert et al., 2007). Pairwise classification relies on two input examples instead of one and predicts whether the two input examples belong to the same class or to different classes. This is of particular advantage if only a subset of classes is known for training. For later use, a support vector machine (SVM) that 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 input examples should not influence the classification result (symmetry). A common approach to enforce this symmetry is the use of selected kernels. For pairwise SVMs, another approach was suggested. Bar-Hillel

et al. (2004a) propose the use of training sets with a symmetric structure. We will discuss both approaches to obtain symmetry in a general way. Based on this, we will provide conditions when these approaches lead to the same classifier. Moreover, we show empirically that the approach of using selected kernels is three to four times faster in training.

A typical pairwise classification task arises in face recognition. There, one is often interested in the interclass generalization, where none of the persons in the training set is part of the test set. We will demonstrate that training sets with many classes (persons) are needed to obtain a good performance in the interclass generalization. The training on such sets is computationally expensive. Therefore, we discuss an efficient implementation of pairwise SVMs. This enables the training of pairwise SVMs with several millions of pairs. In this way, for the labeled faces in the wild database, a performance is achieved which is superior to the current state of the art.

This paper is structured as follows. In Section 2 we give a short introduction to pairwise classification and discuss the symmetry of decision functions obtained by pairwise SVMs. Afterwards, in Section 3.1, we analyze the symmetry of decision functions from pairwise SVMs that rely on symmetric training sets. The new connection between the two approaches for obtaining symmetry is established in Section 3.2. The efficient implementation of pairwise SVMs is discussed in Section 4. Finally, we provide performance measurements in Section 5.

The main contribution of the paper is that we show the equivalence of two approaches for obtaining a symmetric classifier from pairwise SVMs and demonstrate the efficiency and good interclass generalization performance of pairwise SVMs on large scale problems.

2. Pairwise Classification

Let X be an arbitrary set and let m training examples $x_i \in X$ with $i \in M := \{1,\dots,m\}$ be given. 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 a *positive pair*. Otherwise, we set $y_{ij} := -1$ and call (x_i,x_j) a *negative pair*.

In pairwise classification the aim is to decide whether the examples of a pair $(a,b) \in X \times X$ belong to the same class or not. In this paper, we will make use of pairwise decision functions $f : X \times X \to \mathbb{R}$. Such a function predicts whether the examples a,b of a pair (a,b) belong to the same class ($f(a,b) > 0$) or not ($f(a,b) < 0$). Note that neither a,b need to belong to the set of training examples nor the classes of a,b need to belong to the classes of the training examples.

A common tool in machine learning are kernels $k : X \times X \to \mathbb{R}$. Let $\mathcal{H}$ denote an arbitrary real Hilbert space with scalar product $\langle\cdot,\cdot\rangle$. For $\phi : X \to \mathcal{H}$,

$$k(s,t) := \langle \phi(s), \phi(t) \rangle$$

defines a *standard* kernel.

In pairwise classification one often uses *pairwise* kernels $K : (X \times X) \times (X \times X) \to \mathbb{R}$. In this paper we assume that any pairwise kernel is symmetric, that is, it holds that

$$K((a,b),(c,d)) = K((c,d),(a,b))$$

for all $a,b,c,d \in X$, and that it is positive semidefinite (Schölkopf and Smola, 2001). For instance,

$$K_D((a,b),(c,d)) := k(a,c) + k(b,d), \tag{1}$$

$$K_T((a,b),(c,d)) := k(a,c) \cdot k(b,d) \tag{2}$$

are symmetric and positive semidefinite. We call K_D *direct sum pairwise kernel* and K_T *tensor pairwise kernel* (cf. Schölkopf and Smola, 2001).

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

$$f(a,b) = f(b,a) \quad \text{for all } a,b \in X.$$

Now, let us assume that $I \subseteq M \times M$ is given. Then, the pairwise decision function f obtained by a pairwise SVM can be written as

$$f(a,b) := \sum_{(i,j)\in I} \alpha_{ij} y_{ij} K((x_i,x_j),(a,b)) + \gamma \tag{3}$$

with bias $\gamma \in \mathbb{R}$ and $\alpha_{ij} \geq 0$ for all $(i,j) \in I$. Obviously, if K_D (1) or K_T (2) are used, then the decision function is not symmetric in general. This motivates us to call a kernel K *balanced* if

$$K((a,b),(c,d)) = K((a,b),(d,c)) \quad \text{for all } a,b,c,d \in X$$

holds. Thus, if a balanced kernel is used, then (3) is always a symmetric decision function. For instance, the following kernels are balanced

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

$$K_{TL}((a,b),(c,d)) := \frac{1}{2}\left(k(a,c)k(b,d)+k(a,d)k(b,c)\right), \tag{5}$$

$$K_{ML}((a,b),(c,d)) := \frac{1}{4}\left(k(a,c)-k(a,d)-k(b,c)+k(b,d)\right)^2, \tag{6}$$

$$K_{TM}((a,b),(c,d)) := K_{TL}((a,b),(c,d)) + K_{ML}((a,b),(c,d)). \tag{7}$$

Vert et al. (2007) call K_{ML} *metric learning pairwise kernel* and K_{TL} *tensor learning pairwise kernel*. Similarly, we call K_{DL}, which was introduced in Bar-Hillel et al. (2004a), *direct sum learning pairwise kernel* and K_{TM} *tensor metric learning pairwise kernel*. For representing some balanced kernels by projections see Brunner et al. (2011).

3. Symmetric Pairwise Decision Functions and Pairwise SVMs

Pairwise SVMs lead to decision functions of the form (3). As detailed above, if a balanced kernel is used within a pairwise SVM, one always obtains a symmetric decision function. For pairwise SVMs which use K_D (1) as pairwise kernel, it has been claimed that any symmetric set of training pairs leads to a symmetric decision function (see Bar-Hillel et al., 2004a). We call a set of training pairs symmetric, if for any training pair (a,b) the pair (b,a) also belongs to the training set. In Section 3.1 we prove the claim of Bar-Hillel et al. (2004a) in a more general context which includes K_T (2). Additionally, we show in Section 3.2 that under some conditions a symmetric training set leads to the same decision function as balanced kernels if we disregard the SVM bias term γ. Interestingly, the application of balanced kernels leads to significantly shorter training times (see Section 4.2).

3.1 Symmetric Training Sets

In this subsection we show that the symmetry of a pairwise decision function is indeed achieved by means of symmetric training sets. To this end, let $I \subseteq M \times M$ be a symmetric index set, in other words if (i, j) belongs to I then (j, i) also belongs to I. Furthermore, we will make use of pairwise kernels K with

$$K((a,b),(c,d)) = K((b,a),(d,c)) \quad \text{for all } a,b,c,d \in X. \tag{8}$$

As any pairwise kernel is assumed to be symmetric, (8) holds for any balanced pairwise kernel. Note that there are other pairwise kernels that satisfy (8), for instance for the kernels given in Equations 1 and 2.

For $I_R, I_N \subseteq I$ defined by $I_R := \{(i, j) \in I | i = j\}$ and $I_N := I \setminus I_R$ let us consider the dual pairwise SVM

$$\begin{aligned} &\min_{\alpha} \; G(\alpha) \\ \text{s. t.} \quad &0 \leq \alpha_{ij} \leq C \quad \text{for all } (i,j) \in I_N \\ &0 \leq \alpha_{ii} \leq 2C \quad \text{for all } (i,i) \in I_R \\ &\sum_{(i,j)\in I} y_{ij}\alpha_{ij} = 0. \end{aligned} \tag{9}$$

with

$$G(\alpha) := \frac{1}{2} \sum_{(i,j),(k,l)\in I} \alpha_{ij}\alpha_{kl}y_{ij}y_{kl}K((x_i,x_j),(x_k,x_l)) - \sum_{(i,j)\in I} \alpha_{ij}.$$

Lemma 1 *If I is a symmetric index set and if* (8) *holds, then there is a solution $\hat{\alpha}$ of* (9) *with $\hat{\alpha}_{ij} = \hat{\alpha}_{ji}$ for all $(i, j) \in I$.*

Proof By the theorem of Weierstrass there is a solution α^* of (9). Let us define another feasible point $\tilde{\alpha}$ of (9) by

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

For easier notation we set $K_{ij,kl} := K((x_i,x_j),(x_k,x_l))$. 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 (8) we further obtain

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

The last equality holds since I is a symmetric training set. Hence, $\tilde{\alpha}$ is also a solution of (9). Since (9) is convex (cf. Schölkopf and Smola, 2001),

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

solves (9) for any $\lambda \in [0, 1]$. Thus, $\hat{\alpha} := \alpha^{1/2}$ has the desired property. ■

Note that a result similar to Lemma 1 is presented by Wei et al. (2006) for Support Vector Regression. They, however, claim that any solution of the corresponding quadratic program has the described property.

Theorem 2 *If I is a symmetric index set and if* (8) *holds, then any solution α of the optimization problem* (9) *leads to a symmetric pairwise decision function $f : X \times X \to \mathbb{R}$.*

Proof For any solution α of (9) let us define $g_\alpha : X \times X \to \mathbb{R}$ 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 $f_\alpha(a,b) = g_\alpha(a,b) + \gamma$ for some appropriate $\gamma \in \mathbb{R}$. If α^1 and α^2 are solutions of (9) then $g_{\alpha^1} = g_{\alpha^2}$ can be derived by means of convex optimization theory. According to Lemma 1 there is always a solution $\hat{\alpha}$ of (9) with $\hat{\alpha}_{ij} = \hat{\alpha}_{ji}$ for all $(i,j) \in I$. Obviously, such a solution leads to a symmetric decision function $f_{\hat{\alpha}}$. Hence, f_α is a symmetric decision function for all solutions α. ■

3.2 Balanced Kernels vs. Symmetric Training Sets

Section 2 shows that one can use balanced kernels to obtain a symmetric pairwise decision function by means of a pairwise SVM. As detailed in Section 3.1 this can also be achieved by symmetric training sets. Now, we show in Theorem 3 that the decision function is the same, regardless whether a symmetric training set or a certain balanced kernel is used. This result is also of practical value, since the approach with balanced kernels leads to significantly shorter training times (see the empirical results in Section 4.2).

Suppose J is a largest subset of a given symmetric index set I satisfying

$$((i,j) \in J \wedge j \neq i) \quad \Rightarrow \quad (j,i) \notin J.$$

Now, we consider the optimization problem

$$\begin{aligned} &\min_{\beta} H(\beta) \\ \text{s.t.} \quad &0 \leq \beta_{ij} \leq 2C \quad \text{for all } (i,j) \in J \\ &\sum_{(i,j)\in J} y_{ij}\beta_{ij} = 0 \end{aligned} \tag{10}$$

with

$$H(\beta) \coloneqq \frac{1}{2} \sum_{(i,j),(k,l)\in J} \beta_{ij}\beta_{kl} y_{ij} y_{kl} \hat{K}_{ij,kl} - \sum_{(i,j)\in J} \beta_{ij}$$

and

$$\hat{K}_{ij,kl} \coloneqq \frac{1}{2}\left(K_{ij,kl} + K_{ji,kl}\right), \tag{11}$$

where K is an arbitrary pairwise kernel. Obviously, $\hat{K}$ is a balanced kernel. For instance, if $K = K_D$ (1) then $\hat{K} = K_{DL}$ (4) or if $K = K_T$ (2) then $\hat{K} = K_{TL}$ (5). The assumed symmetry of K yields

$$\hat{K}_{ij,kl} = \hat{K}_{ij,lk} = \hat{K}_{ji,kl} = \hat{K}_{ji,lk} = \hat{K}_{kl,ij} = \hat{K}_{lk,ij} = \hat{K}_{kl,ji} = \hat{K}_{lk,ji}. \tag{12}$$

Note that (12) holds not only for kernels given by (11) but for any balanced kernel.

Theorem 3 *Let the functions $g_\alpha : X \times X \to \mathbb{R}$ and $h_\beta : X \times X \to \mathbb{R}$ be defined by*

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

where I is a symmetric index set and J is defined as above. Additionally, let K fulfill (8) *and $\hat{K}$ be given by* (11)*. Then, for any solution α^* of* (9) *and for any solution β^* of* (10) *it holds that $g_{\alpha^*} = h_{\beta^*}$.*

Proof By means of convex optimization theory it can be derived that g_α is the same function for any solution α. The same holds for h_β and any solution β. Hence, due to Lemma 1 we can assume that α^* is a solution of (9) with $\alpha^*_{ij} = \alpha^*_{ji}$. For $J_R \coloneqq I_R$ and $J_N \coloneqq J \setminus J_R$ we define $\bar{\beta}$ by

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

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

$$\begin{aligned} (i,j) \in J_N: \qquad \bar{\beta}_{ij}\hat{K}_{ij,kl} &= \frac{\bar{\beta}_{ij}}{2}(K_{ij,kl} + K_{ji,kl}) = \frac{\alpha^*_{ij} + \alpha^*_{ji}}{2}\left(K_{ij,kl} + K_{ji,kl}\right) \\ &= \alpha^*_{ij}K_{ij,kl} + \alpha^*_{ji}K_{ji,kl}, \qquad (13) \\ (i,i) \in J_R: \qquad \bar{\beta}_{ii}\hat{K}_{ii,kl} &= \frac{\bar{\beta}_{ii}}{2}(K_{ii,kl} + K_{ii,kl}) = \alpha^*_{ii}K_{ii,kl}. \end{aligned}$$

Then, $y_{ij} = y_{ji}$ implies

$$h_{\bar{\beta}} = g_{\alpha^*}. \qquad (14)$$

In a second step we prove that $\bar{\beta}$ is a solution of problem (10). By using $y_{kl} = y_{lk}$, the symmetry of K, (13), (12), and the definition of $\bar{\beta}$ one obtains

$$\begin{aligned} &2G(\alpha^*) + 2\sum_{(i,j)\in I} \alpha^*_{ij} \\ &= \sum_{(i,j)\in I} \alpha^*_{ij} y_{ij} \left(\sum_{(k,l)\in J_N} y_{kl}\left(\alpha^*_{kl}K_{ij,kl} + \alpha^*_{lk}K_{ij,lk}\right) + \sum_{(k,k)\in J_R} y_{kk}\alpha^*_{kk}K_{ij,kk} \right) \\ &= \sum_{(i,j)\in J_N \cup J_R} \alpha^*_{ij} y_{ij} \sum_{(k,l)\in J} \bar{\beta}_{kl} y_{kl} \hat{K}_{ij,kl} + \sum_{(i,j)\in J_N} \alpha^*_{ji} y_{ji} \sum_{(k,l)\in J} \bar{\beta}_{kl} y_{kl} \hat{K}_{ji,kl} \\ &= \sum_{(i,j)\in J_N} \bar{\beta}_{ij} y_{ij} \sum_{(k,l)\in J} \bar{\beta}_{kl} y_{kl} \hat{K}_{ij,kl} + \sum_{(i,i)\in J_R} \bar{\beta}_{ii} y_{ii} \sum_{(k,l)\in J} \bar{\beta}_{kl} y_{kl} \hat{K}_{ii,kl} \\ &= 2H(\bar{\beta}) + 2\sum_{(i,j)\in J} \bar{\beta}_{ij}. \end{aligned}$$

Then, the definition of $\bar{\beta}$ implies

$$G(\alpha^*) = H(\bar{\beta}). \qquad (15)$$

Now, let us define $\bar{\alpha}$ by

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

Obviously, $\bar{\alpha}$ is a feasible point of (9). Then, by (8) and (11) we obtain for

$$(k,l) \in J_N: \quad \bar{\alpha}_{kl} K_{ij,kl} + \bar{\alpha}_{lk} K_{ij,lk} = \frac{\beta^*_{kl}}{2}(K_{ij,kl} + K_{ij,lk}) = \beta^*_{kl} \hat{K}_{ij,kl},$$

$$(k,k) \in J_R: \quad \bar{\alpha}_{kk} K_{ij,kk} = \frac{\beta^*_{kk}}{2}(K_{ij,kk} + K_{ij,kk}) = \beta^*_{kk} \hat{K}_{ij,kk}.$$

This, (12), and $y_{kl} = y_{lk}$ yield

$$\begin{aligned} & 2H(\beta^*) + 2 \sum_{(i,j)\in J} \beta^*_{ij} \\ &= \sum_{(i,j)\in J} \beta^*_{ij} y_{ij} \left(\sum_{(k,l)\in J_N} \beta^*_{kl} y_{kl} \frac{1}{2} \left(\hat{K}_{ij,kl} + \hat{K}_{ji,kl} \right) + \sum_{(k,k)\in J_R} \beta^*_{kk} y_{kk} \frac{1}{2} \left(\hat{K}_{ij,kk} + \hat{K}_{ji,kk} \right) \right) \\ &= \frac{1}{2} \sum_{(i,j)\in J} \beta^*_{ij} y_{ij} \left(\sum_{(k,l)\in I} \bar{\alpha}_{kl} y_{kl} \left(K_{ij,kl} + K_{ji,kl} \right) \right). \end{aligned}$$

Then, the definition of $\bar{\alpha}$ provides $\beta^*_{ij} = \bar{\alpha}_{ij} + \bar{\alpha}_{ji}$ for $(i,j) \in J_N$ and $\bar{\alpha}_{ij} = \bar{\alpha}_{ji}$. Thus,

$$2H(\beta^*) + 2 \sum_{(i,j)\in J} \beta^*_{ij} = \sum_{(i,j)\in I} \bar{\alpha}_{ij} y_{ij} \left(\sum_{(k,l)\in I} \bar{\alpha}_{kl} y_{kl} K_{ij,kl} \right) = 2G(\bar{\alpha}) + 2 \sum_{(i,j)\in I} \bar{\alpha}_{ij}$$

follows. This implies $G(\bar{\alpha}) = H(\beta^*)$. Now, let us assume that $\bar{\beta}$ is not a solution of (10). Then, $H(\beta^*) < H(\bar{\beta})$ holds and, by (15), we have

$$G(\alpha^*) = H(\bar{\beta}) > H(\beta^*) = G(\bar{\alpha}).$$

This is a contradiction to the optimality of α^*. Hence, $\bar{\beta}$ is a solution of (10) and $h_{\beta^*} = h_{\bar{\beta}}$ follows. Then, with (14) we have the desired result. ■

4. Implementation

One of the most widely used techniques for solving SVMs efficiently is the sequential minimal optimization (SMO) (Platt, 1999). A well known implementation of this technique is LIBSVM (Chang and Lin, 2011). Empirically, SMO scales quadratically with the number of training points (Platt, 1999). Note that in pairwise classification the training points are the training pairs. If all possible training pairs are used, then the number of training pairs grows quadratically with the number m of training examples. Hence, the runtime of LIBSVM would scale quartically with m. In Section 4.1 we discuss how the costs for evaluating pairwise kernels, which can be expressed by standard kernels, can be drastically reduced. In Section 3 we discussed that one can either use balanced kernels or symmetric training sets to enforce the symmetry of a pairwise decision function. Additionally, we showed that both approaches lead to the same decision function. Section 4.2 compares the needed training times of the approach with balanced kernels and the approach with symmetric training sets.

4.1 Caching the Standard Kernel

In this subsection balanced kernels are used to enforce the symmetry of the pairwise decision function. Kernel evaluations are crucial for the performance of LIBSVM. If we could cache the whole kernel matrix in RAM 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. Note that training sets with 500 training examples already result in 125,250 training pairs. Now, we describe how the costs of kernel evaluations can be drastically reduced. For example, let us select the kernel K_{TL} (5) with an arbitrary standard kernel. For a single evaluation of K_{TL} the standard kernel has to be evaluated four times with vectors of X. Afterwards, four arithmetic operations are needed.

It is easy to see that each standard kernel value is used for evaluating many different elements of the kernel matrix. In general, it is possible to cache the standard kernel values for all training examples. For example, to cache the standard kernel values for 10,000 examples one needs 400MB. Thus, each kernel evaluation of K_{TL} costs four arithmetic operations only. This does not depend on the chosen standard kernel.

Table 1 compares the training times with and without caching the standard kernel values. For these measurements examples from the double interval task (cf. Section 5.1) are used where each class is represented by 5 examples, K_{TL} is chosen as pairwise kernel with a linear standard kernel, a cache size of 100MB is selected for caching pairwise kernel values, and all possible pairs are used for training. In Table 1a the training set of each run consists of $m = 250$ examples of 50 classes with different dimensions n. Table 1b shows results for different numbers m of examples of dimension $n = 500$. The speedup factor by the described caching technique is up to 100.

Dimension n of examples	Standard kernel (time in mm:ss) not cached	cached
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 m of examples	Standard kernel (time in hh:mm) not cached	cached
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 m of examples

Table 1: Training time with and without caching the standard kernel

4.2 Balanced Kernels vs. Symmetric Training Sets

Theorem 3 shows that pairwise SVMs which use symmetric training sets and pairwise SVMs with balanced kernels lead to the same decision function. For symmetric training sets the number of training pairs is nearly doubled compared to the number in the case of balanced kernels. Simultaneously, (11) shows that evaluating a balanced kernel is computationally more expensive compared to the corresponding non balanced kernel.

Table 2 compares the needed training time of both approaches. There, examples from the double interval task (cf. Section 5.1) of dimension $n = 500$ are used where each class is represented by 5 examples, K_T and its balanced version K_{TL} with linear standard kernels are chosen as pairwise kernel, a cache size of 100MB is selected for caching the pairwise kernel values, and all possible pairs are used for training. It turns out, that the approach with balanced kernels is three to four times faster than using symmetric training sets. Of course, the technique of caching the standard kernel values as described in Section 4.1 is used within all measurements.

Number m of examples	Symmetric training set (t in hh:mm)	Balanced 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 2: Training time for symmetric training sets and for balanced kernels

5. Classification Experiments

In this section we will present results of applying pairwise SVMs to one synthetic data set and to one real world data set. Before we come to those data sets in Sections 5.1 and 5.2 we introduce K_{TL}^{lin} and K_{TL}^{poly}. Those kernels denote K_{TL} (5) with linear standard kernel and homogenous polynomial standard kernel of degree two, respectively. The kernels K_{ML}^{lin}, K_{ML}^{poly}, K_{TM}^{lin}, and K_{TM}^{poly} are defined analogously. In the following, detection error trade-off curves (DET curves cf. Gamassi et al., 2004) will be used to measure the performance 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.

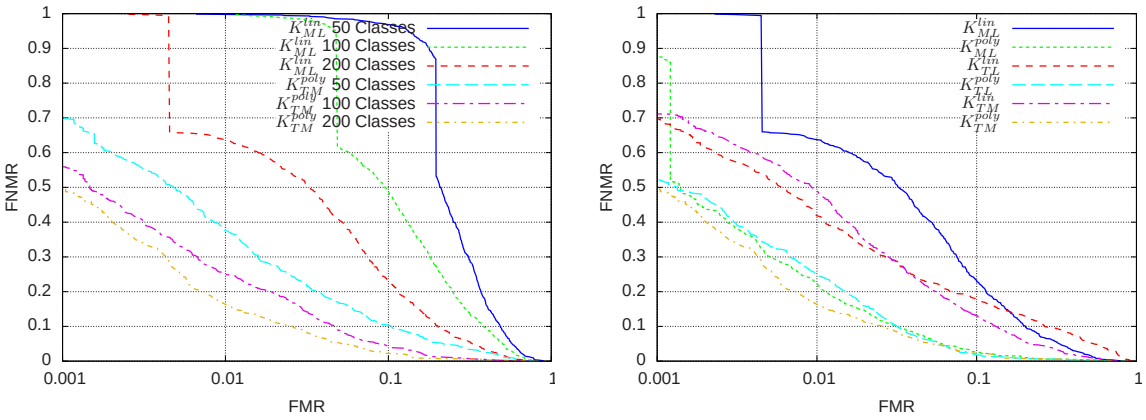
5.1 Double Interval Task

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

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

The class c of such an example is given 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.

For our measurements we selected $n = 500$ and tested all kernels in (4)–(7) with a linear standard kernel and a homogenous polynomial standard kernel of degree two, respectively. We created a test set consisting of 750 examples of 50 classes so that each class is represented by 15 examples. Any training set was generated in such a way that the set of classes in the training set is disjoint from the



(a) Different class numbers in training

(b) Different kernels for 200 classes in training

Figure 1: DET curves for double interval task

set of classes in the test set. We created training sets consisting of 50 classes and different numbers of examples per class. For training all possible training pairs were used.

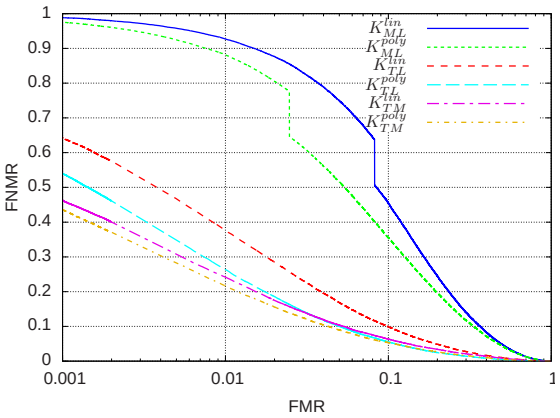
We observed that an increasing number of examples per class improves the performance independently of the other parameters. As a trade-off between the needed training time and performance of the classifier, we decided to use 15 examples per class for the measurements. Independently of the selected kernel, a penalty parameter C of 1,000 turned out to be a good choice. The kernel K_{DS} led to a bad performance regardless of the standard kernel chosen. Therefore, we omit results for K_{DS}.

Figure 1a shows that an increasing number of classes in the training set improves the performance significantly. This holds for all kernels mentioned above. Here, we only present results for K^{lin}_{ML} and K^{poly}_{TM}. Figure 1b shows the DET curves for different kernels where the training set consists of 200 classes. In particular, any of the pairwise kernels which uses a homogeneous polynomial of degree 2 as standard kernel leads to better results than its corresponding counterpart with a linear standard kernel. For FMRs smaller than 0.07 K^{poly}_{TM} leads to the best results, whereas for larger FMRs the DET curves of K^{poly}_{ML}, K^{poly}_{TL}, and K^{poly}_{TM} intersect.

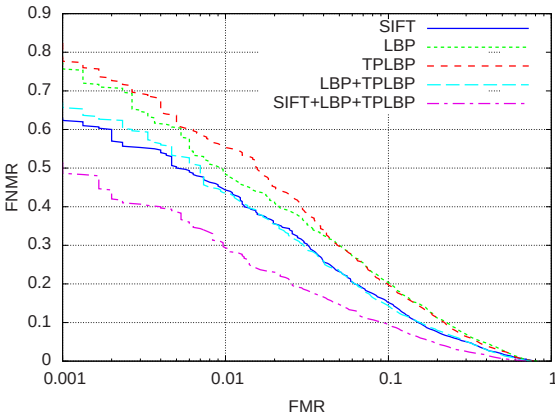
5.2 Labeled Faces in the Wild

In this subsection we will present results of applying pairwise SVMs to the labeled faces in the wild (LFW) data set (Huang et al., 2007). This data set consists of 13,233 images of 5,749 persons. Several remarks on this data set are in order. Huang et al. (2007) suggest two protocols for performance measurements. Here, the unrestricted protocol is used. This protocol is a fixed tenfold cross validation where each test set consists of 300 positive pairs and 300 negative pairs. Moreover, any person (class) in a training set is not part of the corresponding test set.

There are several feature vectors available for the LFW data set. For the presented measurements we mainly followed Li et al. (2012) 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. For this, the aligned images are cropped to 80×150 pixels and are then normalized by passing them through a log function (cf. Li et al., 2012). Afterwards, the



(a) View 1 partition, different kernels, added up decision function values of SIFT, LPB, and TPLBP feature vectors



(b) Unrestricted protocol, K_{TM}^{poly}, different feature vectors, "+" stands for adding up the corresponding decision function values

Figure 2: DET curves for LFW data set

local binary patterns (LBP) (Ojala et al., 2002) and three-patch LBP (TPLBP) (Wolf et al., 2008) are extracted. In contrast to Li et al. (2012), the pose is neither estimated nor swapped and no PCA is applied to the data. As the norm of the LBP feature vectors is not the same for all images we scaled them to Euclidean norm 1.

For model selection, the View 1 partition of the LFW database is recommended (Huang et al., 2007). Using all possible pairs of this partition for training and for testing, we obtained that a penalty parameter C of 1,000 is suitable. Moreover, for each used feature vector, the kernel K_{TM}^{poly} leads to the best results among all used kernels and also if sums of decision function values belonging to SIFT, LBP, and TPLBP feature vectors are used. For example, Figure 2a shows the performance of different kernels, where the decision function values corresponding to SIFT, LBP, and TPLBP feature vectors are added up.

Due to the speed up techniques presented in Section 4 we were able to train with large numbers of training pairs. However, if all pairs were 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 were used, the negative training pairs were randomly selected in such a way that any training set consists of 2,000,000 pairs. The training of such a model took less than 24 hours on a standard PC. In Figure 2b we present the average DET curves obtained for K_{TM}^{poly} and feature vectors based on SIFT, LBP, and TPLBP. Inspired by Li et al. (2012), we determined two further DET curves by adding up the decision function values. This led to very good results. 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 3 the mean equal error rate (EER) and the standard error of the mean (SEM) obtained from the tenfold cross validation are provided for several types of feature vectors. Note, that many of our results are comparable to the 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 Li et al. (2012). If only SIFT-based feature vectors are used, then the best known result is 0.125 ± 0.0040 (EER $\pm$ SEM). With

pairwise SVMs we achieved the same EER but a slightly higher SEM 0.1252 ± 0.0062. If we add up the decision function values corresponding to the LBP and TPLBP feature vectors, then our result 0.1210 ± 0.0046 is worse compared to the state of the art 0.1050 ± 0.0051. One possible reason for this fact might be that we did not swap the pose. Finally, for the added up decision function values corresponding to SIFT, LBP and TPLBP feature vectors, our performance 0.0947 ± 0.0057 is better than 0.0993 ± 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. Note that the results of the commercial system are not directly comparable since it uses outside training data (for reference see Huang et al., 2007).

		SIFT	LBP	TPLBP	L+T	S+L+T	CS
Pairwise	Mean EER	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 EER	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: Mean EER and SEM for LFW data set. S=SIFT, L=LBP, T=TPLBP, +=adding up decision function values, CS=Commercial system `face.com r2011b`

6. Final Remarks

In this paper we suggested the SVM framework for handling large pairwise classification problems. We analyzed two approaches to enforce the symmetry of the obtained classifiers. To the best of our knowledge, we gave the first proof that symmetry is indeed achieved. Then, we proved 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 balanced kernels leads to shorter training times.

We discussed details of the implementation of a pairwise SVM solver and presented numerical results. Those results demonstrate that pairwise SVMs are capable of successfully treating large scale pairwise classification problems. Furthermore, we showed that pairwise SVMs compete very well for a real world data set.

We would like to underline that some of the discussed techniques could be transferred to other approaches for solving pairwise classification problems. For example, most of the results can be applied easily to One Class Support Vector Machines (Schölkopf et al., 2001).

Acknowledgments

We would like to thank the unknown referees for their valuable comments and suggestions.

References

J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, 2009.

A. Bar-Hillel and D. Weinshall. Learning distance function by coding similarity. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pages 65–72. ACM, 2007.

A. Bar-Hillel, T. Hertz, and D. Weinshall. Boosting margin based distance functions for clustering. In C. E. Brodley, editor, *In Proceedings of the 21st International Conference on Machine Learning (ICML '04)*, pages 393–400. ACM, 2004a.

A. Bar-Hillel, T. Hertz, and D. Weinshall. Learning distance functions for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, volume 2, pages 570–577. IEEE Computer Society Press, 2004b.

A. Ben-Hur and W. Stafford Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(1):38–46, 2005.

C. Brunner, A. Fischer, K. Luig, and T. Thies. Pairwise kernels, support vector machines, and the application to large scale problems. Technical Report MATH-NM-04-2011, Institute of Numerical Mathematics, Technische Universität Dresden, October 2011. URL `http://www.math.tu-dresden.de/~fischer`.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–26, 2011. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm (August 2011)`.

K. Duan and S. S. Keerthi. Which is the best multiclass SVM method? An empirical study. In N. C. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Proceedings of the 6th International Workshop on Multiple Classifier Systems*, pages 278–285. Springer, 2005.

M. Gamassi, M. Lazzaroni, M. Misino, V. Piuri, D. Sana, and F. Scotti. Accuracy and performance of biometric systems. In *Proceedings of the 21th IEEE Instrumentation and Measurement Technology Conference (IMTC '04)*, pages 510–515. IEEE, 2004.

M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proceedings of the 12th International Conference on Computer Vision (ICCV '09)*, pages 498–505, 2009. URL `http://lear.inrialpes.fr/pubs/2009/GVS09 (August 2011)`.

S. I. Hill and A. Doucet. A framework for kernel-based multi-category classification. *Journal of Artificial Intelligence Research*, 30(1):525–564, 2007.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

G. B. Huang, M. Ramesh, T. Berg, and E. 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)`.

P. Li, Y. Fu, U. Mohammed, J. H. Elder, and S. J. D. Prince. Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:144–157, 2012.

T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002. URL `http://www.cse.oulu.fi/MVG/Downloads/LBPMatlab (August 2011)`.

P. J. Phillips. Support vector machines applied to face recognition. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 803–809. MIT Press, 1999.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, 1999.

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

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computations*, 13(7):1443–1471, 2001.

J. P. Vert, J. Qiu, and W. Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, 8(Suppl 10):S8, 2007.

L. Wei, Y. Yang, R. M. Nishikawa, and M. N. Wernick. Learning of perceptual similarity from expert readers for mammogram retrieval. In *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 1356–1359. IEEE, 2006.

L. Wolf, T. Hassner, and Y. Taigman. Descriptor based methods in the wild. In *Faces in Real-Life Images Workshop at the European Conference on Computer Vision (ECCV '08)*, 2008. URL `http://www.openu.ac.il/home/hassner/projects/Patchlbp (August 2011)`.

L. Wolf, T. Hassner, and Y. Taigman. Similarity scores based on background samples. In *Proceedings of the 9th Asian Conference on Computer Vision (ACCV '09)*, volume 2, pages 88–97, 2009.