H-0243 (H0611-002) November 8, 2006
Computer Science

IBM Research Report

Superior Multi-Class Classification through a Margin-Optimized Single Binary Problem

Ran El-Yaniv, Dmitry Pechyony
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, Israel

Elad Yom-Tov
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Superior Multi-Class Classification Through a Margin-Optimized Single Binary Problem

Ran El-Yaniv RANI@CS.TECHNION.AC.IL
Dmitry Pechyony PECHYONY@CS.TECHNION.AC.IL
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, Israel
Elad Yom-Tov YOMTOV@IL.IBM.COM
IBM Haifa Research Lab
Haifa 31905, Israel

Editor: TBD

Abstract

The problem of multiclass-to-binary reductions in the context of classification with kernel machines continues to attract considerable attention. Indeed, the current understanding of this problem is rather limited. Despite the multitude of proposed solutions no single method is known to be consistently superior to others. We developed a new multi-class classification method that reduces the multi-class problem to a single binary classifier (SBC). Our method constructs the binary problem by embedding smaller binary problems into a single space. We show that the construction of a good embedding, which allows for large margin classification, can be reduced to the task of learning linear combinations of kernels. We observe that a known margin generalization error-bound for standard binary classification applies to our construction. Our empirical examination of the new method indicates that it can outperform one-vs-all, all-pairs and the error-correcting output coding scheme.

Keywords: multiclass classification, support vector machines, multiple kernel learning

1. Introduction

The widespread practice of employing support vector machines (SVMs) in applications provides a major incentive for the ongoing study of the *multiclass-to-binary reduction* problem to enable the use of binary classifiers for multiclass problems. However, despite numerous ideas on how SVMs can be applied to multi-class classification, the understanding of multi-class reductions appears to be somewhat limited, both theoretically and empirically. The confusion surrounding this problem has only increased with the availability of increasingly clever and sophisticated solutions, whose authors indicate that there is much to gain by using their approaches, but often without providing sufficient comparisons to other available methods.

Currently, the simplest multiclass-to-binary reduction method is the 'one-vs-all', referred to in this paper as OVA.[1]. Two other well-known reductions are the 'all-pairs' approach, based on Friedman (1996) (a.k.a. 'one-vs.-one') and the 'error-correcting output coding' (ECOC) framework pioneered by Sejnowski and Rosenberg (1987) and Dietterich and Bakiri (1995). One of the

1. OVA is also often called 'one-vs-rest'.

first comprehensive comparisons of multi-class reduction methods was performed by Hsu and Lin (2002). They claimed that the all-pairs approach is superior to other methods. Rifkin and Klautau's prominent paper (Rifkin and Klautau, 2004) later presented an in-depth critical assessment of many previous multi-class papers (including the Hsu and Lin paper). The authors stated that OVA is not inferior to all-pairs and ECOC, provided that adequate efforts are devoted to hyper-parameter tuning. Despite the compelling arguments made in the Rifkin and Klautau paper for OVA, there remains an ongoing debate on the relative effectiveness of these three methods.

A lesser-known approach for solving multi-class problems via binary classification is the *Single Binary Classifier* reduction (henceforth, SBC). From our point of view, any multi-class method that relies on a single, standard binary (soft) classifier is an SBC method. SBC reductions can be obtained by embedding the original problem in a higher-dimensional space consisting of the original features, as well as several other dimensions determined by fixed vectors, termed here *extension features*. This embedding is implemented by replicating the training set points so that each original point is concatenated with each of the extension features' vectors. The *binary* labels of the replicated points are set to maintain a particular structure in the extended space. This construction results in an instance of an artificial binary problem, which is fed to a binary learning algorithm that outputs a single soft binary classifier. To classify a new point, the point is replicated and extended similarly and the resulting replicas are fed to the soft binary classifier that generates a number of *signals*, one for each replica. The class is determined as a function of these signals.

The idea of SBC reductions through dimensionality expansion and replication has existed in several rudimentary forms for quite some time. As far as we know, the first documented SBC reduction is the *Kesler Construction* (see Duda and Hart, 1973, Sec. 5.12.1). A different type of SBC reduction is the *single call* method, presented by Allwein et al. (2000), where the embedding is determined via an error correcting matrix as in ECOC. Anguita et al. (2004) reviewed a recent SBC reduction based on orthogonal feature extensions, and concluded that its SBC method is not inferior to OVA and all-pairs.

In this paper, we propose a new type of kernel SBC (henceforth SBC-KERNEL), where instead of using explicit extensions in feature space, we utilize implicit kernel transformations in Hilbert space. A different transformation is used for each class and these transformations are constructed to increase the margin of the resulting binary problem where the entire multi-class problem is embedded. This SBC-KERNEL method is posed and derived as a kernel optimization problem using the SVM objective function.

We present a comparative study of the proposed SBC-KERNEL method, OVA, all-pairs, and ECOC, as well as three previous SBC methods. These results demonstrate impressive performance of SBC-KERNEL relative to the other algorithms, whose results are gained at a higher computational cost. Additionally, we observe that the recent risk bound for kernel machines with learned kernels (Srebro and Ben-David, 2006) can be extended to our setting, showing that SBC-KERNEL indeed generalizes well in cases where both the number of classes and the empirical margin error are small.

2. On Some Known Multi-Class to Binary Reductions

Let $S = \{(x_i, y_i)\}_{i=1}^m$ be a training set of m examples, where x_i are points in some d-dimensional space $\mathcal{X}$ and each y_i is a label in $\mathcal{Y} = \{1, \ldots, c\}$. A *multi-class classifier* h is any function $h : \mathcal{X} \to \mathcal{Y}$. Our goal in multi-class classification is to generate a good multi-class classifier, based on the training set. We measure performance via the standard 0/1-loss function and are interested in a low

average error over out-of-sample examples. We are concerned with *multiclass-to-binary reductions*, which are methods that solve multi-class classification through the use of binary classifiers such as SVMs. The following sections describe several such reductions.

2.1 One-Vs-All

The OVA is one of the simplest methods for multi-class to binary reduction. It uses the original data relabeled in a binary form to train c soft classifiers. Each classifier is trained to distinguish one of the classes from the rest. The multi-class label of a new data point is predicted by first having each of the binary classifiers classify the point. The index of the classifier with the maximal response is chosen as the predicted label. It is hard to trace the exact origin of OVA, but early references date back to Duda and Hart (1973).

2.2 All-Pairs

In this method (Friedman, 1996), we train one binary classifier for each pair of classes. To classify a new instance, we run a majority vote among all binary classifiers. The advantage of the all-pairs method is that it generates the easiest and most natural binary problems of all known methods. However, a weakness of this method is that there are often irrelevant binary classifiers that participate in the votes.

2.3 Error-Correcting Output Coding

In the error-correcting output coding (ECOC) framework, each of the c given classes is assigned a unique binary vector (called a *codeword*) over $\{\pm 1\}$ of length ℓ. This collection of c codewords forms a $c \times \ell$ *coding matrix* M, whose ℓ columns define ℓ binary partitions of the c classes. Given a training set $S = \{(x_i, y_i)\}$, ℓ binary classifiers are trained. The jth classifier f_j is assigned a unique binary partition defined by the jth column of M and is trained using a training set $\{(x_i, M(i,j))\}$. After the learning process is complete, whenever an unseen point x is given, it is classified by all binary classifiers. This results in a vector $\overline{f}(x) = (f_1(x), \ldots, f_\ell(x))$ with $f_j(x)$ being the (binary) response of the jth classifier. The point x is assigned to the class whose matrix row is "closest" to $\overline{f}(x)$. This class assignment mechanism is called *decoding*. The basic ECOC scheme uses a Hamming-based decoding, where the distance between $\overline{f}(x)$ and the rows of M is computed using the Hamming distance. The ECOC technique was pioneered by Sejnowski and Rosenberg (1987) and Dietterich and Bakiri (1995) and was further developed by Allwein et al. (2000).

2.4 Single Binary Classifier (SBC) Reductions

A large family of SBC reductions can be described as follows: Let M be a $c \times \ell$ matrix of *feature extensions*; the ith row of M is denoted by M_i. In the preprocessing stage, we construct c different copies of each training example x_i, where the rth copy of x_i is $z_{i,r} = x_i \circ M_r$, which is the extension of the row vector x_i with the row vector M_r. The resulting set of new instances $\{z_{i,r}\}$, $i = 1, \ldots, m$, $r = 1, \ldots, c$, are assigned binary labels as follows: for each i and r, the instance $z_{i,r}$ is labeled by $y_{i,r} = +1$ iff $y_i = r$ (i.e., the original label y_i of x_i is r). Otherwise, $z_{i,r}$ is labeled by $y_{i,r} = -1$. The resulting binary-labeled set $S' = \{(z_{i,r}, y_{i,r})\}$ is of size cm and each instance (excluding the label) has $d + \ell$ dimensions. In the second stage of binary learning, we apply a standard learning algorithm (e.g., SVM) on the training set S' and the outcome is a soft binary

classifier h_2. To determine a label (in $\mathcal{Y}$) of a new instance x, we generate c copies of x, where the rth copy is $z_r = x \circ M_r$. The label we predict is $\operatorname{argmax}_r h_2(z_r)$.

A special case of this construction is the SBC reduction proposed in Anguita et al. (2004), where the matrix M is taken to be the $c \times c$ identity matrix. We refer to this reduction as SBC-IDENTITY. Another special case is the SBC-SINGLE method, obtained by taking the column $(1, 2, \ldots, c)^T$ as the feature extensions matrix. In other words, a single feature is concatenated to the data such that the rth "replication" of x_i, is $z_{i,r} = x_i \circ r$. Binary labels are assigned to this data exactly as described above.

For example, suppose that $\mathcal{Y} = \{1, 2, 3\}$ and the training set consists of the following three labeled examples: $\{(x_1, 1), (x_2, 2), (x_3, 3), (x_4, 2)\}$. Then, in the case of SBC-IDENTITY, the feature extension matrix is

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

the resulting labeled training set for the binary SBC-IDENTITY problem is

$$\begin{array}{ll}
z_{1,1} = x_1 \circ (1,0,0) & y_{1,1} = +1 \\
z_{1,2} = x_1 \circ (0,1,0) & y_{1,2} = -1 \\
z_{1,3} = x_1 \circ (0,0,1) & y_{1,3} = -1 \\
z_{2,1} = x_2 \circ (1,0,0) & y_{2,1} = -1 \\
z_{2,2} = x_2 \circ (0,1,0) & y_{2,2} = +1 \\
z_{2,3} = x_2 \circ (0,0,1) & y_{2,3} = -1 \\
z_{3,1} = x_3 \circ (1,0,0) & y_{3,1} = -1 \\
z_{3,2} = x_3 \circ (0,1,0) & y_{3,2} = -1 \\
z_{3,3} = x_3 \circ (0,0,1) & y_{3,3} = +1 \\
z_{4,1} = x_4 \circ (1,0,0) & y_{3,1} = -1 \\
z_{4,2} = x_4 \circ (0,1,0) & y_{3,2} = +1 \\
z_{4,3} = x_4 \circ (0,0,1) & y_{3,3} = -1
\end{array}$$

SBC-SINGLE is also a special case of the general 'single-call' SBC reduction of Allwein et al. (2000). Here, given a $c \times \ell$ (ECOC) coding matrix M, each training example (x_i, y_i) is replicated ℓ times to create ℓ new training examples of the form $((x_i, s), M(y_i, s))$, where $M(y_i, s)$ is a binary label. Using this training set, one induces a binary classifier denoted by h_2. For classifying a new point x, we similarly replicate it ℓ times, $z_i = x \circ i$, $i = 1, \ldots, \ell$, and apply h_2 on each of the ℓ instances. As in ECOC, the resulting vector of (soft) classifications $(h_2(z_1), \ldots, h_2(z_\ell))$ is matched to the closest codeword (row) in M to determine the label. The matching can be done using a Euclidian norm (if h_2 is a soft classifier) or a Hamming distance (if h_2 is a hard classifier). We term this SBC reduction SBC-ECOC. Notice that SBC-SINGLE is a special case of SBC-ECOC applied with a matrix M that is the $c \times c$ identity matrix. The SBC-ECOC construction adds a single attribute to each example and replicates it ℓ times. This is in contrast to the family of SBC reductions we describe above, which also use a (feature extension) matrix, extend each example by ℓ additional binary features, and replicate each example c times.

2.5 Single-Machine SVM Constructions

Unlike the SBC approach, *single-machine* constructions typically modify the standard SVM optimization problem to include c separate soft classifiers simultaneously, one for each class. Each

example is labeled according to the classifier giving maximal soft classification. See Section 3.1 in (Rifkin and Klautau, 2004) for a detailed survey of early approaches to single-machine SVM construction.

We note here that recent approaches to learning with structured outputs (e.g., see Tsochantaridis et al. (2005)) and an exponential family formalism of Canu and Smola (2006) also fall into the category of single machine SVM constructions. While these papers do not include any in-depth treatment of the multi-class to binary problem, the instantiations of their approaches have some similarities to our developments. We discuss this issue further in Section 3 (Remarks 1 and 2).

3. Learning SBC Kernel Reductions

Instead of using explicit feature extensions in SBC reductions, as described in Section 2, we propose a general approach that utilizes arbitrary class mappings instead of feature extensions. Similar to 'standard' SBC reductions, each training example x_i is replicated c times and the rth copy of x_i is $z_{i,r} = \phi_r(x_i)$, where $\phi_r(\cdot)$ is an arbitrary transformation corresponding to the rth class. The resulting set of new instances $\{z_{i,r}\}$, $i = 1, \ldots, m$, $r = 1, \ldots, c$, are assigned binary labels as in standard SBC reductions (see Section 2.4) and a binary soft classifier $h_2(\cdot)$ is trained. To predict the label (in $\mathcal{Y}$) of a new instance x, we generate c copies of x, where the rth copy is $z_r = \phi_r(x)$. The label we predict is $\operatorname{argmax}_r h_2(z_r)$.

In general, the transformations $\{\phi_r\}_{r=1}^{c}$ can have any form. High quality transformations should generate an easy-to-learn binary problem such that the resulting binary classifier h_2 allows for accurate multi-class predictions. Relying on kernel methods, we can consider *implicit* transformations given in terms of inner products $\phi_r(x_i) \cdot \phi_s(x_j)$, between the transformed instances. These inner products can be specified as entries of an appropriate kernel matrix of size $cm \times cm$.

Our realization of this scheme is derived as follows: Let M be a $c \times \ell$ feature extension matrix as described in Section 2 and denote its rth row by M_r. Let $\{z_{i,r}\}$ be the replicated (and feature extended) training set. Using an RBF kernel, $K_{RBF}(a, b) = \exp\left(-\frac{\|a-b\|_2^2}{2\sigma^2}\right)$, we have

$$K_{RBF}(z_{i,r}, z_{j,s}) = K_{RBF}(x_i, x_j) \cdot \exp\left(\frac{-\|M_r - M_s\|_2^2}{2\sigma^2}\right) \quad . \tag{1}$$

The resulting dual formulation of the (standard) SVM optimization problem is as follows:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^{cm}} \quad 2\boldsymbol{\alpha}^T \mathbf{e} - \boldsymbol{\alpha}^T G(\boldsymbol{K})\boldsymbol{\alpha} \tag{2}$$

$$\text{such that} \quad \boldsymbol{\alpha}^T \mathbf{y} = 0 \tag{3}$$

$$0 \leq \boldsymbol{\alpha} \leq \frac{C}{m} \quad . \tag{4}$$

where $G(\mathbf{K})$ is a $cm \times cm$ matrix whose $((i-1)\cdot c+r, (j-1)\cdot c+s)$th entry is $y_{i,r} y_{j,s} K_{RBF}(z_{i,r}, z_{j,s})$. We thus have a larger $cm \times cm$ kernel matrix $\mathbf{K} = \mathbf{K}(M)$ whose (u, v) entry, where $u = (i-1)\cdot c+r$ and $v = (j-1) \cdot c + s$, is $K_{RBF}(z_{i,r}, z_{j,s})$.

The binary classifier $h_2(\cdot)$, resulting from the optimization program (2)-(4) has the following form. For any example x the soft classification of its rth copy z_r is

$$h_2(z_r) \stackrel{\Delta}{=} \sum_{j=1}^{m} \sum_{s=1}^{c} y_{j,s} \alpha_{(j-1)\cdot c+s} K_{RBF}(z_{j,s}, z_r) + b. \tag{5}$$

The value of the parameter b is determined using some standard technique from SVM (see Vapnik (1998)).[2]

The matrix $\mathbf{K}(M)$ can be represented as a Kronecker product of two smaller matrices. For any matrices $\mathbf{A}$ and $\mathbf{B}$ of sizes $m \times n$ and $p \times q$, respectively, their *Kronecker product*, denoted $\mathbf{A} \otimes \mathbf{B}$, is the following $mp \times nq$ matrix (Eves, 1980):

$$\mathbf{A} \otimes \mathbf{B} \triangleq \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2n}\mathbf{B} \\ \dots & \dots & \dots & \dots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix} \quad . \tag{6}$$

For matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and a scalar k, we use the following elementary properties of the Kronecker product:

$$\mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) \ , \tag{7}$$

$$k(\mathbf{A} \otimes \mathbf{B}) = \mathbf{A} \otimes (k\mathbf{B}) \ . \tag{8}$$

Let $\mathbf{K}_x$ be an $m \times m$ kernel matrix whose (i, j) entry is $K_{RBF}(x_i, x_j)$. Let $\mathbf{K}_M$ be a $c \times c$ kernel matrix whose (r, s) entry is $K_{RBF}(M_r, M_s)$. From the definition of $\mathbf{K}(M)$ and the definition of the Kronecker product it follows that

$$\mathbf{K}(M) = \mathbf{K}_x \otimes \mathbf{K}_M \ . \tag{9}$$

Our strategy is to take a number n of fixed feature extension matrices $M^{(1)}, \dots, M^{(n)}$ and to search for the best linear combination $\mathbf{K}_{\boldsymbol{\mu}} = \sum_{i=1}^{n} \mu_i \mathbf{K}(M^{(i)})$ that optimizes the SVM objective function. By applying (9) and using properties (7) and (8) of the Kronecker product, we obtain

$$\mathbf{K}_{\boldsymbol{\mu}} = \sum_{i=1}^{n} \mu_i \mathbf{K}(M^{(i)}) = \mathbf{K}_x \otimes \left(\sum_{i=1}^{n} \mu_i \mathbf{K}_{M^{(i)}} \right) \triangleq \mathbf{K}_x \otimes \mathbf{K}_{\mathbf{M}_{\boldsymbol{\mu}}}, \tag{10}$$

where $\mathbf{K}_{\mathbf{M}_{\boldsymbol{\mu}}} \triangleq \sum_{i=1}^{n} \mu_i \mathbf{K}_{M^{(i)}}$ is an $c \times c$ kernel matrix implicitly representing the feature extension matrix that optimizes the SVM objective function. By substituting the value of kernel, defined by (10), to (5) we obtain that the binary classifier $h_2(\cdot)$ has the following form. For any example x the soft classification of its rth copy z_r is

$$h_2(z_r) \triangleq \sum_{j=1}^{m} \sum_{s=1}^{c} y_{j,s} \alpha_{(j-1)\cdot c+s} \sum_{i=1}^{n} \mu_i K_{RBF}(x_j, x) \exp\left(\frac{-\|M_r^{(i)} - M_s^{(i)}\|_2^2}{2\sigma^2} \right) + b. \tag{11}$$

Clearly, different sets of matrices $\{M^{(i)}\}_{i=1}^{n}$ may lead to different realizations. In Section 5, we focus on a particular choice of such matrices that conceptually corresponds to embedding all c one-vs-all binary problems in a single space. The optimization problem we thus face is related to extensive literature on learning the kernel matrix and in particular, to learning linear combinations of matrices. Our derivation of this optimization problem, shown below, is similar to the one presented in Bousquet and Herrmann (2003).

2. One example of setting b is as follows. Let $B = \{(i, r) \mid \alpha_{(i-1)\cdot c+r} > 0\}$. Let $b_{i,r} \triangleq y_{i,r} - \sum_{j=1}^{m} \sum_{s=1}^{c} y_{j,s} \alpha_{(j-1)\cdot c+s} K_{RBF}(z_{i,r}, z_{j,s})$ if $(i, r) \in B$ and zero otherwise. The value of b is set to $\sum_{i=1}^{m} \sum_{r=1}^{c} b_{i,r}/|B|$.

Remark 1 *Tsochantaridis et al (Tsochantaridis et al., 2005, see Equation (7) and Propositions 4 and 5) proposed a kernel-based optimization program for multi-class SVM that is similar to the one we introduce in (2)-(4). Let $G'(\mathbf{K})$ be a $cm \times cm$ matrix whose $((i-1)\cdot c+r,(j-1)\cdot c+s)$ entry is*

$$K_{RBF}(z_{i,r^*(i)},z_{j,s^*(j)}) - K_{RBF}(z_{i,r^*(i)},z_{j,s}) - K_{RBF}(z_{i,r},z_{j,s^*(j)}) + K_{RBF}(z_{i,r},z_{j,s}) \ , \quad (12)$$

where $r^*(i)$ (respectively, $s^*(j)$) is the value of r (respectively, s) such that $y_{i,r}=+1$ ($y_{j,s}=+1$). Tsochantaridis et al. (2005) introduced a combined feature representation $\Psi(x,y)$ of inputs x and outputs y. In an instantiation of $\Psi(x,y)$, defined using a matrix M of feature extensions (see Section 2.4), the optimization problem proposed by Tsochantaridis et al. (2005) becomes

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^{cm}} \quad 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T G'(\boldsymbol{K})\boldsymbol{\alpha} \quad (13)$$

$$\text{such that} \quad 0 \le \boldsymbol{\alpha} \quad (14)$$

$$\textstyle\sum_{(i,r) \text{ such that } r\neq r^*(i)} \alpha_{(i-1)\cdot c+r} \le \frac{C}{m} \ . \quad (15)$$

While the optimization problem (13)-(15) can be considered as a realization of our method, the program (2)-(4) that we utilize (corresponding to standard SVM) is more standard and better explored. In particular, there is a large variety of both exact and approximate algorithmic solutions for the optimization problem (2)-(4).

Remark 2 *The (SVM) single machine construction proposed by Canu and Smola (2006) can also be applied with the kernel (1). In particular, the resulting optimization program is*

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \boldsymbol{\alpha}^T\hat{\mathbf{K}}\boldsymbol{\alpha} + \sum_{i=1}^m \max\left(0, 1-\ell(x_i)\right), \quad (16)$$

where

$$\ell(x_i) \triangleq \min_{r:r\neq r^*(i)} \left(\sum_{j=1}^m \alpha_i \left(K_{RBF}(z_{i,r^*(i)},z_{j,s^*(j)}) - K_{RBF}(z_{i,r},z_{j,s^*(j)}) \right) \right)$$

and $\hat{\mathbf{K}}$ is an $m\times m$ matrix whose (i,j)th entry equals $K_{RBF}(z_{i,r^*(i)},z_{j,s^*(j)})$. This optimization problem (16) can be viewed as a multi-class extension of the Primal SVM formulation by Chapelle (2006).

3.1 Learning Linear Combinations of Basis Kernels

As mentioned in the previous section, our goal is to take a number n of fixed feature extension matrices $M^{(1)},\ldots,M^{(n)}$ and then search for the best linear combination $\mathbf{K}_{\boldsymbol{\mu}} = \sum_{i=1}^n \mu_i \mathbf{K}(M_i)$ that optimizes the SVM objective function. This approach is motivated by the following analysis.

Let $G(\mathbf{K})$ be an $m\times m$ matrix whose (i,j)th entry is $y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)$. Consider the dual formulation of an 1-norm SVM. The optimal dual 'cost' is

$$\mathbf{g}_d(\mathbf{K}) \triangleq \max_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T G(\boldsymbol{K})\boldsymbol{\alpha} \quad (17)$$

$$\text{such that} \quad \boldsymbol{\alpha}^T\mathbf{y} = 0 \quad (18)$$

$$0 \le \boldsymbol{\alpha} \le \frac{C}{m} \ . \quad (19)$$

The corresponding kernelized primal formulation is (Rifkin, 2002)

$$\begin{aligned} \mathbf{g}_p(\mathbf{K}) \triangleq \min_{\mathbf{c}\in\mathbb{R}^m,\boldsymbol{\xi}\in\mathbb{R}^m,b\in\mathbb{R}} \quad & 2C\textstyle\sum_{i=1}^m \xi_i + \mathbf{c}^T\mathbf{K}\mathbf{c} & \\ \text{such that} \quad & y_i\left(\textstyle\sum_{i=1}^m c_i K(x_i,x_j)+b\right) \geq 1-\xi_i & i=1,\ldots,m \\ & \xi_i \geq 0 & i=1,\ldots,m \end{aligned} \tag{20}$$

Our goal is to find a kernel matrix $\mathbf{K}$ that minimizes $\mathbf{g}_p(\mathbf{K})$. Since the strong duality property holds for the optimization problems (17) and (20), $\mathbf{g}_p(\mathbf{K}) = \mathbf{g}_d(\mathbf{K})$ and consequently, $\min_{\mathbf{K}}\{\mathbf{g}_p(\mathbf{K})\} = \min_{\mathbf{K}}\{\mathbf{g}_d(\mathbf{K})\}$. As proved by Lanckriet et al. (2004, see Proposition 15), the function $\mathbf{g}_d(\mathbf{K})$ with the constraints (18) and (19) is convex in $\mathbf{K}$. If we restrict $\mathbf{K}$ to be in a convex closed domain and optimize with respect to $\mathbf{K}$, we can find the global minimum of $\mathbf{g}_d(\mathbf{K})$ using gradient descent methods.

However, a direct optimization over $\mathbf{K}$ can be computationally expensive, since it involves solving semi-definite (SDP) optimization problems. We therefore restrict our attention to kernel matrices that are linear combinations of some fixed set of *basis kernels*, $\{\mathbf{K}^{(j)}\}_{j=1}^n$. Thus, the desired kernel matrix is $\mathbf{K}_{\boldsymbol{\mu}} = \sum_{j=1}^n \mu_j \mathbf{K}^{(j)}$, where the variables to be optimized are $\boldsymbol{\mu} = (\mu_1,\ldots,\mu_n)$. Note that the property $\mathbf{g}_p(\mathbf{K}_{\boldsymbol{\mu}}) = \mathbf{g}_d(\mathbf{K}_{\boldsymbol{\mu}})$ is preserved when using $\mathbf{K}_{\boldsymbol{\mu}}$ instead of $\mathbf{K}$. To ensure that the matrix $\mathbf{K}_{\boldsymbol{\mu}}$ is a valid kernel, namely that $\mathbf{K}_{\boldsymbol{\mu}}$ is a positive semi-definite matrix, we assume that $\mu_i \geq 0$ for all $1 \leq i \leq n$. We also impose the constraint $\sum_{j=1}^n \mu_j \leq R$, for some (arbitrary) R to ensure that the feasible set of kernels lies in a convex closed domain.

The gradient of $\mathbf{g}_d(\mathbf{K})$ is an $m \times 1$ vector and its jth entry is

$$\frac{\partial \mathbf{g}_d(\mathbf{K}_{\boldsymbol{\mu}})}{\partial \mu_i} = -\boldsymbol{\alpha}^T G(\boldsymbol{K}^{(i)})\boldsymbol{\alpha}.$$

Since $\boldsymbol{K}^{(i)}$ is a kernel matrix, it is positive semidefinite. It can be easily verified that the matrix $G(\boldsymbol{K}^{(i)})$ is also positive semi-definite. Hence, for any $1 \leq j \leq n$, $\frac{\partial \mathbf{g}_d(\mathbf{K}_{\boldsymbol{\mu}})}{\partial \mu_i} \leq 0$. Therefore, the function $\mathbf{g}_p(\mathbf{K}_{\boldsymbol{\mu}})$ is monotonically decreasing with μ. Thus, if we start from positive μ_i's and proceed in the direction opposite to the direction of the gradient, they will remain positive throughout the optimization procedure. Hence, we can drop out the constraint $\mu_i \geq 0$ for all $1 \leq i \leq n$. Following Bousquet and Herrmann (2003), who proposed a similar routine, we use a simple gradient descent procedure for finding the optimal $\mathbf{K}^*$:

1. Let $\boldsymbol{\mu} = \boldsymbol{\mu}_0$ be an initial guess of kernel coefficients.
2. Find $\mathbf{g}_d(\mathbf{K}_{\boldsymbol{\mu}})$ by solving (17). Let $\boldsymbol{\alpha}^*$ be the maximizer of (17).
3. Make a gradient step: for all $1 \leq j \leq n$, $\mu_j = \mu_j + (\boldsymbol{\alpha}^*)^T G(\boldsymbol{K}^{(j)})\boldsymbol{\alpha}^*$.
4. Enforce the constraint $\sum_{j=1}^n \mu_j \leq R$.[3]
5. Return to Step 2 unless a termination criterion is reached.

3. If $\sum_j \mu_j > R$ then we normalize $\boldsymbol{\mu}$ such that $\sum_j \mu_j = R$.

4. Error Bound for Large Margin SBC Kernel Reductions

This section shows a large margin error bound for our SBC reductions. This bound is obtained as a simple corollary of the risk bound of Srebro and Ben-David (2006) for kernel machines with learned kernels.

We consider the following setting. Let $\{k_1(\cdot,\cdot),\ldots,k_n(\cdot,\cdot)\}$ be a set of n kernel functions, such that $k_j(x,x) \leq 1$ for any x and $1 \leq j \leq n$. Let $\boldsymbol{\alpha} \stackrel{\triangle}{=} \{\alpha_i\}_{i=1}^m$, $\boldsymbol{\mu} \stackrel{\triangle}{=} \{\mu_j\}_{j=1}^n$ and consider a soft classification of a point x via the hypothesis

$$h_{\boldsymbol{\alpha},\boldsymbol{\mu}}(x) \stackrel{\triangle}{=} \sum_{i=1}^m y_i \alpha_i \sum_{j=1}^n \mu_j k_j(x_i, x). \tag{21}$$

As usual it is assumed that examples $\langle x, y\rangle$ are distributed according to some unknown distribution D. The overall $0/1$ loss of $h_{\boldsymbol{\alpha},\boldsymbol{\mu}}$ is $\ell_0(\boldsymbol{\alpha},\boldsymbol{\mu}) = \Pr_{\langle x,y\rangle\sim D}\{x \cdot h_{\boldsymbol{\alpha},\boldsymbol{\mu}}(x) \leq 0\}$. Let $S = \{\langle x_i, y_i\rangle\}_{i=1}^t$ be a training set of t examples. The empirical margin error of $h_{\boldsymbol{\alpha},\boldsymbol{\mu}}$ is $\hat{\ell}_\gamma(\boldsymbol{\alpha},\boldsymbol{\mu}) \stackrel{\triangle}{=} \frac{|\{i \mid y_i h_{\boldsymbol{\alpha},\boldsymbol{\mu}}(x_i) < \gamma, \langle x_i, y_i\rangle \in S\}|}{t}$.

Theorem 3 (Srebro and Ben-David (2006)) *Suppose that the vector $\boldsymbol{\mu}$ satisfies $\sum_{j=1}^n \mu_j \leq R$. Then with probability at least $1-\delta$ over the random draw of the training set,*

$$\ell_0(\boldsymbol{\alpha},\boldsymbol{\mu}) \leq \hat{\ell}_\gamma(\boldsymbol{\alpha},\boldsymbol{\mu}) + \sqrt{8\frac{2 + n\log\frac{128et^3R}{\gamma^2 n} + 256\frac{R}{\gamma}\log\frac{\gamma et}{8\sqrt{R}}\log\frac{128tR}{\gamma^2} - \log\delta}{t}}. \tag{22}$$

The binary classifier $h_2(\cdot)$, defined in (11), has the form (21).[4] Therefore, we can utilize Theorem 3 to bound the $0/1$ loss of the classifier (11). In what follows we relate the multiclass $0/1$ loss of the SBC reduction to the $0/1$ loss of the underlying binary classifier.

Let $\boldsymbol{\alpha} \stackrel{\triangle}{=} \{\alpha_i\}_{i=1}^{m\cdot c}$. We denote by $(\boldsymbol{\alpha},\boldsymbol{\mu})$ the hypothesis (11) operated with vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$. Let $\ell_M((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle x,y\rangle)$ be the multi-class $0/1$ loss of the SBC classifier, using the hypothesis $(\boldsymbol{\alpha},\boldsymbol{\mu})$ for its binary decisions, over an example $\langle x,y\rangle$, where y is a multi-class label. Let $\{\langle z_r, y_r\rangle\}_{r=1}^c$ be the c replications of x where $z_r = z_r(x)$ is the rth extension of x and y_r is its appropriate binary label. Denote by $\ell_B((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle z_r, y_r\rangle)$ the binary $0/1$ loss of w on the replica $\langle z_r, y_r\rangle$. It is easy to see, from the SBC construction, that

$$\forall\, \langle x,y\rangle,\ \ \ell_M((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle x,y\rangle) \leq \sum_{r=1}^c \ell_B((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle z_r, y_r\rangle)\ . \tag{23}$$

Indeed, if the SBC classifier errs on $\langle x,y\rangle$ (in the multi-class problem), then there is some replica $\langle z_{r_j}, y_{r_j}\rangle$, with $y_{r_j} = -1$, which achieves larger soft classification than the single positive replica of x (i.e., the replica $\langle z_{r_k}, y_{r_k}\rangle$, with $y_{r_k} = 1$). In this case, $(\boldsymbol{\alpha},\boldsymbol{\mu})$ incurs a 0/1 binary loss (of 1) on either z_{r_j} or z_{r_k}. Let $P(x,y)$ be the (unknown) underlying distribution of the data. Let $\ell_M(\boldsymbol{\alpha},\boldsymbol{\mu})$ be the true average 0/1 multi-class error of the SBC classifier. Define $\ell_B(\boldsymbol{\alpha},\boldsymbol{\mu})$ as the true average binary 0/1 loss of $(\boldsymbol{\alpha},\boldsymbol{\mu})$, $\ell_B(\boldsymbol{\alpha},\boldsymbol{\mu}) = \int \frac{1}{c}\sum_{r=1}^c \ell_B((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle z_r, y_r\rangle) dP(x,y)$. This

4. This is under assumption that parameter b in (11) equals zero. This assumption is also used in Srebro and Ben-David (2006) and other papers deriving risk bounds for kernel machines.

definition makes sense because for each test example x, we always construct all the c replicas of x. Using (23) we have

$$\ell_M(\boldsymbol{\alpha},\boldsymbol{\mu}) = \int \ell_M((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle x,y\rangle)dP(x,y) \le \int \sum_{r=1}^{c} \ell_B((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle z_r,y_r\rangle)dP(x,y) \quad (24)$$

$$= c\cdot\int \frac{1}{c}\sum_{r=1}^{c} \ell_B((\boldsymbol{\alpha},\boldsymbol{\mu}),\langle z_r,y_r\rangle)dP(x,y) = c\cdot\ell_B(\boldsymbol{\alpha},\boldsymbol{\mu}). \quad (25)$$

Applying the bound (22) with ℓ_B instead of ℓ_0 and with $t = mc$, we obtain

$$\ell_M(\boldsymbol{\alpha},\boldsymbol{\mu}) \le c\cdot\ell_\gamma(\boldsymbol{\alpha},\boldsymbol{\mu}) + c\cdot\sqrt{8\frac{2+n\log\frac{128e(mc)^3R}{\gamma^2 n}+256\frac{R}{\gamma}\log\frac{\gamma emc}{8\sqrt{R}}\log\frac{128mcR}{\gamma^2}-\log\delta}{mc}}. \quad (26)$$

While this bound is certainly not tight in general, it is useful when the number of classes and the empirical margin binary error are small (in particular, in the realizable case). Note also that the slack term in this bound increases with R. However, with a larger R the search space for optimal $\boldsymbol{\mu}$ also increases. Thus the parameter R expresses a tradeoff between the size of the search space and the value of the slack term in the risk bound.

5. Experiments

As discussed in Section 3, to implement our SBC-KERNEL approach we need to select a set of feature extension matrices. We took c matrices, each of size $c \times c$, where $M^{(r)}$ was taken to be an all-zeros matrix except for a single unit entry, $M^{(r)}(r,r) = 1, r = 1,\ldots,c$. Such a choice of matrices $M^{(r)}$ has the following interpretation.

An SBC reduction with c classes encompasses c binary problems. The rth problem A_r is defined by the rth set of training replicas and aims at separating class r from the other classes. This is also the rth binary problem solved by OVA. The SBC reduction solves all binary problems $\{A_r\}$ simultaneously, in a single joint space, using a single classifier. The success of SBC depends on the relative placement of the problems $\{A_r\}$ in the joint space. The matrix M_r and the coefficient μ_r "separate" the rth problem from the others, by moving the rth problem away from other problems. By optimizing μ_r we attempt to find a good placement of all the problems that increases the overall margin.

5.1 Experimental Procedure

We used five datasets, all from the UCI repository. Their characteristics are summarized in Table 1. Due to the considerable computational load associated with SBC reductions (due to replication), we restricted our experiment to datasets that have a small number of classes. Nominal attributes with t possible values were substituted by t binary features, where the ith binary feature was set to 1 iff the corresponding nominal attribute took the ith possible value. For each feature, its average and standard deviation over the training set was computed, and these were used to normalize the data (training and testing) by subtracting the average and dividing by the standard deviation.

Ten-fold cross-validation (10xCV) was used; namely, in each fold, the union of nine out of ten equally-sized random subsets were used for training, and the tenth for testing. In all our experiments, we used the *SVMTorch* implementation of a binary SVM inducer (Collobert and Bengio, 2001) and

	# Examples	# Features	# Classes
Car	1728	6	4
Page blocks	5473	10	5
Iris	150	4	3
Wine	178	13	3
Vehicle	846	18	3

Table 1: Summary of datasets

applied it with a radial-basis function (RBF) kernel. To optimize the RBF kernel parameters (σ and C) we followed Rifkin and Klautau (2004) and used a simple greedy search via 10xCV *over the training set* as follows. Initial values of σ and C were set to 1. The value of σ was then increased or decreased by a factor of 2 until no improvements were observed for three consecutive attempts. Then, σ was fixed at the best value found and an identical optimization was performed over C.[5]

In addition to our SBC-KERNEL method, we experimented with three other SBC reductions: SBC-SINGLE, SBC-IDENTITY, and SBC-ECOC (using a BCH coding matrix). We also tested the three "classical" reductions: OVA, ALL-PAIRS and ECOC (applied with the same BCH coding matrix). Overall, we tested seven algorithms. Precise descriptions of the known methods appear in Section 2. For all the algorithms tested, we used the same parameter tuning strategy to search for a single best pair of parameters (σ and C) for all the binary classifiers involved in each multi-class application. While this method may favor reductions that utilize a smaller number of binary problems, we believe this is a fair comparison that allocates similar search resources to each algorithm while searching for effective hyper-parameters.

		Car	Page blocks	Iris	Wine	Vehicle	AVERAGE RANK
STANDARD REDUCTIONS	OVA	1.10	3.33	21.33	5.88	25.48	4.8
	ALL-PAIRS	0.76	4.64	24.00	4.71	25.00	5.0
	ECOC	4.36	3.20	6.00	**1.76**	20.48	3.1
SBC REDUCTIONS	SBC-ECOC	3.90	3.42	**4.00**	3.53	20.48	3.4
	SBC-SINGLE	5.64	3.51	66.67	65.29	76.55	6.8
	SBC-IDENTITY	4.36	3.18	6.00	2.35	20.95	3.5
	SBC-KERNEL	**0.70**	**2.96**	**4.00**	2.35	**15.48**	**1.4**

Table 2: 10xCV average test errors rates (%) of seven algorithms on five datasets. Best results for the dataset appear in boldface. Average ranks of the algorithms appear in the last row

We operated SBC-KERNEL with R (arbitrarily) set to square root of the training set size. The termination criterion of the algorithm was chosen to be $\frac{\|\boldsymbol{\mu}_{\text{old}}-\boldsymbol{\mu}_{\text{new}}\|_2}{\|\boldsymbol{\mu}_{\text{old}}\|_2} \leq 0.001$, where $\boldsymbol{\mu}_{\text{old}}$ and $\boldsymbol{\mu}_{\text{new}}$ are the values of $\boldsymbol{\mu}$ at the previous and at the current iteration, respectively.

5. One can consider various ways to improve the optimization routine suggested by Rifkin and Klautau. For example, it is potentially better to jointly optimize over C and σ, but computationally, this would be rather expensive.

5.2 Results

Table 2 presents the errors (%) obtained for each of the seven methods. The best results (lowest errors) in each row appear in boldface. The average *ranks* of the various algorithms appear in the last row of the table. These ranks were computed as averages of row ranks.[6] The best performer in terms of ranks is SBC-KERNEL. The worst performer is SBC-SINGLE. Comparing our results to Rifkin and Klautau (2004) (over the two common datasets `Car` and `Page blocks`), we see that our results for OVA are slightly better, and our results for ALL-PAIRS are similar.

To systematically identify potentially interesting differences between the methods with respect to individual datasets, we followed Rifkin and Klautau (2004) and calculated a 90% bootstrap confidence interval for the differences in performance between *pairs* of algorithms *with respect to individual datasets*. For a dataset with a given train/test partition and two given classifiers c_1 and c_2 (induced on the train set), we drew 10,000 bootstrap samples from the test set. Each bootstrap sample consisted of 10% of the points in the dataset, drawn with equal probability from each of the ten folds. For each sample, we calculated the performance difference of the two classifiers. This difference is a number in $[-100, 100]$, where a negative difference reflects an advantage of c_1 and a positive difference reflects an advantage of c_2. Zero means that the classifiers performed identically over the test set. For a desired confidence level δ (e.g., 90%), we output the confidence interval $[a, b]$ where a is the $(\frac{1-\delta}{2})$-quantile of the 10,000 differences and b is the $(\frac{1+\delta}{2})$-quantile of these differences.[7]

	`Car`	`Page blocks`	`Iris`	`Wine`	`Vehicle`
OVA	$[-1.7, 0.5] \rightarrow$	$[-1.3, 0.4] \rightarrow$	$[-3, 0] \rightarrow$	$[-11.8, 5.9] \rightarrow$	$[-20.2, -4.8] \rightarrow$
ALL-PAIRS	$[-1.2, 0] \rightarrow$	$[-2.9, -0.6] \rightarrow$	$[-40, 0] \rightarrow$	$[-11.8, 5.9] \rightarrow$	$[-20.2, -3.6] \rightarrow$
ECOC	$[-6.4, -1.7] \rightarrow$	$[-0.9, 0.4] \rightarrow$	$[-6.7, 6.7]$	$[-5.9, 5.9]$	$[-14.3, -1.2] \rightarrow$
SBC-ECOC	$[-5.8, -1.2] \rightarrow$	$[-1.3, 0.2] \rightarrow$	$[-6.7, 6.7]$	$[-5.9, 5.9]$	$[-14.9, -1.2] \rightarrow$
SBC-IDENTITY	$[-6.4, -1.7] \rightarrow$	$[-0.9, 0.4] \rightarrow$	$[-6.7, 6.7]$	$[-5.9, 5.9]$	$[-14.3, -1.2] \rightarrow$

Table 3: 90% bootstrap confidence intervals (in %) for the difference in performance between SBC-KERNEL and (from left to right) OVA, ALL-PAIRS, ECOC, SBC-ECOC, and SBC-IDENTITY

Table 3 shows confidence intervals corresponding to $\delta = 90\%$ for pairwise comparisons of SBC-KERNEL with OVA, ALL-PAIRS, ECOC, SBC-ECOC, and SBC-IDENTITY. (A comparison with SBC-SINGLE is not provided due to its poor performance.) An entry where both a and b are negative reflects a statistically significant advantage of the first algorithm (which is SBC-KERNEL in both columns of the table). Specifically, such an entry indicates a probability of over 90% that the first algorithm is better (note that b is a 95%-quantile). Entries of the form $[b, 0]$, where b is negative, suggest a probability of less than 5% that the second algorithm is better. The symmetrically reversed statements (for entries of the form $[0, a]$) hold as well. If 0 is properly included in the interval,

6. For each row, if the errors of all algorithms are distinct, they are assigned the ranks in $\{1, \ldots, 5\}$. When algorithms share exactly the same error, they are all assigned the same average rank.
7. This test was proposed by Rifkin and Klautau (2004) as an alternative to McNemar's test that accounts for the size of the differences between the algorithms' errors (which is ignored by McNemar's test).

then no algorithm is significantly better than another, but the magnitudes of a and b can indicate which algorithm has an advantage. Note that this statistical procedure does not account for the Bonferroni correction for multiple testing. Nevertheless, the statistics presented are based on the actual classifications of individual test points, and therefore provide another useful perspective.

We interpret the results of Table 3 as follows. We consider a confidence interval $[a, b]$ as interesting if $\big||a| - |b|\big| > 0.5$ (i.e., the skew in the advantage of one algorithm is greater than 0.5%); or (ii) $a < b < 0$; or (iii) $0 < a < b$. Other cases are considered not interesting. Interesting cases marked with '$\rightarrow$' indicate where SBC-KERNEL is significantly better than the other algorithm (OVA, ALL-PAIRS, ECOC, SBC-ECOC, or SBC-IDENTITY). Entries marked with '$\leftarrow$' indicate cases where the other algorithm is better. Unmarked entries do not exhibit a statistically significant interesting event. We see that in all cases, SBC-KERNEL is better than both OVA and ALL-PAIRS. It is also better than SBC-ECOC, and SBC-IDENTITY on three of the five datasets, and insignificantly different on the other two.

These results show that SBC, with learning a linear combination of the basic kernels, improves performance as compared to the performance of SBC-IDENTITY and SBC-ECOC. Moreover, we see that SBC without a learned kernel (the SBC-IDENTITY algorithm) is better than OVA. Thus, these results indicate that the sources of the observed improvements are both the SBC reduction and the kernel learning.

During the course of our experiments with the SBC-KERNEL method, we observed a significant correlation in most cases between the class distribution of the training data and the final μ_i weights reached after the optimization. These correlations indicate that the computational overhead might be reduced by directly using these class distributions as the final μ_i kernel weights, thus solving a single SVM problem. We leave this direction for future work.

6. Concluding Remarks

We introduced a powerful family of SBC reductions based on large margin optimization. For a small number of classes, this approach is well motivated by a generalization bound, which is obtained as a corollary of a known generalization bound for binary classification. We tested our method and compared it to six other known methods over UCI datasets with a small number of classes. These tests indicate that the proposed approach can yield superior performance when the number of classes is small.

Many avenues remain open for future research. It would be very interesting to explore other types of feature extension matrices. A direct optimization of these matrices can also be considered but, as discussed in Section 3.1, the resulting optimization problem will no longer be easy to solve. It would also be interesting to reverse-engineer the resulting kernel transformations and identify a single diagonal extension matrix with good performance. While we believe that such a matrix exists, finding it using the RBF kernel and our optimization procedure is difficult since the objective function becomes non-convex.

The main bottleneck in all SBC methods is the data replication, which poses a true bottleneck when considering large problems. For small problems, this computational load is affordable, and as we show, beneficial. To handle large problems, this bottleneck can be alleviated by using fast approximation to SVM optimization (see (Bordes et al., 2005), (Tsang et al., 2005), (Keerthi et al., 2006)).

References

E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

D. Anguita, S. Ridella, and D. Sterpi. A new method for multiclass support vector machines. In *Proceedings on the Intrernational Joint Conference on Neural Networks*, pages 407–412, 2004.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

O. Bousquet and D. Herrmann. On the complexity of learning the kernel matrix. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 399–406, Cambridge, MA, 2003. MIT Press.

S. Canu and A. Smola. Kernel methods and the exponential family. *Neurocomputing*, 69(7–9): 714–720, 2006.

O. Chapelle. Training an support vector machine in the primal. *To appear in Neural Computation*, 2006.

R. Collobert and S. Bengio. SVMTorch: support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

H. Eves. *Elementary matrix theory*. Dover publications, 1980.

J.H. Friedman. Another approach to polychotomous classification. Technical report, Stanford University, 1996.

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

S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 6:1493–1515, 2006.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El-Ghaoui, and M. Jordan. Learning the kernel matrix via semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

R. Rifkin. *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, MIT, 2002.

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

T.G. Sejnowski and C.R. Rosenberg. Parallel networks that learn to pronounce English text. *Journal of Complex Systems*, 1(1):145–168, 1987.

N. Srebro and S. Ben-David. Learning bounds for support vector machines with learned kernels. In *19th Annual Conference on Learning Theory*, pages 169–183, 2006.

I. Tsang, J. Kwok, and P. Cheung. Core vector machines: fast svm training on very large data sets. *Journal of Machine Learning Research*, 5:363–392, 2005.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

V. Vapnik. *Statistical Learning Theory*. Wiley Interscience, 1998.