Building Sparse Large Margin Classifiers

Mingrui Wu MINGRUI.WU@TUEBINGEN.MPG.DE
Bernhard Schölkopf BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Gökhan Bakir GOEKHAN.BAKIR@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany

Abstract

This paper presents an approach to build *Sparse Large Margin Classifiers* (SLMC) by adding one more constraint to the standard *Support Vector Machine* (SVM) training problem. The added constraint explicitly controls the sparseness of the classifier and an approach is provided to solve the formulated problem. When considering the dual of this problem, it can be seen that building an SLMC is equivalent to constructing an SVM with a modified kernel function. Further analysis of this kernel function indicates that the proposed approach essentially finds a discriminating subspace that can be spanned by a small number of vectors, and in this subspace different classes of data are linearly well separated. Experimental results over several classification benchmarks show that in most cases the proposed approach outperforms the state-of-art sparse learning algorithms.

1. Introduction

In the binary classification problem, we are given a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathcal{R}^d$ is the input data, $\mathcal{X}$ is the input space, and $y_i \in \{-1, 1\}$ is the class label. The goal is to build a classification function $\hat{f} : \mathcal{X} \to \{+1, -1\}, \mathbf{x} \to \hat{f}(\mathbf{x})$, which can correctly classify the unseen test data.

Many kernel learning algorithms, such as *Support Vector Machines* (SVM) (Vapnik, 1995), result in a classification function as $\hat{f}(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$, where $f(\cdot)$ takes the form:

$$f(\mathbf{x}) = \sum_{i=1}^{N_{XV}} \hat{\alpha}_i K(\hat{\mathbf{x}}_i, \mathbf{x}) + b \tag{1}$$

where $\hat{\mathbf{x}}_i \in \mathcal{X}$, $1 \le i \le N_{XV}$, are called *Expansion Vectors* (XVs) in this paper[1], N_{XV} is the number of XVs, $\hat{\alpha}_i \in \mathcal{R}$ is the expansion coefficient associated with $\hat{\mathbf{x}}_i$, $b \in \mathcal{R}$ is the bias and $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ is a kernel function.

Usually K is a positive definite kernel (Schölkopf & Smola, 2002), which implicitly introduces a feature space $\mathcal{F}$. Let $\phi(\cdot)$ denote the map from $\mathcal{X}$ to $\mathcal{F}$, then $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$. So (1) can also be written as a linear function

$$f(\mathbf{x}) = \langle \Psi, \phi(\mathbf{x}) \rangle + b \tag{2}$$

where

$$\Psi = \sum_{i=1}^{N_{XV}} \hat{\alpha}_i \phi(\hat{\mathbf{x}}_i) \tag{3}$$

is the linear expansion of XVs in the feature space $\mathcal{F}$.

When solving practical problems, for example in real-time computer vision, in addition to good classification accuracy, high classification speed is also desirable. The time of calculating (1) (or (2)) is proportional to N_{XV}, so if N_{XV} is large, classification speed is slow. Thus several sparse learning algorithms have been proposed to build kernel classifiers with small N_{XV}.

The *Reduced Set* (RS) method (Burges, 1996; Schölkopf & Smola, 2002) was proposed to simplify (1) by determining N_z vectors $\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}$ and corre-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

[1] The $\hat{\mathbf{x}}_i$, $1 \le i \le N_{XV}$ have different names in different kernel learning algorithms. For example, they are called support vectors in SVM, and relevance vectors in the Relevance Vector Machine (Tipping, 2001). In this paper we uniformly call them expansion vectors for the sake of simplicity.

sponding expansion coefficients $\beta_1, \ldots, \beta_{N_z}$ such that

$$\| \Psi - \sum_{j=1}^{N_z} \beta_j \phi(\mathbf{z}_j) \|^2 \tag{4}$$

is minimized. RS methods approximate and replace Ψ in (2) by $\sum_{j=1}^{N_z} \beta_j \phi(\mathbf{z}_j)$, where the integer $N_z < N_{SV}$ is chosen a priori. Therefore the objective of RS method does not relate to classification performance directly, and in order to apply RS methods, we need to build another kernel classifier in advance.

In (Lee & Mangasarian, 2001), the *Reduced Support Vector Machine* (RSVM) algorithm is proposed, which randomly selects N_z vectors from the training set as XVs, and then computes the expansion coefficients. This algorithm can be applied to build sparse kernel classifiers. But as the XVs are chosen randomly, and may not be good representatives of the training data, good classification performance can not be guaranteed when N_z is small (Lin & Lin, 2003).

The *Relevance Vector Machine* (RVM) (Tipping, 2001) is another algorithm which leads to sparse kernel classifiers. The basic idea of RVM is to assume a prior of the expansion coefficients which favors sparse solutions.

In this paper, we propose an algorithm to build *Sparse Large Margin Classifiers* (SLMC)[2] by modifying the standard SVM training problem. By adding a constraint, we explicitly control the sparseness of the classifier.

The rest of this paper is organized as follows. In section 2, we formulate the SLMC training problem and propose an approach to solve it. The details of the approach are described in section 3 and section 4. In section 5, we analyze the proposed algorithm and point out that it actually finds a discriminating subspace of the feature space $\mathcal{F}$. Some comparisons with related approaches are given in section 6. Experimental results are provided in section 7 and we conclude the paper in the last section.

2. Building an SLMC

Our objective is as follows: given an positive integer N_z, we want to build a classifier such that the number of XVs of the classifier equals N_z and the margin of the classifier is as large as possible. This way we build a large margin classifier whose sparseness is explicitly controlled.

To achieve this, we propose to solve the problem (5)–(8), where C is a positive constant, $\mathbf{w} \in \mathcal{F}$ is the weight vector of the decision hyperplane in feature space, $b \in \mathcal{R}$ is the bias of the classifier, $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^\top \in \mathcal{R}^N$ is the vector of slack variables, $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}] \in \mathcal{R}^{d \times N_z}$ is the matrix of XVs and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{N_z}]^\top \in \mathcal{R}^{N_z}$ is the vector of expansion coefficients.

$$\min_{\mathbf{w},\boldsymbol{\xi},b,\boldsymbol{\beta},\mathbf{Z}} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{5}$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \forall i \tag{6}$$

$$\xi_i \geq 0 \quad \forall i \tag{7}$$

$$\mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i \tag{8}$$

It can be seen that the above problem is a standard (soft margin) SVM training problem with one added constraint (8) saying that the weight vector of the decision hyperplane equals the expansion of the $\phi(\mathbf{z}_i)$, $1 \leq i \leq N_z$. Note that the $\mathbf{z}_i$ are also variables of the objective function, so *they need to be computed when solving the optimization problem.*

Because of the constraint (8), the above problem is not convex, thus we propose a gradient based approach described as follows.

Let G denote the objective function (5). According to the constraints (6)–(8), $\mathbf{w}$ and $\boldsymbol{\xi}$ can be completely determined by b, $\boldsymbol{\beta}$ and $\mathbf{Z}$. So G can be written as

$$G(b, \boldsymbol{\beta}, \mathbf{Z}) = \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{9}$$

where $\mathbf{w}$ depends on $\boldsymbol{\beta}$ and $\mathbf{Z}$, while ξ_i depends on $\boldsymbol{\beta}$, $\mathbf{Z}$ and b.

At any fixed $\mathbf{Z}$, G becomes a function of b and $\boldsymbol{\beta}$, which is denoted by $G(b, \boldsymbol{\beta} \mid \mathbf{Z})$ in the following. To compute the minimum of the function G, we define another function $W(\mathbf{Z})$,

$$W(\mathbf{Z}) = \min_{b \in \mathcal{R}, \boldsymbol{\beta} \in \mathcal{R}^{N_z}} G(b, \boldsymbol{\beta} \mid \mathbf{Z}) \tag{10}$$

that is, at any given $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$, the value of $W(\mathbf{Z})$ is the minimum of the following optimization problem,

$$\min_{\mathbf{w},\boldsymbol{\xi},b,\boldsymbol{\beta}} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{11}$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \forall i \tag{12}$$

$$\xi_i \geq 0 \quad \forall i \tag{13}$$

$$\mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i \tag{14}$$

[2] Here we don't use the phrase "sparse SVM" because the XVs of the resulting classifier are not necessarily support vectors, i.e. they may not lie near the classification boundary.

The above problem is the same as problem (5)–(8) except that $\mathbf{Z}$ is not variable but fixed.

For any $\mathcal{A} \subseteq \mathcal{R}^{d \times N_z}$, according to (10), clearly we have

$$\min_{\mathbf{Z} \in \mathcal{A}} W(\mathbf{Z}) = \min_{b, \boldsymbol{\beta}, \mathbf{Z} \in \mathcal{A}} G(b, \boldsymbol{\beta}, \mathbf{Z}) \tag{15}$$

where $b \in \mathcal{R}$ and $\boldsymbol{\beta} \in \mathcal{R}^{N_z}$.

Therefore any (local) minimum of $W(\mathbf{Z})$ is also a (local) minimum of $G(b, \boldsymbol{\beta}, \mathbf{Z})$, which means the (local) minimum of the original problem (5)–(8) can be found by computing the (local) minimum of function $W(\mathbf{Z})$. Here we propose to minimize $W(\mathbf{Z})$ by the gradient based algorithm. To this end, at any given $\mathbf{Z}$, we need to calculate both the function value $W(\mathbf{Z})$ and the gradient $\nabla W(\mathbf{Z})$. These two problems are discussed in the following two sections respectively.

3. Computing $W(\mathbf{Z})$ and $\boldsymbol{\beta}$

To compute the function value of $W(\mathbf{Z})$ at any given $\mathbf{Z}$, we need to solve the convex optimization problem (11)–(14), which is actually a problem of building an SVM with given XVs $\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}$. This problem has already been considered in the RSVM algorithm (Lee & Mangasarian, 2001; Lin & Lin, 2003). But in the RSVM algorithm, only an approximation of the problem (11)–(14) is solved. Here we will propose a different method which exactly solves this problem. (See section 6.2 for a discussion and section 7.6 for a comparison of the experimental results of these two methods.)

Substituting (14) into (11) and (12), we have

$$\min_{\boldsymbol{\xi}, b, \boldsymbol{\beta}} \quad \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}^z \boldsymbol{\beta} + C \sum_{i=1}^N \xi_i \tag{16}$$

$$\text{subject to} \quad y_i(\boldsymbol{\beta}^\top \psi_z(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \forall i \tag{17}$$

$$\xi_i \geq 0 \quad \forall i \tag{18}$$

where

$$\psi_z(\mathbf{x}_i) = [K(\mathbf{z}_1, \mathbf{x}_i), \ldots, K(\mathbf{z}_{N_z}, \mathbf{x}_i)]^\top \tag{19}$$

is the empirical kernel map (Schölkopf & Smola, 2002) and $\mathbf{K}^z$ is the kernel matrix of $\mathbf{z}_i$, i.e. $\mathbf{K}^z_{ij} = K(\mathbf{z}_i, \mathbf{z}_j)$.

Note that when $N_z = N$ and $\mathbf{z}_i = \mathbf{x}_i$, $1 \leq i \leq N$, this is the standard SVM training problem. In contrast, the problem (16)–(18) is to train a linear SVM in a subspace spanned by $\phi(\mathbf{z}_i)$, $1 \leq i \leq N_z$, where $\mathbf{z}_i$ are are not necessarily training examples.

Now we investigate its dual problem. To derive it, we introduce the Lagrangian,

$$\begin{aligned} & L(\boldsymbol{\xi}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) \\ = \; & \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}^z \boldsymbol{\beta} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \gamma_i \xi_i \\ & - \sum_{i=1}^N \alpha_i [y_i(\boldsymbol{\beta}^\top \psi_z(\mathbf{x}_i) + b) - 1 + \xi_i] \end{aligned} \tag{20}$$

with Lagrange multipliers $\gamma_i \geq 0$ and $\alpha_i \geq 0$.

The derivatives of $L(\boldsymbol{\xi}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\gamma})$ with respect to the primal variables must vanish,

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \mathbf{K}^z \boldsymbol{\beta} - \sum_{i=1}^N \alpha_i y_i \psi_z(\mathbf{x}_i) = 0 \tag{21}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^N \alpha_i y_i = 0 \quad \forall i \tag{22}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \gamma_i = 0 \quad \forall i \tag{23}$$

Equation (21) leads to

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1} \sum_{i=1}^N \alpha_i y_i \psi_z(\mathbf{x}_i) \tag{24}$$

Substituting (21)–(23) into (20) and using (24), we arrive at the dual form of the optimization problem:

$$\max_{\boldsymbol{\alpha} \in \mathcal{R}^N} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j) \tag{25}$$

$$\text{subject to} \quad \sum_{i=1}^N y_i \alpha_i = 0 \tag{26}$$

$$\text{and} \quad 0 \leq \alpha_i \leq C \quad \forall i \tag{27}$$

where

$$\hat{K}_z(\mathbf{x}_i, \mathbf{x}_j) = \psi_z(\mathbf{x}_i)^\top (\mathbf{K}^z)^{-1} \psi_z(\mathbf{x}_j) \tag{28}$$

The function $\hat{K}_z(\cdot, \cdot)$ defined by (28) is a positive definite kernel function (Schölkopf & Smola, 2002). To see this, consider the following map[3],

$$\phi_z(\mathbf{x}_i) = \mathbf{T} \psi_z(\mathbf{x}_i) \tag{29}$$

where $\psi_z(\cdot)$ is defined by (19) and

$$\mathbf{T} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \mathbf{V}^\top \tag{30}$$

[3]The map defined in (29) is called the "whitened" empirical kernel map or "kernel PCA map"(Schölkopf & Smola, 2002).

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues of matrix $\mathbf{K}^z$ and $\mathbf{V}$ is a matrix whose columns are eigenvectors of $\mathbf{K}^z$. So

$$\mathbf{T}^\top \mathbf{T} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V} = (\mathbf{K}^z)^{-1} \tag{31}$$

Combining equation (29)and (31) we have

$$\langle \phi_z(\mathbf{x}_i), \phi_z(\mathbf{x}_j) \rangle = \psi_z(\mathbf{x}_i)^\top (\mathbf{K}^z)^{-1} \psi_z(\mathbf{x}_j) = \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j)$$

It can be seen that problem (25)–(27) has the same form as the dual of an SVM training problem. Therefore *given* $\mathbf{Z}$*, computing the expansion coefficients of SVM with kernel function* K *is equivalent to training an SVM with a modified kernel function* $\hat{K}_z$ defined by (28).

Since problem (25)–(27) is the dual of problem (11)–(14), the optima of these two problems are equal to each other. So given $\mathbf{Z}$, assuming $\alpha_i^z, 1 \le i \le N$ are the solution of (25)–(27), then we can compute $W(\mathbf{Z})$ as

$$W(\mathbf{Z}) = \sum_{i=1}^{N} \alpha_i^z - \frac{1}{2} \sum_{i}^{N} \sum_{j=1}^{N} \alpha_i^z \alpha_j^z y_i y_j \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j) \tag{32}$$

According to (24), the expansion coefficients $\boldsymbol{\beta}$ can be calculated as

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1} \sum_{i=1}^{N} \alpha_i^z y_i \psi_z(\mathbf{x}_i) = (\mathbf{K}^z)^{-1} (\mathbf{K}^{zx}) \mathbf{Y} \boldsymbol{\alpha}^z \tag{33}$$

where $\psi_z(\cdot)$ is defined by (19), $\mathbf{K}^{zx}$ is the matrix defined by $\mathbf{K}^{zx}_{ij} = K(\mathbf{z}_i, \mathbf{x}_j)$, $\mathbf{Y}$ is a diagonal matrix of class labels, i.e. $\mathbf{Y}_{ii} = y_i$, and $\boldsymbol{\alpha}^z = [\alpha_1^z, \ldots, \alpha_N^z]^\top$.

4. Computing $\nabla W(\mathbf{Z})$

To calculate $\nabla W(\mathbf{Z})$, we can apply the following lemma, which closely follows (Chapelle et al., 2002):

Lemma 1. *The derivatives of* $W(\mathbf{Z})$ *with respect to* $\mathbf{z}_{uv}$*, which denotes the* v*-th component of vector* $\mathbf{z}_u$*,* $1 \le u \le N_z, 1 \le v \le d$*, can be computed as follows:*

$$\frac{\partial W}{\partial \mathbf{z}_{uv}} = -\frac{1}{2} \sum_{i,j=1}^{N} \alpha_i^z \alpha_j^z y_i y_j \frac{\partial \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{z}_{uv}} \tag{34}$$

where $\boldsymbol{\alpha}_i^z$, $1 \le i \le N$ denote the solution of problem (25)–(27). In other words, $\nabla W(\mathbf{Z})$ can be computed as if $\boldsymbol{\alpha}^z$ did not depend on $\mathbf{Z}$.

A quite similar lemma is proved in (Chapelle et al., 2002) for the case where there is no upper bound on α_i^z. In the present paper α_i^z is upper bounded by C as shown in (27). However lemma 1 can be proved with the same method adopted by (Chapelle et al., 2002).

According to (28),

$$\begin{aligned} \frac{\partial \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{z}_{uv}} &= \left(\frac{\partial \psi_z(\mathbf{x}_i)}{\partial \mathbf{z}_{uv}}\right)^\top (\mathbf{K}^z)^{-1} \psi_z(\mathbf{z}_j) \\ &+ \psi_z(\mathbf{z}_i)^\top (\mathbf{K}^z)^{-1} \frac{\partial \psi_z(\mathbf{x}_j)}{\partial \mathbf{z}_{uv}} \\ &+ \psi_z(\mathbf{x}_i)^\top \frac{\partial (\mathbf{K}^z)^{-1}}{\partial \mathbf{z}_{uv}} \psi_z(\mathbf{x}_j) \end{aligned}$$

where $\frac{\partial (\mathbf{K}^z)^{-1}}{\partial \mathbf{z}_{uv}}$ can be calculated as

$$\frac{\partial (\mathbf{K}^z)^{-1}}{\partial \mathbf{z}_{uv}} = -(\mathbf{K}^z)^{-1} \frac{\partial \mathbf{K}^z}{\partial \mathbf{z}_{uv}} (\mathbf{K}^z)^{-1}$$

So at any given $\mathbf{Z}$, $W(\mathbf{Z})$ and $\nabla W(\mathbf{Z})$ can be computed as (32) and (34) respectively. In our implementation, we use the LBFGS algorithm (Liu & Nocedal, 1989) to minimize $W(\mathbf{Z})$, which is an efficient gradient based optimization algorithm.

5. The Kernel Function $\hat{K}_z$ and Its Corresponding Feature Space $\mathcal{F}_z$

The kernel function $\hat{K}_z$ plays an important role in our approach. In this section, some analysis of $\hat{K}_z$ is provided, which will give us insights into how to build an SLMC.

It is well known that training an SVM with a nonlinear kernel function K in the input space $\mathcal{X}$ is equivalent to building a linear SVM in a feature space $\mathcal{F}$. The map $\phi(\cdot)$ from $\mathcal{X}$ to $\mathcal{F}$ is implicitly introduced by K. In section 3, we derived that for a given set of XVs $\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}$, training an SVM with kernel function K is equivalent to building an SVM with another kernel function $\hat{K}_z$, which is in turn equivalent to constructing a linear SVM in another feature space. Let $\mathcal{F}_z$ denote this feature space, then the map from the $\mathcal{X}$ to $\mathcal{F}_z$ is $\phi_z(\cdot)$, which is explicitly defined by (29).

According to (29), $\phi_z(\mathbf{x}) = \mathbf{T}\psi_z(\mathbf{x})$. To investigate the role of the matrix $\mathbf{T}$, consider $\mathbf{U}^z$ defined by

$$\mathbf{U}^z = [\phi(\mathbf{z}_1), \ldots, \phi(\mathbf{z}_{N_z})]\mathbf{T}^\top$$

Then

$$(\mathbf{U}^z)^\top \mathbf{U}^z = \mathbf{T}\mathbf{K}^z\mathbf{T}^\top = \mathbf{I}$$

where $\mathbf{I}$ is the unit matrix, which means that $\mathbf{T}^\top$ orthonormalizes $\phi(\mathbf{z}_i)$ in the feature space $\mathcal{F}$. Thus the columns of $\mathbf{U}^z$ can be regarded as an orthonormal basis of a subspace of $\mathcal{F}$. For any $\mathbf{x} \in \mathcal{X}$, if we calculate

the projection of $\phi(\mathbf{x})$ into this subspace, we have

$$\begin{aligned}(\mathbf{U}^z)^\top\phi(\mathbf{x}) &= \mathbf{T}[\phi(\mathbf{z}_1),\ldots,\phi(\mathbf{z}_{N_z})]^\top\phi(\mathbf{x}) \\ &= \mathbf{T}[K(\mathbf{z}_1,\mathbf{x}),\ldots,K(\mathbf{z}_{N_z},\mathbf{x})]^\top \\ &= \mathbf{T}\psi_z(\mathbf{x}) = \phi_z(\mathbf{x})\end{aligned}$$

This shows that the subspace spanned by the columns of $\mathbf{U}^z$ is identical to $\mathcal{F}_z$. As $\mathbf{U}^z$ are obtained by orthonormalizing $\phi(\mathbf{z}_i)$, $\mathcal{F}_z$ is a subspace of $\mathcal{F}$ and it is spanned by $\phi(\mathbf{z}_i)$, $1 \le i \le N_z$.

Now that for a given set of XVs $\mathbf{z}_i$, building an SVM with a kernel function K is equivalent to building a linear SVM in $\mathcal{F}_z$, in order to get good classification performance, we have to find a discriminating subspace $\mathcal{F}_z$ where two classes of data are linearly well separated. Based on this point of view, we can see that our proposed approach essentially finds a subspace $\mathcal{F}_z$ where the margin of the training data is maximized.

6. Comparison with Related Approaches

6.1. Modified RS Method

In the second step of the RS method, after the XVs $\mathbf{z}_1,\ldots,\mathbf{z}_{N_z}$ are obtained, the expansion coefficients $\boldsymbol{\beta}$ are computed by minimizing (4), which leads to (Schölkopf & Smola, 2002)

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1}(\mathbf{K}^{zx})\mathbf{Y}\boldsymbol{\alpha} \tag{35}$$

where $\mathbf{K}^{zx}$ and $\mathbf{Y}$ are defined as in (33), and $\boldsymbol{\alpha}$ is the solution of building an SVM with kernel function K on the training data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

We propose to modify the second step of RS method as (33). Clearly (35) and (33) are of the same form. The only difference is that in (35), $\boldsymbol{\alpha}$ is the solution of training an SVM with kernel function K, while in (33), $\boldsymbol{\alpha}^z$ is the solution of training an SVM with the kernel function $\hat{K}_z$, which takes the XVs $\mathbf{z}_i$ into consideration. As $\boldsymbol{\beta}$ calculated by (33) maximizes the margin of the resulting classifier, we can expect a better classification performance of this modified RS method. We will see this in the experimental results.

6.2. Comparison with RSVM and a Modified RSVM Algorithm

One might argue that our approach appears to be similar to the RSVM, because the RSVM algorithm also restricts the weight vector of the decision hyperplane to be a linear expansion of N_z XVs.

However there are two important differences between the RSVM and our approach. The first one (and probably the fundamental one) is that in the RSVM approach, N_z XVs are randomly selected from the training data in advance, but are not computed by finding a discriminating subspace $\mathcal{F}_z$. The second difference lies in the method for computing the expansion coefficients $\boldsymbol{\beta}$. Our method exactly solves the problem (16)–(18) without any simplifications. But in the RSVM approach, certain simplifications are performed, among which the most significant one is changing the first term in the objective function (16) from $\frac{1}{2}\boldsymbol{\beta}^\top\mathbf{K}^z\boldsymbol{\beta}$ to $\frac{1}{2}\boldsymbol{\beta}^\top\boldsymbol{\beta}$. This step immediately reduces the problem (16)–(18) to a standard linear SVM training problem (Lin & Lin, 2003), where $\boldsymbol{\beta}$ becomes the weight vector of the decision hyperplane and the training set becomes $\{\psi_z(\mathbf{x}_i), y_i\}_{i=1}^N$.

On the other hand, our method of computing $\boldsymbol{\beta}$ is to build a linear SVM in the subspace $\mathcal{F}_z$, which is to train a linear SVM for the training data set $\{\phi_z(\mathbf{x}_i), y_i\}_{i=1}^N$.

Now let us compare the two training sets mentioned above, i.e. $\{\phi_z(\mathbf{x}_i), y_i\}_{i=1}^N$ and $\{\psi_z(\mathbf{x}_i), y_i\}_{i=1}^N$. As derived in section 5, $\phi_z(\mathbf{x}_i)$ are calculated by projecting $\phi(\mathbf{x}_i)$ onto a set of vectors, which is obtained by orthonormalizing $\phi(\mathbf{z}_j)$ $(1 \le j \le N_z)$, while $\psi_z(\mathbf{x}_i)$ is calculated by computing the dot production between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{z}_j)$ $(1 \le j \le N_z)$ directly, without the step of orthonormalization.

Analogous to the modified RS method, we propose a modified RSVM algorithm: Firstly, N_z training data are randomly selected as XVs, then the expansion coefficients $\boldsymbol{\beta}$ are computed by (33).

6.3. Comparison with the RVM

The RVM (Tipping, 2001) algorithm and many other sparse learning algorithms, such as sparse greedy algorithms (Nair et al., 2002), or SVMs with l_1-norm regularization (Bennett, 1999), result in a classifier whose XVs are a subset of the training data. In contrast, the XVs of SLMC do not necessarily belong to the training set. This means that SLMC can in principle locate better discriminating XVs. Consequently, with the same number of XVs, SLMC can have better classification performance than the RVM and other sparse learning algorithms which select the XVS only from the training data. This can be seen from the experimental results provided in section 7.6.

6.4. SLMC vs Neural Networks

Since the XVs of the SLMC do not necessarily belong to the training set and training an SLMC is a gradient based process, the SLMC can be thought of as

a neural network with weight regularization (Bishop, 1995). However, there are clear differences between the SLMC algorithm and a feed forward neural network. First, analogous to SVM, SLMC considers the geometric concept of margin, and aims to maximizes it. To this end, the regularizer takes into account the kernel matrix $\mathbf{K}^z$. Furthermore, since SLMC minimizes the "hinge-loss", a tighter bound on the zero-one classification loss, better generalization behavior may be expected than for a neural network algorithm minimizing the squared loss.

On the other hand, analogous to neural networks, we also have an additional regularization via the number N_z determining the number of XVs, which is a clear advantage in some practical applications where runtime constraints exist and the maximum prediction time is known a priori. Note that the prediction time (the number of kernel evaluations) of a soft margin SVM scales linearly with the number of training patterns (Steinwart, 2003).

7. Experimental Results

7.1. Approaches to be Compared

The following approaches are compared in the experiments: Standard SVM, RS method, modified RS method (MRS, cf. section 6.1), RSVM, modified RSVM (MRSVM, cf. section 6.2), relevance vector machine (RVM), and the proposed SLMC approach.

Note that in our experiments, RS and MRS use exactly the same XVs, but they compute the expansion coefficients by (35) and (33) respectively. Similarly RSVM and MRSVM also use the same set of XVs, the difference lies in the method for computing the expansion coefficients.

7.2. Data Sets

Seven classification benchmarks are considered: USPS, Banana, Breast Cancer, Titanic, Waveform, German and Image. The last six data sets are provided by Gunnar Rätsch and are downloaded from http://ida.first.fraunhofer.de/projects/bench. For the USPS data set, 7291 examples are used for training and the remaining 2007 are for testing. For each of the last six data sets, there are 100 training/test splits and we follow the same scheme as (Tipping, 2001): our results show averages over the first 10 of those.

7.3. Parameter Selection

A Gaussian kernel is used in the experiments:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \parallel \mathbf{x} - \mathbf{x}' \parallel^2) \qquad (36)$$

The parameters for different approaches are as follows:

Standard SVM: For the USPS data set, we use the same parameters as in (Schölkopf & Smola, 2002): $C = 10$ and $\gamma = 1/128$. For the other data sets, we use the same parameters provided by Gunnar Rätsch, which are shown on the same website where these data sets are downloaded.

RSVM and MRSVM: We perform 5-fold cross validation on the training set to select parameters for the RSVM. MRSVM uses the same parameters as the RSVM.

RS method: Obviously the RS method uses the same kernel parameter as the standard SVM, since it aims to simplify the standard SVM solution.

SLMC and MRS: In our experiments, they use exactly the same parameters as the standard SVM on all the data sets.

RVM: The results for the RVM are taken directly from (Tipping, 2001), where 5-fold cross validation was performed for parameter selection.

7.4. Experimental Settings

For each data set, firstly a standard SVM is trained with the LIBSVM software[4]. (For USPS, 10 SVMs are built, each trained to separate one digit from all others). Then the other approaches are applied. The ratio N_z/N_{SV} varies from 5% to 10%.

For the RSVM, we use the implementation contained in the LIBSVM Tools[5].

For the RS method, there is still no standard or widely accepted implementation, so we tried three different ones: a program written by ourselves, the code contained in the machine learning toolbox SPIDER[6], and the code contained in the statistical pattern recognition toolbox STPRTOOL[7]. For each data set, we apply these three implementations and select the best one corresponding to the minimal value of the objective function (4). Thus our RS results respect the state of the art in RS methods.

7.5. Choice of the Initial XVs

The initial XVs of the SLMC are randomly selected from the training data. More complicated methods have been tried and the resulting classification results are similar. For example, using K-means algorithm to

[4]From http://www.csie.ntu.edu.tw/~cjlin/libsvm
[5]From http://www.csie.ntu.edu.tw/~cjlin/libsvmtools
[6]From http://www.kyb.mpg.de/bs/people/spider
[7]From http://cmp.felk.cvut.cz/~xfrancv/stprtool

choose the initial XVs can improve the final classification results on the USPS dataset, but the improvement is as small as 0.1%. So here we only report the results obtained by random initialization.

7.6. Numerical Results

Experimental results are shown in Table 1. In Table 1, N_{SV} stands for the number of *Support Vectors* (SVs) of the standard SVM, N_z represents the number of XVs of other sparse learning algorithms.

From Table 1, it can be seen the classification accuracy of SLMC is comparable with the full SVM when $N_z/N_{SV} = 0.1$.

Table 1 also illustrates that SLMC outperforms the other sparse learning algorithms in most cases. Also the SLMC usually improves the classification results of the RS method. In some cases the improvement is large such as on Banana and Image data sets.

When comparing MRS with RS, and MRSVM with RSVM, the results in Table 1 demonstrate that in most cases MRS beats RS, and similarly, MRSVM usually outperforms RSVM a little. This means that for a given set of XVs, computing the expansion coefficients according to (33) is a good choice.

7.7. Some Results on XVs

It is known that the XVs of standard SVM are support vectors, which lie near the classification boundary. Here we give three examples to illustrate what the XVs of SLMC look like.

Example 1. Building an SVM involves solving problem (5)–(7), while building an SLMC is to solve the same problem plus one more constraint (8). If we want to build an SLMC with the same number of XVs as a standard SVM, namely $N_z = N_{SV}$, then the optimal solution of problem (5)–(7) is also a *global optimal* solution of problem (5)–(8), since it satisfies all the constraints. So in this special case, the support vectors of the standard SVM are also an optimal choice of XVs for SLMC.

Example 2. On the USPS data set, we built an SVM on training data with $\gamma = 1/128$, $C = 10$ to separate digit '3' from digit '8'. The resulting SVM has 116 SVs and a test error rate of 1.8%. Then we built an SLMC with the same γ and C, while $Nz = 12$ (i.e. about 10% of the number of SVs). The resulting SLMC also has a test error rate of 1.8%. As shown in Figure 1, the images of the 12 XVs produced by SLMC approach look like digits.

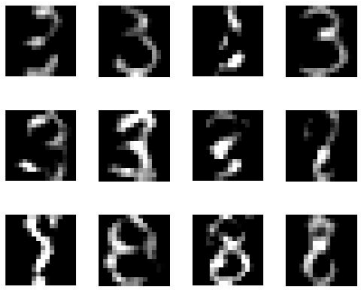

Figure 1. Images of XVs for separating '3' and '8'

7.8. Training Time of SLMC

Building an SLMC is a gradient based process, where each iteration consists of computing the $\phi_z(\mathbf{x}_i)$, $1 \le i \le N$, training a linear SVM over $\{\phi_z(\mathbf{x}_i), y_i\}_{i=1}^N$,[8] and then computing the gradient $\nabla W(\mathbf{Z})$.

Let $T_{SVM}(N, d)$ denote the time complexity of training a linear SVM over a data set containing N d-dimensional vectors, then the time complexity of training an SLMC is

$$O(n \times (NN_z d + T_{SVM}(N, N_z) + N_z^3 + N_z^2 d))$$

where n is the number of iterations of the SLMC training process. In experiments, we found that the SLMC algorithm requires 20–200 iterations to converge.

We cannot directly compare the training time of SLMC with RS methods and RVM (Relevance Vector Machine), because we used C++ to implement our approach, while the publicly available codes of RS methods and the RVM are written in Matlab. Using these implementations and a personal computer with Pentium 4 CPU of 3GHz, one gets the following numbers: On the USPS dataset, SLMC takes 6.9 hours to train, while the RS method takes 2.3 hours. On the Banana dataset, SLMC training is about 1.5 seconds, and RVM training is about 5 seconds.

8. Conclusions

We presented an approach to build sparse large margin classifiers, which essentially finds a discriminating subspace $\mathcal{F}_z$ of the feature space $\mathcal{F}$. Experimental results indicate that this approach often exhibits a better classification accuracy than the sparse learning algorithms

[8] Equivalently we can build an SVM with the kernel function $\hat{K}_z$ over $\{\mathbf{x}_i, y_i\}_{i=1}^N$. But this is much slower because it is time consuming to compute $\hat{K}_z(\cdot, \cdot)$ defined by (28).

Table 1. Results on seven classification benchmarks. The test error rates of each algorithm are presented. The N_{SV} for the last six datasets are the averages over 10 training/test splits. The best result in each group is shown in boldface. The number of XVs of the RVM is not chosen a priori, but comes out as a result of training. So for the RVM, the ratio N_z/N_{SV} is given in order to compare it with other algorithms. For each data set, the result of the RVM is shown in boldface if it is the best compared to the other sparse learning algorithms.

Dataset		USPS	Banana	Breast Cancer	Titanic	Waveform	German	Image
SVM	N_{SV}	2683	86.7	112.8	70.6	158.9	408.2	172.1
	Error(%)	4.3	11.8	28.6	22.1	9.9	22.5	2.8
$N_z/N_{SV} = 5\%$	RS	**4.9**	39.4	28.8	37.4	**9.9**	22.9	37.6
	MRS	**4.9**	27.6	28.8	**23.9**	10.0	22.5	19.4
	RSVM	11.6	29.9	29.5	24.5	15.1	23.6	23.6
	MRSVM	11.5	28.1	29.4	24.8	14.7	23.9	20.7
	SLMC	**4.9**	**16.5**	**27.9**	26.4	**9.9**	**22.3**	**5.2**
$N_z/N_{SV} = 10\%$	RS	**4.7**	21.9	**27.9**	26.6	10.0	22.9	18.3
	MRS	4.8	17.5	29.0	22.6	**9.9**	**22.6**	6.9
	RSVM	8.2	17.5	31.0	22.9	11.6	24.5	14.2
	MRSVM	8.0	16.9	30.3	23.9	11.8	23.7	12.7
	SLMC	**4.7**	**11.0**	**27.9**	**22.4**	**9.9**	22.9	**3.6**
RVM	N_z/N_{SV}(%)	11.8	13.2	5.6	92.5	9.2	3.1	20.1
	Error(%)	5.1	**10.8**	29.9	23.0	10.9	**22.2**	3.9

to which we compared.

A by-product of this paper is a method for calculating the expansion coefficients of SVMs for given XVs. Based on this method we proposed modified version of the RS method and the RSVM. Experimental results show that these two modified algorithms can usually improve the classification accuracy of their counterparts. (One can also try this method on other algorithms such as the RVM.)

Finally, we propose that the technique of adding one more constraint (8) to explicitly control the sparseness can be applied to build other sparse learning algorithms, such as sparse kernel fisher, sparse kernel PCA, sparse one-class SVM and sparse regression algorithms, etc.

References

Bennett, K. P. (1999). Combining support vector and mathematical programming methods for classification. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods*, 307–326. Cambridge MA: The MIT Press.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press.

Burges, C. J. C. (1996). Simplified support vector decision rules. *Proc. 13th International Conference on Machine Learning* (pp. 71–77). Morgan Kaufmann.

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*, 131–159.

Lee, Y.-J., & Mangasarian, O. L. (2001). RSVM: reduced support vector machines. *CD Proceedings of the First SIAM International Conference on Data Mining*. Chicago.

Lin, K.-M., & Lin, C.-J. (2003). A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, *14*, 1449–1459.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Math. Programming*, *45*, 503–528.

Nair, P. B., Choudhury, A., & Keane, A. J. (2002). Some greedy learning algorithms for sparse regression and classification with mercer kernels. *Journal of Machine Learning Research*, *3*, 781–801.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: The MIT Press.

Steinwart, I. (2003). Sparseness of support vector machine. *Journal of Machine Learning Research*, *4*, 1071–1105.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–214.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer Verlag.