Journal of Machine Learning Research 7 (2006) 603–624 Submitted 10/05; Published 4/06

A Direct Method for Building Sparse Kernel Learning Algorithms

Mingrui Wu MINGRUI.WU@TUEBINGEN.MPG.DE
Bernhard Schölkopf BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Gökhan Bakır GOEKHAN.BAKIR@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany

Editor: Nello Cristianini

Abstract

Many kernel learning algorithms, including support vector machines, result in a kernel machine, such as a kernel classifier, whose key component is a weight vector in a feature space implicitly introduced by a positive definite kernel function. This weight vector is usually obtained by solving a convex optimization problem. Based on this fact we present a direct method to build sparse kernel learning algorithms by adding one more constraint to the original convex optimization problem, such that the sparseness of the resulting kernel machine is explicitly controlled while at the same time performance is kept as high as possible. A gradient based approach is provided to solve this modified optimization problem. Applying this method to the support vectom machine results in a concrete algorithm for building sparse large margin classifiers. These classifiers essentially find a discriminating subspace that can be spanned by a small number of vectors, and in this subspace, the different classes of data are linearly well separated. Experimental results over several classification benchmarks demonstrate the effectiveness of our approach.

Keywords: sparse learning, sparse large margin classifiers, kernel learning algorithms, support vector machine, kernel Fisher discriminant

1. Introduction

Many kernel learning algorithms (KLA) have been proposed for solving different kinds of problems. For example the support vector machine (SVM) (Vapnik, 1995) have been widely used for classification and regression, the minimax probability machine (MPM) (Lanckriet et al., 2002) is another competitive algorithm for classification, the one-class SVM (Schölkopf and Smola, 2002) is a useful tool for novelty detection, while the kernel Fisher discriminant (KFD) (Mika et al., 2003) and the kernel PCA (KPCA) (Schölkopf and Smola, 2002) are powerful algorithms for feature extraction.

Many kernel learning algorithms result in a kernel machine (KM) (such as a kernel classifier), whose output can be calculated as

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

where $\mathbf{x} \in \mathcal{X} \subset \mathcal{R}^d$ is the input data, $\mathcal{X}$ is the input space, $\hat{\mathbf{x}}_i \in \mathcal{X}$, $1 \leq i \leq N_{XV}$, 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 and Smola, 2002), which implicitly introduces a feature space $\mathcal{F}$. Let $\phi(\cdot)$ denote the map from $\mathcal{X}$ to $\mathcal{F}$, then

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}.$$

Hence (1) can also be written as a linear function

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

where

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

is the weight vector of the KM and it equals the linear expansion of XVs in the feature space $\mathcal{F}$.

For all the KLAs mentioned above, the vector $\mathbf{w}$ is obtained by solving a *convex* optimization problem. For example, the SVM can be formulated as a quadratic programming problem (Vapnik, 1995), in (Mika et al., 2003), a convex formulation is proposed for KFD and a least squares SVM formulation is established for KPCA in (Suykens et al., 2002).

From the above description, we can see that although many KLAs are proposed for solving different kinds of problems and have various formulations, there are three widely known common points among them. First, each of them results in a KM whose key component is a weight vector $\mathbf{w}$, which can be expressed as the linear expansion of XVs in the feature space $\mathcal{F}$. Second, the vector $\mathbf{w}$ is obtained by solving a convex optimization problem. Third, the output of the resulting KM is calculated as (1) (or (2)).

When solving practical problems, we want the time for computing the output to be as short as possible. For example in real-time image recognition, in addition to good classification accuracy, high classification speed is also desirable. The time of calculating (1) (or (2)) is proportional to N_{XV}. Thus several sparse learning algorithms have been proposed to build KMs with small N_{XV}.

The reduced set (RS) method (Burges, 1996; Schölkopf and Smola, 2002) was proposed to simplify (1) by determining N_z vectors $\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}$ and corresponding expansion coefficients $\beta_1, \ldots, \beta_{N_z}$ such that

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

is minimized. RS methods approximate and replace $\mathbf{w}$ in (2) by $\sum_{j=1}^{N_z} \beta_j \phi(\mathbf{z}_j)$, where $N_z < N_{XV}$. The objective of RS method does not directly relate to the performance of the

1. The $\hat{\mathbf{x}}_i$, $1 \leq i \leq N_{XV}$ have different names in different kernel learning algorithms. For example, they are called support vectors in the 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.

KM it aims to simplify, and in order to apply RS methods, we need to build another KM in advance.

In (Lee and 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 and Lin, 2003).

The relevance vector machine (RVM) (Tipping, 2001) is another algorithm which leads to sparse KMs. The basic idea of the RVM is to assume a prior of the expansion coefficients which favors sparse solutions.

In this paper, based on the common points of KLAs mentioned before, we present a direct method to build sparse kernel learning algorithms (SKLA). In particular, given a KLA, we modify it by adding one more constraint to its corresponding convex optimization problem. The added constraint explicitly controls the sparseness of the resulting KM and a gradient based approach is proposed to solve the modified optimization problem. We will also show that applying this method to the SVM will result in a specific algorithm for building sparse large margin classifiers (SLMC).[2]

The remainder of this paper is organized as follows. In section 2, we describe a direct method for building SKLAs. After this, we will focus on a particular application of this method to the SVM algorithm, leading to a detailed algorithm for building SLMC. The SLMC algorithm is presented in section 3, where we will also point out that it actually finds a discriminating subspace of the feature space $\mathcal{F}$. Some comparisons with related approaches are given in section 4. Experimental results are provided in section 5 and we conclude the paper in the last section.[3]

2. A Direct Method of Building SKLAs

In this section, we propose a direct method for building SKLAs.

2.1 Basic Idea

As mentioned before, many KLAs can be formulated as an optimization problem, which can be written in a general form as follows:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad f(\mathbf{w},b,\boldsymbol{\xi}), \tag{5}$$

$$\text{subject to} \quad g_i(\mathbf{w},b,\boldsymbol{\xi}) \le 0, \quad 1 \le i \le N_g, \tag{6}$$

$$h_j(\mathbf{w},b,\boldsymbol{\xi}) = 0, \quad 1 \le j \le N_h, \tag{7}$$

where $f(\mathbf{w},b,\boldsymbol{\xi})$ is the objective function to be minimized, $\mathbf{w} \in \mathcal{F}$ and $b \in \mathcal{R}$ are respectively the weight vector and the bias of the KM to be built, $\boldsymbol{\xi} = [\xi_1,\ldots,\xi_{N_\xi}]^\top \in \mathcal{R}^{N_\xi}$ is a vector of some auxiliary variables (such as the slack variables in the soft margin training problem),

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.
3. This paper is an extension of our previous work (Wu et al., 2005).

N_ξ is the number of auxiliary variables, N_g is the number of inequality constraints specified by $g_i(\mathbf{w}, b, \boldsymbol{\xi})$, while N_h is the number of equality constraints specified by $h_j(\mathbf{w}, b, \boldsymbol{\xi})$.

Our objective is as follows: given a KLA and a positive integer N_z, we want to modify the given KLA such that the number of XVs of the resulting KM equals N_z while at the same time the performance of the KM should be kept as well as possible. To achieve this, we propose to solve the following problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\beta},\mathbf{Z}} \quad & f(\mathbf{w}, b, \boldsymbol{\xi}), && (8)\\ \text{subject to} \quad & g_i(\mathbf{w}, b, \boldsymbol{\xi}) \leq 0, \quad 1 \leq i \leq N_g, && (9)\\ & h_j(\mathbf{w}, b, \boldsymbol{\xi}) = 0, \quad 1 \leq j \leq N_h, && (10)\\ & \mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i, && (11) \end{aligned}$$

where (8)–(10) are exactly the same as (5)–(7), while $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}] \in \mathcal{R}^{d \times N_z}$ 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.

It can be seen that the above problem is the problem (5)–(7) with one added constraint (11) saying that the weight vector of the resulting KM equals the expansion of the $\phi(\mathbf{z}_i)$, $1 \leq i \leq N_z$. Note that the $\mathbf{z}_i$ are also variables, so they need to be computed when solving the optimization problem.

Due to the constraint (11), solving the problem (8)–(11) will naturally lead to a sparse KM. Further more, since the objective function of the problem (8)–(11) is exactly the same as the original problem (5)–(7), so in principle the performance of the resulting KM can be kept as well as possible.

Because of the non-convex constraint (11), it is difficult to obtain the global optimum of the above problem, thus we propose a gradient based approach. However, our gradient based minimization will be performed *only* on the expansion vectors $\mathbf{Z}$ but not on all the variables. To this end, we define the following the *marginal function* $W(\mathbf{Z})$ which is obtained by keeping the expansion vectors in problem (8)–(11) fixed, i.e. :

$$\begin{aligned} W(\mathbf{Z}) := \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\beta}} \quad & f(\mathbf{w}, b, \boldsymbol{\xi}), && (12)\\ \text{subject to} \quad & g_i(\mathbf{w}, b, \boldsymbol{\xi}) \leq 0, \quad 1 \leq i \leq N_g, && (13)\\ & h_j(\mathbf{w}, b, \boldsymbol{\xi}) = 0, \quad 1 \leq j \leq N_h, && (14)\\ \text{and} \quad & \mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i. && (15) \end{aligned}$$

The above problem is the same as problem (8)–(11) except that $\mathbf{Z}$ is not variable but fixed.

Clearly any global (or local) minimum of $W(\mathbf{Z})$ is also a global (or local) minimum of problem (8)–(11), which means that the minima of the original problem (8)–(11) can be found by computing the minima of function $W(\mathbf{Z})$. Here we propose to minimize $W(\mathbf{Z})$ by the gradient based algorithm. To this end, at any given $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$, 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 next subsection.

2.2 Computing $W(\mathbf{Z})$ and its Gradient $\nabla W(\mathbf{Z})$

To compute the function value of $W(\mathbf{Z})$ at any given $\mathbf{Z}$, we need to solve the optimization problem (12)–(15). Obviously this is a problem dependent task. However as mentioned before, the original optimization problem (5)–(7) of many current KLAs are convex, and (15) is just a linear constraint once $\mathbf{Z}$ is fixed. Therefore the problem (12)–(15) is still convex, which means its global optimum, and thus the function value of $W(\mathbf{Z})$, can be readily computed.

Next we turn to consider how to compute $\nabla W(\mathbf{Z})$, which requires more carefulness since in general $W(\mathbf{Z})$ is not necessarily differentiable. According to the constraint (11), the weight vector $\mathbf{w}$ can be completely determined by $\boldsymbol{\beta}$ and $\mathbf{Z}$, so the functions f, g_i and h_j can be regarded as functions of b, $\boldsymbol{\xi}$, $\boldsymbol{\beta}$ and $\mathbf{Z}$. Without causing confusions, we write them as $f(\mathbf{x}, \mathbf{Z})$, $g_i(\mathbf{x}, \mathbf{Z})$ and $h_j(\mathbf{x}, \mathbf{Z})$ in the following, where $\mathbf{x} := [b, \boldsymbol{\xi}^\top, \boldsymbol{\beta}^\top]^\top$.

Substituting (15) into (12)–(14), we have

$$\min_{\mathbf{x}} \quad f(\mathbf{x}, \mathbf{Z}), \tag{16}$$

$$\text{subject to} \quad g_i(\mathbf{x}, \mathbf{Z}) \le 0, \quad 1 \le i \le N_g, \tag{17}$$

$$h_j(\mathbf{x}, \mathbf{Z}) = 0, \quad 1 \le j \le N_h, \tag{18}$$

and $W(\mathbf{Z})$ is the optimal value of the above optimization problem.

To compute $\nabla W(\mathbf{Z})$, we can apply the following lemma which gives both the conditions when $W(\mathbf{Z})$ is differentiable and an explicit form of the derivative:

Lemma 1 *(Gauvin and Dubeau, 1982): Assume that all the functions f, g_i and h_j in problem (16)–(18) are continuously differentiable and suppose that problem (16)–(18) has a unique optimal solution $\bar{\mathbf{x}}$ at $\mathbf{Z} = \bar{\mathbf{Z}}$. Furthermore let $\bar{\alpha}_i$ and $\bar{\beta}_j$ be the unique corresponding Lagrange multipliers associated with g_i and h_j respectively, $1 \le i \le N_g$, $1 \le j \le N_h$. Assume further that the feasible set of problem (16)–(18) is uniformly compact at $\bar{\mathbf{Z}}$ and that, the optimal solution $\bar{\mathbf{x}}$ is Mangasarian-Fromovitz regular, then the gradient of $W(\mathbf{Z})$ exists at $\bar{\mathbf{Z}}$ and equals*

$$\nabla W(\mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}} = \nabla_{\mathbf{Z}} f(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}} + \sum_{i=1}^{N_g} \bar{\alpha}_i \, \nabla_{\mathbf{Z}} g_i(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}} + \sum_{j=1}^{N_h} \bar{\beta}_j \, \nabla_{\mathbf{Z}} h_j(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}}, \tag{19}$$

where $\nabla_{\mathbf{Z}} f(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}}$ denotes the gradient of $f(\mathbf{x}, \mathbf{Z})$ with respect to $\mathbf{Z}$ at $\mathbf{Z} = \bar{\mathbf{Z}}$ while fixing $\mathbf{x}$ at $\bar{\mathbf{x}}$.

As can be seen that in addition to the uniqueness of the optimal solution and its corresponding Lagrange multipliers, lemma 1 also requires the feasible set to be uniformly compact and the optimal solution to be Mangasarian-Fromovitz regular. The formal definitions of the last two conditions are given in appendix B. Since the properties of the feasible set and the optimal solutions depend on the specific KLA, we will discuss all these conditions for some particular KLAs in subsection 3.3 and appendix A. We will see that the conditions of lemma 1 are very mild for many current KLAs.

3. Building an SLMC

Having described our direct method for building SKLAs, we will focus on a concrete application of this method in the rest of this paper. In particular, we will apply this method to SVMs in order to obtain an algorithm for building SLMCs. We will also analyze its properties and compare it with other related approaches. In this section, we will present the SLMC algorithm by closely following the discussions of section 2.

3.1 Objective

Now we begin to consider the binary classification problem, where we are given a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathcal{X}$ is the input data, and $y_i \in \{-1, 1\}$ is the class label.

Our objective is as follows: given a training data set and 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.

Based on the direct method described in the last section, we need to solve the problem (8)–(11) to achieve this goal. For the moment, (8)–(10) become the SVM training problem and the constraint (11) controlls the sparseness of the resulting classifier. So we should solve the following problem

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\xi},b,\boldsymbol{\beta},\mathbf{Z}} \quad & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i, && (20)\\
\text{subject to} \quad & y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i, && (21)\\
& \xi_i \geq 0, \quad \forall i, && (22)\\
& \mathbf{w} = \sum_{i=1}^{N_z}\phi(\mathbf{z}_i)\beta_i, && (23)
\end{aligned}$$

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.

Following our proposed method, to solve the above problem, we turn to minimize the marginal function $W(\mathbf{Z})$ defined in problem (12)–(15). For the current problem, the value of $W(\mathbf{Z})$ is the minimum of the following optimization problem,

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\xi},b,\boldsymbol{\beta}} \quad & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i, && (24)\\
\text{subject to} \quad & y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i, && (25)\\
& \xi_i \geq 0, \quad \forall i, && (26)\\
& \mathbf{w} = \sum_{i=1}^{N_z}\phi(\mathbf{z}_i)\beta_i. && (27)
\end{aligned}$$

The above problem is the same as problem (20)–(23) except that $\mathbf{Z}$ is not variable but fixed.

Following the discussion in section 2.1, the (local) minimum of the original problem (20)–(23) can be found by computing the (local) minimum of function $W(\mathbf{Z})$ and we will use the gradient based algorithm to do this. In the following two subsections we will discuss how to compute the function value $W(\mathbf{Z})$ and the gradient $\nabla W(\mathbf{Z})$ respectively.

3.2 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 (24)–(27), 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 and Mangasarian, 2001; Lin and Lin, 2003). But in the RSVM algorithm, only an approximation of the problem (24)–(27) is solved. Here we will propose a different method which exactly solves this problem. (See section 4.2 for a discussion and section 5.5 for a comparison of the experimental results of these two methods.)

Substituting (27) into (24) and (25), we have

$$\begin{aligned} \min_{\boldsymbol{\xi},b,\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}^z \boldsymbol{\beta} + C\sum_{i=1}^{N} \xi_i, && (28) \\ \text{subject to} \quad & y_i(\boldsymbol{\beta}^\top \psi_z(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i, && (29) \\ & \xi_i \geq 0, \quad \forall i, && (30) \end{aligned}$$

where

$$\psi_z(\mathbf{x}_i) = [K(\mathbf{z}_1, \mathbf{x}_i), \ldots, K(\mathbf{z}_{N_z}, \mathbf{x}_i)]^\top \tag{31}$$

is the empirical kernel map (Schölkopf and 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 (28)–(30) 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,

$$\begin{aligned} & L(\boldsymbol{\xi}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) && (32) \\ = \; & \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}^z \boldsymbol{\beta} + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \gamma_i \xi_i \\ & - \sum_{i=1}^{N} \alpha_i [y_i(\boldsymbol{\beta}^\top \psi_z(\mathbf{x}_i) + b) - 1 + \xi_i], \end{aligned}$$

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,

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \mathbf{K}^z \boldsymbol{\beta} - \sum_{i=1}^{N} \alpha_i y_i \psi_z(\mathbf{x}_i) = 0, \tag{33}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0, \quad \forall i, \tag{34}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \gamma_i = 0, \quad \forall i. \tag{35}$$

Equation (33) leads to

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1} \sum_{i=1}^{N} \alpha_i y_i \psi_z(\mathbf{x}_i). \tag{36}$$

Substituting (33)–(35) into (32) and using (36), we arrive at the dual form of the optimization problem:

$$\max_{\boldsymbol{\alpha} \in \mathcal{R}^N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j), \tag{37}$$

$$\text{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \tag{38}$$

$$\text{and} \quad 0 \leq \alpha_i \leq C, \quad \forall i, \tag{39}$$

where

$$\hat{K}_z(\mathbf{x}_i, \mathbf{x}_j) = \psi_z(\mathbf{x}_i)^\top (\mathbf{K}^z)^{-1} \psi_z(\mathbf{x}_j). \tag{40}$$

The function $\hat{K}_z(\cdot, \cdot)$ defined by (40) is a positive definite kernel function (Schölkopf and Smola, 2002). To see this, consider the following map,[4]

$$\phi_z(\mathbf{x}_i) = \mathbf{T} \psi_z(\mathbf{x}_i), \tag{41}$$

where $\psi_z(\cdot)$ is defined by (31) and

$$\mathbf{T} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \mathbf{V}^\top, \tag{42}$$

where $\boldsymbol{\Lambda}$ 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

$$\mathbf{T}^\top \mathbf{T} = \mathbf{V} \boldsymbol{\Lambda}^{-1} \mathbf{V} = (\mathbf{K}^z)^{-1}. \tag{43}$$

Combining equation (41)and (43) 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 (37)–(39) has the same form as the dual of an SVM training problem. Therefore *given* $\mathbf{Z}$*, computing the expansion coefficients of SVM with kernel*

4. The map defined in (41) is called the "whitened" empirical kernel map or "kernel PCA map"(Schölkopf and Smola, 2002).

function K is equivalent to training an SVM with a modified kernel function $\hat{K}_z$ defined by (40).

Since problem (37)–(39) is the dual of problem (24)–(27), 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 (37)–(39), then we can compute $W(\mathbf{Z})$ as

$$W(\mathbf{Z}) = \sum_{i=1}^{N} \alpha_i^z - \frac{1}{2} \sum_{i}^{N} \sum_{j=1}^{N} \alpha_i^z \alpha_j^z y_i y_j \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j). \tag{44}$$

According to (36), the expansion coefficients $\boldsymbol{\beta}$ can be calculated as

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1} \sum_{i=1}^{N} \alpha_i^z y_i \psi_z(\mathbf{x}_i) = (\mathbf{K}^z)^{-1} (\mathbf{K}^{zx}) \mathbf{Y} \boldsymbol{\alpha}^z, \tag{45}$$

where $\psi_z(\cdot)$ is defined by (31), $\mathbf{K}^{zx}$ is the matrix defined by $\mathbf{K}_{ij}^{zx} = K(\mathbf{z}_i, \mathbf{x}_j)$, $\mathbf{Y}$ is a diagonal matrix of class labels, i.e. $\mathbf{Y}_{ii} = y_i$, and $\boldsymbol{\alpha}^z = [\alpha_1^z, \ldots, \alpha_N^z]^\top$.

3.3 Computing $\nabla W(\mathbf{Z})$ of SLMC

To compute $\nabla W(\mathbf{Z})$, we can apply lemma 1 to the soft margin SVM training problem (37)–(39) and yield the following result.

Corollary 2 *In the soft margin SVM training problem (37)–(39), assume that the kernel function $\hat{K}_z(\cdot, \cdot)$ is strictly positive definite and the resulting support vectors come from both positive and negative classes,*[5] *then the derivatives of $W(\mathbf{Z})$ with respect to $\mathbf{z}_{uv}$, which denotes the v-th component of vector $\mathbf{z}_u$, $1 \le u \le N_z, 1 \le v \le d$, exists and can be computed as follows:*

$$\frac{\partial W}{\partial \mathbf{z}_{uv}} = -\frac{1}{2} \sum_{i,j=1}^{N} \alpha_i^z \alpha_j^z y_i y_j \frac{\partial \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{z}_{uv}}, \tag{46}$$

where α_i^z, $1 \le i \le N$ denote the solution of problem (37)–(39). In other words, $\nabla W(\mathbf{Z})$ can be computed as if α^z did not depend on $\mathbf{Z}$.[6]

The proof of corollary 2 is given in appendix C.

As can be seen in corollary 2, to apply lemma 1 to calculate $\nabla W(\mathbf{Z})$, we only need to make two assumptions on problem (37)–(39): The kernel function $\hat{K}_z(\cdot, \cdot)$ is strictly positive definite and the resulting support vectors come from both classes. Certainly these are *not* strict assumptions for most practical applications. Similarly one can verify that lemma 1 can also be applied to many other KLAs with mild assumptions, such as the one-class SVM.

5. Let α_i^z, $1 \le i \le N$ denote the optimal solution of problem (37)–(39), support vectors are those input data $\mathbf{x}_i$ whose corresponding α_i^z are larger than 0 (Vapnik, 1995).
6. In corollary 2, α_i^z is bounded above by C as shown in (39). A similar conclusion is proposed in (Chapelle et al., 2002) for the hard margin SVM training problem, where there is no upper bound on α_i^z. This implies that the feasible set is not compact, hence lemma 1 can not be applied any more. Actually in (Chapelle et al., 2002), only the uniqueness of the optimal solution is emphasized, which, to our knowledge, is not enough to guarantee the differentiability of the marginal function $W(\mathbf{Z})$.

And we will show another example in appendix A on applying our direct method to sparsify the KFD algorithm (Mika et al., 2003).

According to (40),

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

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 (44) and (46) respectively. In our implementation, we use the LBFGS algorithm (Liu and Nocedal, 1989) to minimize $W(\mathbf{Z})$, which is an efficient gradient based optimization algorithm.

3.4 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 2.2, 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 (41).

According to (41), $\phi_z(\mathbf{x}) = \mathbf{T}\psi_z(\mathbf{x})$. 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.

4. Comparison with Related Approaches

In this section we compare the SLMC algorithm with related approaches.

4.1 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 and Smola, 2002)

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1}(\mathbf{K}^{zx})\mathbf{Y}\boldsymbol{\alpha}, \tag{47}$$

where $\mathbf{K}^{zx}$ and $\mathbf{Y}$ are defined as in (45), 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 (45). Clearly (47) and (45) are of the same form. The only difference is that in (47), $\boldsymbol{\alpha}$ is the solution of training an SVM with kernel function K, while in (45), $\boldsymbol{\alpha}^z$ 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 (45) 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.

4.2 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 (28)–(30) 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 (28) 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 (28)–(30) to a standard linear SVM training problem (Lin and 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 3.4, $\phi_z(\mathbf{x}_i)$ 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 $\phi(\mathbf{z}_j)$ $(1 \leq j \leq N_z)$ 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 (45).

4.3 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 5.5.

4.4 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,[7] 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 an SVM, the SLMC considers the geometric concept of margin, and aims to maximizes it. To this end, the regularizer takes into account the kernel matrix $\mathbf{K}^z$. Second, SLMC minimizes the "hinge-loss", which is different from the loss functions adopted by neural networks.[8] Therefore, both the regularizer and the loss function of SLMC are different from those of traditional perceptrons.

Furthermore, the SLMC algorithm is just an application of our 'direct sparse' method. It is straightforward to apply this method to build sparse one-class SVM algorithm (Schölkopf and Smola, 2002), to which there is no obvious neural network counterpart.

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 an 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).

5. Experimental Results

Now we conduct some experiments to investigate the performance of the SLMC algorithm and compare it with other related approaches.

7. This is also similar to the recent work of (Snelson and Ghahramani, 2006) on building sparse Gaussian processes, which was done at almost the same time with our previous work (Wu et al., 2005).
8. Note that the shape of the hinge-loss is similar to that of the loss function adopted in logistic regression, where the logarithm of the logistic sigmoid function (Bishop, 1995) is involved. Here the logistic sigmoid function refers to $y(x) = \frac{1}{1+\mathrm{e}^{-x}}$. So the shape of the hing-loss is different from that of the loss function used by the perceptron.

5.1 Approaches to be Compared

The following approaches are compared in the experiments: Standard SVM, RS method, modified RS method (MRS, cf. section 4.1), RSVM, modified RSVM (MRSVM, cf. section 4.2), 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 (47) and (45) respectively. Similarly RSVM and MRSVM also use the same set of XVs, the difference lies in the method for computing the expansion coefficients.

5.2 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 can be 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.

5.3 Parameter Selection

A Gaussian kernel is used in the experiments:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \parallel \mathbf{x} - \mathbf{x}' \parallel^2). \tag{48}$$

The parameters for different approaches are as follows:

Standard SVM: For the USPS data set, we use the same parameters as in (Schölkopf and Smola, 2002): $C = 10$ and $\gamma = 1/128$. For the other data sets, we use the parameters provided by Gunnar Rätsch, which are shown on the same website where these data sets are downloaded.[9]

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: 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.

5.4 Experimental Settings

For each data set, first a standard SVM is trained with the LIBSVM software.[10] (For the USPS, ten 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.[11]

9. See http://ida.first.fraunhofer.de/projects/bench
10. From http://www.csie.ntu.edu.tw/~cjlin/libsvm
11. From http://www.csie.ntu.edu.tw/~cjlin/libsvmtools

For the RS method, there is still no standard or widely accepted implementation, so we try three different ones: a program written by ourselves, the code contained in the machine learning toolbox SPIDER,[12] and the code contained in the statistical pattern recognition toolbox STPRTOOL.[13] For each data set, we apply these three implementations and select the best one corresponding to the minimal value of the objective function (4).

5.5 Numerical Results

Experimental results are shown in Table 1, where the initial XVs of the SLMC are randomly selected from the training data. 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.

Data Set		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

Table 1: Results on seven classification benchmarks. The test error rates of each algorithm are presented. The N_{SV} for the last six data sets 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.

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

12. From http://www.kyb.mpg.de/bs/people/spider
13. From http://cmp.felk.cvut.cz/~xfrancv/stprtool

a little. This means that for a given set of XVs, computing the expansion coefficients according to (45) is a good choice.

5.6 Some Implementation Details

In table 1, we report the results obtained by random initialization. The K-means algorithm has also been tried to choose the initial XVs and resulting classification results are similar. To illustrate this quantitatively, in table 2, we present the results obtained by using the K-means algorithm for initialization.

Data Set		USPS	Banana	Breast Cancer	Titanic	Waveform	German	Image
N_z/N_{SV} = 5%	random	4.9	16.5	27.9	26.4	9.9	22.3	5.2
	k-means	4.9	16.2	26.8	24.4	9.9	22.7	6.0
N_z/N_{SV} = 10%	random	4.7	11.0	27.9	22.4	9.9	22.9	3.6
	k-means	4.6	10.9	27.3	23.2	9.9	22.7	3.8

Table 2: Results of the SLMC algorithm, obtained by random initialization and k-means initialization.

In our proposed approach, we need to compute the inverse of $\mathbf{K}^z$ (see for example (45)). Theoretically if the Gaussian kernel is used and $\mathbf{z}_i$, $1 \leq i \leq N_z$, are different from each other, then $\mathbf{K}^z$ should be full rank, whose inverse can be computed without any problems. However in experiments, we do observe the cases where $\mathbf{K}^z$ was ill conditioned. But we find out the reason for this is that there are some duplicated data points in the data sets, which are accidentally selected as the initial XVs. For example, in the first training set of the Image data set, the first and the 521st data point are exactly the same. So in experiments, we remove the duplicated points as a preprocessing step.

5.7 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 two examples to illustrate what the XVs of SLMC look like.

Example 1. Building an SVM involves solving problem (20)–(22), while building an SLMC is to solve the same problem plus one more constraint (23). 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 (20)–(22) is also a *global optimal* solution of problem (20)–(23), 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.

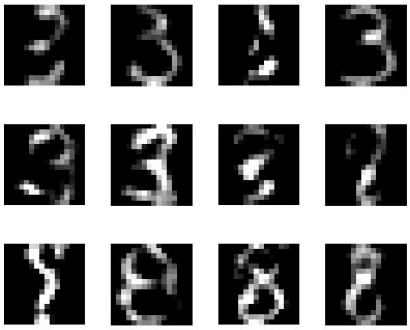

Figure 1: Images of XVs for separating '3' and '8'.

5.8 Training Time of SLMC

Building an SLMC is a gradient based process, where each iteration consists of computing the $\phi_z(\mathbf{x}_i)$, $1 \leq i \leq N$, training a linear SVM over $\{\phi_z(\mathbf{x}_i), y_i\}_{i=1}^N$,[14] 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 find 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 code of RS methods and the RVM is written in Matlab. Using these implementations and a personal computer with a Pentium 4 CPU of 3GHz, one gets the following numbers: On the USPS data set, SLMC takes 6.9 hours to train, while the RS method takes 2.3 hours. On the Banana data set, SLMC training is about 1.5 seconds, and RVM training is about 5 seconds.

Training an SLMC is time consuming on large data sets. However in practice, once the training is finished, the trained KM will be put into use processing large amount of test data. For applications where processing speed is important, such as real time computer vision, sparse KMs can be of great value. This is the reason why several SKLAs have been developed although they are more expensive than the standard SVM algorithm.

Furthermore, some kernel learning algorithms are not sparse at all, such as kernel ridge regression, KFD, KPCA, which means that all the training data need to be saved as the XVs in the resulting KMs trained by these algorithms. Hence building sparse versions of these algorithms can not only accelerate the evaluation of the test data, but also dramatically

14. 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 (40).

reduce the space needed for storing the trained KM. An example of this will be given in appendix A.

6. Conclusions

We present a direct method to build sparse kernel learning algorithms. There are mainly two advantages of this method: First, it can be applied to sparsify many current kernel learning algorithms. Second it simultaneously considers the sparseness and the performance of the resulting KM. Based on this method we propose 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, at comparable sparsity, than the sparse learning algorithms 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 a modified version of the RS method and the RSVM. Experimental results show that these two modified algorithms can improve the classification accuracy of their counterparts. One could also try this method on other algorithms such as the RVM.

Possible future work may include applying the proposed method to other kernel learning algorithms and running the direct method greedily to find the XVs one after another in order to accelerate the training procedure.

Acknowledgments

We would like to acknowledge the anonymous reviewers for their comments that significantly improved the quality of the manuscript.

Appendix A. Sparse KFD

We have derived the SLMC algorithm as an example of our direct sparse learning approach. Here we will show another example to build sparse KFD (Mika et al., 2003).

In the KFD algorithm, we need to consider the following Rayleigh quotient maximization problem (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004).[15]

$$\max_{\mathbf{w}\in\mathcal{F}} \frac{\mathbf{w}^\top \mathbf{A}\mathbf{w}}{\mathbf{w}^\top \mathbf{B}\mathbf{w} + \mu \|\mathbf{w}\|^2}. \tag{49}$$

In (49), $\mathbf{A}$ and $\mathbf{B}$ are respectively the between-class and within-class variances of the training data in the feature space $\mathcal{F}$, while μ is a regularization parameter. It can be seen that both $\mathbf{A}$ and $\mathbf{B}$ are positive definite.

15. As mentioned before, convex formulations have been proposed for the KFD (Mika et al., 2003; Suykens et al., 2002). Here we only consider the traditional non-convex formulation whose solution can be easily obtained via eigen decomposition. This also illustrates that our method can also be applied to non-convex optimization problems in some special cases.

The following is an equivalent form of (49)

$$\min_{\mathbf{w}\in\mathcal{F}} \quad -\mathbf{w}^\top \mathbf{A}\mathbf{w}, \tag{50}$$

$$\text{subject to} \quad \mathbf{w}^\top \hat{\mathbf{B}}\mathbf{w} = 1, \tag{51}$$

where $\hat{\mathbf{B}} = \mathbf{B} + \mu\mathbf{I}$, implying that $\hat{\mathbf{B}}$ is strictly positive definite.

Following our proposed approach, in order to build sparse KFD (SKFD), we have to solve the following problem:

$$\min_{\mathbf{w}\in\mathcal{F},\boldsymbol{\beta}\in\mathcal{R}^{N_z},\mathbf{Z}\in\mathcal{R}^{d\times N_z}} \quad -\mathbf{w}^\top \mathbf{A}\mathbf{w}, \tag{52}$$

$$\text{subject to} \quad \mathbf{w}^\top \hat{\mathbf{B}}\mathbf{w} = 1, \tag{53}$$

$$\mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i. \tag{54}$$

Substituting (54) into (52) and (53), we have

$$\min_{\boldsymbol{\beta}\in\mathcal{R}^{N_z},\mathbf{Z}\in\mathcal{R}^{d\times N_z}} \quad -\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta}, \tag{55}$$

$$\text{subject to} \quad \boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top = 1, \tag{56}$$

where $\mathbf{A}^z = (\phi(\mathbf{Z}))^\top \mathbf{A}(\phi(\mathbf{Z}))$ and $\mathbf{B}^z = (\phi(\mathbf{Z}))^\top \hat{\mathbf{B}}(\phi(\mathbf{Z}))$, where with a little abuse of symbols, we use $\phi(\mathbf{Z})$ to denote the matrix $[\phi(\mathbf{z}_1), \ldots, \phi(\mathbf{z}_{N_z})]$. Note that in the problem (55)–(56), $\mathbf{A}^z \in \mathcal{R}^{N_z\times N_z}$ is positive definite, while $\mathbf{B}^z \in \mathcal{R}^{N_z\times N_z}$ is strictly positive definite.

As before, we define the marginal function $W(\mathbf{Z})$ as the minimal value of the following optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathcal{R}^{N_z}} \quad -\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta}, \tag{57}$$

$$\text{subject to} \quad \boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top = 1. \tag{58}$$

Note the above problem is the same as the problem (55)–(56) except that in the above problem, $\mathbf{Z}$ is fixed rather than variable.

Now we need to consider how to compute $W(\mathbf{Z})$ and $\nabla W(\mathbf{Z})$. To compute the function value $W(\mathbf{Z})$, we need to solve the problem (57)–(58). By the Lagrange multiplier method, this problem can be solved by solving the following unconstrained optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathcal{R}^{N_z},\lambda\in\mathcal{R}} J(\boldsymbol{\beta},\lambda) = -\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta} + \lambda(\boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top - 1), \tag{59}$$

with Lagrange multiplier $\lambda \in \mathcal{R}$.

The derivative of $J(\boldsymbol{\beta},\lambda)$ with respect to $\boldsymbol{\beta}$ and λ must vanish, leading to

$$\mathbf{A}^z\boldsymbol{\beta} = \lambda\mathbf{B}^z\boldsymbol{\beta}, \tag{60}$$

$$\boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top = 1. \tag{61}$$

Equation (60) shows that $\boldsymbol{\beta}$ should be an eigenvector of the matrix $(\mathbf{B}^z)^{-1}\mathbf{A}^z$ and λ should be the corresponding eigenvalue. Left multiplying both sides of equation (60) by $\boldsymbol{\beta}^\top$ and using (61), we have

$$\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta} = \lambda. \tag{62}$$

Since $-\boldsymbol{\beta}^\top \mathbf{A}^z \boldsymbol{\beta}$ is the objective function (57) we are minimizing, we can see that its minimal value should equal the negative of the largest eigenvalue of $(\mathbf{B}^z)^{-1}\mathbf{A}^z$. Therefore the function value $W(\mathbf{Z})$ is obtained.

Let $\bar{\boldsymbol{\beta}}$ and $\bar{\lambda}$ denote the optimal solution and the corresponding Lagrange multiplier of problem (57)–(58), as derived above, $\bar{\lambda}$ is the largest eigenvalue of $(\mathbf{B}^z)^{-1}\mathbf{A}^z$ and $\bar{\boldsymbol{\beta}}$ is the corresponding eigenvector multiplied by a constant such that equation (61) is satisfied.

As mentioned above, $\mathbf{B}^z$ is strictly positive definite. Here we assume that that there is an unique eigenvector corresponding to $\bar{\lambda}$. As equation (58) is the only constraint of problem (57)–(58), the optimal solution $\bar{\boldsymbol{\beta}}$ is Mangasarian-Fromovitz regular. And it is straightforward to verify that the set of feasible solutions $\mathcal{S}(\mathbf{Z})$ is uniformly compact if $\mathbf{B}^z$ is strictly positive definite. Therefore according to lemma 1, the derivative of $W(\mathbf{Z})$ with respect to $\mathbf{z}_{uv}$, which denotes the v-th component of vector $\mathbf{z}_u$, $1 \le u \le N_z, 1 \le v \le d$, exists and can be computed as follows:

$$\frac{\partial W(\mathbf{Z})}{\partial \mathbf{z}_{uv}} = -\bar{\boldsymbol{\beta}}^\top \frac{\partial \mathbf{A}^z}{\partial \mathbf{z}_{uv}} \bar{\boldsymbol{\beta}} + \bar{\lambda} \bar{\boldsymbol{\beta}}^\top \frac{\partial \mathbf{B}^z}{\partial \mathbf{z}_{uv}} \bar{\boldsymbol{\beta}}. \tag{63}$$

Now that both the function value $W(\mathbf{Z})$ and the gradient $\nabla W(\mathbf{Z})$ can be computed, the (local) optimum of the problem (52)–(54) can be computed by the gradient based algorithm.

After obtaining the XVs $\mathbf{z}_i$ by solving the problem (52)–(54), we can take the solution β_i of this problem as the expansion coefficients. However we can not get the bias b for the resulting KM in this way (c.f equation (1)). As mentioned before, having $\mathbf{z}_i$, we can apply our proposed method that the expansion coefficients and the bias can be calculated by solving the problem (37)–(39).

Experimental results on six classification benchmarks for the proposed SKFD algorithm are provided in table 3.

Data Set		Banana	Breast Cancer	Titanic	Waveform	German	Image
SVM	N_{SV}	86.7	112.8	70.6	158.9	408.2	172.1
	Error(%)	11.8	28.6	22.1	9.9	22.5	2.8
KFD	N_z	400	200	150	400	700	1300
	Error(%)	10.8	25.8	23.2	9.9	23.7	3.3
N_z/N_{SV} = 10%	RS	21.9	27.9	26.6	10.0	22.9	18.3
	MRS	17.5	29.0	**22.6**	**9.9**	**22.6**	6.9
	RSVM	17.5	31.0	22.9	11.6	24.5	14.2
	MRSVM	16.9	30.3	23.9	11.8	23.7	12.7
	SKFD	**10.8**	**25.5**	23.5	**9.9**	23.5	**4.0**

Table 3: Results on six classification benchmarks. The SKFD is initialized with the k-means algorithm. The best results among the sparse learning algorithms are in boldface. Similarly as before, the results reported here are the averages over the first 10 training/test splits, except for the KFD algorithm, whose results are taken directly from (Schölkopf and Smola, 2002), which are the averages over all the 100 training/test splits. For the KFD algorithm, the number of expansion vectors N_z is the same as the number of training data since KFD is not sparse at all.

The results presented in table 3 validate the effectiveness of the proposed SKFD algorithm. Furthermore, the original KFD algorithm is not sparse all all, i.e. all the training data need to be stored as the XVs. Therefore the proposed SKFD algorithm not only accelerates the test phase, but also significantly reduces the space needed for storing the resulting KM. For example, the Banana data set contains 400 training data, implying that on average only $86.7 \times 10\% = 8.7$ XVs need to stored in the resulting KM trained by the SKFD, saving about $1 - 8.7/400 = 97.8\%$ storage compared with the orignial KFD algorithm.

Appendix B. Formal Definitions of the Two Conditions in Lemma 1

For each $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$, let $\mathcal{S}(\mathbf{Z})$ denote the feasible set of problem (16)–(18)

$$\mathcal{S}(\mathbf{Z}) = \{\mathbf{x} \mid g_i(\mathbf{x}, \mathbf{Z}) \leq 0,\ 1 \leq i \leq N_g\} \cap \{\mathbf{x} \mid h_j(\mathbf{x}, \mathbf{Z}) = 0,\ 1 \leq j \leq N_h\}.$$

Definition 3 *(Gauvin and Dubeau, 1982) The feasible set $\mathcal{S}(\mathbf{Z})$ of problem (16)–(18) is uniformly compact at $\bar{\mathbf{Z}}$ if there is a neighborhood $\mathcal{N}(\bar{\mathbf{Z}})$ of $\bar{\mathbf{Z}}$ such that the closure of $\bigcup_{\mathbf{Z} \in \mathcal{N}(\bar{\mathbf{Z}})} \mathcal{S}(\mathbf{Z})$ is compact.*

Definition 4 *(Mangasarian, 1969) For any $\mathbf{Z}$, a feasible point $\bar{\mathbf{x}} \in \mathcal{S}(\mathbf{Z})$ of problem (16)–(18) is said to be Mangasarian-Fromovitz regular if it satisfies the following Mangasarian-Fromovitz regularity condition:*

1. *There exists a vector $\mathbf{v}$ such that*

$$\mathbf{v}^\top \nabla_{\mathbf{x}} g_i(\mathbf{x}, \mathbf{Z})|_{\mathbf{x}=\bar{\mathbf{x}}} < 0, \quad i \in \{i \mid g_i(\bar{\mathbf{x}}, \mathbf{Z}) = 0\}, \tag{64}$$

$$\mathbf{v}^\top \nabla_{\mathbf{x}} h_j(\mathbf{x}, \mathbf{Z})|_{\mathbf{x}=\bar{\mathbf{x}}} = 0, \quad 1 \leq j \leq N_h, \tag{65}$$

2. *The gradients $\{\nabla_{\mathbf{x}} h_j(\mathbf{x}, \mathbf{Z})|_{\mathbf{x}=\bar{\mathbf{x}}},\ 1 \leq j \leq N_h\}$ are linearly independent,*

where $\nabla_{\mathbf{x}}$ denotes the gradient with respect to the variables $\mathbf{x}$.

Appendix C. Proof of Corollary 2

Proof The proof is just to verify all the conditions in lemma 1.

First, for the soft margin SVM training problem (37)–(39), it is known that if the kernel function $\hat{K}_z(\cdot, \cdot)$ is strictly positive definite then the problem has an unique solution and the corresponding Lagrange multipliers are also unique.

Second, it can be seen that at any $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$, the set of feasible solutions $\mathcal{S}(\mathbf{Z})$ is compact and does not depend on $\mathbf{Z}$, therefore $\mathcal{S}(\mathbf{Z})$ is uniformly compact at any $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$.

Third, obviously the second part of the Mangasarian-Fromovitz regularity condition holds since there is only one equality constraint. Now we prove that the first part of the Mangasarian-Fromovitz regularity condition also holds by constructing a vector $\mathbf{v} = [v_1, \ldots, v_N]^\top \in \mathcal{R}^N$ that satisfies both (64) and (65). For ease of description, we partition the index set $\{1, \ldots, N\}$ into the following three subsets according to α_i^z $(1 \leq i \leq N)$, which are the solution of problem (37)–(39): $S_1 = \{i \mid \alpha_i^z = 0\}$, $S_2 = \{i \mid \alpha_i^z = C\}$ and $S_3 = \{i \mid 0 < \alpha_i^z < C\}$. Furthermore, for any vector $\mathbf{t} = [t_1, \ldots, t_N]^\top \in \mathcal{R}^N$, we use $\mathbf{t}_{s_k}$ $(1 \leq k \leq 3)$ to denote the sub-vector of $\mathbf{t}$ that is composed of t_i for $i \in S_k$, $1 \leq k \leq 3$.

First, to make (64) hold, we can simply assign an arbitrary positive value to v_i if $i \in S_1$, and an arbitrary negative value to v_j if $j \in S_2$. To make (65) true, we distinguish two cases:

Case 1, $|S_3| > 0$. In this case, the vector $\mathbf{v}_{s_3}$ can be easily computed as

$$\mathbf{v}_{s_3} = -\frac{\mathbf{y}_{s_3}}{\|\mathbf{y}_{s_3}\|^2} \sum_{i \in S_1 \cup S_2} v_i y_i,$$

where $\mathbf{y} = [y_1, \ldots y_N]^\top$. The above equation results in $\mathbf{v}^\top \mathbf{y} = 0$, which is the same as (65).

Case 2, $|S_3| = 0$. In this case, all the resulting support vectors correspond to α_i^z for $i \in S_2$, which come from both classes according to the assumptions in corollary 2. Therefore the left side of equation (65) equals

$$\begin{aligned} \mathbf{v}^\top \mathbf{y} &= \sum_{i \in S_1} v_i y_i + \sum_{i \in S_2,\ y_i > 0} v_i y_i + \sum_{i \in S_2,\ y_i < 0} v_i y_i \\ &= \sum_{i \in S_1} v_i y_i + \sum_{i \in S_2,\ y_i > 0} v_i - \sum_{i \in S_2,\ y_i < 0} v_i. \end{aligned} \tag{66}$$

Recall that we construct $\mathbf{v}$ such that $v_i < 0$ for $i \in S_2$. So if $\mathbf{v}^\top \mathbf{y} = 0$, equation (65) already holds. If $\mathbf{v}^\top \mathbf{y} > 0$, we can always decrease the values (or equivalently, increase the absolute values) of v_i in the second term of equation (66) to make $\mathbf{v}^\top \mathbf{y} = 0$, while at the same time to keep $v_i < 0$ for $i \in S_2$ so that (64) still holds. Similarly, if $\mathbf{v}^\top \mathbf{y} < 0$, we can decrease the values of v_i in the third term of equation (66).

Thus the first part of the Mangasarian-Fromovitz regularity condition also holds. Hence the optimal solution of problem (37)–(39) is Mangasarian-Fromovitz regular.

Therefore all the conditions in lemma 1 are satisfied and (46) follows from (19) since the constraints of problem (37)–(39) do not depend on $\mathbf{Z}$, which means both the second and the third terms in (19) are $\mathbf{0}$. ■

References

K. P. Bennett. Combining support vector and mathematical programming methods for classification. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, pages 307–326. The MIT Press, Cambridge MA, 1999.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proc. 13th International Conference on Machine Learning*, pages 71–77. Morgan Kaufmann, 1996.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

J. Gauvin and F. Dubeau. Differential properties of the marginal function in mathematical programming. *Mathematical Programming Study*, 19:101–119, 1982.

G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.

Y. Lee and O. L. Mangasarian. RSVM: reduced support vector machines. In *CD Proceedings of the First SIAM International Conference on Data Mining*, Chicago, 2001.

K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14:1449–1459, 2003.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.

O. L. Mangasarian. *Nonlinear Programming*. McGraw-Hill, New York, 1969.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. J. Smola, and K.-R. Mueller. Constructing descriptive and discriminative non-linear features: Rayleigh coefficients in kernel feature spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):623–628, 2003.

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

B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1259–1266. MIT Press, Cambridge, MA, 2006.

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

J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

M. Wu, B. Schölkopf, and G. Bakir. Building sparse large margin classifiers. In L. D. Raedt and S. Wrobel, editors, *Proc. 22th International Conference on Machine Learning*, pages 1001–1008. ACM, 2005.