McMaster University

Advanced Optimization Laboratory

Title:

Self-Adaptive Support Vector Machines: Modelling and Experiments

Authors:

Peng Du, Jiming Peng, Tamás Terlaky

AdvOl-Report No. 2004/10

August 2004, Hamilton, Ontario, Canada

Self-Adaptive Support Vector Machines: Modelling and Experiments

Peng Du*, Jiming Peng, and Tamás Terlaky

August 9, 2004

Abstract

In this paper, we introduce a bi-level optimization formulation for the problems of model and feature selection in Support Vector Machines (SVMs). To select the best model, we embed the standard convex quadratic problem of SVM training into a lower level problem in a bi-level optimization model where the optimal objective value of the quadratic problem of SVMs is minimized over the feasible range of the kernel parameters at the upper level of the bi-level model. Since the quadratic problem's optimal objective value is a continuous function of the continuous kernel parameters, implicitly defined over a reasonable

*Department of Computing and Software, McMaster University, Hamilton, Ontario L8S 4L7, Canada. Email: dup2, pengj, terlaky@mcmaster.ca. The first author was supported by an NSERC scholarship. The research of the second author was supported by the NSERC grant # RPG 249635-02 and a PREA award. The research of the third author was supported by the NSERC grant #RPG 0048923 and the Canada Research Chair Program. This research was also supported by the MITACS project "New Interior Point Methods and Software for Convex Conic-Linear Optimization and Their Application to Solve VLSI Circuit Layout Problems".

interval, the solution of this bi-level problem always exists. The problem of feature selection can be handled in the same framework where the kernel functions are extended by introducing independent kernel parameters for all the features in the original space.

Two approaches for solving the bi-level problem of model and feature selection are considered as well. Experimental results show that the bi-level formulation provides a powerful tool for model selection.

Key Words: Support Vector Machines (SVMs), Machine Learning, Model Selection, Feature Selection, Bi-Level Programming.

1 Introduction

We deal with the classification problem in this paper, i.e., given a training set $Z = \{(x^1, y^1), (x^2, y^2), \ldots, (x^\ell, y^\ell)\}$ of ℓ instances, where each instance carries n attributes ($x^i = (x^i_1, \ldots, x^i_n)^T \in R^n$) and a class label $y^i \in \{1, -1\}$, the main task of classification is to construct a decision boundary separating the two classes such that the probability of classification error, or misclassification, made by the resulting decision boundary on a new instance is minimized. The process of constructing the best decision boundary for the training set Z can be considered as a process of learning the information contained in Z. There are many algorithms for this type of problems in the literature [17]. In the present paper, we only discuss Support Vector Machines (SVMs).

The notion of SVM, first introduced by Vapnik in 1995 [17], represents a set of particular learning algorithms. Mathematically speaking, a SVM is an optimization problem that tries to find a hyperplane in the original input space (denoted by $\mathcal{X}$) to separate a given training set Z correctly and leave

as much distance as possible from the closest instances to the hyperplane on both sides. The distance from the closest instances to the separating hyperplane is called margin, and the instances that realize the maximal margin are called support vectors. If the training set is not linearly separable, then a nonlinear boundary has to be constructed. In this situation, the original input space is first mapped into a higher-dimensional space, called *feature space* denoted by $\mathcal{F}$. Then, we search the feature space for a hyperplane that can separate the instances in $\mathcal{F}$. The mapping from $\mathcal{X}$ to $\mathcal{F}$ is defined by a so-called *kernel function*. Another way to handle the inseparable case is allowing misclassification. This is typically done by introducing a penalty factor C in the optimization model and the total penalty is found by summing up penalties on each misclassification. In this case, instead of finding the hyperplane with the maximal margin, we are trying to find a hyperplane that minimizes the sum of the reciprocal of the margin and the total penalty. The combined penalty function is used as the objective in the optimization model.

Since their first introduction, SVMs have been extensively studied by many researchers and recognized as one of the main learning algorithms for real world classification problems, such as hand-written digit recognition [13], image recognition [14] and protein homology detection [10]. However, some questions still remain open. For instance, how to select the right kernel functions for a classification task, how to find appropriate values for the kernel parameters, and how to penalize a misclassification and how to decide automatically the contribution from each attribute to the final decision. The first two issues are associated with the model selection, while the last issue is called feature selection in the machine learning community.

The issue of model selection has been investigated in the field of machine

learning. In [3], Chapelle and Vapnik first presented new functionals for support vector machine model selection. These new functionals can be used to estimate the generalization error. More results regarding the generalization error of SVMs were also presented by Chapelle and Vapnik in [4], where a gradient descent algorithm was proposed to minimize the estimated generalization error over a set of kernel parameters. The estimated generalization error used in [4] is the smoothed test error based on some probability model.

The problem of feature selection for SVMs was first studied by Bradley et al. [2]. Later, Weston et al. [18] introduced a method for the feature selection problem in 2000. In 2002, Chapelle and Vapnik [4] suggested another method in which each feature in an original input space carries its own kernel parameters. If the value of a kernel parameter is very small, the corresponding feature can be removed from the data set without influencing the classification accuracy significantly.

In this paper, we propose a bi-level optimization approach that can help us in both the model and feature selection. Our idea is similar to that in [4]. However, the objective function in the master problem of our bi-level model is different from what used in [4]. Instead of the estimated generalization error, we simply minimize the objective obtained by solving the subproblem in the bi-level framework. This is because in our SVM model, we have already used a parameter to penalize the misclassification error. Or, in other words, we use the sum of the squared margin and the penalized misclassification error to replace the estimated generalization error. The corresponding SVM is called *Self-Adaptive* SVM, or simply SASVM, as the proposed approach is able to automatically tune the kernel parameters to the best values for a given data set.

The paper is organized as follows. In Section 2, we describe briefly

the optimization model for SVMs and discuss various kernel functions. In Section 3, we introduce self-adaptive SVMs. We also discuss how to extend the kernel functions to handle the feature selection problem. Two strategies for solving the bi-level problem for model and feature selections are examined in Section 4. Then, we present some experimental results for both model and feature selections. Finally, in Section 5, some conclusions are presented along with suggestions for future work.

2 SVMs and Kernel Functions

Let Z be the training set. A linear decision function in the original input space is typically given by [17]:

$$f(x; w, b) = w^T x + b, \tag{1}$$

where $w \in R^n$ and $b \in R$ stand for the weight vector and bias. SVMs try to find the optimal w and b by solving the following optimization problem:

$$\begin{array}{lrcll} \min\limits_{w,b} & \frac{1}{2} w^T w + C\xi^T e & & & \\ \text{s.t.} & y^i(w^T x^i + b) & \geq & 1 - \xi_i, & i = 1, \ldots, \ell, \\ & \xi_i & \geq & 0, & i = 1, \ldots, \ell, \end{array} \tag{2}$$

where $\xi_i,\ i = 1, \ldots, \ell$ are slack variables that represent the classification errors, C is the penalty factor and e is the all one vector with appropriate dimension. The problem (2) is a standard convex quadratic programming problem. Its dual problem can be written as follows:

$$\begin{array}{lrcl} \max\limits_{\alpha} & \alpha^T e - \frac{1}{2}\alpha^T \tilde{\mathcal{K}} \alpha & & \\ \text{s.t.} & y^T \alpha & = & 0, \\ & Ce & \geq \alpha \geq & \mathbf{0}, \end{array} \tag{3}$$

where $\alpha \in R^\ell$ is a vector of the dual variables, $y = [y^1, y^2, \cdots, y^\ell]^T$, and $\tilde{\mathcal{K}} \in R^{\ell \times \ell}$ with $\tilde{\mathcal{K}}_{ij} = y^i y^j ({x^i}^T x^j)$. Problem (3) is a standard quadratic programming problem as well, for which efficient algorithms are available. Furthermore, if the problem (3) is a strictly convex quadratic problem, then the optimal solution is unique. Let α^* denote the unique optimal solution of problem (3), then the optimal decision function can be written as:

$$f(x; w^*, b^*) = f(x; \alpha^*, b^*) = \left(\sum_{i=1}^{\ell} y^i \alpha_i^* x^i \right)^T x + b^*, \tag{4}$$

where $w^* = \sum_{i=1}^{\ell} y^i \alpha_i^* x^i$. Observe that b does not appear in the dual problem, so b^* must be found by making use of the primal constraints and the complementary conditions, i.e., b^* is chosen so that:

$$y^i (f(x^i; \alpha^*, b^*) = 1, \quad \forall i : 0 < \alpha_i^* < C. \tag{5}$$

Solving the quadratic problem (3) reveals the optimal hyperplane in $\mathcal{X}$ with respect to the given training set Z and penalty factor C. In order to learn a nonlinear decision function for a linearly inseparable training set, a common strategy [17] involves mapping the representation of the data from $\mathcal{X}$ to a feature space $\mathcal{F}$:

$$x = (x_1, x_2, \ldots, x_n) \mapsto \Phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_N(x)).$$

Here $\phi_i(x) : \mathcal{X} \to \mathcal{F}, i \in \{1, \cdots, N\}$ are usually nonlinear functions representing the new features in the N dimensional feature space $\mathcal{F}$. Then, we find a hyperplane in the feature space $\mathcal{F}$. To simplify the notation, we still use w as the weight vector and b the bias in the feature space $\mathcal{F}$. The primal

problem to be solved in the space $\mathcal{F}$ is:

$$\begin{array}{lll} \min\limits_{w,b} & \frac{1}{2}w^T w + C\xi^T e & \\ \text{s.t.} & y^i(w^T\Phi(x^i)+b) \;\geq\; 1-\xi_i, & i=1,\ldots,\ell \\ & \xi_i \;\geq\; 0, & i=1,\ldots,\ell, \end{array} \tag{6}$$

where $w \in R^N$ is the weight vector of the separating hyperplane. If w^* and b^* are the optimal solution of (6), then the optimal decision function in primal form in $\mathcal{F}$ can be expressed as:

$$f(x; w^*, b^*) = \sum_{i=1}^{N} w_i^* \phi_i(x) + b^* = (w^*)^T\Phi(x) + b^*. \tag{7}$$

The corresponding dual problem in $\mathcal{F}$ is:

$$\begin{array}{lll} \max\limits_{\alpha} & e^T\alpha - \frac{1}{2}\sum\limits_{i,j=1}^{\ell} y^i y^j \alpha_i \alpha_j (\Phi(x^i)^T\Phi(x^j)) & \\ \text{s.t.} & y^T\alpha \;=\; 0, & \\ & Ce \;\geq\; \alpha \geq \mathbf{0}. & \end{array} \tag{8}$$

Note that in the above process, we take indeed two steps to build a nonlinear decision boundary in an original input space $\mathcal{X}$. The first step is to map the data representation from $\mathcal{X}$ into $\mathcal{F}$ by $\Phi(x)$. Then, in the second step, a linear learning machine is applied to find the optimal hyperplane in $\mathcal{F}$. An easy way to combine these two steps into one is to compute the kernel matrix $\mathcal{K}$ with $\mathcal{K}_{ij} = \Phi(x^i)^T\Phi(x^j)$, then solve the corresponding quadratic optimization problem to find the classifier in the feature space $\mathcal{F}$. Such a method is called a *kernel* method [7].

Definition 2.1 *A function $K : R^n \times R^n \to R$ is a kernel function if there exists a function $\Phi(x) : R^n \to R$ such that*

$$K(x,z) = \Phi(x)^T\Phi(z), \quad \forall x, z \in \mathcal{X}.$$

It can be proved that any functions $K : R^n \times R^n \to R$ is a kernel function if the corresponding kernel matrix $\mathcal{K}$ is positive semidefinite [7].

The following are few sample kernel functions [17]:

1. The Gaussian kernel $K(x, z; \sigma) = \exp(-\sigma\|x - z\|^2)$,
2. Polynomial kernel $K(x, z; d) = (x^T z + c)^d$,
3. Sigmoid kernel $K(x, z; \sigma) = \tanh(\sigma \cdot x^T z + 1)$.

Now we can rewrite problem (8) as:

$$\begin{array}{lrcl} \max\limits_{\alpha} & \alpha^T e - \frac{1}{2}\alpha^T \tilde{\mathcal{K}} \alpha & & \\ \text{s.t.} & y^T \alpha & = & 0, \\ & Ce & \geq \alpha \geq & \mathbf{0}, \end{array} \tag{9}$$

where $\tilde{\mathcal{K}}_{ij} = y^i y^j \Phi(x^i)^T \Phi(x^j)$. Note that the dual problem is the same as the problem (3) except that a different kernel matrix $\tilde{\mathcal{K}}$ is used. This makes it clear that the introduction of non-linear mappings does not influence the formulation of the dual problem except for a matrix, which is the only part related to the non-linear mapping.

3 Self-adaptive Support Vector Machines

3.1 SASVMs for Model Selection

The immediate question arisen from the introduction of kernel functions is how to decide the suitable values for kernel parameters and what principle we should follow to choose the values for kernel parameters. In this section, we provide partial answers to these issues. To be more specific, we restrict our discussion to the case of SVMs where the kernel functions are parameterized

by σ. For convenience, the notation $f(x;\sigma)$ means that f is a function of x with a parameter σ.

For any fixed kernel parameter σ, problem (9) is a standard quadratic problem that can be solved by several standard optimization solvers like LOQO [16], MINOS [5] etc. Suppose that σ can take any values in a certain range. Then the number of corresponding non-linear mappings becomes infinite. Let us denote the whole set of mappings by $\mathcal{M}$. We concentrate on the following model selection problem: find the particular mapping in $\mathcal{M}$ that minimizes $\|w^*\|^2 + C\xi e$, which is the objective in a SVM model with a fixed kernel. This leads to the following bi-level optimization problem:

$$\begin{array}{lll} \min\limits_{\Phi(x)\in\mathcal{M}} & (\tilde{w}^*)^T\tilde{w}^* + C\xi^T e & \\ \text{s.t.} & w^* = \arg\min\limits_{w,b} & w^T w + C\xi^T e \\ & & \text{s.t.} \quad y^i(w^T\Phi(x^i)+b) \geq 1-\xi_i, \quad i=1,\ldots,\ell, \\ & & \qquad\quad \xi_i \geq 0, \qquad\quad i=1,\ldots,\ell. \end{array} \tag{10}$$

The upper level problem is usually called the leader's problem, and the lower level problem the follower's problem. In our setting (10), the decision variable of the leader's problem is the mapping $\Phi(x)$ drawn from the whole space of mappings $\mathcal{M}$, and the variables of the follower's problem are w in appropriate dimensions and b.

However, although we have derived problem (10) for model selection, it is nontrivial to solve it. The difficulty comes from two aspects. First, the follower's problem is not defined completely until the mapping is given explicitly. In other words, when the mappings are different, the formulations of the follower's problems are different as well. Secondly, it is unclear how to find a descent search direction in the mapping space $\mathcal{M}$. These difficulties can be released partially if we consider the dual of the follower's problem, as

we can define explicitly the optimal value of the dual problem as a function of the parameter σ. Correspondingly, we rewrite problem (10) by means of the parameter σ with the follower's problem replaced by its dual problem

$$\begin{array}{ll} \min\limits_{\sigma} & F(\sigma, \alpha(\sigma)) = \alpha^T(\sigma)e - \frac{1}{2}\alpha^T(\sigma)\tilde{\mathcal{K}}(\sigma)\alpha(\sigma) \\ \text{s.t.} & \alpha(\sigma) = \arg\max\limits_{\alpha} \quad \alpha^T e - \frac{1}{2}\alpha^T\tilde{\mathcal{K}}(\sigma)\alpha \\ & \qquad\qquad \text{s.t.} \quad y^T\alpha = 0, \\ & \qquad\qquad\qquad Ce \geq \alpha \geq \mathbf{0}, \end{array} \tag{11}$$

where $\tilde{\mathcal{K}}(\sigma)$ is a matrix parametric in σ, $(\tilde{\mathcal{K}}(\sigma))_{ij} = K(x^i, x^j; \sigma)$. Now, instead of minimizing over a space of mappings, we can seek the optimal kernel parameter σ that minimize the objective in (11). We call it the *bi-level problem (BLP)* for SASVMs.

By solving the BLP of SASVMs for a given kernel function type and the penalty factor C, we are simultaneously tuning σ, α and b to appropriate values. In problem (11), we treat the penalty factor C as a given constant. It is natural to deal with the kernel parameter and penalty factor separately because they play different roles in machine learning. The kernel parameter affects the underlying mapping from $\mathcal{X}$ to $\mathcal{F}$, and thus the VC dimension of the hypothesis space [17], while the penalty factor maintains the tradeoff between the maximal margin and the number of the misclassifications. If the classification task is critical and the cost of misclassification is very expensive, then we should use a relatively larger penalty parameter to reduce the misclassification error.

3.2 SASVMs for Feature Selection

For the feature selection, we follow the idea introduced in [4]. A vector $\beta \in R^n$ is applied to describe the contributions of features to the final decision

boundary. A small value of an element of β means that the corresponding feature does not contribute much to the classification, consequently we can remove it from the data set without affecting the classification accuracy significantly.

The vector β can be integrated into the kernel functions. For instance, the Gaussian kernel functions can be extended as

$$K_{\text{ext}}(x^i, x^j; \sigma; \beta) = \exp\left(-\frac{\sum_{k=1}^{n} \beta_k (x_k^i - x_k^j)^2}{2\sigma_k^2}\right). \tag{12}$$

Note that setting the vector β equal to 1 gives rise to the original Gaussian RBF kernel function. Similarly, polynomial kernels can also be extended as follows:

$$K_{\text{ext}}(x^i, x^j; d; \beta) = \left(\sum_{k=1}^{n} \beta_k (x_k^i x_k^j) + c\right)^d, \tag{13}$$

and linear kernel function can be extended as:

$$K_{\text{ext}}(x^i, x^j; \beta) = \sum_{k=1}^{n} \beta_k (x_k^i x_k^j). \tag{14}$$

However, there is no guarantee that the resulting kernel matrices are still positive semidefinite. The indefiniteness of the extended kernel matrix creates extra difficulty in solving the follower's problem as the resulting follower's problem is no longer convex.

4 Solving the BLP of SASVMs

4.1 Converting Into an Equivalent One-level Problem

One way to solve problem (11) is to convert the bi-level problem into a one-level problem by replacing the follower's problem by its corresponding KKT conditions. This approach is valid because, due to the convexity of the follower's problem, replacing the follower's problem by its corresponding KKT

conditions won't change the value of the objective at the upper level. Consequently, we obtain the following optimization problem with complementary constraints

$$\begin{aligned} \min_{\alpha,\sigma,r,u,v} \quad & F(\sigma,\alpha(\sigma)) = e^T\alpha - \tfrac{1}{2}\bar{\alpha}^T\mathcal{K}(\alpha)\bar{\alpha} \\ \text{s.t.} \quad & e - \bar{y} + ry + u - v = \mathbf{0}, \\ & v^T(Ce-\alpha) = 0, \\ & u^T\alpha = 0, \\ & y^T\alpha = 0, \\ & Ce \geq \alpha, \\ & \alpha \geq \mathbf{0}, \\ & \sigma \geq \mathbf{0}, \\ & u \geq \mathbf{0}, \\ & v \geq \mathbf{0}. \end{aligned} \tag{15}$$

For a training set of size ℓ, problem (15) has $\ell + 2$ non-linear equality constraints, one linear constraint, together with non-negativity and box constrains on the decision variables. It is not easy to solve this problem especially when the size of a training set is large.

Several non-linear optimization solvers including MINOS and LOQO are tested on a training set of 100 points of two classes scattered in a 2-dimensional space under the AMPL [9] environment. Among those solvers, only LOQO managed to solve this problem successfully in a reasonable CPU time. Our experiment also shows that, even for a training set with 200 points in two classes, problem (15) can not be solved efficiently by LOQO. However, in the real world, it is quite common to have to have a training set with size more than 1000 points. Hence from a viewpoint of practical application and efficiency, we have to find more efficient ways to solve the BLP of SASVMs.

4.2 Derivative Free Approach

The Derivative Free Optimization (DFO) algorithms are a special class of algorithms that are designed to solve the problem

$$\min_{x \in R^n} f(x),$$

for the case where f is a complex function whose gradient $\nabla f(x)$ is not available or very expensive to compute. For example, when $f(x)$ is not explicitly defined or $f(x)$ can only be computed via very complex computer simulation, then we have to refer to DFO because DFO requires only the objective values at limited number of points. For a basic introduction to DFO methods, the readers are referred to [6, 8]

For the bi-level problems of (11), the objective in the leader's problem can be evaluated by solving the follower's problem. Further, if we restrict us to the model selection problem with Gaussian kernel functions only, then problem (11) reduces to an optimization problem in R at the upper level. In case of feature selection, the size of the optimization problem at the upper level is exactly the same as the number of the features in $\mathcal{X}$. Since in most cases the number of the features in $\mathcal{X}$ is much less that the number of instances in a training set, the size of the problem at the upper level is usually small. The main effort is to solve the standard quadratic optimization problems. These facts motivate us to use the DFO approach to solve the BLPs of SASVMs. We mention that in our experiments, we use the DFO package developed in the thesis [8].

5 Experimental Results

5.1 Hand Written Digit Recognition

In this section, we report our experimental results based on SASVMs. We first test the USPS handwritten digits database [12], a well known data set in the machine learning community. This data set contains normalized grey scale images of size 16×16, divided into a training set of 7291 images and a test set of 2007 images. A human error rate estimated to be 2.5% indicates that it is a hard recognition task.

Only RBF kernels are tested for this data set. This is because, as observed by Vapnik [17], the performance of SVMs based on polynomial kernels and RBF kernels are quite similar while RBF kernels provide a good chance for kernel parameter tuning due to the introduction of the parameter σ. Table 1 shows the performance of SASVM on the USPS data set at different penalty values. We can see that if the penalty is too big, SASVM tends to overfit the training set. Hence, even though SASVM can change the kernel parameter automatically, the user still need to control the choice of the penalty parameter to achieve the best possible generalized performance. The lowest error rate of 4.73% achieved by SASVM is close to the error rate of 4.1% achieved by other SVM classifers [17]. This shows that our bi-level approach provides an efficient way to tune the kernel parameter.

Table 1: SASVM Overall Performance on USPS Data Set

Penalty C	0.1	1	10	13	17	25	50	100
Training accuracy	96.04	99.79	99.99	99.99	99.99	99.99	1	1
Testing accuracy	91.98	94.92	95.27	95.27	95.27	95.22	95.12	95.12

5.2 Breast Cancer Diagnosis

The breast cancer diagnosis data set [15] is another commonly used data set in machine learning. It contains 569 instances, among which 357 instances are in the class "benign" and the remaining instances are "malignant". There are totally 30 features, which are all real values, computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. The target of learning is to distinguish benign from malignant ones masses. The best predictive accuracy of 97.5% [1] is obtained using one separating plane in the space formed from only 3 features (Worst Area, Worst Smoothness and Mean Texture).

Following the work done by [1], we used the repeated 10-fold cross validation to estimate the performance of SASVM on this data set. Since SASVM performs poorly on the 3 features used in [1], we decided to use all the 30 features. For each fold, different penalties were tried until the best testing accuracy was found. Table 2 shows the accuracy on the training and testing set at each fold and the corresponding penalty. The average accuracy on the testing set is 94.56%.

Table 2: SASVM Performance on Breast Cancer Data Set

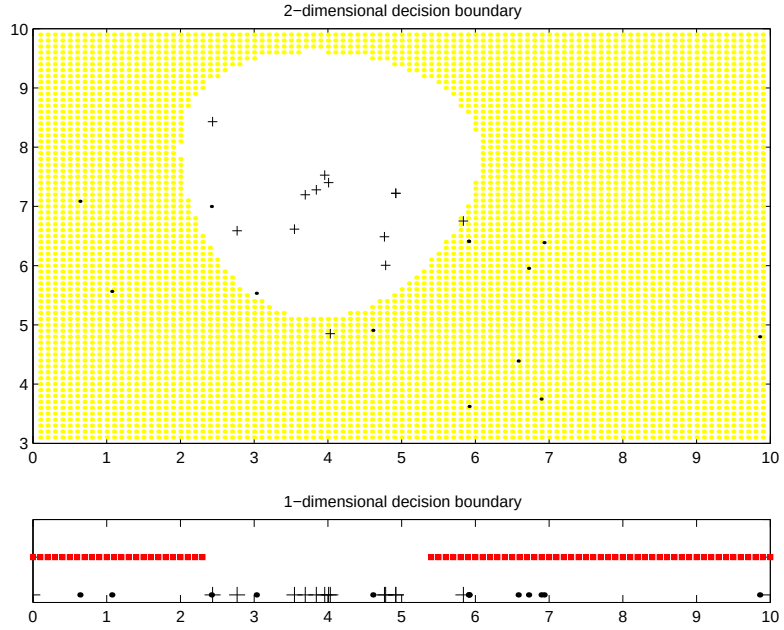
Fold Number	1	2	3	4	5	6	7	8	9	10
Penalty C	2.0	1.0	5	0.5	0.5	1.0	1.0	0.3	0.3	1.0
Training accur.	96.48	95.31	1	94.34	94.14	94.74	94.53	92.77	93.25	95.34
Testing accur.	92.98	92.98	92.98	94.74	96.49	92.98	96.49	98.25	91.23	96.49

5.3 Feature Selection Experiment

The second data set we examined is a data set of 26 points scattered in 2-dimensional space. The corresponding bi-level problem is converted into

a one-level problem by the means of KKT condition replacement. We find the optimal values for the kernel parameter found by LOQO [16] as $\sigma^* = [0.575913, 1.5386e^{-9}]^T$. The kernel parameter associated with the second feature is very small. Thus, we remove this feature from the training set and run the optimization process again. As expected, the optimal kernel value for the remaining feature is unchanged. This verifies our observation that the second feature does not play a significant role in defining the decision boundary. Figure 1 displays the optimal decision boundaries for

Figure 1: Optimal Decision Boundaries in R^2 and R



this data set as it is found by LOQO before and after feature selection. The upper part of Figure 1 displays the optimal boundary in the original 2-dimensional space where a common kernel parameter is used for both features. The generalization accuracy as estimated by SVM^{Light} [11] is 61.54%. The lower part of Figure 1 displays the decision boundary in 1-dimensional

space, where the points are projected onto 1-dimensional space. The number of misclassifications made by the 1-dimensional decision function are 3, one less than the misclassifications made by the 2-dimensional decision function. Meanwhile, the 1-dimensional decision function is simpler than the one in the 2-dimensional space in the sense it involved only one feature. The generalization accuracy in this situation is 76.92%, which is much higher than the generalization accuracy achieved in the 2-dimensional space.

6 Conclusions and Future Work

We have proposed a bi-level optimization model for the problem of model and feature selection in SVMs. By solving the reformulated bi-level optimization problem, we can automatically tune the parameters used in the kernel functions to find the optimal model in SVMs. This provides a systematic way of learning from the training set. It is worthwhile mentioning that in our bi-level model, we use directly the solution obtained from the follower's problem as the objective at the upper level. This avoids the estimation of the generalization performance.

Two approaches for solving the BLP of SASVMs are examined. Our limited experiments indicate that the direct algorithm based on the KKT system for the follower's problem is feasible only for small-size problems, while the DFO method provides an efficient approach for attacking the BLP of SASVMs.

Experiments also show that the performance of SASVM is significantly influenced by the choice of the penalty parameter. This issue is not addressed in the present SASVM package. It is natural to consider whether we can put the penalty factor into our bi-level optimization framework. More study

is still needed to find a suitable value for the penalty parameter.

Finally, we point out that although SASVM can deal with the problem of model and feature selection, there are still challenges ahead, in particular regarding to feature selection. For example, it is worthwhile investigating under what conditions the extended kernel matrix is still positive semidefinite. We hope future study can help us to address these questions.

References

[1] K.P. Bennett, *Decision tree construction via linear programming*, Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp 97-102, 1992.

[2] P.S. Bradley, O.L. Mangasarian, and W.N. Street, *Feature selection via mathematical programming*, INFORMS Journal on Computing **10** (1998), 209–217.

[3] O. Chapelle and V. Vapnik, *Model selection for support vector machines*, Advances in Neural Information Processing System (T. K. Leen S. A. Solla and K. R. Muller, eds.), vol. 12, MIT Press, 2000.

[4] ———, *Choosing multiple parameters for support vector machine*, Machine Learning **46** (2002), 131–159.

[5] J.W. Chinneck, *MINOS(IIS): Infeasibility analysis using MINOS*, Computers and Operations Research **21** (1994), no. 1, 1–9.

[6] A. Conn, K. Scheinberg, and Ph.L. Toint, *Recent progress in unconstrained nonlinear optimization without drivatives*, Mathematical Programming **79** (1997).

[7] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machine*, Cambridge University Press, 2000.

[8] E. Fan, *Global optimization of Lennard-Jones atomic clusters*, Master Thesis, Department of Computing and Software, McMaster University, 2002.

[9] R. Fourer, D. Gay, and B. Kernighan, *AMPL: A mathematical programming language*, Duxbury Press/Brooks/Cole Publishing Company, 2002.

[10] T.S. Jaakola and D. Haussler, *Exploiting generative models in discriminative classifiers*, Advances in Neural Infromation Processing Systems (Cambridge MA, USA) (S.A. Solla M.S. Kearns and D.A. Cohn, eds.), MIT Press, 1998.

[11] T. Joachims, *Estimating the generalization performance of a svm efficiently*, Proceedings of ICML-00, 17th International Conference on Machine Learning (Stanford, US) (Pat Langley, ed.), Morgan Kaufmann Publishers, San Francisco, US, 2000, pp. 431–438.

[12] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.J. Jackel, *Handwritten digit recognition with backpropagation network*, Advances in Neural Information Processing Systems 2, Morgan Kaufman, 1990.

[13] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, *Comparison of learning algorithms for handwritten digit recogni-*

tion, In: *International Conference on Artificial Neural Networks* (F. Fogelman and P. Gallinari, eds.), 1995, pp. 53–60.

[14] M. Pontil and A. Verri, *Object recognition with support vector machines*, IEEE Trans. On PAMI **20** (1998), 637–646.

[15] W.N. Street, W.H. Wolberg, and O.L. Mangasarian, *Nuclear feature extraction for breast tumor diagnosis*, IS&T/SPIE: International Symposium on Electronic Imaging: Science and Technology (San Jose, CA), vol. 1905, 1993.

[16] R.J. Vanderbei, *LOQO: An interior point code for quadratic programming*, Optimization Methods and Software **11** (1999), 451–484.

[17] V. Vapnik, *The nature of statistical learning theory*, Springer-Verlag, New York, US, 1999.

[18] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, *Feature selection for SVMs*, NIPS (2000), 668–674.