Multi-Objective Programming in SVMs

Jinbo Bi BIJ2@RPI.EDU
Department of Mathematical Sicences, Rensselaer Polytechnic Institute, Troy, NY 12180 USA

Abstract

We propose a general framework for support vector machines (SVM) based on the principle of multi-objective optimization. The learning of SVMs is formulated as a multi-objective program (MOP) by setting two competing goals to minimize the empirical risk and minimize the model capacity. Distinct approaches to solving the MOP introduce various SVM formulations. The proposed framework enables a more effective minimization of the VC bound on the generalization risk. We develop a feature selection approach based on the MOP framework and demonstrate its effectiveness on hand-written digit data.

1. Introduction

We examine the learning process of finding a function $f \in \mathcal{F}$ that minimizes the generalization risk where $\mathcal{F}$ is a set of possible functions. For classification problems, which we will focus on in this paper, the generalization risk of a given decision boundary f is defined as the probability that a data point is misclassified using the decision model constructed on f. The empirical risk is computed as the misclassification rate of f on sample data. An upper bound on the generalization risk $R(f)$ in VC theory (Vapnik, 1998) typically takes a form as

$$R_{emp}(f) + \Psi(\frac{h}{\ell}) \tag{1}$$

where $R_{emp}(f)$ is the empirical risk for a given function f chosen from $\mathcal{F}$, h is a measure of the capacity of $\mathcal{F}$, named as the VC dimension of $\mathcal{F}$, and ℓ is the amount of training data. The function Ψ is basically a monotonically increasing function in terms of the ratio h/ℓ.

The bounds suggest us that to achieve a small generalization risk, the learning process prefers a small empirical risk and a small capacity of $\mathcal{F}$. The VC dimension h is the best-known measure of the capacity of $\mathcal{F}$. Better complexity measures exist, but are usually more difficult to evaluate. The VC dimension is more applicable to manipulating under practical and algorithmic circumstances. Therefore the small capacity can be obtained by minimizing the VC dimension of $\mathcal{F}$. The goal of learning processes thus becomes to minimize both the empirical risk and the VC dimension. However, they are conflicting goals in the sense that when h is small, the empirical risk may be large due to insufficient learning. In contrast, the learning machine may require a large h to obtain a small empirical risk, and may suffer the "overfitting" phenomenon.

Multi-objective programming is an optimization technique for solving problems with multiple conflicting goals. Mathematically, objectives are said to be conflicting if optimal solutions corresponding to each individual objective are not the same within the feasible region. A multi-objective program (MOP) for the learning process can be formulated in principle as follows:

$$\begin{array}{ll} \min & R_{emp}(f) \\ \min & h(\mathcal{F}) \\ \text{s.t.} & f \in \mathcal{F}. \end{array} \tag{2}$$

Note that the function class $\mathcal{F}$ itself is an adjustable variable in the MOP (2) because altering the VC dimension of $\mathcal{F}$ has impact on the choices of $\mathcal{F}$. Usually we define a type of possible functions beforehand, for example, consider the linear functions. The function class $\mathcal{F}$ is a subset of linear functions which presents the desired VC dimension. The concrete formulations of the multi-objective optimization can be derived by specifying the type of the functions in $\mathcal{F}$, the computation of the empirical risk and an estimate of the VC dimension. Distinct specifications introduce variants of multi-objective programs (MOP). When a MOP is successfully formulated as desired, the question arises as to how we can solve it. A lot of research has been devoted to the study of various approaches to solving MOPs. Depending on the specific MOPs, appropriate techniques can be developed and used to solve them. Traditional methods for solving MOPs include the weighted sum method, the constraint method, weighted metric methods, goal programming methods, etc. Later around evolutionary algorithms become popular for finding solutions to MOPs. The classic SVM with the hyper parameter C, derived using the weighted sum method, is just one way to solve the MOP. Under the MOP framework, we propose a feature selection approach that allows the learning process to reduce the dimensionality of the problem without losing prediction accuracy.

For a MOP with two conflicting objectives as given in Problem (2), each objective corresponds to a different optimal solution. We have to find a compromise in the objectives. The fundamental difference between single- and multi-objective optimization is that for a MOP, we can find a set of optimal solutions where no single solution can be said to be better than any other. Solving a MOP often implies to search for the set of optimal solutions as opposed to a lone solution for a single-objective program. For a learning process, we do not need to spread the entire set of optimal solutions because VC bounds can provide information to help us locate the best compromise.

In the next section, we briefly review the principle of multi-objective optimization and traditional approaches to solving a MOP. A concrete MOP formulation based on the above framework (2) is rigorously derived and analyzed in Section 3. Employing distinct methods to solve the proposed MOP with proper simplifications yields various learning algorithms, including the classic SVM (C-SVM) and rigorous SVM (RSVM) as discussed in Section 4. In Section 5, we develop an approximation scheme for solving the proposed MOP without simplifications, aimed at producing acceptable solutions rather than optimal solutions. Based on the MOP framework, we propose a feature selection approach in Section 6 and demonstrate that it can reduce the dimensionality without loss of prediction accuracy in Section 7.

2. Review of MOP

In this section, we state the MOP with two objectives in its general form

$$\begin{array}{ll} \min_{\mathbf{x}} & Obj_1(\mathbf{x}) \\ \min_{\mathbf{x}} & Obj_2(\mathbf{x}) \\ \text{s.t.} & \mathbf{g}(\mathbf{x}) \le \mathbf{0},\ \mathbf{d}(\mathbf{x}) = \mathbf{0}, \\ & \mathbf{x}^{low} \le \mathbf{x} \le \mathbf{x}^{up}. \end{array} \tag{3}$$

The bold face of a lower-case letter indicates that it is a vector. Here $\mathbf{g}$ and $\mathbf{d}$ are vectors of functions of appropriate dimension, and $\mathbf{x}^{low}$, $\mathbf{x}^{up}$ are the lower and upper bounds on $\mathbf{x} \in \mathbb{R}^n$. All constraints together define the feasible region $F = \{\mathbf{x} : \mathbf{g}(\mathbf{x}) \le 0,\ \mathbf{d}(\mathbf{x}) = \mathbf{0},\ \mathbf{x}^{low} \le \mathbf{x} \le \mathbf{x}^{up}\}$. We introduce the concepts of domination and Pareto-optimality.

2.1. Pareto-optimality

A solution $\mathbf{x}^1$ is said to *dominate* another solution $\mathbf{x}^2$ (Eschenauer et al., 1986), if 1. the solution $\mathbf{x}^1$ is no worse than $\mathbf{x}^2$ in all objectives, i.e., $Obj_1(\mathbf{x}^1) \le Obj_1(\mathbf{x}^2)$ and $Obj_2(\mathbf{x}^1) \le Obj_2(\mathbf{x}^2)$; 2. the solution $\mathbf{x}^1$ is strictly better than $\mathbf{x}^2$ in at least one objective, i.e., $\exists\ i \in \{1, 2\}$, $Obj_i(\mathbf{x}^1) < Obj_i(\mathbf{x}^2)$.

A feasible point is a *Pareto-optimal* or *non-dominated* solution if the point is not dominated by any other point in the feasible set F. Typically the Pareto-optimal set consists of all Pareto-optimal solutions. We use a toy example to illustrate the definitions as in Figure 1(above). This problem has two quadratic objective functions, and the Pareto-optimal set is the interval $[A, B]$. In general, the Pareto-optimal frontier in the objective function space is used to illustrate the optimality in the MOP context (Figure 1(below)). Each point in the figure represents a pair of objective values corresponding to a $\mathbf{x} \in F$. The filled circles correspond to Pareto-optimal solutions and they fall on the Pareto-optimal frontier.

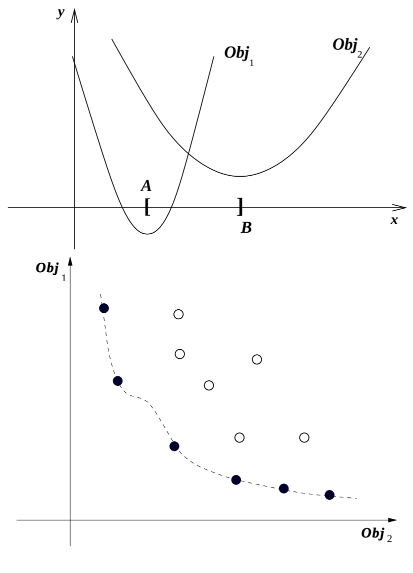

Figure 1. Above: The Pareto-optimal set of two quadratic objective functions is $[A, B]$. *Below:* The Pareto-optimal frontier of two objective functions in general.

The question arises as how we can find a Pareto-optimal solution to a MOP. Traditional methods avoid the inherent complexity in a multi-objective program and convert multiple objectives into a single objective by using certain schemes and user-specified parameters. Many studies compare different methods of such conversions, and provide reasons in favor of one conversion over another. We describe two simple and widely-used methods for such conversions. They will serve as the basis of our approaches to solving the MOP arisen in learning problems in the later sections.

2.2. Traditional methods

The weighted sum method transform two objectives into a single objective by multiplying each objective with a pre-defined weight and adding them together. The weight of an objective is usually chosen in proportion to the objective's relative importance in the problem. Determining an appropriate weight vector also relies on the scaling of each objective function. Usually the objectives are scaled appropriately so that each has the same order of magnitude when choosing the weights. The composite objective function can thus be written as

$$\min\{c_1 Obj_1(\mathbf{x}) + c_2 Obj_2(\mathbf{x}) : \mathbf{x} \in F\} \tag{4}$$

where the weights c_1 and c_2 are non-negative and at least one of them is not zero.

Solving Problem (4) yields Pareto-optimal solutions if the weight is positive for both objectives. Different weight vectors do not necessarily lead to different Pareto-optimal solutions. It does not imply that any Pareto-optimal solution can be obtained by using a positive weight vector unless the MOP is convex. A MOP (3) is convex if all objective functions are convex as well as the feasible region is convex, (equivalently, all inequality constraints are convex and equality constraints are linear). For any Pareto-optimal solution $\hat{\mathbf{x}}$ to a convex MOP, there exists a positive weight vector $\mathbf{c}$ such that $\hat{\mathbf{x}}$ is a solution to Problem (4).

If the MOP is not convex, the Pareto-optimal frontier may have non-convex portions as shown in Figure 1 (below, the dotted line). The non-convex parts of the Pareto-optimal set cannot be obtained by minimizing the combinations of the objectives as formed in Problem (4). To alleviate the difficulties faced by the weighted sum approach in solving problems with non-convex pattern, the constraint method was proposed. The MOP is reformulated by keeping one of the objectives and restricting the rest of the objectives within user-specified values. For instance, if we treat the second objective in MOP (3) by a constraint, the modified problem becomes

$$\min\{Obj_1(\mathbf{x}) : Obj_2(\mathbf{x}) \le \delta,\ \mathbf{x} \in F\}. \tag{5}$$

Any Pareto-optimal solution of a MOP can be obtained by solving the constraint problem (5) for a proper upper bound δ regardless of the non-convexity of the Pareto-optimal frontier. One disadvantage for this method is that the solution to the problem (5) largely depends on δ which has to be chosen within the minimal or maximal value of the objective.

Other approaches to solving MOPs include the weighted metric methods, Benson's method, goal programming methods, and some interactive methods. Evolutionary algorithms are popular tools for solving multi-objective optimization. They all have advantages and disadvantages in one way or another.

3. The Concrete MOP Formulation

A concrete formulation of the MOP (2) can be derived by specifying how to calculate the $R_{emp}(f)$ (the first objective) and how to estimate the VC dimension $h(\mathcal{F})$ (the second objective). These specifications depend on the definition of the set of possible functions $\mathcal{F}$.

SVMs construct decision models based on linear functions. Nonlinear models can be obtained via the so-called kernel substitution. By using a kernel, the original input vector $\mathbf{x}_i$ is transformed to $\mathbf{z}_i = \Phi(\mathbf{x}_i)$ which is in a usually high dimensional feature space denoted as $\mathbb{Z}$. A kernel function in the input space corresponds to an inner product in the corresponding feature space. The feature space is uniquely determined by the kernel function and its parameters. For example, a common type of kernels is polynomials, and the parameter for this type of kernels is the order of the polynomial. Let us generally denote a kernel by $k_{\mathbf{s}}(\cdot,\cdot)$ with parameter(s) $\mathbf{s}$, and the corresponding mapping operator by $\Phi_{\mathbf{s}}$. Note that for a given type of kernels, the feature space or the mapping is solely dependent on the choices of the parameter(s) $\mathbf{s}$.

To construct a decision model, we first describe the smallest ball containing all transformed vectors $\Phi_{\mathbf{s}}(\mathbf{x}_i)$. Assume that the transformed vectors are centered to have mean $\mathbf{0}$. This can always be done by an appropriate transformation of the kernel. For an arbitrary kernel $k(\mathbf{x}, \tilde{\mathbf{x}})$ and the corresponding feature space $\mathbb{Z}$, the following kernel

$$\begin{aligned}\hat{k}(\mathbf{x}, \tilde{\mathbf{x}}) = k(\mathbf{x}, \tilde{\mathbf{x}}) - \tfrac{1}{\ell}\textstyle\sum_{i=1}^{\ell} k(\mathbf{x}, \mathbf{x}_i)\\ -\tfrac{1}{\ell}\textstyle\sum_{i=1}^{\ell} k(\mathbf{x}_i, \tilde{\mathbf{x}}) + \tfrac{1}{\ell^2}\textstyle\sum_{i,j=1}^{\ell} k(\mathbf{x}_i, \mathbf{x}_j).\end{aligned}$$

maps the input vectors to vectors in $\mathbb{Z}$ with mean $\mathbf{0}$. We then approximately look for the ball $B_R = \{\mathbf{z} \in \mathbb{Z} : (\mathbf{z} \cdot \mathbf{z}) \le R\}$ with center at the origin and the smallest possible radius $\sqrt{R}$, which contains the images $\Phi_{\mathbf{s}}(\mathbf{x}_i),\ i = 1, \cdots, \ell$. This implies that for any i,

$(\Phi_{\mathbf{s}}(\mathbf{x}_i) \cdot \Phi_{\mathbf{s}}(\mathbf{x}_i)) = k_{\mathbf{s}}(\mathbf{x}_i, \mathbf{x}_i) \leq R$.

In the feature space $\mathbb{Z}$ determined by $k_{\mathbf{s}}$, consider the set of hyperplanes $\{\mathbf{z} \in \mathbb{Z} : (\mathbf{w} \cdot \mathbf{z}) + b = 0\}$ that separate the transformed data $(\Phi_{\mathbf{s}}(\mathbf{x}_i), y_i)$ with a margin, i.e., satisfy $\min_{i=1,\cdots,\ell} |(\mathbf{w} \cdot \Phi_{\mathbf{s}}(\mathbf{x}_i)) + b| = 1$ with $(\mathbf{w} \cdot \mathbf{w}) \leq W$. The margin is calculated as $1/\sqrt{W}$. For any hyperplane in the set of separating hyperplanes, the decision model can be constructed as $g_{\mathbf{w},b} = \operatorname{sgn}((\mathbf{w} \cdot \mathbf{z}) + b)$ where $\mathbf{z} \in B_R$. The domain of the decision model is B_R. The VC dimension h of the set $\{g_{\mathbf{w},b} : (\mathbf{w} \cdot \mathbf{w}) \leq W\}$ has an upper bound as

$$h \leq RW + 1 \tag{6}$$

provided the dimension of the feature space is larger than RW. This is often the case encountered in practice. This upper bound is tight when data vectors are uniformly distributed right on the surface of the ball.

The decision model $g_{\mathbf{w},b}$ classifies the vectors in B_R with the margin $1/\sqrt{W}$. In many practical applications, such a decision model does not exist. To allow for the possibility of errors, the slack variables $\xi_i \geq 0,\ i = 1, \cdots, \ell$ are introduced (Cortes & Vapnik, 1995; Vapnik, 1998) such that

$$y_i\left((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b)\right\} \geq 1 - \xi_i.$$

Now we specify the set of functions $\mathcal{F} = \{g_{\mathbf{w},b} : (\mathbf{w} \cdot \mathbf{w}) \leq W\}$ in the learning process. Then $RW + 1$ can be regarded as an estimate of the VC dimension of $\mathcal{F}$. The empirical risk for this set of functions is computed as $\sum_{i=1}^{\ell} \xi_i$. We thus formulate the MOP in variables $\mathbf{w}$, b, $\boldsymbol{\xi}$, $\mathbf{s}$, W and R as

$$\begin{array}{lll}
\min & \sum_{i=1}^{\ell} \xi_i & (7) \\
\min & RW & (8) \\
\text{s.t.} & y_i\left((\mathbf{w} \cdot \Phi_{\mathbf{s}}(\mathbf{x}_i)) + b\right) \geq 1 - \xi_i, & \\
 & \xi_i \geq 0,\ i = 1, \cdots, \ell, & (9) \\
 & (\mathbf{w} \cdot \mathbf{w}) \leq W, & (10) \\
 & k_{\mathbf{s}}(\mathbf{x}_i, \mathbf{x}_i) \leq R,\ i = 1, \cdots, \ell. & (11)
\end{array}$$

We refer to this problem as the master MOP (MMOP). Note that it is equivalent if we remove the constraint (10) instead write the second objective as $R(\mathbf{w} \cdot \mathbf{w})$. Then the problem has fewer constraints, but a more complicated objective. The question of which way is more computationally efficient is not examined here. The above formulation MMOP was used in our experiments. The MMOP serves as a mechanism to minimize the VC bound (1). Once a solution $(\hat{\mathbf{w}}, \hat{b}, \hat{\mathbf{s}}, \hat{W}, \hat{R}, \hat{\boldsymbol{\xi}})$ is determined, the MOP approach constructs the optimal decision model based on the separating hyperplane $\{\mathbf{z} : (\hat{\mathbf{w}} \cdot \mathbf{z}) + \hat{b} = 0\}$ in the feature space characterized by $k_{\hat{\mathbf{s}}}$. In addition, the model is chosen from the set $\mathcal{F}$ with VC dimension close to $\hat{R}\hat{W} + 1$. In the next section, we investigate approaches to solving the MMOP. By exploiting these approaches, we obtain distinct SVM formulations.

4. Deriving SVM Formulations

As we introduced in Section 2, solving a MOP often involves converting multiple objectives to a single one. To derive the class of SVM algorithms (Boser et al., 1992; Vapnik, 1998) based on the MOP framework, the master MOP has been simplified. For a given kernel with fixed parameter $\mathbf{s}$, the value of $k_{\mathbf{s}}(\mathbf{x}_i, \mathbf{x}_i)$ for any $\mathbf{x}_i$ is correspondingly fixed, and $R = \max\{k_{\mathbf{s}}(\mathbf{x}_i, \mathbf{x}_i),\ i = 1, \cdots, \ell\}$ is a constant. The constraint (11) can then be removed since it does not take effect when optimizing the MMOP. Minimizing the RW is equivalent to directly minimizing $(\mathbf{w} \cdot \mathbf{w})$ and omitting the constraint (10). Now the MMOP is simplified to the following MOP

$$\begin{array}{ll}
\min & \sum_{i=1}^{\ell} \xi_i \\
\min & (\mathbf{w} \cdot \mathbf{w}) \\
\text{s.t.} & y_i\left((\mathbf{w} \cdot \Phi_{\mathbf{s}}(\mathbf{x}_i)) + b\right) \geq 1 - \xi_i, \\
 & \xi_i \geq 0,\ i = 1, \cdots, \ell.
\end{array} \tag{12}$$

Note that the constraints here are linear in terms of $\mathbf{w}$, b and $\boldsymbol{\xi}$. This MOP has convex objectives and linear constraints, so it has convex Pareto frontier.

4.1. Classic SVMs

The weighted sum method becomes a good choice for finding the Pareto-optimal set of MOP (12). Any Pareto-optimal solution to the problem (12) can be obtained by minimizing the composite objective function

$$\frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C \sum_{i=1}^{\ell} \xi_i$$

for an appropriate value of C. This actually provides a new perspective to explain the foundation of SVMs besides the regularization theory (Evgeniou et al., 2000). In practice, we do not need the entire Pareto-optimal set. Instead we search for the particular Pareto-optimal solution that minimizes the bound (1). In other words, C should be tuned in such a way that the obtained model $g_{\mathbf{w},b}$ produces the smallest value of the risk bound (1).

4.2. Rigorous SVMs

Similarly, the constraint method can also be applied to the MOP (12) to explore its Pareto frontier. Suppose we minimize the empirical risk and restrict the VC dimension by forming a constraint on the second objective as $(\mathbf{w}\cdot\mathbf{w}) \le W$. Here W is no longer a variable as in the MMOP, but a user-specified parameter. The resulting optimization problem called "rigorous SVM" (RSVM) (Vapnik, 1998)is stated as follows:

$$\begin{array}{lc} \min & \sum_{i=1}^{\ell} \xi_i \\ \text{s.t.} & (\mathbf{w}\cdot\mathbf{w}) \le W, \\ & y_i\left((\mathbf{w}\cdot\Phi_{\mathbf{s}}(\mathbf{x}_i)) + b\right) \ge 1 - \xi_i, \\ & \xi_i \ge 0,\ i = 1,\cdots,\ell. \end{array} \tag{13}$$

Solutions to RSVM depend on the choices of W. W should be selected within a reasonable range to achieve small values of the risk bound. Basically, the VC dimension is an integer in $[1,\ell]$. A proper range for W can be designated based on the analysis of the bound (1) and the property of VC dimension (Vapnik, 1995).

4.3. More General Cases

In many practical situations, we also adjust the kernel parameter $\mathbf{s}$ in the learning process. The radius of the ball $\sqrt{R}$ may vary when reducing the dimensionality or creating a composite kernel (Lanckriet et al., 2002) where the above simplification may be undesirable. We have to solve the non-convex MMOP itself. It may have non-convex frontier, so the constraint method is more applicable to solving the MMOP than the weighted sum method in order not to miss the opportunity to identify a Pareto-optimal solution. In general, to achieve a Pareto-optimal solution, we can optimize one of the objectives (7) and (8) on all variables $\mathbf{w}$, b, $\boldsymbol{\xi}$, $\mathbf{s}$, R and W with a constraint constructed on the other objective. Unfortunately, the resulting problems suffer the difficulties, such as the unknown mapping Φ and the strong nonlinearity of the problems, and thus may not be practically applicable. Proper approximation schemes are needed to create acceptable solutions, not necessarily Pareto-optimal solutions. We propose such an approximation procedure to simplify the computation at expense of potentially losing optimality.

5. Acceptable Approximation

The scheme is motivated by the constraint method and can be viewed as an approximate way to solve the problem formed by the constraint method. It is an iterative procedure with each iteration consisting of two consecutive steps. The first step is to optimize the empirical risk subject to the constraint on the VC dimension. The second step is to optimize RW with a restriction on the empirical risk. We partition the variables into two groups $(\mathbf{w}, b)$ and $(\mathbf{s}, R, W)$, and the slackness $\boldsymbol{\xi}$ is included in both groups. The empirical risk (7) is optimized on $(\mathbf{w}, b, \boldsymbol{\xi})$ and the VC dimension (8) is optimized on $(\mathbf{s}, R, W, \boldsymbol{\xi})$.

In the first step, we fix $\mathbf{s}$, so the radius parameter of the ball R becomes a constant. Following the same arguments in Section 4, the constraint (11) can be omitted. We restrict $(\mathbf{w}\cdot\mathbf{w})$ within a fix value W. The problem is converted to the RSVM (13) in variables $\mathbf{w}$, b, $\boldsymbol{\xi}$ to minimize the empirical risk. After we obtain the optimal solution $(\mathbf{w}, b, \boldsymbol{\xi})$ with respect to the current values of $\mathbf{s}$ and W, we proceed to the second step. In the second step, we set the variables $\mathbf{w}$ and b to the solution found in the first step. Now the problem optimizes the VC dimension over variables $\mathbf{s}$, R, W and $\boldsymbol{\xi}$. Moreover, the objective (7) is restricted to be no more than the optimal objective value obtained in the first step. Then we use the optimal $\mathbf{s}$ and W obtained in the second step in the next iteration.

The first step focuses on improving the performance of the classification model by minimizing the empirical risk with a fixed VC dimension. The classification model is constructed on a linear function in the feature space particularly defined by $\mathbf{s}$ found in last iteration. By optimizing on $\mathbf{s}$, the second step seeks a feature space, for which a smaller VC dimension can be possibly achieved with the empirical risk preserved. The second step is not aimed directly at enhancing the learning performance because it does not search for a classification model, instead for a kernel function.

Algorithm 1:

1. Initialize $\mathbf{s}^0$ and W^0 with appropriate values. Set $t = 1$.

2. Solve the dual formulation of RSVM (13) (Vapnik, 1998) with the fixed $\mathbf{s}^{t-1}$ and W^{t-1},

$$\begin{array}{lc} \min\limits_{\boldsymbol{\alpha}} & \sqrt{W^{t-1}\sum\limits_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j k_{\mathbf{s}^{t-1}}(\mathbf{x}_i,\mathbf{x}_j)} - \sum\limits_{i=1}^{\ell}\alpha_i \\ \text{s.t.} & \sum_{i=1}^{\ell}\alpha_i y_i = 0, \\ & 0 \le \alpha_i \le 1,\ i = 1,\cdots,\ell, \end{array} \tag{14}$$

and compute the optimal b^t, $\mathbf{w}^t = \sum_{i=1}^{\ell}\alpha_i^t y_i \Phi_{\mathbf{s}^{t-1}}(\mathbf{x}_i)$ where $\boldsymbol{\alpha}^t$ is constructed by dividing the optimal solution $\hat{\boldsymbol{\alpha}}$ to Problem (14) (Vapnik, 1998) by

$$\gamma = \frac{\sqrt{\sum_{i,j=1}^{\ell}\hat{\alpha}_i\hat{\alpha}_j y_i y_j k_{\mathbf{s}^{t-1}}(\mathbf{x}_i,\mathbf{x}_j)}}{\sqrt{W^{t-1}}}.$$

Calculate the corresponding optimal objective value E^t of Primal (13).

3. Substitute the $\mathbf{w}^t$ and b^t into the MMOP, and restrict the first objective to be no more than E^t. Solve

the resulting optimization problem

$$\begin{array}{ll}\min\limits_{\mathbf{s},R,W,\boldsymbol{\xi}} & RW \\ \text{s.t.} & y_i\left(\sum_{j=1}^{\ell}\alpha_j^t y_j k_{\mathbf{s}}(\mathbf{x}_j,\mathbf{x}_i)+b^t\right)\geq 1-\xi_i, \\ & \xi_i\geq 0,\ i=1,\cdots,\ell, \\ & \sum_{i=1}^{\ell}\xi_i\leq E^t, \\ & \sum_{i,j=1}^{\ell}\alpha_i^t\alpha_j^t y_i y_j k_{\mathbf{s}}(\mathbf{x}_i,\mathbf{x}_j)\leq W, \\ & k_{\mathbf{s}}(\mathbf{x}_i,\mathbf{x}_i)\leq R,\ i=1,\cdots,\ell,\end{array} \tag{15}$$

to obtain $\mathbf{s}^t$ and W^t.

4. Determine if more iterations are needed, for instance, if either E^t or $H^t = R^tW^t$ is decreased, set $t = t+1$, and go to Step 2; otherwise, stop.

This scheme does not guarantee to achieve a Pareto-optimal solution to the MMOP due to the decomposition of the variables into two sets, and the partial optimization of each step on only a subset of variables. However Proposition 1 shows that it improves the solution in the way that each iteration produces a model f in $\mathcal{F}$ whose empirical risk is no larger than that at the previous iteration. The VC dimension of $\mathcal{F}$ is no larger than the one at the last iteration. If the algorithm can strictly reduce both objectives to some degree, it identifies acceptable solutions relying on the users preference. Furthermore, this scheme is computationally tractable since it does not require the mapping $\Phi_{\mathbf{s}}$, nor the explicit definition of a kernel as long as the kernel matrix can be computed in terms of $\mathbf{s}$ as a positive semi-definite matrix.

Proposition 1 (Approximation Performance) *Let E^{t-1} and H^{t-1} be the optimal objective values respectively of the first step and the second step at the previous iteration. Let E^t and H^t be the corresponding optimal values at the current iteration. Then we have $E^t \leq E^{t-1}$ and $H^t \leq H^{t-1}$.*

Proof. This proof is based on the fact that the optimal objective value of a problem has to be no worse than the objective value of any feasible point. An iteration of Algorithm 1 starts with solving Problem (14) with fixed $\mathbf{s}^{t-1}$ and W^{t-1}. For the sake of simplicity, we consider the corresponding primal problem (13). Realize that $\mathbf{s}^{t-1}$ and W^{t-1} were obtained by optimizing Problem (15) at last iteration. By examining the constraints of Problem (15), the solution $(\bar{\mathbf{w}} = \sum\alpha_i^{t-1}y_i\Phi_{\mathbf{s}^{t-1}}(\mathbf{x}_i), \bar{b} = b^{t-1})$ is feasible to Problem (13) with $\sum\xi_i \leq E^{t-1}$. Since E^t is the optimal objective value of Problem (13) at the current iteration, $E^t \leq E^{t-1}$. Following the same line of arguments, we can show that $H^t \leq H^{t-1}$ too. ∎

6. Feature Selection

Based on the MOP framework, we propose a feature selection approach aimed at reducing the dimensionality without impairing the model prediction accuracy. The feature selection is performed by associating each feature x_i with a scaling factor s_i. The larger values of s_i indicate more useful variables, and the dimension x_i corresponding to a $s_i = 0$ is vanished in the model. We define a kernel function as $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i'\mathbf{S}\mathbf{x}_j$ where $\mathbf{S}$ is a diagonal matrix with diagonal entries $s_i \geq 0$. The mapping introduced by this kernel can be explicitly expressed as $\mathbf{x} = (x_1\ x_2 \cdots x_n)' \mapsto \sqrt{\mathbf{S}}\mathbf{x} = (\sqrt{s_1}x_1\ \sqrt{s_2}x_2 \cdots \sqrt{s_n}x_n)'$ that defines a feature space. Algorithm 1 constructs decision models based on a function from the set $\{f(\mathbf{x}) = \mathbf{w}'\mathbf{S}^t\mathbf{x} + b : \mathbf{w}'\mathbf{S}^t\mathbf{w} \leq W^t\}$ at the t^{th} iteration. Notice that there is no need to center this kernel since the images of input data have mean $\mathbf{0}$ already if input data are centered.

The feature selection approach, which we call MOPFS, can be regarded as a special case of Algorithm 1 with $k_{\mathbf{s}}(\mathbf{x}_i, \mathbf{x}_j)$ replaced by $\mathbf{x}_i'\mathbf{S}\mathbf{x}_j$ and $\mathbf{s} = (s_1\ s_2 \cdots s_n)'$. Notice that all constraints in Problem (15) become linear in terms of $\mathbf{s}$, so Problem (15) is merely a quadratic program. Step 3 of Algorithm 1 has been slightly modified to fit our goal to reduce the number of features. We minimize the objective $RW + c\sum_{i=1}^n s_i$ where c is chosen as a small number relative to RW so that if two solutions $\mathbf{s}$ exist, the modification prefers the sparse one with a few s_i non-zero. In order to take into account the result obtained at previous iterations, we transform input data by $\mathbf{x}_i = \sqrt{\mathbf{S}^{t-1}}\mathbf{x}_i$ in problem (15), and solve problem (15) gives the optimal $\hat{\mathbf{S}}$. Then the actual scaling matrix at the t^{th} iteration becomes $\mathbf{S}^t = \hat{\mathbf{S}}\mathbf{S}^{t-1}$ in terms of original data. Suppose that the algorithm runs T iterations. The final scaling matrix $\mathbf{S}^T = \hat{\mathbf{S}}^T \cdots \hat{\mathbf{S}}^1\mathbf{S}^0$ where the initial scaling matrix $\mathbf{S}^0 = \mathbf{I}$, an identity matrix of appropriate dimension.

7. Computational Results

The goals of our experimental study were to assess the generalization performance and computational behaviors of the proposed MOPFS approach, and compare the approach to other feature selection methods. Other methods include three filter methods and a SVM-based feature elimination approach called VS-SSVM (Bi et al., 2003). The filter methods chose the same amount of features as that in MOPFS according to Pearson correlation coefficients, Fisher criterion scores and Kolmogorov-Smirnov (KS) test (We-

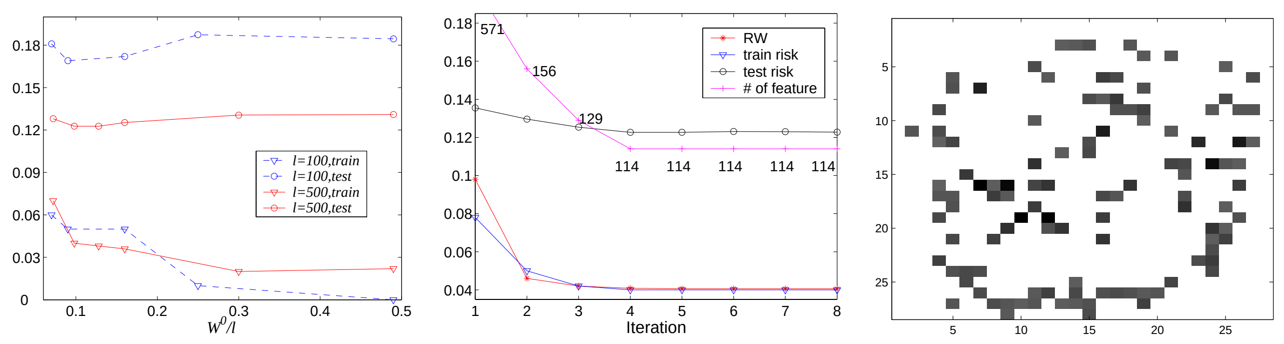

Figure 2. Left: results with varying W^0. Dotted lines stand for $\ell = 100$, and solid lines for $\ell = 500$. Middle: values of RW, the training and test error rates at each iteration. The curve for RW actually represents the ratio RW/ℓ. The numbers of selected features are also included by rescaling to fit in the figure with the numbers beside the curve. Right: the selected 114 features obtained by MOPFS. Each feature is gray scaled according to its weight s_i ($\ell = 500,\ W^0 = 49$).

ston et al., 2000). In the VS-SSVM method, sparse linear SVMs were constructed to generate linear models based on 20 distinct partitions of training data. The final set of selected features were the aggregate of non-zero weighted features found by each of the 20 linear models. We conducted all the experiments on the MNIST database of handwritten digits, downloaded from *http://yann.lecun.com/exdb/mnist/*. The digits had been size-normalized and centered in a 28×28 image. We try to solve the classification problem of distinguishing odd numbers from even numbers. The database contains 60,000 digits. We took the first 100 and 500 digits respectively as two training sets. The 1000 digits after the 10,000th digit and the last 10,000 digits of the database were used as the validation set and the test set, respectively. For fair comparison, all methods were followed by RSVM training to construct their final classifiers. The parameter W^0 in RSVM was optimized based on the validation set for the reduced data from *each* feature selection method. Then the classifiers obtained with the best W^0 were evaluated on the test set to calculate R_{tst} shown in Table 1.

The data were preprocessed in the following way: examples were centered to have mean $\mathbf{0}$ by subtracting the mean of the training examples; then each variable (totally $28 \times 28 = 784$ variables) was scaled to have standard deviation 1; after that, each example was normalized to have ℓ_2-norm equal 1. Note that the test data should be blinded to the learning algorithm. Hence the test data were preprocessed using the mean and standard deviation of each variable computed on training data. By performing this preprocessing, the input data were transformed to the surface of the unit ball ($R^0 = 1$) centered at the origin. Then W^0+1 provided a firm estimate of h of the initial $\mathcal{F}$. This step of preprocessing also removed the variables that have all values 0, thus only 571 variables were remained.

We used a preliminary solver written in C++ available at *www.cs.rpi.edu/˜bij2/rsvm.html* to solve the dual RSVM (14) and MINOS 5.5 optimization software to solve Problem (15). Algorithm 1 can be viewed as a constraint method for solving a MOP, so the initial W^0 plays a crucial role in the trade-off of the training error versus model capacity. We examined the performance of MOPFS for a large range of choices of W^0. W^0 was chosen such that $h/\ell \in [0.05, 0.5]$. Figure 2(left) plots the training and test risks versus W^0/ℓ. The empirical risk decreases monotonically with W^0 increasing whereas the test risk curves have minimum points at about 0.1ℓ. As an example, Figure 2(middle) shows the computational behaviors of MOPFS in each iteration for $\ell = 500$ and $W^0 = 49$. We can see a decrease of the test risk as the VC dimension and the empirical risk decrease. Meanwhile, the number of features is dramatically reduced from 571 to 114. In all MOPFS experiments across various ℓ and W^0, the generalization performance was either enhanced or preserved with iteration running forth. Figure 2(right) visualizes the selected 114 features in the original image setting. The comparison as shown in Table 1 reveals that the elimination of features hardly improved or even impaired the generalization ability for the three ranking methods compared with the model constructed on all variables. MOPFS and VS-SSVM performed similarly on the larger dataset, but VS-SSVM exhibited poor prediction accuracy for $\ell = 100$ though greatly reducing features. The sparsity of support vectors could also be enforced with more features eliminated. We leave extensive comparison with other feature selection approaches on more data to future research.

Table 1. Comparison of our approach MOPFS with the filter methods and the VS-SSVM approach for training sizes equal to 100 (left) and 500 (right). The N_Feat, N_SV, R_{trn} and R_{tst} represent the numbers of selected features and support vectors, and the training and test risks. The ratio h/ℓ was computed as $(W^0 + 1)/\ell$ where W^0 was tuned based on the validation set. The FULL models without feature selection are also included

Method	$h/\ell(100)$	N_Feat	N_SV	R_{trn}	R_{tst}	$h/\ell(500)$	N_Feat	N_SV	R_{trn}	R_{tst}
Full	0.25	571	65	0.04	0.1898	0.15	571	214	0.070	0.1322
MOPFS	0.10	43	35	0.05	0.1687	0.10	114	158	0.040	0.1227
Pearson	0.16	43	42	0.06	0.1882	0.13	114	163	0.088	0.1502
Fisher	0.16	43	40	0.07	0.1890	0.10	114	168	0.074	0.1415
KStest	0.16	43	39	0.05	0.1880	0.10	114	175	0.092	0.1510
VS-SSVM	0.16	38	47	0.05	0.1910	0.13	162	152	0.048	0.1205

8. Conclusion

This work basically addresses two issues. The first issue concerns the fundamentals of constructing SVMs. A learning process needs to perform capacity control while minimizing the empirical risk in order to minimize the generalization risk. VC dimension is often used as an effective measure of the model capacity. An upper bound shows that it relates to the margin of separation and the radius of the smallest ball containing empirical data. SVMs (Section 4.1 and 4.2) usually seek the optimal decision model which produces the largest margin between the decision boundary and each of the classes. They do not explicitly regulate the radius of the ball. The MOP framework proposed herein provides us an approach to controlling the radius of the ball as well as the margin. It thus enables more rigorous implementation of the learning theory. Existing SVM formulations can be viewed as special cases of the MOP with appropriate simplifications, and thus are incorporated in this framework. The second issue is that we address the feature selection problem by developing an approach under this MOP framework. It performed better on real-world data sets of hand-written digits than some existing methods, showing that the MOP framework can be practically useful.

Open problems include the development of more efficient approximation schemes for solving the MOP. A major problem of our scheme is that it can get trapped at a local minimizer. For example, if we fix $\mathbf{s}$ in Problem (14) to obtain the $\hat{\mathbf{w}}$ and $\hat{b}$, solving Problem (15) with $\hat{\mathbf{w}}$ and $\hat{b}$ may not generate a new $\mathbf{s}$ because the initial value of $\mathbf{s}$ is likely to be optimal to Problem (15). The MOP framework may be more useful in transductive inference where the labelling of empirical data is incomplete since the information of unlabelled data can be easily incorporated into the calculation of the radius of the sphere containing data.

Acknowledgements

The author thanks Vladimir Vapnik for insightful suggestions and the referees for their useful comments.

References

Bi, J., Bennett, K. P., Embrechts, M., Breneman, C., & Song, M. (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, *3*, 1229–1243.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory* (pp. 144–152). Pittsburgh, PA: ACM Press.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273–279.

Eschenauer, H. A., Koski, J., & Osyczka, A. (1986). *Multicriteria design optimization: Procedures and applications.* New York: Springer-Verlag.

Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, *13*, 1–50.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. (2002). Learning the kernel matrix with semi-definite programming. *Proceedings of International Conference on Machine Learning.*

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

Vapnik, V. N. (1998). *Statistical learning theory.* New York: John Wiley and Sons Inc.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for SVMs. *Neural Information Processing Systems* (pp. 668–674).