Controlling Overfitting with Multi-Objective Support Vector Machines

Ingo Mierswa
Artificial Intelligence Unit
Department of Computer Science
University of Dortmund
ingo.mierswa@uni-dortmund.de

ABSTRACT

Recently, evolutionary computation has been successfully integrated into statistical learning methods. A Support Vector Machine (SVM) using evolution strategies for its optimization problem frequently deliver better results with respect to the optimization criterion and the prediction accuracy. Moreover, evolutionary computation allows for the efficient large margin optimization of a huge family of new kernel functions, namely non-positive semidefinite kernels as the Epanechnikov kernel. For these kernel functions, evolutionary SVM even outperform other learning methods like the Relevance Vector Machine. In this paper, we will discuss another major advantage of evolutionary SVM compared to traditional SVM solutions: we can explicitly optimize the inherent trade-off between training error and model complexity by embedding multi-objective optimization into the evolutionary SVM. This leads to three advantages: first, it is no longer necessary to tune the SVM parameter C which weighs both conflicting criteria. This is a very time-consuming task for traditional SVM. Second, the shape and size of the Pareto front give interesting insights about the complexity of the learning task at hand. Finally, the user can actually see the point where overfitting occurs and can easily select a solution from the Pareto front best suiting his or her needs.

Track: Genetics-Based Machine Learning and Learning Classifier Systems

Categories and Subject Descriptors: I.2.6 [Computing Methodologies]: Learning

General Terms: Algorithms, Theory, Experimentation

Keywords: Support vector machines, machine learning, kernel methods, evolution strategies

1. INTRODUCTION

Recently, several approaches were proposed where evolutionary algorithms are used to solve large margin optimization problems [14, 11]. Although the latter only performed evolutionary optimization on the less efficient primary optimization problem, both publications demonstrate the interest in this new intersection of three highly active research areas, namely machine learning, statistical learning theory, and evolutionary algorithms. This intersection allows for powerful enhancements of traditional Support Vector Machines.

Support Vector Machines (SVM) solve supervised classification tasks. A set of data points is divided into several classes and the machine learning method should learn a decision function in order to decide into which class an unseen data point should be classified. The SVM searches for an optimal decision function by maximizing the margin between data points of different classes. This not only allows an efficient optimization procedure but also the definition of an error bound for the generalization error. Furthermore, the usage of kernel functions allows the learning of non-linear decision functions. Since SVM guarantee an optimal solution for the given data set, they are currently one of the mostly used learning methods. Furthermore, many other optimization problems can also be formulated as large margin problem [22].

Usually, the optimization problem posed by SVM is solved with quadratic programming. However, there are some drawbacks with these approaches. First, no unique global optimum exists for kernel functions which are not positive semidefinite. Non-positive semidefinite kernel functions are functions which resemble a (partial) distance instead of a similarity measure. In these cases, quadratic programming is not able to find satisfying solutions at all. Moreover, most implementations do not even terminate [7]. There exist several useful non-positive kernels [13], among them the sigmoid kernel which simulates a neural network [3, 20]. Therefore, a more generic optimization scheme based on evolutionary strategies was recently proposed which allows such non-positive kernels without the need for omitting the more efficient dual optimization problem [15]. It has been shown that the evolutionary implementation leads to as good results as traditional SVM on a broad variety of real-world benchmark data sets. For non-positive semidefinite kernel functions it always outperform traditional SVM and other related learning methods as the Relevance Vector Machine.

Former applications of evolutionary algorithms to SVM include the optimization of method and kernel parameters [5, 17], the selection of optimal feature subsets [6], and the creation of new kernel functions by means of genetic programming [9]. The latter is particularly interesting since it cannot be guaranteed that the resulting kernel functions are again positive semidefinite. In contrast to these approaches, we embed evolutionary algorithms into the learning machine itself and solve the optimization problem of SVM in its dual form. By doing this, we can avoid another drawback connected to traditional SVM learning. Although the statistical learning theory takes into account both the training error and the model complexity (see Section 2), the user still has to define a weighting factor for both conflicting criteria. The search for this parameter is usually a non-trivial and very time consuming task.

In this paper, we not only propose to embed evolutionary algorithms into SVM but to embed multi-objective optimization schemes. This allows, for a first time, to explicitely optimize the inherent trade-off which is the basic idea of statistical learning theory without applying time-consuming outer wrapper approaches for optimizing the trade-off. This goal differs from the first attempts to incorporate multivariate performance measures into SVM [10] which cannot be used for competing criteria and does not solve the general trade-off between training error and capacity.

The result of the proposed approach is a Pareto front in the space of training error vs. model complexity and gives interesting insights into the nature of the problem at hand. By using a hold-out data set as a test set for the resulting models we derive a second front showing the generalization error. Both, the Pareto front and the generalization error plot allows for a quick selection of the final solution from the Pareto front without the time-consuming optimization of a weighting factor.

1.1 Outline

In Section 2 we give a short introduction into the concept of regularized risk minimization and the ideas of statistical learning theory. This allows us to formalize the optimization problem of SVM for the classification of given data points in Section 3. The constrained optimization problem developed in this section will be divided into two sub problems which will be transformed into their dual forms in Section 4. Both objectives will be used in a multi-objective evolution strategies algorithm which solves the SVM problem while the trade-off between training error and model complexity is explicitely kept in the resulting Pareto fronts (see Section 5). Finally, we give some examples of results on synthetical and real-world benchmark data sets in Section 6 before we conclude this paper in Section 7.

2. REGULARIZED RISK MINIMIZATION

We first give a short discussion about the idea of regularized risk minimization before we state the optimization problem which should be solved. Machine learning methods following this paradigm have a solid theoretical foundation and it is possible to define bounds for prediction errors.

Let the instance space be defined as the Cartesian product $X = X_1 \times \ldots \times X_m$ of attributes $X_i \subseteq \mathbb{R}$. Let Y be another set of possible labels. X and Y are random variables obeying a fixed but unknown probability distribution $P(X, Y)$. *Supervised Machine Learning* tries to find a function $f(x, \gamma)$ which predict the value of Y for a given input $x \in X$. The function class f depends on a vector of parameters γ, e. g. if f is the class of all polynomials, γ might be the degree. We define a *loss function* $L(Y, f(X, \gamma))$ in order to penalize errors during prediction [8]. Every convex function with arity 2, positive range, and $L(y, y) = 0$ can be used as loss function [19]. This leads to a possible criterion for the selection of a function f, the *expected risk*:

Definition 1. *Let X be a vector of random variables and Y another random variable obeying a fixed but unknown probability distribution $P(X, Y)$. For a given loss function $L(Y, f(X, \gamma))$ the* expected risk *is defined as*

$$R(\gamma) = \int L(y, f(x, \gamma))dP(x, y).$$

Since the underlying distribution is not known we are not able to calculate the expected risk. However, instead of estimating the probability distribution in order to allow this calculation, we directly estimate the expected risk by using a set of known data points $T = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq X \times Y$. T is usually called *training data*. Using this set of data points we can calculate the *empirical risk*:

Definition 2. *Let $T = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq X \times Y$ be an item set and let $L(Y, f(X, \gamma))$ be a loss function. The* empirical risk *is defined as*

$$R_{emp}(\gamma) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i, \gamma)).$$

If training data is sampled according to $P(X, Y)$, the empirical risk approximates the expected risk if the number of samples grows:

$$\lim_{n \to \infty} R_{emp}(\gamma) = R(\gamma).$$

It is, however, a well known problem that for a finite number of samples the minimization of $R_{emp}(\gamma)$ alone does not lead to a good prediction model [23]. For each loss function L, each candidate γ, and each set of tuples $T' \subseteq X \times Y$ with $T \cap T' = \emptyset$ exists another parameter vector γ' so that $L(y, f(x, \gamma)) = L(y, f(x, \gamma'))$ for all $x \in T$ and $L(y, f(x, \gamma)) > L(y, f(x, \gamma'))$ for all $x \in T'$. Therefore, the minimization of $R_{emp}(\gamma)$ alone does not guarantee the optimal selection of a parameter vector γ for other samples according to the distribution $P(X, Y)$. This problem is often referred to as overfitting.

At this point we use one of the main ideas of statistical learning theory. Think of two different functions perfectly approximating a given set of training points. The first function is a linear function, i. e. a simple hyperplane in the considered space $\mathbb{R}^m$. The second function also hits all training points but is strongly wriggling in between. Naturally, if we have to choose between these two approximation functions, we tend to select the simpler one, i. e. the linear hyperplane in this example. This derives from the observation that more simple functions behave better on unseen examples than unnecessary complicated functions. Since the mere minimization of the empirical risk according to the training data is not appropriate to find a good generalization, we incorporate the *capacity* [23] of the used function into the optimization problem leading to the *regularized risk*:

Definition 3. *Let Ω be strictly monotonic increasing function. The* regularized risk *is defined as*

$$R_{reg}(\gamma) = R_{emp}(\gamma) + \lambda\Omega(\gamma).$$

This risk functional is also known as *structural risk* since it takes the structural complexity into account. Ω is a function which measures the capacity of the function class f depending on the parameter vector γ (see [18] for more details). Since the empirical risk is usually a monotonically decreasing function of Ω, both criteria are conflicting and we use λ to manage the trade-off between training error and capacity.

In the next section we will introduce concrete functions for R_{emp} and Ω which leads to the concept of Support Vector Machines. This allows for the definition of a convex optimization problem which can be efficiently solved. Moreover, we can embed non-linear transformations into the optimization problem in form of kernel-functions. Instead of optimizing the weighted sum of both criteria we then explicitly optimize the trade-off between both conflicting criteria by means of multi-objective optimization.

3. SUPPORT VECTOR MACHINES

As discussed in the previous section, we need to use a class of functions whose capacity can be controlled. In this section, we will discuss a special form of regularized risk minimization, namely a large margin approach. All large margin methods have one thing in common: they embed regularized risk minimization by maximizing a margin between a linear function and the nearest data points. The most prominent large margin method for classification tasks is the *Support Vector Machine* (SVM).

We constrain the number of possible values of Y to 2, without loss of generality these values should be -1 and $+1$. In this case, finding a function f in order to decide which of both predictions is correct for an unseen data point is referred to as *classification learning* for the classes -1 and $+1$. If the data points are linearly separable, a linear hyperplane must exist separating both classes.

Definition 4. *A* separating hyperplane *is defined as*

$$H = \{x | \langle w, x\rangle + b = 0\},$$

where w is normal to the hyperplane, $|b|/||w||$ is the perpendicular distance of the hyperplane to the origin (offset or bias), and $||w||$ is the Euclidean norm of w.

The vector w and the offset b define the position and orientation of the hyperplane in the input space. These hyperplane parameters correspond to the function parameters γ discussed above. After the optimal parameters w and b were found, the prediction of new data points can be calculated as

$$f(x, w, b) = \operatorname{sgn}(\langle w, x\rangle + b),$$

which is one of the reasons why we constrained the classes.

If the classes are linearly separable at all, an infinite number of different hyperplanes exist which perfectly separate the given data points. However, one would intuitively choose the hyperplane which has the biggest amount of safety margin to both sides of the data points. It can be shown that the capacity of the class of separating hyperplanes decreases with increasing margin [18]. We can define the *margin* as the perpendicular distance of the nearest point(s) to the hyperplane. Consider two points x_1 and x_2 on opposite sides of the margin. That is $\langle w, x_1\rangle + b = +1$ and $\langle w, x_2\rangle + b = -1$ and $\langle w, (x_1 - x_2)\rangle = 2$. The margin is then given by $1/||w||$. Instead of maximizing $1/||w||$ we could also minimize $\frac{1}{2}||w||^2$ which will result into more simple equations later. Additionally, if all given data points are correctly classified by the hyperplane the products of the true class and the prediction must be positive and the following must hold:

$$\forall i : y_i\left(\langle w, x_i\rangle + b\right) \geq 0. \quad (1)$$

Normalizing w and b in a way that the point(s) closest to the hyperplane satisfy $|\langle w, x_i\rangle + b| = 1$ we can transform equation (1) into

$$\forall i : y_i\left(\langle w, x_i\rangle + b\right) \geq 1.$$

We now consider the case that the given set of data points is not linearly separable - even after using a non-linear kernel function. The optimization problem discussed in the previous section would not have a solution since in this case constraint (1) could not be fulfilled for all i. We relax this constraint by introducing positive slack variables $\xi_i, i = 1, \ldots, n$:

$$\forall i : y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i.$$

In order to minimize the number of wrong classifications we add a correction term $C\sum_{i=1}^{n}\xi_i$ to the objective function. The basic form of the primal SVM optimization problem then becomes:

$$\text{minimize } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \quad (2)$$

$$\text{subject to } \forall i : y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \quad (3)$$

$$\text{and } \forall i : \xi_i \geq 0.$$

This problem is solved by the evolutionary SVM proposed by [11]. However, we will see later in Section 5 that solving this problem in its primal form is very inefficient and has several other drawbacks.

The parameter C in equation 2 is a user defined weight for the both conflicting parts of the optimization criterion. In later parts of this paper we will discuss how multi-objective optimization can be exploited to omit this parameter and deliver the full Pareto front for all possible trade-offs between complexity and training error. Before doing this, we will transform the primal problem into a new problem which can efficiently be solved. After introducing positive Lagrange multipliers $\alpha_i, i = 1, \ldots, n$, one for each of the inequality constraints, and setting the derivatives to zero we get the Wolfe dual (see [24] for details):

Problem 1. *The SVM problem is defined as:*

$$\textit{maximize } \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k\left(x_i, x_j\right)$$

$$\textit{subject to } 0 \leq \alpha_i \leq C \textit{ for all } i = 1, \ldots, n$$

$$\textit{and } \sum_{i=1}^{n}\alpha_i y_i = 0.$$

The slacking variables ξ_i vanished and the variables α_i are constrained by the upper bound C. Please note that the examples x_i only occur in scalar products which was replaced by a kernel function $k = \langle\Phi(x_i), \Phi(x_j)\rangle$ for a (non-linear) mapping Φ in an arbitrary dot product space. This allows the search for a linear separating hyperplane in high-dimensional spaces after a non-linear transformation and, hence, the separating of non-linearly separable data. The evolutionary SVM approach proposed by [14] solves this dual problem stated above which can be performed in a more efficient way.

We can calculate the optimal normal vector w from an optimal vector α by

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i.$$

The optimal offset can be calculated with help of equation (3). Please note, that w is a linear combination of those data points x_i with $\alpha_i \neq 0$. These data points are called *support vectors*, hence the name support vector machine. Only support vectors determine the position and orientation of the separating hyperplane, other data points might as well be omitted during learning.

4. EXPLICIT TRADE-OFF BETWEEN ERROR AND COMPLEXITY

Since traditional SVM are, for example, not able to optimize for non-positive semidefinite kernel functions and approaches like Relevance Vector Machines are hardly feasible for real-world problems, it is a very appealing idea to replace the usual quadratic programming approaches by an *evolution strategies* (ES) optimization [1] or by *particle swarm optimization* (PSO) [12]. Embedding evolutionary computation into Support Vector Machines has the additional advantage of a straightforward application of multi-objective selection schemes in order to simultaneously optimize several conflicting criteria. In this work, we divide the criteria of equation 2 into two optimization targets while the weighting factor C can be omitted. This leads to the following two optimization problems:

$$\begin{aligned} &\text{minimize } \frac{1}{2}||w||^2 \\ &\text{subject to } \forall i: y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \\ &\text{and } \forall i: \xi_i \geq 0 \end{aligned} \tag{4}$$

and

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n} \xi_i \\ &\text{subject to } \forall i: y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \\ &\text{and } \forall i: \xi_i \geq 0. \end{aligned} \tag{5}$$

We will transform both objectives into their dual form in order to allow the efficient optimization of the problems including the usage of kernel functions.

4.1 First Objective: Maximizing the Margin

We introduce positive Lagrange multipliers into equation 4 but need multipliers α for the first set of inequality constraints and multipliers β for the second set of inequality constraints:

$$L_p^{(1)} = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \left(\left(y_i \langle w, x_i\rangle + b\right) + \xi_i - 1\right) - \sum_{i=1}^{n} \beta_i \xi_i$$

In order to find a solution we have to find the minimum by setting the derivatives to 0;

$$\begin{aligned} \frac{\partial L_p^{(1)}}{\partial w}(w, b, \xi, \alpha, \beta) &= w - \sum_{i=1}^{n} y_i \alpha_i x_i = 0, \\ \frac{\partial L_p^{(1)}}{\partial b}(w, b, \xi, \alpha, \beta) &= \sum_{i=1}^{n} \alpha_i y_i = 0, \\ \frac{\partial L_p^{(1)}}{\partial \xi_i}(w, b, \xi, \alpha, \beta) &= -\alpha_i - \beta_i = 0. \end{aligned}$$

Plugging the derivatives into the primal objective function $L_p^{(1)}$ delivers

$$\begin{aligned} L_p^{(1)} &= \frac{1}{2}||w||^2 - \sum_{i=1}^{n} -\alpha_i y_i \left\langle \sum_{j=1}^{n} \alpha_j y_j x_j, x_i \right\rangle + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \end{aligned}$$

The Wolfe dual must be maximized which leads to the formalization of the first objective of the multi-objective SVM setting. The resulting problem is very similar to the dual SVM problem stated above but without the upper bound C for the α_i (again, the dot product is replaced by a kernel function k):

PROBLEM 2. *The first SVM objective (maximize margin) is defined as:*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k\left(x_i, x_j\right) \\ &\textit{subject to } \alpha_i \geq 0 \textit{ for all } i = 1, \ldots, n \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

4.2 Second Objective: Minimizing the Number of Training Errors

The second problem states that the sum of errors, i.e. the sum of the slack variables ξ_i, should be minimized. This optimization must be performed under the same inequality constraints as for the first objective. We add positive Lagrange multipliers α and β:

$$L_p^{(2)} = \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(\left(y_i \langle w, x_i\rangle + b\right) + \xi_i - 1\right) - \sum_{i=1}^{n} \beta_i \xi_i$$

The derivatives must again be set to 0 which leads to slightly different conditions on the derivatives of $L_p^{(2)}$:

$$\begin{aligned} \frac{\partial L_p^{(2)}}{\partial w}(w, b, \xi, \alpha, \beta) &= -\sum_{i=1}^{n} y_i \alpha_i x_i = 0, \\ \frac{\partial L_p^{(2)}}{\partial b}(w, b, \xi, \alpha, \beta) &= \sum_{i=1}^{n} \alpha_i y_i = 0, \\ \frac{\partial L_p^{(2)}}{\partial \xi_i}(w, b, \xi, \alpha, \beta) &= 1 - \alpha_i - \beta_i = 0. \end{aligned}$$

Plugging the derivatives into the $L_p^{(2)}$ cancels out most terms because of the first two derivatives:

$$L_p^{(2)} = \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \xi_i + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \beta_i \xi_i$$

Together with the third derivative we can replace the β_i by $1 - \alpha_i$ leading to

$$\begin{aligned} L_p^{(2)} &= \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \alpha_i \xi_i + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i \end{aligned}$$

The Wolfe dual must again be maximized which leads to the second objective of the multi-objective SVM setting. Maximizing the sum of α_i corresponds to transforming each example into a support vector. In the limit, this means that the training set is merely memorized instead of generalized which is an indication of overfitting or training error minimization respectively.

PROBLEM 3. *The second SVM objective (minimize error) is defined as:*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i \\ &\textit{subject to } \alpha_i \geq 0 \textit{ for all } i = 1, \ldots, n \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

5. MOEA FOR SVM LEARNING

After all objectives and constraints for the multi-objective setting of SVM learning are defined, we will discuss some consequences of these results in this section.

5.1 Using the Dual instead of the Primal Problem

In Section 4, we defined the two conflicting criteria in their dual form (Problems 2 and 3) which should be optimized by multi-objective evolutionary algorithms (MOEA). Of course, it would also be possible to directly optimize both original criteria $1/2||w||^2$ and $\sum \xi_i$ depicted in equation 2. That is, we could directly optimize the weight vector w and the offset b as it was proposed by [11]. As mentioned before, there are two drawbacks: first, the costs of calculating the fitness function would be much higher for the original optimization problem since the fulfillment of all n constraints must be recalculated for each new hyperplane. We would have to apply the mapping Φ explicitely for each training example (which would not be possible for infinite mappings like the mapping of the RBF kernel) and we would have to calculate the dot product in $\forall i : y_i (\langle w, x_i \rangle + b) \geq 1$ from equation (3) each iteration anew. These calculations cause very high calculation costs, especially for high-dimensional data sets. Second, it would not be possible to allow non-linear learning with efficient kernel functions in the original formulation of the problem, since the mapping could only be performed on single examples x_i instead of pairs in a dot product. Finally, the kernel matrix K with $K_{ij} = k(x_i, x_j)$ can be calculated beforehand and the training data is never used during optimization again. This further reduces the needed runtime for optimization since the kernel matrix calculation is done only once and no dot product calculations are necessary during the optimization. Therefore, we decide to optimize the more efficient dual optimization problem as it was suggested by [14].

This is a nice example for a case, where transforming the objective function beforehand is both more efficient and allows enhancements which would not have been possible before. Transformations of the fitness functions became a very interesting topic recently in EC research [21].

5.2 Definition of the Objectives

The Problems 2 and 3 can be used as objectives for the MOEA. Both objectives share a common term, the sum $\sum_{i=1}^{n} \alpha_i$. Since this sum as part of the first objective is not conflicting with the second objective as a whole, we can simply omit the calculation of the sum of α_i for the first objective.

Another efficiency improvement can be achieved by formulating the problem with $b = 0$. All solution hyperplanes must then contain the origin and the equality constraints $\sum_{i=1}^{n} \alpha_i y_i = 0$ will vanish [2]. This is a mild restriction for high-dimensional spaces since the number of degrees of freedom is only decreased by one. However, during optimization we do not have to cope with this equality constraint and do not need to calculate it each generation anew.

If the equality constraint should be fulfilled (e.g. for small numbers of dimensions where omitting the constraint would make a difference), it can simply be defined as a third objective by maximizing $-\left|\sum_{i=1}^{n} \alpha_i y_i\right|$. The whole set of objectives is then given as a maximization of the terms

$$\begin{aligned} &-\sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j), \\ &\sum_{i=1}^{n} \alpha_i, \\ \text{and } &-\left| \sum_{i=1}^{n} \alpha_i y_i \right| \end{aligned}$$

subject to $\alpha_i \geq 0$ for all $i = 1, \ldots, n$.

5.3 Implementation: evoSVM

We developed a support vector machine based on evolution strategies optimization. Individuals are the real-valued vectors $\alpha = (\alpha_i, \ldots, \alpha_n)$. For mutation, we used the hybrid mutation proposed by [14] to get sparser solutions, i.e. solutions where many α_i are zero. Crossover probability is high (0.9). The individuals are initialized with 0 to further support sparsity. The maximum number of generations is 1000. The population size is 100. We use NSGA-II as the multi-objective selection scheme [4]. NSGA-II employs a selection technique which first sorts all individuals into levels of non-domination. Individuals from the first levels are added to the next generation until the desired population size is reached. Before adding individuals from the last possible level this level is sorted with respect to the crowding distance in order to preserve diversity in the population.

5.4 Selecting a Solution from the Pareto Set

The first idea of supporting the user in selecting a final solution from the Pareto front might be to just calculate the first objective in its original form and check which individual provides the highest value for

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j).$$

Data set	n	m	**Source**	σ	**Default**
Spiral	1000	2	Synthetical	1.000	50.00
Checkerboard	1000	2	Synthetical	1.000	50.00
Sonar	208	60	UCI	1.000	46.62
Diabetes	768	8	UCI	0.001	34.89
Lupus	87	3	StatLib	0.001	40.00
Crabs	200	7	StatLib	0.100	50.00

Table 1: The evaluation data sets. n is the number of data points, m is the dimension of the input space. The kernel parameter σ was optimized with a grid parameter search. The last column contains the default error, i. e. the error for always predicting the major class.

The corresponding model is the maximum margin model for the given data set without respecting the training errors since the values α_i were not bounded during the optimization. Although this solution is interesting in its own, this model is often not the desired one.

Alternatively, one could use another pointer to where in the Pareto front one should search for the final solution. We suggest to keep a small hold-out set of the data points of size k. These k data points were part of the input training set and are not used by the learner during the multi-objective optimization. After the optimization has finished and the Pareto front for all objectives is derived, the learner is applied to all k data points of the hold-out set. The prediction error for each individual is calculated with the binary loss

$$l(y, f(x)) = \begin{cases} 1 \text{ if } y \neq f(x) \\ 0 \text{ otherwise} \end{cases}$$

which leads to the error Err_p for the learned decision function f_p of the p-th individual:

$$Err_p = \sum_{q=1}^{k} l\left(y_q, f_p(x_q)\right).$$

Plotting all errors Err_p together with the training set errors results in another front which can be compared to the original Pareto front. The user should examine places where the training error and the generalization error are close together and should avoid areas where the generalization performance is much worse then the achieved objectives. The plots of both fronts together are a powerful tool to control overfitting: displaying the effect of overfitting in the generalization performance plot for all possible models ease the selection of an optimal model without getting in danger of too much overfitting.

6. EXAMPLES

The experiments in this section do not prove the ability to solve the SVM problem with evolutionary algorithms which was already done by previous work [14, 15]. It has been shown that the proposed evolutionary optimization SVM frequently outperform the quadratic programming counterparts, especially for non-positive semidefinite kernels.

In this section, we show the benefit of the transformation of the original SVM problem into an efficient multi-objective formalization by showing the Pareto fronts for several benchmark data sets. We use a RBF kernel for all SVM and determine the best parameter value for σ with a grid search parameter optimization. Possible parameters were 0.001, 0.01, 0.1, 1 and 10. A description of all data sets together with the optimal kernel parameter value σ for each data set is given in Table 1. All experiments were performed with the machine learning environment YALE [16][1], the new SVM implementation is called *evoSVM* within this framework.

Figure 1 shows all results. *The left plot* for each data set shows the resulting Pareto front delivered by the multi-objective evolutionary SVM proposed in this paper. The y-axis denotes the first optimization objective from Section 5.2 (margin size) and the x-axis shows the second objective (training error). The third objective is omitted in the plots for the sake of simplicity. *The right plot* shows the prediction errors for the training set and a hold-out test set (cf. Section 5.4). The x-Axis simply denotes a counter over all Pareto-optimal solutions found during the optimization ordered by their training errors. The y-axis denotes the prediction error for the training (+) and testing (×) data, i.e. on the hold-out set. The hold-out test set was a randomly sampled subset of size 20% of the given training set.

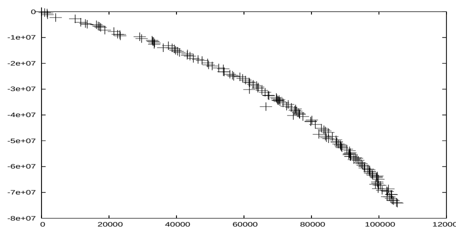
The generalization ability plotted on the right side clearly shows the location where overfitting occurs and the training error is still minimized while the test error remains or get worse. You can detect this area in the right plots at places where the training error (+) and the testing error (×) diverge. Since the x-axis in the right plots correspond to a counter of solutions in the Pareto front, ordered by its training errors which corresponds to the x-axis in the left plot, you can find the interesting solutions in the Pareto front in the same area as on the right side.

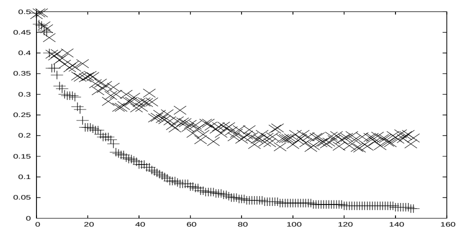
Please note that these types of plots could also be achieved for other learning schemes (e.g. usual SVM) by iteratively applying the learner for different parameter settings and produce the set of models in this way. The approach proposed in this paper has the advantage that all models are calculated in one single run which is far less time-consuming.

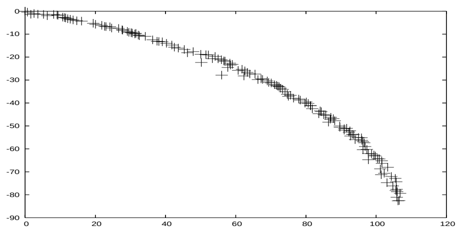
7. CONCLUSION

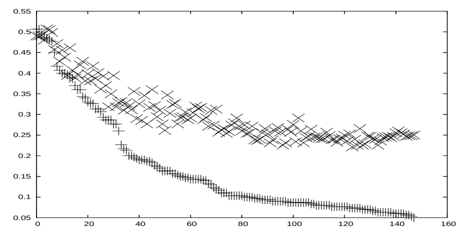
Recently, evolutionary computation was connected with statistical learning theory. The idea of large margin methods was very successful in many applications from machine learning and data mining. Embedding evolution strategies as the optimization scheme of a Support Vector Machine results in even better learning methods which frequently outperform traditional SVM. This is especially true in the case of learning with non-positive semidefinite kernel functions where traditional SVM implementations are not able to find an optimum in feasible time.

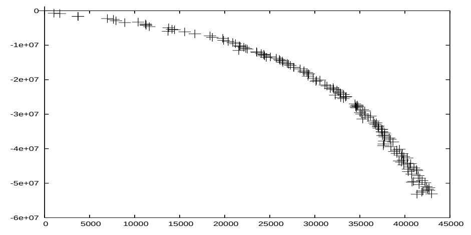
[1]`http://yale.sf.net/`

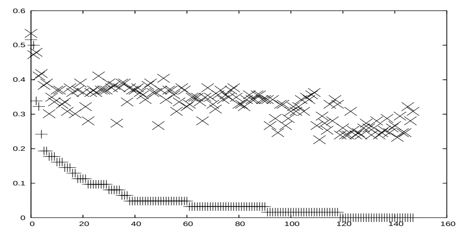

(a) Spiral Pareto

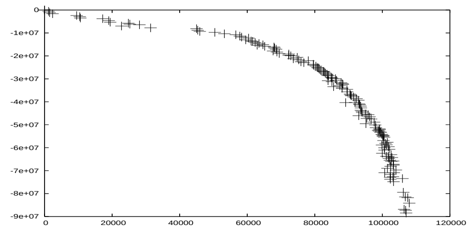

(b) Spiral Generalization

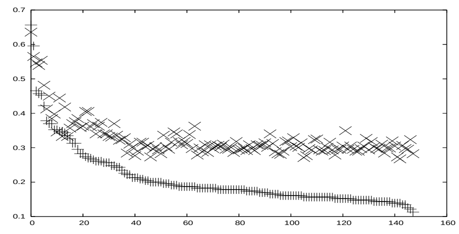

(c) Checkerboard Pareto

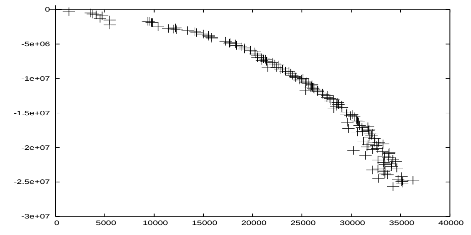

(d) Checkerboard Generalization

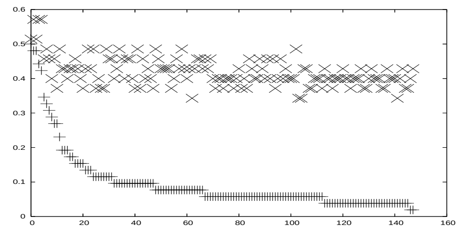

(e) Sonar Pareto

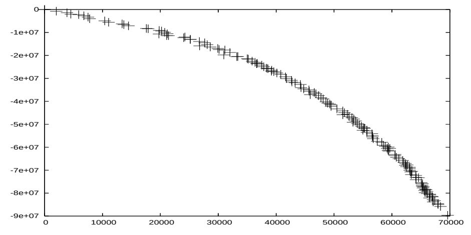

(f) Sonar Generalization


(g) Diabetes Pareto


(h) Diabetes Generalization


(i) Lupus Pareto


(j) Lupus Generalization


(k) Crabs Pareto

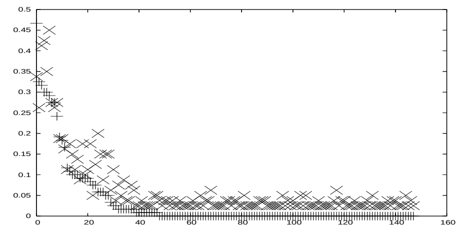

(l) Crabs Generalization

Figure 1: The left plot for each dataset shows the Pareto front delivered by the SVM proposed in this paper (x: training error, y: margin size). The right plot shows the training (+) and testing ($\times$) errors (on a hold-out set of 20%) for all individuals of the resulting Pareto fronts (x: Pareto solution counter, y: errors).

In this paper, we demonstrated how the trade-off between training error and model complexity can be made explicit. We divided the optimization problem of SVM in two parts and transformed both parts into its dual form of its own. These transformations reduce the runtime for fitness evaluation and provide space for other well-known improvements like incorporating arbitrary kernel functions for non-linear classification tasks.

We exploited the new objectives by employing a multi-objective evolutionary algorithm after some consequences of the explicit trade-off optimization were discussed. These include the possibility of further reduce runtime by using only parts of the objectives and the optional usage of a hold-out set in order to produce a hint which areas of the resulting Pareto front should be inspected by the user. This turns the Pareto front of all solutions between minimal training error and minimal model complexity into a powerful tool for controlling the overfitting of machine learning methods. Please note that all information about these plots are collected in one single run of the algorithm in contrast to wrapper approaches where the learner must be performed once for each point of such an overfitting plot.

The idea of statistical learning theory, i.e. taking the model complexity into account, is simple and appealing. Current approaches, however, did not make use of the inherent trade-off but demanded the definition of a weighting factor of the conflicting criteria from the user. The multi-objective evolutionary SVM proposed in this paper is the first solution explicitely solving the basic problem of statistical learning theory.

8. ACKNOWLEDGMENTS

This work was supported by the *Deutsche Forschungsgemeinschaft (DFG)* within the *Collaborative Research Center "Reduction of Complexity for Multivariate Data Structures"*.

9. REFERENCES

[1] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Journal Natural Computing*, 1(1):2–52, 2002.

[2] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] G. Camps-Valls, J. Martin-Guerrero, J. Rojo-Alvarez, and E. Soria-Olivas. Fuzzy sigmoid kernel for support vector classifiers. *Neurocomputing*, 62:501–506, 2004.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: Nsga-ii. Technical report, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, 2002.

[5] F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. In *Proc. of the 12th European Symposium on Artificial Neural Networks (ESANN 2004)*, pages 519–524, 2004.

[6] H. Frp̈hlich, O. Chapelle, and B. Schölkopf. Feature selection for support vector machines using genetic algorithms. *International Journal on Artificial Intelligence Tools*, 13(4):791–800, 2004.

[7] B. Haasdonk. Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, 2001.

[9] T. Howley and M. Madden. The genetic kernel support vector machine: Description and evaluation. *Artificial Intelligence Review*, 2005.

[10] T. Joachims. A support vector method for multivariate performance measures. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 377–384, 2005.

[11] S.-H. Jun and K.-W. Oh. An evolutionary statistical learning theory. *International Journal of Computational Intelligence*, 3(3):249–256, 2006.

[12] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. of the International Conference on Neural Networks*, pages 1942–1948, 1995.

[13] H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods, March 2003.

[14] I. Mierswa. Evolutionary learning with kernels: A generic solution for large margin problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)*, 2006.

[15] I. Mierswa. Making indefinite kernel learning practical. Technical report, Collaborative Research Center 475, University of Dortmund, 2006.

[16] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. YALE: Rapid prototyping for complex data mining tasks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, 2006.

[17] T. Runarsson and S. Sigurdsson. Asynchronous parallel evolutionary model selection for support vector machines. *Neural Information Processing*, 3(3):59–67, 2004.

[18] B. Schölkopf and A. J. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

[19] A. Smola, B. Schölkopf, and K.-R. Müller. General cost functions for support vector regression. In *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 79–83, 1998.

[20] A. J. Smola, Z. L. Ovari, and R. C. Williamson. Regularization with dot-product kernels. In *Proc. of the Neural Information Processing Systems (NIPS)*, pages 308–314, 2000.

[21] T. Storch. On the impact of objective function transformations on evolutionary and black-box algorithms. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 833–840, 2005.

[22] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proc. of the International Conference on Machine Learning (ICML)*, 2005.

[23] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[24] J.-P. Vert, K. Tsuda, and B. Schölkopf. *Kernel Methods in Computational Biology*, chapter A primer on kernel methods, pages 35–70. MIT Press, 2004.