Optimal Parameter Selection in Support Vector Machines[1]

K. Schittkowski[2]

Abstract

The purpose of the paper is to apply a nonlinear programming algorithm to compute kernel and related parameters of a support vector machine (SVM) by a two-level approach. Available training data are split into two groups, one set for formulating a quadratic SVM with L_2-soft margin and another one for minimizing the generalization error, where the optimal SVM variables are inserted. Subsequently, the SVM is again solved, but now for the entire set of training data, and the total generalization error is evaluated for a separate set of test data. Derivatives of the functions by which the optimization problem is defined, are evaluated in an analytical way, where an existing Cholesky decomposition needed for solving the quadratic SVM, is exploited. The approach is implemented and tested on a couple of standard data sets with up to 4,800 patterns. The results show a significant reduction of the generalization error, an increase of the margin, and a reduction of the number of support vectors in all cases where the data sets are sufficiently large. By a second set of test runs, kernel parameters are assigned to individual features. Redundant attributes are identified and suitable relative weighting factors are computed.

Keywords: machine learning; support vector machine; SVM; Gaussian kernel; kernel parameter; sequential quadratic programming; SQP; nonlinear programming;

[1]Sponsored by EU Network of Excellence PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) under contract no IST 2002-506778

[2]Department of Computer Science, University of Bayreuth, 95440 Bayreuth, Germany

1 Introduction

During the last ten years, support vector machines (SVM) became an important alternative to neural networks for machine learning. Meanwhile there is a large number of applications, see e.g.

`http://www.kernel-machines.org/`

Especially from the viewpoint of mathematical programming, SVMs are extremely interesting and touch a large variety of important topics, for example duality theory, convex optimization, large scale linear and quadratic optimization, semi-definite optimization, least squares optimization, L_1-optimization, and interior point methods. In a series of papers, Mangasarian and co-workers investigate support vector machines from the viewpoint of mathematical programming, see Bradley et al. [3], Fung and Mangasarian [9], Mangasarian and Musicant [11], and Mangasarian [12, 13].

As shown below in more detail, a support vector machine is identified by a kernel function to determine a certain similarity measure, a method compute a classification error, and an approach to handle outliers. Although being a well established technique, it is still difficult to find a suitable kernel function and especially to predetermine numerical parameter values in a specific situation. An alternative to statistical methods is to be presented in this paper based on a mathematical optimization model.

We proceed from a set of n training data (x_i, y_i), $i = 1, \ldots, n$, where $x_i \in \mathbb{R}^m$ represent certain patterns with m attributes, and $y_i \in \{-1, 1\}$ are labels. Only binary classification of the data $x_1, \ldots, x_n$ is considered, i.e., we are looking for a separating hyperplane

$$h(x) := w^T x - \gamma \tag{1}$$

with

$$\begin{array}{rll} & \min\ w^T w + \frac{1}{\nu} z^T z & \\ w \in \mathbb{R}^m, \gamma \in \mathbb{R}, z \in \mathbb{R}^n & : & Y(X^T w - \gamma e) \geq e - z\ , \\ & & z \geq 0\ , \end{array} \tag{2}$$

where $X = (x_1, \ldots, x_n)$ and where Y is a diagonal matrix containing the labels $y_1, \ldots, y_n$, and $e = (1, \ldots, 1)^T$. z is a vector of slack variables introduced for the case that the data are not completely separable, also called the soft margin. To reduce the influence of the slack variables as much as possible related to the overall goal to maximize the margin, a weight $\nu > 0$ is introduced. If $z = 0$, the data are strictly separable and the objective function is identical to $w^T w$, by which the margin is maximized, see Christianini and Shawe-Taylor [6] or Shawe-Taylor and Christianini [18].

It is easy to see that the dual program of (2) is given by

$$\begin{array}{rll} & \min\ \frac{1}{2}\, \alpha^T (Y X^T X Y + \nu I)\alpha - e^T \alpha & \\ \alpha \in \mathbb{R}^n : & y^T \alpha = 0\ , & \\ & \alpha \geq 0\ . & \end{array} \tag{3}$$

Here, I denotes the n by n unit matrix and $y := (y_1, \ldots, y_n)^T$ is a vector containing the labels. The advantage is that the slack variables of the primal formulation with a soft margin vanish completely. α denotes the dual variables and the bias γ is the multiplier of the equality constraint in (3) as will be discussed later in more detail.

The next step is to consider the matrix $X^T X$ with coefficients $x_i^T x_j$, $i, j = 1, \ldots, n$. To allow a more flexible nonlinear separation, $\mathbb{R}^n$ is mapped into the so-called feature space F by a function $\phi : \mathbb{R}^n \to F$. F is a generally unknown Hilbert space with inner product $\langle ., . \rangle$. The idea is to separate the data linearly in the feature space F, as outlined above. The only change of (3) consists of replacing the inner products $x_i^T x_j$ by their mapped inner products $\langle \phi(x_i), \phi(x_j) \rangle$, $i, j = 1, \ldots, n$. Since, however, F is unknown, so-called *kernel trick* consists of defining a function

$$k(x, \overline{x}, p) := \langle \phi(x), \phi(\overline{x}) \rangle \tag{4}$$

and to replace $x_i^T x_j$ in (3) by $k(x_i, x_j, p)$, where $k(x, \overline{x}, p)$ is a suitable kernel function depending on two given data vectors $x, \overline{x} \in \mathbb{R}^m$, by which a certain distance or similarity between x and $\overline{x}$ is to be measured. Moreover, the kernel function depends on a parameter vector $p \in \mathbb{R}^{n_p}$.

A typical and widely used kernel function is the Gaussian or RBF kernel

$$k(x, \overline{x}, p) = \exp(-p\|x - \overline{x}\|^2) \quad , \tag{5}$$

where the distance is measured in the L_2-norm. In this case, we have $n_p = 1$. Under certain assumptions on $k(x, \overline{x}, p)$, the existence of a feature mapping ϕ satisfying (4) can be shown, see Christianini and Shawe-Taylor [6].

For any two data sets $\{x_1, \ldots, x_n\}$ and $\{\overline{x}_1, \ldots, \overline{x}_{\overline{n}}\}$, we denote by X and $\overline{X}$ the corresponding matrices $X = (x_1, \ldots, x_n)$ and $\overline{X} = (\overline{x}_1, \ldots, \overline{x}_{\overline{n}})$. The kernel matrix of the two data sets is then defined by

$$K(X, \overline{X}, p) = \left(k(x_i, \overline{x}_j, p)\right)_{i=1,n,j=1,\overline{n}} \quad . \tag{6}$$

Note that $K(X, \overline{X}, p)$ is an n by $\overline{n}$ matrix depending on a parameter vector $p \in \mathbb{R}^{n_p}$. In case of $X = \overline{X}$, we also assume throughout this paper that $K(X, X, p) \in \mathbb{R}^{n \times n}$ is positive definite for all $p \in \mathbb{R}^{n_p}$.

Thus, we finally obtain the standard quadratic programming support vector machine with L_2-soft margin

$$\begin{array}{rl} & \min \ \frac{1}{2}\, \alpha^T (Y K(X, X, p) Y + \nu I)\alpha - e^T \alpha \\ \alpha \in \mathbb{R}^n : & y^T \alpha = 0 \\ & \alpha \geq 0 \quad . \end{array} \tag{7}$$

From a solution $\alpha(p, \nu) \in \mathbb{R}^n$ of (3) depending on the parameter vector p and the weight ν, we immediately get the so-called geometric margin of the separating hyperplane

$$\mu = \left(e^T \alpha(p, \nu) - \nu \alpha(p, \nu)^T \alpha(p, \nu)\right)^{-1/2} \quad . \tag{8}$$

Note that the margin depends also on the weight factor ν which must be selected in an appropriate way. It is easy to see that the bias γ of the linear hyperplane (1) in the feature space it the multiplier corresponding to the equality constraint $y^T\alpha = 0$, now denoted by $\gamma(p, \nu)$.

To evaluate the training accuracy, it is assumed that a second set of test data is given, (x_i^t, y_i^t), $i = 1, \ldots, n_t$, from where the expression

$$f(x_i^t, y_i^t, p, \nu) = y_i^t(K(x_i^t, X, p)Y\alpha(p, \nu) - \gamma(p, \nu)) \tag{9}$$

for $i = 1, \ldots, n_t$ leads to the training or generalization error

$$e(X^t, y^t, p, \nu) := \frac{1}{n_t}\left(n_t - |\{f(x_i^t, y_i^t, p, \nu) : f(x_i^t, y_i^t, p, \nu) > 0, i = 1, \ldots, n_t\}|\right) \quad . \tag{10}$$

Here, $|\{\ldots\}|$ denotes the cardinality of a set, $X^t := (x_1^t, \ldots, x_n^t)$ and $y^t := (y_1^t, \ldots, y_n^t)^T$, and we define

$$f(X^t, y^t, p, \nu) := (f(x_1^t, y_1^t, p, \nu), \ldots, f(x_{n_t}^t, y_{n_t}^t, p, \nu))^T \quad .$$

In a real life situation, it is sometimes difficult to find a suitable parameter p and a weight ν. Statistical analysis, e.g., a principal component analysis as proposed by Debnath and Takahashi [8], factorial design or similar techniques, see Cherkassy and Ma [5] or Anguita et al. [1], often lead to the necessity to solve a large number of quadratic programs of the form (3), especially if the number of parameters, n_p, becomes large. A typical situation is, for example, the assignment of one parameter of the RBF kernel to each attribute. Formal optimization approaches are found in Ayat, Cheriet, and Suen [2] and in Chapelle et al. [4]. In the latter paper, also alternative cost functions are discussed based on leave-one-out bounds.

Section 2 contains a brief outline of the underlying strategy how to formulate an optimization problem which can solved by a gradient-based nonlinear programming algorithm. For the numerical tests of this paper, we choose the sequential quadratic programming (SQP) algorithm NLPQLP20 of Schittkowski [17]. The calculation of analytical gradients of the solution of a quadratic programming subproblem is crucial for the efficiency of any gradient-based code. It is shown how they are evaluated and how an existing decomposition of a part of the kernel matrix can be exploited.

To illustrate the computational performance, some numerical test results are summarized in Section 2 based on a few small size standard data sets and the Gaussian kernel (5). Optimization variables are the kernel parameter p and the weight ν of the L_2-soft margin, see (7). A more interesting question is whether the same idea is applicable also to the case where an individual kernel parameter is assigned to each feature of our experiment. By a further set of test runs, we show some results where the conclusions are very similar and depend on the size of the training data.

2 Optimal Selection of Kernel Parameters

The idea is to split the available training data into two subsets. The first one is of size n and is denoted by a matrix X and a diagonal label matrix Y or a label vector y, respectively.

Both are used to compute the dual variables $\alpha(p,\nu)$ and the multiplier $\gamma(p,\nu)$ of the equality constraint from (7). Then there is another set of size n_t, (x_i^t, y_i^t), $i = 1, \ldots, n_t$, which is used to simulate the error of the training data.

The optimization problem consists of minimizing this error in the L_1-norm over all parameters p and all positive weights ν, i.e.,

$$p \in \mathbb{R}^{n_p}, \nu \in \mathbb{R}: \quad \begin{array}{l} \min \ \sum_{i=1}^{n_t} |f(x_i^t, y_i^t, p, \nu)^-| \\ p_l \le p \le p_u \ , \quad \nu \ge 0 \ , \end{array} \tag{11}$$

where an evaluation of the objective function requires the solution of the implicitly given quadratic program (7). The upper index '-' defines the violation of the separation condition, i.e., $a^- := \min\{0, a\}$, and p_l, p_u are suitable bounds for the kernel parameters.

However, (11) is non-differentiable and the standard trick is to introduce slack variables $z = (z_1, \ldots, z_{n_t})^T$ to get the equivalent smooth optimization problem

$$z \in \mathbb{R}^{n_t}, p \in \mathbb{R}^{n_p}, \nu \in \mathbb{R} \ : \quad \begin{array}{l} \min \ e^T z \\ f(X^t, y^t, p, \nu) + z \ge 0 \ , \\ p_l \le p \le p_u \ , \quad z \ge 0 \ , \quad \nu \ge 0 \ . \end{array} \tag{12}$$

To apply an efficient gradient-based optimization method, we have to be able to compute derivatives of the constraint functions $f(X^t, y^t, p, \nu)$ subject to the parameter p and the weight ν. Since this function implicitly depends of the solution of the quadratic programm (7), we have to analyze first the question, how to get derivatives of the solution $\alpha(p,\nu)$ and the multiplier of the equality constraint $y^T\alpha = 0$, i.e., $\gamma(p,\nu)$.

To simplify the notation, we consider first a quadratic program of the form

$$\alpha \in \mathbb{R}^n: \quad \begin{array}{l} \min \ \frac{1}{2} \ \alpha^T A(x) \ \alpha - e^T\alpha \\ y^T\alpha = 0 \\ \alpha \ge 0 \end{array} \tag{13}$$

with $y = (y_1, \ldots, y_n)^T \in \mathbb{R}^n$ and a positive definite matrix $A(x)$ depending on a scalar parameter $x \in \mathbb{R}$. It is assumed that $A(x)$ is continuously differentiable subject to x. From the corresponding Lagrangian function

$$L(\alpha, \gamma, u) = \frac{1}{2} \ \alpha^T A(x)\alpha - e^T\alpha - \gamma \ y^T\alpha - u^T\alpha$$

with a scalar multiplier γ for the equality constraint and a multiplier vector $u \in \mathbb{R}^n$ for the bounds we obtain the Karush-Kuhn-Tucker (KKT) optimality conditions

$$\begin{array}{l} A(x)\alpha - e - \gamma y - u = 0 \ , \\ y^T\alpha = 0 \ , \\ u^T\alpha = 0 \ , \\ \alpha \ge 0 \ , \\ u \ge 0 \ . \end{array} \tag{14}$$

It is furthermore assumed that the quadratic program (13) is not degenerate in the sense that (13) satisfies the constraint qualification and the strict complementary condition for all parameter values x. Both conditions are for example violated if $y = e$. Then $\alpha = 0$, $\gamma = -1$, and $u = 0$ solve (14) and we get a degenerate stationary point.

Under these assumptions, we know that the multipliers γ and u are unique and that we can perform the subsequent perturbation analysis leading to the derivatives we are looking for. We choose a sufficiently small $\epsilon > 0$ and replace $A(x)$ by $A(x_\epsilon)$ in (13) with $x_\epsilon := x + \epsilon$. The Karush-Kuhn-Tucker conditions of the perturbed problem are

$$\begin{aligned} & A(x_\epsilon)\ \alpha_\epsilon - e - \gamma_\epsilon\ y - u_\epsilon = 0 \ , \\ & y^T \alpha_\epsilon = 0 \ , \\ & u_\epsilon^T \alpha_\epsilon = 0 \ , \\ & \alpha_\epsilon \geq 0 \ , \\ & u_\epsilon \geq 0 \ . \end{aligned} \tag{15}$$

Let α, γ, and u be a solution of (14) with $\alpha_i > 0$ if and only if $i \in \mathtt{I}$, $\mathtt{I} \subset \{1, \ldots, n\}$, $\alpha = (\alpha_1, \ldots, \alpha_n)^T$. Since a solution α_ϵ, γ_ϵ, and u_ϵ of (15) depends continuously on ϵ, we can choose an ϵ sufficiently small so that also $\alpha_i^\epsilon > 0$ for all $i \in \mathtt{I}$, $\alpha_\epsilon = (\alpha_1^\epsilon, \ldots, \alpha_n^\epsilon)^T$. The strict complementary assumption guarantees that $u_j = 0$ and $u_j^\epsilon = 0$ for all $j \in \mathtt{J}$, $\mathtt{J} := \{1, \ldots, n\} - \mathtt{I}$, where $u = (u_1, \ldots, u_n)^T$ and $u_\epsilon = (u_1^\epsilon, \ldots, u_n^\epsilon)^T$. On the other hand, we also have $\alpha_j = 0$ and $u_j > 0$ for all $j \in \mathtt{J}$. By further reduction of ϵ, if necessary, we get $u_j^\epsilon > 0$ for all $j \in \mathtt{J}$ and thus also $\alpha_j^\epsilon = 0$ for all $j \in \mathtt{J}$.

We conclude that the sets of active bounds of (14) and (15) are identical, moreover that the corresponding derivatives are zero, i.e.,

$$\frac{\partial}{\partial x} u_i = 0 \text{ for all } i \in \mathtt{I} \ , \quad \frac{\partial}{\partial x} \alpha_j = 0 \text{ for all } j \in \mathtt{J} \ . \tag{16}$$

Now we suppose that we know the active set of the bound constraints for α which is identical to the active set of the perturbed problem for a sufficiently small ϵ. Let $A_{\mathtt{I}}(x)$ be the matrix $A(x)$ obtained by deleting all rows and columns not belonging to $\mathtt{I}$, and $A_{\mathtt{I}}(x_\epsilon)$ the corresponding perturbed matrix. By deleting now the same coefficients in y, we obtain $y_{\mathtt{I}}$. Then the Karush-Kuhn-Tucker conditions (14) and (15) are

$$\begin{aligned} & A_{\mathtt{I}}(x)\ \alpha_{\mathtt{I}} - e - \gamma\ y_{\mathtt{I}} = 0 \ , \\ & {y_{\mathtt{I}}}^T \alpha_{\mathtt{I}} = 0 \ , \end{aligned} \tag{17}$$

and

$$\begin{aligned} & A_{\mathtt{I}}(x_\epsilon)\ \alpha_{\epsilon\mathtt{I}} - e - \gamma_\epsilon\ y_{\mathtt{I}} = 0 \ , \\ & {y_{\mathtt{I}}}^T \alpha_{\epsilon\mathtt{I}} = 0 \ . \end{aligned} \tag{18}$$

By combining both equations, we get

$$\begin{aligned} A_{\mathtt{I}}(x_\epsilon)\,\alpha_{\epsilon\mathtt{I}} - A_{\mathtt{I}}(x)\,\alpha_{\mathtt{I}} &= (A_{\mathtt{I}}(x_\epsilon) - A_{\mathtt{I}}(x))\,\alpha_{\epsilon\mathtt{I}} + A_{\mathtt{I}}(x)(\alpha_{\epsilon\mathtt{I}} - \alpha_{\mathtt{I}}) \\ &= (\gamma_\epsilon - \gamma)\,y_{\mathtt{I}} \end{aligned} \tag{19}$$

or, after dividing by ϵ and going to the limit $\epsilon \to 0$,

$$A_{\mathtt{I}}(x)\,\frac{\partial}{\partial x}\alpha_{\mathtt{I}}(x) = \frac{\partial}{\partial x}\gamma(x)\,y_{\mathtt{I}} - \frac{\partial}{\partial x}A_{\mathtt{I}}(x)\,\alpha_{\mathtt{I}}(x)\ . \tag{20}$$

Here we introduce again the argument x for the KKT point $\alpha_{\mathtt{I}}$ and γ).

In addition, we have the identity

$$y_{\mathtt{I}}{}^T\alpha_{\mathtt{I}}(x) = y_{\mathtt{I}}{}^T\alpha_{\epsilon\mathtt{I}}(x) = 0\ , \tag{21}$$

which is valid for all solutions of the perturbed problem (15). We conclude that in the limit

$$y_{\mathtt{I}}{}^T\frac{\partial}{\partial x}\alpha_{\mathtt{I}}(x) = 0\ , \tag{22}$$

Since $A_{\mathtt{I}}(x)$ is positive definite and since therefore $\frac{\partial}{\partial x}\alpha_{\mathtt{I}}(x)$ can be eliminated from (20), we get the derivative of the multiplier $\gamma(x)$ from

$$\frac{\partial}{\partial x}\gamma(x) = \frac{1}{y_{\mathtt{I}}{}^T A_{\mathtt{I}}(x)^{-1}\,y_{\mathtt{I}}}\,y_{\mathtt{I}}{}^T A_{\mathtt{I}}(x)^{-1}\,\frac{\partial}{\partial x}A_{\mathtt{I}}(x)\,\alpha_{\mathtt{I}}(x)\ . \tag{23}$$

The same investigation can be performed for each parameter of A for which we want to compute the first partial derivatives.

Next, we consider again the support vector machine (7). We assume that the kernel matrix depends on a parameter vector $p \in \mathbb{R}^{n_p}$. By successively replacing x in $A(x)$ in (13) by p_i, $i = 1, \ldots, n_p$, and ν, we get the subsequent theorem.

Theorem 2.1 *Assume that the quadratic support vector machine (7) with L_2-soft margin is given, where the kernel function $K(X, X, p)$ depends smoothly on a parameter vector $p \in \mathbb{R}^{n_p}$. Then the partial derivatives of the solution $\alpha(p, \nu)$ and of the multiplier $\gamma(p, \nu)$ of the support*

vector machine (7) subject to the equality constraint $y^T\alpha = 0$ are computed from

$$\begin{aligned}
\frac{\partial}{\partial p_i}\alpha_{\mathcal{I}}(p,\nu) &= A_{\mathcal{I}}(p,\nu)^{-1}\left(\frac{\partial}{\partial p_i}\gamma(p,\nu)\; y_{\mathcal{I}} - \frac{\partial}{\partial p_i}Y_{\mathcal{I}}K_{\mathcal{I}}(X,X,p)Y_{\mathcal{I}}\,\alpha_{\mathcal{I}}(p,\nu)\right) \;\;, \\
\frac{\partial}{\partial p_i}\alpha_{\mathcal{J}}(p,\nu) &= 0 \;\;, \\
\frac{\partial}{\partial p_i}\gamma(p,\nu) &= \frac{1}{{y_{\mathcal{I}}}^T A_{\mathcal{I}}(p,\nu)^{-1}y_{\mathcal{I}}}\;{y_{\mathcal{I}}}^T A_{\mathcal{I}}(p,\nu)^{-1}\frac{\partial}{\partial p_i}\left(Y_{\mathcal{I}}K_{\mathcal{I}}(X,X,p)Y_{\mathcal{I}}\right)\;\alpha_{\mathcal{I}}(p,\nu) \;\;, \\
\frac{\partial}{\partial \nu}\alpha_{\mathcal{I}}(p,\nu) &= A_{\mathcal{I}}(p,\nu)^{-1}\left(\frac{\partial}{\partial \nu}\gamma(p,\nu)y_{\mathcal{I}} - \alpha_{\mathcal{I}}(p,\nu)\right) \;\;, \\
\frac{\partial}{\partial \nu}\alpha_{\mathcal{J}}(p,\nu) &= 0 \;\;, \\
\frac{\partial}{\partial \nu}\gamma(p,\nu) &= \frac{1}{{y_{\mathcal{I}}}^T A_{\mathcal{I}}(p,\nu)^{-1}y_{\mathcal{I}}}\;{y_{\mathcal{I}}}^T A_{\mathcal{I}}(p,\nu)^{-1}\,\alpha_{\mathcal{I}}(p,\nu)
\end{aligned}$$

for $i = 1,\ldots,n_p$, where $\mathcal{J}$ is the set of all active bounds of (7), $\mathcal{I} := \{1,\ldots,n\} - \mathcal{J}$ and where $A(p,\nu) := YK(X,X,p)Y + \nu I$. $A_{\mathcal{I}}(p,\nu)$, $K_{\mathcal{I}}(X,X,p)$, and $Y_{\mathcal{I}}$ denote the submatrices of $A(p,\nu)$, $K(X,X,p)$, and Y, respectively, obtained by deleting all rows and columns belonging to $\mathcal{J}$. $\alpha_{\mathcal{I}}$, $y_{\mathcal{I}}$ and $\alpha_{\mathcal{J}}$, , $y_{\mathcal{J}}$ are subvectors of α and y, respectively, containing only coefficients belonging to $\mathcal{I}$ and $\mathcal{J}$.

The previous theorem enables us to compute the derivatives of the restrictions in our relaxed optimization problem (12), see also (11), subject to p and ν. The additional partial derivatives with respect to z are easily obtained. From (9) we obtain

$$\begin{aligned}
\frac{\partial}{\partial p_i}f(x_i^t,y_i^t,p,\nu) &= y_i^t((\frac{\partial}{\partial p_i}K(x_i^t,X,p)Y\alpha(p,\nu) + K(x_i^t,X,p)Y\frac{\partial}{\partial p_i}\alpha(p,\nu)) \\
&\quad -\frac{\partial}{\partial p_i}\gamma(p,\nu)) \;\;, \\
\frac{\partial}{\partial \nu}f(x_i^t,y_i^t,p,\nu) &= y_i^t(K(x_i^t,X,p)Y\frac{\partial}{\partial \nu}\alpha(p,\nu)) - \frac{\partial}{\partial \nu}\gamma(p,\nu))
\end{aligned} \tag{24}$$

for $i = 1,\ldots,n_p$. After inserting the partial derivatives obtained from Theorem 2.1, we get the derivatives as required by the optimization algorithm.

Finally, we choose a third set of n_e data with known labels, (x_i^e, y_i^e), $i = 1, \ldots, n_e$, which is used to evaluate the error function $e(X^e, y^e, p^\star, \nu^\star)$, see (10), for an optimal parameter vector $p^\star \in \mathbb{R}^{n_p}$ with corresponding multiplier $\nu^\star$. Here we have $X^e := (x_1^e,\ldots,x_{n_e}^e)$ and $y^e := (y_1^e,\ldots,y_{n_e}^e)^T$. Since the evaluation data (X^e, y^e) must be different from the training data (X, y) and the test data (X^t, y^t) by which the error of the training data is measured, we get a performance criterion which is independent from the data by which the optimal parameter set is computed.

3 Implementation and Numerical Tests

Each step of the proposed approach requires the solution of a strictly convex quadratic program (7) with one equality constraint and lower bounds for the variables, which is solved by the primal-dual method of Goldfarb and Idnani [10] based on numerically stable orthogonal decompositions, see Powell [14]. The corresponding Fortran subroutine is called QL, see Schittkowski [16]. Since an initial Cholesky decomposition can be computed in the calling program and passed to the solver, we decompose $A(p,\nu)$ in the form

$$A(p,\nu) = U(p,\nu)^T U(p,\nu)$$

with an upper triangular matrix $U(p,\nu)$. The particular advantage is that the same Cholesky decomposition can be applied to compute the derivatives of the constraints of (12) by which the generalization error is estimated. Since, however, rows and columns of active bounds are deleted from $A(p,\nu)$, the Cholesky decomposition must be reevaluated.

The nonlinear programming problem (12) is solved by the Fortran code NLPQLP, see Schittkowski [15, 17]. In certain error cases, a non-monotone line search is applied by which the stability is significantly improved, see Dai and Schittkowski [7]. Functions and gradients must be provided by reverse communication and NLPQLP is executed with termination accuracy ACC=10^{-12} for small size problems and ACC=10^{-6} for the larger data sets. The Fortran codes are compiled by the Intel Visual Fortran Compiler, Version 8.0, under Windows XP, and executed on a Pentium IV processor with 2.8 GHz and 2 GB memory.

The intention of the numerical tests is to show the feasibility of our approach. 12 standard data sets are selected, most of them from the UCI Machine Learning Repository

`http://www.ics.uci.edu/ mlearn/MLRepository.html`

Table 1 contains the name of the data set, the number of features m, the number of training data n used to formulate and solve the SVM (7), the number of training data n_t which are used to formulate the error function in (12), and the number of independent test data n_e used to compare the classification error before and after solving (12). Proceeding from the available data files, they are chosen according to the relation 1:2:1 besides of the toy problem *checkerb*, for which data are randomly generated.

Unscaled data sets are scaled so that all data are between 0 and 1. Note that the bias γ of the linear hyperplane (1) in the feature space corresponds to the multiplier corresponding to the equality constraint $y^T\alpha = 0$, now denoted by $\gamma(p,\nu)$.

To evaluate the generalization error based on the entire set of training data, i.e., of $n+n_t$, we proceed from parameter values p and ν, and solve (7) based now on all available training data. We use the notation

$$\overline{X} := \begin{pmatrix} X \\ X^t \end{pmatrix} \quad , \quad \overline{Y} := \begin{pmatrix} Y & 0 \\ 0 & Y^t \end{pmatrix} \quad , \overline{y} := \begin{pmatrix} y \\ y^t \end{pmatrix} \tag{25}$$

and solve the extended quadratic program (7), i.e.,

$$\alpha \in \mathbb{R}^{n+n_t} : \begin{array}{l} \min \ \frac{1}{2}\, \alpha^T \, (\overline{Y} K(\overline{X},\overline{X},p)\overline{Y} + \nu I)\alpha - e^T\alpha \\ \overline{y}^T\alpha = 0 \ , \\ \alpha \geq 0 \ . \end{array} \tag{26}$$

problem	m	n	n_t	n_e
sonar	60	26	52	26
tictacto	9	239	479	240
breacanc	9	71	143	72
banana	2	250	500	250
checkerb	2	200	200	200,000
heartdis	13	67	135	68
german	24	250	500	250
breawis	9	170	341	172
adu1new	123	401	802	402
adu2new	123	566	1,132	567
adu3new	123	796	1,592	797
adu4new	123	1,195	2,390	1,196

Table 1: Data Sets for Binary Classification

Let $\overline{\alpha}(p,\nu)$ be the optimal solution and $\overline{\gamma}(p,\nu)$ be the corresponding multiplier of the equality constraint. Proceeding from an independent set of n_e test data, we compute the error function

$$f(x_i^e, y_i^e, p, \nu) = y_i^e(K(x_i^e, \overline{X}, p)\ \overline{Y}\ \overline{\alpha}(p,\nu) - \overline{\gamma}(p,\nu)) \tag{27}$$

for $i = 1, \ldots, n_e$, see also (9), and in addition

$$e(X^e, y^e, p, \nu) := \frac{1}{n_e}\left(n_e - |\{f(x_i^e, y_i^e, p, \nu) : f(x_i^e, y_i^e, p, \nu) > 0, i = 1, \ldots, n_e\}|\right) \quad , \tag{28}$$

see (10).

Now we insert the optimal parameters $(p^\star, \nu^\star)$ and also the initial values (p_0, ν_0) by which the optimization code is started, and use $e^\star := e(p^\star, \nu^\star)$ and $e^0 := e(p^0, \nu^0)$ to evaluate the generalization error before and after the optimization cycle.

The index set

$$S_v(p,\nu) := \{i : \overline{\alpha}_i(p,\nu) > 0, i = 1, \ldots, n + n_t\} \tag{29}$$

defines the support vectors of the SVM. Note that $\overline{\alpha}(p,\nu)$ denotes the multiplier vector of a corresponding primal problem in the feature space where mapped data are linearly separated, see (2) and (3). Thus, $S_v(p,\nu)$ also identifies the primal variables active for the primal problem in the feature space. Too many active primal variables can lead to overfitting, an undesired effect caused for example by a too low number of training data. It is even possible, that all available training data are perfectly separated with a maximized margin, but the resulting test function leads to an insufficient separation of any other data. Thus, we report also the percentage of support vectors before and after applying the optimization procedure, denoted by

$$s_v(p,\nu) := \frac{100}{n + n_t}|S_v(p,\nu)| \quad ,$$

for the optimal and the initial variables, $s_v^\star := s_v(p^\star, \nu^\star)$ and $s_v^0 := s_v(p^0, \nu^0)$.

An important effect of applying a support vector machine is to increase also the margin by which the data are separated, see (2). Since we consider data sets which are not completely separable, we allow violation of the separation condition and evaluate the margin by (8), which depends now also on (p, ν),

$$\mu(p,\nu) = \left(e^T \overline{\alpha}(p,\nu) - \nu \overline{\alpha}(p,\nu)^T \overline{\alpha}(p,\nu)\right)^{-1/2} \quad .$$

The displayed margin values are denoted by $\mu^\star := \mu(p^\star, \nu^\star)$ and $\mu^0 := \mu(p^0, \nu^0)$. Finally, we report also the optimal parameters $\nu^\star$ and $p^\star$ for the case $n_p = 1$, and the total number of iterations of the optimization algorithm, n_{it}.

For our numerical tests, we use the Gaussian or RBF kernel (5),

$$k(x, \overline{x}, p) = \exp(-p\|x - \overline{x}\|^2) \quad .$$

First, we apply this kernel to all features with one common parameter p and $n_p = 1$. The initial value for p is set to one, i.e., $p_0 = 1$, if data are not scaled, and to $p_0 = 1/m$ otherwise. The second optimization parameter, the weight factor ν, is initially set to $\nu = 0.001$. Results are summarized in Table 2.

No improvement of the generalization error is observed for *sonar*, the smallest data sets with only 104 available patterns. Some other small data sets, *breacanc*, *banana*, *heartdis*, and *german*, seem to be overfitted. In all other cases, we observe a significant reduction of the generalization error and the number of support vectors. Moreover, the margin is increased in most cases. The number of iterations roughly corresponds to the number of SVMs to be solved until an optimal parameter $p^\star$ and and an optimal weight $\nu^\star$ are reached. The large number if iterations in some cases, e.g., *banana*, *heartdis*, *adu2new*, *adu3new*, and *adu4new*, indicates that the optimization process is not very stable. Possible reasons are numerical instabilities during the solution of (7), too many local minima, very *flat* surfaces of the objective function, or non-unique solutions. Even a violation of the assumptions under which the differentiability of a KKT point of the SVM (7) is shown, is possible.

Now we apply one individual kernel parameter to each feature to have the possibility to weight the attributes and to select redundant ones. If $x = (x_1, \ldots, x_m)^T$ and $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_m)^T$ are two feature vectors and $p = (p_1, \ldots, p_m)^T$ a vector of m kernel parameters, we now define an extended kernel function by

$$k(x, \overline{x}, p) = exp\left(-\sum_{i=1}^{m} p_i (x_i - \overline{x}_i)^2\right) \quad , \tag{30}$$

see also (5). Instead of one kernel parameter, we list the mean value $\overline{p}^\star$ and in addition the number of parameters which approach the lower bound zero, n_r, see Table 3. This figure is an indication whether there are redundant features or not. In this case, the number of optimization variables of the nonlinear programming problem (12) increases by the number of features m minus one.

In general, the number of iterations is much higher compared to the one-parameter situation. Similar to the results of Table 2, we do not observe significant improvement of

problem	$e^\star$	e^0	$s_v^\star$	s_v^0	$\mu^\star$	μ^0	$\nu^\star$	$p^\star$	n_{it}
sonar	23.08	23.08	100	91	0.1845	0.1037	0.55	$0.13 \cdot 10^{+1}$	21
tictacto	0.83	14.58	53	99	0.0056	0.0437	$0.14 \cdot 10^{-2}$	$0.13 \cdot 10^{-1}$	101
breacanc	20.83	36.11	100	78	0.7483	0.0094	$0.68 \cdot 10^{+1}$	0.94	78
banana	12.00	12.80	94	35	0.3380	0.0152	$0.11 \cdot 10^{+2}$	$0.17 \cdot 10^{+1}$	318
checkerb	5.95	6.72	8	7	0.0030	0.0014	0.00	$0.15 \cdot 10^{+2}$	57
heartdis	5.88	13.24	97	51	0.4102	0.0086	$0.31 \cdot 10^{+1}$	0.13	134
german	27.20	35.20	91	100	0.0767	0.5496	0.13	$0.21 \cdot 10^{+1}$	50
breawis	4.07	4.65	39	19	0.0011	0.0217	$0.20 \cdot 10^{-4}$	$0.11 \cdot 10^{-3}$	66
adu1new	17.91	25.62	77	100	0.0805	0.0347	0.34	$0.19 \cdot 10^{-2}$	41
adu2new	18.34	25.04	77	100	0.2130	0.0294	$0.35 \cdot 10^{+1}$	$0.21 \cdot 10^{-1}$	122
adu3new	15.93	24.87	76	99	0.2054	0.0256	$0.39 \cdot 10^{+1}$	$0.16 \cdot 10^{-1}$	137
adu4new	18.73	25.77	73	99	0.1762	0.0213	$0.41 \cdot 10^{+1}$	$0.13 \cdot 10^{-1}$	209

Table 2: Performance Results for One Kernel Parameter

the performance index for the small sized data set *sonar*. The generalization error $e^\star$ and the margin $\mu^\star$ are not improved significantly, but the average number of support vectors is reduced in most cases. The results for the four data sets *adu1new* to *adu4new* are somewhat different from the others. There are nearly no improvements of our performance criteria, but we observe at least a much larger relative number of redundant attributes. Again, the slow convergence is an indication of numerical instabilities either in the numerical algorithm, the calculation of gradients, or the data.

It is interesting to see that the number of iterations, n_{it}, is not related to the dimension of the optimization problem (12) which consists of up to $n_t + m + 1 = 2,514$ variables and $n_t = 1,195$ constraints for *adu4new*. In this case, a quadratic programming subproblem of size $n = 1,195$ must be solved in each iteration for evaluating the constraint function values and their derivatives. Moreover, the applied sequential quadratic programming algorithm requires the internal solution of another quadratic program of the same size as the nonlinear program to compute a search direction. Both are solved by the code QL of Schittkowski [16], see also the initial comments of this section. To call the SQP code NLPQLP, we need internal working space of more than $2.5(n_t + m + 1)^2 + (n_t + m + 1)n_t \approx 2 \cdot 10^7$ double precision real variables. The quadratic program 26) by which the generalization error is computed, has up to $n + n_t = 3,585$ variables, and is again solved by the dense algorithm QL.

4 Conclusions

We present an approach to predetermine optimal kernel and weighting parameters of a support vector machine. A nonlinear programming algorithm is applied to an optimization problem, by which the generalization error of one part of the training data is minimized over solutions of a quadratic SVM subject to another subset of the training data. We consider binary classification subject to the classical SVM with soft margin and penalization in the

problem	$e^\star$	e^0	$s_v^\star$	s_v^0	$\mu^\star$	μ^0	$\nu^\star$	$\overline{p}^\star$	n_r	n_{it}
sonar	23.08	23.08	100	91	0.2165	0.1037	0.78	$0.10 \cdot 10^{+1}$	2	90
tictacto	0.83	14.58	34	99	0.0002	0.0437	$0.18 \cdot 10^{-5}$	$0.28 \cdot 10^{-2}$	0	402
breacanc	18.06	36.11	94	78	0.3562	0.0094	$0.15 \cdot 10^{+1}$	$0.86 \cdot 10^{+1}$	1	324
banana	11.20	12.80	39	35	0.0275	0.0152	$0.31 \cdot 10^{-2}$	0.69	0	85
checkerb	6.14	6.72	9	7	0.0027	0.0014	0.00	$0.14 \cdot 10^{+2}$	0	52
heartdis	10.29	13.24	80	51	0.0277	0.0086	$0.74 \cdot 10^{-2}$	$0.90 \cdot 10^{-1}$	6	284
german	26.80	35.20	90	100	0.0046	0.5496	$0.27 \cdot 10^{-3}$	$0.95 \cdot 10^{-1}$	4	211
breawis	3.49	4.65	53	19	0.0453	0.0217	$0.66 \cdot 10^{-1}$	0.18	2	66
adu1new	19.90	25.62	77	100	0.1219	0.0347	$0.13 \cdot 10^{+1}$	0.58	40	410
adu2new	19.93	25.04	77	100	0.1827	0.0294	$0.31 \cdot 10^{+1}$	0.44	36	500
adu3new	17.44	24.87	78	99	0.2186	0.0256	$0.61 \cdot 10^{+1}$	0.42	59	387
adu4new	18.81	25.77	75	99	0.1658	0.0213	$0.57 \cdot 10^{+1}$	0.44	33	288

Table 3: Performance Results for m Kernel Parameters

L_2-norm, but the idea is easily transferred to related models, e.g., penalization in the L_1-norm. Also the Gaussian kernel used throughout the paper, can be replaced by any other kernel function or even a combination of different kernels.

Since the test data by which the overall efficiency of the SVM is measured, are independent from the training data, we are able to evaluate the accuracy before and after optimization. If the number of training data is not to small, a significant reduction of the generalization error is observed. Also the number of support vectors is decreased and the margin is increased.

However, the achievements depend on the choice of the initial parameters by which the SQP algorithm is started, and must be interpreted very carefully. They represent the improvement when little or no information is available about a suitable choice of kernel parameters. Another difficulty is that the applied gradient-based optimization algorithm is only able to approximate a local solution, and we do no know whether better local minima exist. Since, however, a significant error reduction is obtained at least if the number of training data is not too small, and since the number of solutions of quadratic SVMs is reasonably small, the optimization approach seems to have some advantages over statistical methods, especially in case of many kernel parameters.

Especially in case of many kernel parameters, obtained for example by assigning individual weights to attributes, we observe slow convergence. Further investigations are necessary to understand this situation in more detail and to find out where the numerical instability comes from.

Each evaluation of the constraints of our optimization problem requires the full solution of a quadratic programming (QP) subproblem. Although the Cholesky decomposition can be exploited for computing analytical derivatives efficiently, more research efforts are required to handle very large training sets either by replacing the dense QP solver by an iterative method for large scale quadratic programs or by applying a large scale optimization routine directly to the full problem, where also the dual variables of the SVM are treated as optimization variables and where the optimality conditions of the QP lead to nonlinear constraints.

5 Acknowledgements

The author acknowledges the financial support provided by the EU Network of Excellence PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning).

References

[1] Anguita D., Boni A., Ridella S., Rivieccio F., Sterpi D. (2005): *Theoretical and practical model selection methods for support vector classifiers*, in: Support Vector Machines: Theory and Applications, L. Wang ed.

[2] Ayat N.E., Cheriet M., Suen C.Y. (2002): *Empirical error based optimization of SVM kernels: Application to digit image recognition*, Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2002, Niagara, Canada, IEEE Computer Society

[3] Bradley P.S., Fayyad U.M., Mangasarian O.L. (1999): *Mathematical programming for data mining: Formulations and challenges*, INFORMS Journal on Computing, Vol. 11, 217-238

[4] Chapelle O., Vapnik V., Bousquet O., Mukhherjee S. (2002): *Choosing multiple parameters for support vector machines*, Machine Learning, Vol. 46, No. 1, 131-159

[5] Cherkassy V., Ma Y. (2004): *Practical selection of SVM parameters and noise estimation for SVM regression*, Neural Networks, Vol. 17, No. 1, 113-26

[6] Christianini N., Shawe-Taylor J. (2000): *An Introduction to Support Vector Machines*, Cambridge University Press

[7] Dai Y.H., Schittkowski K. (2005):*A sequential quadratic programming algorithm with non-monotone line search*, submitted for publication

[8] Debnath R., Takahashi H. (2004): *Analyzing the bahavior of distribution of data in the feature space of SVM with Gaussian kernel,* Neural Information Processing - Letters and Reviews, Vol. 5, No. 3, 41-48

[9] Fung G, Mangasarian O.L. (2004): *A feature selection Newton method for support vector machine classification*, Computational Optimization and Applications, Vol. 28, 185-202

[10] Goldfarb D., Idnani A. (1983): *A numerically stable method for solving strictly convex quadratic programs*, Mathematical Programming, Vol. 27, 1-33

[11] Mangasarian O.L., Musicant D.R. (2001): *Lagrangian support vector machines*, Journal of Machine Learning Research, Vol. 1, 161-177

[12] Mangasarian O.L. (2002): *A finite Newton method for classification*, Optimization Methods and Software, Vol. 17, 913-929

[13] Mangasarian O.L. (2003): *Data mining via support vector machines*, in: System Modeling and Optimization XX, E.W. Sachs, R. Tichatschke eds., Kluwer Academic Publishers, Boston 2003, 91-112

[14] Powell M.J.D. (1983): *On the quadratic programming algorithm of Goldfarb and Idnani*. Report DAMTP 1983/Na 19, University of Cambridge, Cambridge

[15] Schittkowski K. (1985/86): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, 485-500

[16] Schittkowski K. (2003): *QL: A Fortran code for convex quadratic programming-User's guide*, Report, Department of Mathematics, University of Bayreuth

[17] Schittkowski K. (2004): *NLPQLP20: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search-User's guide*, Report, Department of Mathematics, University of Bayreuth

[18] Shawe-Taylor J., Christianini N. (2004): *Kernel Methods for Pattern Anaysis*, Cambridge University Press