An SMO algorithm for the Potential Support Vector Machine

Tilman Knebel
Department of Electrical Engineering and Computer Science
Technische Universität Berlin
Franklinstr. 28/29, 10587 Berlin, Germany
tk@cs.tu-berlin.de

Sepp Hochreiter
Institute of Bioinformatics
Johannes Kepler University Linz
Altenbergerstr. 69, 4040 Linz, Austria
hochreit@bioinf.jku.at

Klaus Obermayer
Department of Electrical Engineering and Computer Science
Technische Universität Berlin
Franklinstr. 28/29, 10587 Berlin, Germany
oby@cs.tu-berlin.de

Abstract

We describe a fast Sequential Minimal Optimization (SMO) procedure for solving the dual optimization problem of the recently proposed Potential Support Vector Machine (P-SVM). The new SMO consists of a sequence of iteration steps, in which the Lagrangian is optimized either with respect to one (single SMO) or to two (dual SMO) of the Lagrange multipliers while keeping the other variables fixed. An efficient selection procedure for Lagrange multipliers is given and two heuristics for improving the SMO procedure are described: block optimization and annealing of the regularization parameter ϵ. A comparison between the different variants show, that the dual SMO including block optimization and annealing performs most efficient in terms of computation time. In contrast to standard Support Vector Machines (SVMs), the P-SVM is applicable to arbitrary dyadic datasets, but benchmarks are provided against libSVM's ϵ-SVR and C-SVC implementations for problems which are also solvable by standard SVM methods. For those problems computation time of the P-SVM is comparable or somewhat higher than the standard SVM. The number of support vectors found by the P-SVM is usually much smaller for the same generalization performance.

1 Introduction

Learning from examples in order to predict is one of the standard tasks in machine learning. Many techniques have been developed to solve classification and regression problems, but by far most of them were specifically designed for vectorial data. However, for many datasets a vector-based description is inconvenient or may even be wrong and other representations like dyadic data (Hofmann and Puzicha (1998); Hoff (2005)) which are based on relationships between objects (Fig. 1, left), are more appropriate.

Support Vector Machines (SVMs, Schölkopf and Smola (2002); Vapnik (1998)) are a successful class of algorithms for solving supervised learning tasks. Although SVMs have been originally developed for vectorial data, the actual predictor and the learning procedure make use of a relational representation. Given a proper similarity measure between objects, SVMs learn to predict attributes based on a pairwise "similarity" matrix, which is usually called the Gram matrix (cf. Fig. 1, right).

Standard Support Vector Machines, however, underlie some technical as well as conceptual restrictions (for a more detailled discussion see Hochreiter and Obermayer (2006a)). Firstly, SVMs operate on pairwise data and cannot be extended in a straightforward way to general dyadic representations (cf. Fig. 1). The similarity measure (kernel) has to be positive definite[1] and the general case of dyadic data, where the relationships between two different sets of objects are quantified, cannot be handled directly[2]. True dyadic data, however, occur in several application areas including bioinformatics (genes vs. probes in DNA microarrays, molecules vs. TAE descriptors for predicting certain properties of proteins) or information retrieval (word frequencies vs. documents, mutual hyperlinks between web-pages). Hence learning algorithms which are able to operate on this data are of high value.

Secondly, standard SVM solutions also face a couple of technical disadvantages. The solution of an SVM learning problem for example is scale sensitive, because the final predictor depends upon how the training data had been scaled (see, e.g. Figs. 2 and 3 of Hochreiter and Obermayer (2006a)). This problem of scale sensitivity is avoided by the P-SVM approach through a modified cost function, which softly enforces a whitening of the data in feature space. A second disadvantage of standard SVMs relates to the fact, that all margin errors translate into support vectors. The number of support vectors can, therefore, be larger than necessary, for example, if many training data points are from regions in data space where classes of a classification problem overlap. Finally, the P-SVM method leads to an expansion of the predictor into a sparse set of the descriptive "row" objects (cf. Fig. 1, right) rather than training data points as in standard SVM approaches. It can therefore be used as a wrapper method for feature selection applications and previous results on quite challenging datasets had been promising (cf. Hochreiter and Obermayer (2004, 2006b)).

[1]At least the final Gram matrix of the training examples should be.

[2]Some workarounds exits. The relational dataset can, e.g., be treated like a vectorial representation as in the feature map method (Schölkopf and Smola (2002)).

	a	b	c	d	e	f
α	0	2	0	-1	-7	-8
β	1	7	2	0	0	-2
χ	8	-9	-1	0	1	2

	a	b	c	d	e
a	1	-0.1	0.2	0.9	-0.5
b	-0.1	1	0.2	0.1	0.3
c	0.2	0.2	1	-0.2	-0.2
d	0.9	0.1	-0.2	1	0.5
e	-0.5	0.3	-0.2	0.5	1

Figure 1: **Left:** Cartoon of a dyadic data set. "Column" objects $\{a, b, \dots\}$, whose attributes should be predicted, are quantitatively described by a matrix of numerical values representing their mutual relationships with the descriptive "row" objects $\{\alpha, \beta, \dots\}$. **Right:** Pairwise data: Special case of dyadic data, where row and column objects coincide.

The practical application of SVMs, however, requires an efficient implementation of the dual optimization problem (Hochreiter and Obermayer (2006a)). In the following we describe an implementation, which is based on the idea of Sequential Minimal Optimization (SMO, Platt (1999); Keerthi et al. (2001, 2003); Lai (2003)) and we will compare several variants of the P-SVM SMO with respect to computation time for a number of standard benchmark datasets. The results demonstrate, that an efficient implementation of the P-SVM exists which makes its application to general, real world dyadic datasets feasible. This SMO implementation of the P-SVM is available via the web-site: `http://ni.cs.tu-berlin.de/software/psvm`. The optimal SMO version is compared with libSVM's ϵ-SVR and C-SVC for learning problems which are also solvable by standard SVM approaches. For those problems computation time of the P-SVM is comparable or somewhat higher than the standard SVM. The number of support vectors found by the P-SVM is usually much smaller than the number of support vectors found by the corresponding ϵ-SVR and C-SVC for the same generalization performance, which gives the P-SVM method an advantage especially for large datasets.

2 The P-SVM Optimization Problem

Let $\boldsymbol{X}_\phi$ be the matrix of L data vectors from a feature space ϕ, $\boldsymbol{w}$ the normal vector of a separating hyperplane, y the L attributes (binary in case of classification, or real valued in case of regression) of the data vectors, and $\boldsymbol{K}$ the $L \times P$ kernel matrix of scalar products between the L vectors in feature space and the P complex features provided by the measurement process. Then the P-SVM "primal" optimization problem has the form (see Hochreiter and Obermayer

(2006a)):

$$\begin{aligned}
\min_{\boldsymbol{w},\boldsymbol{\xi}^+,\boldsymbol{\xi}^-} \quad & \frac{1}{2}\,\|\boldsymbol{X}_\phi^\top\,\boldsymbol{w}\|^2 \;+\; C\mathbf{1}^\top\left(\boldsymbol{\xi}^+ \;+\; \boldsymbol{\xi}^-\right) & (1)\\
\text{s.t.} \quad & \boldsymbol{K}^\top\;\left(\boldsymbol{X}_\phi^\top\;\boldsymbol{w}\;-\;\boldsymbol{y}\right)\;+\;\boldsymbol{\xi}^+\;+\;\epsilon\,\mathbf{1}\;\geq\;\mathbf{0}\\
& \boldsymbol{K}^\top\;\left(\boldsymbol{X}_\phi^\top\;\boldsymbol{w}\;-\;\boldsymbol{y}\right)\;-\;\boldsymbol{\xi}^-\;-\;\epsilon\,\mathbf{1}\;\leq\;\mathbf{0}\\
& \mathbf{0}\;\leq\;\boldsymbol{\xi}^+,\boldsymbol{\xi}^-\quad ,
\end{aligned}$$

where K is normalized column-wise to zero mean and variance L^{-1}. The parameters C and ϵ correspond to the two different regularization schemes, which have been suggested for the P-SVM method, where ϵ-regularization has been proven more useful for feature selection and C-regularization for classification or regression problems (Hochreiter and Obermayer (2006a)). $\boldsymbol{\xi}^+$ and $\boldsymbol{\xi}^-$ are the vectors of the slack variables describing violations of the constraints.
In order to derive the dual version, we consider the Lagrangian

$$\begin{aligned}
& L \;=\; \frac{1}{2}\,\boldsymbol{w}^\top\;\boldsymbol{X}_\phi\;\boldsymbol{X}_\phi^\top\;\boldsymbol{w}\;+\;\;C\mathbf{1}^\top\left(\boldsymbol{\xi}^+\;+\;\boldsymbol{\xi}^-\right) & (2)\\
& -\;\left(\boldsymbol{\alpha}^+\right)^\top\;\left(\boldsymbol{K}^\top\left(\boldsymbol{X}_\phi^\top\;w\;-\;\boldsymbol{y}\right)\;+\;\boldsymbol{\xi}^+\;+\;\epsilon\right)\\
& +\;\left(\boldsymbol{\alpha}^-\right)^\top\;\left(\boldsymbol{K}^\top\left(\boldsymbol{X}_\phi^\top\;\boldsymbol{w}\;-\;\boldsymbol{y}\right)\;-\;\boldsymbol{\xi}^-\;-\;\epsilon\right)\\
& -\;\left(\boldsymbol{\mu}^+\right)^\top\;\boldsymbol{\xi}^+\;-\;\left(\boldsymbol{\mu}^-\right)^\top\;\boldsymbol{\xi}^-\quad ,
\end{aligned}$$

where the $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$ are the Lagrange multipliers for the residual error constraints and the $\boldsymbol{\mu}^+$ and $\boldsymbol{\mu}^-$ are the Lagrange multipliers for the slack variable constraints ($\boldsymbol{\xi}^+,\boldsymbol{\xi}^-\;\geq\;0$). Setting the derivative of L with respect to $\boldsymbol{w}$ equal to zero leads to

$$\boldsymbol{X}\;\boldsymbol{X}_\phi^\top\;\boldsymbol{w}\;=\;\boldsymbol{X}\;\boldsymbol{K}\;(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-) \tag{3}$$

which is always fulfilled, if

$$\boldsymbol{X}_\phi^\top\;\boldsymbol{w}\;=\;\boldsymbol{K}\;(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-)\;\;. \tag{4}$$

If $\boldsymbol{X}\boldsymbol{X}_\phi^\top$ does not have full rank, all solutions $\boldsymbol{w}$ of eq. (3) lie within a subspace which depends on $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$.
Setting the derivative of L with respect to $\boldsymbol{\xi}^+,\boldsymbol{\xi}^-$ equal to zero leads to

$$\begin{aligned}
& C\mathbf{1}\;-\;\boldsymbol{\alpha}^+\;-\;\boldsymbol{\mu}^+\;=\;\mathbf{0}\quad\text{and} & (5)\\
& C\mathbf{1}\;-\;\boldsymbol{\alpha}^-\;-\;\boldsymbol{\mu}^-\;=\;\mathbf{0}\,. & (6)
\end{aligned}$$

Inserting eq. (4) into eq. (2) leads to the dual optimization problem

$$\begin{aligned}
\min_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-} \quad & \frac{1}{2}\left(\boldsymbol{\alpha}^+\;-\;\boldsymbol{\alpha}^-\right)^\top\;\boldsymbol{K}^\top\;\boldsymbol{K}\;\left(\boldsymbol{\alpha}^+\;-\;\boldsymbol{\alpha}^-\right) & (7)\\
& -\,\boldsymbol{y}^\top\;\boldsymbol{K}\;\left(\boldsymbol{\alpha}^+\;-\;\boldsymbol{\alpha}^-\right)\;+\;\epsilon\;\mathbf{1}^\top\left(\boldsymbol{\alpha}^+\;+\;\boldsymbol{\alpha}^-\right)\\
\text{s.t.} \quad & \mathbf{0}\;\leq\;\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-\;\leq\;C\mathbf{1}\;\;.
\end{aligned}$$

The final P-SVM predictor is given by

$$\boldsymbol{y}_{pred} \;=\; f(\boldsymbol{K}\,\boldsymbol{\alpha} \;+\; b)\;\;, \tag{8}$$

where $f(\boldsymbol{h}) = \text{sign}(\boldsymbol{h})$ for classification and $f(\boldsymbol{h}) = \boldsymbol{h}$ for regression. The thresehold variable b is determined by the mean of the label vector $\boldsymbol{y}$ (cf. eq. 18 of Hochreiter and Obermayer (2006a)).
The Karush-Kuhn-Tucker (KKT) conditions state that for the optimal solution the product between dual variables and the l.h.s. of the primal constraints in eq. (1) is zero. Using

$$\begin{aligned} \boldsymbol{Q} \;&:=\; \boldsymbol{K}^{\top}\boldsymbol{K}\;, \\ \boldsymbol{F} \;&:=\; \boldsymbol{Q}\left(\boldsymbol{\alpha}^{+} - \boldsymbol{\alpha}^{-}\right) - \boldsymbol{K}^{\top}\boldsymbol{y} \end{aligned} \tag{9}$$

we obtain

$$F_j \;+\; \epsilon \;+\; \xi_j^{+} \;\;\geq\;\; 0 \tag{10}$$

$$F_j \;-\; \epsilon \;-\; \xi_j^{-} \;\;\leq\;\; 0 \tag{11}$$

for the primal constraints in eq. (1) and

$$\alpha_j^{+}\;\left(F_j \;+\; \epsilon \;+\; \xi_j^{+}\right) \;=\; 0 \tag{12}$$

$$\alpha_j^{-}\;\left(F_j \;-\; \epsilon \;-\; \xi_j^{-}\right) \;=\; 0 \tag{13}$$

$$\mu_j^{+}\;\xi_j^{+} \;=\; 0 \tag{14}$$

$$\mu_j^{-}\;\xi_j^{-} \;=\; 0 \tag{15}$$

for the KKT condition. Following the procedure described in Keerthi et al. (2001) we obtain

$$\begin{aligned} F_j + \epsilon \;<\; 0 \;\; &\overset{(10)}{\Longrightarrow} \;\; \xi_j^{+} \;>\; 0 \;\; \overset{(14)}{\Longrightarrow} \;\; \mu_j^{+} \;=\; 0 \;\; \overset{(5)}{\Longleftrightarrow} \;\; \alpha_j^{+} \;=\; C\;, \\ F_j - \epsilon \;>\; 0 \;\; &\overset{(11)}{\Longrightarrow} \;\; \xi_j^{-} \;>\; 0 \;\; \overset{(15)}{\Longrightarrow} \;\; \mu_j^{-} \;=\; 0 \;\; \overset{(6)}{\Longleftrightarrow} \;\; \alpha_j^{-} \;=\; C\;, \\ F_j + \epsilon \;>\; 0 \;\; &\xrightarrow{(12),(1)} \;\; \alpha_j^{+} \;=\; 0\;, \text{ and} \\ F_j - \epsilon \;<\; 0 \;\; &\xrightarrow{(13),(1)} \;\; \alpha_j^{-} \;=\; 0\;, \end{aligned}$$

where the equation numbers above the arrows denote the equations which have been used. Hence KKT conditions are met if

$$\begin{aligned} \alpha_j^{+} \;=\; 0 \;\; &\Longrightarrow \;\; F_j \;+\; \epsilon \;\geq\; 0\;, \\ 0 \;<\; \alpha_j^{+} < C \;\; &\Longrightarrow \;\; F_j \;+\; \epsilon \;=\; 0\;, \\ \alpha_j^{+} \;=\; C \;\; &\Longrightarrow \;\; F_j \;+\; \epsilon \;\leq\; 0\;, \end{aligned} \tag{16}$$

and

$$\begin{aligned} \alpha_j^{-} \;=\; 0 \;\; &\Longrightarrow \;\; F_j \;-\; \epsilon \;\leq\; 0\;, \\ 0 \;<\; \alpha_j^{-} < C \;\; &\Longrightarrow \;\; F_j \;-\epsilon \;=\; 0\;, \\ \alpha_j^{-} \;=\; C \;\; &\Longrightarrow \;\; F_j \;-\epsilon \;\geq\; 0. \end{aligned} \tag{17}$$

Because α_j^+ and α_j^- are never both larger than zero [3] ,

$$\begin{array}{lllllll} \alpha_j^+ & > 0 & \xrightarrow{(12),(1)} & F_j + \epsilon \ \leq \ 0 & \Longrightarrow & F_j - \epsilon \ < \ 0 & \xrightarrow{(13),(1)} \ \alpha_j^- \ = \ 0, \\ \alpha_j^- & > 0 & \xrightarrow{(13),(1)} & F_j - \epsilon \ \geq \ 0 & \Longrightarrow & F_j + \epsilon \ > \ 0 & \xrightarrow{(12),(1)} \ \alpha_j^+ \ = \ 0, \end{array}$$

we can use

$$\boldsymbol{\alpha} \ = \ \boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^- \tag{18}$$

in order to rewrite eq. (7) in compact form:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}} & \frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{Q} \, \boldsymbol{\alpha} \ - \ \boldsymbol{y}^\top \ \boldsymbol{K} \boldsymbol{\alpha} \ + \ \epsilon \, \|\boldsymbol{\alpha}\|_1 \\ \text{s.t.} & -C\mathbf{1} \ \leq \alpha_j \ \leq \ C\mathbf{1} \ . \end{array} \tag{19}$$

The term $\epsilon \, \|\boldsymbol{\alpha}\|_1 = \epsilon \mathbf{1}^\top \left(\boldsymbol{\alpha}^+ + \boldsymbol{\alpha}^-\right)$ enforces an expansion of $\boldsymbol{w}$ which is sparse in the terms with nonzero values α_j. This occurs, because for large enough values of ϵ, this term forces all Lagrange multipliers α_j toward zero except for the components which are most relevant for classification or regression. If $\boldsymbol{K}^\top \boldsymbol{K}$ does not have full rank, finite values of ϵ lift the degeneracy of the solutions.

Since the primal, eqs. (1), and dual, eqs. (7), optimization problems are different from that of standard SVM methods (there is, for example, no equality constraint), a new SMO method is necessary. We will describe this new SMO method in the following section.

3 The SMO Procedure for the P-SVM Method

3.1 The Optimization Step

The new SMO procedure involves a sequence of optimization steps in which the Lagrangian, eq. (19), is optimized with respect to two Lagrange multipliers α_i and α_k while keeping the remaining parameters fixed. We will start with the description of the optimization step and will then describe the block optimization and annealing heuristics. Because the dual optimization problem of the P-SVM lacks an equality constraint, it is also possible to derive a single SMO procedure, where only one Lagrange parameter is optimized at every step. This single SMO is described in Appendix A. The benchmark results of section 4, however, show, that best results are usually obtained using the "dual" SMO procedure, hence the dual is to be preferred.
Using

$$\begin{aligned} \alpha_j^+ &= \begin{cases} \alpha_j^{\text{old}} + \Delta\alpha_j & \text{if } \ \alpha_j^{\text{old}} + \Delta\alpha_j \geq 0 \\ 0 & \text{if } \ \alpha_j^{\text{old}} + \Delta\alpha_j < 0 \end{cases} \\ \alpha_j^- &= \begin{cases} 0 & \text{if } \ \alpha_j^{\text{old}} + \Delta\alpha_j \geq 0 \\ -\alpha_j^{\text{old}} - \Delta\alpha_j & \text{if } \ \alpha_j^{\text{old}} + \Delta\alpha_j < 0 \end{cases} \end{aligned} \tag{20}$$

[3] $F_j + \epsilon \ = \ F_j - \epsilon \ = \ 0$ is impossible if $\epsilon > 0$.

and setting the derivatives of eqs. (7) with respect to $\Delta\alpha_i$ and $\Delta\alpha_k$ to zero, we obtain

$$\begin{pmatrix} Q_{ii} & Q_{ik} \\ Q_{ki} & Q_{kk} \end{pmatrix} \begin{pmatrix} \Delta\alpha_i \\ \Delta\alpha_k \end{pmatrix} + \begin{pmatrix} F_i \\ F_k \end{pmatrix} + \epsilon \begin{pmatrix} s_i \\ s_k \end{pmatrix} \overset{!}{=} \mathbf{0}, \tag{21}$$

where

$$s_{i,k} = \begin{cases} 1 & \text{if } \alpha_{i,k}^{\text{old}} + \Delta\alpha_{i,k} \geq 0 \\ -1 & \text{if } \alpha_{i,k}^{\text{old}} + \Delta\alpha_{i,k} < 0 \end{cases} .$$

Note that $Q_{ik} = Q_{ki}$ and $\boldsymbol{K}$ is normalized (cf. Hochreiter and Obermayer (2006a)) so that the diagonal elements of $\boldsymbol{Q}$ are equal to 1 ($Q_{ii} = Q_{kk} = 1$). The unconstrained minimum of eq. (19) with respect to α_i and α_k is then given by

$$\begin{aligned} \alpha_i^{\text{new}} &= \alpha_i^{\text{old}} + \Delta\alpha_i , \\ \alpha_k^{\text{new}} &= \alpha_k^{\text{old}} + \Delta\alpha_k , \end{aligned}$$

where

$$\begin{aligned} \Delta\alpha_i &= \frac{(F_k + \epsilon s_k)\ Q_{ik} - (F_i + \epsilon s_i)}{1 - Q_{ik}^2} , \qquad (22) \\ \Delta\alpha_k &= \frac{(F_i + \epsilon s_i)\ Q_{ik} - (F_k + \epsilon s_k)}{1 - Q_{ik}^2} . \end{aligned}$$

The evaluation of eqs. (22) requires an estimation of $s_{i,k}$. For all $j \in \{i, k\}$ we assume $\alpha_j^{\text{new}} < 0$ if $\alpha_j^{\text{old}} < 0$, which implies $s_j = -1$. If $\alpha_j^{\text{old}} > 0$ we assume $\alpha_j^{\text{new}} > 0$, which implies $s_j = 1$. If $\alpha_j^{\text{old}} = 0$ we consider both $s_j = 1$ and $s_j = -1$ and use the solution with the largest $|\Delta\alpha_j|$. If $\alpha_i^{\text{old}} = \alpha_k^{\text{old}} = 0$ we use the solution for which one of the values $|\Delta\alpha_i|$ or $|\Delta\alpha_k|$ is largest.

The unconstrained minimum $(\alpha_i^{\text{new}}, \alpha_k^{\text{new}})$ has now to be checked against the box constraints $L_i \leq \alpha_i^{\text{new}} \leq U_i$ and $L_k \leq \alpha_k^{\text{new}} \leq U_k$,

$$L_i = \begin{cases} -C & \text{if } s_i = -1 \\ 0 & \text{if } s_i = 1 \end{cases} \qquad L_k = \begin{cases} -C & \text{if } s_k = -1 \\ 0 & \text{if } s_k = 1 \end{cases} \tag{23}$$

$$U_i = \begin{cases} 0 & \text{if } s_i = -1 \\ +C & \text{if } s_i = 1 \end{cases} \qquad U_k = \begin{cases} 0 & \text{if } s_k = -1 \\ +C & \text{if } s_k = 1 \end{cases} \tag{24}$$

and the values of the Lagrange multipliers have to be properly corrected, if the unconstrained minimum is located outside this range. The location of the unconstrained optimum can appear in six non-trivially different configurations relative to the position of the box, (Fig. 2A-F). For case (A) the minimum is given by $(\alpha_i^{\text{new}}, \alpha_k^{\text{new}})$. For cases (B) and (C) the minimum can be determined by (1) setting α_i (B) or α_k (C) to the bound which is closest to the unconstrained minimum, (2) calculating F_k (B) or F_i (C) for the new values, and (3) correcting α_k, $\Delta\alpha_k = -F_k - s_k\epsilon$ (B) or α_i, $\Delta\alpha_i = -F_i - s_i\epsilon$ (C), cf. eq. (28). If the new values for α_k (B) or α_i (C) violate the constraints, they have again to be set to the nearest bound. In order to identify case (D) α_k has to be evaluated after steps (1) and (2). For case (D) the new α_k fulfills the previously violated box constraint. If it violates the other constraint, it has to be set to the corresponding bound. Case (E) is identified accordingly. For the remaining case (F) the optimal value is given by setting both α_i and α_k to its nearest corner with respect to the unconstrained minimum.

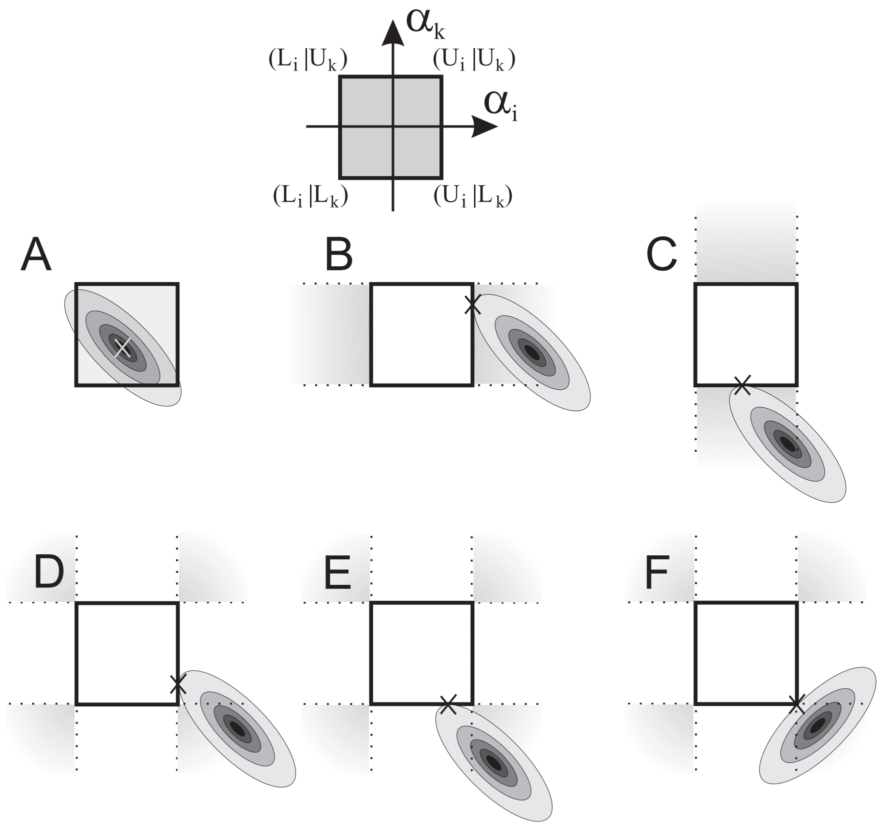

Figure 2: Illustration of the six non-trivial configurations (A - F) of the unconstrained minimum (shaded ellipsoids with contour lines) of the objective function, eq. (19), relative to the box constraints (square box). The crosses indicate the location of the minimum after correction. The shaded areas outside the boxes indicate all possible locations of the unconstrained minimum which correspond to a particular configuration.

After initialization ($\boldsymbol{\alpha} = \mathbf{0}$), SMO iterations start, and for every iteration step two indices i, k must be chosen. Our choice is based on a modified heuristics described in Platt (1999). The Karush-Kuhn-Tucker (KKT) conditions are violated for all α_j for which the corresponding values d_j,

$$d_j = \begin{cases} \max(-F_j - \epsilon, F_j - \epsilon) & \text{if } \alpha_j = 0 \\ |F_j + \epsilon| & \text{if } 0 < \alpha_j < C \\ F_j + \epsilon & \text{if } \alpha_j = C \\ |-F_j + \epsilon| & \text{if } -C < \alpha_j < 0 \\ -F_j + \epsilon & \text{if } \alpha_j = -C \ , \end{cases} \tag{25}$$

are larger than zero. From those α_j we choose α_i such that

$$i = \arg\max_j \ d_j , \quad j = 1, ..., P,$$

and we terminate the SMO if d_i is below a predefined threshold λ. After α_i has been chosen, we compute the location of the unconstrained minimum, evaluate the box constraints for all pairs (i, k), $k \in [1, .., i-1, i+1, .., P]$, and then choose k such that

$$k = \arg\max_j \ \{\max\left(|\Delta\alpha_i| \, , \ |\Delta\alpha_j|\right)\} \, , \quad j = 1, ..., i-1, i+1, ..., P \quad . \tag{26}$$

3.2 Block Optimization

During optimization it often happens that long sequences of SMO iterations occur, which involve the same Lagrange parameters. For example, if three Lagrange parameters are far from optimal, the SMO procedure may perform pairwise updates at every iteration step, during which the value of the third parameter increases its distance from its optimal value. To avoid oscillations we recommend a block update, where eq. (19) is solved simultanously for more than two Lagrange multipliers. The algorithm, which is based on a sequence of Cholesky decompositions and constraint satisfactions, is listed in Appendix B.

A block update is initiated if one of the following conditions is fulfilled: (1) a number b_c of different Lagrange parameters hit either $+C$ or $-C$ since the last block update, (2) a number b_n of different Lagrange parameters were changed since the last block update, or (3) the number of different Lagrange parameters which were updated since the last block update reaches b_f times the number of SMO iterations since the last block update. The first and second condition limit the computational complexity and the number of block updates while the latter condition avoids extended oscillations of the Lagrange parameters. A convenient choice is $b_c = 4$, $b_n = 21$, and $b_f = 3$. The block update is then performed using all Lagrange parameters which are non-zero and which are changed at least once since the last block update.

3.3 ϵ Annealing

In Hochreiter and Obermayer (2006a) it was shown, that the number of support vectors depends on the regularization parameter ϵ. If ϵ is large, many constraints are fulfilled and the number of support vectors is low. The smaller ϵ gets, the larger the number of support vectors becomes. Since SMO algorithms are fast for problems with a small number of support vectors, we suggest to solve the dual optimization using an annealing strategy for the parameter ϵ. Since all KKT conditions (eqs. 16,17) are met for $\epsilon \geq \left\| \boldsymbol{K}^T \boldsymbol{y} \right\|_\infty$, annealing starts with a lower value of ϵ, for example $\epsilon = 0.1 \cdot \left\| \boldsymbol{K}^T \boldsymbol{y} \right\|_\infty$. ϵ is then decreased between SMO iterations according to an exponential schedule (e.g. $\epsilon^{\text{new}} = 0.9 \cdot \epsilon^{\text{old}}$). Since it is not important to find the optimum of eqs. (19) for intermediate values of ϵ the termination parameter λ_{ann} for the SMO procedure is set to $\lambda_{ann} = 4\lambda$ until the selected final value of ϵ is reached.

4 Experiments

In this section we compare our P-SVM implementation with the ϵ-SVR (regression, hyperparameters ϵ and C) and C-SVC (classification, hyperparameter C) from the libsvm implementation (Chang and Lin (2001)) with respect to the computational costs of training and testing, the number of support vectors, and the estimated generalization error of the final predictor. Computing time (in seconds) is measured on a 2 GHz workstation. Training time refers to the number of seconds CPU-time needed until a convergence criterion was reached (P-SVM: $\lambda = 0.05$ C-SVC,ϵ-SVR: $\delta_{term} = 0.001$). The P-SVM is applied using the "dual"-SMO, block optimization, and ϵ-annealing. Testing time refers to the number of seconds spent on predicting the attributes of a test dataset of a given size. Benchmark datasets were chosen from the UCI repository (Newman et al. (1998)) and from `http://ni.cs.tu-berlin.de/software/psvm`, and both hyperparameters ϵ and C were optimized by grid search for each dataset.

Table 1 shows the benchmark results. Libsvm is faster during training, but needs many more support vectors than the P-SVM to obtain its best result. The cross-validation mean squared error (MSE) of the P-SVM is on average below the one of libsvm. The P-SVM learning procedure is computationally more expensive than the libSVM implementation of standard SVMs, but the number of support vectors, i.e. the computing time during test, is much less.

Dataset	P-SVM	libSVM
abalone (UCI):	C=5000, $\epsilon = 0.003$,	C=110, $\epsilon = 1.8$,
RBF kernel, $\gamma = 1$,	no. of SVs: 71,	no. of SVs: 1126,
4177 examples,	training time=45 sec,	training time=5.2 sec,
8 features,	testing time=0.89 sec,	testing time=18 sec,
regression	MSE (20-fold)=4.417	MSE (20-fold)=4.419
space (UCI, rescaled):	C=1000, $\epsilon = 0.0006$,	C=100, $\epsilon = 0.01$,
RBF kernel, $\gamma = 1$,	no. of SVs: 86,	no. of SVs: 2786,
3107 examples,	training time=60 sec,	training time=34 sec,
6 features,	testing time=1.0 sec,	testing time=36 sec,
regression	MSE (10-fold)=0.011	MSE (10-fold)=0.010

Dataset	P-SVM	libSVM
cluster2x (psvm): RBF kernel, $\gamma = 10^{-4}$, 1000 examples, 15 features, regression	C=100, $\epsilon = 0.0016$, no. of SVs: 139, training time=4.5 sec, testing time=2.0 sec, MSE (20-fold)=0.1050	C=70, $\epsilon = 0.19$, no. of SVs: 564, training time=1.3 sec, testing time=13 sec, MSE (20-fold)=0.1187
cluster2a (psvm): RBF kernel, $\gamma = 10^{-4}$, 200 examples, 10 features, regression	C=28, $\epsilon = 0.0013$, no. of SVs: 149, training time=0.26 sec, testing time=2.0 sec, MSE (30-fold)=1.17	C=1000, $\epsilon = 0.6$, no. of SVs: 130, training time=0.32 sec, testing time=4.5 sec, MSE (30-fold)=3.09
heart (UCI, rescaled): RBF kernel, $\gamma = 0.05$, 270 examples, 13 features, regression	C=1000, $\epsilon = 0.44$, no. of SVs: 9, training time=0.031 sec, testing time=0.12 sec, MSE (50-fold)=0.483	C=0.6, $\epsilon = 0.41$, no. of SVs: 131, training time=0.11 sec, testing time=5.5 sec, MSE (50-fold)=0.495
arcene (NIPS2003): linear kernel, 100 examples, 10000 features, classification	C=100, $\epsilon = 0.002$, no. of SVs: 83, training time=0.30 sec, testing time=0.33 sec, accuracy=86% (30-fold)	C=100, no. of SVs: 79, training time=0.59 sec, testing time=1.6 sec, accuracy=87% (30-fold)

Table 1: Comparison between the P-SVM and the libSVM implementations of ϵ-SVR and C-SVC with respect to: the number of support vectors, CPU-time for training (in seconds), CPU-time for testing (in seconds), the mean squared generalization error (MSE) for ϵ-SVR and the percentage of correctly classified datapoints for C-SVC, determined using n-fold cross-validation. CPU-time for testing is obtained for a dataset of 10^5 (*arcene*: 100) examples. γ denotes the width of the selected RBF kernel, Each feature of the datasets *heart* and *space* was linearly scaled to the range $[-1, +1]$. The SMO-parameters for the P-SVM are $\lambda = 0.05, \lambda_{ann} = 4, b_c = 4, b_n = 21, b_f = 3$, for libSVM we used the default configuration $\delta_{term} = 0.001$, shrinking activated, and kernel caching with 40 MB.

Table 2 shows benchmark results for the different SMO-variants for the P-SVM implementation, including single- vs. dual-SMO, block optimization and ϵ-annealing. Best results are obtained for the dual-SMO implementation using both block optimization and ϵ-annealing. In numerical simulation one often observes that support vectors for high values ϵ remain support vectors also for lower values. Since the number of Lagrange multipliers which are changed at least once during the SMO procedure is always greater or equal to the final number of nonzero elements, ϵ-annealing significantly reduces the number of Lagrange multipliers which are changed at least once, but needs a somewhat larger number of SMO iterations. If the reduction in CPU-time due to the smaller number of rows of $\boldsymbol{Q}$, which need to be calculated ($O(P \cdot L)$ per row of $\boldsymbol{Q}$), is larger than the increase in time due to the larger number of SMO iterations, ϵ-annealing should be used. This is usually the case, if block optimization and the dual-SMO is applied. Then ϵ-annealing provides an additional computational advantage (cf. table 2).

dbl	blk	ann	abalone		space		cluster2x		arcene	
+	+	+	45	(255)	60	(427)	4.5	(366)	0.30	(194)
+	+	-	90	(1147)	82	(1351)	4.9	(878)	0.12	(195)
+	-	+	1618	(260)	663	(373)	67	(347)	1.26	(192)
+	-	-	1517	(982)	623	(1282)	16	(815)	0.51	(194)
-	+	+	101	(201)	60	(343)	8.8	(363)	1.68	(188)
-	+	-	85	(329)	56	(580)	6.9	(459)	0.75	(191)
-	-	+	5850	(156)	1214	(240)	184	(351)	27	(188)
-	-	-	4848	(182)	1030	(300)	171	(370)	13	(187)

Table 2: Benchmark result for the different SMO variants of the P-SVM implementation. The columns 'dbl' (1), 'blk' (2), and 'ann' (3) indicate whether the dual-SMO instead of the single-SMO (1), block optimization (2), or ϵ-annealing (3) was applied. Numbers denote the CPU-time (in seconds) needed for solving the dual problem of the PSVM using hyperparameters which minimize the cross-validation error. The total number of evaluated rows of $\boldsymbol{Q}$ is given by numbers in parantheses and equals the number of different Lagrange parameters which were changed at least once during the SMO procedure.

Table 3 shows, how computation time scales with the number of training examples for the dataset ‘adult’ from the UCI repository, if an RBF-kernel is applied. Training time is approximately a factor of two larger for the P-SVM, but on the same order of magnitude for the same generalization performance (79%, 5-fold cross-validation). The SMO implementation of the P-SVM is, therefore, competitive to standard SVM methods for standard SVM problems.

number of examples	P-SVM τ_n	P-SVM $n_{\boldsymbol{Q}}$	C-SVC τ_n
1000	0.6	76	1
2000	3.1	142	3
3000	8.0	166	8
4000	17	202	9
5000	29	226	14
6000	45	243	20
7000	67	270	35
8000	96	292	42
9000	132	329	55
10000	166	330	63

Table 3: Comparison between the P-SVM and the C-SVC of the libSVM implementation for the dataset 'adult' with respect to the CPU-time τ (in seconds) as a function of the training set size. $n_{\boldsymbol{Q}}$ denotes the total number of evaluated rows of $\boldsymbol{Q}$. The parameters are: RBF-kernel $\gamma = 10^{-8}$, P-SVM: $\epsilon = 0.4$, $C = 1$, C-SVC: $C = 10$.

The P-SVM, however, is the only method so far, which can handle indefinite kernel functions and arbitrary dyadic data. This feature can lead to an enormous speed-up of the P-SVM compared to standard SVM methods, as is shown in table 4 for the dataset 'adult'. Table 4 shows benchmark results of the P-SVM in dyadic data mode and the C-SVC of libSVM for a linear kernel. While the generalization performance (measured through 5-fold cross-validation) is 84% for the P-SVM converged to 79% for the C-SVC, the computation time is a factor of 8554. CPU-time is roughly proportional to the number of training objects for both the P-SVM and C-SVC, while the training time is much less for the P-SVM.

number of examples	P-SVM τ_n	C-SVC τ_n
10000	0.30	3195
20000	0.58	6780
30000	0.97	9810
40000	1.48	12660

Table 4: Comparison between the P-SVM and the C-SVC of the libSVM implementation for the dataset 'adult' with respect to the CPU-time τ (in seconds) as a function of the training set size. The P-SVM operates in dyadic mode directly on the dataset, while a linear kernel is used for the C-SVC. The parameters are $\epsilon = 1$, $C = 1000$ for the P-SVM, and $C = 1$ for the C-SVC.

Acknowledgements

We thank A. Paus, C. Sannelli, and J. Mohr for testing the SMO. This work was funded by the BMBF (grant no. 01GQ0414) and BMWA (grant no. 01MD501).

A The Single SMO Procedure

Due to the missing equality constraint in the dual problem of the P-SVM, an SMO procedure can be derived where only one Langrange multiplier is optimized in every iteration. Setting the derivatives of eq. (7) with respect to $\Delta\alpha_i$ to zero and using eqs. (20), we obtain

$$Q_{ii} \cdot \Delta\alpha_i + F_i + \epsilon s_i \overset{!}{=} 0 \tag{27}$$

where

$$s_i = \begin{cases} 1 & \text{if } \alpha_i^{\text{old}} + \Delta\alpha_i \geq 0 \\ -1 & \text{if } \alpha_i^{\text{old}} + \Delta\alpha_i < 0 \end{cases} .$$

Note that $\boldsymbol{K}$ is normalized (cf. Hochreiter and Obermayer (2006a)) so that the diagonal elements of $\boldsymbol{Q}$ are equal to 1. The unconstrained minimum of eq. (19) with respect to α_i is then given by

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + \Delta\alpha_i ,$$

where

$$\Delta\alpha_i = - F_i - \epsilon s_i . \tag{28}$$

The evaluation of eq. (28) requires an estimation of s_i. We assume $\alpha_i^{\text{new}} < 0$ if $\alpha_i^{\text{old}} < 0$, which implies $s_i = -1$. If $\alpha_i^{\text{old}} > 0$ we assume $\alpha_i^{\text{new}} > 0$, which implies $s_i = 1$. If $\alpha_i^{\text{old}} = 0$ we consider both $s_i = 1$ and $s_i = -1$ and use the solution with the largest $|\Delta\alpha_i|$. The application of the box constraints is similar to the dual case described in section 3.

B A Recursive Algorithm for the Block Update

The block update procedure for solving eqs. (19) can be described using the following pseudocode:

```
function  $\boldsymbol{\alpha}^{new} = optimize(\boldsymbol{\alpha}^{old})$
{
  $\boldsymbol{\alpha}^{new}$ := $solvecholesky(\boldsymbol{\alpha}^{old})$
  $\boldsymbol{\alpha}^{new}$ := $bound(\boldsymbol{\alpha}^{new})$
  $\boldsymbol{\alpha}^{new}_{bound}$ := $ListOfBoundedAlphas(\boldsymbol{\alpha}^{new})$
  if ($|\boldsymbol{\alpha}^{new}_{bound}| > 4$)  return  $\boldsymbol{\alpha}^{old}$
  if ($|\boldsymbol{\alpha}^{new}_{bound}| > 0$)  $\boldsymbol{\alpha}^{new}$ := $reoptimize(\boldsymbol{\alpha}^{new}, \boldsymbol{\alpha}^{new}_{bound}, \boldsymbol{\alpha}^{new}_{bound}$.front)
  return  $\boldsymbol{\alpha}^{new}$
}

function  $\boldsymbol{\alpha}^{new} = reoptimize(\boldsymbol{\alpha}^{old}, \boldsymbol{\alpha}^{old}_{bound}, b)$
{
  $\boldsymbol{\alpha}^{new}$ := $optimize(\boldsymbol{\alpha}^{old} \setminus b)$  |  $apply(b)$
  $\forall i \in \boldsymbol{\alpha}^{old}_{bound} \setminus b$
    $i^{new}$ := find element from $\boldsymbol{\alpha}^{new}$ which belongs to $i$
    if ($i^{new}$ == $i$)
      $\boldsymbol{\alpha}^{new}$ := $reoptimize(\boldsymbol{\alpha}^{old}, \boldsymbol{\alpha}^{old}_{bound} \setminus b, i)$
      return  $\boldsymbol{\alpha}^{new}$
    endif
  return  $\boldsymbol{\alpha}^{new}$
}
```

The central function is *optimize*, which recursively uses *optimize* and *reoptimize*. $\boldsymbol{\alpha}^{old}$ and $\boldsymbol{\alpha}^{new}$ are sets of Lagrange multipliers, which are implemented as linked lists of variable lengths. $\boldsymbol{\alpha}^{old}_{bound}$ and $\boldsymbol{\alpha}^{new}_{bound}$ are subsets of the corresponding $\boldsymbol{\alpha}^{old}$ and $\boldsymbol{\alpha}^{new}$. b and i are single elements of $\boldsymbol{\alpha}^{old}_{bound}$ or $\boldsymbol{\alpha}^{new}_{bound}$. The function *solvecholesky* solves the unconstrained problem using the Cholesky decomposition and the function *bound* sets all Lagrange parameters which violate a constraint to their nearest bounds. *ListOfBoundedAlphas* returns a linked list of Lagrange multipliers whose values are on a bound. The function *optimize* is followed by "| $apply(b)$", which updates $\boldsymbol{F}$ (cf. eq. (9)) using the Lagrange parameter b. After using the modified $\boldsymbol{F}$ during the corresponding optimization procedure, $\boldsymbol{F}$ is restored to its old values. The operator "$\boldsymbol{\alpha} \setminus b$" removes Lagrange parameter b from the linked list of Lagrange parameters $\boldsymbol{\alpha}$. The algorithm terminates after finding an exact solution, or if more than four Lagrange parameters reach the bound.

References

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

S. Hochreiter and K. Obermayer. Support vector machines for dyadic data. *Neural Computation*, 18(6):1472–1510, 2006a.

S. Hochreiter and K. Obermayer. Nonlinear feature selection with the potential support vector machine. In *Feature Extraction: Foundations and Applications* I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, 419-438, Springer Publishers, 2006b.

S. Hochreiter and K. Obermayer. *Gene Selection for Microarray Data.* In B. Schölkopf and K. Tsuda and J.-P. Vert, editors, Kernel Methods in Computational Biology, pages 319-355, MIT Press, 2004.

P. D. Hoff. Bilinear Mixed-Effects Models for Dyadic Data. *Journal of the American Statistical Assosciation*, 100(469):286–295, 2005.

T. Hofmann and J. Puzicha. Unsupervised Learning from Dyadic Data. Technical Report TRT-98-042, *International Computer Science Insitute*, Berkeley, CA, 1998.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, 13(3):637–649, 2001.

S. S. Keerthi, S. K. Shevade. SMO algorithm for least-squares SVM formulations. *Neural Computation*, 15(2):487–507, 2003.

D. Lai, N. Mani, and M. Palaniswami. A new method to select working sets for faster training for support vector machines. Technical Report MESCE-30-2003, *Dept. Electrical and Computer Systems Engineering Monash University*, Australia, 2003.

D.J. Newman, S. Hettich, C.L. Blake, C.J. Merz. UCI Repository of machine learning databases, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Irvine, CA: University of California, Department of Information and Computer Science, 1998.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

B. Schölkopf and A. J. Smola. *Learning with kernels – Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, 2002.

V. N. Vapnik. *Statistical Learning Theory.* Adaptive and learning systems for signal processing, communications, and control. Wiley, New York, 1998.