SoftDoubleMaxMinOver: Perceptron-like Training of Support Vector Machines

Thomas Martinetz (Senior Member IEEE)
Institute for Neuro- and Bioinformatics
University of Lübeck
D-23538 Lübeck
martinetz@informatik.uni-luebeck.de
Kai Labusch
Institute for Neuro- and Bioinformatics
University of Lübeck
D-23538 Lübeck
labusch@inb.uni-luebeck.de
Daniel Schneegaß
Siemens AG, Corporate Technology
Learning Systems
D-81739 Munich
daniel.schneegass.ext@siemens.com

keywords: incremental learning, Support Vector Machine, maximum margin classification

***Abstract*—The well-known MinOver algorithm is a slight modification of the perceptron algorithm and provides the maximum margin classifier without a bias in linearly separable two class classification problems. DoubleMinOver as an extension of MinOver, which now includes a bias, is introduced. An $\mathcal{O}(t^{-1})$ convergence is shown, where t is the number of learning steps. The computational effort per step increases only linearly with the number of patterns. In its formulation with kernels selected training patterns have to be stored. A drawback of MinOver and DoubleMinOver is that this set of patterns does not consist of support vectors only. DoubleMaxMinOver, as an extension of DoubleMinOver, overcomes this drawback by selectively forgetting all non-support vectors after a finite number of training steps. It is shown how this iterative procedure that is still very similar to the perceptron algorithm can be extended to classification with soft margins and be used for training least squares Support Vector Machines (SVM). On benchmarks the SoftDoubleMaxMinOver algorithm achieves the same performance as standard SVM software.**

I. Introduction

The Support Vector Machine (SVM) [1], [2] has become a standard tool in classification and regression tasks (e.g. [3], [4], [5]). A major drawback, particularly for industrial applications where easy and robust implementation is an issue, is its complicated training procedure. A large Quadratic-Programming problem has to be solved, requiring sophisticated numerical optimization routines which many users do not want or cannot implement by themselves. They have to rely on existing software packages, which are hardly comprehensive and, at least in some cases, error-free. This is in contrast to most Neural Network approaches where learning has to be simple and incremental almost by definition. The pattern-by-pattern nature of learning in Neural Networks like, for example, the perceptron usually leads to simple training procedures which can easily be implemented. It is desirable to have similar training procedures also for the SVM.

Several approaches for obtaining more or less simple incremental learning algorithms for the SVM have been introduced [6], [7], [8], [9], [10], [11], [12], [13]. Among the first and most widespread are the Kernel-Adatron by Friess, Cristianini, and Campbell [6] and the Sequential-Minimal-Optimization algorithm (SMO) by Platt [7]. Like the MinOver algorithm by Krauth and Mézard [14], which we will build on, the Adatron was introduced for constructing synaptic weight matrices of optimal stability in spin-glass models of Neural Networks [15], [16]. Friess et al. adapted the Adatron to the problem of maximum margin classification with kernels. The Adatron and the MinOver algorithm are very similar and can both be derived from constrained gradient descent. The Adatron converges faster; however, the MinOver learning step is even simpler (exponential versus $1/t$ convergence, quadratic versus linear time complexity per iteration step). Navone and Downs [17] give a comparison of both algorithms on common benchmark problems. They report that for reasons which require further investigations the MinOver algorithm shows a learning behaviour which is advantageous in a number of aspects.

Vishwanathan et al. [12] present a method which guarantees to converge to the exact maximum margin solution after a finite number of steps. They use an active set strategy and claim to be considerably faster than the SMO algorithm by Platt [7]. Their work is based on an approach of Cauwenberghs and Poggio [10], which converges after n iterations, where n is the number of considered samples. Their method has the additional advantage of being able to learn incrementally and decrementally, thereby enabling an efficient calculation of the leave-one-out error. Another approach for incremental or decremental learning was introduced by Kim et al. [18]. Their approach is based on updating and downdating the QR decomposition of the least squares Support Vector Machine, which allows to efficiently compute the updated solution when data points are added or removed. In [19] they show how their method can be applied to drug design by iteratively including new compounds into the training set.

Another interesting approach is the work of Tsang et al. [13]. Training an SVM is reduced to the problem of finding the minimum enclosing ball around a set of points. They apply the so-called core set, which has been introduced for finding approximate solutions for the minimum enclosing ball (MEB) problem [13]. The method converges fast as the convergence rate does not depend on the number of samples, but it only provides approximate solutions. The loss in precision speeds up the algorithm.

In this paper, we will revisit and extend the MinOver algorithm to obtain perceptron-like procedures for training Support Vector Machines. Their learning steps can be motivated from gradient descent. With their pattern-by-pattern nature (in the sense that, like for the perceptron, one pattern after the other is chosen for learning from a given training set) these procedures are suitable for incremental and decremental learning. After a finite number of iterations only support vectors are taken into account. The method can easily be extended to kernels while keeping its simplicity.

First, we will reformulate MinOver. In this reformulation the MinOver algorithm is a slight modification of the perceptron learning rule and could hardly be simpler. Then, in Section II, we extend MinOver to DoubleMinOver which remains as simple as MinOver but now provides the maximum margin solution also for linear classifiers with a bias. We present a convergence proof and show that an $\mathcal{O}(t^{-1})$ convergence can be expected, where t is the number of learning steps.

The convergence of MinOver to the maximum margin hyperplane in linearly separable classification tasks is well-known; however, the solution is not based solely on support vectors. The same is true for DoubleMinOver. Therefore, in Section III we extend DoubleMinOver to DoubleMaxMinOver in a similar way as we extended MinOver to MaxMinOver in [20]. The DoubleMaxMinOver algorithm provides the maximum margin hyperplane based only on support vectors. In the last part of this paper we will show how DoubleMaxMinOver can be extended to SoftDoubleMaxMinOver for classification with soft margins. SoftDoubleMaxMinOver yields solutions also in cases where the classification problem is not linearly separable. In the last section we will present results and comparisons on common benchmark problems.

II. The DoubleMinOver Algorithm

Given a set of patterns $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^D, i = 1,\dots,N\}$ with corresponding class labels $y_i \in \{-1,1\}, i = 1,\dots,N$ such that the class $\mathcal{X}^+ \subseteq \mathcal{X}$ of patterns with label $+1$ and the class $\mathcal{X}^- \subseteq \mathcal{X}$ of patterns with label -1 are linearly separable. We want to find the hyperplane which separates the patterns of these two classes with maximum margin. The hyperplane for classification is determined by its normal vector $\mathbf{w} \in \mathbb{R}^D$ and its bias $b \in \mathbb{R}$. It achieves a separation of the two classes, if

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) > 0 \qquad \text{for all} \qquad i = 1,\dots,N\,.$$

The margin Δ of this separation is given by

$$\Delta(\mathbf{w}, b) = \min_{\mathbf{x}_i \in \mathcal{X}} \left[y_i(\mathbf{w}^T\mathbf{x}_i - b)/\|\mathbf{w}\| \right].$$

Maximum margin classification is obtained by the $\mathbf{w}_*$, $\|\mathbf{w}_*\| = 1$ and b_* for which $\Delta(\mathbf{w}_*, b_*) = \Delta^*$ becomes maximal.

The Support Vector Machine (SVM) as introduced in [1], [21] determines $\mathbf{w}_*$ and b_* by solving a quadratic programming problem. It minimizes $\|\mathbf{w}\|^2$ under the constraints $y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1, i = 1,\dots,N$, the so-called hard margin case. The solution is equivalent to $\mathbf{w}_*$ up to a scaling factor. In its dual formulation one has to find $\alpha_i \in \mathbb{R}_0^+, i = 1,\dots,N$ which maximize

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

under the constraint $\sum_{i=1}^N y_i\alpha_i = 0$. The solution yields

$$\mathbf{w}_* = \sum_{i=1}^{N} y_i \alpha_i \mathbf{x}_i\,. \tag{1}$$

Only a subset of the α_i will be non-zero. Those $\mathbf{x}_i$ for which $\alpha_i > 0$ are called support vectors [21]. For them $y_i(\mathbf{w}_*^T\mathbf{x}_i - b_*) = \Delta^*$ is valid. Hence, the set $\mathcal{X}_S \subseteq \mathcal{X}$ of these support vectors determine $\mathbf{w}_*$ and b_*. In the following we introduce a class of algorithms which determine $\mathbf{w}_*$ and b_* iteratively, i.e., which solve the quadratic programming problem of the hard-margin SVM by employing a perceptron-like pattern-by-pattern optimization procedure.

A simple, perceptron-like pattern-by-pattern algorithm which provides the maximum margin classification in linearly separable cases is the well-known MinOver algorithm introduced by [14] in the context of constructing synaptic weight matrices of optimal stability in spin-glass models of Neural Networks. However, it only provides the maximum margin solution if no bias b is included. In the MinOver algorithm the current normal vector $\mathbf{w}_t$ converges to $\mathbf{w}_*$ with increasing number of iterations t. This is valid as long as a full separation, i.e. a $\mathbf{w}_*$ with $\Delta^* > 0$ exists. The MinOver algorithm works like the perceptron algorithm. However, instead of choosing an arbitrary misclassified pattern out of the training set $\mathcal{X}$, with each training step t the "mostly misclassified" pattern is chosen. The "mostly misclassified" pattern $\mathbf{x}_{\min}(t)$ is the one with minimum distance (overlap) $y_i\mathbf{w}^T\mathbf{x}_i$ from the separating hyperplane ($b = 0$). Hence the name MinOver. In contrast to the perceptron, however, learning proceeds even if all patterns are classified correctly. In this case the distance $y_i\mathbf{w}^T\mathbf{x}_i$ of the "mostly misclassified" pattern $\mathbf{x}_{\min}(t)$ has become positive.

As a first step we slightly modify MinOver such that a bias b can be included. Instead of looking for the pattern with minimum distance in $\mathcal{X}$, now we look for the pattern

$\mathbf{x}_{\min^+}(t) \in \mathcal{X}^+$ with minimum distance in $\mathcal{X}^+$ and for the pattern $\mathbf{x}_{\min^-}(t) \in \mathcal{X}^-$ with minimum distance in $\mathcal{X}^-$ and use both for the next adaptation step. Hence the name DoubleMinOver. In Algorithm 1 the pseudo code for DoubleMinOver is listed.

Algorithm 1 With $t_{\max}$ denoting the number of desired iterations, DoubleMinOver works as follows:

```
$\mathbf{w} \leftarrow 0$
**for** $t = 0$ to $t_{\max}$ **do**
    $\mathbf{x}_{\min^+} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^+} \left(\mathbf{w}^T \mathbf{x}_i\right)$
    $\mathbf{x}_{\min^-} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^-} \left(-\mathbf{w}^T \mathbf{x}_i\right)$
    $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_{\min^+} - \mathbf{x}_{\min^-}$
**end for**
$b \leftarrow \frac{1}{2} \mathbf{w}^T \left(\mathbf{x}_{\min^+} + \mathbf{x}_{\min^-}\right)$.
```

Why does it work? For a given $\mathbf{w}$, the margin $\Delta(\mathbf{w}, b)$ is maximized if $b(\mathbf{w})$ is chosen such that the margin to both classes is equal, i.e., if

$$\min_{\mathbf{x}_i \in \mathcal{X}^+} \left[\mathbf{w}^T \mathbf{x}_i - b(\mathbf{w})\right] = \min_{\mathbf{x}_i \in \mathcal{X}^-} \left[-(\mathbf{w}^T \mathbf{x}_i - b(\mathbf{w}))\right] .$$

This leads to the expression for the bias in Algorithm 1

$$b(\mathbf{w}) \quad = \quad \frac{1}{2} \left(\mathbf{w}^T \mathbf{x}_{\min^+} + \mathbf{w}^T \mathbf{x}_{\min^-}\right) .$$

DoubleMinOver then looks for the $\mathbf{w}$ which maximizes

$$\begin{aligned} \Delta(\mathbf{w}) \quad &:= \quad \Delta(\mathbf{w}, b(\mathbf{w})) \\ &= \quad \min_{\mathbf{x}_i \in \mathcal{X}} \frac{y_i(\mathbf{w}^T \mathbf{x}_i - b(\mathbf{w}))}{\|\mathbf{w}\|} \\ &= \quad \min_{\mathbf{x}_i \in \mathcal{X}^+} \frac{\mathbf{w}^T \mathbf{x}_i - b(\mathbf{w})}{\|\mathbf{w}\|} = \min_{\mathbf{x}_i \in \mathcal{X}^-} \frac{-\mathbf{w}^T \mathbf{x}_i + b(\mathbf{w})}{\|\mathbf{w}\|} \\ &= \quad \frac{1}{2} \left(\min_{\mathbf{x}_i \in \mathcal{X}^+} \frac{\mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|} + \min_{\mathbf{x}_i \in \mathcal{X}^-} \frac{-\mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|} \right) \\ &= \quad \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T \left(\mathbf{x}_{\min^+} - \mathbf{x}_{\min^-}\right) . \end{aligned}$$

In this form the adaptation step for $\mathbf{w}_t$ in DoubleMinOver can be motivated from gradient ascent on $\Delta(\mathbf{w})$. Note, however, that this view is just a motivation and, e.g., neglects the normalization of $\mathbf{w}$.

Note that it is sufficient to determine b only once at the end of the training procedure. This is due to the fact that $\mathbf{x}_{\min^+}(t)$ and $\mathbf{x}_{\min^-}(t)$ do not depend on the choice of b.

A. On the convergence of DoubleMinOver

Krauth and Mézard proved an $\mathcal{O}(t^{-1/2})$ convergence for MinOver [14] within the context of spin-glass Neural Networks. Here, we adapt this proof to DoubleMinOver. In addition, we show that DoubleMinOver converges even as $\mathcal{O}(t^{-1})$. First, we introduce some notations and have to prove some lemmata.

Notations:

i) With

$$\mathcal{Z} := \left\{\mathbf{x}^+ - \mathbf{x}^- \mid \mathbf{x}^+ \in \mathcal{X}^+, \mathbf{x}^- \in \mathcal{X}^-\right\}$$

we can write

$$\Delta(\mathbf{w}) = \frac{1}{2} \min_{\mathbf{z} \in \mathcal{Z}} \frac{\mathbf{w}^T \mathbf{z}}{\|\mathbf{w}\|} \quad \text{and} \quad \Delta^* = \Delta(\mathbf{w}_*) .$$

With $\mathbf{z}_{\min}(t) := \arg \min_{\mathbf{z} \in \mathcal{Z}} \mathbf{w}_t^T \mathbf{z} = \mathbf{x}_{\min^+}(t) - \mathbf{x}_{\min^-}(t)$ we obtain for the learning step

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{z}_{\min}(t) .$$

ii) Let $\mathcal{X}_S^+ := \mathcal{X}_S \cap \mathcal{X}^+$ be the set of support vectors in $\mathcal{X}^+$ and $\mathcal{X}_S^- := \mathcal{X}_S \cap \mathcal{X}^-$ be the set of support vectors in $\mathcal{X}^-$. We call

$$\mathcal{Z}_S := \left\{\mathbf{x}^+ - \mathbf{x}^- \mid \mathbf{x}^+ \in \mathcal{X}_S^+, \mathbf{x}^- \in \mathcal{X}_S^-\right\}$$

the set of support vectors in $\mathcal{Z}$. For each $\mathbf{z} \in \mathcal{Z}_S$ $\mathbf{w}_*^T \mathbf{z} = 2\Delta^*$ is valid.

iii) γ_t denotes the angle between $\mathbf{w}_t$ and $\mathbf{w}_*$.

iv) $\mathbf{u}_t := \mathbf{w}_t - (\mathbf{w}_*^T \mathbf{w}_t)\mathbf{w}_*$ denotes the projection of $\mathbf{w}_t$ onto the plane perpendicular to $\mathbf{w}_*$, i.e., it holds

$$\mathbf{w}_t = \cos\gamma_t \|\mathbf{w}_t\| \mathbf{w}_* + \mathbf{u}_t \qquad \text{with} \qquad \mathbf{u}_t^T \mathbf{w}_* = 0 . \quad (2)$$

v) $\mathbf{s}(\mathbf{z}) := \mathbf{z} - (\mathbf{w}_*^T \mathbf{z})\mathbf{w}_*$ denotes the projection of $\mathbf{z} \in \mathcal{Z}$ onto the plane perpendicular to $\mathbf{w}_*$. This yields $\mathcal{S} := \{\mathbf{s}(\mathbf{z}) \mid \mathbf{z} \in \mathcal{Z}\}$ as the set of projections and $\mathcal{S}_S := \{\mathbf{s}(\mathbf{z}) \mid \mathbf{z} \in \mathcal{Z}_S\}$ as its subset where the $\mathbf{z}$ are support vectors.

Lemma 1: For each $t \in \mathbb{N}_0$ it holds $\mathbf{u}_t^T \mathbf{z}_{\min}(t) \leq 0$.

Proof: We introduce $\mathbf{z}_\varepsilon := \arg \min_{\mathbf{z} \in \mathcal{Z}} (\mathbf{w}_* + \varepsilon \mathbf{u}_t)^T \mathbf{z}$, $\varepsilon \in \mathbb{R}_0^+$. For $\varepsilon = 0$, $\mathbf{z}_\varepsilon \in \mathcal{Z}_S$ is valid. Since $\mathcal{Z}$ is a discrete and finite set of points in $\mathbb{R}^D$, there is an $a \in \mathbb{R}^+$ and $\mathbf{z}_0 \in \mathcal{Z}_S$ such that for $0 \leq \varepsilon < a$ $\mathbf{z}_\varepsilon = \mathbf{z}_0$ is valid. $\mathbf{u}_t^T \mathbf{z}_0 \leq 0$, since otherwise by choosing $\varepsilon \in \mathbb{R}^+$ sufficiently small

$$\begin{aligned} \Delta(\mathbf{w}_* + \varepsilon \mathbf{u}_t) \quad &= \quad \frac{1}{2} \min_{\mathbf{z} \in \mathcal{Z}} \frac{(\mathbf{w}_* + \varepsilon \mathbf{u}_t)^T \mathbf{z}}{\|\mathbf{w}_* + \varepsilon \mathbf{u}_t\|} \\ &= \quad \frac{1}{2} \frac{(\mathbf{w}_* + \varepsilon \mathbf{u}_t)^T \mathbf{z}_0}{\|\mathbf{w}_* + \varepsilon \mathbf{u}_t\|} \\ &= \quad \frac{1}{2} \frac{2\Delta^* + \varepsilon \mathbf{u}_t^T \mathbf{z}_0}{(1 + \varepsilon^2 \|\mathbf{u}_t\|^2)^{\frac{1}{2}}} \\ &\geq \quad \frac{1}{2} \left(2\Delta^* + \varepsilon \mathbf{u}_t^T \mathbf{z}_0\right) \left(1 - \frac{1}{2} \varepsilon^2 \|\mathbf{u}_t\|^2\right) \\ &\geq \quad \Delta^* \\ &\quad\; + \frac{\varepsilon}{2} \left(\mathbf{u}_t^T \mathbf{z}_0 - \frac{1}{2} \varepsilon \|\mathbf{u}_t\|^2 \left(2\Delta^* + \varepsilon \mathbf{u}_t^T \mathbf{z}_0\right) \right) \\ &> \quad \Delta^* , \end{aligned}$$

i.e., a larger margin than Δ^* would result by choosing $\mathbf{w}_* + \varepsilon \mathbf{u}_t$ instead of $\mathbf{w}_*$ (contradiction).

With Equation (2) we obtain $\mathbf{w}_t^T \mathbf{z}_0 = 2 \cos\gamma_t \|\mathbf{w}_t\| \Delta^* + \mathbf{u}_t^T \mathbf{z}_0$. Further, $\mathbf{w}_t^T \mathbf{z}_{\min}(t) \geq 2 \cos\gamma_t \|\mathbf{w}_t\| \Delta^* + \mathbf{u}_t^T \mathbf{z}_{\min}(t)$. Subtracting both equations, we obtain $\mathbf{w}_t^T \mathbf{z}_{\min}(t) - \mathbf{w}_t^T \mathbf{z}_0 \geq \mathbf{u}_t^T \mathbf{z}_{\min}(t) - \mathbf{u}_t^T \mathbf{z}_0$. Since $0 \geq \mathbf{w}_t^T \mathbf{z}_{\min}(t) - \mathbf{w}_t^T \mathbf{z}_0$, also $0 \geq \mathbf{u}_t^T \mathbf{z}_{\min}(t) - \mathbf{u}_t^T \mathbf{z}_0$ and, hence, $\mathbf{u}_t^T \mathbf{z}_{\min}(t) \leq \mathbf{u}_t^T \mathbf{z}_0$. Since $\mathbf{u}_t^T \mathbf{z}_0 \leq 0$, we can conclude the proof. ■

Theorem 1: Let $\Delta_t = \Delta(\mathbf{w}_t)$ be the margin provided by DoubleMinOver after t iteration steps and $(\Delta^* - \Delta_t)/\Delta^*$ be its relative deviation from the maximum margin. For each $t > 0$

$$0 \leq \frac{\Delta^* - \Delta_t}{\Delta^*} \leq \frac{(R/\Delta^*)^2}{\sqrt{t}} + \frac{1}{2}\frac{(R/\Delta^*)^2}{t}$$

is valid, with $R := \max_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x}\|$.

Proof: First, we show that $\|\mathbf{u}_t\|^2 \leq 4R^2 t$. This can be seen from

$$\begin{aligned} \|\mathbf{u}_{t+1}\|^2 - \|\mathbf{u}_t\|^2 &= \|\mathbf{u}_t + \mathbf{z}_{\min}(t) - [\mathbf{z}_{\min}(t)^T\mathbf{w}_*]\mathbf{w}_*\|^2 \\ &\quad -\|\mathbf{u}_t\|^2 \\ &= 2\mathbf{u}_t^T\mathbf{z}_{\min}(t) + \|\mathbf{z}_{\min}(t)\|^2 \\ &\quad -[\mathbf{z}_{\min}(t)^T\mathbf{w}_*]^2 \\ &\leq \|\mathbf{z}_{\min}(t)\|^2 \\ &\leq 4R^2 \ , \end{aligned}$$

where we have used $\mathbf{u}_t^T\mathbf{w}_* = 0$ and Lemma 1 (note that we start with $\mathbf{u}_0 = 0$). Since also

$$\mathbf{w}_*^T\mathbf{w}_t = \mathbf{w}_*^T\sum_{\tau=0}^{t-1}\mathbf{z}_{\min}(\tau) \geq 2\Delta^*\, t$$

is valid, we obtain the bounds

$$\sin\gamma_t \leq \gamma_t \leq \tan\gamma_t = \frac{\|\mathbf{u}_t\|}{\mathbf{w}_*^T\mathbf{w}_t} \leq \frac{2R\sqrt{t}}{2\Delta^* t} = \frac{R/\Delta^*}{\sqrt{t}} \ . \tag{3}$$

Together with Equation (2) this yields

$$\begin{aligned} 2\Delta^* \geq 2\Delta_t &= \frac{\mathbf{w}_t^T\mathbf{z}_{\min}(t)}{\|\mathbf{w}_t\|} \\ &= \mathbf{w}_*^T\mathbf{z}_{\min}(t)\cos\gamma_t \\ &\quad + \frac{\mathbf{u}_t^T\mathbf{z}_{\min}(t)}{\|\mathbf{u}_t\|}\sin\gamma_t \qquad (4) \\ &\geq 2\Delta^*\cos\gamma_t - 2R\sin\gamma_t \\ &\geq 2\Delta^*(1-\gamma_t^2/2) - 2R\gamma_t \\ &\geq 2\Delta^* - 2\frac{R^2/\Delta^*}{\sqrt{t}} - \frac{R^2/\Delta^*}{t} \ . \end{aligned}$$

A proper rearrangement concludes the proof. ■

Hence, with increasing t the margin converges to the maximum margin as $\mathcal{O}(t^{-1/2})$. In the following we show that even an $\mathcal{O}(t^{-1})$ convergence can be expected. But first we have to prove that after a finite number of iterations $\mathbf{z}_{\min}(t)$ will always be a support vector.

Lemma 2: It exists a $t_0 \in \mathbb{N}_0$ such that $\mathbf{z}_{\min}(t) \in \mathcal{Z}_S$ for $t \geq t_0$.

Proof: Let us assume that there exists no such $t_0 \in \mathbb{N}_0$. We have $\tilde{\Delta} := \frac{1}{2}\min_{\mathbf{z}\in\mathcal{Z}\setminus\mathcal{Z}_S}\mathbf{w}_*^T\mathbf{z} > \Delta^*$. Then for any (arbitrarily large) $t_0 \in \mathbb{N}_0$ there is a $t \geq t_0$ such that $\mathbf{z}_{\min}(t) \notin \mathcal{Z}_S$. Then $\mathbf{w}_*^T\mathbf{z}_{\min}(t) \geq 2\tilde{\Delta}$ and with Equation (4) we obtain

$$\begin{aligned} 2\Delta^* &\geq \mathbf{w}_*^T\mathbf{z}_{\min}(t)\cos\gamma_t + \frac{\mathbf{u}_t^T\mathbf{z}_{\min}(t)}{\|\mathbf{u}_t\|}\sin\gamma_t \\ &\geq 2\tilde{\Delta}\cos\gamma_t - 2R\sin\gamma_t \ . \end{aligned}$$

Since $\cos\gamma_t \to 1$ and $\sin\gamma_t \to 0$ for $t\to\infty$, this leads to a contradiction. ■

Lemma 3: There are $\beta_{\mathbf{z}} \in \mathbb{R}_0^+, \mathbf{z}\in\mathcal{Z}$ with $\beta_{\mathbf{z}} > 0$ only for $\mathbf{z}\in\mathcal{Z}_S$ such that

$$\mathbf{w}_* = \sum_{\mathbf{z}\in\mathcal{Z}}\beta_{\mathbf{z}}\mathbf{z} = \sum_{\mathbf{z}\in\mathcal{Z}_S}\beta_{\mathbf{z}}\mathbf{z} \ .$$

Proof: We use Equation (1). For $\mathbf{z} = \mathbf{x}_i - \mathbf{x}_j$, $\mathbf{x}_i \in \mathcal{X}^+, \mathbf{x}_j \in \mathcal{X}^-$ we choose $\beta_{\mathbf{z}} = \frac{\alpha_i\alpha_j}{A}$ with $A = \frac{1}{2}\sum_{i=1}^N\alpha_i$. Since $\alpha_i \in \mathbb{R}_0^+, i = 1,\ldots,N$ and $\alpha_i > 0$ for support vectors only, the same conditions hold for $\beta_{\mathbf{z}}$. Further, we obtain

$$\begin{aligned} \sum_{\mathbf{z}\in\mathcal{Z}}\beta_{\mathbf{z}}\mathbf{z} = \sum_{\mathbf{z}\in\mathcal{Z}_S}\beta_{\mathbf{z}}\mathbf{z} &= \sum_{\mathbf{x}_i\in\mathcal{X}^+}\sum_{\mathbf{x}_j\in\mathcal{X}^-}\frac{\alpha_i\alpha_j}{A}(\mathbf{x}_i - \mathbf{x}_j) \\ &= \frac{\sum_{\mathbf{x}_j\in\mathcal{X}^-}\alpha_j}{A}\sum_{\mathbf{x}_i\in\mathcal{X}^+}\alpha_i\mathbf{x}_i \\ &\quad - \frac{\sum_{\mathbf{x}_i\in\mathcal{X}^+}\alpha_i}{A}\sum_{\mathbf{x}_j\in\mathcal{X}^-}\alpha_j\mathbf{x}_j \\ &= \sum_{\mathbf{x}_i\in\mathcal{X}^+}\alpha_i\mathbf{x}_i - \sum_{\mathbf{x}_j\in\mathcal{X}^-}\alpha_j\mathbf{x}_j \\ &= \mathbf{w}_* \ . \end{aligned}$$

We used $\sum_{\mathbf{x}_j\in\mathcal{X}^+}\alpha_i = \sum_{\mathbf{x}_j\in\mathcal{X}^-}\alpha_j = A$, which can be derived from the constraint $\sum_{i=1}^N y_i\alpha_i = 0$. ■

Theorem 2: $(\Delta^* - \Delta_t)/\Delta^*$ converges to zero as $\mathcal{O}(t^{-1})$. This $\mathcal{O}(t^{-1})$ convergence is tight.

Proof: We prove Theorem 2 by proving that $\|\mathbf{u}_t\|$ remains bounded. From Equation (3) we can discern that in this case we obtain an $\mathcal{O}(t^{-1})$ bound for the angle γ_t and, hence, an $\mathcal{O}(t^{-1})$ convergence of the margin Δ_t to the maximum margin Δ^*.

We have $\mathbf{u}_{t+1} = \mathbf{u}_t + \mathbf{s}(\mathbf{z}_{\min}(t))$. As we know from Lemma 2, there is a $t_0 \in \mathbb{N}_0$ such that $\mathbf{s}(\mathbf{z}_{\min}(t)) \in \mathcal{S}_S$ for $t \geq t_0$. We discriminate two cases:

i) $\mathbf{s} = 0$ for each $\mathbf{s}\in\mathcal{S}_S$. Then $\|\mathbf{u}_t\|$ remains bounded since it does not change anymore for $t \geq t_0$.
ii) $\mathbf{s} \neq 0$ for at least one $\mathbf{s}\in\mathcal{S}_S$. With $\tilde{\mathbf{u}}_t$ we denote the projection of $\mathbf{u}_t$ onto the linear subspace spanned by $\mathcal{S}_S$. Because of Lemma 2, $\|\mathbf{u}_t\|$ remains bounded, if $\|\tilde{\mathbf{u}}_t\|$ remains bounded. As soon as $t \geq t_0$, we have $\tilde{\mathbf{u}}_{t+1} = \tilde{\mathbf{u}}_t + \mathbf{s}(\mathbf{z}_{\min}(t))$ and, hence, for the change of $\|\tilde{\mathbf{u}}_t\|^2$

$$\|\tilde{\mathbf{u}}_{t+1}\|^2 - \|\tilde{\mathbf{u}}_t\|^2 = 2\tilde{\mathbf{u}}_t^T\mathbf{s}(\mathbf{z}_{\min}(t)) + \|\mathbf{s}(\mathbf{z}_{\min}(t))\|^2 \ .$$

We show by contradiction that there is an $\varepsilon > 0$ such that $\tilde{\mathbf{u}}_t^T\mathbf{s}(\mathbf{z}_{\min}(t)) \leq -\varepsilon\|\tilde{\mathbf{u}}_t\|$ for $t \geq t_0$. Then $\|\tilde{\mathbf{u}}_{t+1}\|^2 - \|\tilde{\mathbf{u}}_t\|^2 \leq -2\varepsilon\|\tilde{\mathbf{u}}_t\| + 4R^2$. The negative contribution to the change of $\|\tilde{\mathbf{u}}_t\|^2$ increases with $\|\tilde{\mathbf{u}}_t\|$, which keeps it bounded.

Let $\mathcal{Q} := \{\mathbf{q} \mid \mathbf{q}\in span(\mathcal{S}_S) \text{ and } \|\mathbf{q}\| = 1\}$ be the set of all vectors of unit norm within the linear subspace spanned by $\mathcal{S}_S$. For $t \geq t_0$ we have

$$\begin{aligned}
\mathbf{z}_{\min}(t) &= \arg\min_{\mathbf{z}\in\mathcal{Z}_S} \mathbf{w}_t^T\mathbf{z} \\
&= \arg\min_{\mathbf{z}\in\mathcal{Z}_S} (\cos\gamma_t\|\mathbf{w}_t\|\mathbf{w}_* + \mathbf{u}_t)^T\mathbf{z} \\
&= \arg\min_{\mathbf{z}\in\mathcal{Z}_S} (2\cos\gamma_t\|\mathbf{w}_t\|\Delta^* + \mathbf{u}_t^T\mathbf{z}) \\
&= \arg\min_{\mathbf{z}\in\mathcal{Z}_S} \mathbf{u}_t^T\mathbf{z} \\
&= \arg\min_{\mathbf{z}\in\mathcal{Z}_S} \mathbf{u}_t^T(\mathbf{z} - (\mathbf{w}_*^T\mathbf{z})\mathbf{w}_*) \\
&\qquad \text{since} \qquad \mathbf{u}_t^T\mathbf{w}_* = 0 \\
&= \arg\min_{\mathbf{z}\in\mathcal{Z}_S} \mathbf{u}_t^T\mathbf{s}(\mathbf{z}) \\
&= \arg\min_{\mathbf{z}\in\mathcal{Z}_S} \tilde{\mathbf{u}}_t^T\mathbf{s}(\mathbf{z})\ .
\end{aligned}$$

Hence,

$$\begin{aligned}
\tilde{\mathbf{u}}_t^T\mathbf{z}_{\min}(t) &= \min_{\mathbf{s}\in\mathcal{S}_S} \tilde{\mathbf{u}}_t^T\mathbf{s} \\
&\leq \left(\max_{\mathbf{q}\in\mathcal{Q}}\min_{\mathbf{s}\in\mathcal{S}_S} \mathbf{q}^T\mathbf{s}\right)\|\tilde{\mathbf{u}}_t\| \\
&\leq -\varepsilon\|\tilde{\mathbf{u}}_t\|
\end{aligned}$$

with $\varepsilon := -\max_{\mathbf{q}\in\mathcal{Q}}\min_{\mathbf{s}\in\mathcal{S}_S} \mathbf{q}^T\mathbf{s}$. Let $\mathbf{q}^* := \arg\max_{\mathbf{q}\in\mathcal{Q}}\min_{\mathbf{s}\in\mathcal{S}_S} \mathbf{q}^T\mathbf{s}$. Lemma 3 and $\mathbf{q}^{*T}\mathbf{z} = \mathbf{q}^{*T}\mathbf{s}(\mathbf{z})$ yields

$$\mathbf{q}^{*T}\mathbf{w}_* = 0 = \sum_{\mathbf{z}\in\mathcal{Z}_S} \beta_\mathbf{z}\,\mathbf{q}^{*T}\mathbf{z} = \sum_{\mathbf{z}\in\mathcal{Z}_S} \beta_\mathbf{z}\,\mathbf{q}^{*T}\mathbf{s}(\mathbf{z}) \tag{5}$$

with $\beta_\mathbf{z} > 0\ \forall\mathbf{z} \in \mathcal{Z}_S$. Let us assume that $\max_{\mathbf{q}\in\mathcal{Q}}\min_{\mathbf{s}\in\mathcal{S}_S} \mathbf{q}^T\mathbf{s} = \min_{\mathbf{s}\in\mathcal{S}_S} \mathbf{q}^{*T}\mathbf{s} = -\varepsilon \geq 0$. Then, because of (5), $\mathbf{q}^{*T}\mathbf{s}(\mathbf{z})$ has to be zero for each $\mathbf{z} \in \mathcal{Z}_S$, or, equivalently, $\mathbf{q}^{*T}\mathbf{s} = 0$ for each $\mathbf{s} \in \mathcal{S}_S$. This, however, is not possible, since $\mathbf{q}^* \in \mathcal{Q}$, i.e., $\mathbf{q}^*$ lies within the subspace spanned by $\mathcal{S}_S$ and $\mathbf{q}^* \neq 0$.

The $\mathcal{O}(t^{-1})$ convergence bound for $\tan\gamma_t$ is a tight bound. For example in case of the four training patterns $\mathbf{x}_1 = (1,1)^T$, $\mathbf{x}_2 = (-1,1)^T$, $\mathbf{x}_3 = (1,-1)^T$, and $\mathbf{x}_4 = (-1,-1)^T$ within $\mathbb{R}^D, D = 2$ with class labels $y_1 = +1$, $y_2 = +1$, $y_3 = -1$, and $y_4 = -1$ (in case of $D > 2$ this might be a subspace). It is easy to see that by construction $\mathbf{w}_* = (0,1)^T$ and $\mathbf{w}_t = (\pm 2, 2t)^T$ for $t = 1,3,5,\ldots$ and $\mathbf{w}_t = (0,2t)^T$ for $t = 0,2,4,\ldots$. Then $\tan\gamma_t = \pm 1/t$ for $t = 1,3,5,\ldots$ and $\tan\gamma_t = 0$ for $t = 0,2,4,\ldots$. ■

Remark 1: In Theorem 1 we assumed that the DoubleMinOver algorithm starts with $\mathbf{w}_{t=0} = 0$. However, it is easy to verify that the $\mathcal{O}(t^{-1/2})$ convergence rate is valid also for $\mathbf{w}_{t=0} \neq 0$. But then all the lemmata and theorems we have proven so far are valid for $\mathbf{w}_{t=0} \neq 0$.

In Fig. 1 we demonstrate the learning process on a 2-dimensional toy problem. 50 data points for each class had to be linearly classified with maximum margin. The dividing line is shown for different time steps. Fig. 1 also shows a convergence plot of the angle between $\mathbf{w}$ and $\mathbf{w}_*$. In the double-logarithmic scale the $\mathcal{O}(t^{-1})$ convergence becomes evident.

B. DoubleMinOver with kernels

The vector $\mathbf{w}_t$ which determines the dividing hyperplane is given by

$$\mathbf{w}_t = \sum_{\tau=0}^{t-1}(\mathbf{x}_{\min^+}(\tau) - \mathbf{x}_{\min^-}(\tau)) = \sum_{i=1}^{N} y_i n_i \mathbf{x}_i\ ,$$

where $n_i \in \mathbb{N}_0$ is the number of times $\mathbf{x}_i \in \mathcal{X}$ has been used for training up to step t. Since $\mathbf{w}_t/\|\mathbf{w}_t\| \to \mathbf{w}_*$ for $t \to \infty$, comparison with Equation (1) shows that $n_i/\|\mathbf{w}_t\| \to \alpha_i$ for $t \to \infty$. Since $\sum_{i=1}^N y_i n_i = 0$ at each time step, the constraint $\sum_{i=1}^N y_i\alpha_i = 0$ of the optimization problem's dual formulation is permanently fulfilled during the learning process [1].

The expression which decides the class assignment by being smaller or larger than zero can be written as

$$\mathbf{w}^T\mathbf{x} - b = \sum_{i=1}^{N} y_i n_i \mathbf{x}_i^T\mathbf{x} - b. \tag{6}$$

If the input patterns $\mathbf{x} \in \mathbb{R}^D$ are transformed into another (usually higher dimensional) feature space $\mathbf{\Phi}(\mathbf{x})$ before classification, DoubleMinOver has to work with $\mathbf{\Phi}(\mathbf{x}_i)$ as training patterns. Due to Equation (6), we do not have to do it explicitly. With $K(\mathbf{x}_i,\mathbf{x}) = \mathbf{\Phi}(\mathbf{x}_i)^T\mathbf{\Phi}(\mathbf{x})$ as the kernel which corresponds to the transformation $\mathbf{\Phi}(\mathbf{x})$, the r.h.s. of (6) transforms into

$$\sum_{i=1}^{N} n_i y_i K(\mathbf{x}_i,\mathbf{x}) - b\ .$$

In this kernel formulation the training step of the DoubleMinOver algorithm simply consists of searching for $\mathbf{x}_{\min^+}(t)$ and $\mathbf{x}_{\min^-}(t)$ and increasing their corresponding n_i. In Algorithm 2 the pseudo code is listed.

If the numbers

$$h(\mathbf{x}_i) = y_i\sum_{j=1}^{N} y_j n_j K(\mathbf{x}_j,\mathbf{x}_i) \qquad i = 1,\ldots,N \tag{7}$$

are stored and updated appropriately, also in its kernel formulation the computational effort for an iteration step of DoubleMinOver increases linearly with the number of training patterns N. Instead of performing a computation of the entire sum (7) in each learning step, one can keep these values from the previous step and obtain the new sum by adding only the kernel values of those patterns that were selected in the current

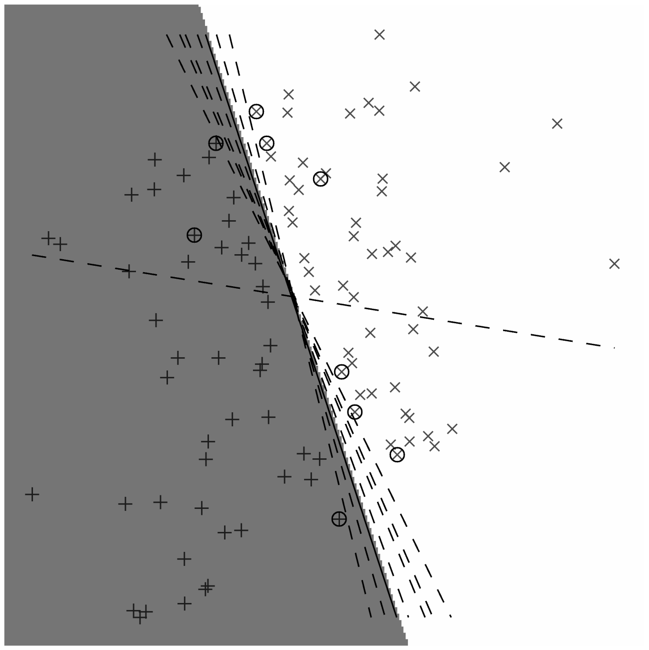
[1] The alternative approach of extending the input vectors by a bias unit $\hat{\mathbf{x}}_i = (\mathbf{x}_i, 1)^T$ together with standard MinOver does not comply with this constraint and, hence, does not yield the correct solution. One obtains

$$\begin{aligned}
\hat{\mathbf{w}}^T\hat{\mathbf{x}} &= \sum_{i=1}^{N} y_i n_i \hat{\mathbf{x}}_i^T\hat{\mathbf{x}} = \sum_{i=1}^{N} y_i n_i(\mathbf{x}_i^T\mathbf{x} + 1) \\
&= \sum_{i=1}^{N} y_i n_i \mathbf{x}_i^T\mathbf{x} + \sum_{i=1}^{N} y_i n_i
\end{aligned}$$

with

$$b = -\sum_{i=1}^{N} y_i n_i \neq 0$$

in general.



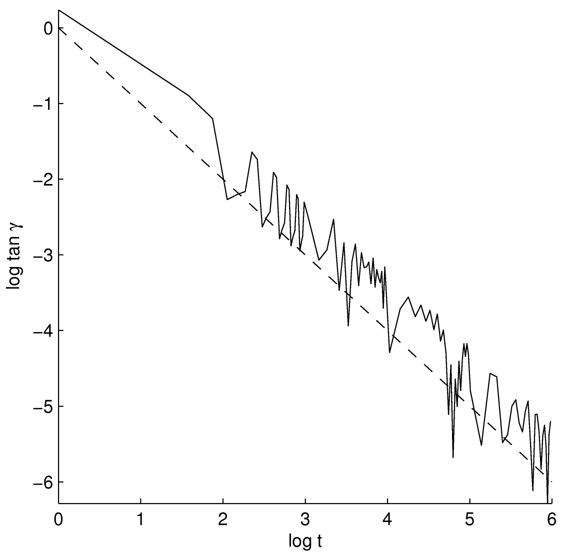


Fig. 1. Left: Learning process of the DoubleMinOver algorithm on an artificial 2-dimensional dataset consisting of 100 training samples. The dashed lines show the separating plane at different time steps. The solid line is the final result of the DoubleMinOver learning algorithm. Training samples that are part of the solution, i.e. that were selected for learning, are marked by a circle. Right: Double logarithmic plot of the convergence of $\tan\gamma$, with γ as the angle between $\mathbf{w}_t$ and $\mathbf{w}_*$. The dashed line demonstrates the $\mathcal{O}(t^{-1})$ convergence.

Algorithm 2 Kernel formulation of the DoubleMinOver algorithm. With $h\left(\mathbf{x}_i\right) = y_i \sum_{j=1}^{N} y_j n_j K(\mathbf{x}_j, \mathbf{x}_i)$:

```
$n_i \leftarrow 0 \ \ \forall i = 1, \ldots, N$
for $t = 0$ to $t_{\max}$ do
    $\mathbf{x}_{\min^+} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^+} h\left(\mathbf{x}_i\right)$
    $\mathbf{x}_{\min^-} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^-} h\left(\mathbf{x}_i\right)$
    $n_{\min^+} \leftarrow n_{\min^+} + 1$
    $n_{\min^-} \leftarrow n_{\min^-} + 1$
end for
$b \leftarrow \frac{1}{2}\left(h\left(\mathbf{x}_{\min^+}\right) - h\left(\mathbf{x}_{\min^-}\right)\right)$.
```

step. Initialized with $h\left(\mathbf{x}_i\right) = 0$, at the end of each iteration the $h\left(\mathbf{x}_i\right), i = 1, \ldots, N$ are updated according to

$$h\left(\mathbf{x}_i\right) \leftarrow h\left(\mathbf{x}_i\right) + y_i \left(K(\mathbf{x}_{\min^+}, \mathbf{x}_i) - K(\mathbf{x}_{\min^-}, \mathbf{x}_i)\right) \ .$$

In Fig. 2 we show the learning result of DoubleMinOver in its kernel formulation with Gaussian kernels, again on a 2-dimensional toy problem. 100 data points for each class had to be classified with maximum margin. Circles indicate those data points for which $n_i > 0$. Note that these data points are not only support vectors.

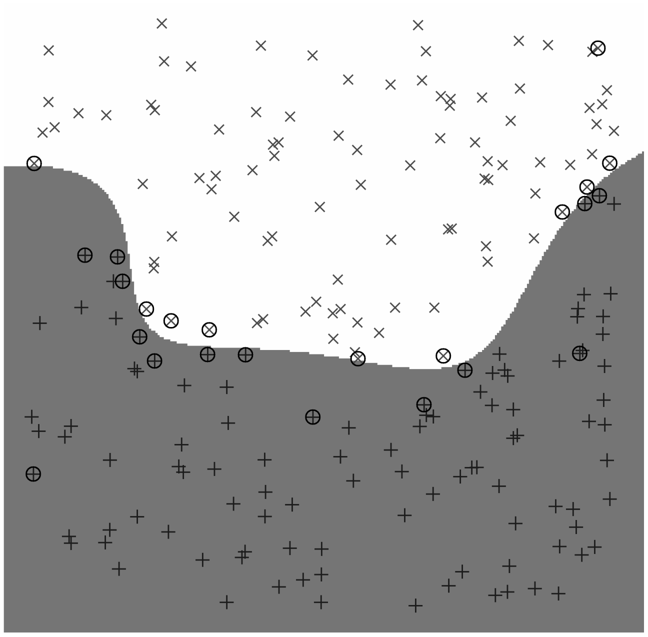


Fig. 2. Result of DoubleMinOver with Gaussian kernels on an artificial 2-dimensional dataset consisting of 200 training samples. The gray area depicts the class boundary. Circles indicate those data points which were used for learning, i.e. for which $n_i > 0$.

III. DoubleMaxMinOver

A drawback of the DoubleMinOver algorithm is that at the end of the training procedure n_i is non-zero not only for the support vectors. Let $\mathcal{V}_t = \{\mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}, n_i > 0\} \subseteq \mathcal{X}$ be the set of training patterns for which n_i is non-zero at time step t. Since the maximum margin hyperplane is determined solely by the set $\mathcal{X}_S \subseteq \mathcal{X}$ of support vectors, at the end of the training $\mathcal{V}_t = \mathcal{X}_S$ should be valid. However, in particular in the beginning of the learning procedure, DoubleMinOver selects training data points for learning which finally turn out not to be support vectors of the maximum margin hyperplane. These data points are superfluous, need not be stored, and might even be detrimental for convergence. It would be desirable to have an algorithm as simple as DoubleMinOver which yields a

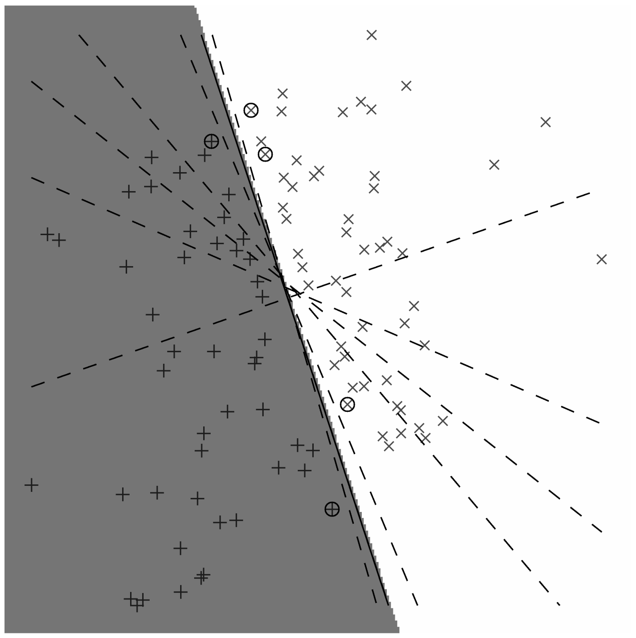
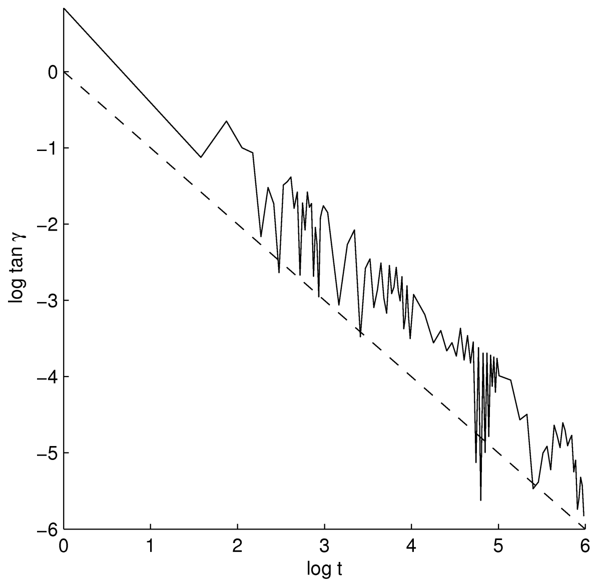



Fig. 3. Left: The same 2-dimensional dataset as in Fig. 1, this time approached with DoubleMaxMinOver. The dashed lines depict the separating plane at different time steps. Again, the solid line is the final training result. Data points for which $n_i > 0$ holds are marked by a circle. It can be seen that only support vectors remain. Right: Double logarithmic plot of the convergence of $\tan\gamma$. As DoubleMinOver, DoubleMaxMinOver converges as $\mathcal{O}(t^{-1})$.

solution for which $\mathcal{V}_t = \mathcal{X}_S$ is valid at the end of the training.

For this purpose, we introduce DoubleMaxMinOver as a simple extension of DoubleMinOver which overcomes this drawback and leads to a solution which is determined by support vectors only. DoubleMaxMinOver not only learns by adding training patterns, but also by selectively forgetting what has been learned before. As in DoubleMinOver, we look for the training patterns $\mathbf{x}_{\min^+}(t) \in \mathcal{X}^+$ and $\mathbf{x}_{\min^-}(t) \in \mathcal{X}^-$ with minimum distance to the current hyperplane given by $\mathbf{w}_t$. Now, at the same time we also look for the patterns $\mathbf{x}_{\max^+}(t) \in \mathcal{V}_t^+$ and $\mathbf{x}_{\max^-}(t) \in \mathcal{V}_t^-$ with maximum distance (overlap) to the current hyperplane. Hence the name DoubleMaxMinOver. $\mathcal{V}_t^+ = \mathcal{V}_t \cap \mathcal{X}^+$ denotes the subset of the current $\mathcal{V}_t$ containing all patterns with class label $y_i = +1$, and $\mathcal{V}_t^- = \mathcal{V}_t \cap \mathcal{X}^-$ the subset containing all patterns with class label $y_i = -1$. In addition to adding $\mathbf{x}_{\min^+}(t)$ and $\mathbf{x}_{\min^-}(t)$ to $\mathbf{w}_t$, the patterns $\mathbf{x}_{\max^+}(t)$ and $\mathbf{x}_{\max^-}(t)$ which were selected for learning in at least one of the preceding learning steps are substracted from $\mathbf{w}_t$. In the kernel representation we not only increase coefficients n_i, but also decrease n_i values. This corresponds to eventually removing elements from $\mathcal{V}_t$.

We will see that this forgetting is advantageous in two aspects: (i) it reduces the number of patterns in the set $\mathcal{V}_t$ which have to be memorized in the kernel representation and (ii) after a finite number of learning steps the learning set $\mathcal{V}_t$ consists of support vectors only. Without kernels, DoubleMaxMinOver works as given in Algorithm 3. Since we subtract patterns, we now add $\mathbf{x}_{\min^+}(t)$ and $\mathbf{x}_{\min^-}(t)$ twice, respectively. This ensures that $\mathbf{w}_t$ is increasing, which is important for fast convergence.

Algorithm 3 DoubleMaxMinOver (for $t = 0$ one chooses $\mathbf{x}_{\min^+} = \mathbf{x}_{\max^+}$ and $\mathbf{x}_{\min^-} = \mathbf{x}_{\max^-}$):

```
$\mathbf{w} \leftarrow 0$
**for** $t = 0$ to $t_{\max}$ **do**
    $\mathbf{x}_{\min^+} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^+} \left(\mathbf{w}^T \mathbf{x}_i\right)$
    $\mathbf{x}_{\min^-} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^-} \left(-\mathbf{w}^T \mathbf{x}_i\right)$
    $\mathbf{x}_{\max^+} \leftarrow \arg \max_{\mathbf{x}_i \in \mathcal{V}_t^+} \left(\mathbf{w}^T \mathbf{x}_i\right)$
    $\mathbf{x}_{\max^-} \leftarrow \arg \max_{\mathbf{x}_i \in \mathcal{V}_t^-} \left(-\mathbf{w}^T \mathbf{x}_i\right)$
    $\mathbf{w} \leftarrow \mathbf{w} + 2(\mathbf{x}_{\min^+} - \mathbf{x}_{\min^-}) - (\mathbf{x}_{\max^+} - \mathbf{x}_{\max^-})$
**end for**
$b \leftarrow \frac{1}{2}\mathbf{w}^T \left(\mathbf{x}_{\min^+} + \mathbf{x}_{\min^-}\right)$
```

A. On the convergence of DoubleMaxMinOver

In the following, we will prove that DoubleMaxMinOver converges (Lemma 4), that it indeed yields the maximum margin solution based solely on support vectors (Theorem 3), and that it converges as $\mathcal{O}(t^{-1})$ (Theorem 4).

Lemma 4: The angle γ_t between $\mathbf{w}_*$ and the $\mathbf{w}_t$ provided by DoubleMaxMinOver converges to zero.

Proof: We introduce

$$\mathcal{Z}_t := \left\{\mathbf{x}^+ - \mathbf{x}^- \mid \mathbf{x}^+ \in \mathcal{V}_t^+, \mathbf{x}^- \in \mathcal{V}_t^-\right\} .$$

With $\mathbf{z}_{\max}(t) := \arg\max_{\mathbf{z} \in \mathcal{Z}_t} \mathbf{w}_t^T \mathbf{z}$ the learning step of DoubleMaxMinOver can be expressed as

$$\mathbf{w}_{t+1} = \mathbf{w}_t + 2\mathbf{z}_{\min}(t) - \mathbf{z}_{\max}(t) .$$

For the angle γ_t we can write

$$\begin{aligned}\cos\gamma_t &= \frac{\mathbf{w}_*\mathbf{w}_t}{\|\mathbf{w}_t\|}\\ &= \frac{\mathbf{w}_*}{\|\mathbf{w}_t\|}\sum_{\tau=0}^{t-1}\left(2\mathbf{z}_{\min}(\tau)-\mathbf{z}_{\max}(\tau)\right)\ .\end{aligned}$$

Now we exploit the fact that for each $\mathbf{z}_{\max}(\tau)$, $0<\tau<t$ there is a $\mathbf{z}_{\min}(\tau')$, $\tau'<\tau$ such that $\mathbf{z}_{\max}(\tau)=\mathbf{z}_{\min}(\tau')$. For $\tau=\tau'=0$ we have $\mathbf{z}_{\max}(\tau)=\mathbf{z}_{\min}(\tau')$ by construction (see header of Algorithm 3). Hence, each $\mathbf{z}_{\max}(\tau)$, $0\le\tau<t$ in the sum can be canceled out on cost of a $\mathbf{z}_{\min}(\tau')$, $0\le\tau'<t$ and we obtain

$$\begin{aligned}\cos\gamma_t &= \frac{\mathbf{w}_*}{\|\mathbf{w}_t\|}\sum_{\tau=0}^{t-1}\mathbf{z}_{\min}(\tau)\\ &\ge \frac{2\Delta^* t}{\|\mathbf{w}_t\|}\ .\end{aligned}\tag{8}$$

Now we show by induction that $\|\mathbf{w}_t\|\le 2\Delta^*t+6R\sqrt{t}$ for all $t\in\mathbb{N}_0$. The case $t=0$ is trivial. For $t\to t+1$ we obtain

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t+2\mathbf{z}_{\min}(t)-\mathbf{z}_{\max}(t)\|^2\\ &= \|\mathbf{w}_t\|^2+2\mathbf{w}_t^T(2\mathbf{z}_{\min}(t)-\mathbf{z}_{\max}(t))\\ &\quad +\|2\mathbf{z}_{\min}(t)-\mathbf{z}_{\max}(t)\|^2\\ &\le \|\mathbf{w}_t\|^2+2\mathbf{w}_t^T\mathbf{z}_{\min}(t)+2(\mathbf{w}_t^T\mathbf{z}_{\min}(t)\\ &\quad -\mathbf{w}_t^T\mathbf{z}_{\max}(t))+(\|2\mathbf{z}_{\min}(t)\|+\|\mathbf{z}_{\max}(t)\|)^2\\ &\le \|\mathbf{w}_t\|^2+4\Delta^*\|\mathbf{w}_t\|+36R^2\ .\end{aligned}$$

In the last step we used $\mathbf{w}_t^T\mathbf{z}_{\min}(t)\le 2\Delta^*\|\mathbf{w}_t\|$, $\mathbf{w}_t^T\mathbf{z}_{\min}(t)-\mathbf{w}_t^T\mathbf{z}_{\max}(t)\le 0$, and $\|\mathbf{z}\|\le 2R$ for each $\mathbf{z}\in\mathcal{Z}$. Now we take the induction hypothesis $\|\mathbf{w}_t\|\le 2\Delta^*t+6R\sqrt{t}$ and can conclude the induction step:

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &\le (2\Delta^*t+6R\sqrt{t})^2+4\Delta^*(2\Delta^*t+6R\sqrt{t})\\ &\quad +36R^2\\ &\le (t^2+2t)(2\Delta^*)^2+24\Delta^*R\sqrt{t}(t+1)\\ &\quad +36R^2(t+1)\\ &\le (2\Delta^*)^2(t+1)^2+24\Delta^*R\sqrt{t+1}(t+1)\\ &\quad +(6R)^2(t+1)\\ &\le \left(2\Delta^*(t+1)+6R\sqrt{t+1}\right)^2\ .\end{aligned}$$

With (8) we then obtain

$$\cos\gamma_t\ge\frac{2\Delta^*t}{2\Delta^*t+6R\sqrt{t}}\ge 1-\frac{3R/\Delta^*}{\sqrt{t}}\ .$$

Hence, $\cos\gamma_t\to 1$ and, therefore, $\gamma_t\to 0$ for $t\to\infty$. ■

Theorem 3: It exists a $t_1\in\mathbb{N}_0$ such that $\mathcal{V}_t=\mathcal{X}_S$ for $t\ge t_1$.

Proof: Since $\lim\limits_{t\to\infty}\gamma_t=0$, Lemma 2 also applies to DoubleMaxMinOver and there is a $t_0\in\mathbb{N}_0$ such that $\mathbf{z}_{\min}(t)\in\mathcal{Z}_S$ for $t\ge t_0$. Hence, the number of non-support vectors within $\mathcal{V}_t$ and the n_i of non-support vectors within $\mathcal{V}_t$ will not increase anymore for $t\ge t_0$. But then there has to be a $t'\in\mathbb{N}_0$ such that $\mathbf{z}_{\max}(t)\in\mathcal{Z}_S$ for $t\ge t'$. Let us assume that there is a non-support vector $\mathbf{z}'$ which remains to be an element of $\mathcal{V}_t$. Then there is a $t''\in\mathbb{N}_0$ such that $\mathbf{w}_t\mathbf{z}'\le\mathbf{w}_t\mathbf{z}_{\max}(t)$ for all $t\ge t''$. This, however, is not possible since $\lim_{t\to\infty}\mathbf{w}_t\mathbf{z}'=\Delta$ and $\lim_{t\to\infty}\mathbf{w}_t\mathbf{z}_{\max}(t)=\Delta^*$ with $\Delta>\Delta^*$. ■

Theorem 4: Let Δ_t be the margin provided by DoubleMaxMinOver after t iteration steps. Its relative deviation $(\Delta^*-\Delta_t)/\Delta^*$ from the maximum margin Δ^* converges to zero at least as $\mathcal{O}(t^{-1})$. This bound is tight.

Proof: We introduce

$$\tilde{\mathcal{Z}}:=\{2\mathbf{z}-\mathbf{z}'\mid\mathbf{z}\in\mathcal{Z}_S,\mathbf{z}'\in\mathcal{Z}_S\}\ .$$

We have $\mathcal{Z}_S\subseteq\tilde{\mathcal{Z}}$. Since $\mathbf{w}_*^T\tilde{\mathbf{z}}=2\Delta^*$ for each $\tilde{\mathbf{z}}\in\tilde{\mathcal{Z}}$, the maximum margin solution for $\tilde{\mathcal{Z}}$ is equivalent to the maximum margin solution for $\mathcal{Z}$.
We introduce $\tilde{\mathbf{z}}_{\min}(t):=\arg\min\limits_{\tilde{\mathbf{z}}\in\tilde{\mathcal{Z}}}\mathbf{w}_t^T\tilde{\mathbf{z}}$. Because of Theorem 3, for $t\ge t_1$ we have $\mathcal{V}_t=\mathcal{X}_S$ and, therefore, $\tilde{\mathbf{z}}_{\min}(t)=2\mathbf{z}_{\min}(t)-\mathbf{z}_{\max}(t)$. The learning step then reads as $\mathbf{w}_{t+1}=\mathbf{w}_t+\tilde{\mathbf{z}}_{\min}(t)$. That means that for $t\ge t_1$ DoubleMaxMinOver on $\mathcal{Z}$ is equivalent to DoubleMinOver on $\tilde{\mathcal{Z}}$ and we can apply Theorem 1. ■

B. *DoubleMaxMinOver with kernels*

Algorithm 4 Kernel formulation of the DoubleMaxMinOver algorithm. With $h(\mathbf{x}_i)=y_i\sum\limits_{j=1}^{N}y_jn_jK(\mathbf{x}_j,\mathbf{x}_i)$:

```
$n_i \leftarrow 0\ \ \forall i=1,\dots,N$
**for** $t=0$ to $t_{max}$ **do**
    $\mathbf{x}_{\min^+} \leftarrow \arg\min_{\mathbf{x}_i\in\mathcal{X}^+} h(\mathbf{x}_i)$
    $\mathbf{x}_{\min^-} \leftarrow \arg\min_{\mathbf{x}_i\in\mathcal{X}^-} h(\mathbf{x}_i)$
    $\mathbf{x}_{\max^+} \leftarrow \arg\max_{\mathbf{x}_i\in\mathcal{V}_t^+} h(\mathbf{x}_i)$
    $\mathbf{x}_{\max^-} \leftarrow \arg\max_{\mathbf{x}_i\in\mathcal{V}_t^-} h(\mathbf{x}_i)$
    $n_{\min^+} \leftarrow n_{\min^+}+2$
    $n_{\min^-} \leftarrow n_{\min^-}+2$
    $n_{\max^+} \leftarrow n_{\max^+}-1$
    $n_{\max^-} \leftarrow n_{\max^-}-1$
**end for**
$b \leftarrow \frac{1}{2}\left(h(\mathbf{x}_{\min^+})-h(\mathbf{x}_{\min^-})\right)$.
```

In its kernel formulation DoubleMaxMinOver looks as given in Algorithm 4. As for DoubleMinOver, $\sum_{i=1}^N n_i=2t$ and the constraint $\sum_{i=1}^N y_in_i=0$ is always fulfilled. As we have shown in Theorem 3, after a finite number of iterations $\mathcal{V}_t$ will consist only of support vectors, i.e., n_i will be non-zero only for support vectors.

In Fig. 3 we show the same example as in Fig. 1, this time the result achieved with DoubleMaxMinOver in its kernel formulation. Again, circles indicate those data points for which $n_i>0$. Note, this time all these data points are support vectors. In Fig. 3, also the $\mathcal{O}(t^{-1})$ convergence is demonstrated.

Fig. 4 shows the same example as in Fig. 2, this time approached with DoubleMaxMinOver instead of DoubleMinOver. As in Fig. 3, we see that at the end only support

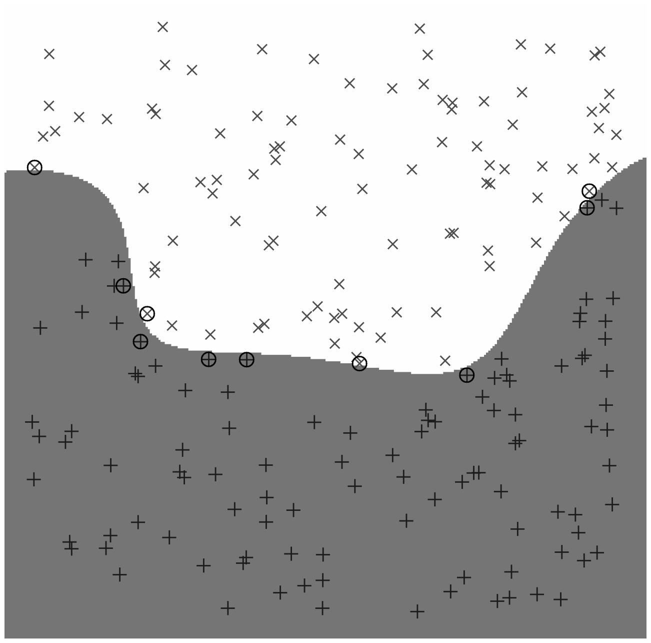


Fig. 4. The same 2-dimensional dataset as in Fig. 2. The gray area indicates the class boundary obtained with DoubleMaxMinOver and Gaussian kernels. Again, data points for which $n_i > 0$ holds are marked by a circle. In contrast to Fig. 2 data points that are not close to the class boundary are removed from the solution and only support vectors remain.

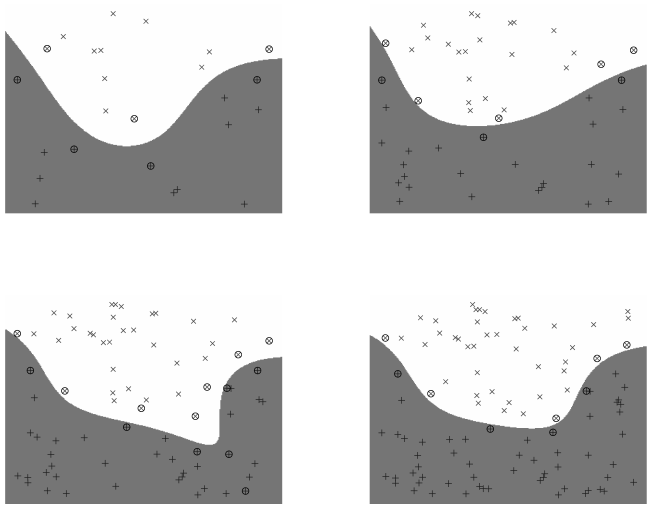


Fig. 5. Incremental learning with DMMO. Top left: Initial state obtained by performing 100 training iterations with 25 training samples. Then successively 25 training samples were added. Each time another 100 training iterations were performed. Initially the n_i of the added training samples are set to zero while keeping the n_i values of the "old" training samples.

vectors determine the solution. In Fig. 5 we applied DoubleMaxMinOver to an incrementally growing training set, i.e, we successively added 25 training samples to the training set until 100 data points were reached. Each time training samples were added we performed another 100 training iterations of DoubleMaxMinOver. The n_i of the new training samples were initialized by zero while keeping the n_i of the "old" training samples. It can be seen how the class boundary adepts to the enlarged training set. Support vectors that become non-support vectors by adding further training samples are forgotten.

IV. SoftDoubleMaxMinOver

So far linear separability of the patterns was required. Since this is often not fulfilled in practice, the concept of a ”soft margin” was introduced in [1], [2]. With a soft margin training patterns are allowed to be misclassified at a certain cost. Instead of minimizing $\|\mathbf{w}\|^2$ under the constraints $y_i \left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1$, a certain error represented by so-called slack variables ξ_i is allowed.

There are two ways of incorporating these slack variables into the objective function. In its original form and usually, the SVM solves the so-called 1-norm soft margin classification problem. In this case it minimizes $\|\mathbf{w}\|^2 + C\sum_{i=1}^N \xi_i$ under the constraints $y_i \left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1 - \xi_i$ with $\xi_i \geq 0$. The scalar parameter C determines the ”hardness” of the margin. The smaller C, the softer the margin. Note, that admitting a soft margin is identical to the relaxation of the Lagrangian, which allows to solve the Quadratic Programming problem even if the training patterns are not linearly separable. The second variation is the 2-norm soft margin problem, where $\|\mathbf{w}\|^2 + C\sum_{i=1}^N \xi_i^2$ is minimized under the constraints $y_i \left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1 - \xi_i$. Again, the scalar parameter C determines the ”hardness” of the margin. Compared to the 1-norm case, slight deviations from the hard margin constraints are punished less, but large values of ξ_i contribute much more to the cost function. Both versions, of course, provide different solutions, but usually there is hardly a difference in classification performance. In the following we will show how DoubleMinOver as well as DoubleMaxMinOver are able to solve the 2-norm soft margin classification problem.

In Cristianini and Shawe-Taylor [22] it is shown, by converting into the dual and eliminating $\boldsymbol{\xi}$, that solving the 2-norm soft margin classification problem within a feature space implicitly defined by a kernel $K(\mathbf{x}, \mathbf{x}')$ is equivalent to solving the hard margin problem within a feature space defined by the kernel $\hat{K}(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + C^{-1}\delta_{ij}$ for each $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, where δ_{ij} denotes the Kronecker delta which is 1 for $i = j$ and 0 otherwise. Within the feature space defined by $\hat{K}(\mathbf{x}, \mathbf{x}')$ the training data are linearly separable by construction. This conversion of the soft margin classification problem into a hard margin problem with modified kernels is not possible in the 1-norm case. This is the reason why we pursue the 2-norm case.

After conversion of the soft margin problem into a linearly separable hard margin problem by modifying the kernel we can directly apply DoubleMinOver as well as DoubleMaxMinOver. As stated already in Section II, DoubleMinOver as well as DoubleMaxMinOver are perceptron-like pattern-by-pattern solvers of the hard-margin SVM-problem. By solving the hard margin problem formulated with the kernel $\hat{K}(\mathbf{x}, \mathbf{x}')$ we solve the 2-norm soft margin classification problem with the kernel $K(\mathbf{x}, \mathbf{x}')$. With

$$\hat{h}\left(\mathbf{x}_i\right) = y_i \sum_{j=1}^N y_j n_j \left(K(\mathbf{x}_j, \mathbf{x}_i) + \frac{\delta_{ij}}{C}\right) = h\left(\mathbf{x}_i\right) + \frac{n_i}{C}$$

the SoftDoubleMaxMinOver algorithm in its kernel formulation then works as shown in Algorithm 5. Having determined the n_i and b via SoftDoubleMaxMinOver, the class assignment

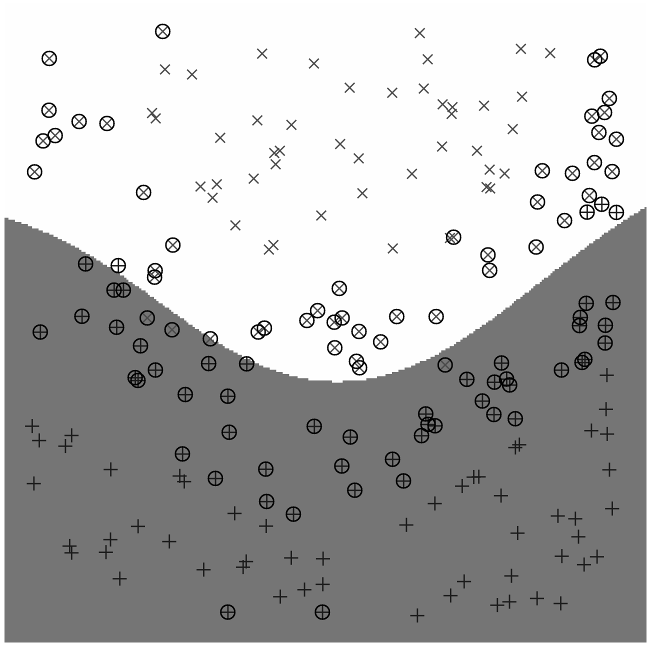


Fig. 6. The same 2-dimensional dataset as in Fig. 2 and Fig. 4, this time approached with SoftDoubleMaxMinOver. Obviously, the resulting class boundary is smoother now. Again, circles indicate the support vectors for which $n_i > 0$.

of a new pattern $\mathbf{x}$ takes place, of course, based on the original kernel. The decision depends on whether

$$\sum_{i=1}^{N} y_i n_i K(\mathbf{x}_i, \mathbf{x}) - b$$

is larger or smaller than zero.

Algorithm 5 Kernel formulation of the SoftDoubleMaxMinOver algorithm. With $h(\mathbf{x}_i) = y_i \sum_{j=1}^{N} y_j n_j K(\mathbf{x}_j, \mathbf{x}_i)$:

```
$n_i \leftarrow 0 \;\; \forall i = 1, \ldots, N$
**for** $t = 0$ to $t_{\max}$ **do**
    $\mathbf{x}_{\min^+} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^+} \left(h(\mathbf{x}_i) + \frac{n_i}{C}\right)$
    $\mathbf{x}_{\min^-} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^-} \left(h(\mathbf{x}_i) + \frac{n_i}{C}\right)$
    $\mathbf{x}_{\max^+} \leftarrow \arg \max_{\mathbf{x}_i \in \mathcal{V}_t^+} \left(h(\mathbf{x}_i) + \frac{n_i}{C}\right)$
    $\mathbf{x}_{\max^-} \leftarrow \arg \max_{\mathbf{x}_i \in \mathcal{V}_t^-} \left(h(\mathbf{x}_i) + \frac{n_i}{C}\right)$
    $n_{\min^+} \leftarrow n_{\min^+} + 2$
    $n_{\min^-} \leftarrow n_{\min^-} + 2$
    $n_{\max^+} \leftarrow n_{\max^+} - 1$
    $n_{\max^-} \leftarrow n_{\max^-} - 1$
**end for**
$b \leftarrow \frac{1}{2}\left(h(\mathbf{x}_{\min^+}) - h(\mathbf{x}_{\min^-})\right) + \frac{1}{2C}\left(n_{\min^+} - n_{\min^-}\right)$
```

Fig. 6 shows the result obtained with SoftDoubleMaxMinOver on the same dataset as in Fig. 2 and Fig. 4. The hardness parameter C was set to $1/2$. The class boundary is smoother now than in Fig. 2 and Fig. 4.

V. Experimental results on benchmark problems

To validate and compare the performance of SoftDoubleMaxMinOver[2] we tested it on a number of common classification benchmark problems. The classification benchmarks stem from the UCI[3], DELVE[4] and STATLOG [23] collection. We compare our results with those reported in the SVM-benchmark repository of the Fraunhofer Institute[5] and results we obtained with the 1-norm-SVM of the OSU-SVM Matlab Toolbox[6] that is based on the SMO-algorithm (in the following called SMO-SVM) [7].

Each result reported in the benchmark repository of the Fraunhofer Institute is based on 100 different partitionings of the respective benchmark data into training and test sets (except for the splice and image benchmark results which stem from 20 partitionings). For classification they used the standard 1-norm-SVM with RBF-kernels. The reported classification result is the average and standard deviation over all 100 realizations. Each partitioning is available from this repository.

Table I lists the average classification errors we obtained with SoftDoubleMaxMinOver and the SMO-SVM on the different benchmark problems. We used the default parameter settings of the OSU-SVM Toolbox. Like the Fraunhofer Institute we used RBF-kernels, and we took their kernel widths γ. The C values in SoftDoubleMaxMinOver and the SMO-SVM where chosen such that the error is minimized. On all benchmarks the simple SoftDoubleMaxMinOver is as fast as and achieves results comparable to those of the SMO-SVM and those reported in the Fraunhofer benchmark repository. Only a few training steps are necessary. On the *ringnorm*, the *image* and the *splice* benchmark both the SMO-SVM as well as SoftDoubleMaxMinOver are worse than the Fraunhofer reference. Since convergence is guaranteed, by either performing more iterations for SoftDoubleMaxMinOver or tweaking the parameters of the SMO-SVM, one can, of course, obtain comparable results for these benchmarks, too. This assumes that on these benchmarks it does not make a significant difference using the 2-norm instead of the 1-norm soft margin.

VI. Conclusions

The well-known MinOver algorithm as a perceptron-like procedure for obtaining maximum margin hyperplanes without a bias was extended to the so-called DoubleMinOver algorithm which yields the maximum margin hyperplane with a bias. In its kernel formulation DoubleMinOver learns by iteratively selecting patterns from the training set. The computational effort increases like $\mathcal{O}(N)$ with the number of training patterns N. We proved an $\mathcal{O}(t^{-1/2})$ convergence, and based on this proof we could show that even an $\mathcal{O}(t^{-1})$ convergence can be expected, with t as the number of iteration steps.

[2]A SoftDoubleMinOver package is available at http://www.inb.uni-luebeck.de/maxminover
[3]UCI Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
[4]DELVE Datasets: http://www.cs.utoronto.ca/~delve/index.html
[5]Benchmark Repository: http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm
[6]OSU SVM Classifier Toolbox: http://sourceforge.net/projects/svm/

			SoftDoubleMaxMinOver		SMO-SVM		Reference
Benchmark	#TR	#TE	Seconds/Iter.	ERR(%)	Seconds	ERR(%)	ERR_{REF}(%)
banana	400	4900	0.030/200	11.6 ± 0.83	0.031	10.4 ± 0.46	12.0 ± 0.66
br-cancer	200	77	0.019/100	27.1 ± 4.96	0.012	28.2 ± 4.62	26.0 ± 4.74
diabetis	468	300	0.060/300	23.3 ± 1.78	0.065	23.1 ± 1.82	24.0 ± 1.73
fl-solar	666	400	0.148/300	32.4 ± 1.80	0.229	32.3 ± 1.82	32.0 ± 1.82
german	700	300	0.142/200	24.1 ± 2.67	0.177	24.0 ± 2.17	24.0 ± 2.07
heart	170	100	0.010/100	15.5 ± 3.22	0.006	15.2 ± 3.21	16.0 ± 3.26
image	1300	1010	0.811/2000	13.1 ± 4.33	0.812	9.8 ± 0.62	3.0 ± 0.60
ringnorm	400	7000	0.030/300	2.6 ± 0.41	0.021	2.5 ± 0.38	1.7 ± 0.12
splice	1000	2175	0.615/500	16.1 ± 0.65	0.654	14.9 ± 0.78	11.0 ± 0.66
titanic	150	2051	0.034/1500	22.4 ± 0.96	0.013	22.3 ± 1.04	22.0 ± 1.02
waveform	400	4600	0.047/300	11.4 ± 0.59	0.045	10.7 ± 0.53	10.0 ± 0.43
thyroid	140	75	0.004/200	4.2 ± 2.40	0.003	4.1 ± 2.42	4.8 ± 2.19
twonorm	400	7000	0.057/200	2.4 ± 0.13	0.033	2.4 ± 0.14	3.0 ± 0.23

#Tr : number of training data, #Te : number of test data

TABLE I

CLASSIFICATION RESULTS OBTAINED WITH SOFTDOUBLEMAXMINOVER ON STANDARD BENCHMARKS. FOR COMPARISON THE RESULTS OBTAINED WITH THE SMO ALGORITHM (USING THE OSU-SVM TOOLBOX) AND THOSE REPORTED IN THE FRAUNHOFER BENCHMARK REPOSITORY (LAST COLUMN) ARE LISTED.

Since DoubleMinOver does not use only support vectors for its solution, we extended DoubleMinOver to DoubleMaxMinOver. DoubleMaxMinOver learns not only by iteratively selecting patterns from the training set, but also by removing patterns which have been used for learning before. We proved that the computational effort and convergence remain the same as for DoubleMinOver, but now the final result is based solely on support vectors.

We showed a way of extending DoubleMaxMinOver to 2-norm soft margins. This is achieved by using DoubleMaxMinOver with an appropriately modified kernel. Hence, SoftDoubleMaxMinOver remains as simple as DoubleMaxMinOver and obeys the same convergence characteristics. With SoftDoubleMaxMinOver which is closely related to the perceptron algorithm a complete pattern-by-pattern SVM is realized. In experiments on common benchmark problems the SoftDoubleMaxMinOver algorithm provided the same classification performance as common state-of-the-art SVM-software.

VII. ACKNOWLEDGEMENT

The authors would like to thank Tom Binder for valuable suggestions and improving the notation. We are grateful to the anonymous reviewers for their helpful comments which lead to significant improvements of the manuscript.

REFERENCES

[1] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, 1995.

[2] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[3] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. *Int.Conf.on Artificial Neural Networks*, pages 53–60, 1995.

[4] E. Osuna, R. Freund, and F. Girosi. Training support vector machines:an application to face detection. *CVPR'97*, pages 130–136, 1997.

[5] Bernhard Schölkopf. *Support Vector Learning*. PhD thesis, Technische Universität Berlin, 1997. Published by: R. Oldenbourg Verlag, Munich.

[6] T.T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. *Proc. 15th International Conference on Machine Learning*, 1998.

[7] J.C. Platt. *Advances in Kernel Methods - Support Vector Learning*, chapter Fast Training of Support Vector Machines using Sequential Minimal Optimization, pages 185–208. MIT Press, 1999.

[8] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. *IEEE-NN*, 11(1):124–136, January 2000.

[9] A. Kowalczyk. *Advances in Large Margin Classifiers*, chapter Maximal margin perceptron, pages 61–100. MIT Press, 2000.

[10] Gert Cauwenberghs and Tomaso Poggio. Incremental and Decremental Support Vector Machine Learning. In *NIPS*, pages 409–415, 2000.

[11] Y. Li and P.M. Long. The Relaxed Online Maximum Margin Algorithm. *Machine Learning*, 46(1-3):361–387, 2002.

[12] S. V. N. Vishwanathan, Alex J. Smola, and M. Narasimha Murty. Simple SVM. In *ICML*, pages 760–767, 2003.

[13] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *J. Mach. Learn. Res.*, 6:363–392, 2005.

[14] W. Krauth and M. Mezard. Learning algorithms with optimal stability in neural networks. *J.Phys.A*, 20:745–752, 1987.

[15] J. K. Anlauf and M. Biehl. The AdaTron: an adaptive perceptron algorithm. *Europhys. Lett.*, 10:687–692, 1989.

[16] W. Kinzel. Statistical mechanics of the perceptron with maximal stability. *Lecture Notes in Physics*, 368:175–188, 1990.

[17] H.D. Navone and T. Downs. Variations on a Kernel-Adatron Theme. *VII Internacional Congress on Information Engineering, Buenos Aires*, 2001.

[18] H. Kim, B. Drake, and H. Park. Adaptive nonlinear discriminant analysis by regularized minimum squared errors. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):603–612, 2006.

[19] H. Kim and H. Park. Incremental and decremental least squares support vector machine and its application to drug design. In *Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB 2004)*, pages 656–657, 2004.

[20] T. Martinetz. MaxMinOver: A simple incremental learning procedure for support vector classification. In *Proc. of the International Joint Conference on Neural Networks*, pages 2065–2070. IEEE Press, 2004.

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

[22] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[23] R. King, C. Feng, and A. Shutherland. STATLOG: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3):259–287, May/June 1995.

Thomas Martinetz is full professor of computer science and the director of the Institute for Neuro- and Bioinformatics at the University of Lübeck. He studied Physics at the TU München and obtained his doctoral degree in Biophysics at the Beckman Institute for Advanced Science and Technology of the University of Illinois at Urbana-Champaign. From 1991 to 1996 he led the project Neural Networks for automation control at the Corporate Research Laboratories of the Siemens AG in Munich. From 1996 to 1999 he was Professor for Neural Computation at the Ruhr-University of Bochum and head of the Center for Neuroinformatics. Thomas Martinetz is Chairman of the German Chapter of the European Neural Network Society.

Kai Labusch studied computer science at the University of Lübeck, where he graduated 2004. He now works as research assistant at the Institute for Neuro- and Bioinformatics of the University of Lübeck, where he pursues a PhD degree. He works in the field of Blind Source Separation, Sparse Coding and Support Vector Machines.

Daniel Schneegaß studied computer science from 2000 to 2005 and graduated from the University of Lübeck as Diplom-Informatiker in 2005. Since 2005 he has been working on his Ph.D. thesis as research associate at the Learning Systems Department of the Siemens AG, Corporate Technology, Information & Communications in Munich. His main research interests include Reinforcement Learning, Kernel Machines and Support Vector Machines, Neural Networks as well as Dynamical Systems and Recurrent Neural Networks.