MaxMinOver: A Simple Incremental Learning Procedure for Support Vector Classification

Thomas Martinetz
Institute for Neuro- and Bioinformatics
University of Lübeck
D-23538 Lübeck
E-mail: martinetz@informatik.uni-luebeck.de

***Abstract*— The well-known MinOver algorithm is a simple modification of the perceptron algorithm and provides the maximum margin classifier in a linearly separable two class classification problem. In its dual formulation selected training patterns which determine the separating hyperplane have to be stored. A drawback of MinOver is that this set of patterns does not consist only of support vectors. With MaxMinOver an extension of MinOver by a simple forgetting procedure is introduced. It is shown that this forgetting not only reduces the number of patterns which have to be stored, but also improves convergence bounds. After a finite number of training steps, the set of stored training patterns will consist only of support vectors. It is shown how this simple and iterative procedure can also be extended to classification with soft margins. The SoftMaxMinOver algorithm exhibits close connections to the ν-Support-Vector-Machine.**

I. Introduction

The Support-Vector-Machine (SVM) [2], [15] has been applied very successfully in many classification and regression tasks (e.g. [7], [11], [13]). Like Neural Networks is has become a standard tool which has to be considered in addition to classical approaches like Bayesian or polynomial classifiers. A major drawback in applications, however, is its complex training procedure. A large Quadratic Programming problem has to be solved, which requires optimization routines from numerical libraries. And if the training set exceeds a certain size, one has to rely on partitioning and on heuristics. Most users do not want or cannot implement these non-trivial training procedures by themselves but have to rely on existing software libraries. These libraries are, in many cases at least, hardly comprehensive or error-free.

This is in contrast to most neural network approaches. Learning in Neural Networks has to be iterative and incremental almost by definition. One does not expect the nervous system to store a whole set of training data first, and then start learning. Learning takes place on-line, pattern-by-pattern. The iterative and incremental nature of learning in Neural Networks usually leads to simple training procedures which can easily be implemented. It is desirable to have similar training procedures also for the SVM.

A number of different approaches to obtain more or less simple incremental training procedures for the SVM have been introduced so far [3], [12], [4], [8]. We want to mention in particular the Kernel-Adatron algorithm by Friess, Christianini, and Campbell [3]. Like the MinOver algorithm by Krauth and Mézard [6] which we will revisit, the Adatron was introduced for constructing synaptic weight matrices of optimal stability in spin-glass models of Neural Networks [1], [5]. 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 constraint gradient descent. The Adatron converges faster, however, the MinOver algorithm is even more simple and applicable to on-line learning. Navone and Downs [10] 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.

In this paper we will give a reformulation of the MinOver algorithm and an alternative convergence proof. The MinOver algorithm is a slight modification of the perceptron learning rule and can hardly be more simple. It converges against the maximum margin hyperplane in a linearly separable classification task, however, the solution is not based on support vectors only. We will introduce the MaxMinOver algorithm as an extension of MinOver. MaxMinOver is as simple as MinOver, however, it provides the maximum margin hyperplane based only on support vectors. We will adapt it not only to classification with kernels, which is trivial, but also to classification with soft margins. The paper presents the basic concepts and analytical results on different aspects of MaxMinOver and SoftMaxMinOver. Experimental studies will be provided in future work.

A. The problem

Given a linearly separable set of patterns $\mathbf{x}_\nu \in \mathbb{R}^D$, $\nu = 1, \dots, N$ with corresponding class labels $y_\nu \in \{-1, 1\}$. 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 offset $b \in \mathbb{R}$. It achieves a separation of the two classes, if

$$y_\nu(\mathbf{w}^T\mathbf{x}_\nu - b) > 0 \qquad \text{for all} \qquad \nu = 1, \dots, N$$

is valid. The margin Δ of this separation is given by

$$\Delta = \min_\nu [y_\nu(\mathbf{w}^T\mathbf{x}_\nu - b)/||\mathbf{w}||].$$

For convenience we introduce $\mathbf{z}_\nu = y_\nu(\mathbf{x}_\nu, -1) \in \mathbb{R}^{D+1}$ and $\mathbf{v} = (\mathbf{w}, b) \in \mathbb{R}^{D+1}$. We obtain $\Delta(\mathbf{v}) = \min_\nu[\mathbf{v}^T\mathbf{z}_\nu/||\mathbf{w}||]$. With

$$d(\mathbf{v}) = \min_\nu[\mathbf{v}^T\mathbf{z}_\nu/||\mathbf{v}||]$$

we introduce the margin of separation of the augmented patterns $(\mathbf{x}_\nu, -1)$ in the $(D+1)$-space. Since the $\mathbf{v}$ which provides the maximum margin d^* in the $(D+1)$-space also provides the maximum margin Δ^* in the D-dimensional subspace of the original patterns $\mathbf{x}_\nu \in \mathbb{R}^D$,

$$\begin{aligned}\mathbf{v}_* = (\mathbf{w}_*, b_*) &= \arg\max_{||\mathbf{v}||=1}[\min_\nu(\mathbf{v}^T\mathbf{z}_\nu)/||\mathbf{w}||] \\ &= \arg\max_{||\mathbf{v}||=1}[\min_\nu(\mathbf{v}^T\mathbf{z}_\nu)]\end{aligned}$$

is valid. Instead of looking for the normalized $\mathbf{v}$ which provides the maximum Δ, we look for the normalized $\mathbf{v}$ which provides the maximum d. Both $\mathbf{v}_*$ are identical. This is illustrated in Fig. 1. Since $||\mathbf{v}_*||^2 = ||\mathbf{w}_*||^2 + b_*^2 = 1$, we obtain Δ^* from d^* and $\mathbf{v}_* = (\mathbf{w}_*, b_*)$ through

$$\Delta^* = \frac{d^*}{||\mathbf{w}_*||} = \frac{d^*}{\sqrt{1-b_*^2}}.$$

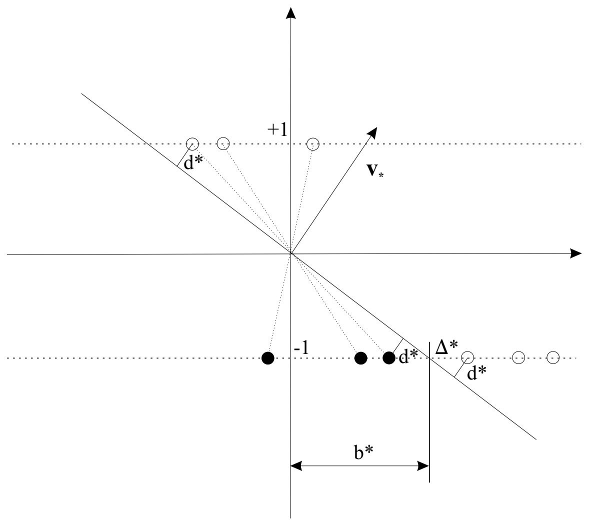

Fig. 1. In this illustration the patterns $\mathbf{x}_\nu$ of the two classes are points in a one-dimensional space. The augmented patterns lie in the plane, shifted along the y-axes to -1. $\mathbf{v}_t$ is a line through the origin. $\mathbf{v}_*$ yields the separation with maximum margin d^*. This separation is equivalent to the maximum margin separation in the original 1D-space with a margin of Δ^*. The white dots on the $+1$-line are the black dots multiplied by their class label $y_\nu = -1$. The white dots altogether are the $\mathbf{z}_\nu$. Obviously, we have to look for the $\mathbf{v}$ which maximizes the minimum $\mathbf{v}^T\mathbf{z}_\nu$.

II. The MinOver algorithm revisited

A simple and iterative algorithm which provides the maximum margin classification in a linearly separable case is the well-known MinOver algorithm introduced by [6] in the context of constructing synaptic weight matrices of optimal stability in spin-glass models of Neural Networks. The MinOver algorithm yields a vector $\mathbf{v}_t$ the direction of which converges against $\mathbf{v}_*$ with increasing number of iterations t. This is valid as long as a full separation, i.e. a $\mathbf{v}_*$ with $\Delta^* > 0$ exists.

The MinOver algorithm works like the perceptron algorithm, with the slight modification that for training always the pattern $\mathbf{z}_\alpha(t)$ out of the training set $\mathcal{T} = \{\mathbf{z}_\nu | \nu = 1, \ldots, M\}$ with the worst, i.e. the minimum residual margin (overlap) $\mathbf{v}^T\mathbf{z}_\nu$ is chosen. Hence, the name MinOver.

Compared to [6] we present a modified formulation of the MinOver algorithm, with the number of desired iterations t_{max} prespecified:

0. Set $t = 0$, choose a t_{max}, and set $\mathbf{v}_{t=0} = 0$.
1. Determine the $\mathbf{z}_\alpha(t)$ out of the training set $\mathcal{T}$ for which the margin $\mathbf{v}_t^T\mathbf{z}$ is minimal.
2. Set $\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{z}_\alpha(t)$.
3. Set $t = t + 1$ and go to 1.) if $t < t_{max}$.

The algorithm can be motivated from gradient ascent. Gradient ascent on the target function $E(\mathbf{v}) = \min_\nu(\mathbf{v}^T\mathbf{z}_\nu)$ leads to the adaptation step $\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{z}_\alpha(t)$. Note, however, that this view is just a motivation and, e.g., neglects the normalization of $\mathbf{v}$.

A. Convergence bounds for MinOver

Krauth and Mézard gave a convergence proof for MinOver [6]. We give a modified proof adapted to our modified version of MinOver. To obtain a convergence bound we look at the angle γ_t between $\mathbf{v}_t$ and $\mathbf{v}_*$. The cosine of this angle is given by

$$\cos\gamma_t = \frac{\mathbf{v}_*^T\mathbf{v}_t}{||\mathbf{v}_t||}.$$

In the Appendix we show that the norm of the vector $\mathbf{v}_t$ is bounded by

$$d^*t \;\leq\; ||\mathbf{v}_t|| \;\leq\; d^*t + R\sqrt{t}, \tag{1}$$

with R as the norm of the augmented pattern with maximum length, i.e., $R = \max_\nu ||\mathbf{z}_\nu||$. Since also

$$\begin{aligned}\mathbf{v}_*^T\mathbf{v}_t &= \mathbf{v}_*^T\sum_{\tau=0}^{t-1}\mathbf{z}_\alpha(\tau) \\ &\geq d^*t\end{aligned}$$

is valid, for the angle γ_t we obtain the bounds

$$\begin{aligned}\cos\gamma_t &\geq \frac{d^*t}{||\mathbf{v}_t||} \qquad (2) \\ &\geq \frac{d^*t}{d^*t + R\sqrt{t}} \\ &\geq 1 - \frac{R/d^*}{\sqrt{t}}.\end{aligned}$$

The angle between the hyperplane provided by this algorithm and the maximum margin hyperplane converges to zero with increasing number of training steps.

B. MinOver in its dual formulation

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

$$\begin{aligned}\mathbf{v}_t &= \sum_{\tau=0}^{t-1} \mathbf{z}_\alpha(\tau) \\ &= \sum_{\mathbf{z}_\nu \in \mathcal{V}} n_\nu(t)\mathbf{z}_\nu\end{aligned}$$

with $\mathcal{V}$ as the set of all patterns which have been used for learning so far. The coefficient $n_\nu(t) \in \mathbb{N}$ denotes the number of times $\mathbf{z}_\nu \in \mathcal{V}$ has been used for training up to step t. $\sum_\mathcal{V} n_\nu(t) = t$ and $V = |\mathcal{V}| \leq t$ is valid.

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

$$\mathbf{v}^T \begin{pmatrix} \mathbf{x} \\ -1 \end{pmatrix} = \sum_{\mathbf{x}_\nu \in \mathcal{V}} n_\nu y_\nu (\mathbf{x}_\nu^T \mathbf{x}) - b \tag{3}$$

with

$$b = \sum_{y_\nu \in \mathcal{V}} n_\nu y_\nu. \tag{4}$$

In the dual formulation the training of the MinOver algorithm consists of either adding the training pattern $\mathbf{z}_\alpha(t)$ as a further data point to the learning set $\mathcal{V}$ or, if $\mathbf{z}_\alpha(t)$ is already element of $\mathcal{V}$, to increase the corresponding n_α by one.

C. MinOver with kernels

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, MinOver has to work with $\mathbf{z}_\nu = y_\nu(\mathbf{\Phi}(\mathbf{x}_\nu), -1)^T$. Due to Equation (3) we do not have to do it explicitly. With $K(\mathbf{x}_\nu, \mathbf{x}) = \mathbf{\Phi}^T(\mathbf{x}_\nu)\mathbf{\Phi}(\mathbf{x})$ as the kernel which corresponds to the transformation $\mathbf{\Phi}(\mathbf{x})$, we obtain

$$\mathbf{v}^T \mathbf{z} = y \left(\sum_{\mathbf{x}_\nu \in \mathcal{V}} n_\nu y_\nu K(\mathbf{x}_\nu, \mathbf{x}) - b \right), \tag{5}$$

with the b of Equation (4).

In its dual formulation the MinOver algorithm is simply an easy procedure of selecting data points out of the training set:

0. Set $t = 0$, choose a t_{max}, and set $\mathcal{V} = \emptyset$.
1. Determine the $\mathbf{x}_\alpha(t)$ out of the training data set for which the margin $\mathbf{v}_t^T \mathbf{z}$ according to Equation (5) is minimal.
2. If $\mathbf{x}_\alpha(t) \notin \mathcal{V}$, add $\mathbf{x}_\alpha(t)$ to $\mathcal{V}$ and assign to it an $n_\alpha = 1$. If $\mathbf{x}_\alpha(t) \in \mathcal{V}$ already, increase its n_α by one.
3. Set $t = t + 1$ and go to 1.) if $t < t_{max}$.

III. The MaxMinOver algorithm

A major drawback of the MinOver algorithm is that the set $\mathcal{V}$ at the end of the training procedure does not only consist of support vectors. Support vectors are training patterns with minimum margin Δ^* with respect to the maximum margin hyperplane given by $\mathbf{v}_*$. This hyperplane is determined solely by the set $\mathcal{S} \subseteq \mathcal{T}$ of these support vectors. Hence, it would be desirable to have an algorithm as simple as MinOver which yields a solution based only on $\mathcal{S}$.

We introduce an extension of the MinOver algorithm which leads to a learning set $\mathcal{V}$ which consists only of support vectors. We call the extended algorithm MaxMinOver. The MaxMinOver algorithm not only learns by adding training patterns but also by selectively forgetting what has been learned before. With each training step t, in addition to adding a new training pattern $\mathbf{z}_\alpha(t)$ to the normal vector $\mathbf{v}_t$, we also look for an old training pattern $\mathbf{z}_\nu \in \mathcal{V}$ which can be subtracted from $\mathbf{v}_t$. In the dual representation we not only add training patterns to $\mathcal{V}$ or increase their coefficients n_ν, but also remove patterns from $\mathcal{V}$ or decrease their n_ν.

We show that this forgetting is advantageous in three aspects: (i) it reduces the number of patterns in the set $\mathcal{V}$ which have to be memorized in the dual representation, (ii) after a finite number of learning steps the learning set $\mathcal{V}$ consists of support vectors only, and (iii) it improves the convergence bound we obtained for MinOver.

As in the MinOver algorithm, we first look for the training pattern $\mathbf{z}_\alpha(t)$ with minimum margin. However, at the same time we also look for the pattern $\mathbf{z}_\omega(t) \in \mathcal{V}$ with maximum margin (overlap). Hence, the name MaxMinOver. If the difference between this maximum margin and the minimum margin exceeds a certain threshold, the pattern $\mathbf{z}_\omega(t)$ which was selected for learning in at least one of the preceding learning steps is now selected for "dememorization".

A. The idea

The MinOver algorithm selects training data points for learning which finally turn out not to be support vectors of the maximum margin hyperplane. The MaxMinOver algorithm is based on the idea that these data points are superfluous, need not to be stored, and might even be detrimental for convergence.

But when should an old training data point be subtracted from $\mathbf{v}_t$ (removed from $\mathcal{V}$)? As we can see from Equation (2), the denominator of the convergence bound for $\cos\gamma_t$ is determined by the number of data points which have been added to $\mathbf{v}_t$ and increases linearly with t. The numerator is given by the norm of $\mathbf{v}_t$. The smaller $||\mathbf{v}_t||$ can be kept while the denominator is increasing with t, the faster convergence. To keep the denominator increasing linearly with t, we add $\mathbf{z}_\alpha(t)$ twice if a $\mathbf{z}_\omega(t)$ is subtracted. And $\mathbf{z}_\omega(t)$ is subtracted if the norm of $\mathbf{v}_t$ stays at least as small as without forgetting. This is the case, if

$$(\mathbf{v}_t + 2\mathbf{z}_\alpha(t) - \mathbf{z}_\omega(t))^2 \leq (\mathbf{v}_t + \mathbf{z}_\alpha(t))^2 \tag{6}$$

or, condition (6) rearranged, if

$$\begin{aligned}\mathbf{v}_t^T \mathbf{z}_\omega(t) - \mathbf{v}_t^T \mathbf{z}_\alpha(t) \quad &\geq \quad \mathbf{z}_\alpha^T(t)(\mathbf{z}_\alpha(t) - \mathbf{z}_\omega(t)) \\ &\quad + \frac{1}{2}(\mathbf{z}_\alpha(t) - \mathbf{z}_\omega(t))^2\end{aligned}$$

is valid. The right hand side of this inequality is always smaller than $4R^2$. Hence, if

$$\mathbf{v}_t^T \mathbf{z}_\omega(t) - \mathbf{v}_t^T \mathbf{z}_\alpha(t) \geq 4R^2,$$

then condition (6) is fulfilled. Each time the difference between the maximum margin $\mathbf{v}_t^T \mathbf{z}_\omega(t)$ and the minimum margin $\mathbf{v}_t^T \mathbf{z}_\alpha(t)$ reaches the threshold $4R^2$, it is advantageous to perform a learning step with forgetting:

$$\mathbf{v}_{t+1} = \mathbf{v}_t + 2\mathbf{z}_\alpha(t) - \mathbf{z}_\omega(t).$$

In the Appendix we show that such a forgetting step improves the convergence bounds for $\cos\gamma_t$.

B. The algorithm

The MaxMinOver algorithm is determined by the following steps:

0. Set $t = 0$, choose a t_{max}, and set $\mathbf{v}_{t=0} = 0$.
1. Determine the $\mathbf{z}_\alpha(t)$ out of the training set for which the margin $\mathbf{v}_t^T \mathbf{z}$ is minimal.
2. Determine the $\mathbf{z}_\omega(t)$ out of the learning set $\mathcal{V}$ for which the margin $\mathbf{v}_t^T \mathbf{z}$ is maximal.
3. If $\mathbf{v}_t^T \mathbf{z}_\omega(t) - \mathbf{v}_t^T \mathbf{z}_\alpha(t) \geq 4R^2$, set $\mathbf{v}_{t+1} = \mathbf{v}_t + 2\mathbf{z}_\alpha(t) - \mathbf{z}_\omega(t)$. Otherwise set $\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{z}_\alpha(t)$.
4. Set $t = t + 1$ and go to 1.) if $t \leq t_{max}$.

In the dual representation, e.g. for realizing classification with kernels, MaxMinOver again is a simple procedure of selecting data points:

0. Set $t = 0$, choose a t_{max}, and set $\mathcal{V} = \emptyset$.
1. Determine the $\mathbf{z}_\alpha(t)$ out of the training data set for which the margin $\mathbf{v}_t^T \mathbf{z}$ according to Equation (5) is minimal.
2. Determine the $\mathbf{z}_\omega(t)$ out of the learning set $\mathcal{V}$ for which the margin $\mathbf{v}_t^T \mathbf{z}$ according to Equation (5) is maximal.
3. $\mathbf{v}_t^T \mathbf{z}_\omega(t) - \mathbf{v}_t^T \mathbf{z}_\alpha(t) < 4R^2$: if $\mathbf{z}_\alpha(t) \notin \mathcal{V}$, add $\mathbf{z}_\alpha(t)$ to $\mathcal{V}$ and assign to it an $n_\alpha = 1$. If $\mathbf{z}_\alpha(t) \in \mathcal{V}$ already, increase its n_α by one.
   $\mathbf{v}_t^T \mathbf{z}_\omega(t) - \mathbf{v}_t^T \mathbf{z}_\alpha(t) \geq 4R^2$: if $\mathbf{z}_\alpha(t) \notin \mathcal{V}$, add $\mathbf{z}_\alpha(t)$ to $\mathcal{V}$ and assign to it an $n_\alpha = 2$. If $\mathbf{z}_\alpha(t) \in \mathcal{V}$ already, increase its n_α by two. Decrease n_ω by one. If $n_\omega = 0$, remove $\mathbf{z}_\omega$ from $\mathcal{V}$.
4. Set $t = t + 1$ and go to 1.) if $t < t_{max}$.

C. The forgetting of all non-support vectors

After a finite number of learning steps t the learning set $\mathcal{V}$ will only consist of support vectors. This can be seen from the following arguments: As soon as $\mathbf{v}_t$ is close enough to $\mathbf{v}_*$, the $\mathbf{z}_\alpha(t)$ will always be support vectors with margin $d(\mathbf{v}_*) = d^*$. A similar result was obtained for decomposition methods for SVMs [9]. For each $\mathbf{z}_\nu \in \mathcal{V}$ which is not a support vector $\mathbf{v}_*^T(\mathbf{z}_\nu - \mathbf{z}_\alpha(t)) > 0$ and, since $\mathbf{v}_t$ is close enough to $\mathbf{v}_*$, also $\mathbf{v}_t^T(\mathbf{z}_\nu - \mathbf{z}_\alpha(t)) > 0$ will be valid. Since the length of $\mathbf{v}_t$ is steadily increasing, according to Equation (1) at least with $d^* t$, after a finite number of steps $\mathbf{v}_t^T(\mathbf{z}_\nu - \mathbf{z}_\alpha(t))$ will reach the threshold $4R^2$ and $\mathbf{z}_\nu$ will be forgotten.

The formal proof looks like follows: We decompose $\mathbf{v}_t$ into

$$\mathbf{v}_t = \cos\gamma_t ||\mathbf{v}_t|| \mathbf{v}_* + \mathbf{u}_t \qquad \text{with} \qquad \mathbf{u}_t^T \mathbf{v}_* = 0$$

and obtain

$$d^* \geq \frac{\mathbf{v}_t^T \mathbf{z}_\alpha(t)}{||\mathbf{v}_t||} = \mathbf{v}_*^T \mathbf{z}_\alpha(t) \cos\gamma_t + \frac{\mathbf{u}_t^T \mathbf{z}_\alpha(t)}{||\mathbf{u}_t||} \sin\gamma_t \,.$$

If $\mathbf{z}_\alpha(t)$ is not a support vector, $\mathbf{v}_*^T \mathbf{z}_\alpha(t) > d^*$ is valid. Since the prefactor of the sinus is bounded and γ_t converges to zero, after a finite number of training steps the right hand side will be larger than d_*. Hence, after a finite number of learning steps the $\mathbf{z}_\alpha(t)$ can only be support vectors.

But then for each $\mathbf{z}_\nu$ which is not a support vector, we obtain

$$\begin{aligned} \frac{\mathbf{v}_t^T}{||\mathbf{v}_t||}(\mathbf{z}_\nu - \mathbf{z}_\alpha(t)) &= \mathbf{v}_*^T(\mathbf{z}_\nu - \mathbf{z}_\alpha(t)) \cos\gamma_t \\ &\quad + \frac{\mathbf{u}_t^T}{||\mathbf{u}_t||}(\mathbf{z}_\nu - \mathbf{z}_\alpha(t)) \sin\gamma_t \,. \end{aligned}$$

Since $\mathbf{v}_*^T(\mathbf{z}_\nu - \mathbf{z}_\alpha(t)) > 0$ and the prefactor of the sinus is bounded again, after a finite t the expression $\mathbf{v}_t^T(\mathbf{z}_\nu - \mathbf{z}_\alpha(t))/||\mathbf{v}_t||$ will always be larger than an $a > 0$. Since $||\mathbf{v}_t||$ is increasing at least with $d^* t$, after a finite number of learning steps $\mathbf{v}_t^T(\mathbf{z}_\nu - \mathbf{z}_\alpha(t)) \geq 4R^2$ will be valid. But this means that each $\mathbf{z}_\nu \in \mathcal{V}$ which is not a support vector will be forgotten after a finite number of learning steps.

IV. SoftMaxMinOver

So far we have studied the problem of separating two classes perfectly with a hyperplane and with maximum margin. This required linear separability of the patterns. However, this is not always the case. For this purpose the concept of a "soft margin" was introduced [2], [15]. With a soft margin training patterns are allowed to be misclassified for a certain cost. With MaxMinOver we can easily realize a soft margin.

Instead of searching for the hyperplane which maximizes the margin of the closest training pattern, we now search for the hyperplane which maximizes the average margin of the $K = \nu M$ closest training patterns (do not mix this ν with the ν we have used as pattern index). Instead of maximizing $E(\mathbf{v}) = \min_\nu(\mathbf{v}^T \mathbf{z}_\nu)$, we now want to maximize

$$\begin{aligned} E(\mathbf{v}) &= \frac{1}{K}\sum_{i=0}^{K-1} \mathbf{v}^T \mathbf{z}_{\alpha_i} \\ &= \mathbf{v}^T \bar{\mathbf{z}}_\alpha \end{aligned}$$

with $\mathbf{z}_{\alpha_0}$ as the training pattern with the smallest margin, $\mathbf{z}_{\alpha_1}$ as the training pattern with the second smallest margin, etc. and $\mathbf{z}_{\alpha_{K-1}}$ as the training pattern with the Kth smallest margin to the hyperplane given by $\mathbf{v}$.

Maximizing the average minimum margin of a set of training patterns corresponds to minimizing the so-called slack variables ξ_ν of these patterns with respect to a certain overall margin. The slack variables were introduced for the soft margin of the Support-Vector-Machine [2], [15]. This is illustrated in Fig. 2. The overall margin, which is given by $1/||\mathbf{w}||$ in the primal formulation of the SVM [15], corresponds to the Kth smallest margin $\mathbf{v}_t^T \mathbf{z}_{\alpha_{K-1}}$. Within the corridor defined by this margin we find $K = \nu M$ training patterns. Maximizing their average distance from the dividing hyperplane is equivalent to minimizing the sum $\sum_{i=1}^{K} \xi_{\alpha_i}$ of their slack variables.

The algorithm works like the one for the hard margin, but instead of adding $\mathbf{z}_\alpha(t)$ we now add $\bar{\mathbf{z}}_\alpha(t)$ to $\mathbf{v}_t$. In the dual formulation all those patterns of the K closest ones which are not yet member of $\mathcal{V}$ are added to this set and their n-values are set to $n = 1$. Those patterns which are already member of $\mathcal{V}$ increase their n by 1.

In an analog way the forgetting is organized. Now, for the forgetting not the pattern with the maximum margin but the K patterns $\mathbf{z}_{\omega_0}, \mathbf{z}_{\omega_1}, \ldots, \mathbf{z}_{\omega_{K-1}}$ within $\mathcal{V}$ with the maximum average margin $\bar{\mathbf{z}}_\omega(t)$ are determined. $\mathbf{z}_{\omega_0}$ denotes the pattern with the largest margin, $\mathbf{z}_{\omega_1}$ the pattern with second largest margin, etc. and $\mathbf{z}_{\omega_{K-1}}$ the pattern within $\mathcal{V}$ with the Kth largest margin. A forgetting takes place if $\mathbf{v}_t^T \bar{\mathbf{z}}_\omega(t) - \mathbf{v}_t^T \bar{\mathbf{z}}_\alpha(t) \geq 4R_K^2$, with R_K as the average norm of the K largest patterns. Clearly, $R_K \leq R = \max_\nu ||\mathbf{z}_\nu||$ is valid. Now, with a forgetting step $\bar{\mathbf{z}}_\omega(t)$ is subtracted from $\mathbf{v}_t$. In the dual formulation the corresponding n-values are reduced by 1 and, eventually, patterns are removed from $\mathcal{V}$, if their n-values reach zero.

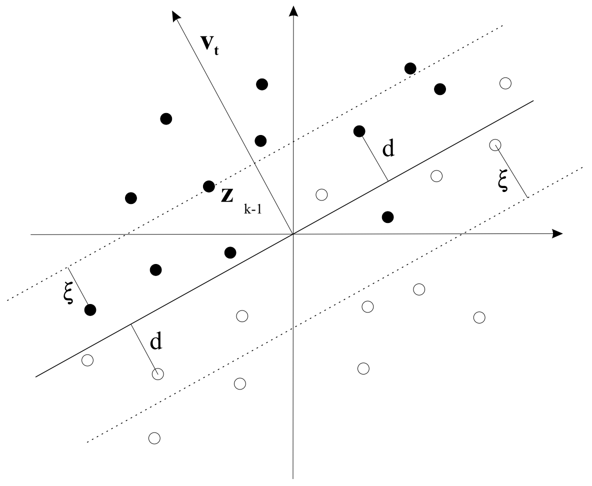

Fig. 2. The margin $\mathbf{v}_t^T \mathbf{z}_{\alpha_{K-1}} / ||\mathbf{v}_t||$ of the data point which is Kth-closest to the hyperplane $\mathbf{v}_t$ determines the corridor within which all data points which are considered for the averaging of SoftMaxMinOver are lying. Minimizing the average slack $\langle \xi \rangle$ is equivalent to maximizing the average margin $\langle d \rangle$.

A. On the convergence of SoftMaxMinOver

SoftMaxMinOver is equivalent to MaxMinOver on a training data set given by all K over M possible patterns $\bar{\mathbf{z}}_\nu$ which result from taking the average of K original patterns $\mathbf{z}_\nu$. The pattern with minimum and maximum margin in this set is obtained by taking the average of the K original patterns with minimum and maximum margin, respectively, as described above. Hence, convergence is guaranteed given a positive maximum margin d_K^* on this set. Since $d_1^* \leq d_2^* \leq \ldots d_M^*$ is valid, and, since for $K = M$ the training data set consists of one $\bar{\mathbf{z}}_\nu$ only with $d_M^* = ||\bar{\mathbf{z}}_{\nu=1}|| > 0$, there exists a smallest K for which SoftMaxMinOver converges. As a lower bound for $\cos\gamma_t$ we obtain

$$\cos\gamma_t \geq 1 - \frac{R_K/d_K^*}{\sqrt{t}}.$$

Convergence speed increases with increasing K, since d_K^* is increasing and R_K is decreasing.

B. Some remarks on SoftMaxMinOver

Usually one does not know which K one has to choose to obtain convergence. However, if appropriate kernels are used, e.g. Gaussians, convergence is given even for $K = 1$. This is true for all kernels which correspond to transformations $\mathbf{\Phi}(\mathbf{x})$ into infinite dimensional feature spaces. In these feature spaces all pattern sets are linearly separable.

With ν we denoted the fraction K/M of the training patterns which are considered for averaging. There is a close connection to the ν-Support-Vector-Machine. As for the ν-SVM [14], obviously also for SoftMaxMinOver

(i) ν is an upper bound on the fraction of margin errors (and hence also on the fraction of training errors). A margin error is given if a training data point lies within the margin corridor, i.e. in case of SoftMaxMinOver if $\mathbf{v}_t^T \mathbf{z}_\nu < \mathbf{v}_t^T \mathbf{z}_{\alpha_{K-1}}$ is valid.

(ii) ν is a lower bound on the fraction of support vectors, i.e. $K = \nu M \leq |\mathcal{V}|$ is valid.

As for the ν-SVM, also for SoftMaxMinOver there is a $\nu_{max} = 2\min(M_+, M_-)/M$ up to which ν-values make sense only. M_+ and M_- denote the number of patterns with class labels $y_\nu = +1$ and $y_\nu = -1$, respectively. In contrast to the ν-SVM, SoftMaxMinOver guarantees convergence even for $\nu = 1$. However, the solutions might not be reasonable anymore. This requires further analysis in future work.

V. Conclusions

The well-known MinOver algorithm as a simple and incremental procedure for obtaining maximum margin hyperplanes can be extended to the so-called MaxMinOver algorithm which, in contrast to MinOver, yields the maximum margin hyperplane based only on its support vectors. This is achieved by an additional forgetting procedure. In its dual formulation MaxMinOver learns by simply iteratively selecting patterns from the training set and, eventually, removing some of them again later on. The computational effort increases linearly with the number of training patterns.

We have given an alternative proof and bounds for the convergence of MinOver. Within this framework we have shown that the additional forgetting procedure improves these convergence bounds. We have given a proof that after a finite number of learning steps MaxMinOver provides a separating hyperplane which is determined only by support vectors of the maximum margin hyperplane.

We have shown a way of extending MaxMinOver to soft margins. Instead of taking the patterns of minimum and maximum margin for learning and forgetting, the average of the $K = \nu M$ patterns with minimum and maximum margin is taken. We have analyzed the convergence of this SoftMaxMinOver algorithm, and we have shown that convergence speed increases with increasing ν. However, also the computational effort for each learning step increases with ν. For a fixed ν the SoftMaxMinOver algorithm scales like $\mathcal{O}(N^2)$ with the number N of training patterns.

Perhaps most interestingly, the simple and incremental SoftMaxMinOver algorithm exhibits close connections to the

ν-Support Vector Machine. A deeper analysis of these connections will be the subject of future work. In future work also experimental results and comparisons with existing approaches have to be presented, in particular with decomposition methods like SMO [12].

APPENDIX

In this Appendix we give upper and lower bounds for the norm of $\mathbf{v}_t$, for MinOver as well as for MaxMinOver. We show that for MaxMinOver one can expect a faster convergence.

A. Bounds for MinOver

Starting the MINOVER algorithm with $\mathbf{v}_{t=0} = 0$, we obtain as a lower bound for the norm of the normal vector

$$||\mathbf{v}_t|| \;\geq\; \mathbf{v}_*^T \mathbf{v}_t \;=\; \mathbf{v}_*^T \sum_{\tau=0}^{t-1} \mathbf{z}_\alpha(\tau) \;\geq\; d^* t.$$

An upper bound is given by

$$||\mathbf{v}_t|| \;\leq\; d^*\, t + R\,\sqrt{t}.$$

This can easily be shown by induction: The case $t = 0$ is trivial. For $t \rightarrow t+1$ we obtain

$$\begin{aligned}
\mathbf{v}_{t+1}^2 &= \mathbf{v}_t^2 + 2\mathbf{v}_t^T \mathbf{z}_\alpha(t) + \mathbf{z}_\alpha^2(t) \\
&\leq \mathbf{v}_t^2 + 2d^* ||\mathbf{v}_t|| + \mathbf{z}_\alpha^2(t) \\
&\leq (d^* t + R\sqrt{t})^2 + 2d^*(d^* t + R\sqrt{t}) + R^2 \\
&\leq (t^2 + 2t)(d^*)^2 + 2d^* R\sqrt{t}(t+1) + R^2(t+1) \\
&\leq (d^*)^2(t+1)^2 + 2d^* R\sqrt{t+1}(t+1) + R^2(t+1) \\
&\leq \left(d^*(t+1) + R\sqrt{t+1}\right)^2.
\end{aligned}$$

We have used $\mathbf{v}_t^T \mathbf{z}_\alpha(t) \leq d^* ||\mathbf{v}_t||$.

B. Bounds for MaxMinOver

Starting the MaxMinOver algorithm with $\mathbf{v}_{t=0} = 0$, we obtain the same lower bound for the norm of the normal vector as for MinOver.

For a finite number of learning steps t_{max}, we can assume that there is a $\epsilon > 0$ such that for each learning step with forgetting $\mathbf{v}_t^T \mathbf{z}_\omega(t) - \mathbf{v}_t^T \mathbf{z}_\alpha(t) \geq 4R^2(1+\epsilon)$ is valid. Then one obtains as an upper bound

$$\mathbf{v}_t^2 \;\leq\; (d^*\, t + R\,\sqrt{t})^2 - 8\epsilon\, t_f\, R^2,$$

with t_f as the number of learning steps which included a forgetting.

Again proof by induction: The case $t = 0$ is trivial. For $t \rightarrow t+1$ and $t_f \rightarrow t_f$ (learning without forgetting), we can follow the line of the proof for the upper bound with MINOVER and obtain

$$\begin{aligned}
\mathbf{v}_{t+1}^2 &= (\mathbf{v}_t + \mathbf{z}_\alpha(t))^2 \\
&\leq (d^* t + R\sqrt{t})^2 - 8\epsilon t_f R^2 \\
&\quad + 2d^* \sqrt{(d^* t + R\sqrt{t})^2 - 8\epsilon t_f R^2} + R^2 \\
&\leq (t^2 + 2t)(d^*)^2 + 2d^* R\sqrt{t}(t+1) \\
&\quad + R^2(t+1) - 8\epsilon t_f R^2 \\
&\leq \left(d^*(t+1) + R\sqrt{t+1}\right)^2 - 8\epsilon t_f R^2.
\end{aligned}$$

For $t \rightarrow t+1$ and $t_f \rightarrow t_f + 1$ (learning with forgetting) we obtain

$$\begin{aligned}
\mathbf{v}_{t+1}^2 &= (\mathbf{v}_t + \mathbf{z}_\alpha(t) + \mathbf{z}_\alpha(t) - \mathbf{z}_\omega(t))^2 \\
&\leq (\mathbf{v}_t + \mathbf{z}_\alpha(t))^2 + 2\mathbf{v}_t^T(\mathbf{z}_\alpha(t) - \mathbf{z}_\omega(t)) + 8R^2.
\end{aligned}$$

For the first term we can use the upper bound we have obtained above. The second term is not larger than $-8R^2(1+\epsilon)$ by construction, which finally leads to

$$\mathbf{v}_{t+1}^2 \leq \left(d^*(t+1) + R\sqrt{t+1}\right)^2 - 8\epsilon(t_f + 1)R^2.$$

Compared with MinOver we obtain a reduction of the upper bound with each forgetting. Through Equation (2) this improves the bound for the convergence of γ_t.

ACKNOWLEDGMENT

The author would like to thank Kai Labusch and Amir Madany for their help preparing the manuscript.

REFERENCES

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

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] Y. Li and P.M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387, 2002.

[9] Ch.-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 13(5):1045–1052, 2002.

[10] H.D. Navone and T. Downs. Variations on a kernel-adatron theme. *VII Internacional Congress on Information Engineering, Buenos Aires*, 2001.

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

[12] 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.

[13] B. Schölkopf. Support vector learning, 1997.

[14] B. Schölkopf, A. J. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural Computation*, 12:1083–1121, 2000.

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