MaxMinOver Regression: A Simple Incremental Approach for Support Vector Function Approximation

Daniel Schneegaß[1,2], Kai Labusch[1], and Thomas Martinetz[1]

[1] Institute for Neuro- and Bioinformatics
University at Lübeck, D-23538 Lübeck, Germany
martinetz@informatik.uni-luebeck.de
[2] Information & Communications, Learning Systems
Siemens AG, Corporate Technology, D-81739 Munich, Germany
daniel.schneegass.ext@siemens.com

Abstract. The well-known MinOver algorithm is a simple modification of the perceptron algorithm and provides the maximum margin classifier without a bias in linearly separable two class classification problems. In [1] and [2] we presented DoubleMinOver and MaxMinOver as extensions of MinOver which provide the maximal margin solution in the primal and the Support Vector solution in the dual formulation by dememorising non Support Vectors. These two approaches were augmented to soft margins based on the ν-SVM and the C2-SVM. We extended the last approach to SoftDoubleMaxMinOver [3] and finally this method leads to a Support Vector regression algorithm which is as efficient and its implementation as simple as the C2-SoftDoubleMaxMinOver classification algorithm.

1 Introduction

The Support-Vector-Machine (SVM) [4], [5] is a very efficient, universal and powerful tool for classification and regression tasks (e.g. [6], [7], [8]). 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, which requires sophisticated numerical optimisation 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, in some cases at least, error-free. This is in contrast to most Neural Network approaches where learning has to be simple and incremental almost by definition. 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.

Several approaches for obtaining more or less simple incremental training procedures for Support Vector classification and regression have been introduced so far [9], [10], [11], [12]. We want to mention in particular the Kernel-Adatron by Friess, Cristianini, and Campbell [9] and the Sequential-Minimal-Optimisation algorithm (SMO) by Platt [10]. This is the most widespread iterative training procedure for SVMs. It is fast and robust, but it is still not yet of a pattern-by-pattern nature and well suited as a

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 150–158, 2006.

starting point for an online learning scheme. It is not yet as easy to implement as one is used from Neural Network approaches.

According to this goal in [2] and [1] we had revisited and extended the MinOver algorithm. The so-called DoubleMaxMinOver [3] method which we briefly revisit in this paper as the combination of DoubleMinOver and MaxMinOver converges provable to the Support Vector solution in the dual representation for classification tasks. Moreover, the angle γ_t between the correct solution $\mathbf{w}^*$ and the interim solution $\mathbf{w}_t$ after t steps is bounded by $O(t^{-1})$.

Furthermore we introduced a soft margin approach based on the C2-SVM error model. In a straightforward way this model can be adapted for regression [13]. We consequently show that we similarly obtain our regression algorithm as an extension of DoubleMaxMinOver for classification as well. We give a proof for its convergence properties and consider benchmark results.

2 The Support Vector Machine

The SVM represents a linear classifier or function approximator, respectively. For given sets of input vectors $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ and output vectors $Y = \{y_1, \ldots, y_l\}$ the SV solution for the inference of $f(\mathbf{x}) = y$ is always the best one in the sense that in classification the margin between the separating hyperplane and the two classes is largest and in regression the obtained curve is the flattest one. In both cases one only needs a few input vectors, the so-called Support Vectors, to describe the solution. This property is very important. The lower the number of Support Vectors, the better the expected generalisation capability [14].

The goal is to solve the optimisation problem

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} = \min$$
$$\forall i \in \{1, \ldots, l\} : y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1$$

in classification, which is apparently equivalent to the maximisation of the margin holding $\|\mathbf{w}\|$, and

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} = \min$$
$$\forall i \in \{1, \ldots, l\} : |\mathbf{w}^T\mathbf{x}_i - b - y_i| \leq \epsilon.$$

in regression tasks. Note that classification and regression are geometrically analogous in the following sense. While in classification the Support Vectors are the ones lying on the margin, that is having smallest distance to the separating hyperplane, in regression the Support Vectors lie on the edges of the ϵ-tube around the function values. In classification all other vectors are located more or less far away from the margin. In regression they lie within the ϵ-tube. Hence SV classification can be seen as the perspective from inside to outside the hyperplane area while SV regression is the other way around.

An important advantage of SVMs is furthermore that the classifier or function approximator $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} - b$ can be formulated in terms of the observations, i.e. as $f(\mathbf{x}) = \left(\sum_{i=1}^{l} \alpha_i \mathbf{x}_i^T\right)\mathbf{x} - b$. Therefore, it is possible to replace the inner product $\mathbf{x}^T\mathbf{z}$ by a positive definite Kernel $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$ which leads to the implicit use of arbitrary feature spaces, whose definition has not necessarily to be known, making profoundly non-linear problems linear.

3 The C2-SoftDoubleMaxMinOver Algorithm for Classification

The DoubleMinOver algorithm [1] as the first extension of MinOver is a simple maximal margin learning machine. Starting from any interim solution $\mathbf{w}_t$ the method selects two vectors each of both classes having the smallest margin to the hyperplane defined by $V = \{\mathbf{v} | \mathbf{w}^T\mathbf{v} - b = 0\}$, that is $i_{a,\min} = \arg\min_{i,y_i=a} af(\mathbf{x}_i)$, respectively. Afterwards the Lagrange-coefficients α_i will be increased by 1 to change the direction of the weight vector given by $\forall \mathbf{x} : f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x})$ towards these two input vectors and increase their margins. By induction it can be seen that $\mathbf{w}_t$ indeed tends to $\mathbf{w}^*$ while $t \to \infty$. It has been shown [15] that the angle γ_t between the correct solution $\mathbf{w}^*$ and the interim solution $\mathbf{w}_t$ after t steps is bounded by $O(t^{-1})$. Furthermore the time complexity per iteration is $O(l)$, where l is the number of input vectors.

In order to achieve the dual SV solution, not to decrease the convergence speed and to hold the optimisation constraints, in DoubleMaxMinOver the vectors with the largest margin will be decreased, if it is known that they cannot be Support Vectors while the ones with smallest margin will be increased twice. In addition we introduced a soft margin approach working with an extended kernel $K'(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{i,j}}{C}$. It can be seen that indeed the data remains linearly separable by construction [13] within this extended feature space. Later we will see that all these statements can straightforwardly be adopted for our regression approach.

The combination of this soft margin, the DoubleMinOver, and the MaxMinOver approaches finally leads to the C2-SoftDoubleMaxMinOver [1,2,3] method, which is as fast as the LibSVM toolbox on standard benchmarks and performs comparable results.

4 The MaxMinOver Regression Approach

More general than the classification task is the one of regression. Now the goal is to approximate a function $g(\mathbf{x})$ by using

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) - b.$$

This can be interpreted as linear regression within an appropriate feature space, again defined by the Kernel K. To get a scope for the minimisation of $\|\mathbf{w}\|$ the user defines

an ϵ-tube around the real function values $y_i = g(\mathbf{x}_i)$. As in our soft margin classification approach we consequently use the C2-SVM model for regression. If $C \to \infty$ the Support Vector solution is in a way the simplest or the flattest one while making no more error than ϵ. The Support Vectors are the vectors with Lagrange-coefficients $\alpha_i \neq 0$ lying on the edges of the ϵ-tube. The introduction of a finite C is not only necessary as regularisation parameter, but also if it cannot be guaranteed that a regression with an error of at most ϵ is achievable at all. Fortunately the geometrical interpretation is similar to the one in the C2-SVM classification, so it is not difficult to see that also in the regression case an unequivocal Support Vector solution exists, which can be interpreted

Algorithm 1. The MaxMinOver Regression Algorithm

Require: given set of normed input vectors $X = \{(\mathbf{x}_i, y_i), i \in \{1, \ldots, l\}\}$ with normed function values and parameters C and ϵ

Ensure: calculates the minimal hyperplane regressing the input data given in dual representation

set $\forall i : \alpha_i \leftarrow 0$, $t \leftarrow 0$, $f \leftarrow \left(\mathbf{x}_i \to \sum_{j=1}^{l} \alpha_j K(\mathbf{x}_j, \mathbf{x}_i) + \frac{\alpha_j}{C}\right)$

$R \leftarrow \sqrt{\max_{i,j,y_i=1,y_j=-1} \left(K(\mathbf{x}_i, \mathbf{x}_i) - 2K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_j, \mathbf{x}_j)\right)}$

while the desired precision is not reached **do**

set $t \leftarrow t + 1$

find $i_{\min} = \arg\min_i \left(f(\mathbf{x}_i) - y_i\right)$

find $i_{\max} = \arg\max_i \left(f(\mathbf{x}_i) - y_i\right)$

find $i_{minNSV} = \arg\max_{i,\alpha_i>0} \left(f(\mathbf{x}_i) - y_i\right)$

find $i_{maxNSV} = \arg\min_{i,\alpha_i<0} \left(f(\mathbf{x}_i) - y_i\right)$

if $\left(f(\mathbf{x}_{i_{\max}}) - y_{i_{\max}}\right) - \left(f(\mathbf{x}_{i_{\min}}) - y_{i_{\min}}\right) > 2\epsilon$ **then**

if $\left(f(\mathbf{x}_{i_{minNSV}}) - y_{i_{minNSV}}\right) - \left(f(\mathbf{x}_{i_{\min}}) - y_{i_{\min}}\right) > \frac{4R^2+16(2+3\epsilon^2+4\epsilon)}{t}$ **then**

set $\alpha_{i_{\min}} \leftarrow \alpha_{i_{\min}} + \frac{1}{t} + \min\left(\frac{1}{t}, \alpha_{i_{minNSV}}\right)$

set $\alpha_{i_{minNSV}} \leftarrow \alpha_{i_{minNSV}} - \min\left(\frac{1}{t}, \alpha_{i_{minNSV}}\right)$

else

set $\alpha_{i_{\min}} \leftarrow \alpha_{i_{\min}} + \frac{1}{t}$

end if

if $\left(f(\mathbf{x}_{i_{\max}}) - y_{i_{\max}}\right) - \left(f(\mathbf{x}_{i_{maxNSV}}) - y_{i_{maxNSV}}\right) > \frac{4R^2+16(2+3\epsilon^2+4\epsilon)}{t}$ **then**

set $\alpha_{i_{\max}} \leftarrow \alpha_{i_{\max}} - \frac{1}{t} - \min\left(\frac{1}{t}, -\alpha_{i_{maxNSV}}\right)$

set $\alpha_{i_{maxNSV}} \leftarrow \alpha_{i_{maxNSV}} + \min\left(\frac{1}{t}, -\alpha_{i_{maxNSV}}\right)$

else

set $\alpha_{i_{\max}} \leftarrow \alpha_{i_{\max}} - \frac{1}{t}$

end if

else

if $\left(f(\mathbf{x}_{i_{\min}}) - y_{i_{\min}}\right) - \left(f(\mathbf{x}_{i_{minNSV}}) - y_{i_{minNSV}}\right) > \frac{4R^2+16(2+3\epsilon^2+4\epsilon)}{t}$ **then**

set $\alpha_{i_{minNSV}} \leftarrow \alpha_{i_{minNSV}} - \min\left(\frac{1}{t}, \alpha_{i_{minNSV}}\right)$

end if

if $\left(f(\mathbf{x}_{i_{\max}}) - y_{i_{\max}}\right) - \left(f(\mathbf{x}_{i_{maxNSV}}) - y_{i_{maxNSV}}\right) > \frac{4R^2+16(2+3\epsilon^2+4\epsilon)}{t}$ **then**

set $\alpha_{i_{maxNSV}} \leftarrow \alpha_{i_{maxNSV}} + \min\left(\frac{1}{t}, -\alpha_{i_{maxNSV}}\right)$

end if

end if

end while

set $b \leftarrow \frac{1}{2}\left(\left(f(\mathbf{x}_{i_{\min}}) - y_{i_{\min}}\right) + \left(f(\mathbf{x}_{i_{\max}}) - y_{i_{\max}}\right)\right)$

as the solution with at most ϵ error within the extended feature space defined by $K'(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}, \mathbf{z}) + \frac{\delta(\mathbf{x},\mathbf{z})}{C}$. This is as reasonable as the statement that the preconditioned linear equation system $\left(K + \frac{1}{C} E_l\right) \alpha = y$, which is used in kernalized Ridge Regression [13], has a solution for all but finitely many $C < \infty$, even if the bias $b = 0$ and $\epsilon = 0$.

As a first approach consider the direct adaptation of the DoubleMinOver algorithm. As in the classification problem, where we choose one input vector each of both classes with the smallest margin, we now choose the two input vectors making the largest positive and negative regression error, respectively. Apparently, using an appropriate step width, the algorithm converges to any feasible solution, that is all input vectors lie within the ϵ-tube and the above constraint holds true by construction in each iteration.

Still if the Lagrange-coefficient of any of the input vectors will be included faulty as potential Support Vectors, we need an instrumentation to turn back this decision, if they are indeed non Support Vectors. Once again as in the classification task, where we choose the input vectors each of both classes with the largest margin, we now choose the two vectors lying furthest away from its edge of the ϵ-tube and dememorise them by controlling the Lagrange-coefficients back without loosing convergence speed (see algorithm 1). Consequently this is the MaxMinOver Regression algorithm. The difficult looking dememorisation criterion will be explained in the next section.

4.1 On the Convergence of MaxMinOver Regression

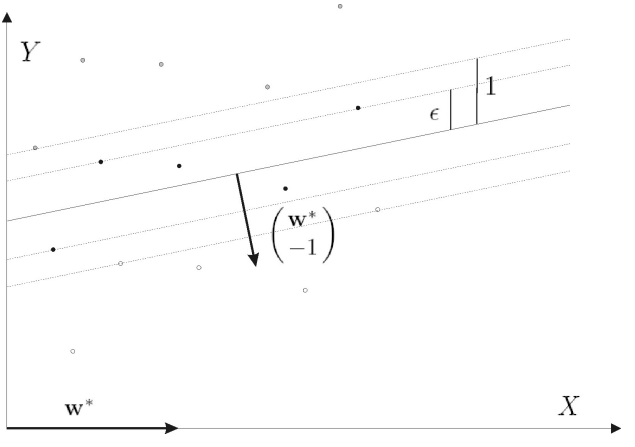

Fig. 1. Construction of the classification problem. The black dots are the true samples which we want to approximate. By shifting them orthogonal to the SVM fit curve in both senses of direction and assigning different classes (bright ($\mathbf{x}_i^1$) and dark gray ($\mathbf{x}_i^{-1}$) dots) we construct a classification problem whose maximal margin separating hyperplane is exactly the regression fit.

In the following without loss of generality we assume $C \to \infty$ and suppose that a solution with no more error than ϵ exists. Otherwise we choose an appropriate $C > 0$ and reset $K'(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}, \mathbf{z}) + \frac{\delta(\mathbf{x},\mathbf{z})}{C}$. First of all, the Support Vector solution and only the

SV solution is a fixed point of the algorithm. This can be seen as follows. Once we have reached the SV solution, the outer condition $(f(\mathbf{x}_{i_{\max}}) - y_{i_{\max}}) - (f(\mathbf{x}_{i_{\min}}) - y_{i_{\min}}) > \epsilon + \epsilon$ of the algorithm is evaluated to false, all points have to be within the ϵ-tube. The left hand side of the inner conditions is always 0, because only Support Vectors have non-zero Lagrange coefficients and all Support Vectors lie exactly on the edge of the ϵ-tube, while the right hand side is always positive. Hence nothing would change and the algorithm has finished.

On the other hand, it can further be seen that no other configuration of α can be a fixed point. If any pair of vectors lie outside of the ϵ-tube, then the outer condition would be fulfilled. Otherwise there has to be at least one vector within the ϵ-tube and not on its edge having non-zero Lagrange coefficient and therefore leading to a positive fulfillment of the inner condition after a finite number of iterations.

Now we show that the MaxMinOver Regression algorithm indeed approaches the Support Vector solution, by ascribing the regression problem to a classification problem. We construct the classification problem within the extended $(\dim(Z)+1)$-dimensional space, where Z is the feature space and the additional dimension represents the function values y, and demonstrate that the classification and the regression algorithms both work uniformly in the limit. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ be the set of input vectors with its function values $Y = \{y_1, \ldots, y_n\}$ and $\mathbf{w}^*$ the Support Vector solution of the regression problem. Of course, if one uses a non-linear feature space, then $\mathbf{x}_i$ has to be replaced by $\Phi(\mathbf{x}_i)$ or $\Phi'(\mathbf{x}_i)$, respectively. We construct the set $\hat{X} = \{\mathbf{x}_1^1, \mathbf{x}_1^{-1}, \ldots, \mathbf{x}_l^1, \mathbf{x}_l^{-1}\}$ from X by substituting

$$\mathbf{x}_i^1 = \begin{pmatrix} \mathbf{x}_i \\ y_i \end{pmatrix} + \frac{1+\epsilon}{\|\mathbf{w}^*\|^2+1} \begin{pmatrix} \mathbf{w}^* \\ -1 \end{pmatrix}$$

$$\mathbf{x}_i^{-1} = \begin{pmatrix} \mathbf{x}_i \\ y_i \end{pmatrix} - \frac{1+\epsilon}{\|\mathbf{w}^*\|^2+1} \begin{pmatrix} \mathbf{w}^* \\ -1 \end{pmatrix}$$

and assigning the classes $y_i^a = a$. The Support Vector solution of this classification problem is $\hat{\mathbf{w}}^* = \begin{pmatrix} \mathbf{w}^* \\ -1 \end{pmatrix}$ apart from its norm with functional margin 1, because

$$\min_{i,y} \max_b y((\hat{\mathbf{w}}^*)^T \mathbf{x}_i^y - b) = \min_{i,y} \max_b y((\mathbf{w}^*)^T \mathbf{x}_i - y_i - b) \quad (1)$$

$$+1+\epsilon \quad (2)$$

$$= -\epsilon + 1 + \epsilon$$

$$= 1$$

and for each vector $\hat{\mathbf{v}} \neq \hat{\mathbf{w}}^*$ (with $\|\hat{\mathbf{v}}\| = \|\hat{\mathbf{w}}^*\|$) it holds both $\hat{\mathbf{v}}^T \hat{\mathbf{w}}^* \frac{1+\epsilon}{\|\mathbf{w}^*\|^2+1} < (\hat{\mathbf{w}}^*)^T \hat{\mathbf{w}}^* \frac{1+\epsilon}{\|\mathbf{w}^*\|^2+1} = 1+\epsilon$ (compare eqn. part 2) and at least for the Support Vectors either $\mathbf{v}^T \mathbf{x}_i - y_i - b \geq \epsilon$ or $y_i - \mathbf{v}^T \mathbf{x}_i + b \geq \epsilon$ using any bias b (compare eqn. part 1), because $\mathbf{w}^*$ is the Support Vector solution of the actual regression problem.

Now suppose an interim solution $\mathbf{w}$. Then MaxMinOver for classification and for regression choose the same vectors in the next iteration. It holds for the first two find-statements of algorithm 1

$$\begin{aligned}\arg\min_i \left(\mathbf{w}^T\mathbf{x}_i - y_i\right) &= \arg\min_i \left(\mathbf{w}^T\mathbf{x}_i - y_i + S\right) \qquad (3)\\ &= \arg\min_i \hat{\mathbf{w}}^T\mathbf{x}_i^1\\ \arg\max_i \left(\mathbf{w}^T\mathbf{x}_i - y_i\right) &= \arg\max_i \left(\mathbf{w}^T\mathbf{x}_i - y_i - S\right) \qquad (4)\\ &= \arg\max_i \hat{\mathbf{w}}^T\mathbf{x}_i^{-1}\\ &= \arg\min_i \left(-\hat{\mathbf{w}}^T\mathbf{x}_i^{-1}\right)\\ \text{with } S &= \frac{(1+\epsilon)(\mathbf{w}^T\mathbf{w}^* + 1)}{\|\mathbf{w}^*\|^2 + 1}\end{aligned}$$

and further for the the second two statements concerning the dememorisation of non Support Vectors

$$\begin{aligned}\arg\max_{i,\alpha_i>0} \left(\mathbf{w}^T\mathbf{x}_i - y_i\right) &= \arg\max_{i,\alpha_i>0} \left(\mathbf{w}^T\mathbf{x}_i - y_i + S\right) \qquad (5)\\ &= \arg\max_{i,\alpha_i>0} \hat{\mathbf{w}}^T\mathbf{x}_i^1\\ \arg\min_{i,\alpha_i<0} \left(\mathbf{w}^T\mathbf{x}_i - y_i\right) &= \arg\min_{i,\alpha_i<0} \left(\mathbf{w}^T\mathbf{x}_i - y_i - S\right) \qquad (6)\\ &= \arg\min_{i,\alpha_i<0} \hat{\mathbf{w}}^T\mathbf{x}_i^{-1}\\ &= \arg\max_{i,\alpha_i<0} \left(-\hat{\mathbf{w}}^T\mathbf{x}_i^{-1}\right).\end{aligned}$$

Note that the constraints $\alpha_i \geq 0$, respectively $\alpha_i \leq 0$ implicitly hold for eq. 3, respectively eq. 4 while for eqn. 5 and 6 it is necessary to choose only potential Support Vectors of the correct classes for dememorisation.

The presented interrelationship implies that, if the classification algorithm will increment or decrement some $y_i\alpha_i$, then the regression algorithm does so as well. But while $\|\hat{\mathbf{w}}^{classification}\|$ increases linearly in time and converges to $\hat{\mathbf{w}}^*$ (apart from its norm), $\|\hat{\mathbf{w}}^{regression}\|$ must not increase arbitrarily. There exists a time $t_{start} < t$, from which on $\exists C_1, C_2 \in \mathbb{R} : C_1 t < -\hat{\mathbf{w}}_{l+1}^{classification} < C_2 t$ with $C_1 < C_2$. But for $\hat{\mathbf{w}}^{regression}$ is $\hat{\mathbf{w}}_{l+1}^{regression} = -1$ implicitly given. Hence $\|\hat{\mathbf{w}}^{regression}\|$ has to be constrained and therefore we choose an appropriate step width of order $O\left(\frac{1}{t}\right)$ (instead of normalising after each iteration). Although the harmonic series diverges and hence any possible solution vector within the feature space is reachable, we want to emphasize that the convergence speed can be tuned significantly by normalising Y or choosing an appropriate constant factor for the step width.

We derive the criterion for dememorisation of non Support Vectors from the classification method, where $R^2_{classification} \leq \max_{i,j,y_i=1,y_j=-1} (\mathbf{x}_i - \mathbf{x}_j)^2$ has already been proven [1,2]. As far this criterion must only be an upper bound, we estimate

$$\begin{aligned} R^2_{regression} &= \max_{i,j} \left(\left(\begin{pmatrix} \mathbf{x}_i \\ y_i \end{pmatrix} + \frac{1+\epsilon}{\|\mathbf{w}^*\|^2 + 1} \begin{pmatrix} \mathbf{w}^* \\ -1 \end{pmatrix} \right) \right. \\ &\quad \left. - \left(\begin{pmatrix} \mathbf{x}_j \\ y_j \end{pmatrix} - \frac{1+\epsilon}{\|\mathbf{w}^*\|^2 + 1} \begin{pmatrix} \mathbf{w}^* \\ -1 \end{pmatrix} \right) \right)^2 \\ &\leq R^2_{classification} + 4 \left(1 + \frac{3\epsilon^2 + 4\epsilon + 1}{\|\mathbf{w}^*\|^2 + 1} \right) \\ &\leq R^2_{classification} + 4 \left(2 + 3\epsilon^2 + 4\epsilon \right). \end{aligned}$$

Note that we do not need to know the solution $\mathbf{w}^*$. Hence it is possible to apply this estimation in practice.

5 Experimental Results on Artificial Data

To evaluate the MaxMinOver Regression algorithm we constructed randomly generated artificial datasets and modified the variance, the regression error, the parameters C and ϵ, the number of iteration steps, the distribution of the data points, the number of training examples, and the dimension of the feature space. The table shows the averaged results of these evaluations. It can be seen that the regression error is comparable to the one achieved by the LibSVM Toolbox. The higher the number of iterations the better is the performance of our method and the closer to the results of the LibSVM, whose step width was not changed. Only the number of Support Vectors is sometimes different. More iteration steps in both methods should lead to a convergence of the number of Support Vectors to each other.

Table 1. Averaged regression results obtained with MaxMinOver Regression (B) on artificial data. For comparison averaged results obtained with the ϵ-SVR of the LibSVM Toolbox (A) are listed. The simple MaxMinOver Regression algorithm achieves comparable results with a few training steps. (Caption: $\mathbf{w}$ norm of weight vector, L_2 squared error, $L_{2,\epsilon}$ squared error tolerating error ϵ, $|SV|$ number of Support Vectors)

Parameters		Results								
steps	distr	number	$\|\mathbf{w}_A\|$	$\|\mathbf{w}_B\|$	$L_{2,A}$	$L_{2,B}$	$L_{2,\epsilon,A}$	$L_{2,\epsilon,B}$	$\|SV_A\|$	$\|SV_B\|$	$\frac{\|\mathbf{w}_A - \mathbf{w}_B\|}{\|\mathbf{w}_A\|}$
1200			2.0218	2.0431	0.0052	0.0043	0.0007	0.0008	106	111	0.0368
2400			2.0231	2.0358	0.0052	0.0041	0.0007	0.0008	107	107	0.0289
6000			2.0372	2.0426	0.0051	0.0045	0.0007	0.0007	107	89	0.0134
	uniform		2.3792	2.3922	0.0054	0.0044	0.0003	0.0004	71	102	0.0137
	normal		0.9718	0.9853	0.0045	0.0041	0.0020	0.0020	215	105	0.0639
		50	1.9600	1.9827	0.0080	0.0063	0.0006	0.0006	42	43	0.0342
		500	2.0947	2.0983	0.0024	0.0023	0.0009	0.0010	171	162	0.0183

6 Conclusions

Regression is an important mathematical problem which occurs in a wide variety of practical applications. Support Vector regression achieves results with a high generalisation capability. Different complicated Support Vector regression approaches have been introduced in the past.

The main goal of this paper is hence to show that even Support Vector regression can be dealt with in a simple way with the MaxMinOver Regression approach. In benchmarks the method achieves performances as good as the LibSVM Toolbox. Both, its simplicity and its good performance makes this approach interesting for implementations in industrial environments.

References

1. Martinetz, T., Labusch, K., Schneegass, D.: Softdoubleminover: A simple procedure for maximum margin classification. Proc. of the International Conference on Artificial Neural Networks (2005) 301–306
2. Martinetz, T.: Maxminover: A simple incremental learning procedure for support vector classification. Proc. of the International Joint Conference on Neural Networks (IEEE Press) (2004) 2065–2070
3. Schneegass, D., Martinetz, T., Clausohm, M.: Onlinedoublemaxminover: A simple approximate time and information efficient online support vector classification method. Proc. of the European Symposium on Artificial Neural Networks (2006 (in preparation))
4. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (1995) 273–297
5. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)
6. LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., Simard, P., Vapnik, V.: Comparison of learning algorithms for handwritten digit recognition. Int.Conf.on Artificial Neural Networks (1995) 53–60
7. Osuna, E., Freund, R., Girosi, F.: Training support vector machines:an application to face detection. CVPR'97 (1997) 130–136
8. Schölkopf, B.: Support vector learning (1997)
9. Friess, T., Cristianini, N., Campbell, C.: The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. Proc. 15th International Conference on Machine Learning (1998)
10. Platt, J.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Advances in Kernel Methods - Support Vector Learning. MIT Press (1999) 185–208
11. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: A fast iterative nearest point algorithm for support vector machine classifier design. IEEE-NN **11**(1) (2000) 124–136
12. Li, Y., Long, P.: The relaxed online maximum margin algorithm. Machine Learning **46(1-3)** (2002) 361–387
13. Cristianini, N., Shawe-Taylor, J.: Support Vector Machines And Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
14. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, Inc., New York (1998)
15. Martinetz, T.: Minover revisited for incremental support-vector-classification. Lecture Notes in Computer Science **3175** (2004) 187–194