ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

OnlineDoubleMaxMinOver: A Simple Approximate Time And Information Efficient Online Support Vector Classification Method

Daniel Schneegaß[1,2,3], Thomas Martinetz[1], and Michael Clausohm[2]

1- University at Luebeck, Institute for Neuro- and Bioinformatics,
Ratzeburger Allee 160, D-23538 Luebeck, Germany

2- Clausohm Software GmbH, {CIRCLE Development,Executive Board},
Neubrandenburger Straße 46, D-17039 Neverin, Germany

3- Siemens AG, Corporate Technology, Learning Systems,
Otto-Hahn-Ring 6, D-81739 Munich, Germany

Abstract. We present the OnlineDoubleMaxMinOver approach to obtain the Support Vectors in two class classification problems. With its linear time complexity and linear convergence the algorithm achieves a competitive speed. We approach the problem of the impossibility of perfect non trivial online Support Vector Learning by parameterising the exactness. Even in the case of linearly inseparable data within the feature space the method converges to a solution expressible by a finite amount of information while observing an arbitrarily large number of input vectors. The results of the online method are comparable to the batch ones, occasionally even better.

1 Introduction

The Support Vector Machine [1, 2] became a sophisticated and widely used tool for classification and regression tasks. Many fast and robust methods to calculate the Support Vector solution were invented. We want to mention especially the work of Platt [3] and Anlauf and Biehl [4] which has some similarities to MaxMinOver. The Sequential Minimisation Optimisation algorithm holds to be the fastest batch learning method in practice for classification as well as regression tasks.

Online learning for Support Vector Machines is characterised by a principle problem. By geometrical arguments it can be seen immediately that in the worst case the maximal margin hyperplane depends on all visited input vectors of the convex hulls of both classes. Especially in high-dimensional spaces the amount of vectors belonging to the convex hull is asymptotically as big as the number of data vectors. But the solution itself can be formed by only a few data vectors, the so-called Support Vectors. It is always easy to construct a trivial online learning method, which simply adds each new observed input vector to the set of all other already observed data vectors and calculates the next batch Support Vector solution. Using any batch method with time complexity $\Theta(T)$ would lead to an online learning method with complete time complexity $\Theta(lT)$ after observing l samples. Obviously this is not at all satisfactorily. Only if the convex

hulls of both classes contain just an asymptotically sufficiently small number of input vectors, e.g. a constant one, then the calculation of the participation in the convex hull of a new observed input vector, that is solving a linear programming problem, is of course possible in constant time as well. If not, then it is principally not possible to obtain the exact Support Vector solution with a time complexity of $o(lT')$ with $\Theta(T')$ as the time complexity of the fastest batch learning method. All methods like e.g. [5] lead only to approximate solutions, all like [6] need at least linear time per iteration.

2 Support Vector Classification

A data set $X \in \mathbb{R}^n$ and its labels $Y \in \{-1, 1\}^n$ are given. The goal is to find a vector $\mathbf{w}$ and a scalar b which realise classification using the function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$ holding $\forall i : f(\mathbf{x}_i) = y_i$. Apart from its norm a unique vector $\mathbf{w}^* = \arg\max_{\|\mathbf{w}\|=1} \min_i y_i(\mathbf{w}^T\mathbf{x}_i + b)$ provides the maximal margin solution, whose class separating hyperplane has the largest possible distance to its closest input vectors. This is identical to $\mathbf{w}^T\mathbf{w} = \min, \forall i : y_i \left(\mathbf{w}^T\mathbf{x} + b\right) \geq 1$.

Using the well-known Lagrange formalism it turns out that the solution of the weight vector can be written as $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$, where α_i are the Lagrange coefficients with every strictly positive α_i representing a hard constraint and, therefore, a Support Vector which is classified most critically. Hence, it is possible to write the classifier finally as

$$f(\mathbf{x}) \quad = \quad \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b\right).$$

Important is the fact that the inner product $\mathbf{x}^T\mathbf{z}$ can be substituted by a symmetric and positive definite kernel $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$ which implicitly transforms the input space into an appropriate feature space. In the case of linearly non-separable data sets within the feature space it is necessary to find a possibility to tolerate slight errors. This can be achieved by introducing so-called slack variables and modifying the optimisation problem to e.g.

$$\begin{aligned} \mathbf{w}^T\mathbf{w} + C\|\xi\|_2^2 &= \min \\ \forall i : y_i \left(\mathbf{w}^T\mathbf{x}_i + b\right) &\geq 1 - \xi_i. \end{aligned}$$

This special error model leads to a modified kernel $K'(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{i,j}}{C}$ with δ as the Kronecker symbol, which is essentially the same as is used in kernalized Ridge Regression [2]. Hence, straightforwardly any Hard Margin Support Vector method can be directly extended to an appropriate Soft Margin method by using this error model.

3 DoubleMaxMinOver

The MaxMinOver method [7] is a simple and easy implementable incremental but batch learning algorithm to obtain the Support Vector solution in classification tasks. It is based on the MinOver algorithm [8] which converges to the

maximal margin solution. The MaxMinOver method is extended by a so-called dememorisation criterion which leads to a removing of input vectors which are proven not to be Support Vectors.

The DoubleMaxMinOver algorithm is in turn an extension of MaxMinOver. In each iteration step the worst and the best classified input vectors of each class have to be searched. Then the corresponding Lagrange coefficients $\alpha_{i_{1,\min}}$ and $\alpha_{i_{-1,\min}}$ of the worst input vectors are each incremented by one, if the integrated dememorisation criterion is not fulfilled. Otherwise these coefficients are incremented by two and the $\alpha_{i_{1,\max}}$ and $\alpha_{i_{-1,\max}}$ of the best classified input vectors are each decremented by one (as long as $\alpha_{i_{1,\max}}, \alpha_{i_{-1,\max}} > 0$). Apparently, the algorithm needs linear time for each iteration. Furthermore, a linear convergence speed in the exactness of $\mathbf{w}^*$ measured as $E = \sin \angle (\mathbf{w}^*, \mathbf{w})$ was proven. Due to the limited space we have to refer to [7, 8, 9] for further details.

4 OnlineDoubleMaxMinOver

4.1 The Hard Margin Case

As already mentioned above, in online Support Vector Learning there exists a basic problem. The order of observed input vectors can be unfavorable. In general the trivial approach of calculating the new Support Vector solution by taking the interim set of Support Vectors and the new input vector does not lead to correct solutions. On the contrary, it is necessary to take asymptotically all already observed input vectors into account. This problem has to be incorporated in DoubleMaxMinOver as well.

Hence, in the hard margin case we simply extend the dememorisation criterion. We introduce a new parameter $A \geq 1$ and decide input vectors $\mathbf{x}_{i_{1,\min}}$ respectively $\mathbf{x}_{i_{-1,\min}}$ definitely not to be Support Vectors, if $\mathbf{w}^T(\mathbf{x}_{i_{1,\max}} - \mathbf{x}_{i_{1,\min}}) > 4AR^2$, respectively $\mathbf{w}^T(\mathbf{x}_{i_{-1,\min}} - \mathbf{x}_{i_{-1,\max}}) > 4AR^2$ with R as an upper bound of the greatest distance within the feature space between two input vectors of both classes. Then these vectors will be removed. The larger A, the more exact is the obtained solution.

The time complexity of the OnlineDoubleMaxMinOver algorithm remains linear, if the number of Support Vectors is bounded by a constant $D(X, K)$, e.g. by theoretical propositions [1]. This can be seen as follows. After a finite number of steps the dememorisation criterion will not be fulfilled only by current approximate Support Vectors [7]. The number of approximate Support Vectors is bounded by a constant as well, because the considered data set $\hat{X} \subset X$ is a subset of the complete input set X[1]. So after a finite number of observations only a constant number of potential Support Vectors has to be taken into account. With a general number n_{SV} of approximate Support Vectors the time complexity finally is $O(n_{SV} l)$.

[1] E.g. the number of the so-called essential Support Vectors is bounded by $n \leq \left(\frac{R^2}{(\delta^*)^2}, d\right)$, where R is the radius of the sphere containing all input vectors, δ^* the margin, and d the dimension of the feature space [1]. By removing some input vectors, the inequality apparently remains valid.

4.2 The Soft Margin Case

The Soft Margin case is much more complicated. If one uses the common error model described above, or a comparable one, then the number of Support Vectors is in general a fraction of the whole number of observed input vectors. This fraction depends on the parameter C. That is, the number of Support Vectors $n_{SV} \to \infty$ will be arbitrarily large with an arbitrarily large number of observed input vectors $l \to \infty$. Therefore, the error model has to be changed slightly with the goal of bounding the number of Support Vectors by a constant, independent of the number of observations.

First of all, in the case of an infinite number of observations the above primal optimisation problem can be generalised to a continuous problem as

$$\begin{aligned} \|\mathbf{w}\| + C\|\xi\|_{L_2}^2 &= \min \\ \forall \mathbf{x} \in X : y(\mathbf{x})(\mathbf{w}^T\mathbf{x} + b) &\geq 1 - \xi(\mathbf{x}), \end{aligned}$$

which in the dual perspective leads to the modified kernel $K'(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}, \mathbf{z}) + \frac{\delta(\mathbf{x},\mathbf{z})}{C}$ with $\delta(\mathbf{x}, \mathbf{z}) = \delta(\mathbf{x} - \mathbf{z})$ as the Dirac function. By weakening its hardness to

$$\delta(\mathbf{x}, \mathbf{z}) = \begin{cases} d(\mathbf{x} - \mathbf{z}) & : \quad y(\mathbf{x}) = y(\mathbf{z}) \\ 0 & : \quad y(\mathbf{x}) \neq y(\mathbf{z}) \end{cases}$$

e.g. $d(\mathbf{x}) = \sqrt{n}e^{-n\mathbf{x}^T\mathbf{x}}$ with an appropriate parameter n we obtain the Kernel enlargement defined as

$$\begin{aligned} \delta_K(\mathbf{x}, \mathbf{z}) &= \begin{cases} d(\Phi(\mathbf{x}) - \Phi(\mathbf{z})) & : \quad y(\mathbf{x}) = y(\mathbf{z}) \\ 0 & : \quad y(\mathbf{x}) \neq y(\mathbf{z}) \end{cases} \\ d(\Phi(\mathbf{x}) - \Phi(\mathbf{z})) &= \sqrt{n}e^{-n(\Phi(\mathbf{x})-\Phi(\mathbf{z}))^T(\Phi(\mathbf{x})-\Phi(\mathbf{z}))} \\ &= \sqrt{n}e^{-n(K(\mathbf{x},\mathbf{x})+K(\mathbf{z},\mathbf{z})-2K(\mathbf{x},\mathbf{z}))}. \end{aligned}$$

If $\delta = \delta^1 + \delta^{-1}$ can be formulated as the sum of two autocorrelations of positive functions $f_1, f_{-1} \in L_2$, then the feature space constructed this way can be formulated in terms of the transformation function $\hat{\Phi} : X \to (\Phi(X) \times (\Phi(X) \to \mathbb{R}) \times (\Phi(X) \to \mathbb{R}))$ as follows

$$\begin{aligned} \hat{\Phi}(\mathbf{x}) &= (\Phi(\mathbf{x}), \mathbf{z} \to f_1(\Phi(\mathbf{x}), \mathbf{z}), \mathbf{z} \to f_{-1}(\Phi(\mathbf{x}), \mathbf{z})) \\ f_a(\mathbf{x}, \mathbf{z}) &= f_{a,\mathbf{x}}(\mathbf{x} - \mathbf{z}) \geq 0 \\ f_{a,\mathbf{x}}(\mathbf{x}') &= 0, y(\mathbf{x}) \neq a \\ \int_{\Phi(X)} f_{a,\mathbf{x}}(\mathbf{x} - \mathbf{z}) f_{a,\mathbf{x}'}(\mathbf{x}' - \mathbf{z}) d\mathbf{z} &= \begin{cases} d(\mathbf{x} - \mathbf{x}') & : \quad y(\mathbf{x}) = y(\mathbf{x}') = a \\ 0 & : \quad y(\mathbf{x}) \neq a \vee y(\mathbf{x}') \neq a \end{cases} \\ &= \delta^a(\mathbf{x}, \mathbf{x}'). \end{aligned}$$

The method always converges, because by construction at least the weight vector

$$\begin{aligned} \mathbf{w} &= (\mathbf{w}', \mathbf{w}_1, \mathbf{w}_{-1}) \in (\Phi(X) \times (\Phi(X) \to \mathbb{R}) \times (\Phi(X) \to \mathbb{R})) \\ \forall a \in \{1, \ldots, D\} : \mathbf{w}'_a &= 0 \\ \forall a \in \{-1, 1\} : \mathbf{w}_a &= (\mathbf{x} \to a) \end{aligned}$$

separates the two classes within the Soft Margin feature space correctly.

In the primal perspective this modification leads to a low-pass filtering of the slack variables by d. Together with some requirements on that function and the closeness of the input vector's support indeed the number of Support Vectors is bounded by a constant. Intuitively, any chosen Support Vector $\mathbf{x}$ with a non zero $\xi(\mathbf{x})$ shares the error with its direct neighbors and therewith already "pays" for them. Then these neighbors may not be taken into account as Support Vectors anymore. Due to the lack of space we cannot illustrate it in a formal way.

Algorithm 1 The C2-OnlineSoftMaxMinOver Algorithm

Require: given generator $P : \mathbb{N} \to \mathbb{R}^d \times \{1, -1\}$ creating set of input data and parameter C, R an upper bound for the greatest distance within the feature space between two input vectors of both classes, A an appropriate factor regularising the exactness of the solution
Ensure: calculates a soft margin hyperplane given in dual representation
set $K'(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}, \mathbf{z}) + \frac{\delta_{K(\mathbf{x},\mathbf{z})}}{C}$, $t \leftarrow 0$, $\mathbf{x} \leftarrow$ nil, $y \leftarrow$ nil, $\alpha \leftarrow$ nil, $l \leftarrow 0$
while the desired precision is not reached **do**
  set $t \leftarrow t + 1$, $l \leftarrow l + 1$, $(\mathbf{x}_l, y_l) \leftarrow P(t)$, $\alpha_l \leftarrow 0$
  if it exists i with $y_i = 1$ and it exists i with $y_i = -1$ **then**
    find $i_{1,\min} = \arg\min_{i,y_i=1} \sum_{j=1}^{l} \alpha_j y_j K'(\mathbf{x}_j, \mathbf{x}_i)$
    find $i_{-1,\min} = \arg\max_{i,y_i=-1} \sum_{j=1}^{l} \alpha_j y_j K'(\mathbf{x}_j, \mathbf{x}_i)$
    find $i_{1,\max} = \arg\max_{i,y_i=1,\alpha_i>0} \sum_{j=1}^{l} \alpha_j y_j K'(\mathbf{x}_j, \mathbf{x}_i)$
    find $i_{-1,\max} = \arg\min_{i,y_i=-1,\alpha_i>0} \sum_{j=1}^{l} \alpha_j y_j K'(\mathbf{x}_j, \mathbf{x}_i)$
    if $\sum_{i=1}^{l} \alpha_i y_i \left(K'\left(\mathbf{x}_i, \mathbf{x}_{i_{1,\max}}\right) - K'\left(\mathbf{x}_i, \mathbf{x}_{i_{1,\min}}\right)\right) > 4R^2$ **then**
      set $\alpha_{i_{1,\min}} \leftarrow \alpha_{i_{1,\min}} + 2$ and $\alpha_{i_{1,\max}} \leftarrow \alpha_{i_{1,\max}} - 1$
    else
      set $\alpha_{i_{1,\min}} \leftarrow \alpha_{i_{1,\min}} + 1$
    end if
    if $\sum_{i=1}^{l} \alpha_i y_i \left(K'\left(\mathbf{x}_i, \mathbf{x}_{i_{-1,\min}}\right) - K'\left(\mathbf{x}_i, \mathbf{x}_{i_{-1,\max}}\right)\right) > 4R^2$ **then**
      set $\alpha_{i_{-1,\min}} \leftarrow \alpha_{i_{-1,\min}} + 2$ and $\alpha_{i_{-1,\max}} \leftarrow \alpha_{i_{-1,\max}} - 1$
    else
      set $\alpha_{i_{-1,\min}} \leftarrow \alpha_{i_{-1,\min}} + 1$
    end if
    for all $i, y_i \sum_{j=1}^{l} \alpha_j y_j \left(K'(\mathbf{x}_j, \mathbf{x}_i) - K'\left(\mathbf{x}_j, \mathbf{x}_{i_{y_i,\min}}\right)\right) > 4AR^2, \alpha_i = 0$ **do**
      remove $\mathbf{x}_i$, y_i and α_i and set $l \leftarrow l - 1$
    end for
  end if
end while
set $b \leftarrow \left(\sum_{i=1}^{l} \alpha_i \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)\right) \left(\sum_{i=1}^{l} \alpha_i\right)^{-1}$

5 Experiments and Conclusion

We tested OnlineDoubleMaxMinOver and compared it with its batch version on the repository of the Fraunhofer Institute [10], whose performance has been compared convincingly with the state-of-the-art, the SMO-algorithm [3], in [9]. Table 1 shows that the online version is comparable to its batch version. As long as the width of the low-pass filter is small enough, the results are even better. But this has to be paid by a slightly larger number of Support Vectors.

We showed that OnlineDoubleMaxMinOver is a very simple approximate online Support Vector Learning method. As its batch version it is furthermore very efficient in running time and can therefore be used for a wide range of practical applications where online learning is an important requirement.

Table 1: Comparison of the classification results of C2-SoftDoubleMaxMinOver as batch and online method on standard benchmarks. The results are averaged over different σ and C. The Gaussian Kernel was used. Low-passes are the Delta Kernel, which is actually an all-pass and the Gaussian Kernel. σ_K is the width of the low-pass and ϵ the width of a Gaussian noise on the input data.

Parameter					Results		
Batch/Online	No. Iter.	Low-pass	σ_K	ϵ	err_{Train}	err_{Test}	n_{SV}
Batch	1000				0.12	0.30	350
Batch	5000				0.12	0.30	356
Online	5000	Delta		0	0.11	0.29	377
Online	10000	Delta		0	**0.10**	**0.28**	383
Online	5000	Gaussian	0.0002	0	**0.10**	**0.28**	372
Online	5000	Gaussian	0.001	0	0.11	0.29	371
Online	5000	Gaussian	0.005	0	0.11	**0.28**	363
Online	5000	Gaussian	0.01	0	0.12	0.29	363
Online	5000	Gaussian	0.05	0	0.14	0.31	352
Online	5000	Gaussian	0.1	0	0.14	0.31	**339**
Online	5000	Gaussian	0.0002	0.01	0.14	0.30	379
Online	5000	Gaussian	0.001	0.01	0.14	0.31	374
Online	5000	Gaussian	0.005	0.01	0.14	0.31	372
Online	5000	Gaussian	0.01	0.01	0.14	0.31	368
Online	5000	Gaussian	0.05	0.01	0.15	0.32	359
Online	5000	Gaussian	0.1	0.01	0.17	0.33	350

References

[1] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Cambridge University Press, Cambridge, 2000.

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

[3] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

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

[5] Liva Ralaivola and Florence d'Alché Buc. Incremental support vector machine learning: A local approach. In *ICANN*, pages 322–330, 2001.

[6] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2000.

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

[8] T. Martinetz. Minover revisited for incremental support-vector-classification. *Lecture Notes in Computer Science*, 3175:187–194, 2004.

[9] Thomas Martinetz, Kai Labusch, and Daniel Schneegass. Softdoubleminover: A simple procedure for maximum margin classification. *Proc. of the International Conference on Artificial Neural Networks*, pages 301–306, 2005.

[10] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm.