An improvement of AdaBoost to avoid overfitting

Gunnar Rätsch Takashi Onoda† Klaus Robert Müller
Email:{raetsch,onoda,klaus}@first.gmd.de

GMD FIRST Rudower Chaussee 5, 12489 Berlin, Germany
†Permanent address: Communication & Information Research Lab., CRIEPI
2-11-1, Iwado Kita, Komae-shi, Tokyo 201 Japan

ABSTRACT

Recent work has shown that combining multiple versions of weak classifiers such as decision trees or neural networks results in reduced test set error. To study this in greater detail, we analyze the asymptotic behavior of AdaBoost. The theoretical analysis establishes the relation between the distribution of margins of the training examples and the generated voting classification rule. The paper shows asymptotic experimental results with RBF networks for the binary classification case underlining the theoretical findings. Our experiments show that AdaBoost does overfit, indeed. In order to avoid this and to get better generalization performance, we propose a regularized improved version of AdaBoost, which is called *AdaBoost$_{reg}$*. We show the usefulness of this improvement in numerical simulations.

KEYWORDS: ensemble learning, AdaBoost, margin distribution, generalization, support vectors, RBF networks

1. Introduction

An ensemble is a collection of neural networks or other types of classifiers (hypotheses) that are trained for the same task. Boosting and other ensemble learning methods have been used recently with great success for several applications, e.g. OCR [12, 7].

In this work we investigate the functioning of the ensemble learning process in greater detail by looking at the asymptotic character of AdaBoost as a particular example of an ensemble method. The learning characteristics can be studied nicely from the margins perspective (section 2 and 3). Discussing the error function of AdaBoost we can introduce the intuition of annealing and suggest strategies to improve AdaBoost in practice (section 4).

2. AdaBoost algorithm

Let $\{h_t(\mathbf{x}) : t = 1, \ldots, T\}$ be an ensemble of T hypotheses defined on input vector $\mathbf{x}$ and $\mathbf{c} = [c_1 \ldots c_T]$ their weights satisfying $c_t > 0$ and $\sum_t c_t = 1$. In the binary classification case, the output is one of two class labels, i.e. $h_t(\mathbf{x}) = \pm 1$.

The ensemble generates the label which is the weighted majority of the votes: $\mathrm{sgn}\left(\sum_t c_t h_t(\mathbf{x})\right)$. In order to train this ensemble of T hypotheses $\{h_t(\mathbf{x})\}$ and $\mathbf{c}$, several algorithms have been proposed: bagging, where the weighting is simply $c_t = 1/T$ [3] and AdaBoost/Arcing, where the weighting scheme is more complicated [11, 4].

In the following we give a brief description of the AdaBoost/Arcing algorithms. We use a special form of Arcing, which is equivalent to AdaBoost [4]. In the binary classification case we define the margin for an input-output pair $\mathbf{z}_i = (y_i, \mathbf{x}_i), (i = 1, \ldots, N)$ by

$$mg(\mathbf{z}_i, \mathbf{c}) = y_i \sum_{t=1}^{T} c_t h_t(\mathbf{x}_i), \tag{1}$$

which is between -1 and 1, if $|\mathbf{c}| = 1$. The correct class is predicted, if the margin at $\mathbf{z}$ is positive. When the positivity of the margin value increases, the decision correctness (stability) becomes larger. AdaBoost asymptotically minimizes a function of the margin $mg(\mathbf{z}_i, \mathbf{c})$ [10]

$$g(\mathbf{b}) = \sum_i \exp\left\{-\frac{|\mathbf{b}|}{2} mg(\mathbf{z}_i, \mathbf{c})\right\} \tag{2}$$

where $|\mathbf{b}| = \sum_t b_t$ (starting from $\mathbf{b} = 0$). Note that b_t is the unnormalized weighting of the hypothesis h_t. The quantity $\mathbf{c}$ is simply a normalized version of $\mathbf{b}$, i.e. $\mathbf{c} = \mathbf{b}/|\mathbf{b}|$.

In order to find the hypothesis h_t the learning examples $\mathbf{z}_i$ are weighted in each iteration t with $w_t(\mathbf{z}_i)$. Using a bootstrap on this weighted sample we train h_t, alternatively a weighted error function can be used (e.g. weighted MSE). The weights $w_t(\mathbf{z}_i)$ are computed according to[1]

$$w_t(\mathbf{z}_i) = \frac{\exp\{-|\mathbf{b}_{t-1}| mg(\mathbf{z}_i, \mathbf{c}_{t-1})/2\}}{\sum_j \exp\{-|\mathbf{b}_{t-1}| mg(\mathbf{z}_j, \mathbf{c}_{t-1})/2\}} \tag{3}$$

and the training error ϵ_t of h_t is computed as

$$\epsilon_t = \sum_{i=1}^{N} w_t(\mathbf{z}_i) I(y_i \neq h_t(\mathbf{x}_i)), \tag{4}$$

where $I(true) = 1$ and $I(false) = 0$.

For each given hypothesis h_t we have to find a weight b_t, such that $g(\mathbf{b}_t)$ is minimized. One can optimize this parameter by a line search or directly by analytic minimization [4, 10], which gives

$$b_t = \log \frac{1 - \epsilon_t}{\epsilon_t}. \tag{5}$$

Interestingly, we have

$$w_t(\mathbf{z}_i) = \frac{\partial g(\mathbf{b}_{t-1}) / \partial mg(\mathbf{z}_i, \mathbf{b}_{t-1})}{\sum_{j=1}^{N} \partial g(\mathbf{b}_{t-1}) / \partial mg(\mathbf{z}_j, \mathbf{b}_{t-1})}, \tag{6}$$

which is a gradient of $g(\mathbf{b}_{t-1})$ with respect to the margins. This $w_t(\mathbf{z}_i)$ will give a hypothesis h_t which is an approximation to the best possible hypothesis h_t^* that would be obtained by minimizing $g(\mathbf{b})$ directly. Note that, the weighted minimization (bootstrap, weighted LS) will not necessarily give

[1] This direct way for computing the weights is equivalent to the update rule of AdaBoost [10, 8].

Algorithm AdaBoost

Input: N examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Initialize: $w_1(\mathbf{z}_i) = 1/N$ for all $i = 1 \ldots N$

Do for $t = 1, \ldots, T$,

1. Train neural network with respect to weighted example w_t and obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$
2. calculate the training error ϵ_t of h_t: $\epsilon_t = \sum_{i=1}^{l} w_t(\mathbf{z}_i) I(h_t(\mathbf{x}_i) \neq y_i)$, abort if $\epsilon_t \geq \frac{1}{2}$
3. set $b_t = \log \frac{\epsilon_t}{1-\epsilon_t}$.
4. update weights w_t (Z_t is a normalization constant): $w_{t+1}(\mathbf{z}_i) = w_t(\mathbf{z}_i) \exp\{-b_t I(h_t(\mathbf{x}_i) = y_i)\} / Z_t$

Output: final hypothesis: $f(\mathbf{x}) = \frac{1}{|\mathbf{b}|} \sum_{t=1}^{T} b_t h_t(\mathbf{x})$

Figure 1: The AdaBoost algorithm

h_t^*, even if ϵ_t is minimized [10]. AdaBoost is therefore an approximate gradient descent method which minimizes $g(\mathbf{b})$ asymptotically.
For an algorithmic description see pseudocode in Fig. 1.

3. Asymptotic analysis

Inspecting Eq. (3) more closely, we see that AdaBoost uses a softmax function [1] with a parameter $|\mathbf{b}|$ that we would like to interpret as an annealing parameter. The annealing parameter $|\mathbf{b}|$ depends on the values of b_t, and if $\epsilon_t < 1/2 - \gamma$ ($\gamma > 0$) then it increases at least linearly with the number of iterations. Experimentally we observed values of $|\mathbf{b}|$ are from 10^1 to 10^3 after hundreds of iterations.

From Eq. (5), we observe that if the training error ϵ_t takes a small value, b_t becomes large. So, strong learners can reduce their training errors strongly and will make $|\mathbf{b}|$ large after only a few boosting steps. This case reaches at the asymptotical point faster, but does not necessarily achieve the best generalization performance. The reason is the same as for an usual annealing process. To reduce the annealing speed, the complexity of the base hypotheses has to be decreased. This reasoning also shows that the character of the annealing process will vary strongly depending on the strength of the learner.

A decrease of $g(\mathbf{c}, |\mathbf{b}|) := g(\mathbf{b})$, where $\mathbf{c} = \mathbf{b}/|\mathbf{b}|$, is predominantly achieved by improvements of the margin $mg(\mathbf{z}_i, \mathbf{c})$. If the margin $mg(\mathbf{z}_i, \mathbf{c})$ is negative, then the error $g(\mathbf{c}, |\mathbf{b}|)$ takes clearly a big value, which is additionally amplified by $|\mathbf{b}|$. So, AdaBoost tries to decrease the negative margin efficiently to improve the error $g(\mathbf{c}, |\mathbf{b}|)$.

Now, let us consider the asymptotic case, where the number of iterations and therefore also $|\mathbf{b}|$ take large values. In this case, when the values of all $mg(\mathbf{z}_i, \mathbf{c}), i = 1, \cdots, N$, are almost the same, small differences are amplified strongly in $g(\mathbf{c}, |\mathbf{b}|)$. For example, when the margin $mg(\mathbf{z}_i, \mathbf{c}) = 0.2$ and another margin $mg(\mathbf{z}_j, \mathbf{c}) = 0.3$ and $|\mathbf{b}|$ is 10^3, the difference is amplified to the difference between $\exp\left\{-\frac{10^3 \times 0.2}{2}\right\} = e^{-100}$ and $\exp\left\{-\frac{10^3 \times 0.3}{2}\right\} = e^{-150}$ in $g(\mathbf{c}, |\mathbf{b}|)$. Obviously the function $g(\mathbf{c}, |\mathbf{b}|)$ is asymptotically very sensitive to small differences between margins. Therefore, the margins $mg(\mathbf{z}_i, \mathbf{c})$ should converge to a fixed stability asymptotically and training patterns from the margin area (boundary area between classes) will have the same stability asymptotically (cf. Fig. 3). From Eq. (3), when the annealing parameter $|\mathbf{b}|$ takes a very big value, AdaBoost learning becomes a hard competition case: only the patterns with smallest margin will get high weights, other patterns are effectively neglected in the learning process.

To recapitulate our findings: (a) training patterns, which are in the margin area, have asymptotically the same stability and (b) the value of the stabilities is decided by the specific AdaBoost annealing process and the speed of the annealing process is an implicit function of the strength of the learner in the training process. In order to confirm that the above theoretical analysis is correct, asymptotic numerical simulations are made. In all simulations, radial basis function (RBF) networks with adaptive centers [1, 10] are used as learners. Fig. 2 (left) shows an example of our training data which is generated subject to uniformly distributed noise $U(0.0, \sigma^2)$. In our simulation, the σ^2 consists of 0.0, 0.09, and 0.16. AdaBoost is applied to these data sets. Fig. 3 shows margin distributions after 10^4 AdaBoost iterations at different σ^2 (left) and for different strengths of the base hypotheses (right). From these figures, it becomes apparent that the margin distribution asymptotically makes a step at fixed stability for training patterns which are in the margin area (cf. (a)). From Fig. 3 (left) one can see the influence of noise in the data and the strength of the base hypotheses on the minimal stability. If the noise level is high or the complexity is low, one gets higher training errors ϵ_t and for this reason different annealing behaviour, which gives different stabilities (cf. (b)).

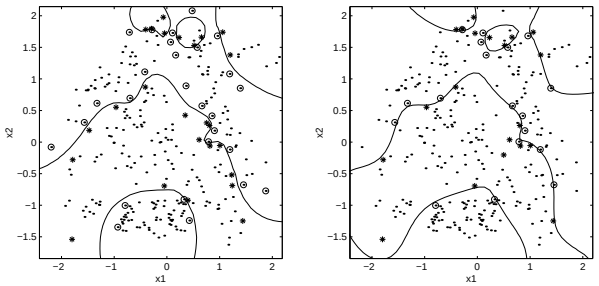


Figure 2: Training patterns with decision lines for AdaBoost (left) and SVM (right) for a low noise case with similar generalisation errors. RBF networks with 13 centers were used. The training patterns are shown as a dot, the positive and negative boundary samples are marked with '$*$' and '$\odot$' respectively.

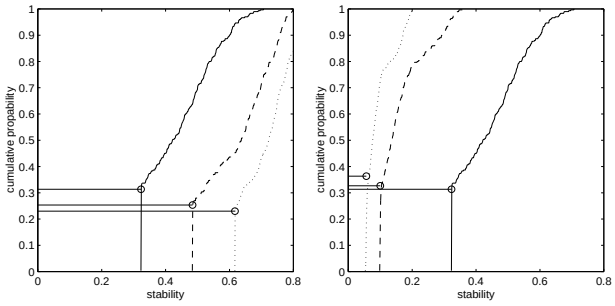


Figure 3: Margin distributions for AdaBoost for (left) different noise levels ($\sigma^2 = 0\%$ (dotted), 9% (dashed), 16% (solid)) with fixed number of RBF-centers for the base hypothesis and (right) with 7 (dotted), 13 (dashed), 30 (solid) centers with $\sigma^2 = 0.16\%$ after 10^4 AdaBoost iterations.

These numerical results support our theoretical asymptotic analysis.

Interestingly, the margin distribution of AdaBoost resembles the one of Support Vector Machines (SVMs) for the separable case [2, 5, 13]. In one example (cf. Fig. 2) almost all patterns, that are support vectors, also lie within the step part of the margin distribution for AdaBoost. AdaBoost achieves a *hard margin* asymptotically, such as the SVMs for the separable case.

The property of AdaBoost to produce a big margin area (no pattern in the area, i.e. a hard margin), will not always lead to the best generalization ability (cf. [5, 10]). This is especially true, if the training patterns have classification or input noise. In our experiments with noisy data, we often observed that AdaBoost made overfitting (for a high number of boosting iterations). Fig. 4 shows a typical overfitting behaviour in the generalization error for AdaBoost. Here, already after only 80 boosting iterations the best generalisation performance is achieved. Quinlan [9] and Grove et al. [6] also observed that the generalization performance of AdaBoost is often worse than that of the single classifier, if the data has classification noise.

The first reason for the overfitting is the increasing value of $|\mathbf{b}|$: noisy patterns (e.g. bad labeled) can asymptotically have an "unlimited" influence to the decision line leading to overfitting (cf. Eq. (3)). Another reason is the classification with a hard margin, which also means that all training patterns will asymptotically be correctly classified (without any capacity limitation!). In the presence of noise this may be not the right concept, because the best decision line (e.g. Bayes) usually will not give a training error of zero in the presence of noise. So, the achievement of large hard margins in the presence of noise will produce hypotheses which are too complex for the problem.

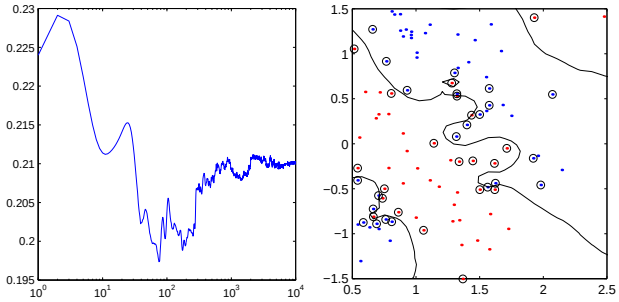


Figure 4: Typical overfitting behaviour in the generalisation error as a function of the number of iterations (left) and a typical decision line (right) generated by AdaBoost using RBF networks in the case with noise (here: 30 centers and $\sigma^2 = 16\%$; smoothed)

4. Improving AdaBoost

4.1. An analogy to weight decay

In order to avoid overfitting and to get a good generalization performance, we propose an improvement of the original AdaBoost by an analogy to weight decay [1]. We introduce slack variables similar to the support vector algorithm [5, 13].

We know that all training patterns will get non-negative stability after many iterations. Due to this fact AdaBoost often produces high weights for the difficult training patterns and this property will lead to overfitting, as observed in Fig. 4.

Asymptotically, we get the inequalities [10]

$$mg(\mathbf{z}_i, \mathbf{c}) \geq \rho \qquad \text{for all } i = 1, \ldots, l, \tag{7}$$

where ρ is the minimum margin of the patterns. Having a closer look to Eq. (2) we can see the relation between ρ and $g(\mathbf{b})$ for a sufficient large value of $|\mathbf{b}|$: as $g(\mathbf{b})$ is minimized, ρ is maximized. After many iterations, these inequalities are satisfied for $\rho \geq 0$ [11] and the hard margin is achieved, what can lead to overfitting. Therefore, we introduce some variables ζ_i - the slack variables - and get

$$mg(\mathbf{z}_i, \mathbf{c}) \geq \rho - C\zeta_i^t, \qquad \zeta_i^t > 0. \tag{8}$$

In these inequalities, ζ_i^t are positive and if a training pattern has high weights in the previous iterations, the ζ_i^t should be increasing. In this way, for example, we do not force outliers to be classified according to their possibly wrong labels, but we allow for some errors. In this sense we get a tradeoff between the margin and the importance of a pattern in the training process (depends on the constant $C \geq 0$). If we choose $C = 0$ in Eq. (8), the original AdaBoost algorithm is retrieved. If C is chosen too high, the data is not taken seriously.

In analogy to weight decay, we choose

$$\zeta_i^t = \left(\sum_{r=1}^{t} c_r w_r(\mathbf{z}_i) \right)^2, \tag{9}$$

where the inner sum is the cumulative weight of the pattern in the previous iterations. By this ζ_i^t, AdaBoost is not changed for easy classifiable patterns, but is changed for difficult patterns. This ζ_i^t gives difficult patterns big weights which are far from $\frac{1}{N}$.

From Eq. (8), we can derive a new error function:

$$\tilde{g}_{reg}(\mathbf{c}_t, |\mathbf{b}_t|) = \sum_i \exp\left\{ -\frac{|\mathbf{b}_t|}{2} mg(\mathbf{z}_i, \mathbf{c}_t) - C\zeta_i^t \right\} \tag{10}$$

By this error function, we can control the tradeoff between the weight, which the pattern has in the last iteration, and the achieved margin. The weight $w_t(\mathbf{z}_i)$ of a pattern is computed as the derivative of Eq. (10) subject to $mg(\mathbf{z}_i, \mathbf{b}^{t-1})$ (cf. Eq. (6)) and is given by

$$w_t(\mathbf{z}_i) = \frac{\exp\left\{ |\mathbf{b}_{t-1}| (mg(\mathbf{z}_i, c_{t-1}) - \zeta_i^{t-1})/2 \right\}}{\sum_{j=1}^{l} \exp\left\{ |\mathbf{b}_{t-1}| (mg(\mathbf{z}_j, c_{t-1}) - \zeta_j^{t-1})/2 \right\}}. \tag{11}$$

Thus we can get an update rule for the weight of a training pattern in the t-th boosting iteration [10]

$$\begin{aligned} w_t(\mathbf{z}_i) \quad = \quad & w_{t-1}(\mathbf{z}_i) \exp\{ b_{t-1} I(y_i \neq h_{t-1}(\mathbf{x}_i)) \\ & + C\zeta_i^{t-2} |\mathbf{b}_{t-2}| - C\zeta_i^{t-1} |\mathbf{b}_{t-1}| \}. \end{aligned} \tag{12}$$

It is more difficult to compute the weight b_t of the t-th hypothesis. Especially, it is hard to derive the weight analytically. However, we can get b_t by a line search procedure over Eq. (10), which has an unique solution because $\frac{\partial}{\partial b_t} g_{reg} > 0$ is satisfied. This line search can be implemented efficiently.

It is expected that AdaBoost with this modification (in following we call it AdaBoost$_{reg}$) can avoid overfitting and will show better generalization performance.

4.2. Numerical experiments

In order to evaluate the performance of our new algorithm, we make a comparison between the single classifier, the original AdaBoost algorithm, AdaBoost$_{reg}$ (with RBF nets) and a Support Vector Machine (with RBF kernel). We use seven artificial and real world datasets from the StatLog, UCI and DELVE benchmark repositories: breast cancer[2], diabetes, german, heart, twonorm, new-thyroid (all binary classification problems).

At first we generate 100 partitions into training and test set ($\approx 60\% : 40\%$). On each partition we train the classifier and get its test set error. This performance is averaged and we get the entries in table 1.

As base hypotheses we used RBF nets as in [1] with adaptive centers (some conjugate gradient iterations to optimize positions and widths of the centers). In all experiments with AdaBoost we combined 200 hypotheses. Clearly, this number of iterations may be not optimal, however Adaboost with optimal early stopping is not better than AdaBoost$_{reg}$.

The regularization parameter C of AdaBoost$_{reg}$ and the parameters (C, σ) of the SVM are optimized on the first five training datasets. On each training set 5-fold-cross validation is used to find the best model for this dataset[3]. Finally, the model parameters are computed as the median of the five estimations. This way of estimating the parameters is surely not possible in practice, but will make this comparison more robust and the results more reliable.

Our experiments on noisy data show, that (a) the results of AdaBoost are in almost all cases worse than the single classifier (overfitting) and (b) the results of AdaBoost$_{reg}$ are in all cases (much) better than that of AdaBoost and better than that of the single classifier. The results of AdaBoost$_{reg}$ are comparable to the results of the SVM, which is known to be an excellent classifier.

Therefore, AdaBoost is useful for low noise cases, where the classes are separable (as shown for OCR[12, 7]). AdaBoost$_{reg}$ extends the applicability of boosting to "difficult separable" cases and should be applied, if the data is noisy.

Table 1: Comparison among four methods: Single RBF classifier, AdaBoost(AB), AdaBoost$_{reg}$(AB$_{reg}$) and a Support Vector Machine(SVM): estimation of generalization error (in %) on seven datasets.

	RBF	AB	**AB$_{reg}$**	SVM
cancer	26.6±4.5	30.7±4.5	**26.1±4.5**	26.0±4.7
diabetes	24.1±1.9	26.5±2.3	**23.9±1.6**	23.5±1.7
german	24.7±2.4	27.3±2.3	**25.0±2.1**	23.6 ± 2.1
heart	17.1±3.3	20.3±3.4	**16.6±3.7**	16.0±3.3
splice	9.9±1.0	10.3±0.6	**9.6±0.7**	10.7±0.7
twonorm	2.9±0.3	3.0±0.3	**2.7±0.2**	3.0±0.2
thyroid	4.5±2.1	4.4±2.2	**4.4±2.1**	4.8±2.2

5. Discussion and Conclusion

We have shown that AdaBoost perform an approximate gradient decent on a specific error function (cf. Eq.(2)), that optimizes the margin. Asymptotically all emphasis is concentrated on the difficult patterns with small margins, easy patterns effectively do not contribute to the error measure and are neglected in the training process (very much similar to support vectors).

It is shown theoretically and experimentally that the cumulative stability distribution of the training patterns in the margin area converges asymptotically to a step and the offset of the step is decided through an annealing type process governed by $|\mathbf{b}|$ and the noise level present in the data. Our numerical simulations clarify that the asymptotic margin distribution of AdaBoost is very similar to the margin distribution of a SVM, although the representation found by AdaBoost is often less sparse than for SVMs.

We also observe that the AdaBoost algorithm can overfit and get bad generalization performance for noisy data. In order to solve this problem, we propose an improved algorithm based on the analogy to the weight decay. Our numerical simulations show that the proposed method can avoid overfitting and get better generalization performance.

Our future work will concentrate on a continuing improvement of AdaBoost type algorithms for other real world applications. Also, a further analysis of the relation between AdaBoost and Support Vector Machine from margin's point of view seems promising, with particular focus on the question of what good margin distributions should look like.

Acknowledgements: We thank for valuable discussions with A. Smola, B. Schölkopf, T. Frieß and D. Schuurmans. Partial funding from EC STORM project is greatfully acknowledged.

[2] This breast cancer domain was obtained from the University Medical Centre, Inst. of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

[3] The parameters are only near-optimal, Only 10 possible values for each parameter are tested

References

[1] C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[2] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[4] L. Breiman. Prediction games and arcing algorithms. Technical Report 504, Statistics Department, University of Berkeley, December 1997.

[5] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

[6] A.J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artifical Intelligence*, 1998. To appear.

[7] Y. LeCun, L. Botou L. Jackel, H. Drucker C. Cortes, J. Denker, I. Guyon, U. Müller, E. Säckinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks*, pages 261–276, 1995.

[8] T. Onoda, G. Rätsch, and K.-R. Müller. An asymptotic analysis of adaboost in the binary classification case. In *Proceedings ICANN'98*, 1998. To appear.

[9] J. Quinlan. Boosting first-order learning. In S. Arikawa and A. Sharma, editors, *Proceedings of the 7th International Workshop on Algorithmic Learning Theory*, volume 1160 of *LNAI*, pages 143–155, Berlin, October23–25 1996. Springer.

[10] G. Rätsch. Ensemble learning methods for classification. 1998. Diploma thesis in german.

[11] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Machine Learning*, pages 148–156, 1998.

[12] H. Schwenk and Y. Bengio. Adaboosting neural networks: Application to on-line character recognition. In *Proceedings ICANN'97*, volume 1327 of *LNCS*, pages 967–972, Berlin, 1997. Springer.

[13] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.