Soft Margins for AdaBoost

Gunnar Rätsch, GMD[1] Takashi Onoda, GMD[2]
Klaus-R. Müller, GMD[3]

NeuroCOLT2 Technical Report Series

NC-TR-1998-021

August, 1998[4]

Produced as part of the ESPRIT Working Group
in Neural and Computational Learning II,
NeuroCOLT2 27150

[1]raetsch@first.gmd.de GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany
[2]onoda@criepi.denken.or.jp CRIEPI, 2-11-1, Iwado Kita, Komae-shi, Tokyo, Japan
[3]klaus@first.gmd.de GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany
[4]Received 10-AUG-1998

Abstract

Recently ensemble methods like AdaBoost were successfully applied to character recognition tasks, seemingly defying the problems of overfitting.

This paper shows that although AdaBoost rarely overfits in the low noise regime it clearly does so for higher noise levels. Central for understanding this fact is the margin distribution and we find that AdaBoost achieves – doing gradient descent in an error function with respect to the margin – asymptotically a *hard margin* distribution, i.e. the algorithm concentrates its resources on a few hard-to-learn patterns (here an interesting overlap emerge to Support Vectors). This is clearly a sub-optimal strategy in the noisy case, and regularization, i.e. a mistrust in the data, must be introduced in the algorithm to alleviate the distortions that a difficult pattern (e.g. outliers) can cause to the margin distribution. We propose several regularization methods and generalizations of the original AdaBoost algorithm to achieve a *soft margin* – a concept known from Support Vector learning. In particular we suggest (1) regularized AdaBoost$_{Reg}$ using the soft margin directly in a modified loss function and (2) regularized linear and quadratic programming (LP/QP-) AdaBoost, where the soft margin is attained by introducing slack variables.

Extensive simulations demonstrate that the proposed regularized AdaBoost-type algorithms are useful and competitive for noisy data.

Keywords: AdaBoost, Arcing, Large Margin, Hard Margin, Soft Margin, Classification, Support Vectors

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 [29, 16]. So far the reduction of the generalization error by AdaBoost has not been completely understood.

For **low noise** cases, several lines of explanation have been proposed as candidates for explaining the well functioning of Boosting methods [28, 7, 27]. Recent studies with **noisy** patterns [25, 14, 26] have shown that it is clearly a myth that Boosting methods will not overfit. In this work, we try to understand why AdaBoost exhibits virtually no overfitting for low noise and strong overfitting for high noise data. We propose improvements of AdaBoost to achieve noise robustness and to avoid overfitting.

In section 2 we analyze AdaBoost asymptotically. Due to their similarity, we will refer in the following to *AdaBoost* [10] and *unnormalized Arcing* [8] (with exponential function) as *AdaBoost-type algorithms* (ATA). We especially have a focus on the error function of ATAs and find that the function can be written in terms of the margin and every iteration of AdaBoost tries to minimize this function stepwise by maximizing the margin [22, 26, 12]. From the asymptotical analysis of this function, we can introduce the *hard margin* concept. We show connections to Vapnik's maximum margin classifiers, to Support Vector (SV) learning [4] and to linear programming (LP). Bounds on the size of the margin are given.

Noisy patterns have shown that AdaBoost can overfit: this holds for boosted decision trees [25], RBF nets [26] and also other kinds of classifiers. In section 3 we explain why the property of AdaBoost to enforce a hard

margin must necessarily lead to overfitting in the presence of noise or in the case of overlapping class distributions.

Because the hard margin plays a central role in causing overfitting, we propose to relax the hard margin in section 4 and allow for misclassifications by using the *soft margin* concept, that has already been successfully applied to Support Vector Machines (cf. [9]). Our view is that the margin concept is the key for the understanding of both, SVMs and ATAs. So far, we only know how a margin distribution should look like, that a learner has to achieve for optimal classification in the no-noise case: then a large hard margin is clearly the best choice [30]. However, for noisy data there is always the trade-off between believing in the data or mistrusting it, as the very data point could be mislabeled or an outlier. This leads to the introduction of regularization, which reflects the prior knowledge that we have about the problem. We will introduce a regularization strategy (analogous to weight decay) into AdaBoost and subsequently extend the LP-AdaBoost algorithm of Grove & Schuurmans [14] by slack variables to achieve soft margins. Furthermore, we propose QP-AdaBoost and show its connections to SVMs.

Finally, in section 5 numerical experiments on several artificial and real-world data sets show the validity and competitiveness of our regularization approach. The paper is concluded by a brief discussion.

2 Analysis of AdaBoost's Learning Process

2.1 Algorithm

Let $\{h_t(\mathbf{x}) : t = 1, \ldots, T\}$ be an ensemble of T hypotheses defined on an input vector $\mathbf{x} \in X$ and $\mathbf{c} = [c_1 \ldots c_T]$ their weights satisfying $c_t \geq 0$ and $\sum_{t=1}^{T} c_t = 1$. We will consider only the binary classification case; most results can be transfered easily to the classification with more than two classes [27]. 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 $\overline{f}(\mathbf{x}) \equiv \overline{f}_T(\mathbf{x})$ which is the weighted majority of the votes, where

$$f_T(\mathbf{x}) := \sum_{t=1}^{T} c_t h_t(\mathbf{x}) \qquad \text{and} \qquad \overline{f}_T(\mathbf{x}) := \operatorname{sign}(f_T(\mathbf{x})) \ . \tag{1}$$

In order to train the ensemble, i.e. to find T appropriate hypotheses $\{h_t(\mathbf{x})\}$ and the weights $\mathbf{c}$ for the convex combination, several algorithms have been proposed: e.g. Windowing [24], Bagging [5] and Boosting/Arcing (AdaBoost [10], ArcX4 [7]). Bagging, where the weighting is simply $c_t = 1/T$, and Boosting/Arcing, where the weighting scheme is more complicated, are the most well-known ensemble learning algorithms. In the sequel, we will focus on Boosting/Arcing, i.e. AdaBoost-type algorithms. We omit a detailed description of the ATA and give only the pseudo-code in figure 1, for details see e.g. [10, 7, 8].

In the binary classification case, we can define the margin for an input-output pair $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ by

$$\operatorname{mg}(\mathbf{z}_i, \mathbf{c}^t) = y_i f_t(\mathbf{x}_i) = y_i \sum_{r=1}^{t} c_r^t h_r(\mathbf{x}_i), \tag{8}$$

Algorithm AdaBoost(ϕ)

Input: l examples $\mathbf{Z} = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l) \rangle$

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

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

1. Train neural network with respect to the weighted sample set $\{\mathbf{Z}, \mathbf{w}\}$ and
   obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$
2. Calculate the training error ϵ_t of h_t:

$$\epsilon_t = \sum_{i=1}^{l} w_t(\mathbf{z}_i) I(h_t(\mathbf{x}_i) \neq y_i) \ , \tag{2}$$

   abort if

$$\epsilon_t = 0 \qquad \text{or} \qquad \epsilon_t \geq \phi - \Delta \ , \tag{3}$$

   where Δ is a small constant
3. Set

$$b_t = \log \frac{\epsilon_t (1 - \phi)}{\phi (1 - \epsilon_t)}. \tag{4}$$

4. Update weights $\mathbf{w}_t$:

$$w_{t+1}(\mathbf{z}_i) = w_t(\mathbf{z}_i) \exp\{-b_t I(h_t(\mathbf{x}_i) = y_j)\} / Z_t \ , \tag{5}$$

   where Z_t is a normalization constant, such that $\sum_{i=1}^{l} w_{t+1}(\mathbf{z}_i) = 1$.

Output: Final hypothesis

$$f(\mathbf{x}) = \sum_{t=1}^{T} c_t h_t(\mathbf{x}), \tag{6}$$

where

$$c_t := \frac{b_t}{|\mathbf{b}|} \qquad \text{and} \qquad |\mathbf{b}| := \sum_{t=1}^{T} |b_t| \tag{7}$$

Figure 1 The AdaBoost-type algorithm (ATA) [22]. For $\phi = \frac{1}{2}$, we retrieve the original AdaBoost algorithm [10]. ATA is a specialization of unnormalized Arcing [6] (with exponential function).

where $i = 1, \ldots, l$ and l denotes the number of training patterns. The margin at $\mathbf{z}$ is positive, if the right class of the pattern is predicted. As the positivity of the margin value increases, the decision correctness, i.e. decision stability, becomes larger. Moreover, if $\sum_{r=1}^{t} c_r =: |\mathbf{c}^t| = 1$, then $\mathrm{mg}(\mathbf{z}_i, \mathbf{c}^t) \in [-1, 1]$. The margin ρ of a decision line is the smallest margin of a pattern in the training set, i.e.

$$\rho = \min_{i=1,\ldots,l} \mathrm{mg}(\mathbf{z}_i) \ .$$

We define $d(\mathbf{z}_i, \mathbf{c})$ as the rate of incorrect classification (cf. *edge* in Breiman [6]) for one pattern by

$$d(\mathbf{z}_i, \mathbf{c}^t) \quad = \quad \sum_{r=1}^{t} c_r^t I(y_i \neq h_r(\mathbf{x}_i)) = \frac{1}{2}(1 - \mathrm{mg}(\mathbf{z}_i, \mathbf{c}^t)), \tag{9}$$

where $I(true) = 1$ and $I(false)$=0. We will also use this definition with $\mathbf{b}$ (instead of $\mathbf{c}$) which is just an unnormalized version of $\mathbf{c}$, i.e. usually we have $|\mathbf{b}| \neq 1$ (cf. (4) and (7) in figure 1).

2.2 Error Function of AdaBoost

An important question in the analysis of ATAs is what kind of error function is optimized. From the algorithmic formulation (cf. figure 1), it is not straight forward to understand what the aim of this algorithm is. So to consider why one should use the weights of the hypotheses c_t and of the patterns $w_t(\mathbf{z}_i)$ in the manner of equation (4) and (5), let us remember the following facts:

1. The weights $w_t(\mathbf{z}_i)$ in the t-th iteration are chosen such that the previous hypothesis has exactly a weighted training error ϵ of $1/2$ [28].
2. The weight c_t of a hypothesis is chosen such that it minimizes a functional G introduced in Breiman [8]. Essentially, this functional depends on the rate of incorrect classification of all patterns and is defined by

   $$G(b_t, \mathbf{b}^{t-1}) = \sum_{i=1}^{l} \exp\left\{d(\mathbf{z}_i, \mathbf{b}^t) - \phi|\mathbf{b}^t|\right\} \ , \tag{10}$$

   where ϕ is a constant. This functional can be minimized analytically [8, 26] and one gets the explicit form of equation (4) as solution of $\frac{\partial G(\mathbf{b}^t)}{\partial b_t} = 0$.
3. To train the t-th hypothesis (step 1 in figure 1) we can either use bootstrap replicates of the training set (sampled according to $\mathbf{w}_t$) or minimize a weighted error function of the base learning algorithm. We observed that the convergence of the ATA is faster if the weighted error function is used.

Taking a closer look at the definition of G, one finds that the computation of the sample distribution $\mathbf{w}_t$ (cf. equation (5)) can be derived directly from G. Let us assume G is the error function which is minimized by the ATA. Essentially, G defines a loss function over all margin distributions,

which depends on the value of $|\mathbf{b}|$. The larger the margins $\mathrm{mg}(\mathbf{z}_i)$ (i.e. the smaller the rate of incorrect classification), the smaller the value of G.

The gradient $\frac{\partial G}{\partial \mathrm{mg}(\mathbf{z}_i)}$ gives an answer to the question, which pattern should increase its margin to decrease G maximimally (gradient descent). This information can be used to compute a sample distribution $\mathbf{w}_t$ for training the next hypothesis h_t. If it is important to increase the margin of a pattern $\mathbf{z}_i$, the weight $w_t(\mathbf{z}_i)$ should be high – otherwise low (because the distribution $\mathbf{w}_t$ sums to one). Surprisingly, this is exactly what ATAs are doing.

Lemma 1 *The computation of the pattern distribution* $\mathbf{w}_{t+1}$ *in the* t*-th iteration is equivalent to normalizing the gradient of* $G(b_{t+1}, \mathbf{b}^t)$ *with respect to* $\mathrm{mg}(\mathbf{z}_i, \mathbf{b}^t)$*, i.e.*

$$w_{t+1}(\mathbf{z}_i) = \frac{\partial G(b_{t+1}, \mathbf{b}^t)}{\partial \mathrm{mg}(\mathbf{z}_i, \mathbf{b}^t)} \Big/ \sum_{j=1}^{l} \frac{\partial G(b_{t+1}, \mathbf{b}^t)}{\partial \mathrm{mg}(\mathbf{z}_j, \mathbf{b}^t)} \,. \tag{11}$$

The proof can found in the appendix A.

From Lemma 1, the analogy to a gradient descent method is (almost) complete. In a gradient descent, at first we compute the gradient of the error function with respect to the parameters which are to be optimized. This corresponds to computing the gradient of G with respect to the margins. At second, the step size in this direction is determined (usually by a line-search). This is comparable to the minimization of G mentioned in point 2 in the list above.

Therefore, AdaBoost can be related to a gradient descent method, which aims to minimize the functional G by constructing an ensemble of classifiers [22, 26, 12]. This also explains point 1 in the list, because in a gradient descent method, the new search direction is perpendicular to the previous one.

But the analogy is not perfect. There is a "gap" between having the pattern distribution and having a classifier. It is difficult to find a classifier which minimizes G by only knowing the pattern distribution [26].

As we mentioned above, there are two ways of incorporating the sample distribution. The first way is to create a bootstrap replicate, which is sampled according to the pattern distribution. Usually there are a lot of random effects, which hide the "true" information contained in the distribution. Therefore, some information is lost – the gap is larger. The more direct way is to use a weighted error function and employ weighted minimization (Breiman [8]), therefore we will need more iterations with bootstrap than with weighted minimization[1]. The fastest convergence can be obtained, if one uses G directly for finding the hypothesis (cf. [12]). These considerations explain point 3 in the list.

[1] In Friedman et al. [12] it was mentioned, that sometimes the randomized version shows a better performance, than the version with weighted minimization. In connection with the discussion in section 3 this becomes clearer, because the randomized version will show an overfitting effect (possibly much) later and overfitting maybe not observed, whereas it was observed using the more efficient weighted minimization.

2.3 AdaBoost as an Annealing Process

From definition of G and d, equation (11) can also be written as

$$w_{t+1}(\mathbf{z}_i) = \frac{\exp\left(-\frac{1}{2}\mathrm{mg}(\mathbf{z}_i, \mathbf{c}^t)\right)^{|\mathbf{b}^t|}}{\sum_{j=1}^{l} \exp\left(-\frac{1}{2}\mathrm{mg}(\mathbf{z}_j, \mathbf{c}^t)\right)^{|\mathbf{b}^t|}} \ . \tag{12}$$

Inspecting this equation more closely, we see that AdaBoost uses a soft-max function [3] with parameter $|\mathbf{b}|$ that we would like to interpret as an annealing parameter [22]. If the temperature $\vartheta := 1/|\mathbf{b}|$ is high, the system is in a state with high energy – all patterns have relevant high weights. If the temperature goes down, the patterns with smallest margin will get higher and higher weights. In the limit, we arrive at the maximum function. Only the pattern(s) with the highest rate of incorrectness d (i.e. smallest margin) will be considered and get a non-zero weight.

Lemma 2 *If, in the learning process of an ATA with $0 < \phi < 1$, all weighted training errors ϵ_t are bounded by $\epsilon_t \leq \phi - \Delta$ ($0 < \Delta < \phi$ fixed), then $|\mathbf{b}|$ increases at least linearly with the number of iterations t.*

Proof 3 *With (4), the smallest value for b_t is achieved, if $\epsilon_t = \phi - \Delta$. Then we have $b_t = \log \frac{q}{q-\Delta}$, where $q := \phi(1 - \phi + \Delta)$. We also have $q > \Delta > 0$, because from $\phi(1-\phi) > \Delta(1-\phi)$ we get $\phi(1-\phi+\Delta) > \Delta$ and hence also $q - \Delta > 0$. Therefore, the smallest value of b_t is $\log \frac{q}{q-\Delta}$, which is always bigger than a constant γ, which only depends on ϕ and Δ. Thus, we have $|\mathbf{b}^t| > t\gamma$.* ■

If the annealing speed is low, the achieved solution should have larger margins. The reason is the same as for an usual annealing process [15]: in the error surface, a better local minimum could be obtained locally, if the annealing is slow enough. From equation (4), 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 ATA iterations. The asymptotic point is reached faster. To reduce the annealing speed, ϕ or the complexity of the base hypotheses has to be decreased (with the constraint $\epsilon_t < \phi - \Delta$).

Figure 2 shows some error functions for classification. Among them, the ATA error function for different values of $|\mathbf{b}|$ and ϕ is shown. In figure 2 (left), the AdaBoost Error ($\phi = \frac{1}{2}$), the Squared Error $(y - f(x))^2$ and the Kullback-Leibler Error $\ln \mathrm{mg}(\mathbf{z}) / \ln 2$ are plotted. Squared and Kullback-Leibler Error are very similar to the error function of AdaBoost for $|\mathbf{b}| = 3$. As $|\mathbf{b}|$ increases (in our experiments often up to 10^2 after 200 iterations), the ATA error function approximates a $0/\infty$ loss: all patterns with margin smaller than 0 (or more general $1 - 2\phi$) get loss ∞; all others have loss 0. If it is possible to reduce the error of all patterns to 0 (as in the AdaBoost case), then this is asymptotically ($|\mathbf{b}| \to \infty$) equivalent to the 0/1-loss around 0 ($1 - 2\phi = 0$ for original AdaBoost with $\phi = \frac{1}{2}$). The loss function for $\phi \neq \frac{1}{2}$, shown in figure 2 (right), demonstrate the different offsets of the step exhibited by the 0/1 loss.

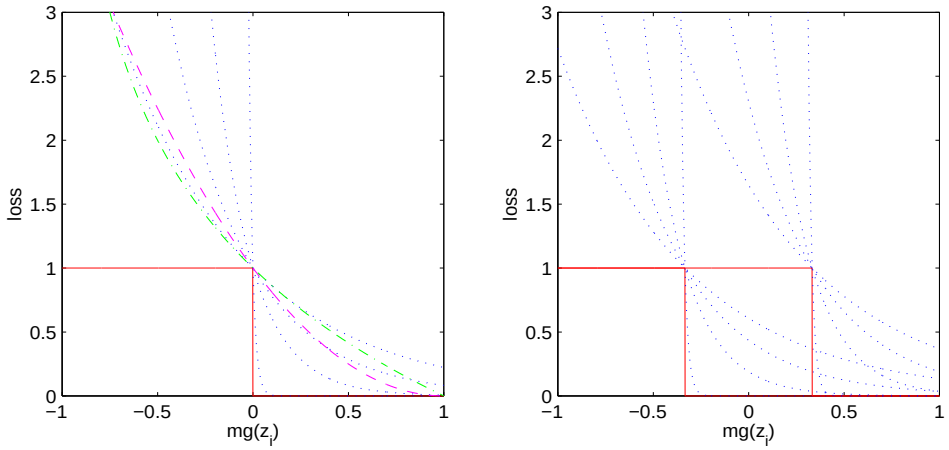

Figure 2 Loss functions for estimating a function $f(x)$ for classification. The abscissa shows the margin $yf(x)$ of a pattern and the y-coordinate shows the monotone loss for that pattern: 0/1-Loss (solid), Squared Error (dashed), Kullback-Leibler Error (dash-dotted) and (10) for $|\mathbf{b}| = \{3, 5, 10, 100\}$. On the left panel is $\phi = 1/2$ and on the right plot ϕ is one out of $\{1/3, 2/3\}$. ϕ controls the position of the step of the 0/1-Loss ($\rho = 1 - 2\phi$), which is asymptotically approximated by the AdaBoost loss function.

2.4 Asymptotical Analysis

2.4.1 How large is the Margin?

The main point in the explanation of ATA's good generalization performance is the size of the (hard) margin that can be achieved [28, 8]. In the low noise case, the hypothesis with the largest margin will have a good generalization performance [30, 28]. Thus, it is interesting to see how large the margin is and on what it is depending.
Generalizing theorem 5 of Freund et al. [10] to the case $\phi \neq \frac{1}{2}$ we get

Theorem 4 *Assume, $\epsilon_1, \ldots, \epsilon_T$ are the weighted classification errors of $h_1, \ldots, h_T$, which were generated while running an ATA and $1 > \phi > \max_{t=1,\ldots,T} \epsilon_t$. Then the following inequality holds for all $\theta \in [-1, 1]$:*

$$\frac{1}{l}\sum_{i=1}^{l} I\left(y_i f(\mathbf{x}_i) \leq \theta\right) \leq \left(\varphi^{\frac{1+\theta}{2}} + \varphi^{-\frac{1-\theta}{2}}\right)^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}} \;, \quad (13)$$

where f is the final hypothesis and $\varphi = \frac{\phi}{1-\phi}$.

The proof can be found in appendix B.

Corollary 5 *An ATA will asymptotically ($t \to \infty$) generate margin distributions with a margin ρ, which is bounded by*

$$\rho \geq \frac{\ln\left(\phi\epsilon^{-1}\right) + \ln\left((1-\phi)(1-\epsilon)^{-1}\right)}{\ln\left(\phi\epsilon^{-1}\right) - \ln\left((1-\phi)(1-\epsilon)^{-1}\right)} \;, \quad (14)$$

where $\epsilon = \max_t \epsilon_t$, if $\epsilon \leq (1-\rho)/2$ is satisfied.

Proof 6 *The maximum of* $\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}$ *with respect to* ϵ_t *is reached for* $\frac{1}{2}(1-\theta)$ *and for* $0 \leq \epsilon_t \leq \frac{1}{2}(1-\theta)$ *it is increasing monotonically in* ϵ_t. *Therefore, we can replace* ϵ_t *by* ϵ *in equation* (13) *for* $\theta \leq \rho$:

$$\mathbf{P}_{(\mathbf{x},y)\sim\mathbf{Z}}[yf(\mathbf{x}) \leq \theta] \leq \left(\left(\varphi^{\frac{1+\theta}{2}} + \varphi^{-\frac{1-\theta}{2}}\right)\epsilon^{\frac{1-\theta}{2}}(1-\epsilon)^{\frac{1+\theta}{2}}\right)^T \ .$$

If the basis on the right hand side is smaller than 1, *then asymptotically we have* $\mathbf{P}_{(\mathbf{x},y)\sim\mathbf{Z}}[yf(\mathbf{x}) \leq \theta] = 0$. *Asymptotically, there is no example that has a smaller margin than* θ. *For the biggest possible margin* $\theta_{\max}$ *we have*

$$\left(\varphi^{\frac{1+\theta_{\max}}{2}} + \varphi^{-\frac{1-\theta_{\max}}{2}}\right)\epsilon^{\frac{1-\theta_{\max}}{2}}(1-\epsilon)^{\frac{1+\theta_{\max}}{2}} = 1 \ .$$

We can solve this equation for $\theta_{\max}$ *and we get*

$$\theta_{\max} = \frac{\ln\left(\phi\epsilon^{-1}\right) + \ln\left((1-\phi)(1-\epsilon)^{-1}\right)}{\ln\left(\phi\epsilon^{-1}\right) - \ln\left((1-\phi)(1-\epsilon)^{-1}\right)} \ .$$

We get the assertion, because ρ *is always bigger or equal* $\theta_{\max}$. ■

From equation (14), we can see the interaction between ϕ and ϵ: if the difference of ϵ and ϕ is small, the right hand side of (14) is small. The smaller ϕ the more important is this difference. From theorem 7.2 of [8] we also have the weaker bound $\rho \geq 1-2\phi$ and so, if ϕ is small then ρ must be large, i.e. choosing a small ϕ results in a larger margin on the training patterns. An increase of the complexity of the basis algorithm leads to an increased ρ, because the error ϵ_t will decrease.

2.4.2 Support Patterns

A decrease in the functional $G(\mathbf{c}, |\mathbf{b}|) := G(\mathbf{b})$ (with $\mathbf{c} = \mathbf{b}/|\mathbf{b}|$) is predominantly achieved by improvements of the margin $\mathrm{mg}(\mathbf{z}_i, \mathbf{c})$. If the margin $\mathrm{mg}(\mathbf{z}_i, \mathbf{c})$ 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 (cf. Lemma 2). In this case, the values of all $\mathrm{mg}(\mathbf{z}_i, \mathbf{c}), i = 1, \cdots, l$, are almost the same, small differences are amplified strongly in $G(\mathbf{c}, |\mathbf{b}|)$.

For example, when the margin $\mathrm{mg}(\mathbf{z}_i, \mathbf{c}) = 0.2$ and another margin $\mathrm{mg}(\mathbf{z}_j, \mathbf{c}) = 0.3$ and $|\mathbf{b}|$ is 10^2, the difference is amplified to the difference between $\exp\left\{-\frac{10^2 \times 0.2}{2}\right\} = e^{-10}$ and $\exp\left\{-\frac{10^2 \times 0.3}{2}\right\} = e^{-15}$ in $G(\mathbf{c}, |\mathbf{b}|)$, i.e. to a factor of $e^5 \approx 150$.

Obviously the function $G(\mathbf{c}, |\mathbf{b}|)$ is asymptotically very sensitive to small differences between margins of the training patterns. From equation (12), when the annealing parameter $|\mathbf{b}|$ takes a 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. Therefore, the margins $\mathrm{mg}(\mathbf{z}_i, \mathbf{c})$ are expected to asymptotically converge to a fixed value ρ and a subset of the training patterns will asymptotically have the same smallest margin ρ. We call these patterns *Support Patterns* (cf. figure 3).

In order to confirm that the above theoretical analysis is correct, asymptotic numerical simulations on toy data (several Gauss blobs in two dimensions; cf. figure 4) are made. The training data is generated from

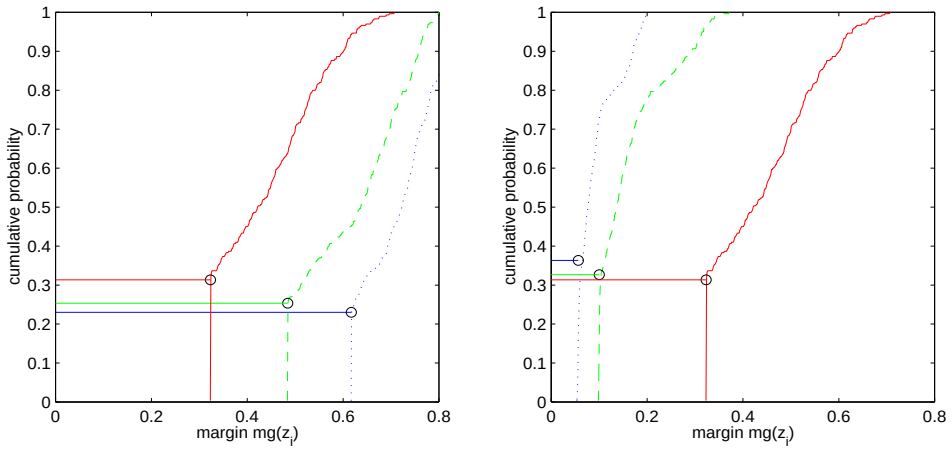

Figure 3 Margin distributions for AdaBoost for different noise levels $\sigma^2 = 0\%$ (dotted), 9% (dashed), 16% (solid) with RBF nets (13 centers) as base hypotheses (left) and with 7 (dotted), 13 (dashed), 30 (solid) centers in the base hypotheses for data with $\sigma^2 = 16\%$ (right) after 10^4 AdaBoost iterations. These graphs experimentally confirm the expected trends from equation (14).

several (nonlinearly transformed) Gaussian and uniform blobs[2], which are additionally disturbed by uniformly distributed noise $U(0.0, \sigma^2)$. In our simulations, we used 300 patterns and σ^2 is one out of 0%, 9%, and 16%.

In all simulations, radial basis function (RBF) networks with adaptive centers are used as learners (cf. Appendix C for a detailed description). Figure 3 shows the margin distributions after 10^4 AdaBoost iterations at different noise levels 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 size of the margin for some training patterns. From figure 3, one can see the influence of noise in the data and the strength of the base hypotheses on the margin ρ. If the noise level is high or the complexity is low, one gets higher training errors ϵ_t and therefore a smaller value of ρ. These numerical results support our theoretical asymptotic analysis.

Interestingly, the margin distributions of ATAs resembles the one of Support Vector Machines (SVMs) for the separable case [4, 9, 30] (cf. figure 6). In an example (cf. figure 4) almost all patterns, that are Support Vectors (SVs), also lie within the step part of the margin distribution for AdaBoost. So, AdaBoost achieves a *hard margin* asymptotically, such as the SVMs for the separable case.

In an earlier study [26] we observed, that usually there is high overlap among the Support Vectors and Support Patterns. Intuitively this is clear, because the most difficult patterns are in the margin area. They are emphasized strongly and become Support Patterns or Support Vectors asymptotically. The degree of overlap depends on the kernel (SVM) and on the base hypothesis (ATA) which are used. For the SVM with RBF kernel the highest overlap was achieved, when the average widths of the RBF networks was used as kernel width for the Support Vector Ma-

[2] A detailed description of the generation of the toy data used in the asymptotical simulations can be found in the Internet http://www.first.gmd.de/~raetsch/data/banana.txt.

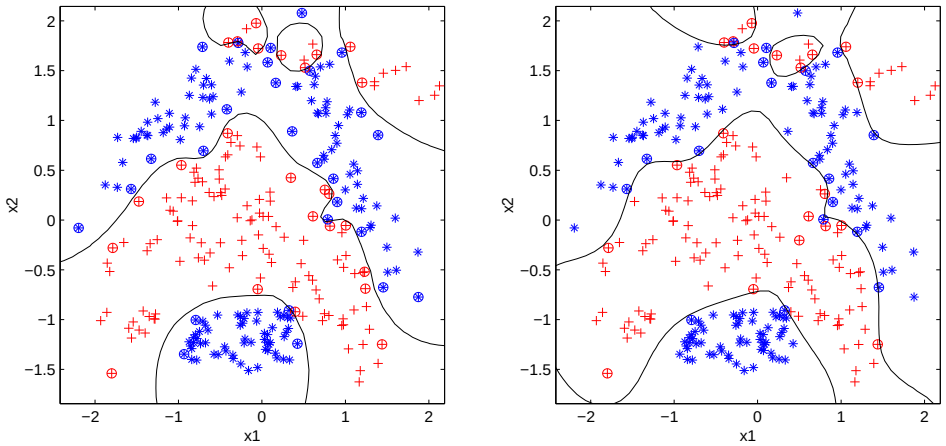

Figure 4 Training patterns with decision lines for AdaBoost (left) with RBF nets (13 centers) and SVM (right) for a low noise case with similar generalization errors. The positive and negative training patterns are shown as '+' and '∗' respectively, the Support Patterns and Support Vectors are marked with 'o'.

chine [26]. We have observed the similarity of Support Patterns (SP) of AdaBoost and SV of the SVM also in several other applications.

In the sequel, we can often assume the asymptotical case, where a hard margin is achieved. The more hypotheses we combine, the better is this approximation. And indeed, if for example $|\mathbf{b}| > 10^2$ (as often after 200 AdaBoost iterations on benchmark data used in section 5), already then the approximation to a hard margin is good (cf. equation (12)). This is illustrated by figure 5, which shows typical ATA margin distributions after $20, 50, 200$ and 10^4 iterations.

To recapitulate our findings of this section:

1. AdaBoost-type algorithms aim to minimize a functional, which depends on the margin distribution. The minimization is done through an approximate gradient descent with respect to the margin (cf. [26, 12]).
2. Annealing is a part of the algorithm. It depends on an annealing parameter $|\mathbf{b}|$, which controls how good the 0/1-loss (around $1 - 2\phi$) is approximated. The size of the margin is decided by a certain annealing process. The speed of annealing depends on the parameter ϕ and is an implicit function of the strength of the learner in the training process.
3. Some training patterns, which are in the area of the decision boundary, have asymptotically the same margin. We call these patterns Support Patterns. They have a large overlap to the SVs found by a SVM.
4. Asymptotically, a hard margin is achieved, which is comparable to the one of the original SV approach [4].
5. Larger hard margins can be achieved, if ϵ_t and/or ϕ are small (cf. Corollary 5). For the low noise case, a choice of $\theta \neq \frac{1}{2}$ can lead to a better generalization performance, as shown for OCR in [22].

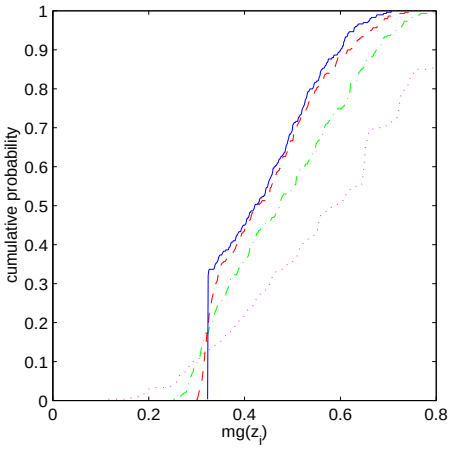

Figure 5 Typical margin distribution graphs of (original) AdaBoost after 20 (dotted), 70 (dash-dotted), 200 (dashed) and 10^4 (solid) iterations. Here, the toy example (300 patterns, $\sigma = 16\%$) and RBF networks with 30 centers are used. After already 200 iterations the asymptotical convergence is almost reached.

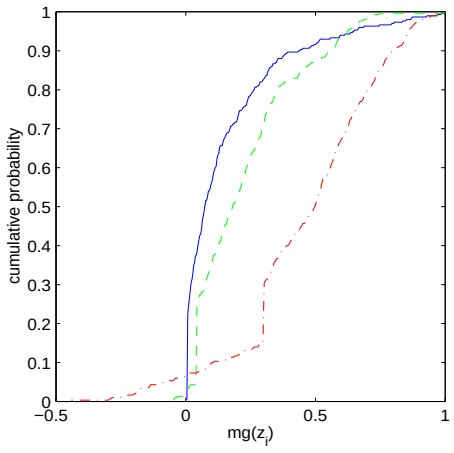

Figure 6 Typical margin distribution graphs (normalized) of a SVM with hard margin (solid) and soft margin with $C = 10^{-3}$ (dashed) and $C = 10^{-1}$ (dash-dotted). Here, the same toy example and a RBF kernel (width=0.3) is used. The generalization error of the SVM with hard margin is more than two times larger as with $C = 10^{-1}$.

3 Hard Margin and Overfitting

In this section, we give reasons why the ATA is *not noise robust* and exhibits suboptimal generalization ability in the presence of noise. We give several references and examples why the hard margin approach will fail in general if noise is present. According to our understanding, noisy data has at least one of the following properties: (a) overlapping class probability distributions, (b) outliers and (c) mislabeled patterns. All three kinds of noise appear very often in data analysis. Therefore the development of a noise robust version of AdaBoost is very important.

The first theoretical analysis of AdaBoost in connection with margin distributions was done by Schapire et al. [28]. Their main result is a bound on the generalization error $\mathbf{P}_{\mathbf{z}\sim\mathcal{D}}[\mathrm{mg}(\mathbf{z}) \leq 0]$ depending on the VC-dimension d of the base hypotheses class and on the margin distribution on the training set. With probability at least $1 - \delta$

$$\mathbf{P}_{\mathbf{z}\sim\mathcal{D}}[\mathrm{mg}(\mathbf{z}) \leq 0] \leq \mathbf{P}_{\mathbf{z}\sim\mathbf{Z}}[\mathrm{mg}(\mathbf{z}) \leq \theta] + \mathcal{O}\left(\frac{1}{\sqrt{l}}\left(\frac{d\log^2(l/d)}{\theta^2} + \log(1/\delta)\right)\right) \tag{15}$$

is satisfied, where $\theta > 0$ and l denotes the number of patterns. It was stated that the reason for the success of AdaBoost, compared to other ensemble learning methods (e.g. Bagging), is the maximization of the margin. They experimentally observed that AdaBoost maximizes the margin of patterns which are most difficult, i.e. have the smallest margin. However, by increasing the minimum margin of a few patterns, AdaBoost also reduces the margin of the rest of the other patterns.

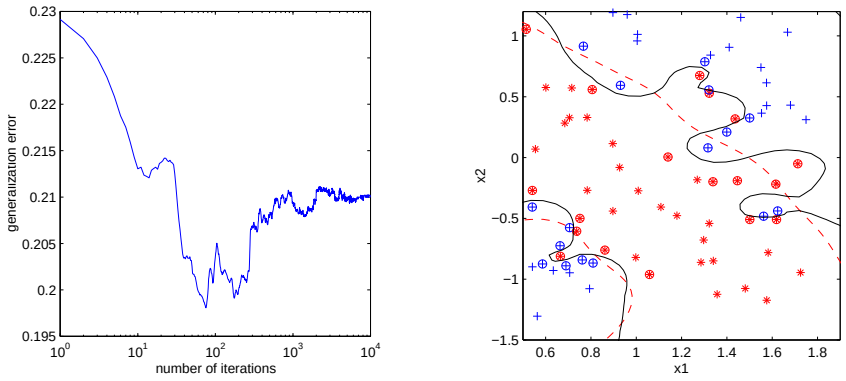

Figure 7 Typical overfitting behavior in the generalization error (smoothed) as a function of the number of iterations (left) and a typical decision line (right) generated by AdaBoost (10^4 iterations) using RBF networks (30 centers) in the case of noisy data (300 patterns, $\sigma^2 = 16\%$). The positive and negative training patterns are shown as '+' and '$*$' respectively, the Support Patterns are marked with 'o'. An approximation to the Bayes decision line is plotted dashed.

In Breiman [8], the connection between the smallest margin and the generalization error was analyzed experimentally and could not be confirmed on noisy data.

In Grove et al. [14] the Linear Programming (LP) approach of Freund et al. [11] and Breiman [8] was extended and used to maximize the smallest margin of an existing ensemble of classifiers. Several experiments with LP-AdaBoost on UCI benchmarks (often noisy data) were made and it was unexpectedly observed, that LP-AdaBoost performs in almost all cases worse than the original AdaBoost algorithm, even if the smallest margins are larger.

Our experiments have shown that as the margin increases, the generalization performance becomes better on datasets with almost no noise [22] (e.g. OCR), however, on noisy data, we also observed that AdaBoost overfits (for a moderate number of combined hypotheses).

As an example for overlapping classes, figure 7 (left) shows a typical overfitting behavior in the generalization error for AdaBoost on the same data as in section 2. Here, already after only 80 AdaBoost iterations the best generalization performance is achieved. From equation (14) it is clear that AdaBoost will asymptotically achieve a positive margin (if $\phi < \frac{1}{2}$) and all training patterns are classified according to their possibly wrong labels (cf. figure 7 (right)), because the complexity of the combined hypotheses increases more and more. The achieved decision line is far away from the Bayes optimal line (cf. dashed line in figure 7 (right)).

To discuss the bad performance of a hard margin classifier in presence of outliers and mislabeled patterns, we analyze the toy example in figure 8. Let us first consider the case without noise (left). Here, we can estimate the optimal separating hyper-plane correctly. In figure 8 (middle) we have an outlier, which corrupts the estimation. AdaBoost will certainly concentrate its weights to this outlier and spoil the good estimate that we would get without outlier. Next, let us consider more complex decision lines. Here the overfitting problem gets even more distinct, if we can gen-

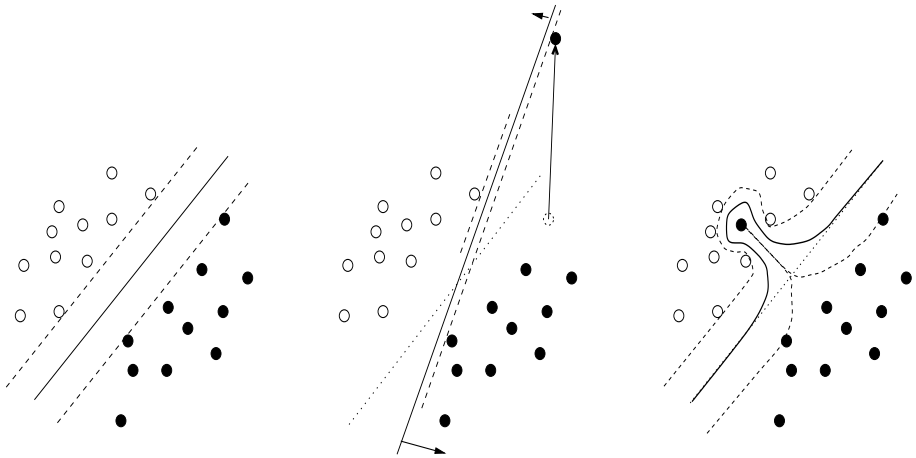

Figure 8 The problem of finding a maximum margin "hyper-plane" on reliable data (left), data with outlier (middle) and with a mislabeled pattern (right). The solid line shows the resulting decision line, whereas the dashed line marks the margin area. In the middle and on the left the original decision line is plotted with dots. The hard margin implies noise sensitivity, because only one pattern can spoil the whole estimation of the decision line.

erate more and more complexity through combining a lot of hypotheses. Then all training patterns even mislabeled ones or outliers can be classified correctly. In figure 7 (right) and figure 8 (right) we see that the decision surface is much too shaky and gives a bad generalization.

From these cartoons, it becomes apparent that AdaBoost is noise sensitive and maximizing the smallest margin in the case of noisy data can (and will) lead to bad results. Therefore, we need to allow for a possibility of mistrusting the data.

From the bound (15) it is indeed not obvious, that we should minimize the smallest margin: the first term on the right hand side of equation (15) takes the whole margin distribution into account. If we would allow a non-zero training error in the settings of figure 8, then the first term of the right hand side of (15) becomes non-zero ($\theta > 0$). But then θ can be larger, such that the second term is much smaller. In Mason et al. [18] a similar bound was used to optimize the margin distribution (a piecewise linear approximation) directly. This approach was more successful on noisy data than a maximization of the smallest margin.

In the following we introduce the possibility to mistrust parts of the data, which leads to the *soft margin* concept.

4 Improvements using a Soft Margin

The original SV algorithm [4] had similar problems as the ATA with respect to hard margins. In the SV approach, training errors on data with overlapping classes were not allowed and the generalization performance was poor on noisy data. The introduction of soft margins then gave a new algorithm, which achieved much better results compared to the original algorithm [9] (cf. figure 6).

In the sequel, we will show how to use the soft margin idea for ATAs. In section 4.1 we change the error function (10) by introducing a new term, which controls the importance of a pattern compared to its achieved margin. In section 4.2 we show how the soft margin idea can be built into the LP-AdaBoost algorithm and in section 4.3 we show an extension to quadratic programming – QP-AdaBoost – with its connections to the Support Vector approach.

4.1 Margin vs. Influence of a Pattern

First, we propose an improvement of the original AdaBoost by using an regularization term in (10) in analogy to weight decay. For this we define the *influence* of a pattern to the combined hypotheses h_r by

$$\mu_t(\mathbf{z}_i) := \sum_{r=1}^{t} c_r w_r(\mathbf{z}_i) \ ,$$

which is the (weighted) average weight of a pattern computed during the ATA's learning process (cf. pseudo-code in figure 1). A pattern which is very often misclassified (i.e. difficult to classify), will have a high average weight (high influence). The definition of the influence clearly depends on the base hypotheses space $\mathcal{H}$.

From Corollary 5 and Theorem 2 of [8], all training patterns will get a margin $\mathrm{mg}(\mathbf{z}_i)$ larger or equal than $1-2\phi$ after many iterations (cf. figure 2 and discussion in section 2). Asymptotically, we get the following inequalities

$$\mathrm{mg}(\mathbf{z}_i, \mathbf{c}) \geq \rho \qquad \text{for all } i = 1, \ldots, l, \tag{16}$$

where $\rho \geq 1-2\phi$ (or even better bounded by equation (14)). We can see the relation between ρ and $G(\mathbf{b})$ for a sufficient large value of $|\mathbf{b}|$ in equation (10): as $G(\mathbf{b})$ is minimized, ρ is maximized. After many iterations, these inequalities are satisfied and as long as $\phi \leq \frac{1}{2}$, the hard margin $\rho \geq 0$ is achieved [28], what will lead to overfitting in the case of noise. In the following we will consider only the case $\phi = \frac{1}{2}$ – generalizations are straight forward.

We define a *soft margin* of a pattern $\widetilde{\mathrm{mg}}(\mathbf{z}_i)$ as a trade-off between the margin and the influence of the pattern to the final hypothesis as follows

$$\widetilde{\mathrm{mg}}(\mathbf{z}_i, \mathbf{c}^t) := \mathrm{mg}(\mathbf{z}_i, \mathbf{c}^t) + C\mu_t(\mathbf{z}_i)^p \ , \tag{17}$$

where $C \geq 0$ is a fixed constant and p a fixed exponent. With C and p one can modify this trade-off. We can reformulate AdaBoost's optimization process in terms of soft margins. With (16) and (17) we get

$$\widetilde{\mathrm{mg}}(\mathbf{z}_i, \mathbf{c}) \geq \rho \ , \tag{18}$$

which is equivalent to

$$\mathrm{mg}(\mathbf{z}_i, \mathbf{c}) \geq \rho - C\zeta^t(\mathbf{z}_i) \ , \tag{19}$$

where we use $\zeta^t(\mathbf{z}_i) := \mu_t(\mathbf{z}_i)^p$, for simplicity. Other functional forms of ζ can also be used (depending on our prior).

In these inequalities, $\zeta^t(\mathbf{z}_i)$ are positive and if a training pattern has high weights, $\zeta^t(\mathbf{z}_i)$ is increasing. In this way, we do not force outliers to

be classified according to their possibly wrong labels (if this would imply a high influence), but we allow for some errors. So we get the desired trade-off between margin and influence. If we choose $C = 0$ in equation (19), the original AdaBoost algorithm is retrieved. If C is chosen high, the data is not taken seriously and for $C \to \infty$ we (almost) retrieve the Bagging algorithm [5].

From inequality (18), we can derive the new error function (cf. equation (10)), which aims to maximize the soft margin ($\phi = \frac{1}{2}$):

$$\begin{aligned} G_{Reg}(\mathbf{b}^t) &= \sum_{i=1}^{l} \exp\left\{-\frac{1}{2}\widetilde{\mathrm{mg}}(\mathbf{z}_i, \mathbf{b}^t)\right\} \\ &= \sum_{i=1}^{l} \exp\left\{-\frac{1}{2}\left[\mathrm{mg}(\mathbf{z}_i, \mathbf{b}^t) + C|\mathbf{b}^t|\mu_t(\mathbf{z}_i)^p\right]\right\} . \end{aligned} \tag{20}$$

The weight $w_{t+1}(\mathbf{z}_i)$ of a pattern is computed as the derivative of equation (20) subject to $\widetilde{\mathrm{mg}}(\mathbf{z}_i, \mathbf{b}^t)$ (cf. lemma 1)) and is given by

$$w_{t+1}(\mathbf{z}_i) = \frac{\partial G_{Reg}(\mathbf{b}^t)}{\partial \widetilde{\mathrm{mg}}(\mathbf{z}_i, \mathbf{b}^t)} = \frac{\exp\{\widetilde{\mathrm{mg}}(\mathbf{z}_i, \mathbf{b}^t)/2\}}{\sum_{j=1}^{l} \exp\{\widetilde{\mathrm{mg}}(\mathbf{z}_j, \mathbf{b}^t)/2\}}. \tag{21}$$

For $p = 1$ we get an update rule for the weight of a training pattern in the t-th iteration [26]

$$w_{t+1}(\mathbf{z}_i) = w_t(\mathbf{z}_i) \exp\{b_t I(y_i \neq h_t(\mathbf{x}_i)) - C\zeta^t(\mathbf{z}_i)|\mathbf{b}^t|\},$$

for $p = 2$ we get

$$w_{t+1}(\mathbf{z}_i) = w_t(\mathbf{z}_i) \exp\{b_t I(y_i \neq h_t(\mathbf{x})) - C\zeta^t(\mathbf{z}_i)|\mathbf{b}^t| + C\zeta^{t-1}(\mathbf{z}_i)|\mathbf{b}^{t-1}|\}.$$

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 [23] over (20), which has an unique solution because $\frac{\partial}{\partial b_t} G_{Reg}(\mathbf{b}^t) > 0$ is satisfied for $b_t > 0$. The line search procedure can be implemented efficiently.

We can interpret this approach as regularization analogous to weight decay, whereby we want to incorporate the prior knowledge that some patterns are probably not reliable. Therefore, in the noisy case we prefer hypotheses, which do not rely on only a few patterns with high weights[3]. Instead, we are looking for hypotheses with smaller values of $\zeta(\mathbf{z}_i)$. So by this regularization, AdaBoost is not changed for easy classifiable patterns, but only for the most difficult patterns.

The variables $\zeta(\mathbf{z}_i)$ in equation (19) can also be interpreted as slack-variables (cf. SV approach and next section), which are non-linearly involved in the error function. Bigger values of $\zeta(\mathbf{z}_i)$ for some patterns allow a larger (soft-) margin ρ.

Summarizing, this modification of AdaBoost is constructed to produce a soft margin and therefore to avoid overfitting.

For a comparison of the soft margin distributions of a single RBF classifier and AdaBoost$_{Reg}$ see figure 9.

[3]Interestingly, also the (soft) SVM generates much more SV in the high noise case than in the low noise case. Therefore, the SVM shows a trend to need more patterns to find a hypothesis if the patterns are noisy [30].

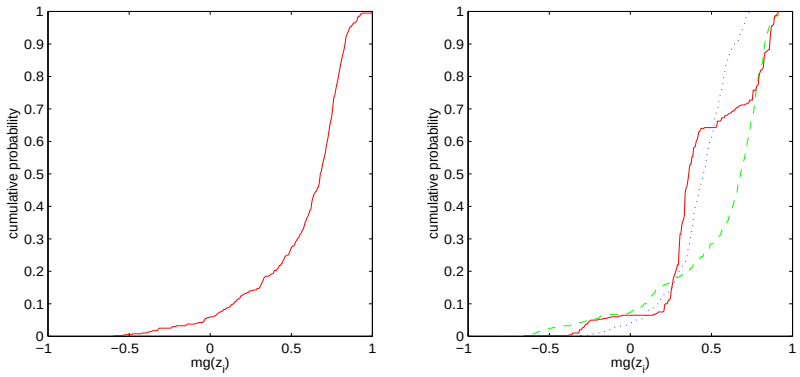

Figure 9 Margin distribution graphs of the RBF base hypothesis (scaled) trained with Mean Squared Error (left) and AdaBoost$_{Reg}$ (right) with different values of C for the toy data set after 10^3 iterations. Note that for some values for C the graphs of AdaBoost$_{Reg}$ are quite similar to the graphs of the singe RBF net.

4.2 Linear Programming with Slack Variables

Grove et al. [14] showed how to use linear programming to maximize the smallest margin for a given ensemble and proposed LP-AdaBoost (cf. algorithm (23)). In their approach, they first compute a gain (or margin) matrix $G \in \{\pm 1\}^{l \times T}$ for the given hypotheses set, which is defined by

$$G_{i,t} = y_i h_t(\mathbf{x}_i) \ . \tag{22}$$

The matrix G gives information, which hypothesis contributes a positive (or negative) part to the margin of a pattern and is used to formulate the following maxi-min problem: find a weight vector $\mathbf{c} \in \mathbb{R}^T$ for hypotheses $\{h_t\}_{t=1}^T$, which maximizes the smallest margin $\rho := \min_{i=1,\dots,l} \mathrm{mg}(\mathbf{z}_i)$. That can be solved by linear programming [17]:

Maximize ρ subject to

$$\begin{aligned} \sum_{t=1}^{T} G_{i,t} c_t &\geq \rho \qquad & i = 1, \dots, l \\ c_t &\geq 0 \qquad & t = 1, \dots T \\ \sum_{t=1}^{T} c_t &= 1 \end{aligned} \tag{23}$$

This linear program achieves a larger hard margin than the original AdaBoost algorithm. From the reasoning in section 3, LP-AdaBoost can not generalize well on noisy data, since it even stronger overemphasizes difficult patterns, e.g. outliers. Now, we define again a soft-margin for a pattern $\widetilde{\mathrm{mg}}'(\mathbf{z}_i) = \mathrm{mg}(\mathbf{z}_i) + \xi_i$ to introduce regularization for LP-AdaBoost.

Technically, this approach is equivalent to the introduction of slack variables to LP-AdaBoost and we arrive at the algorithm LP$_{Reg}$-AdaBoost [26], which solves the following linear program:

Maximize $\rho - C \sum_{i=1}^{l} \xi_i$ subject to

$$\begin{aligned} \sum_{t=1}^{T} c_t G_{i,t} &\geq \rho - \xi_i, & i &= 1, \ldots, l \\ \xi_i &\geq 0, & i &= 1, \ldots, l \\ c_t &\geq 0, & t &= 1, \ldots, T \\ \sum_{t=1}^{T} c_t &= 1 \end{aligned} \tag{24}$$

This modification allows that some patterns have smaller margins than ρ (especially lower than 0). There is a trade-off: (a) make all margins bigger than ρ and (b) maximize $\rho - C \sum_i \xi_i$. This trade-off is controlled by the constant C.

4.3 Quadratic Programming and the connection to Support Vector Machines

In the following section, we extend the LP$_{Reg}$-AdaBoost algorithm to quadratic programming by using similar techniques as in Support Vector Machines [4, 9, 17]. This gives interesting insights to the connection between SVMs and AdaBoost.

We start with transforming the LP$_{Reg}$-AdaBoost algorithm, which maximizes ρ, while $|\mathbf{c}|$ is kept fixed, to an linear program, in which ρ is fixed (to e.g. 1) and $|\mathbf{b}|$ is minimized. Unfortunately, there is no equivalent linear program. But we can use Taylor expansions[4] to get the following linear program (compare with linear programming approaches related to SV learning e.g. [31, 13, 1]):

Minimize $\|\mathbf{b}\|_1 + C \sum_i \xi_i$ subject to

$$\begin{aligned} \sum_{t=1}^{T} b_t G_{i,t} &\geq 1 - \xi_i, & t &= 1, \ldots, T, \\ b_t &\geq 0, & t &= 1, \ldots, T, \\ \xi_i &\geq 0, & i &= 1, \ldots, l\ . \end{aligned} \tag{25}$$

Essentially, this is the same algorithm as (24), but the slack variables are acting differently, because only the Taylor expansion of $1/|\mathbf{b}|$ was used. Therefore, we will achieve different soft margins as in the previous section (cf. figure 10).

Instead of using the l_1 norm in the optimization objective of (25), we can also use the l_p norm. Clearly, each p will imply its own soft margin characteristics. For $p = 2$ it will lead to an algorithm similar to the SVM.

[4] From (24), it is straight forward to get the following problem, which is for any fixed $S > 0$ equivalent to (24):

Minimize $\frac{\rho^+}{S} + \frac{C}{S} \sum_i \xi_i^+$ subject to $S \sum_{t=1}^{T} c_t G_{i,t} \geq \rho^+ - \xi_i^+$,
where $\rho^+ := S\rho$, $\xi_i^+ := S\xi_i$, $b_t \geq 0$, $\xi_i^+ \geq 0$, $\sum_t b_t = S$.

In this problem we can set ρ^+ to 1 and try to optimize S. To retrieve a linear program, we use the Taylor expansion around 1: $\frac{1}{S} = S + \mathcal{O}((S-1)^2)$ and $\sum_i \xi_i^+ / S = \sum_i \xi_i^+ + \mathcal{O}(S-1)$. For $S = |\mathbf{b}|$ we get algorithm (25).

The optimization objective of a SVM is to find a function $h^{\mathbf{w}}$ which minimizes a functional of the form [30]

$$E = \|\mathbf{w}\|_2^2 + C \sum_{i=1}^{l} \xi_i \ , \tag{26}$$

subject to the constraints

$$y_i h(\mathbf{x}_i) \geq 1 - \xi_i \qquad \text{and} \qquad \xi_i \geq 0, \qquad \text{for } i = 1, \ldots, l \ .$$

Here, the variables ξ_i are the slack-variables, which make a soft margin possible. The norm of the parameter vector $\mathbf{w}$ is a measure of the complexity and the size of the margin of hypothesis $h^{\mathbf{w}}$ [30]. With functional (26), we get a trade-off (controlled by C) between the complexity of the hypothesis and the "grade how much the hypothesis may differ from the training patterns" ($\sum_i \xi_i$).

For ensemble learning, we do not (yet) have such a measure of complexity. Empirically, we observed the following: the more different the weights for the hypotheses are, the higher the complexity of the ensemble. With this in mind, we can use the l_p norm ($p > 1$) of the hypotheses weight vector $\|\mathbf{b}\|_p$ as a complexity measure. For example, assume $|\mathbf{b}| = 1$, then $\|\mathbf{b}\|_p$ this is a small value, if the elements are approximately equal (analogous to bagging) and $\|\mathbf{b}\|_p$ has high values, when there are some strongly emphasized hypotheses (far away from bagging). This is intuitively clear, because Bagging generates usually less complex classifiers (with lower $\|\mathbf{b}\|_p$) than, for example, LP-AdaBoost (which can generate very sparse representations for which $\|\mathbf{b}\|_p$ is large).

Note that this arguments holds only if the hypotheses are weak enough, otherwise the $\|\mathbf{b}\|_p$ will not carry the desired complexity information.

Hence, we can apply the optimization principles of SVMs to AdaBoost and get the following quadratic optimization problem:

$$\text{Minimize } \|\mathbf{b}\|_2^2 + C \textstyle\sum_i \xi_i \ ,$$

with the constraints given in equation (25). This algorithm, we call it QP_{Reg}-AdaBoost, is motivated by the connection to LP_{Reg}-AdaBoost (cf. algorithm (25)) and by the analogy to the Support Vector algorithm.

It is expected, that QP_{Reg}-AdaBoost achieves large improvements over the solution of the original AdaBoost algorithm – especially in the case of noise. In comparison with LP_{Reg}-AdaBoost we expect a similar performance. Each "type of soft margin", which is implied by the norm of the weight vector, can have merits, which may be needed by some specific dataset.

Summarizing, we introduced a soft margin to AdaBoost by (a) regularizing the objective function (10), (b) LP_{Reg}-AdaBoost, which uses slack variables and (c) QP_{Reg}-AdaBoost, which has an interesting relation to SVMs. For an overall comparison of the margin distributions of original AdaBoost, SVM, AdaBoost_{Reg} and LP/QP-AdaBoost see figures 5, 6, 9 and 10.

5 Experiments

In order to evaluate the performance of our new algorithms, we make an extensive comparison among the single RBF classifier, the original AdaBoost algorithm, AdaBoost_{Reg}, L/QP_{Reg}-AdaBoost and a Support Vector Machine (with RBF kernel).

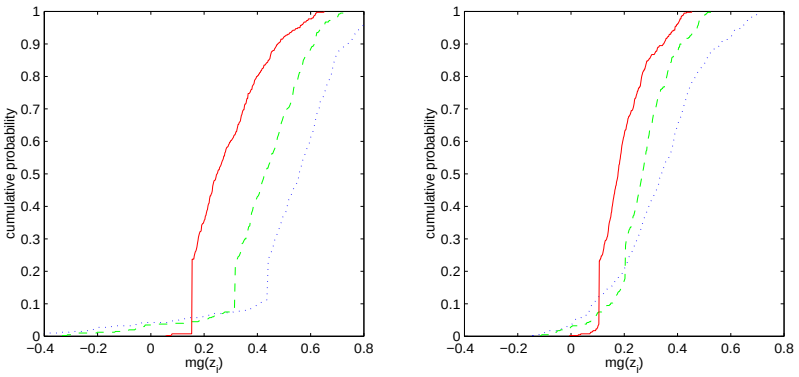

Figure 10 Margin distribution graphs of LP_{Reg}-AdaBoost (left) and QP_{Reg}-AdaBoost (right) with different values of C for the toy data set after 10^3 iterations. LP_{Reg}-AdaBoost sometimes generates margins on the training set, which are either 1 or -1 (step in the distribution).

5.1 Experimental Setup

For this, we use 13 artificial and real world datasets from the UCI, DELVE and STATLOG benchmark repositories: banana (toy data set used in the previous sections), breast cancer[5], diabetes, german, heart, image segment, ringnorm, flare sonar, splice, new-thyroid, titanic, twonorm, waveform. Some of the problems are originally not binary classification problems, hence a random partition into two classes was used[6]. At first we generate 100 partitions into training and test set (mostly $\approx$ 60% : 40%). On each partition we train a classifier and get its test set error.

In all experiments, we combined 200 hypotheses. Clearly, this number of hypotheses may be not optimal, however AdaBoost with optimal early stopping is often worse than any of the soft margin algorithms.

As base hypotheses we used RBF nets with adaptive centers as described in appendix C. On each data set we used cross validation to find the best single classifier model, which is then used in the ensemble learning algorithms.

The parameter C of the regularized versions of AdaBoost and the parameters (C, σ) of the SVM are optimized by the first five training datasets. On each training set 5-fold-cross validation is used to find the best model for this dataset[7]. 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.

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

[6]A random partition generates a mapping $\mathbf{m}$ of n to two classes. For this a random ± 1 vector $\mathbf{m}$ of length n is generated. The positive classes (and the negative respectively) are then concatenated.

[7]The parameters selected by the cross validation are only near-optimal. Only 10-20 values for each parameter are tested in two stages: first a global search (i.e. over a wide range of the parameter space) was done to find a good guess of the parameter, which becomes more precise in the second stage.

Table 1 Comparison among the six methods: Single RBF classifier, AdaBoost(AB), AdaBoost$_{Reg}$ (AB$_R$;p=2), L/QP$_{Reg}$-AdaBoost (L/QP$_R$-AB) and a Support Vector Machine (SVM): Estimation of generalization error in % on 13 datasets (best method in bold face, second emphasized). AdaBoost$_{Reg}$ gives the best overall performance.

	RBF	AB	AB$_R$	LP$_R$-AB	QP$_R$-AB	SVM
Banana	*10.8±0.6*	12.3±0.7	10.9±0.4	**10.7±0.4**	10.9±0.5	11.5±0.6
B.Cancer	27.6±4.7	30.4±4.7	26.5±5.5	26.8±6.1	**25.9±4.6**	*26.0±4.7*
Diabetes	24.1±1.9	26.5±2.3	*23.9±1.6*	24.1±1.9	25.4±2.2	**23.5±1.7**
German	24.7±2.4	27.5±2.5	*24.3±2.1*	24.8±2.2	25.2±2.1	**23.6±2.1**
Heart	17.1±3.3	20.3±3.4	16.6±3.7	**14.5±3.5**	17.2±3.4	*16.0±3.3*
Image	3.3±0.6	**2.7±0.7**	**2.7±0.6**	*2.8±0.6*	**2.7±0.6**	3.0±0.6
Ringnorm	*1.7±0.2*	1.9±0.3	**1.6±0.1**	2.2±0.5	1.9±0.2	*1.7±0.1*
F.Sonar	34.4±2.0	35.7±1.8	*34.2±2.2*	34.8±2.1	36.2±1.8	**32.4±1.8**
Splice	*9.9±1.0*	10.3±0.6	**9.5±0.7**	*9.9±1.4*	10.3±0.6	10.8±0.6
Thyroid	*4.5±2.1*	**4.4±2.2**	**4.4±2.1**	4.6±2.2	**4.4±2.2**	4.8±2.2
Titanic	23.3±1.3	*22.6±1.2*	*22.6±1.2*	24.0±4.4	22.7±1.1	**22.4±1.0**
Twonorm	*2.9±0.3*	3.0±0.3	**2.7±0.2**	3.2±0.4	3.0±0.3	3.0±0.2
Waveform	10.6±1.0	10.8±0.6	**9.8±0.8**	10.5±1.0	10.1±0.5	*9.9±0.4*
Mean %	6.6±5.8	11.9±7.9	1.7±1.9	8.9±10.8	5.8±5.5	4.6±5.4
Winner %	14.8±8.5	7.2±7.8	26.0±12.4	14.4±8.6	13.2±7.6	23.5±18.0

Note, to perform the simulations of this setup we had to train more than 3×10^6 adaptive RBF nets and to solve more than 10^5 mathematical programming problems – a task that would have taken altogether 2 years of computing time on a single Ultra-SPARC machine, if we hadn't distributed it over 30 computers.

5.2 Experimental Results

In table 1 the average generalization performance (with standard deviation) over the 100 partitions of the data sets are shown. The second last line in table 1 shows the line 'Mean %', which is computed as follows: For each dataset the average error rates of all classifier types are divided by the minimum error rate and 1 is subtracted. These resulting numbers are averaged over the 13 datasets and the variance is given, too. The last line shows the probabilities that a particular method wins, i.e. gives the smallest generalization error, on the basis of our experiments (mean and variance over all 13 datasets).
Our experiments on noisy data (cf. table 1) show that:

- The results of AdaBoost are in almost all all cases worse than the single classifier. This shows clearly the overfitting of AdaBoost. It is not able to deal with noise in the data.
- The averaged results for AdaBoost$_{Reg}$ are slightly better (Mean% and Win%) than the results of the SVM, which is known to be an excellent classifier. The single RBF classifier wins less often than the SVM (for a comparison in the regression case see [20]).
- L/QP$_{Reg}$-AdaBoost improves the results of AdaBoost. This is due to the established soft margin. But the results are not as good as the results of AdaBoost$_{Reg}$ and the SVM. One reason is that the hypotheses generated by AdaBoost (aimed to construct a hard margin) may provide not the appropriate basis to generate a good soft margin with the mathematical programming approaches.

- ◇ We can observe that quadratic programming gives slightly better results than linear programming. This may be due to the fact that the hypotheses coefficients generated by LP_{Reg}-AdaBoost are more sparse (smaller ensemble); bigger ensembles may have a better generalization ability (e.g. due to the reduction of variance [7]). Furthermore, with QP-AdaBoost we prefer ensembles, which have approximately equal weighted hypotheses. As stated in section 4.3, this implies a lower complexity of the combined hypothesis, which can lead to a better generalization performance.
- ◇ The results of AdaBoost_{Reg} are in all cases (much) better than those of AdaBoost and better than that of the single RBF classifier. AdaBoost_{Reg} wins most often and shows the best average performance. This demonstrates the noise robustness of the proposed algorithm.

The slightly inferior performance of SVM compared to AdaBoost_{Reg} may be explained with (a) the fixed σ of the RBF-kernel for SVM (loosing multi-scale information), (b) the coarse model selection, and (c) a bad error function of the SV algorithm (noise model).

Summarizing, the original AdaBoost algorithm is only useful for low noise cases, where the classes are easily separable (as shown for OCR [29, 16]). L/QP_{Reg}-AdaBoost can improve the ensemble structure through introducing a soft margin: the same hypotheses (just with another weighting) can result in an ensemble, which shows a much better generalization performance.

The hypotheses, which are used by L/QP_{Reg}-AdaBoost may be suboptimal, because they are not part of the optimization process, which aims for a soft margin. AdaBoost_{Reg} does not have this problem: the hypotheses are generated such that they are appropriate to form the desired soft-margin. AdaBoost_{Reg} extends the applicability of Boosting/Arcing methods to non-separable cases and should be applied, if the data is noisy.

6 Conclusion

We have shown that AdaBoost performs an approximate gradient decent in an error function, that optimizes the margin (cf. equation 10, see also [8, 22, 12]). 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 margin distribution of the training patterns in the margin area converges asymptotically to a step and therefore AdaBoost asymptotically achieves a *hard margin* for classification. The asymptotic margin distribution of AdaBoost is very similar to the margin distribution of a SVM (for the separable case), accordingly the patterns lying in the step part (Support Patterns) show a large overlap to the Support Vectors found by a SVM. However, the representation found by AdaBoost is often less sparse than for SVMs.

We discussed in detail that AdaBoost-type algorithms, and hard margin classifiers in general, are noise sensitive and able to overfit. We introduced three regularization-based AdaBoost algorithms to alleviate the overfitting problem of AdaBoost-type algorithms for high noise data: (1) direct incorporation of the regularization term into the error function

(AdaBoost$_{Reg}$), use of (2) linear and (3) quadratic programming with slack variables. The essence of our proposal is to achieve a soft margin (through regularization term and slack variables) in contrast to the hard margin classification used before. The soft-margin approach allows to control how much we trust the data, so we are permitted to ignore noisy patterns (e.g. outliers) which would otherwise have spoiled our classification. This generalization is very much in the spirit of Support Vector Machines that also trade-off the maximization of the margin and the minimization of the classification errors by introducing slack variables.

In our experiments on noisy data the proposed regularized versions of AdaBoost: AdaBoost$_{Reg}$ and L/QP$_{Reg}$-AdaBoost show a more robust behavior than the original AdaBoost algorithm. Furthermore, AdaBoost$_{Reg}$ exhibits a better overall generalization performance than all other algorithms including the Support Vector Machines. We conjecture that this unexpected result is mostly due to the fact that SVM can only use one σ and therefore loose multi-scaling information. AdaBoost does not have this limitation, since we use RBF nets with adaptive kernel widths as base hypotheses.

Our future work will concentrate on a continuing improvement of AdaBoost-type algorithms for noisy real world applications. Also, a further analysis of the relation between AdaBoost (QP$_{Reg}$-AdaBoost) and Support Vector Machines from the margin's point of view seems promising, with particular focus on the question of what good margin distributions should look like. Moreover, it is interesting to see how the techniques established in this work can be applied to AdaBoost in a regression scenario (cf. [2]).

Acknowledgements: We thank for valuable discussions with B. Schölkopf, A. Smola, T. Frieß, D. Schuurmans and B. Williamson. Partial funding from EC STORM project number 25387 is gratefully acknowledged.

A Proof of Lemma 1

Proof 7 *We define* $\pi_t(\mathbf{z}_i) := \prod_{r=1}^{t} \exp(-b_r I(h_r(\mathbf{z}_i) = y_i))$ *and from definition of* G *and* d *we get*

$$\begin{aligned}
\frac{\frac{\partial G}{\partial \mathrm{mg}(\mathbf{z}_i, \mathbf{b}^t)}}{\sum_{j=1}^{l} \frac{\partial G}{\partial \mathrm{mg}(\mathbf{z}_j, \mathbf{b}^t)}} &= \frac{\exp\left(-\frac{1}{2}\mathrm{mg}(\mathbf{z}_i, \mathbf{b}^t)\right)}{\sum_{j=1}^{l} \exp\left(-\frac{1}{2}\mathrm{mg}(\mathbf{z}_i, \mathbf{b}^t)\right)} \\
&= \frac{\pi_t(\mathbf{z}_i)}{\sum_{j=1}^{l} \pi_t(\mathbf{x}_j)} \\
&= \frac{\pi_t(\mathbf{z}_i)}{\widetilde{Z}_t} ,
\end{aligned}$$

where $\widetilde{Z}_t := \sum_{i=1}^{l} \pi_t(\mathbf{z}_i)$. *By definition, we have* $\pi_t(\mathbf{z}_i) = \pi_{t-1}(\mathbf{z}_i) \exp(-b_t I(h_t(\mathbf{z}_i) = y_i))$ *and* $\pi_1(\mathbf{z}_i) = 1/l$. *Thus, we get*

$$\begin{aligned}
w_{t+1}(\mathbf{z}_i) = \frac{\pi_t(\mathbf{z}_i)}{\widetilde{Z}_t} &= \frac{\pi_{t-1}(\mathbf{z}_i) \exp(-b_t I(h_t(\mathbf{z}_i) = y_i))}{\widetilde{Z}_t} \\
&= \frac{w_{t-1}(\mathbf{z}_i) \widetilde{Z}_{t-1} \exp(-b_t I(h_t(\mathbf{z}_i) = y_i))}{\widetilde{Z}_t} \\
&= \frac{w_{t-1}(\mathbf{z}_i) \exp(-b_t I(h_t(\mathbf{z}_i) = y_i))}{Z_t} ,
\end{aligned}$$

where $Z_t = \widetilde{Z}_t \widetilde{Z}_{t-1}$ *(cf. step 4 in figure 1).* ■

B Proof of Theorem 4

The proof follows the one of Theorem 5 in [28]. Theorem 4 is a generalization for $\phi \neq \frac{1}{2}$.

Proof 8 *If $yf(x) \leq \theta$, then we have*

$$y \sum_{t=1}^{T} b_t h_t(x) \leq \theta \sum_{t=1}^{T} b_t \ ,$$

and also

$$\exp\left\{ -\frac{y}{2} \sum_{t=1}^{T} b_t h_t(x) + \frac{\theta}{2} \sum_{t=1}^{T} b_t \right\} \geq 1 \ .$$

Thus,

$$\begin{aligned} \mathbf{P}_{(\mathbf{x},y)\sim\mathbf{Z}}[yf(\mathbf{x}) \leq \theta] &\leq \frac{1}{l} \sum_{i=1}^{l} \exp\left\{ -\frac{y_i}{2} \sum_{t=1}^{T} b_t h_t(\mathbf{x}_i) + \frac{\theta}{2} \sum_{t=1}^{T} b_t \right\} \\ &= \frac{\exp\left(\frac{\theta}{2} \sum_{t=1}^{T} b_t\right)}{l} \sum_{i=1}^{l} \exp\left\{ -\frac{y_i}{2} \sum_{t=1}^{T} b_t h_t(\mathbf{x}_i) \right\} \ , \end{aligned}$$

where

$$\begin{aligned} & \sum_{i=1}^{l} \exp\left\{ -\frac{y_i}{2} \sum_{t=1}^{T} b_t h_t(\mathbf{x}_i) \right\} \\ = & \sum_{i=1}^{l} \exp\left\{ -\frac{y_i}{2} \sum_{t=1}^{T-1} b_t h_t(\mathbf{x}_i) \right\} \exp\left\{ -\frac{y_i}{2} b_T h_T(\mathbf{x}_i) \right\} \\ = & \sum_{i:h_T(\mathbf{x}_i)=y_i} \exp\left\{ -\frac{y_i}{2} \sum_{t=1}^{T-1} b_t h_t(\mathbf{x}_i) \right\} e^{-b_T/2} + \\ & + \sum_{i:h_T(\mathbf{x}_i)\neq y_i} \exp\left\{ -\frac{y_i}{2} \sum_{t=1}^{T-1} b_t h_t(\mathbf{x}_i) \right\} e^{b_T/2} \\ = & \left(\sum_{i=1}^{l} \exp\left\{ -\frac{y_i}{2} \sum_{t=1}^{T-1} b_t h_t(\mathbf{x}_i) \right\} \right) \left((1-\epsilon_T) e^{-b_T/2} + \epsilon_T e^{b_T/2} \right) \ , \end{aligned}$$

because

$$\epsilon_T = \frac{1}{\sum_{j=1}^{l} w_j^T} \sum_{i:h_T(\mathbf{x}_i)\neq y_i} w_i^T \ .$$

With $\sum_{i=1}^{l} \exp(0) = l$ (for $t = 1$), we get recursively

$$\mathbf{P}_{(\mathbf{x},y)\sim\mathbf{Z}}[yf(\mathbf{x}) \leq \theta] \leq \exp\left(\frac{\theta}{2} \sum_{t=1}^{T} b_t \right) \prod_{t=1}^{T} \left((1-\epsilon_t) e^{-b_t/2} + \epsilon_t e^{b_t/2} \right) \ .$$

Plugging in the definition for b_t we get

$$\begin{aligned}
\mathbf{P}_{(\mathbf{x},y)\sim\mathbf{Z}}[yf(\mathbf{x}) \le \theta] &\le \left(\prod_{t=1}^{T}\frac{1-\epsilon_t}{\epsilon_t}\ \prod_{t=1}^{T}\frac{\phi}{1-\phi}\right)^{\theta/2}\cdot \\
&\quad\cdot\left(\sqrt{\frac{\phi}{1-\phi}}+\sqrt{\frac{1-\phi}{\phi}}\right)^{T}\prod_{t=1}^{T}\sqrt{(1-\epsilon_t)\epsilon_t} \\
&= \left(\left(\frac{\phi}{1-\phi}\right)^{\frac{1+\phi}{2}}+\left(\frac{1-\phi}{\phi}\right)^{\frac{1-\phi}{2}}\right)^{T}\cdot \\
&\quad\cdot\prod_{t=1}^{T}\sqrt{(1-\epsilon_t)^{1+\theta}\epsilon_t^{1-\theta}} \\
&= \left(\varphi^{\frac{1+\theta}{2}}+\varphi^{-\frac{1-\theta}{2}}\right)^{T}\prod_{t=1}^{T}\sqrt{\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}}\ .
\end{aligned}$$

■

C RBF nets with adaptive centers

The RBF nets used in the experiments are an extension of the method of Moody and Darken [19], since centers and variances are also adapted (see also [3, 21]). The output of the network is computed as a linear superposition of K basis functions

$$f(\mathbf{x}) = \sum_{k=1}^{K} w_k\, g_k(\mathbf{x})\ , \tag{27}$$

where w_k, $k = 1,\ldots,K$, denotes the weights of the output layer. The Gaussian basis functions g_k are defined as

$$g_k(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mu_k\|^2}{2\,\sigma_k^2}\right), \tag{28}$$

where μ_k and σ_k^2 denote means and variances, respectively. In a first step, the means μ_k are initialized with K-means clustering and the variances σ_k are determined as the distance between μ_k and the closest μ_i ($i \neq k, i \in \{1,\ldots,K\}$). Then in the following steps we perform a gradient descent in the regularized error function (weight decay)

$$E = \frac{1}{2}\sum_{i=1}^{l}\left(y_i - f(\mathbf{x}_i)\right)^2 + \frac{\lambda}{2l}\sum_{k=1}^{K} w_k^2. \tag{29}$$

Taking the derivative of equation (29) with respect to RBF means $\mu_\mathbf{k}$ and variances σ_k we obtain

$$\frac{\partial E}{\partial \mu_k} = \sum_{i=1}^{l}(f(\mathbf{x}_i)-y_i)\frac{\partial}{\partial \mu_k}f(\mathbf{x}_i),\ \text{with}\quad \frac{\partial}{\partial \mu_k}f(\mathbf{x}_i) = w_k\frac{\mathbf{x}_i-\mu_k}{\sigma_k^2}g_k(\mathbf{x}_i) \tag{30}$$

Algorithm RBF-Net

Input:

Sequence of labeled training patterns $\mathbf{Z} = \langle(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\rangle$

Number of RBF centers K

Regularization constant λ

Number of iterations T

Initialize:

Run K-means clustering to find initial values for μ_k and determine $\sigma_k, k = 1, \ldots, K$, as the distance between μ_k and the closest μ_i $(i \neq k)$.

Do for $t = 1 : T$,

1. Compute optimal output weights $\mathbf{w} = \left(G^\top G + 2\frac{\lambda}{l}\mathbf{I}\right)^{-1} G^\top \mathbf{y}$

2a. Compute gradients $\frac{\partial}{\partial \mu_k} E$ and $\frac{\partial}{\partial \sigma_k} E$ as in (31) and (30) with optimal $\mathbf{w}$ and form a gradient vector $\mathbf{v}$

2b. Estimate the conjugate direction $\overline{\mathbf{v}}$ with Fletcher-Reeves-Polak-Ribiere CG-Method [23]

3a. Perform a line search to find the minimizing step size δ in direction $\overline{\mathbf{v}}$; in each evaluation of E recompute the optimal output weights $\mathbf{w}$ as in line 1

3b. Update μ_k and σ_k with $\overline{\mathbf{v}}$ and δ

Output: Optimized RBF net

Figure 11 Pseudo-code description of the RBF net algorithm, which is used as base learning algorithm in the simulations with AdaBoost

and

$$\frac{\partial E}{\partial \sigma_k} = \sum_{i=1}^{l} \left(f(\mathbf{x}_i) - y_i\right) \frac{\partial}{\partial \sigma_k} f(\mathbf{x}_i), \quad \text{with} \quad \frac{\partial}{\partial \sigma_k} f(\mathbf{x}_i) = w_k \frac{\|\mu_k - \mathbf{x}_i\|^2}{\sigma_k^3} g_k(\mathbf{x}_i) \ . \tag{31}$$

These two derivatives are employed in the minimization of equation (29) by a conjugate gradient descent with line search, where we always compute the optimal output weights in every evaluation of the error function during the line search. The optimal output weights $\mathbf{w} = [w_1, \ldots, w_K]^\top$ in matrix notation can be computed in closed form by

$$\mathbf{w} = \left(G^T G + 2\frac{\lambda}{l}\mathbf{I}\right)^{-1} G^T \mathbf{y}, \quad \text{where} \quad G_{ik} = g_k(\mathbf{x}_i) \tag{32}$$

and $\mathbf{y} = [y_1, \ldots, y_l]^\top$ denotes the output vector, and $\mathbf{I}$ an identity matrix. For $\lambda = 0$, this corresponds to the calculation of a pseudo-inverse of G.

So, we simultaneously adjust the output weights and the RBF centers and variances (see figure 11 for pseudo-code of this algorithm). In this way, the network fine-tunes itself to the data after the initial clustering step, yet, of course, overfitting has to be avoided with careful tuning of the regularization parameter, the number of centers K and the number of iterations (cf. [3]). In our experiments we always used $\lambda = 10^{-6}$ and up to ten CG iterations.

References

[1] K. Bennett. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods - SV Learning*, Cambridge, MA, 1998. MIT Press. forthcoming.

[2] A. Bertoni, P. Campadelli, and M. Parodi. A boosting algorithm for regression. In W.Gerstner, A.Germond, M.Hasler, and J.-D. Nicoud, editors, *Proceedings ICANN'97, Int. Conf. on Artificial Neural Networks*, volume V of *LNCS*, pages 343–348, Berlin, 1997. Springer.

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

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

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

[6] L. Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California, June 1997.

[7] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, July 1997.

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

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

[10] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT: European Conference on Computational Learning Theory*. LNCS, 1994.

[11] Y. Freund and R.E. Schapire. Game theory, on-line prediction and boosting. In *Proc. 9th Annu. Conf. on Comput. Learning Theory*, pages 325–332. ACM Press, New York, NY, 1996.

[12] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity, July 1998.

[13] T.T. Frieß and R.F. Harrison. Perceptrons in kernel feature space. RR RR-720, Dept. of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, 1998.

[14] 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.

[15] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *J. Statistical Physics*, 34:975–986, 1984.

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

[17] O.L. Mangasarian. *Nonlinear Programming.* McGraw-Hill, New York, NY, 1969.

[18] Llew Mason, Peter Bartlett, and Jonathan Baxter. Improved generalization through explicit optimization of margins. Technical report, Deparment of Systems Engineering, Australian National University, 1998.

[19] J. Moody and C.J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[20] K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Using support vector machines for time series prediction. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proceedings ICANN'97, Int. Conf. on Artificial Neural Networks*, volume 1327 of *LNCS*, pages 999–1004, Berlin, 1997. Springer.

[21] K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Using support vector machines for time series prediction. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning.* MIT Press, Cambridge, MA, 1998. to appear.

[22] T. Onoda, G. Rätsch, and K.-R. Müller. An asymptotic analysis of AdaBoost in the binary classification case. In *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'98)*, March 1998. In Press.

[23] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, second edition, 1992.

[24] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992.

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

[26] G. Rätsch. Ensemble learning methods for classification. April 1998. Diploma thesis (in German). http://www.first.gmd.de/~raetsch/diplom.ps.gz.

[27] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of COLT'98*, March 1998.

[28] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[29] H. Schwenk and Y. Bengio. AdaBoosting neural networks. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'97)*, volume 1327 of *LNCS*, pages 967–972, Berlin, 1997. Springer.

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

[31] J. Weston, A Gammerman, M. O. Stitson, V. Vapnik, V.Vovk, and C. Watkins. Density estimation using sv machines. Technical Report CSD-TR-97-23, Royal Holloway, University of London, Egham, UK, 1997.