An asymptotic analysis of AdaBoost in the binary classification case

T. Onoda*, G. Rätsch, K.-R. Müller
GMD FIRST Rudower Chaussee 5, 12489 Berlin, Germany
e-mail:{onoda, raetsch, klaus}@first.gmd.de

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 type algorithms. 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 for the binary classification case underlining the theoretical findings. Finally, the relation between the model complexity and noise in the training data, and how to improve AdaBoost type algorithms in practice are discussed.

1 Introduction

An ensemble is a collection of neural networks or other types of classifiers (predictors) 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 [6, 4]. In this work we investigate the functioning of the ensemble learning process in greater detail by looking at the asymptotic character of boosting as a particular example of an ensemble method. This character can be studied nicely from the margins perspective. Discussing the error function of boosting we can introduce the intuition of annealing and suggest strategies to improve boosting methods in practice.

2 AdaBoost type algorithms

Let $\{h_t(\mathbf{x}) : t = 1, \ldots, T\}$ be an ensemble of T predictors 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: $\text{sign}\left(\sum_t c_t h_t(\mathbf{x})\right)$. In order to train this ensemble of T predictors $\{h_t(\mathbf{x})\}$ and $\mathbf{c}$, several algorithms have been proposed: bagging, where the weighting is simply $c_t = 1/T$ [2] and AdaBoost/Arcing, where the weighting scheme is more complicated [5, 3]. In the following we give a brief description of the

*permanent address: Communication & Information Research Lab. CRIEPI, 2-11-1 Iwado kita, Komae-shi, Tokyo 201-8511, Japan.

AdaBoost/Arcing algorithms[1]. 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

$$mg(\mathbf{z}_i, \mathbf{c}) = y_i \sum_{t=1}^{T} c_t h_t(\mathbf{x}_i), \qquad mg(\mathbf{z}_i, \mathbf{c}) \in [-1, 1]. \tag{1}$$

The right 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. We define $d(\mathbf{z}_i, \mathbf{c})$ as the rate of incorrect classification

$$d(\mathbf{z}_i, \mathbf{c}) \quad = \quad \sum_t c_t I(y_i \neq h_t(\mathbf{x}_i)) = \left(\frac{1}{2} - \frac{mg(\mathbf{z}_i, \mathbf{c})}{2} \right), \tag{2}$$

where $I(y_i \neq h_t(\mathbf{x}_i))$ ($I(true) = 1$ and $I(false)$=0) is the non-negative loss function. AdaBoost type learning minimizes a function of $d(\mathbf{z}_i, \mathbf{c})$

$$g(\mathbf{b}) = \sum_i \exp\{|\mathbf{b}|(d(\mathbf{z}_i, \mathbf{c}) - \phi)\}$$

where $|\mathbf{b}| = \sum b_t$ (starting $\mathbf{b}$=0). Note that $\mathbf{b}$ is the unnormalized weighting of the hypotheses h_t. The quantity $\mathbf{c}$ is simply a normalized version of $\mathbf{b}$, i.e. $\mathbf{c} = \mathbf{b}/|\mathbf{b}|$. The algorithm has a parameter $\phi \in [0, 1]$ and for $\phi = 1/2$ we retrieve the original AdaBoost.
In order to find the predictor h_t the learning examples $\mathbf{z}_i$ are weighted with $w^t(\mathbf{z}_i)$ (this is not to be confused with the normalized (unnormalized) weights of the hypothesis $\mathbf{c}_t$ ($\mathbf{b}_t$)). Using a bootstrap on this weighted sample distribution we train h_t, alternatively weighted mean squared error training can be employed. The weights $w^t(\mathbf{z}_i)$ are then updated according to

Algorithm AdaBoost(ϕ)
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 distribution w^t and obtain hypothesis $h_t : \mathbf{x} \mapsto \{-1, 1\}$
2. calculate the error ϵ_t of h_t with (4)
3. set $b_t = \log \frac{\epsilon_t}{1-\epsilon_t} - \log \frac{\phi}{1-\phi}$.
4. update example distribution w^t with (3)
Output: $f(\mathbf{x}) = \frac{1}{|\mathbf{b}|} \sum_{t=1}^{T} b_t h_t(\mathbf{x})$

Figure 1: The AdaBoost type algorithm

$$w^t(\mathbf{z}_i) = \frac{\exp\{d(\mathbf{z}_i, \mathbf{b}) - \phi|\mathbf{b}|\}}{\sum_j \exp\{d(\mathbf{z}_j, \mathbf{b}) - \phi|\mathbf{b}|\}} = \frac{\exp\{-|\mathbf{b}| mg(\mathbf{z}_i, \mathbf{c})/2\}}{\sum_j \exp\{-|\mathbf{b}| mg(\mathbf{z}_j, \mathbf{c})/2\}}, \tag{3}$$

and the training error ϵ_t of h_t is computed as

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

We can minimize $g(\mathbf{b})$ by selecting b_t as follows,

$$b_t = \log \frac{\phi}{1-\phi} + \log \frac{1-\epsilon_t}{\epsilon_t}, \tag{5}$$

[1] Due to their similarity, we will refer in the following to AdaBoost[5] and unnormalized Arcing[3] (with exponential function) as *AdaBoost type algorithms.*

For an algorithmic description see pseudocode in Fig. 1. Interestingly,

$$w^t(\mathbf{z}_i) = \frac{\partial g(\mathbf{b})}{\partial mg(\mathbf{z}_i, \mathbf{b})} / \sum_j \frac{\partial g(\mathbf{b})}{\partial mg(\mathbf{z}_j, \mathbf{b})} \ ,$$

which is a gradient of $g(\mathbf{b})$ with respect to the margins. This $w^t(\mathbf{z}_i)$ will give a predictor h_t which is an approximation of a true predictor h_t^* minimizing $g(\mathbf{b})$. Note that, the weighted minimization (bootstrap, weighted LS) will not always give h_t^*, even if ϵ_t is minimized. AdaBoost type algorithms are therefore approximate gradient descent methods which minimize $g(\mathbf{b})$.

3 Asymptotic character for AdaBoost

Theoretical asymptotic analysis: Inspecting Eq.(3) more closely we see that AdaBoost type rules use 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 increases linearly with the number of iterations (see Fig. 2). From Eq. (5), we observe that if the training error ϵ_t takes 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 the point of convergence 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, ϕ has to be decreased, with the constraint $\phi > \epsilon_t$. This reasoning also shows that AdaBoost is not independent of the hypothesis class being used and the character of the annealing process will vary strongly depending on the strength of the learner.
From Eq. (1), $g(\mathbf{b})$ can be rewritten as

$$g(\mathbf{b}) = \sum_i \exp\left\{ -\frac{|\mathbf{b}| mg(\mathbf{z}_i, \mathbf{c})}{2} + \left(\frac{1}{2} - \phi\right) |\mathbf{b}| \right\} = g(\mathbf{c}, |\mathbf{b}|). \tag{6}$$

So, a reduction of $g(\mathbf{c}, |\mathbf{b}|)$ 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 type algorithms try to decrease the negative margin maximally in order to improve the error $g(\mathbf{c}, |\mathbf{b}|)$.
Now, let us consider an 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

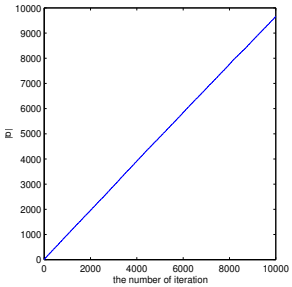

Figure 2: Relation between $|\mathbf{b}|$ and the number of iterations for one experiment

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 (3), when the annealing parameter $|\mathbf{b}|$ takes a very big value, AdaBoost type 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: (1) training patterns, which are in the margin area, have the same stability and (2) the value of the stabilities is decided by the specific AdaBoost type annealing process. And (3) the speed of the annealing process depends on the fixed parameter ϕ from Eq.(3) and on $|\mathbf{b}|$ which is an implicit function of the strength of the learner in the training process.
In order to get a good generalization error, we should either have a slow cooling schedule, i.e. we should set b_t as small as possible or ϕ should be adapted. A slower schedule, however, also takes more computing time, so there is a trade-off. In real world problems as OCR and pattern recognition, usually a strong learner is adopted as basis predictor[6, 4]. It follows that ϵ_t takes small values and $|\mathbf{b}|$ will be large. Therefore, in order to improve performance, we have to adapt ϕ to the value of ϵ_t, i.e. ϕ should be chosen *smaller* than $1/2$.

Experimental asymptotic analysis: This section presents asymptotic numerical simulations to underline our discussion from last section. In all simulations, radial basis function (RBF) networks with adaptive centers [1] are used as learners. Fig. 3 shows an example of our training data (several 2D Gauss blobs with added normal distributed noise $N(0, \sigma^2)$ with $\sigma = 0.0, 0.05, 0.15, 0.25$).

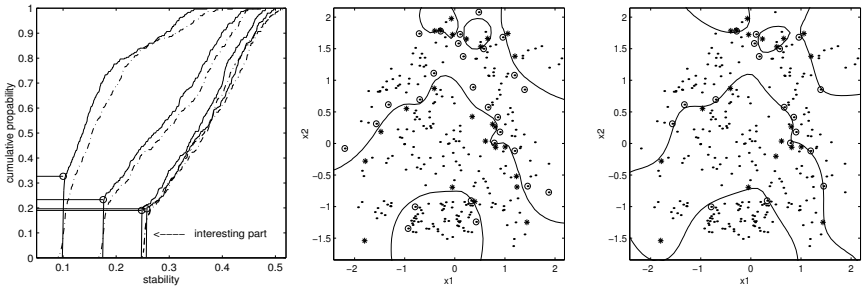

Figure 3: Margin distributions for AdaBoost at $\sigma^2 = 0.0, 0.05, 0.15, 0.25$ after 1000 (dash-dot) and 10000 (solid) iterations (left). Decision lines for AdaBoost (middle) and SVM (right) for a low noise case with similar generalisation errors. AdaBoost'ed RBF networks with 13 centers were used.

AdaBoost is applied to these data sets. Fig. 3 shows margin distributions for AdaBoost's iteration times 1000 and 10000 at different σ^2. 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. These numerical results support our theoretical asymptotic analysis. The margin distribution of AdaBoost resembles the one of Support Vector Machines (SVMs) [7]. From Fig. 3, we can observe that the decision line constructed by AdaBoost is similar to the one constructed by SVMs. All patterns, that are support vectors, also lie within the step part of the margin distribution for AdaBoost.

In the analysis below, we call the height of the step δ and the off-set of the step ρ. Fig. 4 shows how δ and ρ depend on the noise in the training data and the complexity of the basis predictor. From this figure, we see that increasing noise leads to decreasing ρ. The reason for this is the higher error ϵ_t. Generalizing theorem 5 of [5] we can prove an asymptotical lower bound on ρ:

$$\rho \geq \frac{\ln\left(\phi\epsilon^{-1}\right) + \ln\left((1-\phi)(1-\epsilon)^{-1}\right)}{\ln\left(\phi\epsilon^{-1}\right) - \ln\left((1-\phi)(1-\epsilon)^{-1}\right)} \ , \tag{7}$$

where $\epsilon = \max_t \epsilon_t$. We can see the interaction between ϕ and ϵ: if the difference of ϵ and ϕ is small, the right side of (7) is small. The smaller ϕ the more important is this difference. From theorem 7.2 of [3] we also have the weaker bound $\rho \geq 1 - \phi$ and so, if ϕ is small then ρ must be big. Therefore, choosing ϕ small maximizes the minimum margin.

From Fig. 4, we can also observe that an increase of the complexity of the basis predictor leads to an increased ρ, because the error ϵ_t will decrease. A model, which can represent higher complexity, can achieve a smaller ϵ_t.

Higher noise will increase δ, due to the increased margin area. In order to

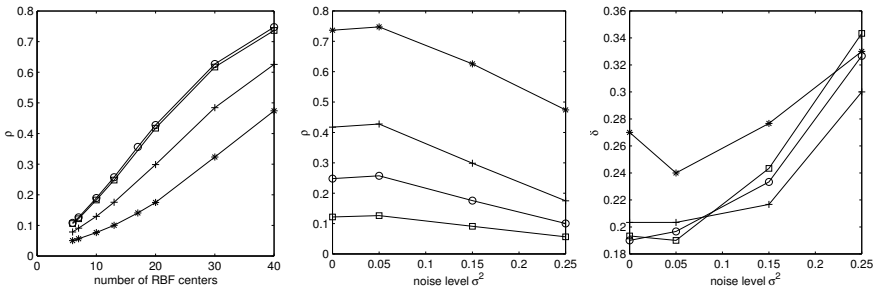

Figure 4: The relation between the model complexity and ρ for $\sigma^2 = 0.0(\square)$, $0.05(\mathrm{o})$, $0.15(+)$, $0.25(*)$ (left) and of ρ (middle) and δ (right) on noise for $7, 13, 20, 40$ RBF centers (same symbol order).

clarify the capability of AdaBoost using strong learners, AdaBoost is applied to USPS data using RBF networks. These networks have 20 centers and can represent high complexity. The table

ϕ	0.05	0.1	0.2	0.3	0.4	**0.5**	0.6	SVM
20 iter.	26	22	27	23	29	**29**	29	25
50 iter.	24	27	30	27	27	**32**	30	25

NB: For $\phi = 0.05$ and 61 iterations we got an error of only 22.

compares the number of misclassified patterns on the test set of SVM [8] and AdaBoosted RBF networks after 20/50 boosting iterations. Each method classifies between the digit "2" and all other digits, the number of training patterns is 7291 and the number of test patterns is 2007. The training error ϵ_t of RBF networks is ≈ 0.04. When ϕ is large, we can observe that the misclassification rate of RBF networks with AdaBoost is larger than the one of SVMs. But, when ϕ chosen small, we see a superior performance of AdaBoosted RBF networks compared to SVMs (cf. Table). This result shows that the selection of ϕ adapting to the achieved ϵ_t values is very useful for real world problems.

From our simulations we observe a slight overfitting effect as a function of the number of boosting iterations. Also here, due to the increased $|\mathbf{b}|$ value, noisy

patterns get a large emphasis leading to overfitting. So early stopping can avoid overfitting for AdaBoost type algorithms (further details go beyond the scope of this contribution).

4 Discussion and Conclusion

We have shown that AdaBoost type algorithms perform approximate gradient decent in a specific error function (cf. Eq.(6)), 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.
The asymptotic analysis for AdaBoost type algorithms also shows that the adaptation of ϕ to the hypothesis training error ϵ_t is very useful for real world problems. We also observe that early stopping helps to avoid overfitting. 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 off-set of the step is decided through a specific annealing type process governed by $|\mathbf{b}|$ and the noise level present in the data. In this context a generalization of the bound from [5] is given. 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 less sparse than for SVMs.
Our future work will concentrate on a continuing improvement of AdaBoost type algorithms for 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 and B. Schölkopf and for funding from DFG (JA 379/71).

References

[1] C. M. Bishop. *Neural Networks for Pattern Recogn.* Clarendon Press, 1995.

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

[3] L. Breiman. Prediction games and arcing algorithms. Tech.Rep. 504, Berkeley Stat.Dept., 1997.

[4] Y. LeCun et al. *Neural Networks*, p. 261–276, 1995.

[5] R. Schapire, Y. Freund, P. Bartlett, W. Lee. *ML*, p. 148–156, 1998.

[6] H. Schwenk and Y. Bengio. In *ICANN'97*, p. 967–972, 1997.

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

[8] B. Schölkopf et al., IEEE Trans. Signal Proc., 45(11), 2758-2765, 1997.