11 Robust Ensemble Learning

Gunnar Rätsch
GMD FIRST
Kekuléstr. 7
12489 Berlin, Germany
raetsch@first.gmd.de

Bernhard Schölkopf
Microsoft Research Limited
St. George House, 1 Guildhall Street
Cambridge CB2 3NH, UK
bsc@microsoft.com

Alexander J. Smola
Department of Engineering
Australian National University
Canberra ACT 0200, Australia
Alex.Smola@anu.edu.au

Sebastian Mika
GMD FIRST
Kekuléstr. 7
12489 Berlin, Germany
mika@first.gmd.de

Takashi Onoda
CRIEPI
2-11-1, Iwado Kita, Komae-shi
Tokyo, Japan
onoda@criepi.denken.or.jp

Klaus–Robert Müller
GMD FIRST
Kekuléstr. 7
12489 Berlin, Germany
klaus@first.gmd.de

AdaBoost and other ensemble methods have successfully been applied to a number of classification tasks, seemingly defying problems of overfitting. AdaBoost performs gradient descent in an error function with respect to the margin, asymptotically concentrating on the patterns which are hardest to learn. For noisy problems, however, this can be disadvantageous. Indeed, theoretical analysis has shown that the margin *distribution*, as opposed to just the minimal margin, plays a crucial role in understanding this phenomenon. Loosely speaking, some outliers should be tolerated if this has the benefit of substantially increasing the margin on the remaining points.

We propose new boosting algorithms which, similar to ν-Support-Vector Classification, allows for the possibility of a pre-specified fraction ν of points to lie in the margin area or even on the wrong side of the decision boundary.

Unlike other regularized boosting algorithms (Mason et al., 1999; Rätsch et al., 1998), this gives a nicely interpretable way of controlling the trade-off between minimizing the training error and capacity.

11.1 Introduction

Boosting and related Ensemble learning methods have been recently used with great success in applications such as Optical Character Recognition (Drucker et al., 1993; LeCun et al., 1995; Schwenk and Bengio, 1998) (see also Section 1.5).

The idea of a large (minimum) margin explains the good generalization performance of AdaBoost in the low noise regime. However, AdaBoost performs worse on noisy tasks (Quinlan, 1996b; Rätsch et al., 1998), such as the *iris* and the *breast cancer* benchmark data sets (Blake et al., 1998). On the latter tasks, a large margin on *all* training points cannot be achieved without adverse effects on the generalization error. This experimental observation was supported by the study of Schapire et al. (1998) (see also Theorem 1.15), where the generalization error of ensemble methods was bounded by the sum of the fraction of training points which have a margin smaller than some value ρ, say, plus a complexity term depending on the base hypotheses and ρ. While this worst-case bound[1] can only capture part of what is going on in practice, it nevertheless already conveys the message that in some cases it pays to allow for some points which have a small margin, or are misclassified, if this leads to a larger overall margin on the remaining points.

To cope with this problem, it was mandatory to construct *regularized* variants of AdaBoost, which traded off the number of margin errors and the size of the margin (Mason et al., 1999; Rätsch et al., 1998). This goal, however, was so far achieved in a heuristic way by introducing regularization parameters which have no immediate interpretation and which cannot be adjusted easily.

1. Note that the complexity term depends on the VC dimension of the base hypothesis, which is a worst case capacity measure.

The present chapter addresses this problem in two ways. Primarily, it makes an *algorithmic* contribution (including pseudocode) to the problem of constructing regularized boosting algorithms. However, compared to the previous efforts, it parameterizes the trade-off in a much more intuitive way: its only free parameter directly determines the fraction of margin errors.

This, in turn, is also appealing from a *theoretical* point of view, since it involves a parameter which controls a quantity that plays a crucial role in the generalization error bounds (cf. also Mason et al. (1999); Schapire et al. (1998)). Furthermore, it allows the user to roughly specify this parameter once a reasonable estimate of the expected error (possibly from other studies) can be obtained, thus reducing the training time.

11.2 Boosting and the Linear Programming Solution

Before deriving new algorithms, we briefly discuss the properties of the solution generated by standard AdaBoost and, closely related, Arc-GV (Breiman, 1998), and discuss the relation to a linear programming (LP) solution over the class of base hypotheses G. Let us recall the definitions from Section 1.4: Let $\{g_t(\mathbf{x}) : t = 1, \ldots, T\}$ be a sequence of hypotheses and $\boldsymbol{\alpha} = [\alpha_1 \ldots \alpha_T]$ their weights satisfying $\alpha_t \geq 0$. The hypotheses g_t are elements of a hypotheses class[2] $G = \{g : \mathbf{x} \mapsto \{\pm 1\}\}$, which is defined by a base learning algorithm L.

The ensemble generates the label which is the weighted majority of the votes by $\operatorname{sgn}(f(\mathbf{x}))$ where

$$f(\mathbf{x}) = \sum_t \frac{\alpha_t}{\|\boldsymbol{\alpha}\|_1} g_t(\mathbf{x}). \tag{11.1}$$

margins

In order to express that f and therefore also the margin ρ depend on $\boldsymbol{\alpha}$ and for the ease of notation we define (cf. also Definition 1.2)

$$\rho(\mathbf{z}, \boldsymbol{\alpha}) := y f(\mathbf{x}) \text{ where } \mathbf{z} := (\mathbf{x}, y) \text{ and } f \text{ is defined as in (11.1).} \tag{11.2}$$

Likewise we set

$$\rho(\boldsymbol{\alpha}) := \min_{1 \leq i \leq m} \rho(\mathbf{z}_i, \boldsymbol{\alpha}) \ , \tag{11.3}$$

i.e. we will use the *normalized* margin.

minimization objective

The minimization objective of AdaBoost can be expressed in terms of the margins

$$\mathcal{G}(\boldsymbol{\alpha}) := \sum_{i=1}^{m} \exp(-\|\boldsymbol{\alpha}\|_1 \rho(\mathbf{z}_i, \boldsymbol{\alpha})) \ . \tag{11.4}$$

2. Most of the work is also applicable to a hypothesis class of the type $G = \{g : \mathbf{x} \mapsto [-1, 1]\}$, except for all things which depend on the definition of ϵ_t.

In every iteration AdaBoost tries to minimize this error by a stepwise maximization of the margin. It is widely believed that AdaBoost tries to maximize the *smallest margin* on the training set (Breiman, 1998; Freund and Schapire, 1997; Frean and Downs, 1998; Schapire et al., 1998; Rätsch et al., 1998). Strictly speaking, however, a general proof is missing. It would imply that AdaBoost asymptotically approximates (up to scaling) the solution of the following linear programming problem over the complete hypothesis set G (cf. Grove and Schuurmans (1998), assuming a *finite* number of basis hypotheses):

linear programming

$$\begin{array}{lll} \text{maximize} & \rho & \\ \text{subject to} & \rho(\mathbf{z}_i, \boldsymbol{\alpha}) \geq \rho & \text{for all } 1 \leq i \leq m \\ & \alpha_t, \rho \geq 0 & \text{for all } 1 \leq t \leq |G| \\ & \|\boldsymbol{\alpha}\|_1 = 1 & \end{array} \tag{11.5}$$

Since such a linear program cannot be solved exactly for a infinite hypothesis set in general, it is interesting to analyze approximation algorithms for this kind of problems. First we have a look at the asymptotic properties of two Boosting algorithms (AdaBoost and Arc-GV).

Considering the optimization strategy of AdaBoost, it essentially consists of two parts: (i) the selection of a hypothesis and then (ii) the weighting of this hypothesis. In turn, we address their benefits (or shortcomings, respectively) for achieving convergence to the LP solution.

choosing hypotheses

The first part is done by adaptively re-weighting the patterns in each iteration t

$$w_{t+1}(\mathbf{z}_i) = \frac{\partial \mathcal{G}(\boldsymbol{\alpha}^t)/\partial \rho(\mathbf{z}_i, \boldsymbol{\alpha}^t)}{\sum_{j=1}^m \partial \mathcal{G}(\boldsymbol{\alpha}^t)/\partial \rho(\mathbf{z}_j, \boldsymbol{\alpha}^t)} = \frac{\exp(-\|\boldsymbol{\alpha}^t\|_1 \rho(\mathbf{z}_i, \boldsymbol{\alpha}^t))}{\sum_{j=1}^m \exp(-\|\boldsymbol{\alpha}^t\|_1 \rho(\mathbf{z}_j, \boldsymbol{\alpha}^t))}, \tag{11.6}$$

where $\boldsymbol{\alpha}^t = [\alpha_1, \ldots, \alpha_t, 0, 0, \ldots]$ is the weight vector after the t-th iteration. AdaBoost assigns the highest weights to the patterns with the smallest margin, i.e. to those patterns for which the inequality on the margin in (11.5) is (almost) an equality. In the case of an equality for a pattern in (11.5) the Lagrange multiplier would be non-zero (if $\boldsymbol{\alpha}$ is optimal). Increasing the weight will eventually lead to a larger margin of the corresponding pattern and, hence, to an increase of ρ. This way of finding a new hypothesis seems to be appropriate. After all, Arc-GV, which has been proven to converge (Breiman, 1998), is identical to AdaBoost in the first part (cf. page 212), i.e. in choosing the hypotheses.

choosing coefficients

The second part is computing the weight of the selected hypothesis α_t. We first consider AdaBoost. Then we show what has to be changed to get Arc-GV and the desired convergence property.

AdaBoost minimizes the function $\mathcal{G}$ with respect to the weight α_t for the new hypothesis. However, minimizing $\mathcal{G}$ does not guarantee maximization of $\rho(\boldsymbol{\alpha})$ (cf. Figure 11.1). In particular, in the t-th iteration, the minimization of $\mathcal{G}$ wrt. α_t amounts to:

$$\alpha_t := \operatorname*{argmin}_{\alpha_t \geq 0} \sum_{i=1}^m \exp\left[-(\|\boldsymbol{\alpha}^{t-1}\|_1 + \alpha_t)\rho\left(\mathbf{z}_i, \boldsymbol{\alpha}^t\right)\right] \tag{11.7}$$

$$= \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right), \tag{11.8}$$

where ϵ_t is defined as in Algorithm 1.2 (cf. footnote 2). Now suppose that $\|\boldsymbol{\alpha}^{t-1}\|_1$ is large enough and $\rho(\boldsymbol{\alpha}^t) > 0$. Then $\mathcal{G}$ may be minimized either by maximizing the margin (which brings us closer to the LP solution) or by increasing $\|\boldsymbol{\alpha}^t\|_1 = \|\boldsymbol{\alpha}^{t-1}\|_1 + \alpha_t$ (which increases the slope of the cost function). Therefore, in this case, one will obtain values of α_t somewhat larger than the value maximizing ρ (see Figure 11.1 for an example). In other cases it may occur that α_t is chosen too small.

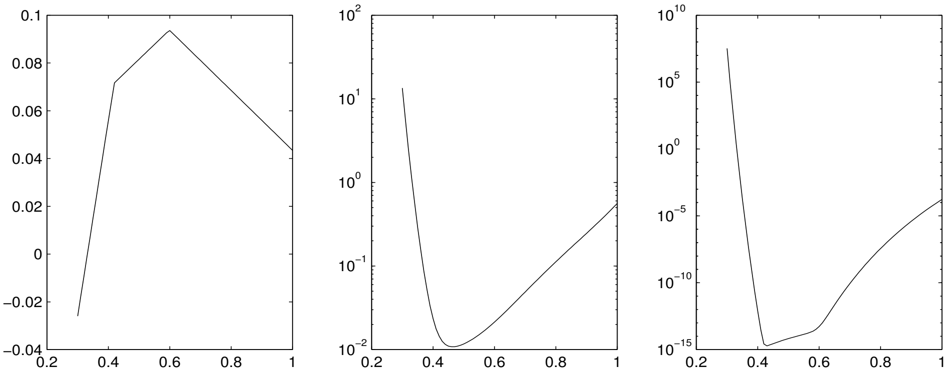

Figure 11.1 Illustration of the non-optimal selection of α_t in (11.7) (on a toy example): Left: $\rho(\boldsymbol{\alpha})$; middle and right: $\mathcal{G}(\boldsymbol{\alpha})$ for two different stages in the boosting process ($\|\boldsymbol{\alpha}\|_1 = 30$ and 300). We set $\boldsymbol{\alpha} = \boldsymbol{\alpha} + (1-\lambda)\lambda^{-1}\|\boldsymbol{\alpha}\|_1\mathbf{e}_t$, where $\mathbf{e}_t$ being the t-th unit vector. In all three cases, λ is shown on the abscissae. Clearly, the maximum of ρ and the minimum of G do not match.

A different, particularly simple strategy would be to choose α_t such that the margin is maximal, i.e.:

$$\alpha_t := \operatorname*{argmax}_{\alpha_t > 0} \min_{1 \le i \le m} \rho(\mathbf{z}_i, \boldsymbol{\alpha}^t).$$

Unfortunately, there exist examples where this strategy leads to non-optimal margins: Assume we have $n > 2$ hypotheses and $m > 2$ patterns. Now we run the algorithm and get at some iteration the margins $[\rho(\mathbf{z}_1), \rho(\mathbf{z}_2), \ldots, \rho(\mathbf{z}_m)] := [y_1 f(\mathbf{x}_1), y_2 f(\mathbf{x}_2), \ldots, y_m f(\mathbf{x}_m)]$ with $\rho(\mathbf{z}_1) = \rho(\mathbf{z}_2) < \min(\rho^{\mathrm{lp}}, \rho(\mathbf{z}_i))$, $(i = 3, \ldots, m)$, on the training set. Furthermore, suppose two hypotheses have the margins $[-1, 1, \ldots]$ and $[1, -1, \ldots]$. If one of these is chosen, then α_t will be 0, but there exists a solution such that $\rho(\boldsymbol{\alpha}) > \rho(\mathbf{z}_1)$ using all n hypotheses. (The rest of the hypotheses can be constructed in such a way that there exists no hypothesis which would increase the margin.) Interestingly, AdaBoost would choose $\alpha_t > 0$ and would eventually be able to solve this dilemma (for $\rho(\mathbf{z}_1) > 0$).

Breiman (1998) proposed a modification of AdaBoost – Arc-GV – making it possible to show the asymptotic convergence of $\rho(\boldsymbol{\alpha}^t)$ to the global solution ρ^{lp}:

Breiman's convergence results

Theorem 11.1 (Breiman, 1998)
Choose α_t in each iteration as

$$\alpha_t := \operatorname*{argmin}_{\alpha\in[0,1]} \sum_{i=1}^{m} \exp\left[-\|\boldsymbol{\alpha}^t\|_1 \left(\rho(\mathbf{z}_i,\boldsymbol{\alpha}^t) - \rho(\boldsymbol{\alpha}^{t-1})\right)\right], \tag{11.9}$$

and assume that the base learner always finds the hypothesis $g \in G$, which minimizes the weighted training error with respect to the weights given in (11.6). Then

$$\lim_{t\to\infty} \rho(\boldsymbol{\alpha}^t) = \rho^{\mathrm{lp}} \ ,$$

where ρ^{lp} is the maximum possible margin for a combined classifier from G.

Arc-GV's objective

Note that the algorithm above can be derived from a slightly modified error function:

$$\mathcal{G}_{\mathrm{gv}}(\boldsymbol{\alpha}^t) := \sum_{i=1}^{m} \exp\left[-\|\boldsymbol{\alpha}^t\|_1 \left(\rho(\mathbf{z}_i,\boldsymbol{\alpha}^t) - \rho(\boldsymbol{\alpha}^{t-1})\right)\right]. \tag{11.10}$$

The optimization problem in (11.9) can be solved analytically and we get (Breiman, 1998)

$$\alpha_t = \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) + \log\left(\frac{1-\rho(\boldsymbol{\alpha}^{t-1})}{1+\rho(\boldsymbol{\alpha}^{t-1})}\right) \ . \tag{11.11}$$

Thus, we just have an additional term which only depends on the margin in the last iteration. It always has the opposite sign as $\rho(\boldsymbol{\alpha}^{t-1})$.

toy experiment

The question one might ask now is, whether to use AdaBoost or Arc-GV in practice, or more specifically, whether Arc-GV converges fast enough to profit from its asymptotic properties? To this end, we conduct an experiment with about 25 (appropriate) hypotheses on a data set with 50 patterns[3]. In this finite hypothesis class setting we can easily solve the linear program maximizing the margin and can expect that AdaBoost and also Arc-GV reach the asymptotic regime after a reasonable number of iterations. First, we generate appropriate hypotheses and data sets, i.e. learning problems, such that we get a selection of ρ^{lp} (some negative and some positive). Then we run AdaBoost and Arc-GV on these sets and record $\rho(\boldsymbol{\alpha}^t)$ after 100 and 1000 iterations. Figure 11.2 shows the results of this experiment and we can observe that (a) AdaBoost has problems finding the optimal combination if $\rho^{\mathrm{lp}} < 0$, (b) Arc-GV's convergence does not depend on ρ^{lp}, and (c) for $\rho^{\mathrm{lp}} > 0$ AdaBoost usually converges to the maximum margin solution faster than Arc-GV.

observations

Observation (a) becomes clear from (11.4): $\mathcal{G}(\boldsymbol{\alpha})$ will not converge to 0 and $\|\boldsymbol{\alpha}\|_1$ can be bounded by some value. Thus the asymptotic case cannot be reached,

3. With the `banana shape` dataset from http://www.first.gmd.de/~raetsch/data/banana.html.

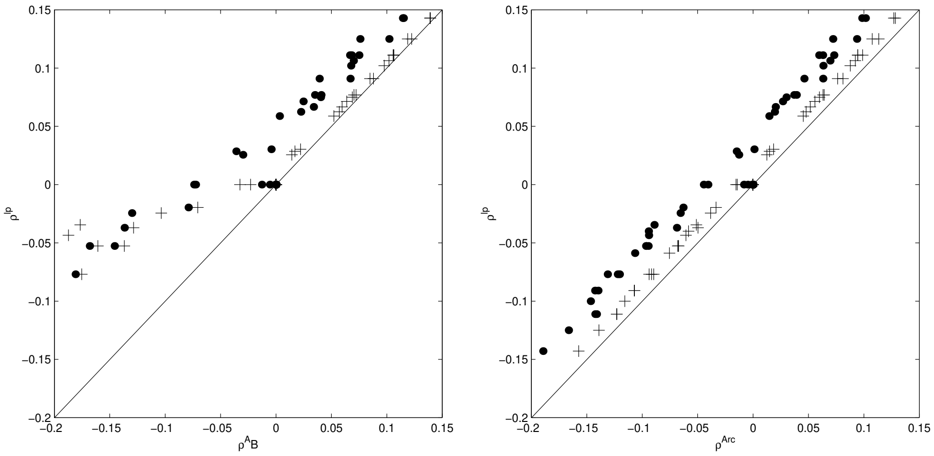

Figure 11.2 AdaBoost's and Arc-GV's margins (on the x–axis) vs. the optimal margin ρ^{lp} (y–axis) for a toy problem (a dot after 100, a '+' after 1000 iterations).

whereas for Arc-GV the optimum is always found. Having a look at equations (11.8) and (11.11) we see that in AdaBoost $\|\boldsymbol{\alpha}\|_1$ will converge faster to ∞ than in Arc-GV (for $\rho^{\mathrm{lp}} > 0$), supporting the third observation.
Moreover, the number of iterations necessary to converge to a *good* solution seems to be reasonable, but for an (almost) *optimal* solution the number of iterations is unexpectedly high. This implies that for real world hypothesis sets the number of iterations needed to find an almost optimal solution can become prohibitive, but we conjecture that in practice a reasonably good approximation to the optimum is provided by both, AdaBoost and Arc-GV.

11.3 ν-Algorithms

For the LP-AdaBoost approach it has been shown for noisy problems, that the generalization performance is usually not as good as the one of AdaBoost (Grove and Schuurmans, 1998; Breiman, 1998; Rätsch, 1998). From Theorem 1.15 this fact becomes clear, as the minimum of the right hand side of inequality (1.50) need not necessarily be achieved with a maximum margin. We now propose an algorithm where we can directly control the number of margin errors and therefore also the contribution of both terms in inequality (1.50) separately (cf. Theorem 11.3). We first consider a small hypothesis class and end up with a linear program – ν-LP-AdaBoost. In subsection 11.3.2 we then combine this algorithm with the ideas from section 11.2 and get two algorithms – ν-Arc and RoBoost – which approximate the ν-LP solution.

11.3.1 ν-LP-AdaBoost

Let us consider the case where we have given a (finite) set $G = \{g : \mathbf{x} \mapsto [-1,1]\}$ of T hypothesis. To find the coefficients $\boldsymbol{\alpha}$ for the combined hypothesis $f(\mathbf{x})$ we extend the LP-AdaBoost algorithm (Grove and Schuurmans, 1998; Rätsch, 1998; Rätsch et al., 1998) and solve the following linear optimization problem, similar in spirit to (Schölkopf et al., 1998d):

$$\begin{array}{lll} \text{maximize} & \rho - \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i & \\ \text{subject to} & \rho(\mathbf{z}_i,\boldsymbol{\alpha}) \geq \rho - \xi_i & \text{for all } 1 \leq i \leq m \\ & \xi_i, \alpha_t, \rho \geq 0 & \text{for all } 1 \leq t \leq T \text{ and } 1 \leq i \leq m \\ & \|\boldsymbol{\alpha}\|_1 = 1 & \end{array} \tag{11.12}$$

This algorithm does not force all margins to be beyond zero and we get a *soft margin* classification with a regularization constant $\frac{1}{\nu m}$. The following proposition shows that ν has an immediate interpretation:

Proposition 11.2
Suppose we run the algorithm given in (11.12) on some data with the resulting optimal $\rho > 0$. Then

1. ν upper-bounds the fraction of margin errors.
2. $1 - \nu$ is an upper bound on the fraction of patterns with a margin larger than ρ.

For an outline of the proof see Figure 11.3 (cf. also Schölkopf et al. (1998d); Graepel et al. (1999)).

SVM connection

Interestingly, if we choose the hypothesis set G with $T = m$ hypotheses to be

$$g_i(\mathbf{x}) = y_i k(\mathbf{x}_i, \mathbf{x}), (i = 1 \ldots m) , \tag{11.13}$$

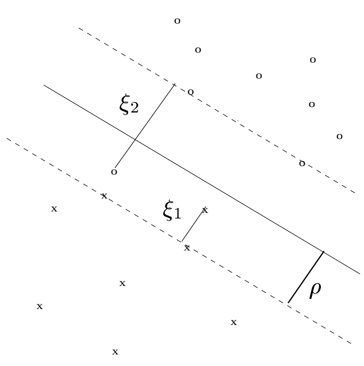

Figure 11.3 Graphical proof of the ν-property. Imagine decreasing ρ, starting from some large value. The first term in $\nu\rho - \frac{1}{m}\sum_{i=1}^{m}\xi_i$ (cf. (11.12)) will decrease proportionally to ν, while the second term will decrease proportionally to the fraction of points outside of the margin area. Hence, ρ will shrink as long as the latter fraction is larger than ν. At the optimum, it therefore must be $\leq \nu$ (Proposition 11.2, 1). Next, imagine increasing ρ, starting from 0. Again, the change in the first term is proportional to ν, but this time, the change in the second term is proportional to the fraction of patterns in the margin area or *exactly on* the margin. Hence, ρ will grow as long as the latter fraction is smaller than ν, eventually leading to Proposition 11.2, 2.

then the algorithm above is equivalent to the Linear Programming Machines (LPMs) (Graepel et al., 1999) with some kernel $k(\cdot,\cdot)$ (except for the additional constraints $\alpha_t \geq 0$). In ν-LP-AdaBoost we are not restricted to a fixed kernel type as we can use arbitrary base hypotheses (and also an arbitrary number of hypotheses – as long as we can fit the problem into a computer).

robustness

Since the slack variables ξ_i only enter the cost function linearly, their absolute size is not important. Loosely speaking, this is due to the fact that for the optimum of the primal objective function, only derivatives wrt. the primal variables matter, and the derivative of a linear function is constant.

In the case of SVMs, where the hypotheses can be thought as vectors in some feature space, this statement can be translated into a precise rule for distorting training patterns without changing the solution: we can move them locally parallel to the above vector. This yields a desirable *resistance* (or robustness) property. For general base hypotheses, it is difficult to state the allowed class of transformations in input space. Nevertheless, note that the algorithm essentially depends on the *number* of outliers, not on the size of the margin error (Schölkopf et al., 1998d).

11.3.2 ν-Arc and RoBoost

Suppose, we have a very large (but finite or at least with finite covering number $\mathcal{N}_\epsilon$) base hypothesis class G. Then it is very difficult to solve (11.12) as (11.5) directly. To this end, we propose two algorithms – ν-Arc and RoBoost – that can approximate the solution of (11.12).

rewriting ν-LP-AdaBoost

The optimal ρ for fixed margins $\rho(\mathbf{z}_i,\boldsymbol{\alpha})$ in (11.12) can be written as

$$\rho_\nu(\boldsymbol{\alpha}) := \operatorname*{argmax}_{\rho\in[0,1]} \left(\rho - \frac{1}{\nu m}\sum_{i=1}^{m}(\rho - \rho(\mathbf{z}_i,\boldsymbol{\alpha}))_+\right). \tag{11.14}$$

where $(\xi)_+ := \max(\xi,0)$. Setting $\xi_i := (\rho_\nu(\boldsymbol{\alpha}) - \rho(\mathbf{z}_i,\boldsymbol{\alpha}))_+$ and subtracting $\frac{1}{\nu m}\sum_{i=1}^m \xi_i$ from the resulting inequality on both sides, yields (for all $1 \leq i \leq m$)

$$\rho(\mathbf{z}_i,\boldsymbol{\alpha}) + \xi_i \geq \rho_\nu(\boldsymbol{\alpha}) \tag{11.15}$$

$$\rho(\mathbf{z}_i,\boldsymbol{\alpha}) + \xi_i - \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i \geq \rho_\nu(\boldsymbol{\alpha}) - \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i\ . \tag{11.16}$$

Two more substitutions are needed to transform the problem into one which can be solved by the AdaBoost algorithm, i.e. (11.6) and (11.7). In particular we have to get rid of the slack variables ξ_i again by absorbing them into quantities similar to $\rho(\mathbf{z}_i,\boldsymbol{\alpha})$ and $\rho(\boldsymbol{\alpha})$.

introducing the soft margin

This works as follows: on the right hand side of (11.16) we have the objective function (cf. (11.12)) and on the left hand side a term that depends nonlinearly on $\boldsymbol{\alpha}$. Defining

$$\tilde{\rho}_\nu(\boldsymbol{\alpha}) := \rho_\nu(\boldsymbol{\alpha}) - \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i \tag{11.17}$$

$$\tilde{\rho}_\nu(\mathbf{z}_i, \boldsymbol{\alpha}) := \rho(\mathbf{z}_i, \boldsymbol{\alpha}) + \xi_i - \frac{1}{\nu m}\sum_{i=1}^{m} \xi_i, \tag{11.18}$$

which we substitute for $\rho(\boldsymbol{\alpha})$ and $\rho(\mathbf{z}, \boldsymbol{\alpha})$ in (11.5), respectively, we obtain a new optimization problem. Note that $\tilde{\rho}_\nu(\boldsymbol{\alpha})$ and $\tilde{\rho}_\nu(\mathbf{z}_i, \boldsymbol{\alpha})$ play the role of a *corrected* margin. We obtain a non-linear min-max problem

$$\begin{array}{lrl} \text{maximize} & \tilde{\rho}(\boldsymbol{\alpha}) & \\ \text{subject to} & \tilde{\rho}(\mathbf{z}_i, \boldsymbol{\alpha}) \geq \tilde{\rho}(\boldsymbol{\alpha}) & \text{for all } 1 \leq i \leq m \\ & \alpha_t \geq 0 & \text{for all } 1 \leq t \leq T \\ & \|\boldsymbol{\alpha}\|_1 = 1 & \end{array}, \tag{11.19}$$

which Arc-GV and AdaBoost can solve approximately (cf. section 11.2). Hence, replacing the margin ρ by $\tilde{\rho}(\mathbf{z}, \boldsymbol{\alpha})$ in equations (11.4), (11.6) and (11.7), we obtain two new algorithms which we refer to as *ν-Arc* and *RoBoost*. Algorithm 11.1 contains the pseudocode of RoBoost. To get the ν-Arc algorithm, the two marked

Algorithm 11.1 : RoBoost

argument: `Training sample,` $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathcal{X}$, $Y = \{y_1, \ldots, y_m\} \subset \{\pm 1\}$
`Number of iterations,` T
returns: `Convex combination of functions from` G.
function $E = \tilde{G}(\boldsymbol{\alpha})$

$\rho_\nu := \underset{\rho \in [-1,1]}{\operatorname{argmax}}\ \rho - \frac{1}{\nu m}\sum_{i=1}^{m}(\rho(\mathbf{z}_i, \boldsymbol{\alpha}) - \rho)_+$

`Set` $\xi_i := (\rho(\mathbf{z}_i, \boldsymbol{\alpha}) - \rho_\nu)_+$ `for all` $i = 1, \ldots, m$

* $E := \sum_{i=1}^{m} \exp\left[-\|\boldsymbol{\alpha}\|_1 \left(\rho(\mathbf{z}_i, \boldsymbol{\alpha}) + \xi_i - \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i\right)\right]$

end
function `RoBoost`(X, Y, T, ν)
for all $i = 1, \ldots, m$
$w^1(\mathbf{z}_i) := 1/m$
endfor
for all t `from` $\{1, \ldots, T\}$
$g_t := L(X, Y, \mathbf{w}^t)$
* $\alpha_t := \underset{\alpha_t \geq 0}{\operatorname{argmax}}\ \tilde{G}(\boldsymbol{\alpha}^t)$
for all i `from` $i = 1, \ldots, m$
$w^{t+1}(\mathbf{z}_i) := \exp\left[-\rho(\mathbf{z}_i, \boldsymbol{\alpha}^t) - (\rho(\mathbf{z}_i, \boldsymbol{\alpha}^t) - \rho_\nu^t)_+\right]$
endfor
$\mathbf{w}^{t+1} := \mathbf{w}^{t+1}/\|\mathbf{w}^{t+1}\|_1$
endfor
return $f = \frac{\sum_{t=1}^{T}\alpha_t g_t}{\sum_{i=1}^{T}\alpha_t}$.
end

L is a learning algorithm that chooses a classifier from G to minimize weighted training error. For ν-`Arc` only the lines marked with '$*$' have to be changed (see text).

lines in Algorithm 11.1 have to be replaced by

$$E := \sum_{i=1}^{m} \exp\left[-\|\boldsymbol{\alpha}\|_1 \left(\rho(\mathbf{z}_i, \boldsymbol{\alpha}) + \xi_i - \frac{1}{\nu m}\sum_{i=1}^{m} \xi_j - \tilde{\rho}_\nu(\boldsymbol{\alpha}^{t-1})\right)\right],$$

where $\xi_i \equiv (\rho(\mathbf{z}_i, \boldsymbol{\alpha}) - \rho_\nu)_+$ and $\alpha_t := \operatorname{argmax}_{\alpha_t \in [0,1]} \tilde{G}(\boldsymbol{\alpha}^t)$.

computational complexity

Clearly, the computational costs for determining the hypothesis weight α_t is considerably higher than for AdaBoost. The implementation given in Algorithm 11.1 is $\mathcal{O}(m^2 \log^2(m))$ of simple operations (maybe there exist better implementations). This might be a problem, if the base hypothesis is very simple and computing α_t takes a relatively high fraction of the computing time. To avoid this problem, one can use similar approximation approaches as used in Schapire and Singer (1998).

the ν bound

We can now state interesting properties for RoBoost and ν-Arc by using Theorem 1.15 that bounds the generalization error $R(f)$ for ensemble methods. In our case $R_\rho(f) \leq \nu$ by construction (i.e. the number of patterns with a margin smaller than ρ, cf. Proposition 11.2), thus we get the following simple reformulation of this bound:

Theorem 11.3

Let $p(\mathbf{x}, y)$ be a distribution over $\mathcal{X} \times [-1, 1]$, and let X be a sample of m examples chosen iid according to p. Suppose the base-hypothesis space G has VC dimension h, and let $\delta > 0$. Then with probability at least $1 - \delta$ over the random choice of the training set X, Y, every function $f \in \operatorname{co}(G)$ generated by the algorithms above satisfies the following bound for all $\nu \in (0, 1)$ with $\rho_\nu > 0$.

$$R(f) \leq \nu + \sqrt{\frac{c}{m}\left(\frac{h \log^2(m/h)}{\rho_\nu^2} + \log\left(\frac{1}{\delta}\right)\right)} \tag{11.20}$$

The tradeoff in minimizing the right hand side between the first and the second term is controlled directly by an easy interpretable regularization parameter ν.

11.4 Experiments

In a first study, we show a set of toy experiments to illustrate the general behavior of Arc-GV and RoBoost. As base hypothesis class G we use RBF networks (Rätsch et al., 1998), and as data a two-class problem generated from several 2D Gauss blobs (cf. footnote 3 on page 212) where we randomly flipped $\sigma = 0\%, 10\%, 20\%, 25\%$ of the labels. We obtained the following results:

- ν-Arc and RoBoost lead to approximately νm patterns that are effectively used in the training of the base learner: Figure 11.4 (upper) shows the fraction of patterns that have high average weights during the learning process (i.e. $\sum_{t=1}^{T} w_t(\mathbf{z}_i) > 1/2m$). We find that the number of the latter increases (almost) linearly with ν. This follows from (11.17) as the (soft) margin of patterns with $\rho(\mathbf{z}, \boldsymbol{\alpha}) < \rho_\nu$ is set to ρ_ν and therefore the weight of those patterns will be the same.

- Both algorithms lead to the fraction ν of margin errors (cf. dashed line in Figure 11.4) exactly as predicted in Proposition 11.2.
- The (estimated) test error, averaged over 10 training sets, exhibits a rather flat minimum in ν (figure 11.4 (lower)). This indicates that just as for ν-SVMs, where corresponding results have been obtained, ν is a well-behaved parameter in the sense that a slight misadjustment it is not harmful.
- The ν algorithms are more robust against label noise than AdaBoost and Arc-GV (which we recover for $\nu = 0$). As illustrated in figure 11.5, also for increasing noise σ the minimum around the optimal ν stays reasonably flat. This coincides with the interpretation of Theorem 11.3 that an optimal ν should increase with the noise level.
- The results of ν-Arc and RoBoost are almost the same. This coincides with the observations in section 11.2 for $\rho > 0$.
- Finally, a good parameter of ν can already be inferred from prior knowledge of the expected error. Setting it to a value similar to the latter provides a good starting point for further optimization (cf. Theorem 11.3).

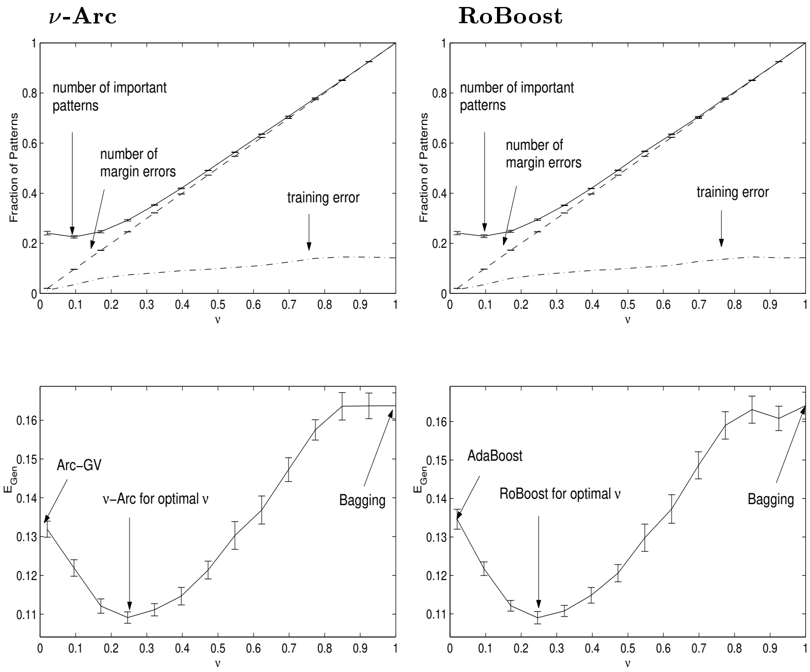

Figure 11.4 Toy experiment ($\sigma = 0$): the upper shows the average fraction of *important* patterns, the av. fraction of margin errors and the av. training error for different values of the regularization constant ν for ν-Arc/RoBoost. The bottom plots show the corresponding generalization error. In both cases the parameter ν allows us to reduce the test errors to values much lower than for the hard margin algorithm (for $\nu = 0$ we recover Arc-GV/AdaBoost and for $\nu = 1$ we get Bagging.)

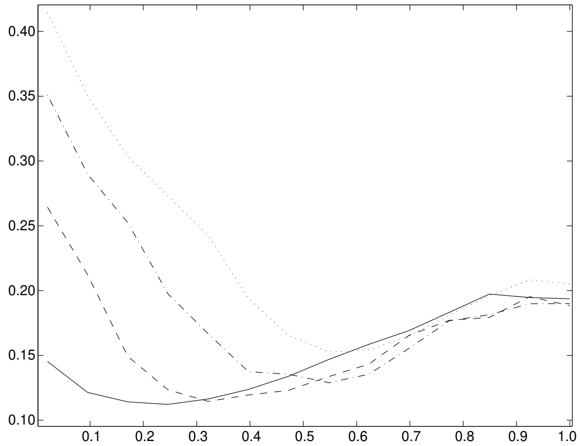

Figure 11.5 Illustration of RoBoost's robustness. Depicted is RoBoost's generalization error over ν for different label noise levels (on the training set; solid=0%, dashed=10%, dot-dashed=20%, and dotted=25%). Also, the minimal generalization error is better than for plain AdaBoost, which achieved 12.3, 13.8, 15.8 and 17.7 respectively. Especially for higher noise levels RoBoost performed relatively better (best results 11.2%, 11.5%, 12.9%, and 15.3% respectively).

Note that for $\nu = 1$ we recover the Bagging algorithm (if we used bootstrap samples), as the weights of all patterns will be the same ($w_t(\mathbf{z}_i) = 1/m$ for all $i = 1, \ldots, m$) and also the hypothesis weights will be constant ($\alpha_t \sim 1/T$ for all $t = 1, \ldots, T$) . This can be seen by setting $\rho_\nu = 1$ and α_t to an arbitrary positive constant (as $\mathcal{G}(\boldsymbol{\alpha})$ will not depend on α_t in this case).

Finally, we present a small comparison on ten benchmark data sets obtained from the UCI benchmark repository (Merz and Murphy, 1998). We analyze the performance of single RBF networks, AdaBoost, ν-Arc and RBF-SVMs (we assume that ν-Arc and RoBoost perform very similar). For AdaBoost and ν-Arc we use RBF networks (Rätsch et al., 1998) as base hypothesis. The model parameters of RBF (number of centers etc.), ν-Arc (ν) and SVMs (σ, C) are optimized using 5-fold cross-validation. More details on the experimental setup can be found in (Rätsch et al., 1998). Figure 11.1 shows the generalization error estimates (after averaging over 100 realizations of the data sets) and the confidence interval. The results of best classifier and the classifiers that are not significantly worse are set in bold face. To test the significance, we used a t-test ($p = 80\%$). On eight out of the ten data sets, ν-Arc performs significantly better than AdaBoost. This clearly shows the superior performance of ν-Arc and supports this soft margin approach for AdaBoost. Furthermore, we find that the performances of ν-Arc and the SVM are comparable. In three cases the SVM performs better and in two cases ν-Arc performs best. Summarizing, AdaBoost is useful for low noise cases, where the classes are separable. ν-Arc/RoBoost extends the applicability of boosting to problems that are difficult to separate and should be applied if the data are noisy.

	RBF	AB	ν-Arc	SVM
Banana	10.8±0.06	12.3±0.07	**10.6±0.05**	11.5±0.07
B.Cancer	27.6±0.47	30.4±0.47	**25.8±0.46**	**26.0±0.47**
Diabetes	24.3±0.19	26.5±0.23	**23.7±0.20**	**23.5±0.17**
German	24.7±0.24	27.5±0.25	24.4±0.22	**23.6±0.21**
Heart	17.6±0.33	20.3±0.34	**16.5±0.36**	**16.0±0.33**
Ringnorm	**1.7±0.02**	1.9±0.03	**1.7±0.02**	**1.7±0.01**
F.Sonar	34.4±0.20	35.7±0.18	34.4±0.19	**32.4±0.18**
Thyroid	**4.5±0.21**	**4.4±0.22**	**4.4±0.22**	4.8±0.22
Titanic	23.3±0.13	**22.6±0.12**	23.0±0.14	**22.4±0.10**
Waveform	10.7±0.11	10.8±0.06	**10.0±0.07**	**9.9±0.04**

Table 11.1 Generalization error estimates and confidence intervals. The best classifiers for a particular data set are marked in bold face (see text).

11.5 Conclusion

We analyzed the AdaBoost algorithm and found that Arc-GV and AdaBoost are suitable for approximating the solution of non-linear min-max problems over huge hypothesis classes. We re-parameterized the LP_{Reg}-AdaBoost algorithm (cf. Grove and Schuurmans (1998); Rätsch et al. (1998); Rätsch (1998)) and introduced a new regularization constant ν that controls the fraction of patterns inside the margin area. The new parameter is highly intuitive and has to be optimized only on a fixed interval $[0, 1]$.

Using the fact that AdaBoost can approximately solve min-max problems, we found a formulation of AdaBoost –RoBoost– and similarly of Arc-GV –ν-Arc– that implements the ν-idea for Boosting by defining an appropriate *soft margin* (cf. also Bennett and Mangasarian (1992)). The present paper extends previous work on regularizing boosting (DOOM (Mason et al., 1999), $\mathrm{AdaBoost}_{\mathrm{Reg}}$ (Rätsch et al., 1998)) and shows the utility and flexibility of the soft margin approach for AdaBoost.

We found empirically that the generalization performance in RoBoost and ν-Arc depends only slightly on the optimal choice of the regularization constant. This makes model selection (e.g. via cross-validation) much easier.

Future work will study the detailed regularization properties of the regularized versions of AdaBoost, in particular how it compares to ν-LP Support Vector Machines.

Acknowledgments

Partial funding from EC STORM project grant (25387) and DFG grant (Ja 379/52,71) is gratefully acknowledged. This work was done while AS and BS were at GMD FIRST.