An Introduction to Boosting and Leveraging

Ron Meir[1] and Gunnar Rätsch[2]

[1] Department of Electrical Engineering, Technion, Haifa 32000, Israel
rmeir@ee.technion.ac.il,
http://www-ee.technion.ac.il/~rmeir
[2] Research School of Information Sciences & Engineering
The Australian National University, Canberra, ACT 0200, Australia
Gunnar.Raetsch@anu.edu.au
http://mlg.anu.edu.au/~raetsch

Abstract. We provide an introduction to theoretical and practical aspects of Boosting and Ensemble learning, providing a useful reference for researchers in the field of Boosting as well as for those seeking to enter this fascinating area of research. We begin with a short background concerning the necessary learning theoretical foundations of weak learners and their linear combinations. We then point out the useful connection between Boosting and the Theory of Optimization, which facilitates the understanding of Boosting and later on enables us to move on to new Boosting algorithms, applicable to a broad spectrum of problems. In order to increase the relevance of the paper to practitioners, we have added remarks, pseudo code, "tricks of the trade", and algorithmic considerations where appropriate. Finally, we illustrate the usefulness of Boosting algorithms by giving an overview of some existing applications. The main ideas are illustrated on the problem of binary classification, although several extensions are discussed.

1 A Brief History of Boosting

The underlying idea of Boosting is to combine simple "rules" to form an ensemble such that the performance of the single ensemble member is improved, i.e. "boosted". Let $h_1, h_2, \ldots, h_T$ be a set of hypotheses, and consider the composite ensemble hypothesis

$$f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}). \tag{1}$$

Here α_t denotes the coefficient with which the ensemble member h_t is combined; both α_t and the learner or hypothesis h_t are to be learned within the Boosting procedure.

The idea of Boosting has its roots in PAC learning (cf. [188]). Kearns and Valiant [101] proved the astonishing fact that learners, each performing only slightly better than random, can be combined to form an arbitrarily good ensemble hypothesis (when enough data is available). Schapire [163] was the first

S. Mendelson, A.J. Smola (Eds.): Advanced Lectures on Machine Learning, LNAI 2600, pp. 118–183, 2003.

to provide a provably polynomial time Boosting algorithm, while [55] were the first to apply the Boosting idea to a real-world OCR task, relying on neural networks as base learners.

The *AdaBoost* (**Ada**ptive **Boost**ing) algorithm by [67, 68, 70] (cf. Algorithm 2.1) is generally considered as a first step towards more practical Boosting algorithms. Very similar to AdaBoost is the *Arcing* algorithm, for which convergence to a linear programming solution can be shown (cf. [27]). Although the Boosting scheme seemed intuitive in the context of algorithmic design, a step forward in transparency was taken by explaining Boosting in terms of a stage-wise gradient descent procedure in an exponential cost function (cf. [27, 63, 74, 127, 153]). A further interesting step towards practical applicability is worth mentioning: large parts of the early Boosting literature persistently contained the misconception that Boosting would not overfit even when running for a large number of iterations. Simulations by [81, 153] on data sets with higher noise content could clearly show overfitting effects, which can only be avoided by regularizing Boosting so as to limit the complexity of the function class (cf. Section 6).

When trying to develop means for achieving robust Boosting it is important to elucidate the relations between Optimization Theory and Boosting procedures (e.g. [71, 27, 48, 156, 149]). Developing this interesting relationship opened the field to new types of Boosting algorithms. Among other options it now became possible to rigorously define Boosting algorithms for regression (cf. [58, 149]), multi-class problems [3, 155], unsupervised learning (cf. [32, 150]) and to establish convergence proofs for Boosting algorithms by using results from the Theory of Optimization. Further extensions to Boosting algorithms can be found in [32, 150, 74, 168, 169, 183, 3, 57, 129, 53].

Recently, Boosting strategies have been quite successfully used in various real-world applications. For instance [176] and earlier [55] and [112] used boosted ensembles of neural networks for OCR. [139] proposed a non-intrusive monitoring system for assessing whether certain household appliances consume power or not. In [49] Boosting was used for tumor classification with gene expression data. For further applications and more details we refer to the Applications Section 8.3 and to `http://www.boosting.org/applications`.

Although we attempt a full and balanced treatment of most available literature, naturally the presentation leans in parts towards the authors' own work. In fact, the Boosting literature, as will become clear from the bibliography, is so extensive, that a full treatment would require a book length treatise. The present review differs from other reviews, such as the ones of [72, 165, 166], mainly in the choice of the presented material: we place more emphasis on robust algorithms for Boosting, on connections to other margin based approaches (such as support vector machines), and on connections to the Theory of Optimization. Finally, we discuss applications and extensions of Boosting.

The content of this review is organized as follows. After presenting some basic ideas on Boosting and ensemble methods in the next section, a brief overview of the underlying theory is given in Section 3. The notion of the margin and its

connection to successful Boosting is analyzed in Section 4, and the important link between Boosting and Optimization Theory is discussed in Section 5. Subsequently, in Section 6, we present several approaches to making Boosting more robust, and proceed in Section 7 by presenting several extensions of the basic Boosting framework. Section 8.3 then presents a short summary of applications, and Section 9 summarizes the review and presents some open problems.

2 An Introduction to Boosting and Ensemble Methods

This section includes a brief definition of the general learning setting addressed in this paper, followed by a discussion of different aspects of ensemble learning.

2.1 Learning from Data and the PAC Property

We focus in this review (except in Section 7) on the problem of *binary classification* [50]. The task of binary classification is to find a rule (hypothesis), which, based on a set of observations, assigns an object to one of two classes. We represent objects as belonging to an input space $\mathbb{X}$, and denote the possible outputs of a hypothesis by $\mathbb{Y}$. One possible formalization of this task is to estimate a function $f : \mathbb{X} \to \mathbb{Y}$, using input-output training data pairs generated independently at random from an unknown probability distribution $P(\boldsymbol{x}, y)$,

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{-1, +1\}$$

such that f will correctly predict unseen examples $(\boldsymbol{x}, y)$. In the case where $\mathbb{Y} = \{-1, +1\}$ we have a so-called *hard classifier* and the label assigned to an input $\boldsymbol{x}$ is given by $f(\boldsymbol{x})$. Often one takes $\mathbb{Y} = \mathbb{R}$, termed a *soft classifier*, in which case the label assignment is performed according to $\mathrm{sign}(f(\boldsymbol{x}))$. The true performance of a classifier f is assessed by

$$L(f) = \int \lambda(f(\boldsymbol{x}), y) dP(\boldsymbol{x}, y), \tag{2}$$

where λ denotes a suitably chosen loss function. The risk $L(f)$ is often termed the *generalization error* as it measures the expected loss with respect to examples which were not observed in the training set. For binary classification one often uses the so-called 0/1-loss $\lambda(f(\boldsymbol{x}), y) = \mathbf{I}(yf(\boldsymbol{x}) \leq 0)$, where $\mathbf{I}(E) = 1$ if the event E occurs and zero otherwise. Other loss functions are often introduced depending on the specific context.

Unfortunately the risk cannot be minimized directly, since the underlying probability distribution $P(\boldsymbol{x}, y)$ is unknown. Therefore, we have to try to estimate a function that is *close* to the optimal one based on the available information, i.e. the training sample and properties of the function class F from which the solution f is chosen. To this end, we need what is called an induction principle. A particularly simple one consists of approximating the minimum of the risk (2) by the minimum of the *empirical risk*

$$\hat{L}(f) = \frac{1}{N} \sum_{n=1}^{N} \lambda(f(\boldsymbol{x}_n), y_n). \tag{3}$$

From the law of large numbers (e.g. [61]) one expects that $\hat{L}(f) \to L(f)$ as $N \to \infty$. However, in order to guarantee that the function obtained by minimizing $\hat{L}(f)$ also attains asymptotically the minimum of $L(f)$ a stronger condition is required. Intuitively, the complexity of the class of functions F needs to be controlled, since otherwise a function f with arbitrarily low error on the sample may be found, which, however, leads to very poor generalization. A sufficient condition for preventing this phenomenon is the requirement that $\hat{L}(f)$ converge *uniformly* (over F) to $L(f)$; see [50, 191, 4] for further details.

While it is possible to provide conditions for the learning machine which ensure that asymptotically (as $N \to \infty$) the empirical risk minimizer will perform optimally, for small sample sizes large deviations are possible and *overfitting* might occur. Then a small generalization error cannot be obtained by simply minimizing the training error (3). As mentioned above, one way to avoid the overfitting dilemma – called *regularization* – is to *restrict* the size of the function class F that one chooses the function f from [190, 4]. The intuition, which will be formalized in Section 3, is that a "simple" function that explains most of the data is preferable to a complex one which fits the data very well (Occam's razor, e.g. [21]).

For Boosting algorithms which generate a complex composite hypothesis as in (1), one may be tempted to think that the complexity of the resulting function class would increase dramatically when using an ensemble of many learners. However this is provably not the case under certain conditions (see e.g. Theorem 3), as the ensemble complexity saturates (cf. Section 3). This insight may lead us to think that since the complexity of Boosting is limited we might not encounter the effect of overfitting at all. However when using Boosting procedures on noisy real-world data, it turns out that *regularization* (e.g. [103, 186, 143, 43]) is mandatory if overfitting is to be avoided (cf. Section 6). This is in line with the general experience in statistical learning procedures when using complex non-linear models, for instance for neural networks, where a regularization term is used to appropriately limit the complexity of the models (e.g. [2, 159, 143, 133, 135, 187, 191, 36]).

Before proceeding to discuss ensemble methods we briefly review the strong and weak PAC models for learning binary concepts [188]. Let S be a sample consisting of N data points $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, where $\boldsymbol{x}_n$ are generated independently at random from some distribution $P(\boldsymbol{x})$ and $y_n = f(\boldsymbol{x}_n)$, and f belongs to some known class F of binary functions. A *strong PAC learning algorithm* has the property that for every distribution P, every $f \in \mathsf{F}$ and every $0 \leq \epsilon, \delta \leq 1/2$ with probability larger than $1-\delta$, the algorithm outputs a hypothesis h such that $\Pr[h(\boldsymbol{x}) \neq f(\boldsymbol{x})] \leq \epsilon$. The running time of the algorithm should be polynomial in $1/\varepsilon, 1/\delta, n, d$, where d is the dimension (appropriately defined) of the input space. A *weak PAC learning algorithm* is defined analogously, except that it is only required to satisfy the conditions for *particular* ε and δ, rather than all pairs. Various extensions and generalization of the basic PAC concept can be found in [87, 102, 4].

2.2 Ensemble Learning, Boosting, and Leveraging

Consider a combination of hypotheses in the form of (1). Clearly there are many approaches for selecting both the coefficients α_t and the base hypotheses h_t. In the so-called *Bagging* approach [25], the hypotheses $\{h_t\}_{t=1}^T$ are chosen based on a set of T bootstrap samples, and the coefficients α_t are set to $\alpha_t = 1/T$ (see e.g. [142] for more refined choices). Although this algorithm seems somewhat simplistic, it has the nice property that it tends to reduce the variance of the overall estimate $f(\boldsymbol{x})$ as discussed for regression [27, 79] and classification [26, 75, 51]. Thus, Bagging is quite often found to improve the performance of complex (unstable) classifiers, such as neural networks or decision trees [25, 51].

For Boosting the combination of the hypotheses is chosen in a more sophisticated manner. The evolution of the AdaBoost algorithm can be understood from Figure 1. The intuitive idea is that examples that are misclassified get higher weights in the next iteration, for instance the examples near the decision boundary are usually harder to classify and therefore get high weights after a few iterations. This idea of iterative reweighting of the training sample is essential to Boosting.

More formally, a non-negative weighting $\mathbf{d}^{(t)} = (d_1^{(t)}, \ldots, d_N^{(t)})$ is assigned to the data at step t, and a weak learner h_t is constructed based on $\mathbf{d}^{(t)}$. This weighting is updated at each iteration t according to the weighted error incurred by the weak learner in the last iteration (see Algorithm 2.1). At each step t, the weak learner is required to produce a small weighted empirical error defined by

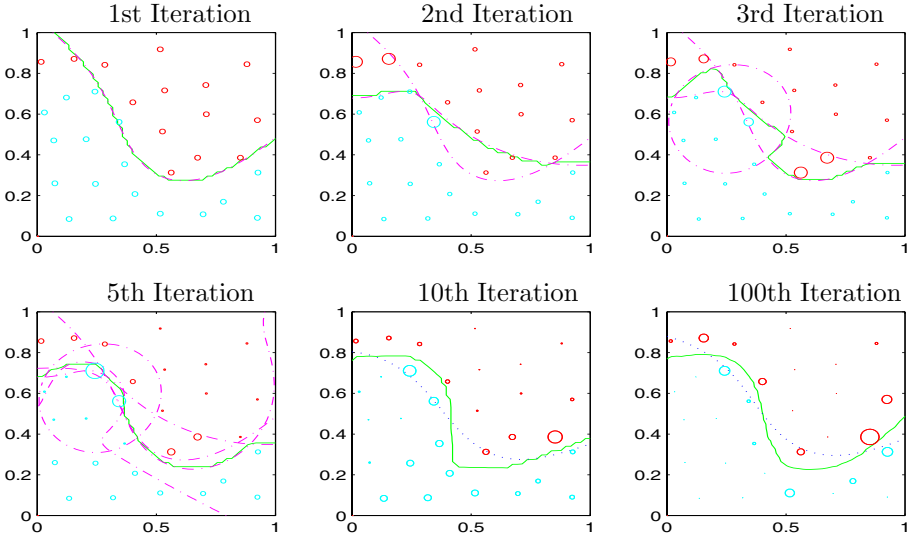

Fig. 1. Illustration of AdaBoost on a 2D toy data set: The color indicates the label and the diameter is proportional to the weight of the examples in the first, second, third, 5th, 10th and 100th iteration. The dashed lines show the decision boundaries of the single classifiers (up to the 5th iteration). The solid line shows the decision line of the combined classifier. In the last two plots the decision line of Bagging is plotted for a comparison. (Figure taken from [153].)

Algorithm 2.1 The AdaBoost algorithm [70].

1. **Input:** $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of Iterations T
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$
3. **Do for** $t = 1, \ldots, T$,
   a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto \{-1, +1\}$, i.e. $h_t = \mathcal{L}(S, \mathbf{d}^{(t)})$
   b) Calculate the weighted training error ε_t of h_t:
   $$\varepsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n)) \ ,$$
   c) Set:
   $$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$
   d) Update weights:
   $$d_n^{(t+1)} = d_n^{(t)} \exp\left\{-\alpha_t y_n h_t(\boldsymbol{x}_n)\right\} / Z_t \ ,$$
   where Z_t is a normalization constant, such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\varepsilon_t = 0$ or $\varepsilon_t \geq \frac{1}{2}$ and set $T = t - 1$.
5. **Output:** $f_T(\boldsymbol{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(\boldsymbol{x})$

$$\varepsilon_t(h_t, \mathbf{d}^{(t)}) = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n)). \tag{4}$$

After selecting the hypothesis h_t, its weight α_t is computed such that it minimizes a certain loss function (cf. step (3c)). In AdaBoost one minimizes

$$\mathrm{G}^{AB}(\alpha) = \sum_{n=1}^{N} \exp\left\{-y_n \left(\alpha h_t(\boldsymbol{x}_n) + f_{t-1}(\boldsymbol{x}_n)\right)\right\}, \tag{5}$$

where f_{t-1} is the combined hypothesis of the previous iteration given by

$$f_{t-1}(\boldsymbol{x}_n) = \sum_{r=1}^{t-1} \alpha_r h_r(\boldsymbol{x}_n). \tag{6}$$

Another very effective approach, proposed in [74], is the *LogitBoost algorithm*, where the cost function analogous to (5) is given by

$$\mathrm{G}^{LR}(\alpha) = \sum_{n=1}^{N} \log\left\{1 + \exp\left(-y_n(\alpha h_t(\boldsymbol{x}_n) + f_{t-1}(\boldsymbol{x}_n))\right)\right\} \ . \tag{7}$$

An important feature of (7) as compared to (5) is that the former increases only linearly for negative values of $y_n(\alpha h_t(\boldsymbol{x}_n) + f_{t-1}(\boldsymbol{x}_n))$, while the latter increases exponentially. It turns out that this difference is important in noisy situations, where AdaBoost tends to focus too much on outliers and noisy data. In Section 6 we will discuss other techniques to approach this problem. Some further details concerning LogitBoost will be given in Section 5.2.

For AdaBoost it has been shown [168] that α_t in (5) can be computed analytically leading to the expression in step (3c) of Algorithm 2.1. Based on the new combined hypothesis, the weighting $\mathbf{d}$ of the sample is updated as in step (3d) of Algorithm 2.1. The initial weighting $\mathbf{d}^{(1)}$ is chosen uniformly: $d_n^{(1)} = 1/N$.

The so-called *Arcing algorithms* proposed in [27] are similar to AdaBoost, but use a different loss function.[1] Each loss function leads to different weighting schemes of the examples and hypothesis coefficients. One well-known example is *Arc-X4* where one uses a fourth order polynomial to compute the weight of the examples. Arcing is the predecessor of the more general leveraging algorithms discussed in Section 5.2. Moreover, for one variant, called *Arc-GV*, it has been shown [27] to find the linear combination that solves a linear optimization problem (LP) (cf. Section 4).

The AdaBoost algorithm presented above is based on using binary (hard) weak learners. In [168] and much subsequent work, real-valued (soft) weak learners were used. One may also find the hypothesis weight α_t and the hypothesis h_t in parallel, such that (5) is minimized. Many other variants of Boosting algorithms have emerged over the past few years, many of which operate very similarly to AdaBoost, even though they often do not possess the PAC-boosting property. The PAC-boosting property refers to schemes which are able to guarantee that weak learning algorithms are indeed transformed into strong learning algorithms in the sense described in Section 2.1. Duffy and Helmbold [59] reserve the term 'boosting' for algorithms for which the PAC-boosting property can be proved to hold, while using 'leveraging' in all other cases. Since we are not overly concerned with PAC learning per se in this review, we use the terms 'boosting' and 'leveraging' interchangeably.

In principle, any leveraging approach iteratively selects one hypothesis $h_t \in H$ at a time and then updates their weights; this can be implemented in different ways. Ideally, given $\{h_1, \ldots, h_t\}$, one solves the optimization problem for all hypothesis coefficients $\{\alpha_1, \ldots, \alpha_t\}$, as proposed by [81, 104, 48]. In contrast to this, the greedy approach is used by the original AdaBoost/Arc-GV algorithm as discussed above: only the weight of the *last* hypothesis is selected, while minimizing some appropriate cost function [27, 74, 127, 153]. Later we will show in Sections 4 and 5 that this relates to barrier optimization techniques [156] and coordinate descent methods [197, 151]. Additionally, we will briefly discuss relations to information geometry [24, 34, 110, 104, 39, 147, 111] and column generation techniques (cf. Section 6.2).

An important issue in the context of the algorithms discussed in this review pertains to the construction of the weak learner (e.g. step (3a) in Algorithm 2.1).

[1] Any convex loss function is allowed that goes to infinity when the margin goes to minus infinity and to zero when the margin goes to infinity.

At step t, the weak learner is constructed based on the weighting $\mathbf{d}^{(t)}$. There are basically two approaches to taking the weighting into consideration. In the first approach, we assume that the learning algorithm can operate with reweighted examples. For example, if the weak learner is based on minimizing a cost function (see Section 5), one can construct a revised cost function which assigns a weight to each of the examples, so that heavily weighted examples are more influential. For example, for the least squares error one may attempt to minimize $\sum_{n=1}^{N} d_n^{(t)}(y_n - h(\boldsymbol{x}_n))^2$. However, not all weak learners are easily adaptable to the inclusion of weights. An approach proposed in [68] is based on resampling the data with replacement based on the distribution $\mathbf{d}^{(t)}$. The latter approach is more general as it is applicable to any weak learner; however, the former approach has been more widely used in practice. Friedman [73] has also considered sampling based approaches within the general framework described in Section 5. He found that in certain situations (small samples and powerful weak learners) it is advantageous to sample a subset of the data rather than the full data set itself, where different subsets are sampled at each iteration of the algorithm. Overall, however, it is not yet clear whether there is a distinct advantage to using either one of the two approaches.

3 Learning Theoretical Foundations of Boosting

3.1 The Existence of Weak Learners

We have seen that AdaBoost operates by forming a re-weighting $\mathbf{d}$ of the data at each step, and constructing a base learner based on $\mathbf{d}$. In order to gauge the performance of a base learner, we first define a *baseline* learner.

Definition 1. *Let $\mathbf{d} = (d_1, \ldots, d_N)$ be a probability weighting of the data points S. Let S_+ be the subset of the positively labeled points, and similarly for S_-. Set $D_+ = \sum_{n:y_n=+1} d_n$ and similarly for D_-. The baseline classifier f_{BL} is defined as $f_{BL}(x) = \operatorname{sign}(D_+ - D_-)$ for all x. In other words, the baseline classifier predicts $+1$ if $D_+ \geq D_-$ and -1 otherwise. It is immediately obvious that for any weighting $\mathbf{d}$, the error of the baseline classifier is at most $1/2$.*

The notion of weak learner has been introduced in the context of PAC learning at the end of Section 2.1. However, this definition is too limited for most applications. In the context of this review, we define a weak learner as follows. A learner is a weak learner for sample S if, given any weighting $\mathbf{d}$ on S, it is able to achieve a weighted classification error (see (4)) which is strictly smaller than $1/2$.

A key ingredient of Boosting algorithms is a weak learner which is required to exist in order for the overall algorithm to perform well. In the context of binary classification, we demand that the weighted empirical error of each weak learner is strictly smaller than $\frac{1}{2} - \frac{1}{2}\gamma$, where γ is an *edge* parameter quantifying the deviation of the performance of the weak learner from the baseline classifier introduced in Definition 1. Consider a weak learner that outputs a binary classifier h based on a data set, $S = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ each pair $(\boldsymbol{x}_n, y_n)$ of which is weighted by a non-negative weight d_n. We then demand that

$$\varepsilon(h, \mathbf{d}) = \sum_{n=1}^{N} d_n \mathbf{I}[y_n \neq h(\boldsymbol{x}_n)] \leq \frac{1}{2} - \frac{1}{2}\gamma, \qquad (\gamma > 0). \tag{8}$$

For simple weak learners, it may not be possible to find a strictly positive value of γ for which (8) holds without making any assumptions about the data. For example, consider the two-dimensional `xor` problem with $N = 4$, $\boldsymbol{x}_1 = (-1, -1), \boldsymbol{x}_2 = (+1, +1), \boldsymbol{x}_3 = (-1, +1), \boldsymbol{x}_4 = (+1, +1)$, and corresponding labels $\{-1, -1, +1, +1\}$. If the weak learner h is restricted to be an axis-parallel half-space, it is clear that no such h can achieve an error smaller than $1/2$ for a uniform weighting over the examples.

We consider two situations, where it is possible to establish the strict positivity of γ, and to set a lower bound on its value. Consider a mapping f from the binary cube $\mathbb{X} = \{-1, +1\}^d$ to $\mathbb{Y} = \{-1, 1\}$, and assume the true labels y_n are given by $y_n = f(\boldsymbol{x}_n)$. We wish to approximate f by combinations of binary hypotheses h_t belonging to H. Intuitively we expect that a large edge γ can be achieved if we can find a weak hypothesis h which correlates well with f (cf. Section 4.1). Let H be a class of binary hypotheses (Boolean functions), and let D be a distribution over $\mathbb{X}$. The correlation between f and H, with respect to D, is given by $C_{\mathsf{H},D}(f) = \sup_{h \in \mathsf{H}} \mathbf{E}_D\{f(\boldsymbol{x})h(\boldsymbol{x})\}$. The distribution-free correlation between f and H is given by $C_{\mathsf{H}}(f) = \inf_D C_{\mathsf{H},D}(f)$. [64] shows that if $T > 2\log(2)dC_{\mathsf{H}}(f)^{-2}$ then f can be represented exactly as $f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} h_t(\boldsymbol{x})\right)$. In other words, if H is highly correlated with the target function f, then f can be exactly represented as a convex combinations of a small number of functions from H. Hence, after a sufficiently large number of Boosting iterations, the empirical error can be expected to approach zero. Interestingly, this result is related to the Min-Max theorem presented in Section 4.1.

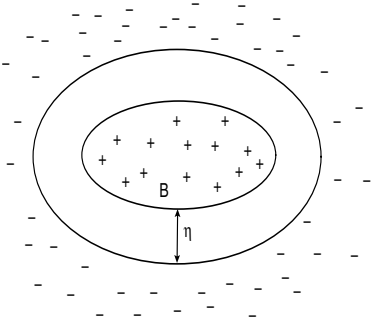

Fig. 2. A single convex set containing the positively labeled examples separated from the negative examples by a gap of η.

The results of [64] address only the case of Boolean functions and a known target function f. An important question that arises relates to the establishment of *geometric* conditions for which the existence of a weak learner can be guaranteed. Consider the case where the input patterns $\boldsymbol{x}$ belong to $\mathbb{R}^d$, and let H, the class of weak learners, consisting of linear classifiers of the form $\operatorname{sign}(\mathbf{w}^\top \boldsymbol{x} + b)$. It is not hard to show [122], that for any distribution $\mathbf{d}$ over a training set of

distinct points, $\varepsilon(h, \mathbf{d}) \leq \frac{1}{2} - c/N$, where c is an absolute constant. In other words, for any $\mathbf{d}$ one can find a linear classifier that achieves an edge of at least c/N. However, as will become clear in Section 3.4 such an advantage does not suffice to guarantee good generalization. This is not surprising, since the claim holds even for arbitrarily labeled points, for which no generalization can be expected. In order to obtain a larger edge, some assumptions need to be made about the data. Intuitively, we expect that situations where the positively and negatively labeled points are well separated are conducive to achieving a large edge by linear classifiers. Consider, for example, the case where the positively labeled points are enclosed within a compact convex region in $K \subset \mathbb{R}^d$, while the remaining points are located outside of this region, such that the distance between the oppositely labeled points is at least η, for some $\eta > 0$ (see Figure 2). It is not hard to show that in this case [122] $\gamma \geq \gamma_0 > 0$, namely the edge is strictly larger than zero, *independently* of the sample size (γ_0 is related to the number of faces of the smallest convex polytope that covers only the positively labeled points, see also discussion in Section 4.2).

In general situations, the data may be strongly overlapping. In order to deal with such cases, much more advanced tools need to be wielded based on the theory of Geometric Discrepancy [128]. The technical details of this development are beyond the scope of this paper. The interested reader is referred to [121, 122] for further details. In general, there has not been much work on establishing geometric conditions for the existence of weak learners for other types of classifiers.

3.2 Convergence of the Training Error to Zero

As we have seen in the previous section, it is possible under appropriate conditions to guarantee that the weighted empirical error of a weak learner is smaller than $\frac{1}{2} - \frac{1}{2}\gamma$, $\gamma > 0$. We now show that this condition suffices to ensure that the empirical error of the composite hypothesis converges to zero as the number of iterations increases. In fact, anticipating the generalization bounds in Section 3.4, we present a somewhat stronger result. We establish the claim for the AdaBoost algorithm; similar claims can be proven for other Boosting algorithms (cf. Section 4.1 and [59]).

Keeping in mind that we may use a real-valued function f for classification, we often want to take advantage of the actual value of f, even though classification is performed using $\text{sign}(f)$. The actual value of f contains information about the confidence with which $\text{sign}(f)$ is predicted (e.g. [4]). For binary classification ($y \in \{-1, +1\}$) and $f \in \mathbb{R}$ we define the *margin* of f at example $(\boldsymbol{x}_n, y_n)$ as

$$\rho_n(f) = y_n f(\boldsymbol{x}_n). \tag{9}$$

Consider the following function defined for $0 \leq \theta \leq \frac{1}{2}$,

$$\varphi_\theta(z) = \begin{cases} 1 & \text{if } z \leq 0, \\ 1 - z/\theta & \text{if } 0 < z \leq \theta, \\ 0 & \text{otherwise .} \end{cases}$$

Let f be a real-valued function taking values in $[-1, +1]$. The empirical *margin error* is defined as

$$\hat{L}^{\theta}(f) = \frac{1}{N} \sum_{n=1}^{N} \varphi_{\theta}(y_n f(\boldsymbol{x}_n)). \tag{10}$$

It is obvious from the definition that the classification error, namely the fraction of misclassified examples, is given by $\theta = 0$, i.e. $\hat{L}(f) = \hat{L}^0(f)$. In addition, $\hat{L}^{\theta}(f)$ is monotonically increasing in θ. We note that one often uses the so-called 0/1-margin error defined by

$$\tilde{L}^{\theta}(f) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{I}(y_n f(\boldsymbol{x}_n) \leq \theta).$$

Noting that $\varphi_{\theta}(yf(\boldsymbol{x})) \leq \mathbf{I}(yf(\boldsymbol{x}) \leq \theta)$, it follows that $\hat{L}^{\theta}(f) \leq \tilde{L}^{\theta}(f)$. Since we use $\hat{L}^{\theta}(f)$ as part of an upper bound on the generalization error in Section 3.4, the bound we obtain using it is tighter than would be obtained using $\tilde{L}^{\theta}(f)$.

Theorem 1 ([167]). *Consider AdaBoost as described in Algorithm 2.1. Assume that at each round t, the weighted empirical error satisfies $\varepsilon(h_t, \mathbf{d}^{(t)}) \leq \frac{1}{2} - \frac{1}{2}\gamma_t$. Then the empirical margin error of the composite hypothesis f_T obeys*

$$\hat{L}^{\theta}(f_T) \leq \prod_{t=1}^{T} (1 - \gamma_t)^{\frac{1-\theta}{2}} (1 + \gamma_t)^{\frac{1+\theta}{2}}. \tag{11}$$

Proof. We present a proof from [167] for the case where $h_t \in \{-1, +1\}$. We begin by showing that for *every* $\{\alpha_t\}$

$$\tilde{L}^{\theta}(f_T) \leq \exp\left(\theta \sum_{t=1}^{T} \alpha_t\right) \left(\prod_{t=1}^{T} Z_t\right). \tag{12}$$

By definition

$$\begin{aligned} Z_t &= \sum_{n=1}^{N} d_n^{(t)} e^{-y_n \alpha_t h_t(x_n)} \\ &= \sum_{n: y_n = h_t(x_n)} d_n^{(t)} e^{-\alpha_t} + \sum_{n: y_n \neq h_t(x_n)} d_n^{(t)} e^{\alpha_t} \\ &= (1 - \varepsilon_t) e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}. \end{aligned}$$

From the definition of f_T it follows that

$$yf_T(x) \leq \theta \quad \Rightarrow \quad \exp\left(-y \sum_{t=1}^{T} \alpha_t h_t(x) + \theta \sum_{t=1}^{T} \alpha_t\right) \geq 1,$$

which we rewrite as

$$\mathbf{I}[Y f_T(X) \leq \theta] \leq \exp\left(-y \sum_{t=1}^{T} \alpha_t h_t(x) + \theta \sum_{t=1}^{T} \alpha_t\right). \tag{13}$$

Note that

$$\begin{aligned} d_n^{(T+1)} &= \frac{d_n^{(T)} \exp\left(-\alpha_T y_n h_T(x_n)\right)}{Z_T} \\ &= \frac{\exp\left(-\sum_{t=1}^{T} \alpha_t y_n h_t(x_n)\right)}{N \prod_{t=1}^{T} Z_t} \qquad \text{(by induction).} \end{aligned} \tag{14}$$

Using (13) and (14) we find that

$$\begin{aligned} \tilde{L}^{\theta}(f) &= \frac{1}{N} \sum_{n=1}^{N} I[y_n f_T(x_n) \leq \theta] \\ &\leq \frac{1}{N} \sum_{n=1}^{N} \left[\exp\left(-y_n \sum_{t=1}^{T} \alpha_t h_t(x_n) + \theta \sum_{t=1}^{T} \alpha_t\right)\right] \\ &= \frac{1}{N} \exp\left(\theta \sum_{t=1}^{T} \alpha_t\right) \sum_{n=1}^{N} \exp\left(-y_n \sum_{t=1}^{T} \alpha_t h_t(x_n)\right) \\ &= \exp\left(\theta \sum_{t=1}^{T} \alpha_t\right) \left(\prod_{t=1}^{T} Z_t\right) \underbrace{\sum_{n=1}^{N} d_n^{(T+1)}}_{=1} \\ &= \exp\left(\theta \sum_{t=1}^{T} \alpha_t\right) \left(\prod_{t=1}^{T} Z_t\right). \end{aligned}$$

Next, set $\alpha_t = (1/2)\log((1-\varepsilon_t)/\varepsilon_t)$ as in Algorithm 2.1, which easily implies that

$$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}.$$

Substituting this result into (12) we find that

$$\tilde{L}^{\theta}(f) \leq \prod_{t=1}^{T} \sqrt{4\varepsilon_t^{1-\theta}(1-\varepsilon_t)^{1+\theta}}$$

which yields the desired result upon recalling that $\varepsilon_t = 1/2 - \gamma_t/2$, and noting that $\hat{L}^{\theta}(f) \leq \tilde{L}^{\theta}(f)$.

In the special case that $\theta = 0$, i.e. one considers the training error, we find that

$$\hat{L}(f_T) \leq e^{-\sum_{t=1}^{T} \gamma_t^2/2},$$

from which we infer that the condition $\sum_{t=1}^{T} \gamma_t^2 \to \infty$ suffices to guarantee that $\hat{L}(f_T) \to 0$. For example, the condition $\gamma_t \geq c/\sqrt{t}$ suffices for this.[2] Clearly

[2] When one uses binary valued hypotheses, then $\gamma_t \geq c/t$ is already sufficient to achieve a margin of at least zero on all training examples (cf. [146], Lemma 2.2, 2.3).

this holds if $\gamma_t \geq \gamma_0 > 0$ for some positive constant γ_0. In fact, in this case $\hat{L}^\theta(f_T) \to 0$ for any $\theta \leq \gamma_0/2$.

In general, however, one may not be interested in the convergence of the empirical error to zero, due to overfitting (see discussion in Section 6). Sufficient conditions for convergence of the error to a nonzero value can be found in [121].

3.3 Generalization Error Bounds

In this section we consider binary classifiers only, namely $f : \mathbb{X} \mapsto \{-1, +1\}$. A learning algorithm can be viewed as a procedure for mapping any data set $S = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ to some hypothesis h belonging to a hypothesis class H consisting of functions from $\mathbb{X}$ to $\{-1, +1\}$. In principle, one is interested in quantifying the performance of the hypothesis $\hat{f}$ on future data. Since the data S consists of randomly drawn pairs $(\boldsymbol{x}_n, y_n)$, both the data-set and the generated hypothesis $\hat{f}$ are random variables. Let $\lambda(y, f(\boldsymbol{x}))$ be a loss function, measuring the loss incurred by using the hypothesis f to classify input $\boldsymbol{x}$, the true label of which is y. As in Section 2.1 the expected loss incurred by a hypothesis f is given by $L(h) = \mathbf{E}\{\lambda(y, f(\boldsymbol{x}))\}$, where the expectation is taken with respect to the *unknown* probability distribution generating the pairs $(\boldsymbol{x}_n, y_n)$. In the sequel we will consider the case of binary classification and 0/1-loss

$$\lambda(y, f(\boldsymbol{x})) = \mathbf{I}[y \neq f(\boldsymbol{x})],$$

where $\mathbf{I}[\mathcal{E}] = 1$, if the event $\mathcal{E}$ occurs and zero otherwise. In this case it is easy to see that $L(f) = \Pr[y \neq h(\boldsymbol{x})]$, namely the probability of misclassification.

A classic result by Vapnik and Chervonenkis relates the empirical classification error of a binary hypothesis f, to the probability of error $\Pr[y \neq f(\boldsymbol{x})]$. For binary functions f we use the notation $P(y \neq f(\boldsymbol{x}))$ for the probability of error and $\hat{P}(y \neq f(\boldsymbol{x})$ for the empirical classification error. Before presenting the result, we need to define the VC-dimension of a class of binary hypotheses F. Consider a set of N points $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$. Each binary function f defines a dichotomy $(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)) \in \{-1, +1\}^N$ on these points. Allowing f to run over all elements of F, we generate a subset of the binary N-dimensional cube $\{-1, +1\}^N$, denoted by F_X, i.e. $F_X = \{(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)) : f \in \mathsf{F}\}$. The VC-dimension of F, VCdim(F), is defined as the maximal cardinality of the set of points $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ for which $|F_X| = 2^N$. Good estimates of the VC dimension are available for many classes of hypotheses. For example, in the d-dimensional space $\mathbb{R}^d$ one finds for hyperplanes a VC dimension of $d + 1$, while for rectangles the VC dimension is $2d$. Many more results and bounds on the VC dimension of various classes can be found in [50] and [4].

We present an improved version of the classic VC bound, taken from [11].

Theorem 2 ([192]). *Let F be a class of $\{-1, +1\}$-valued functions defined over a set $\mathbb{X}$. Let P be a probability distribution on $\mathbb{X} \times \{-1, +1\}$, and suppose that N-samples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ are generated independently at random according to P. Then, there is an absolute constant c, such that for any integer N, with probability at least $1 - \delta$ over samples of length N, every $f \in \mathsf{F}$ satisfies*

$$P(y \neq f(\boldsymbol{x})) \leq \hat{P}(y \neq f(\boldsymbol{x})) + c\sqrt{\frac{\mathrm{VCdim}(\mathsf{F}) + \log(1/\delta)}{N}}. \tag{15}$$

We comment that the original bound in [192] contained an extra factor of $\log N$ multiplying $\mathrm{VCdim}(\mathsf{F})$ in (15).

The *finite-sample* bound presented in Theorem 2 has proved very useful in theoretical studies. It should also be stressed that the bound is *distribution free*, namely it holds independently of the underlying probability measure P. Moreover, it can be shown ([50], Theorem 14.5) to be optimal in rate in a precise minimax sense. However, in practical applications it is often far too conservative to yield useful results.

3.4 Margin Based Generalization Bounds

In spite of the claimed optimality of the above bound, there is good reason to investigate bounds which improve upon it. While such an endeavor may seem futile in light of the purported optimality of (15), we observe that the bound is optimal only in a distribution-free setting, where no restrictions are placed on the probability distribution P. In fact, one may want to take advantage of certain regularity properties of the data in order to improve the bound. However, in order to retain the appealing distribution-free feature of the bound (15), we do not want to impose any a-priori conditions on P. Rather, the idea is to construct a so-called *luckiness* function based on the data [179], which yields improved bounds in situations where the structure of the data happens to be 'simple'. In order to make more effective use of a classifier, we now allow the class of classifiers F to be *real-valued*. In view of the discussion in Section 3.1, a good candidate for a luckiness function is the margin-based loss $\hat{L}^{\theta}(f)$ defined in (10). The probability of error is given by $L(f) = P(y \neq \mathrm{sign}(f(\boldsymbol{x})))$.

For real-valued classifiers F, define a new notion of complexity, related to the VC dimension but somewhat more general. Let $\{\sigma_1, \ldots, \sigma_N\}$ be a sequence of $\{-1, +1\}$-valued random variables generated independently by setting each σ_n to $\{-1, +1\}$ with equal probability. Additionally, let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be generated independently at random according to some law P. The Rademacher complexity [189] of the class F is given by

$$R_N(\mathsf{F}) = \mathbf{E} \sup_{f \in \mathsf{F}} \left| \frac{1}{N} \sum_{n=1}^{N} \sigma_n f(\boldsymbol{x}_n) \right|,$$

where the expectation is taken with respect to both $\{\sigma_n\}$ and $\{\boldsymbol{x}_n\}$. The Rademacher complexity has proven to be essential in the derivation of effective generalization bounds (e.g. [189, 11, 108, 10]).

The basic intuition behind the definition of $R_N(\mathsf{F})$ is its interpretation as a measure of correlation between the class F and a random set of labels. For very rich function classes F we expect a large value of $R_N(\mathsf{F})$ while small function classes can only achieve small correlations with random labels. In the special case that the class F consists of binary functions, one can show that $R_N(\mathsf{F}) = \mathcal{O}(\sqrt{\mathrm{VCdim}(\mathsf{F})/N})$. For real-valued functions, one needs to extend the

notion of VC dimension to the so-called pseudo-dimension (e.g. [4]), in which case one can show that $R_N(\mathsf{F}) = \mathcal{O}(\sqrt{\mathrm{Pdim}(\mathsf{F})/N})$. It is not hard to show that $\mathrm{VCdim}(\mathrm{sign}(\mathsf{F})) \leq \mathrm{Pdim}(\mathsf{F})$.

The following theorem (cf. [108]) provides a bound on the probability of misclassification using a margin-based loss function.

Theorem 3 ([108]). *Let F be a class of real-valued functions from $\mathbb{X}$ to $[-1,+1]$, and let $\theta \in [0,1]$. Let P be a probability distribution on $\mathbb{X} \times \{-1,+1\}$, and suppose that N-samples $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ are generated independently at random according to P. Then, for any integer N, with probability at least $1-\delta$ over samples of length N, every $f \in \mathsf{F}$ satisfies*

$$L(f) \leq \hat{L}^{\theta}(f) + \frac{4R_N(\mathsf{F})}{\theta} + \sqrt{\frac{\log(2/\delta)}{2N}}. \tag{16}$$

In order to understand the significance of Theorem 3, consider a situation where one can find a function f which achieves a low margin error $\hat{L}^{\theta}(f)$ for a large value of the margin parameter θ. In other words, $\hat{L}^{\theta}(f) \approx \hat{L}(f)$ for large values of θ. In this case the second term on the r.h.s. in (16) can be made to be smaller than the corresponding term in (15) (recall that $R_N(\mathsf{F}) = \mathcal{O}(\sqrt{\mathrm{Pdim}(\mathsf{F})/N})$ for classes with finite pseudo-dimension), using the standard VC bounds. Note that Theorem 3 has implications concerning the size of the margin θ. In particular, assuming F is a class with finite VC dimension, we have that the second term on the r.h.s. of (16) is of the order $\mathcal{O}(\sqrt{\mathrm{VCdim}(\mathsf{F})/N\theta^2})$. In order that this term decrease to zero it is mandatory that $\theta \gg 1/\sqrt{N}$. In other words, in order to lead to useful generalization bounds the margin must be sufficiently large with respect to the sample size N.

However, the main significance of (16) arises from its application to the specific class of functions arising in Boosting algorithms. Recall that in Boosting, one generates a hypothesis f which can be written as $f_T(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$, namely a linear combination with non-negative coefficients of base hypotheses $h_1, \ldots, h_t$, each belonging to the class H. Consider the class of functions

$$\mathrm{co}_T(\mathsf{F}) = \left\{ f : f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \ : \ \alpha_t \geq 0, \quad \sum_{t=1}^{T} \alpha_t = 1, \quad h_t \in \mathsf{H} \right\},$$

corresponding to the ℓ_1-normalized function output by the Boosting algorithm. The convex hull is given by letting $T \to \infty$ in co_T.

The key feature in the application of Theorem 3 to Boosting is the observation that for any class of real-valued functions H, $R_N(\mathrm{co}_T(\mathsf{H})) = R_N(\mathsf{H})$ for any T (e.g. [11]), namely the Rademacher complexity of $\mathrm{co}_T(\mathsf{H})$ is not larger than that of the base class H itself. On the other hand, since $h_t(\boldsymbol{x})$ are in general non-linear functions, the linear combination $\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$ represents a potentially highly complex function which can lead to very low margin error $\hat{L}^{\theta}(f_T)$. Combining these observations, we obtain the following result, which improves upon the first bounds of this nature presented in [167].

Corollary 1. *Let the conditions of Theorem 3 hold, and set* $\mathsf{F} = \mathrm{co}_T(\mathsf{H})$. *Then, for any integer* N, *with probability at least* $1-\delta$ *over samples of length* N, *every* $f \in \mathrm{co}_T(\mathsf{H})$ *satisfies*

$$L(f) \leq \hat{L}^{\theta}(f) + \frac{4R_N(\mathsf{H})}{\theta} + \sqrt{\frac{\log(2/\delta)}{2N}}. \tag{17}$$

Had we used a bound of the form of (15) for $f \in \mathrm{co}_T(\mathsf{H})$, we would have obtained a bound depending on $\mathrm{Pdim}(\mathrm{co}_T(\mathsf{H}))$, which often grows linearly in T, leading to inferior results. The important observation here is that the complexity penalty in (17) is independent of T, the number of Boosting iterations.

As a final comment we add that a considerable amount of recent work has been devoted to the derivation of so-called data-dependent bounds (e.g. [5, 9, 92]), where the second term on the r.h.s. of (16) is made to depend explicitly on the data. Data-dependent bounds depending explicitly on the weights α_t of the weak learners are given in [130]. In addition, bounds which take into account the full margin distribution are presented in [180, 45, 181]. Such results are particularly useful for the purpose of model selection, but are beyond the scope of this review.

3.5 Consistency

The bounds presented in Theorems 2 and 3, depend explicitly on the data, and are therefore potentially useful for the purpose of model selection. However, an interesting question regarding the statistical consistency of these procedures arises. Consider a generic binary classification problem, characterized by the class conditional distribution function $P(y|\boldsymbol{x})$, where $\tau(\boldsymbol{x}) = P(y=1|\boldsymbol{x})$ is assumed to belong to some target class of functions $\mathcal{T}$. It is well known [50] that in this case the optimal classifier is given by the Bayes classifier $f_B(\boldsymbol{x}) = \mathrm{sign}(\tau(\boldsymbol{x}) - \frac{1}{2})$, leading to the minimal error $L_B = L(f_B)$. We say that an algorithm is *strongly consistent* with respect to $\mathcal{T}$ if, based on a sample of size N, it generates an empirical classifier $\hat{f}_N$ for which $L(\hat{f}_N) \to L_B$ almost surely for $N \to \infty$, for every $\tau \in \mathcal{T}$. While consistency may seem to be mainly of theoretical significance, it is reassuring to have the guarantee that a given procedure ultimately performs optimally. However, it turns out that in many cases inconsistent procedures perform better for *finite* amounts of data, than consistent ones. A classic example of this is the so-called James-Stein estimator ([96, 160], Section 2.4.5).

In order to establish consistency one needs to assume (or prove in specific cases) that as $T \to \infty$ the class of functions $\mathrm{co}_T(\mathsf{H})$ is dense in $\mathcal{T}$. The consistency of Boosting algorithms has recently been established in [116, 123], following related previous work [97]. The work of [123] also includes rates of convergence for specific weak learners and target classes $\mathcal{T}$. We point out that the full proof of consistency must tackle at least three issues. First, it must be shown that the specific algorithm used converges as a function of the number of iterations - this is essentially an issue of optimization. Furthermore, one must show that the function to which the algorithm converges itself converges to the optimal

estimator as the sample size increases - this is a statistical issue. Finally, the approximation theoretic issue of whether the class of weak learners is sufficiently powerful to represent the underlying decision boundary must also be addressed.

4 Boosting and Large Margins

In this section we discuss AdaBoost in the context of large margin algorithms. In particular, we try to shed light on the question of whether, and under what conditions, boosting yields large margins. In [67] it was shown that AdaBoost quickly finds a combined hypothesis that is consistent with the training data. [167] and [27] indicated that AdaBoost computes hypotheses with large margins, if one continues iterating after reaching zero classification error. It is clear that the margin should be as large as possible on most training examples in order to minimize the complexity term in (17). If one assumes that the base learner always achieves a weighted training error $\epsilon_t \leq 1/2 - \gamma/2$ with $\gamma > 0$, then AdaBoost generates a hypothesis with margin larger than $\gamma/2$ [167, 27]. However, from the Min-Max theorem of linear programming [193] (see Theorem 4 below) one finds that the achievable margin ρ^* is at least γ [69, 27, 72].

We start with a brief review of some standard definitions and results for the margin of an example and of a hyperplane. Then we analyze the asymptotic properties of a slightly more general version, called AdaBoost$_\varrho$, which is equivalent to AdaBoost for $\varrho = 0$, while assuming that the problem is separable. We show that there will be a subset of examples – the *support patterns* [153] – asymptotically having the same smallest margin. All weights $\mathbf{d}$ are asymptotically concentrated on these examples. Furthermore, we find that AdaBoost$_\varrho$ is able to achieve larger margins, if ϱ is chosen appropriately. We briefly discuss two algorithms for adaptively choosing ϱ to maximize the margin.

4.1 Weak Learning, Edges, and Margins

The assumption made concerning the base learning algorithm in the PAC-Boosting setting (cf. Section 3.1) is that it returns a hypothesis h from a fixed set H that is slightly better than the baseline classifier introduced in Definition 1. This means that for any distribution, the error rate ε is consistently smaller than $\frac{1}{2} - \frac{1}{2}\gamma$ for some fixed $\gamma > 0$.

Recall that the error rate ε of a base hypothesis is defined as the weighted fraction of points that are misclassified (cf. (8)). The weighting $\mathbf{d} = (d_1, \ldots, d_N)$ of the examples is such that $d_n \geq 0$ and $\sum_{n=1}^N d_n = 1$. A more convenient quantity to measure the quality of the hypothesis h is the *edge* [27], which is also applicable and useful for real-valued hypotheses:

$$\gamma(h, \mathbf{d}) = \sum_{n=1}^{N} d_n y_n h(\boldsymbol{x}_n). \tag{18}$$

The edge is an affine transformation of $\varepsilon(h, \mathbf{d})$ in the case when $h(\boldsymbol{x}) \in \{-1, +1\}$: $\varepsilon(h, \mathbf{d}) = \frac{1}{2} - \frac{1}{2}\gamma(h, \mathbf{d})$ [68, 27, 168]. Recall from Section 3.2 that the margin of a function f for a given example $(\boldsymbol{x}_n, y_n)$ is defined by $\rho_n(f) = y_n f(\boldsymbol{x}_n)$ (cf. (9)).

Assume for simplicity that H is a finite hypothesis class $\mathsf{H} = \{\tilde{h}_j \mid j = 1, \ldots, J\}$, and suppose we combine all possible hypotheses from H. Then the following well-known theorem establishes the connection between margins and edges (first noted in connection with Boosting in [68, 27]). Its proof follows directly from the duality of the two optimization problems.

Theorem 4 (Min-Max-Theorem, [193]).

$$\gamma^* = \min_{\mathbf{d}} \max_{\tilde{h} \in \mathsf{H}} \left\{ \sum_{n=1}^{N} d_n y_n \tilde{h}(\boldsymbol{x}_n) \right\} = \max_{\boldsymbol{w}} \min_{1 \le n \le N} y_n \left\{ \sum_{j=1}^{J} w_j \tilde{h}_j(\boldsymbol{x}_n) \right\} = \varrho^*, \tag{19}$$

where $\mathbf{d} \in \mathcal{P}^N$, $\boldsymbol{w} \in \mathcal{P}^J$ *and* $\mathcal{P}^k$ *is the* k-dimensional probability simplex.

Thus, the minimal edge γ^* that can be achieved over all possible weightings $\mathbf{d}$ of the training set is equal to the maximal margin ϱ^* of a combined hypothesis from H. We refer to the left hand side of (19) as the *edge minimization problem*. This problem can be rewritten as a linear program (LP):

$$\begin{aligned} \min_{\gamma, \mathbf{0} \le \mathbf{d} \in \mathbb{R}^N} \quad & \gamma \\ \text{s.t.} \quad & \sum_{n=1}^{N} d_n = 1 \text{ and } \sum_{n=1}^{N} d_n y_n \tilde{h}(\boldsymbol{x}_n) \le \gamma \quad \forall \tilde{h} \in \mathsf{H}. \end{aligned} \tag{20}$$

For any non-optimal weightings $\mathbf{d}$ and $\boldsymbol{w}$ we always have $\max_{\tilde{h} \in \mathsf{H}} \gamma(\tilde{h}, \mathbf{d}) \ge \gamma^* = \varrho^* \ge \min_{n=1,\ldots,N} y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n)$, where

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=1}^{J} w_j \tilde{h}_j(\boldsymbol{x}). \tag{21}$$

If the base learning algorithm is guaranteed to return a hypothesis with edge at least γ for any weighting, there exists a combined hypothesis with margin at least γ. If $\gamma = \gamma^*$, i.e. the lower bound γ is as large as possible, then there exists a combined hypothesis with margin exactly $\gamma = \varrho^*$ (only using hypotheses that are actually returned by the base learner). From this discussion we can derive a sufficient condition on the base learning algorithm to reach the maximal margin: If it returns hypotheses whose edges are at least γ^*, there exists a linear combination of these hypotheses that has margin $\gamma^* = \varrho^*$. This explains the termination condition in Algorithm 2.1 (step (4)).

Remark 1. Theorem 4 was stated for finite hypothesis sets H. However, the same result holds also for countable hypothesis classes. For uncountable classes, one can establish the same results under some regularity conditions on H, in particular that the real-valued hypotheses h are uniformly bounded (cf. [93, 149, 157, 147]).

To avoid confusion, note that the hypotheses indexed as elements of the hypothesis set H are marked by a tilde, i.e. $\tilde{h}_1, \ldots, \tilde{h}_J$, whereas the hypotheses returned by the base learner are denoted by $h_1, \ldots, h_T$. The output of AdaBoost and similar algorithms is a sequence of pairs (α_t, h_t) and a combined hypothesis $f_t(\boldsymbol{x}) = \sum_{r=1}^{t} \alpha_r h_r(\boldsymbol{x})$. *But how do the α's relate to the w's used in Theorem 4?* At every step of AdaBoost, one can compute the weight for each hypothesis $\tilde{h}_j$ in the following way:[3]

$$w_j^t = \sum_{r=1}^{t} \alpha_r \mathbf{I}(h_r = \tilde{h}_j), \quad j = 1, \ldots, J. \tag{22}$$

It is easy to verify that $\sum_{r=1}^{t} \alpha_r h_r(\boldsymbol{x}) = \sum_{j=1}^{J} w_j^t \tilde{h}_j(\boldsymbol{x})$. Also note, if the α's are positive (as in Algorithm 2.1), then $\|\boldsymbol{w}\|_1 = \|\boldsymbol{\alpha}\|_1$ holds.

4.2 Geometric Interpretation of p-Norm Margins

Margins have been used frequently in the context of Support Vector Machines (SVMs) [22, 41, 190] and Boosting. These so-called *large margin algorithms* focus on generating hyperplanes/functions with large margins on most training examples. Let us therefore study some properties of the maximum margin hyperplane and discuss some consequences of using different norms for measuring the margin (see also Section 6.3).

Suppose we are given N examples in some space $\mathbb{F}$: $S = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, where $(\mathbf{x}_n, y_n) \subseteq \mathbb{F} \times \{-1, 1\}$. Note that we use $\mathbf{x}$ as elements of the *feature space* $\mathbb{F}$ instead of $\boldsymbol{x}$ as elements of the *input space* $\mathbb{X}$ (details below). We are interested in the separation of the training set using a hyperplane A through the origin[4] $A = \{\mathbf{x} \mid \langle \mathbf{x}, \boldsymbol{w} \rangle = 0\}$ in $\mathbb{F}$ determined by some vector $\boldsymbol{w}$, which is assumed to be normalized with respect to some norm. The *ℓ_p-norm margin* of an example $(\boldsymbol{x}_n, y_n)$ with respect to the hyperplane A is defined as

$$\rho_n^p(\boldsymbol{w}) := \frac{y_n \langle \boldsymbol{x}_n, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|_p},$$

where the superscript $p \in [1, \infty]$ specifies the norm with respect to which $\boldsymbol{w}$ is normalized (the default is $p = 1$). A positive margin corresponds to a correct classification. The *margin of the hyperplane A* is defined as the minimum margin over all N examples, $\rho^p(\boldsymbol{w}) = \min_n \rho_n^p(\boldsymbol{w})$.

To maximize the margin of the hyperplane one has to solve the following convex optimization problem [119, 22]:

$$\max_{\boldsymbol{w}} \ \rho^p(\boldsymbol{w}) = \max_{\boldsymbol{w}} \min_{1 \le n \le N} \ \frac{y_n \langle \boldsymbol{x}_n, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|_p}. \tag{23}$$

[3] For simplicity, we have omitted the normalization implicit in Theorem 4.

[4] This can easily be generalized to general hyperplanes by introducing a bias term.

The form of (23) implies that without loss of generality we may take $\|\boldsymbol{w}\|_p = 1$, leading to the following convex optimization problem:

$$\begin{array}{ll} \max\limits_{\rho,\boldsymbol{w}} & \rho \\ \text{s.t.} & y_n \langle \boldsymbol{x}_n, \boldsymbol{w} \rangle \geq \rho, \quad n = 1, 2, \ldots, N \\ & \|\boldsymbol{w}\|_p = 1. \end{array} \tag{24}$$

Observe that in the case where $p = 1$ and $w_j \geq 0$, we obtain a *Linear Program* (LP) (cf. (19)). Moreover, from this formulation it is clear that only a few of the constraints in (24) will typically be active. These constraints correspond to the most difficult examples, called the *support patterns* in boosting and *support vectors* in SVMs.[5]

The following theorem gives a geometric interpretation to (23): Using the ℓ_p-norm to normalize $\boldsymbol{w}$ corresponds to measuring the distance to the hyperplane with the dual ℓ_q-norm, where $1/p + 1/q = 1$.

Theorem 5 ([120]). *Let* $\mathbf{x} \in \mathbb{F}$ *be any point which is not on the plane* $A := \{\tilde{\mathbf{x}} \mid \langle \tilde{\mathbf{x}}, \boldsymbol{w} \rangle = 0\}$. *Then for* $p \in [1, \infty]$*:*

$$\frac{|\langle \mathbf{x}, \boldsymbol{w} \rangle|}{\|\boldsymbol{w}\|_p} = \|\mathbf{x} - A\|_q, \tag{25}$$

where $\|\mathbf{x} - A\|_q = \min_{\tilde{\mathbf{x}} \in A} \|\mathbf{x} - \tilde{\mathbf{x}}\|_q$ *denotes the distance of* $\mathbf{x}$ *to the plane* A *measured with the dual norm* ℓ_q.

Thus, the ℓ_p-margin of $\mathbf{x}_n$ is the *signed* ℓ_q*-distance* of the point to the hyperplane. If the point is on the correct side of the hyperplane, the margin is positive (see Figure 3 for an illustration of Theorem 5).

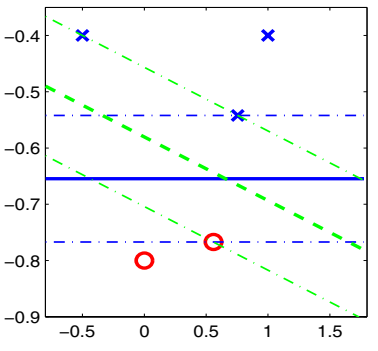

Fig. 3. The maximum margin solution for different norms on a toy example: ℓ_∞-norm (solid) and ℓ_2-norm (dashed). The margin areas are indicated by the dash-dotted lines. The examples with label $+1$ and -1 are shown as 'x' and '$\circ$', respectively. To maximize the ℓ_∞-norm and ℓ_2-norm margin, we solved (23) with $p = 1$ and $p = 2$, respectively. The ℓ_∞-norm often leads to fewer supporting examples and sparse weight-vectors.

Let us discuss two special cases: $p = 1$ and $p = 2$:

[5] In Section 4.4 we will discuss the relation of the Lagrange multipliers of these constraints and the weights $\mathbf{d}$ on the examples.

Boosting ($p = 1$). In the case of Boosting, the space $\mathbb{F}$ is spanned by the base hypotheses. One considers the set of base hypotheses that could be generated by the base learner – the base hypothesis set – and constructs a mapping Φ from the input space $\mathbb{X}$ to the "feature space" $\mathbb{F}$:

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} \tilde{h}_1(\boldsymbol{x}) \\ \tilde{h}_2(\boldsymbol{x}) \\ \vdots \end{bmatrix} : \mathbb{X} \to \mathbb{F}. \tag{26}$$

In this case the margin of an example is (cf. (22))

$$\rho_n^1(\boldsymbol{w}) = \frac{y_n \langle \mathbf{x}_n, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|_1} = \frac{y_n \sum_j w_j \tilde{h}_j(\boldsymbol{x}_n)}{\sum_{j=1}^J w_j} = \frac{y_n \sum_t \alpha_t h_t(\boldsymbol{x}_n)}{\sum_t \alpha_t},$$

as used before in Algorithm 2.1, where we used the ℓ_1-norm for normalization and without loss of generality assumed that the w's and α's are non-negative (assuming H is closed under negation). By Theorem 5, one therefore maximizes the ℓ_∞-norm distance of the mapped examples to the hyperplane in the feature space $\mathbb{F}$. Under mild assumptions, one can show that the maximum margin hyperplane is aligned with most coordinate axes in the feature space, i.e. many w_j's are zero (cf. Figure 3 and Section 6.3).

Support Vector Machines ($p = 2$). Here, the feature space $\mathbb{F}$ is implicitly defined by a Mercer kernel $\mathrm{k}(\boldsymbol{x}, \boldsymbol{y})$ [131], which computes the inner product of two examples in the feature space $\mathbb{F}$ [22, 175]. One can show that for every such kernel there exists a mapping $\Phi : \mathbb{X} \to \mathbb{F}$, such that $\mathrm{k}(\boldsymbol{x}, \boldsymbol{y}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{X}$. Additionally, one uses the ℓ_2-norm for normalization and, hence, the Euclidean distance to measure distances between the mapped examples and the hyperplane in the feature space $\mathbb{F}$:

$$\rho_n^2(\boldsymbol{w}) = \frac{y_n \langle \Phi(\boldsymbol{x}_n), \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|_2} = \frac{y_n \sum_{i=1}^N \beta_i \, \mathrm{k}(\boldsymbol{x}_n, \boldsymbol{x}_i)}{\sqrt{\sum_{i,j=1}^N \beta_i \beta_j \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)}},$$

where we used the Representer Theorem [103, 172] that shows that the maximum margin solution $\boldsymbol{w}$ can be written as a sum of the mapped examples, i.e. $\boldsymbol{w} = \sum_{n=1}^N \beta_n \Phi(\boldsymbol{x}_n)$.

4.3 AdaBoost and Large Margins

We have seen in Section 3.4 that a large value of the margin is conducive to good generalization, in the sense that if a large margin can be achieved with respect to the data, then an upper bound on the generalization error is small (see also discussion in Section 6). This observation motivates searching for algorithms which maximize the margin.

Convergence properties of AdaBoost. We analyze a slightly generalized version of AdaBoost. One introduces a new parameter, ϱ, in step (3c) of Algorithm 2.1 and chooses the weight of the new hypothesis differently:

$$\alpha_t = \frac{1}{2}\log\frac{1+\gamma_t}{1-\gamma_t} - \frac{1}{2}\log\frac{1+\varrho}{1-\varrho}.$$

This algorithm was first introduced in [27] as *unnormalized Arcing with exponential function* and in [153] as *AdaBoost-type algorithm*. Moreover, it is similar to an algorithm proposed in [71] (see also [177]). Here we will call it AdaBoost$_\varrho$. Note that the original AdaBoost algorithm corresponds to the choice $\varrho = 0$.

Let us for the moment assume that we chose ϱ at each iteration differently, i.e. we consider sequences $\{\varrho_t\}_{t=1}^T$, which might either be specified before running the algorithm or computed based on results obtained during the running of the algorithm. In the following we address the issue of how well this algorithm, denoted by AdaBoost$_{\{\varrho_t\}}$, is able to increase the margin, and bound the fraction of examples with margin smaller than some value θ.

We start with a result generalizing Theorem 1 (cf. Section 3.2) to the case $\varrho \neq 0$ and a slightly different loss:

Proposition 1 ([153]). *Let γ_t be the edge of h_t at the t-th step of AdaBoost$_\varrho$. Assume $-1 \leq \varrho_t \leq \gamma_t$ for $t = 1, \ldots, T$. Then for all $\theta \in [-1, 1]$*

$$\hat{L}^\theta(f_T) \leq \frac{1}{N}\sum_{n=1}^N \mathbf{I}(y_n f_T(\boldsymbol{x}_n) \leq \theta) \leq \prod_{t=1}^T \left(\frac{1-\gamma_t}{1-\varrho_t}\right)^{\frac{1-\theta}{2}} \left(\frac{1+\gamma_t}{1+\varrho_t}\right)^{\frac{1+\theta}{2}} \tag{27}$$

The algorithm makes progress, if each of the products on the right hand side of (27) is smaller than one.

Suppose we would like to reach a margin θ on all training examples, where we obviously need to assume $\theta \leq \varrho^*$ (here ϱ^* is defined in (19)). The question arises as to which sequence of $\{\varrho_t\}_{t=1}^T$ one should use to find such a combined hypothesis in as few iterations as possible (according to (27)). One can show that the right hand side of (27) is minimized for $\varrho_t = \theta$ and, hence, one should always use this choice, independent of how the base learner performs.

Using Proposition 1, we can determine an upper bound on the number of iterations needed by AdaBoost$_\varrho$ for achieving a margin of ϱ on all examples, given that the maximum margin is ϱ^* (cf. [167, 157]):

Corollary 2. *Assume the base learner always achieves an edge $\gamma_t \geq \varrho^*$. If $0 \leq \varrho \leq \varrho^* - \nu$, $\nu > 0$, then AdaBoost$_\varrho$ will converge to a solution with margin of at least ϱ on all examples in at most $\lceil \frac{2\log(N)(1-\varrho^2)}{\nu^2} \rceil + 1$ steps.*

Maximal Margins. Using the methodology reviewed so far, we can also analyze to what value the maximum margin of the original AdaBoost algorithm converges *asymptotically*. First, we state a lower bound on the margin that is achieved by AdaBoost$_\varrho$. We find that the size of the margin is not as large as it could be theoretically based on Theorem 4. We briefly discuss below two algorithms that are able to maximize the margin.

As long as each factor on the r.h.s. of (27) is smaller than 1, the bound decreases. If it is bounded away from 1, then it converges exponentially fast to zero. The following corollary considers the asymptotic case and provides a lower bound on the margin when running AdaBoost_ϱ forever.

Corollary 3 ([153]). *Assume AdaBoost_ϱ generates hypotheses $h_1, h_2, \ldots$ with edges $\gamma_1, \gamma_2, \ldots$. Let $\gamma = \inf_{t=1,2,\ldots} \gamma_t$ and assume $\gamma > \varrho$. Then the smallest margin ρ of the combined hypothesis is asymptotically ($t \to \infty$) bounded from below by*

$$\rho \geq \frac{\log(1-\varrho^2) - \log(1-\gamma^2)}{\log\left(\frac{1+\gamma}{1-\gamma}\right) - \log\left(\frac{1+\varrho}{1-\varrho}\right)} \geq \frac{\gamma+\varrho}{2}. \tag{28}$$

From (28) one can understand the interaction between ϱ and γ: if the difference between γ and ϱ is small, then the middle term of (28) is small. Thus, if ϱ is large (assuming $\varrho \leq \gamma$), then ρ must be large, i.e. choosing a larger ϱ results in a larger margin on the training examples.

Remark 2. Under the conditions of Corollary 3, one can compute a lower bound on the hypothesis coefficients in each iteration. Hence the sum of the hypothesis coefficients will increase to infinity at least linearly. It can be shown that this suffices to guarantee that the combined hypothesis has a large margin, i.e. larger than ϱ (cf. [147], Section 2.3).

However, in Section 4.1 we have shown that the maximal achievable margin is at least γ. Thus if ϱ is chosen to be too small, then we guarantee only a *suboptimal asymptotic margin*. In the original formulation of AdaBoost we have $\varrho = 0$ and we guarantee only that AdaBoost_0 achieves a margin of at least $\gamma/2$.[6] This gap in the theory led to the so-called *Arc-GV* algorithm [27] and the *Marginal AdaBoost* algorithm [157].

Arc-GV. The idea of of Arc-GV [27] is to set ϱ to the margin that is currently achieved by the combined hypothesis, i.e. depending on the previous performance of the base learner:[7]

$$\varrho_t = \max\left(\varrho_{t-1}, \min_{n=1,\ldots,N} y_n f_{t-1}(\boldsymbol{x}_n)\right). \tag{29}$$

There is a very simple proof using Corollary 3 that Arc-GV asymptotically maximizes the margin [147, 27]. The idea is to show that ϱ_t converges to ρ^* – the maximum margin. The proof is by contradiction: Since $\{\varrho_t\}$ is monotonically increasing on a bounded interval, it must converge. Suppose $\{\varrho_t\}$ converges to a value ϱ^* smaller than ρ^*, then one can apply Corollary 3 to show that the margin of the combined hypothesis is asymptotically larger than $\frac{\rho^*+\varrho^*}{2}$. This leads to a contradiction, since ϱ_t is chosen to be the margin of the previous iteration. This shows that Arc-GV asymptotically maximizes the margin.

[6] Experimental results in [157] confirm this analysis and illustrate that the bound given in Corollary 3 is tight.

[7] The definition of Arc-GV is slightly modified. The original algorithm did not use the max.

Marginal AdaBoost. Unfortunately, it is not known how fast Arc-GV converges to the maximum margin solution. This problem is solved by Marginal AdaBoost [157]. Here one also adapts ϱ, but in a different manner. One runs AdaBoost$_\varrho$ repeatedly, and determines ϱ by a line search procedure: If AdaBoost$_\varrho$ is able to achieve a margin of ϱ then it is increased, otherwise decreased. For this algorithm one can show fast convergence to the maximum margin solution [157].

4.4 Relation to Barrier Optimization

We would like to point out how the discussed algorithms can be seen in the context of *barrier optimization*. This illustrates how, from an optimization point of view, Boosting algorithms are related to linear programming, and provide further insight as to why they generate combined hypotheses with large margins.

The idea of barrier techniques is to solve a sequence of unconstrained optimization problems in order to solve a constrained optimization problem (e.g. [77, 136, 40, 156, 147]). We show that the exponential function acts as a *barrier function* for the constraints in the maximum margin LP (obtained from (24) by setting $p = 1$ and restricting $w_j \geq 0$):

$$\begin{array}{ll} \max\limits_{\rho,\boldsymbol{w}} & \rho \\ \text{s.t.} & y_n \langle \boldsymbol{x}_n, \boldsymbol{w} \rangle \geq \rho \qquad n = 1, 2, \ldots, N \\ & w_j \geq 0, \ \sum w_j = 1. \end{array} \tag{30}$$

Following the standard methodology and using the *exponential barrier function*[8] $\beta \exp(-z/\beta)$ [40], we find the *barrier objective* for problem (30):

$$\mathrm{F}_\beta(\boldsymbol{w}, \rho) = -\rho + \beta \sum_{n=1}^{N} \exp\left[\frac{1}{\beta}\left(\rho - \frac{y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n)}{\sum_j w_j}\right)\right], \tag{31}$$

which is minimized with respect to ρ and $\boldsymbol{w}$ for some fixed β. We denote the optimum by $\boldsymbol{w}_\beta$ and ρ_β. One can show that any limit point of $(\boldsymbol{w}_\beta, \rho_\beta)$ for $\beta \to 0$ is a solution of (30) [136, 19, 40]. This also holds when one only has a sequence of approximate minimizers and the approximation becomes better for decreasing β [40]. Additionally, the quantities $\tilde{d}_n = \exp(\rho_\beta \sum_j w_j - y_n f_{\boldsymbol{w}_\beta}(\boldsymbol{x}_n))/Z$ (where Z is chosen such that $\sum_n \tilde{d}_n = 1$) converge for $\beta \to 0$ to the optimal Lagrange multipliers d_n^* of the constraints in (30) (under mild assumptions, see [40], Theorem 2, for details). Hence, they are a solution of the edge minimization problem (20).

To see the connection between Boosting and the barrier approach, we chose $\beta = \left(\sum_j w_j\right)^{-1}$. If the sum of the hypothesis coefficients increases to infinity (cf. Remark 2 in Section 4.3), then β converges to zero. Plugging in this choice of β into (31) one obtains: $-\rho + (\sum_j w_j)^{-1} \sum_{n=1}^{N} \exp\left[\rho \sum_j w_j - y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n)\right]$. Without going into further details [147, 150], one can show that this function

[8] Other barrier functions are $\beta \log(z)$ (log-barrier) and $\beta z \log(z)$ (entropic barrier).

is minimized by the AdaBoost$_\varrho$ algorithm. Thus, one essentially obtains the exponential loss function as used in AdaBoost$_\varrho$ for $\rho = \varrho$.

Seen in this context, AdaBoost$_\varrho$ is an algorithm that finds a *feasible solution*, i.e. one that has margin at least ϱ. Also, it becomes clear why it is important that the sum of the hypothesis coefficients goes to infinity, an observation already made in the previous analysis (cf. Remark 2): otherwise β would not converge to 0 and the constraints may not be enforced. Arc-GV and Marginal AdaBoost are extensions, where the parameter ρ is optimized as well. Hence, AdaBoost, AdaBoost$_\varrho$, Arc-GV and Marginal AdaBoost can be understood as particular implementations of a barrier optimization approach, asymptotically solving a linear optimization problem.

5 Leveraging as Stagewise Greedy Optimization

In Section 4 we focused mainly on AdaBoost. We now extend our view to more general *ensemble learning methods* which we refer to as *leveraging algorithms* [56]. We will relate these methods to numerical optimization techniques. These techniques served as powerful tools to prove the convergence of leveraging algorithms (cf. Section 5.4).

We demonstrate the convergence of ensemble learning methods such as AdaBoost [70] and Logistic Regression (LR) [74, 39], although the techniques are much more general. These algorithms have in common that they iteratively call a base learning algorithm $\mathcal{L}$ (a.k.a. *weak learner*) on a weighted training sample. The base learner is expected to return at each iteration t a hypothesis h_t from the hypothesis set H that has small *weighted training error* (see (4)) or large edge (see (18)). These hypotheses are then linearly combined to form the final or *composite hypothesis* f_T as in (1). The hypothesis coefficient α_t is determined at iteration t, such that a certain objective is minimized or approximately minimized, and it is fixed for later iterations.

For AdaBoost and the Logistic Regression algorithm [74] it has been shown [39] that they generate a composite hypothesis minimizing a loss function G – in the limit as the number of iterations goes to infinity. The loss depends only on the output of the combined hypothesis f_T on the training sample. However, the assumed conditions (discussed later in detail) in [39] on the performance of the base learner are rather strict and can usually not be satisfied in practice. Although parts of the analysis in [39] hold for any strictly convex cost function of Legendre-type (cf. [162], p. 258), one needs to demonstrate the existence of a so-called *auxiliary function* (cf. [46, 39, 114, 106]) for each cost function other than the exponential or the logistic loss. This has been done for the general case in [147] under very mild assumptions on the base learning algorithm and the loss function.

We present a family of algorithms that are able to generate a combined hypothesis f converging to the minimum of some loss function G$[f]$ (if it exists). Special cases are AdaBoost [70], Logistic Regression and LS-Boost [76]. While assuming rather mild conditions on the base learning algorithm and the loss function G, *linear convergence rates* (e.g. [115]) of the type $\mathrm{G}[f_{t+1}] - \mathrm{G}[f^*] \leq$

$\eta(\mathrm{G}[f_t] - \mathrm{G}[f^*])$ for some fixed $\eta \in [0, 1)$ has been shown. This means that the deviation from the minimum loss converges exponentially to zero (in the number of iterations). Similar convergence rates have been proven for AdaBoost in the special case of *separable data* (cf. Section 4.3 and [70]). In the general case these rates can only be shown when the hypothesis set is finite. However, in practice one often uses infinite sets (e.g. hypotheses parameterized by some real valued parameters). Recently, Zhang [199] has shown *order one convergence* for such algorithms, i.e. $\mathrm{G}[f_{t+1}] - \mathrm{G}[f^*] \leq c/t$ for some fixed $c > 0$ depending on the loss function.

5.1 Preliminaries

In this section and the next one we show that AdaBoost, Logistic Regression and many other leveraging algorithms generate a combined hypothesis f minimizing a particular loss G on the training sample.

The composite hypothesis $f_{\boldsymbol{w}}$ is a linear combination of the base hypotheses:

$$f_{\boldsymbol{w}} \in \mathrm{lin}(\mathsf{H}) := \left\{ \sum_{\tilde{h} \in \mathsf{H}} w_{\tilde{h}} \tilde{h} \mid \tilde{h} \in \mathsf{H}, w_{\tilde{h}} \in \mathbb{R} \right\},$$

where the $\boldsymbol{w}$'s are the combination coefficients. Often one would like to find a combined function with small classification error, hence one would like to minimize the 0/1-loss:

$$\mathrm{G}^{0/1}(f, S) := \sum_{n=1}^{N} \mathbf{I}(y_n \neq \mathrm{sign}(f_{\boldsymbol{w}}(\boldsymbol{x}_n))).$$

However, since this loss is non-convex and not even differentiable, the problem of finding the best linear combination is a very hard problem. In fact, the problem is provably intractable [98]. One idea is to use another loss function which bounds the 0/1-loss from above. For instance, AdaBoost employs the exponential loss

$$\mathrm{G}^{AB}(f_{\boldsymbol{w}}, S) := \sum_{n=1}^{N} \exp(-y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n)),$$

and the LogitBoost algorithm [74] uses the Logistic loss:

$$\mathrm{G}^{LR}(f_{\boldsymbol{w}}, S) := \sum_{n=1}^{N} \log_2 \left(1 + \exp(-y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n))\right).$$

As can be seen in Figure 4, both loss functions bound the classification error $\mathrm{G}^{0/1}(f, S)$ from above. Other loss functions have been proposed in the literature, e.g. [74, 127, 154, 196].

It can be shown [116, 196, 28] that in the infinite sample limit, where the sample average converges to the expectation, minimizing either $\mathrm{G}^{AB}(f_{\boldsymbol{w}}, S)$ or

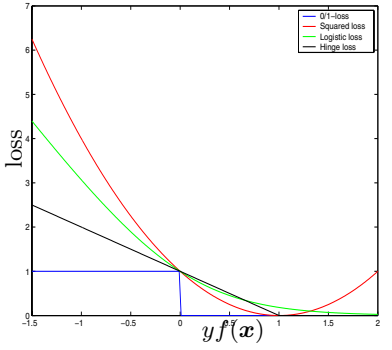

Fig. 4. The exponential (dashed) and logistic (solid) loss functions,plotted against $\rho = yf(\boldsymbol{x})$. Both bound the 0/1-loss from above.

$\mathrm{G}^{LR}(f_{\boldsymbol{w}}, S)$ leads to the same classifier as would be obtained by minimizing the probability of error. More precisely, let $\mathrm{G}^{AB}(f) = \mathbf{E}\{\exp(-yf(\boldsymbol{x}))\}$, and set $\mathrm{G}^{0/1}(f) = \mathbf{E}\{\mathbf{I}[y \neq \mathrm{sign}(f(\boldsymbol{x}))]\} = \Pr[y \neq \mathrm{sign}(f(\boldsymbol{x}))]$. Denote by f^* the function minimizing $\mathrm{G}^{AB}(f)$. Then it can be shown that f^* minimizes $\Pr[y \neq \mathrm{sign}(f(\boldsymbol{x}))]$, namely achieves the Bayes risk. A similar argument applies to other loss functions. This observation forms the basis for the consistency proofs in [116, 123] (cf. Section 3.5).

Note, that the exponential and logistic losses are both convex functions. Hence, the problem of finding the global optimum of the loss with respect to the combination coefficients can be solved efficiently. Other loss functions have been used to approach multi-class problems [70, 3, 164], ranking problems [95], unsupervised learning [150] and regression [76, 58, 149]. See Section 7 for more details on some of these approaches.

5.2 A Generic Algorithm

Most Boosting algorithms have in common that they iteratively run a learning algorithm to greedily select the next hypothesis $h \in \mathsf{H}$. In addition, they use in some way a weighting on the examples to influence the choice of the base hypothesis. Once the hypothesis is chosen, the algorithm determines the combination weight for the newly obtained hypothesis. Different algorithms use different schemes to weight the examples and to determine the new hypothesis weight.

In this section we discuss a generic method (cf. Algorithm 5.2) and the connection between the loss function minimized and the particular weighting schemes. Many of the proposed algorithms can be derived from this scheme. Let us assume the loss function $\mathrm{G}(f, S)$ has the following additive form[9]

$$\mathrm{G}(f, S) := \sum_{n=1}^{N} \mathrm{g}(f(\boldsymbol{x}_n), y_n), \tag{32}$$

and we would like to solve the optimization problem

[9] More general loss functions are possible and have been used, however, for the sake of simplicity, we chose this additive form.

Algorithm 5.2 – A Leveraging algorithm for the loss function G.

1. **Input:** $S = \langle (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \rangle$, No. of Iterations T
   Loss function $\mathrm{G} : \mathbb{R}^N \to \mathbb{R}$
2. **Initialize:** $f_0 \equiv 0$, $d_n^1 = \mathrm{g}'(f_0(\boldsymbol{x}_n), y_n)$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots, T$,
   a) Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $\mathsf{H} \in h_t : \mathbb{X} \to \mathbb{Y}$
   b) Set $\alpha_t = \mathrm{argmin}_{\alpha \in \mathbb{R}} \mathrm{G}[f_t + \alpha h_t]$
   c) Update $f_{t+1} = f_t + \alpha_t h_t$ and $d_n^{(t+1)} = \mathrm{g}'(f_{t+1}(\boldsymbol{x}_n), y_n)$, $n = 1, \ldots, N$
4. **Output:** f_T

$$\min_{f \in \mathrm{lin}(\mathsf{H})} \mathrm{G}(f, S) = \min_{\boldsymbol{w}} \left\{ \sum_{n=1}^{N} \mathrm{g}(f_{\boldsymbol{w}}(\boldsymbol{x}_n), y_n) \right\}. \tag{33}$$

In step (2) of Algorithm 5.2 the example weights are initialized using the derivative g' of the loss function g with respect to the first argument at $(f_0(\boldsymbol{x}_n), y_n)$ (i.e. at $(0, y_n)$).

At each step t of Algorithm 5.2 one can compute the weight w_j^t assigned to each base hypothesis $\tilde{h}_j$, such that (cf. (22))

$$f_{\boldsymbol{w}^t} := f_t = \sum_{\tilde{h} \in \mathsf{H}} w_{\tilde{h}}^t \tilde{h} = \sum_{r=1}^{t} \alpha_t h_t.$$

In iteration t of Algorithm 5.2 one chooses a hypothesis h_t, which corresponds to a weight (coordinate) w_{h_t}. This weight is then updated to minimize the loss (cf. step (3b)): $w_{h_t}^{(t+1)} = w_{h_t}^{(t)} + \alpha_t$. Since one always selects a single variable for optimization at each time step, such algorithms are called *coordinate descent methods*.

Observe that if one uses AdaBoost's exponential loss in Algorithm 5.2, then $d_n^{(t+1)} = \mathrm{g}'(f_t(\boldsymbol{x}_n), y_n) = y_n \exp(-y_n f_t(\boldsymbol{x}_n))$. Hence, in distinction from the original AdaBoost algorithm (cf. Algorithm 2.1), the d's are multiplied by the labels. This is the reason why we will later use a different definition of the edge (without the label).

Consider what would happen, if one always chooses the same hypothesis (or coordinate). In this case, one could not hope to prove convergence to the minimum. Hence, one has to assume the base learning algorithm performs well in selecting the base hypotheses from H. Let us first assume the base learner always find the hypothesis with maximal edge (or minimal classification error in the hard binary case):

$$h_t = \operatorname*{argmax}_{\tilde{h} \in \mathsf{H}} \left\{ \sum_{n=1}^{N} d_n^t \tilde{h}(\boldsymbol{x}_n) \right\}. \tag{34}$$

This is a rather restrictive assumption, being one among many that can be made (cf. [168, 199, 152]). We will later significantly relax this condition (cf. (40)).

Let us discuss why the choice in (34) is useful. For this we compute the gradient of the loss function with respect to the weight $w_{\tilde{h}}$ of each hypothesis $\tilde{h}$ in the hypothesis set:

$$\frac{\partial \mathrm{G}(f_{\boldsymbol{w}^t}, S)}{\partial w_{\tilde{h}}} = \sum_{n=1}^{N} \mathrm{g}'(f_t(\boldsymbol{x}_n), y_n)\tilde{h}(\boldsymbol{x}_n) = \sum_{n=1}^{N} d_n^t \tilde{h}(\boldsymbol{x}_n). \tag{35}$$

Hence, h_t is the coordinate that has the largest component in the direction of the gradient vector of G evaluated at $\boldsymbol{w}^t$.[10] If one optimizes this coordinate one would expect to achieve a large reduction in the value of the loss function.

This method of selecting the coordinate with largest absolute gradient component is called *Gauss-Southwell-method* in the field of numerical optimization (e.g. [115]). It is known to converge under mild assumptions [117] (see details in [152] and Section 5.4).

Performing the explicit calculation in (35) for the loss functions G^{AB} (see (5)) yields the AdaBoost approach described in detail in Algorithm 2.1. In this case $\mathrm{g}(f(\boldsymbol{x}), y) = \exp(-yf(\boldsymbol{x}))$ implying that $d_n^{(t)} \propto y_n \exp(-y_n f_{t-1}(\boldsymbol{x}_n))$, which, up to a normalization constant and the factor y_n is the form given in Algorithm 2.1.

The LogitBoost algorithm [74] mentioned in Section 2.2 can be derived in a similar fashion using the loss function G^{LR} (see (7)). In this case we find that $d_n^{(t)} \propto y_n/(1 + \exp(y_n f_{t-1}(\boldsymbol{x}_n)))$. Observe that in this case, the weights assigned to misclassified examples with very negative margins is much smaller than the exponential weights assigned by AdaBoost. This may help to explain why LogitBoost tends to be less sensitive to noise and outliers than AdaBoost [74]. Several extensions of the AdaBoost and LogitBoost algorithms based on Bregman distances were proposed in [39] following earlier work of [104] and [110] (cf. Section 5.3).

5.3 The Dual Formulation

AdaBoost was originally derived from results on online-learning algorithms [70], where one receives at each iteration one example, then predicts its label and incurs a loss. The important question in this setting relates to the speed at which the algorithm is able to learn to produce predictions with small loss. In [35, 106, 105] the total loss was bounded in terms of the loss of the best predictor. To derive these results, Bregman divergences [24] and *generalized projections* were extensively used. In the case of boosting, one takes a dual view [70]: Here the set of examples is fixed, whereas at each iteration the base learning algorithm generates a new hypothesis (the "online example"). Using online-learning techniques, the convergence in the online learning domain – the *dual domain* – has been analyzed and shown to lead to convergence results in the *primal domain* (cf. Section 4.3). In the primal domain the hypothesis coefficients $\boldsymbol{w}$ are optimized, while the weighting $\mathbf{d}$ are optimized in the dual domain.

[10] If one assumes the base hypothesis set to be closed under negation ($h \in \mathsf{H} \Rightarrow -h \in \mathsf{H}$), then this is equivalent to choosing the hypothesis with largest *absolute* component in the direction of the gradient.

In [104, 110] AdaBoost was interpreted as *entropy projection* in the dual domain. The key observation is that the weighting $\mathbf{d}^{(t+1)}$ on the examples in the t-th iteration is computed as a *generalized projection* of $\mathbf{d}^{(t)}$ onto a hyperplane (defined by linear constraints), where one uses a generalized distance $\Delta(\mathbf{d}, \tilde{\mathbf{d}})$ measure. For AdaBoost the *unnormalized relative entropy* is used, where

$$\Delta(\mathbf{d}, \tilde{\mathbf{d}}) = \sum_{n=1}^{N} d_n \log(d_n/\tilde{d}_n) - d_n + \tilde{d}_n. \tag{36}$$

The update of the distribution in steps (3c) and (3d) of Algorithm 2.1 is the solution to the following optimization problem ("generalized projection"):

$$\begin{aligned} &\mathbf{d}^{(t+1)} = \operatorname{argmin}_{\mathbf{d}\in\mathbb{R}_+^N} \Delta(\mathbf{d}, \mathbf{d}^{(t)}) \\ &\text{with} \quad \textstyle\sum_{n=1}^{N} d_n y_n \tilde{h}_t(\boldsymbol{x}_n) = 0, \end{aligned} \tag{37}$$

Hence, the new weighting $\mathbf{d}^{(t+1)}$ is chosen such that the edge of the previous hypothesis becomes zero (as e.g. observed in [70]) and the *relative entropy* between the new $\mathbf{d}^{(t+1)}$ and the old $\mathbf{d}^{(t)}$ distribution is as small as possible [104]. After the projection the new $\mathbf{d}$ lies on the hyperplane defined by the corresponding constraint (cf. Figure 5).

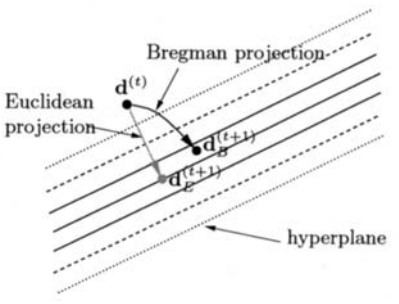

Fig. 5. Illustration of generalized projections: one projects a point $\mathbf{d}^{(t)}$ onto a plane $\tilde{\mathrm{H}}$ by finding the point $\mathbf{d}^{(t+1)}$ on the plane that has the smallest generalized distance from $\mathbf{d}^{(t)}$. If $\mathcal{G}(\mathbf{d}) = \frac{1}{2}\|\mathbf{d}\|_2^2$, then the generalized distance is equal to the squared Euclidean distance, hence, the projected point $\mathbf{d}_E^{(t+1)}$ is the closest (in the common sense) point on the hyperplane. For another Bregman function $\mathcal{G}$, one finds another projected point $\mathbf{d}_B^{(t+1)}$, since closeness is measured differently.

The work of [104, 110] and later of [39, 47, 111, 196] lead to a more general understanding of Boosting methods in the context of Bregman distance optimization and Information Theory. Given an arbitrary strictly convex function $\mathcal{G} : \mathbb{R}_+^N \to \mathbb{R}$ (of Legendre-type), called a *Bregman function*, one can define a *Bregman divergence* ("generalized distance"):

$$\Delta_{\mathcal{G}}(\boldsymbol{x}, \boldsymbol{y}) := \mathcal{G}(\boldsymbol{x}) - \mathcal{G}(\boldsymbol{y}) - \langle \nabla\mathcal{G}(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle. \tag{38}$$

AdaBoost, Logistic regression and many other algorithms can be understood in this framework as algorithms that iteratively project onto different hyper-

planes.[11] A precursor of such algorithms is the Bregman algorithm [24, 34] and it is known that the sequence $\{\mathbf{d}^{(t)}\}$ converges to a point on the intersection of all hyperplanes which minimizes the divergence to the first point $\mathbf{d}^{(0)}$ – in AdaBoost the uniform distribution. Hence they solve the following optimization problem:[12]

$$\begin{array}{l} \min_{\mathbf{d}\in\mathbb{R}^N_+} \Delta_{\mathcal{G}}(\mathbf{d}, \mathbf{d}^{(0)}) \\ \text{with} \quad \sum_{n=1}^N d_n y_n \tilde{h}(\boldsymbol{x}_n) = 0 \quad \forall \tilde{h} \in \mathsf{H}. \end{array} \tag{39}$$

An interesting questions is how generalized projections relate to coordinate descent discussed in Section 5.2. It has been shown [147] (see also [117, 107]) that a generalized projection in the dual domain onto a hyperplane (defined by h; cf. (37)) corresponds to the optimization of the variable corresponding to h in the primal domain. In particular, if one uses a Bregman function $\mathcal{G}(\mathbf{d})$, then this corresponds to a coordinate-wise descent in the loss function $\mathrm{G}(\sum_{j=1}^J w_j \tilde{h}_j(X) + (\nabla\mathcal{G})^{-1}(\mathbf{d}^{(0)}))$, where $\tilde{h}_j(X) = (\tilde{h}_j(\boldsymbol{x}_1), \ldots, \tilde{h}_j(\boldsymbol{x}_N))$ and $\mathrm{G} : \mathbb{R}^N \to \mathbb{R}$ is the convex conjugate of $\mathcal{G}$.[13] In the case of AdaBoost one uses $\mathcal{G}(\mathbf{d}) = \sum_n d_n \log d_n$, leading to the relative entropy in (36) and $(\nabla\mathcal{G})^{-1}(\mathbf{d}^{(0)})$ is zero, if $\mathbf{d}^{(0)}$ is uniform.

5.4 Convergence Results

There has recently been a lot of work on establishing the convergence of several algorithms, which are all very similar to Algorithm 5.2. Since we cannot discuss all approaches in detail, we only provide an overview of different results which have been obtained in the past few years (cf. Table 1). In the following few paragraphs we briefly discuss several aspects which are important for proving convergence of such algorithms.

Conditions on the Loss Function. In order to prove convergence to a *global minimum* one usually has to assume that the loss function is *convex* (then every local minimum is a global one). If one allows arbitrary loss functions, one can only show convergence to a local minimum (i.e. gradient converges to zero, e.g. [127]). Moreover, to prove convergence one usually assumes the function is reasonably *smooth*, where some measure of smoothness often appears in convergence rates. For some results it is important that the second derivatives are strictly positive, i.e. the loss function is *strictly convex* ("function is not flat") – otherwise, the algorithm might get stuck.

[11] In [48] a linear programming approach ("column generation") for simultaneously projection onto a growing set of hyperplanes has been considered. Some details are described in Section 6.2.

[12] These algorithms also work, when the equality constraints in (37) are replaced by inhomogeneous inequality constraints: Then one only projects to the hyperplane, when the constraint is not already satisfied.

[13] In the case of Bregman functions, the convex conjugate is given by $\mathrm{G}(\mathbf{o}) = \langle(\nabla\mathcal{G})^{-1}(\mathbf{o}), \mathbf{o}\rangle - \mathcal{G}((\nabla\mathcal{G})^{-1}(\mathbf{o}))$.

Relaxing Conditions on the Base learner. In Section 5.2 we made the assumption that the base learner always returns the hypothesis maximizing the edge (cf. (34)). In practice, however, it might be difficult to find the hypothesis that exactly minimizes the training error (i.e. maximizes the edge). Hence, some relaxations are necessary. One of the simplest approaches is to assume that the base learner always returns a hypothesis with some edge larger that say γ ("γ-relaxed", e.g. [70, 57]). This condition was used to show that AdaBoost's training error converges exponentially to zero. However, since the edge of a hypothesis is equal to the gradient with respect to its weight, this would mean that the gradient will not converge to zero. In the case of AdaBoost this actually happens and one converges to the minimum of the loss, but the length of the solution vector does not converge (cf. Section 3.2). More realistically one might assume that the base learner returns a hypothesis that approximately maximizes the edge – compared to the best possible hypothesis. [197] considered an additive term converging to zero ("α-relaxed") and [152] used a multiplicative term ("β-relaxed"). We summarize some senses of approximate edge-maximization (e.g. [70, 197, 151, 147])[14]

$$
\begin{aligned}
&\alpha\text{-relaxed:} \quad \sum_{n=1}^{N} d_n^t h_t(\boldsymbol{x}_n) \geq \operatorname*{argmax}_{h \in \mathsf{H}} \left\{ \sum_{n=1}^{N} d_n^t h(\boldsymbol{x}_n) \right\} - \alpha, \\
&\gamma\text{-relaxed:} \quad \sum_{n=1}^{N} d_n^t h_t(\boldsymbol{x}_n) \geq \gamma, \\
&\tau\text{-relaxed:} \quad \sum_{n=1}^{N} d_n^t h_t(\boldsymbol{x}_n) \geq \tau \left(\operatorname*{argmax}_{h \in \mathsf{H}} \left\{ \sum_{n=1}^{N} d_n^t h(\boldsymbol{x}_n) \right\} \right), \\
&\beta\text{-relaxed:} \quad \sum_{n=1}^{N} d_n^t h_t(\boldsymbol{x}_n) \geq \beta \operatorname*{argmax}_{h \in \mathsf{H}} \left\{ \sum_{n=1}^{N} d_n^t h(\boldsymbol{x}_n) \right\},
\end{aligned} \tag{40}
$$

for some fixed constants $\alpha > 0, \beta > 0, \gamma > 0$ and some strictly monotone and continuous function $\tau : \mathbb{R} \to \mathbb{R}$ with $\tau(0) = 0$.

Size of the Hypothesis Set. For most convergence results the size of the hypothesis set does not matter. However, in order to be able to attain the minimum, one has to assume that the set is bounded and closed (cf. [149]). In [39, 48, 151] finite hypothesis sets have been assumed in order to show convergence and rates. In the case of classification, where the base hypotheses are discrete valued, this holds true. However, in practice, one often wants to use real valued, parameterized hypotheses. Then the hypothesis set is uncountable. [197, 199] presents rather general convergence results for an algorithm related to that in Algorithm 5.2, which do not depend on the size of the hypothesis sets and is applicable to uncountable hypothesis sets.

[14] Note that the label y_n may be absorbed into the d's.

Regularization Terms on the Hypothesis Coefficients. In Section 6 we will discuss several regularization approaches for leveraging algorithms. One important ingredient is a penalty term on the hypothesis coefficients, i.e. one seeks to optimize

$$\min_{\boldsymbol{w}\in\mathbb{R}^J} \left\{ \sum_{n=1}^{N} \mathrm{g}(f_{\boldsymbol{w}}(\boldsymbol{x}_n), y_n) + C\mathbf{P}(\boldsymbol{w}) \right\} \tag{41}$$

where C is the regularization constant and $\mathbf{P}$ is some regularization operator (e.g. some ℓ_p-norm). We have seen that AdaBoost implicitly penalizes the ℓ_1-norm (cf. Section 4.3); this is also true for the algorithm described by [183, 154, 147, 197, 199]. The use of the ℓ_2-norm has been discussed in [56, 57, 183]. Only a small fraction of the analyses found in the literature allow the introduction of a regularization term. Note, when using regularized loss functions, the optimality conditions for the base learner change slightly (there will be one additional term depending on the regularization constant, cf. [147, 199]). Some regularization functions will be discussed in Section 6.3.

Convergence Rates. The first convergence result was obtained [70] for AdaBoost and the γ-relaxed condition on the base learner. It showed that the objective function is reduced at every step by a factor. Hence, it decreases exponentially. Here, the convergence speed does not directly depend on the size of the hypothesis class and the number of examples. Later this was generalized to essentially exponentially decreasing functions (like in Logistic Regression, but still α-relaxed condition, cf. [59]). Also, for finite hypothesis sets and strictly convex loss one can show the exponential decrease of the loss towards its minimum [151] (see also [117]) – the constants, however, depend heavily on the size of the training set and the number of base hypotheses. The best known rate for convex loss functions and arbitrary hypothesis sets is given in [197]. Here the rate is $\mathcal{O}(1/t)$ and the constants depend only on the smoothness of the loss function. In this approach it is mandatory to regularize with the ℓ_1-norm. Additionally, the proposed algorithm does not exactly fit into the scheme discussed above (see [197] for details).

Convexity with respect to parameters. In practical implementations of Boosting algorithms, the weak learners are parameterized by a finite set of parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_s)$, and the optimization needs to be performed with respect to $\boldsymbol{\theta}$ rather than with respect to the weak hypothesis h directly. Even if the loss function G is convex with respect to h, it is not necessarily convex with respect to $\boldsymbol{\theta}$. Although this problem may cause some numerical problems, we do not dwell upon it in this review.

6 Robustness, Regularization, and Soft-Margins

It has been shown that Boosting rarely overfits in the low noise regime (e.g. [54, 70, 167]); however, it clearly does so for higher noise levels (e.g. [145, 27, 81,

Table 1. Summary of Results on the Convergence of Greedy Approximation Methods

Ref.	Cost Function	Base Learner	Hypothesis Set	Convergence	Rate	Regularization
[70]	exponential loss	γ-relaxed	uncountable	global minimum, if loss zero	$\mathcal{O}(\mathrm{e}^{-t})$	no
[127]	Lipschitz differentiable	strict	uncountable	gradient zero	no	no
[127]	convex, Lipschitz differentiable	strict	uncountable	global minimum	no	$\|\cdot\|_1$ possible
[27]	convex, positive, differentiable	strict	countable	global minimum	no	$\|\cdot\|_1$ mandatory
[56]	essentially exp. decreasing	γ-relaxed	uncountable	global minimum, if loss zero	$\mathcal{O}(t^{-\frac{1}{2}})$, then $\mathcal{O}(\mathrm{e}^{-t})$	no
[39]	strictly convex, differentiable***	strict**	finite	global optimum (dual problem)	no	no
[48]	linear*	strict	finite	global minimum	no	$\|\cdot\|_1$ mandatory
[58]	squared loss, ε-loss	γ-relaxed	uncountable	global minimum		$\|\cdot\|_2$ mandatory
[151]	strictly convex, differentiable***	β-relaxed	finite	global minimum, unique dual sol.	$\mathcal{O}(\mathrm{e}^{-t})$	$\|\cdot\|_1$ possible
[149]	linear*	β-relaxed	uncountable, compact	global minimum	no	$\|\cdot\|_1$ possible
[147]	strictly convex	τ-relaxed	uncountable, compact	global minimum	no	$\|\cdot\|_1$ mandatory
[197]	convex	strict	uncountable	global infimum	$\mathcal{O}(1/\mathrm{t})$	$\|\cdot\|_1$ mandatory

* This also includes piecewise linear, convex functions like the soft-margin or the ε-insensitive loss.
** Extended to τ-relaxed in [147].
*** There are a few more technical assumptions. These functions are usually refereed to as functions of Legendre-type [162, 13].

153, 127, 12, 51]). In this section, we summarize techniques that yield state-of-the-art results and extend the applicability of boosting to the noisy case.

The margin distribution is central to the understanding of Boosting. We have shown in Section 4.3 that AdaBoost asymptotically achieves a *hard margin* separation, i.e. the algorithm concentrates its resources on a few hard-to-learn examples. Since a hard margin is clearly a sub-optimal strategy in the case of noisy data, some kind of regularization must be introduced in the algorithm to

alleviate the distortions that single difficult examples (e.g. outliers) can cause the decision boundary.

We will discuss several techniques to approach this problem. We start with algorithms that implement the intuitive idea of limiting the *influence* of a single example. First we present $AdaBoost_{Reg}$ [153] which trades off the influence with the margin, then discuss *BrownBoost* [65] which gives up on examples for which one cannot achieve large enough margins within a given number of iterations. We then discuss *SmoothBoost* [178] which prevents overfitting by disallowing overly skewed distributions, and finally briefly discuss some other approaches of the same flavor.

In Section 6.2 we discuss a second group of approaches which are motivated by the margin-bounds reviewed in Section 3.4. The important insight is that it is not the *minimal* margin which is to be maximized, but that the whole margin distribution is important: *one should accept small amounts of margin-errors on the training set, if this leads to considerable increments of the margin.* First we describe the *DOOM* approach [125] that uses a non-convex, monotone upper bound to the training error motivated from the margin-bounds. Then we discuss a linear program (LP) implementing a soft-margin [17, 153] and outline algorithms to iteratively solve the linear programs [154, 48, 146].

The latter techniques are based on modifying the AdaBoost margin loss function to achieve better noise robustness. However, DOOM as well as the LP approaches employ ℓ_∞-margins, which correspond to a ℓ_1-penalized cost function. In Section 6.3 we will discuss some issues related to other choices of the penalization term.

6.1 Reducing the Influence of Examples

The weights of examples which are hard to classify are increased at every step of AdaBoost. This leads to AdaBoost's tendency to exponentially decrease the training error and to generate combined hypotheses with large *minimal margin.* To discuss the suboptimal performance of such a *hard margin classifier* in the presence of outliers and mislabeled examples in a more abstract way, we analyze Figure 6. Let us first consider the noise free case [left]. Here, we can estimate a separating hyperplane correctly. In Figure 6 [middle] we have one outlier, which corrupts the estimation. Hard margin algorithms will concentrate on this outlier and impair the good estimate obtained in the absence of the outlier. Finally, let us consider more complex decision boundaries (cf. Figure 6 [right]). Here the overfitting problem is even more pronounced, if one can generate more and more complex functions by combining many hypotheses. Then all training examples (even mislabeled ones or outliers) can be classified correctly, which can result in poor generalization.

From these cartoons, it is apparent that AdaBoost and any other algorithm with large hard margins are *noise sensitive.* Therefore, we need to relax the hard margin and allow for a possibility of "mistrusting" the data. We present several algorithms which address this issue.

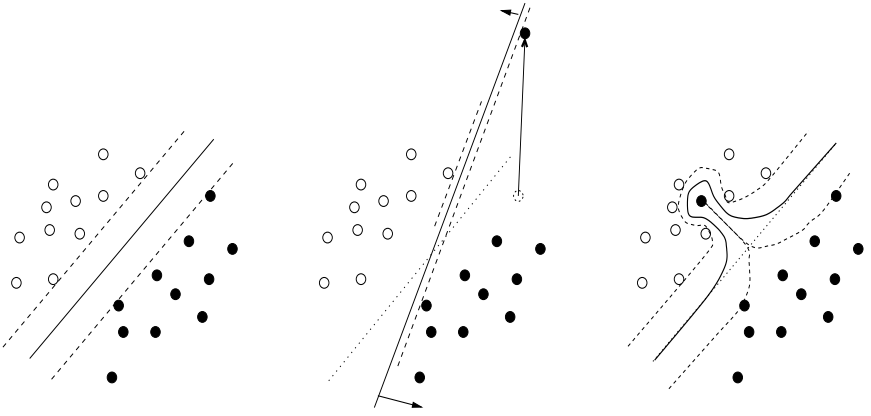

Fig. 6. The problem of finding a maximum margin "hyperplane" on reliable data [left], data with outlier [middle] and with a mislabeled example [right]. The solid line shows the resulting decision boundary, whereas the dashed line marks the margin area. In the middle and on the left the original decision boundary is plotted with dots. The hard margin implies noise sensitivity, because only one example can spoil the whole estimation of the decision boundary. (Figure taken from [153].)

AdaBoost$_{Reg}$. Examples that are mislabeled and usually more difficult to classify should not be forced to have a positive margin. If one knew beforehand which examples are "unreliable", one would simply remove them from the training set or, alternatively, not require that they have a large margin. Assume one has defined a non-negative *mistrust* parameter ζ_n, which expresses the "mistrust" in an example $(\boldsymbol{x}_n, y_n)$. Then one may relax the hard margin constraints (cf. (23) and (24)) leading to:

$$\max_{\boldsymbol{w},\rho} \ \rho \quad \text{s.t.} \quad \rho_n(\boldsymbol{w}) \geq \rho - C\zeta_n, \quad n = 1, \ldots, N \tag{42}$$

where $\rho_n(\boldsymbol{w}) = y_n \langle \boldsymbol{x}_n, \boldsymbol{w} \rangle / \|\boldsymbol{w}\|_1$, C is an *a priori* chosen constant.[15] Now one could maximize the margin using these modified constraints (cf. Section 4).

Two obvious questions arise from this discussion: first, how can one determine the ζ's and second, how can one incorporate the modified constraints into a boosting-like algorithm. In the AdaBoost$_{Reg}$ algorithm [153] one tries to solve both problems simultaneously. One uses the example weights computed at each iteration to determine which examples are highly influential and hard to classify. Assuming that the hard examples are "noisy examples", the algorithm chooses the mistrust parameter at iteration t, $\zeta_n^{(t)}$, as the amount by which the example $(\boldsymbol{x}_n, y_n)$ *influenced* the decision in previous iterations:

$$\zeta_n^{(t)} = \frac{\sum_{r=1}^{t} \alpha_r d_n^{(r)}}{\sum_{r=1}^{t} \alpha_r},$$

[15] Note that we use $\boldsymbol{w}$ to mark the parameters of the hyperplane in feature space and α to denote the coefficients generated by the algorithm (see Section 4.1).

where the α_t's and $d_n^{(t)}$'s are the weights for the hypotheses and examples, respectively. Moreover, one defines a new quantity, the *soft margin* $\tilde{\rho}_n(\boldsymbol{w}) := \rho_n(\boldsymbol{w}) + C\zeta_n$ of an example $(\boldsymbol{x}_n, y_n)$, which is used as a replacement of the margin in the AdaBoost algorithm. So, the idea in AdaBoost$_{Reg}$ is to find a solution with large *soft margin* on all examples, i.e. maximize ρ w.r.t. $\boldsymbol{w}$ and ρ such that $\tilde{\rho}_n(\boldsymbol{w}) \geq \rho$, $n = 1, \ldots, N$. In this case one expects to observe less overfitting.

One of the problems of AdaBoost$_{\text{Reg}}$ is that it lacks a convergence proof. Additionally, it is not clear what the underlying optimization problem is. The modification is done at an algorithmic level, which makes it difficult to relate to an optimization problem. Nevertheless, it was one of the first boosting-like algorithm that achieved state-of-the art generalization results on noisy data (cf. [146]). In experimental evaluations, it was found that this algorithm is among the best performing ones (cf. [153]).

BrownBoost. The *BrownBoost* algorithm [65] is based on the *Boosting by Majority* (BBM) algorithm [64]. An important difference between BBM and AdaBoost is that BBM uses a *pre-assigned number of boosting iterations*. As the algorithm approaches its predetermined termination, it becomes less and less likely that examples which have large negative margins will eventually become correctly labeled. Then, the algorithm "gives up" on those examples and concentrates its effort on those examples whose margin is, say, a small negative number [65]. Hence, the influence of the most difficult examples is reduced in the final iterations of the algorithm. This is similar in spirit to the AdaBoost$_{Reg}$ algorithm, where examples which are difficult to classify are assigned reduced weights in later iterations.

To use BBM one needs to pre-specify two parameters: (i) an upper bound on the error of the weak learner $\varepsilon \leq \frac{1}{2} - \frac{1}{2}\gamma$ (cf. Section 3.1) and (ii) a "target error" $\epsilon > 0$. In [65] it was shown that one can get rid of the γ-parameter as in AdaBoost. The target error ϵ is interpreted as the training error one wants to achieve. For noisy problems, one should aim for non-zero training error, while for noise free and separable data one can set ϵ to zero. It has been shown that letting ϵ in the BrownBoost algorithm approach zero, one recovers the original AdaBoost algorithm.

To derive BrownBoost, the following "thought experiment" [65] – inspired by ideas from the theory of Brownian motion – is made: Assume the base learner returns a hypothesis h with edge larger than γ. Fix some $0 < \delta < \gamma$ and define a new hypothesis h' which is equal to h with probability δ/γ and random otherwise. It is easy to check that the expected edge of h' is δ. Since the edge is small, the change of the combined hypothesis and the example weights will be small as well. Hence, the same hypothesis may have an edge greater than δ for several iterations (depending on δ) and one does not need to call the base learner again. The idea in BrownBoost is to let δ approach zero and analyze the above "dynamics" with differential equations. The resulting BrownBoost algorithm is very similar to AdaBoost but has some additional terms in the example weighting, which depend on the remaining number of iterations. In addition, in each iteration one needs to solve a differential equation with boundary conditions to compute how long a given hypothesis "survives" the above described cycle, which

determines its weight. More details on this algorithm and theoretical results on its performance can be found in [65]. Whereas the idea of the algorithm seems to be very promising, we are not aware of any empirical results illustrating the efficiency of the approach.

SmoothBoost. The skewness of the distributions generated during the Boosting process suggests that one approach to reducing the effect of overfitting is to impose limits on this skewness. A promising algorithm along these lines was recently suggested in [178] following earlier work [53]. The SmoothBoost algorithm was designed to work effectively with malicious noise, and is provably effective in this scenario, under appropriate conditions. The algorithm is similar to AdaBoost in maintaining a set of weights $d_n^{(t)}$ at each boosting iteration, except that there is a cutoff of the weight assigned to examples with very negative margins. The version of SmoothBoost described in [178] requires two parameters as input: (i) κ, which measures the desired error rate of the final classifier, (ii) and γ which measures the guaranteed edge of the hypothesis returned by the weak learner. Given these two parameters SmoothBoost can be shown to converge within a finite number of steps to a composite hypothesis f such that $|\{n : y_n f(\boldsymbol{x}_n) \leq \theta\}| < \kappa N$, where $\theta \leq \kappa$. Moreover, it can be shown that the weights generated during the process obey $d_n^{(t)} \leq 1/\kappa N$, namely the weights cannot become too large (compare with the ν-LP in Section 6.2). Although the convergence time of SmoothBoost is larger than AdaBoost, this is more than compensated for by the robustness property arising from the smoothness of the distribution. We observe that SmoothBoost operates with binary or real-valued hypotheses.

While SmoothBoost seems like a very promising algorithm for dealing with noisy situations, it should be kept in mind that it is not fully adaptive, in that both κ and γ need to be supplied in advance (cf. recent work by [30]). We are not aware of any applications of SmoothBoost to real data.

Other approaches aimed at directly reducing the effect of difficult examples can be found in [109, 6, 183].

6.2 Optimization of the Margins

Let us return to the analysis of AdaBoost based on margin distributions as discussed in Section 3.4. Consider a base-class of binary hypotheses H, characterized by VC dimension $\mathrm{VCdim}(\mathsf{H})$. From (16) and the fact that $R_N(\mathsf{H}) = \mathcal{O}(\sqrt{\mathrm{VCdim}(\mathsf{H})/N})$ we conclude that with probability at least $1 - \delta$ over the random draw of a training set S of size N the generalization error of a function $f \in \mathrm{co}(\mathsf{H})$ with margins $\rho_1, \ldots, \rho_N$ can be bounded by

$$L(f) \leq \frac{1}{N} \sum_{n=1}^{N} \mathbf{I}(\rho_n \leq \theta) + \mathcal{O}\left(\sqrt{\frac{\mathrm{VCdim}(\mathsf{H})}{N\theta^2}} + \sqrt{\frac{\log(1/\delta)}{2N}} \right) \tag{43}$$

where $\theta \in (0, 1]$ (cf. Corollary 1). We recall again that, perhaps surprisingly, the bound in (43) is independent of the number of hypotheses h_t that contribute

to defining $f = \sum_t \alpha_t h_t \in \text{co}(\mathsf{H})$. It was stated that a reason for the success of AdaBoost, compared to other ensemble learning methods (e.g. Bagging [25]), is that it generates combined hypotheses with large margins on the training examples. It asymptotically finds a linear combination $f_{\boldsymbol{w}}$ of base hypotheses satisfying

$$\rho_n(\boldsymbol{w}) = \frac{y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n)}{\sum_j w_j} \geq \rho \quad n = 1, \ldots, N, \tag{44}$$

for some large margin ρ (cf. Section 4). Then the first term of (43) can be made zero for $\theta = \rho$ and the second term becomes small, if ρ is large. In [27, 81, 157] algorithms have been proposed that generate combined hypotheses with even larger margins than AdaBoost. It was shown that as the margin increases, the generalization performance can become better on data sets with almost no noise (see also [167, 138, 157]). However, on problems with a large amount of noise, it has been found that the generalization ability often degrades for hypotheses with larger margins (see also [145, 27, 153]).

In Section 4 we have discussed connections of boosting to margin maximization. The algorithms under consideration approximately solve a linear programming problem, but tend to perform sub-optimally on noisy data. From the margin bound (43), this is indeed not surprising. The minimum on the right hand side of (43) is not necessarily achieved with the maximum (hard) margin (largest θ). In any event, one should keep in mind that (43) is only an upper bound, and that more sophisticated bounds (e.g. [180, 45, 181]), based on looking at the full margin distribution as opposed to the single hard margin θ, may lead to different conclusions.

In this section we discuss algorithms, where the number of margin errors can be controlled, and hence one is able to control the contribution of both terms in (43) separately. We first discuss the DOOM approach [125] and then present an extended linear program – the ν-LP problem – which allows for margin errors. Additionally, we briefly discuss approaches to solve the resulting linear program [48, 154, 147].

DOOM. (Direct Optimization Of Margins) The basic idea of [125, 127, 124] is to replace AdaBoost's exponential loss function with another loss function with ℓ_1-normalized hypothesis coefficients. In order to do this one defines a class of B-admissible margin cost functions, parameterized by some integer M.

These loss functions are motivated from the margin bounds of the kind discussed in Section 3.4, which have a free parameter θ. A large value of $M \sim \frac{1}{\theta^2}$ corresponds to a "high resolution" and a high effective complexity of the convex combination (small margin θ), whereas smaller values of M correspond to larger θ and therefore smaller effective complexity.

The B-admissibility condition ensures that the loss functions appropriately take care of the trade-off between complexity and empirical error. Following some motivation (which we omit here), [125, 127, 124] derived an optimal family of margin loss functions (according to the margin bound). Unfortunately, this loss function is non-monotone and non-convex (cf. Figure 7, [left]), leading to a very difficult optimization problem (NP hard).

The idea proposed in [125] is to replace this loss function with a piece-wise linear loss function, that is monotone (but non-convex, cf. Figure 7, [right]):

$$C_\theta(z) = \begin{cases} 1.1 - 0.1z & : -1 \le z \le 0, \\ 1.1 - z/\theta & : \ 0 \le z \le \theta, \\ 0.1(1-z)/(1-\theta) & : \ \theta \le z \le 1. \end{cases}$$

The optimization of this margin loss function to a global optimum is still very difficult, but good heuristics for optimization have been proposed [125, 127, 124]. Theoretically, however, one can only prove convergence to a local minimum (cf. Section 5.4).

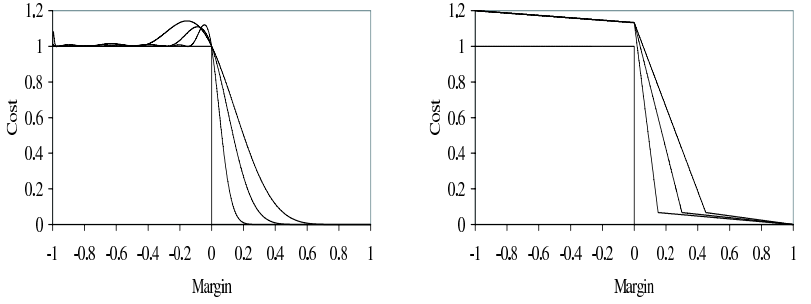

Fig. 7. [left] The "optimal" margin loss functions $C_M(z)$, for $M = 20, 50$ and 200, compared to the 0/1-loss. [right] Piecewise linear upper bound on the functions $C_M(z)$ and the 0/1-loss (Figure taken from [125].)

Despite the above mentioned problems the DOOM approach seems very promising. Experimental results have shown that, in noisy situations, one can considerably improve the performance compared to AdaBoost. However, there is the additional *regularization parameter* θ (or M), which needs to be chosen appropriately (e.g. via cross validation).

Linear Programming Approaches. We now discuss ideas aimed at directly controlling the number of margin errors. Doing so, one is able to directly control the contribution of both terms on the r.h.s. of (43) separately. We first discuss an extended linear program, the ν-LP problem, analyze its solution and review a few algorithms for solving this optimization problem.

The ν-LP Problem. Let us consider the case where we have given a set $\mathsf{H} = \{h_j : \boldsymbol{x} \mapsto [-1, 1], j = 1, \ldots, J\}$ of J hypotheses. To find the coefficients $\boldsymbol{w}$ for the combined hypothesis $f_{\boldsymbol{w}}(\boldsymbol{x})$ in (21), we extend the linear problem (30) and solve the following linear optimization problem (see also [17, 154]), which we call the ν-LP problem:

$$\begin{array}{ll} \max\limits_{\rho,\boldsymbol{w},\boldsymbol{\xi}} & \left\{\rho - \dfrac{1}{\nu N}\sum\limits_{n=1}^{N}\xi_n\right\} \\ \text{s.t.} & y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n) \geq \rho - \xi_n \quad n = 1,\ldots,N \\ & \xi_n,\ w_j \geq 0 \qquad\qquad\quad j = 1,\ldots,J, \quad n = 1,\ldots,N \\ & \sum_{j=1}^{J} w_j = 1, \end{array} \tag{45}$$

where $\nu \in (1/N, 1]$ is a parameter of the algorithm. Here, one does not force all margins to be large and obtains a *soft margin* hyperplane.[16] The dual optimization problem of (45) (cf. (46)) is the same as the edge minimization problem (20) with one additional constraint: $d_n \leq \frac{1}{\nu N}$. Since d_n is the Lagrange multiplier associated with the constraint for the n-th example, its size characterizes how much the example influences the solution of (45). In this sense, the ν-LP also implements the intuition discussed in Section 6.1. In addition, there seems to be a direct connection to the SmoothBoost algorithm.

The following proposition shows that ν has an immediate interpretation:

Proposition 2 (ν-Property, e.g. [174, 78, 150]). *The solution to the optimization problem (45) possesses the following properties:*

1. *ν upper-bounds the fraction of margin errors.*
2. *$1 - \nu$ is greater than the fraction of examples with a margin larger than ρ.*

Since the slack variables ξ_n only enter the cost function linearly, their gradient is constant (once $\xi_n > 0$) and the absolute size is not important. Loosely speaking, this is due to the fact that for the optimum of the primal objective function, only derivatives w.r.t. the primal variables matter, and the derivative of a linear function is constant. This can be made more explicit: Any example *outside* the margin area, i.e. satisfying $y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n) > \rho$, can be shifted arbitrarily, as long as it does not enter the margin area. Only if the example is exactly on the edge of the margin area, i.e. $y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n) = \rho$, then (almost) no local movement is possible without changing the solution. If the example $(\boldsymbol{x}_n, y_n)$ is in the margin area, i.e. $\xi_n > 0$, then it has been shown that almost any local movements are allowed (cf. [147] for details).

A Column Generation Approach. Recently, an algorithm for solving (45) has been proposed [48] (see also in the early work of [81]). It uses the *Column Generation (CG) method* known since the 1950's in the field of numerical optimization (e.g. [136], Section 7.4). The basic idea of column generation is to iteratively construct the *optimal* ensemble for a *restricted subset* of the hypothesis set, which is iteratively extended. To solve (45), one considers its dual optimization problem:

$$\begin{array}{ll} \min\limits_{\mathbf{d},\gamma} & \gamma \\ \text{s.t.} & \sum_{n=1}^{N} y_n d_n h_j(\boldsymbol{x}_n) \leq \gamma \quad j = 1,\ldots,J \\ & \mathbf{d} \geq \mathbf{0},\ \sum_{n=1}^{N} d_n = 1 \\ & d_n \leq \frac{1}{\nu N} \qquad\qquad n = 1,\ldots,N. \end{array} \tag{46}$$

[16] See Figure 4 for the soft margin loss, here one uses a scaled version.

At each iteration t, (46) is solved for a small subset of hypotheses $\mathsf{H}_t \subseteq \mathsf{H}$. Then the base learner is assumed to find a hypothesis h_t that violates the first constraint in (46) (cf. τ-relaxation in Section 5.4). If there exists such an hypothesis, then it is added to the restricted problem and the process is continued. This corresponds to generating a *column* in the primal LP (45). If all hypotheses satisfy their constraint, then the current dual variables and the ensemble (primal variables) are optimal, as all constraints of the *full* master problem are fulfilled.

The resulting algorithm is a special case of the set of algorithms known as *exchange methods* (cf. [93], Section 7 and references therein). These methods are known to converge (cf. [48, 149, 93] for finite and infinite hypothesis sets). To the best of our knowledge, no convergence rates are known, but if the base learner is able to provide "good" hypotheses/columns, then it is expected to converge much faster than simply solving the complete linear program. In practice, it has been found that the column generation algorithm halts at an optimal solution in a relatively small number of iterations. Experimental results in [48, 149] show the effectiveness of the approach. Compared to AdaBoost, one can achieve considerable improvements of the generalization performance [48].

A Barrier Algorithm. Another idea is to derive a barrier algorithm for (45) along the lines of Section 4.4. Problem (45) is reformulated: one removes the constant $\frac{1}{\nu N}$ in the objective of (45), fixes $\rho = 1$ and adds the sum of the hypothesis coefficients multiplied with another regularization constant C (instead of $\frac{1}{\nu N}$). One can in fact show that the modified problem has the same solution as (45): for any given ν one can find a C such that both problems have the same solution (up to scaling) [48]. The corresponding barrier objective F_β for this problem can be derived as in Section 4.4. It has two sets of parameters, the combination coefficients $\boldsymbol{w}$ and the slack variables $\boldsymbol{\xi} \in \mathbb{R}^N$. By setting $\nabla_{\boldsymbol{\xi}} \mathrm{F}_\beta = \mathbf{0}$, we can find the minimizing $\boldsymbol{\xi}$ for given β and $\boldsymbol{w}$, which one plugs in and obtains

$$\mathrm{F}_\beta(\boldsymbol{w}) = C \sum_{j=1}^{J} w_j + \beta \sum_{n=1}^{N} \log \left[1 + \exp\left(\frac{1 - y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n)}{\beta} \right) \right] + \beta N. \tag{47}$$

If one sets $\beta = 1$ and $C = 0$ (i.e. if we do not regularize), then one obtains a formulation which is very close to the logistic regression approach as in [74, 39]. The current approach can indeed be understood as a leveraging algorithm as in Section 5.2 *with regularization*. Furthermore, if we let β approach zero, then the scaled logistic loss in (47) converges to the *soft-margin loss* $\sum_{n=1}^{N} \max(0, 1 - y_n f_{\boldsymbol{w}}(\boldsymbol{x}_n))$.

Using the techniques discussed in Section 5, one can easily derive algorithms that optimize (47) for a fixed β up to a certain precision. Here one needs to take care of the ℓ_1-norm regularization, which can be directly incorporated into a coordinate descent algorithm (e.g. [147]). The idea is to reduce β, when a certain precision is reached and then the optimization is continued with the modified loss function. If one chooses the accuracy and the rate of decrease of β appropriately, one obtains a practical algorithm, for which one can show asymptotic convergence (for finite hypothesis sets see [147, 149]).

An earlier realization of the same idea was proposed in [154], and termed ν-Arc. The idea was to reformulate the linear program with soft margin into a non-linear program with maximum hard margin, by appropriately redefining the margin (similar to AdaBoost$_{Reg}$ in Section 6.1). Then, as a heuristic, Arc-GV (cf. Section 4.3) was employed to maximize this newly defined margin.[17]

The barrier approach and ν-Arc led to considerable improvements compared to AdaBoost on a large range of benchmark problems [154, 147]. Since they try to solve a similar optimization problem as the column generation algorithm, one would expect very minor differences. In practice, the column generation algorithm often converges faster, however, it has a problem in combination with some base learners [48]: In the CG approach, most of the example weights will be zero after a few iterations (since the margin in these examples is larger than ρ and the margin constraints are not active). Then the base learner may have problems to generate good hypotheses. Since the barrier approach is much "smoother", this effect is reduced considerably.

6.3 Regularization Terms and Sparseness

We will now discuss two main choices of regularizers that have a drastic influence on the properties of the solution of *regularized* problems. Let us consider optimization problems as in Section 5 with a regularization term in the objective:[18]

$$\begin{aligned} &\min_{\boldsymbol{w}} \left\{ \sum_{n=1}^{N} \mathrm{g}(f_{\boldsymbol{w}}(\boldsymbol{x}_n), y_n) + C \sum_{j=1}^{J} p(w_j) \right\} \\ &\text{s.t. } \mathbf{0} \leq \boldsymbol{w} \in \mathbb{R}^J, \end{aligned} \tag{48}$$

where C is a regularization constant, g is a loss function and $p : \mathbb{R}_+ \to \mathbb{R}_+$ is a differentiable and monotonically increasing function with $p(0) = 0$. For simplicity of the presentation we assume the hypothesis set is finite, but the statements below also hold for countable hypothesis sets; in the case of uncountable sets some further assumptions are required (cf. [93], Theorem 4.2, and [149, 148, 198]).

We say a set of hypothesis coefficients (see (21)) is sparse, if it contains $\mathcal{O}(N)$ non-zero elements. The set is not sparse if it contains e.g. $\mathcal{O}(J)$ non-zero elements, since the feature space is assumed to be much higher dimensional than N (e.g. infinite dimensional). We are interested in formulations of the form (48) for which the optimization problem is tractable, and which, at the same time, lead to sparse solutions. The motivation for the former aspect is clear, while the motivation for sparsity is that it often leads to superior generalization results (e.g. [91, 90]) and also to smaller, and therefore computationally more efficient, ensemble hypotheses. Moreover, in the case of infinite hypothesis spaces, sparsity leads to a precise representation in terms of a *finite* number of terms.

[17] A MATLAB implementation can be downloaded at `http://mlg.anu.edu.au/~raetsch/software`.

[18] Here we force the w's to be non-negative, which can be done without loss of generality, if the hypothesis set is closed under negation.

Let us first consider the ℓ_1-norm as regularizer, i.e. $p(w_j) = w_j$. Assume further that $\boldsymbol{w}^*$ is the optimal solution to (48) for some $C \geq 0$. Since the regularizer is linear, there will be a $(J-1)$-dimensional subspace of $\boldsymbol{w}$'s having the same ℓ_1-norm as $\boldsymbol{w}^*$. By further restricting $f_{\boldsymbol{w}}(\boldsymbol{x}_n) = f_{\boldsymbol{w}^*}(\boldsymbol{x}_n)$, $n = 1, \ldots, N$, one obtains N additional constraints. Hence, one can choose a $\boldsymbol{w}$ from a $J - N - 1$ dimensional space that leads to the same objective value as $\boldsymbol{w}^*$. Therefore, there exists a solution that has at most $N+1$ non-zero entries and, hence, the solution is sparse. This observations holds for arbitrary loss functions and any *concave* regularizer such as ℓ_1-norm [147]. Note that $N + 1$ is an upper bound on the number of non-zero elements – in practice much sparser solutions are observed [37, 48, 23, 8].

In neural networks, SVMs, matching pursuit (e.g. [118]) and many other algorithms, one uses the ℓ_2-norm for regularization. In this case the optimal solution $\boldsymbol{w}^*$ can be expressed as a linear combination of the mapped examples in feature space (cf. (26)),

$$\boldsymbol{w}^* = \sum_{n=1}^{N} \beta_n \Phi(\boldsymbol{x}_n).$$

Hence, if the vectors $\Phi(\boldsymbol{x}_n)$ are not sparse and the β's are not all zero, the solution $\boldsymbol{w}^*$ is also unlikely to be sparse (since it is a linear combination of non-sparse vectors). Consider, for instance, the case where the J vectors $(h_j(\boldsymbol{x}_1), \ldots, h_j(\boldsymbol{x}_N))$ $(j = 1, \ldots, J)$, interpreted as points in an N dimensional space are in *general position* (any subset of N points span the full N dimensional space. For instance, this occurs with probability 1 for points drawn independently at random from a probability density supported over the whole space). If $J \gg N$, then one can show that there is a fraction of at most $\mathcal{O}\left(\frac{N}{J-N}\right)$ coefficients that are zero. Hence, for large J almost all coefficients are non-zero and the solution is not sparse. This holds not only for the ℓ_2-norm regularization, but for any other strictly convex regularizer [147]. This observation can be intuitively explained as follows: There is a $J - N$ dimensional subspace leading to the same output $f_{\boldsymbol{w}}(\boldsymbol{x}_n)$ on the training examples. Assume the solution is sparse and one has a small number of large weights. If the regularizer is *strictly* convex, then one can reduce the regularization term by distributing large weights over many other weights which were previously zero (while keeping the loss term constant).

We can conclude that the type of regularizer determines whether there exists an optimal hypothesis coefficient vector that is sparse or not. Minimizing a convex loss function with a strictly concave regularizer will generally lead to non-convex optimization problems with potentially many local minima. Fortunately, the ℓ_1-norm is *concave and convex*. By employing the ℓ_1-norm regularization, one can obtain sparse solutions while retaining the computational tractability of the optimization problem. This observation seems to lend strong support to the ℓ_1-norm regularization.

7 Extensions

The bulk of the work discussed in this review deals with the case of binary classification and supervised learning. However, the general Boosting framework is much more widely applicable. In this section we present a brief survey of several extensions and generalizations, although many others exist, e.g. [76, 158, 62, 18, 31, 3, 155, 15, 44, 52, 82, 164, 66, 38, 137, 147, 16, 80, 185, 100, 14].

7.1 Single Class

A classical unsupervised learning task is density estimation. Assuming that the unlabeled observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ were generated independently at random according to some unknown distribution $P(\boldsymbol{x})$, the task is to estimate its related density function. However, there are several difficulties to this task. First, a density function need not always exist — there are distributions that do not possess a density. Second, estimating densities exactly is known to be a hard task. In many applications it is enough to estimate the support of a data distribution instead of the full density. In the *single-class approach* one avoids solving the harder density estimation problem and concentrate on the simpler task, i.e. estimating *quantiles* of the multivariate distribution.

So far there are two independent algorithms to solve the problem in a boosting-like manner. They mainly follow the ideas proposed in [184, 173] for kernel feature spaces, but have the benefit of better interpretability of the generated combined hypotheses (see discussion in [150]). The algorithms proposed in [32] and [150] differ in the way they solve very similar optimization problems – the first uses column generation techniques whereas the latter uses barrier optimization techniques (cf. Section 6.2). For brevity we will focus on the underlying idea and not the algorithmic details. As in the SVM case [173], one computes a hyperplane in the feature space (here spanned by the hypothesis set H, cf. Section 4.2) such that a pre-specified fraction of the training example will lie beyond that hyperplane, while at the same time demand that the hyperplane has maximal ℓ_∞-distance (margin) to the origin – for an illustration see Figure 8. This is realized by solving the following linear optimization problem:

$$\begin{array}{ll} \min\limits_{\boldsymbol{w}\in\mathbb{R}^J, \boldsymbol{\xi}\in\mathbb{R}^N, \rho\in\mathbb{R}} & -\nu\rho + \frac{1}{N}\sum\limits_{n=1}^{N}\xi_n \\ \text{s.t.} & \sum_{j=1}^{J} w_j h_j(\boldsymbol{x}_n) \geq \rho - \xi_n, \quad n = 1, \ldots, N \\ & \|\boldsymbol{w}\|_1 = 1, \xi_n \geq 0, \qquad\qquad n = 1, \ldots, N \end{array} \tag{49}$$

which is essentially the same as in the two-class soft-margin approach (cf. Section 6.2), but with all labels equal to +1. Hence, the derivation appropriate optimization algorithms is very similar.

7.2 Multi-class

There has been much recent interest in extending Boosting to multi-class problems, where the number of classes exceeds two. Let $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$

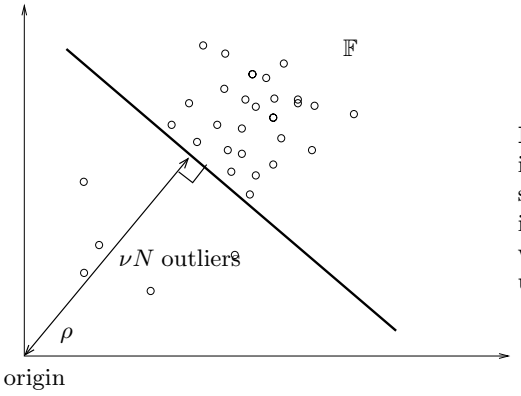

Fig. 8. Illustration of single-class idea. A hyperplane in the feature space $\mathbb{F}$ is constructed that maximizes the distance to the origin while allowing for ν outliers. (Figure taken from [134].)

be a data set, where, for a K-class problem, $\boldsymbol{y}_n$ is a K-dimensional vector of the form $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with a 1 in the k-th position if, and only if, $\boldsymbol{x}_n$ belongs to the k-th class. The log-likelihood function can then be constructed as [20] (see also Page 84)

$$G = \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ y_{n,k} \log p(k|\boldsymbol{x}_n) + (1 - y_{n,k}) \log(1 - p(k|\boldsymbol{x}_n)) \right\}, \tag{50}$$

where $y_{n,k}$ is the k'th component of the vector $\boldsymbol{y}_n$, and $p(k|\boldsymbol{x}_n)$ is the model probability that $\boldsymbol{x}_n$ belongs to the k'th class. Using the standard softmax representation,

$$p(k|\boldsymbol{x}) = \frac{e^{F_k(\boldsymbol{x})}}{\sum_{k'=1}^{K} e^{F_{k'}(\boldsymbol{x})}},$$

and substituting in (50) we obtain a function similar in form to others considered so far. This function can then be optimized in a stagewise fashion as described in Section 5.2, yielding a solution to the multi-class problem. This approach was essentially the one used in [74].

Several other approaches to multi-category classification were suggested in [168] and applied to text classification in [169]. Recall that in Boosting, the weights of examples which are poorly classified by previous weak learners become amplified. The basic idea in the methods suggested in [168] was to maintain a set of weights for both examples and labels. As boosting progresses, training examples and their corresponding labels that are hard to predict correctly get increasingly higher weights. This idea led to the development of two multi-category algorithms, titled AdaBoost.MH and AdaBoost.MR. Details of these algorithms can be found in [168].

In addition to these approaches, there has recently been extensive activity on recoding multi-class problems as a sequence of binary problems, using the

idea of error-correcting codes [52]. A detailed description of the approach can be found in [3] in a general context as well as in the Boosting setup. Two special cases of this general approach are the so-called *one-against-all* and *all-pairs* approaches, which were extensively used prior to the development of the error correcting code approach. In the former case, given a k-class classification problem, one constructs k (usually soft) binary classifiers each of which learns to distinguish one class from the rest. The multi-category classifier then uses the highest ranking soft classifier. In the all-pairs approach, all possible $\binom{k}{2}$ pairs of classification problems are learned and used to form the multi-category classifier using some form of majority voting. Using soft classifiers helps in removing possible redundancies. Connections of multi-class boosting algorithms with column generations techniques are discussed in [155].

Finally, we comment that an interesting problem relates to situations where the data is not only multi-class, but is also multi-labeled in the sense that each input $\boldsymbol{x}$ may belong to several classes [168]. This is particularly important in the context of text classification, where a single document may be relevant to several topics, e.g. sports, politics and violence.

7.3 Regression

The learning problem for regression is based on a data set $S = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, where in contrast to the classification case, the variable y can take on real values, rather than being restricted to a finite set. Probably the first approach to addressing regression in the context of Boosting appeared in [70], where the problem was addressed by translating the regression task into a classification problem (see also [18]). Most of the Boosting algorithms for binary classification described in this review, are based on a greedy stage-wise minimization of a smooth cost function. It is therefore not surprising that we can directly use such a smooth cost function in the context of regression where we are trying to estimate a smooth function. Probably the first work to discuss regression in the context of stage-wise minimization appeared in [76] (LS-Boost and other algorithms), and later further extended by others (e.g. [158, 182, 195]). The work in [156] further addressed regression in connection with barrier optimization (essentially with a two-sided soft-margin loss – the ϵ-insensitive loss). This approach is very similar in spirit to one of the approaches proposed in [57] (see below). Later it was extended in [149, 147] to arbitrary strictly convex loss functions and discussed in connection with infinite hypothesis sets and semi-infinite programming. The resulting algorithms were shown to be very useful in real world problems.

We briefly describe a regression framework recently introduced in [57], where several Boosting like algorithms were introduced. All these algorithms operate using the following scheme, where different cost functions are used for each algorithm. (i) At each iteration the algorithm modifies the sample to produce a new sample $\tilde{S} = \{(\boldsymbol{x}_1, \tilde{y}_1), \ldots, (\boldsymbol{x}_N, \tilde{y}_N)\}$ where the y's have been modified, based on the residual error at each stage. (ii) A distribution $\mathbf{d}$ over the modified samples $\tilde{S}$ is constructed and a base regression algorithm is called. (iii) The base learner produces a function $f \in \mathsf{F}$ with some edge on $\tilde{S}$ under $\mathbf{d}$ (for a definition of

an edge in the regression context see [57]). (iv) A new composite regressor of the form $F + \alpha f$ is formed, where α is selected by solving a one-dimensional optimization problem.

Theoretical performance guarantees for the regression algorithms introduced in [57] were analyzed in a similar fashion to that presented in Section 3 for binary classification. Under the assumption that a strictly positive edge was obtained at each step, it was shown that the training error converges exponentially fast to zero (this is analogous to Theorem 1 for classification). Significantly, the optimality of the base learners was not required for this claim, similarly to related results described in Section 5 for classification. Finally, generalization error bounds were derived in [57], analogous to Theorem 3. These bounds depend on the edge achieved by the weak learner.

7.4 Localized Boosting

All of the Boosting approaches discussed in this review construct a composite ensemble hypothesis of the form $\sum_t \alpha_t h_t(\boldsymbol{x})$. Observe that the combination parameters α_t do not depend on the input $\boldsymbol{x}$, and thus contribute equally at each point in space. An alternative approach to the construction of composite classifiers would be to allow the coefficients α_t themselves to depend on $\boldsymbol{x}$, leading to a hypothesis of the form $\sum_t \alpha_t(\boldsymbol{x}) h_t(\boldsymbol{x})$. The interpretation of this form of functional representation is rather appealing. Assume that each hypothesis $h_t(\boldsymbol{x})$ represents and expert, while $\alpha_t(\boldsymbol{x})$ assigns a non-negative weight which is attached to the prediction of the t-th expert, where we assume that $\sum_t \alpha_t(\boldsymbol{x}) = 1$. Given an input $\boldsymbol{x}$, each of the experts makes a prediction $h_t(\boldsymbol{x})$, where the prediction is weighted by a 'confidence' parameter $\alpha_t(\boldsymbol{x})$. Note that if $\alpha_t(\boldsymbol{x})$ are indicator functions, then for each input $\boldsymbol{x}$, there is a single hypothesis h_t which is considered. For linear hypotheses we can think of this approach as the representation of a general non-linear function by piece-wise linear functions. This type of representation forms the basis for the so-called mixture of experts models (e.g. [99]). An important observation concerns the complexity of the functions $\alpha_t(\boldsymbol{x})$. Clearly, by allowing these functions to be arbitrarily complex (e.g. δ-functions), we can easily fit any finite data set. This stresses the importance of regularization in any approach attempting to construct such a mixture of expert representation.

A Boosting like approach to the construction of mixture of experts representations was proposed in [129] and termed *Localized Boosting*. The basic observations in this work were the relationship to classic mixture models in statistics, where the EM algorithm has proven very effective, and the applicability of the general greedy stagewise gradient descent approaches described in Section 5.2. The algorithm developed in [129], termed *LocBoost*, can be thought of a stagewise EM algorithm, where similarly to Boosting algorithms a single hypothesis is added on to the ensemble at each stage, and the functions $\alpha_t(\cdot)$ are also estimated at each step. Regularization was achieved by restricting α_t to simple parametric forms. In addition to the algorithm described in detail in [129], gen-

eralization bounds similar to Theorem 3 were established. The algorithm has been applied to many real-world data-sets leading to performance competitive with other state-of-the-art algorithms.[19]

7.5 Other Extensions

We briefly mention other extensions of Boosting algorithms. In [140] a method was introduced for learning multiple models that use different (possibly overlapping) sets of features. In this way, more robust learning algorithms can be constructed. An algorithm combining Bagging and Boosting, aimed at improving performance in the case of noisy data was introduced in [109]. The procedure simply generates a set of Bootstrap samples as in Bagging, generates a boosted classifier for each sample, and combines the results uniformly. An online version of a Boosting algorithm was presented in [141], which was shown to be comparable in accuracy to Boosting, while much faster in terms of running time. Many more extensions are listed at the beginning of the present section.

8 Evaluation and Applications

8.1 On the Choice of Weak Learners for Boosting

A crucial ingredient for the successful application of Boosting algorithms is the construction of a good weak learner. As pointed out in Section 3, a weak learner which is too weak cannot guarantee adequate performance of the composite hypothesis. On the other hand, an overly complex weak learner may lead to overfitting, which becomes even more severe as the algorithm proceeds. Thus, as stressed in Section 6, regularization, in addition to an astute choice of weak learner, plays a key role in successful applications.

Empirically we often observed that a base learner that already performs quite well but are slightly too simple for the data at hand, are best suited for use with boosting. When one uses bagging, then using base learners that are slightly too complex perform best. Additionally, real-valued (soft) hypotheses often lead to considerable better results.

We briefly mention some weak learners which have been used successfully in applications.

Decision trees and stumps. Decision trees have been widely used for many years in the statistical literature (e.g. [29, 144, 85]) as powerful, effective and easily interpretable classification algorithms that are able to automatically select relevant features. Hence, it is not surprising that some of the most successful initial applications of Boosting algorithms were performed using decision trees as weak learners. Decision trees are based on the recursive partition of the input

[19] A MATLAB implementation of the algorithm, as part of an extensive Pattern Recognition toolbox, can be downloaded from
`http://tiger.technion.ac.il/~eladyt/classification/index.htm`.

space into a set of nested regions, usually of rectangular shape. The so-called *decision stump* is simply a one-level decision tree, namely a classifier formed by splitting the input space in an axis-parallel fashion once and then halting. A systematic approach to the effective utilization of trees in the context of Boosting is given in [168], while several other approaches based on logistic regression with decision trees are described in [76]. This work showed that Boosting significantly enhances the performance of decision trees and stumps.

Neural networks. Neural networks (e.g. [88, 20]) were extensively used during the 1990's in many applications. The feed-forward neural network, by far the most widely used in practice, is essentially a highly non-linear function representation formed by repeatedly combining a fixed non-linear transfer function. It has been shown in [113] that any real-valued continuous function over $\mathbb{R}^d$ can be arbitrarily well approximated by a feed-forward neural network with a single hidden layer, as long as the transfer function is *not* a polynomial. Given the potential power of neural networks in representing arbitrary continuous functions, it would seem that they could easily lead to overfitting and not work effectively in the context of Boosting. However, careful numerical experiments conducted by [176] demonstrated that AdaBoost can significantly improve the performance of neural network classifiers in real world applications such as OCR.

Kernel Functions and Linear combinations. One of the problems of using neural networks as weak learners is that the optimization problems often become unwieldy. An interesting alternative is forming a weak learner by linearly combining a set of fixed functions, as in

$$h^{\gamma}(\boldsymbol{x}) := \sum_{k=1}^{K} \beta_k G_k(\boldsymbol{x}), \quad \boldsymbol{\beta} \in \mathbb{R}^K. \tag{51}$$

The functions G_k could, for example, be kernel functions, i.e. $G_k(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_k)$, centered around the training examples. The set $\{h^{\boldsymbol{\beta}}\}$ is an infinite hypothesis set and is unbounded, if $\boldsymbol{\beta}$ is unbounded. Hence, maximizing the edge as discussed in Section 5.4, would lead to diverging β's. Keeping in mind the overfitting issues discussed in Section 6, we attempt to restrict the complexity of the class of functions by limiting the norm of $\boldsymbol{\beta}$. We discuss two approaches.

ℓ_1-norm Penalized Combinations. Here we set $\mathsf{H} := \{h^{\boldsymbol{\beta}} \mid \|\boldsymbol{\beta}\|_1 \leq 1, \boldsymbol{\beta} \in \mathbb{R}^N\}$. Then finding the edge-maximizing $\boldsymbol{\beta}$ has a closed form solution. Let

$$k^* = \operatorname*{argmax}_{k=1,\ldots,K} \left\{ \sum_{n=1}^{N} d_n g_k(\boldsymbol{x}_n) \right\}.$$

Then $h^{\boldsymbol{\beta}}$ with $\boldsymbol{\beta} = (0, \ldots, 0, \beta_{k^*}, 0, \ldots, \mathbf{0})$ maximizes the edge, where $\beta_{k^*} = \operatorname{sign}\left(\sum_{n=1}^{N} d_n k(\boldsymbol{x}_{k^*}, \boldsymbol{x}_n)\right)$. This means that we will be adding in exactly one basis function $g_{k^*}(\cdot)$ per iteration. Hence, this approach reduces to the case of

using the single functions. A variant of this approach has been used in [183]. A related classification function approach was used for solving a drug discovery task in [149].

In [149] so-called active kernel functions have been used, where the set of functions were kernel-like functions $h_{n,\mathbf{p}}(\boldsymbol{x}) = \mathrm{k}_{\mathbf{p}}(\boldsymbol{x}, \boldsymbol{x}_n)$, and $\mathrm{k}_{\mathbf{p}}$ is a SVM kernel function parameterized by $\mathbf{p}$ (e.g. the variance in a RBF kernel). In this case one has to find the example $\boldsymbol{x}_n$ *and* the parameter $\mathbf{p}$ that maximize the edge. This may lead to a hard non-convex optimization problem, for which quite efficient heuristics have been proposed in [149].

ℓ_2-norm Penalized combinations. Another way is to penalize the ℓ_2-norm of the combination coefficients. In this case $\mathsf{H} := \{h^{\boldsymbol{\gamma}} \mid \|\boldsymbol{\gamma}\|_2 \leq 1, \boldsymbol{\gamma} \in \mathbb{R}^N\}$, and one attempts to solve

$$\boldsymbol{\gamma} = \operatorname*{argmin}_{\boldsymbol{\gamma}} \left\{ \sum_{n=1}^{N} (d_n - h^{\boldsymbol{\gamma}}(\boldsymbol{x}_n))^2 + C\|\boldsymbol{\gamma}\|_2^2 \right\},$$

The problem has a particularly simple solution: $\gamma_k = \frac{1}{C} \sum_{n=1}^{N} d_n g_k(\boldsymbol{x}_n)$. A similar approach in the form of RBF networks has been used in [153] in connection with regularized versions of boosting.

8.2 Evaluation on Benchmark Data Sets

Boosting and Leveraging have been applied to many benchmark problems. The AdaBoost algorithm and many of its early variants were tested on standard data sets from the UCI repository, and often found to compare favorably with other state of the art algorithms (see, for example, [68, 168, 51]. However, it was clear from [146, 168, 51] that AdaBoost tends to overfit if the data is noisy and no regularization is enforced. More recent experiments, using the regularized forms of Boosting described in Section 6 lead to superior performance on noisy real-world data. In fact, these algorithms often do significantly better than the original versions of Boosting, comparing very favorably with the best classifiers available (e.g. SVMs). We refer the reader to [153] for results of these benchmark studies. These results place regularized boosting techniques into the standard toolbox of data analysis techniques.

8.3 Applications

The list of applications of Boosting and Leveraging methods is growing rapidly. We discuss three applications in detail and then list a range of other applications.

Non-intrusive Power Monitoring System. One of the regularized approaches described in Section 6, ν-Arc, was applied to the problem of power appliances monitoring [139]. The most difficult problem for power companies is the handling of *short-term peak loads* for which additional power plants need to be built to provide security against a peak load instigated power failure. A

prerequisite for controlling the electric energy demand, however, is the ability to correctly and non-intrusively detect and classify the operating status of electric appliances of individual households. The goal in [139] was to develop such a non-intrusive measuring system for assessing the status of electric appliances. This is a hard problem, in particular for appliances with inverter systems,[20] whereas non-intrusive measuring systems have already been developed for conventional on/off (non-inverter) operating electric equipments (cf. [83, 33]). The study in [139] presents a first evaluation of machine learning techniques to classify the operating status of electric appliances (with and without inverter) for the purpose of constructing a non-intrusive monitoring system. In this study, RBF networks, K-nearest neighbor classifiers (KNNs) (e.g. [42]), SVMs and ν-Arc (cf. Section 6.2) were compared.

The data set available for this task is rather small (36 examples), since the collection and labeling of data is manual and therefore expensive. As a result of this, one has to be very careful with finding good model parameters. All model parameters were found using the computationally expensive but generally reliable *leave-one-out* method.

The results reported in [139] demonstrate that the ν-Arc algorithm with RBF networks as base learner performs better on average than all other algorithms studied, followed closely by the SVM classifier with a RBF kernel. These results suggest that the goal of a control system to balance load peaks might be a feasible prospect in the not too distant future.

Tumor Classification with Gene Expression Data. Micro-array experiments generate large datasets with expression values for thousands of genes but not more than a few dozen of examples. Accurate supervised classification of tissue samples in such high-dimensional problems is difficult but often crucial for successful diagnosis and treatment (in typical cases the sample size is in the range of 20-100 and the number of features varies between 2,000 and 20,000; clearly here the potential for overfitting is huge). The goal is is to predict the unknown class label of a new individual on the basis of its gene expression profile. Since this task is of great potential value, there have been many attempts to develop effective classification procedures to solve it.

Early work applied the AdaBoost algorithm to this data; however, the results seemed to be rather disappointing. The recent work in [49] applied the LogitBoost algorithm [74], using decision trees as base learners, together with several modifications, and achieved state of the art performance on this difficult task. It turned out that in order to obtain high quality results, it was necessary to preprocess the data by scoring each individual feature (gene) according to its discriminatory power using a non-parametric approach (details can be found in [49]). Moreover, it was found that the simple *one-against-all* approach to multi-category classification led to much better results than the direct multi-class approach presented in [74] based on the log-likelihood function (cf. Section 7.2). Interestingly, the authors found that the quality of the

[20] An inverter system controls for example the rotation speed of a motor (as in air-conditioners) by changing the frequency of the electric current.

results degraded very little as a function of the number of Boosting iterations (up to 100 steps). This is somewhat surprising given the small amount of data and the danger of overfitting. The success of the present approach compared to results achieved by AdaBoost, tends to corroborate the assertions made in [74] concerning the effectiveness of the LogitBoost approach for noisy problems.

Text Classification. The problem of text classification is playing an increasingly important role due to the vast amount of data available over the web and within internal company repositories. The problem here is particularly challenging since the text data is often multi-labeled, namely each text may naturally fall into several categories simultaneously (e.g. Sports, Politics and Violence). In addition, the difficulty of finding an appropriate representation for text is still open.

The work in [169] presented one of the first approaches to using Boosting for text classification. In particular, the approaches to multi-class multi-label classification developed in [168], were used for the present task. The weak learner used was a simple decision stump (single level decision tree), based on terms consisting of single words and word pairs. The text categorization experiments reported in [169] were applied to several of the standard text classification benchmarks (Reuters, AP titles and UseNet groups) and demonstrated that the approach yielded, in general, better results than other methods to which it was compared. Additionally, it was observed that Boosting algorithms which used real-valued (soft) weak learners performed better than algorithms using only binary weak learners. A reported drawback of the approach was the very large time required for training.

Other applications. We briefly mention several applications of Boosting and Leveraging methods in other problems.

The group at AT&T has been involved in many applications of Boosting approaches beyond the text classification task discussed above. For example, the problems of text filtering [170] and routing [95] were addressed as well as that of ranking and combining references [66]. More recently the problem of combining prior knowledge and boosting for call classification in spoken language dialogue was studied in [161] and applications to the problem of modeling auction price uncertainty was introduced in [171].

Applications of boosting methods to natural language processing has been reported in [1, 60, 84, 194], and approaches to Melanoma Diagnosis are presented in [132]. Some further applications to Pose Invariant Face Recognition [94], Lung Cancer Cell Identification [200] and Volatility Estimation for Financial Time Series [7] have also been developed.

A detailed list of currently known applications of Boosting and Leveraging methods will be posted on the web at the Boosting homepage `http://www.boosting.org/applications`.

9 Conclusions

We have presented a general overview of ensemble methods in general and the Boosting and Leveraging family of algorithms in particular. While Boosting was introduced within a rather theoretical framework in order to transform a poorly performing learning algorithm into a powerful one, this idea has turned out to have manifold extensions and applications, as discussed in this survey.

As we have shown, Boosting turns out to belong to a large family of models which are greedily constructed by adding on a single base learner to a pool of previously constructed learners using adaptively determined weights. Interestingly, Boosting was shown to be derivable as a stagewise greedy gradient descent algorithm attempting to minimize a suitable cost function. In this sense Boosting is strongly related to other algorithms that were known within the Statistics literature for many years, in particular additive models [86] and matching pursuit [118]. However, the recent work on Boosting has brought to the fore many issues which were not studied previously. (i) The important concept of the margin, and its impact on learning and generalization has been emphasized. (ii) The derivation of sophisticated finite sample data-dependent bounds has been possible. (iii) Understanding the relationship between the strength of the weak learner, and the quality of the composite classifier (in terms of training and generalization errors). (iv) The establishment of consistency (v) The development of computationally efficient procedures.

With the emphasis of much of the Boosting work on the notion of the margin, it became clear that Boosting is strongly related to another very successful current algorithm, namely the Support Vector Machine [190]. As was pointed out in [167, 134, 150], both Boosting and the SVM can be viewed as attempting to maximize the margin, except that the norm used by each procedure is different. Moreover, the optimization procedures used in both cases are very different.

While Boosting constructs a complex composite hypothesis, which can in principle represent highly irregular functions, the generalization bounds for Boosting turn out to lead to tight bounds in cases where large margins can be guaranteed. Although initial work seemed to indicate that Boosting does not overfit, it was soon realized that overfitting does indeed occur under noisy conditions. Following this observation, regularized Boosting algorithms were developed which are able to achieve the appropriate balance between approximation and estimation required to achieve excellent performance even under noisy conditions. Regularization is also essential in order to establish consistency under general conditions.

We conclude with several open questions.

1. While it has been possible to derive Boosting-like algorithms based on many types of cost functions, there does not seem to be at this point a systematic approach to the selection of a particular one. Numerical experiments and some theoretical results indicate that the choice of cost function may have a significant effect of the performance of Boosting algorithms (e.g. [74, 126, 154]).
2. The selection of the best type of weak learner for a particular task is also not entirely clear. Some weak learners are unable even in principle to represent

complex decision boundaries, while overly complex weak learners quickly lead to overfitting. This problem appears strongly related to the notoriously difficult problem of feature selection and representation in pattern recognition, and the selection of the kernel in support vector machines. Note, however, that the single weak learner in Boosting can include multi-scaling information whereas in SVMs one has to fix a kernel inducing the kernel Hilbert space. An interesting question relates to the possibility of using very different types of weak learners at different stages of the Boosting algorithm, each of which may emphasize different aspects of the data.

3. An issue related to the previous one is the question of the existence of weak learners with provable performance guarantees. In Section 3.1 we discussed sufficient conditions for the case of linear classifiers. The extension of these results to general weak learners is an interesting and difficult open question.
4. A great deal of recent work has been devoted to the derivation of flexible data-dependent generalization bounds, which depend explicitly on the algorithm used. While these bounds are usually much tighter than classic bounds based on the VC dimension, there is still ample room for progress here, the final objective being to develop bounds which can be used for model selection in actual experiments on real data. Additionally, it would be interesting to develop bounds or efficient methods to compute the leave-one-out error as done for SVMs in [36].
5. In the optimization section we discussed many different approaches addressing the convergence of boosting-like algorithms. The result presented in [197] is the most general so far, since it includes many special cases which have been analyzed by others. However, the convergence rates in [197] do not seem to be optimal and additional effort needs to be devoted to finding tight bounds on the performance. In addition, there is the question of whether it is possible to establish super-linear convergence for some variant of leveraging, which ultimately would be lead to much more efficient leveraging algorithms. Finally, since many algorithms use parameterized weak learners, it is often the case that the cost function minimized by the weak learners is not convex with respect to the parameters (see Section 5.4). It would be interesting to see whether this problem could be circumvented (e.g. by designing appropriate cost functions as in [89]).

Acknowledgements. We thank Klaus-R. Müller for discussions and his contribution to writing this manuscript. Additionally, we thank Shie Mannor, Sebastian Mika, Takashi Onoda, Bernhard Schölkopf, Alex Smola and Tong Zhang for valuable discussions. R.M. acknowledges partial support from the Ollendorff Center at the Electrical Engineering department at the Technion and from the fund for promotion of research at the Technion. G.R. gratefully acknowledge partial support from DFG (JA 379/9-1, MU 987/1-1), NSF and EU (NeuroColt II). Furthermore, Gunnar Rätsch would like to thank UC Santa Cruz, CRIEPI in Tokyo and Fraunhofer FIRST in Berlin for their warm hospitality.

References

1. S. Abney, R.E. Schapire, and Y. Singer. Boosting applied to tagging and pp attachment. In *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.
2. H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automat. Control*, 19(6):716–723, 1974.
3. E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.
4. M. Anthony and P.L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.
5. A. Antos, B. Kégl, T. Linder, and G. Lugosi. Data-dependent margin-based generalization bounds for classification. *JMLR*, 3:73–98, 2002.
6. J.A. Aslam. Improving algorithms for boosting. In *Proc. COLT*, San Francisco, 2000. Morgan Kaufmann.
7. F. Audrino and P. Bühlmann. Volatility estimation with functional gradient descent for very high-dimensional financial time series. *Journal of Computational Finance.*, 2002. To appear. See `http://stat.ethz.ch/~buhlmann/bibliog.html`.
8. J.P. Barnes. Capacity control in boosting using a p-convex hull. Master's thesis, Australian National University, 1999. supervised by R.C. Williamson.
9. P. Bartlett, P. Boucheron, and G. Lugosi. Model selction and error estimation. *Machine Learning*, 48:85–2002, 2002.
10. P.L. Bartlett, O. Bousquet, and S. Mendelson. Localized rademacher averages. In *Procedings COLT'02*, volume 2375 of *LNAI*, pages 44–58, Sydney, 2002. Springer.
11. P.L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 2002. to appear 10/02.
12. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithm: Bagging, boosting and variants. *Machine Learning*, 36:105–142, 1999.
13. H.H. Bauschke and J.M. Borwein. Legendre functions and the method of random Bregman projections. *Journal of Convex Analysis*, 4:27–67, 1997.
14. S. Ben-David, P. Long, and Y. Mansour. Agnostic boosting. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 507–516, 2001.
15. K. P. Bennett and O. L. Mangasarian. Multicategory separation via linear programming. *Optimization Methods and Software*, 3:27–39, 1993.
16. K.P. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *Proc. ICML*, 2002.
17. K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.
18. 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.
19. D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.
20. C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

21. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.
22. B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.
23. P.S. Bradley and O.L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 82–90. Morgan Kaufmann, San Francisco, CA, 1998.
24. L.M. Bregman. The relaxation method for finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Math. and Math. Physics*, 7:200–127, 1967.
25. L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.
26. L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, July 1997.
27. L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1518, 1999. Also Technical Report 504, Statistics Department, University of California Berkeley.
28. L. Breiman. Some infinity theory for predictor ensembles. Technical Report 577, Berkeley, August 2000.
29. L. Breiman, J. Friedman, J. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.
30. N. Bshouty and D. Gavinsky. On boosting with polynomially bounded distributions. *JMLR*, pages 107–111, 2002. Accepted.
31. P. Buhlmann and B. Yu. Boosting with the l2 loss: Regression and classification. *J. Amer. Statist. Assoc.*, 2002. revised, also Technical Report 605, Stat Dept, UC Berkeley August, 2001.
32. C. Campbell and K.P. Bennett. A linear programming approach to novelty detection. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 395–401. MIT Press, 2001.
33. J. Carmichael. Non-intrusive appliance load monitoring system. Epri journal, Electric Power Research Institute, 1990.
34. Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms and Application*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997.
35. N. Cesa-Bianchi, A. Krogh, and M. Warmuth. Bounds on approximate steepest descent for likelihood maximization in exponential families. *IEEE Transaction on Information Theory*, 40(4):1215–1220, July 1994.
36. O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.
37. S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, 1995.
38. W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.
39. M. Collins, R.E. Schapire, and Y. Singer. Logistic Regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1-3):253–285, 2002. Special Issue on New Methods for Model Selection and Model Combination.
40. R. Cominetti and J.-P. Dussault. A stable exponential penalty algorithm with superlinear convergence. *J.O.T.A.*, 83(2), Nov 1994.
41. C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

42. T.M. Cover and P.E. Hart. Nearest neighbor pattern classifications. *IEEE transaction on information theory*, 13(1):21–27, 1967.
43. D.D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimates. *The Annals of Statistics*, 18(4):1676–1695, 1990.
44. K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In N. Cesa-Bianchi and S. Goldberg, editors, *Proc. Colt*, pages 35–46, San Francisco, 2000. Morgan Kaufmann.
45. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.
46. S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April 1997.
47. S. Della Pietra, V. Della Pietra, and J. Lafferty. Duality and auxiliary functions for Bregman distances. Technical Report CMU-CS-01-109, School of Computer Science, Carnegie Mellon University, 2001.
48. A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Journal of Machine Learning Research*, 46:225–254, 2002.
49. M. Dettling and P. Bühlmann. How to use boosting for tumor classification with gene expression data. Preprint. See `http://stat.ethz.ch/~dettling/boosting`, 2002.
50. L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of Mathematics. Springer, New York, 1996.
51. T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 1999.
52. T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Aritifical Intelligence Research*, 2:263–286, 1995.
53. C. Domingo and O. Watanabe. A modification of AdaBoost. In *Proc. COLT*, San Francisco, 2000. Morgan Kaufmann.
54. H. Drucker, C. Cortes, L.D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6, 1994.
55. H. Drucker, R.E. Schapire, and P.Y. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705–719, 1993.
56. N. Duffy and D.P. Helmbold. A geometric approach to leveraging weak learners. In P. Fischer and H. U. Simon, editors, *Computational Learning Theory: 4th European Conference (EuroCOLT '99)*, pages 18–33, March 1999. Long version to appear in TCS.
57. N. Duffy and D.P. Helmbold. Boosting methods for regression. Technical report, Department of Computer Science, University of Santa Cruz, 2000.
58. N. Duffy and D.P. Helmbold. Leveraging for regression. In *Proc. COLT*, pages 208–219, San Francisco, 2000. Morgan Kaufmann.
59. N. Duffy and D.P. Helmbold. Potential boosters? In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 258–264. MIT Press, 2000.
60. G. Escudero, L. Màrquez, and G. Rigau. Boosting applied to word sense disambiguation. In *LNAI 1810: Proceedings of the 12th European Conference on Machine Learning, ECML*, pages 129–141, Barcelona, Spain, 2000.
61. W. Feller. *An Introduction to Probability Theory and its Applications*. Wiley, Chichester, third edition, 1968.

62. D.H. Fisher, Jr., editor. *Improving regressors using boosting techniques*, 1997.
63. M. Frean and T. Downs. A simple cost function for boosting. Technical report, Dep. of Computer Science and Electrical Engineering, University of Queensland, 1998.
64. Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.
65. Y. Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001.
66. Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proc. ICML*, 1998.
67. 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.
68. Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.
69. Y. Freund and R.E. Schapire. Game theory, on-line prediction and boosting. In *Proc. COLT*, pages 325–332, New York, NY, 1996. ACM Press.
70. Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
71. Y. Freund and R.E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.
72. Y. Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999. Appeared in Japanese, translation by Naoki Abe.
73. J. Friedman. Stochastic gradient boosting. Technical report, Stanford University, March 1999.
74. J. Friedman, T. Hastie, and R.J. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 2:337–374, 2000. with discussion pp.375–407, also Technical Report at Department of Statistics, Sequoia Hall, Stanford University.
75. J.H. Friedman. On bias, variance, 0/1–loss, and the corse of dimensionality. In *Data Mining and Knowledge Discovery*, volume I, pages 55–77. Kluwer Academic Publishers, 1997.
76. J.H. Friedman. Greedy function approximation. Technical report, Department of Statistics, Stanford University, February 1999.
77. K.R. Frisch. The logarithmic potential method of convex programming. Memorandum, University Institute of Economics, Oslo, May 13 1955.
78. T. Graepel, R. Herbrich, B. Schölkopf, A.J. Smola, P.L. Bartlett, K.-R. Müller, K. Obermayer, and R.C. Williamson. Classification on proximity data with LP-machines. In D. Willshaw and A. Murray, editors, *Proceedings of ICANN'99*, volume 1, pages 304–309. IEE Press, 1999.
79. Y. Grandvalet. Bagging can stabilize without reducing variance. In *ICANN'01*, Lecture Notes in Computer Science. Springer, 2001.
80. Y. Grandvalet, F. D'alché-Buc, and C. Ambroise. Boosting mixture models for semi-supervised tasks. In *Proc. ICANN*, Vienna, Austria, 2001.
81. 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.

82. V. Guruswami and A. Sahai. Multiclass learning, boosing, and error-correcting codes. In *Proc. of the twelfth annual conference on Computational learning theory*, pages 145–155, New York, USA, 1999. ACM Press.
83. W. Hart. Non-intrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12), 1992.
84. M. Haruno, S. Shirai, and Y. Ooyama. Using decision trees to construct a practical parser. *Machine Learning*, 34:131–149, 1999.
85. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction*. Springer series in statistics. Springer, New York, N.Y., 2001.
86. T.J. Hastie and R.J. Tibshirani. *Generalized Additive Models*, volume 43 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 1990.
87. D. Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Information and Computation*, 100:78–150, 1992.
88. S.S. Haykin. *Neural Networks : A Comprehensive Foundation*. Prentice-Hall, second edition, 1998.
89. D.P. Helmbold, K. Kivinen, and M.K. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, 1999.
90. R. Herbrich. *Learning Linear Classifiers: Theory and Algorithms*, volume 7 of *Adaptive Computation and Machine Learning*. MIT Press, 2002.
91. R. Herbrich, T. Graepel, and J. Shawe-Taylor. Sparsity vs. large margins for linear classifiers. In *Proc. COLT*, pages 304–308, San Francisco, 2000. Morgan Kaufmann.
92. R. Herbrich and R. Williamson. Algorithmic luckiness. *JMLR*, 3:175–212, 2002.
93. R. Hettich and K.O. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, September 1993.
94. F.J. Huang, Z.-H. Zhou, H.-J. Zhang, and T. Chen. Pose invariant face recognition. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 245–250, Grenoble, France, 2000.
95. R.D. Iyer, D.D. Lewis, R.E. Schapire, Y. Singer, and A. Singhal. Boosting for document routing. In A. Agah, J. Callan, and E. Rundensteiner, editors, *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 70–77, McLean, US, 2000. ACM Press, New York, US.
96. W. James and C. Stein. Estimation with quadratic loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability*, volume 1, pages 361–380, Berkeley, 1960. University of California Press.
97. W. Jiang. Some theoretical aspects of boosting in the presence of noisy data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
98. D.S. Johnson and F.P. Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6:93–107, 1978.
99. M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.
100. M. Kearns and Y. Mansour. On the boosting ability og top-down decision tree learning algorithms. In *Proc. 28th ACM Symposium on the Theory of Computing,*, pages 459–468. ACM Press, 1996.
101. M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, January 1994.
102. M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

103. G.S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.
104. J. Kivinen and M. Warmuth. Boosting as entropy projection. In *Proc. 12th Annu. Conference on Comput. Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.
105. J. Kivinen, M. Warmuth, and P. Auer. The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. *Special issue of Artificial Intelligence*, 97(1–2):325–343, 1997.
106. J. Kivinen and M.K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, 1997.
107. K.C. Kiwiel. Relaxation methods for strictly convex regularizations of piecewise linear programs. *Applied Mathematics and Optimization*, 38:239–259, 1998.
108. V. Koltchinksii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Ann. Statis.*, 30(1), 2002.
109. A. Krieger, A. Wyner, and C. Long. Boosting noisy data. In *Proceedings, 18th ICML*. Morgan Kaufmann, 2001.
110. J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 125–133, New York, NY, 1999. ACM Press.
111. G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural information processings systems*, volume 14, 2002. to appear. Longer version also NeuroCOLT Technical Report NC-TR-2001-098.
112. Y.A. LeCun, L.D. Jackel, L. Bottou, A. Brunot, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, U.A. Müller, E. Säckinger, P.Y. Simard, and V.N. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman-Soulié and P. Gallinari, editors, *Proceedings ICANN'95 — International Conference on Artificial Neural Networks*, volume II, pages 53–60, Nanterre, France, 1995. EC2.
113. M. Leshno, V. Lin, A. Pinkus, and S. Schocken. Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate any Function. *Neural Networks*, 6:861–867, 1993.
114. N. Littlestone, P.M. Long, and M.K. Warmuth. On-line learning of linear functions. *Journal of Computational Complexity*, 5:1–23, 1995. Earlier version is Technical Report CRL-91-29 at UC Santa Cruz.
115. D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Co., Reading, second edition, May 1984. Reprinted with corrections in May, 1989.
116. Gábor Lugosi and Nicolas Vayatis. A consistent strategy for boosting algorithms. In *Proceedings of the Annual Conference on Computational Learning Theory*, volume 2375 of *LNAI*, pages 303–318, Sydney, February 2002. Springer.
117. Z.-Q. Luo and P. Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.
118. S. Mallat and Z. Zhang. Matching Pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.
119. O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.
120. O.L. Mangasarian. Arbitrary-norm separating plane. *Operation Research Letters*, 24(1):15–23, 1999.

121. S. Mannor and R. Meir. Geometric bounds for generlization in boosting. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 461–472, 2001.
122. S. Mannor and R. Meir. On the existence of weak learners and applications to boosting. *Machine Learning*, 48(1-3):219–251, 2002.
123. S. Mannor, R. Meir, and T. Zhang. The consistency of greedy algorithms for classification. In *Procedings COLT'02*, volume 2375 of *LNAI*, pages 319–333, Sydney, 2002. Springer.
124. L. Mason. *Margins and Combined Classifiers*. PhD thesis, Australian National University, September 1999.
125. L. Mason, P.L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. Technical report, Department of Systems Engineering, Australian National University, 1998.
126. L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A. J. Smola, P.L. Bartlett, B. Schölkopf, and C. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 1999.
127. L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 221–247. MIT Press, Cambridge, MA, 2000.
128. J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer Verlag, 1999.
129. R. Meir, R. El-Yaniv, and Shai Ben-David. Localized boosting. In *Proc. COLT*, pages 190–199, San Francisco, 2000. Morgan Kaufmann.
130. R. Meir and T. Zhang. Data-dependent bounds for bayesian mixture models. unpublished manuscript, 2002.
131. J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.
132. S. Merler, C. Furlanello, B. Larcher, and A. Sboner. Tuning cost-sensitive boosting and its application to melanoma diagnosis. In J. Kittler and F. Roli, editors, *Proceedings of the 2nd Internationa Workshop on Multiple Classifier Systems MCS2001*, volume 2096 of *LNCS*, pages 32–42. Springer, 2001.
133. J. Moody. The *effective* number of parameters: An analysis of generalization and regularization in non-linear learning systems. In S. J. Hanson J. Moody and R. P. Lippman, editors, *Advances in Neural information processings systems*, volume 4, pages 847–854, San Mateo, CA, 1992. Morgan Kaufman.
134. K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.
135. N. Murata, S. Amari, and S. Yoshizawa. Network information criterion — determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5:865–872, 1994.
136. S. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, NY, 1996.
137. Richard Nock and Patrice Lefaucheur. A robust boosting algorithm. In *Proc. 13th European Conference on Machine Learning*, volume LNAI 2430, Helsinki, 2002. Springer Verlag.

138. T. Onoda, G. Rätsch, and K.-R. Müller. An asymptotic analysis of AdaBoost in the binary classification case. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'98)*, pages 195–200, March 1998.
139. T. Onoda, G. Rätsch, and K.-R. Müller. A non-intrusive monitoring system for household electric appliances with inverters. In H. Bothe and R. Rojas, editors, *Proc. of NC'2000*, Berlin, 2000. ICSC Academic Press Canada/Switzerland.
140. J. O'Sullivan, J. Langford, R. Caruana, and A. Blum. Featureboost: A meta-learning algorithm that improves model robustness. In *Proceedings, 17th ICML*. Morgan Kaufmann, 2000.
141. N. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proc. KDD-01*, 2001.
142. R. El-Yaniv P. Derbeko and R. Meir. Variance optimized bagging. In *Proc. 13th European Conference on Machine Learning*, 2002.
143. T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.
144. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.
145. J.R. Quinlan. Boosting first-order learning. *Lecture Notes in Computer Science*, 1160:143, 1996.
146. G. Rätsch. Ensemble learning methods for classification. Master's thesis, Dep. of Computer Science, University of Potsdam, April 1998. In German.
147. G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Computer Science Dept., August-Bebel-Str. 89, 14482 Potsdam, Germany, October 2001.
148. G. Rätsch. Robustes boosting durch konvexe optimierung. In D. Wagner et al., editor, *Ausgezeichnete Informatikdissertationen 2001*, volume D-2 of *GI-Edition – Lecture Notes in Informatics (LNI)*, pages 125–136. Bonner Köllen, 2002.
149. G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1-3):193–221, 2002. Special Issue on New Methods for Model Selection and Model Combination. Also NeuroCOLT2 Technical Report NC-TR-2000-085.
150. G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller. Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE PAMI*, 24(9), September 2002. In press. Earlier version is GMD TechReport No. 119, 2000.
151. G. Rätsch, S. Mika, and M.K. Warmuth. On the convergence of leveraging. NeuroCOLT2 Technical Report 98, Royal Holloway College, London, August 2001. A short version appeared in NIPS 14, MIT Press, 2002.
152. G. Rätsch, S. Mika, and M.K. Warmuth. On the convergence of leveraging. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural information processings systems*, volume 14, 2002. In press. Longer version also NeuroCOLT Technical Report NC-TR-2001-098.
153. G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, March 2001. also NeuroCOLT Technical Report NC-TR-1998-021.
154. G. Rätsch, B. Schölkopf, A.J. Smola, S. Mika, T. Onoda, and K.-R. Müller. Robust ensemble learning. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 2000.
155. G. Rätsch, A.J. Smola, and S. Mika. Adapting codes and embeddings for polychotomies. In *NIPS*, volume 15. MIT Press, 2003. accepted.

156. G. Rätsch, M. Warmuth, S. Mika, T. Onoda, S. Lemm, and K.-R. Müller. Barrier boosting. In *Proc. COLT*, pages 170–179, San Francisco, 2000. Morgan Kaufmann.
157. G. Rätsch and M.K. Warmuth. Maximizing the margin with boosting. In *Proc. COLT*, volume 2375 of *LNAI*, pages 319–333, Sydney, 2002. Springer.
158. G. Ridgeway, D. Madigan, and T. Richardson. Boosting methodology for regression problems. In D. Heckerman and J. Whittaker, editors, *Proceedings of Artificial Intelligence and Statistics '99*, pages 152–161, 1999.
159. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
160. C. P. Robert. *The Bayesian Choice: A Decision Theoretic Motivation*. Springer Verlag, New York, 1994.
161. M. Rochery, R. Schapire, M. Rahim, N. Gupta, G. Riccardi, S. Bangalore, H. Alshawi, and S. Douglas. Combining prior knowledge and boosting for call classification in spoken language dialogue. In *International Conference on Accoustics, Speech and Signal Processing*, 2002.
162. R.T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathemathics. Princeton University Press, New Jersey, 1970.
163. R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
164. R.E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the 14th International Conference*, pages 313–321, 1997.
165. R.E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.
166. R.E. Schapire. The boosting approach to machine learning: An overview. In *Workshop on Nonlinear Estimation and Classification*. MSRI, 2002.
167. R.E. Schapire, Y. Freund, P.L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.
168. R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999. also Proceedings of the 14th Workshop on Computational Learning Theory 1998, pages 80–91.
169. R.E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
170. R.E. Schapire, Y. Singer, and A. Singhal. Boosting and rocchio applied to text filtering. In *Proc. 21st Annual International Conference on Research and Development in Information Retrieval*, 1998.
171. R.E. Schapire, P. Stone, D. McAllester, M.L. Littman, and J.A. Csirik. Modeling auction price uncertainty using boosting-based conditional density estimations noise. In *Proceedings of the Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.
172. B. Schölkopf, R. Herbrich, and A.J. Smola. A generalized representer theorem. In D.P. Helmbold and R.C. Williamson, editors, *COLT/EuroCOLT*, volume 2111 of *LNAI*, pages 416–426. Springer, 2001.
173. B. Schölkopf, J. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. TR 87, Microsoft Research, Redmond, WA, 1999.
174. B. Schölkopf, A. Smola, R.C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000. also NeuroCOLT Technical Report NC-TR-1998-031.

175. B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
176. H. Schwenk and Y. Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.
177. R.A. Servedio. PAC analogoues of perceptron and winnow via boosting the margin. In *Proc. COLT*, pages 148–157, San Francisco, 2000. Morgan Kaufmann.
178. R.A. Servedio. Smooth boosting and learning with malicious noise. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 473–489, 2001.
179. J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. Inf. Theory*, 44(5):1926–1940, September 1998.
180. J. Shawe-Taylor and N. Cristianini. Further results on the margin distribution. In *Proceedings of the twelfth Conference on Computational Learning Theory*, pages 278–285, 1999.
181. J. Shawe-Taylor and N. Cristianini. On the genralization of soft margin algorithms. Technical Report NC-TR-2000-082, NeuroCOLT2, June 2001.
182. J. Shawe-Taylor and G. Karakoulas. Towards a strategy for boosting regressors. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 247–258, Cambridge, MA, 2000. MIT Press.
183. Y. Singer. Leveraged vector machines. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 610–616. MIT Press, 2000.
184. D. Tax and R. Duin. Data domain description by support vectors. In M. Verleysen, editor, *Proc. ESANN*, pages 251–256, Brussels, 1999. D. Facto Press.
185. F. Thollard, M. Sebban, and P. Ezequel. Boosting density function estimators. In *Proc. 13th European Conference on Machine Learning*, volume LNAI 2430, pages 431–443, Helsinki, 2002. Springer Verlag.
186. A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems*. W.H. Winston, Washington, D.C., 1977.
187. K. Tsuda, M. Sugiyama, and K.-R. Müller. Subspace information criterion for non-quadratic regularizers – model selection for sparse regressors. *IEEE Transactions on Neural Networks*, 13(1):70–80, 2002.
188. L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
189. A.W. van der Vaart and J.A. Wellner. *Weak Convergence and Empirical Processes*. Springer Verlag, New York, 1996.
190. V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.
191. V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
192. V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.
193. J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Math. Ann.*, 100:295–320, 1928.
194. M.A. Walker, O. Rambow, and M. Rogati. Spot: A trainable sentence planner. In *Proc. 2nd Annual Meeting of the North American Chapter of the Assiciation for Computational Linguistics*, 2001.

195. R. Zemel and T. Pitassi. A gradient-based boosting algorithm for regression problems. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 696–702. MIT Press, 2001.
196. T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. Technical Report RC22155, IBM Research, Yorktown Heights, NY, 2001.
197. T. Zhang. A general greedy approximation algorithm with applications. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.
198. T. Zhang. On the dual formulation of regularized linear systems with convex risks. *Machine Learning*, 46:91–129, 2002.
199. T. Zhang. Sequential greedy approximation for certain convex optimization problems. Technical report, IBM T.J. Watson Research Center, 2002.
200. Z.-H. Zhou, Y. Jiang, Y.-B. Yang, and S.-F. Chen. Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, 24(1):25–36, 2002.