Chapter 2
Boosting Algorithms: A Review of Methods, Theory, and Applications

Artur J. Ferreira and Mário A.T. Figueiredo

2.1 Introduction

Boosting is a class of machine learning methods based on the idea that a combination of simple classifiers (obtained by a *weak learner*) can perform better than any of the simple classifiers alone. A *weak learner* (WL) is a learning algorithm capable of producing classifiers with probability of error strictly (but only slightly) less than that of random guessing (0.5, in the binary case). On the other hand, a *strong learner* (SL) is able (given enough training data) to yield classifiers with arbitrarily small error probability.

An ensemble (or committee) of classifiers is a classifier build upon some combination of WLs. The strategy of boosting, and ensembles of classifiers, is to learn many *weak* classifiers and combine them in some way, instead of trying to learn a single *strong* classifier. This idea of building ensembles of classifiers has gained interest in the last decade [67]; the rationale is that it may be easier to train several simple classifiers and combine them into a more complex classifier than to learn a single complex classifier. For instance, instead of training a large *neural network* (NN), we may train several simpler NNs and combine their individual outputs in order to produce the final output (as illustrated in Fig. 2.1).

Letting $H_m: \mathscr{X} \rightarrow \{-1, +1\}$ be the mth weak binary classifier (for $m = 1, \ldots, M$), and $\mathbf{x} \in \mathscr{X}$ some input pattern to be classified, there are many ways to combine the outputs $H_1(\mathbf{x}), \ldots, H_M(\mathbf{x})$ into a single class prediction [67].

A.J. Ferreira (✉)
Instituto de Telecomunicações, and Instituto Superior de Engenharia de Lisboa – Polytechnic Institute of Lisbon, ADEETC – Gabinete 16, Rua Conselheiro Emidio Navarro, 1959-007 Lisboa, Portugal
e-mail: arturj@isel.pt

M.A.T. Figueiredo
Instituto de Telecomunicações, and Instituto Superior Técnico – Technical University of Lisbon, Torre Norte, Piso 10, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
e-mail: mario.figueiredo@lx.it.pt

C. Zhang and Y. Ma (eds.), *Ensemble Machine Learning: Methods and Applications*,
DOI 10.1007/978-1-4419-9326-7_2, © Springer Science+Business Media, LLC 2012

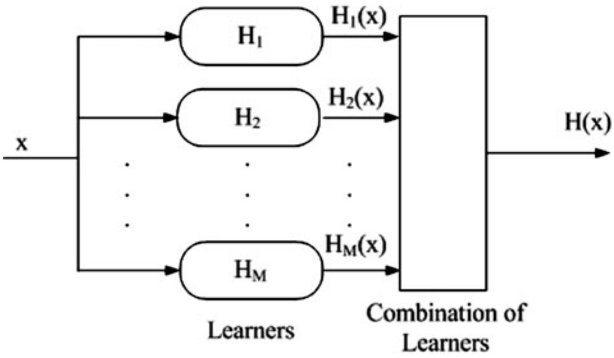

Fig. 2.1 The concept of ensemble of classifiers. The outputs of the weak learners $H_m(\mathbf{x})$ with $m \in \{1, \ldots, M\}$ are combined to produce the output of the ensemble of classifiers given by $H(\mathbf{x})$

For example, assuming that the classifiers err independently of each other, a majority vote combination should yield a lower probability of error than any of the individual classifiers. Considering a weighted linear combination of the outputs of the weak classifiers, the ensemble prediction function $H : \mathscr{X} \rightarrow \{-1, +1\}$ is given by

$$H(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m \; H_m(\mathbf{x})\right), \tag{2.1}$$

where $\alpha_1, \ldots, \alpha_M$ is a set of weights (a simple majority vote results if all the weights are equal).

Among the many different ways in which ensembles of classifiers can be learned and combined [67], boosting techniques exhibit, in addition to good practical performance, several theoretical and algorithmic features that makes them particularly attractive [58, 82, 98]. Essentially, boosting consists of repeatedly using the base weak learning algorithm, on differently weighted versions of the training data, yielding a sequence of weak classifiers that are combined as in (2.1). The weighting of each instance in the training data, at each round of the algorithm, depends on the accuracy of the previous classifiers, thus allowing the algorithm to focus its *attention* on those samples that are still incorrectly classified. The several variants of boosting algorithms differ in their choice of base learners and criterion for updating the weights of the training samples. AdaBoost (which stands for *adaptive boosting*) is arguably the best-known boosting algorithm, and was responsible for sparking the explosion of interest in this class of algorithms that happened after the publication of the seminal works of Freund and Schapire [47–50],

2.1.1 Chapter Outline

The remaining sections of this chapter are organized as follows. Section 2.2 addresses the foundations and origins of boosting algorithms, as a class of methods to improve the accuracy of learning algorithms, by building ensembles of classifiers; the connection of boosting with other machine learning techniques, such as bootstrap and bagging is mentioned. Section 2.3 describes AdaBoost and discusses some of its theoretical properties, regarding training error (TE), generalization error (GE), and the problem of overfitting. In Section 2.4, we describe variants of AdaBoost and their properties, including extensions for multiclass problems, while boosting algorithms for semi-supervised learning (SSL) are discussed in Section 2.5. Section 2.6 discusses several successful applications of batch and online boosting algorithms and presents an experimental evaluation of some boosting algorithms, compared to other machine learning techniques on standard benchmark datasets. Section 2.7 provides a summary and a discussion on boosting algorithms. Finally, Section 2.8 ends the chapter with some bibliographic and historical remarks.

2.2 The Origins of Boosting and Adaptive Boosting

2.2.1 Bootstrapping and Bagging

Bootstrapping [37, 38] is a general purpose sample-based statistical method in which several (nondisjoint) training sets are obtained by drawing randomly, *with* replacement, from a single base dataset. In a dataset with N samples, each instance is selected with probability $1/N$; consequently, after N draws (with large N), the probability that a given instance was not selected is

$$\left(1 - \frac{1}{N}\right)^N \approx \exp(-1) \approx 0.368; \tag{2.2}$$

the validity of this approximation is illustrated in Fig. 2.2, showing that it is quite accurate even with only a moderately large N. This implies that each sample contains roughly 63.2% of the instances.

Classically, bootstrapping is used to infer some statistic $T(P)$ about a (say infinitely large) population P, from N samples thereof: $Z = \{z_1, \ldots, z_N\}$. The idea is to obtain B sets $Z_b^* \subseteq Z$, for $b = 1, \ldots, B$, each containing N random samples (with replacement) from Z, from which B estimates of $T(P)$ are obtained. These estimates are then averaged into a final estimate; it is also possible to obtain variance estimates or confidence intervals. The procedure is formally described in Algorithm 1.

Bagging (which stands for *bootstrap aggregation* [11]) is a technique which uses bootstrap sampling to reduce the variance and/or improve the accuracy of some predictor (it may be used in classification and regression). Consider a size-N dataset $Z = \{z_1, z_2, \ldots, z_N\}$, where now $z_i = (\mathbf{x_i}, y_i)$, where y_i is a class label, in classification problems, or a real number, in regression problems. The rationale of bagging is to learn a set of B predictors (each from a bootstrap sample $Z_b^* \subseteq Z$,

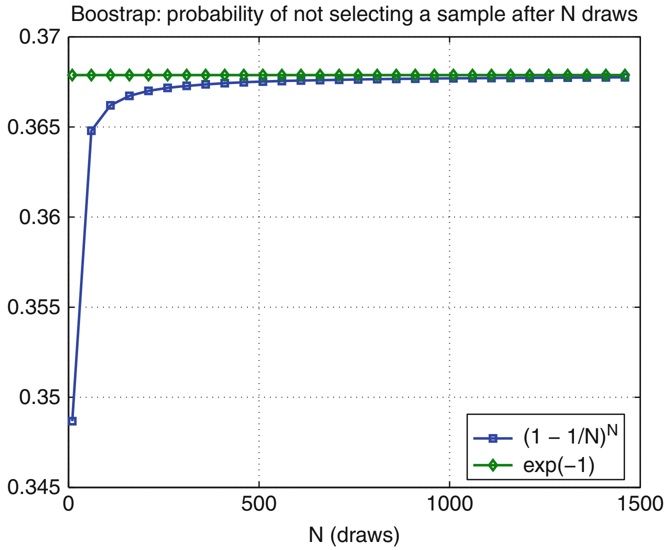

Fig. 2.2 The Bootstrap procedure: probability of not selecting a training sample after N draws and its approximation $\exp(-1)$

Algorithm 1 Bootstrap Procedure

Input: Size-N sample $Z = \{z_1, z_2, \dots, z_N\}$ of a (potentially infinite) population P.
B, number of bootstrap samples.

Output: Estimate $\widehat{T}(P)$ of the population statistic.

1: **for** $b = 1$ to B **do**
2: Draw, with replacement, N samples from Z, obtaining the bth bootstrap sample Z_b^*.
3: Compute, for each sample Z_b^*, the estimate of the statistic $\widehat{T}(Z_b^*)$.
4: **end for**
5: Compute the bootstrap estimate $\widehat{T}(P)$ as the average of $\widehat{T}(Z_1^*),\dots,\widehat{T}(Z_B^*)$.
6: Compute the accuracy of the estimate, using, e.g., the variance of $\widehat{T}(Z_1^*),\dots,\widehat{T}(Z_B^*)$.

for $b = 1, \dots, B$) and then produce a final predictor by combining (by averaging, in regression, or majority voting, in classification) this set of predictors. The combination of multiple predictors decreases the expected error because it reduces the variance component of the bias–variance decomposition [58]. The reduction on this variance component is proportional to the number of classifiers applied in the ensemble. The bagging procedure, for binary classification, is described in Algorithm 2.

As compared to the process of learning a classifier in a conventional way, that is, from the full training set, bagging has two main advantages:

- increases classifier stability and accuracy;
- reduces classifier variance, in terms of the bias–variance decomposition [58].

Algorithm 2 Bagging Procedure for Classification

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
B, number of bootstrap samples.
Output: $H : \mathscr{X} \rightarrow \{-1, +1\}$, the final classifier.

1: **for** $b = 1$ to B **do**
2: Draw, with replacement, N samples from Z, obtaining the bth bootstrap sample Z_b^*.
3: From each bootstrap sample Z_b^*, learn classifier H_b.
4: **end for**
5: Produce the final classifier by a majority vote of $H_1, \ldots, H_B$, that is, $H(\mathbf{x}) = \text{sign}\left(\sum_{b=1}^{B} H_b(\mathbf{x})\right)$.

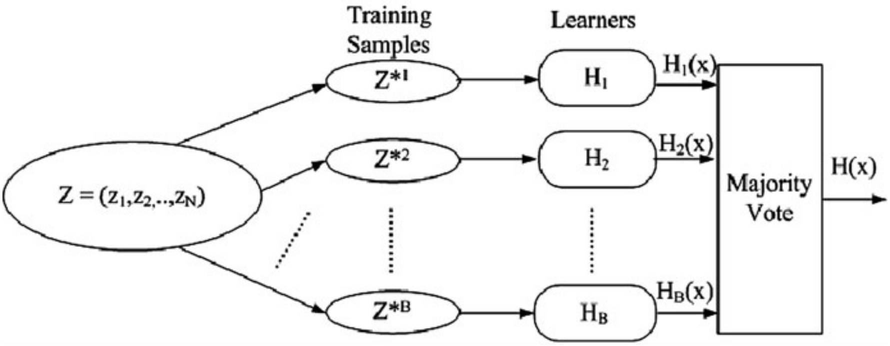

Fig. 2.3 The bagging approach to classification. Using bootstrap, we produce several training samples; each of these samples is fed into a weak learner. The final classification decision is produced by a majority vote on the weak learners output

The use of the bagging technique improves the classification results whenever the base classifiers are unstable, this being the main reason why the bagging approach works well for classification. Figure 2.3 depicts the bagging approach for classification.

For further reading on bagging, see [12, 13, 95, 139, 140]. In [139], the authors argue that for very weak learners (e.g., decision stumps, which are tree classifier with only one inner node), the base classifiers built from bootstrap samples are strongly correlated. As a consequence, a simple bagged classifier with these very weak learners has very little improvement compared to a single classifier trained from the same data. To overcome this problem, they propose the *local lazy learning bagging* (LLLB) approach, where base learners are trained from a small subset surrounding each test instance. The experimental results on real-world datasets show that the LLLB method significantly outperforms standard bagging.

2.2.2 Weak and Strong Learners

Weak and *strong* learning are fundamental concepts at the heart of boosting algorithms, so we briefly review their formal definitions. These concepts are rooted in the theory of PAC (probably approximately correct) learning [114], where they are defined as follows. Consider an hypothesis, i.e., a classification rule $f : \mathscr{X} \rightarrow \{-1, +1\}$, such that $f \in \mathscr{F}$, where $\mathscr{F}$ is some class of functions from $\mathscr{X}$ to $\{-1, +1\}$. Consider also a set of examples of that hypothesis, i.e., a set of pairs $\{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$ such that $y_i = f(\mathbf{x}_i)$ and the $\mathbf{x}_i$ are samples of some distribution P. A *strong* learner is capable of, given enough data, producing an arbitrarily good classifier with high probability, that is, for every P, $f \in \mathscr{F}$, $\varepsilon \geq 0$, and $\delta \leq 1/2$, it outputs, with probability no less than $1 - \delta$, a classifier $h : \mathscr{X} \rightarrow \{-1, +1\}$ satisfying $\mathbb{P}_P [h(\mathbf{x}) \neq f(\mathbf{x})] \leq \varepsilon$. Furthermore, the time complexity of the algorithm can be at most polynomial in $1/\varepsilon$, $1/\delta$, N, and the dimension of $\mathscr{X}$ [82].

A WL is formally defined in a similar way as a strong one, but with weaker quantification with respect to ε and δ. Given a particular (rather than "for every") pair $\varepsilon_0 \geq 0$, and $\delta_0 \leq 1/2$, a WL outputs, with probability no less than $1 - \delta_0$, a classifier $h : \mathscr{X} \rightarrow \{-1, +1\}$ satisfying $\mathbb{P}_P [h(\mathbf{x}) \neq f(\mathbf{x})] \leq \varepsilon_0$. Underlying the idea of boosting is the fact, proved by Schapire [95] that it is possible to obtain a SL by combining WLs.

2.2.3 Boosting Algorithms

The first boosting procedure was proposed by Schapire in [95], where the key result is that weak and strong learnability are equivalent, in the sense that strong learning can be performed by combining WLs. The boosting procedure proposed in [95] is described in detail in Algorithm 3.

Algorithm 3 Boosting Procedure for Classification

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
Output: A classifier $H : \mathscr{X} \rightarrow \{-1, +1\}$.

1: Randomly select, without replacement, $L_1 < N$ samples from Z to obtain Z_1^*.
2: Run the WL on Z_1^*, yielding classifier H_1.
3: Select $L_2 < N$ samples from Z, with half of the samples misclassified by H_1, to obtain Z_2^*.
4: Run the WL on Z_2^*, yielding classifier H_2.
5: Select all samples from Z on which H_1 and H_2 disagree, producing Z_3^*.
6: Run the WL on Z_3^*, yielding classifier H_3.
7: Produce the final classifier as a majority vote: $H(\mathbf{x}) = \text{sign}\left(\sum_{b=1}^{3} H_b(\mathbf{x})\right)$.

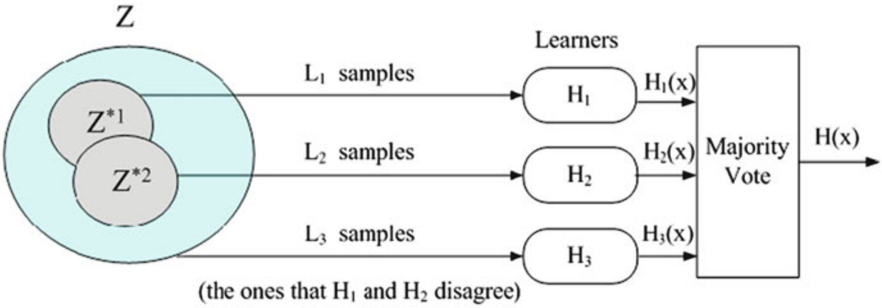

Fig. 2.4 A graphical idea of the first boosting approach proposed in [95]. Notice that each learner can be itself learned by the boosting algorithm in a recursive fashion

As can be seen in Algorithm 3, the training set is randomly divided without replacement into three partitions, Z_1^*, Z_2^*, and Z_3^*. For a given instance, if the first two classifiers (H_1 and H_2) agree on the class label, this is the final decision for that instance. The set of instances on which they disagree defines the partition Z_3^*, which is used to learn H_3. Schapire has shown that this learning method is strong, in the sense defined above. Moreover, the error can be further reduced by using this approach recursively, that is, each learner can itself be obtained by a boosting procedure. Figure 2.4 illustrates the boosting approach.

After this proposal by Schapire, Freund [44] proposed a new boosting algorithm based on, and improving, the ideas presented in [95]. That algorithm improves the accuracy of algorithms for learning binary classifiers, by combining a large number of classifiers, each of which is obtained by running the given learning method on a different set of examples. As in [95], Freund's new proposals also suffered from several drawbacks, namely the need for a very large training set, due to the fact that this set is divided into subsets.

2.2.4 Relationship Between Boosting, Bagging, and Bootstrapping

Figure 2.5 shows the connection between bootstrapping, bagging, and boosting, focusing on what they produce and how they handle the training data. The figure emphasizes the fact that these three techniques are all built upon random sampling, being that bootstrapping and bagging perform sampling with replacement while boosting does not. Bagging and boosting have in common the fact that both provide final classifiers that are majority votes of the individual classifiers.

In [29], a comparison of the effectiveness of randomization, bagging, and boosting for improving the performance of the decision-tree algorithm C4.5 [88] is presented. The experimental results show that for cases with little or no classification

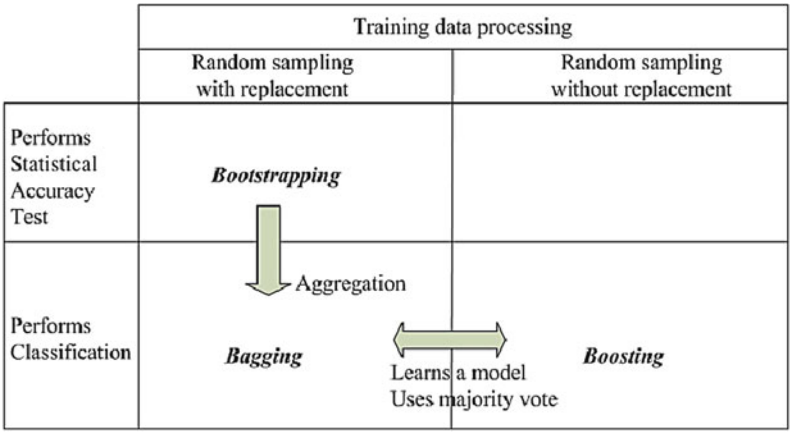

Fig. 2.5 Bootstrapping, bagging, and boosting: what they yield and how they handle the training data

noise, randomization is competitive with (and perhaps slightly superior to) bagging but not as accurate as boosting. For situations with substantial classification noise, bagging is much better than boosting, and sometimes better than randomization.

2.3 The AdaBoost Algorithm

After their initial separate work on boosting algorithms, Freund and Schapire proposed the *adaptive boosting* (AdaBoost) algorithm [47], [48], [50]. The key idea behind AdaBoost is to use *weighted* versions of the *same* training data instead of randomly subsamples thereof. The same training set is repeatedly used and, for this reason, it does not need to be very large, unlike earlier boosting methods.

The AdaBoost algorithm is now a well known and deeply studied method to build ensembles of classifiers with very good performance [58]. The algorithm learns a set of classifiers, using a WL, in order to produce the final classifier of the form (2.1). The weak classifiers[1] are obtained sequentially, using reweighted versions of the training data, with the weights depending on the accuracy of the previous classifiers. The training set is always the same at each iteration, with each training instance weighted according to its (mis)classification by the previous classifiers. This allows the WL at each iteration to focus on patterns that were not well classified by the previous weak classifiers. It is important to chose WLs to obtain the base classifiers, allowing them to learn without decreasing significantly the weight of the previously correctly classified instances. If the base learner is too strong, it may achieve high accuracy, leaving only outliers and noisy instances with significant weight to be

[1] We refer to a classifier learned by a WL as a weak classifier.

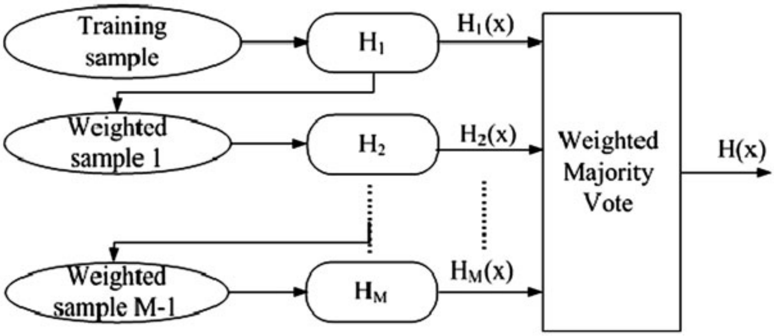

Fig. 2.6 Graphical idea of the adaptive boosting algorithm (adapted from [58]). Each weak learner is trained on a different weighted version of the training data sample. There is no sampling of the training data and the weight of each instance for the following round depends on the performance of the previous learner

Algorithm 4 (Discrete) AdaBoost algorithm for binary classification

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
Output: A classifier $H : \mathscr{X} \to \{-1, +1\}$.

1: Initialize the weights $w_i^{(1)} = 1/N$, $i \in \{1, \ldots, N\}$, and set $m = 1$.
2: **while** $m \leq M$ **do**
3: Run weak learner on Z, using weights $w_i^{(m)}$, yielding classifier $H_m : \mathscr{X} \to \{-1, +1\}$.
4: Compute $\mathrm{err}_m = \sum_{i=1}^{N} w_i^{(m)}\ h\left(-y_i\ H_m(\mathbf{x_i})\right)$, the weighted error of H_m.
5: Compute $\alpha_m = \frac{1}{2} \log\left(\frac{1-\mathrm{err}_m}{\mathrm{err}_m}\right)$. {/* Weight of weak classifier. */}
6: For each sample $i = 1, \ldots, N$, update the weight $v_i^{(m)} = w_i^{(m)} \exp(-\alpha_m\ y_i\ H_m(\mathbf{x_i}))$.
7: Renormalize the weights: compute $S_m = \sum_{j=1}^{N} v_j$ and, for $i = 1, \ldots, N$, $w_i^{(m+1)} = v_i^{(m)} \Big/ S_m$.
8: Increment the iteration counter: $m \leftarrow m + 1$
9: **end while**
10: Final classifier: $H(\mathbf{x}) = \mathrm{sign}\left(\sum_{j=1}^{M} \alpha_j\ H_j(\mathbf{x})\right)$.

learned in the following rounds. Figure 2.6 depicts the structure of AdaBoost, which is described in detail in Algorithm 4.

The function $h : \mathbb{R} \to \{0, 1\}$ used in line 4 of the algorithm is the Heaviside function, defined as $h(x) = 1$, if $x \geq 0$, and $h(x) = 0$, if $x < 0$. Consequently, since both y_i and $H_m(\mathbf{x_i})$ take values in $\{-1, +1\}$, we have that $h\left(-y_i\ H_m(\mathbf{x_i})\right) = 1$, if $y_i \neq H_m(\mathbf{x_i})$, and $h\left(-y_i\ H_m(\mathbf{x_i})\right) = 0$, if $y_i = H_m(\mathbf{x_i})$, and err_m is the weighted error rate of the mth classifier.

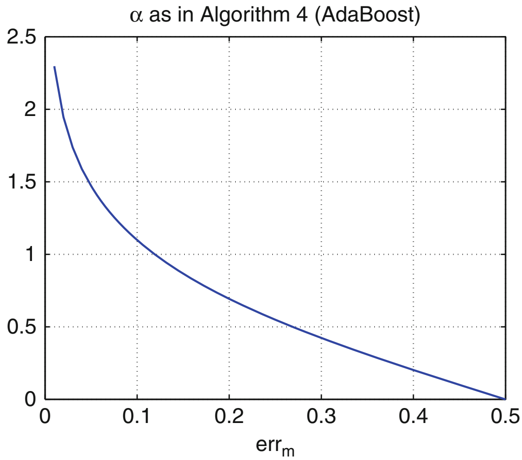

Fig. 2.7 The computation of α_m ($\geq$0) as a function of the weighted classification error for each weak learner. As the error tends to 0.5 (random guessing) the contribution (importance) of the weak learner decreases

Line 3 requires some explanation: what does it mean to run a weak learning algorithm on a weighted version of the training set? It means that the goal of the WL is to obtain a classifier, say H_m, belonging to a given family of classifiers $\mathscr{H}$, that satisfies

$$\sum_{i=1}^{N} w_i \, h\left(-y_i \; H_m(\mathbf{x_i})\right) \leq \frac{1}{2} - \varepsilon, \tag{2.3}$$

for some small positive ε. Notice that only if $w_i = 1/N$, for $i = 1, \ldots, N$ (e.g., at the first iteration of AdaBoost) does the left hand side of (2.3) coincides with the classical error rate on the training set. The existence of such weak classifiers is an important ingredient of boosting, and the interested reader is referred to [82] for more details. Notice that the *weakness* of the classifier is usually controlled by letting $\mathscr{H}$ contain only simple classifiers; for example, when $\mathscr{X} = \mathbb{R}^d$, the family $\mathscr{H}$ may contain only linear rules of the form $H(\mathbf{x}) = \text{sign}(\mathbf{u}^T \mathbf{x} + r)$, where $\mathbf{u} \in \mathbb{R}^d$ and $r \in \mathbb{R}$, which is sometimes known as a perceptron, or rules based on a single component of the input, i.e., of the form $H(\mathbf{x}) = \text{sign}(u \;\, x_j + t)$, where $u \in \{-1, +1\}$ and $t \in \mathbb{R}$, which is called a decision stump.

Notice that the AdaBoost algorithm can actually handle weak classifiers with weighted error rate larger than 1/2; of course, by simply inverting the output of such a classifier, we obtain a classifier with weighted error rate less than 1/2. Such an inversion is automatically performed by AdaBoost, because if $\text{err}_m > 1/2$, the corresponding weight α_m is negative, as is clear from its expression in line 5 of Algorithm 4. Figure 2.7 shows how α_m evolves as a function of the weighted classification error err_m for each weak learner.

Table 2.1 shows the connection between the boosting algorithm (Algorithm 3) and AdaBoost (Algorithm 4). We compare these algorithms in terms of how the training data is processed, the number of classifiers and how the final decision is produced.

Table 2.1 Summary of the main differences between Algorithms 3 (Boosting) and 4 (AdaBoost), regarding how training data is used, the number of samples, the number of classifiers, and the decision mechanism

	Boosting (Algorithm 3)	*AdaBoost (Algorithm 4)*
Data usage	Random sampling, no replacement	Weighting (no sampling)
Number of samples	Three	One
Number of classifiers	Three	Up to M
Decision	Majority vote	Weighted majority vote

A key issue when using the weights for the instances, is that the following learner is provided with more information about the importance of each instance and how the previous learners were (or not) able to deal with that instance. This does not happen in bagging nor boosting.

Notice that a straightforward consequence of the instance weighting scheme is that, after the M AdaBoost rounds, the misclassified patterns assigned with higher weights are "hard" patterns to learn; these patterns are probably outliers. This is a kind of side effect of AdaBoost, which can be used for outlier detection on a given training set.

The AdaBoost algorithm has also been extended for regression tasks. In [3], the prediction error is compared against a threshold to mark it as an error or not and then the AdaBoost version for classification is used. In [31], the probabilities kept by the algorithm are modified based on the magnitude of the error; instances with large error on the previous learners have a higher probability of being chosen to train the following base learner. The median or weighted average is then applied to combine the predictions of the different base learners.

2.3.1 Some Theoretical Properties

We now review several properties of AdaBoost that were shown by Freund and Schapire [50, 100], namely the exponential decay of the TE rate.

The first result shows that the TE of the classifier obtained after M boosting rounds is upper bounded by the product of the normalizing constants of the weights of all the rounds, that is,

$$\mathrm{TE} = \frac{1}{N}\sum_{i=1}^{N} h(-y_i\, H(\mathbf{x}_i)) \leq \prod_{j=1}^{M} S_j, \tag{2.4}$$

where S_j is the normalizing constant used in line 7 at iteration j (the proof of this result can be found in Appendix A).

The second result shows how the TE depends on the weighted error rates of the weak classifiers (denoted err_m). Assume that $\mathrm{err}_m = 1/2 - \gamma_m$, with $\gamma_m > \gamma > 0$, for all $m = 1, \ldots, M$. Then

$$\mathrm{TE} \leq \exp\left(-2\, M\, \gamma^2\right), \tag{2.5}$$

that is, the TE decreases exponentially with M, and does so at a rate that depends on γ (the proof of this result can be found in Appendix A).

The *expected test error* commonly addressed as the generalization error (*GE*) also has an upper bound, as demonstrated in [50]. The GE of the final classifier is upper bounded, with high probability by

$$\mathrm{TE} + \tilde{O}\left(\sqrt{\frac{M\,d}{N}}\right), \tag{2.6}$$

where d is the Vapnik–Chervonenkis (VC) [9, 117] dimension of the set of base classifiers. This result shows that there is a trade-off controlled by the "richness" or "complexity" of the base (weak) classifiers; "stronger" base classifiers allow the TE to be lower, but correspond to a larger CV dimension; on the other hand, simpler classifiers have a lower CV dimension, but require more boosting rounds to decrease the TE.

It has been found empirically that the GE usually does not increase as the size of ensemble becomes very large; moreover, it is often observed that the GE continues to decrease even after the TE has reached zero (see Fig. 2.12). In [99], it is shown that this behavior is related to the distribution of margins of the training examples with respect to the generated voting classification rule. The margin of an example is defined as the difference between the number of correct votes and the maximum number of votes received by any incorrect label.

2.3.2 Different Views of AdaBoost

It has been argued that one explanation for the success of AdaBoost is its ability to increase the *margin* between positive and negative examples [99]. This view provides a connection between margin-based discriminative learning (as in *support vector machines*—SVM [102]) and boosting.

The adaptive boosting techniques can be considered as a greedy optimization method for minimizing the exponential loss function

$$\frac{1}{N}\sum_{i=1}^{N}\exp\left(-y_i\, f(\mathbf{x_i})\right) = \sum_{i=1}^{N}\exp\left(-y_i \sum_{m=1}^{M}\alpha_m H_m(\mathbf{x_i})\right), \tag{2.7}$$

by learning H_m and choosing the most adequate value of α_m at each round. Detailed analysis of boosting and different views of how this learning procedure behaves can be found in [35, 50, 58, 80, 81].

In [21], a unified view of boosting and logistic regression [58] is described. These learning problems are cast in terms of optimization of Bregman distances, due to their high similarity under this framework. For both problems, new sequential and parallel algorithms are proposed and their potential advantages over existing methods are shown. A general proof of convergence for AdaBoost is also presented.

Some connections of AdaBoost with game-theory, linear programming, logistic regression, and estimation of probabilities and outliers are discussed in [97, 100].

An evaluation of bagging and boosting using both NNs and decision trees as learners is carried out in [77]. The experimental results show two important conclusions. The first is that, even though bagging almost always produces a better classifier than any of its individual component classifiers and is relatively impervious to overfitting, it does not generalize any better than a baseline NN ensemble method. The second is that, although boosting is a powerful technique that can usually produce better ensembles than bagging, it is more susceptible to noise and overfitting.

In [42], AdaBoost is evaluated on synthetic and real data using two types of WLs: generative classifiers and radial basis function classifiers. The AdaBoost algorithm with these WLs shows good convergence properties. On benchmark data, boosting of these WLs attains results close to the Real AdaBoost algorithm (with decision trees) and SVM, constituting a low computational complexity competitive choice.

2.4 Variants of AdaBoost

In this section, we review several variants of AdaBoost (although we do not claim to have an exhaustive list), both for binary and multiclass supervised learning problems. Many of these variants have proven to be successful in different types of learning scenarios.

The proposal of the AdaBoost algorithm stimulated a significant amount of research on this type of learning technique, exploiting its theoretical properties and experimental performance. From this research, several variants of AdaBoost have emerged, some targeted at specific problems, such as, for example, face detection and text categorization (TC). Those variants follow the overall structure of AdaBoost (learn a weak classifier, compute the amount of error, update the weights of the training patterns and repeat the process), but introduce changes on several aspects, such as the weight update expression and classifier management.

2.4.1 Detailed Analysis of Some Variants

After AdaBoost was introduced, several modified versions (variants) have been proposed, developed, and compared with AdaBoost. This section addresses some of these variants (shown in the timeline of Fig. 2.8) for supervised learning of binary classifiers.

The following subsections describe in detail some of these variants for binary classification. These variants were selected to be presented in more detail, because they are either the first variants to appear after AdaBoost was proposed or they bring quite different new ideas into the adaptive boosting scheme. These variants have in common the fact that all of them were proved to be successful in real-world machine learning problems.

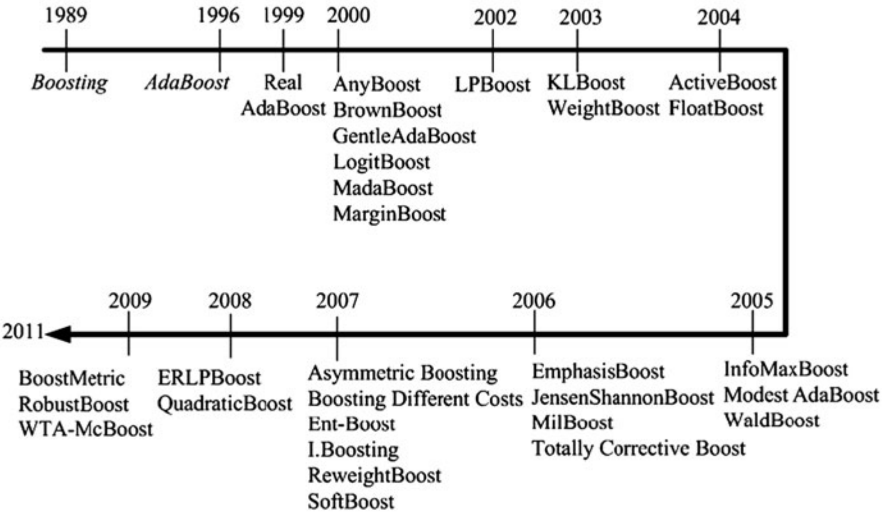

Fig. 2.8 A (possibly incomplete) timeline of AdaBoost variants for supervised learning of binary classifiers, as of 2011

Algorithm 5 Real AdaBoost

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
Output: A classifier $H : \mathscr{X} \to \{-1, +1\}$.

1: Initialize the weights $w_i = 1/N, i \in \{1, \ldots, N\}$.
2: **for** $m = 1$ to M **do**
3: Fit the class probability estimate $p_m(\mathbf{x}) = \hat{P}_w(y = 1|\mathbf{x})$, using w_i.
4: Set $\mathrm{H}_m = \frac{1}{2}\log\left((1 - p_m(\mathbf{x}))p_m(\mathbf{x})\right) \in \mathscr{R}$.
5: Update the weights: $w_i \leftarrow w_i \exp(-y_i H_m(\mathbf{x_i}))$
6: Renormalize to weights.
7: **end for**
8: Final classifier: $H(\mathbf{x}) = \mathrm{sign}\left(\sum_{j=1}^{M} \alpha_j \; H_j(\mathbf{x})\right)$.

2.4.1.1 Real AdaBoost

The first variant we consider is Real AdaBoost [52, 100], where the term *real* refers to the fact that the algorithm uses real-valued "classifiers" (i.e., before thresholding). This real value can be seen as the probability, or degree of confidence, that a given input pattern belongs to a class, considering the current weight distribution for the training set. The Real AdaBoost algorithm is presented as Algorithm 5.

Comparing Real AdaBoost with AdaBoost, we see that the major differences are in lines 3 and 4. In the Real AdaBoost algorithm, these steps consist of computing

Algorithm 6 Logit Boost

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
Output: A classifier $H : \mathscr{X} \rightarrow \{-1, +1\}$.

1: Initialize the weights $w_i = 1/N, i \in \{1, \ldots, N\}$.
2: **for** $m = 1$ to M and while $H_m \neq 0$ **do**
3: Compute the working response $z_i = \dfrac{y_i^* - p(\mathbf{x_i})}{p(\mathbf{x_i})(1 - p(\mathbf{x_i}))}$ and weights $w_i = p(\mathbf{x_i})(1 - p(\mathbf{x_i}))$.
4: Fit $H_m(\mathbf{x})$ by a weighted least-squares of z_i to $\mathbf{x_i}$, with weights w_i.
5: Set $H(\mathbf{x}) = H(\mathbf{x}) + \frac{1}{2} H_m(\mathbf{x})$ and $p(\mathbf{x}) = \dfrac{\exp(H(\mathbf{x}))}{\exp(H(\mathbf{x})) + \exp(-H(\mathbf{x}))}$.
6: **end for**
7: Final classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j \; H_j(\mathbf{x})\right)$.

and using estimates of the probabilities that each training pattern belongs to a class, under the current weight distribution. Standard AdaBoost classifies the input patterns and computes the weighted error rate.

2.4.1.2 Logit Boost

The Logit Boost variant consists of using adaptive Newton steps to fit an additive logistic model [51, 52]. Instead of minimizing the exponential loss, Logit Boost minimizes the logistic loss (negative conditional log-likelihood). Algorithm 6 details the Logit Boost algorithm.

2.4.1.3 Gentle AdaBoost

The Gentle AdaBoost [52] algorithm improves over Real AdaBoost by using Newton steps, providing a more reliable and stable ensemble, since it puts less emphasis on outliers. Instead of fitting a class probability estimate, Gentle AdaBoost (described in Algorithm 7) uses weighted least-squares regression [58] at each iteration. The main difference between Gentle and Real AdaBoost is on the use of the estimates of the weighted class probabilities in order to perform the update. The algorithm is *gentle* because it is considered to be both conservative and more stable as compared to Real AdaBoost. Gentle AdaBoost does not require the computation of log ratios which can be numerically unstable (since they involve quotients, maybe with the denominator approaching zero). Experimental results on benchmark data show that the conservative Gentle AdaBoost has similar performance to Real AdaBoost and Logit Boost, and in many cases outperforms these other two variants.

Algorithm 7 Gentle AdaBoost

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
Output: A classifier $H : \mathscr{X} \to \{-1, +1\}$.

Initialize the weights $w_i = 1/N$, $i \in \{1, \ldots, N\}$.
for $m = 1$ to M **do**
  Train $H_m(\mathbf{x})$ by weighted least-squares of y_i to $\mathbf{x_i}$, with weights w_i.
  Update $H(\mathbf{x}) \leftarrow H(\mathbf{x}) + H_m(\mathbf{x})$.
  Update $w_i \leftarrow w_i \exp(-y_i H_m(\mathbf{x_i}))$ and renormalize to $\sum_i w_i = 1$.
end for
Final classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j \; H_j(\mathbf{x})\right)$.

Algorithm 8 Modest AdaBoost

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
Output: A classifier $H : \mathscr{X} \to \{-1, +1\}$.

1: Initialize the weights $w_i = 1/N$, $i \in \{1, \ldots, N\}$.
2: **for** $m = 1$ to M and while $H_m \neq 0$ **do**
3: Train $H_m(\mathbf{x})$ by weighted least-squares of y_i to $\mathbf{x_i}$, with weights w_i.
4: Compute "inverted" distribution $\overline{w}_i = (1 - w_i)$ and renormalize to $\sum_i \overline{w}_i = 1$.
5: Compute $P_m^{+1} = P_w(y = +1, H_m(\mathbf{x}))$, $\overline{P}_m^{+1} = P_{\overline{w}}(y = +1, H_m(\mathbf{x}))$.
6: Compute $P_m^{-1} = P_w(y = -1, H_m(\mathbf{x}))$, $\overline{P}_m^{-1} = P_{\overline{w}}(y = -1, H_m(\mathbf{x}))$,
7: Set $H_m(\mathbf{x}) = \left(P_m^{+1}\left(1 - P_m^{+1}\right) - P_m^{-1}\left(1 - P_m^{-1}\right)\right)$
8: Update $w_i \leftarrow w_i \exp(-y_i H_m(\mathbf{x_i}))$ and renormalize to $\sum_i w_i = 1$.
9: **end for**
10: Final classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j \; H_j(\mathbf{x})\right)$.

2.4.1.4 Modest AdaBoost

The Modest AdaBoost algorithm [119] is known to have lower GE and higher TE, as compared to Real and Gentle AdaBoost variants. Algorithm 8 shows the details of Modest AdaBoost, which, as compared to the previous variants, uses a different weighting scheme for the correctly and incorrectly classified patterns, using an "inverted" distribution.

The standard distribution w_i assigns high weights to training samples misclassified by earlier steps. On the contrary, $\overline{w}_i$ gives higher weights to samples that are already correctly classified by earlier steps.

Lines 5 and 6 deal with the direct and "inverted" distributions, using the expressions $P_m^{+1} = P_w(y = +1, H_m(\mathbf{x}))$ and $P_m^{-1} = P_w(y = -1, H_m(\mathbf{x}))$; these expressions compute how good is the current weak classifier at predicting class labels. On the other hand, the expressions $\overline{P}_m^{+1} = P_{\overline{w}}(y = +1, H_m(\mathbf{x}))$ and

$\overline{P}_m^{-1} = P_{\overline{w}}(y = -1, H_m(\mathbf{x}))$ estimate how well our current WL $H_m(\mathbf{x})$ is working on the data that has been correctly classified by previous steps.

The update $H_m(\mathbf{x}) = (P_m^{+1}(1-\overline{P}_m^{+1}) - P_m^{-1}(1-\overline{P}_m^{-1}))$ decreases weak classifiers contribution, if it works "too well" on data that has been already correctly classified with high margin. This way, the algorithm is named *Modest* because the classifiers tend to work only in their domain, as defined by w_i.

2.4.1.5 Float Boost

The Float Boost [72,73] variant is composed of the following stages: 1-initialization; 2-forward inclusion; 3-conditional exclusion; 4-output. All of these stages, with the exception of stage 3, are similar to those of AdaBoost and other variants as discussed so far. The novelty here is the conditional exclusion stage, in which the least significant weak classifier is removed from the set of classifiers, subject to the condition that the removal leads to an error below some threshold. The Float Boost algorithm details are described as Algorithm 9.

2.4.1.6 Emphasis Boost

The Emphasis Boost variant uses a *weighted emphasis* (WE) function [53]. Each input pattern is weighted according to a criterion (parameterized by λ), through the WE function, in such a way that the training process focuses on the "critical" patterns (near the classification boundary) or on the quadratic error of each pattern. Algorithm 10 presents the details of Emphasis Boost.

The WE function is defined by

$$w_i = \exp\left(\lambda\left(\sum_{j=1}^{m}\left(\alpha_j H_j(x_i) - y_i\right)^2\right) - (1-\lambda)\left(\sum_{j=1}^{m} H_j(x_i)\right)^2\right) \tag{2.8}$$

and controls where the emphasis is placed. This flexible formulation allows choosing how much to consider the *proximity* terms by means of a weighting parameter ($0 \leq \lambda \leq 1$). This way, we have a *boosting by weighting boundary and erroneous samples* technique. Regarding the value of λ, three particular cases are interesting enough to be considered:

- $\lambda = 0$, focus on the "critical" patterns because only the "proximity" to the boundary is taken into account

$$w_i = \exp\left[-\left(\sum_{j=1}^{m} H_j(\mathbf{x_i})\right)^2\right]. \tag{2.9}$$

Algorithm 9 Float Boost

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
N examples $N = a + b$; a examples have $y_i = +1$ and b examples have $y_i = -1$.
$J(H_M)$, the cost function and the maximum acceptable cost J^*.
Output: A classifier $H : \mathscr{X} \rightarrow \{-1, +1\}$.

1: {1 - Initialization stage.}
2: Initialize the weights $w_i^{(0)} = 1/2a$, for those examples with $y_i = +1$.
3: Initialize the weights $w_i^{(0)} = 1/2b$, for those examples with $y_i = -1$.
4: $J_m^{\text{min}} = J^* \; m = \{1, \ldots, M\text{max}\}$.
5: $M = 0$, $\mathscr{H}_0 = \{\}$.
6: {2 - Forward inclusion stage.}
7: $M \leftarrow M + 1$.
8: Learn $H_m(x)$ and α_M.
9: Update $w_i^{(M)} \leftarrow w_i^{(M-1)} \exp(-y_i \alpha_M H_M(\mathbf{x_i}))$ and renormalize to $\sum_i w_i = 1$.
10: $\mathscr{H}_M = \mathscr{H}_{M-1} \bigcup \{H_M\}$.
11: **if** $J_M^{\text{min}} > J(H_M)$ **then**
12: $\quad J_M^{\text{min}} = J(H_M)$.
13: **end if**
14: {3 - Conditional exclusion stage.}
15: $h' = \text{arg min}_{h \,\in\, \mathscr{H}_M} J(H_M - h)$.
16: **if** $J(H_M - h') < J_{M-1}^{\text{min}}$ **then**
17: $\quad \mathscr{H}_{M-1} = \mathscr{H}_M - h'$.
18: $\quad J_{M-1}^{\text{min}} = J(H_M - h')$
19: $\quad M \leftarrow M - 1$
20: $\quad$ **if** $h' = h'_m$ **then**
21: $\qquad$ Recalculate $w_i^{(j)}$ and h_j for $j = \{m', \ldots, M\}$.
22: $\qquad$ Goto line 15.
23: $\quad$ **else**
24: $\qquad$ **if** $M = M\text{max}$ or $J(\mathscr{H}_M) < J^*$ **then**
25: $\qquad\quad$ Goto line 32.
26: $\qquad$ **else**
27: $\qquad\quad$ Goto line 7.
28: $\qquad$ **end if**
29: $\quad$ **end if**
30: **end if**
31: {4 - Output stage.}
32: Final classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j \; H_j(\mathbf{x})\right)$.

- $\lambda = 0.5$, we get the classical Real AdaBoost emphasis function

$$w_i = \exp\left[\left(\frac{\sum_{j=1}^{m}(\alpha_j H_j(\mathbf{x_i}) - y_i)^2}{2}\right) - \frac{(\sum_{j=1}^{m} H_j(\mathbf{x_i}))^2}{2}\right]. \tag{2.10}$$

Algorithm 10 Emphasis Boost

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
λ, weighting parameter ($0 \leq \lambda \leq 1$).
Output: $H(\mathbf{x})$, a classifier suited for the training set.

Initialize the weights $w_i = 1/N,\ i \in \{1, \ldots, N\}$.
for $m = 1$ to M and while $H_m \neq 0$ **do**
  Fit a classifier $H_m(\mathbf{x})$ to the training data using weights w_i.
  Let $\text{err}_m = \sum_{i=1}^{N} w_i y_i H_m(\mathbf{x_i}) \Big/ \sum_{i=1}^{N} w_i$.
  Compute $\alpha_m = 0.5 \log((1 + \text{err}_m)/(1 - \text{err}_m))$.
  Set $w_i = \exp\left(\lambda \left(\sum_{j=1}^{m} \left(\alpha_j H_j(\mathbf{x_i}) - y_i\right)^2\right) - (1-\lambda)\left(\sum_{j=1}^{m} H_j(\mathbf{x_i})\right)^2\right)$.
  Renormalize to $\sum_i w_i = 1$.
end for
Final classifier: $H(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j \; H_j(\mathbf{x})\right)$.

- $\lambda = 1$, the emphasis function only pays attention to the quadratic error of each pattern

$$w_i = \exp\left[\sum_{j=1}^{m} (\alpha_j H_j(\mathbf{x_i}) - y_i)^2\right]. \tag{2.11}$$

The key issue with this algorithm is the choice of λ.

2.4.1.7 Reweight Boost

In the Reweight Boost variant [92], the weak classifiers are stumps (decision trees with a single node). The main idea is to consider as base classifier for boosting, not only the last weak classifier, but a classifier formed by the last r selected weak classifiers, using a classifier reuse technique. Algorithm 11 presents the details of the Reweight Boost variant.

2.4.1.8 Other Variants

For the sake of both completeness of this chapter and fairness to the many authors of AdaBoost variants, in this subsection we describe further variants for binary classification. We show the name of each variant as well as its main characteristics.

The *KLBoost* [74] variant uses *Kullback–Leibler* (KL) [22] divergence and operates as follows. First, classification is based on the sum of histogram divergences along corresponding global and discriminating linear features. Then,

Algorithm 11 Reweight Boost

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
M, the maximum number of classifiers.
r, the last r selected weak classifiers .
Output: $H(\mathbf{x})$, a classifier suited for the training set.

Initialize the weights $w_i = 1/N, i \in \{1, \ldots, N\}$.
for $m = 1$ to M and while $H_m \neq 0$ **do**
 Fit a classifier $H_m(\mathbf{x})$ to the training data using weights w_i.
 Get combined classifier H_t^r from $H_t, H_{t-1}, \ldots, H_{\max(t-r,1)}$.
 Let $\mathrm{err}_m = \sum_{i=1}^N w_i y_i H_m(\mathbf{x_i}) \Big/ \sum_{i=1}^N w_i$.
 Compute $\alpha_m = 0.5 \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
 Set $w_i \leftarrow w_i \exp\left(-\alpha_m y_i H_t^r(\mathbf{x_i})\right)$.
 Renormalize to $\sum_i w_i = 1$.
end for
Final classifier: $H(\mathbf{x}) = \mathrm{sign}\left(\sum_{j=1}^{M} \alpha_j \ H_j(\mathbf{x})\right)$.

these linear KL features, are iteratively learned by maximizing the projected KL divergence in a boosting manner. Finally, the coefficients to combine the histogram divergences are learned by minimizing the recognition error, once a new feature is added to the classifier. This contrasts with conventional AdaBoost, in which the coefficients are empirically set. Because of these properties, KLBoosting classifier generalizes very well and has been applied to high-dimensional spaces of image data.

One of the experimental drawbacks of AdaBoost is that it can not improve the performance of *Naïve Bayes* (NB) [34,130] classifier as expected. *ActiveBoost* [124] overcomes this difficulty by using active learning to mitigate the negative effect of noisy data and introduce instability into the boosting procedure. Empirical studies on a set of natural domains show that ActiveBoost has clear advantages with respect to the increasing of the classification accuracy of NB when compared against AdaBoost.

The *Jensen–Shannon Boosting* [61] incorporates *Jensen–Shannon* (JS) divergence into AdaBoost. JS divergence is advantageous in that it provides a more appropriate measure of dissimilarity between two classes and it is numerically more stable than other measures such as KL divergence.

Infomax Boosting [76] is an efficient feature pursuit scheme for boosting. It is based on the infomax principle, which seeks optimal feature that achieves maximal mutual information with class labels. Direct feature pursuit with infomax is computationally prohibitive, so an efficient gradient ascent algorithm is proposed, based on the quadratic mutual information, nonparametric density estimation and fast Gauss transform. The feature pursuit process is integrated into a boosting framework as infomax boosting. It is similar to Real AdaBoost, but with the following differences:

- features are general linear projections;
- generates optimal features;
- uses KL divergence to select features;
- finer tuning on the coefficients.

Ent-Boost [68] uses entropy measures. The class entropy information is used to automatically subspace splitting and optimal weak classifier selection. The number of bins is estimated through a discretization process. KL divergence is applied to probability distribution of positive and negative samples, to select the best weak classifier in the weak classifier set.

The *MadaBoost* [30] algorithm consists on a modification of the weighting scheme of AdaBoost. This variant mitigates the problems that AdaBoost suffers from noisy data, improving its performance.

The *SoftBoost* algorithm [126] is a totally corrective algorithm which optimizes the soft margin and tries to produce a linear combination of hypotheses. The term *soft* means that the algorithm does not concentrate too much on outliers and hard to classify examples. It allows them to lie below the margin (with wrong predictions) but penalizes them linearly via slack variables. SoftBoost tries to avoid the problem of overfitting as in AdaBoost when using training data with high degree of noise.

The *linear programming boosting* (*LPBoost*) [27] algorithm maximizes the margin between training samples of different classes; this way, it belongs to the class of margin-maximizing supervised classification algorithms. The boosting task consists of constructing a learning function in the label space that minimizes misclassification error and maximizes the soft margin, formulated as a linear program which can be efficiently solved using column generation techniques, developed for large-scale optimization problems. Unlike gradient boosting algorithms, which may converge in the limit only, LPBoost converges in a finite number of iterations to a global solution, being computationally competitive with AdaBoost. The optimal solutions of LPBoost are very sparse in contrast with gradient-based methods. Empirical findings show that LPBoost converges quickly, often faster than other formulations.

LPBoost performs well on natural data, but there are cases where the number of iterations is linear in the number of training samples instead of logarithmic. By simply adding a relative entropy regularization to the linear objective of LPBoost, we get *entropy-regularized LPBoost* ERLPBoost [127], for which there is a logarithmic iteration bound. As compared to a previous algorithm, named SoftBoost, it has the same iteration bound and better GE. ERLPBoost does not suffer from this problem and has a simpler motivation. A detailed theoretical and experimental comparison between LPBoost and AdaBoost can be found in [69].

The *MarginBoost* algorithm [81] is a variant of the more general algorithm *AnyBoost* [81]. MarginBoost is also a general algorithm. It chooses a combination of classifiers to optimize the sample average of any cost function of the margin. MarginBoost performs gradient descent in function space, at each iteration choosing a base classifier to include in the combination so as to maximally reduce the cost function. As in AdaBoost, the choice of the base classifier corresponds to a minimization problem involving weighted classification error. That is, for a certain weighting of the training data, the base classifier learning algorithm attempts to return a classifier that minimizes the weight of misclassified training examples.

The general class of algorithms named AnyBoost consists of gradient descent algorithms for choosing linear combinations of elements of an inner product space so as to minimize some functional cost. Each component of the linear combination is chosen to maximize a certain inner product. In MarginBoost, this inner product corresponds to the weighted TE of the base classifier.

Brown Boost [45] uses a nonmonotonic weighting function such as examples far from the boundary decrease in weight, trying to achieve a given target error rate. It de-emphasizes outliers when it seems clear that they are too hard to classify correctly, being an adaptive version of Freund's boost-by-majority algorithm [44]. This variant reveals an intriguing connection between boosting and Brownian motion.

The *Weight Boost* algorithm [63] uses input-dependent weighting factors for WLs. It tries to cope with two possible problems of AdaBoost: suffer from overfitting, especially for noisy data; the assumption that the combination weights are fixed constants and therefore does not take particular input patterns into consideration. A learning procedure which is guaranteed to minimize TEs is devised. Empirical studies show that Weight Boost almost always achieves a considerably better classification accuracy than AdaBoost. Furthermore, experiments on data with artificially controlled noise indicate that the Weight Boost algorithm is more robust to noise than AdaBoost.

Asymmetric Boosting [79] is a cost-sensitive extension of boosting. It is derived from decision-theoretic principles, which exploit the statistical interpretation of boosting to determine a principled extension of the boosting loss. Similarly to AdaBoost, the cost-sensitive extension minimizes this loss by gradient descent on the functional space of convex combinations of WLs, and produces large margin detectors. Asymmetric boosting is fully compatible with AdaBoost, in the sense that it becomes the latter when errors are weighted equally.

In [59] we have an asymmetric boosting method, *Boosting with Different Costs*. The motivation is as follows; traditional boosting methods assume the same cost for misclassified instances from different classes, and in this way focus on good performance with respect to overall accuracy. This method is more generic than AdaBoost, and is designed to be more suitable for problems where the major concern is a low false positive (or negative) rate, such as SPAM filtering.

The *Quadratic Boost* [86] algorithm improves AdaBoost with a quadratic combination of base classifiers. It operates by constructing an intermediate learner on the combined linear and quadratic terms. A new method for iterative optimization is proposed; first a classifier is trained by randomizing the labels of the training examples. Subsequently, the input learner is called repeatedly with a systematic update of the labels of the training examples in each round. The quadratic-boosting algorithm converges under the condition that the given base learner minimizes the empirical error. The experimental results show that quadratic boosting compares favorably with AdaBoost on large datasets at the cost of the training time.

The *WaldBoost* [106] variant has near optimal time and error rate trade-off. It integrates the AdaBoost algorithm for measurement selection and ordering and the joint probability density estimation, with the optimal sequential probability ratio test decision strategy. It is suited for computer vision classification problems, in which both the error and time characterize the quality of a decision.

In [75] feature reweighting is integrated into the boosting scheme, which not only weights the samples but also weights the features iteratively; it is named *I.Boosting*. To avoid overfitting problems, a relevance feedback mechanism is applied into the boosting framework. I.Boosting is implemented using *adaptive discriminant analysis* (ADA) as base classifiers. The experimental results show the superior performance of I.Boosting over AdaBoost.

In [46] we have a new boosting algorithm, motivated by the large margins theory for boosting. The experimental results point out that the new algorithm is significantly more robust against label noise than existing boosting algorithms.

The algorithm proposed in [85] combines the base learners with symmetric functions. Among its properties of practical relevance, we have significant resistance against noise, and its efficiency even in an agnostic learning setting. Experimental results show the reliability of the classifiers built.

The *MilBoost* [123] variant uses cost functions from the *multiple instance learning* (MIL) literature combined with the AnyBoost framework. The feature selection criterion of MILBoost is modified to optimize the performance of the Viola–Jones cascade method for object detection (see Section 2.6.1). Experiments with this variant show improvement on the detection rate, as compared to previous approaches. This increased detection rate is a consequence of simultaneously learning the locations and scales of the objects in the training set along with the parameters of the classifier.

The *totally corrective boosting* [128], the weight update of each patterns is analyzed as the minimization of the relative entropy, subject to linear constraints. The algorithm is "totally corrective" in the sense that it takes into account the outputs of all the past WLs; the "corrective" versions only take into account the last WL results. A connection with margin maximization is also shown for totally corrective versions. The experimental results show that the totally corrective versions of AdaBoost attain smaller combinations of WLs than the corrective ones, being competitive with LPBoost (itself a totally corrective boosting algorithm with no regularization, for which there is no iteration bound known). An asymmetric totally corrective boosting approach for real-time object detection is proposed in [125].

In [105], the *BoostMetric* algorithm is proposed. The goal of this algorithm is to learn a semidefinite metric using boosting techniques. It is a generalization of AdaBoost in the sense that the WL is a matrix instead of a classifier, being simple and efficient. It attains better performance than many existing metric learning methods.

A boosting algorithm called *winner-take-all multiple category boosting* (*WTA-McBoost*) was proposed in [135]. On the learning process, the example subcategory labels are modified in order to make better object/nonobject decision. Multiple subcategory boosting classifiers are learned simultaneously with the assumption that the final classification of an example will only be determined by the highest score of all the subcategory classifiers (the winner will take all). The subcategory labels of the examples are dynamically assigned in this process, reducing the risk of having outliers in each subcategory. The WTA-McBoost algorithm uses confidence-rated prediction with asymmetric cost and is thus very efficient to train and test. The algorithm is successfully applied by building a multiview face detector.

The standard boosting procedure is extended to train a two-layer classifier dedicated to handwritten character recognition [43]. This learning scheme relies on a hidden layer and an output layer to obtain a final classification decision. The classical AdaBoost procedure is extended to train a multilayered structure by propagating the error through the output layer. This extension allows for the selection of optimal WLs by minimizing a weighted error, in both the output layer and the hidden layer.

2.4.2 Multiclass Variants

Since the first binary classification versions of AdaBoost, several generalizations of this algorithm to the multiclass case have been proposed. As a result of this direction of research, several multiclass AdaBoost variants have been proposed, as depicted in the timeline shown in Fig. 2.9. Similarly to what we did in Section 2.4.1.8, for each variant we will point out its main features as well as the connections among them. Many variants address multiclass classification problems as the concatenation of binary problems (a multiclass classification problem can be reformulated as a set of binary problems), while other (more recent) variants apply and combine multiclass classifiers directly.

AdaBoost.M1 and *AdaBoost.M2* [50] are multiclass extensions of (Discrete) AdaBoost (Algorithm 4). They differ between themselves in the way they treat each class. In the M1 variant, the weight of a base classifier is a function of the error rate. In M2, the sampling weights are increased for instances for which the pseudo-loss exceeds 0.5. The *AdaBoost.M1W* [39] algorithm changes AdaBoost.M1 as follows. In AdaBoost.M1, the weight of a base classifier is a function of the error rate. For AdaBoost.M1W this function is such that it gets positive, if the error rate is less than the error rate of random guessing. *BoostMA* [40] is also a simple modification of AdaBoost.M2 with the advantage that the base classifier minimizes the confidence-rated error, whereas for AdaBoost.M2 the base classifier should minimize the pseudo-loss. This makes BoostMA more easily applicable to already existing base classifiers; it also tends to converge faster than AdaBoost.M2.

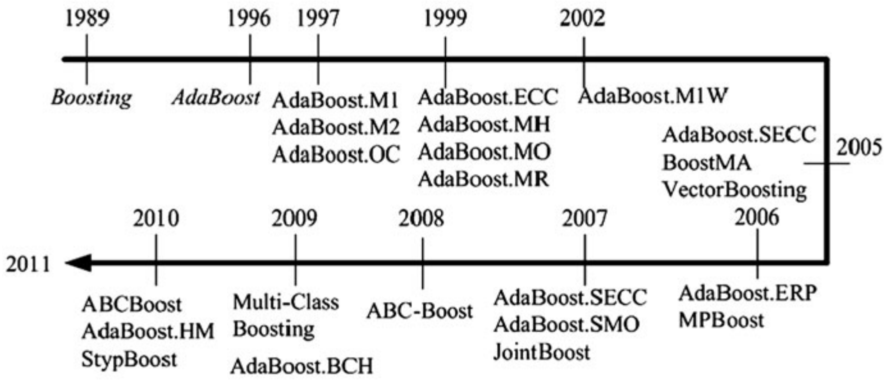

Fig. 2.9 A (possibly incomplete) timeline of AdaBoost variants for supervised learning, on multiclass problems, as of 2011

AdaBoost.MH [100] is a multiclass and *multilabel*[2] version of AdaBoost based on Hamming loss [100]. AdaBoost.MH generalizes AdaBoost, being tailored for multilabel text categorization tasks with decision stumps as WLs. *MPBoost* [41] further improves AdaBoost.MH augmenting its efficiency by performing a multiple pivot selection at each boosting iteration. Both these algorithms use binary features.

The *AdaBoost.MO* algorithm [100] performs a stage-wise functional gradient descent procedure on a given cost function. *AdaBoost.MR* [100] is a multiclass, multilabel version of AdaBoost based on ranking loss. *AdaBoost.OC* [96], where OC stands for output codes and *AdaBoost.ECC* [56], where ECC stands for error-correcting codes are similar algorithms; AdaBoost.OC is a shrinkage version of AdaBoost.ECC, which performs a stage-wise functional gradient descent procedure on an exponential loss cost function.

The *Vector Boosting* [60] algorithm is an extension of the Real AdaBoost in which both its WL and its final output are vectors rather than scalars. The idea of Vector Boosting comes from the *multiclass multilabel* (MCML) version of the Real AdaBoost, which assigns a set of labels for each sample and decomposes the original problem into k orthogonal binary ones. The major problem of this algorithm is that for each binary classification problem, a sample is regarded as either positive or negative. However, in many complicated cases, it is not tenable since some samples are neither positive nor negative for certain binary classification problems of which they are independent, which makes the MCML version of Real AdaBoost inapplicable.

AdaBoost.ERP [70] is AdaBoost.ECC with repartitioning. This algorithm improves two well-known issues of the quality of the ensemble learned by AdaBoost.ECC: the performance of the base learner; the error-correcting ability of the coding matrix. A coding matrix with strong error-correcting ability may not be

[2] A given instance can be classified into one or more classes.

overall optimal if the binary problems are too hard for the base learner. A trade-off between error-correcting and base learning is then proposed. The coding matrix is modified according to the learning ability of the base learner.

In [109], shrinkage is applied as regularization in AdaBoost.MO and AdaBoost.ECC and leads two new algorithms named *AdaBoost.SMO* and *AdaBoost.SECC* (the shrinkage versions of MO and ECC, respectively). A similar proposal for AdaBoost.SECC can also be found in [110].

In [138], we have a new algorithm named *Multiclass AdaBoost* that directly extends the AdaBoost algorithm to the multiclass case without reducing it to multiple two-class problems. The algorithm is equivalent to a forward stage-wise additive modeling algorithm that minimizes a novel exponential loss for multiclass classification. The algorithm is highly competitive in terms of misclassification error rate.

The *AdaBoost.BCH* algorithm [65] is a multiclass boosting algorithm which solves a C class problem by using $C - 1$ binary classifiers arranged by a hierarchy that is learned on the classes based on their closeness. AdaBoost is then applied to each binary classifier. AdaBoost.BCH requires less computation than AdaBoost.MH, with better or comparable generalization.

In [71], the concept of *adaptive base class boost* (*ABC-Boost*) for multiclass classification is addressed deriving *ABC-MART*, a concrete implementation of ABC-Boost. For binary classification, ABC-MART recovers MART and for multiclass classification, ABC-MART considerably improves MART, as evaluated on several public datasets.

AdaBoost.HM was proposed in 2010 [64]. It is based on *hypothesis margin* and directly combines multiclass weak classifiers, instead of learning binary WLs. The hypothesis margin maximizes the output about the positive class and minimizes the maximal outputs about the negative classes. Upper bounds on the TE of AdaBoost.HM are derived and compared against AdaBoost.M1 upper bounds. The WLs are feedforward NNs. AdaBoost.HM yields higher classification accuracies than both the AdaBoost.M1 and the AdaBoost.MH algorithms, being computationally efficient in training.

Recently, a totally corrective multiclass boosting was proposed [57]. After an analysis of some methods that extend two-class boosting to multiclass, a column-generation based totally corrective framework for multiclass boosting learning is derived, using the Lagrange dual problems. Experimental results show that the new algorithms have comparable generalization capability but converge much faster than their counterparts.

StypBoost [129] is a bilinear boosting algorithm, which extends the multiclass boosting framework of JointBoost to optimize a bilinear objective function. This allows style parameters to be introduced to aid classification, where style is any factor which the classes vary with systematically, modeled by a vector quantity. The algorithm allows learning with different styles. It is applied successfully to two object class segmentation tasks: road surface segmentation and general scene parsing.

JointBoost [107, 113] is a method where boosted one-versus-all classifiers are trained jointly and are forced to share features. It has been demonstrated to lead both to higher accuracy and smaller classification time, compared to using one-versus-all classifiers that were trained independently and without sharing features.

2.4.2.1 Analysis of Multiclass Boosting Algorithms

In [109] the AdaBoost.MO, AdaBoost.OC, and AdaBoost.ECC algorithms are studied. It is shown that MO and ECC perform stage-wise functional gradient descent on a cost function defined over margin values, and that OC is a shrinkage version of ECC. The AdaBoost.SMO and AdaBoost.SECC are the shrinkage versions of MO and ECC, respectively.

A unifying framework for studying the solution of multiclass categorization problems, by reducing them to multiple binary problems that are then solved using a margin-based binary learning algorithm is proposed in [2]. The proposed framework unifies some of the most popular approaches in which each class is compared against all others, or in which all pairs of classes are compared to each other, or in which output codes with error-correcting properties are used. A general method for combining the classifiers generated on the binary problems is proposed. A generic empirical multiclass loss bound given the empirical loss of the individual binary-learning algorithms is proven. The scheme and the corresponding bounds apply to many popular classification learning algorithms including SVM, AdaBoost, regression, logistic regression, and decision-tree algorithms. A multiclass GE analysis for general output codes with AdaBoost is provided.

The ability of boosting to achieve drastic improvements compared to the individual WLs has been noticed by several researchers. For two-class problems it has been observed that AdaBoost, is quite unaffected by overfitting. However, for the case of noisy data, it is also known that AdaBoost can be improved considerably by introducing some regularization technique. In speech-related problems one often considers multiclass problems and boosting formulations have been used successfully to solve them. Under this context, [90] reviews and extends the existing multiclass boosting algorithms to derive new boosting algorithms, which are more robust against outliers and noise in the data; these algorithms are also able to exploit prior knowledge about relationships between the classes.

In [110], a new interpretation of AdaBoost.ECC and AdaBoost.OC is presented. AdaBoost.ECC performs stage-wise functional gradient descent on a cost function, defined in the domain of margin values; AdaBoost.OC is a shrinkage version of AdaBoost.ECC. AdaBoostBCH has slower training and higher generalization ability as compared to AdaBoost.ECC [65].

2.5 Boosting for Semi-Supervised Learning

Semi-supervised learning (SSL) [15] has attracted considerable research efforts in the last few years. In many learning problems, we have a large amount of data available, but only a subset of it is labeled. In this section, we review several boosting algorithms for SSL, shown in the timeline of Fig. 2.10.

2.5.1 MixtBoost

The MixtBoost algorithm [55] was the first variant of AdaBoost to for SSL; the authors address the question: *can boosting be adapted for SSL learning?* The base classifiers are mixture models; thus, MixtBoost can be seen as boosting of mixture models.

The main ingredients of AdaBoost are the *loss* and the *margin*. The simplest way to generalize to SSL is to define these quantities for unlabeled data. This generalization to unlabeled data should not affect the labeled examples and should penalize inconsistencies between the classifier output and the available information. A loss definition is proposed for unlabeled data, from which margins are defined. The missing labels are interpreted as the absence of class information with the key idea that the pattern belongs to a class, but the class is unknown.

2.5.2 SSMarginBoost

The SSMarginBoost algorithm is an extension of MarginBoost to SSL [26] that explores the *clustering* assumption of SSL [15] and the large margin criterion.

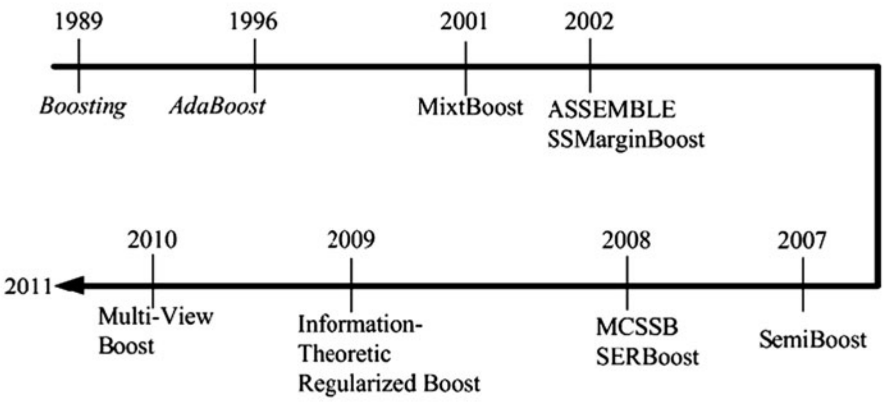

Fig. 2.10 A (possibly incomplete) timeline of AdaBoost variants for semi-supervised learning on binary and multiclass problems, as of 2011

The margin definition is extended to unlabeled data and the gradient descent algorithm for optimizing the resulting margin cost function is derived. SSMarginBoost can be applied with any base classifier able to handle unlabeled data, by means of mixture models trained with an *expectation-maximization* (EM) algorithm [28, 34].

2.5.3 ASSEMBLE

In [6], an adaptive semi-supervised ensemble method, named ASSEMBLE, was proposed. The method constructs ensembles based on both labeled and unlabeled data, by alternating between assigning "pseudo-classes" to the unlabeled data using the existing ensemble and constructing the next base classifier using both the labeled and pseudo-labeled data. This algorithm corresponds to maximizing the classification margin in hypothesis space as measured on both the labeled and unlabeled data. Unlike alternative approaches, ASSEMBLE does not require a SSL method for the base classifier. It can be used in conjunction with any cost-sensitive classification algorithm for both two-class and multiclass problems. As in SSMarginBoost, ASSEMBLE adopts the MarginBoost notation and strategy adapted to the margin measured on both the labeled and unlabeled data; the key difference is that ASSEMBLE assigns "pseudo-classes" to the unlabeled data.

Moreover, ASSEMBLE using decision trees won the Neural Information Processing Systems (NIPS) 2001 Unlabeled Data Competition. It achieves good results on several benchmark datasets using both decision trees and neural networks.

2.5.4 SemiBoost

The SemiBoost [78] algorithm was proposed as a boosting framework aiming at improving the classification accuracy of any given supervised learning algorithm by using the available unlabeled examples. The main advantages of SemiBoost over previous approaches are: (1) performance improvement of any supervised learning algorithm, using unlabeled data; (2) efficient computation by the iterative boosting algorithm and (3) exploiting both the SSL *manifold* and *cluster* assumptions [15].

An empirical study on 16 different datasets and text categorization demonstrates that SemiBoost improves the performance of several commonly used supervised learning algorithms, by using a large number of unlabeled examples.

2.5.5 MultiClass Semi-Supervised Boosting

The *multiclass semi-supervised boosting* (MCSSB) algorithm was proposed in [115]. Compared to the existing semi-supervised boosting methods, MCSSB has the advantage to exploit both classification confidence and similarities among examples, when deciding the pseudo-labels for unlabeled examples. This way, it overcomes the shortcoming of the multiclass approach *one-against-the-rest* applied on binary classifiers. Empirical evidence on several datasets shows that the MCSSB algorithm performs better than the previous algorithms for SSL.

2.5.6 SERBoost

The problem of bad scaling behavior of many SSL methods on large scale vision problems is addressed in [93]. Based on the *expectation regularization* (ER) principle, the SERBoost SSL boosting algorithm is proposed. It can be applied to large scale vision problems and its complexity is dominated by the base learners. The algorithm provides a margin regularizer for the boosting cost function and shows a principled way of utilizing prior knowledge. As compared to supervised and semi-supervised methods, SERBoost shows improvement both in terms of classification accuracy and computational speed.

2.5.7 Information Theoretic Regularization Boosting

An SSL boosting algorithm that incrementally builds linear combinations of weak classifiers through generic functional gradient descent, using both labeled and unlabeled training data, was proposed in [136]. The approach is based on extending the information regularization framework to boosting, bearing loss functions that combine log loss on labeled data with the information-theoretic measures to encode unlabeled data. Even though the information-theoretic regularization terms make the optimization nonconvex, a simple sequential gradient descent optimization algorithm is applied. This approach attains good results on synthetic, benchmark, and real world tasks as compared to supervised and semi-supervised boosting algorithms.

2.5.8 Multiview Boosting

A multiview boosting algorithm was proposed in [94], that, unlike other approaches, specifically encodes the uncertainties over the unlabeled samples in terms of given priors. Instead of ignoring the unlabeled samples during the training phase of each view, it uses the different views to provide an aggregated prior which is then used as a regularization term inside a semi-supervised boosting method, for multiclass problems. The algorithm uses priors as a regularization component over the unlabeled data. Since the priors may contain a significant amount of noise, a new loss function for the unlabeled regularization is introduced, being robust to noisy priors.

2.5.9 Extensions on Semi-Supervised Boosting Algorithms

Different strategies have been applied to extend boosting algorithms to SSL problems. Typically, these strategies do not take into account the local smoothness constraints among data into account during ensemble learning. A local smoothness regularizer to semi-supervised boosting algorithms based on the universal optimization framework of margin cost functionals was proposed in [17]. This regularizer is applicable to existing SSL boosting algorithms to improve their generalization and speed up their training.

In [18], the problem of using all the three SSL assumptions (*smoothness*, *cluster*, and *manifold*) during boosting is addressed. A novel cost functional consisting of the margin cost on labeled data and the regularization penalty based on unlabeled data is proposed. Thus, minimizing the proposed cost functional with a greedy stage-wise optimization procedure leads to a generic boosting framework for SSL.

A local smoothness regularizer for SSL boosting algorithm, based on the universal optimization framework of margin cost functionals, was proposed in [17]. This regularizer is applicable to existing SSL boosting algorithms to improve their generalization and speed up their training.

2.6 Experimental Evaluation

In this section, we discuss the application of AdaBoost and its variants on a wide variety of problems. We compare the boosting algorithms with other machine learning techniques, exploiting the theoretical properties of AdaBoost.

2.6.1 Successful Applications

Besides its nice theoretical properties, the AdaBoost algorithm and its variants have been found to work very well on problems from different domains. Empirical evidence from many researchers has shown the adequacy of boosting algorithms for real-world problems. This section outlines some successful applications of AdaBoost and its variants for binary and multiclass problems. There are many papers which evaluate AdaBoost and its variants for many types of problems; see for instance [5,29,33,62,77,89,104]. It has been shown empirically that AdaBoost with decision trees has excellent performance, being considered the best "off-the-shelf" classification algorithm [5, 58].

The first boosting algorithms were tested on a *optical character recognition* (OCR) problem of optical handwritten digits, with a set of 118,000 instances in boosting multilayer perceptrons [32].

Table 2.2 Summary of the use of boosting algorithms for face detection (adapted from [72])

Face Detector	*AdaBoost Variant*	*Weak Learner*
Viola–Jones [120]	Discrete AdaBoost	Stubs
Float Boost	Float Boost [72, 73]	1D Histograms
KLBoost	KLBoost [74]	1D Histograms
Schneiderman	Real AdaBoost [100]	One group of nD Histograms

In [122], a *pedestrian detection* system that integrates image intensity information with motion information is proposed. A detection-style algorithm scans a detector over two consecutive frames of a video sequence. The detector is trained using AdaBoost to take advantage of both motion and appearance information to detect a walking person. The detector combines two sources of information.

The breast cancer detection problem is addressed in [111]. A data preprocessing, feature selection and Modest AdaBoost algorithm, are applied to the breast cancer survival databases in Thailand. For this task, Modest AdaBoost outperforms Real and Gentle AdaBoost variants.

In [101], boosting is applied to *multiclass text categorization* tasks. The approach named BoosTexter has comparable results to other text-categorization algorithms, on a variety of tasks. The BoosTexter system is also applied to *speech categorization* to call-type identification from unconstrained spoken customer responses.

For *face detection*, boosting algorithms have been the most effective of all those developed so far, achieving the best results. They produce classifiers with about the same error rate than neural networks, with faster training [72]. Table 2.2 summarizes the use of boosting algorithms for face detection (a *stub* is a decision tree with a single decision node).

In [25] a fast and efficient face detection method has been devised, which relies on the AdaBoost algorithm and a set of Haar wavelet-like features. The face detection problem was been addressed also with *Asymmetric Boosting* [79], where it is shown to outperform a number of previous heuristic proposals for cost-sensitive boosting. For an updated literature on face detection and the use of boosting and other machine learning techniques, see [133].

A method for selecting edge-type features for *iris recognition* is proposed in [16]. The AdaBoost algorithm is used to select a filter bank from a pile of filter candidates. The decisions of the weak classifiers associated with the filter bank are linearly combined to form a strong classifier. The boosting algorithm can effectively improve the recognition accuracy at the cost of a slight increase on the computation time.

A new approach, proposing two *particle swarm optimization* (PSO) methods within AdaBoost for *object detection*, for constructing weak classifiers in AdaBoost is proposed in [83]. The experiments show that using PSO for selecting features and evolving associated weak classifiers in AdaBoost is more effective than for selecting features only for this problem.

In [131] a *face recognition* method using AdaBoosted low dimensional and discriminant Gabor features is proposed. AdaBoost is successfully applied to face recognition by introducing the intra-face and extra-face difference space in the Gabor feature space. By using the proposed method, only hundreds of Gabor features are selected. Experiments shown that these hundreds of Gabor features are enough to achieve good performance comparable to that of methods using the complete set of Gabor features.

A *feature selection approach* based on Gabor wavelets and AdaBoost is proposed in [137]. The features are first extracted by a Gabor wavelet transform. For each individual, a small set of significant features are selected by the AdaBoost algorithm from the pool of the Gabor wavelet features. In the feature selection process, each feature is the basis for a weak classifier. In each round of AdaBoost learning, the feature with the lowest error of weak classifiers is selected. The results from the experiments have shown that the approach successfully selects meaningful and explainable features for face verification. The experiments suggest that the feature selection algorithm for face verification selects the features corresponding to the unique characteristics rather than common characteristics, and a large example size statistically shows the benefits of AdaBoost feature selection.

The problem of *classifying music by genre* by partitioning songs into smaller pieces and classifying each one separately is addressed in [7]. The choice of features together with an AdaBoost.MH classifier proved to be the most effective method for genre classification at the MIREX 2005 international contest in music information extraction, and the second-best method for recognizing artists.

In [84], 2D cascaded AdaBoost, a novel classifier designing framework, is presented and applied to the *eye localization* problem. There are two cascade classifiers in two directions: the first one is a cascade designed by bootstrapping the positive samples; the second one, as the component classifiers of the first one, is cascaded by bootstrapping the negative samples. The proposed structure is applied to eye localization and evaluated on four public face databases, and extensive experimental results verified the effectiveness, efficiency, and robustness of the proposed method.

AdaBoost can also improve the performance of a strong learning algorithm as proposed in [103]: a NN based *online character recognition system*. AdaBoost can be used to learn automatically a great variety of writing styles even when the amount of training data for each style varies a lot. The system achieves about 1.4% error on a handwritten digit database of more than 200 writers.

In [87] the use of boosting and SVM is explored for the *segmentation of white-matter lesions in the MR scans of human brain*. Simple features are generated from proton density scans. Radial basis function-based AdaBoost technique and SVM are employed for this task. The classifiers are trained on severe, moderate, and mild cases. The results indicate that the proposed approach can handle MR field inhomogeneities quite well.

A *visual object detection* framework that is capable of processing images extremely rapidly while achieving high detection rates is proposed in [121]. The learning algorithm, based on AdaBoost, selects a small number of critical visual features and yields extremely efficient classifiers. The method combines classifiers in a cascade allowing background regions of the image to be quickly discarded while spending more computation on promising object-like regions. A set of experiments in the domain of *face detection* is presented.

A framework for *classifying face images* using AdaBoost and domain-partitioning based classifiers is addressed in [118]. The most interesting aspect of this framework is its ability to build classification systems with high accuracy in dynamical environments, which achieve, at the same time, high processing and training speed. This framework is applied to the specific problem of gender classification using different features, on standard face databases.

In [112] an approach for *image retrieval* using a very large number of highly selective features and efficient online learning is proposed. This approach is predicated on the assumption that each image is generated by a sparse set of visual "causes" and that images which are visually similar share causes between them. A mechanism for computing a very large number of highly selective features which capture some aspects of this causal structure (with over 45,000 highly selective features) is proposed. At query time a user selects a few example images, and boosting is used to learn a classification function in this feature space. The boosting procedure learns a simple classifier which only relies on 20 of the features. As a result, a very large database of images can be scanned rapidly.

The *boosting-based multimodal speaker detection* (BMSD) algorithm is proposed in [132]. It performs *speaker detection, identifying the active speaker* in a video, which can be very helpful for remote participants to understand the dynamics of the meeting. This algorithm fuses audio and visual information at feature level by using boosting to select features from a combined pool of both audio and visual features simultaneously. It achieves a very accurate speaker detector with extremely high efficiency.

2.6.1.1 Online Boosting

In the recent years, some attention has been given to *online boosting*, in which the training examples become available one at a time [4, 14, 23, 24].

A new family of topic-ranking algorithms for multilabeled documents is proposed in [23, 24]. The algorithms are simple to implement being both time and memory efficient. Experiments with the proposed family of topic-ranking algorithms on standard corpora, show that these algorithms attain adequate results, outperforming other topic-ranking adaptations of well-known classifiers.

The problem of *online adaptation of binary classifiers for tracking* is addressed in [54]. Online learning allows for simple classifiers since only the current view of the object from its surrounding background needs to be discriminated. However, online adaptation has one key problem: each update of the tracker may introduce an error which, finally, can lead to tracking failure (drifting). A novel online semi-supervised boosting method which significantly alleviates the drifting problem in tracking applications was proposed in [54]. This allows to limit the drifting problem while still staying adaptive to appearance changes. The main idea is to formulate the update process in a semi-supervised fashion as combined decision of a given prior and an online classifier without any parameter tuning.

A boosting framework that can be used to derive online boosting algorithms for various cost functions was proposed in [4]. Within this framework, online boosting algorithms for logistic regression, least squares regression, and multiple instance learning are derived.

In [14] a *real-time vision-based vehicle detection* system employing an online boosting algorithm is proposed. It is an online AdaBoost approach for a cascade of strong classifiers instead of a single strong classifier. The idea is to develop a cascade of strong classifiers for vehicle detection that is capable of being online trained in response to changing traffic environments. The proposed online boosting method can improve system adaptability and accuracy to deal with novel types of vehicles and unfamiliar environments.

TransientBoost [108] is an online learning algorithm, which is highly adaptive but still robust. It uses an internal multiclass representation and models reliable and unreliable data in separate classes. Unreliable data is considered transient, and thus highly adaptive learning parameters are applied to adapt to fast changes in the scene while errors fade out fast. In contrast, the reliable data is preserved completely and not harmed by wrong updates. The algorithm is applied successfully on the tasks of object detection and object tracking.

2.6.2 Comparison with Other Machine Learning Techniques

In this section, we show some detailed experimental results comparing AdaBoost variants, on well-known public domain datasets. We also describe some public domain software packages with code for AdaBoost and its variants.

Table 2.3 briefly describes the datasets used in the experiments, shown by increasing dimensionality. These datasets have several types of data and represent many different learning problems and are available from the UCI Repository [8].[3] We also have some datasets from bioinformatics (micro-array and gene expression data)[4] as well as datasets of the NIPS2003 FS Challenge[5] namely, Arcene, Madelon, Gisette, and Dexter.

The Leptograpsus Crabs dataset is from Ripley's book [91], and is publicly available at http://www.stats.ox.ac.uk/pub/PRNN/. The Crabs dataset is considered as a two-class problem for male/female detection. The Phoneme[6] dataset holds log-periodograms to represent speech phonemes as used in [58].

---

[3] http://archive.ics.uci.edu/ml/datasets.html

[4] http://www.gems-system.org/

[5] http://www.nipsfsc.ecs.soton.ac.uk

[6] http://orange.biolab.si/datasets/phoneme.htm

Table 2.3 Datasets with binary problems used in the experiments: P and N are the number of features and patterns, respectively. The datasets are shown by increasing dimensionality

Dataset	*P*	*N*	*Type of data / Classification problem*
Crabs	5	200	Classify crabs by gender
Phoneme	5	5404	Speech phoneme classification
Abalone	8	4177	Predict the age of abalone from physical measurements
Pima	8	768	The Pima Indians diabetes detection
Contraceptive	9	1473	Predict the current contraceptive method choice
Hepatitis	19	155	Detect if patients lived or died from hepatitis
WBCD	30	569	Wisconsin breast cancer diagnostic database
Ionosphere	34	351	Radar data–signals returned from the ionosphere
SpamBase	54	4601	Sparse BoW data/classify email as SPAM or not
Madelon	500	4400	Float data/artificial dataset, highly nonlinear and difficult
Colon	2000	62	Colon cancer detection
Gisette	5000	13500	Dense integer/distinguish handwritten digits "4" and "9"
DLBCL	5470	77	Dense integer/Lymphoma detection from medical analysis
Leukemia	7129	72	Cancer detection from medical analysis
Example 1	9947	2600	Sparse BoW (subset of Reuters)/text classification
Arcene	10000	900	Dense integer/detect cancer versus normal patterns
Prostate Tumor	10509	102	Cancer detection from medical analysis
Dexter	20000	2600	Same data as Example 1 with 10053 distractor features

The Abalone and Pima Indians are well-known datasets from the UCI Repository; their tasks is to predict the age of abalones from their shell measurements and to predict the presence of Diabetes in the Pima Indians population, respectively.

The Contraceptive dataset has the task to predict the current contraceptive method choice, for married women who were either not pregnant or do not know if they were at the time of interview. It is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey.

The task of Hepatitis dataset is to classify if patients lived or died from hepatitis, given a set of medical analysis. The WBCD dataset is the well-known Wisconsin breast cancer database. The Ionosphere dataset is a binary classification problem on radar data; "good" radar returns are those showing evidence of some type of structure in the ionosphere and "bad" returns are those that do not; their signals pass through the ionosphere. The SpamBase dataset has sparse bag-of-words floating-point data; the task is to classify email messages as SPAM or nonSPAM. We have considered only the first 54 features which constitute a *bag-of-words* representation. The Madelon dataset is an artificial problem of the NIPS2003 FS challenge[7]; it is a difficult problem, because it is multivariate and highly nonlinear. The Colon, DLBCL, Leukemia, and Prostate Tumor datasets deal with the problem of cancer detection from microarray data.

[7] http://clopinet.com/isabelle/Projects/NIPS2003/#challenge

In the case of Example 1,[8] each pattern is a 9947-dimensional BoW vector. The Dexter dataset has the same data as Example 1 (with different train, test, and validation partitions) with 10053 additional distractor features, at random locations; it was created for the NIPS 2003 FS challenge. For both datasets, the task is learn to classify Reuters articles as being about "corporate acquisitions" or not.

The Arcene and Gisette datasets also belong to the NIPS2003 FS challenge, and their tasks are: to distinguish cancer versus normal patterns from mass-spectrometric data; to separate the highly confusable handwritten digits "4" and "9."

2.6.2.1 Software Packages for Boosting Algorithms

There are several software packages, freely available online that include implementations of boosting algorithms:

- The GML AdaBoost Matlab Toolbox[9] provides implementations of Real, Gentle, and Modest AdaBoost.
- The ENTOOL Matlab Toolbox http://www.j-wichard.de/entool/ which has many machine learning techniques and includes Real, Gentle, and Modest AdaBoost from the GML Toolbox.
- A Java implementation is available at http://jboost.sourceforge.net/, including AdaBoost, LogitBoost, RobustBoost, and BoosTexter.
- The well-known WEKA machine learning package includes AdaBoost.M1 and MultiBoost classifiers, and it is available at http://www.cs.waikato.ac.nz/~ml/weka/.
- A C++ implementation of the MPBoost algorithm, is available at this internet address http://www.esuli.it/mpboost.
- An efficient C++ implementation of various boosting algorithms can be found in http://www.stat.purdue.edu/~vishy/.
- An open-source implementation of BoosTexter[10] (see Section 2.6.1) can be found at http://code.google.com/p/icsiboost/.
- A BoostMetric implementation as well as other boosting algorithms are available at http://code.google.com/p/boosting/.
- In http://cseweb.ucsd.edu/~yfreund/adaboost/index.html, we have a Java Applet that shows how AdaBoost behaves during training.
- The *generalized boosted regression models* (GBM) implements extensions to AdaBoost and gradient boosting machine. It includes regression methods for least squares, absolute loss, quantile regression, logistic, Poisson, Cox proportional hazards partial likelihood, and AdaBoost exponential loss. It is available at http://cran.r-project.org/web/packages/gbm/index.html.

---

[8] http://svmlight.joachims.org/

[9] http://graphics.cs.msu.ru/ru/science/research/machinelearning/adaboosttoolbox

[10] http://www.cs.princeton.edu/~schapire/boostexter.html

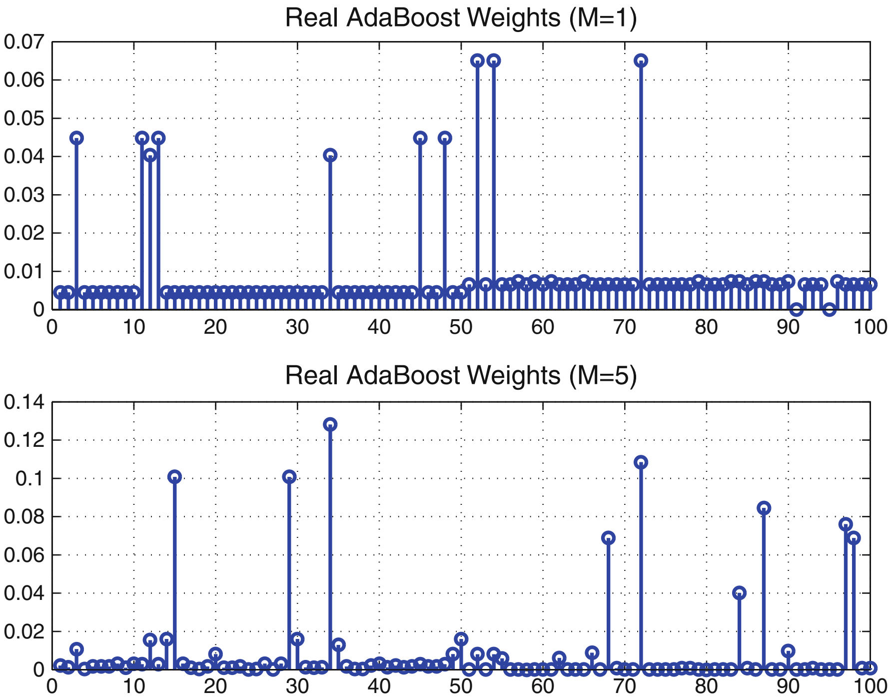

Fig. 2.11 The weights assigned to each pattern, after training Real AdaBoost with $M \in \{1, 5\}$ rounds (learners), on the Ionosphere dataset

The PRTools toolbox [36] available at http://www.prtools.org/prtools.html has many machine learning techniques, but it lacks implementations on boosting algorithms.[11]

2.6.2.2 Analysis of Training and Test Error

Figure 2.11 shows the weights of each pattern, after training Real AdaBoost with $M \in \{1, 5\}$ rounds, on the Ionosphere dataset using 100 training patterns. Notice that at the beginning of the first round, the weights have an uniform distribution with $1/N$.

On the first few iterations, the weight of many patterns is changed in such a way that we get a distribution which is quite different from the uniform.

[11] As of version PRTools 4.0, available at the time of this writing (July, 2011).

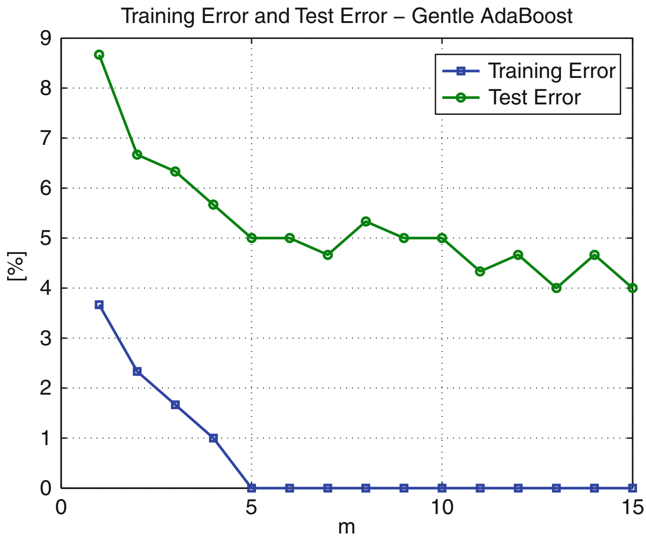

Fig. 2.12 The training error and the test error for Gentle AdaBoost on the WBCD dataset, with $M = 15$ learners

On Fig. 2.12, we have the TE and the test error of Gentle AdaBoost on the WBCD dataset, as a function of the number of WLs.

We see that the TE drops fast on the first few iterations. Even after the TE reaches zero, the test error continues to drop. Figure 2.13 shows the test error rates for Real, Gentle, and Modest AdaBoost classifiers, on the WBCD and Pima datasets, as a function of the number of WLs.

For these three classifiers, we have an adequate test set error rate on both datasets. On the WBCD dataset, the best performance is achieved by Gentle AdaBoost and for the Pima dataset Real AdaBoost attains the best results.

2.6.2.3 Comparison with Other Classifiers

The reported results in Table 2.4 are averages over ten different random replications of different training/testing partitions, for the standard datasets described in Table 2.3. We compare Real, Gentle, and Modest AdaBoost with linear SVM [10, 20, 116] and K-nearest neighbor (KNN) [1] classifiers from the PRTools toolbox. The linear kernel SVM classifiers are trained up to 20,000 iterations and the KNN classifier uses $K = 3$ neighbors. Regarding AdaBoost variants we use the ENTOOL toolbox with $M = 15$ WLs (tree nodes).

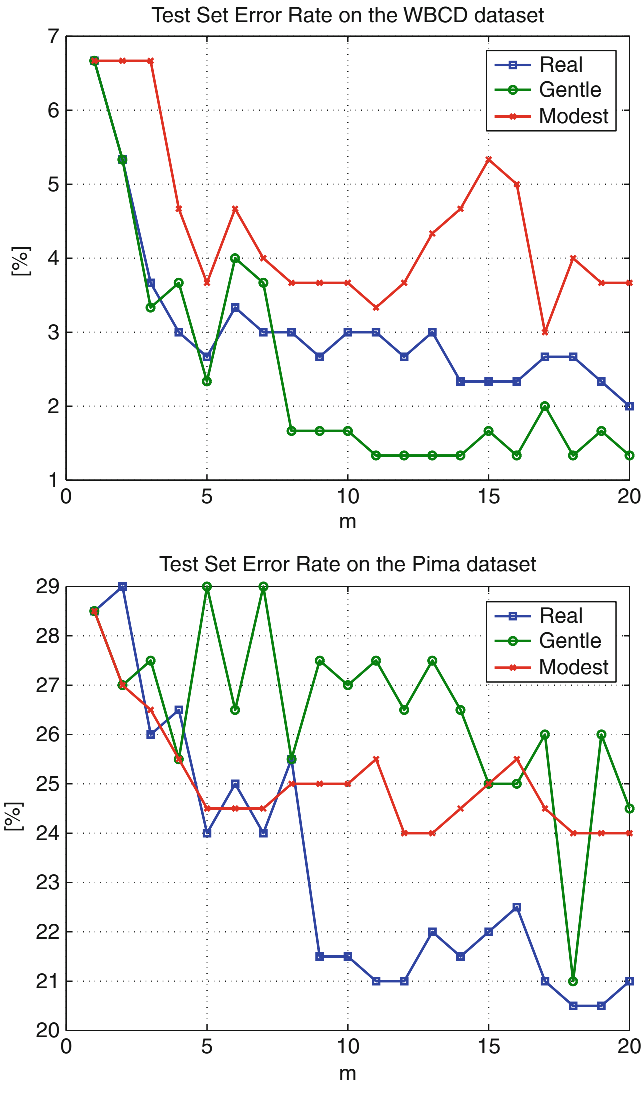

Fig. 2.13 Test error rates for Real, Gentle, and Modest AdaBoost classifiers, on the WBCD and Pima datasets

In many low and medium dimensional datasets, one of the AdaBoost variants attains adequate results being better than SVM and KNN. However, for the higher-dimensional datasets SVM and KNN tend to perform slightly better than these AdaBoost variants.

Table 2.4 Experimental comparison of three AdaBoost variants with linear SVM and 3-NN classifiers. We show the average ± standard deviation of the test set error rate, for ten runs with different random training/test partitions on standard datasets of Table 2.3. The best results are in bold face

Dataset	*Real*	*Gentle*	*Modest*	*SVM*	*3-NN*
Crabs	23.50 ± 11.56	23.50 ± 12.26	19.50 ± 12.12	**5.50 ± 4.97**	33.50 ± 15.28
Phoneme	24.45 ± 4.50	23.70 ± 3.25	23.90 ± 2.63	26.70 ± 2.49	**22.90 ± 1.73**
Abalone	26.25 ± 3.26	26.15 ± 3.12	25.65 ± 3.98	**23.05 ± 4.56**	28.55 ± 3.24
Pima	23.05 ± 3.35	**22.20 ± 3.28**	23.60 ± 2.17	25.85 ± 2.52	28.25 ± 3.50
Contraceptive	**34.35 ± 4.45**	34.50 ± 4.29	35.50 ± 3.06	37.95 ± 3.61	40.05 ± 3.20
Hepatitis	**19.00** ± 5.80	22.25 ± 7.77	20.50 ± 9.92	25.25 ± 7.77	41.25 ± 7.38
WBCD	3.57 ± 1.81	**2.40 ± 1.64**	3.83 ± 2.24	4.80 ± 1.10	7.10 ± 1.28
Ionosphere	9.60 ± 5.02	7.60 ± 2.99	**7.30 ± 2.36**	12.70 ± 4.42	18.20 ± 2.86
SpamBase	14.60 ± 2.68	14.60 ± 3.11	14.10 ± 3.04	**13.57 ± 2.39**	18.33 ± 1.52
Madelon	50.12 ± 2.29	49.78 ± 1.98	**49.45 ± 1.85**	50.97 ± 1.85	50.58 ± 1.97
Colon	**1.67 ± 5.27**	**1.67 ± 5.27**	**1.67 ± 5.27**	13.33 ± 4.10	15.56 ± 6.70
Gisette	13.67 ± 1.59	11.06 ± 0.90	10.62 ± 1.51	7.79 ± 0.67	11.39 ± 1.81
DLBCL	14.00 ± 10.16	19.00 ± 9.94	17.00 ± 10.59	**4.67 ± 4.22**	18.67 ± 7.73
Leukemia	37.00 ± 17.51	37.00 ± 17.51	37.00 ± 17.51	**11.00 ± 7.75**	13.50 ± 9.44
Example 1	12.07 ± 2.62	9.62 ± 1.57	11.63 ± 1.42	**4.30 ± 0.61**	10.98 ± 1.13
Arcene	31.10 ± 6.26	32.00 ± 5.60	29.70 ± 6.36	31.00 ± 0.94	**20.90 ± 6.06**
Prost. Tumor	5.75 ± 4.87	7.25 ± 3.11	4.75 ± 4.03	**3.50 ± 2.49**	15.63 ± 4.87
Dexter	18.13 ± 3.07	15.30 ± 1.06	14.57 ± 1.66	**10.10 ± 0.80**	25.97 ± 4.63

2.7 Summary and Discussion

The AdaBoost algorithm (and its variants) has many practical advantages, which, combined with theoretical guarantees, makes it a very attractive general purpose learning method. On the practical side, boosting algorithms are simple to implement and debug. The base learner and the number of iterations (learners) are the only two important choices to be made.

AdaBoost had been shown to be resistant to overfitting, despite the fact that it can produce combinations involving very large numbers of base classifiers. However, recent studies have shown that this is not the case, even for base classifiers as simple as decision stumps. The success of the boosting algorithms depends on the amount of data available for training as well as on the type of WL.

There are dozens of variants for binary and multiclass problems that have been proven successful on many problems. In recent years, the research on boosting algorithms has been focused mainly on multiclass and semi-supervised problems. The study of online boosting algorithms, in which the training examples arrive one at a time, as contrary to the batch mode, is also a fruitful field of research with many successful algorithms. Many of these algorithms are applied to real-time computer vision problems, such as detection or tracking. This is a focus of intensive current research.

AdaBoost and its variants have also been combined with different machine learning techniques, such as random subspaces [66], genetic algorithms [19], and rotation forest [134].

2.8 Bibliographical and Historical Remarks

There are many papers and tutorials addressing the many boosting algorithms, proposing variants for the multiclass case and/or SSL problems. In this section, we point out some of these elements that can be found in the literature.

For the origins of boosting algorithms, one can be interested in reading about the bootstrapping [37, 38] and bagging (bootstrap aggregation) [11]) techniques. Bootstrap was initially proposed in 1982, but it regained interest in the decade of 1990–1999, in which bagging was proposed. The use of bagging for classification problems is addressed in many papers, see [12, 13, 95, 139, 140] for many applications. The seminal paper of Schapire [95], proposing the first provable polynomial time boosting procedure is a must read. For a comparison of the effectiveness of randomization, bagging, and boosting see [29].

The seminal papers in the middle of the decade of 1990–1999 introducing *adaptive boosting* (AdaBoost) algorithm [47, 48, 50], with the idea that we can *weight* the data instead of resampling it, are also a must read. In the second half of the decade 1990–1999, the AdaBoost algorithm was also extended for regression tasks, as addressed in [3, 31], for instance.

From 1999 until this date, there are dozens of extensions and variants of AdaBoost for supervised and semi-supervised binary and multiclass problems, covering a wide range of successful applications. In this chapter, rather than the theoretical aspects of boosting algorithms, we tried to cover as many variants and successful applications as possible. Section 2.4 covers many variants for supervised learning, whereas Section 2.5 addresses the semi-supervised variants on binary and multiclass problems. In Section 2.6, we have described a wide range of applications. The vast majority of these variants and successful applications were published in the decade of 2000–2009. Online boosting, in contrast with batch boosting, has received a great deal of attention in recent research; we covered many approaches for online boosting on Section 2.6.1.1.

For further reading on adaptive boosting algorithms, please see [82], which complements well this chapter. Whereas we have aimed at covering a wide range of variants, [82] focuses more on theoretical and practical aspects of boosting and ensemble learning. The webpage http://cbio.mskcc.org/~aarvey/boosting_papers.html has many papers, tutorials, and links to software on boosting algorithms. There are also some tutorials about boosting available on-line.[12,13]

[12]http://www.site.uottawa.ca/~stan/csi5387/boost-tut-ppr.pdf

[13]http://www.stat.purdue.edu/~vishy/

The web pages of boosting and adaptive boosting pioneers R. Schapire[14] and Y. Freund[15] have many useful information about boosting algorithms, being useful for both experienced researchers as well as to the new researchers entering this exciting field.

Appendix A: Proofs

A.1: Proof of (2.4)

Consider the final weights, $w_i^{(M+1)}$, and explicitly write the recursion that starts at $w_i^{(1)} = 1/N$ (recall that S_j is the normalizing constant used in line 7 of Algorithm 4, AdaBoost at iteration j)

$$w_i^{(M+1)} = w_i^{(M)} \frac{\exp\left(-\alpha_M y_i H_M(\mathbf{x}_i)\right)}{S_M} = \frac{\prod_{j=1}^{M} \exp\left(-\alpha_j y_i H_j(\mathbf{x}_i)\right)}{N \prod_{j=1}^{M} S_j}. \tag{2.12}$$

This can be re-written as

$$w_i^{(M+1)} = \frac{\exp\left(-y_i \sum_{j=1}^{M} \alpha_j H_j(\mathbf{x}_i)\right)}{N \prod_{j=1}^{M} S_j} = \frac{\exp\left(-y_i\ f(\mathbf{x}_i)\right)}{N \prod_{j=1}^{M} S_j}, \tag{2.13}$$

where $f(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j H_j(\mathbf{x})$, from which we can conclude that

$$\exp\left(-y_i\ f(\mathbf{x}_i)\right) = w_i^{(M+1)}\ N \prod_{j=1}^{M} S_j. \tag{2.14}$$

Now, noticing that $H(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$, and recalling that h denotes the Heaviside function (defined above), we have

$$h(-y_i\ H(\mathbf{x}_i)) = h(-y_i\ f(\mathbf{x}_i)) \leq \exp\left(-y_i\ f(\mathbf{x}_i)\right) = w_i^{(M+1)}\ N \prod_{j=1}^{M} S_j. \tag{2.15}$$

We can now write and bound the TE rate of H,

$$\text{TE} = \frac{1}{N}\sum_{i=1}^{N} h(-y_i\ H(\mathbf{x}_i)) \leq \frac{1}{N}\sum_{i=1}^{N} w_i^{(M+1)}\ N \prod_{j=1}^{M} S_j = \prod_{j=1}^{M} S_j, \tag{2.16}$$

because $\sum_{i=1}^{N} w_i^{(M+1)} = 1$, thus concluding the proof of (2.4). □

[14]http://www.cs.princeton.edu/~schapire/boost.html

[15]http://cseweb.ucsd.edu/~yfreund/papers/index.html

A.2: Proof of (2.5)

Let us plug the expression for α_m (line 5 of AdaBoost) into the expression of the normalizing factor S_m, and use the fact that $y_i\, H_m(\mathbf{x}_i) = 1$, if and only if $H_m(\mathbf{x}_i) = y_i$, while $y_i\, H_m(\mathbf{x}_i) = -1$, if and only if $H_m(\mathbf{x}_i) \neq y_i$,

$$S_m = \sum_{i=1}^{N} w_i^{(m)} \exp\left(y_i\, H_m(\mathbf{x}_i) \log \sqrt{\frac{\mathrm{err}_m}{1-\mathrm{err}_m}} \right) \tag{2.17}$$

$$= \sqrt{\frac{\mathrm{err}_m}{1-\mathrm{err}_m}} \sum_{i:y_i=H(\mathbf{x}_i)} w_i^{(m)} + \sqrt{\frac{1-\mathrm{err}_m}{\mathrm{err}_m}} \sum_{i:y_i\neq H(\mathbf{x}_i)} w_i^{(m)} \tag{2.18}$$

$$= 2\sqrt{\mathrm{err}_m(1-\mathrm{err}_m)}. \tag{2.19}$$

Recalling that $\mathrm{err}_m = 1/2 - \gamma_m$, we have

$$S_m = 2\sqrt{\left(\frac{1}{2} - \gamma_m\right)\left(\frac{1}{2} + \gamma_m\right)} \tag{2.20}$$

$$= \sqrt{1 - 4\gamma_m^2} \tag{2.21}$$

$$= \exp\left(\frac{1}{2} \log\left(1 - 4\gamma_m^2\right)\right) \tag{2.22}$$

$$\leq \exp\left(-2\, \gamma_m^2\right), \tag{2.23}$$

where in (2.23) we have used the inequality $\log u \leq u - 1$ (often referred to as the Gibbs inequality). Plugging this inequality into (2.4), and invoking the assumption $\gamma_m > \gamma$, yields

$$\mathrm{TE} \leq \exp\left(-2 \sum_{m=1}^{M} \gamma_m^2\right) \leq \exp\left(-2\, M\, \gamma^2\right) \tag{2.24}$$

thus proving inequality (2.5).

References

1. D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. In *Machine Learning*, pages 37–66, 1991.
2. E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.
3. R. Avnimelech and N. Intrator. Boosting regression estimators. *Neural Computation*, 11:491–513, 1999.

4. B. Babenko, M. Yang, and S. Belongie. A family of online boosting algorithms. In *Learning09*, pages 1346–1353, 2009.
5. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.
6. K. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 289–296, New York, NY, USA, 2002. ACM.
7. J. Bergstra and B. Kégl. Meta-features and adaboost for music classification. In *Machine Learning Journal : Special Issue on Machine Learning in Music*, 2006.
8. C. Blake and C. Merz. UCI repository of machine learning databases. Technical report, University of California, Irvine, Department of Informatics and Computer Science, 1999.
9. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36:929–965, 1989.
10. B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.
11. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
12. L. Breiman. Bias, variance, and arcing classifiers. Technical report, UC Berkeley, CA, 1996.
13. P. Bühlmann and B.Yu. Analyzing bagging. *Annals of Statistics*, 30:927–961, 2002.
14. W.-C. Chang and C.-W. Cho. Online Boosting for Vehicle Detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(3):892–902, June 2010.
15. O. Chapelle, B. Schalkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, Cambridge, MA, 2006.
16. K. Chen, C. Chou, S. Shih, W. Chen, and D. Chen. Feature selection for iris recognition with adaboost. *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007*, 2:411–414, 2007.
17. K. Chen and S. Wang. Regularized boost for semi-supervised learning. In *Neural Information Processing Systems*, 2007.
18. K. Chen and S. Wang. Semi-supervised Learning via Regularized Boosting Working on Multiple Semi-supervised Assumptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2010.
19. H. Chouaib, O. Terrades, S. Tabbone, F. Cloppet, and N. Vincent. Feature selection combining genetic algorithm and adaboost classifiers. In *ICPR08*, pages 1–4, 2008.
20. N. Christiani and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel based learning methods*. Cambridge University Press, Cambridge, MA, 2000.
21. M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Machine Learning*, volume 48, pages 158–169, 2000.
22. T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.
23. K. Crammer and Y. Singer. A new family of online algorithms for category ranking. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 151–158, New York, NY, USA, 2002. ACM.
24. K. Crammer, Y. Singer, J. K., T. Hofmann, T. Poggio, and J. Shawe-Taylor. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:2003, 2003.
25. D. Cristinacce and T. Cootes. Facial feature detection using adaboost with shape constraints. In *British Machine Vision Conference (BMVC)*, pages 231–240, 2003.
26. F. d'Alché-Buc, Y. Grandvalet, and C. Ambroise. Semi-supervised MarginBoost. In *Advances in Neural Information Processing Systems* 14, pages 553–560, MIT Press, Cambridge, MA, 2001.
27. A. Demiriz, K. P. Bennett, and J. S. Taylor. Linear Programming Boosting via Column Generation. *Machine Learning*, 46(1-3):225–254, 2002.
28. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, B*, 39:1–38, 1977.

29. T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.
30. C. Domingo and O. Watanabe. MadaBoost: A modification of AdaBoost. In *Proceedings of the 13th Annual Conference on Computational Learning Theory (COLT)*, pages 180–189, Palo Alto, CA, 2000.
31. H. Drucker. Improving regressors using boosting techniques. In *Proceedings of the Fourteenth International Conference on Machine Learning* ICML 97, pages 107–115, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
32. H. Drucker, R. Schapire, and P. Simard. Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 42–49, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.
33. H. Drucker and C. Tortes. Boosting decision trees. In *Advances in Neural Information Processing Systems*, volume 8, pages 479–485. MIT Press, 1996.
34. R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.
35. N. Duffy and D. Helmbold. Potential boosters? In *Advances in Neural Information Processing Systems 12*, pages 258–264. MIT Press, New York, NY, 2000.
36. R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. Ridder, D. Tax, and S. Verzakov. PRTools4.1, a Matlab Toolbox for Pattern Recognition. Technical report, Delft University of Technology, 2007.
37. B. Efron. The jackknife, the bootstrap and other resampling plans. *Society for Industrial and Applied Mathematics (SIAM)*, 1982.
38. B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.
39. G. Eibl and K. Pfeiffer. How to make adaboost.M1 work for weak classifiers by changing only one line of the code. In *Machine Learning: Thirteenth European Conference*, volume 1, pages 109–120, 2002.
40. G. Eibl and K. Pfeiffer. Multiclass boosting for weak classifiers. *Journal of Machine Learning Research*, 6:189–210, 2005.
41. A. Esuli, T. Fagni, and F. Sebastiani. MP-Boost: A multiple-pivot boosting algorithm and its application to text categorization. In *Proceedings of the 13th International Symposium on String Processing and Information Retrieval (SPIRE'06)*, 2006.
42. A. Ferreira and M. Figueiredo. Boosting of (very) weak classifiers. In *6th Portuguese Conference on Telecommunications, Conftele'07*, Peniche, Portugal, 2007.
43. F. Fleuret. Multi-layer boosting for pattern recognition. *Pattern Recognition Letters*, 30:237–241, February 2009.
44. Y. Freund. Boosting a Weak Learning Algorithm by Majority. *Information and Computation*, 121(2):256–285, 1995.
45. Y. Freund. An adaptive version of the boost by majority algorithm. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 102–113, 2000.
46. Y. Freund. A more robust boosting algorithm. *http://arxiv.org/abs/0905.2138*, 2009.
47. Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and application to boosting. In *European Conference on Computational Learning Theory – EuroCOLT*. Springer, 1994.
48. Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996.
49. Y. Freund and R. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332. ACM Press, 1996.
50. Y. Freund and R. 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.
51. J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.
52. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

53. V. Gómez-Verdejo, M. Ortega-Moral, J. Arenas-Gárcia, and A. Figueiras-Vidal. Boosting of weighting critical and erroneous samples. *Neurocomputing*, 69(7–9):679–685, 2006.
54. H. Grabner, C. Leistner, and H. Bischof. Semi-supervised On-Line Boosting for Robust Tracking. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision ECCV 2008*, volume 5302, *Lecture Notes in Computer Science*, chapter 19, pages 234–247. Springer, Berlin, Heidelberg, 2008.
55. Y. Grandvalet, F. Buc, and C. Ambroise. Boosting mixture models for semi-supervised learning. In *ICANN International Conference on Artificial Neural Networks*, volume 1, pages 41–48, Vienna, Austria, 2001.
56. V. Guruswami and A. Sahai. Multiclass learning, boosting, and error-correcting codes. In *12th Annual Conference on Computational Learning Theory (COLT-99)*, Santa Cruz, USA, 1999.
57. Z. Hao, C. Shen, N. Barnes, and B. Wang. Totally-corrective multi-class boosting. In *ACCV10*, volume 6495/2011, *Lecture Notes in Computer Science*, pages 269–280, 2011.
58. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, New York, NY, 2001.
59. J. He and B. Thiesson. Asymmetric gradient boosting with application to SPAM filtering. In *Fourth Conference on Email and Anti-Spam (CEAS) 2007*, August 2–3, Mountain View, California, USA.
60. C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 446–453, 2005.
61. X. Huang, S. Li, and Y. Wang. Jensen–Shannon boosting learning for object recognition. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 144–149, 2005.
62. J. Jackson and M. Craven. Learning sparse perceptrons. In *Advances in Neural Information Processing Systems*, volume 8, pages 654–660. MIT Press, 1996.
63. R. Jin, Y. Liu, L. Si, J. Carbonell, and A. Hauptmann. A new boosting algorithm using input-dependent regularizer. In *Proceedings of Twentieth International Conference on Machine Learning (ICML 03)*. AAAI Press, 2003.
64. X. Jin, X. Hou, and C.-L. Liu. Multi-class adaboost with hypothesis margin. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 65–68, Washington, DC, USA, 2010. IEEE Computer Society.
65. G. Jun and J. Ghosh. Multi-class boosting with class hierarchies. *Multiple Classifier Systems*, 5519:32–41, 2009.
66. H. Kong and E. Teoh. Coupling adaboost and random subspace for diversified fisher linear discriminant. In *International Conference on Control, Automation, Robotics and Vision (ICARCV) 06*, pages 1–5, 2006.
67. L. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, Hoboken, NJ, 2004.
68. D. Le and S. Satoh. Ent-boost: Boosting using entropy measures for robust object detection. *Pattern Recognition Letters*, 2007.
69. H. Li and C. Shen. Boosting the minimum margin: LPBoost vs. AdaBoost. *Digital Image Computing: Techniques and Applications*, 0:533–539, 2008.
70. L. Li. Multiclass boosting with repartioning. In *23rd International Conference on Machine Learning (ICML 07)*, Pennsylvania, USA, 2006.
71. P. Li. ABC-Boost: adaptive base class boost for multi-class classification. In *International Conference on Machine Learning (ICML)*, pages 79–632, 2009.
72. S. Li and A. Jain. *Handbook of Face Recognition*. Springer, New York, NY, 2005.
73. S. Li and Z. Zhang. Floatboost learning and statistical face detection. *Transactions on Pattern Analysis and Machine Intelligence*, 26(9):23–38, 2004.
74. C. Liu and H. Shum. Kullback–Leibler boosting. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 587–594, Madison, Wisconsin, USA, 2003.

75. Y. Lu, Q. Tian, and T. Huang. Interactive boosting for image classification. In *Proceedings of the 7th International Conference on Multiple Classifier Systems*, MCS'07, pages 180–189, Berlin, Heidelberg, 2007. Springer-Verlag.
76. S. Lyu. Infomax boosting. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 533–538, 2005.
77. R. Maclin. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551. AAAI Press, 1997.
78. P. Mallapragada, R. Jin, A. Jain, and Y. Liu. SemiBoost: Boosting for Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, 2009.
79. H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *Proceedings of the 24th International Conference on Machine Learning, (ICML)*, pages 609–619, New York, NY, USA, 2007. ACM.
80. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, pages 512–518, MIT Press, Cambridge, MA, 1999.
81. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. *Advances in Large Margin Classifiers*, 1:109–120, 2000.
82. R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*. Springer Verlag, 2006.
83. A. Mohemmed, M. Zhang, and M. Johnston. A PSO Based Adaboost Approach to Object Detection. *Simulated Evolution and Learning*, pages 81–90, 2008.
84. Z. Niu, S. Shan, S. Yan, X. Chen, and W. Gao. 2d cascaded adaboost for eye localization. In *18th International Conference on Pattern Recognition*, 2006.
85. R. Nock and P. Lefaucheur. A Robust Boosting Algorithm. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 319–331, 2002. Springer Berlin/Heidelberg.
86. T. Pham and A. Smeulders. Quadratic boosting. *Pattern Recoginition*, 41:331–341, January 2008.
87. A. Quddus, P. Fieguth, and O. Basir. Adaboost and Support Vector Machines for White Matter Lesion Segmentation in MR Images. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 463–466. IEEE, 2005.
88. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
89. J. Quinlan. Bagging, Boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.
90. G. Rätsch. Robust multi-class boosting. In *Eurospeech 2003, 8th European Conference on Speech Communication and Technology*, page 9971000, Geneva, Switzerland, 2003.
91. B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, MA, 1996.
92. J. Rodriguez and J. Maudes. Boosting recombined weak classifiers. *Pattern Recognition Letters*, 29(8):1049–1059, 2007.
93. A. Saffari, H. Grabner, and H. Bischof. Serboost: Semi-supervised boosting with expectation regularization. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision European Conference on Computer Vision (ECCV) 2008*, volume 5304 *Lecture Notes in Computer Science*, pages 588–601. Springer Berlin/Heidelberg, 2008.
94. A. Saffari, C. Leistner, M. Godec, and H. Bischof. Robust multi-view boosting with priors. In *11th European Conference on Computer Vision (ECCV)*, pages 776–789, Berlin, Heidelberg, 2010. Springer-Verlag.
95. R. Schapire. The strength of weak learnability. In *Machine Learning*, volume 5, pages 197–227, 1990.
96. R. Schapire. Using output codes to boost multiclass learning problems. In *14th International Conference on Machine Learning (ICML)*, pages 313–321, Tennessee, USA, 1997.

97. R. Schapire. Theoretical views of boosting. In *Proceedings of the 4th European Conference on Computational Learning Theory*, EuroCOLT '99, pages 1–10, London, UK, 1999. Springer-Verlag.
98. R. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*, Berkeley, 2002. Springer.
99. R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 322–330, Nashville, TN, 1997.
100. R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
101. R. Schapire and Y. Singer. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168, 2000.
102. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
103. H. Schwenk and Y. Bengio. Adaboosting neural networks: Application to on-line character recognition. In *International Conference on Artificial Neural Networks'97, LNCS, 1327, 967–972*, pages 967–972. Springer, 1997.
104. H. Schwenk and Y. Bengio. Boosting Neural Networks. *Neural Comp.*, 12(8):1869–1887, 2000.
105. C. Shen, J. Kim, L. Wang, and A. van den Hengel. Positive semidefinite metric learning with boosting. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS'09)*, pages 1651–1659, Vancouver, BC, Canada, December 2009. MIT Press.
106. J. Sochman and J. Matas. "Waldboost" learning for time constrained sequential detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) – Volume 2*, pages 150–156, Washington, DC, USA, 2005. IEEE Computer Society.
107. A. Stefan, V. Athitsos, Q. Yuan, and S. Sclaroff. Reducing jointboost-based multiclass classification to proximity search. In *Computer Vision and Pattern Recognition (CVPR)*, pages 589–596. IEEE, 2009.
108. S. Sternig, M. Godec, P. Roth, and H. Bischof. Transientboost: On-line boosting with transient data. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010*, pages 22–27, San Francisco, CA.
109. Y. Sun, S. Todorovic, and J. Li. Unifying multi-class adaboost algorithms with binary base learners under the margin framework. *Pattern Recognition Letters*, 28:631–643, 2007.
110. Y. Sun, S. Todorovic, J. Li, and D. Wu. Unifying the error-correcting and output-code adaboost within the margin framework. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 872–879, New York, NY, USA, 2005. ACM.
111. J. Thongkam, O. Xu, Y. Zhang, F. Huang, and G. Adaboosts. Breast cancer survivability via adaboost algorithms. In *Proceedings of the second Australasian workshop on Health data and knowledge management – Volume 80*, HDKM '08, pages 55–64, Darlinghurst, Australia, 2008. Australian Computer Society, Inc.
112. K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition – CVPR*, volume 1, pages 228–235, 2000.
113. A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854 – 869, March 2007.
114. L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
115. H. Valizadegan, R. Jin, and A. K. Jain. Semi-Supervised Boosting for Multi-Class Classification. In *ECML PKDD '08: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases – Part II*, pages 522–537, Berlin, Heidelberg, 2008. Springer-Verlag.
116. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 1999.

117. V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.
118. R. Verschae, J. Ruiz-del-solar, and M. Correa. Gender classification of faces using adaboost. In *Lecture Notes in Computer Science (CIARP 2006) 4225*, page 78. Springer, 2006.
119. A. Vezhnevets and V. Vezhnevets. Modest AdaBoost - teaching AdaBoost to generalize better. *Graphicon*, 12(5):987–997, 2005.
120. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, Hawaii, 2001.
121. P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.
122. P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *International Conference on Computer Vision – ICCV*, pages 734–741, 2003.
123. P. Viola, J. Platt, and C. Zhang. Multiple instance boosting for object detection. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1417–1424, Cambridge, MA, 2006. MIT Press.
124. L. Wang, S. Yuan, L. Li, and H. Li. Boosting naïve Bayes by active learning. In *Third International Conference on Machine Learning and Cybernetics*, volume 1, pages 41–48, Shanghai, China, 2004.
125. P. Wang, C. Shen, N. Barnes, H. Zheng, and Z. Ren. Asymmetric totally-corrective boosting for real-time object detection. In *Asian Conference on Computer Vision (ACCV)*, pages I: 176–188, 2010.
126. M. Warmuth, K. Glocer, and G. Rätsch. Boosting algorithms for maximizing the soft margin. In *Advances in Neural Information Processing Systems NIPS*, pages 1–8, MIT Press, 2007.
127. M. Warmuth, K. Glocer, and S. Vishwanathan. Entropy regularized LPBoost. In *Proceedings of the 19th International Conference on Algorithmic Learning Theory*, ALT '08, pages 256–271, Springer-Verlag, Berlin, Heidelberg, 2008.
128. M. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 1001–1008, New York, NY, USA, 2006. ACM.
129. J. Warrell, P. Torr, and S. Prince. Styp-boost: A bilinear boosting algorithm for learning style-parameterized classifiers. In *British Machine Vision Conference (BMVC)*, 2010.
130. J. Webb, J. Boughton, and Z. Wang. Not so naïve Bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.
131. P. Yang, S. Shan, W. Gao, S. Z. Li, and D. Zhang. Face recognition using ada-boosted gabor features. In *Proceedings of the 16th International Conference on Face and Gesture Recognition*, pages 356–361, 2004.
132. C. Zhang, P. Yin, Y. Rui, R. Cutler, P. Viola, X. Sun, N. Pinto, and Z. Zhang. Boosting-based multimodal speaker detection for distributed meeting videos. *IEEE Transactions on Multimedia*, 10(8):1541–1552, December 2008.
133. C. Zhang and Z. Zhang. *Boosting-Based Face Detection and Adaptation*. Morgan and Claypool Publishers, 2010.
134. C. Zhang and J. Zhang. Rotboost: A technique for combining rotation forest and adaboost. *Pattern Recognition Letters*, 29(10):1524–1536, July 2008.
135. C. Zhang and Z. Zhang. Winner-take-all multiple category boosting for multi-view face detection. Technical report, One Microsoft Way, Redmond, WA 98052, USA, 2010.
136. L. Zheng, S. Wang, Y. Liu, and C.-H. Lee. Information theoretic regularization for semi-supervised boosting. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 1017–1026, New York, NY, USA, 2009. ACM.
137. M. Zhou, H. Wei, and S. Maybank. Gabor wavelets and AdaBoost in feature selection for face verification. In *Proceedings of the Workshop on Applications of Computer Visions*, pages 101–109, Graz, Austria, 2006.

138. J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–360, 2009.
139. X. Zhu, C. Bao, and W. Qiu. Bagging very weak learners with lazy local learning. In *International Conference on Pattern Recognition (ICPR)*, pages 1–4, 2008.
140. X. Zhu and Y. Yang. A lazy bagging approach to classification. *Pattern Recognition*, 41:2980–2992, 2008.