On Boosting Improvement: Error Reduction and Convergence Speed-Up

Marc Sebban and Henri-Maxime Suchier

EURISE – Université Jean Monnet de Saint-Etienne
23, rue du Dr Paul Michelon, 42023 Saint-Etienne cedex 2, France
{Marc.Sebban, Henri.Maxime.Suchier}@univ-st-etienne.fr

Abstract. Boosting is not only the most efficient ensemble learning method in practice, but also the one based on the most robust theoretical properties. The adaptive update of the sample distribution, which tends to increase the weight of the misclassified examples, allows to improve the performance of any learning algorithm. However, its ability to avoid overfitting has been challenged when boosting is applied to noisy data. This situation is frequent with the modern databases, built thanks to new data acquisition technologies, such as the Web. The convergence speed of boosting is also penalized on such databases, where there is a large overlap of probability density functions of the classes to learn (large Bayesian error). In this article, we propose a slight modification of the weight update rule of the algorithm ADABOOST. We show that by exploiting an adaptive measure of a local entropy, computed from a neighborhood graph built on the examples, it is possible to identify not only the *outliers* but also the examples located in the Bayesian error region. Taking into account this information, we correct the weight of the examples to improve the boosting performances. A broad experimental study shows the interest of our new algorithm, called *i*ADABOOST.

1 Introduction

A large number of studies in machine learning have focused during the last decade on classifier aggregation methods, which aim at improving by voting techniques the performances of a single classifier. Among these methods, the most usually used are probably *bagging* [1], *arcing* [2], and *boosting* [3,4]. In this article, we only focus on this third approach, which received over the last few years a spectacular interest. The two main reasons for this growing popularity are probably due, on the one hand to its simplicity of implementation, and on the other hand to the large number of recently enacted theorems on bounds, margins, or boosting convergence [5,6]. Boosting is known to improve the performances of any learning algorithm, assumed *a priori* unstable, called *weak learner*. The strategy consists in successively training the algorithm (T times) on various probability distributions $D_t(x)$ over the learning sample LS , and in combining the resulting classifiers (called *weak hypotheses*) in an efficient single classifier H . The central point of boosting, and its algorithm ADABOOST [7,8] (see Algorithm

1), is the weight update rule. At each round, the current distribution favors the weights of examples mislabeled ($y(x) \neq h_t(x)$) by the previous hypothesis, that characterizes well the *adaptativity* of *AdaBOOST*.

The first experimental results have shown that boosting seems to be immune against overfitting. Actually, not only the empirical error on the learning sample decreases exponentially with the number of iterations, but also the generalization error drops, even when the empirical error reaches its minimum (even 0). These results incited over the last few years researchers to find theoretical justifications for this behavior. These works made it possible to establish the link between boosting and margin maximization¹ [5], and in particular (i) to show that it is possible to bound the theoretical error by a term decreasing with the margin increase, and (ii) to prove that margins on the training examples increase with the boosting iterations. Parallel to these fundamental results, some studies on boosting are concerned with the improvement of the algorithm *ADABOOST*, which is today confronted with two main problems.

Data : A learning sample LS , a number of iterations T , a weak learner WL

Result : An aggregated classifier H

Initialize distribution: $\forall x \in LS, D_1(x) = \frac{1}{|LS|}$;

for $t = 2$ **to** T **do**

$h_t = WL(LS, D_t)$;

$\epsilon_t = \sum_{e: y(x) \neq h_t(x)} D_t(x)$; $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$;

Distribution Update $/* \forall x \in LS, D_{t+1}(x) = \frac{D_t(x)e^{-\alpha_t y(x)h_t(x)}}{Z_t} */$;
 $/* Z_t$ is a Normalization Factor $*/$;

Return H s.t. $H(x) = \frac{1}{T} (\sum_{t=1}^T \alpha_t h_t(x))$;

Algorithm 1: Pseudo-code of *ADABOOST*.

Firstly, the emergence of very large but often strongly noisy modern databases forces the researchers to study and improve the noise tolerance capacities of boosting. Indeed, while the success of *ADABOOST* is indisputable, there is increasing evidence that the algorithm is quite susceptible to noise [9], resulting in overfitting. Recent works then tried to limit these risks of overfitting. *ADABOOST* tending (wrongly) to exponentially increase the weight of the *outliers*, some algorithms aim at controlling the update rule [10,11,12].

Secondly, the other drawback of the *real world* databases is not directly relating to boosting performances in terms of error, but rather on the convergence speed of *ADABOOST*. We will see in this article that in the presence of a high *overlap* between the probability densities of the classes, the optimal error of the learning algorithm is reached after many iterations (T very large). In other words, *ADABOOST* “loses” time, and thus iterations, by reweighting examples which

¹ The margin expresses the degree of confidence in the class prediction for an example.

theoretically do not deserve any attention, since they belong to the Bayesian error region. Such instances are frequent in the modern databases, which often present a non-null Bayesian error. The rare studies aiming at increasing the convergence speed [13] are more theoretical than usable in practice.

In this article, we deal with the two problems previously mentioned. We propose a modification of the weight update rule which (i) keeps the exponential function to not challenge the error bounds, (ii) but avoids applying it to noisy data and (iii) sets to zero the weights of examples in Bayesian region. In order to detect, at a given step t , not only the noisy data but also those at the border of the classes, we use information contained in a neighborhood graph built on the learning set. The graph is constructed only once, and only the node weights vary during the boosting procedure. We compute from the neighborhood of each example an entropy allowing to evaluate the level of local information. This non-monotone function thus makes it possible to assess, with the current weights, if each example deserves or not to be reweighted. Our procedure consists in assigning to each example a coefficient which estimates, in a way, the confidence that one can have in the new weight $D_{t+1}(x)$ calculated in ADABOOST.

This article is organized as follows. In Section 2, we present a synthetic state of the art of the main approaches, past or in progress, dealing with boosting improvement. Section 3 is devoted to the presentation of our new weight update rule, and to our algorithm *i*ADABOOST. In Section 4, we carry out a broad experimental study aiming at comparing, on many databases, the performances of ADABOOST and *i*ADABOOST, in terms of error and convergence speed.

2 State of the Art

In this section, we carry out a survey on the main methods aiming at improving boosting. We divide them in two categories according to their goal: managing noisy data and dealing with the speed of convergence.

2.1 Noise Tolerance

An experimental study was presented in [9], aiming in particular at comparing the performances of boosting and bagging, on a large number of benchmarks. The main information highlighted in this study is the weak tolerance of both methods to noisy data. More surprisingly, ADABOOST reaches a higher error rate than bagging. Dietterich gives a convincing explanation of the reason of this behavior. He shows that boosting tends to assign the examples to which noise was added much higher weight than other instances. As a result, hypotheses generated in later iterations cause the combined hypothesis to overfit the noise. In order to illustrate this phenomenon, ADABOOST was run on an *a priori* linearly separable sample, artificially corrupted by a given percentage of noise (see Fig. 1). We used *stumps* (decision trees with only a root node) to learn the data. *Stumps* have all the characteristics of good weak hypotheses [10] because they have, according to the *bias-variance* terminology, a low variance and a high bias. Fig. 1 also shows

Fig. 1. An example of overfitting on noisy data. ADABOOST finds the right separator as of the first iteration. It then constraints itself to learn outliers, leading to an overfitting.

the results in terms of success rate estimated by cross-validation, over the first 500 iterations. It is clearly seen that ADABOOST quickly diverges, confirming the overfitting phenomenon on noisy data.

To improve the noise tolerance, two different strategies can be considered. Since ADABOOST is noise sensitive, a first approach would consist in removing noisy data before the boosting procedure. This kind of preprocessing is a matter for *prototype selection* (PS) which tries to *a priori* delete irrelevant or noisy data [14,15]. PS aims not only at reducing storage complexity of costly algorithms but also at improving their performances. To reach these goals, three main types of examples have to be suppressed with efficient heuristics (see Fig. 2). The first category corresponds to outliers (example 1 on the figure). The second concerns examples located in the Bayesian error region (example 2), which do not bring discriminant information to the classifier. The last category represents the instances at the center of a cluster (example 3). Since those examples are correctly labeled by instances located on the convex hull of the cluster, they can be considered as useless, and thus suppressed to reduce storage constraints.

The second approach for treating noisy data aims at taking them into account during the induction process, *i.e.* during the construction of the weak hypotheses. Generally, suggested strategies to avoid overfitting propose a modification of the weight update rule. In MADABOOST [12], the modification is very simple. Because the uncontrolled growth of the weights seems to be the root of the problems of ADABOOST, MADABOOST bounds the weight assigned to each example by its initial probability. In this way, the weights of the examples can not become arbitrarily large as it happens in ADABOOST. While the drawback of this approach is that the boosting speed is much slower than the one for ADABOOST, the authors prove that their moderate weight scheme has the boosting properties. In [11], Freund presents an adaptive version of his “boost by majority” algorithm, suggested in [3], in a new algorithm, called BROWNBOOST. In this approach, the algorithm is optimized to minimize the training error within a pre-assigned number k of boosting iterations. As the algorithm approaches its predetermined end, it becomes less and less likely that examples which have large negative margins will become correctly labeled. According to this remark, Freund uses the following weighting function, which corresponds in fact to the

Fig. 2. Categories of irrelevant examples.

probability density of a binomial variable, which depends on k , on the index i of the current iteration, on the number of correct classifications made so far, and on the success probability $1 - \gamma$ imposed to each weak hypothesis.

$$\alpha_r^i = \binom{k-i-1}{\lfloor \frac{k}{2} \rfloor - r} \left(\frac{1}{2} + \gamma \right)^{\lfloor \frac{k}{2} \rfloor - r} \left(\frac{1}{2} - \gamma \right)^{\lfloor \frac{k}{2} \rfloor - i - 1 + r}$$

This is a non-monotone function, which looks quite like the exponential function for small negative margins. But beyond a certain value, the weight drops, particularly at the end of the k iterations. The advantage of this approach is that outliers will probably be detected, and their weight will thus stop to increase. However, they can be too lately identified, leading to a negative influence on the resulting hypothesis. In GENTLE ADABOOST [10], the authors also use a weighting scheme that uses a function with a lower growth than the exponential one. Boosting is viewed as an approximation to additive modeling on the logistic scale. By fitting an additive model of different and potentially simple functions, it expands the class of functions that can be approximated. Eventually, in [16], the authors present boosting as an optimization problem, in which slack variables are introduced, that permits to relax the constraint over high weighted examples, which are the ones having the worst negative margins.

2.2 Convergence Speed

If the negative impacts of noisy data on boosting performances have been frequently mentioned in the literature, the causes of a slowing down of convergence have rarely been studied. However, we can easily assess the consequences of a high level of density overlap. Indeed, the weight of examples located in Bayesian error region are alternatively increased, because mislabeled, and decreased to allow the correct classification of examples of the other class. This, of course, has an impact on the final classifier H , which will generally need more iterations, and thus more weak hypotheses, to reach its optimum. To illustrate our remarks, ADABOOST was run on an artificial sample containing a Bayesian error of 20%. Fig. 3 shows the success rate over 500 iterations. It can be seen that the optimal success rate of 80%, known *a priori*, is nearly reached after 400 iterations.

In order to reduce the number of weak hypotheses, we proposed in [13] to modify Schapire and Singer's theorem [6], pointed out below, which proves that the minimization of the error is ensured by minimizing each Z_t .

Theorem 1. *The following bound holds on the training error of H :*

Fig. 3. An example of delayed convergence due to the presence of an overlap.

$$\frac{|\{i : H(x) \neq y_i\}|}{m} \leq \prod_t Z_t$$

We modified this theorem in [13] by integrating Bayes risk, allowing to encompass situations where many examples are sharing the same description. Consider the learning sample $LS = (x_1, y(x_1)), \dots, (x_m, y(x_m))$ containing m examples with representation x and class $y(x)$. We define $n^+(x)$ and $n^-(x)$ respectively as the number of positive and negative examples sharing this representation. Let

$$\delta(x) = \frac{|n^+(x) - n^-(x)|}{(n^+(x) + n^-(x))}$$

Theorem 2. *The following bound holds on the training error of H :*

$$\frac{|\{i : H(x) \neq y_i\}|}{m} \leq \left(\prod_t Z_t\right) E_{D_{T+1}}[\delta(x)] + \epsilon^*$$

where ϵ^* is the minimal error on LS and $E_{D_{T+1}}[\delta(x)]$ is the expectation of $\delta(x)$ on distribution D_{T+1} , such that

$$D_{T+1}(x') = \frac{e^{-y(x') \sum_t \alpha_t h_t(x')}}{m \prod_t Z_t}$$

The interest of this theorem is to integrate Bayes risk in this upper bound. It suggests to annihilate the effect of the examples located in the Bayesian error region, the reweighting process favoring those for which $\delta(x)$ is highly in favor of one class. We modified the distribution $D_t(x)$ with the following update rule:

$$\forall x', D'_t(x') = \frac{\sum_{x \in LS^*} D_t(x') [\pi(x, x')] \delta(x)}{\sum_{x'' \in LS} \sum_{x \in LS^*} D_t(x'') [\pi(x, x'')] \delta(x)}$$

Where LS^* represents the set of examples containing only one instance of a given representation, where $[\pi(x, x')] = 1$ if the predicate "x and x' are identical descriptions" is true and $[\pi(x, x')] = 0$ if not. Note that if $LS^* = LS$, we then

have Schapire and Singer's results. The first expected effect of this theorem is related to a faster convergence to the optimal risk. From artificial experiments in feature selection [13], we have observed that the construction of up to 30% weak hypotheses can be saved. Nevertheless, from a practical point of view, the situations where the representation of two examples from different classes is strictly the same are very rare, especially with high representation dimensions. In the next section, we will try, through the use of the information contained in a neighborhood graph, to estimate if an example is potentially located in the Bayesian region, even if it does not share its representation with other examples.

Note that other approaches, which do not belong to the two categories stated before, also tried to improve boosting performances. For example, algorithm *i*BOOST [17] aims at specializing weak hypotheses on examples they are supposed to correctly label. Thanks to boolean variables, each example is weighted in $H(x)$ by a coefficient representing its adequacy with each h_t . The experimental results show a quite good improvement of ADABOOST. In [18], a new boosting algorithm is proposed, in which a test example gets the class of the weighted majority of examples having received the same proportion of vote among the T weak hypotheses. Noise tolerance as well as margin growth are also theoretically studied there. Finally, through REGIONBOOST [19], a new hypothesis weighting scheme is proposed. Weighting is evaluated at the moment of the vote thanks to the k nearest neighbors of the example to label. This approach permits to specialize each classifier on specific areas of the representation space. In spite of a performance improvement, the results show a certain noise sensitivity.

3 The Algorithm *i*ADABOOST

The different strategies listed above for dealing with noisy data are not totally satisfying. Firstly, definitively removing noisy data during a preprocessing stage does not provide a moderate decision rule. Beyond the gain in terms of storage requirements, wrongly removed examples could lead to dramatic effects on the classifier performances. Secondly, as we said before, the step-by-step discovery of noisy data as done in BROWNBOOST is not also completely relevant. Actually, an outlier could have time to damage the final classifier. Moreover, the use of functions different from the exponential one (as in MADABOOST), is likely to call into question some theoretical results on bounds. Finally, the use of the efficient soft margins [16] does not solve the problems linked to bi-modal distributions (with two peaks of density), where relevant instances could be seen as outliers, using weak learners such as stumps. To avoid forcing ADABOOST to learn either noisy data or examples that would become too hard to learn during the boosting process, we are going to build a local information criterion around each example. This one will allow us not only to estimate overfitting risks, but also to evaluate if an example is located in the Bayesian error region. We build a geometrical graph, which will permit us to measure the information around each example. We consider here a very simple graph, the k -nearest-neighbor graph (k NN) [20], built on LS .

Definition 1. Let $N(x)$ be the neighborhood of x such that, $N(x) = \{x' \in LS/x' \text{ is one of the } k\text{NN of } x \text{ using the Euclidean distance}\}$.

Note that it is possible to use other graphs, such as Gabriel graph, Relative Neighborhood graph, minimal spanning tree, Delaunay triangulation [21]. The properties of the k NN graph, and in particular the fact of having an error rate bounded by twice the Bayesian error [22], led us to choose this graph.

Definition 2. Let $n_t^+(x)$ (resp. $n_t^-(x)$), be the sum of the weights $D_t(x')$ of the examples x' having the same label as x (resp. opposite label of x) in $N(x)$.

$$n_t^+(x) = \sum_{x' \in N(x): y(x')=y(x)} D_t(x') \quad n_t^-(x) = \sum_{x' \in N(x): y(x') \neq y(x)} D_t(x')$$

Definition 3. Let $I_t(x)$, the level of confidence in the new weight $D_{t+1}(x)$, at iteration t . $I_t(x)$ is an application from $[-1, +1]$ in $[0, 0.25]$, defined as follows:

$$I_t(x) = |\gamma_t(x)|(1 - |\gamma_t(x)|) \quad \text{where} \quad \gamma_t(x) = \frac{n_t^+(x) - n_t^-(x)}{n_t^+(x) + n_t^-(x)}$$

In order to have a coefficient ranging between 0 and 1, we will rather use $4I_t(x)$ for measuring the confidence in $D_{t+1}(x)$. In order to adjust the weights, we use the following reweighting scheme, where Z'_t is the new normalization coefficient.

$$D'_{t+1}(x) = \frac{4I_t(x)D_{t+1}(x)}{Z'_t}$$

The function $I_t(x)$ is graphically described in Fig. 4. Note that this function was built by merging two ideas coming from boosting and prototype selection. The first makes it possible to solve problems due to the systematic exponential growth of the weight of mislabeled examples. We aim here at using a non-monotone function, while if necessary keeping the exponential update for examples which deserve it. Note that for negative values of $\gamma_t(x)$, $I_t(x)$ looks like the binomial distribution used in [3], except that it does not depend on the iteration index, and can detect an outlier right at the beginning of the process.

The second idea takes into account the different categories of irrelevant examples listed in prototype selection. $I_t(x)$ deals with outliers, overlaps and clusters as well. Indeed, irrelevant examples will be detected by a null $|\gamma_t(x)|$ (Bayesian error region), or equal to 1 (outlier or center of cluster). In all cases, $I_t(x)$ will then be equal to 0, and the example will be annihilated. Between these two minima, the confidence in $D_{t+1}(x)$ is a non-monotone function of $\gamma_t(x)$, which expresses the information level around each example. We introduced all these ideas in a new algorithm, called *i*ADABOOST. Its pseudocode (see Algorithm 2) strongly looks like to ADABOOST's one, since the main modification concerns the information collected within the neighborhood graph, and its use in the computation of the confidence coefficient of $D_{t+1}(x)$.

Fig. 4. The function $I_t(x)$.

Data : A learning sample LS , a number of iterations T , a weak learner WL

Result : An aggregated classifier H

Build the k nearest neighbor graph on LS ;

Initialize distribution: $\forall x \in LS, D_1(x) = \frac{1}{|LS|}$;

for $t = 2$ **to** T **do**

$h_t = WL(LS, D_t)$;

$\epsilon_t = \sum_{e: y(x) \neq h_t(x)} D_t(x)$ and $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$;

Distribution Update: $\forall x \in LS, D_{t+1}(x) = D_t(x)e^{-\alpha_t y(x) h_t(x)}$;

and $D'_{t+1}(x) = \frac{4I_t(x)D_{t+1}(x)}{Z'_t}$;

where $I_t(x) = |\gamma_t(x)|(1 - \gamma_t(x))$ and $\gamma_t(x) = \frac{|n^+(x) - n^-(x)|}{(n^+(x) + n^-(x))}$;

Return H s.t. $H(x) = \frac{1}{T}(\sum_{t=1}^T \alpha_t h_t(x))$;

Algorithm 2: Pseudo-code of $iADABOOST$.

4 Experimental Results

The goal of this section is to compare, in terms of error and convergence speed, the algorithms $ADABOOST$ and $iADABOOST$. Again, we use here stumps as weak hypotheses, and a 10-fold-cross-validation procedure. The kNN graph is built using different values of k . The results listed in this section are those obtained with the optimal value. In order to highlight the ability of $iADABOOST$ to deal with both problems of noisy data and class overlap, we achieved two types of experiments. In the first part, we worked on 11 databases of the UCI repository [23]², in which we introduced 10% of noise. This way to proceed ensures then the presence of a minimum number of outliers in order to test the efficiency of our approach. The second series of experiments aims at controlling the ability of $iADABOOST$ to improve the speed of convergence. We built a learning sample with two linearly separable classes, and we artificially generated a density overlap of $\alpha\%$ of each distribution (α varying from 10 to 50%).

² They have been selected in the limited set of bi-class databases provided by the UCI.

Table 1. Performances in terms of success on 11 bases of the UCI repository.

BASE	STUMP	ADABOOST % succ	iADABOOST	
	% succ		k	% succ
WHITE HOUSE	85.89	85.21	5	85.89
PIMA	66.05	67.61	14	69.17
IONOSPHERE	65.61	70.90	5	74.61
GLASS	64.00	68.8	6	69.3
VEHICLE	60.89	67.7	7	68.5
TIC-TAC-TOE	59.45	68.25	8	69.57
XD6	55.80	69.71	6	69.53
HEART	73.38	72.28	4	73.39
ECHOCARDIOGRAM	55.05	65.43	7	66.92
BREAST	81.79	84.64	5	87.35
CAR	71.60	74.19	10	74.20
AVERAGE	67.23	72.25	7	73.49

Fig. 5. Scatter plot on 11 databases.

Tab. 1 shows results obtained during the first study. We indicated the success rate (*% succ*) for ADABOOST and *i*ADABOOST. In order to show the effect of boosting, we also mentioned the success rate (*stump*) obtained with only one weak hypothesis built from the initial distribution. Finally the optimal value of *k* is indicated for each base. Note that in order to ensure a better comparison, the same folds were used during the cross-validation. The observation of the results shows that the positive effects of *i*ADABOOST are indisputable. Indeed, on 10 bases, our algorithm improves the performances of ADABOOST. On all the 11 databases, the gain with our method is about 1.2 (72.25 vs 73.49), that is highly statistically significant, using a Student test. Another concise way to display the results is proposed in Fig. 5. Each dot represents a database. A dot over the bisecting line expresses that *i*ADABOOST is better than ADABOOST.

Table 2 presents the results about the convergence speed of both algorithms. Several remarks can be made. The first one is relative to the iteration from which the algorithm has reached its optimum. Except for a small overlap level (10%), *i*ADABOOST always reaches its optimum before ADABOOST. The second characterizes the difference between the two algorithms in terms of convergence. Indeed the higher the overlapping rate is, the higher the difference between the two convergence iterations ($T_2 - T_1$) becomes. ADABOOST is less and less efficient for quickly learn the two classes. Finally we tested, thanks to a measure of dispersion, if *i*ADABOOST is more stable than ADABOOST. We calculated the standard deviation (σ_1 and σ_2) over the success rates, from the stabilization iteration of *i*ADABOOST to the end ($T=1000$). One can note that globally, our algorithm offers a higher stability (ranging from 0.1 to 0.3) than ADABOOST, which is more disturbed by the successive weight updates (from 0.26 to 1.51).

Table 2. Performances in terms of convergence speed.

OVERLAP.	<i>i</i> ADABOOST		ADABOOST		$T_2 - T_1$
	T_1	σ_1	T_2	σ_2	
10 %	200	0.29	200	0.30	0
20 %	150	0.11	250	0.26	100
30 %	170	0.24	350	0.56	180
40 %	300	0.24	800	0.82	500
50 %	375	0.28	1000	1.51	625

5 Conclusion

We proposed in this article a modification of the weight update of ADABOOST in order to better take into account, not only noisy data, penalizing the success rate of the final classifier, but also the examples located in the Bayesian error region, slowing boosting convergence. Beyond the excellent performances obtained by our algorithm *i*ADABOOST, we are currently working on theoretical justifications of such an update, and its effects on error bounds. It will be also important to theoretically verify that the use of this update rule does not call into question the main principles of boosting, and that margins keep on increasing with the number of iterations.

References

1. Breiman, L.: Bagging predictors. *Machine Learning* **24** (1996) 123–140
2. Breiman, L.: Bias, variance, and arcing classifiers. Technical Report 460, Department of Statistics, University of California, Berkeley (1996)
3. Freund, Y.: Boosting a weak learning algorithms by majority. *Information and Computation* **121** (1995) 256–285
4. Schapire, R.E.: The strength of weak learnability. *Machine Learning* (1990)
5. Schapire, R.E., Freund, Y., Bartlett, P.L., Lee, W.S.: Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics* **26** (1998) 1651–1686

6. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. In Press, A., ed.: Eleventh Annual Conference on Computational Learning Theory. (1998) 80–91
7. Freund, Y., Schapire, R.: A decision-theoretic generalization of online learning and an application to boosting. *International Journal of Computer and System Sciences* **55**(1) (1997) 119–139
8. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In Kaufmann, M., ed.: Thirteenth International Conference on Machine Learning. (1996) 148–156
9. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning* (1999) 1–22
10. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Technical report (1998)
11. Freund, Y.: An adaptive version of the boost by majority algorithm. *Machine Learning* **43** (2001) 293–318
12. Domingo, C., Watanabe, O.: Madaboost: a modification of adaboost. In Press, A., ed.: Third Annual Conference on Computational Learning Theory. (2000) 180–189
13. Nock, R., Sebban, M.: A bayesian boosting theorem. *Pattern Recognition Letters* **22** (3-4) (2001) 413–419
14. Sebban, M., Nock, R., Lallich, S.: Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems. In: *Journal of Machine Learning Research*. (2003)
15. Wilson, D., Martinez, T.: Reduction techniques for exemplar-based learning algorithms. *Machine Learning* (1998)
16. Ratsch, G., Onoda, T., Muller, K.R.: Regularizing adaboost. In: M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, conference NIPS. (1998)
17. Kwek, S., Nguyen, C.: iboost: Boosting using an i nstance-based exponential weighting scheme. In Springer Verlag, L., ed.: Thirteenth European Conference on Machine Learning. (2002) 245–257
18. Nock, R., Lefaucheur, P.: A robust boosting algorithm. In: Thirteenth European Conference on Machine Learning. (2002)
19. Maclin, R.: Boosting classifiers regionally. In: AAAI/IAAI. (1998) 700–705
20. Cover, T., Hart, P.: Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory* **IT13** (1967) 21–27
21. Preparata, F., Shamos, M.: Pattern recognition and scene analysis. Springer Verlag (1985)
22. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees with misclassification costs. Chapman and Hall (1984)
23. Merz, C.J., Murphy, P.M.: Uci repository of machine learning databases. www.ics.uci.edu/mllearn/mlrepository.html (1996)