Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 171 (2007) 25–41
Artificial Intelligence
www.elsevier.com/locate/artint

A ℝeal generalization of discrete AdaBoost ☆

Richard Nock [a,*], Frank Nielsen [b]

[a] *Université des Antilles-Guyane, UFR DSE—Ceregmia, Campus de Schoelcher, BP 7209, 97275 Schoelcher, Martinique, France*
[b] *SONY CS Labs (FRL), 3-14-13 Higashi Gotanda, Shinagawa-Ku, Tokyo 141-0022, Japan*

Received 1 June 2006; received in revised form 16 October 2006; accepted 16 October 2006

Available online 21 November 2006

Abstract

Scaling discrete AdaBoost to handle real-valued weak hypotheses has often been done under the auspices of convex optimization, but little is generally known from the original boosting model standpoint. We introduce a novel generalization of discrete AdaBoost which departs from this mainstream of algorithms. From the theoretical standpoint, it formally displays the original boosting property, as it brings fast improvements of the accuracy of a weak learner up to arbitrary high levels; furthermore, it brings interesting computational and numerical improvements that make it significantly easier to handle "as is". Conceptually speaking, it provides a new and appealing scaling to $\mathbb{R}$ of some well known facts about discrete (ada)boosting. Perhaps the most popular is an iterative weight modification mechanism, according to which examples have their weights decreased iff they receive the right class by the current discrete weak hypothesis. In our generalization, this property does not hold anymore, as examples that receive the right class can still be reweighted higher with real-valued weak hypotheses. From the experimental standpoint, our generalization displays the ability to produce low error formulas with particular cumulative margin distribution graphs, and it provides a nice handling of those noisy domains that represent Achilles' heel for common Adaptive Boosting algorithms.

Keywords: AdaBoost; Boosting; Ensemble learning

1. Introduction

In supervised learning, it is hard to exaggerate the importance of *boosting* algorithms. Loosely speaking, a *boosting* algorithm repeatedly trains a moderately accurate learner, gets its *weak* hypotheses, combines them, to finally output a *strong* classifier which boosts the accuracy up to arbitrary high levels [14,15]. (Discrete) *Adaboost*, undoubtfully the most popular provable boosting algorithm [7], uses weak hypotheses with outputs restricted to the discrete set of classes that it combines via leveraging coefficients in a linear vote. Strong theoretical issues have motivated the extension of this *discrete* AdaBoost [8] to handle real-valued weak hypotheses as well [8,17,26,29]. Even when only few of them are true generalizations of discrete AdaBoost [17,29], virtually all share a strong background in convex

☆ Extends the paper from the same name that was awarded the Best Paper Award at the 17th European Conference on Artificial Intelligence (2006).

* Corresponding author. Fax: (+596) 596 72 74 03.
E-mail addresses: Richard.Nock@martinique.univ-ag.fr (R. Nock), Nielsen@csl.sony.co.jp (F. Nielsen).
URLs: http://www.univ-ag.fr/~rnock (R. Nock), http://www.csl.sony.co.jp/person/nielsen/ (F. Nielsen).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2006.10.014

optimization originally rooted in a "key" to boosting in AdaBoost: a strictly convex exponential loss integrated into a weight update rule for the examples, loss which upperbounds the error and approximates the expected binomial log-likelihood. However, very little is often known for these algorithms from the seminal boosting model standpoint [14,15,27], a model which roughly requires convergence to reduced true risk under very weak assumptions (with high probability).

In this paper, we propose a new real AdaBoost, a generalization of discrete AdaBoost that handles arbitrary real-valued weak hypotheses. With respect to former real AdaBoosts, the weight update is fundamentally different as it does not integrate anymore the convex exponential loss; also, the leveraging coefficients for the weak hypotheses differ in the output; finally, these leveraging coefficients are given in closed form and their computation can now easily be delayed until the end of boosting, which is not the case for conventional real AdaBoosts [8,17,29]. The major theoretical key feature of this algorithm is that it is a provable boosting algorithm in the original sense. Another point is that it saves computation time with respect to previous generalizations of discrete AdaBoost, that need to approximate the solution of a convex minimization problem at each boosting iteration [17,29]. From the experimental standpoint, the weight update rule, which does not require anymore the approximation of logarithms or exponentials, is less prone to numerical errors. Finally, it prevents or reduces some numerical instabilities that previous generalizations [17,29] face when the weak hypotheses reach perfect, or perfectly wrong, classification. This might explain why experiments clearly display that our algorithm handles noise more efficiently than discrete or real AdaBoosts. Noise handling has soon be described as AdaBoost's potential main problem, see [2].

As a matter of fact, it is quite interesting that our algorithm is indeed a generalization of discrete AdaBoost, as when the weak hypotheses have outputs constrained to the set of classes, both algorithms coincide. From this standpoint, our paper also brings a relevant *conceptual* contribution to boosting. Indeed, we give a complete generalization to $\mathbb{R}$ of popular (discrete) boosting properties, and this is sometimes clearly not trivial. For example, discrete AdaBoost is very often presented as an algorithm that reweights lower the examples that have received the right class. Scaled to $\mathbb{R}$, *this is not true anymore*. Roughly speaking, provided a so-called *Weak Learning Assumption* holds (which states that the classifier is slightly different from random), lower reweighting occurs **only** for examples that receive the right class, **and** on which a measure of the classifier's confidence exceeds a measure of its average confidence (over all examples, known as a *margin*). Only on the discrete prediction framework do these two properties coincide. Furthermore, this scaling property does not hold for previous real AdaBoosts [8,17,26,29].

Section 2 presents some definitions, followed by a section on our generalization of discrete AdaBoost. Section 4 presents and discusses experimental results, and a last section concludes the paper.

2. Definitions and related work

Our framework is rooted into the original weak/strong learning and boosting frameworks, and Valiant's PAC (Probably Approximately Correct) model of learnability [7,15,30]. We have access to a *domain* $\mathcal{X}$ of observations, which could be $\{0,1\}^n$, $\mathbb{R}^n$, etc. Here, n is the number of description variables. More precisely, we collect *examples*, that is, couples (observation, class) written $(\boldsymbol{x}, y) \in \mathcal{X} \times \{-1,+1\}$. "+1" is called the *positive* class (or label), and "−1" the *negative* class. In this paper, we deal only with the two-classes case. Well known transformations exist that allow its extension to multiclass, multilabel frameworks [29]. In this paper, boldfaces such as $\boldsymbol{x}$ denote n-dimensional vectors, calligraphic faces such as $\mathcal{X}$ denote sets and blackboard faces such as $\mathbb{S}$ denote subsets of $\mathbb{R}$, the set of real numbers. Unless explicitely stated, sets are enumerated following their lower-case, such as $\{\boldsymbol{x}_i\colon\ i = 1,2,\ldots\}$ for vector sets, and $\{x_i\colon\ i = 1,2,\ldots\}$ for other sets (and for vector entries). We make the assumption that examples are sampled independently, following an unknown but fixed distribution D over $\mathcal{X} \times \{-1,+1\}$. Our objective is to induce a classifier or *hypothesis* $H : \mathcal{X} \to \mathbb{R}$, that matches the best possible the examples drawn according to D.

For this objective, we define a *strong* learner as an algorithm which is given two parameters $0 < \varepsilon, \delta < 1$, samples according to D a set $\mathcal{S}$ of m examples, and returns a classifier or *hypothesis* $H : \mathcal{X} \to \mathbb{R}$ such that with probability $\geqslant 1 - \delta$, its true risk $\epsilon_{\mathcal{D},H}$ is bounded as follows:

$$\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}\left[\operatorname{sign}\left(H(\boldsymbol{x})\right) \neq y\right] = \epsilon_{\mathcal{D},H} \leqslant \varepsilon. \tag{1}$$

Here, $\operatorname{sign}(a)$ is $+1$ iff $a \geqslant 0$, and -1 otherwise. The time complexity of the algorithm is required to be polynomial in relevant parameters, among which $1/\varepsilon$, $1/\delta$, n. To be rigorous, the original models [15,30] also mention dependences on concepts that label the examples. Examples are indeed supposed to be labeled by a so-called *target* concept,

Input: sample $\mathcal{S} = \{(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i \in \mathcal{X}, y_i \in \{-1, +1\}\}$
$\boldsymbol{w}_1 \leftarrow \boldsymbol{u}$;
for $t = 1, 2, \ldots, T$ **do**
  Get $(h_t : \mathcal{X} \to \mathbb{S}) \leftarrow WL(\mathcal{S}, \boldsymbol{w}_t)$;
  Find $\alpha_t \in \mathbb{R}$;
  Update: $\forall 1 \leqslant i \leqslant m$,

$$w_{t+1,i} \leftarrow w_{t,i} \times \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))/Z_t; \quad (2)$$

end
Output: $H_T(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$

Fig. 1. An abstraction of AdaBoost.

which is unknown but fixed. Distribution D is in fact used to retrieve the examples from this target concept, and the time complexity of the algorithm is also required to be polynomial in its size. Hereafter, we shall omit for the sake of clarity this notion of target concept, which is not important for our purpose, since our analysis may also be fit to handle it as well. A *weak* learner (*WL*) has basically the same constraints, with two notable exceptions: (i) the weak hypotheses it delivers have outputs that can be restricted to a subset $\mathbb{S} \subseteq \mathbb{R}$, and (ii) (1) is only required to hold with $\varepsilon = 1/2 - \gamma$ for some $\gamma > 0$ a constant or inverse polynomial in relevant parameters (this still has to be verified regardless of D). Since predicting the classes at random, such as with an unbiased coin, would yield $\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[\text{sign}(\text{random}(\boldsymbol{x})) \neq y] = 1/2, \forall D$, it comes that a weak learner is only required to perform slightly better than random prediction. In the original models, it is even assumed that δ is also an inverse polynomial in relevant parameters, which makes that the constraints on *WL* are somehow the lightest possible from both the statistical and the computational standpoints. The (discrete) *Weak Learning Assumption* (WLA) assumes the existence of *WL* [14,27]. Simple simulation arguments of *WL* [16] allow to show that the weakening on δ is superficial, as we can in fact weak learn with the same arbitrary δ as for strong learning. However, the conditions on ε are dramatically different, and the question of whether weak and strong learning are equivalent models has been a tantalizing problem until the first proof that there exists *boosting* algorithms that strong learn under the *sole* access to *WL* [27], thus proving that these models *are* indeed equivalent. This first boosting algorithm outputs a large tree-shaped classifier with majority votes at the nodes, each node being built with the help of *WL*. The point is that it is not easy to implement, and it does not yield classifiers that correspond to familiar concept representations.

AdaBoost [7] has pioneered the field of easily implementable boosting algorithms, for which $\mathbb{S} = \{-1, +1\}$. After [7], we refer to it as *discrete* AdaBoost (see Fig. 1 for an abstraction of the algorithm). Basically, AdaBoost uses a weak learner as a subprocedure and an initial distribution $\boldsymbol{u}$ over $\mathcal{S}$, generally uniform, which is repeatedly skewed towards the hardest to classify examples. After T rounds of boosting, its output, H_T, is a linear combination of the weak hypotheses. Below, we give a useful abstraction of AdaBoost, in which Z_t is the normalization coefficient, the elements of $\mathcal{S}$ are enumerated $s_i = (\boldsymbol{x}_i, y_i)$ and their successive weight vector is noted $\boldsymbol{w}_t$, for $t \geqslant 1$. In discrete Adaboost, we would have:

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_{\boldsymbol{w}_t, h_t}}{\epsilon_{\boldsymbol{w}_t, h_t}}, \quad (3)$$

where "ln" denotes the natural logarithm. The two key steps in AdaBoost are the choice of α_t and the weight update rule (see Fig. 1). They are strongly related and follow naturally if we seek to minimize the following observed *exponential loss* [8,29] through the induction of H_T:

$$\epsilon^{\exp}_{\boldsymbol{w}_1, H_T} = \mathbf{E}_{(\boldsymbol{x},y)\sim \boldsymbol{w}_1}\big(\exp\big(-y H_T(\boldsymbol{x})\big)\big), \quad (4)$$

with $\mathbf{E}_{\cdot}$ the mathematical expectation. Since $I[\text{sign}(H_T(\boldsymbol{x})) \neq y] \leqslant \exp(-y H_T(\boldsymbol{x}))$ (with I the indicator function), $\epsilon_{\boldsymbol{w}_1, H_T} \leqslant \epsilon^{\exp}_{\boldsymbol{w}_1, H_T}$, and so minimizing the exponential loss amounts to minimizing the empirical risk as well [8,17, 26,28], and it turns out that it brings a boosting algorithm as well [7,28]. There are other excellent reasons to focus on the exponential loss instead of the empirical risk: it is smooth differentiable and it approximates the binomial log-likelihood [8]. Its stagewise minimization brings both the weight update in (2), and the following choice for α_t:

$$\alpha_t = \arg\min_{\alpha \in \mathbb{R}} \mathbf{E}_{(\boldsymbol{x},y)\sim \boldsymbol{w}_t}\big(\exp\big(-\alpha y h_t(\boldsymbol{x})\big)\big) = \arg\min_{\alpha \in \mathbb{R}} Z_t. \quad (5)$$

One more reason, and not the least, to focus on the exponential loss, is that it brings a stagewise maximization of *margins*. While the empirical risk focuses only on a *binary* classification task (the class assigned is either good or bad), margins scale it to *real* classification, as they integrate both the binary classification task (sign) *and* a real magnitude which quantifies a "confidence" in the label given. Large margin classification can bring very fast true risk minimization [28]. Margins justify to scale $\mathbb{S} = \{-1, +1\}$ to an interval such as $[-1, +1]$ [8,29]; in this case, the sign of the output gives the class predicted. Whenever we still enforce $h_t : \mathcal{X} \to \{-1, +1\}$, (5) admits a closed-form solution, which is naturally (3), and the boosting algorithm is discrete AdaBoost [7,8].

In many domains, real-valued classification with AdaBoost encompasses by far the concept itself. One of the most important and challenging application field for ensemble classifiers is vision [32,33], a domain in which it is easy to obtain weak classifiers via simple features. The task requires however both fast and accurate combinations, which is everything but simple. In these pioneering papers, the authors have chosen to use AdaBoost, which brings the accuracy of the combination. To satisfy the condition of fast processing, the authors consider simple features for weak classifiers, and they choose the discrete version of AdaBoost [7], which makes it necessary to discretize the real values of features by thresholding, and thus eventually loses some useful information (it costs also a little time to compute the thresholds).

Relaxing $\mathbb{S}$ to $[-1, +1]$ can still be handled by algorithm AdaBoost (Fig. 1) and (5), *but* (5) does *not* have a closed-form solution in this case [29]. The algorithm obtained is called *real* AdaBoost, and can be found in [8,17,29]; it is popular, as demonstrated by many applications in language or image processing [11,21,35]. Iterative techniques exist for its fast approximation [24], but they have to be performed at *each* boosting iteration (which buys overall a significant load increase), and it may be the case that the solution found lies *outside* the boosting regime if the number of approximation steps is too small. *Approximations* exist to (5), but they do not necessarily yield a valid generalization of discrete AdaBoost [12,24]. For the purpose of fast processing, some authors have devised various *ad-hoc* approximations of real AdaBoost, but almost all of them are not known to be formal boosting algorithms: for example, Huang et al. [9] use a version of real AdaBoost in which $\alpha_t = 1$, and the update rule is adapted from discrete AdaBoost (they roughly do the same in [10]). Friedman et al. [8], Ridgeway [26] also pick $\alpha_t = 1$. Ridgeway [26] further proposes to leverage the vote by arbitrary values $\lambda \in (0, 1]$ to dampen the variation of the leveraging coefficients. Other authors have devised modifications of discrete AdaBoost to integrate class-dependent misclassification costs, that can be viewed as lifting discrete AdaBoost to handle (few) real values [6,19]. Finally, many papers have modified AdaBoost, to optimize other losses; many of them are rooted into the maximization of the expected binomial log-likelihood [8,18] instead of just the exponential loss (4).

3. Our $\mathbb{R}$eal generalization of AdaBoost

We now give up with the direct minimization of (4), and scale $\mathbb{S} = \{-1, +1\}$ up to any subset of $\mathbb{R}$ itself. This means that the weak classifiers can even be authorized to output values outside interval $[-1, +1]$. Suppose we replace the weight update (2) and Eq. (5) by what follows:

$$w_{t+1,i} \leftarrow w_{t,i} \times \left(\frac{1 - (\mu_t y_i h_t(\boldsymbol{x}_i)/h_t^\star)}{1 - \mu_t^2} \right), \tag{6}$$

$$\alpha_t = \frac{1}{2h_t^\star} \ln \frac{1 + \mu_t}{1 - \mu_t}. \tag{7}$$

Here, we have fixed $h_t^\star = \max_{1 \leqslant i \leqslant m} |h_t(\boldsymbol{x}_i)| \in \mathbb{R}$, the maximal value of h_t over $\mathcal{S}$, and:

$$\mu_t = \frac{1}{h_t^\star} \sum_{i=1}^{m} w_{t,i} y_i h_t(\boldsymbol{x}_i) \in [-1, +1] \tag{8}$$

the normalized *margin* of h_t over $\mathcal{S}$. For the sake of clarity, we suppose in the following subsection that $\forall 1 \leqslant t \leqslant T, h_t^\star < \infty$. Infinite values for $h_t^\star$ can be handled in two ways: either we bias/threshold the output of h_t to make it finite [29], or we code it as ∞, which yields $\alpha_t = 0$ and $\mu_t = \sum_{i:|h_t(\boldsymbol{x}_i)|=\infty} w_{t,i} \mathrm{sign}(y_i h_t(\boldsymbol{x}_i))$. In both cases, $\boldsymbol{w}_{t+1}$ is a valid distribution and the properties below are kept. Let us call AdaBoost$_{\mathbb{R}}$ our real generalization of discrete AdaBoost. Notice that (6) and (7) can be retrieved from AdaBoost via the following *gentle* approximations: (i) approximate the leveraging coefficient as $\alpha_t \approx \mu_t$ (first order approximation to the logarithm in (7)), and (ii) approximate the

exponential update rule via a first-order approximation of the exponential, $\exp(z) \approx 1 + z$. These two approximations are typically not the ones carried out for the implementations of real AdaBoost, as they usually prefer to keep the exponential update rule (AdaBoost's main "trademark"), and the eventual approximations of the leveraging coefficients approximate the solution of (5) via logarithmic quantities [24] (they do not carry out "one more approximation").

3.1. Basic properties

We first show that AdaBoost$_{\mathbb{R}}$ is indeed a generalization of discrete AdaBoost.

Lemma 1. *When* $\mathbb{S} = \{-1, +1\}$, AdaBoost$_{\mathbb{R}}$ = discrete Adaboost.

Proof. In this case, we have $h_t^\star = 1$ and $\mu_t = 1 - 2\epsilon_{\boldsymbol{w}_t,h_t}$, which brings that Eq. (7) is also $\alpha_t = (1/2)\ln((1 - \epsilon_{\boldsymbol{w}_t,h_t})/\epsilon_{\boldsymbol{w}_t,h_t})$, i.e. like in discrete AdaBoost. Our update rule simplifies to:

$$w_{t+1,i} \leftarrow \frac{w_{t,i}(1 - y_i h_t(\boldsymbol{x}_i) + 2y_i h_t(\boldsymbol{x}_i)\epsilon_{\boldsymbol{w}_t,h_t})}{2\epsilon_{\boldsymbol{w}_t,h_t}(1 - \epsilon_{\boldsymbol{w}_t,h_t})},$$

i.e.:

$$w_{t+1,i} \leftarrow \begin{cases} w_{t,i}/(2(1 - \epsilon_{\boldsymbol{w}_t,h_t})) & \text{iff} \quad y_i h_t(\boldsymbol{x}_i) = +1, \\ w_{t,i}/(2\epsilon_{\boldsymbol{w}_t,h_t}) & \text{iff} \quad y_i h_t(\boldsymbol{x}_i) = -1. \end{cases}$$

This is the same expression for the weight update of discrete AdaBoost. □

Now, we show that AdaBoost$_{\mathbb{R}}$ is a boosting algorithm for arbitrary real-valued weak hypotheses. In fact, we show a little bit more, and for this objective, we define the *margin of* H_T *on example* $(\boldsymbol{x}, y)$ as:

$$\nu_T\big((\boldsymbol{x}, y)\big) = \frac{\exp(y H_T(\boldsymbol{x})) - 1}{\exp(y H_T(\boldsymbol{x})) + 1} \in [-1, +1]. \tag{9}$$

This definition of margin extends a previous one for discrete AdaBoost [34], and its choice is discussed in Section 3.3. $\forall \theta \in [-1, +1]$ we also define the classifier's "margin error" as the proportion of examples whose margin does not exceed θ (see also [28]):

$$\nu_{\boldsymbol{u},H_T,\theta} = \sum_{i=1}^{m} u_i I\big[\nu_T\big((\boldsymbol{x}_i, y_i)\big) \leqslant \theta\big]. \tag{10}$$

Whenever no example has zero margin (i.e. H_T predicts a label for all examples), $\epsilon_{\boldsymbol{u},H_T} = \nu_{\boldsymbol{u},H_T,0}$, and $\nu_{\boldsymbol{u},H_T,\theta}$ generalizes $\epsilon_{\boldsymbol{u},H_T}$. Ties are extremely seldom, but even when they occur, $\nu_{\boldsymbol{u},H_T,0}$ is still an upperbound for $\epsilon_{\boldsymbol{u},H_T}$. We let H^* denote some real-valued prediction that matches the *empirical* Bayes rule, computed over $\mathcal{S}$: for any observation $\boldsymbol{x} \in \mathcal{S}$, the sign of $H^*(\boldsymbol{x})$ would be the majority class over all examples of $\mathcal{S}$ whose observation matches $\boldsymbol{x}$. We now prove a first theorem on AdaBoost$_{\mathbb{R}}$.

Theorem 1. $\forall \mathbb{S} \subseteq \mathbb{R}$, $\forall \theta \in [-1, +1]$, *after* $T \geqslant 1$ *iterations, we have*:

$$\nu_{\boldsymbol{u},H_T,\theta} \leqslant \epsilon_{\boldsymbol{u},H^*} + \max\left\{1, \left(\frac{1+\theta}{1-\theta}\right)\right\} \times \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\mu_t^2\right). \tag{11}$$

Proof. We need the following simple lemma.

Lemma 2. $\forall a \in [-1, 1], \forall b \in [-1, 1]$, $1 - ab \geqslant \sqrt{1 - a^2}\exp(-\frac{b}{2}\ln\frac{1+a}{1-a})$.

Proof. The function in the right-hand side is strictly convex in b for $a \neq 0$, and both functions match for $b = \pm 1$ and $a = 0$. Writing the right-hand side $(1 + a)^{(1-b)/2}(1 - a)^{(1+b)/2}$ implies that its limits when $a \to \pm 1$ are zero. □

Consider some example $(\boldsymbol{x}_i, y_i) \in \mathcal{S}$ and some $1 \leqslant t \leqslant T$. The way we use Lemma 2 is simple: fix $a = \mu_t$ and $b = y_i h_t(\boldsymbol{x}_i)/h_t^\star$. They satisfy the assumptions of the lemma, and we obtain:

$$1 - \left(\mu_t y_i h_t(\boldsymbol{x}_i)/h_t^\star\right) \geqslant \sqrt{1-\mu_t^2}\exp\left(-\frac{y_i h_t(\boldsymbol{x}_i)}{2h_t^\star}\ln\frac{1+\mu_t}{1-\mu_t}\right). \tag{12}$$

Unraveling the weight update rule, we obtain:

$$w_{T+1,i} \times \prod_{t=1}^{T}(1-\mu_t^2) = u_i \times \prod_{t=1}^{T}\left(1 - \left(\mu_t y_i h_t(\boldsymbol{x}_i)/h_t^\star\right)\right). \tag{13}$$

Using T times (12) on the right-hand side of (13) and simplifying yields:

$$(w_{T+1,i}/u_i) \times \prod_{t=1}^{T}\sqrt{1-\mu_t^2} \geqslant \exp\left(-y_i H_T(\boldsymbol{x}_i)\right). \tag{14}$$

For any $(\boldsymbol{x}_i, y_i) \in \mathcal{S}$, we have $1 = I[\nu_T((\boldsymbol{x}_i, y_i)) \leqslant \theta] + I[\nu_T((\boldsymbol{x}_i, y_i)) \geqslant -\theta] - I[-\theta \leqslant \nu_T((\boldsymbol{x}_i, y_i)) \leqslant \theta], \forall \theta \geqslant 0$, and $1 = I[\nu_T((\boldsymbol{x}_i, y_i)) \leqslant \theta] + I[\nu_T((\boldsymbol{x}_i, y_i)) \geqslant -\theta] + I[\theta < \nu_T((\boldsymbol{x}_i, y_i)) < -\theta], \forall \theta < 0$.

We let $u_{\boldsymbol{x}}^*$ (resp. $u_{\boldsymbol{x}}^\circ$) denote the total weight in $\boldsymbol{u}$ for the examples of $\mathcal{S}$ whose observation matches $\boldsymbol{x}$, and whose class is (resp. is not) the one chosen by the empirical Bayes rule. We also let $y_{\boldsymbol{x}}^*$ denote this Bayes class. Finally, we let $\mathcal{S}^* \subseteq \mathcal{S}$ to be the set that contains, for each observation $\boldsymbol{x}$ present in $\mathcal{S}$, exactly one example $(\boldsymbol{x}, y_{\boldsymbol{x}}^*)$. Suppose $\theta \geqslant 0$. $\forall (\boldsymbol{x}, y_{\boldsymbol{x}}^*) \in \mathcal{S}^*$, since $\nu_T((\boldsymbol{x}_i, y_i)) = -\nu_T((\boldsymbol{x}_i, -y_i)), \forall 1 \leqslant i \leqslant m$, we obtain:

$$\begin{aligned}
&\sum_{i=1}^{m} u_i I[\boldsymbol{x}_i = \boldsymbol{x}] I\left[\nu_T\left((\boldsymbol{x}_i, y_i)\right) \leqslant \theta\right] \\
&\quad = I\left[\nu_T\left((\boldsymbol{x}, y_{\boldsymbol{x}}^*)\right) \leqslant \theta\right] \times u_{\boldsymbol{x}}^* + I\left[\nu_T\left((\boldsymbol{x}, y_{\boldsymbol{x}}^*)\right) \geqslant -\theta\right] \times u_{\boldsymbol{x}}^\circ \\
&\quad = I\left[\nu_T\left((\boldsymbol{x}, y_{\boldsymbol{x}}^*)\right) \leqslant \theta\right] \times (u_{\boldsymbol{x}}^* - u_{\boldsymbol{x}}^\circ) + \left(1 + I\left[-\theta \leqslant \nu_T\left((\boldsymbol{x}, y_{\boldsymbol{x}}^*)\right) \leqslant \theta\right]\right) \times u_{\boldsymbol{x}}^\circ \\
&\quad \leqslant u_{\boldsymbol{x}}^\circ + I\left[\nu_T\left((\boldsymbol{x}, y_{\boldsymbol{x}}^*)\right) \leqslant \theta\right] \times u_{\boldsymbol{x}}^*,
\end{aligned} \tag{15}$$

since $I[-\theta \leqslant \nu_T((\boldsymbol{x}, y_{\boldsymbol{x}}^*)) \leqslant \theta] \leqslant I[\nu_T((\boldsymbol{x}_i, y_i)) \leqslant \theta]$. It is easy to show that inequality (15) also holds when $\theta < 0$, even when the inequality becomes much looser in this case. We obtain with (15):

$$\begin{aligned}
&\sum_{i=1}^{m} u_i I\left[\nu_T\left((\boldsymbol{x}_i, y_i)\right) \leqslant \theta\right] \\
&\quad = \sum_{i=1}^{m} \sum_{(\boldsymbol{x}, y_{\boldsymbol{x}}^*) \in \mathcal{S}^*} u_i I[\boldsymbol{x}_i = \boldsymbol{x}] I\left[\nu_T\left((\boldsymbol{x}_i, y_i)\right) \leqslant \theta\right] \\
&\quad = \sum_{(\boldsymbol{x}, y_{\boldsymbol{x}}^*) \in \mathcal{S}^*} \sum_{i=1}^{m} u_i I[\boldsymbol{x}_i = \boldsymbol{x}] I\left[\nu_T\left((\boldsymbol{x}_i, y_i)\right) \leqslant \theta\right] \\
&\quad \leqslant \epsilon_{\boldsymbol{u}, H^*} + \sum_{(\boldsymbol{x}, y_{\boldsymbol{x}}^*) \in \mathcal{S}^*} (u_{\boldsymbol{x}}^* + u_{\boldsymbol{x}}^\circ) \rho_{\boldsymbol{x}} I\left[\nu_T\left((\boldsymbol{x}, y_{\boldsymbol{x}}^*)\right) \leqslant \theta\right],
\end{aligned} \tag{16}$$

with $\rho_{\boldsymbol{x}} = u_{\boldsymbol{x}}^*/(u_{\boldsymbol{x}}^* + u_{\boldsymbol{x}}^\circ)$. Consider some example $(\boldsymbol{x}, y_{\boldsymbol{x}}^*) \in \mathcal{S}^*$. Then, we show:

$$I\left[\nu_T\left((\boldsymbol{x}, y_{\boldsymbol{x}}^*)\right) \leqslant \theta\right] \leqslant \max\left\{1, \left(\frac{1+\theta}{1-\theta}\right)\right\} \sum_{i=1}^{m} u_i I[\boldsymbol{x}_i = \boldsymbol{x}] \frac{\exp(-y_i H_T(\boldsymbol{x}_i))}{u_{\boldsymbol{x}}^* + u_{\boldsymbol{x}}^\circ}. \tag{17}$$

If $I[\nu_T((\boldsymbol{x}, y_{\boldsymbol{x}}^*)) \leqslant \theta] = 0$, (17) is true since the right-hand side cannot be negative. So, suppose $I[\nu_T((\boldsymbol{x}, y_{\boldsymbol{x}}^*)) \leqslant \theta] = 1$, i.e. $\exp(-y_{\boldsymbol{x}}^* H_T(\boldsymbol{x})) \geqslant (1-\theta)/(1+\theta)$. Fix for short $z = -y_{\boldsymbol{x}}^* H_T(\boldsymbol{x})$. We have:

$$\begin{aligned}
\sum_{i=1}^{m} u_i I[\boldsymbol{x}_i = \boldsymbol{x}] \exp\left(-y_i H_T(\boldsymbol{x}_i)\right) &= u_{\boldsymbol{x}}^* \exp(z) + u_{\boldsymbol{x}}^\circ \exp(-z) \\
&= (u_{\boldsymbol{x}}^* - u_{\boldsymbol{x}}^\circ)\exp(z) + u_{\boldsymbol{x}}^\circ\left(\exp(z) + \exp(-z)\right).
\end{aligned}$$

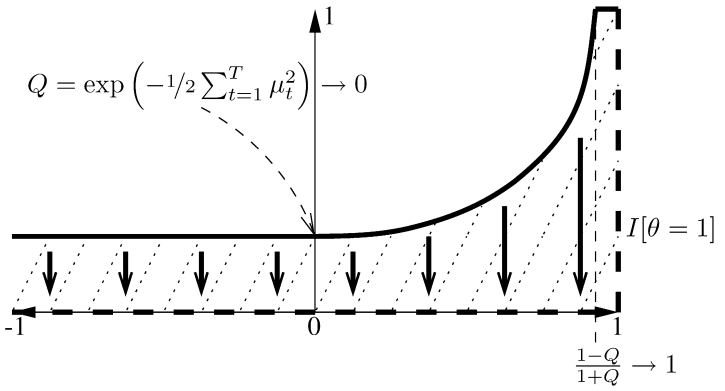

Fig. 2. Graphical representation of Theorem 1 (with $\epsilon_{\boldsymbol{u},H^*} = 0$): the bold curve upperbounds the empirical margin distribution graph (located inside the dotted area). As t increases, provided each $|\mu_t|$ is not too small, this bold curve converges (bold arrows) towards the step function $I[\theta = 1]$ (bold dashed curve), which means that all examples receive the right class with infinite confidence (see text for details).

If $\theta \geqslant 0$, $\exp(z) + \exp(-z) \geqslant 2 \geqslant 2(1-\theta)/(1+\theta)$, and we immediately obtain $(u^*_{\boldsymbol{x}} - u^\circ_{\boldsymbol{x}})\exp(z) + u^\circ_{\boldsymbol{x}}(\exp(z) + \exp(-z)) \geqslant (u^*_{\boldsymbol{x}} + u^\circ_{\boldsymbol{x}}) \times (1-\theta)/(1+\theta)$, from which the right-hand side of (17) is $\geqslant 1$. Now, if $\theta < 0$, $\exp(z) > 1$, and we obtain $(u^*_{\boldsymbol{x}} - u^\circ_{\boldsymbol{x}})\exp(z) + u^\circ_{\boldsymbol{x}}(\exp(z) + \exp(-z)) \geqslant u^*_{\boldsymbol{x}} + u^\circ_{\boldsymbol{x}}$, and again the right-hand side of (17) is $\geqslant 1$. There remains to plug (17) in (16), and then use (14), to obtain:

$$\begin{aligned}
&\sum_{i=1}^{m} u_i I\left[\nu_T\left((\boldsymbol{x}_i, y_i)\right) \leqslant \theta\right] \\
&\quad \leqslant \epsilon_{\boldsymbol{u},H^*} + \max\left\{1, \left(\frac{1+\theta}{1-\theta}\right)\right\} \sum_{(\boldsymbol{x},y^*_{\boldsymbol{x}}) \in \mathcal{S}^*} \rho_{\boldsymbol{x}} \sum_{i=1}^{m} u_i I[\boldsymbol{x}_i = \boldsymbol{x}] \exp\left(-y_i H_T(\boldsymbol{x}_i)\right) \\
&\quad \leqslant \epsilon_{\boldsymbol{u},H^*} + \max\left\{1, \left(\frac{1+\theta}{1-\theta}\right)\right\} \prod_{t=1}^{T} \sqrt{1-\mu_t^2} \sum_{(\boldsymbol{x},y^*_{\boldsymbol{x}}) \in \mathcal{S}^*} \rho_{\boldsymbol{x}} \sum_{i=1}^{m} w_{T+1,i} I[\boldsymbol{x}_i = \boldsymbol{x}].
\end{aligned}$$

The two last sums are an expectation of $\rho_{(.)}$ computed using distribution $\boldsymbol{w}_{T+1}$; since $\rho_{(.)} \leqslant 1$, this expectation is $\leqslant 1$. The statement of the theorem follows after remarking that $\sqrt{1-a^2} \leqslant \exp(-a^2/2), \forall a \in [-1, 1]$. □

Fig. 2 gives a visual interpretation of Theorem 1, when $\epsilon_{\boldsymbol{u},H^*} = 0$: provided each normalized margin is not too small in absolute value, there is a fast convergence of the empirical margin distribution graph towards right classification with infinite confidence for all examples. When the empirical Bayes rule has $\epsilon_{\boldsymbol{u},H^*} > 0$ this convergence is established towards the step function $\epsilon_{\boldsymbol{u},H^*} + I[\theta = 1](1 - \epsilon_{\boldsymbol{u},H^*})$: all examples still receive infinite confidence in their classification, *even* those receiving the wrong label.

3.2. AdaBoost$_{\mathbb{R}}$ *boosts labels and confidences*

Theorem 1 generalizes a well-known convergence theorem for AdaBoost's empirical risk [29] ($\theta = 0$). This generalization is important, as it says that virtually *any* margin error is subject to the same convergence rate towards zero, and not simply the empirical risk. Thus, more than a single point, it gives also a complete curve $f(\theta)$ upperbounding the (empirical) margin distribution graph, which plots the margin error (10) as a function of $\theta \in [-1, 1]$ [28]. To prove that AdaBoost$_{\mathbb{R}}$ is a boosting algorithm, we need a WLA that a real-valued *WL* should satisfy. Its formulation for real-valued hypotheses follows that for the discrete case [14,15,29]: basically, it amounts to say that we want h_t to perform significantly different from *random*, a case which can be represented by $\mu_t = 0$. A natural choice is thus fixing the WLA to be ($\forall t \geqslant 1$):

(real)**WLA** $|\mu_t| \geqslant \gamma$, for the same $\gamma > 0$ as in the discrete WLA.

This, in addition, provides us with a generalization of the discrete WLA [14,27], since we have in this case $\mu_t = 1 - 2\epsilon_{\boldsymbol{w}_t,h_t}$. This brings that either $\epsilon_{\boldsymbol{w}_t,h_t} \leqslant (1/2) - \gamma/2$, or $\epsilon_{\boldsymbol{w}_t,h_t} \geqslant (1/2) + \gamma/2$. It has been previously remarked that this second condition, although surprising at first glance since the empirical risk is worse than random, is in fact equivalent to the first from the boosting standpoint, as it "reverses" the polarity of learning: when h_t satisfies the second constraint, $-h_t$ satisfies the first [8].

Now proving that AdaBoost$_{\mathbb{R}}$ is a boosting algorithm amounts first to using Theorem 1 in the conventional weak/strong learning frameworks, i.e. fix $\theta = 0$ and $\epsilon_{\boldsymbol{u},H^*} = 0$, to obtain under the WLA that after T iterations of AdaBoost$_{\mathbb{R}}$, we have $\epsilon_{\boldsymbol{u},H_T} \leqslant \exp(-T\gamma^2/2)$. Thus, if we run AdaBoost$_{\mathbb{R}}$ for $T = \Omega((1/\gamma^2)\ln m)$ iterations, we get an H_T consistent with $\mathcal{S}$. Since T is polynomial in all relevant parameters, classical VC-type bounds on the deviation of the true risk for linear separators [16,31] immediately bring the following theorem.

Theorem 2. $\forall \mathbb{S} \subseteq \mathbb{R}$*, provided WLA holds,* AdaBoost$_{\mathbb{R}}$ *is a boosting algorithm.*

This theorem relies on the use of Theorem 1 with $\theta = 0$, that is, it does not take into account the influence of the margins' magnitude. We can integrate it in a somewhat stronger boosting-type result, that says that AdaBoost$_{\mathbb{R}}$ does more than fitting well the classes under the WLA: it also brings large confidences into right classification. This amounts to integrate one more parameter (θ) in the strong learning model as described in (1). Suppose that some $-1 < \theta' < 1$ is also given by the user along with ε, δ, and replace the strong learning condition in (1) by this one:

$$\mathbf{Pr}[\nu_{D,H,\theta'} \leqslant \varepsilon] \geqslant 1 - \delta, \tag{18}$$

and the time complexity of the algorithm has to be polynomial also in $1/(1-\theta')$. This means that, with high probability, we want to limit the probability that some example drawn according to D has a "small" local margin. From the standpoint of the true margin distribution graph, when the event $\nu_{D,H,\theta'} \leqslant \varepsilon$ is satisfied, the curve is located below the step function $\varepsilon + I[\theta \geqslant \theta'](1-\varepsilon)$. From Theorem 1, we just have to make $T = \Omega((1/\gamma^2)\ln(m/(1-\theta')))$ steps to have $\nu_{\boldsymbol{u},H_T,\theta'} = 0$. We also remark that the condition $\nu_T((\boldsymbol{x}_i, y_i)) > \theta'$ is equivalent to stating $y_i(H_T(\boldsymbol{x}_i) - y_i \ln((1+\theta')/(1-\theta'))) > 0$, i.e. the $(T+1)$-dimensional linear separator $H_T(\boldsymbol{x}_i) - y_i \ln((1+\theta')/(1-\theta'))$ is consistent with $\mathcal{S}$. There finally remains to use the same arguments as for Theorem 1 to prove that the WLA is also enough to strong learn in the model described by (18), thus proving a boosting-type result integrating both labels and confidences.

3.3. Discussion

Perhaps one of the most important difference with discrete AdaBoost and offsprings [5,7,8,29] lies in the fact that they have been early motivated or built around the appealing intuition that reweighting favors the hard examples on discrete AdaBoost, and more precisely that examples receiving the right class are reweighted lower. This property is appealing, and has certainly participated to their spread and use. However, when scaling the binary classification problem ($\mathbb{S} = \{-1, +1\}$) to $\mathbb{R}$, for this property to fully integrate the extended framework, it should rely entirely on margins (classes + confidences) instead of just classes. This becomes true with AdaBoost$_{\mathbb{R}}$: lower reweighting occurs only for examples on which the current weak classifier's "performance" exceeds its average margin (when $\mu_t > 0$):

$$y_i h_t(\boldsymbol{x}_i)/h_t^\star \geqslant \mu_t. \tag{19}$$

Thus, there can be examples that receive the right class by h_t, and yet that have their weights increased. When $\mu_t < 0$, the polarity of boosting (and reweighting) is reversed in the same way as when $\epsilon_{\boldsymbol{w}_t,h_t} > 1/2$ for discrete AdaBoost. Finally, these properties are true generalization of discrete AdaBoost's, as all coincide again on the discrete case.

3.3.1. Margins

The normalized margin of weak hypothesis h_t in (8) can also be written as $\mu_t = \mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_t}(\nu_t((\boldsymbol{x}, y)))$, with

$$\nu_t\big((\boldsymbol{x}, y)\big) = y \times \frac{h_t(\boldsymbol{x})}{h_t^\star} \in [-1, 1] \tag{20}$$

being the margin of h_t on example $(\boldsymbol{x}, y)$. Except from its output domain, this "local" margin might not seem to bear any other similarity with our corresponding definition for the "strong" hypothesis H_T, (9). Moreover, (20) is evidently much closer to a previous definition coined for real AdaBoosts in [8,28,29]: here, (9) is replaced by:

$$v_T\big((\boldsymbol{x}, y)\big) = y \times \frac{H_T(\boldsymbol{x})}{\sum_{t=1}^{T} \alpha_t} \in [-1, +1], \tag{21}$$

with the assumption that $\alpha_t \geqslant 0, \forall 1 \leqslant t \leqslant T$. In fact, (20) and (21) exactly match when (i) all weak hypotheses have output in $\{-1, 1\}$, (ii) all weak hypotheses have empirical risk $\leqslant 1/2$ on $\boldsymbol{w}_t$, and (iii) there exists $(\boldsymbol{x}_i, y_i) \in \mathcal{S}$ with $H_T(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t$, i.e. the maximal possible value of H_T is realized over $\mathcal{S}$. Thus, the best lifting of (20) to H_T might seem (21) at first glance.

However, the outside appearances are misleading. The reason why (9) turns out to be more convenient comes from the models approximated by the AdaBoost family. It is indeed known since [8] that various offsprings of AdaBoost (including discrete and real versions) can be viewed as carrying out a direct or approximate additive fitting of the symmetric logistic transform:

$$H_T(\boldsymbol{x}) = \ln \frac{\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = +1|\boldsymbol{x}]}{\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = -1|\boldsymbol{x}]}, \tag{22}$$

with the probabilities $\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[.|\boldsymbol{x}]$ to be estimated while learning (this connection is made crisp in [18]). AdaBoost$_{\mathbb{R}}$ can also be viewed as an approximation algorithm of this family. Plugging (22) in (9) yields:

$$\begin{aligned} &v_T\big((\boldsymbol{x}, y)\big) = y \times \delta^*_{\boldsymbol{x}} \in [-1, +1], \\ &\delta^*_{\boldsymbol{x}} = 2\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = +1|\boldsymbol{x}] - 1. \end{aligned} \tag{23}$$

This time, we obtain a local margin similar to (20). This margin has a fundamental property that (21) approximates poorly with (22): it turns out to be the theoretical local margin of the *true Bayes rule* (as opposed to the empirical Bayes rule, see Section 3), with real prediction ($\in [-1, 1]$) computed as $\delta^*_{\boldsymbol{x}} = \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = +1|\boldsymbol{x}] - \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = -1|\boldsymbol{x}]$.

This real prediction, which leverages Bayes rule to real values, has the remarkable property to be the best possible in the sense of Bregman divergences. More precisely, Theorem 1 in [1] yields that regardless of the (properly defined) Bregman divergence $\mathrm{B}(. \parallel .)$ measuring the proximity between a true label y and a real prediction p, we shall always have:

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \delta^*_{\boldsymbol{x}} = \arg\min_{p\in\mathbb{R}} \mathbf{E}_{(\boldsymbol{x},y)\sim \tilde{D}_{\boldsymbol{x}}}\big(\mathrm{B}(y \parallel p)\big).$$

Here, $\tilde{D}_{\boldsymbol{x}}$ is distribution D with support restricted to examples $(\boldsymbol{x}, .)$, normalized with the total weight of these examples. By means of words, $\delta^*_{\boldsymbol{x}} = \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = +1|\boldsymbol{x}] - \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = -1|\boldsymbol{x}]$ is the value that best summarizes on average the different classes of observation $\boldsymbol{x}$ [1]. This quantity is also called *gentle* logistic approximation in [8], and it is reported to be more stable than the full logistic model itself.

There are more reasons to prefer (9) over (21). The first reason is more technical. Theorem 1 shows that $v_{\boldsymbol{u},H_T,\theta}$ vanishes under the WLA regardless of the value of $\theta \in (-1, 1)$ with margin definition (9). Upperbounds for $v_{\boldsymbol{u},H_T,\theta}$ with Eq. (21) are not as easy to read, as all would require to vanish that θ be smaller than fluctuating upperbounds that can be $\ll 1$ [28,29]. It does not seem that it is the boosting algorithm which is responsible, as in our case, using (21) would *not* yield a vanishing $v_{\boldsymbol{u},H_T,\theta}$ when $\theta \geqslant \max_t |\mu_t|/2$, a situation identical to previous analyses [7,28,29]. The second reason is an experimental consequence of the first. Definition (9) makes cumulative margin distributions easier to read, since there is no fluctuating theoretical upperbound <1 for "boostable" margins.

Following (9), we can fully characterize the margin distribution graph of the true Bayes rule. Indeed, this is just the sum of "local" margin distribution graphs, built for each possible observation $\boldsymbol{x} \in \mathcal{X}$. For some observation $\boldsymbol{x}$, denote its true Bayes class:

$$y^*_{\boldsymbol{x}} = \arg\max_{b\in\{-1,+1\}} \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = b|\boldsymbol{x}].$$

Fig. 3 presents an example of a local margin distribution graph. To better catch the picture, consider a domain with zero Bayes error, which is noisified with $\eta \in [0, 1/2]$ class noise rate: each example gets its class flipped with probability η. In this case, we would have $|\delta^*_{\boldsymbol{x}}| = 1 - 2\eta$, and the intermediate stair (for which $z = \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[(\boldsymbol{x}, -y^*_{\boldsymbol{x}})]$ in Fig. 3), would be located at $z = \eta$ (see the experiments for examples).

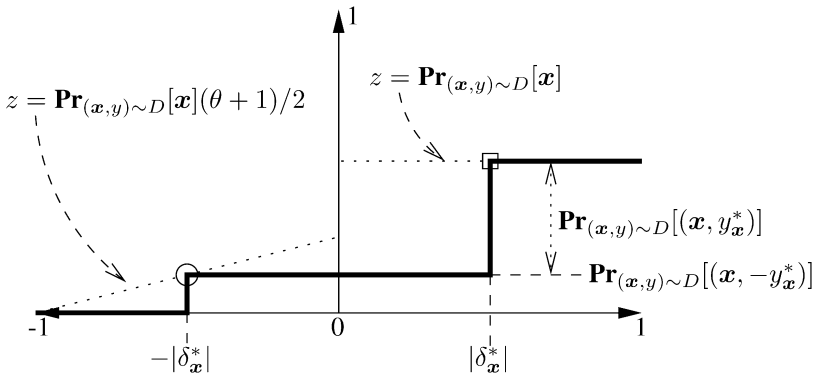

Fig. 3. Local margin distribution graph for the true Bayes rule and for a single observation $\boldsymbol{x} \in \mathcal{X}$ (bold curve). The dotted line with equation $z = \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[\boldsymbol{x}](\theta+1)/2$, depicts the location inside which the negative stair of Bayes rule's prediction is located as a function of $\delta^*_{\boldsymbol{x}}$ (the circle). The dotted line with equation $z = \mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[\boldsymbol{x}]$ does the same for the positive stair (the square; see text for details).

The reasons why margin distribution graphs, better than just errors, are good tools to evaluate the goodness-of-fit of H_T, come again from the models built. In Valiant's seminal PAC model [30], we only have to use the sign of H_T, eventually leaving its absolute value, if needed, to represent some sort of confidence in the prediction [29]. Thus, using labels and counting errors is enough. The logistic framework [8] makes it possible to do more, by integrating sign and confidence into an estimator of the class conditional probabilities:

$$\hat{\mathbf{Pr}}[y = +1|\boldsymbol{x}] = \frac{\exp(H_T(\boldsymbol{x}))}{1 + \exp(H_T(\boldsymbol{x}))}, \tag{24}$$

$$\hat{\mathbf{Pr}}[y = -1|\boldsymbol{x}] = \frac{1}{1 + \exp(H_T(\boldsymbol{x}))}. \tag{25}$$

Thus, when plotting both the margin distribution graph of H_T on testing, and that of the true Bayes rule, the goodness-of-fit should be as better as both curves come closer to each other, meaning not only that the labels tend to be the same, but also that the local class conditional probabilities tend to match. Lifting the usual properties of classifiers in the classical labels/errors framework, to the logistic framework, is not immediate. Consider for example overfitting. Such a situation typically occurs when a classifier becomes "so" complicated that it starts to model better $\mathcal{S}$ at the expense of the whole domain itself, meaning that its true risk increases while its empirical risk may still decrease. This situation has an impact on class conditional probabilities as well, but it remains "punctual", as for example it only means a greater intersection between the (test) margin distribution graph and axis $\theta = 0$. In fact, overfitting as defined here only means differences with respect to the threshold $1/2$ for the class conditional probabilities, but it scarcely means anything for their estimation. For example, a variation of 0.01 of $\hat{\mathbf{Pr}}[y = +1|\boldsymbol{x}]$ may be enough to flip the label predicted for observation $\boldsymbol{x}$ (e.g. passing from 0.495 to 0.505), while a variation of 0.49 of $\hat{\mathbf{Pr}}[y = +1|\boldsymbol{x}]$ may not be enough to flip it (e.g. passing from 0.005 to 0.495). In the logistic framework, overfitting rather comes from differences between the estimations and the true values of class conditional probabilities, or similarly from the differences between the corresponding margin distribution graphs. Consider for example a situation in which, for many $\boldsymbol{x} \in \mathcal{X}$, we have $\hat{\mathbf{Pr}}_H[y = +1|\boldsymbol{x}] = 0$ for a first classifier H (in the PAC framework vocabulary, we predict class -1 with infinite confidence), $\hat{\mathbf{Pr}}_{H'}[y = +1|\boldsymbol{x}] \in [0.40, 0.45]$ for another classifier H', while $\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[y = +1|\boldsymbol{x}] \in [0.49, 0.50)$ for the true Bayes rule. Clearly, H and H' achieve the same true risk over all corresponding examples. However, H predicts the *wrong* label for nearly half of the examples with observation $\boldsymbol{x}$, still with *infinite* confidence, while H' is more cautious as its wrong predictions have confidence approaching that of the true Bayes rule (approximately zero). When it comes that such a situation (of H) occurs and/or as it becomes more visible, we can say that there starts to be overfitting. Visually, the corresponding margin distribution graph becomes "sticked" to axis $\theta = -1$. It is worthwhile remarking that the margin distribution graph of the true Bayes rule can *never* be sticked to this axis, regardless of the domain, see Fig. 3. Thus, overfitting also means margin distribution graphs that cannot match Bayes rule. The experimental section presents examples of such curves. More generally, when comparing two algorithms (different from Bayes rule), the "best" of both is the one which misclassifies the examples with the smallest possible

confidences, i.e. the one with the margin distribution graph below the other for values of $\theta \leqslant 0$. From this standpoint, the classical true risk comparison in Valiant's PAC framework is thus a particular case of the logistic framework.

To finish up with margins, let us comment on (8). When H_T is a linear separator, it is particularly relevant to perform boosting with domain-partitioning weak hypotheses [8,23], i.e. classifiers that fit to a local, bucket-wise computation of the conditional probabilities in (22). The most popular examples are decision trees, but other possible examples include decision lists and symmetric functions [23]. In such cases, the margin of a classifier as depicted in (8) admits nice expressions related to Bregman divergences [17]. Suppose for short that domain $\mathcal{X}$ is partitioned by h_t into a finite number of subsets, whose general term is $\mathcal{X}_\ell$. Let $\tilde{\boldsymbol{w}}_{t,\ell}$ be the two-dimensional distribution vector whose entries, written $\tilde{w}^-_{t,\ell}$ and $\tilde{w}^+_{t,\ell}$, respectively denote the normalized proportion of examples (with respect to $\boldsymbol{w}_t$) from the negative and positive classes in $\mathcal{S} \cap \mathcal{X}_\ell$, thus satisfying $\tilde{w}^-_{t,\ell} + \tilde{w}^+_{t,\ell} = 1$. Let $\tilde{\boldsymbol{w}}_t$ denote the distribution induced by the whole partition, with $\tilde{w}_{t,\ell} = \sum_{i:s_i \in \mathcal{X}_\ell} w_{t,i}$. There are essentially two ways to define real values for $h_{t,\ell}$, the output of h_t on some $\boldsymbol{x} \in \mathcal{X}_\ell$. The first is to use the logistic prediction, and define $h_{t,\ell} = (1/2)\ln(\tilde{w}^+_{t,\ell}/\tilde{w}^-_{t,\ell})$ [8,29]. The second is to use its gentler alternative, and define $h_{t,\ell} = \tilde{w}^+_{t,\ell} - \tilde{w}^-_{t,\ell}$ [8]. In both cases, it is straightforward to show that the classifier's margin simplifies to:

$$\mu_t = \frac{1}{2h^\star_t}\mathbf{E}_{\ell\sim\tilde{\boldsymbol{w}}_t}\big(\mathrm{B}(\tilde{\boldsymbol{w}}_{t,\ell} \parallel \mathbf{1} - \tilde{\boldsymbol{w}}_{t,\ell})\big),$$

where $\mathrm{B}(. \parallel .)$ is a Bregman divergence [17]: the Kullbach–Leibler divergence for the logistic prediction, and the L_2^2 divergence for its gentler alternative. Since a Bregman divergence quantifies a distortion that is non negative, and zero iff its two arguments are equal, this helps to see the classifier's margin, and the WLA, as the expectation of local discrepancies between the positive and the negative class: the larger they are, the better is the classifier. Obviously, if we had picked discrete values $h_{t,\ell} \in \{-1, +1\}$, such as the local majority class, we would have obtained $\mu_t = 1 - 2\epsilon_{\boldsymbol{w}_t,h_t}$.

3.3.2. Computations and numerical stability

A first difference with previous generalizations of discrete AdaBoosts that do not fix *ad hoc* values for α_t [8,17,29] is computational. Eq. (5) has no closed form solution in the general case, so they all need to approximate α_t. The problem is convex and single variable, so its approximation is simple, but it needs to be performed at *each* iteration, which buys a significant additional computation time with respect to AdaBoost$_\mathbb{R}$, for which α_t is exact. Approximating has another drawback: if not good enough, the current iteration may lie outside the boosting regime [8,17,29].

The extensions of discrete AdaBoost [8,17,29] face technical and numerical difficulties to compute α_t when h_t or $-h_t$ reaches consistency, that is, when $\epsilon_{\boldsymbol{w}_t,h_t}$ approaches its extremal values, 0 or 1. On the extreme values, there is no finite solution to Eq. (5), and thus theoretically no weight update. In our case, the problem does not hold anymore, as the multiplicative update of Eq. (6) is never zero nor infinite if we adopt the convention that $0/0 = 1$. Indeed, a numerator equals zero iff all numerators equal zero iff all denominators equal zero. Thus, zeroing any numerator or denominator, which amounts to making either perfect or completely wrong classification for h_t on $\mathcal{S}$, brings *no weight change* in $\boldsymbol{w}_{t+1}$.

A second, well known technical difficulty for some extensions of discrete AdaBoost [8,17,29], occurs when the empirical risk approaches 0 or 1, regions where $|\alpha_t|$ has extremely large regimes. In this case, the numerical approximations to exponentials in Eq. (5), with the approximations of α_t, make the computation of the weights very instable. Clearly, large multiplicative coefficients for the weight update are possible for AdaBoost$_\mathbb{R}$. However, instability is less pronounced, since we have also split the computation of the leveraging coefficients and that of the weight update, allowing the computation of *all* α_t to be delayed till the end of boosting.

4. Experiments

Experiments were carried out on 25 domains, most of which come from the UCI repository [3]. Domains with more than two classes (indicated *2C*) were transformed in a two class problem by grouping all classes but the first into one: sometimes, this brought domains with highly unbalanced classes, thereby complicating even further the learning task. On each domain, we have run discrete AdaBoost [7], AdaBoost$_\mathbb{R}$ and the real AdaBoost of [17,29]. *WL* is set to a rule (monomial) learner, with fixed maximal rule size r (attribute number) [22,25]. When the output is restricted to $\{-1, +1\}$, we pick for the output of h_t (when triggered) the majority class according to $\boldsymbol{w}_t$. When the output is $\mathbb{R}$

(unrestricted), we use the same method as for decision trees with real values at their leaves [29]. Let $\tilde{w}_t^b$ be the total weight in $\boldsymbol{w}_t$ of the examples that fire rule h_t, and that belong to class $b1$, with $b \in \{+, -\}$. Then, for all the examples that trigger the rule, the output of h_t is a local approximation of the logistic transform (22), [29]:

$$h_t = \frac{1}{2} \log \frac{\tilde{w}_t^+}{\tilde{w}_t^-}.$$

The same computations are done for the set of examples that do not trigger the rule (majority class or logistic approximation). h_t is grown following a procedure similar to decision trees, the repetitive minimization of an index function, which is Matsushita's error in our case [4,20], see [22,29] for details. True risks are estimated using a 10-fold stratified cross validation procedure on the whole data. Since each rule h_t has two possible outputs, yh_t has four possible values, and so the analytic solution for α_t of discrete AdaBoost does not apply for real AdaBoost [17,29]. The true α_t is approximated from (5) using a simple dichotomous search until the relative error does exceed 10^{-6}, using results of [24] to make it faster. With this, we have empirically found that the execution time for real AdaBoost [8,17,29] is still on average more than 100 times that of discrete AdaBoost and AdaBoost$_{\mathbb{R}}$.

4.1. General results

We first give some general comments on results that were obtained at early and reasonable stages of boosting, namely after $T = 10$ and $T = 50$ steps of boosting, for a rule learner configured with $r = 2$. Fig. 4 summarizes the results obtained. While AdaBoost$_{\mathbb{R}}$ tends to perform the best, interesting patterns emerge from the simulated domains from which we know everything about the concept to approximate. Easy domains such as Monks(1+2) [3] are those

Domain	$T = 10$			$T = 50$		
	D	U	T	D	U	T
Balance (2C)	**8.73**	**8.73**	*9.05*	4.44	**3.81**	*4.92*
Breast-Wisc	**4.51**	4.65	*4.79*	4.22	**3.38**	*4.51*
Bupa	*34.57*	*34.57*	**32.85**	*30.81*	**27.71**	28.57
Echocardio	*31.43*	**26.43**	30.71	*30.00*	**25.71**	27.86
Glass2	*22.94*	**18.24**	19.41	*17.65*	*17.65*	**15.88**
Hayes Roth (2C)	16.47	*24.11*	**14.71**	*19.41*	**14.71**	15.88
Heart	18.51	**16.67**	*18.89*	*19.63*	**16.67**	*19.63*
Heart-Cleve	*23.87*	21.29	**19.68**	**17.42**	19.35	*20.97*
Heart-Hungary	*19.33*	*19.33*	**16.33**	*21.33*	**16.33**	18.33
Hepatitis	16.47	**15.29**	*17.05*	**14.71**	15.29	*18.23*
Horse	17.10	**16.31**	*18.16*	20.00	**16.31**	*33.68*
Labor (2C)	*18.33*	**6.67**	11.67	*8.33*	**5.00**	*8.33*
Lung cancer (2C)	**27.50**	**27.50**	*30.00*	**25.00**	**25.00**	*30.00*
LEDeven	**9.76**	*17.07*	11.22	10.24	**9.51**	*10.98*
LEDeven+17	22.68	**21.46**	*25.60*	*26.34*	**25.36**	26.10
Monks1	*25.18*	*25.18*	**16.00**	13.39	*17.50*	**1.50**
Monks2	*34.75*	33.93	**32.28**	34.26	*37.70*	**10.17**
Monks3	2.85	*3.57*	**2.14**	**1.43**	*1.79*	*1.79*
Parity	45.93	**45.19**	*47.78*	**46.30**	*47.78*	**46.30**
Pima (2C)	**24.42**	24.55	*25.32*	24.94	**24.03**	*25.71*
Vehicle (2C)	**26.00**	*27.17*	26.35	25.41	**25.29**	*25.88*
Votes	**4.78**	5.00	*5.45*	**3.86**	5.00	*5.68*
Votes w/o	*9.78*	**8.86**	*9.78*	**10.23**	**10.23**	*10.45*
XD6	20.32	*21.64*	**19.51**	*15.74*	14.92	**13.77**
Yeast (2C)	*28.93*	28.73	**26.80**	26.93	*27.33*	**26.67**
#best	7	11	9	7	15	6
#second	9	6	5	9	4	5
#*worst*	9	8	11	9	6	14

Fig. 4. Estimated true risks on 25 domains, comparing discrete AdaBoost [7] (D), AdaBoost$_{\mathbb{R}}$ (U) and the real AdaBoost of [8,17,29] (T). For each domain, we put in emphasis the **best** algorithm(s) and the *worst* algorithm(s) out of the three. The last 3 rows count the number of times each algorithm counts respectively among the **best**, second, and *worst*.

on which real AdaBoost performs the best and converges the fastest. Increasing further T makes that real AdaBoost outstrips even more the two other algorithms. However, as the domain gets complicated, AdaBoost$_{\mathbb{R}}$becomes the algorithm that beats the other two when T increases. Consider the following domain ordering, from the easiest to the hardest: Monks(1+2) (no noise, no irrelevant attributes), XD6 (10% class noise, one irrelevant attribute), LEDeven (10% attribute noise), LEDeven+17 (LEDeven+17 irrelevant attributes) [3,22]. Real AdaBoost beats the other two algorithms on XD6, but another experiment on larger classifiers ($r = 3$; $T = 100$) reveals that AdaBoost$_{\mathbb{R}}$becomes the winner and approaches Bayes risk with 11.15% error, while discrete and real AdaBoost respectively achieve 11.47% and 12.46% error (statistically worse in that latter case). Winning occurs even sooner on LEDeven ($T = 50$) and LEDeven+17 ($T = 10$). One reason for this phenomenon might be the fact that the reweighting scheme of AdaBoost$_{\mathbb{R}}$is actually gentler than the others, especially on noisy examples: discrete and real AdaBoost are subject to very large weight update, due to the exponential update rule and the fact that higher reweighting can occur on the sole basis of the binary classification result (good/bad class), even when the classifier has minute confidence on the label it predicts. This cannot happen in our case if the classifier's margin is negative; whenever it is positive, examples that receive the right class can still be reweighted higher, thus counterbalancing higher reweighting for eventual noisy examples. Gentler updating, such as by thresholding, has soon been proposed as a line of research to improve noise handling [5]. In fact, it may well be also useful to tackle overfitting.

4.2. Noise handling and overfitting

In order to shed some more light on noise handling, we have drilled down into the results of domains LEDeven and XD6 [3], by plugging in variable noise rates to see the way the margin errors degrade when the problems get noisier, and harder. LEDeven is a seven bits problem that describe the ten digits of old pocket calculators. Examples are picked uniformly at random and the ten possible classes and grouped in two: even/odd. Each description variable gets flipped with $\eta\%$ chances (in the original domain, $\eta = 10\%$). XD6 is an $n = 10$ problem that describes a noisy disjunctive normal form formula. Name $v_1, v_2, \ldots, v_{10}$ ten Boolean variables. Observations are picked at random, and then labeled positive iff $(v_1 \wedge v_2 \wedge v_3) \vee (v_4 \wedge v_5 \wedge v_6) \vee (v_7 \wedge v_8 \wedge v_9)$ is true. Thus, v_{10} is irrelevant in the strongest sense [13]. Afterwards, with $\eta\%$ chances, the class gets flipped (in the original domain, $\eta = 10\%$). LEDeven and XD6 cover a broad range of difficulties to study noise handling and overfitting: while LEDeven has variable attribute noise, XD6 has variable class noise, an irrelevant attribute and more unbalanced classes.

We have computed margin distribution graphs on training and testing for both domains, and for small and large values of parameters r and T (respectively 2..6 for r, and 20..1000 for T). Each time, $\mathcal{S}$ is simulated with $m = 300$ examples. On test margin distribution graphs, we have computed the exact curve for Bayes rule in order to make comparisons following Section 3.3.1. Among the numerous curves obtained, we have chosen to summarize the whole results obtained, and report here only the most important curves, obtained for the largest rules ($r = 6$) and various noise rates $\eta \in \{10\%, 20\%, 30\%, 40\%\}$: see Figs. 5 and 6. Remark that the shape of Bayes curves are more complex for LEDeven, as noise affects attributes instead of classes. In both domains, the training margins clearly display that the real AdaBoost of [8,17,29] is the fastest to converge to perfect classification, followed by discrete AdaBoost [7], and then by AdaBoost$_{\mathbb{R}}$("us"). The phenomenon is naturally more prominent for XD6, as the domain is easier to handle than LEDeven, and more prominent as r increases, as it allows a faster fitting of data. The empirical margin distribution graphs for real AdaBoost exhibit two-steps plateaus when the rules become complex ($r = 6$, domain XD6), one plateau near 0% for θ smaller than 0.1 (approximately), and one plateau which increases with η for larger values of θ. This plateau shape is in good accordance with Theorem 1, which proves that after a sufficiently large number of boosting rounds under the WLA, the empirical margin distribution graph is bounded above by a plateau located at the empirical Bayes risk.

The *test* margin distribution graphs display a completely different pattern. On both domains and a majority of curves ($r, \eta, T, \theta \leqslant 0$), AdaBoost$_{\mathbb{R}}$ beats both other algorithms. The phenomenon becomes even more visible as r, η or T increase. A glimpse at the curves for $r = 6$, $\eta = 40\%$ for both domains (Figs. 5 and 6) is enough to see this phenomenon, as well as the fact that the test curves of real AdaBoost also tend to exhibit a multi plateaus shape that becomes more visible as η increases. This shape observed on testing for real AdaBoost on these cases (which is also observable—though less visible—for discrete AdaBoost, and almost not observed for AdaBoost$_{\mathbb{R}}$) indicates that the confidence in classification becomes virtually "infinite" for almost *all* examples, i.e. for those that receive the right class, *and also* for many receiving the wrong class. As discussed in Section 3.3.1, this, we think, indicates a tendency

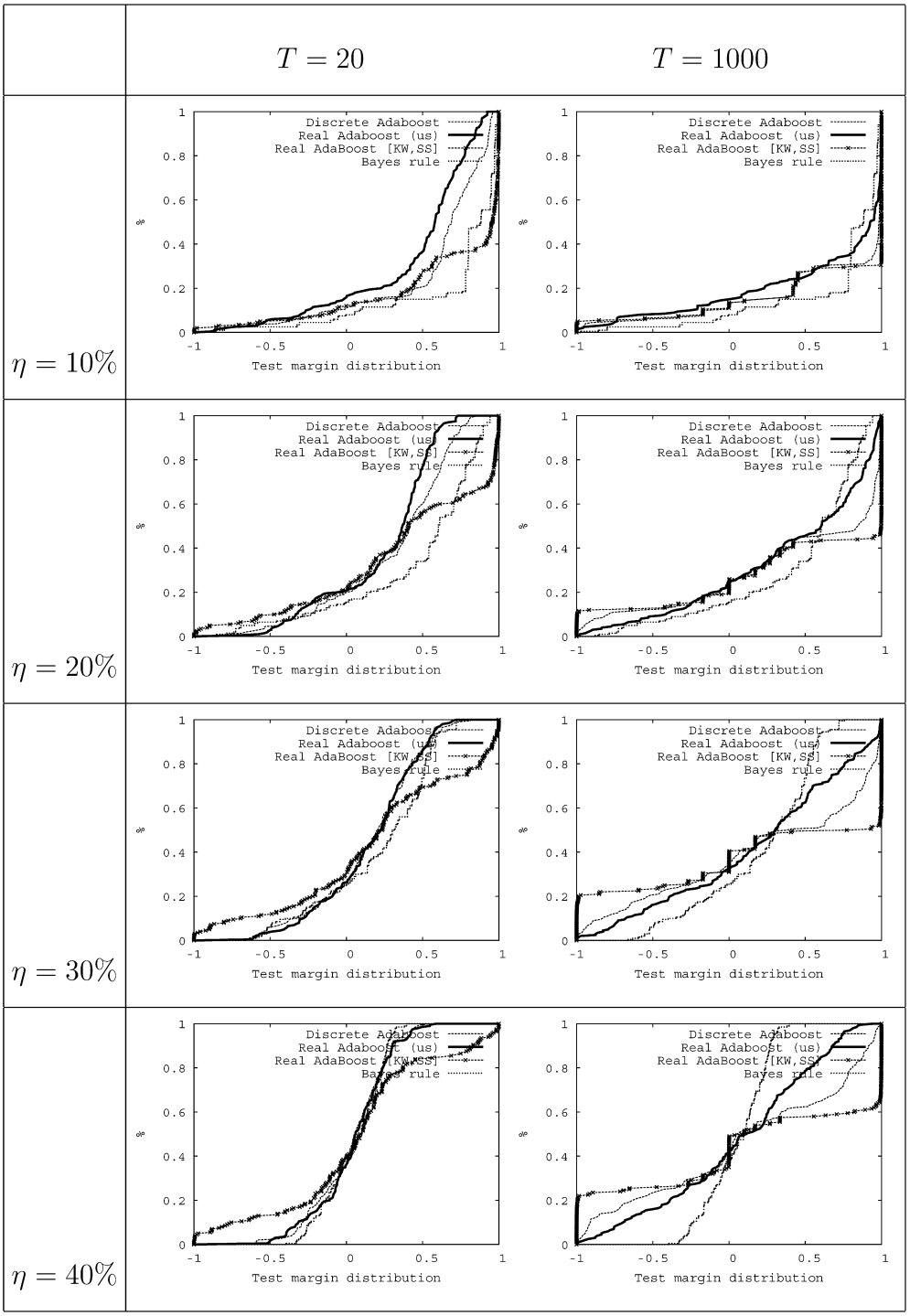

Fig. 5. Test margin distribution graphs on domain LEDeven, with different attribute noise rates ($r = 6$, see text for details).

to overfit while trying to model these noisy data. This tendency is clearly less pronounced for AdaBoost$_{\mathbb{R}}$, and if we look at the margin distribution graphs for $\theta \leqslant 0$, the fact that AdaBoost$_{\mathbb{R}}$'s curve are almost systematically below both others tends to indicate that AdaBoost$_{\mathbb{R}}$ performs indeed sensibly better than both discrete and real AdaBoosts (see Section 3.3.1). It is worthwhile remarking that Theorem 1 provides a rough primer for the appearance of these plateau

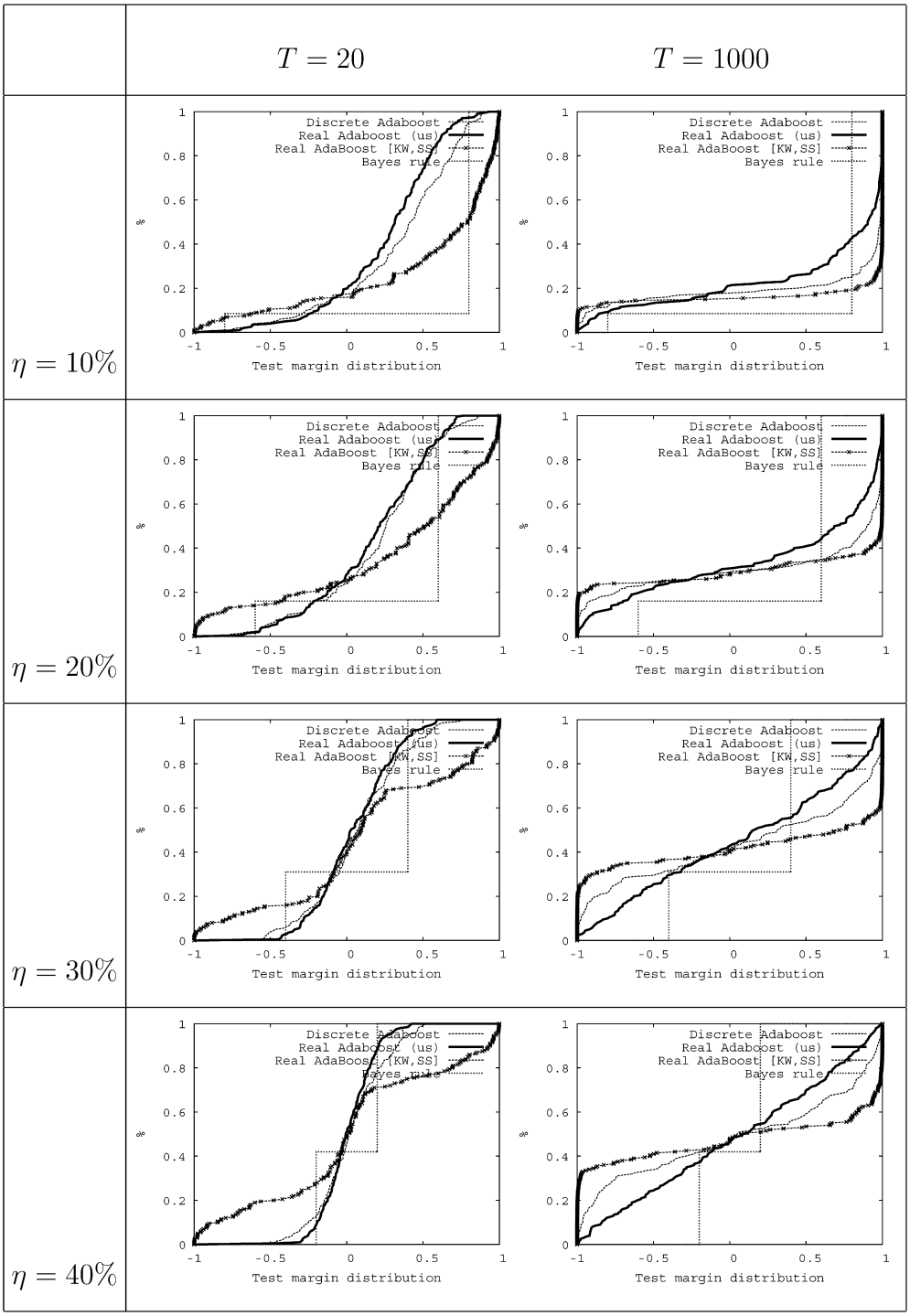

Fig. 6. Test margin distribution graphs on domain XD6, with different class noise rates ($r = 6$, see text for details).

shapes even on testing, as when m increases, $\epsilon_{\boldsymbol{u},H^*}$ converges towards ϵ_{D,H^*}, so that Theorem 1, combined with the results of [28], brings that up to statistical penalties, there should still be a Bayes plateau that upperbounds the test margin errors. However, the results of Theorem 1 apply to all algorithms: discrete, real AdaBoosts, and AdaBoost$_{\mathbb{R}}$. The fact that the exponential rates of convergence known for the empirical risk are the same for all algorithms indicates

that real AdaBoost might converge much faster in practice. However, this might not be systematic: worst-case results for the top-down induction of decision trees [23] indicate that this exponential rate of convergence might well be the best possible. Tests with AdaBoost$_{\mathbb{R}}$ for very large values of T (typically, tens or hundreds of thousands) tend to make appear the plateau shapes, but this is obtained for an iteration regime outside most that would be used on common datasets, even noisy or hard. It seems thus reasonable to think that the gentler updates of AdaBoost$_{\mathbb{R}}$ are indeed the keys to beating the other AdaBoosts on such domains.

5. Conclusion

In this paper, we have proposed a new generalization of discrete AdaBoost to handle weak hypotheses with real values. Our algorithm, AdaBoost$_{\mathbb{R}}$, departs from usual generalizations as it does not rely explicitly on the exact minimization of the exponential loss, a loss that upperbounds the empirical risk. While we formally prove that our generalization is a boosting algorithm in the original sense, it provides interesting computational and numerical features with respect to former real extensions of discrete AdaBoost, as well as a generalization of well-known facts about discrete boosting. Theoretical and experimental results give insights into the way all algorithms compare with respect to each others, and give some clues that might be helpful to obtain boosting algorithms with a better handling of hard or noisy domains.

Acknowledgements

We would like to thank the reviewers for insightful comments that helped to significantly improve the quality of the paper. R. Nock gratefully thanks Sony Computer Science Laboratories Inc., Tokyo, for a visiting grant during which part of this work was done.

References

[1] A. Banerjee, S. Merugu, I. Dhillon, J. Ghosh, Clustering with Bregman divergences, in: Proc. of the SIAM International Conference on Data Mining, 2004.
[2] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning 36 (1999) 105–139.
[3] C.L. Blake, E. Keogh, C. Merz, UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
[4] L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer, 1996.
[5] C. Domingo, O. Watanabe, MadaBoost: A modification of AdaBoost, in: Proc. of the 13th International Conference on Computational Learning Theory, 2000.
[6] W. Fan, S.-J. Stolfo, J. Zhang, P.-K. Chan, AdaCost: Misclassification cost-sensitive boosting, in: Proc. of the 16th International Conference on Machine Learning, 1999.
[7] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1997) 119–139.
[8] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: A statistical view of boosting, Annals of Statistics 28 (2000) 337–374.
[9] C. Huang, H. Ai, Y. Li, S. Lao, Vector boosting for rotation invariant multi-view face detection, in: Proc. of the 11th IEEE International Conference on Computer Vision, 2005.
[10] C. Huang, B. Wu, H. Ai, S. Lao, Omni-directional face detection based on real AdaBoost, in: Proc. of the 10th IEEE International Conference on Image Processing, 2004.
[11] R. Huang, J.-H.-L. Hansen, Dialect/accent classification via boosted word modeling, in: Proc. of the 30th IEEE International Conference on Acoustics, Speech and Signal Processing, 2005.
[12] J.-C. Janodet, R. Nock, M. Sebban, H.-M. Suchier, Boosting grammatical inference with confidence oracles, in: Proc. of the 21st International Conference on Machine Learning, 2004.
[13] G.H. John, R. Kohavi, K. Pfleger, Irrelevant features and the subset selection problem, in: Proc. of the 11th International Conference on Machine Learning, 1994.
[14] M. Kearns, Thoughts on hypothesis boosting, ML class project, 1988.
[15] M. Kearns, L. Valiant, Cryptographic limitations on learning boolean formulae and finite automata, in: Proc. of the 21st ACM Symposium on the Theory of Computing, 1989.
[16] M.J. Kearns, U.V. Vazirani, An Introduction to Computational Learning Theory, MIT Press, 1994.
[17] J. Kivinen, M. Warmuth, Boosting as entropy projection, in: Proc. of the 12th Int. Conf. on Comp. Learning Theory, 1999.
[18] G. Lebanon, J. Lafferty, Boosting and maximum likelihood for exponential models, in: Advances in Neural Information Processing Systems 14, 2001.
[19] J. Leskovec, J. Shawe-Taylor, Linear programming boosting for uneven datasets, in: Proc. of the 20th International Conference on Machine Learning, 2003.

[20] K. Matsushita, Decision rule, based on distance, for the classification problem, Annals of the Institute of Statistical Mathematics 8 (1956) 67–77.
[21] R. Nishii, S. Eguchi, Supervised image classification by contextual AdaBoost based on posteriors in neighborhoods, IEEE Transactions on Geoscience and Remote Sensing 43 (2005) 2547–2554.
[22] R. Nock, Inducing interpretable Voting classifiers without trading accuracy for simplicity: Theoretical results, approximation algorithms, and experiments, Journal of Artificial Intelligence Research 17 (2002) 137–170.
[23] R. Nock, F. Nielsen, On domain-partitioning induction criteria: Worst-case bounds for the worst-case based, Theoretical Computer Science 321 (2004) 371–382.
[24] R. Nock, F. Nielsen, On weighting clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 1223–1235.
[25] B. Popescu, J.H. Friedman, Predictive learning via rule ensembles, Tech. Report, Stanford University, 2005.
[26] G. Ridgeway, The state of boosting, Computing Science and Statistics 31 (1999) 172–181.
[27] R.E. Schapire, The strength of weak learnability, Machine Learning (1990) 197–227.
[28] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: A new explanation for the effectiveness of voting methods, Annals of Statistics 26 (1998) 1651–1686.
[29] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (1999) 297–336.
[30] L.G. Valiant, A theory of the learnable, Communications of the ACM 27 (1984) 1134–1142.
[31] V. Vapnik, Statistical Learning Theory, John Wiley, 1998.
[32] P. Viola, M.-J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2004) 137–154.
[33] P. Viola, M.-J. Jones, D. Snow, Detecting pedestrians using patterns of motion appearance, International Journal of Computer Vision 63 (2005) 153–161.
[34] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation, Morgan Kaufmann, 1999.
[35] Z. Yang, M. Li, H. Ai, An experimental study on automatic face gender classification, in: Proc. of the 18th International Conference on Pattern Recognition, 2006.