A $\mathbb{R}$eal generalization of discrete AdaBoost

Richard Nock[1] and **Frank Nielsen**[2]

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; 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. Our generalization to real values makes that decreasing weights affect only the examples on which the hypothesis' *margin* exceeds its average margin. Thus, while both properties coincide on the discrete case, 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 distributions, and it provides a nice handling of those noisy domains that represent Achilles' heel for common Adaptive Boosting algorithms.

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 to arbitrary high levels [9, 10]. (discrete) *Adaboost*, undoubtfully the most popular provable boosting algorithm [5], 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 [6] to handle real-valued weak hypotheses as well [6, 11, 15, 18]. Even when only few of them are true generalizations of discrete AdaBoost [11, 18], virtually all share a strong background in convex 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 [10, 9, 16], 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 [6, 11, 18]. 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 [11, 18]. 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 [11, 18] face when the weak hypotheses reach perfect, or perfectly wrong, classification.

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*: lower reweighting occurs **only** for examples on which the classifier's *margin* exceeds its average margin. Only on the discrete prediction framework do these two properties coincide. Furthermore, this scaling property does not not hold for previous real AdaBoosts [6, 11, 15, 18]. For some reasons, our scaling smoothes predictions, and it 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 [1].

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

Our framework is rooted into the original weak/strong learning and boosting frameworks, and Valiant's PAC model of learnability [5, 10, 19]. 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

[1] Université Antilles-Guyane, Département Scientifique Interfacultaire, Campus de Schoelcher, B.P. 7209, 97275 Schoelcher, Martinique, France. E-mail: rnock@martinique.univ-ag.fr

[2] Sony Computer Science Laboratories Inc., 3-14-13 Higashi Gotanda, Shinagawa-Ku, Tokyo 141-0022, Japan. E-mail: Frank.Nielsen@csl.sony.co.jp

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 frameworks exist that allow its extension to multiclass, multilabel frameworks [18]. 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 $\{\boldsymbol{x}_i : i = 1, 2, ...\}$ for vector sets, and $\{x_i : i = 1, 2, ...\}$ 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* $H : \mathcal{X} \to \mathbb{S}$, 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 examples, and returns a classifier or *hypothesis* $H : \mathcal{X} \to \mathbb{S}$ such that with probability $\geq 1 - \delta$, its true risk is bounded as follows:

$$\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[\textit{sign}(H(\boldsymbol{x})) \neq y] = \epsilon_{\mathcal{D},H} \quad \leq \quad \varepsilon \ . \tag{1}$$

Here, $\textit{sign}(a)$ is $+1$ iff $a \geq 0$, 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 [19, 10] also mention dependences on concepts that label the examples. Examples are indeed supposed to be labeled by a so-called *target* concept, 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 the notable exception that (1) is only required to hold with $\varepsilon = 1/2 - \gamma$ for some $\gamma > 0$ a constant or inverse polynomial in relevant parameters, and this still has to be verified regardless of D. Since predicting the classes at random, such as with an unbiased coin, would yield $\mathbf{Pr}_{(\boldsymbol{x},y)\sim D}[\textit{sign}(\mathrm{random}(\boldsymbol{x})) \neq y] = 1/2, \forall D$, 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 [9, 16]. Simple simulation arguments of WL [8] 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 [16], 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 [5] has pioneered the field of easily implementable boosting algorithms, for which $\mathbb{S} = \{-1, +1\}$. After [5], we refer to it as *discrete* AdaBoost. Basically, AdaBoost uses a weak learner as a subprocedure and an initial distribution $\boldsymbol{w}_1$ over $\mathcal{S}$ which is repeatedly skewed towards the hardest to classify examples. After T rounds

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 $t = 1, 2, ..., T$ **do**
  Get $(h_t : \mathcal{X} \to \mathbb{S}) \leftarrow WL(\mathcal{S}, \boldsymbol{w}_t)$;
  Find $\alpha_t \in \mathbb{R}$;
  Update: $\forall 1 \leq i \leq m$,

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

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

Algorithm 1: An abstraction of AdaBoost

of boosting, its output, H_T, is a linear combination of the weak hypotheses. Algorithm 1 gives a useful abstraction of AdaBoost, in which $\boldsymbol{u}$ is the uniform distribution, 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 $t \geq 1$. In discrete Adaboost, we would have:

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

where "ln" denotes the natural logarithm. The two key steps in AdaBoost are the choice of α_t and the weight update rule. They are strongly related and follow naturally if we seek to minimize the following observed *exponential loss* [6, 18] through the induction of H_T:

$$\epsilon^{\exp}_{\boldsymbol{w}_1,H_T} \quad = \quad E_{(\boldsymbol{x},y)\sim \boldsymbol{w}_1}(\exp(-yH_T(\boldsymbol{x}))) \ , \tag{4}$$

with $E_{\cdot}$ the mathematical expectation. Since $I[\textit{sign}(H_T(\boldsymbol{x})) \neq y] \leq \exp(-yH_T(\boldsymbol{x}))$ (with I the indicator function), $\epsilon_{\boldsymbol{w}_1,H_T} \leq \epsilon^{\exp}_{\boldsymbol{w}_1,H_T}$, and so minimizing the exponential loss amounts to minimizing the empirical risk as well [6, 11, 15, 17], and it turns out that it brings a boosting algorithm as well [5, 17]. 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 [6]. Its stagewise minimization brings both the weight update in (2), and the following choice for α_t:

$$\alpha_t \quad = \quad \arg\min_{\alpha\in\mathbb{R}} E_{(\boldsymbol{x},y)\sim \boldsymbol{w}_t}(\exp\left(-\alpha y h_t(\boldsymbol{x})\right)) \tag{5}$$

$$\quad = \quad \arg\min_{\alpha\in\mathbb{R}} Z_t \ . \tag{6}$$

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 *and* a real magnitude which quantifies a "confidence" in the label given. Large margin classification can bring very fast true risk minimization [17]. Margins justify to scale $\mathbb{S} = \{-1, +1\}$ to an interval such as $[-1, +1]$ [6, 18]; 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 [6, 5]. Relaxing $\mathbb{S}$ to $[-1, +1]$ yields *real* AdaBoost [6, 11, 18]. Unfortunately, (5) does *not* have a closed-form solution in this case [18]. Iterative techniques exist for its fast approximation [13], 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.

Many other extensions have been proposed to scale $\mathbb{S} = \{-1,+1\}$ up to $[-1,+1]$ or even $\mathbb{R}$ in AdaBoost, but virtually very few of those that do not generalize discrete AdaBoost have been proven so far to be boosting algorithms in the original sense. For space considerations, we focus our comparison only on very few of them that bear at least close similarities with true real generalizations of discrete AdaBoost [6, 11, 15, 18]. These algorithms, that fit to Algorithm 1, mainly differ on the choice of the leveraging coefficient α_t. For example, [6, 15] pick $\alpha_t = 1$, a choice for which the boosting property is not shown to hold.

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 $\mathbb{R}$ itself; suppose we replace the weight update (2) and eq. (5) by what follows:

$$w_{t+1,i} \leftarrow w_{t,i} \times \left(\frac{1 - (\mu_t y_i h_t(\boldsymbol{x}_i)/h_t^\star)}{1-\mu_t^2} \right) . \quad (7)$$

$$\alpha_t = \frac{1}{2h_t^\star} \ln \frac{1+\mu_t}{1-\mu_t} . \quad (8)$$

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

$$\mu_t = \frac{1}{h_t^\star} \sum_{i=1}^m w_{t,i} y_i h_t(\boldsymbol{x}_i) \in [-1,+1] \quad (9)$$

the normalized *margin* of h_t over $\mathcal{S}$. For the sake of clarity, we suppose in the following subsection that $\forall 1 \leq t \leq T, h_t^\star < \infty$. 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 [18], or we code it as ∞, making $\alpha_t = 0$ and $\mu_t = \sum_{i:|h_t(\boldsymbol{x}_i)|=\infty} w_{t,i} \textit{sign}(y_i h_t(\boldsymbol{x}_i))$. 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.

3.1 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. (8) 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:

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

i.e.:

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

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:

$$\begin{aligned} \nu_T((\boldsymbol{x},y)) &= \tanh(y H_T(\boldsymbol{x})/2) \\ &= \frac{\exp(y H_T(\boldsymbol{x})) - 1}{\exp(y H_T(\boldsymbol{x})) + 1} \in [-1,+1] . \end{aligned} \quad (10)$$

This definition of margin extends a previous one for discrete AdaBoost [21], and its choice is discussed in Section 3.2. $\forall \theta \in [-1,+1]$ we also define the classifier's "margin error" as the proportion of examples whose margin does not exceed θ:

$$\nu_{\boldsymbol{u},H_T,\theta} = \frac{1}{m} \sum_{i=1}^m I[\nu_T((\boldsymbol{x}_i,y_i)) \leq \theta] . \quad (11)$$

Clearly, $\epsilon_{\boldsymbol{u},H_T} = \nu_{\boldsymbol{u},H_T,0}$, and $\nu_{\boldsymbol{u},H_T,\theta}$ generalizes $\epsilon_{\boldsymbol{u},H_T}$. We now prove a first Theorem on AdaBoost$_{\mathbb{R}}$.

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

$$\nu_{\boldsymbol{u},H_T,\theta} \leq \left(\frac{1+\theta}{1-\theta} \right) \times \exp\left(-\frac{1}{2} \sum_{t=1}^T \mu_t^2 \right) . \quad (12)$$

Proof: We need the following simple Lemma.

Lemma 2 $\forall a \in [-1,1], \forall b \in [-1,1]$,

$$1 - ab \geq \sqrt{1-a^2} \exp\left(-\frac{b}{2} \ln \frac{1+a}{1-a} \right) .$$

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 $1 \leq t \leq T$. 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:

$$\begin{aligned} & 1 - (\mu_t y_i h_t(\boldsymbol{x}_i)/h_t^\star) \\ & \geq \sqrt{1-\mu_t^2} \exp\left(-\frac{y_i h_t(\boldsymbol{x}_i)}{2h_t^\star} \ln \frac{1+\mu_t}{1-\mu_t} \right) . \end{aligned} \quad (13)$$

Unraveling the weight update rule, we obtain:

$$\begin{aligned} & w_{T+1,i} \times \prod_{t=1}^T (1-\mu_t^2) \\ & = u_i \times \prod_{t=1}^T (1 - (\mu_t y_i h_t(\boldsymbol{x}_i)/h_t^\star)) . \end{aligned} \quad (14)$$

Using T times (13) on the right-hand side of (14) and simplifying yields:

$$(w_{T+1,i}/u_i) \times \prod_{t=1}^T \sqrt{1-\mu_t^2} \geq \exp\left(-y_i H_T(\boldsymbol{x}_i)\right) . \quad (15)$$

Since for any real q, $I[q \leq 0] \leq \exp(-q)$, we have:

$$\begin{aligned} I\left[\nu_T((\boldsymbol{x}_i,y_i)) \leq \theta\right] &= I\left[y_i H_T(\boldsymbol{x}_i) - \ln \frac{1+\theta}{1-\theta} \leq 0 \right] \\ &\leq \exp\left(-y_i H_T(\boldsymbol{x}_i) + \ln \frac{1+\theta}{1-\theta} \right) \\ &= \left(\frac{1+\theta}{1-\theta} \right) \times \exp(-y_i H_T(\boldsymbol{x}_i)) \\ &\leq \frac{w_{T+1,i}}{u_i} \left(\frac{1+\theta}{1-\theta} \right) \times \prod_{t=1}^T \sqrt{1-\mu_t^2} , \end{aligned}$$

Where the last line uses ineq. (15). There only remains to sum that last ineq. for all examples of $\mathcal{S}$, and use the fact that $\sqrt{1-a^2} \leq$

$\exp(-a^2/2), \forall a \in [-1,1]$. Given that $\boldsymbol{w}_{T+1}$ is a distribution and $u_i = 1/m$, we immediately obtain the statement of the Theorem. □

Theorem 1 generalizes a well-known convergence Theorem for AdaBoost's empirical risk [18] ($\theta = 0$). 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 as given by (11). 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 [9, 10, 18]: 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 ($\forall t \geq 1$):

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

This, in addition, provides us with a generalization of the discrete WLA [9, 16], since we have in this case $\mu_t = 1 - 2\epsilon_{\boldsymbol{w}_t,h_t}$. This brings that either $\epsilon_{\boldsymbol{w}_t,h_t} \leq (1/2) - \gamma/2$, or $\epsilon_{\boldsymbol{w}_t,h_t} \geq (1/2) + \gamma/2$. 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 [6].

Now proving that AdaBoost$_{\mathbb{R}}$ is a boosting algorithm amounts first to using Theorem 1 with $\theta = 0$, to obtain under the WLA that after T iterations of AdaBoost$_{\mathbb{R}}$, we have $\epsilon_{\boldsymbol{u},H_T} \leq \exp(-T\gamma^2/2)$. Thus, if we run AdaBoost$_{\mathbb{R}}$ for $T = \Omega((1/\gamma^2)\ln m)$, 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 [8, 20] immediately bring the following Theorem.

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

3.2 Discussion

Perhaps one of the most important difference with discrete AdaBoost and its numerous offsprings [5, 6, 4, 18] lies in the fact that they tend to reweight lower the examples that have received the right class. 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 margin exceeds its average margin (when $\mu_t > 0$):

$$y_i h_t(\boldsymbol{x}_i)/h_t^\star \quad \geq \quad \mu_t \ . \tag{16}$$

Thus, there can be examples that receive the right class by h_t, and 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.

Margins. It may be helpful to compare our definition of margin with those recently used for real extensions of AdaBoost [6, 17, 18]. Eq. (10) is generally replaced by:

$$\nu_T((\boldsymbol{x},y)) \quad = \quad \frac{yH_T(\boldsymbol{x})}{\sum_{t=1}^T \alpha_t} \in [-1,+1] \ . \tag{17}$$

Eq. (17) is almost equivalent to the margin of a classifier that we have used for weak hypotheses (μ_t). However, eq. (10), which defines the margin of an *example*, appears to be more convenient for three reasons. The first is statistical, as our definition is in direct relationship with additive logistic models [6]. Suppose we fit H_T to the additive logistic model :

$$H_T(\boldsymbol{x}) \quad = \quad \ln \frac{p[y=+1|\boldsymbol{x}]}{p[y=-1|\boldsymbol{x}]} \ , \tag{18}$$

with the probabilities $p[.|\boldsymbol{x}]$ to be estimated while learning. This brings for (10):

$$\nu_T((\boldsymbol{x},y)) \quad = \quad y(2p[y=+1|\boldsymbol{x}] - 1) \ . \tag{19}$$

The quantity $2p[y=+1|\boldsymbol{x}] - 1 = p[y=+1|\boldsymbol{x}] - p[y=-1|\boldsymbol{x}]$ which replaces $H_T(\boldsymbol{x})/\sum_{t=1}^T \alpha_t = \ln(p[y=+1|\boldsymbol{x}]/p[y=-1|\boldsymbol{x}])/\sum_{t=1}^T \alpha_t$ in (17) is the *gentle* logistic approximation of [6], a quantity which is much more stable than the full logistic model itself. The second reason is more technical. Theorem 1 shows that $\nu_{\boldsymbol{u},H_T,\theta}$ vanishes under the WLA regardless of the value of $\theta \in (-1,1)$ with margin definition (10). Upperbounds for $\nu_{\boldsymbol{u},H_T,\theta}$ with eq. (17) are not as easy to read, as all require to vanish that θ be smaller than fluctuating upperbounds that can be $\ll 1$ [17, 18]. It does not seem that it is the boosting algorithm which is responsible, as in our case, using (17) would *not* yield a vanishing $\nu_{\boldsymbol{u},H_T,\theta}$ when $\theta \geq \max_t |\mu_t|/2$, a situation identical to previous analyses [5, 17, 18]. The third reason is an experimental consequence of the second. Definition (10) makes cumulative margin distributions easier to read, since there is no fluctuating theoretical upperbound < 1 for "boostable" margins.

Computations and numerical stability. A first difference with previous generalizations of discrete AdaBoosts that do not fix *ad hoc* values for α_t [6, 11, 18] 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 [6, 11, 18].

The extensions of discrete AdaBoost [6, 11, 18] 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. (7) 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 [6, 11, 18], 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 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.

Figure 1. Estimated true risks on 25 domains, comparing discrete AdaBoost [5] (D), AdaBoost$_{\mathbb{R}}$ (U) and the real AdaBoost of [6, 11, 18] (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*.

	$T = 10$			$T = 50$		
Domain	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

4 Experiments

Experiments were carried out on 25 domains, most of which come from the UCI repository [2]. 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 ran discrete AdaBoost [5], AdaBoost$_{\mathbb{R}}$ and the real AdaBoost of [11, 18]. *WL* is set to a rule (monomial) learner, with fixed maximal rule size (attribute number) r [12, 14]. 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 [11, 18]. The true α_t is approximated from (5) using a simple dichotomous search until the relative error does exceed 10^{-6}, using results of [13] to make it faster. Empirically, the execution time for real AdaBoost [6, 11, 18] is on average more than 100 times that of discrete AdaBoost and AdaBoost$_{\mathbb{R}}$.

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 $r = 2$. Figure 1 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) [2] are those 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) [2, 12]. 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, 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 [4].

Noise handling. In order to shed some more light on noise handling, we have drilled down into the results on domain LEDeven+17 [2]. Its basis is a seven bits problem that describes 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 10% chances, and seventeen attributes are added, that are irrelevant in the strongest sense [7]. Thus, there is a total of $n = 24$ description variables. Figures 2 and 3 displays cumulative margin distributions that were obtained for couple $(r, T) \in \{2, 4, 6\} \times \{10, 100, 200, 1000\}$. The training margins clearly display that the real AdaBoost of [6, 11, 18] is the fastest to converge to perfect classification, followed by discrete AdaBoost [5], and then by AdaBoost$_{\mathbb{R}}$("us"). The *testing* margins display a completely different pattern: on ten out of twelve results, the estimated true risk is the lowest for AdaBoost$_{\mathbb{R}}$, and on eight out of ten, the second best is discrete AdaBoost (with a difference best-worst sometimes reaching 6%). Real AdaBoost [6, 11, 18] beats discrete AdaBoost when the classifiers built get complicated: $T \geq 200$ and $r \in \{4, 6\}$. The clear symmetric sigmoid shape (plateau) observed on testing for real AdaBoost on these cases (which is also observable —though less pregnant— 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 those receiving the wrong class. This, we think, indicates a tendency 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 curves for $\theta \leq 0$, the fact that AdaBoost$_{\mathbb{R}}$'s curve are almost systematically below both others tends to indicate that AdaBoost$_{\mathbb{R}}$ performs sensibly better than both discrete and real AdaBoosts. This is in accordance with the fact that plotting Bayes rule's margin curves, following the prediction of the logistic model in (18), seems to always yield a curve shape closer to AdaBoost$_{\mathbb{R}}$. To see this, Figure 4 plots cumulative margin distributions for Bayes rule's, for varying amounts of attribute noise in LEDeven+17, ranging from 5% to 50% by steps of 5%. Curves were

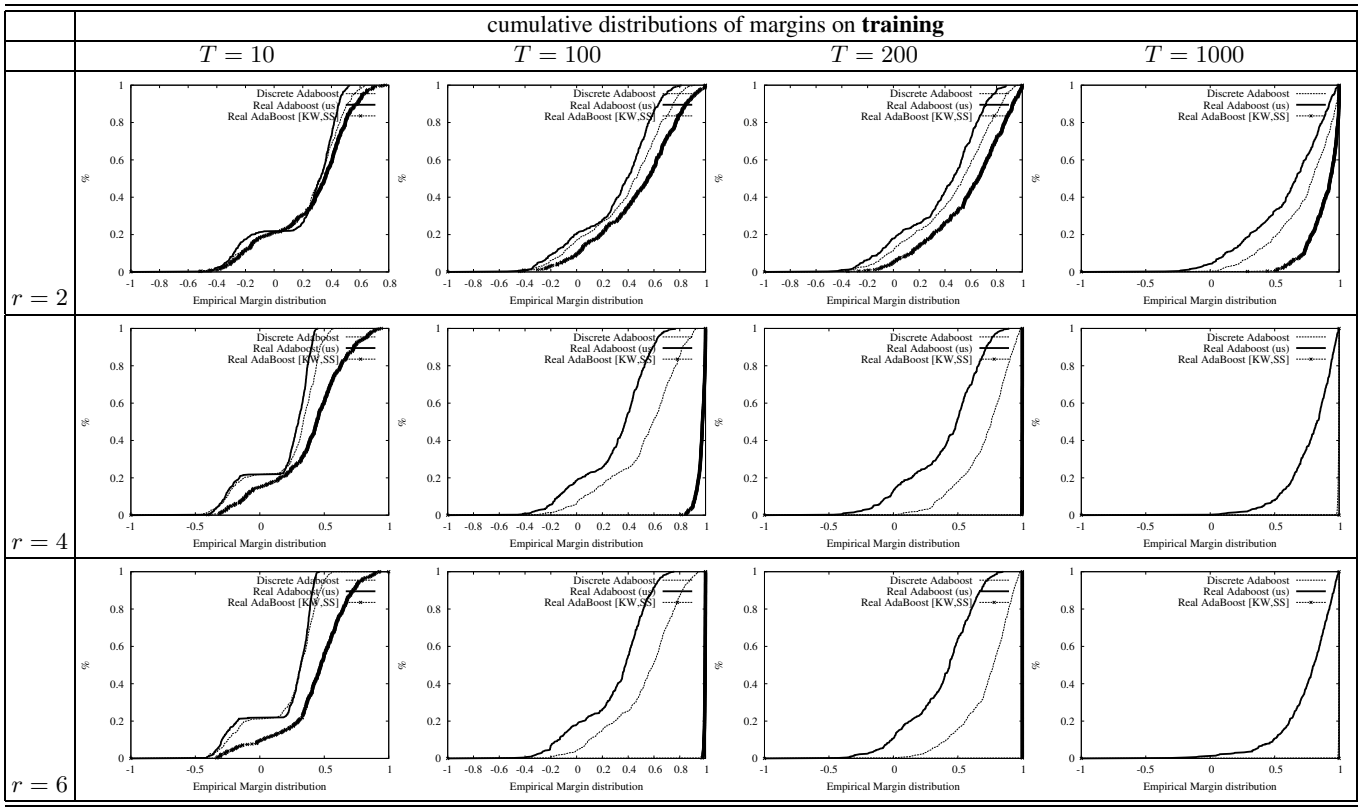

Figure 2. Empirical margin distributions on domain LEDeven+17. An algorithm is as better than another one as its margin distribution is located below the other for negative margin values. The empirical risk of an algorithm is approximately the intersection between its margin distribution and the line $x = 0$.

generated by applying the logistic prediction to random samples containing 300 examples each. It is clear from these curves that the shape that exhibits real AdaBoost as T or r increase, with many examples badly classified with very large confidence, is absent from the logistic prediction. Bayes rule's, along with AdaBoost$_{\mathbb{R}}$, seem to be more conservative in their predictions. Since the logistic model is fit by discrete and real AdaBoosts [6], this suggests that real AdaBoost, followed by discrete AdaBoost, seem to rapidly overfit the model.

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. Among future works, we plan to investigate further the boosting model of learning, and boosting algorithms in a somehow stronger approach. The classical weak and strong learning frameworks are discrete frameworks, as the accuracy of a classifier solely relies on classes. When dealing with real-valued predictions, an accurate framework should take into account both the sign (class), *and* the magnitude (confidence) of the prediction, as a good classifier should basically obtain large confidences with right classes, and not simply right classes. The margin error definition in (11) could be accurate to capture both notions, in a model that would basically replace $\epsilon_{\mathcal{D},H}$ in (1) by $\nu_{D,H,\theta}$ in (12), with $0 < \theta < 1$ user-fixed, like ε and δ. We believe that strong results are possible: indeed, Theorem 1 shows that all margin errors at fixed $\theta < 1$ vanish with T under the WLA. It should thus be possible to obtain not only classifiers whose true risk is as reduced as desired, *but also* whose confidences are as large as desired. Finally, Theorem 1 does not integrate (the empirical) Bayes risk. A careful integration of this risk, or directly the margin error of Bayes rule, should help to see the way the algorithm behaves on hard problems, and perhaps the way it converges to Bayes rule [3].

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for insightful comments on the paper. R. Nock would like to warmly thank Sony Computer Science Laboratories Inc., Tokyo, for a visiting grant during which part of this work was done.

REFERENCES

[1] E. Bauer and R. Kohavi, 'An empirical comparison of voting classification algorithms: Bagging, boosting, and variants', *Machine Learning*, **36**, 105–139, (1999).

[2] C. L. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[3] P. Bühlmann and B. Yu, 'Boosting with the L_2 loss: regression and classification', *Journal of the American Statistical Association*, **98**, 324–339, (2003).

[4] C. Domingo and O. Watanabe, 'MadaBoost: a Modification of AdaBoost', in *Proc. of the 13 th International Conference on Computational Learning Theory*, pp. 180–189, (2000).

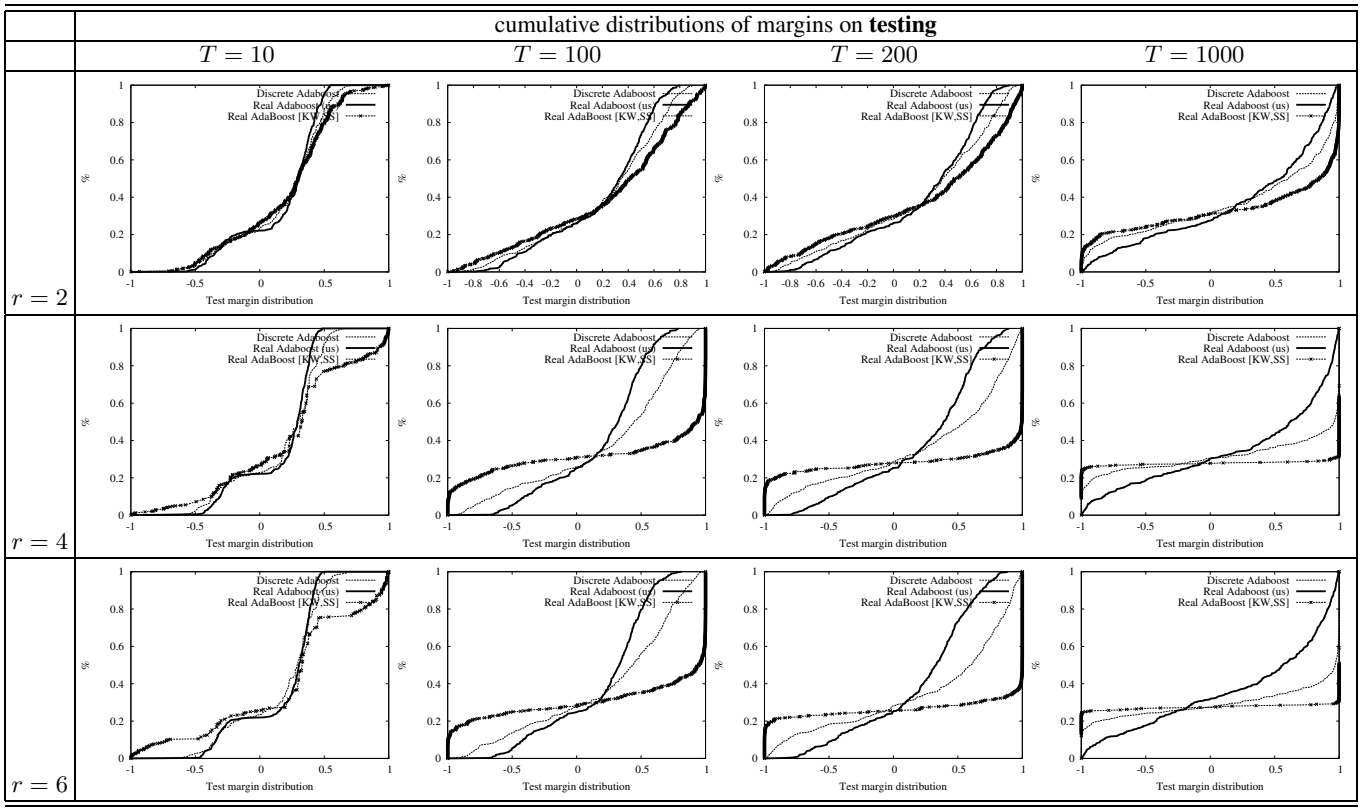

Figure 3. Test margin distributions on domain LEDeven+17. An algorithm is as better than another one as its margin distribution is located below the other for negative margin values. The estimated true risk of an algorithm is approximately the intersection between its margin distribution and the line $x = 0$.

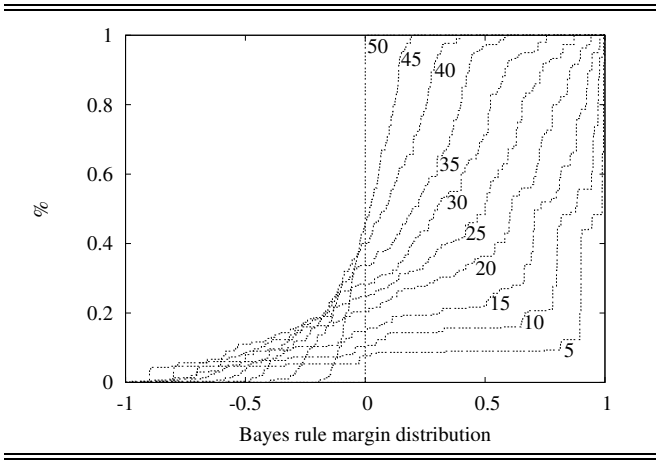

Figure 4. Cumulative distributions of margins for Bayes rule's prediction with the logistic model, on LEDeven+17 with varying amounts of attribute noise, in the set $\{5\%, 10\%, 15\%, 20\%, 25\%, 30\%, 35\%, 40\%, 45\%, 50\%\}$ (see text for details).

[5] Y. Freund and R. E. Schapire, 'A Decision-Theoretic generalization of on-line learning and an application to Boosting', *Journal of Computer and System Sciences*, **55**, 119–139, (1997).

[6] J. Friedman, T. Hastie, and R. Tibshirani, 'Additive Logistic Regression : a Statistical View of Boosting', *Annals of Statistics*, **28**, 337–374, (2000).

[7] G. H. John, R. Kohavi, and K. Pfleger, 'Irrelevant features and the subset selection problem', in *Proc. of the 11 th International Conference on Machine Learning*, pp. 121–129, (1994).

[8] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*, M.I.T. Press, 1994.

[9] M.J. Kearns. Thoughts on hypothesis boosting, 1988. ML class project.

[10] M.J. Kearns and L. Valiant, 'Cryptographic limitations on learning boolean formulae and finite automata', *Proc. of the 21 th ACM Symposium on the Theory of Computing*, 433–444, (1989).

[11] J. Kivinen and M. Warmuth, 'Boosting as entropy projection', in *Proc. of the 12 th Int. Conf. on Comp. Learning Theory*, pp. 134–144, (1999).

[12] R. Nock, 'Inducing interpretable Voting classifiers without trading accuracy for simplicity: theoretical results, approximation algorithms, and experiments', *Journal of Artificial Intelligence Research*, **17**, 137–170, (2002).

[13] R. Nock and F. Nielsen, 'On weighting clustering', *IEEE Transasctions on Pattern Analysis and Machine Intelligence*, (2006). to appear.

[14] B. Popescu and J. H. Friedman, 'Predictive learning via rule ensembles', Technical report, Stanford University, (2005).

[15] G. Ridgeway, 'The state of Boosting', *Computing Science and Statistics*, **31**, 172–181, (1999).

[16] R. E. Schapire, 'The strength of weak learnability', *Machine Learning*, 197–227, (1990).

[17] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, 'Boosting the margin : a new explanation for the effectiveness of voting methods', *Annals of statistics*, **26**, 1651–1686, (1998).

[18] R. E. Schapire and Y. Singer, 'Improved boosting algorithms using confidence-rated predictions', *Machine Learning*, **37**, 297–336, (1999).

[19] L. G. Valiant, 'A theory of the learnable', *Communications of the ACM*, **27**, 1134–1142, (1984).

[20] V. Vapnik, *Statistical Learning Theory*, John Wiley, 1998.

[21] I. Witten and E. Frank, *Data Mining: practical Machine Learning tools and techniques with Java implementation*, Morgan Kaufmann, 1999.