$\mathbb{R}$eal Boosting *a la Carte* with an Application to Boosting Oblique Decision Trees

Claudia Henry **Richard Nock**
U. Antilles-Guyane / CEREGMIA
97275 Schoelcher, Martinique, France
{chenry,rnock}@martinique.univ-ag.fr

Frank Nielsen
SONY CS Labs Inc.
Tokyo 141-0022, Japan
Nielsen@csl.sony.co.jp

Abstract

In the past ten years, boosting has become a major field of machine learning and classification. This paper brings contributions to its theory and algorithms. We first unify a well-known top-down decision tree induction algorithm due to [Kearns and Mansour, 1999], and discrete AdaBoost [Freund and Schapire, 1997], as two versions of a same higher-level boosting algorithm. It may be used as the basic building block to devise simple provable boosting algorithms for complex classifiers. We provide one example: the first boosting algorithm for Oblique Decision Trees, an algorithm which turns out to be simpler, faster and significantly more accurate than previous approaches.

1 Introduction

Loosely speaking, a *boosting* algorithm repeatedly trains moderately accurate learners from which it gets its *weak* hypotheses, and combines them to output a *strong* classifier which boosts the accuracy to arbitrary high levels [Kearns and Valiant, 1989]. A pioneering paper [Schapire, 1990] proved the existence of such boosting algorithms, and another one drew the roots of their most popular representative: AdaBoost, that builds a linear combination of the predictions of the weak hypotheses [Freund and Schapire, 1997]. Another paper due to [Kearns and Mansour, 1999] proved that some of the most popular top-down decision tree induction algorithms are also boosting algorithms in disguise [Breiman *et al.*, 1984; Quinlan, 1993]. These two kinds of algorithms are outwardly different from each other: while AdaBoost repeatedly modifies *weights* on the training examples and directly minimizes a convex *exponential loss*, top-down decision tree induction algorithms do not modify weights on the examples, and they minimize the expectation of a concave, so-called *permissible* function [Kearns and Mansour, 1999].

This explains why the starting point of this paper is a quite surprising result, as we prove that these two kinds of algorithms are in fact the *same* algorithm, whose induction is performed on two different classifier "graphs". The apparent perceptual differences in the algorithms stem only from different structural properties of the classifiers. This suggests a generic induction scheme which gives, for many classifiers that meet a simple assumption, a corresponding boosting algorithm. Using a recent result due to [Nock and Nielsen, 2006], this algorithm uses real-valued weak hypotheses, but it does not face the repetitive minimization of the exponential loss faced by previous $\mathbb{R}$eal boosting algorithms [Friedman *et al.*, 2000; Schapire and Singer, 1999]. When the classifier is an oblique decision tree, the algorithm obtained has three key features: it is fast, simple, and the first provable boosting algorithm that fully exploits this class. This is important, as the problem of inducing oblique decision trees has a longstanding history, and the algorithms available so far are either time consuming or complex to handle formally [Breiman *et al.*, 1984; Cantú-Paz and Kamath, 2003; Heath *et al.*, 1993; Murthy *et al.*, 1994; Shawe-Taylor and Cristianini, 1998]. The following Section presents definitions; it is followed by a Section on our theoretical results, and a Section that discusses them and gives our boosting algorithm for oblique decision trees; two last Sections discuss experiments, and conclude. For the sake of clarity, proofs have been postponed to an appendix.

2 Definitions and problem statement

Bold-faced variables such as $\boldsymbol{w}$ and $\boldsymbol{x}$, represent vectors. Unless otherwise stated, sets are represented by calligraphic upper-case alphabets, *e.g.* $\mathcal{X}$, and (unless explicitely stated) 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. Consider the following supervised learning setting. Let $\mathcal{X}$ denote a domain of observations of dimension n, such as $\mathbb{R}^n$, $\{0, 1\}^n$, etc. . We suppose a set $\mathcal{S}$ of $|\mathcal{S}| = m$ examples, with $|.|$ the cardinal notation, defined as $\mathcal{S} = \{s_i = \langle \boldsymbol{x}_i, y_i \rangle \in \mathcal{X} \times \{-1, +1\} : i = 1, 2, ..., m\}$, onto which a discrete distribution $\boldsymbol{w}_1$ is known. "+1" is called the positive class, and "-1" the negative class. Our primary objective is related to building an accurate classifier (or hypothesis) $H : \mathcal{X} \rightarrow \mathbb{R}$. The goodness of fit of H on $\mathcal{S}$ may be evaluated by two quantities:

$$\varepsilon(H, \boldsymbol{w}_1) = E_{\boldsymbol{w}_1}(1_{[\mathrm{sign}(H(\boldsymbol{x})) \neq y]}) \ , \quad (1)$$

$$\varepsilon_{\exp}(H, \boldsymbol{w}_1) = E_{\boldsymbol{w}_1}(\exp(-yH(\boldsymbol{x}))) \ . \quad (2)$$

Here, $\mathrm{sign}(a) = +1$ iff $a \geq 0$ and -1 otherwise, $1_{[.]}$ is the 0/1 indicator variable, and $E_{.}()$ is the expectation. (1) is the conventional empirical risk, and (2) is an upperbound, the exponential loss [Schapire and Singer, 1999]. We are interested in

```
Input: $\mathcal{S}, \boldsymbol{w}_1$;
for $t = 1, 2, ..., T$
    $h_t \leftarrow \mathrm{Tree}(\mathcal{S}, \boldsymbol{w}_t)$;
    $\alpha_t \leftarrow \arg\min_{\alpha\in\mathbb{R}} E_{\boldsymbol{w}_t}(\exp(-y\alpha h_t(\boldsymbol{x})))$;
    for $i = 1, 2, ..., m$
        $w_{t+1,i} \leftarrow w_{t,i}\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))/Z_t$;
```

AdaBoost + Trees

```
Input: $\mathcal{S}, \boldsymbol{w}_1$;
for $t = 1, 2, ..., T$
    let $\ell \in \mathrm{leaves}(H_{t-1})$ containing
            examples of both classes;
    $h_t \leftarrow \mathrm{Stump}(\mathcal{S}_\ell, \boldsymbol{w}_1)$;
    $\ell \leftarrow h_t$;
```

TopDown_DT (CART, C4.5)

```
Input: $\mathcal{S}, \boldsymbol{w}_1$;
for $t = 1, 2, ..., T$
    let $\ell \in \mathrm{leaves}(H_{t-1})$ containing
            examples of both classes;
    $h_t \leftarrow \mathrm{StumpLS}(\mathcal{S}_\ell, \boldsymbol{w}_1)$;
    $\ell \leftarrow h_t$;
```

TopDown_ODT (OC1, SADT)

Figure 1: An abstraction of 3 core greedy procedures of some popular induction algorithms (see text for references and details).

the stagewise, greedy fitting of H to $\mathcal{S}$, from scratch. A large number of classes of classifiers have popular induction algorithms whose core procedure follows this scheme, including Decision Trees (DT) [Breiman *et al.*, 1984; Quinlan, 1993], Branching Programs (BP) [Mansour and McAllester, 2002] and Linear Separators (LS) [Freund and Schapire, 1997; Nock and Nielsen, 2006]. Virtually all these procedures share another common-point: the components that are used to grow H are also classifiers. These are the DTs for AdaBoost with trees, the internal node splits for the DTs and BPs (akin to decision stumps), etc.

Now, given some fixed class of classifiers in which we fit H, here is the problem statement. A weak learner (WL) is assumed to be available, which can be queried with a sample $\mathcal{S}'$ and (discrete) distribution $\boldsymbol{w}'$ on $\mathcal{S}'$; the assumption of its existence is called a Weak Learning Assumption (WLA). It returns in polynomial time[1] a classifier h on which only the following can be assumed: $\varepsilon(h, \boldsymbol{w}') \leq 1/2 - \gamma$ for some $\gamma > 0$ [Schapire and Singer, 1999]. Given $0 < \epsilon < 1$, is it possible to build in polynomial time some H with $\varepsilon(H, \boldsymbol{w}_1) \leq \epsilon$, after having queried T times WL for classifiers $h_1, h_2, ..., h_T$, for some $T > 0$? Provided additional assumptions are made for its generalization abilities (see Section "Discussion" below), this algorithm is called a boosting algorithm [Freund and Schapire, 1997; Kearns and Valiant, 1989].

3 Unifying boosting properties

For the sake of clarity, we now plug T in subscript on H; An element of LS is a classifier H_T with $H_T(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$, where $\alpha_t \in \mathbb{R}$ is a leveraging coefficient that may be interpreted as a confidence in h_t. An element of DT is a rooted directed tree in which each internal node supports a single boolean test over some observation variable, and each leaf is labeled by a real. The classification of some observation proceeds by following, from the root, the path whose tests it satisfies, until it reaches a leaf which gives its class. Figure 2 (left) presents an example of DT ($n = 2$), where v_1, v_2 are Boolean description variables. Finally, Oblique Decision Trees (ODT) generalizes DT, as linear combinations of variables are authorized for the internal nodes, which allows splits that are oblique instead of just axis-parallel [Breiman *et al.*, 1984].

Figure 1 gives an abstract view of some of the most popular induction algorithms, or at least of their core procedure which induces a large classifier: AdaBoost with trees [Freund and Schapire, 1997; Schapire and Singer, 1999], C4.5 [Quinlan, 1993], CART [Breiman *et al.*, 1984], OC1 [Murthy *et al.*, 1994], SADT [Heath *et al.*, 1993]; the first algorithm for the induction of ODT dates back to the mid-eighties; it was a descendant of CART [Breiman *et al.*, 1984]. Most of the induction algorithms for DT or ODT integrate a pruning stage; here, we are only interested in the greedy induction scheme. WL has various forms: it induces a decision tree on the whole sample $\mathcal{S}$ and on a distribution which is repeatedly modified for AdaBoost; it induces a decision tree with a single internal node (a stump) on the subset of $\mathcal{S}$ that reaches leaf ℓ (noted $\mathcal{S}_\ell$) and on a distribution which is not modified for CART and C4.5; it replaces this ordinary, axis-parallel stump, by a linear separator stump (also called stump for the sake of simplicity) for OC1 and SADT. The choice of a split for DT and ODT relies on the repetitive minimization of an "impurity" criterion which is the expectation, over the leaves of the current tree H, of a so-called permissible function [Kearns and Mansour, 1999]:

$$\varepsilon_{\mathrm{imp}}(H, \boldsymbol{w}_1, f) \quad = \quad E_{\ell\sim\mathrm{leaves}(H)}\left(f\left(\frac{w^{+}_{1,\mathcal{S}_\ell}}{w_{1,\mathcal{S}_\ell}}\right)\right) \quad ; (3)$$

here, $w_{1,\mathcal{S}'}$ is the total weight of $\mathcal{S}'$ in $\boldsymbol{w}_1$ and $w^{+}_{.}$ is its weight further restricted to the positive class. In the expectation, the weight of a leaf ℓ is $w_{1,\mathcal{S}_\ell}$. The permissible function $f : [0, 1] \rightarrow [0, 1]$ has to be concave, symmetric around $1/2$, and with $f(0) = f(1) = 0$ and $f(1/2) = 1$. Examples of permissible functions are $f(z) = 4z(1 - z)$ for Gini index [Breiman *et al.*, 1984], $f(z) = -z\log z - (1 - z)\log(1 - z)$ for the entropy [Quinlan, 1993] ($\log$ base-2), or even $f(z) = 2\sqrt{z(1 - z)}$ for the optimal choice of [Kearns and Mansour, 1999]. Remark that we have $\varepsilon_{\mathrm{imp}}(H, \boldsymbol{w}_1, f) \geq \varepsilon_{\mathrm{imp}}(H, \boldsymbol{w}_1, 2\min\{z, 1 - z\}) = 2\varepsilon(H, \boldsymbol{w}_1)$, for any permissible f, and so minimizing (3) amount to minimizing the empirical risk of H as well. The reason why this is a *better* choice than focusing directly on the empirical risk comes from the concavity of the permissible function [Kearns and Mansour, 1999]. Though seemingly very different from each other, AdaBoost and DT induction algorithms from the CART-family were independently proven to be boosting algorithms [Freund and Schapire, 1997; Kearns and Mansour, 1999; Schapire and Singer, 1999]. So far, no such formal

[1] For the sake of simplicity, polynomial time means polynomial in the relevant parameters throughout the paper.

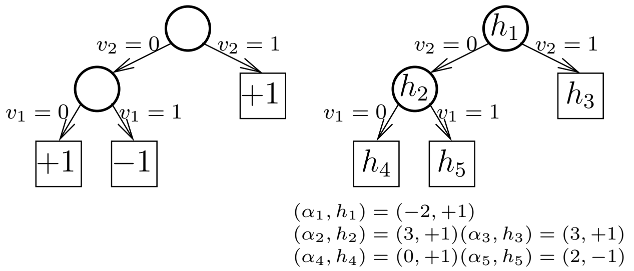
$v_2 = 0$
$v_2 = 1$
$v_1 = 0$
$v_1 = 1$
$+1$
$+1$
-1
h_1
$v_2 = 0$
$v_2 = 1$
h_2
h_3
$v_1 = 0$
$v_1 = 1$
h_4
h_5
$(\alpha_1, h_1) = (-2, +1)$
$(\alpha_2, h_2) = (3, +1)(\alpha_3, h_3) = (3, +1)$
$(\alpha_4, h_4) = (0, +1)(\alpha_5, h_5) = (2, -1)$

Figure 2: A DT with 3 leaves and 2 internal nodes (left) and an equivalent representation which fits to (**A**) (right).

boosting algorithm or proof exist for ODT that would take full advantage of their structure combining oblique stumps and decision trees. For this objective, let us shift to a more general standpoint on our problem, and address the following question: what kind of classifiers may be used to solve it ? Consider the following assumption, that H_T represents some kind of local linear classifier:

> (assumption **A**) $\forall T > 0$, denote $\mathcal{H}_T = \{h_1, h_2, ..., h_T\}$ the set of outputs of WL; then, there exists a function which maps any observation to a non-empty subset of $\mathcal{H}_T$, $g_{H_T} : \mathcal{X} \rightarrow 2^{\mathcal{H}_T} \backslash \{\emptyset\}$. $\forall \boldsymbol{x} \in \mathcal{X}$, $g_{H_T}(\boldsymbol{x})$ is required to be computable polynomially in n and the size of H_T; it describes the classification function of H_T, in the following way:
>
> $$H_T(\boldsymbol{x}) \quad = \sum_{h_t \in g_{H_T}(\boldsymbol{x})} \alpha_t h_t(\boldsymbol{x}) \ , \forall \boldsymbol{x} \in \mathcal{X} \ .$$

By means of words, the classification of H_T is obtained by summing the outputs of some elements of $\mathcal{H}_T$. Many classifiers satisfy (**A**), such as decisition trees, decision lists, linear separators, etc.

Definition 1 *The **decision graph** of H_T (on $\mathcal{X}$) is an oriented graph $G = (\mathcal{H}_T, \mathcal{E})$; an arc $(h_t, h_{t'}) \in \mathcal{E}$ iff $t < t'$ and there exists some $\boldsymbol{x} \in \mathcal{X}$ such that $h_t, h_{t'} \in g_{H_T}(\boldsymbol{x})$ and no $h_{t''}$ with $t < t'' < t'$ is in $g_{H_T}(\boldsymbol{x})$.*

Remark that G is acyclic, and g_{H_T} maps each observation of $\mathcal{X}$ to some path of G. We also define $\mathcal{P}$ as representing the set of *paths* of G that is mapped from $\mathcal{X}$ by g_{H_T}:

$$\mathcal{P} \quad = \quad \{p \subseteq \mathcal{H}_T : \exists \boldsymbol{x} \in \mathcal{X}, p = g_{H_T}(\boldsymbol{x})\} \ .$$

The simplest case of G is obtained when $g_{H_T}(\boldsymbol{x}) = \mathcal{H}_T, \forall \boldsymbol{x} \in \mathcal{X}$: H_T is a linear separator, and G a single path from h_1 to h_T. Examples of classes of classifiers different from linear separators and that fit to (**A**) include DT and ODT. In this case, G is a tree and weak hypotheses are in fact constant predictions on the tree nodes. Figure 2 (right) displays the way we can represent a DT so as to meet (**A**). Remark that there exists many possible equivalent transformations of a DT; the way we induce H_T shall favor a single one out of all, as follows. We define two types of subsets of $\mathcal{S}$ (with $p \in \mathcal{P}$ and $1 \leq t \leq T+1$):

$$\mathcal{S}_p \quad = \quad \{\langle \boldsymbol{x}_i, y_i \rangle \in \mathcal{S} : p = g_{H_T}(\boldsymbol{x}_i)\} \ , \tag{4}$$

$$\mathcal{S}_t \quad = \quad \cup_{p \in \mathcal{P} : h_t \in p} \mathcal{S}_p \ . \tag{5}$$

```
Input: $\mathcal{S}, \boldsymbol{w}_1$;
for $t = 1, 2, ..., T$
  compute $\mathcal{S}_t$;
  $h_t \leftarrow \mathrm{WL}(\mathcal{S}_t, \boldsymbol{w}_t)$;
  for $i = 1, 2, ..., m$
    $w_{t+1,i} \leftarrow w_{t,i} \times \begin{cases} \frac{1-(\mu_t y_i h_t(\boldsymbol{x}_i)/h_t^\star)}{1-\mu_t^2} & \text{if } s_i \in \mathcal{S}_t \\ 1 & \text{if } s_i \in \mathcal{S} \backslash \mathcal{S}_t \end{cases}$ ;
for $t = 1, 2, ..., T$
  $\alpha_t \leftarrow (1/(2h_t^\star)) \ln((1+\mu_t)/(1-\mu_t))$;
GenericGreedy (G$^2$)
```

Figure 3: The generic greedy induction of H_T. The computation of $\mathcal{S}_t$ depends on the decision graph of H_t.

We also extend the weight notation, and let $w_{t',\mathcal{S}_p} = \sum_{s_i \in \mathcal{S}_p} w_{t',i}$ and $w_{t',\mathcal{S}_t} = \sum_{s_i \in \mathcal{S}_t} w_{t',i}$ (with $p \in \mathcal{P}$ and $1 \leq t' \leq T+1$). We also define $h_t^\star = \max_{s_i \in \mathcal{S}_t} |h_t(\boldsymbol{x}_i)| \in \mathbb{R}$, the maximal absolute value of h_t over $\mathcal{S}_t$. After [Nock and Nielsen, 2006], we define two notions of *margins*: the first is the normalized margin h_t over $\mathcal{S}$:

$$\mu_t \quad = \quad \frac{1}{h_t^\star w_{t,\mathcal{S}_t}} \sum_{s_i \in \mathcal{S}_t} w_{t,i} y_i h_t(\boldsymbol{x}_i) \in [-1, +1] \ . \tag{6}$$

With the help of this definition, Figure 3 presents the generic greedy induction of H_T. Remark that when H_T is a linear separator, G^2 is the AdaBoost$_{\mathbb{R}}$ boosting algorithm of [Nock and Nielsen, 2006]. The second margin definition is the margin of H_T on example $\langle \boldsymbol{x}, y \rangle$ [Nock and Nielsen, 2006]:

$$\nu_T(\langle \boldsymbol{x}, y \rangle) \quad = \quad \tanh(y H_T(\boldsymbol{x})/2) \in [-1, +1] \ . \tag{7}$$

This margin comes to mind from the relationships between boosting and the logistic prediction $H_T(\boldsymbol{x}) = \log(\mathbf{Pr}[y = +1|\boldsymbol{x}]/\mathbf{Pr}[y = -1|\boldsymbol{x}])$ [Friedman *et al.*, 2000]. In this case, (7) becomes $\nu_T(\langle \boldsymbol{x}, y \rangle) = y(\mathbf{Pr}[y = +1|\boldsymbol{x}] - \mathbf{Pr}[y = -1|\boldsymbol{x}])$, whose expectation is the normalized margin (6) of the Bayesian prediction in $[-1, +1]$ (also called gentle logistic approximation [Friedman *et al.*, 2000]). Following [Nock and Nielsen, 2006], we define the *margin error* of H_T ($\forall \theta \in [-1, +1]$):

$$\nu_{\boldsymbol{w}_1, H_T, \theta} \quad = \quad E_{\boldsymbol{w}_1}(1_{[\nu_T(s) \leq \theta]}) \ , \tag{8}$$

and we have $\varepsilon(H_T, \boldsymbol{w}_1) \leq \nu_{\boldsymbol{w}_1, H_T, 0}$. Thus, minimizing $\nu_{\boldsymbol{w}_1, H_T, 0}$ would amount to minimize $\varepsilon(H_T, \boldsymbol{w}_1)$ as well. The following Theorem shows much more, as under some weak conditions, we can show that $\nu_{\boldsymbol{w}_1, H_T, \theta}$ is minimized for *all* values of $\theta \in [-1, +1)$ (and not only for $\theta = 0$).

Theorem 1 *After $T \geq 1$ queries of WL, the classifier obtained, H_T, satisfies:*

$$\nu_{\boldsymbol{w}_1, H_T, \theta} \leq \left(\frac{1+\theta}{1-\theta}\right) \times \sum_{p \in \mathcal{P}} w_{T+1, \mathcal{S}_p} \prod_{h_t \in p} \sqrt{1 - \mu_t^2} \ , \tag{9}$$

for any $\theta \in [-1, +1]$.

(proof in appendix). To read Theorem 1, consider the following WLA for real-valued weak hypotheses, borrowed from [Nock and Nielsen, 2006]:

(WLA) $|\mu_t| \geq \gamma$, for some $\gamma > 0$ ($\forall t \geq 1$).

Under the WLA, Theorem 1 states that $\nu_{\boldsymbol{w}_1, H_T, \theta} \leq K_\theta \exp(-\min_{p \in \mathcal{P}} |p|\gamma^2/2)$, with K_θ constant whenever θ is a constant $\in [-1, +1)$. In other words, provided the induction in G^2 is performed so as to keep paths with roughly equal size, such as by a breadth-first induction of the decision graph, the margin error is guaranteed to decrease exponentially fast. To see how G^2 fits to tree-shaped decision graphs, consider the following assumption:

(assumption **B**) (i) each $h_t \in \mathcal{H}_T$ is a constant, and (ii) G is a rooted tree.

Assumption (**B**) basically restricts H_T to be a tree of any arity, still in which any kind of classifier may be used to split the internal nodes. As in (3), we use notation $w^{+/-}_{.}$ as the index' weight for class "+1" or "-1", respectively.

Theorem 2 *Suppose that (**A**) and (**B**) hold. Then:*

$$H_T(\boldsymbol{x}) \quad = \quad \frac{1}{2} \ln \frac{w^+_{1,\mathcal{S}_t}}{w^-_{1,\mathcal{S}_t}} \ , \qquad (10)$$

$\forall \boldsymbol{x} \in \mathcal{X}$ and h_t is the leaf of $g_{H_T}(\boldsymbol{x})$. Furthermore, (2) simplifies as:

$$\varepsilon_{\exp}(H_T, \boldsymbol{w}_1) = \sum_{h_t \text{ leaf of } G} w_{1,\mathcal{S}_t} \times 2 \sqrt{\frac{w^+_{1,\mathcal{S}_t}}{w_{1,\mathcal{S}_t}} \left(1 - \frac{w^+_{1,\mathcal{S}_t}}{w_{1,\mathcal{S}_t}}\right)} \qquad (11)$$

$$= \quad \varepsilon_{\mathrm{imp}}(H_T, \boldsymbol{w}_1, 2\sqrt{z(1-z)}) \ . \qquad (12)$$

(proof in appendix).

4 Discussion

4.1 Kearns and Mansour's algorithm, AdaBoost and G^2

The similarity between (12) and (3) with $f(z) = 2\sqrt{z(1-z)}$ is immediate, and quite surprising as it shows the identity between a convex loss and the expectation of a concave loss. However, this is not a coincidence. Indeed, Theorem 2 shows a rather surprising result: the choice of the weak hypotheses does not impact at all on H_T (see (10)). When (**A**) and (**B**) hold, the only way to modify H_T is thus through its decision graph, *i.e.* on the choice of the splits of the tree. There is a simple way to choose them, which is to do the same thing as the most popular LS boosting algorithms [Friedman *et al.*, 2000; Schapire and Singer, 1999]: repeatedly minimize the exponential loss in (2). Because of Theorem 2, this amounts to the minimization of the impurity criterion in (3) with $f(z) = 2\sqrt{z(1-z)}$. This is *exactly* the DT induction algorithm proposed by [Kearns and Mansour, 1999] that meets the representation optimal bound.

On the other hand, when H_T is a linear separator, there is no influence of the decision graph on the induction of H_T as it is a single path from h_1 to h_T. The only way to modify H_T is thus through the choice of the weak hypotheses. Suppose that each weak hypothesis has output restricted to the set of classes, $\{-1, +1\}$. In this case, $\alpha_t = (1/2)\ln((1 - \varepsilon(h_t, \boldsymbol{w}_t))/\varepsilon(h_t, \boldsymbol{w}_t)) = \arg\min_{\mathbb{R}} E_{\boldsymbol{w}_1}(\exp(-y h_t(\boldsymbol{x})))$ (Figure 1), and G^2 matches *exactly* discrete AdaBoost [Freund and Schapire, 1997].

Thus, decision trees and linear separators are somehow extremal classifiers with respect to G^2. Finally, when H_T is a linear separator without restriction on the weak hypotheses, G^2 specializes to $\mathrm{AdaBoost}_{\mathbb{R}}$ [Nock and Nielsen, 2006].

4.2 All boosting algorithms

In the original boosting setting, the examples are drawn independently according to some unknown but fixed distribution D over $\mathcal{X}$, and the goal is to minimize the *true* risk $\varepsilon(H_T, D)$ with high probability, *i.e.* we basically wish that $\varepsilon(H_T, D) \leq \epsilon$ with probability $\geq 1 - \delta$ over the sampling of $\mathcal{S}$ [Freund and Schapire, 1997; Kearns and Valiant, 1989]. Two sufficient conditions for a polynomial time induction algorithm to satisfy this constraint are (i) return H_T with $\varepsilon(H_T, \boldsymbol{w}_1) = 0$, and (ii) prove that structural parameters of the class of classifiers to which H_T belongs satisfy particular bounds [Freund and Schapire, 1997; Shawe-Taylor and Cristianini, 1998]. Theorem 1 is enough to prove that (i) holds under fairly general conditions for algorithm G^2 in Figure 3 provided WLA holds. For example, $T = (2/\gamma^2)\log(1/\epsilon)$ iterations for LS and $T = (1/\epsilon)^{2/\gamma^2}$ for DT are enough to have $\varepsilon(H_T, \boldsymbol{w}_1) \leq \epsilon$ from Theorem 1. Fixing $\epsilon < \min_i w_{1,i}$ easily yields (i). The bound for LS is the same as AdaBoost (discrete or real) [Schapire and Singer, 1999], while that for DT improves upon the exponent constant of [Kearns and Mansour, 1999]. Finally, (ii) is either immediate, or follows from mild assumptions on G and WL. As a simple matter of fact, (i) and (ii) also hold when inducing ODT with G^2.

4.3 Recursive Boosting and Oblique Decision Trees

The preceeding Subsections suggest that G^2 could be used not only to build H_T, but also as the core procedure for WL. For example, it comes from Theorem 2 that AdaBoost + trees without pruning (Figure 1) is equivalent to G^2 growing linear separators, in which WL is G^2 growing a decision tree, in which WL returns any constant weak hypothesis. In the general case, we would obtain a recursive/composite design of the "master" G^2, via G^2 itself, and the recursion would end until we reach a WL that can afford exhaustive search for simple classifiers (*e.g.* axis-parallel stumps, constant classifiers, etc.), instead of calling again G^2. However, G^2 can *also* be used to build the decision graph of H_T, in the same recursive fashion. Consider for example the class ODT. The internal nodes' splits are local classifiers from LS that decide the path based on the sign of their output, or equivalently, on the class they would give. This suggest to build the tree with G^2 on *both* the linear separators in the internal nodes of H_T (use $\mathcal{S}_\ell$ to split leaf ℓ, where the linear separator uses ordinary decision stumps), *and* on the tree shape as well. Call BoostODT this ODT induction algorithm. It turns out that it brings a boosting algorithm, that takes full advantage of the ODT structure. However, this time, it is enough to assume the WLA *one level deeper*, *i.e.* only for the stumps of the linear separators, and not for the splits of the oblique decision tree.

Theorem 3 *BoostODT is a boosting algorithm.*

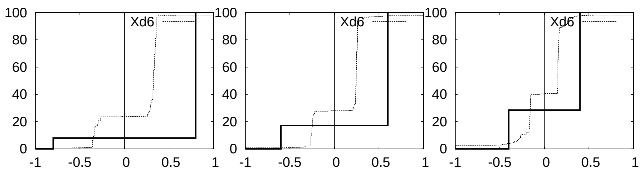

Figure 4: Margin distributions for BoostODT ($T_1 = 50$) on domain XD6, with 10% (left), 20% (center) and 30% class noise (right). Stair-shaped (bold plain) curves are the theoretical margins for the logistic model (see text).

The proof, omitted due to the lack of space, builds upon Theorem (1) plus Lemma 2.1 in [Mansour and McAllester, 2002], and structural arguments in [Freund and Schapire, 1997; Shawe-Taylor and Cristianini, 1998]. The proof emphasizes the relative importance of sizes: suppose that each linear separator contains the same number of weak hypotheses (T_1), and the tree is complete with T_2 internal nodes; then, having

$$T_1 \log T_2 \quad = \quad \Omega((1/\gamma^2)\log(1/\epsilon))$$

is enough to have $\varepsilon(H_T, \boldsymbol{w}_1) \leq \epsilon$. From an experimental standpoint, this suggests to build trees with $T_1 \gg T_2$.

5 Experiments

We have performed comparisons on a testbed of 25 domains with two classes, most of which can be found on the UCI repository of ML databases [Blake *et al.*, 1998]. Comparisons are performed via ten-fold stratified cross-validation, against OC1 and AdaBoost in which the weak learner is C4.5 unpruned. BoostODT was ran for $T_1 \in \{50, 200\}$ (the weak hypotheses of the linear separators are decision stumps) and $T_2 = 4$. To make fair comparisons, we ran AdaBoost for a total number of $T = 5$ boosting iterations. This brings fair comparisons, as an observation is classified by 5 nodes (including leaves) in BoostODT, and 5 unpruned trees in AdaBoost. Before looking at the results, BoostODT has proven to be much faster than OC1 in practice (ten to *hundred* times faster). OC1's time complexity is $O(nm^2 \log m)$ [Murthy *et al.*, 1994], without guarantee on its result, while BoostODT's is $O(nm)$ under the WLA, with the guarantee to reach empirical consistency in this case. Complexity reductions have almost the same order of magnitude with respect to SVM based induction of ODT [Shawe-Taylor and Cristianini, 1998]. Table 1 summarizes the results obtained. With rejection probabilities p ranging from less than .05 to less than .005 for the hypothesis H_0 that BoostODT does not perform better, the four sign tests comparing both runs of BoostODT to its opponents are enough to display its better performances, and this is confirmed by student paired t-tests. There is more: we can tell from simulated domains that BoostODT performs as better as the domain gets harder. It is the case for the Monks domains, and the LEDeven domains. BoostODT is indeed beaten by both opponents on LEDeven, while it beats both on LEDeven+17 (=LEDeven +17 irrelevant variables).

Looking at these simulated domains, we have drilled down the results of BoostODT. Using (8), we have plotted on Figure 4 its margin error curves on domain XD6 with variable class noise (see *e.g.* [Nock and Nielsen, 2006] for a description of the domain), averaged over the test folds [Nock and Nielsen, 2006]. The *margin curve* obtained is compared to that of the logistic prediction of [Friedman *et al.*, 2000], which can be computed exactly. The approximation of the logistic model by BoostODT is quite remarkable. Indeed, its margin curves display the single stair-shape of a theoretical logistic model for a domain XD6 with 8-13% additional class noise, uniformly distributed among the ODT leaves.

Domain	BoostODT		OC1	AdaBoost
	$T_1 = 50$	$T_1 = 200$		+C4.5
Adult-strech	**0.00**	**0.00**	0.15	**0.00**
Breast-cancer	28.67	**27.27**	37.76	33.92
Breast-cancer-w.	**3.72**	3.86	6.01	4.43
Bupa	**28.12**	**28.12**	37.97	31.59
Colic	**17.66**	18.21	23.91	18.21
Colic.ORIG	**13.86**	15.76	16.03	32.07
Credit-a	**14.64**	14.93	20.43	15.51
Credit-g	26.50	**25.40**	31.10	28.40
Diabetes	25.91	**23.44**	34.64	29.82
Hepatitis	20.65	**18.71**	20.65	**18.71**
Ionosphere	7.41	**6.27**	9.12	7.41
Kr-vs-kp	3.38	3.35	3.69	**0.69**
Labor	**10.53**	**10.53**	15.79	12.28
LEDeven	15.25	14.75	**9.75**	10.50
LEDeven+17	22.75	**21.50**	38.25	30.75
Monks1	25.36	25.36	**0.00**	2.16
Monks2	1.00	0.67	**0.33**	26.96
Monks3	**1.44**	2.17	2.71	2.53
Mushroom	**0.00**	**0.00**	0.05	**0.00**
Parity	47.66	47.27	46.88	**44.14**
Sick	2.15	1.91	2.41	**1.09**
Sonar	13.94	**12.50**	33.17	18.27
Vote	3.91	**3.45**	4.37	5.29
XD6	20.57	18.77	**5.33**	5.40
Yellow-small	**0.00**	**0.00**	**0.00**	**0.00**
#Wins ($T_1 = 50$)	**17**(10/4)		**5**(3)	**8**(3)
#Wins ($T_1 = 200$)		**18**(10/7)	**5**(3)	**8**(3)

Table 1: Results on 25 domains. For each domain, bold faces indicate the lowest errors. In each row "#Wins ($T_1 = z$)", bold faces denote the number of times the corresponding algorithm in column is the best over three columns: BoostODT($T_1 = z$), OC1 and AdaBoost+C4.5 ($z \in \{50, 200\}$). Furthermore, the four numbers in parentheses in each row are the number of *significant* wins (student paired t-test, $p = .05$), for BoostODT vs OC1, BoostODT vs AdaBoost+C4.5, and OC1 vs BoostODT, AdaBoost+C4.5 vs BoostODT (from left to right).

6 Conclusion

Perhaps one main contribution of this paper is to show that formal boosting is within reach using the same unified algorithm, for a wide variety of formalisms not restricted to the most popular included in this paper (such as decision lists [Rivest, 1987], simple rules [Nock, 2002], etc.). Another contribution, quite surprising, is to show that a boosting algorithm follows immediately even for complex combinations of these formalisms, such as linear combinations of oblique decision trees, decision trees in which splits are decided by decision lists, etc. This is crucial, as our last contribution, the first boosting algorithm for the class of oblique decision trees, contrasts in simplicity with respect to previous approaches on inducing oblique decision trees. In future works, we plan to evaluate the experimental and theoretical potentials of these boosting algorithms for these other formalisms.

Acknowledgments and code availability

R. Nock gratefully thanks Sony CS Labs Tokyo for a visiting grant during which this work was started. Resources related to BoostODT, including related Java classes for the Weka platform, can be obtained from the authors.

References

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

[Breiman *et al.*, 1984] L. Breiman, J. H. Freidman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth, 1984.

[Cantú-Paz and Kamath, 2003] E. Cantú-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Trans. on Evo. Comp.*, 7:54–68, 2003.

[Freund and Schapire, 1997] Y. Freund and R. E. Schapire. A Decision-Theoretic generalization of on-line learning and an application to Boosting. *JCSS*, 55:119–139, 1997.

[Friedman *et al.*, 2000] J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression : a Statistical View of Boosting. *Annals of Statistics*, 28:337–374, 2000.

[Heath *et al.*, 1993] D. Heath, S. Kasif, and S. Salzberg. Learning oblique decision trees. In *Proc. of the 13 th IJCAI*, pages 1002–1007, 1993.

[Kearns and Mansour, 1999] M.J. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. *JCSS*, 58:109–128, 1999.

[Kearns and Valiant, 1989] M.J. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Proc. of the 21 th ACM STOC*, pages 433–444, 1989.

[Mansour and McAllester, 2002] Y. Mansour and D. McAllester. Boosting using branching programs. *JCSS*, 64:103–112, 2002.

[Murthy *et al.*, 1994] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *JAIR*, 2:1–32, 1994.

[Nock and Nielsen, 2006] R. Nock and F. Nielsen. A ℝeal Generalization of discrete AdaBoost. In *Proc. of the 17 th ECAI*, pages 509–515, 2006.

[Nock, 2002] R. Nock. Inducing interpretable Voting classifiers without trading accuracy for simplicity: theoretical results, approximation algorithms, and experiments. *JAIR*, 17:137–170, 2002.

[Quinlan, 1993] J. R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann, 1993.

[Rivest, 1987] R.L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

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

[Schapire, 1990] R. E. Schapire. The strength of weak learnability. *Machine Learning*, pages 197–227, 1990.

[Shawe-Taylor and Cristianini, 1998] J. Shawe-Taylor and N. Cristianini. Data dependent structural risk minimization for perceptron decision trees. In *NIPS* 10*, pages 336–342, 1998.

7 Appendix

Proof of Theorem 1. Let us focus on a single $p \in \mathcal{P}$, and consider an example $s_i \in \mathcal{S}_p$. Using the proof of Theorem 1 in [Nock and Nielsen, 2006], we obtain:

$$w_{1,i} 1_{[\nu_T(s_i) \leq \theta]} \quad \leq \quad w_{T+1,i} \left(\frac{1+\theta}{1-\theta}\right) \times \prod_{h_t \in p} \sqrt{1-\mu_t^2} \ ,$$

Computing $\nu_{\boldsymbol{w}_1, H_T, \theta} = E_{\boldsymbol{w}_1}(1_{[\nu_T(s_i) \leq \theta]})$ and simplifying for each $p \in \mathcal{P}$ yields the statement of the Theorem.

Proof of Theorem 2. We need the following simple Lemma (proof straightforward).

Lemma 1 *Whenever (**A**) and (i) of (**B**) are satisfied, we have* $w^-_{t+1,\mathcal{S}_t} = w^+_{t+1,\mathcal{S}_t}, \forall 1 \leq t \leq T$.

Consider a leaf h_t of G, and let $p = g_{H_T}(\boldsymbol{x})$, for an observation $\boldsymbol{x}$ that reaches this leaf. We first suppose that $\mathcal{S}_t$ contains both positive and negative examples. This implies that $(\alpha_{t'} h_{t'})$ is finite, $\forall h_{t'} \in p$. All the positive examples (resp. negative examples) of $\mathcal{S}_t$ have seen their weights modified in the *same* way through induction. Let $\sigma_{b,t} = 1 + b\mu_t$, with $b \in \{+,-\}$. When H_T is built, the total weight, according to $\boldsymbol{w}_{T+1}$, of the positive (resp. negative) examples of $\mathcal{S}_t$, satisfy $w^+_{T+1,\mathcal{S}_t} = w^+_{1,\mathcal{S}_t} / \left(\prod_{h_{t'} \in p : h_{t'} > 0} \sigma_{+,t'} \prod_{h_{t'} \in p : h_{t'} < 0} \sigma_{-,t'}\right)$ (resp. $w^-_{T+1,\mathcal{S}_t} = w^-_{1,\mathcal{S}_t} / (\prod_{h_{t'} \in p : h_{t'} < 0} \sigma_{+,t'} \prod_{h_{t'} \in p : h_{t'} > 0} \sigma_{-,t'})$). From Lemma 1, we have $w^+_{T+1,\mathcal{S}_t} = w^-_{T+1,\mathcal{S}_t}$, and we obtain $(w^+_{1,\mathcal{S}_t} / w^-_{1,\mathcal{S}_t}) = \left(\prod_{h_{t'} \in p : h_{t'} > 0} (\sigma_{+,t'} / \sigma_{-,t'})\right) \times \left(\prod_{h_{t'} \in p : h_{t'} < 0} (\sigma_{-,t'} / \sigma_{+,t'})\right)$. Taking half the logarithms of both sides and rearranging, we obtain:

$$\frac{1}{2} \ln \frac{w^+_{1,\mathcal{S}_t}}{w^-_{1,\mathcal{S}_t}} = \sum_{h_{t'} \in p} \frac{h_{t'}}{2h^\star_{t'}} \ln \frac{1+\mu_{t'}}{1-\mu_{t'}} = \sum_{h_{t'} \in p} \alpha_{t'} h_{t'}(\boldsymbol{x}) = H_T(\boldsymbol{x}) \ ,$$

as claimed. The case where $\mathcal{S}_t$ does not contain examples of both classes is immediate. Indeed, that means that at least one of $w^+_{1,\mathcal{S}_t}$ and $w^-_{1,\mathcal{S}_t}$ is zero as G is a tree, and we obtain that $(1/2) \ln(w^+_{1,\mathcal{S}_t} / w^-_{1,\mathcal{S}_t}) = H_T(\boldsymbol{x}) = \pm\infty$. Finally, (11) is the simplification of (2) with the new expressions for $H_T(.)$.