Boosting Grammatical Inference with Confidence Oracles

Jean-Christophe Janodet JANODET@UNIV-ST-ETIENNE.FR
EURISE, Faculty of Sciences, 23 rue Paul Michelon, University of Jean Monnet, 42023 Saint-Etienne, FRANCE

Richard Nock RICHARD.NOCK@MARTINIQUE.UNIV-AG.FR
DSI, Université Antilles-Guyane, BP 7209, 97275 Schoelcher, FRANCE (Martinique)

Marc Sebban MARC.SEBBAN@UNIV-ST-ETIENNE.FR
EURISE, Faculty of Sciences, 23 rue Paul Michelon, University of Jean Monnet, 42023 Saint-Etienne, FRANCE

Henri-Maxime Suchier HENRI.MAXIME.SUCHIER@UNIV-ST-ETIENNE.FR
EURISE, Faculty of Sciences, 23 rue Paul Michelon, University of Jean Monnet, 42023 Saint-Etienne, FRANCE

Abstract

In this paper we focus on the adaptation of boosting to grammatical inference. We aim at improving the performances of state merging algorithms in the presence of noisy data by using, in the update rule, additional information provided by an oracle. This strategy requires the construction of a new weighting scheme that takes into account the confidence in the labels of the examples. We prove that our new framework preserves the theoretical properties of boosting. Using the state merging algorithm RPNI*, we describe an experimental study on various datasets, showing a dramatic improvement of performances.

1. Introduction

Grammatical inference is a subtopic of machine learning whose aim consists in learning models of languages (sets of words or sequences). Among them, deterministic finite automata (DFA) are finite state machines which take words as input, and accept or reject them according if they characterize or not the concept to learn. In machine learning, a DFA can be seen as a classifier which separates the set of all words in two classes, a positive one (for the accepted words) and a negative one (for the rejected words). A practical reason which explains the efforts made to learn such classifiers comes from the fact that many applications, *e.g.* speech recognition, pattern matching, language processing, *etc.*, can take advantage of the structural and semantic properties of DFA (de la Higuera, 2004).

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

Among the algorithms aiming at learning DFA, those based on state merging have been widely studied during the last decade, and particularly EDSM (Lang et al., 1998) and the well known RPNI (Oncina & García, 1992). Both of them learn from a sample $E = E_+ \cup E_-$, and try to infer, by state merging, a small DFA that accepts all the words of E_+ (called positive examples), and rejects all those of E_- (called negative ones). For instance, the DFA in Figure 1 was computed by RPNI with $E_+ = \{b, abb, ab, bab, \lambda\}$ and $E_- = \{aa, ba\}$, where λ denotes the empty string. A word is accepted if its parsing starts from the initial state 0, follows transitions, and ends in a final state (always positive and graphically described by a double circle). Notice that this is the case for all the elements of E_+ ending in the unique final state 0. The negative words are rejected, *i.e.* their parsing do not reach a final state.

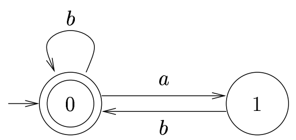

Figure 1. A DFA with two states and three transitions.

RPNI is an *exact* learning algorithm because it fits the data. It is proven that if E contains some special (*characteristic*) words, then RPNI is able to infer, in polynomial time, the DFA that produced the data (Oncina & García, 1992), called the *target* DFA. However, the specificities of modern databases, which often contain noisy data, challenge these theoretical properties. Since RPNI is not immune to overfitting, noisy data have dramatic effects, and penalize the DFA in terms of number of states and error rate. Noise tolerance in such algorithms is thus a crucial problem studied in grammatical inference, confirming the same trend as in the whole machine learning field.

The first attempt to limit the risk of overfitting in state merging algorithms was proposed by Sebban and Janodet (2003); they tried to relax the merging rule of RPNI without endangering the generalization error of the DFA. They introduced the new algorithm RPNI*, whose merging rule, based on statistical inference theory, allows to tolerate the presence of noisy words. From an experimental standpoint, this yields a significant improvement over RPNI's performances. But their approach is doubly limited: first, they provide theoretical results showing difficulties to infer the target DFA for noise levels over 10%. Second, due to its statistical nature, the use of RPNI* can result in a *wrongly* accepted merging. Actually, while RPNI does not admit *any* negative example in a final state, positive and negative words can share the same final state with RPNI* due to the relaxation of the merging constraint. While one hopes that such negative words are indeed noisy data, one can be afraid of accepting truly negative words, which would lead to serious problems.

In fact, the underlying problem of RPNI* has to be linked to a couple of recurrent questions in machine learning: (i) is it possible to optimize the performances of a learning algorithm (here a state merging one)? (ii) is it possible to control this optimization even in the presence of noisy data (that would avoid here to merge "true" negative and positive examples in the same state)? Indisputably, we think that both of these questions are a matter for boosting, despite the fact that it has never been adapted to DFA inference. The strategy of boosting, and its algorithm ADABOOST (Freund & Schapire, 1996; Freund & Schapire, 1997) (see Algorithm 1), consists in successively training T times a learning algorithm WL on various probability distributions $\mathbf{w}_t$ over the learning sample E, and in combining the resulting classifiers h_t (called *weak hypotheses*) in an efficient single classifier H_T. At each round, the current distribution favors (exponentially) the weights of examples misclassified by the previous hypothesis.

Some recent works tried to limit the risk of overfitting of boosting in the presence of noise by answering the following question: when does a misclassified example really deserve a weight increase? ADABOOST tending (wrongly) to exponentially increase the weight of the outliers, new approaches aim then at controlling

for all $x \in E$ **do** $w_0(x) \longleftarrow 1/|E|$;
for $t = 0$ **to** T **do**
  $h_t \longleftarrow \text{WL}(E, \mathbf{w}_t)$;
  $\epsilon_t \longleftarrow \sum_{\{e \in E: y(x) \neq h_t(x)\}} w_t(x)$;
  $c_t \longleftarrow (1/2)\ln((1-\epsilon_t)/\epsilon_t)$;
  $Z_t \longleftarrow \sum_{e \in E} w_t(x)\exp(-c_t y(x) h_t(x))$;
  for all $x \in E$ **do**
    $w_{t+1}(x) \longleftarrow w_t(x)\exp(-c_t y(x) h_t(x))/Z_t$;
return H_T **such that** $H_T(x) = \sum_{t=0}^{T} c_t h_t(x)$;

Algorithm 1: Pseudo-code of ADABOOST.

the update rule by using weighting functions smoother than the original exponential one (Friedman et al., 1998; Domingo & Watanabe, 2000; Freund, 2001). Rather than using other functions, that may challenge properties of convergence, we keep in this paper the exponential function, but we assume that we have access to a helpful oracle. This oracle, that we will simulate in practice, provides a real-valued confidence on the label of the learning words. A positive confidence means that we can be sure about the word's label, whereas a negative confidence conveys a certain doubt about it.

It is crucial to precise that a negative confidence *does not* mean that the word belongs to the other class, but rather that we do not know what is the right class. Intuitively, one could think that such words must be removed from E, but following Blum and Mitchell (1998), we claim that examples without label may in fact be of significant help for learning, provided one knows how to use the information carried out by their description. In grammatical inference, words allow to guess the structure (states and transitions) of a target DFA, whereas their labels determine if the states are final or not. It is thus of significant interest to take into account the examples with a suspicious label in boosting procedures. Nevertheless, the use of the confidence oracle, for dealing with noisy data, requires the modification of the standard update rule. Actually, while ADABOOST is originally designed to increase the weights of all the misclassified examples, we only want here to raise the weights of noise free misclassified ones. Although this strategy is useful for improving any machine learning algorithm, we think that it is particularly well suited to improve the performances of state merging algorithms such as RPNI*.

This paper is organized as follows. After some details about RPNI* (Section 2), we provide a new theoretical boosting scheme adapted to the use of an oracle (Section 3). We modify the standard weighting rule of ADABOOST, and we prove that our weighting scheme has the boosting theoretical properties. Then, we propose in Section 4 a strategy to simulate an efficient oracle satisfying these properties. This strategy is based on the construction of a k nearest neighbor graph (Cover & Hart, 1967) over the set of words. In Section 5, we show on various databases the interest of boosting to optimize state merging algorithms, such as RPNI*.

2. The State Merging Algorithm RPNI*

We aim at giving a loose description of RPNI* (see Algorithm 2 and Sebban and Janodet (2003) for more details). An *alphabet* Σ is a nonempty finite set of *letters* and a *word* is any sequence $x = l_1 l_2 \dots l_n$ of letters. A *deterministic finite automaton* (DFA) is a tuple $A = \langle Q, \delta, s_0, F \rangle$ such that Q is a set of states, $\delta : Q \times \Sigma \to Q$ is a transition function, $s_0 \in Q$ is an initial state and $F \subseteq Q$ is a set of final states. A state s *contains* a word x iff $\delta(s_0, x) = s$. A word x is *accepted* by A if $\delta(s_0, x) \in F$ and *rejected* otherwise. For instance, the DFA of Figure 1 is defined by $Q = \{0, 1\}$, $s_0 = 0$, $F = \{0\}$, $\delta(0, a) = 1$ and $\delta(0, b) = \delta(1, b) = 0$. The word abb is accepted since $\delta(0, abb) = \delta(1, bb) = \delta(0, b) = 0 \in F$. With the same reasoning, ba is rejected because $\delta(0, ba) = 1 \notin F$.

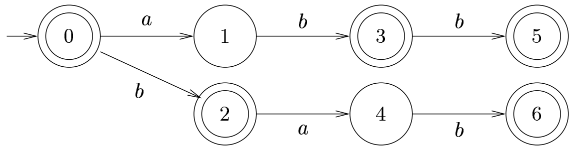

Figure 2. An example of PTA.

The first task RPNI* achieves is the construction of the PTA (*prefix tree acceptor*) of the words of E_+, that is the greatest trimmed DFA accepting only the words of E_+ (see Figure 2). Its states are numbered following the hierarchical order over the prefixes of E_+. Then RPNI* runs along these states following the ordering. Merging two states means to collapse them into a new one, whose number is the smallest of the two merged ones. With RPNI, two states are mergeable if no negative example is contained in a final state. RPNI* relaxes this constraint in order to authorize the presence of noisy words. A state is *positive* (or final) if it contains strictly more positive words (of E_+) than negative ones (of E_-), and *negative* otherwise. A negative (resp. positive) word is *misclassified* if it is contained by a positive (resp. negative) state. Finally, a merging is *statistically acceptable* if the proportion of misclassified words in the whole DFA after the merging is not significantly higher than the proportion of misclassified words computed before the merging. This way to proceed allows to avoid overfitting phenomena.

```
A ⟵ PTA(E+);
for i = 1 to N − 1 do
    j ⟵ 0; continue ⟵ true;
    while (j < i) and continue do
        B ⟵ compute_merging(A, i, j);
        C ⟵ compute_final_states(B, E+, E−);
        if statistically_acceptable(C, E+, E−) then
            A ⟵ C; continue ⟵ false;
        j ⟵ j + 1;
return(A);
```

Algorithm 2: Pseudo-code of RPNI*.

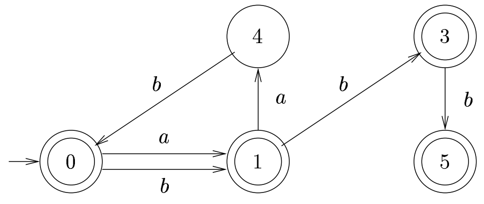

Figure 3. The effect of noisy data on the DFA.

In order to show the relevance of RPNI*, let us consider that the noisy word $bbbb$ is inserted in E_- in the example already described in introduction. According to the target DFA (Figure 1), this word should be positive. Figure 3 shows the new profoundly modified DFA inferred by RPNI, expressing well the fact that it is not immune to overfitting. Run on this problem, RPNI* is able to infer the target DFA, handling optimally the noise. However, one hopes that misclassified examples are really noisy, that is the case for $bbbb$ on this simple example. But due to its statistical nature, RPNI* can statistically infer a bad merging. Although one can theoretically reduce this risk, it cannot be null, and can have dramatic effects on RPNI*'s performances.

We aim here at collecting information about the confidence in the label of each example, and using it in a boosting procedure. Assuming that this information is given by an oracle, we would be able to detect which misclassified examples deserve to be learned or not. In other words, we would be able to separate the misclassified examples that must be subject to a weight increase, and those that deserve a weight drop. However, this requires the construction of a new optimal weighting scheme. That is the aim of the next section.

3. Boosting using a Confidence Oracle

Let E be the set of $|E| = m$ examples. Every $x \in E$ has a label $y(x)$ that can be positive ($+1$) or negative (-1). We assume that we have access to a *confidence oracle*, which gives to every example x a non-null real-valued confidence $q(x) \in [-1,1], q(x) \neq 0$. $q(x)$ does not change throughout learning. A positive confidence means that x belongs to the class it is assigned to in E. A negative confidence means that we cannot be confident in its label. We partition E into the following sets: $E_N = \{x \in E : q(x) < 0\}$ and $E_{\overline{N}} = E \backslash E_N$. Since we cannot be confident in the labels of E_N, we aim at minimizing by boosting the empirical error on $E_{\overline{N}}$, such that $\forall T > 0$,

$$\varepsilon(E_{\overline{N}}, H_T) = (1/|E_{\overline{N}}|) \sum_{x \in E_{\overline{N}}} [\![y(x) \neq H_T(x)]\!], \quad (1)$$

with $[\![\pi]\!]$ the indicator value in $\{0,1\}$ of the predicate π, and $H_T(x) = \sum_{t=0}^{T} c_t h_t(x)$ the whole combined hypothesis (using the same notations as in Algorithm 1). Let $\mathcal{P}_m$ be the m-dimensional probability simplex, and $\mathbf{u} \in \mathcal{P}_m$ the uniform vector. We assume a discrete probability vector $\mathbf{w}_t \in \mathcal{P}_m$ containing for each $x \in E$ its weight at time t, with $\mathbf{w}_0 = \mathbf{u}$ (as in ADABOOST).

We aim here at finding the new optimal update rule. It is known since Kivinen and Warmuth (1999) that standard boosting, as defined in Freund and Schapire (1997); Schapire and Singer (1999), is a particular case of the constrained optimization of a Bregman divergence. Let us recast this problem in our setting. The strategy is to repeatedly compute $\mathbf{w}_{t+1}$ from $\mathbf{w}_t$ following the minimization of the *information divergence*, $\mathbf{1}.\mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t) = \sum_{x \in E} \mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)(x)$, with $\mathbf{i}(.,.)$ the vector whose component for some $x \in E$ is:

$$\mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)(x) = \left(\mathbf{w}_{t+1} \ln \frac{\mathbf{w}_{t+1}}{\mathbf{w}_t} - \mathbf{w}_{t+1} + \mathbf{w}_t \right)(x).$$

The information divergence is convex in $\mathbf{w}_{t+1}$. One wishes to minimize $\mathbf{1}.\mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)$ subject to two constraints. The first one is straightforward: it expresses the fact that $\mathbf{w}_{t+1}$ is a distribution ($\mathbf{w}_{t+1} \in \mathcal{P}_m$):

$$\mathbf{1}.\mathbf{w}_{t+1} - 1 \quad = \quad 0. \quad (2)$$

In our case, this constraint remains the same. The second one integrates the loss function for each example in E. In classical boosting, it expresses that the last hypothesis found, h_t, is in one sense the worst on the new distribution, since it performs on average like random guessing: $\sum_{x \in E} (w_{t+1} y h_t)(x) = 0$. This constraint, together with a so-called *weak learning assumption* (Kearns & Vazirani, 1994), ensures that the new hypothesis h_{t+1} built on $\mathbf{w}_{t+1}$ is significantly different from h_t, thereby learning something "new" over E. In our case, this constraint is a bit different:

$$\begin{aligned} \mathbf{w}_{t+1}.\mathbf{a}_t &= 0, \quad (3) \\ \forall x \in E_{\overline{N}} : a_t(x) &= q(x)(y h_t)(x), \\ \forall x \in E_N : a_t(x) &= -q(x). \end{aligned}$$

Comparing it with boosting's second constraint, (3) says that $\sum_{x \in E_{\overline{N}}} (w_{t+1} q y h_t)(x) = \sum_{x \in E_N} (w_{t+1} q)(x)$. Since $\forall x \in E_N$, $q(x) < 0$, the right member is strictly negative, which makes h_t, the last hypothesis built, particularly bad on $E_{\overline{N}}$ with a distribution depending on both $\mathbf{w}_{t+1}$ and $\mathbf{q}$. These are clearly differences with classical boosting, which would yield an average (random) performance, on $\mathbf{w}_{t+1}$ alone and over E. The minimization of the information divergence under constraints (2) and (3) is obtained as the solution to ($\forall x \in E$):

$$\begin{aligned} &\partial_{\mathbf{w}_{t+1}} \mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)(x) \quad (4) \\ &+ \left[b_t \partial_{\mathbf{w}_{t+1}} (\mathbf{1}.\mathbf{w}_{t+1} - 1) + c_t \partial_{\mathbf{w}_{t+1}} (\mathbf{w}_{t+1}.\mathbf{a}_t) \right](x) = 0 \end{aligned}$$

with b_t and c_t Lagrange multipliers. Solving (4) for $\mathbf{w}_{t+1}$ brings $\forall x \in E$:

$$\left(\ln \frac{\mathbf{w}_{t+1}}{\mathbf{w}_t} + b_t \mathbf{1} + c_t \mathbf{a}_t \right)(x) = 0$$
$$\Leftrightarrow w_{t+1}(x) = w_t(x) \exp(-c_t a_t(x)) \;/\; \exp(b_t).$$

Constraint (2) yields:

$$b_t = \ln \sum_{x \in E} w_t(x) \exp(-c_t a_t(x)), \quad (5)$$

The term inside the "ln" is the normalization coefficient for $\mathbf{w}_{t+1}$:

$$Z_t = \sum_{x \in E} w_t(x) \exp(-c_t a_t(x)). \quad (6)$$

We shall see hereafter that c_t is in fact > 0 (Lemma 2). Since the oracle is confident in the label of any $x \in E_{\overline{N}}$ ($q(x) > 0$), and since for any such example,

$$w_{t+1}(x) = w_t(x) \exp(-c_t q(x)(y h_t)(x)) / Z_t,$$

it comes that the weight of any $x \in E_{\overline{N}}$ increases in the next distribution iff it is misclassified by the current hypothesis ($(y h_t)(x) = -1$), and it decreases otherwise. On the other hand, the weight of all suspicious examples $x \in E_N$ ($q(x) < 0$) will ineluctably decrease:

$$w_{t+1}(x) = w_t(x) \exp(c_t q(x)) / Z_t.$$

We will see in the next section that such an update rule improves RPNI*'s behavior. The last unknown, c_t, is obtained from constraint (3) as the solution to:

$$\sum_{x \in E} (w_t a_t)(x) \exp(-c_t a_t(x)) \quad = \quad 0. \quad (7)$$

Notice that this is also stating $(\partial Z_t / \partial c_t) = 0$. Since Z_t is a convex function of c_t, solving (4) ultimately returns to the problem of minimizing Z_t. To see why solving (4) is so important, let us temporarily shift to the error committed by H_T.

Lemma 1 $\varepsilon(E_{\overline{N}}, H_T) \leq (m/|E_{\overline{N}}|)\prod_{t\leq T} Z_t$.

Proof: Because $\forall x \in E_{\overline{N}}$, $q(x) > 0$, we have for any such example $[\![H_T(x) \neq y(x)]\!] \leq \exp(-q(x)y(x)\sum_{t\leq T} c_t h_t(x))$. On the other hand, $w_{T+1}(x) = w_T(x)\exp(-q(x)y(x)c_T h_T(x))/Z_T$. Unraveling the update rule, we get $\forall x \in E_{\overline{N}} : w_{T+1}(x) = w_0(x)\exp(-q(x)y(x)\sum_{t\leq T} c_t h_t(x))/(\prod_{t\leq T} Z_t)$, hence $[\![H_T(x) \neq y(x)]\!] \leq m w_{T+1}(x)(\prod_{t\leq T} Z_t)$. Summing over all $x \in E_{\overline{N}}$, we get $\varepsilon(E_{\overline{N}}, H_T) \leq (m/|E_{\overline{N}}|)(\prod_{t\leq T} Z_t)(\sum_{x\in E_{\overline{N}}} w_{T+1}(x))$. Using the fact that $\sum_{x\in E_{\overline{N}}} w_{T+1}(x) \leq 1$, we get the Lemma. □

Since the term factoring the product of the normalization coefficients is constant given E and $E_{\overline{N}}$, Lemma 1 means that in order to drive down the error on $E_{\overline{N}}$ to zero, we should concentrate on minimizing each Z_t. This makes us kill two birds in one shot: solving (7) yields both the solution to (4), and (hopefully) the minimization of the error on $E_{\overline{N}}$ according to Lemma 1. Let us concentrate on the solution to (7), under the assumption: $\forall t \geq 0, \exists \gamma_t > 0$ a constant such that:

$$\sum_{x\in E_{\overline{N}}} (w_t q y h_t)(x) \quad / \sum_{x\in E_{\overline{N}}} (w_t q)(x) = \gamma_t. \tag{8}$$

Following boosting, we refer to (8) as a weak learning assumption (WLA), in particular since if h_t were random, we would have $\gamma_t = 0$. In other words, (8) traduces the fact that h_t must outperform random guessing by only a small amount for being considered as a weak hypothesis. Our WLA has two differences with classical boosting (Kearns & Vazirani, 1994). First, it is local: it puts emphasis on a subset of E. Second, it relies on a distribution which is not exactly $\mathbf{w}_t$, but a distribution $\mathbf{w}_{q,t} \in \mathcal{P}_m$ which is leveraged by $\mathbf{q}$, as follows: $\forall x \in E, w_{q,t}(x) = |q(x)|w_t(x)/Q_t$, with $Q_t = \sum_{x\in E} |q(x)|w_t(x)$ the normalization coefficient. Notice that $\forall \mathbf{w}_t \in \mathcal{P}_m$, $Q_t \neq 0$ because every example has $q(.) \neq 0$. We extend this definition to that of $W_{q,t}(E') = \sum_{x\in E'} \mathbf{w}_{q,t}(x)$ for all $E' \subseteq E$.
To complete our discussion on constraint (3), define $E^+_{\overline{N},t} = \{x \in E_{\overline{N}} : (yh_t)(x) = +1\}$ and $E^-_{\overline{N},t} = \{x \in E_{\overline{N}} : (yh_t)(x) = -1\}$. The WLA is equivalent to:

$$W_{q,t}(E^+_{\overline{N},t}) - W_{q,t}(E^-_{\overline{N},t}) = \gamma_t W_{q,t}(E_{\overline{N}}). \tag{9}$$

So we cannot choose $h_{t+1} = h_t$ under the WLA, since otherwise (3) would imply $W_{q,t+1}(E^+_{\overline{N},t}) - W_{q,t+1}(E^-_{\overline{N},t}) < 0$, while (9) would imply $W_{q,t+1}(E^+_{\overline{N},t}) - W_{q,t+1}(E^-_{\overline{N},t}) > 0$. Thus the WLA enforces h_{t+1} to learn something "new".

Lemma 2 *The solution to (7) is unique and strictly positive, as it satisfies:*

$$\frac{1}{2\overline{q}}\ln\frac{1 - W_{q,t}(E^-_{\overline{N},t})}{W_{q,t}(E^-_{\overline{N},t})} \leq c_t \leq \frac{1}{2\underline{q}}\ln\frac{1 - W_{q,t}(E^-_{\overline{N},t})}{W_{q,t}(E^-_{\overline{N},t})},$$

with $\underline{q} = \min_{x\in E}|q(x)|$ *and* $\overline{q} = \max_{x\in E}|q(x)|$.

Proof: We define

$$g(c) \quad = \quad \sum_{x\in E} (w_t a_t)(x)\exp(-c a_t(x)). \tag{10}$$

$g'(c) = -\sum_{x\in E}(w_t a_t^2)(x)\exp(-c a_t(x)) < 0$. Thus, $g(c)$ is strictly decreasing on $\mathbb{R}$. Since $\lim_{c\to+\infty} g(c) = -\infty$ (provided some $x \in E$ has $a_t(x) < 0$) and $\lim_{c\to-\infty} g(c) = +\infty$ (provided some $x \in E$ has $a_t(x) > 0$), eq. (7) has a single solution. We have:

$$\begin{aligned}\frac{g(0)}{Q_t} &= W_{q,t}(E^+_{\overline{N},t}) - W_{q,t}(E^-_{\overline{N},t}) + W_{q,t}(E_N)\\ &= \gamma_t W_{q,t}(E_{\overline{N}}) + W_{q,t}(E_N)\\ &= \gamma_t + (1-\gamma_t)W_{q,t}(E_N).\end{aligned} \tag{11}$$

We obtain $g(0) > 0$, which yields $c_t > 0$ in eq. (7). Moreover,

$$\begin{aligned}\frac{g(c_t)}{Q_t} = 0 = \quad &\textstyle\sum_{x\in E^+_{\overline{N},t}\cup E_N} w_{q,t}(x)\exp(-c_t|q(x)|)\\ &-\textstyle\sum_{x\in E^-_{\overline{N},t}} w_{q,t}(x)\exp(c_t|q(x)|).\end{aligned}$$

As $\underline{q} \leq |q(x)| \leq \overline{q}$ for all $x \in E$, we get

$$W_{q,t}(E^+_{\overline{N},t}\cup E_N)e^{-c_t\overline{q}} - W_{q,t}(E^-_{\overline{N},t})e^{c_t\overline{q}} \leq 0 \quad \text{and}$$
$$W_{q,t}(E^+_{\overline{N},t}\cup E_N)e^{-c_t\underline{q}} - W_{q,t}(E^-_{\overline{N},t})e^{c_t\underline{q}} \geq 0.$$

Solving for c_t yields the inequalities we claimed. □

Thus $c_t = r\ln((1 - W_{q,t}(E^-_{\overline{N},t}))/W_{q,t}(E^-_{\overline{N},t}))$ is solution to eq. (7), for some $r \in [1/2\overline{q}, 1/2\underline{q}]$. This bound is tighter than the one of Schapire and Singer (1999) (for whom $c_t \in \mathbb{R}$), and yields a very-fast dichotomic approximation to c_t. Suppose we want an approximation $\hat{c}_t$ such that $|\hat{c}_t - c_t|/|c_t| < \epsilon$ for some $0 < \epsilon \leq 1$. Then, it is is enough to make $\mathcal{O}(\ln(\overline{q}/\underline{q}) + \ln(1/\epsilon))$ dichotomic rounds. It is very convenient, given that the computation of $\hat{c}_t$ is repeated T times.

Lemma 3 $\forall t \geq 0$,

$$Z_t \leq \sqrt{1 - \left(\frac{\underline{q}\gamma_t}{\overline{q}}\right)^2} < \exp\left(-\frac{1}{2}\left(\frac{\underline{q}\gamma_t}{\overline{q}}\right)^2\right).$$

Proof: We have $\forall x \in [-1,1], \forall \eta \in \mathbb{R}$:

$$\exp(-\eta x) \leq \frac{1+x}{2}\exp(-\eta) + \frac{1-x}{2}\exp(\eta), \quad (12)$$

since function $x \mapsto \exp(-\eta x)$ is convex and the right-hand side is the equation of the line crossing the two points $(-1, \exp(\eta))$ and $(1, \exp(-\eta))$. Plugging $\eta = c_t\overline{q}$ and $x = a_t(x)/\overline{q}, \forall x \in E$, yields

$$\exp(-c_t a_t(x)) \leq \frac{\overline{q}+a_t(x)}{2\overline{q}}\exp(-c_t\overline{q}) + \frac{\overline{q}-a_t(x)}{2\overline{q}}\exp(c_t\overline{q}). \quad (13)$$

Let us name $\ell(x, c_t)$ the right-hand side of ineq. (13). We have

$$Z_t \quad \leq \quad \inf_{c\in\mathbb{R}} \sum_{x\in E} w_t(x)\ell(x,c). \quad (14)$$

The real c minimizing the right-hand side of (14) is

$$c \quad = \quad \frac{1}{2\overline{q}} \ln \frac{\overline{q} + \sum_{x\in E}(w_t a_t)(x)}{\overline{q} - \sum_{x\in E}(w_t a_t)(x)},$$

from which we get

$$Z_t \quad \leq \quad \sqrt{1 - \left(\frac{g(0)}{\overline{q}}\right)^2}, \quad (15)$$

with $g(.)$ defined as in eq. (10). Using the fact that $g(0) \geq \gamma_t Q_t$ (from eq. (11)), and $Q_t \geq \underline{q}$, we obtain the statement of the Lemma. $\square$

Provided $\underline{q}$ and $\overline{q}$ are constants (it shall be the case in our experiments), the maximal value of Z_t is also a constant < 1 under the WLA, thus yielding the exponential convergence of $\varepsilon(E_{\overline{N}}, H_T)$ towards 0.

4. An Oracle Simulation

We describe in this section an empirical approach for simulating an oracle, which must satisfy at best the theoretical behavior previously mentioned. The main objective is to ensure that a positive confidence given by the oracle means that the example is (very probably) correctly labeled. We provide here a neighborhood-based strategy allowing to geometrically distinguish suspicious words from the others.

4.1. Neighborhood-based Confidence

We think that the k nearest neighbor graph (kNN) (Cover & Hart, 1967) is suitable for treating such a problem. kNN's classification rule assesses the class of an unknown example x by computing the majority class among the k examples of E that are the closest to x. This method, beyond the fact that its limit error is bounded by twice the Bayesian error, is very tolerant to the presence of noisy data. Actually, an isolated outlier can only have a limited impact in the classification rule. The dual property, that we can deduce from the previous remark, is that a noisy data can be probably detected by analyzing its neighborhood. Let us define the margin function $m(x)$ of an example x of class $y(x)$, computable from a kNN graph:

$$m(x) = (N_{y(x)} - N_{\overline{y}(x)})/(N_{y(x)} + N_{\overline{y}(x)}),$$

where $N_{y(x)}$ (resp. $N_{\overline{y}(x)}$) denotes the number of examples among x's k nearest neighbors from the same class $y(x)$ (resp. the opposite class $\overline{y}(x)$). Hence an example with only neighbors from the opposite class (resp. from the same class) will receive -1 (resp. 1) as margin. Even if $m(x)$ seems to assess in a way the confidence value $q(x)$ used in the previous section, setting exactly $q(x) = m(x)$ would not be judicious. Actually, the kNN classifier allows "only" to ensure that the limit error will be bounded by twice the Bayesian error, that is not in fact our main concern. We aim here at providing a relevant measure of confidence, ensuring that a positive $q(x)$ means that x has a correct label in (100-ϵ) percent of cases (with ϵ small). If $m(x)$ is slightly higher than 0, this condition is far from being satisfied. To overcome this drawback, we fix a parameter $\beta \in [0, 1[$ and compute $q(x)$ as follows: $q(x) = (m(x) - \beta)/(1 - \beta)$ $\forall\ x$ such that $\beta \leq m(x) \leq 1$ (thus, $q(x) > 0$ iff $m(x) > \beta$); $q(x) = (m(x) - \beta)/(1 + \beta)$ $\forall\ x$ such that $-1 \leq m(x) \leq \beta$. Roughly speaking, high confidences are only given to examples having a large majority of neighbors from the same class. While some examples are probably (weakly) relevant, *i.e.* having a margin just slightly higher than 0, they will receive a negative confidence. This choice will be useful in grammatical inference, where reducing the impact of relevant data is better than inferring a DFA from noisy words.

4.2. A Probability-Based Distance Function

Before computing confidence values, we must use a relevant distance function between words. In this context, various distances have been studied, and the most popular one is probably the *edit distance*. However, we observed in preliminary experiments that we could obtain better results by using "data-oriented" distances, built after a transformation of the representation space. We decided to use bigram language models for describing words as vectors. Bigrams are a well-structured knowledge representation that has proven useful in the recognition of textual documents. This

Table 1. Error rates obtained on 11 datasets with 5, 10, 15 and 20% of noise.

		5%					10%				
DATASET	SIZE	kNN	RPNI	RPNI*	BOOST	PERF	kNN	RPNI	RPNI*	BOOST	PERF
AGARICUS	5644	0.0%	7.2%	7.2%	4.3%	0.0%	0.0%	14.0%	10.0%	6.9%	0.0%
TICTACTOE	809	9.2%	39.5%	34.5%	28.3%	4.9%	16.0%	41.9%	40.7%	34.5%	10.5%
USF	1871	16.5%	33.6%	31.2%	26.1%	26.4%	16.8%	35.7%	33.0%	31.2%	31.7%
WF	1887	16.4%	29.3%	25.3%	21.4%	20.6%	19.3%	29.8%	31.2%	22.7%	26.1%
BASE1	2540	24.8%	46.4%	41.5%	41.1%	35.6%	27.8%	52.1%	44.8%	43.7%	36.2%
BASE2	1505	9.6%	48.9%	21.9%	19.9%	14.3%	9.6%	39.2%	13.6%	30.5%	12.3%
BASE3	1532	10.4%	43.9%	27.0%	26.7%	13.0%	11.0%	39.0%	39.0%	12.7%	12.7%
BASE4	2969	0.5%	39.7%	2.0%	10.0%	0.0%	1.0%	45.7%	29.2%	23.4%	0.0%
BASE5	2179	0.9%	36.7%	36.7%	22.2%	19.2%	1.3%	45.9%	18.6%	25.0%	16.2%
BASE6	2004	17.7%	42.9%	7.0%	13.0%	2.0%	21.6%	45.9%	42.9%	25.6%	1.2%
BASE7	10000	23.0%	46.1%	39.7%	37.2%	39.7%	25.1%	49.4%	42.7%	43.7%	39.4%
AVERAGE	2994.5	11.7%	37.7 %	24.9%	22.7%	16.0%	13.6%	39.9%	31.4 %	27.3%	16.9%
		15%					20%				
AGARICUS	5644	1.0%	14.0%	10.0%	6.1%	0.0%	2.0%	23.2%	18.2%	11.9%	0.0%
TICTACTOE	809	15.4%	44.0%	40.7%	40.7%	9.8%	17.9%	40.1%	40.1%	38.8%	11.7%
USF	1871	20.5%	34.9%	32.0%	27.7%	27.7%	23.7%	48.8%	32.2%	30.6%	28.8%
WF	1887	20.4%	32.5%	36.2%	24.6%	24.6%	22.2%	35.4%	32.0%	31.7%	22.7%
BASE1	2540	28.2%	47.0%	42.9%	42.3%	38.5%	31.1%	51.2%	45.7%	45.2%	35.2%
BASE2	1505	12.9%	39.8%	39.8%	37.5%	10.6%	16.9%	46.8%	46.8%	45.1%	12.6%
BASE3	1532	12.7%	47.0%	39.4%	35.5%	14.9%	18.5%	49.5%	49.5%	41.6%	20.5%
BASE4	2969	1.5%	44.2%	16.0%	26.9%	0.0%	3.3%	43.4%	35.0%	30.4%	0.0%
BASE5	2179	2.0%	44.5%	43.8%	31.4%	15.1%	2.2%	41.0%	39.4%	34.6%	9.4%
BASE6	2004	24.9%	50.6%	42.9%	33.1%	1.7%	28.9%	46.6%	41.9%	39.2%	1.0%
BASE7	10000	27.1%	48.0%	40.7%	43.2%	40.7%	29.2%	48.0%	44.3%	41.8%	40.5%
AVERAGE	2994.5	15.1%	40.6%	34.9%	31.7%	16.7%	17.8%	43.1%	38.6%	35.5%	16.6%

makes it possible to apply geometric, and other mathematical techniques, which are well defined for vectors, but not for words in general. The probability of a letter of a word depends only on the identity of the preceding letter. Given a word with n letters $x = l_1 l_2 l_3 \ldots l_n$, $p(x) = \prod_{i=1}^{n} p(l_i/l_{i-1})$. To make $p(l_1/l_0)$ meaningful, a distinguished token <BOW> (for Beginning Of the Word) is added and we take l_0 as <BOW>. Here, we aim at computing an euclidean distance from a two-dimension numerical representation of words. Let us define $P_+(x)$ and $P_-(x)$, $P_b(x)$ denoting the probability of x relatively to the subset E_b. To allow comparisons between words of different lengths, we normalized each probability using the geometric mean, as follows: $P'_b(x) = \sqrt[n+1]{P_b(x)}$. Given $P'_+(x)$ and $P'_-(x)$, we can now (i) see x as a point in an euclidean space with coordinates $P'_+(x)$ and $P'_-(x)$, (ii) compute the euclidean distance, and (iii) so build the kNN graph.

5. Experiments and Results

The motivation of our experiments was not only to validate our theoretical results but also to assess the performances of the kNN graph as a good candidate for simulating an oracle. To achieve these tasks, we used three types of datasets. The first one concerns well known benchmarks of the UCI repository: AGARICUS and TICTACTOE. The second represents real-world datasets: a first one (USF) comes from the US Social Security Administration and takes the census of the first thousand first names the more frequently given in the USA in 2002; a second one (WF) contains all the worldwide first names that begin with the letter "a". For both of them, the concept to learn is the gender of the individuals. The last type of datasets concerns artificial ones. We generated 7 synthetic databases (BASE1,...,BASE7) made of random words with variable length and labeled according if they contain *a priori* fixed patterns or not. To test our method on a large dataset, BASE7 contains 10000 words. For all the 11 databases, we introduced noise, by switching the class of a given percentage of words and we assessed the error rate using a noise-free test set.

Table 1 compares RPNI, RPNI* and our boosting approach (BOOST) after 200 iterations for the different levels of noise. Two other columns have been added in this Table. The first one (kNN) describes the performances of the kNN graph as a single classifier (*i.e.* without boosting, without DFA). The interest of these results is not only to show how well a kNN graph simulates the oracle, but also to estimate the gain obtainable using it in a boosting procedure. The second column (PERF) presents the results obtained from a perfect oracle, *i.e.* one that always knows if a data is noisy or not. The goal of this column is to establish the optimal (but unobtainable) error rate for each database. This experimental study was achieved with 5, 10, 15 and 20% of noise. We tried different values of k (for the kNN graph) and kept the optimal value (assessed by cross-validation) for each database.

The following remarks can be made. First of all, the excellent results obtained in the column kNN justify the use of the kNN graph as a simulated oracle. While RPNI* constituted a first efficient solution for improving RPNI (32.5% versus 40.3% on average on all the noise levels), our new oracle-based boosting procedure results in a dramatic improvement of the performances. Actually, the additional information provided by the kNN graph, and used in our boosting method, permits to significantly decrease RPNI*'s error rate (29.3% versus 32.5% on average). This difference is statistically significant using a Student paired-t test. Another interesting remark is that the use of our simulated oracle results in final hypotheses whose performances are not so far, after 200 iterations, from the ones issued from the perfect oracle (29.3% versus 16.5% on average). Finally, note that the previously described behaviors remain almost the same even with the increase of the noise level.

In order to show the exponential convergence of the generalization error all along the iterations, Figure 4 describes the behavior of the algorithm on 4 characteristic databases (for 5% of noise). It brings to the fore the rapid effect of boosting on the performances (often before the 20^{th} iteration).

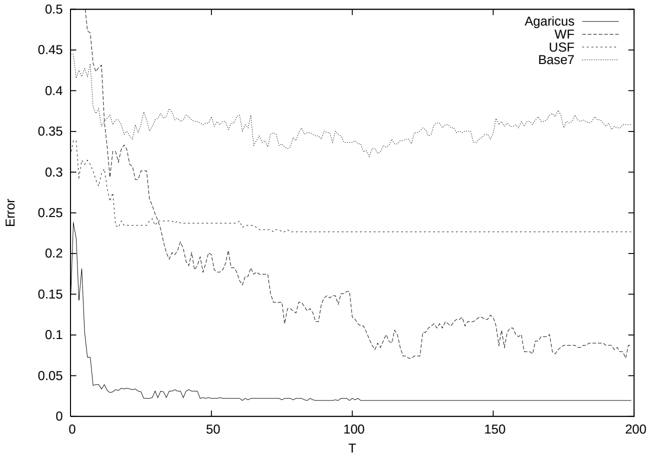

Figure 4. Generalization error in function of iterations.

6. Conclusion

We introduced a new boosting scheme that allowed to improve the performances of a grammatical inference algorithm in the presence of noisy data. As far as we know, our approach on using a real-valued confidence oracle for boosting is new, even outside grammatical inference, and deserves to be applied with other weak learners. Moreover, we proposed a heuristic based on neighborhoods for efficiently building this oracle. The study of other types of oracles, such as hidden Markov models, also deserves further investigations.

References

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. 12^{th} *Int. Conf. on Computational Learning Theory* (pp. 92–100).

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, *13*, 21–27.

de la Higuera, C. (2004). A bibliographic survey on grammatical inference. *Pattern Recognition*. To appear.

Domingo, C., & Watanabe, O. (2000). Madaboost: a modification of adaboost. *Third Annual Conf. on Computational Learning Theory* (pp. 180–189).

Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine Learning*, *43*, 293–318.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithms. *Thirteenth Int. Conf. on Machine Learning* (pp. 148–156).

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

Friedman, J., Hastie, T., & Tibshirani, R. (1998). *Additive logistic regression: a statistical view of boosting* (Technical Report).

Kearns, M. J., & Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. M.I.T. Press.

Kivinen, J., & Warmuth, M. (1999). Boosting as entropy projection. *Twelfth Int. Conf. on Computational Learning Theory* (pp. 134–144).

Lang, K., Pearlmutter, B., & Price, R. (1998). Results of the abbadingo one DFA learning competition. *Fourth Int. Colloquium on Grammatical Inference* (pp. 1–12).

Oncina, J., & García, P. (1992). *Inferring regular languages in polynomial update time*, vol. 1 of *Machine Perception and Artificial Intelligence*, 49–61. World Scientific.

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

Sebban, M., & Janodet, J. (2003). On state merging in grammatical: a statistical approach for dealing with noisy data. *Twentieth Int. Conf. on Machine Learning* (pp. 688–695).