Rapport Technique GRAPPA
GRAPPA-0104

# Boosting Correct Least General Generalizations

Fabien Torre

GRAppA - Université Charles de Gaulle - Lille 3

**Résumé**

Nous explorons dans cet article l'utilisation de moindres généralisés corrects comme apprenant faible dans des techniques de *boosting* [FS96]. Nous établissons dans un premier temps que cette idée est pertinente : les expérimentations sur des problèmes classiques de l'UCI [BM98] montrent qu'AdaBoost instancié avec un apprenant faible à base de moindres généralisés obtient des taux d'erreur plus faibles que C4.5, GloBo [Tor99] (système utilisant les moindres généralisés mais sans *boosting*) et AdaBoost muni d'un apprenant faible plus classique. Puis, devant la constatation que notre nouvel apprenant faible peut être coûteux en temps de calcul, nous proposons un mode de génération des hypothèses qui peut être distribué sur différentes machines et une définition du poids des hypothèses *a posteriori*. Cela aboutit au nouvel algorithme GloBoost et, à nouveau, les expérimentations nous encouragent dans cette voie : GloBoost obtient des performances comparables à celles d'AdaBoost. Nous discutons finalement des résultats obtenus, de leur signification vis-à-vis du *boosting* et des perspectives ouvertes.

**Abstract**

The primary goal of this paper is to propose a new weak learner, namely *least general generalization* and to boost it using an algorithm that does not manage a weight distribution over the training examples and computes hypotheses weights *a posteriori*. First, experiments conducted on benchmarks from the UCI Repository [BM98] show that AdaBoost boosting *least general generalization* obtains smaller error than reference systems. The computation time needed by these experiments leads us then to define a method that could be easily distributed. This method, called GloBoost, is able to produce hypotheses independently of one another and then give a weight to each produced hypothesis. New experiments are then conducted and show low error rates for both AdaBoost and GloBoost. Moreover, GloBoost has the advantage to be naturally distributable to different computers.

# Boosting Correct Least General Generalizations

Fabien Torre

## 1 Introduction

AdaBoost [FS97] works by setting a weight for each example and then by calling a base learner on the weighted examples to obtain an hypothesis that concentrates on examples with great weights. Then, examples weights are modified with regard to this hypothesis (weights of well classified examples are decreased and those of misclassified are increased) and the process is iterated. More formally, Algorithm 1 shows AdaBoost for two classes and weak hypotheses that abstain [SS99]. This version defines three quantities: $W_+$ the sum of weights of examples that are well classified, $W_-$ the sum of weights of misclassified examples and $W_0$ the sum of weights of examples on which the hypothesis abstains.

**Algorithm 1** AdaBoost

**Input:** $n$ examples $x_i$ and their classes $y_i \in \{-1, +1\}$ ; $T$ a number of rounds of boosting ; $A$ a weak learner taking as input a set of weighted and labelled examples.

**Output:** $H$ the final classifier.

**for** $i = 1$ to $n$ **do**
  $w_i = 1/n$ {initialization of examples weights}
**end for**
**for** $t = 1$ to $T$ **do**
  $h_t = A(\{(x_i, y_i, w_i)\})$
  for $b \in \{-, 0, +\} : W_b = \sum_{i:\text{sign}(y_i h_t(x_i))=b} w_i$
  $\alpha_t = \frac{1}{2}\log\left(\frac{W_+ + \frac{1}{2}W_0}{W_- + \frac{1}{2}W_0}\right)$
  $Z_t = \sum_i^n [w_i exp(-\alpha_t y_i h_t(x_i))]$
  **for** $i = 1$ to $n$ **do**
    $w_i = \frac{w_i exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
  **end for**
**end for**
return $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Historically, AdaBoost comes from the PAC framework [Val84] in which a strong result has been shown: every learning algorithm that provides hypotheses a little bit better than random guessing can be improved to achieve an error as small as wanted and this, in polynomial time [Sch90, Fre95]. However, real data do not likely fit theoretical conditions that guarantee an effective boosting in the PAC framework: one can neither be sure to have enough examples, nor know if the number of rounds of boosting is great enough, nor guarantee that the base learner used is a weak learner (i.e. it provides hypotheses better than random guessing for any distribution on the training examples).

In spite of these objections, many experimental evaluations have shown that AdaBoost effectively achieves small generalization errors in a wide variety of domains [FS96, SS00]. Many works aimed to explain these good performances. For instance, [Bre96a] analyses AdaBoost's performances by decomposing the error in bias and variance and claims that AdaBoost decreases the bias component of the error, like all *arcing classifiers* and like *bagging* [Bre96b]. This idea is criticized in [FS98] where AdaBoost is shown decreasing both bias and variance. Another proposed explanation is that boosting tends to produce large margins [SFBL97]. Following this idea, new algorithms have been proposed to explicitly maximize the margin [RW02, RW03, GS98, Har99] but these methods are finally outperformed by AdaBoost. A last approach consists in analyzing AdaBoost in the framework of Monte Carlo Theory [ES03]. This work suggests explanations on how AdaBoost, and more generally, *ensemble learning* methods work.

Another open issue is how to choose the weak learner to be used? The authors of AdaBoost indicate that a good weak learner is *an algorithm that gives small errors when run under Adaboost* [FS98]. [Bre96a] gives a more practical answer by claiming that a weak learner must be unstable: a good weak learner must show large variations in the learned hypotheses when the training set undergoes small variations. In this paper, we defend the idea that a weak learner can be unstable even if the produced hypotheses misclassify no training example. We precisely propose to use a weak learner that provides only correct hypotheses: such an hypothesis can abstain on some examples, but not be wrong. These correct hypotheses are the *correct least general generalizations*.

The *least general generalizations* have been extensively used in machine learning. In particular, they allow the construction of understandable theories. The main drawback is the computation time needed by correct least general generalizations when the number of available examples grows. The use of an expensive weak learner raises the problem of distributing the computation made by AdaBoost (and now this parallelization is difficult since AdaBoost computes an hypothesis in terms of previous ones), and weighting the weak hypotheses *a posteriori*.

An observation is often repeated about AdaBoost: the generalization error continues to decrease far beyond the point in which the training error reaches zero. Moreover, we have observed on very simple problems that the generalization error still decreases after all possible hypotheses have been produced by the weak learner. In other words, an hypothesis can appear several times and will finally votes with a weight corresponding to the sum of all its $\alpha_t$. This sum seems to be a best weight than any $\alpha_t$ and one can hope to compute such a weight without the AdaBoost adaptative process.

The paper is organized as follows. Section 2 introduces the *correct least general generalization* and its use as a weak learner in AdaBoost. Section 3 presents experimental results, together with results of more usual systems, that justify our interest in *least general generalization*. In section 4, the replacement of AdaBoost by a method called GloBoost is proposed ; GloBoost produces weak hypotheses and defines their weights *a posteriori*. Section 5 describes experimental results for GloBoost. Finally, section 6 presents a summary of our work and some perspectives for further research.

# 2 Boosting Least General Generalizations

## 2.1 Correct least general generalizations

Let us start by defining the *least general generalization* (*lgg*) of two examples as the most specific hypothesis that covers them. With an attribute-value representation, this hypothesis is unique and can be defined as the conjunction of least general generalizations of each attribute:

- the generalization of two discrete values $v_1$ and $v_2$ is $v_1$ if $v_1 = v_2$, otherwise the undefined value `?`,
- the generalization of two continuous values $v_1$ and $v_2$ is the smallest interval that contains the two values, namely $[v_1, v_2]$.

The following example describes this computation:

| | **age** | **smoker** | **sex** | ***class*** |
|---|---|---|---|---|
| $\mathbf{e_1}$ | 25 | no | male | positive |
| $\mathbf{e_2}$ | 35 | no | female | positive |
| $\mathbf{g_1}$ | $[25, 35]$ | no | ? | positive |

We can then use *lgg* to generalize an hypothesis and an example:

| | **age** | **smoker** | **sex** | ***class*** |
|---|---|---|---|---|
| $\mathbf{g_1}$ | $[25, 35]$ | no | ? | positive |
| $\mathbf{e_3}$ | 30 | yes | male | positive |
| $\mathbf{g_2}$ | $[25, 35]$ | ? | ? | positive |
| $\mathbf{e_4}$ | 40 | no | female | positive |
| $\mathbf{g_3}$ | $[25, 40]$ | ? | ? | positive |

From the supervised learning point of view, it is interesting to generalize together examples of the same class, but being careful not to subsume examples of other classes. This method formally defined by Algorithm 2 provides a *correct least general generalization* (*clgg*) that is moreover *maximally correct* (i.e. adding another training example involves the loss of the correctness). Let us also remark that the produced hypothesis depends on the order in which the examples are considered.

**Algorithm 2** Correct_LGG_1

**Input:** $s$ a seed example, $P = \{p_1, \ldots, p_n\}$ an *ordered* set of $n$ examples of the class of the seed, $N$ a set of counterexamples. The algorithm uses the `LGG` operation and a subsumption test $\succeq$ (to decide whether an hypothesis covers an example or not).

**Output:** $g$ a generalization of $P$, maximally correct with respect to $N$.

```
g = s
for i = 1 to n do
  g' = LGG(g, p_i)
  if (∀n ∈ N : NOT(g' ⪰ n)) then
    g = g' {if g' is correct, it becomes the current generalization.}
  end if
end for
return g
```

The next example shows three possible executions of the Algorithm 2 (examples that cannot be generalized without losing the correctness are underlined):

| seed | other positive examples | $\rightarrow$ | maximally correct cluster |
|---|---|---|---|
| $p_1$ | $p_5\ p_8\ \underline{p_2}\ p_{14} \ldots$ | $\rightarrow$ | $\{p_1, p_5, p_8, p_{14}, \ldots\}$ |
| $p_2$ | $\underline{p_{14}}\ p_3\ \underline{p_1}\ p_{12} \ldots$ | $\rightarrow$ | $\{p_2, p_3, p_{12}, \ldots\}$ |
| $p_3$ | $p_7\ \underline{p_4}\ \underline{p_{18}}\ \underline{p_{12}} \ldots$ | $\rightarrow$ | $\{p_3, p_7, \ldots\}$ |

This example shows that the order of positive candidates changes for each call to the algorithm and also that correctness property leads to generalizations that alone do not cover all the positive examples. In other words, such a generalization can only conclude on a single class, the class of the seed. For all other examples (examples of different class or examples of the seed class but not involved into the generalization), the hypothesis abstains.

This simple idea has been used to define an AQ-like learning algorithm named GloBo. This method uses every example as seed and try to add to this seed all examples of the same class in a random order. Finally, GloBo only keeps the minimal set of hypotheses that covers all training examples. The aim of this minimal covering is to produce an understandable theory. This approach has been validated during the PTE Challenge [SKB99] where GloBo has been shown as the best system among those providing understandable theories.

In this paper, we renounce the understandability in order to boost *clgg* and thus improve predictive performances.

## 2.2 Definition of a new weak learner

Algorithm 3 describes an adapted algorithm to compute *correct least general generalizations*, which is designed to be a weak learner for AdaBoost. For this purpose, the method has to take examples weights into account in order to first generalize examples with great weights. Let us emphasize that produced hypotheses must still be correct: whatever the weight of a negative example, generalizations are not allowed to cover it. Finally, positive examples are merely sorted by decreasing weights and then Algorithm 2 is reused.

**Algorithm 3** Correct_LGG_2 (Weak Learner)

**Input:** $n$ labelled and weighted examples $(x_i, y_i, w_i)$.
**Output:** $g$ a *clgg* that covers examples with important weights.
*target* = a randomly chosen class
*seed* = the example of class *target* that has the greatest weight
$P = \{x_i | y_i = target, x_i \neq seed\}$
$N = \{x_i | y_i \neq target\}$
***sort*** $P$ ***by decreasing weights***
return `Correct_LGG_1`(*seed*,$P$,$N$) {Call to Algorithm 2}

This weak learner can be used by AdaBoost (Algorithm 1). It allows to simplify the definition of $\alpha_t$ (hypotheses weights) since a *clgg* either well classifies a training example or abstains, but never makes a mistake. In other words, $W_- = 0$, so $W_0 = 1 - W_+$ and finally

$$\alpha_t = \frac{1}{2}\log\left(\frac{1+W_+}{1-W_+}\right)$$

The next section presents some experimental results obtained by AdaBoost boosting *clgg*.

# 3 Experiments with AdaBoost

This section describes some experiments that we ran on the following algorithms: C4.5, GloBo, a system using *clgg* but without boosting, AdaBoost boosting *decision stumps* [FS96] and the method we have described, namely AdaBoost boosting *clgg*. We compared these methods on twenty benchmark problems available from the UCI Repository [BM98]. For every problem, 10-fold cross validation is used*. Let us specify that randomized algorithms (GloBo and AdaBoost with *clgg*) are rerun 10 times on each fold and the results averaged.

Results for 100 rounds of boosting are shown in Table 1. Moreover, this table indicates, for each algorithm, the error rates averaged over all datasets. AdaBoost with *clgg* is better than C4.5 for 8 datasets, better than GloBo for 15 datasets. and better than AdaBoost with stumps for only 5 datasets.

Table 2 shows results for similar experiments where each version of AdaBoost is run for 1000 rounds. These results improve our first experiments. For 1000 rounds of boosting, AdaBoost with *clgg* outperforms C4.5 on 13 datasets and GloBo on all datasets. Furthermore, *clgg* clearly appears now like the best weak learner on most of the datasets (*clgg* outperforms stumps on 12 datasets).

With regard to 100 rounds of boosting, performances get better for both decision stumps and *clgg*. Nevertheless, this behavior is more obvious in the case of *clgg*: whereas decision stumps make the results worse for 8 datasets, *clgg* improves its performances on all datasets.

These observations encourage us to increase again the number of rounds of boosting. Unfortunately, *clgg* is a complex operation that needs much computation time:

*The datasets and cross validation files used can be found at the following url: http://www.grappa.univ-lille3.fr/~torre/Recherche/Experiments/.

Table 1: Error rates (%) of C4.5, GLOBO and ADABOOST for twenty problems from the UCI Repository. The weak learner used by ADABOOST is either the decision stumps or *clgg*; each version is run for 100 rounds of boosting.

| | | | **Adaboost** | |
|---|---|---|---|---|
| **Dataset** | **C4.5** | **GLOBO** | **stumps** | **clgg** |
| audiology | 18.20 | 20.76 | **13.43** | 25.78 |
| breast-cancer | 4.87 | 3.89 | 4.16 | **3.02** |
| car | **7.69** | 10.17 | 13.42 | 17.92 |
| cmc | 48.07 | 50.60 | **44.80** | 56.32 |
| crx | **14.79** | 16.50 | 15.80 | 17.82 |
| dermatology | 6.23 | 8.51 | **3.51** | 6.58 |
| ecoli | 15.89 | 23.88 | **15.59** | 23.24 |
| glass | 28.72 | 5.57 | 23.62 | **4.70** |
| hepatitis | 20.70 | 20.09 | **17.64** | 17.88 |
| horse-colic | **13.63** | 22.29 | 18.44 | 20.90 |
| house-votes-84 | **3.22** | 7.39 | 4.62 | 4.78 |
| ionosphere | 7.96 | 8.74 | **6.78** | 8.51 |
| iris | **5.33** | 7.47 | **5.33** | 5.93 |
| pima | 29.29 | 27.04 | **25.13** | 29.68 |
| promoters | 18.17 | 22.60 | **6.83** | 12.93 |
| sonar | 28.97 | 31.09 | **16.86** | 25.25 |
| tic-tac-toe | 14.40 | 1.11 | 13.78 | **0.23** |
| vowel | 21.21 | 23.99 | **20.71** | 21.30 |
| wine | 8.83 | 8.94 | 14.18 | **5.41** |
| zoo | 7.51 | 3.89 | 3.55 | **3.27** |
| *Average* | *16.18* | *16.23* | ***14.41*** | *15.57* |

Table 2: Error rates (%) of the two versions of AdaBoost for 100 and 1000 rounds.

| **Problems** | **stumps** | | **clgg** | |
|---|---|---|---|---|
| | **100** | **1000** | **100** | **1000** |
| audiology | **13.43** | 15.79 | 25.78 | 19.22 |
| breast-cancer | 4.16 | 4.16 | **3.02** | 3.10 |
| car | 13.42 | 13.36 | 17.92 | **9.74** |
| cmc | **44.80** | 44.87 | 56.32 | 49.61 |
| crx | 15.80 | 16.53 | 17.82 | **14.47** |
| dermatology | **3.51** | 4.38 | 6.58 | 4.63 |
| ecoli | **15.59** | **15.59** | 23.24 | 19.25 |
| glass | 23.62 | 22.53 | 4.70 | **4.59** |
| hepatitis | **17.64** | 18.23 | 17.88 | 18.39 |
| horse-colic | 18.44 | 21.71 | 20.90 | **18.42** |
| house-votes-84 | **4.62** | 5.31 | 4.78 | 6.12 |
| ionosphere | **6.78** | 12.19 | 8.51 | 7.44 |
| iris | **5.33** | **5.33** | 5.93 | **5.33** |
| pima | 25.13 | 25.39 | 29.68 | **25.13** |
| promoters | **6.83** | 18.67 | 12.93 | 7.00 |
| sonar | 16.86 | **15.03** | 25.25 | 23.34 |
| tic-tac-toe | 13.78 | 1.67 | 0.23 | **0.00** |
| vowel | 20.71 | 14.65 | 21.30 | **9.30** |
| wine | 14.18 | 18.07 | **5.41** | 5.73 |
| zoo | 3.55 | 3.55 | **3.27** | 3.32 |
| *Average* | *14.41* | *14.85* | *15.57* | ***12.71*** |

to build a single hypothesis, one must try to generalize the seed with each example of the target class and validate this attempt by checking that no counterexample is subsumed by the resulting generalization. So *clgg* has a quadratic time complexity in the number of examples.

Consequently, we need a more efficient way of using *clgg*. One way consists in distributing the *clgg* production to different computers and then affecting a weight to each hypothesis. This is the subject of the next section.

# 4 Boosting *clgg* with GloBoost

This section presents a method to produce weighted hypotheses independently of one another. This aim seems very difficult to reach using AdaBoost. For this purpose, we need to be able to produce hypotheses independently and then to associate a weight to each of these hypotheses. Let us start with the latter goal by defining the weights of available hypotheses, regardless the way used to obtain them.

## 4.1 Definition *a posteriori* of hypotheses weights

The first weight named *adalike* aims to simulate the AdaBoost process. Like in AdaBoost, $w_i$ the weight of an example $x_i$ is first defined. It uses $T_i$ the number of hypotheses that cover this example:

$$w_i = \exp(-T_i)$$

This formula refers to AdaBoost that multiplies the weight of an example by $\exp(-\alpha_t)$ for each hypothesis $h_t$ covering the example. Once these weights are defined, they are normalized by dividing each of them by $Z = \sum_i w_i$. Now, the weight of an hypothesis $h$ can be defined: $W_+$ is the sum of weights of examples that are subsumed by $h$ and $n$ is the number of times this hypothesis is appeared during the production. Finally, the weight *adalike* of $h$ is defined as follows:

$$\alpha(h) = \frac{n}{2}\log\left(\frac{1+W_+}{1-W_+}\right)$$

$n$ appears here to simulate the sum of weights done by AdaBoost when the weak learner gives the same hypothesis several times.

Three simpler weights are also considered:

- *covering*: each distinct hypothesis $h$ votes with a weight corresponding to the number of training examples covered by $h$,
- *frequency*: each distinct hypothesis $h$ votes with a weight which is the number of times $h$ appeared during the production process,
- *uniform*: all distinct hypotheses vote with the same weight.

## 4.2 Generating hypotheses without AdaBoost

The next step is to produce hypotheses without using AdaBoost. For this purpose, we propose GloBoost (Algorithm 4) that produces *clgg* independently of one another: such a generalization is built on a randomly chosen seed and examples of the same class randomly shuffled. When hypotheses has been produced, GloBoost computes weights for each of them. In the long run, GloBoost will allow us to distribute the production of hypotheses to several computers.

**Algorithm 4** GloBoost

**Input:** $n$ labelled examples $(x_i, y_i)$ ; $T$ the number of rounds.
**Output:** $H$ the final classifier.

```
for t = 1 to T do
    target = a class randomly selected
    seed = an example randomly selected in the target class
    P = {x_i | y_i = target, x_i ≠ seed}
    N = {x_i | y_i ≠ target}
    randomly shuffle P
    h_t = Correct_LGG_1(seed,P,N) {Call to Algorithm 2, page 5}
end for
for all t such as 1 ≤ t ≤ T do
    α_t = weight(h_t) {function to be defined}
end for
return H(x) = sign( Σ_{t=1}^{T} α_t . h_t(x) )
```

# 5 Experiments

In this section, previous propositions are evaluated: generation of hypotheses by GloBoost and definition of weights *a posteriori*. Experimental protocol and datasets are identical with those described in section 3.

## 5.1 Experimental evaluation of weights

First AdaBoost is used to produce hypotheses. This boosting phase provides a set of hypotheses weighted by AdaBoost and the described weights are then computed for each of these hypotheses. In other words, this procedure provides several voting systems all built on the same set of hypotheses. Our intuition based on the different works that aim to maximize the margin [SFBL97, RW02, RW03, GS98, Har99], is that AdaBoost behaves optimally in the tuning of hypotheses weights. Consequently, this evaluation aims to measure the loss produced by leaving the weights of AdaBoost.

Results for 100 and 1000 rounds of boosting are shown in Table 3. First, one can observe that AdaBoost is the best to define the weights of hypotheses produced by itself, in particular when the number of rounds of boosting grows. The *adalike* weight, by 100 rounds of boosting, obtains the smallest errors for nine datasets but

Table 3: Error rates in terms of weights associated to *clgg* produced by AdaBoost.

| | Dataset | adaboost | adalike | covering | frequency | uniform |
|---|---|---|---|---|---|---|
| **100 rounds of boosting** | audiology | 25.78 | **24.14** | 28.32 | 25.97 | 26.25 |
| | breast-cancer | 3.02 | 4.17 | **2.96** | 4.63 | 4.65 |
| | car | 17.92 | **17.89** | 18.14 | 18.39 | 18.39 |
| | cmc | 56.32 | 56.31 | **56.27** | 56.62 | 56.62 |
| | crx | 17.82 | **17.12** | 17.77 | 20.45 | 20.43 |
| | dermatology | 6.58 | **6.55** | 7.20 | 7.18 | 7.32 |
| | ecoli | **23.24** | 24.21 | 25.25 | 25.37 | 25.32 |
| | glass | **4.70** | **4.70** | 4.83 | **4.70** | 4.83 |
| | hepatitis | 17.88 | **15.65** | 18.98 | 20.67 | 20.72 |
| | horse-colic | **20.90** | 21.06 | 21.24 | 23.16 | 23.40 |
| | house-votes-84 | **4.78** | 23.51 | 4.91 | 30.94 | 6.21 |
| | ionosphere | 8.51 | 27.95 | 9.61 | **8.14** | **8.14** |
| | iris | 5.93 | **5.07** | 6.00 | 6.27 | 6.27 |
| | pima | 29.68 | **29.61** | 30.23 | 32.46 | 32.46 |
| | promoters | **12.93** | 16.12 | 16.45 | 15.50 | 15.33 |
| | sonar | 25.25 | 40.70 | 25.28 | **24.26** | **24.26** |
| | tic-tac-toe | **0.23** | **0.23** | **0.23** | 0.71 | 1.13 |
| | vowel | 21.30 | 22.60 | 21.38 | **20.14** | **20.14** |
| | wine | 5.41 | 16.40 | **5.35** | 5.46 | **5.35** |
| | zoo | 3.27 | 3.60 | 4.02 | **3.16** | 4.02 |
| | *Average* | ***15.57*** | *18.88* | *16.22* | *17.71* | *16.56* |
| **1000 rounds of boosting** | audiology | 19.22 | **19.17** | 26.89 | 20.66 | 21.72 |
| | breast-cancer | 3.10 | 10.36 | **3.00** | 3.96 | 3.94 |
| | car | 9.74 | **9.56** | 9.74 | 15.83 | 10.20 |
| | cmc | 49.61 | 49.24 | **48.98** | 51.25 | 51.03 |
| | crx | **14.47** | 26.50 | 14.60 | 21.22 | 20.71 |
| | dermatology | **4.63** | 12.93 | 5.92 | 5.51 | 7.15 |
| | ecoli | **19.25** | 23.92 | 21.02 | 22.46 | 21.62 |
| | glass | **4.59** | 8.65 | 6.33 | **4.59** | 6.33 |
| | hepatitis | **18.39** | 22.74 | 19.28 | 21.21 | 23.06 |
| | horse-colic | 18.42 | **18.17** | 19.54 | 19.80 | 18.86 |
| | house-votes-84 | 6.12 | 34.84 | **5.07** | 30.06 | 5.80 |
| | ionosphere | **7.44** | 57.79 | 9.23 | 7.64 | 7.56 |
| | iris | **5.33** | 6.67 | 6.00 | 6.00 | 6.80 |
| | pima | 25.13 | **24.75** | 25.55 | 29.37 | 29.39 |
| | promoters | **7.00** | 29.38 | 12.83 | 8.02 | 13.63 |
| | sonar | 23.34 | 49.52 | 22.53 | **21.83** | **21.83** |
| | tic-tac-toe | **0.00** | **0.00** | **0.00** | 0.72 | 2.71 |
| | vowel | 9.30 | 34.43 | 9.85 | 9.17 | **9.16** |
| | wine | 5.73 | 45.90 | **4.62** | 5.23 | 4.67 |
| | zoo | **3.32** | 4.30 | 3.91 | 3.55 | 3.99 |
| | *Average* | ***12.71*** | *24.44* | *13.74* | *15.40* | *14.51* |

it behaves very badly on five other problems. Unfortunately, this behavior becomes worse for 1000 rounds of boosting. The weights *covering* and *frequency* obtain interesting results. In particular, the *covering* weight seems to be very close to the AdaBoost one. It should be observed that the proposed weights are just defeated by the AdaBoost's weight and they outperform both C4.5 and GloBo evaluated in section 3, Table 1. In conclusion of this experiments, reasonable weights are now available to replace the AdaBoost one.

## 5.2 Experimental evaluation of GloBoost

GloBoost is now evaluated with the different available weights: *adalike*, *covering*, *frequency* and *uniform*. Results are shown in Table 4.

On the one hand, one can observe once again aberrant errors of the *adalike* weight. On the other hand, *covering* and *frequency* weights confirm their interest so much so that they outperform AdaBoost on some datasets. Furthermore, this tendency is emphasized for 1000 rounds of boosting. This result is very surprising: these weights increased the error of AdaBoost when this one generates the hypotheses and one would expect that the error increases again when AdaBoost is replaced by purely randomized generator. On the contrary, these weights appear to be more adapted to this mode of production.

Finally, let us precise that in the case of GloBoost, covering and frequency are very correlated and that explains why the performances of these two weights are similar.

# 6 Conclusion and future works

Two new weak learners have been defined that both provide *correct least general generalizations*. The former, that favors hypotheses covering examples with great weights, has been used with AdaBoost. The latter randomly computes correct hypotheses which are then weighted by our new system called GloBoost. The GloBoost system currently uses the *frequency* weight. This weight allows good voting systems, and what is more, can be computed for each hypothesis without considering the others.

Finally, experiments have been conducted in order to evaluate this system. Table 5 summarizes the experimental results obtained for the main systems compared in this paper. Surprisingly, the generalization error achieved by GloBoost is similar to the AdaBoost one, with the advantage that the algorithm of GloBoost can be easily distributed.

First, it would be interesting to define some more randomized weak learners in order to increase their instability and observe consequences on the generalization error. Moreover, this randomization could be easier for weak learners computing *least general generalization*.

Another issue discussed in this paper is one of hypotheses weights: how the weight associated to an hypothesis by AdaBoost can be computed by a non-adaptative process? We suppose that the use of correct weak hypotheses can also make easier this task. In particular, we have to explain why the *adalike* weight behaves badly

Table 4: Error rates (%) of GloBoost with the different weights.

| | Dataset | adalike | covering | frequency | uniform |
|---|---|---|---|---|---|
| **100 rounds** | audiology | **26.27** | 28.82 | 26.73 | 27.14 |
| | breast-cancer | 3.86 | 4.34 | **3.81** | 3.82 |
| | car | **22.12** | 22.22 | 22.25 | 22.27 |
| | cmc | 54.19 | **54.16** | 54.25 | 54.24 |
| | crx | 15.12 | **14.17** | 14.45 | 14.45 |
| | dermatology | 7.42 | 5.84 | 5.49 | **5.13** |
| | ecoli | 23.49 | 23.94 | 23.49 | **23.46** |
| | glass | **4.59** | 4.73 | **4.59** | 4.73 |
| | hepatitis | **16.99** | 17.33 | 18.04 | 18.18 |
| | horse-colic | 19.74 | 19.52 | **19.43** | **19.43** |
| | house-votes-84 | 9.75 | 5.19 | 5.26 | **4.98** |
| | ionosphere | 18.28 | 7.47 | **7.22** | **7.22** |
| | iris | **6.33** | 6.73 | 6.67 | 6.67 |
| | pima | 27.33 | **26.65** | 27.02 | 27.02 |
| | promoters | 15.90 | 16.47 | **14.38** | 14.98 |
| | sonar | 36.88 | 26.06 | **25.85** | **25.85** |
| | tic-tac-toe | **0.54** | **0.54** | 0.58 | 2.00 |
| | vowel | 28.81 | 29.07 | **28.22** | **28.22** |
| | wine | 14.17 | 4.39 | 4.32 | **4.27** |
| | zoo | 3.37 | 3.99 | **3.20** | 3.87 |
| | *Average* | *17.76* | *16.08* | ***15.76*** | *15.90* |
| **1000 rounds** | audiology | 18.10 | 24.35 | **17.07** | 20.67 |
| | breast-cancer | 5.91 | 4.39 | **3.65** | 3.65 |
| | car | **10.45** | 11.37 | 11.52 | 11.84 |
| | cmc | 50.05 | **47.02** | 47.67 | 47.63 |
| | crx | 24.15 | **13.90** | 13.93 | 13.91 |
| | dermatology | 13.14 | 4.14 | 3.94 | **3.08** |
| | ecoli | 23.72 | 19.48 | **19.19** | 19.28 |
| | glass | 8.69 | 6.33 | **4.55** | 6.33 |
| | hepatitis | 18.70 | **17.16** | 17.62 | 17.84 |
| | horse-colic | 21.81 | 16.98 | **16.54** | 16.65 |
| | house-votes-84 | 29.72 | 4.38 | 5.00 | **4.20** |
| | ionosphere | 57.47 | **6.36** | 7.05 | 7.07 |
| | iris | 12.33 | 5.87 | **5.67** | 5.73 |
| | pima | 30.15 | **24.59** | 24.88 | 24.88 |
| | promoters | 27.60 | 10.88 | **6.03** | 9.07 |
| | sonar | 49.21 | **23.90** | 23.96 | 23.96 |
| | tic-tac-toe | 0.24 | 0.39 | **0.14** | 7.08 |
| | vowel | 25.95 | 14.51 | 13.44 | **13.42** |
| | wine | 47.49 | **3.98** | 4.37 | 4.03 |
| | zoo | 4.38 | **3.63** | 3.88 | 3.80 |
| | *Average* | *23.96* | *13.18* | ***12.50*** | *13.21* |

Table 5: Summary: comparison of GloBoost with C4.5, GloBo and AdaBoost boosting either *decision stumps* or *clgg*. GloBoost (*frequency*) and AdaBoost are run for 1000 rounds.

| | | | **Adaboost** | | |
|---|---|---|---|---|---|
| **Dataset** | **C4.5** | **GloBo** | **stumps** | ***clgg*** | **GloBoost** |
| audiology | 18.20 | 20.76 | **15.79** | 19.22 | 17.07 |
| breast-cancer | 4.87 | 3.89 | 4.16 | **3.10** | 3.65 |
| car | **7.69** | 10.17 | 13.36 | 9.74 | 11.52 |
| cmc | 48.07 | 50.60 | **44.87** | 49.61 | 47.67 |
| crx | 14.79 | 16.50 | 16.53 | 14.47 | **13.93** |
| dermatology | 6.23 | 8.51 | 4.38 | 4.63 | **3.94** |
| ecoli | 15.89 | 23.88 | **15.59** | 19.25 | 19.19 |
| glass | 28.72 | 5.57 | 22.53 | 4.59 | **4.55** |
| hepatitis | 20.70 | 20.09 | 18.23 | 18.39 | **17.62** |
| horse-colic | **13.63** | 22.29 | 21.71 | 18.42 | 16.54 |
| house-votes-84 | **3.22** | 7.39 | 5.31 | 6.12 | 5.00 |
| ionosphere | 7.96 | 8.74 | 12.19 | 7.44 | **7.05** |
| iris | **5.33** | 7.47 | **5.33** | **5.33** | 5.67 |
| pima | 29.29 | 27.04 | 25.39 | 25.13 | **24.88** |
| promoters | 18.17 | 22.60 | 18.67 | 7.00 | **6.03** |
| sonar | 28.97 | 31.09 | **15.03** | 23.34 | 23.96 |
| tic-tac-toe | 14.40 | 1.11 | 1.67 | **0.00** | 0.14 |
| vowel | 21.21 | 23.99 | 14.65 | **9.30** | 13.44 |
| wine | 8.83 | 8.94 | 18.07 | 5.73 | **4.37** |
| zoo | 7.51 | 3.89 | 3.55 | **3.32** | 3.88 |
| *Average* | *16.18* | *16.23* | *14.85* | *12.71* | ***12.50*** |

on some problems.

A last perspective would consist in theoretically analyzing the performances of GloBoost. According to us, the best framework for this purpose should be the Monte Carlo theory that has been already used to explain some aspects of bagging and boosting [ES03].

# References

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html], 1998.

[Bre96a] L. Breiman. Bias, variance, and arcing classifiers, 1996. Technical Report 460, Statistics Department, University of California.

[Bre96b] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[ES03] Roberto Esposito and Lorenza Saitta. Monte Carlo Theory as an Explanation of Bagging and Boosting. In Georg Gottlob and Toby Walsh, editors, *Proceeding of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 499–504. Morgan Kaufman, 2003.

[Fre95] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[FS98] Yoav Freund and Robert E. Schapire. Discussion of the paper Arcing Classifiers by Leo Breiman. *The Annals of Statistics*, 26:824–832, 1998.

[GS98] Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, pages 692–699, 1998.

[Har99] Michael Harries. Boosting a strong learner: evidence against the minimum margin. In *Proc. 16th International Conf. on Machine Learning*, pages 171–180. Morgan Kaufmann, 1999.

[RW02] Gunnar Rätsch and Manfred Warmuth. Maximizing the margin with boosting. In Jyrki Kivinen and Robert H. Sloan, editors, *Proceedings of the Annual Conference on Computational Learning Theory (COLT)*, volume 2375 of *LNCS*, pages 334–350. Springer, 2002.

[RW03] Gunnar Rätsch and Manfred K. Warmuth. Efficient margin maximizing with boosting, 2003. submitted to Journal of Machine Learning Research (JMLR).

[Sch90] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[SFBL97] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning (ICML)*, pages 322–330. Morgan Kaufmann, 1997.

[SKB99] A. Srinivasan, R.D. King, and D.W. Bristol. An assessment of submissions made to the predictive toxicology evaluation challenge. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 270–276. Morgan Kaufmann, 1999.

[SS99] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[SS00] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[Tor99] F. Torre. GloBo : un algorithme stochastique pour l'apprentissage supervisé et non-supervisé. In M. Sebag, editor, *Actes de la Première Conférence d'Apprentissage*, pages 161–168, 1999.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, pages 1134–1142, 1984.