Journal of Machine Learning Research 3 (2002) 863-885 Submitted 12/01; Published 12/02

Stopping Criterion for Boosting-Based Data Reduction Techniques: from Binary to Multiclass Problems

Marc Sebban MARC.SEBBAN@UNIV-ST-ETIENNE.FR
Eurise, Faculty of Sciences
University of Jean Monnet
42023 Saint-Etienne Cedex 2, France

Richard Nock RNOCK@MARTINIQUE.UNIV-AG.FR
GRIMAAG, Scientific Department
French West Indies and Guiana University
97275 Schoelcher Cedex, Martinique, France

Stéphane Lallich LALLICH@UNIV-LYON2.FR
ERIC, Department of Economics
University of Lyon 2, 69676 Bron Cedex, France

Editors: Carla E. Brodley and Andrea Danyluk

Abstract

So far, boosting has been used to improve the quality of moderately accurate learning algorithms, by weighting and combining many of their *weak* hypotheses into a final classifier with theoretically high accuracy. In a recent work (Sebban, Nock and Lallich, 2001), we have attempted to adapt boosting properties to data reduction techniques. In this particular context, the objective was not only to improve the success rate, but also to reduce the time and space complexities due to the storage requirements of some costly learning algorithms, such as nearest-neighbor classifiers. In that framework, each *weak* hypothesis, which is usually built and weighted from the learning set, is replaced by a single learning instance. The weight given by boosting defines in that case the relevance of the instance, and a statistical test allows one to decide whether it can be discarded without damaging further classification tasks. In Sebban, Nock and Lallich (2001), we addressed problems with two classes. It is the aim of the present paper to relax the class constraint, and extend our contribution to multiclass problems. Beyond data reduction, experimental results are also provided on twenty-three datasets, showing the benefits that our boosting-derived weighting rule brings to weighted nearest neighbor classifiers.

1. Introduction

Some of the earliest approaches to classification are also among the simplest: they do not induce concept representations (decision trees, neural networks, etc.), but exploit simple structures of the learning set, such as neighborhoods, to classify instances. Among them, the most popular is probably the 1-Nearest-Neighbor (NN) algorithm (Cover and Hart, 1967), and its generalization, the k-NN rule, which classifies an unknown instance according to a local vote by its k-nearest neighbors. Its use was widely spread and encouraged by early theoretical results linking its generalization error to Bayes risk. Under mild regularity assumptions on the underlying statistics, for any metric, the large-sample risk incurred is less than twice the Bayes risk. Even more, the risk paid off for finite samples can be very reasonable under similar assumptions (Nock and Sebban, 2001b). However, from a practical point of view, this algorithm has several problems, as mentioned in Breiman et al. (1984): (i) it is computationally expensive because it stores all the instances in memory; (ii) it is intolerant to noisy instances; (iii) it is intolerant to irrelevant attributes and (iv) it is sensitive to the chosen distance function.

The deletion of noisy instances and irrelevant attributes is addressed by data reduction techniques. Recent complexity theoretic results show that some related optimization problems are very hard to approximate (Nock and Sebban, 2000). This advocates for the use of heuristics for data reduction. In this paper, we only focus on prototype selection, which consists of identifying and eliminating irrelevant instances. Prototype selection concerns both storage complexity (first problem listed above) and noise tolerance (second problem). The last two problems are not discussed in this paper. Many solutions have been proposed to select relevant features (John, Kohavi and Pfleger, 1994; Koller and Sahami, 1996; Sebban, 1999) and to define new distance functions (Wilson and Martinez, 1997).

Many prototype selection methods have been suggested to improve the standard NN algorithm using different strategies: removing correctly classified examples (Hart, 1968; Gates, 1972), identifying and eliminating mislabeled instances (Brodley and Friedl, 1996), deleting misclassified or irrelevant instances (Wilson and Martinez, 2000; Sebban and Nock, 2000), identifying relevant prototypes by Monte-Carlo sampling (Skalak, 1994), etc. Recently, we proposed an adaptation of boosting to prototype selection (Nock and Sebban, 2001a) in the PSBOOST algorithm. Boosting, as used in the well known ADABOOST algorithm (Freund and Schapire, 1997), generates a final combined classifier whose error on the learning set is small by weighting and combining T *weak* hypotheses, each of which may have a large error. Here, T is the number of boosting rounds, a parameter fixed in advance. Freund and Schapire (1996) proposed reducing the number of instances used by each weak hypothesis to speed up the NN classifier. As far as we know, this work was the first attempt to use boosting in prototype selection, although their goal was not to improve the accuracy. The objective of PSBOOST is to obtain a good balance between storage requirements and generalization accuracy. Its principle is to use each instance as a weak hypothesis: the confidence weight given by boosting becomes in our case an indication of the instance's relevance. Experimental results indicate the efficiency of this approach (Nock and Sebban, 2001a). Inspired by boosting, PSBOOST suffers from the same important drawback: the control of the number of boosting rounds, that is, the size N_p of the final prototype set in our framework. Nock and Sebban (2001a) studied the balance between a small value of N_p which allows high storage reduction but decreases the accuracy, and a large value which allows us to control the generalization accuracy but still needs high storage requirements. The results obtained reveal the crucial need for a method fixing as accurately as possible this parameter (Nock and Sebban, 2001a). A first attempt to cope with this problem is provided by Sebban, Nock and Lallich (2001), but it holds only for problems with two classes.

In this paper, we relax the class constraint, thereby extending our framework to multiclass problems. We draw up a statistical test based on the normalization factor Z, the criterion minimized in ADABOOST, and optimized in PSBOOST as well. Experimental results display the ability of this criterion to obtain a significant size reduction, together, on average, with an increase of the accuracy. This generalized version of PSBOOST, called PSBOOST2_MC, also displays experimentally its ability to address the first two problems (storage requirement and noise tolerance) of k-NN classifiers.

A significant drawback of k-NN classifiers is that they require fixing k in advance. This is clearly not an easy task in real-world domains. While a small value of k is often sufficient for noise free problems, the k-NN rule requires thorough investigations for complex problems, often leading to the testing of many values of k. To cope with this problem, in this paper, we extend our algorithm to another kind of neighborhood-based classifier, whose geometry does not rely on *ad hoc* parameters. The underlying neighborhood graph is called the Relative Neighborhood Graph (RNG) (Toussaint, 1980). Experimental results again display the ability of our algorithm to improve classifiers based on the RNG, even in the presence of noise.

The final contribution of this paper is not restricted to data reduction. In Sebban, Nock and Lallich (2001), it is argued that the instance's weighting rule derived from boosting deserves investigations for its use in weighted nearest neighbors classifiers. We provide in this paper experimental results on a body of twenty-three datasets. They display significant improvements obtained when using boosting-derived weights.

In the rest of this paper, after having briefly recalled the main properties of boosting and PSBOOST in Section 2, we describe in Section 3 our statistical criterion for automatically halting the selection procedure, and the new version of our algorithm, called PSBOOST2. In Section 4, we describe the RNG, before presenting a large experimental study (Section 5). We make some observations in Section 6, and we explain why PSBOOST2 is suited for reducing storage while controlling the classifier accuracy. In Section 7, we present the extension of the test to multiclass problems. The use of the instance weights in weighted classifiers is discussed in Section 8, before our final conclusion.

2. Adapting Boosting to Data Reduction

In this section, we recall the main properties of boosting and PSBOOST.

2.1 Properties of Boosting

Boosting resides in combining many (T) *weak* hypotheses produced from various distributions $D_t(e)$ over the learning set (*LS*). The pseudocode of the original boosting algorithm, called ADABOOST (Freund and Schapire, 1997) is described in Figure 1. At each stage t, ADABOOST decreases (resp. increases) the weight of learning instances, *a priori* labeled $y(e)$, which are correctly (resp. incorrectly) classified by the current weak hypothesis h_t. Boosting thus forces the weak learner to learn the hardest examples. The weighted combination $H(e)$ of all the weak hypotheses results in a better performing model. Schapire and Singer (1998) proved that, in order to minimize learning error, one must seek to minimize Z_t in each round of boosting, requiring the use of a specific confidence α_t.

In order to present our adaptation of boosting to storage reduction with neighborhood-based classifiers, we first introduce several notations proposed by Schapire and Singer (1998). Suppose that $y(e) \in \{-1;1\}$ and that the output of each weak hypothesis h_t is restricted to $-1,0,+1$. Let W^{-1}, W^0 and W^{+1} be defined by

$$W^b = \sum_{e \in LS: y(e)h_t(e)=b} D_t(e) \ .$$

```
ADABOOST (LS, W, T)
   Initialize distribution D_1(e) = 1/|LS|
   for any e ∈ LS;
   For t = 1, 2, ..., T
      Train weak learner W on LS using D_t
      and get a weak hypothesis h_t;
      Compute the confidence α_t = 1/2 log((1-ε_t)/ε_t);
      Where ε_t = Σ_{y(e)≠h_t(e)} D_t(e) is the error
      of h_t.
      Update:
          ∀e ∈ LS:   D_{t+1}(e) = D_t(e) exp(-α_t y(e) h_t(e)) / Z_t;
      /*Z_t is a Normalization Factor*/
   endFor
   Return the classifier
```

$$H(e) = sign(\sum_{t=1}^{T} \alpha_t h_t(e))$$

Figure 1: Pseudocode for ADABOOST.

Using symbols + and - for +1 and -1, we can calculate the normalization factor Z as:

$$\begin{aligned} Z_t &= \sum_{e \in LS} D_t(e) \exp(-\alpha_t y(e) h_t(e)) \\ &= \sum_{b} \sum_{e \in LS: y(e)h_t(e)=b} D_t(e) \exp(-\alpha_t b) \\ &= W^0 + W^- \exp(\alpha_t) + W^+ \exp(-\alpha_t) \ . \end{aligned}$$

Z_t is then minimized when

$$\alpha_t = \frac{1}{2} \log \left(\frac{W^+}{W^-} \right) \ . \tag{1}$$

Freund and Schapire's original ADABOOST algorithm would instead have made the more conservative choice

$$\alpha_t = \frac{1}{2} \log \left(\frac{W^+ + \frac{1}{2}W^0}{W^- + \frac{1}{2}W^0} \right) ,$$

giving a normalization coefficient Z which Freund and Schapire (1997) upper bound by

$$Z_t \leq 2\sqrt{(W^+ + \frac{1}{2}W^0)(W^- + \frac{1}{2}W^0)} \ .$$

2.2 PSBOOST

Suppose now that each weak hypothesis h_t is not a classifier produced from the whole learning set (LS), but rather a given example e. In ADABOOST, the confidence α_t is a function of the prediction error of h_t on LS. Replacing h_t by e requires a more sophisticated error measure that we can call the pseudo-loss, as used in Freund and Schapire (1996). While the loss of a classifier h_t is based on its ability to correctly classify *all the instances*, the pseudo-loss of e must take into account its influence only *on its neighborhood* in LS.

Definition 1 *Let $N(e)$ be the neighborhood of an instance e of the learning set LS:* $N(e) = \{e' \in LS : e'$ *is one of the k-nearest neighbors of e in the oriented k-NN graph*$\}$.

Note that the above definition can be extended to other neighborhood graphs.

Definition 2 *Let $R(e)$ be the reciprocal neighborhood of an instance e of the learning set LS:* $R(e) = \{e' \in LS : e \in N(e')\}$.

Stated differently, $R(e)$ represents the set of instances which have e in their neighborhood. Whenever the neighborhood relationship can be represented by a directed graph, such as for the k-NN rule, we generally have $R(e) \neq N(e)$. If we consider e as a weak hypothesis, its output takes three possible values in the case with two classes:

- $y(e) \in \{-1; 1\}$ for any instance in $R(e)$,
- 0 for any instance not in $R(e)$.

Let W_e^+ (resp. W_e^-) be the fraction of instances in $R(e)$ having the same class as e (resp. a different class from e), and let W_e^0 be the fraction of instances to which e gives a null vote (those not in $R(e)$). Then, the example e we choose at each round t of boosting should be the one minimizing the following coefficient:

$$Z_e = 2\sqrt{\left(W_e^+ + \frac{1}{2}W_e^0\right)\left(W_e^- + \frac{1}{2}W_e^0\right)}\,, \tag{2}$$

and the confidence α_e can be calculated as

$$\alpha_e = \frac{1}{2}\log\left(\frac{W_e^+ + \frac{1}{2}W_e^0}{W_e^- + \frac{1}{2}W_e^0}\right)\,. \tag{3}$$

Note that we use here the less optimal quantities given by Freund and Schapire (1997) and not those proposed by Schapire and Singer (1998). Our choice basically increases the influence of W_e^0, since parameter W^0 is absent from the weighting coefficient in Equation 1. This choice is motivated by the fact that in our case, many instances do not belong to the reciprocal neighborhood $R(e)$ of some instance e, resulting in a value for W_e^0 eventually much higher than in the weak hypotheses that abstain Schapire and Singer's (1998) model. In our approach, a small W_e^0 (of course combined with a high W_e^+) indicates a high local influence of e, and then is considered an interesting candidate for the selection. Note that once a prototype is selected, it will still be considered as in other reciprocal neighborhoods, but of course not as a candidate.

The pseudocode of our algorithm PSBOOST is described in Figure 2. Note that, in this section, the confidence α_e is only used for selecting the prototypes and not for generating a weighted classifier, which is the subject of the last section of this paper. This choice is motivated by the fact that our

PSBOOST (LS,N_p)
  Initialize distribution $D_1(e) = 1/|LS|$
  for any $e \in LS$;
  Initialize candidates set $LS_* = LS$;
  Initialize $LS' = \emptyset$
  For $t = 1,2,...,N_p$
    $e = \arg\min_{e' \in LS_*} Z_{e'}$;
    If $\alpha_e < 0$ Then EndLoop;
    $LS' = LS' \cup e$
    $LS_* = LS_* - \{e\}$
    Update:
      $\forall e' \in R(e)$:
      $D_{t+1}(e') = \frac{D_t(e')\exp(-\alpha_e y(e')y(e))}{Z_e}$;
      $\forall e' \in LS \backslash R(e)$: $D_{t+1}(e') = \frac{D_t(e')}{Z_e}$;
    /$*Z_e$ is a normalization coefficient$*$/
  endFor
  Return LS'

Figure 2: Pseudocode for PSBOOST. The output of this algorithm is the prototype subset LS'.

	k-NN	CF		PSRCG		PSBOOST	MC	RT3		PSBOOST*
DataSets	*Acc.*	*Acc.*	*% prot*	*Acc.*	*% prot*	*Acc.*	*Acc.*	*Acc.*	*% prot*	*Acc.*
AUDIOLOGY	73.40	73.80	69.8	73.70	86.1	73.84	72.21	69.40	10.7	70.00
AUSTRAL	80.55	79.55	84.2	78.41	58.5	80.27	75.41	70.67	10.9	71.68
BIGPOLE	59.94	60.23	69.8	59.92	88.1	58.91	59.94	58.89	17.5	57.29
BREAST	96.89	96.76	97.5	97.04	10.0	96.19	97.04	87.72	1.2	81.24
BRIGHTON	95.80	94.59	97.6	94.6	32.4	94.46	93.60	90.56	16.4	90.27
BUPA	62.67	65.31	73.0	66.76	79.0	68.77	65.90	63.35	10.8	62.45
ECHOCARDIO	60.00	69.18	68.0	62.58	60.3	61.87	62.63	58.30	5.8	61.87
GERMAN	72.85	72.05	77.8	70.87	73.7	72.65	69.88	69.87	10.6	72.65
GLASS2	72.65	73.31	81.5	72.10	72.7	74.49	70.88	55.07	11.7	60.29
HARD	47.12	45.36	63.4	44.91	90.9	47.85	46.32	48.27	13.4	50.55
HEART	74.21	73.15	81.0	74.56	52.1	79.92	73.13	74.91	6.9	72.72
HEPATITIS	81.73	82.38	88.6	81.20	46.6	76.63	77.95	68.25	9.6	76.56
HORSE	68.98	68.43	79.6	69.48	74.1	72.95	71.87	70.30	9.5	67.89
IONOSPHERE	75.75	74.12	86.4	71.46	65.0	76.73	74.61	74.06	8.4	76.20
LED+17	72.76	75.64	82.3	69.93	81.2	76.12	68.95	64.62	19.8	70.43
LED	89.79	89.20	91.3	87.63	36.9	89.20	86.83	66.79	4.4	74.68
PIMA	67.44	67.82	76.1	66.79	68.2	67.18	64.87	62.65	6.7	65.87
VEHICLE	70.99	70.30	78.8	70.17	69.2	69.95	68.70	58.82	7.8	68.51
WHITEHOUSE	90.76	89.91	93.7	91.47	30.0	90.57	92.60	81.52	4.5	90.11
XD6	80.60	80.46	85.5	80.29	72.0	80.63	80.00	72.26	14.6	74.70
AVERAGE	**74.75**	**75.08**	**81.3**	**74.19**	**62.4**	**75.46**	**73.70**	**68.31**	**8.7**	**70.80**

Table 1: Results (accuracy *Acc.* and percentage of selected instances *% prot*) for *k*NN ($k = 5$), CF, PSRCG, PSboost, MC (Monte-Carlo), RT3, PSboost*; PSboost (resp. PSboost*) means that PSboost is run with exactly the same number of prototypes than PSRCG (resp. RT3)

original goal is to select the most relevant instances from LS. Once the selection is done, the output LS' can be then used as a standard learning set. In order to assess the efficiency of our selection method, we compare the performances of LS and LS' without any other optimization strategy (for instance by generating a weighted classifier).

Some useful observations can be made about the value of Z_e and its contribution to removing irrelevant instances in LS. First, if an instance e belongs to a region with very few instances, it will not belong to many reciprocal neighborhoods, resulting in a large W_e^0, preventing the achievement of small Z_e. Secondly, if a prototype belongs to a region with evenly distributed instances, W_e^+ and W_e^- tend to be balanced, and this again, prevents to obtain small Z_e. Note that with our strategy, a cluster of instances of the same class could be all picked for LS', resulting in a redundancy in the final subset. A way to solve this drawback would consist in applying a post-process to remove redundancy. For example, Sebban and Nock (2000) proposed, in another context, to compute an information measure from a $(k+1)$-NN graph. Only instances at the center of clusters keep a null uncertainty with $k+1$ neighbors. Removing such instances allows the deletion of the useless instances from the clusters.

Note in Figure 2 that the user must provide a value for N_p, the number of prototypes. In this paper, we provide a theoretical framework for automatically determining N_p using a statistical test. Nock and Sebban (2001a) carried out a large comparative study between PSBOOST and the state-of-the-art prototype selection algorithms for which we recall the main results (obtained by cross-validation) in Table 1. CF corresponds to the Consensus Filter (Brodley and Friedl, 1996), PSRCG was proposed by Sebban and Nock (2000), RT3 by Wilson and Martinez (2000), and MC corresponds to Monte-Carlo sampling as proposed by Skalak (1994) (for more details see Nock and Sebban (2001a)). Although these results are interesting, the parameter N_p must be fixed in advance, and that constitutes a drawback for PSBOOST in its original version.

3. Theoretical Stopping Criterion

In this section, we describe our statistical criterion for automatically halting the selection procedure.

3.1 A Random Framework for Test Construction

In this section, we propose a theoretical framework for determining the number of weak hypotheses N_p. Our strategy is based on a statistical test. Let H_0 be the null hypothesis of this test, which expresses the idea that a given e does not statistically contribute to give information about the labelling of its reciprocal neighborhood. Informally, as long as H_0 can be kept, such an instance can be removed *without* reasonably endangering further classification tasks. This requires a statistic that assesses for a given candidate e the validity of H_0, and for which we provide the statistical law under H_0. For a given risk θ, we stop the selection if and only if all the candidates have a *p-value* higher than θ. Stated differently, the algorithm stops if the best current candidate does not allow the rejection of H_0 with a risk smaller than θ. We provide here a theoretical framework for binary problems. The extension to multiclass problems is discussed in Section 7.

A possible way of proceeding consists in considering under H_0 that, in the reciprocal neighborhood $R(e)$, the true class Y is randomly distributed with a given probability π_0 (if $y(e)=1$) or $1-\pi_0$ (if $y(e)=-1$). Two ways are possible to fix π_0:

1. Choose π_0 equal to the global proportion in LS of *positive* learning instances (those for which $y(e)=1$) .

2. Use $\pi_0 = 0.5$ to satisfy a majority vote rule for a 2-class problem, often used in classification tasks. Stated differently, we test if e classifies instances in $R(e)$ better than a simple coin toss.

Let $H_0(\pi_0)$ be the corresponding null hypothesis. Under $H_0(\pi_0)$, an instance of the reciprocal neighborhood $R(e)$ belongs to the same class as e with probability π_0 (resp. $1-\pi_0$) if $y(e) = 1$ (resp. $y(e) = -1$).

3.2 Law of W_e^+ under H_0

In our approach, an instance e is selected by minimizing the quantity Z_e, while ensuring a positive confidence α_e (which avoids the selection of mislabeled instances).

$$\begin{aligned} Z_e &= 2\sqrt{(W_e^+ + \frac{1}{2}W_e^0)(W_e^- + \frac{1}{2}W_e^0)} \\ &= 2\sqrt{(W_e^+ + \frac{1}{2}W_e^0)(1 - W_e^+ - \frac{1}{2}W_e^0)} \ , \end{aligned}$$

because $W_e^+ + W_e^- + W_e^0 = 1$. Then, Z_e depends on the value of W_e^+ in $R(e)$:

$$\begin{aligned} W_e^+ &= \sum_{e' \in R(e): y(e') = y(e)} D_t(e') \\ &= \sum_{e' \in R(e)} D_t(e') I_{\{y(e') = y(e)\}} \ , \end{aligned}$$

where the boolean variable $I_{\{y(e')=y(e)\}}$ is 1 iff $y(e') = y(e)$, and 0 otherwise. If $H_0(\pi_0)$ is true, $I_{\{y(e')=y(e)\}}$ follows a binomial law $B(1, p)$, where $p = \pi_0$ if $Y(e) = 1$ else $p = 1 - \pi_0$. Considering that W_e^+ depends on examples $i, i = 1, 2, .., |R(e)|$ (the size of the reciprocal neighborhood), we propose the following simplification:

$$W_e^+ = \sum_{i=1}^{|R(e)|} D_t(i) I_i \ .$$

There are two different ways to construct the distribution of W_e^+ under H_0 to compute the critical value of W_e^+, called $W_{1-\theta}^+$. We recall here that the critical value defines the bound of the rejection region of H_0, and corresponds to the $(1-\theta)$-percentile of the distribution of W_e^+ under H_0. In the two following approaches, we assume that the $D_t(i)$ are not random variables, even if in theory, they depend on the labels of the examples. First, the distribution can be assessed by a *normal approximation*. In this case, under $H_0(\pi_0)$, W_e^+ is a weighted sum of $|R(e)|$ variables I_i, where the I_i are independently and identically distributed. The mean and variance of W_e^+ are:

$$\begin{aligned} E(W_e^+/H_0) &= \sum_{i=1}^{|R(e)|} D_t(i) E(I_i) \\ &= p \sum_{i=1}^{|R(e)|} D_t(i) \\ Var(W_e^+/H_0) &= \sum_{i=1}^{|R(e)|} D_t^2(i) Var(I_i) \\ &= p(1-p) \sum_{i=1}^{|R(e)|} D_t^2(i) \ . \end{aligned}$$

The other way to proceed would consist of *simulating the distribution* of W_e^+, which can deal with cases where the approximation constraints are not satisfied. For balanced weights (when W_e^+ and W_e^- are close), $|R(e)| > 10$ is enough to satisfy these constraints. In an unbalanced case, W_e^+ must be larger.

3.3 Statistical Test

Without a criterion for halting the selection, PSBOOST requires the provision of the number N_p of weak hypotheses. Such a strategy may lead to the selection of an instance for which the null hypothesis H_0 would not be rejected. By introducing a statistical test using the critical value $W^+_{1-\theta}$, we keep only instances e for which W_e^+ is exceptionally high under H_0 (*i.e.*, $W_e^+ > W^+_{1-\theta}$). Among these, we choose at a given stage of the selection the one that minimizes Z, or equivalently Z^2. The procedure is stopped if for all the instances e, $W_e^+ < W^+_{1-\theta}$.

3.3.1 ASSESSING THE CRITICAL VALUE OF W_e^+

We assess $W^+_{1-\theta}$ either by normal approximation or by simulation, which is computationally expensive, but sometimes necessary if the approximation conditions are not satisfied. By approximation, $W^+_{1-\theta}$ is easily defined as follows:

$$W^+_{1-\theta} = p\sum_{i=1}^{|R(e)|} D_t(i) + u_{1-\theta}\sqrt{p(1-p)\sum_{i=1}^{|R(e)|} D_t^2(i)} \ ,$$

where $u_{1-\theta}$ is the $(1-\theta)$-percentile of the normal law $N(0,1)$. If the approximation constraints are not satisfied, we can artificially construct a distribution of W_e^+, by simulating $|R(e)|$ independent observations I_i according to $B(1,p)$, and computing the weighted sum $\sum_{i=1}^{|R(e)|} D_t(i)I_i$. By repeating this procedure N times, an estimate of $W^+_{1-\theta}$ is the $(1-\theta)$-percentile of the N samples.

3.3.2 DECISION RULE

An instance e is selected by minimizing Z_e:

$$Z_e = 2\sqrt{(W_e^+ + \frac{1}{2}W_e^0)(W_e^- + \frac{1}{2}W_e^0)} \ ,$$

while ensuring a positive confidence α_e:

$$\alpha_e = \frac{1}{2}\log\left(\frac{W_e^+ + \frac{1}{2}W_e^0}{W_e^- + \frac{1}{2}W_e^0}\right) \ .$$

At each stage of the selection, our procedure minimizes the quantity $Z^2 = 4F(1-F)$, where $F = W_e^+ + \frac{1}{2}W_e^0$. The critical value of F with the risk θ is directly deduced from $W^+_{1-\theta}$:

$$\begin{aligned} F_{1-\theta} &= W^+_{1-\theta} + \frac{1}{2}W_e^0 \\ &= p\sum_{i=1}^{|R(e)|} D_t(i) + \frac{1}{2}W_e^0 + u_{1-\theta}\sqrt{p(1-p)\sum_{i=1}^{|R(e)|} D_t^2(i)} \ . \end{aligned}$$

Under $H_0(\pi_0)$, $F_{1-\theta}$ can in theory be smaller than 0.5 when $p < 0.5$. In this case, if two candidates satisfy the first condition ($W_e^+ > W_{1-\theta}^+$), their confidences are then negative, and paradoxically we will choose the candidate e which presents the smaller value F_e ($F_e \in [F_{1-\theta}, 0.5]$), by minimizing Z_e. This situation, possible when p is very close to 0, in fact rarely occurs because there is almost always a candidate e' for which $F_{1-\theta} > 0.5$ and $F_{e'} > F_{1-\theta}$ ($F_{e'} \in [F_{1-\theta}, 1]$), often resulting in $Z_{e'} < Z_e$. This fact has been confirmed by an experimental study. Actually, on 18 datasets, using a 5-fold cross-validation resulting in 90 different databases, we noted that this situation never occurred. However, the neighborhood-based classifiers, such as the k-nearest-neighbors, usually use a majority decision rule with a threshold 0.5 (in the case of 2 classes). In such a context, it is more suitable to test the null hypothesis $H_0(0.5)$, which means that we select only the instance e that classifies, in the reciprocal neighborhood $R(e)$, significantly better than a simple toss. In this case, $F > 1 - F$, and then $\alpha > 0$. Under $H_0(0.5)$, we have always $p = 0.5$, and the previous formulae for $F_{1-\theta}$ can be simplified:

$$
\begin{aligned}
F_{1-\theta} &= \frac{1}{2}\left(\sum_{i=1}^{|R(e)|} D_t(i) + W_e^0\right) + \frac{1}{2}u_{1-\theta}\sqrt{\sum_{i=1}^{|R(e)|} D_t^2(i)} \\
&= \frac{1}{2} + \frac{1}{2}u_{1-\theta}\sqrt{\sum_{i=1}^{|R(e)|} D_t^2(i)}\ .
\end{aligned}
$$

We deduce the critical values of Z^2 and α with the risk θ, called c_θ and $\alpha_{1-\theta}$:

$$
\begin{aligned}
c_\theta &= (2\sqrt{F_{1-\theta}(1-F_{1-\theta})})^2 \\
&= 1 - u_{1-\theta}^2 \sum_{i=1}^{|R(e)|} D_t^2(i) \\
\alpha_{1-\theta} &= \frac{1}{2}\log\frac{F_{1-\theta}}{1-F_{1-\theta}} \\
&= \frac{1}{2}\log\frac{1 + u_{1-\theta}\sqrt{\sum_{i=1}^{|R(e)|} D_t^2(i)}}{1 - u_{1-\theta}\sqrt{\sum_{i=1}^{|R(e)|} D_t^2(i)}}\ .
\end{aligned}
$$

Then, we select the instance e if and only if $Z_e^2 < c_\theta$ or $\alpha_e > \alpha_{1-\theta}$. Note that, while we select the instance e for which Z_e is minimum, we use in the decision rule the law of Z_e and not the one of $\min_e Z_e$. According to the level of dependence of $Z_e's$, the risk is in fact contained between θ and $(\theta.|LS|)$. A simulation procedure would allow us to have more information about this problem. Then, note that θ is more a control parameter than the probability of type 1 error. The new version of our algorithm, called PSBOOST2, is described in Figure 3.

4. The Relative Neighborhood Graph

While PSBOOST2 was originally proposed for improving the k-NN algorithm, our theoretical framework is independent of the geometrical structure used for the construction of the reciprocal neighborhood $R(e)$. So, let us consider another neighborhood graph, called the Relative Neighborhood

```
PSBOOST2 (LS)
  Initialize D_1(e) = 1/|LS| for any e ∈ LS;
  Initialize candidates set LS_* = LS;
  Initialize LS' = ∅
  Repeat
    Temp = {e' ∈ LS_* : W_e^+ > W_{1-θ}^+}
    e = arg min_{e'∈Temp} Z_{e'};
    If α_e > α_{1-θ} Then
      Stop ← False
      LS' = LS' ∪ e
      LS_* = LS_* − {e}
      Update:
      ∀e' ∈ R(e):
      D_{t+1}(e') = D_t(e') e^{−α_e y(e') y(e)} / Z_e;
      ∀e' ∈ LS\R(e):   D_{t+1}(e') = D_t(e') / Z_e;
    Else Stop ← True
    endIf
  Until Stop=True
  Return LS'
```

Figure 3: Pseudocode for PSBOOST2.

Graph (RNG). Introduced by Toussaint (1980), the RNG is a connected graph in which, if two instances are linked by an edge, then they satisfy the following property:

$$d(a,b) \quad \leq \quad \min_{c \in LS, c \neq a,b} \max(d(a,c), d(b,c)).$$

This definition means that $L_{a,b}$, which corresponds to the intersection of two hyperspheres, with centers a and b and with radius equal to the distance between a and b, does not contain any other point of the learning set LS (Figure 4 describes an example). The RNG can naturally be used in a neighborhood-based classifier. We present here a general framework for problems with an arbitrary number of classes and an arbitrary geometrical structure used for building the neighborhood graph.

Definition 3 *Let C_i be the set of learning instances belonging to the i-th class: $\forall i = 1,..,c$, $C_i = \{e \in LS : y(e) = i\}$ where c is the number of classes.*

Definition 4 *Let $O(e')$ be the c-dimension vector whose components are noted $O_i(e')$, $i = 1,..,c$, each being the proportion of instances in the neighborhood of e' belonging to the i-th class:*

$$O_i(e') \quad = \quad \frac{|N(e') \cap C_i|}{|N(e')|}, \forall i = 1,2,..,c \ ,$$

where $N(e')$ is the set of neighbors of e' (linked by an edge to e') in the neighborhood graph.

Note that definition 4 also applies to new instances, not belonging to the learning set.

Definition 5 *Let $\phi(e')$ be the class given to e' by the classifier ϕ from the neighborhood graph (RNG or k-NN):*

$$\phi(e') \quad = \quad \arg\max_i O_i(e') \ .$$

According to these definitions, the new instance e' in Figure 4 would be labeled "*black*" from its neighbors 1, 2 and 3.

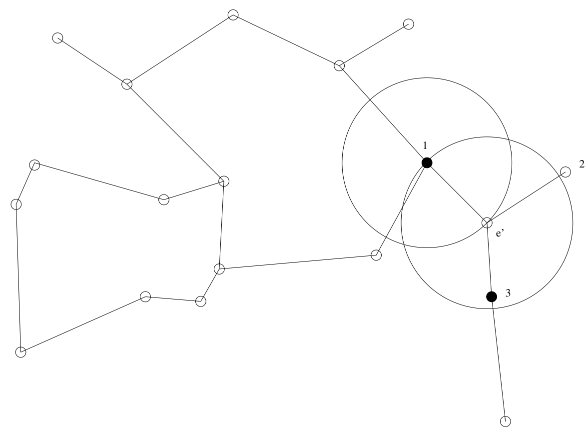

Figure 4: Relative Neighborhood Graph: the intersection of the two hyperspheres does not contain any instance of the learning set.

5. Experimental Results

In this section, we assess the efficiency of PSBOOST2 according to the two following performance measures: *generalization accuracy* and *storage reduction*. We used 18 datasets, most of which come from the UCI database repository (Merz and Murphy, 1996). The experimental method was the following: a f-fold cross-validation (here $f = 5$) was performed on each database to obtain estimates of the true performance of the classifier. We used two neighborhood-based classifiers according to the geometrical structures listed above (k-NN, here $k = 3$, and the RNG). The decision rule used for classifying an instance consists of a majority vote of the neighbors. Each database DB is divided into f disjoint sets DB_i. PSBOOST2 is applied on each combination $DB - DB_i$. The classifier uses the resulting subset of instances $(DB - DB_i)_{subset}$ for classifying the instances in DB_i. For each classifier, we obtain an accuracy estimate by averaging results over the f sets.

Note that we did not conduct a large comparative study between PSBOOST2 and the state-of-the-art prototype selection algorithms because it was already carried out for PSBOOST by Nock and Sebban (2001a), of which the main results are described in Table 1. These results have shown the difficulties that the standard prototype selection algorithms have in controlling the two performance measures. From the results described in Table 2, we can make the following remarks:

1. The learning set size is highly reduced (nearly 45% of the original size on average), while controlling the generalization accuracy. While the accuracy is slightly reduced for the Relative Neighborhood Graph by an amount that is not significant using a Student paired t-test

	kNN	PSBOOST2		RAN	RNG	PSBOOST2		RAN
Dataset	*Acc.*	*Acc.*	*% pr*	*Acc.*	*Acc.*	*Acc.*	*% pr*	*Acc.*
ECHO	59.2	63.4	37	64.9	56.3	62.7	37	61.9
HEPAT.	83.1	79.9	57	79.3	73.0	75.5	58	71.8
HEART	78.1	82.1	48	74.4	74.1	74.8	54	74.2
AUDIO	75.2	71.2	60	71.2	70.9	60.3	39	62.5
BIGPOLE	59.5	60.2	40	57.2	54.6	58.2	26	47.7
HORSE	72.3	73.4	46	71.0	64.3	67.5	38	67.8
IONO	80.4	80.4	51	78.8	72.5	73.5	36	68.2
XD6	79.8	79.1	77	77.3	79.5	71.0	57	72.0
BREAST	96.7	96.9	68	95.6	95.5	94.5	88	95.0
W.H.	91.4	92.0	68	90.9	91.1	89.5	80	88.8
GLASS2	71.9	72.0	40	64.5	67.7	66.5	34	54.5
HARD	50.0	48.3	26	45.9	54.8	64.8	13	65.7
LED24	73.5	76.5	48	69.6	74.0	68.1	43	62.8
LED2	83.9	88.1	31	88.7	88.7	85.1	41	83.5
PIMA	69.8	69.3	30	68.0	69.6	69.1	40	70.0
AUSTRAL	79.7	76.8	58	78.7	76.8	73.9	57	72.8
GERMAN	69.9	71.3	47	68.3	70.0	70.6	51	69.3
VEHICLE	70.9	70.3	40	68.1	71.9	71.7	47	71.1
AVERAGE	74.7	75.2	47	72.9	72.5	72.1	47	70.0

Table 2: Effect of PSBOOST2 on learning set size and generalization accuracy on 18 datasets; k-NN, RNG correspond respectively to the accuracy on DB_i, using the whole learning set, with a 3-NN classifier and a voting rule based on the RNG; PSBOOST2 is described by its accuracy (*Acc.*) and its storage requirement (% pr); RAN corresponds to the accuracy achieved from a learning subset of same size (LS') randomly selected in $|LS|$.

over accuracies, the predictive accuracy of the post-PSBOOST2 nearest neighbor classifier is increased (74.7% vs. 75.2%), even though this superiority is not significant with a p-value near 0.5. Therefore, it seems to confirm experimentally that PSBOOST2 is suited to control the generalization accuracy while significantly reducing the data.

2. A simple strategy for assessing the relevance of PSBOOST2 consists in comparing the selected subset (LS_1) with another one (LS_2) of the same size but randomly selected from LS. Such a procedure allows one to estimate the *quality* of the selected prototypes. We made this comparison (columns *PSBoost2/Acc.* and *Ran* in Table 2). Our strategy achieves a significantly higher accuracy than a random one, and this also tends to confirm the efficiency of PSBOOST2.

6. Some Insights into the Performances of PSBOOST2

In this section, we explain why PSBOOST2 is suited for reducing storage while controlling the classifier accuracy.

6.1 PSBOOST2 and Margin Maximization

A partial explanation of PSBOOST2's performances may rely on the margin maximization principle. This principle is in fact not recent, and was originally suggested in Vapnik (1982) for support vector machines (SVMs) with optimal margins. Even though the objective in both approaches consists in finding classifiers which maximize margins on learning data, a detailed study of their mechanisms shows that they slightly differ (Schapire et al., 1998). In SVMs the sum of squared outputs of the base hypotheses and the sum of the squared weights are both assumed to be bounded (l_2 norm), while in boosting the maximum value of the base hypotheses (l_∞ norm) and the sum of the absolute values of the weights (l_1 norm) are assumed to be bounded. Support vector machines give rise to a quadratic programming problem, whereas the optimization in boosting can be seen as a linear programming problem.

In Schapire et al. (1998), the authors prove that achieving a large margin on *LS* results in an improved bound on the generalization. They also prove that ADABOOST is suited to maximizing the number of learning examples with large margin. They define classification *margin* as the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label. The margin is then a number in the range [-1,+1] and an example is correctly classified if it has a positive margin. The margin also corresponds to a degree of confidence in the classification. In order to assess the effect of PSBOOST2 for maximizing margins, we computed for the k-NN classifier the margin gain g_i for each dataset i over the 5 folds (before and after PSBOOST2). We first observe that over the 18 datasets, the average margin gain $G = \frac{1}{18}\sum g_i = 0.24$. This might be an experimental explanation for the accuracy's control in PSBOOST2. Even more, a second observation displays the ability of PSBOOST2 to increase margins, as all datasets have a margin gain $g_i > 0$.

6.2 The Filter Precision of PSBOOST2

Brodley and Friedl (1996) provided a method for evaluating the ability of a data reduction technique to identify and eliminate mislabeled instances (called *filter precision*). This procedure in a way assesses the sensitivity to noise. Consider a learning set artificially corrupted by a given percentage of noise. One defines the 3 following sets: the set D of instances discarded, the set M of instances *a priori* corrupted, the set $M \cap D$ of corrupted instances discarded by the data reduction technique. Brodley and Friedl defined $P(E)$ as an estimate of the probability of retaining bad data:

$$P(E) = \frac{|M| - |M \cap D|}{|M|}.$$

While the original 18 datasets probably already contain noisy data, we decided to calculate $P(E)$ for different artificial noise levels. We corrupted the original data successively with 5, 10, ..., 35% noise. Table 3 reports $P(E)$ averaged over all datasets and all folds for the k-NN and the *RNG* classifiers.

In the presence of noise, the subset of instances (described by its accuracy Acc_{aft}) selected by PSBOOST2 is always better than the original learning set (Acc_{bef}). The accuracy is actually improved after prototype selection and this trend seems to speed up with the noise level. This phenomenon is not really surprising. Indeed, noise smoothes class distributions near their frontiers. These "dangerous regions" tend precisely to be discarded by PSBOOST2.

NOISE	$P(E)$ WITH kNN			$P(E)$ WITH RNG		
	Acc_{bef}	Acc_{aft}	$P(E)$	Acc_{bef}	Acc_{aft}	$P(E)$
5%	71.7	72.5	0.07	68.6	68.7	0.15
10%	67.9	69.3	0.08	65.9	66.7	0.15
15%	64.1	67.6	0.07	62.5	63.9	0.17
20%	63.5	66.0	0.08	59.1	60.6	0.16
25%	61.2	64.1	0.08	58.5	59.5	0.17
30%	58.7	61.1	0.08	56.4	58.3	0.19
35%	56.3	60.1	0.09	54.1	56.1	0.18

Table 3: PSBOOST2's filter precision

7. Extension to Multiclass Problems

In this section, we present the extension of the test to multiclass problems.

7.1 Test on $\overline{Z}_e$

So far, we have only treated binary problems. Many real-world learning problems are in fact multiclass with many more possible labels. Two main strategies have been proposed to deal with this extension to multiclass problems. The first one consists in creating one binary problem for each of the c classes. Then, we test one class j against all the other classes, answering the following question: "Does the example belong to the j^{th} class or not?" This approach is called *one-against-all* (Allwein, Schapire and Singer, 2000). The second one consists in testing all pairs of classes (Hastie and Tibshirani, 1998). For each distinct pair of classes c_1, c_2, the examples labeled c_1 are considered positive, those labeled c_2 are negative. All other examples are ignored. This approach is called *all-pairs*. An interesting comparison is presented in Allwein, Schapire and Singer (2000). In our approach, we decided to choose the first method (*one-against-all*) which requires the construction of c binary problems.

In the test proposed for solving binary problems (see Section 3.3), a candidate is selected when the corresponding $Z_e = 2\sqrt{F_e(1-F_e)}$ is minimum (where $F_e = W_e^+ + \frac{1}{2}W_e^0$), while $F_e > F_{1-\theta}$. We recall that $F_{1-\theta}$ is the critical value of F_e at the risk θ under $H_0(\pi_0)$, the hypothesis that the true class is randomly attributed with a given probability π_0, in the reciprocal neighborhood $R(e)$.

In this section, for multiclass problems, we denote by $F_{j,e}$ the value of F_e when the class j is tested against the others. We propose to select the candidate e for which the quantity $Z_e = 2\sqrt{\overline{F}_e(1-\overline{F}_e)}$ is minimum, when $\overline{F_e}$ is defined as follows:

$$\overline{F_e} = \frac{1}{c}\sum_{j=1}^{c} F_{j,e} \ .$$

The suspensive condition to select e is the following: $\overline{F}_e > \overline{F}_{1-\theta}$, where $\overline{F}_{1-\theta}$ is the critical value of $\overline{F}_e$ at the risk θ under the null hypothesis. When the class j is tested against the others, the null hypothesis, denoted by $H_0(\pi_{j0})$, means that the class j is randomly distributed with a given probability π_{j0} in $R_j(e)$, the reciprocal neighborhood of e when the class j is tested against the others. Then, note that $R_j(e)$ changes with the value j. In order to find the critical value $\overline{F}_{1-\theta}$, we have to define $\overline{\mu}$ and $\overline{\sigma}^2$ such as:

$$\begin{aligned}
\overline{\mu} &= E(\overline{F}_e) \\
&= \frac{1}{c}\sum_{j=1}^{c} E(F_{j,e}) \\
&= \frac{1}{c}\sum_{j=1}^{c}\Big(\sum_{e'\in R_j(e)} p_j D_{jt}(e') + \frac{1}{2}W^0_{j,e}\Big) \\
\overline{\sigma}^2 &= Var(\overline{F}_e) \\
&= \frac{1}{c^2}\sum_{j=1}^{c} Var(F_{j,e}) \\
&= \frac{1}{c^2}\sum_{j=1}^{c}\sum_{e'\in R_j(e)} p_j(1-p_j)D^2_{jt}(e') \ ,
\end{aligned}$$

where $D_{jt}(e')$ is the distribution at the stage t of the boosting, when the class j is tested against all the others. We note $p_j = \pi_{j0}$ if $Y(e) = j$ else $p_j = 1-\pi_{j0}$. According to the simplification proposed in Section 3.2,

$$\begin{aligned}
\overline{\mu} &= \frac{1}{c}\sum_{j=1}^{c}\Big(\sum_{i=1}^{|R_j(e)|} p_j D_{jt}(i) + \frac{1}{2}W^0_{j,e}\Big) \\
\overline{\sigma}^2 &= \frac{1}{c^2}\sum_{j=1}^{c}\sum_{i=1}^{|R_j(e)|} p_j(1-p_j)D^2_{jt}(i) \ .
\end{aligned}$$

We assume in the calculation of $\overline{\sigma}^2$ the independence of the $F_{j,e}$. Said differently, we consider that the knowledge of $R_j(e)$, from which $F_{j,e}$ is computed, does not contain information about the nature of the reciprocal neighborhood $R_l(e)$, when $j \neq l$. Actually, even if the quantity $|R_j(e)|$ remains the same $\forall j$, the labels and the weights of the neighbors in $R_j(e)$ will differ according to the tested class j. From this point of view, covariances can be considered as insignificant.

Moreover, note that variables $F_{j,e}$ are computed from independent variables, then they are not too far from a normal distribution. Furthermore, as mentioned before, they are approximately independent. Then, we can claim that $\overline{F_e}$ is very close to a normal distribution. We can determine the critical values $\overline{F}_{1-\theta}$ and c_θ for $\overline{F}_e$ and Z^2_e:

$$\begin{aligned}
\overline{F}_{1-\theta} &= \overline{\mu} + u_{1-\theta}\overline{\sigma} \\
c_\theta &= 4\overline{F}_{1-\theta}(1-\overline{F}_{1-\theta}) \ .
\end{aligned}$$

Note that for the special case where $p_j = 0.5$ (for satisfying an absolute decision rule), the previous formulae are highly simplified. Actually,

$$\begin{aligned}
\sum_{i=1}^{|R_j(e)|} p_j D_{jt}(i) + \frac{1}{2}W^0_{j,e} &= \frac{1}{2}(W^+_{j,e} + W^-_{j,e}) + \frac{1}{2}W^0_{j,e} \\
&= \frac{1}{2} \ .
\end{aligned}$$

```
PSBOOST2_MC (LS)
   Initialize D_{j1}(e) = 1/|LS| for any e ∈ LS;
   Initialize candidates set LS_* = LS;
   Initialize LS' = ∅
   Repeat
      Temp = {e' ∈ LS_* : W̄_e^+ > W̄_{1-θ}^+}
      e = arg min_{e' ∈ Temp} Z_{e'};
      If α_e > α_{1-θ} Then
         Stop ← False
         LS' = LS' ∪ e
         LS_* = LS_* − {e}
         Update:
         For j=1, 2, ..,c
            ∀e' ∈ R_j(e):
            D_{j,t+1}(e') = D_{jt}(e') e^{−α_e M(y(e'),j) M(y(e),j)} / Z_e;
            ∀e' ∈ LS\R_j(e):   D_{t+1}(e') = D_t(e') / Z_{j,e};
         EndFor
      endIf
      Else Stop ← True
   Until Stop=True
   Return LS'
```

Figure 5: Pseudocode for PSBOOST2_MC.

Then,

$$\begin{aligned}\overline{\mu} &= \frac{1}{c}\sum_{j=1}^{c}\left(\frac{1}{2}\right)\\ &= \frac{1}{2}\ .\end{aligned}$$

And,

$$\overline{\sigma}^2 = \frac{1}{4c^2}\sum_{j=1}^{c}\sum_{i=1}^{|R_j(e)|} D_{jt}^2(i)\ .$$

The pseudocode of our extended algorithm, called PSBOOST2_MC, is described in Figure 5. Note that we use in this algorithm the coding matrix $M(y(e), j)$ which was originally given by Dietterich and Bakiri (1995). For the *one-against-all* approach, M is a $c \times c$ matrix in which all diagonal elements are positive $(+1)$ and all other elements are negative (-1). When a class j is tested against the others, the current label of the instance e is the value $M(y(e), j)$, where $y(e) \in \{1, 2, .., c\}$.

7.2 Experimental Results

Table 4 presents the properties (name, number of classes, learning set size and number of features) of the eight tested datasets. In order to assess the relevance of our multiclass statistical test, we

DATASET	# CLASSES	$\|LS\|$	# FEATURES
WAVES	3	500	21
ABALONE	3	1000	8
GLASS	6	214	9
BALANCE	3	625	4
IRIS	3	150	4
LED	10	500	7
LED+17	10	500	24
DERMATOLOGY	6	366	34

Table 4: Multiclass classification problems.

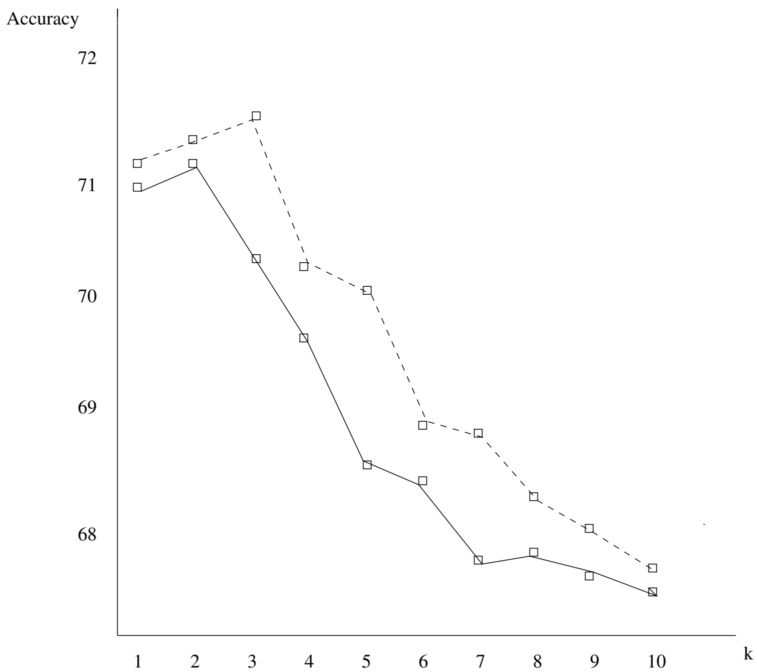

Figure 6: Contribution of PSBOOST2_MC on multiclass problems: the solid line corresponds to the accuracy of a standard k-NN classifier, built from the whole learning sample; the dashed-line represents the success rate computed from the reduced learning set.

used many values of k ($k = 1,2,..,10$) in the k-nearest neighbor classifier. Except for this detail, the experimental method remains the same as the previous study, namely the 5-fold cross-validation. A graphic synthesis of the results is presented on Figure 6. Each point of this figure is the average over the eight datasets, each of them tested five times during the cross-validation. Therefore, one point corresponds to the average of forty accuracies. Beyond data reduction, the results display the positive contribution of PSBOOST2_MC to the accuracy's increase: for all values of k, the accuracies achieved from the reduced learning set are indeed higher than without data reduction.

8. Weighted Classifiers using Instance Confidences

Beyond prototype selection, this section aims at exploring an issue that was raised by Sebban, Nock and Lallich (2001): the use of boosting-derived weights for weighted nearest neighbor rules. In such a context, the classification rule (as defined in Section 4) must be slightly modified, since the classification rule does not handle classes anymore, but real weights in favor of each class. The following definition for $O(e')$ replaces Definition 4:

Definition 6 *Let $O(e')$ be the c-dimension vector whose components are noted $O_i(e')$, $i = 1,2,..,c$, each being the sum of weights of the instances in the neighborhood of e' belonging to the i-th class:*

$$O_i(e') \quad = \sum_{e \in N(e'):y(e)=i} \alpha_e, \forall i = 1,2,..,c \ .$$

Note that α_e is still the confidence of the instance e when e is selected, but we end up selecting all instances. The weighting algorithm is a slight variant of PSBOOST2_MC, in which the condition $W_e^+ > W_e^-$ is removed. This little algorithmic difference is crucial, as some instances may now have a negative weight. This still makes sense, because the new rule leverages the neighborhood vote in favor of some classes, or in disfavor of others when negative weights abound.

Experimental studies have been conducted with a k-nearest neighbor classifier, for $k = 1,2,..,20$. We applied our approach on twenty-three datasets. Rather than presenting the twenty-three curves (one for each dataset), we synthesize the results in one figure, where each point is the average of 5 (folds) $\times$ 23 (datasets) = 115 accuracies. Results are presented in Figure 7. It appears that the performance of the standard k-NN rule is almost systematically improved by leveraging votes with the boosting weights. Even more, a Student paired t-test reveals that the difference between the standard k-NN and our weighted k-NN is significant for all values $k = 1,2,..,11$. For k large enough ($k \geq 12$), the difference becomes insignificant. This can be explained by the fact that large values of k tend to smooth neighborhood distributions (ultimately, they become the whole sample's), for which weighting brings no significant difference.

Another concise way to display the results consists in putting separately the results for each dataset, as an average over the different values of k. Instead of identifying the good values of k, we identify the good datasets, candidate for an improvement with our weighted nearest neighbor rule. We choose to take into account only the values of $k < 12$, for which weighting brings on average a statistical advantage. The results are presented in Table 5 and graphically represented in Figure 8. We can note that for 17 datasets, a weighted decision rule provides better results than the unweighted rule. Among them, 7 datasets (*Balance, Echocardiogram, German, Horse Colic, Led, Pima* and *Vehicle*) see important improvements, ranging from 1% to > 5%. In contrast, only one dataset sees significant accuracy decrease (*Car*, 96.0% vs. 93.9%).

9. Conclusions and Future Research

This paper explores a method for prototype selection based on boosting, and gives statistical criteria for stopping the selection of instances, a crucial problem for the approach (Nock and Sebban, 2001a) as well as for usual boosting algorithms. The whole approach is cast into multiclass classification problems, thereby relaxing the class cardinality constraint of Sebban, Nock and Lallich (2001). So far, the framework proposed in this paper holds only for neighborhood-based classifiers. An

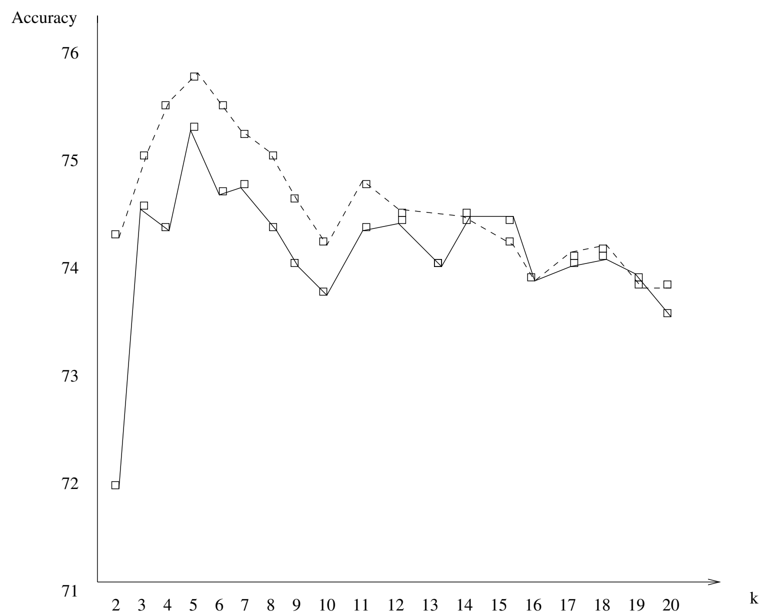

Figure 7: Comparison between a standard k-NN classifier (solid line) and a weighted classifier using the relevance of each instance (dashed-line).

interesting direction of research consists in finding such a method tailored to processing data for induction algorithms, such as, for example, decision tree induction.

Furthermore, we have shown that instead of reducing the learning set size, the boosting-derived weights can be experimentally used in weighted nearest neighbor rules, with statistical advantage compared to the usual, unweighted rules. Because it boils down to making boosting with instances as weak learners that abstain, and because nearest neighbor rules are among the earliest, simplest and still widely used classifiers, this algorithm certainly deserves theoretical investigations to cast, among all, the boosting theory and results (Freund and Schapire, 1997; Schapire and Singer, 1998).

Acknowledgements

The authors wish to thank the referees and Colin de la Higuera for many useful remarks. Our warmest thanks go to Xiaoli Zhang and Carla Brodley, whose invaluable and numerous comments were the basis of a significant improvement of the paper.

References

Erin Allwein, Robert E. Schapire and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1, 113–141, 2000.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen and C.J. Stone. *Classification And Regression Trees*. Wadsworth, 1984.

DATASET	k-NN	WEIGHTED k-NN
AUSTRAL	78.8	**78.9**
BALANCE	82.3	**87.5**
BIGPOLE	**64.7**	64.2
BREAST CANCER	96.5	**96.7**
CAR	**96.0**	93.9
HEART	66.0	**66.7**
ECHOCARDIOGRAM	63.8	**65.2**
GERMAN	72.1	**73.1**
GLASS	**60.7**	60.4
GLASS2	**70.8**	**70.8**
HEART2	78.4	**79.2**
HEPATITIS	78.8	**79.5**
HORSE-COLIC	70.6	**73.3**
IRIS	**95.5**	95.1
LED	64.5	**68.1**
LED+17	24.5	**25.2**
LED2	87.4	**88.1**
LED+17 (2)	**71.2**	70.5
PIMA	68.2	**69.8**
TIC TAC TOE	75.3	**75.6**
VEHICLE	70.0	**71.4**
WHITEHOUSE	**91.3**	90.9
XD6	**79.5**	**79.5**

Table 5: Average accuracy computed over twenty-three datasets by a standard k-NN and a weighted k-NN, where $k = 1, 2, .., 11$; an accuracy in bold font means that this accuracy is the best value.

Carla Brodley and Mark A. Friedl. Identifying and eliminating mislabeled learning instances. In *Proceedings of the Thirteen National Conference on Artificial Intelligence*, pages 799–805, AAAI Press, 1996.

Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13, 21–27, 1967.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2, 263–286, 1995.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, Morgan Kaufmann, 1996.

Yoav Freund and Robert E. Schapire. A decision theoretic generalization of online learning and an application to boosting. *International Journal of Computer and System Sciences*, 55(1), 119-139, 1997.

Geoffrey W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18, 431–433, 1972.

Peter E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14, 515–516, 1968.

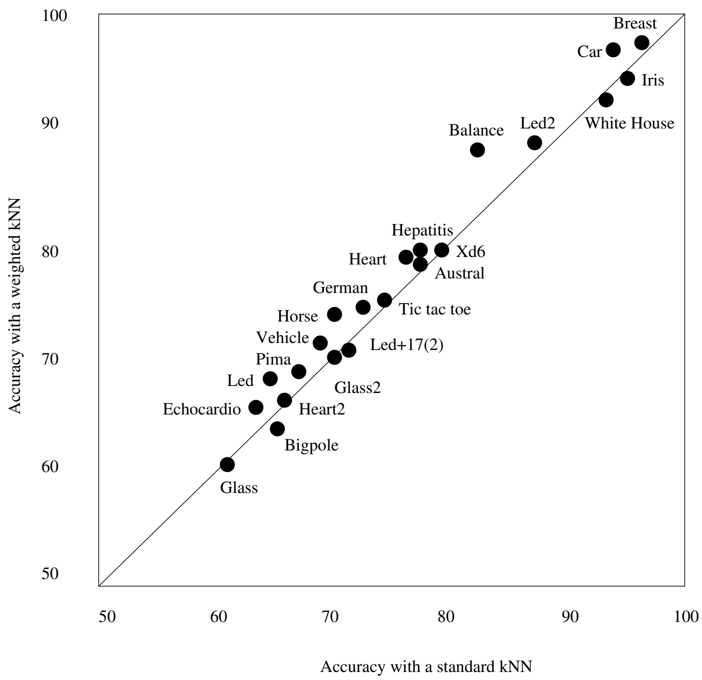

Figure 8: Scatterplot (standard k-NN classifier vs. weighted k-NN classifier) on twenty-three datasets. Each dot under the $y = x$ line depicts a dataset for which the weighted k-NN is not better than the standard k-NN.

Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling, *The Annals of Statistics*, 26(2), 451–471, 1998.

George H. John, Ron Kohavi and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, Morgan Kaufmann, 1994.

Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292, Morgan Kaufmann, 1996.

Christopher J. Merz and Philip M. Murphy. UCI repository of machine learning databases, HTTP://WWW.ICS.UCI.EDU/MLEARN/MLREPOSITORY.HTML, 1996.

Richard Nock and Marc Sebban. Advances in adaptive prototype weighting and selection. *International Journal on Artificial Intelligence Tools*, 10 (1-2), 137–156, 2001a.

Richard Nock and Marc Sebban. An improved bound on the Finite Sample Risk of the Nearest Neighbor Rule. *Pattern Recognition Letters*, 22, 413-419, 2001b.

Richard Nock and Marc Sebban. Sharper bounds for the hardness of prototype and feature selection. In *Proceedings of the International Conference on Algorithmic Learning Theory*, pages 224–237, Springer Verlag, 2000.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 80–91, ACM Press, 1998.

Robert E. Schapire, Yoav Freund, Peter L. Bartlett and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods.*Annals of Statistics*, 26, 1651–1686, 1998.

Marc Sebban. On feature selection: a new filter model. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, pages 230–234, AAAI Press, 1999.

Marc Sebban and Richard Nock. Instance pruning as an information preserving problem. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 855–862, Morgan Kaufmann, 2000.

Marc Sebban, Richard Nock, and Stephane Lallich. Boosting Neighborhood-Based Classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 505–512, Morgan Kaufmann, 2001.

David Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 293–301, Morgan Kaufmann, 1994.

Godfried Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12, 261–268, 1980.

Vladimir N. Vapnik. *Estimation of dependences based on empirical data*. Springer-Verlag, 1982.

Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6, 1–34, 1997.

Randall Wilson and Tony R. Martinez. Reduction techniques for exemplar-based learning algorithms. *Machine Learning*, 38, 257–286, 2000.