A Boosting-Based Prototype Weighting and Selection Scheme

Richard Nock and **Marc Sebban**
TRIVIA-Department of Mathematics and Computer Science
Université des Antilles-Guyane
97159 Pointe-A-Pitre, France
{rnock,msebban}@univ-ag.fr

Abstract

Prototype Selection (PS), *i.e.*, search for relevant subsets of instances, is an interesting Data Mining problem. Original studies of Hart and Gates consisted in producing stepwise a *Condensed* or *Reduced* set of prototypes, evaluated using the accuracy of a Nearest Neighbor rule.
We present in this paper a new approach to PS. It is inspired by a recent classification technique known as Boosting, whose ideas were previously unused in that field. Three interesting properties emerge from our adaptation. First, the accuracy, which was the standard in PS since Hart and Gates, is no longer the reliability criterion. Second, PS interacts with a *prototype weighting scheme*, *i.e.*, each prototype receives periodically a real confidence, its significance, with respect to the currently selected set. Finally, Boosting as used in PS allows to obtain an algorithm whose time complexity compares favorably with classical PS algorithms.
Experiments lead to the following conclusion: the output of the algorithm on fourteen benchmarks is often more accurate than those of three state-of-the-art PS algorithms.

Introduction

With the development and the popularization of new data acquisition technologies such as the World Wide Web (WWW), computer scientists have to analyze potentially huge data bases (DB). Available technology to analyze data has been developed over the last decades, and covers a broad spectrum of techniques, algorithms, statistics, etc. However, data collected are subject to make interpretation tasks hazardous, not only because of their eventually large quantities, but also when these are raw collections, of low quality. The development of the WWW participates to the increase of both tendencies. The reduction of their effects by a suitable preprocessing of data becomes then an important issue in the fields of Data Mining and Machine Learning.

Two main types of algorithms can be used to facilitate knowledge processing. The first ones reduce the number of description variables of a DB, by selecting relevant attributes, and are commonly presented as *feature selection* algorithms (John, Kohavi and Pfleger 1994). The second ones reduce the number of individuals of a DB, by selecting relevant instances, and are commonly presented as *prototype selection* (PS) algorithms (Aha 1990, Gates 1972, Hart 1968, Skalak 1994, Zhang 1992). The principal effect of both types of algorithms is to improve indirectly the reliability and accuracy of post-processing stages, particularly for machine learning algorithms, traditionally known to be sensitive to noise (Blum and Langley 1997). They have also an important side effect. By reducing the "useful" DB size, these strategies reduce both space and time complexities of subsequent processing phases. One may also hope that induction algorithms ran afterwards will obtain smaller and more interpretable formulas, since they are trained on reduced and less noisy datasets.

PS raises the problem of defining relevance for a prototype subset. From the statistical viewpoint, relevance can be understood as the contribution to the overall accuracy, that would be *e.g.* obtained by a subsequent induction. We emphasize that removing prototypes does not necessarily lead to a degradation of the results: we have observed experimentally that a little number of prototypes can have performances comparable to those of the whole sample, and sometimes higher. This definition brings two particular criteria for characterizing less relevant or irrelevant prototypes. The first one obviously concerns noisy regions of the data. Here, ultimately, votes can be considered as randomized and class-conditional probabilities are evenly reparted, which makes such regions excellent candidates for PS. The second one concerns representativeness. The corresponding prototypes typically belong to regions with very few elements. Therefore, their vote is statistically a very poor estimator.

Historically, PS has been firstly aimed at improving the efficiency of the Nearest Neighbor (NN) classifier (Hart 1968). Conceptually, the NN classifier (Cover and Hart 1967) is probably the simplest classification rule. Its use was also spread and encouraged by early theo-

retical results linking its generalization error to Bayes. However, from a practical point of view, this algorithm is not suited to very large DB because of the storage requirements it imposes. Actually, this approach involves storing all the instances in memory. Pioneer work in PS firstly searched to reduce this storing size. In (Hart 1968), Hart proposes a *Condensed NN Rule* to find a *Consistent Subset, CS*, which correctly classifies all of the remaining points in the sample set. However, this algorithm will not find a *Minimal Consistent Subset, MCS*. The *Reduced NN Rule* proposed by Gates (Gates 1972) tries to overcome this drawback, searching in Hart's *CS* the minimal subset which correctly classifies all the learning instances. However, this approach is efficient if and only if Hart's *CS* contains the *MCS* of the learning set, which is not always the case. In such approaches, we have no idea about the relevancy of each instance selected in the prototype subset.

More recently, in (Skalak 1994), Skalak proposes two algorithms to find sets of prototypes for NN classification. The first one is a Monte Carlo sampling algorithm, and the second applies random mutation hill climbing. In these two algorithms, the size of the prototype subset is fixed in advance. Skalak proposes to fix this parameter to the number of classes. Even if this strategy obtains good results for simple problems, the prototype subset is too simple for complex problems with overlaps, *i.e.* when various matching observations have different classes. Moreover, these algorithms require to fix another parameter which contributes to increasing the time complexity: the number of samples (in the Monte Carlo method) or the number of mutation in the other approach. This parameter also depends on the complexity of the domain, and the size of the data. All these user-fixed parameters make the algorithms much user-dependent.

In this paper, we propose a new way to deal with PS. Its main idea is to use recent results about Freund and Schapire's AdaBoost Boosting algorithm (Freund and Schapire 1997, Schapire and Singer 1998). This is a classification technique in which an induction algorithm is repetitively trained, over a set of examples whose distribution is periodically modified. The current distribution favors examples that were badly classified by the previous outputs of the induction algorithm, called *weak hypotheses*. This ensures that the induction algorithm is always trained on an especially hard set of instances (Schapire and Singer 1998). The final output consists of a weighted majority vote of all outputs, where the weight of each weak hypothesis is a real confidence in its predictive abilities. Our adaptation of AdaBoost to PS has the following original features:

- Each step of the algorithm consists in choosing a *prototype* instead of calling for a weak hypothesis. This removes the time spent for repetitive induction. In the PS framework, we avoid the principal criticism often made to Boosting (Quinlan 1996): the prohibitive time complexity.
- Each selected prototype receives a weight, equivalent to that of Boosting for weak hypotheses. This weight can be thoroughly explained in terms of relevance: representativeness and usefulness.
- The best prototype, having the highest coefficient, is kept at each stage of the algorithm. Equivalently, we minimize a criterion derived from Boosting, which is not the accuracy.
- When a prototype is chosen, the distribution of all remaining prototypes is modified. This favors those that are not well explained by the currently selected set.
- The algorithm stops at the user's request. Therefore, one can fix the desired size of the subset. This is the only user-dependent parameter of the algorithm.

When comparing this approach to the previously cited ones, some differences appear, one of which is extremely important to us. The central mechanism for PS is a *dynamic weighting* scheme. Each selected prototype is given a real number which can be reliably interpreted in terms of relevance. Furthermore, whenever a prototype is selected, the distribution of the remaining ones is modified. This will influence and guide the choice of *all* future prototypes, toward those being reliable *and* completing accurately the previously selected prototypes.

The remaining of this paper is organized as follows. First, we introduce the notion of weak hypothesis, and its link with prototype weighting and selection. This is the central mechanism for adapting Boosting to PS. Then, we present the whole PS algorithm. Finally, we present some comparisons with the three previously cited methods on fourteen benchmarks.

From Instances to Prototypes and Weak Hypotheses

Let LS be the set of available instances, to which we usually refer as the *learning sample* in classical machine learning studies. The objective of PS is to select a representative subset of the instances. In this subset, the instances selected become *prototypes*.

Given one new instance to be classified, the classical k-NN algorithm proceeds by taking the majority class among its k nearest neighbors. k-NN is a simple class of local voting procedures where votes are basically unweighted. In a more general setting, if we replace it by an ordinary voting scheme where each voting instance becomes one complex formula, and weighted votes are allowed, then Boosting gives an efficient way to make the whole construction.

Boosting is concerned by the stepwise construction of voting methods, by repeatedly asking for voting, *weak hypotheses*, which, put altogether, form an accurate *strong hypothesis*. The original algorithm of Boosting,

ADABOOST(LS, W, T)
Initialize distribution $D_1(e) = 1/|LS|$ for any $e \in LS$;
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;
Update:
$\forall e \in LS$: $D_{t+1}(e) = \frac{D_t(e)e^{-\alpha_t y(e) h_t(e)}}{Z_t}$;
/*Z_t is a normalization coefficient*/
endFor
Return the classifier

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

Figure 1: Pseudocode for ADABOOST.

called ADABOOST (Freund and Schapire 1997, Schapire and Singer 1998), gives to each currently selected weak hypothesis h_t a real weight α_t, adjusting its vote into the whole final set of hypotheses. While reliable hypotheses receive a weight whose magnitude is large, unreliable hypotheses receive a weight whose magnitude tends to zero. Figure 1 presents ADABOOST in its most general form, described in the two classes framework: the sign of the output formula H gives the class of an observation. When there are $c > 2$ classes, the algorithm builds c voting procedures, for discriminating each class against all others.

The key step of ADABOOST is certainly the *distribution update*. In the initial set of instances, each element can be viewed as having an appearance probability equal to $1/|LS|$ multiplied by its number of occurrences. At run time, ADABOOST modifies this distribution so as to re-weight higher all instances previously badly classified. Suppose that the current weak hypothesis h_t receives a large positive α_t. In ADABOOST's most general setting, each weak hypothesis h_t is allowed to vote into the set $[-1; +1]$, but this is not a restriction, since the role of the α_t is precisely to extend the vote to $\mathbb{R}$ itself. A negative observation e badly classified by h_t has, before renormalization, its weight multiplied by $e^{-\alpha_t y(e) h_t(e)}$. Since $h_t(e) > 0$, $\alpha_t > 0$ and $y(e) = -1 < 0$, the multiplicative factor is > 1, which tends indeed to re-weight higher the example. This would be the same case for badly classified, positive observations.

The adaptation of ADABOOST to NN and then to Prototype Selection (PS) is almost immediate. Suppose that we have access for each instance e to its *reciprocal neighborhood* $R(e) = \{e' \in LS : e \in N(e')\}$, where $N(.)$ returns the neighborhood. Suppose we want to weight all instances in LS. If we consider e as a weak hypothesis, $R(e)$ gives all points in LS for which e will give a vote. The output of this weak hypothesis as such, takes two possible values:

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

For multiclass problems, with $c > 2$ classes, we make the union of c biclass problems, each of which discriminates one class, called $+$, against all others, falling into the same class, $-$. Our PS algorithm is ran separately on each problem, and the overall selected subset of prototypes is the union of each biclass output. This allows to save the notation $+1/-1$ for $y(e)$. The standard ADABOOST does not give the way to choose weak hypotheses. Suppose that we have access to more than one h_t. Which one do we choose ? In our specific framework, the whole set of weak hypotheses is the set of remaining instances, and this question is even more crucial. Freund, Schapire and Singer have proposed a nice way to solve the problem. Name W_e^+ (resp. W_e^-) as the fraction of instances in $R(e)$ having the same class as e (resp. a different class from e), and W_e^0 is the fraction of instances to which e gives a null vote (those not in $R(e)$). Formally,

$$W_e^+ = \sum_{e' \in R(e): y(e') = y(e)} D_t(e')$$

$$W_e^- = \sum_{e' \in R(e): y(e') \neq y(e)} D_t(e')$$

$$W_e^0 = \sum_{e' \in LS \setminus R(e): y(e')} D_t(e')$$

Then, the example e we choose at time t should be the one minimizing the following coefficient:

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

and the confidence α_e can be calculated as

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

In these formulae, the subscript e replaces the t subscript of ADABOOST without loss of generality, since we choose at each time t an example $e \in LS$. In the framework of PS, examples with negative α_e are likely to represent either noise or exceptions. Though exceptions can be interesting for some Data Mining purposes, they are rather risky when relevance supposes accuracy, and they also prevent reaching small subsets of prototypes. We have therefore chosen not to allow the choice of prototypes with negative values of α_e. Algorithm 2, called PSBOOST, presents our adaptation of ADABOOST to PS. Remark that whenever the best remaining instance e has an $\alpha_e < 0$, the algorithm stops and return the current subset of prototypes. It must be noted that this situation was never encountered experimentally.

It is not the purpose of this paper to detail formal reasons for calculating α_e as such, as well as the choice

PSBOOST(LS, N_p)

Input: A learning sample LS of instances, an integer $N_p < |LS|$

Output: A prototype subset $LS' \subseteq LS$, with $|LS'| = N_p$

Initialize distribution $D_1(e) = 1/|LS|$ for any $e \in LS$;

Initialize candidates set $LS_* = LS$;

Initialize $LS' = \emptyset$

For $t = 1, 2, ..., N_p$

$e = argmax_{e' \in LS_*} \alpha_{e'}$;

If $\alpha_e < 0$ Then EndLoop;

$LS' = LS' \cup e$

$LS_* = LS_* - \{e\}$

Update:

$\forall e' \in R(e)$: $D_{t+1}(e') = \frac{D_t(e') e^{-\alpha_{e'} y(e') y(e)}}{Z_e}$;

$\forall e' \in LS_* \backslash R(e)$: $D_{t+1}(e') = \frac{D_t(e')}{Z_e}$;

/*Z_e is a normalization coefficient*/

endFor

Return LS'

Figure 2: Pseudocode for PSBOOST.

of Z_e. Informally, however, we can present a few crucial points of the proofs in the general ADABOOST's settings. First, minimizing the accuracy of the voting procedure can actually be done rapidly by minimizing the normalization coefficient Z_t of ADABOOST. Coefficients α_t can be calculated so as to minimize Z_t, which gives their preceding formula. Then, putting these α_t in the normalization coefficient Z_t gives the formula above. For more information, we refer the reader to the articles (Freund and Schapire 1997, Schapire and Singer 1998). Our adaptation of ADABOOST to PS is mainly heuristic, since we do not aim at producing a classifier, but rather at selecting an *unweighted* set of prototypes.

Some useful observations can be done about the signification of Z_e in the light of what is relevance. First, Z_e takes into account representativeness as defined in the introduction. Indeed, if an instance e belongs to a region with very few prototypes, it is unlikely to vote for many other instances, and W_e^0 will be large, preventing to reach small Z_e. Second, Z_e particularly takes noise into account. Indeed, 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 reach small Z_e. Finally, the distribution update allows to make relevance quite *adaptive*, which appears to be very important when selecting prototypes one by one. Indeed, the distribution is modified whenever a new prototype is selected, to favor future instances that are not well explained by the currently selected prototypes. Finally, note that, as Z_e does, α_e also takes into account relevance, in the same way. The higher α_e, the more relevant prototype e.

Experimental Results and Comparisons

In this section, we apply algorithm PSBOOST. We present some experimental results on several datasets, most of which come from the UCI database repository[1]. Dataset LED is the classical LED recognition problem (Breiman and al. 1984), but to which the original ten classes are reduced to two: even and odd. $LED24$ is LED to which seventeen irrelevant attributes are added. $H2$ is a hard problem consisting of two classes and ten features per instance. There are five features irrelevant in the strongest sense (John, Kohavi and Pfleger 1994). The class is given by the XOR of the five relevant features. Finally, each feature has 10% noise. The $XD6$ problem was previously used by (Buntine and Niblett 1992): it is composed of ten attributes, one of which is irrelevant. The target concept is a disjunctive normal form formula over the nine other attributes. There is also classification noise. Other problems were used as they appeared in the UCI repository in the 1999 distribution.

Each original set is divided into a learning sample LS (2/3 of the instances) and a "validation" set VS (the remaining third). In order to compare performances of PSBOOST with the Hart's, Gates's and Skalak's algorithms, we decided to fix in advance $k = 1$. The experimental set-up applied is the following:

1. the Hart's algorithm CNN is run. We determine the *consistent subset* CS and use this one as a learning sample to test VS with a simple 1-NN classifier.
2. the Gates's algorithm RNN is run. We determine the *minimal consistent subset* MCS and use this one to test VS.
3. the Skalak's algorithm by a Monte Carlo (MC) sampling is applied, with $n_s = 100$ samples. We choose the number of prototypes $N_p = |CS|$, for comparisons.
4. PSBOOST (PSB in the table) is run, fixing in advance $N_p = |CS|$. Note that $|CS|$ is not *a priori* the optimal size of the prototype subset. We fix $N_p = |CS|$ for comparisons.
5. Finally, we compute the accuracy with $|CS|$ prototypes, randomly (*Ran*) chosen into the learning sample LS.

All results are presented in table 1. In the high majority of cases (9 among 14), the $|CS|$ prototypes selected by PSBOOST allow to obtain the best accuracies among all five algorithms. Moreover, if we compare an average accuracy on all the datasets (that has a sense only for the present comparison), we can conclude that PSBOOST seems to be the best approach, not only in accuracy terms, but also as a good compromise between performances and complexity of the algorithm. Moreover, it is very close to the accuracy of the standard 1-NN classifier using all learning instances, that shows

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

Dataset	1NN	CNN	RNN	PSB	MC	Ran
LED24	66.5	64.0	62.5	70.5	66,0	59.0
LED	83.0	69.5	69.5	83.5	82,5	80.0
W. House	92.5	86.0	86.5	88,5	92,0	88.0
Hepatitis	70.9	74.6	72.7	71.8	74,5	61.8
Horse	70.8	67.3	67.3	73.8	69,6	60.1
Echo.	59.0	63.9	63.9	59.0	63,0	60.1
Vehicle	68.2	61.9	61.9	69.7	69,0	61.1
H2	56.6	59.9	59.9	56.1	58,0	54.7
Xd6	73.5	72,5	72,5	71.0	74,0	69.0
Breast W	97.7	96.3	96.3	97.7	97.7	94.0
Pima	69.7	59.0	59.0	73.7	69,0	66.0
German	67.4	60.8	60.8	68.8	65,6	65.6
Ionosph.	91.4	81.9	80.2	87.1	84,3	80.0
Tic-tac-toe	78.2	76.0	76.3	78.5	74,6	74.4
Average	75.1	71.0	70.7	75.0	74,4	69.6

Table 1: Comparisons between five prototype selection algorithms on 14 benchmarks. 1-NN is the standard 1-NN classifier using all learning instances

the interest of our approach which does not compromise the generalization accuracy.

Conclusion

We have proposed in this paper an adaptation of Boosting to Prototype Selection. As far as we know, this is the first attempt to use Boosting in that field. Even more, Boosting was previously sparsely used in Machine Learning to Boost Nearest Neighbor classifiers (Freund and Schapire 1996). In this work, the use of Boosting was also much different from ours, not only in the motivations (Classification, to recognize handwritten digits), but also in the way to grow the strong hypotheses. In particular, weak hypotheses consisted in whole Nearest Neighbor classifiers, and not in simple instances. In the field of PS, the novelty of the Boosting approach stems from the following crucial aspect. PS is achieved by an *adaptive weighting scheme*. Each prototype receives a weight which quantifies its relevance, both in terms of representativeness and usefulness. Equivalently, the prototype to look for can be found at each stage of the algorithm by optimizing a criterion being not the accuracy, which was in turn a common aspect to previous state-of-the art PS algorithms. After a prototype is selected, each remaining instance updates a distributional weight, so as to be weighted higher if it is badly explained by the currently selected prototypes. This is the adaptive part of the selection mechanism.

While we do not use, in this paper, the weight in any classification rule, we can note that this strategy is similar to build a weighted classifier. It deserves then future investigations and comparisons in the field of weighting instances. Actually, first experiments using all the weighted learning instances in a weigthed classification rule (not exclusively a NN rule), seem to show the efficiency of such a method, and will be the subject of future thorough studies.

References

D.W. AHA. A study of instance-based algorithms for supervised learning tasks: Mathematical, empirical, and psychological evaluations. In *Ph.D Dissertation, Dept. of Information and Computer Science*, 1990.

A.L. BLUM and P. LANGLEY. Selection of relevant features and examples in machine learning. *Issue of Artificial Intelligence*, 1997.

L. BREIMAN, J.H. FRIEDMAN, R.A. OLSHEN, and C.J STONE. *Classification And Regression Trees*. Chapman & Hall, 1984.

W. BUNTINE and T. NIBLETT. A further comparison of splitting rules for decision tree induction. *Machine Learning*, pages 75–85, 1992.

S. COST and S. SALZBERG. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, pages 57–78, 1993.

T.M. COVER and P.E. HART. Nearest neighbor pattern classification. *IEEE. Trans. Info. Theory*, IT13:21–27, 1967.

Y. FREUND and R.E. SCHAPIRE. Experiments with a new boosting algorithm. *Proc. of the 13th International Conference on Machine Learning*, 1996.

Y. FREUND and R.E. SCHAPIRE. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, pages 119–139, 1997.

G.W. GATES. The reduced nearest neighbor rule. *IEEE Trans. Inform. Theory*, pages 431–433, 1972.

P.E. HART. The condensed nearest neighbor rule. *IEEE Trans. Inform. Theory*, pages 515–516, 1968.

G.H. JOHN, R. KOHAVI, and K. PFLEGER. Irrelevant features and the subset selection problem. In *Eleventh International Conference on Machine Learning*, pages 121–129, 1994.

J. R. QUINLAN. Bagging, Boosting and C4.5. In *Proc. of AAAI-96*, pages 725–730, 1996.

R. SCHAPIRE and Y. SINGER. Improved boosting algorithms using confidence-rated predictions. In *Eleventh Annual Conference on Computational Learning Theory*, 1998.

D.B. SKALAK. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *11th International Conference on Machine Learning*, pages 293–301, 1994.

J. ZHANG. Selecting typical instances in instance-based learning. In *Proc. of the Ninth International Machine Learning Workshop*, pages 470–479, 1992.