Data Min Knowl Disc (2010) 21:91–129
DOI 10.1007/s10618-010-0173-y

Ensembles of jittered association rule classifiers

Paulo J. Azevedo · Alípio Mário Jorge

Received: 27 March 2009 / Accepted: 6 March 2010 / Published online: 3 April 2010

Abstract The ensembling of classifiers tends to improve predictive accuracy. To obtain an ensemble with N classifiers, one typically needs to run N learning processes. In this paper we introduce and explore Model Jittering Ensembling, where one single model is perturbed in order to obtain variants that can be used as an ensemble. We use as base classifiers sets of classification association rules. The two methods of jittering ensembling we propose are Iterative Reordering Ensembling (IRE) and Post Bagging (PB). Both methods start by learning one rule set over a single run, and then produce multiple rule sets without relearning. Empirical results on 36 data sets are positive and show that both strategies tend to reduce error with respect to the single model association rule classifier. A bias–variance analysis reveals that while both IRE and PB are able to reduce the variance component of the error, IRE is particularly effective in reducing the bias component. We show that Model Jittering Ensembling can represent a very good speed-up w.r.t. multiple model learning ensembling. We also compare Model Jittering with various state of the art classifiers in terms of predictive accuracy and computational efficiency.

Keywords Ensembles · Associative classification · Model jittering

Responsible editor: Johannes Fürnkranz and Arno Knobbe.

P. J. Azevedo (✉)
CCTC, Departamento de Informática, Universidade do Minho, Braga, Portugal
e-mail: pja@di.uminho.pt

A. M. Jorge
DCC - Faculdade de Ciências, Universidade do Porto, Porto, Portugal
e-mail: amjorge@fc.up.pt

A. M. Jorge
LIAAD-INESC Porto L.A., Porto, Portugal

1 Introduction

Association rules can be used for classification purposes with success. It has been empirically proved that predictive performance can be improved by enabling a wider search in the set of patterns supported by the data (Li et al. 2001; Liu et al. 1998; Meretakis and Wüthrich 1999). Given a set of association rules, it is a challenge to devise the best possible way to perform classification, mainly because an enormous number of rules can be produced with reasonable computational resources.

This paper builds up on recent work of the authors, which exploited the use of low-cost ensemble learning (with a single learning process) to further improve the results of association rule classifiers (Jorge and Azevedo 2005; Azevedo and Jorge 2007b). Here we provide a unifying perspective of the previously presented approaches and introduce the concept of Model Jittering Ensembling (MJE). We also present a more extensive and deeper evaluation, showing experimental results for a higher number of data sets and comparing our approach with other existing rule based classifiers, one decision tree based classifier, bagging and boosting. Additionally, we perform an empirical and theoretical study on the computational effort of MJE approaches.

The high level idea of MJE for improving classification with association rules is to generate a first set of rules and then to obtain replications of this set by either sampling the rules in a manner similar to bagging (Breiman 1996) or by reordering it in a process inspired in boosting (Freund and Schapire 1995; Schapire 1990). The replications are then used as an ensemble.

The main feature of this ensemble approach is that the learning process that generates the rules runs only once, whereas in boosting and bagging there are as many learning processes as classifiers generated. In boosting, a sequence of models is generated from iteratively reweighted sets of examples. The weights of the examples are changed, so that misclassified examples get more attention in the next iteration. Bagging is the generation of several models from bootstrap samples of the same original data set. For both approaches, the final decision is obtained by combining the answers of the classifiers in the ensemble. This is also the case in Model Jittering Ensembling.

Model Jittering Ensembling with association rule based classifiers is also a case of learning global model from local patterns (Knobbe et al. 2008), with a further step. First, local patterns (association rules) are discovered. These are then filtered and gathered to obtain a primary global model. This base model, made from local patterns, is particularly adequate for jittering, which is the further step. The ensemble of base models we obtain is the final global model.

In this paper we present and formalize the general process of MJE, and provide further and wider empirical evidence of the strengths and weaknesses of two instances of this approach. We compare MJE with other approaches and see how this new approach is situated in terms of predictive performance. Our direct competitors are a state of the art tree learner, bagging and boosting applied to it, and state of the art rule learners using association rules or not. Empirical results show that MJE improves the performance of single association rule classifiers, and does well with respect to its competitors.

A bias-variance study indicates that IRE improves the predictive accuracy of classification with association rules mainly by reducing the bias component of the

classification error. PB is also successful, although to a lesser extent than IRE, and tends to reduce variance. A study of the speed-up of MJE w.r.t. single model association rule classification reveals very high efficiency values for the two proposed approaches.

In summary, the aims of this work are twofold. First, we exploit for the first time the path of ensembling by perturbing/jittering the models after learning. Second, we apply, with advantage, this general strategy to the process of learning classifiers from association rules.

We also revisit the research done on classification with association rules, particularly on the task of combining rules to produce a decision, and on ensemble learning.

2 Classification with association rules

Association rule (AR) discovery (Agrawal et al. 1996), consists of taking a set of transactions $D = \{T \mid T \text{ is a set of items } i\}$, a minimal support threshold σ and a minimal confidence threshold ϕ, and outputs all the rules of the form $A \rightarrow B$, where A and B are sets of items in D and $sup(A \cup B) \geq \sigma$ and $sup(A \cup B)/sup(A) \geq \phi$. $sup(X)$ is the support or the relative frequency of an item set X observed in D. $sup(A \cup B)/sup(A)$ is the confidence of $A \rightarrow B$, denoted as $conf(A \rightarrow B)$.

AR discovery can be directly applied to tabular data sets, such as the typical UCI data set, with one column for each attribute by regarding each example as a set of items of the form $< attribute = value >$. Likewise, continuous attributes can be dealt with if discretized in advance.

Despite the fact that an association rule algorithm finds ALL rules that satisfy σ and ϕ, the discovery process can be relatively fast and discovery time grows linearly with the number of examples (clearly shown in Agrawal et al. 1996 for the algorithm AprioriHybrid). This provides a scalable heuristic-free process that makes possible to avoid greedy methods such as decision trees.

The discovery of association rules can then be seen as a step preceding model building, or a computationally feasible way of having a non-greedy search on the space of rules. Association rules have been proposed for the first time as complete and competitive classification models by Liu et al. (1998). A classification rule model built from such an unrestrained set of rules can potentially be more accurate than another using a greedy search approach, according to results reported by different authors (Li et al. 2001; Liu et al. 1998; Meretakis and Wüthrich 1999).

In *CBA* (Liu et al. 1998), the produced classifier was a decision list, and each new case was classified by the best rule that applied to it, i.e., the rule with highest confidence. Later, Li et al. (2001) proposed *CMAR* and the use of multiple rules, instead of just one, to classify each new case. The subset of rules that apply to the new case are grouped by answered class, and each of these groups is assessed with a weighted χ^2 heuristic that tried to identify the strongest group. Jovanoski and Lavrac (2001) have studied the effect of simple voting and other simple strategies to improve the prediction ability of a set of association rules ($AprioriC$). Some other approaches have been proposed, such as *HARMONY* by Wang and Karypis (2005), where high confidence rules are generated and grouped w.r.t. their class labels. A class label is assigned to a new case by adding the confidence values of the rules that apply. Earlier

we have proposed the ensemble strategies based on multiple sets of association rules which are under study in this paper (Azevedo and Jorge 2007b; Jorge and Azevedo 2005).

In a slightly different line of work, Meretakis and Wüthrich (1999) suggested a well founded procedure to combine multiple rules by using the confidence of the rules to determine the most likely class for each case, in a kind of naïve Bayes approach with less independence assumptions. Experimental results also indicate the advantage of using association rules for classification.

2.1 Obtaining classifiers from association rules

We can regard classification using association rules as a particular case of the general problem of model combination. Either because we see each rule as a separate model or because we consider subsets of the rules as separate models. We first build a set of rules R. Then we select a subset M of rules that will be used in classification, and finally we choose a prediction strategy π that obtains a decision for a given unknown case x. To optimize predictive performance we can fine tune one or more of these three steps.

2.1.1 Strategy for the generation of rules

A standard approach is to employ a sort of coverage strategy (Liu et al. 1998). All association rules are derived. Then, one chooses the best rule, removes the covered cases and repeat the selection of rules until all cases are covered. In Li et al. (2001) this standard coverage strategy is generalised to allow more redundancy between rules. A case is only removed from the training data when it is covered by a pre-defined number of rules. In *HARMONY* (Wang and Karypis (2005)) rules are generated in a way similar to Liu et al. (1998) and Li et al. (2001), but without the need of setting an explicit value for minimal confidence or other measure apart from support. In our work, we build the set of rules separately using the *CAREN* system (Jorge and Azevedo 2005). *CAREN* is specialized in generating association rules for classification and employs a bitwise depth-first frequent pattern mining algorithm.

2.1.2 Choice of the rule subset

We can use the whole set of rules for prediction, and count on the predictive strategy to dynamically select the most relevant ones. Selection of rules is based on some measure of quality, or combination of measures. The structure of rules can also be used, for example for discarding rules that are generalizations of others. Discarding rules that are potentially irrelevant or harmful for prediction is called *pruning* (Li et al. 2001; Liu et al. 1998; Wang and Karypis 2005).

2.1.3 Strategy for prediction

Most of the previous work on using association rules for classification has been done on this topic. The simplest approach is to go for the rule with the highest quality (best

rule), typically measured as confidence, sometimes combined with support (Liu et al. 1998). Other approaches combine the rules by some kind of *committee method*, such as voting (Jovanoski and Lavrac 2001), or some form of weighted voting (Li et al. 2001; Wang and Karypis 2005).

2.1.4 Rule selection

Rule selection or pruning, can be done right after rule generation. However, most of the rule selection techniques can be applied when the rules are being generated. Pruning techniques rely on the elimination of rules that do not improve more general versions. For example, rule $\{a, b, c\} \rightarrow g$, may be pruned away if rule $\{a, c\} \rightarrow g$ has similar or better predictive accuracy. *CBA* (Liu et al. 1998) uses pessimistic error pruning. Another possibility is to simply use some measure of *improvement* (Bayardo et al. 1999) on a chosen rule quality measure. In our experiments we use the statistical measure of $Fisher$ to evaluate the improvement of rules (Webb 2006). This statistical test for deriving significant rules is described in more detail in the following section.

At modeling time we can still reduce the set of rules by choosing only the K-best ones overall, or the K-best ones for each class (Jovanoski and Lavrac 2001), where K is a user provided parameter. This technique may reduce the number of rules in the model dramatically, but the choice of the best value for N is not clear. In *HARMONY* (Wang and Karypis 2005), a few pruning techniques are used during rule generation, since the rules are built for classification from the first moment. These techniques are non-heuristic, thus maintaining the completeness of the process, while improving computational efficiency.

2.2 Deriving significant rules

A very large number of association rules can be obtained from a given data set. Many of those rules will not be significant, in the sense that they do not add anything (or much) to other existing rules, and must be discarded for computational reasons. In classification tasks, non-significant rules degrade computational efficiency and can negatively affect predictive performance. In our work, we follow the proposal of Webb (2006) to derive significant rules. A rule is significant if the confidence improvement towards all its generalizations is statistically significant.

Definition 1 The rule $x \rightarrow y$ is significant if $\forall z \rightarrow y, z \subset x$ we have that the difference $conf(x \rightarrow y) - conf(z \rightarrow y)$ is statistically significant for a given significance level (α).

If $conf(x \rightarrow y) - conf(z \leftarrow y) > 0$ we say that $x \rightarrow y$ is a productive rule. Following Webb's proposal, *CAREN* implements a test between a rule and its direct generalizations. The direct generalizations of a rule $x \rightarrow y$ are $\emptyset \rightarrow y$ and $(x - a) \rightarrow y$ where a is a single item. Statistical significance is evaluated using a Fisher Exact Test. Furthermore, Webb (2008) proposes a methodology to control type I errors based on layered critical values. The idea is to apply a Bonferroni adjustment on each level (size of rules antecedents). Thus each level will have its own adjusted α'. Although *CAREN*

also implements this methodology, we have observed empirically it tends to degrade predictive accuracy. Thus, in this paper, models were derived using an unadjusted $\alpha = 0.05$.

The process described by Webb is potentially incomplete due to two reasons: First, it only considers direct generalizations of a rule. Second, applied on a depth-first search association rules algorithm (like the one used in Webb 2006 and the one implemented in *CAREN*) it also suffers from incompleteness. Being a depth-first approach, the algorithm derives association rules along the search process. Consequently, when deriving a new rule not all its generalizations are available which disables a complete test. Thus, the pruning process (as implemented) is not guaranteed to be complete. However, this approach is sound, i.e., it guarantees the enforcement of:

$$\begin{aligned} &\forall a \rightarrow cons \in ResultSet,\ \forall x \subset a\ \wedge\ x \rightarrow cons \in ResultSet, \\ &\quad Fisherpvalue(x \rightarrow cons, a \rightarrow cons) \leq \alpha \end{aligned} \tag{1}$$

where $ResultSet$ is the set of association rules returned at the end of the mining process. Hence, completeness can only be ensured by a post-pruning process. The same problem was identified in Webb et al. (2003) using Magnum Opus.

CAREN implements two versions of this pruning process. One is a test applied along the search and the other is a post-pruning test. In this paper we use the latter version.

2.3 Combining the decisions of rules

In this section we describe the two simplest strategies for using association rule sets as classification models. In the discussion we assume we have a static set R of classification association rules, and a predefined set of classes G and that we want to classify cases with description x, where the description of a case is a set of statements involving independent attributes. The set of rules that apply to the case, or that fire upon the case with description x will be $F(x)$ defined as:

$$\{(x' \rightarrow class = g) \in R \mid x' \subseteq x, g \in G\} \tag{2}$$

2.3.1 Best rule

This strategy, from here on referred to as *BestRule* classifies using one single rule $BestRule_x$, is:

$$BestRule_x = arg \max_{r \in F(x)} meas(r) \tag{3}$$

The *meas* used is a function that assigns to each rule a value of its predictive power. *Confidence* is the natural choice when it comes to prediction. It estimates the posterior probability of C given A, and is defined as Eq. 4.

$$confidence(A \rightarrow C) = \frac{sup(A \cup C)}{sup(A)} \tag{4}$$

Conviction is another interest measure (Brin et al. 1997) somewhat inspired in the logical definition of implication and attempts to measure the degree of implication of a rule. Conviction is infinite for logical implications (confidence 1), and is 1 if A and C are independent, and it sometimes outperforms confidence in terms of prediction (Jorge and Azevedo 2005; Azevedo and Jorge 2007a). It is defined as Eq. 5

$$conviction(A \rightarrow C) = \frac{1 - sup(C)}{1 - confidence(A \rightarrow C)} \tag{5}$$

Another measure sometimes used in classification is *Laplace*. It is a confidence estimator that takes support into account, becoming more pessimistic as the support of A decreases. It ranges within [0, 1[and is defined as Eq. 6

$$Lapl(A \rightarrow C) = \frac{sup(A \cup C) + 1}{sup(A) + 2} \tag{6}$$

The prediction given by the best rule is the best guess we can have with one single rule. When the best rule is not unique we can break ties maximizing support (Liu et al. 1998). A kind of *BestRule* strategy, combined with a coverage rule generation method, provided encouraging empirical results when compared with state of the art classifiers on some data sets from UCI (Merz and Murphy 1996).

Our implementation of Best Rule prediction follows closely the rules ordering described in *CMAR* (Li et al. 2001). Thus, R_1 precedes R_2, in terms of rule application, is defined as:

$$\begin{aligned} R_1 \prec R_2 \; if \; & meas(R_1) > meas(R_2) \\ & or \; meas(R_1) == meas(R_2) \wedge sup(R1) > sup(R2) \\ & or \; meas(R_1) == meas(R_2) \wedge sup(R1) == sup(R2) \wedge ant(R1) < ant(R2). \end{aligned}$$

where $meas$ is the used interest measure and ant is the length of the antecedent.

2.3.2 Voting

This strategy, from here on referred to as Voting, combines the rules $F(x)$ that fire upon a case x. The answer of each rule is a *vote*, and the final decision is obtained by assigning a specific weight to each vote, according to its perceived quality. In the case of association rules, this can be done using one of the above defined measures.

$$prediction_{wv} = arg \max_{g \in G} \sum_{x' \in antecedents(F(x))} vote(x', g). \max meas(x' \rightarrow g) \tag{7}$$

A voting variant, VtopK selects the $TopF(x, K)$ best K rules per class from within the ones that fire (Jovanoski and Lavrac 2001) given an example x.

$$pred_{vK} = arg \max_{g \in G} \sum_{x' \in antecedents(TopF(x,K))} vote(x', g). \max meas(x' \rightarrow g) \quad (8)$$

2.4 *CAREN* classification procedure

The association rule generator we are using is *CAREN* (Azevedo 2003). *CAREN* implements an association rule algorithm to derive rule based prediction models. For a given class attribute and training data set, *CAREN* derives association rules where the class values are the consequent. Rule pruning is performed using a Fisher exact test, as described above. Like *CMAR, CAREN* is a rule based rather than an itemset based algorithm. That is, frequent patterns (itemsets) derivation and rule generation occurs in one single step, opposed to itemset based algorithms that make use of two separated steps. This feature tends to yield new pruning opportunities. For example, for a consequent c and an itemset P, where c is the most dominant class in the examples covering P, if $sup(c \cup P) < minsup$ then there is no need to search for a superset P' of P since any rule of the form $P' \rightarrow c$ cannot pass $minsup$. For the same reasons, the statistical based pruning procedures can also yield additional opportunities for pruning.

The set of rules derived constitute the prediction model. Unlike *CMAR* and *CBA, CAREN* does not perform a database coverage procedure. Instead, it considers all derived rules. *CAREN* produces a prediction for a test case by selecting those rules that cover it. Prediction can be performed by a *BestRule* or *Voting* strategy. *BestRule* uses a decision list of rules. Rules are ordered by the ranking described above (see Sect. 2.3.1), first introduced in *CMAR*. The rule at the top of the rank that covers the case provides the prediction. *Voting* uses the values of the interest measures of the rules as weights. The most voted class is the prediction.

3 Related work

Ensemble learning has concentrated a large number of proposals in the literature. In Bauer and Kohavi (1999) a study on the performance of several voting methods (including Bagging and Boosting) was presented. A careful analysis of the bias/variance error decomposition is described as means to explain the error reduction yielded by the different voting methods variants.

Leo Breiman has proposed output smearing and output flipping as methods for generating multiple versions of a data set (Breiman 2000). Both are methods for perturbing outputs through randomization. Output Smearing refers to the adding of Gaussian noise to the outputs. One obvious way of perturbing the output in classification is to alter some of the class labels, but bearing in mind that it is important to keep the class compositions relatively invariant. The extent of this change is measured by a single real parameter, called the flip rate. This procedure is referred to as Output Flipping. Breiman shows experimentally that both perform consistently better than bagging.

In Frank and Pfahringer (2006), the authors also propose a method to improve Bagging. The idea is to modify the base learner so that different models can be generated from the same data. This is typically done by turning the base learner into a randomized version of itself, e.g. by choosing randomly among the best splits at each node of a decision tree The authors refer to this method as "input smearing" because they randomly modify the attribute values of an instance, thus smearing it out in instance space. Hence, they sample from a kernel density estimator of the underlying distribution to form new training sets, in addition to sampling from the data itself. This can be viewed as "smearing out" the resampled training data to generate new data sets. The authors show that the amount of "smear" can be controlled by a parameter. The motivation for using input smearing is that it may be possible to increase the diversity of the ensemble by modifying the input further than bagging does.

A novel version of model aggregation obtained from bagging is described in Martínez-Muñoz and Suárez (2006). The main idea is to derive an ordering on the models and to consider the top of the order. This is obtained by halting the bagging process earlier. Only a small part of the models is selected (15 to 30% of the total). This fraction of models are expected to perform best when aggregated. We have adopted a similar strategy for our *PostBagging* implementation.

Gama (2003) proposes an iterative version of Naive Bayes. The aim is to boost accuracy by iteratively updating the distribution tables yielded from Naive Bayes to improve the probability class distribution associated with each training case. The end product is a single model rather then an ensemble. Our iterative updating of each rule predictive measure within each trial can be seen as a form of improving probability class distribution.

Other authors view rule learning for classification tasks as a special kind of ensemble learning. Each rule is a viewed as a weak learner and the ensemble is defined by assigning a weight to each rule from a rule space, with the possibility of having weights equal to zero (rule selection). Different proposals use different approaches for finding the weights. *SLIPPER* (Cohen and Singer 1999) builds one rule at a time, updating the distribution of the training examples using the principles of AdaBoost. This is done for a fixed number of iterations T. The resulting weighted rule set is used by summing the weights of the rules that apply to a given case. *SLIPPER* is reported as having high predictive accuracy on binary classification data sets when compared with *RIPPER* (Cohen 1995) and $C4.5rules$. The rule set tends to be small.

Lightweight Rule Induction (*LRI*) by Weiss and Indurkhya (2000) proceeds by building one DNF rule at a time (instead of the usual conjunctive rule). After obtaining each rule, the number of errors of rules built so far is updated which affects the generation of further rules. This is similar in spirit to the adaptiveness of boosting. However, rules are unweighted and each class gets the same number of rules. Given a new case, its class is the most voted one. Experimental results show some advantage over *SLIPPER*.

Friedman and Popescu (2005) proposed *RuleFit*, which starts by building a pool of rules. This is obtained by bagging the data and getting a number of classification trees. Each tree is converted into a set of rules. Then, a linear combination of the rules is learned from the data by a regularized linear regression. Experiments indicate that the rule ensemble thus obtained is better than some tree ensembles approaches.

Rückert and Kramer (2006) generate a number of rules and then treat them as attributes of an SVM-like problem. The rule weights are obtained by an optimization procedure. This approach reports competitive results with *SLIPPER*.

More recently, Dembczyński et al. (2008) proposed *MLRules* which obtains a rule ensemble by building rules which minimize negative loglikelihood. The process for building each rule is influenced by previously generated rules, again in the spirit of adaptive boosting. Their experimental results shows that *MLRules* is similar to *LRI*, but superior to *SLIPPER* and *RuleFit*. Their approach is named *MLRules* and has two variants: $MLRules_G$ (which gives the best results) and $MLRules_N$. The difference between the two is that one (G) uses a gradient method as a heuristic to minimize the negative loglikelihood, while the other (N) uses the Newton method.

4 Model jittering ensembling

Model Jittering Ensembling is the production of multiple models by repeatedly perturbing an original one. The nature of the perturbation can be varied and must be adequate to the structure of the model. In our case, we try two approaches on classification models which are sets of association rules. In the first approach we sample rules from an original rule set. In the second approach we reorder the rules, i.e., produce several decision lists. What we observe is that the combinations of sets of rule samples and sets of decision lists obtained from the same rule set improve the results of classification when compared to the use of the original rule set.

In general, ensembles can be produced by perturbing the input data (as it is the case of bagging (Breiman 1996) and boosting Freund and Schapire (1996), as well as output smearing Breiman (2000)), by perturbing or varying the input parameters of the learning algorithm or by using different learning algorithms on the same data (heterogeneous ensembles). What we are proposing here is ensemble production by perturbing the output (the model) of the learning algorithm. When the input data is perturbed, we mostly reduce the variance component of the error. When we change the input parameters or combine complementary algorithms we may also affect the error's bias component. In the case of model jittering, we are able to reduce both variance and bias.

A given model is jittered by applying to it an operator that produces a model of the same type. If the model is a sequence of association rules a simple operator $Sample_K$ can be defined by sampling K rules from the original model. This operator preserves the ordering of the rules. Another operator $Reorder_D$ can reorder the rules by recalculating their measures on a set of examples D. In general, any model composed of smaller parts can be easily jittered by sampling, reordering, clipping, combining parts or changing numerical properties associated to its parts. The special case of an association rule sequence or set is particularly adequate for model jittering due to their highly modular nature. The general algorithm of MJE is shown as Algorithm 1. The generic *JitteringOp* operator takes the original model M and optionally a validation set V (which can be the training set itself). The jittering operator may also have memory and take into account previous perturbation models produced in the same sequence. The operator may also have other specific additional arguments.

Algorithm 1: Model jittering ensembling general algorithm

Input: M: model; V: validation data; K: the number of models in the ensemble
Output: E: set of models (ensemble)

```
1 E ← ∅
2 for i in 1 to K do
3     M_i ← JitteringOp(M[,V,E])
4     E ← E ∪ M_i
```

In this paper we introduce two model jittering ensembling methods for sets of association rules: *PostBagging* and *IterativeReordering*. *PostBagging* is developed around a sampling jittering operator and *IterativeReordering* is based on a reordering operator that recomputes the interest measures of the rules. They will be described in detail in the next two sections.

5 Postbagging

Bagging is the generation of several models from bootstrap samples of the same original data set D (Breiman 1996). The prediction given by the set of resulting models for one example e is done by averaging the predictions of the different models. Bagging has the effect of improving the results of an unstable classifier by reducing its variance (Hastie et al. 2001). Domingos (1997) suggests that, in the case of decision trees, bagging works because it increases the probability of choosing more complex models.

In the case of classification from association, we obtain a large set of rules R that contain many alternative possible models. In this section we describe the technique of *PostBagging* (PB), first introduced in Jorge and Azevedo (2005), and set it as a case of jittering ensembling. PB consists in sampling repeatedly the set of rules a posteriori to obtain an ensemble of models similarly to bagging. The models in a particular ensemble will be similar, but their differences will tend to reflect the variability of rule sets obtained from the same source of data.

New cases are classified by obtaining the prediction of each of the models in the ensemble (and this can be done with any strategy), and using simple voting to combine those predictions. Experimental evaluation indicates that this technique can obtain good results when compared to a *BestRule* or a voting approach, or even to decision tree learners, such as *c4.5* (Quinlan 1993).

We will now describe the *PostBagging* algorithm (Algorithm 2) in detail, as a particular case of the general MJE algorithm. First we obtain a set of association rules R from a data set D using some association rule generator (referred to as $AssocRuleGen$). Each association rule is in fact a triple $< r, absup(r), measure(r) >$, where r is the rule itself, $absup(r)$ is the rule's absolute support, and $measure(r)$ is the rule's predictive measure. Then, we proceed to the model jittering by building a number of *bags* from R using the *PostBaggingOp* operator (Algorithm 3). Each bag is a sample with a pre-defined size of the rule set. Sampling is performed with replacement. Duplicate rules are discarded. The number of bags (K) is 30 by default, and the size T of each

Algorithm 2: PostBagging

Input: D: training_set; K: the number of bags (default 30); T: size of each bag (default $\min(|R|, \max(50, 0.1\times|R|))$)
Output: E: the set of bags/models (ensemble)

```
1 R ← AssocRuleGen(D)
2 E ← ∅
3 for i in 1 to K do
4     M_i ← PostBaggingOp(R as M,T)
5     E ← E ∪ M_i
```

Algorithm 3: PostBaggingOp jittering operator

Input: R: rule set; T: size of each bag
Output: M_i: rule set (bag)

```
1 M_i ← sample with replacement T rules from R
```

bag is 10%, with an absolute minimum of 50 rules. These defaults have been set in preliminary experiments and should not be regarded as necessarily ideal.

5.1 Selecting the top bags

To enhance the performance of the basic *PostBagging* method we apply the idea of using only the composition of the best models. A similar idea of model aggregation obtained from bagging is described in Martínez-Muñoz and Suárez (2006). The main idea is to derive an ordering of the models and to consider the top ones. Only a small part of the models is selected (15 to 30% of the total). This fraction of models are expected to perform best when aggregated. Similarly, we rank the derived bags to choose only the top N models (being N a user provided parameter). In our case, ranking is obtained by computing accuracy of each bag in a validation set, which can be the training set itself, using *BestRule* prediction with a specific interest measure. Thus, after Algorithm 2, each bag is evaluated on training data. To classify new cases, only this fraction of bags are considered. In this case, the final prediction is determined by weighted voting, where the weight of each bag corresponds to its accuracy over the validation set. When there is no bag selection, the voting is uniform.

The classification of a single example x using a set of bags $\{M_i\}$ is done by applying a chosen prediction strategy π to each of the bags. The most voted class is then output as the overall prediction (Algorithm 4). The weight of each model w_i is its accuracy or 1, as explained above. In this paper, this *PostBagging* strategy is referred to as PB.BR.meas, where *meas* is a given interest measure.

6 Iterative reordering

The second jittering operator for association rule models we describe is *IterativeReordering* (IRE). *IterativeReordering* of a set of Association Rules proceeds

Algorithm 4: Ensemble classifier

Input: $\{M_i\}$: set of models; π: prediction strategy; x: unlabelled example; w_i: set of weights
Output: c: class
for *i in 1 to* $|\{M_i\}|$ **do**
    $c_i \leftarrow \pi(M_i, x)$
$c \leftarrow \arg_c \max \sum_{i:c_i=c} w_i$

by re-evaluating the interest and support of each rule according to its performance on a specific data set. This new evaluation works by running the rules on the training set. After that, the interest of each rule is redefined according to its accuracy on this set. The support of a rule is also redefined but as a measure of the usage of the rule in classification. The redefinition yields a new ordering on the original set of rules, which becomes a new model. This process is applied iteratively, thus obtaining a set of rule models that can be aggregated. In this section we formalize *IterativeReordering* as a case of jittering ensembling. IRE was first introduced by Azevedo and Jorge (2007b).

The main idea of IRE is to obtain the models by iteratively reweighting and reordering the rules of the original rule set (Algorithm 5). The initial rule set R is obtained using the association rule generator ($AssocRuleGen$). As above, each rule is represented by a triple $< r, absup(r), measure(r) >$. The values of $measure(r)$ can be used to sort the rules for classification. This predictive measure can be confidence, conviction or Laplace. Other measures could be used, but we have focused on the most predictive ones (Azevedo and Jorge 2007a). The Iterative Reordering operator *IterativeReordOp* is then called K times to generate K models from R.

The operator *IterativeReordOp* (Algorithm 6) returns R itself as model M_1 with original measure values for each rule (base case). The absolute support of each rule will be referred to as $usage(r, M_0, V)$. Model M_2 is obtained by reweighting the rules over a given set of examples, which we call the validation set. In our experiments we have used the training set itself as validation set. Reweighting is performed by evaluating each example $x \in V$ on model M_1 using the *BestRule* approach. Here we count how many times each rule is used in M_1 ($usage(r, M_1, V)$) and how many times it gives the right answer ($hits(r, M_1, V)$). M_2 will contain the rules which have been used and the predictive measure of each rule $measure(r, M_2, V)$ will be updated with the freshly assessed values of $usage(r, M_1, V)$ and $hits(r, M_1, V)$. In subsequent iterations the update will be done in a similar way.

Algorithm 5: Iterative reordering

Input: D: training set; K: max iterations
Output: E: the set of bags/models (ensemble)
$R \leftarrow$ AssocRuleGen(D)
$E \leftarrow \emptyset$
for *i in 1 to* K **do**
    $M_i \leftarrow$ IterativeReordOp(R,D as V,i,E)
    $E \leftarrow E \cup M_i$

Algorithm 6: IterativeReordOp: iterative reordering jittering operator

Input: R: set of rules; V: validation set; i: current iteration;
E: set $\{M_1, \ldots, M_{i-1}\}$ of models from previous iterations
Output: M_i: model

if $i == 1$ **then**
  $M_i \leftarrow R$, with $absup(r)$ as $usage(r, M_0, V)$ and $measure(r)$ as $measure(r, M_1, V)$
else
  if $i == 2$ **then**
    $Pool \leftarrow M_1$
  else
    $Pool \leftarrow M_{i-1} \cup \{triple \in M_{i-2} \mid rule(triple) \notin rules(M_{i-1})\} \cup \{triple \in M_1 \mid rule(triple) \notin [rules(M_{i-2}) \cup rules(M_{i-1})]\}$
  foreach $x \in V$ **do**
    $r \leftarrow$ BestRule(x,$Pool$)
    increment $usage(r, M_{i-1}, V)$
    if $consequent(r) == class(x)$ **then**
      increment $hits(r, M_{i-1}, V)$
  foreach $r \in rules(R)$ **do**
    update $measure(r, M_i, V)$ **from** $usage(r, M_{i-1}, V)$ and $hits(r, M_{i-1}, V)$ (Eqs. 9 to 13)
  $M_i \leftarrow \{< r, usage(r, M_{i-1}, V), measure(r, M_i, V) > \mid usage(r, M_{i-1}, V) > 0\}$

In general, if confidence is the predictive measure in M_{i-1} then in M_i, the confidence of rule $A \rightarrow C$ is:

$$conf(A \rightarrow C, M_i, V) = \frac{hits(A \rightarrow C, M_{i-1}, V)}{usage(A \rightarrow C, M_{i-1}, V)} \tag{9}$$

where

$$hits(A \rightarrow C, M, V) = \#\{x \in V \mid (A \rightarrow C) == BestRule(M, x) \ : \ x \sqsupseteq A \wedge x \sqsupseteq C\} \tag{10}$$

$$usage(A \rightarrow C, M, V) = \#\{x \in V \mid (A \rightarrow C) == BestRule(M, x) \ : \ x \sqsupseteq A\} \tag{11}$$

$BestRule(M, x)$ represents the best rule in M that applies to x. Other interest measures can be defined referring to confidence ($conf$) and support (sup). Conviction is defined as:

$$conv(A \rightarrow C, M_i, V) = \frac{1 - sup(C, V)}{1 - conf(A \rightarrow C, M_{i-1}, V)} \tag{12}$$

where $sup(C, V)$ is the support of class C in validation set V. The measure of Laplace becomes:

$$Lapl(A \rightarrow C, M_i, V) = \frac{hits(A \rightarrow C, M_{i-1}, V) + 1}{usage(A \rightarrow C, M_{i-1}, V) + 2} \tag{13}$$

This reweighting can lead to a different rule ordering in terms of the predictive measure. After M_2, each model in the sequence M_i is obtained from the previous ones in a similar way, until we obtain K models. The main difference for $i > 2$ is that we take rules from a *Pool* (line 7) which contains rules (triples) from M_{i-1}, triples from M_{i-2} with rules not occurring in M_{i-1} and triples from M_1 with rules not occurring in either. This is done so that we have the possibility to consider all rules at each iteration. Notice that each model M_i only has the rules which have been used by the *BestRule* method over the respective *Pool* ($usage(r, M_{i-1}, V) > 0$).

Example 1 Suppose we have the following set of rules in R, produced by $AssocRuleGen$ over a data set V:

$$<r_1, 100, 0.8>$$
$$<r_2, 90, 0.7>$$
$$<r_3, 10, 1>$$

M_1 will be equal to R (line 2). In subsequent iterations, we will have $usage(r_1, M_0, V) = 100$ and $measure(r_1, M_1, V) = 0.8$. Similarly for the other rules. Now, in the second iteration we build M_2. The *Pool* will be equal to M_1 (line 5). Then we apply the rules in the *Pool* using *BestRule* and measure their usage ($usage(r_i, M_1, V)$) and hits ($hits(r_i, M_1, V)$). For each rule we recompute its predictive measure ($measure(r_i, M_2, V)$) from usage and hits (line 13). Suppose r_3 has usage zero and M_2 is:

$$<r_1, usage(r_1, M_1, V) = 90, measure(r_1, M_2, V) = 0.75>$$
$$<r_2, usage(r_2, M_1, V) = 100, measure(r_2, M_2, V) = 0.8>$$

M_2 has only two rules, but their order is not the same as before. In the third iteration we build M_3. The *Pool* is made of M_2 plus $< r_3, 10, 1 >$ (line 7). The rules in this *Pool* are applied using *BestRule* and we obtain $usage(r_i, M_2, V)$ and $hits(r_i, M_2, V)$. Then we calculate $measure(r_i, M_3, V)$. Suppose now only the first rule survives, with recalculated usage, hits and measure. In the fourth iteration, the *Pool* will be made of r_1 as it appears in M_3, r_2 with values from M_2, and r_3 from M_1. Assuming that M_4 corresponds to the last iteration, the final ensemble will be the set $\{M_1, M_2, M_3, M_4\}$ (Algorithm 5).

Similarly to AdaBoost (Freund and Schapire 1995) model construction, the Iterative Reordering process can stop earlier either if a very good or very bad model accuracy is achieved. In our experiments the condition for an early stop has been $accuracy(M_i) < 0.5 \vee accuracy(M_i) > 0.99$.

The ensemble $\{M_i, i = 1 \ldots K\}$ is used to classify new cases (Algorithm 4). In the case of IRE we adopted two weighting strategies. In the first one, the weight w_i is the global accuracy of the model M_i on the data. In the second, the weight w_i the local interest measure of the best rule that fired within the M_i model. In the remainder

(mainly in the experimental section), the latter will be referred as *IRE.loc.meas* and the former as $IRE.glob.meas$, where *meas* is the interest measure.

6.1 Discussion

The intended effect of IRE is that rule ordering is recomputed taking into account global effects on accuracy, instead of local ones. In Iterative Reordering Ensembling, a new model is generated by changing the order of the rules, where rules with more errors go down. Thus, misclassified examples improve their chance of being well classified. One important difference w.r.t. boosting is the fact that one single learning step is used, whereas in boosting there are as many learning steps as models in the ensemble.

In terms of implementation, notice that the minimal required information to represent rule models (also known as trials) is the $usage$ and $hits$ associated with each rule. A matrix with $2 \times K$ (for K trials) is enough to represent the ensemble. We do not actually need to represent the rules multiple times.

7 Experimental validation

We have conducted experiments comparing the predictive performance of the jittering ensembling approach with the single model best-rule strategy. We have tried different prediction measures and state-of-the-art algorithms. We have used 36 UCI data sets (Merz and Murphy 1996), which greatly extends our previous studies (Jorge and Azevedo 2005; Azevedo and Jorge 2007b). The data sets are described in Table 1. As a reference algorithm, we used the decision tree inducer $J48$, implemented in WEKA (Witten and Frank 2005) as a version of $C4.5$ (Quinlan 1993). We have also measured the predictive performance of applying Bagging and AdaboostM1 to $J48$, also from WEKA. Default parameters were used with these algorithms (ten iterations, bag size = 100%). All the association rule classifiers were generated with *CAREN*[1] (version 2.5.2), our own association rule engine implementation. The jittering ensembling approaches are also implemented in *CAREN*.

Additionally, we ran six other classifiers on the data sets in order to be able to situate the current performance of the jittering approaches. We have considered two classical rule based classifiers, *RIPPER* (Cohen 1995) and *PART* (Frank and Witten 1998) (WEKA implementations), one association rule based classifier *CMAR* (Li et al. 2001) and three other classifiers based on rules ensembles, *SLIPPER* (Cohen and Singer 1999) (kindly provided by William W. Cohen), *RuleFit* (Friedman and Popescu 2005) (available R implementation[2]) and $MLRules_G$ (gradient) (Dembczyński et al.

[1] http://www.di.uminho.pt/~pja/class/caren.html.

[2] http://www-stat.stanford.edu/~jhf/R-RuleFit.html.

Table 1 Data sets used for the empirical evaluation

Data set	#Expls	#Class	#Attr	#Num	Nulls
Adult48	48842	2	14	6	Yes
Anneal	898	6	38	6	Yes
Australian	690	2	14	6	No
Breast	699	2	9	9	Yes
Chess-kr-k	28056	18	6	0	No
Chess-kr-kp	3196	2	36	0	No
Cleveland	302	5	13	6	Yes
crx	690	2	15	6	Yes
Flare	1066	2	10	0	No
German	1000	2	20	7	No
Glass	214	7	9	9	No
Heart	270	2	13	6	No
Hepatitis	155	2	19	6	Yes
Horse	368	2	22	7	Yes
House-vote	435	2	16	0	Yes
Hypo	3163	2	25	7	Yes
Iono	351	2	34	34	No
Iris	150	3	4	4	No
Led7	3200	10	7	0	No
Lympho	148	4	18	0	No
Mushroom	8123	2	22	0	Yes
Nursery	12960	5	8	0	No
Pageblocks	5473	5	10	10	No
Pendigits	10992	10	16	16	No
Pima	768	2	8	8	No
Sat	6435	6	36	36	No
Segment	2310	7	19	19	No
Shut	58000	7	9	9	No
Sonar	208	2	60	60	No
Soybean	307	19	35	35	No
Tic-tac-toe	958	2	9	0	No
Vehicle	846	4	18	18	No
Waveform	5000	3	21	21	No
Wine	178	3	13	13	No
Yeast	1484	10	8	8	No
Zoo	101	7	16	0	No

2008) (WEKA add-on provided by the authors). Since we were not able to obtain a running version of *CMAR* we have used our own implementation following the description in Li et al. (2001). We should stress that we were not able to reproduce the results

reported in that paper. Other authors have reported the same difficulty (Zimmermann and Raedt 2004).

CMAR was implemented by making use of the *CAREN* association rules engine and following the description in Li et al. (2001). Like *CAREN*, it also finds frequent patterns and generates rules in one step. Rule pruning uses improvement (Bayardo et al. 1999) between a rule and all its generalizations. In Li et al. (2001) this is referred as confidence difference. The minimal difference threshold is provided by the user. In our experiments we used 0.02 as confidence difference (as suggested by Li et al. 2001). A χ^2 test between the antecedent and consequent of a rule is also performed to discard rules that are not positively correlated. This step is executed as soon as a rule is derived. The rule rank is the one described in Sect. 2.3.1. We have also implemented the database coverage algorithm for rule selection with the delta coverage threshold. As in Li et al. (2001), we used delta = 4. Finally, we implemented the voting schema described in *CMAR* (multiple rules classification using weighted χ^2 voting) to derive predictions. For experiments with *CMAR* we used the same minsups and minconf as with *CAREN* methods.

Error estimation for all the methods presented were obtained with stratified tenfold cross-validation using WEKA generated folds. *SLIPPER* and *RuleFit* have only been applied on two classes data sets (also referred as binary data sets) due to limitations of the methods or of the implementation, in the case of *SLIPPER* (Dembczyński et al. 2008).

For the single model association rule classifiers, we used three variants, using the strategy *BestRule* with three measures (confidence, conviction and Laplace). Recent experiments with many other measures (*CAREN* Azevedo 2008 implements 13 different interest measures), as well as previous experiments (Azevedo and Jorge 2007a) showed that these are the three most predictive ones. We have also used two voting strategies (*Voting* and $VtopK$ with $K = 5$) combined with the three measures mentioned above. *Voting* is a weighted voting strategy using Eq. 7. $VtopK$ criteria is defined as Eq. 8. For the model jittering approaches, we ran *PostBagging* and *IterativeReordering* with each of the three measures.

In the case of *CAREN* methods and our *CMAR* implementation, numerical attributes were discretized for the training sets, and for each fold, using *CAREN*'s implementation of the supervised discretization method of Fayyad and Irani (1993). Test set information is not used for discretizing the training data. The test set is evaluated using the original raw format. Nulls in the training sets were replaced by most frequent value within each class. Nulls in the test sets were ignored. All other methods used the original raw data sets.

The minimal supports (minsup) used depend on the size of the training data $|T|$. If $|T| < 1000$, then minsup is 0.01. If $1000 \leq |T| < 2000$ minsup is 0.005, if $2000 \leq |T| < 10000$ minsup is 0.001, if $10000 \leq |T| < 20000$ minsup is 0.005, if $|T| \geq 20000$ minsup 0.0001. The key idea is that we move to a lower level of minsup when the absolute minimum support for a given training set reaches ten examples (i.e. $1000 \times 0.01 = 10$, $2000 \times 0.005 = 10$, etc.). For computational reasons *CAREN* was not able to run in a reasonable amount of time on three of the data sets with these values of minsup: the "sat" data set, where the minimal applicable minsup was 0.02, "waveform" (0.01) and "kr–kp" (0.1). Also for computational reasons we have lim-

ited the size of the rules for this last data set to a maximum of seven items, including the consequent. Minimal confidence was 0.5. We have also used the $Fisher$ filter to eliminate spurious rules. For the model jittering approaches we have used the training set as validation set.

For *IRE* we have used the two different model combination strategies described in Sect. 6: local (IRE.loc), where each model votes with the decision of its best rule; and global (IRE.glob), where each model votes with the decision of its best rule, weighted with the accuracy of the model on the validation set. The parameters of the jittering strategies were as follows. Each *PostBagging* process generated 300 models (bags), where each model contains 30% of the original rules. From these models, only the top 51 (the number is odd to avoid ties) were selected for the ensemble. This selection step in a bagging process is important for predictive accuracy (Martínez-Muñoz and Suárez 2006). In the case of *IterativeReordering* we used ten iterations (models).

Since *SLIPPER* and *RuleFit* only apply to binary datasets, we perform a first set of experiments with all the data sets, but without those methods. Then, a second set of experiments on the 17 binary data sets, using is all the methods, is performed. Estimated error rates obtained are shown in Appendix (Tables 10 to 13). From the error rates, we obtained the ranks of each method, for each experiment. We discuss the results in the following sections.

7.1 Results with all the data sets

The ranks for the methods that can deal with any number of classes are shown in Tables 2 and 3. With no restricting the number of classes, we observe that $MLrules_G$ is the top ranked method (Table 2). This is followed by $Bagging(J48)$ and $AdaBoost(J48)$. $IRE.glob.conv$ is the fourth overall and the top ranked association rule classifier. To assess the statistical significance of the results we have performed a Friedman test (Demšar 2006). The test rejects the null hypothesis that all methods perform equally (p-value is $2.2e^{-16}$). The corresponding Nemenyi critical value, for $\alpha = 0.05$, is of 5.499 (for 22 models and 36 data sets). A post-hoc analysis using this critical value accepts the hypothesis that all top methods from $MLRules_G$ to *RIPPER* are in the same group, since the difference of their ranks is below 5.499. Under this perspective, the best *IRE*, which has a mean rank of 7.92 significantly outranks the methods with mean rank above 13.419 (=7.92+5.499). These are mostly *CAREN* voting strategies and two PostBagging. The method *CMAR* is also significantly outranked.

We also observe the following:

- All *IRE* methods, independently of the measure, are competitive with the state of the art approaches. W.r.t. *PB*, only PB.BR.conv is in this group.
- IRE.glob.conv significantly outranks Vtop5.lapl, Voting.conf and Voting.lapl.
- W.r.t. the *PB* strategies, *AdaBoost* is only significantly better than PB.BR.lapl. $Bagging$ is also significantly better than PB.BR.conf.
- All *IRE* methods rank higher than the respective *BestRule* for the same measure. Although we cannot observe a statistically significant evidence for this, this is a very consistent trend.

Table 2 Ranks of the 11 best algorithms for data sets with any number of classes

	MLRules	Bagging	ADABoost	IRE.glob.conv	IRE.loc.conf	J4.8	PART	IRE.loc.lapl	PB.BR.conv	RIPPER	BR.conv
Mean	4.94	5.83	7.74	7.92	8.82	9.21	9.63	9.88	10.17	10.28	11.28
Adul48	1.0	2.0	15.0	9.0	10.0	3.0	6.0	8.0	12.0	14.0	4.0
Aust	1.0	3.0	10.0	12.0	14.5	4.0	13.0	7.0	7.0	2.0	18.0
Brea	3.0	6.0	7.0	5.0	2.0	17.0	21.0	1.0	8.5	13.0	11.5
Clev	4.0	13.0	16.0	14.5	17.5	21.0	22.0	14.5	1.0	20.0	5.0
Flar	10.0	9.0	18.0	21.0	13.5	1.0	17.0	13.5	19.5	6.0	19.5
Germ	1.0	3.5	19.5	5.5	16.0	19.5	7.0	13.0	10.0	12.0	5.5
Hear	7.0	15.5	21.0	9.0	5.5	20.0	22.0	9.0	14.0	19.0	18.0
Hepa	7.5	7.5	1.0	11.0	4.0	6.0	3.0	17.0	5.0	19.0	13.0
Hous	3.5	1.5	3.5	12.0	6.0	1.5	7.0	9.0	8.0	5.0	17.0
Lymp	12.0	17.5	21.0	7.5	7.5	17.5	13.0	4.0	1.0	15.0	3.0
Pima	7.0	16.0	22.0	8.0	5.0	18.0	11.0	6.0	2.0	1.0	3.0
Segm	3.0	2.0	1.0	8.0	7.0	4.0	5.0	9.0	11.5	6.0	15.5
Shut	10.0	15.0	3.0	1.0	2.0	12.0	9.0	6.0	7.0	16.0	4.0
Vehi	3.0	1.0	2.0	8.0	7.0	4.0	5.0	9.0	15.0	6.0	20.5
Wave	1.0	10.0	15.0	11.5	13.0	22.0	21.0	14.0	6.0	20.0	19.0
Yeas	2.0	1.0	4.0	6.0	9.0	7.0	11.0	17.0	5.0	3.0	8.0
Glas	3.0	1.0	2.0	5.0	4.0	16.0	7.0	6.0	10.0	13.0	12.0
Wine	3.0	1.5	1.5	5.0	5.0	11.0	18.0	9.0	7.5	14.0	16.0
Zoo	7.0	9.0	2.0	5.5	5.5	9.0	9.0	14.0	3.5	19.0	15.5
Hypo	4.0	2.0	6.0	9.0	12.0	1.0	3.0	18.0	7.0	5.0	8.0
Hors	1.0	2.0	15.0	7.0	16.0	3.5	5.0	6.0	17.0	3.5	8.0
Led7	8.0	1.0	6.5	4.5	2.0	6.5	4.5	3.0	13.0	22.0	17.0
Iris	2.5	2.5	18.0	13.0	13.0	1.0	13.0	13.0	6.5	13.0	6.5

Table 2 continued

	MLRules	Bagging	ADABoost	IRE.glob.conv	IRE.loc.conf	J4.8	PART	IRE.loc.lapl	PB.BR.conv	RIPPER	BR.conv
crx	2.0	1.0	15.0	13.0	12.0	3.5	6.0	5.0	10.5	3.5	18.0
tic.	13.0	14.0	8.0	1.0	2.5	18.0	12.0	2.5	11.0	7.0	4.0
Soyb	9.0	3.0	1.0	4.0	6.0	8.0	5.0	7.0	12.5	2.0	14.5
Sona	1.5	3.0	1.5	10.5	10.5	4.0	13.0	9.0	19.0	17.0	21.5
Sat	3.0	2.0	1.0	7.0	8.0	6.0	5.0	9.0	10.0	4.0	11.0
Anne	1.0	7.0	2.0	5.5	12.5	10.0	4.0	15.0	16.0	3.0	5.5
Iono	1.0	14.0	3.0	5.0	2.0	17.0	6.0	4.0	11.5	18.0	8.5
Mush	8.0	8.0	8.0	8.0	8.0	8.0	8.0	16.0	8.0	8.0	8.0
kr.k	22.0	4.0	1.0	8.0	12.0	11.0	17.0	10.0	18.0	21.0	5.0
kr.kp	1.0	3.5	2.0	6.0	9.0	3.5	8.0	7.0	17.0	5.0	11.0
Nurse	3.0	5.0	1.0	4.0	18.0	7.0	2.0	17.0	13.0	8.0	10.0
Page	3.0	1.0	4.0	8.0	12.0	6.0	5.0	19.0	11.0	2.0	7.0
Pend	6.0	2.0	1.0	7.0	8.0	4.0	3.0	9.0	11.5	5.0	14.5

First line shows mean rank of the algorithm in the column. The columns are sorted by mean rank

Table 3 Ranks of the 11 worst algorithms for data sets with any number of classes

	PB.BR.conf	BR.conf	Vtop5.conf	Vtop5.conv	Voting.conv	BR.lapl	PB.BR.lapl	CMAR	Vtop5.lapl	Voting.conf	Voting.lapl
Mean	12.12	13.03	13.03	13.43	13.47	14.40	14.47	14.65	15.74	16.19	16.78
Adl48	18.0	11.0	20.0	5.0	7.0	13.0	19.0	22.0	21.0	16.0	17.0
Aust	9.0	17.0	14.5	19.0	21.0	11.0	7.0	5.0	16.0	21.0	21.0
Brea	11.5	14.0	15.5	10.0	8.5	22.0	20.0	4.0	15.5	18.5	18.5
Clev	6.0	19.0	7.5	2.5	2.5	17.5	10.0	12.0	7.5	10.0	10.0
Flar	13.5	13.5	2.0	5.0	4.0	13.5	13.5	22.0	3.0	7.0	8.0
Germ	21.0	14.5	14.5	3.5	11.0	17.0	22.0	2.0	18.0	8.5	8.5
Hear	12.0	17.0	5.5	12.0	9.0	15.5	12.0	3.5	3.5	1.5	1.5
Hepa	21.0	16.0	14.5	9.0	2.0	18.0	20.0	22.0	14.5	11.0	11.0
Hous	10.0	14.0	15.5	18.0	20.0	13.0	11.0	19.0	15.5	21.5	21.5
Lymp	6.0	10.0	2.0	20.0	22.0	10.0	5.0	14.0	10.0	16.0	19.0
Pima	9.0	10.0	20.5	14.0	15.0	13.0	12.0	4.0	20.5	19.0	17.0
Segm	11.5	15.5	10.0	22.0	21.0	19.0	18.0	20.0	17.0	13.0	14.0
Shut	8.0	5.0	11.0	13.5	13.5	19.0	18.0	22.0	17.0	20.0	21.0
Vehi	16.0	20.5	14.0	19.0	12.0	22.0	18.0	13.0	17.0	10.0	11.0
Wave	5.0	18.0	3.0	11.5	2.0	17.0	7.0	16.0	4.0	8.0	9.0
Yeas	15.0	10.0	19.5	13.5	13.5	12.0	16.0	22.0	19.5	19.5	19.5
Glas	11.0	8.0	20.0	18.0	19.0	9.0	14.0	21.0	22.0	15.0	17.0
Wine	7.5	16.0	5.0	19.5	19.5	16.0	10.0	13.0	12.0	21.0	22.0
Zoo	3.5	15.5	1.0	20.5	20.5	22.0	17.0	11.0	18.0	12.0	13.0
Hypo	13.0	11.0	14.0	10.0	15.0	21.0	20.0	22.0	19.0	16.0	17.0
Hors	12.0	10.0	13.0	18.0	19.0	10.0	10.0	20.0	14.0	21.5	21.5
Led7	10.0	16.0	13.0	9.0	19.0	15.0	11.0	18.0	13.0	20.5	20.5
Iris	6.5	6.5	20.5	13.0	13.0	6.5	6.5	17.0	20.5	20.5	20.5

Table 3 continued

	PB.BR.conf	BR.conf	Vtop5.conf	Vtop5.conv	Voting.conv	BR.lapl	PB.BR.lapl	CMAR	Vtop5.lapl	Voting.conf	Voting.lapl
crx	10.5	17.0	16.0	19.0	20.0	9.0	8.0	7.0	14.0	21.5	21.5
tic.	15.0	5.5	21.0	9.5	9.5	5.5	17.0	16.0	22.0	20.0	19.0
Soyb	12.5	14.5	11.0	20.0	19.0	18.0	17.0	10.0	16.0	21.0	22.0
Sona	18.0	21.5	5.0	12.0	8.0	20.0	14.5	16.0	14.5	6.5	6.5
Sat	12.0	15.0	18.0	16.0	19.0	14.0	13.0	20.0	17.0	21.0	22.0
Anne	17.0	12.5	21.0	8.0	9.0	14.0	20.0	11.0	22.0	18.0	19.0
Iono	11.5	8.5	15.0	19.5	19.5	7.0	10.0	13.0	16.0	21.0	22.0
Mush	8.0	8.0	8.0	18.5	18.5	8.0	17.0	8.0	20.0	21.0	22.0
kr.k	19.0	7.0	13.5	2.0	3.0	6.0	20.0	9.0	15.5	13.5	15.5
kr.kp	15.5	11.0	18.5	13.0	14.0	11.0	15.5	20.0	18.5	21.5	21.5
Nurse	14.0	11.0	19.0	9.0	6.0	12.0	15.0	16.0	20.0	21.0	22.0
Page	16.0	15.0	17.0	10.0	9.0	22.0	21.0	18.0	20.0	13.0	14.0
Pend	11.5	14.5	10.0	21.5	21.5	20.0	16.0	19.0	13.0	17.0	18.0

First line shows mean rank of the algorithm in the column. The columns are sorted by mean rank

- *PB* methods rank higher than the respective *BestRule* for the measures confidence and conviction, but not for Laplace. The advantage of *PB* is less clear than the one of *IRE*.
- All jittering approaches beat the voting approaches (for the same measure), except for PB.BR.lapl.
- All BestRule approaches outrank the corresponding voting strategies for the same measure.
- The best *IRE* approach ranks higher than $J48$ but not higher than *AdaBoost* or $Bagging$.
- The two best *IRE* approaches beat *PART* and *RIPPER*.
- While all *BestRule* approaches are worse than $J48$, *PART* and *RIPPER*, the two best *IRE* approaches manage to "boost" *BestRule* results in order to outrank those methods.
- All $Vtop5$ approaches beat the respective *Voting* approach.

As we can see from the above statements, despite the intuitive evidence of the superiority of *IRE* w.r.t. *BestRule*, the standard procedure for multiple comparisons does not allow to unveal this tendency. This is explained by the adjustments made on the critical values in order to avoid spurious conclusions obtained by mere repetition of tests. This is obviously the case when one wants to prove that method A is better than any method B from a large set of methods. In our case, we want to test very specific pre-defined hypotheses concerning the virtues of jittering ensembling. In particular we want to assess if *IRE* with a given measure (out of the three mentioned) is able to improve *BestRule* for the same measure. For this, we performed three independent Wilcoxon signed rank tests (one sided, using *R* Statistical Package, R Development Core Team 2004). All three rejected the null hypothesis that IRE.meas performs equally well as BR.meas ($\alpha = 0.05$). Equivalent tests for *PostBagging* w.r.t *BestRule* show a statistically significant advantage of PB.BR.conv over BR.conv, but not for the other two measures.

7.2 Results on the binary data sets

The experiment for the binary data sets only, includes two more methods: *SLIPPER* and *RuleFit* (Tables 4 and 5). In this case *SLIPPER* is the top method beating $MLRules_G$ (the second best) on 12 out of 17 data sets. These experiments, with a different group of data sets, do not confirm the results claimed in Dembczyński et al. (2008). The *IRE* approaches are once more well positioned.

The Friedman test for these data sets only, also rejects the hypothesis that all methods perform equally well (p-value is $4.07e^{-12}$). The Nemenyi critical value for 24 methods and 17 data sets is 8.821 ($\alpha = 0.05$). This implies that the methods from *SLIPPER* to $BR.conv$ (sorted by mean rank) are in the same group. For this subgroup of data sets we observe once more that the three *IRE* methods are better than any of the *BestRule* methods. W.r.t. to *PB*, advantage over *BestRule* is only observed in the case of conviction.

This experiment with the binary data sets confirmed most of the previous observations:

Table 4 Ranks of the methods for 8 of the 17 binary data sets

	Mean	Adul48	Aust	Brea	Flar	Germ	Hear	Hepa	Hous
SLIPPER	4.29	12.0	6.0	1.0	2.0	1.0	1.0	1.0	1.0
MLRules	5.15	1.0	1.0	4.0	11.0	2.0	9.0	8.5	4.5
Bagging	7.68	2.0	3.0	7.0	10.0	5.5	17.5	8.5	2.5
J4.8	9.91	3.0	4.0	19.0	1.0	21.5	22.0	7.0	2.5
IRE.loc.conf	9.91	11.0	16.5	3.0	14.5	18.0	7.5	5.0	7.0
IRE.loc.lapl	10.12	8.0	9.0	2.0	14.5	15.0	11.0	18.0	10.0
IRE.glob.conv	10.29	10.0	14.0	6.0	23.0	7.5	11.0	12.0	13.0
RIPPER	10.53	16.0	2.0	15.0	7.0	14.0	21.0	20.0	6.0
PART	11.24	6.0	15.0	23.0	18.0	9.0	24.0	4.0	8.0
ADABoost	11.68	17.0	12.0	9.0	19.0	21.5	23.0	2.0	4.5
PB.BR.conv	12.47	14.0	9.0	10.5	21.5	12.0	16.0	6.0	9.0
RuleFit	12.65	9.0	5.0	8.0	20.0	3.0	2.0	24.0	22.0
BR.conv	12.85	4.0	20.0	13.5	21.5	7.5	20.0	14.0	18.0
Vtop5.conv	14.00	5.0	21.0	12.0	6.0	5.5	14.0	10.0	19.0
BR.conf	14.32	13.0	19.0	16.0	14.5	16.5	19.0	17.0	15.0
Voting.conv	14.35	7.0	23.0	10.5	5.0	13.0	11.0	3.0	21.0
CMAR	14.74	24.0	7.0	5.0	24.0	4.0	5.5	23.0	20.0
BR.lapl	14.79	15.0	13.0	24.0	14.5	19.0	17.5	19.0	14.0
PB.BR.conf	14.97	20.0	11.0	13.5	14.5	23.0	14.0	22.0	11.0
Vtop5.conf	15.24	22.0	16.5	17.5	3.0	16.5	7.5	15.5	16.5
PB.BR.lapl	16.15	21.0	9.0	22.0	14.5	24.0	14.0	21.0	12.0
Vtop5.lapl	17.15	23.0	18.0	17.5	4.0	20.0	5.5	15.5	16.5
Voting.conf	17.71	18.0	23.0	20.5	8.0	10.5	3.5	12.0	23.5
Voting.lapl	17.82	19.0	23.0	20.5	9.0	10.5	3.5	12.0	23.5

First column shows mean rank of the algorithm in the column. The lines are sorted by mean rank

- All *IRE* methods, independently of the measure, are competitive with the state of the art approaches. W.r.t. *PB*, only PB.BR.conv is in this group.
- All *IRE* methods rank higher than the respective *BestRule* for the same measure.
- *PB* methods rank higher than the respective *BestRule* for the measure confidence, but not for Laplace and conviction.
- All jittering approaches beat the voting approaches with no exceptions in this case.
- All BestRule approaches outrank the corresponding voting strategies for the same measure.
- The two best *IRE* approaches beat *PART* and *RIPPER*.
- While all *BestRule* approaches are worse than *PART* and *RIPPER*, the two best *IRE* approaches manage to "boost" *BestRule* results in order to outrank those methods.
- All $Vtop5$ approaches beat the respective *Voting* approach.

Table 5 Ranks of the methods for the other nine binary data sets

	Mean	Pima	Hypo	Hors	crx	tic.	Sona	Iono	Mush	kr–kp
SLIPPER	4.29	1.0	3.0	1.0	1.0	16.0	1.0	1.0	21.0	3.0
MLRules	5.15	9.0	5.0	2.0	3.0	14.0	2.5	2.0	8.0	1.0
Bagging	7.68	18.0	2.0	3.0	2.0	15.0	5.0	16.0	8.0	5.5
J4.8	9.91	20.0	1.0	4.5	4.5	20.0	6.0	19.0	8.0	5.5
IRE.loc.conf	9.91	6.0	13.0	17.0	13.0	2.5	12.5	3.0	8.0	11.0
IRE.loc.lapl	10.12	8.0	19.0	7.0	6.0	2.5	11.0	6.0	16.0	9.0
IRE.glob.conv	10.29	10.0	10.0	8.0	14.0	1.0	12.5	7.0	8.0	8.0
RIPPER	10.53	2.0	6.0	4.5	4.5	7.0	19.0	20.0	8.0	7.0
PART	11.24	13.0	4.0	6.0	7.0	13.0	15.0	8.0	8.0	10.0
ADABoost	11.68	24.0	7.0	16.0	16.0	8.0	2.5	5.0	8.0	4.0
PB.BR.conv	12.47	3.0	8.0	18.0	11.5	12.0	21.0	13.5	8.0	19.0
RuleFit	12.65	7.0	24.0	24.0	24.0	9.0	4.0	4.0	24.0	2.0
BR.conv	12.85	4.0	9.0	9.0	19.0	4.0	23.5	10.5	8.0	13.0
Vtop5.conv	14.00	16.0	11.0	19.0	20.0	10.5	14.0	21.5	18.5	15.0
BR.conf	14.32	12.0	12.0	11.0	18.0	5.5	23.5	10.5	8.0	13.0
Voting.conv	14.35	17.0	16.0	20.0	21.0	10.5	10.0	21.5	18.5	16.0
CMAR	14.74	5.0	23.0	21.0	8.0	18.0	18.0	15.0	8.0	22.0
BR.lapl	14.79	15.0	22.0	11.0	10.0	5.5	22.0	9.0	8.0	13.0
PB.BR.conf	14.97	11.0	14.0	13.0	11.5	17.0	20.0	13.5	8.0	17.5
Vtop5.conf	15.24	22.5	15.0	14.0	17.0	23.0	7.0	17.0	8.0	20.5
PB.BR.lapl	16.15	14.0	21.0	11.0	9.0	19.0	16.5	12.0	17.0	17.5
Vtop5.lapl	17.15	22.5	20.0	15.0	15.0	24.0	16.5	18.0	20.0	20.5
Voting.conf	17.71	21.0	17.0	22.5	22.5	22.0	8.5	23.0	22.0	23.5
Voting.lapl	17.82	19.0	18.0	22.5	22.5	21.0	8.5	24.0	23.0	23.5

First column shows mean rank of the algorithm in the column. The lines are sorted by mean rank

Notable exceptions are:

- IRE.glob.conv significantly outranks all Vtop5 and Voting strategies.
- W.r.t. the *PB* strategies, *AdaBoost* and *Bagging* are not shown to be significantly better than any *PB* strategy.
- The best *IRE* approach ranks lower than *J*48 but higher than *AdaBoost*.

7.3 Summary of results

The above results show that some jittering ensembling techniques are able to improve the predictive accuracy of single model approaches with the sames measure and even compete with state of the art approaches. It is not expectable that model jittering approaches have the same level of predictive performance as data jittering approaches, such as bagging and boosting, since the latter invest K times more learning effort, being K the number of iterations. Nevertheless, we proved our point that model jittering en-

sembling improves single model approaches in *BestRule* association rule classification. It is also interesting to notice the advantage of *BestRule* approaches with respect to weighted *Voting* (rule ensembles), either unrestricted or considering top five rules per class. Similar results, although with fewer data sets, can be found in Azevedo and Jorge (2007a). It is also interesting to note the combined effect of *IRE* with conviction as compared to *BestRule* with confidence.

Despite the weak results of *Voting*, the success of the methods *MLRules* and *SLIPPER* (with advantage for the latter on binary data sets) indicate that ensembles of rules can be a powerful predictive tool. *RuleFit* had less successful results.

In the following sections we will look into the error decomposition and try to determine the specific effect of each of the jittering ensembling techniques on the error components. We will also look at the behavior of these methods for the hard cases in a data set. Finally we will perform a study on the computational properties of *IRE* and *PB*.

7.4 Bias–variance analysis

To understand why *IRE* and *PB* improves the results of a *BestRule* classifier we have performed a bias–variance analysis as described by Kohavi and Wolpert (1996). For each data set we proceed as follows. We divide the examples in two sets D and E. This last set is used for evaluation and is a stratified sample, without replacement, with half the size of the original data set. From the set D we generate 50 simple random samples, without replacement, with half the size of D. Each one of these samples is used as training, and the results of the obtained models on E are used to estimate the contribution of the bias and of the variance to the global error. Numerical attributes of D are discretized as already described before generating the 50 samples. For each data set we decompose the error into bias and variance for *BestRule*, *IterativeReordering* and *PostBagging*. The parameters used were the same as in the experiments reported above.

Figure 1 shows the results of the bias–variance analysis for 30 of the 36 data sets and for 3 strategies: *BestRule*, IRE.loc and PB.BR. For all the strategies we have used the confidence interest measure. Each pie chart shows the proportion of data sets where bias or variance was reduced by one strategy with respect to another.

The two pie charts on the column "IRE<BR" show that the strategy *IterativeReordering* reduces both bias and variance with respect to the *BestRule* approach, but mostly bias. The middle column shows that, with respect to *BestRule*, *PostBagging* reduces mostly variance, whereas bias reduction is almost in a tie situation (16 out of 30 data sets). This is confirmed by the two pie charts on the right, with *IterativeReordering* more successful than *PostBagging* in the bias component, but less successful in the variance component.

We can thus hypothesize that the error reduction caused by *IRE* is mainly due to the reduction of the bias component, whereas the success of *PostBagging* is due only to the reduction in variance. Since the variance component will converge to zero with the size of the data sets, one can expect that *IRE* will be advantageous for large data sets. Indeed, in our experiments, we observe that *IRE* wins over *PostBagging* for the

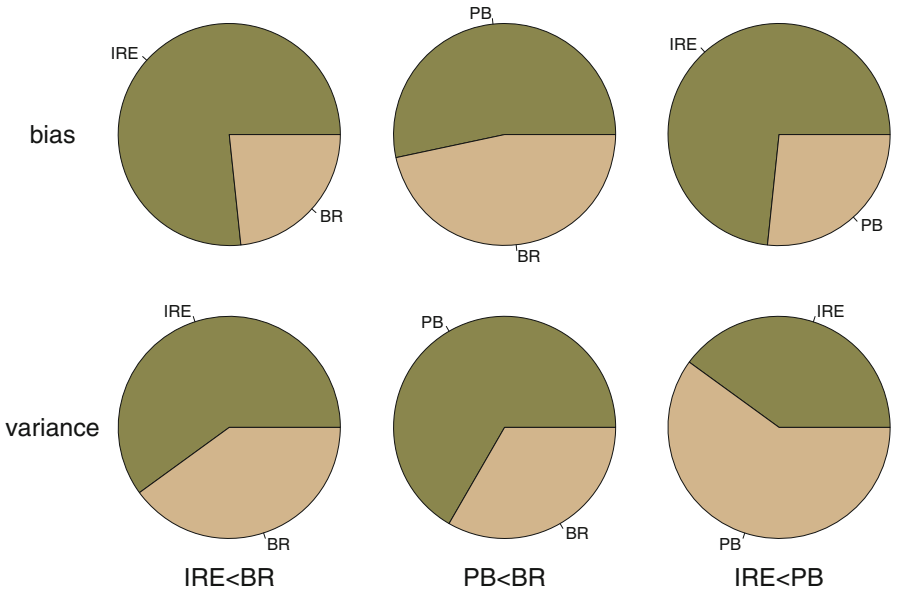

Fig. 1 Each pie chart on line "bias" and column "S1 <S2" displays the proportion of data sets where strategy S1 reduced the bias component of the error with respect to the strategy S2. Similarly for line "variance"

two largest data sets ("sat" and "adult48") (Tables 10 and 13). We should also note that dealing with large data sets is not particularly complicated since the time taken for the generation of association rules grows linearly with the number of examples (Agrawal et al. 1996). However, computational time for association rule generation is very sensitive to data set density and minimal support.

7.5 Success border performance

In another set of experiments, we have observed how the answers of the models in the ensemble compare with the answers given by the single model. In Fig. 2 we can see the result for the *yeast* data set. The x axis represents the test examples and the y axis the percentage of correct answers given by the two strategies for each case. In the case of the *BestRule*, this percentage is either 0 (failure) or 1 (success). In the case of the ensemble approach we have the percentage of models in the ensemble that gave the correct answer. The examples in the x axis are sorted by the success of the *BestRule* and than by the percentage of successes of the ensemble.

With this analysis we can see that there is a good number of "easy cases" and of "hard cases". These are the ones at the right and left end of the plot, respectively. The cases in the middle are in a grey area. These are the ones that can be more easily recovered by IRE. To be successful, the IRE approach must recover more examples (improve the answer of the *BestRule*) than the ones it loses (degrades the answer of the *BestRule*). In the case of *yeast*, we can see that many cases are recovered (crosses above the 0.5 horizontal line, and to the left of the vertical solid line), although some

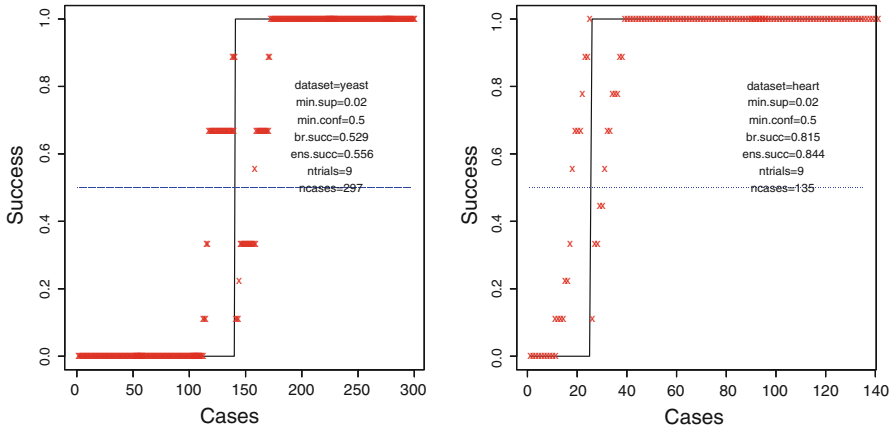

Fig. 2 *Left* Percentage of correct answers per test case for the *yeast* data set (*crosses*) shown against the correct answers given by the *BestRule* approach (*solid crisp line*). *Right* Similar analysis for the *heart* data set

others are lost (below the horizontal line and to the right of the vertical one). In the case of the *heart* data set (Fig. 2), similar observations can be made. Notice the small number of test examples that perform worse in the ensemble method than in the *BestRule* prediction. Such figures provide a fairly good idea of the lower limit of the jittering strategy, since only the cases on the grey area are expectable to be recovered. The figures clearly show that most of the cases, for both datasets, have a unanimous voting from all the models in the ensemble.

7.6 Computational effort

One important motivation for developing model jittering ensembling is to assess how much we can improve single models by looking at them from different perspectives after performing one single learning step. Another motivation is to reduce the learning effort of traditional ensemble approaches. This is particularly important if the effort of learning a single model is relatively high, as it is the case of association rule based classification. If a model is very fast to learn, then traditional ensembling should be the way to go.

In general, we can consider that the effort of learning a model (for a fixed number of examples) is L times the effort of applying the model (evaluation). If we assume that evaluation time is equal to 1, then learning K models for an ensemble takes $K \times L$, without loss of generality. The time for generating replicate models in PB and in IRE is dominated by the time of evaluation of the examples in the validation set. If the evaluation time is 1 then the time taken by PB or IRE to generate K models is K. Therefore, the estimated speed-up of model jittering, a process where you learn once and evaluate K times, with respect to traditional ensemble learning, is

$$SpeedUp_{MJ} = \frac{L \times K}{L + K} \tag{14}$$

Table 6 Time analysis: data set and rule set characterization

Data set	$\|R\|$	N	$\|R_{PB}\|$	$\|R_{IRE}\|$
Aust	184	621	90	40
Segm	2955	2079	1478	213
Led7	217	2880	108	44
Mush	3683	7311	1841	33
Shut	415	52200	207	146
Vehicle1–3	420	255	210	39
Vehicle1–6	2488	510	1244	121
Vehicle1–10	3934	846	1967	180

Table 7 Time analysis: learning, generation and evaluation real times (s)

Data set	$Learn_R$	Gen_{PB}	Gen_{IRE}	Ev_{PB}	Ev_{IRE}	Ev_{BR}
Aust	10.029	0.520	0.520	0.230	0.230	0.244
Segm	15.480	4.030	1.800	0.692	0.457	0.338
Led7	3.147	1.866	1.533	0.522	0.470	0.302
Mush	24.988	35.739	6.861	4.310	2.288	1.013
Shut	24.970	19.269	31.825	3.976	3.050	2.835
Vehicle1–3	2.404	0.456	0.345	0.175	0.230	0.191
Vehicle1–6	36.028	2.363	0.621	0.383	0.284	0.260
Vehicle1–10	64.220	4.263	1.068	0.755	0.446	0.347

It is expected that this generic law also depends on the size of the data set (N), and on the number of rules generated and selected for the ensemble models. Nevertheless, we will in the following experimentally test it.

In Table 6 we describe the data sets we have used to analyze the speed-ups of MJE. There we can see for each of the data sets (of varying sizes N) the number of rules generated (R), and the average number of rules of the models generated by PB and IRE. These experiments were performed using the whole data sets, for $K = 10$. In PB we have first generated 100 models with 50% of the rules and then selected the top 10 models. The data sets are already known, except for the "vehicle$\{i–j\}$". These were obtained by concatenating folds i to j of a tenfold partition of "vehicle".

Table 7 presents the time, in seconds, for generating the whole rule set $Learn_R$, and for generating (Gen) and evaluating (Ev) the jittered models. These are real times and include the reading of the examples from a file. Ev times exclude the time for loading the model. Gen times include the evaluation of the examples in the validation set.

In Table 8 we have the observed learning factors L for PB, IRE and plain *BestRule* approach, i.e., the quotient between learning time and the respective evaluation time for the same examples. The learning factors for PB and IRE are very high w.r.t. the factor of *BestRule* L_{BR}. This is because evaluation in PB and IRE is optimized and is not proportional to the number of models K. What happens is that all the models have rules from the same pool. Previously to evaluating one example on the K models,

Table 8 Time analysis: learning factors and speed-ups

Data set	L_{PB}	L_{IRE}	L_{BR}	SU_{PB}	SU_{IRE}	SU_{Est}
Aust	436	436	41	9.5	9.5	8.0
Segm	224	339	46	7.9	9.0	8.2
Led7	60	67	10	6.3	6.7	5.1
Mush	58	109	25	4.1	7.8	7.1
Shut	63	82	9	5.6	4.4	4.7
Vehicle1–3	137	105	13	8.4	8.7	5.6
Vehicle1–6	941	1269	139	9.4	9.8	9.3
Vehicle1–10	851	1440	185	9.4	9.8	9.5

we identify which rules apply and then only some marginal computations have to be made. Therefore, the actual learning factor we will consider in the following is L_{BR}.

The observed speed-ups for the methods PB and IRE, SU_{meth}, are obtained by dividing $K \times Learn_R$ by $Learn_R + Gen_{meth}$. The estimated speed-ups SU_{Est} are obtained with Eq. 14. As we can observe, there is a reasonable agreement between the expected value and the assessed ones, which is in favour of our analysis. Notice that our speed-up estimator does not consider any of the features of the data set. It is also worth noticing that the efficiency of PB and IRE w.r.t. *BestRule*, measured as SU/K is also very high, becoming 90% or higher for the data sets "Australian", "segment", "vehicle1-6" and "vehicle1-10". In the case of the "vehicle" data sets we also observe an increase of efficiency for the larger samples.

7.6.1 Complexity

Learning (generating the association rules) from N examples is at least $O(N)$ (assuming fixed parameters) (Agrawal et al. 1996). Learning K models is necessarily $O(K\ N)$. The *PostBagging* process can be divided in the sampling phase, where the rules are sampled, and the validation phase, where the models are evaluated and the top performing ones are selected. The overall complexity of *PostBagging*, in the sampling phase, grows at most linearly with the number of models K, and also (when actually sampling) with the number of rules of the initial rule set $|R|$ and the size of the generated models $|M|$. So, the sampling phase has order $O(K\ |R|\ |M|)$. In the validation phase each one of the K models must be evaluated on each of the N examples. The cost of evaluating one example is approximately $O(log(|M|)$, since the rules are stored in an efficient hash-like structure (Azevedo 2005). So the overall effort for validation in *PostBagging* is $O(K\ N\ log(|M|))$.

For *IterativeReordering* we have $O(K\ N\ log(|M|))$ for validation, since the effort will be dominated by the evaluation of examples. Therefore the speed-up should be similar to what we have in post-bagging assuming that the size of the models is equivalent. In practice this depends on parameters and other factors such as the number of rules of the models obtained by jittering.

Table 9 Time analysis: comparison of IRE with other methods

	Adult48		Shuttle		Wave		Mushrooms	
	Time	Error	Time	Error	Time	Error	Time	Error
AdaBoost	65.48	0.1616	11.38	0.0014	3.77	0.1818	0.03	0.0000
Bagging	35.80	0.1384	11.67	0.0035	2.72	0.1764	0.13	0.0000
IRE.cv	14.68	0.1487	10.07	0.0010	3.30	0.1778	4.23	0.0000
J48	3.78	0.1390	1.47	0.0031	0.32	0.2402	0.02	0.0000
MLRules	24h+	0.1426	21.12	0.0017	8.72	0.1616	15.47	0.0000
SLIPPER	26.17	0.1497					2.03	0.0059

Time is in minutes (decimal), with the exception of MLRules in column 1, which is in hours

7.6.2 Comparison with other methods

We have also compared the time taken by the best *IRE* with the most competitive methods in our study (Table 9) on two large data sets ("adult48" and "shuttle" with about 50K rows) and two of medium size ("mushroom" with about 8K and "wave" with about 5K). These are real times, assessed as averages over tenfold cross validation. *IRE* is faster than *AdaBoost* on *J*48 in the two large ones and one of the medium ones. These better times are obtained without sacrificing the error. The "mushroom" data set is a relatively difficult case for *IRE*, in terms of computational efficiency. The comparison with *Bagging* on *J*48 is not as favourable to *IRE*. Our approach is faster twice, but in the case of "adult48" at the cost of error. *MLRules* is relatively slow, but has a good predictive performance for these data sets. *SLIPPER* is also relatively slow (results are shown only for the two of these data sets which are binary), without profiting in terms of predictive accuracy. *J*48 is relatively fast at the cost of some accuracy.

8 Discussion

We have seen that the two techniques proposed for jittering models of association rules improve the predictive accuracy of the respective single models. Jittering a model produces similar copies of the model which will behave mostly as the original one, but will differ on harder cases, where some improvement may occur (see Sect. 7.5). Both *IterativeReordering* and *PostBagging* tend to reduce the variance component of the error. This is expectable due to the ensembling effect of both strategies. In the case of *IterativeReordering*, model jittering is even more effective in reducing the bias component of the error. This is more surprising, but is similar to what happens in boosting and in contrast to the case of bagging, where the reduction of the error is mainly due to the reduction in variance (Bauer and Kohavi 1999).

The intuitive explanation for the reduction of the bias component is that the single model *BestRule* approach is tied to a particular rule ordering, and it is hard to find an ordering that maximizes the number of examples correctly classified. This constraint

seems to be softened by combining similar versions of the rule set with different orderings. This is basically what Iterative Reordering delivers. IRE is able to avoid some limitations of the hypothesis language and gain more descriptive power.

How generalizable is model jittering ensembling to other machine learning techniques such as decision trees or nearest neighbor? Clearly this remains an open question. Association rule models are easily jittered due to the high modularity and granularity of the rule sets. There is also a high level of redundancy, despite the filters used (such as Fisher's), which is convenient if the jittering strategy involves deleting part of the model (it is the case of *PostBagging*, and to some extent also of *IRE*). In other words, *MJE* couples quite well with the "LeGo" framework of learning global models from local patterns (Knobbe et al. 2008), since such global models are easy to jitter. One could also devise jittering operators for other model families with some structure, such as decision trees. One possibility would be to generate ensembles of randomly pruned trees. In the case of nearest neighbor classifiers, jittering the model is almost equivalent to jittering the data.

Other proposals using randomization in model construction are described in the literature, such as "Millions of random rules" (Pfahringer et al. 2004) and "Random Forests" (Breiman 2000). Model Jittering, however, can also be performed without randomization, as it is the case of the *IRE* approach.

In terms of efficiency, our experiments try to answer the question of how interesting is it to build ensembles with one learning process only. Model Jittering is able to speed up efficiently a single model association rule learner and obtain learning times comparable (for some data sets) to competing methods and ensemble approaches such as *ADABoost* and *Bagging*. Taking into consideration that we are also able to obtain good predictive accuracies, we can claim that Model Jittering Ensembling is at least an interesting research path to follow, and a potentially useful tool right now.

One interesting question is how would *IRE* and *PB* behave if we had a smaller validation set, instead of using the training set as we did. The methods would be much faster, but the errors could increase. On the other hand, using a separate validation set for assessing bags and reordering rules could have a positive effect on the reduction of potential overfitting.

Methods based on ensembles of rules, such as *SLIPPER*, and *MLRules*, have very good predictive results, better than *IRE*, although not significantly. These methods also have the advantage of producing a single rule set which is potentially more readable than ensembles of rule sets. *RuleFit* is another method based on rules ensembles. However, the experimental results we have obtained with it are not better than the ones of *IRE*. Interestingly, *SLIPPER* and *MLRules* are relatively slow, at least on some data sets. We had disappointing results with our implementation of *CMAR* and were unable to confirm the virtues of its strategy.

9 Conclusions

Classification using association rules can be improved through ensembling. We have introduced Model Jittering Ensembling for association rule classifiers. In particular, the strategies of *PostBagging* and *IterativeReordering* are two instances of this general

proposal which produces ensembles running one single learning process. *PostBagging* (PB) generates multiple models by repeatedly sampling the rule set. In our case, we also make a selection of the best *PostBagging* generated models. *IterativeReorderingEnsembling* (IRE), is a procedure that generates multiple models with one single learning step. First, a rule set is obtained from the data. Then, replications of this initial set are obtained by iteratively recalculating the predictive measures of the rules in the set. As far as we know, our proposal is the only ensembling approach that produces multiple models by perturbing an original one. We have formalized the Model Jittering general algorithm (for association rules) and the two presented Model Jittering operators.

Experimental results with 36 data sets suggest that Model Jittering Ensembling improves *BestRule* prediction and is competitive when compared to state of the art methods such as *J*48, *PART*, *RIPPER* and *RuleFit*. *IRE* is not significantly worse than Bagging or Boosting *J*48. The comparison of *PostBagging* with the other methods is not as favourable but still indicates that *PB* is not significantly worse than its main competitors with the measure conviction. Bias–variance decomposition indicates that most of the improvement achieved by *IRE* is explained by a reduction of the bias component. This is possibly explained by the ability of the ensembling technique avoiding being tied to one particular ordering of the rules. *PostBagging* mainly reduces the variance component of the error.

Rule ensemble methods *SLIPPER* and *MLRules* obtained very good accuracies (the latter was only applied on two-class data sets). As a side result we observed a tendency of *SLIPPER* to outrank *MLRules*.

One potential advantage of Model Jittering Ensembling is that it obtains multiple models by perturbing an original one. After producing the original model (set of rules), producing the jittered models is relatively inexpensive (depending on the jittering operator). Despite that low variety, their combination results in an effective improvement with respect to the single model. We have shown that Model Jittering obtains good speed-ups and has, on some hard datasets, times comparable to its competitors. In this respect, the current implementation of *IRE* is much faster than the one of *MLRules*. However, methods such as *MLRules* and *SLIPPER* have the important advantage of generating smaller (more readable) models than Model Jittering Ensembling.

Our approach shows some difficulties on some of the dense datasets. These difficulties are in the generation of association rules itself, and not in the ensembling strategies. The choice of parameters of *IRE* and *PB* is still not entirely justified, either empirically or theoretically. This issue certainly deserves a deeper study.

It would be also valuable to investigate more thoroughly the relationship between *IRE* and *ADABoost*, and how much rule reweighting is a proxy for example reweighting.

Many paths can be explored using this simple idea of Model Jittering. Applying Model Jittering Ensembling to association rules can be further exploited. Generalizing the model jittering approach to other models such as decision trees or Bayesian Networks is also an open issue.

Acknowledgments This work was partially supported by FCT project Rank! (PTDC/EIA/ 81178/2006) and by AdI project Palco3.0 financed by QREN and Fundo Europeu de Desenvolvimento Regional (FEDER), and also supported by Fundação Ciência e Tecnologia, FEDER e Programa de Financiamento Plurianual de Unidades de I & D. Thanks are due to William Cohen for kindly providing the executable code for the SLIPPER implementation. Our gratitude goes also to our anonymous reviewers who have helped to significantly improve this paper by sharing their knowledge and their informed criticism with the authors.

Appendix

Tables 10 and 11 show the estimated error rates, using tenfold cross validation of each method on each of the binary data sets. Tables 12 and 13 show the estimated error rates, using tenfold cross validation of each method which applies to non-binary data sets on each data set.

Table 10 Average error rates obtained with the algorithms on 9 of the 17 binary data sets

	Adul48	Aust	Brea	Flar	Germ	Hear	Hepa	Hous	Pima
J4.8	0.1390	0.1391	0.0544	0.1764	0.2930	0.2333	0.1613	0.0368	0.2617
Bagging	0.1384	0.1362	0.0415	0.1867	0.2600	0.1889	0.1677	0.0368	0.2591
ADABoost	0.1616	0.1522	0.0429	0.1914	0.2930	0.2407	0.1419	0.0391	0.2760
PART	0.1420	0.1594	0.0615	0.1895	0.2770	0.2444	0.1548	0.0529	0.2552
RIPPER	0.1553	0.1348	0.0472	0.1848	0.2810	0.2259	0.2000	0.0460	0.2230
CMAR	0.1744	0.1464	0.0358	0.2074	0.2580	0.1667	0.2387	0.0877	0.2330
MLRules	0.1349	0.1341	0.0329	0.1886	0.2410	0.1737	0.1677	0.0391	0.2461
SLIPPER	0.1497	0.1434	0.0314	0.1801	0.2160	0.1036	0.0816	0.0322	0.1745
RuleFit	0.1458	0.1406	0.0421	0.1998	0.2424	0.1538	0.3386	0.1240	0.2375
BR.conf	0.1501	0.1696	0.0515	0.1886	0.2840	0.1963	0.1929	0.0646	0.2539
BR.conv	0.1400	0.1725	0.0458	0.2036	0.2720	0.2037	0.1862	0.0714	0.2317
BR.lapl	0.1528	0.1565	0.0643	0.1886	0.2870	0.1889	0.1996	0.0645	0.2565
Voting.conf	0.1637	0.1797	0.0558	0.1848	0.2780	0.1593	0.1742	0.1403	0.2618
Voting.conv	0.1440	0.1797	0.0429	0.1830	0.2800	0.1741	0.1546	0.1013	0.2579
Voting.lapl	0.1638	0.1797	0.0558	0.1858	0.2780	0.1593	0.1742	0.1403	0.2605
Vtop5.conf	0.1711	0.1652	0.0529	0.1820	0.2840	0.1704	0.1871	0.0690	0.2631
Vtop5.conv	0.1406	0.1768	0.0443	0.1839	0.2600	0.1815	0.1679	0.0783	0.2566
Vtop5.lapl	0.1718	0.1667	0.0529	0.1829	0.2880	0.1667	0.1871	0.0690	0.2631
PB.BR.conf	0.1642	0.1493	0.0458	0.1886	0.2940	0.1815	0.2121	0.0576	0.2526
PB.BR.conv	0.1503	0.1478	0.0429	0.2036	0.2790	0.1852	0.1612	0.0530	0.2304
PB.BR.lapl	0.1646	0.1478	0.0572	0.1886	0.2970	0.1815	0.2062	0.0577	0.2552
IRE.loc.conf	0.1480	0.1652	0.0329	0.1886	0.2850	0.1704	0.1600	0.0508	0.2356
IRE.glob.conv	0.1473	0.1580	0.0372	0.2046	0.2720	0.1741	0.1742	0.0599	0.2486
IRE.loc.lapl	0.1443	0.1478	0.0315	0.1886	0.2830	0.1741	0.1988	0.0531	0.2435

BR BestRule, *IRE* IterativeReordering, *PB* PostBagging, *conf* confidence, *conv* conviction, *lapl* Laplace

Table 11 Average error rates obtained with the algorithms on the other 8 of the 17 binary data sets

	Hypo	Hors	crx	tic-	Sona	Iono	Mush	kr–kp
J4.8	0.0070	0.1467	0.1420	0.1639	0.2115	0.1083	0.0000	0.0563
Bagging	0.0073	0.1440	0.1333	0.0835	0.1971	0.1026	0.0000	0.0563
ADABoost	0.0104	0.1957	0.1710	0.0282	0.1923	0.0741	0.0000	0.0501
PART	0.0085	0.1495	0.1478	0.0689	0.2452	0.0855	0.0000	0.0939
RIPPER	0.0092	0.1467	0.1420	0.0219	0.2644	0.1254	0.0000	0.0720
CMAR	0.0389	0.2364	0.1493	0.1190	0.2560	0.0969	0.0000	0.1868
MLRules	0.0089	0.1386	0.1348	0.0710	0.1923	0.0655	0.0000	0.0100
SLIPPER	0.0082	0.1248	0.1246	0.1075	0.1102	0.0342	0.0059	0.0322
RuleFit	0.1005	0.3801	0.2676	0.0318	0.1929	0.0734	0.1330	0.0229
BR.conf	0.0171	0.1822	0.1783	0.0188	0.2845	0.0912	0.0000	0.0957
BR.conv	0.0139	0.1797	0.1870	0.0178	0.2845	0.0912	0.0000	0.0957
BR.lapl	0.0348	0.1822	0.1522	0.0188	0.2798	0.0855	0.0000	0.0957
Voting.conf	0.0209	0.2583	0.2043	0.1847	0.2274	0.2279	0.0180	0.4255
Voting.conv	0.0193	0.2123	0.2000	0.0345	0.2312	0.1711	0.0009	0.1058
Voting.lapl	0.0212	0.2583	0.2043	0.1837	0.2274	0.2307	0.0183	0.4255
Vtop5.conf	0.0183	0.1904	0.1739	0.2035	0.2267	0.1053	0.0000	0.1514
Vtop5.conv	0.0158	0.2014	0.1884	0.0345	0.2407	0.1711	0.0009	0.0970
Vtop5.lapl	0.0319	0.1931	0.1696	0.2077	0.2557	0.1083	0.0010	0.1514
PB.BR.conf	0.0183	0.1849	0.1580	0.1117	0.2652	0.0968	0.0000	0.1083
PB.BR.conv	0.0133	0.2013	0.1580	0.0512	0.2702	0.0968	0.0000	0.1101
PB.BR.lapl	0.0345	0.1822	0.1507	0.1211	0.2557	0.0940	0.0006	0.1083
IRE.loc.conf	0.0174	0.1957	0.1594	0.0094	0.2367	0.0714	0.0000	0.0942
IRE.glob.conv	0.0142	0.1796	0.1623	0.0094	0.2367	0.0743	0.0000	0.0857
IRE.loc.lapl	0.0231	0.1740	0.1449	0.0094	0.2362	0.0742	0.0004	0.0923

Table 12 Average error rates obtained with the algorithms on 9 of the 19 non binary data sets

	Clev	Lymp	Segm	Shut	Vehi	Wave	Yeas	Glas	Wine
J4.8	0.4719	0.2365	0.0325	0.0031	0.2742	0.2402	0.4414	0.3411	0.0562
Bagging	0.4389	0.2365	0.0255	0.0034	0.2340	0.1764	0.3922	0.2430	0.0281
ADABoost	0.4554	0.2500	0.0160	0.0014	0.2400	0.1818	0.4239	0.2523	0.0281
PART	0.5215	0.1824	0.0372	0.0022	0.2849	0.2142	0.4454	0.3037	0.0899
RIPPER	0.4686	0.2230	0.0463	0.0036	0.3144	0.2018	0.4191	0.3318	0.0787
CMAR	0.4324	0.1967	0.1121	0.0415	0.3700	0.1840	0.4522	0.3561	0.0735
MLRules	0.4224	0.1757	0.0299	0.0026	0.2447	0.1510	0.4043	0.2804	0.0393
BR.conf	0.4655	0.1557	0.0840	0.0019	0.4043	0.1868	0.4448	0.3093	0.0791
BR.conv	0.4256	0.1424	0.0840	0.0018	0.4043	0.1882	0.4414	0.3279	0.0791
BR.lapl	0.4622	0.1557	0.1069	0.0046	0.4125	0.1864	0.4455	0.3136	0.0791
Voting.conf	0.4291	0.2300	0.0823	0.0124	0.3569	0.1756	0.4515	0.3374	0.2369

Table 12 continued

	Clev	Lymp	Segm	Shut	Vehi	Wave	Yeas	Glas	Wine
Voting.conv	0.4190	0.2562	0.2143	0.0034	0.3581	0.1628	0.4461	0.3465	0.1239
Voting.lapl	0.4291	0.2367	0.0831	0.0126	0.3581	0.1758	0.4515	0.3422	0.2425
Vtop5.conf	0.4260	0.1352	0.0597	0.0031	0.3747	0.1722	0.4515	0.3513	0.0399
Vtop5.conv	0.4190	0.2429	0.2156	0.0034	0.3996	0.1778	0.4461	0.3461	0.1239
Vtop5.lapl	0.4260	0.1557	0.0861	0.0037	0.3889	0.1730	0.4515	0.3606	0.0624
PB.BR.conf	0.4259	0.1490	0.0649	0.0021	0.3782	0.1740	0.4468	0.3186	0.0402
PB.BR.conv	0.4156	0.1152	0.0649	0.0021	0.3759	0.1742	0.4367	0.3139	0.0402
PB.BR.lapl	0.4291	0.1486	0.0900	0.0038	0.3912	0.1744	0.4475	0.3323	0.0458
IRE.loc.conf	0.4622	0.1495	0.0485	0.0014	0.3262	0.1782	0.4428	0.2864	0.0399
IRE.glob.conv	0.4520	0.1495	0.0494	0.0010	0.3332	0.1778	0.4394	0.2909	0.0399
IRE.loc.lapl	0.4520	0.1429	0.0532	0.0019	0.3404	0.1792	0.4508	0.2948	0.0454

Table 13 Average error rates obtained with the algorithms on 10 of the 19 non-binary data sets

	Zoo	Led7	Iris	Soyb	Sat	Anne	kr.k	Nurse	Page	Pend
J4.8	0.0693	0.2666	0.0400	0.1498	0.1414	0.0791	0.4341	0.0295	0.0312	0.0345
Bagg	0.0693	0.2622	0.0467	0.1140	0.0995	0.0601	0.4026	0.0269	0.0274	0.0220
Boost	0.0495	0.2666	0.0667	0.0879	0.0929	0.0445	0.3874	0.0051	0.0298	0.0099
PART	0.0693	0.2644	0.0600	0.1270	0.1304	0.0546	0.4561	0.0079	0.0300	0.0318
RIPP	0.1287	0.3075	0.0600	0.1107	0.1301	0.0457	0.6063	0.0302	0.0276	0.0380
CMR	0.0773	0.2816	0.0667	0.2184	0.2345	0.0791	0.4278	0.0706	0.0577	0.1155
MLR	0.0594	0.2672	0.0467	0.1564	0.1167	0.0167	0.6445	0.0263	0.0280	0.0428
BRcf	0.0991	0.2800	0.0533	0.2735	0.1823	0.0902	0.4232	0.0366	0.0470	0.1034
BRcv	0.0991	0.2803	0.0533	0.2735	0.1804	0.0557	0.4165	0.0364	0.0393	0.1034
BRlp	0.1973	0.2794	0.0533	0.3159	0.1820	0.1047	0.4183	0.0403	0.0733	0.1156
V.cf	0.0782	0.2875	0.0733	0.3743	0.3189	0.1459	0.4386	0.3723	0.0459	0.1109
V.cv	0.1373	0.2841	0.0600	0.3161	0.2188	0.0780	0.3932	0.0291	0.0400	0.2572
V.lp	0.0882	0.2875	0.0733	0.3808	0.3190	0.1493	0.4408	0.3724	0.0464	0.1124
V5.cf	0.0391	0.2753	0.0733	0.2444	0.1970	0.1549	0.4386	0.3559	0.0521	0.0630
V5.cv	0.1373	0.2734	0.0600	0.3162	0.1845	0.0691	0.3931	0.0309	0.0404	0.2572
V5.lp	0.1182	0.2753	0.0733	0.2930	0.1963	0.1593	0.4408	0.3577	0.0623	0.0876
Bg.cf	0.0582	0.2744	0.0533	0.2474	0.1812	0.1404	0.4783	0.0551	0.0512	0.0809
Bg.cv	0.0582	0.2753	0.0533	0.2474	0.1776	0.1115	0.4664	0.0528	0.0424	0.0809
Bg.lp	0.1082	0.2747	0.0533	0.2961	0.1813	0.1493	0.4783	0.0564	0.0696	0.1039
IR.cf	0.0591	0.2625	0.0600	0.1299	0.1546	0.0902	0.4384	0.2737	0.0453	0.0515
IR.cv	0.0591	0.2644	0.0600	0.1267	0.1538	0.0557	0.4253	0.0264	0.0397	0.0513
IR.lp	0.0891	0.2631	0.0600	0.1432	0.1565	0.1058	0.4280	0.2541	0.0608	0.0529

Methods names are abbreviated

References

Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo AI (1996) Fast discovery of association rules. In: Advances in knowledge discovery and data mining. AAAI/MIT Press, pp 307–328

Azevedo PJ (2003) CAREN—A Java based apriori implementation for classification purposes. Technical report, Universidade do Minho, Departamento de Informática

Azevedo PJ (2005) A data structure to represent association rules based classifiers. Technical report, Universidade do Minho, Departamento de Informática

Azevedo PJ (2008) CAREN—class project association rule engine. http://www.di.uminho.pt/~pja/class/caren.html

Azevedo PJ, Jorge AM (2007a) Comparing rule measures for predictive association rules. In: Kok JN, Koronacki J, de Mántaras RL, Matwin S, Mladenic D, Skowron A (eds) ECML. Lecture Notes in Computer Science, vol 4701. Springer, pp 510–517

Azevedo PJ, Jorge AM (2007b) Iterative reordering of rules for building ensembles without relearning. In: Corruble V, Takeda M, Suzuki E (eds) Discovery science. Lecture Notes in Computer Science, vol 4755. Springer, pp 56–67

Bauer E, Kohavi R (1999) An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach Learn 36(1–2):105–139

Bayardo RJ, Agrawal R, Gunopulos D (1999) Constraint-based rule mining in large, dense databases. In: ICDE. IEEE Computer Society, pp 188–197

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140

Breiman L (2000) Randomizing outputs to increase prediction accuracy. Mach Learn 40(3):229–242

Brin S, Motwani R, Ullman JD, Tsur S (1997) Dynamic itemset counting and implication rules for market basket data. In: Peckham J (ed) SIGMOD conference. ACM Press, pp 255–264

Cohen WW (1995) Fast effective rule induction. In: Machine learning. Proceedings of the twelfth international conference on machine learning (ICML 1995), Tahoe City, CA, USA, 9–12 July, 1995, pp 115–123

Cohen WW, Singer Y (1999) A simple, fast, and effictive rule learner. In: AAAI/IAAI, pp 335–342

Dembczyński K, Kotlowski W, Slowinski R (2008) Maximum likelihood rule ensembles. In: Cohen WW, McCallum A, Roweis ST (eds) ICML. In: ACM international conference proceeding series, vol 307. ACM, pp 224–231

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Domingos P (1997) Why does bagging work? A bayesian account and its implications. In: KDD, pp 155–158

Fayyad UM, Irani KB (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI, pp 1022–1029

Frank E, Pfahringer B (2006) Improving on bagging with input smearing. In: Ng WK, Kitsuregawa M, Li J, Chang K (eds) PAKDD. Lecture Notes in Computer Science, vol 3918. Springer, pp 97–106

Frank E, Witten IH (1998) Generating accurate rule sets without global optimization. In: Shavlik JW (ed) ICML. Morgan Kaufmann, San Fransisco, MA, pp 144–151

Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In: Vitányi PMB (ed) EuroCOLT. Lecture Notes in Computer Science, vol 904. Springer, pp 23–37

Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: ICML, pp 148–156

Friedman JH, Popescu B (2005) Predictive learning via rule ensembles. Technical report, Stanford University

Gama J (2003) Iterative bayes. Theor Comput Sci 292(2):417–430

Hastie T, Tibshirani R, Friedman JH (2001) The elements of statistical learning. Springer, New York

Jorge A, Azevedo PJ (2005) An experiment with association rules and classification: post-bagging and conviction. In: Hoffmann AG, Motoda H, Scheffer T (eds) Discovery science. Lecture Notes in Computer Science, vol 3735. Springer, pp 137–149

Jovanoski V, Lavrac N (2001) Classification rule learning with apriori-c. In: Brazdil P, Jorge A (eds) EPIA. Lecture Notes in Computer Science, vol 2258. Springer, pp 44–51

Knobbe A, Crémilleux B, Fürnkranz J, Scholz M (2008) From local patterns to global models: the LeGo approach to data mining. In: From local patterns to global models: proceedings of the ECML PKDD 2008 Workshop

Kohavi R, Wolpert D (1996) Bias plus variance decomposition for zero-one loss functions. In: ICML, pp 275–283

Li W, Han J, Pei J (2001) CMAR: accurate and efficient classification based on multiple class-association rules. In: Cercone N, Lin TY, Wu X (eds) ICDM. IEEE Computer Society, Washington, DC, pp 369–376

Liu B, Hsu W, Ma Y (1998) Integrating classification and association rule mining. In: KDD '98: proceedings of the fourth ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 80–86

Martínez-Muñoz G, Suárez A (2006) Pruning in ordered bagging ensembles. In: Cohen WW, Moore A (eds) Machine learning. In: Proceedings of the twenty-third international conference (ICML 2006), Pittsburgh, PA, USA, 25–29 June, 2006. ACM international conference proceeding series, vol 148. ACM, pp 609–616

Meretakis D, Wüthrich B (1999) Extending naïve bayes classifiers using long itemsets. In: KDD. pp 165–174

Merz CJ, Murphy P (1996) UCI repository of machine learning database. http://www.cs.uci.edu/~mlearn

Pfahringer B, Holmes G, Wang C (2004) Millions of random rules. In: Workshop on advances in inductive rule learning, 15th European conference on machine learning (ECML), Pisa

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Fransisco, MA

R Development Core Team (2004) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN:3-900051-00-3

Rückert U, Kramer S (2006) A statistical approach to rule learning. In: Cohen WW, Moore A (eds) Machine learning. In: Proceedings of the twenty-third international conference (ICML 2006), Pittsburgh, PA, USA, 25–29 June, 2006. ACM international conference proceeding series, vol 148. ACM, pp 785–792

Schapire RE (1990) The strength of weak learnability. Mach Learn 5:197–227

Wang J, Karypis G (2005) Harmony: efficiently mining the best rules for classification. In: Proceedings of 2005 SIAM international data mining conference, SDM

Webb GI (2006) Discovering significant rules. In: Eliassi-Rad T, Ungar LH, Craven M, Gunopulos D (eds) Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining, Philadelphia, PA, USA, 20–23 August, 2006. ACM, pp 434–443

Webb GI (2008) Layered critical values: a powerful direct-adjustment approach to discovering significant patterns. Mach Learn 71(2–3):307–323

Webb GI, Butler SM, Newlands DA (2003) On detecting differences between groups. In: Getoor L, Senator TE, Domingos P, Faloutsos C (eds) Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, Washington, DC, USA, 24–27 August, 2003. ACM, pp 256–265

Weiss SM, Indurkhya N (2000) Lightweight rule induction. In: Langley P (ed) ICML. Morgan Kaufmann, San Fransisco, MA, pp 1135–1142

Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques. Morgan Kaufmann Series in Data Management Systems. 2. Morgan Kaufmann, San Fransisco, MA

Zimmermann A, Raedt LD (2004) Corclass: correlated association rule mining for classification. In: Suzuki E, Arikawa S (eds) Discovery science. Lecture Notes in Computer Science, vol 3245. Springer, pp 60–72