Rule Stacking: An approach for compressing an ensemble of rule sets into a single classifier

Technical Report TUD-KE-2010-05

Jan-Nikolas Sulzmann, Johannes Fürnkranz
Technische Universität Darmstadt

Knowledge Engineering Group, Technische Universität Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Knowledge
Engineering

Abstract

In this paper, we present an approach for compressing a rule-based pairwise classifier ensemble into a single rule set that can be directly used for classification. The key idea is to re-encode the training examples using information about which of the original ruler covers the example, and to use them for training a rule-based meta-level classifier. We not only show that this approach is more accurate than using the same classifier at the base level (which could have been expected for such a variant of stacking), but also demonstrate that the resulting meta-level rule set can be straight-forwardly translated back into a rule set at the base level. Our key result is that the rule sets obtained in this way are of comparable complexity to those of the original rule learner, but considerably more accurate.

1 Introduction

Ensemble learning is a mixed blessing. On the one hand, it promises an increase in performance compared to a single classifier, on the other hand, the use of a classifier ensemble has several obvious disadvantages: The first problem is that the predictions of the individual ensemble members must be decoded into a single global prediction. The most common solutions are voting methods, which use the prediction of each base classifier as a (weighted) vote, or the similarity to prediction vectors for the individual classes, as, e.g., used in error correcting output codes [3]. The second problem is that the size of the classifier ensemble is a bottleneck for the overall prediction time since, independent of the employed decoding method, several or all classifiers of the ensembles are needed for the prediction. Finally, the resulting prediction is also harder to explain and justify, which is particularly crucial for rule-based classifiers. While it can be argued that a single rule set is comprehensible to experts in the application domain, the predictions of an ensemble of rule sets are much harder to explain and communicate.

Both problems can be solved if the ensemble of classifiers can be transformed or better be compressed into single global classifier. A standard approach for achieving this goal is to train the ensemble classifier, generate additional training examples which are labeled by the trained meta classifier, and use this larger training set for training a comprehensible classifier, which mimicks the performance of the ensemble classifier. This approach has, e.g., been advocated in the MetaCost algorithm, in which bagging is used for deriving improved probability estimates [4].

A different approach is to employ stacking at the meta level [12]. Stacking generates a global classifier by training a meta-level learner for combining the predictions of the base-level classifiers. This classifier is trained on a meta data set, which, in the simplest case, consists of the predictions of the classifier ensemble. This approach only partially solves the above-mentioned problems, because the theory at the meta-level involves the predictions of the base level, which is still a problem for both efficiency and comprehensibility.

In this paper, we propose an approach that generates a single global classifier from an ensemble of rule-based classifiers. This classifier consists of a single rule set, which is composed of rules taken from the base classifiers. The base classifiers themselves can be discarded after training. Essentially, this approach consists of a modified stacking step, in which the meta classifier is trained using the information by which rule an instance is covered (instead of directly using the predictions), and a transformation step that re-transforms the meta classifier into rules that directly apply to the original data set.

In Section 2 we give a short introduction into rule learning and define the functions we need for the generation of the meta data. Thereafter, we describe our approach for compressing an ensemble of classifiers into a global classifier in the original data format in Section 3, focusing on the meta data generation and the compressing step of our approach. In the subsequent Section 5 we describe our experimental setup and results, which are summarized in the conclusions in section 6.

2 Rule Learning

In classification rule mining, one searches for a set of rules that describes the data as accurately as possible. Rule learning algorithms come in many flavors and variants, which differ by their search algorithms and by the expressiveness of the learned rules [6]. In the simplest case, a set of propositional rules is learned from a dataset I which consists of nominal and numeric attributes. In the following, we focus on this scenario, but we note that the approach is also applicable to more expressive types of rules, such as relational rules.

The *premise* of a rule consists of a conjunction of number of conditions, and in our case, the *conclusion* of the rule is a single class value. Thus, a conjunctive classification rule r has the following form:

$$condition_1 \wedge \cdots \wedge condition_{|r|} \implies class \tag{2.1}$$

The *length* of a rule $|r|$ is the number of its conditions. Each of these conditions consists of an *attribute*, a *value* selected from the attribute's domain, and a *comparison operator* determined by the attribute type. For nominal attributes, this comparison is a test of equality, whereas in the case of numerical attributes, the test is either less than (or equal) or greater than (or equal).

If all conditions are met by an instance x, the instance is said to be *covered* by the rule ($r \supseteq x$) and the class value of the rule is predicted for the instance. Consequently, the rule is called a *covering rule* for this instance. If the rules generated by a rule learner are organized and should be used in a specific order, the rule set is called a *decision list*. If an instance is to be classified, the rules of a decision list are tested in the given order and the prediction of the first covering rule is used. We will deal with multiple rule sets or decision lists, which are generated by an ensemble of rule learners C is a set of $|C|$ rule learners.

For a given classifier c its decision list can be written as an ordered set of rules R_c:

$$R_c = \{r_1, \cdots, r_{n_c}\}, \tag{2.2}$$

where r_1 is the first rule of the decision list and r_{n_c} its last rule, typically a default rule that predicts the majority class. The index k of a rule r_k represents its position in the decision list of its corresponding classifier, and n_c denotes the size of the decision list or respectively the number of rules of the classifier c. If necessary, we will use an elevated classifier identifier c for denoting rules from different classifiers (e.g., r_k^c denotes the k-th rule of classifier c). We can thus also define an order $\lhd$ for two rules r_i and r_j of the same classifier c:

$$r_i \lhd r_j, \text{ iff } i \leq j \tag{2.3}$$

Using the order $\lhd$, we get a minimum function $\min_{\lhd}(R)$ for a given rule set $R \subseteq R_c$ which determines the first rule according to the ordering of the decision list.

Given a rule r and an instance i, the function covers(r, i) determines whether the rule covers the instance or not:

$$\text{covers}(r, i) = \begin{cases} \textsf{true}, & \text{if } r \supseteq i \\ \textsf{false}, & \text{otherwise.} \end{cases} \tag{2.4}$$

We can now define a few functions that will be used in the remainder of the paper. First, the set of all covering rules $\text{Cov}(c, i)$ for a given classifier and a given instance i is defined as

$$\text{Cov}(c, i) = \{r \in R_C | \, \text{covers}(r, i)\} \tag{2.5}$$

Given a classifier c and a instance i, the function first(c, i) determines the first covering rule in the decision list of the classifier:

$$\text{first}(c, i) = \min_{\lhd}(\text{Cov}(c, i)) \tag{2.6}$$

Finally, the function index(c, i) determines the index of the first covering rule in decision list for a given classifier and a given instance i:

$$\text{index}(c, i) = \arg\min_{k}(r_k \in \text{Cov}(c, i)) \tag{2.7}$$

3 Rule Stacking

In general, ensemble learning has to take care of two problems. On the one hand, the predictions of the ensemble of classifiers must decoded into a global prediction, e.g. by voting methods. On the other hand, in the test phase, the complete ensemble of classifiers is needed to make a prediction. Both problems can be solved if we can compress the ensemble of classifiers into a single global classifier which can be directly applied to instances in the original data format.

3.1 Stacking

The above-mentioned problems are partly solved by the ensemble method stacking, which generates a global classifier based on the predictions of the ensemble of classifier [12]. However, stacking still needs the predictions of all classifiers of the ensemble. The key idea of stacking is to use these base classifiers as attributes and their predictions respectively as their (attribute) values. In this way the original instances can be transformed into meta instances. Each meta instance consists of the predictions it receives from each classifier. For training the meta classifiers, the meta instances are labeled with the class labels of the corresponding base-level instances, as shown in Figure 3.1. Thus the resulting meta level model is based only on the predictions of the base classifiers. In the testing phase the test instance is transformed into a meta instance by determining the prediction of each base classifier. Afterwards the instance is classified by the prediction of the second level classifier using the meta instance as the input.

Prior work has shown that the simple version of stacking described above does not perform as well as other ensemble techniques, and several improvements have been proposed. Most notably, it has been shown that instead of using the class label at the meta level, it is beneficial augment the meta data set with the confidences of the base level classifiers into their predictions [10]. Subsequently, it was shown that it may be even better to use the entire predicted class probability distribution [9].

Attributes			Class
$x_{1,1}$	$\cdots$	$x_{1,\|A\|}$	+
$x_{2,1}$	$\cdots$	$x_{2,\|A\|}$	-
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$x_{\|C\|,1}$	$\cdots$	$x_{\|C\|,\|A\|}$	+

(a) original data set

c_1	c_2	$\cdots$	$c_{\|C\|}$	Class
+	+	$\cdots$	-	+
-	+	$\cdots$	+	-
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
-	-	$\cdots$	+	+

(b) training set for stacking

Figure 3.1: Illustration of the standard stacking scheme, which uses the predictions of the base classifiers on the original datasets as features of the meta-level dataset.

3.2 Motivation for Rule Stacking

Obviously, all of these variants of stacking do not completely solve the above-mentioned problems of ensemble learning. For this reason, we considered to modify the standard stacking method for a rule learning scenario considering decision list rule learners only. The main idea behind this is to change the representation of the instances at the meta level of a stacking classifier so that it uses the information by which base-level rules the instance is covered as features.

The motivation for this approach is two-fold. First we hope that the covering information yields more information than the prediction alone, which has been shown to lead to better results in the above-mentioned improvements to stacking [10, 9]. Obviously, knowing which rule covers an example is more informative than simply knowing the predicted class value, and implicitly also captures the predicted confidence or class probability distributions, which, in a rule-based classifier, are determined by the quality of the rules that cover an example. Second, as we will see further below, the resulting meta level model can be re-translated into the original data format since it only consists of a conjunction of the original rules.

In the following sections, we will show how the meta data is generated (Section 3.3) and how the resulting meta model can be transformed into a global global classifier which can be directly used on the original instances (Section 3.4). The complete approach is illustrated in Figure 3.3.

3.3 Generating the meta data

The main difference between the standard stacking scheme and our approach is the generation of the meta data. For the base classifiers and the meta classifier we considered only rule learners that generates decision lists. As rule learners provides more information than just its prediction, namely the information which rules cover a given test instance, we want to exploit this covering information. So the attributes of our meta data set are not the predictions of the base learners but either the information which is the first rule of a given base learner that covers the instance or the information if a specific rule covers the instance. This information can, essentially, be encoded at the meta level in three different ways:

Nominal Features: For each base classifier c we create a nominal attribute whose domain consists of the identifiers of its rules $(r_1^c, \cdots, r_{n_c}^c)$ used as nominal values. A meta instance is composed of the respective first covering rule of each classifier and if known its original class value.

Meta instance format: (cf. Figure 3.2 (a))

$$(\text{first}(c_1, i), \cdots, \text{first}(c_{|C|}, i), class)$$

Numerical Features: For each base classifier c we create a numerical attribute whose domain consists of its rule indices $(\{[1, \cdots, n_c\})$ used as numerical values. A meta instance is composed of the indices of the respective first covering rule of each classifier and if known its original class value.

Meta instance format: (cf. Figure 3.2 (b))

$$(\text{index}(c_1, i), \cdots, \text{index}(c_{|C|}, i), (class))$$

Binary Features: For each rule of the classifier ensemble we create a binary attribute with boolean values. The attribute value is true if the corresponding rule covers the instance, else false. Thus, a meta instance is composed of a number of boolean values and if known its original class value.

Meta instance format: (cf. Figure 3.2 (c))

$$(\text{covers}(r_1^{c_1}, i), \cdots, \text{covers}(r_{n_{c_1}}^{c_1}, i), \cdots, \text{covers}(r_{n_{|C|}}^{c_{|C|}}, i), class)$$

c_1	c_2	$\cdots$	$c_{\|C\|}$	Class
$\text{first}(c_1, i_1)$	$\text{first}(c_2, i_1)$	$\cdots$	$\text{first}(c_{\|C\|}, i_1)$	+
$\text{first}(c_1, i_2)$	$\text{first}(c_2, i_2)$	$\cdots$	$\text{first}(c_{\|C\|}, i_2)$	-
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\text{first}(c_1, i_{\|I\|})$	$\text{first}(c_2, i_{\|I\|})$	$\cdots$	$\text{first}(c_{\|C\|}, i_{\|I\|})$	+

(a) using nominal features

$r_1^{c_1}$	$\cdots$	$r_{n_1}^{c_1}$	$\cdots$	$r_{n_{\|C\|}}^{c_{\|C\|}}$	Class
$\text{covers}(r_1^{c_1}, i_1)$	$\cdots$	$\text{covers}(r_{n_1}^{c_1}, i_1)$	$\cdots$	$\text{covers}(r_{n_{\|C\|}}^{c_{\|C\|}}, i_1)$	+
$\text{covers}(r_1^{c_1}, i_2)$	$\cdots$	$\text{covers}(r_{n_1}^{c_1}, i_2)$	$\cdots$	$\text{covers}(r_{n_{\|C\|}}^{c_{\|C\|}}, i_2)$	-
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\text{covers}(r_1^{c_1}, i_{\|I\|})$	$\cdots$	$\text{covers}(r_{n_1}^{c_1}, i_{\|I\|})$	$\cdots$	$\text{covers}(r_{n_{\|C\|}}^{c_{\|C\|}}, i_{\|I\|})$	+

(b) using binary features

c_1	c_2	$\cdots$	$c_{\|C\|}$	Class
$\text{index}(c_1, i_1)$	$\text{index}(c_2, i_1)$	$\cdots$	$\text{index}(c_{\|C\|}, i_1)$	+
$\text{index}(c_1, i_2)$	$\text{index}(c_2, i_2)$	$\cdots$	$\text{index}(c_{\|C\|}, i_2)$	-
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\text{index}(c_1, i_{\|I\|})$	$\text{index}(c_2, i_{\|I\|})$	$\cdots$	$\text{index}(c_{\|C\|}, i_{\|I\|})$	+

(c) using numerical features

Figure 3.2: Illustration of the modified meta data sets:

Note that the resulting meta data set using nominal or numerical features has the same number of attributes as the number of base classifiers, analogous to the standard stacking approach. In the case of binary features, the resulting meta data set has as many attributes as the total number of rules in the classifier ensemble.

Let us illustrate the generation of the meta data set with the help of a toy example. In our experiments we created an ensemble of rule sets via pairwise class binarization (cf. Section 4). One of the employed data set was *zoo* that records the characteristics of animals divided in different classes, e.g. insects or invertebrates. The last classifier of the ensemble tries to distinguish exactly these two animal classes:

ID	Rule
r_1	(airborne = true) $\Longrightarrow$ type=insect
r_2	(predator = false) $\wedge$ (legs $\geq$ 6) $\Longrightarrow$ type=insect
r_3	$\Longrightarrow$ type=invertebrate

Assuming that we want to transform the following instances (only relevant attribute values are shown):

Name	Airborne	Predator	Legs	Type
Termite	false	false	6	insect
Lobster	false	true	6	invertebrate
Crow	true	true	2	bird

in the respective meta data format, we would get the following values for the attributes belonging to the given classifier (only these attribute values are shown):

Name	Nominal Features	Binary Features	Numerical Features
Termite	$(\cdots, r_2, \text{insect})$	$(\cdots, \text{false}, \text{true}, \text{true}, \text{insect})$	$(\cdots, 2, \text{insect})$
Lobster	$(\cdots, r_3, \text{invert.})$	$(\cdots, \text{false}, \text{false}, \text{true}, \text{invert.})$	$(\cdots, 3, \text{invert.})$
Crow	$(\cdots, r_1, \text{bird})$	$(\cdots, \text{true}, \text{false}, \text{true}, \text{bird})$	$(\cdots, 1, \text{bird})$

3.4 Re-transforming the meta classifier

In this section we show how to re-transform the rule sets obtained at the meta level into the original data format so that it can be directly used for classification. This simple idea is the key advantage of our approach which distinguishes it from previous works because we eventually obtain a single classifier that directly operates on the base level, but maintains the accuracy of the meta-level classifier because it is composed of all relevant rules from the rule-based ensemble. As a result, we do not need to store the original ensemble, nor do we need to transform a test instance into the meta format.

Let us first recall the format of the rules we obtain from the meta-level classifier for the various meta-level classifiers described above. In the case of nominal features, the rules encode whether a specific rule is the first covering rule of given classifier ($c_k = r_l$). Similarly, the condition of a binary feature tests whether a specific rule covers an example or not (r_l^c = true/false). In both cases, the truth value of the meta-level condition can be established by only testing the conditions of a single base rule, all other rules of the base classifier c can be obtained.

These facts in mind, we can examine how we can compress an ensemble of rule sets using nominal and binary features. Since we know that the global rule sets consists of conditions which are based on rules of the base classifiers, we can distinguish two cases. In the first case, the global rule consists of only one condition, so we can directly replace the condition of the global rule with the conditions of the base rule. In the second case the global rule consists of more than one condition hence the conditions must be merged. Each global condition corresponds to a test if a base rule covers an example and consequently corresponds to a conjunction of the conditions of the involved base rules. Thus, the global conditions can be merged by concatenating the conjunctions of the conditions of their corresponding base rules. This approach is illustrated in Figure 3.3, step 4 to 5.

For numerical features, a conjunction of meta-level features may eventually denote whether the first covering rule lies in a given interval of the base rules (determined by an interval of indices), i.e., the conditions can be of the form $r_k \leq c_l \leq r_m$. Obviously, in this case we must test a group of base rules, and therefore the resulting global rule set could be more compact than in the nominal or binary case. On the other hand, a back transformation of one condition of this type does not result in a simple conjunction but in a more complex DNF expression (a disjunction of conjunctions). If a meta-level rule combines multiple such interval features into a conjunctive rule, the premise of the resulting base-level rule is a conjunction of DNF expressions. Using these features directly in base-level rules results in much more complex rules. One may consider to compress these complex expressions into a simpler DNF expressions, but we have not yet dealt with this issue because our experiments showed that this complex approach does not offer any gain in predictive accuracy over the simpler nominal or binary encodings. We decided not to include these results in the following, because it is unclear how we should count the complexity of such conjunctions of DNF expressions.

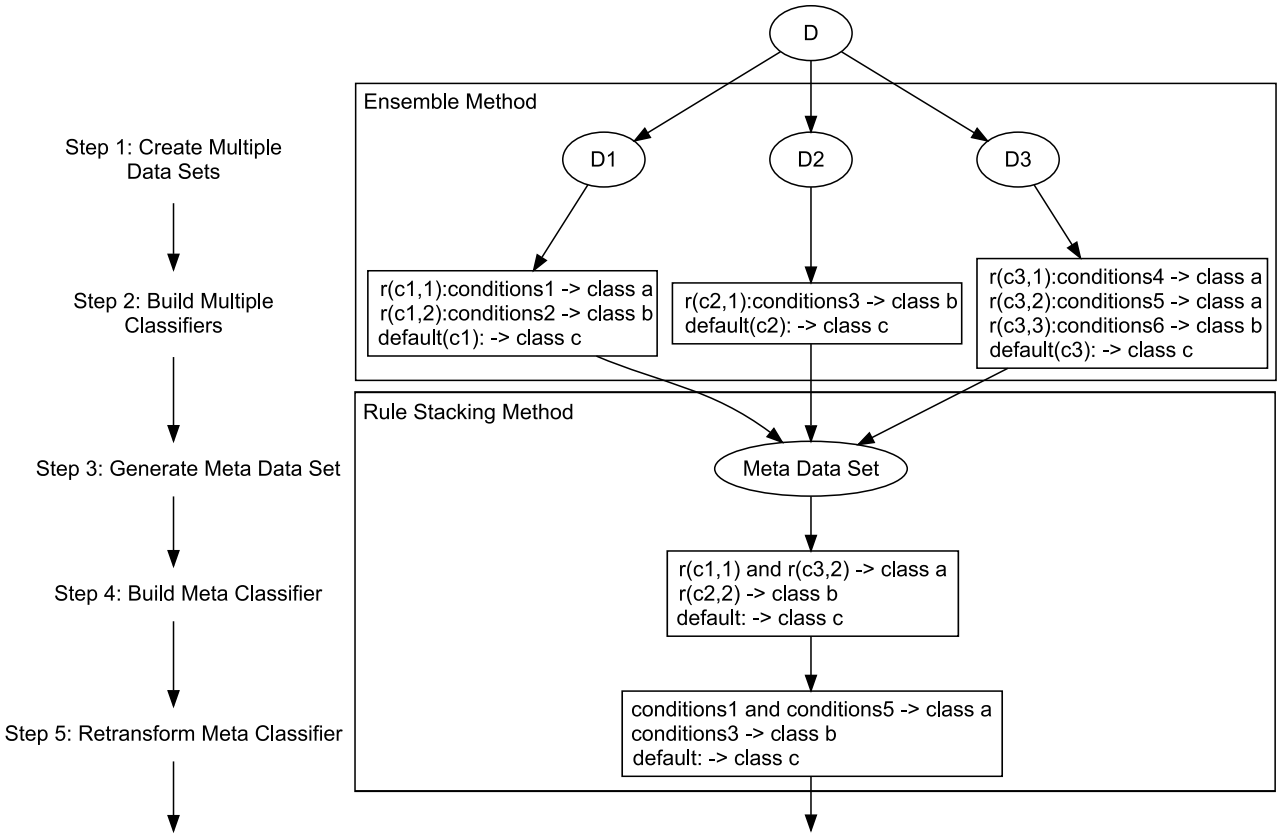

Figure 3.3: Schematic illustration of the Rule Stacking algorithm.

4 Experimental Setup

The goal of our experiments is to evaluate whether the rule stacking approach can maintain the improved performance of a classifier ensemble, and how significant the obtained compression of the ensemble is.

We performed our experiments within the WEKA framework [11]. For the rule generation we employed the rule learner JRip, the Weka implementation of Ripper [2], arguably one of the most accurate rule learning algorithms today. Contrary to William Cohen's original implementation, this re-implementation does not support the unordered mode, however the ordered mode of ripper was only used as it is more adequate for our purposes. Additionally both pruning methods of Ripper, the incremental reduced error pruning and pre-pruning using a minimum description length heuristic were applied.

For the generation of the classifier ensemble, we decided to use a pairwise class binarization. There are several reasons for this choice. On the one hand, it is known that a pairwise ensemble of Ripper has a better performance than a single standard Ripper [5], and we want to see whether this performance can be obtained. On the other hand, a key disadvantage of this pairwise decomposition is that while training can be performed efficiently (essentially in the same time as a one-against-all or a bagging ensemble consisting of c classifiers), we have to store a quadratic number of classifiers. In recent experiments on a large multilabel classification task with 4000 labels, the large memory demand resulting from the need for storing 8,000,000 base-level classifiers turned out to be the key bottleneck of the approach, which required a solution which was tailored to the use of perceptrons as base classifiers [8, 7]. The approach introduced in this paper may also be viewed as a solution for this problem that is tailored to rule-based classifiers. The final reason for choosing pairwise classifiers is that the diversity of the learning tasks in the individual ensemble members is considerably higher than for sampling-based ensemble methods such as bagging, because each base-level classifier tackles a different binary learning problem, whereas bagging tackles different training sets for the same classifier. We expect that this higher diversity makes it harder to compress the rules into a single classifier.

We evaluated the above setup on 14 multiclass data sets of the UCI repository [1]. Since the number of classes differs highly in these data sets we get a great range of different ensemble sizes. We compared our approach to the standard JRip and to its pairwise variant using a pairwise class binarization. For this comparison we considered the accuracy of each classifier and the size of the generated model measured by the number of rules and the total number of conditions of all rules.

5 Experiments

Table 5.1 shows the results of our experiments. On average, the accuracy of the standard JRip was the lowest in our experiments followed by its pairwise variant. This is consistent with the results reported in [5] (which were performed with a different implementation of the algorithms). Both variants of rule stacking, using nominal and binary features, yielded a higher accuracy than JRip, while maintaining the accuracy of the pairwise variant.

Considering the size of the model, the pairwise variant of JRip has expectedly the largest models in terms of the number of rules and conditions, whereas the smallest models were generated by the standard JRip. The models learned by the nominal rule stacking approach are of a comparable complexity to the rules learned by the original learner, and typically considerably smaller than the total complexity of the models learned with the pairwise approach (a notable exception being the *waveform-5000* dataset). The binary approach is somewhat more complex with respect to the number of rules, but considerably more complex with respect to the number of conditions. This can be partly explained with the fact that we did not eliminate redundant conditions from the resulting rules, which would lead to a further decrease in rule lengths.

The key result of these experiments is that rule stacking, in particular in the nominal encoding, maintains the high improvement in accuracy of the pairwise variant of JRip, while providing a very good compression of the ensemble of classifiers. As a result, we obtain rule sets that are of comparable complexity to those learned with JRip but are considerably more accurate.

Table 5.1: Comparison of the performance of the standard JRip, its pairwise variant (PW) and our approach (RS) using nominal (Nom.) and binary features (Bin.): number of classes ($|C|$) and pairwise problems (PP). accuracy and size of the Model (number of rules and conditions).

			Accuracy				Rules				Conditions			
					RS				RS				RS	
Data Set	$\|C\|$	PP	Jrip	PW	Nom.	Bin.	Jrip	PW	Nom.	Bin.	Jrip	PW	Nom.	Bin.
anneal	6	15	98,33	98,44	98,55	98,55	7	29	8	8	8	23	15	14
anneal.ORIG	6	15	95,32	94,88	95,10	95,21	14	36	12	12	37	38	27	36
balance-scale	3	3	80,80	78,72	80,00	79,04	12	15	9	9	39	35	24	35
glass	7	21	68,69	68,22	68,69	69,63	8	42	11	10	18	39	36	35
hypothyroid	4	6	99,34	99,39	99,31	99,39	5	16	6	6	11	21	15	19
iris	3	3	94,00	92,67	92,67	92,67	4	6	3	3	3	4	3	3
lymph	4	6	77,70	79,73	79,73	79,73	6	14	6	6	8	11	9	9
segment	7	21	95,71	96,45	96,36	96,15	24	63	16	24	63	72	32	85
soybean	19	171	91,95	92,83	92,09	90,92	26	355	24	26	45	199	36	48
splice	3	3	93,70	94,55	94,70	94,86	14	15	8	11	55	38	31	54
vehicle	4	6	68,56	71,51	69,62	69,98	17	31	20	19	43	55	67	77
vowel	11	55	69,70	80,81	78,79	78,28	48	199	48	52	138	260	151	293
waveform-5000	3	3	79,20	79,22	78,04	78,92	30	46	49	46	121	163	243	281
zoo	7	21	86,14	87,13	93,07	92,08	6	43	7	7	6	23	7	11
Average			85,65	86,75	86,91	86,81	15,79	65	16,21	17,07	42,5	70,07	49,71	71,43

6 Conclusions

In this paper we introduced a novel approach to compress an ensemble of base rule learners into global classifier which can be directly used to classify instances in the original data format without use of the base classifiers. Our experiments, using a pairwise class binarization to generate an classifier ensemble and JRip, an implementation of Ripper, as the base and meta classifier, showed that we maintain the performance of the classifier ensemble and that the model size of the compressed global classifier is comparable to the model size of a directly applied JRip.

So far, our results are limited to pairwise ensembles, but we expect that they should carry over to bagging and boosting ensembles as well, because in these ensembles, the correlation between the individual learning tasks is much higher than for pairwise classification, where each classifier encodes a different learning task. As a result, the resulting base-level rule sets should be easier to combine.

Acknowledgments:

This research was supported by the German Science Foundation (DFG) under grants FU 580/2 and FU 580/3.

Bibliography

[1] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007. 8

[2] W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pages 115–123, Lake Tahoe, CA, 1995. Morgan Kaufmann. 8

[3] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research (JAIR)*, 2:263–286, 1995. 2

[4] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 155–164, San Diego, CA, 1999. ACM. 2

[5] J. Fürnkranz. Integrative windowing. *Journal of Artificial Intelligence Research*, 8:129–164, 1998. 8, 9

[6] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, February 1999. 3

[7] E. Loza Mencía and J. Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In W. Daelemans, B. Goethals, and K. Morik, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Disocvery in Databases (ECML-PKDD-2008), Part II*, volume 5212 of *Lecture Notes in Computer Science*, pages 50–65, Antwerp, Belgium, Sept. 2008. Springer. 8

[8] E. Loza Mencía and J. Fürnkranz. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In E. Francesconi, S. Montemagni, W. Peters, and A. Wyner, editors, *Semantic Processing of Legal Texts*, volume 6036 of *Lecture Notes in Artificial Intelligence*, pages 192–215. Springer-Verlag, 1st edition, May 2010. In press. 8

[9] A. K. Seewald. How to make stacking better and faster while also taking care of an unknown weakness. In C. Sammut and A. G. Hoffmann, editors, *Proceedings of the 19th International Conference (ICML-02)*, pages 554–561, Sydney, Australia, 2002. Morgan Kaufmann. 4

[10] K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999. 4

[11] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2005. 8

[12] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992. 2, 4