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

Jan-Nikolas Sulzmann and Johannes Fürnkranz

TU Darmstadt, Knowledge Engineering,
Hochschulstr. 10, 64289 Darmstadt, Germany
{sulzmann, juffi}@ke.tu-darmstadt.de

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 rules cover 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 [7]. 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 a 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 [8].

A different approach is to employ stacking at the meta level [17]. 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 alternative 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 Sections 4 and 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 [10]. 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:

$$condition_1 \wedge \cdots \wedge condition_{|r|} \implies class \tag{1}$$

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

$$\text{covers}(r, i) = \begin{cases} \mathsf{true}, & \text{if } r \supseteq i \\ \mathsf{false}, & \text{otherwise.} \end{cases} \tag{2}$$

We will denote the set of all covering rules $\text{Cov}(c, i)$ for a given classifier and a given instance i as

$$\text{Cov}(c, i) = \{r \in R_C | \, \text{covers}(r, i)\} \tag{3}$$

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 [17]. 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 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 [15]. Subsequently, it was shown that it may be even better to use the entire predicted class probability distribution [14].

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

Fig. 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 [15, 14]. 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 classifier which can be directly used on the original instances (Section 3.4). The complete approach is illustrated in Figure 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

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

Fig. 2: Illustration of the meta-level datasets

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 the information which rules of the given base learners cover the instance. This information can be encoded at the meta level by creating one binary attribute for each rule in the ensemble, as shown in Figure 2. 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.

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

Please note that the default rules of the base classifiers are ignored in the meta data generation.

We have also experimented with other encodings. One nominal feature for each base-level classifier could be used for indicating which rule of the corresponding classifier covers the example first. The same information could also be encoded as a numerical features which specifies the index of the first rule that covers the example. However, both approaches did not perform better than the simple binary approach, but made the decoding process more complex because they essentially assume a decision list, i.e., each rule consists not only of its body, but also of the negation of the bodies of all previous rules. For this reason, we do not further consider these approaches in this paper.

Let us illustrate the generation of the meta data set with the help of a toy example. One of the data sets used in our experiments was *zoo*, which records the characteristics of animals divided in different classes, e.g. insects or invertebrates. The last classifier of the learned pairwise 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	Meta-Level Features
Termite	$(\cdots, \text{false}, \text{true}, \text{true}, \text{insect})$
Lobster	$(\cdots, \text{false}, \text{false}, \text{true}, \text{invert.})$
Crow	$(\cdots, \text{true}, \text{false}, \text{true}, \text{bird})$

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 on compressing 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.

The condition of a binary feature tests whether a specific rule covers an example or not ($\text{covers}(r_l^c, i) = \text{true/false}$). 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 ignored. 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. The re-transformation is illustrated in Figure 3, step 4 to 5.

The situation is somewhat more complicated if negated meta conditions are allowed, because a negated meta condition corresponds to a negated conjunction of base conditions. We currently simply add the negated conjunction directly to the re-transformed rule. This has the effect that the resulting rule set is no longer in disjunctive normal form. One may are argue that in this case, rule stacking may have a somewhat unfair advantage over ordinary rule learners which are confined to conjunctions in their rule bodies. For this reason, we report results on both variants, the one where negated meta conditions are allowed, and the one where they are forbidden, after which the re-transformed rules are again in DNF and directly comparable to a conventional rule learning algorithm.

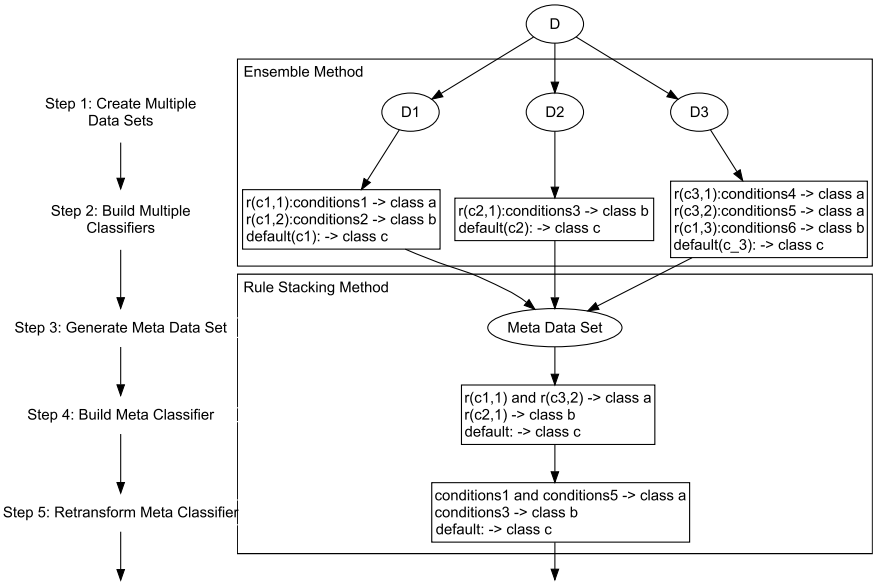


Fig. 3: Schematic illustration of the Rule Stacking algorithm.

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 [16]. For the rule generation we employed the rule learner JRip, the Weka implementation of Ripper [4], 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 for learning rule sets instead of decision lists, but as we only deal with binary base-level classification problem, there is no practical difference between these two modes. 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 [9], 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 [13, 12]. 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 24 multiclass data sets of the UCI repository [2]. Since the number of classes differs highly in these data sets we get a great range of different ensemble sizes. In our experiments we always employed binary features at the meta level and either allowed negated meta conditions (w.N.) or we did not (wo.N.). We compared our rule stacking approach (RS) to the standard JRip and to its pairwise variant using a pairwise class binarization (PW). 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. The accuracy of each classifier was computed using a 10-fold cross validation. The number of conditions of our re-transformed meta classifier was determined by removing duplicate conditions, so each condition is only counted once.

For the evaluation of the results we used the Friedman test with a post-hoc Nemenyi test as proposed in [5]. The significance level was set to 5% for both tests. The results of the Nemenyi tests are depicted as critical distance charts (abbreviated CD chart).

5 Experiments

Table 1 shows the detailed results of our experiments. We applied a Friedman test to each evaluation measure: accuracy, number of rules and number of conditions. In all three cases, the test rejected the equality of the classifiers. So post-hoc Nemenyi tests were performed, the corresponding CD-Charts are depicted in Figure 4.

For accuracy, the Nemenyi test identified two groups of equivalent classifiers (see Figure 4(a)). One can see that the pairwise variant of JRip significantly outperforms its standard variant, confirming the results of [11]. Moreover, both variants of rule stacking are not significantly worse than the pairwise variant, but allowing negations at the meta level clearly seems to be preferable. While this variant is somewhat worse than the original pairwise version of JRip, this difference is quite small, and not significant. It is still significantly better than conventional JRip. On the other hand, the variant that does not allow negations still seems to be preferable over conventional JRip, but this result is not statistically significant. Similar observations can be made if the average accuracies of the classifiers are compared in Table 1.

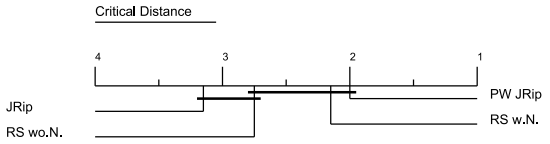
Table 1: Comparison of the performance of the standard JRip, its pairwise variant (PW) and our approach (RS) allowing negated meta conditions (w.N.) or not (wo.N.): number of classes ($|C|$) and pairwise problems (PP), accuracy and size of the Model (number of rules and conditions).

			Accuracy				Rules				Conditions			
					RS				RS				RS	
Data Set	$\|C\|$	PP	Jrip	PW	w.N.	wo.N.	Jrip	PW	W.N.	WO.N.	Jrip	PW	W.N.	WO.N.
anneal	6	15	95,32	94,88	95,21	94,99	14	36	12	12	37	38	36	28
autos	6	15	73,17	75,12	74,15	76,59	13	41	14	14	25	42	30	29
balance-scale	3	3	80,80	78,72	79,04	73,76	12	15	9	2	39	35	24	2
bridges v1	6	15	61,90	62,86	65,71	61,90	5	33	6	7	6	21	12	14
car	4	6	86,46	90,34	87,50	87,91	49	66	37	38	195	210	143	146
cmc	3	3	52,41	54,85	53,36	52,55	5	12	4	4	14	26	15	16
dermatology	6	15	86,89	91,26	92,62	91,53	15	39	8	13	27	32	23	39
ecoli	8	28	81,25	81,55	81,25	80,65	10	55	9	8	19	35	22	18
glass	7	21	68,69	68,22	69,63	68,69	8	42	10	11	18	39	34	37
hypothyroid	4	6	99,34	99,39	99,39	99,39	5	16	6	6	11	21	19	16
lymph	4	6	77,70	79,73	79,73	79,73	6	14	6	6	8	11	9	9
optdigits	10	45	90,78	94,96	93,01	92,51	74	220	80	90	312	391	614	657
pageblocks	5	10	96,84	97,02	96,93	96,93	14	36	15	17	30	54	44	53
segment	7	21	95,71	96,45	96,15	92,16	24	63	24	25	63	72	85	78
solar-flare-c	8	28	85,40	85,22	85,34	85,34	2	33	2	2	4	11	3	3
soybean	19	171	91,95	92,83	90,92	91,65	26	355	26	27	45	199	48	56
splice	3	3	93,70	94,55	94,86	94,73	14	15	11	12	55	38	43	48
thyroid_hyper	5	10	98,49	98,67	98,70	98,70	5	20	6	6	14	29	21	20
thyroid_rep	4	6	98,94	99,05	99,02	99,02	8	14	6	6	22	18	16	16
vehicle	4	6	68,56	71,51	69,98	69,27	17	31	19	14	43	55	77	56
vowel	11	55	69,70	80,81	78,28	75,25	48	199	52	61	138	260	292	325
waveform-5000	3	3	79,20	79,22	78,92	79,08	30	46	46	46	121	163	316	284
yeast	10	45	58,09	57,88	57,48	57,35	15	127	18	16	38	153	69	59
zoo	7	21	86,14	87,13	92,08	91,09	6	43	7	7	6	23	12	15
Average		23,21	82,39	83,84	83,72	82,95	17,71	65,46	18,04	18,75	53,75	82,33	83,63	84,33
Average Rank			3,13	2	2,15	2,73	1,83	3,96	1,98	2,23	1,71	3,33	2,44	2,52

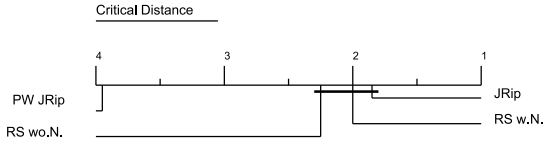
Considering the number of induced rules (see Figure 4(b)), we discern two disjunct groups of equivalent classifiers. Pairwise JRip is the single member of the worst group, i.e., it typically induces larger rule sets than the other classifiers. All other classifiers belong to the best group of classifiers, hence their rule sets are of a comparable size. This finding is also reflected in the average number of induced rules for each classifier in Table 1.

At last, we compare the number of rule conditions of our re-transformed meta classifier (see Figure 4(c)). Here, the results are more diverse. In essence, both versions of the rule stacking approach lie in the middle of these two classifiers, being neither significantly worse than JRip nor significantly better than pairwise JRip. The detailed results in Table 1 confirm this diversity, mixing results where the size of the resulting theory is even smaller than JRip's (e.g., *car* or *soybean*), with results where it is considerably higher than the simple pairwise approach (e.g. *optdigits* or *waveform500*). The latter results dominate the average values.

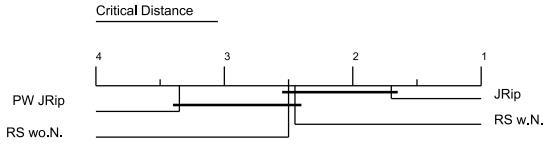
The key result of these experiments is that rule stacking, in particular if negated meta conditions are allowed, maintains the high improvement in accuracy of the pairwise variant of JRip, while often providing a good compression of the ensemble of classifiers. As a result, we often obtain rule sets that are of



(a) accuracy



(b) number of rules



(c) number of rule conditions

Fig. 4: Critical Distance Charts.

comparable complexity to those learned with JRip but are considerably more accurate.

We still have to investigate the reasons for the extreme values in some cases. However, note that they are not too critical to our original goal because we do not have to store these rules in the explicit DNF version. In particular, pointers to previously used rule bodies allow that conjunctions that are used more than once can be stored as separate rules, which need to be evaluated only once for each example. In a way, this may also be viewed as an approach to automated feature construction.

6 Conclusions

In this paper we introduced an algorithm that allows to compress a rule set that has been learned by a pairwise classifier into a single rule set. The resulting rule set is often of comparable size to the one directly learned by JRip, and considerably less complex that the original pairwise ensemble. In terms of the number of rules, this advantage is consistent and significant. However, in terms of the number of conditions, there are also a few cases where the re-transformed rules are considerably more complex because of rules that are used multiple times in various rules. However, we note that a structured representation of these rules (e.g., through generated meta-level features) may still provide a useful compression of the pairwise theories.

Our work may be viewed in the context of approaches that induce rules from opaque concepts such as neural networks [1], support vector machines [6], or ensembles [8]. Our approach differs from these approaches in that it extracts rules from interpretable concepts, and therefore does not need to consider the predictions of these models, but can directly use the learned rules. Such ideas have already been used in somewhat different context (cf., e.g., [3]), but its implementation in terms of a stacking framework and, most importantly, with the goal of compressing pairwise theories, is new. We still need to investigate whether these results generalize to arbitrary rule-based ensemble classifiers.

References

1. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl.-Based Syst. 8(6), 373–389 (1995)
2. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007), `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`
3. van den Bosch, A.: Using induced rules as complex features in memory-based language learning. In: Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning. pp. 73–78. Association for Computational Linguistics, Morristown, NJ, USA (2000)
4. Cohen, W.W.: Fast effective rule induction. In: Prieditis, A., Russell, S. (eds.) Proceedings of the 12th International Conference on Machine Learning (ML-95). pp. 115–123. Morgan Kaufmann, Lake Tahoe, CA (1995)
5. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
6. Diederich, J. (ed.): Rule Extraction from Support Vector Machines, Studies in Computational Intelligence, vol. 80. Springer (2008)
7. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research (JAIR) 2, 263–286 (1995)
8. Domingos, P.: 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). pp. 155–164. ACM, San Diego, CA (1999)
9. Fürnkranz, J.: Integrative windowing. Journal of Artificial Intelligence Research 8, 129–164 (1998)
10. Fürnkranz, J.: Separate-and-conquer rule learning. Artificial Intelligence Review 13(1), 3–54 (February 1999)
11. Fürnkranz, J.: Round robin classification. Journal of Machine Learning Research 2, 721–747 (2002), `http://www.ai.mit.edu/projects/jmlr/papers/volume2/fuernkranz02a/html/`
12. Loza Mencía, E., Fürnkranz, J.: Efficient pairwise multilabel classification for large-scale problems in the legal domain. In: Daelemans, W., Goethals, B., Morik, K. (eds.) Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Disocvery in Databases (ECML-PKDD-2008), Part II. Lecture Notes in Computer Science, vol. 5212, pp. 50–65. Springer, Antwerp, Belgium (Sep 2008)

13. Loza Mencía, E., Fürnkranz, J.: Efficient multilabel classification algorithms for large-scale problems in the legal domain. In: Francesconi, E., Montemagni, S., Peters, W., Wyner, A. (eds.) Semantic Processing of Legal Texts, Lecture Notes in Artificial Intelligence, vol. 6036, pp. 192–215. Springer-Verlag, 1st edn. (May 2010), in press
14. Seewald, A.K.: How to make stacking better and faster while also taking care of an unknown weakness. In: Sammut, C., Hoffmann, A.G. (eds.) Proceedings of the 19th International Conference (ICML-02). pp. 554–561. Morgan Kaufmann, Sydney, Australia (2002)
15. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)
16. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco, CA, 2nd edn. (2005)
17. Wolpert, D.H.: Stacked generalization. Neural Networks 5(2), 241–260 (1992)