

# An Experimental Study of Using Rule Induction Algorithm in Combiner Multiple Classifier\*

Jerzy Stefanowski<sup>1</sup> and Sławomir Nowaczyk<sup>2</sup>

<sup>1</sup>*Institute of Computing Sciences, Poznań University of Technology, 60-965 Poznań, Poland*

*E-mail : Jerzy.Stefanowski@cs.put.poznan.pl*

<sup>2</sup>*Department of Computer Science, Lund University, 221 00 Lund, Sweden*

*E-mail : Slawomir.Nowaczyk@cs.lth.se*

**Abstract:** Multiple classifiers consist of sets of subclassifiers, whose individual predictions are combined to classify new objects. These approaches attract an interest of researchers as they can outperform single classifiers on wide range of classification problems. This paper presents an experimental study of using the rule induction algorithm MODLEM in the multiple classifier scheme called combiner, which is a specific meta learning approach to aggregate answers of component classifiers. Our experimental results show that the improvement of predictive accuracy depends on the independence of errors made by the base classifiers. Moreover, we summarise our experience with using MODLEM as component in other multiple classifiers, namely bagging and  $n$  classifiers.

## I. Introduction

In the last decades one can observe a growing research interest in fields of *machine learning* and *knowledge discovery* [18]. Let us remark that recent development of technologies for collecting and storing data has led to huge data repositories, which are extremely difficult for humans to analyse and to summarise. Therefore, many data mining techniques are being developed in order to support extracting various knowledge representations from such large data bases [7]. One of the main tasks considered in knowledge discovery is *supervised classification*, where learning process is provided with a set of training examples of target concepts (classes). Each example corresponds to a single object to be classified and is described by a finite of attributes (features). The goal of learning is to discover a rule or a function (in machine learning often called a *hypothesis*) which maps such descriptions into those classes [18, 20]. An algorithm, which consists of knowledge representation (learned from some training set) and the strategy of its usage, forms a *classifier*, which can be used to predict classes of new coming objects. A typical measure used to evaluate classifier's performance is *classification accuracy*, i.e. percentage of correctly classified examples.

Several algorithms have been proposed over the years for inducing various knowledge representations and various classifiers (a review is available, for example, in [16, 18, 26, 14]). Although those algorithms are very effective for many classification problems, they do not always lead to satisfactory classification accuracy in more complex and difficult cases. As it is shown in theoretical studies and confirmed in empirical comparative studies ([19, 6]), there is no single best algorithm to be used for all data sets. Each and every algorithm has its own area of superiority and is "specialised" to solve some classes of learning problems.

Since different algorithms have different strengths and weaknesses – a fact which makes some of them more suitable for certain problems than others – in the last decade many researches attempted to increase classification accuracy by combining several classifiers into an integrated system. Such systems are known under several names: *multiple classifiers*, *ensembles*, *committees* or *classifier fusion* [6, 14, 29]. They are shown, both theoretically and experimentally, to outperform single classifiers on a wide range of problems. These classifiers can be constructed in many ways, such as by changing the distributions of examples in the learning set, by manipulating the input features, by using different learning algorithms on the same data or by manipulating the output targets. The predictions of base classifiers are typically combined by voting.

Our research interest includes creating classifiers by using *rules*. It is widely believed that rules are one of the most popular type of knowledge used in practice, mainly due to the fact that they are very expressive and human-readable at the same time. A number of algorithms has been developed to induce such rules, for a review see, e.g., [9, 16, 18, 26]. Here we consider a rule induction algorithm called MODLEM, introduced by Stefanowski in [25]. In the previous work [27] it has been successfully applied inside two different multiple classifier schemes: the *bagging* and the *multi-class  $n$*  classifiers. In both cases the experiments

showed a significant improvement of classification accuracy on many diverse data sets. However, both the above multiple classifiers are homogeneous ones (i.e. the same learning algorithm is used over different data sets obtained from the original training set). It is our belief that checking other, more heterogeneous, concepts of multiple classifiers is worthwhile as well.

Therefore, in this study we want to extend our previous research by using rule induction algorithm MODLEM in another type of multiple classifier – one called *combiner*, introduced in [3]. In this type of classifier several different learning algorithms are applied to the same data, in order to generate base classifiers have heterogeneous knowledge representations. Then, their predictions are combined by an additional – metalevel – classifier into a final classification decision of the system.

The main aim of this paper is to experimentally verify, on several benchmark data sets, whether the combiner classifier which includes the base classifier also induced by the MODLEM algorithm can achieve higher classification accuracy than single classifiers used as its components. Furthermore, we intend to study the impact of diversification of base classifiers on classification accuracy of the combiner. We hope that those additional results will extend knowledge about internal working conditions of this multiple classifier, as the main researchers have rather focused their attention on the global performance (cf. e.g. papers [3, 4, 17]).

Finally, we will summarise the experience of using the MODLEM algorithm in the framework of different types of multiple classifiers.

This paper is organised as follows. In the next section we briefly describe the rule induction algorithm MODLEM. Section 3 contains general issues concerning construction of multiple classifiers. Then, in section 4 we summarise current results of using MODLEM in the framework of bagging and  $n$  multiple classifiers. Section 5 presents the concept of combiner classifier, and its experimental evaluation is described in section 6. Final section contains discussion of the obtained results and some closing remarks.

## II. Rule Induction by MODLEM Algorithm

The rules induced from examples are represented as logical expressions of the following form:

*IF (conditions) THEN (decision class),*

where conditions are conjunctions of elementary tests on values of attributes, and decision part of the rule indicates the assignment of an object (which satisfies the condition part) to a given decision class.

A number of algorithms for inducing rules of this kind have already been proposed, for review see, e.g. [9, 25, 26]. In our study we will use the algorithm called MODLEM, introduced by Stefanowski in [25]. Due to the size of this paper we will skip the formal presentation of this algorithm and only discuss its main idea; for more details c.f. [25, 26, 28]. The MODLEM algorithm is based on the scheme of a

*sequential covering* and it heuristically generates a *minimal set* of decision rules for every *decision concept* (decision class or its rough approximation — in case of inconsistent descriptions of examples elements of the rough set theory are used [23]). Such a set of rules attempts to cover all (or the most significant) *positive examples* of the given concept and not to cover any *negative examples* (or cover as few of them as possible). The main procedure for rule induction starts by creating a first rule, i.e. sequentially choosing the “best” elementary conditions according to chosen criteria (we have used *class entropy* in our experiments).

When a rule is found and stored, all positive learning examples that match this rule are removed from consideration. The process is iteratively repeated as long as some positive examples of the decision concept remain still uncovered. Then, the procedure is sequentially repeated for each other decision class. Let us comment that in the basic version of the MODLEM algorithm elementary conditions are evaluated by using one of two measures, either *class entropy* or *Laplace accuracy* [25, 26]. It is also possible to consider a lexicographic order of two criteria, measuring the rule positive cover and then its conditional probability (an idea originally considered by Grzymala in his LEM2 algorithm or its latest, very interesting, modification called MLEM2).

Moreover, a distinguishing feature of the MODLEM algorithm is its ability to handle numerical attributes directly, during rule induction, without any need for preliminary discretisation phase. In MODLEM elementary conditions on numerical attributes are represented as either  $(a < va)$  or  $(a = va)$ , where  $a$  denotes an attribute and  $va$  its value. The choice of the threshold  $va$  results from checking several possible candidates and evaluating the temporary condition over uncovered examples by means of the chosen evaluation measure. Moreover, if the same attribute is chosen twice while building a single rule, one may also obtain the condition  $(a = [v1, v2])$  that results from an intersection of two conditions  $(a < v2)$  and  $(a \geq v1)$  such that  $v1 < v2$ . On nominal attributes, equivalent conditions are  $(a = va)$ . For more details about the function finding best elementary conditions see, e.g., [11, 25].

If the input data contains *inconsistent examples*, i.e. examples described by the same attribute values but belonging to different classes, the rough sets theory [23] can be used to handle them. In this theory inconsistencies are neither removed from consideration nor aggregated by probabilistic operations. Instead, lower and upper approximations of each class are computed. On their basis two corresponding sets of rules: certain and possible, are induced.

When the set of induced rules is applied to classify new objects, a classification strategy introduced by Grzymala in LERS system is used, which takes into account support of all rules completely matched and also allows partially matches if no rule fits the description of the tested example [10].

### III. Constructing Multiple Classifiers

One of the basic assumptions of the inductive learning states that any hypothesis found to approximate the unknown target function *sufficiently well* over a *sufficiently large* set of training examples will also approximate the target function *well* over unobserved examples [20]. Even though many different hypotheses may approximate target concept over learning set just as well, an algorithm must choose only one of them at learning time, depending on so called *inductive bias*. According to [20], inductive bias can be seen as a minimal set of assumptions sufficient for a hypothesis found by a particular algorithm to follow deductively from the information given by data set. We repeat arguments from [6] that, in practice, quite often inductive bias assumptions are only partially true for a given set of data: only a limited part of learning sample is available or the target function is defined outside the representation space. Therefore, an algorithm cannot perfectly generalise the unknown target function and it is clear that some algorithms are better suited for particular problems than others. One of the attempts to overcome these obstacles is the idea and development of multiple classifiers.

A multiple classifier is a set of single (standard) classifiers whose individual predictions are combined in some way when classifying new objects. There are two main decisions to be made when designing ensembles of classifiers: first, how to create *diverse* base classifiers, and second, how to integrate their predictions.

More discussion on motivation for creating integrated systems is given in [6, 29]. A number of methodological studies say that combining identical classifiers is useless [12, 8]. Therefore, a necessary condition for an effective integration is that base classifiers demonstrate a substantial *level of disagreement*, i.e. the mistakes each of them makes are unique and, as much as possible, different from mistakes made by its peers. In other words, base classifiers should make classification errors independently with respect to one another.

Diversified base classifiers can be generated in many ways, see, for example, discussions in [6, 26, 29]. In general, either *homogeneous* or *heterogeneous classifiers* are constructed.

In the first category, the same learning algorithm is used over different samples of the data set. The bestknown examples are either boosting and bagging techniques which manipulate training data by including (or excluding) particular examples, or methods that manipulate set of attributes, e.g. the one discussed in [5]. In addition, multiple classifiers can be trained over different samples or partitions of data sets, even distributed over computation nodes. For more review of various methods see e.g. [8, 14, 29].

In the second category, different learning *algorithms* are applied to the same data set, and the diversity of results comes from heterogeneous knowledge representations and different evaluation criteria used by them. The *combiner* approach, mentioned in Introduction, is an example of

heterogeneous multiple classifier.

Another issue is how predictions of base classifiers should be aggregated into a final classification decision of the combined system. In general, there are two kinds of methods: *group combination* or *specialised selection*. In the former case all base classifiers are asked to classify each new object, while in the latter only those classifiers “expertised” in particular object are consulted. *Voting* is the most common method used to combine classifier decisions. In an *equal weighted* voting, the prediction of each base classifier is treated as a single vote for the particular class and the class with the highest number of votes is selected as the final classification. Alternatively, the vote of each classifier may be *weighted*, for example, by the estimation of its accuracy or by its “confidence” in a particular prediction.

There are also more advanced aggregation rules, using Bayesian rule or fuzzy aggregation operations — for review see [29]. A method used in our experimental evaluation of *combiner* is sometimes called *explicitly trained combination rule* and we will discuss it in more detail in section V.

### IV. Current Experience with Using MODLEM<sub>2</sub> Algorithm in Bagging and $n$ -classifier

Up to now the rule induction algorithm MODLEM was applied to two different multiple classifiers, i.e., the bagging and  $n$  classifier. The bagging, introduced by Breiman [2], is a popular approach based on diversification of input data. Several bootstrap samples are obtained from the original learning set by uniformly sampling objects from this set with replacement; the exact details are given in [2, 27]. Afterwards, the same learning algorithm is run several times, once for each bootstrap sample. The generated base classifiers are, finally, combined by an equal voting strategy in order to produce the final answer.

The use of MODLEM to generate base classifiers in the bagging was studied in detail in [27], where classification accuracies obtained were compared against those of a single classifier (which was also induced by MODLEM). In those experiments, several data sets were used, most of them coming from ML Irvine repository [1]. The results of obtained showed that the bagging technique significantly (in the sense of paired tStudent test with a significance level  $\alpha=0.05$ ) outperformed the single classifier on 14 data sets out of 18 that have been used. The difference between single and multiple classifiers was nonsignificant on 3 data sets only (ones that are relatively easy to learn, such as *iris* and *bricks* — which are known to be characterised by a linear separation between classes).

Moreover, it was observed that performance of the bagging approach was slightly worse for data sets with smaller number of examples (such as *buses* and *zoo* — which seemed to be too small for efficient sampling), and significantly better performance for data sets containing a high number of examples. For a number of these data sets

we observed an substantial increase of predictive accuracy, e.g. for *hsv* — over 10%, for *bupa* — approximately 10% and for *hepatitis* — 5.43%. The other goal was to examine the influence of choosing the number of subclassifiers on the final classification accuracy. Here the results were not unambiguous, as for some data sets increasing this number was useful, while for others it was not.

The second set of experiments (carried out in the study [27]) concerned the  $n$  classifier, which is a specialised approach to solve *multiple class learning problems*. Its construction is consistent with the idea of the classification by *pairwise coupling* [13]. Generally, the main idea is to decompose the  $n$  class problem into  $(n-1)n/2$  two class

problems (where  $n$  is a number of decision classes). Each pair of classes is discriminated by a *binary classifier* specialised for it, i.e. trained on examples coming from these classes only and predicting assignment to one of these classes. All binary classifiers are generated using the same learning algorithm. While classifying new instance, all binary classifiers are applied to it, each producing its own prediction. The final decision is the class which wins the most of those pairwise comparisons. In our implementation, we used a more advanced *weighted* majority voting rules, where the vote of each classifier is modified by its credibility — a quantity calculated on a basis of its classification performance during learning phase (more details about this technique can be found in [15]). Recently, some researchers have also studied more sophisticated aggregation approaches, where a part of the most relevant binary classifiers take part in the final decision [22].

Experiments with using MODLEM algorithm to create  $n$  classifier were presented in [27], where this multiple classifier was compared against a single classifier (also induced by MODLEM) on 11 data sets, all concerning multiclass learning problems (with a number of classes varied from 3 up to 14). These results showed that the  $n$  classifier significantly (again, in the sense of paired  $t$  test with a significance level  $\alpha = 0.05$ ) outperformed the single classifier on 7 out of 11 problems, e.g. the increase in classification accuracy happened for *hist* — over 3.7%, *glass* — around 2.7%, *automobile* — 2.5% and, finally, *metadata* — 2.6%. These improvements were not as high as in the bagging case but still they did occur for many difficult multiclass problems. Again, there was no improvement for easier problems (e.g. *iris*), but multiple classifiers performance was better for data sets with higher number of examples.

Comparing results of both multiple classifiers, it seems that bagging is a more universal approach but  $n$  classifier, which is in fact a specialised approach for learning multiple classes, is indeed slightly better for many multiclass, difficult data sets. Generally, our experiments showed that MODLEM can be successfully applied in both of those multiple classifier schemes. Moreover, we noticed that its classification performance in the multiple classifiers is

somewhat similar to that of C4.5 decision tree learning algorithm. According to Breiman [2], good algorithms for use in homogeneous ensembles are so called *unstable algorithms*, i.e. ones whose output classifier undergoes major changes in response to very small changes in the training data. Indeed, MODLEM is this kind of algorithm.

## V. The Combiner Strategy for Aggregation of Base Classifiers Predictions

In the current paper we focus on the approach called *combiner*, proposed by Chan & Stolfo. It is based on the idea of *metalearning*, which is loosely defined as "learning from learned knowledge" [3]. This goal is achieved in a kind of layered architecture, where the classifiers at the base level (i.e. base classifiers) receive the original data as input. The predictions they output are then merged by an additional, higherlevel classifier, so called *metaclassifier*, into a final prediction of the system.

For a more comprehensive overview let us remark that in their paper Chan & Stolfo originally proposed three general metalevel strategies, which use different ways of merging base classifier predictions, so called *arbiter*, *combiner* and *hybrid* strategies. However, taking into account the results presented in [3, 4] (where authors showed that the combiner constructed with different kinds of decision trees works better than voting strategies on several biomedical data sets) and the simplicity of the concept, we chose only the *combiner* approach for our experiments.

The base classifiers are generated from the same data set by different learning algorithms, which means that they have different inductive bias, use different search strategies and knowledge representations. Predictions made by the base classifiers on a set of validation examples, together with correct decision classes, form a meta-level training set — as illustrated in Figure 1.

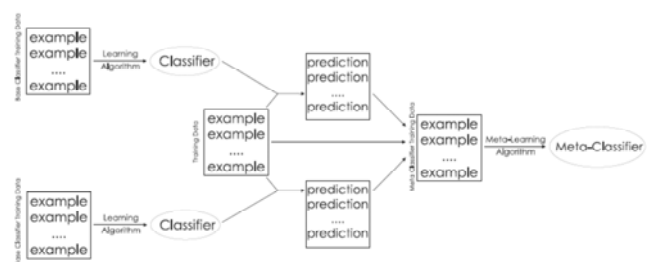
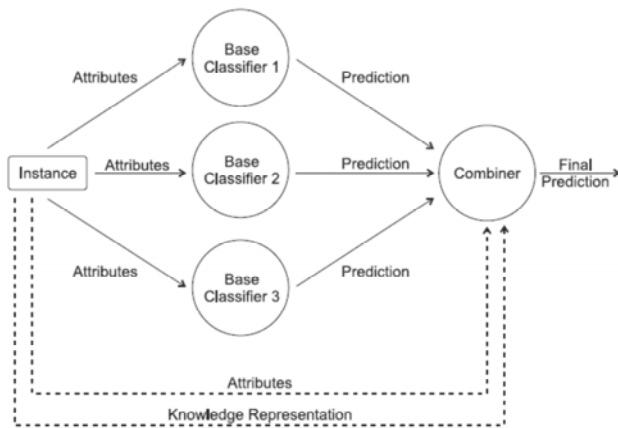


Figure 1 : Constructing the meta-classifier

In general, there are three types of information available to meta-classifier: first, it can utilise original description of examples given in learning data set; second, predictions of base classifiers for every example are available; and third, knowledge description generated by base classifiers can potentially also be useful. In our experiments we only use the second option for construction of meta-level training set (for a discussion of advantages and drawbacks of each

method see [3, 4]). After a meta-level training set is created, an extra learning algorithm can be applied to it. It learns how to combine base classifiers predictions into a final decision, thus inducing a meta-classifier: *combiner*.

While classifying a new instance, predictions of all base classifiers are first computed. From them, a new meta-level instance is created, which is then presented to the combiner (meta-classifier) and classified by it — see Figure 2. The motivations for this strategy is: “to coalesce the predictions from the base classifiers by learning the relationship or correlation between these predictions and the correct final decision” [4]. There is some hope that these indirect relationships may be easier to learn than the relationships between original attributes and decision classes and that the meta-classifier may correct some mistakes made by the base classifiers [3, 21].



**Figure 2 :** Classification of new examples by a combiner

Following Chan & Stolfo construction, we decided to use three base classifiers in our experiments. We use three wellknown learning algorithms to induce them: KNN, C4.5 and MODLEM algorithms. KNN is implemented as 5 nearest neighbour with an Euclidean distance for numerical attributes and  $\{0, 1\}$  values for nominal ones. C4.5 tree learner is an implementation supplied by Quinlan [24] and is used as a pruned tree with standard options (except for our use of GainRatio for selecting tests in tree nodes). The MODLEM algorithm is used with Entropy criterion to evaluate elementary conditions.

The metaclassifier is a typical Naive Bayesian — this choice was also inspired by results of [3], where authors claim that this algorithm was the most successful one as a combiner, on data sets they used. More details on our experimental framework are available in [21].

## VI. Experiments

The first purpose of our experiments is to verify whether the combiner classifier, with MODLEM algorithm inducing one of base classifiers, could achieve a higher classification accuracy than its components. Moreover, we want to

identify conditions under which the combiner aggregation strategy may improve the final prediction accuracy. We are interested in verifying experimentally in what situations the combiner is able to correct mistakes made by base classifiers. Let us remind that the main assumption states that the multiple classifier may improve predictive accuracy if its components make errors independently of each other. Thus, we extend our experiments by collecting more information about relative interactions between wrong and correct predictions of base classifiers and the overall final decision of the combiner.

All experiments were performed on the benchmark data sets, which were coming either from Machine Learning Repository at the University of California at Irvine or from author’s case studies (data sets *ACL*, *hsv*, *imidasolium*). Due to the size of the paper we skip their precise characteristics, only presenting most general information in Table 1 (for more details see [1]). The particular data were either chosen in order to provide comparison with previous studies using MODLEM in the multiple classifiers, or to have more diversified experimental setting – taking into account number of examples and decision classes.

Some of these data sets were slightly modified. If the data contained missing attribute values, they were substituted in preprocessing by the average or mode values in the examples belonging to the same class. For MODLEM, inconsistent descriptions were handled by computing rough approximations. The classification accuracy was estimated as an average value obtained by using a 10-fold cross validation technique.

First we evaluate the overall classification accuracy of the combiner compared against the use of single base classifiers on the same data set. Results are presented in Table 2 and will be discussed in section 7. Moreover, in our experiments we gather additional information about base classifier predictions, both in situations when the combiner makes a correct class assignment for testing example and when it makes a wrong decision. We also store extra information about decisions made by base classifiers – noting whether they were the same or different, as well as correct or incorrect. Some exemplary results are presented in Table 3.

**Table 1 :** Descriptions of the data sets.

Data set	No. attributes	No. examples	No. classes
ACL	7	140	2
Automobile	44	159	6
Bank	6	66	2
Cleveland	14	303	5
Bupa	7	345	2
Ecoli	8	336	8
Glass	10	214	7
HSV	12	122	4
Imidasolium	9	201	5
Lymphography	19	148	4

Meta Data	44	528	5
Pima	9	768	2
Voting	17	300	2
Yeast	9	1484	10
Zoo	17	101	7

**Table 2 :** The comparison of classification accuracies [%] obtained by the combiner and the single classifiers

Data set	k-NN	C4.5	MOD-LEM	Combiner
ACL	84.29	85.00	85.00	84.29
Automobile	76.08	85.50	89.30	84.90
Bank	93.94	92.42	95.45	95.45
Cleveland	52.10	53.14	54.46	54.46
Bupa	63.19	62.32	68.10	69.12
Ecoli	85.71	83.63	80.65	85.42
Glass	68.80	65.42	69.63	71.50
HSV	56.56	51.64	55.74	59.02
Imidasolium	58.21	58.21	60.70	66.67
Lymphography	77.03	80.41	81.08	83.11
Meta-data	49.62	51.33	47.54	51.33
Pima	71.22	67.58	73.30	74.78
Voting	92.41	94.67	93.00	94.67
Yeast	57.80	52.10	54.30	58.36
Zoo	95.05	93.07	92.08	95.05

**Table 3 :** Predictions of base classifiers

Data set	Details of predictions	No. of base classifiers with wrong answers			
		3	2	1	0
ACL	TestPart [%]	11	4	4	81
	Cor-Com [%]	0	0	100	100
	SameClas [%]	100	100	100	-
	2DiffClas [%]	0	0	-	-
	3DiffClas [%]	0	-	-	-
Glass	TestPart [%]	12	18	22	48
	Cor-Com [%]	0	11	100	100
	SameClas [%]	50	28	100	-
	2DiffClas [%]	43	72	-	-
	3DiffClas [%]	7	-	-	-
HSV	TestPart [%]	26	16	26	32
	Cor-Com [%]	0	6	100	100
	SameClas [%]	75	78	100	-
	2DiffClas [%]	21	22	-	-
	3DiffClas [%]	4	-	-	-
Imidasolium	TestPart [%]	20	18	26	36
	mCor-Com [%]	0	27	100	100
	SameClas [%]	72	62	100	-
	2DiffClas [%]	22	38	-	-
	3DiffClas [%]	6	-	-	-
Yeast	TestPart [%]	25	19	22	34
	Cor-Com [%]	0	11	100	100
	SameClas [%]	65	65	100	-
	2DiffClas [%]	35	35	-	-
	3DiffClas [%]	10	-	-	-

Due to the format of this paper, we used the following abbreviations in the table: *TestPart* – a percentage of the testing set; *CorCom* – percentage of these examples

correctly classified by the combiner; *SameClas* – a percentage of testing examples with all base classifiers indicating the same decision class; *2DiffClas* – a percentage of examples with base classifiers indicating two different decisions classes; *3Diff-Clas* – a percentage of examples with each base classifier indicating different decision class. We present representative results only.

To analyse these predictions, consider for example data set *Imidasolium*. We can observe that: for 20% of all examples none of the base classifiers answered correctly and the combiner never answered correctly; for 18% of all examples only one of the base classifiers answered correctly and the combiner answered correctly in 27% of such cases (i.e. for 5% of all examples); for 26% of all examples two of the base classifiers answered correctly and the combiner always answered correctly; 35% of examples were correctly classified by all base classifiers, as well as by the combiner. Moreover, one can notice that when none of the base classifiers answered correctly, in 72% of cases (i.e. for 14.5% of all examples) all base classifiers indicated the same (wrong) decision class and that in 22% of cases (i.e. for 4.5% of all examples) they indicated two different decision classes, etc. Finally, in Table 4 we present data sets where different multiple classifiers were used — we summarise some of previous results obtained for bagging and  $n^2$  classifiers [27]. The symbol “-” denotes that calculations were not performed for the  $n^2$  classifier, most often because the data set did not contain more than two decision classes.

**Table 4 :** Classification accuracies [%] for different multiple classifiers

Data set	Bagging	$n^2$ -classifier	Combiner
Automobile	83.00	87.90	84.90
Bank	95.22	-	95.45
Bupa	76.28	-	69.12
Ecoli	85.70	81.34	85.42
Glass	74.82	74.82	71.50
HSV	64.75	-	59.02
Meta-data	-	49.80	51.33
Pima	75.78	-	74.78
Voting	93.33	-	94.67
Yeast	-	55.74	58.36
Zoo	93.89	94.46	95.05

## VII. Discussion and Final Remarks

Let us start by discussing the results of the experiments, where the classification performance of the combiner was compared against the use of the single classifiers being its components.

- The first observation is that the combiner strategy did not improve the classification accuracy so much as the previously studied bagging and  $n^2$  classifier. The differences of accuracies between the combiner and

the best single classifier were *insignificant* for 10 of 15 data sets. The improvements were observed for 4 data sets: *HSV*, *imidasolium*, *lymphography*, *yeast* and the significant decrease for 1 data set (*Automobile*). We should be cautious to say general conclusions about their characteristics but it seems the improvements were observed for rather difficult data sets, where all base classifiers were rather weak (accuracy close to 50%, except *lymphography*). Easy problems, as e.g. *ACL*, *bank*, *voting* could not be improved by the multiple classifier. Looking into the accuracies of single, base classifiers we can also notice that single MODLEM induced classifier performed quite well comparing to others.

- A more detailed analysis of relationships among predictions of base classifiers leads to some interesting observations. The combiner often performed well when the data set contained many examples for which exactly *two* out of three base classifiers answered correctly. First, however, let us note that the combiner never misclassified an example which was correctly classified by all three of base classifiers. On the other hand, it never correctly classified an example which was misclassified by *all three* of base classifiers.
- Therefore, it is interesting to analyse what happened in case of examples on which a subset of base classifiers made a mistake. First, let us point out that there were some data sets (for example, *ACL*), which contained a high number of examples for which all base classifiers provided *the same, wrong* decision class (see Table 3). For such data sets the combiner usually failed to provide the correct final classification, even for cases where only two base classifiers were wrong. Clearly, in those sets there was not much chance of improvement, as the combiner did not have enough information to correct base classifiers' mistakes — it appears that the decisions of the base classifiers were strongly correlated.
- On the other hand, for those data sets on which we observed significant improvement of prediction accuracy (e.g. *yeast*, *imidasolium*), we noticed different distributions of base predictions (again, see Table 3). Generally, when base classifiers misclassified examples, they more often indicated two or three *different* decision classes. So, the base classifiers were more independent and each seemed to be more specialised for some subsets of examples. In those cases the main assumption of uncorrelated errors seemed to be better founded, thus giving more chance to the combiner strategy to learn meta-level corrections.

Comparing all considered multiple classifiers together (see Table 4) we should be cautious as the number of the results on common data sets is limited. It that none of the

multiple classifiers is the best for all data sets, each of the classifiers has its own area of superiority. The relative merits of the particular approaches depend on the specificity of analysed classification problems.

Other issues to be considered are costs of constructing the particular multiple classifier. In general, we could expect that they need more computations than the standard single classifier. In case of bagging the additional costs mainly depends on the number of base classifiers. However, previous results [27] have showed that the use of MODLEM algorithm in the  $n^2$  classifier has not increased the computational costs, unlike some of other learning algorithms. So, MODLEM is particularly well suited for this multiple classifier – the reasons were discussed in [27, 28]. For the combiner there are additional costs, which depend on the learning algorithms used for base classifiers and for the metaclassifier, as well as on costs of constructing extra examples in the metalearning set.

To sum up, our results show that the rule induction algorithm MODLEM can be efficiently used within the framework of different multiple classifiers for data sets concerning more "complex" decision concepts.

## References

- [1] C. Blake, E. Keogh, C.J. Mertz, Repository of Machine Learning, University of California at Irvine 1999.
- [2] L. Breiman, Bagging predictors. *Machine Learning*, 24 (2), 1996, 123–140
- [3] P.K. Chan, S.J. Stolfo, A comparative evaluation of voting and metalearning on partitioned data. In *Proceedings of the 12th International Conference on Machine Learning*, San Francisco, 1995, 90–98.
- [4] P.K. Chan, S. Stolfo, On the accuracy of metalearning for scalable data mining. *Journal of Intelligent Information Systems*, 8 (1), 1997, 5–28.
- [5] K.J. Cherkauer, Human ExpertLevel Performance on a Scientific Image Analysis Task by a System Using Combined Artificial Neural Networks, *Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, 1996, 15–21.
- [6] T.G. Dietterich, Ensemble methods in machine learning. In: J. Kittler, F. Roli (eds.), *Proceedings of 1st International Workshop on Multiple Classifier Systems*, LNCS vol. 1857, Springer Verlag, 2000, 1–15.
- [7] U. M. Fayyad, G. PiatetskyShapiro, P. Smyth, R. Uthurusamy, From data mining to knowledge discovery. In *Advances in Knowledge Discovery and Data Mining*, 1996, 1–36
- [8] J. Gama, Combining classification algorithms, PhD Thesis, University of Porto, 1999.
- [9] J.W. GrzymalaBusse, LERS — a system for learning from examples based on rough sets. In: Slowinski R. (ed.) *Intelligent Decision Support*.

- Handbook of Applications and Advances of the Rough Sets Theory*. Kluwer, 1992, 3–18.
- [10] J.W. GrzymalaBusse, Managing uncertainty in machine learning from examples. In: *Proceedings 3rd International Symposium in Intelligent Systems*, Wigry, Poland, IPI PAN Press, 1994, 70–84.
- [11] J.W. GrzymalaBusse, J. Stefanowski, Three approaches to numerical attribute discretization for rule induction. *International Journal of Intelligent Systems*, 16 (1), 2001, 29–38.
- [12] L. Hansen, P. Salamon, Neural network ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (10), 1990, 993–1001.
- [13] T. Hastie, R. Tibishirani, Classification by pairwise coupling. In: *Advances in Neural Information Processing Systems 10*, (NIPS97), MIT Press, 1998, 507–513.
- [14] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley, 2004.
- [15] J. Jelonek, J. Stefanowski, Experiments on solving multiclass learning problems by the  $n^2$  classifier. In: *Proceedings of 10th European Conference on Machine Learning ECML 98*, LNAI vol. 1398, Springer Verlag, 1998, 172–177.
- [16] W. Klossgen, J.M. Żytkow (eds.), *Handbook of Data Mining and Knowledge Discovery*, Oxford Press, 2002.
- [17] C. Merz, Combining Classifiers Using Correspondence Analysis. In: *Advances in Neural Information Processing Systems*, vol. 10, 1998, 33–58.
- [18] R.S. Michalski, I. Bratko, M. Kubat (eds.), *Machine learning and data mining*, John Wiley & Sons, 1998.
- [19] D. Mitchell, C.J. Spiegelhalter, C. Taylor, *Machine Learning, Neural and Statistical Classification*, 1994.
- [20] T.M. Mitchell, *Machine learning*, 1997.
- [21] S. Nowaczyk, J. Stefanowski, Experimental evaluation of classifiers based on combiner strategy. Institute of Computing Sciences, Poznan University of Technology, *Research Report RA001/02*, March 2002.
- [22] S.H. Park, J. Furnkranz, Efficient pairwise classification. In: J.N. Kok et al. (eds.) *Proceedings 18th European Conference on Machine Learning ECML 2007*, LNAI vol. 4701, SpringerVerlag, 2007, 658–665.
- [23] Z. Pawlak, *Rough sets. Theoretical aspects of reasoning about data*. Kluwer Academic Publishers, Dordrecht, 1991.
- [24] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo CA, 1993.
- [25] J. Stefanowski, The rough set based rule induction technique for classification problems. In: *Proceedings of 6th European Conference on Intelligent Techniques and Soft Computing EUFIT'98*, Aachen 7–10 Sept. 1998, 109–113.
- [26] J. Stefanowski, *Algorithms of rule induction for knowledge discovery*. (In Polish), Habilitation Thesis published as Series Rozprawy no. 361, Poznan University of Technology Press, Poznan 2001.
- [27] J. Stefanowski, The bagging and  $n^2$  classifiers based on rules induced by MODLEM. In: *Proceedings of the 4th Int. Conference Rough Sets and Current Trends in Computing*, RSCTC'2004, Uppsala, Sweden, June 2004, LNAI vol. 3066, SpringerVerlag, 2004, 488–497.
- [28] J. Stefanowski, On combined classifiers, rule induction and rough sets. *Transactions on Rough Sets*, 6, LNAI, vol. 4374 Journal Subline, SpringerVerlag, 2007, 329350.
- [29] G. Valentini, F. Masuli, Ensembles of Learning Machines. In: R. Tagliaferri, M. Marinaro (eds), *Neural Nets WIRN Vietri-2002*, Springer-Verlag LNCS, vol. 2486, 2002, 3–19.