The bagging and n^2-classifiers based on rules induced by MODLEM

Jerzy Stefanowski

Institute of Computing Science
Poznań University of Technology, 60-965, Poznań, Poland
Jerzy.Stefanowski@cs.put.poznan.pl

Abstract. An application of the rule induction algorithm MODLEM to construct multiple classifiers is studied. Two different such classifiers are considered: the bagging approach, where classifiers are generated from different samples of the learning set, and the n^2-classifier, which is specialized for solving multiple class learning problems. This paper reports results of an experimental comparison of these multiple classifiers and the single, MODLEM based, classifier performed on several data sets.

1 Introduction

Classification is one of common tasks performed in knowledge discovery and machine learning. It includes assigning a decision class label to a set of unclassified objects described by a fixed set of attributes. Different learning algorithms can be applied to induce various forms of classification knowledge from provided learning examples. This knowledge can be successively used to classify new objects. In this sense learning process results in creating *classification system* – shortly called *classifier*. Typical measure used to evaluate such systems is a *classification accuracy*, i.e. a percentage of correctly classified testing examples.

Recently, there has been observed a growing interest in increasing classification accuracy by integrating different classifiers into one composed classification system. In proper circumstances such a composed system could better classify new (or testing) examples than its component classifiers used independently. Experimental evaluations have confirmed that the use of multiple classifiers leads to improving classification accuracy in many problems. The author and his cooperators have also carried research within this framework, e.g. introducing the architecture of the n^2-classifier [10], examining the influence of the choice of learning algorithms or attribute selection on the performance of the composed classification system [10, 11, 15].

In this paper we consider classifiers based on *rules induced from examples*. A number of various algorithms have already been developed to induce such rules, for a review see, e.g., [15, 17]. Although different rule based classifiers have been proved to be efficient for several learning problems, they may not led to satisfactory high classification accuracy for some other, more difficult, data sets. Therefore, it is interesting to check how the performance of rule based classifiers could be improved by using the framework of multiple classifiers.

The main aim of this paper is to examine the application of the rule induction algorithm, called MODLEM, in two different multiple classifiers: the bagging approach and the n^2-classifier. The bagging is an approach that combines homogeneous classifiers generated from different distributions of learning examples. The n^2-classifier is a more specialized approach to solve multiple class learning problems by using a set of binary classifiers, which are responsible for distinguishing between pairs of classes. The rule induction algorithm MODLEM has been previously introduced by the author in [13].

In this study we want to check experimentally how much both multiple classifiers could improve the classification accuracy compared against the single rule classifier. Moreover, we will evaluate the computational costs of creating these combined classifiers. Finally, we would like to discuss properties of learning problems which could be essential for an efficient usage of both multiple classifiers.

2 Multiple classifiers – general issues

The combined classification system (multiple classifier) is a set of single classifiers whose individual predictions are combined in some way to classify new objects. Combining identical classifiers is useless. The member classifier should have a substantial level of disagreement, i.e. they make error independently with respect to one another [4]. As to combining classification predictions from single classifiers, generally, there are either *group* or *specialized decision making* [6]. In the first method all base classifiers are consulted to classify a new object while the other method chooses only these classifiers whose are expertised for this object. *Voting* is the most common method used to combine single classifiers. The vote of each classifier may be weighted, e.g., by the posterior probability referring to its performance on the training data.

The integration of single classifiers into one composed system has been approached in many ways, for some review see, e.g. [4, 6, 14, 15]. In general, one can distinguish two categories: using either *homogeneous* or *heterogeneous classifiers*. In the first category, the same learning algorithm is used over different samples of the data set. Particular attention has been put to *bagging* and *boosting* approaches that manipulate the training data in order to generate diversified classifiers. In a case of heterogeneous classifiers, one can apply a set of different learning algorithms to the same data set and their predictions could be combined, e.g., by *stacked generalization* or *meta-learning*.

3 The bagging approach

The *Bagging* approach (**B**ootstrap **agg**regat**ing**) was introduced by Breiman [2]. It aggregates by voting classifiers generated from different bootstrap samples. The *bootstrap sample* is obtained by uniformly sampling with replacement objects from the training set. Each sample has the same size as the original set, however, some examples do not appear in it, while others may appear more than once. For a training set with m examples, the probability of an example being selected at

least once is $1-(1-1/m)^m$. For a large m, this is about 1 - $1/e$. Each bootstrap sample contains, on the average, 63.2% unique examples from the training set.

Given the parameter T which is the number of repetitions, T bootstrap samples $S_1, S_2, \ldots, S_T$ are generated. From each sample S_i a classifier C_i is induced by the same learning algorithm and the final classifier C^* is formed by aggregating T classifiers. A final classification of object x is built by a uniform voting scheme on $C_1, C_2, \ldots, C_T$, i.e. is assigned to the class predicted most often by these sub-classifiers, with ties broken arbitrarily. The approach is presented briefly below. For more details see [2].

(**input** LS learning set; T number of bootstrap samples; LA learning algorithm
output C^* classifier)
begin
  for $i = 1$ **to** T **do**
  begin
    S_i := bootstrap sample from LS; {sample with replacement}
    $C_i := LA(S_i)$; {generate a sub-classifier}
  end; {end for}
  $C^*(x) = \arg\max_{y \in K_j} \sum_{i=1}^{T} (C_i(x) = y)$ {the most often predicted class}
end

Experimental results presented in [2, 3, 12] show a significant improvement of classification accuracy while using decision tree classifiers. For more theoretical discussion on the justification of bagging the reader is referred to [2].

4 The n^2-classifier

This kind of multiple classifier is a specialized approach to solve *multiple class learning problems*. Although the standard way to solve multi-class learning problems includes the direct use of the multiple class learning algorithm such as, e.g. algorithm for inducing decision trees, there exist more specialized methods dedicated to this problem, e.g., one-per-class method, distributed output codes classification schemes, error-correcting techniques and they can outperform the direct use of the single multiclass learning algorithms, see e.g. discusions [10, 14].

The n^2-classifier is composed of $(n^2 - n)/2$ base binary classifiers (where n is a number of decision classes; $n > 2$). The main idea is to discriminate each pair of the classes: (i, j), $i, j \in [1..n], i \neq j$, by an independent binary classifier C_{ij}. Each base binary classifier C_{ij} corresponds to a pair of two classes i and j only. Therefore, the specificity of the training of each base classifier C_{ij} consists in presenting to it a subset of the entire learning set that contains only examples coming from classes i and j. The classifier C_{ij} yields a binary classification indicating whether a new example $\mathbf{x}$ belongs to class i or to class j. Let us denote by $C_{ij}(\mathbf{x})$ the classification of an example $\mathbf{x}$ by the base classifier C_{ij}.

The complementary classifiers: C_{ij} and C_{ji} (where $i, j \in < 1 \ldots n >$; $i \neq j$) solve the same classification problem – a discrimination between class i-th and

j-th. So, they are equivalent ($C_{ij} \equiv C_{ji}$) and it is sufficient to use only (n^2 - n)/2 classifiers $C_{ij}(i < j)$, which correspond to all combination of pairs of n classes.

An algorithm providing final classification assumes that a new example $\mathbf{x}$ is applied to all base classifiers C_{ij}. As a result, their binary predictions $C_{ij}(\mathbf{x})$ are computed. The final classification should be obtained by a proper aggregation of these predictions. The simplest aggregation rule is based on finding a class that wins the most pairwise comparisons. The more sophisticated approach, considered in this paper, includes a *weighted* majority voting rules, where the vote of each classifier is modified by its credibility, which is calculated as its classification performance during learning phase; more details in [10].

The quite similar approach was independently introduced by Friedman [5]. Then it was extended and experimentally studied by Hastie and Tibshirani [9], which called it *classification by pairwise coupling*. The experimental studies, e.g. [5, 9, 10], have shown that such multiple classifiers perform usually better than the standard classifiers. Previously the author and J.Jelonek have also examined how the choice of a learning algorithm influences the classification performance of the n^2-classifier [10]. Additionally, they have considered different approaches of attribute selection for each pairwise binary classifier [11].

5 The rule induction by the MODLEM algorithm

The rule induction algorithm, called MODLEM, has been introduced by Stefanowski in [13], see also its more precise description in [15] or [8]. Due to the size of this paper we skip the formal presentation of this algorithm and we only discuss its main idea. It 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 examples). 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 as little as possible of them). The main procedure for rule induction scheme starts from creating a first rule by choosing sequentially the 'best' elementary conditions according to chosen criteria (i.e., the first candidate for the condition part is one elementary condition; If it does not fulfill the requirement to be accepted as a rule, then the next - currently best evaluated - elementary condition is added to the candidate condition part, etc.; This specialization is performed until the rule could be accepted). When the rule is stored, all learning positive examples that match this rule are removed from consideration. The process is iteratively repeated while some significant positive examples of the decision concept remain still uncovered. Then, the procedure is sequentially repeated for each set of examples from a succeeding decision concept. In the basic version of the MODLEM algorithm elementary conditions are evaluated by using one of two measures either *class entropy* or *Laplace accuracy* [13, 15]. It is also possible to consider a lexicographic order of two criteria measuring the rule positive cover and then its conditional probability (originally considered by Grzymala in his LEM2 algorithm or its last, quite interesting modification called MLEM2).

The extra specificity of the MODLEM algorithm is handling directly numerical attributes during rule induction while elementary conditions of rules are created, without any preliminary discretization phase [8]. In MODLEM elementary conditions are represented as either $(a < v_a)$ or $(a \geq v_a)$, where a denotes an attribute and v_a is its value. If the same attribute is chosen twice while building a single rule, one may also obtain the condition $(a = [v_1, v_2))$ that results from an intersection of two conditions $(a < v_2)$ and $(a \geq v_1)$ such that $v_1 < v_2$. For nominal attributes, these conditions are $(a = v_a)$. For more details about the function finding best elementary conditions see, e.g., [8, 13].

Finally, the unordered set of induced rules is applied to classify examples using the classification strategy introduced by Grzymala in LERS system [7], which takes into account strength of all rules completely matched and also allows partially matches if no rule fits the description of the tested example.

6 Experiments

The first aim of experiments is to check how much two different techniques discussed in this paper, could increase a classification accuracy of the rule classifier induced by the MODLEM algorithm. Although we can expect such an improvement, we want to evaluate its amount and compare both approaches. Thus, on several benchmark data sets the use of the single rule based classifier is compared against the bagging classifier and the n^2 classifier, which include sub-classifiers also trained in an appropriated way by MODLEM. The second aim of this experiment is to evaluate the computational time of creating these multiple classifiers. We would like to verify whether the potential classification improvement is not burden with too high costs.

The MODLEM algorithm is used with the entropy measure to choose elementary conditions. All experiments are performed on the benchmark data sets, which are coming either from Machine Learning Repository at the University of California at Irvine [1] or from author's case studies, see [15]. Due to the paper size we skip their detailed characteristics . The classification accuracy is estimated by a *stratified* version of 10-*fold cross-validation technique*, i.e. the training examples are partitioned into 10 equal-sized blocks with similar class distributions as in the original set.

In this paper we partly summarize some results already obtained by the author in his preliminary studies (bagging [16] and n^2 [15]). However, we extend them by new data sets. Furthermore, we add new elements concerning evaluation of computational costs. Let us remarks the due to the specifies of each multiple classifier the sets of data are not identical for each of them. The bagging is a more universal approach to create an efficient classifier. Therefore, we used a few "easier" data sets (e.g., *iris*, *bank* or *buses*), where standard, single classifiers are expected to be sufficient and a larger number of more difficult data sets (having different characteristics – the choice was done according to the number of objects and characteristics of attributes). We also took into account some multiple-class learning problem, to compare with another multiple classifiers.

For the n^2-classifier - which is a specialized approach for multiple-class learning problems - we considered a set of multiple-class data; Here the choice is inspired by our earlier experiments with this kind of classifier [10, 11].

Table 1. Comparison of classification accuracies [%] obtained by the single MODLEM based classifier and the bagging approach

Name of dataset	Single MODLEM	Bagging - with different T 3	5	7	10
bank	93.81 ± 0.94	95.05 ± 0.91	94.95 ± 0.84	95.22 ± 1.02	93.95* ± 0.94
buses	97.20 ± 0.94	98.05* ± 0.97	99.54 ± 1.09	97.02* ± 1.15	97.45* ± 1.13
zoo	94.64 ± 0.67	93.82* ± 0.68	93.89* ± 0.71	93.47 ± 0.73	93.68 ± 0.70
hepatitis	78.62 ± 0.93	82.00 ± 1.14	84.05 ± 1.1	81.05 ± 0.97	84.0 ± 0.49
iris	94.93 ± 0.5	95.13* ± 0.46	94.86* ± 0.54	95.06* ± 0.53	94.33* ± 0.59
automobile	85.23 ± 1.1	82. 98 ± 0.86	83.0 ±0.99	82.74 ±0.9	81.39 ± 0.84
segmentation	85.71 ± 0.71	86.19* ± 0.82	87.62 ± 0.55	87.61 ± 0.46	87.14 ± 0.9
glass	72.41 ± 1.23	68.5 ± 1.15	74.81 ± 0.94	74.25 ± 0.89	76.09 ± 0.68
bricks	90.32* ± 0.82	90.3 * ± 0.54	89.84* ± 0.65	91.21* ± 0.48	90.77* ± 0.72
vote	92.67 ± 0.38	93.33* ± 0.5	94.34 ± 0.34	95.01 ± 0.44	96.01 ± 0.29
bupa	65.77 ± 0.6	64.98* ± 0.76	76.28 ± 0.44	70.74 ± 0.96	75.69 ± 0.7
election	88.96 ± 0.54	90.3 ± 0.36	91.2 ± 0.47	91.66 ± 0.34	90.75 ± 0.55
urology	63.80 ± 0.73	64.8 ± 0.83	65.0 ± 0.43	67.40 ± 0.46	67.0 ± 0.67
german	72.16 ± 0.27	73.07* ± 0.39	76.2 ± 0.34	75.62 ± 0.34	75.75 ± 0.35
crx	84.64 ± 0.35	84.74* ± 0.38	86.24 ± 0.39	87.1 ± 0.46	89.42 ± 0.44
pima	73.57 ± 0.67	75.78* ± 0.6	74.35* ± 0.64	74.88 ± 0.44	77.87 ± 0.39

While creating the bagging classifier, we have to tune the parameter T being the number of bootstrap samples and sub-classifiers. We have decided to check it experimentally, as the literature review has not given clear conclusions. Inspired by good results obtained by Quinlan for small numbers of T (for decision trees [12]), we examined the following values of T: 3, 5, 7 and 10. The results of these experiments are given in Table 1. For each dataset, the first column shows the classification accuracy obtained by a single classifier over the 10 cross-validations. Standard deviation is also given. The next columns contain results for the bagging classifiers with changing the number of sub-classifiers. An asterisk indicates that difference for these compared classifiers and a given data set are not statistically significant (according to two-paired t-Student test).

The experiments with the n^2-classifier were performed on 11 data sets, all concerning multiple-class learning problems. The number of classes varies from 3 up to 14. The MODLEM algorithm was again used to create sub-classifiers from subsets of learning examples coming from each pair of classes. Classification accuracies are presented in Table 2 - the second and third columns (presented in a similar way as in Table 1).

Then, let us move to the discussion of computation costs for each multiple classifier. An extra computation time for the bagging is easy to evaluate. If T

Table 2. Comparison of classification accuracies [%] and computation times [s] for the single MODLEM based classifier and the n^2-classifier also based on decision rules induced by MODLEM algorithm

Name of data set	Accuracy of single MODLEM (%)	Accuracy of n^2_{MODLEM} (%)	Time of comput. MODLEM	Time of comput. n^2_{MODLEM}
automobile	85.25 ± 1.3	87.96 ± 1.5	15.88 ± 0.4	5.22 ± 0.3
cooc	55.57 ± 2.0	59.30 ± 1.4	4148,7 ± 48.8	431.51 ± 1.6
ecoli	79.63 ± 0.8	81.34 ± 1.7	27.53 ± 0.5	11.25 ± 0.7
glass	72.07 ± 1.2	74.82 ± 1.4	45.29 ± 1.1	13.88 ± 0.4
hist	69.36 ± 1.1	73.10 ± 1.4	3563.79 ± 116.1	333.96 ± 0.8
meta-data	47.2 ± 1.3	49.83 ± 1.9	252.59 ± 78.9	276.71 ± 5.21
iris	94.2 ± 0.6	95.53* ± 1.2	0.71 ± 0.04	0.39 ± 0.04
soybean-large	91.09 ± 0.9	91.99* ± 0.8	26.38 ± 0.3	107.5 ± 5.7
vowel	81.81 ± 0.5	83.79 ± 1.2	3750.57 ± 30.4	250.63 ± 0.7
yeast	54.12 ± 0.7	55.74 ± 0.9	1544.3 ± 13.2	673.82 ± 9.4
zoo	94.64 ± 0.5	94.46* ± 0.8	0.30 ± 0.02	0.34 ± 0.12

classifiers are generated, than the approach requires approximately T times the computational effort of learning the single classifier by the MODLEM algorithm. The construction of the n^2-classifier is a more interesting case. In our previous works [10, 11] we noticed that the increase of classification accuracy (for other learning algorithms than MODLEM) is burden with increasing the computational costs (sometimes quite high). Here, for using MODLEM, the results are just opposite. Table 2 (two last columns) contains results of computation times (average value over 10 folds with standard deviations). Let us remark that all calculations have been performed on the same PC machine.

7 Discussion of results and final remarks

First, let us discuss results of the experiments for each multiple classifier.

The bagging classifier significantly outperformed the single classifier on 11 of 16 data. The differences between compared classifiers were non-significant for 3 data sets (*buses*, *iris* and *bricks*) and the single classifier won only for *zoo* and *automobile*. We could comments that the worse performance of the bagging classifier occurred for rather "easier" data (characterized by a linear class separation). However, the bagging was a winner for more difficult problems. One can also notice the slightly worse performance of the bagging for quite small data (e.g. *buses*, *zoo* - which seemed to be too small for sampling), while it much improved for data sets containing the higher number of examples. Considering the number of sub-classifier T, it seems to be difficult to determine the best one value. For majority of data sets, the highest accuracy was obtained for T equal to 7 or 10. For few data set we have performed additional experiments with increasing T up to 20 [16]. However, we have not observed an improvement, except *glass* and *pima*.

The results obtained for the n^2-classifier indicate a significant improvement of the classification accuracy for the majority of multiple-class learning problems (7 of 11). Again, the multiple classifier was not useful for easier problems (e.g. *iris*). The differencies between compared classifiers were not significant for smaller number of examples. Moreover, similarly for using the bagging, the data set *zoo* was "too diffcult" - it was the only data, where the single classifier was slightly better than the n^2-classifier. Coming back to our previous results for the n^2-classifier [10] we can remark that the comparable classification improvements were observed for the case of using decision trees.

Comparing results of both multiple classifier should be very cautious as we had a quite limited number of common data sets. It seems that the n^2-classifier, which is in fact a specialized approach to learning multiple classes, is slightly better – compare results for *auto*, *glass* and even *zoo*. However, we should perform more experiments on a larger number of data sets.

The analysis of computation costs leads us to quite intriguing observation on using the MODLEM algorithm within the n^2-classifier. Generally, using it does not increase the computation time. What is even more astonishing, for the majority data sets (8 of 11) constructing the n^2-classifier requires even less time (from 2 up to 10 times less) than training the standard single classifier. However one should not be puzzled about this observation. Let us first remind the idea behind pairwise classification. Friedman argues [5] that the general all classes learning methods are limited in that for each there are broad classes of ("complex", non-linear) decision concepts with which they have difficulty. Even for universal approximators the learning sample size may place such limits. However, each pairwise decisions is more likely to be a simpler function of input attributes. This is especially when each decision class is well separated from most of the others. So, pairwise decision boundaries between each pair of classes could be simpler and can be quite often aproximated with linear functions while for the standard multiple class approach the decision boundary could be more complicated and more difficult to learn, e.g. with non-linear approximators.

Here, let us remind that for each of n decision classes the MODLEM algorithm sequentially generates the set of rules discriminating positive examples of the given class from all negative examples belonging to *all* other (n - 1) classes. So, besides the more complex decision boundaries (as discussed above), the computation time of this algorithm may also increase with the higher number of examples and classes. In the case of n^2-classifier the task is simpler, as it is sufficient to find these elementary condition which discriminate *two* classes only. Intuitively, we could expect that much smaller number of attributes is sufficient to distinguish a pair of classes. Moreover, having a smaller number of examples from two classes, the number of different attribute values should also be smaller (therefore, a smaller number of conditions is tested while inducing rules). This hypothesis is somehow confirmed by a detailed analysis of the characteristics of rule sets induced by the single standard classifier and the n^2-classifier. For instance, for *ecoli* data the MODLEM algorithm (used as a standard multiple-class approach) induced 46 rules, which contain totally 171 elementary conditions (on

average 3.7 per each rule); Each rule covers on average 9.3 examples. The n^2-classifier contains 118 rules (for all binary sub-classifier) using 217 conditions (on average 1.8 per rule); However each rule covers 26.5 examples! Similar observations have been made for the many of other data sets. It seems that in our experiments creating subspaces of attributes dedicated for discriminating pairs of classes has been more efficient than using the same set of attributes for distinguishing all decision classes at the same time.

The bagging classifier needs more computations and the additional costs depends on T - the number of sub-classifiers. Coming back to the expected improvement of classification accuracy, the bagging is more general approach than n^2 specialized for multiple-classes. Simiarly to the n^2-classifier, bagging also works better for more "complex/non-linear" decision concepts. We could expect it as according to Breiman the bagging should be constructed with *unstable* learning algorithms, i.e. ones whose output classifier undergoes major changes in response to small changes in learning data. Similar to decision tree inducers the algorithm MODLEM is the unstable algorithm in the sense of this postulate.

To sum up, the results of our experiments have shown that the MODLEM algorithm can be efficiently used within the framework of two considered multiple classifiers for data sets concerning more "complex" decision concepts. The n^2-classifier is particularly well suited for multiple class data where exist "simpler" pairwise decision boundaries between pairs of classes. However, the relative merits of these new approaches depends on the specifies of particular problems and a training sample size.

Let us notice that there is a disadvantage of the multiple classifiers - loosing a simple and easy interpretable structure of knowledge represented in a form decision rules. These are ensembles of diversified rule sets specialized for predictive aims not one set of rules in a form for a human inspection.

For future research, it could be interesting to consider yet another techniques for aggregating predictions from sub-classifier. In particular it concerns the n^2-classifier, whose sub-classifiers are trained to distinguish particular pairs of classes only. Therefore, they could be excluded (or weaken) from voting for examples likely coming from different classes.

References

1. Blake C., Koegh E., Mertz C.J.: Repository of Machine Learning, University of California at Irvine (1999).
2. Breiman L.: Bagging predictors. Machine Learning, 24 (2), (1996) 123–140.
3. Bauer E., Kohavi R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36 (1/2), (1999) 105–139.
4. Dietrich T.G.: Ensemble methods in machine learning. In: Proc. of 1st Int. Workshop on Multiple Classifier Systems, (2000) 1–15.
5. Friedman J.: Another approach to polychotomous classification, Technical Report, Stanford University (1996).
6. Gama J.: Combining classification algorithms. Ph.D. Thesis, University of Porto (1999).

7. Grzymala-Busse J.W.: Managing uncertainty in machine learning from examples. In: Proc. 3rd Int. Symp. in Intelligent Systems, Wigry, Poland, IPI PAN Press, (1994) 70–84.
8. Grzymala-Busse J.W., Stefanowski J.: Three approaches to numerical attribute discretization for rule induction. International Journal of Intelligent Systems, 16 (1), (2001) 29–38.
9. Hastie T., Tibshirani R., Classification by pairwise coupling. In: Jordan M.I. (eds.) Advances in Neural Information Processing Systems: 10 (NIPS-97), MIT Press, (1998) 507-513.
10. Jelonek J., Stefanowski J.: Experiments on solving multiclass learning problems by the n^2-classifier. In: Proceedings of 10th European Conference on Machine Learning ECML 98, Springer LNAI no. 1398, (1998) 172–177.
11. Jelonek J., Stefanowski J.: Feature selection in the n^2-classifier applied for multiclass problems. In: Proceedings of the AI-METH 2002 Conference on Artificial Intelligence Methods, Gliwice, (2002) 297-301.
12. Quinlan J.R.: Bagging, boosting and C4.5. In: Proceedings of the 13th National Conference on Artificial Intelligence, (1996) 725–730.
13. Stefanowski J.: 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.
14. Stefanowski J.: Multiple and hybrid classifiers. In: Polkowski L. (ed.) Formal Methods and Intelligent Techniques in Control, Decision Making, Multimedia and Robotics, Post-Proceedings of 2nd Int. Conference, Warszawa, (2001) 174–188.
15. Stefanowski J.: Algorithims of rule induction for knowledge discovery. (In Polish), Habilitation Thesis published as Series Rozprawy no. 361, Poznan Univeristy of Technology Press, Poznan (2001).
16. Stefanowski J.: Bagging and induction of decision rules. In: Int. Symposium on Intelligent Systems; Post-Proceedings of the IIS'2002. Series: Advances of Soft Computing, Physica Verlag, Heidelberg, (2002) 121-130.
17. Klosgen W., Żytkow J.M. (eds.): Handbook of Data Mining and Knowledge Discovery, Oxford Press (2002).