UDC 004.021

A Comparison of the Bagging and the Boosting Methods Using the Decision Trees Classifiers

Kristína Machová[1], Miroslav Puszta[1], František Barčák[1], and Peter Bednár[1]

[1] Department of Cybernetics and Artificial Intelligence, Technical University, Letná 9, 04200 Košice, Slovakia
Kristina.Machova@tuke.sk, Miroslav.Puszta@accenture.com, frantisek.barcak@accenture.com, Peter.Bednar@tuke.sk

Abstract. In this paper we present an improvement of the precision of classification algorithm results. Two various approaches are known: bagging and boosting. This paper describes a set of experiments with bagging and boosting methods. Our use of these methods aims at classification algorithms generating decision trees. Results of performance tests focused on the use of the bagging and boosting methods in connection with binary decision trees are presented. The minimum number of decision trees, which enables an improvement of the classification performed by the bagging and boosting methods, was found. The tests were carried out using the Reuter's 21578 collection of documents as well as documents from an Internet portal of TV broadcasting company Markíza. The comparison of our results on testing the bagging and boosting algorithms is presented.

1. Introduction

The precision of classification depends on many aspects, for example on the selection of a suitable classification algorithm for a given task, on the selection of a training set and so on. In frame of this paper, we have focused on experiments with training set samples, with the aim to improve the precision of classification results. At present, two various approaches are known. The first approach is based on an idea of making various samples of the training set. A classifier is generated for each of these training set samples by a selected machine learning algorithm. In this way, for *k* variations of the training set we get *k* particular classifiers. The result will be given as a combination of individual particular classifiers. This method is called Bagging in the literature [1]. Another similar method called Boosting [6], [7] performs experiments over training sets as well. This method works with weights of training examples. Higher weights are assigned to incorrectly classified examples. That means, that the importance of these examples is emphasized. After the weights of training sets are updated, a new (particular) classifier is generated. A final classifier is calculated as a combination of

particular classifiers. The presented paper focuses on the bagging and boosting method in combination with Decision trees in the role of particular classifiers. We decided to base our experiments with bagging and boosting on the text categorisation task.

2. Bagging

Bagging is a method for improving results of classification algorithms. This method was formulated by Leo Breiman and its name was deduced from the phrase "**b**ootstrap **agg**regat**ing**" [1]. More information about bagging can be found in [3], [4] and [9].

In case of classification into two possible classes, a classification algorithm creates a classifier H: D → {-1,1} on the base of a training set of example descriptions (in our case played by a document collection) D. The bagging method creates a sequence of classifiers H_m, m=1,…,M in respect to modifications of the training set. These classifiers are combined into a compound classifier. The prediction of the compound classifier is given as a weighted combination of particular classifier predictions according to:

$$H(d_i, c_j) = sign\left(\sum_{m=1}^{M} \alpha_m H_m(d_i, c_j)\right) \quad (1)$$

The meaning of the above given formula can be interpreted as a voting procedure. An example d_i is classified to the class c_j for which the majority of particular classifiers vote. Articles [2] and [6] describe the theory of classifier voting. Parameters α_m, m=1,…,M are determined in such way that more precise classifiers have stronger influence on the final prediction than less precise classifiers. The precision of base classifiers H_m can be only a little bit higher than the precision of a random classification. That is why these classifiers H_m are called weak classifiers.

We experimented with the following bagging algorithm [1]:

1. Initialisation of the training set *D*
2. for *m = 1, ..., M*

2.1. Creation of a new set D_m of the same size $|D|$ by random selection of training examples from the set *D* (some of examples can be selected repeatedly and some may mot be selected at all).

2.2. Learning of a particular classifier H_m: $D_m \rightarrow R$ by a given machine learning algorithm based on the actual training set D_m.

3. Compound classifier *H* is created as the aggregation of particular classifiers H_m*: m = 1, ...,M* and an example d_i is classified to the class c_j in accordance with the number of votes obtained from particular classifiers H_m. according to the formula (**1**).

If it is possible to influence the learning procedure performed by the classifier H_m directly, classification error can be minimized also by H_m while keeping parameters α_m constant.

The above-described algorithm represents an approach called **base version of bagging**. There are some other strategies called **bagging like strategies,** which divide the original training set into N subsets of the same size. Each particular classifier is learned using one of these subsets. A compound classifier is created as the aggregation of particular classifiers. The most known methods are: disjoint partitions, small bags, no replication small bags and disjoint bags. An illustrative example of the subset selection process to form new training subsets from an original one is presented in the rest of this section. The example of the original training set containing sixteen training examples is depicted in Fig. 1.

Fig. 1. Original training set D

The method of **disjoint partitions** uses random selection of examples. Each example is selected only once. An example of four new subsets, created from the original training set in Fig. 1 is presented in Fig. 2. In general, if N subsets are created from the original training set, then each of them contains 1/N part from the original set. The union of particular subsets equals the original training set. For very large original set, partitions enable parallel learning of base classifiers.

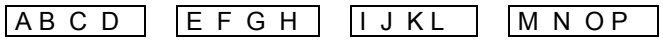

Fig. 2. Disjoint partitions

Classifier H obtained from the aggregation of particular classifiers H_m learnt on “disjoint partitions”, achieves the best results from all „bagging like strategies“.

In the method of **small bags**, each subset is generated independently from the other subsets by random selection of training examples with the possibility to select an example repeatedly. An example can be located in several subsets and/or several times in one subset as well. The training sets illustrated in Fig. 3 were obtained from the original set in Fig. 1. Union of particular partitions in the method “small bags” does not guarantee to provide the original training set. Classifiers using the “small bags” reach the worst results from all „bagging like strategies“.

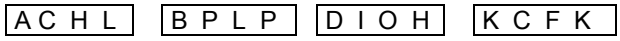

Fig. 3. Small bags

In the method of **no replication small bags**, each subset is generated independently from the other subsets by random selection of training

examples without any replication of examples in the same subset. An example can occur in one subset, several subsets, or no subset. The training sets illustrated in Fig. 4 were obtained from the original set in Fig. 1. Union of particular partitions does not guarantee to represent the original training set.

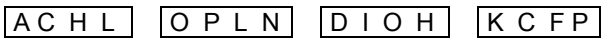

Fig. 4. No replication small bags

The last method from the above mentioned ones is the method of **disjoint bags**. In this method, size of each subset does not equal 1/|*D*| but is (slightly) greater. First, examples, which occur in the original training set, are distributed into subsets. Selection of training examples is performed in the same way as in the method of "disjoint partitions". Then, one or more examples are randomly selected and replicated within each subset. If it occurs in a subset, then exactly one copy is included in this subset. The number of replications has to be the same in each subset. An example of resulting division of training examples is illustrated in Fig. 5.

Fig. 5. Disjoint bags

Union of particular partitions does not provide the original training set. Classifiers using "disjoint bags" are known to reach the same or sometimes better results as those classifiers using „disjoint partitions".

3. Boosting

Similarly as bagging algorithm, the boosting method creates a sequence of classifiers H_m, m=1,…,M in respect to modifications of the training set. These classifiers are combined into a resulting classifier according to the formula (**1**). The training set is modified by a weight distribution over individual training examples - documents $d_i \in D$. The set of weights is assigned uniformly before learning of the first classifier. For each iteration, the weights of training examples, which were classified incorrectly by the previous classifier $H_{m-1,}$ are increased. The weights of those training examples, which were classified correctly, are decreased. In this way, the learning of next classifier focuses on incorrectly classified training examples.

We do not consider as necessity to present all boosting algorithms, we experimented with. Only the AdaBoost.MH2 [7] algorithm will be presented in this paper. This algorithm represents a generalization of the basic form of the algorithm for multiple classifying into more than two classes. This algorithm creates classifiers H_m: DxC→R, which define prediction for each class $c_j \in C$. Similarly to the classifying into two classes, H classifies documents into a class $c_j \in C$ according to the decision function sign[$H(d_j, c_j)$]. The difference

from basic algorithm is following: the weight distribution is assigned to combinations of training examples and classification classes.

We experimented with the following boosting algorithm:

1.Initialise weight distribution $w_{1(i,j)} = 1 / (|D||C|)$, $i = 1, ..., |D|$, $j = 1, ..., |C|$

2.For pre $m = 1, ..., M$

2.1.Create a classifier $H_m: D \times C \rightarrow R$ using a given algorithm for actual weight distribution $w_{m(i,j)}$.

2.2.Determine parameter $\alpha_m \in R$.

2.3.Modify the weight distribution according to the rule

$$w_{m+1(i,j)} = \frac{w_{m(i,j)} \exp(-\alpha_m y_{i,j} H_m(d_i, c_j))}{Z_m} \quad (\mathbf{2})$$

where Z_m is a normalisation constant ensuring that $\sum_{i=1}^{|D|} \sum_{j=1}^{|C|} w_{m+1(i,j)} = 1$ holds.

3. The output is the decision function of the final classifier in the form of the formula (**1**):

Variable $y_{i,j}$ is defined as $y_{i,j} = +1$ if $d_i \in c_j$ and as $y_{i,j} = -1$ if $d_i \notin c_j$.

In our experiments we used a modified version of this algorithm. The advantage of this modified version is that weight calculation does not depend on precision of calculations, and there are no problems with number rounding. Therefore, this algorithm is suitable for document classification, which this paper is devoted to.

4. Text categorization

We decided to base our experiments with bagging and boosting on the text categorization task [8]. The aim is to find an approximation of an unknown function $\Phi: D \times C \rightarrow \{true, false\}$ where D is a set of documents and $C = \{c_1, ..., c_{|C|}\}$ is a set of predefined categories. The value of the function Φ for a pair $\langle d_i, c_j \rangle$ is true (false) if the document d_i belongs (doesn't belong) to the category c_j. The function $\hat{\Phi}: D \times C \rightarrow \{true, false\}$, which approximates Φ, is called a classifier. Definition of the text categorization task is based on these additional suppositions:

- Categories are only nominal labels and there is no (declarative or procedural) knowledge about their meaning.
- Categorisation is based solely on knowledge extracted from text of the documents.

This definition is a general one and does not require availability of other resources. The constraints may not hold in operational conditions when any

kind of knowledge can be used to make the process of categorization more effective.

Based on a particular application it may be possible to limit the number of categories for which the function Φ has the value true for a given document d_i. If the document d_i can be classified exactly into one class $c_j \in C$, C represents the set of disjoint classes. The case, when each document can be classified into an arbitrary number $k = 0, ..., |C|$ of classes from the set C, is called multiple classification and C represents the set of overlapping classes.

Binary classification represents a special case when a document can be classified into one of two classes. Algorithms for binary classification can be used for multiple classifications as well. If classes are independent from each other (i.e. for each pair of classes c_j, c_k, $j \neq k$ holds that the value $\Phi(d_i, c_j)$ is independent from the value $\Phi(d_i, c_k)$), the problem of multiple classification can be decomposed into $|C|$ independent binary classification problems into classes $\{c_i, \bar{c}_i\}$ for $i = 0, ..., |C|$. In this case a classifier for the category c_j stands for the function $\hat{\Phi}_j : D \rightarrow \{true, false\}$, which approximates the unknown function $\Phi_j : D \rightarrow \{true, false\}$.

With respect to the above mentioned decomposition, we used binary decision tree [5] (decision tree performing binary classification) in the role of a particular (base) classifier.

5. Classifier efficiency evaluation

The evaluation of classifier efficiency can be based on the degree of match between prediction $\hat{\Phi}(d_i, c_j)$ and actual value $\Phi(d_i, c_j)$ calculated over all documents. Quantitatively it is possible to evaluate the effectiveness in terms of precision and recall (similarly to evaluating methods for information retrieval). For classification of documents belonging to the class c_j it is possible to define precision π_j as conditional probability:

$Pr(\Phi(d_i, c_j) = true \mid \hat{\Phi}(d_i, c_j) = true)$. Similarly, recall ρ_j can be defined as conditional probability $Pr(\hat{\Phi}(d_i, c_j) = true \mid \Phi(d_i, c_j) = true)$. Probabilities π_j and ρ_j can be estimated from a contingence Table 1. , according to (**3**).

$$\pi_j = \frac{TP_j}{TP_j + FP_j}, \quad \rho_j = \frac{TP_j}{TP_j + FN_j} \tag{3}$$

where TP_j and TN_j (FP_j and FN_j) are the numbers of correctly (incorrectly) predicted positive and negative examples of the class c_j.

Overall precision and recall for all classes can be calculated in two ways. Micro averaging is defined in (**4**).

Table 1. Contingence table for category cj

	$\Phi(d_i, c_j) = true$	$\Phi(d_i, c_j) = false$
$\hat{\Phi}(d_i, c_j) = true$	TP_j	FP_j
$\hat{\Phi}(d_i, c_j) = false$	FN_j	TN_j

$$\pi^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{j=1}^{|C|} TP_j}{\sum_{j=1}^{|C|} (TP_j + FP_j)} \qquad (4)$$

$$\rho^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{j=1}^{|C|} TP_j}{\sum_{j=1}^{|C|} (TP_j + FN_j)}$$

While, macro averaging is given by the following equations (**5**):

$$\pi^{M} = \frac{\sum_{j=1}^{|C|} \pi_j}{|C|} \qquad \rho^{M} = \frac{\sum_{j=1}^{|C|} \rho_j}{|C|} \qquad (5)$$

The selection of a particular way of averaging depends on a given task. For example, micro averaging reflects mainly classification of cases belonging to frequently occurring classes, while macro averaging is more sensitive to classification of cases from less frequent classes.

Precision and recall can be combined into one measure, for example according to the formula (**6**):

$$F_{\beta} = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho} \qquad (6)$$

Where, β expresses trade off between F_{β} and π and ρ. Very often it can be seen the use of the function F_1 combining precision and recall using equal weights.

6. Experiments

A series of experiments was carried out using a binary decision tree as a base classifier. Data from two sources were employed. The first one was the *Reuter's-21578*[1] document collection, which comprises Reuter's documents from 1987. The documents were categorised manually. To experiment, we used a XML version of this collection. The collection consists of 674 categories and contains 24242 terms. The documents were divided

[1] Most experiments were carried out using this document collection, if not stated otherwise.

into a training and test sets – the training sets consists of 7770 documents and 3019 forms the test set. After stemming and stop-words removal, the number of terms was reduced to 19864.

The other document collection, used to perform experiments, was formed by documents from an Internet portal of the Markiza broadcasting company. The documents were classified into 96 categories according to their location on the Internet portal www.markíza.sk. The collection consists of 26785 documents in which 280689 terms can be found. In order to ease experiments, the number of terms was reduced to 70172. This form of the collection was divided into the training and test sets using ratio 2:1. The training set is formed by 17790 documents and the test one by 8995 documents. Documents from this collection are in the Slovak language unlike the first collection, which is in English language.

In order to create decision trees, the famous C4.5 algorithm was used. This algorithm is able to form perfect binary trees over training examples for each decision category. To test the boosting method, weak classifiers (not perfect) are necessary. Therefore, the trees generated by the C4.5 method were subsequently pruned.

We used a pruning method, which estimates accuracy using the training set for parameter setting. The method is based on the pessimistic error estimation. Namely, C4.5 constructs the pessimistic estimation by calculating standard deviation of estimated accuracy given binomial distribution.

6.1. Experiments with bagging

Bagging efficiency testing. Results achieved by classifiers, based on the bagging algorithm, were compared with those generated by perfect decision trees. Fig. 6 depicts differences between precision of the bagging-based classifier and the precision of the perfect decision tree classifier. Data are shown for each classification class separately (the classes are ordered decreasingly according to their frequency).

The results can be interpreted in such way that the bagging-based method provides worse results than perfect decision trees for more frequent classes. On the other hand, for less frequent classes the results of the perfect decision tree are better.

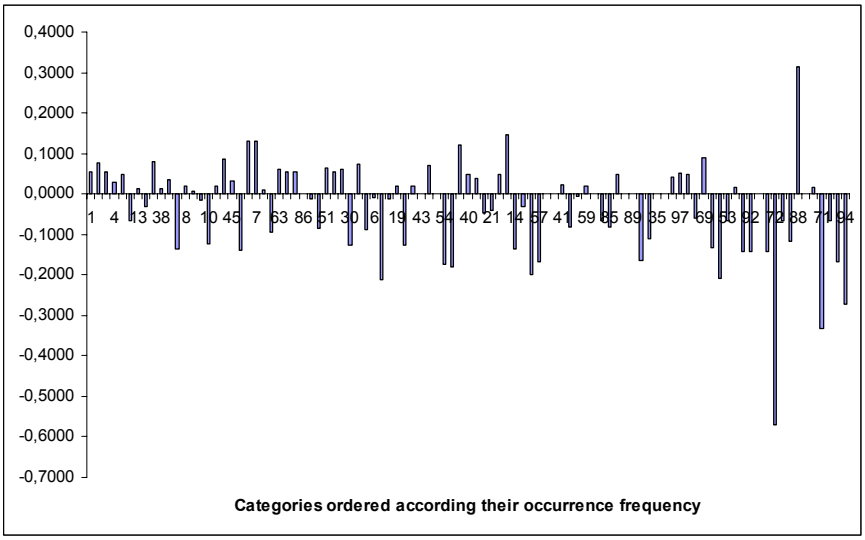

Fig. 6. Precision differences between the bagging-based classifier and the perfect decision tree for data from the Reuter's collection (categories are ordered according to their occurrence frequency; from less frequent classes to more frequent classes)

Experiments with different number of classifiers. In order to explore dependence of the efficiency of the bagging-based classifier on the number of classifiers, additional experiments were carried out. 200 classifiers limited the number of generated binary decision trees in the bagging algorithm. That means each category was classified by a sum of not more than 200 classifiers. Subsequently, the number of used classifiers was reduced and implications on the classifier efficiency were studied. In order to enable comparison with non-bagging classifier, the efficiency of a perfect binary decision tree was represented on the Fig. 7 with a straight line.

The Fig. 7 illustrates that efficiency of the classifiers based on the bagging method does not depend on the quality of particular classifiers (represented by the pruning trees), since the values are almost the same for every pruning method. As far as different parameters are concerned, bagging is superior in respect to precision for the number of used classifiers greater than 20. Using 20 or more classifiers, the F1 measure is practically constant. Considering recall, the situation slightly differs. The value of the recall parameter increases with using bigger number of classifiers – with the threshold value 40 classifiers approximately.

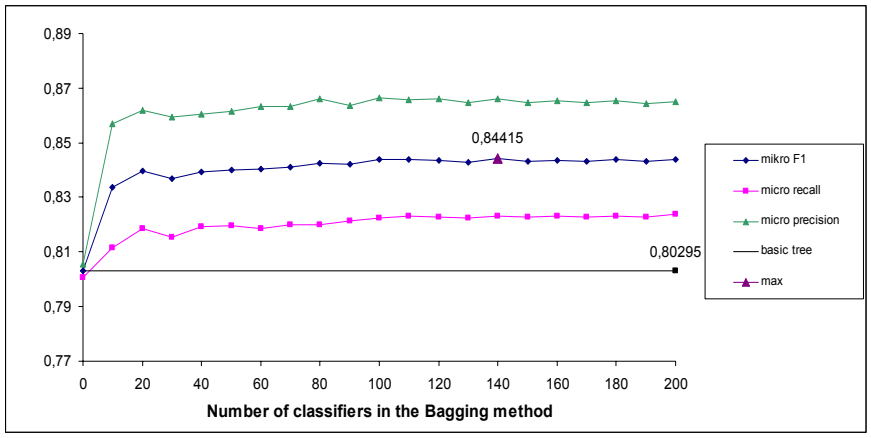

Fig. 7. Efficiency differences between the bagging-based classifiers and a perfect decision tree for data from the Reuter's collection

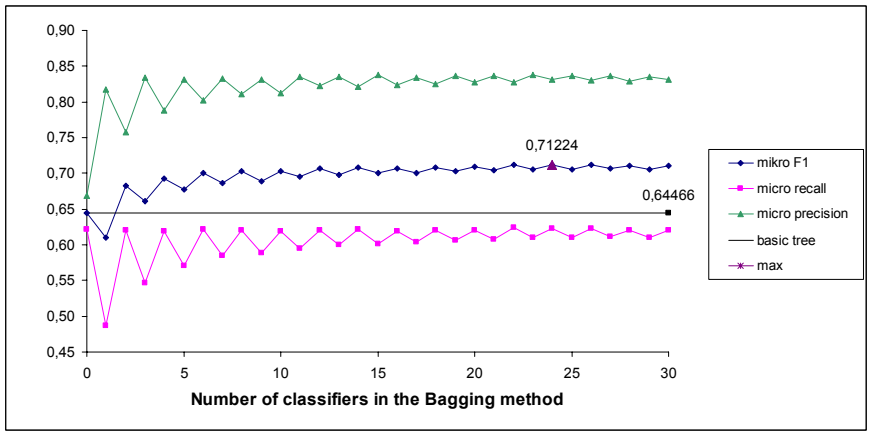

Fig. 8. Efficiency differences between the bagging-based classifiers and a perfect decision tree for data from the Markiza collection

Similar experiments were carried out using data from the Internet portal of the Markiza broadcasting company. The results are illustrated on Fig. 8. The same parameter setting was used for both the bagging-based classifier and the decision tree classifier. The Fig 8 illustrates that as far as different parameters are concerned, the bagging method is superior for the number of classifiers greater than 10 (approximately).

6.2. Experiments with boosting

Boosting efficiency testing. Experiments have proven that one of the best classifiers, based on the boosting algorithm, is the one for generating decision trees with pruning on confidence level CF=0.4. Results achieved by this classifier were compared with those generating by perfect decision trees. Fig. 9 depicts differences between precisions of the boosting classifier and the perfect decision tree for data from the Reuter's collection. Data are shown for each classification class separately (the classes are ordered decreasingly according to their frequency).

Similar experiments were carried out using data from the Internet portal of the Markiza company. The results are illustrated on Fig. 10. The same parameter setting was used for both the boosting based classifier and decision tree classifier.

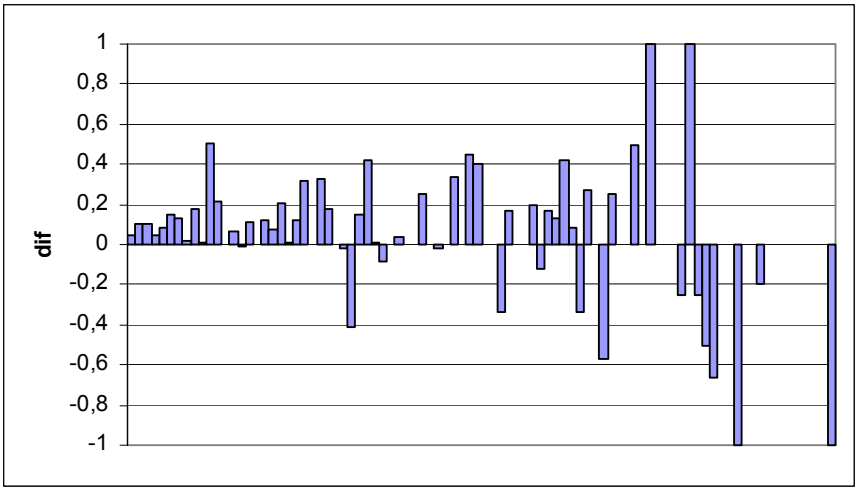

Fig. 9. Precision differences between boosting-based classifier and a perfect decision tree for data from the Reuter's collection

The results can be interpreted in such a way, that the boosting method provides better results while for frequent classes the difference is minimal.

Experiments with different number of classifiers. In order to explore dependence of boosting classifier efficiency on the number of classifiers, additional experiments were carried out for different ways of pruning. First, a set of classifiers with different pruning values was trained. The number of generated binary decision trees of the boosting algorithm was limited by 100 classifiers. That means each category was classified by a weighted sum of not more than 100 classifiers. Subsequently, the number of used classifiers was reduced and implications on the classifier efficiency were studied. In

order to enable comparison with non-boosting classifier, the efficiency of a perfect binary decision tree was depicted on the following figures as a broken line.

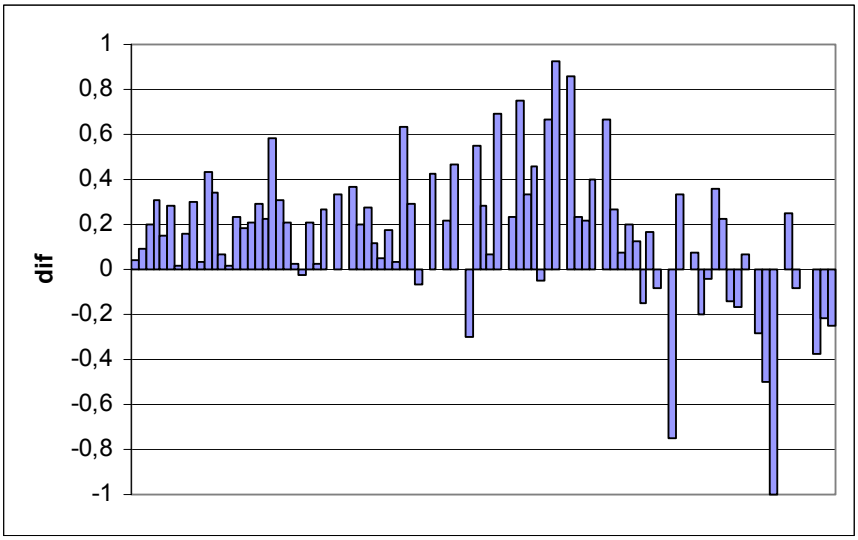

Fig.10. Precision differences between boosting-based classifier and a perfect decision tree for data from the Markiza collection

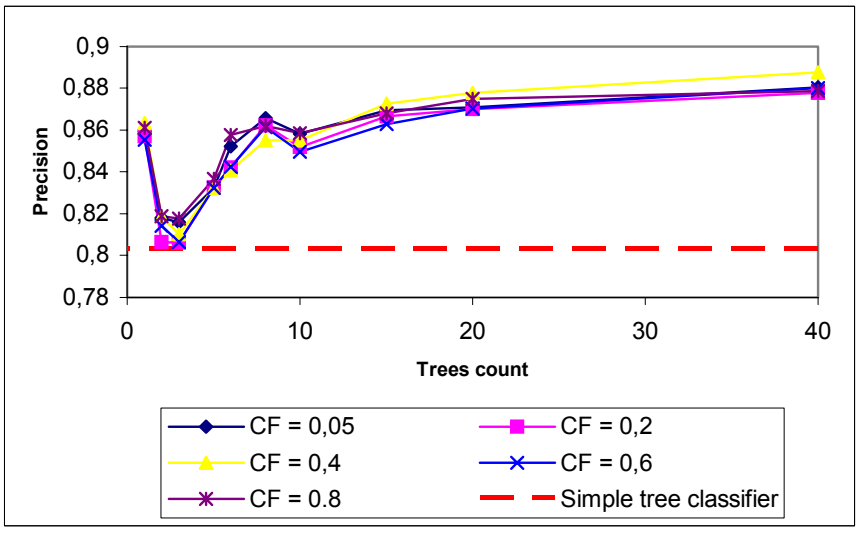

Fig. 11. Relationship between precision and the number of trees in the boosting classifier

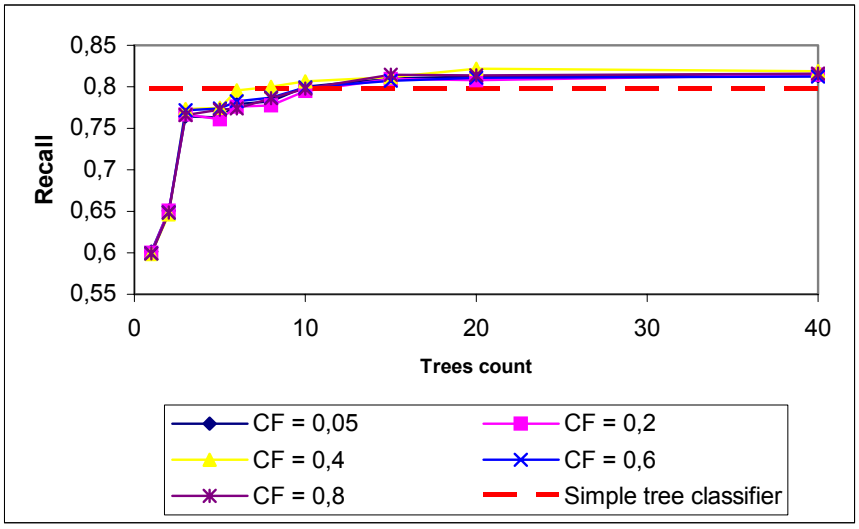

Fig. 12. Relationship between recall and the number of trees in the boosting classifier

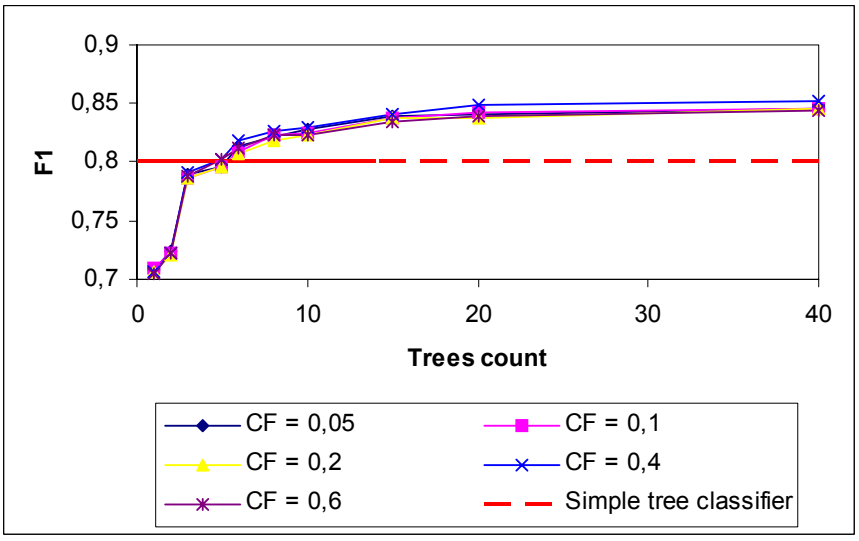

Fig. 13. Relationship between F1 parameter and the number of trees in the boosting classifier

The last three figures illustrate that efficiency of classifiers based on the boosting method does not depend on the quality of particular classifiers (represented by the pruning values), since the graphs are almost the same for every pruning method.

As far as different parameters are concerned, Fig. 13 presents that boosting is superior for the number of classifiers greater than 5. Using 20 or more classifiers, F1 is practically constant and better by 5% than perfect binary tree. Considering precision (Fig. 11), the situation slightly differs. For very small number of classifiers (1 or 2), precision of the boosting-based classifier is better – it proves a hypothesis that precision of decision trees can be increased by pruning. Increasing the number of classifiers implicates decreasing of the precision first (but still better than that of the perfect classifier) with subsequent increasing (up to a constant value around using 35 classifiers). The minimum number of classifiers necessary to achieve successful precision is about 15. Recall is depicted in Fig. 12. Small number of classifiers clearly does not suffice and cannot compete with the perfect binary tree. The value of the recall parameter increases with using bigger number of classifiers – the number 10 was sufficient to compete with perfect tree. The next increase in the number of used classifiers prefers boosting before the perfect tree.

7. Conclusions

In order to draw a conclusion from our experiments, several statements can be formulated. The bagging and boosting algorithms are suitable means to increase efficiency of the classification algorithms, which have low values of precision and recall[2]. Both mentioned parameters could be increased. Considering the same efficiency for a perfect decision tree and bagging-based or boosting-based classifiers, the minimum number of classifiers necessary to achieve this efficiency can be found. Also, The minimum number of classifiers necessary to achieve successful and precision, recall or F1 efficiency of bagging and boosting methods can be found.

It would be suitable to provide some theoretical analysis on the minimum number of classifiers, but in our work, we focused on practical verification, that bagging and boosting algorithms could increase efficiency of the week classifiers. This practical verification was based on finding the minimum number of classifiers necessary to achieve mentioned efficiency.

As far as disadvantages of bagging and boosting are concerned, the loss of simplicity and illustrative-ness of this classification scheme can be observed. Increased computational complexity is a bit discouraging as well.

The comparison of the bagging and boosting methods prefers boosting to bagging. The bagging is a simpler method. The boosting achieves better results. For example, the experiments with bagging efficiency show, that bagging-based classifiers are better than the perfect decision tree only for low occurrence frequency. The same experiments wits boosting-based classifiers gave substantially better results. The experiments with different number of classifiers on the Reuter's-21578 document collection provided the following

[2] Mainly recall for binary trees.

results. For bagging method, the minimum number of classifiers necessary to achieve successful precision is about 20 and to achieve successful recall about 40. For boosting method, the minimum number of classifiers necessary to achieve successful precision is about 15 and to achieve successful recall is only about 10. The best results for successful efficiency of boosting method according to parameter F1 were achieved using at least 5 classifiers. The better results of boosting in this field can be explained by the fact, that this method uses the weighting of training examples. The weights of those training examples, which were classified incorrectly, are increased. It enables the next particular classifier to switch its focus on incorrectly classified training examples.

In the future, we would like to perform some experiments with boosting in combination with some algorithm generating threshold units. Combining boosting and bagging can be interesting too.

8. Acknowledgements

The work presented in this paper was supported by the Slovak Grant Agency of Ministry of Education and Academy of Science of the Slovak Republic within the 1/1060/04 project ”Document classification and annotation for the Semantic web”.

9. References

1. Breiman, L.: Bagging predictors. Technical Report 421, Department of Statistics, University of California at Berkeley, 1994.
2. Breiman, L.: Arcing the edge. Technical report 486, at UC Berkeley, 1996.
3. Friedman, J., Springer, L.: <http://www-stat.stanford.edu/people/faculty/friedman.html>
4. Hastie, T.: <http://www-stat.stanford.edu/%7Ehastie/>Robert
5. Quinlan, J.R.: Bagging, boosting and C4.5. In Proc. of the Fourteenth National Conference on Artificial Intelligence, 1996.
6. Schapire, R., Freund, Y.: Boosting the margin: A new explanation for the efectiveness of voting methods. The annals of statistics, 26(5):1651-1686, 1998.
7. Schapire, R.E., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning, 37(3), 1999, 297-336.
8. Schapire, R.E., Singer, Y.: BoostTexter: A Boosting – based System for Text Categorization. Machine Learning, 39(2/3), 2000, 135-168.
9. Tibshirani, R.: <http://www-stat.stanford.edu/%7Etibs/>Jerome

Kristína Machová graduated (MSc.) with honours in 1985 at the Department of Cybernetics at Technical University in Košice. She defended her PhD. in the field of machine learning in 1996. Since 1985 she is working at the Department of Cybernetics and AI of the Faculty of Electrical Engineering and Informatics at Technical University. Her scientific research is focusing on knowledge based systems, machine learning and meta-learning, basic principles of the learning algorithms, automatic classification of text documents, information retrieval and so on. In addition to this, she also investigates the questions related to the adaptive web and semantic web.

Miroslav Puszta graduated (MSc.) with honours in 2004 at the Department of Cybernetics and Artificial Intelligence at Technical University in Košice. His interests are focusing on machine learning, boosting algorithms, classification of text documents and information retrieval. Now he is working in Accenture Bratislava s.r.o, as analyst with SAP – System Applications and Products in Data Processing.

František Barčák graduated (MSc.) with honours in 2004 at the Department of Cybernetics and Artificial Intelligence at Technical University in Košice. His interests are focusing on bagging methods, machine learning, automatic classification of text documents and information retrieval. Now he is working in Accenture Bratislava s.r.o.

Peter Bednár graduated (MSc.) with honours in 2001 at the Department of Cybernetics at Technical University in Košice. He is an assistant in the Centre for Information Technologies at the Faculty of Electrical Engineering and Informatics. He studied Artificial Intelligence at the Technical University of Košice like PhD student. His theme of the PhD thesis is automatic classification of text documents. His research in artificial intelligence: semantic web, knowledge discovery, knowledge management, ontology engineering, information retrieval, data mining and text-mining. He participated in the HPSE-CT-2001-00065 ProductivityGap, IST-1999-20364 WEBOCRACY, INCO - COPERNICUS GOAL projects and national VEGA projects. He has experiences with developing of applications for semantic web and data and text-mining tasks.