A Bagging Method using Decision Trees in the Role of Base Classifiers

Kristína Machová[1], František Barčák[2] , Peter Bednár[3]

[1]Department of Cybernetics and Artificial Intelligence, Technical University, Letná 9, 04200 Košice, Kristina.Machova@tuke.sk

[2] Department of Cybernetics and Artificial Intelligence, Technical University, Letná 9, 04200 Košice, frantisek.barcak@accenture.com

[3]Department of Cybernetics and Artificial Intelligence, Technical University, Letná 9, 04200 Košice, Peter.Bednár@tuke.sk

Abstract: This paper describes a set of experiments with bagging – a method, which can improve results of classification algorithms. Our use of this method aims at classification algorithms generating decision trees. Results of performance tests focused on the use of the bagging method on binary decision trees are presented. The minimum number of decision trees, which enables an improvement of the classification performed by the bagging method was found. The tests were carried out using the Reuters 21578 collection of documents as well as documents from an internet portal of TV broadcasting company Markíza.

Keywords: classification algorithms, bagging, binary decision trees, text categorisation, recall and precision

1 Introduction

Nowadays, information and data are stored everywhere, mainly on the Internet. To serve us, information had to be transformed into the form, which people can understand, i.e. into the form of knowledge. This transformation represents a large space for various machine learning algorithms, mainly classification ones. The quality of the transformation heavily depends on the precision of classification algorithms in use.

The precision of classification depends on many aspects. Two of most important aspects are the selection of a classification algorithm for a given task and the selection of a training set. 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 [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 emphasised. After the weights are updated, a new (base) classifier is generated. A final classifier is calculated as a combination of base classifiers. The presented paper focuses on the bagging method in combination with Decision trees in the role of base classifiers.

2 Bagging

Bagging is a method for improving results of machine learning 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 individual classifier predictions:

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

The meaning of the above given formula can be interpreted as a voting procedure. An example d_i is classified to the class 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]:
A bagging algorithm for multiple classification into several classes.

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.

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

If it is possible to influence the learning procedure performed by the classifier H_m directly, classification error can be minimised 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 work with smaller size of the training set of example descriptions. These strategies use a combination of the bagging method and the cross-validation method. The **cross-validation** represents the division of the training set into N subsets of D/N size. One of these subsets is used as the training set and the other subsets play the role of test sets.

In "bagging like strategies" the original training set is divided into N subsets of the same size. Each subset is used to create one classifier – a particular classifier is learned using this subset. 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 original training set containing sixteen examples is depicted in Figure 1.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P

Figure 1: Original training set *D*.

The method of **disjoint partitions** uses random selection to select examples. Each example is selected only once. An example of four new subsets, created from the original training set in Figure 1, is presented in Figure 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. Union of particular subsets equals the original training set. For very large original set, partitions enable parallel learning of base classifiers.

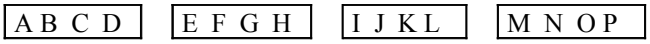

Figure 2: Disjoint partitions.

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 Figure 3 were obtained from the original set in Figure 1. Union of particular partitions does not guarantee to provide the original training set. Classifiers using the small bags reach the worst results from all „bagging like strategies".

Figure 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. An example can occur in one subset, several subsets, or no subset. If it occurs in a subset, then exactly one copy is included in the subset. The training sets illustrated in Figure 4 were obtained from the original set in Figure 1. Union of particular partitions does not guarantee to represent the original training set.

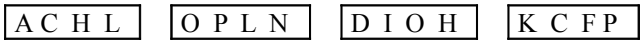

Figure 4: No replication small bags.

The last method from the above mentioned ones is the method of **disjoint bags**. In this method, size of each subset does not equal $|D|$ 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. The number of replications has to be the same in each subset. An example of resulting division of training examples is illustrated in Figure 5. Each example from the original set occurs (once or more times) exactly in one subset.

Figure 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 Text categorisation

We decided to base our experiments with bagging on the text categorisation 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 if the document d_i belongs 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 categorisation 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 categorisation 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 $c_j \in C$, it is the case of the classification into one class and 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. Classifiers (and algorithms for their creation) for binary classification can be used for multiple classification 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 $\{c_i, \overline{c}_i\}$ 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 (decision tree performing binary classification) in the role of a base classifier.

4 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 $d_i \in T$ (or $d_i \in V$). 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 Table 1 in the following way:

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

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.

Table 1. Contingence table for category c_j.

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

Overall precision and recall for all classes can be calculated in two ways. Micro averaging is defined in the following way:

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

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

while macro averaging is given by the following equations:

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

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 following formula:

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

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.

Lacking training data (when it is not possible to select a sufficiently representative test set), it is possible to estimate classification efficiency using cross validation when the set of all examples Ω is divided into k subsets T_1, ..., T_k of the same size. For each subset a classifier $\hat{\phi}_i$ is constructed using Ω - T_i as a training set and T_i in the role of the test set. Final estimation can be calculated by averaging results of classifiers $\hat{\phi}_i$ relevant to all subsets.

5 Experiments

A series of experiments was carried out using binary decision trees as base classifiers. Data from two sources were employed. The first one was the *Reuters-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 into training and test sets – the training set consists of 7770 documents and 3019 documents form 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 the Internet portal of the Markiza broadcasting company. The documents were classified into 96 categories according to their location on the Internet portal www.markiza.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 set by 8995 documents. Documents from this collection are in the Slovak language unlike the first collection, whose documents are written in English.

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

5.1 Bagging efficiency testing

Results achieved by classifiers, based on the bagging algorithm, were compared with those generated by perfect decision trees. Figure 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 their frequency).

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

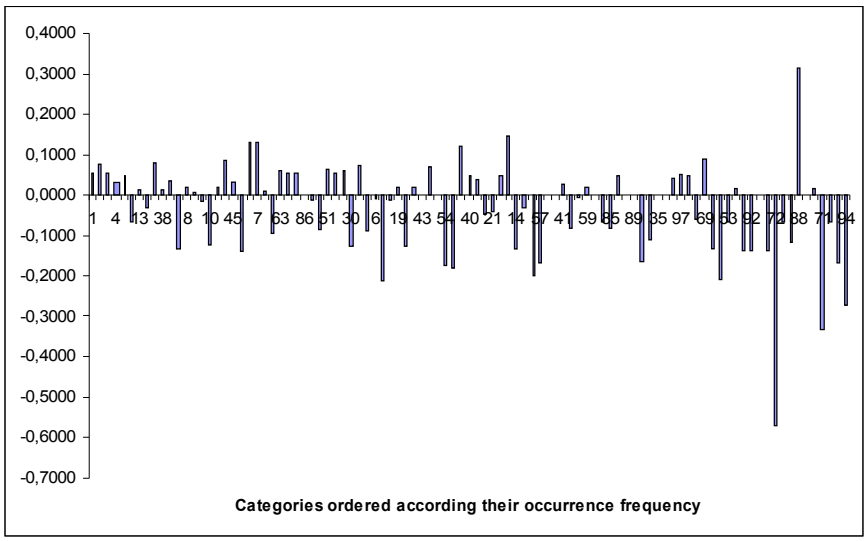

Figure 6 Precision differences between a bagging-based classifier and a perfect decision tree for data from the Reuter's collection.

The results can be interpreted in such way that the bagging-based method provides better 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.

5.1 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. The number of iterations (i.e. the number of generated binary decision trees) of the bagging algorithm was limited by 200 classifiers. 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 Figure 7 as a black line.

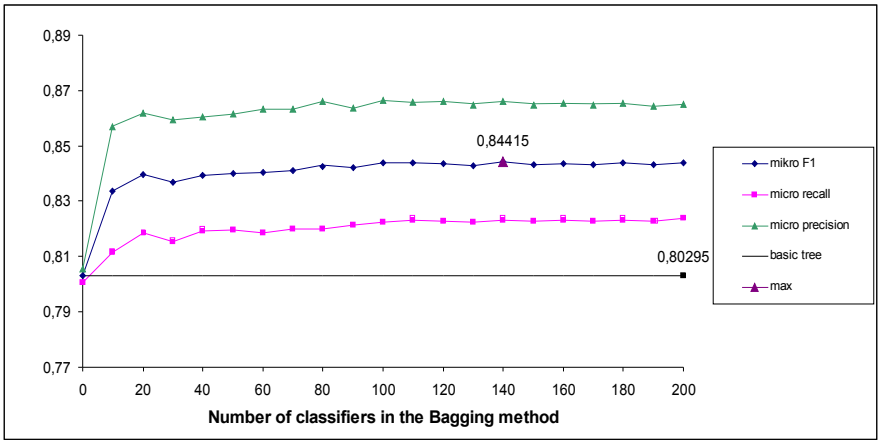

Figure 7: Efficiency differences between the bagging-based classifiers and a perfect decision tree for data from the Reuter's collection.

The Figure 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 values), 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.

Similar experiments were carried out using data from the Internet portal of the Markiza broadcasting company. The results are illustrated on Figure 8. The same parameter setting was used for both the bagging-based classifier and the decision tree classifier.

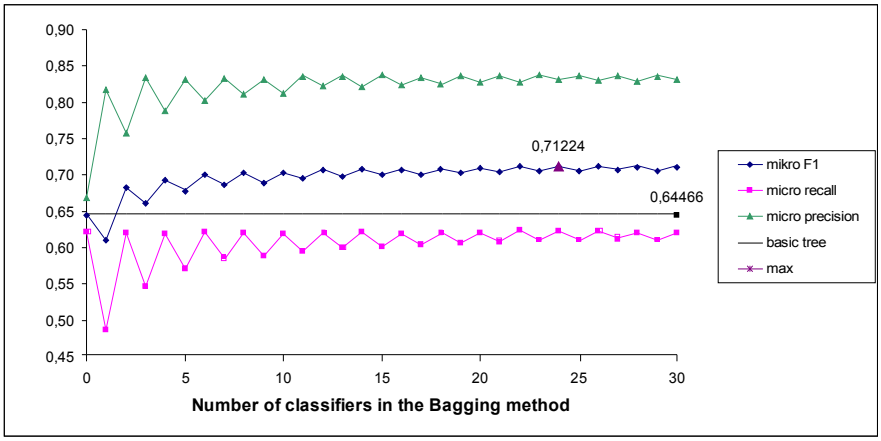

Figure 8 Efficiency differences between the bagging-based classifiers and a perfect decision tree for data from the Markiza collection.

The Figure 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 Conclusion

In order to draw a conclusion from our experiments, several statements can be formulated. The bagging method is a suitable mean for increasing efficiency of standard machine learning algorithms.

Considering the same efficiency for a perfect decision tree and bagging-based classifiers, minimum number of classifiers necessary to achieve this efficiency can be found.

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

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”.

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