A Boosting method in Combination with Decision Trees

Kristína Machová[1], Miroslav Puszta[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, Miroslav.Puszta@accenture.com

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

Abstract: This paper describes boosting – a method, which can improve results of classification algorithms. The use of this method aims at classification algorithms generating decision trees. A modification of the AdaBoost algorithm was implemented. Results of performance tests focused on the use of the boosting method on binary decision trees are presented. The minimum number of decision trees, which enables improvement of the classification performed by a base machine learning algorithm, was found. The tests were carried out using the Reuters 21578 collection of documents as well as documents from an internet portal of TV Markíza.

Keywords: classification algorithms, boosting, binary decision trees, text categorisation

1 Introduction

We live in an information society. Information and data are stored everywhere, mainly on the Internet. To serve us, this 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 apects are the selection of a classification algorithm for a given task and the selection of a training set. Basic principles of machine learning algorithms can be

used in order to select/construct a suitable machine learning algorithm [2]. In the 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* resulting classifers. The result will be given as a combination of individual classifiers. This method is called Bagging in the literature [1]. Another similar method called Boosting [3] performs experiments over training sets as well. This method works with weights of training examples. Higher weights are assigned to uncorectly classified examples. That means, that the importance of these examples is emphasized. After the weights are updated, a new classifier is generated. A final classifier is calculated as a combination of base classifiers. The presented paper focuses on this method.

2 Boosting

Boosting is a method for improving results of machine learning classification algorithms. In case of classification into two possible classes, an algorithm creates on the base of a training set of documents D a classificator H: D → {-1,1}. The boosting method creates a sequence of classifiers Hm, m=1,…,M in respect to modifications of the training set. These classifiers are combined into a resulting classifier. The prediction of the resulting classifier is given as a weighted combination of individual clasifier predictions:

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

Parameters α_m, m=1,…,m are determined in such way, that more precise classifiers influence the resulting prediction more than less precise ones. Precision of base classifiers H_m can be only a little bit higher than precission of a random classification. That is why these classifiers Hm are called weak classifiers.

The training set is modified by a weight distribution over individual documents d_i Є D. The set of weights is assigned uniformly before learning of the first classifier. For each next iteration, the weights of training examples, which were classified uncorectly by the previous clasifier $H_{m-1,}$ are increased. The weights of those training examples, which were classified corectly, are decreased. In this way, the learning of next classifier focuses on uncorectly clasified training examples.

We do not consider to be necessary to present all boosting algorithms, we experimented with. Only the AdaBoost.MH2 algorithm will be presented in this paper. This algorithm represents a generalisation of the basic form of the

algorithm for multiple classification into more than two classes. This algorithm creates clasificators H_m: DxC→R, which define prediction for each class c_j Є C. Similary to the classification into two classes, H classifies documents into a class cj Є C according to decision function sign[H(d_j, c_j)]. The difference from basic algorithm is, that a weight distribution is assigned to combinations of training examples and classification classes.

A boosting algorithm for multiple classification into several classes.

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

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

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:

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

Variable $y_{i,j}$ is defined as $y_{i,j} = +1$ if $d_i \in c_j$ and as $y_{i,j} = -1$ if $d_i \notin c_j$.
Classification error on training examples is bound by the formula:

$$\frac{1}{|D||C|} \sum_{i=1}^{|D|} \sum_{j=1}^{|C|} I(H(d_i, c_j) \neq y_{i,j}) \leq \prod_{m=1}^{M} Z_m$$

From the given limit it is possible to calculate value of α_m for a given classifier H_m similarly as for (Algorithm I) as minimisation of the normalisation constant

$$Z_m = \sum_{i=1}^{|D|} \sum_{j=1}^{|C|} w_{m(i,j)} \exp(-\alpha_m y_{i,j} H(d_i, c_j))$$

If it is possible to influence the learning of the classifier H_m directly, classification error can be minimised also by specifying prediction H_m when keeping parameters α_m constant.

In our experiments we used a modified version of the 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.

3 Text categotization

We decided to base our experiments with boosting on the text categorisation task. 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 is for a pair $\langle d_i, c_j \rangle$ true if document d_i belongs to the category c_j. The learned function $\hat{\Phi} : D \times C \rightarrow \{true, false\}$ which approximates $\hat{\Phi}$ is called a classifier. Definition of text categorisation 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 other resources. These 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 represents 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 classification as well. If classes are independent from each other (i.e. for each class c_j, c_k and $j \neq k$ is the value of $\Phi(d_i, c_j)$ 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 c_j category stands for the function $\hat{\Phi}_j : D \to \{true, false\}$, which approximates unknown function $\Phi_j : D \to \{true, false\}$.
With respect to the abovementioned decomposition, we used binary decision tree in the role of a base classifier.

4 Classifier efficiency evaluation

The evaluation of classifier efficiency can be based on a 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 effectivity in terms of precision and recall (similarly to evaluating methods for information retrieval).
For classification of documents from 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.

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

where TP_j and TN_j (FP_j and FN_j) is the number 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
$\hat{\Phi}(d_i, c_j) = false$	TN_j	FN_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 parameter β expresses trade off between F_{β} and π and ρ. Very often can be seen 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 Ω is divided into k test subsets T_1, ..., T_k.. For each subset a classifier $\hat{\Phi}_i$ is learned using Ω - T_k. as a training set. Final estimation can be calculated by averaging results of classifiers $\hat{\Phi}_i$ relevant to all test subsets. Cross validation can be employed for parameter optimisation instead of validation set.

5 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

Reuters21578[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 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 , whose documents are in English.

In order to create decision trees, the famous C45 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 a pessimistic error estimation. Namely, C4.5 constructs the pessimistic estimation by calculating standard deviation of estimated accuracy given binomial distribution.

5.1 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 perfect decision trees. Figure 1 depicts differences between precisions of the boosting classifier and the perfect decision tree generating one. 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.

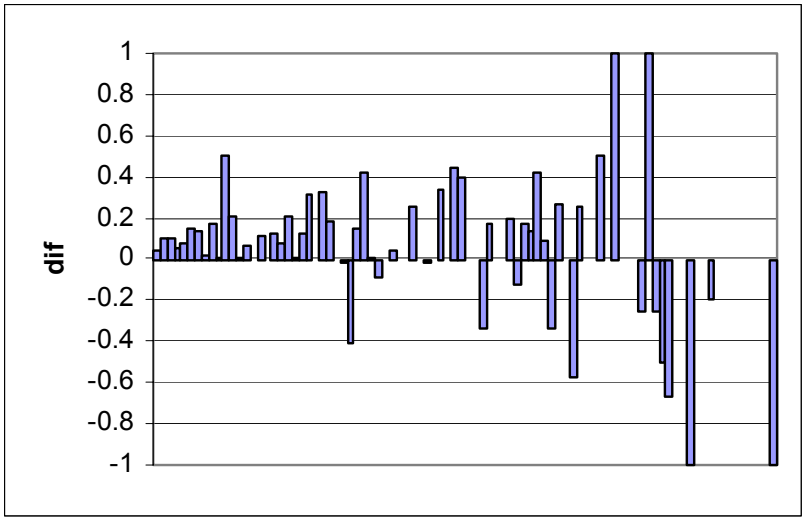

Figure 1 Precision differences between boosting-based classifier and a perfect decision tree for data from the Reuters collection

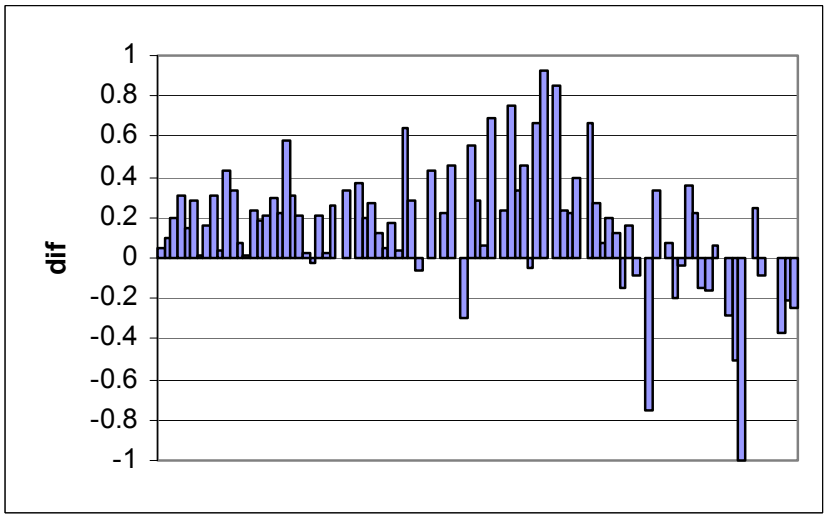

Figure 2 Precision differences between boosting-based classifier and a perfect decision tree for data from the Markiza collection

Similar experiments were carried out using data from the Internet portal of the Markiza company. The results are illustrated on
Figure 2. The same parameter setting was used for both the boosting based classifier and decision tree classifier.
The results can be interpreted in such a way, that the boosting method provides better results while for frequent classes are the difference is minimal.

5.2 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 iterations (i.e. 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 red line.

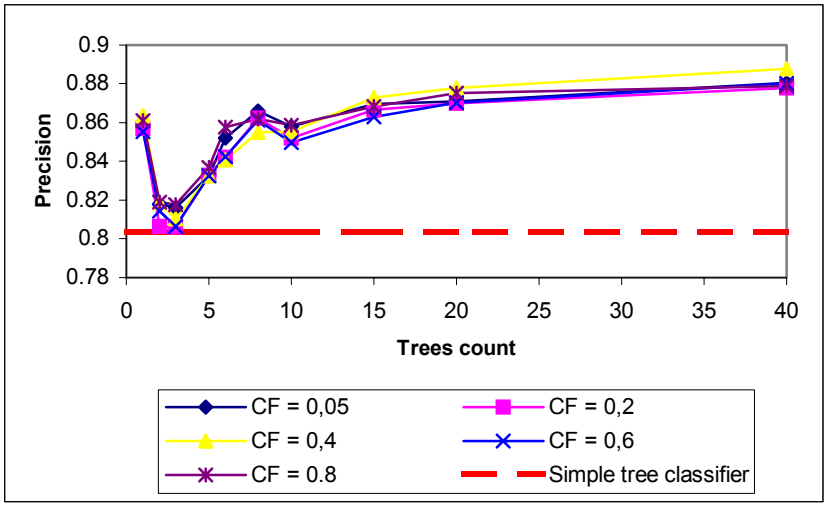

Figure 3 Relationship between precision and the number of trees in the boosting classifier

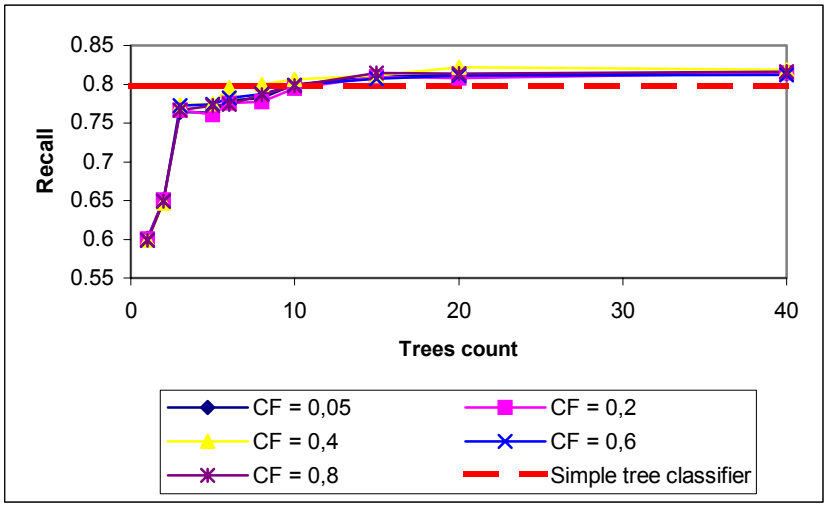

Figure 4 Relationship between recall and the number of trees in the boosting classifier

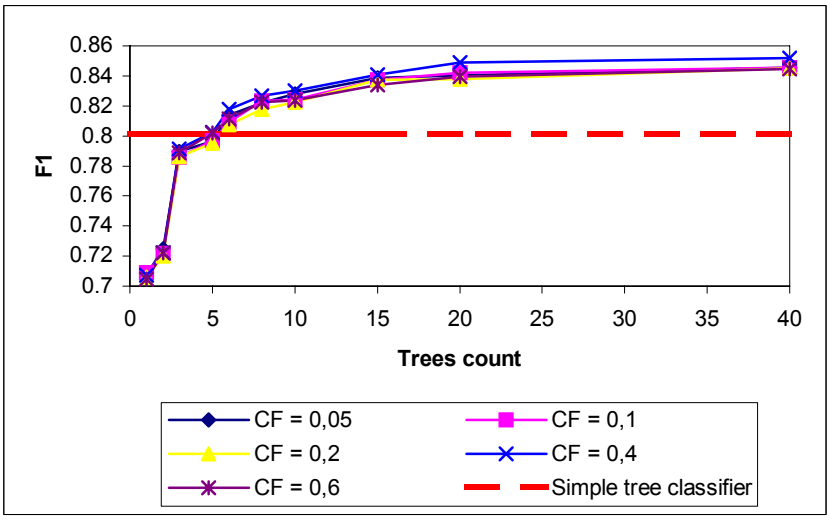

Figure 5 Relationship between ... parameter and the number of trees in the boosting classifier

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.

Figure 5 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 (Figure 3), 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). Recall is depicted in Figure 4. 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 over the perfect tree.

6 Conclusion

In order to draw a conclusion from our experiments, several statements can be formulated. Building binary trees have proven, that their classification quality heavily depends on pruning.

The boosting algorithm is a suitable mean for increasing efficiency of those algorithms with low values of precision and recall[2]. Both these parameters can be increased.

Considering the same efficiency for a perfect tree and boosting (with minimum number of classifiers necessary to achieve this efficiency), it would be possible to compare complexity of both decision schemes. For example, it would be possible to count the number of nodes of the perfect tree and all trees classifying into the same class for boosting.

As far as disadvantages of boosting are considered, the loss of simplicity and illustrative ness 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”.

[2] Mainly recall for binary trees.

References

[1] Breiman, L.: Bagging predictors. *Technical Report 421, Department of Statistics, University of California at Berkeley*, 1994.

[2] Machová, K., Paralič, J.: Basic Principles of Cognitive Algorithms Design. Proc. of the IEEE International Conference Computational Cybernetics, Siófok, Hungary, 2003, 245-247, ISBN 963 7154 175.

[3] Schapire, R.E., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning, 37(3)*, 1999, 297-336.