

4/1 (2006), 131–142

tmcs@inf.unideb.hu
<http://tmcs.math.klte.hu>

Teaching
Mathematics and
Computer Science

An improvement of the classification algorithm results

KRISTÍNA MACHOVÁ, MIROSLAV PUSZTA and PETER BEDNÁR

Abstract. One of the most important aspects of the precision of a classification is the suitable selection of a classification algorithm and a training set for a given task. Basic principles of machine learning can be used for this selection [3]. In this paper, we have focused on improving the precision of classification algorithms results. Two kinds of approaches are known: Boosting and Bagging. This paper describes the use of the first method – boosting [6] which aims at algorithms generating decision trees. A modification of the AdaBoost algorithm was implemented. Another similar method called Bagging [1] is mentioned. 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 and documents from an internet portal of TV Markíza.

Key words and phrases: classification algorithms, boosting, binary decision trees, text categorization.

ZDM Subject Classification: P80, R40.

1. Introduction

Nowadays, information and data are stored 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 involves a number of 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 facts. Two of most important aspects are the selection of a classification algorithm for a given task and the selection of a training set. In this paper, we have focused on experiments with training set samples, with the aim to improve the precision of classification results. At present, two kinds of approaches are known. The first approach is based on an idea of taking different 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 classifiers. 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 [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 are updated, a new classifier is generated. A final classifier is calculated as a combination of base classifiers. The present paper focuses on this method.

2. Boosting

In case of classification into two classes, an algorithm of the boosting creates on the basis of a training set of documents D a classifier $H: D \rightarrow \{-1, 1\}$. The boosting method creates a sequence of classifiers H_m , $m = 1, \dots, M$ 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 classifier predictions:

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

Parameters α_m , $m = 1, \dots, M$ are determined in such way that more precise classifiers influence the resulting prediction more than less precise ones. Precision of the base classifiers $H_m >$ random classification. That is why these classifiers H_m are called weak classifiers. Boosting method tends to over fit in these situations, when the base classifiers are weak. If they were strong, there would be no need for boosting method.

The training set is modified by a weight distribution over individual documents $d_i \in D$. 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 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 the next classifier focuses on incorrectly classified training examples.

We do not intend to present all boosting algorithms, we experimented with. Only the AdaBoost.MH2 [6] 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 classifier $H_m: D \times C \rightarrow \mathbb{R}^{|C|}$, which defines prediction for each class $c_j \in C$. Similarly to the classification into two classes, H classifies documents into a class $c_j \in C$ according to the decision function $\text{sign}[H(d_j, c_j)]$. 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 the weight distribution $w_{1(i,j)} = 1/(|D| \cdot |C|)$, $i = 1, \dots, |D|$, $j = 1, \dots, |C|$.
2. For $m = 1, \dots, M$
 - 2.1. Create the classifier $H_m: D \times C \rightarrow \mathbb{R}$ using a given algorithm for actual weight distribution $w_{m(i,j)}$.
 - 2.2. Determine the parameter $\alpha_m \in \mathbb{R}$, $\alpha_m = \log(1 - r_m/r_m)$, where r_m is the weighting error rate.
 - 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:

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

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

In our experiments we used a modified version of the algorithm. This modification has an implementing character and it solves minor numerical problems. 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.

We decided to base our experiments with boosting on the text categorisation task [7]. The aim is to find an approximation of an unknown function $\Phi: D \times C \rightarrow \{true, false\}^{|C|}$ where D is a set of documents and $C = \{c_1, \dots, c_{|C|}\}$ 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 Φ 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 of 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 $k = 0, \dots, |C|$ 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 pair of classes 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 $\{c_i, \bar{c}_i\}$ for $i = 0, \dots, |C|$. In this case a classifier for c_j category stands for the function $\hat{\Phi}_j: D \rightarrow \{true, false\}$, which approximates unknown function $\Phi_j: D \rightarrow \{true, false\}$. With respect to the above mentioned decomposition, we used binary decision tree [4] in the role of a base classifier.

3. Evaluation of classifiers efficiency

We were using precision and recall for evaluation of classifier efficiency. These measures are widely used in information retrieval, in particular for document retrieval. These two measures together provide more information about the efficiency of multiple classification. For classification of documents from class c_j it is possible to define *precision* π_j as conditional probability $\Pr(\Phi(d_i, c_j) = true | \hat{\Phi}(d_i, c_j) = true)$. Similarly, *recall* ρ_j can be defined as conditional probability $\Pr(\hat{\Phi}(d_i, c_j) = true | \Phi(d_i, c_j) = true)$. Probabilities π_j and ρ_j can be estimated from a contingency 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. The contingency 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

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

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 π and ρ . Possible values for β : $\beta \in [0, \infty]$, where $\beta = 0$ represents π and $\beta = \infty$ represents ρ . Finally $\beta = 1$ represents harmonic average between π and ρ . Very often the function F_1 combining precision and recall using equal weights can be seen.

In the case of 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, \dots, T_k . For each subset a classifier $\hat{\Phi}_i$ is learned using $\Omega \setminus T_k$ 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.

4. 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 *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 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 <http://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 one by 8995 documents. Documents from this collection are in the Slovak language unlike the first collection, which is in English.

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

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 algorithm constructs the pessimistic estimation by calculating standard deviation of estimated accuracy given binomial distribution.

4.1. 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 x -axis label represents classes, which are ordered decreasingly according their frequency).

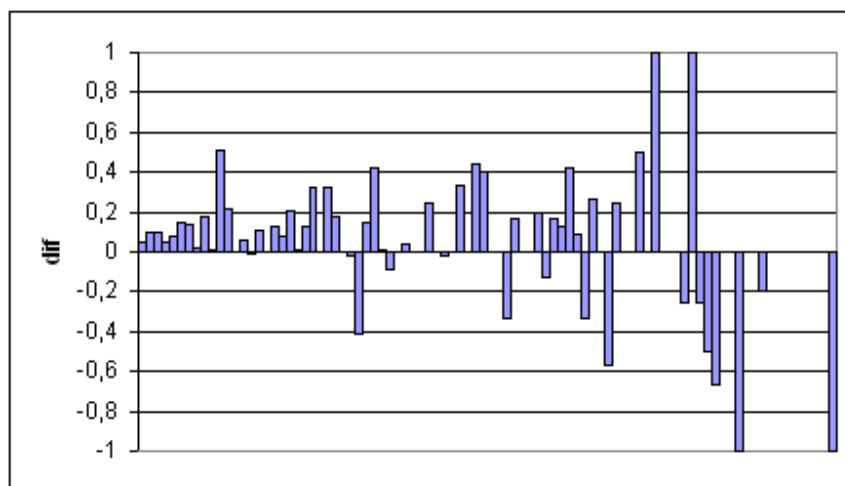


Figure 1. The precision differences between boosting-based classifier and a perfect decision tree for data from the Reuters collection.

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.

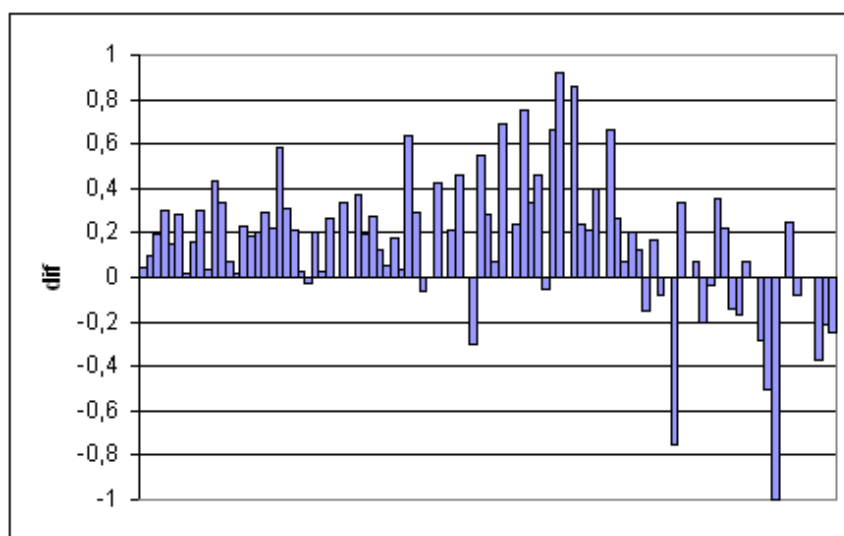


Figure 2. Precision differences between boosting-based classifier and a perfect decision tree for data from the Markiza collection (the x -axis label represents classes, which are ordered decreasingly according their frequency).

There are great differences in some cases in Figure 1 and Figure 2. Extreme differences between the classifiers for rare categories are caused by the instable estimation of recall/precision on the very small number of the testing examples (less than 10). The results can be interpreted in such a way, that the boosting method provides better results while for frequent classes the difference is minimal.

4.2. 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 line.

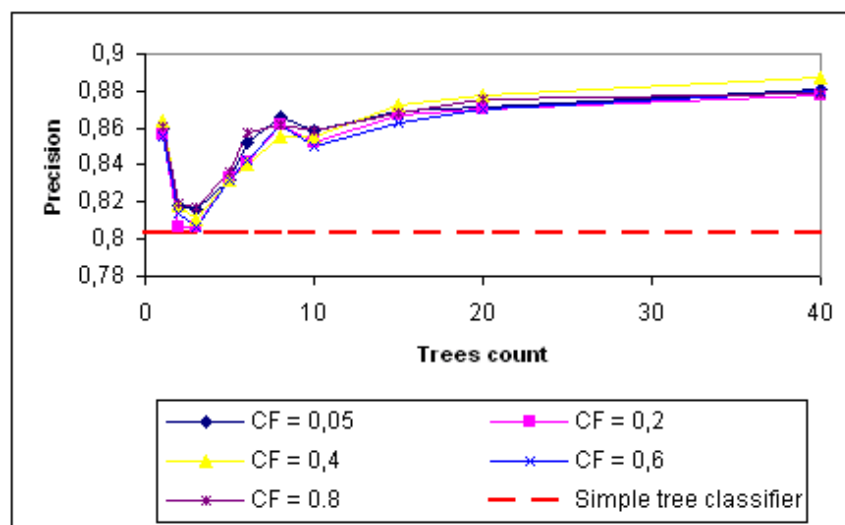


Figure 3. Relationship between precision and the number of trees in the boosting classifier.

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, Figure 5 presents that boosting is superior for the number of classifiers greater than 5. Using 20 or more classifiers, F_1 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

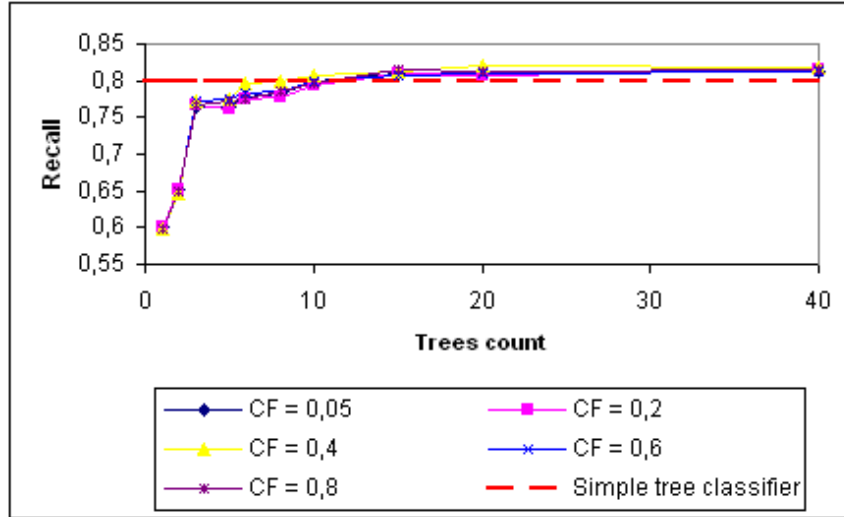


Figure 4. Relationship between recall and the number of trees in the boosting classifier.

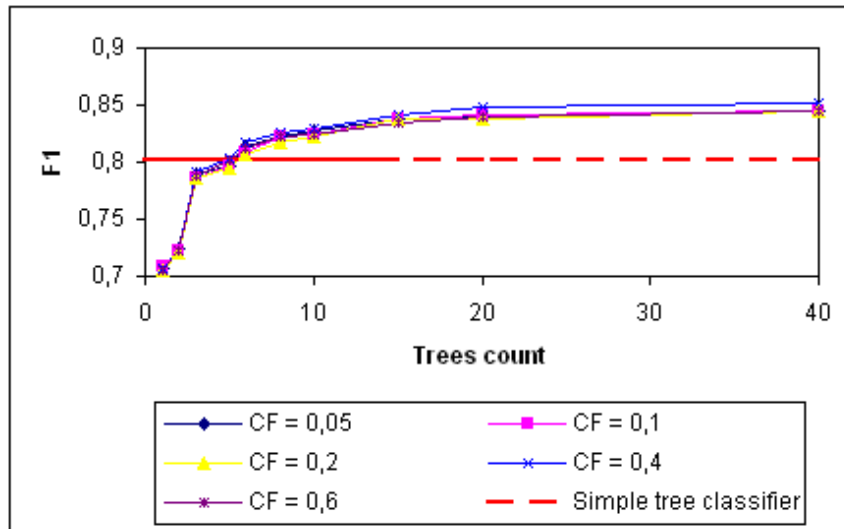


Figure 5. Relationship between F_1 parameter and the number of trees in the boosting classifier.

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.

In fact we could achieve statistically better estimation of efficiency by cross validation. However, *Reuters-21578*² document collection is used usually in text categorization as a standard database with predefined division of training set on training and testing part. It enables direct comparison with other methods.

5. Conclusion

The boosting algorithm is a suitable mean for increasing efficiency of these algorithms, which have low values of *precision* and *recall*³. Both mentioned 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. 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.

In the future, we would like to make some experiments with boosting in combination with some algorithm generating threshold units, for example Heuristic Criteria Table [2], or Inductive Inference Algorithm [5].

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] L. Breiman, Bagging predictors, *Machine Learning* **26** (1996), 123–140.
- [2] P. Langley, *Elements of Machine Learning*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996, 419 pp..

²Most experiments were carried out using this document collection, if not given otherwise.

³Mainly recall for binary trees.

142 K. Machová, M. Puzta and P. Bednár : An improvement of the class. algorithm results

- [3] K. Machová, J. Paralič, Basic Principles of Cognitive Algorithms Design, *Proc. of the IEEE International Conference Computational Cybernetics*, Siófok, Hungary (2003), 245–247, ISBN 963 7154 175.
- [4] J. R. Quinlan, Bagging, boosting and C4.5., *Proc. of the Fourteenth National Conference on Artificial Intelligence* (1996).
- [5] L. Samuelis, L. Fasianok, The role of Inductive Inference in the Design of Intelligent Tutoring Systems, *Acta Elektrotechnica et Informatica* **4**, no. 4, Košice (2004), 54–58, ISSN 1335-8243.
- [6] R. E. Schapire, Y. Singer, Improved Boosting Algorithms Using Confidence-rated Predictions, *Machine Learning* **37**, no. 3 (1999), 297–336.
- [7] R. E. Schapire, Y. Singer, BoostTexter: A Boosting-based System for Text Categorization, *Machine Learning* **39**, no. 2/3 (2000), 135–168.

KRISTÍNA MACHOVÁ
DEPARTMENT OF CYBERNETICS AND ARTIFICIAL INTELLIGENCE
TECHNICAL UNIVERSITY
LETNÁ 9
04200 KOŠICE
SLOVAKIA

E-mail: Kristina.Machova@tuke.sk

(Received October, 2005)