Heterogeneous Forests of Decision Trees.

Krzysztof Grąbczewski and Włodzisław Duch

Department of Informatics, Nicholas Copernicus University, Grudziądzka 5, 87-100 Toruń, Poland. http://www.phys.uni.torun.pl/kmk

Abstract. In many cases it is better to extract a set of decision trees and a set of possible logical data descriptions instead of a single model. The trees that include premises with constraints on the distances from some reference points are more flexible because they provide nonlinear decision borders. Methods for creating heterogeneous forests of decision trees based on Separability of Split Value (SSV) criterion are presented. The results confirm their usefulness in understanding data structures.

1 Introduction

Recent trends in computational intelligence (CI) and data mining prove that understanding of data becomes increasingly more important. Especially in medicine CI expert systems always provide some explanations of diagnoses inferred by decision support tools. Human experts may not be satisfied with a single description of the problem they examine, even if it is quite accurate. Different experts may stress different factors in their evaluations of the problem, corresponding to different models of the data they are used to.

One of the most attractive ways to describe the classification process is to use logical rules, which can be easily extracted from decision trees. Usually data mining systems provide their users with just a single set of rules (the one that seems to be the most precise). At best several sets of rules with increasing accuracy are provided [1]. However, many sets of rules with similar complexity and accuracy may exist, using for example different feature subsets, bringing more information of interest to the domain expert. Therefore methods that are aimed at finding many different descriptions of the same data are worth investigation.

Multiple solutions to the problem may be even more informative when they use different types of rule premises. The tests usually concern the features describing data, but to make a decision tree heterogeneous they may include conditions that test the distances from the data vectors to a selected point in the feature space.

We have augmented our decision tree based on the *Separability of Split Value* (SSV) criterion [3, 4] with the capability to generate heterogeneous forests of trees instead of single trees. The algorithms have been tested on several datasets coming from the UCI repository of machine learning problems [5].

2 SSV criterion

The SSV criterion is one of the most efficient among criteria used for decision tree construction [3, 4]. It's basic advantage is that it can be applied to both continuous and discrete features, which means that methods based on it can operate on raw data without the need for any data preprocessing. The *split* value (or *cut-off point*) is defined differently for continuous and discrete features. In the case of continuous features the split value is a real number, in other cases it is a subset of the set of alternative values of the feature. The best split value is the one that separates the largest number of pairs of objects from different classes, so for both types of features *left side* (*LS*) and *right side* (*RS*) of a split value s of feature f for a given dataset D can be defined:

$$\begin{aligned} \mathrm{LS}(s,f,D) &= \begin{cases} \{x \in D : f(x) < s\} & \text{if } f \text{ is continuous} \\ \{x \in D : f(x) \notin s\} & \text{otherwise} \end{cases} \\ \mathrm{RS}(s,f,D) &= D - \mathrm{LS}(s,f,D) \end{aligned} \tag{1}$$

where $f(x)$ is the f's feature value for the data vector x. The *separability of a split value* s is defined as:

$$\mathrm{SSV}(s) = 2 * \sum_{c \in C} |LS(s,f,D) \cap D_c| * |RS(s,f,D) \cap (D - D_c)| \tag{2}$$

$$- \sum_{c \in C} \min(|LS(s,f,D) \cap D_c|, |RS(s,f,D) \cap D_c|) \tag{3}$$

where C is the set of classes and D_c is the set of data vectors from D which belong to class $c \in C$. Similar criterion has been used for design of neural networks by Bobrowski *et al.* [6].

Among all the split values which separate the maximal number of pairs of vectors from different classes the best is the one that separates the smallest number of pairs of vectors belonging to the same class. For every dataset containing vectors which belong to at least two different classes, for each feature which has at least two different values, there exists a split value with largest separability. When the feature being examined is continuous and there are several different split values with maximal separability close to each other, the split value closest to the average of all of them is selected. To avoid such situations it is good to examine split values that are natural for a given dataset (i.e. centered between adjacent feature values that occur in the data vectors). If there are non-maximal (regarding separability) split values between two maximal points or if the feature is discrete, then the best split value is selected randomly.

Decision tree is constructed recursively by searching for best splits. At each stage when the best split is found and the subsets of data resulting from the split are not completely pure (i.e. contain data belonging to more than one class) each of the subsets is being analyzed the same way as the whole data. The decision tree built this way gives maximal possible accuracy (100% if there are no contradictory examples in the data) which usually means that the created model overfits the data. To remedy this a cross validation training is performed to find the optimal pruning parameters for the tree. Optimal pruning produces a tree capable of good generalization of the patterns used in the tree construction process.

3 Heterogeneous SSV Trees

The separability measure can be applied not only to the features supplied in the dataset. It can be calculated for other values like linear or nonlinear combinations of features or the distances from the data vectors to given points in the feature space.

Although the computational complexity of the analysis of such new features is not very high [2], the problem becomes very complex because the number of possible features to be analyzed is infinite. In the case of distance features there are some "natural" restrictions on reference points selection (for example they may be searched among the training data vectors).

Because different distance measures lead to very different decision borders, such enhancement may significantly simplify the structure of the resulting tree (i.e. discover simple class structures).

4 Construction of Forests

There are two ways to generate forests of trees, or more generally different models of the data, related to the variance due to the data sample and the variance due to the flexibility of the data model itself. The simplest way of finding multiple solutions to a problem is to apply a data mining technique to different samples of data depicting the same problem. Most computational intelligence methods, including decision trees, are not quite stable, for some data sets small perturbations of the training data may lead to remarkably different results [7]. It can be easily seen in cross validation tests, where each training pass is performed for different data selection. In section 5 we present some examples of application of this method.

Another possibility of forest generation is to explore beam search strategy which looks for the best tree (it is more efficient than multiple data resampling and adaptive model training). In each of beam search stages there is a beam of *n* decision trees (search states). Each stage adds a single split to the trees in current beam, so very simple trees in first beams and more complex structures in next stages are created. The usual technique to search for the best tree is to build the shortest tree with maximal accuracy possible (the first final state found), and then prune it to obtain good generalization (pruning is determined in SSV empirically by means of cross validation). This method has proven to be quite accurate, but it is justified strictly in the case of the shortest tree, other trees may require different pruning parameters. To find a forest all trees that appear in a beam at any stage of the search process are ordered by their estimated accuracy. To estimate the accuracy a cross validation on the training data is performed and a validation error is assigned to each beam position which is a pair (s, p) where s is search stage (beam index) and p is the position in that beam (the states in a beam can be sorted by the errors for the training set). The precise algorithm goes as follows:

– Perform n-fold cross validation for the training set. For each part:
    - Run beam search process (do not stop when first final state is found - stop when all the states in current beam are final);

    - for each pair (s,p) where $s \in \{1,\dots,m\}$, m is the number of search stages (the number of beams generated), $p \in \{1,\dots,k_s\}$, k_s is the number of states in sth beam, calculate the validation error $E_V = E_{TRN} + nE_{TST}$, where E_{TRN} and E_{TST} are the numbers of misclassified samples in the training and test (validation) parts, respectively.
- Run beam search for the whole training data.
- Assign an average validation error to all pairs (s,p) that correspond to some states in the generated beams.
    - Discard the pairs that do not correspond to a state in any of cross validation parts (the idea is that such search states are not common and would not generalize well);
    - When calculating the average values remove the maximal and the minimal ones.
- Sort all the beam states positions with increasing average validation error.
- Present a required number of leading states as the forest.

5 Results

We have applied the above algorithm to several real life datasets from the UCI repository [5]. In all the cases we have found some sets of logical rules describing the data. The rule sets (in fact decision trees) which after sorting got to the beginning of the state list were very similar to the best known logical description of that data. Several alternative sets have been found for each analyzed dataset. The notation d(_,vnnn) used in the rules means the square of euclidean distance to the vector nnn).

The hypothyroid dataset was created from medical tests screening for hypothyroid problems. Since most people were healthy 92.5% of cases belong to the normal group, and 8% of cases belonging to the primary hypothyroid or compensated hypothyroid group. 21 medical facts were collected in most cases, with 6 continuous test values and 15 binary values. A total of 3772 cases are given for training (results from one year) and 3428 cases for testing (results from the next year). Thus from the classification point of view this is a 3 class problem with 22 attributes.

The forest search algorithm has found many rule sets of very high accuracy and confirmed that the highest results for this dataset may be obtained with methods that cut the space perpendicularly to the axes. Because of the rectangular decision borders needed in this case, the premises with distance tests occur at quite deep levels of the trees.

One of the most interesting rule sets is 99.84% accurate (6 errors) on the training set and 99.39% (21 errors) on the test set (all the values of continuous features are multiplied here by 1000):

1. if TSH $> 6.05 \wedge$ FTI $< 64.72 \wedge$ (T3 $< 11.5 \vee$ d(_,v2917) < 1.10531) then primary hypothyroid
2. if TSH $> 6.05 \wedge$ FTI $> 64.72 \wedge$ on_thyroxine $= 0 \wedge$ thyroid_surgery $= 0 \wedge$ TT4 < 150.5 then compensated hypothyroid
3. else healthy.

The Wisconsin breast cancer dataset contains 699 instances, with 458 benign (65.5%) and 241 (34.5%) malignant cases. Each instance is described by 9 attributes with integer value in the range 1-10 and a binary class label. For 16 instances one attribute is missing. The data had been standardized before the application of the algorithm.

The results for this dataset are surprisingly good - single premise rules are more accurate, than much more complicated homogeneous rule sets known so far. Some of the rules chosen for the forest are presented in Table 1.

Rules	Accuracy / Sensitivity / Specificity
1. if d(_,v303) < 62.7239 then malignant 2. else benign	97.3% / 97.9% / 96.9%
1. if d(_,v681) < 51.9701 then malignant 2. else benign	97.4% / 98.8% / 96.8%
1. if d(_,v613) < 65.3062 then malignant 2. else benign	97.1% / 97.9% / 96.7%
1. if d(_,v160) < 36.4661 then malignant 2. else benign	97.1% / 98.8% / 96.3%
1. if d(_,v150) < 39.5778 then malignant 2. else benign	97.1% / 98.8% / 96.3%

Table 1. Classification rules for the Wisconsin breast cancer data.

The appendicitis data contains only 106 cases, with 8 attributes (results of medical tests), and 2 classes: 88 cases with acute appendicitis (class 1) and 18 cases with other problems (class 2). For this small dataset very simple classification rules have been found by Weiss and Kapouleas [9] using their PVM (Predictive Value Maximization) approach and by us in some of our earlier approaches [1, 3].

The heterogeneous forest algorithm discovered several new descriptions of this data which are simple, accurate and very sensitive. Some of them are presented in Table 2 (the distances were calculated for standardized data).

Rules	Accuracy / Sensitivity / Specificity
1. if d(_,v73) < 12.0798 $\vee$ MBAA > 1174.5 then class 1 2. else class 2	92.5% / 98.8% / 66.7%
1. if d(_,v22) < 16.4115 $\vee$ MBAP > 12 then class 1 2. else class 2	92.5% / 98.8% / 66.7%
1. if d(_,v22) < 16.4115 $\vee$ MBAA > 1174.5 then class 1 2. else class 2	92.5% / 98.8% / 66.7%
1. if d(_,v8) < 13.385 $\wedge$ HNEA $\notin (9543, 9997)$ then class 1 2. if d(_,v8) > 13.385 $\wedge$ MBAA > 1174.5 $\wedge$ MNEP > 51 then class 1 3. else class 2	96.2% / 98.8% / 85.7%

Table 2. Classification rules for the appendicitis data.

6 Discussion

Medical datasets often contain small number of examples (sometimes 100 or less) with relatively large number of features. In such cases of large spaces with sparse samples it is quite likely that different logical expressions may accidently classify some data well, thus many data mining systems may find solutions which are precise but not meaningful according to the experts' knowledge. Providing experts with several alternative descriptions gives more chance that they find interesting explanations compatible with their experience and notice new regularities in the data, leading to a better understanding of the problem. Heterogeneous forests are capable of finding simple classification descriptions when nonlinear decision borders are necessary and can find reference vectors which sometimes may be used as prototypes for similarity based methods.

A forest consists of trees which very often differ in their decisions (different samples are misclassified by different trees). Such trees may successfully act as a committee giving answers interpretable as probabilities. It can also yield better classification accuracy, but even if it does not, it will provide an interesting solution to the problem of combining the comprehensibility of decisions with high accuracy and confidence.

Another possible extension of this work is to automatically detect how many trees are interesting enough to compose the forest. This can be done by means of forest committees or the evaluation of statistical significance [8] of the classification accuracies differences.

Acknowledgments: Support by the Polish Committee for Scientific Research, grant 8 T11C 006 19, is gratefully acknowledged.

References

1. Duch W, Adamczak R. and Grąbczewski K. (2001) Methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks* **12**: 277-306
2. Duch W, Grąbczewski K. (2002) Heterogeneous adaptive systems. *World Congress of Computational Intelligence*, Honolulu, May 2002
3. Grąbczewski K, Duch W. (1999) A general purpose separability criterion for classification systems, *4th Conference on Neural Networks and Their Applications*, Zakopane, Poland, pp. 203-208
4. Grąbczewski K. Duch W. (2000) The Separability of Split Value Criterion, *5th Conference on Neural Networks and Soft Computing*, Zakopane, Poland, pp. 201-208
5. Blake, C.L, Merz, C.J. (1998) UCI Repository of machine learning databases http://www.ics.uci.edu/ mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.
6. Bobrowski L, Krętowska M, Krętowski M. (1997) Design of neural classifying networks by using dipolar criterions. *3rd Conf. on Neural Networks and Their Applications*, Kule, Poland
7. Breiman L. (1998) Bias-Variance, regularization, instability and stabilization. In: Bishop, C. (Ed.) Neural Networks and Machine Learning. Springer, Berlin, Heidelberg, New York
8. Dietterich T. (1998) Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, Neural Computation **10**, 1895-1923
9. S.M. Weiss, I. Kapouleas. "An empirical comparison of pattern recognition, neural nets and machine learning classification methods", in: *Readings in Machine Learning*, eds. J.W. Shavlik, T.G. Dietterich, Morgan Kauffman Publ, CA 1990