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. Methods for creating forests of decision trees based on Separability of Split Value (SSV) criterion are presented. Preliminary 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.

We have augmented our decision tree based on the *Separability of Split Value* (SSV) criterion [2,3] with the capability to generate forests of trees, instead of single trees. The algorithms have been tested on several datasets coming from the UCI repository of machine learning problems [4].

2 SSV criterion

The SSV criterion is one of the most efficient among criteria used for decision tree construction [2,3]. 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.* [5].

Among all the split values that separate the maximal number of pairs of vectors from different classes the one which 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 that generalizes well the patterns used for tree construction.

3 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 [6]. It can be easily seen in cross validation tests, where each training pass is performed for different data selection. In section 4 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. 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 $s \in \{1, \ldots, m\}$, m is the number of search stages (the number of beams generated), $p \in \{1, \ldots, k_s\}$, 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.

4 Results

We have applied the above algorithm to three real life datasets from the UCI repository [4]. In all the three 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. In addition some alternative sets have been found for each analyzed dataset.

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 pointed to the following rules as the first (the best) one (all values of continuous features are multiplied here by 1000):

1. if TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\wedge$ thyroid surgery = no then primary hypothyroid
2. if TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no $\wedge$ TT4 $<$ 150.5 then compensated hypothyroid
3. else healthy

These rules are 99.79% accurate (8 errors) on the training data and 99.33% accurate (23 errors) on the test data.
The second set of rules in the forest gives accuracy of 99.76% (9 errors) and 99.36% (22 errors) on training and test sets respectively:

1. if TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 then primary hypothyroid
2. if TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no $\wedge$ TT4 $<$ 150.5 then compensated hypothyroid
3. else healthy

These sets of rules occur several times in the forest, since they can be obtain in several different ways (equivalent forms). At the fifth position in the forest there is another (slightly simpler) set of rules (99.52% on training, 98.89% on test)

1. if TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\wedge$ thyroid surgery = no then primary hypothyroid
2. if TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no then compensated hypothyroid
3. else healthy

Further positions bring even more simple rules. For instance the 13th set, with training accuracy 99.50% and test accuracy 98.92% is:

1. if TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 then primary hypothyroid
2. if TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid surgery = no $\wedge$ on thyroxine = no 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 meaning of features is: $F_2 =$ Uniformity of Cell Size, $F_3 =$ Uniformity of Cell Shape, $F_5 =$ Single Epithelial Cell Size, $F_6 =$ Bare Nuclei, $F_8 =$ Normal Nucleoli. Some of the rule sets chosen for the forest are presented below.

1. if $F_3 < 2.5$ then benign	95.6% accuracy (25 err. + 6 unclass.)
2. if $F_6 < 2.5 \wedge F_5 < 3.5$ then benign	95.0% sensitivity
3. else malignant	95.9% specificity
1. if $F_3 < 2.5$ then benign	95.0% accuracy (35 err.)
2. if $F_5 < 2.5 \wedge F_8 < 2.5$ then benign	93.8% sensitivity
3. else malignant	95.6% specificity
1. if $F_3 < 2.5$ then benign	95.1% accuracy (34 err.)
2. if $F_5 < 2.5 \wedge F_2 < 2.5$ then benign	95.9% sensitivity
3. else malignant	94.8% specificity
1. if $F_3 < 2.5$ then benign	95.1% accuracy
2. if $F_5 < 2.5 \wedge F_6 < 2.5$ then benign	95.0% sensitivity
3. else malignant	95.2% specificity
1. if $F_3 < 2.5$ then benign	95.1% accuracy
2. if $F_6 < 2.5 \wedge F_3 < 4.5$ then benign	94.2% sensitivity
3. else malignant	95.6% specificity

The Ljubljana breast cancer data contains 286 cases, of which 201 are no-recurrence-events (70.3%) and 85 are recurrence-events (29.7%). There are 9 attributes with 2-13 different values each. This is a difficult and noisy data. In our forest two rule sets occur at the beginning:
75.5% accuracy (70 errors), probably due to small sample and without real meaning, 30.5% sensitivity, 94.5% specificity:

1. if breast = left $\wedge$ inv-nodes > 2.5 then recurrence-events
2. else no-recurrence-events

Quite understandable rule making 68 errors, 31.8% sensitivity, 95.0% specificity:

1. if deg-malig $> 2.5 \wedge$ inv-nodes > 2.5 then class recurrence-events
2. else no-recurrence-events

Further some more detailed rules can be found. First of them stands at position 22 and gives 76.92% accuracy (66 errors), 47.1% sensitivity, 89.6% specificity:

1. if deg-malig $> 2.5 \wedge$ inv-nodes > 2.5 then class recurrence-events if deg-malig $> 2.5 \wedge$ inv-nodes $< 2.5 \wedge$ (tumor-size = 30-34 $\vee$ tumor-size = 25-29 $\vee$ tumor-size = 50-54) then class recurrence-events
2. else no-recurrence-events

Although the overall accuracy may be similar different sets of rules may have significantly different values of sensitivity and specificity.

5 Discussion

Medical datasets often contain small number of examples (sometime 100 and less) with relatively large number of features. In such cases many data mining systems may find solutions which are precise but not meaningful according to the experts' knowledge. In large spaces with sparse samples it is quite likely that several different logical expressions may accidently classify some data well. Such a case is evident in the Ljubliana cancer data. Providing experts with several alternative descriptions gives more chance that they will find interesting explanations in common with their experience and will notice new regularities in the data that may give them a better understanding of the problem.

Forests consist of trees which very often differ in their decisions (different samples are misclassified by different trees). It will be very interesting to see the results obtained using trees from the forest as committees. In such a case the committee of decision trees can give answers which may be interpreted as probabilities of object's belonging to a given class. Of course it can also yield better classification accuracy, but even if it does not, it provides 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 add a method to automatically detect how many trees give interesting results and should be kept in the forest. This can be done by means of forest committees. One also can use methods to evaluate statistical significance [7] of the differences among classification accuracies of trees to select the trees that give insignificantly different results.

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. 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
3. Grąbczewski K. Duch W. (2000) The Separability of Split Value Criterion, *5th Conference on Neural Networks and Soft Computing*, Zakopane, Poland.
4. 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.
5. 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
6. Breiman, L. (1998): Bias-Variance, regularization, instability and stabilization. In: Bishop, C. (Ed.) Neural Networks and Machine Learning. Springer, Berlin, Heidelberg, New York
7. Dietterich T. (1998) Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, *Neural Computation* **10**, 1895-1923