Prototype-Based Threshold Rules

Marcin Blachnik[1] and Włodzisław Duch[2,3]

[1] Division of Computer Methods, Department of Electrotechnology and Metallurgy, The Silesian University of Technology, Krasińskiego 8, 40-019 Katowice, Poland
marcin.blachnik@polsl.pl
[2] Department of Informatics, Nicolaus Copernicus University, Grudziądzka 5, Toruń, Poland
[3] School of Computer Engineering, Nanyang Technological University, Singapore
Google: Duch

Abstract. Understanding data is usually done extracting fuzzy or crisp logical rules using neurofuzzy systems, decision trees and other approaches. Prototype-based rules are an interesting alternative providing in many cases simpler, more accurate and more comprehensible description of the data. Algorithm for generation of threshold prototype-based rules are described and a comparison with neurofuzzy systems on a number of datasets provided. Results show that systems for data understanding generating prototypes deserve at least the same attention as that enjoyed by the neurofuzzy systems.

1 Introduction

Data mining and knowledge discovery algorithms are focused on understanding of data structures, still one of most important challenges facing computational intelligence. Data understanding requires extraction of crisp or fuzzy logical rules. For some datasets surprisingly accurate and simple logical rules may be generated [1], but in some cases sets of logical rules may be too large or too complicated to be useful. Crisp rules partition the input space into hyperboxes and thus even relatively simple tasks that require oblique decision borders may lead to complicated sets of rules. All major data mining software suits use as their important component decision trees for crisp logical rule extraction. C4.5 [3] and CART [4] are the most popular algorithms used in many packages, but there are many others, for example SSV trees [5] used in the Ghostminer package [6]. Inductive machine learning algorithms are rarely used in data mining systems. Fuzzy rules (F-rules) are more flexible and have been used in many successful real word applications reported in literature [7]. Unfortunately this group of methods is also limited; usually they are less transparent then crisp logical rules (C-rules) and may be difficult to understand, therefore they are rarely found in data mining software suits. F-rules work well with continuous numerical attributes but the real word applications often require analysis of mixed data, including symbolic or nominal attributes, which are not supported directly by fuzzy rules.

An alternative approach to data understanding, based on similarity [8] rather than logic, extracts rules based on prototypes (P-rules). People making decisions rarely use logic, but most often use their memory to recall similar cases and outcomes of previous decisions. In similarity-based learning framework (SBL) two major types of P-rules have been defined [9]: nearest neighbor rules, where classification decisions

I. King et al. (Eds.): ICONIP 2006, Part III, LNCS 4234, pp. 1028 – 1037, 2006.

are based on rules assigning query vectors to the same class that majority of the closest prototypes belong to, and the threshold rules, where each prototype has an associated distance threshold which defines subspace around the prototype with associated class label.

One way to define prototypes threshold rules is by using heterogeneous decision trees (HDT) [10], a classical decision tree algorithm that is extended to use new attributes based on distances between training vectors. This algorithm has found some of the most accurate and simplest descriptions of several datasets. Another approach to threshold rules is based on a Prototype Threshold Decision List (PTDL), where linear list of ordered rules is created. In this paper the PTDL algorithm is introduced and compared with HDT. The next section describes how P-rules support different types of attributes, the third section presents threshold rules decision list algorithm, in the fourth section results of numerical experiments are presented, and the last section contains conclusions and discussion.

2 Heterogeneous Distance Functions

Real word datasets often contain different types of features, creating serious difficulties for large group of computational intelligence algorithms, including most methods based on fuzzy rules [7][11][13]. P-rules solve the problem of combination of continuous, discrete, symbolic and nominal attributes using heterogeneous distance function (HDF).

HDFs introduced by Wilson and Martinez [14] allow for calculation of distance for all types of attributes. Several types of HDF have been introduced, based on an assumption that distances are additive:

$$D(\mathbf{x},\mathbf{r})^{\alpha} = \sum_{i=1}^{n} d(x_i, r_i)^{\alpha} \tag{1}$$

where $D(\mathbf{x},\mathbf{r})$ is the distance between two vectors and $d(x_i, r_i)$ is the distance calculated for a single dimension. In the SBL framework [8] HDF allow for treating different types of features in a natural way. For real-valued or ordered discrete features Minkovski's distances (2) are used and for symbolic features probabilistic distance functions (3) are used, for example:

$$D_{Mink}(\mathbf{x},\mathbf{r})^{\alpha} = \sum_{i=1}^{n} |x_i - r_i|^{\alpha} \tag{2}$$

$$D_{VMD}(\mathbf{x},\mathbf{r})^{\alpha} = \sum_{i=1}^{n}\sum_{j=1}^{C} \left| p(c_j \mid x_i) - p(c_j \mid r_i) \right|^{\alpha} \tag{3}$$

where **x** and **r** are respectively data and reference vectors, n is the number of features, C is the number of classes, and α is the value of exponent (α=2 for Euclidean functions). $p(c_j \mid x_i)$ and $p(c_j \mid r_i)$ are calculated as

$$p\left(c_j|x_i\right) = \frac{Nx_{ij}}{Nx_i} \tag{4}$$

where Nx_i is the number of instances in the training set that have value x for attribute i, and Nx_{ij} is the some as Nx_i but for class j.

This types of distance functions are additive, so the overall distance function can be calculated as a sum of contributions from both types of distance measures, depending on the attribute types (4):

$$D(\mathbf{x},\mathbf{r})^{\alpha} = D_{Mink}\left(\mathbf{x_a},\mathbf{r_a}\right)^{\alpha} + D_{VDM}\left(\mathbf{x_b},\mathbf{r_b}\right)^{\alpha} \tag{5}$$

where $\mathbf{x_a}$ and $\mathbf{r_a}$ are subsets of continuous attributes of vectors $\mathbf{x}$ and $\mathbf{r}$, and $\mathbf{x_b}$ and $\mathbf{r_b}$ are subsets of their symbolic features. Features should be normalized to assure that each distance component has the some or comparable influence.

In P-rules α parameter have significant influence on the shape of decision borders. Changing α value from 1 to ∞ different shapes of hypersurfaces of constant value are obtained. For α equal 1 rhomboidal shape is obtained, for $\alpha=2$ spherical, higher α values lead to rectangular shapes, and for $\alpha=\infty$ lines of constant distance reach a square shape. This aspect of P-rules can allow for smooth transition to crisp logical rules if it is necessary. Also fuzzy rules can be extracted from datasets in this way, as discussed in [15].

3 Threshold Rules

P-rules based on distances from prototypes create tessellation of the input space, with most distance functions leading to convex polytope cells with hyperplane faces. Some cells are infinite, and the use of only two prototypes $\mathbf{r}_1$, $\mathbf{r}_2$ is equivalent to the linear discrimination, defining single hyperplane perpendicular to the line that joins them. Threshold based rules are not based on competition for the closest prototype, but simply assign all vectors $\mathbf{x}$ with $D(\mathbf{x},\mathbf{r})<\theta$ to the same class as the prototype $\mathbf{r}$. The effect is somehow similar to the use of basis expansion networks with localized functions, such as RBFs with Gaussian functions. However, the emphasis here is not on approximation but on data understanding, generation of a small number of simple rules with distance functions based only on relevant features.

A constructive algorithm is recommended, creating first quite general P-rules, and then more detailed rules and possible exceptions until the whole input space is covered. Two strategies to solve the problem of adding new rules developed so far are based on:

- Heterogeneous Decision Trees;
- ordered prototype threshold decision list.

3.1 Heterogeneous Decision Trees

Standard decision trees, such as the C4.5 [3], CART [4] or SSV trees [5], use only one type of test to split the data, $T(x_i < \theta)$, dividing the range of feature values x_i into

two or more branches. Heterogeneous decision trees (HDT), introduced in [10] use at least two qualitatively different types of tests. Adding the second test $T(D(\mathbf{x},\mathbf{r}_k) < \theta_k)$ based on similarity to prototypes provides localized decision borders to the hyperplanes contributed by the standard tests. In the simplest case all training vectors are initially taken as prototypes, using the square [N x N] distance matrix $D(\mathbf{r}_i,\mathbf{r}_j)$, where N is the number of input vectors. Then prototype vectors are consecutively removed and accuracy checked, until a small number of prototypes is left and accuracy starts to degrade. Similarities may be calculated either using Euclidean distances or Gaussian kernel functions.

Combination of hyperplanes obtained from binary splits of features with spherical decision borders from distance based threshold tests is quite powerful and may lead to interesting rules, although the search for the prototype by elimination of the training vectors is a rather costly procedure, with complexity of $O(N^2)$. This approach applied to the Wisconsin Breast Cancer data generated a single distance based P-rule with 97.3% accuracy, providing the simplest and most accurate description of this data found so far [10].

3.2 Prototype Threshold Decision List Algorithm

Heterogeneous classification trees used for extraction of prototype threshold rules create hierarchical sets of rules. An alternative is given by a covering algorithm that creates ordered list of rules that may overlap, called here Prototype Threshold Decision List (PTDL). This algorithm is based on similar criteria like HDT algorithms, however individual rules are stored in an ordered list, starting from the most general rule to the most detailed. Because they are overlapping this list of rules should be applied in an order, beginning from the most specific (and least reliable), and if its conditions are not fulfilled the next more general rule should be checked. In the end if none of the rules may be applied, the output label is assigned to the *else* condition, covering all the remaining vectors (Fig. 1).

PTDL searches among all training vectors for a prototype that maximizes appropriate decision tree criterion, like separability (SSV), the Gini index (CART) or Information Gain (C4.5). Each prototype and threshold define particular rule, splitting the data into vectors in the subspace covered by this prototype with selected threshold (vectors that fall inside of the rule borders), and the remaining vectors that fall outside of this subspace (outside of the rule). For multiclass problems these two types of prototype threshold rules should be explicitly distinguished: inside rules with $D(\mathbf{x},\mathbf{r}_k) < \theta_k$, and outside rules with $D(\mathbf{x},\mathbf{r}_k) \geq \theta_k$, where each rule is defined for one particular class. In the first case prototype $\mathbf{r_k}$ belong to the subspace, and in the second case it does not belong to the subspace defined by the prototype and threshold. For two class problems distinguishing between these two types of rules is not important, however for the multiclass problems it has significant meaning, increasing generalization and model simplicity.

The sketch of the PTDL code is presented in Fig. 2, where for simplicity two class problem is described. In the first step `CreateList` function calculates distances or similarities between all training vectors, storing them in the square matrix `D` of size

`NxN` where `N` is the number of training vectors. Then search for all possible splits of each training vector that may increase criterion value is performed (`for` loop). Only splits between pairs of neighboring vectors in each column of matrix `D` belonging to different classes are considered, because only such situation guarantees maximization of the criterion function. The middle points between these pairs of vectors are taken as thresholds. All parameters: the criterion value (`C_Crit`), threshold (`C_Threshol`) , and rule consequence – class labels are calculated by the function `CalcCriterion`, which returns column vectors with appropriate parameter values for currently analyzed `i`-th training vector .

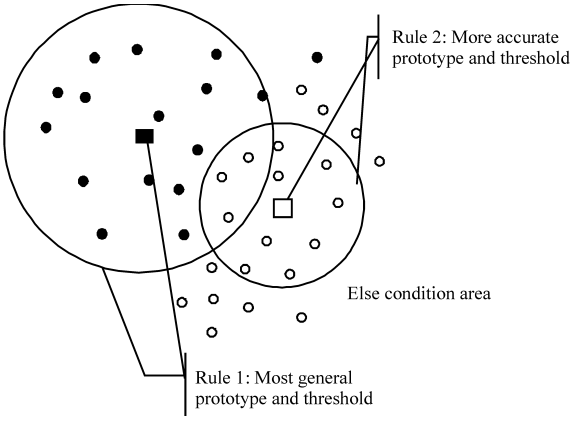

Fig. 1. Example of threshold rules: Rule 1 – most general; Rule 2 – more accurate; and Else area in the remaining subspace

The best among N training vectors with appropriate threshold maximizing particular criterion function is stored in the rule set list. When a new rule is accepted all training vectors are classified with the current set of rules to mark all vectors that are incorrectly classified and should be used to search for further rules. The PTDL algorithm stops if the maximum number of rules is reached, or when all vectors are correctly classified.

This straightforward covering algorithm does not assure good generalization. To remedy its weakness optimal number of rules is found using internal crossvalidation on the training data (as it is done in the SSV trees [5][10]). Using k-fold crossvalidation test for each fold a new decision list is created. In the end at each level of the list appropriate criterion is checked (Gini has been used here, but information gain, balanced accuracy, separability or other criteria may be optimized), and the optimal number of rules that maximize the desired criterion is selected. Rule extraction algorithms frequently generate quite different sets of rules, suffering from high variance of solutions. To avoid such situation the difference between accuracy and standard deviation in crossvalidation calculations is optimized, selecting highly accurate low variance solutions.

```
function [P,PLab,TH] = CreateList(T,TLab,MaxRules)
1.  input:
       T - training set
       TLab - labels of training vectors
       MaxRules - maximum number of rules

2.  output:
       P - set of prototypes
       PLab-set of labels associated with each prototype P
       TH - set of thresholds for appropriate prototypes

3.  var
       N - Number of training vectors
       D - distance matrix NxN
       RulN - number of created rules
       splits - list of possible splits where evaluation
       of the criterion is calculated
       C_Crit - vector of criterion values calculated for
                each split
       C_Threshold - appropriate threshold for each split
       C_PLab - class label for current split
       CurLab - Class labels predicted by set of rules,
                initially all vectors are wrong classified
       MXcrit - maximum criterion value for i-th prototype
       idx - index of best split

begin
4.  D = dist(T,T);// distances between training vectors;
5.  RulN=1;
6.  while (RulN < MaxRuls) or ErrorsN == 0
7.      for I = 1:N  // considered each training vector
                     as prototype
8.      splits = FindPossibleSplits(D,Lab); //find all
                                  possible thresholds
9.     [C_Crit,C_Threshold,C_PLab]=
       CalcCriterion(Dat,LabT,splits,CurLab); //For each
       threshold calculate criterion value
10.    [MXcrit,idx]=max(C_Crit);//Find max. criterion value
11.       if MXcrit > bestCrit
12.          bestCrit = MXcrit;
13.          P(RulN) = T(i);
14.          TH(RulN) = C_Threshold(idx);
15.          PLab = C_PLab(idx);
16.          RulN = RulN+1;
17.       end;
18.     end
19.     CurLab = ApplyRules(D,Lab,P,PLab,TH);
20.   endwhile;
21. end;
```

Fig. 2. The PTDL algorithm code

4 Experiments with Real Datasets

To compare results obtained with PTDL, HDT and other well established methods WEKA software was used with two popular rule extraction methods: C4.5 decision

tree and the Decision Table algorithm, as implemented in WEKA [19]. Results obtained with the neurofuzzy rule extraction system NefClass [12][17] are also given for comparison. The NefClass calculations were carefully optimized changing the number and the type of membership functions to obtain the best solution (the difference between accuracy and standard deviation).

4.1 Datasets

For tests six different datasets were used, all from the UCI machine learning database repository [16], except for Lancet data obtained from the authors of [16]. Each data represents a two class problem with mixed type of attributes. A summary of these datasets follows:

Appendicitis – small dataset with 7 attributes and 106 cases, 85 from the first class and 21 from the second class. From this dataset 2 most relevant features were selected using SSV tree and all tests were performed for these two features.

Cleveland Heart Disease (Cleveland) – 5 continuous attributes and 8 discrete, 303 vectors describing healthy and sick persons; 6 cases with missing values were removed, leaving 297 vectors.

Ionosphere – two different types of radar signals reflected from ionosphere; 351 vectors with 34 attributes.

Lancet dataset – 692 breast cancer cases, 235 malignant, 457 benign, characterized by age plus 10 binary indicators obtained from fine-needle aspirates of breast lumps, with the final diagnosis confirmed by biopsy.

Pima Indians Diabetes (Diabetes) – 768 cases describing results of tests for diabetes, with 500 healthy and 268 cases sick people, 8 features.

Wisconsin Breast Cancer (Wisconsin) – well known breast cancer data from a Wisconsin hospital, with 241 cases of malignant, and 458 of benign tumors, each case described by 9 discrete features.

4.2 Classification Results

10-fold stratified crossvalidation calculations on each dataset were performed using 5 algorithms that generate crisp and fuzzy rules, providing estimates of their generalization. Mean accuracy obtained on test partitions is presented in Table 1. The best results obtained for each dataset are marked as bold.

From Table 1 it is evident that the accuracy of the PTDL algorithm is almost always among the best among algorithms tested, creating a small number of rules and achieving in most cases best results. The Appendicitis dataset is very small and although NefClass has produced slightly better result it has used much larger number of fuzzy rules. For the Cleveland dataset only three P-rules were created by PTDL, reaching significantly higher accuracy than other systems. In the diabetes case all rule-based results are relatively poor, while MLP or SVM results on this dataset reach 77.5±2.5%, close to the simple linear discrimination analysis (LDA) reported in [20]. Therefore a single P-rule based on the shortest distance to two prototypes is sufficient in this case instead of the threshold based rules. PTDL did surprisingly well on the Ionosphere data, but HDT has an advantage here, achieving almost the same accuracy with only 3 rules. Insignificant differences are found on the Lancet data, with an

exception of C4.5 rules that are less accurate, with PTDL using just 3 P-rules and NefClass 85 F-rules. Also on the Wisconsin dataset only two P-rules were used to reach the highest accuracy with the lowest standard deviation.

Table 1. Classification results, accuracy (Acc) and standard deviation (Std) in %, the number of rules estimates the complexity of the model

	C4.5			Decision Table			NefClass			PTDL (Gini)			HDT (Gini)		
10 x CV	Acc	Std	Rules	Acc	Std	Rules	Acc	Std	Rules	Acc	Std	Rules	Acc	Std	Rules
Appendicitis	85.82	8,51	3	82,00	11,65	2	**87.73**	**8.6**	**33**	85.77	8.6	5	83.78	9,0	3
Cleveland	76.77	7,17	17	82.09	9,14	8	82.82	6.8	6	**84.21**	**5.1**	**3**	80.83	6,1	5
Diabetes	74,48	4,42	20	**74,87**	**5,16**	**32**	73.83	2,3	5	70.43	3.5	8	71.74	4.1	2
Ionosphere	**94.94**	**2.5**	**9**	93,06	3,66	23	72,67	6.7	9	93.45	3.1	15	93.15	2,9	3
Lancet	92,29	4,62	18	90,33	4,42	22	94.51	2.6	85	93,94	2,5	3	**94,51**	**2,1**	**4**
Wisconsin	94,58	2,87	11	95.75	1,65	20	94.86	2,6	6	**97.66**	**1.4**	**2**	96.93	1.85	1

5 Conclusions and Future Works

The prototype threshold decision list (PTDL) rule extraction algorithm presented in this paper is a simple method that creates a small number of accurate P-rules. Results obtained on several benchmark datasets are quite encouraging, even though only one criterion (Gini) has been considered so far and the simplest heterogeneous distance functions have been used. In a few cases these results are significantly better comparing to crisp rules obtained with C4.5 decision trees or decision table, or F-rules generated by the NefClass, a leading neurofuzzy algorithm. In some cases P-rules based on nearest neighbors rather than thresholds should lead to better results. As show in [15] prototype-based rules may be converted directly into fuzzy rules, therefore algorithms generating P-rules provide and interesting and little explored alternative to the neurofuzzy approaches.

The PTDL algorithm has the following advantages:

- it supports all attribute types;
- different types of rules may be generated, depending on the desired requirements: C-rules, P-rules or F-rules;
- it is simple to program and provides flexible decision borders;
- various distance functions may be used to improve generalization;
- small number of accurate and comprehensible rules are generated.

These properties make PTDL algorithm a very interesting and promising tool for data analysis. It can be further extended by adding various feature selection techniques. In PTDL the output of each rule is binary and each rule may operate on a different, independent, locally relevant subset of attributes. This is not quite true for

the nearest neighbor type of P-rules where common feature space is required for pairs of prototypes, although different pairs may operate in different subspaces.

Unfortunately the PTDL algorithm has some limitations. Its computational complexity is relatively high, requiring $O(N^2)$ operations for N training vectors to calculate all distances between the training vectors. All distance-based algorithms have $O(N^2)$ complexity and are thus much slower than simple decision trees and therefore quite costly to use on datasets with very large number of vectors. However, initial clusterization or a similar technique will significantly reduce the effort [21]. For example, joint information obtained from the whole dataset and clustered prototypes may be used, in the first step selecting best prototypes from all those obtained after clusterization, and then making a local search among training vectors close to the selected prototype to tune the rules. A combination of P-rules in both threshold and the nearest neighbor style may lead to the best of both worlds: localized decision regions combined with the hyperplanes, that sometimes are necessary for high accuracy (as in the case of Pima Indian Diabetes data). If a small number of features is used to evaluate similarity P-rules have simpler interpretation (the case is more similar to a given prototype than to any other) than combinations of features used in definition of hyperplanes.

These and other improvements of the PTDL algorithm will be explored in the near future. However, it is already clear that P-rules deserve at least as much attention as that enjoyed by the neurofuzzy systems.

Acknowledgement. Both authors are grateful for the support by the Polish Committee for Scientific Research, research grant 2005-2007.

References

[1] W. Duch, R. Setiono, J.M. Zurada, Computational intelligence methods for understanding of data. Proc. of the IEEE ,Vol. 92(5), pp. 771-805, 2004.

[2] The handbook of data mining, Ed. Nong Ye, Lawrence Erlbaum Associates, London 2003.

[3] J.R. Quinlan, C4.5: Programs for machine learning. Morgan Kaufman, CA, 1993.

[4] L. Breiman, J.H. Friedman, R.A. Oslhen, and C.J. Stone, Classification and Regression Trees. Belmont, CA: Wadsworth International Group, 1984.

[5] K. Grąbczewski and W. Duch, The separability of split value criterion. 5th Conference on Neural Network and Soft Computing, Polish Neural Network Society, Zakopane, Poland, 2000, pp. 201-208.

[6] N. Jankowski, K. Grąbczewski, W. Duch, A. Naud and R. Adamczak, Ghostminer data mining software, http://www.fqspl.com.pl/ghostminer/

[7] W. Pedrycz, Fuzzy set technology in knowledge discovery, Fuzzy Sets and Systems Vol. 98, pp. 279-290, 1998.

[8] W. Duch, Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics Vol. 29(4), pp. 937-968, 2000.

[9] W. Duch and K. Grudziński, Prototype based rules - a new way to understand the data. Proc. of the International Joint Conference on Neural Networks (IJCNN) 2001, Washington D.C, USA, pp. 1858-1863.

[10] K. Grąbczewski and W. Duch, Heterogenous forests of decision trees. Springer Lecture Notes in Comp. Science, Vol. 2415, pp. 504-509, 2002.
[11] B. Kosko, Neural Networks and Fuzzy Systems. Prentice Hall, 1992.
[12] D. Nauck, F. Klawonn and R. Kruse, Foundations on Neuro-Fuzzy Systems. J. Wiley, New York, 1997.
[13] S.K. Pal and S. Mitra, Neuro-Fuzzy Pattern Recognition. J. Wiley, New York, 1999.
[14] D.R. Wilson, T.R. Martinez, Improved Heterogeneous Distance Functions, Journal of Artificial Intelligence Research, Vol. 6, pp. 1-34, 1997.
[15] W. Duch and M. Blachnik, Fuzzy rule-based system derived from similarity to prototypes, Lecture Notes in Computer Science, Vol. 3316, pp. 912-917, 2004.
[16] C.J. Mertz and P.M. Murphy, UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-databases.
[17] D. Nauck and U. Nauck, http://fuzzy.cs.uni-magdeburg.de/nefclass/nefclass.html
[18] A.J. Walker, S.S. Cross, and R.F. Harrison, Visualization of biomedical datasets by use of growing cell structure networks: a novel diagnostic classification technique. Lancet, Vol. 354, pp. 1518-1522, 1999.
[19] I.H. Witten and E. Frank, Data Mining: Practical machine learning tools and techniques, 2nd Ed, Morgan Kaufmann, San Francisco, 2005.
[20] D. Michie, D.J. Spiegelhalter, and C.C. Taylor (eds), Machine Learning, Neural and Statistical Classification. Ellis Horwood, 1994.
[21] G. Shakhnarovish, T. Darrell, P. Indyk (eds), Nearest-Neighbor Methods in Learning and Vision, MIT Press, 2005.