A general purpose separability criterion for classification systems

Krzysztof Grąbczewski and Włodzisław Duch,
Department of Computer Methods, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland.
E–mail: {kgrabcze,duch}@phys.uni.torun.pl

Abstract

A separability criterion which can be applied to different classification algorithms is presented. It is designed to find out the best (from the point of view of separation of different classes) divisions of continuous features or the best combinations of symbolic values of discrete features. It has been tested on several datasets to construct classification decision trees and decision rules. It can also be useful in continuous features discretization and estimation of feature's importance, which forms a first step of neural algorithms for extraction of logical rules from data.

I. INTRODUCTION

THERE are many different approaches to the task of learning classification from examples. Many of them strive to obtain the best possible accuracy without bothering about the comprehensibility of the knowledge gained. Such systems play a role of 'black boxes' which are not able to give any explanation of their decisions. In many fields (for instance in medicine) it is extremely important to understand decisions of computer support systems. Very often we are interested not only in assigning a class label to the input data, but also in an explanation of the decision and its analysis. In such cases we need a short, precise and comprehensible description of classification problem which usually means a small set of crisp logical rules. From this point of view the most important property of a classification system is its ability to find the most informative features describing the objects that are classified, because it guarantees as compact decision rules as possible.

If the objects are described by continuous features it is extremely important to divide the feature space into possibly the smallest number of intervals, giving the best separation of different classes. Definition of the linguistic variables is the first step in most methods of extraction of logical rules from data, including our MLP2LN method [1]. In the next section a criterion for selection of optimal intervals is presented. In the third section applications of this criterion to decision trees, logical rule generation and feature selection methods is discussed. The fourth section presents a few illustrative applications and a summary closes this paper.

II. THE CRITERION

IN this section a criterion offering very good separability of objects with different class labels is presented. 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.

The criterion is based on a simple idea that the best split is the one that separates the largest number of pairs of objects from different classes. 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.

In all cases we can define *left side* (*LS*) and *right side* (*RS*) of a split value s of feature f for given dataset D:

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

where $f(x)$ is the f's feature value for the data vector x.

The *separability of a split value s* is defined as:

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

where C is the set of classes and D_c is the set of data vectors from D which belong to class c.

The higher the separability of a split value the better. According to this criterion the best split value is the one which separates the maximal number of pairs of vectors from different classes and among all the split values which satisfy this condition - the one which separates the smallest number of pairs of vectors belonging to the same class.

Notice that one of the most important property of the separability criterion is that points beyond the borders of feature values existing in the dataset have the separability equal to 0, and all points between the borders have positive separability. This means that 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 of maximal separability.

When the feature being examined is continuous and there are several different split values of maximal separability close to each other, the most reasonable heuristics to use is to select the split value closest to the average of all of them. To avoid such situations it is good to examine split values which are natural for a given dataset (i.e. the middles 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 selection of the best split value may be arbitrary.

III. APPLICATIONS

THE separability criterion can be used in several different applications. First of all it can be used to build classification decision trees. Thanks to the ease of finding the best cuts generated trees may be small, which also means that they can be converted into a small number of crisp logical rules. Moreover, it can be used in algorithms aiming at discretization of continuous features. It can also serve for analysis of feature importance and for selection of features.

A. Decision tree

Construction of the decision trees is a natural application of the separability criterion. The simplest method of building such a tree is to use best first search. We evaluate the separability of each possible cut point of each continuous feature and of each subset of the set of values of each discrete feature. The best splits are selected and the space (and dataset as well) is divided into two parts by the first two branches of the binary tree. The criterion is then applied recursively to each of the resulting parts of the input space (with their corresponding

data subsets). The tree is finished when it classifies the data with maximal accuracy. 100% accuracy is possible only if there are no contradictory examples in the dataset.

The accuracy of 100% usually means overfitting, so to avoid that we use a special technique **to maximize the generalization** capacity of the resulting tree. The simplest way to improve generalization of a decision tree is by pruning the leaves and branches responsible for classification of single or very few data vectors. To find the best way of pruning 10-fold crossvalidation for the training set is performed. In each crossvalidation pass unseen samples are used to find the best way of tree pruning. We can not memorize which leaves overfit the data, because the final tree may be quite different than the tree built for the training data available during crossvalidation (90% of the data in 10-fold crossvalidation). Therefore an optimal *degree of pruning* is determined. Pruning with the degree of n means cutting off all the pairs of leaves, which reduce the number of errors of their parent by not more than n. In each pass of the crossvalidation the number of errors counted for the test part of the data is checked. The optimal degree of pruning is the maximal degree (natural number) corresponding to the minimal total crossvalidation test error (sum of all crossvalidation test errors).

The best first search algorithm applied to the problem of finding an optimal decision tree seems to be a bit different than the original algorithm, as there is no chance to go back to nodes of the search tree visited earlier. It is so because growing the decision tree can not enlarge the error value. To diminish this drawback we use beam search instead of best first search. It is really capable of finding better results, however it is slightly more time consuming.

B. Rule generator

It is easy to convert the decision tree into a set of crisp logical rules (each branch of the tree represents one rule). However, the rules containing premises describing all the nodes from the root of the tree to it's leaves can be more complex than necessary. Especially in bigger trees it may turn out that the decisions made at the very beginning are not very important for classification of data vectors which end up in a leaf. They are very important for a large data set, but not necessarily for smaller, localized samples. Therefore redundant rule antecedents should be removed. To find out which premises are spurious they are deleted one by one and a check of the accuracy is made. If the accuracy is decreased the premise should be kept.

C. Discretization and feature selection methods

The separability criterion can be used in several different ways to discretize a continuous feature. For instance the same algorithm can be followed as for the construction of a decision tree, but the possible cut points should be checked only for that feature. After several recursive runs as large number of cut points as necessary is obtained. The recursive process can also be stopped when the subsequent splits do not significantly improve the separability. The recursive process is necessary, because usually features have just one or two maximal cut points and when the data is split into two parts at least one best split value for each of the parts will certainly be found in the next stage.

Sometimes all the split values of a given feature have very low separability, and together with some cut points of another feature they give much better results. In the process of building a decision tree beam search is quite efficient in finding such combinations of features. However, in discretization process it is useless unless we descretize all the features simultaneously. Searching for all feature split values at the same time may take into account mutual

interaction of features, so it is much more reasonable.

Given a discretization of all the continuous features the separability of a single split value can be easily generalized to the separability of a set of split values, which can be used for feature selection.

IV. RESULTS

SINCE the main goal is a comprehensive description of data given for classification the results are presented as sets of crisp logical rules. The algorithms based on the separability criterion presented here have been tested on well known benchmark datasets. Most of them come from the UCI repository [7].

A. Appendicitis data

The appendicitis dataset contains only 106 cases, with 8 attributes (results of medical tests), and 2 classes: 88 cases with acute appendicitis and 18 cases with other problems. We have selected this dataset because a very simple classification rule has been found by Weiss and Kapouleas [8] using their PVM (Predictive Value Maximization) approach. Since PVM makes exhaustive search testing all possible simple rules (therefore it is applicable to very small datasets only) this solution should indeed be very close to the simplest possible.

The decision tree built with SSV converted into logical rules gives just two rules per class. Because there are no "don't know" answers, only the rules for one of the cases need to be presented, the other class can be summarized using the else condition.

First rule obtained using separability criterion gives 91.5% accuracy. The second one covers additional three data vectors, increasing the accuracy to 94.3%. The example of the second rule confirms that decisions made at the top of the tree do not have to be important to distinguish the classes in small local areas. In Table I results of different methods for this dataset are compared.

R_1: HNEA $< 7520.5 \wedge$ MBAP $< 12 \rightarrow$ class 0
R_2: HNEA $\in (9543.5, 9997.5) \rightarrow$ class 0
R_3: ELSE $\rightarrow$ class 1

TABLE I
Results for the appendicitis dataset

Method	Error	Leave-one-out error
k-NN	–	17.9%
Bayes	11.3%	17.0%
CART	9.4%	15.1%
MLP+backprop	9.8%	14.2%
PVM [8]	8.5%	10.4%
C-MLP2LN [1], 1 neuron	8.5%	(est.) 10.4%
C-MLP2LN, 2 neurons	5.7%	
SSV rules	5.7%	

B. Hypothyroid data

This is somewhat larger dataset [7], with 3772 cases for training, 3428 cases for testing, 22 attributes, and 3 classes: primary hypothyroid, compensated hypothyroid and normal (no hypothyroid). It has already been quite thoroughly examined with different systems. It seems impossible to find a better solution than the already known - especially, that some of the results use global optimization strategies like ASA (adaptive simulated annealing [4]).

Rules obtained from the separability criterion are:

R_1: TSH $>$ 6.05 $\wedge$ FTI $<$ 64.72 $\wedge$ thyroid-surgery = 0 $\rightarrow$ class 1
R_2: TSH $>$ 6.05 $\wedge$ FTI $>$ 64.72 $\wedge$ thyroid-surgery = 0 $\wedge$ on-thyroxine = 0 $\wedge$ TT4 $<$ 150.5 $\rightarrow$ class 2
R_3: ELSE $\rightarrow$ class 3

These rules give very high accuracy, matching the best results although the solution has been found with fully automatic rule extraction approach. Results are summarized in Table 2. It is worth noting that the error of the best neural network classifiers is still twice as large (1.5%) as the error made by these simple rules.

TABLE II
Results for the hypothyroid dataset

Method	Train. error	Test error
k-NN	–	4.73%
Bayes	2.97%	3.94%
MLP+backprop	0.40%	1.55%
Cascade correl.	0.00%	1.52%
C-MLP2LN	0.32%	0.93%
C-MLP2LN rules + ASA [4]	0.11%	0.64%
PVM	0.21%	0.67%
CART	0.21%	0.64%
SSV rules	0.21%	0.67%

C. Other datasets

We have also tested our method on some other datasets such as Wisconsin breast cancer, Cleveland heart and mushrooms data.

For the Wisconsin breast cancer data we have obtained a very simple (compared to others) set of rules which is 97.4% accurate:

R_1: F4 $<$ 2.5 $\wedge$ F7 $<$ 1.5 $\rightarrow$ class 1
R_2: F4 $<$ 2.5 $\wedge$ F2 $<$ 5.5 $\rightarrow$ class 1
R_3: F4 $>$ 2.5 $\wedge$ F7 $<$ 2.5 $\wedge$ F6 $<$ 3.5 $\rightarrow$ class 1
R_4: ELSE $\rightarrow$ class 2

Two rules describing the Cleveland heart data are 85.8% accurate:

R_1: ca = 0.0 $\wedge$ (thal = 0 $\vee$ exang = 0) $\rightarrow$ class 1
R_2: cp $\neq$ 2 AND slope $\neq$ 2 $\rightarrow$ class 1

R_3: ELSE $\rightarrow$ class 2

For the mushroom dataset SSV tree has easily found 100% accurate solution which can be described as three logical rules. It is one of the simplest equivalent sets of rules we know of.

R_1: odor $\notin$ {a,l,n} $\rightarrow$ class 1
R_2: odor $\in$ {a,l,n} $\wedge$ spore-print-color $\in$ {r,w} $\wedge$ population = v $\wedge$ habitat $\notin$ {l,p} $\rightarrow$ class 1
R_3: odor $\in$ {a,l,n} $\wedge$ spore-print-color $\in$ {r,w} $\wedge$ population $\neq$ v $\wedge$ gill-size $\neq$ b $\rightarrow$ class 1
R_4: ELSE $\rightarrow$ class 0

V. Summary

The results we have obtained for several benchmark datasets prove that methods based on the separability criterion introduced in this paper produce very accurate and compact description of the data. Moreover, decision tree building algorithm based on beam search finds small trees (which also means simple logical rules) fully automatically. The parameters of the method (like the beam width or the number of crossvalidation parts) do not need to be precisely adjusted in order to obtain good results. This makes the algorithm a powerful, user-independent tool for extraction of crisp logical rules from raw data.

Acknowledgement: We would like to thank the Polish Committee for Scientific Research, grant no. 8T11F 014 14 for partial support.

References

[1] W. Duch, R. Adamczak and K. Grąbczewski, Extraction of logical rules from neural networks, *Neural Processing Letters* 7: 211-219, 1998

[2] R. Andrews, J. Diederich, A.B. Tickle, A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks, *Knowledge-Based Systems* 8 (1995) 373–389.

[3] R. Andrews, S. Geva, Rule extraction from a constraint back propagation MLP, Proc. 5th Australian Conference on Neural Networks, Brisbane, Queensland (1994), pp. 9-12

[4] W. Duch, R. Adamczak, K. Grąbczewski and G. Żal, Hybrid neural-global minimization method of logical rule extraction, *Int. Journal of Advanced Computational Intelligence* (in print, 1999)

[5] W. Duch, G.H.F. Diercksen, Feature Space Mapping as a universal adaptive system, *Computer Physics Communications* 87 (1995) 341–371

[6] M. Ishikawa, Rule extraction by succesive regularization, in: Proc. of the 1996 IEEE ICNN, Washington, June 1996, pp. 1139–1143.

[7] C.J. Mertz, P.M. Murphy, UCI repository of machine learning databases, available at the address
http://www.ics.uci.edu/pub/machine-learning-databases;

[8] S.M. Weiss, I. Kapouleas, in: J.W. Shavlik and T.G. Dietterich, Eds. *Readings in Machine Learning*, Morgan Kauffman Publ. CA 1990