Induction of Multivariate Decision Trees by Using Dipolar Criteria

Leon Bobrowski[1,2] and Marek Krętowski[1]

[1] Institute of Computer Science, Technical University of Białystok, Poland
[2] Institute of Biocybernetics and Biomedical Engineering, PAS, Poland
mkret@ii.pb.bialystok.pl

Abstract. A new approach to the induction of multivariate decision trees is proposed. A linear decision function (hyper-plane) is used at each non-terminal node of a binary tree for splitting the data. The search strategy is based on the dipolar criterion functions and exploits the basis exchange algorithm as an optimization procedure. The feature selection is used to eliminate redundant and noisy features at each node. To avoid the problem of over-fitting the tree is pruned back after the growing phase. The results of experiments on some real-life datasets are presented and compared with obtained by state-of-art decision trees.

1 Introduction

Decision trees are hierarchical, sequential classification structures that recursively partition the feature space. The tree contains terminal nodes (leaves) and internal (non-terminal) nodes. A terminal node generates no descendants, but is designated by a class label, while an internal node contains a split, which tests the value of an expression of the attributes. Each distinct outcome of the test generates one child node, hence all non-terminal nodes have two or more child nodes. An object is classified through traversing a path from the root node until a terminal node. At each non-terminal node on the path, the associated test is used to decide which child node the feature vector is passed to. At the terminal node the class label is used to classify the input vector.

Decision trees are extensively investigated especially in statistics, machine learning and pattern recognition (see [13] for a very good multi-disciplinary survey). Most of the research effort focuses on the univariate tests for symbolic or numeric (continuous-valued) attributes (e.g. C4.5 [14], CHAID [10]). An univariate test compares a value of a single attribute to a constant, so this is equivalent to partitioning the set of observations with an axis-parallel hyper-plane. A multivariate decision tree may use as the split an expression, which exploits more than one feature (see Fig. 1). A special case of the multivariate tree that we are particular interested in is an oblique decision tree ([12]). A test in such type of the tree uses a linear combination of the attributes.

Several methods for generating multivariate trees have been introduced so far. One of the first trials was done in CART (*Classification And Regression Trees*) [5]. The system is able to search for a linear combination of the continuous-valued attributes instead of using only a single attribute. The CART system has the

D.A. Zighed, J. Komorowski, and J. Żytkow (Eds.): PKDD 2000, LNAI 1910, pp. 331–336, 2000.

strong preference for univariate tests and chooses multivariate one very rare. In LMDT (*Linear Machine Decision Trees*) [6] each test is constructed by training a linear machine and eliminating variables in a controlled manner. Murthy et al. [12] introduce OC1 (*Oblique Classifier 1*), the algorithm that combines deterministic and randomized procedures to search for a good tree. The method was applied to classify a set of patients with breast cancer and showed excellent accuracy. Another tree was proposed by Chai et al., [7]. BTGA (*Binary Tree-Genetic Algorithm*) uses the maximum impurity reduction as the optimality criterion of linear decision function at each non-terminal node. The modified BTGA was used to the pap smear classification and demonstrated the high sensitivity along with the lowest possible false alarm rate. Recently, *Ltree* (*Linear tree*) was presented, which combines a decision tree with a linear discrimination by means of constructive induction [9]. At each node a new instance space is defined by insertion of new attributes that are projections over the hyper-planes given by a linear discrimination function and new attributes are propagated downward.

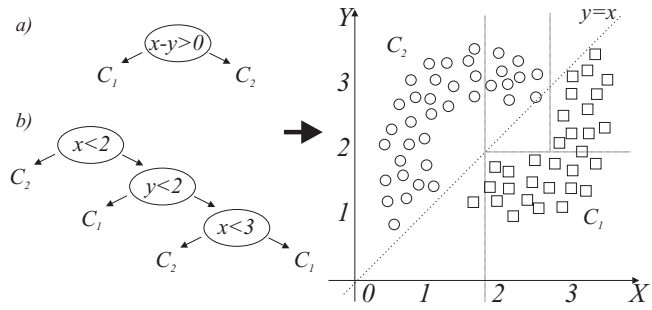

Fig. 1. An example of the simple two-class problem and its possible solutions by various decision trees: (a) multivariate and (b) univariate. If an associated test is true then we choose the left outcome, otherwise the right one.

2 Dipolar Criteria

We will take into account the N-dimensional feature vectors $\mathbf{x} = [x_1, ..., x_N]^T$ ($x_i \in \{0, 1\}$ or $x_i \in R^1$) which can belong to one of the K classes ω_k ($k = 1, ..., K$). The learning set $C_k = \{\mathbf{x}^j(k)\}(j = M_{k-1} + 1, ..., M_{k-1} + m_k)$ contains m_k feature vectors $\mathbf{x}^j(k)$ from the class ω_k, where M_{k-1} is the number of objects $\mathbf{x}^j$ in the first $(k-1)$ sets C_k.

The feature space could be divided into two regions by the following hyper-plane $H(\mathbf{w}, \theta) = \{\mathbf{x} : \langle \mathbf{w}, \mathbf{x} \rangle = \theta\}$, where $\mathbf{w} = [w_1, ..., w_N]$ ($\mathbf{w} \in R^N$) is the weight vector, θ is the threshold and $\langle \mathbf{w}, \mathbf{x} \rangle$ is the inner product. In the special case the weight $\mathbf{w}$ could be reduced to the unit vector $\mathbf{e}^i = [0, ..., e_i = 1, ..., 0]$. The hyper-plane $H(\mathbf{e}^i, \theta)$ is parallel to all but the i-th axis of the feature space. The learning sets are linearly separable if each of these sets C_k could be separated by some hyper-plane from the sum of all remaining sets C_i.

For simplicity of notion we introduce augmented feature space, where $\mathbf{y} = [1, x_1, ..., x_N]^T$ is the augmented feature vector, $\mathbf{v} = [-\theta, w_1, ..., w_N]$ is the weight vector and $H(\mathbf{v}) = \{\mathbf{y} : \langle \mathbf{v}, \mathbf{y} \rangle = 0\}$ is the hyper-plane.

Dipoles We recall some basic definition, for more detailed description please refer to [3] and [4]. A dipole is a pair $(\mathbf{x}^j(k), \mathbf{x}^{j'}(k'))$ of the feature vectors $\mathbf{x}^j(k)$ from the learning sets C_k, where $0 \leq j < j' \leq m$, and m is the number of the vectors in all learning sets. We call the dipole $(\mathbf{x}^j(k), \mathbf{x}^{j'}(k'))$ *mixed* if and only if the objects constituting it belong to the different learning sets C_k $(k \neq k')$. Similarly, a pair of the vectors from the same class ω_k constitutes the *pure* dipole.

Hyper-plane $H(\mathbf{v})$ splits the dipole $(\mathbf{y}^i, \mathbf{y}^j)$ if and only if: $\langle \mathbf{v}, \mathbf{y}^i \rangle \cdot \langle \mathbf{v}, \mathbf{y}^j \rangle < 0$. It means that the input vectors $\mathbf{y}^i$ and $\mathbf{y}^j$ are situated on opposite sides of the dividing hyper-plane. The necessary and sufficient condition for preserving the separability of the learning sets by the hyper-plane is the division of all mixed dipoles by the hyper-plane. The tree induction is connected with the repeated search for such a hyperplane $H(\mathbf{v})$ which "divides a possible *high* number of mixed dipoles and a possible *low* number of pure ones". These requests are related to demand for a low error rate of the resulting tree.

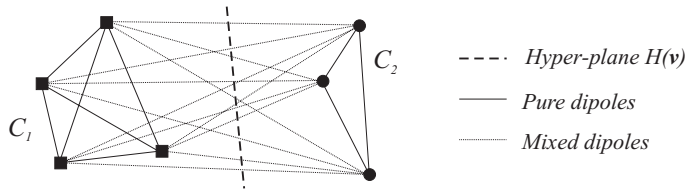

Fig. 2. A simple situation where only one hyper-plane $H(\mathbf{v})$ is enough to divide all mixed dipoles and no pure one.

Dipolar Criterion Function The search for "good" hyper-planes $H(\mathbf{v})$ could be based on minimization of some criterion functions like the perceptron one [8]. The dipolar criterion function $\Psi(\mathbf{v})$ has been proposed for this purpose [4]. Lets first define a penalty function, which is associated with each input vector $\mathbf{y}^j(k)$:

$$\varphi_j^+(\mathbf{v}) = \begin{cases} \delta^j - \langle \mathbf{v}, \mathbf{y}^j \rangle & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle < \delta^j \\ 0 & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle \geq \delta^j \end{cases} \tag{1}$$

and

$$\varphi_j^-(\mathbf{v}) = \begin{cases} \delta^j + \langle \mathbf{v}, \mathbf{y}^j \rangle & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle < -\delta^j \\ 0 & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle \geq -\delta^j \end{cases} \tag{2}$$

where $\delta^j (\delta^j \geq 0)$ is the parameter defining the margin. If $\delta^j = 0$, the penalty function is related to the error correction algorithm used in the perceptron [8]. We usually assume, that the margin is equal to 1. We can force to move $H(\mathbf{v})$ to obtain situation when $\mathbf{y}^j(k)$ is on a positive (if we use $\varphi_j^+(\mathbf{v})$) or a negative $(\varphi_j^-(\mathbf{v}))$ side of $H(\mathbf{v})$. With each dipole we associate two functions, depending on our goal. Thus for a mixed dipole $(\mathbf{y}^i, \mathbf{y}^j)$ we use two function with opposite signs, because we are interested in cutting all dipoles of such type:

$$\varphi_{ij}^m(\mathbf{v}) = \varphi_i^+(\mathbf{v}) + \varphi_j^-(\mathbf{v}) \quad (\text{or } \varphi_{ij}^m(\mathbf{v}) = \varphi_j^+(\mathbf{v}) + \varphi_i^-(\mathbf{v})) \tag{3}$$

To avoid dividing a pure dipole both end-points of it should be situated on the same side of the hyper-plane $H(\mathbf{v})$, so the penalty function is defined as follows:

$$\varphi_{ij}^p(\mathbf{v}) = \varphi_i^+(\mathbf{v}) + \varphi_j^+(\mathbf{v}) \quad (\text{or } \varphi_{ij}^p(\mathbf{v}) = \varphi_j^-(\mathbf{v}) + \varphi_i^-(\mathbf{v})) \tag{4}$$

The choice of the form of $\varphi_{ij}^m(\mathbf{v})$ ($\varphi_{ij}^p(\mathbf{v})$) is devoted to an orientation of a dipole and for the simplicity reason we do not take into account in this paper. More detailed explanation of the orientation one can find in [4]. The dipolar criterion function $\Psi(v)$ could be represented in the following manner:

$$\Psi(\mathbf{v}) = \sum_{(i,j)\in I_p} \alpha_{ij} \cdot \varphi_{ij}^p(\mathbf{v}) + \sum_{(i,j)\in I_m} \alpha_{ij} \cdot \varphi_{ij}^m(\mathbf{v}), \tag{5}$$

where α_{ij} determines a relative importance (price) of dipole $(\mathbf{y}^i, \mathbf{y}^j)$ and $I_p(I_m)$ is the set of the pure (mixed) dipoles.

It is worth to emphasize that the mechanism of dipole's prices is very flexible. In all experiments presented in the paper we used a very simple strategy - we differentiate only prices for mixed (1.0) and pure dipoles (0.01). We work on a little bit more complicated strategy which exploits the length of dipoles.

Because $\Psi(\mathbf{v})$ is the sum of the convex and piece-wise linear penalty functions $\varphi_j^+(\mathbf{v})$ and $\varphi_j^-(\mathbf{v})$, $\Psi(\mathbf{v})$ is also a function of this type. The basis exchange algorithms, similar to linear programming methods, have been developed as an efficient tool for minimization of such a function [2]. In fact the minimization has been carried out by combining the basis exchange algorithms with a search for the adequate orientations of dipoles [4].

3 Multivariate Tree Induction

Most of the existing tree induction systems proceed in a greedy, top-down fashion. At each non-terminal node, starting from the root, the best split is learned by using a criterion of optimality. The learned decision function divides the training subset into two (or more) subsets generating child nodes. The process is repeated at each newly created child node until a stopping condition is satisfied and the node is declared as a terminal node.

We are taking into consideration the binary decision tree. At the i-th non-terminal node the set S_i of the input vectors from the training data, which reaches the node is divided into two subsets S_i^L and S_i^R by the hyper-plane $H(\mathbf{v}_i)$ (or in a case of an univariate tree by $H(\mathbf{e})$):

$$S_i^L = \{\mathbf{y} : \mathbf{y} \in S_i \text{ and } \langle \mathbf{v}_i, \mathbf{y} \rangle \geq 0\} \tag{6}$$

$$S_i^R = \{\mathbf{y} : \mathbf{y} \in S_i \text{ and } \langle \mathbf{v}_i, \mathbf{y} \rangle < 0\} \tag{7}$$

All feature vectors from the set S_i^L constitute the left child of the i-th node and S_i^R the right one. To find the optimal dividing hyper-plane $H(\mathbf{v}_i)$ the dipolar criterion function $\Psi(\mathbf{v}_i)$ is used, which is constituted exclusively on dipoles created from the feature vectors belonging to S_i.

Feature Selection At each internal node of the tree we are interested in obtaining the simplest possible test, hence we try to exclude noisy (or/and irrelevant) features. This way we increase understandability of the model and what is even more important, we can improve performance. There exist many feature selection algorithms (for a good review, in the context of multivariate trees see [6]). Our goal is not the comparison of various methods, so we have to choose one.

We use Heuristic Sequential Search proposed in [6]. It is a combination of Sequential Backward Elimination (SBE) and Sequential Forward Selection (SFS) and it works as follows: firstly finds the best test based on all features and similarly the best test using only one feature. Depending which one is better SFS or SBE search is chosen. The only problem which we have to solve is: how to compare various subsets of features. We prefer the subset (and the hyper-plane build on it), which divides more mixed dipoles (and less pure ones, when a tie is observed). During the feature selection process one cannot forget the problem of "underfitting" the training data. The number of objects used to find a test should be several times greater than the number of attributes used [8] (in all experiments: "x5"). Such a situation occurs often near the leaves of the tree.

Avoiding Over-fitting It is well known fact, that the data over-fitting can decrease the classifier's performance in a significant way, especially in a noisy domain. There exist two the most common approaches to avoid this problem and to determine the correct tree size: to stop the tree growing before the perfect classification of the training data or to post-prune the tree after the full partitioning of the instance space ([5]). Although the post-pruning is definitely superior, but usually both techniques are combined and in our system we decided to follow this idea. We use a very simple stopping rule - when the number of cases in a node is lower than N_{stop} then the node is declared as terminal ($N_{stop} = 5$ in all experiments). We have chosen also one of the simplest pruning methods: *reduced-error pruning* ([14]). It exploits a separate part of the training data (in all experiments 30%) as a validation set used during the pruning phase to determine the error rates of the sub-tree.

4 Experimental Results

We have tested our approach on several datasets taken from UCI repository ([1]). The left part of Table 1 shows the classification error rates obtained by our method and C4.5, LMDT (results are taken from [11]). To estimate the error rate of an algorithm we use the testing set (when provided) or the ten-fold stratified cross-validation procedure. In the right part of Table 1 we try to present the complexity of obtained models. We include the number of nodes in the whole tree, but it is not the best measure of the complexity, especially if we want to compare univariate and multivariate trees. And even if we compare the multivariate trees we need to be careful (e.g. in the multi-class domain LMDT in each node includes more than one hyper-plane). In case of trees generated using our method we also provide the number of features used in all tests.

5 Conclusions

In the paper we presented how to induce decision trees based on dipolar criteria. The preliminary experimental results indicate that both classification accuracy and complexity are comparable with the results obtained by other systems. Furthermore many places for the possible improvements still exist. Especially the pruning strategy is very simple and we hope that applying more sophisticated method will help to improve the results. The feature selection at each non-terminal node is the most time consuming part of the induction process. At the

moment we are working on integrating the feature selection with searching for the best hyper-plane, which will significantly reduce the learning time. One of the future research directions could be devoted to incorporating the variable misclassification cost into the decision tree induction. We think that it could be easily done only by a modification of dipole's prices.

Table 1. The error rates (left part) and the number of nodes in the whole tree (right part; the number of features used in all test is given in brackets) of compared systems

Dataset	Our method	C4.5	LMDT	Our method	C4.5	LMDT
breast-w	0.038	0.042	0.035	3 (7)	11	3
heart	0.191	0.196	0.163	3 (10)	23	3
housing	0.248	0.221	0.251	23 (71)	36	11
pima	0.233	0.242	0.249	13 (22)	18	11
smoking	0.292	0.305	0.350	69 (139)	1	61
vehicle	0.243	0.277	0.215	25 (122)	65	13
waveform	0.213	0.261	0.176	11 (65)	54	4

Acknowledgements: The presented work was partially supported by the grant 8 T11E 029 17 from the State Committee for Scientific Research (KBN) in Poland and by the grant W/II/1/2000 from Technical University of Białystok.

References

1. Blake, C., Merz, C.: UCI Repository of machine learning databases, available online: http://www.ics.uci.edu/ mlearn/MLRepository.html (1998)
2. Bobrowski, L.: Design of piecewise linear classifiers from formal neurons by some basis exchange technique. Pattern Recognition **24**(9) (1991) 862–870
3. Bobrowski, L.: Piecewise-linear classifiers, formal neurons and separability of the learning sets, In: Proc. of 13^{th} Int. Conf. on Pattern Recognition (1996) 224–228
4. Bobrowski, L.: Data mining procedures related to the dipolar criterion function, In: Applied Stochastic Models and Data Analysis (1999) 43–50.
5. Breiman, L., Friedman, J., Olshen, R., Stone C.: Classification and Regression Trees. Wadsworth Int. Group (1984)
6. Brodley, C., Utgoff, P.: Multivariate decision trees. Mach. Learning 19 (1995) 45–77
7. Chai, B., Huang, T., Zhuang, X., Zhao, Y., Sklansky, J., Piecewise-linear classifiers using binary tree structure and genetic algorithm. Pattern Recognition **29**(11) (1996) 1905–1917
8. Duda, O.R., Heart, P.E.: Pattern Classification and Scene Analysis. J. Wiley (1973)
9. Gama, J., Brazdil, P.: Linear tree. Inteligent Data Analysis **3**(1) (1999) 1–22
10. Kass, G. V.: An exploratory technique for investigating large quantities of categorical data, Applied Statistics **29**(2) (1980) 119-127
11. Lim, T., Loh, W., Shih, Y.: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Mach. Learning **40**(3) (2000) 203–228
12. Murthy, S., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research 2 (1994) 1–33
13. Murthy, S.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery 2 (1998) 345–389
14. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)