An Evolutionary Algorithm for Oblique Decision Tree Induction

Marek Krętowski

Faculty of Computer Science, Białystok Technical University
Wiejska 45a, 15-351 Białystok, Poland
e-mail: mkret@ii.pb.bialystok.pl

Abstract. In the paper, a new evolutionary approach to induction of oblique decision trees is described. In each non-terminal node, the specialized evolutionary algorithm is applied to search for a splitting hyperplane. The feature selection is embedded into the algorithm, which allows to eliminate redundant and noisy features at each node. The experimental evaluation of the proposed approach is presented on both synthetic and real datasets.

1 Introduction

Decision trees (DT) have been extensively investigated in statistics, machine learning and pattern recognition (see [11] for a very good multi-disciplinary survey) and now they are one of the most popular classification tools. Clones of the most renowned induction algorithms e.g. CART [4] or C4.5 [12] are included in virtually every data mining system. Advantages of the DT-based approach include natural representation, fast induction (especially in case of univariate splits) and ease of interpretation of obtained predictions.

The simplest variant of a decision tree is so called *univariate* tree. In each non-terminal node, it exploits a test, which is based on a single attribute. Such a split is equivalent to partitioning the set of objects with an axis-parallel hyperplane. The use of univariate tests may lead to very complicated classifier if decision borders are not axis-parallel. *Oblique* decision trees allow to avoid the aforementioned problem by using more flexible test based on a linear combination of attributes. It should be noted however that finding the optimal oblique split is generally much more difficult.

Several algorithms for oblique trees induction have been introduced so far. One of the first trials is done in CART [4]. The system is able to search for linear combinations of the continuous-valued attributes and also to simplify them by feature elimination procedure. Generally CART prefers univariate tests and chooses oblique one very rare. Murthy *et al.* [10] introduce OC1 (*Oblique Classifier 1*), the algorithm that combines deterministic (hill-climbing) 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 interesting approach was proposed by Gama *et al.* [8]. Their *Linear tree* system combines an

L. Rutkowski et al. (Eds.): ICAISC 2004, LNAI 3070, pp. 432–437, 2004.

univariate tree with a linear discrimination by means of constructive induction. 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. A system proposed by Bobrowski *et al.* [2] is based on the dipolar criterion functions and exploits the basis exchange algorithm as an optimization procedure. The system can be treated as a predecessor of the approach proposed in the paper.

Evolutionary algorithms (EA) [9] are stochastic search techniques, which have been successfully applied to many optimization problems. The success of EAs is attributed to their ability to avoid local optima, which is their main advantage over greedy search methods. One of the first applications of evolutionary approach to induction of oblique tree is presented in Binary Tree-Genetic Algorithm (BTGA) system [6]. In this approach, a linear decision function at each non-terminal node is searched by standard genetic algorithm with binary representation. The maximum impurity reduction is adopted as the optimality criterion. Recently, Cantu-Paz *et al.* [5] present two extensions of the OC1 algorithm by using two standard algorithms: (1+1) evolution strategy and simple genetic algorithm. The empirical results show that their system is able to find competitive classifiers quickly and that EA-based systems scale better than traditional methods to size of the training dataset.

In the paper, a new specialized evolutionary algorithm for searching the hyper-plane in non-terminal modes of the oblique decision tree is proposed. The most important innovations concern the fitness function and genetic operators.

2 Oblique Tree Induction

Lets assume that a learning set is composed of M objects belonging to one of K classes. Each object is described by a feature vector $\mathbf{x}^j = [x_1^j, ..., x_N^j]^T$ $(j = 1, ..., M)(\mathbf{x}^j \in R^N)$. The feature space could be divided into two regions by the 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.

A *dipole* [3] is a pair $(\mathbf{x}^i, \mathbf{x}^j)$ of the feature vectors. The dipole is called *mixed* if and only if the objects constituting it belong to two different classes and a pair of the vectors from the same class constitutes the *pure* dipole. Hyper-plane $H(\mathbf{w}, \theta)$ splits the dipole $(\mathbf{x}^i, \mathbf{x}^j)$ if and only if:

$$(\langle \mathbf{w}, \mathbf{x}^i \rangle - \theta) \cdot (\langle \mathbf{w}, \mathbf{x}^j \rangle - \theta) < 0 \tag{1}$$

It means that the input vectors $\mathbf{x}^i$ and $\mathbf{x}^j$ are situated on the opposite sides of the dividing hyper-plane.

Like most of the existing tree induction algorithm the presented system proceeds in a greedy, top-down fashion. At each non-terminal node, starting from the root, the best split is learned by using the evolutionary algorithm described below. The main components (e.g. fitness function, specific genetic operator) of the algorithm are based on the concept of dipoles. The learned hyper-plane divides the training subset into two subsets generating child nodes. The process is

repeated at each newly created child node until the stop condition (the number of objects is lower than the fixed value or all objects are from the same class) is satisfied and the node is declared as terminal one.

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. Post-pruning procedure allows to avoid the problem and to improve the generalization power. In the system, a modified Quinlan's pessimistic pruning is utilized.

2.1 Evolutionary Algorithm for Hyper-plane Searching

Representation, initialization and termination condition. The searched hyper-plane $H(\mathbf{w}, \theta)$ is represented in chromosomes as $N+1$ real numbers corresponding to weight vector $\mathbf{w}$ and threshold θ. Initial population is created based on simple algorithm: for each chromosome one mixed dipole $(\mathbf{x}^i, \mathbf{x}^j)$ is randomly drawn and $H_{ij}(\mathbf{w}, \theta)$ is placed to split it:

$$\mathbf{w} = \mathbf{x}^i - \mathbf{x}^j \qquad \text{and} \qquad \theta = \frac{1}{2}[\langle \mathbf{w}, \mathbf{x}^i \rangle + \langle \mathbf{w}, \mathbf{x}^j \rangle]. \tag{2}$$

$H_{ij}(\mathbf{w}, \theta)$ is perpendicular to the segment connecting the opposite ends of the dipole (placed in halfway).

The algorithm terminates if the fitness of the best individual in population does not improve during the fixed number of generations (default value is equal to 200) or the maximum number of generations is reached (default value: 1000).

Fitness function. From the dipolar point of view the optimal hyper-plane should divide as many as possible mixed dipoles and try not to split pure dipoles. This leads to the following criterion function:

$$F(\mathbf{w}, \theta) = f_{mixed} + \alpha \cdot (1 - f_{pure}), \tag{3}$$

where f_x is the fraction of divided dipoles (concerning only mixed or pure dipoles) and α is user supplied coefficient, which allows to control the importance of pure dipoles. It should be noted that α should be less than 1.0 (default value: 0.01). Otherwise, this can lead to such a situation, where none of dipoles is divided, which is obviously useless.

It is commonly accepted that the dividing hyper-plane should be as simple as possible. Embedding the feature selection mechanism into fitness function enables to eliminate noisy (or/and irrelevant) features. It results in increased understandability of the test and what is even more important, it can improve the overall performance of the classifier. The final form of the fitness function is as follows:

$$Fitness(\mathbf{w}, \theta) = F(\mathbf{w}, \theta) \cdot [(1 - \beta) + \beta \frac{n}{N}], \tag{4}$$

where n is the number of non-zero weights in hyper-plane (i.e. number of features used in the test) and β ($\beta \in 0..1$) is the user supplied parameter destined for controlling the complexity of the test (default value: 0.2).

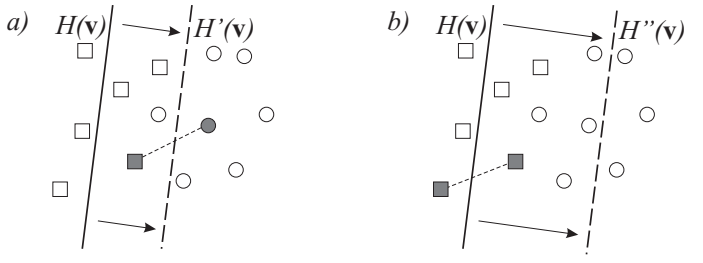

Fig. 1. Examples of dipolar operator in action: (a) splitting the mixed dipole, and (b) avoiding to split the pure dipole.

The problem which is directly connected with feature selection is "underfitting" the training data [7], which often occurs near the leaves of the tree. The number of objects used to search for a split has to be significantly greater than the number of features used. In the presented system, the maximal number of features in test is restricted based on the number of available training objects.

Genetic operators. The standard real number representation of the hyperplane in the chromosome is not especially suited for the feature selection. There is no special mechanism for representing the elimination of features (zero weight corresponding to an excluded feature is not very likely). It means that at least one genetic operator (*i.e.* mutation in the proposed solution) has to be modified to significantly increase the probability of the feature drop. The modified mutation operator has a strong preference in assigning zero value to weights.

Apart from the slightly modified mutation and the standard two-point crossover, a new specialized dipolar operator is proposed. The dipolar operator works as follows: first, the dipole type is drawn (mixed or pure). If the mixed type is chosen, one dipole is drawn from the set of not divided mixed dipoles and the hyper-plane is shifted to separate the pair of feature vectors. The new position is obtained by modifying only one randomly chosen feature. In case of the pure type, one dipole is drawn from the set of divided pure dipoles. The hyper-plane is shifted to avoid of separation of objects from the same class and similarly like in the mixed case only one randomly chosen weight is modified. In figure 1, two examples of dipolar operator behaviour are presented.

As a selection mechanism the proportional selection with linear scaling is applied. Additionally, the chromosome with the highest value of the fitness function in each iteration is copied to the next population (elitist strategy).

3 Experimental Results

Two types of experiments were performed. In the first series 4 datasets with analytically defined classes were analyzed. For these datasets the optimal solution

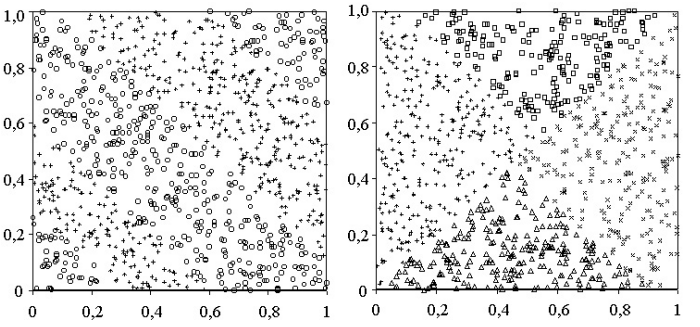

Fig. 2. Synthetic datasets: "zebra" and rotated "chessboard".

is known. Analogous tests are described in [10], but original datasets are not publicly available, hence direct comparison is not possible. In the second series, a few well-known real-life datasets taken from UCI repository [1] were analyzed.

All results were obtained by using 5-times repeated 10-fold stratified cross-validation. Number of leaves (terminal nodes) is given as a complexity measure of the obtained classifier. The induction times were measured on PC (PII 350MHz).

All synthetical datasets have 2000 of feature vectors belonging to 2 or 4 classes. First two datasets (LS2 and LS10, where the final number denotes dimension) represent simple 2-class Linearly Separable problems. The optimal hyperplanes are defined as follows: $x_1 - x_2 = 0$ and $x_1 + x_2 + x_3 + x_4 + x_5 - x_6 - x_7 - x_8 - x_9 - x_{10} = 0$. Two remaining synthetic datasets are depicted in figure 2. The results of the experiments (the quality of classification, the complexity of the tree and time needed to induce the classifier) are presented in table 1.

Table 1. The results of the experiments

Dataset	Quality[%]	Complexity[#leaves]	Time[s]
LS2	99.9	2	10
LS10	98.3	8	20
Zebra	99.4	6	23
Chessboard	99.6	6	17
Breast	95.2	11	28
Heart	78.2	20	29
Segmentation	95.1	51	453
Waveform	77.5	39	98

For all synthetical domains the proposed system based on EA performed well, especially in the classification accuracy. LS10 dataset was the most difficult artificial domain in the study [10], where OC1 system obtained 97.2% with 14 leaves. The extended OC1 (GA) described in [5] gave more compact tree (9 leaves), but

the classification quality was decreased (95.4%). The proposed system obtained slightly better results for this dataset: 98.3% with 8 nodes.

Concerning real datasets the classification quality of the system is equivalent to obtained by OC1 and its variants, however it seems that number of leaves is higher in the proposed system. This could be associated with different pruning algorithms utilized by systems.

4 Conclusion

In the paper, the search for optimal hyper-plane in each non-terminal node of the oblique decision tree is performed by a specialized evolutionary algorithm with embedded feature selection. Proposed fitness function and key genetic operator are based on dipole concept. Results of the experimental validation are promising and furthermore there are still many places for possible improvements (e.g. local fitting the hyper-plane position to enlarge the margin between the feature vectors from different classes).

Acknowledgments. The author is grateful to Prof. Leon Bobrowski for his support and useful comments. This work was supported by the grant W/WI/1/02 from Białystok Technical University.

References

1. Blake, C., Keogh, E., Merz, C.: *UCI repository of machine learning databases*,[http://www.ics.uci.edu/˜mlearn/MLRepository.html]. Irvine, CA: University of California, Dept. of Computer Science (1998).
2. Bobrowski, L., Krętowski, M.: Induction of multivariate decision trees by using dipolar criteria. In *Principles of Data Mining and Knowledge Discovery, PKDD'00.* Springer LNCS 1910, (2000)
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. Breiman, L., Friedman, J., Olshen, R., Stone C.: *Classification and Regression Trees.* Wadsworth Int. Group (1984)
5. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. *IEEE Transcations on Evolutionary Computation* **7**(1) (2003) 54–68.
6. 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.
7. Duda, O., Heart, P., Stork, D.: *Pattern Classification.* 2^{nd} edn. J. Wiley (2001).
8. Gama, J., Brazdil, P.: Linear tree. *Inteligent Data Analysis* **3**(1) (1999) 1–22.
9. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs.* 3^{rd} edn. Springer (1996).
10. Murthy, S., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* 2 (1994) 1–33.
11. Murthy, S.: Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery* 2 (1998) 345–389.
12. Quinlan, J.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann (1993).