Differential Evolution and Perceptron Decision Trees for Classification Tasks

Lopes, R. A.[1], Freitas, A. R. R.[1], Pedrosa Silva, R. C.[1], and Guimarães, F. G.[2]

[1] Graduate Program in Electrical Engineering, Federal University of Minas Gerais
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil
{rodolfo.ufop,alandefreitas,rcpsilva}@gmail.com
[2] Department of Electrical Engineering, Federal University of Minas Gerais
Belo Horizonte, MG 31270-901, Brazil
fredericoguimaraes@ufmg.br

Abstract. Due to its predictive capacity and applicability in different fields, classification has been one of the most important tasks in data mining. In this task, the Perceptron Decision Trees (PDT) have been used with good results. Thus, this paper presents a Differential Evolution algorithm that evolves PDTs. Furthermore, we also present the concept of legitimacy which is used to reduce the costs of solution evaluation, a time consuming part of the algorithm. The experiments comparing our method with other seven well known classifiers, show that the proposed approach is competitive and has potential to build very accurate models. The best solutions found by it were the best ones in the majority of the tested databases.

Keywords: Evolutionary Computation, Data Classification, Differential Evolution, Perceptron Decision Trees

1 Introduction

Due to its predictive capacity and applicability in different fields, classification has been one of the most important tasks in data mining. It consists, basically, in finding a model that describes and distinguishes data classes. The classification model is induced based on a training data set, then it is used to examine features of a new instance and assign to it a predefined class.

In this context, Perceptron Decision Trees (PDT) have been used in many real-world classification tasks with good results [1–3]. They are decision trees whose nodes test a linear combination of the attributes partitioning the input space by hyperplanes in general positions [4].

In this paper we show an approach to generate PDTs, based on a Differential Evolution (DE) algorithm. DE is an important Evolutionary Algorithm which works with a population of solutions and iteratively searches high quality solutions (Section 3). Thus, DE will work with a population of PDTs iteratively minimizing their classification errors.

By describing the methodology of this work, this paper introduces the following contributions:

- An algorithm based on Differential Evolution for evolving Perceptron Decision Trees (Section 4)
- An approach for representation and manipulation of PDT in the context of DE solutions (Section 4.1-4.2)
- A strategy for replacing solutions (Section 4.3), in which the worst classifiers give place to new ones
- A method for controlling the legitimacy of the evaluation (Section 4.4), which makes evaluation more legitimate when it seems to be necessary

Comparative tests with well known classifiers are presented in Section 5. Those results show that even though a more extensive study is needed, the proposed algorithm is very promising for building accurate classification models (Section 6).

2 The Classification Task

Classification consists in predicting the value of a user-specified goal attribute (the class) based on the values of other attributes, called predicting attributes. In a movie classification example, the goal attribute might be the *Movie Genre*, taking on the values (classes) "action" or "drama", while the predicting attributes might be the title, cast and soundtrack.

In the classification task we want to build a model which maps predicting attribute values to classes accurately. For this purpose, the data being mined is usually divided into two mutually exclusive data sets, the training set and the test set. In a first step, called training, the classification algorithm builds the model accessing the values of both the predicting attributes and classes of each example in the training set.

Once the training process is finished and the algorithm has built the model, its predictive performance is evaluated on the test set, which was not seen during the training phase. Thus, if the performance of the built model is good enough, it will be used to classify new instances for which the classification is not known.

3 Evolving Perceptron Decision Trees with Differential Evolution

Differential Evolution (DE) [5] is an important Evolutionary Algorithm (EA) developed to improve the quality of a solution over many iterations. In DE new candidate solutions are created by combining a parent individual with several other individuals of the same population. Then, this candidate solution replaces the parent if it has a better fitness value. EA have been successfully used to evolve classifiers, such as Oblique Decision Trees (ODT) [6].

Through the basic iterative process of DE, as the one described below, it is expected that a satisfactory solution will be found:

- Generate many solutions $\mathbf{x}$ with random positions in the search space.

- Initialize the crossover probability CR and differential weight F
- Until a halting criterion is not met, the following steps are repeated:
  - For each solution $\mathbf{x}$:
    - Choose other three solutions $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ from the population that are different from $\mathbf{x}$.
    - In order to generate a new solution $\mathbf{y}$, for each position i of the solution:
      - Choose a random index j between 1 and the number of variables.
      - Generate a real valued number r between 0 and 1 with uniform distribution
      - If $(r < CR \vee j == i)$, $\mathbf{y}_i = \mathbf{a}_i + F(\mathbf{b}_i - \mathbf{c}_i)$, else $\mathbf{y}_i = \mathbf{x}_i$
    - If $\mathbf{y}$ is better than $\mathbf{x}$, $\mathbf{x}$ is replaced by $\mathbf{y}$
- Return the best solution found

The values F, CR and the population size must be chosen by the user. Given a suitable formulation of a data classification problem, DE can then evolve classification rules for the test database.

3.1 Modeling a DE solution

As mentioned in Section 2, the classification task involves a model generation. A common approach for classifiers is building binary classification trees, in which, each internal node indicates a test over one attribute, each branch indicates a possible result of the test and each leaf node corresponds to a class. In this way, the tests associated with each node are equivalent to axis-parallel hyperplanes in the input space [4]. For an example on the evolution of these ODTs, see [6].

In this work, we use a different approach for defining each tree node. In this approach the internal nodes test a linear combination of the attributes. With the equation $\mathbf{wx} + \theta = z$, where $\mathbf{w}$ as weight scalars, $\mathbf{x}$ are the data attributes and θ is a constant, we can divide the search space by hyperplanes in general positions, which are not necessarily axis-parallel. With this approach, we define a PDT.

Thus, the search space can be divided considering all data attributes at each step, differently from the conventional binary tree. In case we want to consider only the attribute i with the PDT, the vector $\mathbf{w}$ must be constituted of zeros, apart from the position $\mathbf{w}_i$, where its value will be 1.

Many authors have employed similar concepts of aggregating many linear classifiers in a tree under different names [7–9], including PDT [4].

4 Methodology

In this section we describe how DE was used for evolving PDT and the whole procedure used in the comparisons with other well known classifiers to validate our approach.

4.1 Representation of a Classifier

In our method, each PDT is a complete binary tree with linear classifiers in the internal nodes. Each of those nodes is defined by a vector of weights $\mathbf{w}$ of size n and a constant θ, being n the number of attributes of the classification problem. Hence, the number of classifiers in a PDT is $2^{(h-1)}-1$, being h a control variable defined as the depth of the PDT.

Besides the classifiers, each PDT contains $2^{(h-1)}$ leaf nodes, which contain a possible classification for a sample. Those nodes are defined in the discrete space. Thus, a complete PDT can be defined by the matrices of size $n \times 2^{h-1}-2$ for the weights, $1 \times 2^{h-1}-1$ for the constants and $1 \times 2^{h-1}$ for the leaf nodes. This is the size of each solution employed by the DE.

While the individual size is $O(2^{h-1}-1)$, the evaluation cost is still $O(h)$ because only one possible path is searched through the PDT. Each of the $2^{h-1}-1$ columns of the classifier matrices defines a classifier. After using the linear classifier at position i of n, the classifier at position $2i$ is used if the output of the classifier is < 0 or the classifier $2i+1$ is used otherwise.

As the total error will be the standard of comparison between the PDTs in the population, the objective function value of each of the solutions is defined as the number of misclassifications, including false positives and false negatives. This objective function, naturally, must be minimized by the DE.

4.2 Operators

Simple operators, as shown in Section 3 were employed for the generation of new solutions. Values of F and CR are defined as random values between 0.4 and 1 and between 0.9 and 1, respectively [5]. Those values are altered at each iteration of a generation.

As for the leaf nodes, a discrete approach must be used for the crossover of solutions [10]. The approach used is inspired in the idea that DE uses a vector of differences to alter the solution, however now this vector represents swap movements between two possible positions.

In doing so, the movement described as $\mathbf{b}_i - \mathbf{c}_i$ is only performed if $\mathbf{a}_i = \mathbf{b}_i$. If $\mathbf{a}_i = \mathbf{c}_i$, the movement $\mathbf{c}_i - \mathbf{b}_i$ is applied. In a third possible case, where $\mathbf{a}_i \neq \mathbf{b}_i$ and $\mathbf{a}_i \neq \mathbf{c}_i$, no movement is performed.

In this case, the value F is used to define if the operation should happen. The operation occurs if a randomly generated number is less than F.

4.3 Replacement of Individuals

As the replacement of individuals is made one by one, it may happen that some solutions are stagnant in bad points of the search space. In order to avoid this problem, a new individual survival operator was developed to keep always new candidates in the population.

At each generation, new random individuals are generated to replace the $x\%$ worst individuals. Moreover, the individuals with an accuracy rate less than $1/n_{classes}$ are also replaced.

4.4 Legitimacy

In the initial generations, where all the individuals are still very random, not many comparisons are needed to perceive that some are better than the others. In most cases, with the employment of classifiers in less than 10 samples it is possible to define clearly which PDTs are the best ones. The same does not apply after many generations, when most solutions classify the samples with a low error rate.

Having this in mind, and the high cost of evaluating solutions, we propose an approach to reduce the time spent to evaluate solutions in the first generations.

The key idea is that initial solutions do not need to be evaluated with the same legitimacy as the solutions in the last generations, where a more refined analysis is necessary to distinguish good solutions. The value l defines the legitimacy utilized in the evaluation of individuals. At a generation with legitimacy l, only l samples are tested in the evaluation of each solution.

The initial value of this parameter was defined as twice the number of attributes of each samples. After each generation, the value l is updated accordingly to the equation $l = \lceil \min(N, l * \alpha) \rceil$ where N is the number of available samples for test and α is the rate of increase of the legitimacy for each generation.

The parameter α was defined as $1+\delta/\sigma$, where δ is the speed of legimitization, defined as 10^{-6}, and σ is the standard deviation of the objective function value of the solutions. Thus, when the solutions are still very diverse, the parameter α is smaller.

It is important to notice that, whenever the objective function value is calculated, the legitimacy value used to obtain the objective function value is also stored. Thus, if the individual is not replaced by a new one after t generations and the legitimacy value has already increased, the individual is reevaluated with the new legitimacy value l. The value of t was defined as 10.

5 Results

In this section we present the computational results. Our approach was compared with other well known classifiers, such as: J48 [11], BFTree [12], RandomTree [13], IB1 [14], RBF [15], MLPs [16] and Naive Bayes [17]. The results of these methods were obtained from the use of the Weka Framework[3] [13].

For the comparisons, eight databases[4] were taken from the UCI repository [18]. They were randomly divided into two sets, 85% for training and 15% for tests. For fair comparison between the methods, the same sets were used by all of them.

The control parameters of the proposed approach were defined in a series of preliminary tests and they are: Population size = 50 individuals; Maximum number of generations = 2000; PDT height (h) = 5 and Replacement rate =

[3] Available in `http://www.cs.waikato.ac.nz/ml/weka/`

[4] Breast Tissue, Vertebral Column, Ecoli, Glass Identification, Hill-Valey, Iris, Libras Movement and Sonar vs. Rocks

0.1. An increase in the value of the PDT height would lead to a better capacity of representation as well as an increase in computational cost per generation.

Due to the stochastic nature of our method the results related to it present a synthesis of 30 independent runs. On Table 1 a Kruskal-Wallis test ($\alpha = 0.05$) was used to compare the averaged results obtained by our method with other algorithms. We can see among the tree-based classifiers that the proposed approach was significantly better than the J48 and the RandomTree for most databases. It has also had competitive results in relation to the BFTree and IB1. Among the other types of classifiers, the proposed approach was significantly better than Naive Bayes. The RBF and the MPL performed significantly better than the proposed approach.

Table 2 depicts the results obtained by each one of the methods. For our approach, that will be called PDT for simplicity, we present the worst (Min), mean, and best (Max) results, as well as the standard deviation over 30 runs. The best results of our approach were better in 5 of 8 databases. The other methods had at most the best results in only two databases each. This reveals the potential of the representation model. However, in order to improve the average results, some study and effort must be employed in reducing the variance of the solutions derived by it.

Table 3 shows the error obtained by our method in the training phase. We can observe that in the databases in which our results were poor, the method had difficulties in minimizing the errors also during the training phase, terminating the process with high errors.

6 Conclusion

This work presents an approach to build classifier models based on Perceptron Decisions Trees. For the model training we use an adaptation of the Differential Evolution algorithm which works with continuous and discrete variables. Once the evaluation of solutions has a high cost in the presented procedure, this work also presents the concept of legitimacy, which allows a decrease in the solution evaluation time in the first generations.

The results indicate the potential to build very accurate models as the best solutions found were the best ones in the majority of the tested databases. Despite the success of the best solutions, the statistical test revealed that the average performance of the method is only competitive when compared with other important tree-based classifiers. Besides, the errors obtained in the training phase showed that the method had difficulties in minimizing the error in some databases.

In order to improve the average performance, the use of a validation set can be extracted from the training set and work as test set inside the training. The validation set can broaden the generalization capacity of the classifiers, improving its accuracy and it can be used to give support to both the stopping criteria and the replacement mechanism.

Table 1. Comparison between PDT and each other techniques

Database	PDT x J48	PDT x BFTree	PDT x RandomTree	PDT X IB1	PDT X RBF	PDT X MLP	PDT X NaiveBayes
Breast T.	worse	worse	worse	worse	worse	worse	worse
V. Column	equal	equal	better	better	worse	worse	better
Ecoli	better	equal	better	better	worse	worse	worse
Glass I.	worse	worse	worse	worse	equal	worse	better
Hill-V.	better	better	better	better	better	better	better
Iris	better	better	better	better	worse	better	better
Libras M.	worse	worse	worse	worse	worse	worse	worse
Sonar vs. R.	better	better	better	worse	better	worse	better

Table 2. PDT Results - Accuracy Rate - Test set

Database	PDT - Min.	PDT - Mean(Std.)	PDT - Max.	J48	BFTree	RandTree	IB1	RBF	MLP	NaiveBayes
Breast T.	0.3750	0.5292 (0.0910)	0.6875	0.625	**0.8125**	0.75	0.6875	**0.8125**	0.6875	**0.8125**
V. Column	0.7660	0.8411 (0.0294)	**0.9149**	0.851	0.851	0.8297	0.7659	0.8723	0.8723	0.8297
Ecoli	0.6863	0.7673 (0.0430)	**0.8627**	0.745	0.7647	0.6666	0.745	0.8235	**0.8627**	**0.8627**
Glass I.	0.4242	0.5808 (0.0690)	0.6667	0.6969	0.7575	**0.7878**	0.7272	0.606	0.6969	0.4848
Hill-V.	0.9341	0.9945 (0.0140)	**1.0**	0.5109	0.5769	0.6318	0.6483	0.5054	0.5769	0.5219
Iris	0.8261	0.9435 (0.0305)	**0.9565**	0.8695	0.8695	0.8695	**0.9565**	**0.9595**	**0.9565**	0.913
Libras M.	0.1852	0.3185 (0.0616)	0.4074	0.7222	0.6111	0.6666	**0.7592**	0.6666	0.7407	0.574
Sonar vs. R.	0.6563	0.7729 (0.0614)	**0.9063**	0.5312	0.625	0.7187	0.875	0.75	0.875	0.6562

Table 3. Error Rate - Training

Database	Minimum Error	Mean Error (Std.)	Maximum Error
Breast Tissue	0.2222	0.2952 (0.0473)	0.4111
Vertebral Column	0.1065	0.1343 (0.0130)	0.1559
Ecoli	0.0982	0.1304 (0.0307)	0.2105
Glass Identification	0.2099	0.3098 (0.0472)	0.4199
Hill-Valley	0.0	0.0054 (0.0133)	0.0602
Iris	0.0	2.6247e-004 (0.0014)	0.0079
Libras Movement	0.5327	0.5874 (0.0289)	0.6438
Sonar vs. Rocks	0.0511	0.0977 (0.0219)	0.1591

Acknowledgements

This work has been supported by the Brazilian agencies CAPES, CNPq (305506 / 2010-2), FAPEMIG (CEX APQ-04611-10) and by a Marie Curie International Research Staff Exchange Scheme Fellowship within the 7th European Community Framework Programme.

References

1. Bennett, K.P., Auslender, L., Wu, D., Ave, S.: On support vector decision trees for database marketing. Technical report, Department of Mathematical Sciences Math Report No. 98-100, Rensselaer Polytechnic Institute (1998)
2. Bennett, K.P., Mangasarian, O.L.: Multicategory discrimination via linear programming. Optimization Methods and Software **3** (1992) 27–39
3. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research **2** (1994) 1–32
4. Bennett, K., Cristianini, N., Shawe-Taylor, J., Wu, D.: Enlarging the margins in perceptron decision trees. Machine Learning **41**(3) (2000) 295–313
5. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4) (1997) 341–359
6. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. Evolutionary Computation, IEEE Transactions on **7**(1) (2003) 54–68
7. Bennett, K., Mangasarian, O.: Multicategory discrimination via linear programming. Optimization Methods and Software **3**(1-3) (1994) 27–39
8. Breiman, L.: Classification and regression trees. Chapman & Hall/CRC (1984)
9. Brodley, C., Utgoff, P.: Multivariate decision trees. Machine Learning **19**(1) (1995) 45–77
10. Prado, R.S., Pedrosa Silva, R.C., Guimarães, F.G., Neto, O.M.: Using differential evolution for combinatorial optimization: A general approach. In: Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics (SMC). (2010) 11–18
11. Quinlan, J.R.: C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). 1 edn. Morgan Kaufmann (January 1993)
12. Shi, H.: Best-first decision tree learning. Master's thesis, University of Waikato, Hamilton, NZ (2007)
13. Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I.H., Trigg, L.: Weka. In Maimon, O., Rokach, L., eds.: Data Mining and Knowledge Discovery Handbook. Springer US (2005) 1305–1314
14. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning **6**(1) (1991) 37–66
15. Buhmann, M.D., Buhmann, M.D.: Radial Basis Functions. Cambridge University Press, New York, NY, USA (2003)
16. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. MIT Press, Cambridge, MA, USA (1986) 318–362
17. Zhang, H.: Exploring conditions for the optimality of naïve bayes. International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) **19**(2) (2005) 183–198
18. Frank, A., Asuncion, A.: UCI machine learning repository (2010)