World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:7, 2009

Balancing Neural Trees to Improve Classification Performance

Asha Rani,Christian Micheloni* and Gian Luca Foresti

Abstract—In this paper, a neural tree (NT) classifier having a simple perceptron at each node is considered. A new concept for making a balanced tree is applied in the learning algorithm of the tree. At each node, if the perceptron classification is not accurate and unbalanced, then it is replaced by a new perceptron. This separates the training set in such a way that almost the equal number of patterns fall into each of the classes. Moreover, each perceptron is trained only for the classes which are present at respective node and ignore other classes. Splitting nodes are employed into the neural tree architecture to divide the training set when the current perceptron node repeats the same classification of the parent node. A new error function based on the depth of the tree is introduced to reduce the computational time for the training of a perceptron. Experiments are performed to check the efficiency and encouraging results are obtained in terms of accuracy and computational costs.

Keywords—Neural Tree, Pattern Classification, Perceptron, Splitting Nodes.

I. Introduction

Decision tree and neural networks are two powerful tools for pattern classification. Several researches have been conducted using these two tools in an alternative manner. A performance comparison based study of these two approaches is given by Atlas et al. [1] for the purpose of classification of various real life applications such as load forecasting, power security and vowel recognition. They have found that both tools have their own advantages as well as drawbacks. Most of the existing top-down decision tree design methods make use of single feature splits at successive stages of the tree design. While computationally attractive, single feature splits generally lead to large trees and inferior performance. On the other hand, one cannot decide an ideal architecture of a neural network (number of hidden layers and number of nodes in each hidden layer) for a particular set of training data. In this way, an hybridization of these two methodologies called neural tree [2], [16], [14] came in existence to solve tough problems.

Some approaches proposed in this area were motivated by the lack of a reliable procedure for determining the appropriate size of feedforward neural networks in practical applications. These approaches used decision trees to help to determine the topology of neural networks in order to facilitate learning and/or improve generalization by controlling the number of nodes and connections [2], [14], [7], [6]. Some approaches were motivated by the lack of a powerful procedure for determining the appropriate splits or tests of decision trees. These approaches used neural networks to refine the splits or even directly embedded neural networks functioning as splits in decision trees in order to improve generalization [17], [18], [15], [12], [16].

Apart from these approaches, several other efforts have been made to hybrid decision tree and neural network into one structure. A new concept called split node has been introduced in simple perceptron based NT [5] for splitting the training set in two parts when the current perceptron node repeats the same classification of the parent node. This strategy has been provided a guaranteed convergence in any case of the tree building process and to reduce misclassification. In [19] Zhou and Chen have presented a hybrid decision tree (HDT) for the simulation of human reasoning by using symbolic learning to do qualitative analysis and by neural processing to do subsequent quantitative analysis. In [4] Foresti and Micheloni have proposed a generalized neural tree (GNT) model by normalizing the activation values of each node so that these can be interpreted as a probability. The main novelty of the GNT consists in the definition of a new training rule that performs an overall optimization of the tree. Each time the tree is increased by a new level, the whole tree is re-evaluated. An adaptive high-order neural tree (AHNT) [3] has been proposed by composing high order perceptron (HOP) instead of simple perceptron in neural tree model. First-order nodes divide the input space with hyperplanes, while HOPs divide the input space arbitrarily, but at the expense of increased complexity.

Recently, a new neural tree classifier called NNTree [9] has been proposed for designing tree-structured pattern classifier. Instead of using information gain ratio as splitting criterion, a new criterion has been introduced for NNTree design. It has been shown that the new criterion captures well the intuitive goal of reducing the rate of misclassification. The performance of NNTree has been evaluated through its applications in letter recognition, satellite image classification, splice-junction and protein coding region identification. Experimental comparison has been made with other related algorithms in terms of better or comparable classification accuracy with significantly smaller trees and fast classification times.

In this paper, we propose a new NT architecture attempting to reduce the size of the tree and improving the classification. Behaviour of a perceptron is noticed at each node and if the classification done by this is not good enough and unbalanced then it is replaced by such a perceptron that separates the training set in such a way that almost equal number of patterns fall into each of the classes. Moreover, a perceptron learns only for the classes which are present at respective node and ignore other classes. Splitting nodes are employed into the neural tree architecture only to divide the training set when the current perceptron node repeats the same classification of the parent node. A new error function based on depth of

Department of Mathematics and Computer Science University of Udine, Via Della Scienze 206, Udine-33100, Italy.
*Corresponding Author email:*christian.micheloni@dimi.uniud.it*

tree is introduced to reduce the computational time in learning of a perceptron. Experiments are performed for classification of synthetic as well as real data sets. Results are compared with other classical and well established methods in terms of classification accuracy.

The paper is structured as follows: Section II contains a description of a perceptron based neural tree. The proposed learning and classification algorithms are given in section III. Experimental results and comparison study are presented in IV. Finally, section V contains the concluding remarks.

II. Standard Neural Tree

A neural tree (NT) is a decision tree with each intermediate/non-terminal node being a simple perceptron. It is constructed by partitioning the training set consisting of feature vectors and their corresponding class labels for generating the tree in a recursive manner. This procedure involves three steps: splitting nodes, determining which nodes are terminal nodes, and assigning class labels to terminal nodes. In NT, a leaf/terminal node covers the set/subset of elements of only one class. By contrast, an intermediate node covers the set/subset of elements belonging to more than one class. Thus, NT are class discriminators which recursively partition the training set to get nodes belonging to a single class. When a node contains all patterns of the same class, it is labelled as a leaf node.

At each node a perceptron takes N patterns as input and generates M output called activation values corresponding to each class presented at that particular node. Each perceptron is characterized by a sigmoidal activation function $\sigma_i(net) = (1+e^{-net})^{-1}$. Thus the activation value o_i^k of the ith neuron generated by the kth pattern is given by

$$o_i^k = \frac{1}{(1+e^{-\sum_{j=1}^{N}(W_{ij}x_j^k)})}$$

where w_{ij} are the ij^{th} element of the weight matrix W. The NT structure contains several levels $l \in [1, L]$ called depth of the tree.

III. Learning and Classification Algorithm

A. Learning Algorithm

The aim of the training procedure is to build the NT. In other words, training a neural tree takes a training set S as input and returns the neural tree with optimal weights on each node on the output. The proposed learning algorithm is summarized by the following steps:

1) Create a single perceptron without hidden layers and initialize its weight matrix in such a way that the hyperplane generated by these weights passes through the centre of mass of the training set. Set $S = S_{root}$.
2) Start training the perceptron by updating the weight matrix in order to minimize the classification error by optimizing a cost function. Note that the perceptron is forced to consider only the classes which are actually presented at that node instead of all.
3) If the uniformity factor (β_i) [8], which indicates the strength of the class i at a node after classification, is more than a defined threshold ϵ then this node is made a leaf node labelled with the dominating class. If $\beta_i > \epsilon$ for all i, the current perceptron ends with M leaf nodes, one for each class. We will explain β in more details later. Go to step 8.
4) Otherwise, if training set is divided into $m \leq M$ groups,$(S_1, \ldots S_m)$, a new level of m child nodes are created and each subset S_i is assigned to the corresponding child node for the training process. A pattern is assigned to a class based on its highest activation value.
5) For each child node, repeat step 1 on the corresponding local training set S_i.
6) If the classification done by a perceptron at any node is not accurate and unbalanced, its weights are replaced by the initial weights which divide the training set with the hyperplane passing through the centre of mass. Such a solution allows a balanced distribution of patterns among all classes.
7) If the S can not be further divided in any group, that is the perceptron repeats the classification done by the parent node then S is divided into two groups S_l and S_r using the splitting rule [5] producing two child nodes. Go to step 2.
8) If all current nodes are leafs, the algorithm ends. Otherwise, go to step 2 to train the remaining perceptrons.

Let $\overline{\delta}$ be the error defined as the mean error computed on all output neurons and patterns,

$$\overline{\delta} = \frac{1}{KM}\sum_{k=1}^{K}\sum_{i=1}^{M}\delta_i^k$$

where K is the total number of patterns at current node and the error δ is the difference of output o_i^k and the target T_i^k computed as

$$\delta_i^k = t_i^k - o_i^k, i = 1, ..., M$$

where

$$t_i^k = \begin{cases} 1, & i = i^k \\ 0, & otherwise \end{cases}$$

is the target vector. Each perceptron is trained with the patterns of the training set until either the error $\overline{\delta}$ becomes less than $\frac{1}{m^l}$ where m is the number of classes present at the current node and l is the current depth of the tree or there is no more reduction in the error for a given number $Wait$ of epochs .

Three different types of splitting are used in the proposed algorithm. The first is the splitting done by a trained perceptron, the second is the splitting done by an untrained perceptron and the third is the splitting done by using a split node.

Before explaining the adopted strategy, to make the process clear a brief notation needs to be introduced. The local error in total classification as well as for each class is calculated at the respective node. Local error is the correctness measure of the classification and it is defined as

$$E_t = 1 - \frac{K_c}{K_t}$$

where K_c is the number of correctly classified patterns and K_t is the total number of patterns present at the current node.

The respective local error for each class is defined as

$$E_i = 1 - \frac{K_{c_i}}{K_{t_i}}$$

where K_{c_i} is the number of correctly classified patterns of class i, and K_{t_i} is the total number of patterns classified as class i.

The studied strategy bases its decision about which splitting criteria must be adopted on the basis of the classification errors. In particular,it has been noticed that if the total error is relevant and at the same time the difference between the lowest and highest errors among the classes is considerable, then the degree of correctness in classification is low and the resulting tree will be unbalanced. Such a situation does not allow to reach a good trade-off between error and depth of the tree. Thus, such a perceptron is not acceptable. To avoid this situation, the proposed solution replaces such a perceptron with the initial untrained perceptron which ensure the uniform distribution of patterns among all the classes present at that node. The resulting NT is therefore more balances. Summarizing, the criterion to judge whether the perceptron classification is accurate or not is the following

$$E_t > \frac{E_{init}}{M} \qquad \text{and} \qquad (E_{max} - E_{min}) > E_t$$

where $E_{max} = \max_i \{E_i\}$, $E_{min} = \min_i \{E_i\}$ and E_{init} is the initial error when the perceptron is not trained. For example, in Figure 1 the training results obtained with a standard NT and the proposed NT are shown. At root node the perceptron is able to separate only a few number of patterns which yields to a high total error $E_t = 0.3593$ and relevant difference in the errors among the two classes, i.e $E_1 = 0$ and $E_2 = 0.4181$. Using the standard NT training procedure, thus keeping such a perceptron, yields to get a deeper NT. On the other hand, adopting the proposed strategy, the root perceptron is replaced with a perceptron whose weights define an hyperplanes passing through the centre of mass of the training set. Such a node divides better the training set allowing to define a more balanced NT thus a NT with a lower depth.

In the case a perceptron is not able to separate the training set in more than one class the third type of splitting is adopted. For such a purpose a splitting rule [5] is considered to divide the data in two groups with almost equal cardinality.

Now let us to better explain the third step of the proposed algorithm. A node is considered as a leaf node if all the patterns falling in its local training set belongs to the same class. To avoid over-fitting, a pre-pruning strategy is applied [9]. Let C_{ij} represents the number of elements of class j covered by i-th child node, where $i = 1, 2, ..., m$ and $j = 1, 2, ..., m$; and β_i represents the uniformity of the distribution of class elements in $i-th$ child node [9]. The value of β_i at ith node is given by

$$\beta_i = \frac{A}{B} \tag{1}$$

where $A = \max_j \{C_{ij}\}$ and $B = \sum_{j=1}^{m} C_{ij}$.

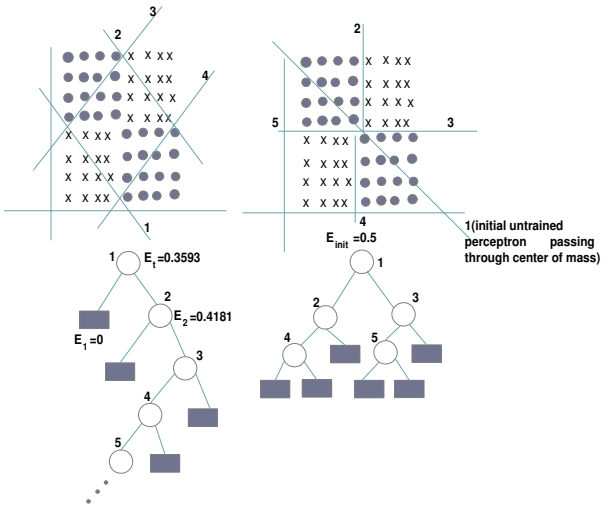

Fig. 1. Tree formation with 2D dataset using(a) Standard NT, (b) Proposed NT

To decide whether to keep training the NT by adding a new child perceptron or introducing a leaf node, the β_i value is measured and compared with a threshold $\epsilon \in (.97, 1)$.

1) If $\beta_i < \epsilon$, then current node needs to be processed further by a perceptron.
2) If $\beta_i > \epsilon$ then the $i-th$ node is made a leaf node of class j for which C_{ij} is maximum. The further class elements falling into the current node are classified to the most probable class of current node that is the class that has maximum number of training patterns in the current node.

B. Classification Algorithm

For the classification task, unknown patterns are presented to the root node. The class is obtained by traversing the tree top to down. Starting from the root, the activation values of the current node gives the next node to consider, until reaching a leaf node, which gives the class of the input pattern. Each node applies the winner-takes-all rule so that

$$\mathbf{x} \in \text{class} \;\; i \Leftrightarrow \sigma(w_j.x) \leq \sigma(w_i.x) \text{ for all } j \neq i \tag{2}$$

where σ is the sigmoidal activation function,w_i is the vector of weights of the connections from inputs to $i-th$ output, and $\mathbf{x}$ is the input pattern. This rule determines the classes associated with the leaf nodes as well as the paths of the internal nodes. Since during the training process some nodes have been trained on a subset of all classes, during classifications the missing classes are still not considered on such a nodes by setting their activation values to zero.

IV. Experimental Results

Several experiments have been performed to evaluate the performance of the proposed neural tree algorithm. Experimental results have been obtained on synthetic as well as real data. The results have been compared with several other

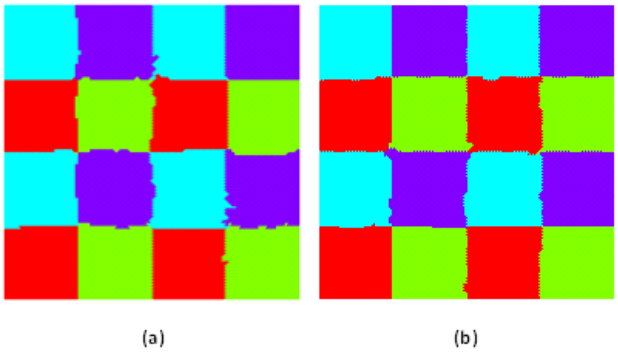

Fig. 2. Classification of the four-class chessboard dataset using (a) Standard NT [5], (b) the proposed NT

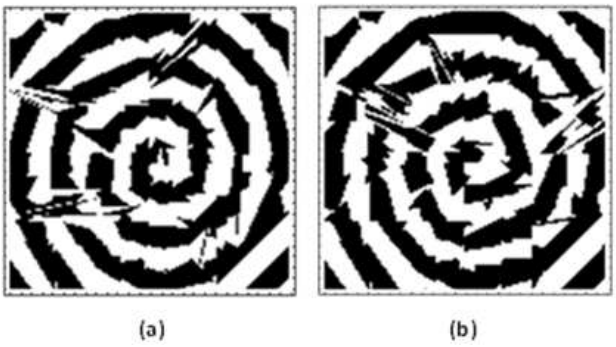

Fig. 3. Classification of the two-class spiral dataset using(a) Standard NT [5], (b) the proposed NT

existing methods. The performance of the proposed neural tree algorithm is compared with standard NT model [5] for synthetic data set. The comparison has been made with different classification algorithms like C4.5 [13], MLP [8], NNTree [9] and CA [10] for real data.

A. Results for Synthetic data

In the first experiment, a training set composed by 1600 patterns belonging to four classes uniformly distributed on a chess-board has been considered. The results obtained by standard [5] and proposed NT with learning rate $\eta = 0.19$, $Wait = 10^3$ and uniformity parameter $\beta = 0.99$ are shown in Figure 2. The main difference between a standard and the proposed algorithms is regarding the size of the tree: the proposed NT is composed by 42 nodes (including 1 split) with depth L=6, while the standard NT is composed by 66 nodes (including 20 split) with depth L=11. The classification accuracy is 98.80% using the proposed NT and it is 97.61% for the standard NT.

The second experiment has been conducted on a more complex data set often used to test the performance of NT algorithms. This training set is composed by 1997 patterns distributed along two interleaved spirals. The parameters of NT used for this training set were: learning rate $\eta = 0.70$, $Wait = 10^3$ and uniformity parameter $\beta = 0.99$. Figure 3 shows the classification results for spiral data using a standard and the proposed NT respectively. The tree obtained using the proposed algorithm contains 284 nodes (including 20 split) with depth L=14, while the standard NT is composed with 435 nodes (including 89 split) with depth L=15. The classification accuracy is 88.81% using the proposed NT and 87.41% using standard NT. It is clear from the presented results that the concept of making a balanced tree in proposed algorithm not only increases the classification accuracy but also reduce the depth of tree (i.e., computational cost).

TABLE I
CLASSIFICATION ACCURACY OF DIFFERENT ALGORITHMS FOR LETTER DATABASE

Algorithms	Classification accuracy (%)
Proposed NT	83.90
Standard NT	not converged
NNTree	82.9
MLP	67.2
C4.5	86.6

B. Results for Real Data

The performance of the proposed NT is tested on two different real data sets used for letter recognition and satellite image classification. The NT parameters are considered as learning rate $\eta = 0.90$, $Wait = 10^3$ and uniformity parameter $\beta = 0.97$ for letter database and learning rate $\eta = 0.57$, $Wait = 10^3$ and uniformity parameter $\beta = 0.97$ for landsat image database.

1) Letter recognition: The dataset used here is constructed by David J. Slate, Odesta Corporation, Evanston, IL 60201. The objective is to classify each of a large number of black and white rectangular pixel displays as one of the 26 capital letters of the English alphabet. One-shot train and test is used for the classification. The character images produced are based on 20 different fonts and each letter within these fonts is randomly distorted to produce a file of $20,000$ unique images. For each image, 16 numerical attributes are calculated using edge counts and measures of statistical moments which are scaled and discretized into a range of integer values from 0 to 15.

The results obtained for this dataset is given in Table I. The proposed NT performs better than the NNTree [9], MLP [8]. However, the classification accuracy is less when compared to the C4.5 [13] algorithm. The reason behind that is the use of simple perceptron in the proposed NT. On the other hand, there is no need to decide many parameters like number of hidden layers, number of nodes in each hidden node, etc. (in case of multi-layer perceptron) in the proposed NT model.

2) Satellite image classification: The original Landsat data for this database is generated from data purchased from NASA by the Australian Center for Remote Sensing, and used for research at the University of New South Wales. The sample database is generated taking a small section (82 rows and 100 columns) from the original data [11]. The data are divided into train and test set with 4435 examples in training set and

TABLE II
CLASSIFICATION ACCURACY OF DIFFERENT ALGORITHMS FOR SATELLITE IMAGE DATABASE

Algorithms	Classification accuracy (%)
Proposed NT	85.25
Standard NT	not converged
NNTree	87.1
CA	77.5
C4.5	85.2

2000 in testing set. The results on this database are presented in Table II in terms of classification accuracy.

V. CONCLUSIONS

In this paper, a new strategy to build a neural tree classifier based on simple perceptrons has been presented. A new concept of making a balanced tree in each node is implemented to increase the accuracy and reduce the depth of the proposed NT. A modified criterion based on classification error has been used for detecting the irregular behaviour of perceptrons at each node. The final weights of the perceptron which behaves irregularly, are replaced by initial weights in such a way that the hyperplane crosses the centre of mass of the training set. The efficiency of the proposed NT in classification has been shown by experimental results on synthetic as well as on real data sets. It is concluded that the proposed NT is computationally efficient and also able to classify complex data sets with good accuracy.

ACKNOWLEDGEMENTS

This work is partially supported by Ministry of Italian University and Scientific Research (MIUR).

REFERENCES

[1] L. Atlas, R. Cole, Y. Muthusamy, A. Lippman, J. Connor, D. Park, M. El-Sharkawi, and R.J. Marks, A performance comparison of trained multilayer perceptrons and trained classification trees. Proceedings of the IEEE, vol. 78(10), pp. 1614–1619, 1990.
[2] G. Deffuant, Neural units recruitment algorithm for generation of decision trees. Proceedings of the International Joint Conference on Neural Networks, San Diego, CA, vol. 1, pp. 637–642, 1990.
[3] G.L. Foresti and T. Dolso, An Adaptive High-Order Neural Tree for Pattern Recognition, *IEEE Trans. on Systemas, Man, Cybernatics- part B: Cybernatics*, Vol. 34 (2),pp. 988–996, 2004.
[4] G.L. Foresti and C. Micheloni, Generalized Neural Trees for Pattern Classification, *IEEE Trans. on Neural Networks*, Vol. 13(6), pp. 1540–1547, 2002.
[5] G. L. Foresti and G. G. Pieroni, Exploiting neural trees in range image understanding, *Pattern Recognit. Lett.*, vol. 19 (9), pp. 869-878, 1998.
[6] M. Kubat, Decision trees can initialize radial-basis function networks. IEEE Transactions on Neural Networks, vol. 9(5), pp. 813–821, 1998.
[7] T. Li, L. Fang, and A. Jennings,. Structurally adaptive self-organizing neural trees. Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD, vol. 3, pp. 329-334, 1992.
[8] R. Lippmann, An introduction to computing with neural nets, *IEEE Acoust. Speech Signal Process. Mag.*,vol. 4 (2) pp. 4–22, 1987.
[9] P. Maji, Efficient Design of Neural Network Tree using a Single Spilitting Criterion, *Nerocomputing*, vol. 71, pp. 787–800, 2008.
[10] P. Maji, C. Shaw, N. Ganguly, B.K. Sikdar, P.P. Chaudhuri, Theory and application of cellular automata for pattern classification, *Fundamenta Informaticae*, vol. 58(34), pp. 321354, 2003.
[11] D. Michie, D.J. Spiegelhalter and C.C. Taylor, Machine Learning, Neural and Statistical Classification, *Ellis Horwood, Chichesten*, UK, 1994.
[12] S.K. Murthy,S. Kasif, and S. Salzberg, A system for induction of oblique decision trees. Journal of Artificial Intelligence Research,vol. 2, pp. 1–32, 1994.
[13] J.R. Quinlan, C4.5: Programs for Machine Learning, *Morgan Kaufmann, Los Atlos*, CA, 1993.
[14] T.D. Sanger, A tree-structured adaptive network for function approximation in high-dimensional spaces, IEEE Transactions on Neural Networks, vol. 2(2), pp. 285–293, 1991.
[15] A. Sankar, and R.J. Mammone, Optimal pruning of neural tree networks for improved generalization. Proceedings of the International Joint Conference on Neural Networks, Seattle, WA, vol 2, pp 219- 224, 1991.
[16] I.K. Sethi, and J.H. Yoo, Structure-driven induction of decision tree classifiers through neural learning. Pattern Recognition, vol. 30(11), pp. 1893–1904, 1997.
[17] P.E. Utgoff, N.C. Berkman, and J.A. Clouse, Decision tree induction based on efficient tree restructuring. Machine Learning, vol. 29(1), pp. 5–44, 1997.
[18] P.E. Utgoff, and C.E. Brodley, An incremental method for finding multivariate splits for decision trees. Proceedings of the 7th International Conference on Machine Learning, Austin, TX: Morgan Kaufmann, pp. 58–65, 1990.
[19] Z. Zhaou and Z. Chen, Hybrid decision tree, Knowladge based systems, vol. 15(8), pp. 515–528, 2002.