Efficient design of neural network tree using a new splitting criterion

Pradipta Maji*

Machine Intelligence Unit, Indian Statistical Institute, 203, Barrackpore Trunk Road, Kolkata 700 108, West Bengal, India

Abstract

This paper presents the design of a hybrid learning model, termed as neural network tree (NNTree). It incorporates the advantages of both decision tree and neural network. An NNTree is a decision tree, where each non-terminal node contains a neural network. The idea of the proposed method is to use the framework of multilayer perceptron to design tree-structured pattern classifier. At each non-terminal node, the multilayer perceptron partitions the dataset into m subsets, m being the number of classes in the dataset present at that node. The NNTree is designed by splitting the non-terminal nodes of the tree by maximizing classification accuracy of the multilayer perceptron. In effect, it produces a reduced height m-ary tree. The versatility of the proposed scheme is illustrated through its application in diverse fields. The effectiveness of the hybrid algorithm, along with a comparison with other related algorithms, has been demonstrated on a set of benchmark datasets. Simulation results show that the NNTree achieves excellent performance in terms of classification accuracy, size of the tree, and classification time.

Keywords: Data mining; Pattern classification; Neural network; Decision tree; Bioinformatics; Deoxyribonucleic acid; Gene

1. Introduction

The inter-networked society has been experiencing an explosion of data. However, the explosion is paradoxically acting as an impediment in acquiring knowledge. The meaningful interpretation of these large volume of data is increasingly becoming difficult. Consequently, researchers, practitioners, and entrepreneurs from diverse fields are trying to develop sophisticated techniques for knowledge extraction, which leads to the promising field of data mining/knowledge discovery in databases [22,31]. It can be defined as the nontrivial extraction of implicit, previously unknown, and potentially useful information from databases [30].

Data mining is an interdisciplinary research area spanning several disciplines such as database systems [39], machine learning [9,24,27], intelligent information systems, statistics [10,18,37], and expert systems [26]. Different methods like association rules [13], characterization [14,15], classification [17], clustering [5], etc. have been applied for mining different kinds of knowledge.

One of the important problems in the emerging field of data mining is classification [1]. In classification, a training set consisting of feature vectors (objects) and their corresponding class labels is used to develop an accurate description or model for each class using the features present in the data. In effect, it partitions the feature space into regions, each region for each category of input data; and attempts to assign every data point in the feature space to one of the possible classes. The class descriptions are used to classify future test data for which the class labels are unknown [16].

Applications of classification include medical diagnosis, performance prediction, selective marketing, etc. Solutions based on Bayesian classification [5], neural networks [17,20], genetic algorithms [11], decision trees [4,23,32,36], etc. have been proposed. But, the search for new and better solutions continues specifically to classify large volume of dataset generated in the inter-networked society of cyber-age.

In this background, many pattern classifiers have been proposed, integrating the advantages of decision tree and

neural network. One of the early pattern classifiers based on this concept is Entropy Nets due to Sethi [34]. It derives a multilayer feedforward neural network from a decision tree. The knowledge represented by the decision tree is translated into the architecture of a neural network whose connections can be retrained by a back propagation algorithm. On the other hand, the ANN-DT [33] uses neural network to generate outputs for examples interpolated from the training dataset and then extracts a univariate binary decision tree from the network. Another method which also extracts decision tree from neural network is [40]. The design of a tree-structured neural network using genetic programming is proposed in [19]. In [12,38,42–44], designs of NNTrees have been introduced. The NNTree is a decision tree with each non-terminal node being a neural network. In [35], Sethi and Yoo have proposed a decision tree whose hierarchy of splits is determined in a global fashion by a neural learning algorithm. Recently, Zhou and Chen [45] have introduced a hybrid learning approach named HDT that embeds neural network in some leaf nodes of a binary decision tree.

To design an NNTree, the most important and time consuming step is splitting the non-terminal nodes of the tree. There are many criteria for splitting non-terminal nodes. One of the most popular criteria is the information gain ratio which is used in C4.5 [32]. Designs of NNTree have so far mostly concentrated around binary tree with information gain ratio used to partition the available dataset at each non-terminal node [12,38,42–44]. However, this structure generates larger height of the tree. In effect, it increases classification time and error rate in classifying test samples. Also none of the work has so far dealt with the application of NNTree to biological dataset.

In the above background, the paper presents the design and applications of an NNTree. The NNTree proposed here adopts an approach which is completely different from [12,38,40,42–45]. The neural networks used in this design are multilayer perceptrons with m output nodes, m being the number of classes in the given dataset. Unlike [12,38,42–45], the NNTree designed here splits each non-terminal node by maximizing (minimizing) classification accuracy (error) of the multilayer perceptron rather than using information gain ratio. So, the proposed design always generates a reduced height m-ary tree. The back propagation algorithm is used recursively at each non-terminal node to find a multilayer perceptron. The effectiveness of the proposed algorithm, along with a comparison with individual components of the hybrid scheme as well as other related algorithms, has been demonstrated on benchmark datasets.

The structure of the rest of this paper is as follows. Section 2 presents the design of neural network based tree-structured pattern classifier. While Section 3 addresses the problem of letter recognition and satellite image classification, Sections 4 and 5 present the application of NNTree in splice-junction and protein coding region identification problems. In order to validate the design of proposed model, extensive experimental results are also reported in these sections. Concluding remarks are given in Section 6.

2. MLP based tree-structured pattern classifier

A neural network tree (NNTree) is a decision tree with each intermediate/non-terminal node being a multilayer perceptron (MLP). It is constructed by partitioning the training set consisting of feature vectors and their corresponding class labels in such a way as to recursively generate the tree. This procedure involves three steps: splitting nodes, determining which nodes are terminal nodes, and assigning class labels to terminal nodes. In this tree, 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, NNTrees are class discriminators which recursively partition the training set to get nodes belonging to a single class. Fig. 1 shows an MLP based NNTree.

To classify a training set $S = \{S_1, \ldots, S_i, \ldots, S_m\}$ consisting of m classes, an MLP has to be designed with m neurons in output layer. If a pattern belongs to ith class, ith output neuron is selected. That is, the content of the ith neuron is 1. At this moment, content of all other output neurons are 0s. So, the output layer represents m distinct m-dimensional vectors, each representing a unique location/node. Thus, the training set S gets distributed into m locations/nodes using an MLP.

Let, $\acute{S}$ be the set of elements in a node. If $\acute{S}$ belongs to only one class, then label that node as that class. Otherwise,

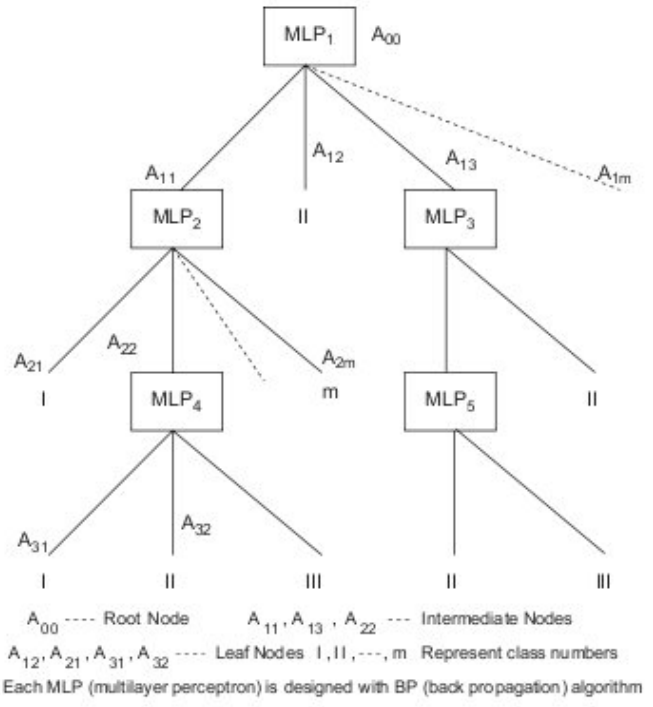

Fig. 1. MLP based tree-structured pattern classifier.

this process is repeated recursively for each node until all the patterns in each node/location belong to only one class. Single or multiple nodes of the tree may form a leaf/terminal node representing a class, or it may be an intermediate/non-terminal node. A leaf node represents a location that contains the set/subset of elements of only one class. By contrast, an intermediate node refers to a location that contains the elements belonging to more than one class. In effect, an intermediate node represents the decision to build a new MLP for the elements of multiple classes covered by the location of the earlier level. The above discussions are formalized in Algorithm 1.

Algorithm 1. NNTree_Building

Input: Training set $S = \{S_1, \ldots, S_i, \ldots, S_m\}$
Output: NNTree (set of MLPs)

Partition(S, m);

Partition$(\tilde{S}, \tilde{m})$

(i) Generate an MLP with $\tilde{m}$ output neurons.
(ii) Distribute the training set $\tilde{S}$ into $\tilde{m}$ locations (nodes).
(iii) Evaluate the distribution of patterns in each node.
(iv) If all the patterns ($\acute{S}$) of a location (node) belong to one particular class, then label the location (leaf node) for that class.
(v) If for the set of patterns ($\acute{S}$) of a location belonging to $\acute{m}$ classes, **Partition** $(\acute{S}, \acute{m})$.
(vi) Stop.

In Fig. 1, the node A_{00} is the root node. So, the MLP_1 corresponding to node A_{00} distributes the training sets $S = \{S_1, \ldots, S_i, \ldots, S_m\}$ into m locations denoted by $A_{11}, A_{12}, \ldots, A_{1m}$. Now, A_{11} is an intermediate node as the elements covered by this location belong to multiple classes (here m) which are distributed again by MLP_2 into m number of locations—$A_{21}, A_{22}, \ldots, A_{2m}$. A_{13} is also an intermediate node, but it covers the elements of classes II and III only. So, MLP_3 corresponding to node A_{13} generates two locations/nodes to distribute this elements. Whereas A_{12} is a leaf/terminal node as it contains the elements of only one class (here class II). Similarly, $A_{21}, A_{2m}, A_{31}, A_{32}, \ldots$, are the leaf/terminal nodes as they cover the elements of single class.

In designing an NNTree for a given dataset, there are two options:

(i) design an NNTree that correctly classifies all the training samples (referred to as a perfect tree), and select the smallest perfect tree; and
(ii) construct an NNTree that is not perfect but has the smallest possible error rate in classification of test samples.

The second type of tree is of greater interest for real life pattern recognition task. Regardless of the type of tree (perfect or otherwise), it is usually desirable to keep the size of the tree as small as possible. Because, smaller trees are more efficient in terms of both tree storage requirements and test time; and tend to generalize better for the unseen test samples that are less sensitive to the statistical irregularities and idiosyncrasies of the training data. So, the basic criteria for NNTree design are as follows:

(i) minimize error rate that would lead to maximum classification accuracy;
(ii) less number of nodes in the tree—that is, minimum number of locations of the selected NNTree; and
(iii) least height of the NNTree.

2.1. Selection of MLP

The following two steps are implemented at each intermediate node to pick up the best possible NNTree:

(i) evaluation of candidate MLPs—that is, evaluation of distribution of the elements of different classes in different locations of an MLP; and
(ii) selection of a location using the best distribution in the intermediate nodes ensuring maximum classification accuracy.

The complexity lies in determining the best distribution for each intermediate node. The optimal NNTree is evolved through the application of back propagation algorithm [17,20] recursively at each intermediate node.

2.2. Splitting and stopping criteria

Splitting an intermediate node involves the design of a new MLP to classify the subset of input elements of different classes covered by the node/location of the MLP of earlier level of the tree. A location is considered as a leaf node if all the training examples falling into the current location belong to the same class. In other words, a node (location) is split as long as there are class elements that belong to different classes.

To avoid overfitting, a prepruning strategy is needed. Let C_{ij} represent the number of elements of class j covered by ith location, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, m$; and β_i indicates the uniformity of the distribution of class elements in the ith location. The value of β_i corresponding to ith node (location) is given by

$$\beta_i = \frac{\mathscr{A}}{\mathscr{B}} \quad \text{where } \mathscr{A} = \max_j\{C_{ij}\}; \quad \mathscr{B} = \sum_{j=1}^{m} C_{ij}. \tag{1}$$

The diversity of the current node is measured as

$$\delta_i = 1 - \beta_i = 1 - \frac{\mathscr{A}}{\mathscr{B}}. \tag{2}$$

When current node (location) is to split, its β_i value is measured and compared with a threshold value ε (= 0.9988).

(i) If $\beta_i < \varepsilon$, then current node is split. That is, partition the examples of ith location.
(ii) If $\beta_i > \varepsilon$, that is, $\delta_i \simeq 0$, then the learning process terminates and the ith location indicates the class j for which C_{ij} is maximum. The future class elements falling into current node (location) are classified to the most probable class of current node— that is, the class that has the maximum number of training examples in current location.
(iii) In some cases, even $\beta_i < \varepsilon$, there exists a possibility where desired MLP is not available. That is, it is not possible to find an MLP which can distribute the given training samples into multiple locations. This occurs when the training examples of different classes are highly correlated. In that case, the learning process is terminated. The future class elements are classified as class j for which C_{ij} is maximum, that is, the most probable class of the current node.

Suppose, after evaluating the distribution of patterns of each class, the pattern set S is partitioned into S_a and S_b, where S_a and S_b represent the pattern set belonging to leaf nodes, and intermediate nodes, respectively. The goodness of the splitting or partition is given by the figure of merit (FM), where

$$\mathrm{FM} = \frac{|S_a|}{|S|} = 1 - \frac{|S_b|}{|S|} \quad \text{where } S_a \cup S_b = S, \tag{3}$$

where $|S|$ represents the cardinality of the set S. The value of FM indicates the classification accuracy of an intermediate/non-terminal node.

The next three sections present the applications of NNTree in letter recognition, satellite image classification, splice-junction and protein coding regions identification in anonymous sequences of DNA, respectively. The proposed method is implemented in C language and run in LINUX environment having machine configuration Pentium IV, 3.2 GHz, 1 MB cache, and 1 GB RAM. For ease of discussions, in rest of the paper, the following terminologies are used:

- η and α represent the learning rate and momentum constant of back propagation algorithm.
- H_n is the number of neurons in the hidden layer of MLP.
- L is the depth of the NNTree, which is equal to the number of levels from the root to the leaf nodes.
- B represents the breadth of the NNTree, which is the number of intermediate nodes in each level of the tree.
- Classification accuracy is defined as the percentage of samples that are correctly classified.
- Classification time is defined as the time required to classify all the samples.

3. Letter and satellite image classification

This section presents the application of NNTree in letter recognition and satellite image classification. An analysis regarding accuracy of the NNTree is also presented here.

3.1. Letter recognition

The dataset used here is constructed by David J. Slate, Odesta Corporation, Evanston, IL 60201. The objective here 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. As one of the fonts used, Gothic Roman, appears very different from the others, perfect classification performance is unlikely to be possible with this dataset [25].

The experimental results on this dataset are presented in Tables 1 and 2 and Figs. 2 and 3. Subsequent discussions analyze the results presented in these tables and figures in respect of classification accuracy—both training and testing.

3.1.1. Generalization of NNTree

Here, the aim is to generate an NNTree that is good at classifying sequences/patterns similar to, but not identical to, patterns in the training set. That is, the generated NNTree has the ability to act as a general pattern classifier.

Tables 1 and 2 represent the generalization capability of the NNTree on letter database. Figs. 2 and 3 show the mean accuracy with error bar at each level of the NNTree. Extensive experiments are performed for different values of η, α, and H_n to evaluate the classification accuracy of both training and test samples. For a particular set of η, α, and H_n, 15 independent runs have been performed by initializing weights of the MLPs with different sets of random numbers. The mean, maximum, and minimum accuracy are calculated based on these 15 runs. The results reported in Figs. 2 and 3

Table 1
Classification accuracy for $\eta = 0.50$

Depth of tree	$\alpha = 0.70$ and $H_n = 15$			$\alpha = 0.40$ and $H_n = 20$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	73.1	66.4	1	78.4	66.2	1
2	87.4	81.0	26	87.8	82.3	26
3	89.7	82.2	142	89.7	82.6	146
4	90.2	82.4	109	90.2	82.8	132
5	90.3	82.5	42	90.4	82.9	40
6	90.3	82.5	8	90.4	82.9	7

Table 2
Classification accuracy for $\eta = 0.70$

Depth of tree	$\alpha = 0.80$ and $H_n = 15$			$\alpha = 0.90$ and $H_n = 20$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	81.5	66.9	1	59.7	57.7	1
2	88.6	81.8	26	85.2	77.1	26
3	90.2	82.2	139	88.7	82.0	206
4	90.8	82.4	112	91.9	82.5	162
5	90.9	82.5	43	93.4	82.6	54
6	90.9	82.5	11	93.5	82.6	22

establish the fact that the difference between mean, minimum, and maximum accuracy of both training and testing significantly reduces with the increase in L irrespective of the values of η, α, and H_n. The mean (average) classification accuracy of training and testing, reported in Tables 1 and 2, confirm that the evolved NNTree can generalize the letter database irrespective of the values of η, α, and H_n. Initially as the value of L increases, the value of B also increases. But, for $L \geqslant 3$, the values of β_l (Eq. (1)) of most of the nodes become equal to 1. In effect, the value of B decreases, as the value of L increases.

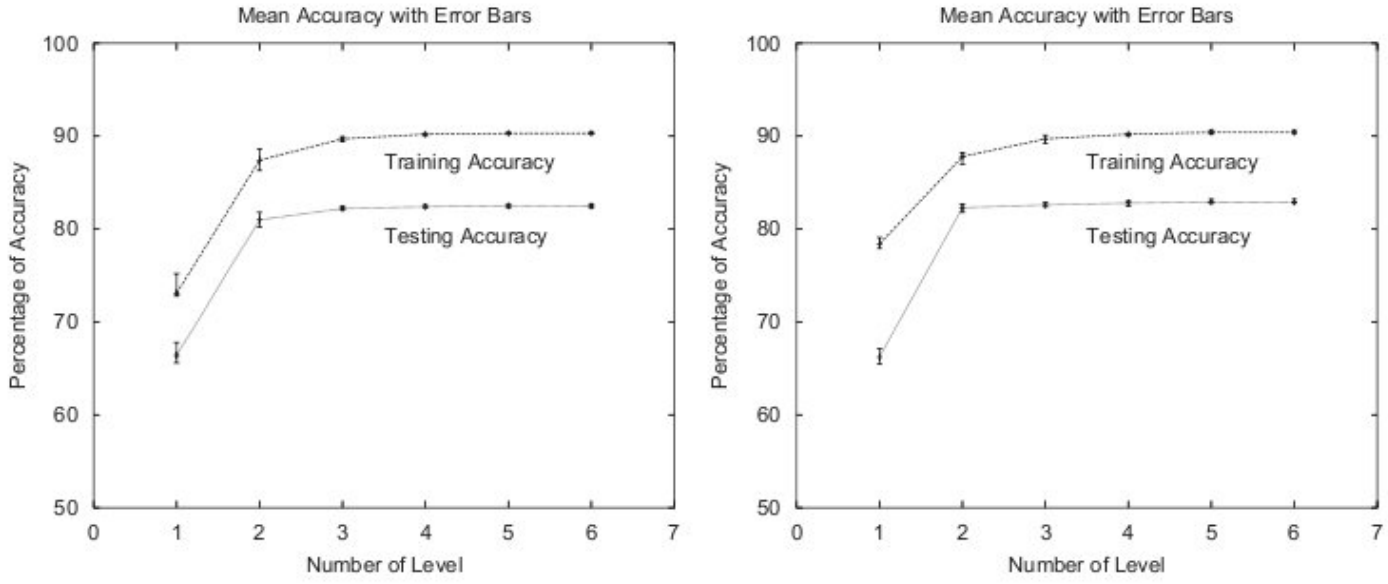

Fig. 2. Performance of NNTree on letter database for $\eta = 0.50$: $\alpha = 0.70$ and $H_n = 15$, and $\alpha = 0.40$ and $H_n = 20$.

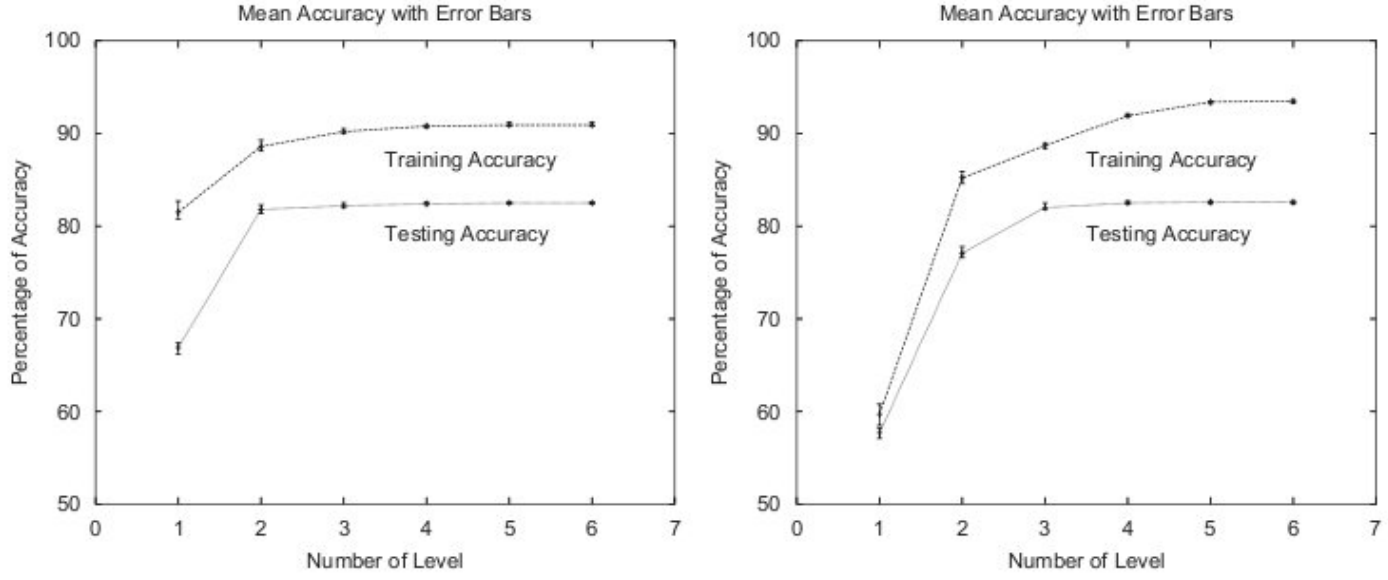

Fig. 3. Performance of NNTree on letter database for $\eta = 0.70$: $\alpha = 0.80$ and $H_n = 15$, and $\alpha = 0.90$ and $H_n = 20$.

Table 3
Performance of NNTree and C4.5 on letter database

Algorithms/methods	Classification accuracy (%)	Total nodes	Height of tree	Classification time (ms)
NNTree	82.9	352	6	8713
C4.5	86.6	2107	20	14,352

Table 4
Classification accuracy of different algorithms for letter database

Algorithms	Accuracy (%)
NNTree	82.9
MLP	67.2
Bayesian	87.4
C4.5	86.6
CA	84.4

Table 5
Classification accuracy for $\eta = 0.50$ and $\alpha = 0.70$

Depth of tree	$H_n = 15$			$H_n = 20$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	86.7	84.6	1	87.5	86.2	1
2	89.9	85.9	6	89.6	86.8	6
3	90.2	86.6	9	90.1	87.1	11
4	90.3	86.6	8	90.1	87.1	6
5	90.3	86.6	1	90.1	87.1	1

Table 6
Classification accuracy for $\eta = 0.70$ and $\alpha = 0.80$

Depth of tree	$H_n = 10$			$H_n = 20$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	85.9	83.7	1	87.3	85.3	1
2	89.3	86.1	6	90.0	86.7	6
3	90.0	86.3	10	90.2	86.8	10
4	90.1	86.3	8	90.3	86.8	5
5	90.1	86.3	1	90.4	86.8	1

3.1.2. Comparison between NNTree and C4.5

Table 3 presents the detailed results of NNTree, comparing each of the factors—classification accuracy, total number of intermediate nodes, height of the tree, and classification time with the most popular decision tree algorithm C4.5 [32]. The source code of C4.5 is obtained from http://www.cse.unsw.edu.au/~quinlan. From the results reported in Table 3, it is established that NNTree achieves comparable classification accuracy with significantly lesser number of intermediate nodes, smaller height of the tree and fast classification time with respect to C4.5.

3.1.3. Classification accuracy

Finally, Table 4 compares the classification accuracy of NNTree with that of different classification algorithms namely Bayesian [5], C4.5 [32], MLP [17,20], and cellular automata (CA) [21]. The experimental results of Table 4 clearly establish the fact that the classification accuracy of NNTree is comparable to that of different algorithms reported in published literature.

3.2. Satellite image classification

The original Landsat data for this database is generated from data purchased from NASA by the Australian Centre 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 [25]. The data are divided into train and test set with 4435 examples in train set and 2000 in test set. The results on this database are presented in Tables 5 and 6 and Figs. 4 and 5 with respect to classification accuracy.

3.2.1. Generalization of NNTree

Tables 5 and 6 represent the generalization capability of the NNTree on satellite database for different values of η, α, and H_n. The mean classification accuracy of training and testing confirm that the evolved NNTree can generalize the satellite database irrespective of the values of η, α, and H_n.

In case of satellite database, as the value of L increases, the value of B also increases. Because, initially for all the nodes, the values of $\beta_l < \varepsilon$. But, for $L \geqslant 3$, the values of β_l of most of the nodes become equal to 1. Hence, the value of B decreases with the increase in L when $L \geqslant 3$.

Figs. 4 and 5 show the mean, minimum, and maximum accuracy of both training and testing on this database. The results show that as the value of L increases, the standard deviations of both training and testing accuracy reduce.

3.2.2. Comparison between NNTree and C4.5

Table 7 compares the performance of NNTree with that of C4.5 with respect to classification accuracy, total number of intermediate nodes, height of the tree, and classification time. From the results reported here confirm that NNTree achieves better performance in terms of classification accuracy, total number of intermediate nodes, height of the tree, and classification time with respect to C4.5.

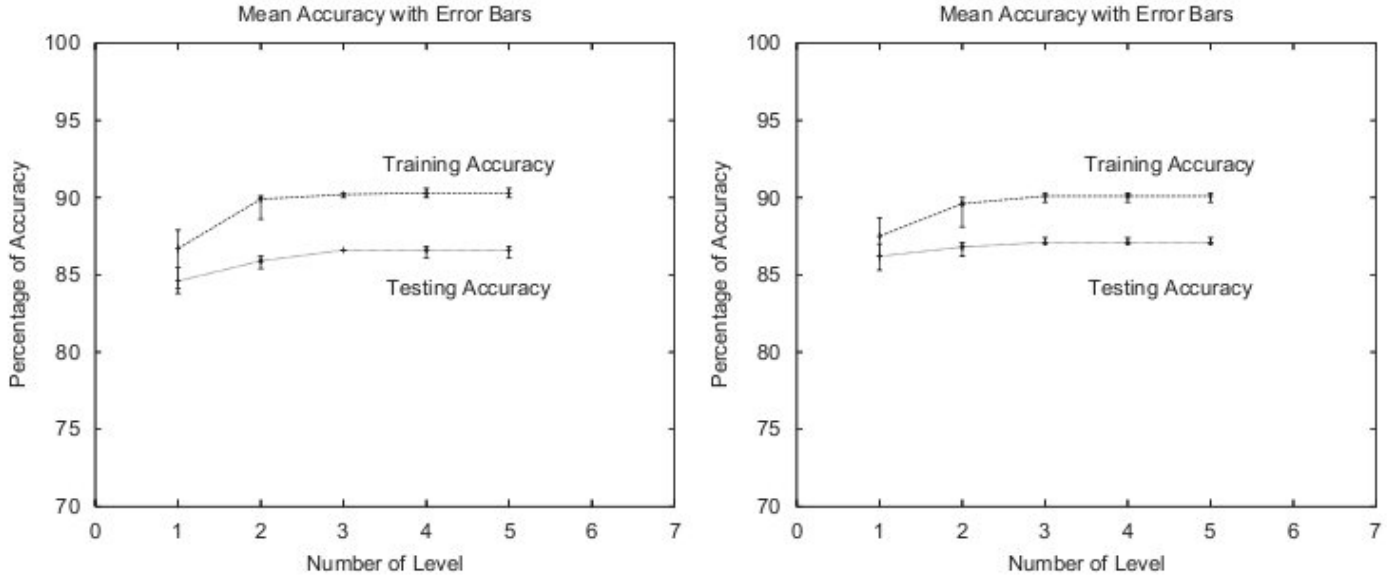

Fig. 4. Performance of NNTree on satellite database for $\eta = 0.50$ and $\alpha = 0.70$: $H_n = 15$ and $H_n = 20$.

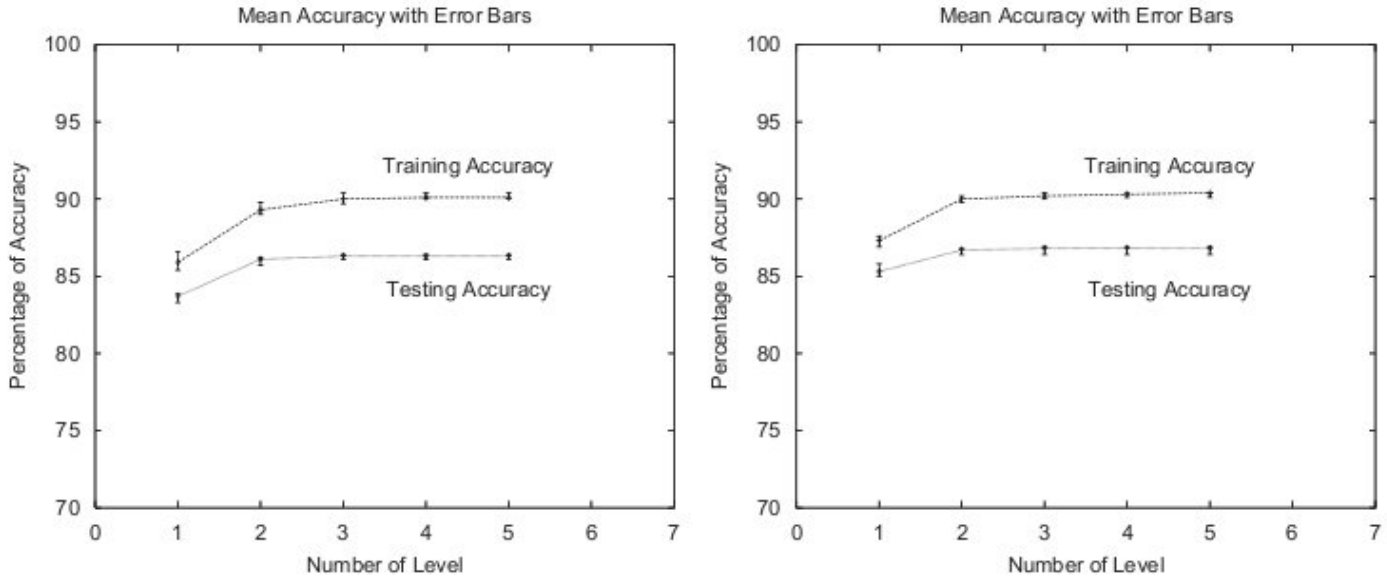

Fig. 5. Performance of NNTree on satellite database for $\eta = 0.70$ and $\alpha = 0.80$: $H_n = 10$ and $H_n = 20$.

3.2.3. Classification accuracy

Finally, Table 8 compares the classification accuracy of NNTree with that of Bayesian [5], C4.5 [32], MLP [17,20], and CA [21]. The experimental results of Table 8 establish that the classification accuracy of NNTree is higher than that of different classification algorithms.

4. Identification of splice-junction in DNA

In this section, the application of NNTree in finding splice-junction in anonymous sequences of DNA is presented. The proposed method is evaluated for benchmark dataset analyzing classification accuracy.

In bioinformatics, one of the major task is the recognition of certain DNA subsequences important in the expression of genes. Basically, a DNA sequence is a string over alphabet $D = \{A, C, G, T\}$. DNA contains the information by which a cell constructs protein molecules. The cellular expression of protein proceeds by the creation of a 'message' ribonucleic acid (mRNA) copy from the DNA template. This mRNA is then translated into a protein. One of the most unexpected findings in molecular biology is that large pieces of the mRNA are removed before it is translated further [3]. The utilized sequences are known as exons while the removed sequences are known as introns, or intervening sequences. The points at which DNA is removed are known as splice-junctions.

Table 7
Performance of NNTree and C4.5 on satellite database

Algorithms/methods	Classification accuracy (%)	Total nodes	Height of tree	Classification time (ms)
NNTree	87.1	25	5	1317
C4.5	85.2	433	21	2255

Table 8
Classification accuracy of different algorithms for satellite database

Algorithms	Accuracy (%)
NNTree	87.1
MLP	86.2
Bayesian	85.4
C4.5	85.2
CA	77.5

Table 9
Classification accuracy for $\eta = 0.50$ and $\alpha = 0.70$

Depth of tree	$H_n = 10$			$H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	99.2	91.4	1	98.7	91.6	1
2	99.6	93.5	3	99.7	94.2	2
3	99.9	93.5	1	99.9	94.2	2

Table 10
Classification accuracy for $\alpha = 0.70$

Depth of tree	$\eta = 0.90$ and $H_n = 10$			$\eta = 0.80$ and $H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	84.2	81.6	1	82.3	82.6	1
2	89.3	84.3	3	87.6	84.9	3
3	91.7	84.6	6	90.3	85.6	6
4	93.7	84.8	8	93.5	85.6	8
5	95.0	84.9	5	95.3	85.7	8
6	96.4	85.1	6	96.5	85.7	7

The splice-junction problem is to determine into which of the following three categories a specified location in a DNA sequence falls: (1) exon/intron borders, referred to as donors; (2) intron/exon borders, referred to as acceptors; and (3) neither.

4.1. Description of dataset

The dataset used in this problem is a processed version of the Irvine Primate splice-junction database [25]. Each of the 3186 examples in the database consists of a window of 60 nucleotides, each represented by one of four symbolic values ($\{A, C, G, T\}$) and the classification of the middle point in the window as one of intron–exon boundary, or neither of these. Processing involved the removal of a small number of examples (4), conversion of the original 60 symbolic attributes to 180 binary attributes and the conversion of symbolic class labels to numeric labels. The training set of 2000 is chosen randomly from the dataset and the remaining 1186 examples are used as the test set.

4.2. Experimental results

The experimental results on dataset reported in earlier subsection are presented in Tables 9–12 and Figs. 6 and 7. Tables 9 and 10 represent the classification accuracy of both training and test samples for different values of η, α, and H_n. The classification accuracy of training and testing confirm that the NNTree can generalize the splice-junction database irrespective of the values of η, α, and H_n. From Figs. 6 and 7, it is seen that the standard deviations of both training and testing accuracy reduce with the increase in L.

For splice-junction database, at $\eta = 0.50$ and $\alpha = 0.70$, most of the nodes of the NNTree have β_i values greater than ε. So, the learning process terminates at $L = 3$ irrespective of the value of H_n. Whereas, for other values of η and α, the values of β_i for most of the nodes of the NNTree are less than ε when $L \leqslant 6$. So, for $L \leqslant 6$, most of the nodes are intermediate nodes. At $L = 7$, though $\beta_i < \varepsilon$ for most of the nodes, the training examples of different classes are so correlated that an MLP cannot be found corresponding to each node, which can classify the dataset present at that node. Hence, the NNTree stops to grow.

Table 11 compares the performance of NNTree with C4.5 with respect to classification accuracy, total number of intermediate nodes, height of the tree, and classification time. For splice-junction database, the classification accuracy of NNTree is higher than that of C4.5, while the number of intermediate nodes, height of the tree and classification time of NNTree are significantly smaller than C4.5.

Finally, Table 12 compares the classification accuracy of NNTree with that of different classification algorithms—Bayesian [5], C4.5 [32], MLP [17,20], and CA [21]. The experimental results of Table 12 clearly establish the fact that the classification accuracy of NNTree is higher than that of different classification algorithms.

5. Identification of protein coding region in DNA

This section presents the application of NNTree for finding protein-coding (exon) regions in anonymous

Table 11
Performance of NNTree and C4.5 on splice-junction database

Algorithms/methods	Classification accuracy (%)	Total nodes	Height of tree	Classification time (ms)
NNTree	94.2	5	3	517
C4.5	93.3	127	12	655

Table 12
Classification accuracy for splice-junction database

Algorithms	Accuracy (%)
NNTree	94.2
MLP	91.4
Bayesian	90.3
C4.5	93.3
CA	87.9

sequences of DNA. The proposed method is evaluated for few sequences and an analysis regarding the accuracy of the proposed method is also presented.

Over the past 20 years, researchers have identified a number of features of exonic DNA that appear to be useful in distinguishing between coding and non-coding regions [2,6,7,41]. These features include both statistical and information-theoretic measures, and in many cases are based on knowledge of the biology underlying DNA sequences and transcription processes. These features are summarized in a survey by Fickett and Tung [8], who also have developed several benchmark features and databases for future experiments on this problem.

Previous research on automatic identification of protein coding regions has considered methods such as linear discriminants [7,8] and neural networks [6,41]. These systems have used measures such as codon frequencies, dicodon frequencies, fractal dimensions, repetitive hexamers, and other features to identify exons in relatively short DNA sequences. The standard experimental study considers a limited window (i.e., a subsequence) of a fixed length, for example 100 base pairs, and computes features based on that window alone. The goal is to identify the window as either all-coding or all-non-coding.

5.1. Data and method

The data used for this study are the human DNA data collected by Fickett and Tung [8]. All the sequences are taken from GenBank in May 1992. Fickett and Tung have provided the 21 different coding measures that they surveyed and compared. The benchmark human data include three different datasets. For the first dataset, non-overlapping human DNA sequences of length 54 have been extracted from all human sequences, with shorter pieces at the ends discarded. Every sequence is labeled according to whether it is entirely coding, entirely non-coding, or mixed, and the mixed sequences (i.e., overlapping the exon-intron boundaries) are discarded. The dataset also includes the reverse complement of every sequence. This means that one-half of the data is guaranteed to be from the non-sense strand of the DNA, which makes the problem of identifying coding regions somewhat harder. For the current study, the same division into training and test data have been used as in the benchmark study [8]. The training set is used exclusively to construct an MLP based tree-structured pattern classifier (NNTree), and the tree is then used to classify the test set. In addition to the 54-base dataset, the datasets containing 108 and 162 bases have been used. The sizes of these datasets are shown in Table 13, which gives the number of non-overlapping windows in each set. No information about reading frames is used in this study. Every window is either all-coding or all-non-coding, but the reading frame of each window is unknown. This choice of window lengths and experimental method follows that used by Fickett and Tung [8], and the problem here is what they defined as protein coding region.

All the attributes in a dataset are normalized to facilitate NNTree learning. Suppose, the possible value range of an attribute attr_i is $(\text{attrval}_{i,\min}, \text{attrval}_{i,\max})$, and the real value that class element j takes at attr_i is attrval_{ij}, then the normalized value of attrval_{ij} is

$$\text{Normalize}(\text{attrval}_{ij}) = \frac{\text{attrval}_{ij} - \text{attrval}_{i,\min}}{\text{attrval}_{i,\max} - \text{attrval}_{i,\min}} \quad (4)$$

Next subsection presents an extensive experimental analysis regarding the classification accuracy of the proposed MLP based tree-structured pattern classifier.

5.2. Experimental results

In this subsection, the results of NNTree for Fickett and Tung's dataset are presented. Values are given for the percentage accuracy on both training and test set. Results of NNTree on each of the dataset are given in Tables 14-19. The mean accuracy of training and testing confirm that the evolved NNTree can generalize the datasets presented in Table 13 irrespective of the number of attributes, tuples, α, η, and H_n.

In case of Fickett and Tung database, for $L \leqslant 6$, the values of β_l for all possible nodes/locations of the NNTree are less than ε. So, all the nodes are intermediate/non-terminal nodes for $L \leqslant 6$. Hence, the NNTree have been

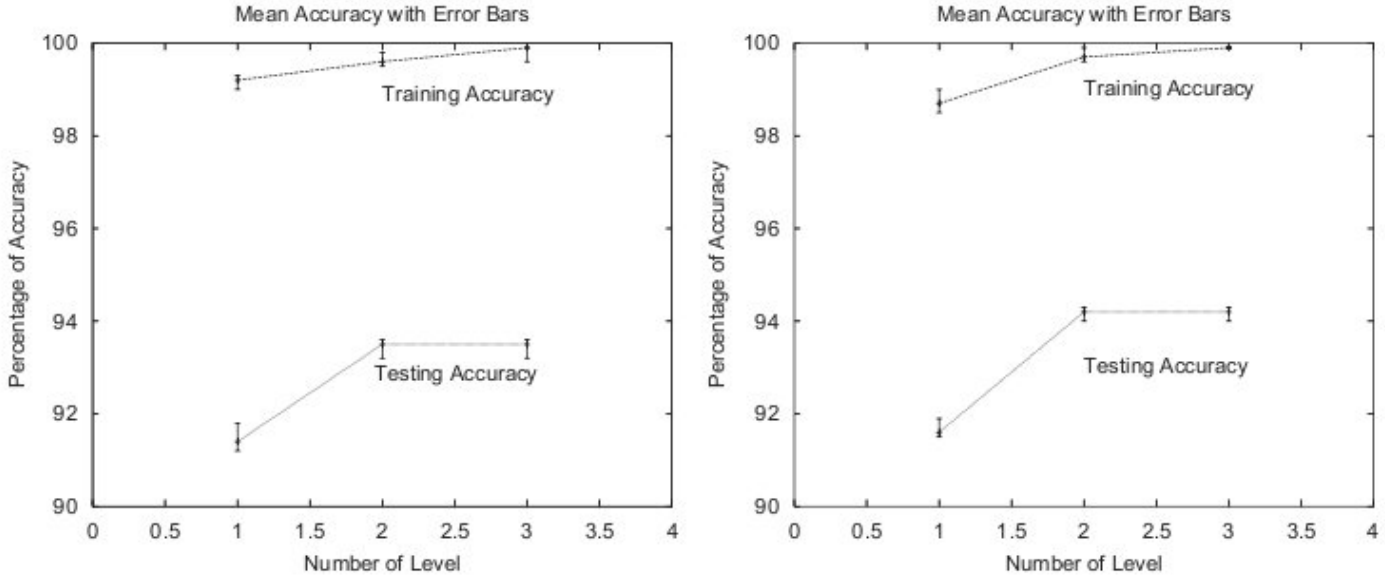

Fig. 6. Performance of NNTree on splice-junction database for $\eta = 0.50$ and $\alpha = 0.70$: $H_n = 10$ and $H_n = 15$.

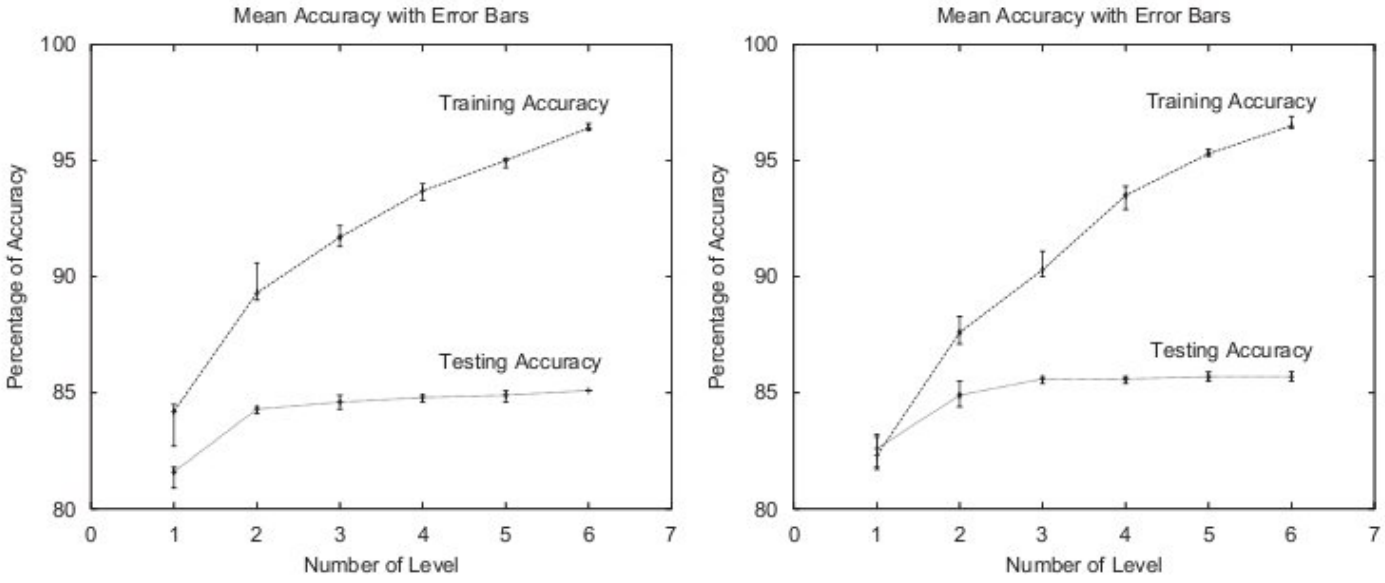

Fig. 7. Performance of NNTree on splice-junction database for $\alpha = 0.70$: $\eta = 0.90$ and $H_n = 10$, and $\eta = 0.80$ and $H_n = 15$.

grown by splitting all these non-terminal nodes. At $L = 7$, though the value of $\beta_i < \varepsilon$ for each non-terminal node, the training samples of two classes in each non-terminal node are highly correlated. So, at $L = 7$, an MLP cannot be found, corresponding to an intermediate node, which can classify the training samples. So, the learning process terminates at this stage, and the nodes are considered as leaf nodes indicating the class that has the maximum number of training examples in current location.

Figs. 8–10 show the classification accuracy with error bar of NNTree on different DNA sequences. All the results reported in Figs. 8–10 establish the fact that the proposed NNTree can generalize a DNA dataset irrespective of its sequence length. Also, the standard deviations of training and testing accuracy are very small.

Table 20 compares the classification accuracy of NNTree with that of OC1 [28,29], MLP, and other related algorithms. The OC1, proposed by Murthy et al. [28,29], is an oblique decision tree algorithm that combined deterministic hill-climbing with two forms of randomization to find a good oblique split at each intermediate node of a decision tree. All the results reported in Table 20 establish the fact that the classification accuracy of NNTree is higher than that of existing algorithms. Also,

Table 13
Benchmark datasets proposed by Fickett and Tung

Dataset	Human 54	Human 108	Human 162
Training set—coding	20,456	7086	3512
Training set—non-coding	125,132	58,118	36,502
Training set total	145,588	65,204	40,014
Test set—coding	22,902	8192	4226
Test set—non-coding	122,138	57,032	35,602
Test set total	145,040	65,224	39,868

Table 14
Human DNA 54 bp at $\eta = 0.50$ and $\alpha = 0.70$

Depth of tree	$H_n = 10$			$H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	57.2	54.7	1	57.5	54.9	1
2	84.6	82.5	2	84.5	81.8	2
3	85.2	82.6	4	85.4	82.0	4
4	86.0	82.7	8	86.0	82.1	8
5	86.5	82.9	16	86.4	82.2	16
5	86.5	82.9	16	86.7	82.4	32

Table 15
Human DNA 54 bp at $\eta = 0.70$ and $\alpha = 0.70$

Depth of tree	$H_n = 10$			$H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	55.6	53.2	1	53.6	50.2	1
2	77.3	75.5	2	76.1	71.3	2
3	80.0	78.0	4	83.3	78.7	4
4	82.2	79.9	8	85.2	82.4	8
5	83.5	80.9	16	85.3	82.4	16
6	84.4	81.3	32	85.7	82.6	32

Table 16
Human DNA 108 bp at $\eta = 0.50$ and $\alpha = 0.70$

Depth of tree	$H_n = 10$			$H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	58.4	55.7	1	59.0	56.1	1
2	86.8	82.5	2	87.1	81.0	2
3	88.3	82.6	4	87.8	81.8	4
4	89.6	82.7	8	89.3	82.9	8
5	90.2	82.8	16	90.1	83.5	16
6	90.7	83.1	32	92.2	83.5	32

the results reported here establish the fact that the proposed NNTree can generalize a DNA dataset irrespective of its sequence length.

Table 17
Human DNA 108 bp at $\eta = 0.70$ and $\alpha = 0.70$

Depth of tree	$H_n = 10$			$H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	55.3	52.5	1	57.4	55.0	1
2	78.9	76.3	2	77.6	74.4	2
3	82.5	79.7	4	85.2	79.3	4
4	84.9	81.9	8	90.8	82.7	8
5	86.5	82.6	16	93.5	82.9	16
6	87.7	83.4	32	93.7	83.5	32

Table 18
Human DNA 162 bp at $\eta = 0.50$ and $\alpha = 0.70$

Depth of tree	$H_n = 10$			$H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	59.1	56.9	1	61.0	57.5	1
2	83.5	77.3	2	81.1	71.5	2
3	85.1	78.8	4	84.9	79.6	4
4	88.0	82.9	8	89.9	83.7	8
5	91.4	84.2	16	91.2	84.3	16
6	91.7	84.4	32	91.3	84.3	32

Table 19
Human DNA 162 bp at $\eta = 0.70$ and $\alpha = 0.70$

Depth of tree	$H_n = 10$			$H_n = 15$		
	Training	Testing	Breadth	Training	Testing	Breadth
1	55.2	53.3	1	58.3	52.8	1
2	82.8	72.5	2	77.8	70.1	2
3	88.1	77.6	4	85.9	76.8	4
4	90.9	83.9	8	89.9	82.3	8
5	93.1	84.2	16	92.7	84.0	16
6	93.1	84.2	32	93.2	84.2	32

6. Conclusion and future directions

This paper presents the design of a hybrid learning algorithm, termed as NNTree. It uses multilayer perceptron for designing tree-structured pattern classifier. Instead of using information gain ratio as splitting criterion, a new criterion is introduced in this paper for NNTree design. This criterion captures well the intuitive goal of reducing the rate of misclassification.

The performance of NNTree is evaluated through its applications in letter recognition, satellite image classification, splice-junction and protein coding region identification. Experimental comparison with other related algorithms provide better or comparable classification accuracy with significantly smaller trees and fast classification

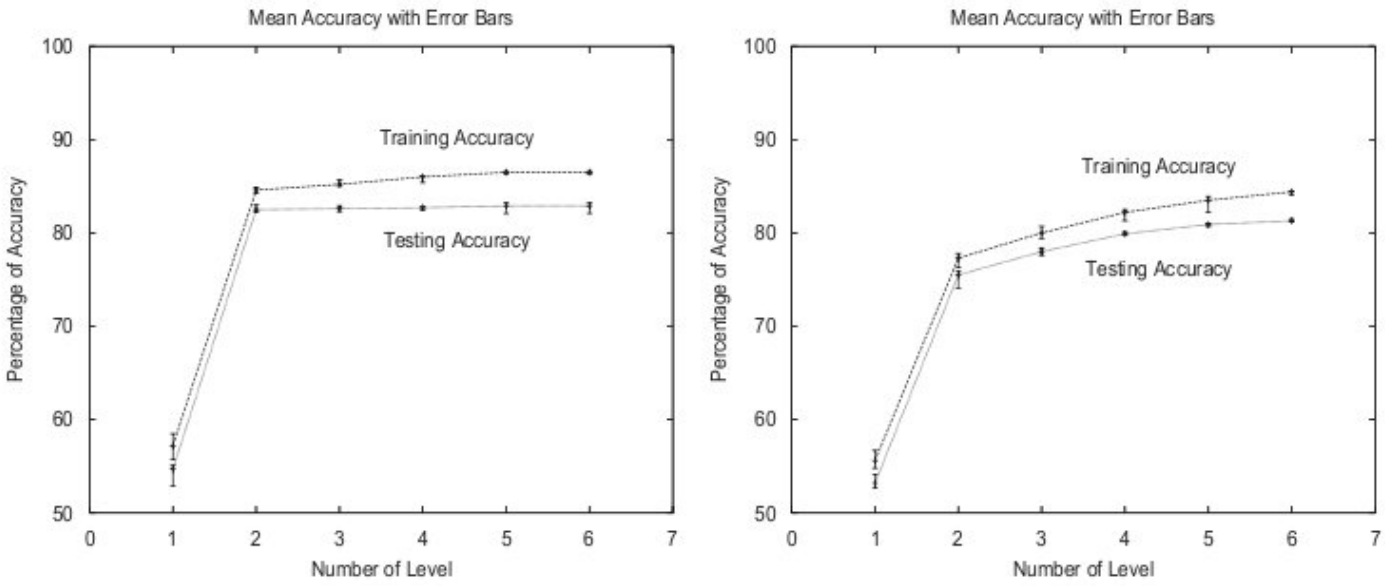

Fig. 8. Performance of NNTree on 54bp human DNA sequence for $\alpha = 0.70$ and $H_n = 10$: $\eta = 0.50$ and $\eta = 0.70$.

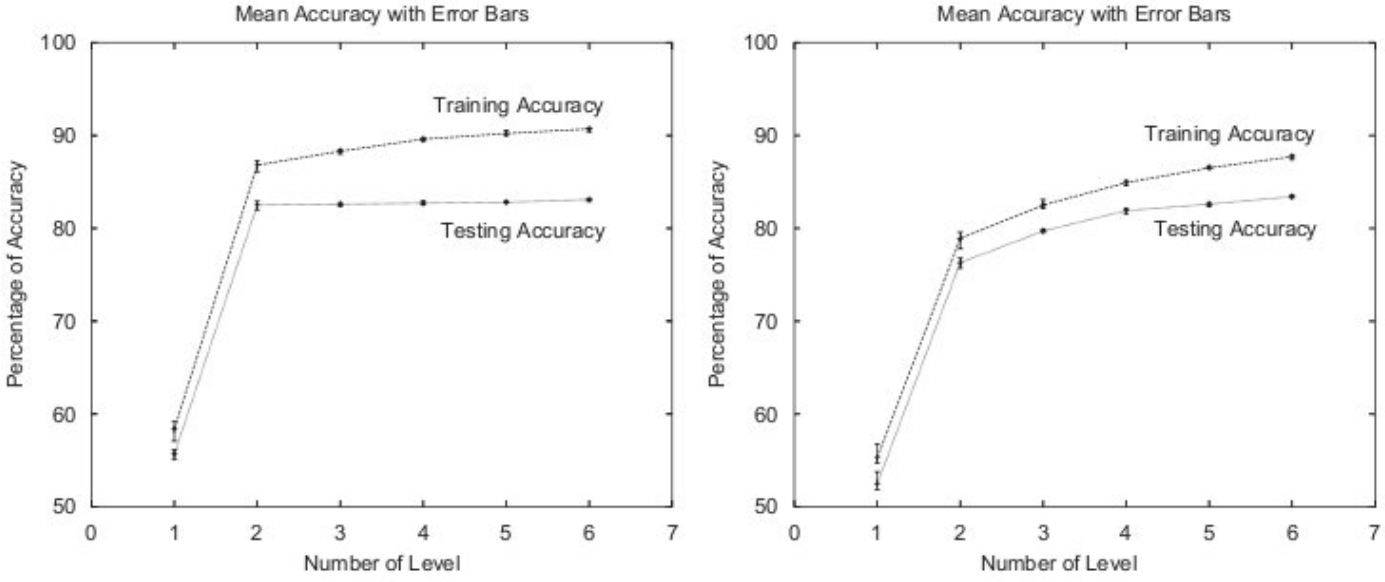

Fig. 9. Performance of NNTree on 108bp human DNA sequence for $\alpha = 0.70$ and $H_n = 10$: $\eta = 0.50$ and $\eta = 0.70$

times. Also, it performs well for datasets with large number of examples and attributes. Extensive experimental results reported in this paper confirm that the proposed NNTree is crucial over conventional techniques for classification.

Also, the sizes of the trees produced by both C4.5 and NNTree have been compared in terms of total number of nodes and height of the trees. A smaller tree is desirable since it provides more compact class descriptions, unless the smaller tree size leads to a loss in accuracy. The results show that the NNTree achieves trees that are significantly smaller than the trees generated by C4.5.

The promise exhibited by the NNTree suggests several avenues for further investigation. One possible improvement would be to include fuzzy multilayer perceptron in each intermediate node of the tree to handle uncertainties in class definition. Aside from developing a good classifier, the proposed model may be very much useful to solve many other related problems like protein structure prediction, RNA structure prediction, etc. The investigation, besides having significance in pattern classification and bioinformatics research, has potential in soft computing research.

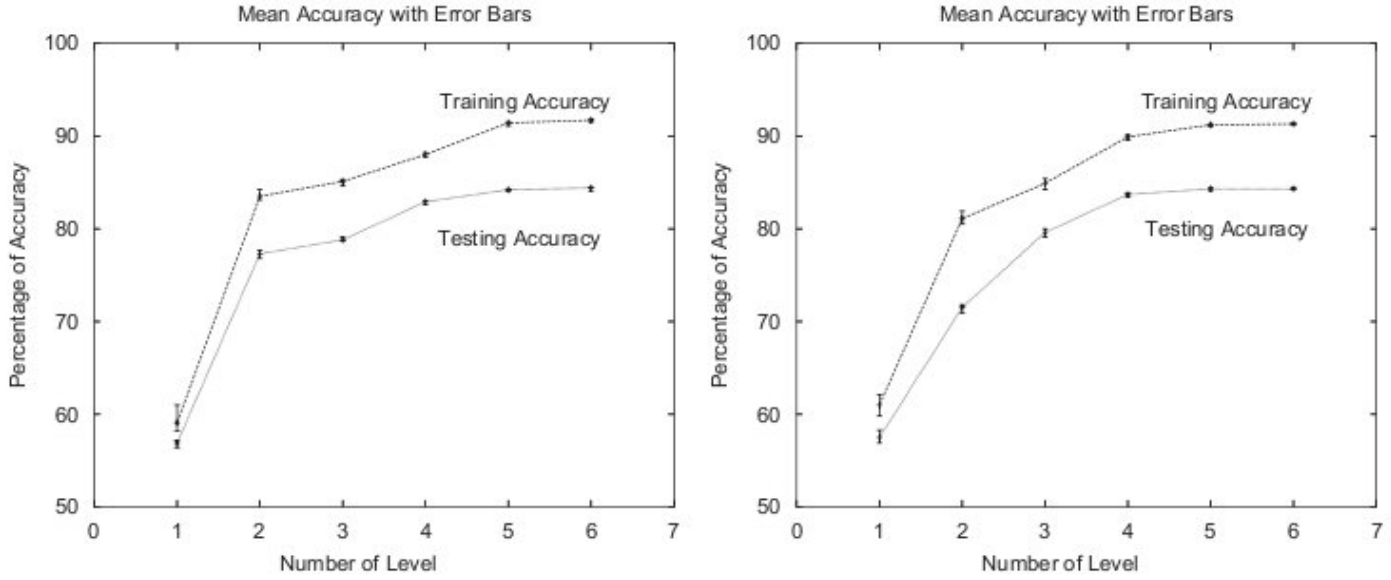

Fig. 10. Performance of NNTree on 162bp human DNA sequence for $\eta = 0.50$ and $\alpha = 0.70$: $H_n = 10$ and $H_n = 15$

Table 20
Classification accuracy for human DNA 54, 108, and 162 bp

Algorithms	54 bp	108 bp	162 bp
NNTree	82.9	83.5	84.4
OC1	73.9	83.7	84.2
MLP	54.9	56.1	57.5
Position asymmetry	70.7	77.6	81.7
Fourier	69.5	77.4	82.0
Hexamer	69.8	71.4	73.8
Dicodon usage	69.8	71.2	73.7

Acknowledgments

The author would like to thank the anonymous referees for providing helpful comments and valuable criticisms on the original version of the manuscript which have greatly improved the presentation of the paper.

References

[1] R. Agrawal, T. Imielinski, A. Swami, Database mining: a performance perspective, IEEE Trans. Knowl. Data Eng. 5 (6) (1993) 914–925.
[2] B.E. Blaisdell, A prevalent persistent global nonrandomness that distinguishes coding and non-coding eucaryotic nuclear DNA sequence, J. Mol. Evol. 19 (2) (1983) 122–133.
[3] R.J. Breathnach, J.L. Mandel, P. Chambon, Ovalbumin gene is split in chicken DNA, Nature 270 (5635) (1977) 314–319.
[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, 1984.
[5] P. Cheeseman, J. Stutz, Bayesian classification (AutoClass): theory and results, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA, 1996, pp. 153–180.
[6] R. Farber, A. Lapedes, K. Sirotkin, Determination of Eucaryotic protein coding regions using neural networks and information theory, J. Mol. Biol. 226 (2) (1992) 471–479.
[7] J. Fickett, Recognition of protein coding regions in DNA sequences, Nucleic Acids Res. 10 (17) (1982) 5303–5318.
[8] J. Fickett, C.S. Tung, Assessment of protein coding measures, Nucleic Acids Res. 20 (24) (1992) 6441–6450.
[9] D. Fisher, Improving inference through conceptual clustering, Proceedings of AAAI Conference, Seattle, Washington, 1987.
[10] S. Fotheringham, E.P. Rogerson, Spatial Analysis and GIS, Taylor & Francis, London, 1994.
[11] D.E. Goldberg, Genetic Algorithms in Search, Optimizations and Machine Learning, Morgan Kaufmann, Los Atlos, CA, 1989.
[12] H. Guo, S.B. Gelfand, Classification trees with neural network feature extraction, IEEE Trans. Neural Networks 3 (6) (1992) 923–933.
[13] E.H.S. Han, G. Karypis, V. Kumar, Scalable parallel data mining for association rules, IEEE Trans. Knowl. Data Eng. 12 (3) (2000) 337–352.
[14] J. Han, Y. Cai, N. Cercone, Data-driven discovery of quantitative rules in relational databases, IEEE Trans. Knowl. Data Eng. 5 (1) (1993) 29–40.
[15] J. Han, Y. Fu, Exploration of the Power of Attribute-Oriented Induction in Data Mining, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI, MIT Press, Cambridge, MA, 1996, pp. 399–421.
[16] J. Han, M. Kamber, Data Mining, Concepts and Techniques, Morgan Kaufmann, Los Atlos, CA, 2001.
[17] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the theory of neural computation, Santa Fe Institute Studies in the Sciences of Complexity, Addison Wesley, Reading, MA, 1991.
[18] L. Kaufmann, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.
[19] J.R. Koza, Genetic Programming—I, The MIT Press, Cambridge, MA, 1994 (Fourth Printing).
[20] R. Lippmann, An introduction to computing with neural nets, IEEE Acoust. Speech Signal Process. Mag. 4 (2) (1987) 4–22.

[21] P. Maji, C. Shaw, N. Ganguly, B.K. Sikdar, P.P. Chaudhuri, Theory and application of cellular automata for pattern classification, Fundamenta Informaticae 58 (3–4) (2003) 321–354.
[22] C.J. Matheus, P.K. Chan, G. Piatetsky-Shapiro, Systems for knowledge discovery in databases, IEEE Trans. Knowl. Data Eng. 5 (6) (1993) 903–913.
[23] M. Mehta, R. Agrawal, J. Rissanen, SLIQ: a fast scalable classifier for data mining, in: Proceedings of International Conference on Extending Database Technology, Avignon, France, 1996.
[24] R.S. Michalski, J.M. Carbonnel, T.M. Mitchell, Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann, Los Atlos, CA, 1983.
[25] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, Chichesten, UK, 1994.
[26] J. Mingers, Expert systems—Rule induction with statistical data, J. Oper. Res. Soc. 38 (1) (1987) 39–47.
[27] T.M. Mitchell, Generalization as search, Artif. Intell. 18 (1982) 203–226.
[28] S.K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, J. Artif. Intell. Res. 2 (1994) 1–32.
[29] S.K. Murthy, S. Kasif, S. Salzberg, R. Beigel, OC1 randomized induction of oblique decision trees, Proceedings of the 11th National Conference on Artificial Intelligence, The AAAI Press, The MIT Press, Cambridge, MA, 1993, pp. 322–327.
[30] G. Piatetsky-Shapiro, U. Fayyad, P. Smith, From data mining to knowledge discovery: an overview, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI, MIT Press, Cambridge, MA, 1996, pp. 1–35.
[31] G. Piatetsky-Shapiro, W.J. Frawley, Knowledge Discovery in Databases, AAAI, MIT Press, Cambridge, MA, 1991.
[32] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Atlos, CA, 1993.
[33] G.P. Schmitz, C. Aldrich, F.S. Gouws, ANN-DT: an algorithm for extraction of decision trees from artificial neural networks, IEEE Trans. Neural Networks 10 (6) (1999) 1392–1401.
[34] I.K. Sethi, Entropy nets: from decision trees to neural networks, Proc. IEEE 78 (10) (1990) 1605–1613.
[35] I.K. Sethi, J.H. Yoo, Structure-driven induction of decision tree classifiers through neural learning, Pattern Recognition 30 (11) (1997) 1893–1904.
[36] J. Shafer, R. Agrawal, M. Mehta, SPRINT: a scalable parallel classifier for data mining, in: Proceedings of 22nd VLDB Conference, 1996.
[37] G. Shaw, D. Wheeler, Statistical Techniques in Geographical Analysis, David Fulton, London, 1994.
[38] H.H. Song, S.W. Lee, A self-organizing neural network tree for large-set pattern classification, IEEE Trans. Neural Networks 9 (6) (1998) 369–380.
[39] M. Stonebraker, Readings in Database Systems, second ed., Morgan Kaufmann, Los Atlos, CA, 1993.
[40] H. Tsukimoto, Extracting rules from trained neural networks, IEEE Trans. Neural Networks 11 (2) (2000) 377–389.
[41] E. Uberbacher, R. Mural, Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach, Proc. Natl. Acad. Sci. USA 88 (24) (1991) 11261–11265.
[42] Q.F. Zhao, Neural network tree: integration of symbolic and nonsymbolic approaches, Vol. NC2000-57, Technical Report of IEICE, 2000.
[43] Q.F. Zhao, Evolutionary design of neural network tree: integration of decision tree, neural network and GA, in: Proceedings of IEEE Congress on Evolutionary Computation 2001, pp. 240–244
[44] Q.F. Zhao, Training and retraining of neural network trees, in: Proceedings of INNS-IEEE International Joint Conference on Neural Networks, 2001, pp. 726–731.
[45] Z.H. Zhou, Z.Q. Chen, Hybrid decision tree, Knowledge Based System 15 (8) (2002) 515–528.

Dr. Pradipta Maji received B.Sc (Hons) in Physics in 1998, M.Sc in Electronics Science in 2000, and Ph.D. in Computer Science in 2005, all from Jadavpur University, India. Currently, he is a Lecturer at the Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India. He is also associated with the Center for Soft Computing Research: A National Facility, Indian Statistical Institute, Kolkata, India. He has published more than 30 papers in international journals and conferences. He is also reviewer of many international journals. His research interests include pattern recognition, bioinformatics, medical image processing, cellular automata, neural networks, soft computing, etc.