

Chapter 2

Neural Network Tree for Identification of Splice Junction and Protein Coding Region in DNA

2.1 Introduction

The internetworked society has been experiencing an explosion of biological data. However, the explosion is paradoxically acting as an impediment in acquiring knowledge. The meaningful interpretation of this large volume of biological data is increasingly becoming difficult. Consequently, researchers, practitioners, and entrepreneurs from diverse fields are trying to develop sophisticated techniques to store, analyze, and interpret this biological data for knowledge extraction, which leads to evolve the new field called bioinformatics. This field has arisen in parallel with the development of automated high-throughput methods of pattern recognition and machine learning. The development of high-throughput methods for biological and biochemical discovery yields a variety of experimental data such as DNA sequences, gene expression patterns, chemical structures, and so forth. Hence, bioinformatics encompasses everything from data storage and retrieval to the identification and presentation of features within data such as finding genes within DNA sequence, finding similarities between sequences, structural predictions, and correlation between sequence variation and clinical data [1, 4–6, 25].

Two of the important problems in bioinformatics are splice-site or splice-junction prediction and identification of protein coding regions in DNA sequences. Genes contain their information as a specific sequence of nucleotides or bases that are found in DNA molecules. These specific sequences of bases encode instructions on how to make proteins. The regions of a gene that code for proteins are termed as exons. These exons occupy only a small region of the gene. Whereas in prokaryotic gene, the mRNA (messenger ribonucleic acid) is a mere transcribed copy of the DNA, in eukaryotic gene, the RNA copy of the DNA contains noncoding segments, which are termed as introns, and they should be precisely spliced out to produce the mRNA. This means that introns are parts of a gene that are not used in protein synthesis and exons are the protein coding regions in that gene. The points at which DNA is removed are known as splice sites. The splice-site identification 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. Another important problem is the identification of protein coding region, that is, exon in anonymous sequences of DNA. Identifying the coding regions and splice sites is of vital importance in understanding the processing of genes.

Many new mathematical or computational approaches are being introduced as well as existing ones getting refined, but the search for new and better solutions continues specifically to analyze large volume of DNA data sets generated in the internetworked society of cyber-age. To address these problems, a new neural network tree (NNTree) based pattern classifier [11, 12] is presented in this chapter for finding the splice-site and protein coding regions in DNA sequences. The idea of this new method is to use the framework of decision tree and neural network.

Over the years, the decision trees (DT) are successfully used in many diverse areas such as radar signal classification, character recognition, remote sensing, medical diagnosis, expert system, speech recognition, and also in other different fields [18]. The decision tree classifier is one of the possible approaches in multistage decision making. The most important feature of DT is their capability to break up a complex decision into a union of several simpler decisions hoping the final solution obtained this way would resemble the intended desired solution. On the other hand, the subject of artificial neural network (ANN) has become very popular in many areas such as signal processing and pattern recognition [8, 10, 23, 26]. Additionally, neural networks are models of nonsymbolic approaches. However, nonsymbolic learners are usually black boxes. It is not known what has been learned ever if correct answers are got. Another key problem in using neural networks is that the number of free parameters is usually too large to be determined efficiently.

Even though neural networks and DT are two very different techniques for pattern recognition or classification, both are capable of generating arbitrarily complex decision boundaries from a given set of training samples or training examples, and usually neither has to make any assumptions about the distribution of the underlying processes. The neural networks are usually more capable of providing incremental learning than DT, whereas decision trees are sequential in nature, compared to massive parallelism usually present in neural networks. Thus, DT are typical models for symbolic approaches, and neural networks are models for nonsymbolic approaches. Basically, symbolic approaches can provide comprehensive rules but cannot adapt to changing environments efficiently. On the contrary, nonsymbolic approaches can adapt to changing environments but cannot provide comprehensible rules.

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 [20]. 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 [19] uses neural network to generate outputs for examples interpolated from the training data set and then extracts a univariate binary decision tree from the network. Another method which also extracts decision tree from neural network is reported

in [24]. The design of a tree-structured neural network using genetic programming is proposed in [9]. In [7, 22, 27–29], designs of NNTree have been introduced. The NNTree is a decision tree with each nonterminal node being a neural network. In [21], 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 [30] 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 nonterminal nodes of the tree. There are many criteria for splitting nonterminal nodes. One of the most popular criteria is the information gain ratio which is used in C4.5 [18]. Designs of NNTree have so far mostly concentrated around binary tree with information gain ratio used to partition the available data set at each nonterminal node [7, 22, 27–29]. 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 the NNTree to biological data set.

In the above background, this chapter presents the design and applications of an NN-based tree-structured pattern classifier (NNTree) to address the problem of finding splice-site and protein coding region in DNA sequences [11, 12]. The NNTree reported here adopts an approach which is completely different from the methods mentioned in [7, 22, 24, 27–30]. The neural networks used in this design are multilayer perceptrons (MLP) with m output nodes; m being the number of classes in the given data set. Unlike [7, 22, 27–30], the NNTree designed here splits each nonterminal node by maximizing (respectively, minimizing) classification accuracy (respectively, classification error) of the MLP rather than using information gain ratio. So, the current design always generates a reduced height m -ary tree. The backpropagation algorithm is used recursively at each nonterminal node to find a multilayer perceptron. The effectiveness of the new algorithm, along with a comparison with individual components of the hybrid scheme as well as other related algorithms, has been demonstrated on several benchmark data sets.

The structure of the rest of this chapter is as follows: Sect. 2.2 presents the design of a neural network based tree-structured pattern classifier, called NNTree. Sections 2.3 and 2.4 present the application of the NNTree in splice-junction and protein coding region identification problems, respectively. In order to validate the design of current model, extensive experimental results are also reported in these sections. Concluding remarks are given in Sect. 2.5.

2.2 Neural Network Based Tree-Structured Pattern Classifier

A neural network tree (NNTree) is a decision tree with each intermediate or nonterminal node being a 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

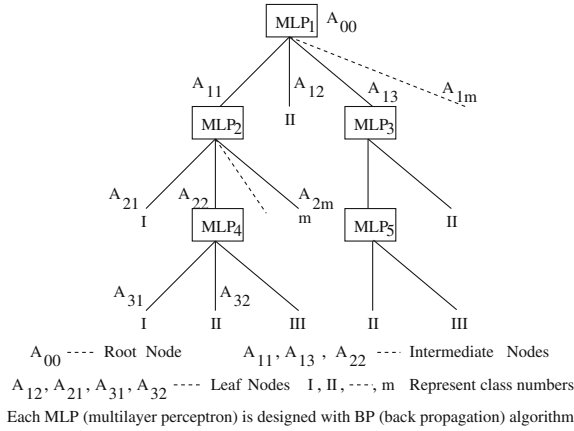


Fig. 2.1 MLP-based tree-structured pattern classifier

which nodes are terminal nodes, and assigning class labels to terminal nodes. In this tree, a leaf or terminal node covers the set or subset of elements of only one class. By contrast, an intermediate node covers the set or subset of elements belonging to more than one class. Thus, the NNtrees are class discriminators which recursively partition the training set to get nodes belonging to a single class. Figure 2.1 shows an MLP-based NNtree.

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

Let, \hat{S} be the set of elements in a node. If \hat{S} belongs to only one class, then label that node as that class. Otherwise, this process is repeated recursively for each node until all the patterns in each node or location belong to only one class. Single or multiple nodes of the tree may form a leaf or terminal node representing a class, or it may be an intermediate or nonterminal node. A leaf node represents a location that contains the set or 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 next.

Input: Training set $S = \{S_1, \dots, S_i, \dots, S_m\}$

Output: NNtree (set of MLPs)

Partition(S, m);

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

1. Generate an MLP with \tilde{m} output neurons.
2. Distribute the training set \tilde{S} into \tilde{m} locations (nodes).
3. Evaluate the distribution of patterns in each node.
4. If all the patterns (\tilde{S}) of a location (node) belong to one particular class, then label the location (leaf node) for that class.
5. If for the set of patterns (\tilde{S}) of a location belonging to \acute{m} classes, **Partition**(\tilde{S}, \acute{m}).
6. End.

In Fig. 2.1, the node A_{00} is the root node. So, the MLP₁ corresponding to node A_{00} distributes the training set $S = \{S_1, \dots, S_i, \dots, S_m\}$ into m locations denoted by $A_{11}, A_{12}, \dots, A_{1m}$. 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 MLP₂ into m number of locations - $A_{21}, A_{22}, \dots, A_{2m}$. A_{13} is also an intermediate node, but it covers the elements of classes II and III only. So, MLP₃ corresponding to node A_{13} generates two locations or nodes to distribute these elements. Whereas A_{12} is a leaf or terminal node as it contains the elements of only one class (here class II). Similarly, $A_{21}, A_{2m}, A_{31}, A_{32}, \dots$, are the leaf or terminal nodes as they cover the elements of single class.

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

1. design an NNTree that correctly classifies all the training samples (referred to as a perfect tree), and select the smallest perfect tree; and
2. 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 both in terms of 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 the NNTree design are as follows:

1. minimize error rate that would lead to maximum classification accuracy;
2. less number of nodes in the tree, that is, minimum number of locations of the selected NNTree; and
3. least height of the NNTree.

2.2.1 Selection of Multilayer Perceptron

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

1. evaluation of candidate MLPs, that is, evaluation of distribution of the elements of different classes in different locations of an MLP; and

2. 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 [8, 10] recursively at each intermediate node.

2.2.2 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 or 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} be the number of elements of class j covered by i th location, where $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, m$; and β_i indicates the uniformity of the distribution of class elements in the i th location. The value of β_i corresponding to i th node (location) is given by

$$\beta_i = \frac{\mathcal{A}}{\mathcal{B}}; \quad (2.1)$$

$$\text{where } \mathcal{A} = \max_j \{C_{ij}\}; \text{ and } \mathcal{B} = \sum_{j=1}^m C_{ij}. \quad (2.2)$$

The diversity of the current node is measured as

$$\delta_i = 1 - \beta_i = 1 - \frac{\mathcal{A}}{\mathcal{B}}. \quad (2.3)$$

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

1. If $\beta_i < \varepsilon$, then current node is split. That is, partition the examples of i th location.
2. If $\beta_i > \varepsilon$, that is, $\delta_i \simeq 0$, then the learning process terminates and the i th 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.
3. 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 examples 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

$$FM = \frac{|S_a|}{|S|} = 1 - \frac{|S_b|}{|S|}; \quad (2.4)$$

where $S_a \cup S_b = S$ and $|S|$ represent the cardinality of the set S . The value of FM indicates the classification accuracy of an intermediate or nonterminal node.

For ease of discussions, in the rest of the chapter, 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.

The NNTree 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.

2.3 Identification of Splice-Junction in DNA Sequence

In this section, the application of the NNTree in finding the splice junction in anonymous sequences of DNA is presented. The performance of the NNTree is evaluated for benchmark data set analyzing classification accuracy.

In bioinformatics, one of the major tasks is the recognition of certain DNA subsequences those are 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 messenger 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 [2]. 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. The splice-junction problem is to determine into which of the following three categories a specified location in a DNA sequence falls: (1)

Table 2.1 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 2.2 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

exon/intron borders, referred to as donors; (2) intron/exon borders, referred to as acceptors; and (3) neither.

2.3.1 Description of Data Set

The data set used in this problem is a processed version of the Irvine Primate splice-junction database [14]. Each of the 3,186 examples in the database consists of a window of 60 nucleotides, each represented by one of four symbolic values ($\{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$) 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 four examples, 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 2,000 is chosen randomly from the data set and the remaining 1,186 examples are used as the test set.

2.3.2 Experimental Results

The experimental results on data set reported in earlier subsection are presented in Tables 2.1, 2.2, 2.3, 2.4, Figs. 2.2, 2.3. Tables 2.1 and 2.2 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 confirms that the

Table 2.3 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 2.4 Classification accuracy for splice-junction database

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

NNTree can generalize the splice-junction database irrespective of the values of η , α , and H_n . From Figs. 2.2 and 2.3, 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 $L \leq 6$. So, for $L \leq 6$, 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 data set present at that node. Hence, the NNTree stops to grow.

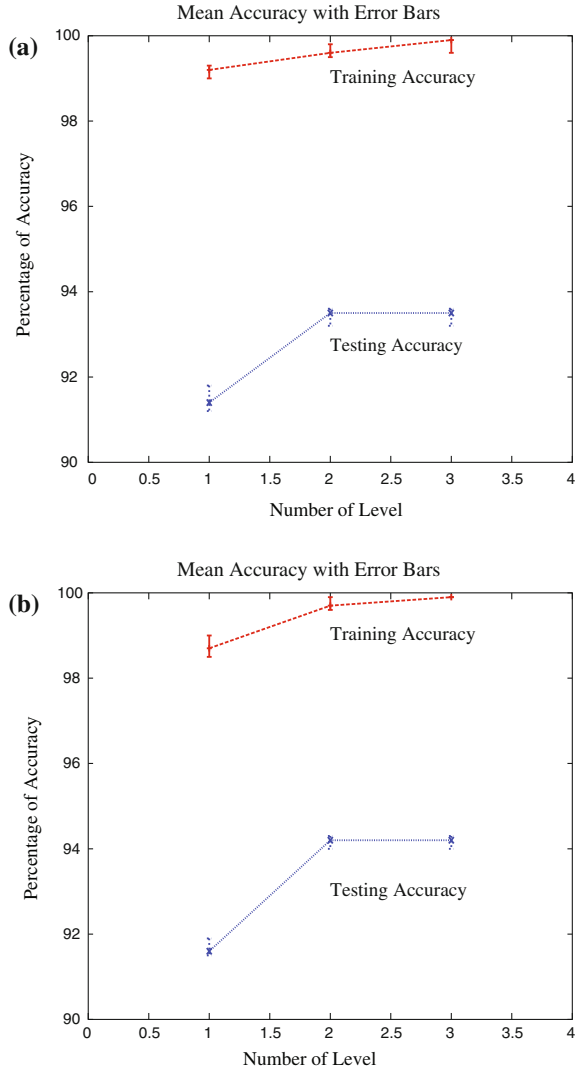
Table 2.3 compares the performance of the NNTree with C4.5, a popular decision tree algorithm [18], with respect to classification accuracy, total number of intermediate nodes, height of the tree, and classification time. For splice-junction database, the classification accuracy of the NNTree is higher than that of the C4.5, while the number of intermediate nodes, height of the tree, and classification time of the NNTree are significantly smaller than C4.5.

Finally, Table 2.4 compares the classification accuracy of the NNTree with that of different classification algorithms, namely, Bayesian [3], C4.5 [18], MLP [8, 10], and cellular automata (CA) [13]. The experimental results of Table 2.4 clearly establish the fact that the classification accuracy of the NNTree is higher than that of several other classification algorithms.

2.4 Identification of Protein Coding Region in DNA Sequence

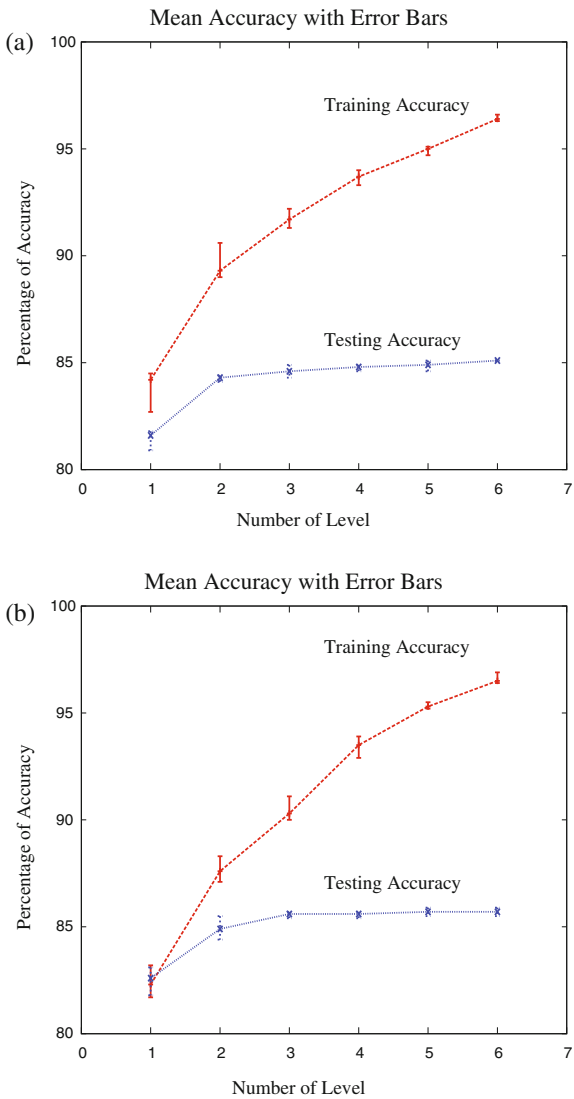
This section presents the application of the NNTree for finding protein coding (exon) regions in anonymous sequences of DNA. The performance of the NNTree is evaluated for few sequences and an analysis regarding the accuracy of the method is also presented.

Fig. 2.2 Performance of NNTree on splice-junction database for $\eta = 0.50$ and $\alpha = 0.70$. **a** $H_n = 10$; **b** $H_n = 15$



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 noncoding regions [1, 4, 5, 25]. 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 [6], who also have developed several benchmark features and databases for future experiments on this problem.

Fig. 2.3 Performance of the NNTree on splice-junction database for $\alpha = 0.70$. **a** $\eta = 0.90$ and $H_n = 10$; **b** $\eta = 0.80$ and $H_n = 15$



Previous research on automatic identification of protein coding regions has considered methods such as linear discriminants [5, 6] and neural networks [4, 25]. These systems have used measures such as codon frequencies, dicodon frequencies,

Table 2.5 Benchmark data sets proposed by Fickett and Tung

Data Set	Human 54	Human 108	Human 162
Training set—coding	20,456	7,086	3,512
Training set—noncoding	125,132	58,118	36,502
Training set total	145,588	65,204	40,014
Test set—coding	22,902	8,192	4,226
Test set—noncoding	122,138	57,032	35,602
Test set total	145,040	65,224	39,868

fractal dimensions, repetitive hexamers, and other features to identify exons in relatively short DNA sequences. The standard experimental study considers a limited window (that is, 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-noncoding.

2.4.1 Data and Method

The data used for this study are the human DNA data collected by Fickett and Tung [6]. 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 includes three different data sets. For the first data set, nonoverlapping 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 noncoding, or mixed, and the mixed sequences (that is, overlapping the exon–intron boundaries) are discarded. The data set also includes the reverse complement of every sequence. This means that one-half of the data is guaranteed to be from the nonsense 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 [6]. 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 data set, the data sets containing 108 and 162 bases have been used. The sizes of these data sets are shown in Table 2.5, which gives the number of nonoverlapping windows in each set. No information about reading frames is used in this study. Every window is either all-coding or all-noncoding, but the reading frame of each window is unknown. This choice of window length and experimental method follows that used by Fickett and Tung [6] and the problem here is what they defined as a protein coding region.

2.4.2 Feature Set

All of the features that have been used are derived from the 21 protein coding measures, which are proposed by Fickett and Tung [6]. A single coding measure is not necessarily the same thing as a single measure. Typically, a coding measure is a vector of measurements on a DNA subsequence.

2.4.2.1 Dicodon Measure

A dicodon is a subsequence of six consecutive nucleotides such as **TAGGAC**. The dicodon measure is the list of the 4,096 frequencies of every possible dicodon (six consecutive bases from the 4 letter alphabet). The dicodon frequency feature is a vector of the 4,096 dicodon frequencies across the input sequence, where the dicodon counts are accumulated only at locations whose starting point is a multiple of 3 (that is, starting at the 0th, 3rd, 6th, . . . nucleotides in the sequence). To convert the dicodon measure to a single number, the 4,096 dicodon frequencies are computed on each window and plugged into the hyperplane equation. This gives a single number that becomes the dicodon discriminant.

2.4.2.2 Hexamer-1 and Hexamer-2 Measures

The hexamer-1 and hexamer-2 measures are identical to dicodons, except that 1 and 2 offsets them. The hexamer-1 frequency feature is likewise a vector of 4,096 dicodon frequencies, except that the counts are accumulated at positions 1, 4, 7, . . .; the hexamer-2 frequency feature is defined analogously.

2.4.2.3 Open Reading Frame Measure

The open reading measure is simply the longest sequence of codons in the window that does not contain a stop codon. That is, the open reading frame feature is the length, in codons, of the longest sequence of codons (aligned with locations 0, 3, . . .) in the data string which does not contain a stop codon.

2.4.2.4 Run Measure

The run measure is a vector of length 14; for each nontrivial subset $S \subset \{\mathbf{A}, \mathbf{C}, \mathbf{T}, \mathbf{G}\}$, the run feature contains an entry that gives the length of the longest contiguous

subsequence having all entries from S . For example, if the entry of the run feature, which corresponds to $\{\mathbf{C}, \mathbf{G}\}$, is 4, then it means that the longest consecutive substring containing only \mathbf{C} and \mathbf{G} is of length 4. The run measure counts the number of repeats or runs of a single base or any set of bases from the set $(\mathbf{A}, \mathbf{C}, \mathbf{T}, \mathbf{G})$. Thus, it includes 14 nontrivial subsets of the four bases, and for each subset runs are counted separately.

2.4.2.5 Position Asymmetry Measure

The asymmetry feature is a vector of length four which measures, for each nucleotide \mathbf{A} , \mathbf{C} , \mathbf{G} , and \mathbf{T} , the extent to which the nucleotide is asymmetrically distributed over the three codon positions. The position asymmetry measure counts for each of the four bases, the frequency of the base in each of the three codon positions. Thus, $f(b, i)$ is the frequency of base b in position i and

$$\mu(b) = \sum_i \frac{f(b, i)}{3}. \quad (2.5)$$

Asymmetry is then defined as

$$\text{asymm}(b) = \sum_i (f(b, i) - \mu(b))^2. \quad (2.6)$$

2.4.2.6 Codon Usage Measure

The codon usage feature is a vector of the 64 codon frequencies. The codon usage measure is simply the frequencies of the 64 possible codons in the test window. The counts are accumulated only at locations 0, 3, ...

2.4.2.7 Diamino Acid Usage Measure

The diamino acid frequency is a vector of the 441 amino acid frequencies which are obtained by translating from the nucleotide sequence to an amino acid string (stop codons are treated as a 21st amino acid); like the dicodon frequency feature, counts are accumulated only at locations 0, 3, ...

2.4.2.8 Fourier Measure

Let $E(x, y)$ be the equality predicate that has value 1 if $x = y$ and 0 otherwise. The n th Fourier coefficient for a window W of length $2M$ is then defined as:

Table 2.6 Human DNA 54bp 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
6	86.5	82.9	16	86.7	82.4	32

$$F(n) = \sum_p \sum_m E(W_m, W_{m-p}) \exp\left(\frac{\Pi_{inp}}{M}\right) \quad (2.7)$$

where W_m represents the m th base in the window. The Fourier measure is then just $F(2M/2)$, $F(2M/3)$, \dots , $F(2M/9)$, which corresponds to the Fourier coefficients for periods 2 through 9.

After generating all protein coding measures, all the attributes in a data set are normalized to facilitate the NNTree learning. Suppose, the possible value range of an attribute \mathcal{A}_i is $(\mathcal{A}_{i,\min}, \mathcal{A}_{i,\max})$, and the real value that class element j takes at \mathcal{A}_i is \mathcal{A}_{ij} , then the normalized value of \mathcal{A}_{ij} is given as follows:

$$\overline{\mathcal{A}}_{ij} = \frac{\mathcal{A}_{ij} - \mathcal{A}_{i,\min}}{\mathcal{A}_{i,\max} - \mathcal{A}_{i,\min}}. \quad (2.8)$$

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

2.4.3 Experimental Results

In this subsection, the results of the NNTree for three Fickett and Tung's data sets are presented. Values are given for the percentage accuracy on both training and test set. Results of the NNTree on each of the data set are given in Tables 2.6, 2.7, 2.8, 2.9, 2.10, and 2.11. The mean accuracy of training and testing confirm that the evolved NNTree can generalize the data sets presented in Table 2.5 irrespective of the number of attributes, tuples, α , η , and H_n .

In case of Fickett and Tung database, for $L \leq 6$, the values of β_i for all possible nodes or locations of the NNTree are less than ε . So, all the nodes are intermediate or nonterminal nodes for $L \leq 6$. Hence, the NNTree has been grown by splitting all these nonterminal nodes. At $L = 7$, though the value of $\beta_i < \varepsilon$ for each nonterminal node, the training samples of two classes in each nonterminal node are highly correlated. So, at $L = 7$, an MLP cannot be found, corresponding to an intermediate node,

Table 2.7 Human DNA 54bp 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 2.8 Human DNA 108bp 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

Table 2.9 Human DNA 108bp 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 2.10 Human DNA 162bp 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 2.11 Human DNA 162bp 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

Fig. 2.4 Performance of NNTree on 54bp human DNA sequence for $\alpha = 0.70$ and $H_n = 10$. **a** $\eta = 0.50$; **b** $\eta = 0.70$

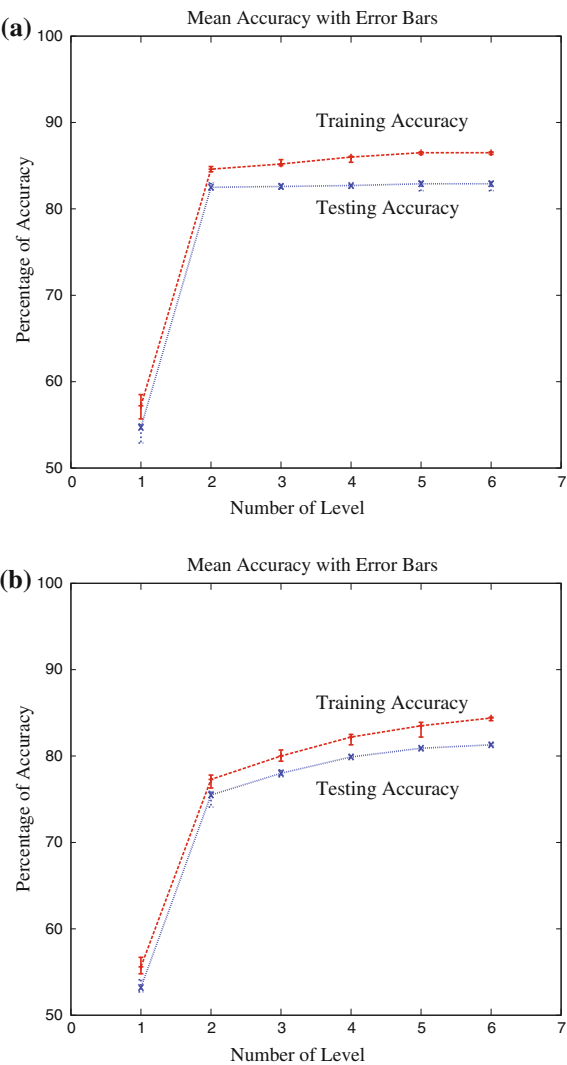
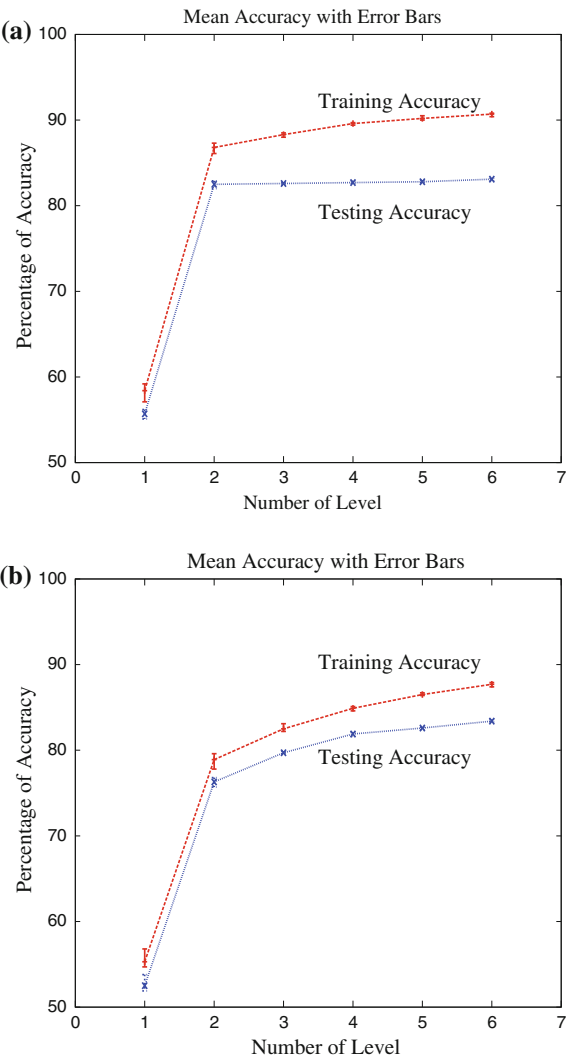


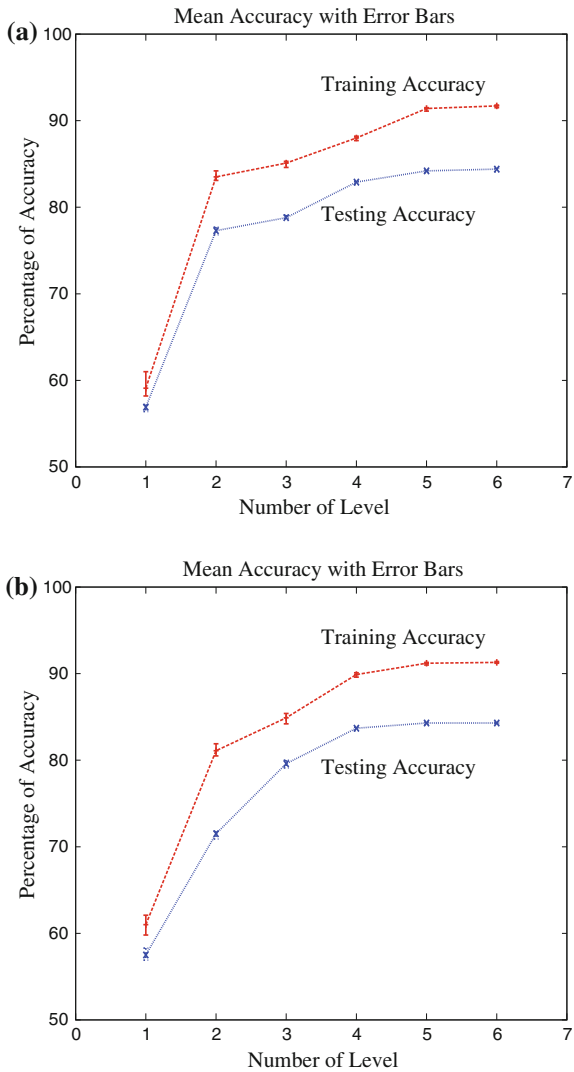
Fig. 2.5 Performance of NNTree on 108bp human DNA sequence for $\alpha = 0.70$ and $H_n = 10$. **a** $\eta = 0.50$; **b** $\eta = 0.70$



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 the current location.

Figures 2.4, 2.5, and 2.6 show the classification accuracy with error bar of the NNTree on different DNA sequences. All the results reported in Figs. 2.4, 2.5, and 2.6 establish the fact that the NNTree can generalize a DNA sequence data set irrespective of its sequence length. Also, the standard deviations of training and testing accuracy are very small.

Fig. 2.6 Performance of NNTree on 162bp human DNA sequence for $\eta = 0.50$ and $\alpha = 0.70$. **a** $H_n = 10$; **b** $H_n = 15$



Finally, Table 2.12 compares the classification accuracy of the NNTree with that of OC1 [15, 16], MLP, and other related algorithms. The OC1, proposed by Murty et al. [15, 16], 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 2.12 establish the fact that the classification accuracy of the NNTree is higher than that of existing algorithms. Also, the results reported here establish the fact that the NNTree can generalize a DNA data set irrespective of its sequence length.

Table 2.12 Classification accuracy for human DNA 54, 108, and 162bp

Algorithms	54bp	108bp	162bp
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

2.5 Conclusion and Discussion

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

The performance of the NNTree is evaluated through its applications in splice-junction and protein coding region identification. Experimental comparisons with other related algorithms provide better or comparable classification accuracy with significantly smaller trees and fast classification times. Extensive experimental results reported in this chapter confirm that the 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 the C4.5.

However, both DNA and protein sequences are nonnumeric variables as they are strings of nucleotides and amino acids, respectively. Hence, for most pattern recognition algorithms, they cannot be used as direct inputs. They, therefore, have to be encoded prior to input. To convert a DNA sequence into numeric values, two methods are reported in this chapter: one is distributed encoding method [17] and the other one is the feature extraction method proposed by Fickett and Tung [6]. In the next chapter, a new encoding method is reported to encode the DNA or protein sequences into numeric values for directly applying different pattern recognition algorithms on them.

References

1. Blaisdell BE (1983) A prevalent persistent global nonrandomness that distinguishes coding and non-coding eucaryotic nuclear dna sequence. *J Mol Evol* 19(2):122–133
2. Breathnach RJ, Mandel JL, Chambon P (1977) Ovalbumin gene is split in chicken DNA. *Nature* 270:314–319

3. Cheeseman P, Stutz J (1996) Bayesian classification (AutoClass): theory and results. In: Fayyad UM, Piatetsky-Shapiro G, Smith P, Uthurusamy R (eds) *Advances in knowledge discovery and data mining*. AAAI/MIT Press, Cambridge, pp 153–180
4. Farber R, Lapedes A, Sirotkin K (1992) Determination of eucaryotic protein coding regions using neural networks and information theory. *J Mol Biol* 226(2):471–479
5. Fickett J (1982) Recognition of protein coding regions in DNA sequences. *Nucleic Acids Res* 10(17):5303–5318
6. Fickett J, Tung CS (1992) Assessment of protein coding measures. *Nucleic Acids Res* 20(24):6441–6450
7. Guo H, Gelfand SB (1992) Classification trees with neural network feature extraction. *IEEE Trans Neural Networks* 3(6):923–933
8. Hertz J, Krogh A, Palmer RG (1991) *Introduction to the theory of neural computation*. Addison Wesley, Santa Fe institute studies in the sciences of complexity
9. Koza JR (1994) *Genetic programming-II: automatic discovery of reusable programs*. MIT Press, Cambridge, ISBN 0262111896
10. Lippmann R (1987) An introduction to computing with neural nets. *IEEE Acoust Speech Signal Process Mag* 4(2):4–22
11. Maji P (2008) Efficient design of neural network tree using a new splitting criterion. *Neuro-computing* 71(4–6):787–800
12. Maji P, Das C (2008) Pattern classification using NNtree: design and application for biological data set. *J Intell Syst* 17(1–3):51–71
13. Maji P, Shaw C, Ganguly N, Sikdar BK, Chaudhuri PP (2003) Theory and application of cellular automata for pattern classification. *Fundamenta Informaticae* 58:321–354
14. Michie D, Spiegelhalter DJ, Taylor CC (1994) *Machine learning, neural and statistical classification*. Ellis Horwood, Chichester
15. Murty SK, Kasif S, Salzberg S (1994) A system for identification of oblique decision trees. *J Artif Intell Res* 2(1):1–32
16. Murty SK, Kasif S, Salzberg S, Beigel R (1993) OC1: randomized induction of oblique decision trees. In: *Proceedings of the 11th national conference on artificial intelligence*, AAAI/MIT Press, pp 322–327
17. Qian N, Sejnowski TJ (1988) Predicting the secondary structure of globular proteins using neural network models. *J Mol Biol* 202(4):865–884
18. Quinlan JR (1993) *C4.5: programs for machine learning*. Morgan Kaufmann, San Francisco
19. Schmitz GP, Aldrich C, Gouws FS (1999) ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. *IEEE Trans Neural Networks* 10(6):1392–1401
20. Sethi IK (1990) Entropy nets: from decision trees to neural networks. *Proc IEEE* 78(10):1605–1613
21. Sethi IK, Yoo JH (1997) Structure-driven induction of decision tree classifiers through neural learning. *Pattern Recogn* 30(11):1893–1904
22. Song HH, Lee SW (1998) A self-organizing neural network tree for large-set pattern classification. *IEEE Trans Neural Networks* 9(6):369–380
23. Tay ALP, Zurada JM, Wong LP, Xu J (2007) The hierarchical fast learning artificial neural network (hieflann): an autonomous platform for hierarchical neural network construction. *IEEE Trans Neural Networks* 18(6):1645–1657
24. Tsukimoto H (2000) Extracting rules from trained neural networks. *IEEE Trans Neural Networks* 11(2):377–389
25. Uberbacher E, Mural R (1991) Locating protein-coding regions in human dna sequences by a multiple sensor-neural network approach. *Proc Nat Acad Sci USA* 88(24):11,261–11,265
26. Wilamowski BM, Yu H (2010) Neural network learning without backpropagation. *IEEE Trans Neural Networks* 21(11):1793–1803
27. Zhao QF (2000) *Neural network tree: integration of symbolic and nonsymbolic approaches*. Technical Report of IEICE

28. Zhao QF (2001) Evolutionary design of neural network tree-integration of decision tree, neural network and GA. In: Proceedings of the IEEE congress on evolutionary computation, pp. 240–244
29. Zhao QF (2001) Training and retraining of neural network trees. In: Proceedings of the INNS IEEE international joint conference on neural networks, pp. 726–731
30. Zhou ZH, Chen ZQ (2002) Hybrid decision tree. *Knowl-Based Syst* 15(8):515–528

<http://www.springer.com/978-3-319-05629-6>

Scalable Pattern Recognition Algorithms
Applications in Computational Biology and
Bioinformatics

Maji, P.; Paul, S.

2014, XXII, 304 p. 55 illus., 10 illus. in color., Hardcover

ISBN: 978-3-319-05629-6