Back-propagation and counter-propagation neural networks for phylogenetic classification of ribosomal RNA sequences

Cathy Wu* and Sailaja Shivakumar

Department of Epidemiology/Biomathematics, The University of Texas Health Center at Tyler, Tyler, TX 75710, USA

Received May 25, 1994; Revised and Accepted September 7, 1994

ABSTRACT

A neural network system has been developed for rapid and accurate classification of ribosomal RNA sequences according to phylogenetic relationship. The molecular sequences are encoded into neural input vectors using an n-gram hashing method. A SVD (singular value decomposition) method is used to compress and reduce the size of long and sparse n-gram input vectors. The neural networks used are three-layered, feed-forward networks that employ supervised learning paradigms, including the back-propagation algorithm and a modified counter-propagation algorithm. A pedagogical pattern selection strategy is used to reduce the training time. After trained with ribosomal RNA sequences of the RDP (Ribosomal Database Project) database, the system can classify query sequences into more than one hundred phylogenetic classes with a 100% accuracy at a rate of less than 0.3 CPU second per sequence on a workstation. When compared to other sequence similarity search methods, including Similarity Rank, Blast and Fasta, the neural network method has a higher classification accuracy at a speed of about an order of magnitude faster. The software tool will be made available to the biology community, and the system may be extended into a gene identification system for classifying indiscriminately sequenced DNA fragments.

INTRODUCTION

The accelerated growth of the molecular sequencing data generated by the Human Genome Project necessitates the development of advanced computational tools for rapid annotation and easy organization of genomic sequences. Database similarity searches presently represent the most common approach to decipher sequence information. A sequence classification method can be used as an alternative approach to the database search/organization problem with several advantages: (1) speed, because the search time grows linearly with the number of

sequence classes (families), instead of the number of sequence entries; (2) sensitivity, because the search is based on information of a homologous family, instead of any sequence alone; and (3) automated family assignment.

Several sequence classification/clustering systems have been devised. The system may be based on 'unsupervised' algorithms, such as a multivariate statistical technique [1], an algorithm that uses a binary similarity comparison followed by an unsupervised learning procedure [2], and Kohonen's self-organized neural map [3]. Or it may be based on a 'supervised' learning, such as back-propagation neural network [4]. A supervised learning paradigm is needed if new sequences are to be classified according to a predefined family relationship in the database, such as superfamilies of the Protein Identification Resource database [5] or phylogenetic classes of the Ribosomal Database Project (RDP) database [6].

There are many different types of classification learning systems, broadly categorized into parametric and nonparametric models [7]. The nonparametric classifiers do not require a priori recognition of certain specific patterns and make few assumptions in characterizing the samples. Some examples include the nearest-neighbor families of statistical methods and neural network algorithms, such as back-propagation (BP) [8] and counter-propagation (CP) [9].

As a technique for computational analysis, neural network technology has been applied to many studies involving sequence data analysis [10]. BP networks currently represent the most popular learning paradigm, and have been used to predict protein secondary and tertiary structures [11, 12, 13], to distinguish encoding regions from non-coding sequences [14, 15], to predict bacterial promoter sequences [16], and to classify molecular sequences [4, 17]. CP, also a supervised learning algorithm, is closely related to the nearest-neighbor classifier. The network essentially functions as a nearest-match lookup table. CP network has an interesting capability to extract the statistical properties of the input data, and can usually be trained very rapidly.

This study extends the designs of our BP protein classification system into a ribosomal RNA (rRNA) sequence classification system and introduces several enhancements and new designs.

*To whom correspondence should be addressed

The RDP database was chosen for the following reasons. First, a rRNA sequence classification system is useful: it can be used to help organize the RDP database and annotate unknown rRNA sequences. Second, the RDP database is one of the very few molecular sequence databases that organize their entries according to family relationship. Moreover, the placement of each individual entry is manually checked to ensure accuracy. This makes it an ideal data set for training the neural network. Third, RDP is a nucleic acid database. Although its size is relatively small, it can be used as a pilot system for the eventual development of a full-scale system for nucleic acid classification and gene identification.

This paper describes the rRNA sequence classification system that employs BP algorithm and a modified CP algorithm designed for this application. The system performance is compared with the existing methods available for phylogenetic classification of rRNA sequences, including Similarity Rank [6], Blast [18], and Fasta [19]. The paper also provides a detailed analysis of the neural network results and discusses future system enhancement and application.

MATERIALS AND METHODS

Data sets

The neural network system is designed to classify new (unknown) sequences into predefined (known) classes. In other words, it maps molecular sequences (input) into sequence classes (output) (Figure 1). In this study, the system is trained with the RDP database (Release 3.0, August 1, 1993) and maps rRNA sequences into phylogenetic classes [20]. In RDP, each rRNA sequence entry is fully aligned with its relatives to identify their similarities. The percentage of similarity is then converted to an evolutionary distance and used to place the entry on a phylogenetic tree. The tree is an assembly of partially overlapping subtrees that are inferred using maximum likelihood analysis [21].

The RDP database has three collections, LSU for large subunit rRNA, SSU_Euk for eukaryotic small subunit rRNA, and SSU_Prok for prokaryotic small subunit rRNA. Both the SSU_Prok and LSU collections are provided with phylogenetic trees and can be used as training sets for the neural networks. A total of 109 and nine phylogenetic classes, respectively, are derived directly from the trees, using the SSU_Prok.phylo and LSU.phylo files in the RDP database for 1379 SSU_Prok and 72 LSU sequences. In the classification scheme, no one class at the non-leaf level has more than 30 sequence entries (i.e., for any class that has more than 30 entries at one given level, it is further subdivided into classes at its sublevel, unless it has reached the leaf level). The number of 30 is chosen to reduce the number of sequence alignments required to compare an unknown entry against all other entries of the same class. Table 1 gives a partial list of the phylogenetic classes of SSU_Prok sequences. The classes may be at different levels in the tree. Using the numbering system in the RDP database, we have, for example, classes 1.2, 1.1.1, 1.1.3.1 and 2.16.1.1.1 (i.e., levels 2 to 5, with level 5 being the leaf level). Note that a class in the leaf node may have more than 30 entries. For example, there are 80 entries in the Mycobacteria class (class 69, 2.16.1.1.1, Table 1). The finest level of classification would be to use one neural network class to represent every single leaf node on the phylogenetic tree. This yields a total of 119 classes for SSU_Prok sequences. (The results reported in this paper are based on 109 classes, but similar results were obtained using 119 classes).

Among all 1379 SSU_Prok entries, approximately two thirds (958 sequences) are used for training, the remaining 421 sequences, selected from every third entries of all classes, are used for prediction. Therefore, the training and prediction sequences are in completely disjointed sets. This is essential for providing a meaningful evaluation of the neural network system performance. For the LSU data set, 51 of the 72 sequences are

Figure 1. A neural network system for RNA sequence classification. The sequences are first converted by an n-gram sequence encoding method into neural net input vectors. Long n-gram input vectors can be compressed by a SVD method to reduce vector size (dimension). The neural networks then maps the sequence vectors into predefined classes according to sequence information embedded in the neural interconnections after network training. The neural networks used are three-layered, feed-forward networks that employ back-propagation or counter-propagation learning algorithms.

used for training, and the remaining 21 sequences, selected again from every third entries of each class, are used for prediction.

Sequence encoding schema

The sequence encoding schema is used to convert molecular sequences (character strings) into input vectors (numbers) of the neural network classifier (Figure 1). An ideal encoding scheme should satisfy the basic coding assumption so that similar sequences are represented by 'close' vectors.

N-gram method. We have been using an n-gram hashing function [4, 17] that extracts and counts the occurrences of patterns of n consecutive residues (i.e., a sliding window of size n) from a sequence string. The counts of the n-gram patterns from each encoding method are scaled to fall between 0 and 1 and used as input vectors for the neural network, with each unit of the vector representing an n-gram pattern. The size of the input vector for each n-gram extraction is m^n , where m is the size of the alphabet. The original sequence string is represented by three different alphabet sets in the encoding: set N = {A, U, G, C}, set R = {R (purines A, G), Y (pyrimidines U, C)}, and set S = {S (strong hydrogen binding G, C), W (weak binding A, U)}.

SVD (singular-value decomposition) method. The major drawback of the n-gram method is that the size of the input vector for a large n-gram size (i.e., long oligomers) is large, which makes the network training difficult and time-consuming. SVD is one approach to reduce the size of n-gram vectors and to extract the semantics of the n-gram patterns [22]. The method was adopted from the Latent Semantic Indexing (LSI) analysis [23] used in the field of information retrieval and information filtering. In SVD, a large matrix is decomposed into a set of k orthogonal factors from which the original matrix can be approximated by linear combination. In the present study, a term-by-sequence matrix is used to represent the n-gram patterns (terms) of different RNA sequences. The large and sparse matrix is decomposed into singular triplets, i.e., the singular values, and the corresponding left and right singular vectors. With k being the chosen number of dimensions in the reduced model, the right singular vectors corresponding to the k-largest singular values are then used as the input vectors for neural networks to represent the n-gram patterns.

Back-propagation (BP) neural networks

The BP neural networks used in this study are three-layered, feed-forward networks (Figure 1) [4]. The size of the input layer (i.e., number of input units) is dictated by the sequence encoding schema chosen. The size is m^n with n-gram encoding; and the size is the reduced dimension (k), if the n-gram vector is compressed by SVD. The output layer size is determined by the number of classes represented in the network, with each output unit representing one phylogenetic class. The hidden size is determined heuristically, usually a number between input and output sizes. Several preliminary studies were conducted to determine the optimum values for various network parameters. In this study, a hidden size of 100 is chosen, and the networks are trained using weight matrices initialized with random weights ranging from -0.3 to 0.3. Other network parameters include the learning factor of 0.3, momentum term of 0.2, a constant bias term of -1.0, and error threshold of 0.01.

Pedagogical pattern selection strategy

One major problem frequently encountered when using BP neural networks is the long training time. In BP, all patterns in the training set are usually presented equally to the neural network (i.e., uniform presentation). An alternative pattern selection strategy, a pedagogical EDR (error-dependent repetition) method, is developed recently [24] to favor the selection of patterns producing high error values to the disadvantage of the patterns already mastered by the network. In this study, a modified EDR is used, in which training proceeds as described [24] for the first 400 iterations, followed by another 100 iterations using all training patterns with uniform presentation.

Counter-propagation (CP) neural networks

A modified CP algorithm with supervised LVQ (learning vector quantizer) and dynamic node allocation is used (The algorithm will be published in detail elsewhere and can be obtained directly from wu@jason.uthct.edu). The forward-only CP network has three layers: an input layer, a Kohonen layer and a Grossberg outstar conditioning layer. As in BP network, the sizes of the input and output (Grossberg) layers are the size of the input vector and the number of output classes, respectively; and the size of the hidden (Kohonen) layer needs to be determined like a parameter. As a pattern classifier, a CP network uses the Kohonen layer to determine winning units for the input patterns, and uses the Grossberg layer to map the winners into their classes. The Kohonen layer essentially is a LVQ network, introduced by Kohonen, that performs nearest-neighbor classification [25]. Its training involves two steps, to select a winner based on the Euclidian distance between the input and weight vectors, and to update the winner's weight vector.

In our modified algorithm, the Kohonen layer starts with zero nodes, then nodes are added dynamically when any given training pattern fails to be assigned to the Kohonen units that represent its class. The weight vector of a newly added node is initialized to be the input vector of the wrong (untrainable) pattern. A supervised learning that selects two winners with punishment mechanism is used for weight update: if the first winner is correct, update with positive weights, if incorrect, punish the winner with negative weights and find the second winner; if the second winner is correct, update with positive weights, if incorrect, allocate a new unit. The Kohonen learning rate for weight adjustment is 0.2.

System implementation

The system software has three components: a preprocessor to create from input sequence files the training and prediction patterns, a neural network program to classify input patterns, and a postprocessor to summarize classification results. The preprocessor has two programs, one for n-gram extraction, the other for SVD computation. The SVD program employs a single-vector Lanczos method [26] and is part of the SVDPACKC package (available from Netlib). Two separate neural network programs are used for BP and CP algorithms, respectively. All programs have been coded in C language and implemented on the Cray Y-MP8/864 supercomputer of the Center for High Performance Computing of the University of Texas System. The programs have also been ported to several other UNIX computer platforms, including a DEC alpha workstation, a Sun SPARC Center, and a Silicon Graphics workstation.

Evaluation mechanism

The system performance is measured by its speed (CPU time) and predictive accuracy. The predictive accuracy is evaluated using target classes derived from the phylogenetic tree directly, and compared with the result of the Similarity Rank program from the RDP e-mail server [6], as well as Blast and Fasta database search programs. The Similarity Rank program (kindly provided by Niels Larsen, University of Illinois) searches query sequences against the database sequences by comparing oligomers (currently heptamers for SSU_Prok database search) and lists the database sequences that are most similar to query sequences according to their similarity scores. The similarity score is expressed as the number of shared oligomers (between the query sequence and database sequence) divided by the number of unique oligomers in either the query sequence or the database sequence, whichever is lower.

The neural network predictive accuracy is expressed with three terms: the total number of correct patterns (true positives), the total number of incorrect patterns (false positives), and the total number of unidentified patterns (negatives). A sequence entry is considered to be accurately classified if one of its top-ranking classes matches the target class with a classification score above the threshold (i.e., the cut-off value). The classification score ranges from 1.0 for perfect match to 0.0 for no match. In this study, results for correct patterns are reported using two measures, the first-fit (the class with the highest score matches the target) and the first three-fits (one of the classes with the three highest scores matches the target) at a cut-off classification score of 0.1 or 0.5. The low threshold is used to identify as many true positives as possible, whereas the high threshold is used to avoid false positives.

RESULTS

Comparisons of n-gram and SVD sequence encoding methods

The classification results of BP networks using different n-gram and SVD methods for the SSU_Prok data set are shown in Table 2. The n-gram encoding methods are represented by a two-letter code: the first character is a letter designating the alphabet set

N, R or S; the second character is a digit representing the order of the n-gram (i.e., the length of the oligomers). Among several tested (some not shown in Table 2), the best n-gram encoding method is nrs5, which concatenates n5, r5 and s5, penta-gram patterns of all three alphabet sets. The vector has 1088 units, which is the sum of the three vector sizes (i.e., $4^5 + 2^5 + 2^5$). With the nrs5 method, the predictive accuracy is 96.44% counting only first-fit, or 99.05% counting first three-fits at a threshold of 0.1.

With the SVD method, various n-gram vectors are reduced to smaller dimensions, designated by the number following the underscore in the code (e.g., n8_100 represents an input vector of octa-gram nucleic acid patterns with a reduced dimension of 100). Three observations were made from the SVD results (Table 2): (1) among the three alphabet sets, the nucleic acid alphabet set (N) gave a much higher predictive accuracy than did the purine-pyrimidine set (R) and the strong-weak hydrogen binding set (S); (2) among different n-gram sizes, octa-gram appears to be optimal (i.e., the ranking from highest to lowest predictive accuracies is $n8 > n9 > n78 > n67 > n7 > n6$); and (3) among different reduced dimensions, the 100-dimensional representation appears to be optimal (i.e., with a ranking of $100 > 150 > 120 > 80$). The best result, given by n8_100, shows a 97.39% predictive accuracy in first-fit and 99.52% accuracy in first three-fits at a threshold of 0.1.

The results of the SVD encoding show that classification accuracy can be improved even with a much smaller neural network (e.g., comparing nrs5, which has 1088 input units, with n8_100, which has 100 input units). Significantly, the information embedded in n8 n-gram patterns alone is sufficient for a close to 100% classification. This information would be very difficult to capture without the SVD reduction, however. It would be difficult to input the n8 n-gram vectors directly to the neural network because the network would be too large (with 65,536 input units) and the vector too sparse (too many zeros) for the neural network to be trained effectively.

Pattern selection strategy

The effects of different pattern selection strategies on the system performance were studied. With the uniform strategy, the number

Table 1. A partial list of phylogenetic classes of SSU_Prok ribosomal RNA sequences used in neural network training and prediction

Class number	Number of sequences	Class ID number ^a	Class name ^a
1	8	1.1.1	Archaea.Euryarchaeota.Methanococcales
2	19	1.1.2	Archaea.Euryarchaeota.Methanobacteriales
3	12	1.1.3.1	Archaea.Euryarchaeota.Methanomicrobiales
4	22	1.1.3.2	Archaea.Euryarchaeota.Methanosarcinales
..	..		
10	17	1.2	Archaea.Crenarchaeota
11	1	2.1	Bacteria.Thermophilic__Oxygen__Reducers
12	8	2.2	Bacteria.Thermatogales
..	..		
68	18	2.15	Bacteria.Fusobacteria__And__Relatives
69	80	2.16.1.1.1	Bacteria.Gram__Positive__Phylum.High__G+C__Subdiv.
..	..	Mycobacteria	
..	..		
107	14	2.16.7.2.1	Bacteria.Gram__Positive__Phylum.Lactobacillus__Subgroup1
108	11	2.16.7.2.2	Bacteria.Gram__Positive__Phylum.Lactobacillus__Subgroup2
109	26	2.16.7.2.3	Bacteria.Gram__Positive__Phylum.Lactobacillus__Subgroup3

^aThe class ID number and name are directly taken from SSU_Prok.phylo, the phylogenetic listing file in RDP database, release 3.0.

of patterns presented is the number of training patterns times the number of iterations (i.e., $383,200 = 958 \times 400$). In the EDR, only patterns of high errors are selected in most iterations, therefore, a smaller number of patterns are presented which results to a training of approximately one order of magnitude faster. However, the predictive accuracy is also reduced. The best result is obtained with the modified EDR, where EDR is used for the first 400 iterations and uniform presentation is used for the next 100 iterations. With the modified EDR, a similar level of accuracy given by the uniform strategy is achieved with only one third of the training time.

Comparisons of the back-propagation, nearest-neighbor and counter-propagation classifiers

Table 3 shows the comparisons of the three classifiers on network architecture, computational time and predictive accuracy for the

SSU_Prok data set. The BP network is the smallest among all three classifiers, because unlike CP, it does not require the hidden layer to have at least as many units as the output layer.

While the nearest-neighbor classifier requires little time to train (i.e., converged in one iteration after initialization with input vectors), the prediction (input-output mapping) takes more time because the network is large. Since the prediction time on a nearest-neighbor classifier is proportional to the number of training patterns (i.e., the hidden size), this classifier is not suitable for application to the classification problem of large and growing databases.

The CP networks train rapidly, which is usually one to two orders of magnitude faster than BP networks with the new pattern selection strategy. For all the testing conducted (not all shown), the total CPU time required to train the neural networks ranges between less than 15 minutes to 2.5 hours on a DEC workstation,

Table 2. The effects of encoding methods on the predictive accuracy of back-propagation networks for SSU_Prok data set after training for 400 iterations with uniform pattern presentation

Encoding method ^a	Size of N-gram vector	Size of input vector	Pattern trained(%)	Predictive accuracy (%)		Incor. ²	Uniden. ^b
				First-fig	Three-fits		
N-gram							
n5	1,024	1024	99.06	96.44	98.09	1.43	0.48
nrs5	1,088	1088	99.16	96.44	99.05	0.71	0.24
SVD							
nrs5__100	1,088	100	100.00	93.57	97.39	2.61	0.00
nrs6__100	4,224	100	99.48	96.20	98.34	1.43	0.24
nrs8__100	66,048	100	99.79	96.20	98.10	1.90	0.00
n6__100	4,096	100	98.43	93.82	96.44	3.33	0.24
n67__100	20,480	100	99.58	95.72	99.26	0.71	0.00
n7__100	16,384	100	99.90	94.77	99.29	0.71	0.00
n78__100	81,920	100	98.75	96.91	98.81	1.19	0.00
n8__100	65,536	100	99.06	97.39	99.52	0.48	0.00
n9__100	262,144	100	99.58	97.15	99.52	0.48	0.00
n8__150	65,536	150	99.58	96.67	99.05	0.96	0.00
n8__120	65,536	120	98.23	96.67	97.86	2.14	0.00
n8__80	65,536	80	98.23	94.77	98.10	1.66	0.24
rs10__100	2,048	100	99.27	94.54	95.96	3.80	0.24
rs12__100	8,192	100	99.48	95.49	97.39	2.61	0.00
n8rs12__100	73,728	100	100.00	95.96	99.29	0.71	0.00
rs15__100	65,536	100	99.58	96.44	98.57	1.43	0.00

^aPlease see text for code representation.

^bIncor. = Incorrect patterns, Uniden. = Unidentified patterns.

Table 3. Comparisons of back-propagation, nearest-neighbor and counter-propagation algorithms on the network architecture, computational time and predictive accuracy for SSU_Prok data set

Classification algorithm ^a	Network configuration	Training time ^b	Prediction time ^b	Patterns (%)	Predictive trained First-fit	accuracy(%) Three-fits
nrs5 Encoding						
BP	1088×100×109	8321	14.7	100.00	96.20	99.29
kNN	1088×958×109	221	97.2	100.00	96.20	99.29
CP	1088×178×109	590	18.8	100.00	94.54	98.34
Average (BP, CP)					96.20	99.29
n8__100 Encoding						
BP	100×100×109	2424	1.9	100.00	97.86	99.29
kNN	100×958×109	19	8.5	100.00	95.01	99.29
CP	100×193×109	87	3.4	100.00	95.25	99.52
Average (BP, CP)					98.34	100.00

^aBP: Back-Propagation; kNN: k Nearest-Neighbor; CP: Counter-Propagation; Average: classification results obtained from averaging the BP and CP classification scores.

^bBoth training time and prediction time are measured in CPU seconds on a DEC alpha workstation. The time reported is for neural network training only (i.e., does not include preprocessing time).

and less than 15 minutes on the Cray Y-MP. With the n8__100 encoding method, it takes a total of 730 CPU seconds on the DEC workstation for preprocessing the 958 training patterns, with 134 seconds for n-gram encoding and 596 seconds for SVD computation. Since the network training time for BP and CP networks using the n8__100 method is 2424 and 87 DEC seconds, respectively (Table 3), the total training time is 53 DEC minutes for BP network and 14 DEC minutes for CP network.

Both BP and CP networks give speedy prediction. For all testing conducted, the total CPU time required to classify an unknown sequence on a DEC alpha workstation averages between 0.12 and 0.26 CPU seconds, including preprocessing and postprocessing time. The n8__100 encoding of 421 prediction patterns takes a total of 105 DEC seconds, with 59 second for n-gram processing and 46 seconds for SVD computation. The actual network prediction and postprocessing takes very little time, i.e., 1.9 and 3.4 seconds for BP and CP networks, respectively (Table 3). Therefore, the total time required to predict all 421 sequences using both BP and CP methods is roughly 110 seconds, which averages to 0.26 DEC seconds per query sequence. It is worth noting that, the n-gram and SVD computation time required for preprocessing query sequences is determined by the size of the n-gram vectors, and is independent

of the number of training sequences. Thus, the speed of neural network classification method would not be constrained by the rapid growth of databases.

When comparing predictive accuracy of the three classifiers, it is found that (1) among all encoding methods tested (not all shown), n8__100 is the best for all three classification networks, (2) if only first-fit is considered, the BP gives best prediction results; with three-fits, however, the results are comparable among all three classifiers. Complementary results are obtained from different learning algorithms which, when combined, lead to improved accuracy. When classification scores of n8__100 encoding method from the BP and CP networks are averaged, a 100% predictive accuracy is achieved counting first three-fits, and a more than 98% accuracy is obtained counting only first-fit at the threshold of 0.1 (Table 3).

Comparisons of neural network (NN), Similarity Rank (SR), Blast and Fasta methods

Table 4 shows the comparisons of the four methods using the SSU__Prok data set of 958 training and 421 prediction sequences. The NN result used for the comparison is the average result of the BP and CP networks with n8__100 encoding algorithm. The SR, Blast and Fasta results are obtained by searching the query

Table 4. Comparisons of neural network, similarity rank, Blast and Fasta methods on the classification speed and predictive accuracy for SSU__Prok data set

Method	Classification speed ^a		Predictive accuracy (%) ^b				Incorrect
	Total	Single	First-fit	Second-fit	Third-fit	Three-fits	
Neural Network ^c		110	0.26	98.34	1.42	0.24	100.00
Similarity rank	492		1.17	96.67	2.61	0.24	99.52
Blast	1122		2.67	90.50	4.04	3.09	97.63
Fasta	1617		3.84	94.77	2.85	0.95	98.57

^aThe classification speed is measured in CPU seconds on a DEC alpha workstation. The time reported for neural network includes preprocessing and network classification time. The 'total' is total time for classifying all 421 query sequences, 'single' time is the averaged classification time per sequence.

^bThe predictive accuracy is measured by counting the first three-fits. The 'three-fits' is the total of first, second and third-fits, 'incorrect' is for patterns not correctly classified within the first three-fits.

^cThe neural network result is the average result of the BP and CP networks using the n8__100 encoding algorithm, as shown in Table 3.

Table 5. Partial classification results of the neural network system and Similarity Rank program

Query sequence	Target class	Neural network Result	Score	Similarity rank Result	Score
Her.aurant	2.4.0.0.0	First-Fit	0.67	Second-Fit	0.44
D.radiodur	2.4.0.0.0	First-Fit	0.40	Incorrect ^a	---
Osp.pusill	2.12.1.2.0	First-Fit	0.54	Second-Fit	0.58
Spg.capsul	2.12.1.6.0	First-Fit	0.98	Second-Fit	0.69
Aqb.spirit	2.12.1.9.0	First-Fit	0.73	Second-Fit	0.72
Rps.marina	2.12.1.9.0	First-Fit	0.58	Second-Fit	0.65
Tms.thyasi	2.12.3.4.0	Second-Fit	0.61	First-Fit	0.64
Pas.BTax05	2.12.3.7.1	Third-Fit	0.46	First-Fit	0.79
H.haemolyt	2.12.3.7.1	Second-Fit	0.61	Second-Fit	0.76
Pas.pneumo	2.12.3.7.2	Second-Fit	0.56	First-Fit	0.78
H.sp202	2.12.3.7.3	Second-Fit	0.59	Second-Fit	0.82
Dmn.tiedje	2.12.4.1.1	First-Fit	0.92	Second-Fit	0.51
Cynd.cal__C	2.13.2.0.0	First-Fit	0.86	Second-Fit	0.54
Cynp.par__C	2.13.2.0.0	First-Fit	0.80	Second-Fit	0.48
Glb.violac	2.13.3.0.0	First-Fit	0.57	Third-Fit	0.45
Scp.erythr	2.16.1.1.3	First-Fit	0.97	Second-Fit	0.63
str.ALV	2.16.3.0.0	Second-Fit	0.54	Incorrect	---
C.thysu39E	2.16.5.1.9	Second-Fit	0.58	First-Fit	0.67

^aA sequence pattern is considered to be incorrectly classified if none of the top three classes matched the target class.

sequences (prediction set) against the database sequences (training set). The NN prediction results compare favorably with other search methods at a faster speed (Table 4). When considering the first-fit patterns only, the predictive accuracies are 98.3, 96.7, 94.8, and 90.5%, respectively, for the NN, SR, Fasta and Blast methods. The same ranking is obtained counting the first three-fits, with predictive accuracy ranging from 100% (NN) to 97.6% (Blast) (Table 4). Among the 421 query sequences, two are missed by all four methods, counting first-fit only. One of them (H.sp202) is classified by all four methods into class 2.12.3.7.1, instead of class 2.12.3.7.3 as in the RDP phylogenetic tree (although class 2.12.3.7.3 is a close second-fit in all methods).

Inspection of the seven non-first-fit patterns of the NN result shows only one is not also missed by at least one other method. This pattern (C.thysu39E), correctly classified as a second-fit by the NN, is a fragmentary sequence and belongs to a class containing only two training patterns. Examination of the non-first-fit patterns of other three methods shows that four patterns are missed by SR only (i.e., they are correctly classified by all other three methods), four patterns are missed by Fasta only, and 23 patterns by Blast only. Since SR is the current method of choice for sequence similarity search in the Ribosomal Database Project, further comparisons are made for all non-first-fit patterns from either NN or SR prediction (Table 5).

Presently, with the SSU_Prok data set (release 3.0, a total of 1379 entries), the classification speed of the NN system is about an order of magnitude faster than other methods compared. The speed, including both sequence preprocessing time and network classification time, is about five times faster than SR and approximately ten to fifteen times faster than Blast and Fasta (Table 4). Unlike other three methods, the NN classification time will not increase with the accumulation of database entries. A recent test using SSU_Prok database in RDP release 4.0 (June 18, 1994, 2251 total entries) shows that, while the prediction time remains the same with NN method, the time using SR, Blast or Fasta methods has increased by more than 60%, directly proportional to the number of entries increased.

Analysis of neural network results

Two observations are made from the neural network classification results: (1) the majority of first-fit classes (i.e., 83.6%) have classification scores of greater than 0.9, indicating high degree of similarity between the query sequences and training sequences, and (2) for query sequences whose correct classes appear as second or third-fits, their first and second-fit classes are also closely located on the phylogenetic tree.

Effect of sequence length and family size. It has been shown [17] that most protein sequences misclassified by the neural networks are fragmentary sequences, sequences of small families and/or distantly related sequences. These factors are also evaluated for the rRNA sequences. The SSU_Prok prediction set has sequences ranging from 200 to 1555 bases long. The majority (about 85%) of the sequences are full-length with more than 1300 bases, while 19 (out of 421) sequences are fragments of shorter than 500 bases. Eight of the 421 sequences belong to classes of less than three members. The results show that all 19 fragmentary sequences are correctly classified as first-fits, including the shortest sequence of only 200 residues long (env.NH49-8). Among the eight sequences of small families, all are correctly classified as first-fits, but one pattern (C.thysu39E of 868 bases

long) as a second-fit. Therefore, although the sequence length or family size alone may not be a sufficient factor for misclassification, a fragmentary sequence in a single-membered or double-membered class is more likely to be incorrectly classified or classified with a lower classification score, as seen in C.thysu39E.

System specificity. To evaluate the specificity of the neural network system (i.e., ability to distinguish true from false positives), three negative data sets of different distances to the training set are used for prediction. Data set one, for true negatives of remote relationship, consists of 958 SSU_Prok sequences for training and 236 SSU_Euk sequences for prediction. Data set two, for closely related negatives, has 943 training sequences in 107 classes (i.e., 15 training sequences for classes 2.16.7.1.2 and 2.16.7.1.9 are removed from the 958 SSU_Prok sequences) and six prediction sequences for the now-undefined classes of 2.16.7.1.2 and 2.16.7.1.9. Data set three, for more distantly related negatives, has 946 training sequences in 107 classes (i.e., 12 sequences for classes 1.1.1 and 2.2 are removed) and four prediction sequences of the undefined classes of 1.1.1 and 2.2.

Using the average results of the BP and CP networks at a high threshold of 0.5, all eukaryotic sequences are correctly classified as 'unidentified', as they are all true negatives. Only the last (236th) sequence is predicted with a classification score of 1.0 to class 2.12.3.6.3, the class which *E.coli* sequence belongs to. (p.s. The SSU_Euk data set has 235 eukaryotic sequences, and the last sequence is the *E.coli* sequence, used as a reference for the data set). At the same threshold of 0.5, the prediction of the 421 SSU_Prok sequence yields a result of 409 first-fits (97.15%), six second-fits, one incorrect (false positive) pattern (pattern number 178) and five unidentified (false negative) patterns. All six negative patterns in data set two are classified into classes closely located on the phylogenetic tree (i.e., 2.16.7.1.1 or 2.16.7.1.3 for sequences in class 2.16.7.1.2, and 2.16.7.1.3 or 2.16.7.2.3 for sequences in class 2.16.7.1.9), although the classification scores are only slightly above the threshold of 0.5. All four negative patterns of data set three are 'unidentified' at the threshold of 0.5. The results show that the neural network classification scores are directly related to the degree of phylogenetic relationship, and a distant relationship is indicated for full-length sequences if they are given low classification scores.

Small data set. All above observations and discussions are based on the results of the SSU database, which is well represented with phylogenetically diverse as well as closely related organisms. Studies were also conducted for a smaller data set, i.e., the LSU database which consists of only 72 sequences in nine classes. With the small numbers of training patterns (i.e., 51 sequences) and output classes (i.e., 9 classes), smaller input and hidden sizes are used for the neural networks. The n8_50 method (i.e., octagram patterns of nucleic acids with a reduced dimension of 50) is used for sequence encoding, and the network architecture is $50 \times 30 \times 9$ and $50 \times 23 \times 9$, respectively, for the back-propagation and counter-propagation networks. After training, the neural network correctly classifies all 21 prediction sequences as first-fit patterns and yields a predictive accuracy of 100%. The result shows that the neural network method is robust and useful even for small databases.

DISCUSSION

This study extends our protein classification system for classification of ribosomal RNA sequences. It indicates that the sequence encoding methods apply equally well to both the protein and nucleic acid sequences, although the former has a 20-letter alphabet and the latter has a four-letter alphabet. This research further evaluates several mechanisms that can improve system performance. First, minimal network architecture is adopted to improve network generalization and to speed up network training. The SVD computation, which has reduced the sizes of input vector and weight matrix significantly, is one approach towards this goal. Secondly, a pedagogical pattern selection strategy is used to select training patterns of high errors, thereby, reduce total network training time. Thirdly, a modified counter-propagation algorithm with supervised Kohonen learning and dynamic node allocation is developed for this application, and used as an alternative learning paradigm to reduce training time. Currently, efforts are being made to optimize the n-gram encoding and SVD computation, since the total network classification time is dominated by the preprocessing time.

The neural network method compares favorably, in terms of both accuracy and speed, to other methods currently available for phylogenetic classification of rRNA sequences. The neural network method predicts better than the Similarity Rank (which is based on word counts of 7-mers for SSU rRNA), possibly due to two reasons. First, the SVD method used for sequence encoding permits the use of longer, and thus, more discriminatory n-gram patterns (i.e., 8-mers). Furthermore, the SVD method is capable of extracting higher-order semantics and improving the detection of related sequences that may or may not contain the exact n-gram terms [22]. And secondly, the neural network uses supervised learning algorithms to provide clustering with non-linear separations, therefore, it is better than a pure statistical mapping, as been illustrated by the results of the k-nearest neighbor method. The Similarity Rank is very similar to the k-nearest neighbor method, except that its similarity is not measured by an Euclidian distance, but expressed by the ratio of numbers of shared oligomers to unique oligomers. The result of the direct comparisons between Similarity Rank and neural network method for phylogenetic classification should be interpreted with caution, however. It should be noted that Similarity Rank was designed as a similarity search tool for RDP database, and that a similarity search result may not always equate to a phylogenetic classification result. It is possible that a sequence may have a shorter distance to a sequence in a different phylogenetic class than sequences in the same class. (This may be explained by the various rates of evolution between organisms, which causes certain lineage to develop faster from its ancestor than others.) The neural network classification accuracy is significantly better than that of the Blast or Fasta database search methods, probably due to the high degrees of similarities among rRNA sequences. The percentages of sequence identities between most query sequences and training sequences are more than 70 to 80% over long stretches of the entire sequences, even for training sequences of different classes. This results to very high Blast and Fasta scores for all top-hits (most scores are in the thousands) and leads to less accurate class separation. Furthermore, to accurately reflect the phylogenetic relationship, a good short match must rank higher than a less good longer one. Fasta and Blast both do the opposite, as a consequence, the best matches to partial sequences are often missed.

Currently, the classification speed of the neural network system is about an order of magnitude faster than other methods. The classification time of the neural network method will remain low even if there is a 100 to 1000 fold increase in the database size, because the total network classification time is dominated by the preprocessing time, and the preprocessing time is independent of the number of training sequences. On the other hand, the times for the Similarity Rank, Blast and Fasta methods are all directly proportional to the number of sequence entries in the database. With the rate gap widening due to the continuing growth of the database, the neural network approach will become even more important.

The major applications of the classification neural networks are rapid sequence annotation and automated family assignment. Classification of unknown sequences into predefined classes on a phylogenetic tree, without sequence alignment, is a rapid means of sequence annotation. Ribosomal RNA sequences are now the method of choice for surveying the biosphere. It is possible to take a sample from anywhere and use universal primers to the conserved regions of the 16S-like rRNAs and obtain sequence data about what is the sample. The rRNA classification system developed in this study can be used to screen large collections of short ribosomal RNA sequences, such as those produced by environmental characterizations, in order to identify those worthy of more complete sequence determination. The neural classification system can also be used to automate family assignment and help build the phylogenetic tree. Like other sequence databases, the size of the RDP database continues to grow, with a doubling rate over the last five years. The neural network tool could be used by the RDP database to help handle the influx of new data in a timely manner (in collaboration with Carl Woese, University of Illinois).

The neural network tool is generally applicable to any databases that are developed according to family relationships because the neural network employs a 'supervised' learning algorithm. The designs of the neural system can be easily extended to classify other nucleic acid sequences. Preliminary studies have been conducted to classify DNA sequences (containing both protein-encoding regions and intervening sequences) directly into protein superfamilies with satisfactory results. It is, therefore, possible to develop a gene identification system that can classify indiscriminately sequenced DNA fragments.

Presently, a distribution version of the neural network system has been developed and ported to several UNIX machines. The version, consisting of the prediction program and the weight files obtained from off-line training, can be used for speedy on-line sequence classification. The system will be made available to the research community via the use of the RDP e-mail server (in collaboration with Niels Larsen) and an anonymous ftp on our DEC alpha workstation (diana.uthct.edu) (please contact wu@jason.uthct.edu for details). With the adoption of minimal network architecture and other designs discussed in this study, it is now also feasible to develop a version that can do both on-line training and prediction.

ACKNOWLEDGEMENTS

This study is supported in part by grant number R29 LM05524 from the National Library of Medicine. The work is also supported by the University Research and Development Grant Program of the Cray Research, Inc. Its contents are solely the

responsibility of the authors and do not necessarily represent the official views of the granting agencies. The authors wish to acknowledge the computer system support of the Center for High Performance Computing of the University of Texas System, the database support of the Ribosomal Database Project, and valuable discussions from Drs. Carl Woese and Niels Larsen. The authors also thank the editor and the referees for their excellent suggestions.

REFERENCES

1. van Heel, M. (1991) *J. Mol. Biol.*, 220, 877–887.
2. Harris, N., Hunter, L. and States, D. (1992) in *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI Press.
3. Ferran, E. A., Pflugfelder, B. and Ferrara, P. (1994) *Protein Science*, 3, 507–521.
4. Wu, C. H., G. Whitson, J. McLarty, A. Ermongkonchai and T. Chang. (1992) *Protein Science*, 1, 667–677.
5. Barker, W. C., George, D. G., Mewes, H. -W., Pfeiffer, F. and Tsugita, A. (1993) *Nucleic Acids Res.*, 21 (13) Database Issue, 3038–3092.
6. Larsen, N., Olsen, G. J., Maidak, B. L., McCaughey, M. J., Overbeek, R., Macke, T., Marsh, T. L., and Woese, C. R. (1993) *Nucleic Acids Res.*, 21 (13) Database Issue, 3021–3023.
7. Weiss, S. M. and Kulikowski, C. A. (1991) *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers, Inc. San Mateo, CA.
8. Rumelhart, D. E. and McClelland, J. L. (eds) (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press.
9. Hecht-Nielsen, R. (1987) *Applied Optics*, 26, 4979–4984.
10. Hirst, J. D. and Sternberg, M. J. E. (1992) *Biochemistry*, 31, 7211–7218.
11. Bohr, H., Bohr, J., Brunak, S., Cotterill, R.M.J., Fredholm, H., Lautrup, B. and Peterson, S.B. (1990) *FEBS Letters*, 261, 43–46.
12. Kneller, D. G., Cohen, F. E. and Langridge, R. (1990) *J. Mol. Biol.*, 214, 171–182.
13. Qian, N. and Sejnowski, T. J. (1988) *J. Mol. Biol.*, 202, 865–884.
14. Farber, R., Lapedes, A. and Sirotkin, K. (1992) *J. Mol. Biol.*, 226, 471–479.
15. Uberbacher, E.C. and Mural, R. J. (1991) *Proc. Natl. Acad. Sci., USA* 88, 11261–11265.
16. O'Neill, M. C. (1992) *Nucleic Acids Res.*, 20, 3471–3477.
17. Wu, C. H. (1993) *Computers & Chemistry*, 17 (2), 219–227.
18. Altschul, S. F., Gish, W., Miller, W., Myers E. W. and Lipman, D. J. (1990) *J. Mol. Biol.*, 215, 403–410.
19. Pearson, W. R. and Lipman, D. J. (1988). *Proc. Natl. Acad. Sci., USA* 85, 2444–2448.
20. Woese, C. R. (1987) *Microbiological Reviews*, 51, 221–271.
21. Felsenstein, J. (1988) *Annu. Rev. Genet.*, 22, 521–565.
22. Wu, C. H., Berry, M., Shivakumar, S. and McLarty, J. (1994) *Machine Learning*, (In Press).
23. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990) *J. Am. Soc. for Information Science*, 41, 391–407.
24. Cachin, C. (1994) *Neural Networks*, 7, 175–181.
25. Simpson, P. K. (1990) *Artificial Neural Systems: Foundations, paradigms, applications, and implementations*. Pergamon Press, New York.
26. Berry, M.W. (1992) *Intn'l J. Supercomputer Applications*, 6, 13–49.