Application of Neural Networks to Biological Data Mining: A Case Study in Protein Sequence Classification

Jason T. L. Wang, Qicheng Ma
Department of Computer and Information Science
New Jersey Institute of Technology
Newark, NJ 07102, USA
{jason,qicheng}@cis.njit.edu

Dennis Shasha
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA
shasha@cs.nyu.edu

Cathy H. Wu
National Biomedical Research Foundation (NBRF-PIR)
Georgetown Univ. Medical Ctr.
Washington, DC 20007, USA
wuc@nbrf.georgetown.edu

ABSTRACT
Biological data mining aims to extract significant information from DNA, RNA and proteins. The significant information may refer to motifs, functional sites, clustering and classification rules. This paper presents an example of biological data mining: the classification of protein sequences using neural networks. We propose new techniques to extract features from protein data and use them in combination with the Bayesian neural network to classify protein sequences obtained from the PIR protein database maintained at the National Biomedical Research Foundation. To evaluate the performance of the proposed approach, we compare it with other protein classifiers built based on sequence alignment and machine learning methods. Experimental results show the high precision of the proposed classifier and the complementarity of the tools studied in the paper.

Keywords
Bioinformatics, biological data mining, machine learning, neural networks, sequence alignment, feature extraction from protein data

1. INTRODUCTION
As a result of the Human Genome Project and related efforts, DNA, RNA and protein data accumulate at an accelerating rate. Mining these biological data to extract useful knowledge is essential in genome processing. This subject has recently gained significant attention in the data mining community [8]. We present here a case study of the subject: the application of neural networks to protein sequence classification.

The problem studied here can be stated formally as follows. Given are an unlabeled protein sequence S and a known superfamily $\mathcal{F}$. We want to determine whether or not S belongs to $\mathcal{F}$. (We refer to $\mathcal{F}$ as the *target class* and the set of sequences not in $\mathcal{F}$ as the non-target class.) In general, a superfamily is a group of proteins that share similarity in structure and function. If the unlabeled sequence S is detected to belong to $\mathcal{F}$, then one can infer the structure and function of S. This process is important in many aspects of computational biology. For example, in drug discovery, if sequence S is obtained from some disease X and it is determined that S belongs to the superfamily $\mathcal{F}$, then one may try a combination of the existing drugs for $\mathcal{F}$ to treat the disease X.

1.1 Feature Extraction from Protein Data
From a one-dimensional point of view, a protein sequence contains characters from the 20-letter amino acid alphabet $\mathcal{A}$ = {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. An important issue in applying neural networks to protein sequence classification is how to encode protein sequences, i.e., how to represent the protein sequences as the input of the neural networks. Indeed, sequences may not be the best representation at all. Good input representations make it easier for the neural networks to recognize underlying regularities. Thus, good input representations are crucial to the success of neural network learning.

We propose here new encoding techniques that entail the extraction of high-level features from protein sequences. The best high level features should be "relevant". By "relevant," we mean that there should be high mutual information between the features and the output of the neural networks, where the mutual information measures the average reduction in uncertainty about the output of the neural networks given the values of the features.

Another way to look at these features is that they capture both the global similarity and the local similarity of protein sequences. The global similarity refers to the overall similarity among multiple sequences whereas the local similarity refers to motifs (or frequently occurring substrings) in the sequences. Sections 2 and 3 elaborate on how to find the global and local similarity of the protein sequences. Section 4 presents our classification algorithm, which employs the Bayesian neural network originated from Mackay [5]. Section 5 evaluates the performance of the proposed classifier. Section 6 compares our approach with other protein classifiers. Section 7 concludes the paper.

2. GLOBAL SIMILARITY OF PROTEIN SEQUENCES

To calculate the global similarity of protein sequences, we adopt the 2-gram encoding method proposed in [9]. The 2-gram encoding method extracts and counts the occurrences of patterns of two consecutive amino acids (residues) in a protein sequence. For instance, given a protein sequence `PVKTNVK`, the 2-gram amino acid encoding method gives the following result: 1 for `PV` (indicating `PV` occurs once), 2 for `VK` (indicating `VK` occurs twice), 1 for `KT`, 1 for `TN`, 1 for `NV`.

We also adopt the 6-letter exchange group $\{e_1, e_2, e_3, e_4, e_5, e_6\}$ to represent a protein sequence,where $e_1 \in \{\mathtt{H}, \mathtt{R}, \mathtt{K}\}$, $e_2 \in \{\mathtt{D}, \mathtt{E}, \mathtt{N}, \mathtt{Q}\}$, $e_3 \in \{\mathtt{C}\}$, $e_4 \in \{\mathtt{S}, \mathtt{T}, \mathtt{P}, \mathtt{A}, \mathtt{G}\}$, $e_5 \in \{\mathtt{M}, \mathtt{I}, \mathtt{L}, \mathtt{V}\}$, $e_6 \in \{\mathtt{F}, \mathtt{Y}, \mathtt{W}\}$. Exchange groups represent conservative replacements through evolution. For example, the above protein sequence `PVKTNVK` can be represented as $e_4e_5e_1e_4e_2e_5e_1$. The 2-gram exchange group encoding for this sequence is: 1 for e_4e_5, 2 for e_5e_1, 1 for e_1e_4, 1 for e_4e_2, 1 for e_2e_5.

For each protein sequence, we apply both the 2-gram amino acid encoding and the 2-gram exchange group encoding to the sequence. Thus, there are $20 \times 20 + 6 \times 6 = 436$ possible 2-gram patterns in total. If all the 436 2-gram patterns are chosen as the neural network input features, it would require many weight parameters and training data. This makes it difficult to train the neural network—a phenomenon called "curse of dimensionality." Different methods have been proposed to solve the problem by careful feature selection and by scaling of the input dimensionality. We propose here to select relevant features (i.e. 2-grams) by employing a distance measure to calculate the relevance of each feature. Let X be a feature and let x be its value. Let $P(x|Class = 1)$ and $P(x|Class = 0)$ denote the class conditional density functions for feature X, where $Class\_1$ represents the target class and $Class\_0$ is the non-target class. Let $D(X)$ denote the distance function between $P(x|Class = 1)$ and $P(x|Class = 0)$. The distance measure rule prefers feature X to feature Y if $D(X) > D(Y)$, because it is easier to distinguish between $Class\_1$ and $Class\_0$ by observing feature X than feature Y. In our work, each feature X is a 2-gram pattern. Let c denote the occurrence number of the feature X in a sequence S. Let l denote the total number of 2-gram patterns in S and let $len(S)$ represent the length of S. We have $l = len(S) - 1$. Define the feature value x for the 2-gram pattern X with respect to the sequence S as $x = c/(len(S) - 1)$. For example, suppose S = `PVKTNVK`. Then the value of the feature `VK` with respect to S is 2/(7-1) = 0.33.

Because a protein sequence may be short, random pairings can have a large effect on the result. $D(X)$ can be approximated by the Mahalonobis distance [6] as $D(X) = (m_1 - m_0)^2/(d_1^2 + d_0^2)$, where m_1 and d_1 (m_0 and d_0, respectively) are the mean value and the standard deviation of the feature X in the positive (negative, respectively) training dataset. The mean value m and the standard deviation d of the feature X in a set $\mathcal{S}$ of sequences are defined as $m = (\sum_{i=1}^{N} x_i)/N$, $d = \sqrt{(\sum_{i=1}^{N}(x_i - m)^2)/(N - 1)}$, where x_i is the value of the feature X with respect to sequence S_i $\in \mathcal{S}$, and N is the total number of sequences in $\mathcal{S}$.

Let $X_1, X_2, \ldots, X_{N_g}$, $N_g \ll 436$, be the top N_g features (2-gram patterns) with the largest $D(X)$ values.[1] Intuitively, these N_g features occur more frequently in the positive training dataset and less frequently in the negative training dataset. For each protein sequence S (whether it is a training or a test sequence), we examine the N_g features in S, calculate the feature values, and use the N_g feature values as input feature values to the Bayesian neural network for the sequence S.

To compensate for the possible loss of information due to ignoring the other 2-gram patterns, a linear correlation coefficient (LCC) between the values of the 436 2-gram patterns with respect to the protein sequence S and the mean value of the 436 2-gram patterns in the positive training dataset is calculated and used as another input feature value for S. Specifically, the LCC of S is defined as:

$$LCC(S) = \frac{436 \sum_{j=1}^{436} x_j \overline{x_j} - \sum_{j=1}^{436} x_j \sum_{j=1}^{436} \overline{x_j}}{\sqrt{436 \sum_{j=1}^{436} x_j^2 - (\sum_{j=1}^{436} x_j)^2} \sqrt{436 \sum_{j=1}^{436} \overline{x_j}^2 - (\sum_{j=1}^{436} \overline{x_j})^2}} \quad (1)$$

where $\overline{x_j}$ is the mean value of the jth 2-gram pattern, $1 \leq j \leq 436$, in the positive training dataset and x_j is the feature value of the jth 2-gram pattern with respect to S.

3. LOCAL SIMILARITY OF PROTEIN SEQUENCES

The local similarity of protein sequences refers to frequently occurring motifs in the protein sequences. Let $\mathcal{T}_p = \{S_1, \ldots, S_k\}$ be the positive training dataset. We use a previously developed sequence mining tool Sdiscover [7] to find the regular expression motifs of the forms $*X*$ and $*X*Y*$ where each motif has length $\geq Len$ and approximately matches, within Mut mutations, at least $Occur$ sequences in $\mathcal{T}_p$. Here, a mutation could be a mismatch, an insertion, or a deletion of a letter (residue); Len, Mut, and $Occur$ are user-specified parameters. X and Y are segments of a sequence, i.e., substrings made up of consecutive letters, and $*$ is a variable length don't care (VLDC) symbol. The length of a motif is the number of the non-VLDC letters in the motif. When matching a motif with a sequence S_i, a VLDC symbol in the motif is instantiated into an arbitrary number of residues in S_i at no cost. For example, when matching a motif `*VLHGKKVL*` with a sequence `MNVLAHGKKVLKWK`, the first $*$ is instantiated into `MN` and the second $*$ is instantiated into `KWK`. The number of mutations between the motif and the sequence is 1, representing the cost of inserting an `A` in the motif.

Often, the number of motifs returned by Sdiscover is enormous. It's useful to develop a measure to evaluate the significance of these motifs. We propose here to use the minimum description length (MDL) principle [3, 8] to calculate the significance of a motif. The MDL principle states that the best model (a motif in our case) is the one that minimizes the sum of the length, in bits, of the description of the model and the length, in bits, of the description of the data (the positive training sequences in $\mathcal{T}_p$ in our case) encoded by the model.

---

[1] Our experimental results show that choosing $N_g \geq 30$ can yield a reasonably good performance provided one has sufficient (e.g. > 200) training sequences.

3.1 Evaluating the Significance of Motifs

We adopt information theory in its fundamental form[3, 8] to measure the significance of different motifs. The theory takes into account the probability of an amino acid in a motif (or sequence) when calculating the description length of the motif (or sequence). Specifically, Shannon showed that the length in bits to transmit a symbol b via a channel in some optimal coding is $-log_2 P_x(b)$, where $P_x(b)$ is the probability with which the symbol b occurs. Given the probability distribution P_x over an alphabet $\Sigma_x = \{b_1, b_2, \ldots, b_n\}$, we can calculate the description length of any string $b_{k_1} b_{k_2} \ldots b_{k_l}$ over the alphabet Σ_x by $-\sum_{i=1}^{l} log_2 P_x(b_{k_i})$.

In our case, the alphabet Σ_x is the protein alphabet $\mathcal{A}$ containing 20 amino acids. The probability distribution P can be calculated by examining the occurrence frequencies of amino acids in the positive training dataset $\mathcal{T}_p$. One straightforward way to describe (or encode) the sequences in $\mathcal{T}_p$, referred to as Scheme 1, is to encode sequence by sequence, separated by a delimiter \$. Let $dlen(S_i)$ denote the description length of sequence $S_i \in \mathcal{T}_p$. Then $dlen(S_i) = -\sum_{j=1}^{20} n_{a_j} log_2 P(a_j)$ where $a_j \in \mathcal{A}$, $1 \leq j \leq 20$; n_{a_j} is the number of occurrences of a_j in S_i. For example, suppose S_i = `MNVLAHGKKVLKWK` is a sequence in $\mathcal{T}_p$. Then $dlen(S_i) = -(log_2 P(\mathtt{M}) + log_2 P(\mathtt{N}) + 2log_2 P(\mathtt{V}) + 2log_2 P(\mathtt{L}) + log_2 P(\mathtt{A}) + log_2 P(\mathtt{H}) + log_2 P(\mathtt{G}) + 4log_2 P(\mathtt{K}) + log_2 P(\mathtt{W}))$ Let $dlen(\mathcal{T}_p)$ denote the description length of $\mathcal{T}_p = \{S_1, \ldots, S_k\}$. If we ignore the description length of the delimiter \$, then the description length of $\mathcal{T}_p$ is given by $dlen(\mathcal{T}_p) = \sum_{i=1}^{k} dlen(S_i)$

Another method to encode the sequences in $\mathcal{T}_p$, referred to as Scheme 2, is to encode a regular expression motif, say M_j, and then encode the sequences in $\mathcal{T}_p$ based on M_j. Specifically, if a sequence $S_i \in \mathcal{T}_p$ can approximately match M_j, then we encode S_i based on M_j. Otherwise we encode S_i using Scheme 1. Let us use an example to illustrate Scheme 2. Consider, for example, $M_j = *\mathtt{VLHGKKVL}*$. We encode M_j as 1, $*$, V, L, H, G, K, K, V, L, $*$, \$0 where 1 indicates one mutation is allowed in matching M_j with S_i, and \$0 is a delimiter to signal the end of the motif. Let $\sum_1$ denote the alphabet $\{a_1, a_2, \ldots, a_{20}, *, \$0\}$, where $a_1, a_2, \ldots, a_{20}$ are the 20 amino acids. Let P_1 denote the probability distribution over the alphabet $\sum_1$. $P_1(\$0)$ can be approximated by the reciprocal of the average length of motifs. $P_1(*) = n(P_1(\$0))$, $P_1(a_i) = (1-(n+1)P_1(\$0))P(a_i)$, where n denotes the number of VLDCs in the motif M_j. For a motif of the form $*X*$, n is 2; for a motif of the form $*X*Y*$, n is 3.

Given P_1, we can calculate the description length of a motif as follows. Let $M_j = *a_{j_1} a_{j_2}, \ldots, a_{j_k}*$. Let $dlen(M_j)$ denote the description length, in bits, of the motif M_j. Then, $dlen(M_j) = -(2log_2 P_1(*) + log_2 P_1(\$0) + \sum_{i=1}^{k} log_2 P_1(a_{j_i}))$. For instance, consider again $M_j = *\mathtt{VLHGKKVL}*$. We have $dlen(M_j) = -(2log_2 P_1(*) + log_2 P_1(\$0) + 2log_2 P_1(\mathtt{V}) + 2log_2 P_1(\mathtt{L}) + log_2 P_1(\mathtt{H}) + log_2 P_1(\mathtt{G}) + 2log_2 P_1(\mathtt{K}))$. Sequences that are approximately matched by the motif M_j can be encoded with the aid of the motif. For example, consider again $M_j = *\mathtt{VLHGKKVL}*$ and S_i = `MNVLAHGKKVLKWK`. M_j matches S_i with one mutation, representing the cost of inserting an A in the third position of M_j. The first VLDC symbol is instantiated into MN and the second VLDC symbol is instantiated into KWK. We can thus rewrite S_i as MN $\bullet$ SS_i $\bullet$ KWK where SS_i is VLAHGKKVL and $\bullet$ denotes the concatenation of strings. Therefore we can encode S_i as M, N, \$1; 1, (O_I, 3, A); K, W, K, \$1. Here \$1 is a delimiter, 1 indicates that one mutation occurs when matching M_j with S_i and (O_I, 3, A) indicates that the mutation is an insertion that adds the letter A to the third position of M_j. In general, the mutation operations involved and their positions can be observed using approximate string matching algorithms. The description length of the encoded S_i based on M_j, denoted $dlen(S_i, M_j)$, can be calculated easily.

Suppose there are h sequences $S_{p_1} \ldots S_{p_h}$ in the positive training dataset $\mathcal{T}_p$ that can approximately match the motif M_j. The weight (or significance) of M_j, denoted $w(M_j)$, is defined as $w(M_j) = \sum_{i=1}^{h} dlen(S_{p_i}) - (dlen(M_j) + \sum_{i=1}^{h} dlen(S_{p_i}, M_j))$. Intuitively, the more sequences in $\mathcal{T}_p$ approximately matching M_j and the less bits we use to encode M_j and to encode those sequences based on M_j, the larger weight M_j has.

Using Sdiscover, one can find a set $\mathcal{S}$ of regular expression motifs of the forms $*X*$ and $*X*Y*$ from the positive training dataset $\mathcal{T}_p$ where the motifs satisfy the user-specified parameter values Len, Mut and $Occur$. We choose the top N_l motifs with the largest weight. Let $\mathcal{R}$ denote this set of motifs. Suppose a protein sequence S (whether it is a training sequence or a test sequence) can approximately match, within Mut mutations, m motifs in $\mathcal{R}$. Let these motifs be $M_1, \ldots, M_m$. The local similarity (LS) value of S, denoted $LS(S)$, is defined as $LS(S) = \max_{1 \leq i \leq m}\{w(M_i)\}$, if $m \neq 0$; $LS(S) = 0$, otherwise. This LS value is used as an input feature value of the Bayesian neural network for the sequence S. Note that we use the max operator here to maximize discrimination. In general, positive sequences will have large LS values with high probabilities and have small LS values with low probabilities. On the other hand, negative sequences will have small LS values with high probabilities and have large LS values with low probabilities.

4. THE BAYESIAN NEURAL NETWORK CLASSIFIER

We adopt the Bayesian neural network (BNN) originated from Mackay [5] to classify protein sequences. There are $N_g + 2$ input features, including N_g 2-gram patterns, the LCC feature described in Section 2 and the LS feature described in Section 3. Thus, a protein sequence is represented as a vector of $N_g + 2$ real numbers. The BNN has one hidden layer containing multiple hidden units. The output layer has one output unit, which is based on the logistic activation function $f(a) = 1/(1 + e^{-a})$. The BNN is fully connected between the adjacent layers.

Let $\mathcal{D} = \{\mathbf{x}^{(m)}, t_m\}, 1 \leq m \leq N$, denote the training dataset including both positive and negative training sequences. $\mathbf{x}^{(m)}$ is an input feature vector including the $N_g + 2$ input feature values, and t_m is the binary (0/1) target value for the output unit. That is, t_m equals 1 if $\mathbf{x}^{(m)}$ represents a protein sequence in the target class, and 0 otherwise.

Let $\mathbf{x}$ denote the input feature vector for a protein sequence, which could be a training sequence or a test sequence. Given the architecture $\mathbf{A}$ and the weights $\mathbf{w}$ of the BNN, the output value y can be uniquely determined from the input vector $\mathbf{x}$. Because of the logistic activation function $f(a)$ of the output unit, the output value $y(\mathbf{x};\mathbf{w},\mathbf{A})$ can be interpreted as $P(t=1|\mathbf{x},\mathbf{w},\mathbf{A})$, i.e., the probability that $\mathbf{x}$ represents a protein sequence in the target class given $\mathbf{w}$, $\mathbf{A}$. The likelihood function of the data $\mathcal{D}$ given the model is calculated by $P(\mathcal{D}|\mathbf{w},\mathbf{A}) = \Pi_{m=1}^{N} y^{t_m}(1-y)^{1-t_m} = exp(-G(\mathcal{D}|\mathbf{w},\mathbf{A}))$. Here $G(\mathcal{D}|\mathbf{w},\mathbf{A})$ is the cross-entropy error function, i.e. $G(\mathcal{D}|\mathbf{w},\mathbf{A}) = -\sum_{m=1}^{N} t_m log(y) + (1-t_m) log(1-y)$.

The $G(\mathcal{D}|\mathbf{w},\mathbf{A})$ is the objective function in a non-Bayesian neural network training process and is minimized. This process assumes all possible weights are equally likely. The weight decay is often used to avoid overfitting on the training data and poor generalization on the test data by adding a term $\frac{\alpha}{2}\sum_{i=1}^{q} w_i^2$ to the objective function, where α is the weight decay parameter (hyperparameter), $\sum_{i=1}^{q} w_i^2$ is the sum of the square of all the weights of the neural network, and q is the number of weights. This objective function is minimized to penalize the neural network with weights of large magnitudes. Thus, it penalizes an over-complex model and favors a simple model. However, there is no precise way to specify the appropriate value of α, which is often tuned offline. In contrast, in the Bayesian neural network, the hyperparameter α is interpreted as the parameter of a model, and is optimized online during the Bayesian learning process. We adopt the Bayesian training of neural networks described in [5] to calculate and maximize the evidence of α, namely $P(\mathcal{D}|\alpha,\mathbf{A})$. The training process employs an iterative procedure; each iteration involves three levels of inference.

In classifying an unlabeled test sequence S represented by its input feature vector $\mathbf{x}$, the output of the BNN, $P(t=1|\mathbf{x},\mathbf{w},\mathbf{A})$, is the probability that S belongs to the target class. If the probability is greater than the decision boundary 0.5, S is assigned to the target class; otherwise S is assigned to the non-target class.

5. PERFORMANCE OF THE BNN CLASSIFIER

We carried out a series of experiments to evaluate the performance of the proposed BNN classifier on a Pentium II PC running the Linux operating system. The data used in the experiments were obtained from the International Protein Sequence Database [2], release 62, in the Protein Information Resource (PIR) maintained by the National Biomedical Research Foundation (NBRF-PIR) at the Georgetown University Medical Center. This database, accessible at `http://www-nbrf.georgetown.edu`, currently has 172,684 sequences.

Four positive datasets were considered; they were globin, kinase, ras, and ribitol superfamilies, respectively, in the PIR protein database. The globin superfamily contained 831 protein sequences with lengths ranging from 115 residues to 173 residues. The kinase superfamily contained 350 protein sequences with lengths ranging from 151 residues to 502 residues. The ras superfamily contained 386 protein sequences with lengths ranging from 106 residues to 322 residues. The ribitol superfamily contained 319 protein sequences with lengths ranging from 129 residues to 335 residues. The negative dataset contained 1,650 protein sequences, also taken from PIR protein database, with lengths ranging from 100 residues to 200 residues; these negative sequences did not belong to any of the four positive superfamilies. Both the positive and negative sequences were divided into training sequences and test sequences where the size of the training dataset equaled the size of the test dataset multiplied by an integer r. With the same training data, we tested several BNN models with different numbers of hidden units. When there were 2 hidden units, the evidence obtained was the largest, so we fixed the number of hidden units at 2. Models with more hidden units would require more training time while achieving roughly the same performance.

The parameter values used in the experiments were as follows. N_g (number of 2-gram patterns used by BNN) was 60; N_l (number of motifs used by BNN) was 20; Len (length of motifs for Sdiscover) was 6; Mut (mutation number for Sdiscover) was 2; $Occur$ (occurrence frequency of motifs for Sdiscover) was 1/10; r (size ratio) was 2. The measure used to evaluate the performance of the BNN classifier is *precision*, PR, which is defined as $PR = (NumCorrect/NumTotal) \times 100\%$, where $NumCorrect$ is the number of test sequences classified correctly and $NumTotal$ is the total number of test sequences. We present the results for the globin superfamily only; the results for the other three superfamilies were similar.

5.1 Results

In the first experiment, we considered only 2-gram patterns and evaluated their effect on the performance of the proposed BNN classifier. The performance improves initially as N_g increases. The reason is that the more 2-gram patterns we use, the more precisely we represent the protein sequences. However, when N_g is too large (e.g. > 90), the training data is insufficient and the performance degrades. In the second experiment, we considered only motifs found by Sdiscover and studied their effect on the performance of the classifier. 1,597 motifs were found, with lengths ranging from 6 residues to 34 residues. The more motifs one uses, the better performance one achieves. However, that would also require more time in matching a test sequence with the motifs. We experimented with other parameter values for Len, Mut and $Occur$ used in Sdiscover. The results didn't change as these parameters changed. We also tested the effects made by each individual feature and their combinations. We found that the best performance is achieved when all the features are used, in which case the PR of the BNN classifier is 98%.

6. COMPARISON OF THREE PROTEIN CLASSIFIERS

The purpose of this section is to compare the proposed BNN classifier with the commonly used BLAST classifier [1] built based on sequence alignment and the SAM classifier [4] built based on hidden Markov models. The BLAST version number was 2.0.10. We used default values for the parameters in BLAST. The SAM version number was 3.0; we used internal weighting method 2 as suggested in [4].

For BLAST, we aligned an unlabeled test sequence S with the positive training sequences (i.e. those in the target class, e.g., the globin superfamily) and the negative training sequences in the non-target class using the tool. If S's score was below the threshold of the e value which was fixed at 10, then S was undetermined. Otherwise, we assigned S to the class containing the sequence best matching S.

For SAM, we employed the program buildmodel to build the HMM model by using only the positive training sequences. We then calculated the log-odds scores for all the training sequences using the program hmmscore. The log-odds scores were all negative real numbers; the scores (e.g. -100.3) for the positive training sequences were generally smaller than the scores (e.g. -4.5) for the negative training sequences. The largest score S_p for the positive training sequences and the smallest score S_n for the negative training sequences were recorded. Let B_{high} = max $\{S_p, S_n\}$ and B_{low} = min $\{S_p, S_n\}$. We then calculated the log-odds scores for all the test sequences using the program hmmscore. If the score of an unlabeled test sequence S was smaller than B_{low}, S was classified as a member of the target class, e.g., a globin sequence. If the score of S was larger than B_{high}, S was classified as a member of the non-target class. If the score of S was between B_{low} and B_{high}, S was undetermined.

In addition to the three basic classifiers BNN, BLAST and SAM, we developed an ensemble of classifiers, called COMBINER, that employs an unweighted voter and works as follows. If the BNN, BLAST, and SAM agree on the classification results, the result of COMBINER is the same as the results of the three classifiers; if two classifiers agree on the classification results, the result of COMBINER is the same as the results of these two classifiers; if none of the classifiers agrees on the classification results, the result of COMBINER is undetermined. We found that in comparison with BLAST and SAM, the BNN classifier is faster, without yielding any undetermined sequence. COMBINER achieves the highest precision ($> 99\%$) among all the classifiers for all the four superfamilies globin, kinase, ras, and ribitol.

7. CONCLUSION

In this paper we have presented a Bayesian neural network approach to classifying protein sequences. The main contributions include (i) the development of new algorithms for extracting the global similarity and the local similarity from the sequences that are used as input features of the Bayesian neural network; (ii) the development of new measures for evaluating the significance of 2-gram patterns and frequently occurring motifs used in classifying the sequences; (iii) experimental studies in which we compare the performance of the proposed BNN classifier with two other classifiers, namely BLAST and SAM, on four different superfamilies of sequences in the PIR protein database.

The main findings of our work include the following. (1) The three studied classifiers, BNN, SAM and BLAST, complement each other; combining them yields better results than using the classifiers individually. (2) The training phase, which is done only once, of the two learning-based classifiers BNN and SAM may take some time. After the classifiers are trained, they run significantly faster than the alignment tool BLAST in sequence classification.

8. ACKNOWLEDGMENTS

This work was supported in part by NSF grants IRI-9531548 and IRI-9531554, and by a grant from Novartis Pharmaceuticals Corporation. We thank SIGKDD reviewers for their thoughtful comments. We also thank Dr. David Mackay for sharing the Bayesian neural network software with us, and Dr. Richard Hughey for providing us with the SAM software.

9. REFERENCES

[1] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped Blast and PSI-Blast: A new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[2] W. C. Barker, J. S. Garavelli, D. H. Haft, L. T. Hunt, C. R. Marzec, B. C. Orcutt, G. Y. Srinivasarao, L. S. L. Yeh, R. S. Ledley, H. W. Mewes, F. Pfeiffer, and A. Tsugita. The PIR-international protein sequence database. *Nucleic Acids Research*, 26(1):27–32, 1998.

[3] A. Brazma, I. Jonassen, E. Ukkonen, and J. Vilo. Discovering patterns and subfamilies in biosequences. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 34–43, 1996.

[4] R. Karchin and R. Hughey. Weighting hidden Markov models for maximum discrimination. *Bioinformatics*, 14(9):772–782, 1998.

[5] D. J. C. Mackay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):698–714, 1992.

[6] V. V. Solovyev and K. S. Makarova. A novel method of protein sequence classification based on oligopeptide frequency analysis and its application to search for functional sites and to domain localization. *Computer Applications in the Biosciences*, 9(1):17–24, 1993.

[7] J. T. L. Wang, G. W. Chirn, T. G. Marr, B. A. Shapiro, D. Shasha, and K. Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 115–125, 1994.

[8] J. T. L. Wang, B. A. Shapiro, and D. Shasha (eds.). *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications.* Oxford University Press, New York, 1999.

[9] C. H. Wu and J. McLarty. *Neural Networks and Genome Informatics.* Elsevier Science, 2000.