New Techniques for Extracting Features from Protein Sequences*

Jason T. L. Wang† Qicheng Ma‡ Dennis Shasha§ Cathy H. Wu¶

Abstract

In this paper we propose new techniques to extract features from protein sequences. We then use the features as inputs for a Bayesian neural network (BNN) and apply the BNN to classifying 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 bioinformatics tools studied in the paper.

Keywords: Bioinformatics, biological data mining, neural networks, feature extraction from sequences.

---

*This work was supported in part by NSF grants IRI-9531548, IRI-9531554, IIS-9988345, IIS-9988636, and by a grant from Novartis Pharmaceuticals Corporation.

†Department of Computer and Information Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102 (jason@cis.njit.edu). Contact author, Phone: (973) 596-3396, Fax: (973) 596-5777.

‡Life Science Informatics Unit, Department of Functional Genomics, Novartis Pharmaceuticals Corporation, Summit, NJ 07901 (qicheng.ma@pharma.novartis.com).

§Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012 (shasha@cs.nyu.edu).

¶Bioinformatics Program, Protein Information Resource, National Biomedical Research Foundation (NBRF-PIR), Georgetown University Medical Center, 3900 Reservoir Road, NW, Washington, DC 20007 (wuc@nbrf.georgetown.edu).

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 bioinformatics community [1, 3, 10, 13, 15, 35]. We present here a case study in extracting features from protein sequences and using them together with a Bayesian neural network to classify the sequences.

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 bioinformatics and computational biology [26, 36, 38]. 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.

There are several approaches available for protein sequence classification. One approach is to compare the unlabeled sequence S with the sequences in the target class and the sequences in the non-target class using an alignment tool such as BLAST [2]. One then assigns S to the class containing the sequence best matching S.

The second method for protein sequence classification is based on hidden Markov models (HMMs) [24]. The HMM method (e.g., SAM [20] and HMMer [14]) employs a machine learning algorithm, which uses probabilistic graphical models to describe time series and sequence data. It was originally applied to speech recognition [27], and now is also applied to modeling and analyzing protein superfamilies. It is a generalization of the position specific scoring matrix to include insertion and deletion states. Often, an HMM is built for each (super)family. One then scores the unlabeled sequence S with respect to the HMM of a (super)family [31]. If the score is more significant than a cut-off value, then S is regarded as a member of the (super)family.

Another approach for protein sequence classification is to iteratively build a model either based on hidden Markov models (e.g. SAM-T99 [23]) or based on a position specific weight matrix (e.g. PSI-BLAST [2]). The unlabeled sequence S is used as a seed sequence and iteratively searched against the (super)family either by the HMM or by the position specific weight matrix.

In the study presented here, we will compare our approach with BLAST, SAM, and the iterative method using SAM-T99. With hidden Markov models, we choose SAM rather than HMMer because the former outperforms the latter in protein sequence classification [22]. With iterative methods, we choose SAM-T99 rather than PSI-BLAST because the former is more sensitive than the latter in homolog detection [26]. We choose BLAST as a point of comparison because it represents a different computing paradigm, namely performing classification simply via alignment. One interesting finding from our work is that the compared classification methods complement each other; combining them yields higher precision than using them individually, as our experimental results will show later. This is consistent with a previous report [33] in which we gave a preliminary analysis of the complementarity among our approach, BLAST and SAM.

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 [19].

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 [25]. Section 5 evaluates the performance of the proposed classifier. Section 6 compares our approach with the 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, also known as 2-tuple, method as described in [39]. The 2-gram encoding method extracts various patterns of two consecutive amino acid residues in a protein sequence and counts the number of occurrences of the extracted residue pairs.[1] 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`, and 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 [37], 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. These exchange groups are effectively equivalence classes of amino acids and are derived from PAM [12].[2] For example, the above protein sequence `PVKTNVK` can be

[1] The total number of possible patterns from 2-gram encoding is n^2 where n is the number of different letters, namely 20, in the protein alphabet.

[2] Both PAM and BLOSUM [18] are amino acid substitution matrices; the latter is derived from the BLOCKS database [17].

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, and 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 $20^2 + 6^2 = 436$ possible 2-grams in total. If all the 436 2-grams 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 [9, 37]. We propose here to select relevant features (i.e. 2-grams) by employing a distance measure to calculate the relevance of each feature.[3]

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 $P(x|Class = 0)$, defined as [6]

$$D(X) = \int |P(x|Class = 1) - P(x|Class = 0)| dx \tag{1}$$

The distance measure prefers feature X to feature Y if $D(X) > D(Y)$. Intuitively, this means it is easier to distinguish between $Class\_1$ and $Class\_0$ by observing feature X than feature Y. That is, X appears often in $Class\_1$ and seldom in $Class\_0$ or vice versa. In our work, each feature X is a 2-gram. Let c denote the occurrence number of the feature X in a sequence S. Let l denote the total number of 2-grams 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 X with respect to the sequence S as

$$x = \frac{c}{len(S) - 1} \tag{2}$$

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

[3] The term "distance" is from [6, 11], which address feature selection for classification.

the result. We approximate $D(X)$ in Equation (1) by [30]

$$D(X) = \frac{(m_1 - m_0)^2}{d_1^2 + d_0^2} \tag{3}$$

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. Intuitively, in Equation (3), the larger the numerator is (or the smaller the denominator is), the larger the interclass distance is, and therefore the easier to separate $Class\_1$ from $Class\_0$ (and vice versa).

The mean value m and the standard deviation d of the feature X in a set $\mathcal{S}$ of sequences are defined as

$$m = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4}$$

$$d = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2} \tag{5}$$

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-grams) with the largest $D(X)$ values.[4] 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 an unlabeled test sequence), we examine the N_g features in S, calculate their values as defined in Equation (2), 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-grams, a linear correlation coefficient (LCC) between the values of the 436 2-grams with respect to the protein sequence S and the mean value of the 436 2-grams in the positive training dataset is calculated and used as another input feature value for S. Specifically, the LCC

[4] Our experimental results show that choosing $N_g \geq 30$ can yield reasonably good performance provided one has sufficient (e.g. > 200) training sequences. We have also experimented with different combinations of 2-grams, e.g., using the top N_g features together with the bottom N_g features with the smallest $D(X)$ values. The results are worse than using the top N_g features alone.

of S is defined as:

$$LCC(S) = \frac{436 \sum_{j=1}^{436} x_j \overline{x_j} - \sum_{j=1}^{436} x_j \sum_{j=1}^{436} \overline{x_j}}{\sqrt{436 \sum_{j=1}^{436} x_j^2 - (\sum_{j=1}^{436} x_j)^2} \sqrt{436 \sum_{j=1}^{436} \overline{x_j}^2 - (\sum_{j=1}^{436} \overline{x_j})^2}} \tag{6}$$

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

3 Local Similarity of Protein Sequences

In contrast to the 2-grams that occur from the beginning to the end of a sequence (thus referred to as global similarities), the local similarity of protein sequences refers to frequently occurring motifs where a motif is composed of substrings occurring in local regions of a sequence. Let $\mathcal{T}_p = \{S_1, \ldots, S_k\}$ be the positive training dataset. We use a previously developed sequence mining tool Sdiscover [32, 34] 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.

The Sdiscover tool is based on a heuristic that works by taking a small sample $\mathcal{K}$ of sequences from the given set of sequences $\mathcal{T}_p$ and storing them in a generalized suffix tree (GST) [21]. The GST can be constructed asymptotically in $O(n)$ time and space where n is the total length of all sequences in the sample. The heuristic then traverses the GST to generate candidate regular expression motifs and compares these candidate motifs with all the sequences in $\mathcal{T}_p$ to calculate their occurrence numbers. Given a candidate motif M

and a sequence S in $\mathcal{T}_p$, one can determine whether M is within Mut mutations of S in $O(Mut \times |S|)$ time when $O(|M|) = O(\log |S|)$ [40]. Thus Sdiscover can find all the motifs satisfying user-specified parameter values in time $O(n \times Mut \times m \times k)$, where n is the total length of all sequences in the sample $\mathcal{K}$, m is the average length of the sequences in $\mathcal{T}_p$ and k is the total number of sequences in $\mathcal{T}_p$, although the tool is practically much faster due to several optimization heuristics implemented for speeding up the traversal of the GST.

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 [7, 28, 36] 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.

3.1 Evaluating the Significance of Motifs

We adopt information theory in its fundamental form [7, 29] 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 [29] 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

$$-\sum_{i=1}^{l} log_2 P_x(b_{k_i}) \tag{7}$$

In our case, the alphabet Σ_x is the protein alphabet $\mathcal{A}$ containing 20 amino acids. The probability distribution P_x, or P in our case, 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

$$dlen(S_i) = -\sum_{j=1}^{20} n_{a_j} log_2 P(a_j) \tag{8}$$

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

$$dlen(S_i) = -(log_2 P(\mathtt{M}) + log_2 P(\mathtt{N}) + 2log_2 P(\mathtt{V}) + 2log_2 P(\mathtt{L}) + log_2 P(\mathtt{A}) + log_2 P(\mathtt{H}) +$$

$$log_2 P(\mathtt{G}) + 4log_2 P(\mathtt{K}) + log_2 P(\mathtt{W})) \tag{9}$$

Let $dlen(\mathcal{T}_p)$ denote the description length of $\mathcal{T}_p = \{S_1, \ldots, S_k\}$. Then the description length of $\mathcal{T}_p$ is given by

$$dlen(\mathcal{T}_p) = \sum_{i=1}^{k} dlen(S_i) + (k-1) \times dlen(\$) \tag{10}$$

Since the description length of the delimiter \$, $dlen(\$)$, is insignificant, we can ignore it and hence

$$dlen(\mathcal{T}_p) = \sum_{i=1}^{k} dlen(S_i) \tag{11}$$

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.[5] Let us use an example to illustrate Scheme 2. Consider, for example, $M_j = *$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.

[5] The actual number of sequences in $\mathcal{T}_p$ that are encoded by Scheme 2 is dependent on motif. For each motif used in the study presented here, more than 1/10 of the sequences are encoded based on the motif using Scheme 2.

Given P_1, we can calculate the description length of a motif by substituting the probability distribution P_1 for the probability distribution P_x in Equation (7). Specifically, 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

$$dlen(M_j) = -(2log_2P_1(*) + log_2P_1(\$0) + \sum_{i=1}^{k} log_2P_1(a_{j_i})) \tag{12}$$

For instance, consider again $M_j = *\texttt{VLHGKKVL}*$. We have

$$dlen(M_j) = -(2log_2P_1(*) + log_2P_1(\$0) + 2log_2P_1(\texttt{V}) + 2log_2P_1(\texttt{L}) + log_2P_1(\texttt{H}) + log_2P_1(\texttt{G}) + 2log_2P_1(\texttt{K})) \tag{13}$$

Sequences that are approximately matched by the motif M_j can be encoded with the aid of the motif. For example, consider again $M_j = *\texttt{VLHGKKVL}*$ and $S_i = \texttt{MNVLAHGKKVLKWK}$. 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 [40]. The description length of the encoded S_i based on M_j, denoted $dlen(S_i, M_j)$, can be calculated easily as in Equation (12).

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

$$w(M_j) = \sum_{i=1}^{h} dlen(S_{p_i}) - (dlen(M_j) + \sum_{i=1}^{h} dlen(S_{p_i}, M_j)) \tag{14}$$

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 weights. Let $\mathcal{R}$ denote this set of motifs. Suppose a protein sequence S (whether it is a training sequence or an unlabeled 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

$$LS(S) = \begin{cases} \max_{1 \leq i \leq m}\{w(M_i)\} & \text{if } m \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

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.

Remark. Essentially, the proposed scheme is to count amino acids in a sequence (or motif). This scheme is not complete in the sense that different sequences may have the same description length when they have the same number of the same amino acids. Second, there may be multiple ways to align a motif M with a sequence S and hence the description length of the encoded sequence S based on M may not be unique. As a consequence, the weight of a motif defined in Equation (14) may not be unique (in which case the proposed heuristic randomly picks one). There are several other approaches for finding motifs of different forms and for calculating their significance values (see, e.g. [7, 8, 16, 36]). However, motifs have relatively little effect on PIR sequence classification and a combination of the proposed techniques already yields a very high precision, as our experimental results show later.

4 The Bayesian Neural Network Classifier

We adopt the Bayesian neural network (BNN) originated from Mackay [25] to classify protein sequences.[6] There are $N_g + 2$ input features, including N_g 2-grams, the LCC

[6]Software available at `http://wol.ra.phy.cam.ac.uk/pub/mackay/README.html`.

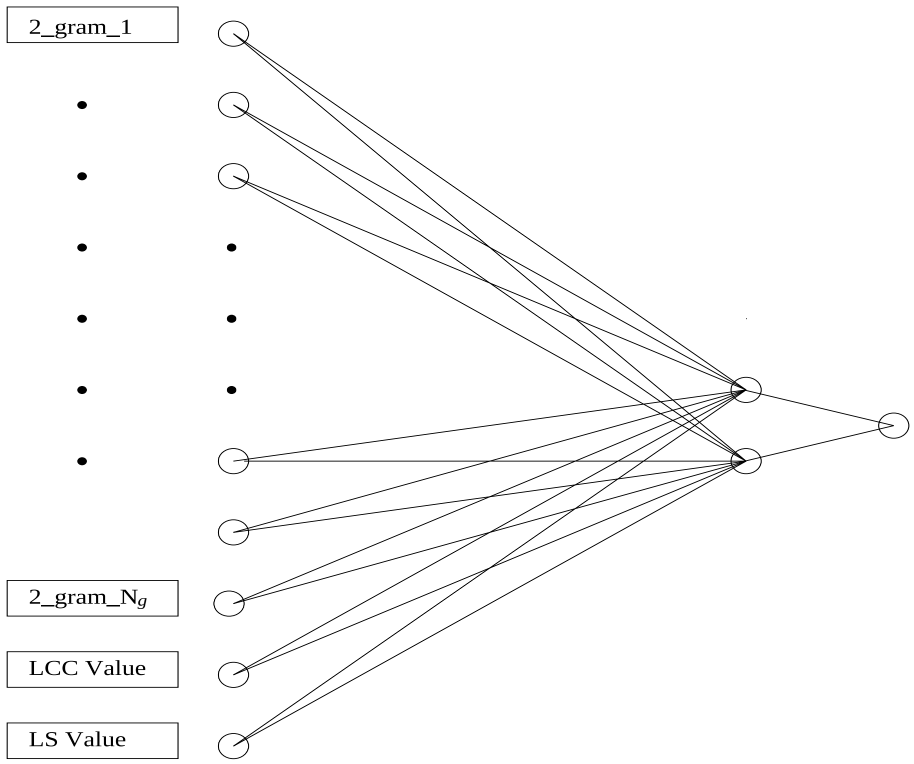

Figure 1: The Bayesian neural network architecture.

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 $f(a) = \frac{1}{1+e^{-a}}$. The BNN is fully connected between the adjacent layers. Fig. 1 illustrates an example BNN model with 2 hidden units.

Let $\mathcal{D} = \{\mathbf{x}^{(m)}, t_m\}, 1 \le m \le N$, 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

$$P(\mathcal{D}|\mathbf{w},\mathbf{A}) = \Pi_{m=1}^{N} y^{t_m}(1-y)^{1-t_m} = exp(-G(\mathcal{D}|\mathbf{w},\mathbf{A})) \tag{16}$$

where $G(\mathcal{D}|\mathbf{w},\mathbf{A})$ is the cross-entropy error function,

$$G(\mathcal{D}|\mathbf{w},\mathbf{A}) = -\sum_{m=1}^{N} t_m log(y) + (1-t_m) log(1-y) \tag{17}$$

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 squares 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 [25] 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. Fig. 2 illustrates the training process of the BNN.

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. In general, for an unlabeled test sequence S with m amino acids, it takes $O(m)$ time to calculate 2-gram feature values, $O(m)$ time to calculate the LCC feature value, and $O(m \times n)$ time to calculate the LS feature value where n is the total length of the motifs chosen, and constant time for calculating the probability $P(t=1|\mathbf{x},\mathbf{w},\mathbf{A})$. Thus, the time complexity of our approach to classifying the unlabeled test sequence S is $O(m \times n)$.

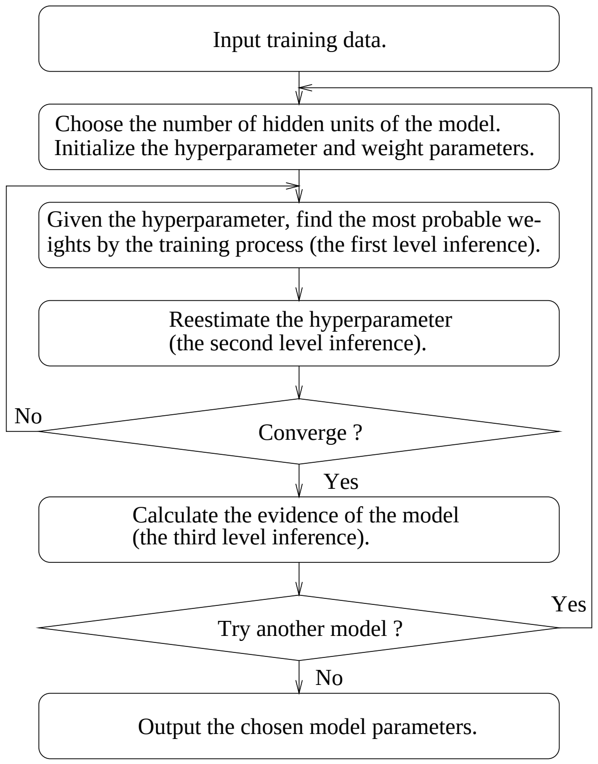

Figure 2: The training process of the Bayesian neural network.

5 Performance of the BNN Classifier

5.1 Data

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 [5], 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://pir.georgetown.edu`, currently has 172,684 sequences. Table 1 summarizes the data used in the experiments.

Four positive datasets were considered; they were globin, kinase, ras, and ribitol superfamilies, respectively, in the PIR protein database. The negative dataset contained 1,650 protein sequences, also taken from the PIR protein database, with lengths ranging from 100 residues to 200 residues; these negative sequences did not belong to any of the four

Dataset	N	L_m	L_x
Globin	831	115	173
Kinase-related transforming protein	350	151	502
Ras transforming protein	386	106	322
Ribitol dehydrogenase	319	129	335
Negative sequences	1,650	100	200

Table 1: Data used in the experiments. N is the number of sequences, L_m is the minimal length of the sequences, and L_x is the maximal length of the sequences.

positive superfamilies. Both the positive and negative sequences were randomly 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 (cf. Fig. 2), 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.

Table 2 summarizes the parameters and base values used in the experiments. The measure used to evaluate the performance of the BNN classifier is *precision*, PR, which is defined as

$$PR = \frac{NumCorrect}{NumTotal} \times 100\% \tag{18}$$

where $NumCorrect$ is the number of test sequences classified correctly and $NumTotal$ is the total number of test sequences. In general, precision is a comprehensive measure in the sense that it considers true positives, false positives, true negatives, false negatives and unclassified sequences;[7] it is used here to find the best parameter values of the proposed BNN classifier. A false positive is a non-target member sequence that was misclassified as a target member sequence. A false negative refers to a sequence in the target class (e.g. the globin superfamily) that was misclassified as a non-target member. We present the results for the globin superfamily only; the results for the other three superfamilies were

[7]Note that the BNN classifier does not yield any unclassified sequence. By contrast, the three other classifiers BLAST, SAM and SAM-T99 we will compare with yield unclassified sequences, as our experimental results will show later.

Parameter	Meaning	Value
N_g	Number of 2-grams used by BNN	60
N_l	Number of motifs used by BNN	20
Len	Length of motifs for Sdiscover	6
Mut	Mutation number for Sdiscover	2
$Occur$	Occurrence frequency of motifs for Sdiscover	1/10
r	size ratio	2

Table 2: Parameters and their base values for the proposed BNN classifier.

similar.

5.2 Results

In the first experiment, we considered only 2-grams and evaluated their effect on the performance of the proposed BNN classifier. Fig. 3 graphs PR as a function of N_g. It can be seen that the performance improves initially as N_g increases. The reason is that the more 2-grams 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 general, the larger N_g is, the more input features the BNN classifier has, and thus the larger training dataset BNN requires. In our case, there are 561 positive training sequences and 1,089 negative training sequences. When $N_g > 90$, these data become too few to yield reasonably good performance. Figuring out how big the parameter N_g should be requires some tuning. We have not yet worked out a theory for it.

In the second experiment, we considered only motifs found by Sdiscover and studied their effect on the performance of the BNN classifier. 1,597 motifs were found, with lengths ranging from 6 residues to 34 residues. Fig. 4 graphs PR as a function of N_l. It can be seen that 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.[8] We experimented with other parameter values for Len, Mut and $Occur$ used in Sdiscover. The results didn't change as these parameters changed.

[8] The time spent in matching a test sequence with the motifs is linearly proportional to the number of the motifs one uses.

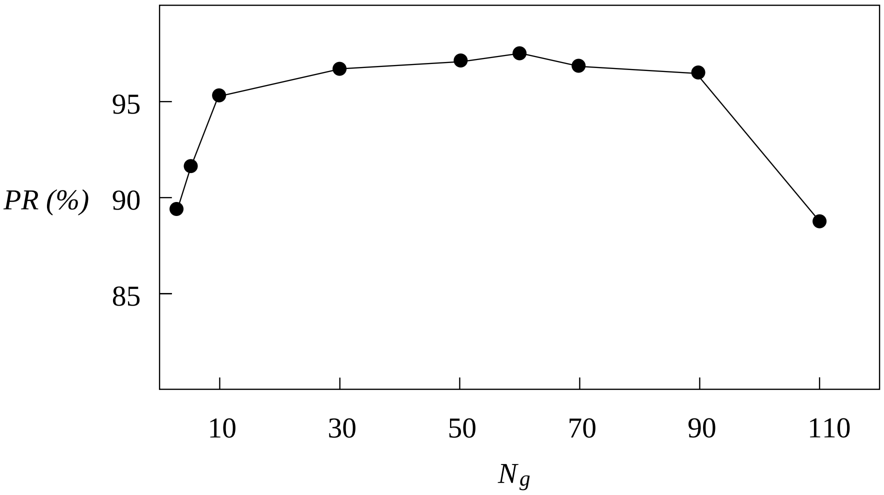

Figure 3: Impact of N_g in the BNN classifier.

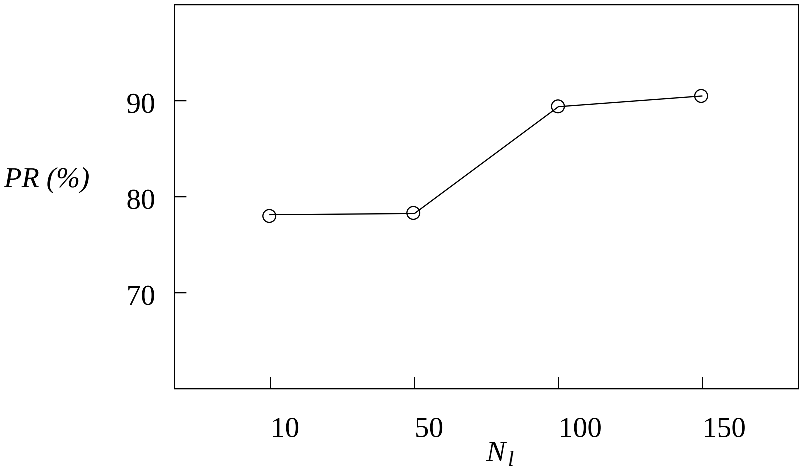

Figure 4: Effect of N_l in the BNN classifier.

Fig. 5 compares the effects of the various types of features introduced in the paper. To isolate the effects of these features, we began by using only one type of features and then using their combinations. It can be seen that features generated from global similarities yield better results than that generated from local similarities. This happens because PIR superfamilies are categorized based on the global similarities of sequences. Note also that the best performance is achieved when all the features are used.

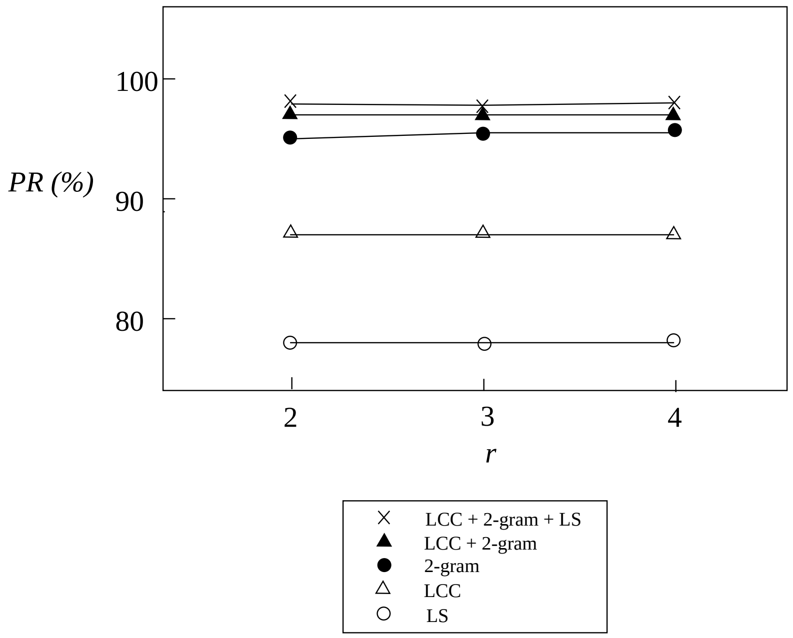

Figure 5: Comparison of the various types of features used by the BNN classifier.

Tool	Underlying techniques
BNN	Bayesian neural networks
BLAST	Similarity search and pairwise alignment
SAM	Hidden Markov models
SAM-T99	Iterative hidden Markov models

Table 3: The bioinformatics tools studied in the paper.

6 Comparison of Four Protein Classifiers

The purpose of this section is to compare the proposed BNN classifier with the BLAST classifier [2] built based on sequence alignment, and with the SAM and SAM-T99 classifiers [23] built based on hidden Markov models. Table 3 summarizes the studied tools. The parameter values for the BNN classifier were as shown in Table 2. The BNN classifier used both 2-grams and regular expression motifs.

The BLAST version number was 2.0.10. We used default values for the parameters in BLAST. For this tool, 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 shown in Table 1 using the tool. If S's score was below the threshold of the expectation (e) value of BLAST, S was undetermined or unclassified. Otherwise, we assigned S to the class containing the sequence best matching S.

The SAM version number was 3.2.1. For this tool, we employed the program buildmodel to build the HMM model by using only the positive training sequences. We then calculated the log-odds scores [14] for all the training sequences using the program hmmscore.[9] 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 unlabeled 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 unclassified or undetermined.

The SAM-T99 version number was also 3.2.1. For this tool, we built an HMM (target model) for each unlabeled test sequence S. We then scored all the training sequences using the HMM target model. If the lowest score of the training sequences was higher than the expectation value (E-value) of the HMM target model, the test sequence S was undetermined or unclassified.[10] Otherwise, we assigned the test sequence to the class containing the training sequence having the lowest E-value. The target model was built in four iterations. In the first iteration, SAM-T99 used BLAST to compare the test sequence S with sequences in the non-redundant protein database maintained at National Center for Biotechnology Information (NCBI) and chose a set of close homologs to build an initial HMM. It also did a BLAST search of the test sequence S against the non-redundant protein

[9] We used log-odds scores, as opposed to E-values, for this tool because the E-value for a training sequence was calculated with respect to the training dataset while the E-value for a test sequence was calculated with respect to the test dataset. These two kinds of E-value were not directly comparable.

[10] The E-value of the HMM target model used in the study presented here was 20. We have experimented with other E-values and the results were worse.

database with a very loose cut-off value to get a pool of potential homologs. Then the HMM obtained from the previous iteration was compared against the pool of potential homologs with a looser cut-off value than that of the previous iteration to find weaker homologs. These weaker homologs were included to build a new HMM for the next iteration. This whole process was repeated three times.

In comparing the relative performance of these tools, we use four more measures in addition to the precision PR defined in the previous section: $specificity$, $sensitivity$, $unclassified_p$ and $unclassified_n$ where

$$\text{specificity} = (1 - \frac{N_{fp}}{N_{ng}}) \times 100\% \tag{19}$$

$$\text{sensitivity} = (1 - \frac{N_{fn}}{N_{po}}) \times 100\% \tag{20}$$

$$\text{unclassified}_p = \frac{N_{up}}{N_{po}} \times 100\% \tag{21}$$

$$\text{unclassified}_n = \frac{N_{un}}{N_{ng}} \times 100\% \tag{22}$$

N_{fp} is the number of false positives, N_{fn} is the number of false negatives, N_{up} is the number of undetermined positive test sequences, N_{un} is the number of undetermined negative test sequences, N_{ng} is the total number of negative test sequences, and N_{po} is the total number of positive test sequences. Note that in contrast to PR, $specificity$ and $sensitivity$ do not consider unclassified sequences. That's why we also add the $unclassified_p$ and $unclassified_n$ measures for performance evaluation.

In the first experiment, we studied the effect of the threshold of the e value in the BLAST classifier. Fig. 6 shows the impact of e values on the performance of BLAST. It can be seen that with e = 10, BLAST performs well. With smaller e values (e.g. 0.1), the specificity of BLAST can approach 100% with very few false positives while the number of unclassified sequences is enormous. Thus, we fixed the threshold of the e value at 10 in subsequent experiments.

Tables 4, 5, 6, and 7 summarize the results and classification times, in seconds, of the four studied tools, referred to as basic classifiers, on the four superfamilies in Table

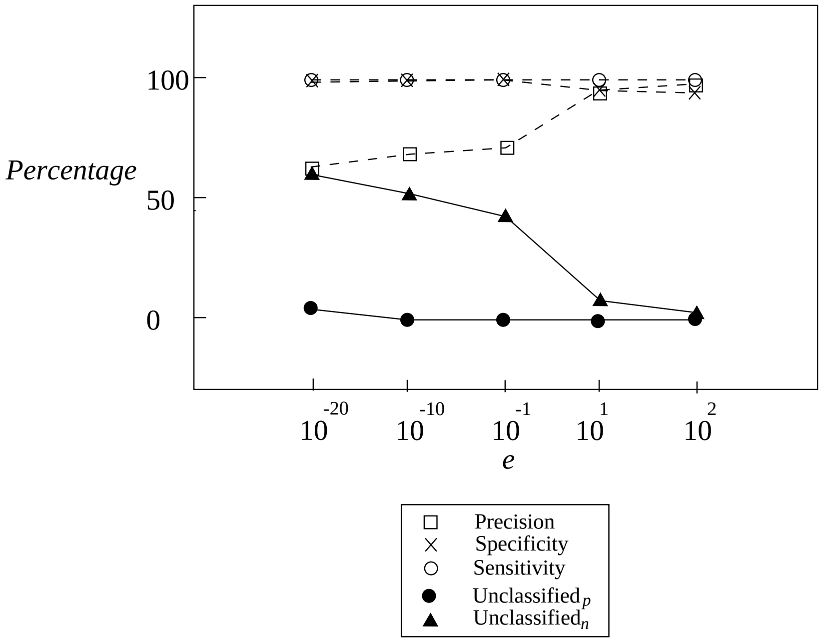

Figure 6: Impact of e values for BLAST.

	BNN	BLAST	SAM	SAM-T99	COMBINER
Precision	98.0%	92.7%	95.3%	93.2%	99.8%
Specificity	98.0%	95.7%	99.8%	94.9%	99.6%
Sensitivity	98.0%	100.0%	99.6%	100.0%	100.0%
Unclassified$_p$	0.0%	0.0%	1.1%	0.0%	0.0%
Unclassified$_n$	0.0%	6.7%	6.2%	5.1%	0.0%
CPU time	36	1,515	80	848,961	—

Table 4: Comparison of the studied classifiers on the globin superfamily.

	BNN	BLAST	SAM	SAM-T99	COMBINER
Precision	99.0%	86.2%	99.4%	92.6%	99.6%
Specificity	98.8%	87.8%	99.5%	93.1%	99.5%
Sensitivity	100.0%	100.0%	100.0%	100.0%	100.0%
Unclassified$_p$	0.0%	0.0%	0.0%	0.0%	0.0%
Unclassified$_n$	0.0%	4.4%	0.2%	2.0%	0.0%
CPU time	30	1,214	63	2,168,005	—

Table 5: Comparison of the studied classifiers on the kinase superfamily.

	BNN	BLAST	SAM	SAM-T99	COMBINER
Precision	98.7%	91.0%	95.5%	91.6%	99.7%
Specificity	99.3%	95.0%	99.8%	92.2%	99.6%
Sensitivity	96.1%	100.0%	100.0%	100.0%	100.0%
Unclassified$_p$	0.0%	0.0%	3.1%	0.0%	0.0%
Unclassified$_n$	0.0%	6.0%	4.6%	2.5%	0.0%
CPU time	29	1,232	64	637,424	—

Table 6: Comparison of the studied classifiers on the ras superfamily.

	BNN	BLAST	SAM	SAM-T99	COMBINER
Precision	96.6%	88.5%	99.4%	90.4%	99.4%
Specificity	97.0%	92.6%	100.0%	91.1%	99.2%
Sensitivity	94.3%	100.0%	100.0%	100.0%	100.0%
Unclassified$_p$	0.0%	0.0%	2.0%	0.0%	0.0%
Unclassified$_n$	0.0%	6.2%	0.3%	2.5%	0.0%
CPU time	27	1,212	62	747,821	—

Table 7: Comparison of the studied classifiers on the ribitol superfamily.

Classification results	Percentage of the test sequences
All classifiers agreed and all were correct	80.88%
All classifiers agreed and all were wrong	0.07%
The classifiers disagreed and one of them was correct	18.91%
The classifiers disagreed and all were wrong	0.14%

Table 8: Complementarity among the four studied tools BNN, BLAST, SAM and SAM-T99. The percentages in the table add up to 100%.

1.[11] In addition to the basic classifiers, we developed an ensemble of classifiers, called COMBINER, that employs a weighted voter and works as follows. If a basic classifier gives an "undetermined" verdict, the classifier is regarded as "abstaining" and its verdict is not counted. The result of COMBINER is the same as the result of the majority of the remaining classifiers. If there is a tie on the verdicts given by the remaining classifiers, the result of COMBINER is the same as the result of the BNN classifier. We see that in comparison with BLAST, SAM and SAM-T99, the BNN classifier is faster, yielding fewer unclassified sequences. COMBINER achieves the highest precision and SAM-T99 requires most time among all the classifiers.

Table 8 shows the complementarity of the four studied tools BNN, BLAST, SAM and SAM-T99. We see that when all the four classifiers agree on their classification result, the result is correct with probability 80.88%/(80.88%+0.07%) = 99.91%.

[11] In examining SAM-T99's CPU time, we note that the time spent for classifying the kinase sequences shown in Table 5 is much higher than the times spent for classifying the other sequences shown in Tables 4, 6 and 7. The reason is that for each kinase sequence, there are many homologs in the non-redundant protein database maintained at NCBI. Thus, SAM-T99 gets more homologs for a kinase sequence than the homologs for the other sequences, and consequently it takes more time to build the HMM target model for the kinase sequence.

7 Conclusion

In this paper we have presented a Bayesian neural network approach to classifying protein sequences. The main contributions of our work include:

- 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;
- the development of new measures for evaluating the significance of 2-grams and frequently occurring motifs used in classifying the sequences;
- experimental studies in which we compare the performance of the proposed BNN classifier with three other classifiers, namely BLAST, SAM and SAM-T99, on four different superfamilies of sequences in the PIR protein database.

The main findings of our work include the following.

- The four studied classifiers, BNN, BLAST, SAM and SAM-T99, complement each other; combining them yields better results than using the classifiers individually.
- The training phase, which is done only once, of the BNN classifier may take some time. After the classifier is trained, it runs significantly faster than BLAST and SAM-T99 in sequence classification.

Future research directions include:

- comparison of motifs generated by different tools in combination with learning-based tools such as neural networks and hidden Markov models when applied to sequence classification in PIR, PROSITE [4, 34] and other protein databases;
- generalization of the classifiers in combination with graph matching algorithms to analyze the sequence-structure relationship in protein and DNA sequences.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments and constructive suggestions, which helped to improve the paper. 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 package.

References

[1] C. F. Allex, J. W. Shavlik, and F. R. Blattner. Neural network input representations that produce accurate consensus sequences from DNA fragment assemblies. *Bioinformatics* **15**(9), 723–728, 1999.

[2] 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.

[3] T. L. Bailey and W. N. Grundy. Classifying proteins by family using the product of correlated p-values. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology*, 1999, pp. 10–14.

[4] A. Bairoch. The PROSITE dictionary of sites and patterns in proteins, its current status. *Nucleic Acids Research* **21**, 3097–3103, 1993.

[5] W. C. Barker, J. S. Garavelli, H. Huang, P. B. McGarvey, B. Orcutt, G. Y. Srinivasarao, C. Xiao, L. S. Yeh, R. S. Ledley, J. F. Janda, F. Pfeiffer, H. W. Mewes, A. Tsugita, and C. H. Wu. The protein information resource (PIR). *Nucleic Acids Research* **28**(1), 41–44, 2000.

[6] M. Ben-Bassat. Use of distance measures, information measures and error bounds in feature evaluation. In: P. R. Krishnaiah and L. N. Kanal (eds.), *Classification, Pattern Recognition and Reduction of Dimensionality: Handbook of Statistics*, volume 2, North-Holland Publishing Company, North-Holland, 1982, pp. 773–791.

[7] 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*, 1996, pp. 34–43.

[8] A. Califano. SPLASH: Structural pattern localization and analysis by sequential histograms. *Bioinformatics*, 2000. URL: `http://www.research.ibm.com/splash/`.

[9] N. A. Chuzhanova, A. J. Jones, and S. Margetts. Feature selection for genetic sequence classification. *Bioinformatics* **14**(2), 139–143, 1998.

[10] M. W. Craven and J. W. Shavlik. Machine learning approaches to gene recognition. *IEEE Expert* **9**(2), 2–10, 1994.

[11] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis* **1**(3), 1997. Electronic Journal URL: `http://www-east.elsevier.com/ida`.

[12] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure* **15** suppl. 3, 345–358, 1978.

[13] S. Eddy. Profile hidden Markov models. *Bioinformatics* **14**(9), 755–763, 1999.

[14] S. Eddy, G. Mitchison, and R. Durbin. Maximum discrimination hidden Markov models of sequence consensus. *Journal of Computational Biology* **2**, 9–23, 1995.

[15] W. N. Grundy and T. L. Bailey. Family pairwise search with embedded motif models. *Bioinformatics* **15**(6), 463–470, 1999.

[16] R. Hart, A. Royyuru, G. Stolovitzky, and A. Califano. Systematic and automated discovery of patterns in PROSITE families. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, 2000.

[17] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research* **19**, 6565–6572, 1991.

[18] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the USA* **89**, 10915–10919, 1992.

[19] H. Hirsh and M. Noordewier. Using background knowledge to improve inductive learning of DNA sequences. In *Proceedings of the Tenth Annual Conference on Artificial Intelligence for Applications*, 1994, pp. 351–357.

[20] R. Hughey and A. Krogh. Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *Computer Applications in the Biosciences* **12**(2), 95–107, 1996.

[21] L. C. K. Hui. Color set size problem with applications to string matching. In: A. Apostolico, M. Crochemore, Z. Galil, and U. Manber (eds.), *Combinatorial Pattern Matching*, Lecture Notes in Computer Science, volume 644, Springer-Verlag, 1992, pp. 230–243.

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

[23] K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics* **14**(10), 846–856, 1998.

[24] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology* **235**(5), 1501–1531, 1994.

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

[26] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *Journal of Molecular Biology* **284**(4), 1201–1210, 1998.

[27] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**(2), 257–286, 1989.

[28] J. Rissanen. Modeling by shortest data description. *Automatica* **14**, 465–471, 1978.

[29] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal* **27**, 379–423, 623–656, 1948.

[30] 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.

[31] E. L. Sonnhammer, S. R. Eddy, and R. Durbin. PFAM: A comprehensive database of protein domain families based on seed alignments. *Proteins* **28**(3), 405–420, 1997.

[32] 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*, 1994, pp. 115–125.

[33] J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu. Application of neural networks to biological data mining: A case study in protein sequence classification. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 305–309.

[34] J. T. L. Wang, T. G. Marr, D. Shasha, B. A. Shapiro, G. W. Chirn, and T. Y. Lee. Complementary classification approaches for protein sequences. *Protein Engineering* **9**(5), 381–386, 1996.

[35] J. T. L. Wang, S. Rozen, B. A. Shapiro, D. Shasha, Z. Wang, and M. Yin. New techniques for DNA sequence classification. *Journal of Computational Biology* **6**(2), 209–218, 1999.

[36] 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.

[37] C. H. Wu, M. Berry, Y. S. Fung, and J. McLarty. Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine Learning* **21**, 177–193, 1995.

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

[39] C. H. Wu, G. Whitson, J. McLarty, A. Ermongkonchai, and T. C. Chang. Protein classification artificial neural system. *Protein Science* **1**(5), 667–677, 1992.

[40] S. Wu and U. Manber. Fast text searching allowing errors. *Communications of the ACM* **35**(10), 83–91, 1992.