Identification of motifs with insertions and deletions in protein sequences using self-organizing neural networks

Derong Liu[a,b,c,*], Xiaoxu Xiong[a], Zeng-Guang Hou[b], Bhaskar DasGupta[c]

[a] *Department of Electrical and Computer Engineering, University of Illinois, Chicago, Illinois, USA*

[b] *The Key Laboratory for Complex Systems and Intelligence Science, Chinese Academy of Sciences, Beijing, P. R. China*

[c] *Department of Computer Science, University of Illinois, Chicago, Illinois, USA*

Abstract

The problem of motif identification in protein sequences has been studied for many years in the literature. Current popular algorithms of motif identification in protein sequences face two difficulties, high computational cost and the possibility of insertions and deletions. In this paper, we provide a new strategy that solve the problem more efficiently. We develop a self-organizing neural network structure with multiple levels of subnetworks to make an intelligent classification of the subsequences obtained from protein sequences. We maintain a low computational complexity through the use of this multi-level structure so that the classification of each subsequence is performed with respect to a small subspace of the whole input space. The new definition of pairwise distance between motif patterns provided in this paper can deal with up to two insertions/deletions allowed in a motif, while other existing algorithm can only deal with one insertion or deletion. We also maintain a high reliability using our self-organizing neural network since it will grow as needed to make sure all input patterns are considered and are given the same amount of attention. Simulation results show that our algorithm significantly outperforms existing algorithms in both accuracy and reliability aspects.

Keywords: DNA; Motif identification; Multiple sequence alignment; Neural network; Self-organization

1 Introduction

DNA, RNA and protein are the three most important molecules of life. DNA and RNA are made of four different letters, and protein are made of 20 different letters. DNA, RNA and protein sequences can be thought of as being composed of motifs interspersed in relatively unconstrained sequences. A motif is a short stretch of a molecule that forms a highly constrained sequence, usually 8 to 20 letters long. Motif discovery is a basic problem in computational biology, as sequence similarity usually implies homology and functional similarity of the proteins or genes encoded by such sequences (Attwood & Parry-Smith, 1999).

The expression of a motif can be in one of the following three forms.

1) Use an actual sequence as the description of a motif. Such a sequence is also called a consensus sequence. Each column of the consensus sequence is the letter that appears most frequently in all known examples of that motif, e.g., $ACTTATAA$ and $AGTTATAA$ are two examples of consensus sequence of a motif.

2) Use the so-called "degenerate" expression to show all possible letters for each column of a motif. For example, the expression
$$A-[CG]-T-T-[AC]-[TCG]-A-A \tag{1}$$
indicates that $AGTTCTAA$ and $ACTTAGAA$ are two of the possible occurrences; see, for example, Linhart & Shamir (2002) for similar concepts used in the design of degenerate primers.

3) Use a more biologically plausible representation to describe a motif. In this case, a probability matrix can be used to assign a different probability to each possible letter in each column of the motif (Bailey & Gribskov, 1998). For example, Table 1 shows a probability matrix representation of the motif given by (1). This matrix representation not only gives the possibility of which letter can appear in each column of the motif, but also shows the probability of their appearances. For example, the sixth column of this motif will have letters C, G, and T appearing with probabilities of 20%, 30%, and 50%, respectively.

*Corresponding author. Address: Department of Electrical and Computer Engineering, University of Illinois, Chicago, Illinois 60607, USA. Tel.: (312) 355-4475; fax: (312) 996-6465. *E-mail addresses:* dliu@ece.uic.edu (D. Liu), xxiong@cil.ece.uic.edu (X. Xiong), hou@compsys.ia.ac.cn (Z. G. Hou), dasgupta@cs.uic.edu (B. DasGupta).

Table 1: Frequency of each letter appearing in every column of a motif

	1	2	3	4	5	6	7	8
A	1.0	0.0	0.0	0.0	0.67	0.0	1.0	1.0
C	0.0	0.5	0.0	0.0	0.33	0.2	0.0	0.0
G	0.0	0.5	0.0	0.0	0.0	0.3	0.0	0.0
T	0.0	0.0	1.0	1.0	0.0	0.5	0.0	0.0

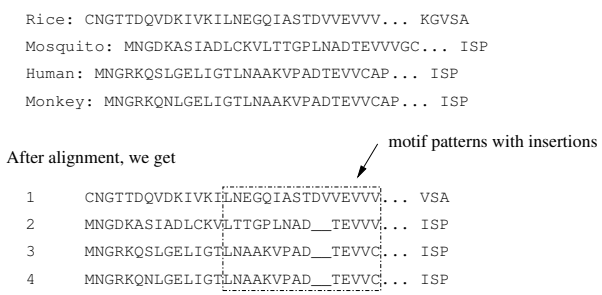

Figure 1: The alignment of protein pieces from rice, mosquito, human and monkey

Multiple sequence alignment is another basic problem in computational biology. One of the potential applications of motif discovery is multiple sequence alignment in which identified motifs are used as marks for sequence alignments. The alignment of a set of sequences is basically a matrix where the rows correspond to the sequences in the set, possibly with some spaces inserted, or some gaps in between (Thompson, Higgins, & Gibson, 1994). Figure 1 shows an example of alignment of pieces of protein sequences gathered from rice, mosquito, human and monkey. These protein pieces are obtained from the Swiss-Prot protein library[1]. By aligning protein sequences, we can discover similarities and changes in the group of sequences, which may help make further decision including gene classification as well as finding cause of disease.

In the motif discovery problem, we have to deal with motifs with mutations, insertions and deletions. Current motif finding algorithm such as MEME (Bailey & Elkan, 1995), Gibbs sampling (Lawrence, et al., 1993) and WINNOWER (Pevzner & Sze, 2000), perform well in finding motifs with only mutations. When dealing with insertions and deletions, especially when there are more than one consecutive insertion or deletion in the motif patterns, these algorithms have trouble in identifying motifs. The Bayesian algorithm (Xie, Li, & Bina, 2004) can deal with cases with insertion and deletion, but not with more than one consecutive insertion or deletion. Insertions and deletions bring great difficulties to the motif discovery problem because they make the result less predictable and more variable. The motif discovery problem in protein sequences can be described as finding similar fields with certain length, with certain maximum number of columns mutated and with certain number of tolerable insertions or deletions. In this paper, we consider the case with at most two insertions or deletions or their combinations in a single motif pattern. Both of the two insertions or deletions can be consecutive.

This paper is organized as follows. In Section 2, we introduce the self-organizing neural network structure for identifying motifs with insertions and/or deletions, after introducing a new definition for calculating the distance between a pair of two subsequences. In Section 3, we provide two simulation examples using generated data set and real protein sequences. In Section 4, we conclude the present paper.

2 Self-Organizing Neural Network for Motif Identification

In this section, we introduce our self-organizing neural network approach for the identification of motifs with insertions and deletions (Xiong, Liu, & Zhang, 2005). The neural network structure has been used in our previous work for identifying motifs with only mutations (Liu, Xiong, & DasGupta, 2005). In this paper, we will use self-organizing neural networks for identifying motifs with insertions and/or deletions, after introducing a new definition for calculating the distance between a pair of two subsequences.

In the present algorithm, subsequences will be divided into 3-letter groups. The calculation of the pairwise distance is based on the definition of distance between a pair of 3-letter groups.

[1] Swiss-Prot, http://us.expasy.org/sprot/.

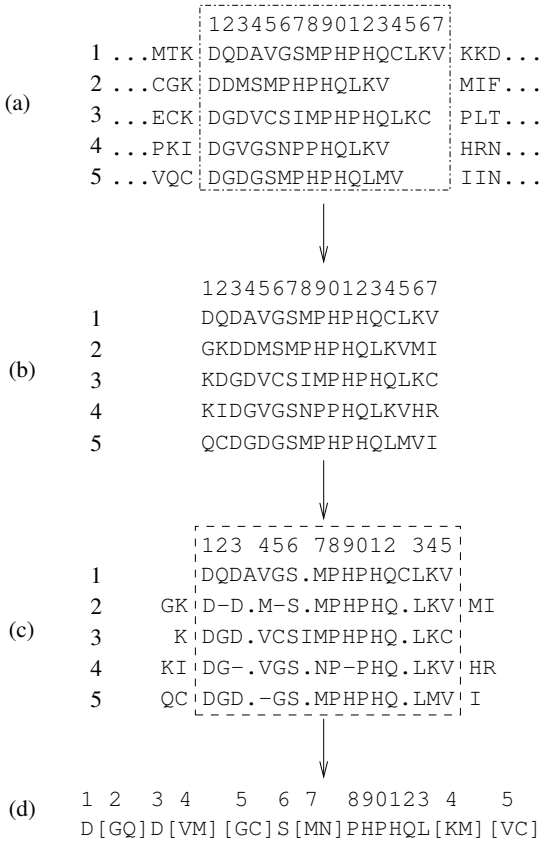

Figure 2: The aligned motif patterns with at most 2 insertions/deletions ('$-$' indicates a deletion in this sequence and '.' indicates an insertion in aother sequence). (a) Protein sequence segments. (b) Test patterns of length $= 17$. (c) Alignment. (d) Motif expression, length $= 15$.

2.1 Encoding of the Input Patterns

In the preceding section, we assume that there are at most two consecutive letter insertions or deletions in a motif pattern. Under this assumption, we analyze a group of motif patterns from protein sequences, e.g., patterns in Figure 2. We observe that in these patterns, when they are aligned according to identified motifs, column i of a pattern can be aligned to one of the columns in the range from $i-4$ to $i+4$ of other patterns. The two aligned columns can have a maximum index difference of 4. The extreme case happens when one of the pattern has two insertions and the other pattern has two deletions, e.g., pattern #1 and pattern #2 in Figure 2. Before we align the patterns, pattern #1 has insertions at column 4 (letter "A") and column 14 (letter "C"), while pattern #2 has deletions at column 2 (between two letter "D"s) and column 4 (between letters "M" and "S"). We notice that columns 15 to 17 of pattern #1 should be aligned to columns 11 to 13 of pattern #2. In the illustration of Figure 2, we assume the length of motif to be 15. Since we allow each appearance of motif to have a maximum of 2 letter insertions, we can choose test subsequences of length 17.

Test subsequences are obtained from the given protein sequences. We call these test subsequences input patterns to the self-organizing neural network. We use a sliding window to get all input patterns. Let L be the length of a protein sequence, and let m be the length of the motif to be identified. The length of the sliding window will be $m' = m + 2$. Placing the sliding window at the beginning of the sequence, we get the first input pattern with length of $m+2$. Shift the window one column at a time to get all the input patterns. The total number of input patterns we get from the sequence will be $L - m' + 1 = L - m - 1$. Due to the effects of sliding window, when input patterns are finally aligned, the maximum column index difference becomes 2 (a positive difference of 2 or a negative difference of 2) between any pairs of input patterns. In order to define the pairwise similarity value (or distance) between a pair of input patterns that may have column index difference up to 2 due to insertions and deletions, we put every 3 consecutive letters into a group. We will now consider the distance between groups of letters. Comparing between two appearances of a motif, every group of three consecutive letters in one appearance will have at least one letter in common with the corresponding group in the other appearance, assuming a maximum of 2 letter insertions or deletions when there is no mutations. These groups can be obtained by applying sliding window of length 3 to each input pattern. For each input pattern with length of m', we will get $m = m' - 2$ groups of letters, each group with length of 3. For example, for input pattern #2 of Figure 2 (b), we get the following groups of 3 letters:
{ GKD, KDD, DDM, DMS, MSM, SMP, MPH, PHP, HPH, PHQ, HQL, QLK, LKV, KVM, VMI }.
Next, we will encode all the input patterns using binary digits. Each protein letter will be encoded using a 20-digit binary vector, with one digit flipped from $\{1, 1, ..., 1, 0, 0, ..., 0\}$ (ten 1s followed by ten 0s). This coding strategy guarantees that the coded vectors of any two different protein letters have exactly two digits that are different. Encoding a protein pattern means encoding all the letter groups of that pattern. For example, each of the above groups of 3 letters will be encoded by a $60 = 20 \times 3$ digit binary code.

2.2 Pairwise Distance Between Input Patterns

First we study the 3-letter groups from corresponding positions of two input patterns. We want to determine the distance or the similarity value between two 3-letter groups. We consider two possibilities. The first one is that there is one insertion or one deletion in either of the groups. If this is the case, a sub-group of 2 letters from one of the groups may match a 2-letter sub-group from the other group. All these 2-letter sub-groups should follow the same order as they appear in the 3-letter groups. For example, 2-letter sub-groups are AC, CG or AG from ACG. Such 2-letter sub-groups from a 3-letter group will always have 3 possibilities. Second, there are two insertions or two deletions in either one of the 3-letter groups. In this case, as long as there is a common letter appearing in both groups, we would grant a relatively smaller pairwise distance value between these 3-letter groups. The above strategy can be expressed by the following mathematical description. Each input pattern (length $m' = m + 2$) is converted into a test pattern with m 3-letter groups. After encoding, each input pattern will be in the form of binary vectors $P = \{p_1, p_2, ..., p_m\}$, where the length of each binary vector p_i is 60. Each binary vector p_i has 3 letters. Each binary vector can be expressed as $p_{i,1-20}$, $p_{i,21-40}$ and $p_{i,41-60}$. Now we form sub-groups using two of the three vectors following the same order as they have in the 60-digit 3-letter group vector. We will get 3 sub-groups and we denote them as

$$p'_i = \big(p'_{i[1]}, p'_{i[2]}, p'_{i[3]}\big) = \big(\{p_{i,1-20}p_{i,21-40}\}, \{p_{i,1-20}p_{i,41-60}\}, \{p_{i,21-40}p_{i,41-60}\}\big).$$

The distance between any two given vectors p_i and q_i will be defined as

$$\begin{aligned} d_i = \min_{j,k,l \in \{1,2,3\}, j \neq k \neq l} \Bigg(& \sum_{r=1}^{40} |p'_{i[1],r} - q'_{i[j],r}| \\ & + \sum_{r=1}^{40} |p'_{i[2],r} - q'_{i[k],r}| + \sum_{r=1}^{40} |p'_{i[3],r} - q'_{i[l],r}| \Bigg) \\ & + \sum_{k=1}^{20} \big| (p_{i,k} + p_{i,k+20} + p_{i,k+40}) \\ & - (q_{i,k} + q_{i,k+20} + q_{i,k+40}) \big|. \end{aligned} \tag{2}$$

Let $\mathcal{D}(P, Q)$ denotes the pairwise distance of any two input patterns P and Q, each with $m + 2$ letters. We get

$$\mathcal{D}(P, Q) = \sum_{i=1}^{m} d_i. \tag{3}$$

Equation (2) shows the distance between two 3-letter groups. In the equation, the first part reflects the minimum sum of distances between the sub-groups obtained from both 3-letter groups. For each 3-letter group, we get 3 possible sub-groups. Then there are a total of $6 = 3!$ possible cases of putting together any two sub-groups from each group. The second part of (2) is an absolute value that reflects the sum of distances between single letters from each group. For example, if we want to calculate the pairwise distance between two 3-letter groups ACG and tag, the first part of (2) reflects the sum of the distances between AG and ag, CG and tg, and AC and ta. The second part reflects the sum of the distances between A and a, G and g, and C and t. We use this strategy because the given two 3-letter groups are no longer alignable due to insertions and deletions. The present definition of distance in (2) gives a good representation of the similarity between a pair of 3-letter groups.

2.3 Neural Network Structure

This subsection describes the structure of our self-organizing neural network for motif discovery. The basic structure forms the subnetworks used in our self-organizing neural networks and contains two layers, i.e., an input layer and an output layer. The number of output neurons of a subnetwork is the same as the number of categories classified by this subnetwork. The number of input neurons is determined by the projected length of motifs after encoding, e.g., $m' \times 20$, where $m' = m + 2$ and m is the length of projected motifs. The input patterns are obtained from the given protein sequences by taking all subsequences with the same length of m'. Each output neuron represents a category that has been classified by a subnetwork and each output category is represented by the connection weights from all input neurons to the corresponding output neuron. Subnetworks perform the function of classification in a hierarchical manner. The first subnetwork is placed at the top level and it performs a very rough classification, e.g., dividing the input space into 4–8 categories. The second subnetwork is placed at the next level and it usually divides the input space into 16–32 categories, which indicates a slightly more detailed classification of the input space. The last subnetwork in our self-organizing neural network will be placed at the lowest level and it classifies all the input patterns into either a motif or a non-motif category with one or a few patterns. Typically, the number of output neurons will be very large for the last subnetwork and gradually reduced to a small number for the first subnetwork. Figure 3 shows the structure of our self-organizing neural network with three subnetworks. In the structure shown in the figure, there are four input neurons and three subnetworks. The first subnetwork has 2 output neurons, the second subnetwork has 3 output neurons, and the third subnetwork has 4 output neurons. Each of the output neurons represents a category that has been created and it is represented by the connection weights to the output neuron. The output category α of the first subnetwork contains two patterns (a and b) and the other contains one pattern (c). The output category a of the second subnetwork contains two patterns (1 and 2) and the other two categories each contains just one pattern. The output categories 1 and 2 of the third subnetwork represent two motifs while categories 3 and 4 are not motifs (if we desire to have at least three patterns for each motif identified).

The inputs to each subnetwork are binary codes from protein patterns. We set the number of input neurons to be $M = m' \times 60$. The output neurons of each subnetwork are categories classified by that subnetwork. Our target motif sets will be obtained from the outputs of the lowest subnetwork. Outputs of the higher subnetwork are categories with coarser classification. The weights connected to each category are calculated as an m-dimensional vector indicating the center of that category.

2.4 Rules for Weight Update and Output Node Creation

When an input pattern is presented to our self-organizing neural network, it will be either classified to an existing output category or put into a new category by every subnetwork. An output category of a lower level subnetwork is said to

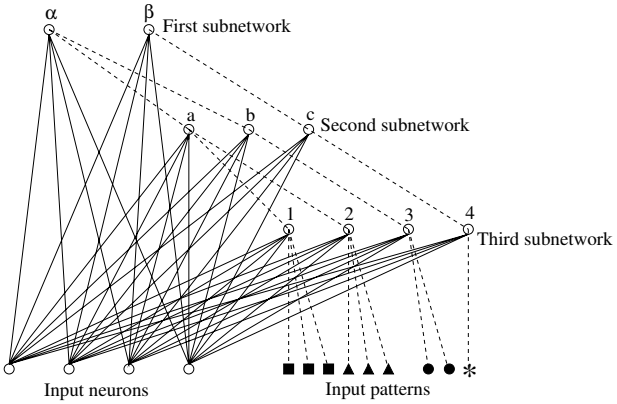

Figure 3: Structure of the self-organizing neural networks

belong to an output category of a higher level subnetwork if one or more input patterns are classified to both of these two output categories. The connection weights for each category of the last subnetwork (at the lowest level) are calculated as the center of the category, i.e., the geometric center of all input patterns that are currently classified into the category associated with the corresponding output neuron. The connection weights for an output category of all other subnetworks (except the last subnetwork) are calculated as the geometric center of all categories from the immediate lower level of subnetwork that belong to this category.

When a new input pattern is introduced to a subnetwork, its classification to an output neuron of every subnetwork involves the following three steps.

1) A similarity test will be performed between the new input pattern and the output neurons of the subnetwork at each level. At the top level, all the output categories will be tested. At the next level, only those neurons classified to the winner of the upper level will be tested, so on and so forth. By this means only a small number of categories will need to be tested to save computation time. These groups of neurons form a tree of classification as shown in Figure 3. For the network example shown in Figure 3, an input pattern will be first compared to the two categories at the top level. At the next level, it will be either compared to $\{a, b\}$ or $\{c\}$ depending on which of the two output categories at the first level becomes the winning category. The pairwise distances between the input pattern and patterns in each category are calculated according to (3). The largest pairwise distance within each category is determined and compared to a threshold value. If the largest pairwise distance of a category is smaller than the threshold value, we will regard this category as a winning category.
2) There may be more than one category that wins in the first step, but only one of them will be the final winner at each level. In this step we perform a distance test between the input pattern and the center of each winning category in Step 1. This test is calculated by comparing the input pattern with the connection weights to each neuron in all winning categories. The minimum of these distances is determined and thus a final winning category is also determined. This step works similarly to the winner-takes-all networks (Haykin, 1999).
3) If in the preceding step, there are still more than one winning category (categories have the same minimum distance from the input pattern to the center), the final winning category will be chosen as the one that has the most number of members.

Assume that there are a total of L subnetworks for $l = 1, 2, \cdots, L$. Assume that there are M input neurons and the lth subnetwork has N_l output neurons. The patterns obtained from the given protein sequences are used as input sequences to each subnetwork of our self-organizing neural network. The outputs of the last subnetwork correspond to classifications of all subsequences into motifs and non-motif categories.

We denote the input patterns as x^i, $i = 1, 2, \cdots$. Suppose that t input patterns have been presented to the network and have been classified. When a new input pattern, i.e., the $(t+1)$st pattern x^{t+1}, is introduced to the network, we do the similarity test in the three steps described above. In the lth subnetwork, if output neuron q is one of the categories that have been classified into the winning category in the $(l-1)$st subnetwork, the pairwise distance calculations in category q of the lth subnetwork are described as

$$d_j^l = \mathcal{D}(x^{t+1}, e_j^{l+1}),\ j = 1, 2, \cdots, p_q,$$

where

$$e_j^{l+1} = \begin{cases} x^j, & \text{if } l = L \text{ and } x^j \text{ belongs to the} \\ & \text{category } q \text{ of the lowest level,} \\ w_j^{l+1}, & \text{if } 1 \le l < L \text{ and } w_j^{l+1} \text{ belongs} \\ & \text{to the category } q \text{ of the } l\text{th level,} \end{cases} \tag{4}$$

$\mathcal{D}$ is defined in (3) and p_q is the number of patterns in category q. We then perform the following threshold tests. If

$$\max_{1\le j\le p_q}\{d_j^l\} < \rho_l, \tag{5}$$

this new input pattern will be considered to match the category q of the lth subnetwork. The threshold value ρ_l in (5) takes different values for different levels of subnetworks. We note that all pairwise distances in this category will be less than the threshold ρ_l if (5) is satisfied for the new input pattern since all other patterns are previously classified into this category using the same threshold value.

If there is only one category wins the similarity test, this new input pattern will be classified into the category q of the lth subnetwork. Otherwise we need to perform further similarity test described in Step 2. We calculate the distance from the new input pattern to the center of each winning category and then choose the winner that has the smallest distance to the new input pattern. If there are more than one category having the same smallest distance, we will pick the category that has the most number of patterns in it as the final winning category.

If there is a category q wins in (5), this new input pattern will be classified into the winning category of the lth subnetwork. Otherwise, the new input pattern cannot be classified into any existing category at this level.

We describe in the following some more details about our calculation procedure.

a) We start from the top level, i.e., the first subnetwork, and work down the levels one by one, when classifying a new input pattern. After a winning category is determined at the lth level, the input pattern will only be compared to those patterns at the $(l+1)$st level that are classified to belong to the winning category at the lth level.

b) If the threshold test in (5) is successful for $l = 1, 2, \cdots, L$, we perform the following updates for the Lth subnetwork:

$$\begin{aligned} w_{kq}^L &:= \frac{1}{p_q^L+1}\sum_{j=1}^{p_q^L+1} x_k^j \\ &= \frac{1}{p_q^L+1}\left[p_q^L w_{kq}^L + x_k^{t+1}\right], \\ p_q^L &:= p_q^L + 1, \end{aligned}$$

where $k = 1, 2, \cdots, M$ and the (p_q^L+1)st term in $\sum_{j=1}^{p_q^L+1} x_k^j$ indicates the kth component of the new input pattern x^{t+1}. We perform the following updates for the rest of subnetworks:

$$\begin{aligned} w_{kq}^l &:= \frac{1}{p_q^L}\sum_{j=1}^{p_q^L} w_{kj}^{l+1},\ k = 1, 2, \cdots, M, \\ &\text{for } l = L-1, L-2, \cdots, 2, 1. \end{aligned}$$

c) If the threshold test in (5) is successful for $l = 1, 2, \cdots, L_1$, where $L_1 < L$, we will add an output neuron to subnetworks $L_1+1, L_1+2, \cdots, L$. Each of these newly added categories will contain only one pattern and the weights of the new categories are chosen as

$$\begin{aligned} w_{kn}^l = x_k^{t+1},\ k = 1, 2, \cdots, M,\ n = N_l + 1, \\ \text{for } l = L_1+1, L_1+2, \cdots, L. \end{aligned}$$

We also update the number of output neurons for these subnetworks as

$$N_l := N_l + 1,\ p_{N_l} = 1,\ l = L_1+1, L_1+2, \cdots, L.$$

In this case, it is not necessary to perform threshold tests for subnetworks $L_1+1, L_1+2, \cdots, L$ anymore. For subnetworks $1, 2, \cdots, L_1$, we will perform the following updates:

$$\begin{aligned} p_q^{L_1} &:= p_q^{L_1} + 1 \\ w_{kq}^l &:= \frac{1}{p_q^l}\sum_{j=1}^{p_q^l} w_{kj}^{l+1},\ k = 1, 2, \cdots, M, \\ &\text{for } l = L_1, L_1-1, \cdots, 2, 1. \end{aligned}$$

2.5 Optimal Choice of Consecutive Patterns

Consecutive patterns are input patterns from the same protein sequence and with their location differences being smaller than the desired length of motif. For example, $x^t, x^{t+1}, x^{t+2}, \cdots$ are consecutive patterns. According to the pairwise distance defined earlier, consecutive patterns can win the similarity test in the same category. Apparently, they should not be included in the same category. We need to determine which of the consecutive patterns should really be in the category. During the classification procedure of a new input pattern, if there is a consecutive pattern to this new input pattern in a category, we need to determine whether this new input pattern matches better with the category than its consecutive pattern that is already in the category. To achieve this, we use the average pairwise distance within a category. When we find a consecutive pattern in the category to be checked, we make pairwise similarity tests between the new input pattern and all patterns in the category except its consecutive one. If (5) is satisfied for this category, we need to examine whether the new input pattern is a better choice than its consecutive pattern in the category. We compare two average pairwise distance values. The first one is the average of all the pairwise distance values from the new input to all the patterns in the category except the consecutive pattern. The second one is the average of all the pairwise distance values from the consecutive pattern to the rest of patterns in the category. For example if the qth category in the lth subnetwork wins, meanwhile the σth pattern in this category, x^σ is a consecutive pattern to the new input pattern x^{t+1} ($\sigma \neq t+1$), we use the following equation to calculate the average pairwise distance values for x^k, $k = \sigma, t+1$,

$$d_q^{\text{avg}}(x^k) = \frac{1}{p_q - 1} \sum_{j=1, j\neq\sigma}^{p_q} \mathcal{D}(x^k, x^j). \tag{6}$$

If $d_q^{\text{avg}}(x^{t+1}) < d_q^{\text{avg}}(x^\sigma)$, we choose the new pattern as a replacement of the consecutive pattern classified earlier. We delete the old pattern from the category and add the new one. Change the weights of the winning category accordingly. If $d_q^{\text{avg}}(x^{t+1}) > d_q^{\text{avg}}(x^\sigma)$, we skip this category and do the similarity test for the rest of categories.

2.6 Order Randomization of Input Patterns

In our approach, the order in which the input patterns are presented to the network will be chosen randomly. To avoid the problem of missing classifications of some patterns, we will perform multiple trials with randomly selected order of presentation for the whole set of input patterns. After the learning procedure of each input cycle, we may get a certain number of output categories in the lowest subnetwork. Some of these categories are kept for the next cycle. For each category, the number of patterns classified in the category decides whether this category will be kept. If the following condition for the qth category is satisfied,

$$p_q^L \geq \lambda,$$

we will keep this category, where p_q^L is the number of patterns in the qth category of the Lth subnetwork and λ is determined by the user, e.g., $\lambda = 3$.

After picking out these categories, we use them to initialize the network for the next cycle in the lowest subnetwork. This lowest subnetwork will be used to initialize all other subnetworks. Following the three steps we mentioned in the last section, we build categories from subnetwork to subnetwork until the initialization of the network for the new input cycle is done.

In each cycle after the first one, we validate every input pattern before we apply it to the network. It is necessary to determine if this pattern is already classified in an existing category. If it is, we will skip this input pattern. We do this in order to prevent classifying the same pattern into more than one category or to the same category twice. From simulation results, we find that usually two or three trials with order randomization are enough for the motif discoveries in protein sequences.

2.7 Alignment of Motif Patterns

The outputs of the lowest subnetwork are the target motif sets or non-motif categories. Every motif set contains a group of patterns. These patterns are with the length of $m+2$. They are not yet the final aligned motif sets with insertions/deletions. We need to align them and to get a consensus form from them. At the current stage of our research, the alignment of motif patterns will be done manually for each set identified using our algorithm. Due to the length of 8–20 letters for projected motifs, their alignment can easily be performed by hand once they have been classified into to a category.

3 Simulation Results

Example 1 We first test our algorithm for motif identifying in randomly generated protein sequences. We randomly generate 20 protein sequences with average length of 300 letters in each sequence. We generate a motif set of 20 appearances

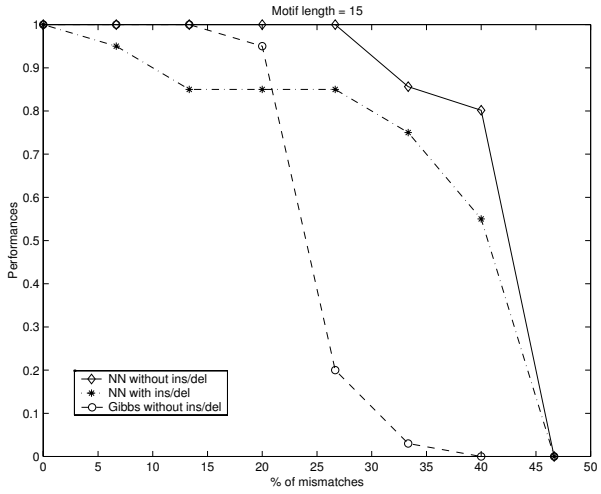

Figure 4: Comparison result for motif length = 15

with insertions, deletions and mutations at random columns in each appearance. We insert these appearances into random locations of the protein sequences. It is not necessary that each sequence has exactly one motif appearance. Some sequences may have more than one appearance and others may have none. The length of the motif is fixed. The number of mutations in the appearance are not fixed. Let R denote the motif set we generated and T denote the motif set identified by the self-organizing neural network. Then the performance of the motif discovery algorithm is defined as

$$\mathrm{PER} = \frac{|R \cap T|}{|R \cup T|}. \tag{7}$$

Figure 4 shows the comparison of three cases of identifying motifs with length of 15. The horizontal axis represents the percentage of mismatch of the motifs (i.e., ϵ/M, where ϵ is the number of letters that is tolerable in the representation of a motif), and the vertical axis indicates the average performance of 8 simulation runs defined above. In this simulation, we fix the distribution of the number of insertions and deletions. 30 percent of the motif appearances we generated have two insertions or deletions or their combination. 40 percent of them have only one insertion or deletion. The rest appearances do not have insertion or deletion. Besides these insertions and deletions, all the appearances have a certain number of columns mutated according to the horizontal axis of the figure. In Figure 4, we compare our results to that of the Gibbs algorithm.

From the figure we can see that compared with the Gibbs algorithm, our algorithm has much better performance. Let (15, 4) denote the case of setting motif length to 15 and number of mismatch columns to 4. Our neural network algorithm performs well and finds nearly all the patterns generated in the case of (15, 4) while the Gibbs algorithm finds only 20% of all patterns. In the case of having insertions and deletions, for (15,6), our algorithm finds 50% of all patterns while the Gibbs algorithm missed all patterns.

Example 2 In this example we will apply our algorithm to motif discovery problem in DNA Repair Protein RAD51 homolog protein sequences. The 10 samples are collected from Swiss-Prot Genes Library. Their names and descriptions are listed in Table 2. We apply these 10 sequences to our network, and get 17 motif sets that have at least 5 appearances in each set. A typical output of the aligned motif set is shown in Table 3.

In the table, a motif set with 11 appearances is listed, along with a consensus sequence. All the appearances are aligned in the format we introduced in Figure 2 (c). These motif appearances can be located by the sequence and column indexes listed in the table. The total number of mutations of each appearance to the consensus sequence is also shown in the table. In the last column of the table, we list the total number of insertions and deletions in each appearance.

4 Conclusions

In this paper, we studied the problem of motif discovery in protein sequences with insertions and deletions. We developed a self-organizing neural network structure with multiple levels of subnetworks to make an intelligent classification of the subsequences obtained from the protein sequences. We maintain a low computational complexity through the use of the multi-level structure so that each subsequence's classification is performed with respect to a small subspace of the whole input space. Our algorithm can find motifs with up to 2 insertions, deletions or their combinations. The simulation results show that the performance of our algorithm is more accurate and costs less computation than existing algorithms.

Table 2: Entry name, accession number and length of DNA repair protein RAD51 sequences

Index	Protein Name	Accession number	Length
1	RA51_CHICK	P37383	339
2	RA51_CRIGR	P70099	339
3	RA51_DROME	Q27297	336
4	RA51_HUMAN	Q06609	339
5	RA51_LYCES	Q40134	342
6	RA51_MOUSE	Q08297	339
7	RA51_RABIT	O77507	339
8	RA51_SCHPO	P36601	365
9	RA51_USTMA	Q99133	339
10	RA51_YEAST	P25454	400

Acknowledgements

D. Liu was supported in part by Open Research Project Grant (ORP-0501) from KLCSIS-IA-CAS. B. DasGupta was supported in part by NSF grants CCR-0296041, CCR-0206795, CCR-0208749 and IIS-0346973.

References

Bailey, T. L., & Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, *21*(1–2), 51–80.

Attwood, T. K., & Parry-Smith, D. J. (1999). *Introduction to Bioinformatics*, (pp. 132–144), New York: Prentice Hall.

Bailey, T. L., & Gribskov, M. (1998). Combining evidence using p-values: Application to sequence homology searches. *Bioinformatics*, *14*(1), 48–54.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (pp. 443–483), 2nd edition, Upper Saddle River, NJ: Prentice Hall.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., & Wootton, J. C. (1993). Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, *262*(5131), 208–214.

Linhart, C, & Shamir, R. (2002). The degenerate primer design problem. *Bioinformatics*, *18*(Suppl. 1), S172–S181.

Liu, D., Xiong, X., & DasGupta, B. (2005). A self-organizing neural network structure for motif identification in DNA sequences. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, 129–134, Tucson, AZ.

Pevzner, P. A., & Sze, S. H. (2000). Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 269–278, San Diego, CA.

Thompson, J. D., Higgins, D. G., & Gibson, T. J. (1994). CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, *22*(22), 4673–4680.

Xie, J., Li, K. C., & Bina, M. (2004). A Bayesian insertion/deletion algorithm for distant protein motif searching via entropy filtering. *Journal of the American Statistical Association*, *99*(466), 409–420.

Xiong, X., Liu, D., & Zhang, H. (2005). A self-organizing neural network approach for the identification of motifs with insertions and deletions in protein sequence. In *Proceedings of the International Joint Conference on Neural Networks*, Montreal, Canada.

Table 3: Patterns and their locations in the RA_51 protein sequences

Patterns	Contents	Sequence	Location	Mutations	Insertions + Deletions
Consensus	LQGG.IETGSITELF.GEF	-	-	-	-
1	LQGG.IETGSITELF.GEF	1	113	0	0
2	LQGG.IETGSITEMF.GEF	2	113	1	0
3	L-GG.IETGSITEIF.GEF	3	110	1	1
4	LQGG.IETGSITEMF.GEF	4	113	1	0
5	LEGG.IETGSITEIFYGEF	5	116	2	1
6	LQGG.IETGSITEMF.GEF	6	113	1	0
7	LQGG.IETGSITEMF.GEF	7	113	1	0
8	LQGG.VETGSITELF.GEF	8	135	1	0
9	L-GG.METGSITEL-.GEF	9	113	1	2
10	L-GGKVETGSITELF.GEF	10	171	1	2

('−' indicates a deletion in this sequence and '.' indicates an insertion in aother sequence.)