A Self-Organizing Neural Network Structure for Motif Identification in DNA Sequences

Derong Liu, Xiaoxu Xiong
Department of Electrical and Computer Engineering
University of Illinois at Chicago
Chicago, IL 60607, USA
Tel: (312) 355-4475, Fax: (312) 996-6465
E-mail: dliu@ece.uic.edu, xxiong@cil.ece.uic.edu

Bhaskar DasGupta
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
Tel: (312) 355-1319, Fax: (312) 413-0024
E-mail: dasgupta@cs.uic.edu

***Abstract*— In this paper, we study the problem of subtle signal discoveries in unaligned DNA and protein sequences. Motifs, also known as approximate common substrings, are good examples of subtle signals in DNA and protein sequences. The problem of motif identification in DNA and protein sequences has been studied for many years in the literature. Major hurdles at this point include computational complexity and reliability of the searching algorithms. We will develop a self-organizing neural network for solving the problem of motif identification in DNA and protein sequences. Our network contains several layers with each layer performing classifications at different level. The top layer divide the input space into a small number of regions and the bottom layer classifies all input patterns into motifs and non-motif patterns. Depending on the number of input patterns to be classified, several layers between the top layer and the bottom layer are needed to perform intermediate classification. We maintain a low computational complexity through the use of the layered structure so that each pattern's classification is performed with respect to a small subspace of the whole input space. We also maintain a high reliability using our self-organizing neural network since the network will grow as needed to make sure all input patterns are considered and are given the same amount of attention. Finally, simulation results show that our algorithm significantly outperforms existing algorithms, especially in the reliability aspect. Our algorithm can identify motifs with higher accuracy than existing algorithms.**

Keywords: Subtle signal, motif finding, self-organize, neural network, unsupervised, DNA.

I. Introduction

DNA, RNA and protein sequences can be thought of as being composed of motifs interspersed in relatively unconstrained sequence. A motif is a short stretch of a molecule that forms a highly constrained sequence [2]. 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 position 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 a more complicated expression to show all possible letters for each position of a motif. For example, the expression

$$A-[CG]-T-T-[AC]-[TCG]-A-A \quad (1)$$

indicates that $AGTTCTAA$ and $ACTTAGAA$ are two of the possible occurrences.
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 at each position in the motif [3]. For example, Table I 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 position of the motif, but also shows the probability of their appearances. For example, the sixth position of this motif will have letters C, G, and T appearing with probabilities of 20%, 30%, and 50%, respectively.

TABLE I
FREQUENCY OF EACH LETTER APPEARS IN EVERY POSITION 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

Generally speaking, the subtle signal finding problem in DNA sequences can be described as follows: Given a set of unaligned DNA or protein sequences, project the length of motifs and locate all motifs with the projected length that these sequences hold. It is not necessary for all the sequences to have the same motif. Some sequences may have more than one repetitions of a motif and some motifs

0-7803-8812-7/05/$20.00

may not show up in every sequence. The appearances of the same motif in the sequences are not necessarily the same. A subsequence is determined to be a motif if it matches a possible appearance indicated by (1) or by the matrix representation in Table I. Obviously, information provided in Table I is more than that in (1). Here the frequency or probability of letters in each position of a motif is in $[0, 1]$. Usually the frequency of the letter that appeared most frequently should be larger than 40% [27].

References [10], [21], [25] presented an unsupervised learning method for finding contiguous motifs. This kind of motifs has some biological properties of interest such as being DNA binding sites for a regulatory protein. The work in [10], [21], [25] showed that unsupervised learning method is a good approach for dealing with the problem of finding motifs. An algorithm called MEME is proposed in [1] for identifying contiguous motifs. This algorithms is an extension to the expectation maximization algorithm for motif finding. In this paper, we will develop an algorithm based on a new structure of self-organizing neural networks and we will compare the performance of our algorithm with that of [1]. The signal finding problem described in [27] projects the length of motifs as well as the maximum number of letters that can be mismatched in a pattern. In this case, the target patterns to be found are described by a given length and by how many letters that can be mismatched.

II. Self-Organizing Neural Networks for Motif Identification

A. Subsequences and Encoding

We consider the case where all motifs to be identified from a given set of DNA or protein sequences have the same length. In general, the consensus sequence of a motif and the motif itself are not known *a priori* and we have to obtain them by using identification algorithms. What one obtains after the use of identification algorithms are specific appearances of a motif, usually with a few mismatched letter positions comparing to the motif consensus sequence. For a given set of DNA or protein sequences, in order to identify motifs in these sequences, we have to specify the maximum number of letter mismatches that can be tolerated (comparing to the consensus form) in addition to projecting the length of motifs to be found.

Test patterns, which we call input sequences or input patterns, can be obtained from the given set of DNA or protein sequences once the projected length of motifs is given. Figure 1 shows a sketch of how input patterns are obtained from a DNA sequence. In the figure, the projected length of motifs is seven. All subsequences of seven connected letters obtained using a sliding window (see Figure 1) from the given DNA or protein sequences will form the set of input patterns. For a DNA sequence

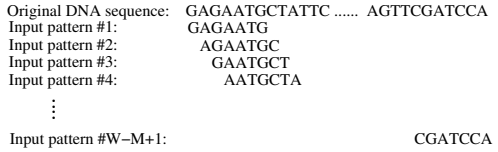

Fig. 1. An illustration on how to obtain input patterns ($M = 7$) from a given DNA sequence

TABLE II
Encoder table for DNA letters

Standard	1	1	0	0
A	1	1	0	1
C	1	1	1	0
G	1	0	0	0
T	0	1	0	0

of length W, we can obtain $W - M + 1$ input patterns if the projected length of motifs is M.

Letters used in DNA or protein sequences will be encoded using binary numbers. All letters will be encoded using binary code with the same length, for example, four for DNA and RNA sequences and 20 for protein sequences. Table II shows an example of binary codes designed for DNA sequences. There are four letters in this case and each letter is encoded by flipping one bit of the standard code '1 1 0 0'. Letters coded this way will have exactly the same Hamming distance between any pair of letters which is an important desired feature. Also, the scheme shown in Table II can also guarantee that ones and zeros will appear on average the same number of times.

B. A New Structure of Self-Organizing Neural Networks

This subsection describes the structure of our self-organizing neural networks for subtle signal 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 and the number of input neurons equals the projected length of motifs. The input patterns are obtained from the given DNA or protein sequences by taking all subsequences with the same length as the length of projected motifs (often in terms of the number of binary digits after encoding). 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 functions of classification in a hierarchical manner. The first subnetwork is placed at the top level and it performs a very rough classification, e.g., divide the input space into 4–8 categories. The second subnetwork is placed at the next level and it usually divides the input space into

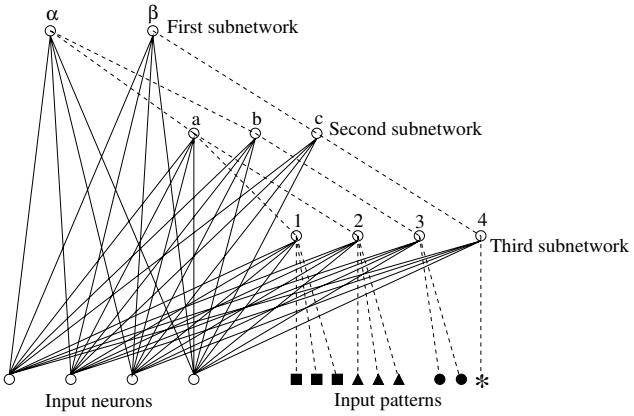

Fig. 2. Structure of the self-organizing neural networks

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 large for the last subnetwork and gradually reduced to a small number for the first subnetwork. Figure 2 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 appearances for each motif identified).

C. Rules for Weight Update and Output Node Creation

When an input pattern is applied to our self-organizing neural network, it will be classified to an output category by every subnetwork. An output category of a lower level subnetwork is said to belong to an output category of a higher level subnetwork if one or more input patterns are classified to belong to 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 lower level of subnetwork that belong to this category.

When a new input pattern is applied to a subnetwork, its classification to an output category of every subnetwork involves the following two steps.

1) The distance between the input pattern and each output category is calculated by comparing the input pattern with the connection weights from the input neurons to that category. The minimum of these distances is determined and thus a winning category is also determined. This step works similarly to the winner-take-all networks [16]. These winning neurons form the tree of classification as in Figure 2. For the example network shown in Figure 2, an input pattern will be first compared to the two categories $\{\alpha\}$ and $\{\beta\}$ at the first 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.
2) Within the winning category, the similarity of all patterns in this category including the new pattern will be calculated and compared to a threshold value. If the similarity value is less than the threshold, the new pattern will be classified into the winning category. Otherwise, the new pattern cannot be classified into the winning category.

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 form the given DNA or protein sequences are used as input sequences to each subnetwork of our self-organizing neural network and the outputs of the last subnetwork correspond to classifications of all subsequences into motifs and non-motif categories. The projected length of motif sequences possibly existing in the input sequences is the same as M.

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 lth subnetwork, the distances from the new input pattern to those categories of the lth subnetwork that belong to the $(l-1)$st winning category W_q^{l-1} is calculated as

$$y_n^l = \sum_{m=1}^{M} |x_m^{t+1} - w_{mn}^l|, \text{ for } n \in W_q^{l-1}$$

where x_m^{t+1} is the mth component of the input pattern x^{t+1} and w_{mn}^l is the connection weight of the lth subnetwork from the mth input neuron to the nth output neuron after

the presentation of the tth input sequence. Denote

$$y_q^l = \min_{n \in W_q^{l-1}} \{y_n^l\}$$

i.e., the qth output category of the lth subnetwork is the winning category that has the smallest distance to the new input pattern. Assume that the qth output category of the lth subnetwork contains p_q^l patterns from the $(l-1)$st subnetwork. Within this winning category q, we will calculate the similarity value of all the $p_q^l + 1$ patterns including the new input pattern. The similarity value of a group of patterns is calculated as the maximum of the pairwise distance between all pairs of patterns in the group.

For the winning category q determined above, we calculate the distances from the new input pattern to all other patterns in the category as

$$d_j^l = \sum_{m=1}^{M} \left| x_m^{t+1} - e_{mj}^{l+1} \right|, \ j = 1, 2, \cdots, p_q,$$

where

$$e_{mj}^{l+1} = \begin{cases} x_m^j, & \text{if } l = L \text{ and } x_m^j \text{ belongsto the} \\ & \text{category } q \text{ of the } (l-1)\text{st level} \\ w_{mj}^{l+1}, & \text{if } 1 \le l < L \text{ and } w_{mj}^{l+1} \text{ belongs to} \\ & \text{thecategory } q \text{ of the } l\text{th level.} \end{cases} \tag{2}$$

We then perform the following threshold tests. If

$$\max_{1 \le j \le p_q} \{d_j^l\} < \rho_l \tag{3}$$

then this new input pattern will be classified into the category q of the lth subnetwork. Otherwise, the new input pattern cannot be classified into any existing category at this level. The threshold value ρ_l in (3) will be determined later and it takes different values for different subnetwork. We note that all pairwise distances in this category will be less than the threshold ρ_l if (3) is satisfied for the new input pattern since all other patterns are previously classified into this category using the same threshold test.

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 level one by one, when classifying a new input pattern. After a winning category has been 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 and are denoted by W_q^l.

b) If the threshold tests in (3) are successful for $l = 1, 2, \cdots, L$, we perform the following updates for the Lth subnetwork:

$$w_{mq}^L := \frac{1}{p_q^L + 1} \sum_{j=1}^{p_q^L + 1} x_m^j$$

$$= \frac{1}{p_q^L + 1} \left[p_q^L \times w_{mq}^L + x_m^{t+1} \right], \ m = 1, 2, \cdots, M,$$

$$p_q^L := p_q^L + 1,$$

where $x^{p_q^L+1}$ indicates the new input pattern x^{t+1} for convenience. We perform the following updates for the rest of subnetworks:

$$w_{mq}^l := \frac{1}{p_q^L} \sum_{j=1}^{p_q^L} w_{mj}^{l+1},$$

$$m = 1, 2, \cdots, M, \ l = L-1, L-2, \cdots, 2, 1.$$

c) If the threshold tests in (3) are 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

$$w_{mn}^l = x_m^{t+1}, \ m = 1, 2, \cdots, M,$$

$$n = N_l + 1, \ l = L_1 + 1, L_1 + 2, \cdots, L.$$

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

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

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:

$$p_q^{L_1} := p_q^{L_1} + 1$$

$$w_{mq}^l := \frac{1}{p_q^l} \sum_{j=1}^{p_q^l} w_{mj}^{l+1},$$

$$m = 1, 2, \cdots, M, \ l = L_1, L_1 - 1, \cdots, 2, 1.$$

D. Order Randomization of Input Patterns

The classification of an input pattern to a category using the present self-organizing neural networks will be affected by the order in which input patterns are presented to the network. Since the center of each category is affected by the locations of all its members, when a new member is introduced, or the location of an old member is updated, the center of the category under consideration will change. Figure 3(a) shows a case when an input pattern fails to be classified into a category to which it most likely belongs. The area covered by the category C is not all covered by the larger category Q. When the new input pattern is presented, it will first try to see if it can be classified to the category represented by the large circle Q. If the matching fails, then it stops searching the lower level categories. However, the center of circle Q may be different for

different trials depending on the order in which input patterns are presented. If the new input pattern illustrated in Figure 3 is presented earlier, it might have the opportunity to be classified to category Q. Figure 3(b) shows such a case where the center of circle Q is different.

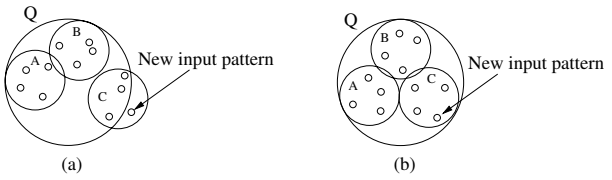

Fig. 3. (a) A new input pattern fails to be classified into category C. (b) The classification succeeded in a different trial.

In our approach, the order in which the input patterns are presented to the network will be chosen randomly. To avoid the problem shown in Figure 3(a), we will perform multiple trials with randomly selected order of presentation for the whole set of input patterns. For each trial, we may get slightly different categories and slightly different results in identified motifs. 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 is subject to be kept. If the following condition for the qth category is satisfied,

$$p_q^L >= \lambda,$$

we will keep this category, where λ is determined by the problem, e.g., $\lambda = 3$.

After picking out these categories, we use them to initialize the network in the next cycle in the lowest subnetwork. After that, we use the lowest subnetwork to initialize all other subnetworks. Following the two steps we mentioned in the last section, we build categories from subnetwork to subnetwork till the initialization of the network for this input cycle is done.

In each input cycle after the first one, we validate every input patterns before we put it into the network. It is necessary for us to see if this pattern is already classified in any category exists. If the result is positive, we will skip this input pattern. We do this in order to prevent classifying the same pattern into more than one categories.

III. Simulation Results

We consider an example in this section to show the applicability of the present results.

Example 1: In this example, following [28], we generate i.i.d. samples of DNA sequences with certain lengths. Motifs with random mismatch letters at randomly chosen positions are implanted in these sequences. The performance of the algorithm is defined as follows:

$$P_{erf} = \frac{|R \cap T|}{|R \cup T|} \tag{4}$$

where R is the motif set generated and T is the motif set identified. In the figures shown in this section, 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 as a representation of a motif), and the vertical axis indicates the performance averaged over 8 such simulations.

Figures 4 to 5 show the performances of the system on finding signals with length of 13 and 17. From the figures we can see that the results are still acceptable even with the mismatch letters up to 30%. After that, the performances drops sharply. The reason of the sharp drop is because that for the 4 letter DNA case, the total number of randomly generated sequences is not large enough, which makes the generated patterns to be often similar to the noise. We have to say thirty percent of mismatch is beyond the need for further study. Comparing to the results obtained in [1] and [23], our simulation results can find motifs with at least one mismatch letter more than the other two algorithm. We can conclude that our algorithm outperforms the MEME and Gibbs algorithms. In this simulation example, we generated 10 DNA sequences with 200 letters in each sequence. The computation time of our algorithm is 3 minutes. Compared with MEME (15 minutes) and Gibbs (12 minutes), our algorithm performs in a faster speed.

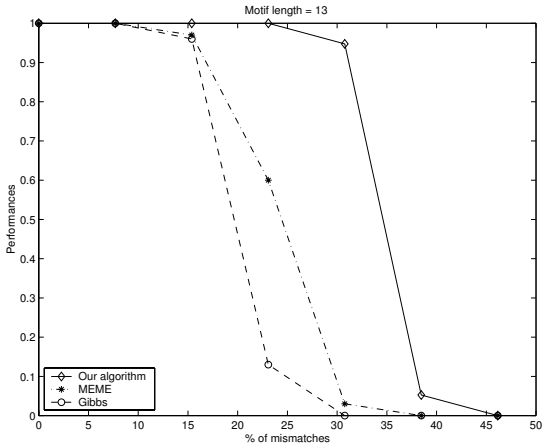

Fig. 4. Comparison results for motif length = 13

IV. Conclusions

In this paper, we studied the problem of subtle signal discoveries in unaligned DNA and protein sequences. We developed a self-organizing neural network structure for solving the problem of motif identification in DNA and

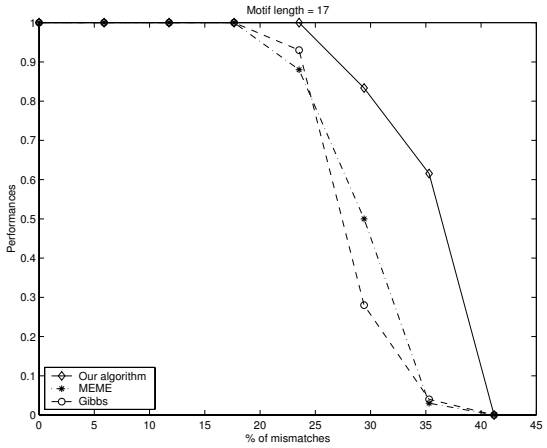

Fig. 5. Comparison results for motif length =17

protein sequences. Our network contains several layers with each layer performing classifications at different level. Our algorithm has a low computational complexity through the use of the layered structure so that each pattern's classification is performed with respect to a small subspace of the whole input space. Our algorithm also has a high reliability using since the self-organizing neural network will grow as needed to make sure all input patterns are considered and are given the same amount of attention. We compare our algorithm with other existing algorithms and show that our algorithm significantly outperforms existing algorithms, especially in the reliability aspect.

REFERENCES

[1] T. L. Bailey and C. Elkan, "Unsupervised learning of multiple motifs in biopolymers using expectation maximization," *Machine Learning,* vol. 21, no. 1–2, pp. 51–80, Oct.-Nov. 1995.

[2] T. L. Bailey and C. Elkan, "The value of prior knowledge in discovering motifs with MEME," *Proceedings of the 3rd International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, July 1995, pp. 21–29.

[3] T. L. Bailey and M. Gribskov, "Combining evidence using p-values: Application to sequence homology searches," *Bioinformatics,* vol. 14, no. 1, pp. 48–54, Feb. 1998.

[4] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing,* vol. 37, no. 1, pp. 54–115, Jan. 1987.

[5] G. A. Carpenter and S. Grossberg, "Search mechanisms for adaptive resonance theory (ART) architectures," *Neural Networks,* vol. 18, no. 22, pp. 201–205, June 1989.

[6] G. A. Carpenter and S. Grossberg, "ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures," *Neural Networks,* vol. 3, no. 2, pp. 129–152, Mar. 1990.

[7] G. A. Carpenter and S. Grossberg, "A self-organizing neural network for supervised learning, recognition, and prediction," *IEEE Communications Magazine,* vol. 30, no. 9, pp. 38–49, Sept. 1992.

[8] G. A. Carpenter, S. Grossberg, and D. Rosen, "Art 2-A: An adaptive resonance algorithm for rapid category learning and recognition," *Neural Networks,* vol. 4, no. 4, pp. 493–504, 1991.

[9] B. Chang, A. Ratnaweera, and S. K. Halgamuge, "Particle swarm optimisation for protein motif discovery," *Genetic Programming and Evolvable Machines,* vol. 5, pp. 203–214, June 2004.

[10] O. Emanuelsson, H. Nielsen, and G. Von Heijne, "ChloroP, A neural network-based method for predicting chloroplast transit peptides and their cleavage sites," *Protein Science,* vol. 8, no. 5, pp. 978–984, May 1999.

[11] D. Frishman and P. Argos, "A neural network for recognizing distantly related protein sequences," in E. Fiesler and R. Beale (eds.), *Handbook of Neural Computation,* New York, IOP Publishing and Oxford University Press, pp. 1–8, 1997.

[12] Y. Gdalyahu, D. Weinshall, and M. Werman, "Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1053–1074, Oct. 2001.

[13] P. Gonnet and F. Lisacek, "Probabilistic alignment of motifs with sequences," *Bioinformatics,* vol. 18, no. 8, pp. 1091–1101, Aug. 2002.

[14] W. N. Grundy, T. L. Bailey, C. P. Elkan, and M. E. Baker, "Meta-MEME: Motif-based hidden Markov models of protein families," *Computer Applications in the Biosciences,* vol. 13, no. 4, pp. 397–406, Aug. 1997.

[15] K. Gulukota, J. Sidney, A. Sette, and C. DeLisi, "Two complementary methods for predicting peptides binding major histocompatibility complex molecules," *Journal of Molecular Biology,* vol. 267, no. 5, pp. 1258–1267 Apr. 1997.

[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Upper Saddle River, NJ: Prentice Hall, 1999.

[17] G. Z. Hertz and G. D. Stormo, "Identifying DNA and protein patterns with statiscally significant alignments of multiple sequences," *Bioinformatics*, vol. 15, no. 7/8, pp. 563–577, 1999.

[18] T. Hofmann and J. Buhmann, "Pairwise data clustering by deterministic annealing," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 1, pp. 1–14, Jan. 1997.

[19] U. Keich and P. Pevzner, "Finding motifs in the twilight zone," *Bioinformatics,* vol. 18, no. 10, pp. 1374–1381, Oct. 2002.

[20] U. Keich and P. Pevzner, "Subtle motifs: Defining the limits of motif finding algorithms," *Bioinformatics,* vol. 18, no. 10, pp. 1382–1390, Oct. 2002.

[21] S. Knudsen, "Promoter2.0: For the recognition of PolII promoter sequences," *Bioinformatics,* vol. 15, no. 5, pp. 356–361, May 1999.

[22] P. Lavoie, J. F. Crespo, and Y. Savaria, "Generalization, discrimination, and multiple categorization using adaptive resonance theory," *IEEE Transactions on Neural Networks,* vol. 10, no. 4, pp. 757–767, July 1999.

[23] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment," *Science*, vol. 262, pp. 208–214, Oct. 1993.

[24] H. Nielsen, J. Engelbrecht, S. Brunak, and G. Von Heijne, "Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites," *Protein Engineering,* vol. 10, no. 1, pp. 1–6, Jan. 1997.

[25] H. Nielsen, J. Engelbrecht, S. Brunak, and G. Von Heijne, "A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites," *International Journal of Neural Systems,* vol. 8, no. 5-6, pp. 581–599, Oct.-Dec. 1997.

[26] A. R. Ortiz, A. Kolinski, and J. Skolnick, "Nativelike topology assembly of small proteins using predicted restraints in Monte Carlo folding simulations," *Processing of the National Academy Sciences of the USA,* vol. 95, no. 3, pp. 1020–1025, Feb. 1998.

[27] P. A. Pevzner and S.-H. Sze, "Combinatorial approaches to finding subtle signals in DNA sequences," *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, San Diego, CA, Aug. 2000, pp. 269–278.

[28] C. Workman and G. Stormo, "ANN-Spec: A method for discovering transcription factor binding sites with improved specificity," *Pacific Symposium of Biocomputing,* vol. 5, pp. 464–475, Jan. 2000.

[29] C. Wu, S. Shivakumar, H. P. Lin, S. Veldurti, and Y. Bhatikar, "Neural networks for molecular sequence classification," *Mathematics and Computers in Simulation,* vol. 40, no. 1-2, pp. 23–33, Dec. 1995.