Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Knowledge-Based Systems 26 (2012) 271–280

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Segmentation of DNA using simple recurrent neural network

Wei-Chen Cheng [a,b], Jau-Chi Huang [a], Cheng-Yuan Liou [a,*]

[a] Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, ROC
[b] Institute of Statistical Science, Academia Sinica, Taiwan, ROC

ARTICLE INFO

Article history:
Received 17 June 2011
Received in revised form 1 September 2011
Accepted 2 September 2011
Available online 17 September 2011

Keywords:
Quasi-regular structure
Elman network
Segmentation of DNA
SARS
H1N1

ABSTRACT

We report the discovery of strong correlations between protein coding regions and the prediction errors when using the simple recurrent network to segment genome sequences. We are going to use SARS genome to demonstrate how we conduct training and derive corresponding results. The distribution of prediction error indicates how the underlying hidden regularity of the genome sequences and the results are consistent with the finding of biologists: predicated protein coding features of SARS genome. This implies that the simple recurrent network is capable of providing new features for further biological studies when applied on genome studies. The HA gene of influenza A subtype H1N1 is also analyzed in a similar way.

1. Introduction

DNA consists of nucleotides. Certain locations of the DNA possess special meanings. The beginning and the end of a gene are two important locations. Segment is the basic unit, or building block to interpret DNA. The intron, exon and transcription factor are sections of DNA and play different roles in the transcription process. A gene is also a segment that can be used for making protein. The collection of segmented DNA can be further analyzed to show how the genes regulate each other and how those segments works. However, the reason that the segments can only exist at certain locations and the rules behind them are still unclear.

There are ways to accomplish the segmentation. One way to locate the beginning and the end of a segment is to search a similar sequence in the database. The idea behind this technique is that there exist similar patterns in different DNA sequences. In other words, the patterns in a strand of DNA sequence may have high possibility to be found in the strand of other DNA sequences. Researchers have dedicated to locate functional regions for decades. Statisticians try to locate the regions which satisfy the assumption of statistical models. Bernaola-Galvan et al. [1] provide a segmentation algorithm based on the Jensen–Shannon entropic divergence. This algorithm is used to decompose long-range correlated DNA sequences into statistically significant, compositionally homogeneous patches. Fujiwara et al. [2] developed a hidden Markov model that represents known sequence characteristics of mitochondrial targeting signals to predict the existence of the mitochondrial targeting signals. The signal is the presequence that directs nascent proteins bearing it to mitochondria. Hidden Markov model were also used in extracting motifs for predicting the binding sites of unknown transcription factors, without a priori knowledge, from functionally related DNA sequences [3]. Machine learning methods are capable of building the models automatically and, then, the huge number of combinations of features can be tested [17,18]. For example, Sonnenburg et al. [4] use the kernel weight to determine the exon start. García-Pedrajas et al. [5] developed the methods to cope with class imbalance problems for decision tree and support vector machine [6,7] in the problems of translation initiation site recognition.

A theory proposed that DNA sequences have language structures [8,9]. There are also attempts [11,12] to study the relationship between biological sequences and the Chomsky hierarchy [10]. The simple recurrent network (SRN) [13] is a hyper-Turing machine [14]. It has been shown [13,15] that it can learn arbitrary underlying grammars and automata from the presentation of sentences. Such automata-like structure is extremely difficult to reach by any statistical ways, for example, hidden Markov model. It is also argued [16] that Elman network can accommodate quasi-regular structure and makes use of this structure for predictions and inferences. Such quasi-grammartical structure cannot be analyzed by any rule-based systems. We expect that the DNA sequence could contain such kind structures. So, this network is a potential candidate to analyze DNA sequence. Specifically, the large prediction errors indicate the segmentation points [13]. We show an example to reveal such quasi-regular structures in the end of Section 3.

* Corresponding author. Fax: +886 223628167.
E-mail address: cyliou@csie.ntu.edu.tw (C.-Y. Liou).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.09.001

SARS genome is used in the first experiment. Then we employ two types of SRN to analyze influenza A virus. One type uses the perceptrons in the hidden layer and the other type uses self-organizing neurons in the hidden layer. The former can be trained by the back-propagation algorithm (BP). The later can be trained by the self-organizing rule. We did extensive simulations to find suitable parameters for SRN. The reason why we analyze the influenza A virus is that its subtype H1N1 was the cause of human influenza in 2009. Its HA (Hemagglutinin) region is responsible for binding the virus to the cell and causes infection [19]. Since hemagglutinin is the major surface protein of the influenza A virus and is essential to the entry process into a cell, it is the primary target of neutralizing antibodies.

2. Architecture

A simple recurrent network (or called Elman network) [13] is a three-layer neural network with the addition of a set of "context neurons" in the first layer, see Fig. 1. These context neurons assemble an inside self-reference layer. In each iteration, the previous state of the hidden layer saved in the context layer together with the input layer activates the hidden layer. This network maintains a stream of states which allows it to perform the sequence-prediction task. This network is proposed to model temporal human behaviors [13], like language. It can discover the underlying structure of words.

Elman generated sentences of varied lengths from fixed words. Those sentences were concatenated and formed a stream of words. Each word was represented as a combination of letters, and each letter was represented by a 5-bit randomly assigned binary vector. The network processed the concatenated binary vectors sequentially and was trained to predict the next letter by using the binary vector of the next letter as the desired output. Elman found that after training, the prediction error is very high at the beginning of a word and declines with the rest letters received. This implies that SRN has learned the various structures of words and is able to segment words from a sequence of letters.

Biologists use biotechnology (ex. polymerase chain reaction) to interact with a virus genome and look for interesting and meaningful regions (segments) of the sequence. Since genetic information is saved in the DNA sequence, we plan to use SRN to segment the sequence in a computational way. Based on the results Elman studied [13], we expect that SRN can learn the genome structure and detect the boundary of the protein coding region according to the prediction error. We further compare our findings with the protein coding regions found by other researchers.

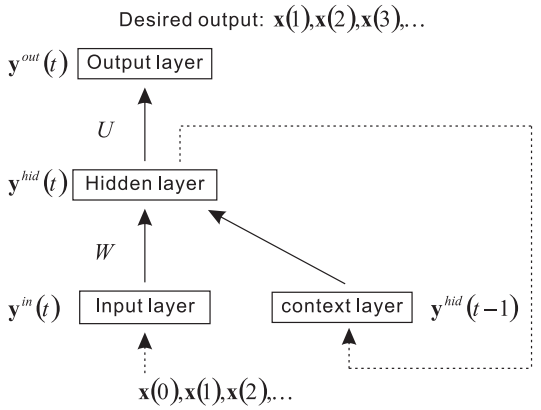

Fig. 1. The structure of the recurrent neural network used in the analysis of DNA sequence.

Consider a genome sequence $\{\mathbf{x}(t),\ t = 0,1,2,\ldots\}$, where $\mathbf{x}(t) \in \{\text{A(adenine), C(cytosine), T(thymine), G(guanine)}\}$. Instead of using 2 bits to encode the four nucleotides, we use 4 bits to prevent non-uniform similarity (cosine or Euclidean distance) for each nucleotide pair because any nucleotide can be joined by ester bonds to the preceding nucleotide without bias. The four nucleotides are $\text{A} \equiv [1,-1,-1,-1]^T$, $\text{C} \equiv [-1,1,-1,-1]^T$, $\text{T} \equiv [-1,-1,1,-1]^T$, and $\text{G} \equiv [-1,-1,-1,1]^T$. Each positive bit indicates one nucleotide. The number of dimensions of the context layer, which is the same as that of the hidden layer, is N. The number of dimensions of output layer is the same as that of the input layer. From extensive experiments, we set 20 hidden neurons in the first part of this work. The network has $M = 4$ input neurons, $N = 20$ hidden neurons, $N = 20$ context neurons, and $M = 4$ output neurons. Let the weight matrix W contain the set of synaptic weights that connects the input layer, context layer and the hidden layer, $W \in R^{N\times(M+N+1)}$. The weight matrix U contains the set of weights that connects the hidden layer and the output layer, $U \in R^{M\times(N+1)}$. The initial values of all synaptic weights in W and U are randomly assigned within the range $[-0.2, 0.2]$. The network is trained to predict the next nucleotide vector. For example, the input nucleotide at time $t = 0$ is $\mathbf{x}(0)$, and its desired output will be $\mathbf{x}(1)$. The input at time $t = 1$ is $\mathbf{x}(1)$, and the desired output will be $\mathbf{x}(2)$. The sequence of nucleotides is presented to the network one after another. For the convenience of mathematical expression, let the desired output $\mathbf{d}(0), \mathbf{d}(1), \mathbf{d}(2), \ldots$ denote the input data at the next time step,

$$\mathbf{d}(0) = \mathbf{x}(1), \quad \mathbf{d}(1) = \mathbf{x}(2), \ldots \tag{1}$$

The error signal at the output of neuron i at time t is defined by

$$e_i(t) = d_i(t) - y_i^{out}(t). \tag{2}$$

The total error is obtained by summing over all neurons in the output layer,

$$\zeta(t) = \frac{1}{2}\sum_{i=1}^{4} e_i^2(t). \tag{3}$$

The input layer $\mathbf{y}^{in}(t)$ consists of the input data at time t and the context layer which copies the activation of the hidden layer at the previous time step,

$$\mathbf{y}^{in}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{y}^{hid}(t-1) \end{bmatrix}. \tag{4}$$

The initial activation of the context layer is set to zero, $\mathbf{y}^{in}(0) = [\mathbf{x}(0)^T, 0 \ldots 0]^T$. The induced local field $v_i^{hid}(t)$ produced at the input of the activation function associated with hidden neuron i is

$$v_i^{hid}(t) = \sum_{j=0}^{M+N} w_{ij} y_j^{in}(t), \quad i \in \{1,\ldots,N\}, \tag{5}$$

where the synaptic weight w_{i0} (corresponding to the fixed input $y_0^{in} = -1$) is the bias. The induced local field $v_i^{out}(t)$ with the output neuron i is

$$v_i^{out}(t) = \sum_{j=0}^{N} u_{ij} y_j^{hid}(t), \quad i \in \{1,\ldots,M\} \tag{6}$$

where the synaptic weight u_{i0} is the bias and $y_0^{hid} = -1$. Hence the function signal y_i^{hid} appearing at the output of neuron i in the hidden layer at time t is

$$y_i^{hid} = f(v_i^{hid}(t)). \tag{7}$$

The y_i^{out} appearing at the output of neuron i in the output layer is

$$y_i^{out} = f(v_i^{out}(t)). \tag{8}$$

Table 1
Information on the 11 SARS genomes.

No.	Accession no.	Length (bps)
1	AY274119.3	29751
2	NC_004718.3	29751
3	AY597011.2	29926
4	AY278491.2	29742
5	AY278554.2	29736
6	AY278741.1	29727
7	AY283794.1	29711
8	AY283795.1	29705
9	AY283796.1	29711
10	AY283797.1	29706
11	AY283798.2	29711

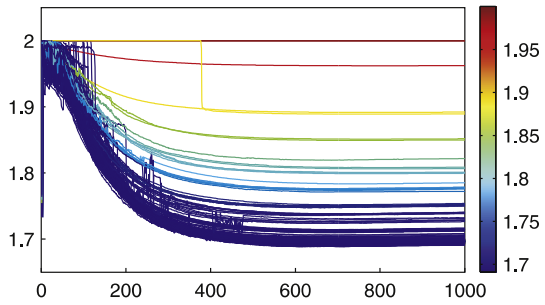

Fig. 2. Recorded 300 learning curves. The colors of curves indicate their converged mean square errors. 287 curves reach to values lower than 1.8.

In this work, we adopt the antisymmetric function, $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$, as the activation function of each neuron,

$$f(x) = \tanh(x), \tag{9}$$

and its derivative is

$$f'(x) = (1+x)(1-x). \tag{10}$$

Hence, the output of each neuron is in the range $[-1,1]$. The initial error is equal to $\zeta(0) = 2$. We expect that the nucleotide with a very large error could be the boundary of a protein coding region. The synaptic weights W and U are adjusted by the back-propagation algorithm [20] which performs gradient descent in error space. These weights are updated slightly in the direction that reduces error as much as possible to accomplish the expectation $\mathbf{d}(t) = \mathbf{x}(t+1) = E(\mathbf{x}(t)) \approx \mathbf{x}(t+1)$. The correction for the weight in W is Δw_{ij} and it is proportional to the partial derivative,

$$\Delta w_{ij}(t) = -\eta(t)\frac{\partial \zeta(t)}{\partial w_{ij}}. \tag{11}$$

where η is a learning rate function. η will be reduced to zero exponentially,

$$\eta(t) = \eta_0 \times e^{-\frac{a\times(t-t_0)}{t_1-t_0}}, \tag{12}$$

where iteration t starts from t_0. η_0 is the initial value of the rate. Set $\eta_0 = 0.5$ and $a = 6$ in this work. The correction for the weight in U is Δu_{ij},

$$\Delta u_{ij}(t) = -\eta(t)\frac{\partial \zeta(t)}{\partial u_{ij}}. \tag{13}$$

3. Analysis of SARS genomes

The SARS-CoV RNA has been detected frequently in respiratory specimens and convalescent-phase serum specimens from the patients having antibodies that react with SARS coronavirus. There is strong evidence that this virus is etiologically associated with the outbreak of SARS [21–23]. The genome has been analyzed by seeking the genes in the database. We select 11 complete genomes of SARS-CoV recorded in GenBank [24]. The accession numbers and their lengths (number of basepairs or bps in brief) are listed in Table 1. Note that the original record is a single-stranded positive sense RNA. Every selected sequence is the cDNA converted from

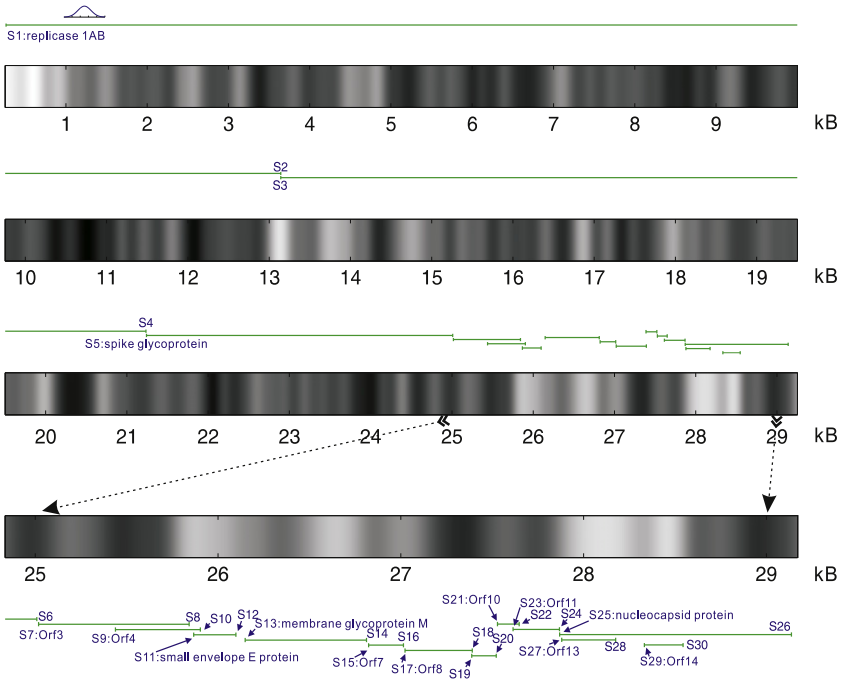

Fig. 3. The averaged prediction errors of the SARS "AY274119.3" genome. Each vertical band in the image shows a value that is averaged over an interval of 501 nucleotides by Gaussian function. This function is plotted on the top left corner. S1 to S30 indicate the beginning points and ending points of biologically identified 15 segments in [25]. Five segments belong to coronavirus. The rest ten segments are still unknown.

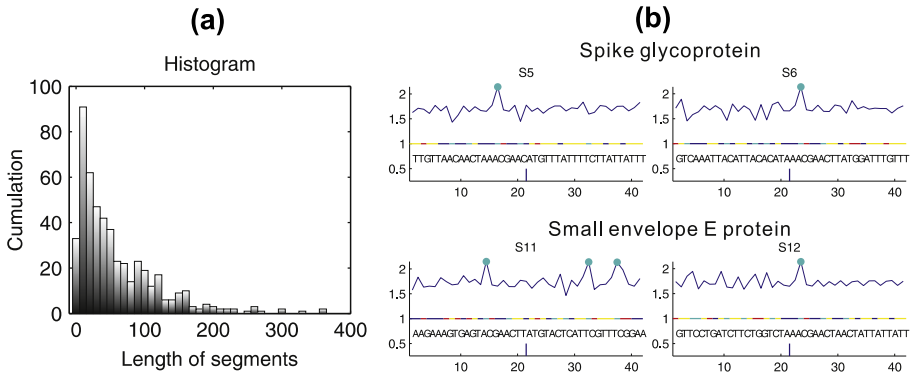

Fig. 4. (a) There are 100 bins in the histogram. Each bin has an interval of length 40 base pairs. (b) The error peaks marked by green color that are near the boundaries of the protein coding regions. The boundaries are marked by blue vertical lines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Comparison of the segmentation locations.

Index	Coding region	Product	Head	Tail	Closest Pt.
1	ATGGAGAGCCT...	Replicase 1A	265		254
	... CATCAACGTTT			13392	13388
2	TTTAAACGGGT...	Replicase 1B	13392		13388
	... TTAACAACTAA			21485	21543
3	ATGTTTATTTT...	Spike glycoprotein	21492		21543
	... ATTACACATAA			25259	25374
4	ATGGATTTGTT...	ORF 3	25268		25374
	... TGCCTTTGTAA			26092	26110
5	ATGATGCCAAC...	ORF 4	25689		25891
	... AGGTACGTTAA			26153	26148
6	ATGTACTCATT...	Small envelope E protein	26117		26110
	... TTCTGGTCTAA			26347	26287
7	ATGGCAGACAA...	Membrane glycoprotein M	26398		26421
	... TAGTACAGTAA			27063	27027
8	ATGTTTCATCT...	ORF 7	27074		27027
	... ATTATCCATAA			27265	27317
9	ATGAAAATTAT...	ORF 8	27273		27317
	... AGACAGAATGA			27641	27504
10	ATGAATGAGCT...	ORF 9	27638		27504
	... CCAAAGTCTAA			27772	28027
11	ATGAAACTTCT...	ORF 10	27779		28027
	... TACAACACTAG			27898	28027
12	ATGTGCTTGAA...	ORF 11	27864		28027
	... GAACAAATTAA			28118	28162
13	ATGTCTGATAA...	Nucleocapsid protein	28120		28162
	... CTCAGGCATAA			29388	29443
14	ATGGACCCCAA...	ORF 13	28130		28162
	... CGGCAAAATGA			28426	28396
15	ATGCTGCCACC...	ORF 14	28583		28593
	... ATTGCTGCTAG			28795	28638

its RNA. There is one-to-one correspondence between cDNA and RNA bases. We will use the cDNA sequence to train the network.

The network processes all 11 genome sequences which are concatenated in a long single sequence. We apply the BP algorithm to adjust its synaptic weights for 1000 epochs. The learning rate is reduced by (12) during the 1000 epochs. After training, we present the sequence again and record the prediction errors for all nucleotides. We repeat this training procedure for 300 times, hence, we obtain 300 trained SRNs and get 300 different prediction error sequences. Fig. 2 plots the 300 learning curves during the training processes. Each error point in a curve is the average error of all nucleotides in the 11 sequences. The network initially outputs $[-1,-1,-1,-1]$ for each input nucleotide pattern and the training makes the output to fit the next nucleotide in the sequence. Therefore, the training error is $\sqrt{\sum_i e_i^2(t)} = 2$ at the beginning. The learning curve does not decrease monotonously because the algorithm updates the weights immediately after presenting one input nucleotide pattern. This figure shows that after 1000 epochs, the 300 networks reached to a local or global minimum in the weight space. Each procedure takes roughly 35 min and the whole exper-

Table 3
List of short segments that have lengths less than 7.

Short segments			
CG	CGAGG	CGAGTT	CGTCTC
CGA	CGCAC	CGATAC	CGTCTG
CGC	CGCAG	CGATTT	CGTGAA
CGT	CGCTT	CGCAAT	CGTGTA
CGAA	CGGTT	CGCGTG	CGTGTT
CGAC	CGTAG	CGCTAC	CGTTTA
CGAT	CGTCA	CGGCCA	CGTTTT
CGCA	CGTTA	CGGTAC	
CGCC	CGTTG	CGTACC	
CGTG	CGTTT	CGTAGT	
CGTT	CGACTC	CGTATA	
CGAGA	CGAGCT	CGTCAG	

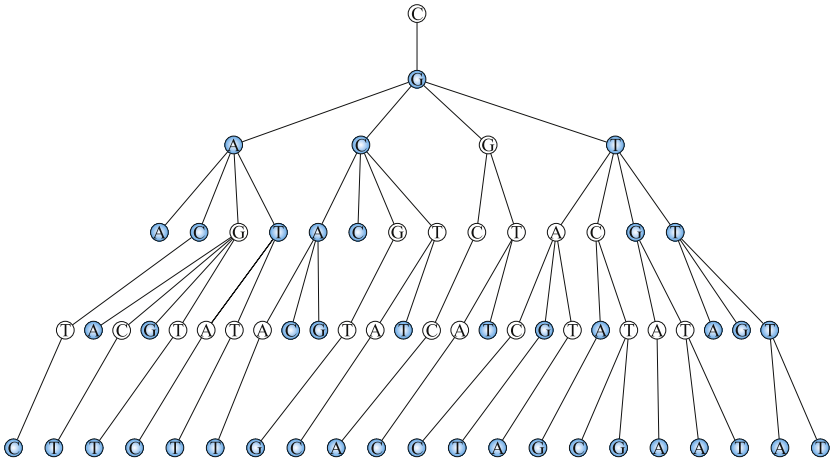

Fig. 5. Tree derived from short segments as listed in Table 3. The nodes which are the ends of protein coding regions are marked in blue color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 4
Setting of parameters (a).

# Samples	# Hidden neurons	Avg. Leng.
50	10	1725.58
50	20	1725.58
50	30	1725.58
50	40	1725.58
50	50	1725.58

Table 5
Setting of parameters (b).

# Samples	# Hidden neurons	Avg. Leng.
50	20	1725.58
100	20	1723.62
150	20	1720.68
200	20	1722.32
250	20	1721.62

Table 6
Setting of parameters (c).

# Samples	# Hidden neurons	Avg. Leng.
50	20	400
50	20	800
50	20	1200
50	20	1600

iment takes 175 hours per machine. Note that we use $\sqrt{\sum_i e_i^2(t)}$ to be the error in this figure.

In Fig. 3, we plot the averaged prediction error for each nucleotide along the genome of "AY274119.3". Error magnitude is represented by the gray level, white represents the largest error and black represents no error. These prediction errors are the averaged error values obtained after the 300 training procedures. To give a clear picture, we further smooth the predicted errors over an interval of 501 nucleotides using a Gaussian function plotted on the top left corner of this figure. This genome has been analyzed in [25], its results are also illustrated in Fig. 3 by green color. The white vertical band near the 13 kB shows that this region has large errors and it is also detected by Marra as the boundary of S2 and S3. Note that kB is the abbreviation of kilo-basepairs. The large region from 26 to 29.5 kB corresponds to the fragments detected in [25,26]. Marra's research shows that there are overlaps of the segments in this area [25]. For this genome, the maximal mean of prediction error is 2.1647, the minimal mean of prediction error is 1.1435, and the median of the mean is 1.7063.

Suppose that the 500 nucleotides which have the highest prediction errors are the boundaries of 501 segments. Fig. 4(a) plots the histogram of their length information. The shortest segment, which is "CG", has only 2 base pairs. The longest segment has 364 base pairs. Note that all 500 peaks are cytosine. Most segments have short lengths. The segment which has a long length implies that this portion of the genome has fewer mutations than other parts. Some of the short segments are codons. These segments may reveal the structural information in the genome sequence. We plot several predicted segmentation points which near the protein coding region in Fig. 4(b). The blue vertical lines on the bottom of Fig. 4(b) indicate the boundaries of the segments obtained by [25]. There are five hits among thirteen known protein coding regions.

Table 2 lists the detailed 15 genes of the identified SARS genome by the research [25]. The "head" means the beginning of a gene and the "tail" means the end of a gene along the genome location. The "Closest Pt." indicates the closest point, segmented by SRN, to the head or tail point. The "ORF" means the open reading frame. The work [25] focuses on the segments which begin with the start codon 'ATG' and end with the stop codon 'TGA', 'TAA', 'TAG'. It then searches the biological meaning of such segments in various databases. Fig. 4(b) shows two biologically identified protein coding regions, spike glycoprotein and small envelope E protein. They belong to coronavirus and have nucleotides ATGTTTATTTT...ATTACACATAA and ATGTACTCATT...TTCTGGTCTAA. These two regions are marked by S5, S6, S11, and S12 in this figure. The SRN finds the stop codon 'TAAA' in three cases and the start codon 'CGAAC' in all four cases.

Among the 501 segments, we list all short segments of lengths shorter than seven base pairs, <7, in Table 3 and construct a tree from them, see Fig. 5. From this tree, we see the number of nodes doesn't grow exponentially with tree layers. This means those segments aren't composed from "A", "C", "T", "G" arbitrarily. They follow certain structural rules and need further biological studies.

We assume DNA sequences are structured like languages which are quasi-regular: they allow the combination of some members of syntactic categories, but not others. For example, the sentences: "I gave food to the orphanage" and "I gave the orphanage food" are both correct. However, if we replace "gave" with "donated", the sentence "I donated the orphanage food" is wrong. From the tree in Fig. 5, we find the combinations of nodes are not symmetrical. It means that SRN has the capability to extract quasi-regular rule from DNA sequences.

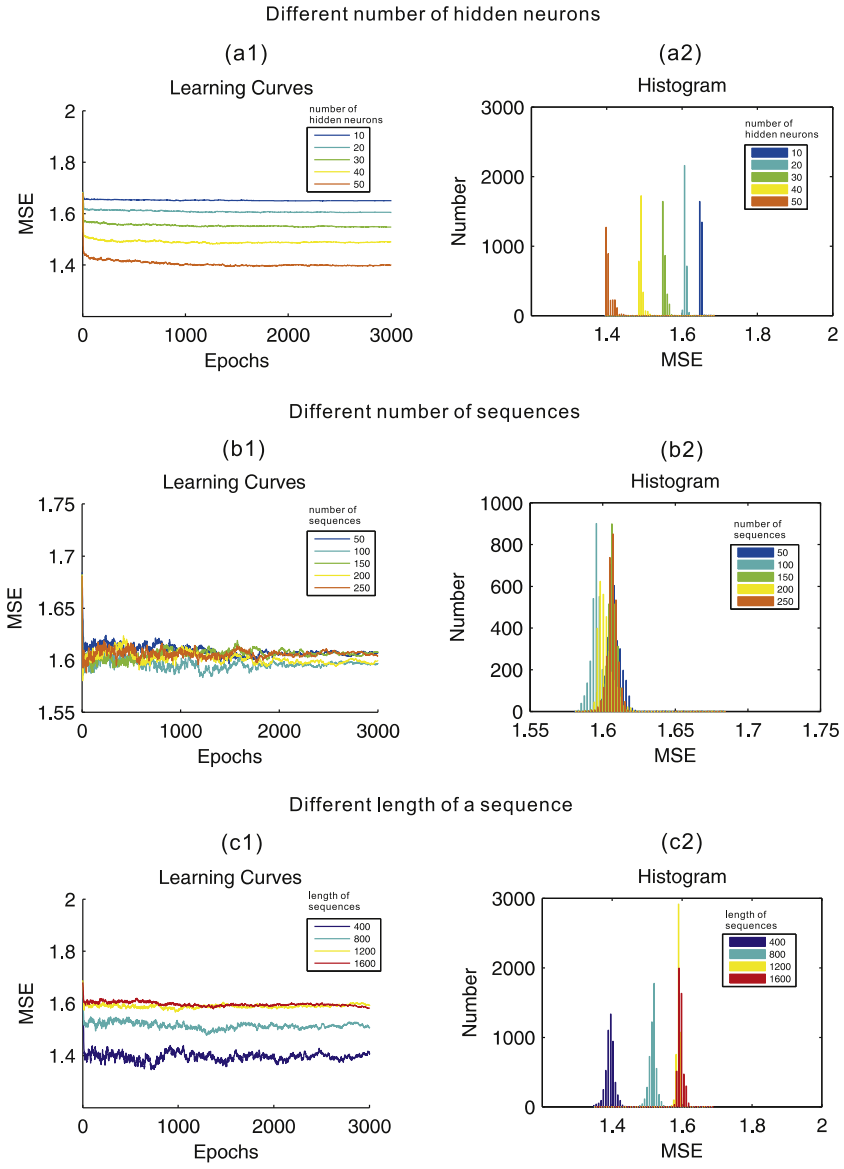

Fig. 6. The networks are trained by back-propagation. (a1) The learning curves with different numbers of hidden neurons. (a2) The histogram of the converged errors from (a1). (b1) The learning curves with different numbers of training DNA sequences. (b2) The histogram of the converged errors from (b1). (c1) The learning curves of different lengths of training DNA sequences. (c2) The histogram of the converged errors from (c1).

4. Analysis of H1N1 sequences

After analyzing the genome of SARS-CoV, we are going to analyze another virus: influenza A subtype H1N1. There are thousands of samples of this virus and it mutates frequently. We download 5580 DNA sequences of the segment HA of this virus [27], whose function is to produce hemagglutinin. They contain duplicate sequences. We do not exclude identical sequences because redundancy may contain useful evolution information. The minimum length of these sequences is 1664 bps. The maximum length of these sequences is 1846 bps. These sequences are not aligned. The original nucleotide sequences will be used in the training of SRN.

We randomly selected 50 sequences in a preliminary study to find suitable experiment settings. The longest sequence has 1791 bps and the shortest sequence has 1696 bps. The test settings are listed in the Tables 4–6. All simulations are repeated for 50 times with different initial weights. We try three different kinds of conditions and each of them changes only one variable. Firstly, Table 4 shows the setting with different number of hidden neurons listed in the column "# hidden neurons". Fig. 6(a) plots the results of the training. The learning curves in Fig. 6(a1) shows that when we use dense neurons in the hidden layer we will get small errors. Each learning curve is the average over 50 repeated simulations. Fig. 6(a2) shows the histogram of the converged errors for all 50 repeated simulations. Secondly, we use different number of sequences to train the network and plot their learning curves. Table 5 lists the number of randomly chosen sequences in different simulations and the average lengths. Fig. 6 (b1) plots the learning curves averaged over 50 simulations with different number of sequences. These curves show that when the number of sequences increases, the durations for convergence do not increase much. This phenomenon reveals that most sequences have similar hidden structures. Small group of sequences contain sufficient information to represent the rest sequences. Fig. 6(b2) shows the distribution of the converged errors. We randomly select 50 sequences and cut the rest portions of these sequences from the beginning. Table 6 lists the different lengths of those sequences. Note that the

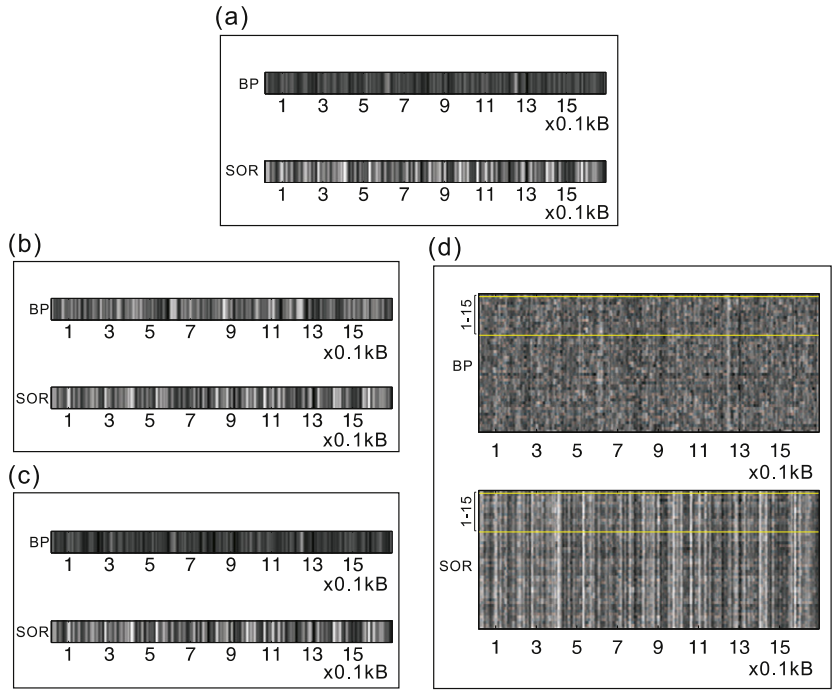

Fig. 7. This figure plots the averaged errors along the closest sequence. (a) The averaged prediction errors of the best trained SRN with lowest converged error. (b) The averaged prediction errors of the best 15 trained SRNs that have smallest 15 converged errors. (c) The averaged prediction errors over all 50 simulations. (d) All prediction errors of the 50 simulation. The performance of these 50 trained networks are sorted from top to bottom.

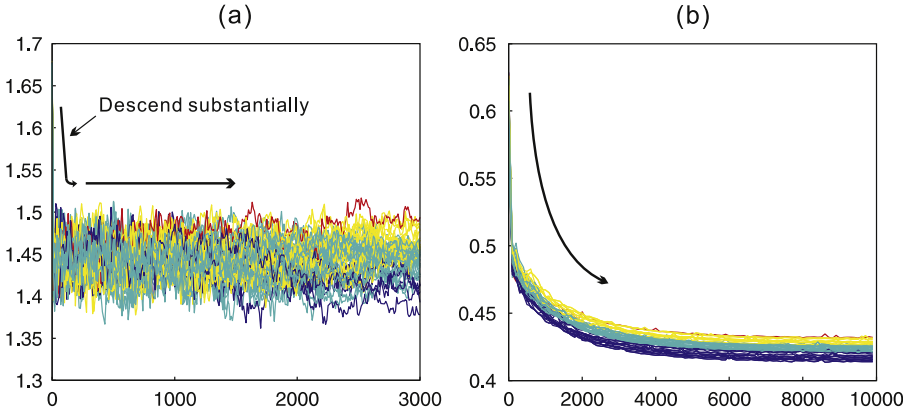

Fig. 8. The plots show all the 50 learning curves of two methods. (a) The learning curves by BP. (b) The learning curves by self-organizing rule.

randomly chosen 50 sequences in different simulations are not the same. Fig. 6(c1) plots the learning curves of different lengths of sequences. Fig. 6(c2) shows the distribution of the converged errors.

The computational complexity for training SRN is $O(PM(t_1 - t_0)(n_1 - n_0))$, where P is the number of sequences. Processing all 5580 sequences is costly. From Fig. 6(b), we see that a suitable subset of the 5580 sequences can do the training task. We employ DISOM (Distance Invariant Self-Organizing Map) [28,29] to select the subset sequences. DISOM can sort the sequences and find their distances to the grandmother virus. We select the 1032 sequences sampled from January to May 2009 to simplify the computation. These 1032 sequences are all different. We use ClusterW2 [30] to align the 1032 sequences. The lengths of aligned sequences are all 1710 bps. The DISOM is employed to project high dimensional data onto a three dimensional space. The 100 viruses closest to the cluster center in this space are retrieved. There are 137 such sequences because some sequences are identical.

We set 40 neurons in the hidden layer and the context layer of the network. The network is trained by the back-propagation algorithm. The experiments are repeated 50 times. We plot the averaged error in Fig. 7 marked by BP for the sequence that is closest to the cluster center.

4.1. Analysis H1N1 using unsupervised simple recurrent network

For comparison, we further use the unsupervised SRN [31,32] to process the H1N1 sequences. The results are plotted in Fig. 7 marked by SOR. This unsupervised SRN was proposed by Voegtlin. The self-organizing neurons are used in the hidden layer and context layer; see Fig. 1. The topology of these neurons is a grid square map. These neurons use time-delay feedback to represent the information hidden in time. This recursive feedback makes this network different from the original self-organizing map [33]. The

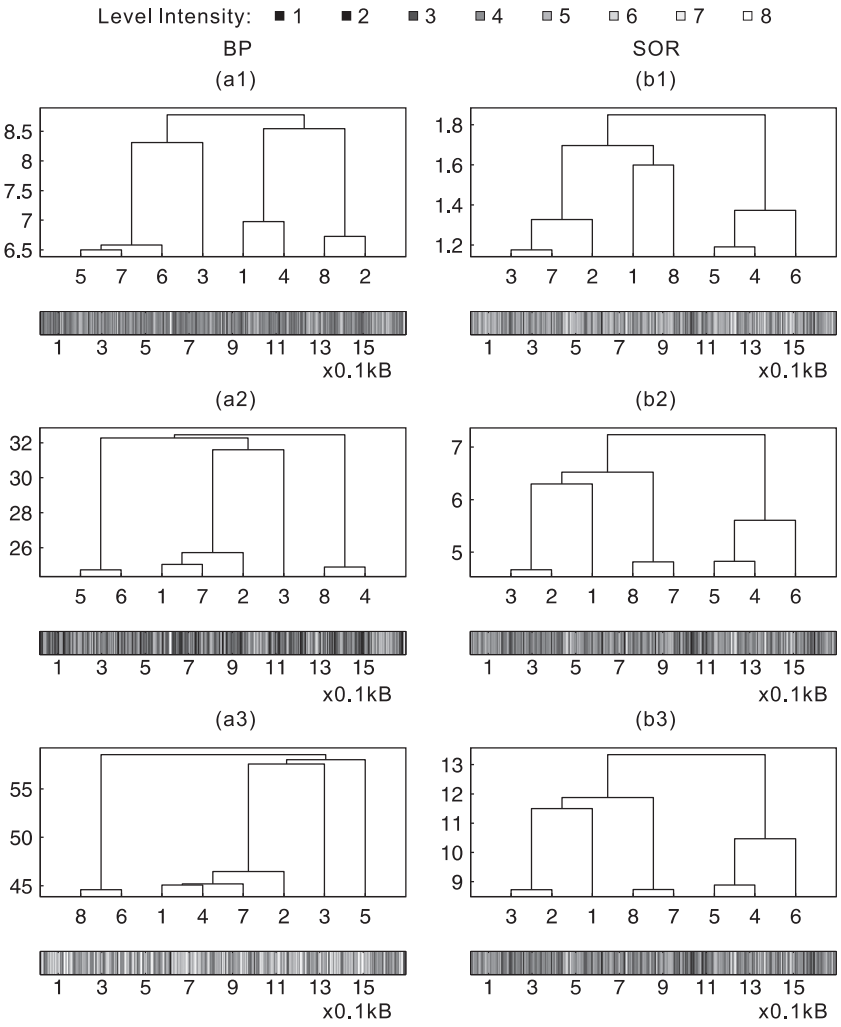

Fig. 9. Hierarchical clustering diagram of the activations of hidden layer in influenza analysis. Each intensity indicates a cluster. The intensities are assigned according to the levels of the leaf nodes. The plots show the results of supervised (a) and unsupervised learning (b) for different trained networks. (1–3) are the trees constructed from the activations of the hidden neurons that have the minimum converged error, minimum 15 converged errors, and all 50 trained networks respectively.

synaptic weights are updated according to the self-organizing rule [33].

The 137 sequences are used to train this unsupervised network. After extensive trials, we set the network with 8×8 hidden neurons and use it to analyze H1N1 virus. The results are marked by SOR in Fig. 7. The training procedure is repeated 50 times. All 50 learning curves are recorded in Fig. 8(b). The learning 50 curves for the SRN with 40 hidden neurons and trained by BP are also plotted in Fig. 8(a). The BP learning curves show that the SRN tries to find information and rules in time and the rules compete against each other. We see that the curve jumps up and down rapidly. But the learning curve obtained by the self-organizing rule is relatively well behaved. The sequence closest to the center is used for calculating the prediction errors and these errors are plotted and marked with SOR in Fig. 7. There are 50 converged errors. We sort these 50 errors from top to bottom and show their prediction errors in Fig. 7(d). Note that Fig. 7(d) plot the smoothed prediction errors by a Gaussian low pass filter with a window size of 31. Stronger intensity indicates higher error in the figure. In supervised BP learning, the nucleotides in the high error regions are less predictable. In unsupervised learning, the high error regions show the nucleotides are away from the statistical center in time domain. The best converged error is plotted on the top of the image Fig. 7(d).

4.2. Clustering hidden activations

In order to visualize the structure in time, we employ the hierarchical clustering method [34] to classify the activations of the hidden layer of SRN. This method was used in Elman's work [13] to group the meanings of words. It aggregates the clusters, which have minimum distances, and constructs a binary tree by merging clusters. After constructing the tree, one can cut the leaf nodes by setting a threshold distance. In the communication between Plate and Elman, they have noticed that the activation of hidden neurons is affected by the input, "... The hidden unit activation patterns are highly dependent upon preceding inputs...", see line 2 of page 199 in [13]. In Fig. 9, we generate the dendrogram with no more than eight leaf nodes instead of four in order to visualize more information. Setting eight clusters in this case means each one of the four clusters, corresponding to the four nucleotides, is further divided into two groups. The colors of the 8 leaf nodes are listed on the top of this figure. The cluster intensities are assigned by the levels of the leaf nodes. This is because nodes in the same cluster should have similar intensities. For example, in Fig. 9(a1), the node 5 has an intensity black which corresponds to grey code 1. Node 7 has an intensity as that of code 2 and node 6 has an intensity code 3 and so on. Similar structures can be found in the two different methods. Group (1,7,2) in (a2) is similar to group (5,4,6) in (b2). Without

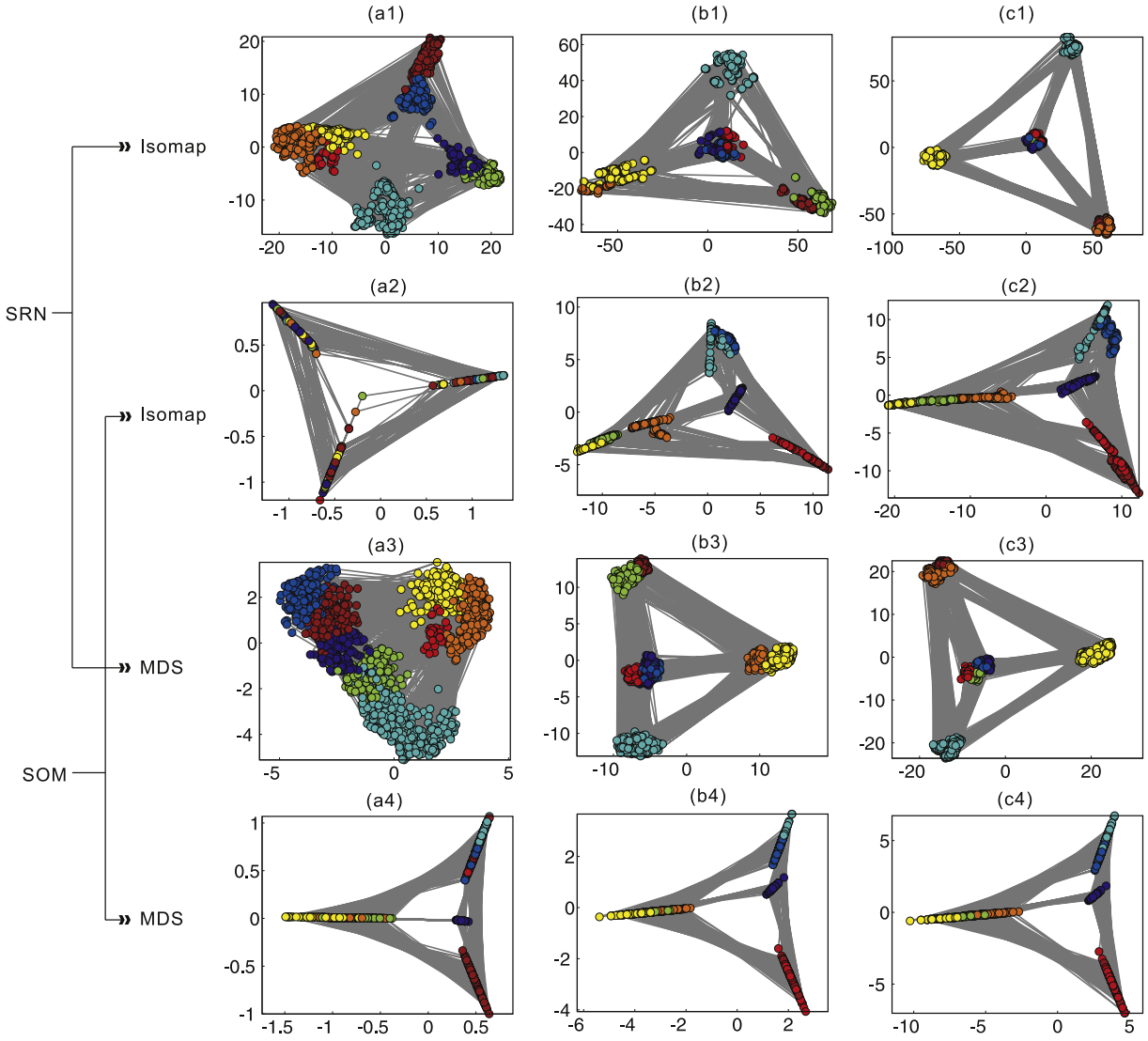

Fig. 10. This figure shows the results of mapping the hidden activations in two dimensional space. The 8 colors are 8 clusters by hierarchical clustering method. The grey links show the transits of hidden states.

considering the link length, (a2) is isomorphic to (b2), this is because there is a bijective mapping from nodes (5,6,1,7,2,3,8,4) in (a2) to nodes (3,2,5,4,6,1,8,7) in (b2). This means these two methods catch similar structure in time.

4.3. Visualizing hidden activations in two dimensional space

The hierarchical clustering process confines the representation of the relations in a tree-like structure. We use Isomap [35] and multidimensional scaling (MDS) [36] to visualize the hidden activations in a two dimensional space, see Fig. 10. The colors of leaf nodes obtained from hierarchical clustering are kept in this figure. The grey links between points show the adjacent temporal relations along the genome sequence. One activation follows the other activation if there is a link between them. In Isomap, the number of neighborhoods are set to 60, 300, 350 in Fig. 10 (a1), (b1) and (c1) respectively. The number of neighborhoods are also set to 60, 300, 350 in Fig. 10 (a2), (b2), and (c2). Fig. 10 (a1–a4) are obtained from the best trained networks. We see that the best trained SRN, in Fig. 10(a1) and (a3), can resolve the activations according to their appearances in the genome sequence. This is, in some sense, similar to the polysemous of a word. Fig. 10(b1–b4) are obtained from the best 15 trained networks. Fig. 10(c1–c4) are obtained from all 50 trained networks.

The residual variances in Fig. 11 show how much information is captured with respect to dimensionality by the two dimension reduction algorithms, Isomap and MDS. The residual variances of Isomap may not decrease monotonously for SRN trained by the BP algorithm. The residual variance decreases as the dimensionality is increased for SRN trained by the self-organizing rule. Four dimensions are enough to catch most variances of the hidden activations for the H1N1 sequences.

5. Summary

This work presents a new technology to study genome sequences. Without any prior biological knowledge and only processing the ATCG sequences, the result is strikingly consistent with the findings from biologists. This implies that we can use this new technology to study more complicated genomes which are still a mystery to biologists. The underlying structures detected by SRN provide new types of features for further biological studies. By ranking the errors, this technology provides the priorities for biologists to choose which part of the genomes is worth to study.

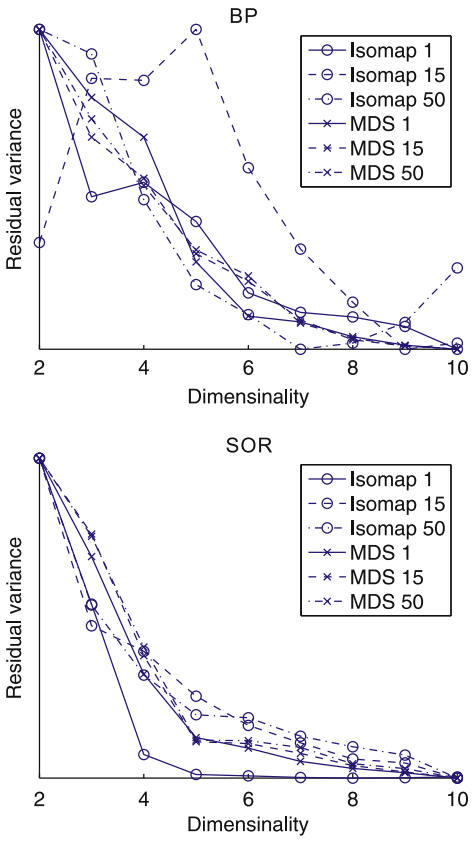

Fig. 11. The residual variances of Isomap (circles), MDS (cross) in different dimensions for the hidden activations of the trained SRN, Isomap 1 and MDS 1 are plots for the best trained network. Isomap 15 and MDS 15 are for the best 15 trained networks. Isomap 50 and Isomap 50 are for all 50 trained networks. Each curve is normalized within zero and one in the *y*-axis.

The results of the proposed segmentation method can be used in distinguishing an artificial DNA segment from an natural segment, because the nucleotides joined together in the natural environment may be different from the one joined in the laboratory.

Acknowledgment

This work was supported by National Science Council under project NSC100-2221-E-002-234-MY3.

References

[1] P. Bernaola-Galvan, R. Roman-Roldan, J. Oliver, Compositional segmentation and long-range fractal correlations in dna sequences, Physical Review E 53 (5) (1996) 5181–5189.
[2] Y. Fujiwara, M. Asogawa, K. Nakai, Prediction of mitochondrial targeting signals using hidden markov models, Genome Informatics (1997) 5360.
[3] T. Yada, Y. Totoki, M. Ishikawa, K. Asai, K. Nakai, Automatic extraction of motifs represented in the hidden markov model from a number of dna sequences, Bioinformatics 14 (4) (1998) 317–325.
[4] S. Sonnenburg, G. Rätsch, C. Schäfer, S.B., Large scale multiple kernel learning, Journal of Machine Learning Research 7 (2006) pp. 1531–1565.
[5] N. García-Pedrajas, J. Pérez-Rodríguez, M. García-Pedrajas, D. Ortiz-Boyer, C. Fyfe, Class imbalance methods for translation initiation site recognition in dna sequences, Knowledge- Based Systems.
[6] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, S.Y. Bang, Pattern classification using support vector machine ensemble, in: Proceedings of the 16th International Conference on Pattern Recognition, 2002, pp. 160–163.
[7] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, S.Y. Bang, Constructing support vector machine ensemble, Pattern Recognition 36 (2003) 2757–2767.
[8] S. Dong, D. Searls, Gene structure prediction by linguistic methods, Genomics 23 (1994) 540–551.
[9] D. Searls, The language of genes, Nature 420 (2002) 211–217.
[10] N. Chomsky, Syntactic Structures, Mouton & Co., 1957.
[11] R. Durbin, S.R. Eddy, A. Krogn, G. Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.
[12] P. Baldi, S. ren Brunak, Bioinformatics: The Machine Learning Approach (Adaptive Computation and Machine Learning), 2nd ed., The MIT Press, 2001.
[13] J. Elman, Finding structure in time, Cognitive Science 14 (1990) 179–211.
[14] H.T. Siegelmann, Analog computation via neural networks, Theoretical Computer Science 131 (1994) 331–360.
[15] T. McQueen, A. Hopgood, T. Allen, J. Tepper, Extracting finite structure from infinite language, Knowledge-Based Systems 18 (2005) 135–141.
[16] J. Elman, Generalization, simple recurrent networks, and the emergence of structure, in: Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, 1998.
[17] S.-Y. Le, W. min Liu, J.V. Maizel, A data mining approach to discover unusual folding regions in genome sequences, Knowledge-Based Systems 15 (2002) 243–250.
[18] K. Ondrej, J. Jakub, J. Dalibor, Use of mobile phones as intelligent sensors for sound input analysis and sleep state detection, Sensors 11 (2011) 6037–6055.
[19] M. Throsby, E. van den Brink, M. Jongeneelen, et al., Heterosubtypic neutralizing monoclonal antibodies cross-protective against H5N1 and H1N1 recovered from human igm+ memory b cells, PloS One 3.
[20] D. Rumelhart, G. Hinton, R. Williams, Learning internal representations by error propagation, in: D. Rumelhart, J. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, 1986, pp. 318–362.
[21] T. Ksiazek, D. Erdman, C. Goldsmith, et al., A novel coronavirus associated with severe acute respiratory syndrome, The New England Journal of Medicine 348 (2003) 1953–1966.
[22] J. Peiris, S. Lai, L. Poon, et al., Coronavirus as a possible cause of severe acute respiratory syndrome, The Lancet 361 (2003) 1319–1325.
[23] C. Drosten, S. Güther, W. Preiser, Identification of a novel coronavirus in patients with severe acute respiratory syndrome., The New England Journal of Medicine 348 (2003) 1967–1976.
[24] D. Benson, I. Karsch-Mizrachi, D. Lipman, J. Ostell, D. Wheeler, Genbank, Nucleic Acids Research 33 (2005) D34–D38.
[25] M. Marra, S. Jones, C. Astell, et al., The genome sequence of the sars-associated coronavirus, Science 300 (2003) 1399–1404.
[26] M. Lai, K. Holmes, Fields Virology, 4th ed., Lippincott Williams & Wilkins, New York, 2001. chapter 36.
[27] Y. Bao, P. Bolotov, D. Dernovoy, et al., The influenza virus resource at the national center for biotechnology information, Journal of Virology 82 (2008) 596–601.
[28] W.-C. Cheng, C.-Y. Liou, Manifold construction based on local distance invariance, Memetic Computing 2 (2010) 149–160.
[29] W.-C. Cheng, C.-Y. Liou, Visualization of influenza a protein segments in distance invariant self-organizing map, International Journal of Intelligent Information and Database Systems.
[30] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. Gibson, D. Higgins, J. Thompson, Multiple sequence alignment with the clustal series of programs, Nucleic Acids Research 31 (2003) 3497–3500.
[31] T. Voegtlin, P. Dominey, Recursive self-organizing maps, in: N. Allinson, H. Yin, L. Allinson, J. Slack (Eds.), Advances in Self-Organizing Maps, Springer, 2001, pp. 210–215.
[32] T. Voegtlin, Recursive self-organizing maps, Neural Networks 15 (2002) 979–991.
[33] T. Kohonen, Self-organized formation of topologically correct feature maps., Biological Cybernetics (1982) 59–69.
[34] S. Johnson, Hierarchical clustering schemes., Psychometrika 32 (3) (1967) 241–254.
[35] J. Tenenbaum, S.V. de, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[36] W. Torgerson, Multidimensional scaling: I theory and method, Psychometrika 17 (1952) 401–419.