computation

MDPI

Review

Self-Organizing Map for Characterizing Heterogeneous Nucleotide and Amino Acid Sequence Motifs

Xuhua Xia [1,2]

[1] Department of Biology, University of Ottawa, 30 Marie Curie, Ottawa, ON K1N 6N5, Canada; xxia@uottawa.ca; Tel.: +1-613-204-2347

[2] Ottawa Institute of Systems Biology, 451 Smyth Road, Ottawa, ON K1H 8M5, Canada

Received: 19 July 2017; Accepted: 25 September 2017; Published: 26 September 2017

Abstract: A self-organizing map (SOM) is an artificial neural network algorithm that can learn from the training data consisting of objects expressed as vectors and perform non-hierarchical clustering to represent input vectors into discretized clusters, with vectors assigned to the same cluster sharing similar numeric or alphanumeric features. SOM has been used widely in transcriptomics to identify co-expressed genes as candidates for co-regulated genes. I envision SOM to have great potential in characterizing heterogeneous sequence motifs, and aim to illustrate this potential by a parallel presentation of SOM with a set of numerical vectors and a set of equal-length sequence motifs. While there are numerous biological applications of SOM involving numerical vectors, few studies have used SOM for heterogeneous sequence motif characterization. This paper is intended to encourage (1) researchers to study SOM in this new domain and (2) computer programmers to develop user-friendly motif-characterization SOM tools for biologists.

Keywords: self-organizing map; machine learning; artificial neural network; motif characterization

1. Introduction

A self-organizing map or SOM [1] is a grid of artificial neurons that are used to learn patterns from training data and use the learned pattern to perform non-hierarchical clustering to represent input vectors as discretized clusters, with vectors in the same cluster sharing similar features. It is used extensively in analyzing transcriptomic data, especially with gene expression measured over a number of time points [2–7]. While SOM has almost always been presented as a non-hierarchical clustering method for numerical vectors (e.g., [1] and pp. 231–250 of [8]), it theoretically can be adapted to any set of objects from which a pairwise distance between two objects can be computed. Thus, it should be applicable for fixed-length strings such as nucleotide or amino acid sequences. For example, the Kozak consensus [9,10] as a motif signal for ribosomes to locate mammalian translation start codon is often represented as a 7mer (RccAUGG). Similarly, 5′ and 3′ splicing sites are typically represented as exon-intron junctions with m nucleotides on the exon side and n nucleotides on the intron side, with a fixed length of m + n nucleotide per sequence [11,12]. Splice sites are often heterogeneous, with some for major-class spliceosomes, some for minor-class spliceosomes, and some for splicing by special mechanisms. For example, the transcript of yeast gene, *HAC1*, is processed by a non-spliceosomal splicing mechanism [13–16], and splice sites of *HAC1* introns are drastically different from those of regular introns [11]. One may use SOM to characterize these heterogeneous splice sites, quantify similarities among them, and potentially discover new type of introns. Several studies have demonstrated the values of using SOM to characterize sequence motifs [17–22], but their efforts do not seem sufficiently appreciated by biologists. Given the many advantages of SOM over conventional

clustering [1], biologists should gain a new perspective on the relationship among sequence motifs through the SOM extension in this paper.

SOM involves setting up a grid of artificial neurons, initializing them either with random values or with values from routine multidimensional scaling methods such as PCA, computing a distance (or similarity) between an input vector and each neuron to identify the winning neuron (which has the shortest distance or greatest similarity to the input vector), revising the features of the winning neuron and its neighbors as a learning process, and continuing with other input vectors until the process is converged (i.e., when the vector values of neurons no longer change). Such a trained SOM can then be used to classify input vectors that are not in the training data.

There are various ways of applying SOM to motif characterization, with the main difference among them being the measurement of distance/similarity between a SOM node and an input motif. I present this new extension of SOM in parallel with the familiar SOM application involving numerical vectors. In this way, the readers will find it easy to understand SOM with sequence motifs as input.

2. Distance or Similarity between Two Vectors

A key element in SOM is to compute a distance or similarity between two vectors, one representing the neuron and the other representing the input. For input with numeric vectors, the neuron is also a numeric vector, and there are many distances defined between two numeric vectors [7]. For vectors **x** and **y** in N-dimensions, one of the simplest distances is the Euclidean distance (d) defined as

$$d = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \quad (1)$$

For SOM on fixed-length sequences, the neuron will be represented either by a sequence or a set of sequences. I present two candidate distances, one for homologous input sequences and one for non-homologous input sequences.

2.1. Distance for Homologous Input Sequences

For homologous input sequences, one may use evolutionary distances derived from substitution models such as JC69 [23], K80 [24], F84 [25,26], HKY85 [27], TN93 [28], and GTR [29,30] models. Distances derived from these models are respectively referred to as JC69 distances, K80 distances, and so on. Each of these distances can be estimated by either independent estimation (IE) or simultaneous estimation (SE). IE uses information in two sequences only, and SE uses information in all pairs of sequences [31,32]. As many biologists, including some practising phylogeneticists, are unaware of the difference between IE and SE methods, I offer a numeric illustration of these two estimation methods by using data in Table 1 and the K80 model.

The K80 model has two parameters which can be expressed either as αt and βt, or as D and κ, where D is evolutionary distance that we are interested in. The expected proportions of transitions and transversions between the two sequences, designated by $E(P)$ and $E(Q)$ and expressed in D and κ, respectively, are

$$\begin{aligned} E(P) &= \tfrac{1}{4} + \tfrac{1}{4}e^{-\frac{4D}{\kappa+2}} - \tfrac{1}{2}e^{-\frac{2D(\kappa+1)}{\kappa+2}} \\ E(Q) &= \tfrac{1}{2} - \tfrac{1}{2}e^{-\frac{4D}{\kappa+2}} \end{aligned} \quad (2)$$

Suppose we have three aligned sequences, S1, S2, and S3, with sequence length of L (=100). The observed number of sites with transitional and transversional differences are shown in Table 1. For IE estimation, one simply substitutes $E(P)$ and $E(Q)$ in Equation (2) by the observed P and Q (which equal N_s/L and N_v/L, respectively (Table 1), and then solves for D and κ. Note that D cannot be estimated between S1 and S3 by this IE approach (Table 1).

For the SE method with K80 distances, we assume the same κ but different D_{ij} (i.e., D_{12}, D_{13}, and D_{23}) and maximize the following ln$L$

$$\begin{aligned}\ln L = {} & N_{s.12}\ln[E(P_{12})] + N_{v.12}\ln[E(Q_{12})] + N_{i.12}\ln[1 - E(P_{12}) - E(Q_{12})] \\ & + N_{s.13}\ln[E(P_{13})] + N_{v.13}\ln[E(Q_{13})] + N_{i.13}\ln[1 - E(P_{13}) - E(Q_{13})] \\ & + N_{s.23}\ln[E(P_{23})] + N_{v.23}\ln[E(Q_{23})] + N_{i.23}\ln[1 - E(P_{23}) - E(Q_{23})]\end{aligned} \quad (3)$$

We take partial derivative of ln$L$ with respect to D_{12}, D_{13}, D_{14}, and κ, set them to zero and solve the four simultaneous equations for the four unknowns. This leads to κ = 6.2385 and the three D_{ij} values shown in Table 1. There is no problem for the SE method to estimate D_{13} which was not possible with the IE method. SE distances were implemented in DAMBE [33,34] as MLCompositeF84, MLCompositeTN93. For SOM, this κ is estimated from all sequence pairs and does not change during the learning process.

Table 1. Observed number of sites between two sequences that are (1) identical (N_i), (2) different by a transition (N_s), and (3) different by a transversion (N_v) from pairwise comparisons among three sequences (S1, S2, and S3) of length 100. Independently estimated K80 distances (D_{IE}) and simultaneously estimated K80 distances (D_{SE}) from the maximum likelihood (ML) framework are included. Note that K80 distance cannot be computed for S1 and S3 using independent estimation method (labelled as 'inapplicable').

Seq. Pair	N_s	N_v	N_i	D_{IE}	D_{SE}	Seq. Pair
S1 vs. S2	9	4	87	0.1451	0.1464	S1 vs. S2
S1 vs. S3	40	30	30	Inapplicable	2.0915	S1 vs. S3
S2 vs. S3	20	10	70	0.4024	0.4116	S2 vs. S3

An alternative, and simpler, approach is simply to use the alignment score as a similarity index. Alignment score can be obtained from either global or local alignment by dynamic programming, or by local string-matching algorithms such as BLAST and FASTA, the latter having been used in SOM for motif characterization [22]. In addition to fixed-length string, SOM has also been used with variable-length strings [35,36]. Such studies reduce sequences into word frequencies (e.g., trinucleotide frequencies) so that each sequence, regardless of its length, is represented as a fixed-length vector for use with conventional SOM. The distances derived from these non-model-based approaches do not correct for multiple substitutions and are not expected to increase linearly with time. For this reason, they are not appropriate for molecular phylogenetics. For unaligned sequences of different lengths, PhyPA [37] is statistically sound and rivals the maximum likelihood method in phylogenetic accuracy when highly divergent sequences are involved.

2.2. Distance for Non-Homologous Sequences

The input sequences used in SOM for motif characterization typically are not homologous. For example, splice sites of introns mentioned before share similarities but are typically non-homologous. In such cases, a position weight matrix (PWM, [38–40]), numerically illustrated in detail [41], is often used to represent a SOM neuron [17–20]. I will illustrate the PWM approach in conjunction with the SOM algorithm. The shortcoming in PWM is that it does not take site-dependence into account. In contrast, Markov models can overcome this problem ([8] pp. 110–114) and have been used in conjunction with SOM [21].

Various other distance/similarity measures have been used for sequence motifs in the SOM context. The simplest ones use word frequencies (e.g., dinucletide, trinucleotide, or longer oligonucleotide frequencies, [35,36,42]). Nucleotide motifs can also be codified into a 3D entity and Euclidian distance can be computed between two such recoded motifs [43,44].

Lorenzo-Redondo et al. [44] mapped each nucleotide to a tetrahedron in which the four vertices are occupied by four nucleotides so that a nucleotide can be represented as {x, y, z}. A nucleotide sequence of length L can then be represented as $\{x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_L, y_L, z_L\}$, making it easy to compute a distance between two sequences of the same length. The tetrahedron could be shaped

in such a way that the distance between A and G, and that between C and T, are smaller than that between a purine and a pyrimidine to accommodate the frequently observed transition bias where observed number of transitions can be nearly 80 times of the observed number of transversions [45]. Delgado et al. [43] used exactly the same approach to convert a sequence to a vector.

One may also use alignment scores derived from BLAST or FASTA, with the latter having been used in the SOM context [22]. FASTA algorithms with different word lengths for local string matching have been illustrated in detail before ([8] pp. 1–22).

Regardless of what distance/similarity one may use for fixed-length strings, the underlying SOM algorithm is the same. In what follows, I will illustrate the approach with PWM. The software SOMBRERO [18,19] implements SOM with PWM.

3. The Algorithmic Details of Self-Organizing Map (SOM)

3.1. Training Data

Training SOM with a set of numeric vectors has been illustrated in detail before ([8] pp. 231–250). Suppose that we have 150 genes whose expression is measured at seven time points, so an input vector could be {2, 1, 34, 37, 3, 2, 66}. For using SOM with sequence motifs, suppose we have a set of 150 nucleotide sequences of a length of 7.

3.2. SOM Grid Size and Initialization

We first need to decide the size of SOM, i.e., the number of nodes (neurons) to have and whether the nodes should be arranged in one dimension, two dimensions, or a higher number of dimensions. Most SOMs use a two-dimensional grid of nodes. Choosing the right number of nodes may be tricky. Too many nodes increase computation and may not achieve sufficient data reduction/simplification, but too few nodes may not provide sufficient fit to the data. The minimum is to have two nodes, which leads to binary classification, i.e., the input vectors will be assigned to one of the two nodes. The rule of thumb is to use the following equation to determine the number of nodes in SOM

$$N_{node} = 3\sqrt{N_{in}} \tag{4}$$

where N_{node} is the number of nodes in SOM rounded to integer, and N_{in} is the number of input vectors. After the first run, one may decide to increase or decrease N_{node} depending on how well the trained SOM fits the input data. We increase the number of neurons to improve the fit to the input data. For our data with 150 sequences, N_{node} = 36.7, so we may just use 36 nodes in a 6 × 6 configuration.

If we work with a set of numerical vectors, we would generate 36 random vectors for the 36 nodes as initial values. These nodes will change with the training process. For fixed-length nucleotide sequences, we will generate 36 random sequences of length 7 (same length as input sequences), with each nucleotide drawn randomly from the pool of nucleotides with frequencies being the same as that of the pooled input sequences. Table 2 (a) shows one such randomly generated sequence, together with the position weight matrix (PWM) derived from the sequences (Table 2 (b)). Each element in **PWM** is

$$\mathbf{PWM}_{i,S_i} = \log_2\left(\frac{P_{i,S_i}}{P_S}\right) \tag{5}$$

where i is site index (=1, 2, . . . , 7), S_i is the nucleotide (A, C, G, or T) at site i, P_{i,S_i} is the site-specific frequency of S_i at site i, and P_S is the background frequency of S. P_S values are typically the pooled frequencies from all input sequences, but can also be specified in different ways for different purposes [41]. Most SOM programs with this approach would demand that the user input background frequencies which could include not only 4 nucleotide or 20 amino acid (AA) frequencies, but also dinucleotide and trinucleotide frequencies or di-AA frequencies. For example, SOMBRERO [18,19], which is such a SOM program, would ask use to input a file containing these frequencies.

PWM in Table 2 (b) is computed after adding a pseudocount of 0.01 to each cell in Table 2 (a), with the background frequencies being 0.3, 0.2, 0.2, and 0.3 for A, C, G, and T, respectively. The pseudocount is necessary to avoid taking logarithm of zero. Note that we have 36 nodes, with each node represented in the form of Table 2. The same binary coding of sequences has been used in several previous studies [17–20,46,47].

Table 2. A matrix representation of sequence "ACCGTTA" (a). The resulting position weight matrix (b) is obtained after adding a pseudocount of 0.01 to each cell in (a), with background frequencies being 0.3, 0.2, 0.2, and 0.3 for A, C, G, and T, respectively.

(a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A	1	0	0	0	0	0	1
C	0	1	1	0	0	0	0
G	0	0	0	1	0	0	0
T	0	0	0	0	1	1	0
(b)							
A	1.695	−4.963	−4.963	−4.963	−4.963	−4.963	1.695
C	−4.379	2.280	2.280	−4.379	−4.379	−4.379	−4.379
G	−4.379	−4.379	−4.379	2.280	−4.379	−4.379	−4.379
T	−4.963	−4.963	−4.963	−4.963	1.695	1.695	−4.963

3.3. Update SOM

SOM is updated for each input vector or input sequence motif, and the updating involves three steps. The first is to identify the winning node for the input. The second is for the winning node to learn from the input. The third is for the neighbors of the winning node to learn from the input. This learning process continues until input vectors or sequence motifs result in negligible learning by SOM.

3.3.1. Identify the Winning Node

For input with numeric vectors, we compute the distance (e.g., Euclidean distance) between an input vector and each of the 36 node vectors, and the node with the smallest distance (or largest similarity index) to the input vector is the winning node and will learn from this input vector. Identification of the winning node with fixed-length nucleotide sequences is the same in principle but differs in operational details. Suppose our input sequence is "GCCATTA". We need to compute a distance or a similarity between this input sequence and each of the 36 nodes. We will use a similarity index called PWM score (*PWMS*) defined as

$$PWMS = \sum_{i=1}^{7} PWM_{i,S_i} \tag{6}$$

Given the input sequence of "GCCATTA", the *PWMS* from the PWM in Table 2 (b) is

$$PWMS = PWM_{1,G} + PWM_{2,C} + \ldots + PWM_{7,A} \approx 0.3016 \tag{7}$$

Note that *PWMS* is a similarity index. The larger the *PWMS*, the more similarity between the input sequence and the node. As we have 36 nodes, we need to compute 36 *PWMSs* to find the winning node which is the one with the largest *PWMS*. Also note that the input sequences do not need to be of the same length but need to be at least as long as the node site dimension. Take nucleotide frequencies for example. If the node is represented as a 4×7 matrix, then the input sequence should be at least seven bases long. Longer input sequences are fine because we can scan the input sequence with a sliding window of seven—i.e., we obtain $PWMS_1$ for sites 1–7, $PWMS_2$ for sites 2–8, and so on—and use the largest $PWMS_i$ as the similarity between the input sequence and the node.

3.3.2. Learning by Revising the Winning Node and Its Neighbors: Numeric Vectors

The winning node and its neighbors will learn from the input vector through the following learning function

$$w_i' = w_i(1 - \alpha) + p_i\alpha \quad (8)$$

where w_i refers to values in the winning node's vector, and p_i refers to values in the input vector. One could devise alternative learning functions different from that specified in Equation (8).

The α parameter in Equation (8) is the learning rate. An α equal to 0 implies $w_i' = w_i$, which means that the winning node does not learn from the input vector and will never change. An α equal to 1 implies $w_i = p_i$, which means that the winning node cannot retain any learned knowledge and will always mirror the features of the input vector. Thus, the α value should be greater than 0 but smaller than 1. In practise, α will initially be large (close to 1), but will diminish with each iteration.

The w_i' values will replace the original w_i values of the winning node, which completes the update of the winning node. The next step is to modify the neighbors of the winning node because the neighboring nodes will also learn from the input vector. Updating the values of neighbors is governed by the following learning function

$$w_i' = w_i(1 - \alpha_n) + p_i\alpha_n \quad (9)$$

where α_n is the learning rate for the neighbors. Again, one can use one of many possible alternative learning functions, but we will just use Equation (9) to keep things simple.

In practice, the learning function of neighbors depends on how we define neighbors, and α_n will be larger for immediate neighbors than for remote neighbors. For obvious reasons, they should also be smaller than α. If we designate α_{n1} as the learning rate for the immediate neighbors (i.e., nodes in physical contact with the winning nodes), α_{n2} as the learning rate for the neighbors of the immediate neighbors, and so on, then one simple way of choosing α_n values is to set $\alpha_{n1} = \alpha/2$, $\alpha_{n2} = \alpha_{n1}/2$, and so on.

For our illustration, we will just define each node to have a maximum of four neighbors, i.e., the one to its left, the one to its right, the one above it, and the one below it. Thus defined, we need only one α_n value which we will set to $\alpha/2$. A winning node in a corner will have only two neighbors, with one above it and one to its right. We repeat this with decreasing α and α_n values with each cycle of iteration until α equals a preset $\alpha_{\min} > 0$ (We do not want to decrease α to zero because SOM will stop learning when $\alpha = 0$).

How should we decrease α and α_n with each cycle of updating? There is no optimal way of decreasing α. In my SOM implementation in AMIADA [7], I used the following equation

$$Q = (1 - \frac{1}{N_v}) \quad (10)$$

where N_v is the number of input vectors. In our case, $N_v = 150$ and $Q = 0.9933$, i.e., α will be multiplied by Q after each cycle of iteration.

Continuing the learning process will eventually lead to convergence, i.e., when the values in the nodes do not change any more or the change is smaller than a pre-fixed small value in two consecutive cycles of iteration. This final set of nodes is a trained SOM. Different nodes in the final trained SOM have different properties reflecting our data structure. For example, some vectors may have high values, some low values, some increasing with time (if the vector values represent measurements at specific time points), some decreasing with time, etc.

The trained SOM can now be used for classification. During the classification stage, the node values do not change. An input vector is assigned to a node if its distance is the smallest to this node which is often referred to as the host node of the input vector. However, before we use the trained SOM to do the classification of new genes, it is crucial to check how well the SOM fits the training data.

This is typically done by first assigning the input vectors to their host nodes, and then computing the distance (or its square) between each gene and its host node.

3.3.3. Learning by Revising the Winning Node and its Neighbors: Fixed-Length Sequences

The handling of fixed-length nucleotide sequences is the same in principle as numeric vectors. Suppose that the input sequence is "GCCATTA" and the winning node is the one in Table 2 (a), then we simply add the new sequence to the original to obtain Table 3 (a), with the corresponding new PWM shown in Table 3 (b). Thus, the winning node, starting from a random sequence, will change to become a sequence profile with site-specific frequencies depending on the input sequences. Its PWM will depend on the changing site-specific frequencies and the constant background frequencies. For updating the neighboring nodes, we simply modify them by adding half of the input sequences, i.e., 0.5G, 0.5C, and so on. The rest of the computation is the same as with numeric vectors as input.

Table 3. Updating the node in Table 2 (a) by the new input sequence "GCCATTA" and the resulting position weight matrix (PWM) obtained in (b) after adding a pseudocount of 0.01 to each cell in (a).

(a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A	1	0	0	1	0	0	2
C	0	2	2	0	0	0	0
G	1	0	0	1	0	0	0
T	0	0	0	0	2	2	0
(b)							
A	0.723	−5.935	−5.935	0.723	−5.935	−5.935	1.716
C	−5.350	2.301	2.301	−5.350	−5.350	−5.350	−5.350
G	1.308	−5.350	−5.350	1.308	−5.350	−5.350	−5.350
T	−5.935	−5.935	−5.935	−5.935	1.716	1.716	−5.935

4. The Fit of SOM to Input Data

Measuring the fit of SOM to the input data has an easy side and a difficult side. The easy side is to characterize the deviation of each input vector to the node vector. This can be measured by Kohonen's quantization error [1] or by the conventional residual sum of squares (RSS). Such deviations between input vectors and SOM nodes can be reduced by increasing the grid size of SOM at the risk of overfitting. There are model selection protocols [48] that can help determine the optimal SOM grid size. The difficult side is to characterize the "data configuration" and the "SOM configuration" [49–51]. Specifically, what information is lost when reducing the multi-dimensional data to two dimensions in a typical SOM grid, and which dimension has lost significant information?

One approximation to this problem is the topographic product [50] based on a simple rationale. That is, if pairs of points are the closest to each other in the original multidimensional space, and if they are also mapped to the same (or neighboring) SOM nodes, then at least the neighborhood relationship is preserved. Operationally, if pairwise distances of points in the original multi-dimensional space correlate highly with pairwise distance of points mapped to the SOM nodes, then the mapping is good, otherwise it is not. Topological measures based on this rationale are termed the 'neighborhood preserving approach' [52–54]. SOM performs well in preserving neighborhood relationships [1]. However, this neighborhood preservation approach is associated with two types of uncertainties. The first is how well a particular characterization of neighborhood relationships—e.g., the neighborhood function expressed in Equation (4) in Villman [54]—captures the true neighborhood relationships. The second is how well neighborhood relationships can summarize the global topology. Increasing SOM dimension from a 2-D grid to higher dimensions will preserve neighborhood relationships better [49] but this is somewhat self-defeating because the main objective of SOM is to visualize spatial relationships in low-dimensions.

Assuming that data topology is well represented by SOM, Kaski and Lagus [51] further developed an index for measuring similarity in topology between two data sets. The rationale can be simplified as follows. Suppose we train and map data set 1 to SOM1 of size (m, n) and designate the number of data points mapped to SOM1 nodes as $N_{1,1}, N_{1,2}, \ldots, N_{2,1}, N_{2,2}, \ldots, N_{m,n}$. We can now map points in data set 2 to SOM1 nodes and obtain another set of $N_{i,j}$ values for data set 2. The correlation between the two sets of $N_{i,j}$ values is a measure of topological similarity between the two data sets. We can also start with data set 2, and train and map the data points to SOM2, and then use SOM2 to obtain two sets of $N_{i,j}$ values and compute their correlations. Such a topological similarity index has a simple interpretation. For example, if our two data sets are transcription factor binding sites from human and mouse, and we found nearly perfect correlation in $N_{i,j}$ values between the two data sets, then we may conclude that gene regulation mechanisms are nearly identical between the two mammalian species.

All these developments are interesting, but their significance to biologists is not clear. Most computational methods adopted by biologists are simply by trial and error and not by mathematical justification. In the case of SOM, mathematical justification is difficult anyway [55]. Computational methods in biology are filtered by 'natural selection', i.e., methods generating biological insights are preserved and those not are eliminated.

5. Software Implementing SOM with PWM

The software tools implementing SOM with the PWM approach are already available, e.g., SOMBRERO [18,19]. Note that SOMBRERO is different from SOMbrero [56] which is an R package for SOM but does not deal specifically with sequence motif discovery and characterization. There are several other papers reporting SOM implementation involve PWM or variations of PWM, e.g., SOMEA [21], READ [47], and READ_{CSF} [46], but I was not able to find a working program.

6. Conclusions

SOM is a versatile neural network algorithm that can be used in a wide variety of biological data analysis. My illustration of its application for characterizing fixed-length sequences should pave the way for using SOM to analyze a variety of sequence motifs involved in gene regulation, facilitating the identification of new drug targets.

Acknowledgments: This study is funded by the Discovery Grant from Natural Science and Engineering Research Council of Canada (RGPIN/261252-2013). Three anonymous reviewers noted numerous problems in a previous version, leading to substantial improvement of the manuscript.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Kohonen, T. *Self-Organizing Maps*; Springer: Berlin, Germany, 2001; Volume 30, p. 501.
2. Ordway, J.M.; Fenster, S.D.; Ruan, H.; Curran, T. A transcriptome map of cellular transformation by the fos oncogene. *Mol. Cancer* **2005**, *4*, 19. [CrossRef] [PubMed]
3. Covell, D.G.; Wallqvist, A.; Rabow, A.A.; Thanki, N. Molecular classification of cancer: Unsupervised self-organizing map analysis of gene expression microarray data. *Mol. Cancer Ther.* **2003**, *2*, 317–332. [PubMed]
4. Xiao, L.; Wang, K.; Teng, Y.; Zhang, J. Component plane presentation integrated self-organizing map for microarray data analysis. *FEBS Lett.* **2003**, *538*, 117–124. [CrossRef]
5. Wang, J.; Delabie, J.; Aasheim, H.; Smeland, E.; Myklebost, O. Clustering of the SOM easily reveals distinct gene expression patterns: Results of a reanalysis of lymphoma study. *BMC Bioinform.* **2002**, *3*, 36. [CrossRef]
6. Toronen, P.; Kolehmainen, M.; Wong, G.; Castren, E. Analysis of gene expression data using self-organizing maps. *FEBS Lett.* **1999**, *451*, 142–146. [CrossRef]
7. Xia, X.; Xie, Z. AMADA: Analysis of microarray data. *Bioinformatics* **2001**, *17*, 569–570. [CrossRef]
8. Xia, X. *Bioinformatics and the Cell: Modern Computational Approaches in Genomics, Proteomics and Transcriptomics*; Springer: New York, NY, USA, 2007; p. 349.

9. Kozak, M. Possible role of flanking nucleotides in recognition of the AUG initiator codon by eukaryotic ribosomes. *Nucleic Acids Res.* **1981**, *9*, 5233–5252. [CrossRef] [PubMed]
10. Xia, X. The +4G site in Kozak consensus is not related to the efficiency of translation initiation. *PLoS ONE* **2007**, *2*, e188. [CrossRef] [PubMed]
11. Ma, P.; Xia, X. Factors affecting splicing strength of yeast genes. *Comp. Funct. Genom.* **2011**, *2011*. [CrossRef]
12. Vlasschaert, C.; Xia, X.; Gray, D.A. Selection preserves Ubiquitin Specific Protease 4 alternative exon skipping in therian mammals. *Sci. Rep.* **2016**, *6*, 20039. [CrossRef] [PubMed]
13. Sidrauski, C.; Cox, J.S.; Walter, P. tRNA ligase is required for regulated mRNA splicing in the unfolded protein response. *Cell* **1996**, *87*, 405–413. [CrossRef]
14. Sidrauski, C.; Walter, P. The transmembrane kinase Ire1p is a site-specific endonuclease that initiates mRNA splicing in the unfolded protein response. *Cell* **1997**, *90*, 1031–1039. [CrossRef]
15. Gonzalez, T.N.; Sidrauski, C.; Dorfler, S.; Walter, P. Mechanism of non-spliceosomal mRNA splicing in the unfolded protein response pathway. *EMBO J.* **1999**, *18*, 3119–3132. [CrossRef] [PubMed]
16. Kaufman, R.J. Stress signaling from the lumen of the endoplasmic reticulum: Coordination of gene transcriptional and translational controls. *Genes Dev.* **1999**, *13*, 1211–1233. [CrossRef] [PubMed]
17. Mahony, S.; Benos, P.V.; Smith, T.J.; Golden, A. Self-organizing neural networks to support the discovery of DNA-binding motifs. *Neural Netw.* **2006**, *19*, 950–962. [CrossRef] [PubMed]
18. Mahony, S.; Golden, A.; Smith, T.J.; Benos, P.V. Improved detection of DNA motifs using a self-organized clustering of familial binding profiles. *Bioinformatics* **2005**, *21* (Suppl. 1), i283–i291. [CrossRef] [PubMed]
19. Mahony, S.; Hendrix, D.; Golden, A.; Smith, T.J.; Rokhsar, D.S. Transcription factor binding site identification using the self-organizing map. *Bioinformatics* **2005**, *21*, 1807–1814. [CrossRef] [PubMed]
20. Mahony, S.; Hendrix, D.; Smith, T.J.; Golden, A. Self-Organizing Maps of Position Weight Matrices for Motif Discovery in Biological Sequences. *Artif. Intell. Rev.* **2005**, *24*, 397–413. [CrossRef]
21. Lee, N.K.; Wang, D. SOMEA: Self-organizing map based extraction algorithm for DNA motif identification with heterogeneous model. *BMC Bioinform.* **2011**, *12* (Suppl. 1), S16. [CrossRef] [PubMed]
22. Kohonen, T.; Somervuo, P. How to make large self-organizing maps for nonvectorial data. *Neural Netw.* **2002**, *15*, 945–952. [CrossRef]
23. Jukes, T.H.; Cantor, C.R. Evolution of protein molecules. In *Mammalian Protein Metabolism*; Munro, H.N., Ed.; Academic Press: New York, NY, USA, 1969; pp. 21–123.
24. Kimura, M. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* **1980**, *16*, 111–120. [CrossRef] [PubMed]
25. Hasegawa, M.; Kishino, H. Heterogeneity of tempo and mode of mitochondrial DNA evolution among mammalian orders. *Jpn. J. Genet.* **1989**, *64*, 243–258. [CrossRef]
26. Kishino, H.; Hasegawa, M. Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea. *J. Mol. Evol.* **1989**, *29*, 170–179. [CrossRef]
27. Hasegawa, M.; Kishino, H.; Yano, T. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.* **1985**, *22*, 160–174. [CrossRef]
28. Tamura, K.; Nei, M. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol. Biol. Evol.* **1993**, *10*, 512–526. [PubMed]
29. Lanave, C.; Preparata, G.; Saccone, C.; Serio, G. A new method for calculating evolutionary substitution rates. *J. Mol. Evol.* **1984**, *20*, 86–93. [CrossRef] [PubMed]
30. Tavaré, S. *Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences*; American Mathematical Society: Providence, RI, USA, 1986; Volume 17, pp. 57–86.
31. Tamura, K.; Nei, M.; Kumar, S. Prospects for inferring very large phylogenies by using the neighbor-joining method. *Proc. Natl. Acad. Sci. USA* **2004**, *101*, 11030–11035. [CrossRef] [PubMed]
32. Xia, X. Information-theoretic indices and an approximate significance test for testing the molecular clock hypothesis with genetic distances. *Mol. Phylogenet. Evol.* **2009**, *52*, 665–676. [CrossRef] [PubMed]
33. Xia, X. DAMBE5: A comprehensive software package for data analysis in molecular biology and evolution. *Mol. Biol. Evol.* **2013**, *30*, 1720–1728. [CrossRef]
34. Xia, X. DAMBE6: New tools for microbial genomics, phylogenetics and molecular evolution. *J. Hered.* **2017**, *108*, 431–437. [CrossRef] [PubMed]
35. Samsonova, E.V.; Kok, J.N.; Ijzerman, A.P. TreeSOM: Cluster analysis in the self-organizing map. *Neural Netw.* **2006**, *19*, 935–949. [CrossRef] [PubMed]

36. Abe, T.; Kanaya, S.; Kinouchi, M.; Ichiba, Y.; Kozuki, T.; Ikemura, T. Informatics for unveiling hidden genome signatures. *Genome Res.* **2003**, *13*, 693–702. [CrossRef] [PubMed]
37. Xia, X. PhyPA: Phylogenetic method with pairwise sequence alignment outperforms likelihood methods in phylogenetics involving highly diverged sequences. *Mol. Phylogenet. Evol.* **2016**, *102*, 331–343. [CrossRef] [PubMed]
38. Staden, R. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res.* **1984**, *12*, 505–519. [CrossRef]
39. Stormo, G.D.; Schneider, T.D.; Gold, L. Quantitative analysis of the relationship between nucleotide sequence and functional activity. *Nucleic Acids Res.* **1986**, *14*, 6661–6679. [CrossRef] [PubMed]
40. Hertz, G.Z.; Hartzell, G.W.; Stormo, G.D. Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci.* **1990**, *6*, 81–92. [CrossRef] [PubMed]
41. Xia, X. Position Weight Matrix, Gibbs Sampler, and the Associated Significance Tests in Motif Characterization and Prediction. *Scientifica* **2012**, *2012*, 917540. [CrossRef]
42. Iwasaki, Y.; Wada, K.; Wada, Y.; Abe, T.; Ikemura, T. Notable clustering of transcription-factor-binding motifs in human pericentric regions and its biological significance. *Chromosome Res.* **2013**, *21*, 461–474. [CrossRef] [PubMed]
43. Delgado, S.; Moran, F.; Mora, A.; Merelo, J.J.; Briones, C. A novel representation of genomic sequences for taxonomic clustering and visualization by means of self-organizing maps. *Bioinformatics* **2015**, *31*, 736–744. [CrossRef] [PubMed]
44. Lorenzo-Redondo, R.; Delgado, S.; Moran, F.; Lopez-Galindez, C. Realistic three dimensional fitness landscapes generated by self organizing maps for the analysis of experimental HIV-1 evolution. *PLoS ONE* **2014**, *9*, e88579. [CrossRef] [PubMed]
45. Xia, X.; Hafner, M.S.; Sudman, P.D. On transition bias in mitochondrial genes of pocket gophers. *J. Mol. Evol.* **1996**, *43*, 32–40. [CrossRef] [PubMed]
46. Tapan, S.; Wang, D. A Further Study on Mining DNA Motifs Using Fuzzy Self-Organizing Maps. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 113–124. [CrossRef] [PubMed]
47. Wang, D.; Tapan, S. A robust elicitation algorithm for discovering DNA motifs using fuzzy self-organizing maps. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 1677–1688. [CrossRef] [PubMed]
48. Burnham, K.P.; Anderson, D.R. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*; Springer: New York, NY, USA, 2002.
49. Bauer, H.; Riesenhuber, M.; Geisel, T. Phase diagrams of self-organizing maps. *Phys. Rev. E* **1996**, *54*, 2807–2810. [CrossRef]
50. Bauer, H.-U.; Pawelzik, K.R. Quantifying the neighborhood preservation of self-organizing feature maps. *Neural Netw.* **1992**, *3*, 570–579. [CrossRef] [PubMed]
51. Kaski, S.; Lagus, K. Comparing self-organizing maps. In *Artificial Neural Networks, In Proceedings of the ICANN 96, 1996 International Conference, Bochum, Germany, 16–19 July 1996*; von der Malsburg, C., von Seelen, W., Vorbrüggen, J.C., Sendhoff, B., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; pp. 809–814.
52. Villmann, T.; Der, R.; Herrmann, M.; Martinetz, T. Topology Preservation in Self-Organizing Feature Maps: General Definition and Efficient Measurement. In *Fuzzy Logik*; Reusch, B., Ed.; Springer: Berlin/Heidelberg, Germany, 1994; pp. 159–166.
53. Villmann, T.; Der, R.; Martinetz, T. A Novel Approach to Measure the Topology Preservation of Feature Maps. In Proceedings of the International Conference on Artificial Neural Networks (ICANN'94), Sorrento, Italy, 26–29 May 1994; Marinaro, M., Morasso, P.G., Eds.; Springer: London, UK, 1994; Volume 1, Parts 1 and 2, pp. 298–301.
54. Villmann, T.; Der, R.; Herrmann, M.; Martinetz, T.M. Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Trans. Neural Netw.* **1997**, *8*, 256–266. [CrossRef] [PubMed]
55. Hammer, B. Challenges in Neural Computation. *Künstl Intell.* **2012**, *26*, 333–340. [CrossRef]
56. Boelaert, J.; Bendhaiba, L.; Olteanu, M.; Villa-Vialaneix, N. SOMbrero: An R Package for Numeric and Non-numeric Self-Organizing Maps. In *Advances in Self-Organizing Maps and Learning Vector Quantization*; Villmann, T., Schleif, F.-M., Kaden, M., Lange, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 219–228.