The Time Adaptive Self-Organizing Map Is a Neural Network Based on Artificial Immune System

Hamed Shah-Hosseini

Abstract—In this paper, the similarities between the mechanisms used in the TASOM (Time Adaptive Self-Organizing Map) neural network and AIS (Artificial Immune Systems) are analyzed. To demonstrate the similarities, AIS mechanisms are incorporated into the TASOM network such as the weight updating is replaced by a mutation mechanism. Learning rate and neighborhood sizes are also replaced by the clonal selection process used in AIS. This new network is called TAISOM. Experimental results with TAISOM are implemented for uniform and Gaussian distributions for one and two-dimensional lattices of neurons. These experiments show that TAISOM learns its environment as expected so that neurons fill the environments quite well and the neurons also preserve the topological ordering.

I. INTRODUCTION

The vertebrate immune system is one of the complex systems of the living organisms. Scientists have used ideas of immune systems to develop artificial systems similar to them, which is called Artificial Immune System (AIS) [1]-[3]. Here, these AIS ideas are compared to the mechanism of a neural network and to see how close AIS and the neural network act similarly. This neural network is the Time Adaptive Self-Organizing Mapping (TASOM).

TASOM uses adaptive learning parameters [4] whose values change based on the changes of the environment and the behavior of input vectors as time passes. It has been shown that the TASOM network is able to work in stationary as well as non-stationary environments. It has been used for adaptive shape representation and adaptive segmentation [5], bilevel thresholding [6], adaptive pattern classification [7], adaptive principal components analysis [8], automatic multilevel thresholding [9], and active contour modeling [10].

The ideas of the AIS mechanism are incorporated into the TASOM network and it is shown that this modified network works surprisingly well and the neurons learn their environment and they keep the topological ordering which is expected from any original TASOM network.

Next section shortly introduces the Artificial Immune Systems and the clonal selection. After that, the mechanism of original TASOM is compared to the mechanisms exist in AIS and the similarity between them are shown. In section IV, experimental results are done for the combined network of TASOM and AIS which is called TAISOM. Concluding remarks are mentioned in the last section.

II. ARTIFICIAL IMMUNE SYSTEMS

Every foreign element that is recognized by the immune system is called an antigen. The cells that belong to our body and are harmless are called “self”, while the foreign elements are called “nonself”. The immune system can distinguish between what is “self” from what is “nonself”.

There are two major groups of immune cells, known as B-cells and T-cells. B-cells recognize antigens free in solution using the molecules on their surfaces which are called antibodies. In contrast, T-cells require antigens to be presented by other accessory cells to recognize the antigens.

When a B-cell receptor recognizes a nonself antigen with a certain affinity, it proliferates and clones itself but with a certain degree of mutation. This process is called “clonal selection”. The proliferation rate of the immune cell is proportional to its affinity with the selective antigen which means that the higher the affinity, the higher the number of offspring generated. In contrast to the proliferation rate, the mutation suffered by each immune cell during reproduction is inversely proportional to the affinity of the cell receptor with the antigen which means the higher the affinity, the smaller the mutation rate.

III. TASOM IN VIEW OF ARTIFICIAL IMMUNE SYSTEMS

The mechanism of a TASOM network is summarized in three parts. Part one deals with learning rate updating. Part two is about the neighborhood size updating. Finally, part three discusses about the learning rule for weight updating of neurons.

A. Learning Rate Updating in AIS View

When an input data enters into a TASOM network, several parameters are updated using the input data. The learning rate is among those parameters. The learning rate change is often nonlinearly proportionate to the distance between the current input vector and the weight vector of the current winning neuron. Thus, the longer the distance between the input and the weight of the winning neuron, the higher the value of the learning rate will be. This expression reminds us of the mechanism that exists in the Immune System.

As it is mentioned for clonal selection in AIS, the mutation rate of an immune cell during reproduction when
facing an antigen, is proportional to a distance measure of the cell receptor with the antigen. In other words, if the antigen is less similar to the immune cell receptor, the mutation rate is higher. Assume that the antigen is correspondent to the current input vector for a TASOM network. The antigen stimulates its nearby cell (or cells). This immune cell is assumed to be correspondent to the winning neuron of the TASOM network. Thus the weight vector of this winning neuron corresponds to the immune cell receptor.

Based on the above assumptions, the mutation rate mechanism of an immune cell is translated to the learning rate mechanism of the TASOM network. The learning rate of TASOM changes by the distance between weight vector of winning neuron and the current input vector. The mutation rate in AIS changes by the distance between the immune cell receptor and the current antigen. The analogy assumed earlier reveals that these two mechanisms work almost the same but in different environments: one for an artificial neural network, TASOM, and the other for AIS, clonal selection.

The aforementioned discussion on the mutation rate in clonal selection and learning rate in TASOM may be summarized as follows:

\[
\text{winning \_ neuron} = \text{immune \_ cell} \quad (1) \\
\text{input \_ vector} \approx \text{antigen} \quad (2) \\
\text{weight \_ of (winning \_ neuron)} \approx \text{receptor \_ of (immune \_ cell)} \quad (3)
\]

Also

\[
\Delta \text{mutation (TASOM)} \approx \Delta \text{mutation (AIS)} \quad (4)
\]

The notation \( \approx \) denotes the analogy. The symbol \( \Delta \) represents the rate of change.

**B. The neighborhood Size Updating in AIS View**

Neighborhood size is another parameter of a TASOM network which is updated as soon as the current input vector recognizes the winning neuron. The neighborhood size of the winning neuron is nonlinearly proportionate to the distance between the weight of the winning neuron and the weights of its neighboring neurons. If this distance increases so does the neighborhood size, and if it decreases so does the neighborhood size.

Similar to the discussion for learning rate updating, we consider the antigen in AIS analogous to the input vector of a TASOM network. Moreover, the weights of neurons are considered to be analogous to the receptors of immune cells. The winning neuron is analogous to the immune cell nearest to the antigen.

In the clonal selection, the proliferation rate of an immune cell is proportional to the affinity of its receptor with the antigen. The number of copies of the immune cell increases as the affinity increases and vice versa. Therefore, the lower the distance between the antigen and the receptor of the immune cell, the higher the proliferation rate of the immune cell will be. To translate this mechanism of AIS into TASOM, we use the analogies made earlier. According to the AIS mechanism, when the input vector (the antigen) is near to the winning neuron’s weight (the receptor of the immune cell); the number of neurons similar to the antigen should increase. In the standard TASOM we keep the number of neurons fixed. Thus, to imitate this increase in the number of neurons (or immune cells) we need to draw the neighboring neurons toward the antigen (or toward the winning neuron). This means that the neighborhood size of the winning neuron should increase. Thus, the mechanism of AIS says that the neighborhood size should increase as the input vector (the antigen) is so near to the winning neuron’s weight. When the antigen is near to the weight of the winning neuron, the antigen can be replaced by the winning neuron’s weight.

Moreover, when the antigen is so near to the weight of winning neuron, it means that the antigen is so far from other immune cells (other neighboring neurons). Now that we have approximated the antigen with the weight of winning neuron, the distance between the input vector and its neighboring weights is translated to the distance between the weight of winning neuron and its neighboring weights, which in fact is the mechanism of any TASOM network for neighborhood size updating.

The following expressions summarize the discussion made about the analogy between the neighborhood size updating of TASOM and proliferation mechanism of AIS:

\[
\| \text{antigen} - \text{weight \_ of (winning \_ neuron)} \| \leq \varepsilon \quad (5)
\]

which \( \varepsilon \) is a small non-negative number. From (2) and above formula we may obtain

\[
\| \text{input \_ vector} - \text{weight \_ of (winning \_ neuron)} \| \leq \varepsilon \quad (6)
\]

When input vector is so near to the winning neuron, then it is far from the neighboring neurons of the winning neuron. So we may conclude from (6)

\[
\| \text{input \_ vector} - \text{weight \_ of (any \_ neighboring \_ neuron)} \| \geq \theta \quad (7)
\]

where \( \theta \) is a large positive number with respect to \( \varepsilon \). We can approximate the input vector with the weight of the winning neuron using (6) as follows

\[
\text{input \_ vector} \approx \text{weight \_ of (winning \_ neuron)} \quad (8)
\]

where notation \( \approx \) means that values on both sides of the notation are almost equal.

Replacing input vector in (7) by its approximate in (8) we can obtain

\[
\| \text{weight \_ of (winning \_ neuron)} - \text{weight \_ of (any \_ neighboring \_ neuron)} \| \geq \theta \quad (9)
\]

**C. The TASOM Learning Rule Weight Updating in AIS View**

The weights in neurons of TASOM are updated by a competitive learning rule with adaptive learning rates and neighborhood sizes. The adaptive learning parameters,
learning rates and neighborhood sizes, are formerly discussed. The weights of the winning neuron \( i \) and its neighbors are updated in a way that the weight vectors move toward the current input vector.

\[
\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + k_{ij}(n+1) \{ \mathbf{x}(n) - \mathbf{w}_j(n) \} \tag{10}
\]

where \( j \) represents the winning neuron and its neighbors and \( i \) represents only the winning neuron. For simplicity of expression, we have assumed that \( k_{ij}(n+1) = \eta_j(n+1) h_{ij}(n+1) \). Here, \( \eta_j(.) \) denotes the learning rate of neuron \( j \) and \( h_{ij}(.) \) is the neighborhood function around neuron \( i \) for neuron \( j \).

If the same input enters to the network repeatedly, the weight vectors of TASOM move closer and closer to the input vector. This is a property of most SOM-based (competitive-based) neural networks. We remind the analogy made earlier between neurons and immune cells. With the help of the analogies made earlier, this property means that the receptors of immune cells should change in a way to better match the nearby antigen.

If we remember the mechanisms of mutation and proliferation in AIS, we realize that the hypothetical mechanism made by the analogies between TASOM and AIS in fact exist in immune cells. The immune cells proliferate with some degree of mutation as soon as they see an antigen nearby. Those who better match the antigen will survive. Consequently, the receptors of immune cells better match the antigens. A fact that has been obtained from the analogies between TASOM weights updating to the matching of the antigens. A fact that has been obtained from the analogies between TASOM weights updating to the matching of the antigens.

\[
\text{receptor}_{\text{immune}}(\text{Cell}(j))(n+1) = \text{best \_ mutation}(\text{receptor}_{\text{immune}}(\text{Cell}(j))(n)) \tag{11}
\]

IV. EXPERIMENTAL RESULTS

In this section, we modify the original TASOM using the ideas taken from AIS and the analogies provided in the previous section and employ this modified TASOM in some benchmark environments. We call the modified TASOM “TAISOM” which stands for Time Adaptive Artificial Immune System Self-Organizing Map.

A. The Modified TASOM Network: TAISOM

Here, we provide the TAISOM algorithm which is a combination of TASOM network and AIS mechanism with the help of the analogies made in this paper. The proposed TAISOM algorithm is summarized in the following steps:

1. Initialization: Choose some small values for the initial weight vectors \( \mathbf{w}_j(0) \), where \( j = 1, 2, ..., N \); and \( N \) is the number of neurons in the lattice. In this paper, all initial weights are simply set to zero unless it is mentioned otherwise. The constant parameters \( \alpha_s \) and \( \beta_s \) can have any values between zero and one. In this paper, we set \( \alpha_s = \beta_s = 0.05 \). Other constant parameters are: \( N_c \) for neighborhood radius updating, \( P_c \) for pool-size updating, and \( U_c \) for weight updating which can be regarded as a mutation step. Here, we set \( P_c = 3 \), \( U_c = 0.01 \). Also, \( N_c \) is set to \( \frac{N}{5} \) and \( \sqrt{2N} \) for one and two-dimensional lattices, respectively.

The components \( s_k(0) \) of the scaling vector \( \mathbf{s}(0) = [s_1(0), ..., s_p(0)]^T \) should be set to small positive values, where \( p \) is the dimension of the input and weight vectors. The parameters \( E_k(0) \) and \( E_2(0) \) may be initialized with some small random values (near zero).

2. Sampling. Get the next input vector \( \mathbf{x}(n) \) from the environment.

3. Similarity matching. Find the winning neuron \( i(\mathbf{x}) \) at time \( n \), using the minimum-distance Euclidean norm:

\[
i(i, j) = \arg \min_j \| \mathbf{x}(n) - \mathbf{w}_j(n) \| \quad \text{and} \quad j = 1, 2, ..., N \tag{12}
\]

where \( \| \mathbf{x}(n) - \mathbf{w}_j(n) \| = \sqrt{\sum_k (x_k(n) - w_{kj}(n))^2} \).

4. Updating the scaling vector: Adjust the scaling value \( s_l(n) \) by the following equations:

\[
s_l(n+1) = \sqrt{\sum_k s_k(n+1)^2} \tag{13}
\]

where \( s(n) = [s_1(n), ..., s_k(n), ..., s_p(n)]^T \) such that

\[
s_k(n+1) = \sqrt{(E_2_k(n+1) - E_k(n+1)^2)^+}, \quad E_2_k(n+1) = E_2_k(n) + \alpha_s (x^2_k(n) - E_2_k(n)), \quad E_k(n+1) = E_k(n) + \beta_s (x^2_k(n) - E_k(n)), \quad (z)^+ = \max(z, 0).
\]

5. Neighborhood size updating: The neighborhood set \( \Lambda_i(n) \) of the winning neuron \( i \) is updated using the scaled distance between the weight of the winning neuron \( \mathbf{w}_i(n) \) and the current input vector \( \mathbf{x}(n) \):

\[
\Lambda_i(n+1) = \{ j \in N \mid d(i, j) \leq R_i(n+1) \} \tag{14}
\]

where \( R_i(n+1) = N_c \cdot \frac{\| \mathbf{x}(n) - \mathbf{w}_i(n) \|}{s_l(n+1) + \| \mathbf{x}(n) - \mathbf{w}_i(n) \|} \).

For simplicity, a memoryless neighborhood updating rule has been used for the neighborhood radius \( R_i(n) \). The notation \( d(i, j) \) is the distance between two neurons \( i \) (the winning neuron) and \( j \) in the lattice of the TASOM network. In a one-dimensional lattice: \( d(i, j) = |i - j| \) where \( |.| \) denotes the absolute value. In a two-dimensional lattice: \( d(i, j) = \|i - j\| \) where \( \| \| \) denotes the Euclidian distance between neurons \( i = (i_1, i_2) \) and \( j = (j_1, j_2) \).
The neighborhood radius of neuron \( i, R_i(n+1) \), is increased by the scaled distance between the current input vector and the winning neuron. It should be mentioned that any non-decreasing function of the scaled distance may be used for the neighborhood radius updating.

6. Weight updating: In this step, the weights of all neurons \( j \) in the neighborhood set \( \Lambda_i(n+1) \) are updated using the following two sub-steps:

6.1. Pool-size updating: The number of mutated neurons \( j \) in the mutation pool is updated using the scaled distance of neuron \( j \) and the current input vector \( x(n) \):

\[
poolsize_j(n+1) = P_c \cdot \left( 1 + \frac{\|x(n) - w_j(n)\|}{sl(n+1)} \right)
\]

Again, for simplicity, a memoryless updating rule for pool size is used here. The pool size of the neuron is thus increased by the scaled distance between the current input vector and the weight vector of the neuron.

6.2. Neuron weight updating: The weight vector of neuron \( j \) is updated by using a mutation mechanism of AIS as follows:

\[
w_j(n+1) = w_{m_j}(n+1)
\]

such that \( r^* = \arg \min_j \|w_j(n) - w_{m_j}(n+1)\| \)

where \( w_{m_j}(n+1) = w_j(n) + U_c \cdot \Delta w_m \) for \( r = 1, \ldots, \text{poolsize}_j(n+1) \); and the mutation vector

\[
\Delta w_m = [\Delta w_{1m}, \ldots, \Delta w_{pm}]^T
\]

where \( \Delta w_{im} = (x_i(n) - w_{j,k}(n)) \) such that \( k \) is a discrete uniform random variable from the range set \( \{1,2,\ldots,p\} \) and \( \Delta w_{im} = 0 \) for all \( i \neq k \).

7. Go to step 2 to get the next input vector.

The following section is for doing experiments with the above algorithm using several distributions.

B. Experiments with the Proposed TAISOM for Uniform and Gaussian Distributions

A one-dimensional lattice of 100 neurons is used to learn a uniform distribution in the region \([0,1] \times [0,1]\). The proposed algorithm is used for modifying the weights producing the intermediate and final weights of Figs. 1-3 after 1000, 50000, and 700000 iterations, respectively. In the figures, weights of neurons are shown by star signs. As it is seen, the topological ordering is preserved here because the one-dimensional lattice doesn‘t cross itself in the input space. Another point is that the weights are distributed uniformly in the input space which help the TAISOM network better approximate the input environment.

A similar experiment is implemented with 100 neurons by a Gaussian distribution with mean \( m = 0 \) and variance \( \nu = 1 \). The intermediate and final weights are shown in Figs. 4-6 after 1000, 50000, and 300000 iterations, respectively.

The weights are distributed similar to the concentration of the Gaussian distribution such that more neurons are concentrated around the point \((0,0)\). The chain of neurons does not cross itself so the topological ordering is also preserved.

For two-dimensional lattices, two experiments are implemented one with the uniform distribution and the other one with the Gaussian distribution. The intermediate and final weights are shown in Figs. 7-9 and 10-12, respectively. Again, it is seen that lattices don’t twist thus keeping topological ordering and the weights are almost distributed according to the concentration of input vectors.

The results obtained here are very similar to those obtained by TASOM [4], [5]. This is very interesting because we have replaced the three most important updating rules of the standard TASOM with some AIS-based rules, and still obtaining comparable results. One conclusion is that TAISOM is virtually equivalent to TASOM confirming that TAISOM may be considered as an AIS-based neural network.

The experiments of Figs. 1-3 are repeated again in Figs. 13-16 with the difference that the initial weights are randomly selected from the region \([0,1] \times [0,1]\). As it is seen, the weights of the one-dimensional lattice gradually fill the input space and at the same time, the lattice frees itself from the initial twisted situation. The weights finally stabilize as shown in Fig. 16 while preserving the topological ordering.

It is shown in the literature [11] that SOM networks are not faithful well enough to their input distributions so the standard SOM should not be used for density matching unless some modifications are incorporated into the networks. However, by changing a constant parameter, the TAISOM can be used for density matching. This property comes with a loss of losing topological ordering. Figs. 17 and 18 show a two-dimensional lattice of 100 neurons with the same initialization, constant parameters, and input distribution of the experiment of Figs. 10-12, except that the parameter \( N_c = 3\sqrt{2} \). The stabilized weights shown in Fig. 17 clearly follow the Gaussian distribution of the input environment such that more neurons are concentrated around the origin and the number of neurons decreases with the distance from the origin. However, as seen in Fig. 18, the topological ordering of the neurons of Fig. 17 is full of crossing. This experiment confirms that in order to obtain density matching, the topological ordering is weakened and vice versa. Thus, the appropriate tuning of the parameters of the TAISOM algorithm depends on what application is to be solved by TAISOM.

V. CONCLUSION

Proliferation and mutation mechanisms are among those ones which are claimed to be analogous to the TASOM learning parameters and weight updating methods. The claims are tested with the proposed TAISOM network for
two different environments in which both one-dimensional and two-dimensional lattices of neurons are tested. The results are very close to those which are obtained by the standard TASOM network [4], [5]. The similarities suggest that AIS concepts are comparable to the ideas used in the standard TASOM. This interesting fact may help both TASOM and AIS for further developments. However, more experiments are needed with much more challenging environments. TASIOM should also be tested in nonstationary environments too. In the future, different updating equations for TASIOM may also be tested.

REFERENCES

Fig. 5. TAISOM weights after 50000 iterations with the Gaussian distribution with mean $m = 0$ and variance $v = 1$.

Fig. 6. The stabilized TAISOM weights after 300000 iterations with the Gaussian distribution with mean $m = 0$ and variance $v = 1$.

Fig. 7. TAISOM weights in a two-dimensional lattice after 1000 iterations with the uniform distribution in $[5,0] \times [0,5]$.

Fig. 8. TAISOM weights in a two-dimensional lattice after 5000 iterations with the uniform distribution in $[0,5] \times [0,5]$.

Fig. 9. The stabilized TAISOM weights in a two-dimensional lattice after 200000 iterations with the uniform distribution in $[0,5] \times [0,5]$.

Fig. 10. The TAISOM weights after 1000 iterations with the Gaussian distribution with mean $m = 0$ and variance $v = 1$. 
Fig. 11. The TAISOM weights in a two-dimensional lattice after 5000 iterations with the Gaussian distribution with mean $m = 0$ and variance $\sigma = 1$.

Fig. 12. The stabilized TAISOM weights in a two-dimensional lattice after 200000 iterations with the Gaussian distribution with mean $m = 0$ and variance $\sigma = 1$.

Fig. 13. TAISOM weights in a one-dimensional lattice after 100 iterations with the uniform distribution in $[0, 5] \times [0, 5]$ with a totally random initialization of weights.

Fig. 14. TAISOM weights in a one-dimensional lattice after 5000 iterations with the uniform distribution in $[0, 5] \times [0, 5]$ with a totally random initialization of weights.

Fig. 15. TAISOM weights in a one-dimensional lattice after 10000 iterations with the uniform distribution in $[0, 5] \times [0, 5]$ with a totally random initialization of weights.

Fig. 16. The stabilized TAISOM weights in a one-dimensional lattice after 500000 iterations with the uniform distribution in $[0, 5] \times [0, 5]$ with a totally random initialization of weights.
Fig. 17. A two-dimensional lattice of 100 neurons with the same initialization, constant parameters, and input distribution of the experiment of Figs. 10-12, except that $N_c = 3\sqrt{2}$. Here, the stabilized weights after 1400000 iterations faithfully represent the Gaussian distribution of the input environment producing the quantization error 0.1875 for 10000 test input samples while the network of Fig. 12 produces the quantization error 0.1989. The difference is not substantial but reflects the network here is more suitable for density matching.

Fig. 18. The TAISOM weights of the experiment of Fig. 17 along with the topological ordering of the neurons. It is seen that the network is full of crossing itself so it doesn’t keep the topological ordering well enough. Therefore, the topological ordering is weakened in order to obtain better density matching.