A New Approach to Artificial Immune Systems and its Application in Constructing On-line Learning Neuro-Fuzzy Systems

Mu-Chun Su[*,1], Po-Chun Wang[2] and Yuan-Shao Yang[1]

[1]*Department of Computer Science and Information Engineering, National Central University, Taiwan*

[2]*Cathay General Hospital, No. 280, Section 4, Ren-Ai Road, Da-An District, Taipei, Taiwan*

Abstract: In this paper, we present an on-line learning neuro-fuzzy system which was inspired by parts of the mechanisms in immune systems. It illustrates how an on-line learning neuro-fuzzy system can capture the basic elements of the immune system and exhibit some of its appealing properties. During the learning procedure, a neuro-fuzzy system can be incrementally constructed. We illustrate the potential of the on-line learning neuro-fuzzy system on several benchmark classification problems and function approximation problems.

Keywords: Artificial immune systems, on-line learning, neural networks, fuzzy systems.

I. INTRODUCTION

The natural immune system is an adaptive learning system that is highly distributive in nature. It effectively employs several mechanisms for defending the body from foreign pathogens. One way in which the immune system does this is by using antibodies which are proteins produced by B cells. When a pathogen enters the body, an immune response is elicited to cause the antibody to bind the pathogen so that the pathogen can be mobilized for elimination. The system is capable of "remembering" each infection so that when the same pathogen is encountered again it can be dealt with more efficiently and effectively. Inspired by the information-processing properties of natural immune systems, artificial immune systems (AISs), which appeal to offer powerful and robust information processing capabilities for solving complex problems, are rapidly emerging [1-6]. Various AISs have been widely applied in solving many problems such as computer security, virus detection, data mining, and robotics [7-9].

In some applications, systems should be able to learn new classes and refine existing classes quickly and without destroying old class information. This property is referred to as *on-line* learning and it is a very appealing property for an efficient pattern recognition system. For systems using off-line learning, once, the systems have been constructed or trained; the only way to accommodate new training data is to require a complete retraining of the systems with both the old and the new information. As such, off-line learning property limits the applicability of these systems. While most of the popular neural networks utilize off-line adaptation, fuzzy ARTMAP networks [10] and fuzzy min-max neural networks [11] are two well-known classes of neural networks with on-line learning capability. In this paper, we introduce an on-line learning neuro-fuzzy system which borrows much of its operation from theories of the natural immune system. It is not our main concern to precisely model the human immune system, but to show that some basic immune principles are useful for on-line learning and problem solving.

The remaining of the paper is as follows. The natural immune system will be briefly reviewed in Section II. Section III will introduce the proposed on-line learning neuro-fuzzy system. The simulation results from system testing are presented in Section IV. Finally, Section V concludes the paper.

II. BRIEF REVIEW OF THE IMMUNE SYSTEM

The immune system is a complex system consisting of various cells, molecules, and organs which are capable of performing several tasks such as pattern recognition, learning, generation of diversity, generalization, and optimization [8, 9]. The main function of a natural immune system is to protect the body from pathogens. When an infectious foreign element invades the body, it is detected and mobilized for elimination. This infectious foreign element will then be remembered by the immune system so that a second exposure to the same foreign element will result in a more rapid and effective response.

The immune system has evolved two mechanisms: (1) **innate** (nonspecific) immunity and (2) **adaptive** (specific or acquired) immunity. Innate immunity can be regarded as natural resistance of the host to foreign antigens because the body is born with the ability to recognize certain microbes and immediately destroy them. It is the first line of defenses against the foreign antigens and it uses non-specific immune response while attaching the foreign antigens. In contrast to the innate immunity the adaptive immunity uses somatically generated antigen receptors which are clonally distributed on two types of lymphocytes: B cells and T cells. These antigen receptors are generated by random processes. The antibody molecules play an important role in the adaptive immunity.

*Address correspondence to this author at the Department of Computer Science and Information Engineering, National Central University, No. 300, Jung-da Road, Chung-li, Tao-yuan, Taiwan 320; Tel: (886-3) 422-7151, Ext. 34500; Fax: (886-3) 422-2681; E-mail: muchun@csie.ncu.edu.tw

1874-0618/08

When a pathogen invades the body, antigen presenting cells, which roam the body, process the pathogen so that its relevant features called antigens, are displayed on the surfaces of the antigen presenting cells. When a B cell encounters antigens, an immune response is elicited to produce antibodies which are molecules attached primarily to the surfaces of B cells. By binding to the antigens, antibody can neutralize them or precipitate their destruction by complement enzymes. If the antibody matches the antigen sufficiently well, its B cells become stimulated and can produce mutated clones to adapt to the antigen.

In addition to mature into plasma cells, B cells can differentiate into long-lived B memory cells. These memory cells recognize antigens similar to those that originally elicited the immune response that resulted in the generation of the memory cells, so that the body's response to a later invasion of the same antigens or similar antigens is much more rapid and powerful (i.e.. produce more abundant proportion of antibodies) than to a new antigen which has never invaded the system. Diversity in the immune system is maintained because of the least stimulated B cells which die daily and are replaced by an equal number of completely new B cells generated by the body marrow.

III. THE IMMUNITY-BASED ON-LINE LEARNING NEURO-FUZZY SYSTEMS

The immune system is a highly complicated system. Many properties of immune systems attract a great amount of attentions from compute scientists and engineers. A rich source of mechanisms in immune systems may serve as metaphors for designing computational systems. Different artificial immunes systems are inspired by different subsets of the available metaphors [12-18].

In this paper, we present an on-line learning neuro-fuzzy system which was inspired by part of the mechanisms in immune systems. Again, one thing should be emphasized is that the focus of the paper is not to precisely model the human immune system, but to show that some basic immune principles are useful for on-line learning. We name the proposed neuro-fuzzy system as the artificial immune system based neuro-fuzzy system (AISNFS). In fact, part of the properties of the early version of AISNFS has been presented in [19-20]. In this paper, we present a complete version of AISNFS and apply it in medical diagnosis and function approximation problems. We use Table **1** to illustrate the similarities of the proposed neuro-fuzzy system and the immune system.

Basically, the architecture of the proposed neuro-fuzzy system resembles the architecture of the RBF network or its variants [21-24]. The major differences between the AISNFS and these well-known networks are as follows. Generally, the RBF and the FBF networks are trained by the **supervised learning** such as orthogonal least squares, backpropagation algorithm, etc. However, our AISNFS is trained in an **on-line learning** environment. In addition, we introduce the strength parameter, s_j, to represent the strength of the *j*th rule. Owing to the appearance of the strength parameter, the AISNFS is different to the RBF and the FBF networks in the way of the computation of the network's outputs. The idea of including the strength parameter into a neuro-fuzzy system has been introduced in one of our previous work [25]. The major differences between the ACSNFIS proposed in [25] and the AISNFS are (1) the ACSNFIS trained in a **reinforcement-learning** environment instead of **on-line learning** environment and (2) the parameter updating rules are totally different.

Table 1. The Similarities of the Proposed AISNFS and the Immune System [19]

AISNFS	Immune System
Rule	B cell
Input data, $\underline{x}$	Antigen
Condition part, $\underline{w}_j$ and $\underline{\sigma}_j$	Antibody
Action part c_j	Neutralization or enzymes
Strength parameter s_j	Concentration of antibodies
Update $\underline{w}_j$, c_j, $\underline{\sigma}_j$	Mutated clones
Update s_j	Concentration updating
Delete rule with small s_j	B cells with least stimulus die daily

Each neuron in an AISNFS corresponds to a fuzzy rule which can be represented in the form as:

R_j: **IF** ($\underline{x}$ is HE_j)

THEN *the system output is* c_j *of the jth rule with the strength* s_j (1)

where $\underline{x}$ is an *n*-dimensional input vector. The fuzzy set, HE_j, defines an *n*-dimensional hyper-ellipsoid,

$$\sum_{i=1}^{n}\left(\frac{x_i - w_{ji}}{\sigma_{ji}}\right)^2 = 1,$$

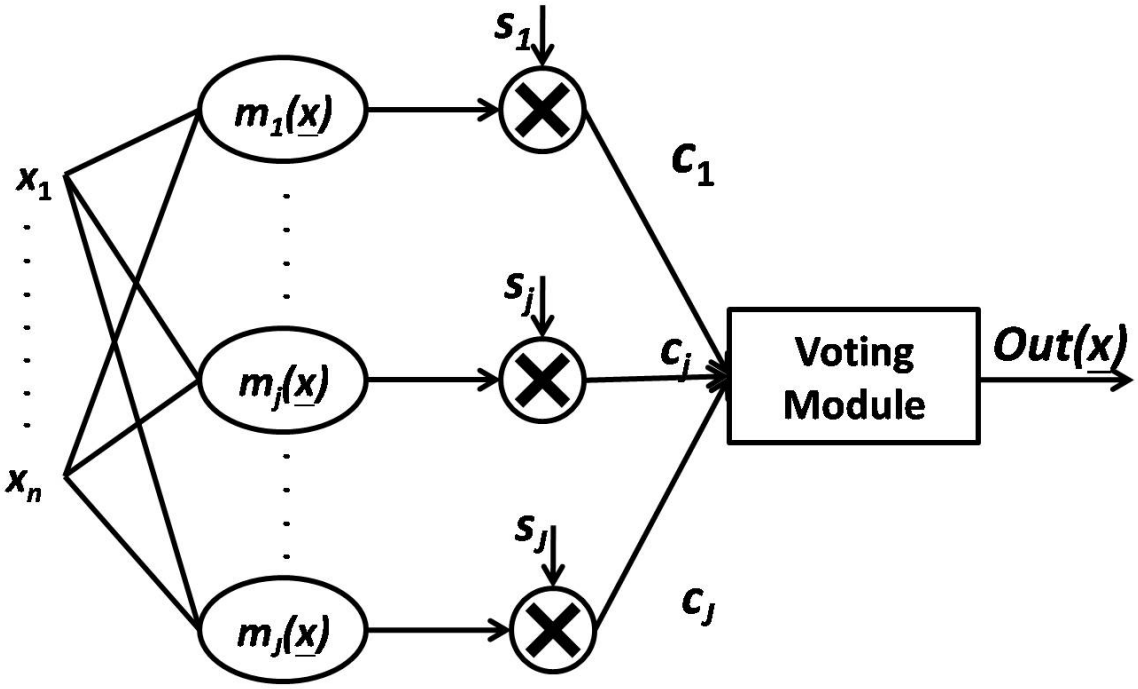
which has the membership function,

$$m_j(\underline{x}) = \exp\left\{-\sum_{i=1}^{n}\left(\frac{x_i - w_{ji}}{\sigma_{ji}}\right)^2\right\},$$

to measure the degree of an input vector, $\underline{x}$, belonging to the fuzzy set HE_j. The parameter c_j is the consequent part of the *j*th rule. Besides, each rule is assigned a strength parameter s_j.

The training problem can be divided into two classes of problems-pattern recognition problems and function approximation problems. While the desired outputs for a pattern recognition problem are discrete, the desired outputs for a function approximation problem may be real-valued. Fig. (**1**) shows the architectures of the AISNFS for solving these

(**a**)



(**b**)

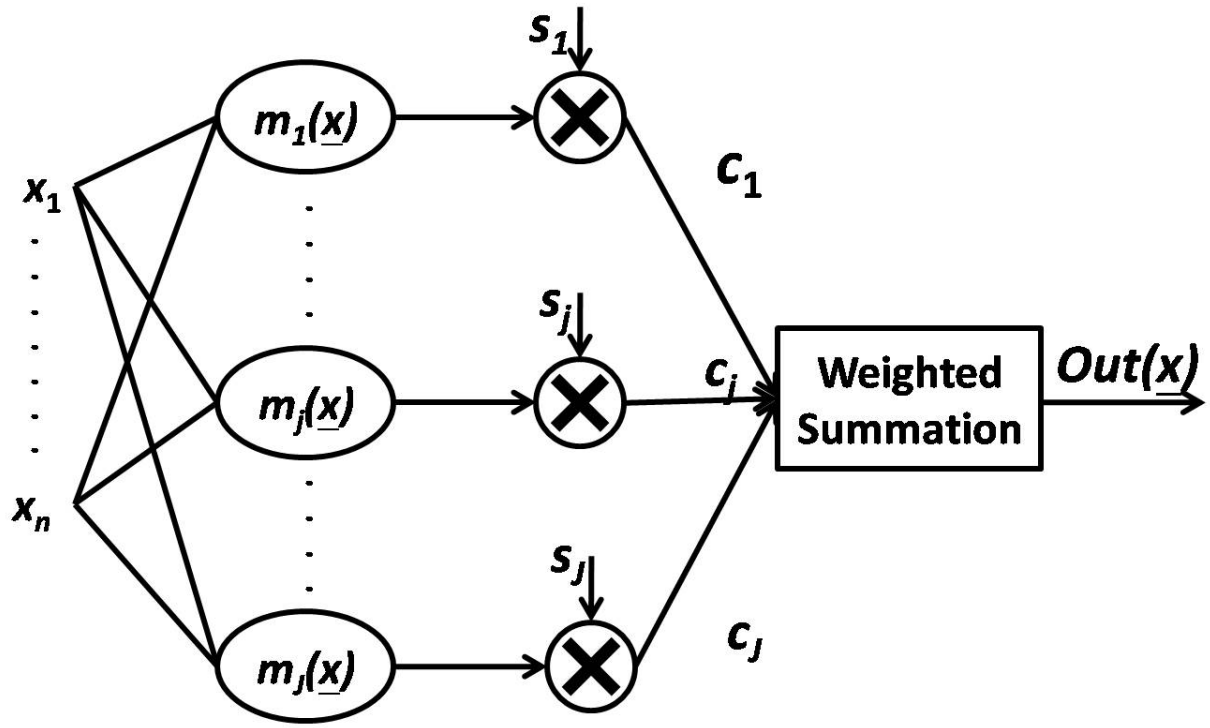


Fig. (1). The architecture of the AISNFS. (**a**) The architecture for pattern recognition problems. (**b**) The architecture for function approximation problems.

two kinds of problems. The differences between two architectures are the module of computing the final system's outputs and some parameters updating rules.

A. Learning Algorithm for Pattern Recognition Problems

The training algorithm starts with no committed neuron in the system and then incrementally adds neurons into the

system to produce correct predictions. The training algorithm for the AISNFS is summarized as follows:

Step 1: Set $J = 0$ and initialize the parameters, η_w, η_σ and η_s, where J is the number of rules in AISNFS, η_w, η_σ and η_s indicate the learning rates.

Step 2: Present an input pattern $\underline{x}$ into the system. Suppose the class label of the input pattern is L.

Step 3: If there is no neuron in the system then go to Step 8 to generate a new neuron to respond to the first data pattern $\underline{x}$. Otherwise, compute the outputs of the neurons in the system using the following equation:

$$m_j(\underline{x}) = \exp\left\{-\sum_{i=1}^{n}\left(\frac{x_i - w_{ji}}{\sigma_{ji}}\right)^2\right\} \quad (2)$$

Step 4: Find the winner j^* by selecting the neuron with the largest similarity which is computed by multiplying the neuron's output, $m_j(\underline{x})$, with its corresponding strength, s_j:

$$s_{j^*}m_{j^*}(\underline{x}) = Arg \max_{j=1,\cdots,J}\left(s_j m_j(\underline{x})\right) \quad (3)$$

Here we assume the present number of neurons in the system is J.

Step 5: Check whether the similarity between the winner and the input pattern $\underline{x}$ is larger than the pre-specified threshold θ, where θ is a positive real-valued constant pre-specified by the user:

$$s_{j^*}m_{j^*}(\underline{x}) \geq \theta \quad (4)$$

If this condition is met, then find the voting set, V_s, consisting of neurons of which similarities are also larger than the threshold θ. Otherwise, go to Step 8 to generate a new neuron.

Step 6: Compute the system output. The system output is computed by the so-called ***voting method***. That is, each neuron in the voting set will vote for the labeling of the input pattern. Each vote is weighted by its corresponding neuron's similarity. The system output is class k if class k has the largest amount of weighted votes.

Step 7: Check whether the system output is consistent with the label of the input data. If consistent, update the winner's weight vector and its strength in the following way:

$$\underline{w}_{j^*}(t+1) = \underline{w}_{j^*}(t) + \eta_w \times \frac{1}{f_{j^*}(t)} \times (\underline{x} - \underline{w}_{j^*}(t)) \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \quad (5)$$

$$\sigma_{j^*i}(t+1) = \sigma_{j^*i}(t) + \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \frac{\partial Out\ (\underline{x})}{\partial \sigma_{j^*i}} \quad (6)$$

$$\sigma_{j^*i}(t) + \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \left(\frac{2(x_i - w_{j^*i}(t))^2}{\sigma_{j^*i}(t)^3} \times s_{j^*}(t) \times m_{j^*}(\underline{x})\right)$$

$$s_{j^*}(t+1) = s_{j^*}(t) + \eta_s \times \frac{1}{f_{j^*}(t)} \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \quad (7)$$

where $f_{j^*}(t)$ represents the winning frequencies for the neuron j^* at time t.

Otherwise,

$$\underline{w}_{j^*}(t+1) = \underline{w}_{j^*}(t) - \eta_w \times \frac{1}{f_{j^*}(t)} \times (\underline{x} - \underline{w}_{j^*}(t)) \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \quad (8)$$

$$\sigma_{j^*i}(t+1) = \sigma_{j^*i}(t) - \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \frac{\partial Out\ (\underline{x})}{\partial \sigma_{j^*i}} \quad (9)$$

$$\sigma_{j^*i}(t) - \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \left(\frac{2(x_i - w_{j^*i}(t))^2}{\sigma_{j^*i}(t)^3} \times s_{j^*}(t) \times m_{j^*}(\underline{x})\right)$$

$$s_{j^*}(t+1) = s_{j^*}(t) - \eta_s \times \frac{1}{f_{j^*}(t)} \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \quad (10)$$

Then go to Step 2.

Step 8: Generate a new neuron whose weight vector and strength are initialized as follows:

$$\underline{w}_{J+1} = \underline{x} \quad (11)$$

$$\underline{\sigma}_{J+1} = \underline{\sigma}_0 \quad (12)$$

$$c_{J+1} = L \quad (13)$$

$$s_{J+1} = s_0 \quad (14)$$

$$J = J + 1 \quad (15)$$

where $\underline{\sigma}_0$ is a vector consisting of n pre-specified real-valued constants and s_0 is the initial value of strength for each rule. Then go to Step 2.

Note that at the end of epoch, we will delete the neurons of which strengths are smaller than the initial value s_0. To improve the training performance, we may continue the training procedure until a pre-specified number of epochs is reached. After the network has been trained, we proceed to the testing mode. In the testing mode, the trained network can be used to predict the output when an input pattern, $\underline{x}$, is presented to the network. First, we find the voting set. If the set is not an empty set then the output is computed by the

voting method. Otherwise, we use the winner's label, c_{j^*}, as the system's output.

B. Learning Algorithm for Function Approximation Problems

Basically, the learning algorithms for function approximation and pattern recognition are very similar except the computation of system's outputs and some parameter updating rules. However, for the clarity purpose we still start from the first step. The algorithm for function approximation is given as follows:

Step 1: Set $J = 0$ and initialize the parameters, η_w, η_σ, η_s and η_c, where J is the number of rules in AISNFS, η_w, η_σ, η_s and η_c indicate the learning rates.

Step 2: Present an input pattern $\underline{x}$ into the system. Suppose the desired output of the input pattern is d.

Step 3: If there is no neuron in the system then go to Step 8 to generate a new neuron to respond to the first data pattern $\underline{x}$. Otherwise, compute the outputs of the neurons in the system using the following equation:

$$m_j(\underline{x}) = \exp\left\{-\sum_{i=1}^{n}\left(\frac{x_i - w_{ji}}{\sigma_{ji}}\right)^2\right\} \tag{16}$$

Step 4: Find the winner j^* by selecting the neuron with the largest similarity which is computed by multiplying the neuron's output, $m_j(\underline{x})$, with its corresponding strength, s_j:

$$s_{j^*}m_{j^*}(\underline{x}) = Arg\max_{j=1,\cdots,J}\left(s_j m_j(\underline{x})\right) \tag{17}$$

Here we assume the present number of neurons in the system is J.

Step 5: Check whether the similarity between the winner and the input pattern $\underline{x}$ is larger than the pre-specified threshold θ:

$$s_{j^*}m_{j^*}(\underline{x}) \geq \theta \tag{18}$$

If this condition is met, then find the voting set, V_s, consisting of neurons of which similarities are also larger than the threshold θ. Otherwise, go to Step 8 to generate a new neuron.

Step 6: Compute the system output. The system output is computed by the weighted summation method as follows:

$$Out(\underline{x}) = \frac{\sum_{j\in V_s} c_j m_j(\underline{x})}{\sum_{j\in V_s} m_j(\underline{x})} \tag{19}$$

That is, all neurons in the voting set will cooperate to compute the network's outputs. The weighting factor for each neuron is provided by the value of membership function, $m_j(\underline{x})$.

Step 7: Compute the difference, e, between the system output and the desired output, $e = |Out(\underline{x}) - d|$. Then check whether the difference is smaller than the pre-specified threshold, θ_{error}, where θ_{error} is a positive real-valued constant pre-specified by the user. If yes, update the winner's strength in the following way:

$$s_{j^*}(t+1) = s_{j^*}(t) + \eta_s \times \frac{1}{f_{j^*}(t)} \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \tag{20}$$

Otherwise, we update the winner's weight vector, strength and its output in the following way:

Case 1: *IF* $d - Out(\underline{x}) > \theta_{error}$

$$\underline{w}_{j^*}(t+1) = \underline{w}_{j^*}(t) + \eta_w \times \frac{1}{f_{j^*}(t)} \times (\underline{x} - \underline{w}_{j^*}(t)) \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \tag{21}$$

$$\sigma_{j^*i}(t+1) = \sigma_{j^*i}(t) + \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \frac{\partial Out\ (\underline{x})}{\partial \sigma_{j^*i}} \sigma_{j^*i}(t) + \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \left(\frac{2(x_i - w_{j^*i}(t))^2}{\sigma_{j^*i}(t)^3} \times s_{j^*}(t) \times m_{j^*}(\underline{x})\right) \tag{22}$$

$$s_{j^*}(t+1) = s_{j^*}(t) - \eta_s \times \frac{1}{f_{j^*}(t)} \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \tag{23}$$

$$c_{j^*}(t+1) = c_{j^*}(t) + \eta_c \times \frac{1}{f_{j^*}(t)} \times (d - Out(\underline{x})) \times \frac{s_{j^*}m_{j^*}(\underline{x})}{\sum_{j\in V_s} s_j m_j(\underline{x})} \tag{24}$$

Case 2: *IF* $d - Out(\underline{x}) < -\theta_{error}$

$$\underline{w}_{j^*}(t+1) = \underline{w}_{j^*}(t) - \eta_w \times \frac{1}{f_{j^*}(t)} \times (\underline{x} - \underline{w}_{j^*}(t)) \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \tag{25}$$

$$\sigma_{j^*i}(t+1) = \sigma_{j^*i}(t) - \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \frac{\partial Out\ (\underline{x})}{\partial \sigma_{j^*i}} \sigma_{j^*i}(t) - \eta_\sigma \times \frac{1}{f_{j^*}(t)} \times \left(\frac{2(x_i - w_{j^*i}(t))^2}{\sigma_{j^*i}(t)^3} \times s_{j^*}(t) \times m_{j^*}(\underline{x})\right) \tag{26}$$

$$s_{j^*}(t+1) = s_{j^*}(t) - \eta_s \times \frac{1}{f_{j^*}(t)} \times s_{j^*}(t) \times m_{j^*}(\underline{x}) \quad (27)$$

$$c_{j^*}(t+1) = c_{j^*}(t) - \eta_c \times \frac{1}{f_{j^*}(t)} \times (Out(\underline{x}) - d) \times \frac{s_{j^*} m_{j^*}(\underline{x})}{\sum_{j \in V_s} s_j m_j(\underline{x})} \quad (28)$$

Then go to Step 2.

Step 8: Generate a new neuron whose weight vector and strength are initialized as follows:

$$\underline{w}_{J+1} = \underline{x} \quad (29)$$

$$\underline{\sigma}_{J+1} = \underline{\sigma}_0 \quad (30)$$

$$c_{J+1} = d \quad (31)$$

$$s_{J+1} = s_0 \quad (32)$$

$$J = J + 1 \quad (33)$$

Then go to Step 2.

Note that at the end of epoch, we will delete the neurons of which strengths are smaller than the initial value s_0. To improve the training performance, we may continue the training procedure until a pre-specified number of epochs is reached. After the network has been trained, we proceed to the testing mode. In the testing mode, the trained network can be used to predict the output when an input pattern, $\underline{x}$, is presented to the network. First, we find the voting set. If the set is not an empty set then the output is computed by the weighted summation method. Otherwise, we use the winner's output, c_{j^*}, as the system output.

IV. SIMULATIONS

Two pattern recognition problems and two function approximation problems were used to test the performance of the proposed on-line learning AISNFS. For all the simulations, the parameters in AISNFS, η_w, η_σ, η_s, η_c and s_0, were set to be 0.01, 0.001, 0.001, 0.01 and 0.3, respectively. The parameter θ_{error} is set to be 0.06 for example 3 and 0.1 for example 4.

Example 1: The Iris Data Set

The iris data set consisting of 150 4-dimensional data points belonging to 3 different classes. There are 50 data points in each class. The data set was randomly divided into a training data set and a testing data set. Each data set consists of 75 data points. Table **2** gives the simulation results. For the comparison purpose, we tabulate the performance achieved by other methods [26-30] in Table **2**. In Table **2**, the highlighted (**bold and shaded**) entries correspond to maximum correct rate of the compared methods. Basically, the recognition performance of these methods are close to each other, however, our on-line learning AISNFS has the on-line learning capability.

To demonstrate the proposed AISNFS owns the on-line learning capability, we first used the 100 data points belonging to the first two classes to train the AISNFS. After training, 11 rules were generated and the recognition performance was 100% correct. Table **3** tabulates the results. Then we fed the remaining 50 data points belonging to the third class into the system. Note that the original 100 data points were no longer presented to the system. Finally, another 16 rules were generated. The recognition performance became 97.3% correct as shown in Table **4**. While original 50 data points belonging to the second class could be 100% recognized after the first training procedure, one data point became miss-classified owing to the introduction of the newly added 50 data points belonging to the third class after the second training procedure. In addition, three data points belonging to class 3 were miss-classified to be class two. Although the recognition performance decreases a little bit the simulation results demonstrates the system has the on-line learning capability.

Example 2: Medical Data Sets

In this experiment, three benchmark data sets are taken from the repository of the University of California at Irvine, Repository of Machine Learning Databases (UCI) [31]. We briefly describe the three data sets:

1. **Contraceptive Method Choice:** The problem is to predict the contraceptive method choice of a woman based on her demographic and socio-economic characteristics. There are three classes, two numerical attributes, seven categorical attributes, and 1473 observations.

Table 2. A Comparison of the Simulation Results of the Proposed AISNFS with the Other Existing Methods

Methods	On-line Learning Capability	Training Data/Testing Data	Performance for the Training Set	Performance for the Testing Set	Number of Neurons (Rules)
AISNFS	yes	75/75	**100%**	**98.7%**	14
Abe and Lan [26]	no	75/75	**100%**	92% to 97.3%	5 to 17
NEFCLASS [27]	no	75/75	96%	97.3%	7
AISNFS	yes	150/0	97.3%		7
SuPFuNIS [28]	no	150/0	**100%**		5
FuNe-I [29]	no	150/0	96.0%		7
EFuNN [30]	no	150/0	95.3%		17

2. **BUPA Liver-Disorders:** The problem is to predict whether a male patient has a liver disorder based on blood tests and alcohol consumption. There are two classes, six numerical attributes, and 345 observations.

3. **Wisconsin Breast Cancer:** The problem is to predict a tissue sample taken from a patient's breast is malignant or benign. There are two classes, nine numerical attributes, and 699 observations. Sixteen instances are removed because they contain a missing attribute value.

Table 3. The Recognition Result for the First Training Procedure in Example 1:100 Data Points Belonging to the First Two Classes

Real \ Classified	1	2
1	50	0
2	0	50

Table 4. The Recognition Result for the Second Training Procedure in Example 1: The Introduction of the 50 Data Points Belonging to the Third Class

Real \ Classified	1	2	3
1	50	0	0
2	0	49	1
3	0	3	47

Lim *et al*. [32] reported the results of applying twenty-two decision tree, nine statistical, and two neural network algorithms against many of the benchmark data sets (including these three databases) taken from the UCI databases. They used ten-fold cross-validation for estimating error rates. For the comparison purpose, we also evaluate the proposed AISNFS using ten-fold cross-validation. We encapsulate the results provided by Lim *et al*. [32] and insert AISNFS in the appropriate location in the ordered list for each classification problem in Table **5**. Obviously, AISNFS outperforms a number of well-respected classification methods on these three data sets. As for some other data sets from the UCI databases, the proposed AISNFS did not perform so well as in the three data sets; however, it still was comparable to some well-respected classification methods.

Example 3: Single-Input-Single-Output (SISO) Function Approximation

The following function, which has been used in literature [28, 33-36] to test the learning capabilities of the proposed models, is used to test the performance of the proposed AISNFS:

$$y(x) = 0.2 + 0.8(x + 0.7\sin(2\pi x)), \quad 0 \le x \le 1 \tag{34}$$

where x is the input of the network and $y(x)$ is the output of the network. For the comparison purpose, we constructed the same training data set and testing data set in the same way as in [28, 33-36]. That is, twenty-one training patterns were generated at intervals of 0.05 and 101 testing patterns were taken at intervals of 0.01. The root mean square error is used as the performance index here. Table **6** compares the test accuracy performance for different models along with the number of rules and adjustable parameters used in achieving

Table 5. Error Rate Comparison of Classifiers on the Three UCI Databases. These Results are Taken from [32] Except for the Results of the Proposed AISNFS

Contraceptive Method Choice			BUPA Liver-Disorders			Wisconsin Breast Cancer		
Rank	**Methods**	**Error Rate**	**Rank**	**Methods**	**Error Rate**	**Rank**	**Methods**	**Error Rate**
1	**AISNFS**	**0.390**	1	**AISNFS**	**0.264**	1	**AISNFS**	**0.0252**
2	POL	0.434	2	OCM	0.279	2	LVQ	0.0278
3	QV1	0.440	3	FM2	0.280	3	QL0	0.0308
4	QV0	0.443	4	POL	0.286	3	QL1	0.0308
5	IM	0.447	5	FM1	0.289	5	IB0	0.0336
6	IC1	0.449	6	C4R	0.292	6	LOG	0.0337
7	C4R	0.450	7	OCL	0.296	6	RBF	0.0337
8	IC0	0.451	8	QL0	0.306	8	LMT	0.0351
8	ST1	0.451	9	C4T	0.308	9	FM1	0.0366
10	SY0	0.457	10	LOG	0.309	10	QU0	0.0380
11	RBF	0.458	11	ST0	0.311	10	FM2	0.0380
…	…	…	…	…	…	…	…	…
33	QDA	0.541	33	CAL	0.420	33	T1	0.0760
34	NN	0.601	34	T1	0.432	34	OCL	0.0848

Table 6. Comparison of the Proposed AISNFS with Other Methods for the SISO Function. These Results are Taken from [28] Except for AISNFS Data

Methods	Number of Rules	Adjustable Parameters	Testing Accuracy
GEPREX [33]	3	9	0.090
FuGeNeSys [34]	5	15	0.856
Lin & CunninghamIII [35]	4	16	0.987
Narazaki & Relascu [36]	na	12	3.19
SuPFuNIS [28]	3	13	0.3310
SuPFuNIS [28]	5	21	0.0972
AISNFS	5	20	**0.0846**

it. With five rules the trained AISNFS is better than all other models.

Example 4: Multi-Inputs-Single-Output (MISO) Function Approximation

The plant to be identified is governed by the difference equation

$$y(t+1) = (0.8 - 0.5\exp(-y^2(t))y(t)) - (0.3 + 0.9\exp(-y^2(t))u(t)) + 0.1\sin(3.1415y(t)) \quad (35)$$

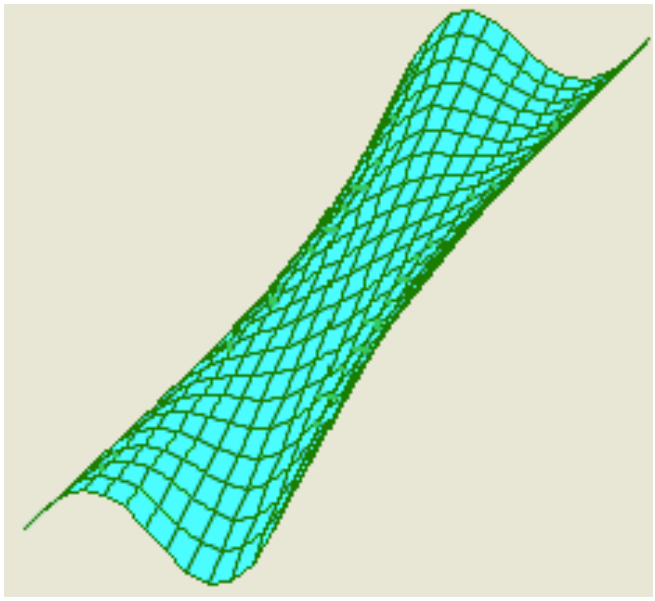
where $y(t)$ and $u(t)$ are the two inputs of the network and $y(t+1)$ is the output of the network.

First, we trained the network using inputs scattered evenly over a square region $[-2,2]\times[-2,2]$. The total number of training data is 441. The trained network was then tested on inputs corresponding to another 400 data points that are uniformly selected from the square region $[-1.9,1.9]\times[-1.9,1.9]$. The root mean square error is used as the performance index here. The simulation results are shown in Fig. (**2**). Table **7** compares the accuracy performance for different models such as a RBF network with 120 hidden nodes and a MLP with total 22 hidden nodes (12×10 nodes in two hidden layers). One may find that the trained AISNFS achieved the lowest root mean square error for the training set and the testing set.

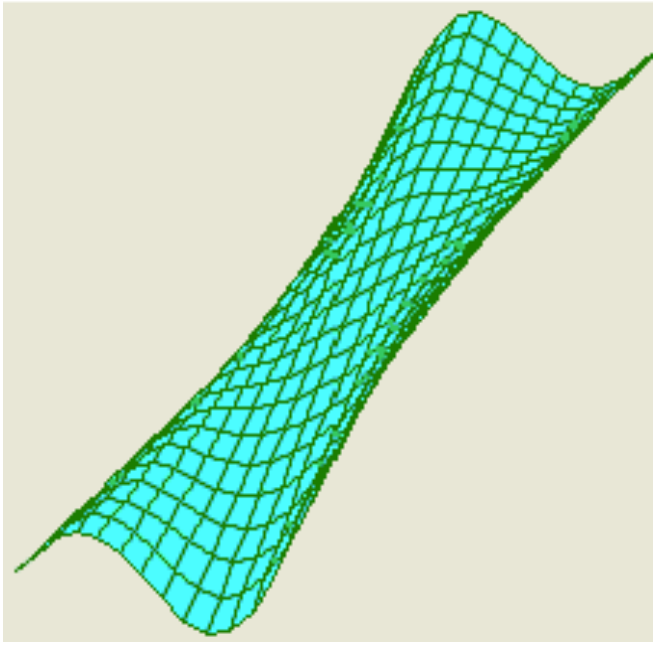
Table 7. Comparison of the Proposed AISNFS with Other Methods for the MISO Function

Methods	Architecture	Training Accuracy	Testing Accuracy
RBF	$2\times120\times1$	0.0105	0.0143
MLP	$2\times12\times10\times1$	0.0090	0.0102
AISNFS	$2\times120\times1$	**0.0067**	**0.0081**

(**a**)



(**b**)

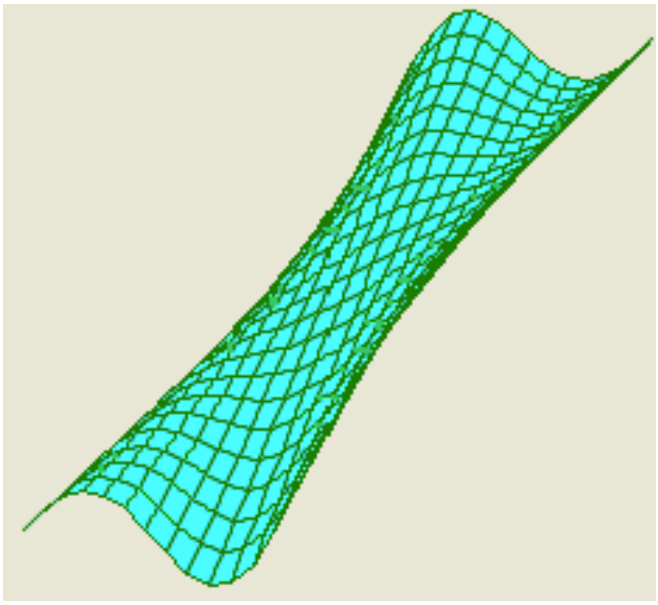


V. CONCLUSIONS

This paper introduces a new on-line learning technique based on some principles from immune systems. One thing should be emphasized is that the focus of the paper is not to

(Fig. 2) contd…..

(**c**)



(**d**)

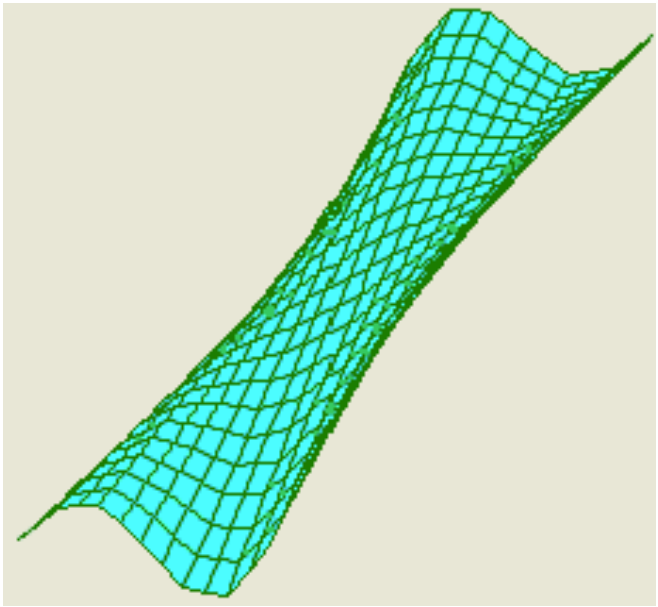


Fig. (2). The results for the MISO function. (**a**) The patterns for training. The function learned by the (**b**) RBF, (**c**) MLP, (**d**) AISNFS.

precisely model the human immune system, but to show that some basic immune principles are useful for on-line learning. Based on the developed learning algorithm, a neuro-fuzzy system called AISNFS can be incrementally constructed. We use the strength parameter, s_j, to represent the strength of the *j*th rule. Owing to the appearance of the strength parameter, the AISNFS is different to the RBF and the FBF networks in the way of the computation of the network's outputs. The simulation results show that the proposed AISNFS is comparable to, and even exceeds performance of several well-developed and widely used algorithms.

REFERENCES

[1] J. E. Hunt and D. E. Cooke, "Learning using an artificial immune system", J Network Comput Appl, vol.19, pp. 189-212, 1996.

[2] D. Dasgupta, "Artificial neural networks and artificial immune systems: similarities and differences," in IEEE International Conference on Systems, Man, and Cybernetics, 1997, pp. 873-878.

[3] D. F. McCoy and V. Devarajan, "Artificial immune systems and aerial image segmentation," in IEEE International Conference on Systems, Man, and Cybernetics, 1997, pp. 867-872.

[4] D. Dasgupta, "Immunity-based intrusion detection system: a general framework," in Proc. of the 22nd National Information Security Conference, 1999.

[5] S. A. Hofmeyr and S. Forrest, "Immunity by design: an artificial immune system," in Proc. of GECCO'99, 1999, pp. 1289-1296.

[6] S. A. Hofmeyr, "An interpretative introduction to the immune system," Design Principles for the Immune System and Other Distributed Autonomous Systems, L. A. Sagel and I. R. Cohen, eds., New York, Oxford University Press, pp. 3-26, 2001.

[7] D. Dasgupta, Artificial immune systems and their applications, Springer-Verlag, 1998.

[8] L. N. De Castro and F. J. Von Zuben, "Artificial immune systems: part I - basic theory and applications," Technical Report - RT DCA 01/99, 1999, pp. 1-95.

[9] L. N. De Castro and F. J. Von Zuben, "Artificial immune systems: part II-a survey of applications," Technical Report - RT DCA 02/00, 2000, pp. 1-65.

[10] G.. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps," IEEE Trans. on Neural Networks, vol. 3, pp. 698-713, 1992.

[11] P. Simpson, "Fuzzy min-max neural networks─ part 1: classification," IEEE Trans. on Neural Networks, vol. 3, no. 5, pp. 776-786, 1992.

[12] D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," in The 5th International Conference on Intelligent Systems, 1996, pp. 82-87.

[13] K. KrishnaKumar and J. Neidhoefer, "Immunized adaptive critics," in IEEE International Conference on Neural Networks, 1997, pp. 2283-2287.

[14] S. A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," Evolutionary Computation, vol. 8, no. 4, pp. 443-473, 2000.

[15] H. Meshref and H. VanLandingham, "Artificial immune systems: application to autonomous agents," in IEEE International Conference on Systems, Man, and Cybernetics, 2000, pp. 61-66.

[16] J. Timmis, M. Neal, and J. Hunt, "An artificial immune system for data analysis," Biosystems, vol. 55, pp. 143-150, 2000.

[17] J. Caeter, "The immune system as a model for pattern recognition and classification," J Am Med Informs Assoc, vol. 7, no. 1, pp. 28-41, 2000.

[18] A. Watkins and L. Boggess, "A new classifier based on resource limited artificial immune systems," in Proceedings of the 2002 Congress on Evolutionary Computation, 2002, vol. 2, pp.1546-1551.

[19] M. C. Su, Y. S. Yang, C. H. Chou, E. Lai, and M. N. Hsiao, "An on-line learning neuro-fuzzy system based on artificial immune systems," in IEEE International Joint Conference on Neural Networks IJCNN, 2004, pp. 1073-1078.

[20] M. C. Su, Y. S. Yang, J. Lee, and G. D. Chen, "A neuro-fuzzy approach for compensating color backlight images," Neural Processing Letters, vol. 23, no. 3, pp. 273-283, 2006.

[21] J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," Neural Comput, vol. 1, pp. 181-194, 1989.

[22] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," IEEE Trans on Neural Netw, vol. 3, no. 5, pp. 807-814, 1992.

[23] H. M. Kim and J. M. Mendel, "Fuzzy basis functions: comparisons with other basis functions," IEEE Trans on Fuzzy Systems, vol. 3, no. 2, pp. 158-168, 1995.

[24] C. W. Le and Y. C. Shin, "Construction of fuzzy basis function networks using adaptive least squares method," in Proc. of IFSA World Congress and 20th NAFIPS Int Conf, 2001, pp. 2630-2635.

[25] M. C. Su, C. H. Chou, and E. Lai, "A self-generating neuro-fuzzy system through reinforcements," in the 2nd WSEAS Int. Conf. on Scientific Computation and Soft Computing, 2002, pp. 332-337.

[26] S. Abe and M. S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," IEEE Trans on Fuzzy Systems, vol. 3, no.1, pp. 18-28, 1995.

[27] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," Fuzzy Sets Syst, vol. 89, no. 3, pp. 277-288, 1997.

[28] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," IEEE Trans on Neural Networks, vol. 13, no. 3, pp. 578-599, 2002.

[29] S. Halgamuge and M. Glesner, "Neural networks in designing fuzzy systems for real world applications," Fuzzy Sets and Systems, vol. 65, pp. 1-12, 1994.

[30] N. Kasabov and B. Woodford, "Rule insertion and rule extraction from evolving fuzz neural networks: Algorithm and applications for building adaptive, intelligent expert systems," in Proc. of IEEE Int Conf Fuzzy Syst, 1999, vol. 3, pp. 1406-1411.

[31] C. L. Blake and C. J. Merz, UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[32] T. S. Lim, W. Y. Loh, and Y. S. Shih, "Comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," Machine Learning, vol. 40, pp. 203-229, 2000.

[33] M. Russo, "Genetic fuzzy learning," IEEE Trans. on Evolutionary Computation, vol. 4, pp. 259-273, 2000.

[34] M. Russo, "FuGeNeSys—a fuzzy genetic neural system for fuzzy modeling," IEEE Trans on Fuzzy Systems, vol.6, pp. 373-388, 1998.

[35] Y. Lin and G. A. Cunningham, "A new approach to fuzzy-neural system modeling, " IEEE Trans on Fuzzy Systems, vol. 3, pp. 190-197, 1995.

[36] H. Narazaki and A. Ralescu, "An improved synthesis method for multilayered neural networks using qualitative knowledge," IEEE Trans on Fuzzy Systems, vol. 1, pp. 125-137, 1993.

Received: February 26, 2008 Revised: March 10, 2008 Accepted: March 11, 2008