A YING-YANG APPROACH TO THE OPTIMIZATION OF FUZZY CEREBELLAR MODEL ARTICULATION

NANYANG TECHNOLOGICAL UNIVERSITY

A YING-YANG APPROACH TO THE OPTIMIZATION OF FUZZY CEREBELLAR MODEL ARTICULATION CONTROLLER

NGUYEN MINH NHUT

2008

NGUYEN MINH NHUT

SCHOOL OF COMPUTER ENGINEERING

2008

A Ying-Yang Approach To The Optimization of Fuzzy Cerebellar Model Articulation Controller

Nguyen Minh Nhut

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2008

NANYANG TECHNOLOGICAL UNIVERSITY

A YING-YANG APPROACH TO THE OPTIMIZATION OF FUZZY CEREBELLAR MODEL ARTICULATION CONTROLLER

The thesis of Degree of Doctor of Philosophy Program of Nanyang Technological University

Nguyen Minh Nhut

Division of Computer Science
School of Computer Engineering

2007

Statement of Originality

I hereby certify that the content of the result of work done by myself and has not been submitted for a higher degree to any other University or Institution.

…….. ……..

Date Signature

ACKNOWLEDGMENTS

First of all, I am greatly indebted to my supervisor Dr. Shi Daming who introduced the problem of FCMAC structure as well as Bayesian Ying Yang learning to me, and has been a helping hand throughout. His knowledge, understanding, advice, and constant support during this research have been invaluable.

Special gratitude must be given to Professor Quek Hook Chai from school of Computer Engineering, Nanyang Technological University and Professor Lei Xu from Chinese University of Hong Kong. They have helped me a lot in improving my understanding on fuzzy neural network and Bayesian Ying Yang learning.

I would like to express my thankfulness to my parents, and my brothers in Vietnam for supporting and encouraging me throughout my studies. The sacrifice they have rendered to me is indescribable.

Finally, I thank all my Vietnamese friends, my lab mates for supporting, sharing their experience and happiness of graduate student's life.

Singapore, June 18th 2007

Nguyen Minh Nhut

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

As an associative memory neural network model, the Cerebellar Model Articulation Controller (CMAC) has attractive properties of fast learning speed and simple computation, but its rigid structure makes it difficult to approximate certain functions. In order to overcome this shortcoming, our research aims at the fuzzification phase and the rule weighting process to improve FCMAC by Bayesian Ying-Yang (BYY) learning, cooperative coevolution computation, and online learning. The contributions of this research can be claimed as follows.

Firstly, Bayesian Ying-Yang learning is embedded in fuzzy CMAC, named FCMAC-BYY, to find the optimal fuzzy sets in the fuzzification phase. Bayesian Ying-Yang learning is motivated from the famous Chinese ancient Ying-Yang philosophy: everything in the universe can be viewed as a product of a constant conflict between opposites – Ying and Yang, a perfect status is reached if Ying and Yang achieves harmony. Apart from the optimal fuzzy sets systematically obtained by BYY, the proposed FCMAC-BYY also enjoys a consistent rule base, intuitive fuzzy logic reasoning and clear semantic meanings.

Secondly, cooperative coevolution computation is integrated into FCMAC-BYY to search for the global solutions. As a matter of fact, FCMAC-BYY suffers from two problems: the fuzzification phase is separated from the learning phase, and the weights of rules are trained by gradient-based methods. Both of these two problems may lead to a local minimum during the learning process. Cooperative coevolutionary learning is hence

introduced to FCMAC-BYY to utilize global search and optimization techniques through the operation of recombination-mutation.

Thirdly, the parameter learning phase in FCMAC-BYY can only be carried out when we already get all the training data, so it is not suitable for online applications where the data are not always available during the training phase. Although the fuzzy clusters of the BYY-based fuzzification are systematically optimized in training, they need to be controlled and adjusted to reduce the output errors in some real time applications. This motivates us to develop an online FCMAC-OBYY, which offers the following advantages. First, the antecedent of the fuzzy rules are dynamically constructed and optimized by the online BYY algorithm without any prior knowledge. Second, the credit assignment is then employed to greatly speed up the learning process.

Finally, the research is realized in a web-based stock prediction system using online FCMAC-OBYY. The system can automatically acquire data from stock exchange companies via internet and perform stock price prediction based on analyses and extraction of patterns from historical data.

Chapter 1

Introduction

Albus proposed the Cerebellar Model Articulation Controller (CMAC) neural network model in 1975. Owing to its fast learning property, simple computation, and ease of implementation by hardware, CMAC has been applied in many real-world applications. However, the original CMAC suffers from enormous memory size requirement and cannot provide a human-like reasoning capability due to its black box training. To address these problems, fuzzy logic is introduced into CMAC to obtain Fuzzy CMAC which can alleviate the memory problem as well as the capability of processing information based on fuzzy inference rules. This research is mainly focused on the fuzzification phase and the rule weighting process to improve the Fuzzy CMAC.

1.1 Problem Statement

The Cerebellar Model Articulation Controller (CMAC) [1, 2], is a type of associative memory neural network that models how a human cerebellum takes inputs, organize its memory and compute the outputs. CMAC is a table-lookup module that represents complex and non-linear functions. The associative mapping built into CMAC assures local generalization: similar inputs produce similar outputs while distant input produce nearly independent outputs.

The CMAC system has the advantages of fast learning speed, simple computation, local generalization, and therefore can be realized by high-speed hardware. The application of CMAC can be found in many areas such as robotic control, signal processing, and pattern recognition [3-5]. However, the original CMAC model of Albus has two major disadvantages: the memory requirement grows exponentially with respect to the number of input variables and difficulty in selecting the memory structure parameters [6, 7]. It is very inefficient in terms of data storage and modeling of the problem space. Moreover, as the trained CMAC is a black box, it is not possible to extract structural knowledge from the system or to incorporate prior knowledge from domain expert.

To address the above problems, some researchers have introduced fuzzy logic into CMAC to obtain a new fuzzy neural system model which is called Fuzzy CMAC, or FCMAC [8-10]. Firstly, the input clusters are represented as fuzzy set, rather than the crisp sets in the original CMAC, greatly alleviate the memory requirement. Secondly, fuzzy CMAC can provide a human-like reasoning capability which is essential to involve expert-knowledge.

Typically, the fuzzification phase in a fuzzy neural network is fulfilled by clustering the training data in each dimension independently. All the existing clustering algorithms can be divided into two groups: clustering with or without pre-specified cluster number [11-14]. However, all of these conventional methods apply “one way” clustering, that is, they consider only either the forward path of mapping the input data into the clusters, or, the

backward path of learning the clusters from the input data. Therefore, they cannot guarantee to achieve an optimal cluster architecture for the input data.

Fuzzy ART [14] introduced by Carpenter *et al.* is self-organized clustering, which groups a given set of input patterns into some categories. One of the characteristics of Fuzzy ART is the use of vigilance parameter to determine the number of clustered groups depending only on the similarity between all input patterns. Another one is the adjustment of the weight vector of clusters through the learning procedure. However, the number of generated clusters is very sensitive to the predefined vigilance threshold value. Moreover, the sequence of the input patterns also affects the clustering results.

Similar to Fuzzy ART, Discrete incremental clustering (DIC) [12] proposed by Tung and Quek has the characteristics of noise tolerance and does not require prior knowledge of the number of clusters present in the training data set. It used the same initial information to create a new cluster for every dimension, but the data distribution is different from dimension to dimension, as a result, DIC cannot obtain an optimal number of clusters.

Lee, Chen and Fu proposed a self-organizing HCMAC neural network [15] which combines a self-organizing input space module and a binary hierarchical CMAC neural network. However, the self-organizing input space module based on Shannon's entropy measure and the golden-section search method cannot guarantee to obtain an optimal input space quantization. More over, the rigid structure of the binary hierarchical CMAC neural network is another disadvantage.

1.2 Motivation and Contribution

In other to overcome the above shortcomings, this section presents the motivations of our research as well as proposes novel approaches which are considered as the contributions of this research.

Fuzzification is actually equivalent to identifying the underling distribution of each dimension of a finite size of the training patterns, so that we can apply it to the unknown data. This research aims to determine the optimal number of fuzzy sets and form clusters in the fuzzification phase to achieve higher generalization ability. In this research, we propose a novel fuzzy CMAC using Bayesian Ying-Yang learning, hereafter referred to as FCMAC-BYY. Bayesian Ying-Yang learning (BYY) [16, 17] is based on the harmony of two representations: the mapping of the input data x into an inner representation cluster c, and the generation of the input data x from an inner representation cluster c. This characteristic allows the proposed fuzzifier to derive the optimal clusters from the input training data. In the inference phase of FCMAC-BYY, the Truth-value restriction (TVR) inference scheme is used to derive the truth-values of the rule weights from the truth-value of the antecedents. BYY, together with TVR, provides the FCMAC-BYY system with an optimal fuzzy rule set, and a consistent rule base, a strong theoretical foundation, more logical and intuitive to the human reasoning process.

On the other hand, a fundamental problem in fuzzy neural network modeling is the simultaneous design of the fuzzy clusters describing qualitatively a behavior and the

neuron weighting linking this description to the output of the network. Fuzzy clusters and neuron weighting, two essential components of fuzzy inference systems, are very different in nature. Fuzzy clusters are symbolic, linguistically interpretable entities, while neuron's weights map membership values into real values output. However, Fuzzy clusters and neuron weighting are strongly interdependent and together comprise the major part of a fuzzy inference system [18]. For many applications, one of the major advantages of fuzzy systems is that they favor interpretability; therefore, finding good fuzzy systems is often an arduous task. Over the past few years, evolutionary algorithms have been employed in some stages of the fuzzy-modeling process due to their powerful search capabilities.

In the second part of this research, cooperative coevolution computation is incorporated into the FCMAC-BYY system to find the optimal structure and connection weights to propose the cooperative coevolutionary FCMAC-EBYY. The FCMAC-EBYY explores the solution domain to search for a global solution which may not be archived by the original FCMAC-BYY. As we know, the Bayesian Ying Yang learning applied to FCMAC suffers from two disadvantages: Firstly, the Bayesian Ying-Yang learning used for optimal fuzzy sets is separated from the learning part of the neural network; it may end up being locally optimized. Secondly, the weights of neurons in the FCMAC-BYY are trained by using gradient-based methods, which may fall into a local minimum during the learning process [19].

However, both the original fuzzification based Ying-Yang learning and the coevolutionary FCMAC-EBYY suffer from two main problems. First of all, the parameter learning phase can only be carried out when we already get all the training data, so that it is not suitable for online learning where the data are not always available during the training phase. Moreover, the fuzzy clusters of the original fuzzification based Ying-Yang learning are systematically optimized by the cluster number selection phase. However, in some applications the number of fuzzy clusters needs to be controlled and adjusted to reduce the output errors. This leads us to develop an online Bayesian Ying Yang learning for fuzzification which is able to automatically generate the precondition part of fuzzy rules online and the number of fuzzy clusters is controlled by a harmony function.

1.3 Report Organization

The structure of this report is outlined as follows. In chapter 2, some research work related to fuzzification, inference schemes, and evolutionary computation algorithms are introduced. The structure and methods applied to FCMAC are given in the next chapter. In chapter 4, we present the Bayesian approach to FCMAC and develop the Bayesian Ying-Yang FCMAC model. The FCMAC using cooperative coevolutionary BYY is proposed in chapter 5. In chapter 6, we developed an Online Bayesian Ying-Yang Learning applied to Fuzzy CMAC. Finally, a web-based stock prediction using online FCMAC-OBYY is described in chapter 7 followed by conclusions and future works in chapter 8.

Chapter 2

Related Work

This chapter is divided into three parts. The first part concentrates on clustering and fuzzification methods. In the second part, three fuzzy inference models are investigated. The truth value restriction scheme which is more logical and intuitive to the human reasoning process is used in our fuzzy neural network. Finally, the investigation on evolutionary computation algorithms is given.

2.1 Clustering and Fuzzification Methods

Typically, the fuzzification phase in a fuzzy neural network is fulfilled by clustering training data in each dimension independently. These clustering techniques can be classified into hierarchical-based and partition-based techniques [20, 21]. The main drawback of hierarchical clustering is static, and the data points assigned to a given cluster in the early stages cannot move to a different cluster. On the other hand, partition-based techniques are dynamic and the data points can move from one cluster to another under certain conditions. In this section, some partition-based clustering techniques and the fuzzification methods developed based on these techniques are investigated.

2.1.1 Fuzzy c-Means Clustering

Fuzzy c-Means clustering (FCM) [11] algorithm is the most common fuzzy clustering algorithm. The basic idea of FCM is very similar to *k*-Means algorithm[22]. It assumes that the number of clusters *c*, is known a *priori*, and tries to minimize the cost function:

$$J_{fcm} = \sum_{i=1}^{i=c} \sum_{k=1}^{k=n} u_{ik}^{m} d_{ik}^{2} \tag{2.1}$$

with

$$d_{ik} = \left\| x_k - v_i \right\| \tag{2.2}$$

where, u_{ik} is the membership value of instance k towards cluster i, v_i is the center of cluster i, c is the number of clusters, and n is the number of instances. And m is a weighting exponent constant which affects the membership values, determining the degree of fuzziness of cluster partition. If $m = 0$, FCM becomes the traditional k-means solution. So, m is usually chosen from 1.5 to 2.

Since the FCM uses the probabilistic constraint that the memberships of a data point across classes must sum to 1, therefore, the following constraint must be obeyed:

$$\sum_{i=1}^{c} u_{ik} = 1; k = 1, \ldots, n, \tag{2.3}$$

The cluster centroids v are computed as:

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik}^m x_k)}{\sum_{k=1}^{n} u_{ik}^m}, \forall i. \tag{2.4}$$

And the membership values are computed as:

$$u_{ik} = \left(\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}} \right)^{1/m-1} \right)^{-1} \quad \forall i, k. \tag{2.5}$$

Minimization of J_{fcm} is performed by a fixed-point iteration scheme known as the Alternating Optimization (AO) technique [23] as follows:

Table 2-1 The Fuzzy c-Means clustering algorithm

Fix the number of cluster C;
Fix m, $1 < m < \infty$;
Set iteration counter l=1;
Guess initial cluster centers
Alternating Optimization (AO)
Repeat
$l = l + 1$
update $u_{ik}^{\ l}$ using (2.5) with $v_i^{\ l-1}$
update $v_i^{\ l}$ using (2.4) with $u_{ik}^{\ l-1}$
Until (l = maximum number of iterations or $\left\| v_i^l - v_i^{l-1} \right\| \le \varepsilon$)

FCM has the advantages of unsupervised learning and always converges. However, it has the major disadvantages of long computational time, sensitivity to the initial information and the problems of "outliers", which are referred to as noise sensitivity.

2.1.2 Competitive Learning

Competitive Learning (CL) [24] is another famous learning algorithm in clustering. CL algorithm and its derivatives have been used very successfully in many applications such as: marketing, image processing, and even in education.

Actually, this algorithm has the same concept with *k*-means algorithm. It is to minimize the total distance from each data point to the cluster center it belongs to. In every loop, first check which cluster does the data point belongs to. After this, update the cluster center according to those data points assigned to that cluster. As a result, the cluster center moves towards to these data points and converge at the center of these data points.

The Competitive Learning algorithm is as follows:

Table 2-2 The Competitive Learning algorithm

Let the set of training patterns $X = \{x_1, x_2, ..., x_n\}$
Fix the number of cluster C;
Set iteration counter l=1;
η be the learning rate, initialized to a value in (0,1)
Guess initial cluster centers $V = \{v_1, v_2, ..., v_c\}$

Repeat

For i=1 **to** n

Find the centroid $v_j \in V$ that is closest to x_i

Update $v_j^l = v_j^{l-1} + \eta(x_i - v_j^{l-1})$

Decrease η

Until (l = maximum number of iterations or $\left\| v_i^l - v_i^{l-1} \right\| \le \varepsilon$)

From Table 2-2, it is observed that the competitive learning is actually a special type of probabilistic clustering. It is a Winner-Takes-All (WTA) version of probabilistic clustering algorithms. However, it has a lower complexity and usually shorter convergence time compared with other types of probabilistic clustering algorithms.

Moreover, competitive learning seems to be a noise resistance algorithm. Although outliers force the centroid toward itself, the effect of this move decreases as the number of loops increases. As a result, the problems of outliers can be overcome if the algorithm runs enough of loops.

However, there are two main problems with competitive learning. The first one is called "died unit". Imagine in a normal two clusters problem, if one centroid is initialized almost at the mean of these two clusters, and the other one is initialized very far away from these two clusters. As a result of competitive learning, both the centroids may be kept at the initial positions rather than move toward the real centers. It is because in competitive learning, only the winning centroid will be updated. If there is a centroid that always loses in the competition, so as never to be updated and it is called a "died unit".

The second problem is about the initialization of centroids. Consider there are two clusters in the data set. One of the clusters, cluster *A*, in the data set has a large variance and large number of instances, and another cluster, cluster *B*, has a small variance and small number of instances. If both centroids are initialized around the mean of the larger cluster (cluster *A*). After the learning, it may also find that rather than getting the result of two clusters with one big and one small, the result may end with the centroid of these two

clusters are very near and the centroids of these clusters are almost same as the centroid of the cluster *A*. The reason of that result is because the clusters attract the cluster centroids toward themselves; however, the attraction for the small cluster *B* is too small compared with the big cluster *A*. As a result, centroids may be trapped in a big cluster and give a wrong solution in clustering.

2.1.3 Self-Organizing Fuzzification

In this section, a self-organizing fuzzification (SOF)[9] is introduced. It is actually the improvement of DIC [12] technique by using Gaussian distribution as the membership function. SOF technique is not limited by the need of having prior knowledge of cluster number and it also preserves the dynamism of partition-based techniques. It has five parameters to control the creation and expansion of fuzzy sets: a width parameter σ, a membership threshold *MT*, a kernel threshold *KT*, a plasticity parameter β and a tendency parameter *TD*.

2.1.3.1 Cluster Creation

A new cluster (fuzzy set) is created when either the membership value of the input data point to the existing best-fit cluster, which has the highest membership value, is below the predefined membership threshold *MT* or the data point falls out the cluster. In Figure 2-1, when the data point falls in the regions x_3, a new cluster is created and its center takes the value of the data point (x_i) that triggers its creation.

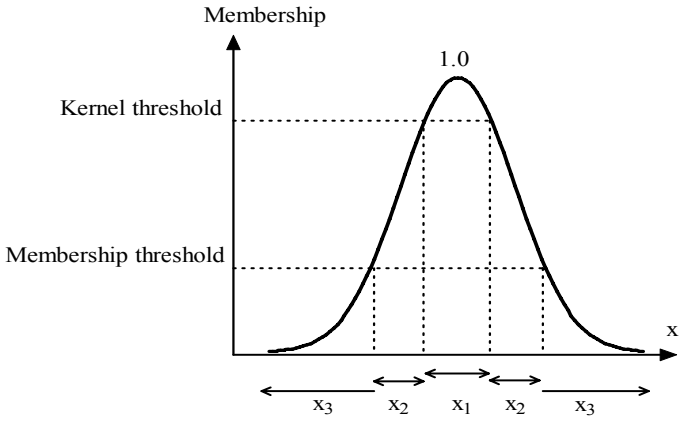


Figure 2-1 Membership regions of Gaussian cluster

Membership threshold *MT*. The *MT* defines the minimum membership value an input data point must have when it is considered as relevant to any existing input clusters or fuzzy sets. In addition, *MT* determines the degree of overlapping of an input cluster with its immediate neighbors, as shown in Figure 2-2, a larger *MT* leads to a highly quantized input space with many highly overlapped clusters, whereas a smaller MT leads to a low quantized input space with sparse clusters.

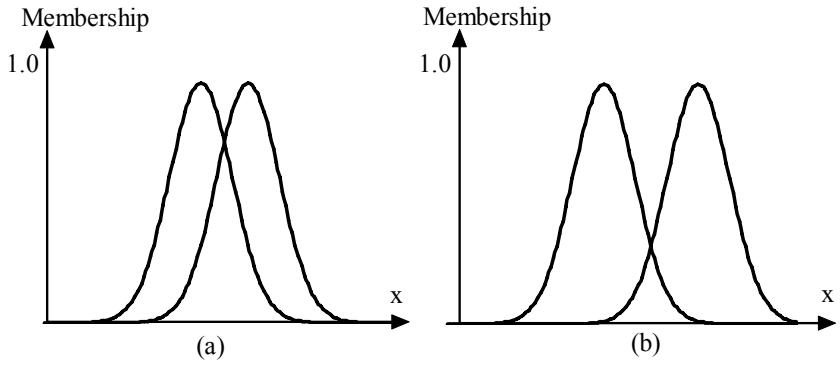


Figure 2-2 The effect of MT on the input clusters, (a) large MT with high-quantized solution, (b) small MT with low quantized solution

Kernel threshold *KT*. The Kernel threshold specifies the minimum membership value an input data point must have before it is considered as belonging to any existing input clusters or fuzzy sets. If the data point falls in a best-fit cluster's kernel region x_1 (Figure 2-1), it is considered as belonging to the cluster and that cluster does not need to expand anymore.

The width parameter σ. The width of a newly cluster is defined by the parameter σ as shown in Figure 2-3. Initially, the width of new created cluster has the value of the predefined parameter σ_0 and the cluster is defined as:

$$\mu_{i,p}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - c_{i,p}}{\sigma_{i,p}}\right)^2\right] \tag{2.6}$$

where $\mu_{i,p}(x_i)$ is the membership value of the *i*th dimension of the input x_i, and $c_{i,p}$ is the center of cluster *p*.

In the training phase, the cluster expands to include more data points by increasing its width:

$$\sigma_{i,p}^{t+1} = \sigma_{i,p}^{t} + \beta\cos(\theta^t) \tag{2.7}$$

The expansion of the clusters and two parameters β and θ are present in detail in the next section.

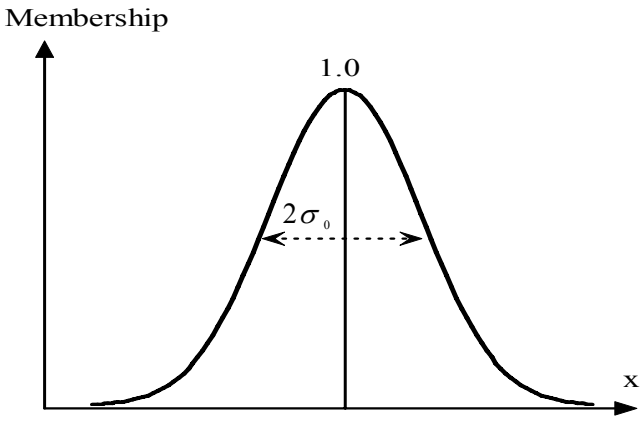


Figure 2-3 The initial width σ_0 of the cluster

2.1.3.2 Cluster Expansion

The cluster is considered to expand when the input data point falls in the regions x_2 in Figure 2-1. It means that, the membership value is greater than the membership threshold and lower than the kernel threshold.

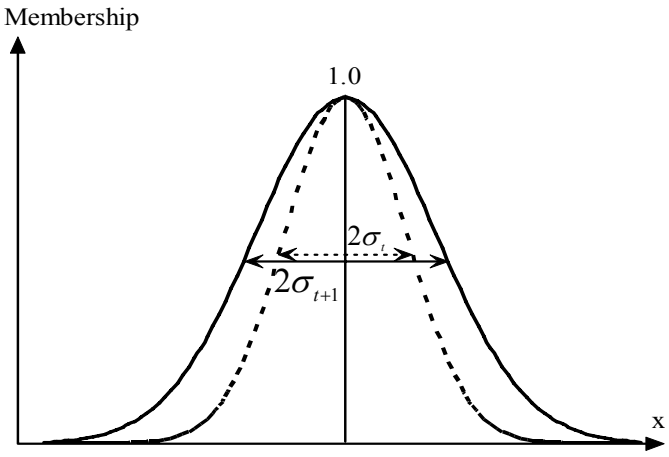


Figure 2-4 The cluster expands by increasing its width

Tendency parameter *TD*. The tendency parameter *TD* is analogous to a cluster's willingness to expand when it is the best-fit cluster to an input data that falls outside its kernel. The parameter *TD* controls the termination of cluster expansion. It decreases with the number of times when the cluster expands its width. The cluster stops expanding its kernel when *TD* reaches zero. The parameter *TD* maintains the relevance of a cluster and prevents it from incorporating too many input data that have low membership values to the cluster. Otherwise, the width of a cluster may become overly large and the semantic meaning of the fuzzy label which the cluster represents may become obscure and poorly defined. The initial value of *TD* of a newly created cluster is preset at 0.5. The rate of decrease depends on the membership value of the input data that triggers the expansion of the cluster as shown in the following equation:

$$TD_{i,p}^{t+1} = TD_{i,p}^{t} + (A - TD_{i,p}^{t}) \times (1 - \mu_{i,p}(x_i))^2 \tag{2.8}$$

where $\mu_{i,p}$ calculated by Equation (2.6) is the membership value of the input data *i* belonging to the fuzzy cluster *P* and *A*= -0.5.

When *TD* is less than or equal to zero, the cluster stops expanding and sets the plasticity parameter β to zero. It must be noted here that A has to be less than zero, otherwise, *TD* can never reach or exceed zero. This is because the value of the term $(1 - \mu_{i,p}(x_i))^2$ is in the range [0, 1) (The case $\mu_{i,p}(x_i) = 0$ is not valid since the point is then not relevant to the expanding cluster). The dynamics of the parameter *TD* is illustrated by Figure 2-5.

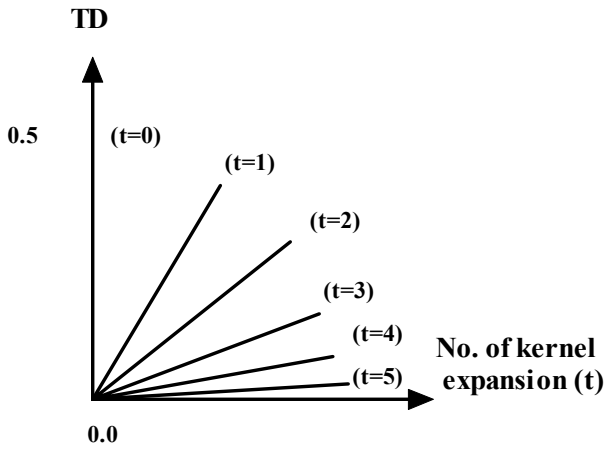


Figure 2-5 The parameter *TD* controls termination of expansion

Hence, when *TD* triggers the cluster expansion, the lower the membership value of the input data, the faster the *TD* decreases and vice versa.

The plasticity parameter β . The expansion of the cluster is controlled by the plasticity parameter β . It determines the amount of the expansion of width of the cluster (fuzzy set) to include new data points. To satisfy the *stability–plasticity* dilemma [25], the initial value for all newly formed input–output clusters is preset to 0.5. The value of the parameter β decreases as the cluster expands its width.

In Figure 2-6, the first quadrant of a cosine waveform is used to model the change of β in a cluster. The parameter θ is intuitively interpreted as the maximum expansion which a cluster (fuzzy set) can have and a predefined parameter *STEP* controls the increment of θ from 0 to 1.57 radians as described in the following equation.

$$\theta^{t+1} = \theta^{t} + STEP \tag{2.9}$$

With the increase of θ, the $\beta\cos(\theta)$ decreases the ratio to the cosine function. Thus, the amount of expansion which a cluster can adopt decreases with the number of expansions and the cluster stops expanding when $\beta\cos(\theta)$ reaches zero or θ reaches 1.57 radians.

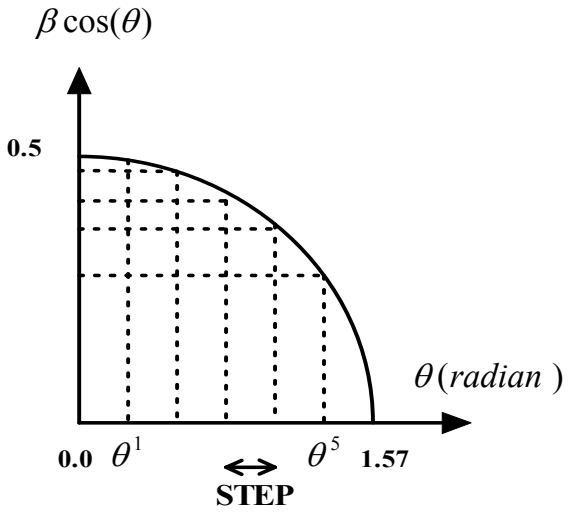


Figure 2-6 Modeling of the plasticity parameter β

Thus, the tendency parameter *TD* and the plasticity parameter β work together to maintain the integrity of the input clusters and the fuzzy labels that they represent.

2.1.3.3 SOF Algorithm

Given the input data set $X=\{x^1....x^k\}$. Initialize STEP, σ, β, *KT*, *MT*, *TD*. The *n*-dimensional input training data is represented as $x^k = \{x_1^k, x_2^k, ...x_n^k\}$.

$\forall k \in \{1...K\}$

$\forall i \in \{1...n\}$

When fuzzy label set is empty, create a new cluster using x_i^k .

Otherwise, find the best-fix cluster Winner:

Winner = arg $\max_{p\in(1...P)} \{\mu_{i,p}(x_i^k)\}$

Update the width of Winner if $\mu_{i,p}(x_i^k)$ >MT

Otherwise, create a new cluster using x_i^k .

Repeat until no cluster changes are observed.

2.2 Fuzzy Inference Methods

In this section, three fuzzy inference methods are described. They are compositional rule of inference (CRI), approximate analogical reasoning schema (AARS), and Truth value restriction method (TVR).

2.2.1 Compositional Rule of Inference Scheme

The compositional rule of inference (CRI) proposed by Zadeh [26] is one of the important inference patterns in fuzzy approximate reasoning. It defines the ways to propagate uncertainties and plays the same role in fuzzy approximate reasoning as modus pones and modus tollens do in exact reasoning. As described following, the operation produce of CRI is mainly based on the composition of fuzzy relations.

Given the fuzzy rule "if x is A then y is B" and knowing the input data is x' where x' is not exactly equal to x. The conclusion of the output y' can be derived as follows:

$$y' = x' \circ R \tag{2.10}$$

where R is the fuzzy relation determined by $A \to B$. In addition to the values of A, B, and x', there are two other factors which will influence the result for y'. The first one is the choice of the implication interpretation for $A \to B$ and the second one is how to carry out the composition operation. According to Zadeh, the composition operation $x' \circ R$ is the max-min composition of x' and R. In this sense, the output y' can be inferred from x' and R as follows:

$$y' = \max_x \min\left(\mu_A(x'), \mu_{A\to B}(x, y)\right) \tag{2.11}$$

In the following, several definitions for the fuzzy implication relation used to calculate the implication relation matrix R will be investigated.

1) Denenes-Rescher Implication:

$$\mu_{A\to B}(x, y) = \max[1 - \mu_A(x), \mu_B(y)] \tag{2.12}$$

2) Zadeh Implication:

$$\mu_{A\to B}(x, y) = \max[\min(\mu_A(x), \mu_B(y)), 1 - \mu_A(x)] \tag{2.13}$$

3) Gödel Implication:

Gödel Implication is a well known implication formula in classical logic. By generalizing it to fuzzy implications, we get

$$\mu_{A\to B}(x,y)=\begin{cases}1 & if\ \mu_A(x)\le\mu_B(y)\\ \mu_B(y) & otherwise\end{cases} \tag{2.14}$$

4) Mamdami Implications: Mamdami implications are most widely used implications in fuzzy control.

$$\mu_{A\to B}(x,y)=\min[\mu_A(x),\mu_B(y)] \tag{2.15}$$

5) Larsen Implication:

$$\mu_{A\to B}(x,y)=\mu_A(x)\cdot\mu_B(y) \tag{2.16}$$

In this section compositional rule of inference (CRI) proposed by Zadeh and several implication relations are investigated. Besides the compositional rule of inference, there are many other fuzzy inferences or fuzzy reasoning procedures. In the next two sections, the approximate analogical reasoning schema (AARS) and truth value restriction method (TVR) will be introduced.

2.2.2 Approximate Analogical Reasoning Schema

Approximate analogical reasoning schema (AARS) is an alternative fuzzy inference method which was proposed by Turksen and Zhong [27]. Given the fuzzy rule "if *x* is A then *y* is B" and the actual input data is *x'* then under the AARS method, the output of the

consequence *y'* is deduced according to the closeness of the input data *x'* using some modification techniques. As shown in Figure 2-7, the AARS has two main components: the *similarity measure* (SM) is used to obtain the similarity between *x* and *x'*, and the *modification function* (MF) is subsequently constructed and used to modify the consequence *y* of the fuzzy rule to deduce an ouput *y'*. When the input *x'* and *x* are close (similar) enough, in comparison to a threshold value τ, then the fuzzy rule is fired and the output *y'* is deduced using the modification function.

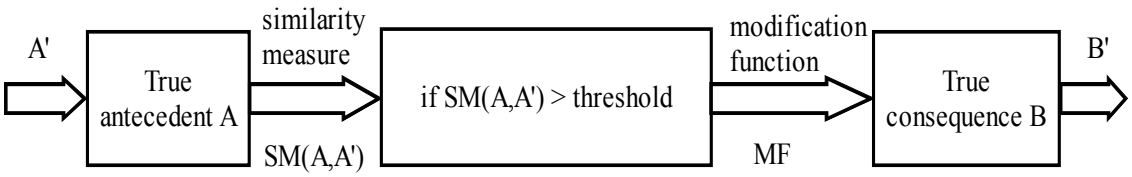


Figure 2-7 Approximate analogical reasoning schema

Similarity Measure. The similarity measure is used to judge that two fuzzy sets are "similar" or "dissimilar". Many measures of similarity have been proposed, and some have been incorporated into linguistic approximation products. Turksen and Zhong [27] defined the similarity measure (SM) between fuzzy sets as a measurement transformed from a distant measure (DM) by using SM=1/(1+DM). While as, Zwick [28] introduced a behavioral experiment in which different distance measures were generalized for fuzzy sets. The following five measures were used to distinguish between degrees of similarity or dissimilarity:

1) Disconsistency Measure:

$$S_A(A, A') = 1 - \sup_{x \in X} \mu_{A \cap A'}(x) \tag{2.17}$$

where $\mu_{A \cap A'}(x) = \min\left[\mu_A(x), \mu_{A'}(x)\right]$ for any $x \in X$.

2) Hausdorff Measure (∞):

$$q_\infty(A, A') = \sup_{\alpha > 0} q(A_\alpha, A'_\alpha). \tag{2.18}$$

3) Hausdorff Measure (*):

$$q_*(A, A') = q(A_\alpha, A'_\alpha) \tag{2.19}$$

where the Hausroff Distance is calculated as:

$$q(U, V) = \max\left\{\sup_{v \in V} \mathit{Inf}_{u \in U} d_2(u, v), \sup_{u \in U} \mathit{Inf}_{v \in V} d_2(u, v)\right\} \tag{2.20}$$

and $d_2(U, V)$ is the Euclidean Distance. Note that the Hausdorff Measure (*): is actually a special instance of the Hausdorff Measure (∞).

4) Kaufman and Gupta Measure (∞):

$$\Delta_\infty(A, A') = \sup_{\alpha > 0} \Delta(A_\alpha, A'_\alpha). \tag{2.21}$$

5) Kaufman and Gupta Measure (*):

$$\Delta_*(A, A') = \Delta(A_\alpha, A'_\alpha) \tag{2.22}$$

where

$$\Delta(A_\alpha, B_\alpha) = \frac{\left(|a_1 - b_1| + |a_2 - b_2|\right)}{2\left(\beta_1 - \beta_2\right)} \tag{2.23}$$

and $[a_1, a_2]$, $[b_1, b_2]$ are the supports of A_α, B_α, respectively, $[\beta_1, \beta_2]$ is the support of both A_α and B_α, $\alpha \in [0,1]$. Similarly, Kaufman and Gupta Measure (*) is a special instance of Kaufman and Gupta Measure (∞).

Modification Functions. There are two types of modification functions introduced by Turksen. They are as follows:

1) Expansion form (More or Less form): The derived conclusion *y'* is obtained using:

$$y' = \min\left(1, \frac{B}{SM}\right)$$
$$\mu_{y'} = \min\left(1, \frac{\mu_y}{SM}\right) \tag{2.24}$$

where μ_y is the membership function of the consequence y of the rule and $\mu_{y'}$ is the membership function of the derived conclusion *y'*.

2) Reduction form: In some sense, this is the imitation of CRI, the derived conclusion *y'* is obtained using:

$$y' = SM \times y$$
$$\mu_{y'} = SM \times \mu_y \tag{2.25}$$

From the above equations, one can observe that the smaller the SM, the greater the reduction of the membership values of y. This imitates the effect of the Min operator in CRI.

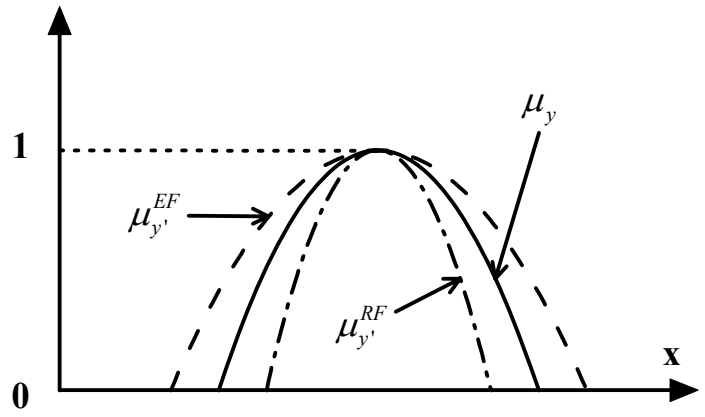


Figure 2-8 Expansion form and reduction form of the modification function

Figure 2-8 shows the effect of the expansion form and the reduction form of the modification function. Obviously, in both types of the modification function, when the input data *x'* and the condition of the fuzzy rule "if *x* is A then *y* is B" are exactly matched (*x'* = *x*), the conclusion *y* = *y'* will be derived due to the maximum of the similarity measure SM = 1. This is one of the advantages of AARS over CRI.

2.2.3 Truth Value Restriction Inference Scheme

Truth value restriction method (TVR) [29, 30] offers a consistent rule base, a strong theoretical foundation and is more logical and intuitive to the human reasoning process as

compared to other alternative techniques such as compositional rule of inference scheme (CRI) [26, 31], Approximate Analogical Reasoning Schema (AARS) [27, 32].

Truth-value restriction uses implication rules to derive the truth-values of the consequents from the truth-value of the antecedents. In the TVR methodology, the degree to which the actual given value of *A'* of a variable *x* agrees with the antecedent value *A* in the proposition *"*IF *x* is *A* THEN *y* is *B"* is represented as a fuzzy subset of truth space $\tau_{AA'}$. This fuzzy subset of truth space, known as truth-value restriction, is used in a fuzzy deduction process to determine the corresponding restriction on the truth-value of the proposition *"y* is *B"* represented by $\tau_{BB'}$. The latter truth value restriction is then 'inverted', which means that a fuzzy proposition *"y* is *B'"* in the *Y* universe of discourse is found such that its agreement with *"y* is *B"* is equal to the truth value restriction derived by the fuzzy inference process as Figure 2-9.

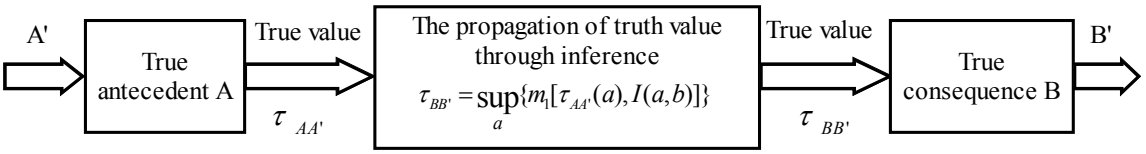


Figure 2-9 Truth value restriction method in fuzzy inference

The fuzzy subset $\tau_{AA'}$ of the truth space is given in the following equation:

$$\tau_{AA'} = \begin{cases} \sup_x \{\mu_{A'}(x) \mid x \in \mu_A^{-1}(a)\}, & \mu_A^{-1}(a) \neq \phi \\ 0, & otherwise \end{cases} \tag{2.26}$$

where μ_A and $\mu_{A'}$ are the respective membership functions of the fuzzy set A and A' defined on the universe of discourse X; x denotes a value in X; a is the membership value of x in fuzzy set A; $\mu_A^{-1}(a)$ is the set of values of x in X that take membership value a in the fuzzy set A; and ϕ denotes the empty set or null set.

The truth-value $\tau_{BB'}$ of the consequent "y is B" can be computed using the following equation:

$$\tau_{BB'}(b) = \sup_a \{m_I[\tau_{AA'}(a), I(a,b)]\} \tag{2.27}$$

where m_I is the forward reasoning function (usually a T-norm operation); and I is the implication rule.

The truth-value $\tau_{BB'}$ is subsequently "inverted" to determine the inferred conclusion. That is, a fuzzy proposition "y is B'" is computed using the *truth function modification* (TFM) process such that the degree of possibility that the proposition "y is B" is true given "y is B'" is described by the truth-value $\tau_{BB'}$. That is, the derivation of the fuzzy set B' from the truth-value $\tau_{BB'}$ is performed using the below equation:

$$\begin{aligned} B' &= \tau_{BB'} \circ B \\ \Rightarrow \underbrace{b'}_{\mu_{B'}(y)} &= \tau_{BB'}(\underbrace{b}_{\mu_B(y)}) \\ \Rightarrow \mu_{B'}(y) &= \tau_{BB'}(\mu_B(y)) \end{aligned} \tag{2.28}$$

where b and b' are the respective truth-values of the propositions "y is B" and "y is B'"; and μ_B and $\mu_{B'}$ are the membership functions of the fuzzy sets B and B' respectively.

While as in CRI, the input fuzzy set is directly used to obtain the consequent fuzzy set, both AARS and the TVR methods can be classified as truth degree based fuzzy inference models. On contrary to CRI, in truth degree based fuzzy inference models, the truth degrees of the input facts are employed instead of the original fuzzy sets to deduce the final conclusion. Such truth degree based fuzzy inference processes have the advantage of independent of the particular membership function involved. In fact, in those methods, the truth degree between the observed fact and the condition of the fuzzy rule is transformed into a final conclusion by a function, whether directly or indirectly.

Although both AARS and the TVR method use the truth degree to deduce the conclusion, they have different ways of representing the truth degree. In the TVR method, truth degree is represented by a fuzzy set that is the truth value. While in AARS, truth degree is represented by a real value that is the similarity measure. Comparing these two forms, one can observe that the crisp nature of the similarity measure makes the inference process of AARS simpler, however, it decreases the accuracy of the system. While the truth value in the TVR method is able to capture more global and intuitive features of the observed fact.

In the TVR model, the computed truth-values of the antecedents can be effectively propagated in the hybrid structure of a neural fuzzy system. The truth-value of the

proposition in the antecedent is computed and allowed to propagate through the network. It is this value that is used to calculate the proposition in the consequent. This treatment makes the TVR a viable inference scheme for implementation in a neural fuzzy system.

2.3 Survey on Evolutionary Computation

This section introduces Evolutionary Computation (EC) which is emerging as a new engineering computational paradigm.

2.3.1 Evolutionary Computation

Evolutionary Computation (EC) is a modern search technique which uses computational models of processes of evolution and selection. Evolutionary algorithms (EAs) are motivated by the concepts and mechanisms of Darwinian [33] evolution and natural selection to solve problems in many fields of engineering and science. Each of these techniques constitutes a different approach; however, they are inspired by the same principles of natural evolution. EC uses fitness functions and competitive evolution as the evaluating methodologies, and emulates biological evolutionary theories to solve the complex optimization problems. EC has been widely used for generating fuzzy rules and tuning membership functions [34] and successfully applied to many applications [35-40].

Strong resemblance to biological processes as well as their initial applications for modeling complex adaptive systems [34] influenced the terminology used by EC researchers. It borrows a lot from genetics, evolutionary theory and cellular biology. Thus, a candidate solution to a problem is called an individual while the entire set of

current solutions is called a population. For some problem domains, a population may be broken into several sub-populations. The actual representation of an individual is called a genome or a chromosome. Each genome consists of a sequence of genes, i.e., attributes that describe an individual. A value of a gene is called an allele. When individual solutions are modified to produce new candidate solutions, they are said to be breeding and the new candidate solution is called an offspring or a child. During the evaluation of a candidate solution, it receives a grade called fitness, which indicates the quality of the solution in the context of a given problem. When the current population is replaced by offspring, the new population is called a new generation. Finally, the entire process of searching for an optimal solution is called evolution.

2.3.2 Evolutionary Algorithms

Evolutionary algorithms are a family of population-based search algorithms that simulate the evolution of individual structures by interrelated processes of selection, reproduction, and variation. There is a variety of EAs that have been proposed and studied. They all share a common set of underlying assumptions but differ in the breeding strategy to be used and representation on which EAs operate.

Historically, three major EAs have been developed: Evolution Strategies (ES), Evolutionary Programming (EP), and Genetic Algorithms (GAs) [34]. These algorithms have been mostly used to evolve solutions to parameterized problem domains. On the other hand, the fourth major EA developed more recently, Genetic Programming (GP)

[41], has been used to evolve actual computer programs to solve a number of computational tasks.

Basically, EC can be understood as a search and optimization process in which a population of solutions undergoes a process of gradual changes. This process depends on the fitness as a formal measure of perceived performance of the individual solutions as defined by the environment which is called objective function.

A traditional EA consists of the following steps:

1. Initialize the population
2. Evaluate all members of the population
   While the termination condition is not satisfied
   {
   3. Select individual(s) in the population to be parent(s)
   4. Create new individuals by applying the variation operators to the copies of parent(s)
   5. Evaluate new individuals
   6. Replace some/all of the individuals in the current population with the new individuals
   }

Before an actual evolutionary process begins, an initial population of individuals/solutions is created. Traditionally, the initial population is created randomly by several other initialization techniques have also been used or started from a set of previously known or arbitrarily assumed solutions. After that, each individual in the initial population is then evaluated and assigned a fitness value.

Using the fitness scores, the selection mechanism chooses a subset of the current population as parents to create new individuals. When the selection mechanism uses bias toward individuals with better fitness, the created offspring will, more likely, have higher fitness. Once the set of parents has been selected, the new individuals are created by copying them and applying variation operators.

There are several common selection strategies used within EC community. Fitness-proportional selection [34] normalizes the fitness values of all individuals in the population and assigns these normalized values as probabilities that their respective individuals will be selected. Ranked selection works by first ranking all individuals in the population by their fitness, and use these ranks, rather than actual fitness values, to determine selection probabilities of the individuals. A common form of ranked selection is a linear ranking where individuals are first sorted in an increasing order according to their fitness values. Each individual is then selected with a probability based on some linear function of its sorted rank.

Another popular selection strategy is a tournament selection. In this strategy, a pool of n individuals is picked at random from the population. Each of the individuals in the pool is selected independently and it might be the case that the same individual will be selected multiple times. After that, an individual from the pool with highest fitness value is selected to form the new population. This procedure is repeated as many times as necessary to create either an entirely new population or a subset of it. The pool size is a parameter that controls the magnitude of the selection pressure.

Finally, the truncation selection chooses only a certain proportion of the best individuals in the population. This strategy is the most popular within the ES community, where it is used in two basic objective functions: (μ, λ) and (μ+λ). In the former case, the selection operates on the offspring population only, whereas in the latter case, it selects individuals from a joint population of both parents and offspring.

The two most popular variation operators are mutation and recombination. Mutation acts on a single individual and works by applying some variation to one or more genes in the individual's chromosome. This operation is similar to a variation operator used in other search mechanisms like hill climbing or simulated annealing. On the other hand, recombination operates on multiple individuals (usually two) and combines parts of these individuals to create new ones. The newly created individuals are evaluated and assigned fitness values. Then, either all or only a subset of the current population is replaced by these new individuals. If the entire population is replaced by the new individuals then the algorithm is called generational EA. Otherwise, if only a subset of the original population is replaced then the algorithm is called a steady-state EA. Steps 3-6 of the canonical EA defined earlier are performed until an assumed stopping criterion is met, which is usually defined as an arbitrary number of generations or fitness function evaluations.

2.3.3 Evolutionary Computation and Neural Network Design

This basic evolutionary process described above is called a "simple evolutionary algorithm" in a sense that it contains the minimal set of features necessary to be a

Darwinian evolutionary system. These simple EAs have surprisingly useful properties, primarily related to solving difficult global optimization problems. They perform well when applied to problems with nonlinear, stochastic, temporal, or chaotic components, where the traditional optimization techniques, like gradient descent, hill climbing, and purely random search, are generally unsatisfactory. It is in this context that much of the work on neural network applications has taken place historically: using simple EAs for design optimization.

The three main issues in applying EAs to a neural network design problem are:

1. Selecting an appropriate representation for neural network designs.
2. Defining efficient genetic operators.
3. Providing an adequate evaluation function for estimating the "fitness" of generated solutions (hypothesis space).

An appropriate representation of a neural network system is one of the most crucial elements of evolutionary design. This issue is particularly important when creativity/novelty of designs produced in evolutionary processes is one of the major goals. The process of creating an efficient and adequate representation of a neural network system for evolutionary design is complicated and involves elements of both science and art. One has to take into account not only important aspects of understanding traditional modeling of a neural network system, but also relevant computational issues that include search efficiency, scalability, and mapping between a search space

(genotypic space) and a space of actual designs (phenotypic space). A more detailed discussion of EA representations is presented in Section 5.

Appropriate choice and implementation of genetic operators, i.e., mutation and recombination operators, and careful tuning of their rates is an important issue as it can have a big impact on the success of EAs and has, therefore, been a subject of both theoretical as well as experimental investigations. Any particular implementation of a mutation or recombination operator is representation dependent. Thus, for example GAs with binary string representations use the bit-flip mutation and 1-, or 2-point crossover, while ES with real-valued vectors use the Gaussian mutation and a recombination operator that swaps/averages parents' alleles. Genetic operators are primary sources of exploration in EAs. On the other hand, selection mechanisms provide EAs with exploitative power. Thus, by properly defining and controlling the variation mechanisms (genetic operators), one can achieve a higher level goal of finding "an effective balance between further exploration of unexplored regions of the search space and exploiting the regions already explored".

Another important issue in successful application of EAs is to choose an adequate fitness evaluation function for a problem domain. Evaluation functions provide EAs with feedback about the fitness of each individual in the population. EAs use this feedback to bias the search process in order to improve the population's average fitness. Naturally, the details of a particular fitness function are problem specific. Table 2-3 modified from [42] provides a description of all commonly used EAs in terms of decisions that are made

during an implementation of a particular EA. The particular decisions are summarized in terms of attributes and their values. Using this characterization, it is then straightforward to describe a given EA, GA or ES, and its relationship to other EAs.

Table 2-3 Summary on attribute used in EA implementation

No	Attribute			Attribute Values				
				1	2	3	4	5
1	Solution representation	Encoding		Binary	Real-valued	Grap-based	Computer code	Other
		Length		Fixed	Variable			
2	Population initialization	Mechanism		Random generation	Selection from a group of known solutions	User defined		
		Population size		1	Fixed	Variable		
3	Parent selection mechanism			Truncation	Ranking	Fitness proportional	Tournament	Uniform
4	Variation mechanism	Mutation	type	Bit-flip	Gaussian	Sub tree	User defined	
			rate	0	Fixed	Adaptive	Random	
		Crossover	type	N-point	Swap	Uniform	Sub tree	User define
			rate	0	Fixed	Adaptive	random	
5	Survival selection mechanism			Truncation	Ranking	Fitness proportional	Tournament	Uniform

2.3.4 Coevolutionary Computation

Another important branch in evolutionary computation research that has recently received significant research attention is coevolution. Coevolutionary computation is referred as a phenomenon occurring when two or more populations simultaneously evolve. Coevolutionary computation models consist not a single objective fitness function but each individual's fitness is a subjective function of its interactions with other individuals from coevolving populations [18, 43]. Biologicalcoevolution encountered in many natural processes has been an inspiration for coevolutionary algorithms. Initial ideas of modeling coevolutionary behavior were formulated by Maynard Smith [44] and Axelrod

[45]. These ideas were then further extended and resulted in a new optimization procedure called a coevolutionary genetic algorithm. Coevolutionary computation can be classified into two main categories: competitive coevolution and cooperative coevolution, the details are given as follows.

The competitive approach to coevolution has been since widely used in many game-theoretic models that arise in various disciplines, including economics, decision sciences, social sciences, etc. Competitive coevolutionary models are especially suitable for problem domains where it is difficult to explicitly formulate an objective fitness function such as AI game-playing strategies, etc. Recently, competitive coevolutionary models have been employed to co-evolve cellular automata and the training cases for the majority classification problem [46] or applied to constrained optimization problems [47]. More details on competitive coevolution can be referred in [48].

Another approach to coevolution is cooperative coevolutionary models [18, 49, 50]. The motivation for these models comes from problem domains where explicit notions of modularity have to be introduced. This model also provides appropriate framework for evolving solutions in the form of co-adapted subcomponents, and hence it is suitable for many network design problems. Usually, complex problems are decomposed into simpler problems and solved independently. Cooperative coevolutionary works fine for problem applications where the principle of superposition can be applied, i.e., for problems that can be linearly decomposed as well as complex designs where nonlinear interactions take place among the subcomponents and make interacting members highly dependent on one

another. For these applications, cooperative coevolutionary model is more suitable because it allows for an explicit subcomponent co-adaptation.

2.4 Summary

In this chapter, some representative clustering and fuzzification models, inference schemes, and evolutionary computation algorithms are introduced. The Fuzzy c-means clustering has the advantages of unsupervised learning and always converges, but it suffers from the "outliers" problems. On the contrary, the competitive learning shows robustness against noise but suffers from the problems of "died unit". Both these two methods are very sensitive to the initial information and require a prior knowledge of number of clusters.

Unlike hierarchical-based and partition-based clustering techniques, the self-organizing fuzzification and evolving self-organizing fuzzification methods require no prior knowledge of the number of clusters and they also preserve the dynamism of partition-based techniques. However, like most of the clustering techniques, they are very sensitive to initial values when creating new clusters. They simply apply the same values to initialize new clusters for all dimensions; however the data distributions of dimensions in the real applications are sometimes very different. Therefore, they cannot guarantee to obtain an optimal cluster architecture for the various input data. This problem will be addressed by the Bayesian Ying Yang fuzzification proposed in Section 4.

In our research, TVR inference scheme is used to derive the truth-values of the rule weights from the truth-value of the antecedents. In comparison with the CRI, the truth value restriction method is more logical and human-like reasoning due to its truth degree-based fuzzy inference process. Moreover, the truth value in the TVR is able to capture more global and intuitive features of the observed fact as compared to the crisp similarity measure of the AARS method.

The field of evolutionary design continues to rapidly grow and develop in many exciting new directions, and evolutionary computation is becoming a computational paradigm that is increasingly attractive for researchers. In this section, the investigation of several evolutionary computation algorithms is given. The descriptions and related works on coevolutionary computation models are discussed. Furthermore, in chapter 5, the advantages of cooperative coevolutionary computation learning are integrated in our FCMAC-BYY to search for the global optimal solution.

In the next chapter, the overviews of the original CMAC as well as the operation of FCMAC structures are first presented. The Hierarchical FCMAC structure, which consisted of multiple 2-dimensional differentiable FCMACs arranged in a tree-like structure, is then briefly introduced.

Chapter 3

Fuzzy CMAC Structures

In this chapter, the classical CMAC is first reviewed. The operation of FCMAC is then described, followed by the construction algorithm. The structure of FCMAC given here is a general framework in which the fuzzification phase can be carried out by the self-organizing fuzzification introduced in chapter 2 or the Bayesian Ying-Yang fuzzification proposed in chapter 4. Finally, the Hierarchical FCMAC structure is also briefly introduced.

3.1 Classical CMAC

CMAC applies a table-lookup technique and has local generalization ability that are dependent on the overlap of the association vectors [1, 2]. In this technique, the input space is divided into discrete states and each state is quantized. The input space of CMAC algorithm can be illustrated as in Figure 3-1 which shows the structure of CMAC with two-dimensional and 4 overlapping layers input. The *n*-dimensional input vector is $\mathbf{x}=[x_1, x_2,\ldots, x_n]^T$, and each dimension input x_i is quantized into discrete regions, called blocks. Each two adjacent blocks overlap with each other. The generalization ability of CMAC is determined by the width and the lapping degree of the blocks.

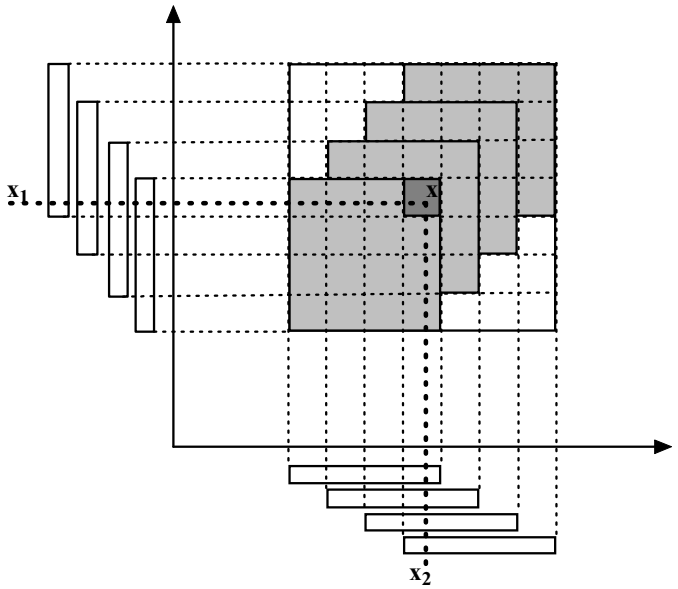


Figure 3-1 Illustration of the input space of CMAC

The traditional CMAC uses constant value assigned to each block; therefore, the overlapping degrees are uniform. *CMAC* has attractive characteristics of local generalization, output superposition and incremental learning. It also has a faster learning time because of the limited number of computations per cycle so that is easily realizable in hardware. However, *CMAC* suffers from inherent disadvantages like its inefficiency in storing data storage, since the required memory size grows exponentially with respect to the number of input variables and its weak performance in classifying inputs which are similar and highly overlapping. In CMAC, the data for a quantized state are constant and the derivative information is not preserved. In order to overcome this problem, FCMAC uses fuzzy set (fuzzy label) as the input clusters instead of crisp set. While only numeric values can be associated with CMAC, Fuzzy CMAC can model a problem using linguistic variables based a set of If-Then fuzzy rules. Thus, the FCMAC network becomes more robust, highly intuitive and easily comprehended.

3.2 FCMAC Structure

The main difference between the FCMAC and the original CMAC is that the association layer in the FCMAC is the rule layer and each associative cell represents a fuzzy rule that links input cluster to the output cluster, so that the input data is first fuzzificated into fuzzy clusters before fed into the system. As shown in Figure 3-2, the FCMAC neural network can be viewed as a 5-layer hierarchical structure as follows:

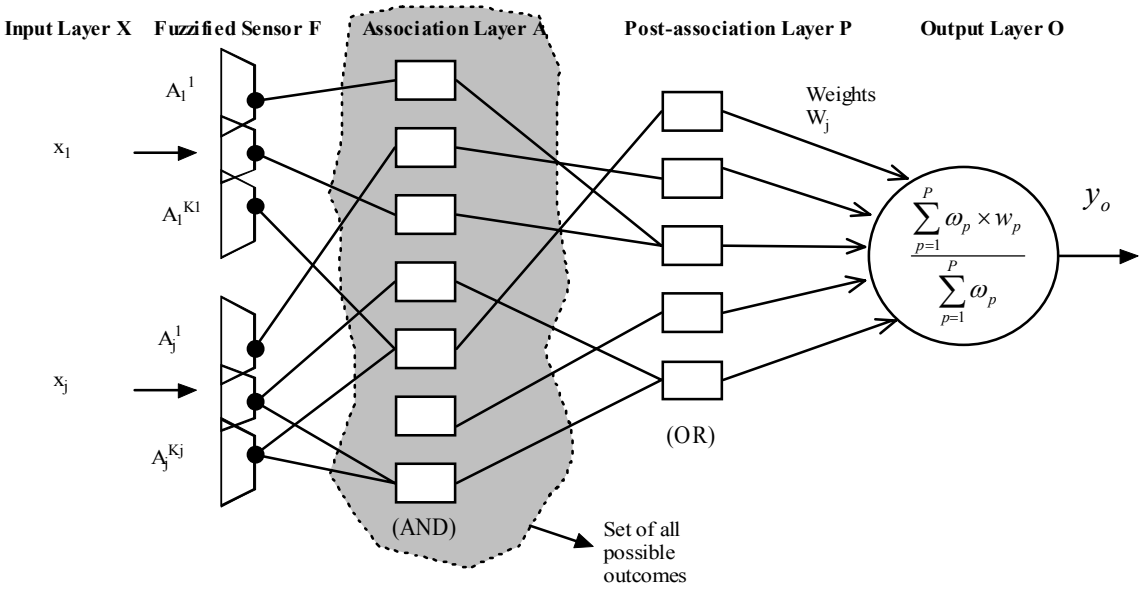


Figure 3-2 Block diagram of FCMAC

(1) **Input space X.** This is the first layer where the input X is obtained from the raw data (or using sensors for hardware realization).

(2) **Fuzzicatied sensor F**. In this layer, the fuzzification method, which is either self-organizing fuzzification (SOF) or Bayesian Ying-Yang fuzzification (BYY), is conducted

on the input training data set to obtain fuzzy labels. Each neuron (sensor) in this layer represents a particular cluster and a group of sensors is associated with input variables. In contrary to the binary outputs of the traditional CMAC, the outputs of these sensors are real numbers from 0 to 1, which are corresponded to their membership values.

(3) **Association Layer A.** This layer is the rule layer and each association cell represents a fuzzy rule. In the case of lack of memory, this layer is considered as a *conceptual/logical memory space.* The *AND* operation is carried out to ensure that any cell is active only when *all the inputs* to it are fired. The weight of fuzzy rules is derived by truth value restriction scheme in this research.

(4) **Post association Layer P.** To address the problem of a large memory size required in Layer A, it can be mapped to a *physical memory space* P. This is done by either *linear mapping or hashing* [9]. The *logical OR operation* makes any cell in this layer fired if *any of its connected inputs* is activated.

(5) **Output Layer O.** This layer is fully connected to layer P. The defuzzification center of area (COA) method is used to compute the output of the structure.

3.2.1 Fuzzification

First of all, for each dimension, either Self-organizing Fuzzification (SOF) introduced in chapter 2 or Bayesian Ying-Yang fuzzification (BYY) described in chapter 4 is

conducted on the input training data set to obtain fuzzy clusters. Each fuzzy cluster is represented by a neuron in the Fuzzified Sensor Layer. Each combination of the fuzzy cluster in this layer becomes a neuron in the next layer, Association Layer.

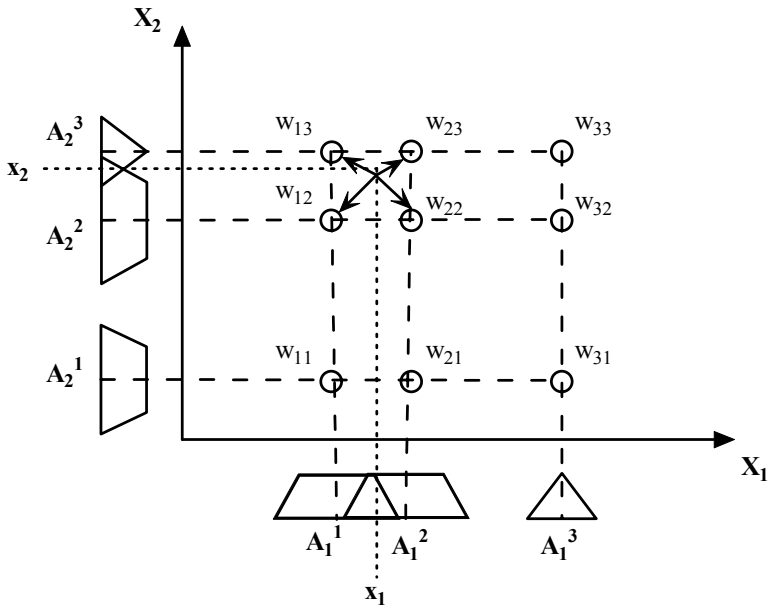


Figure 3-3 Illustration activated cells by input data X in the sensor layer

As illustrated in Figure 3-3, two-dimensional data are considered with three fuzzy clusters, i.e. A_j^i, represented in each dimension in the sensor layer. There are nine combinations of these fuzzy clusters; it means that there are nine cells in the association layer. Each cell represents a fuzzy rule. However, a cell is activated when all its corresponding fuzzy clusters are activated by the input data. In Figure 3-3, only four neurons (w_{12}, w_{22}, w_{13}, w_{23}) of the corresponding clusters $A_1^1, A_1^2, A_2^1, A_2^2$ are activated by the input data $X = \{x_1, x_2\}$. The strength of activation or the membership value of the

input data and the fuzzy cluster depends on the Euclidean distance between them. In other words, the closer the data point to the cluster, the higher the membership value.

3.2.2 Inference Scheme

In this research, the Truth value restriction (TVR) inference scheme described in Section 2.2.3 is used to derive the truth-values of the rule weights from the truth-value of the antecedents. Given the input-output data $(x\ , y\ ) = (x_1, x_2, ..., x_n, y\ )$, the simplified fuzzy inference rules are shown by the following implications:

$$\text{Rule}^p = \text{If } x_1 \text{ is } A_1^{kj}, x_2 \text{ is } A_2^{kj}, ..., x_n \text{ is } A_n^{kj} \text{ Then } y \text{ is } w_p \tag{3.1}$$

where $p=1,2, ..., P$, x_j is the jth input variable, y is an output variable, P is the number of fuzzy rules and w_p is the weight of the pth fuzzy inference rule. While A_j^{kj} is the kth fuzzy cluster of the jth dimension, $k^j=1,2,...,K^j$. The total fuzzy cluster number K^j of each dimension is obtained by the fuzzifier. Given the input data x, the total matching degrees ω_p of the antecedent of rule pth using TVR inference:

$$\omega_p = \prod_{j=1}^{n} \mu_{A_j^{kj}}(x_j), \qquad \text{for j=}1,2, \ldots, n \tag{3.2}$$

where $\mu_{A_j^{kj}}(x_j)$ is a membership value, and ω_p is the total membership value of the antecedent part.

3.2.3 Defuzzification

The output of FCMAC is derived by the following equation using center of area (COA) method [51]:

$$y_o = \frac{\sum_{p=1}^{P} \omega_p \times w_p}{\sum_{p=1}^{P} \omega_p}, \quad \text{for } p=1,2,\ldots,P \tag{3.3}$$

3.2.4 Hashing

In high-dimension problems, the number of input sensors is the number of clusters obtained from the clustering methods; this may result in a very large lookup table in the association layer. For illustration, let us assume that there are nine inputs and each input is classified into three fuzzy sets. This would give us a huge number of $3^9 = 19683$ table entries. To circumvent this problem, our model makes use of a hashing algorithm.

Hashing has the effect of mapping a large table from the logical memory to a smaller table in the physical memory [52]. If we assume that the address of the large logical memory table ranges from 1 *to* N, and the address of the small physical memory table ranges from 1 *to* n, then the hash-coding function $H(k)$ must satisfy:

$$1 \le H(k) \le n, \quad \text{where} \quad 1 \le k \le N \tag{3.4}$$

Hash-coding performance is typically measured by the number of collisions. Extensive tests have shown that the following hash-coding function works well:

$$H(k) = 1 + k \bmod n \tag{3.5}$$

where $k \bmod n$ means the remainder modulo n.

When we can obtain the correct set of weights for a particular set of inputs, we can calculate the output using the Equations (3.2) and (3.3). It is noticed that the weighted average mean approach is used to calculate the output rather than just the simple mean.

3.2.5 Training

Similar to CMAC, the Least Mean Square (LMS) approach is adopted to train the FCMAC network. The weights between the post-association layer and the response unit (Figure 3-2) are initialized to zero and then updated through training.

The weights are updated to achieve least mean square errors:

$$w_p^{t+1} = w_p^t + \eta \times \left(y^d - y_o \right) \tag{3.6}$$

where $0 < \eta < 1$, y^d is the desired output and y_o is the output of the FCMAC calculated by (3.3).

The training stops either when the specified number of cycles or completed or the error converges below a certain threshold *ke*:

$$y^d - y_o < k_e \tag{3.7}$$

3.2.6 Construction Algorithm

The detailed algorithm to construct the proposed FCMAC is shown as follows:

Step 1: The raw input data is presented to the network.

Step 2: Either Self-organizing Fuzzification (SOF) technique or Bayesian Ying-Yang fuzzification (BYY) is then conducted on this input data set to obtain the fuzzy sets.

Step 3: The fuzzy sets are fed into the network and fuzzification is done on the data sets to obtain the membership functions and store the respective weights in the look-up table.

Step 4: The tolerance error of the inference rules, ε, is specified.

Step 5: Hashing is done on the larger look-up table to obtain the smaller physical memory table.

Step 6: The fuzzy inference rules are constructed by continuously updating the weights according to Equation (3.6).

Step 7: The training stops either when the specified number of cycles are completed or the error converges below a certain threshold.

3.3 Hierarchical FCMAC

Hierarchical FCMAC (HFCMAC) was presented by Han-Ming Lee [53] which consisted of multiple 2-dimensional differentiable FCMACs arranged in a tree-like structure. The unique structure reduces large dimensional problems into smaller two-dimensional ones, which require less memory.

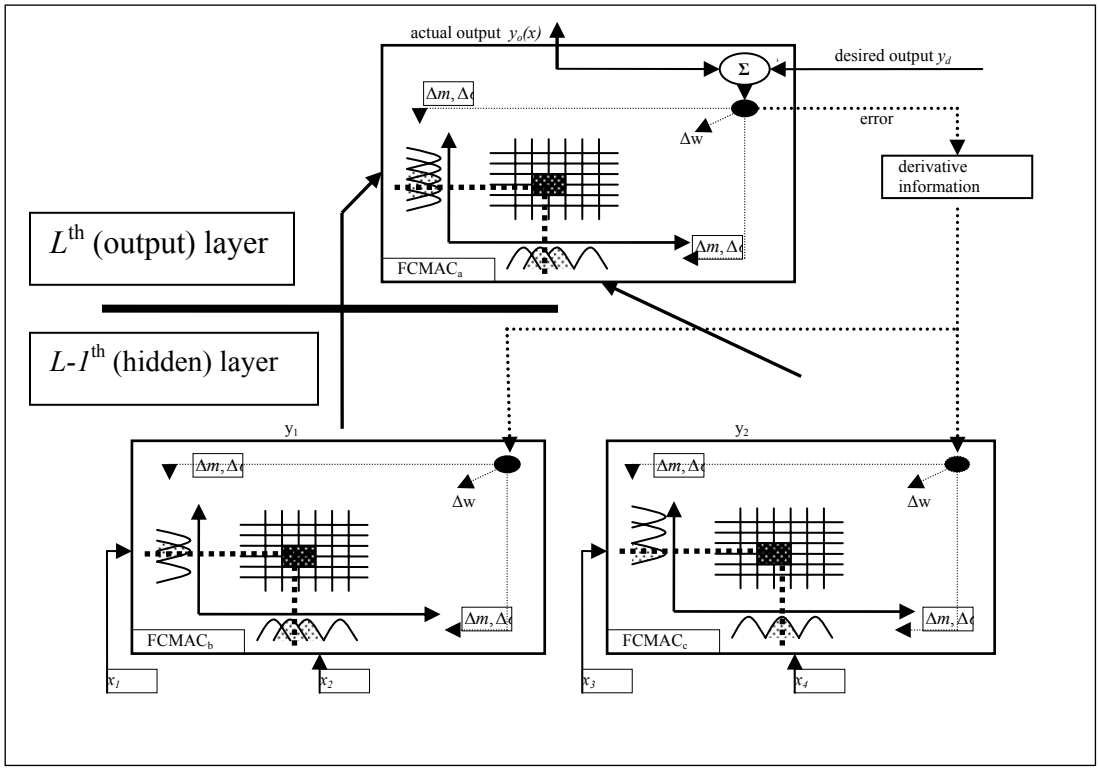


Figure 3-4 A detailed illustration of HFCMAC

An overview of a simple 4-input HCMAC with 2-dimensional FCMAC is shown in Figure 3-4. The FCMACs are arranged in a binary tree structure with the output as the root. Inputs for the output layer (layer L) are provided by the outputs of the 2 FCMACs in

the previous layer (layer L-1). This basic structure can be expanded to accommodate more inputs by cascading the structure to increase the number of hidden layers.

If Gaussian membership function is used, the output of each FCMAC of the conventional HFCMAC using 2-dimensional FCMACs [15] can be mathematically expressed as follows:

$$y_o = \sum_{j=1}^{N_h} \left[a_j(x).w_j \left(\prod_{i=1}^{N_v} e^{-(x_i - m_{ji}/\sigma_{ji})^2} \right) \right] \tag{3.8}$$

where N_v is the total number of dimensions (in this case 2), x represents an input vector, $a_j(x)$ is the jth neuron of association memory selection vector for a specific input state x,, w_j is corresponding weight of the jth neuron. m_{ji} and σ_{ji} are the corresponding hypercube center and the corresponding hypercube radius, respectively.

The derivative information takes the error from the upper levels and computes the error in the lower level FCMACs so that the learning rules of the FCMAC in the L^{th}-l hidden layer can be derived as FCMAC from the L^{th} hidden layer. And similarly, the learning rules of the FCMAC in the L^{th}-2 hidden layer can be derived and so on. In summary, the difference between the desired and the actual outputs must be first back-propagated from the output layer to the L^{th}-l hidden layer using the derivatives information, continue to be back-propagated from the L^{th}-l hidden layer to the input layer using the derivatives information.

3.4 Summary and Discussion

FCMAC uses fuzzy set as input cluster, rather than crisp sets in the original CMAC. While only numeric values can be associated with CMAC, Fuzzy CMAC can model a problem using linguistic variables based on a set of If-Then fuzzy rules. This feature enables FCMAC to reduce the memory requirement of the network by a great deal as compared to the traditional CMAC. Moreover, fuzzy CMAC can provide a human-like thinking ability and is able to incorporate domain expert prior knowledge.

As mentioned in Section 3.3, Hierarchical Fuzzy Cerebellar Model Arithmetic Controller (HFCMAC) neural network was introduced to solve the memory hungry problem [53] while reduce the complexity of high dimensional problems and be able to preserve data. Using a HFCMAC structure, a high dimensional problem is divided into smaller two-dimensional blocks and thus greatly reduces the memory requirement as compared to a conventional FCMAC. However, the derivation of updates for each FCMAC block followed the hierarchical structure makes the entire network more complex. The output error is back-propagated from the output layer though all the hidden layers to the input layer by very complex computations. This prevents some fuzzification methods from finding the optimal fuzzy rule for each layer as well as the optimal solution for the whole structure in result.

Inspired by the famous ancient Chinese Ying-Yang philosophy, in the next chapter, we propose a novel Bayesian Ying-Yang approach to fuzzification which is used to obtain the optimal fuzzy sets.

Chapter 4

FCMAC Using Bayesian Ying Yang Learning

From the previous chapters, we see that self-organizing fuzzification can determine the cluster number, their centers and width automatically. However, most of the existing self-organizing fuzzification techniques cannot provide the optimal fuzzy sets. In this chapter, a novel approach will be employed to obtain the optimal fuzzy sets.

4.1 BYY Learning Approach to Fuzzification

Fuzzification in a fuzzy neural network is actually equivalent to finding a cluster solution, c, to represent the input data, x. Treating both x and c as random processes, the joint distribution $p(x,c)$ can be calculated by either of these two formulae:

$$p(x,c) = p(c \mid x)p(x), \tag{4.1}$$

$$p(x,c) = p(x \mid c)p(c), \tag{4.2}$$

However, the result of Equation (4.1) is not equal to that of Equation (4.2) unless c is the optimal solution. Notice that x and c are dialectical: First, x is visible but c is invisible. Second, x decides c in training but c decides x in running. This interesting phenomenon

complies with the famous ancient Chinese Ying-Yang philosophy: First, everything in the universe can be viewed as a product of a constant conflict between opposites – Ying and Yang, with Ying referring to negative, female and invisible, whereas Yang referring to positive, male and visible. Second, the optimal status is reached if Ying and Yang achieve harmony.

In this section, the fuzzifier using Bayesian Ying-Yang [16, 17] is employed to automatically form the fuzzy set for each dimension. As shown in Figure 4-1, the BYY fuzzification system considers two complement representation of the joint distribution of input pattern x and fuzzy cluster *c*.

The first one is the forward/training model $Mc = p(c) = \int p(c \mid x) p(x) dx$. Mc is called a Yang/(visible) model which focuses on the mapping function of the visible input data x into an invisible cluster representation c via a forward propagation distribution $p(c|x)$. In this process, the input data x are visible and considered as known, whereas the clusters c are invisible and considered as unknown. By this model, the given input data are transferred into unknown fuzzy clusters. The process is regarded as an unsupervised learning process.

The second one is the backward/running model $Mx = p(x) = \int p(x \mid c) p(c) dc$. Mx is called Ying/(invisible) model which focuses on the generation function of the invisible input data x from a visible cluster representation c via a backward propagation

distribution $p(x|c)$. In this process, clusters c are visible and considered as known, whereas the input data x are invisible and considered as unknown. By this model, the input data are generated from the constructed fuzzy clusters. The process is regarded as a supervised learning process.

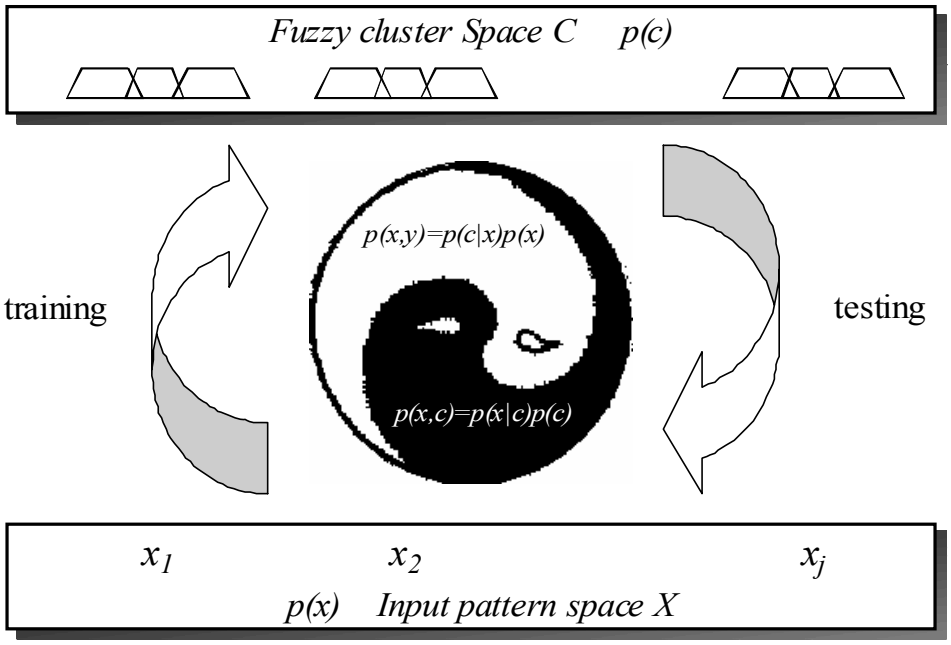


Figure 4-1 Fuzzification VS Bayesian Ying-Yang harmony

Under the Ying-Yang harmony principle, the difference between the two Bayesian representations in (4.1) and (4.2) should be minimized. Thus, the trade-off between the forward/training *Mc* model and the backward/running *Mx* model is optimized. It means that the input data are well mapped into the clusters and at the same time the clusters also well cover the input data. Therefore, the entire BYY fuzzification system should be the least complexity. In other words, the proposed FCMAC-BYY in this section has the

highest generalization ability when the harmony between the Ying and Yang model is achieved.

4.2 Gaussian Membership Function

In this research, Gaussian membership function is used, and the joint probability density of the *j*th dimension that consists of K^j Gaussians is:

$$p(x_i^j, \Theta^j) = \sum_{c=1}^{K^j} \alpha_y^j G(x_i^j, m_c^j, \sigma_c^j) \tag{4.3}$$

and the Gaussian density function:

$$G(x_i^j, m_c^j, \sigma_c^j) = \exp\left[-\frac{1}{2}\left(\frac{x_i^j - m_c^j}{\sigma_c^j} \right)^2 \right] \tag{4.4}$$

where x^j is the input data of the *j*th dimension, $\Theta^j \equiv \left\{ \theta_c^j \equiv \alpha_c^j, m_c^j, \sigma_c^j \right\}_{c=1}^{K^j}$ is a set of finite mixture model parameter, α_c^j is the prior probability, m_c^j and σ_c^j refer to the mean value and the width of the *c*th cluster of the *j*th dimension. These parameters can be estimated by maximum likelihood (ML) learning with the EM algorithm [54, 55] based on a given data set $D = \{X_i\}_{i=1}^N$.

4.3 Fuzzification with Bayesian Ying-Yang learning

Fuzzification using BYY involves two phases, namely, parameter learning and cluster number selection [56]. Parameter learning does the task of determining all the unknown parameter Θ^j for a specific number of clusters K^j. Then, cluster number selection is made to select the optimal number of clusters K^j* from a collection of specific BYY systems with different value of number of clusters K^j.

4.3.1 Parameter Learning

In the first phase, parameter learning, the Kullback-Leibler divergence [57], more details can be referred in Appendix C, is used to evaluate the difference of the joint probability between Ying and Yang:

$$\min_{\Theta} KL_{Mc,Mx}(\Theta) = \int\int P(c \mid x)\, P(x) \ln \frac{P(c \mid x) P(x)}{P(x \mid c) P(c)} dx\, dc \tag{4.5}$$

The minimization of the above Kullback-Leibler divergence will produce the optimal parameter Θ^{j*} at each value of cluster K^j. The above difference can be minimized by iterative execution of the following two steps:

Step1: Fix $M_c = M_c^{old}$, get $M_x^{new} = \arg\min_{M_x} KL_{M_x,M_y}$

Step2: Fix $M_x = M_x^{old}$, get $M_c^{new} = \arg\min_{M_c} KL_{M_x,M_c}$

which lead KL_{M_x,M_c} to converge. In the case of free $P(c\,|\,x)$, the probabilities $p\ (x\,|\,c)$, $p\ (c\,|\,x)$, and $p\ (c)$ can be calculated as follows:

$$\begin{aligned} p\ (x\,|\,c) &= P(x\,|\,\theta_c) \\ p\ (c\,|\,x) &= \sum_{j=1}^{k} P(c_j\,|\,x)\delta(c-j) \\ p\ (c) &= \sum_{j=1}^{k} \alpha_j (c-j) \end{aligned} \tag{4.6}$$

subject to

$$\begin{aligned} &P(c_j\,|\,x) > 0, && \sum_{j=1}^{k} P(c_j\,|\,x) = 1; \\ &\alpha_j > 0 && \sum_{j-1}^{k} \alpha_j = 1; \end{aligned} \tag{4.7}$$

where θ_c stands for all the unknown parameters of cluster c (in this case, they are the mean and covariance), k is the number of fuzzy clusters and α_j is the prior probability of the jth cluster. When the number of patterns N is large enough the minimization of KL_{M_x,M_c} given by Equation (4.5) is equivalent to $\min_{\Theta_k} KL_{Mx,Mc}(\Theta_k)$ with $\Theta_k = \{\theta_j\}_{j=1}^{k}$ as follows [16, 17]:

$$KL_{Mx,Mc}(\Theta_k) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} P(c_j\,|\,x_i)\ln P(c_j\,|\,x_i) - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} P(c_j\,|\,x_i)P(x_i\,|\,\theta_j) - \sum_{j=1}^{k}\alpha_j \ln \alpha_j\,. \tag{4.8}$$

Minimization Equation (4.8) is equivalent to $\max_{\Theta_k} L(\Theta_k)$ with

$$\begin{aligned} &L(\Theta_k) = \frac{1}{N}\sum_{i=1}^{N} p(x_i,\Theta_k), \\ \text{where}\qquad &p(x_i,\Theta_k) = \sum_{j=1}^{k}\alpha_j p(x_i\,|\,\theta_j) \end{aligned} \tag{4.9}$$

Therefore, the minimization of Equation (4.8) can be implemented by following two steps of the Expectation-Maximization (EM) algorithm [54, 55]:

E-step: Calculate the posterior probability $P(c^j \mid x_i^j)$

$$P(c^j \mid x_i^j) = \frac{\alpha_c G(x_i^j, m_c^j, \sigma_c^j)}{\sum_{c=1}^{K^j} \alpha_c^j G(x_i^j, m_c^j, \sigma_c^j)} \tag{4.10}$$

for $c^j = 1,...,K^j$ and $i = 1,...,n$

M-step: Update parameters by

$$\alpha_c^{j(new)} = \frac{1}{N}\sum_{i=1}^{N} \frac{\alpha_c^j G(x_i^j, m_c^j, \sigma_c^j)}{\sum_{c=1}^{K^j} \alpha_c^j G(x_i^j, m_c^j, \sigma_c^j)} = \frac{1}{N}\sum_{i=1}^{N} P(c^j \mid x_i^j) \tag{4.11}$$

$$m_c^{j(new)} = \frac{\sum_{i=1}^{N} P(c^j \mid x_i^j) x_i^j}{\sum_{i=1}^{N} P(c^j \mid x_i^j)} = \frac{1}{\alpha_c^j N}\sum_{i=1}^{N} P(c^j \mid x_i^j)\, x_i^j \tag{4.12}$$

$$\sigma_c^{j(new)} = \frac{1}{\alpha_c^j N}\sum_{i=1}^{N} P(c^j \mid x_i^j)(x_i^j - m_c^j)^2 + \left(h^j\right)^2 \tag{4.13}$$

where h^j is the smoothing parameter, and be updated as follows:

$$h_{new}^j = h_{old}^j + \eta g\left(h_{old}^j\right), \tag{4.14}$$

where η is a step length constant and

$$g\left(h_{old}^{j}\right)=\frac{1}{h_{old}^{j}}-h_{old}^{j}\sum_{j=1}^{K^{j}}\frac{\alpha_{y}^{j}}{\sigma_{y}^{j}}-\frac{\sum_{i=1}^{N}\sum_{l=1}^{N}\gamma_{il}^{j}\left(x_{i}^{j}-x_{l}^{j}\right)^{2}}{\left(h_{old}^{j}\right)^{3}} \tag{4.15}$$

with

$$\gamma_{il}^{j}=\frac{\exp\left[-\frac{1}{2}\left(\frac{x_{i}^{j}-x_{l}^{j}}{h_{old}^{j}}\right)^{2}\right]}{\sum_{i=1}^{N}\sum_{l=1}^{N}\exp\left[-\frac{1}{2}\left(\frac{x_{i}^{j}-x_{l}^{j}}{h_{old}^{j}}\right)^{2}\right]} \tag{4.16}$$

The smoothing parameter h^{j} in Equation (4.14) provides an improvement on the EM algorithm to prevent the width of clusters from becoming too small, which would cause over-fitting. This usually occurs when the size of sample data is small. The role of the smoothing parameter h^{j} is reduced as the number of samples increases.

4.3.2 Cluster Number Selection

In the second phase, cluster number selection, the optimal number of fuzzy cluster K^{j*} of each dimension is determined by cluster number selection [16] as follows:

$$K^{j}=\arg\min_{K^{j}}J(K^{j}) \tag{4.17}$$

$$J(K^{j})=\sum_{c=1}^{K^{j}}\left(\frac{1}{2}\alpha_{c}^{j*}\ln\sigma_{c}^{j*}+\frac{1}{2}\frac{\left(h^{j}\right)^{2}}{\left(\sigma_{c}^{j*}\right)^{2}}-\alpha_{c}^{j*}\ln\alpha_{c}^{j*}\right) \tag{4.18}$$

where σ_c^{j*}, α_c^{j*} and h^j are the results of the parameter learning in the first phase.

The above equation is used for detecting the optimal number of fuzzy cluster K^{j*} as the minimum point of $J(K^j)$, which is reduces as K^j increases and reaches its minimum at K^{j*} and then increases as K^j continues to increases. And the superscript "*" means that this component is obtained from the parameter learning in the first phase by Equation (4.11) and Equation (4.13). This step is used to select the best K^{j*} that minimize both the mismatching of the Ying-Yang presentation pair.

In practical, for each dimension j, we start with $K^j=1$, estimate the parameter Θ^j by the EM algorithm based on the given training data, and compute $J(K^j)$. Then, we proceed to $K^j \rightarrow K^j+1$, and compute $J(K^j)$ again. After we gather a series of $J(K^j)$, the optimal cluster number, K^j, is selected from the one with minimal $J(K^j)$.

4.4 Experimental Results

We are now in a position to compare the performance of FCMAC-BYY with other representative fuzzy neural networks like Falcon-ART, Falcon-MART and GenSoFNN-CRI(S). Our experiments were conducted on three benchmark data sets: Iris data, and bank failure classification. The hardware configuration for our experiments is: CPU = Intel Pentium IV 2.2GHz, operating system = Microsoft Window 2000, memory available = 512 Mbytes.

4.4.1 Iris Classification

The Fisher's Iris data set [58] consists of 150 instances of Iris flowers belonging to three classes; namely: Sentosa (class 1), Virginica (class 2) and Versicolor (class 3). There are 50 instances for each of the three classes. Each instance of the flowers consists of four physical attributes. They are sepal length, sepal width, petal length, and petal width. Figure 4-2 shows the data distribution of these four attribute dimensions in Iris data, and we can see that they are quite different from dimension to dimension.

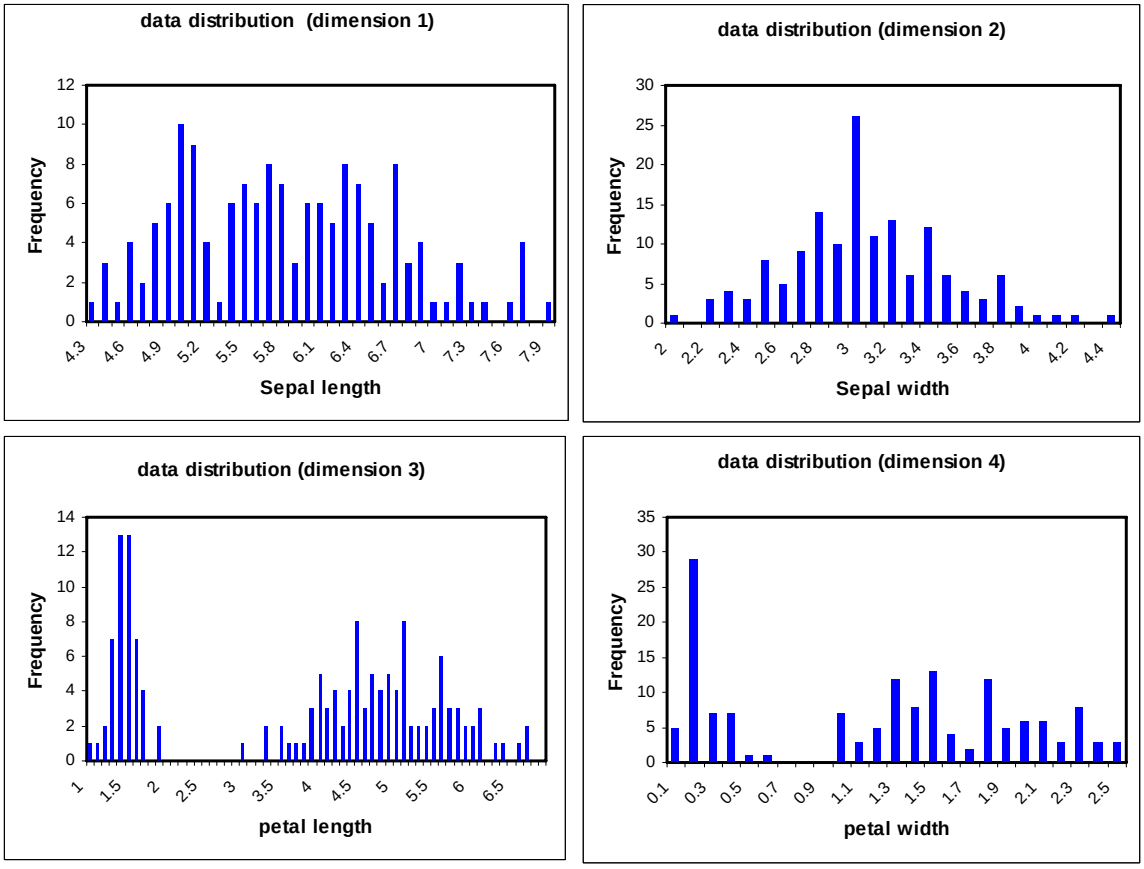


Figure 4-2 Data distribution of the four dimensions in Iris data

The training set consists of 35% of the data, and there are 17 instances from each of the three classes. The test set contains the remaining 65% of the data points and consists of 99 data points. The training and testing sets are randomly generated. The experiment results are cross validated by using three different groups of training and test sets. They are denoted as CV1, CV2, and CV3. The desired outputs of the network are 0, 1 and 2 for classes 1, 2 and 3 respectively.

Experimental results are recorded in term of two classes: *True Acceptance* and *False*. The *True Acceptance (TA)* refers to the total number of correctly classified instances, while the *False* refers to all misclassified instances. The *False class* is subdivided into *False Acceptance* class1 (FA_1), *False Acceptance* class2 (FA_2), *False Acceptance* class3 (FA_3), i.e., wrong instances classified into class 1, class 2, class 3, respectively, and *False Rejection* (FR), i.e., unclassified instances. The formulas are given as follows:

$$TA = \frac{N_{ins\_corr}}{N_{ins\_total}} \times 100\%$$

$$FA_1 = \frac{N_{class1\_wrong}}{N_{mis\_ins}} \times 100\% \qquad FA_2 = \frac{N_{class2\_wrong}}{N_{mis\_ins}} \times 100\%$$

$$FA_3 = \frac{N_{class3\_wrong}}{N_{mis\_ins}} \times 100\% \qquad FR = \frac{N_{un\_class}}{N_{mis\_ins}} \times 100\%$$

Here, N_{ins_corr} refers to the number of correct instances classified,

N_{ins_total} refers to the total instances,

N_{class1_wrong}, N_{class2_wrong}, N_{class3_wrong}, N_{un_class} refer to the wrong instances classified into class 1, class 2, class 3, and unclassified instances respectively,

N_{mis_ins} refers to the number of wrongly instances classified.

From Table 4-1, one can observe that the *False Acceptance* class1 is almost zero, while the *False Acceptance* class2 and *False Acceptance* class3 have high values. It means that most of the wrong instances are classified into classes 2 and 3. This is consistent with the claim made by some authors that while class 1 is well separated from the other two, classes 2 and 3 have significant degree of overlap [59].

Table 4-1 Experimental results of FCMAC-BYY on Iris classification

Experiment	**(TA) %**	**FA_1 %**	**FA_2%**	**FA_3%**	**FR%**
CV1	96.67	0	44.44	11.11	44.44
CV2	96.67	0	57.14	42.85	0
CV3	97.33	0	44.44	44.44	11.11

In Table 4-2, average classification rate is used to evaluate the performance of the FCMAC-BYY across the three data groups. The experimental results of the proposed FCMAC-BYY network for Iris classification are compared against FCMAC using DIC [12]. Both models use Gaussian distribution as the membership functions.

Table 4-2 Comparison of number of clusters obtained by BYY and DIC

	FCMAC-BYY		**FCMAC using DIC**	
	Classification rate%	**Clusters numbers of four dimensions**	**Classification rate%**	**Clusters numbers of four dimensions**
CV1	97.97	$2\times1\times2\times3=12$	96.6	$3\times3\times6\times4=216$
CV2	95.96	$2\times1\times2\times3=12$	96	$3\times4\times6\times3=216$
CV3	95.96	$2\times1\times2\times3=12$	95	$2\times2\times7\times3=84$

Compared against the results of the FCMAC using DIC, the FCMAC-BYY achieves higher accuracy with significantly fewer fuzzy rules or neurons. Table 4-2 indicates that the self-organizing BYY requires less fuzzy clusters than DIC. For example, for CV1, the self-organizing BYY derives 2:1:2:3 fuzzy clusters for the four dimensional input data respectively, while DIC derives 3:3:6:4 fuzzy clusters respectively. In the third dimension, DIC requires 6 or 7 fuzzy clusters, whereas 2 clusters identified by the self-organizing BYY are sufficient. In other words, the self-organizing BYY significantly simplifies the network with higher generalization ability by judiciously optimizing fuzzy rule set by the Ying-Yang approach.

The above results can be explained as follows. Like most of the clustering techniques, DIC is very sensitive to the initial values. DIC simply applies the same values to initialize new clusters for all dimensions; however the data distributions are different from dimension to dimension as shown in Figure 4-2. This drawback of DIC leads to more clusters. BYY fuzzification determines the optimal centroids and widths for each particular number of clusters, and the optimal cluster number for each dimension is then

computed by Equation (4.17). However, BYY fuzzification demands a higher computation cost for such an optimization.

Furthermore, BYY needs more computation time for each epoch than other methods due to its optimization. From Table 4-3, we can see that the average execution time of FCMAC-BYY is higher than FCMAC-DIC; however it is much smaller than that of Falcon-ART. Hence, the computational cost of BYY is still acceptable.

Further comparison of the proposed FCMAC-BYY with other architectures is listed in Table 4-3. These architectures include variations of Falcon network, namely Falcon-ART [60], Falcon-MART [61] and GenSoFNN-CRI(S) [12]. The average classification rate for FCMAC-BYY is 96.60%, which is the highest amongst different architectures. In addition, FCMAC-BYY has a small number of epochs and a less complexity structure due to the optimal clusters obtained by BYY classification.

Table 4-3 Comparison of training epochs, average classification rate and CPU timings of various networks on Iris dataset

Networks	**Training epochs**	**Average classification rate (%)**	**CPU time (s)**
Falcon-ART	1000	75.76	98.75 ± 8.57
Falcon-MART	11	94.95	1.46 ± 0.39
FCMAC-DIC	20	95.87	0.32 ± 0.08
GenSoFNN-CRI(S)	25	96.03	3.91 ± 1.24
FCMAC-BYY	**10**	**96.60**	**6.72 ± 1.36**

4.4.2 Bank Solvency Analysis

Bank failure prediction is an important issue for the regulators of the banking industries. The collapse and failure of a bank could trigger an unfavorable financial impact and generate negative impacts such as a massive bail out cost for the failing bank and loss of confidence from the investors and depositors. Very often, bank failures are due to financial distress. Hence, it is desirable to have an early warning system (EWS) that identifies potential bank failure or high-risk banks through the traits of financial distress. The financial variables (covariates) used in the bank failure prediction application are extracted from the Call Reports, which are downloaded from the website of Federal Reserve Bank, Chicago [62, 63]. The observation period of the survived banks consists of 21 years from January 1980 to December 2000. There are 702 failed banks and 2933 survived banks over the observation period, leading to a total of 3635 banks. Based on statistical investigation, only nine variables shown in Appendix A are selected according to their significance and correlation.

Three different scenarios of experiments are conducted:

- ❖ Bank failure classification based on the last available financial record.
- ❖ Bank failure prediction using financial records one year prior to the last one.
- ❖ Bank failure prediction using financial records two years prior to the last one.

In each scenario, the data set is split into different training sets and test sets. The training sets contain 20% of the data while the test sets contain the remaining 80%. There are five

cross-validation groups, denoted as CV1, CV2, CV3, CV4 and CV5 respectively. The original data set is initially split into two pools: failed and survived banks. For each cross-validation group, the training set is randomly selected from two pools so that the number of survived and failed bank is equal. It is called a "balance" training scenario. The training sets of the five groups are mutually exclusive.

One output is used to differentiate between failed and survived banks. Failed banks are denoted with output "0" while survived (non-failing) banks are identified by output "1". The Ying-Yang FCMAC network is subsequently used to model the inherent relationships between the financial covariates and their impact on the financial solvency of the respective banks. The Ying-Yang FCMAC network is trained using the data instances in the training set and the modeling capability of the trained network is subsequently evaluated using the test set. The simulation is repeated for all the five cross-validation groups. The classification threshold (to discern between failed and survived (non-failing) banks based on the nine input financial covariates) is varied to obtain the receiver-operating-characteristic (ROC) curves shown in Figure 4-3. Type I error is defined as the error of classifying a failed bank as a survived (non-failing) one whereas Type II error is the classification of a non-failing bank as a failed bank. The EER line denotes the case of equal error rates (EER), where the Type I equals Type II errors.

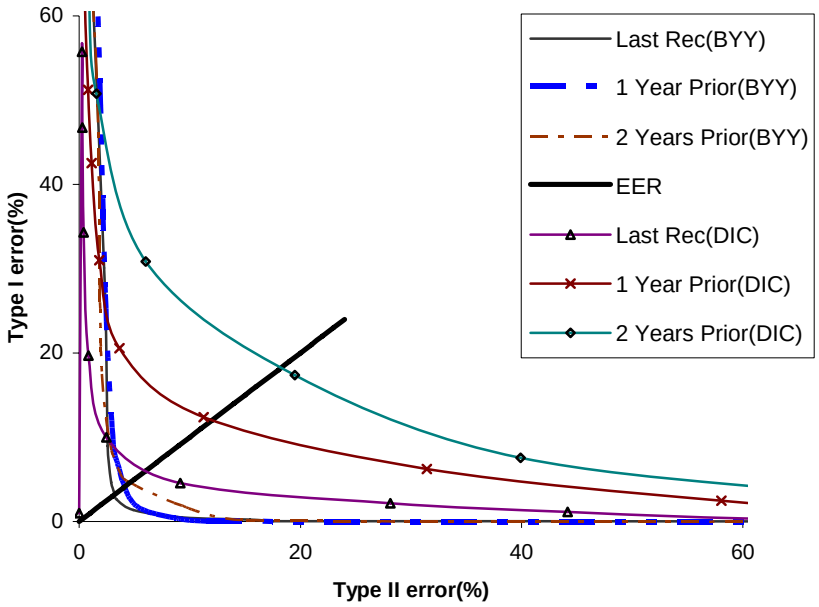


Figure 4-3 ROC curves of bank failure classification results on three scenarios

From Figure 4-3, one can observe that the Ying-Yang FCMAC outperforms to FCMAC-DIC on all three scenarios. The average classification rate for the failed banks of the FCMAC-DIC using the last financial statements is about 90% and it goes down to around 85% with statements obtained one year prior to the last record and subsequently to about 81% with financial statements two years prior to the last available record. Whereas, the values of Ying-Yang FCMAC only deteriorate from 94% in the first scenario to around 92% in the second scenario and subsequently to about 91% in the last scenario.

Table 4-4 shows the comparison on bank failure classification of GenSoFNN-CRI(S) [63], FCMAC-DIC and FCMAC-BYY. One can observe that the FCMAC-BYY has superior performance to both GenSoFNN-CRI(S) and FCMAC-DIC. Noticing that, the classification rate degrades with respect to the prediction period. The longer the

prediction period, the less accurate are the classification and prediction result. However, FCMAC-BYY is less sensitive to the prediction period than the others. Its average classification rate decreases from 94.53% of the last record to 91.71% of the 2 years prior. Whereas, the average classification rate of GenSoFNN-CRI(S) drops from 90.86% of the last record to 72.19% of the 2 years prior.

Table 4-4 Comparison on bank failure classification with 9 inputs of GenSoFNN-CRI(S) (CRI), FCMAC-DIC(DIC) and FCMAC-BYY(BYY)

	Last record			**1 year prior**			**2 years prior**		
	CRI (%)	**DIC (%)**	**BYY (%)**	**CRI (%)**	**DIC (%)**	**BYY (%)**	**CRI (%)**	**DIC (%)**	**BYY (%)**
CV1	92.92	90.61	**93.95**	82.26	90.08	**93.23**	68.29	82.34	**91.73**
CV2	90.62	92.27	**94.96**	81.13	83.22	**92.77**	68.29	82.18	**91.81**
CV3	94.37	91.63	**94.48**	87.55	89.10	**92.64**	81.95	83.93	**91.19**
CV4	86.04	89.80	**94.64**	76.6	86.38	**93.53**	74.15	79.77	**92.06**
CV5	95.21	90.01	**94.60**	86.79	85.77	**92.77**	68.29	77.92	**92.14**
Average Classification rate (%)	91.83	90.86	**94.53**	82.87	86.91	**92.99**	72.19	81.23	**91.71**

4.5 Summary

In this chapter, we introduced a novel Bayesian Ying-Yang approach to fuzzification to propose the FCMAC-BYY. The experiments are conducted on two benchmark datasets and we compared the classification performance of our proposed model with those of representative neural fuzzy systems such as FALCON-ART, Falcon-MART, and GENSOFNN-CRI(S). The experimental results suggest that our proposed model is able to achieve the same accuracy but has simpler structure with higher generalization capability.

The fuzzifier using BYY learning is employed to find the optimal centroids and widths for each particular number of clusters during the parameter learning phase. After that, the optimal number of clusters for each dimension is then determined by the cluster number selection phase. Moreover, the smoothing technique prevents the width of clusters from being singular which usually occurs in the EM algorithm on a small size of samples. However, BYY fuzzification pays the price of computation cost for such an optimization.

As mentioned in Section 1.2, the Bayesian Ying-Yang fuzzifier used for optimal fuzzy sets is separated from the learning part of the neural network. Moreover, the weights of neurons in the FCMAC-BYY are trained by gradient-based methods. These two shortcomings may end up being locally optimized [19]. In the next chapter, cooperative coevolution computation is incorporated into the FCMAC-BYY system to find the optimal structure and connection weights to propose the cooperative coevolutionary FCMAC-BYY (FCMAC-EBYY).

Chapter 5

FCMAC using Coevolutionary-BYY

In Chapter 4, we introduced a novel Bayesian Ying-Yang approach to fuzzification. The fuzzifier using BYY learning is employed to find the optimal centroids and widths for each particular number of clusters during the parameter learning phase. However, the Bayesian Ying Yang Learning applied to FCMAC suffers from two disadvantages: Firstly, the Bayesian Ying-Yang learning used to optimal fuzzy sets is separated from the learning part of the neural network; it may end up being local optimization [64]. Secondly, the weights of neurons in the FCMAC-BYY are trained by using gradient-based methods, which may fall into a local minimum during the learning process [19]. Cooperative coevolution computation is hence introduced to engage in this issue. Coevolution computation used here utilizes global search and optimization techniques to explore the search domains of both fuzzy clusters and neuron weighting of the proposed model.

In this chapter, we integrate cooperative coevolution computation into FCMAC-BYY to propose the coevolutionary FCMAC-BYY (FCMAC-EBYY) network. In the next section, the basic concepts behind coevolutionary learning as well as cooperative coevolution computation are introduced. The integration of cooperative coevolution computation into the FCMAC-BYY network is then proposed.

5.1 Coevolutionary Learning

Evolution computation (EC) has been widely used for generating fuzzy rules and tuning membership functions [65] and successfully applied to many applications [35-40]. EC uses global search and optimization techniques to explore the search space. In doing so, it maintains and evolves a population of trial solutions/chromosomes which consists of possible cluster parameters combinations as solutions. The evolution of the chromosomes is carried out in the iteration of the evolutionary algorithm, which is called a generation. Each generation generally consists of six processes which are namely initialization, evaluation, selection, reproduction (crossover and mutation) and re-insertion.

However, real natural world progresses with not single genetic algorithm but coevolutionary algorithm, which allows simultaneous evolution of two or more species with coupled fitness. Such coupled evolution favors the discovery of complex solutions whenever complex solutions are required [43].

Typically, the coevolving species can be classified into two types: competitive coevolutionary algorithm which obtains exclusivity on a limited resource and cooperative coevolutionary algorithm which gains access to some hard-to-attain resource. In a competitive coevolutionary algorithm the fitness of an individual is based on direct competition with individuals of other species, which in turn evolve separately in their own populations. Increased fitness of one of the species implies a diminution in the fitness of the other species. This evolutionary pressure tends to produce new strategies in

the populations involved so as to maintain their chances of survival. This objective ideally increases the capabilities of each species until they reach an optimum.

On the other hand, cooperative coevolutionary algorithms, also called symbiotic coevolutionary algorithms, involve a number of independently evolving species which together form complex structures, well suited to solve a complex problem. The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favors the development of cooperative strategies and individuals.

The problem of finding a suitable architecture and corresponding weights of the fuzzy neural network is a very complex task [49]. A fuzzy neural network process usually deals with the simultaneous search for the fuzzy cluster structures and connective parameters or the rule's weights. These parameters provide a complete definition of the linguistic knowledge describing the behavior of the fuzzy neural network system and the values mapping this symbolic description into a real-valued world. Thus, construction a fuzzy neural network model can be viewed as two separate but intertwined search processes: the search for the optimal fuzzy membership functions and the search for the corresponding weights.

Therefore, with the above discussions, the fuzzy neural network modeling is an ideal application of cooperative coevolution computation. In the next section, the integration of

cooperative coevolution computation into FCMAC-BYY to propose the cooperative coevolutionary FCMAC-BYY (FCMAC-EBYY) network is presented in details.

5.2 Cooperative Coevolutionary FCMAC-BYY

This section proposes the incorporation of cooperative coevolution computation into FCMAC-BYY system. The proposed FCMAC-EBYY has two coevolving, cooperating species defined for fuzzy clusters configuration and neuron weighting. The block diagram of the system is shown as follows:

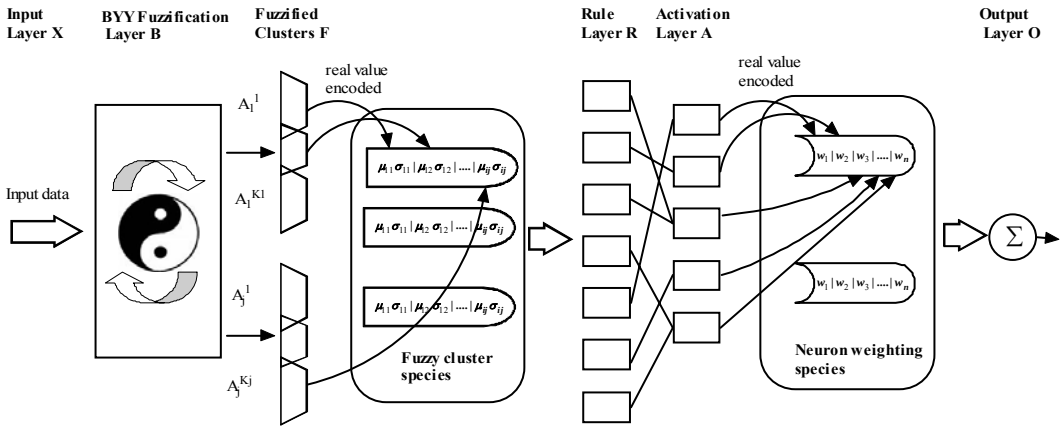


Figure 5-1 Incorporation of cooperative coevolution computation into FCMAC-BYY network

It can be seen from Figure 5-1 that cooperative coevolution computation is integrated into the optimization of fuzzy clusters and neuron weighting of the FCMAC-BYY network. The input data is first fuzzified using BYY fuzzification proposed in Chapter 4. The resultant number of clusters and parameters are then used as the initial population of the fuzzy cluster species, whereas a series of randomized values are used as the initial population of the neuron weighting species. After that, the adjusted parameters are set as

the optimized fuzzy sets. The detailed processes are described in the following paragraphs.

5.2.1 Initialization

The initialization process involves the reading the output from the BYY fuzzification, which obtains the optimal number of clusters and clusters parameters that represent the input patterns. These parameters, namely the mean values (μ) and the width (σ), are then stored in chromosome array as the population of trial solutions of the fuzzy cluster species. After that, a random process is used to initialize the neuron weighting species corresponding to weights of rules. Each combined chromosome represents a combination of clusters and weights parameters as a possible solution of the entire FCMAC-EBYY system.

An example of the chromosome representations is shown in Figure 5-2. μ_{ij} and σ_{ij} represent the mean values and widths of the i^{th} dimension in the j^{th} cluster of fuzzy cluster species respectively, and $w_1, w_2, ..., w_n$ represents the weights of the corresponding rules in the neuron weighting species.

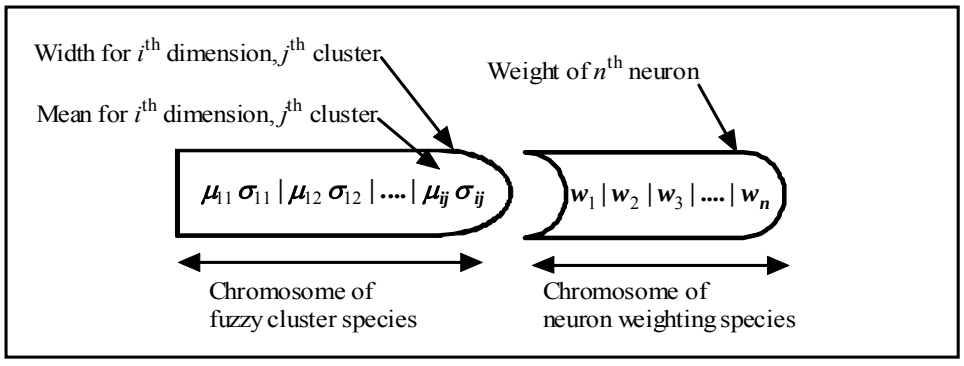


Figure 5-2 Chromosome representation of clusters and neurons

In the experiments conducted in this chapter, two populations of chromosomes are maintained for computation. To produce the initial populations, the following computations are carried out.

1) The BYY fuzzification is first performed to obtain the optimal number of clusters for each dimension from the raw training data. These cluster parameters are then selected and stored in the first positions of the fuzzy cluster species. The rest of the chromosomes are randomly initialized within the range of variation of the training data and stored in the chromosome array to complete the population.

2) In the neuron weighting species, a number of weights corresponding to the number of neurons in the Rule Layer R are randomly initialized to create the chromosomes in the population.

These steps above ensure that the initial populations are a good mixture of the optimized outputs from BYY fuzzification and random processes. In doing so, the network is encouraged to explore the entire region of the solution domain faster, thus improving the quality of the clusters parameters and the learning speed.

5.2.2 Evaluation

The two evaluation methodologies used to operate the evolution of the two populations are instances of a simple genetic algorithm presented in [66]. The purpose of the evaluation process is to determine how well a trial solution would perform when it is given a set of test data and their respective results. Using the training data as a gauge, we

could find the better fitting solutions, in return, boast a high possibility of gaining similar results when they are tested with the test data. The detailed view of the fitness evaluation process is shown in Figure 5-3.

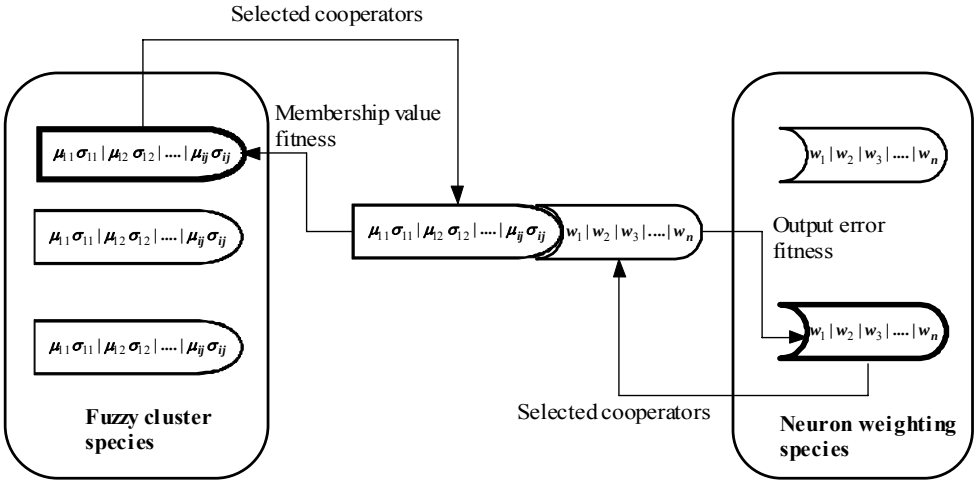


Figure 5-3 Fitness evaluation of the two species in coevolutionary FCMAC-EBYY

In the fuzzy cluster species, the *membership value fitness function* is used to provide a measure of how the chromosomes have performed in the problem domain to ensure that the training data have the highest membership values to the fuzzy clusters. In the other words, the evaluation of the fuzzy cluster species will obtain the fittest fuzzy labels to the training data. On the other hand, *the out error fitness function* is used in the neuron weighting species to train the weight of the neuron in the rule layer. The deviation between the calculated output and the actual output used as the objective function in the evaluation will lead the chromosomes to the global optimization instead of local minimization of gradient-based training of the tradition FCMAC.

As shown in Figure 5-3, an individual undergoing fitness evaluation establishes cooperation with one or more representatives of the other species to construct fuzzy-neural system. In the proposed cooperative coevolutionary FCMAC-EBYY, a selected individual from the fuzzy cluster species will cooperate with the selected individuals from the neuron weighting species to form a full representation of the structure. The details of the selection, crossover, mutation and reinsertion are presented in the following sections.

5.2.3 Selection

Selection is the process of selecting a trial solution for reproduction (crossover and mutation process). Based on their fitness values computed in the evaluation process, the trial solutions are selected to produce offsprings in the reproduction process. Stochastic Universal Sampling (SUS) [67] was employed for this selection. SUS is a single-phase sampling algorithm with minimum spread and zero bias. With this method, the individuals are mapped one-to-one into contiguous segments of a line, where the size of the segment of each individual relates to its fitness value. Then, equally spaced pointers are placed along the line. The number of pointers on the line would correspond to the number of individuals to be selected. A simple illustration of SUS is shown as follows.

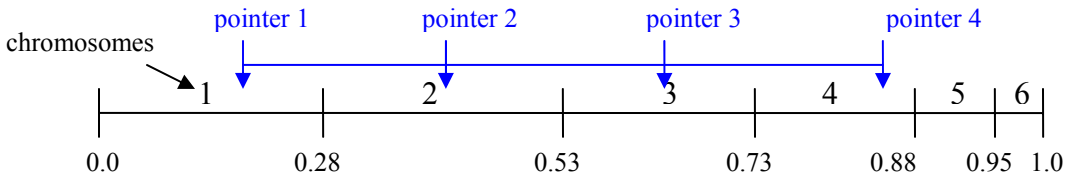


Figure 5-4 SUS selection illustration

As shown in Figure 5-4 the six chromosomes are mapped to contiguous segments of a line, such that each chromosome's segment is equal in size to its fitness. The fitness values of all the chromosomes are normalized to fit into the range within 0 and 1 in this case. Four equally spaced pointers are placed over the line to point at the four selected chromosomes. For this illustration, chromosomes 1, 2, 3 and 4 are selected in the algorithm.

5.2.4 Crossover

The reproduction process consists of the crossover and the mutation processes. In the crossover process, new individuals are reproduced such that they possess parts of both parent's genetic characteristics, just like its counterpart in nature. This process ensures that the offspring will retain the qualities of its parents. Intermediate recombination [68] is utilized here to produce new offspring around and between the values of the parent chromosomes. The crossover operation is illustrated as follows:

$$\begin{aligned} O_1 &= P_1 \times \alpha(P_2 - P_1) \\ O_2 &= P_2 \times \alpha(P_2 - P_1) \end{aligned} \tag{5.1}$$

where α is a scaling factor chosen uniformly at random over the range of [-0.25, 1.25] while P_1 and P_2 are the parent chromosomes. Each variable in the offspring is the result of combining the variables in the parents according to the above expression with a new α chosen for each pair of parent genes. This can be illustrated in Figure 5-5.

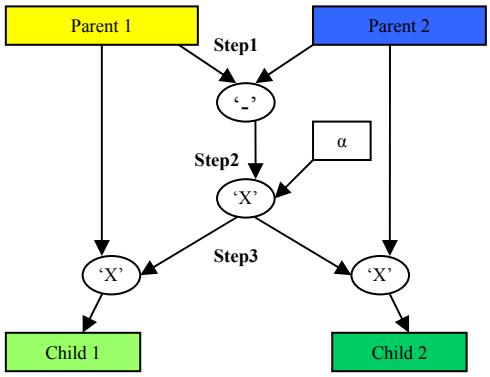


Figure 5-5 Intermediate recombination

Intermediate crossover is capable of producing new variables within a slightly larger hypercube than that defined by the parents but constrained by the range of α. A probabilistic parameter called the crossover rate is applied here to determine the frequency in which the operator is applied. The crossover rate ranges between zero to one, with zero indicating the operator is never applied, and one indicating that the operator is always applied. Then for each pair of parents, a random number is generated between the range of '0' and '1'. If this random number is greater than the crossover rate, the operator is not applied to that pair of parents and the children produced will be identical to the parents.

5.2.5 Mutation

The second process involved in the reproduction process is the mutation. Mutation can be accomplished by random selection of new values within an allowed range. For this

section, using the training data as a guide, the allowable ranges of real values in all the dimensions are computed. The new values, which are selected randomly from the allowable ranges for the respective dimension, would ensure that the mutated values are within reasonable regions. Similar to the crossover operator, a probabilistic parameter called the mutation rate determines the frequency of mutations. In all, mutation allows an increase in the level of possible exploration of the search space without adversely affecting the convergence characteristics.

5.2.6 Reinsertion

After the reproduction process is completed, a new population of chromosomes is produced. This new population will then be combined with the old population to form the resultant population. The *elitist-selection strategy* is employed to select a number of old chromosomes to be reinserted into the resultant population. This strategy chooses a pre-specified number of the fittest chromosomes in the old population to be inserted into the resultant population while the rest of the chromosomes are discarded. The discarded cluster parameters are replaced by chromosomes created from the reproduction process. This is done to retain the best quality chromosomes in the population. Once the resultant population has been produced, it would mark the completion of one generation.

5.2.7 Cooperative Coevolutionary FCMAC-BYY Algorithm

The cooperative coevolutionary FCMAC-EBYY can be presented in pseudo code as follows:

Table 5-1 Pseudo code of cooperative coevolutionary FCMAC-EBYY algorithm

begin FCMAC-EBYY	(1)
g:=0	(2)
for each species S	(3)
Initialize populations $P_S(0)$	(4)
Evaluate populations $P_S(0)$	(5)
end for	(6)
while not done **do**	(7)
for each species S	(8)
g:=g+1	(9)
$E_S(g)$ = elite-select $P_S(g\text{-}1)$	(10)
$P_S'(g)$ = Select $P_S(g\text{-}1)$	(11)
$P_S''(g)$ = crossover $P_S'(g)$	(12)
$P_S'''(g)$ = mutation $P_S''(g)$	(13)
$P_S(g) = P_S'''(g) + E_S(g)$	(14)
Evaluate populations $P_S(g)$	(15)
end for	(16)
end while	(17)
end FCMAC-EBYY	(18)

From Table 5-1, the processes are then repeated until a specified number of generations had been conducted or when a specified performance criterion is met. The fittest chromosome in the last population is then selected and its values will be placed into the cooperative coevolutionary FCMAC-EBYY system as the cluster parameters. The proposed structure will train and adjust the weights parameters to produce the final results. The finalized fuzzy sets together with the weight settings obtained by the BYY fuzzification together with the coevolution computation algorithm allow the FCMAC-EBYY archives a globe optimization solution.

5.3 Experiments and Analysis

The FCMAC-EBYY network is used to conduct experiments on two classification problems and the results are described as follows. Section 5.3.1 describes the cancer subtype classification problem as well as the results from FCMAC-EBYY. The discussions on the bank failure classification are given in Section 5.3.2.

5.3.1 Cancer Subtype Classification

In this experiment, we attempt to carry out the classification of cancer subtypes for acute lymphoblastic leukemia (ALL). The gene expression data has been utilized in the classification of the cancer subtypes. The major challenges in using these data from DNA micro arrays for cancer classification is the analysis and interpretation of the massive data output [69, 70] due to the large number of genes observed; and the extremely distorted ratio between the number of genes monitored and the actual number of tissue samples available. Therefore, feature selection [71] has to be performed prior to the development of a classifier as it has been well studied in the machine-learning community that no classifier is able to perform satisfactory when the number of features far-exceeded the number of training samples available.

The pediatric ALL cancer gene expression dataset used in this experiment is obtained using Affymetrix oligonucleotide micro arrays [72] with 12,600 probe sets. After pre-processing using a variation filter to remove genes with little or no activity in <1% of the samples, a final dataset of 12,558 genes is obtained. The dataset can be obtained online at

[73]. Due to large number of features involved, features selection is carried to prior presentation of the data into the network. In this experiment, a novel feature selection algorithm named Monte Carlo evaluative selection (MCES) [74] is employed to identify the relevant genes (features) that separate the various subclasses of ALL leukemia using the gene expression data. The MCES algorithm employed in this experiment requires a pre-filtering stage to refine the raw feature set of 12,558 genes prior to the identification of the relevant feature genes. In this experiment, a method based on T-statistics (T-Stats) analysis is adopted for the pre-filtering stage. The genes selected by this method is listed in the supplementary data of the pediatric ALL cancer dataset [73].

Within the set of the selected genes, there are a total of seven subclasses of leukaemia tissue samples. They are denoted as: Class 1, BCR-ABL; Class 2, E2A-PBX1; Class 3, Hyper > 50; Class 4, MLL; Class 5, T-ALL; Class 6, TEL-AML1; and Class 7, Others. The member genes within each class are deemed as promising genes with discerning properties that are able to identify samples of their respective target class from the rest of the data samples. MCES algorithm uses this set of genes to identify the relevant feature genes for the ALL cancer classification task. The respective sets of relevant genes for all the six cancer subtypes identified with MCES using the T-statistic is listed as Appendix B. The definitions and detailed differences of each subclass of ALL leukemia can be found at [73].

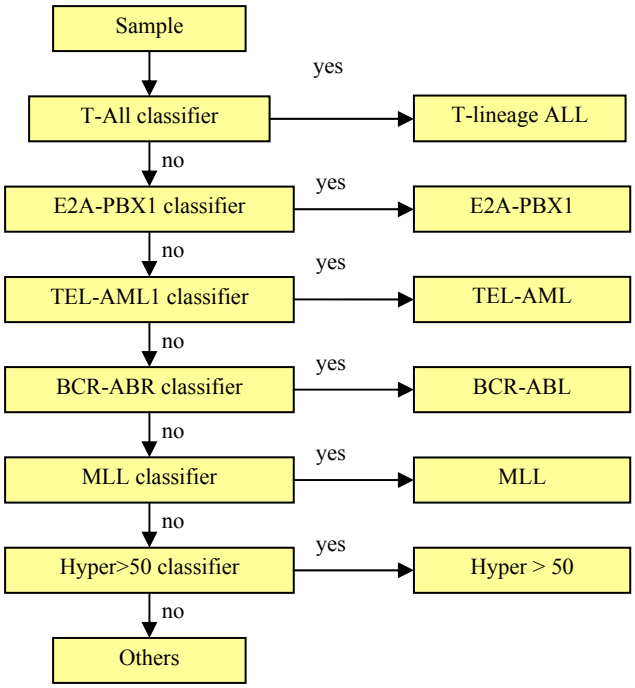


Figure 5-6 Diagnostic decision tree for classification of ALL cancer subtypes

The presentation order of the data samples in the training sets is then shuffled to create ten cross-validation groups (CV1-CV10). The diagnostic decision tree used here to perform classification of the ALL cancer subtype is shown in Figure 5-6. The assigned class is identified with an output '1' while the data samples belonging to the other classes in the decision tree are identified by an output of '0'.

All the classifiers are trained with the training sets and classification performances evaluated using the predefined testing sets. For the experiment, the following settings are used for the implementation.

Table 5-2 Parameters settings for Cancer Subtype Classification

Parameters	Settings
Number of clusters in population	30
Number of BYY clusters in initial population	5
Maximum number of generations	50
Crossover rate	0.94
Mutation rate	0.06
Number of chromosomes retained in reinsertion	4
Learning rate for FCMAC	0.5

The mean classification rates of the FCMAC-EBYY architecture across CV1-CV10 are compared against the classification performances of the various benchmarking systems such as Support Vector Machine (SVM), k-Nearest Neighbor (K-NN), Multi-Layer Perception (MLP), and GenSo-TVR [75] system in Table 5-3. In this table, listed in the squared brackets of the corresponding classes, the number of positive indicates the number of samples belonging to the assigned class, whereas the negative indicates the number of samples not belonging to the assigned class. Sensitivity (Sen) is defined as 100%-type I error while specificity (Spe) is 100%-type II error. Type I error is defined as the error of falsely rejecting a data sample as not belonging to the assigned class while Type II error is the error of falsely accepting a data sample as belonging to the assigned class. The average number of positive and negative samples misclassified by each of the classifiers across CV1-CV10 is provided (in brackets) next to these measures.

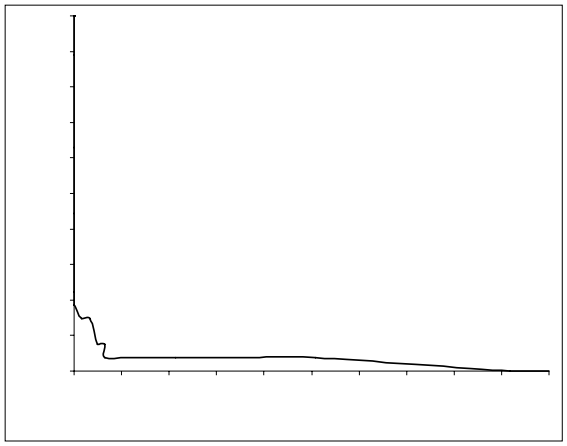
Table 5-3 Comparison on Classification results with T-statistics as pre-filtering

Classifier		**SVM**	**K-NN**	**MLP**	**GenSo-TVR**	**FCMAC -BYY**	**FCMAC -EBYY**
T-ALL	Sen (%)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)
[15+; 97]	Spe (%)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)
E2A-PBX1	Sen (%)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)
[9+; 88]	Spe (%)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)
TEL-AML1	Sen (%)	100 (0)	96(1.08)	100 (0)	90(2.7)	85(4.1)	86(3.8)
[27+; 61]	Spe (%)	97(1.83)	100 (0)	100 (0)	98(1.22)	98(1.1)	100 (0)
BCR-ABL	Sen (%)	33(4.02)	50(3)	83(1.02)	67(1.98)	84 (1)	84 (0.9)
[6+,55-	Spe (%)	100 (0)	100 (0)	98(1.1)	99(0.55)	93(3.7)	97(1.7)
MLL	Sen (%)	100 (0)	100 (0)	100 (0)	70(1.8)	83(1.3)	84(1.2)
[6+; 49]	Spe (%)	100 (0)	94(2.94)	100 (0)	99(0.49)	88(5.8)	96(1.7)
Hyper > 50	Sen (%)	100 (0)	95(1.1)	100 (0)	99(0.22)	100 (0)	100 (0)
[22+; 27]	Spe (%)	93(1.89)	93(1.89)	89(2.97)	85(4.05)	85(4.1)	89(2.9)

An analysis of Table 5-3 shows that the proposed FCMAC-EBYY classification system built using T-statistics as pre-filtering technique has comparable (and in some subtypes better) performances to the SVM, K-NN and MLP for the cancer subtype TALL, E2A-PBX1, BCR-ABL and Hyper>50 classes. Moreover, these three classification systems functioned as black boxes. There is no knowledge on how the computed decisions are derived. On the other hand, FCMAC-EBYY has the capability to solicit inherent knowledge from the numerical gene expression dataset and express the characteristics (of relevant genes) of each cancer subtypes in a form easily accessed and comprehended by the human users (i.e., the IF-THEN fuzzy rules).

The mean classification results of the classifiers for the TEL-AML1, BCR-ABL, MLL and Hyper>50 classes, across CV1 to CV10, with T-statistics as pre-filtering are summarized in the form of receiver-operating-characteristics (ROC) curves shown in Figure 5-7. It is noted that, the ROC curves of two class T-ALL and E2A-PBX1 are not shown because all the classifiers can separate these two classes with 100%. The EER line

denotes the case of equal error rate, where the Type I and Type II errors are the same. The ROC plots are obtained by varying the classification thresholds of the classifiers to evaluate the differentiating capabilities of the FCMAC-EBYY classification system. These results are compared to the results obtained in a fuzzy-neural system named GenSo-TVR system [75] as well as the FCMAC-BYY model.



It can be seen from the diagrams that all the classifiers were able to achieve significant results. On a closer look, the FCMAC-EBYY classifiers can classify the dataset more efficiently and have the lower amount of errors throughout. This affirms our hypothesis that FCMAC-EBYY is able to explore the solution domain to uncover the optimal solution. The observation also shows that FCMAC-EBYY proves to be the better results over the other classifiers.

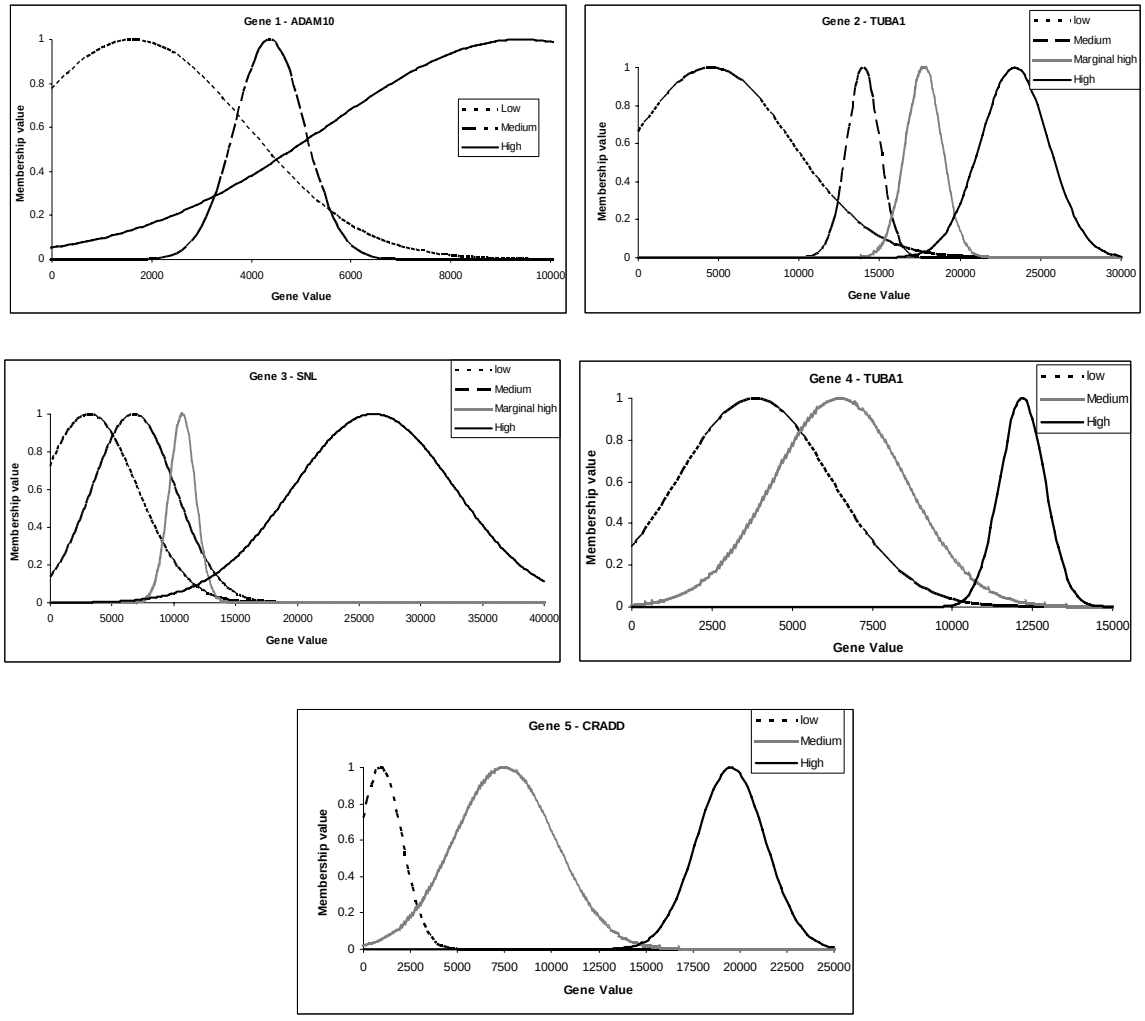


Figure 5-8 Fuzzy clusters extracted from the FCMAC-EBYY for the subclass BCR-ABL

As mentioned earlier, a main advantage of the FCMAC-EBYY model as compared to other techniques such as SVM and ANN is the ability to extract intuitive fuzzy rules from the trained structures. To demonstrate the comprehensiveness of the formulated fuzzy rules, the rule-base of the FCMAC-EBYY classifier for the cancer subtype BCR-ABL trained using the first cross-validation group (CV1) with T-statistics for pre-filtering is analyzed. The computed fuzzy sets for each of the five genes identified for BCR-ABL (see in Appendix B) are shown in Figure 5-8.

Table 5-4 Samples of fuzzy rules extracted from FCMAC-EBYY

Positive rules Rule 1: IF gene1 value is *low* and gene2 value is *marginal high* and gene3 value is *marginal high* and gene4 value is *medium* and gene5 value is *medium* THEN data sample is BCR-ABL Rule 2: IF gene1 value is *low* and gene2 value is *marginal high* and gene3 value is *high* and gene4 value is *medium* and gene5 value is *low* THEN data sample is BCR-ABL
Negative rules Rule 1: IF gene2 value is *low* THEN data sample is not BCR-ABL Rule 2: IF gene1 value is *medium* and gene2 value is *medium* and gene3 value is *medium* and gene4 value is *medium* and gene5 value is *low* THEN data sample is not BCR-ABL

Using the above fuzzy labels, fuzzy rules with semantic meanings can be formulated from a trained FCMAC-EBYY network. Some rules are listed in as examples Table 5-4. Positive rules referred to the rules used to classify a data sample as belonging to subtype BCR-ABL while negative rules denote the fuzzy rules that classify a data sample as not belonging to BCR-ABL. One can interpret that a data belonging to the ALL cancer subtype BCR-ABL generally has marginal high expression values for gene3 and gene4,

and otherwise it has medium expression values for these genes. The ability to formulate and extract knowledge from a given dataset offers FCMAC-EBYY a key advantage over other types of classifiers such as the SVM, MLP and K-NN.

5.3.2 Bank Failure Prediction Classification

In this section, the experiment in Section 4.4.2 is repeated to show the performance of the proposed coevolution structure. The FCMAC-EBYY network is subsequently used to model the inherent relationships between the financial covariates and their impact on the financial solvency of the respective banks. The simulation is repeated for all the five cross-validation groups. For the experiment, the parameters are set as shown in Table 5-5.

Table 5-5 Parameter Settings for Bank failure Prediction Classification

Parameters	Settings
Number of clusters in population	17
Number of BYY clusters in initial population	5
Maximum number of generations	50
Crossover rate	0.95
Mutation rate	0.05
Number of chromosomes retained in reinsertion	4
Learning rate for FCMAC	0.5

The comparison on bank failure prediction classification of FCMAC-EBYY, GenSoFNN-CRI(S) [63], FCMAC-DIC and FCMAC-BYY is shown in Table 5-6. From the table, it can be seen that the FCMAC-EBYY networks have superior performance to all the other systems. The classification rate, as expected, deteriorates with respect to the

prediction period, since logically, longer prediction period will result in less accurate classification and poorer prediction result.

Table 5-6 Comparison of Bank Failure Prediction Classification Results

Models	CV1	CV2	CV3	CV4	CV5	Ave (%)
Last record						
GenSoFNN-CRI (%)	92.92	90.62	94.37	86.04	95.21	91.83
FCMAC-DIC (%)	90.61	92.27	91.63	89.80	90.01	90.86
FCMAC-BYY (%)	93.95	94.96	94.48	94.64	94.60	94.53
FCMAC-EBYY (%)	**96.05**	**95.29**	**95.81**	**95.61**	**95.29**	**95.61**
1 year prior						
GenSoFNN-CRI (%)	82.26	81.13	87.55	76.6	86.79	82.87
FCMAC-DIC (%)	90.08	83.22	89.10	86.38	85.77	86.91
FCMAC-BYY (%)	93.23	92.77	92.64	93.53	92.77	92.99
FCMAC-EBYY (%)	**93.42**	**93.72**	**93.89**	**93.89**	**93.85**	**93.75**
2 years prior						
GenSoFNN-CRI (%)	68.29	68.29	81.95	74.15	68.29	72.19
FCMAC-DIC (%)	82.34	82.18	83.93	79.77	77.92	81.23
FCMAC-BYY (%)	91.73	91.81	91.19	92.06	92.14	91.71
FCMAC-EBYY (%)	**92.05**	**92.53**	**92.57**	**92.70**	**92.31**	**92.43**

The average rate of classification for GenSoFNN-CRI(S) network is observed to 91.83% for the last record classification (Scenario 1). However, as the prediction period extends to two years, the classification rate falls to 72.19% (Scenario 3). The FCMAC-DIC is able to produce a classification rate of 90.86% which eventually declines to 81.23%. Additionally, the FCMAC-BYY network also gives impressive classification rates between 94.53% and 91.71% over the various prediction periods. On the other hand,

FCMAC-EBYY is able to further enhance the classification, giving classification rates which range between 95.61% and 92.43%.

5.4 Summary

Coevolution Computation is inspired by the evolution of biological life in the natural world which progresses with not single genetic algorithm but coevolutionary algorithm. It allows simultaneous evolution of two or more species with couple fitness. Such coupled evolution favors the discovery of complex solutions whenever complex solutions are required. Using global search and optimization techniques, Coevolution computation aims to explore the search space to search of the optimal solution.

In this chapter, cooperative coevolution computation is integrated into the FCMAC-BYY network to search for the global solutions. After obtaining the cluster parameters in BYY fuzzification, cooperative coevolution computation is utilized to simultaneous evolution of two species, namely fuzzy clusters and neuron weighting species. This strategy offers exploring the search domains of both fuzzy clusters and neuron weighting of the proposed model.

The FCMAC-EBYY is put through two experiments to verify its performance, namely cancer subtype classification, and bank failure classification. The results have shown improvements to the classification rates from the cancer subtype classification as well as the bank failure classification.

Chapter 6

An Online FCMAC-OBYY

Nowadays, together with the increasingly infinite amount of data information, automatic adaptation techniques and online learning schemes for stream data are becoming more and more interesting areas for researchers. For online identification tasks, this requires an adaptation of some model parameters in the form of incremental learning steps with new data. This is because a complete rebuilding of the models from time to time with all recorded measurements result in an unacceptable computational effort. Other requirements to incremental learning include refinement of existing knowledge-based models with data, preventing a virtual memory overload in the case of huge databases. The online learning tasks also require an automatic improvement of the accuracy and generalization capability of models, which are initially trained on only a limited data known before hand, during running period [76-78].

Online learning is a major aspect in designing a fuzzy neural network-based controller for dealing with dynamically changing problems. In general, the performance of a fuzzy neural network (FNN) strongly depends on the fuzzy system rules and the membership functions. Typically, the fuzzy rules are constructed by either predefined paradigms [79, 80] or automatic methods for generating the fuzzy rules [12, 81-84]. Since no prior knowledge is required for the fuzzy rules or fuzzy sets, the latter has attracted more and more interests.

Both the original fuzzification based on Ying-Yang learning and the coevolutionary FCMAC-EBYY suffer from two main problems. First of all, the parameter learning phase can only be carried out when we already get all the training data, so they are not suitable for online applications where the data are not always available during the training phase. Moreover, the fuzzy clusters of the original fuzzification based on Ying-Yang learning are systematically performed by the cluster number selection phase. However, in some applications the number of fuzzy clusters needs to be controlled and adjusted to reduce the output errors.

To address the above two problems, in this chapter we propose an online Bayesian Ying Yang (OBYY) learning for fuzzification, which is able to handle online data as well as control the number of fuzzy clusters to satisfy the output error criterion.

6.1 Online BYY Learning for Fuzzification

As shown in Figure 6-1, the OBYY learning for fuzzification consists of two optimization steps. In the first step, the fuzzy clusters are adjusted by the cluster adjustment to maximize the log likelihood or the degree that the input patterns belong to the fuzzy clusters. This is called the input data harmonization step, which obtains the optimal fuzzy clusters and rules that achieve the highest harmony between the input data and fuzzy clusters. In the second step, the output error criterion is taken into account in

the cluster creation and cluster pruning. This is called the output error optimization step and allows the online FCMAC-OBYY to control the overall performance of the model.

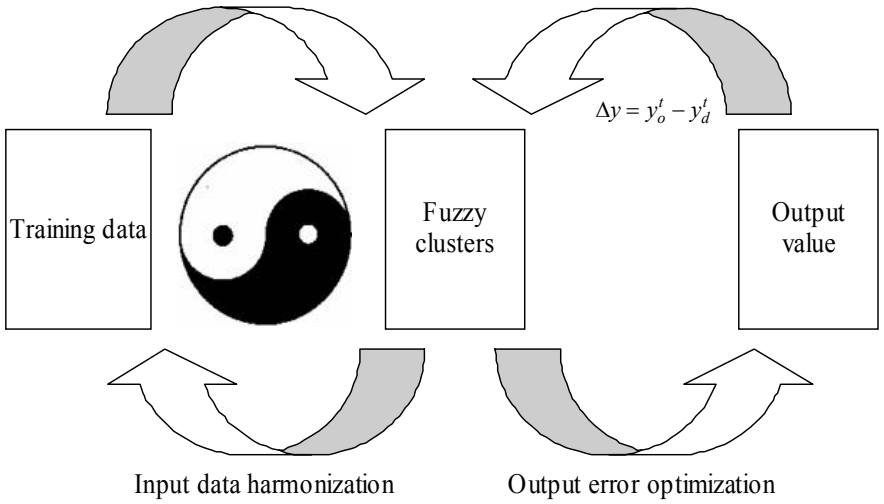


Figure 6-1 Illustration of the online Bayesian Ying-Yang fuzzification

The membership value is defined as the Gaussian basis function acting on each dimensional input:

$$G(x^{(t)}, c_j) = \exp\left[-\frac{1}{2}\left(\frac{x^{(t)} - m_j^{(t)}}{\sigma_j^{(t)}}\right)^2\right] \qquad \text{for } j=1,2,\ldots,J(i) \qquad (6.1)$$

where $J(i)$ is the number of clusters in each dimension, $x^{(t)}$ is the input vector at time t and $m_j^{(t)}$ is the center of the jth cluster with width $\sigma_j^{(t)}$.

6.1.1 Cluster Creation

For each incoming training pattern, the online FCMAC-OBYY will determine whether or not a new fuzzy cluster should be created based on the following two criteria.

The first one is the total firing strength of the activated fuzzy rules.

$$F(x^{(t)}) = \sum_{j=1}^{a(t)} \ln(x^{(t)} \mid \theta_j) \tag{6.2}$$

where $x^{(t)}$ indicates the incoming pattern at time *t*, $\ln(x^{(t)} \mid \theta_j)$ is the log likelihood value indicating the degree that the input data, $x^{(t)}$, belongs to the *j*th cluster for $j \in [1, a(t)]$, and $a(t)$ is the number of activated clusters at time *t,* and θ_j stands for all the parameters of that cluster. When a new data is fed into the network, the total firing strength is computed and compared to the pre-specified threshold value ε. If $F(x^{(t)}) < \varepsilon$ this implies that the total firing strength of the existing network is not satisfied, and a new fuzzy cluster should be created.

The second criterion is based on the error constraint. It is not sufficient to consider the total firing strength of the fuzzy rules as the only criterion to generate new fuzzy clusters; a new fuzzy cluster also should be created when the error of the network does not satisfy the error criterion $y_o^t - y_d^t > k_e$ where k_e is a predefined error criterion threshold, $y_o^{(t)}$ and $y_d^{(t)}$ are the real output and desired output at time *t*, respectively.

6.1.2 Cluster Adjustment

In this section, the original BYY-based fuzzification is modified to handle online training for current patterns. Now, we assume that the number of clusters at time *t* is *k(t)*, then the total probability at time *t* is:

$$p(x^{(t)} \mid \Theta_K^{(t)}) = \sum_{j=1}^{k(t)} \alpha_j^{(t)} p(x^{(t)} \mid \theta_j) \tag{6.3}$$

and the posterior probability in the Expectation step of EM algorithm becomes:

$$P(c_j \mid x^{(t)}) = \frac{\alpha_j^{(t-1)} p(x^{(t)} \mid \theta_j)}{p(x^{(t)}, \Theta_k)} \tag{6.4}$$

where $\alpha_j^{(t-1)}$ is the prior probability at the preceding time, which can be obtained online by previous calculation result of the *j*th cluster. Then, the probability $\alpha_j^{(t)}$ at the moment *t* can be computed as follows [85]:

$$\begin{aligned} \alpha_j^{(t)} &= \frac{1}{t}\sum_{i=1}^{t} p(c_j \mid x^{(i)}) \\ &= \frac{1}{t}\left[(t-1)\alpha_j^{(t-1)} + p(c_j \mid x^{(i)})\right] \end{aligned} \tag{6.5}$$

where

$$\alpha_j^{(t-1)} = \frac{1}{(t-1)}\sum_{i=1}^{t-1} p(c_j \mid x^{(i)}) \tag{6.6}$$

Therefore, at the beginning, if we assume the prior probability $\alpha_j^{(0)}$ is uniform then the posterior probability for the first data $p(c_j \mid x_1)$ in (6.4) can be obtained as well as the probability $\alpha_j^{(1)}$ in (6.5). These values are then stored for the next computation. Based on

the above analysis, the online learning of $\theta_j = \left(m_j, \sigma_j\right)$ can be carried out in the modified maximization step of EM algorithm:

$$\begin{aligned} m_j^{(t)} &= \frac{\sum_{i=1}^{t} p(c_j \mid x^i) x^i}{\sum_{i=1}^{t} p(c_j \mid x^i)} \\ &= \frac{\sum_{i=1}^{t-1} p(c_j \mid x^i) x^i + p(c_j \mid x^t) x^t}{\alpha_j^t t} \\ &= \frac{(t-1)\alpha_j^{t-1} m_j^{(t-1)} + p(c_j \mid x^t) x^t}{\alpha_j^t t} \end{aligned} \tag{6.7}$$

$$\begin{aligned} \sigma_j^{(t)} &= \frac{\sum_{i=1}^{t} p(c_j \mid x^i)(x^i - m_j^t)^2}{\sum_{i=1}^{t} p(c_j \mid x^i)} \\ &\cong \frac{\sum_{i=1}^{t} p(c_j \mid x^i)(x^i)^2 - (m_j^t)^2}{\sum_{i=1}^{t} p(c_j \mid x^i)} \\ &= \frac{\sum_{i=1}^{t-1} p(c_j \mid x^i)(x^i)^2 + p(c_j \mid x^t)(x^t)^2}{\alpha_j^t t} - (m_j^t)^2 \\ &= \frac{(t-1)\alpha_j^{t-1}\left[\sigma_j^{(t-1)} + (m_j^t)^2\right] + p(c_j \mid x^t)(x^t)^2}{\alpha_j^t t} - (m_j^t)^2 \end{aligned} \tag{6.8}$$

Therefore, all the parameters at time *t* can be computed base on these parameters at time *t-1*. This feature allows the proposed online Bayesian Ying-Yang learning for fuzzification to handle online applications which require an automatic improvement of the accuracy and generalization capability of models during running period [76-78].

6.1.3 Cluster Pruning

The objective of the online Bayesian Ying Yang learning for fuzzification is to minimize the difference between the two models of Ying and Yang. This is equivalent to maximize the likelihood. Therefore, the prior probability α_j of each fuzzy cluster becomes a nature criterion for judging the performance of that cluster. The prior probability is checked periodically, i.e. after a certain number of patterns presented to the network, the prior probability of all fuzzy clusters will be computed and the cluster with smallest prior probability value will be pruned. We can perform the cluster adjustment at time *t* as follows:

Step 1: Find the cluster c_j, which has the minimal α_j,

Step 2: Temporarily delete cluster c_j,

Step 3: Perform the error checking $y_o^t - y_d^t < k_e$.

If the online FCMAC-OBYY passes the error criterion, the error is lower than a predefined error criterion threshold k_e defined in Section 6.1.1, it means that the fuzzy clusters over-cover the input patterns and the network has low generalization capability; therefore, that cluster should be deleted. On the contrary, if the network cannot pass the error criterion, it means that the network already has a good condition and that cluster should not be deleted.

6.1.4 Algorithm of OBYY-based Fuzzification

The online Bayesian Ying Yang learning for fuzzification can be summarized as follows:

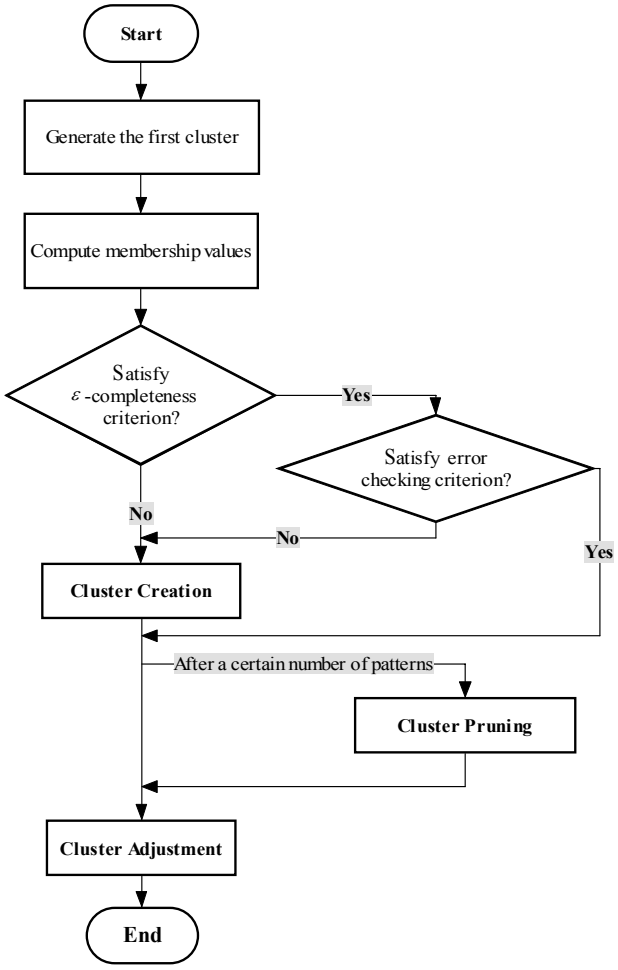


Figure 6-2 Flowchart of the online Bayesian Ying Yang learning for fuzzification

6.2 OBYY-Based Fuzzy CMAC

Based on the learning architectural, the existing fuzzy neural networks can be divided into two categories: global training scheme and local training scheme. In the former, the

output of the network is computed by all the neurons from previous layers. The parameters learning are then done on all the neurons and all the weights are adjusted at each training step. In other words, an incoming training pattern will affect the whole network. Some examples of this category can be found at FALCON [86, 87], GenSoFNN [12], and POPFNN [88]. On the other hand, in the local training structures, the output of the network is computed by only some "activated" neurons which are activated by the input pattern. It means that for each training step, only a subset of neurons that effectively contributed to the output will be trained. That indicates a more efficient algorithm and reduced computational costs. Some candidates of this category can be found at EFuNNs [89], dmEFuNNS [90], and FCMAC [91].

In the global training schemes, the learning process is only concerned with parameter level adaptation within fixed structures and it is time consuming. For large-scale online data problems, it will be too complicated to determine the optimal premise-consequent structures, rule numbers, etc. On the other hand, the local learning structures have fast learning ability with one pass (epoch) training. Moreover, an important feature of these structure is the local generalization which allows only a subset of neurons are changed to adapt a new training pattern while the other neurons still remain. This feature is highly capable of online learning which required only a necessary changing to adapt a new coming data which the rest still remain to keep the historical information.

6.2.1 System Architecture

In this research, the online FCMAC-OBYY that inherited the local generalization is proposed by incorporating with the OBYY algorithm. The proposed online FCMAC-OBYY has five layers of neurons shown in Figure 6-3. In this network model, all nodes/connections are created/connected as input data are presented. An online BYY-based FCMAC is basically a combination of fuzzy linguistic presentation and CMAC model. On one hand, the online FCMAC-OBYY has localization learning abilities and connectionist structures of the CMAC model, which uses mathematical relationship and mapping to learn weighted real values. On the other hand, it has the capability of human oriented representation using linguistic fuzzy labels and fuzzy value outputs.

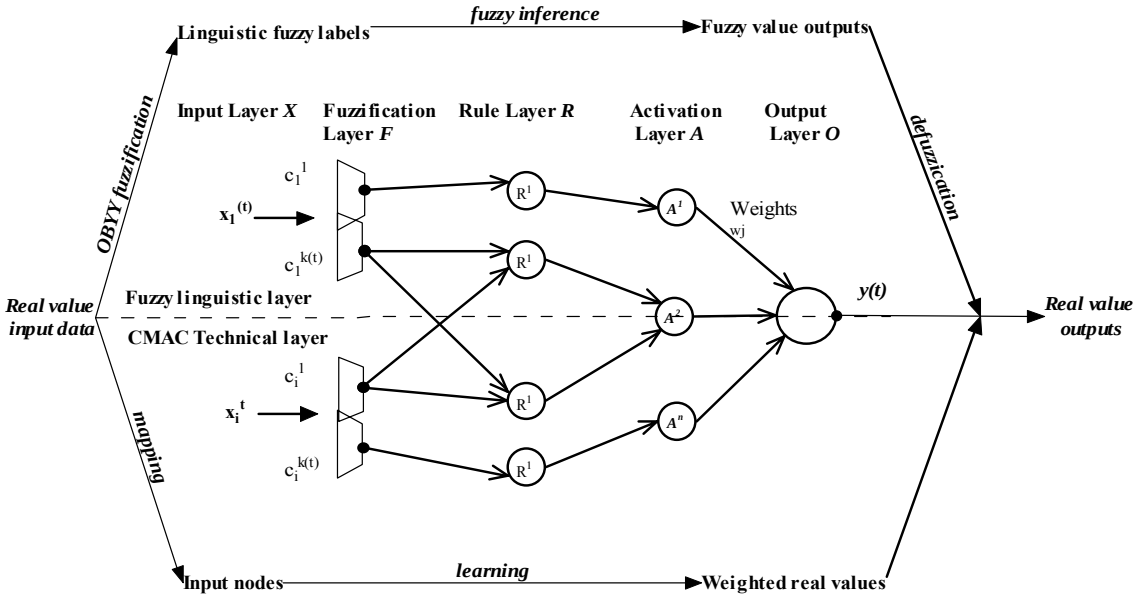


Figure 6-3 The architecture of the online FCMAC-OBYY

1) *Input Layer:* Input variable of the network, $X = [x_1, x_2, \ldots, x_n]$ is obtained from the raw data set as crisp value.

2) *Fuzzification Input Layer:* The second layer represents fuzzy quantization of each input variable. To obtain fuzzy labels, the fuzzy input nodes transfer the input values into membership values which are real numbers from 0 to 1 using Gaussian function. Each node in this layer corresponds to one linguistic value (small, medium, large, etc.) of one of the input variables in Layer 1. All clusters in this layer could be created, adjusted, aggregated, and pruned dynamically based on the online Bayesian Ying-Yang algorithm.

3) *Rule Layer:* The rule layer contains rule nodes that evolve through learning process. Each combination of the fuzzy cluster in the Fuzzification Layer becomes a rule node which represents a fuzzy rule.

4) *Activation Layer:* Each activated rule node in the Rule Layer becomes a firing neuron in this layer. All the neurons here are fully connected to the Output Layer. One novel parameter named "firing degree" is introduced to be relative with the number of times that a neuron is involved in the output calculation. This parameter is motivated from human behavior that not all of the neurons behave equally, but for some particular task the neurons that are fired more frequently than others should maintain its stability; the details are described in the next sub-section.

5) *Output Layer:* The defuzzified values of the output variables are computed by the defuzzification center of area (COA) method.

6.2.2 Credit Assignment Applied to FCMAC-OBYY

The learning speed plays an important role in producing a good learning model for real time applications within real time constraints. In this research, the credit assignment learning is introduced to the original CMAC to speed up the learning process. In the conventional CMAC learning schemes, the errors are distributed equally to all the activated neurons of the current input pattern as shown in Figure 6-4. In this way, the historical contributions of individual neurons to the past patterns are not taken into account.

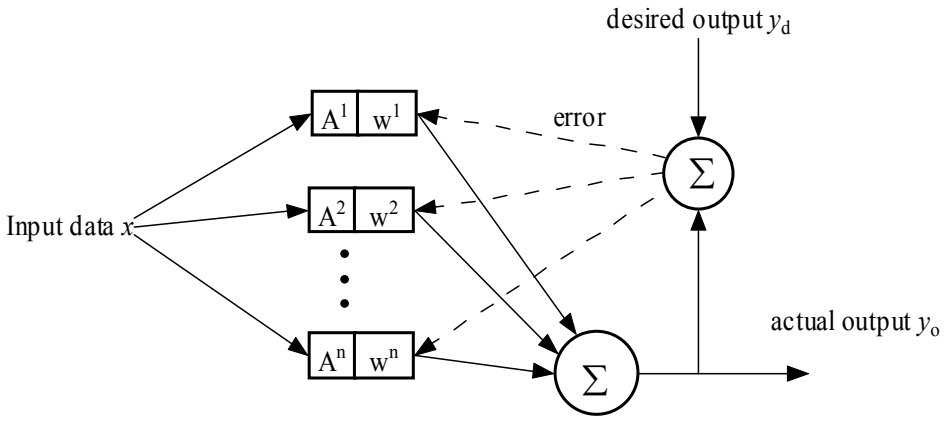


Figure 6-4 The training process of the conventional CMAC

To overcome the above shortcoming, credit assignment is applied to consider the responsibilities of neurons [92]. In the learning process of the online FCMAC-OBYY, a variable, named *f_freq*, is added to each neuron to count the number of times the neuron has been fired. Using this technique, the neurons which are fired more frequently will perform learning at a reduced rate. Prior to this switch, the weights are updated according

to the total number of activated neurons instead. The following formula is applied to the updating of the weights in the online FCMAC-OBYY:

$$w_j^{(t)} = w_j^{(t-1)} + \frac{\alpha\left(\sum_{l=1}^{n} f(l) \Big/ f(l)\right)}{\eta}\left(y_d^t - y_o^t\right) \tag{6.9}$$

and

$$\eta = \sum_{l=1}^{n}\left\{\frac{\sum_{l=1}^{n} f(l)}{f(l)}\right\} \tag{6.10}$$

where $w_j^{(t)}$ is the weight of the jth activated neuron at time t, α is the learning rate while y_d^t and y_o^t are the desired and calculated output respectively, and $f(l)$ returns the variable *f_freq*. Using the derived η of n activated neurons, Equation (6.10) reduces proportionally the learning rate of a activated neuron as its fired frequency increases.

6.3 Experimental Result

We now compare the performance of the online FCMAC-OBYY with other representative fuzzy neural networks. Our experiments were conducted on a chaotic time series Mackey-Glass data set, and a real-time application traffic prediction.

6.3.1 Mackey-Glass Dataset

In this section, the online FCMAC-OBYY will be applied to modeling and predicting the future values of a chaotic time series: the Mackey-Glass (MG) data set [93], which has been used as a benchmark example in the areas of neural networks, fuzzy systems and hybrid systems. This time series is created with the use of the MG time-delay differential equation defined as:

$$x(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{6.11}$$

To obtain values at integer time points, the fourth-order Runge–Kutta method was used to find the numerical solution to the above MG equation. In this experiment, the time step is set to 0.1, $x(0)=1.2, \tau=17$ and $x(t)=0$ for $t<0$. The task is to predict the values $x(t+85)$ from input vectors $[x(t-18)\ x(t-12)\ x(t-6)\ x(t)]$ for any value of the time t.

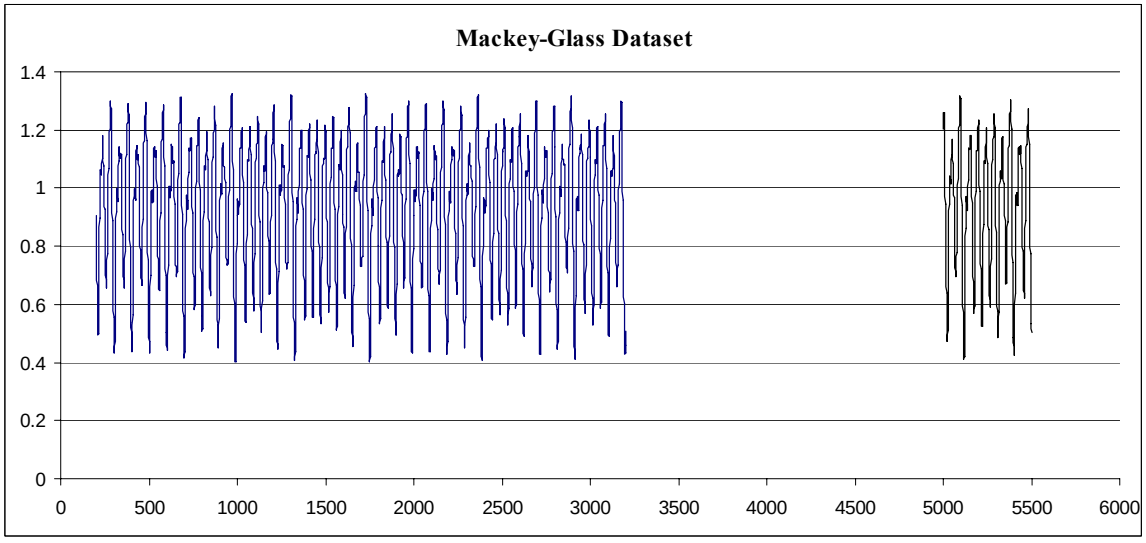


Figure 6-5 The Mackey-Glass time series dataset

As shown in Figure 6-5, the following experiment was conducted: 3000 data points, for $t = 201$ to 3200, are extracted from the time series and used as learning (training) data; 500 data points, for $t = 5001$ to 5500, are used as test data. For each of the aforementioned online models, the learning data is used for the online learning processes, and then the testing data is used with the recalling procedure.

Table 6-1 Prediction results of online learning models on Mackey-Glass test data

Methods	**Fuzzy rules(DENFIS and Online-FCMAC-BYY) Rule nodes (EFuNN) Units (others)**	**NDEI for testing data**
Neural gas	1000	0.062
RAN	113	0.373
RAN	24	0.17
ESOM	114	0.32
ESOM	1000	0.044
EFuNN	193	0.401
EFuNN	1125	0.094
DENFIS	58	0.276
DENFIS	883	0.042
DENFIS with rule insertion	883	0.033
Online FCMAC-OBYY	**16**	**0.072**

To demonstrate the performance of the proposed system, we use some existing online learning models applied on the same task. These models are Neural gas [94], RAN [95], ESOM[78], EFuNN [77], and DENFIS [89]. In this experiment, the non-dimensional error index (NDEI) [93], which is defined as the root mean-square error (RMSE) divided by the standard deviation of the target series, is used as the measurement. As shown in

Table 6-1, the proposed online FCMAC-OBYY can archive a promising result with fewer rules as compared to other models.

The proposed online FCMAC-OBYY model can be used also for offline, batch mode training. In addition to the NDEI, in the case of offline learning, the learning time is also measured as another comparative criterion. The learning time is the CPU-time (in seconds) consumed by each method during the learning process. Table 6-2 shows the comparison results of well-known models including: adaptive neural-fuzzy inference system (ANFIS), a multilayer perceptron trained with MLP-BP, and DENFIS which are implemented by MATLAB, UNIX version 5.5 and reported in [89]. The online FCMAC-OBYY is implemented by Visual C++ with the hardware configuration as follows: CPU = Intel Pentium IV 2.2GHz, operating system = Microsoft Window 2000, memory available = 512 Mbytes.

Table 6-2 Prediction results of off-line learning models on Mackey-Glass data

Methods	Neurons or Rules	Training NDEI	Testing NDEI	Training Time (s)
MLP-BP	60	0.083	0.090	1779
ANFIS	81	0.028	0.029	94210
DENFIS	50	0.017	0.016	351
Online FCMAC-OBYY	**16**	**0.010**	**0.012**	**1.8**

From Table 6-2, one can observe that the fast learning speed of FCMAC together with the credit assignment learning offer the proposed online FCMAC-OBYY a significant improvement on time consumed of the learning process as compared to other representative models.

6.3.2 Traffic Prediction

In this section, the online FCMAC-OBYY is applied to a real-time traffic prediction application. The raw traffic flow data for the simulation was obtained from Tan [96]. The data were collected at a site (Site 29) located at exit 15 along the east-bound Pan Island Expressway (PIE) in Singapore using loop detectors embedded beneath the road surface. There are a total of five lanes at the site, two exit lanes and three straight lanes for the main traffic. For this experiment, only the traffic flow data for the three straight lanes were considered. The traffic data set has four input attributes. The four attributes are time and the traffic density of the three lanes. The purpose of this simulation is to model the traffic flow trend at the site using online FCMAC-OBYY network. It is then used to obtain prediction for the traffic density of each lane at time $t+\tau$, where $\tau = 5, 15, 30, 45, 60$ min. Figure 6-6 shows a plot of the traffic flow density data for the three straight lanes spanning a period of six days from 16:20 on 5th to 10:30 on 10th September 1996.

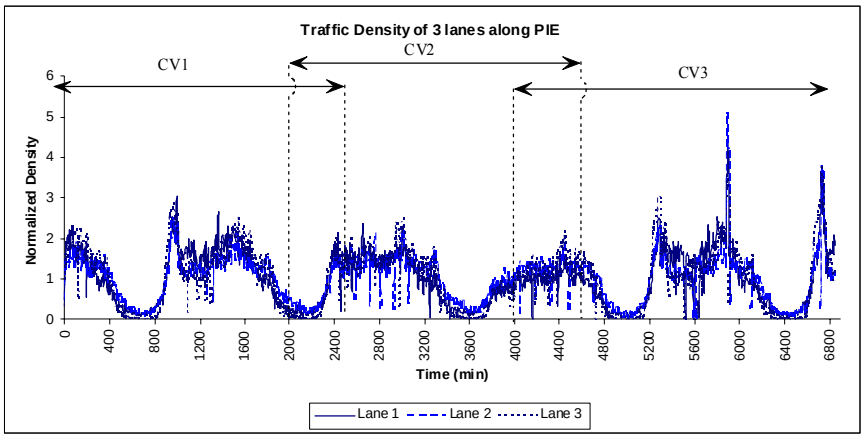


Figure 6-6 Traffic density of three straight lanes along PIE

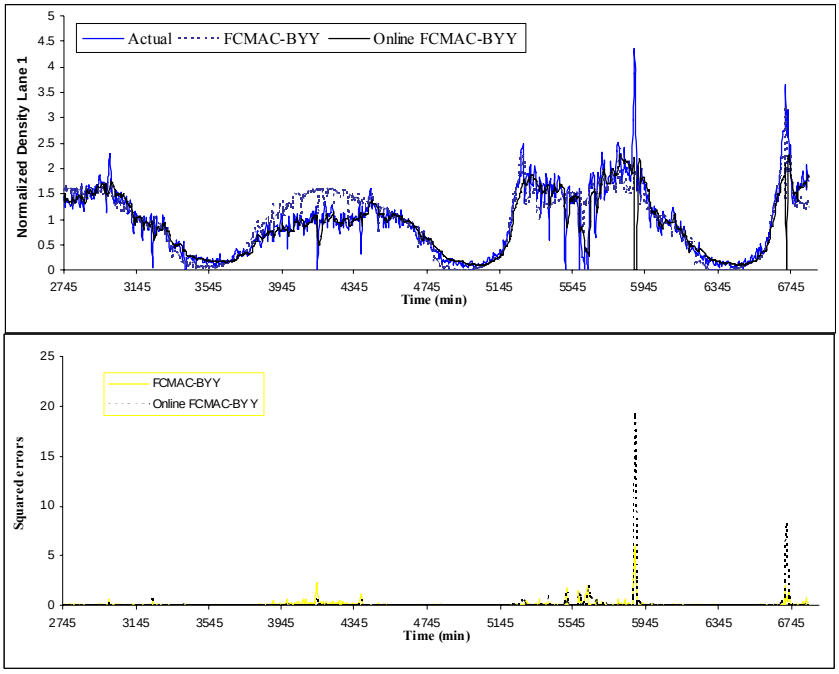


Figure 6-7 Prediction and squared errors at $\tau = 5$ mins

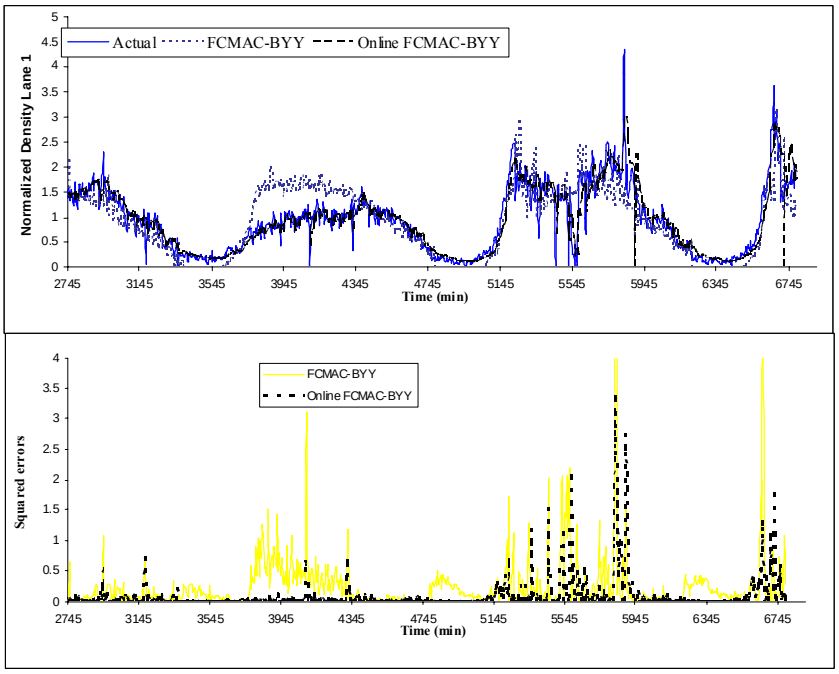


Figure 6-8 Prediction and squared errors at $\tau = 60$ minutes

For the simulation, three cross-validation groups of training and test sets are used. They are CV1, CV2, and CV3. The square of the Pearson product-moment correlation value (denoted as R^2) [97] is used to compute the accuracy of the predicted traffic trends obtained using online FCMAC-OBYY network. The prediction and squared errors of lane 1 density using online FCMAC-OBYY are shown in Figure 6-7 at $\tau = 5$ and in Figure 6-8 at $\tau = 60$ of CV1.

The mean square errors (MSEs) against time interval $\tau = 5$ to $\tau = 60$ for lane 1 using CV1 as training set for the online FCMAC-OBYY, FCMAC-BYY and the GenSoFNN-CRI(S) [12] are shown in Figure 6-9.

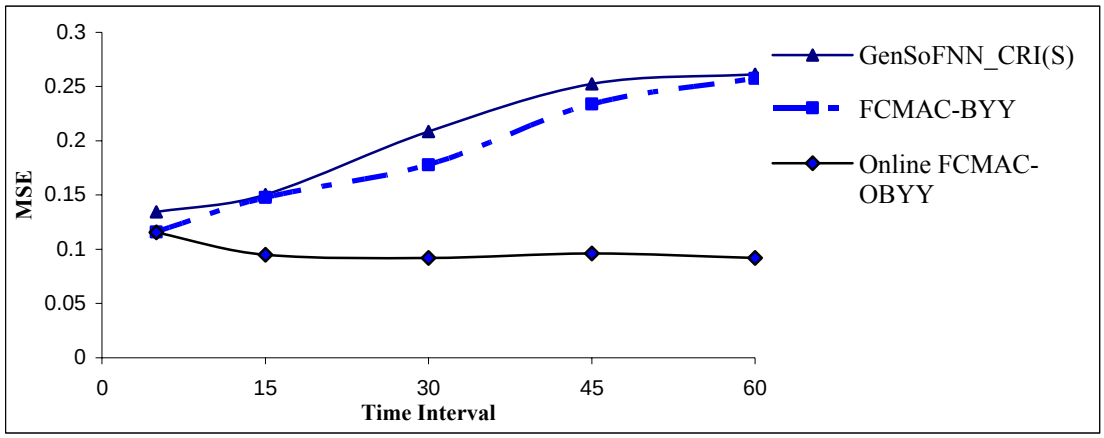


Figure 6-9 Mean squared errors of online FCMAC-OBYY, FCMAC-BYY and GenSoFNN-CRI(S) for $\tau = 5$ to $\tau = 60$

Table 6-3 Simulation results of traffic prediction

Network	Lane1 Var(%)	Lane2 Var(%)	Lane3 Var(%)	Avg Var(%)
Falcon-FCM(CL)	24.17	9.32	30.47	21.32
Falcon-MLVQ(CL)	36.41	25.94	30.21	30.85
Falcon-FKP(CL)	23.87	22.09	35.19	27.05
Falcon-PFKP(CL)	20.78	21.05	28.25	25.70
Falcon-MART	20.78	15.47	20.58	18.94
GenSoFNN-VRI(S)	19.64	19.58	21.09	20.10
FCMAC-BYY	10.75	8.95	15.36	11.68
online FCMAC-OBYY	**10.34**	**8.57**	**13.84**	**10.92**

The "Var" indicator (the change in Avg R^2 value from $\tau = 5$ min to $\tau = 60$ min expected as a percentage of the former) and the "Avg Var", the mean "Var" values across all three lanes, are used for the benchmarking of the various systems. These two indicators reflect the consistency of the predictions made by the benchmarked systems over different time intervals across the three lanes. The online FCMAC-OBYY prediction model is compared with the Falcon-class of network [98], and the GenSoFNN-CRI(S) network [12]. Table 6-3 shows that online FCMAC-OBYY has superior performance over all the networks.

6.4 Summary

This chapter extends BYY learning to the online fuzzification phase of FCMAC. The performance of the proposed model is validated by using a Mackey-Glass dataset and the real traffic flow data. Simulation results show that the online FCMAC-OBYY can approximate the data dynamics accurately and consistently. The proposed system

demonstrated superiority when compared to Neural Gas [94], RAN [95], ESOM[78], EFuNN [77], and DENFIS [89] in the case of Mackey-Glass dataset and Falcon-class of network [98], and the GenSoFNN-CRI(S) network [12] in the case of traffic flow data.

Moreover, the proposed online FCMAC-OBYY is also realized in a web-based stock prediction system in the next chapter. The software can automatically acquire data from stock exchange companies via internet and perform stock price prediction based on analyses and extraction of patterns from historical data.

Chapter 7

A Web-based Stock Prediction System

Stock trading is becoming more and more popular among regular to people. This design attempts to create a low-cost autonomous software that obtains live stock data from the web by data extraction module and uses both live and past data to intelligently predict stock prices by data Prediction module. The targeted market for this software will be the home-users and provide them with a low cost advisory tool which assists them in forecasting stock prices. These forecasts are based on analysis and extraction of patterns from historical data.

Stock data in Singapore is available through different means. In Singapore, the two most popular ways are through the television's TeleText menu and the internet. This research will only focus on stock data available on the internet because of its lower delay timings and its ability to extract and collate information directly into our application. Our software system is designed to aids in recording and analyzing stock data. The designed application allows monitoring and predicting stock values of several companies, with an user-friendly user interface.

7.1 Hardware & Software Requirements

The design was carried out using an Intel® Pentium® 4 Mobile 1.6GHz computer with onboard 512MB RAM and 20GB of HDD space. The operating system used is Windows XP Professional. The following main software was used in developing the software.

- Microsoft Visual Studio 6 and Microsoft Visual Studio 2003.
- Microsoft SQL Server.

This design requires the following software to be installed:

- NET Framework 2.0.
- SQL Server.
- Operation system: Windows XP and later.

The online stock prediction application was coded mainly in C# and C++, it consists of two major parts:

1. Data Extraction.
2. Data Prediction.

The first part is used to extract the data from relevant websites. After that, the downloaded data are stored database. Whereas the second part uses to train the neural network and finally obtains a prediction based on the inputs and trained data. The details of these two modules are given in the next sections.

7.2 Data Extraction

The *data extraction* module must be versatile and intelligent. This is to ensure that it can handle changes in the site's web programming without misrepresenting data. The web extractions will be carried out using a local stock website www.sgx.com of Singapore Exchange Limited as shown follows.

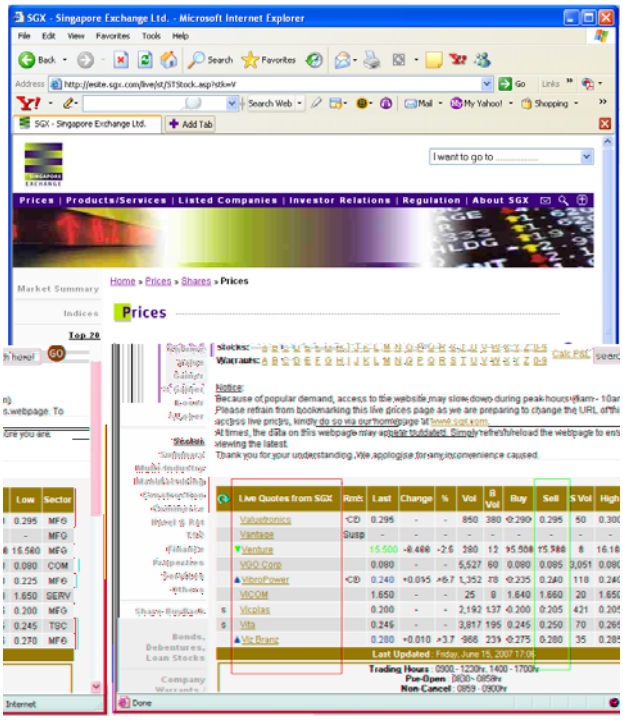


Figure 7-1 Data search and value extraction

The arrangement of the data can be seen in the sample screenshot as shown in Figure 7-1, which is a screenshot of the SGX website of stock prices of companies with the initial '***V***'. The left highlight represents the stock indexes which the search algorithm of the data parser would analyze. The right highlight represents the value which is extracted and sent to the database.

The URL of the SGX Stock index prices are all easily located based on the stock index's initial character. An example would be the stock prices of the stock "*VibroPower*" is found at this URL: *http://stquote.sgx.com/live/st/STStock.asp?stk=**V***. This URL also contains all other stock indexes whose initial starts with a '***V'***. Hence the data search process of this module will consist of the following steps:

1. Requesting the input from the user,
2. Obtain the initial character of the user string
3. Appending the character to string *http://stquote.sgx.com/live/st/STStock.asp?stk=*
4. Initializing connection to webhost (either using *iOpus File & Webpage Downloader* or through C#'s *WebRequest* Class)

7.2.1 Web Download Module

The Web Download module is first used to easily obtain HTML data from a particular website. The Web Download module contained the freeware file *iOpus File & Web Page Downloader*. The downloader is simply used and integrated well with the C# application code.

7.2.2 Web Request Module

The data downloading for the application is then carried out using a powerful built-in library *Web Request class* of C#, which allows for simple web interactions but does not require the use of file handlers to store downloaded files. The *Web Request* Class allows

for the instantiation of a connection to an external address on the network (intranet/internet), from which the data is required.

7.2.3 Parsing Downloaded Information

In the case of the Online Stock Prediction Application, a basic stock request is sent to a local stock market website. Upon receiving the request, the HTML file is parsed and valuable information is extracted and inserted into appropriate tables in the application's MSSQL 2000 database. This HTML file would then be regularly polled as a background process and data collected and inserted into respective database tables periodically. A simplified diagram of the database is shown in Figure 7-2.

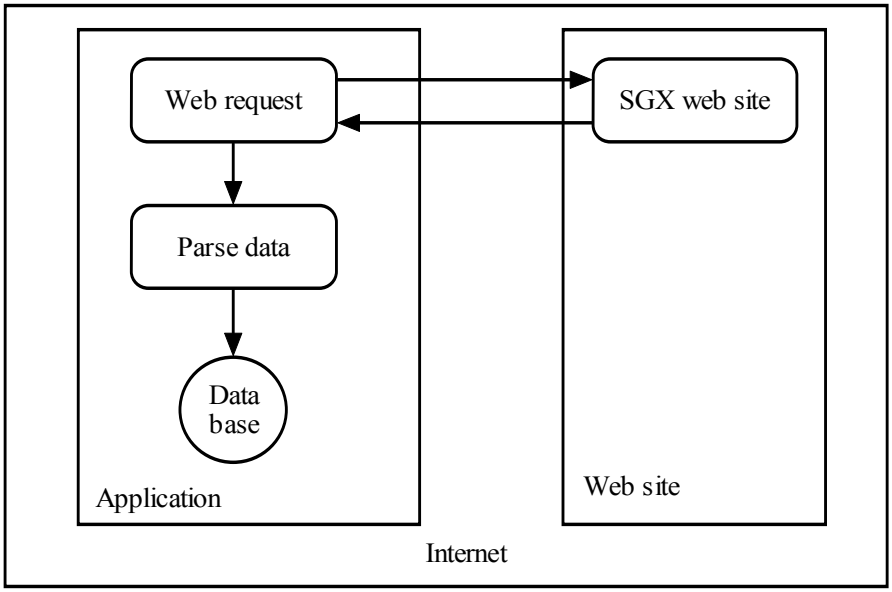


Figure 7-2 Data Extraction Process

7.3 Stock Prediction

Stock prediction involves making use of the features exhibited by a time-series data. The analysis of this data will determine the choice of a suitable forecasting technique. Generally, historical time-series data may reflect one or more of the following features:

Trends: A trend is a gradual upward or downward long term movement in a time-series without time related and irregular effects.

Seasonal and non-seasonal cycles: A seasonal cycle is a repetitive, predictable time related effect. A non-seasonal cycle is a repetitive and unpredictable pattern in the series values.

Pulses and steps: A data set may also experience sudden changes in level. They generally come in two forms. The first is a sudden, temporary shift or pulse in the series level. The second is a sudden, permanent shift or step in the series level.

A basic prediction method involves a few variables. The current time is assumed as *t* and data is needed up to this time $Y_1, Y_2, ..., Y_{t-1}, Y_t$ to make forecasts $F_1, F_2, ..., F_{t-1}, F_t$ of future values of *Y*.

The importance of forecasting has seen the emergence of companies specializing in forecasting packages and software. These range from stand-alone, specialized packages to modules in enterprise management systems. Many of these packages use conventional forecasting methods, such as Simple Exponential Smoothing (SES) [99], Holt's Linear Exponential Smoothing (HLES) [100], Holt-Winter's method with Additive Seasonality (HWA) and Multiplicative Seasonality (HWM) [101]. Another poplar forecasting method

is the Box-Jenkin's Autoregressive Integrated Moving Average (ARIMA) model [102]. Derivatives of the ARIMA model include the Auto Regressive Autoregressive Moving Average (ARARMA) [103] and more recently the delta Non-linear Autoregressive Integrated Moving Average (NARMA) model [104].

Newer forecasting techniques and innovative modifications of present techniques have also emerged. These include pattern matching of historical data [105], which relies on old structures of data that may be used for matching with current structures to generate a future prediction. Segmentation of time-series data to enable further manipulation and extraction of useful information has also been explored successfully [106]. These new techniques have been compared to the classical techniques and have yielded encouraging results indeed. However, these models have various deficiencies and they are not able to identify the traits of financial distress of the stock values and thus function as black boxes.

On the other hand, an online FCMAC-OBYY network, which simulates the human style of reasoning and decision-making when solving complex problems, can overcome the deficiencies of the traditional statistical models and can be employed to handle stock prediction module [107, 108]. This prediction module of our designed application focuses on unvaried time series forecasting which only uses historical data to generate a forecast. The advantage of the stock prediction module is accrued from its online fuzzification technique using Bayesian Ying-Yang learning, which obtains Gaussian clusters from the

stream data as fuzzy rules. Next section describes the integrating Data Extraction module and Data Prediction module.

7.4 Online Stock Prediction using FCMAC-OBYY

In this section, the online FCMAC-OBYY proposed in chapter 6 is implemented as a stock prediction module. Online learning allows on-the-fly learning of new stock values to predict future stock data without reinitializing the neural network.

The MSSQL Database was implemented seamlessly into the application using Microsoft Visual Studio 2003 and integrated the 2 main modules together by storing the information extracted by the Data Extraction module and releasing information to the Data Prediction module. The MSSQL Database also allowed for a meaningful and organized way of storing data based on appropriate nomenclature and data relationships. Data extraction done via polling is a good way to obtain periodic time series data which can be used in time series predictions. Using the .NET Framework's *WebRequest* class allows greater flexibility for the developer because of its small overhead and robust functions. The HTML stream can easily be parsed for information and sent to the database for storage. The polling periods is chosen by the user so as to allow for both short-term and long-term forecasters to use its functions. The application also allows for multiple stocks to be polled at different time periods and is programmed to work based on a Timer in the background. This option does not affect the user's access to the application functions.

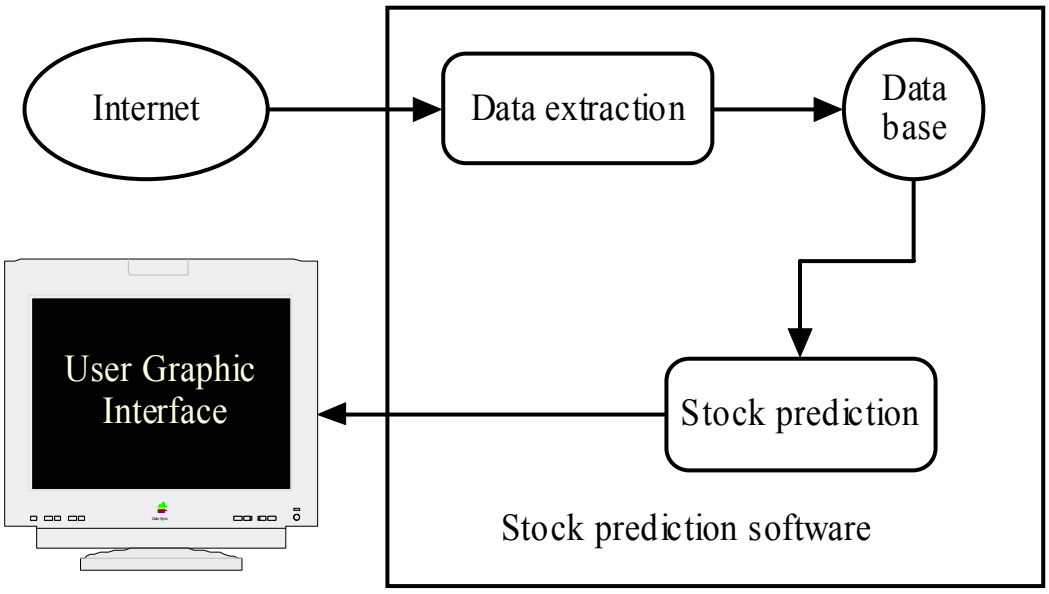


Figure 7-3 Module integration with Database

The MSSQL database was set up and the 6 tables created. Temporary data was added to allow for testing in the next phase. A simple illustration shows the links between the database, Data Extraction and Data Prediction modules. Once the database is properly set up, the testing can finally begin to verify integration of both modules with the Interface module. Figure 7-3 illustrates the database integration to the interface.

The designed application offers the following features:

- Simple to use GUI
- User accounts for multi-user usage
- Able to get stock data from several sites
- Display stock data in a graph or a table, and able to export data to CSV format
- Predict values from existing stock data
- Load several companies stock data at once

- Compare multiple companies at once
- Automatically monitor a company's stock quotes
- Extensible

7.4.1 Monitoring Companies

This feature allows to record the stock quotes of a company over time, and to plot a graph of stock index against time. To monitor a company, the users need to create a "tab" from "Main Menu" > "Tabs" > "Create Tab". The following dialog will be displayed:

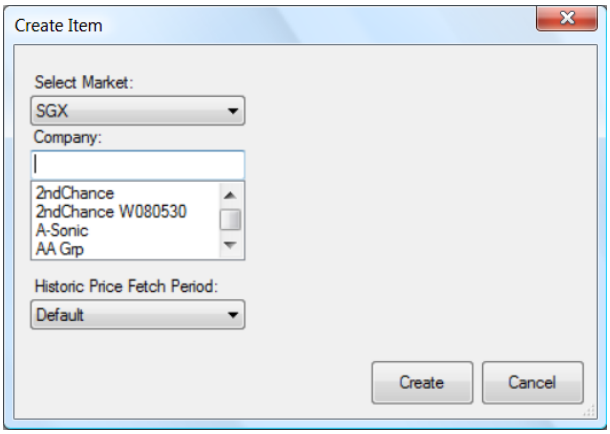


Figure 7-4 Create Tab dialog

After selecting which company wanted to monitor (it can be outside of the default list), click on the "Create" button, a new tab would be created. The company's stock data will be downloaded and stored into the database.

7.4.2 Data Management and Prediction

As shown in Figure 7-5, users can get the stock data of a company at a particular time by clicking on the "Update" button, and choose to view a history of stock data or download a CSV file that contains the stock quotes via the "History" button.

It also allows to shift/zoom the graph using mouse or the buttons next to the graph. Users can view the information in the tab in a separate window by clicking on the "Window" button, and can refresh the view and data points plotted by clicking on the "Refresh" button.

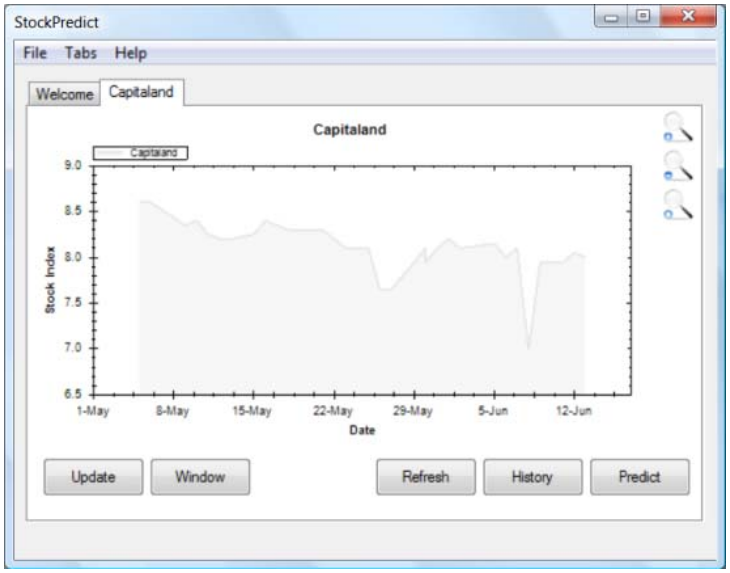


Figure 7-5 Data management and prediction dialog

To predict the future stock data, users can use the "predict" button while monitoring a company via a "tab". A graph will be plotted (as a red graph with a light red fill) to compare the predicted values with the real values.

7.4.3 Tab Groups

"Tab group" feature allows monitor several companies with created tabs. After creating several tabs, users can save the series of tabs as a tab-group (via "Tabs" > "Save Tab Group as"). The next operation time, users can load the saved tab group (via "Tabs" > "Load Tab Group"). Users can also set which tab group to automatically load up the next running time. It also allows to delete a particular tab group (via "Tabs" > "Delete Tab Group").

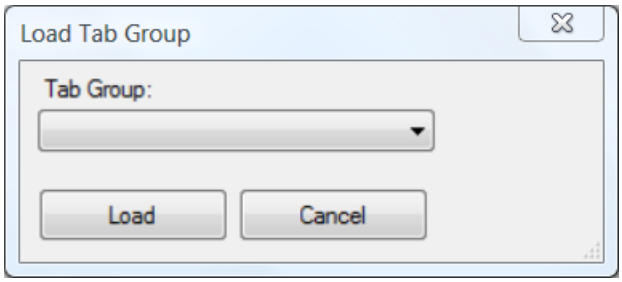


Figure 7-6 Tab group dialog

7.4.4 User Accounts

This feature allows multi-users can use the application. Users can create a user account. via "File" > "Accounts" > "Create", and login via "File" > "Accounts" > "Login". Users can also change their account password via "File" > "Accounts" > "Change Password".

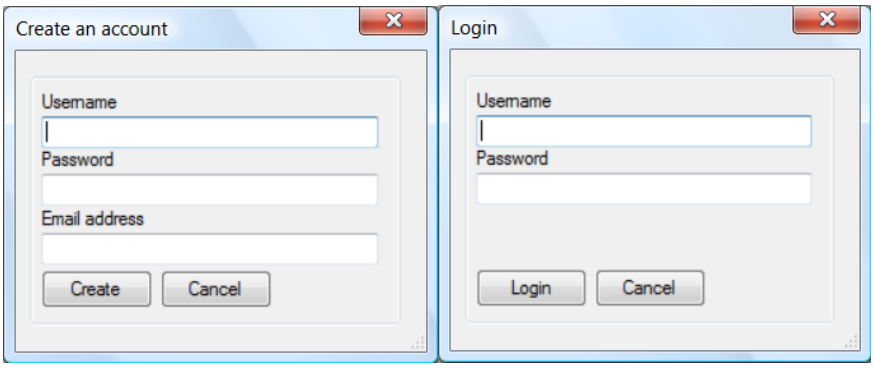


Figure 7-7 User account dialog

7.4.5 Extensions

This feature allows users to add/modify features in the designed application. Users can install extensions by copying them into the "extensions" folder, and then add the extension via the Extension Manager ("Tools" > "Extensions").

7.5 Summary

In this chapter, both Data Extraction and Data Prediction module are successfully integrated to create a software application that is autonomous and robust for online stock prediction. The integration of the both modules was made simple by the use of the functionality of Microsoft .NET Framework and the ease of programming through C#. It coupled with the .NET Framework, which allows developers to create web based applications that interact on the internet without the requirement to know technical details of network protocols. As an integrated software development toolkit, Microsoft Visual Studio 2003 also provides "What You See Is What You Get" interface.

The embedded online FCMAC-OBYY has shown that it can be adapted to the changes of the stock price. Online learning makes sense because retraining historical data would take time and would not work when user requires an immediately prediction. Online learning can be tweaked to play a greater role if the learning rate is set at a higher value. Since the online FCMAC-OBYY allows on-the-fly learning of new stock values, it can predict stock data effectively and efficiently without reinitializing the neural network.

Chapter 8

Conclusions and Future Work

CMAC is an associative feed forward neural network model which simulates our human cerebellum. The original CMAC enjoys the advantages of fast learning speed and local generalization ability, but suffers from huge memory requirement. Fuzzy CMAC can alleviate the memory problem as well as provide human-like reasoning capabilities. However, traditional self-organizing fuzzification techniques cannot provide the optimal fuzzy sets; this research aims to introduce a novel approach to the fuzzification phase to improve the performance of FCMAC for both offline and online applications.

8.1 Conclusions

We first employ Bayesian Ying-Yang learning approach to fuzzification to propose a novel FCMAC-BYY. Treating the input data and the fuzzy sets as a Ying-Yang pair, Bayesian Ying-Yang learning is employed to optimize the number of clusters, their centers and widths in each dimension. The advantage of FCMAC-BYY is accrued from its fuzzification technique using Bayesian Ying-Yang learning, which obtains Gaussian clusters from a raw training data. The BYY requires no prior knowledge of the number of clusters and initial information, but provides the FCMAC network a consistent fuzzy rule base. Another advantage of FCMAC-BYY is the truth-value restriction inference

scheme, which provides FCMAC an intuitive fuzzy logic-reasoning framework. This feature is particularly important in some areas such as financial analysis, medical diagnosis when the domain experts must be involved to analyze the causality.

However the FCMAC-BYY faces two problems: firstly, the fuzzification phase is separated from the learning part of the neural network; secondly, the weights of neurons are trained by using gradient-based methods, which may fall into a local minimum during the learning process. In order to address these two shortcomings, cooperative coevolution computation techniques are successfully infused into the FCMAC-BYY network. Cluster parameters and weights of rules are evolved using two evolutionary techniques in the bid to find the optimal settings. From the experiments results, one can observe that the introduction of cooperative coevolution computation into the FCMAC-BYY system has indeed improved the performance of the system. This is especially true for the cases where the input dataset has a reasonable number of dimensions, like the classification of cancer subtype and bank failure prediction. Moreover, it can be seen from the experiments that the cooperative coevolutionary FCMAC-EBYY is able to take advantage of the strength of evolutionary computation to explore the input domain and find the optimal solution.

Moreover, the online Bayesian Ying Yang (OBYY) learning for fuzzification is proposed with credit assignment to speed up the learning process. In the proposed online FCMAC-OBYY, not only the precondition part is constructed online, but also the structure of the neural network can be generated and self-adaptive during learning. The performance of

the proposed model is validated by using a Mackey-Glass dataset and the real traffic flow data. Simulation results show that the online FCMAC-OBYY can approximate the data dynamics accurately and consistently. The proposed system demonstrated superiority when compared to Neural gas [94], RAN [95], ESOM[78], EFuNN [77], and DENFIS [89] in the case of Mackey-Glass dataset and Falcon-class of network [98], as well as the GenSoFNN-CRI(S) network [12] in the case of processing traffic flow data.

Finally, the research is realized in a web-based stock prediction system using online FCMAC-OBYY. The software can automatically acquire data from stock exchange companies via internet and perform stock price prediction based on analyses and extraction of patterns from historical data. Since the online FCMAC-OBYY allows on-the-fly learning of new stock values, it can predict stock data effectively and efficiently without reinitializing the neural network.

8.2 Future Work

Firstly, the advanced FCMAC-BYY and evolutionary FCMAC-BYY in this research are only applied to classification problems. Our future work includes the application of the proposed methodologies to regression problems.

Secondly, it is noted that the behavioural diversity of the cluster parameters were not examined in this research. Studies in [35] have revealed that behavioural diversity would greatly enhance the coverage of the fuzzy clusters. Further research should take this

factor into consideration. The resultant fuzzy clusters would be able to provide a better mapping of the input data and hence produce a better classification.

Thirdly, the recurrent fuzzy neural network (RFNN), which naturally involves dynamic elements in the form of feedback connections used as internal memories, has been studied by some researchers in the past few years [109-111]. Though the RFNN can be used to solve a variety of problems, the network training using a back-propagation learning algorithm usually cannot guarantee an optimal solution. In practical applications, the back-propagation algorithm might converge to a set of suboptimal weights from which it cannot escape. Therefore, their application domain is not limited to static problems as feed-forward network structures do. Motivated from these complementary features of recurrent architecture, our future works are to integrate both the recurrent architecture and evolutionary learning method into our FCMAC-BYY to propose a new system which offers a more systematic, reliable, and effective solution for online uncertain nonlinear systems with global optimal solution.

Finally, although the proposed online FCMAC-OBYY demonstrates a good local generalization capability and rapid learning speed, it still has a major disadvantage of keeping all the information of the historical data to shape clusters and predict data in the next state. In another word, all the data from the beginning have uniform contribution to the learning process and this is not suitable for online learning in which recent data, which in the majority of real-world applications, are more important and relevant than very old data. On the other hand, the sliding window model [112, 113] is one of the

techniques to keep track of the on-date data that are considered relevant for answering queries. In our future works, the sliding window technique is incorporated to the online FCMAC model in order to eliminated the affect of out-of-date data on the predict decision.

Appendix A

List of financial variables used and their expected effect on bank failure prediction.

Name	Description	Expected Effect of Bankruptcy
CAPADE	Average total equity capital / average total assets	More capital, higher is the ratio, more capacity to absorb losses and hence longer is the time to regulatory closure or financial distress
OLAQLY	(Average (accumulated) loan loss allowance + average (accumulated) transfer risk reserve) / average total loans & lease	More accumulated loan loss provision, higher the ratio; a proxy for overall loan quality
NINMAR	(Total interest income – interest expense) / average total asset	More profitable is the bank higher is the net interest margin, and hence longer is the time to regulatory closure or financial distress.
LIQUID	(Average cash + average federal funds sold) / (average total deposit average fed funds purchased + bank's liability on acceptance + average other liabilities)	Higher is the ratio of liquids assets to short term liabilities, higher is the liquidity, and + to some extend longer is the time to average regulatory closure or financial distress
ADQLLP	(Average (accumulated) loan loss allowance + average (accumulated) transfer risk reserve) / average (accumulated) loans 90+ days late	More provision allocated for problem loans, higher is the ratio
GROWLA	$[(\text{Total loans \& leases, gross})_{t-1} - (\text{Total loans \& leases, gross})_{t}]/(\text{Total loans \& leases, gross})_{t-1}$	A proxy for possible compromise on loan quality due to relatively higher growth of loan assets; higher is the ratio, shorter is time to regulatory closure or financial distress
ROE	(Net income (after tax) + applicable income tax) / average total equity capital	Higher is return on equity before tax longer is the time to regulatory closure or financial distress

Name	Description	Expected Effect of Bankruptcy
NIEOIN	Non interest expense / operating	Higher is the ratio less operationally income efficient is the bank, and hence shorter is the time to regulatory closure or financial distress
PLAQLY	(Annual) loan loss provision / average total loans & leases, gross	More provisions allocated for the period higher is the ratio; intuitively the ratio should also be positively correlated with the growth of loan asset
PROBLO	Average (accumulated) loans 90+ days late / average total loans & leases, gross	Another proxy for overall quality

Appendix B

Relevant genes selected for the ALL cancer subtypes (i.e. BCR-ABL, E2A-PBX1, Hyper > 50, MLL, TALL,and TEL-AML1) by MCES based on T-statistics pre-filtering

Subtype	Affymetrix no.	Gene name	Gene symbol	MCES weight
BCR-ABL	40798_s_at	Disintegrin and metalloproteinase domain 10	ADAM10	0.27760
	36591_at	Tubulin alpha 1 testis specific	TUBA1	0.22779
	39070_at	Singed Drosophila like sea urchin fascin homolog like	SNL	0.17792
	330_s_at	Tubulin, alpha 1, isoform 44	TUBA1	0.15191
	1211_s_at	CASP2 and RIPK1 domain containing adaptor with death domain	CRADD	0.14231
E2A-PBX1	33355_at	Homo sapiens cDNA FLJ12900 fis clone NT2RP2004321 (by CELERA search of target sequence = PBX1)	PBX1	0.75617
Hyper > 50	1447_at	Proteasome prosome macropain subunit beta type 1	PSMB1	0.19138
	41724_at	Accessory proteins BAP31/BAP29	DXS1357E	0.17898
	39867_at	Tu translation elongation factor mitochondrial	TUFM	0.17610
	40875_s_at	Small nuclear ribonucleoprotein 70kD polypeptide RNP antigen	SNRP70	0.16819
	39878_at	Protocadherin 9	PCDH9	0.16432
MLL	1389_at	Membrane metallo-endopeptidase neutral endopeptidase enkephalinase CALLA CD10	MME	0.57293
	33412_at	LGALS1 Lectin, galactoside-binding, soluble, 1 (galectin 1)	LGALS1	0.42637
	40520_g_at	Protein tyrosine phosphatase receptor type C	PTPRC	0.34451
	40519_at	Protein tyrosine phosphatase receptor type C	PTPRC	0.32823
	1520_s_at	Interleukin 1 beta	IL1B	0.22234
	794_at	Protein tyrosine phosphatase non-receptor type 6	PTPN6	0.20051
	307_at	Arachidonate 5-lipoxygenase	ALOX5	0.17356
	37398_at	Platelet/endothelial cell adhesion molecule CD31 antigen	PECAM1	0.15865
	2062_at	Insulin-like growth factor binding protein 7	IGFBP7	0.15317
	174_s_at	Intersectin 2	ITSN2	0.12112
	39705_at	KIAA0700 protein	KIAA0700	0.10828
T-ALL	38319_at	CD3D antigen delta polypeptide TiT3 complex	CD3D	1.30900
TEL-AML1	1488_at	Protein tyrosine phosphatase receptor type K	PTPRK	0.26832
	577_at	Midkine neurite growth-promoting factor 2	MDK	0.24706
	41442_at	Core-binding factor runt domain alpha subunit 2 translocated to 3	CBFA2T3	0.24170
	35614_at	Transcription factor-like 5 basic helix-loop-helix	TCFL5	0.21197
	33162_at	Insulin receptor	INSR	0.21112
	36239_at	POU domain class 2 associating factor 1	POU2AF1	0.19316
	33690_at	cDNA DKFZp434A202 from clone	DKFZp434A202	0.19218
	36985_at	Isopentenyl-diphosphate delta isomerase	IDI1	0.17230
	38652_at	Hypothetical protein FLJ20154	FLJ20154	0.17104
	40745_at	Adaptor-related protein complex 1 beta 1 subunit	AP1B1	0.14523
	41200_at	CD36 antigen collagen type I receptor thrombospondin receptor like 1	CD36L1	0.10462

Appendix C

The Kullback Leibler distance (KL-distance) is a natural distance function from a "true" probability distribution, p, to a "target" probability distribution, q. It can be interpreted as the expected extra message-length per datum due to using a code based on the wrong (target) distribution compared to using a code based on the true distribution [57].

For discrete (not necessarily finite) probability distributions, $p=\{p_1, ..., p_n\}$ and $q=\{q_1, ..., q_n\}$, the KL-distance is defined as follows:

$$\mathrm{KL(p,q)} = \sum_i p_i \ln \frac{p_i}{q_i}$$

For continuous probability densities, the sum is replaced by an integral as follows:

$$\mathrm{KL(p,q)} = \int p(x) \ln \frac{p(x)}{q(x)} dx$$

Author's Publications

A number of publications have results as a consequence of this research. They are listed as follows:

[1] **M. N. Nguyen**, D. Shi, and C. Quek, "FCMAC-BYY: Fuzzy CMAC Using Bayesian Ying-Yang Learning," IEEE Transactions on Systems, Man and Cybernetics-part B, vol. 36, pp. 1180-1190, October, 2006.

[2] **M. N. Nguyen**, D. Shi, C. Quek, and G. S. Ng, "Traffic Prediction Using Ying-Yang Fuzzy Cerebellar Model Articulation Controller," presented at 18th International Conference on Pattern Recognition, 2006. ICPR 2006, 2006.

[3] **M. N. Nguyen**, J. F. Guo, and D. Shi, "ESOFCMAC: Evolving Self-Organizing Fuzzy Cerebellar Model Articulation Controller," presented at International Joint Conference on Neural Networks, 2006. IJCNN '06, 2006.

[4] **M. N. Nguyen**, D. Shi, and C. Quek, "A Nature Inspired Ying-Yang Approach for Intelligent Decision Support in Bank Solvency Analysis," Expert Systems With Applications, 2007 (In press).

[5] **M. N. Nguyen**, D. Shi, and C. Quek, "Self-Organizing Gaussian fuzzy CMAC with Truth Value Restriction," Proceedings of IEEE International Conference of Information Technology and Applications (ICITA), Sydney, Australia., 2005.

[6] **M. N. Nguyen**, U. Omkar, D. Shi, and J. B. Hayfron-Acquah, "Stock Market Price Prediction using Cyclic Self-Organizing Hierarchical CMAC," In:

Proceedings of The 9th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 2006.

[7] J. Fu, **M. N. Nguyen**, and D. Shi, "Stock Prediction Using FCMAC-BYY," Lecture Notes in Computer Science, vol. 4492, 2007.

[8] **M. N. Nguyen** and D. Shi, "An Online Bayesian Ying-Yang Learning Applied to Fuzzy CMAC," Neurocomputing, to appear, 2007.

[9] J. Fu, J. Shi, and **M. N. Nguyen**, "An Early Warning System Based on Fuzzy CMAC," presented at proceedings of International Conference on Machine Learning and Cybernetics (ICMLC), HongKong, 2007.

[10] K. S. Lum, M. **N. Nguyen**, and D. Shi, "GA Based FCMAC-BYY Model for Bank Solvency Analysis," presented at IEEE Congress on Evolutionary Computation (CEC), Singapore, 2007.

Bibliography

[1] J. S. Albus, "Data storage in the cerebellar model articulation controller (CMAC)," *Transaction of the ASME, Dynamic Systems Measurement and Control*, vol. 97, no. 3, pp. 228-233, 1975.

[2] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Transaction of the ASME, Dynamic Systems Measurement and Control*, vol. 97, no. 3, pp. 220-227, 1975.

[3] J. Hu, J. Pratt, and G. Pratt, "Stable adaptive control of a bipedalwalking; Robot with CMAC neural networks," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1050-1056, 1999.

[4] W. T. Miller and F. H. Glanz, "CMAC: An associative neural network alternative to backpropagation," *Proceedings of the IEEE, Special Issue on Neural Networks, II*, vol. 78, pp. 1561-1567, October, 1990.

[5] F. H. Glanz, W. T. Miller, and L. G. Kraft, "An overview of the CMAC neural network," *IEEE Conference on Neural Networks for Ocean Engineering, Washington, DC*, pp. 301-308, 1991.

[6] J. Hu and F. Pratt, "Self-organizing CMAC neural networks and adaptive dynamic control," *IEEE International Conference on Intelligent Control, Cambridge , MA*, pp. 15-17, September, 1999.

[7] A. Menozzi and M.-Y. Chow, "On the training of a multi-resolution CMAC neural network," *Proceedings of IECon'97, New Orleans, LA,*, vol. 3, pp. 1201-1205, 1997.

[8] J. Ozawa, I. Hayashi, and N. Wakami, "Formulation of CMAC-Fuzzy system," *IEEE international Conference on Fuzzy Systems, San Diego, CA,*, pp. 1179-1186, 1992.

[9] M. N. Nguyen, D. Shi, and C. Quek, "Self-Organizing Gaussian fuzzy CMAC with Truth Value Restriction," *Proceedings of IEEE International Conference of Information Technology and Applications (ICITA), Sydney, Australia.*, 2005.

[10] H. Xu, C. M. Kwan, L. Haves, and J. D. Pryor, "Real-time adaptive on-line traffic incident detection," *Fuzzy Sets and System*, pp. 173-183, 1998.

[11] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. New York: Plenum Press, 1981.

[12] W. L. Tung and C. Quek, "GenSoFNN: A Generic Self-Organizing Fuzzy Neural Network," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, September 2002.

[13] G. Leng, G. Prasad, and T. M. McGinnity, "An on-line algorithm for creating self-organizing fuzzy neural networks," *Neural Networks*, vol. 17, pp. 1477-1493, 2004.

[14] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759-771, 1991.

[15] H. M. Lee, C. M. Chen, and Y. F. Lu, "A self-organizing HCMAC neural-network classifier," *IEEE Transactions on neural networks*, vol. 14, no.1, pp. 15-27, 2003.

[16] L. Xu, "BYY harmony learning, structural RPCL, and topological self-organizing on mixture models," *Neural Networks*, vol. 15, pp. 1125-1151, 2002.

[17] L. Xu, "Advances on BYY harmony learning: information theoretic perspective, generalized projection geometry, and independent factor autodetermination," *IEEE Transactions on Neural Networks*, vol. 15, pp. 885-902, 2004.

[18] C. A. Pena-Reyes and M. Sipper, "Fuzzy CoCo: a cooperative-coevolutionary approach to fuzzy modeling," *Fuzzy Systems, IEEE Transactions on*, vol. 9, pp. 727-737, 2001.

[19] W.-Y. Wang and Y.-H. Li, "Evolutionary learning of BMF fuzzy-neural networks using a reduced-form genetic algorithm," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 33, pp. 966-976, 2003.

[20] M. A. Kbir, H. Benkirane, K. Maalmi, and R. Benslimane, "Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules," *Pattern Recognition Letters*, vol. 21, pp. 503-509, 2000.

[21] Y. Lin, G. A. Cunningham, III, and S. V. Coggeshall, "Using fuzzy partitions to create fuzzy systems from input-output data and set the initial weights in a fuzzy neural network," *IEEE Transactions on Fuzzy Systems*, vol. 5, pp. 614-621, 1997.

[22] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *in Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1988.

[23] N. R. Pal, K. Pal, and J. C. Bezdek, "A mixed c-means clustering model," *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 11-21 vol.1, 1997.

[24] H. Yin, R. Lengelle, and P. Gaillard, "Inverse-step competitive learning," presented at IEEE International Joint Conference on Neural Networks, 1991.

[25] C. T. Lin and C. S. G. Lee, "Neural Fuzzy Systems-A Neuro-Fuzzy Synergism to Intelligent Systems," Upper Saddle River, NJ: Prentice-Hall, 1996.

[26] L. A. Zadeh, "Calculus of fuzzy restrictions," *Fuzzy sets and Their Applications to Cognitive and Decision Processes, Edition: New York: Academic*, pp. 1-39, 1975.

[27] I. B. Turksen and Z. Zhong, "An approximate analogical reasoning schema based on similarity measures and interval-valued fuzzy sets," *Fuzzy Sets System*, vol. 34, pp. 323-346, 1990.

[28] R. Zwick, E. Carlstein, and D. V. Budescu, "Measures of similarity between fuzzy concepts: A comparative analysis," *International Journal of Approximate Reasoning*, vol. 1, pp. 221–242, 1987.

[29] C. T. Lin and C. S. G. Lee, "A Neuro-Fuzzy Synergism to Intelligent Systems," *Neural Fuzzy Systems, Upper Saddle River, NJ: Prentice-Hall*, 1996.

[30] C. Quek and R. W. Zhou, "POPFNN: A Pseudo Outer-Product Based Fuzzy Neural Network," *IEEE Transaction on Neural Networks*, vol. 9, pp. 1569-1581, 1996.

[31] K. K. Ang, C. Quek, and M. Pasquier, "POPFNN-CRI(S): Pseudo Outer Product based Fuzzy Neural Network using the Compositional Rule of Inference and Singleton Fuzzifier," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 33, 2003.

[32] C. Quek and R. W. Zhou, "POPFNN-AARS(S): A Pseudo Outer-Product Based Fuzzy Neural Network," *IEEE Transaction on Systems, Man & Cybernetics*, vol. 29, pp. 859-870, 1999.

[33] C. Darwin, *The origin of species by means of natural selection*. London: J. Murray, 1859.

[34] J. H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, Michigan: University of Michigan Press, 1975.

[35] S. Y. Chong and X. Yao, "Behavioral Diversity, Choices, and Noise in the Iterated Prisoner's Dilemma," *IEEE Transactions on Evolutionary Computation*, vol. 9, pp. 540-551, 2005.

[36] J. B. Pollack and A. D. Blair, "Co-evolution in the successful learning of Backgammon strategy," *Machine Learning*, vol. 32(3), pp. 225–240, 1998.

[37] K. Chellapilla and D. B. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE*, vol. 87(9), pp. 1471–1496, 1999.

[38] P. J. Darwen and J. B. Pollack, "Co-evolutionary learning on noisy tasks," *Proceedings of the Congress on Evolutionary Computation*, pp. 1724–1731, 1999.

[39] J. Herrmann, "A genetic algorithm for minimax optimization problems," *Proceedings of the Congress on Evolutionary Computation*, pp. 1099–1103, 1999.

[40] H. A. Mayer and R. Schwaiger, "Evolutionary and coevolutionary approaches to time series prediction using generalized multi-layer perceptrons," *Proceedings of the Congress on Evolutionary Computation*, pp. 275–280, 1999.

[41] J. R. Koza, *Genetic programming : on the programming of computers by means of natural selection*. Cambridge, Mass: MIT Press, 1992.

[42] T. Arciszewski and K. A. De Jong, "Evolutionary computation in civil engineering: research frontiers," presented at Proceedings of the Eight International Conference on Civil and Structural Engineering Computing, Eisenstadt, Vienna, Austria., 2001.

[43] K.-B. Sim, K.-S. Byun, and D.-W. Lee, "Design of fuzzy controller using schema coevolutionary algorithm," *Fuzzy Systems, IEEE Transactions on*, vol. 12, pp. 565-570, 2004.

[44] J. Maynard Smith, *Evolution and the theory of games*: Cambridge University Press, 1982.

[45] R. M. Axelrod, "Evolving new strategies: the evolution of strategies in the iterated prisoner's dilemma," presented at Genetic Algorithms and Simulated Annealing, San Mateo, CA, 1987.

[46] L. Pagie and M. Mitchell, "A comparison of evolutionary and coevolutionary search," *International Journal of Computational Intelligence and Applications*, vol. 2, pp. 53-69, 2002.

[47] J. Paredis, "Co-evolutionary constraints satisfaction," presented at Third International Conference on Parallel Problem Solving from Nature (PPSN-III), Jerusalem, Israel, 1994.

[48] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, pp. 1-29, 1997.

[49] N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez, "COVNET: a cooperative coevolutionary model for evolving artificial neural networks," *Neural Networks, IEEE Transactions on*, vol. 14, pp. 575-596, 2003.

[50] M. A. Potter and K. A. DeJong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, pp. 1-29, 2000.

[51] E. S. Lee and Q. Zhu, *Fuzzy and Evidence Reasoning*: Physica-Verlag, 1995.

[52] Z. Q. Wang, J. L. Schiano, and M. Ginsberg, "Hash-Coding in CMAC Neural Networks," *IEEE International Conference on Neural Networks*, vol. 3, pp. 1698 -1703, 1996.

[53] H. M. Lee, C. M. Chen, and Y. F. Lu, "A Self-organizing HCMAC Neural Network Classifier," *IEEE Transactions on Neural Network*, vol. 14, 2003.

[54] N. M. Laird, A. P. Dempster, and D. B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," *Journal of Royal Statistical Society*, vol. B39, pp. 1-38, 1977.

[55] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the em algorithm," *SIAM Rev.*, vol. 26, pp. 195-239, 1984.

[56] L. Xu, "Bayesian Ying Yang Learning (II): A New Mechanism for Model Selection and Regularization," *Intelligent Technologies for Information Analysis, N. Zhong and J. Liu (eds), Springer*, pp. 661-706, 2004.

[57] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 2(1), pp. 79–86, March, 1951.

[58] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann Eugenics 7, Part II*, pp. 179-188, 1936.

[59] J. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[60] C. J. Lin and C. T. Lin, "An ART-Based Fuzzy Adaptive Learning Control Network," *IEEE Transactions on Fuzzy Systems*, vol. 5, pp. 477-496, 1997.

[61] C. Quek and W. L. Tung, "A novel approach to the derivation of fuzzy membership functions using the Falcon-MART architecture," *Pattern Recognition Letters*, vol. 22, pp. 941-958, 2001.

[62] "Repository for bank data (Online). Available: Federal Reserve Bank of Chicago. URL-http://www.chicagofed.org."

[63] W. L. Tung, C. Quek, and P. Cheng, "GenSo-EWS: a novel neural-fuzzy based early warning system for predicting bank failures," *Neural Networks*, vol. 17, pp. 567-587, 2004.

[64] G. Alpaydin, G. Dundar, and S. Balkir, "Evolution-based design of neural fuzzy networks using self-adapting genetic parameters," *Fuzzy Systems, IEEE Transactions on*, vol. 10, pp. 211-221, 2002.

[65] J. Holland, *Adaptation in Natural and Artificial Systems*: MI press, 1975.

[66] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*. New York: Springer-Verlag, 1994.

[67] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," *Proceedings of Second International Conference on Genetic Algorithms*, pp. 14–21, 1987.

[68] H. Muhlenbein and D. Schlierkamp-Voosen, "Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization.," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 25-49, 1993.

[69] L. D. Miller, P. M. Long, L. Wong, S. Mukherjee, L. M. McShane, and E. T. Liu, "Optimal gene expression analysis by microarrays," *Cancer Cell*, vol. 2(5), pp. 353-61, 2002.

[70] M. E. Futschik, A. Reeve, and N. Kasabov, "Evolving connectionist systems for knowledge Discovery from gene expression data of cancer tissue," *Artificial Intelligence Medicine*, vol. 28(2), pp. 165-89, 2003.

[71] H. Liu and H. Motoda., *Feature selection for knowledge discovery and data mining*. Boston: Kluwer Academic Publishers, 1998.

[72] URL:http://www.affymetrix.com/index.affx.

[73] URL:http://www.stjuderesearch.org/data/ALL1/.

[74] K.H.Quah and C. Quek, "MCES: A Novel Monte Carlo Evaluative Selection Approach for Objective Feature Selections," *in press, IEEE Transactions on Neural Networks*, 2007.

[75] W. L. Tung and C. Quek, "GenSo-FDSS: a neural-fuzzy decision support system for pediatric ALL cancer subtype identification using gene expression data," *Arificial Intelligence in Medicine*, vol. 33, pp. 61-88, 2005.

[76] R. Andonie and L. Sasu, "Fuzzy ARTMAP with input relevances," *IEEE Transactions on Neural Networks*, vol. 17, pp. 929-941, 2006.

[77] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 31, pp. 902-918, 2001.

[78] D. Deng and N. Kasabov, "Evolving self-organizing maps for online learning, data analysis and modeling," presented at JCNN'2000 Neural Networks Neural Computing: New Challenges Perspectives New Millennium,, New York, 2000.

[79] S.-M. Chen and J. R. Hwang, "Temperature prediction using fuzzy time series," *IEEE Transactions on Systems, Man, Cybernetics-Part B*, vol. 30, pp. 263-275, 2000.

[80] K. Huarng, "Heuristic models of fuzzy time series for forecasting," *Fuzzy Sets and Systems*, vol. 123, pp. 369-386, 2001.

[81] M. J. Er and C. Deng, "Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning," *IEEE Trans on Systems, Man and Cybernetics, Part B*, vol. 34, pp. 1478-1489, 2004.

[82] C. F. Juang, "Combination of online clustering and Q-value based GA for reinforcement fuzzy systems," *IEEE Trans on Fuzzy Systems*, vol. 13, pp. 289-302, 2005.

[83] M. N. Nguyen, D. Shi, and C. Quek, "FCMAC-BYY: Fuzzy CMAC Using Bayesian Ying-Yang Learning," *IEEE Transactions on Systems, Man and Cybernetics-part B*, vol. 36, pp. 1180-1190, October, 2006.

[84] S. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generailized dynamic fuzzy neural networks," *IEEE Trans. Fuzzy Systems*, vol. 9, pp. 578-594, 2001.

[85] C.-T. Lin, C.-L. Chang, and W.-C. Cheng, "A recurrent fuzzy cellular neural network system with automatic structure and template learning," *Circuits and Systems I: Regular Papers, IEEE Transactions on [Circuits and Systems I:*

Fundamental Theory and Applications, IEEE Transactions on], vol. 51, pp. 1024-1035, 2004.

[86] W. L. Tung and C. Quek, "Falcon: neural fuzzy control and decision systems using FKP and PFKP clustering algorithms," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 34, pp. 686-695, 2004.

[87] C.-T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Transactions on Computers*, vol. 40, pp. 1320-1336, 1991.

[88] C. Quek, M. Pasquier, and B. B. S. Lim, "POP-TRAFFIC: a novel fuzzy neural approach to road traffic analysis and prediction," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, pp. 133-146, 2006.

[89] N. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transaction on Fuzzy Systems*, vol. 10, pp. 144-154, 2002.

[90] N. Kasabov, *Evolving Connectionist Systems*: Springer, 2003.

[91] M. N. Nguyen, J. F. Guo, and D. Shi, "ESOFCMAC: Evolving Self-Organizing Fuzzy Cerebellar Model Articulation Controller," presented at International Joint Conference on Neural Networks, 2006. IJCNN '06, 2006.

[92] S.-F. Su, T. Tao, and T.-H. Hung, "Credit assigned CMAC and its application to online learning robust controllers," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 33, pp. 202-213, 2003.

[93] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287-289, 1977.

[94] B. Fritzke, "A growing neural gas network learns topologies," *Adv.Neural Inform. Processing Syst*, vol. 7, 1995.

[95] J. Platt, "A resource allocating network for function interpolation," *Neural Comp*, vol. 3, pp. 213-225, 1991.

[96] G. K. Tan, "Feasibility of Predicting Congestion States with Neural Networks." Singapore: Nanyang Technological University, 1997.

[97] R. N. Goldman and J. S. Weinberg, *Statistics: An Introduction*: Englewood Cliffs, NJ: Prentice-Hall, 1985.

[98] W. L. Tung and C. Quek, *PACL-FNNS: a novel class of falcon-like fuzzy neural networks based on positive and negative exemplars*: In C. T. Leondes (Ed.), Intelligent systems: Technology and applications (Vol. II) (pp. 257–320). Fuzzy systems, neural networks and expert systems, Boca Raton: CRC Press, Chapter 10., 2002.

[99] C.E.Holt, "Forecasting trends and seasonal by exponentially weighted averages," *ONR Memorandum*, vol. 52, 1957.

[100] R. G. Brown, *Statistical forecasting for inventory control*. New York: McGraw-Hill, 1959.

[101] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Management Science*, vol. 6, pp. 324-342, 1960.

[102] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*: San Francisco: 2nd Edition, Holden Day, 1976.

[103] E. Parzen, "ARARMA models for time series analysis and forecasting," *Journal of Forecasting*, vol. 1, pp. 67-82, 1982.

[104] D. Bonnet, V. Perrault, and A. Grumbach, "Delta-NARMA neural networks: a connectionist extension of ARARMA models," presented at Proceedings of 5th European Symposium on Artificial Neural Networks (ESANN), 1997.

[105] S. Singh and E. Stuart, "A pattern matching tool for time-series forecasting," presented at Proceedings of the International Conference on Pattern Recognition, 1998.

[106] F. L. Chung, T. C. Fu, R. Luk, and V. Ng, "Evolutionary time series segmentation for stock data mining," presented at Proceedings of the IEEE International Conference on Data Mining (ICDM), 2002.

[107] J. Fu, J. Shi, and M. N. Nguyen, "An Early Warning System Based on Fuzzy CMAC," presented at proceedings of International Conference on Machine Learning and Cybernetics (ICMLC), HongKong, 2007.

[108] M. N. Nguyen, D. Shi, and C. Quek, "A Nature Inspired Ying-Yang Approach for Intelligent Decision Support in Bank Solvency Analysis," *Expert Systems With Applications*, 2007 (In press).

[109] F.-J. Lin, P.-K. Huang, and W.-D. Chou, "Recurrent-Fuzzy-Neural-Network-Controlled Linear Induction Motor Servo Drive Using Genetic Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 54, pp. 1449-1461, 2007.

[110] C.-H. Lee and C.-C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 8, pp. 349-366, 2000.

[111] P. A. Mastorocostas and J. B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification," *IEEE Transactions on Systems, Man and Cybernetics-part B*, vol. 32, pp. 176-190, 2002.

[112] B. Babcock, S.B. M. Datar, R. Motwani, and J. Widom, "Sampling from a Moving Window over Streaming Data," presented at ACM SIGMOD Symposium on Principles of Databases Systems, 2002.

[113] A. S. Jie Cai, Member IEEE and T. Kirubarajan, Senior Member IEEE, "EM-ML Algorithm for Track Initialization using Possibly Noninformative Data," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, 2005.