การเรียนรู้เพิ่มเติมแบบเร็วมากของเซลล์ประสาทเทียมสำหรับการแบ่งกลุ่มข้อมูลโดยใช้เพียงข้อมูลที่เข้ามาใหม่และฟังก์ชันไฮเพอร์อัลลิปซอยด์

นายสายชล ใจเย็น

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

A VERY FAST INCREMENTAL NEURAL LEARNING FOR CLASSIFICATION USING ONLY NEW INCOMING DATUM AND HYPER-ELLIPSOIDAL FUNCTION

Mr. Saichon Jaiyen

A Dissertation Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy Program in Computer Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2011

Thesis Title	A VERY FAST INCREMENTAL NEURAL LEARNING FOR CLASSIFICATION USING ONLY NEW INCOMING DATUM AND HYPER-ELLIPSOIDAL FUNCTION
By	Mr. Saichon Jaiyen
Field of Study	Computer Science
Thesis Advisor	Professor Chidchanok Lursinsap, Ph.D.
Thesis Co-advisor	Suphakant Phimoltares, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

…….. Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr.rer.nat.)

THESIS COMMITTEE

…….. Chairman
(Associate Professor Peraphon Sophatsathit, Ph.D.)

…….. Thesis Advisor
(Professor Chidchanok Lursinsap, Ph.D.)

…….. Thesis Co-advisor
(Suphakant Phimoltares, Ph.D.)

…….. Examiner
(Assistant Professor Saranya Maneeroj, Ph.D.)

…….. External Examiner
(Associate Professor Veera Boonjing, Ph.D.)

…….. External Examiner
(Chularat Tanprasert, Ph.D.)

สายชล ใจเย็น : การเรียนรู้เพิ่มเติมแบบเร็วมากของเซลล์ประสาทเทียมสำหรับการแบ่งกลุ่มข้อมูลโดยใช้เพียงข้อมูลที่เข้ามาใหม่และฟังก์ชันไฮเพอร์อัลลิปซอยด์ (A VERY FAST INCREMENTAL NEURAL LEARNING FOR CLASSIFICATION USING ONLY NEW INCOMING DATUM AND HYPER-ELLIPSOIDAL FUNCTION) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ศ.ดร. ชิดชนก เหลือสินทรัพย์, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม : อ.ดร. ศุภกานต์ พิมลธเรศ 69 หน้า.

งานวิจัยนี้นำเสนออัลกอริทึมการเรียนรู้แบบเร็วมากที่เรียนรู้ข้อมูลใหม่เพียงครั้งเดียวแล้วทิ้งข้อมูลนั้นโดยไม่ต้องใช้ข้อมูลเก่าที่เรียนรู้ไปแล้วด้วยการใช้ฟังก์ชันไฮเพอร์อัลลิปซอยด์ ฟังก์ชันชนิดนี้มีรูปทรงเป็นวงรีและสามารถเรียนรู้ข้อมูลหลายมิติได้โดยการล้อมข้อมูลที่เข้ามาตามการกระจายตัวของข้อมูล นอกจากนี้งานวิจัยนี้ยังเสนอโครงข่ายประสาทเทียมที่ใช้ฟังก์ชัน อัลลิปซอยด์เป็นฟังก์ชันในการเรียนรู้ โครงข่ายประสาทเทียมชนิดนี้มีชั้นซ่อนเพียงแค่ชั้นเดียวซึ่งจะถูกแบ่งเป็นชั้นซ่อนย่อยตามจำนวนกลุ่มของข้อมูลและจำนวนเซลล์ประสาทเทียมในชั้นซ่อนย่อยสามารถเพิ่มขึ้นได้ในระหว่างการเรียนรู้ โครงข่ายประสาทเทียมชนิดนี้สามารถเรียนรู้ข้อมูลได้โดยใช้เวลา O(n) เมื่อ n คือจำนวนข้อมูลสำหรับเรียนรู้ โครงข่ายประสาทเทียมชนิดนี้สามารถเรียนรู้ข้อมูลที่เข้ามาใหม่โดยไม่ต้องใช้ข้อมูลเก่าที่เรียนรู้ไปแล้ว ดังนั้นจึงไม่มีความจำเป็นที่จะต้องเก็บข้อมูลเก่าไว้เพื่อเรียนรู้ข้อมูลใหม่

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาการคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2554 ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม .....

4973891823 : MAJOR COMPUTER SCIENCE

KEYWORDS : CLASSIFICATION / FAST LEARNING / NEURAL NETWORK

SAICHON JAIYEN : A VERY FAST INCREMENTAL NEURAL LEARNING FOR CLASSIFICATION USING ONLY NEW INCOMING DATUM AND HYPER-ELLIPSOIDAL FUNCTION. ADVISOR: PROF. CHIDCHANOK LURSINSAP, Ph.D., CO-ADVISOR: SUPHAKANT PHIMOLTARES, Ph.D., 69 pp.

This research proposes a very fast 1-pass-throw-away learning algorithm based on a hyper-ellipsoidal function that can be translated and rotated to cover the data set during learning process. The translation and rotation of hyper-ellipsoidal function depends upon the distribution of the data set. In addition, the versatile elliptic basis function (VEBF) neural network with one hidden layer is proposed. The hidden layer of the proposed neural network is adaptively divided into subhidden layers according to the number of classes of the training data set. Each subhidden layer can be scaled by incrementing a new node to learn new samples during training process. The learning time is $O(n)$, where n is the number of data. The network can independently learn any new incoming datum without involving the previously learned data. Therefore, there is no need to store all previous data in order to mix with the new incoming data during the learning process.

Department : Mathematics and Computer Science Student's Signature .....

Field of Study : Computer Science Advisor's Signature .....

Academic Year : 2011 Co-advisor's Signature .....

Acknowledgements

I am greatly indebted to my advisor, Professor Dr. Chidchanok Lursinsap, for kindly providing guidance throughout the development of this dissertation. His comments greatly helped me in all the time of research work. What I know today about the good process of research, I have learned from him.

I am also grateful to my co-advisor, Dr. Suphakarn Pimoltares, who gives me wonderful suggestions in Ph.D. research.

I also extend my sincere gratitude and appreciation to Associate Professor Suchada Siripant for her guidance during my education.

I would like to express my gratitude to the Royal Golden Jubilee Ph.D. Program for their scholarship in supporting me during my education.

I would like to express my gratitude to King Mongkut's Institute of Technology Ladkrabang for their support during my education.

I am grateful to the Advanced Virtual and Intelligent Computing (AVIC) Center for their material support in enabling me to accomplish this research. I would also like to thank all my colleagues at the Advanced Virtual Intelligent Computing (AVIC) Center, who give me a lot of useful suggestions and encouragement.

I feel a deep sense of gratitude for my parents who taught me the good things that really matter in life and give me the love and encouragement.

Finally, I would like to thank all whose direct and indirect support helped me completing my dissertation.

Contents

Chapter Page

List of Tables

List of Figures

CHAPTER I

INTRODUCTION

1.1 Problem Identification

Currently, there are many alternative learning algorithms for neural networks used in several applications. All learning algorithms still cannot learn a new data set without mixing with the previously learned data set. All current training algorithms require both new incoming data and those previously trained data together in order to correctly learn the whole data set. Therefore, those previously trained data cannot be discarded after being learned. Some storage space must be wasted to keep these previously trained data. For this reason, the neural network cannot learn a new training data set if the old training data set is discarded. Moreover, many learning algorithms use so many epochs in learning process because of gradient method and optimization techniques. Furthermore, these algorithms consume much time when the very large data set is learned. Some methods solve these problems by modifying optimization process, but they still use a lot of epochs during learning process.

In this dissertation, the problems of learning new data without forgetting the previously learned data and learning new data from their new classes in only one pass are concentrated. In addition, the learning process is not required to access the previously learned data. To overcome these problems, a new neural network with incremental learning capacity and the new incremental learning algorithm that learn a data set in only one pass are developed. This developed learning algorithm is capable of learning new information without forgetting the previously learned information. Furthermore, this learning algorithm is able to accommodate new classes that may accompany with new data without forgetting the previously learned data and the redundant neurons in the neural network can be reduced during learning process.

1.2 Problem Formulation

Learning new data without accessing the previously learned data and learning new data without forgetting the previously learned data are the challenge problem in neural network research. The learning algorithm with both characteristics is called *incremental learning*. However, it is more useful if this algorithm can learn new data in only one pass so that the large data set can be learned. In this dissertation, these problems are concentrated. The problem statements in this dissertation can be formalized as:

1. How can the neural network learn new data without forgetting the previously learned data and this new data are discarded after being learned?

2. How can the neural network learn data set using in only one pass so that the learning time can be $O(n)$, where n is the number of data?

With these problems, the new neural network architecture and new learning algorithm should be created to overcome these problems.

1.3 Research Objective

The main objectives of this dissertation prospectus are:

1. To develop a new incremental learning algorithm to learn a data set in only one pass and the data are discarded after being learned

2. To develop a new neural network architecture with incremental learning capability that can learn new information without using the previously learned data.

1.4 Scope of Work

In this dissertation, the proposed learning process must be coped with the following capabilities:

1. It should be able to learn new data.

2. It should not require accession of the previously learned data.

3. It should preserve the previously learned data after learning new data.

1.5 Research Advantages

It is expected that the new learning algorithm has the following advantages.

1. This algorithm can be used to solve classification problems.

2. This algorithm can learn a data set in only one pass.

3. This algorithm can learn new data without using the previously learned data.

4. A storage space must not be wasted to keep the training data for further learning.

1.6 Outline of the Thesis

The remaining contents are organized as follows. Chapter II reviews the related literatures. Chapter III gives a review of some background knowledge. The new versatile elliptic basis function neural network and learning algorithm are proposed in Chapter IV. The experimental results are given in Chapter V. Chapter VI concludes the research.

CHAPTER II

LITERATURE REVIEW

A radial basis function neural networks is widely applied to several industrial problems such as face recognition, pattern recognition [1], [3], [6], pattern classification [2], time series prediction [4], [7] , and signal processing [5]. The standard RBF neural network consists of three layers. These are an input layer, a hidden layer, and an output layer. The learning algorithm of RBF neural network concerns the selection of the hidden layer neuron centers and estimation of the weights connecting the hidden layer and the output layers. Although the new learning algorithms [8], [9] are proposed continuously, most of the existing learning algorithms for RBF neural networks still use the Gaussian function as a basis function for adjusting the weight that is between a hidden unit to an output unit. The performance of this function depends on the selection of the center [10]. Some methods are primarily used for selecting the center such as k-mean, self-organizing map (SOM), and Gaussian mixture model. For k-mean and SOM, the number of centers cannot be predicted in advance. For the Gaussian mixture model, it is too difficult to compute model parameters. Such shortcomings directly affect the performance improvement.

An elliptical basis function neural network (EBF) [11] is an extension of a radial basis function neural network (RBF). The distinction between the elliptical basis function neural network and the radial basis function neural network is the covariance matrix. The radial basis function is the Gaussian function with diagonal covariance matrix as a basis function while the elliptical basis function is the Gaussian function with full covariance matrices. Although the elliptical basis function is widely applied to many problems [12], [13], [11], the basis function is still the Gaussian function with full covariance matrix. The parameters of this basis function are difficult to compute. Man-Wai Mak [11] proposed the expectation-maximization (EM) algorithm to estimate these parameters. This EM model is based on the gradient method which requires too many epochs to estimate the parameters. Jing Luo [14] applied EBF neural network to fault diagnosis of power transformer, but the

ellipsoidal function was used instead of the Gaussian function. Although this EBF can partition the input space, this ellipsoidal function cannot rotate to cover the data like the Gaussian function. In addition, the structure of both EBF and RBF is fixed during training and is not appropriate for the sequential learning.

Platt [17] proposed the sequential learning algorithm for RBF neural network in which hidden neurons are added sequentially based on the novelty of the new data. This neural network is called Resource Allocation Network (RAN). Kadirkamanathan and Niranjan [18] enhanced RAN using an extended Kalman filter (EKF) for updating the network parameters instead of least-mean square (LMS) algorithm, known as a RAN Extended Kalman Filter (RANEKF). The drawback of RAN and RANEKF is that the hidden neurons can grow up but they are never removed. Yingwei [19] proposed the improvement of RANEKF by introducing a pruning strategy called Minimal Resource Allocating Network (MRAN). Li Yan [20] proposed an improved version of the MRAN algorithm called Extended-MRAN (EMRAN) algorithm in which the parameters that are related to the selected winner neurons are updated by the EKF algorithm. The disadvantage of these proposed neural networks is that there are a lot of parameters chosen by trial and error. Although these sequential learning algorithms are able to learn new information, they are unable to accommodate new classes that may accompany new data.

Carpenter and Grossberg [21] propose a neural network architecture for incremental supervised learning of analog multidimensional maps called fuzzy ARTMAP. The Fuzzy ART neural network composed of two fuzzy Adaptive Resonance Theory (ART) modules denoted ARTa and ARTb. Each of the fuzzy ART modules consists of three neural layers: preprocessing F0; matching F1, and competitive F2. These two fuzzy ART modules are interconnected by map field, Fab, which forms an association between ARTa and ARTb. The inputs are presented at the ARTa module while the corresponding outputs are presented at the ARTb module. The fuzzy ARTa module performs clustering in the input space of data while the fuzzy ARTb performs clustering in the output space of the target data. The map field determines whether the mapping between the inputs and the outputs is the correct one. The drawback of fuzzy ARTMAP neural network is that once a node is created in the network, the node can never be removed. For this reason, it may suffer from the greedy insertion strategy that leads to a complex network structure. Robi Polikar [22] proposed an

algorithm for incremental training of neural network pattern classifiers called Learn++. The proposed algorithm enables supervised neural networks such as the multilayer perceptron (MLP) to accommodate new data including examples that correspond to previously unseen classes. In addition, the algorithm does not require access to previously used data during subsequent incremental learning sessions and it does not forget previous learned knowledge. Learn++ uses ensemble of classifiers by generating multiple hypotheses using training data sampled according to carefully tailored distributions. The outputs of the resulting classifiers are combined using a weighted majority voting procedure. However, Learn++ suffers from the inherent *out-voting* problem when asked to learn new classes, causing it to generate an unnecessarily large number of classifiers. Mu-Chun Su [23] proposed a new approach to incrementally construct a neural network that is capable of learning new information without forgetting old knowledge. The proposed neural network, called Hyper-Spherical ARTMAP network (HS-ARTMAP network), is a synthesis of an RBF-network-like module and an ART-like module. The HS-ARTMAP network is trained via a training algorithm similar to the training algorithm for the fuzzy ARTMAP system.

In this dissertation, a very fast training algorithm to learn a data set in only one pass is proposed. Once a datum is learned, it is discarded. There is no need to use this datum again for the future learning with new incoming data. The structure of proposed neural network consists of three layers like RBF and EBF but the structure is flexible and can be adjusted during the training process.

CHAPTER III

THEORETICAL BACKGROUND

In this dissertation, the idea of radial basis function in the interpolation problem is applied to develop a new elliptic basis function. In addition, the idea of the interpolation problem is also applied to develop radial basis function neural network. However, in this dissertation, the idea of radial basis function neural network is applied to develop a new neural network architecture. Furthermore, some knowledge of geometrical and some knowledge of vector space such as orthonormal basis are combined in order to develop the new elliptic basis function that can be translated and rotated to cover the new data in high dimensional space. The principal component analysis is also applied to approximate the orthonormal basis for a data space. Therefore, the details of these theoretical backgrounds are described in the following sections.

3.1 Interpolation Problem

Interpolation problem is currently one of the principal fields of research in numerical analysis. There are several approaches for solving the interpolation problem including a radial basis function technique. Radial basis functions were first applied by Powell to solve this interpolation problem. The interpolation problem can be stated as follows.

Given a set of *N* different point $\{\mathbf{x}_k \in \mathbb{R}^m | k = 1, 2, ..., N\}$ and a corresponding set of *N* real numbers $\{d_k \in \mathbb{R} | k = 1, 2, ..., N\}$, find a function $F : \mathbb{R}^m \to \mathbb{R}$ that satisfies the interpolation codition:

$$F(\mathbf{x}_k) = d_k, \quad k = 1, 2, ..., N \tag{3.1}$$

The radial-basis functions technique consists of choosing a function F that has the following form (Powell, 1988):

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi \left(\|\mathbf{x} - \mathbf{x}_i\| \right) \tag{3.2}$$

where w_i is a real coefficient, $\{\varphi(\|\mathbf{x}-\mathbf{x}_i\|), \quad i=1,2,...,N\}$ is a set of N random (usually nonlinear) functions, known as *radial basis functions*, and $\|\cdot\|$ represents a norm that is generally Euclidean. The known data vectors, $\mathbf{x}_i \in \mathbb{R}^m, i=1,2,...,N$, are the centers of radial basis functions.

Substituting the interpolation conditions of Equation (3.1) into Equation (3.2) yield,

$$\sum_{i=1}^{N} w_i \varphi(\|\mathbf{x}_k-\mathbf{x}_i\|) = d_k, \quad k=1,2,...,N \tag{3.3}$$

The system of equations (3.3) can be rewritten in a matrix form as follows:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \tag{3.4}$$

where

$$\varphi_{ji} = \varphi(\|\mathbf{x}_j-\mathbf{x}_i\|), \quad i,j=1,2,...,N \tag{3.5}$$

Let

$$\begin{aligned} \mathbf{d} &= [d_1,\ d_{2,\ \dots\ ,}\ d_N]^T \\ \mathbf{w} &= [w_1,\ w_{2,\ \dots\ ,}\ w_N]^T \\ \Phi &= [\varphi_{ij}]_{N\times N} \end{aligned}$$

Equation (3.4) is then rewritten compactly as follows.

$$\Phi\mathbf{w} = \mathbf{d} \tag{3.6}$$

Assuming that Φ is nonsingular, we can solve Equation (3.6) for the weight vector $\mathbf{w}$ as follows.

$$\mathbf{w} = \Phi^{-1}\mathbf{x} \tag{3.7}$$

Micchelli's Theorem Let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of distinct points in $\mathbb{R}^m$. Then the *N-by-N* interpolation matrix Φ, whose *ji-th* element is $\varphi_{ji} = \varphi(\|\mathbf{x}_j-\mathbf{x}_i\|)$, is nonsingular.

There is a large class of radial-basis functions that is covered by Micchelli's theorem. These classes of functions studied in the RBF network include the following functions.

1. Gaussian function:

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right),\ \sigma > 0;\ x, c \in \mathbb{R} \tag{3.8}$$

2. Multiquadrics:

$$\varphi(r) = \left(x^2 + c^2\right)^{1/2},\ c > 0;\ x, c \in \mathbb{R} \tag{3.9}$$

3. Inverse Multiquadrics:

$$\varphi(r) = \frac{1}{\left(x^2 + c^2\right)^{1/2}},\ c > 0;\ x, c \in \mathbb{R} \tag{3.10}$$

3.2 Radial Basis Function Networks

From Equation (3.1) through (3.7), a radial basis function (RBF) network can be constructed in form of a layered structure as illustrated in Figure 3.1. The RBF network consists of three layers detailed as follows.

1. Input layer consists of m source nodes where m is the dimension of the input vector.

2. Hidden layer consists of the computational units known as *hidden neurons*. The number of hidden neurons in the hidden layer is equal to the number of the training samples. Each hidden neurons in the hidden layer is mathematically described by radial basis function as follows.

$$\varphi_j(\mathbf{x}) = \varphi\left(\|\mathbf{x} - \mathbf{x}_j\|\right), \quad j = 1, 2, ..., N$$

   The data vector $\mathbf{x}_j$ is the center of radial basis function. The vector $\mathbf{x}$ is the input signal which is fed into the input layer.

3. Output layer consists of a single computational unit known as *output neuron*. The output neuron in the output layer is mathematically described by the following function,

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi\left(\|\mathbf{x} - \mathbf{x}_i\|\right)$$

   where $\varphi(\cdot)$ is the radial basis function and N is the number of training data.

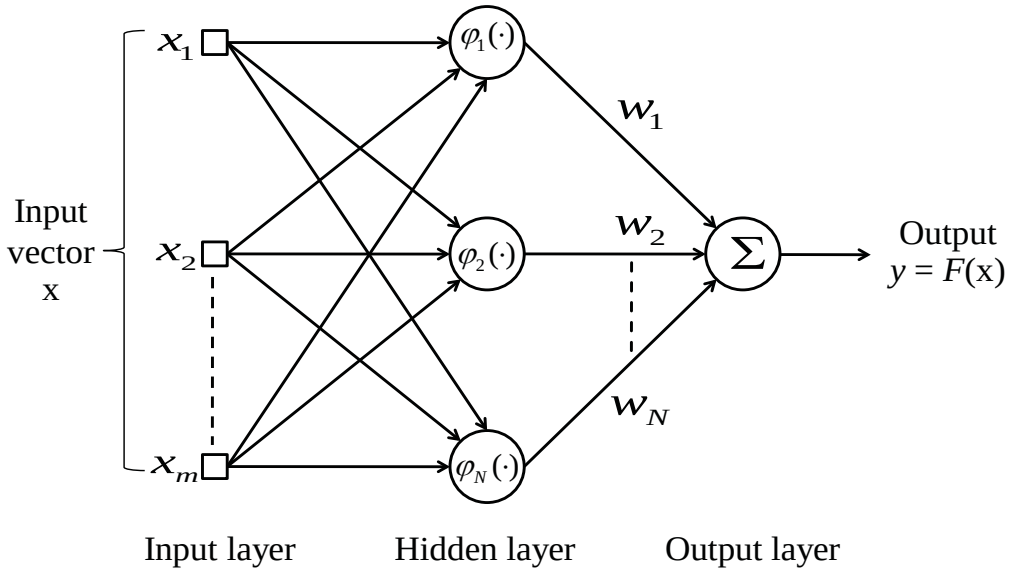


Figure 3.1 Structure of RBF network.

The popular radial basis function used in RBF networks is a Gaussian function defined as

$$\begin{aligned} \varphi_j(\mathbf{x}) &= \varphi(\mathbf{x}-\mathbf{x}_j) \\ &= \exp\left(-\frac{1}{2\sigma_j^2}\|\mathbf{x}-\mathbf{x}_j\|^2\right), \quad j=1,2,...,N \end{aligned} \tag{3.11}$$

where σ_j is a measure of the *width* of the *j*th Gaussian function with center $\mathbf{x}_j$.

3.3 Practical Modifications to the RBF Network

The design of RBF network illustrated in Figure 3.1 via interpolation theory is not practical in several applications when the number of training data is very large. Since the number of hidden neurons in the RBF network of Figure 3.1 is equal to the number of the training data, the computational cost of the network is very expensive. To reduce the computational cost of the network, the interpolation problem with K basis functions which is less than N data points is considered instead. This interpolation problem can be done by

modifying Equation (3.1) as follows.

$$F(\mathbf{x}) = \sum_{i=1}^{K} w_i \varphi\left(\|\mathbf{x} - \mathbf{x}_i\|\right) \tag{3.12}$$

The solution for the weight vector **w** in Equation (3.7) can be computed as follows.

$$\begin{aligned} \mathbf{w} &= \left(\Phi^T \Phi\right)^{-1} \Phi^T \mathbf{d} \\ &= \Phi^{+} \mathbf{d} \end{aligned} \tag{3.13}$$

The matrix Φ^{+} in Equation (3.13) is the *pseudo inverse* of matrix Φ defined as,

$$\Phi^{+} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \tag{3.14}$$

From Equation (3.12) through Equation (3.14), the practical RBF network can be constructed as illustrated in Figure 3.2.

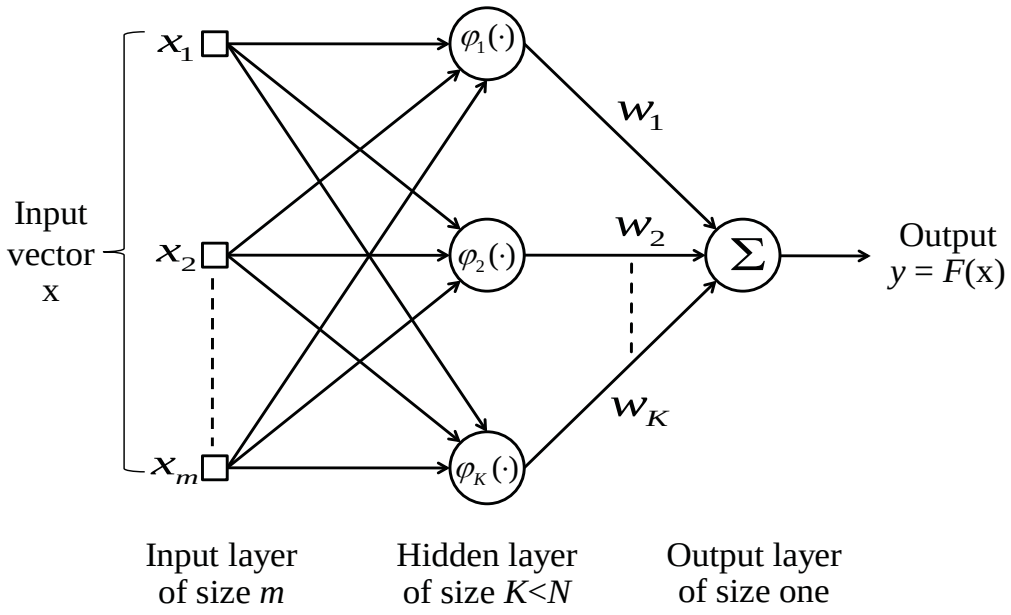


Figure 3.2 Structure of a practical RBF network.

The structure of this RBF network consists of three layers, an input layer, a hidden layer, and an output layer. The RBF network receives the input from the input layer and

transfers this input to the hidden layer. There is not the synaptic weight between the input layer and hidden layer. The hidden neuron receives the input vector from the input layer, and calculates the Euclidean distance between the center of radial basis function and the input vector of the network. Subsequently, the result is passed to the radial basis function. Therefore, the hidden layer performs a nonlinear transformation and maps the input space onto a new space call *feature space*. The output layer performs a linear combiner on this new space as follows.

$$F(\mathbf{x}) = \sum_{i=1}^{K} w_i \varphi \left(\|\mathbf{x} - \mathbf{x}_i\| \right) \tag{3.15}$$

3.4 Principal Components Analysis

The principal component analysis (PCA) is a statistical method applied primarily to transform the input data space into a new lower dimensional space. Let **X** be an *m*-dimensional random vector. The mean of random vector **X** is assumed to be zero. This can be written as:

$$\mathbb{E}(\mathbf{X}) = 0$$

where $\mathbb{E}$ is the statistical expectation operator. Let **q** be a *unit vector*, also of dimension *m*, onto which the vector **X** is to be projected. This projection is defined by the inner product of the vectors **X** and **q** as follows.

$$A = \mathbf{X}^T \mathbf{q} = \mathbf{q}^T \mathbf{X} \tag{3.16}$$

subject to the constraint

$$\|\mathbf{q}\| = (\mathbf{q}^T \mathbf{q})^{1/2} = 1 \tag{3.17}$$

The projection A is a random variable with mean and variance related to the statistics of the random variable **X**. From the assumption that the random vector **X** has zero mean, it implies that the random vector A has zero mean too. The variance of A can be written as

follows.

$$\begin{aligned} \sigma^2 &= \mathbb{E}[A^2] \\ &= \mathbb{E}[(\mathbf{q}^T\mathbf{X})(\mathbf{X}^T\mathbf{q})] \\ &= \mathbf{q}^T\mathbb{E}[\mathbf{X}\mathbf{X}^T]\mathbf{q} \\ &= \mathbf{q}^T\mathbf{R}\mathbf{q} \end{aligned} \tag{3.18}$$

The m-by-m matrix $\mathbf{R}$ is the *correlation matrix* of the random vector $\mathbf{X}$, formally defined as the expectation of the outer product of the vector $\mathbf{X}$ with itself as follows.

$$\mathbf{R} = \mathbb{E}[\mathbf{X}\mathbf{X}^T] \tag{3.19}$$

From Equation (3.18), the variance σ^2 of the projection A is a function of the unit vector $\mathbf{q}$ as shown in the following equation.

$$\begin{aligned} \Psi(\mathbf{q}) &= \sigma^2 \\ &= \mathbf{q}^T\mathbf{R}\mathbf{q} \end{aligned} \tag{3.20}$$

where $\Psi(\mathbf{q})$ is a *variance probe*.

Eigenstructure of Principal Components Analysis

The next problem is how to find the unit vectors $\mathbf{q}$ along which $\Psi(\mathbf{q})$ has extremal values, subject to a constraint on the Euclidean norm of $\mathbf{q}$. To overcome this problem, the vector $\mathbf{q}$ is assumed to be a unit vector such that the variance probe $\Psi(\mathbf{q})$ has an extremal value. Therefore, for any small perturbation $\delta\mathbf{q}$ of unit vector $\mathbf{q}$, it is seen that

$$\Psi(\mathbf{q} + \delta\mathbf{q}) = \Psi(\mathbf{q}) \tag{3.21}$$

From the definition of the variance probe given in Equation (3.20), it can be written as

$$\begin{aligned} \Psi(\mathbf{q} + \delta\mathbf{q}) &= (\mathbf{q} + \delta\mathbf{q})^T\mathbf{R}(\mathbf{q} + \delta\mathbf{q}) \\ &= \mathbf{q}^T\mathbf{R}\mathbf{q} + 2(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} + (\delta\mathbf{q})^T\mathbf{R}\delta\mathbf{q} \end{aligned}$$

The second-order term $(\delta\mathbf{q})^T\mathbf{R}\delta\mathbf{q}$ can be ignored because its value is very mall, and then obtain

$$\begin{aligned} \Psi(\mathbf{q} + \delta\mathbf{q}) &= \mathbf{q}^T\mathbf{R}\mathbf{q} + 2(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} \\ &= \Psi(\mathbf{q}) + 2(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} \end{aligned} \tag{3.22}$$

Substituting Equation (3.21) in (3.22) yields

$$(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} = 0 \tag{3.23}$$

The perturbations are performed under the restriction that the Euclidean norm of the perturbed vector $\mathbf{q} + \delta\mathbf{q}$ remains equal to unity. That is

$$\|\mathbf{q} + \delta\mathbf{q}\| = 1$$

or equivalently,

$$(\mathbf{q} + \delta\mathbf{q})^T(\mathbf{q} + \delta\mathbf{q}) = 1$$

Using Equation (3.17), it can be written as

$$(\delta\mathbf{q})^T\mathbf{q} = 0 \tag{3.24}$$

This means that the perturbations $\delta\mathbf{q}$ must be orthogonal to $\mathbf{q}$, and therefore only a change in the direction of $\mathbf{q}$ is permitted.

Scaling Equation (3.24) with $-\lambda$ and add to Equation (3.23) yields

$$(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} - \lambda(\delta\mathbf{q})^T\mathbf{q} = 0$$

or equivalently,

$$(\delta\mathbf{q})^T(\mathbf{R}\mathbf{q} - \lambda\mathbf{q}) = 0 \tag{3.25}$$

From Equation (3.25), it is sufficient to have

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q} \tag{3.26}$$

This is the equation that governs the unit vectors $\mathbf{q}$ for which the variance probe $\Psi(\mathbf{q})$ has extremal values. The Equation (3.26) is known as the eigenvalue problem, commonly found in linear algebra. The value of λ is called the eigenvalue of the correlation matrix $\mathbf{R}$ and the associated value of $\mathbf{q}$ is called the eigenvector. Let the eigenvalues of the m-by-m matrix $\mathbf{R}$ be denoted by $\lambda_1, \lambda_2, ..., \lambda_m$ and the associated eigenvectors be denoted by $\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_m$, respectively. Thus, for $j = 1, 2, ..., m$, it can be written as

$$\mathbf{R}\mathbf{q}_j = \lambda_j\mathbf{q}_j \tag{3.27}$$

Let the corresponding eigenvalues be arranged in decreasing order:

$$\lambda_1 > \lambda_2 > ... > \lambda_j > ... > \lambda_m \tag{3.28}$$

Let the associated eigenvectors be constructed as an m-by-m matrix:

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_j, ..., \mathbf{q}_m] \tag{3.29}$$

The set of m equations of Equation (3.27) can be combined into a single equation as follows.

$$\mathbf{RQ} = \mathbf{Q}\Lambda \tag{3.30}$$

where Λ is a diagonal matrix defined as follows.

$$\Lambda = diag[\lambda_1, \lambda_2, ..., \lambda_j, ..., \lambda_m] \tag{3.31}$$

The matrix **Q** is an orthogonal matrix that satisfies the conditions of orthogonality:

$$\mathbf{q}_i^T \mathbf{q}_j = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases} \tag{3.32}$$

Equivalently, it can be written as

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$$

It can be deduced that the inverse of matrix **Q** is equal to its transpose as follows.

$$\mathbf{Q}^T = \mathbf{Q}^{-1} \tag{3.33}$$

This means that the Equation (3.30) can be rewritten in a form of the *orthogonal similarity transformation*:

$$\mathbf{Q}^T\mathbf{RQ} = \Lambda \tag{3.34}$$

or in expanded form,

$$\mathbf{q}_j^T \mathbf{R} \mathbf{q}_k = \begin{cases} \lambda_j, & k = j \\ 0, & k \neq j \end{cases} \tag{3.35}$$

From Equation (3.20) and (3.35), it implies that the variance probes and eigenvalues are indeed equal, as shown by

$$\Psi(\mathbf{q}_j) = \lambda_j, \quad j = 1, 2, ..., m \tag{3.36}$$

The two important properties that have found from the eigenstructure of principal components analysis can be summarized as follows.

- The eigenvectors of the correlation matrix **R** pertaining to the zero-mean random vector **X** define the unit vectors $\mathbf{q}_j$, representing the principal directions along which the variance probes $\Psi(\mathbf{q}_j)$ have their extremal values.
- The associated eigenvalues define the extremal values of the variance probes $\Psi(\mathbf{q}_j)$.

3.5 Vector Spaces

In mathematics, a **vector space** is defined as a set V that is closed under two algebraic operations called **vector addition** and **scalar multiplication**. These two algebraic operations must satisfy certain properties. To determine whether a set of objects is a vector space depends on whether the set is closed under vector addition and scalar multiplication and satisfies certain properties. These properties, the *axioms* of a vector space, are given as follows.

Definition 3.1: Vector Space
Let V be a set of elements on which two operations called **vector addition** and **scalar multiplication** are defined. Then V is said to be a **vector space** if the following ten properties are satisfied.

Axioms for Vector Addition:

(i) If **u** and **v** are in V, then **u**+**v** is in V

(ii) For all **u**, **v** in V, **u**+**v** = **v**+**u**

(iii) For all **u**, **v**, **w** in V, **u**+(**v**+**w**) = (**v**+**u**)+**w**

(iv) There is a unique vector **0** in V such that **0**+**u** = **u**+**0** = **0**

(v) For each **u** in V, there exists a vector -**u** such that **u**+(-**u**) = (-**u**)+**u** = **0**

Axioms for Scalar Multiplication:

(vi) If k is any scalar and **u** is in V, then k**u** is in V

(vii) k(**u**+**v**) = k**u**+k**v**

($viii$) (k_1+k_2)**u** = k_1**u**+k_2**u**

(ix) k_1(k_2**u**) = (k_1k_2)**u**

(x) **1u** = **u**

The scalars in Definition 1 may be taken from real numbers. In this case V is referred to as a *real* vector space. When the scalars are allowed to be complex numbers, it obtains a *complex* vector space. Since $\mathbb{R}^2$ has the properties in Definition 1, it is clear that $\mathbb{R}^2$ is a vector space. Moreover, since vectors in $\mathbb{R}^3$ and $\mathbb{R}^n$ have these same properties, it can be concluded that $\mathbb{R}^3$ and $\mathbb{R}^n$ are also vector spaces.

Definition 3.2: Linear Independence
A set of vector $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ is said to be **linearly independent** if the only constants satisfy the equation

$$k_1\mathbf{u}_1 + k_2\mathbf{u}_2 + \cdots + k_n\mathbf{u}_n = \mathbf{0} \tag{3.37}$$

are $k_1 = k_2 = \cdots = k_n = 0$. If the set of vectors is not linearly independent, then it is said to be **linearly dependent**.

In $\mathbb{R}^3$, the vector **i** = $[1, 0, 0]^T$, **j**=$[0, 1, 0]^T$, and **k** = $[0, 0, 1]^T$ are linearly independent. Since the equation $k_1\mathbf{i} + k_2\mathbf{j} + k_3\mathbf{k} = \mathbf{0}$, by equality of vectors, it can be concluded that $k_1 = 0$, $k_2 = 0$, and $k_3 = 0$. Furthermore, any vector in $\mathbb{R}^3$ can be written as a linear combination of the linearly independent vectors **i**, **j**, and **k**. The set of vectors $\{$**i**, **j**, **k**$\}$ forms a **basis** for $\mathbb{R}^3$.

Definition 3.3: Basis for a Vector Space
Let's consider a set of vectors $B = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ in a vector space V. If the set B is linearly independent and if every vector in V can be expressed as a linear combination of these vectors, then B is said to be a **basis** for V.

A vector space may have many bases. It is mentioned previously that the set of vectors $\{$**i**, **j**, **k**$\}$ is a basis for $\mathbb{R}^3$ but it can be proved that $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$, where

$$\mathbf{u}_1 = [1, 0, 0]^T, \mathbf{u}_2 = [1, 1, 0]^T, \mathbf{u}_3 = [1, 1, 1]^T$$

is a linearly independent set.
Proof Let k_1, k_2, and k_3 be three real numbers such that

$$k_1[1, 0, 0]^T + k_2[1, 1, 0]^T + k_3[1, 1, 1]^T = \mathbf{0}$$

Therefore,

$$\begin{aligned} k_1 + k_2 + k_3 &= 0 \\ k_2 + k_3 &= 0 \\ k_3 &= 0 \end{aligned}$$

The solution to this system is, $k_1 = 0$, $k_2 = 0$, and $k_3 = 0$. So the set of vectors $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ is linearly independent.

Furthermore, every vector $\mathbf{a} = (a_1, a_2, a_3)$ can be written as a linear combination $\mathbf{a} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + c_3\mathbf{u}_3$. Therefore, the set of vectors $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ is another basis for $\mathbb{R}^3$. However, the set of vectors $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ is referred to as the **standard basis** for $\mathbb{R}^3$. For the vector space $\mathbb{R}^n$, the standard basis consists of the n vectors

$$\mathbf{e}_1 = [1, 0, 0, \ldots, 0]^T, \mathbf{e}_2 = [0, 1, 0, \ldots, 0]^T, \ldots, \mathbf{e}_n = [0, 0, 0, \ldots, 1]^T.$$

If B is a basis for a vector space V, then for every vector $\mathbf{v}$ in V there exists scalars c_i, $i = 1, 2, \ldots, n$ such that

$$\mathbf{v} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_n\mathbf{u}_n. \tag{3.38}$$

The scalars c_i, $i = 1, 2, \ldots, n$, in the linear combination (3.38) are called **coordinates of v relative to the basis** B.

Definition 3.4: Dimension of a Vector Space
The number of vectors in a basis B for a vector space V is said to be the **dimension** of the space.

Every vector in the vector space $\mathbb{R}^n$ can be expressed as a linear combination of the vectors in the standard basis $B = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$, where

$$\mathbf{e}_1 = [1, 0, 0, \ldots, 0]^T, \mathbf{e}_2 = [0, 1, 0, \ldots, 0]^T, \ldots, \mathbf{e}_n = [0, 0, 0, \ldots, 1]^T.$$

This standard basis $B = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ is an example of an **orthonormal basis**, that is, each $\mathbf{e}_i, where i = 1, 2, \ldots, n$, is mutually orthogonal to one another and is a unit vector, that is,

$$\mathbf{e}_i \cdot \mathbf{e}_j = 0, i \neq j \text{ and } \|\mathbf{e}_i\| = 1, i = 1, 2, \ldots, n.$$

Definition 3.5: Orthonormal Basis

Let V be a finite dimensional vector space with an inner product. A set of basis vectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ for V is called an *orthonormal basis* if

(i) $\mathbf{u}_i \cdot \mathbf{u}_j = 0$ for $i \neq j$

(ii) $\|\mathbf{u}_j\| = 1$ for $j = 1, 2, \ldots, n$

Theorem 3.1: Coordinates Relative to an Orthonormal Basis

Suppose $B = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ is an orthonormal basis for $\mathbb{R}^n$. if **v** is any vector in $\mathbb{R}^n$, then

$$\mathbf{v} = (\mathbf{v} \cdot \mathbf{u}_1)\mathbf{u}_1 + (\mathbf{v} \cdot \mathbf{u}_2)\mathbf{u}_2 + \cdots + (\mathbf{v} \cdot \mathbf{u}_n)\mathbf{u}_n. \tag{3.39}$$

Proof The vector **v** is in $\mathbb{R}^n$ and B is the basis for $\mathbb{R}^n$. Thus, there exists real scalars k_i, $i = 1, 2, \ldots, n$, such that **v** can expressed as the linear combination

$$\mathbf{v} = k_1\mathbf{u}_1 + k_2\mathbf{u}_2 + \cdots + k_n\mathbf{u}_n. \tag{3.40}$$

The scalar k_i are coordinates of **v** relative to the basis B. These coordinates can be found by taking the dot product of **v** with each of the basis vectors:

$$\begin{aligned} \mathbf{v} \cdot \mathbf{u}_i &= (k_1\mathbf{u}_1 + k_2\mathbf{u}_2 + \cdots + k_n\mathbf{u}_n) \cdot \mathbf{u}_i \\ &= k_1(\mathbf{u}_1 \cdot \mathbf{u}_i) + k_2(\mathbf{u}_2 \cdot \mathbf{u}_i) + \cdots + k_n(\mathbf{u}_n \cdot \mathbf{u}_i) \end{aligned} \tag{3.41}$$

Since B is orthonormal, $\mathbf{u}_i \cdot \mathbf{u}_j = 0$ for $i \neq j$ and $\mathbf{u}_i \cdot \mathbf{u}_i = \|\mathbf{u}_i\|^2 = 1$. Hence from Equation (3.41) obtains $k_i = (\mathbf{v} \cdot \mathbf{u}_i)$ for $i = 1, 2, \ldots, n$.

CHAPTER IV

MATHEMATICAL MODEL AND ALGORITHM

In the previous Chapter, some knowledge background related to radial basis function, radial basis function neural network, principal component analysis, and orthonormal basis for a vector space are described in a particular detail. In this Chapter, this knowledge is applied to develop a new learning algorithm and a new neural network architecture. The developing details of these new learning algorithm and new neural network architecture are explained in the following sections. In section 4.1, the details of the proposed neural network architecture are described. Section 4.2 describes how to develop the new ecliptic basis function that can translate and rotate to cover the new data in high dimensional space, and how to compute the parameters of this new developed function such as center, covariance matrix, and orthonormal basis. In Section 4.3, the details of the proposed learning algorithm are described. The main details are how to add a new neuron to the neural network, how to compute the parameters of this new neuron, how to merge the two redundant neurons into a new one neuron, and how to compute the merging parameters of this new neuron. The learning time of this proposed algorithm is also proved to be $O(n)$, where n is the number of the training data. Examples of the proposed learning algorithm are illustrated in order to simplify the understanding.

4.1 Versatile Elliptic Basis Function Neural Network

The structure of the versatile elliptic basis function neural network (VEBF neural network) is shown in Figure 4.1. The network consists of an input layer, a hidden layer, and an output layer. In the input layer, the number of nodes in this layer is equal to the dimension of the input data space. In the hidden layer, the nodes in this layer are separated into sub-hidden layers whose number of sub-hidden layers is equal to the number of classes and all neurons in the same sub-hidden layer are added to cover the training data in the same

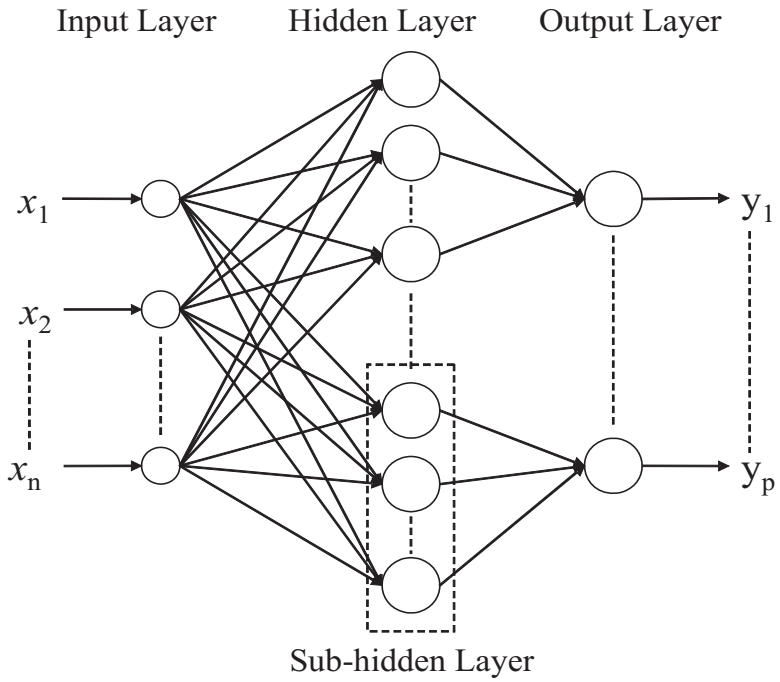


Figure 4.1 Structure of Versatile Elliptic Basis Function neural network.

class as explained in the next section. Furthermore, the nodes in each sub-hidden layer can be automatically increased during the learning process. In the output layer, the number of nodes in this layer is equal to the number of classes in the training data set. Moreover, the nodes in the output layer can be automatically increased to learn a new data from new unseen classes during the learning process.

Initially, there is no node in the network. The nodes in each layer in the network are automatically increased depending on some conditions during the learning process. Suppose that the input layer consists of n nodes and the output layer consists of p nodes. For each given input vector $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ in $\mathbb{R}^n$, the output of the k^{th} hidden neuron can be calculated from the versatile elliptic basis function as follows.

$$\psi_k(\mathbf{x}) = \sum_{i=1}^{n} \frac{((\mathbf{x} - \mathbf{c}_k)^T \mathbf{u}_i^k)^2}{(a_i^k)^2} - 1 \tag{4.1}$$

where $\left\{\mathbf{u}_1^k, \mathbf{u}_2^k, ..., \mathbf{u}_n^k\right\}$ is the orthonormal basis of the k^{th} hidden neuron and the constant a_i^k is the i^{th} *semi-axis length* of the versatile elliptic basis function of the k^{th} hidden neuron. The vector $\mathbf{c}_k = [c_1, c_2, ..., c_n]^T$ is the center of the versatile elliptic basis function of the k^{th} hidden neuron as explained in the next section.

In the output layer, the output at the q^{th} output neuron is obtained from finding a minimum value from the hidden neurons in the q^{th} sub-hidden layer. The output at the q^{th} output neuron in the output layer is defined as

$$y_q(\mathbf{x}) = \min_k(\psi_k(\mathbf{x})), q = 1, ..., p \tag{4.2}$$

where $\psi_k(\mathbf{x})$ is the output at the k^{th} hidden node in the q^{th} sub-hidden layer.

To predict the class label of the input vector **x**, the *decision function* for predicting the class label of a new input vector is defined. The value of this decision function is obtained from the minimum output value among the output neurons generated by a hidden neuron that is covered by input vector **x**. The decision function, $D(\mathbf{x})$, is defined as follows.

$$D(\mathbf{x}) = \begin{cases} k & \text{if } k = \arg\min_q(y_q(\mathbf{x})) \text{ and } y_q(\mathbf{x}) \leq 0 \\ unknown & \text{otherwise} \end{cases} \tag{4.3}$$

From Equation (4.3), it is clear that the input vector **x** is predicted to be the k^{th} class if there exists a hidden neuron in the k^{th} sub-hidden layer covering the input vector **x**. If there is no hidden neuron covering the input vector **x**, it is predicted to be the *unknown* class. Therefore, for any new input vector **x**, the new input vector can be considered as either k^{th} class or *unknown* class. However, the input vectors in the *unknown* class may be classified into the k^{th} class by using only the minimum value from the output layer.

4.2 Versatile Elliptic Basis Function

An orthonormal basis is a set of vectors which form a basis for a vector space. For the vector space $\mathbb{R}^n$ mentioned previously, the standard basis consists of the n vectors

$$\mathbf{e}_1 = [1, 0, 0, \ldots, 0]^T, \mathbf{e}_2 = [0, 1, 0, \ldots, 0]^T, \ldots, \mathbf{e}_n = [0, 0, 0, \ldots, 1]^T.$$

These vectors in the standard basis are mutually orthogonal and are all unit vectors. Thus, this standard basis $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ is the *orthonormal basis*. The coordinates of any vectors in $\mathbb{R}^n$ are relative to this standard basis. Therefore, for each vector **x** = $[x_1, x_2, ..., x_n]^T$ in $\mathbb{R}^n$ and for a new given orthonormal basis $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ in $\mathbb{R}^n$, the coordinate $x_i, i = 1, ..., n$, of the vector **x** relative to the new basis or the new axes can be written as follows.

$$x_i = \mathbf{x}^T\mathbf{u}_i \tag{4.4}$$

Considering the hyper-ellipsoidal equation in n-dimensional space, the hyper-ellipsoidal equation of unrotated and centered at the origin is defined as

$$\frac{x_1^2}{a_1^2} + \frac{x_2^2}{a_2^2} + ... + \frac{x_n^2}{a_n^2} = 1 \tag{4.5}$$

where the constant $a_i, i = 1, ..., n$, is the i^{th} semi-axis length of the hyper-ellipsoid. The simplification of Equation (4.5) can be written as

$$\sum_{i=1}^{n} \frac{x_i^2}{a_i^2} = 1 \tag{4.6}$$

By substituting Equation (4.4) into Equation (4.6) to obtain

$$\sum_{i=1}^{n} \frac{(\mathbf{x}^T\mathbf{u}_i)^2}{a_i^2} = 1 \tag{4.7}$$

where $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ is the orthonormal basis.

From Theorem 3.1 , the Equation (4.7) indicates that the coordinates of hyper-ellipsoid in Equation (4.6) are changed to the new coordinates relative to the new orthonormal basis. Therefore, if the new orthonormal basis is rotated from the original one, the hyper-ellipsoid in Equation (4.7) is rotated along with the new orthonormal basis.

From Equation (4.7), the center of the ellipsoidal equation is located at the origin. This equation can be generalized by translating the center of the ellipsoidal Equation (4.7) from the origin to the new center. Suppose $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ is a vector whose coordinates are related to the original axes with the basis $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$. If the original axes of the hyper-ellipsoidal equation are translated from the origin to a new center $\mathbf{c} = [c_1, c_2, ..., c_n]^T$, the new coordinates of vector $\mathbf{x}$ relative to the basis $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ can be written as

$$x_i' = (\mathbf{x} - \mathbf{c})^T\mathbf{u}_i \tag{4.8}$$

The hyper-ellipsoidal equation rotated and located at the center $\mathbf{c} = [c_1, c_2, ..., c_n]^T$ can be written as

$$\sum_{i=1}^{n} \frac{x_i'^2}{a_i^2} = 1. \tag{4.9}$$

By substituting Equation (4.8) into Equation (4.9), it can be written as

$$\sum_{i=1}^{n} \frac{((\mathbf{x} - \mathbf{c})^T\mathbf{u}_i)^2}{a_i^2} = 1 \tag{4.10}$$

From Equation (4.10), the new elliptic basis function, namely, *Versatile Elliptic Basis Function* (VEBF) is defined as shown in the equation below

$$\psi(\mathbf{x}) = \sum_{i=1}^{n} \frac{((\mathbf{x}-\mathbf{c})^T \mathbf{u}_i)^2}{a_i^2} - 1 \tag{4.11}$$

where $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ is the orthonormal basis, the constant a_i, $i = 1, ..., n$, is the i^{th} semi-axis length of the hyper-ellipsoid, and the vector $\mathbf{c} = [c_1, c_2, ..., c_n]^T$ is the center of the hyper-ellipsoid.

In this research, the versatile elliptic basis function (4.11) is applied in the hidden layer of the newly developed neural network. As previously mentioned, the main purpose of this research is to develop a new learning algorithm that can learn the data only one pass. This learned data can be discarded after learning as it is not required for subsequent learning. Consequently, the parameters such as the center and orthonormal basis of the versatile elliptic basis function (4.11) should be adjusted recursively which will be explained in the next two subsections.

4.2.1 Recursive Mean Computation

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be a set of N samples, where each $\mathbf{x}_j \in \mathbb{R}^n, j = 1, ..., N$, is a feature vector. Let μ be the mean vector (center) of this data set. The mean of this data set can be calculated from the following equation.

$$\mu = \frac{1}{N} \sum_{j=1}^{N} \mathbf{x}_j \tag{4.12}$$

Since Equation (4.12) requires all $\mathbf{x}_j$, $j = 1, 2, \ldots, N$, to compute the present value of μ, it is not suitable for 1-pass-throw-away learning where all $\mathbf{x}_j$, $j = 1, 2, \ldots, N$, are discarded after being learned. In order to compute the present value of μ, the value of μ must be rewritten in a form of recursive relation between the previous value of μ and the present value of $\mathbf{x}_N$. Let μ_{old} be the mean vector of the data set X. If $\mathbf{x}_{N+1} \in \mathbb{R}^n$ is the new data vector added into the data set X, then the recursive relation can be written as follows [16].

$$\mu_{new} = \alpha \mu_{old} + \beta \tag{4.13}$$

where μ_{new} is the new mean vector, $\alpha = \frac{N}{N+1}$ and $\beta = \frac{\mathbf{x}_{N+1}}{N+1}$.

In this research, Equation (4.13) is applied for evaluating the center of the VEBF in Equation (4.11). By using Equation (4.13) for calculating the center of the VEBF, the center of the VEBF can be adjusted in one pass. After the the center of VEBF is adjusted by the current datum, it does not require this datum again for further adjusting because the new mean vector in Equation (4.13) requires the old mean vector, the number of old data, and the new incoming datum. The old mean vector and the number of old data have been stored in the VEBF neural network which will be explained in the next section.

4.2.2 Recursive Covariance Matrix Computation

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be a set of N samples, where each $\mathbf{x}_j \in \mathbb{R}^n, j = 1, ..., N$, is a feature vector. Let $\mu \in \mathbb{R}^n$ be the mean vector of this data set. The covariance matrix of this data set is computed from the following equation.

$$S = \frac{1}{N}\sum_{j=1}^{N}(\mathbf{x}_j - \mu)(\mathbf{x}_j - \mu)^T \tag{4.14}$$

Another form of the covariance matrix of Equation (4.14) can be written as

$$S = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j\mathbf{x}_j^T - \mu\mu^T \tag{4.15}$$

Both covariance matrix equations (4.14) and (4.15) are not suitable for 1-pass-throw-away learning since all data must be presented. Therefore, in case of 1-pass-throw-away learning, only the new incoming datum is presented. The other data are already discarded after being learned. Thus, the present covariance matrix must be rewritten in the form of recursive relation similar to Equation (4.13). The following theorem states this recursive relation.

Theorem 4.1 Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be a set of N data vectors in $\mathbb{R}^n$ and S_{old} be the covariance matrix of the data set X. If a new data vector $\mathbf{x}_{N+1} \in \mathbb{R}^n$ is added into the data set X, then

$$S_{new} = \alpha S_{old} + \kappa \tag{4.16}$$

where $\kappa = \frac{1}{N}(\mu_{new} - \mathbf{x}_{N+1})(\mu_{new} - \mathbf{x}_{N+1})^T$, $\alpha = \frac{N}{N+1}$, μ_{new} is the new mean, and S_{new} is the new covariance matrix.

Proof: Let μ_{old} be the mean vector of the data set X. Since S_{old} is the covariance matrix, from Equation (4.15), it can be written as

$$S_{old} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T - \mu_{old}\mu_{old}^T \tag{4.17}$$

If a new data vector $\mathbf{x}_{N+1} \in \mathbb{R}^n$ is added into the data set X, the new covariance matrix, S_{new}, can be written as

$$S_{new} = \frac{1}{N+1}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T + \frac{\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T}{N+1} - \mu_{new}\mu_{new}^T. \tag{4.18}$$

Subtracting Equation (4.18) by Equation (4.17) yields

$$\begin{aligned}
S_{new} - S_{old} &= \frac{1}{N+1}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T - \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T + \kappa_1 \\
&= \left(-\frac{1}{(N+1)N}\right)\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T + \kappa_1 \\
&= -\frac{1}{(N+1)}\left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T\right) + \kappa_1 \\
&= -\frac{1}{(N+1)}\left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T - \mu_{old}\mu_{old}^T\right) + \kappa_1 + \kappa_2 \\
&= -\frac{1}{N+1}S_{old} + \kappa_1 + \kappa_2
\end{aligned}$$

Therefore,

$$\begin{aligned}
S_{new} &= S_{old} - \frac{1}{N+1}S_{old} + \kappa_1 + \kappa_2 \\
&= \left(1 - \frac{1}{N+1}\right)S_{old} + \kappa_1 + \kappa_2 \\
&= \left(\frac{N}{N+1}\right)S_{old} + \kappa_1 + \kappa_2 \\
&= \alpha S_{old} + \kappa
\end{aligned} \tag{4.19}$$

where $\alpha = \frac{N}{N+1}$, $\kappa = \kappa_1 + \kappa_2$, $\kappa_1 = \frac{\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T}{N+1} - \mu_{new}\mu_{new}^T + \mu_{old}\mu_{old}^T$, $\kappa_2 = -\frac{\mu_{old}\mu_{old}^T}{N+1}$.
Simplifying κ in Equation (4.19)

$$\begin{aligned}
\kappa &= \kappa_1 + \kappa_2 \\
&= \frac{\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T}{N+1} - \mu_{new}\mu_{new}^T + \mu_{old}\mu_{old}^T - \frac{\mu_{old}\mu_{old}^T}{N+1} \\
&= \frac{\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T}{N+1} - \mu_{new}\mu_{new}^T + \frac{N}{N+1}\mu_{old}\mu_{old}^T
\end{aligned} \tag{4.20}$$

The Equation (4.13) can be rewritten as

$$\mu_{old} = \frac{1}{N}\left((N+1)\mu_{new} - \mathbf{x}_{N+1}\right) \tag{4.21}$$

Considering the outer product of μ_{old} and itself which can be expressed as follows.

$$\begin{aligned}\mu_{old}\mu_{old}^T &= \frac{1}{N^2}\left((N+1)\mu_{new} - \mathbf{x}_{N+1}\right)\left((N+1)\mu_{new} - \mathbf{x}_{N+1}\right) \\ &= \frac{1}{N^2}((N+1)^2\mu_{new}\mu_{new}^T - (N+1)\mu_{new}\mathbf{x}_{N+1}^T \\ &\quad -(N+1)\mathbf{x}_{N+1}\mu_{new}^T + \mathbf{x}_{N+1}\mathbf{x}_{N+1}^T)\end{aligned} \tag{4.22}$$

Multiply both terms of Equation (4.22) by $\frac{N}{N+1}$ to obtain

$$\begin{aligned}\frac{N}{N+1}\mu_{old}\mu_{old}^T &= \frac{1}{N(N+1)}((N+1)^2\mu_{new}\mu_{new}^T - (N+1)\mu_{new}\mathbf{x}_{N+1}^T \\ &\quad -(N+1)\mathbf{x}_{N+1}\mu_{new}^T + \mathbf{x}_{N+1}\mathbf{x}_{N+1}^T)\end{aligned} \tag{4.23}$$

Substituting Equation (4.23) into Equation (4.20)

$$\begin{aligned}\kappa &= \frac{\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T}{N+1} - \mu_{new}\mu_{new}^T \\ &\quad +\frac{1}{N(N+1)}((N+1)^2\mu_{new}\mu_{new}^T - (N+1)\mu_{new}\mathbf{x}_{N+1}^T \\ &\quad -(N+1)\mathbf{x}_{N+1}\mu_{new}^T + \mathbf{x}_{N+1}\mathbf{x}_{N+1}^T)\end{aligned} \tag{4.24}$$

All denominators in Equation (4.24) are adjusted to be equal as shown in the following equation.

$$
\begin{aligned}
\kappa &= \frac{1}{N(N+1)}(N\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T) - \frac{1}{N(N+1)}(N(N+1)\mu_{new}\mu_{new}^T) \\
&\quad + \frac{1}{N(N+1)}((N+1)^2\mu_{new}\mu_{new}^T - (N+1)\mu_{new}\mathbf{x}_{N+1}^T \\
&\quad -(N+1)\mathbf{x}_{N+1}\mu_{new}^T + \mathbf{x}_{N+1}\mathbf{x}_{N+1}^T) \\
&= \frac{1}{N(N+1)}(((N+1)^2 - N(N+1))\mu_{new}\mu_{new}^T - (N+1)\mu_{new}\mathbf{x}_{N+1}^T \\
&\quad -(N+1)\mathbf{x}_{N+1}\mu_{new}^T + (N+1)\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T) \\
&= \frac{1}{N(N+1)}((N^2+2N+1-N^2-N)\mu_{new}\mu_{new}^T - (N+1)\mu_{new}\mathbf{x}_{N+1}^T \\
&\quad -(N+1)\mathbf{x}_{N+1}\mu_{new}^T + (N+1)\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T) \\
&= \frac{1}{N(N+1)}((N+1)\mu_{new}\mu_{new}^T - (N+1)\mu_{new}\mathbf{x}_{N+1}^T \\
&\quad -(N+1)\mathbf{x}_{N+1}\mu_{new}^T + (N+1)\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T) \\
&= \frac{1}{N(N+1)}((N+1)(\mu_{new}\mu_{new}^T - \mu_{new}\mathbf{x}_{N+1}^T - \mathbf{x}_{N+1}\mu_{new}^T + \mathbf{x}_{N+1}\mathbf{x}_{N+1}^T)) \\
&= \frac{1}{N(N+1)}((N+1)(\mu_{new} - \mathbf{x}_{N+1}^T)(\mu_{new} - \mathbf{x}_{N+1}^T)^T) \\
&= \frac{1}{N}(\mu_{new} - \mathbf{x}_{N+1})(\mu_{new} - \mathbf{x}_{N+1})^T \qquad (4.25)
\end{aligned}
$$

Since the principal component analysis is applied to find the orthonormal basis for the versatile elliptic basis function (4.11), this technique requires the mean vector and the covariance matrix to compute the eigenvalues and eigenvectors. The set of these eigenvectors forms the basis for a data space. Therefore, Equation (4.16) is applied to find the orthonormal basis for the versatile elliptic basis function (4.11). By using Equation (4.16), the covariance matrix can be adjusted in one pass and the data can be discard after being learned.

4.2.3 Orthonormal Basis Computation and Algorithm

The principal component analysis (PCA) is a statistical method applied primarily to transform the input data space into a new lower dimensional space. The axes of the new coordinate system of a new space are generated by translating the original axes into the coordinate of the mean of a data set and then rotating them. The primary axis represents the direction of maximum variance of the data set. The secondary axis, orthogonal to the primary axis, represents the direction of the second largest variance of the data set, and so

on. In this research, we apply the concept of the PCA for finding the orthonormal basis for the versatile elliptic basis function (VEBF).

Given the data set $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, $\mathbf{x}_j \in \mathbb{R}^n, j = 1, ..., N$, the mean vector and the covariance matrix can be computed from Equation (4.13) and (4.16), respectively. Similarly, eigenvalues and eigenvectors can be computed from the covariance matrix. Since the eigenvectors are orthogonal to one another and their lengths are equal to 1, the set of eigenvectors forms the orthonormal basis for this data space. Let $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ be the orthonormal basis, this orthonormal basis for the VEBF can be computed by the following algorithm.

Orthonormal Basis Computation Algorithm

1. Calculate the mean vector using Equation (4.13).

2. Calculate the covariance matrix using Equation (4.16).

3. Calculate eigenvalues of the covariance matrix in step 2:

$$\lambda_1 > \lambda_2 > ... > \lambda_n \tag{4.26}$$

   where $\lambda_i, i = 1..n$, is the eigenvalue.

4. Calculate eigenvectors from the eigenvalues obtained in step 3:

$$\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n \tag{4.27}$$

   where $\mathbf{u}_i \in \mathbb{R}^n, i = 1..n$, is the eigenvector corresponding to λ_i.

5. Assign the set of eigenvectors in step 4, $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$, to be the set of orthonormal basis for the data space.

Since the set of these eigenvectors forms the basis for the data space, this algorithm is applied to find the orthonormal basis for the versatile elliptic basis function (4.11) which is used in the new developed neural network and its learning algorithm.

4.3 The Proposed Learning Algorithm

Let $X = \{(\mathbf{x}_j, t_j)|1 \leq j \leq N\}$ be a finite set of N training data, where $\mathbf{x}_j$ is a vector in $\mathbb{R}^n$ referred to as a *data vector* and t_j is the *class label* of the vector $\mathbf{x}_j$. Let $\Omega = \{\Omega_k|1 \leq k \leq m\}$ be a set of *m* neurons in the hidden layer. Each Ω_k contains the covariance matrix, the center vector whose dimension is equal to that of the data vector, the semi-axis vector whose elements are the semi-axis lengths, the number of data covered by k^{th} neuron, and the class label. Therefore, Ω_k is denoted as a 5-tuple, $(\mathbf{S}_k, \mathbf{c}_k, \mathbf{a}_k, n_k, d_k)$, where $\mathbf{S}_k$ is the covariance matrix of k^{th} neuron, $\mathbf{c}_k = [c_1^k, c_2^k, ..., c_n^k]^T$ is the center of k^{th} neuron, $\mathbf{a}_k = [a_1^k, a_2^k, ..., a_n^k]^T$ is the semi-axis vector of k^{th} neuron, n_k is the total number of data covered by k^{th} neuron, and d_k is the class label of k^{th} neuron. Let $O = \{O_q|1 \leq q \leq p\}$ be a set of *p* neurons in the output layer.

4.3.1 Geometrical Growth Criterion

Initially, there is no hidden neuron in the VEBF neural network. A new hidden neuron can be automatically added into the network whenever the condition is satisfied. When a training data $(\mathbf{x}_j, t_j)$ is fed into the network, the VEBF neural network find whether there exists a hidden neuron with the same class that is closest to the input vector $\mathbf{x}_j$. If there is no such hidden neuron, a new hidden neuron, defined as $\Omega_{new} = (\mathbf{S}_{new}, \mathbf{c}_{new}, \mathbf{a}_{new}, n_{new}, d_{new})$, and a new output neuron are allocated and added into the network. Let $\mathbf{a}_0 = [a_1, a_2, ..., a_n]^T$ be an initial axes. The parameters of this new hidden neuron are initialized as follows.

$$\mathbf{S}_{new} = \mathbf{0} \tag{4.28}$$

$$\mathbf{c}_{new} = \mathbf{x}_j \tag{4.29}$$

$$\mathbf{a}_{new} = \mathbf{a}_0 \tag{4.30}$$

$$n_{new} = 1; \tag{4.31}$$

$$d_{new} = t_j \tag{4.32}$$

The *zero matrix*, defined as **0** in Equation (4.28), is the matrix which all elements are zero. However, if there exists the closest hidden neuron in the network, this data vector is temporarily considered to be an element of the closest hidden neuron. Let cs be the index of this

closest hidden neuron. The new temporary parameters of the closest hidden neuron including its new center $\mathbf{c}_{cs}^{new}$, the new covariance matrix $\mathbf{S}_{cs}^{new}$, and the new number of elements of this closest hidden neuron n_{cs} are computed but the semi-axis lengths of the versatile elliptic basis function are not computed at this moment. The new semi-axis lengths of VEBF are not computed at this moment because this VEBF with the old semi-axis lengths and these new temporary parameters need to be considered whether it can cover this input vector **x**. However, the closest hidden neuron might not be actually close to the data vector. The following criterion is used to determine whether the closeness is acceptable. If it is not acceptable then a new hidden neuron is introduced and added to the network. The output of the closest hidden neuron is computed using the new temporary parameters along with Equation (4.11) as follows.

$$\psi_{cs}(\mathbf{x}_j) = \sum_{i=1}^{n} \frac{((\mathbf{x}_j - \mathbf{c}_{cs}^{new})^T \mathbf{u}_i^{new})^2}{(a_i^{cs})^2} - 1 \tag{4.33}$$

If $\psi_{cs}(\mathbf{x}_j) > 0$, called *geometrical growth criterion*, is satisfied then a new hidden neuron, $\Omega_{new} = (\mathbf{S}_{new}, \mathbf{c}_{new}, \mathbf{a}_{new}, n_{new}, d_{new})$, is allocated and added into the network. The parameters of this new neuron are initialized as Equation (4.28), (4.29), (4.30), (4.31), and (4.32), respectively.

Therefore, whenever the new input vector is fed into the network and the geometrical growth criterion is satisfied or there is no hidden neuron with the same class of this new input vector, the new hidden neuron is created and then added into the network. The new parameters of this new hidden neuron are initialized by using equations (4.28), (4.29), (4.30), (4.31), and (4.32), respectively. Moreover, a new neuron in the output layer can be automatically added into the network whenever the new data with the new classes are fed into the network.

4.3.2 Merging Strategy

Since the new neurons can be automatically added into the network, the number of neurons in the network will be gradually increased during learning process. However, it is possible that there are some number of redundant neurons in the network. In which case, some neurons may be very close to one another and can be merged together in order to reduce the number of neurons in the network. For this reason, a merging strategy for these

redundant neurons will be introduced in the learning algorithm in order to reduce these redundant neurons and the computational cost.

For merging the two neurons into a new neuron, since all data learned by these two neurons are discarded, the parameters of the new neuron cannot be computed from previous learn data. The new parameters such as the new center and the new covariance must be calculated from the parameters of these two neurons. Thus, the merging center and the merging covariance should be derived in order to evaluate the new parameters. The following theorems state theses merging strategies.

Theorem 4.2 Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{N_1}\}$ be a set of N_1 data vectors and $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{N_2}\}$ be a set of N_2 data vectors. Let μ_1 and μ_2 be the mean vector of the data set **X** and **Y**, respectively. If these two data set **X** and **Y** are merged into the new data set **Z**, then

$$\mu_{new} = \frac{1}{N_1 + N_2} (N_1\mu_1 + N_2\mu_2) \tag{4.34}$$

where μ_{new} is the new mean vector called *combined mean vector* of the new data set **Z**.

Proof: Sice μ_1 is the mean vector of the data set **X**, that is

$$\mu_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} \mathbf{x}_i. \tag{4.35}$$

The Equation (4.35) can be rewritten as

$$N_1\mu_1 = \sum_{i=1}^{N_1} \mathbf{x}_i. \tag{4.36}$$

Sice μ_2 is the mean vector of the data set **Y**, that is

$$\mu_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} \mathbf{y}_i. \tag{4.37}$$

The Equation (4.37) can be rewritten as

$$N_2\mu_2 = \sum_{i=1}^{N_2} \mathbf{y}_i. \tag{4.38}$$

Because μ_{new} is the mean vector of the data set **Z**, this means that

$$\mu_{new} = \frac{1}{N_1 + N_2}(\sum_{i=1}^{N_1} \mathbf{x}_i + \sum_{i=1}^{N_2} \mathbf{y}_i). \tag{4.39}$$

Substituting Equation (4.36) and Equation (4.38) into Equation (4.39) yields

$$\mu_{new} = \frac{1}{N_1 + N_2} \left(N_1 \mu_1 + N_2 \mu_2 \right) \tag{4.40}$$

Theorem 4.3 Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{N_1}\}$ be a set of N_1 data vectors and $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{N_2}\}$ be a set of N_2 data vectors. Let μ_1 and μ_2 be the mean vector of the data set **X** and **Y**, respectively, and $\mathbf{S}_1$ and $\mathbf{S}_2$ be the covariance matrix of the data set **X** and **Y**, respectively. If these two data set **X** and **Y** are merged into the new data set **Z** then

$$\mathbf{S}_{new} = \frac{N_1}{N_1 + N_2}\mathbf{S}_1 + \frac{N_2}{N_1 + N_2}\mathbf{S}_2 + \frac{N_1 N_2}{(N_1 + N_2)^2}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{4.41}$$

where $\mathbf{S}_{new}$ is the new covariance matrix called *combined covariance matrix* of the new data set **Z**.

Proof: Let μ_{new} be the combined mean vector in theorem 4.2. Thus,

$$\mu_{new} = \frac{1}{N_1 + N_2} \left(N_1 \mu_1 + N_2 \mu_2 \right). \tag{4.42}$$

Because $\mathbf{S}_1$ and μ_1 are the covariance matrix and the mean vector of data set **X**, respectively, that is

$$\mathbf{S}_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} \mathbf{x}_i \mathbf{x}_i^T - \mu_1 \mu_1^T \tag{4.43}$$

It follows that

$$\sum_{i=1}^{N_1} \mathbf{x}_i \mathbf{x}_i^T = N_1 \mathbf{S}_1 + N_1 \mu_1 \mu_1^T \tag{4.44}$$

Since $\mathbf{S}_2$ and μ_2 are the covariance matrix and the mean vector of data set **Y**, respectively, that is

$$\mathbf{S}_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} \mathbf{y}_i \mathbf{y}_i^T - \mu_2 \mu_2^T \tag{4.45}$$

Equation (4.45) can be rewritten as

$$\sum_{i=1}^{N_1} \mathbf{y}_i \mathbf{y}_i^T = N_2 \mathbf{S}_2 + N_2 \mu_2 \mu_2^T \tag{4.46}$$

Since $\mathbf{S}_{new}$ is the new covariance matrix of the new data set **Z**, from Equation (4.15) yields

$$\mathbf{S}_{new} = \frac{1}{N_1 + N_2} \left(\sum_{i=1}^{N_1} \mathbf{x}_i \mathbf{x}_i^T + \sum_{i=1}^{N_1} \mathbf{y}_i \mathbf{y}_i^T \right) - \mu_{new} \mu_{new}^T. \tag{4.47}$$

Substituting Equation (4.44) and (4.46) into Equation (4.47) yields

$$\mathbf{S}_{new} = \frac{1}{N_1+N_2}\left(N_1\mathbf{S}_1 + N_1\mu_1\mu_1^T + N_2\mathbf{S}_2 + N_2\mu_2\mu_2^T\right) - \mu_{new}\mu_{new}^T. \tag{4.48}$$

Using Equation (4.42), the outer product of μ_{new} and itself can be written as

$$\begin{aligned}
\mu_{new}\mu_{new}^T &= \frac{1}{(N_1+N_2)^2}\left(N_1\mu_1 + N_2\mu_2\right)\left(N_1\mu_1 + N_2\mu_2\right)^T \\
&= \frac{1}{(N_1+N_2)^2}\left(N_1^2\mu_1\mu_1^T + N_1N_2\mu_1\mu_2^T + N_1N_2\mu_2\mu_1^T + N_2^2\mu_2\mu_2^T\right) \\
&= \frac{N_1^2}{(N_1+N_2)^2}\mu_1\mu_1^T + \frac{N_1N_2}{(N_1+N_2)^2}\mu_1\mu_2^T + \frac{N_1N_2}{(N_1+N_2)^2}\mu_2\mu_1^T \\
&\quad + \frac{N_2^2}{(N_1+N_2)^2}\mu_2\mu_2^T
\end{aligned} \tag{4.49}$$

Equation (4.48) can be expressed as

$$\begin{aligned}
\mathbf{S}_{new} &= \frac{N_1}{N_1+N_2}\mathbf{S}_1 + \frac{N_2}{N_1+N_2}\mathbf{S}_2 + \frac{N_1^2}{(N_1+N_2)^2}\mu_1\mu_1^T + \frac{N_1N_2}{(N_1+N_2)^2}\mu_1\mu_1^T \\
&\quad + \frac{N_1N_2}{(N_1+N_2)^2}\mu_2\mu_2^T + \frac{N_2^2}{(N_1+N_2)^2}\mu_2\mu_2^T - \mu_{new}\mu_{new}^T
\end{aligned} \tag{4.50}$$

Substituting $\mu_{new}\mu_{new}^T$ of Equation (4.49) into Equation (4.50) to obtain

$$\begin{aligned}
\mathbf{S}_{new} &= \frac{N_1}{N_1+N_2}\mathbf{S}_1 + \frac{N_2}{N_1+N_2}\mathbf{S}_2 + \frac{N_1^2}{(N_1+N_2)^2}\mu_1\mu_1^T + \frac{N_1N_2}{(N_1+N_2)^2}\mu_1\mu_1^T \\
&\quad + \frac{N_1N_2}{(N_1+N_2)^2}\mu_2\mu_2^T + \frac{N_2^2}{(N_1+N_2)^2}\mu_2\mu_2^T - \frac{N_1^2}{(N_1+N_2)^2}\mu_1\mu_1^T \\
&\quad - \frac{N_1N_2}{(N_1+N_2)^2}\mu_1\mu_2^T - \frac{N_1N_2}{(N_1+N_2)^2}\mu_2\mu_1^T - \frac{N_2^2}{(N_1+N_2)^2}\mu_2\mu_2^T \\
&= \frac{N_1}{N_1+N_2}\mathbf{S}_1 + \frac{N_2}{N_1+N_2}\mathbf{S}_2 + \frac{N_1N_2}{(N_1+N_2)^2}\mu_1\mu_1^T + \frac{N_1N_2}{(N_1+N_2)^2}\mu_2\mu_2^T \\
&\quad - \frac{N_1N_2}{(N_1+N_2)^2}\mu_1\mu_2^T - \frac{N_1N_2}{(N_1+N_2)^2}\mu_2\mu_1^T \\
&= \frac{N_1}{N_1+N_2}\mathbf{S}_1 + \frac{N_2}{N_1+N_2}\mathbf{S}_2 + \frac{N_1N_2}{(N_1+N_2)^2}\left(\mu_1\mu_1^T - \mu_1\mu_2^T - \mu_2\mu_1^T + \mu_2\mu_2^T\right) \\
&= \frac{N_1}{N_1+N_2}\mathbf{S}_1 + \frac{N_2}{N_1+N_2}\mathbf{S}_2 + \frac{N_1N_2}{(N_1+N_2)^2}(\mu_1-\mu_2)(\mu_1-\mu_2)^T
\end{aligned}$$

Any two hidden neurons can be merged into one new neuron whenever some conditions are satisfied. The Equation (4.34) and (4.41) in Theorem 4.2 and 4.3 are applied to adjust the new parameters of this new neuron. Let $\Omega_x = (\mathbf{S}_x, \mathbf{c}_x, \mathbf{a}_x, n_x, d_x)$ and $\Omega_y = (\mathbf{S}_y, \mathbf{c}_y, \mathbf{a}_y, n_y, d_y)$ be any two hidden neurons x and y in a VEBF neural network, respectively. In this research, the *merging function* is defined in terms of the distance from neuron

x to neuron y as follows.

$$\phi(\mathbf{c}_x, \mathbf{c}_y) = \sum_{i=1}^{n} \frac{((\mathbf{c}_x - \mathbf{c}_y)^T \mathbf{u}_i^y)^2}{(a_i^y)^2} - 1 \tag{4.51}$$

where $\mathbf{u}_i^y$ is the i^{th} basis of the orthonormal basis of the nodes Ω_y, and a_i^y is the i^{th} semi-axis length of the node Ω_y.

If *merging criterion*, $\phi(\mathbf{c}_x, \mathbf{c}_y) \leq \theta$ and $d_x = d_y$, is satisfied, these two hidden neurons are merged into one new hidden neuron $\Omega_{new} = (\mathbf{S}_{new}, \mathbf{c}_{new}, \mathbf{a}_{new}, n_{new}, d_{new})$. The threshold θ is a constant value used to identify whether these two neurons should be merged into a single neuron. Using Equation (4.34) and (4.41), the new parameters of this new hidden neuron can be computed as follows.

$$\mathbf{c}_{new} = \frac{1}{n_x + n_y}(n_x \mathbf{c}_x + n_y \mathbf{c}_y) \tag{4.52}$$

$$\mathbf{S}_{new} = \frac{n_x}{n_x + n_y}\mathbf{S}_x + \frac{n_y}{n_x + n_y}\mathbf{S}_y + \frac{n_x n_y}{(n_x + n_y)^2}(\mathbf{c}_x - \mathbf{c}_y)(\mathbf{c}_x - \mathbf{c}_y)^T \tag{4.53}$$

$$n_{new} = n_x + n_y \tag{4.54}$$

$$a_i^{new} = \sqrt{2\pi|\lambda_i|},\ i = 1, ..., n \tag{4.55}$$

$$d_{new} = d_x \tag{4.56}$$

where λ_i is i^{th} eigenvalue of the new covariance matrix $\mathbf{S}_{new}$.

4.3.3 VEBF Learning Algorithm

Let $\Omega = \{\Omega_k | 1 \leq k \leq m\}$ be the set of m neurons. Therefore, m is the number of elements in the set Ω. In other words, m is the number of neurons in the network. If there is no hidden neuron in the network, the value of m is zero. Let θ be the threshold for merging the two hidden neurons in the network, and $\mathbf{X} = \{(\mathbf{x}_j, t_j) | 1 \leq j \leq N\}$ be a set of N training data. The learning algorithm for VEBF neural network can be summarized as follows:

1. Initialize the semi-axis vector $\mathbf{a}_0 = [a_1, a_2, ..., a_n]^T$ and the value of n_0.

2. Input the training data $(\mathbf{x}_j, t_j)$ into the VEBF neural network.

3. Find a hidden neuron $\Omega_k \in \Omega$ and $d_k = t_j$ such that

$$k = \arg\min_i(||\mathbf{x}_j - \mathbf{c}_i||),\ i = 1, ..., m.$$

If there exists such a neuron Ω_k **then** do the following steps:

(a) Compute the new center $\mathbf{c}_k^{new}$ based on Equation (4.13).

$$\mathbf{c}_k^{new} = \alpha \mathbf{c}_k^{old} + \beta$$

(b) Compute the new covariance matrix $\mathbf{S}_k^{new}$ based on Equation (4.16).

$$\mathbf{S}_k^{new} = \alpha \mathbf{S}_k^{old} + \kappa$$

else

(a) Create a new hidden neuron Ω_{new}.

(b) Adjust the parameters of this new neuron using Equations (4.28), (4.29), (4.30), (4.31), and (4.32).

(c) Set $\Omega = \Omega \cup \Omega_{new}$ and remove $(\mathbf{x}_j, t_j)$ from the training data set $\mathbf{X}$.

(d) Go to step 7.

end

4. Compute the orthonormal basis for Ω_k.

(a) Compute the eigenvalues of covariance matrix $\mathbf{S}_k^{new}$:

$$\lambda_1 > \lambda_2 > ... > \lambda_n$$

(b) Compute the eigenvectors of the covariance matrix $\mathbf{S}_k^{new}$:

$$\mathbf{u}_1^k, \mathbf{u}_2^k, ..., \mathbf{u}_n^k$$

where $\mathbf{u}_i^k$ is the eigenvector corresponding to λ_i.

(c) Assign $\left\{\mathbf{u}_1^k, \mathbf{u}_2^k, ..., \mathbf{u}_n^k\right\}$ be the set of orthonormal basis for Ω_k.

5. Compute $\psi_k(\mathbf{x}_j) = \sum_{i=1}^{n} \frac{((\mathbf{x}_j - \mathbf{c}_k^{new})^T \mathbf{u}_i^k)^2}{(a_i^k)^2} - 1$.

If $\psi_k(\mathbf{x}_j) \leq 0$ **then** update the parameters of Ω_k using these steps:

(a) Update the center of Ω_k, $\mathbf{c}_k = \mathbf{c}_k^{new}$.

(b) Update the covariance matrix of Ω_k, $\mathbf{S}_k = \mathbf{S}_k^{new}$.

(c) Update the parameter $n_k = n_k + 1$.

(d) **If** $n_k > n_0$ **then** update the semi-axis lengths of Ω_k by setting

$$a_i^k = a_i^k + \left|(\mathbf{c}_k^{new} - \mathbf{c}_k^{old})^T \mathbf{u}_i^k\right|, \;\; i = 1, ..., n$$

end

else

(a) Create a new hidden neuron Ω_{new}.

(b) Adjust the parameters of this new neuron using Equations (4.28), (4.29), (4.30), (4.31), and (4.32).

(c) Set $\Omega = \Omega \cup \Omega_{new}$ and remove $(\mathbf{x}_j, t_j)$ from the training data set $\mathbf{X}$.

end

6. Compute the merging function $\phi(\mathbf{c}_k, \mathbf{c}_l)$ and $\phi(\mathbf{c}_l, \mathbf{c}_k)$ such that $d_k = d_l$ for $l = 1, ..., m$ based on Equation (4.51).
   If $\phi(\mathbf{c}_k, \mathbf{c}_l) \leq \theta$ or $\phi(\mathbf{c}_l, \mathbf{c}_k) \leq \theta$ **then** do the following steps:

   (a) Merge the hidden neurons Ω_k and Ω_l into the new hidden neuron Ω_{new} and compute its parameters by using Equations (4.52), (4.53), (4.54), (4.55), and (4.56).

   (b) set $\Omega = \Omega \cup \Omega_{new}$ and then remove Ω_k and Ω_l from the network.

   end

7. **If** the training data set $\mathbf{X}$ is not empty **then** go to step 2
   else, stop training.

Theorem 4.4. The time complexity, T_{alg}, of the proposed learning algorithm is $O(m)$ where m is the number of data.

Proof: Since the computations in steps 1, 2, and 7 do not depend on the number of data, the time used in these steps is constant. Then, the time complexity of these steps is $O(1)$. In step 3, the searching of the minimum distance between the current training data

and the existing neurons can be done by comparing the distant between the current data and the existing neurons. Because the maximum number of the existing neurons is $m-1$, the comparison can be done in $m-1$ time in the worst case. In addition, since computations of the new mean vector and the new covariance matrix are not dependent on the number of data, the time complexity is constant. So, the time complexity of the third step is $O(m)$. In step 4, the computations of eigenvalues and eigenvectors do not depend on the number of data. Then, the time complexity of this step is $O(1)$. In step 5, the computation of this step depends on the dimension of the data which is constant and is independent of the number of data. So, the time complexity of this step is $O(1)$. In step 6, the computations are done in $m-1$ time in the worst case so the time complexity of this step is $O(m)$. From steps 1 to 7, the time complexity is

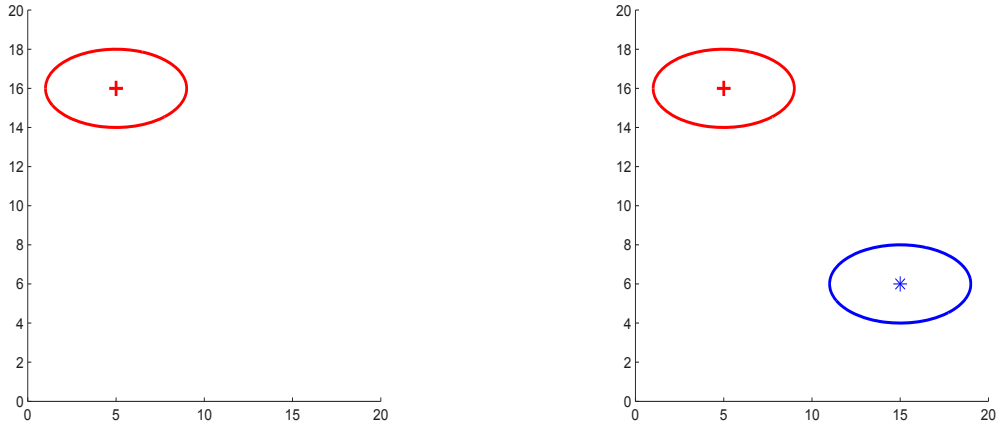
$$T_{alg} \;=\; O(1)+O(1)+O(m)+O(1)+O(1)+O(m)+O(1)=O(m)$$

4.3.4 Training VEBF Neural Network

Once each input data vector is learned, it is discarded from the process forever. Initially, the VBEF neural network is empty. The nodes in the network will be automatically increased during the training process. Consider this simple example. Suppose that X = $\{([5,16]^T,0)$, $([15,6]^T,1)$, $([10,18]^T,0)$, $([5,6]^T,1)$, $([11,16]^T,0)\}$ is a set of training data in $\mathbb{R}^2$. There are two classes in this data set labeled by class 0 and class 1. Suppose the training data in class 0 is illustrated by the plus '+' while the training data of class 1 is illustrated by the star '*'.

Firstly, the training data $([5,16]^T,0)$ is fed into the VEBF neural network. This training data is in class 0. Since there is no VEBF neuron of class 0 in the network, a new VEBF neuron of class 0 is created and all parameters of this new VEBF neuron are computed. This creates VEBF neuron which is shown in Figure 4.2(a) with the versatile elliptic basis function in $\mathbb{R}^2$.

Secondly, the training data $([15,6]^T,1)$ is fed into the VEBF neural network. This training data is in class 1. Because there is no VEBF neuron of class 1 in the network, a new VEBF neuron of class 1 is created and all parameters of this new VEBF neuron are computed. Figure 4.2(b) shows the created VEBF neuron of class 1.

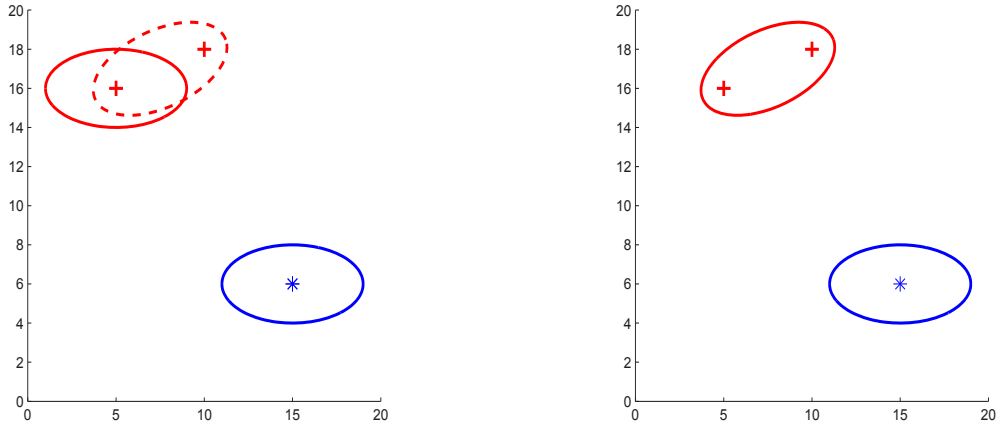


(a) (b)

Figure 4.2: The learning event for the first two input data. (a) The first neuron with class '0' is created in the feature space. (b) The second neuron with class '1' is created next.

Thirdly, the training data $([10, 18]^T, 0)$ is fed into the VEBF neural network. This training data is in class 0. Since there exists a VEBF neuron in class 0, the closest VEBF neuron can be found, and the data is temporarily assigned to this neuron. All new temporary parameters of the this neuron and the output value of this versatile elliptic basis function are computed using these new temporary parameters. Since the computed output value of this function is less than zero, it means that the neuron can cover the data and this training data can be assigned to this neuron. All new temporary parameters now become the actual parameters of this neuron. Figure 4.3(a) shows the VEBF neuron trying to adjust itself to cover the new data. The adjusted neuron to cover the data is shown in Figure 4.3(b).

Fourthly, the training data $([5, 6]^T, 1)$ is fed into the VEBF neural network. This training data is in class 1. Since there exists some VEBF neuron in class 1, the closest VEBF neuron is found. The algorithm tries to assign the data to this neuron. All new temporary parameters of this neuron and the output value of the versatile elliptic basis function are computed using these new temporary parameters. Because this output value is greater than zero, it means that the neuron cannot cover the data. A new VEBF neuron is created and all new parameters of this neuron are assigned. Figure 4.4(a) shows the second neuron trying to adjust itself to cover the new data. Since it cannot cover the data, a new VEBF neuron is created to cover this new data as shown in Figure 4.4(b).

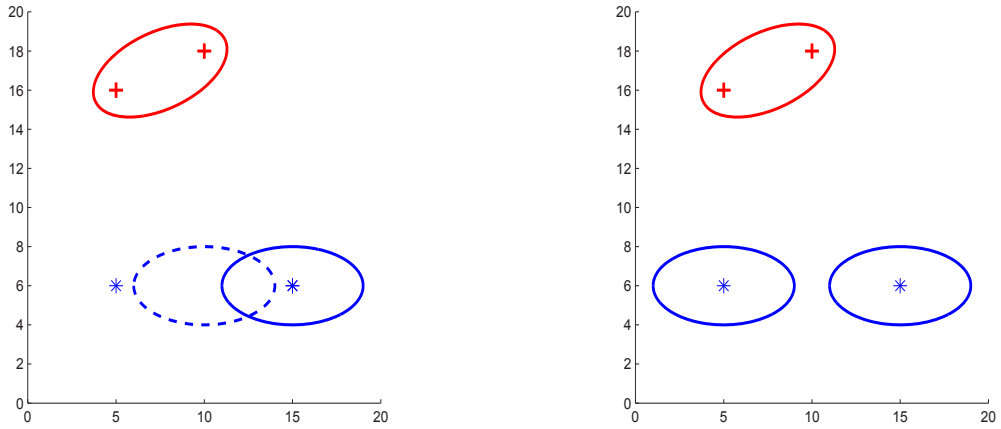


(a) (b)

Figure 4.3: The learning process for the next two data. (a) The first neuron attempts to adjust itself to cover the data (dot line). (b) The adjusted neuron covering the data.

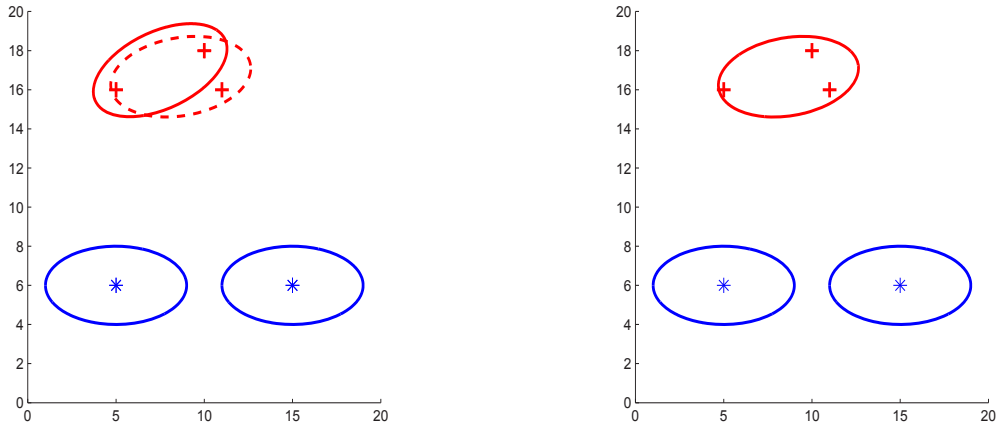
Finally, the training data $([11, 16]^T, 0)$ is fed into the VEBF neural network. This training data is in class 0. Since there exists some VEBF neurons in class 0, the closest VEBF neuron is found. Then, the algorithm tries to assign the data to this neuron. All new temporary parameters of this neuron and the output value of the versatile elliptic basis function are computed using these new temporary parameters. Because the value of this function is less than zero, this training data is assigned to this neuron and all new temporary parameters become the actual parameters of this neuron. Figure 4.5(a) shows the VEBF neuron trying to adjust itself to cover the new data. Since it can cover the new data, the adjusted neuron to cover the data and all new parameters of this node are updated. The adjusted neuron is shown in Figure 4.5(b).

Another example is the spiral data set trained by the proposed algorithm with the constant $n_0 = 10$. The initial semi-axis vector of VEBF is equal to $[0.9, 0.5]^T$. The network learns this data set in only one epoch. After training, the result is illustrated in Figure 4.6.



(a) (b)

Figure 4.4: The event during the fourth data training. (a) The second neuron attempts to adjust itself to cover the data (dot line). (b) The third neuron is created in the feature space.



(a) (b)

Figure 4.5: The event during the fifth data training. (a) The first neuron attempts to adjust itself to cover the data (dot line). (b) The first neuron adjusts itself to cover the data in the feature space.

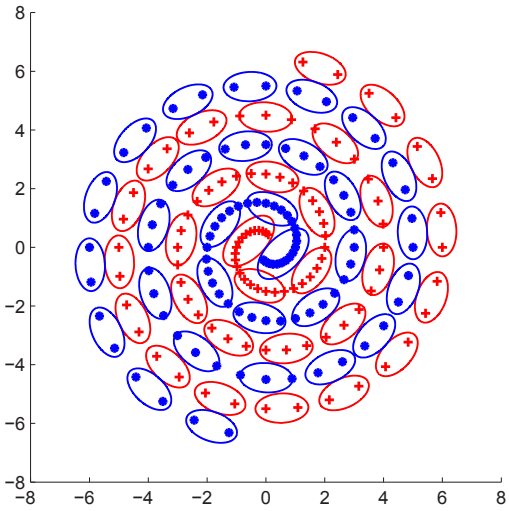


Figure 4.6 The spiral data set trained by our proposed algorithm.

CHAPTER V

EXPERIMENTAL RESULTS

In this research, the performance of VEBF neural network is evaluated on the two-class classification problem and multi-class classification problem. The results are compared with the conventional radial basis function neural network with Gaussian radial basis function (RBF), multilayer perceptron (MLP) for multi-class classification problem and two-class classification problem. In the two-class classification problem, the results are also compared with support vector machine (SVM) because SVM is suitable for the two-class classification problem. In the multi-class classification problem, the results are not compared with support vector machine (SVM) because SVM has a costly computational time. The simulations are done on MATLAB using neural network toolbox. To fairly evaluate the performance, the number of neurons used in VEBF, MLP, and RBF models are equally set in all simulations. However, the number of neurons of RBF and MLP are set according to the proposed model because the numbers of neurons of the RBF and MLP can be fixed in advance but the number of neurons of the proposed model cannot be fixed in advance. The data sets used to train and test are collected from UCI Repository of machine learning database [15]. The properties of the data set are given in Table 5.1. In this experiment, five-fold cross-validation is used to train and test the models. Each data set is divided into five disjoint subsets. Then, four subsets are used as a training set and the other subset is used as a testing set. This process is repeated five times which each of the five subsets is used exactly once as the testing set. Eventually, the results from each testing set are averaged. In multi-class classification problem, the testing data may be classified into the *unknown* class which is considered as misclassification. In two-class classification problem, each datum in the *unknown* class will be classified into the class of the hidden neuron that is closest to this datum in order to reduce the problem of local decision boundary. The proposed neural network (VEBF neural network) is trained by the proposed learning algorithm in only one epoch. The proposed VEBF neural network can incrementally learn the new data sets with-

Table 5.1 Properties of the data sets used in the experiment.

Data sets	No. of attributes	No. of classes	No. of instances
Iris	4	3	150
E.coli	8	8	336
Yeast	8	10	1484
Image Segmentation	19	7	2310
Waveform	21	3	5000
Heart	13	2	270
Spambase	57	2	4601
Sonar	60	2	208
Liver	7	2	345

out the previous learned data sets but the MLP, RBF, and SVM cannot. The initial semi-axis lengths of VEBF neural network, denoted by $a_k, k = 1, \ldots, n$, is computed from

$$a_k = \delta * d_{av} \tag{5.1}$$

where $d_{av} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij}$, d_{ij} is the Euclidean distance between i^{th} and j^{th} instances, and N is the number of data. For VEBF learning algorithm, the constant n_0 is set to 2 and the threshold θ is set to 0 for all experiments. For the MLP model, the simulation runs 10 times with each data set and the best accuracy is chosen. However, the initial width σ_k of the radial basis function is equally set to the initial semi-axis length a_k of the proposed basis function if they can achieve the highest accuracy and set to be larger or smaller than the proposed basis function if they can achieve the highest accuracy with the same number of neurons.

5.1 Multi-Class Classification Problem

5.1.1 Iris Data Set

The iris data set consists of four attributes in three classes. There are 150 instances in this data set. In this experiment, the proposed algorithm uses only one epoch to learn this data set. The initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1/3$

and the initial width of RBF is computed from Equation (5.1) with $\delta = 10$. For the MLP model, the simulation runs 10 times and the best result is chosen. The comparative results of VEBF, RBF, and MLP are illustrated in Table 5.2. From this table, the accuracy of the proposed model and MLP are equal but higher than the accuracy of RBF. However, the training time of the proposed model is less than that of MLP and RBF when the number of neurons is equal.

Table 5.2 The comparative results trained by Iris Data Set.

Testing fold	VEBF			RBF			MLP		
	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)
1	0.08	3	96.67	1.11	3	96.67	1.18	3	96.67
2	0.03	3	100.00	0.13	3	96.67	0.43	3	100.00
3	0.03	3	96.67	0.10	3	100.00	0.43	3	100.00
4	0.03	3	100.00	0.10	3	96.67	0.44	3	96.67
5	0.03	3	96.67	0.10	3	96.67	0.43	3	96.67
Average	0.04	3	98.00	0.31	3	97.33	0.58	3	98.00

5.1.2 Ecoli Data Set

The ecoli data set consists of eight attributes in eight classes. There are 336 instances in this data set. In this experiment, the initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1$ and the initial width of RBF is computed from Equation (5.1) with $\delta = 1$. For the MLP model, the simulation runs 10 times and the best result is chosen. The proposed algorithm used only one epoch to learn this data set. The comparative results of VEBF, RBF, and MLP are shown in Table 5.3. From this table, the accuracy of the proposed model is higher than that of RBF and MLP. In addition, the training time of the proposed model is less than those of RBF and MLP.

5.1.3 Yeast Data Set

The yeast data set consists of eight attributes in 10 classes. There are 1484 instances in this data set. In this experiment, the initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1$ and the initial width of RBF is computed from Equation (5.1)

Table 5.3 The comparative results trained by Ecoli Data Set.

Testing fold	VEBF			RBF			MLP		
	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)
1	0.12	8	88.41	1.13	8	68.12	1.24	8	66.67
2	0.09	8	87.88	0.16	8	65.15	0.49	8	72.73
3	0.09	8	92.54	0.15	8	64.18	0.57	8	62.69
4	0.08	8	79.71	0.14	8	56.52	0.52	8	62.32
5	0.09	8	84.62	0.15	8	67.69	0.43	8	72.31
Average	0.09	8	86.63	0.35	8	64.33	0.65	8	67.34

with $\delta = 1$. For the MLP model, the simulation runs 10 times and the best result is chosen. The proposed algorithm learned this data set in only one epoch. The comparative results of VEBF, RBF, and MLP are illustrated in Table 5.4. From this table, the accuracy of the proposed model is higher than the accuracy of RBF and MLP. Furthermore, the training time of the proposed model is less than that of RBF and MLP.

Table 5.4 The comparative results trained by Yeast Data Set.

Testing fold	VEBF			RBF			MLP		
	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)
1	0.58	17	58.59	3.04	17	41.08	1.86	17	42.09
2	0.48	17	55.70	2.08	17	39.26	2.26	17	42.62
3	0.50	18	56.76	2.16	18	43.24	0.75	18	38.18
4	0.59	17	55.41	2.13	17	35.14	1.44	17	43.58
5	0.53	17	54.88	2.08	17	37.71	1.02	17	36.36
Average	0.54	17.2	56.27	2.30	17.2	39.29	1.47	17.2	40.57

5.1.4 Image Segmentation Data Set

The image segmentation data set consists of 19 attributes in seven classes. There are 2310 instances in this data set. In this experiment, the initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1$ and the initial width of RBF is computed from Equation (5.1) with $\delta = 1$. For the MLP model, the simulation runs 10 times and the best result

is chosen. The proposed algorithm learned this data set in only one epoch. The comparative results of VEBF, RBF, and MLP are shown in Table 5.5. From this table, the accuracy of the proposed model is less than that of the MLP model but higher than the accuracy of the RBF model. However, the training time of the proposed model is less than that of RBF and MLP.

Table 5.5 The comparative results trained by Image Segmentation Data Set.

	VEBF			**RBF**			**MLP**		
Testing fold	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)
1	1.69	13	75.76	5.63	13	39.61	7.70	13	89.18
2	1.38	12	80.74	4.61	12	45.02	3.20	12	87.45
3	1.16	10	76.62	4.19	10	40.26	3.13	10	89.18
4	1.42	12	80.74	4.59	12	45.45	1.71	12	86.36
5	1.38	12	79.00	4.54	12	44.59	3.20	12	81.17
Average	1.41	11.8	78.57	4.71	11.8	42.99	3.79	11.8	86.67

5.1.5 Waveform Data Set

The waveform data set consists of 21 attributes in three classes. There are 5000 instances in this data set. In this experiment, the initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1$ and the initial width of RBF is computed from Equation (5.1) with $\delta = 1$. The proposed algorithm learned this data set in only one epoch. The comparative results of VEBF, RBF, and MLP are illustrated in Table 5.6. From this table, the accuracy of the proposed model is higher than that of RBF and MLP. Moreover, the training time of the proposed model is less than those of RBF and MLP.

5.2 Two-Class Classification Problem

5.2.1 Heart Data Set

The Heart data set consists of 13 attributes in two classes. There are 270 instances in this data set. In this experiment, the VEBF neural network is compared to RBF, MLP, and support vector machine (SVM) with Gaussian kernel. The initial semi-axis lengths of VEBF

Table 5.6 The comparative results trained by Waveform Data Set.

Testing fold	VEBF			RBF			MLP		
	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)	Time (s)	No. of neurons	Accuracy (%)
1	2.65	3	83.82	15.90	3	58.14	2.32	3	74.13
2	3.45	3	85.60	14.71	3	60.10	2.08	3	79.80
3	2.45	3	83.58	14.75	3	60.86	1.83	3	77.08
4	2.46	3	84.50	14.68	3	64.80	7.24	3	78.20
5	2.45	3	85.30	14.78	3	60.50	2.65	3	75.30
Average	2.69	3	84.56	14.96	3	60.88	3.22	3	76.90

is computed from Equation (5.1) with $\delta = 1$. The initial width of RBF is computed from Equation (5.1) with $\delta = 10$. The initial width of the Gaussian kernel of SVM is computed from Equation (5.1) with $\delta = 1$. For the proposed model, the order of training inputs is selected randomly. For the MLP model, the simulation runs 10 times and the best result is chosen. The proposed algorithm learned this data set in only one epoch. The comparative results are illustrated in Figure 5.1. From the results shown in Figure 5.1, the average accuracy of the proposed model is less than that of MLP but higher than those of RBF and SVM. The comparative results of the training time are shown in Table 5.7. From this table, the average training time of the proposed model is less than that of RBF, MLP, and slightly less than the average training time of SVM. Although the average accuracy of MLP is higher than others, the training time of MLP is conversely higher than the rest.

Table 5.7 The comparative results trained by Heart Data Set.

Testing fold	VEBF		RBF		MLP		SVM	
	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons
1	0.21	5	0.97	5	1.15	5	0.68	N/A
2	0.07	2	0.19	2	0.42	2	0.06	N/A
3	0.15	4	0.20	4	0.39	4	0.02	N/A
4	0.13	3	0.17	3	0.37	3	0.02	N/A
5	0.15	5	0.17	5	0.43	5	0.02	N/A
Average	0.14	3.80	0.34	3.80	0.55	3.80	0.16	N/A

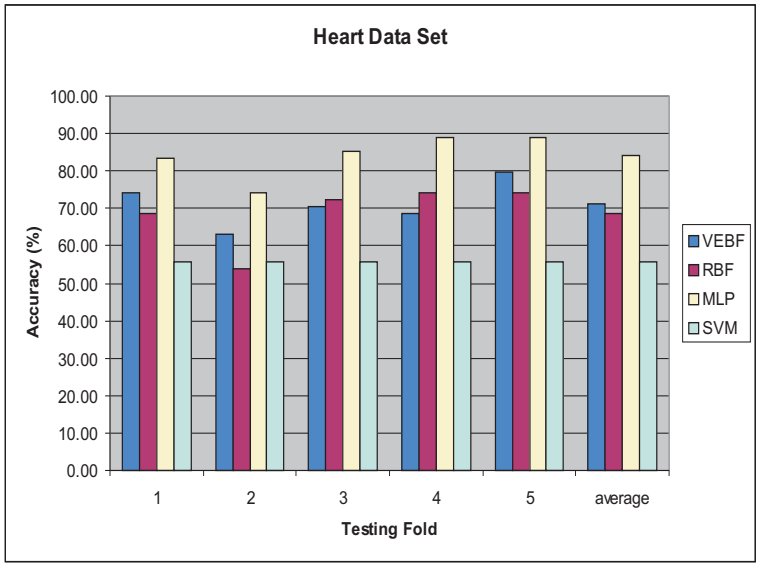


Figure 5.1 The comparative results trained by Heart data set.

5.2.2 Spambase Data Set

The Spambase data set consists of 57 attributes in two classes. There are 4601 instances in this data set. In this experiment, the VEBF neural network is compared to RBF, MLP, and support vector machine (SVM) with Gaussian kernel. The initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1$. The initial width of RBF is computed from Equation (5.1) with $\delta = 10$. The initial width of the Gaussian kernel of SVM is set to 100. For the proposed model, the order of training inputs is selected randomly. The proposed algorithm learned this data set in only one epoch and the comparative results are illustrated in Figure 5.2. From the results shown in Figure 5.2, the average accuracy of the proposed model is slightly less than those of RBF, SVM but less than MLP's. The comparative results of the training time are shown in Table 5.8. From this table, the average time of the proposed model is less than those of RBF and SVM but slightly higher than MLP. It can be seen that when the number of the training data is large, the training time of the SVM is very high as well.

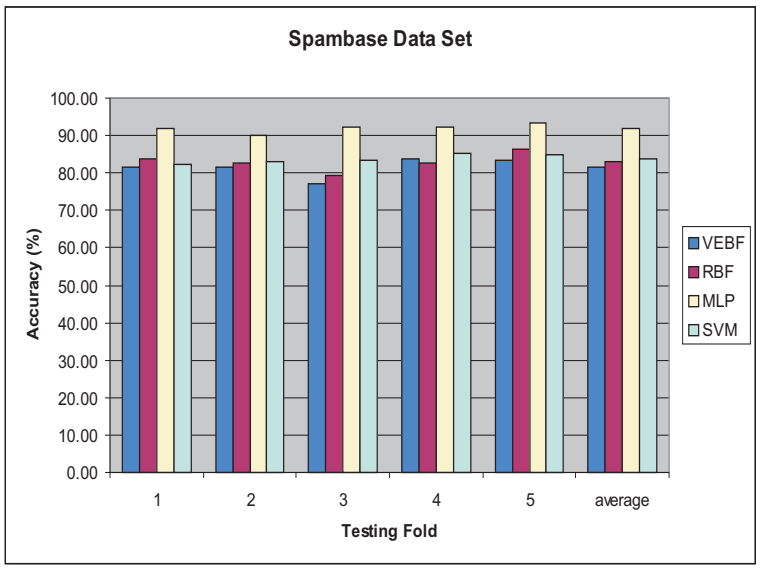


Figure 5.2 The comparative results trained by Spambase data set.

5.2.3 Sonar Data Set

The Sonar data set consists of 60 attributes in two classes. There are 208 instances in this data set. In this experiment, the VEBF neural network is compared to RBF, MLP, and support vector machine (SVM) with Gaussian kernel. The initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1$. The initial width of RBF is computed from Equation (5.1) with $\delta = 10$. The initial width of the Gaussian kernel of SVM is computed from Equation (5.1) with $\delta = 1$. For the proposed model, the order of training inputs is selected randomly. For the MLP model, the simulation runs 10 times and the best result is chosen. The proposed algorithm learned this data in set only one epoch and the comparative results are illustrated in Figure 5.3. From the results shown in Figure 5.3, the average accuracy of the proposed model is slightly less than that of SVM but higher than those of RBF and MLP. The comparative results of the training time are shown in Table 5.9. From this table, the average training time of the proposed model is less than that of MLP but higher than the average training time of RBF and SVM.

Table 5.8 The comparative results trained by Spambase Data Set.

Testing fold	VEBF		RBF		MLP		SVM	
	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons
1	38.31	19.00	37.86	19.00	40.47	19.00	5517.60	N/A
2	37.61	20.00	38.19	20.00	30.44	20.00	3832.60	N/A
3	35.59	18.00	36.59	18.00	19.23	18.00	4806.63	N/A
4	36.82	19.00	37.34	19.00	13.98	19.00	4323.74	N/A
5	35.10	18.00	36.68	18.00	45.37	18.00	4251.20	N/A
Average	36.69	18.80	37.33	18.80	29.90	18.80	4546.35	N/A

Table 5.9 The comparative results trained by Sonar Data Set.

Testing fold	VEBF		RBF		MLP		SVM	
	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons
1	0.47	2.00	0.63	2.00	1.11	2.00	0.81	N/A
2	0.44	2.00	0.16	2.00	0.46	2.00	0.11	N/A
3	0.44	2.00	0.14	2.00	0.45	2.00	0.10	N/A
4	0.43	2.00	0.14	2.00	0.42	2.00	0.08	N/A
5	0.43	2.00	0.13	2.00	0.45	2.00	0.07	N/A
Average	0.44	2.00	0.24	2.00	0.58	2.00	0.23	N/A

5.2.4 Liver Data Set

The Liver data set consists of 7 attributes in two classes. There are 345 instances in this data set. In this experiment, the VEBF neural network is compared to RBF, MLP, and support vector machine (SVM) with Gaussian kernel. The initial semi-axis lengths of VEBF is computed from Equation (5.1) with $\delta = 1$. The initial width of RBF is computed from Equation (5.1) with $\delta = 1$. The initial width of the Gaussian kernel of SVM is computed from Equation (5.1) with $\delta = 1$. For the proposed model, the order of training inputs is selected randomly. For the MLP model, the simulation runs 10 times and the best result is chosen. The proposed algorithm learned this data set in only one epoch and the compara-

Figure 5.3 The comparative results trained by Sonar data set.

tive results are illustrated in Figure 5.4. From the results shown in Figure 5.4, the average accuracy of the proposed model is slightly less than those of RBF and MLP but higher than that of SVM. The comparative results of the training time are shown in Table 5.10. From this table, the average training time of the proposed model is less than that of RBF, MLP, and SVM.

Table 5.10 The comparative results trained by Liver Data Set.

Testing fold	**VEBF**		**RBF**		**MLP**		**SVM**	
	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons	Time (s)	No. of neurons
1	0.21	10.00	0.71	10.00	1.07	10.00	0.67	N/A
2	0.14	9.00	0.21	9.00	0.45	9.00	0.07	N/A
3	0.15	11.00	0.20	11.00	0.45	11.00	0.04	N/A
4	0.11	9.00	0.19	9.00	0.50	9.00	0.04	N/A
5	0.15	8.00	0.19	8.00	0.45	8.00	0.04	N/A
Average	0.15	9.40	0.30	9.40	0.59	9.40	0.17	N/A

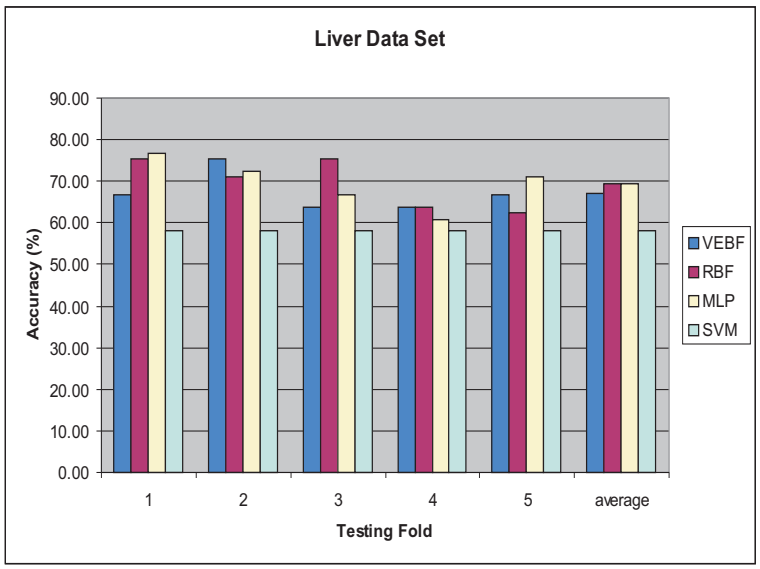


Figure 5.4 The comparative results trained by Liver data set.

5.3 Discussion

In multi-class classification problem, there are five data sets including Iris Data Set, E.coli Data Set, Yeast Data Set, Image Segmentation Data Set, and Waveform Data Set. From the results previously mentioned, it can be seen that the proposed model achieves the higher performance than that of others in four data sets except Image Segmentation Data Set. In this data set, the average accuracy of the proposed model is less than that of the MLP model. Considering the MLP model, the decision function used in the hidden layer and the output layer of MLP is a linear function. Therefore, the distribution of the data in Image Segmentation Data Set is appropriate to a linear classifier. Unfortunately, the proposed model uses a nonlinear classifier. Although the performance of the proposed model is acceptable, it still depends upon the order of training inputs and the initial semi-axis lengths which can affect the number of neurons and its performance.

In two-class classification problem, four data sets are used in this problem which are Heart Data Set, Spambase Data Set, Sonar Data Set, and Liver Data Set. From the results mentioned previously, the average accuracy of the proposed model is higher than that of the RBF model in two data sets which are Heart Data Set and Sonar Data Set. The average accuracy of the proposed model is higher than that of the MLP model in Sonar Data Set.

The average accuracy of the proposed model is higher than that of the SVM model in two data sets which are Heart Data Set and Liver Data Set. Consequently, the proposed model cannot achieve the best performance in two-class classification problem. Considering the function used in each model, the basis function used in the hidden layer of RBF and the kernel function of SVM are a nonlinear function. Therefore, these two models perform a nonlinear mapping from the input space to the output space. If they can map the data from the input space to the output space that can be linearly separable, it might achieve a good performance. Since the decision function used in the output layer of RBF, MLP, and SVM is a linear decision function, the distribution of the data in two-class classification problem may be suitable for the linear decision function. However, the proposed model can incrementally learn the new data set without previous learned data set so it can learn more new data sets without considering the old data set but this cannot be handled by any other models.

Since the different functions are applied to different models, the boundaries of these functions around each group of data are different. Then, the distribution of the data space can affect the performance of these models. Therefore, the distribution of the data space should be further studied for choosing the proper model for a problem.

CHAPTER VI

CONCLUSION

A new elliptic basis function, namely, *Versatile Elliptic Basis Function* (VEBF) based on hyper-ellipsoid and orthonormal basis was proposed. This function can be translated and rotated to cover the data depending on the distribution of the data set in the data space. In addition, a very fast 1-pass-throw-away neural learning algorithm based on the VEBF network was introduced. The network consists of an input layer, a hidden layer, and an output layer. In the hidden layer, the hidden neurons use the VEBF as its basis function. The proposed learning algorithm adjusts the VEBF parameters in only one pass by using only new incoming datum which is not required for subsequent learning. Consequently, this learned datum can be discarded after being learned. For these reasons, this proposed neural network trained by the proposed learning algorithm uses only one epoch to learn a data set. It can learn a new data set without involving the already learned and discarded old data set. The number of neurons in the hidden layer and output layer can be automatically increased during the learning process with some conditions. However, the hidden neurons in the network can be merged into a new neuron whenever some conditions are satisfied. Consequently, the number of redundant neurons in the network can be reduced. The proposed neural network trained by the proposed algorithm can learn the data set in $O(n)$, where n is the number of data. From the experimental results, the proposed model is 7.8 times faster than RBF and 14.5 times faster than MLP in Iris Data Set. In Ecoli Data Set, the proposed model is 3.9 times faster than RBF and 7.2 times faster than MLP. In Yeast Data Set, the proposed model is 4.3 times faster than RBF and 2.7 times faster than MLP. In Image Segmentation Data Set, the proposed model is 3.3 times faster than RBF and 2.7 times faster than MLP. In Waveform Data Set, the proposed model is 5.6 times faster than RBF and 1.2 times faster than MLP. These experimental results signify that the proposed algorithm can learn the data set very fast and the proposed algorithm outperforms the RBF and MLP in multi-class classification problem.

REFERENCES

[1] Meng Joo Er, Shiqian Wu, Juwei Lu, and Hock Lye Toh, ”Face Recognition With Radial Basis Function (RBF) Neural Networks,” *IEEE Trans. Neural Networks*, vol. 13, no. 3, May. 2002.

[2] Suresh S. Salankar, and Dr. Balasaheb M. Patre, ”RBF Neural Network based Model as an Optimal Classifier for the Classification of Radar Returns from the Ionosphere,” in *Proc. IEEE*, 2006.

[3] Douglas Turnbull and Charles Elkan, ”Fast Recognition of Musical Genres Using RBF Networks,” *IEEE Trans. Neural Networks*, vol 17, no. 4, April 2005.

[4] Xiang-Bin Yan, Zhen Wang, Shu-Hua Yu, Yi-Jun Li, ”Time Series Forecasting With RBF Neural Network,” in *Proc. IEEE on Machine Learning and Cybernetics*, August. 2005.

[5] Bor-Shyh Lin, Bor-Shing Lin, Fok-Ching Chong, and Feipei Lai, ”Higher-Order-Statistics-Based Radial Basis Function Networks for Signal Enhancement,” *IEEE Trans. Neural Networks*, vol. 18, no. 3, May. 2007.

[6] Ilias Maglogiannis, Haralambos Sarimveis, C. T. Kiranoudis, Aristotelis A. Chatziioannou, Nikos Oikonomou, and Vassilis Aidinis, ”Radial Basis Function Neural Networks Classification for the Recognition of Idiopathic Pulmonary Fibrosis in Microscopic Images,” *IEEE Trans. Neural Networks*, vol. 12, no. 1, January. 2008.

[7] Henry Leung, Titus Lo, and Sichun Wang, ”Prediction of Noisy Chaotic Time Series Using an Optimal Radial Basis Function Neural Network,” *IEEE Trans. Neural Networks*, vol. 12, no. 5, September. 2001.

[8] Nicolaos B. Karayiannis, ”Reformulated Radial Basis Neural Networks Trained by Gradient Descent,” *IEEE Trans. Neural Networks*, vol. 10, no. 3, May. 1999.

[9] Puneet Singla, Kamesh Subbarao, and John L. Junkins, ”Direction-Dependent Learning Approach for Radial Basis Function Networks,” *IEEE Trans. Neural Networks*, vol. 18, no. 1, January. 2007.

[10] K. Z. Mao, ”RBF Neural Network Center Selection Based on Fisher Ratio Class Separability Measure,” *IEEE Trans. Neural Networks*, vol. 13, no. 5, September. 2002.

[11] Man-Wai Mak, and Sun-Yuan Kung, ”Estimation of Elliptical Basis Function Parameters by the EM Algorithm with Application to Speaker Verification,” *IEEE Trans. Neural Networks*, vol. 11, no. 4, July. 2000.

[12] Jian-Cheng Luo, Qiu-Xiao Chen, Jiang Zheng, Yee Leung Lreis, Yee Leung, and Jiang-Hong Ma, ”An Elliptical Basis Function Network for Classification of Remote-Sensing Images,” in *Proc. IEEE*, 2003.

[13] Richard K. Gordon, and W. Elliott Hutchcraft, ”Using Elliptical Basis Functions in a Meshless Method to Determine Electromagnetic Fields Near Material Interfaces,” in *Proc. IEEE*, 2007.

[14] Jing Luo, Ping-Chen Zai, and Yun-Ni Jian, ”Fault Diagnosis of Power Transformer Based on Ellipsoidal Basis Functional Neural Network,” in *Proc. IEEE on Wavelet Analysis and Pattern Recognition*, 2007.

[15] A. Asuncion and D. Newman, UCI Machine Learning Repository. School of Information and Computer Sciences, Univ. of California, Irvine, 2007.

[16] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[17] Platt, J., ”A Resource-allocating Network for Function Interpolation”, *Neural Computation*, 213-225 (1991).

[18] Kadirkamanathan, V., and Niranjan, M., ”A Function Estimation Approach to Sequential Learning with Neural Networks,” *Neural Computation*, 954-975 (1993).

[19] Yingwei, L., Sundararajan, N., and Saratchandran, P., ”A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function (RBF) Neural Networks,” *Neural Computatation* , 1997.

[20] Li, Y., Sundararajan, N., and Saratchandran, P., ”Analysis of Minimal Radial Basis Function Network Algorithm for Real-time Identification of Nonlinear Dynamic Systems,” *IEEE Proceedings-Control Theory and Applications*, 2000.

[21] Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., and Rosen, D.B., ”Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps,” *IEEE Transactions on Neural Networks*, 1992.

[22] Polikar, R., Upda, L., Upda, S.S., and Honavar, V., ”Learn++: an incremental learning algorithm for supervised neural networks,” *IEEE Transactions on Systems, Man, and Cybernetics*, 2001.

[23] Mu-Chun Su, Jonathan Lee, and Kuo-Lung Hsieh, ”A new ARTMAP-based neural network for incremental learning,” *Neurocomputing*, 2006.

Biography

Name: Mr. Saichon Jaiyen.

Date of Birth: 24^{th} April, 1977.

Educations:

- Ph.D., Program in Computer Science, Department of Mathematics, Chulalongkorn University, Thailand, 2011.
- M.Sc. Program in Computational Science, Faculty of Science, Chulalongkorn University, Bangkok, Thailand, 2004.
- B.Sc. Program in Mathematics, Faculty of Science, Kasetsart University, Bangkok, Thailand, 2000.

Publication papers:

- Saichon Jaiyen, Chidchanok Lursinsap, and Suphakant Phimoltares. A Very Fast Neural Learning for Classification Using Only New Incoming Datum. *IEEE Transactions on Neural Networks*, vol. 21, no. 3, pp. 381 - 392, March 2010.
- Saichon Jaiyen, Chidchanok Lursinsap, and Suphakant Phimoltares. A Versatile Hyper-Ellipsoidal Basis Function for Function Approximation in High Dimensional Space. *Lecture Notes in Computer Science: Advances in Neural Networks*, pp. 756-765, 2009.

Scholarship: The Royal Golden Jubilee Ph.D. Program.