CHAPTER II

VERSATILE ELLIPTIC BASIS FUNCTION NEURAL NETWORK (VEBFNN)

Versatile Elliptic Basis Function Neural Network was introduced by Jaiyen et al. [4]. In Chapter II, the versatile elliptic basis function, structure of VEBFNN with basic elements, how to obtain the set of orthonormal basis vectors and the previous VEBF learning algorithm proposed by [4] are given as follows:

2.1 Versatile Elliptic Basis Function (VEBF)

Given the *n*-dimensional space, the hyperellipsoidal equation in rectangular coordinate system can be mathematically expressed by

$$\sum_{i=1}^{n} \frac{x_i^2}{w_i^2} = \frac{x_1^2}{w_1^2} + \frac{x_2^2}{w_2^2} + \ldots + \frac{x_n^2}{w_n^2} = 1\,, \tag{1}$$

where w_i is scalar representing the width value of i^{th} axis for the hyperellipsoidal, and each scalar x_i from Equation (1) can be viewed as the scalar projection of vector $\mathbf{x} = [x_1\, x_2 \cdots x_n]^T$ on to *n* standard basis vectors $\mathbf{u}_1 = [1\ 0 \cdots 0]^T$, $\mathbf{u}_2 = [0\ 1 \cdots 0]^T$,..., $\mathbf{u}_n = [0\ 0 \cdots 1]^T$. Given any orthonormal basis vectors $\{\mathbf{u}_i\}_{i=1}^{n}$, the scalar x_i is expressed as

$$x_i = \mathbf{x}^T \mathbf{u}_i \tag{2}$$

Based on Equations (1) and (2), the hyperellipsoidal equation for given orthonormal basis vectors $\{\mathbf{u}_i\}_{i=1}^{n}$ is mathematically expressed by

$$\sum_{i=1}^{n} \frac{(\mathbf{x}^T\mathbf{u}_i)^2}{w_i^2} = \frac{(\mathbf{x}^T\mathbf{u}_1)^2}{w_1^2} + \frac{(\mathbf{x}^T\mathbf{u}_2)^2}{w_2^2} + \ldots + \frac{(\mathbf{x}^T\mathbf{u}_n)^2}{w_n^2} = 1\,, \tag{3}$$

From Equation (1), the ellipsoidal is located on the origin as its center. This form can be generalized by translation of axis method. Suppose $\mathbf{x} = [x_1\, x_2 \cdots x_n]^T$ is a vector corresponding to the original axes. Let $\{\mathbf{u}_i\}_{i=1}^{n}$ be the set of n basis vectors. The original axes of the hyperellipsoidal are translated from the origin to the coordinates of $\mathbf{c} = [c_1\, c_2 \cdots c_n]^T \in \mathbb{R}^n$. The new coordinates of vector $\mathbf{x}$ denoted by $\mathbf{x}' = [x_1'\, x_2' \cdots x_n']^T$ is written by

$$x_i' = x_i - c_i, \text{ for } i = 1, 2, \ldots, n \tag{4}$$

Therefore, the hyperellipsoidal equation in the new axes becomes

$$\sum_{i=1}^{n} \frac{((\mathbf{x}\text{-}\mathbf{c})^T \mathbf{u}_i)^2}{w_i^2} = 1 , \tag{5}$$

From Equation (5), the Versatile Elliptic Basis Function (VEBF) is defined as follows:

$$\psi(\mathbf{x}) = \sum_{i=1}^{n} \frac{((\mathbf{x}-\mathbf{c})^T \mathbf{u}_i)^2}{(w_i)^2} - 1 \tag{6}$$

where w_i is the width of the VEBF along the i^{th} axis , $\mathbf{c}$ and $\{\mathbf{u}_i\}_{i=1}^{n}$ are the center vector and the set of orthonormal basis vectors of the VEBF, respectively.

2.2 Structure of Versatile Elliptic Basis Function Neural Network (VEBFNN)

A Versatile Elliptic Basis Function Neural Network (VEBFNN) consists of three distinctive layers called input, hidden and output layers. The input layer is constituted from a data point. The number of neurons in the input layer is equal to the number of attributes. The output layer contains a set of neurons which the number of output neurons is equal to the number of class labels. The hidden layer comprises of the set of sub-hidden layers. Each sub-hidden layer, containing a group of neurons, corresponds to each class label. The structure of VEBF neural network is illustrated in Figure 1. The dashed boxes are sub-hidden layers.

Given an n-dimensional data set with class label $\mathbb{X} = \{(\mathbf{x}_i, t_i) \in \mathbb{R}^n \times \mathbb{I}^+ \mid 1 \le t_i \le r\}$, let $\Gamma = \{\Lambda^1, \Lambda^2, \ldots, \Lambda^r\}$ be a hidden layer containing r sub-hidden layers. The k^{th} sub-hidden layer called Λ^k responsible for data points with the class label

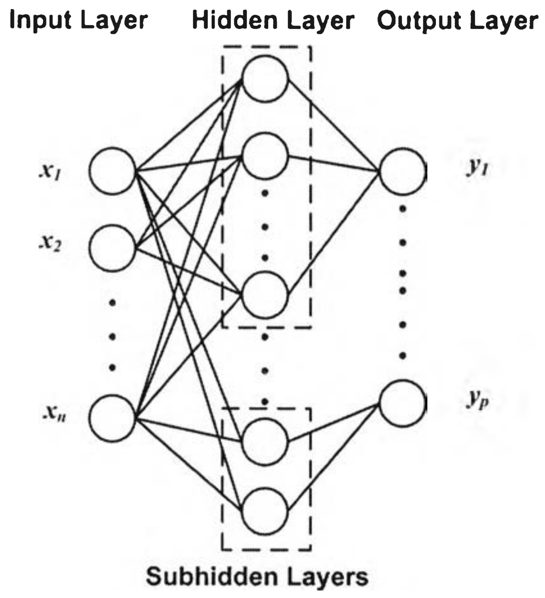

Figure 1: The structure of versatile elliptic basis function neural network.

k and contains the group of hidden neurons expressed as $\Lambda^k = \{\Omega_1^k, \Omega_2^k \ldots, \Omega_{d_k}^k\}$, where Ω_j^k is the j^{th} hidden neuron in the k^{th} sub-hidden layer and d_k is the number of hidden neurons in the k^{th} sub-hidden layer. The Ω_j^k is identified by $\Omega_j^k = (\overline{\mathbf{x}}_j^k, \mathbf{w}_j^k, \mathbf{S}_j^k, N_j^k)$, where $\overline{\mathbf{x}}^j$, $\mathbf{w}_j^k$, $\mathbf{S}_j^k$ and N_j^k are the center vector, width vector, covariance matrix and total number of belonging data of the neuron Ω_j^k. Each hidden neuron applies a Versatile Elliptic Basis Function (VEBF) as an activation function. The VEBF of Ω_j^k is expressed by

$$\psi_j^k(\mathbf{x} : \overline{\mathbf{x}}_j^k, \mathbf{w}_j^k, \mathbf{U}_j^k) = \sum_{l=1}^{n} \frac{((\mathbf{x} - \overline{\mathbf{x}}_j^k)^T \mathbf{u}_{jl}^k)^2}{(w_{jl}^k)^2} - 1 \qquad (7)$$

where $\overline{\mathbf{x}}_j^k = [\overline{x}_{j1}^k \ \overline{x}_{j2}^k \ \cdots \ \overline{x}_{jn}^k]^T \in \mathbb{R}^n$ is a center vector, $\mathbf{U}_j^k$ is a column matrix of orthonormal basis vectors corresponding to the covariance matrix $\mathbf{S}_j^k$ and $\mathbf{w}_j^k = [w_{j1}^k \ w_{j2}^k \ \cdots \ w_{jn}^k]^T \in \mathbb{R}^n$ is a width vector. Based on Equation (7), the definition of a covered data is defined by

Definition 1. For a given input vector $\mathbf{x}_i \in \mathbb{R}^n$, it is said that $\mathbf{x}_i$ is covered by the neuron Ω_j^k if and only if the corresponding versatile elliptic basis function value is less than or equal zero, i.e. $\psi_j^k(\mathbf{x}_i : \overline{\mathbf{x}}_j^k, \mathbf{w}_j^k, \mathbf{U}_j^k) \leq 0$.

The output of the k^{th} output neuron $o^k(\mathbf{x}_i)$ in the output layer is defined as follows:

$$o^k(\mathbf{x}_i) = \min(\{\psi_1^k(\mathbf{x}_i), \psi_2^k(\mathbf{x}_i), \ldots, \psi_{d_k}^k(\mathbf{x}_i)\}), \qquad k = 1, 2, \ldots, d_k \tag{8}$$

The *decision function* $F(\mathbf{x}_i)$ for assigning the class label of the input vector $\mathbf{x}_i$ is defined by

$$F(\mathbf{x}_i) = \min(\{o^1(\mathbf{x}_i), o^2(\mathbf{x}_i), \ldots, o^r(\mathbf{x}_i)\}) \tag{9}$$

For learning process, the structure of VEBF network adapts itself to data distribution by creating the new hidden neuron or adjusting neuron parameters, incrementally and automatically. This process is performed repeatedly until all learning data points are completely covered.

2.2 Orthonormal Basis Computation

In this research, the set of orthonormal basis vectors of a VEBF is derived by Principal Component Analysis (PCA) technique. Principal Component Analysis (PCA), also known as the *Karhunen-Loeve* transformation in communication theory [31], is statistical procedure used to transform a set of data points of possibly correlated inputs in input space into the new feature space in which linearly uncorrelated features, called principal components, are obtained by orthogonal transformation. The transformation is performed in such the way that the first principal component maximizes the variance, and then each succeeding component provides the maximum variance under the orthogonal components condition. The concepts of PCA and orthonormal basis vector computation are provided:

2.2.1 Principal Component Analysis (PCA)

Let $\mathbf{x}$ denote an *n*-dimensional random vector with zero mean identified by

$$E[\mathbf{x}] = \bar{0} \tag{10}$$

where $E[*]$ is the statistical expectation operator and $\bar{0}$ denote the zero vector. Let $\mathbf{u}$ denote a unit vector onto which the vector $\mathbf{x}$ is to be projected. The projection is defined by

$$a = \mathbf{x}^T\mathbf{u} = \mathbf{u}^T\mathbf{x} \tag{11}$$

Based on Equation (10), the mean value of a is also zero, $E[a] = 0$. The variance of a is therefore given by

$$\sigma^2 = E[a^2] \tag{12}$$

From Equations (11) and (12), the variance is given as

$$\begin{aligned}\sigma^2 &= E[(\mathbf{u}^T\mathbf{x})(\mathbf{x}^T\mathbf{u})]\\ &= E[(\mathbf{u}^T\mathbf{x})(\mathbf{x}^T\mathbf{u})]\\ &= \mathbf{u}^T E[\mathbf{x}\mathbf{x}^T]\mathbf{u}\\ &= \mathbf{u}^T\mathbf{S}\mathbf{u}\end{aligned}$$

Thus,

$$\sigma^2 = \mathbf{u}^T\mathbf{S}\mathbf{u}\,. \tag{13}$$

The *n*-by-*n* matrix $\mathbf{S}$ is the covariance matrix of random vector $\mathbf{x}$. From Equation (13), the variance σ^2 of the projection a is the function of the unit vector $\mathbf{u}$ given as:

$$f(\mathbf{u}) = \sigma^2 = \mathbf{u}^T\mathbf{S}\mathbf{u} \tag{14}$$

The function $f(\mathbf{u})$ is called a *variance probe*. To find the set of unit vectors $\mathbf{u}$ along which $f(\mathbf{u})$ has extremal or stationary values with respect to a constraint on the Euclidean norm ($\|\mathbf{u}\|$). If $\mathbf{u}$ is a unit vector such that $f(\mathbf{u})$ has an extreme value, then for any small $\Delta\mathbf{u}$, it leads to that

$$f(\mathbf{u} + \Delta\mathbf{u}) = f(\mathbf{u}) \tag{15}$$

From Equation (14) and (15), $f(\mathbf{u} + \Delta\mathbf{u})$ is expressed as follows

$$f(\mathbf{u} + \Delta\mathbf{u}) = \mathbf{u}^T\mathbf{S}\mathbf{u} + 2(\Delta\mathbf{u})^T\mathbf{S}\mathbf{u} + (\Delta\mathbf{u})^T\mathbf{S}\Delta\mathbf{u} \tag{16}$$

Since the $\Delta\mathbf{u}$ is a small value, the second-order term $(\Delta\mathbf{u})^T\mathbf{S}\Delta\mathbf{u}$ is ignored,

$$f(\mathbf{u} + \Delta\mathbf{u}) = f(\mathbf{u}) + 2(\Delta\mathbf{u})^T\mathbf{S}\mathbf{u} \tag{17}$$

Hence, by applying Equations (15) and (16) implies that

$$(\Delta\mathbf{u})^T\mathbf{S}\mathbf{u} = 0 \tag{18}$$

Not any $\Delta\mathbf{u}$ of $\mathbf{u}$ is admissible. The constraint of $\Delta\mathbf{u}$ is based on the Euclidean norm of $\mathbf{u} + \Delta\mathbf{u}$ by $\|\mathbf{u} + \Delta\mathbf{u}\| = 1$ or equivalently $\|\mathbf{u} + \Delta\mathbf{u}\| = 1$.

$$\|\mathbf{u} + \Delta\mathbf{u}\| = 1 \text{ or } (\mathbf{u} + \Delta\mathbf{u})^T(\mathbf{u} + \Delta\mathbf{u}) = 1$$

Since $\mathbf{u}^T\mathbf{u}=1$ and the $\Delta\mathbf{u}$ is a small value, it obtains that

$$(\Delta\mathbf{u})^T\mathbf{u}=0 \qquad (19)$$

This means that the $\Delta\mathbf{u}$ must be orthogonal to $\mathbf{u}$. From Equations (18) and (19), it obtains that

$$(\Delta\mathbf{u})^T\mathbf{Su}\text{-}\lambda(\Delta\mathbf{u})^T\mathbf{u}=0$$

$$(\Delta\mathbf{u})^T(\mathbf{Su}\text{-}\lambda\mathbf{u})=0$$

$$\mathbf{Su}\text{-}\lambda\mathbf{u}=0$$

$$\mathbf{Su}=\lambda\mathbf{u} \qquad (20)$$

Equation (20) is recognized as the eigenvalue problem. The vector $\mathbf{u}$ and scalar λ are called the eigenvector and corresponding eigenvalue, respectively.

2.2.2 Orthonormal basis vectors by PCA

The orthonormal basis vectors of each VEBF neuron are obtained by the direction of data distribution captured by the set of principal components. The original axes of the input data space are translated into the coordinate of the mean of the data set. The translated axes are rotated by applying PCA procedure. Given a data set $\mathbf{X}\subset\mathbb{R}^n$, the concept of the orthonormal basis vector by PCA procedure can be summarized as follows:

1. Compute the mean vector ($\bar{\mathbf{x}}$)
2. Compute the covariance matrix ($\mathbf{S}$) of the data set $\mathbf{X}$.
3. Compute the set of eigenvalues and corresponding eigenvectors $\{\lambda_i,\mathbf{u}_i\}_{i=1}^n$ of covariance matrix $\mathbf{S}$, where $\lambda_1>\lambda_2>\ldots>\lambda_n$.

The obtained eigenvectors are gathered to form a column matrix, $\mathbf{U}=[\mathbf{u}_1\ \mathbf{u}_2\ \cdots\mathbf{u}_n]$, of orthonormal basis vectors as the local space of a VEBF neuron.

2.3 VEBF Learning Algorithm for one incoming datum

Let $\mathbf{X}=\{(\mathbf{x}_i,t_i)\mid \mathbf{x}_i \in \mathbb{R}^n$ and $i=1,2,\ldots,N\}$ be a finite set of training data, where $\mathbf{x}_i$ is a feature vector referred to as a *data vector* and r_i is the class label of the vector. Let $\Lambda=\{\Omega_1,\Omega_2,\ldots,\Omega_K\}$ be a set of K hidden neurons in VEBFNN. Each of $\Omega_{k\in\{1,2,\ldots,K\}}$ contains the center vector $\mathbf{c}_k$ whose dimension is equal to that of the data vector, the width vector $\mathbf{a}_k$, the covariance matrix $\mathbf{S}_k$, the number of data in the node N_k, and the corresponding class label cl_k, as a 5-tuple $\Omega_k=(\mathbf{c}_k,\mathbf{a}_k,\mathbf{S}_k,N_k,cl_k)$. Let N_0 be a constant for adjusting the width vectors. If there is no hidden neuron in the network, K is set by $K=0$. The θ be a threshold for merging the two hidden neurons in the network. The learning algorithm for VEBFNN can be given as follows.

VEBF Learning Algorithm for one incoming datum:

Step 1: Initialize the width vector $\mathbf{a}_0=[a_1,a_2,\ldots,a_n]^T$.

Step 2: Present the training data with class label $(\mathbf{x}_i,t_i)$ to the VEBFNN.

Step 3: **If** $K\neq 0$ **then** find a hidden neuron Ω_k assigned for the class t_i such that $k=\arg\max_{l\in\{1,2,\ldots,K\}}(\|\mathbf{x}_i-\mathbf{c}_l\|)$.

a) Compute the new center vector $\mathbf{c}'_k$ by

$$\mathbf{c}'_k=\frac{N_k}{N_k+1}\mathbf{c}_k+\frac{\mathbf{x}_i}{N_k+1}.$$

b) Compute the new covariance matrix $\mathbf{S}'_k$ by

$$\mathbf{S}'_k=\frac{N_k}{N_k+1}\mathbf{S}_k+\frac{\mathbf{x}_i\mathbf{x}_i^T}{N_k+1}-\mathbf{c}'_k\mathbf{c}'^T_k+\mathbf{c}_k\mathbf{c}_k^T-\frac{\mathbf{c}_k\mathbf{c}_k^T}{N_k+1}$$

Else Set $K=K+1$ and create new hidden neuron by setting

a) $\mathbf{c}_k=\mathbf{x}_i$,

b) $\mathbf{S}_k=\bar{\mathbf{0}}$ where $\bar{\mathbf{0}}$ is zero matrix,

c) $N_k=1$,

d) $cl_k=t_i$,

e) $\mathbf{a}_k=\mathbf{a}_0$,

f) $\Lambda=\Lambda\cup\{\Omega_k\}$

g) Remove $(\mathbf{x}_i, t_i)$ from the training data set and go to Step 8.

Step 4: Compute the orthonormal basis for Ω_k based on the covariance matrix $\mathbf{S}_k$.

Step 5: Compute $\psi_k(\mathbf{x}_i : \mathbf{c}'_k, \mathbf{w}_k, \mathbf{U}_k)$ based on the new center $\mathbf{c}'_k$.

Step 6: If $\psi_k(\mathbf{x}_i : \mathbf{c}'_k, \mathbf{w}_k, \mathbf{U}_k) < 0$ **then** update parameters of Ω_k by

a) If $N_k > N_0$ then update the width vector $\mathbf{a}_k$ by

$$a_i = a_i + \left|(\mathbf{c}'_k - \mathbf{c}_k)^T \mathbf{u}_i\right|,\ i = 1, \ldots, n$$

b) Update the covariance matrix $\mathbf{S}_k = \mathbf{S}'_k$,

c) Update $N_k = N_k + 1$,

d) Update the center vector $\mathbf{c}_k = \mathbf{c}'_k$.

Else Set $K = K + 1$ and create new hidden neuron by setting

a) $\mathbf{c}_k = \mathbf{x}_i$,

b) $\mathbf{S}_k = \overline{\mathbf{0}}$ where $\overline{\mathbf{0}}$ is zero matrix,

c) $N_k = 1$,

d) $cl_k = t_i$,

e) $\mathbf{a}_k = \mathbf{a}_0$,

f) $\Lambda = \Lambda \cup \{\Omega_k\}$.

Step 7: Compute $\psi_k(\mathbf{c}_l : \mathbf{c}_k, \mathbf{w}_k, \mathbf{U}_k) < \theta$ and $\psi_l(\mathbf{c}_k : \mathbf{c}_l, \mathbf{w}_l, \mathbf{U}_l) < \theta$ for $l = 1, 2, \ldots, n$

Step 8: If $\psi_k(\mathbf{c}_l : \mathbf{c}_k, \mathbf{w}_k, \mathbf{U}_k) < \theta$ or $\psi_l(\mathbf{c}_k : \mathbf{c}_l, \mathbf{w}_l, \mathbf{U}_l) < \theta$ for $l = 1, 2, \ldots, n$ **then** do the following steps:

a) Merge hidden neurons Ω_k and Ω_l into one hidden neuron Ω_m

- $\mathbf{c}_m = \dfrac{1}{N_k + N_l}(N_k\mathbf{c}_k + N_l\mathbf{c}_l)$,
- $\mathbf{S}_m = \dfrac{N_k}{N_k + N_l}\mathbf{S}_k + \dfrac{N_k}{N_k + N_l}\mathbf{S}_l + \dfrac{N_k}{(N_k + N_l)^2}(\mathbf{c}_k\text{-}\mathbf{c}_l)(\mathbf{c}_k\text{-}\mathbf{c}_l)^T$,
- $N_m = N_k + N_l$,

- $a_i = \sqrt{2\pi\lambda_i}$, $i = 1, 2, \ldots, n$ where λ_i is the i^{th} eigenvalue of the covariance matrix $\mathbf{S}_m$,

- $t_m = t_k$

b) Add Ω_m into the network and delete Ω_k and Ω_l from the network

c) Set $K = K - 1$.

Step 9: **If** the training data set is not empty **then** go to Step 2, **else** stop training.