CHAPTER III

MATHEMATICAL MODEL AND ALGORITHM

3.1 Objectives

The main objectives of this study are:

1. To develop a new radial basis function for adjusting weight with less computation, less hidden units, less computation time, and the same or higher accuracy of data learning.

2. To develop a mathematical model for the fault immunization of the center vector of a radial basis function.

3.2 Scope of Work and Constraints

This study is constrained by following conditions:

1. A RBF network is used for pattern classification.

2. The data in the test sets are in a two dimensional space.

3.3 Mathematical Model of Generic Elliptic Radial Basis Function (GERBF)

As mentioned in Chapter II, the mathematical model of a RBF neural network is a type of supervised learning. Prior to further discussion, the learning process of the

RBF network should be briefly summarized. There are two most common approaches for constructing a RBF network for any pattern classification.

The first approach is by predefining the structure of the network consisting of one input layer, one hidden layer, and one output layer [1]. Each hidden neuron performs its classifying task based on a radial basis function, which is normally realized by a Gaussian distribution function. An output neuron generates its output value by computing the dot product between its weight vector and input vectors obtained from the hidden neurons. This approach is simple but it may be suffered from the costly computational time and a large number of redundant hidden neurons.

The second approach, called Resource Allocation Network (RAN), is by gradually allocating a RBF neuron to the network in the hidden layer to cover those specific data vectors of the same class [2, 4]. The activation function of each hidden neuron is also realized by a Gaussian distribution function. The allocation is terminated after all data vectors in the same class are covered by some hidden RBF neurons. This approach is more efficient than the first one. During the RAN allocation, the center and radius of the allocated neuron can be easily computed by several weight adjusting techniques based on a defined cost function, e.g. gradient descent learning or Kalman filtering [3]. Some redundant neurons can be automatically pruned [4].

Both approaches are realized by a Gaussian distribution function with the same width in all dimensions. Zhu, Cai, and Liu [4] improved the RBF learning by combining the multivariate Gaussian function and maximum likelihood classification. The shape of the multivariate Gaussian function can be rotated, shrunk, or stretched accordingly to the density distribution and eigen-direction of the data vectors. This process is rather ex-

pensive due to the covariance matrix computation and inversion of the data vectors.

Fig. 3.1 shows the example of Exclusive or (XOR) problem. There are two classes of data. The circles represent the hidden units which are realized by the Gaussian distribution function, and the dotted ellipse represents the hidden unit which is realized by a multivariate Gaussian function. It is obvious that the Gaussian distribution function needs two hidden units to classify the data while the multivariate Gaussian function needs one hidden unit.

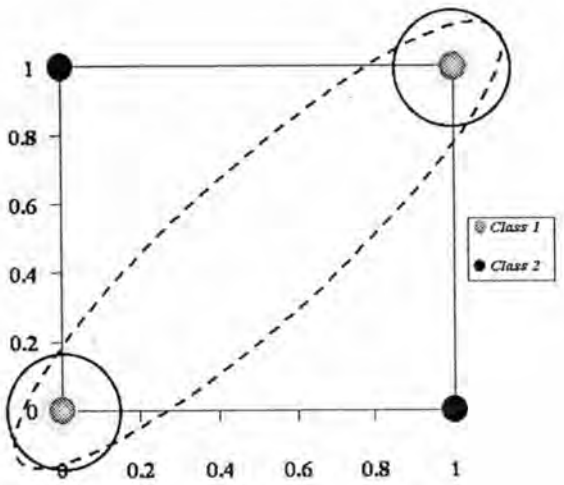


Figure 3.1: XOR Problem

To overcome the costly computation, a new generic elliptic radial basis function (GERBF) with adjustable center and dimensional widths is introduced. The location of the center and the values of all dimensional widths are adjusted using the gradient descent learning rule, which will be described with a cost function in the next topic.

Since the translation, rotation and dimensional width scaling of a radial basis function are the most essential features in order to perform any pattern classification, our proposed generic elliptic basis function must also preserve the properties of translation, rotation

and dimensional width scaling of an RBF. We define the following notations.

$$\begin{aligned}
\mathbf{X} &= [x_1, x_2, \ldots, x_n]^T && : \text{input vector} \\
\mathbf{a} &= [a_1, a_2, \ldots, a_n]^T && : \text{center vector} \\
\mathbf{b} &= [b_1, b_2, \ldots, b_n]^T && : \text{width vector} \\
\mathbf{w} &= (w_{i,j}); \quad 1 \leq i, j \leq n && : \text{rotation matrix} \\
n & && : \text{dimension} \\
i, j & && : \text{indices of dimensions}
\end{aligned}$$

The generic elliptic radial basis function (GERBF) is a composite function consisting of the following two functions.

$$h_k(\mathbf{a}, \mathbf{b}, \mathbf{w}) = \sum_{i=1}^{n} \frac{(\sum_{j=1}^{n} w_{i,j}(x_j - a_j))^2}{b_i^2} - 1 \tag{3.1}$$

$$o_k(h_k) = \frac{1}{1 + e^{-\beta h_k}} \tag{3.2}$$

where $h_k(\mathbf{a}, \mathbf{b}, \mathbf{w})$ is the elliptic activation function of neuron k and $o_k(h_k)$ is the sigmoid function of neuron k. The center of each GERBF function is controlled by $\mathbf{a}$ and $\mathbf{b}$. The shape of the function is rotated in any angle by changing the value of $\mathbf{w}$. The steepness of the sigmoid function is controlled by a constant β and is used to determine whether a given data vector is covered by the elliptic function. If it is covered by this function then the value of $o_k(h_k)$ is approximately one. Otherwise, the value of $o_k(h_k)$ is approximately zero. To simplify the notations, $h_k(\mathbf{a}, \mathbf{b}, \mathbf{w})$ and $o_k(h_k)$ will be shorten as h_k and o_k, respectively.

3.3.1 Cost Function

During learning, the data vectors of the same class will be covered by a set GERBFs. The classification is correct if, for any GERBF neuron, all data vectors covered by this neuron are in the same class. Suppose there are m input data vectors of the same class. This condition is captured by the following cost function.

$$c = \frac{1}{2}(1 - \prod_{i=1}^{m} \phi_i)^2 \tag{3.3}$$

where ϕ_i is the number of GERBFs covering $\mathbf{x}_i$. The cost function is minimum when every data vector of the same class is covered by only one GERBF. The parameters **a**, **b**, and **w** are adjusted by the gradient descent learning rule.

After all data vectors of the same class are covered by some GERBF neurons, the pruning procedure is performed. Since the problem of pruning all redundant neurons can be viewed as a problem of finding minimum sets cover, which is NP-complete, any existing heuristic algorithm can be applied to this problem.

3.4 Mathematical Model of GERBF Fault Immunization Model

According to the previous discussion on the GERBF network learning, it is obvious that the position of a hidden unit is critical to fault immunization of the data. If a hidden unit is located at the right position, more fault immunization will be occurred. The reliability of a GERBF neuron depends upon the changes of the center vector **a**, the dimensional width vector **b**, and the rotational matrix **w**. In this thesis, we concern only the change of the center vector **a** in a two dimensional space. The center vector is considered as the most important parameter compared with the width vector **b**, and the rotational matrix **w**, because once it is located at the right position, the adjustment of the other parameters could be performed consequently. The fault immunization of this center vector can be mathematically defined as follows.

Let $\mathbf{X}_i^{(\alpha)} = (x_{i,1}^{(\alpha)}, x_{i,2}^{(\alpha)})$ be a data vector i covered by a GERBF neuron α and $\mathbf{Y}_j^{(\beta)} = (y_{j,1}^{(\beta)}, y_{j,2}^{(\beta)})$ a data vector j covered by a GERBF neuron β. Neurons α and β cover data

vectors of different classes. Suppose $\mathbf{X}_i^{(\alpha)}$ and $\mathbf{Y}_j^{(\beta)}$ are two nearest data vectors from different classes, called a nearest data vector pair. The line connecting $\mathbf{X}_i^{(\alpha)}$ and $\mathbf{Y}_j^{(\beta)}$ cuts the boundary of the GERBF neuron α at location $\mathbf{Z}_k^{(\alpha)} = (z_{k,1}^{(\alpha)}, z_{k,2}^{(\alpha)})$.

Definition 1. *The inside distance, din_i, of $\mathbf{X}_i^{(\alpha)}$ with respect to $\mathbf{Z}_k^{(\alpha)}$ is equal to* $||\mathbf{X}_i^{(\alpha)} - \mathbf{Z}_k^{(\alpha)}||$.

Definition 2. *The outside distance, $dout_j$, of $\mathbf{Y}_j^{(\beta)}$ with respect to $\mathbf{Z}_k^{(\alpha)}$ is equal to* $||\mathbf{Y}_j^{(\beta)} - \mathbf{Z}_k^{(\alpha)}||$.

The value of din_i indicates the distance that has no other data vector besides $\mathbf{X}_i^{(\alpha)}$ lying from the location of $\mathbf{X}_i^{(\alpha)}$ to the boundary location of $\mathbf{Z}_k^{(\alpha)}$. Similarly, the value of $dout_j$ indicates the distance that has no other data vector besides $\mathbf{Y}_j^{(\beta)}$ lying from the location of $\mathbf{Y}_j^{(\beta)}$ to the boundary location of $\mathbf{Z}_k^{(\alpha)}$. When a GERBF neuron is implemented on a VLSI chip, the location of the center of this neuron can be changed by some uncontrollable phenomena such as heat or electromagnetic field. It implies that the location of $\mathbf{Z}_k^{(\alpha)}$ must be accordingly changed. Based on this fact, we formulate the problem of fault immunization of a GERBF neuron as follows.

Problem Formulation

Given a GERBF neuron of class α and its nearest GERBF neurons, find a new location of the center vector $(a_1^{(\alpha)}, a_2^{(\alpha)})$ such that

$$E_{fault} = \sum_{i,j \in M} (din_i - dout_j)^2$$

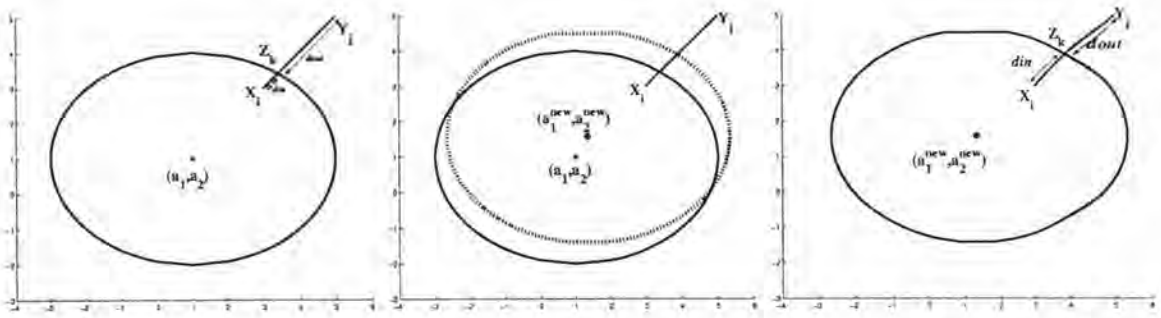
is minimum. M is a set of all nearest data vector pairs.

The success of fault immunization of a GERBF neuron k is measured in terms of immunization degree defined by the relation of din_i and $dout_j$, as follows.

Definition 3. *The immunization degree with respect to a GERBF unit k, F_k, is*

$$F_k = \frac{1}{||M_k||} \sum_{(i,j)\in M_k} \min\left(\frac{din_i}{dout_j}, \frac{dout_j}{din_i}\right)$$

where M_k is the set of all indices of the nearest data vector pairs covered by the GERBF unit k.



(a) (b) (c)

Figure 3.2: Fault immunization process

Fig. 3.2 shows the process of fault immunization. Fig. 3.2(a) shows the GERBF hidden unit whose center is at (a_1, a_2), and a nearest data vectors pair is $\mathbf{X}_i$ and $\mathbf{Y}_j$. In Fig. 3.2(b), the dotted ellipse represents a new location of the hidden unit whose new center is (a_1^{new}, a_2^{new}). Fig. 3.2(c) shows that the new values of *din* and *dout* which imply that the immunization degree (F_k) is obviously increased.

Consider an example of immunization degree in Fig. 3.3, the center is still not at the optimal position whereas the Fig. 3.4 shows the optimal position of the center vector that improves the immunization degree (F_k) of each hidden unit. The members of each class remain the same.

From the Fig. 3.3, there are two GERBF hidden units which are represented by ellipses, and the nearest data vectors pairs which are represented by the dotted lines.

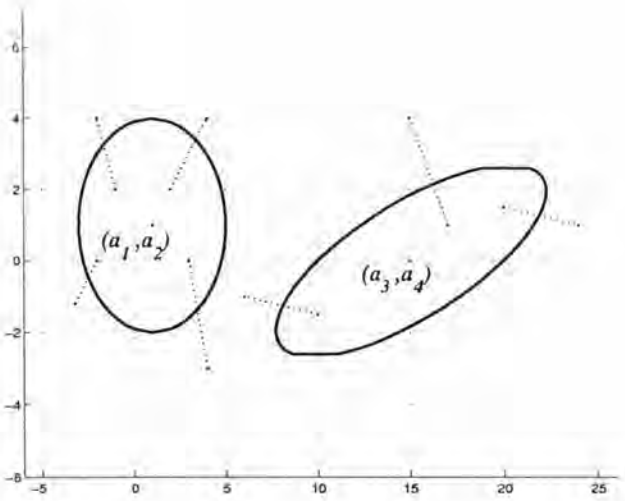


Figure 3.3: An example of the immunization degree when center vector is not at the optimal location

The left hidden unit has four nearest data vectors pairs and the center at (a_1, a_2). The right hidden unit has three nearest data vectors pairs and the center at (a_3, a_4). In the Fig. 3.4, the center of the left and right hidden units are relocated to (a_1^{new}, a_2^{new}) and (a_3^{new}, a_4^{new}), respectively.

From the Eqs. 3.1, the center, which is in a two dimensional space, of the GERBF hidden unit is at (a_1, a_2). Consider the boundary location $\mathbf{Z}_k^{(\alpha)}$. The value of $h_k(\mathbf{a}, \mathbf{b}, \mathbf{w})$ is zero at the boundary of a GERBF hidden unit. Thus,

$$h_k(\mathbf{a}, \mathbf{b}, \mathbf{w}) = \sum_{i=1}^{2} \frac{(\sum_{j=1}^{2} w_{i,j}(z_{k,j}^{(\alpha)} - a_j))^2}{b_i^2} - 1 = 0$$

or

$$\sum_{i=1}^{2} \frac{(\sum_{j=1}^{2} w_{i,j}(z_{k,j}^{(\alpha)} - a_j))^2}{b_i^2} = 1$$

or

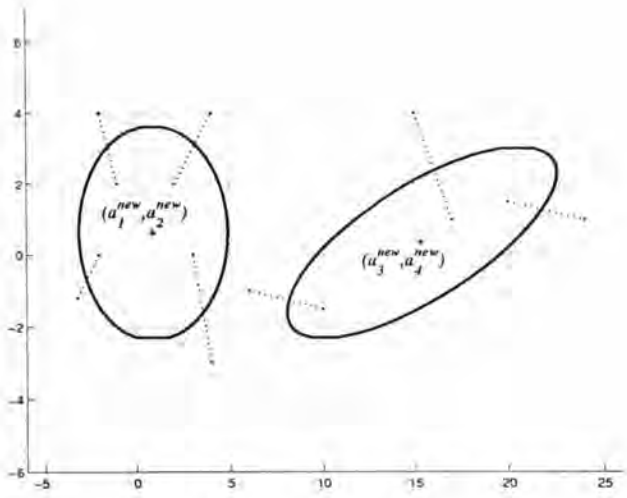


Figure 3.4: An example of the immunization degree when center vector is at the optimal location

$$\frac{(w_{1,1}(z_{k,1}^{(\alpha)} - a_1) + w_{1,2}(z_{k,2}^{(\alpha)} - a_2))^2}{b_1^2} + \frac{(w_{2,1}(z_{k,1}^{(\alpha)} - a_1) + w_{2,2}(z_{k,2}^{(\alpha)} - a_2))^2}{b_2^2} = 1 \qquad (3.6)$$

Consider the line connecting $\mathbf{X}_i^{(\alpha)}$ and $\mathbf{Y}_j^{(\beta)}$, which is a nearest data vector pair, the linear equation of this line is defined by

$$(z_{k,2}^{(\alpha)} - x_{i,2}^{(\alpha)}) = \frac{(y_{j,2}^{(\beta)} - x_{i,2}^{(\alpha)})}{(y_{j,1}^{(\beta)} - x_{i,1}^{(\alpha)})}(z_{k,1}^{(\alpha)} - x_{i,1}^{(\alpha)}) \qquad (3.7)$$

From the Eqs. 3.6 and 3.7, $z_{k,1}^{(\alpha)}$ and $z_{k,2}^{(\alpha)}$ is are determined in terms of a_1 and a_2, and used for finding the distances din_i and $dout_j$, respectively. Thus,

$$din_i = \sqrt{(z_{k,1}^{(\alpha)} - y_{j,1}^{(\beta)})^2 + (z_{k,2}^{(\alpha)} - y_{j,2}^{(\beta)})^2} \qquad (3.8)$$

$$dout_j = \sqrt{(x_{i,1}^{(\alpha)} - z_{k,1}^{(\alpha)})^2 + (x_{i,2}^{(\alpha)} - z_{k,2}^{(\alpha)})^2} \qquad (3.9)$$

The new values of $a_1^{(\alpha)}$ and $a_2^{(\alpha)}$, denoted by $a_1^{(\alpha),new}$ and $a_2^{(\alpha),new}$ of each neuron can

be found by applying a gradient descent learning rule to E_{fault} as follows.

$$\frac{\partial E_{fault}}{\partial a_1^{(\alpha)}} = \Delta a_1^{(\alpha)}, \tag{3.10}$$

$$\frac{\partial E_{fault}}{\partial a_2^{(\alpha)}} = \Delta a_2^{(\alpha)} \tag{3.11}$$

$$a_1^{(\alpha),new} = a_1^{(\alpha),old} + \Delta a_1^{(\alpha)} \tag{3.12}$$

$$a_2^{(\alpha),new} = a_2^{(\alpha),old} + \Delta a_2^{(\alpha)} \tag{3.13}$$

The detail of fault immunization is given in the following algorithm. There are two main steps in the algorithm. The first step is to find all nearest pairs $\mathbf{X}_i^{(\alpha)}$ and $\mathbf{Y}_j^{(\beta)}$ of every GRBF neuron. Then, the second step is to adjust the center of each GERBF neuron to minimize E_{fault}. Let ϵ and T be predefined constants.

3.5 Algorithm of Elliptic Radial Basis Function

The learning algorithm is similar to that of RAN. There are two consecutive procedures. First, all data vectors of the same class are covered by some GERBF neurons. Then, the removal of some redundant neurons is proceeded. The learning process begins with an empty network. A randomly selected data vector is used to introduce the first GERBF neuron to the network. Then, the other subsequent data vectors of the same class are randomly chosen for adjusting the parameters to stretch the shape of the corresponding GERBF to cover these new data vectors. It may be possible that during the stretching process, the shape of the GERBF may unintentionally cover some data vectors from the other classes. In this case, the shape of the GERBF must be shrunk and a new GERBF neuron is introduced to the network. The detail of the first procedure is as follows. Suppose that the data vectors in class A will be covered first.

Covering Algorithm

1. Let $i = 1$.
2. Introduce a GERBF neuron, g_i. and initialize its parameters.
3. Randomly select the first data vector and cover this vector by g_i.
4. **While** all data vectors in class A are not covered **do**
5. Randomly select a new data vector $\mathbf{X}_j$
6. Compute the output of network ($\phi_k(\mathbf{X}_j)$) from Eqs. 3.3
7. **If** $\mathbf{X}_j$ is not covered by G_i **then**
8. Stretch the size of G_i to cover $\mathbf{X}_j$
9. Let $t = 1$
10. **While** some data vectors from other classes are covered and $t \leq T$ **do**
11. shrink the size of G_i and $t = t + 1$
12. **endwhile**
13. **If** $t > T$ **then**
14. $i = i + 1$ and introduce a new GERBF neuron
15. **endif**
16. **Endif**
17. **Endwhile**

Pruning each hidden unit

1. **For** each hidden unit
2. **If** all patterns in this hidden unit are also in another hidden unit **then**
3. Pruned this hidden unit
4. **Endif**
5. **Endfor**

3.6 Algorithm of ERBF Fault Immunization Model

1. **For** each GERBF neuron q **do**
2. Set $t = 1$
3. Find all nearest pairs by
4. Randomly select the data vectors covered by neuron q
5. Find the nearest data vectors from the other classes
6. Adjust the center vector by
7. **While** $E_{fault} > \epsilon$ **and** $t \leq T$ **do**
8. Save the current value of the center vector
9. Adjust the value of center vector
10. $t = t + 1$
11. **Endwhile**
12. **If** $t > T$ **then** restore the value of the center vector

3.7 An Example of GERBF Neural Network Algorithm

Table 3.1 shows an example of GERBF neural network algorithm. There are two classes of input data vectors which will be classified. In Table 3.1 (a), the crosses represent the input data vectors in class A and the dot represent the input data vectors in class B. Suppose that the data vectors in class A will be covered first. The learning process begins with an empty network. The first input data vector is a randomly selected and fed to the network. This vector is covered by the first GERBF neuron which is represented by

an ellipse as illustrated in Table 3.1 (b). Table 3.1 (c) shows the new cross representing another subsequent data vector which will be covered. Line 1 indicates the same GERBF neuron as shown in Table 3.1(b). Line 2 points to the GERBF neuron being adjusted to cover another subsequent data vector. Line 3 represents the GERBF neuron which has completely covered another subsequent data vector. The new cross representing another subsequent data vector is selected to learn as shown in Table 3.1 (d). The previous GERBF neuron is stretched to cover this new data vector but, unfortunately, the GERBF neuron not only covers this new data vector but also covers other data vector from the other classes. The adjusting process is preformed until no misclassification occurs. Table 3.1 (e) gives the classification result. Obviously, there are some redundant GERBF neurons which are represented by the dotted ellipses and must be pruned. The final result of this example is summarized in Table 3.1 (f). Only the relevant GERBF neurons are shown.

3.8 An Example of GERBF Fault Immunization Algorithm

The fault immunization degree of each GERBF neuron in Table Table 3.1 (f) is increased as follows.Table 3.2 (a) depicts a randomly selected input data vector in class A which is represented by the cross having a line connecting to the nearest dot representing the input data vector in class B. The next step is to find the shortest distance from the same dot as shown in Table 3.2 (a) to the cross which is the nearest input data vector in another class. The two connected input data vectors are called the nearest data vector pair as depicted in Table 3.2 (b), Table 3.2 (c) shows the same process is continually repeated at m times to find all the nearest data vector pairs in the GERBF neuron. Table 3.2 (d) illustrates the adjustment of the center of GERBF neuron in order to achieve the higher immunization degree. The new position of the GERBF neuron is represented by the dotted ellipse, and the new position of the center is represented by a triangle. The

same process is repeated for all the rest of GERBF neurons as shown in Table 3.2 (e). Finally, all GERBF neurons in the network achieve the higher immunization degree, and the result of the GERBF fault immunization algorithm is indicated in Table 3.2 (f).

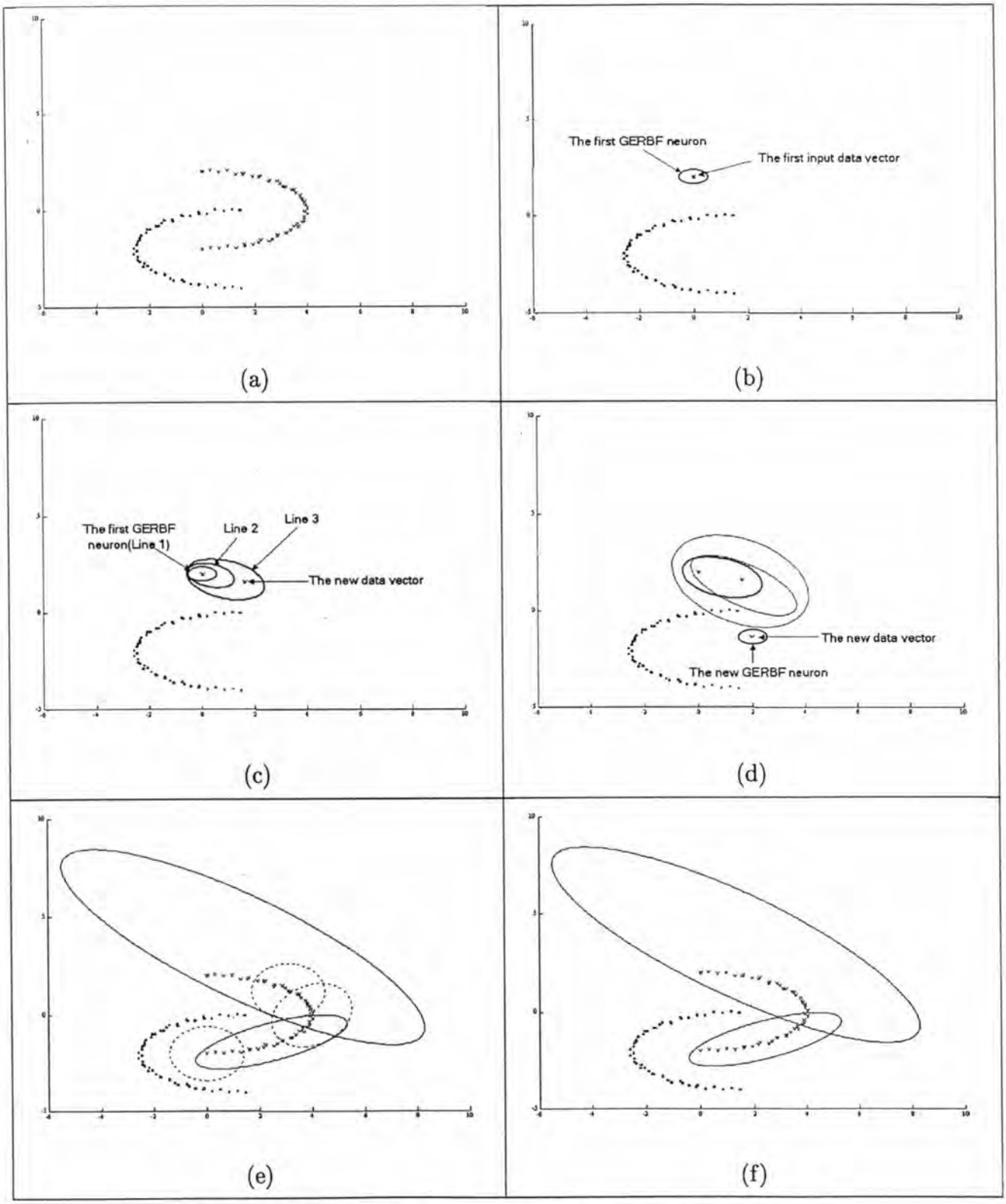


Table 3.1: An example of GERBF neural network algorithm

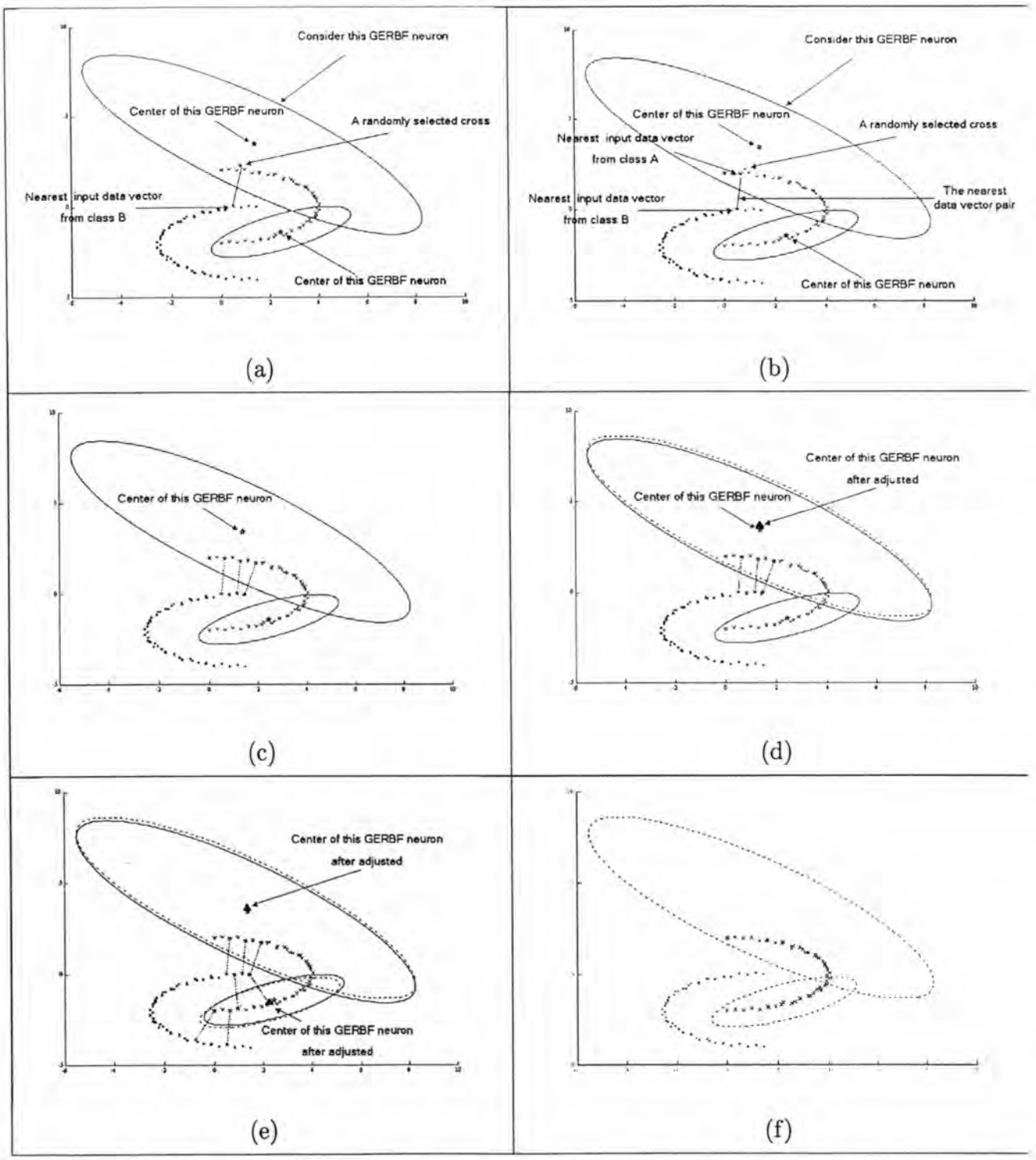


Table 3.2: An example of GERBF fault immunization algorithm.