# Decision Boundary Formation of Neural Networks[1]

C. LEE, E. JUNG, O. KWON, M. PARK, AND D. HONG
Department of Electrical and Electronic Engineering, Yonsei University
134 Shinchon-Dong, Seodaemum-Ku, Seoul 120-749, Korea

*Abstract*: In this paper, we provide a thorough analysis of decision boundaries of neural networks when they are used as a classifier. First, we divide the classifying mechanism of the neural network into two parts: dimension expansion by hidden neurons and linear decision boundary formation by output neurons. In this paradigm, the input data is first warped into a higher dimensional space by the hidden neurons and the output neurons draw linear decision boundaries in the expanded space (hidden neuron space). We also found that the decision boundaries in the hidden neuron space are not completely independent. This dependency of decision boundaries is extended to multiclass problems, providing a valuable insight into formation of decision boundaries in the hidden neuron space. This analysis provides a new understanding of how neural networks construct complex decision boundaries and explains how different sets of weights may provide similar results.

*Key-Words*: neural networks, analysis of decision boundary, dimension expansion, linear boundary, dependent decision boundary.

## 1 Introduction

Neural networks have been successfully used in various pattern recognition problems including character recognition [1], remote sensing [2], and communication [3]. The increasing popularity of neural networks is partly due to their ability to learn and therefore generalize. And neural networks make no prior assumptions about the statistics of input data and can construct complex decision boundaries. Although it is known that neural networks can define arbitrary decision boundaries without assuming any underlying distribution [4], the decision boundary of neural networks are not well understood.

There have been many papers that analyzed how neural networks are working [5-7]. Gibson and Cowan investigated decision regions of multi-layer perceptrons and derived some geometric properties of the decision regions [8]. Makhoul and et. al. showed that neural networks with a single hidden layer are capable of forming disconnected decision regions [9]. Blackmore and et. al investigated decision region approximation of neural networks and compared neural networks with polynomials [10]. Nitta analyzed the decision boundaries of complex valued neural networks [11]. On the other hand, some researchers investigated a learning algorithm based on decision based formulation for pattern classification problems [12]. And Pal and et. al. proposed a method to find decision boundaries for pattern classification using genetic algorithms and made extensive performance comparisons with neural networks and other classifiers, providing insights into the decision boundaries of complex pattern classification problems [13].

In this paper, we systematically analyze the decision boundary of feedforward neural networks and provide a helpful insight and new interpretation into the working mechanism of neural networks. In particular, when neural networks are used as a classifier, we note that the working mechanism of the neural network can be divided into two parts: dimension expansion by hidden neurons and linear decision boundary formation by output neurons. In this context, we define the role of hidden neurons as mapping the original data into a different dimensional space.

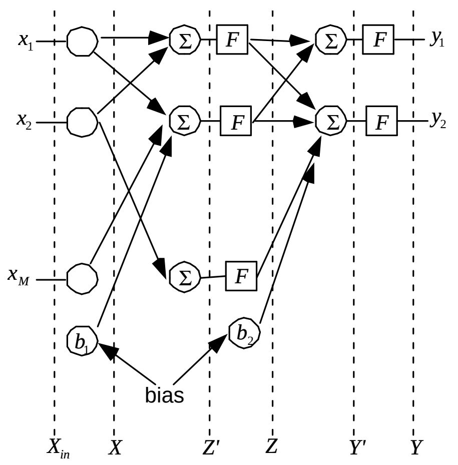

Fig. 1. Example of 3-layer feedforward neural networks (2 pattern classes).

[1] The Korea Science and Engineering Foundation partly supported the publication of this paper through BERC at Yonsei University.

## 2 Feedforward neural networks and terminologies

A typical neural network has the input layer, a number of hidden layers, and the output layer. It may also include bias terms. Fig. 1 shows an example of 3-layer feedforward neural networks (2 pattern classes). The decision rule is to choose the class corresponding to the output neuron with the largest output [14]. First, we assume that the activation function is the sigmoid function:

$$F(x) = \frac{1}{1+e^{-x}} \qquad (1)$$

In the neural network of Fig. 1, we will assume that *input vector* $X_{in}$ is an $M$ x 1 column vector, $X$ an ($M$+1) x 1 column vector, $Z'$ an $N$ x 1 column vector, $Z$ an ($N$+1) x 1 column vector. Vectors in the output layer, $Y'$ and $Y$, are 2 x 1 column vectors. We will call $Z$ *the output vector of hidden neurons*. Each component of the output vector of hidden neurons, $Z$, is computed as follows:

$$z_i = \frac{1}{1+e^{-f_i \bullet X}} \quad i = 1,...,N \text{ and } z_{N+1} = 1.$$

Consequently, points that satisfy $f_i^T X = c$ will end up with the same value. And $f_i^T X = c$ represents a point, a line, a plane or a hyper-plane in the input space depending on the dimension of the input vector. In this paper, we will call $f_i^T X = c$ *equivalent-weight lines*, *equivalent-weight planes*, or *equivalent-weight hyper-planes* depending on the input dimension. Furthermore, $f_i^T X = 0$, which corresponds to $z_i = 0.5$, will be called *middle-weight lines*, *middle-weight planes*, or *middle-weight hyper-planes* depending on the dimension. In Fig. 1, there are two bias terms $b_1, b_2$. Without loss of generality, we can assume $b_1 = b_2 = 1$. It can be easily seen that the relationships between these vectors are as follows:

$$X_{in} = [x_1, x_2, ..., x_M]^T$$

$$X = [x_1, x_2, ..., x_M, b_1]^T$$

$$Z' = \Phi^T X = [f_1, f_2, ..., f_N]^T X = [f_1^T X, f_2^T X, ..., f_N^T X]^T \qquad (2)$$

where $f_i$ is an ($M$+1) x 1 column vector. We will call

$$\Phi = [f_1, f_2, ..., f_N]$$

*the weight matrix between the inputs and the hidden neurons* and $f_i$ *the weight vector between hidden neuron i and the inputs*. And the remaining vectors are obtained as follows:

$$Z = [F(f_1^T X), F(f_2^T X), ..., F(f_N^T X), b_2]^T$$

$$Y' = \Psi^T Z = [y_1, y_2]^T Z = [y_1^T Z, y_2^T Z]^T$$

$$Y = [y_1, y_2]^T = [F(y_1^T Z), F(y_2^T Z)]^T$$

where $y_i$ is an ($N$+1) x 1 column vector. We will call $\Psi = [y_1, y_2]$ *the weight matrix between the output and hidden neurons* and $y_i$ *the weight vector between output neuron i and the hidden neurons*. Furthermore, we will call the space defined by the input vector, $X_{in}$, *the input space* and the space defined by the outputs of the hidden neurons with the last component removed, $[z_1, z_2, ..., z_N]^T = [F(f_1^T X), F(f_2^T X), ..., F(f_N^T X)]^T$, *the hidden neuron space*.

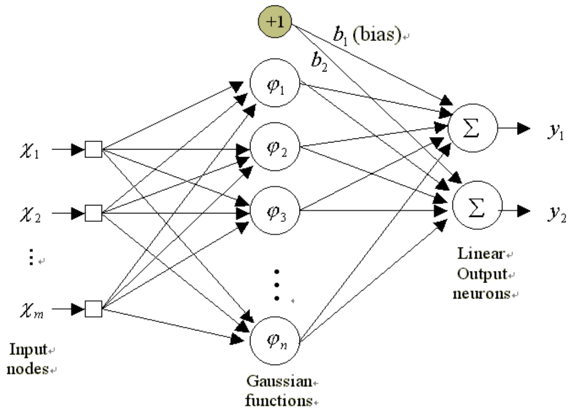

Fig. 2. Example of a radial basis function neural network (2 pattern classes).

Fig. 2 shows an example of a radial basis function neural network with the Gaussian function in the hidden layer. Since the Gaussian function is used as the radial basis function in the hidden neurons, the outputs of the hidden neurons are computed as follows:

$$j_i(X) = \exp(-\|X - \mathrm{M}_i\|^2)$$

where $X = [x_1, x_2, ..., x_m]^\mathrm{T}$, $\mathrm{M}_i = [m_{i1}, m_{i2}, ..., m_{im}]^\mathrm{T}$, $i = 1, ..., n$. $\mathrm{M}_i$ is called the center and is one of the parameters to be updated during the learning process. Similarly, the points that satisfy $\|X - \mathrm{M}_i\| = c$ represent points, a circle, a sphere or a hyper-sphere in the input space depending on the dimension of the input space. In this paper, the points that satisfy $\|X - \mathrm{M}_i\| = 0.833$, which corresponds to $j_i(X) = 0.5$, will be called a half circle, a half sphere or a half hyper-sphere depending on the input dimension. The *i*-th output of the output layer is simply computed as follows:

$$y_i = \sum_{j=1}^{n} w_{ij} j_j = W_i^\mathrm{T} \Phi$$

where $W_i = [w_{i1}, w_{i2}, ..., w_{in}]^\mathrm{T}$, $\Phi = [j_1, j_2, ..., j_n]^\mathrm{T}$

here $w_{ij}$ is the weight between *j*-th hidden neuron and *i*-th output neuron. In order to avoid confusion, we will denote the neural network whose activation function is the sigmoid function as *SIGNN* and the radial basis function neural network as *RBFNN.*

## 3 Dimension expansion of hidden neurons

First, we view the outputs of the hidden neurons as a non-linear-mapping of inputs since a typical activation function is a non-linear function such as the sigmoid function or the Gaussian function. And we observe that adding a hidden neuron is equivalent to expanding the dimension of the hidden neuron space. Thus, if the number of hidden neurons is larger than the number of inputs, the input data will be warped into a higher dimensional space. However, the intrinsic dimension of the data distribution in the hidden neuron space can not exceed the dimension of the original input space. For example, if the dimension of the input vector is 2 and the number of hidden neurons is 3, the data will be distributed on a curved plane in the 3-dimensional space as shown in Fig. 3. In other words, if the number of hidden neurons is larger than the dimension of the input space, the input data will be warped into a higher dimensional space while maintaining the same intrinsic dimension. On the other hand, if $f_i^T X = c_i$ and $f_j^T X = c_j$ are parallel, then the intrinsic dimension in the hidden neuron space can be smaller than the dimension of the original input space. Fig. 4 illustrates an example of the case where the plane in the input space is mapped onto a curved line in the hidden neuron space. Fig. 5 shows an example of dimension expansion of RBFNN with 3 hidden neurons. The corresponding three half circles and the decision boundaries are also displayed. In the next section, we will investigate decision boundaries in the hidden neuron space, which will be always linear boundaries. From this point of view, it can be seen that, when neural networks are used as a classifier, the input data is first mapped non-linearly into a higher dimensional space and then divided by linear decision boundaries. Finally, the linear decision boundaries in the hidden neuron space will be warped into complex decision boundaries in the original input space.

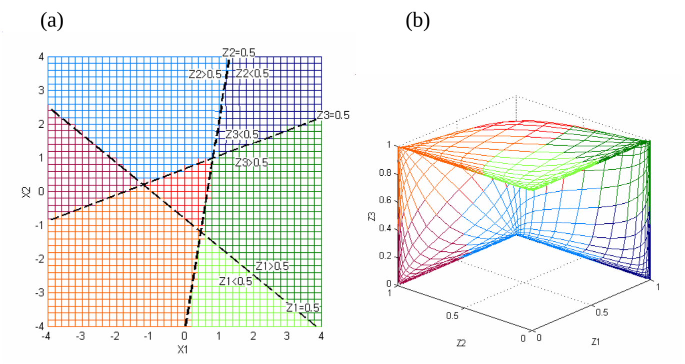

Fig. 3. An example of dimension expansion of SIGNN (a) three middle-weight lines in the input space (b) the corresponding distribution in the hidden neuron space.

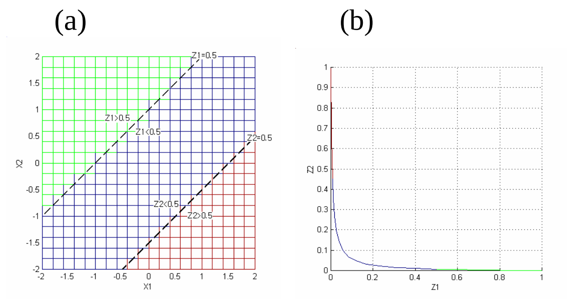

Fig. 4. A case where the plane of the input space is mapped onto a curved line in the hidden neuron space (SIGNN).

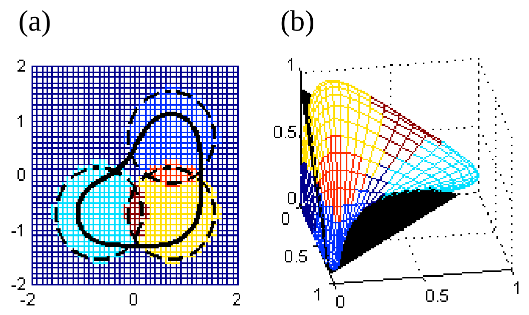

Fig. 5. An example of dimension expansion of RBFNN (a) three half circles, which correspond to 3 hidden neurons, in the 2 dimensional input space (b) the corresponding curved plane in the 3 dimensional hidden neuron space.

## 4 Decision boundaries in the hidden neuron space

In this section, we analyze the decision boundaries of neural networks whose activation function is the sigmoid function. However, it can be easily seen that the same analysis can be applied to RBFNN.

### 4.1 Two pattern classes

First, we will consider decision boundaries of neural networks for a 2-pattern class problem as shown in Fig. 1. Since the decision boundaries between two output neurons is a building block for multiclass problems, a thorough analysis of the decision boundaries of a 2-pattern class problem will provide a valuable insight into how decision boundaries of neural networks are defined. After training is finished, the decision rule is to choose the class corresponding to the output neuron with the largest output. And the decision boundary defined by the neural network in Fig. 1 is given by

$$y_1 = y_2$$

in the space defined by $y_1$ and $y_2$ ($y_1$-$y_2$ plane). And the equivalent decision boundary is given by

$$F(y_1^T Z) = F(y_2^T Z) \qquad (3)$$

in the hidden neuron space that is defined by $z_1, z_2, \ldots, z_N$. Since the sigmoid function of (1) is monotonically increasing, the same decision boundary given by (3) will be obtained in the hidden neuron space with the following equation:

$$y_1^T Z = y_2^T Z$$
$$\Rightarrow (y_1 - y_2)^T Z = C^T Z = 0$$
$$\Rightarrow c_1 z_1 + c_2 z_2 + \ldots + c_N z_N + c_{N+1} b_2 = 0$$
$$\Rightarrow c_1 z_1 + c_2 z_2 + \ldots + c_N z_N + c_{N+1} = 0$$

where $b_2 = 1$ and $C = y_1 - y_2$ is an ($N$+1) x 1 column vector. In other words, the decision boundary between two classes in the hidden neuron space will be always a linear decision boundary, though decision boundaries in the input space are non-linear complex decision boundaries. From this analysis, it can be easily seen that neural networks with two output neurons can be always reduced to neural networks with one output neuron where the weight vector between the hidden neurons and the output neuron is given by $C = y_1 - y_2$. In this way, the complexity of neural networks can be reduced substantially. And the decision rule is as follows:

If $y' = C^T Z = (y_1 - y_2)^T Z > 0,\ decide\, w_1$.

Else, $decide\, w_2$.

## 4.2 Three pattern classes

If there are 3 pattern classes, typically there will be three output neurons and the decision rule is to choose the class corresponding to the output neuron with the largest output. The output vector of output neurons, $Y$, will be given by

$$Y = [y_1, y_2, y_3]^T = [F(y_1^T Z), F(y_2^T Z), F(y_3^T Z)]^T .$$

In the hidden neuron space, the three decision boundaries between each pair of classes will be given by

$$y_1^T Z = y_2^T Z \Rightarrow (y_1 - y_2)^T Z = 0 \qquad (4)$$
$$y_2^T Z = y_3^T Z \Rightarrow (y_2 - y_3)^T Z = 0 \qquad (5)$$
$$y_3^T Z = y_1^T Z \Rightarrow (y_3 - y_1)^T Z = 0 . \qquad (6)$$

Each equation represents a linear decision boundary in the hidden neuron space. Thus, any two of the three equations will have intersection except the trivial case that they are parallel. Let $Z_1$ be a point on the intersection of (4) and (5). In other words,

$$(y_1 - y_2)^T Z_1 = 0$$
$$(y_2 - y_3)^T Z_1 = 0 .$$

Then we can easily show that

$$(y_3 - y_1)^T Z_1 = -(y_1 - y_2)^T Z_1 - (y_2 - y_3)^T Z_1 = 0 .$$

In other words, $Z_1$ will also satisfy (6). Similarly, we can show that any point that is a solution to any two of the three equations will be a solution of the remaining equation. This indicates that the three linear decision boundaries will always meet at the same intersection. Fig. 6 illustrates how the decision boundaries are formed in the hidden neuron space for a 3-class problem. The first decision boundary divides the hidden neuron space into two regions (Fig. 6a). When the second decision boundary between $w_1$ and $w_3$ is added as shown in Fig. 6b, the upper-left region can be classified as $w_3$ ($y_3 > y_1 > y_2$). And the upper-right region is classified as $w_1$ and the lower-right as $w_2$. The lower-left region is not determined yet. The third decision boundary can not divide the upper-left, which is the region that is classified as $w_3$ after the second decision boundary was introduced, as shown in Fig. 6c since it produces a contradiction ($y_3 > y_1, y_1 > y_2, y_2 > y_3$) in the region that is denoted as NA (not allowed). Thus, the third decision boundary should divide the undetermined region as shown in Fig. 6d. However, it can not produce the classification result as shown in Fig. 6e. Let P be a point on the line of $y_3 = y_1 + C$ where C is an arbitrary positive number. As P approaches to the decision boundary between $w_1$ and $w_2$, the difference between $y_2$ and $y_1$ diminishes to zero (Fig. 6f). However, since $y_3 = y_1 + C$ where C can be arbitrarily large, P can not be classified as $w_2$ but would be classified as $w_3$ ($y_3 >> y_1 \approx y_2$). It is noted that the neural network for a 3-pattern class problem divides the hidden neuron space into 3 regions, though the 3 decision boundaries divides the hidden neuron space into 6 regions. As illustrated in the above, if two decision boundaries are given, the remaining decision boundary will be automatically determined since three equations (4-6) are not linearly independent. In other words,

$$(y_3 - y_1)^T Z = -(y_1 - y_2)^T Z - (y_2 - y_3)^T Z .$$

We showed that that neural networks with two output neurons can be always reduced to neural networks with one output neuron. Similarly, neural networks with three output neurons can be always reduced to neural networks with two output neurons where the weight vectors between the hidden neurons and the output neurons are given by $C_1 = y_1 - y_2$ and $C_2 = y_2 - y_3$.

## 4.3 Angles between linear decision Boundaries in the hidden neuron space

In the hidden neuron space, $(y_1 - y_2)^T Z = 0$ represents linear decision boundaries. It can be rewritten as

$$(y_1 - y_2)^T Z = C^T Z = 0$$
$$\Rightarrow c_1 z_1 + c_2 z_2 + \ldots + c_N z_N + c_{N+1} b_2 = 0$$
$$\Rightarrow c_1 z_1 + c_2 z_2 + \ldots + c_N z_N = -c_{N+1}$$

where $b_2 = 1$ and $C = y_1 - y_2$ is an (*N*+1) x 1 column vector. Let $y_i'$ be the vector whose components are the same as those of $y_i$ with the last component of $y_i$ removed. In other words,

$$y_i' = [y_{i,1}, y_{i,2}, ..., y_{i,N}]^T$$

where $y_{i,j}$ is the j-th component of $y_i$. Depending on the dimension, $(y_1' - y_2')^T Z = 0$ may represent a line, a plane or a hyperplane that are perpendicular to vector

$$y_1' - y_2' = [y_{1,1} - y_{2,1}, y_{1,2} - y_{2,2}, ..., y_{1,N} - y_{2,N}]^T .$$

On the other hand, it can be easily seen that

$$(y_3' - y_1') = -(y_1' - y_2') - (y_2' - y_3') . \qquad (7)$$

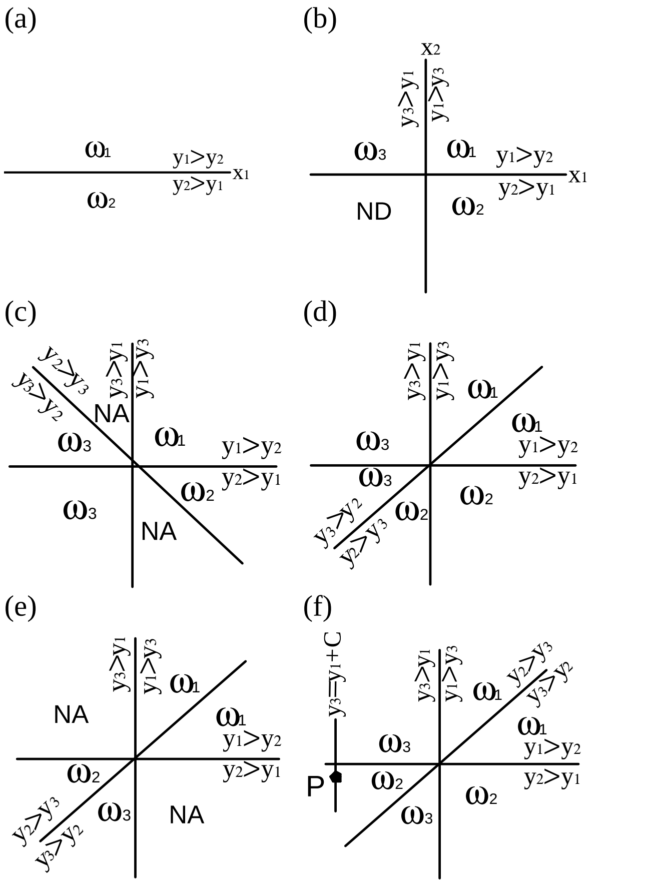

Fig. 6. Decision boundaries for 3 pattern class problems in the hidden neuron space.

The relationship among the three vectors of (7) is illustrated in Fig. 7. The linear decision boundary between class $w_i$ and class $w_j$, which is perpendicular to $y_i' - y_j'$, is also shown in Fig. 7. Since the angle between two lines (or planes) is the same as that between two vectors that are normal to the two lines (or planes), the angle between two decision boundaries in the hidden neuron space is given by

$$\cos q = \frac{(y_i - y_j)^T (y_j - y_k)}{|(y_i - y_j)||(y_j - y_k)|} \qquad (8)$$

where $q$ is the angle between " $w_i - w_j$ decision boundary" and " $w_j - w_k$ decision boundary." Since decision boundaries are not a vector, we may have to take the absolute value of (8). Fig. 7 illustrates the relationship of the angles between the decision boundaries in the hidden neuron space for a 3 pattern class problem. Therefore, it can be said that the angle between two decision boundaries in the hidden neuron space is determined by the directions and magnitudes of vectors $(y_i' - y_j')$, $(y_j' - y_k')$.

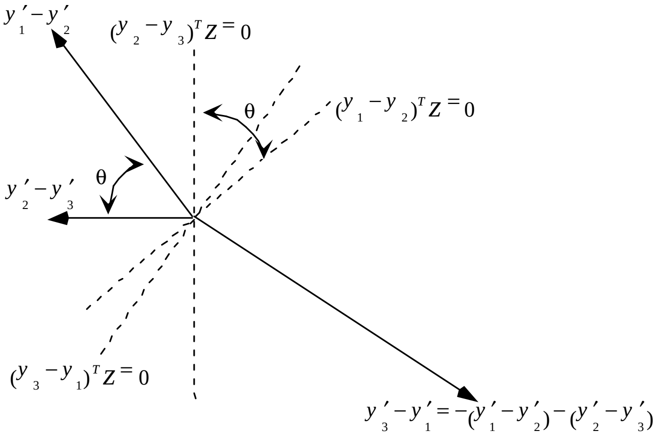

Fig. 7. Angles between decision boundaries for 3 classes in the hidden neuron space.

## 4.4 Decision boundaries in the multiclass problems

If there are *K* pattern classes (*K* output neurons), theoretically there will be $\binom{K}{2}$ decision boundaries in the hidden neuron space. However, only K-1 decision boundaries are independent and the remaining boundaries will be automatically determined. In other words, there are only *K*-1 degree of freedoms. For example, for a 4-pattern class problem, there are 6 decision boundaries in the hidden neuron space. However, we have only 3 degree of freedoms. Once the 3 decision boundaries are given, the remaining 3 boundaries will be automatically determined as shown in Fig. 8, where the solid lines represent the 3 independent decision boundaries and the dotted lines represent the 3 dependent decision boundaries. In other words, if the $w_i - w_j$ decision boundary and the $w_i - w_k$ decision boundary are decided, the $w_j - w_k$ decision boundary is automatically determined. It should pass through the intersection where the $w_i - w_j$ decision boundary and the $w_i - w_k$ decision boundary meet and is given by

$$(y_j - y_k)^T Z = -(y_i - y_j)^T Z + (y_i - y_k)^T Z = 0 .$$

As shown previously, the direction of the $w_j - w_k$ decision boundary is also determined once the $w_i - w_j$ decision boundary and the $w_i - w_k$ decision boundary are decided. For a 4-class problem, there will be 4 intersections (points, lines, planes or hyper-planes) where 3 decision boundaries meet except the trivial case that some of decision boundaries are parallel. At those intersections, the 3 decision regions corresponding to 3 classes will be determined. By combining those regions, one can construct the overall decision boundaries that are displayed with bold lines in Fig. 8. In Fig. 8, the 4 bold dots represent 4 intersections where 3 decision boundaries meet. Although there are $\binom{K}{2}$ decision boundaries for a K-class problem and the decision boundaries divide the hidden neuron space into numerous subregions, the final decision boundaries for classification divide the hidden neuron space into only K regions that are convex.

In general, the linear decision boundaries, which divide the hidden neuron space into K regions, will be warped into complex decision boundaries that may divide non-linearly the original input space into much more than K regions. For example, in the case of two pattern class problems, the decision boundary for SIGNN in the *X*-space is given by

$$c_1 z_1 + c_2 z_2 + \ldots + c_N z_N + c_{N+1} = 0$$

$$\Rightarrow \frac{c_1}{1+e^{-f_1 \bullet X}} + \frac{c_2}{1+e^{-f_2 \bullet X}} + \ldots + \frac{c_N}{1+e^{-f_N \bullet X}} + c_{N+1} = 0$$

$$\Rightarrow \left( \sum_{i=1}^{N} \frac{c_i}{1+e^{-f_i \bullet X}} \right) + c_{N+1} = 0$$

where $C = y_1 - y_2$. Typically, decision boundaries in the input space can be straight lines, curves, circles, planes, curved surfaces, curved closed surfaces, and etc.

## 5 Examples of decision boundaries of neural networks

Without loss of generality, we may assume that input vectors to neural networks are bounded. One can always make input data bounded by given upper and lower bounds by scaling and translation. And we assume that there are two pattern classes and limit the dimension of the input space to 2 for an easy illustration.

### 5.1 Linear decision boundaries in the input space

The simplest linear boundary in the input space would be obtained with one hidden neuron. Assuming SIGNN, the decision boundary will be an equivalent-weight line in this case. More interesting linear boundaries can be obtained if we add more hidden neurons. For instance, Fig. 9 illustrates how a linear decision boundary of SIGNN, which solves an XOR problem, can be obtained with two hidden neurons. In Fig. 9a, the middle-weight lines, *i.e.*, $f_i^T X = 0$, are parallel. Fig. 10 shows another example of linear decision boundaries. In general, if the two middle-weight lines meet obliquely, as in the case of Fig. 10a, the decision boundary in the input space is not linear. However, if the decision boundary in the hidden neuron space is perpendicular to one of the coordinates in the hidden neuron space, we will obtain a linear decision boundary in the input space as shown in Fig. 10**.** On the other hand, Fig. 11 shows an example of linear decision boundary when there are 3 hidden neurons (3 middle-weight lines in the input space). Fig. 12 shows an example of linear decision boundaries of RBFNN with two hidden neurons that are closely located. As can be seen from Figs. 9-12, we have more freedom in drawing a more flexible decision boundary with more hidden neurons.

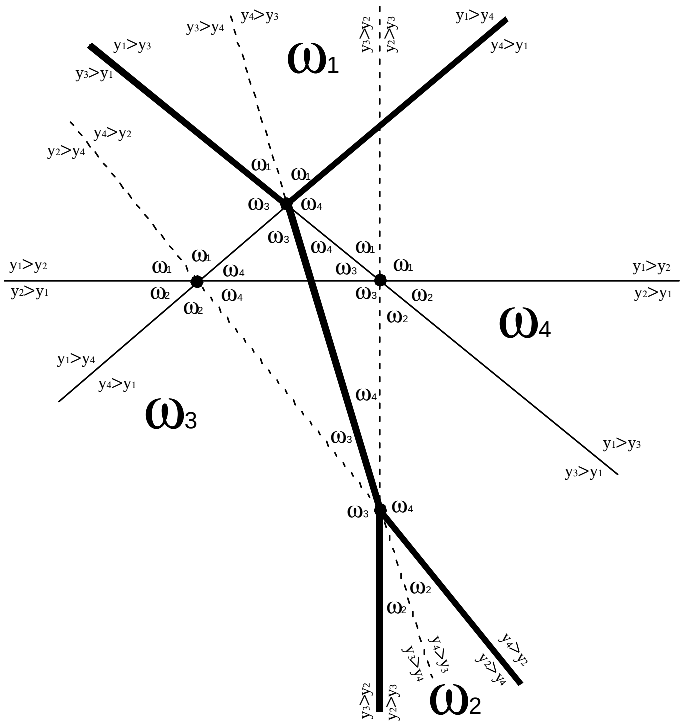

Fig. 8. Six decision boundaries of a 4 pattern class problem in the hidden neuron space.

### 5.2 Convex decision boundaries in the input space

Typically, if middle-weight lines meet obliquely in SIGNN as shown in Figs. 13-14, the decision boundaries in the input space will be convex except some special cases, though the decision boundaries in the hidden neuron space are linear. Usually, the convex decision boundaries in Fig. 13a and Fig. 14a divide the input space into two regions corresponding to two classes. Due to the nature of the sigmoid function, $z_i$ is bounded by (0,1) as shown in Figs 13b and 14b. In Fig. 14b, there are two limit points through which the linear decision boundary passes in the hidden neuron space: (0.25,1) and (1,0.25). Thus, it can be seen that the decision boundary in the input space asymptotically converges to two equivalent-weight lines: $f_1^T X = -1.099$ and $f_2^T X = -1.099$. It is noted that it is impossible to divide the input space into more than 2 regions with two hidden neurons except the special case of Fig. 9 where two middle-weight lines are parallel. However, if two middle-weight lines do not intersect in a given input range, it is still possible to divide the given input space into 3 regions as shown in Fig. 15.

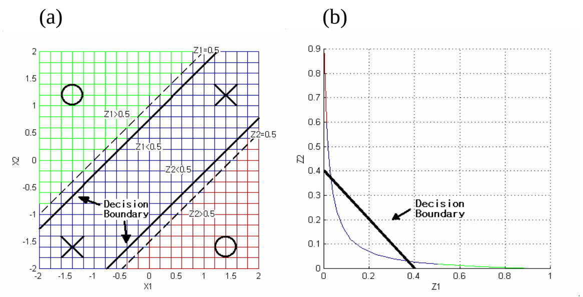

Fig. 9. An example of linear decision boundary of SIGNN (a) middle-weight lines and decision boundaries (bold line) in the input space (b) decision boundary in the hidden neuron space.

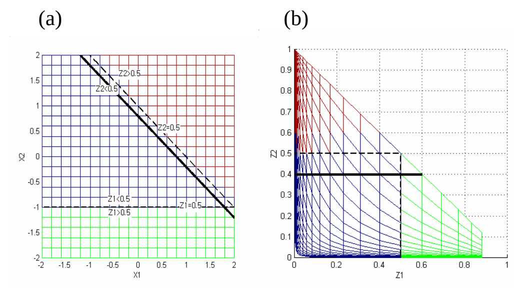

Fig. 10. Another example of linear decision boundary of SIGNN (a) middle-weight lines and decision boundary (bold line) in the input space (b) decision boundary in the hidden neuron space.

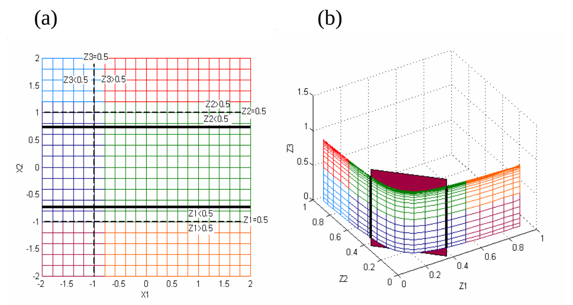

Fig. 11. An example of linear decision boundary of SIGNN when there are 3 hidden neurons middle-weight lines and decision boundaries (bold line) in the input space (b) decision boundary in the hidden neuron space.

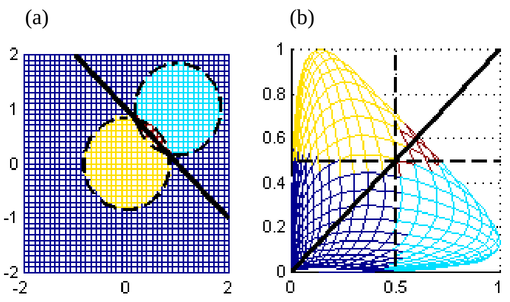

Fig. 12. An example of linear decision boundaries of RBFNN with two hidden neurons that are closely located (a) the input space (b) the hidden neuron space.

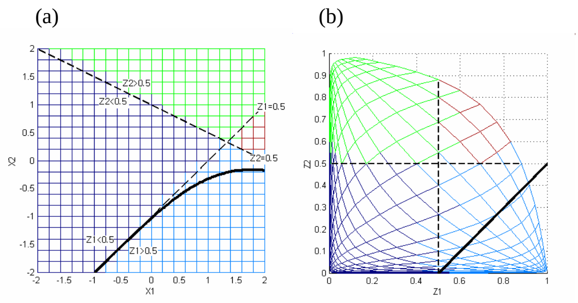

Fig. 13. An example of a convex decision boundary of SIGNN (bold line) (a) the input space (b) the hidden neuron space.

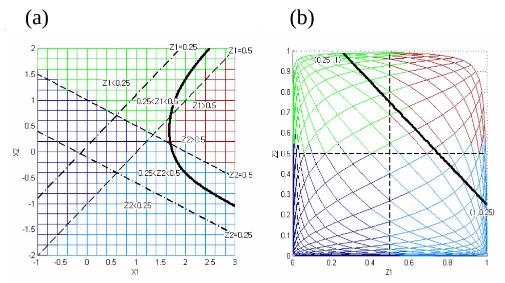

Fig. 14. Another example of convex decision boundaries of SIGNN (bold line) (a) the input space (b) the hidden neuron space.

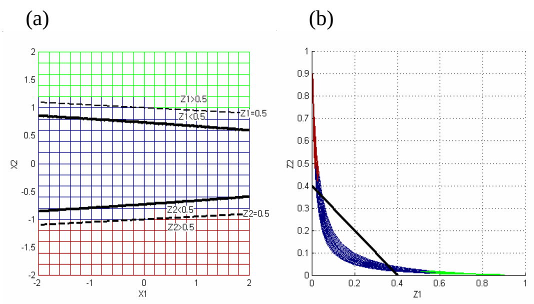

Fig. 15. Dividing the input space into 3 regions with two non-parallel middle-weight line (SIGNN)
(a) the input space (b) the hidden neuron space.

### 5.3 Disconnected and closed decision boundaries in the input space

As we add more hidden neurons, more complex decision boundaries can be obtained. As explained previously, the decision boundary in the hidden neuron space will be always linear. However, with more hidden neurons, we have more freedom in drawing more complex decision boundaries in the input space with linear decision boundaries in the hidden neuron space as shown in Figs. 16-18 (SIGNN). In general, when there are K output neurons, the neural network divides the hidden neuron space into K regions. However, the hidden neurons and the sigmoid function, which maps the entire input space into bounded region in the hidden neuron space, make it possible to divide the input space into more than K regions. For instance, the neural network with 3 hidden neurons can divide the input space into 4 regions (Fig. 16), 3 regions (Fig. 17) and 2 regions (Fig. 18). In particular, the decision boundary in Fig. 18 is a closed boundary. Fig. 19 shows an example of circular decision boundaries of RBFNN with two hidden neurons and Fig. 20 shows how two separate decision boundaries can be obtained in the input space. If the half circles in the input space cross, the data distribution in the hidden neuron space is convex (Fig. 19). If the half circles in the input space do not cross, the data distribution in the hidden neuron space become concave (Fig. 20). Unlike SIGNN, the typical decision boundary of RBFNN is a closed boundary except the special case of Fig. 12.

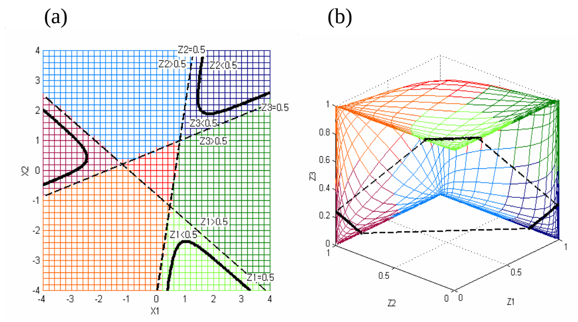

Fig. 16. Dividing the input space into 4 regions with 3 hidden neurons (SIGNN) (a) the input space (b) the hidden neuron space.

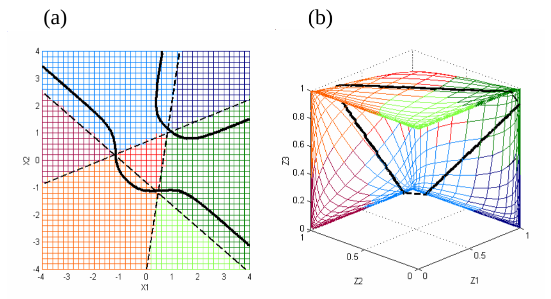

Fig. 17. More complex decision boundaries in the input space with 3 hidden neurons (SIGNN) (a) the input space (b) the hidden neuron space.

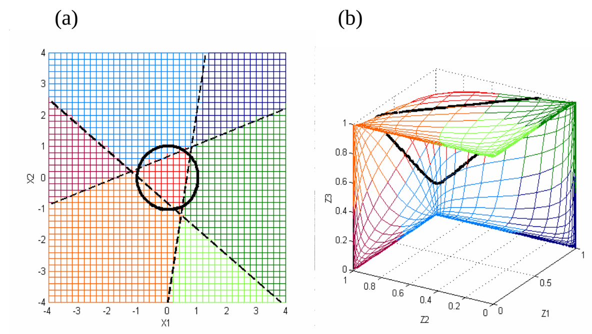

Fig. 18. Closed decision boundary in the input space with 3 hidden neurons (SIGNN) (a) the input space (b) the hidden neuron space.

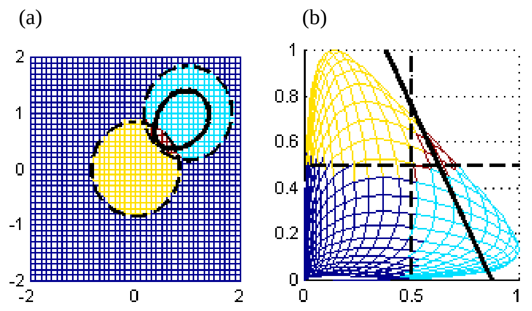

Fig. 19. An example of circular decision boundaries of RBFNN with two hidden neurons (a) the input space (b) the hidden neuron space.

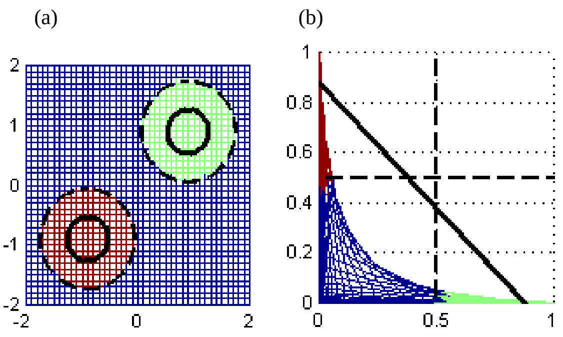

Fig. 20. An example of two circular boundaries with two hidden neurons (RBFNN) (a) the input space (b)the hidden neuron space.

### 5.4 Identical decision boundaries with different weights

Fig. 21 illustrates how neural networks with different weights can define the same decision boundary in the input space. It can be seen that the same decision boundary in the input space will be obtained even though we move the decision plane in the hidden neuron space (Fig. 21b). Although the neural network in Fig. 21a is a trivial one, the same phenomenon can occur for general neural networks as shown in Fig. 22. It has been reported that different sets of weights can provide almost identical performance for a given problem [15]. And these characteristics of decision boundaries in the hidden neuron space may provide some theoretical background how different sets of weights can provide almost identical performance for a given problem.

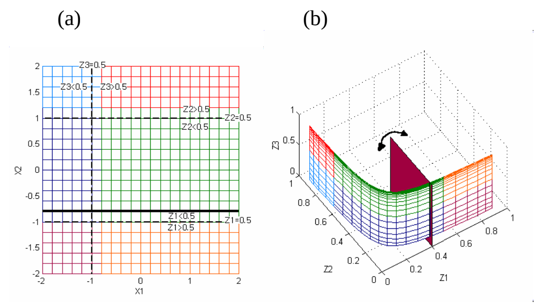

Fig. 21. Obtaining the same decision boundaries with different weights (SIGNN)
(a) the input space (b) the hidden neuron space.

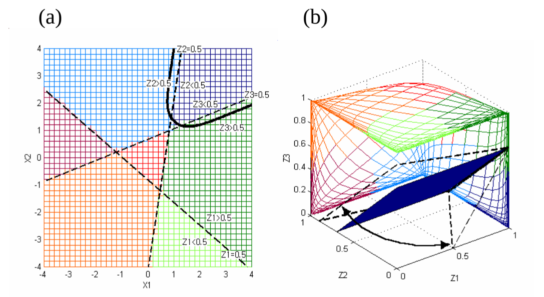

Fig. 22. Another example of obtaining the same decision boundaries with different weights (SIGNN) (a) the input space (b) the hidden neuron space.

## 6 Conclusions

In this paper, we investigated the decision boundaries of neural networks whose activation functions are the sigmoid function and the radial basis function neural network. We divided the classification mechanism of neural networks into two parts: expanding the dimension by hidden neurons and drawing linear boundaries by output neurons. In particular, we analyzed the decision boundaries in the hidden neuron space and found some interesting properties. First, the decision boundaries in the hidden neuron space are always linear boundaries and that the decision boundaries are not completely independent. Finally, we showed how the linear boundaries in the hidden neuron space can define complex decision boundaries in the input space with some interesting properties. The analysis of decision boundaries provides a way to reduce the complexity of neural networks and is helpful in weight initialization.

*References:*

[1] F. Fukushima and N. Wake, "Handwritten Alphanumeric Character Recognition by the Neocognitron," *IEEE Trans. on Neural Networks*, Vol. 2, No. 3, pp. 355-365, 1991.

[2] Jon Alti Benediktsson, Johannes R. Sveinsson, Orkan K. Ersoy, and Philip H. Swain, "Parallel Consensual Neural Networks," *IEEE Trans. Neural Networks*, Vol. 8, No. 1, pp. 54-64, 1997.

[3] K. Lee, S. Choi, S. Ong, C. You, and D. Hong, "Equalization techniques using neural networks for digital versatile disk-read-only memory," *Optical Engineering*, Vol. 38, No. 2, pp. 256-262, 1999.

[4] Chulhee Lee and David. A. Landgrebe, "Decision boundary feature extraction for neural networks," *IEEE Trans. Neural Networks*, Vol. 8, No. 1, pp. 75-83, 1997.

[5] G. J. Gibson, "A combinatorial approach to understanding perceptron capabilities," *IEEE Trans. Neural Networks*, Vol. 4, No. 6, pp. 989-992, 1993.

[6] B. Scholkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Muller, G. Ratsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Networks*, Vol. 10, No. 5, pp. 1000-1017, 1999.

[7] I. Sethi, "Entropy nets: from decision tree to neural networks," *Proceedings of the IEEE*, Vol. 78, No. 10, pp. 1605, 1990.

[8] G. J. Gibson and C. F. N. Cowan, "On the decision regions of multilayer perceptrons," *Proceedings of the IEEE*, Vol. 78, No. 10, pp. 1590-1594, 1990.

[9] J. Makhoul, A. El-Jaroudi, and R. Schwartz, "Partitioning capabilities of two-layer neural networks," *IEEE Trans. Signal Processing*, Vol. 39, No. 6, pp. 1435-1440, 1991.

[10] K. L. Blackmore, R. C. Williamson, and I. Y. Mareels, "Decision region approximation by polynomials or neural networks," *IEEE Trans. Information Theory*, Vol. 43, No. 3, pp. 903-907, 1997.

[11] T. Nitta. "An analysis on decision boundaries in the complex back-propagation network," in *Proc. IEEE Conf. IEEE World Congress on Computational Intelligence*, 2, pp. 934 - 939, 1994.

[12] S. Y. Kung and J. S. Taur, "Decision-based neural networks with signal/image classification applications," *IEEE Tran. Neural Networks*, Vol. 6, No. 1, pp. 170-181, 1995.

[13] S. K. Pal, S. Bandyopadhyay, and C. A. Murthy, "Genetic algorithms for generation of class boundaries," *IEEE Trans. Systems, Man and Cybernetics, Part B*, Vol. 28, No. 6, pp. 816 -828, 1998.

[14] R. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, Vol. 4, No. 2, pp. 4-22, April 1987.

[15] J. Go and C. Lee. "Analyzing weight distribution of neural networks," in *Proc. IEEE IJCNN*, pp. 1999.