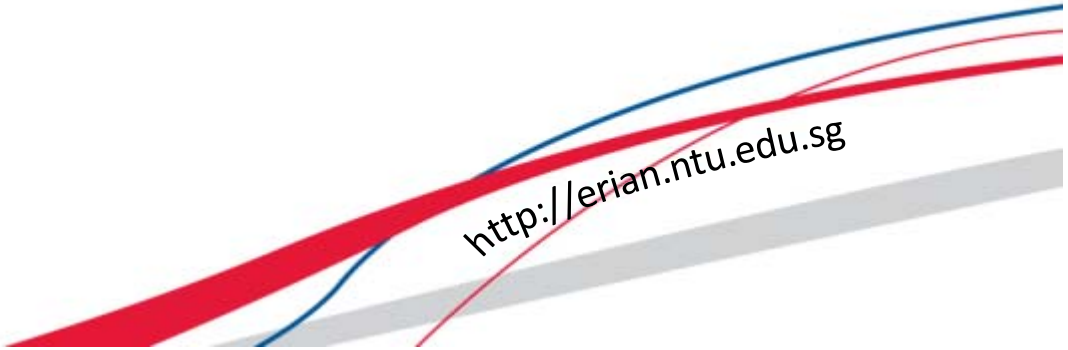
NANYANG TECHNOLOGICAL UNIVERSITY

4. Neuro-Fuzzy Classification



Classification and *Recognition*

Assume that a *set* of input patterns is divided into a number of classes, or categories. In response to an input pattern from the *set*, the network or classifier is supposed to recall the information regarding class membership of the input pattern.

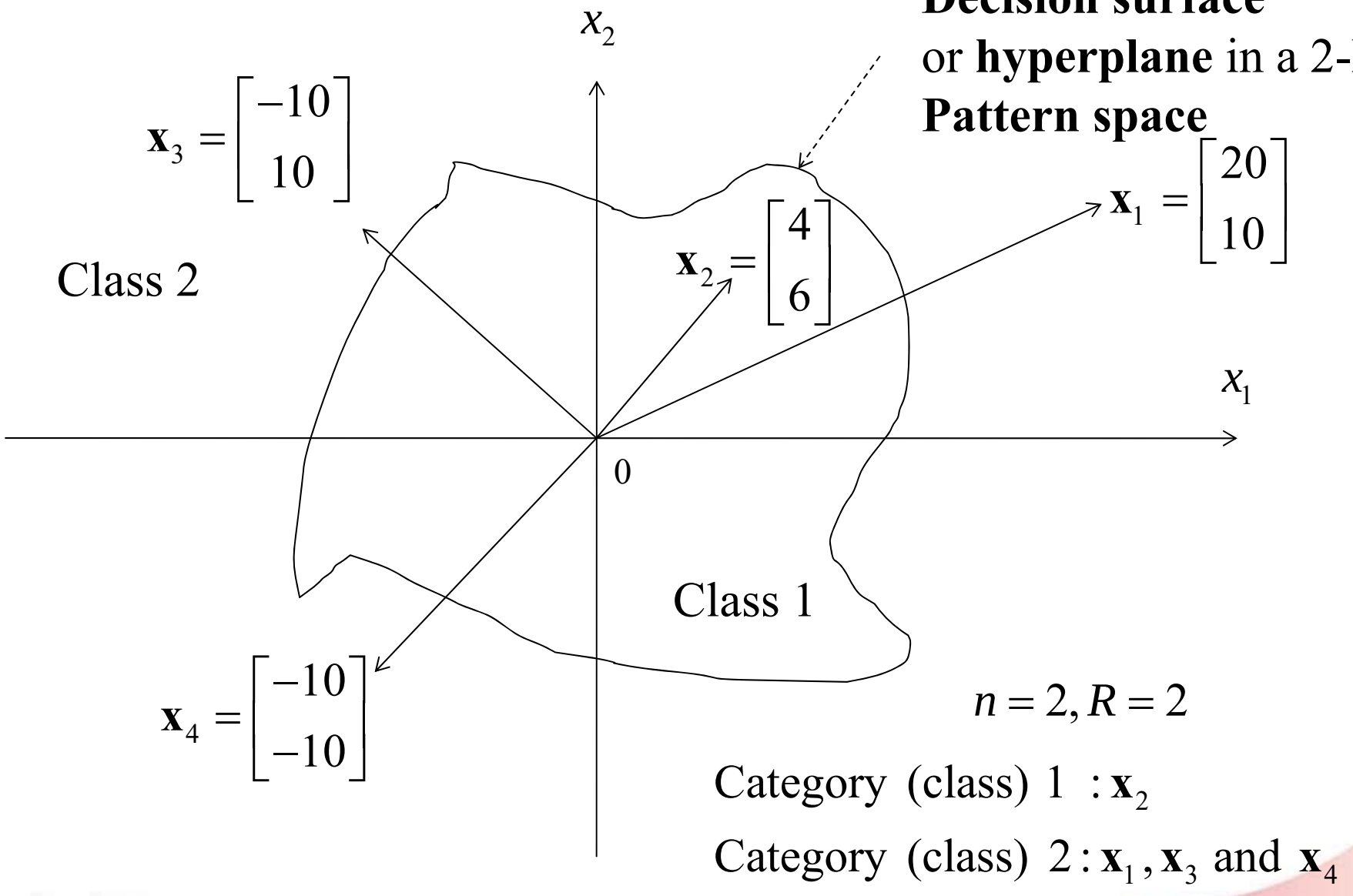


Figure 4.1 **(a)** Nonlinear pattern classification

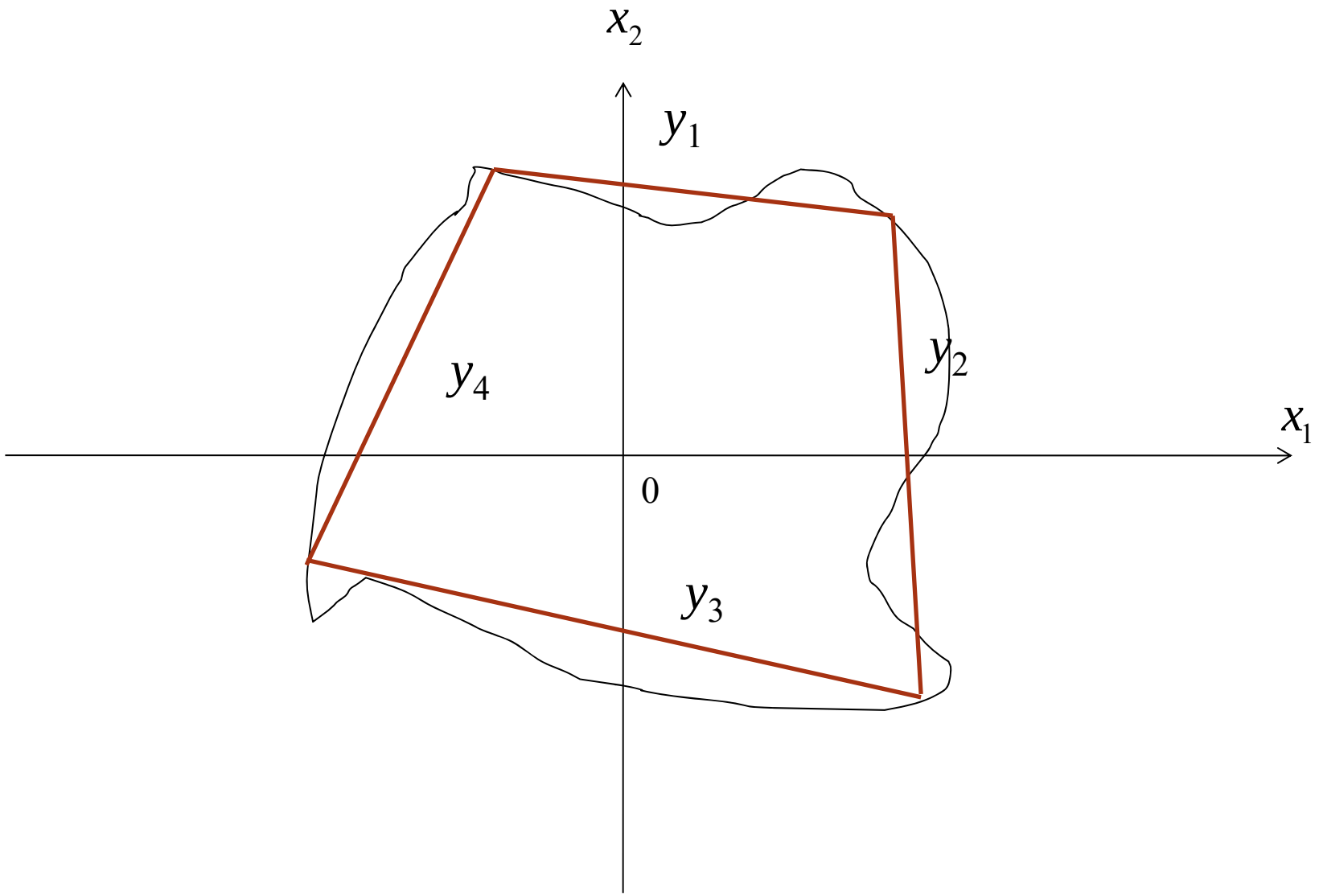
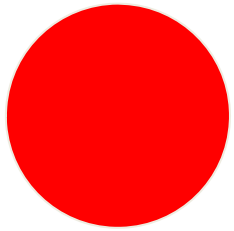
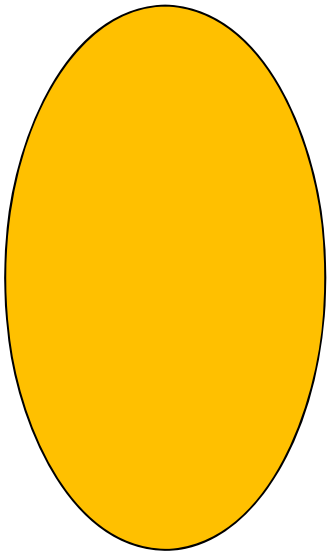


Figure 4.1 (b) Linear Model Approximation

Example 4.0: Classification and Pattern Recognition -- Apple and Orange



Apple



Orange

Using a two-dimensional pattern input vector to classify apple from orange

- **The first feature: Ratio** = $\frac{width}{length}$

Ratio =...**1.01** 1.0 0.99 0.98 …… 0.51 **0.5** 0.49…..

Apple (Ball) Orange (Oval)

- **The second feature: Normalized Frequency** Spectrum (color)

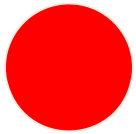
Frequency=..**0.09** 0.1 0.1 …….**0.7** 0.71 …….

Apple (Red) Orange (Yellow)

Define the two-dimensional input pattern vectors

Patter 1:

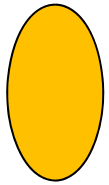
$$\mathbf{x}_1 = \begin{bmatrix} 1.01 \\ 0.09 \end{bmatrix} \begin{matrix} \text{Ball} \\ \text{Red} \end{matrix}$$



Class 1: Apple

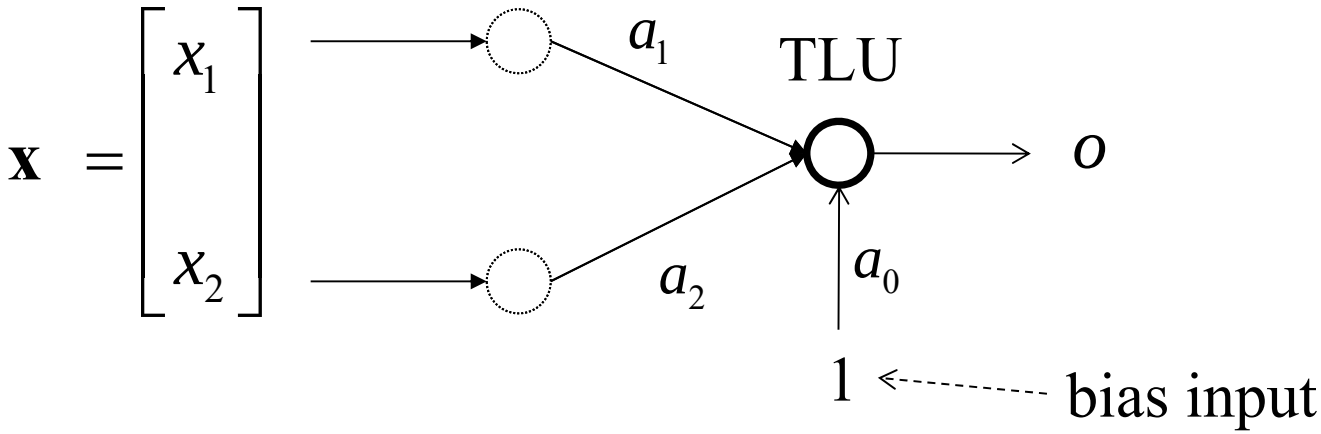
Pattern 2:

$$\mathbf{x}_2 = \begin{bmatrix} 0.5 \\ 0.7 \end{bmatrix} \begin{matrix} \text{Oval} \\ \text{Yellow} \end{matrix}$$



Class 2: Orange

A two-input and single-output neuro-fuzzy classifier



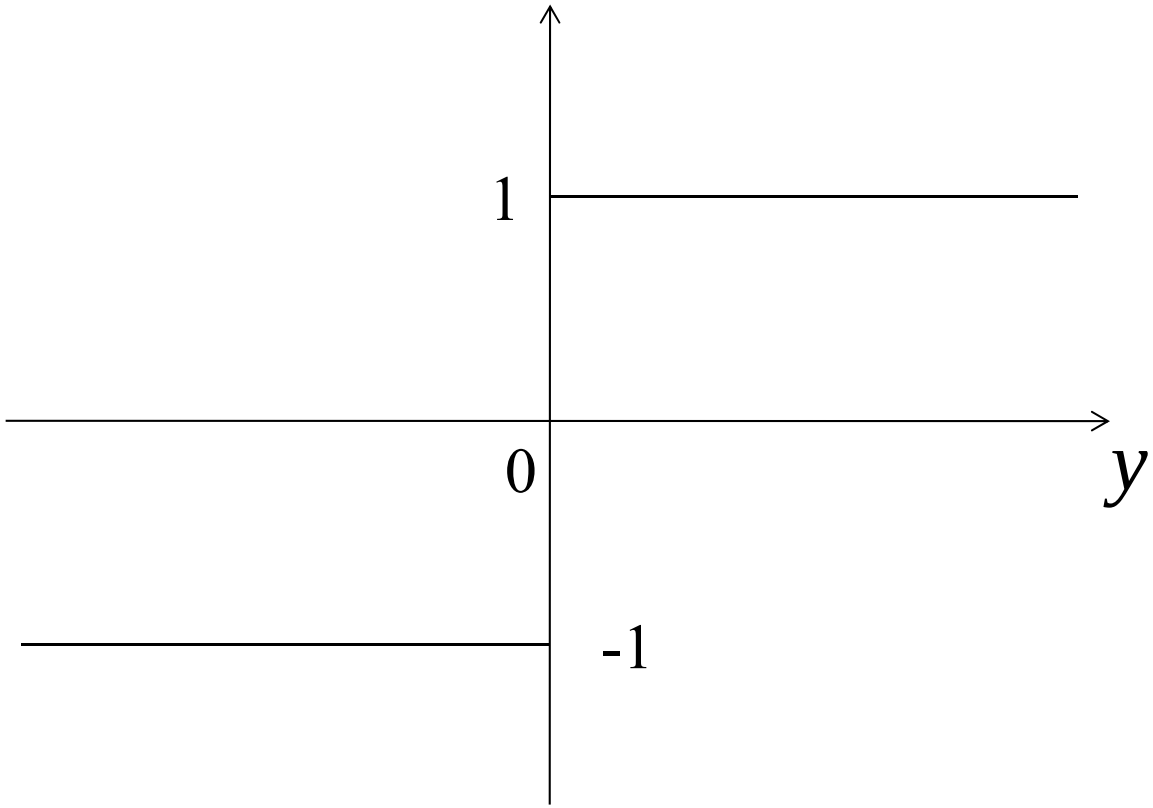
with a single neuron based on a bipolar binary function:

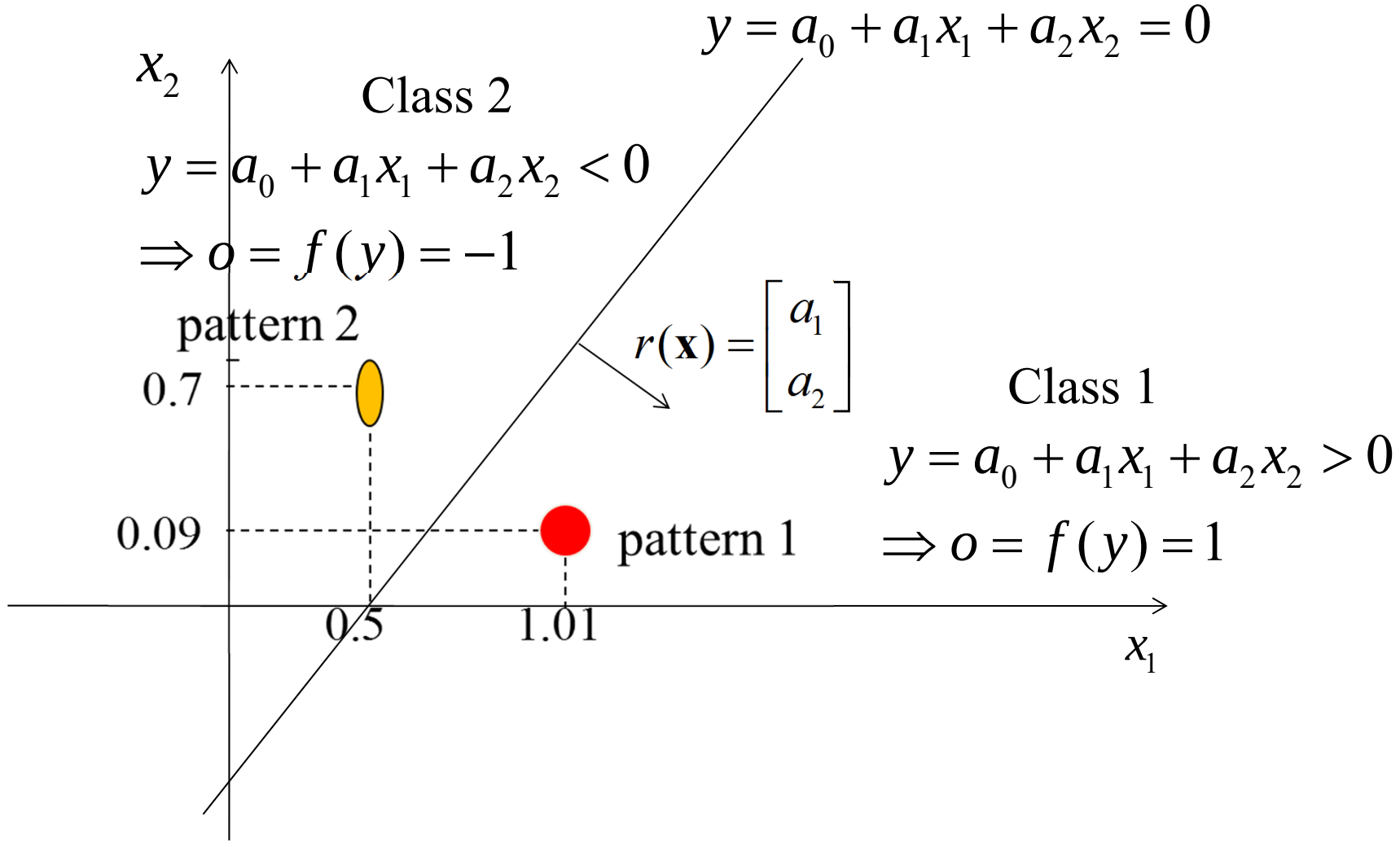
$$o = f(y) = \begin{cases} 1 & y > 0 \\ uncertain & y = 0 \\ -1 & y < 0 \end{cases}$$

$$y = a_0 + a_1 x_1 + a_2 x_2$$

Threshold Logical Unit (TLU): a discrete time neuron activation function

$$o = f(y) = \mathrm{sgn}(y)$$





Concept of a linear hyperplane

4.1 Fuzzy Linear Approximation Model

Modeling, in a general sense, refers to the establishment of a description of a physical system (a hyperplane, etc.) in mathematical terms.

That is, find a mathematical formula or equation that can represent the system, characterizing the input-output behavior of the underlying system.

Such a formulation is by nature a mathematical representation, called a *mathematical model*, of the physical system.

Linear Approximation modeling of the nonlinear system

Consider an unknown system (a nonlinear boundary or hyperplane function in Figure 4.1).

The linear approximation system has inputs x_1, ..., x_n and outputs y_1, ..., $y_{m.}$ They can be measured (or observed) and these data are available to use.

Establishing a mathematical description is called *mathematical linear approximation modeling* for the unknown system.

Such a mathematical description should qualitatively and quantitatively characterize this linear unknown system, in the sense that by inputting x_1, ..., x_n into the mathematical description one can always obtain the corresponding outputs y_1, ..., y_m.

The original nonlinear hyperplane in Figure 4.1 is presented by a set of nonlinear function with neuro-fuzzy network outputs

$$\begin{aligned} o_1 &= f(y_1) \\ &\vdots \\ &\vdots \\ o_N &= f(y_N) \end{aligned} \tag{4.1}$$

The mathematical description of a linear approximation model can be a mathematical formula, such as a mapping or a functional that relates the inputs to the outputs in the form of a set of linear equations in the form

$$\begin{aligned} y_1 &= a_{10} + a_{11}x_1 + \ldots + a_{1n}x_n \\ &\vdots \\ &\vdots \\ y_N &= a_{N0} + a_{N1}x_1 + \ldots + a_{Nn}x_n \end{aligned} \tag{4.2}$$

Linear Fuzzy Systems Modeling:

For many physical systems, they are too vague or too complicated to be described by simple and crisp mathematical formulas or equations.

Interval mathematics and fuzzy logic together can provide an alternative to linear mathematical modeling leading to the so-called *linear fuzzy systems modeling*.

A description using logical linguistic statement in the form

IF (input x1) AND ... AND (input xn)
THEN (output y1) AND ... AND (output ym). (4.3)

Fuzzy systems modeling is to quantify (4.3) by using fuzzy logic and the mathematical functional model (4.2).

4.3 Modeling of Fuzzy Systems Using Linear Approximation

Return to the nonlinear boundary function shown in Figure 4.1.

Model (4.3) can be established by using all the available inputs x_1, ..., x_n and linear approximation outputs y_1, ..., y_m.

Yet this is not the ultimate purpose of mathematical modeling, since the input x_i ($i=1,\ldots n$) come in with uncertainties one does not know what the corresponding output should be.

The main purpose of mathematical linear approximation modeling,

- correctly describe the existing input-output relations through the unknown system
- enable the established model to approximately describe the original nonlinear system., i.e. the hyperplane in Figure 4.1.

Thus, **a general approach** is to

- first quantify the linguistic statement (4.3); and then
- to relate the quantified logical input-output relations by using (4.2).

Recall that a finite fuzzy logic implication statement can always be described by a set of general fuzzy IF-THEN rules in the following multi-input single-output form:

(1) "IF (x_1 is X_{11}) AND ... AND (x_n is X_{1n}) THEN (y is Y_1)."

(2) "IF (x_1 is X_{21}) AND ... AND (x_n is X_{2n}) THEN (y is Y_2)."

…

(N) "IF (x_1 is X_{N1}) AND ... AND (x_n is X_{Nn}) THEN (y is Y_N)."

In the following, X_{11}, ..., X_{Nn} and Y_1, ..., Y_N are all closed intervals.

For simplicity of discussion, first consider the case where $N = 1$ with only one fuzzy IF-THEN rule:

"IF (x_1 is X_1) AND ... AND (x_n is X_n) THEN (y is Y)."

An example is the following rule with constants $\{a_0,a_1,...,a_n\}$:

R^1: IF (x_1 is X_1) AND ... AND (x_n is X_n)
THEN $y = a_0 + a_1x_1 + ... + a_nx_n$.

When a set of particular inputs are available:

$$x_1 = x_1^0 \in X_1, \ldots, x_n = x_n^0 \in X_n, \quad \text{with} \quad \mu_{X_1}(x_1^0), \ldots, \mu_{X_n}(x_n^0),$$

the output y will assume the value

$$y^0 = a_0 + a_1 x_1^0 + \ldots + a_n x_n^0,$$

with membership value given by the *general rule* (see Chapter 2), as

$$\mu_Y(y^0) = \bigvee_{y^0 = a_0 + \ldots + a_n x_n^0} \{\mu_{X_1}(x_1^0) \wedge \ldots \wedge \mu_{X_n}(x_n^0)\}.$$

If there are more than one fuzzy IF-THEN rule (N>1):

R^i: IF (x_1 is X_{i1}) AND ... AND (x_n is X_{in})
THEN $y_i = a_{i0} + a_{i1}x_1 + ... + a_{in}x_n$, i=1,...,N,

then, with the given set of inputs $x_1 = x_1^0 \in X_1, ..., x_n = x_n^0 \in X_n$, namely, the same inputs applied to all the different rules, one will have

$$\begin{aligned} y_1^0 &= a_{10} + a_{11}x_1^0 + ... + a_{1n}x_n^0, \\ y_2^0 &= a_{20} + a_{21}x_1^0 + ... + a_{2n}x_n^0, \\ &\vdots \\ y_N^0 &= a_{N0} + a_{N1}x_1^0 + ... + a_{Nn}x_n^0. \end{aligned} \tag{4.6}$$

The corresponding membership values for these outputs are given as

$$\mu_Y(y_i^0) = \bigvee_{y_i^0 = a_{i0} + \ldots + a_{in} x_n^0} \{\mu_{X_1}(x_1^0) \wedge \ldots \wedge \mu_{X_n}(x_n^0)\}, \quad i = 1, \ldots, N. \tag{4.7}$$

In a typical modeling approach, the final single output, y, is usually obtained via the following *weighted average formula or COG de-fuzzfication*:

$$y = \frac{\sum_{i=1}^{N} \mu_Y(y_i^0) \cdot y_i^0}{\sum_{i=1}^{N} \mu_Y(y_i^0)}, \tag{4.8}$$

.

For the most general situation, assume that all the coefficients $\{a_{i0},a_{i1},...,a_{in} \mid i=1,...,N\}$ are uncertain and belong to certain intervals:

$$a_{i0} \in A_0, \ ..., \ a_{in} \in A_n, \quad i=1,...,N,$$

where, for example,

$$A_0 = [\ min\{a_{10},a_{20},...,a_{N0}\},\ max\{\ a_{10},a_{20},...,a_{N0}\}\ ],$$

$$A_1 = [\ min\{\ a_{11},a_{21},...,a_{N1}\},\ max\{\ a_{11},a_{21},...,a_{N1}\}\ ],$$

$$\vdots$$

$$A_n = [\ min\{a_{1n},a_{2\,n},...,a_{Nn}\ \},\ max\{\ a_{1n},a_{2\,n},...,a_{Nn}\ \}\ ].$$

Thus, with the given inputs $x_1 \in X_1$, $x_2 \in X_2$, ..., $x_n \in X_n$, the output becomes

$$Y = A_0 + A_1 \cdot X_1 + \ldots + A_n \cdot X_n, \tag{4.9}$$

which yields the fuzzy subset (interval) for y, with the membership functions given by the general rule as

$$\mu_{Y,i}(y_i) = \bigvee_{y_i = a_{i0} + \ldots + a_{in} x_n} \{\mu_{X_1}(x_1^0) \wedge \ldots \wedge \mu_{X_n}(x_n^0)\},\ i = 1, \ldots, N. \tag{4.10}$$

Finally, the output y is computed by the weighted average formula, as

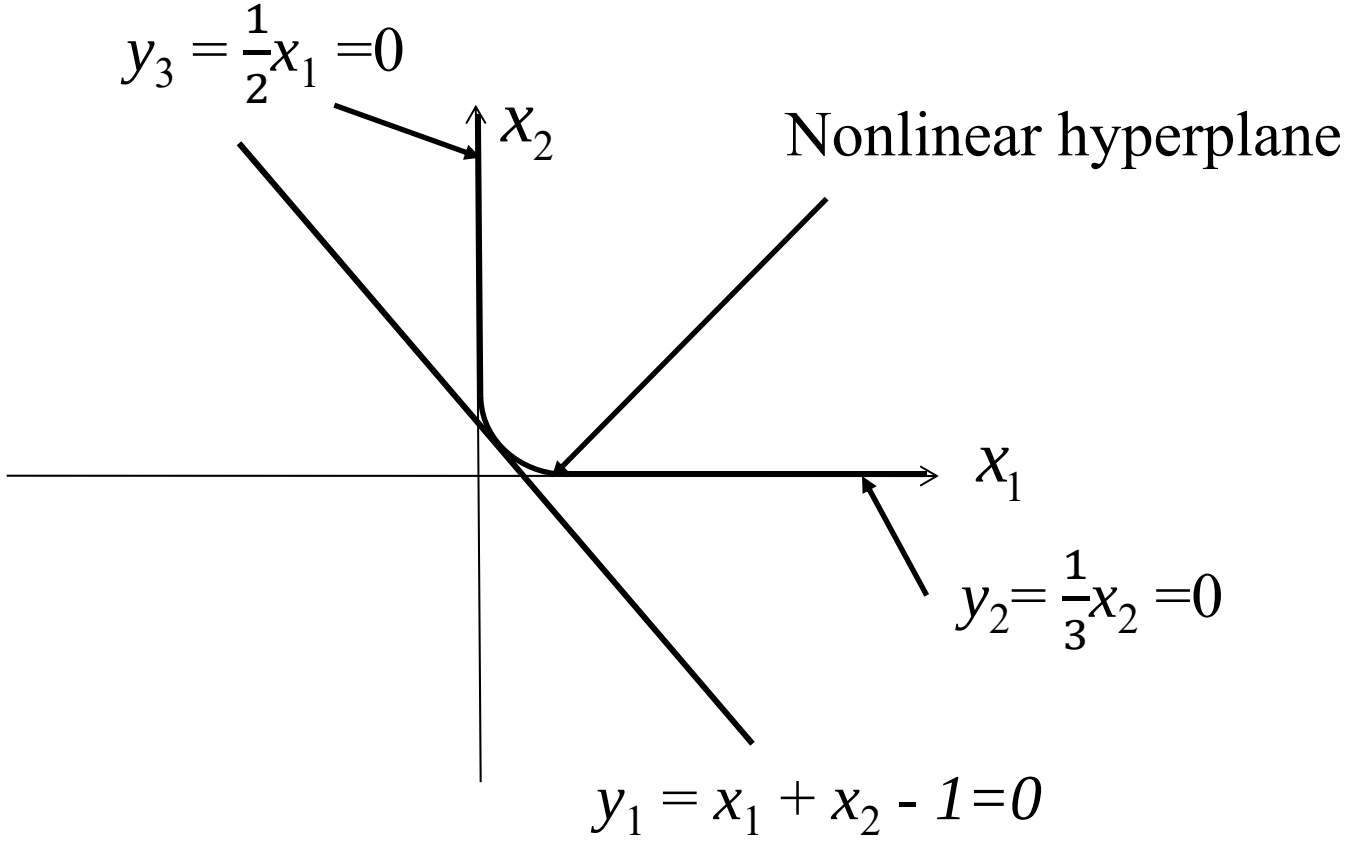
$$y = \frac{\sum_{i=1}^{N} \mu_{Y,i}(y_i) \cdot y_i}{\sum_{i=1}^{N} \mu_{Y,i}(y_i)} = \sum_{i=1}^{N} \beta_i \cdot y_i \tag{4.11}$$

where $$\beta_i = \frac{\mu_{Y,i}(y_i)}{\sum_{i=1}^{N} \mu_{Y,i}(y_i)}.$$

This is a convex combination of the outputs y_i , i=1,...,N.

The three formulas (4.6)-(4.8) or (4.9)-(4.11) are sometimes called the *input-output algorithm* for static fuzzy system modeling under the fuzzy IF-THEN rules R^i, i = 1, ..., N, where the former has constant coefficients while the latter has interval coefficients.

Example 4.1 Linear Approximation of nonlinear hyperplane in Figure 4.2 (a)



(a)

Figure 4.2

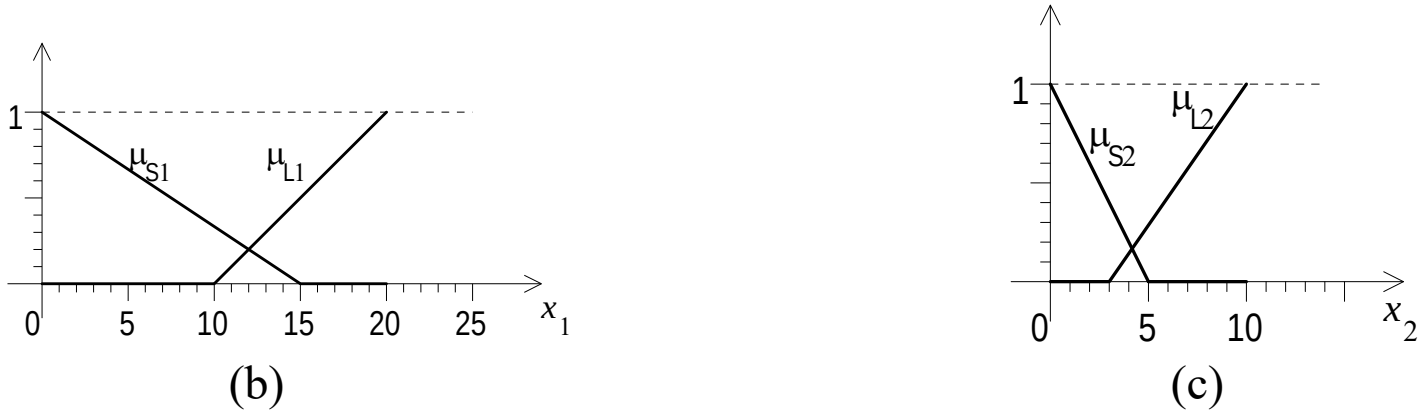


Figure 4.2 The unknown system and its two inputs in Example 4.1

Let inputs x_1 be in the range $X_1 = [0,20]$ and x_2 in $X_2 = [0,10]$. Suppose that X_1 has two associate membership functions, $\mu_{S1}(\cdot)$ and $\mu_{L1}(\cdot)$, describing "small" and "large," respectively, and similarly X_2 has $\mu_{S2}(\cdot)$ and $\mu_{L2}(\cdot)$, as shown in Figure 4.2 (b)-(c).

Now, suppose that from experiments the following input-output relations (fuzzy IF-THEN implication rules) are obtained:

R^1: IF x_1 is small AND x_2 is small THEN $y = x_1 + x_2 - 1$

R^2: IF x_1 is large THEN $y = \frac{1}{2} x_1$

R^3: IF x_2 is large THEN $y = \frac{1}{3} x_2$

Otherwise, $y = 0$.

Let $x_1^0 = 13$ and $x_2^0 = 4$ be given,. It can be found from Figures 4.2 (b)-(c) that

$$\mu_{S_1}(x_1^0) = 2/15,\ \mu_{L_1}(x_1^0) = 3/10,\ \mu_{S_2}(x_2^0) = 1/5,\quad \mu_{L_2}(x_2^0) = 1/7.$$

Then, following the input-output algorithm, compute the following:

(i) The corresponding outputs:

$$y_1^0 = x_1^0 + x_2^0 - 1 = 13 + 4 - 1 = 16$$

$$y_2^0 = \frac{1}{2}x_1^0 = \frac{13}{2}$$

$$y_3^0 = \frac{1}{3}x_2^0 = \frac{4}{3}$$

(ii) The fuzzy set Y:

$$a_{11} = 1, a_{12} = 1, a_{21} = 1/2, a_{22} = 0, a_{31} = 0, a_{32} = 1/3, a_{13} = 1, a_{23} = 0, a_{33} = 0$$

$$\Rightarrow$$

$$A_1 = [\ min\{1, \tfrac{1}{2}, 0\}, max\{1, \tfrac{1}{2}, 0\}\ ] = [0,1],$$

$$A_2 = [\ min\{1, 0, \tfrac{1}{3}\}, \max\{1, 0, \tfrac{1}{3}\}\ ] = [0,1],$$

$$A_3 = [min\{1,0,0\}, max\{1,0,0,\}] = [0,1]$$

$$Y = A_1.X_1 + A_2.X_2 + A_3 = [0,1].[0,20] + [0,1].[0,10] + [-1,0] = [-1,30]$$

(iii) The fuzzy membership values of the outputs:

$$\mu_{Y,1}(y_1^0) = \bigvee_{y_1^0 = x_1^0 + x_2^0 - 1} \{\ \mu_{S_1}(x_1^0) \wedge \mu_{S_2}(x_2^0)\ \}$$

$$= max\{min\{\ \mu_{S_1}(x_1^0)\ ,\ \mu_{S_2}(x_2^0)\ \}\}$$

$$= max\{min\ \{\ \tfrac{2}{15}\ ,\ \tfrac{1}{5}\ \}\}$$

$$= max\{\ \tfrac{2}{15}\ \}$$

$$= \tfrac{2}{15},$$

$$\mu_{Y,2}(y_2^0) = \bigvee_{y_1^0 = \frac{1}{2}x_1^0} \{ min \{ \mu_{L_1}(x_1^0) \} \} = max\{\frac{3}{10}\} = \frac{3}{10},$$

$$\mu_{Y,3}(y_3^0) = \bigvee_{y_1^0 = \frac{1}{3}x_2^0} \{ min \{ \mu_{L_2}(x_2^0) \} \} = max\{\frac{1}{7}\} = \frac{1}{7}.$$

(iv) The average value of the final output, corresponding to the inputs $= 13$ and $= 4$:

$$y^0 = \frac{\sum_{i=1}^{3} \mu_{Y,i}(y_i^0).y_i^0}{\sum_{i=1}^{3} \mu_{Y,i}(y_i^0)} = \frac{\frac{2}{15}.16 + \frac{3}{10}\frac{13}{2} + \frac{1}{7}.\frac{4}{3}}{\frac{2}{15} + \frac{3}{10} + \frac{1}{7}} = 7.4174$$

(v) The three membership functions, one "small" and two "large," for the final output:

$$\mu_S(y) = \begin{cases} -\frac{y}{20} + 1, & 0 \le y \le 20, \\ 0, & 20 \le y \le 30, \end{cases}$$

$$\mu_{L_1}(y) = \begin{cases} 0, & 0 \le y \le 5, \\ \frac{y}{5} - 1, & 5 \le y \le 10, \\ 1 & 10 \le y \le 30, \end{cases}$$

$$\mu_{L_2}(y) = \begin{cases} 0, & 0 \le y \le 1, \\ \frac{3y}{7} - \frac{3}{7}, & 1 \le y \le \frac{10}{3}, \\ 1 & \frac{10}{3} \le y \le 30, \end{cases}$$

in which $\mu_S(y)$ can be computed from μ_{S1} and μ_{S2} by the α-cut operations, and, similarly, $\mu_{L1}(y)$ from $\mu_{L_1}(x_1^0)$ and $\mu_{L2}(y)$ from $\mu_{L_2}(x_2^0)$, respectively.

Here, to find $\mu_{L1}(y)$ from$\mu_{L_1}(x_1^0)$, for instance, one uses α-cut on the membership function curve $\mu_{L1}(x_1)$: $\alpha = (x - 10)/10$, and obtain $x_{11} = 10\alpha + 10$, while $x_{12} = 20$ is obtained from Figure 4.2 (b). Thus, one has $[x_{11},x_{12}] = [10\alpha+10,20]$. It then follows from the implication rule R^2 ($y_2 = \frac{1}{2}x_1$) that the output interval is $[y_{21},y_{22}]$ $= [5\alpha+5,10]$. Then, by setting $y_{21} = 5\alpha + 5$, one has $\alpha = y_{21}/5 - 1$ on $[5,10]$ (since $\mu_{L1}(y = 5) = 0$ and $\mu_{L1}(y = 10) = 1$). These membership functions are finally extended to the interval $Y = [0,30]$, which was determined above. The resulting three output membership functions are shown in Figure 4.3.

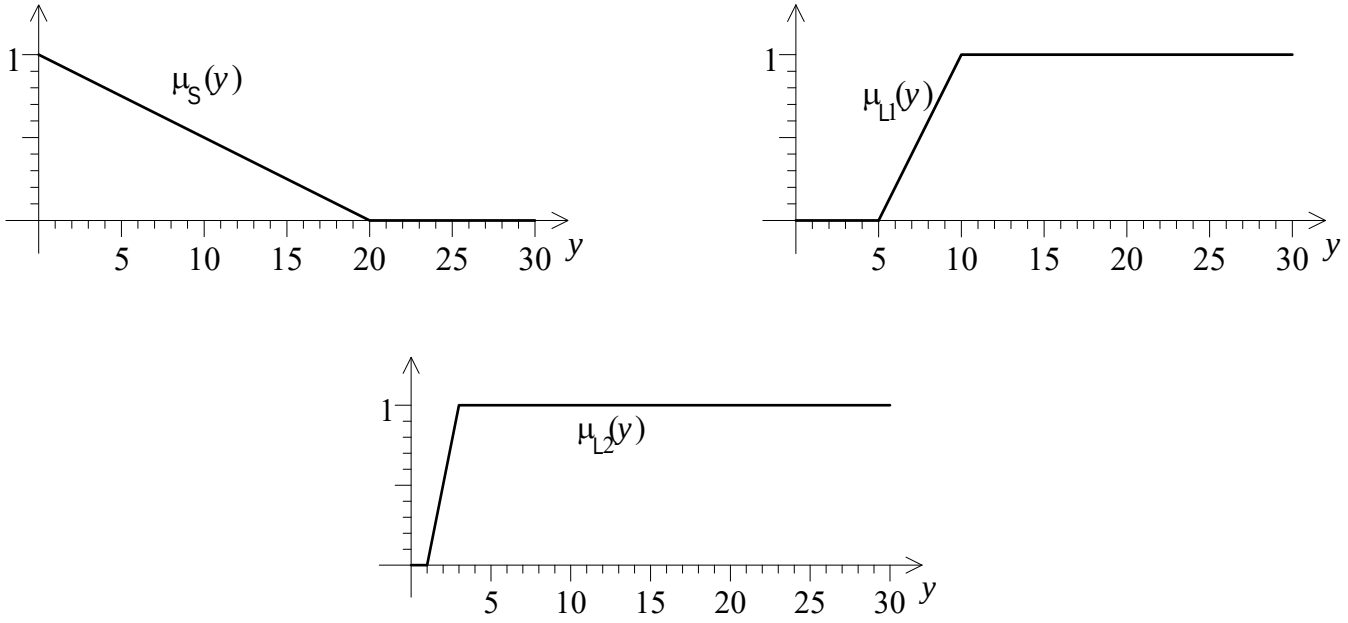
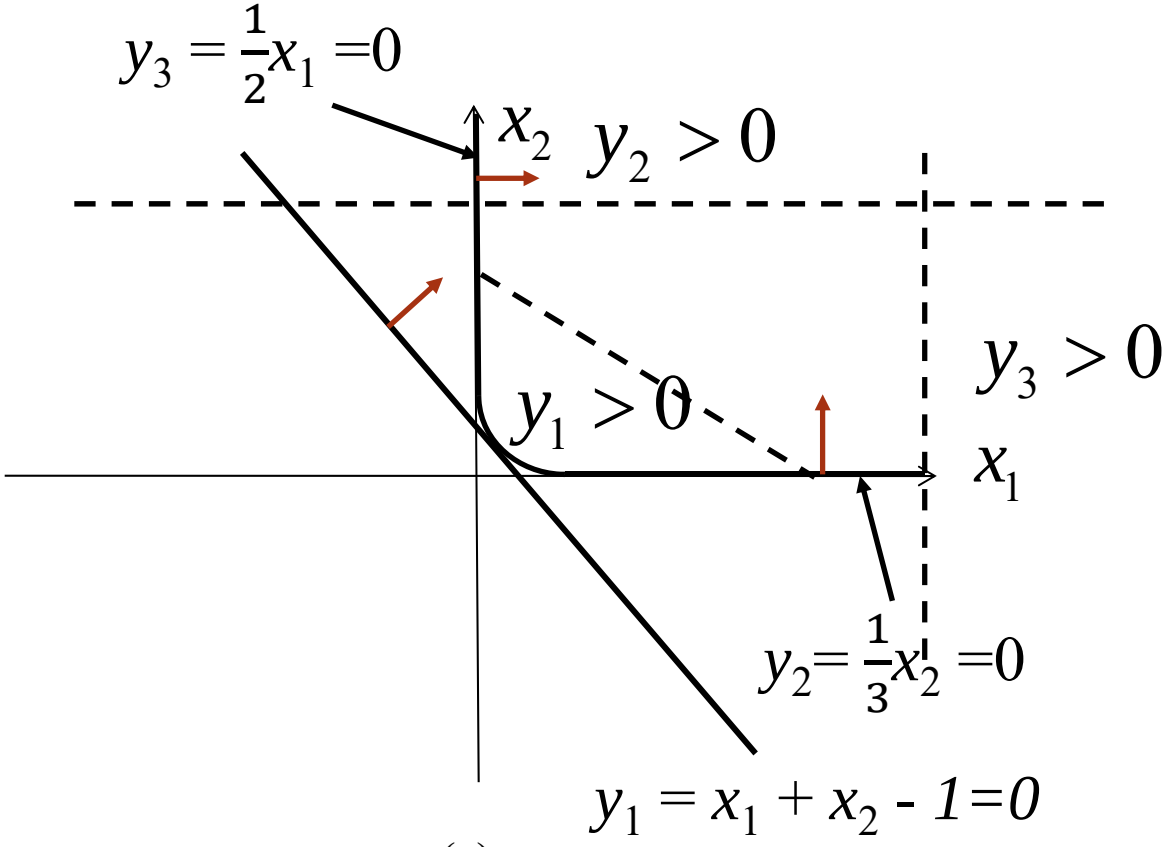


Figure 4.3 The three output membership functions in Example 4.1

Linear Approximation of nonlinear hyperplane in Figure 4.2 (a)



(a)

Figure 4.4

Note that the approach discussed above, including the input-output algorithm and Example 4.1 therein, is about how to describe the input-output relations for an unknown system, where the input-output relations are partially known; namely, the constant coefficients in the relations. In the following, we will use the classical neural network approaches to estimate the parameters, known as the weights of neural networks. We will start with the single output system and use parameter w to replace a to follow the traditional neural network approach:

$$y = w_1 x_1 + \ldots + w_n x_n + w_{n+1} \quad (4.12)$$

4.4 Linear machine and minimum distance classification - parametric algorithm

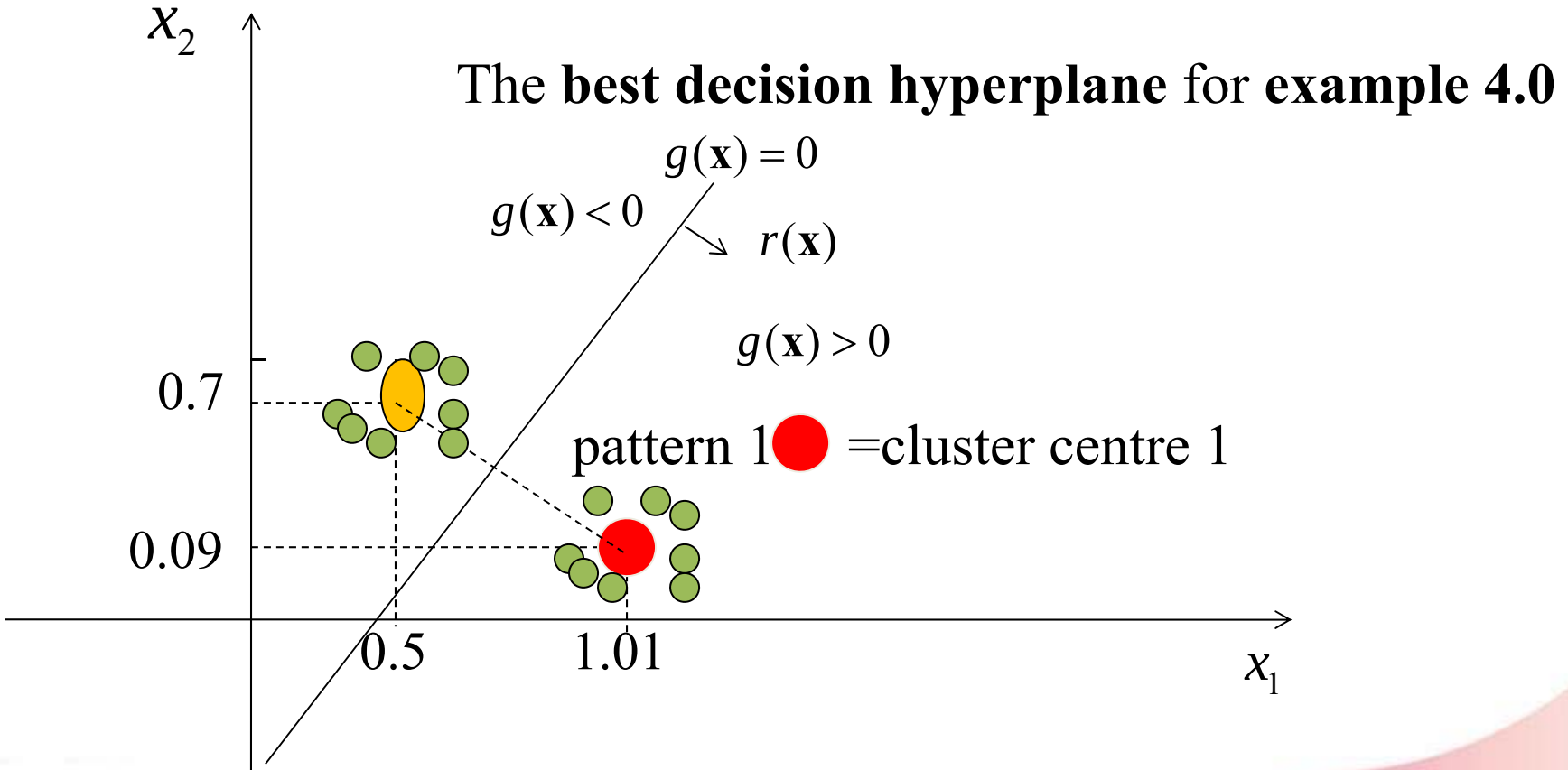
Neural Processing

Recall **- The *process of computation* of output for a given input pattern performed by a network. Its objective is to retrieve the stored information which may have been encoded in the network previously.**

Generalization **- The network is said to generalize well when it sensibly *interpolates* input patterns that are *new* to the network.**

Linear Machine and Minimum Distance Classification

with known **prior information – cluster centres**



4.4.1 Two-class problem

Consider the case that $R = 2$. With a general linear discriminant function $g(\mathbf{x})$, the decision surface in the n-dimensional pattern space is a *hyperplane* and can be expressed as

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1} = 0 \qquad (4.13)$$

or

$$\mathbf{w}^t \mathbf{x} + w_{n+1} = 0$$

or briefly

$$\begin{bmatrix} \mathbf{w} \\ w_{n+1} \end{bmatrix}^t \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = 0$$

where $\mathbf{w}$ is a weight vector to be determined

$$\mathbf{w}^t = \begin{bmatrix} w_1 & w_2 & \ldots & w_n \end{bmatrix}$$

Suppose the center points of the two clusters, denoted as P_1 and P_2, are vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. A decision surface is chosen so that

(1) it contains the midpoint of the line segment connecting P_1 and P_2; and

(2) it is normal to the vector $\mathbf{x}_1$-$\mathbf{x}_2$.

This surface can be described by the equation (see equation (A6.10) of the textbook):

$$(\mathbf{x}_1 - \mathbf{x}_2)^t \mathbf{x} + \frac{1}{2}(\|\mathbf{x}_2\|^2 - \|\mathbf{x}_1\|^2) = 0 \qquad (4.14)$$

where $\|.\|$ denotes the vector *length*.

Comparing (4.13) with (4.14), the weight vector is found to be

$$\mathbf{w} = \mathbf{x}_1 - \mathbf{x}_2$$

$$w_{n+1} = \frac{1}{2}(\|\mathbf{x}_2\|^2 - \|\mathbf{x}_1\|^2) \qquad (4.15)$$

If $\mathbf{x}_1, \mathbf{x}_2$ are known, then a classifier is designed and it is called a two-class minimum-distance classifier.

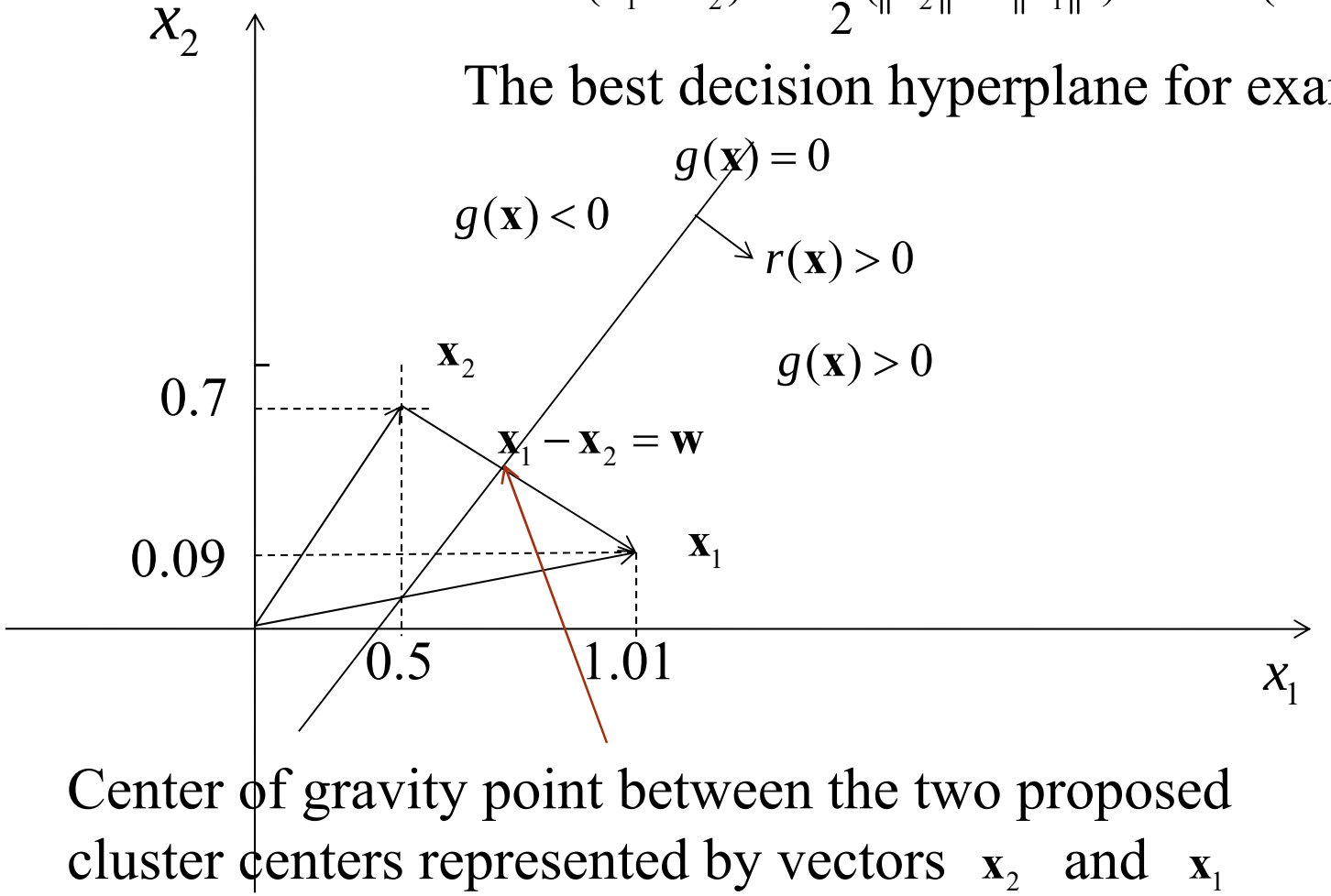
A *minimum-distance classifier* computes the Euclidean distance from the input pattern of unknown classification to each prototype. Then the category number of that *closest* or *smallest distance*, prototype is assigned to the unknown pattern.

Linear Machine and Minimum Distance Classification

$$\mathbf{w}^t\mathbf{x} + w_{n+1} = 0 \qquad (4.13)$$

$$(\mathbf{x}_1 - \mathbf{x}_2)^t\mathbf{x} + \frac{1}{2}(\|\mathbf{x}_2\|^2 - \|\mathbf{x}_1\|^2) = 0 \qquad (4.14)$$

The best decision hyperplane for example 4.0



Center of gravity point between the two proposed cluster centers represented by vectors $\mathbf{x}_2$ and $\mathbf{x}_1$

Example 4.0: (continue) the best solution:

$$\mathbf{w} = \mathbf{x}_1 - \mathbf{x}_2 = \begin{bmatrix} 1.01 - 0.5 \\ 0.09 - 0.7 \end{bmatrix} = \begin{bmatrix} 0.51 \\ -0.61 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$w_3 = \frac{1}{2}(\|\mathbf{x}_2\|^2 - \|\mathbf{x}_1\|^2)$$

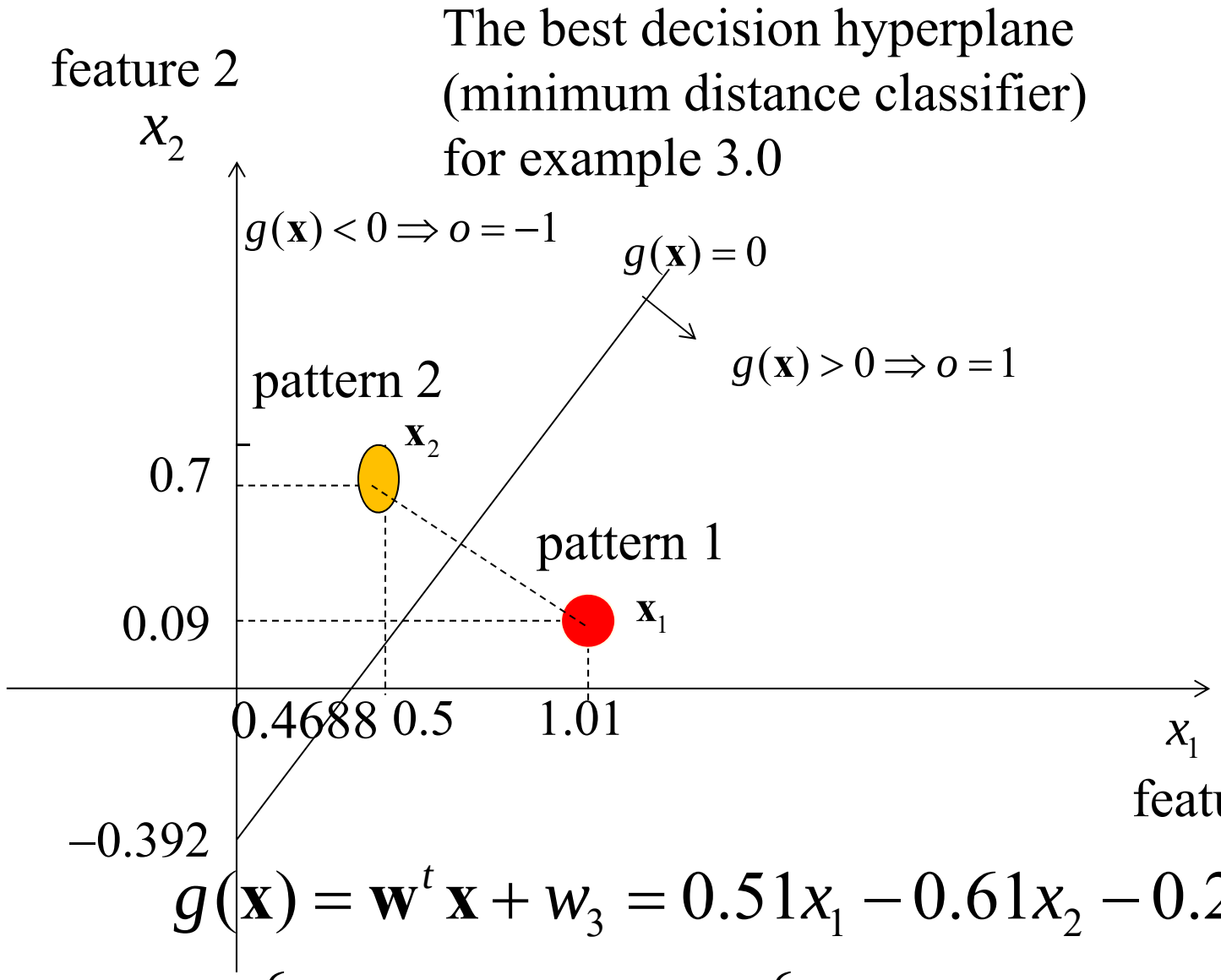
$$= 0.5[(0.5^2 + 0.7^2) - (1.01^2 + 0.09^2)]$$

$$= -0.2391$$

The best decision hyperplane (minimum distance classifier):

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_3 = 0.51x_1 - 0.61x_2 - 0.2391 = 0$$

The best decision hyperplane
(minimum distance classifier)
for example 3.0



$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_3 = 0.51x_1 - 0.61x_2 - 0.2391 = 0$$

$$\Rightarrow \begin{cases} x_1 = 0 \\ x_2 = -0.392 \end{cases} \qquad \begin{cases} x_1 = 0.4688 \\ x_2 = 0 \end{cases}$$

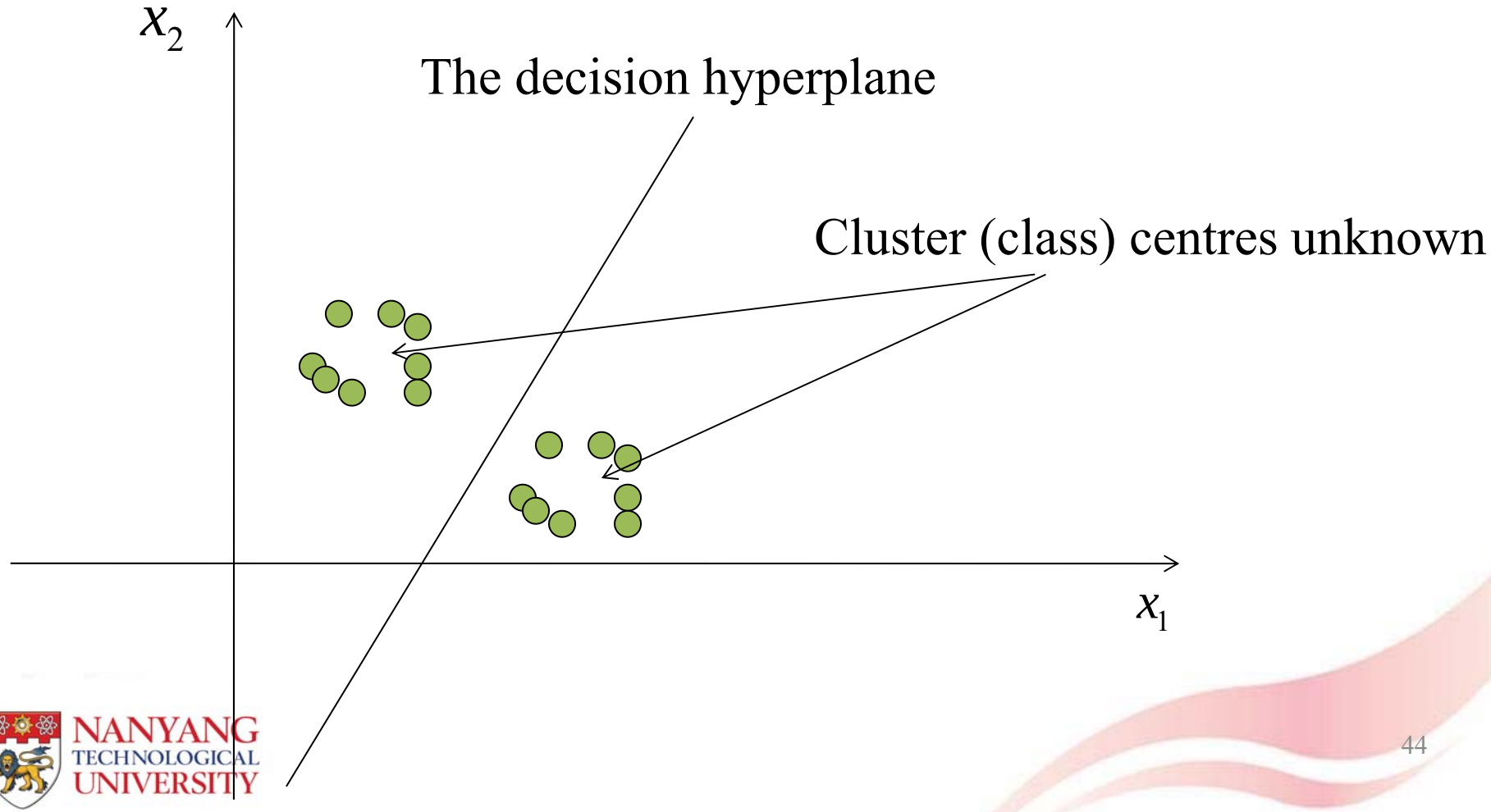
Advantage and disadvantage of the minimum distance classifier

- In a two-cluster(class) classification problem, we need only **two specific input patterns**, which represent the centres of each cluster (class), respectively, to determine the best decision hyperplane.

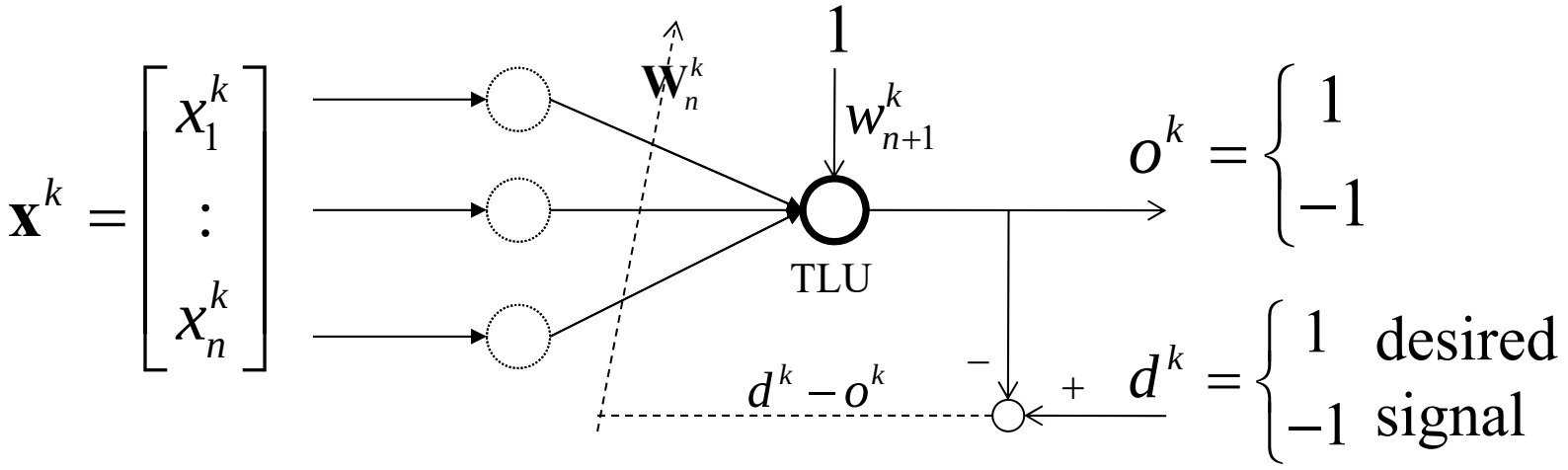
- However, if the true cluster centres are unknown, which is usually the case for practical problem, then there is no solution.

4.5 Perceptron learning – nonparametric algorithm

Question: How do we design the neural network classifier without knowing the cluster centre (**no prior information**)?

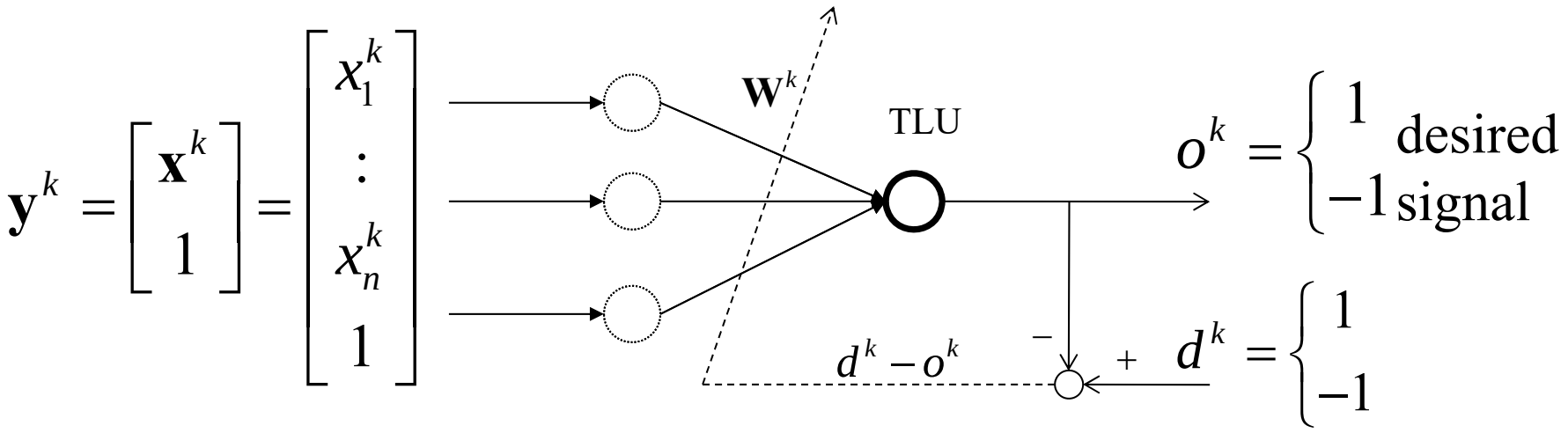


Single layered neural network with a single neuron -a single perceptron learning structure at the current training step *k* through iterative learning



Note: $\mathbf{w}_n^k$ represents a n-dimensional weight vector

A more convenient presentation - a single perceptron learning structure based on the augmented input pattern $\mathbf{y}^k$ at learning step k



Note: $\mathbf{w}^k$ represents a $(n+1)$-dimensional weight vector

For two class classification problem using TLU neuron

$$\text{Define} \quad d^k = \begin{cases} 1 & \text{class } 1 \\ -1 & \text{class } 2 \end{cases} \quad \text{and} \quad o^k = \begin{cases} 1 & \text{class } 1 \\ -1 & \text{class } 2 \end{cases}$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \frac{c}{2}\{d^k - \text{sgn}([\mathbf{w}^k]^t \mathbf{y}^k)\}\mathbf{y}^k = \mathbf{w}^k + \frac{c}{2}\{d^k - o^k\}\mathbf{y}^k$$

$$= \begin{cases} \mathbf{w}^k + c\mathbf{y}^k & \text{if } d^k = 1, o^k = -1 \text{ misclassification, positive increment} \\ \mathbf{w}^k & \text{if } d^k = o^k \text{ correct classification} \\ \mathbf{w}^k - c\mathbf{y}^k & \text{if } d^k = -1, o^k = 1 \text{ misclassification,negtive increment} \end{cases}$$

$1 \geq c > 0$ is the learning rate

The learning procedure and data flow for learning steps *k=1,2,……*

	Input/desired patterns	Original weight		Output		New weight
$k=1$	$\mathbf{y}^1=\mathbf{y}_1, d^1=d_1 \Rightarrow$	$\mathbf{w}^1$	$\Rightarrow$	o^1	$\Rightarrow$	$\mathbf{w}^2$
$\vdots$	$\vdots$					$\vdots$
$k=p$	$\mathbf{y}^p=\mathbf{y}_p, d^p=d_p \Rightarrow$	$\mathbf{w}^p$	$\Rightarrow$	o^p	$\Rightarrow$	$\mathbf{w}^{p+1}$
$k=p+1$	$\mathbf{y}^{p+1}=\mathbf{y}_1, d^{p+1}=d_1$	$\mathbf{w}^{p+1}$		o^{p+1}		$\mathbf{w}^{p+2}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
$k=2p$	$\mathbf{y}^{2p}=\mathbf{y}_p, d^{2p}=d_p$	$\mathbf{w}^{2p}$		o^{2p}		$\mathbf{w}^{2p+1}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$

Example 4.2 Given four pairs $(p = 4)$ of 1-D patterns in two classes

Class 1 $\quad x_1 = 1, x_3 = 3, d_1 = d_3 = 1$

Class 2 $\quad x_2 = -0.5, x_4 = -2, d_2 = d_4 = -1$

The augmented input training pattern vectors are

$$\mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{y}_2 = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}, \mathbf{y}_3 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{y}_4 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

$$d_1 = 1, d_2 = -1, d_3 = 1, d_4 = -1$$

with perceptron learning rule with the initial weight $\mathbf{w}^1 = \begin{bmatrix} -2.5 \\ 1.75 \end{bmatrix}$

$$\mathbf{w}^{k+1} = \mathbf{w}^k \pm \mathbf{y}^k \qquad c=1 \qquad \text{for TLU neuron}$$

Step $k=1$; Training pattern pair $\mathbf{y}^1 = \mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, d^1 = d_1 = 1$ is used based on the initial weight $\mathbf{w}^1 = \begin{bmatrix} -2.5 \\ 1.75 \end{bmatrix}$

$$o^1 = \text{sgn}([\mathbf{w}^1]^t \mathbf{y}^1) = \text{sgn}(\begin{bmatrix} -2.5 & 1.75 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = -1$$

$$d^1 - o^1 = 2$$

$$\mathbf{w}^2 = \mathbf{w}^1 + \mathbf{y}^1 = \begin{bmatrix} -2.5 \\ 1.75 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.5 \\ 2.75 \end{bmatrix}$$

Updated decision line $[\mathbf{w}^2]^t \mathbf{y} = \begin{bmatrix} -1.5 & 2.75 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -1.5 y_1 + 2.75 y_2 = 0$

+ Learning sequence notations for training data pairs:

As learning step k increasing we need to use the 4 given input patterns and desired outputs repeatedly to circle the training data pairs

$$\left.\begin{array}{l}\mathbf{y}^1=\mathbf{y}_1, d^1=d_1\\ \mathbf{y}^2=\mathbf{y}_2, d^2=d_2\\ \mathbf{y}^3=\mathbf{y}_3, d^3=d_3\\ \mathbf{y}^4=\mathbf{y}_4, d^4=d_4\end{array}\right\} p=4, \text{pattern circle one}$$

$$\left.\begin{array}{l}\mathbf{y}^5=\mathbf{y}_1, d^5=d_1\\ \mathbf{y}^6=\mathbf{y}_2, d^6=d_2\\ \ldots.\end{array}\right\} p=4, \text{pattern circle two}$$

Note: $\mathbf{y}$ represents augmented input vector variable;

$\mathbf{y}^k$ represents augmented input vector at learning step $k=1,2,\ldots$

$\mathbf{y}_i$ represents augmented input patterns $1 \le i \le p$

Step $k=1$; Training pattern pair $\mathbf{y}^1 = \mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, d^1 = d_1 = 1$ is used based on the initial weight $\mathbf{w}^1 = \begin{bmatrix} -2.5 \\ 1.75 \end{bmatrix}$

$$o^1 = \text{sgn}([\mathbf{w}^1]^t \mathbf{y}^1) = \text{sgn}(\begin{bmatrix} -2.5 & 1.75 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = -1$$

$$d^1 - o^1 = 2$$

$$\mathbf{w}^2 = \mathbf{w}^1 + \mathbf{y}^1 = \begin{bmatrix} -2.5 \\ 1.75 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.5 \\ 2.75 \end{bmatrix}$$

Updated decision line $[\mathbf{w}^2]^t \mathbf{y} = \begin{bmatrix} -1.5 & 2.75 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -1.5 y_1 + 2.75 y_2 = 0$

Step $k=2$; Training pattern pair $\mathbf{y}^2 = \mathbf{y}_2 = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}, d^2 = d_2 = -1$ is used based on the updated weight $\mathbf{w}^2 = \begin{bmatrix} -1.5 \\ 2.75 \end{bmatrix}$

$$o^2 = \text{sgn}([\mathbf{w}^2]^t \mathbf{y}^2) = \text{sgn}(\begin{bmatrix} -1.5 & 2.75 \end{bmatrix} \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}) = 1$$

$$d^1 - o^1 = -2$$

$$\mathbf{w}^3 = \mathbf{w}^2 - \mathbf{y}^2 = \begin{bmatrix} -1.5 \\ 2.75 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1.75 \end{bmatrix}$$

Updated decision line $[\mathbf{w}^3]^t \mathbf{y} = \begin{bmatrix} -1 & 1.75 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -y_1 + 1.75 y_2 = 0$

Step $k=3$; Training pattern pair $\mathbf{y}^3 = \mathbf{y}_3 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, d^3 = d_3 = 1$ is used based on the updated weight $\mathbf{w}^3 = \begin{bmatrix} -1 \\ 1.75 \end{bmatrix}$

$$o^3 = \text{sgn}([\mathbf{w}^3]^t \mathbf{y}^3) = \text{sgn}(\begin{bmatrix} -1 & 1.75 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix}) = -1$$

$$d^3 - o^3 = 2$$

$$\mathbf{w}^4 = \mathbf{w}^3 + \mathbf{y}^3 = \begin{bmatrix} -1 \\ 1.75 \end{bmatrix} + \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2.75 \end{bmatrix}$$

Updated decision line $[\mathbf{w}^4]^t \mathbf{y} = \begin{bmatrix} 2 & 2.75 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 2y_1 + 2.75y_2 = 0$

Step k=4: Training pattern pair $\mathbf{y}^4 = \mathbf{y}_4 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, d^4 = d_4 = -1$

is used based on the updated weight $\mathbf{w}^4 = \begin{bmatrix} 2 \\ 2.75 \end{bmatrix}$

No misclassification! $d^4 - o^4 = 0$

$$\mathbf{w}^5 = \mathbf{w}^4$$

Step k=5: Training pattern pair $\mathbf{y}^5 = \mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, d^5 = d_1 = 1$

is used based on the updated weight $\mathbf{w}^5 = \begin{bmatrix} 2 \\ 2.75 \end{bmatrix}$

No misclassification! $d^5 - o^5 = 0$

Circling use of the input pattern 1!

$$\mathbf{w}^6 = \mathbf{w}^5$$

No Change of decision line $[\mathbf{w}^4]^t \mathbf{y} = [\mathbf{w}^5]^t \mathbf{y} = [\mathbf{w}^6]^t \mathbf{y} = 0$

Step k=6; Training pattern pair $\mathbf{y}^6 = \mathbf{y}_2 = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}, d^6 = d_2 = -1$ is used based on the updated weight $\mathbf{w}^6 = \begin{bmatrix} 2 \\ 2.75 \end{bmatrix}$

Circling use of the second pair of training data!

$$o^6 = \text{sgn}([\mathbf{w}^6]^t \mathbf{y}^6) = \text{sgn}(\begin{bmatrix} 2 & 2.75 \end{bmatrix} \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}) = 1$$

$$d^6 - o^6 = -2$$

$$\mathbf{w}^7 = \mathbf{w}^6 + \frac{1}{2}\{d^6 - o^6\}\mathbf{y}^6 = \mathbf{w}^6 - \mathbf{y}^6 = \begin{bmatrix} 2 \\ 2.75 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 1.75 \end{bmatrix}$$

Updated decision line $[\mathbf{w}^7]^t \mathbf{y} = \begin{bmatrix} 2.5 & 1.75 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 2.5y_1 + 1.75y_2 = 0$

Step *k=7-9*: readers can verify

$$\mathbf{w}^{10} = \mathbf{w}^{9} = \mathbf{w}^{8} = \mathbf{w}^{7}$$

Step *k=10*:

$$o^{10} = \text{sgn}([\mathbf{w}^{10}]^t \mathbf{y}^{10}) = \text{sgn}(\begin{bmatrix} 2.5 & 1.75 \end{bmatrix} \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}) = 1$$

$$d^{10} - o^{10} = -2$$

$$\mathbf{w}^{11} = \mathbf{w}^{10} - \mathbf{y}^{10} = \begin{bmatrix} 2.5 \\ 1.75 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 0.75 \end{bmatrix}$$

The final decision line: $[\mathbf{w}^{11}]^t \mathbf{y} = \begin{bmatrix} 3 & 0.75 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 3y_1 + 0.75y_2 = 0$

After step *k=11*: There is no more updating for the circle use of all the training data pairs.

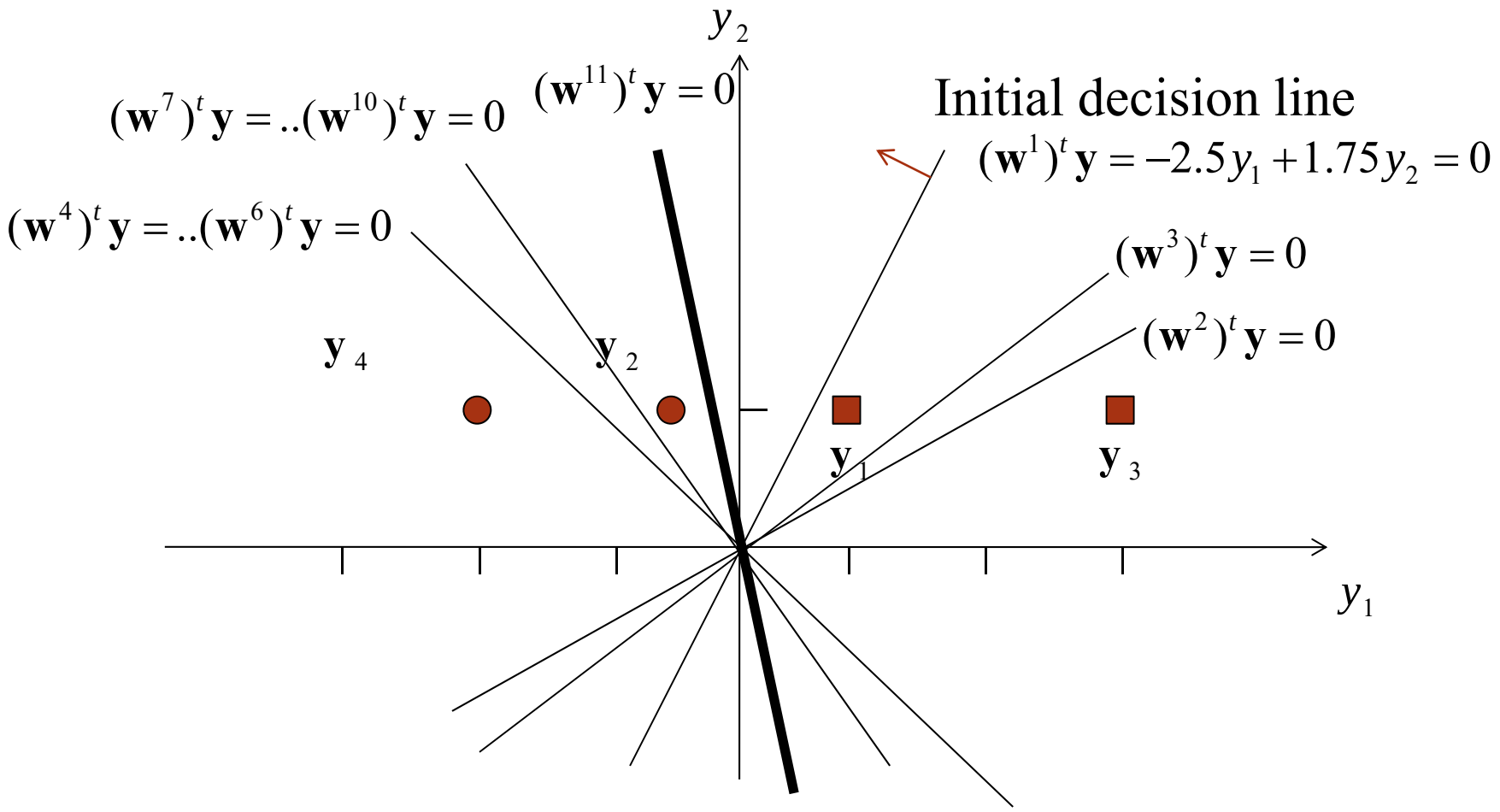
To gain confidence on $\mathbf{w}^{11}$ **, we can go through *one more training cycle*,**

i.e. ***circle to use the four input training patterns.*** *It is found that* $\mathbf{w}^{11}$ is not updated.

This implies the *accumulated training error is zero within this cycle.*

Thus the training stops and $\mathbf{w}^{11}$ **is the correct weights.**

Learning steps at augmented 2-D pattern space of example 3.2



Clearly, the training process can be generalized to train networks with *higher dimensional* pattern vectors and also *interconnected* single layer perceptrons.

It can be shown that for any single binary perceptron, the training will stop after a *finite* number of steps for arbitrarily chosen initial weights if a solution area exists for the training pairs provided. This is called *Perceptron Convergence Theorem*.

- Alternative to *Step 6*: Since the decision line in the pattern space or augmented pattern space does not give a final solution area and need to circle the whole learning procedure again even an one-step training error is going to zero until a complete circle error for all given patterns is zero, therefore, a more convincing method is: pre-determine the weight solution area in the augmented weight space based on **only the given input patterns (no repeated use)**. The final weight of perceptron learning will be located as long as the updated weight falls into the solution region.

A *multilayer feedforward network* with *linear* discriminant functions employed at each layer can be trained to classify linearly non-separable patterns and perform many other complicated tasks.

However, the learning procedure is rather complex, known as the back- prorogation algorithm. We may also take a similar graphic approach in the 2-D input pattern and image planes to look for a quick solution, similar to the perceptron learning procedure in Chapter 3.

4.4 **Linearly Non-Separable Pattern Classification**

Idea:

(1) The first layer of a network is used to map the original non-separable pattern sets in the pattern space into their *image sets,* which are *linearly separable,* in a *image space.*

(2) The second layer is designed to classify the *image sets* in the image space.

- Thus the composite two layer network can eventually classify the patterns that are linearly non-separable in the original pattern space.

A block diagram of this process is shown in Figure

.

Figure A Block Diagram of a Two Layer Network

Design a two layer network to do the following classification:

$$\left\{\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix}\right\}: \quad \textbf{Class 1}$$

$$\left\{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix}\right\}: \quad \textbf{Class 2}$$

The four patterns are shown in the pattern space as in Figure 4.3. Clearly they are linearly non-separable.

Two layered neural network with three neurons

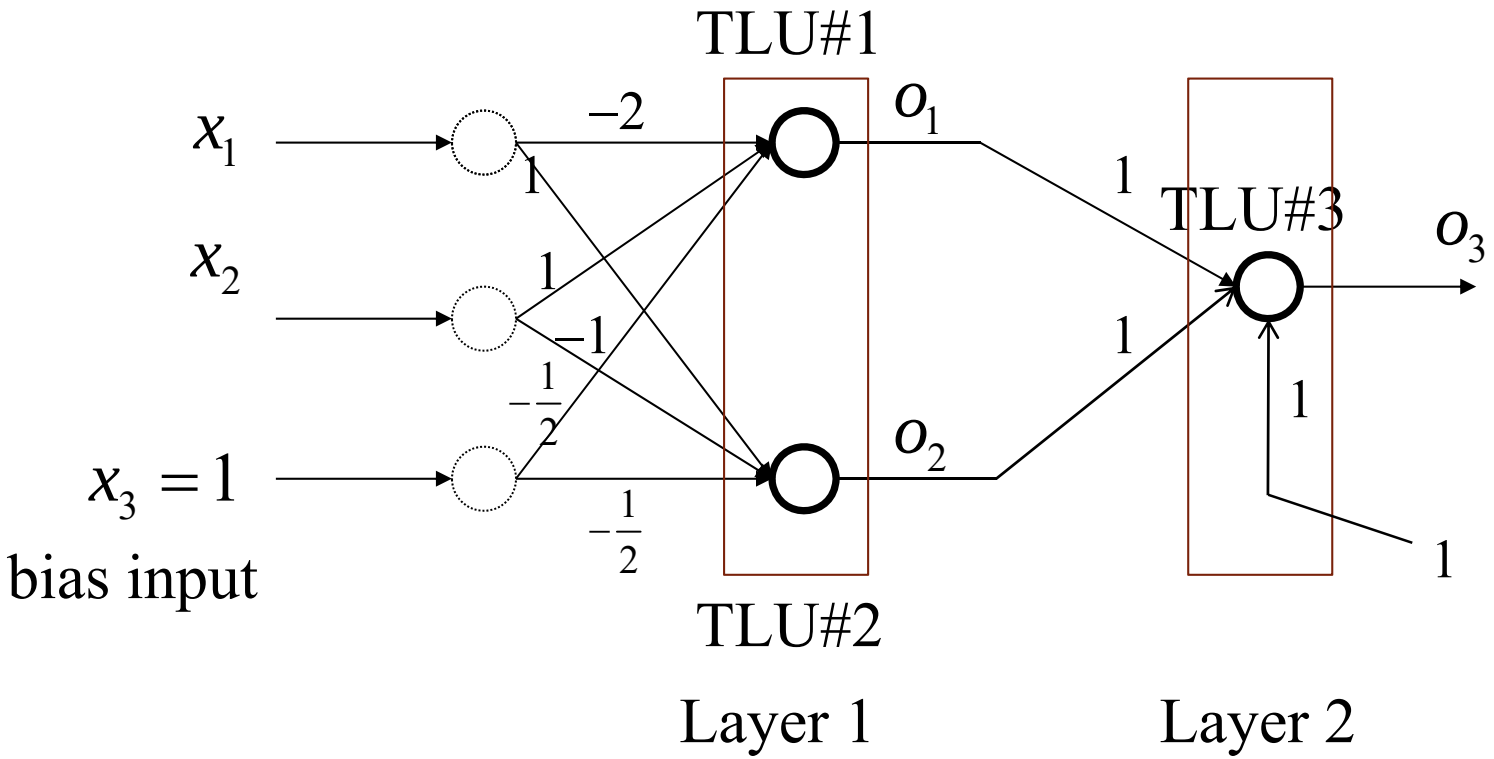


Figure 4.3 for example 4.1

For layer 1

$$\mathbf{w}_1 = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = \begin{bmatrix} -2 & 1 & -\frac{1}{2} \\ 1 & -1 & -\frac{1}{2} \end{bmatrix}$$

$$o_1 = \text{sgn}(-2x_1 + x_2 - \frac{1}{2})$$

$$o_2 = \text{sgn}(x_1 - x_2 - \frac{1}{2})$$

For layer 2

$$\mathbf{w}_2 = \begin{bmatrix} w_{11} & w_{12} & w_{13} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$o_3 = \text{sgn}(o_1 + o_2 + 1)$$

Linearly non-separable patterns

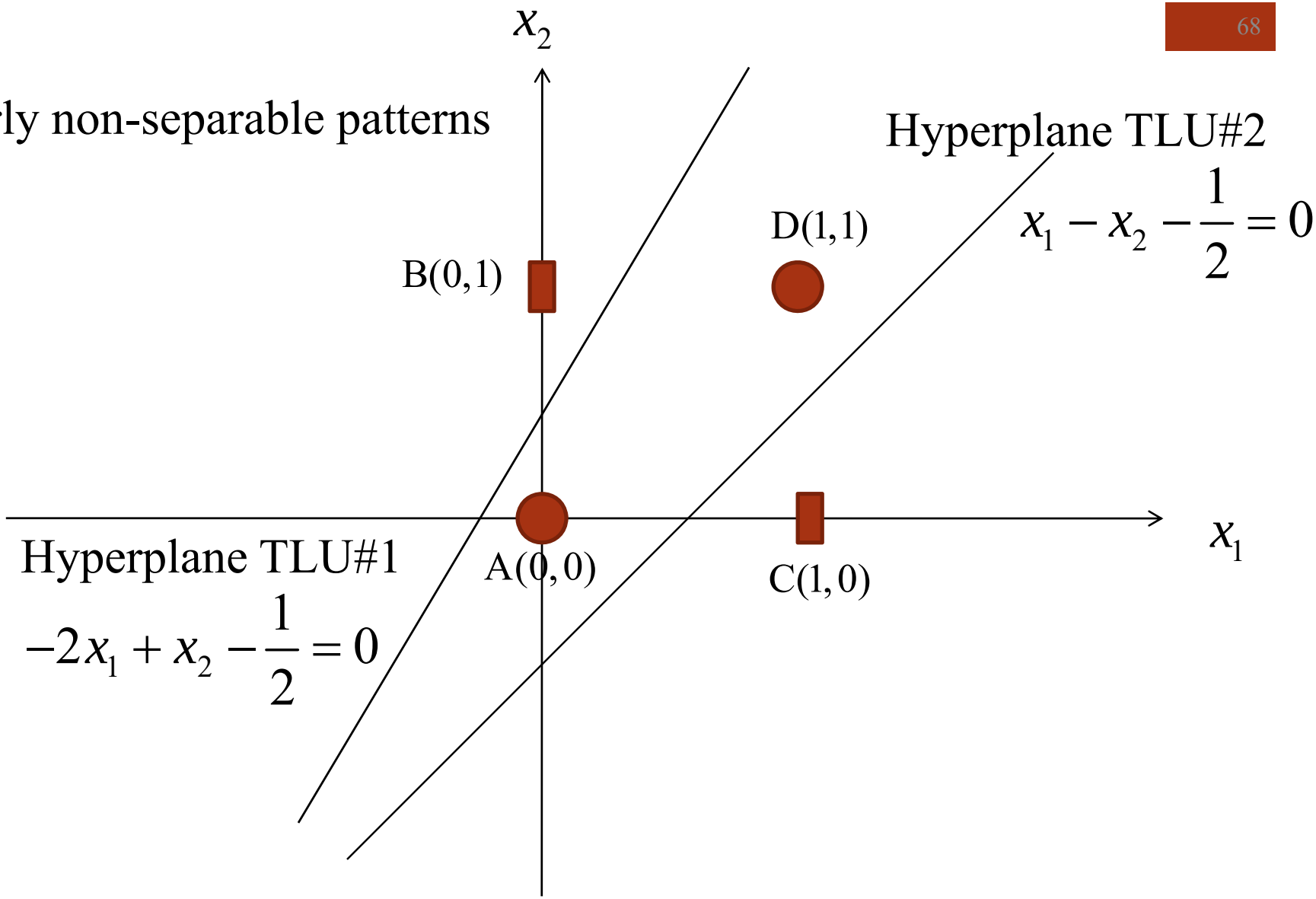


Figure 4.4 Hyperplane in input pattern space of example 4.1

+ Input pattern space x_1, x_2 :

Hyperplane from TLU#1

$$-2x_1 + x_2 - \frac{1}{2} = 0$$

Hyperplane from TLU#2

$$x_1 - x_2 - \frac{1}{2} = 0$$

Image space o_1, o_2 :

The mapping performed by this layer is

$$o_1 = \mathbf{sgn}(-2x_1 + x_2 - \frac{1}{2})$$
$$o_2 = \mathbf{sgn}(x_1 - x_2 - \frac{1}{2}) \qquad (4.2)$$

The images of the four patterns under the mapping are shown in Figure 4.5 in the image space.

Hyperplane TLU#3

$o_1 + o_2 + 1 = 0$

Linearly separable pattern in the image space

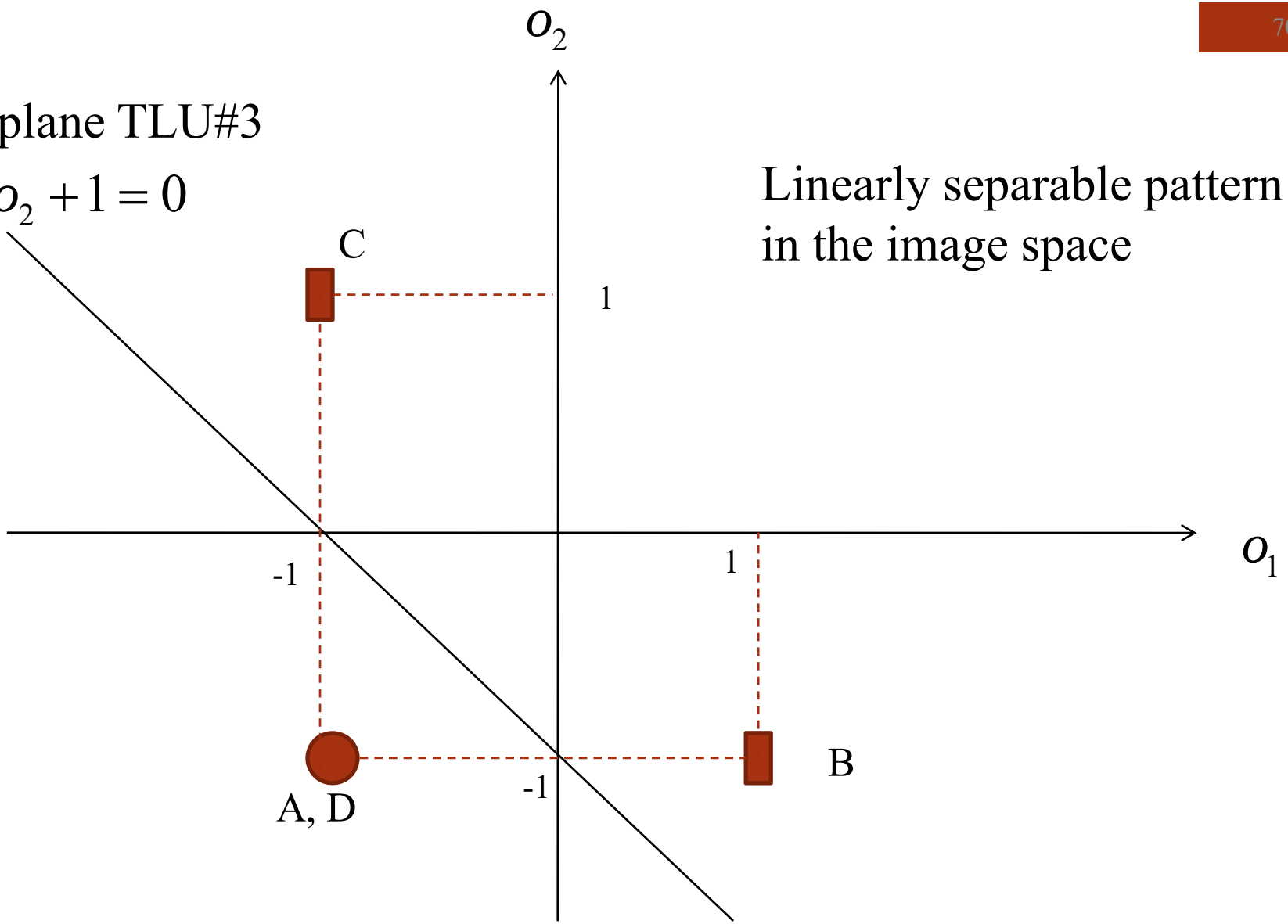


Figure 4.5 Image space of example 4.1

	x_1	x_2	o_1	o_2	$o_1 + o_2 + 1$	o_3	Class
A	0	0	-1	-1	-	-1	2
B	0	1	1	-1	+	1	1
C	1	0	-1	1	+	1	1
D	1	1	-1	-1	-	-1	2

Pattern space and image space data in example 4.1

(2) Design of the Second Layer

From Figure 4.5, an arbitrary decision line can be chosen to separate the images of A,D and B,C in the image space. The line selected is

$$o_1 + o_2 + 1 = 0$$

Thus the TLU #3 implements the decision o_3 as

$$o_3 = \mathbf{sgn}(o_1 + o_2 + 1)$$

Summary:

Classification of Linearly Non-Separable Patterns: