Single-Layer Perceptron Classifiers

Berlin Chen, 2002

Outline

- Foundations of trainable decision-making networks to be formulated
  - Input space to output space (classification space)
- Focus on the classification of linearly separable classes of patterns
  - Linear discriminating functions and simple correction function
  - Continuous error function minimization
- Explanation and justification of perceptron and delta training rules

Classification Model, Features, and Decision Regions

- A pattern is the quantitative description of an object, event, or phenomenon
  - **Spatial patterns**: weather maps, fingerprints …
  - **Temporal patterns**: speech signals …

- Pattern classification/recognition
  - Assign the input data (a physical object, event, or phenomenon) to one of the pre-specified classes (categories)
  - Discriminate the input data within object population via the search for **invariant attributes** among members of the population

Classification Model, Features, and Decision Regions (cont.)

- The block diagram of the recognition and classification system

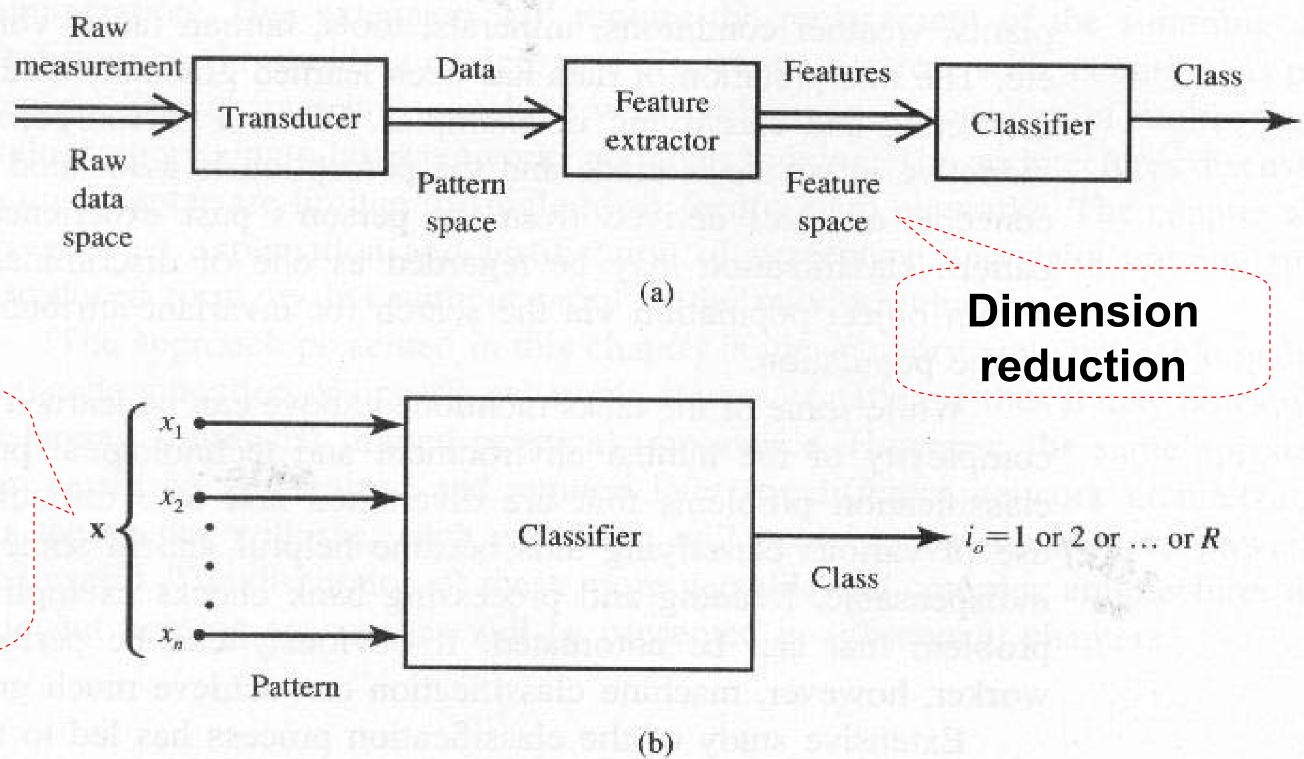


Figure 3.1 Recognition and classification system: (a) overall block diagram and (b) pattern classifier.

A neural network for classification and for feature extraction

Classification Model, Features, and Decision Regions (cont.)

- More about Feature Extraction
  - The compressed data from the input patterns while poses salient information
  - E.g.
    - Speech vowel sounds analyzed in 16-channel filterbanks can provide 16 spectral vectors, which can be further transformed into two dimensions
      - Tone height (high-low) and retraction (front-back)
    - Input patterns to be projected and reduced to lower dimensions

Classification Model, Features, and Decision Regions (cont.)

- More about Feature Extraction

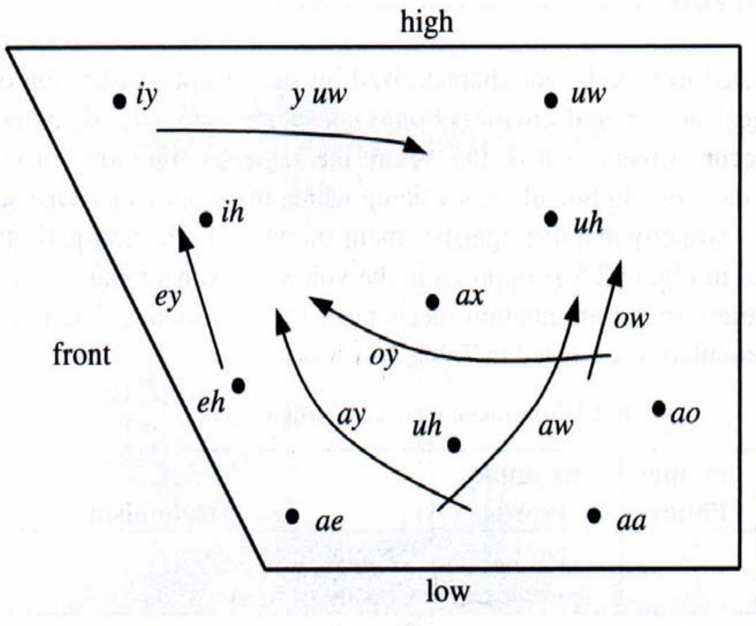
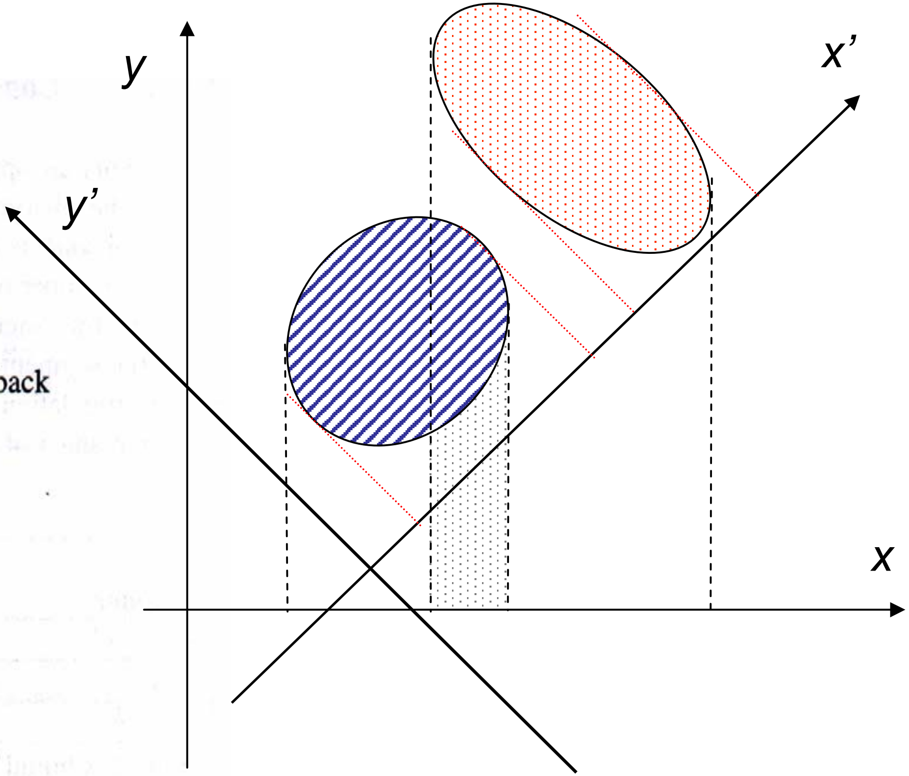


Figure 2.18 Relative tongue positions of English vowels [24].



Classification Model, Features, and Decision Regions (cont.)

- Two simple ways to generate the **pattern vectors** for cases of **spatial** and **temporal** objects to be classified

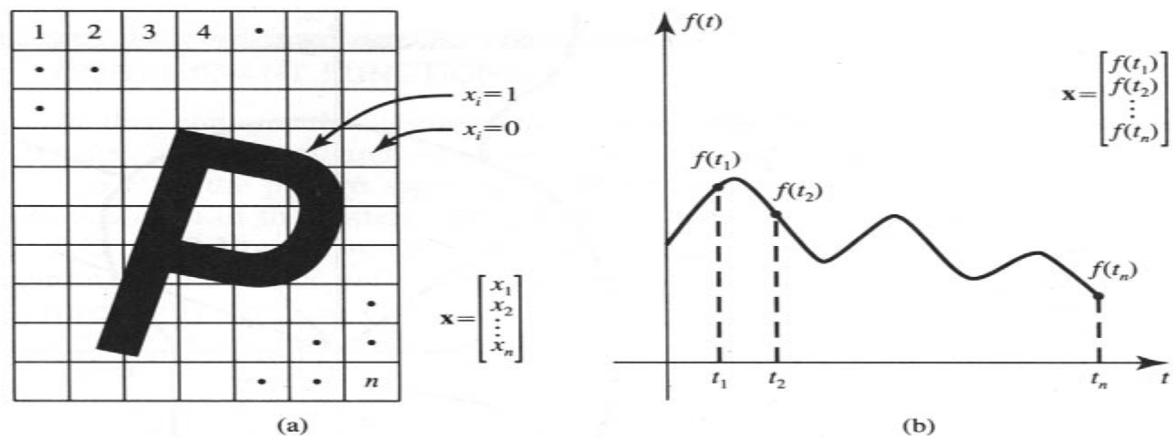


Figure 3.2 Two simple ways of coding patterns into pattern vectors: (a) spatial object and (b) temporal object (waveform).

- A pattern classifier maps input patterns (vectors) in E^n space into numbers (E^1) which specify the membership

$$j = i_0(\boldsymbol{x}),\ \ j = 1, 2, \ldots, R$$

Classification Model, Features, and Decision Regions (cont.)

- Classification described in geometric terms

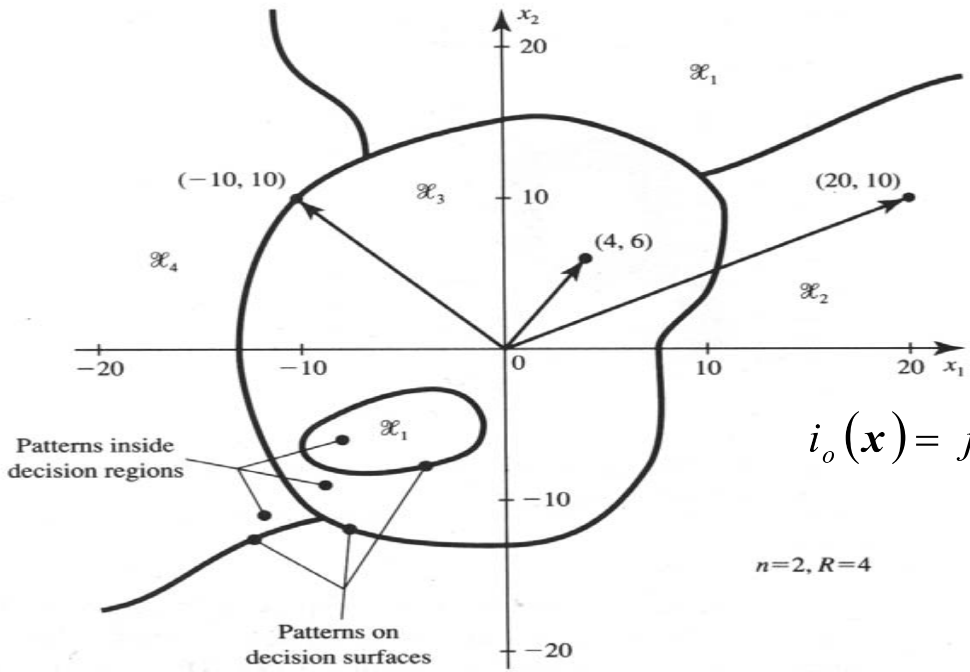


Figure 3.3 Decision regions example.

The decision surfaces here are curved lines

$$i_o(\boldsymbol{x}) = j, \text{ for all } \boldsymbol{x} \in \mathrm{X}_j, \quad j = 1, 2, \ldots, R$$

- **Decision regions**
- **Decision surfaces**: generally, the decision surfaces for n-dimensional patterns may be (n-1)-dimensional hyper-surfaces

Discriminant Functions

- Determine the membership in a category by the classifier based on the comparison of *R* **discriminant functions** $g_1(\boldsymbol{x}), g_2(\boldsymbol{x}),\ldots, g_R(\boldsymbol{x})$
  - When $\boldsymbol{x}$ is within the region $\boldsymbol{X}_k$ if $g_k(\boldsymbol{x})$ has the largest value $i_0(\boldsymbol{x}) = k \text{ if } g_k(\boldsymbol{x}) > g_j(\boldsymbol{x}) \text{ for } k,\ j = 1,\ 2,\ldots,\ R,\ k \neq j$

$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p, \ldots, \boldsymbol{x}_P$

P>>n
Assume the classifier
Has been designed

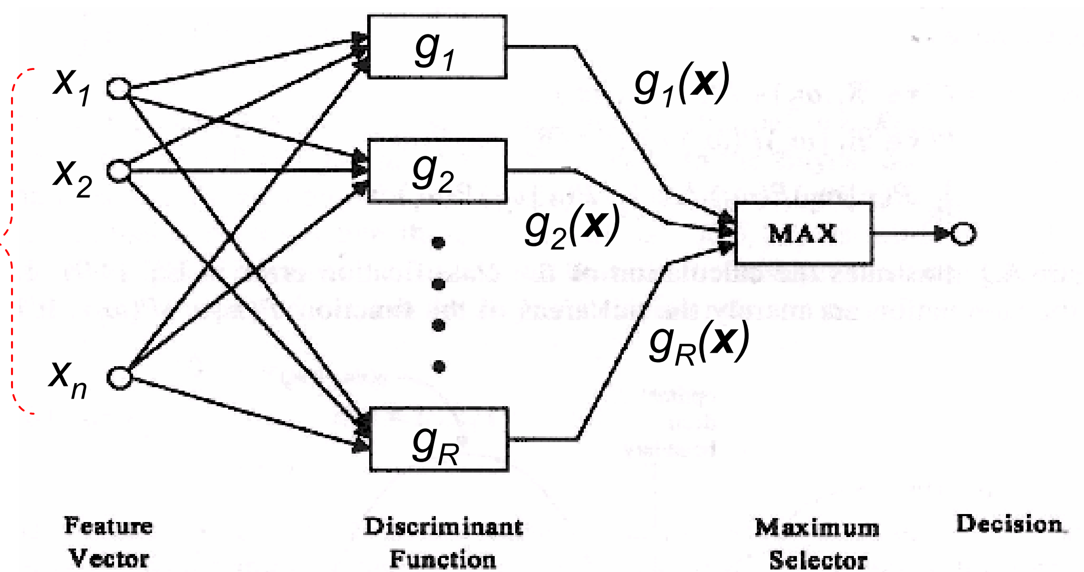


Figure 4.2 Block diagram of a classifier based on discriminant functions [22].

Discriminant Functions (cont.)

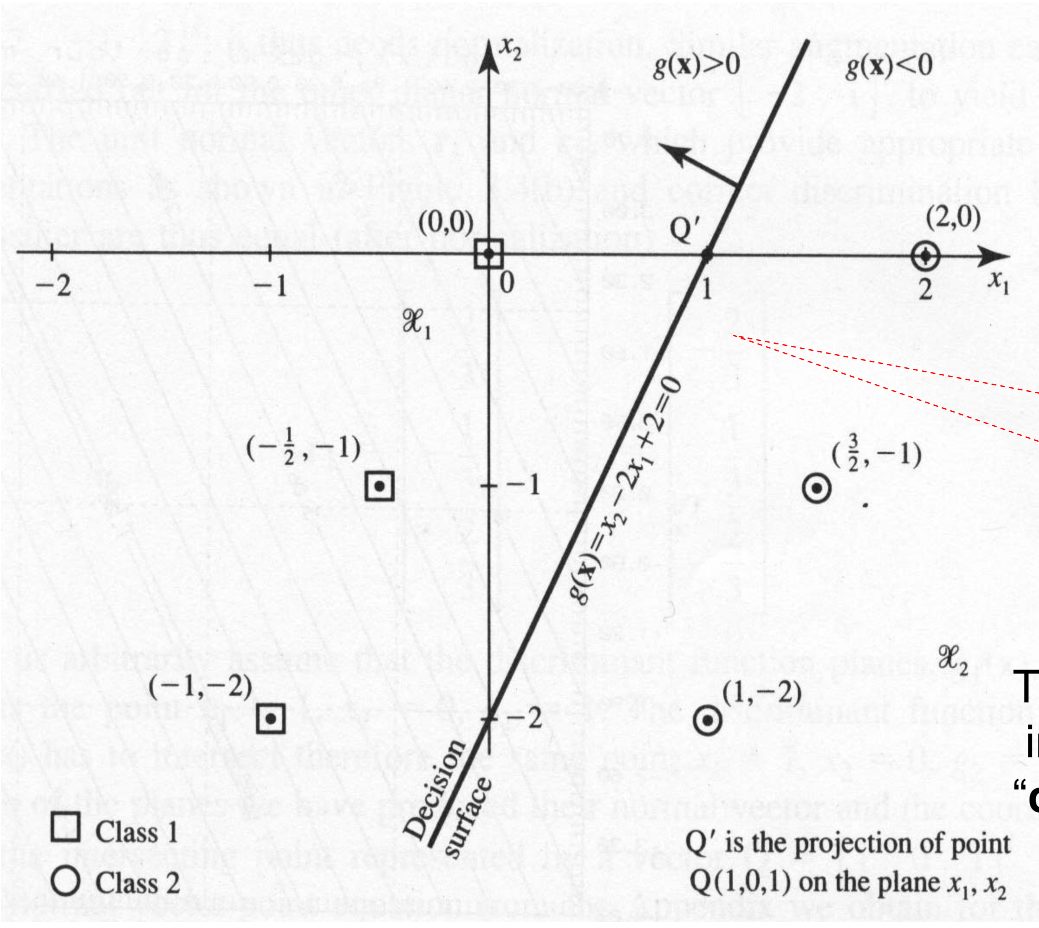
- Example 3.1

Decision surface Equation: $g(\boldsymbol{x}) = g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})$

$$= -2x_1 + x_2 + 2$$

$$g(\boldsymbol{x}) > 0 \; : \text{class1}$$

$$g(\boldsymbol{x}) < 0 \; : \text{class2}$$



The **decision surface** does not uniquely specify the discriminant functions

The classifier that classifies patterns into two classes or categories is called "**dichotomizer**"

"two" "cut"

Discriminant Functions (cont.)

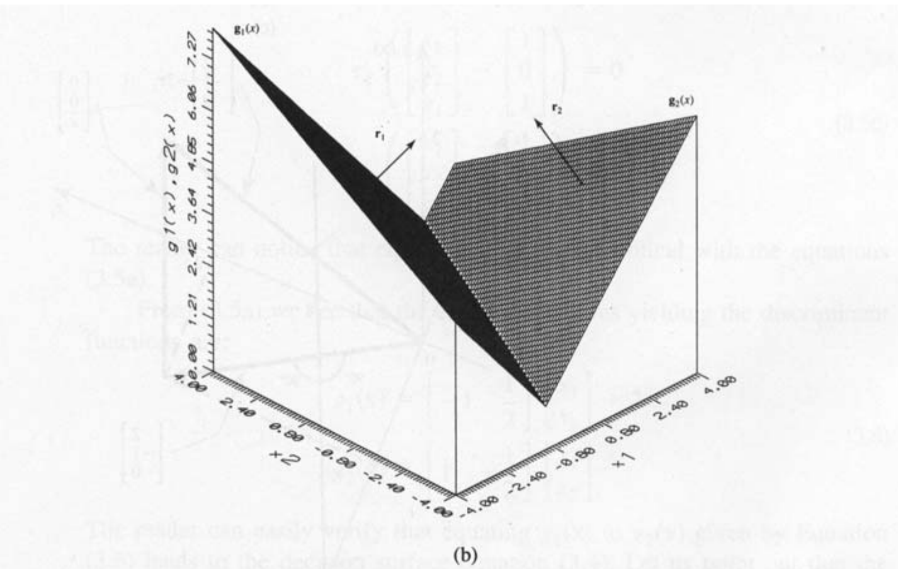


Figure 3.4a,b Illustration for Example 3.1: (a) pattern display and decision surface, (b) discriminant functions.

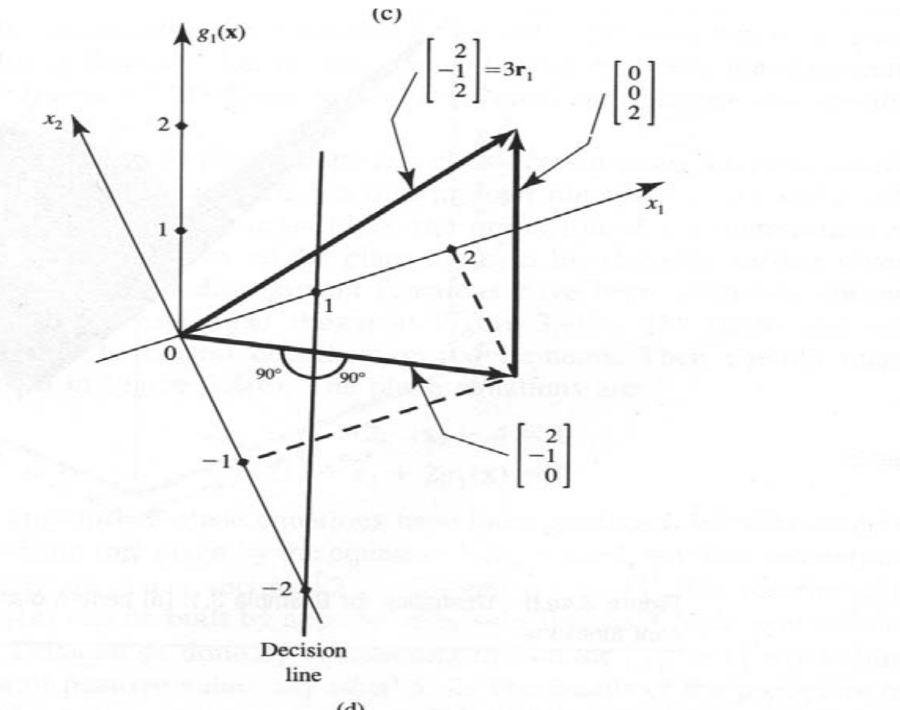


Figure 3.4c,d Illustration for Example 3.1 *(continued)*: (c) contour map of discriminant functions, and (d) construction of the normal vector for $g_1(\mathbf{x})$.

Discriminant Functions (cont.)

$(x-0, y+2, g_1-1)(2,-1,1)=0$
$2x-y-2+g_1-1=0$
$g_1=-2x+y+3$

$(x-0, y+2, g_2-1)(-2,1,1)=0$
$-2x+y+2+g_2-1=0$
$g_2=2x-y-1$

$g=g_1-g_2=0$
$-4x+2y+4=0$
$-2x+y+2=0$

Solution 1

$$g_1(\boldsymbol{x})=\begin{bmatrix}-2 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+3$$

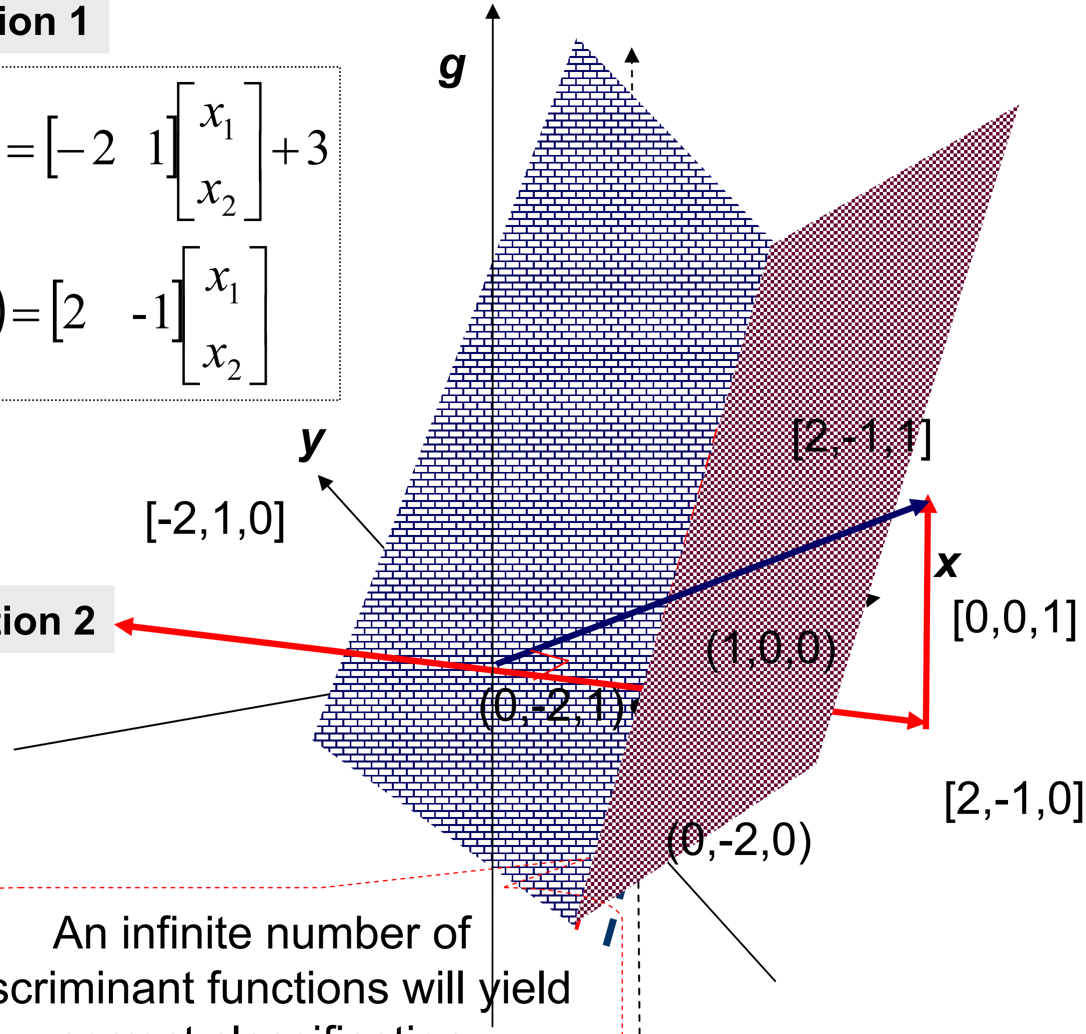
$$g_2(\boldsymbol{x})=\begin{bmatrix}2 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

$(x-0, y+2, g_1-1)(2,-1,2)=0$
$2x-y-2+2g_1-2=0$
$g_1=-x+1/2y+2$

$(x-0, y+2, g_2-1)(-2,1,2)=0$
$-2x+y+2+2g_2-2=0$
$g_2=x-1/2y$

$g=g_1-g_2=0$
$-2x+y+2=0$

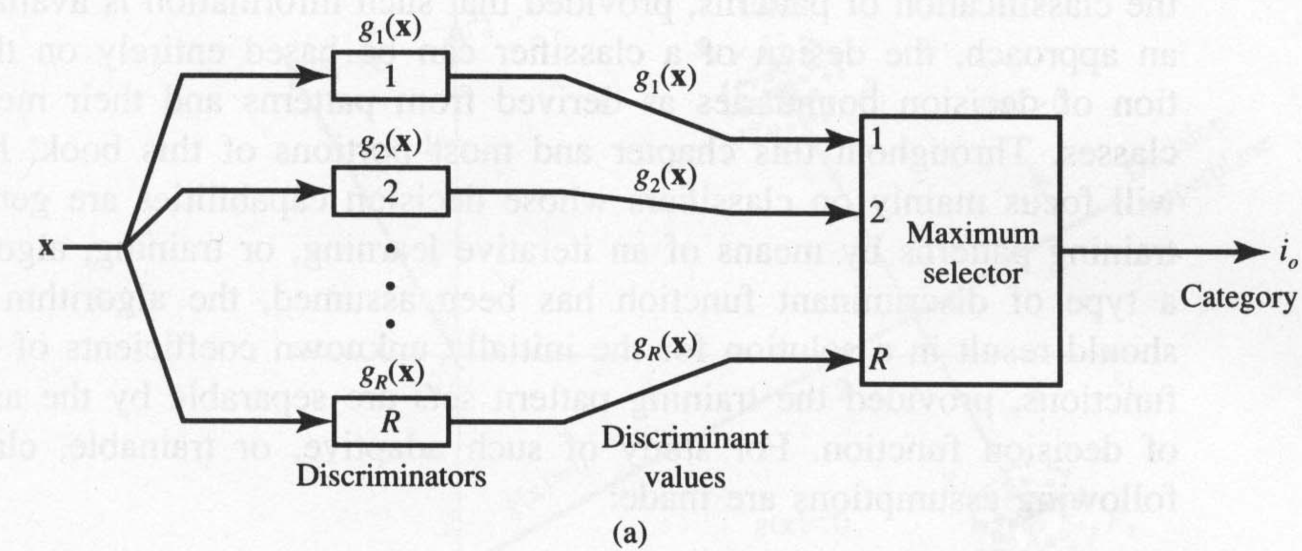
Solution 2



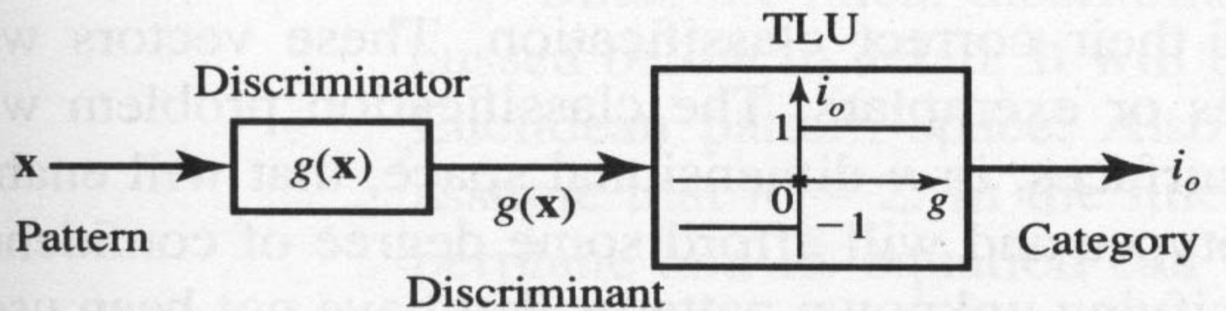
An infinite number of discriminant functions will yield correct classification

Discriminant Functions (cont.)

Multi-class



Two-class

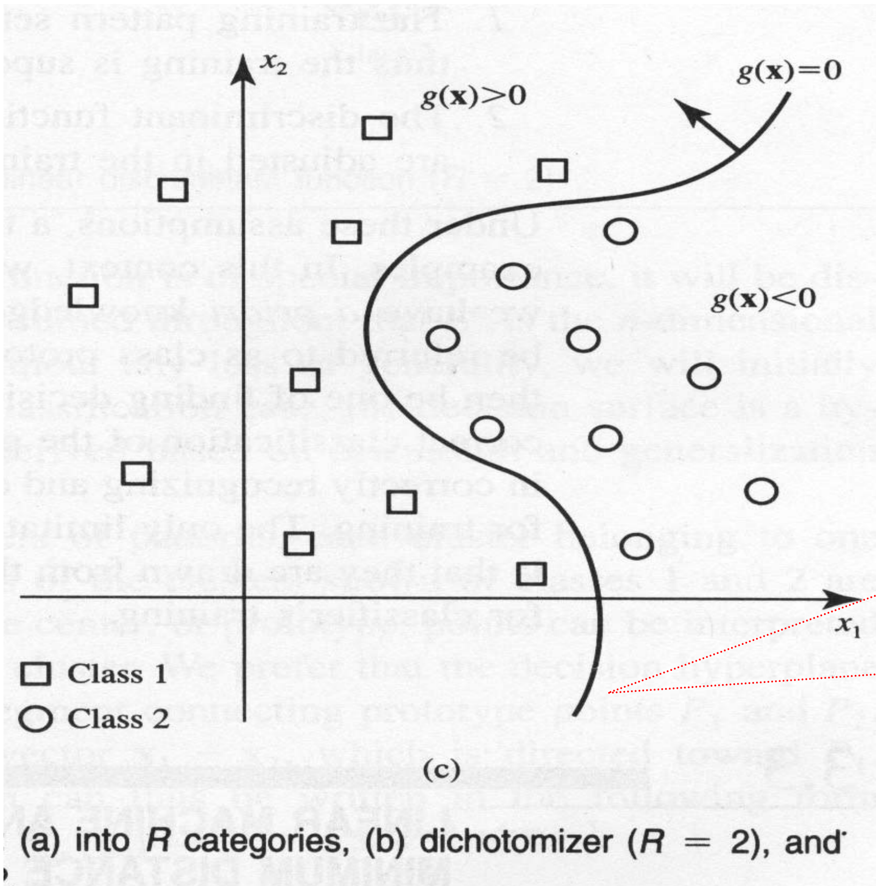


$$g(\boldsymbol{x}) = g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})$$

$$g(\boldsymbol{x}) > 0 \ : \text{class } 1$$
$$g(\boldsymbol{x}) < 0 \ : \text{class } 2$$

subtraction **Sign examination**

Discriminant Functions (cont.)



The design of discriminator for this case is not straightforward.
The discriminant functions may result as nonlinear functions of x_1 and x_2

Bayes' Decision Theory

- A decision-making based on both the posterior knowledge obtained from specific observation data and prior knowledge of the categories
  - Prior class probabilities $P(\omega_i), \ \forall \text{ class } i$
  - Class-conditioned probabilities $P(x|\omega_i), \ \forall \text{ class } i$

$$k = \arg\max_i P(\omega_i|x) = \arg\max_i \frac{P(x|\omega_i)P(\omega_i)}{P(x)} = \arg\max_i \frac{P(x|\omega_i)P(\omega_i)}{\sum_{j=1} P(x|\omega_j)P(\omega_j)}$$

$$k = \arg\max_i P(\omega_i|x) = \arg\max_i P(x|\omega_i)P(\omega_i)$$

Bayes' Decision Theory (cont.)

- Bayes' decision rule designed to minimize the overall risk involved in making decision
  - The expected loss (conditional risk) when making decision δ_i

$$R(\delta_i|x) = \sum_j l(\delta_i|\omega_j, x)P(\omega_j|x), \quad \text{where} \quad l(\delta_i|\omega_j, x) = \begin{cases} 0, & i = j \\ 1, & i \neq j \end{cases}$$

$$= \sum_{j \neq i} P(\omega_j|x)$$

$$= 1 - P(\omega_i|x)$$

    - The overall risk (Bayes' risk)

$$R = \int_{-\infty}^{\infty} R(\delta(x)|x)p(x)dx, \; \delta(x): \text{the selected decision for a sample } x$$

  - Minimize the overall risk (classification error) by computing the conditional risks and select the decision δ_i for which the conditional risk $R(\delta_i|x)$ is minimum, i.e., $P(\omega_i|x)$ is maximum (minimum-error-rate decision rule)

Bayes' Decision Theory (cont.)

- Two-class pattern classification

$$g_1(x) = P(\omega_1|x) \cong P(x|\omega_1)P(\omega_1),\ g_2(x) = P(\omega_2|x) \cong P(x|\omega_2)P(\omega_2)$$

Bayes' Classifier

$$P(x|\omega_1)P(\omega_1) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} P(x|\omega_2)P(\omega_2) \iff$$

Likelihood ratio or log-likelihood ratio:

$$l(x) = \frac{P(x|\omega_1)}{P(x|\omega_2)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \frac{P(\omega_2)}{P(\omega_1)}$$

$$\log l(x) = \log P(x|\omega_1) - \log P(x|\omega_2) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \log P(\omega_2) - \log P(\omega_1)$$

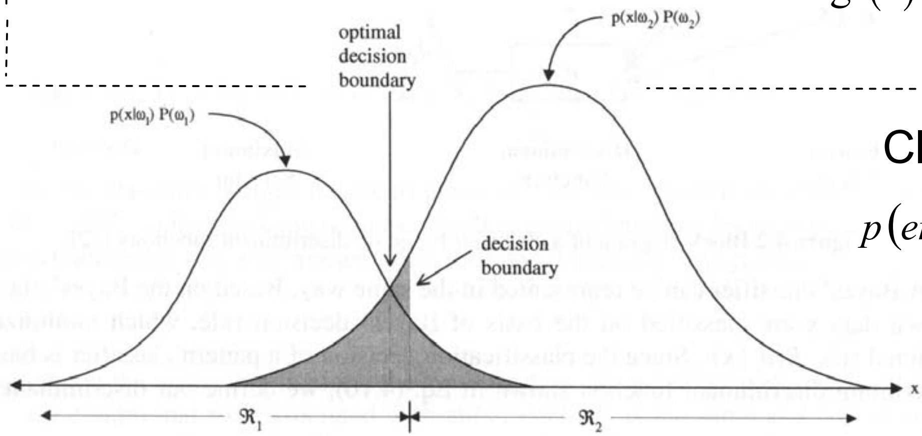


Figure 4.1 Calculation of the likelihood of classification error [22]. The shaded area represents the integral value in Eq. (4.9).

Classification error:

$$\begin{aligned} p(error) &= P(x \in R_1, \omega_2) + P(x \in R_2, \omega_1) \\ &= P(x \in R_1|\omega_2)P(\omega_2) + P(x \in R_2|\omega_1)P(\omega_1) \\ &= \int_{R_1} P(x|\omega_2)P(\omega_2)dx + \int_{R_2} P(x|\omega_1)P(\omega_1)dx \end{aligned}$$

Bayes' Decision Theory (cont.)

- When the environment is multivariate Gaussian, the Bayes' classifier reduces to a linear classifier
  - The same form taken by the perceptron
  - But the linear nature of the perceptron is not contingent on the assumption of Gaussianity

$$P(\boldsymbol{x}|\omega) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

$$\text{Class } \omega_1 : E[\boldsymbol{X}] = \boldsymbol{\mu}_1$$

$$E\left[(\boldsymbol{X}-\boldsymbol{\mu}_1)(\boldsymbol{X}-\boldsymbol{\mu}_1)^t\right] = \boldsymbol{\Sigma}$$

$$\text{Class } \omega_2 : E[\boldsymbol{X}] = \boldsymbol{\mu}_2$$

$$E\left[(\boldsymbol{X}-\boldsymbol{\mu}_2)(\boldsymbol{X}-\boldsymbol{\mu}_2)^t\right] = \boldsymbol{\Sigma}$$

$$P(\omega_1) = P(\omega_2) = \frac{1}{2}$$

Assumptions

Bayes' Decision Theory (cont.)

- When the environment is Gaussian, the Bayes' classifier reduces to a linear classifier (cont.)

$$
\begin{aligned}
\log l(\boldsymbol{x}) &= \log P(\boldsymbol{x}|\omega_1) - \log P(\boldsymbol{x}|\omega_2) \\
&= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) \\
&= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^t \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \frac{1}{2}\left(\boldsymbol{\mu}_2^t \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^t \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1\right) \\
&= \boldsymbol{w}\boldsymbol{x} + b
\end{aligned}
$$

$$
\therefore \log l(\boldsymbol{x}) = \boldsymbol{w}\boldsymbol{x} + b \underset{\omega_2}{\overset{\omega_1}{\gtrless}} 0
$$

Bayes' Decision Theory (cont.)

- Multi-class pattern classification

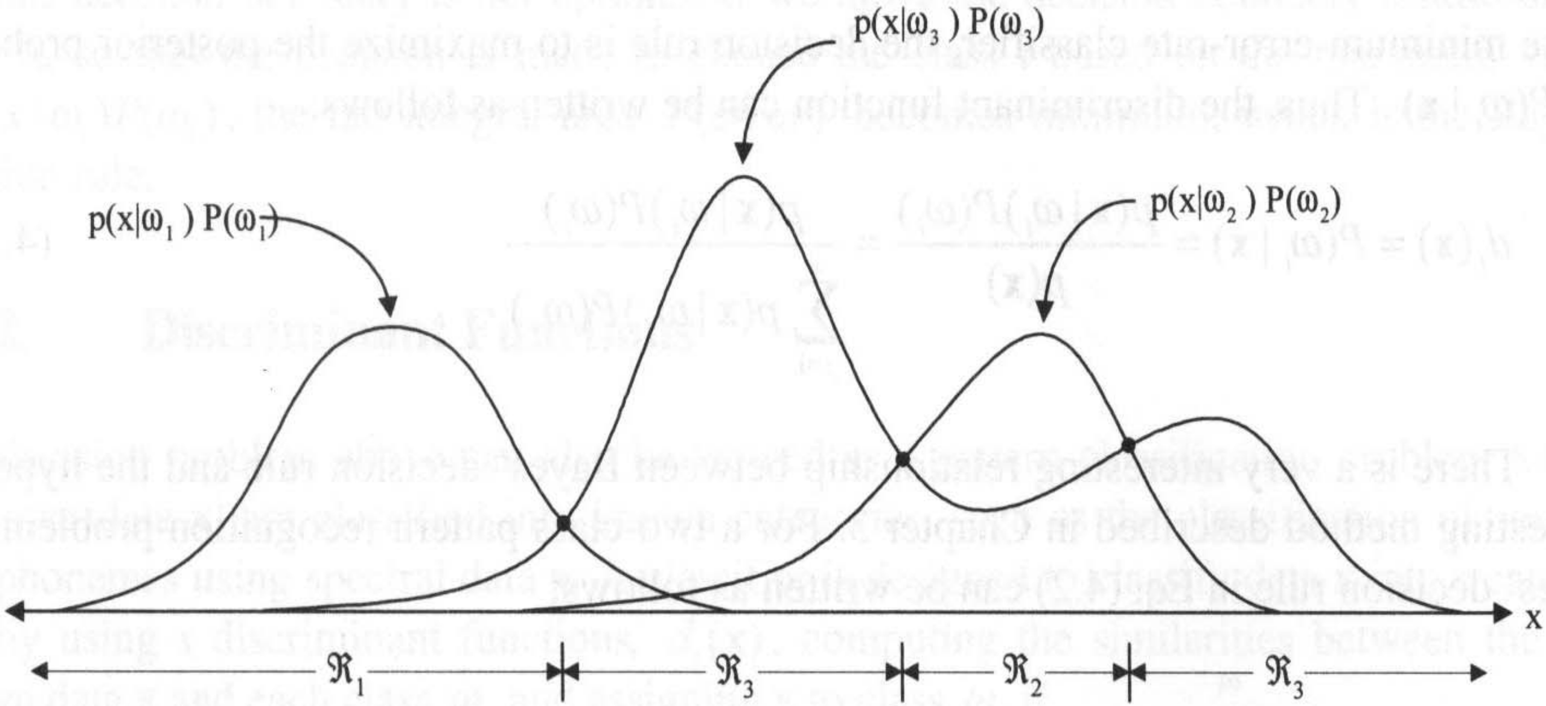


Figure 4.3 An example of decision boundaries and regions. For simplicity, we use scalar variable x instead of a multi-dimensional vector [22].

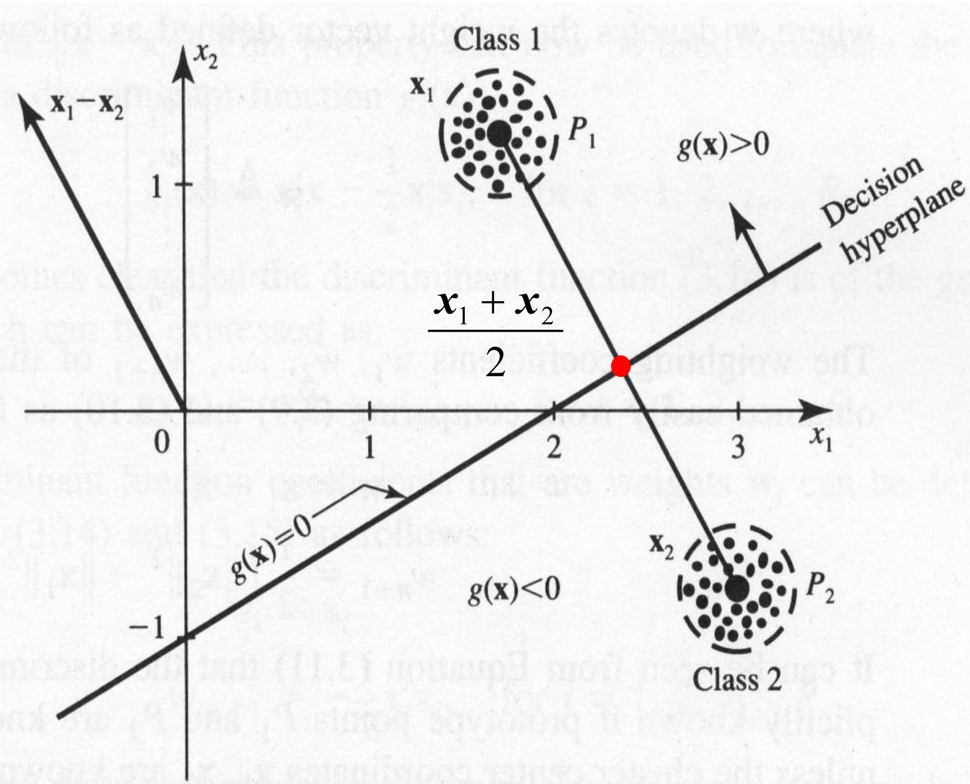
Linear Machine and Minimum Distance Classification

- Find the linear-form discriminant function for two-class classification when the class prototypes are known

- **Example 3.1**: Select the decision hyperplane that contains the midpoint of the line segment connecting center point of two classes

Linear Machine and Minimum Distance Classification (cont.)

The dichotomizer's discriminant function $g(\boldsymbol{x})$:



$$(\boldsymbol{x}_1 - \boldsymbol{x}_2)^t (\boldsymbol{x} - \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2}{2}) = 0$$

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

$$\text{Taken as } \begin{bmatrix} \boldsymbol{w} \\ w_{n+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = 0, \text{ where}$$

Augmented input pattern

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

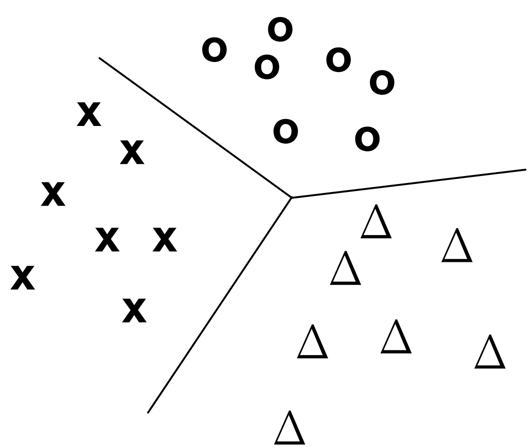
$$w_{n+1} = \frac{1}{2}\left(\|\boldsymbol{x}_2\|^2 - \|\boldsymbol{x}_1\|^2\right)$$

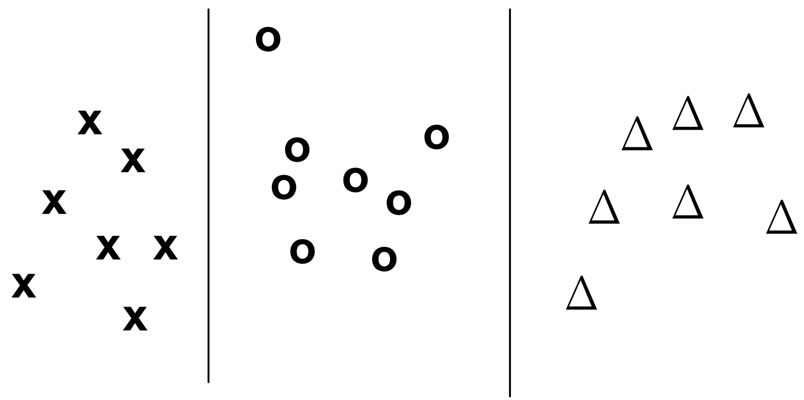
It is a simple minimum-distance classifier.

Linear Machine and Minimum Distance Classification (cont.)

- The linear-form discriminant functions for multi-class classification
  - There are up to R(R-1)/2 decision hyperplanes for *R* pairwise separable classes

Some classes may not be contiguous





Linear Machine and Minimum Distance Classification (cont.)

- Linear machine or minimum-distance classifier
  - Assume the class prototypes are known for all classes
    - Euclidean distance between input pattern ***x*** and the center of class *i*, $\boldsymbol{x}_i$: $\|\boldsymbol{x}-\boldsymbol{x}_i\| = \sqrt{(\boldsymbol{x}-\boldsymbol{x}_i)^t(\boldsymbol{x}-\boldsymbol{x}_i)}$
    - Minimizing $\|\boldsymbol{x}-\boldsymbol{x}_i\|^2 = \boldsymbol{x}^t\boldsymbol{x} - 2\boldsymbol{x}_i^t\boldsymbol{x} + \boldsymbol{x}_i^t\boldsymbol{x}_i$ is equal to

      maximizing $\boldsymbol{x}_i^t\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}_i^t\boldsymbol{x}_i$

      The same for all classes
  - Set the discriminant function for each class *i* to be:

$$g_i(\boldsymbol{x}) = \boldsymbol{x}_i^t\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}_i^t\boldsymbol{x}_i \Longrightarrow g_i(\boldsymbol{x}) = \boldsymbol{w}_i^t\boldsymbol{y}$$

$$g_i(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{w}_i \\ w_{i,n+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}, \text{ where } \begin{array}{l} \boldsymbol{w}_i = \boldsymbol{x}_i \\ w_{i,n+1} = -\frac{1}{2}(\boldsymbol{x}_i^t\boldsymbol{x}_i) \end{array}$$

Linear Machine and Minimum Distance Classification (cont.)

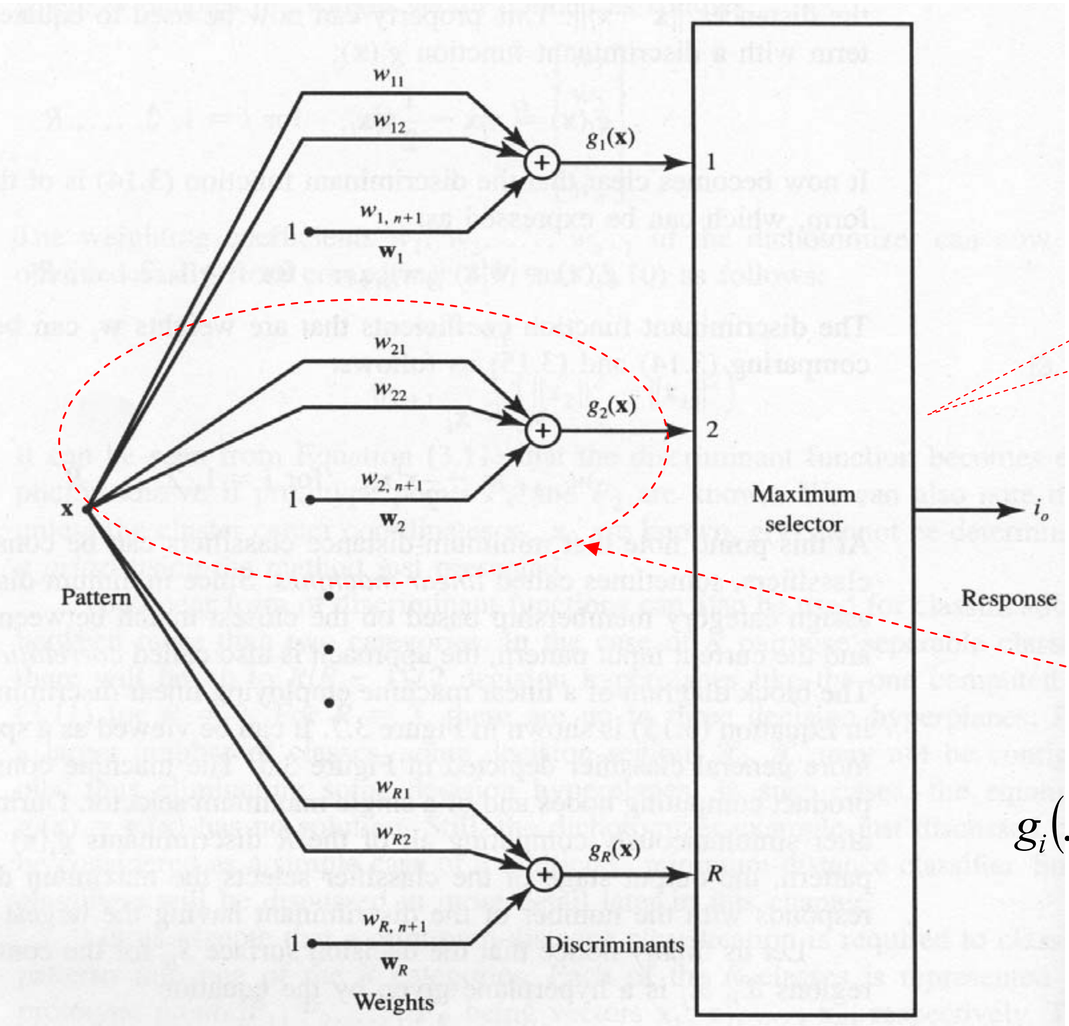


Figure 3.7 A linear classifier.

This approach is also called **correlation** classification

An 1 as the *n+1*'th component of the input pattern

$$g_i(\boldsymbol{x}) = \boldsymbol{x}_i^t \boldsymbol{x} - \frac{1}{2}\boldsymbol{x}_i^t \boldsymbol{x}_i \Rightarrow g_i(\boldsymbol{x}) = \boldsymbol{w}_i^t \boldsymbol{y}$$

Linear Machine and Minimum Distance Classification (cont.)

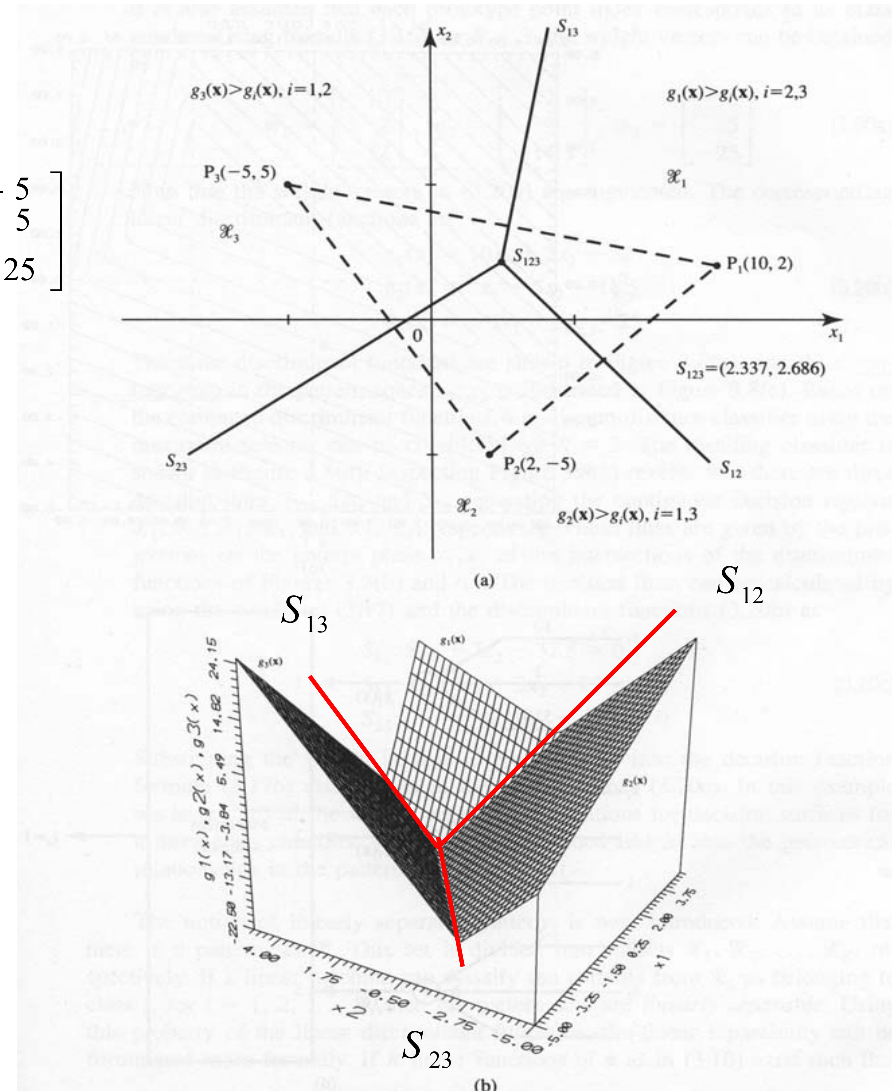
- Example 3.2

$$\boldsymbol{w}_1 = \begin{bmatrix} 10 \\ 2 \\ -52 \end{bmatrix},\ \boldsymbol{w}_2 = \begin{bmatrix} 2 \\ -5 \\ -14.5 \end{bmatrix},\ \boldsymbol{w}_3 = \begin{bmatrix} -5 \\ 5 \\ -25 \end{bmatrix}$$

$$g_1(\boldsymbol{x}) = 10x_1 + 2x_2 - 52$$
$$g_2(\boldsymbol{x}) = 2x_1 - 5x_2 - 14.5$$
$$g_3(\boldsymbol{x}) = -5x_1 + 5x_2 - 25$$

$$S_{12}: 8x_1 + 7x_2 - 37.5 = 0$$
$$S_{13}: -15x_1 + 3x_2 + 27 = 0$$
$$S_{23}: -7x_1 + 10x_2 - 10.5 = 0$$

$$g_i(\boldsymbol{x}) = \boldsymbol{x}_i^t \boldsymbol{x} - \frac{1}{2}\boldsymbol{x}_i^t \boldsymbol{x}_i$$



Linear Machine and Minimum Distance Classification (cont.)

- If R linear discriminant functions exist for a set of patterns such that

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in \text{Class } i,$$
$$i = 1, 2, \ldots, R, j = 1, 2, \ldots, R, \ i \neq j$$

  - The classes are linearly separable

Linear Machine and Minimum Distance Classification (cont.)

P3.3 For the minimum-distance (linear) dichotomizer, the weight and augmented pattern vectors are

$$\mathbf{w} = \begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

(a) Find the equation of the decision surface in the pattern space.

(b) Find the equation of the decision surface in the augmented pattern space.

(c) Compute the new solution weight vector if the two class prototype points are

$$\mathbf{x}_1 = \begin{bmatrix} 2 & 5 \end{bmatrix}^t \text{ and } \mathbf{x}_2 = \begin{bmatrix} -1 & -3 \end{bmatrix}^t.$$

(d) Sketch the decision surfaces for each case in parts (a), (b), and (c).

Linear Machine and Minimum Distance Classification (cont.)

(a) $2x_1-x_2+2=0$, decision surface is a line

(b) $2x_1-x_2+2=0$, decision surface is a plane

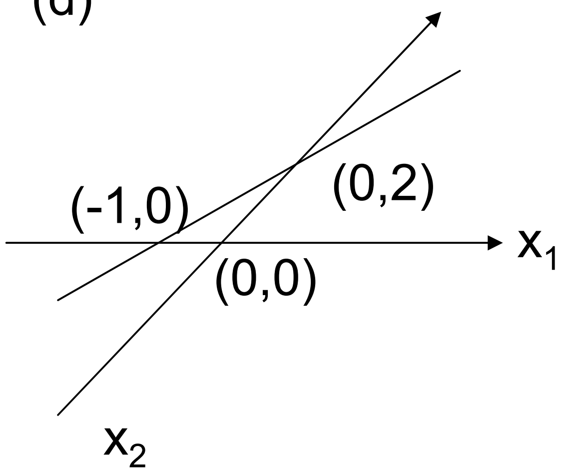
(c) $\mathbf{x}_1=[2,5]$, $\mathbf{x}_2=[-1,-3]$

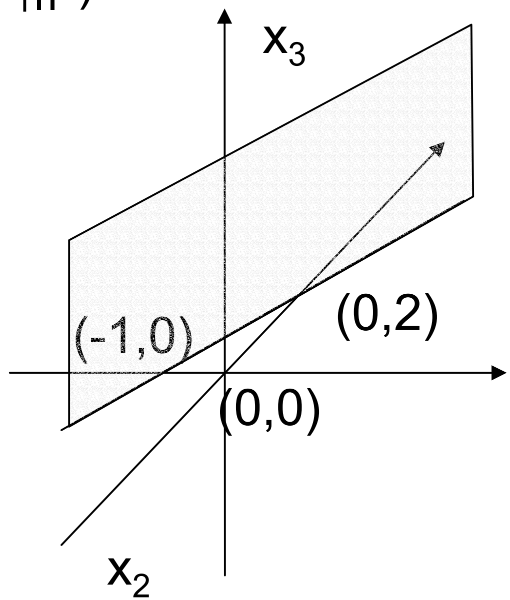
=>The decision surface for minimum distance classifier

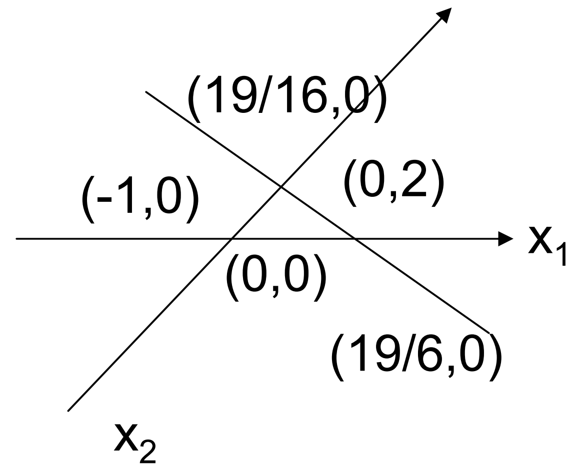
$$(\mathbf{x}_1-\mathbf{x}_2)^t\,\mathbf{x}+1/2\,(||\mathbf{x}_2||^2-||\mathbf{x}_1||^2)^t=0$$

$$3x_1+8x_2-19/2=0$$

(d)







Linear Machine and Minimum Distance Classification (cont.)

- Examples 3.1 and 3.2 have shown that the coefficients (weights) of the linear discriminant functions can be determined if the a priori information about the sets of patterns and their class membership is known

Linear Machine and Minimum Distance Classification (cont.)

- The example of linearly non-separable patterns

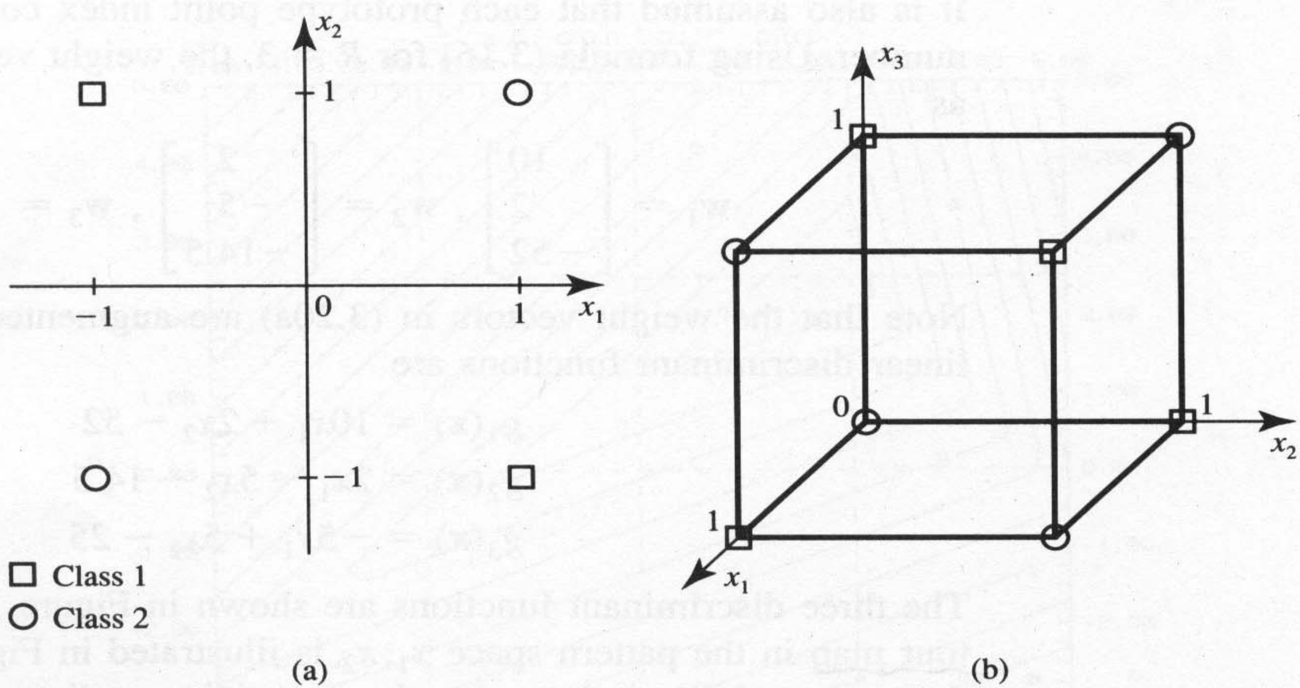
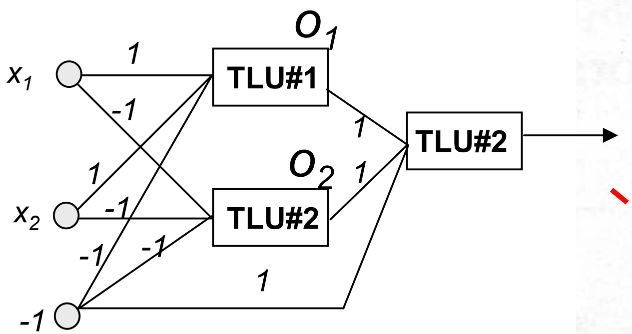
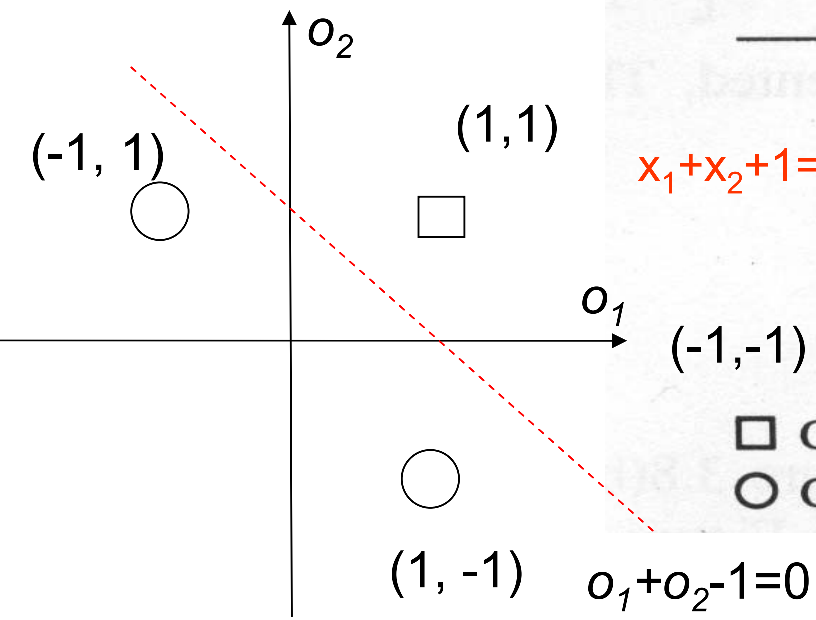
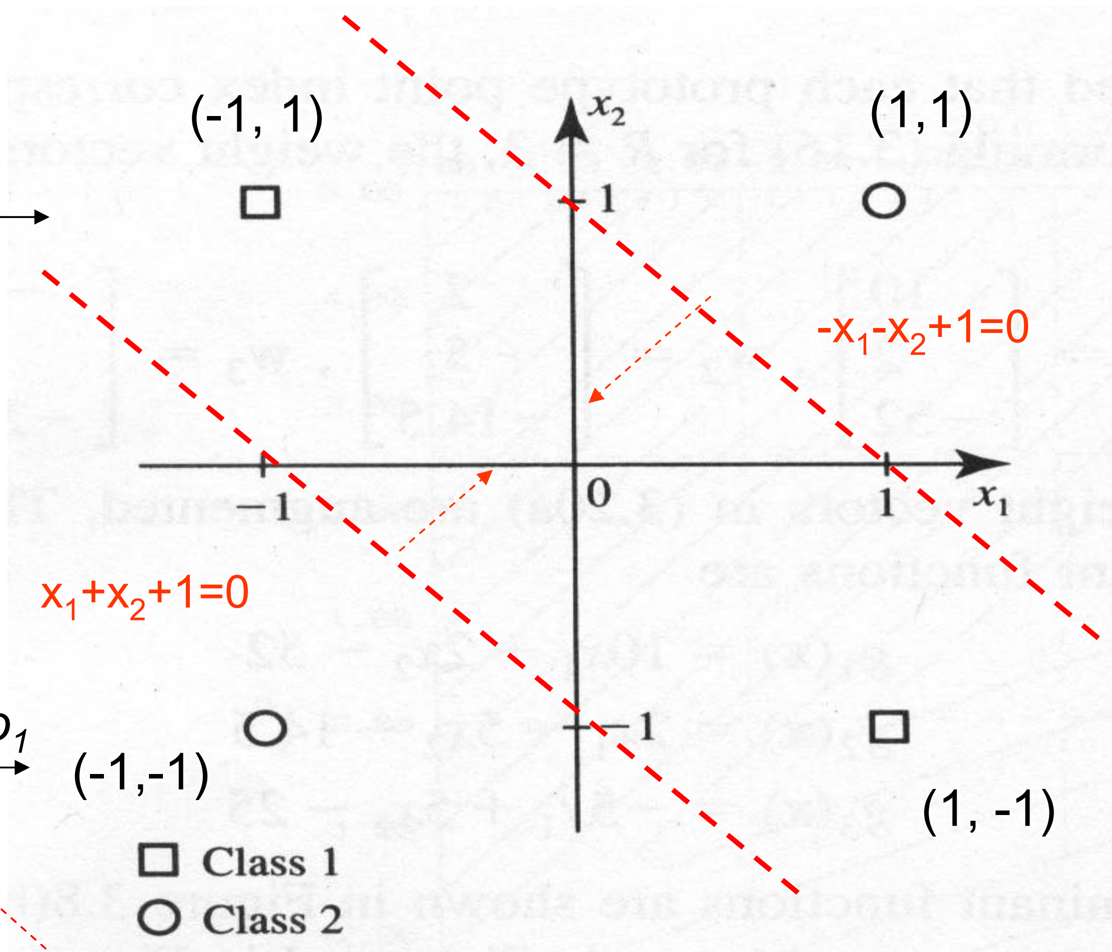


Figure 3.9 Example of parity function resulting in linearly nonseparable patterns ($R = 2$): (a) $x_1 \oplus x_2$ and (b) $x_1 \oplus x_2 \oplus x_3$.

Linear Machine and Minimum Distance Classification (cont.)







Discrete Perceptron Training Algorithm - Geometrical Representations

- Examine the neural network classifiers that derive/training their weights based on the **error-correction scheme**

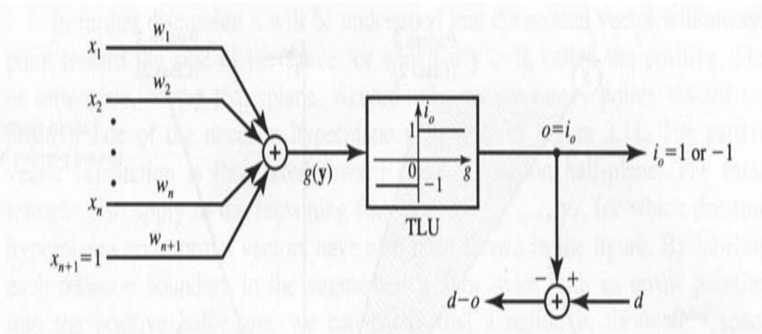


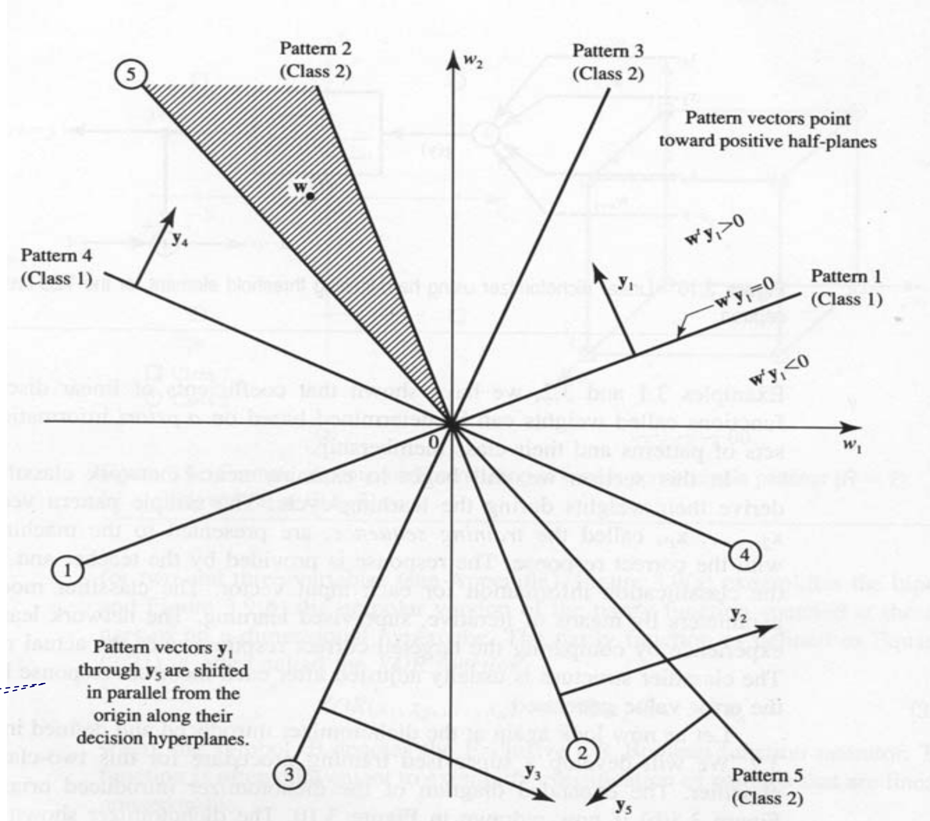
Figure 3.10 Linear dichotomizer using hard-limiting threshold element, or the TLU-based perceptron.

$$g(y) = w^t y$$

Augmented input pattern

Class 1: $w^t y > 0$

Class 2: $w^t y < 0$



Vector Representations in the **Weight Space**

Figure 3.11 Decision hyperplane in augmented weight space for a five pattern set from two classes.

Discrete Perceptron Training Algorithm - Geometrical Representations (count.)

- Devise an analytic approach based on the geometrical representations
  - E.g. the decision surface for the training pattern $\boldsymbol{y}_1$

$\boldsymbol{y}_1$ in Class 1

Weight Space

$\boldsymbol{y}_1$ in Class 2

Weight Space

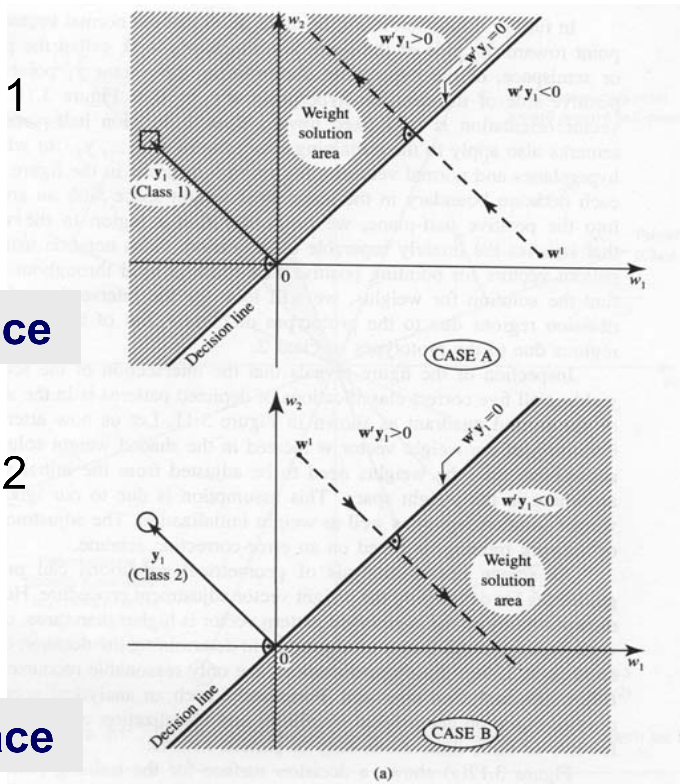


Figure 3.12a Weight adjustments of learning dichotomizer: (a) steepest descent

$$\nabla_w \left(\boldsymbol{w}^t \boldsymbol{y}_1 \right) = \boldsymbol{y}_1$$

Gradient (the direction of steep increase)

If $\boldsymbol{y}_1$ in Class 1:

$$\boldsymbol{w}' = \boldsymbol{w}^1 + c\boldsymbol{y}_1$$

If $\boldsymbol{y}_1$ in Class 2:

$$\boldsymbol{w}' = \boldsymbol{w}^1 - c\boldsymbol{y}_1$$

c controls the size of adjustment

c (>0) is the correction increment (*is two times of the learning constant introduced before*)

Discrete Perceptron Training Algorithm - Geometrical Representations (count.)

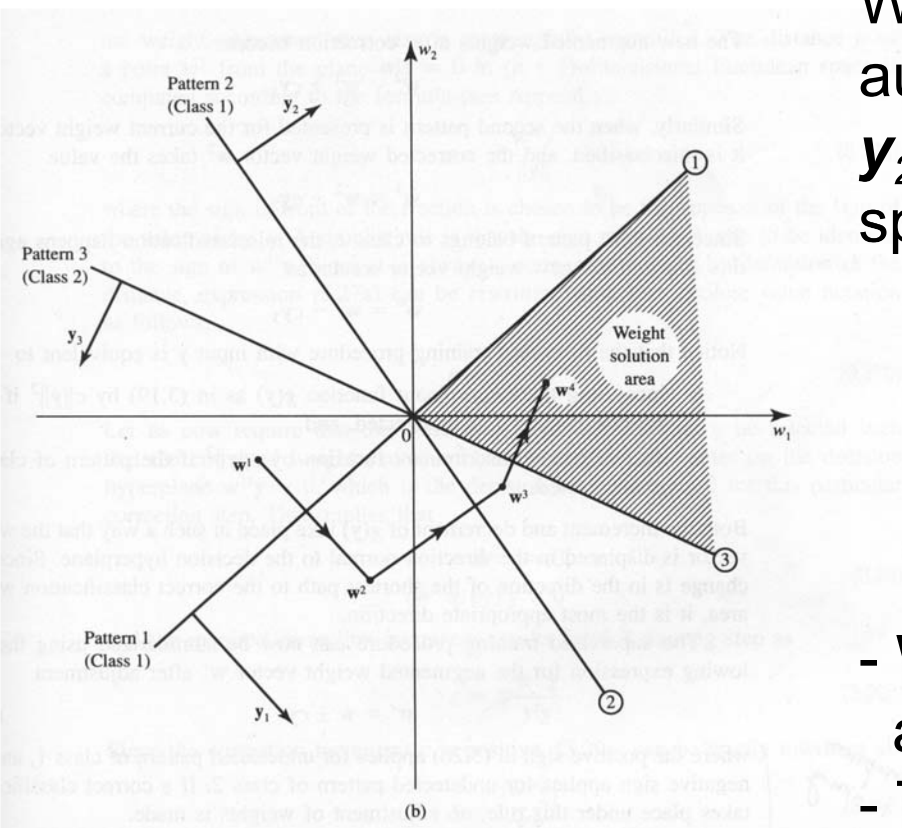


Figure 3.12b Weight adjustments of learning dichotomizer *(continued)*: (b) example.

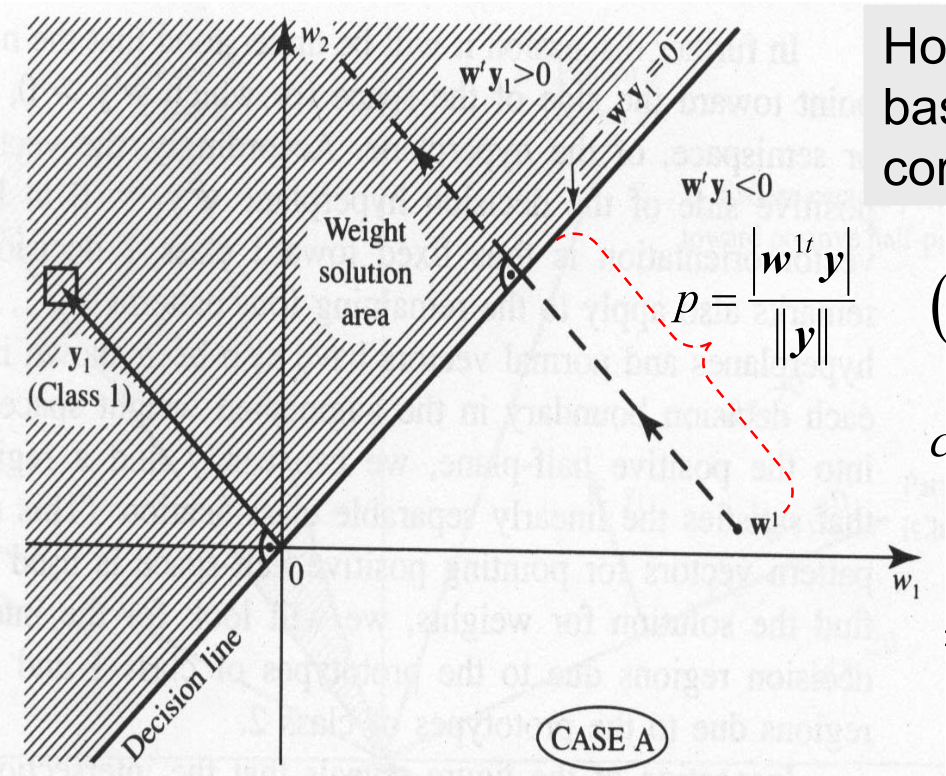
Weight adjustments of three augmented training pattern $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3$, shown in the weight space

$$\boldsymbol{y}_1 \in C_1$$
$$\boldsymbol{y}_2 \in C_1$$
$$\boldsymbol{y}_3 \in C_2$$

- Weights in the shaded region are the solutions
- The three lines labeled are fixed during training

Discrete Perceptron Training Algorithm - Geometrical Representations (count.)

- More about the correction increment *c*
  - If it is not merely a constant, but related to the current training pattern



How to select the correction increment based on the dislocates of $\boldsymbol{w}^1$ and the corrected weight vector $\boldsymbol{w}$

$$\left(\boldsymbol{w}^1 \pm c\boldsymbol{y}\right)^t \boldsymbol{y} = 0$$

$$c = \mp \frac{\boldsymbol{w}^{1t}\boldsymbol{y}}{\boldsymbol{y}^t\boldsymbol{y}} = \frac{|\boldsymbol{w}^{1t}\boldsymbol{y}|}{\|\boldsymbol{y}\|^2}, \text{ because } \quad c > 0$$

$$\Rightarrow c\boldsymbol{y} = \frac{|\boldsymbol{w}^{1t}\boldsymbol{y}|}{\|\boldsymbol{y}\|^2}\boldsymbol{y}$$

Discrete Perceptron Training Algorithm - Geometrical Representations (count.)

- For **fixed correction rule** with *c*=constant, the correction of weights is always the same fixed portion of the current training vector
  - The weight can be initialized at any value

$$\boldsymbol{w}' = \boldsymbol{w} \pm c\boldsymbol{y} \quad \text{or} \quad \begin{aligned} \boldsymbol{w}' &= \boldsymbol{w} + \Delta \boldsymbol{w} \\ \Delta \boldsymbol{w} &= c\left[d - \operatorname{sgn}\left(\boldsymbol{w}^t \boldsymbol{y}\right)\right]\boldsymbol{y} \end{aligned}$$

- For **dynamic correction rule** with *c* dependent on the distance from the weight (i.e. the weight vector) to the decision surface in the weight space

$$\Rightarrow c\boldsymbol{y} = \frac{\left|\boldsymbol{w}^{1t} \boldsymbol{y}\right|}{\|\boldsymbol{y}\|^2} \boldsymbol{y}$$

  - The initial weight should be different from ***0***

Discrete Perceptron Training Algorithm - Geometrical Representations (count.)

- Dynamic correction rule with *c* dependent on the distance from the weight

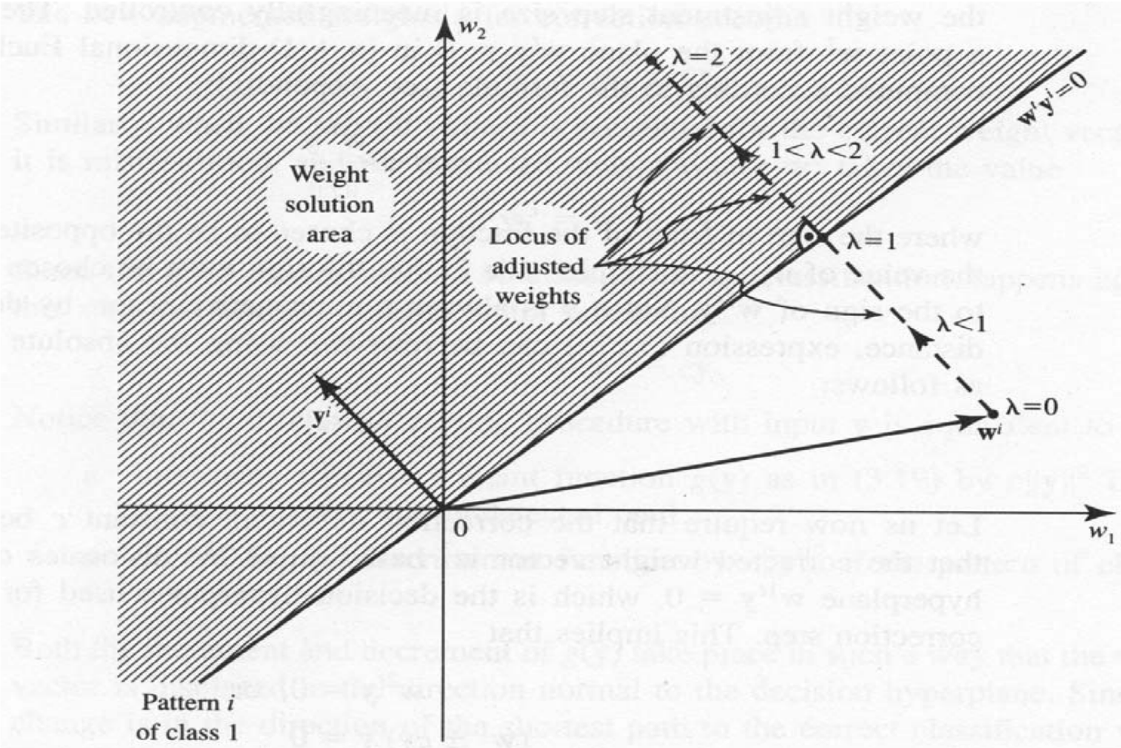


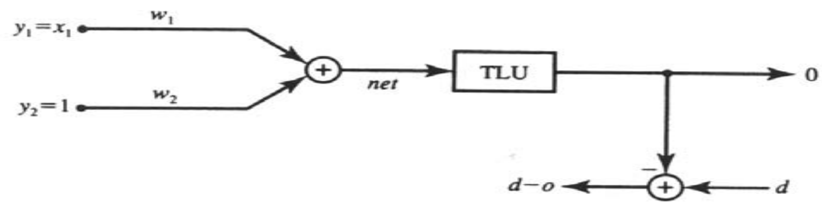
Figure 3.13 Illustration of correction increment value, i'th step.

$$c = \lambda \frac{\left|\boldsymbol{w}^{1t}\boldsymbol{y}\right|}{\left\|\boldsymbol{y}\right\|^2}$$

$$c\boldsymbol{y} = \lambda \frac{\left|\boldsymbol{w}^{1t}\boldsymbol{y}\right|}{\left\|\boldsymbol{y}\right\|}\frac{\boldsymbol{y}}{\left\|\boldsymbol{y}\right\|}$$

Discrete Perceptron Training Algorithm - Geometrical Representations (count.)

- Example 3.3



$$\boldsymbol{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \boldsymbol{y}_2 = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \quad \begin{matrix} \boldsymbol{y}_1 \in C_1 \\ \boldsymbol{y}_2 \in C_2 \end{matrix}$$

$$\boldsymbol{y}_3 = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \quad \boldsymbol{y}_4 = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \quad \begin{matrix} \boldsymbol{y}_3 \in C_1 \\ \boldsymbol{y}_4 \in C_2 \end{matrix}$$

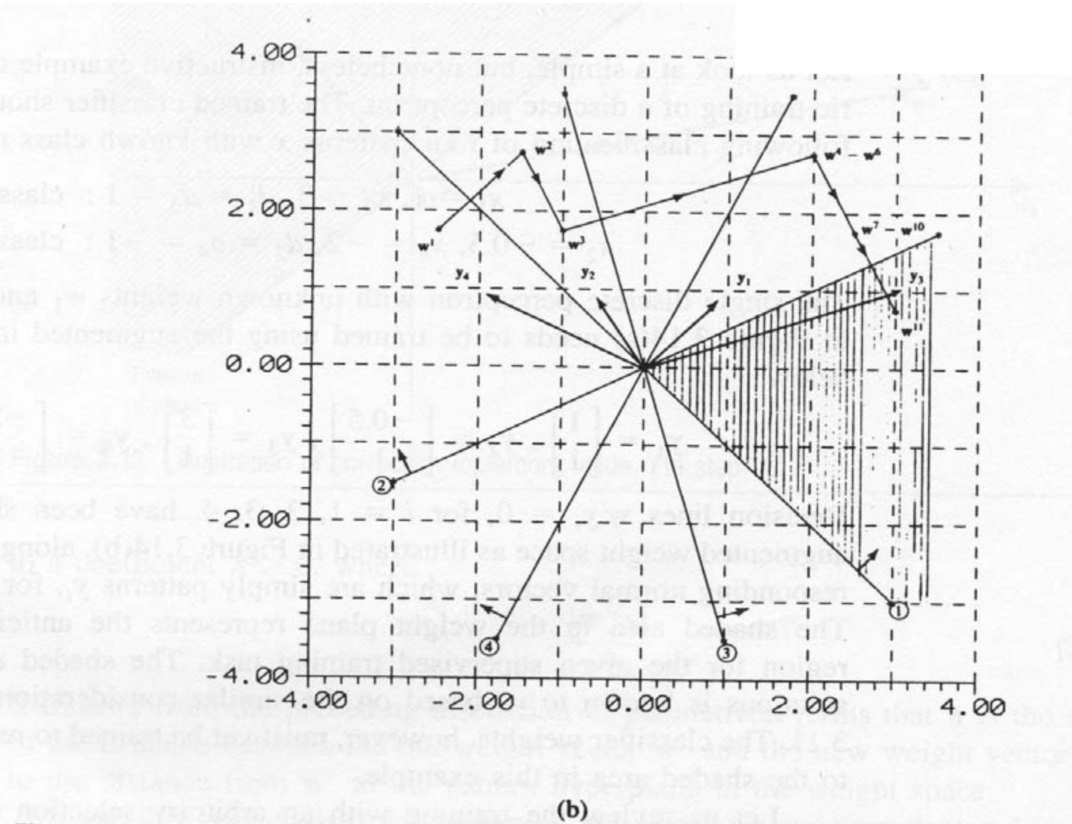


Figure 3.14a,b Discrete perceptron classifier training in Example 3.3: (a) network diagram, (b) fixed correction rule training.

$$\Delta \boldsymbol{w}^k = \frac{c}{2}\left[d_k - \operatorname{sgn}\left(\boldsymbol{w}^{kt}\boldsymbol{y}_j\right)\right]\boldsymbol{y}_j$$

What if $\boldsymbol{w}^{kt}\boldsymbol{y}_j = 0$?
-> interpreted as a mistake and followed by a correlation

Continuous Perceptron Training Algorithm

- Replace the TLU (Threshold Logic Unit) with the sigmoid activation function for two reasons:
  - Gain finer control over the training procedure
  - Facilitate the differential characteristics to enable computation of the error gradient

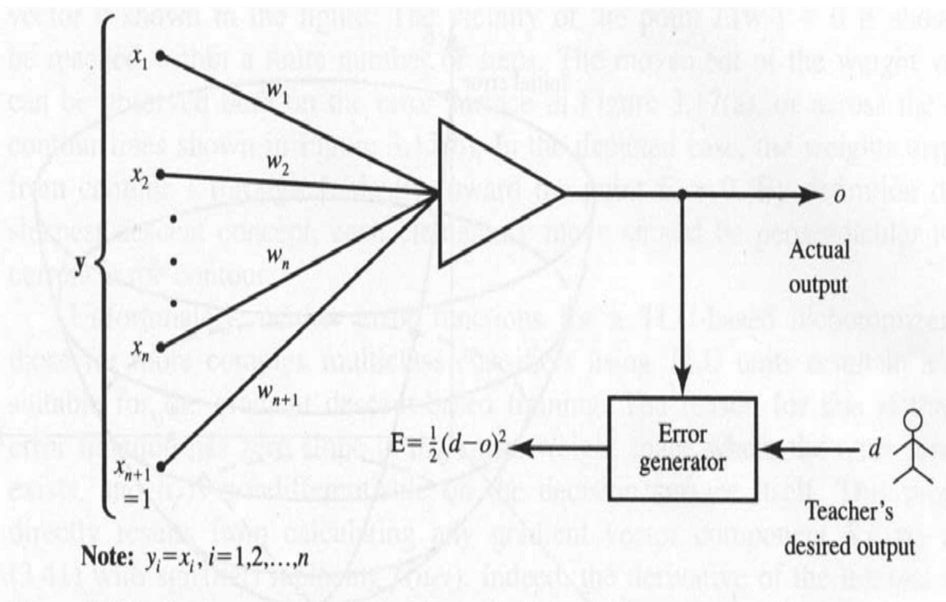


Figure 3.16 Continuous perceptron training using square error minimization.

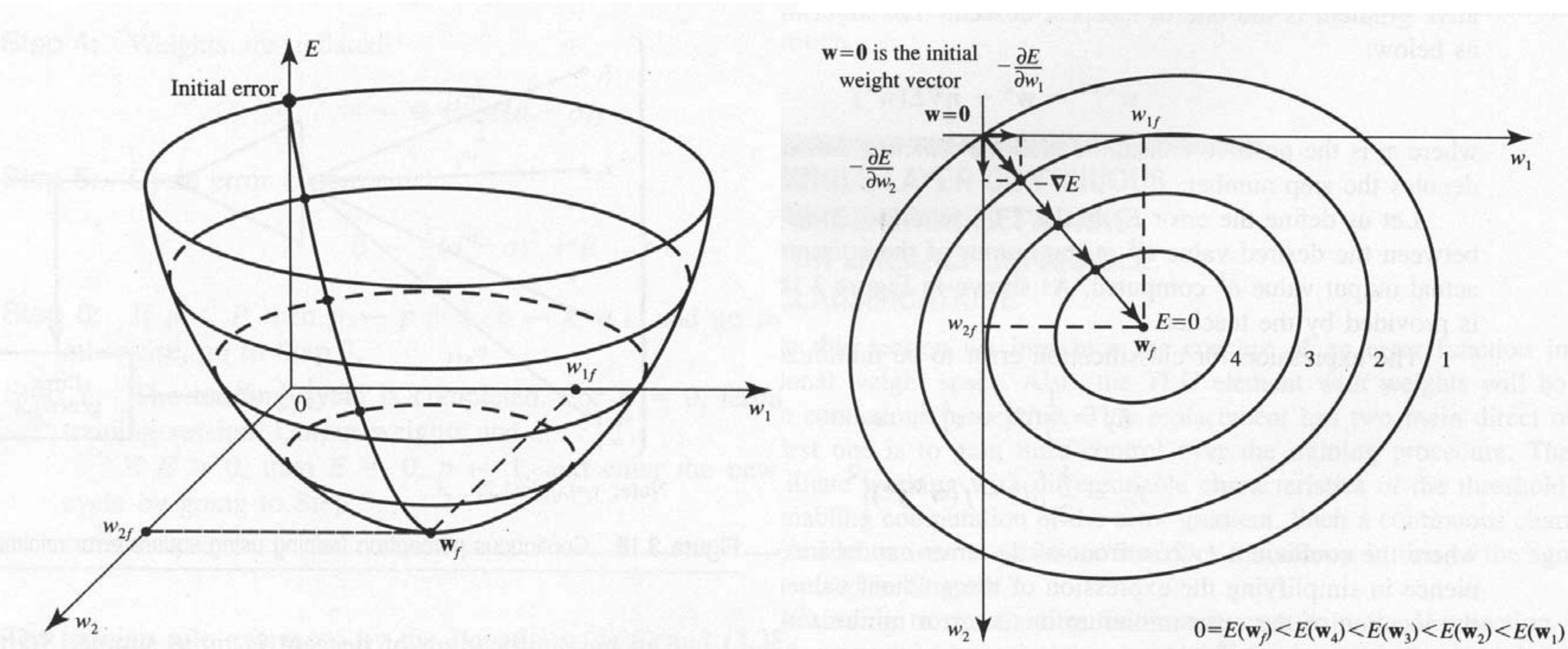
$$\hat{w} = w - \eta \nabla E(w)$$

learning constant

error gradient

Continuous Perceptron Training Algorithm (cont.)

- The new weights is obtained by moving in the direction of the negative gradient along the multidimensional error surface



(a) (b)

Continuous Perceptron Training Algorithm (cont.)

- Define the error as the squared difference between the desired output and the actual output

$$E = \frac{1}{2}(d - o)^2$$

or $$E = \frac{1}{2}\left[d - f\left(\mathbf{w}^t \mathbf{y}\right)\right]^2 = \frac{1}{2}\left[d - f(net)\right]^2$$

$$\nabla E(\mathbf{w}) = \frac{1}{2}\nabla\left(\left[d - f(net)\right]^2\right)$$

$$\nabla E(\mathbf{w}) \stackrel{\Delta}{=} \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \frac{\partial E}{\partial w_2} \\ \cdot \\ \cdot \\ \frac{\partial E}{\partial w_{n+1}} \end{bmatrix} = -(d - o) f'(net) \begin{bmatrix} \frac{\partial (net)}{\partial w_1} \\ \frac{\partial (net)}{\partial w_2} \\ \cdot \\ \cdot \\ \frac{\partial (net)}{\partial w_{n+1}} \end{bmatrix} = -(d - o) f'(net)\mathbf{y}$$

Continuous Perceptron Training Algorithm (cont.)

- Bipolar Continuous Activation Function

$$f(net) = \frac{2}{1+\exp(-\lambda \cdot net)} - 1 \qquad f'(net) = \lambda \cdot \frac{2\exp(-\lambda \cdot net)}{[1+\exp(-\lambda \cdot net)]^2} = \lambda \cdot \left\{1 - [f(net)]^2\right\} = \lambda(1-o^2)$$

$$\hat{\boldsymbol{w}} = \boldsymbol{w} + \frac{1}{2}\eta \cdot \lambda(d-o)(1-o^2)\boldsymbol{y}$$

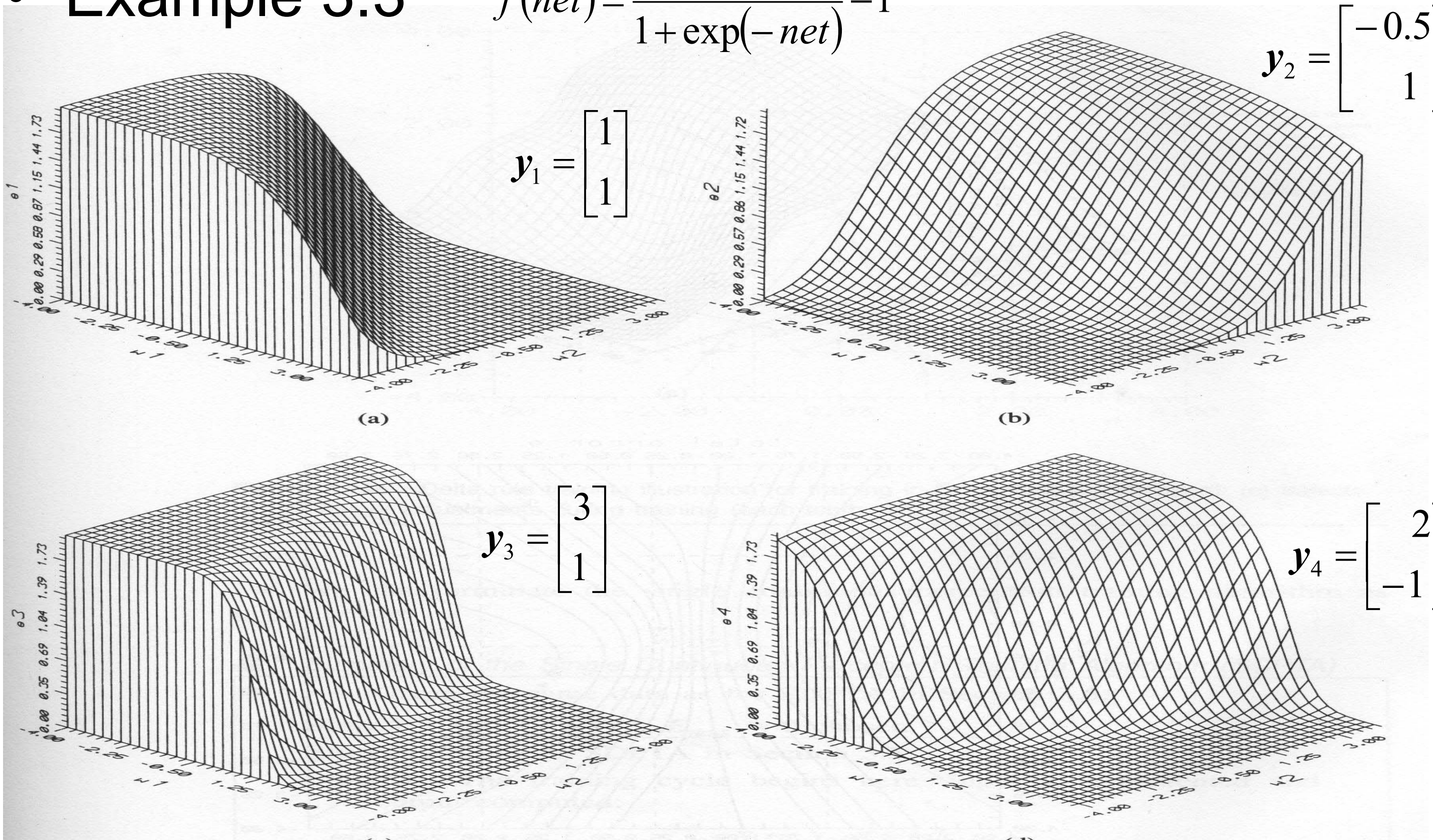
- Unipolar Continuous Activation Function

$$f(net) = \frac{1}{1+\exp(-\lambda \cdot net)} \qquad f'(net) = \frac{\lambda \cdot \exp(-\lambda \cdot net)}{[1+\exp(-\lambda \cdot net)]^2} = \lambda \cdot f(net)[1-f(net)] = \lambda \cdot o(1-o)$$

$$\hat{\boldsymbol{w}} = \boldsymbol{w} + \eta \cdot \lambda \cdot (d-o)o(1-o)\boldsymbol{y}$$

Continuous Perceptron Training Algorithm (cont.)

- Example 3.3 $f(net) = \dfrac{2}{1+\exp(-net)} - 1$



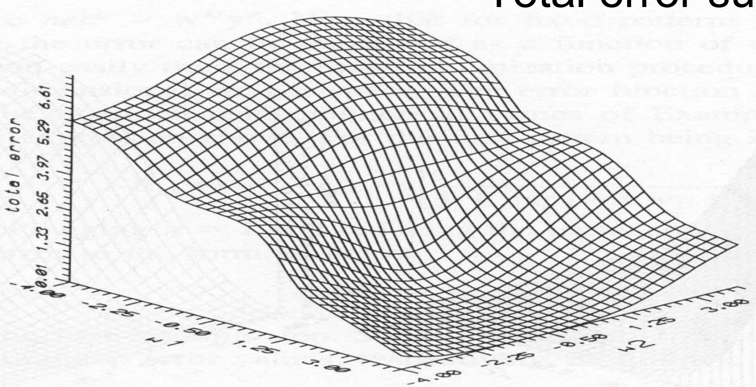
$$\boldsymbol{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \boldsymbol{y}_2 = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \quad \boldsymbol{y}_3 = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \quad \boldsymbol{y}_4 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

Figure 3.18 Error functions for individual patterns, Example 3.4: (a) first pattern, E_1, (b) second pattern, E_2, (c) third pattern, and E_3, and (d) fourth pattern, E_4.

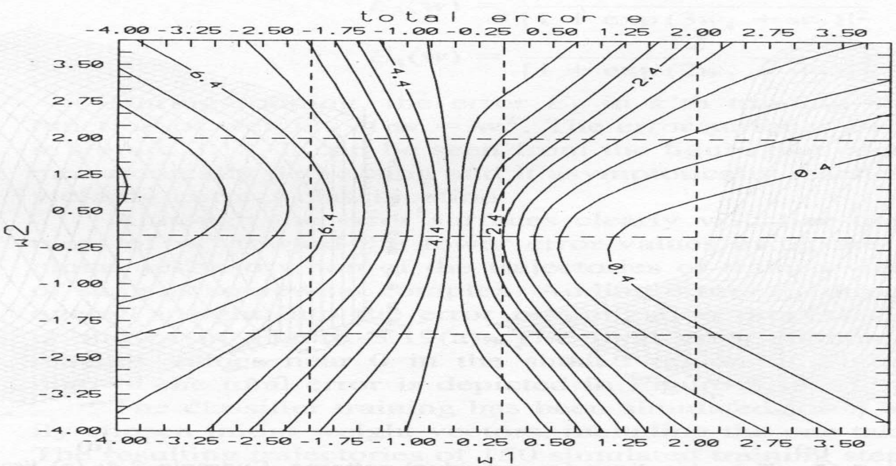
Continuous Perceptron Training Algorithm (cont.)

- Example 3.3

Total error surface



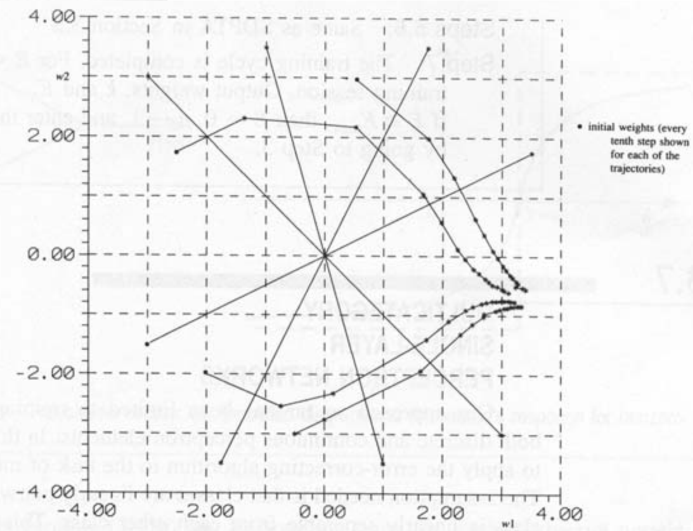
(a)



(b)

Figure 3.19a,b Delta rule training illustration for training in Example 3.4: (a) total error surface, (b) total error contour map.

Trajectories started from four arbitrary initial weights



(c)

Figure 3.19c Delta rule training illustration for training in Example 3.4 *(continued)*: (c) trajectories of weight adjustments during training (each tenth step shown).

Continuous Perceptron Training Algorithm (cont.)

- Treat the last fixed component of input pattern vector as the neuron activation threshold

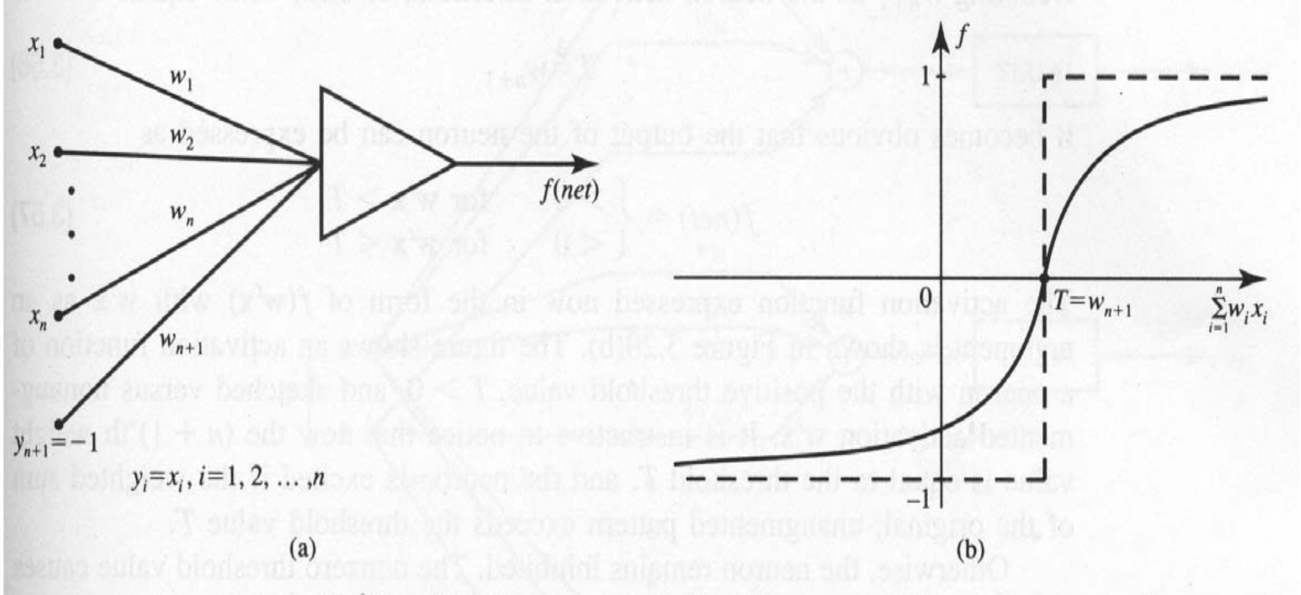
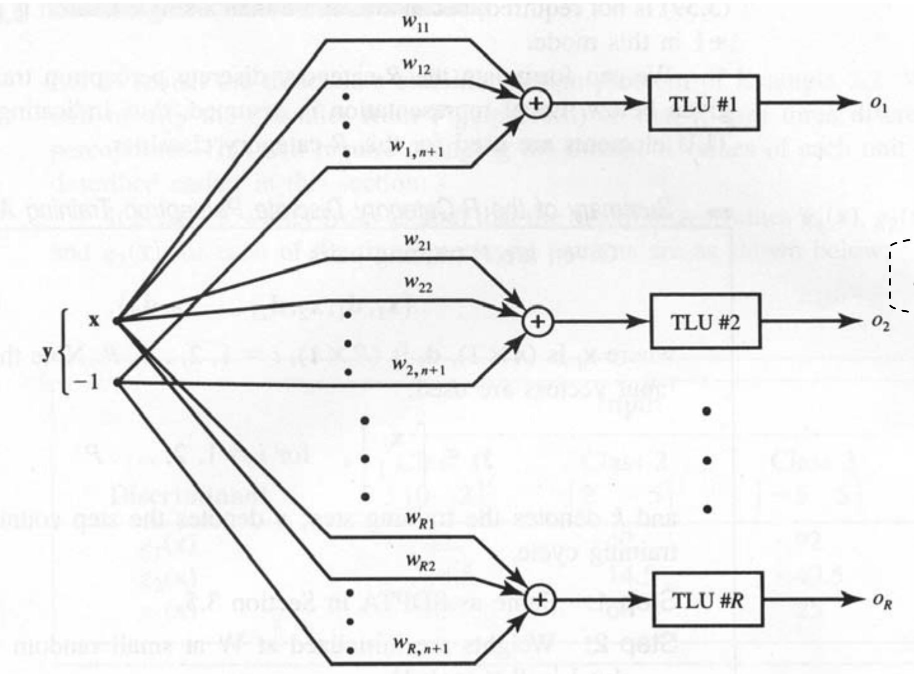


Figure 3.20 The biased neuron: (a) simplified diagram and (b) neuron's response for nonzero threshold.

Continuous Perceptron Training Algorithm (cont.)

- *R*-category linear classifier using *R* discrete bipolar perceptrons
  - Goal: The *i*-th TLU response of +1 is indicative of class *i* and all other TLU respond with -1



$$\hat{\boldsymbol{w}}_i = \boldsymbol{w}_i + \frac{1}{2} c \cdot (d_i - o_i) \boldsymbol{y}$$

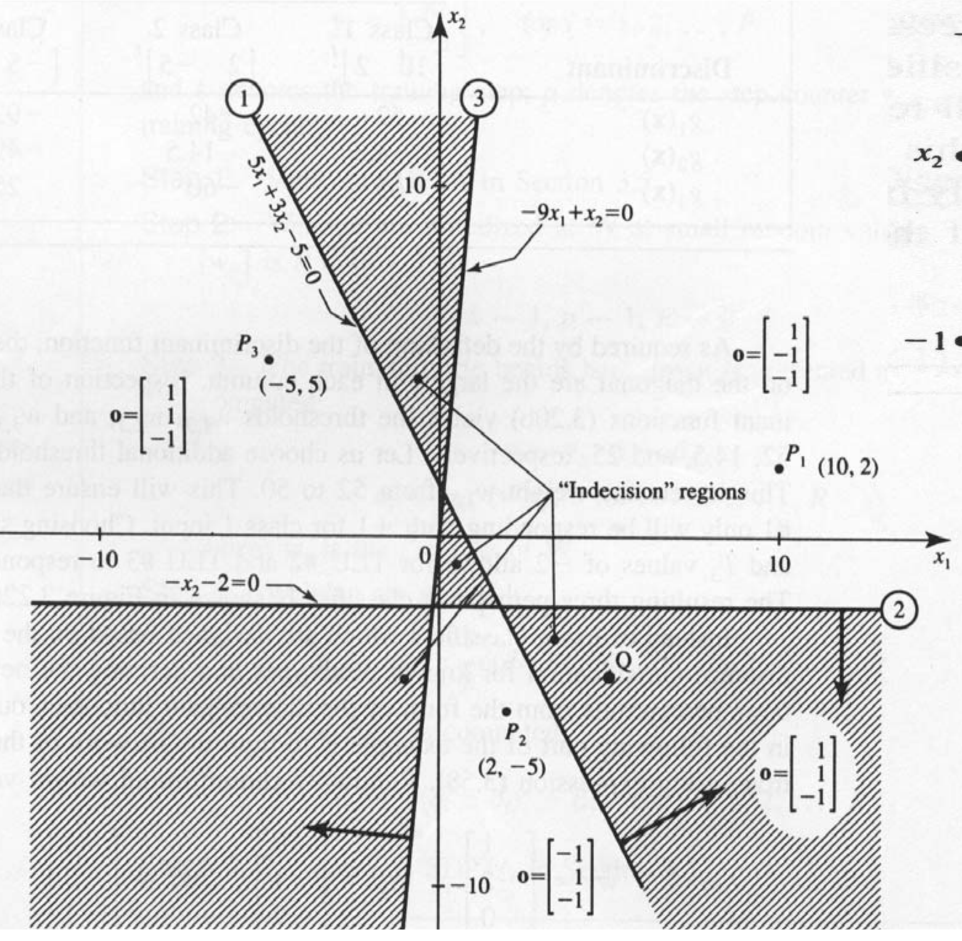
$d_i = 1, d_j = -1, \text{ for } j = 1,2,..,R, \ j \neq i$

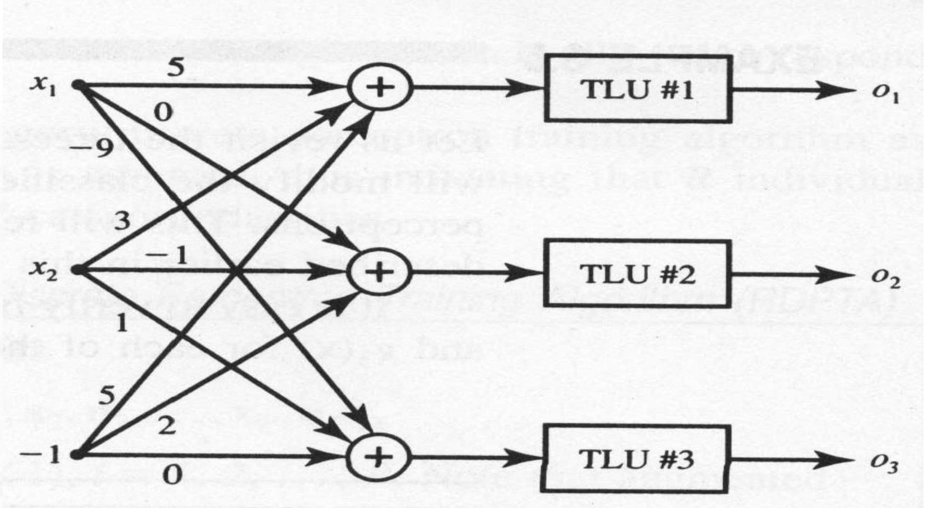
For "local representation"

Figure 3.21 R-category linear classifier using R discrete perceptrons.

Continuous Perceptron Training Algorithm (cont.)

- Example 3.5





Continuous Perceptron Training Algorithm (cont.)

- *R*-category linear classifier using *R* continuous bipolar perceptrons

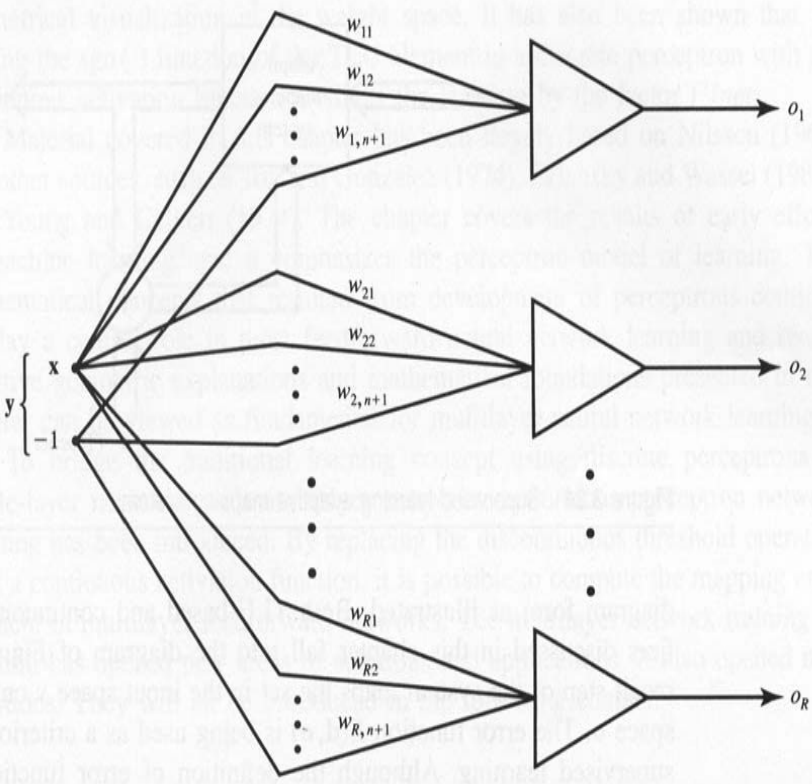


Figure 3.23 *R*-category linear classifier using continuous perceptrons.

$$\hat{\boldsymbol{w}}_i = \boldsymbol{w}_i + \frac{1}{2}\eta \cdot \lambda \left(d_i - o_i\right)\left(1 - o_i^2\right)\boldsymbol{y}$$

for $i = 1, 2, \ldots, R$

$d_i = 1,\ d_j = -1,$ for $j = 1, 2, .., R,\ j \neq i$

Continuous Perceptron Training Algorithm (cont.)

- Error function dependent on the difference vector ***d***-***o***

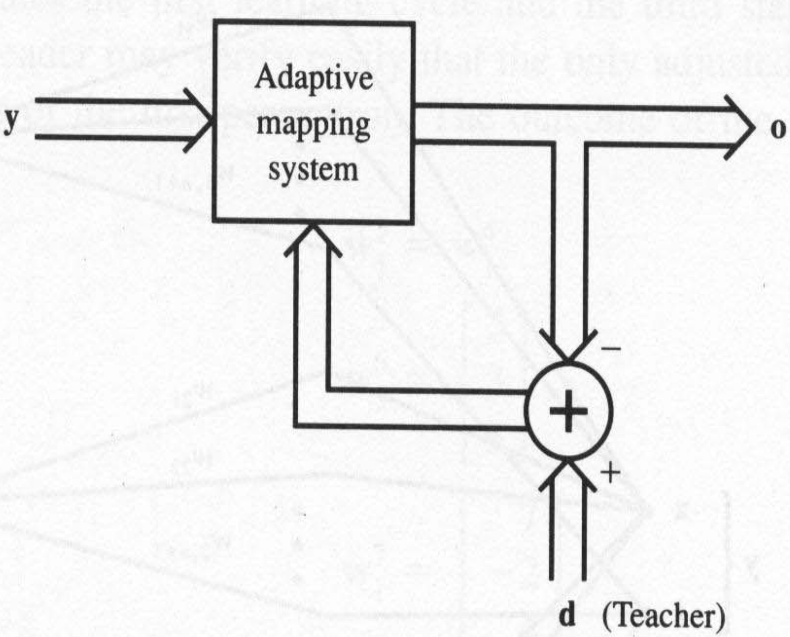


Figure 3.24 Supervised learning adaptive mapping system.

Bayes' Classifier vs. Percepron

- Perceptron operates on the promise that the patterns to be classified are *linear separable* (otherwise the training algorithm will oscillate), while Bayes' classifier assumes the (Gaussian) distribution of two classes certainly do overlap each other
- The perceptron is nonparametric while the Bayes' classifier is parametric (its derivation is contingent on the assumption of the underlying distributions)
- The perceptron is simple and adaptive, and needs small storage, while the Bayes' classifier could be made adaptive but at the expanse of increased storage and more complex computations

Homework

- P3.5, P3.7, P3.9, P3.22