EENG582: Artificial Neural Networks

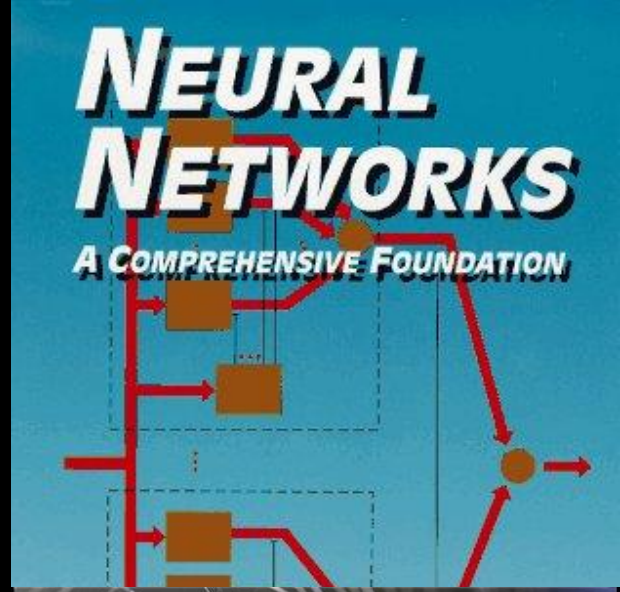
Neural Networks
A Comprehensive Foundation
by S. Haykin

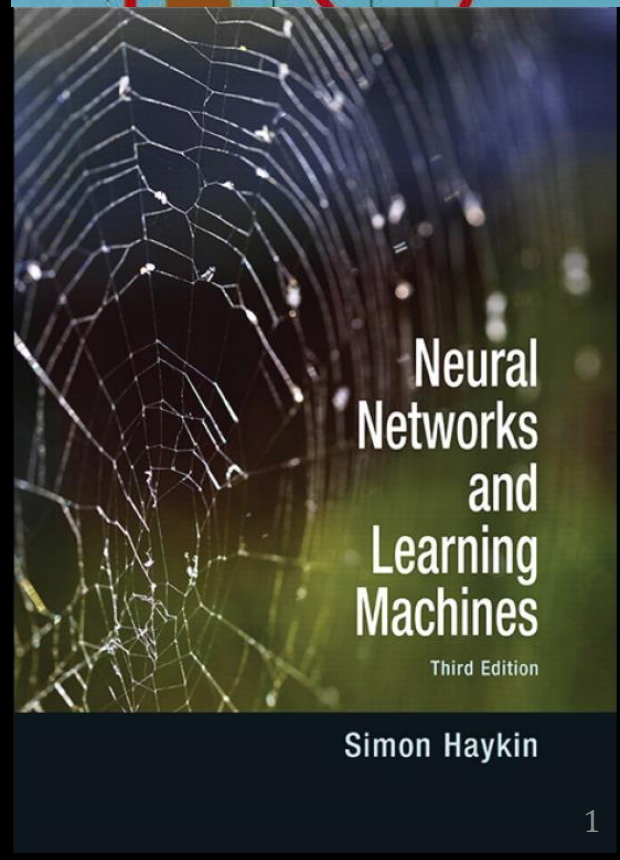
Sections 3.6 – 3.10 Learning Rate / Curve and Convergence

3 Single Layer Perceptrons

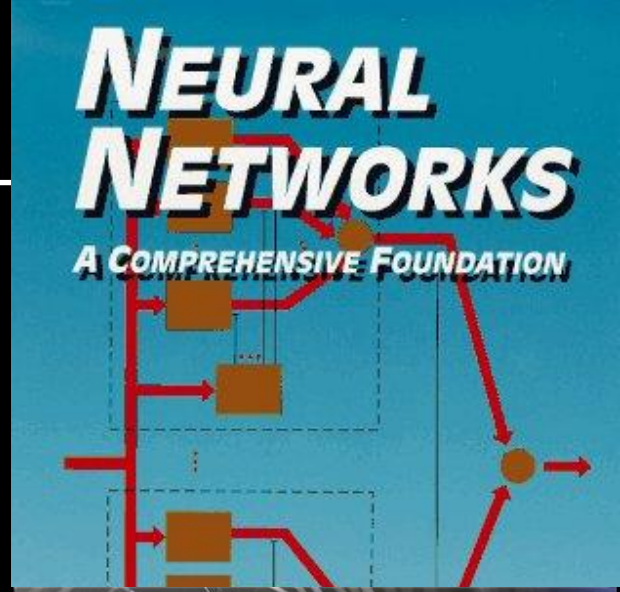
Prof. Dr. Hasan AMCA
Electrical and Electronic Engineering Department
(ee.emu.edu.tr)

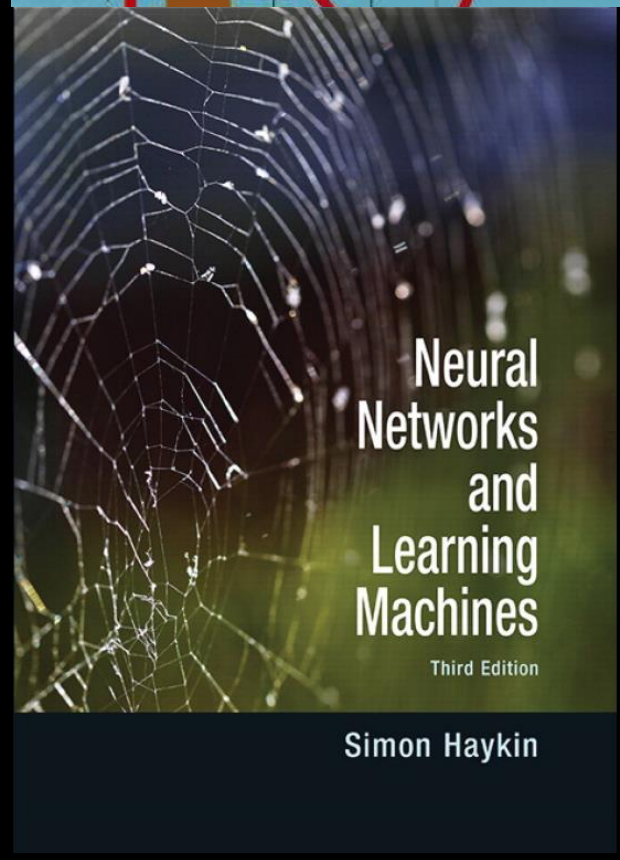
Eastern Mediterranean University
(emu.edu.tr)





3 Single Layer Perceptrons





3.6 Learning Curves

- An informative way of examining the convergence behavior of *LMS* (*adaptive filter*) *algorithm* is to plot the *learning curve* of the filter under varying environmental conditions

- **Learning Curve** is a plot of the *mean-square value of the estimation error*, $\mathcal{E}_{av}(n)$ versus number of iterations n as shown in Fig. 3.4

- To compute the ensemble-averaged learning curve (i.e., plot of $\mathcal{E}_{av}(n)$ vs n), we take the average of these sample learning curves over the ensemble of adaptive filters used in the experiment, thereby smoothing out the effects of noise.

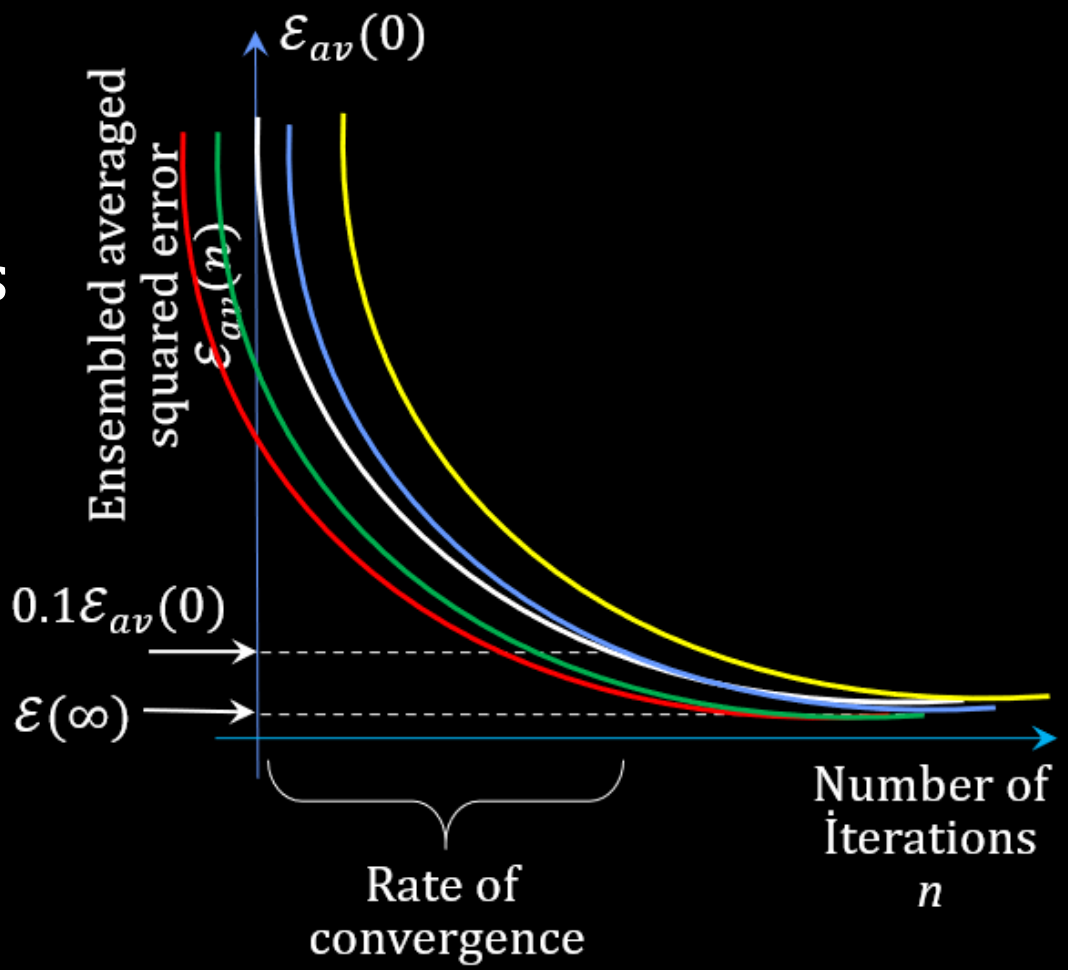


Fig. 3.4 Idealized learning curve of the LMS algorithm.

3.6 Learning Curves

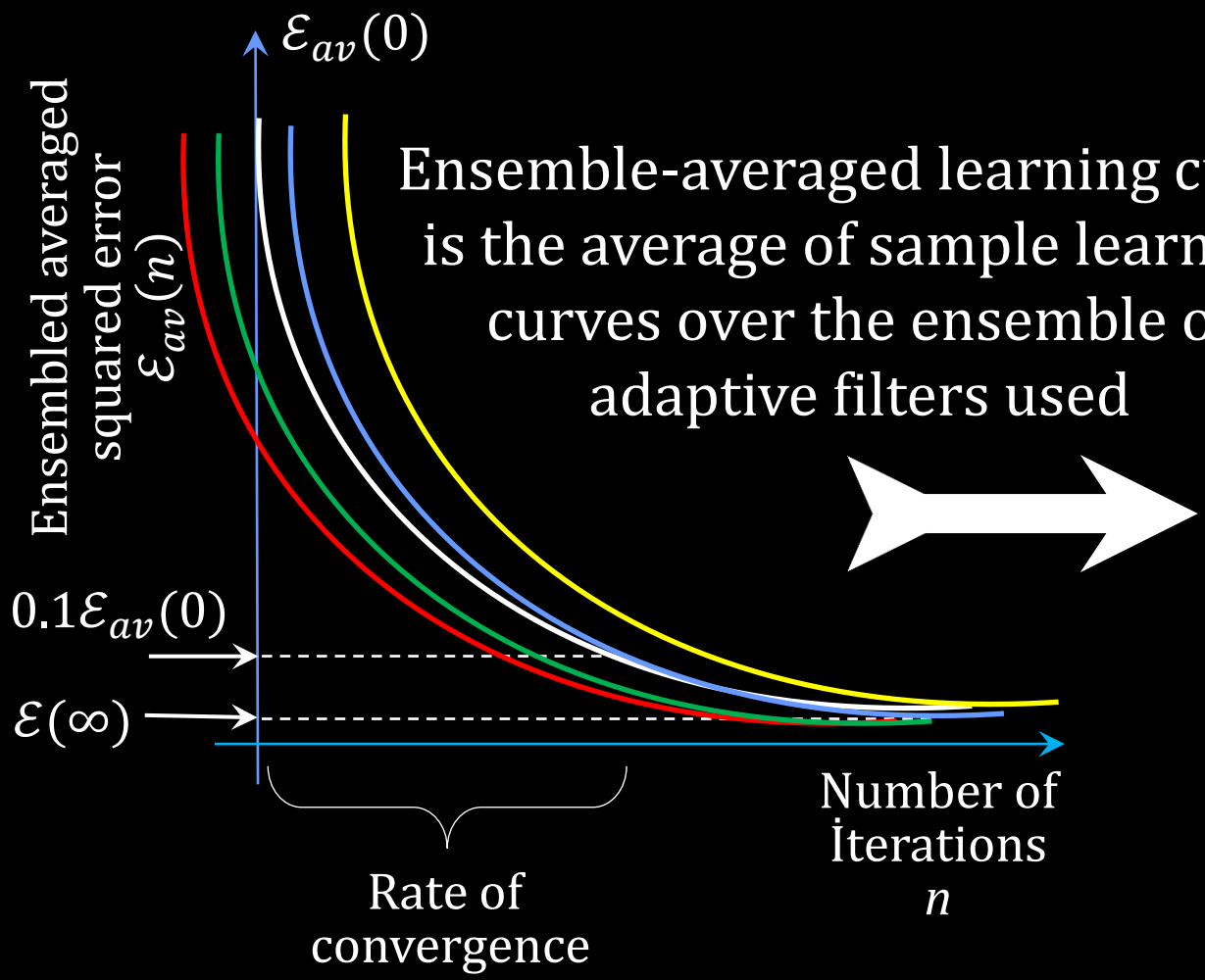


Fig. 3.4 Idealized learning curve of the LMS algorithm.

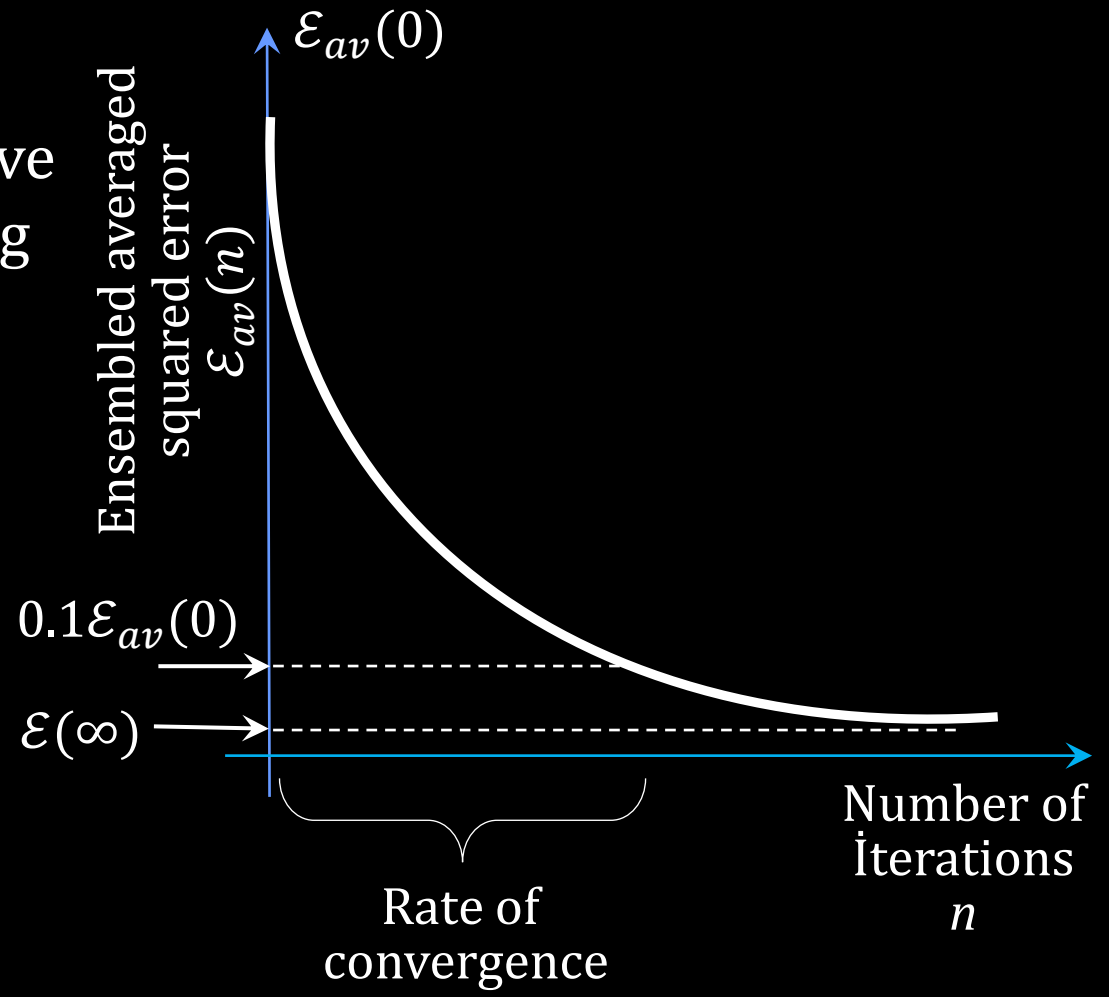


Fig. 3.4 Idealized learning curve of the LMS algorithm.

3.6 Learning Curves

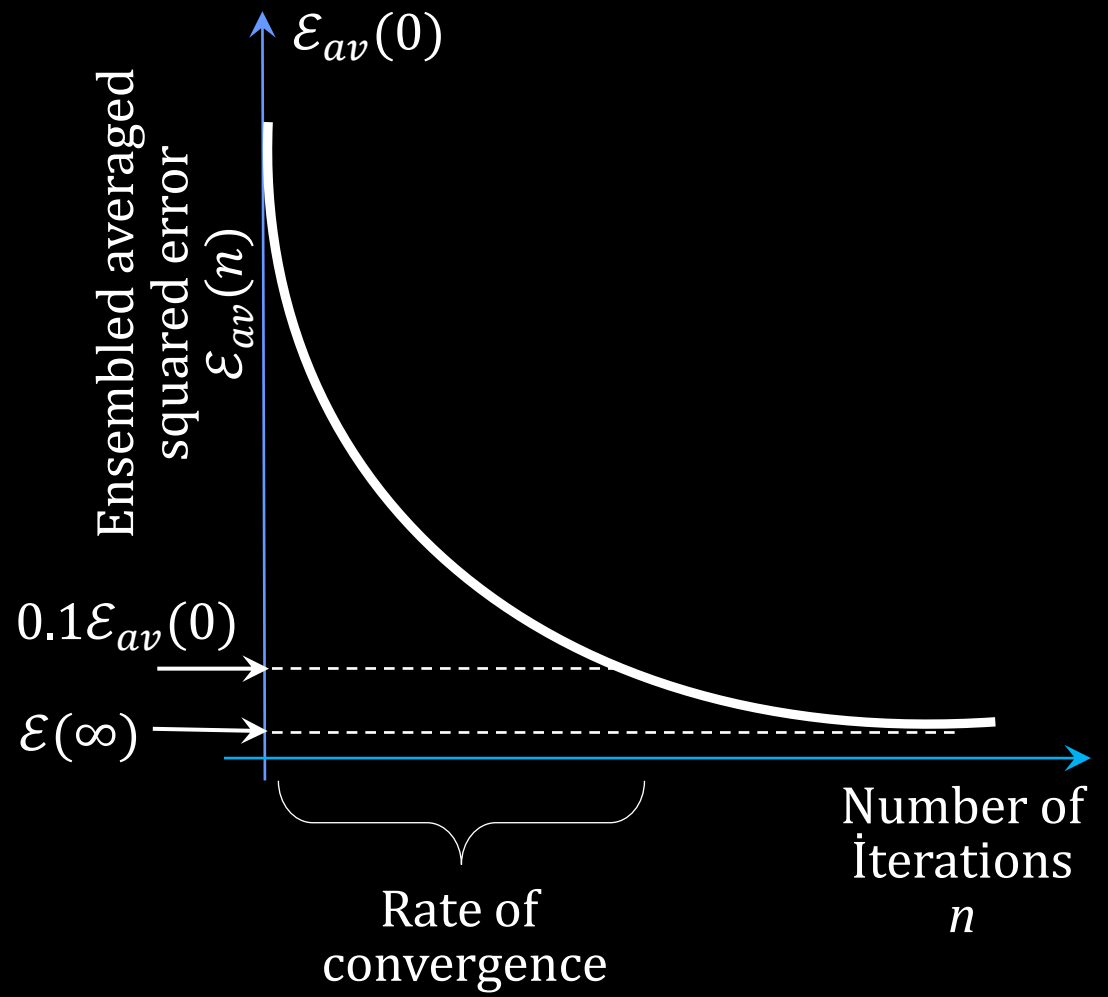


Fig. 3.4 Idealized learning curve of the LMS algorithm.

3.6 Learning Curves

- Considering an experiment involving an ensemble of adaptive filters, with each filter operating under the control of a specific algorithm.
- For each filter we plot squared value of the estimation error $\mathcal{E}_{av}(n)$ (i.e., the difference between the desired response and the actual filter output) versus n
- A sample learning curve so obtained consists of noisy exponentials.
- The ensemble-averaged learning curve starts from a large value $\mathcal{E}_{av}(0)$ determined by the initial conditions, then decreases to a steady-state value $\mathcal{E}_{av}(\infty)$, as illustrated in Fig. 3.4.
- Rate of convergence of the adaptive filter is number of iterations, n required for $\mathcal{E}_{av}(n)$ to decrease to a value, such as 10 % of the initial value $\mathcal{E}_{av}(0)$.

3.6 Learning Curves

- Mis-adjustment of the ensemble-averaged learning curve denoted by $\mathcal{M}$
- $\mathcal{M}$ is given by the minimum mean-square error (MMSE) produced by the Wiener filter, designed on the basis of known values of the correlation matrix $\mathbf{R}_x$ and cross-correlation vector $\mathbf{r}_{xd}$ may be defined as the mis-adjustment for the adaptive filter as follows

$$\mathcal{M} = \frac{\mathcal{E}(\infty) - \mathcal{E}_{min}}{\mathcal{E}_{min}} = \frac{\mathcal{E}(\infty)}{\mathcal{E}_{min}} - 1 \qquad (3.46)$$

- The smaller $\mathcal{M}$ is compared to unity as percentage, the more accurate is the adaptive filtering action of the algorithm.
- E.g. 10 percent greater than the minimum mean square error $\mathcal{E}_{min}$ produced by the corresponding Wiener filter is satisfactory in practice.

3.6 Learning Curves

- Another important characteristic of LMS algorithm is the *settling time*, which is the time constant, τ_{av}, of the exponential approximating the learning curve.
- The smaller the τ_{av}, the faster the LMS algorithm will converge to a "steady state" condition.
- Learning rate, mis-adjustment and time constant are related as follows:

$$\mathcal{M} \propto \eta \propto 1/\tau_{av}$$

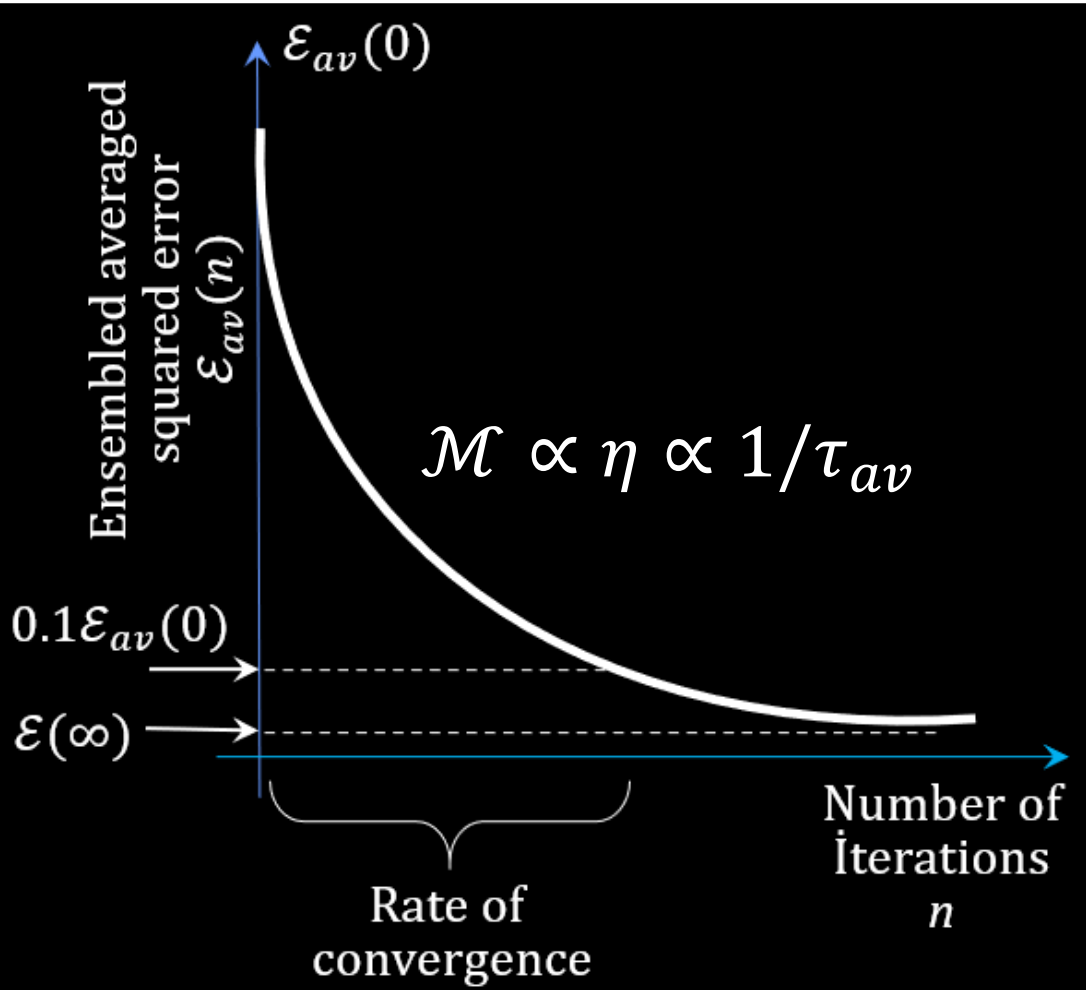


Fig. 3.4 Idealized learning curve of the LMS algorithm.

3.7 Learning-Rate Annealing Schedules

- The learning-rate parameter is maintained constant throughout the computation, as

$$\eta(n) = \eta_0 \quad \text{for all } n \tag{3.47}$$

- In *stochastic approximation*, the learning-rate parameter is time-varying as

$$\eta(n) = \frac{c}{n} \quad \text{where } c = \tau\eta_0 \text{ is a constant} \tag{3.48}$$

- We may define the converge Schedule by

$$\eta(n) = \frac{\eta_0}{1+(n/\tau)} \quad \eta_0 \text{ and } \tau \text{ are constants selected by user} \tag{3.49}$$

- With these approximations, the algorithm operates essentially same as the "standard" LMS algorithm, as indicated in Fig. 3.5.

3.7 Learning-Rate Annealing Schedules

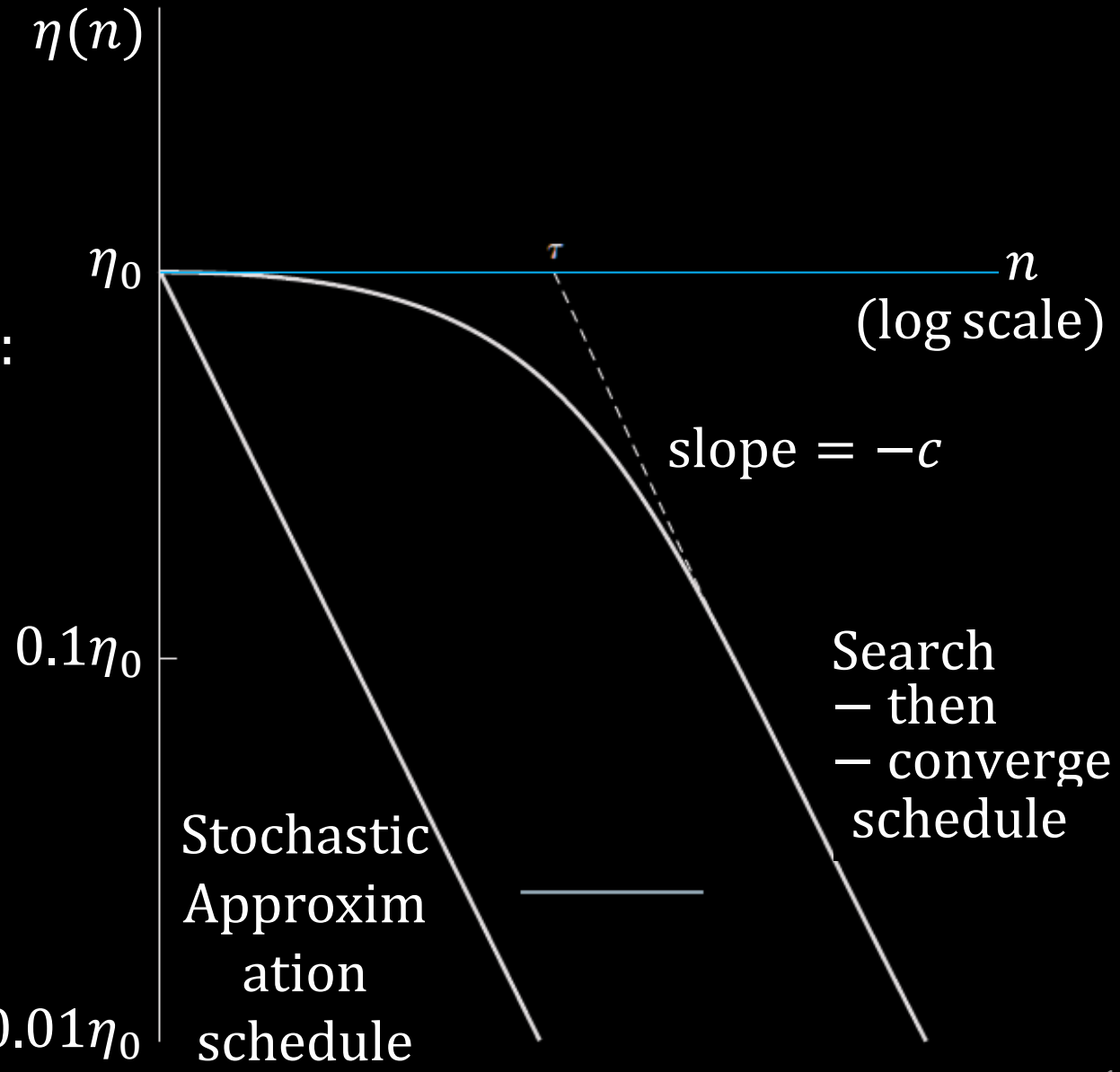


Fig. 3.5 Learning-rate annealing schedules: The horizontal axis, printed in color, pertains to the standard LMS algorithm.

3.8 Perceptron

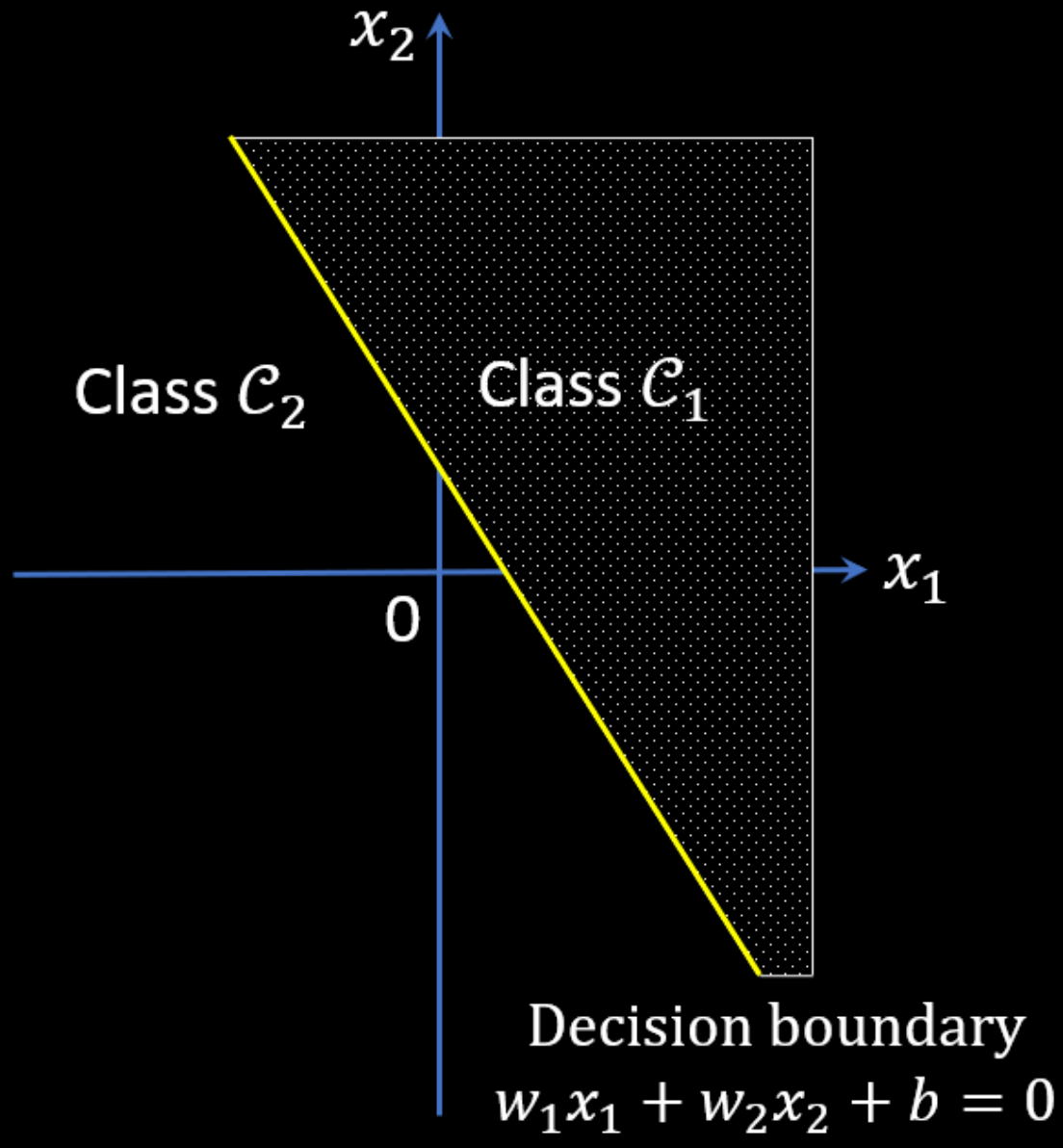
- LMS algorithm is built around a linear neuron whereas the perceptron is built around a nonlinear neuron.
- The neuronal model consists of a linear combiner followed by a hard limiter (performing the signum function), as depicted in Fig. 3.6.
- The induced local field of the neuron at the hard limiter input is

$$v = \sum_{i=1}^{m} w_i x_i + b \qquad (3.50)$$

- The goal of the perceptron is to correctly classify the set of externally applied stimuli $x_1, x_2, \dots, x_m$ into one of two classes, $\mathcal{C}_1$ or $\mathcal{C}_2$.
- The decision rule for the classification is to assign the point represented by the inputs $x_1, x_2, \dots, x_m$ to class $\mathcal{C}_1$ if the perceptron output y is -1 and to class $\mathcal{C}_2$ if it is -1 as shown in Fig. 3.7.

3.8 Perceptron

- For the case of two input variables (x_1, x_2), for which the decision boundary takes the form of a straight line,
  - a point (x_1, x_2) that lies above the boundary line is assigned to class $\mathcal{C}_1$ and
  - a point (x_1, x_2) that lies below the boundary line is assigned to class $\mathcal{C}_1$.
- Note also that the effect of the bias b is merely to shift the decision boundary away from the origin.



3.8 Perceptron

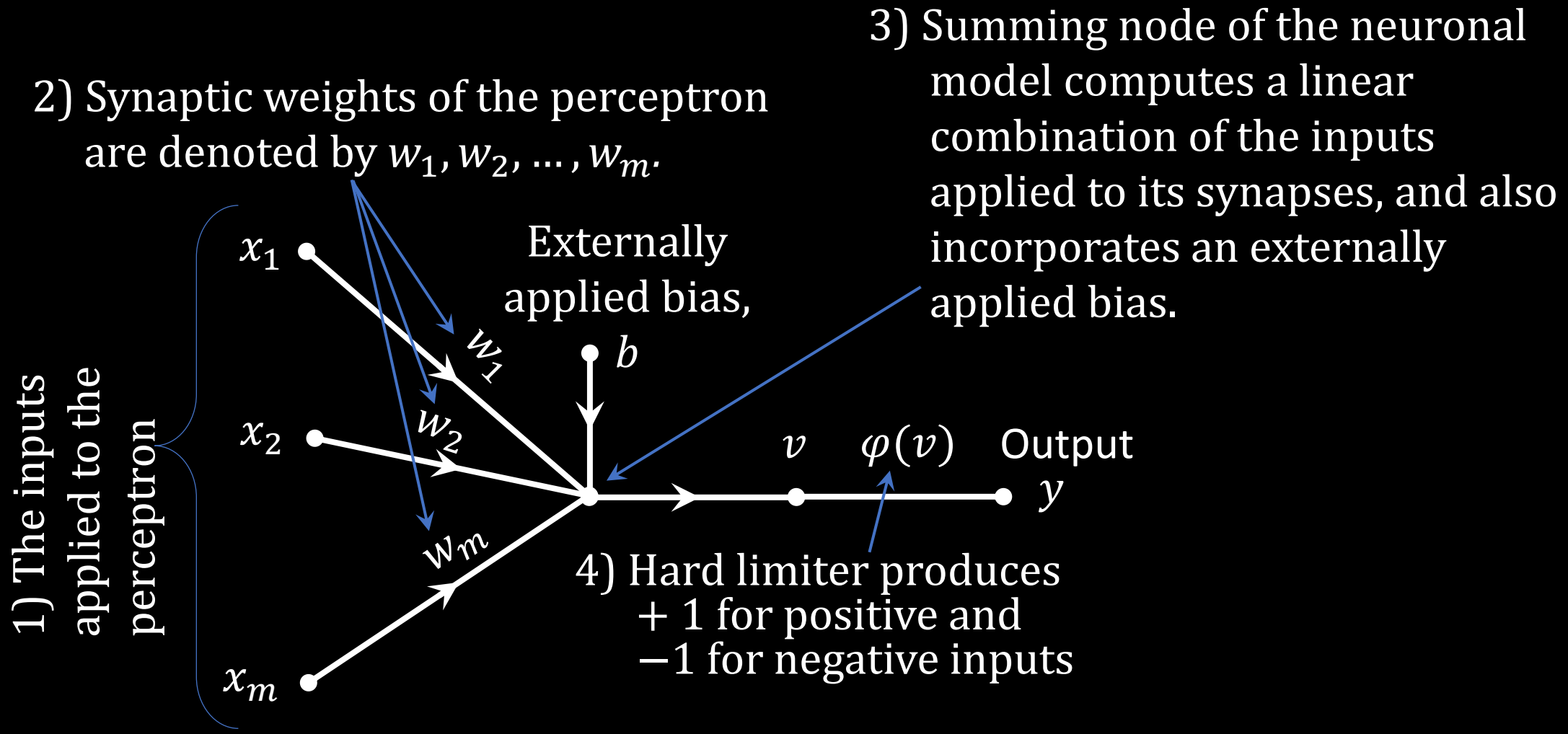
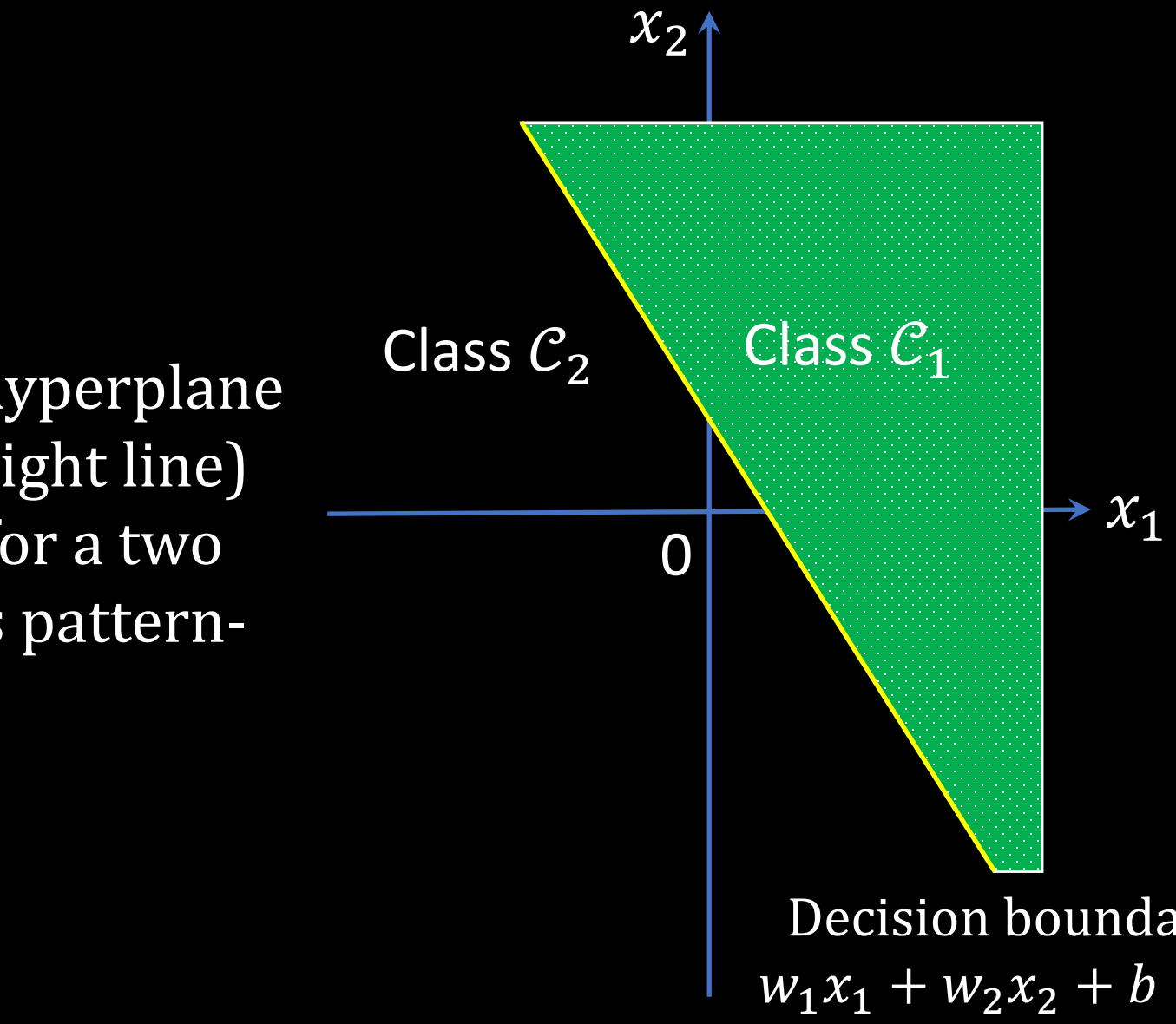


Fig. 3.6 Signal-flow graph of the perceptron.

3.8 Perceptron

Fig. 3.7 Illustration of the hyperplane (in this example, a straight line) as decision boundary for a two dimensional, two-class pattern-classification problem.



3.9 Perceptron Convergence Theorem

- To derive the error-correction learning algorithm for the perceptron, we work with the modified signal-flow graph model in Fig. 3.8.
- We may thus define the (m+1)-by-1 input and weight vectors as

$$\mathbf{x}(n) = [+1, x_1(n), x_2(n), \dots, x_m(n)]^T$$

and

$$\mathbf{w}(n) = [b(n), w_1(n), w_2(n), \dots, w_m(n)]^T$$

- where n denotes the iteration step in applying the algorithm.
- Accordingly, the linear combiner output is written in the compact form

$$v(n) = \sum_{i=0}^{m} w_i(n)x_i(n) = \mathbf{w}^T(n)\mathbf{x}(n) \qquad (3.50)$$

- where $w_0(n)$ represents the bias $b(n)$.

3.9 Perceptron Convergence Theorem

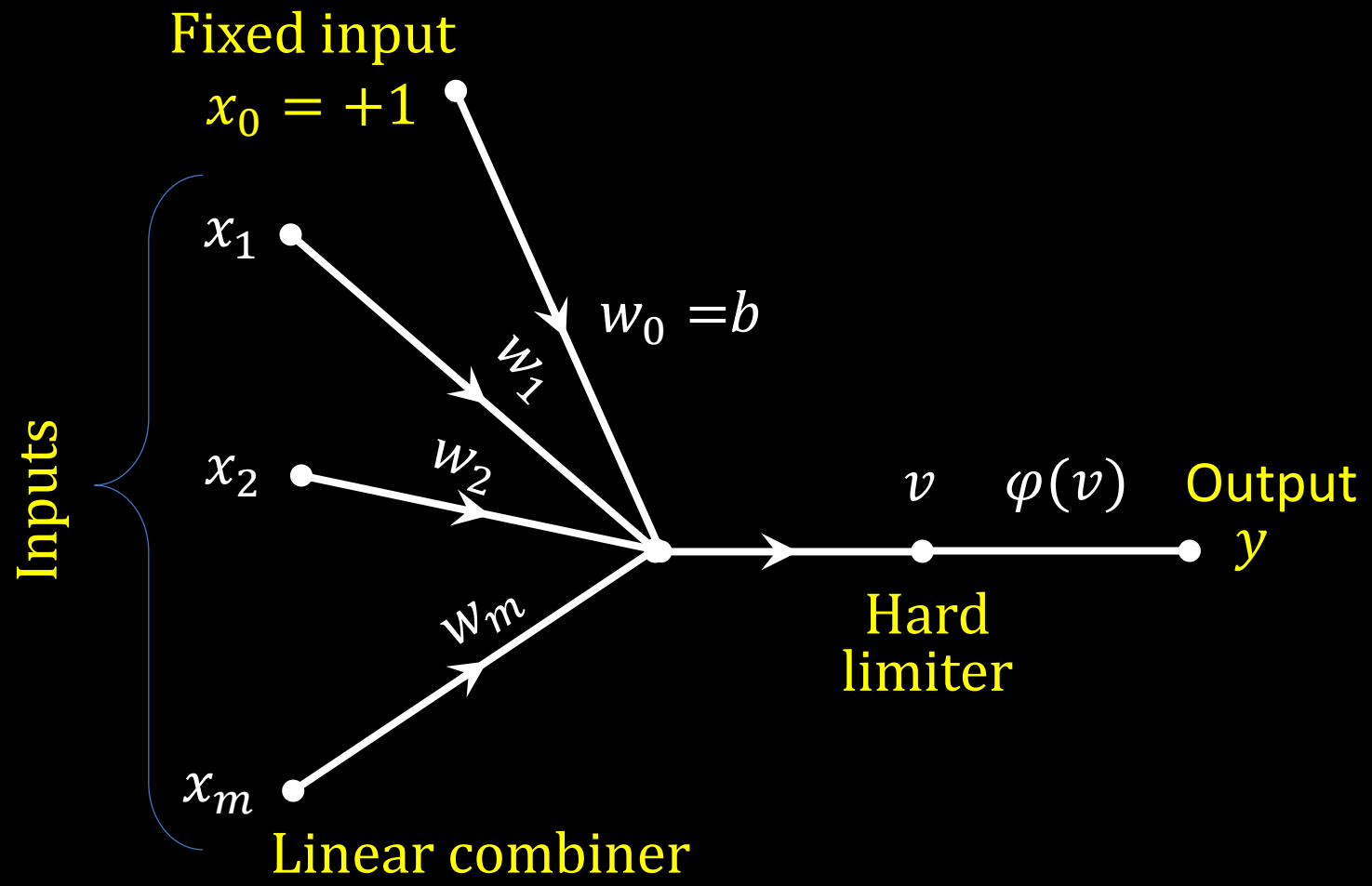


Fig. 3.8 Modified signal-flow graph of the perceptron.

3.9 Perceptron Convergence Theorem

- For fixed n, the equation $\mathbf{w}^T\mathbf{x} = 0$, plotted in an m-dimensional space (plotted for some prescribed bias) with coordinates $x_1, x_2, \ldots, x_m$ defines a hyperplane as the decision surface between two different classes of inputs.
- For the perceptron to function properly, the two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ must be linearly separable so that the patterns to be classified are sufficiently separated from each other to ensure that the decision surface consists of a hyperplane.
- This requirement is illustrated in Fig. 3.9 for the case of a two-dimensional perceptron
  - where the two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are sufficiently separated from each other to draw a hyperplane (in this case a straight line) as the decision boundary.
- If the two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are too close to each other as in Fig. 3.9b, they become nonlinearly separable, that is beyond the computing capability of the perceptron.

3.9 Perceptron Convergence Theorem

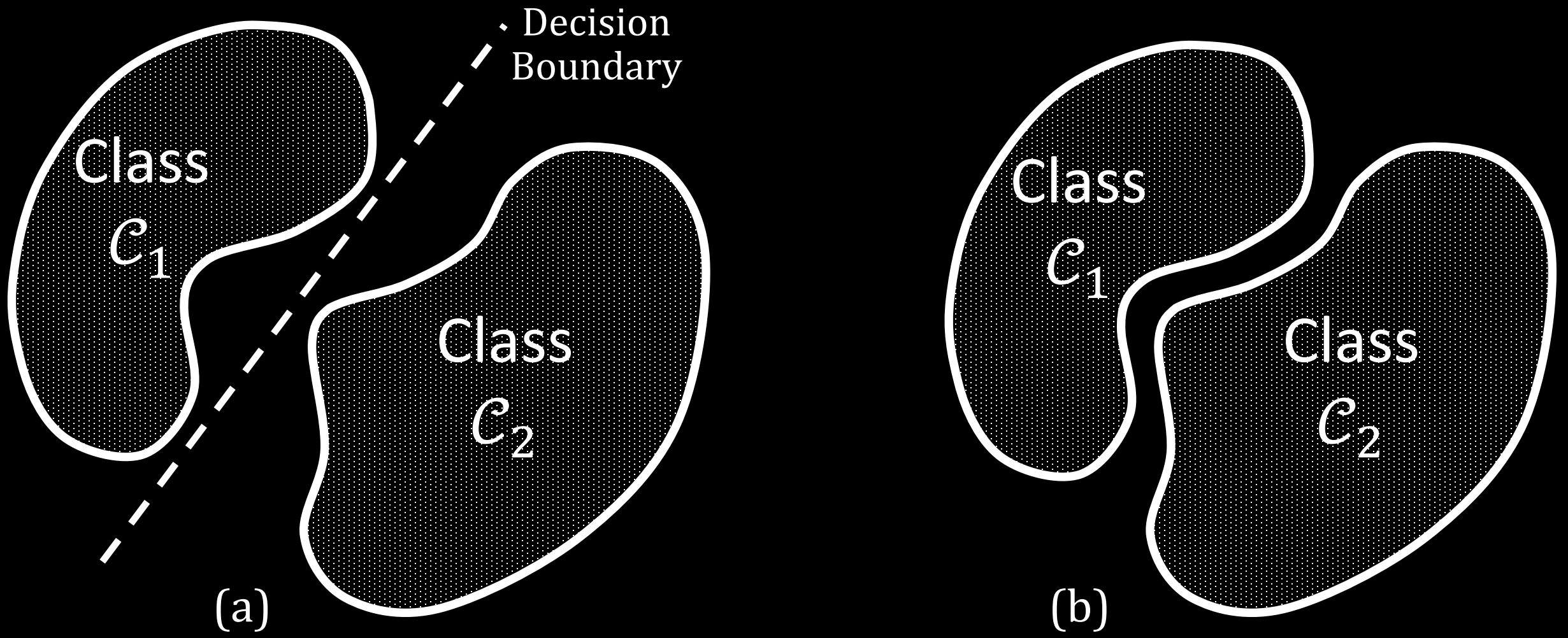
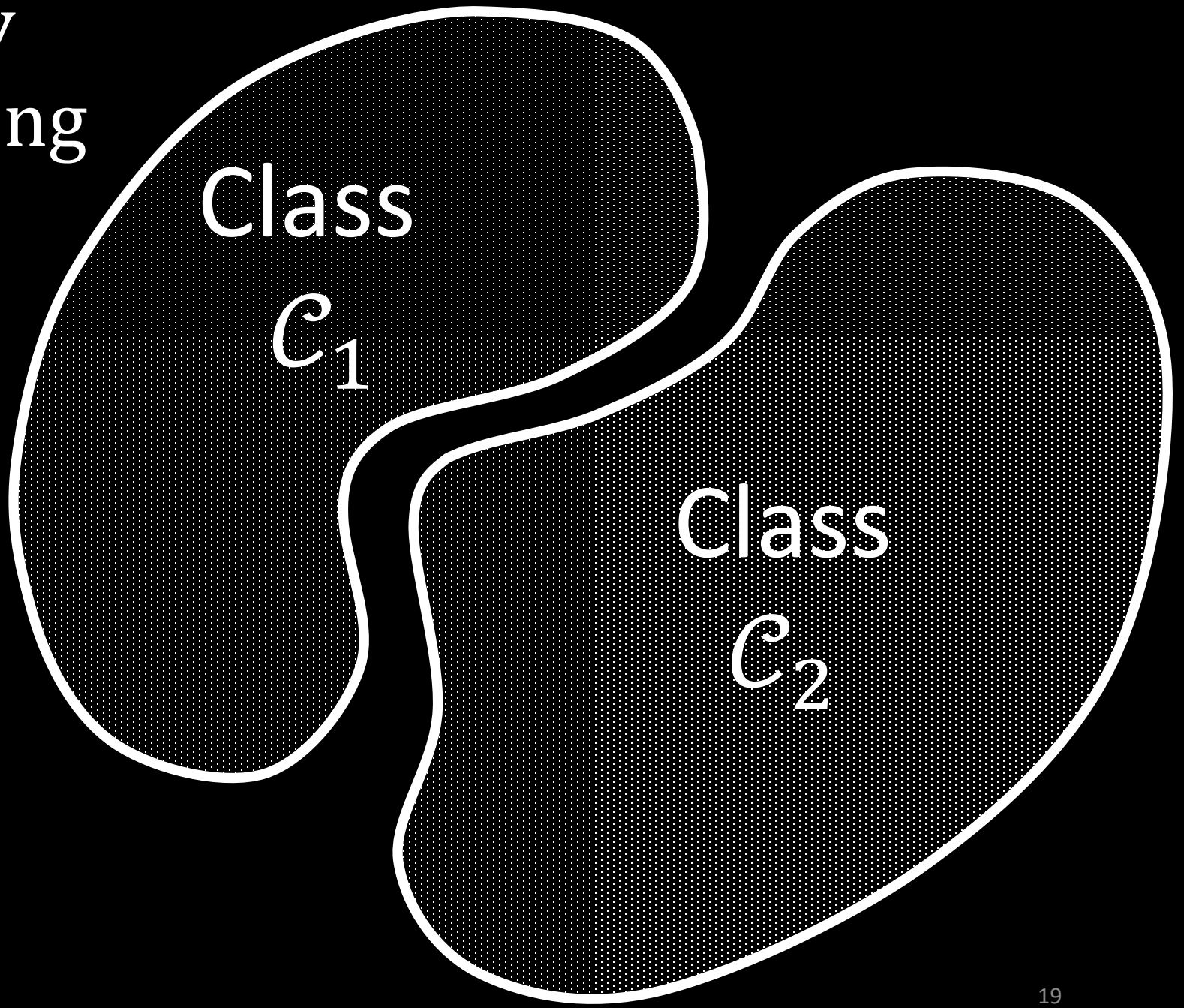


Fig. 3.9 (a) A pair of linearly separable patterns.
(b) A pair of nonlinearly separable

A pair of arbitrary patterns representing the two classes



3.9 Perceptron Convergence Theorem

- When the input variables of the perceptron originate from two linearly separable classes where
  - $\mathcal{H}_1$ is the subset of training vectors $\mathbf{x}_1(1), \mathbf{x}_1(2), \ldots$ that belong to class $\boldsymbol{\mathcal{C}_1}$ and
  - $\mathcal{H}_2$ is the subset of training vectors $\mathbf{x}_2(1), \mathbf{x}_2(2), \ldots$ that belong to class $\boldsymbol{\mathcal{C}_2}$
- The union of $\mathcal{H}_1$ and $\mathcal{H}_2$ is the complete training set $\mathcal{H}$.
- Then the training process involves adjustment of the weight vector $\mathbf{w}$ in such a way that two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are linearly separable such that:

$$\begin{aligned} &\mathbf{w}^T\mathbf{x} > 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathcal{C}_1 \\ &\mathbf{w}^T\mathbf{x} \leq 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathcal{C}_2 \end{aligned} \qquad (3.53)$$

3.9 Perceptron Convergence Theorem

- The algorithm for adapting the weight vector of the elementary perceptron may now be formulated as follows:

  1. If the nth member of the training set, $\mathbf{x}(n)$, is correctly classified by the weight vector $\mathbf{w}(n)$ computed at the nth iteration of the algorithm, no correction is made to the weight vector of the perceptron in accordance with the rule:

$$\begin{aligned} &\mathbf{w}(n+1) = \mathbf{w}(n) \text{ if } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathcal{C}_1 \\ &\mathbf{w}(n+1) = \mathbf{w}(n) \text{ if } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathcal{C}_2 \end{aligned} \tag{3.54}$$

- Otherwise, weight vector of the perceptron is updated accordance to the rule

$$\begin{aligned} &\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n) \text{ if } \mathbf{w}^T(\mathbf{n})\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathcal{C}_2 \\ &\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n) \text{ if } \mathbf{w}^T(\mathbf{n})\mathbf{x}(n) \leq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathcal{C}_1 \end{aligned} \tag{3.55}$$

3.9 Perceptron Convergence Theorem

- Since the classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are assumed to be linearly separable, there exists a solution $\mathbf{w}_0$ for which $\mathbf{w}^T(\mathbf{n})\mathbf{x}(n) > 0$ for the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ belonging to the subset $\mathcal{H}_1$.
- For a fixed solution $\mathbf{w}_0$, we may then define a positive number a as

$$\alpha = \min_{\mathbf{x}(n)\in\mathcal{H}_1} \mathbf{w}_0^T\mathbf{x}(n) \tag{3.58}$$

From (3.57) we get

$$\mathbf{w}_0^T\mathbf{w}(n+1) = \mathbf{w}_0^T\mathbf{x}(1) + \mathbf{w}_0^T\mathbf{x}(2) + \cdots + \mathbf{w}_0^T\mathbf{x}(n) \geq n\alpha \tag{3.59}$$

- The Cauchy-Schwarz inequality, states that

$$\|\mathbf{w}_0\|^2\|\mathbf{w}(n+1)\|^2 \geq \|\mathbf{w}_0^T\mathbf{w}(n+1)\|^2 \tag{3.61}$$

- where $\|.\|$denotes Euclidean norm and inner product $\mathbf{w}_0^T\mathbf{w}(n+1)$ is a scalar.

3.9 Perceptron Convergence Theorem

- From (3.60), we note that,

$$\|\mathbf{w}_0\|^2\|\mathbf{w}(n+1)\|^2 \geq n^2\alpha^2 \quad \text{or} \quad \|\mathbf{w}(n+1)\|^2 \geq \frac{n^2\alpha^2}{\|\mathbf{w}_0\|^2} \tag{3.61}$$

- We can easily prove that for $\eta(n) = 1$ for all n and $\mathbf{w}(0) = \mathbf{0}$ and given that a solution vector $\mathbf{w}_0$ exists, the rule for adapting the synaptic weights of the perceptron must terminate after at most n_{max} iterations.
- Note also from Eqs. (3.58), (3.66), and (3.67) that there is no unique solution for $\mathbf{w}_0$ or n_{max}.

3.9 Perceptron Convergence Theorem

- The *fixed-increment convergence theorem* for the perceptron states that:

  Let the subsets of training vectors $\mathcal{H}_1$ and $\mathcal{H}_2$ be linearly separable. Let the inputs presented to the perceptron originate from these two subsets. The perceptron converges after some n_0 iterations, in the sense that

$\mathbf{w}(n_0) = \mathbf{w}(n_0 + 1) = \mathbf{w}(n_0 + 2) = \cdots$ is a solution vector for $n_0 \leq n_{max}$.

- In Table 3.2, a summary of the *perceptron convergence algorithm* is presented.
- sgn(·) in the table stands for the signum function:

$$\text{sgn}(v) = \begin{cases} +1 & \text{if } v > 0 \\ -1 & \text{if } v < 0 \end{cases} \tag{3.68}$$

- We may express the quantized response $y(n)$ of a perceptron in compact form

$$y(n) = \text{sgn}(\mathbf{w}^T(n)\mathbf{x}(n)) \tag{3.69}$$

Table 3.2 Summary of the Perceptron Convergence Algorithm

Variables and Parameters:

$\mathbf{x}(n)$ = $(m+1)$-by-1 input vector
$= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$
$\mathbf{w}(n)$ = $(m+1)$-by-1 weight vector
$= [b, w_1(n), w_2(n), \ldots, w_m(n)]^T$
b = bias
$y(n)$ = actual response (quantized)
$d(n)$ = desired response
η = learning-rate parameter, a positive constant less than unity

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time-step $n = 1, 2, \ldots$.
2. *Activation.* At time-step n, activate the perceptron by applying continuous-valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.
3. *Computation of Actual Response.* Compute the actual response of the perceptron as

$$y(n) = \text{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$

where sgn(·) is the signum function.

4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$

where

$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$

5. *Continuation.* Increment time step n by one and go back to step 2.

Table 3.2 Summary of the Perceptron Convergence Algorithm

Variables and Parameters:

$\mathbf{x}(n) = (m+1) - \text{by} - 1 \text{ input vector}$

$= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$

$\mathbf{w}(n) = (m+1) - \text{by} - 1 \text{ input vector}$

$= [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T$

$b(n) =$ bias

$y(n) =$ actual response (quantized)

$d(n) =$ desired response

$\eta =$ learning rate parameter

Table 3.2 Summary of the Perceptron Convergence Algorithm

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time step $n = 1, 2, \ldots$.
2. *Activation.* At time step n, activate the perceptron by applying continuous valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.
3. *Computation of Actual Response*. Compute the actual response of the perceptron:

$$y(n) = \text{sgn}(\mathbf{w}^T(n)\mathbf{x}(n))$$

where sgn(.) is the signum function.

4. *Adaptation of Weight Vector*. Update weight vector of perceptron:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta[d(n) - y(n)]\mathbf{x}(n)$$

where $d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ blongs to class } \mathcal{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ blongs to class } \mathcal{C}_2 \end{cases}$

5. *Continuation*. Increment time step n by one and go back to step 2.

3.10 Relation Between the Perceptron and Bayes Classifier for a Gaussian Environment

- When the environment is Gaussian, the **classical pattern classifier** known as the **Bayes Classifier** reduces to a linear classifier.
  - This is the same form taken by the perceptron.
- Let's have a look at this relationship and develop further insight into the operation of the perceptron with a brief review of the **Bayes Classifier**.
- In ***Bayes Classifier*** or ***Bayes Hypothesis Testing Procedure***, we minimize the average risk, denoted by $\mathcal{R}$.

Bayes Classifier

- For a two-class problem, represented by $\mathcal{C}_1$ and $\mathcal{C}_2$, ***average risk*** is given by

$$\begin{aligned}\mathcal{R} = c_{11}p_1 \int_{\mathcal{H}_1} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathcal{H}_2} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x} \\ + c_{21}p_1 \int_{\mathcal{H}_2} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} + c_{12}p_{12} \int_{\mathcal{H}_1} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x}\end{aligned} \tag{3.72}$$

where

p_i = *a priori probability* that observation vector $\mathbf{x}$ (representing a realization of the random vector $\mathbf{X}$) is drawn from subspace $\mathcal{H}_1$, $i = 1, 2$ and $p_1 + p_2 = 1$.

c_{ij}= cost of deciding in favor of class $\mathcal{C}_i$; represented by subspace $\mathcal{H}_i$; when class $\mathcal{C}_j$ is true (i.e., observation vector $\mathbf{x}$ is drawn from subspace $\mathcal{H}_j$ with $(i,j)=1,2$.

$f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_i)$ = conditional probability density function of the random vector $\mathbf{X}$, given that the observation vector $\mathbf{x}$ is drawn from subspace $\mathcal{H}_i$; with $i = 1, 2$.

Bayes Classifier

- The first two terms on r.h.s. of (3.72) represent correct decisions (i.e. correct classifications) while the last two terms represent incorrect decisions (i.e. misclassifications). From (3.72), the *minimum average risk* is given by:

$$\mathcal{R} = c_{21}p_1 + c_{22}p_2 + \int_{\mathcal{H}_1} [p_2(c_{12} - c_{22})f_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_2) - p_1(c_{21} - c_{11})f_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_1)]d\mathbf{x} \quad (3.76)$$

Bayes Classifier

- Since the requirement is to minimize the average risk $\mathcal{R}$, we may therefore deduce the following strategy from (3.76) for optimum classification:
  1. All values of observation vector $\mathbf{x}$ for which the integrand is negative should be assigned to subspace $\mathcal{H}_1$ (i.e. class $\mathcal{C}_1$) for the integral would then make a negative contribution to the risk $\mathcal{R}$.
  2. All values of the observation vector $\mathbf{x}$ for which the integrand is positive should be excluded from subspace $\mathcal{H}_1$ (i.e., assigned to class $\mathcal{C}_2$) for the integral would then make a positive contribution to the risk m.
  3. Values of $\mathbf{x}$ for which the integrand=0 have no effect on the average risk $\mathcal{R}$ and may be assigned arbitrarily. We shall assume that these points are assigned to subspace $\mathcal{H}_2$ (i.e. class $\mathcal{C}_2$).

Bayes Classifier

- On this basis, we may formulate the Bayes classifier we define:

$$\Lambda(\mathbf{x}) = \frac{f_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_1)}{f_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_2)} \tag{3.77}$$

and

$$\xi = \frac{p_2(c_{12}-c_{22})}{p_1(c_{21}-c_{11})} \tag{3.78}$$

- The always +ve quantity $\Lambda(\mathbf{x})$, the ratio of two conditional probability density functions, is called the *likelihood ratio*.
- The always +ve quantity ξ is called the *threshold* of the test. Note that both
- Finally, the Bayes Classifier states that:
- If, for an observation vector $\mathbf{x}$, the likelihood ratio $\Lambda(\mathbf{x})$ is greater than the threshold ξ, assign $\mathbf{x}$ to class $\mathcal{C}_1$. Otherwise, assign it to class $\mathcal{C}_2$.
- Figure 3.10a depicts a block-diagram representation of the Bayes classifier.

Bayes Classifier

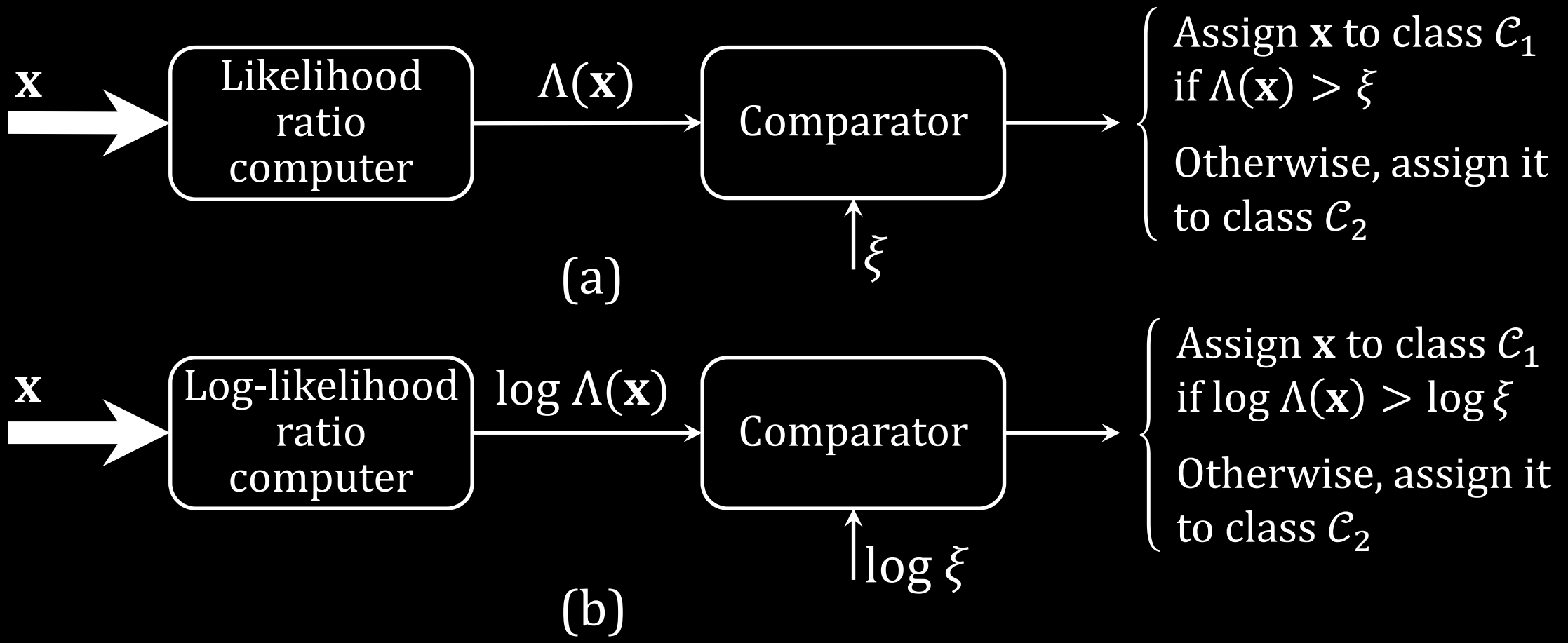


Fig. 3.10 Two equivalent implementations of the Bayes Classifier: (a) Likelihood ratio test, (b) Log-likelihood ratio test.

Bayes Classifier for a Gaussian Distribution

- Considering the special case of a two-class problem, for which the underlying distribution is Gaussian.
- The random vector $\mathbf{X}$ has a mean value that depends on whether it belongs to class $\mathcal{C}_1$ or class $\mathcal{C}_2$, but covariance matrix $\mathbf{C}$ of $\mathbf{X}$ is same for both classes, i.e.

Class $\mathcal{C}_1$: $E[\mathbf{X}] = \boldsymbol{\mu}_1$

$$E[(\mathbf{X} - \boldsymbol{\mu}_1)(\mathbf{X} - \boldsymbol{\mu}_1)^T] = \mathbf{C}$$

Class $\mathcal{C}_2$: $E[\mathbf{X}] = \boldsymbol{\mu}_2$

$$E[(\mathbf{X} - \boldsymbol{\mu}_2)(\mathbf{X} - \boldsymbol{\mu}_2)^T] = \mathbf{C}$$

- The covariance matrix $\mathbf{C}$ is diagonal, which means samples drawn from class $\mathcal{C}_1$ and $\mathcal{C}_2$ are *correlated* and its inverse $\mathbf{C}^{-1}$ exists.

Bayes Classifier for a Gaussian Distribution

- The conditional probability density function of $\mathbf{X}$ is:

$$f_{\mathbf{x}}(\mathbf{x}|\mathcal{C}_i) = \frac{1}{(2\pi)^{m/2}(\det(C))^{m/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right), \quad i = 1, 2 \qquad (3.79)$$

- It is further assumed that

1. The two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are equiprobable

$$p_1 = p_2 = 1/2 \qquad (3.80)$$

1. Misclassifications carry the same cost, and no cost is incurred on correct classifications:

$$c_{21} = c_{12} \qquad \text{and} \qquad \mathrm{c}_{11} = \mathrm{c}_{22} = 0 \qquad (3.81)$$

Bayes Classifier for a Gaussian Distribution

- With this, we need to design Bayes classifier for the two class problem, we get

$$\log \Lambda(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)$$

$$= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{C}^{-1}\mathbf{x} + \frac{1}{2}(\boldsymbol{\mu}_2^T\mathbf{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\mathbf{C}^{-1}\boldsymbol{\mu}_1) \quad (3.82)$$

and

$$\log \xi = 0 \quad (3.83)$$

- From (3.82) and (3.83), the Bayes classifier becomes

$$y = \mathbf{w}^T\mathbf{x} + b \quad (3.84)$$

$$\mathbf{w} = \mathbf{C}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T \quad (3.86)$$

$$b = \frac{1}{2}(\boldsymbol{\mu}_2^T\mathbf{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\mathbf{C}^{-1}\boldsymbol{\mu}_1) \quad (3.87)$$

- Specifically, the classifier consists of a linear combiner with weight vector $\mathbf{w}$ and bias b, as shown in Fig. 3.11.

Bayes Classifier for a Gaussian Distribution

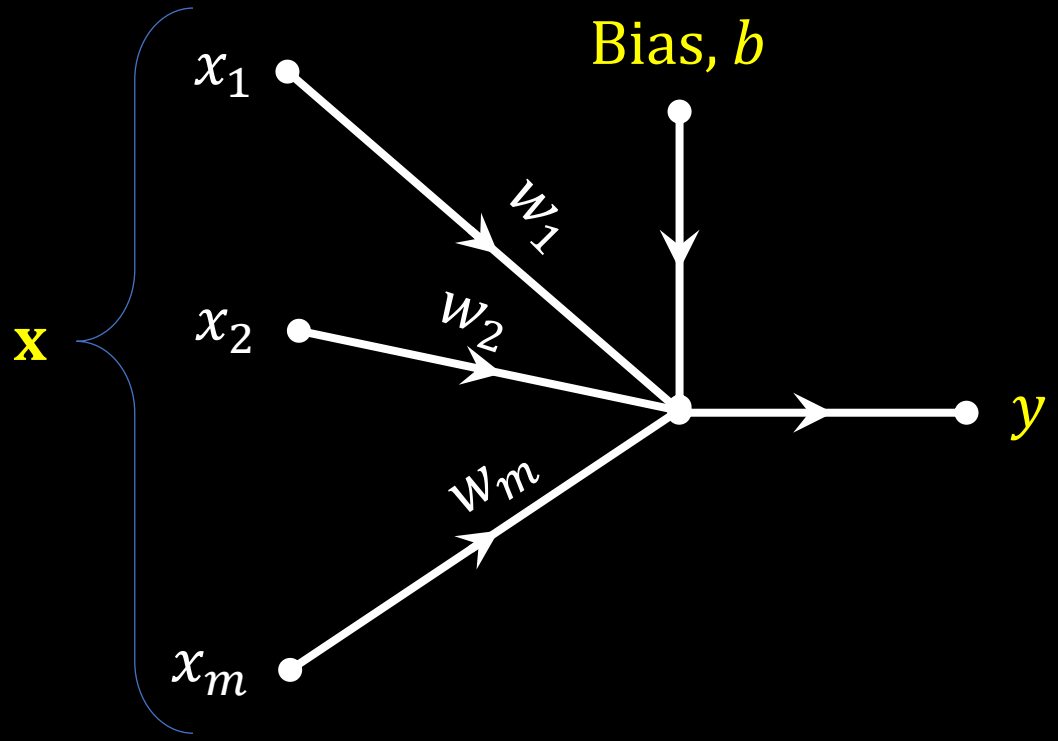


Fig. 3.11 Signal-flow graph of Gaussian classifier.

Bayes Classifier for a Gaussian Distribution

- From (3.84), the log-likelihood ratio test for the two-class problem is:

  If the output y of the linear combiner is positive, assign the observation vector $\mathbf{x}$ *to class* $\mathcal{C}_1$. *Otherwise, assign it to class* $\mathcal{C}_2$.

- *Based on* (3.71) and (3.84), we observe some important differences between the Bayes classifier for the Gaussian environment and the perceptron as follows:

  ~ In the perceptron operation, the patterns to be classified are linearly separable. The Bayes classifier overlap each other and are therefore not separable. The nature of this overlap is illustrated in Fig. 3.12 for the special case of a scalar random variable (i.e., dimensionality $m = 1$).

Bayes Classifier for a Gaussian Distribution

~ The Bayes classifier minimizes the probability of classification error. For example, in the special case illustrated in Fig. 3.12, the Bayes classifier always positions the decision boundary at the point where the Gaussian distributions for the two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ cross each other.

~ The perceptron convergence algorithm is nonparametric in the sense that it makes no assumptions concerning the form of the underlying distributions. In contrast, the Bayes classifier is parametric

~ The perceptron convergence algorithm is both adaptive and simple to implement; its storage requirement is confined to the set of synaptic weights and bias. On the other hand, the design of the Bayes classifier is fixed; it can be made adaptive, but at the expense of increased storage requirements and more complex computations.

Bayes Classifier for a Gaussian Distribution

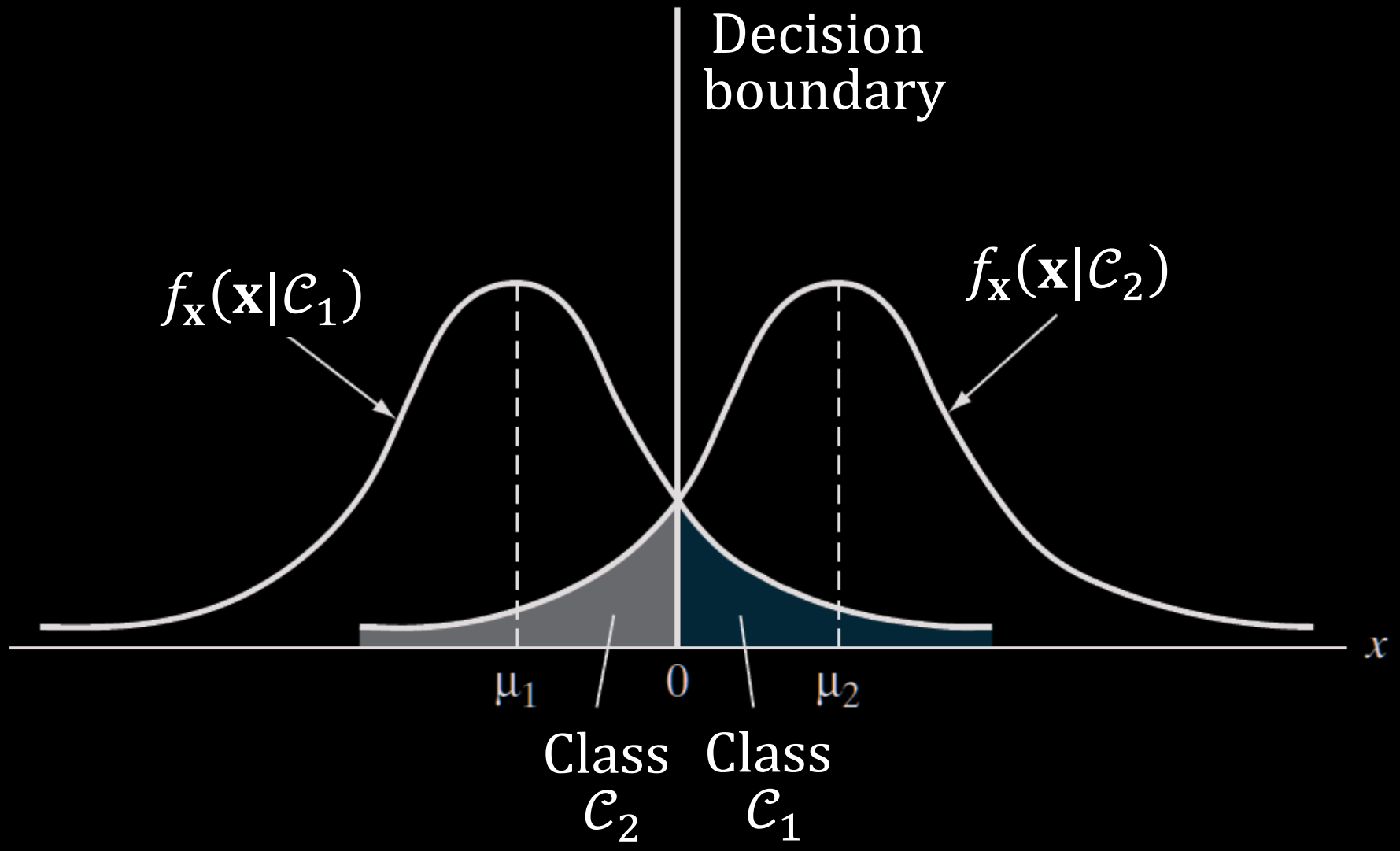


Fig. 3.12 Two overlapping, one-dimensional Gaussian distributions.

Bayes Classifier for a Gaussian Distribution

End of Chapter 3