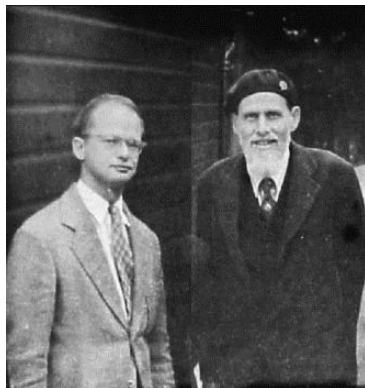


第二章

感知機

2.1 簡介

- 在1943年，McCulloch (neurophysiologist) 和 Pitts (logician) 已提出第一個類神經元的運算模型。
- 神經心理學家 Hebb (psychologist) 提出一種理論，他認為學習現象的發生，乃在於神經元間的突觸產生某種變化。
- Rosenblatt (psychologist) 將這兩種創新學說結合起來，孕育出所謂的感知機 (perceptron)。
- 感知機是由具有可調整的鍵結值 (synaptic weights) 以及閾值 (threshold) 的單一個類神經元 (neuron) 所組成。
- 感知機是各種類神經網路中，最簡單且最早發展出來的類神經網路模型，通常被用來做為分類器 (classifier) 使用。



Pitts 和 McCulloch



Hebb



Frank Rosenblatt

2.2 感知機基本架構 (1)

- 感知機的基本組成元件為一個具有線性組合功能的累加器，後接一個硬限制器 (hard limiter) 而成，如圖 2.1 所示。
- 一般說來，若我們設定硬限制器之輸入為正值時，則類神經元的輸出為 +1；反之，若硬限制器之輸入為負值時，則類神經元的輸出為 -1。

$$u_j = \sum_{i=1}^p w_{ji} x_i$$

$$v_j = u_j - \theta_j$$

$$v_j = \sum_{i=0}^p w_{ji} x_i (= \underline{w}_j^T \underline{x})$$

$$y_j = \varphi(v_j)$$

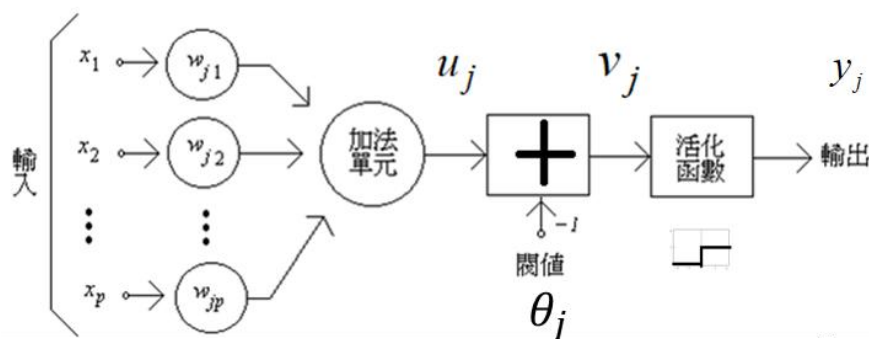


圖2.1：感知機之架構方塊。 $\underline{w}_j = \begin{pmatrix} w_{j0} \\ w_{j1} \\ \vdots \\ w_{jp} \end{pmatrix}$ $\underline{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{pmatrix}$

2.2 感知機基本架構 (2)

- 分類的判斷規則是：若感知機的輸出為 +1，則將其歸類於 C_1 群類；若感知機的輸出為 -1，則將其歸類於 C_2 群類。
- 判斷規則所劃分的只有兩個判斷區域，我們可以將作為分類依據的超平面定義如下：

$$\sum_{i=1}^p w_i x_i - \theta = 0 \quad (2.2)$$

$$y_j = \phi(v_j)$$

$$\phi(v) = \begin{cases} +1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}$$

$$v_j = \sum_{i=1}^p w_{ji} x_i - w_{j0} = \underline{w}_j^T \underline{x} - w_{j0}$$

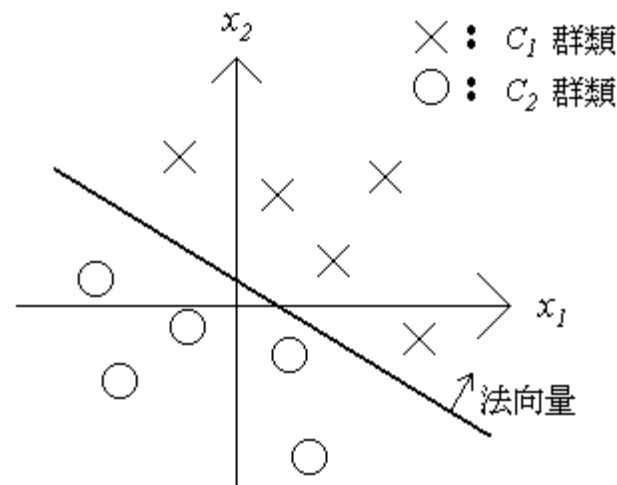


圖2.2：一個具有二維輸入的兩群分類問題。

2.3 感知機收斂定理 (1)

- 步驟一：網路初始化

以隨機的方式來產生亂數，令 $\underline{w}(0)$ 為很小的實數，並且將學習循環 n 設定為 1。

- 步驟二：計算網路輸出值

在第 n 次學習循環時，呈現輸入向量 $\underline{x}(n)$ ，此時類神經元的輸出為：

$$y(n) = \varphi[\underline{w}^T(n)\underline{x}(n)]$$

$$\phi(v) = \begin{cases} +1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}$$

- 步驟三：調整鍵結值向量

- $$\underline{w}(n+1) = \begin{cases} \underline{w}(n) + \eta \underline{x}(n) & \text{如果 } \underline{x}(n) \in C_1 \text{ 和 } \underline{w}^T(n)\underline{x}(n) < 0 \\ \underline{w}(n) - \eta \underline{x}(n) & \text{如果 } \underline{x}(n) \in C_2 \text{ 和 } \underline{w}^T(n)\underline{x}(n) \geq 0 \\ \underline{w}(n) & \text{如果 } \underline{x}(n) \text{ 被正確分類} \end{cases} \quad (2.11)$$

- 步驟四：

將學習循環 n 加 1；回到步驟二。

2.3 感知機收斂定理 (2)

- 調整後的類神經元，若再用相同的輸入範例來加以測試，則其輸出值會更加地接近期待值，其原因如下：

$$\begin{aligned}\underline{w}^T(n+1)\underline{x}(n) &= (\underline{w}(n) \mp \eta \underline{x}(n))^T \underline{x}(n) \\ &= \underline{w}^T(n)\underline{x}(n) \mp \eta \underline{x}^T(n)\underline{x}(n)\end{aligned}$$

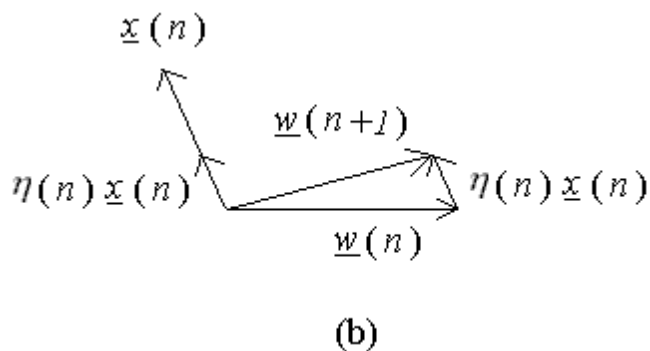
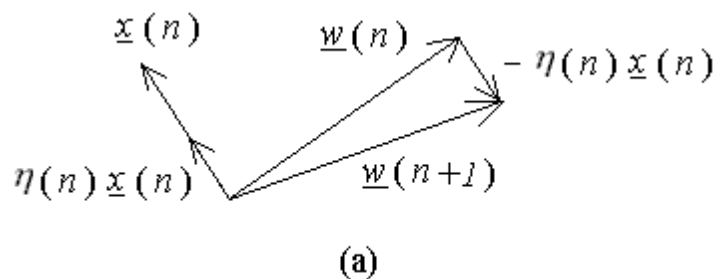


圖2.4：以幾何觀點來分析感知機之訓練過程；
(a) 若 $d = -1$ 且 $y = 1$ ；(b) 若 $d = 1$ 且 $y = -1$ 。

範例2.1：感知機 (1)

訓練資料 $\underline{x}(n)$	期望輸出值 $d(n)$
$(0, 0)^T$	1
$(0, 1)^T$	1
$(1, 0)^T$	-1
$(1, 1)^T$	1

學習率 η 為 0.8，並且將鍵結值的初始值設定為 $(0, 1)$ ，
令活化函數為sgn函數，神經元之閾值為-1

(1) $n=0$

$$y(0) = \text{sgn}[\underline{w}^T(0)\underline{x}(0)] = \text{sgn}\left[(-1, 0, 1) \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}\right] = \text{sgn}[1] = 1$$
$$\underline{w}(1) = \underline{w}(0) = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

$$(1) \ n = 1 \ ,$$

$$y(1) = \text{sgn}[\underline{w}^T(1)\underline{x}(1)] = \text{sgn}\left[(-1,0,1)\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}\right] = \text{sgn}[2] = 1$$

$$\underline{w}(2) = \underline{w}(1) = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

$$(2) \ n = 2 \ ,$$

$$y(2) = \text{sgn}[\underline{w}^T(2)\underline{x}(2)] = \text{sgn}\left[(-1,0,1)\begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}\right] = \text{sgn}[1] = 1$$

$$\underline{w}(3) = \underline{w}(2) - \eta \underline{x}(2) = \begin{pmatrix} -0.2 \\ -0.8 \\ 1 \end{pmatrix}$$

$$(3) \ n = 3 \ ,$$

$$y(3) = \text{sgn}[\underline{w}^T(3)\underline{x}(3)] = \text{sgn}\left[(-0.2,-0.8,1)\begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}\right] = \text{sgn}[0.4] = 1$$

$$\underline{w}(4) = \underline{w}(3) = \begin{pmatrix} -0.2 \\ -0.8 \\ 1 \end{pmatrix}$$

$$(4) \ n = 4 \ ,$$

$$y(4) = \text{sgn}[\underline{w}^T(4)\underline{x}(4)] = \text{sgn}\left[(-0.2,-0.8,1)\begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}\right] = \text{sgn}[0.2] = 1$$

$$\underline{w}(5) = \underline{w}(4) = \begin{pmatrix} -0.2 \\ -0.8 \\ 1 \end{pmatrix}$$

(5) $n = 5$,

$$y(5) = \text{sgn}[\underline{w}^T(5)\underline{x}(1)] = \text{sgn}\left[(-0.2, -0.8, 1) \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}\right] = \text{sgn}[1.2] = 1$$

$$\underline{w}(6) = \underline{w}(5) = \begin{pmatrix} -0.2 \\ -0.8 \\ 1 \end{pmatrix}$$

(6) $n = 6$,

$$y(6) = \text{sgn}[\underline{w}^T(6)\underline{x}(2)] = \text{sgn}\left[(-0.2, -0.8, 1) \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}\right] = \text{sgn}[-0.6] = -1$$

$$\underline{w}(7) = \underline{w}(6) = \begin{pmatrix} -0.2 \\ -0.8 \\ 1 \end{pmatrix}$$

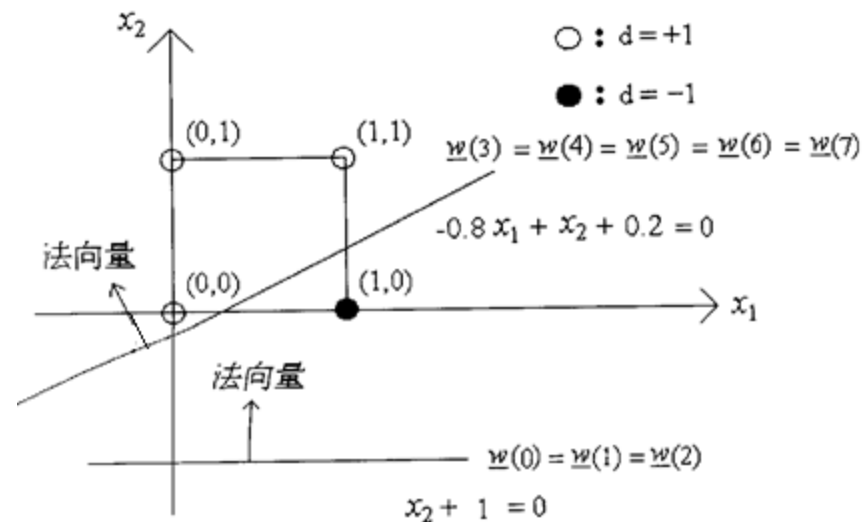


圖 2.5：感知機於疊代次數第 4, 5, 8 次時的鍵結值向量。

範例2.1：感知機 (2)

- 假設當初學習率 η 設定為 0.1，那麼的修正量會不同：

$$\underline{w}(3) = \underline{w}(2) - \eta \underline{x}(2) = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.9 \\ -0.1 \\ 1 \end{pmatrix}$$

- 此時若將相同的輸入測試此新的鍵結值向量，會發現期望輸出值為 -1，但感知機輸出值為 1，分類錯誤，因此仍需修正感知機的鍵結值向量。
- 也就是說，感知機的學習過程並不保證一次就學會；有時更會發生為了學新的輸入卻將原先已正確分類的資料給誤判了。

2.4 Widrow-Hoff 法則 (1)

- 此種由 Widrow 和 Hoff 於1960年提出訓練所謂“適應線性元件” (Adaptive Linear Element 簡稱為 Adaline) 的學習規則，被稱做 Widrow-Hoff 法則或最小均方誤差法。
- 基本上 Adaline 和感知機的架構是一樣的，主要的差別在於訓練法則的不同，感知機的訓練目標是減少分類錯誤，而 Adaline 是減少類神經元的真實輸出與期望輸出間的均

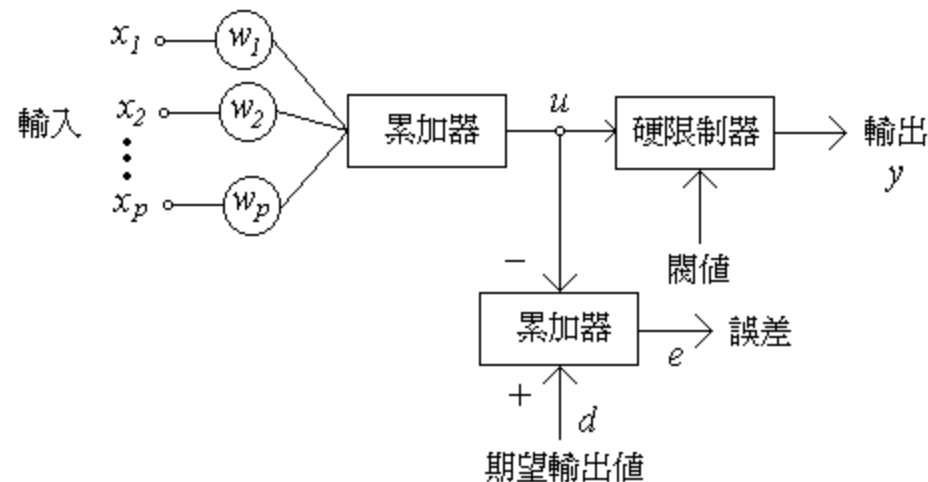


圖2.6：Adaline 之架構方塊圖。

2.4 Widrow-Hoff 法則 (2)

- 給定一組輸入/輸出對， $(x_i, d_i), i = 1, 2, \dots, N$ ， d_i 代表網路的期望輸出值。
- 用來訓練 Adaline 的 LMS 法則就是想辦法找出一個鍵結值向量 \underline{w}^* ，它可以使得誤差 $e_j = d_j - u_j$ 的均方值 (mean square) 最小，其中 $u_j = \sum_{i=1}^p w_i x_i = \underline{w}^T \underline{x}$ 。這個問題的答案在於所謂的 Wiener-Hoff 方程式。
- 定義均方誤差 (mean-square error) 為：

$$\begin{aligned} J &= \frac{1}{2} E[e^2] \\ &= \frac{1}{2} E[(d - u)^2] \end{aligned} \tag{2.17}$$

- 其中 $E[.]$ 是機率期望值運算子。

2.4 Widrow-Hoff 法則 (3)

- 首先將 $u_k = \sum_{k=1}^p w_k x_k$ 代入式(2.17)並且展開可得：

$$J = \frac{1}{2} E[d^2] - \sum_{k=1}^p w_k E[x_k d] + \frac{1}{2} \sum_{j=1}^p \sum_{k=1}^p w_j w_k E[x_j x_k] \quad (2.18)$$

- 我們另外定義式(2.19)中的三個期望值運算如下：

1. $E[d^2]$ 是期望輸出 d 的「均方值」(mean-square value)；令

$$r_d = E[d^2] \quad (2.20)$$

2. $E[dx_k]$ 是期望輸出 d 與輸入信號 x_k 的「交互相關函數」(cross-correlation function)；令

$$r_{dx}(k) = E[dx_k], k = 1, 2, \dots, p \quad (2.21)$$

3. $E[x_j x_k]$ 是輸入信號 x_j 與 x_k 彼此間的「自相關函數」(autocorrelation function)；令

$$r_x(j, k) = E[x_j x_k], j, k = 1, 2, \dots, p \quad (2.22)$$

2.4 Widrow-Hoff 法則 (4)

- 有了以上三個式子的定義，現在我們把式(2.19)重寫如下：

$$J = \frac{1}{2} r_d - \sum_{k=1}^p w_k r_{dx}(k) + \frac{1}{2} \sum_{j=1}^p \sum_{k=1}^p w_j w_k r_x(j, k) \quad (2.23)$$

- 我們將誤差函數 J 對權重 作偏微分，並將此偏微分方程式設為零，因此我們可以得到以下 p 個方程式

$$\begin{aligned} \nabla_{w_k} J &= \frac{\partial J}{\partial w_k} \\ &= -r_{dx}(k) + \sum_{j=1}^p w_j r_x(j, k) = 0, k = 1, 2, \dots, p \end{aligned} \quad (2.24)$$

- 令 w_k^* 為權重 w_k 的最佳值，從式(2.24)可知權重參數的最佳值是由下列 p 個方程式所決定：

$$\sum_{j=1}^p w_j^* r_x(j, k) = r_{xd}(k), k = 1, 2, \dots, p \quad (2.25)$$

- 這組方程式稱為「Wiener-Hopf 方程式」，而根據此權重參數所設計的濾波器則稱之為 Wiener 濾波器。

2.4 Widrow-Hoff 法則 (5)

- 式 (2.25) 其實可以寫成如下的矩陣型式：

$$R\underline{w}^* = P \quad (2.26)$$

- 其中 $\underline{w}^* = [w_1^*, w_2^*, \dots, w_p^*]^T$, $R = E[\underline{x} \cdot \underline{x}^T] = [r_x(j, k)]$ 是一個 $p \times p$ 維度的自相關矩陣，和 $P = E[d x_k] = [r_{dx}(k)]$ 是一個 $p \times 1$ 維度的交互相關向量。
- 很顯然地，Wiener-Hoff 方程式的解為

$$\underline{W}^* = R^{-1}P \quad (2.27)$$

- 問題是，我們通常不知道自相關矩陣與交互相關向量的值是多少。即使知道，我們每次都必須計算出 R 的反矩陣，而求反矩陣的計算量相當龐大，因此實際上的作法是採用 LMS 演算法，以疊代的方式來解 Wiener-Hoff 方程式。

2.4 Widrow-Hoff 法則 (6)

- 當我們無法得知自相關函數 與交互相關函數 的資訊時，以「瞬間估測」(instantaneous estimates) 的技術來估測這兩個相關函數，並且使用梯度坡降法來疊代求解。
- 從式 (2.21) 與式 (2.22) 可以導出：

$$\hat{r}_{dx}(k;n) = x_k(n)d(n) \quad (2.28)$$

$$\hat{r}_x(j,k;n) = x_j(n)x_k(n) \quad (2.29)$$

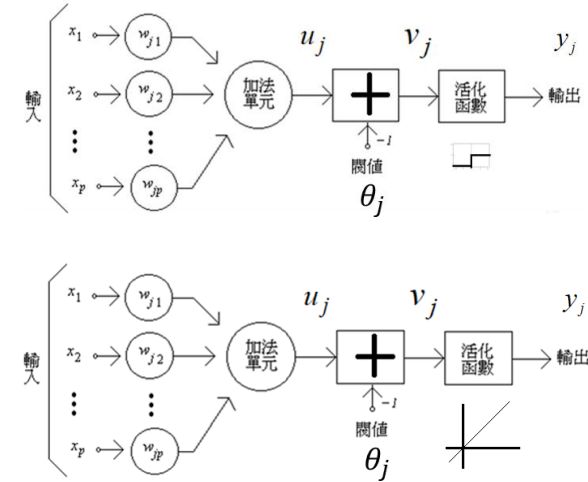
2.4.1 最小均方演算法 (1)

- 代價函數則定義為：

$$e_j(n) = d_j(n) - v_j(n)(d_j(n) - y_j(n))$$

$$E = \frac{1}{2} \left(e_j(n) \right)^2 = \frac{1}{2} \left(d_j(n) - v_j(n) \right)^2$$

- $$v_j(n) = \sum_{i=0}^p w_{ji}(n) x_i(n)$$
- 訓練目標：找到一組參數 $w_{ji}(n)$ 可讓代價函數極小化。

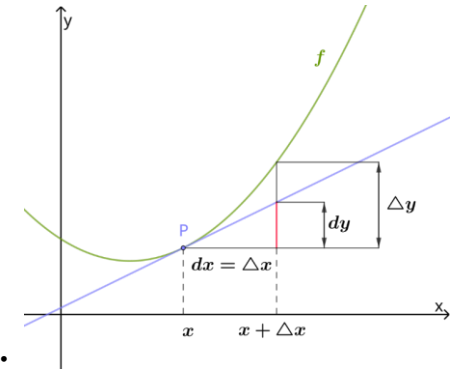


$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n)$$

2.4.1 最小均方演算法 (2)

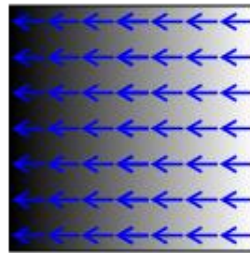
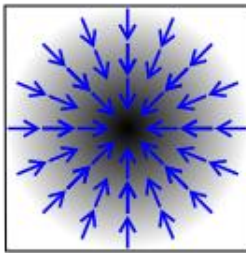
- **梯度：**
- 一維（單變數）函數的梯度就是該函數的微分，是純量值函數：

$$f'(x_0) = \lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

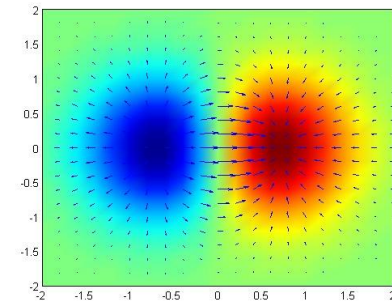


- 多維函數的梯度是由偏微分為分量組成的向量值函數：

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) = \frac{\partial f}{\partial x} \mathbf{i} + \frac{\partial f}{\partial y} \mathbf{j} + \frac{\partial f}{\partial z} \mathbf{k}$$



$$F_1(x, y) = -(x^2 + y^2), \quad F_2(x, y) = -x$$



純量場的值用灰度表示，越暗表示越大的數值，而其相應的梯度用藍色箭頭表示。

函數 $f(x, y) = xe^{-(x^2 + y^2)}$ 的梯度繪製為藍色箭頭

2.4.1 最小均方演算法 (3)

- LMS 法則使用梯度坡降法，所以權重的調整方向是往誤差梯度的反方向修正，計算方式如下：

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) = w_{ji}(n) - \eta \frac{\partial E}{\partial w_{ji}(n)}$$

$$E = \frac{1}{2} \left(e_j(n) \right)^2 = \frac{1}{2} \left(d_j(n) - v_j(n) \right)^2$$

$$\begin{aligned} \Delta w_{ji}(n) &= -\eta \frac{\partial E}{\partial w_{ji}(n)} \\ &= \eta \left(d_j(n) - \sum_{i=0}^p w_{ji}(n) x_i(n) \right) x_i(n) \\ &= \eta \left(d_j(n) - v_j(n) \right) x_i(n) \end{aligned}$$

2.4.1 最小均方演算法 (4)

- 步驟一：權重初始化，令

$$w_{ji}(0) = \text{rand()} \quad i = 0, 1, \dots, p$$

- 步驟二：設定學習循環 $n = 1$ ，並且計算

$$v_j(n) = \sum_{i=0}^p w_{ji}(n)x_i(n)$$

$$e(n) = d_j(n) - v_j(n)$$

$$w_{ji}(n+1) = w_{ji}(n) + \eta e(n)x_i(n)$$

- 步驟三：將學習循環加 1，回到步驟二。

2.4.1 最小均方演算法 (5)

我們將「感知機訓練法」與「最小均方演算法」比較如下：

1. 感知機訓練演繹法主要的目標是調整鍵結值，使得類神經元的輸出 與期望輸出值 相同，亦即分類成功；而 LMS 演繹法強調於如何找出一組鍵結值，使得類神經元的軸突丘細胞膜電位 儘可能地接近 期望值。
2. 在輸入資料為線性不可分割 (linearly inseparable) 的情況下，LMS 演繹法有可能表現得比感知機演繹法還要好，因為前者會收斂至均方誤差的最小值，而後者卻無法收斂，所以訓練過程被中斷後，我們無法保證最後的鍵結值是整個訓練過程中最好的一組鍵結值。
3. 在輸入資料為線性可分割 (linearly separable) 時，感知機演繹法可以保證收斂，並且可以百分之百分類成功；而 LMS 演繹法雖然會收斂，但是卻並不一定可以百分之百分類成功，因為它主要著眼於將均方誤差極小化。

2.4.1 最小均方演算法 (6)

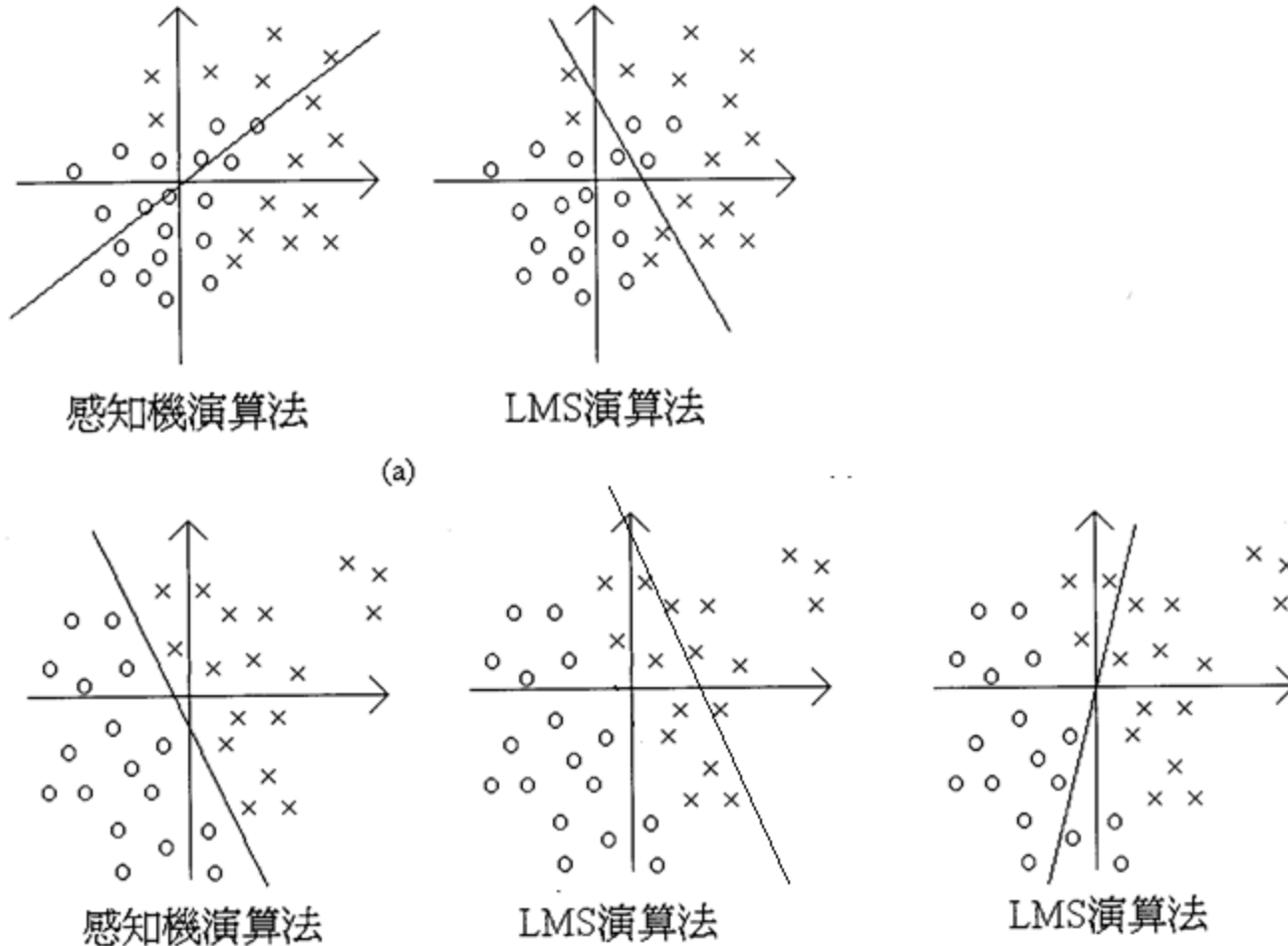


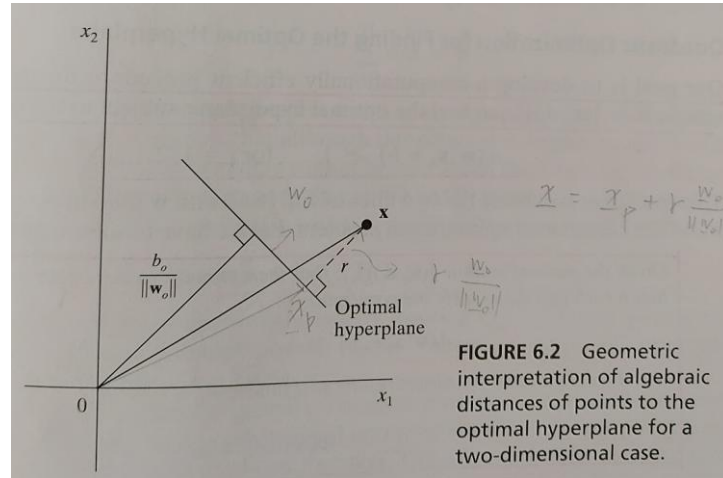
圖2.7：(a) 感知機演繹法與 (b) LMS 演繹法的收斂情形。

2.4.1 最小均方演算法 (7)

$$g(\underline{x}) = v_j = \sum_{i=0}^p w_{ji} x_i = \sum_{i=1}^p w_{ji} x_i - \theta_j = \underline{w}_j^T \underline{x} - \theta_j = r \|\underline{w}_j\|$$

$$r = \frac{g(\underline{x})}{\|\underline{w}_j\|}$$

$$\underline{x} = \underline{x}_p + r \frac{\underline{w}_j}{\|\underline{w}_j\|}$$

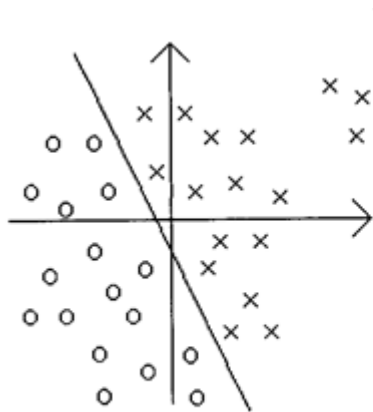


$$g(\underline{x}) = \underline{w}_j^T \underline{x} - \theta_j = \underline{w}_j^T \left(\underline{x}_p + r \frac{\underline{w}_j}{\|\underline{w}_j\|} \right) - \theta_j$$

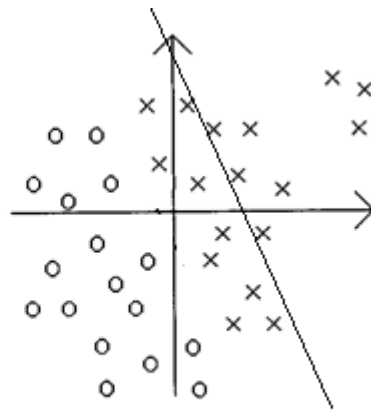
$$= \underline{w}_j^T \underline{x}_p + r \underline{w}_j^T \frac{\underline{w}_j}{\|\underline{w}_j\|} - \theta_j = (\underline{w}_j^T \underline{x}_p - \theta_j) + r \frac{\|\underline{w}_j\|^2}{\|\underline{w}_j\|} = r \|\underline{w}_j\|$$

2.4.1 最小均方演算法 (8)

$$\begin{aligned} E &= \frac{1}{2} \left(e_j(n) \right)^2 = \frac{1}{2} \left(d_j(n) - v_j(n) \right)^2 \\ &= \frac{1}{2} \left(d_j(n) - r \|\underline{w}_j\| \right)^2 \end{aligned}$$



感知機演算法



LMS演算法

2.5 學習率調整方法 (1)

- 應用以梯度坡降法為根本的演算法時，最常遇到的問題就是，如何適當地設定學習率。

$$\begin{aligned} E &= \frac{1}{2} \left(e_j(n) \right)^2 = \frac{1}{2} \left(d_j(n) - v_j(n) \right)^2 = \frac{1}{2} \left(d_j(n) - \sum_{i=0}^p w_{ji}(n) x_i(n) \right)^2 \\ &= \frac{1}{2} d_j^2(n) - d_j(n) \sum_{i=0}^p w_{ji}(n) x_i(n) + \frac{1}{2} \left(\sum_{i=0}^p w_{ji}(n) x_i(n) \right)^2 \end{aligned}$$

- 我們發現均方誤差函數， E ，對每一個鍵結值來說都是個二次方程式。

2.5 學習率調整方法 (2)

- 此二次方程式會形成如圖 2.8 的拋物線函數，也就是說，均方誤差函數對鍵結值來說只有一個全域極小值存在，所以不會有收斂到局部極小值的問題產生，但學習率的大小設定仍會影響能否收斂到全域極小值的結果。

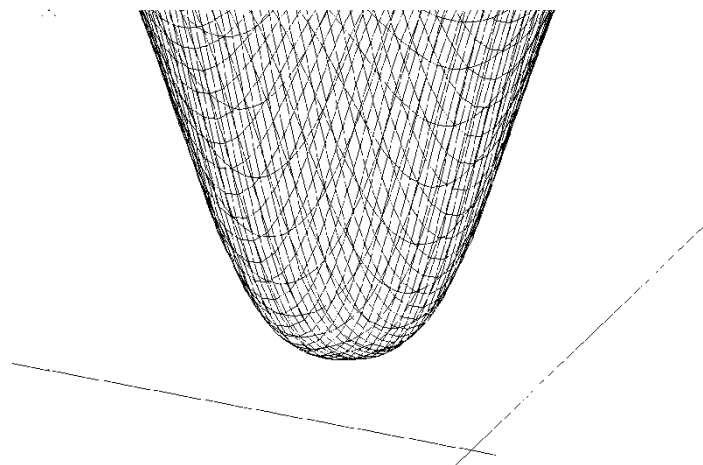
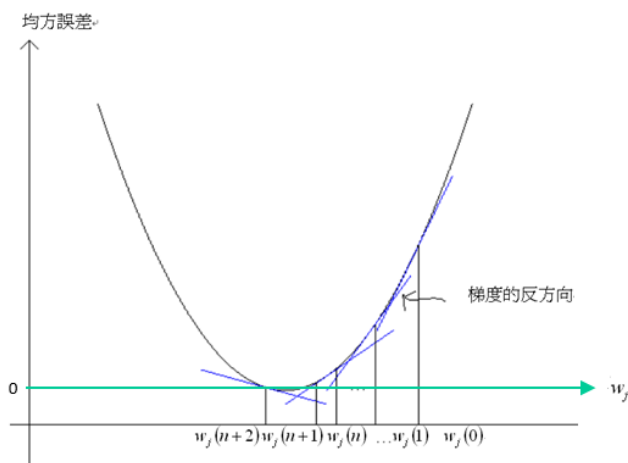


圖 2.8：二次方程式的均方誤差函數。

2.5 學習率調整方法 (3)

$$\eta(n) = \eta_0 \qquad \eta(n) = \frac{c}{n} \qquad \eta(n) = \frac{\eta_0}{1 + (n / \tau)}$$

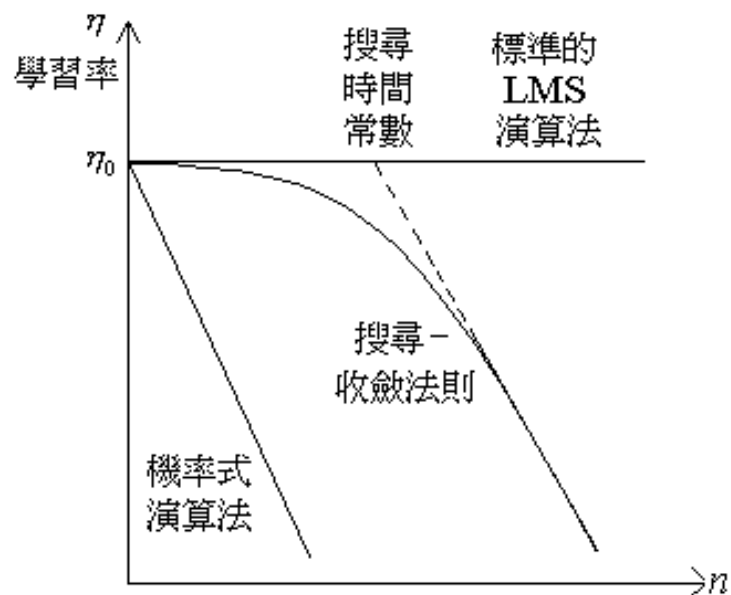


圖2.8：三種設定學習率的方式。

2.6 感知機之進階探討 (1)

Gallant所提出的“口袋演繹法”(pocket algorithm)。

步驟一：設定鍵結值向量的初始值為 $\underline{0}$ 向量。

步驟二：隨機地選擇一個訓練樣本 $\underline{x}(n)$ ，並且輸入至感知機網路中。

步驟三：

(a) 如果目前的鍵結值向量 能夠將訓練樣本 正確地分類成功，且能夠正確分類的訓練樣本數，大於口袋中的那一組鍵結值向量

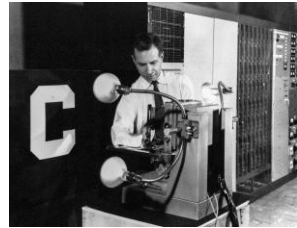
$\underline{w}^*(n)$ 所能夠正確分類的訓練樣本數，則以 $\underline{w}(n)$ 來取代 $\underline{w}^*(n)$ ，並且更正目前所能正確分類的訓練樣本數目。

(b) 如果目前的鍵結值向量 $\underline{w}(n)$ 將訓練樣本 $\underline{x}(n)$ 分類錯誤，則修正目前的鍵結值向量 為：

$$\begin{cases} \underline{w}(n+1) = \underline{w}(n) - \eta(n)\underline{x}(n) & \text{若 } \underline{w}^T(n)\underline{x}(n) \geq 0 \text{ 且 } \underline{x}(n) \in C_2 \\ \underline{w}(n+1) = \underline{w}(n) + \eta(n)\underline{x}(n) & \text{若 } \underline{w}^T(n)\underline{x}(n) < 0 \text{ 且 } \underline{x}(n) \in C_1 \end{cases}$$

步驟四：回到步驟二。

2.6 感知機之進階探討 (2)



- 理論上，感知機是可以用來解決任何複雜的圖樣識別的問題，只是當時沒有一個有效率的訓練演繹法來配合而已。
- 單層的感知機是用超平面來切割空間，因此只能處理線性可分割的資料。
- 雙層感知機可以處理較複雜的問題，譬如說我們限制第二層的感知機只執行邏輯 **AND** 的功能，那麼第一層的感知機便可合力圍起一個凸形 (**convex**) 的決定區域。
- 三層的感知機則可以形成任意形狀的決定區域，譬如說第二層的感知機執行邏輯 **AND** 的功能，而第三層的感知機則執行邏輯 **OR** 的功能，那麼不管是凸形、凹形 (**concave**) 的決定區域都可以輕易地形成。

2.6 感知機之進階探討 (3)

- 當時沒有一種有效的學習演算法可以配套使用。
- 之所以需要三層的架構，是因為我們限制了感知機的活化函數是採用硬限制器，以及限定第二層和第三層的類神經元，分別執行 AND 與 OR 的功能。


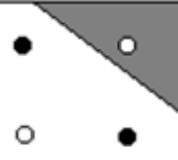

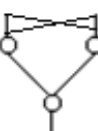
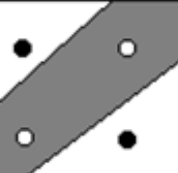




網路架構	決定區域之形狀	決定區域 XOR 問題	決定區域之形狀(通式)
	以超平面為界之半平面		
	凸形之開區間或閉區間		
	任意形狀		

圖2.9：感知機的決定區域之收斂特性分析。

2.7 結語

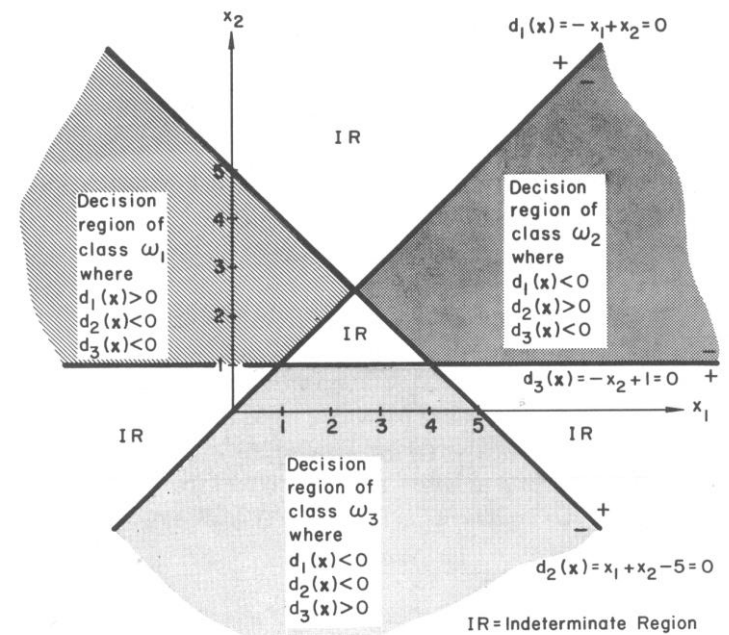
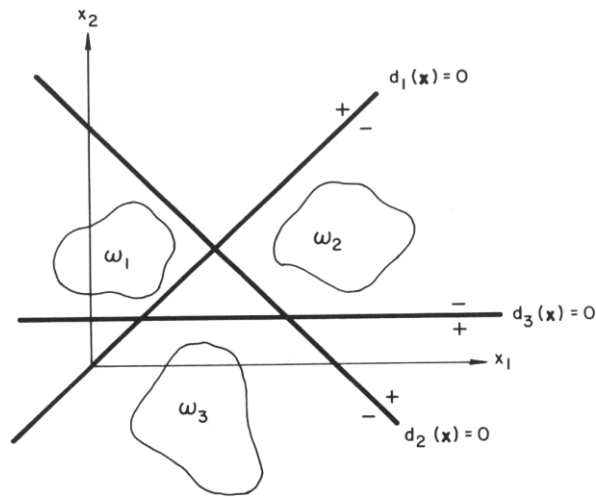
- 一個單層的感知機架構可以百分之百地將線性可分割的資料正確分類，但對於線性不可分割的資料而言，卻無法百分之百成功辨識。
- 為了達成此目地，多層的感知機架構是個變通的方法，問題是在 60 年代的當時，並沒有一個較理想的訓練演繹法可以訓練多層的感知機，由於實際上我們遭遇到問題通常是線性不可分割的資料，因此感知機的能力便被懷疑，導至研究的中斷。
- 直到 80 年代才被發現，由於這個演繹法的出現，使得類神經網路的研究又再度蓬勃發展，我們將在下一章探討此演繹法。

附錄

多類別分類 (1)

- 多類別分類問題：
 - 期望輸出如何設定？
 - 幾個感知機？
- 直覺方式： M 類別的分類問題就用 M 個感知機。
 - 範例： $M = 3$
 - 就設定 $C_1 = 100$ ， $C_2 = 010$ ， $C_3 = 001$ 。
- 此種設定方式會有未決定區域的產生：
 - 超過一個以上的感知機同時輸出是 1。
 - 沒有任何一個感知機的輸出是 1。

多類別分類 (2)



多類別分類 (3)

- 節省方式： M 類別的分類問題就用 $\log_2 M$ 個感知機。
 - 範例： $M=4$ 就用 $\log_2 4 = 2$
 - 設定 $C_1 = 00$ ， $C_2 = 01$ ， $C_3 = 10$ ， $C_4 = 11$ 。
- 若 $\log_2 M$ 是整數，則此種設定方式不會有未決定區域的產生。否則，也會產生未決定區域：
 - 沒有編碼到的區域是未定區域。
 - 範例： $M=5$ ， $\log_2 5 > 2$ ，就用 3 個感知機。
 - 設定 $C_1 = 000$ ， $C_2 = 001$ ， $C_3 = 010$ ， $C_4 = 011$ ， $C_5 = 100$ ，其餘 101，110，111 則是未定區域。