

# 分類技術補充

---

# 您已經被大數據包圍了！

- 「大數據」都聽濫了也沒搞懂，其實自己正生活在大數據的包圍中了
- <https://kknews.cc/news/6kjrj2l.html>
- 怎麼理解什麼是大數據呢？
- <https://kknews.cc/code/vzagk6y.html>

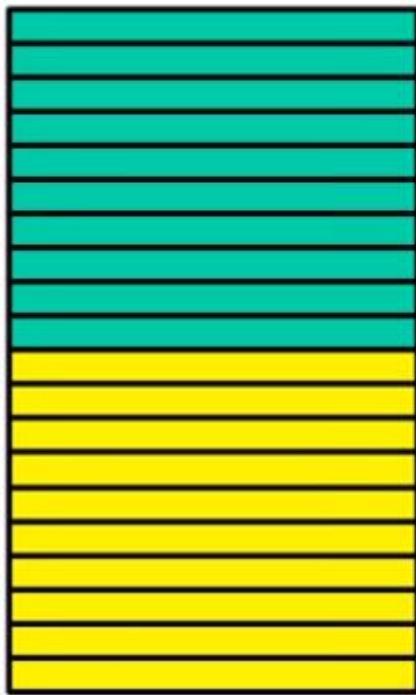
# **Training Dataset and Testing Dataset of a Classifier**

分類技術資料集



# Training Set and Testing Set

## Dataset



10筆綠色資料



10筆黃色資料

20筆資料中綠色類別有十筆，黃色類別有十筆，接下來先進行隨機抽樣(Sampling)讓訓練樣本資料(Training dataset)有30%(6筆)，剩下的測試樣本資料(Testing dataset)有70%(14筆)

# Training Set and Testing Set

Testing Set



Training Set

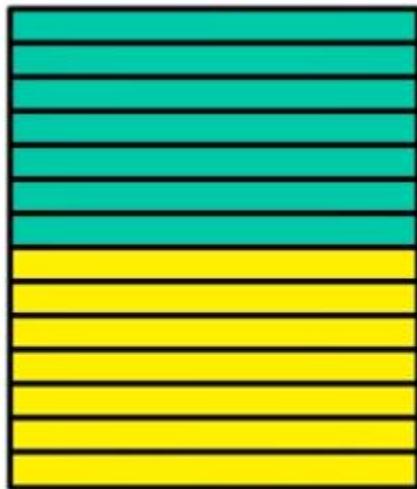
30%

隨機抽樣

70%

# Training Set and Testing Set

**Testing Set**



**70%**

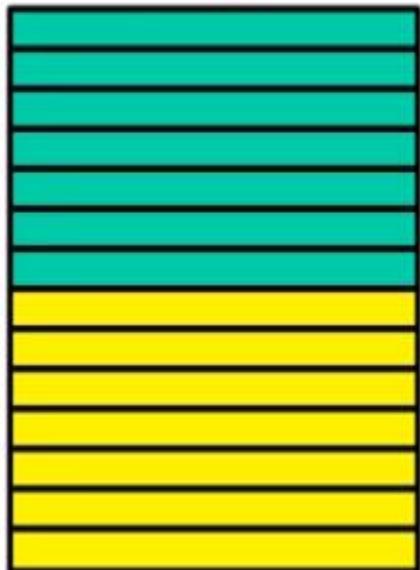
**Training Set**



**30%** 📄

# Training Set and Testing Set

Testing Set



Training Set



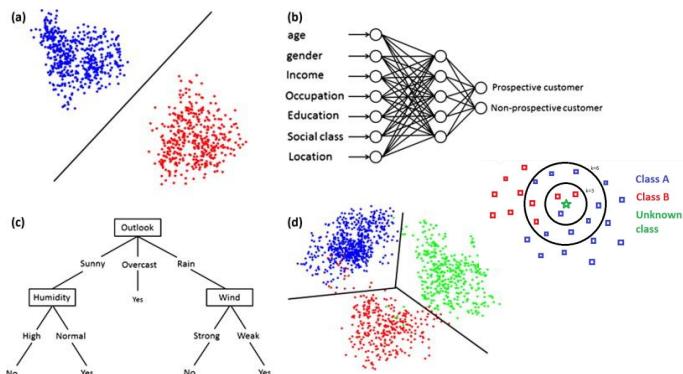
Train

Classifier

選擇方法：  
K-NN  
Decision Tree  
Naïve Bayes  
SVM ANN  
CNN...

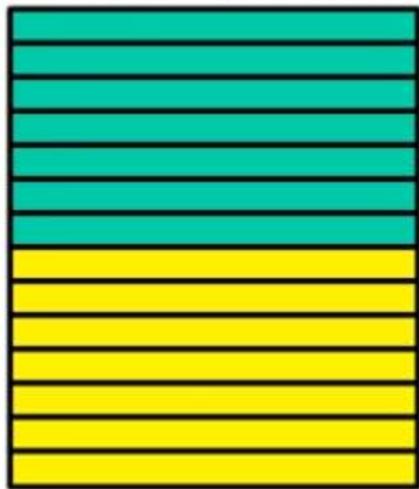
分類器

30%



# Training Set and Testing Set

Testing Set



70%

Training Set

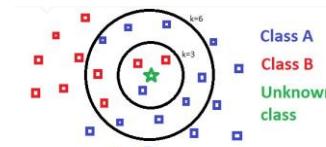


30%

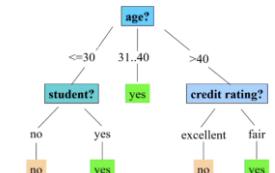
進行測試

Train → Classifier

分類器

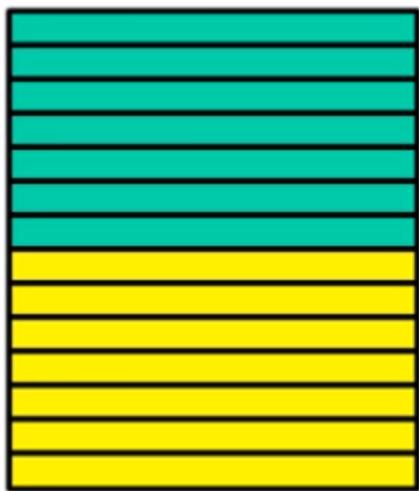


Class A  
Class B  
Unknown class



# Training Set and Testing Set

**Testing Set**



測試集的實際資料

**Training Set**



Train



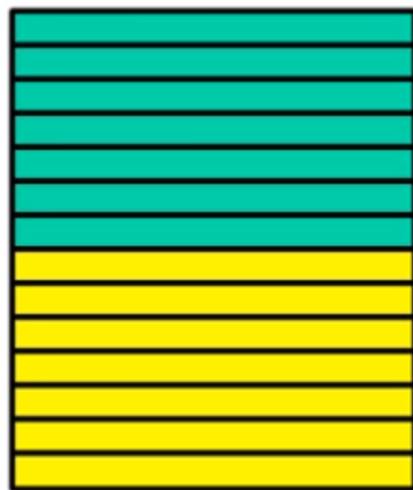
Classifier

測試結果資料  
(即預測結果)



# Training Set and Testing Set

## Testing Set



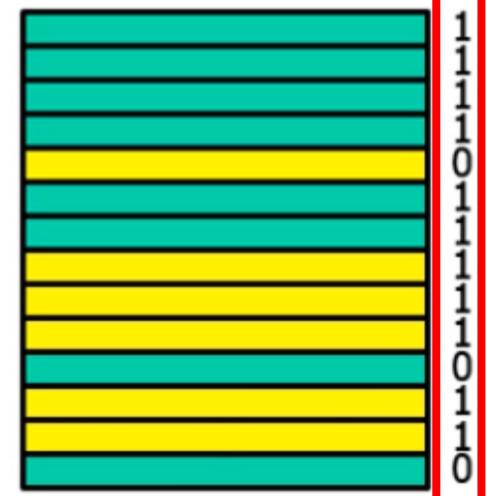
計算整理成  
混淆矩陣  
(Confusion Matrix)

## Training Set



Train

## Classifier

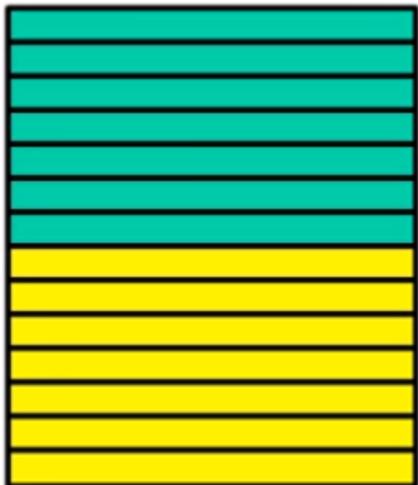


		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

建立混淆矩陣

# Training Set and Testing Set

Testing Set



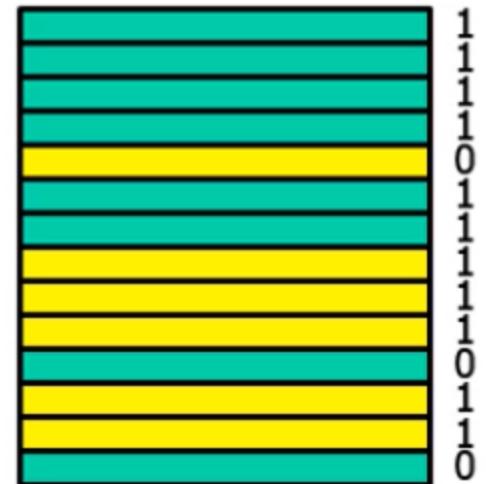
Training Set



Train



Classifier



Classification Test Accuracy =  $11/14 = 0.786$

Classification Test Error =  $3/14 = 0.214$

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

Performance...?

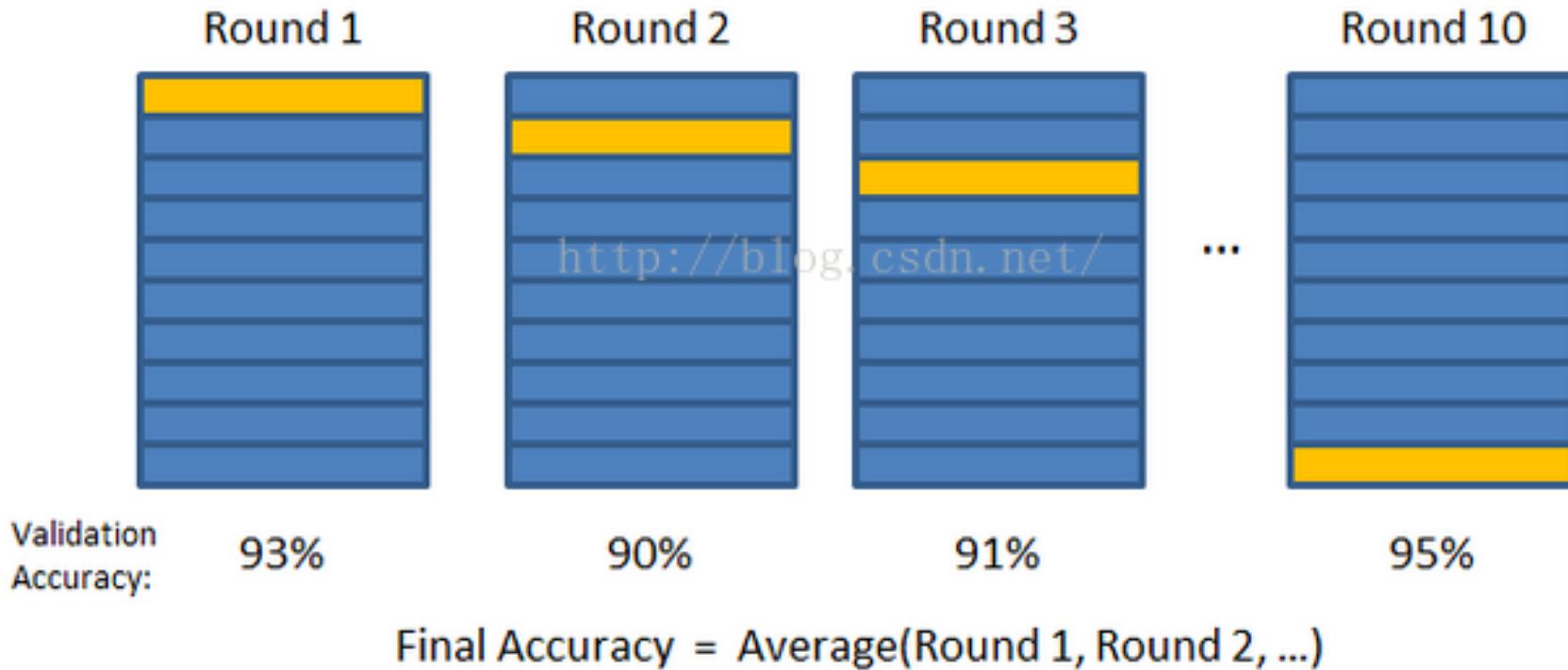
DEMO

# k-fold Cross-validation k折交叉驗證

- 一般來說我們會將數據分為兩個部分，一部分用來訓練，一部分用來測試，交叉驗證是一種統計學上將樣本切割成多個小子集的做測試與訓練。
- 為什麼需要交叉驗證
  - 為了避免依賴某一特定的訓練和測試資料產生偏差。
- 在k交叉驗證中，是使用不同的資料組合來驗證你訓練的模型，舉例來說，假設你有100個樣本，你可以第一次先使用前90個做訓練，另外10個做測試，然後再用第80到90個，不斷重複這個動作，這樣你可以得到不同的訓練/測試資料組合，提供更多數據去驗證。

# k-fold Cross-validation k折交叉驗證

 Validation Set  
 Training Set



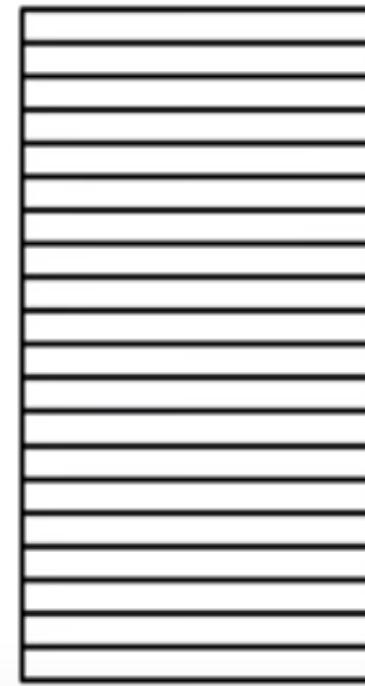
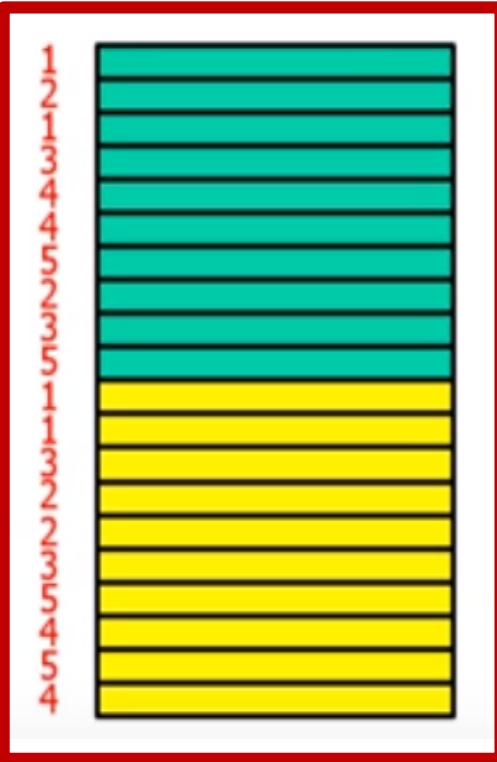
# k-fold Cross-validation k折交叉驗證

- 如上圖，我們將資料分成10等份，其中第1等分用來當作驗證的測試資料，其餘9份拿來訓練，下一輪我們繼續將第2等分拿來當作驗證的測試資料，其餘9份一樣拿來訓練，總共做10次。
- 藉著將10次的準確性(Accuracy)平均，而這個得到的平均值，我們可以自信的說這個數值就是我們的準確度沒有偏差。

# *k*-fold Crossvalidation

資料有20筆，設定  
 $k=5$ 時，則 $20/5=4$ ，  
即每一等分會有四筆

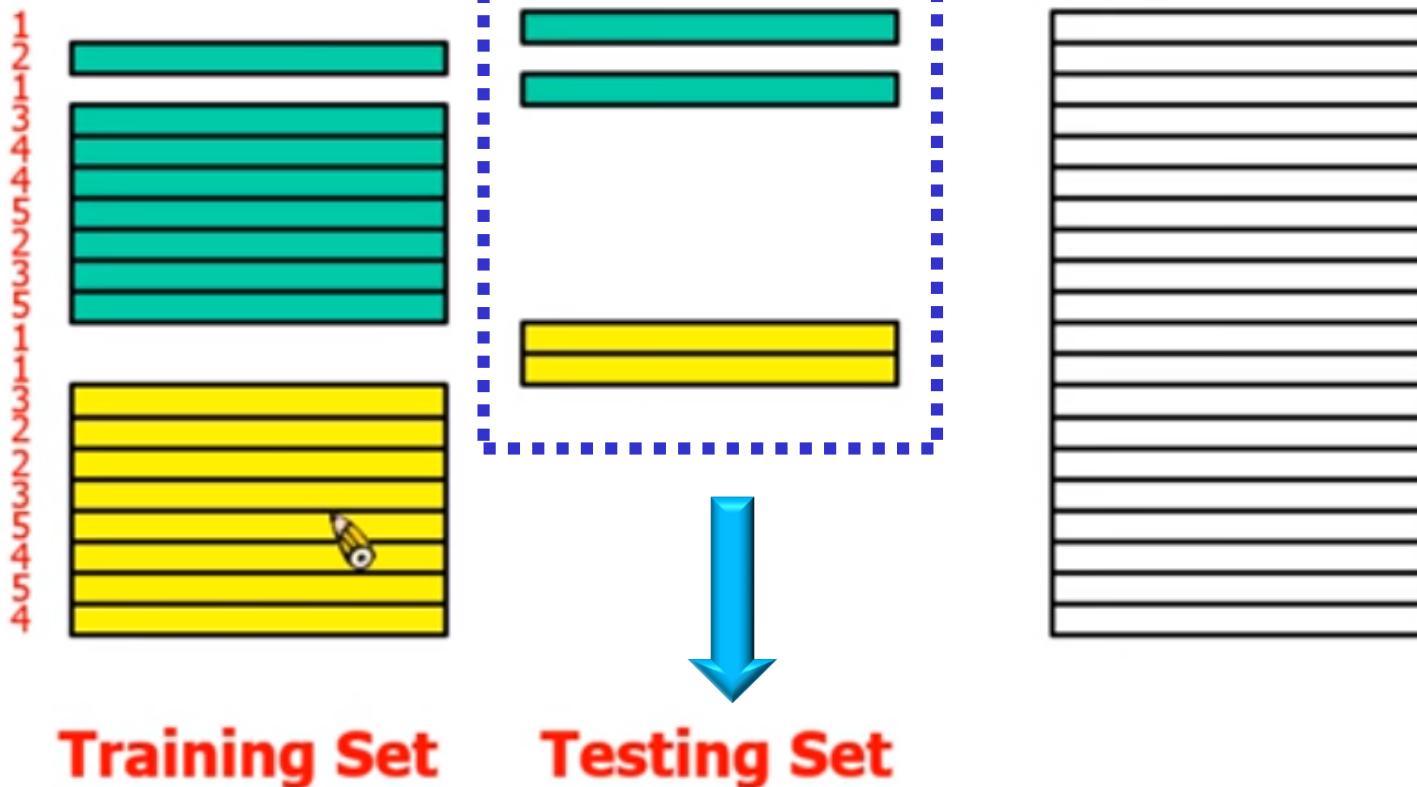
## 原來的Dataset



假設  $k=5$  會將每一類資料隨機分成五等份，誰去切的？  
怎麼切？程式會盡量很平均將資料切成五塊(等份)，  
即程式設計中會將資料切成每一類資料都一致

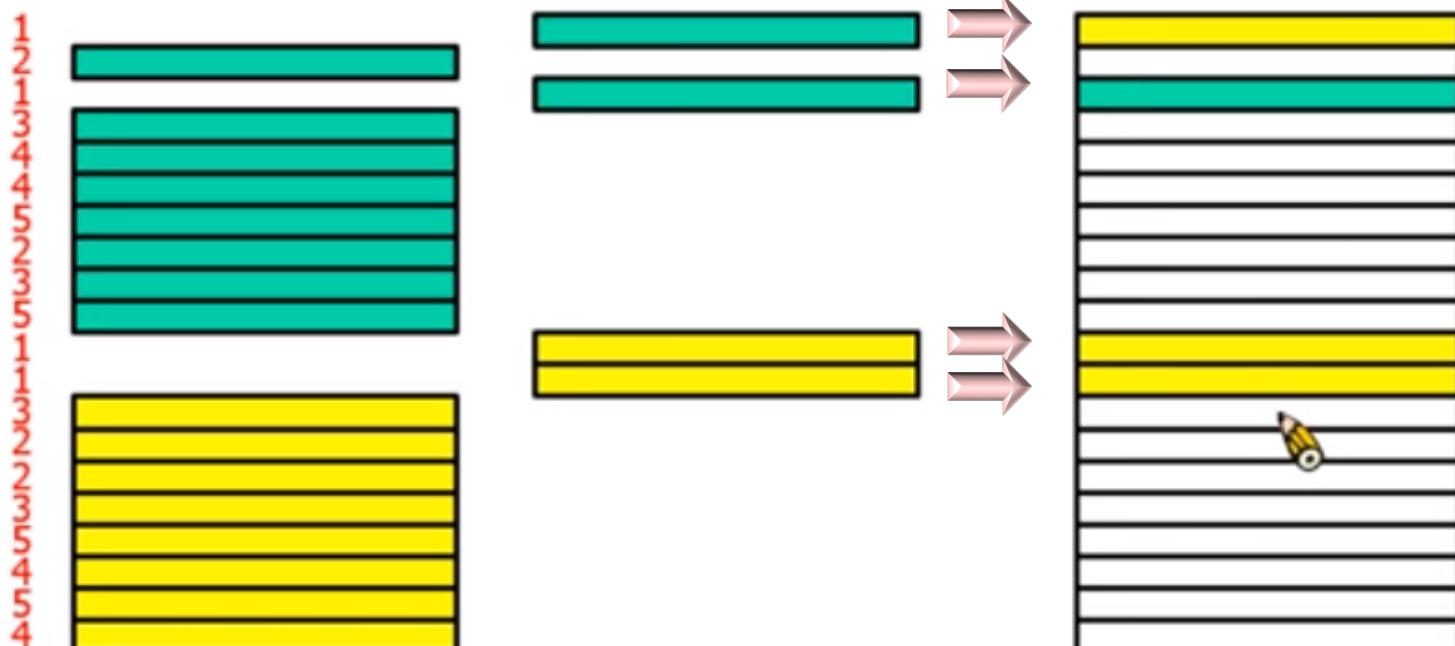
# *k*-fold Crossvalidation

- 1<sup>st</sup> round



# *k*-fold Crossvalidation

- 1<sup>st</sup> round

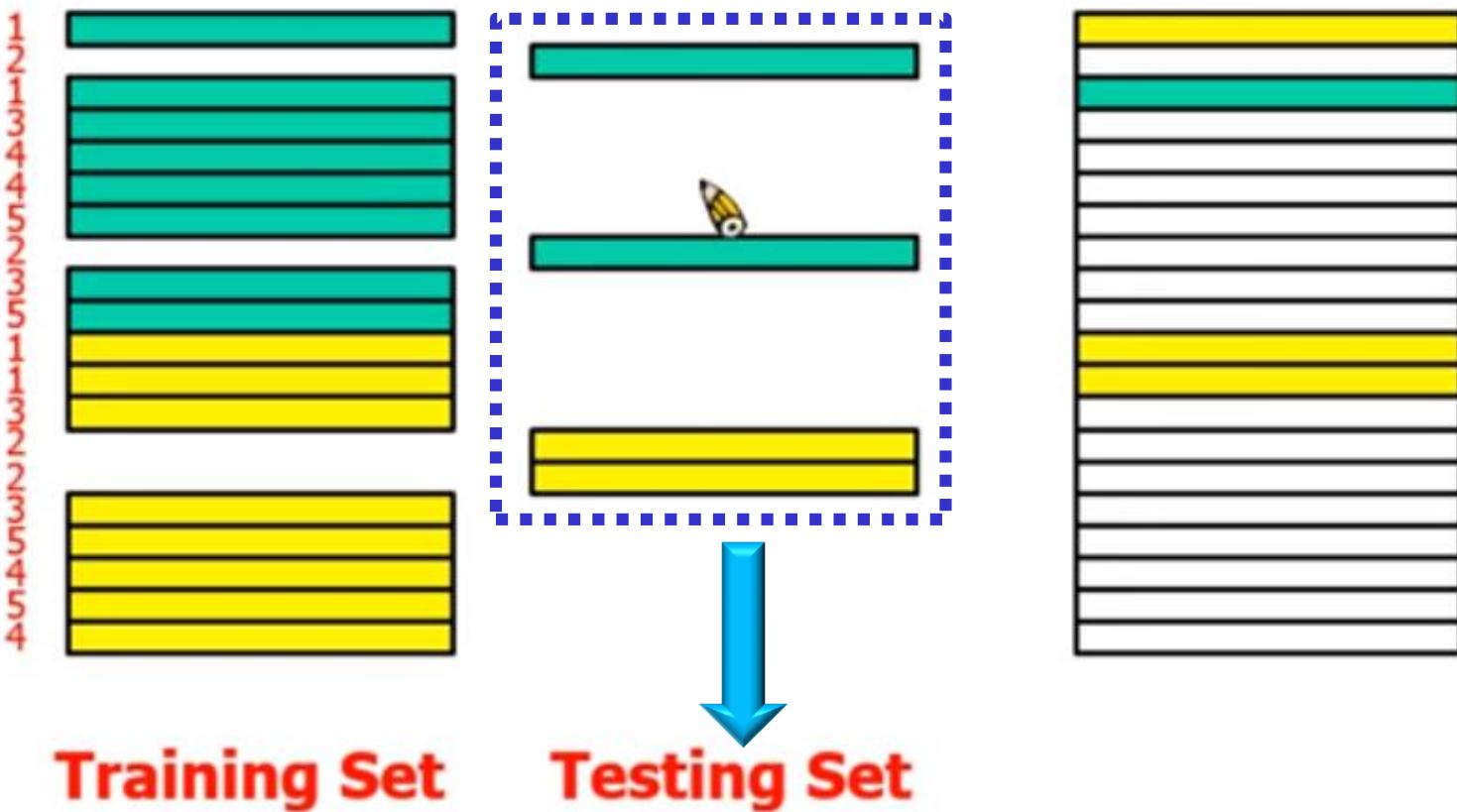


**Training Set**

**Testing Set**

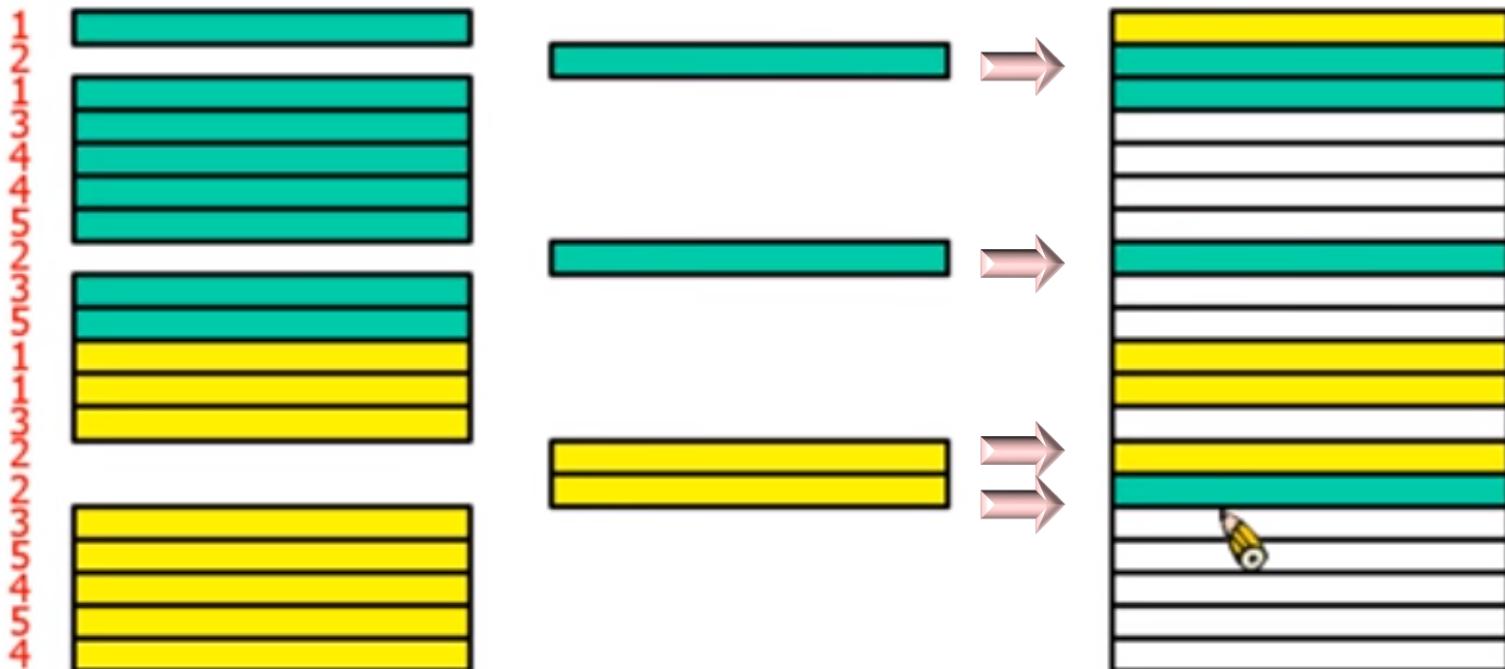
# *k*-fold Crossvalidation

- 2<sup>nd</sup> round



# *k*-fold Crossvalidation

- 2<sup>nd</sup> round

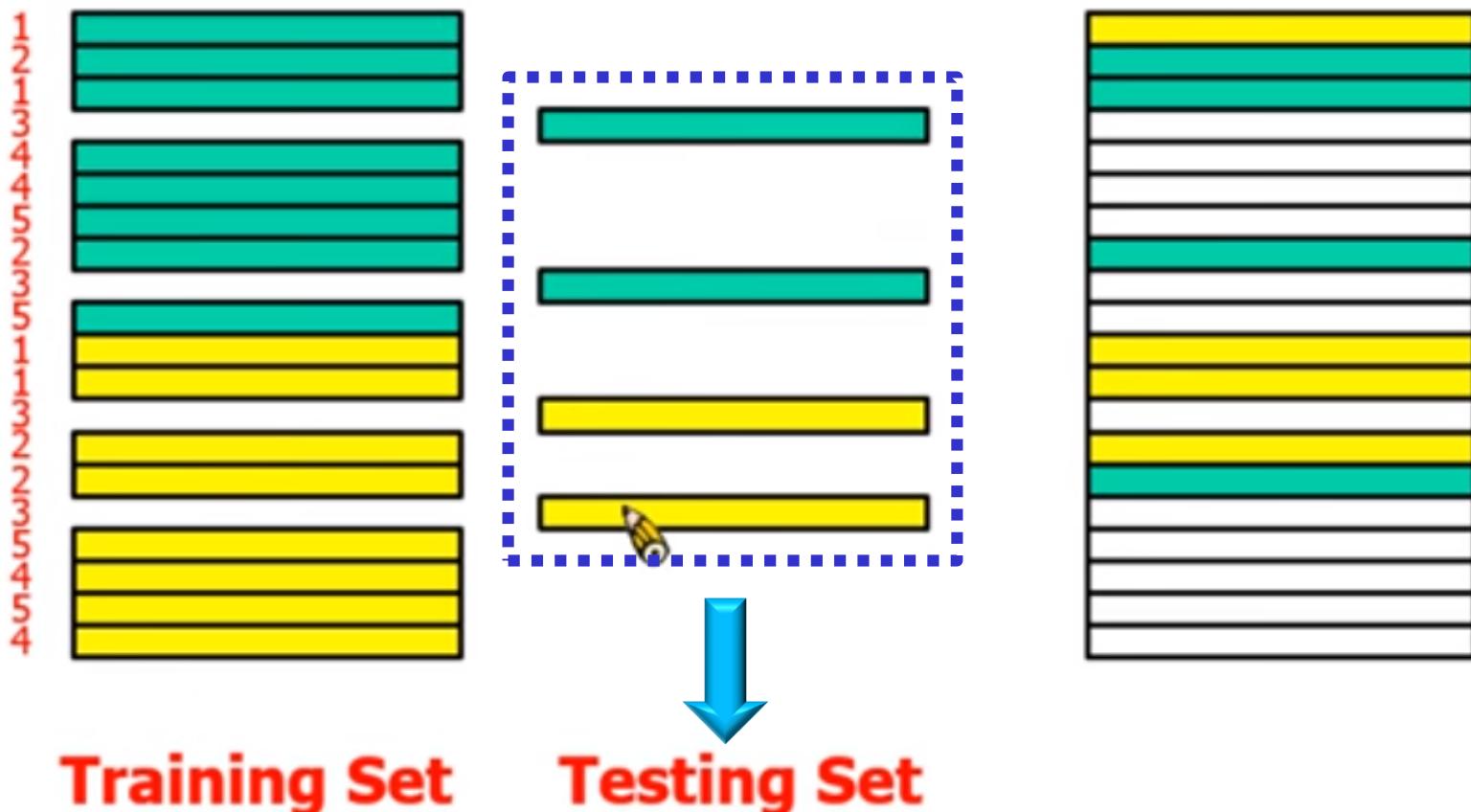


**Training Set**

**Testing Set**

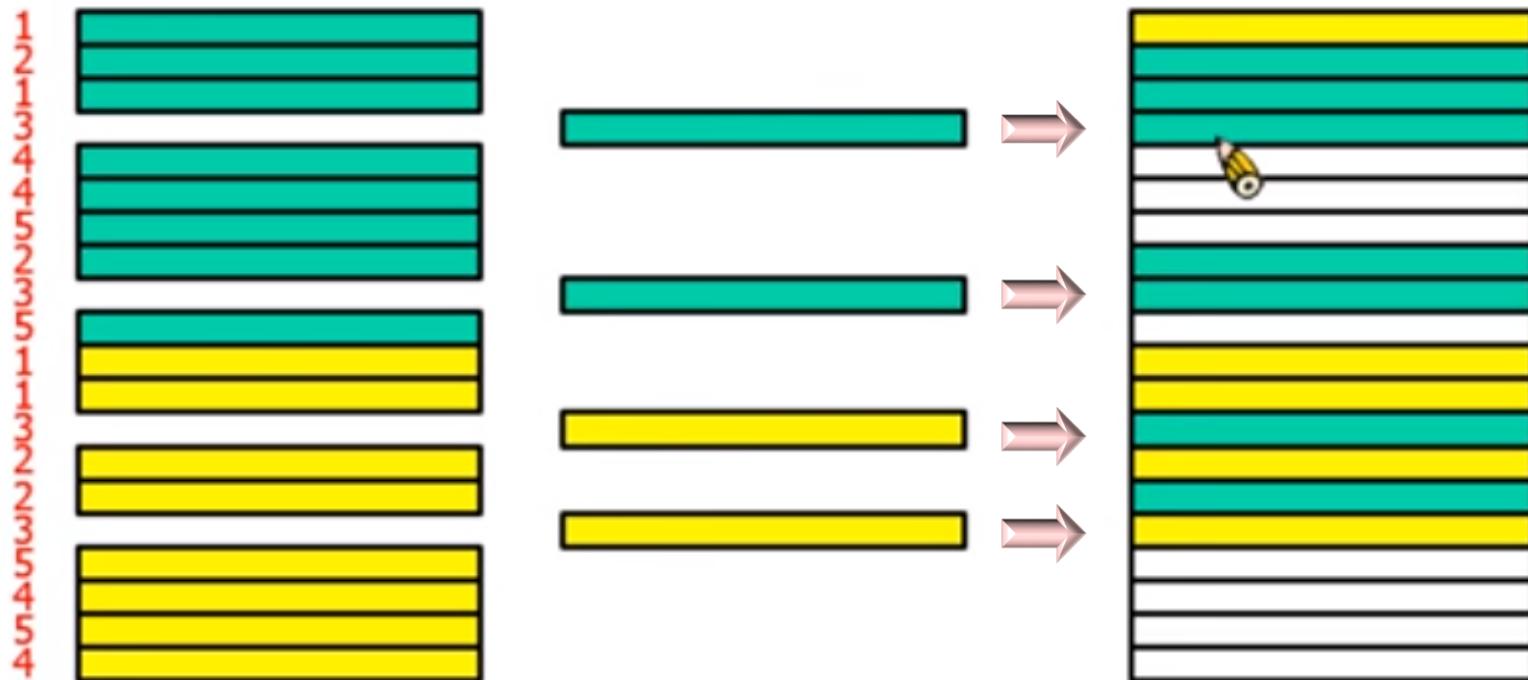
# *k*-fold Crossvalidation

- 3<sup>rd</sup> round



# $k$ -fold Crossvalidation

- 3<sup>rd</sup> round

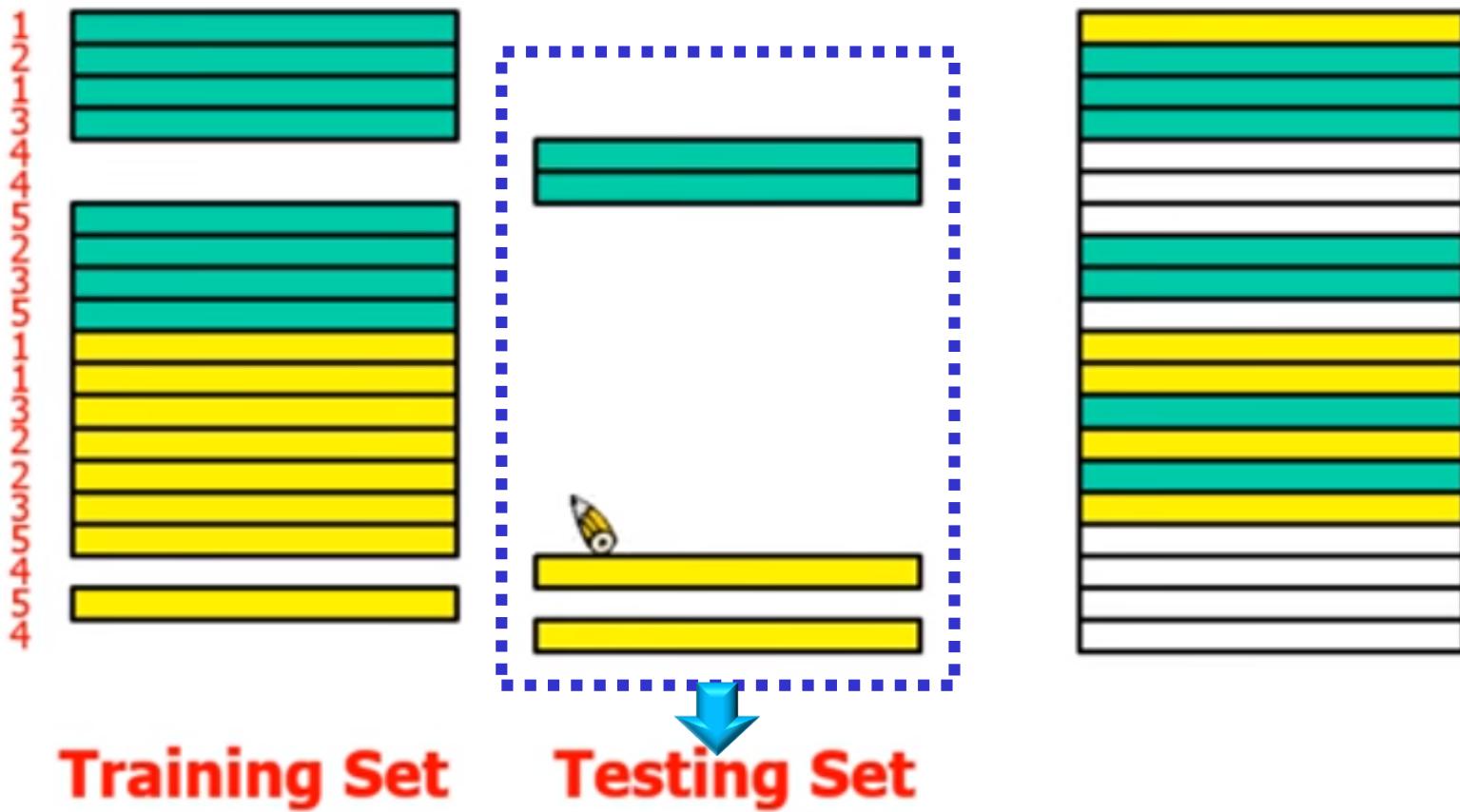


Training Set

Testing Set

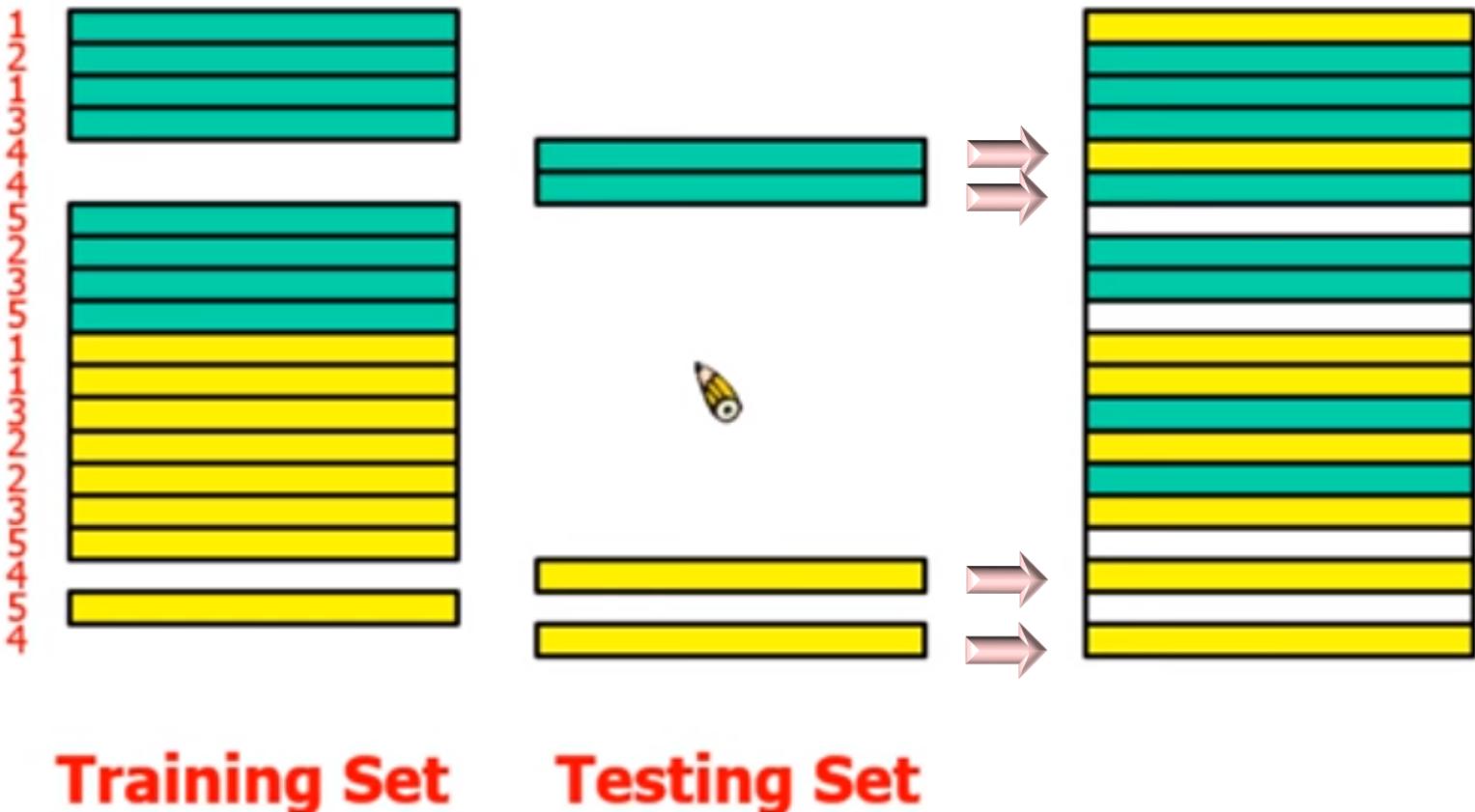
# *k*-fold Crossvalidation

- 4<sup>th</sup> round



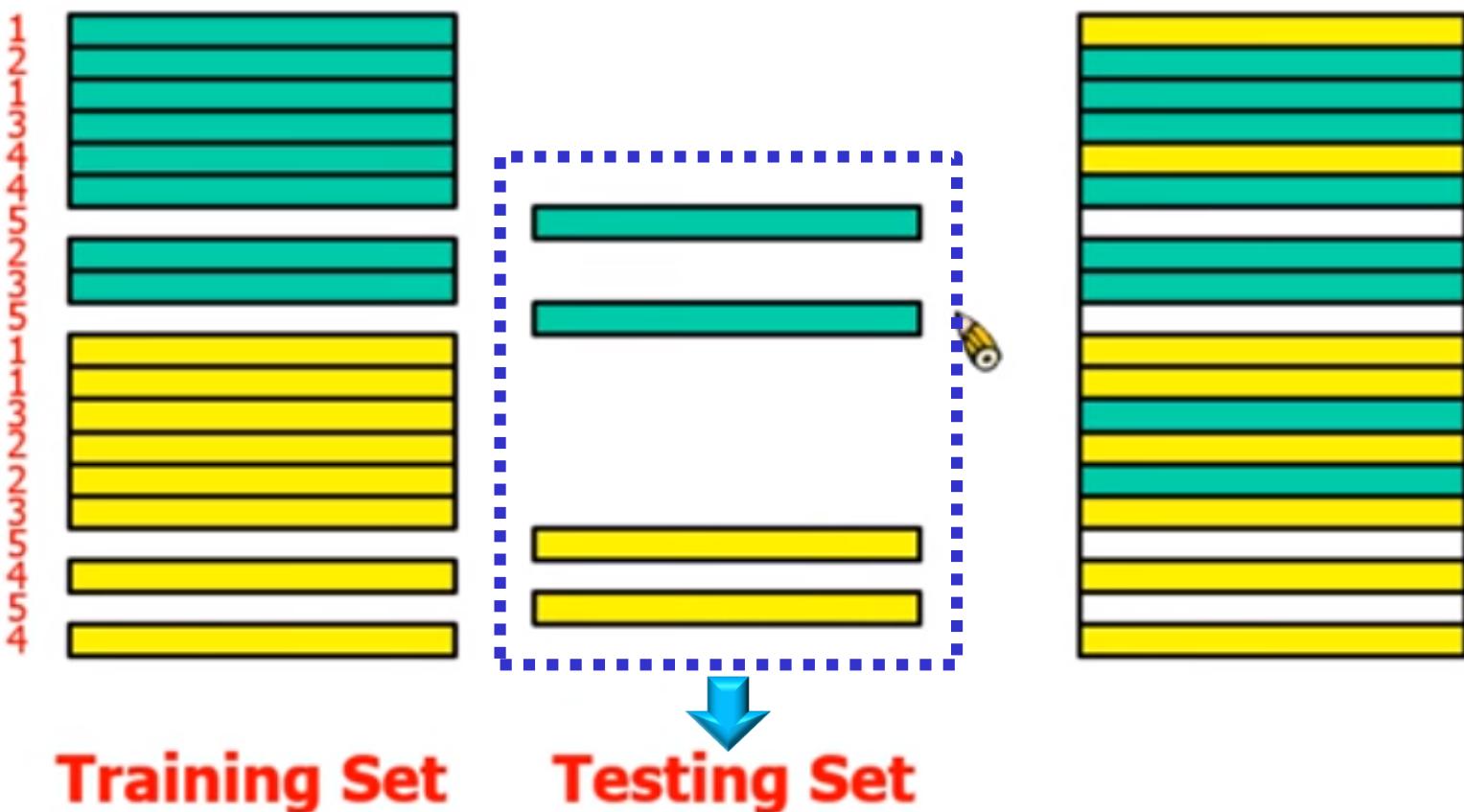
# *k*-fold Crossvalidation

- 4<sup>th</sup> round



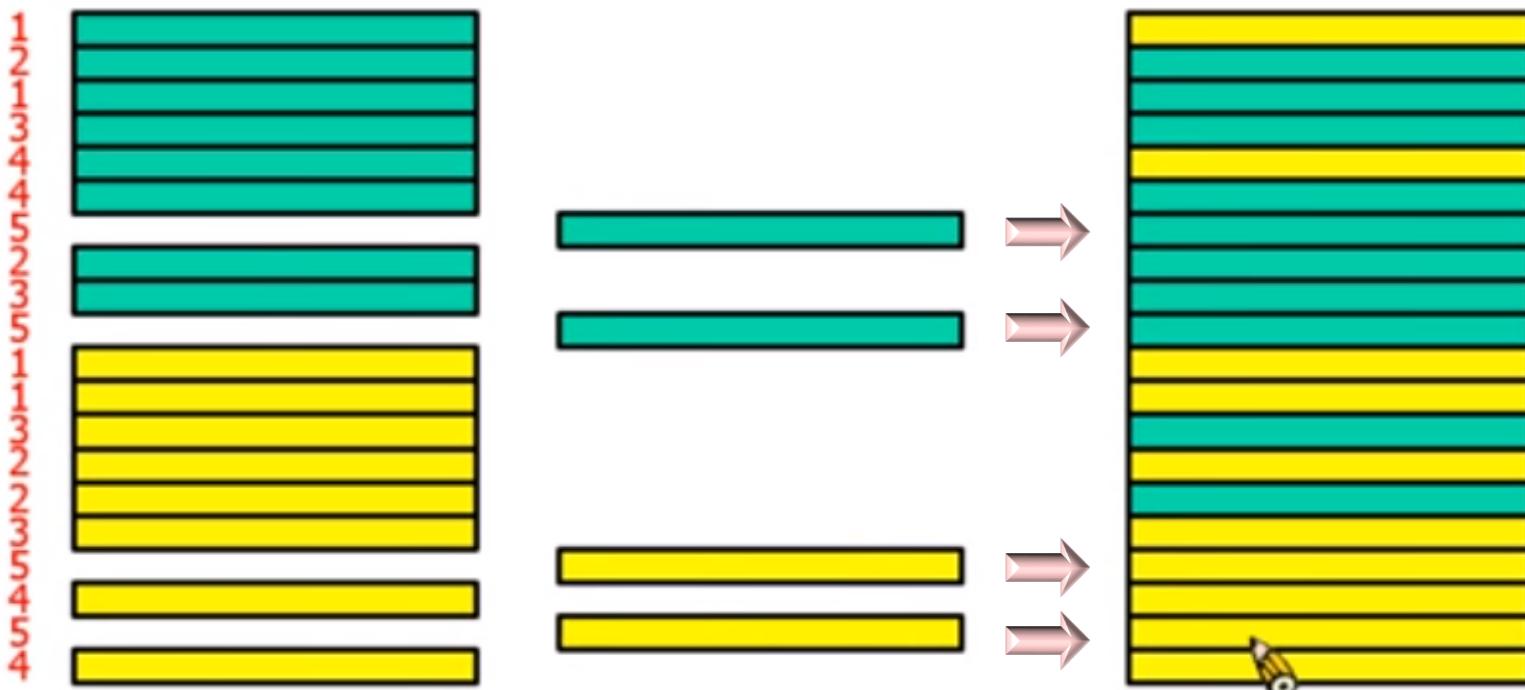
# *k*-fold Crossvalidation

- 5<sup>th</sup> round



# $k$ -fold Crossvalidation

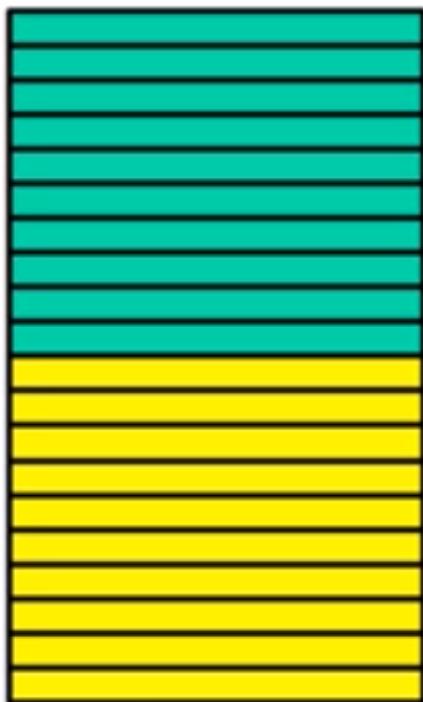
- 5<sup>th</sup> round



**Training Set**

**Testing Set**

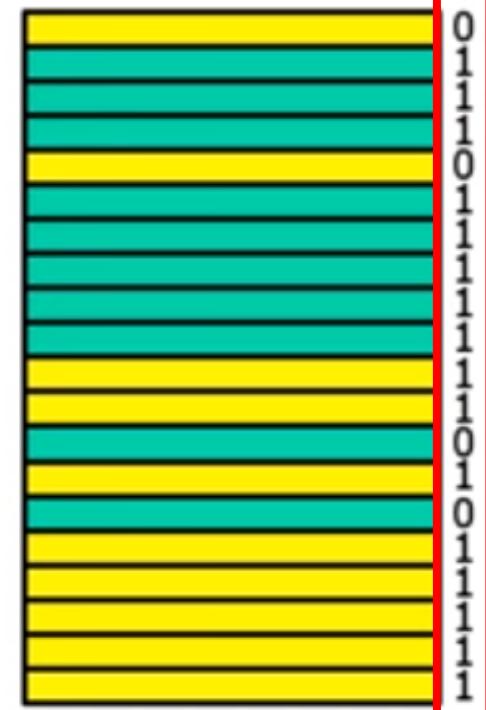
# *k*-fold Crossvalidation



		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

計算整理成  
混淆矩陣  
(Confusion  
Matrix)

建立混淆矩陣



5-fold Crossvalidation Accuracy =  $16/20 = 0.8$

5-fold Crossvalidation Error =  $4/20 = 0.2$

**DEMO**

# Evaluating the Performance of a Classifier

分類技術的評估

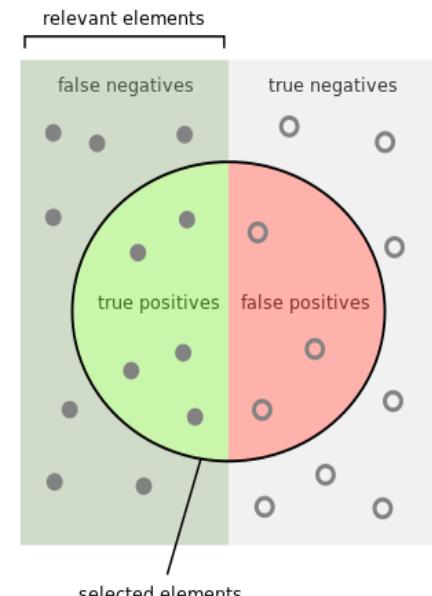
# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a classification model?**
    - Accuracy(正確率)**
    - Precision(精確率/查準率)**
    - Recall(召回率/查全率)**
    - F1-Measure(Precision與Recall的調和平均數)**

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$



# Precision and Recall

- 假設我們的資料有兩類，正類(Positive)與負類(Negative)資料，通常正類資料是指我們有興趣的那一類資料，相反地，負類資料就是我們沒有興趣的那一類資料。
- 接著我們發明一個全新的分類技術演算方法，或者找到一個知名的分類技術演算方法，並依照此方法所說明的每一個步驟，一步一步地將手邊資料不分正或者負全都餵進去這一個方法中執行資料的分類(即學習過程)與建立分類模型。
- 最後，取測試資料再餵進此分類模型中，而每一筆測試資料必定有一個產出的答案，不是”正”就是”負”，此稱資料預測的結果值，然後將此預測結果值跟事先已經知道的真正答案對比較，並記錄對幾筆錯幾筆，然後全部記錄在混淆矩陣中。

# Precision and Recall

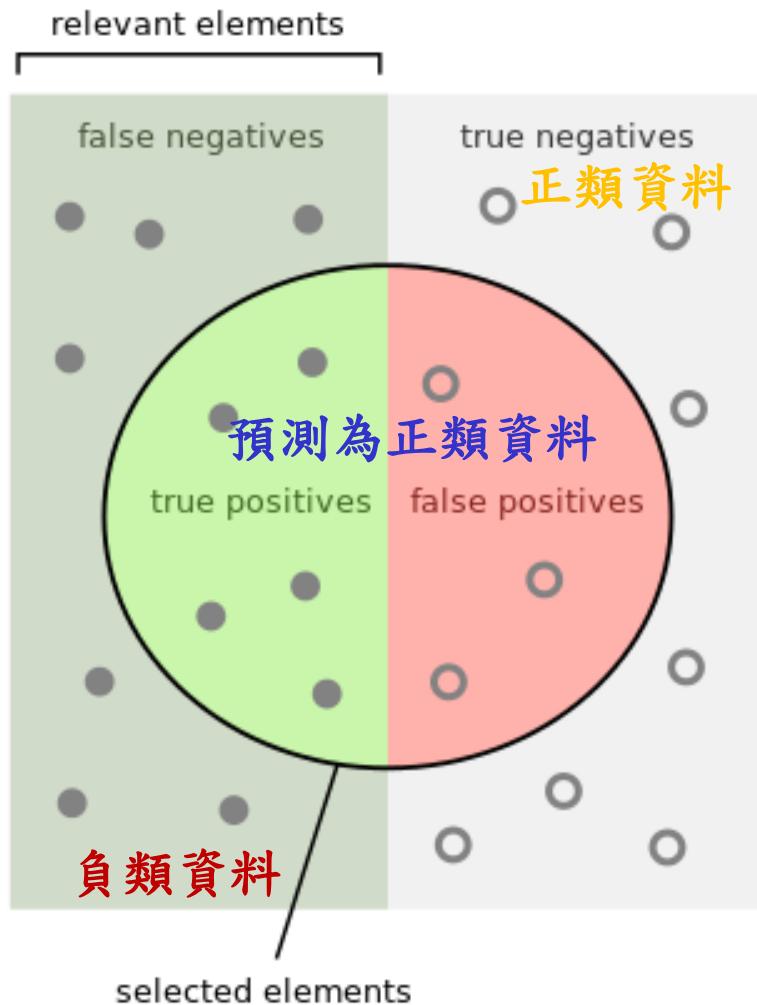
- 精確率(Precision)是針對我們預測結果而言的有多準呢?
  - 它表示的是預測為正的樣本中有多少是真正的正樣本。
  - 那麼預測為正就有兩種可能了，一種就是把正類預測為正類(TP)，另一種就是把負類預測為正類(FP)，也就是

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- 召回率(Recall)是針對我們原來的樣本而言的，它表示的是樣本中的正例有多少被預測正確了。那也有兩種可能，一種是把原來的正類預測成正類(TP)，另一種就是把原來的正類預測為負類(FN)。

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

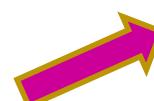
# 維基百科上說明



How many selected items are relevant?

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

您的演算法  
的分類結果



How many relevant items are selected?

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

您的dataset中  
正類資料筆數



# Precision and Recall(查準率與查全率)

- 在資訊檢索或資料探勘的領域中，一個最基本的問題就是到底回傳回來的結果，是不是使用者想要的？回傳的效率有多好？我們可用下面兩個評估檢索結果的方法，叫做【查準率(Precision)】和【查全率(Recall)】。
- **Precision** = Relevant Documents Retrieved / Total Retrieved Documents
  -  (抓回來的相關文章數量) / (抓回文章的總數)
- **Recall** = Relevant Documents Retrieved / Total Relevant Documents
  -  (抓回來的相關文章數量) / (相關文章的總數)
- 從上面的公式可以看出來，Precision和Recall的分子都是Relevant Document Retrieved(抓回來的相關文章數量)，差別的地方在於Precision的分母是【抓回文章的總數】；而Recall的分母則是【相關文章的總數】。

# Precision and Recall

- 範例：
- 假設現在資料庫中有10000筆資料，和美食有關的文章有500篇。使用者在輸入”美食”的關鍵字後，回傳的文章有4000篇，其中有400篇是和美食有關的。
- Precision =  $400 / 4000 = 10\%$   
Recall =  $400 / 500 = 80\%$
- 其結果為：這個搜尋引擎的查準率是10%、查全率是80%。

**回傳回來的結果，是不是使用者想要的？  
回傳的效率有多好？**

# Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix(混淆矩陣):

		PREDICTED CLASS (預測)	
		Class=Yes	Class>No
ACTUAL CLASS (實際)	Class=Yes	a	b
	Class>No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

# Metrics for Performance Evaluation...

		PREDICTED CLASS(預測)	
		Class=Yes	Class>No
ACTUAL CLASS (實際)	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

# Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0(**target**), accuracy is  $9990/10000 = 99.9\%$ 
  - Accuracy is misleading because model does not detect any class 1 example

# Weighted Accuracy

$$\text{Precision (p)} = \frac{a}{a + c}$$

$$\text{Recall (r)} = \frac{a}{a + b}$$

$$\text{F-measure (F)} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$



此為 Precision 與 Recall  
的調合平均數

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

# Metrics for Performance Evaluation...

		PREDICTED CLASS (預測)	
		Class= <b>Green</b>	Class= <b>Yellow</b>
ACTUAL CLASS (實際)	Class= <b>Green</b>	6 (TP)	2 (FN)
	Class= <b>Yellow</b>	1 (FP)	5 (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

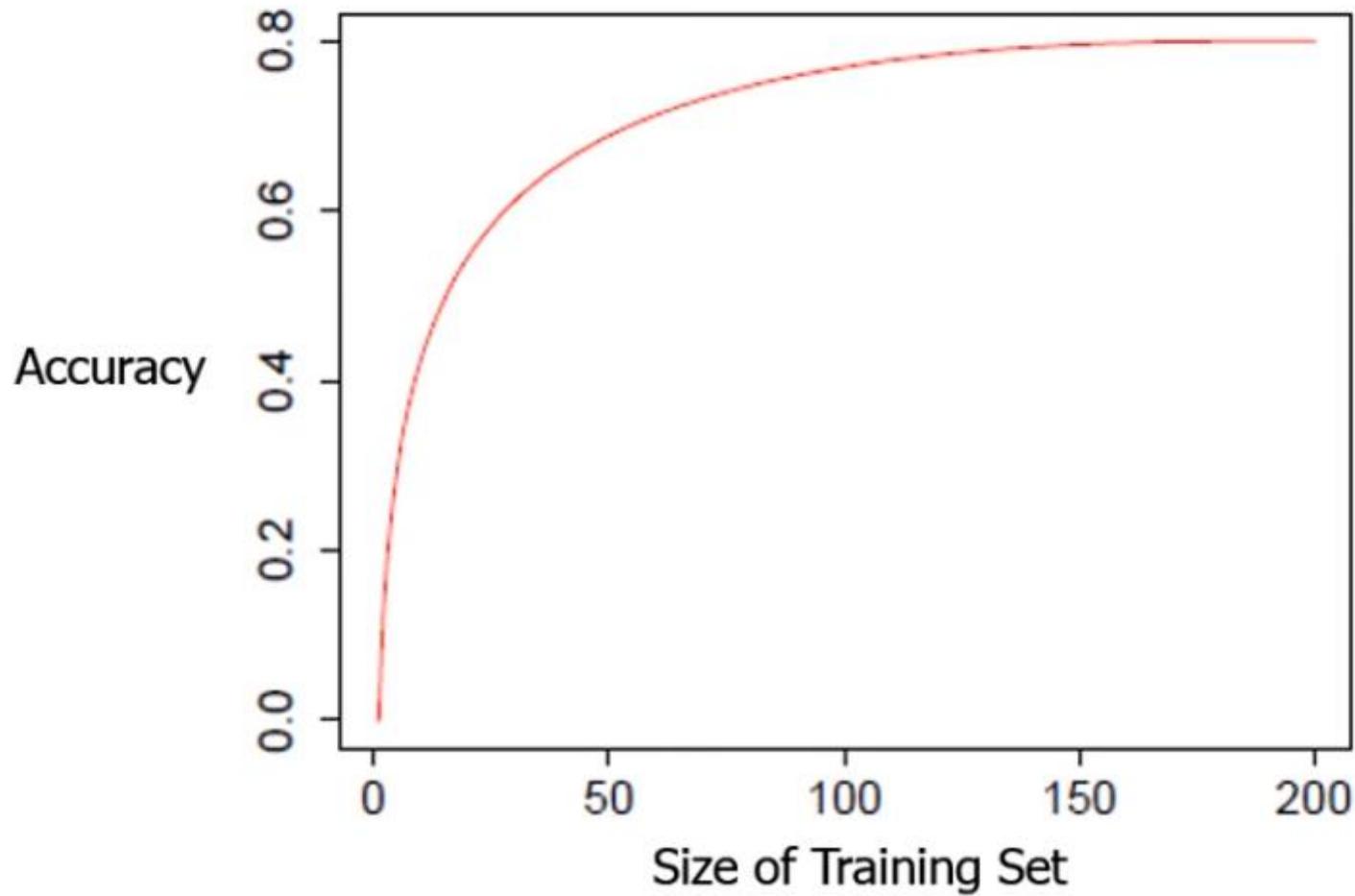


# Training Set and Testing Set

- In order to provide reliable results, it should be run multiple (e.g. 10) times with different training set.



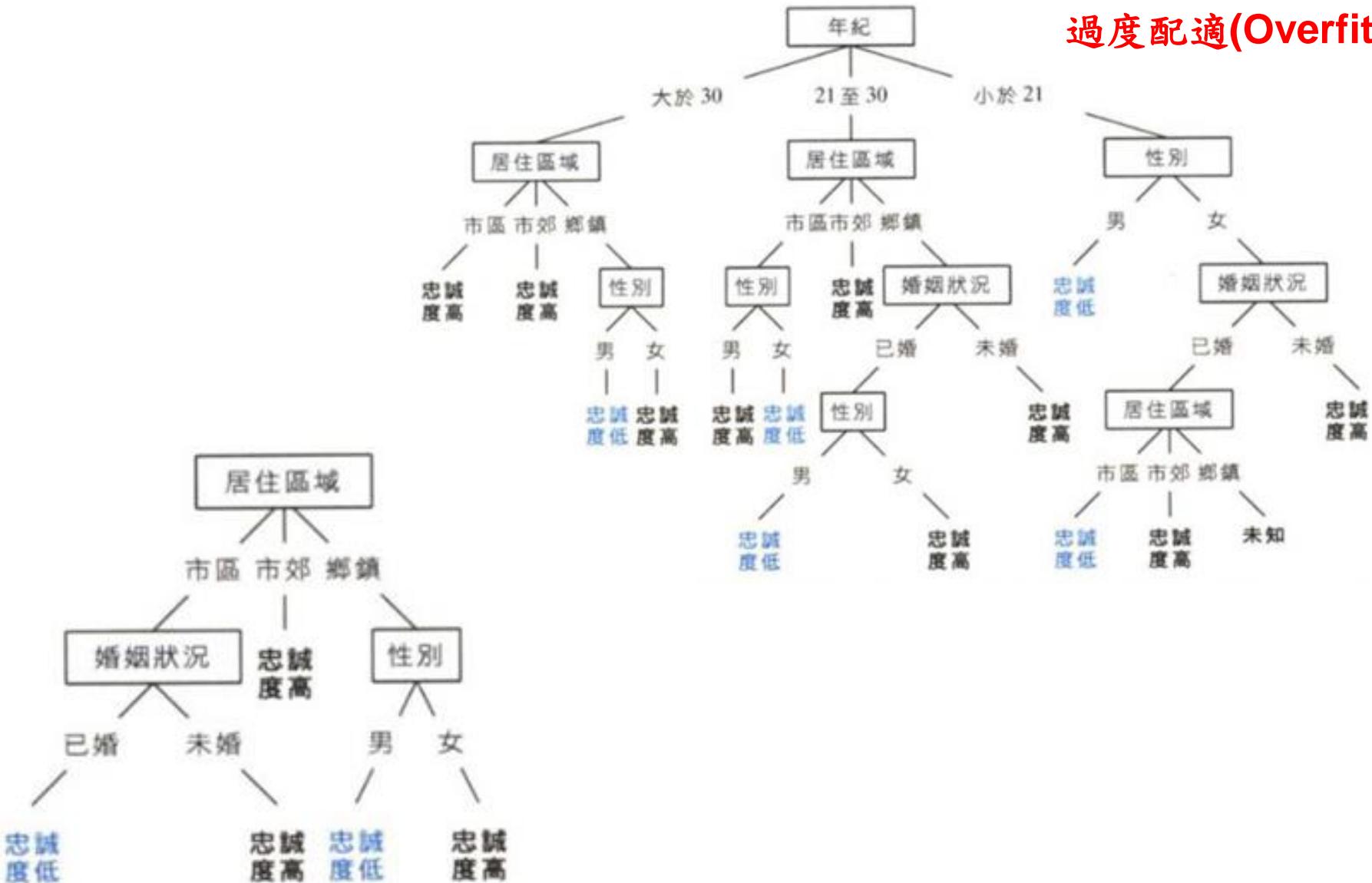
# Size of Training Set vs. Accuracy



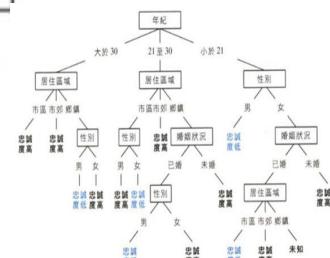
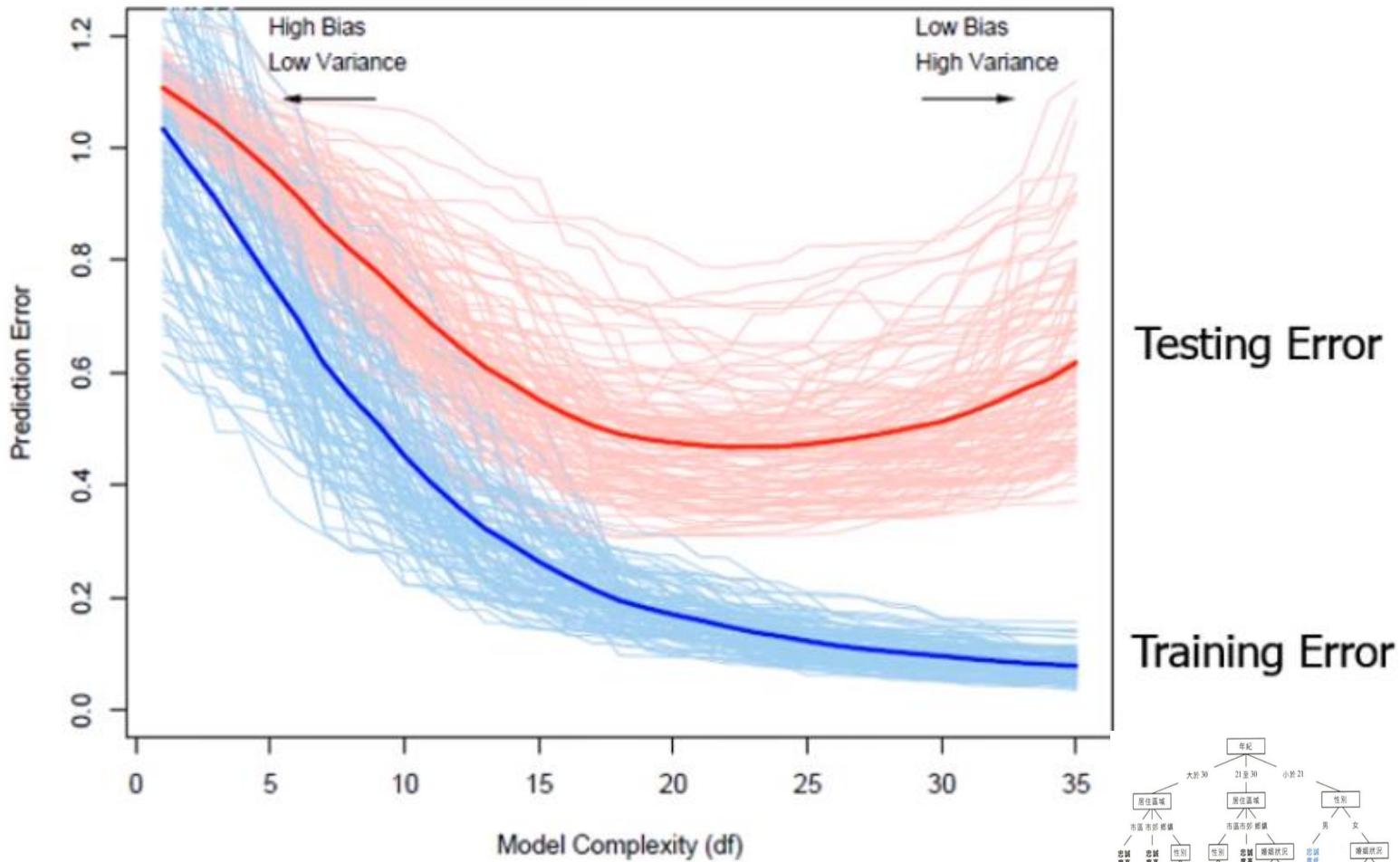
# Training Overfitting

客戶編號	居住區域	年紀	婚姻狀況	性別	品牌忠誠度
1	市區	<=20	已婚	女	低
2	市區	<=20	已婚	男	低
3	市郊	<=20	已婚	女	高
4	鄉鎮	21~30	已婚	女	高
5	鄉鎮	>30	未婚	女	高
6	鄉鎮	>30	未婚	男	低
7	市郊	>30	未婚	男	高
8	市區	21~30	已婚	女	低
9	市區	>30	未婚	女	高
10	鄉鎮	21~30	未婚	女	高
11	市區	21~30	未婚	男	高
12	市郊	21~30	已婚	男	高
13	市郊	<=20	未婚	女	高
14	鄉鎮	21~30	已婚	男	低

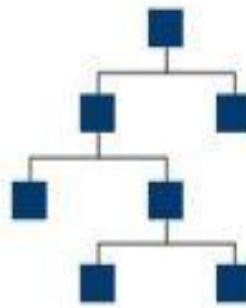
## 過度配適(Overfit)



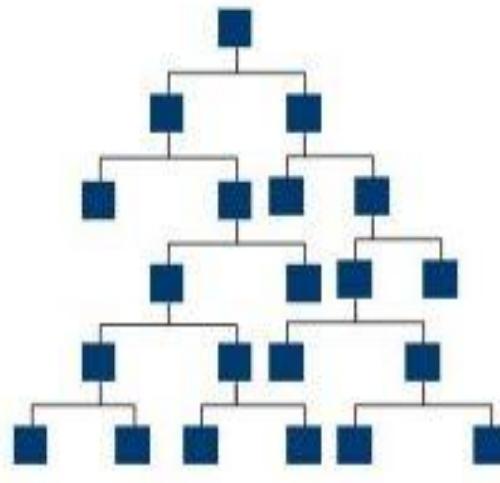
# Training Error vs. Testing Error



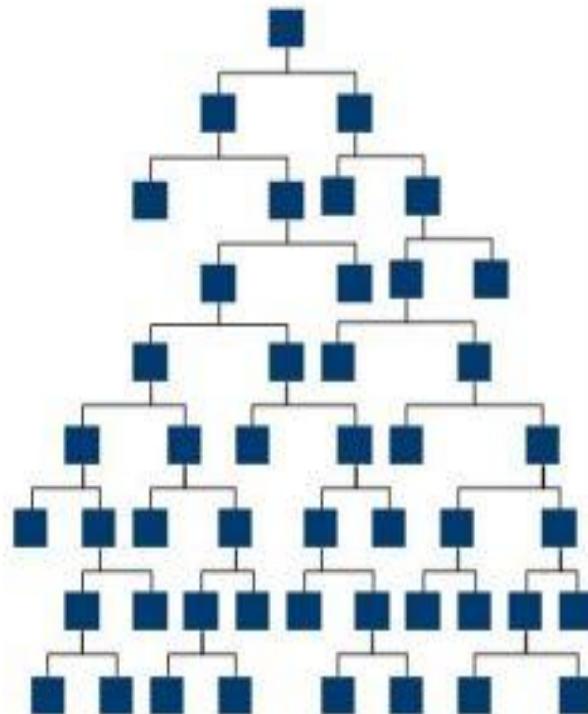
Underfit tree



Optimal tree



Overfit tree



Accuracy on training = 50%

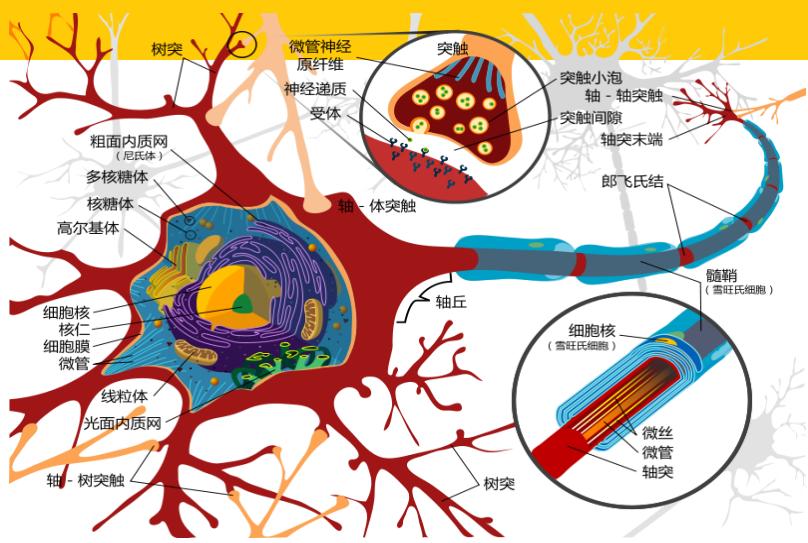
Accuracy on test = 50%

Accuracy on training = 70%

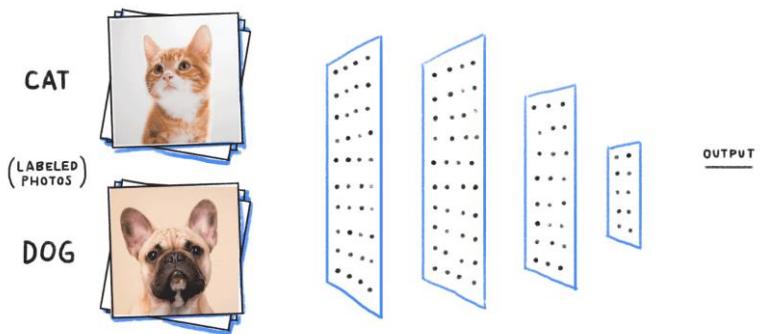
Accuracy on test = 70%

Accuracy on training = 90%

Accuracy on test = 65%

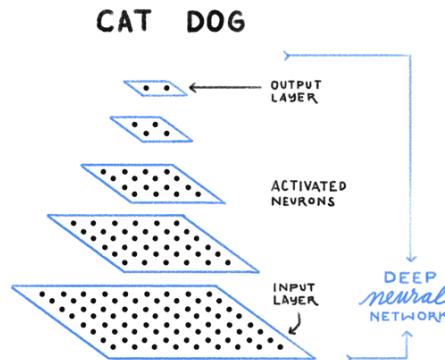


# Basic Concept for ANN Classifier



ANN分類技術

IS THIS A  
CAT or DOG?

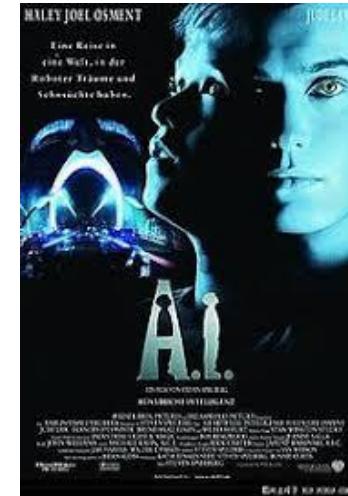


# 聰明的電腦故事--人與機器對決

- 深藍Deep Blue對卡斯帕羅夫
  - 西洋棋
  - 1996年與1997年
- 華生 Watson(人工智慧程式)
  - 《危險邊緣》美國的電視智力競賽節目
  - 2011年
- AlphaGo
  - 圍棋
  - 2014年開始由英國倫敦Google DeepMind開發的人工智慧圍棋軟體

# 紀錄片《遊戲結束：卡斯帕羅夫與電腦（Game Over: Kasparov and the Machine）》

- 1996.2.10 超級電腦「深藍(Deep Blue)」首次挑戰西洋棋世界棋王(卡斯帕羅夫) · 2:4落敗
- 1997.5 超級電腦「更深的藍(Deeper Blue)」第二次挑戰西洋棋世界棋王(卡斯帕羅夫) · 3.5:2.5勝利
  - 2003年一部紀錄片正為此而拍攝，名為《遊戲結束：卡斯巴羅夫與電腦（Game Over: Kasparov and the Machine）》
- 2001 A.I.人工智慧電影，由史蒂芬史匹柏執導



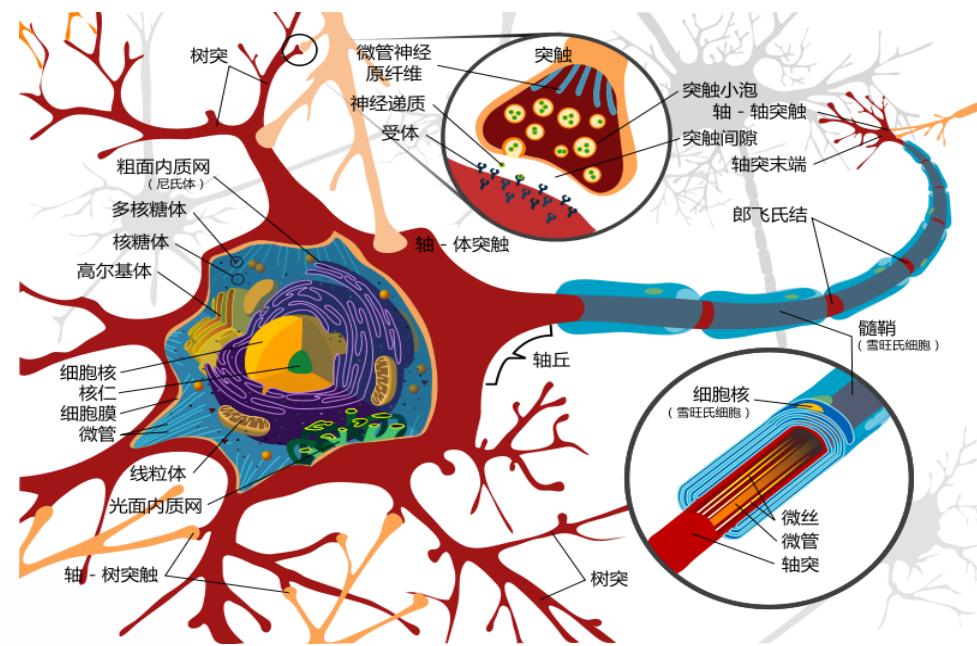
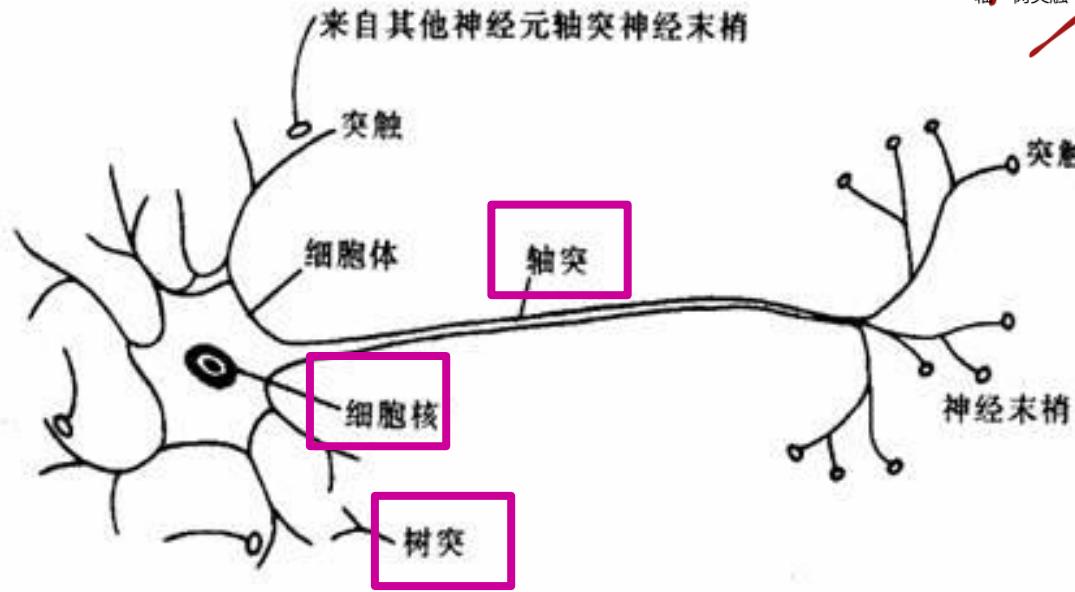
# Artificial Intelligence

- the intelligence exhibited by **machines** or **software**
- the **branch of computer science** that develops machine and software with intelligence
- the study and design of intelligent agents, a system that perceives its environment and takes actions that maximize its chances of success
- the science and engineering of making intelligent machines

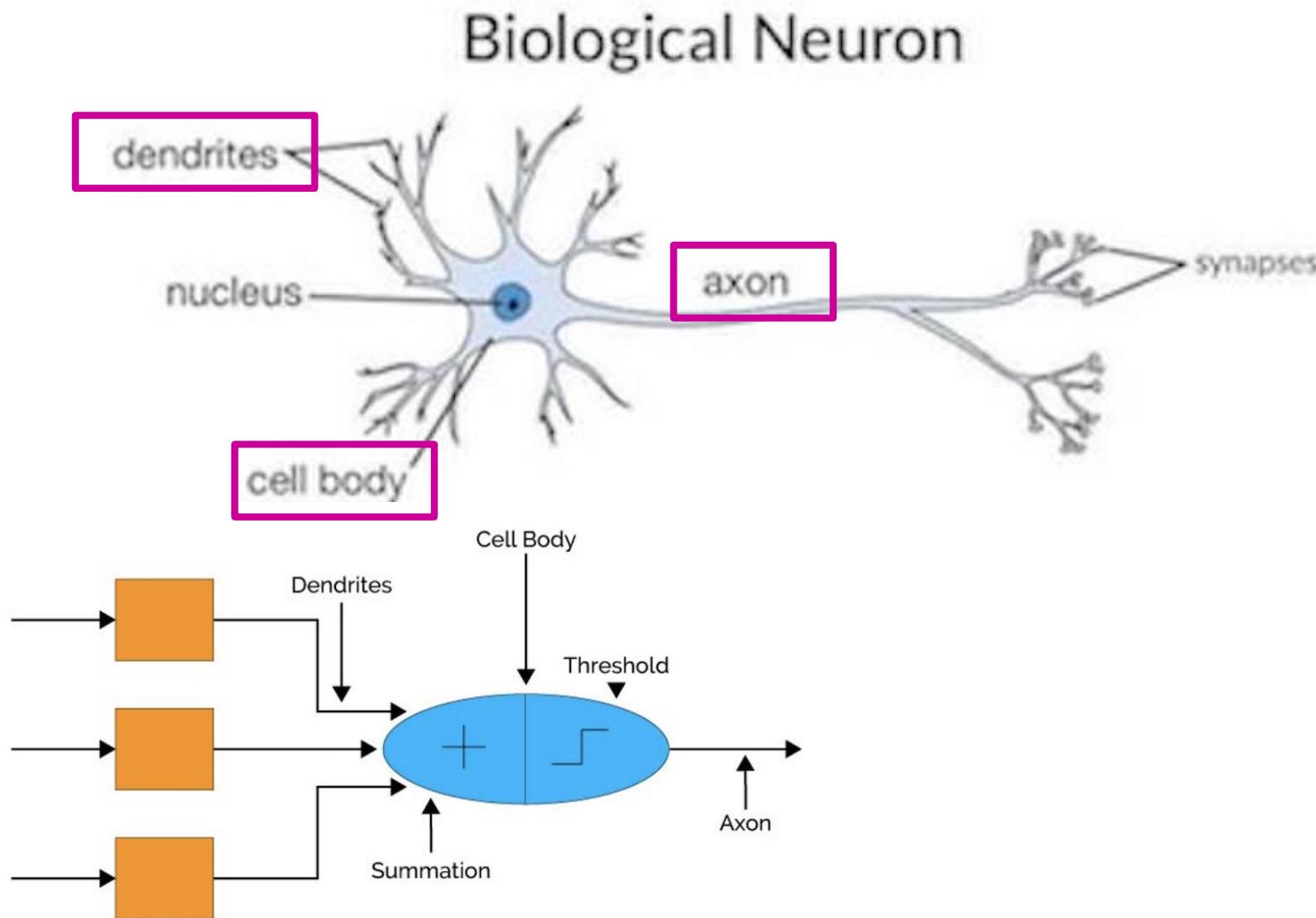
# Tools in Artificial Intelligence

- Search and optimization
- Logic
- Probabilistic methods for uncertain reasoning
- ***Classifiers and statistical learning methods***
- ***Neural networks***
- Control theory
- Languages

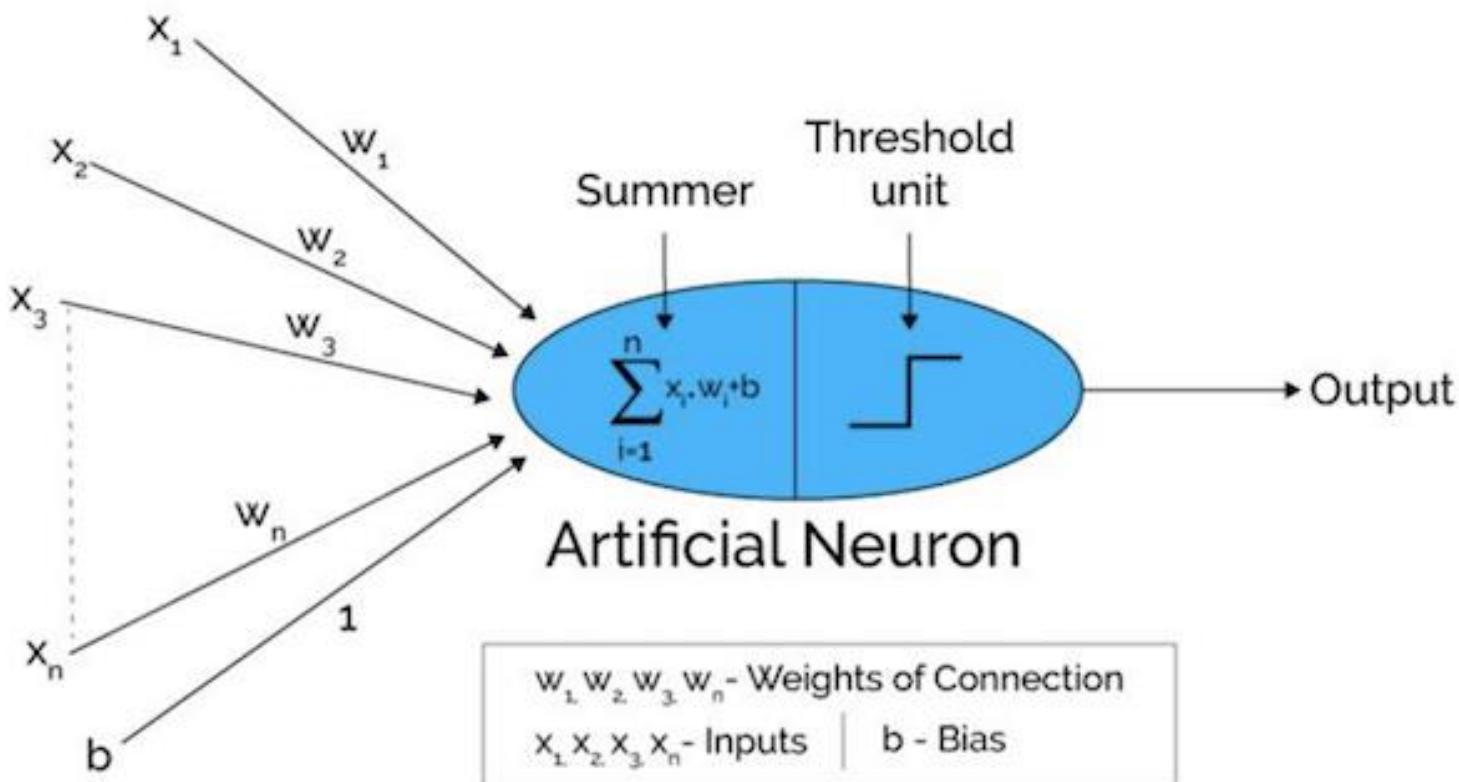
# 神經細胞結構示意圖



# 神經細胞(Neuron)與感知機(Perceptron)



# 感知機(Perceptron)

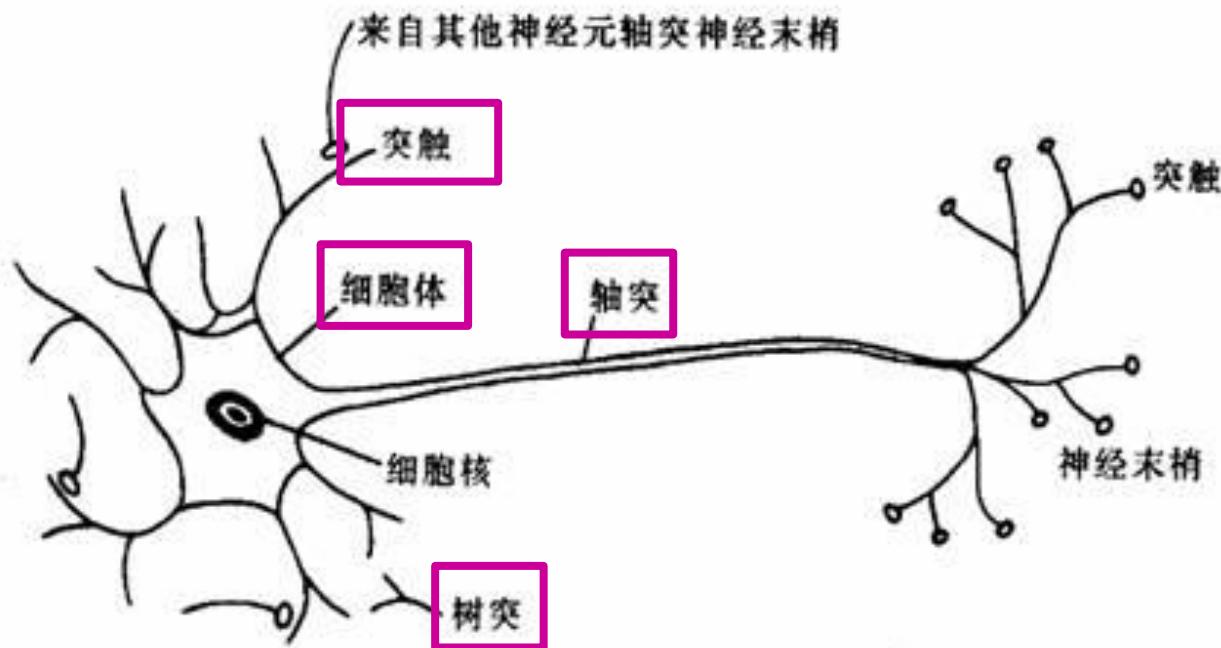


# 感知機(Perceptron)簡介

- 是Frank Rosenblatt在1957年就職於康奈爾航空實驗室（Cornell Aeronautical Laboratory）時所發明的一種人工神經網路。它可以被視為一種最簡單形式的前饋神經網路，是一種二元線性分類器。
- Frank Rosenblatt給出了相應的感知機學習算法，常用的有感知機學習、最小二乘法和梯度下降法。譬如，感知機利用梯度下降法對損失函數進行極小化，求出可將訓練數據進行線性劃分的分離超平面，從而求得感知機模型。

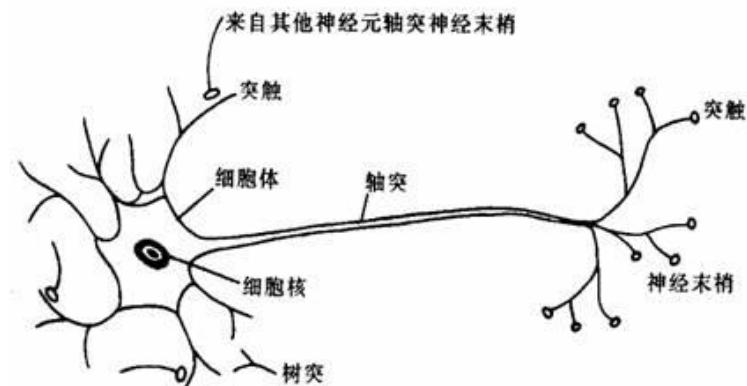
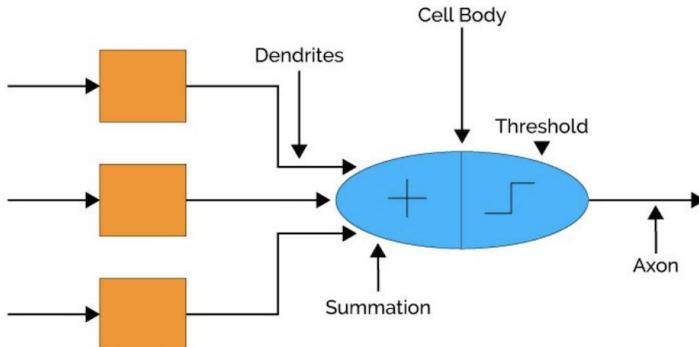
# 感知機(Perceptron)簡介

- 感知機是生物神經細胞的簡單抽象。
- 神經細胞結構大致可分為：樹突、突觸、細胞體及軸突。
- 單個神經細胞可被視為一種只有兩種狀態的機器，
  - 激動時為『是』，而未激動時為『否』。



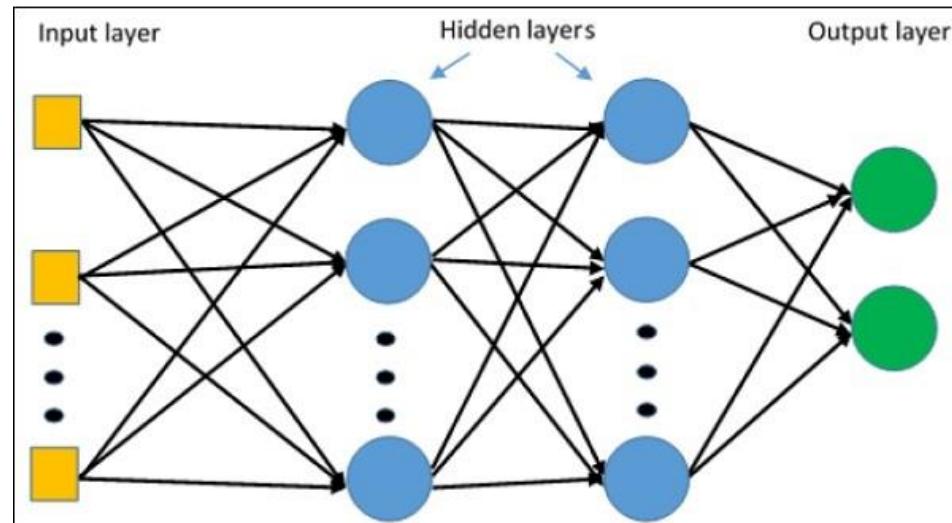
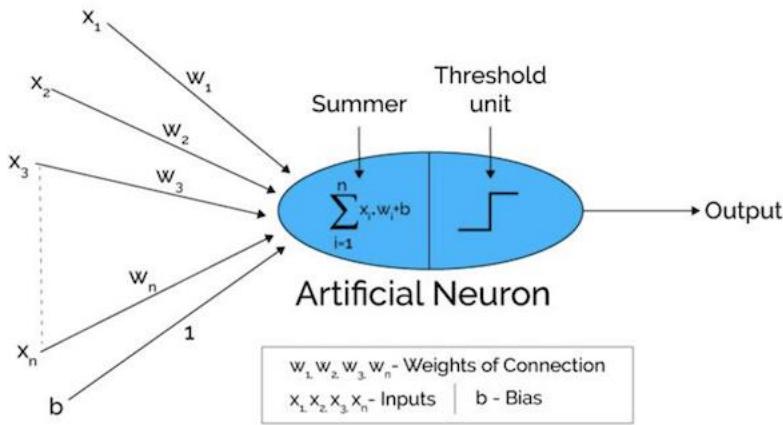
# 感知機(Perceptron)簡介

- 神經細胞的狀態取決於從其它的神經細胞收到的輸入信號量，及突觸的強度（抑制或加強）。
- 當信號量總和超過了某個**閾值**時，細胞體就會激動，產生電脈衝。電脈衝沿著軸突並通過突觸傳遞到其它神經元。
- 為了模擬神經細胞行為，與之對應的感知機基礎概念被提出，
  - 如**權量 (突觸)**、**偏置 (閾值)**及**激活函數 (細胞體)**。



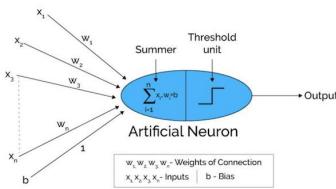
# 感知機(Perceptron)簡介

- 在人工神經網絡領域中，感知機也被指為單層的人工神經網絡，以區別於較複雜的多層感知機（Multi-Layer Perceptron, MLP）。

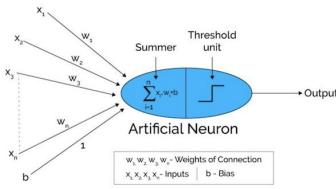
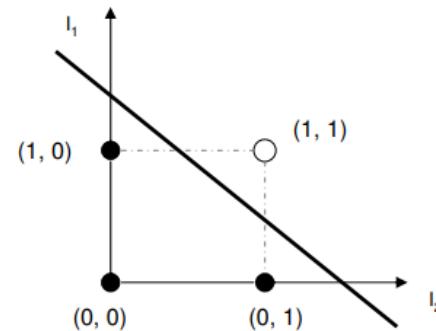


# 感知機(Perceptron)簡介

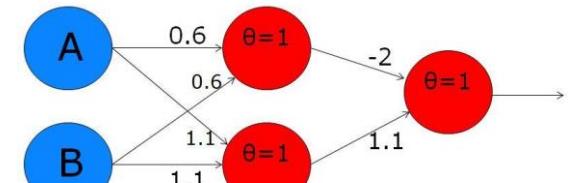
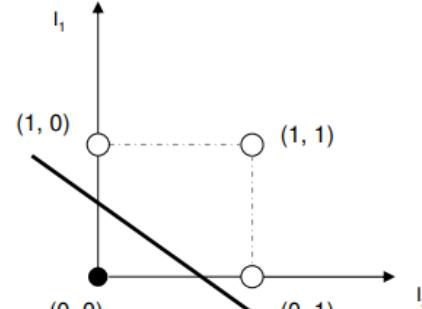
- 作為一種線性分類器，( Single-Layer 單層 ) 感知機可說是最簡單的前向人工神經網絡形式。儘管結構簡單，感知機能夠學習並解決相當複雜的問題。感知機主要的本質缺陷是它不能處理線性不可分問題(linearly inseparable)



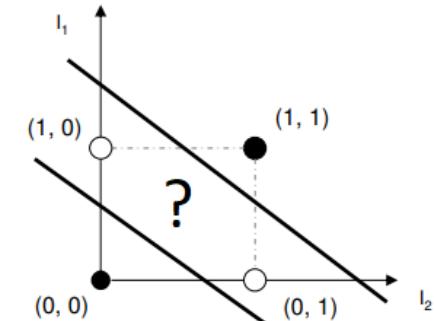
AND		
0	0	0
0	1	0
1	0	0
1	1	1



OR		
0	0	0
0	1	1
1	0	1
1	1	1



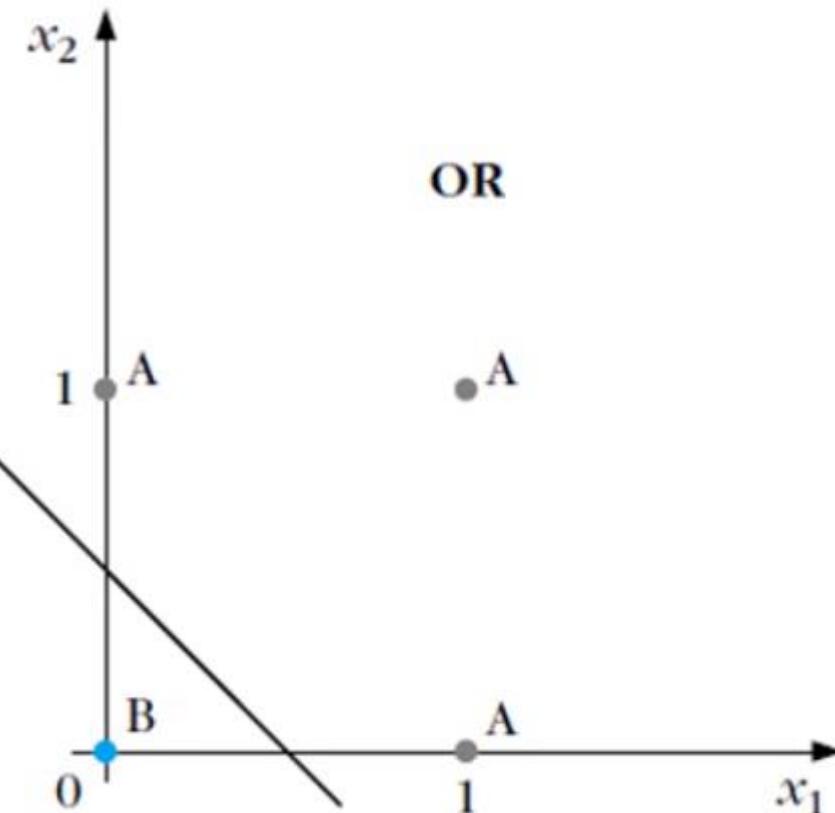
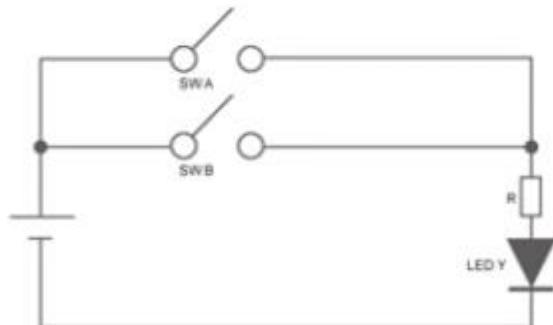
XOR		
0	0	0
0	1	1
1	0	1
1	1	0



# The OR Problem

Table 4.2 Truth Table for AND and OR Problems

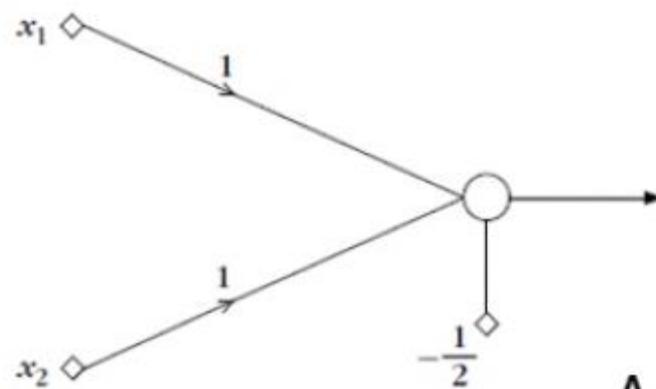
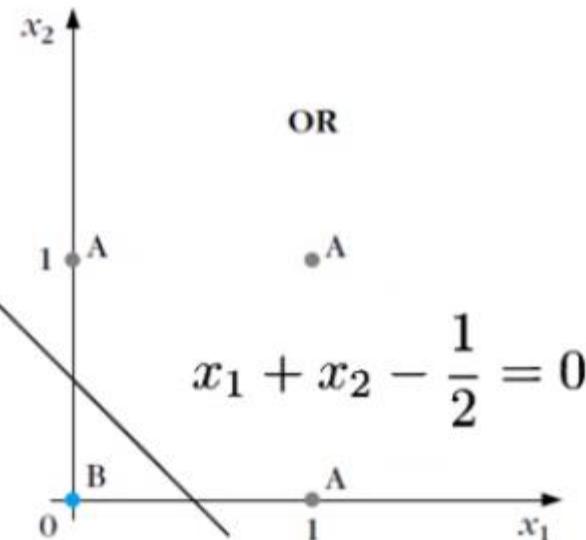
$x_1$	$x_2$	AND	Class	OR	Class
0	0	0	B	0	B
0	1	0	B	1	A
1	0	0	B	1	A
1	1	1	A	1	A



# The OR Problem

**Table 4.2** Truth Table for AND and OR Problems

$x_1$	$x_2$	AND	Class	OR	Class
0	0	0	B	0	B
0	1	0	B	1	A
1	0	0	B	1	A
1	1	1	A	1	A

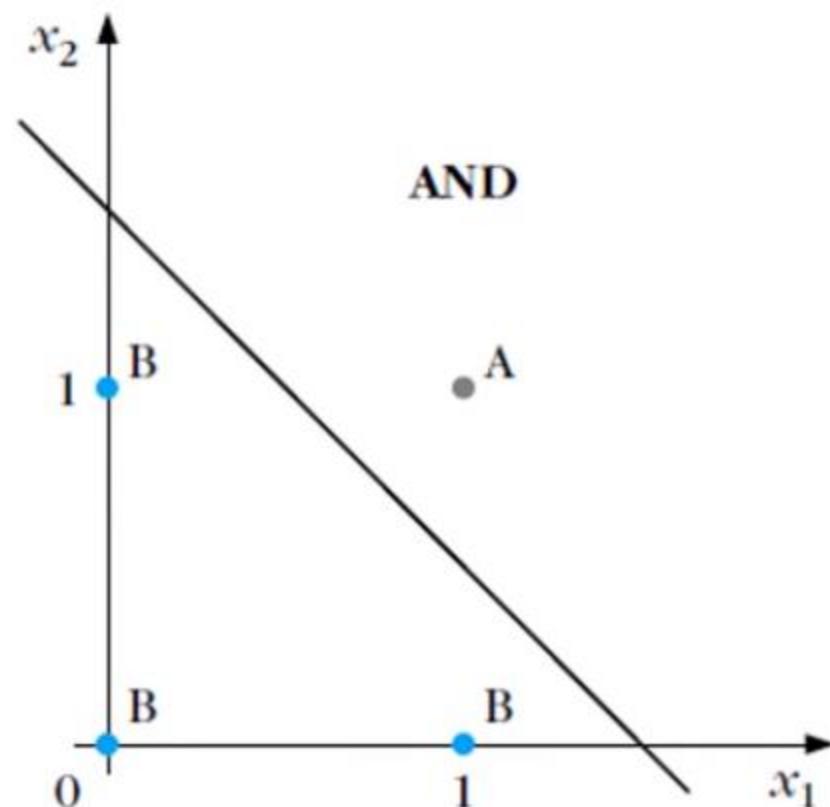
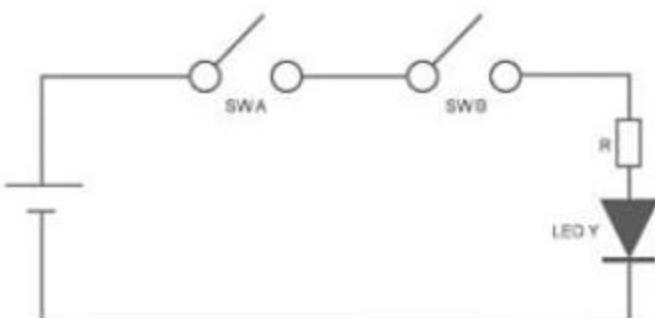


A single-layer perceptron

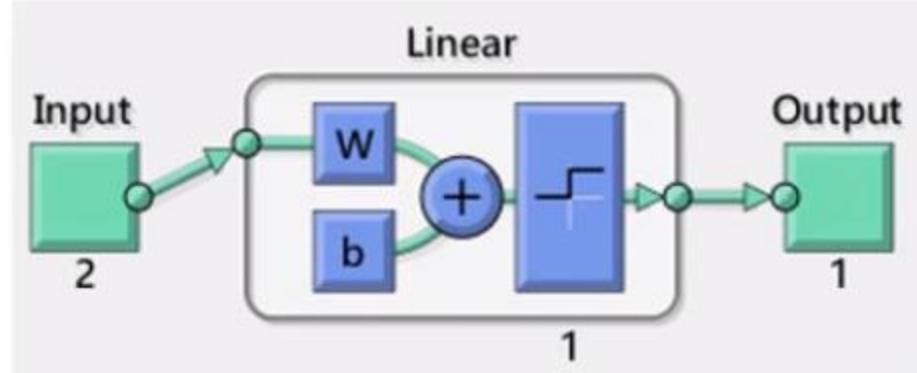
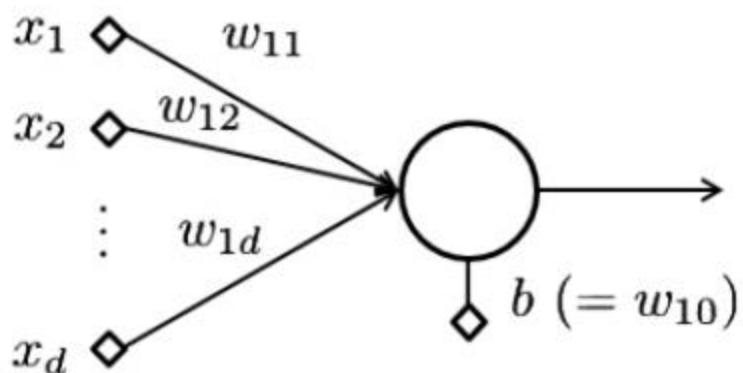
# The AND Problem

**Table 4.2** Truth Table for AND and OR Problems

$x_1$	$x_2$	AND	Class	OR	Class
0	0	0	B	0	B
0	1	0	B	1	A
1	0	0	B	1	A
1	1	1	A	1	A



# Single-layer Perceptron

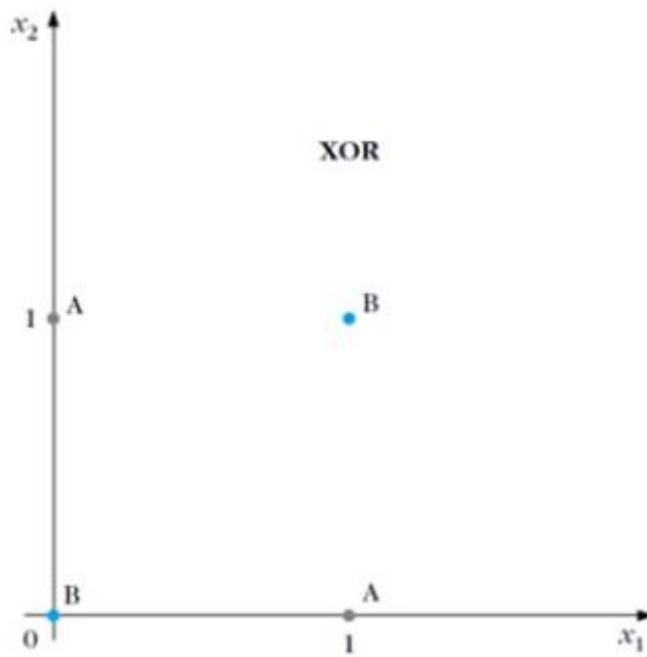


# The XOR Problem

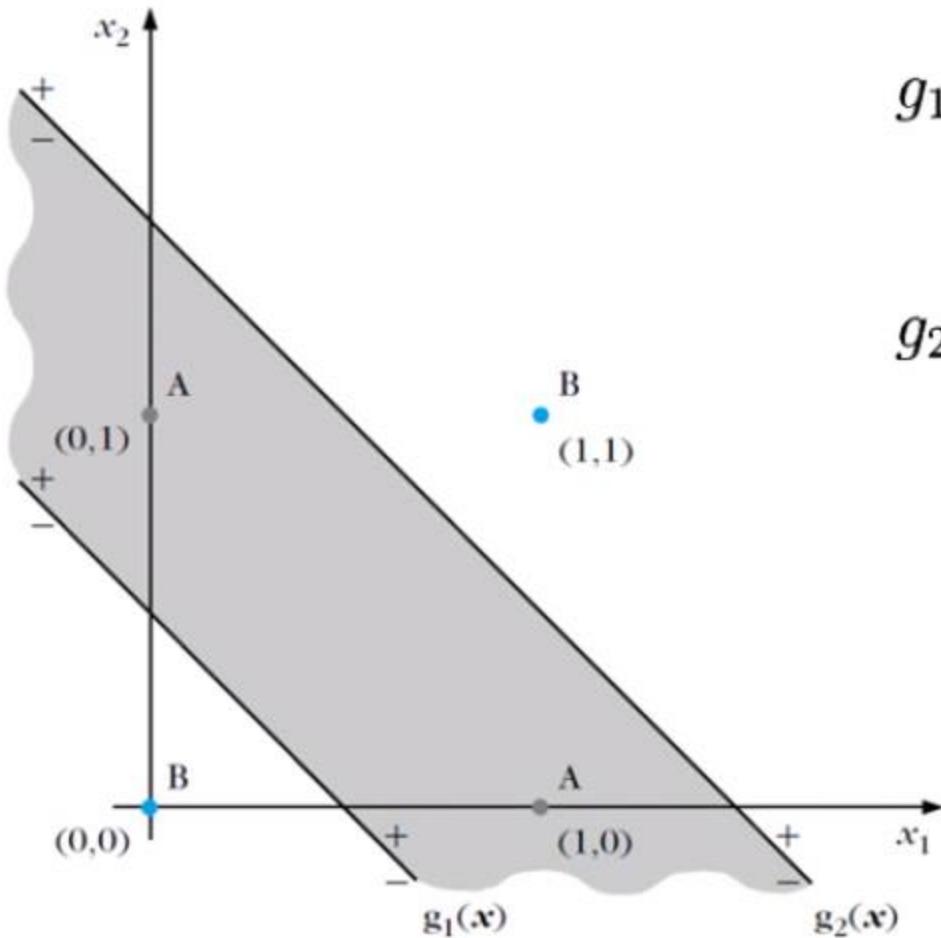
- Exclusive OR (XOR) Boolean function
- **No single straight line** exists that separates the two classes

**Table 4.1** Truth Table for the XOR Problem

$x_1$	$x_2$	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B



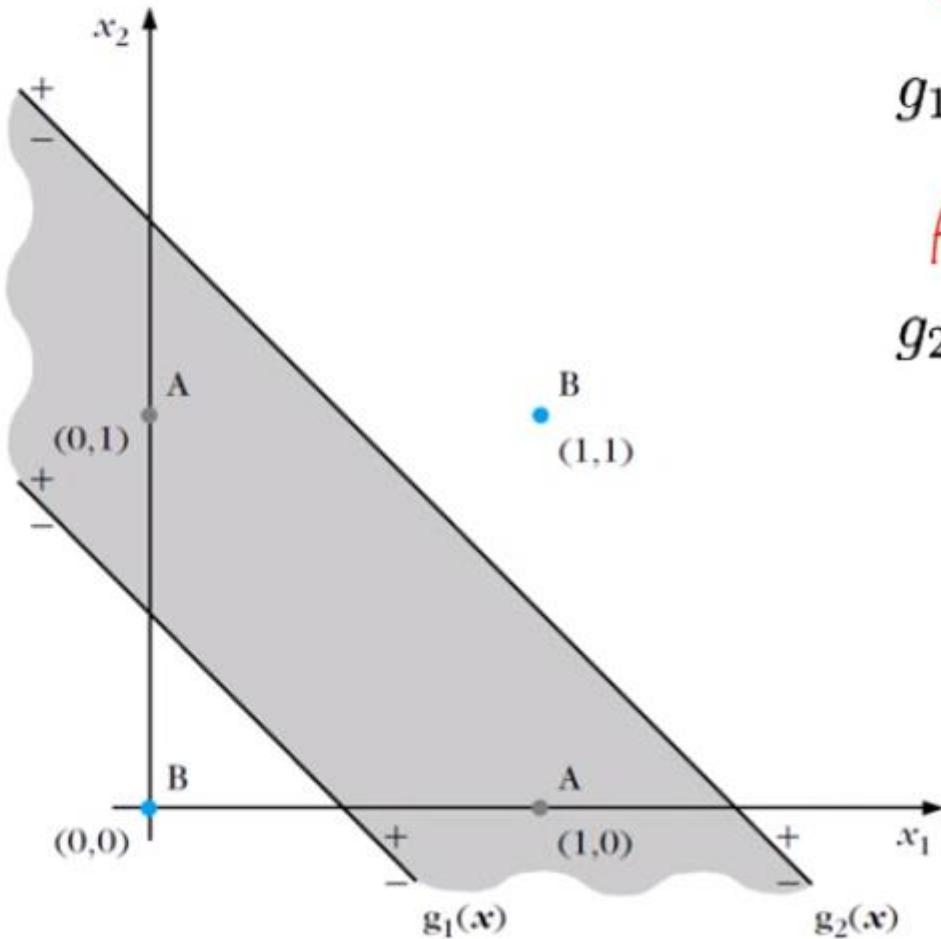
# The XOR Problem



$$g_1(\mathbf{x}) = x_1 + x_2 - \frac{1}{2} = 0$$

$$g_2(\mathbf{x}) = x_1 + x_2 - \frac{3}{2} = 0$$

# The XOR Problem



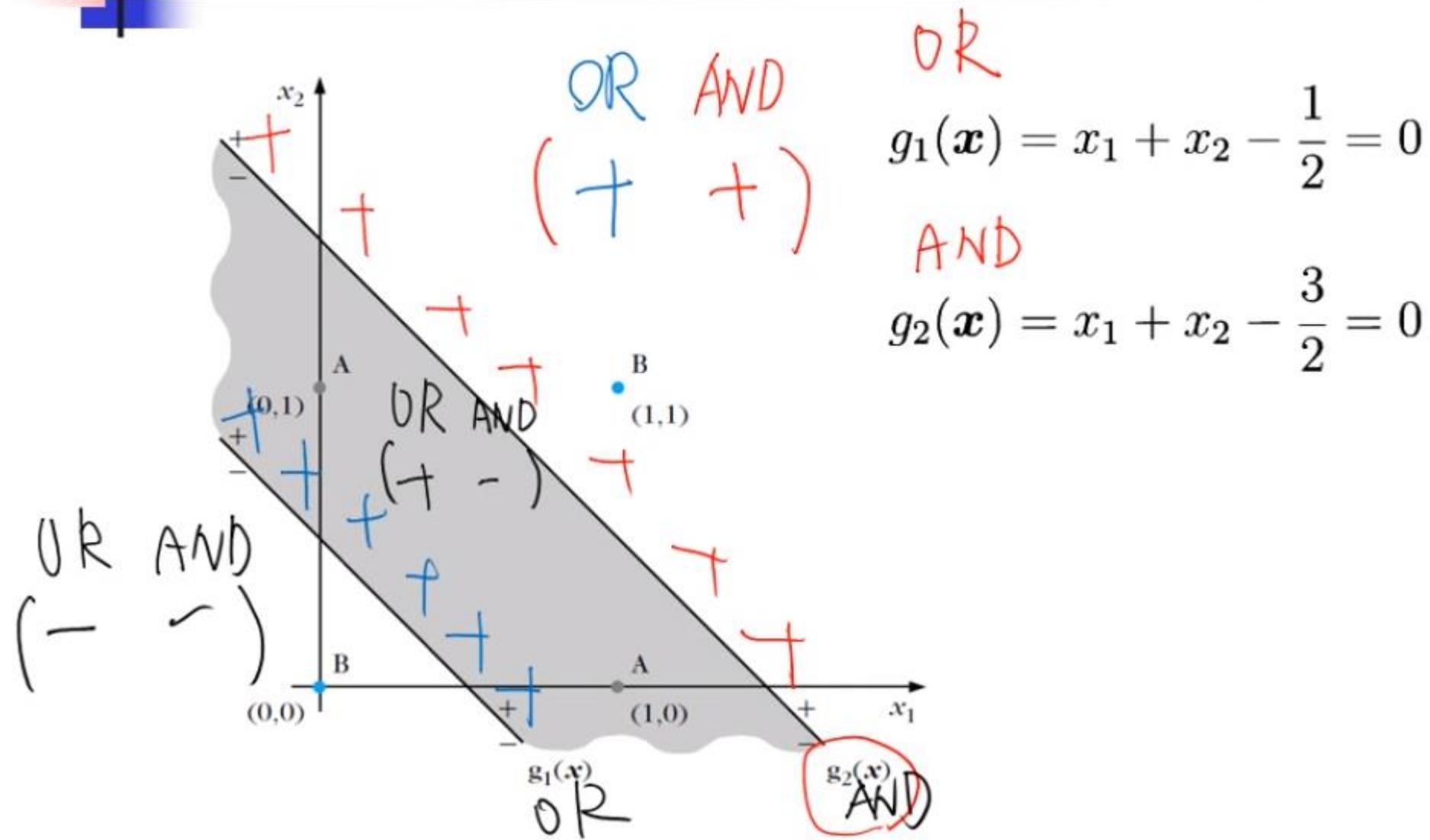
OR

$$g_1(\mathbf{x}) = x_1 + x_2 - \frac{1}{2} = 0$$

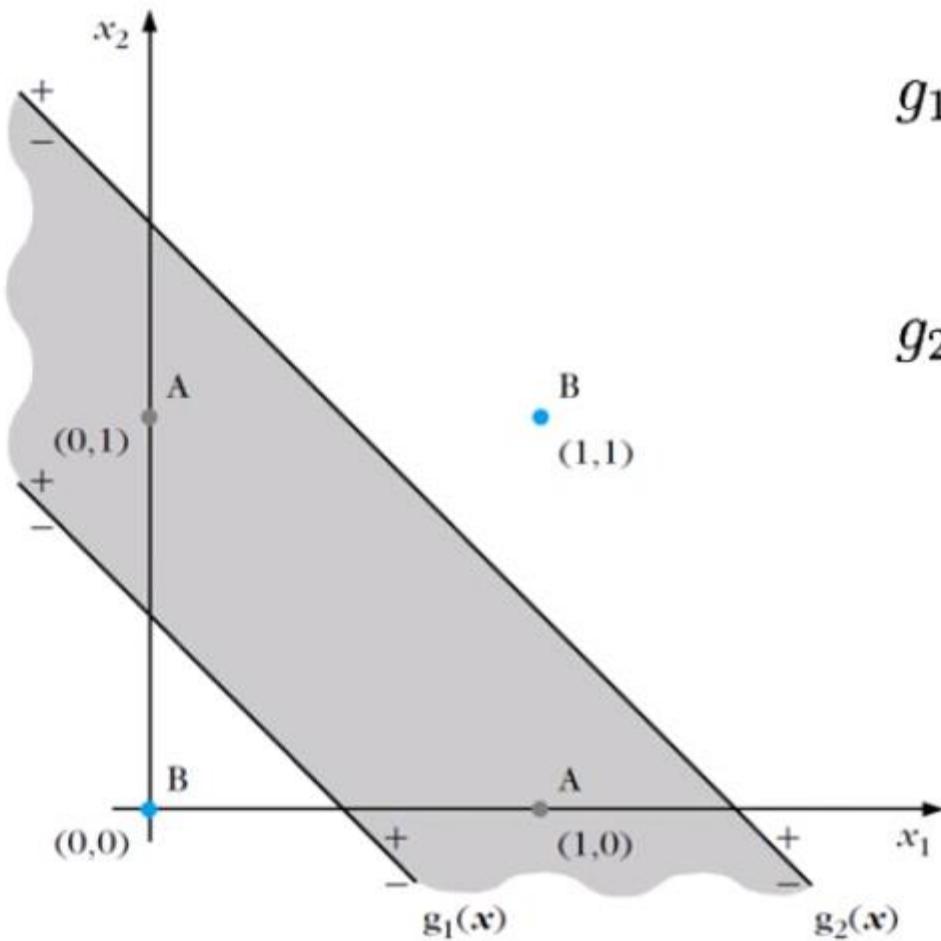
AND

$$g_2(\mathbf{x}) = x_1 + x_2 - \frac{3}{2} = 0$$

# The XOR Problem



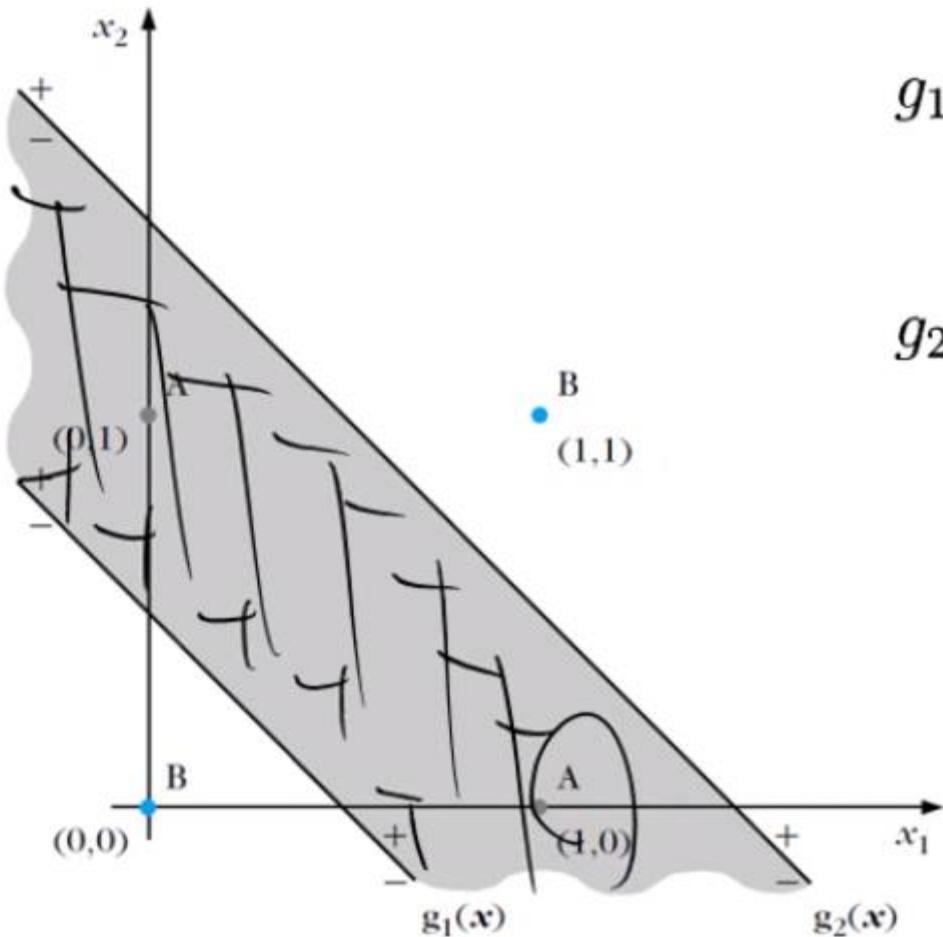
# The XOR Problem



$$g_1(\mathbf{x}) = x_1 + x_2 - \frac{1}{2} > 0$$

$$g_2(\mathbf{x}) = x_1 + x_2 - \frac{3}{2} < 0$$

# The XOR Problem

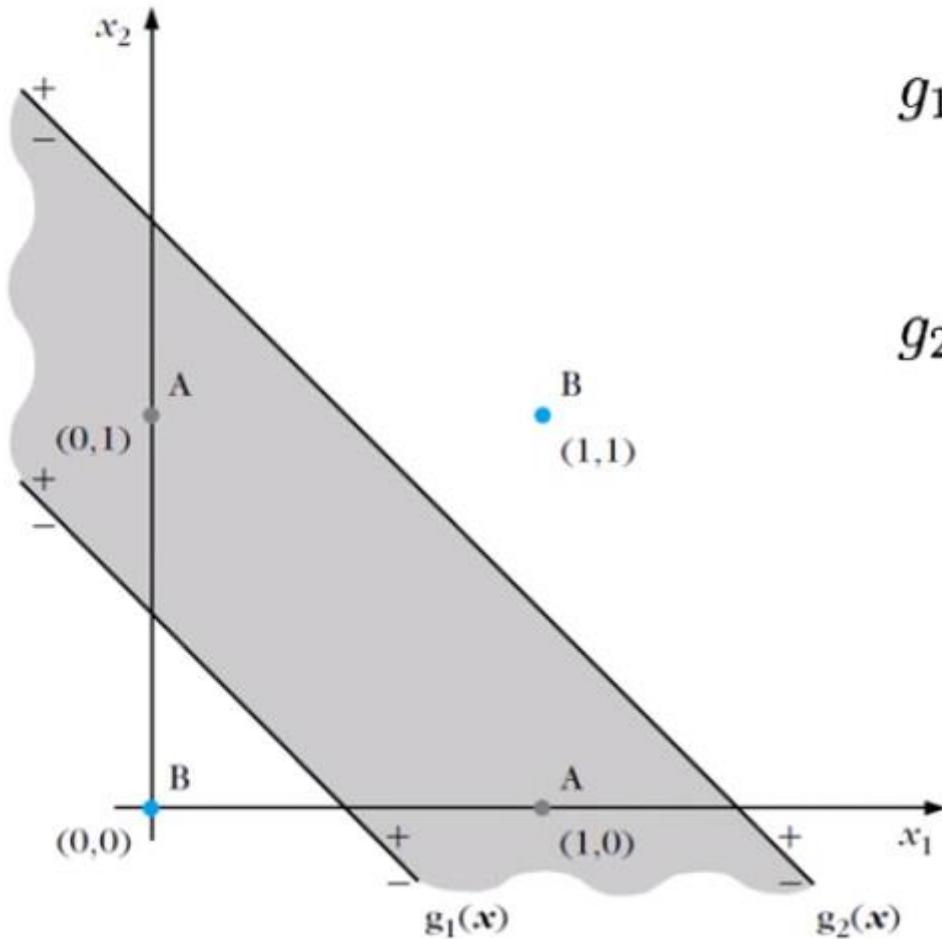


$$g_1(\mathbf{x}) = x_1 + x_2 - \frac{1}{2} > 0$$

$$g_2(\mathbf{x}) = x_1 + x_2 - \frac{3}{2} < 0$$

$\chi \in A$

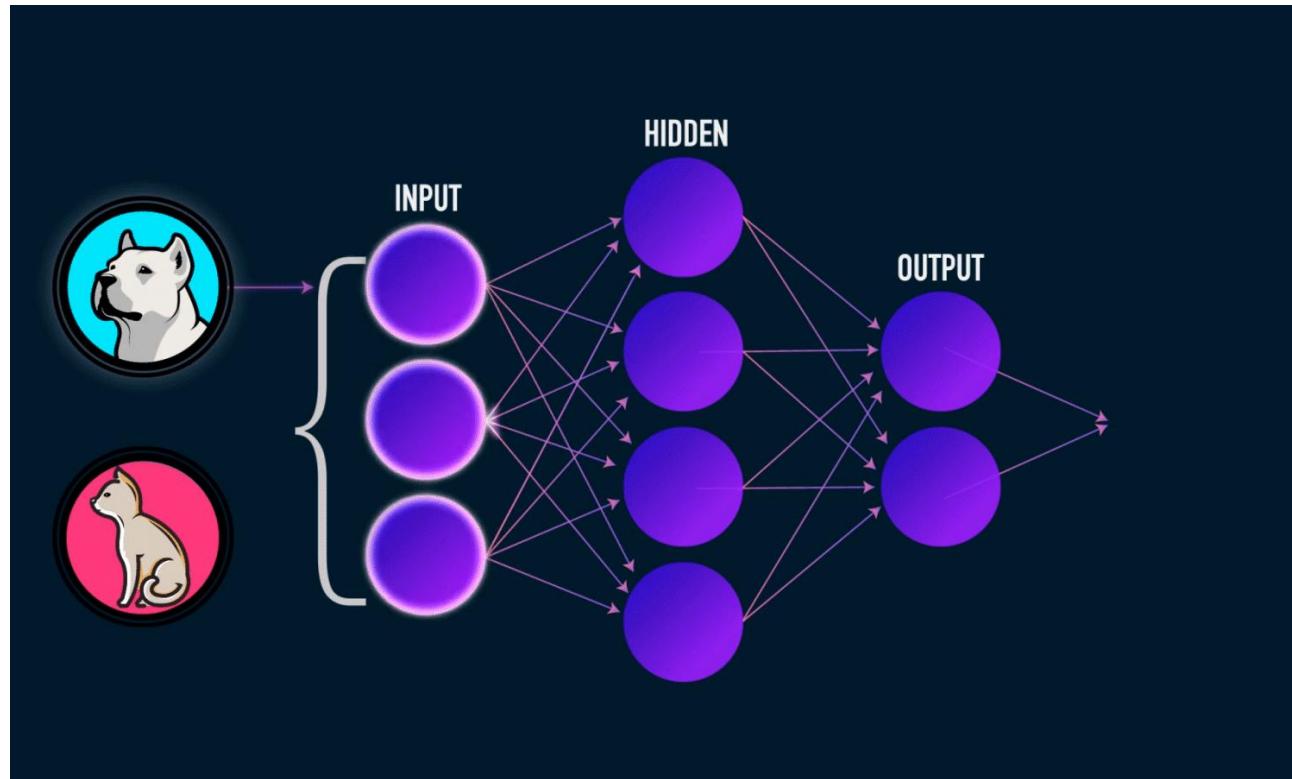
# The XOR Problem



$$g_1(\mathbf{x}) = x_1 + x_2 - \frac{1}{2} < 0$$

$$g_2(\mathbf{x}) = x_1 + x_2 - \frac{3}{2} > 0$$

# 倒傳遞 (backpropagation) 類神經網路



# 倒傳遞 (backpropagation) 類神經網路

- 倒傳遞演算法是多層前饋式類神經網路 (multilayer feed-forward networks) 的學習演算法
- 它疊代式地學習調整權重值來預測資料值組的類別標籤
- 多層前饋式類神經網路由一個輸入層 (input layer)、一個或多個隱藏層 (hidden layer) 以及一個輸出層 (output layer) 所組成，圖8.2 是一個多層前饋式類神經網路的範例。

# 倒傳遞 (backpropagation) 類神經網路

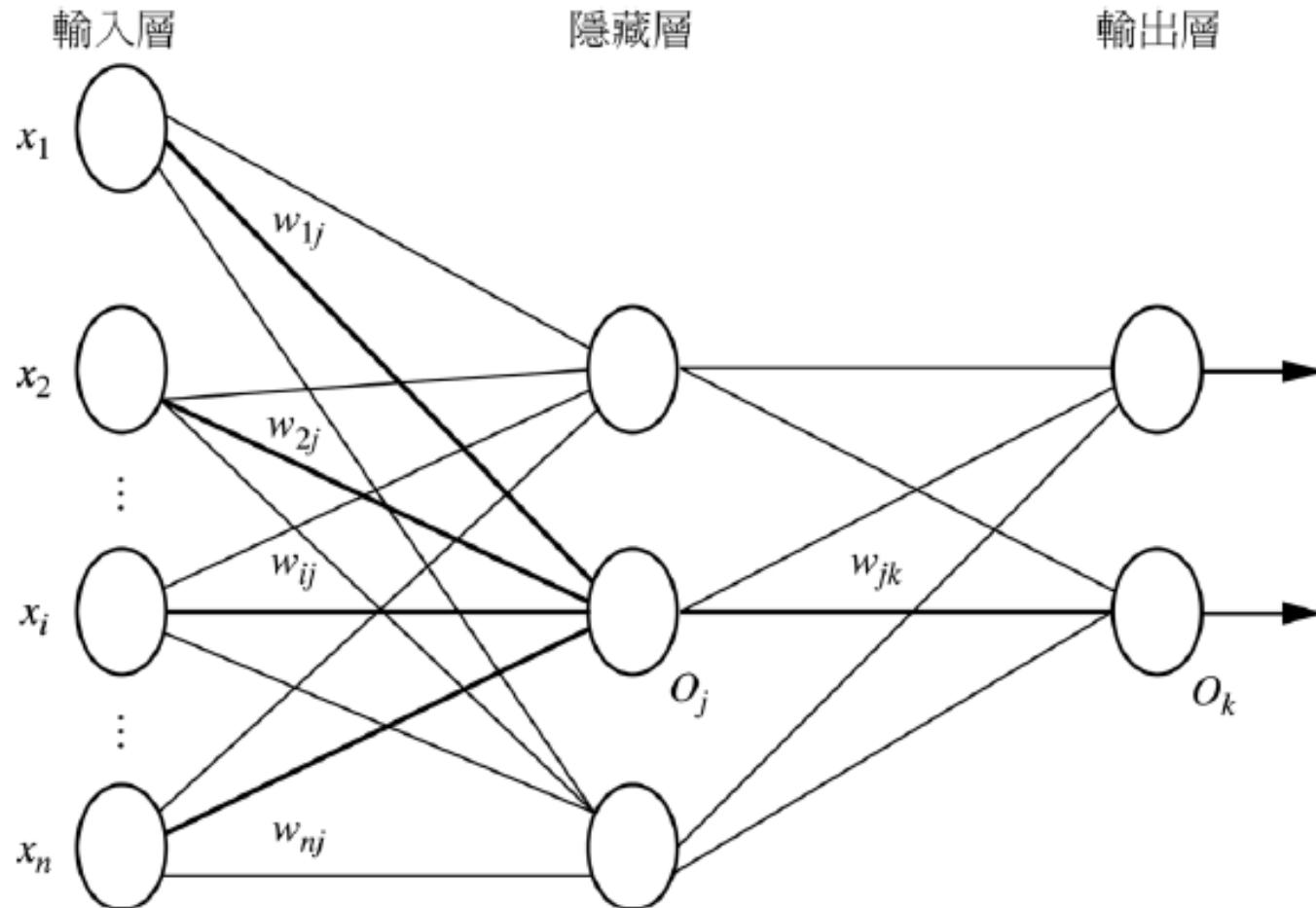


圖 8.2 多層前饋式類神經網路。

# 倒傳遞 (backpropagation) 類神經網路

- 圖8.2 中的多層前饋式類神經網路架構共有兩層的神經單元，
- 所以，我們稱它為兩層式類神經網路 (*two-layer neural network*)。我們並沒有將輸入層納入計算，那是因為輸入層僅僅將資料傳送到下一層。
- 相同地，包含兩層隱藏層的類神經網路，則稱為三層式類神經網路，依此類推。
- 它被稱為前饋式 (*feed-forward*)，是因為沒有網路聯結會繞回至輸入層或是前面的隱藏層。
- 此外，每一個提供輸入的運算單元都與他下一層中的單元是完全連接。

# 倒傳遞 (backpropagation) 類神經網路

- 每一個神經元接收其上一層中單元輸出值的加權總合，做為此神經元的輸入值（如稍後的圖8.4 所示），
- 然後對此加權後的輸入值套用一個非線性激發 (activation) 函數，多層前饋式類神經網路能夠根據輸入值的非線性組合來建立類別預測模型，從統計的觀點來看，它其實是在執行一個非線性迴歸。
- 而多層前饋式類神經網路吸引人的魔力是：只要給予足夠多的隱藏層單元與訓練樣本，它可以近似估計任何函數。

# 倒傳遞 (backpropagation) 類神經網路

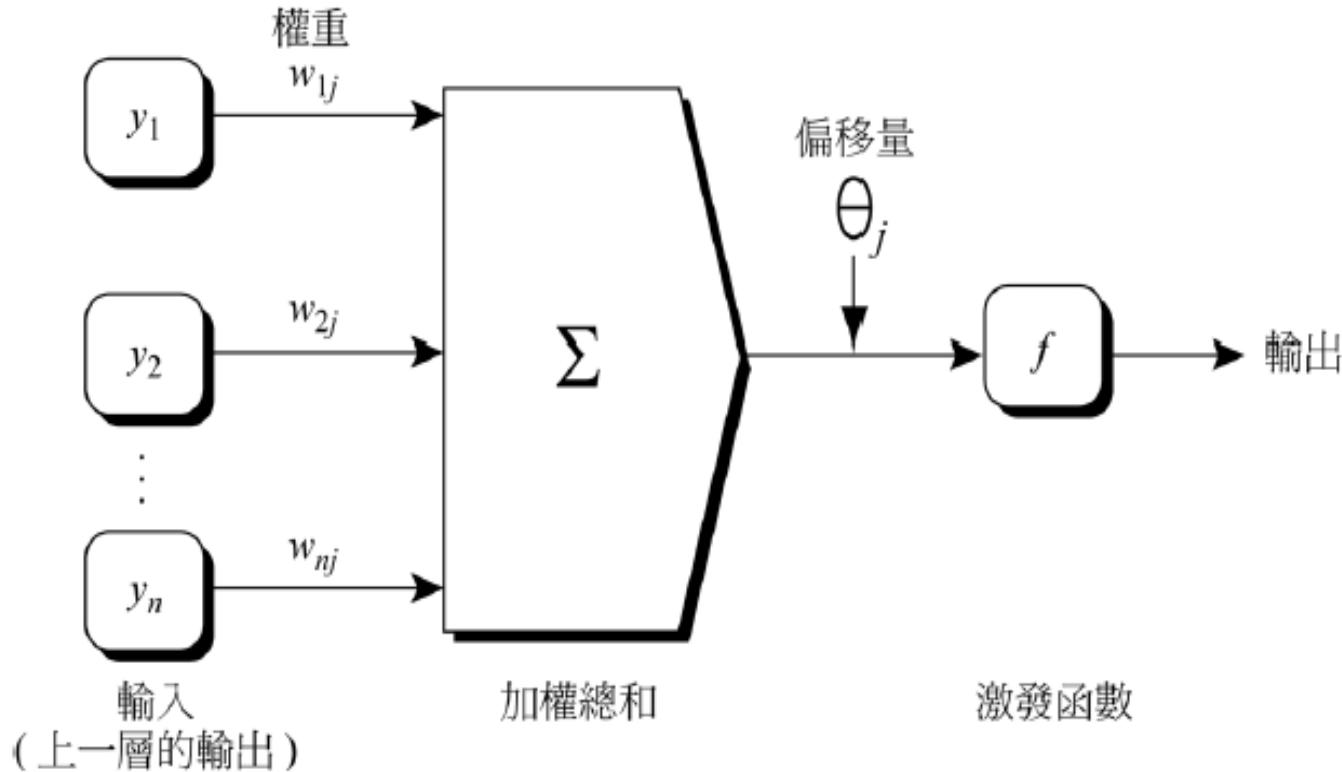


圖 8.4 隱藏層或輸出層中的單元  $j$ : 此單元  $j$  的輸入是前一層的輸出值。這些輸入值與相對應的聯結權重相乘後並進行加總，然後再加上偏移量。最終，將計算結果代入激發函數  $f$ ，即為此單元  $j$  的輸出值。

# 設定類神經網路的拓樸結構

- 在訓練類神經網路之前，使用者必須事先決定網路的拓樸結構 (topology)，包含定義輸入層內的單元數目、隱藏層的數目（如果超過一層的話）、每一個隱藏層內的單元數目，以及在輸出層內的單元數目。
- 對輸入值組中的每一個屬性值進行正規化 (normalizing)，能有助於提升訓練的速度，一般是將連續值屬性正規化至0.0 至1.0 的區間內，對於離散值屬性，則是將它編碼成一個輸入單元對應該屬性中一個屬性值，
- 舉例來說，如果屬性  $A$  共包含三個種可能的屬性值  $\{a_0, a_1, a_2\}$ ，我們可以指派三個輸入單元  $I_0, I_1, I_2$  來代表  $A$  的值，如果  $A = a_1$ ，則  $I_1$  設定為 1，而其餘的  $I_0$  與  $I_2$  則設定為 0，依此類推。

# 設定類神經網路的拓樸結構

- 類神經網路可以用於分類（預測輸入值組的類別標籤）與數值預測（預測一個連續值的輸出），對於分類問題，一個輸出單元可以用來代表兩個類別，輸出值為1 代表一個類別，輸出值為0 則代表另一個類別，當類別數目超過兩個以上時，則是用一個輸出單元對應一個類別。

# 設定類神經網路的拓樸結構

- 並沒有明確的規則告訴我們如何決定隱藏層內的“最佳”的單元數目，
  - 網路拓樸結構設定通常採用嘗試錯誤 (trial-and-error) 的方法，而且網路拓樸結構會影響網路訓練後的正確率，
- 此外，網路聯結上的權重初始值也會影響網路訓練後的正確率。當訓練後的類神經網路的預測正確率是無法接受時，通常會設定不同的拓樸結構與權重初始值，然後再重新開始訓練。
- 交叉驗證 (cross-validation) 可用來估計類神經網路的正確率，以決定是否已找到一個可接受的網路。

# 倒傳遞演算法

- 倒傳遞 (backpropagation) 演算法是一種疊代式的訓練方法，
- 每一次疊代時，它比對每一個資料值組的預測結果與真實的目標值 (*target value*) 之間的誤差，目標值可以是此訓練資料值組的類別標籤（對分類問題）或是一個連續數值（對數值預測問題），
- 對每一個訓練資料值組，我們藉由調整網路聯結上的權重值來最小化網路預測結果與真正目標值間的均方誤差值 (*mean squared error*)，它以向後倒傳遞的方向來修正這些權重值（因此稱為倒傳遞），
- 也就是說，它從輸出層開始，往後饋方向經過每一個隱藏層，直到第一個隱藏層為止，雖然無法證明，但這些權重值通常最終都會收斂，然後演算法停止。
- 圖8.3 顯示倒傳遞演算法的執行步驟，第一次看到類神經網路的學習演算法，你可能會覺得這些步驟很棘手，不過你一但熟悉之後，就會發現這些步驟的概念其實是很簡單的，

# 倒傳遞演算法

運算法：倒傳遞法是多層前饋式類神經網路應用於分類與數值預測時的學習演算法。

輸入：

- $D$  為資料集，它包含訓練資料值組與其對應的目標值；
- $l$  為學習速率；
- $network$  為多層前饋式網路。

輸出：一個訓練過的類神經網路。

方法：

- (1) 設定所有  $network$  中的聯結權重與單元偏移量的初始值；
- (2) **while** 結束條件未滿足 {
- (3)     **for**  $D$  中每個資料值組  $X$  {
- (4)         // 向前傳遞輸入值
- (5)         **for** 輸入層中的每個單元  $j$  {
- (6)              $O_j = I_j$ ; // 在輸入層中的單元，其輸出值便是它的輸入值
- (7)         **for** 每個隱藏層或輸出層中單元  $j$  {
- (8)              $I_j = \sum_i w_{ij} O_i + \theta_j$ ; // 根據前一層的輸出值來計算單元  $j$  的輸入值
- (9)              $O_j = \frac{1}{1+e^{-I_j}}$ ; } // 計算單元  $j$  的輸出值
- (10)         // 倒傳遞錯誤值

# 倒傳遞演算法

```
(11)          for 輸出層中每個單元  $j$ 
(12)           $Err_j = O_j(1 - O_j)(T_j - O_j); //$  計算錯誤值
(13)          for 每個隱藏層中的每個單元  $j$ ，從最後一個隱藏層到第一個隱藏層
(14)           $Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk}; //$  根據上一個隱藏層  $k$  來計算錯
               誤
(15)          for network 中的每個聯結權重  $w_{ij}$  {
(16)               $\Delta w_{ij} = (l)Err_j O_i; //$  計算權重修正值
(17)               $w_{ij} = w_{ij} + \Delta w_{ij}; } //$  權重更新
(18)          for network 中每個偏移量  $\theta_j$  {
(19)               $\Delta \theta_j = (l)Err_j; //$  計算偏移量修正值
(20)               $\theta_j = \theta_j + \Delta \theta_j; } //$  偏移量更新
(21)      }
```

圖 8.3 倒傳遞演算法。

## 範例 8.1 ➤ 倒傳遞學習演算法的計算範例

圖 8.5 為一個多層前饋式類神經網路的範例，假設學習速率為 0.9，初始的權重值及偏移量顯示在表格 8.1 中。第一個餵入的訓練資料值組為  $X = (1, 0, 1)$ ，它對應的類別標籤為 1。

此範例說明當此值組餵入網路後，倒傳遞演算法詳細的計算步驟，首先，計算網路中每一個單元的輸入值與輸出值，這些輸入 / 輸出數值的計算顯示在表格 8.2 中，接著，計算每一個單元的錯誤值，並將錯誤值向後傳遞，這些錯誤值的計算顯示在表格 8.3 中，最後，進行權重值與偏移量的更新，其計算過程顯示在表格 8.4 中。

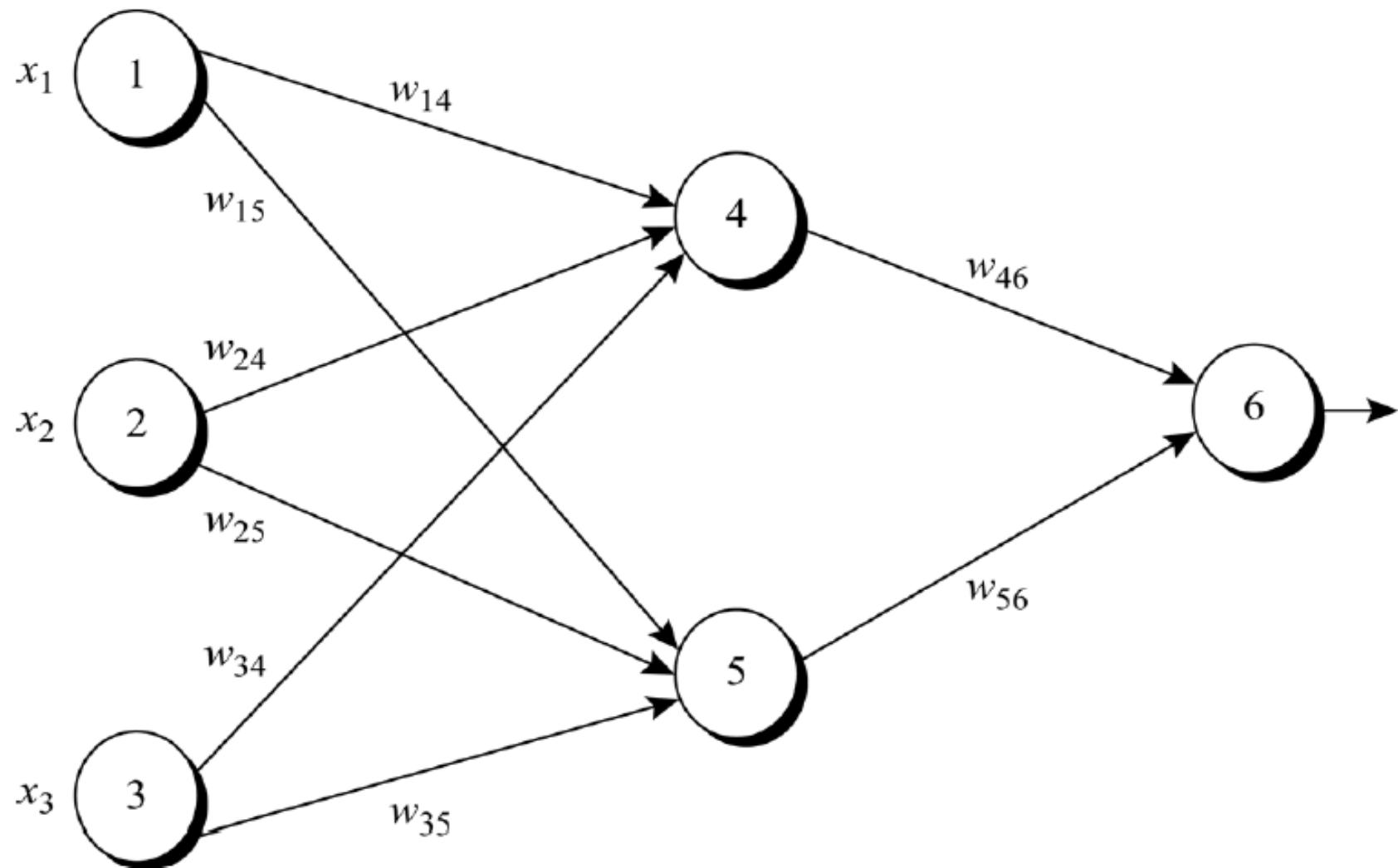


圖 8.5 多層前饋式類神經網路的範例。

表 8.1 輸入、初始權重值與偏移量

$x_1$	$x_2$	$x_3$	$w_{14}$	$w_{15}$	$w_{24}$	$w_{25}$	$w_{34}$	$w_{35}$	$w_{46}$	$w_{56}$	$\theta_4$	$\theta_5$	$\theta_6$
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

11個參數預設值

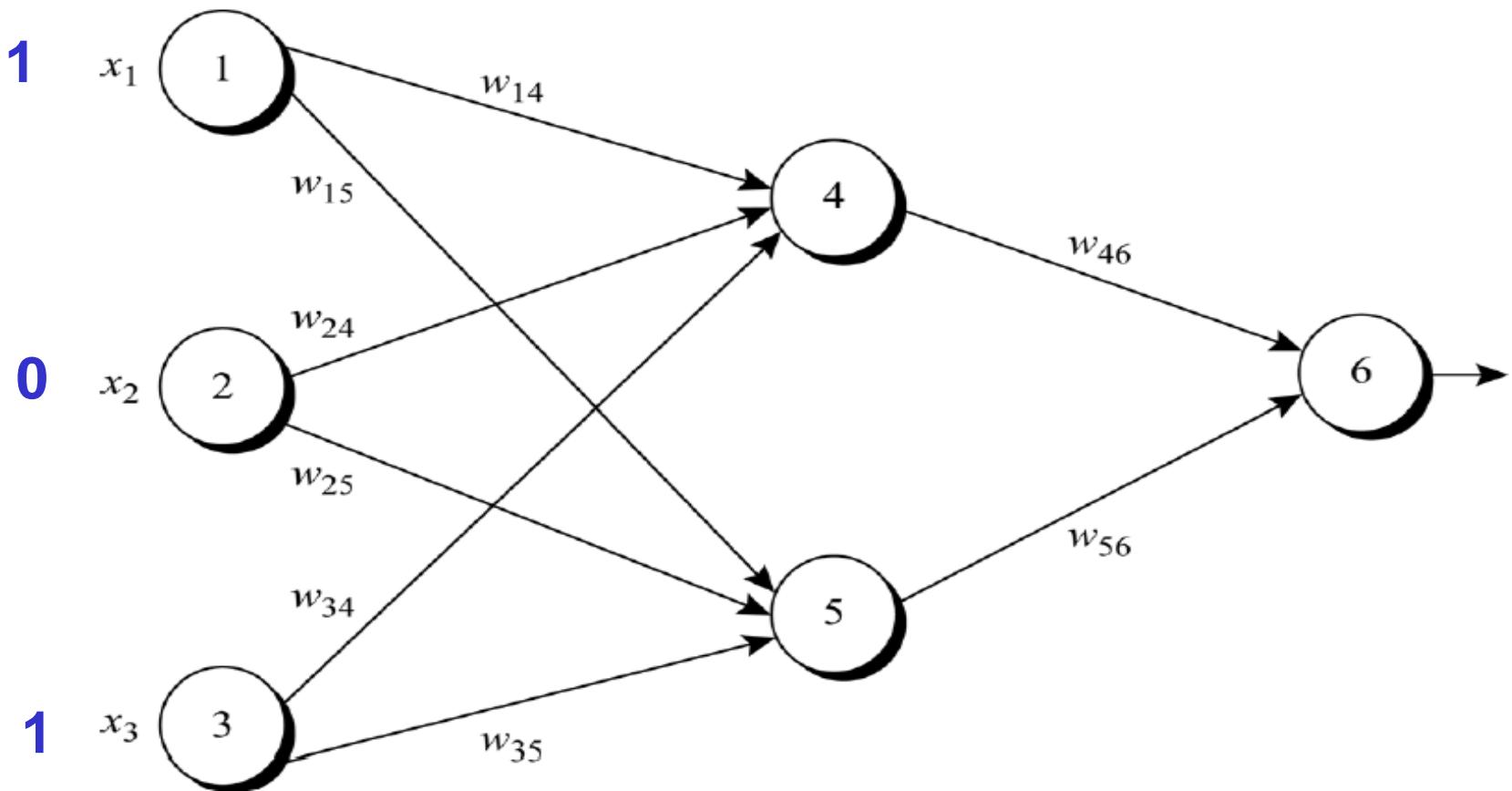


圖 8.5 多層前饋式類神經網路的範例。

表 8.1 輸入、初始權重值與偏移量

$x_1$	$x_2$	$x_3$	$w_{14}$	$w_{15}$	$w_{24}$	$w_{25}$	$w_{34}$	$w_{35}$	$w_{46}$	$w_{56}$	$\theta_4$	$\theta_5$	$\theta_6$
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

表 8.1 輸入、初始權重值與偏移量

$x_1$	$x_2$	$x_3$	$w_{14}$	$w_{15}$	$w_{24}$	$w_{25}$	$w_{34}$	$w_{35}$	$w_{46}$	$w_{56}$	$\theta_4$	$\theta_5$	$\theta_6$
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

表 8.2 輸入值與輸出值的計算

單元 $j$	輸入 $I_j$	輸出 $O_j$
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

$$I_j = \sum_i w_{ij} O_i + \theta_j \quad O_j = \frac{1}{1 + e^{-I_j}}$$

1. 計算  $I_4, I_5$
2. 計算  $O_4, O_5$
3. 計算  $I_6$
4. 計算  $O_6$

表 8.3 計算每個單元的錯誤值

單元 $j$	錯誤值 $Err_j$
6	$(0.474)(1 - 0.474)(1 - 0.474) = 0.1311$
5	$(0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$

### Output layer

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

### Hidden layer

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$

$$w_{ij} = w_{ij} + (l) Err_j O_i \quad \theta_j = \theta_j + (l) Err_j$$

1. 計算  $Err_6$

2. 計算  $\theta_6$

3. 計算  $w_{4,6}, w_{5,6}$

表 8.4 權重值與偏移量更新的計算

權重或偏移量	新值
$w_{46}$	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
$w_{56}$	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
$w_{14}$	$0.2 + (0.9)(-0.0087)(1) = 0.192$
$w_{15}$	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
$w_{24}$	$0.4 + (0.9)(-0.0087)(0) = 0.4$
$w_{25}$	$0.1 + (0.9)(-0.0065)(0) = 0.1$
$w_{34}$	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
$w_{35}$	$0.2 + (0.9)(-0.0065)(1) = 0.194$
<b>Output layer</b>	
$Err_j = O_j(1 - O_j)(T_j - O_j)$	
$\theta_6$	$0.1 + (0.9)(0.1311) = 0.218$
$\theta_5$	$0.2 + (0.9)(-0.0065) = 0.194$
$\theta_4$	$-0.4 + (0.9)(-0.0087) = -0.408$
<b>Hidden layer</b>	
$Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk}$	<ol style="list-style-type: none"> <li>計算 <math>Err_4, Err_5</math></li> <li>計算 <math>\theta_4, \theta_5</math></li> <li>計算 <math>w_{1,4}, w_{2,4}, w_{3,4}, w_{1,5}, w_{2,5}, w_{3,5}</math></li> </ol>

$$w_{ij} = w_{ij} + (l)Err_j O_i \quad \theta_j = \theta_j + (l)Err_j$$

# 如何使用訓練的類神經網路分類器

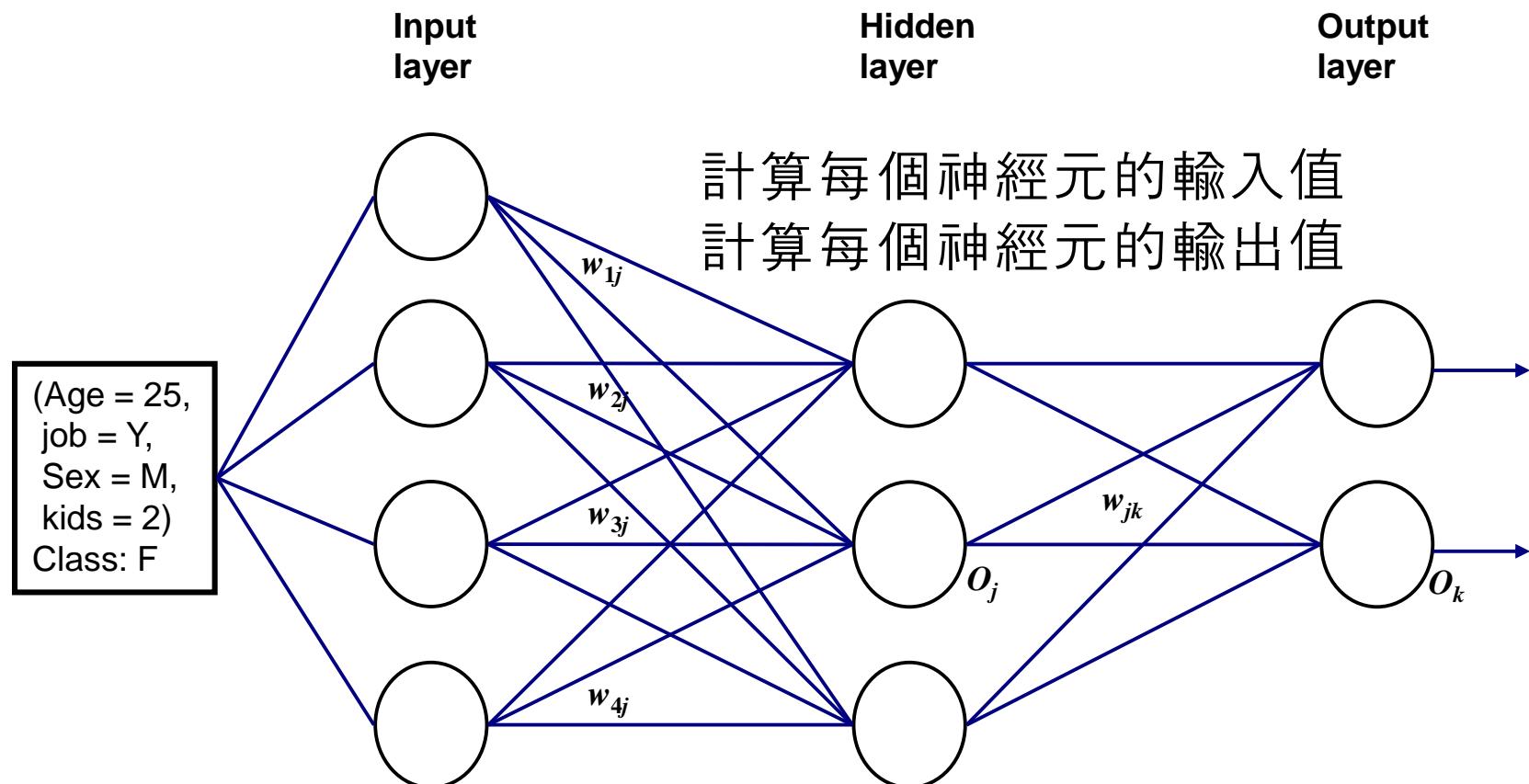
- 「如何利用訓練好的類神經網路來分類輸入資料值組呢？」
- 為了分類一個未知的資料值組 $X$ ，將此值組餵入訓練好的類神經網路，然後計算每一個單元的輸入值與輸出值（不需要倒傳遞錯誤值），
- 如果是一個輸出單元（節點）對應一個類別標籤，則輸出值最高的單元所對應的類別標籤，即為類神經網路預測此資料值組 $X$ 的類別標籤。
- 如果只有一個輸出單元（節點），此時輸出值大於0.5 便代表資料值組 $X$ 屬於正類別，
- 若輸出值小於0.5 則代表資料值組屬於負類別。
- 有數種改良式倒傳遞演算法被提出，包含能夠動態調整網路拓樸結構、學習速率以及其他參數，以及使用不同的錯誤函數。

# 倒傳遞網路之學習過程

- 步驟1：確定網路架構，決定網路層數及各層間神經元數目。
- 步驟2：隨機設定權數與偏權值之初始值於之間。
- 步驟3：隨機選取一訓練組輸入net。
- 步驟4：調整權數及偏權值。
- 步驟5：回到步驟3，直至網路收斂為止。

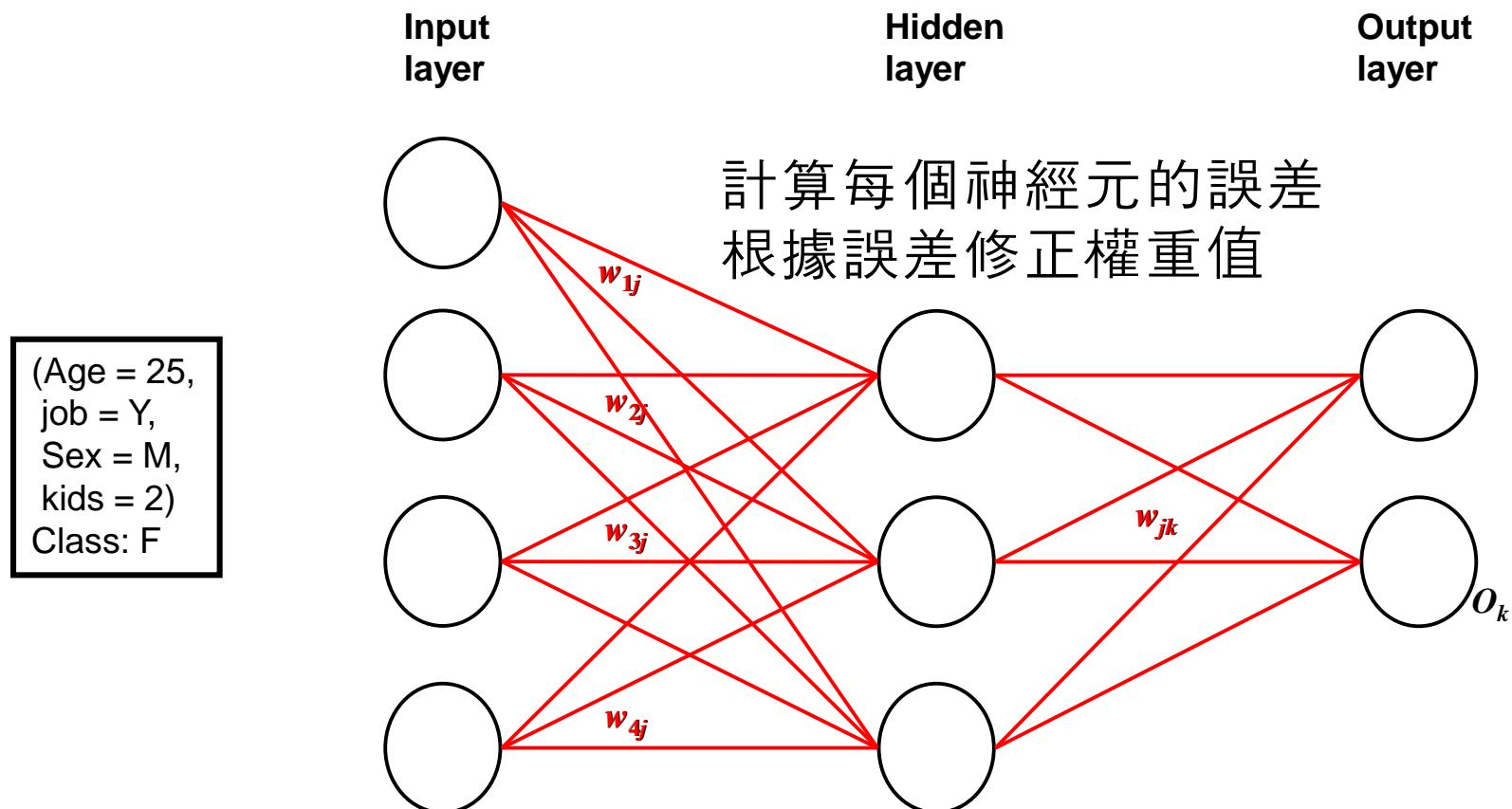
# Backpropagation

- A multilayer feed-forward network

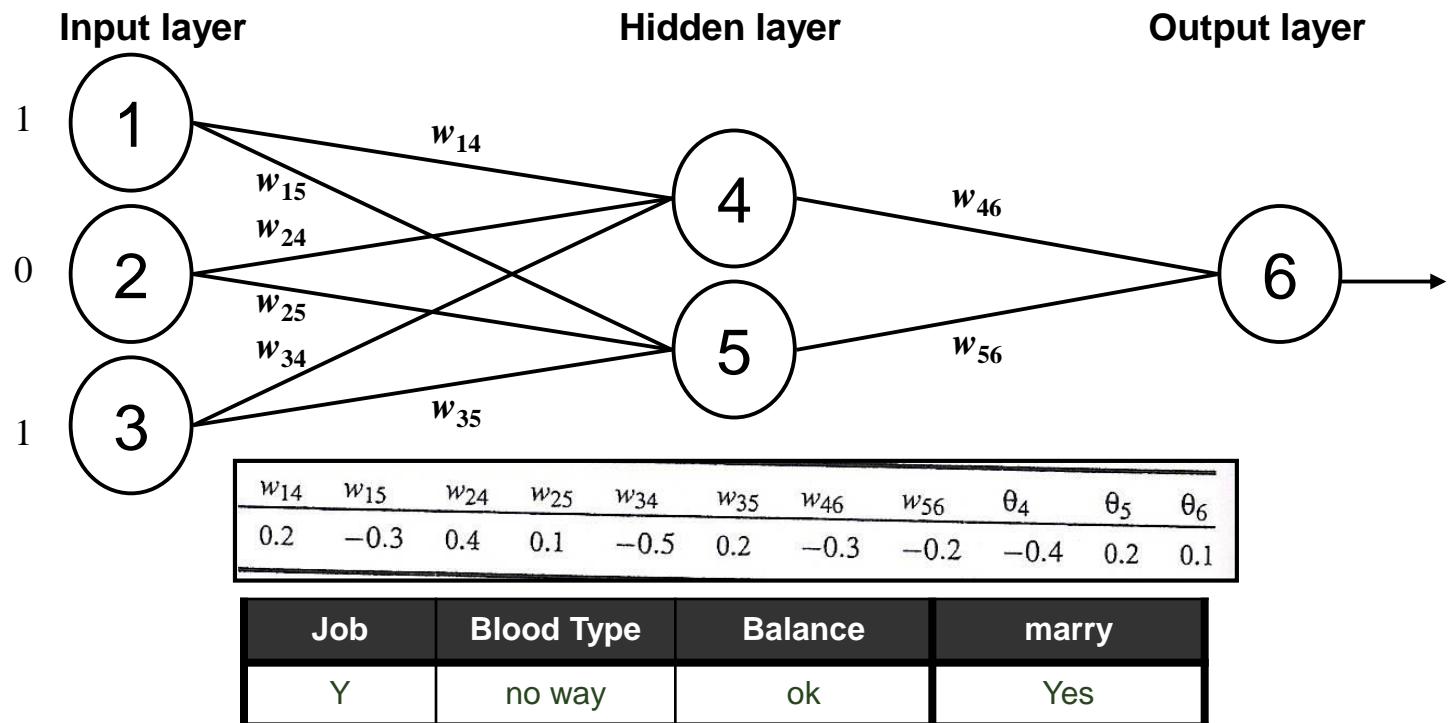


# Backpropagation

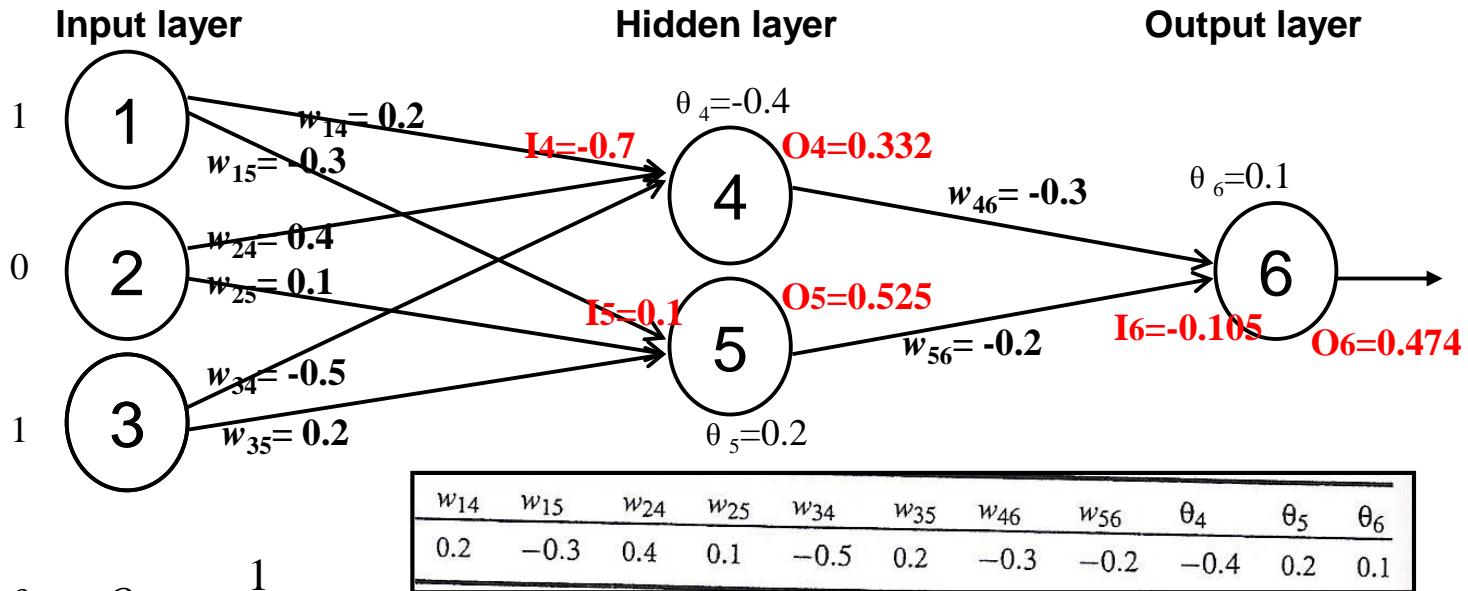
- A multilayer feed-forward network



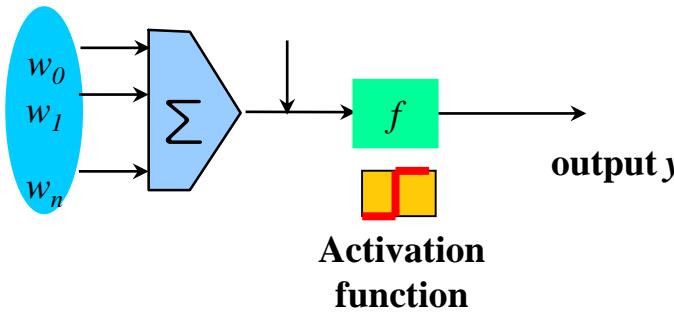
# Backpropagation



# Backpropagation

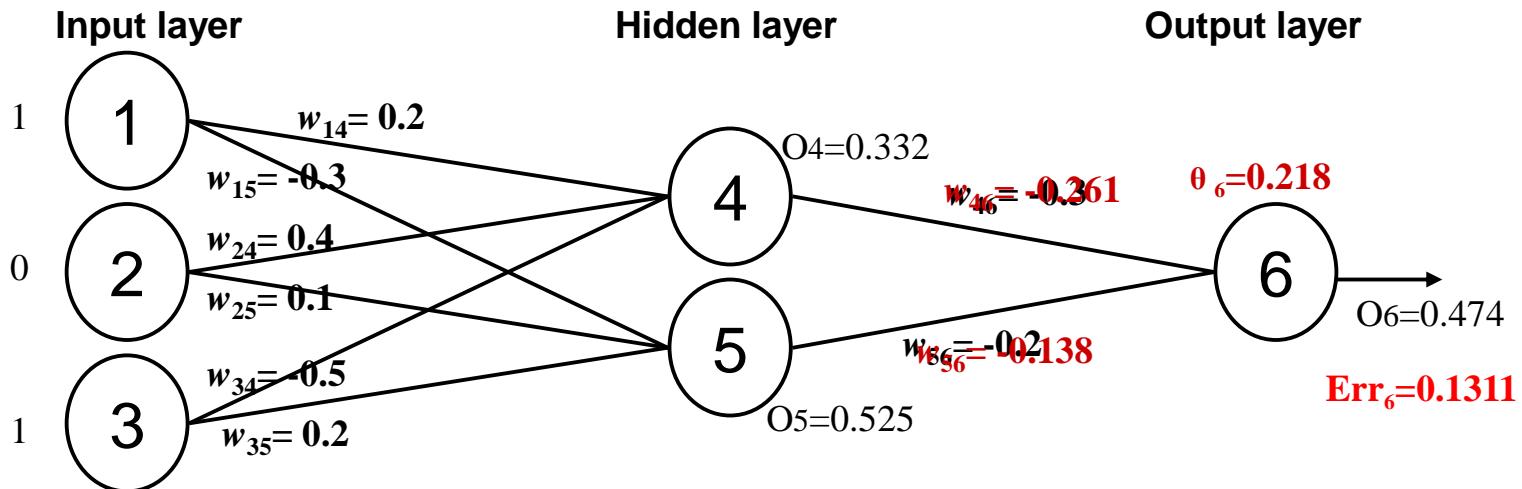


$$I_j = \sum_i w_{ij} O_i + \theta_j \quad O_j = \frac{1}{1+e^{-Ij}}$$



1. 計算  $I_4, I_5$
2. 計算  $O_4, O_5$
3. 計算  $I_6$
4. 計算  $O_6$

# Backpropagation



## Output layer

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

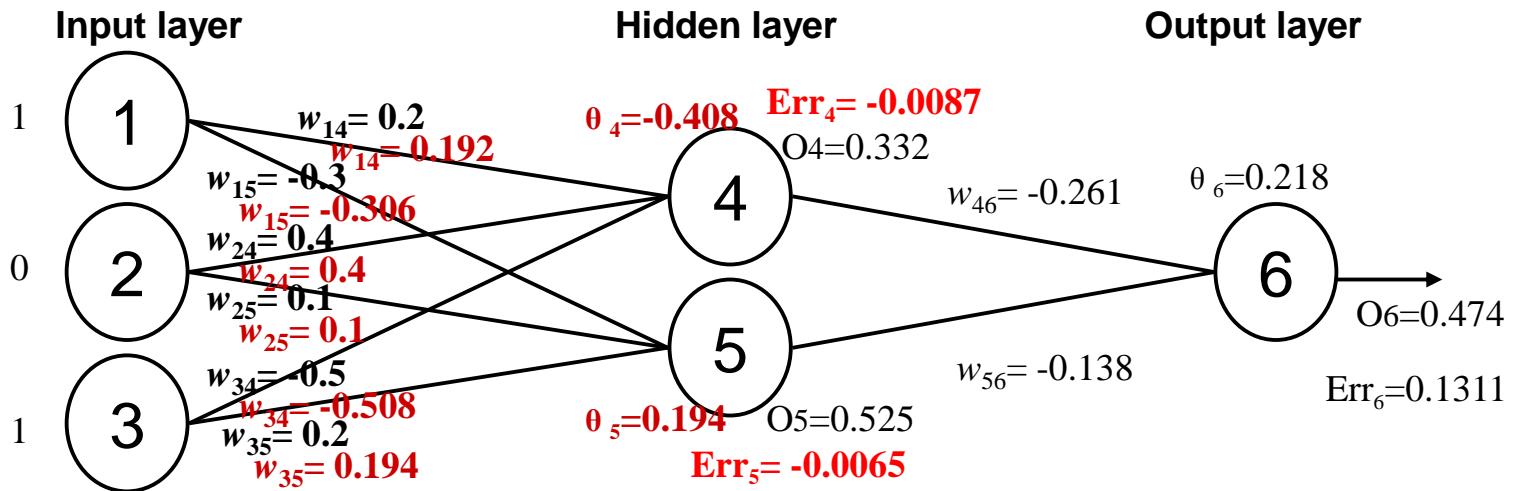
## Hidden layer

$$Err_j = O_j(1-O_j) \sum_k Err_k w_{jk}$$

$$w_{ij} = w_{ij} + (l)Err_j O_i \quad \theta_j = \theta_j + (l)Err_j$$

1. 計算  $Err_6$
2. 計算  $\theta_6$
3. 計算  $w_{4,6}, w_{5,6}$

# Backpropagation



## Output layer

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

## Hidden layer

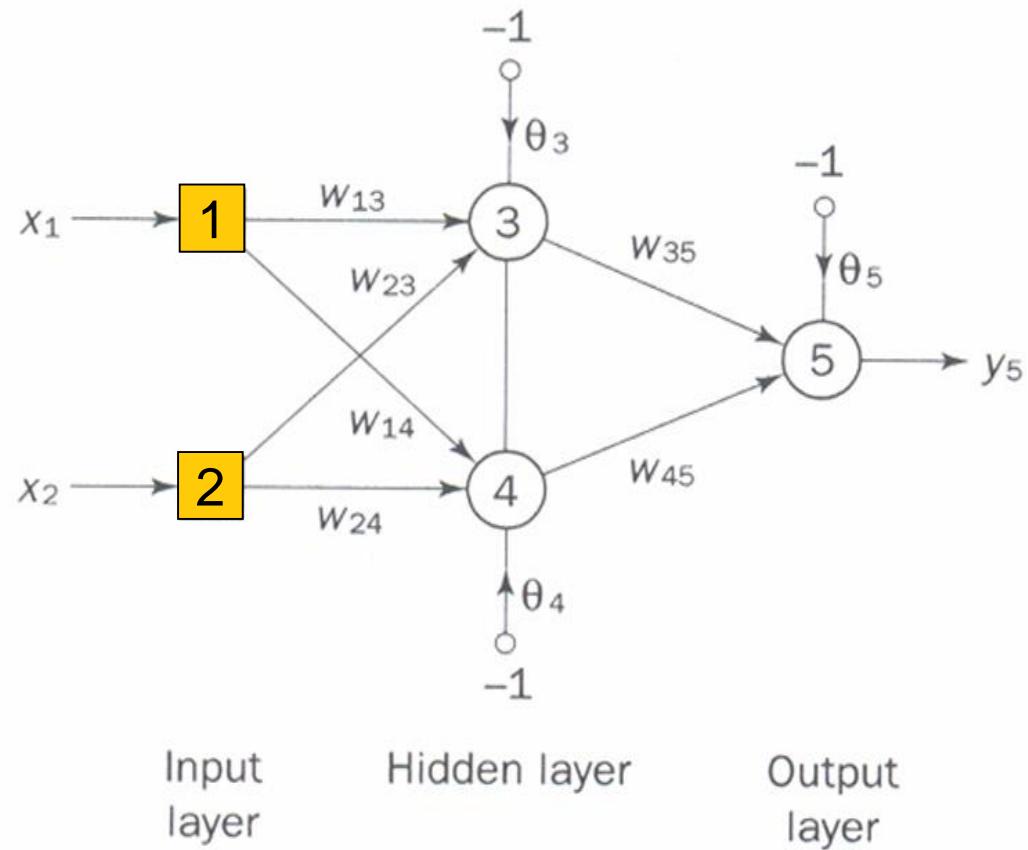
$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$

$$w_{ij} = w_{ij} + (l)Err_j O_i \quad \theta_j = \theta_j + (l)Err_j$$

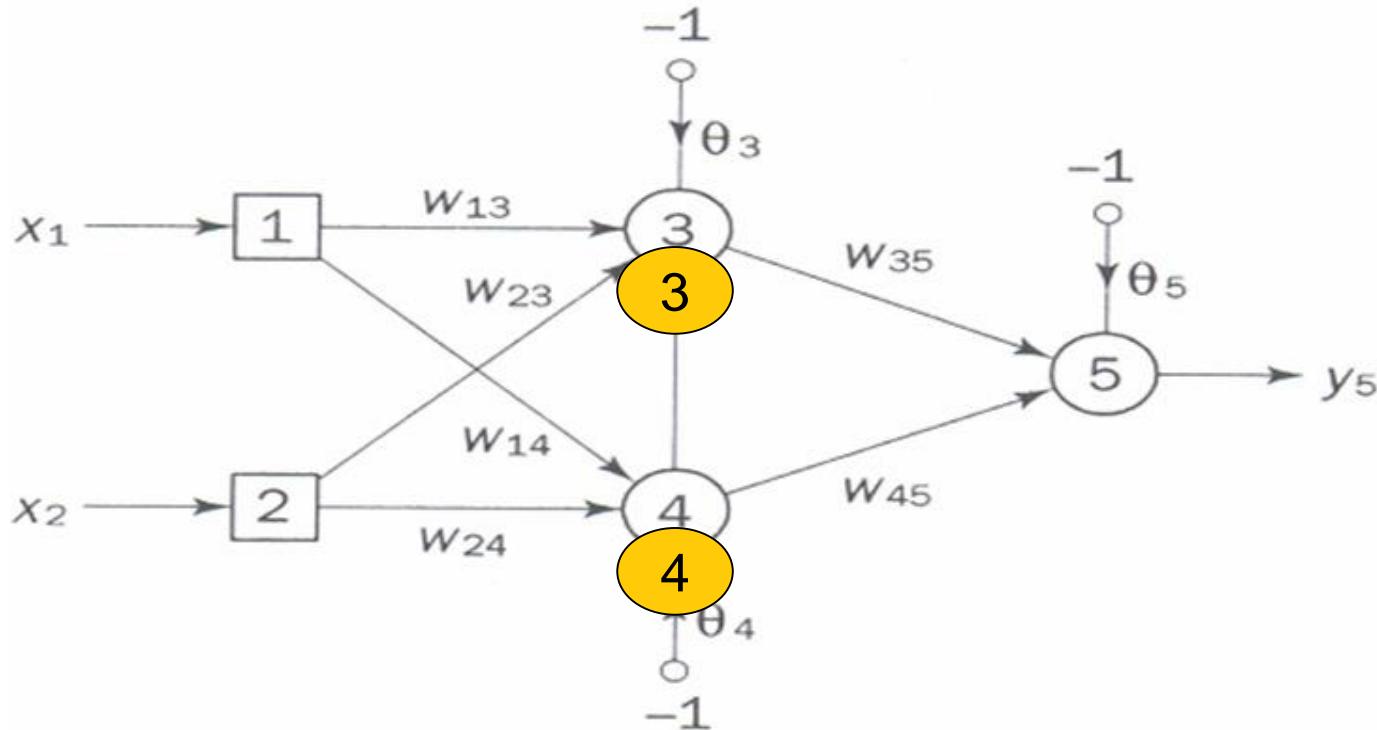
1. 計算  $Err_4, Err_5$
2. 計算  $\theta_4, \theta_5$
3. 計算  $w_{1,4}, w_{2,4}, w_{3,4}, w_{1,5}, w_{2,5}, w_{3,5}$

# Backpropagation練習題

- $X_1=1$
- $X_2=1$
- $w_{13}=0.5$
- $w_{14}=0.9$
- $w_{23}=0.4$
- $w_{24}=1.0$
- $w_{35}=-1.2$
- $w_{45}=1.1$ ,
- $\theta_3=0.8$
- $\theta_4=-0.1$
- $\theta_5=0.3$ ,
- $\alpha=0.1$ (學習率)



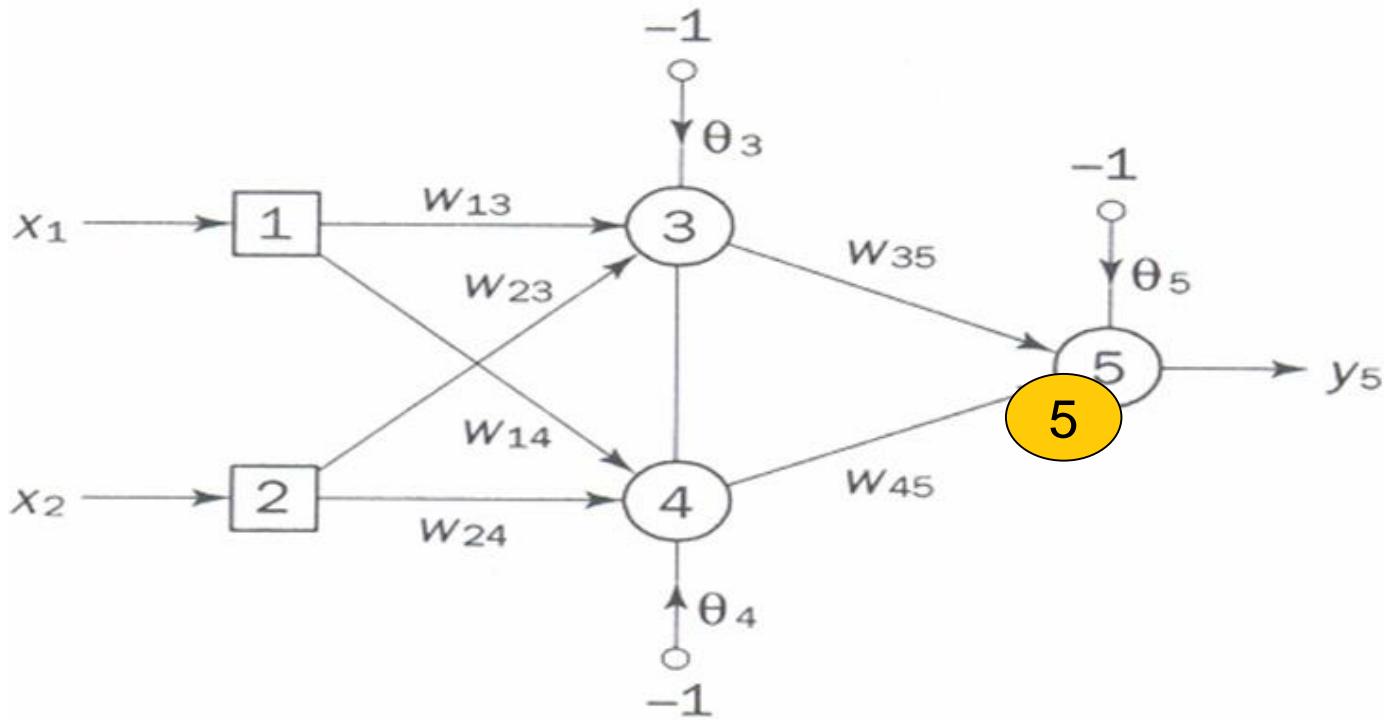
# 計算隱藏層



$$y_3 = \text{sigmoid}(x_1 w_{13} + x_2 w_{23} - \theta_3) = 1/[1 + e^{-(1 \times 0.5 + 1 \times 0.4 - 1 \times 0.8)}] = 0.5250$$

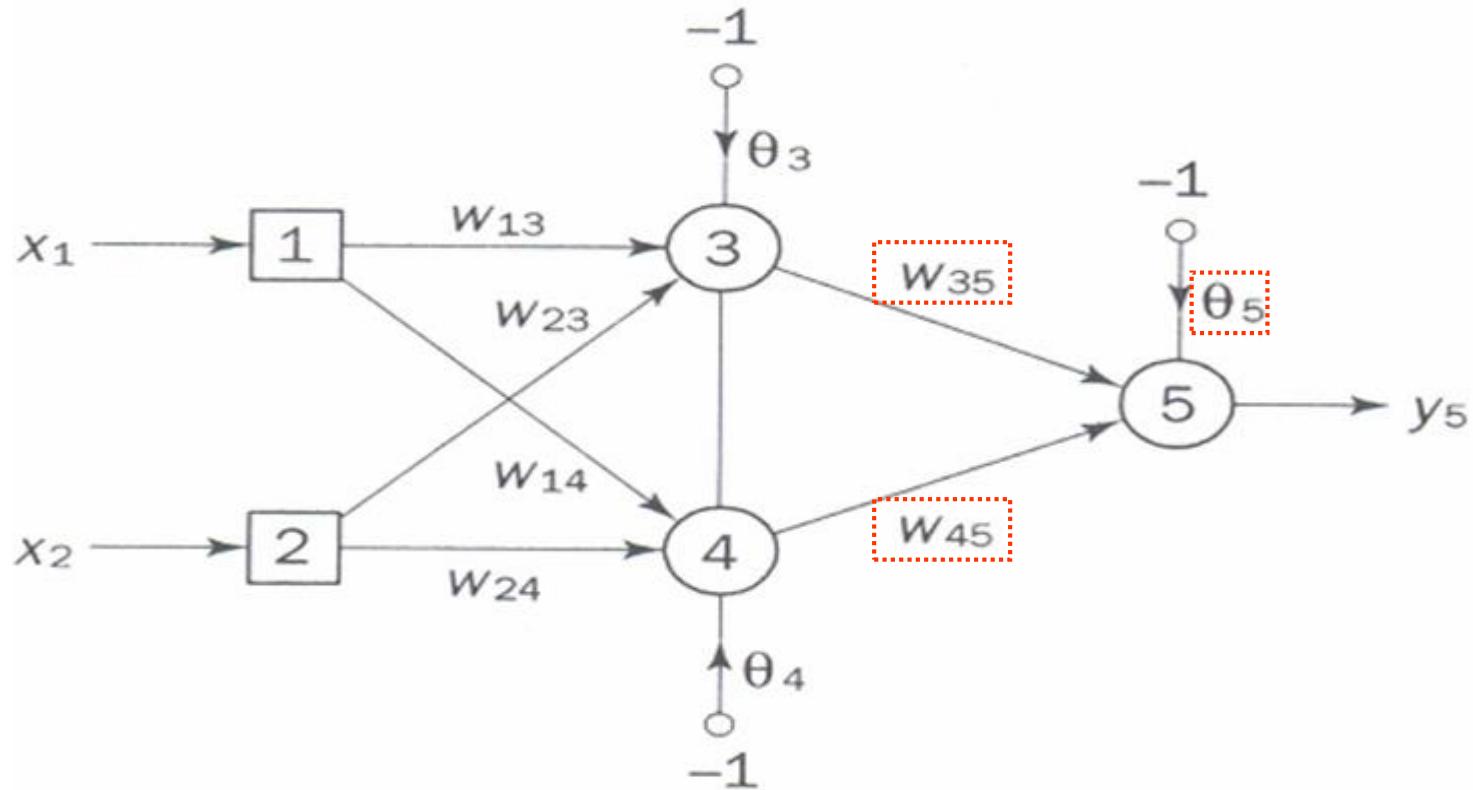
$$y_4 = \text{sigmoid}(x_1 w_{14} + x_2 w_{24} - \theta_4) = 1/[1 + e^{-(1 \times 0.9 + 1 \times 1.0 + 1 \times 0.1)}] = 0.8808$$

# 計算輸出層



$$\begin{aligned}
 y_5 &= \text{sigmoid}(y_3 w_{35} + x_4 w_{45} - \theta_5) \\
 &= 1/[1+e^{-(0.5250 \times 1.2 + 0.8808 \times 1.1 - 1 \times 0.3)}] = 0.5097
 \end{aligned}$$

# 計算輸出層輸出值誤差和誤差梯度 及調整隱藏層 - 輸出層權重值



# 計算輸出層輸出值誤差和誤差梯度 及調整隱藏層 - 輸出層權重值(公式)

$$e = y_{d,5} - y_5 = 0 - 0.5097 = -0.5097$$

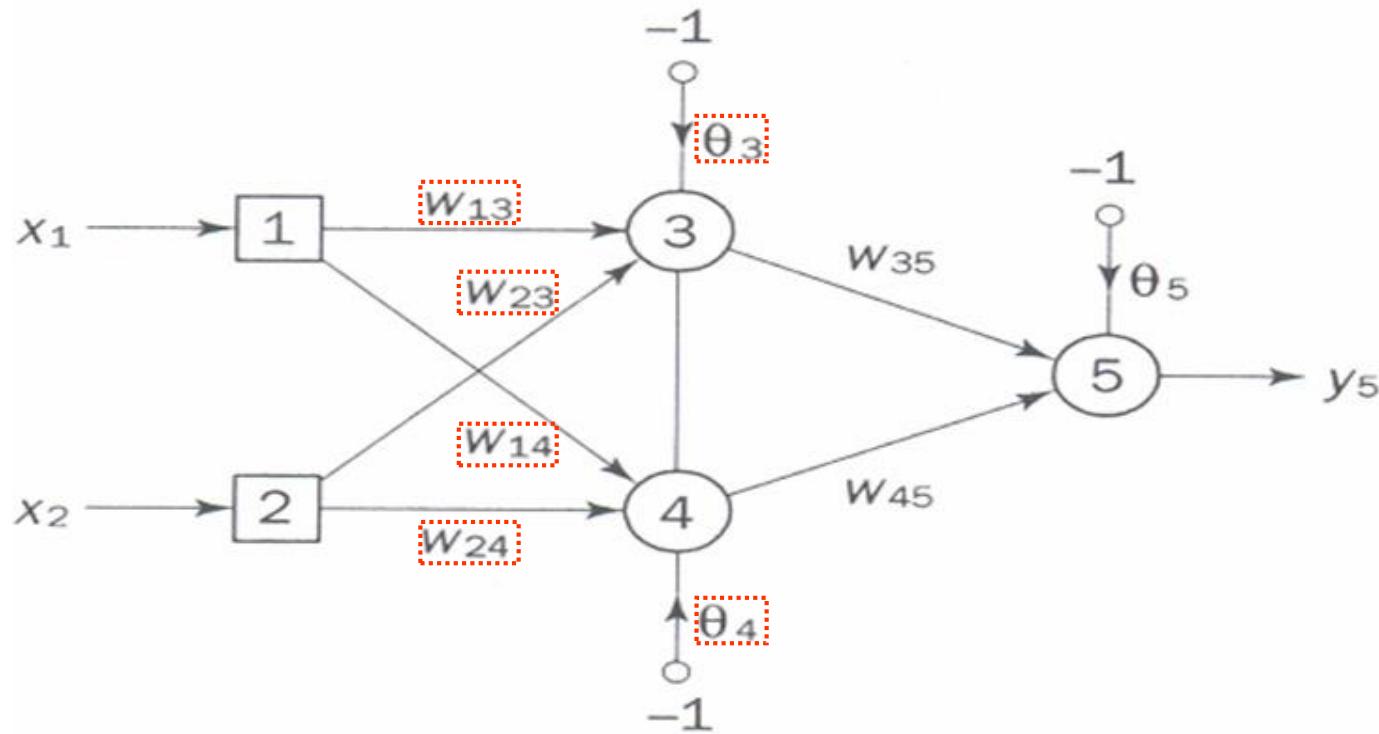
$$\begin{aligned}\delta_5 &= y_5(1-y_5)e \\ &= 0.5097 \times (1-0.5097) \times (-0.5097) = -0.1274\end{aligned}$$

$$\Delta W_{35} = \alpha \times y_3 = \delta_5 = 0.1 \times 0.5250 \times (-0.1274) = -0.0067$$

$$\Delta W_{45} = \alpha \times y_4 = \delta_5 = 0.1 \times 0.8808 \times (-0.1274) = -0.0112$$

$$\Delta \theta_5 = \alpha \times (-1) = \delta_5 = 0.1 \times (-1) \times (-0.1274) = 0.0127$$

# 計算隱藏層輸出值誤差和誤差梯度 及調整隱藏層 - 輸出層權重值



# 計算隱藏層輸出值誤差和誤差梯度 及調整隱藏層 - 輸出層權重值(公式)

$$\delta_3 = y_3(1-y_3) \times \delta_5 \times w_{35} = 0.5250 \times (1-0.5250) \times (-0.1274) \times (-1.2) = 0.0381$$

$$\delta_4 = y_4(1-y_4) \times \delta_5 \times w_{45} = 0.8808 \times (1-0.8808) \times (-0.1274) \times 1.1 = -0.0147$$

$$\Delta W_{13} = \alpha \times x_1 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta W_{23} = \alpha \times x_2 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta \theta_3 = \alpha \times (-1) \times \delta_3 = 0.1 \times (-1) \times 0.0381 = -0.0038$$

$$\Delta W_{14} = \alpha \times x_1 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$

$$\Delta W_{24} = \alpha \times x_2 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$

$$\Delta \theta_4 = \alpha \times (-1) \times \delta_4 = 0.1 \times (-1) \times (-0.0147) = 0.0015$$

更新所有權重及門檻值並進行  
下一個樣本(公式)

$$W_{13} = w_{13} + \Delta w_{13} = 0.5 + 0.0038 = 0.5038$$

$$W_{14} = w_{14} + \Delta w_{14} = 0.9 - 0.0015 = 0.8985$$

$$W_{23} = w_{23} + \Delta w_{23} = 0.4 + 0.0038 = 0.4038$$

$$W_{24} = w_{24} + \Delta w_{24} = 1.0 - 0.0015 = 0.9985$$

$$W_{35} = w_{35} + \Delta w_{35} = -1.2 - 0.0067 = -1.2067$$

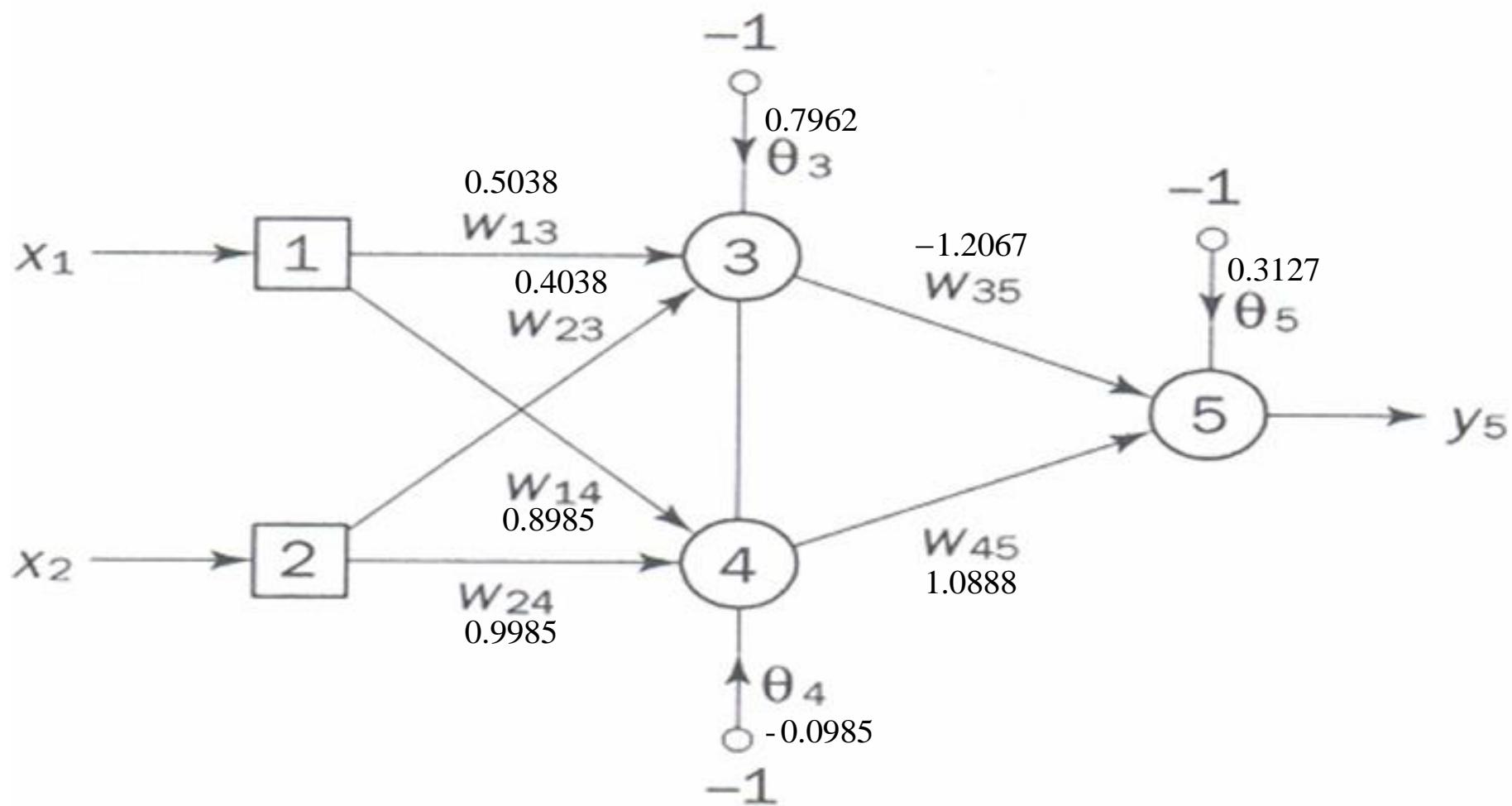
$$W_{45} = w_{45} + \Delta w_{45} = 1.1 - 0.0112 = 1.0888$$

$$\theta_3 = \theta_3 + \Delta \theta_3 = 0.8 - 0.0038 = 0.7962$$

$$\theta_4 = \theta_4 + \Delta \theta_4 = -0.1 + 0.0015 = -0.0985$$

$$\theta_5 = \theta_5 + \Delta \theta_5 = 0.3 - 0.0127 = 0.3127$$

# 完成權重調整 進行下一個樣本參數



# 停止條件

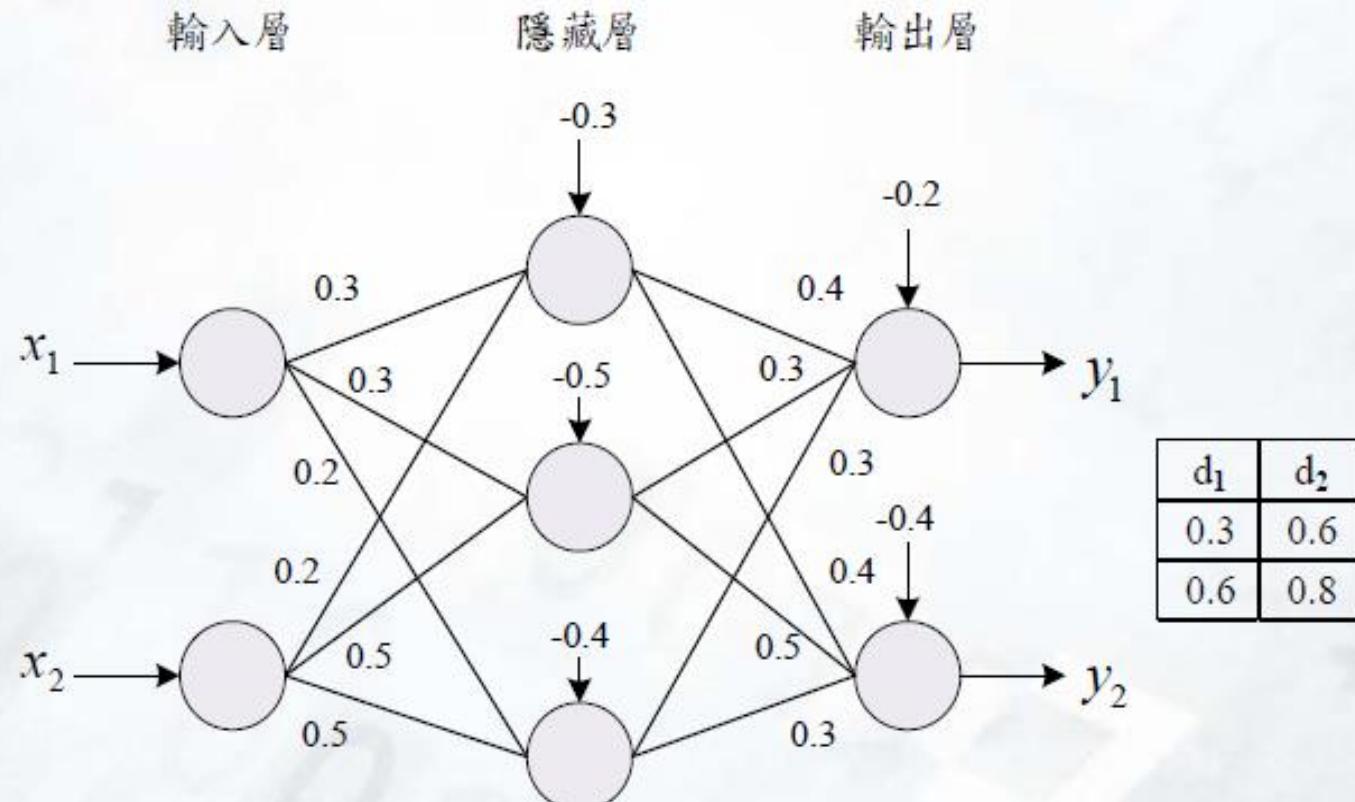
- 每一組樣本數皆計算錯誤(Error)值，當所有樣本測試完，則計算平方誤差總和SSE(Sum of Squared Errors)，直到SSE < 0.001為止

Inputs		Desired output	Actual output	Error	Sum of squared errors
$x_1$	$x_2$	$y_d$	$y_5$	$e$	
1	1	0	0.0155	-0.0155	0.0010
0	1	1	0.9849	0.0151	
1	0	1	0.9849	0.0151	
0	0	0	0.0175	-0.0175	

# 倒傳遞(Backpropagation)類神經網路範例

## 三

根據三層的網路模式架構，欲將兩組訓練資料  $(x_1, x_2)$  與其目標值  $(d_1, d_2)$ ，依照倒傳遞演算法進行模式訓練，其演算過程為



# 建議解答

輸入第一筆資料： $X_1 = 1$ ， $X_2 = 2$ ， $d_1 = 0.3$ ， $d_2 = 0.6$

$$I_{z1} = 0.3 + 0.4 - 0.3 = 0.4, O_{z1} = \frac{1}{1 + e^{-0.4}} = 0.6$$

$$I_{z2} = 0.3 + 1.0 - 0.5 = 0.8, O_{z2} = \frac{1}{1 + e^{-0.8}} = 0.69$$

$$I_{z3} = 0.2 + 1.0 - 0.4 = 0.8, O_{z3} = \frac{1}{1 + e^{-0.8}} = 0.69$$

$$I_{y1} = 0.6 \times 0.4 + 0.69 \times 0.3 + 0.69 \times 0.3 - 0.2 = 0.45, O_{y1} = \frac{1}{1 + e^{-0.45}} = 0.61$$

$$I_{y2} = 0.6 \times 0.4 + 0.69 \times 0.5 + 0.69 \times 0.3 - 0.4 = 0.39, O_{y2} = \frac{1}{1 + e^{-0.39}} = 0.60$$

$$Err_{y1} = 0.61(1 - 0.61)(0.3 - 0.61) = -0.0738$$

$$Err_{y2} = 0.60(1 - 0.60)(0.6 - 0.60) = 0$$

$$Err_{z1} = 0.6(1 - 0.6)(-0.0738 \times 0.4 + 0 \times 0.4) = -0.0071$$

$$Err_{z2} = 0.69(1 - 0.69)(-0.0738 \times 0.3 + 0 \times 0.5) = -0.0047$$

$$Err_{z3} = 0.69(1 - 0.69)(-0.0738 \times 0.3 + 0 \times 0.3) = -0.0047$$

$$E = \frac{1}{2}[(0.3 + 0.0738)^2 + (0.6 - 0)^2] = 0.25$$

# 建議解答II

$$W_{z1y2} = 0.4 + 0.2(0)(0.6) = 0.4$$

$$W_{z2y1} = 0.3 + 0.2(-0.0738)(0.69) = 0.29$$

$$W_{z2y2} = 0.5 + 0.2(0)(0.69) = 0.5$$

$$W_{z3y1} = 0.3 + 0.2(-0.0738)(0.69) = 0.29$$

$$W_{z3y2} = 0.3 + 0.2(0)(0.69) = 0.3$$

$$W_{x1z1} = 0.3 + 0.2(-0.0071)(1) = 0.3$$

$$W_{x1z2} = 0.3 + 0.2(-0.0047)(1) = 0.3$$

$$W_{x1z3} = 0.2 + 0.2(-0.0047)(1) = 0.2$$

$$W_{x2z1} = 0.2 + 0.2(-0.0071)(2) = 0.2$$

$$W_{x2z2} = 0.5 + 0.2(-0.0047)(2) = 0.5$$

$$W_{x2z3} = 0.5 + 0.2(-0.0047)(2) = 0.5$$

$$\theta_{z1} = -0.3 + 0.2(-0.0071) = -0.3$$

$$\theta_{z2} = -0.5 + 0.2(-0.0047) = -0.5$$

$$\theta_{z3} = -0.4 + 0.2(-0.0047) = -0.4$$

$$\theta_{y1} = -0.2 + 0.2(-0.0738) = -0.21$$

$$\theta_{y2} = -0.4 + 0.2(0) = -0.4$$

# 建議解答III

輸入第二筆資料： $X_1 = -1$ ， $X_2 = 1$ ， $d_1 = 0.6$ ， $d_2 = 0.8$

$$I_{z1} = 0.3 + 0.4 - 0.3 = 0.4, O_{z1} = \frac{1}{1 + e^{-0.4}} = 0.6$$

$$I_{z2} = 0.3 + 1.0 - 0.5 = 0.8, O_{z2} = \frac{1}{1 + e^{-0.8}} = 0.69$$

$$I_{z3} = 0.2 + 1.0 - 0.4 = 0.8, O_{z2} = \frac{1}{1 + e^{-0.8}} = 0.69$$

$$I_{y1} = 0.6 \times 0.39 + 0.69 \times 0.29 + 0.69 \times 0.29 - 0.21 = 0.42, O_{y1} = \frac{1}{1 + e^{-0.42}} = 0.60$$

$$I_{y1} = 0.6 \times 0.4 + 0.69 \times 0.5 + 0.69 \times 0.3 - 0.4 = 0.39, O_{y2} = \frac{1}{1 + e^{-0.39}} = 0.60$$

$$Err_{y1} = 0.6(1 - 0.6)(0.6 - 0.6) = 0$$

$$Err_{y2} = 0.60(1 - 0.60)(0.8 - 0.60) = 0.048$$

$$Err_{z1} = 0.6(1 - 0.6)(0.048 \times 0.4 + 0 \times 0.39) = 0.0046$$

$$Err_{z2} = 0.69(1 - 0.69)(0.048 \times 0.5 + 0 \times 0.29) = 0.0051$$

$$Err_{z3} = 0.69(1 - 0.69)(0.048 \times 0.3 + 0 \times 0.3) = 0.0031$$

$$E = \frac{1}{2}[(0.6 - 0)^2 + (0.8 - 0.6)^2] = 0.2$$

$$W_{z1y1} = 0.39 + 0.2(0)(0.6) = 0.39$$

$$W_{z1y2} = 0.4 + 0.2(0.048)(0.6) = 0.41$$

$$W_{z2y1} = 0.29 + 0.2(0)(0.69) = 0.29$$

$$W_{z2y2} = 0.5 + 0.2(0.048)(0.69) = 0.51$$

$$W_{z3y1} = 0.29 + 0.2(0)(0.69) = 0.29$$

$$W_{z3y2} = 0.3 + 0.2(0.048)(0.69) = 0.31$$

$$W_{x1z1} = 0.3 + 0.2(0.0046)(-1) = 0.3$$

$$W_{x1z2} = 0.3 + 0.2(0.0051)(-1) = 0.3$$

$$W_{x1z3} = 0.2 + 0.2(0.0031)(-1) = 0.2$$

$$W_{x2z1} = 0.2 + 0.2(0.0046)(1) = 0.2$$

$$W_{x2z2} = 0.5 + 0.2(0.0051)(1) = 0.5$$

$$W_{x2z3} = 0.5 + 0.2(0.0031)(1) = 0.5$$

$$\theta_{z1} = -0.3 + 0.2(0.0046) = -0.3$$

$$\theta_{z2} = -0.5 + 0.2(0.0051) = -0.5$$

$$\theta_{z3} = -0.4 + 0.2(0.0031) = -0.4$$

$$\theta_{y1} = -0.21 + 0.2(0) = -0.21$$

$$\theta_{y2} = -0.4 + 0.2(0.048) = -0.39$$

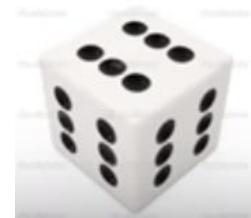
# Term-Document Matrix(TDM)

- D1 : Two for tea and tea for two.
- D2 : Tea for me and tea for you.
- D3 : You for me and me for you.

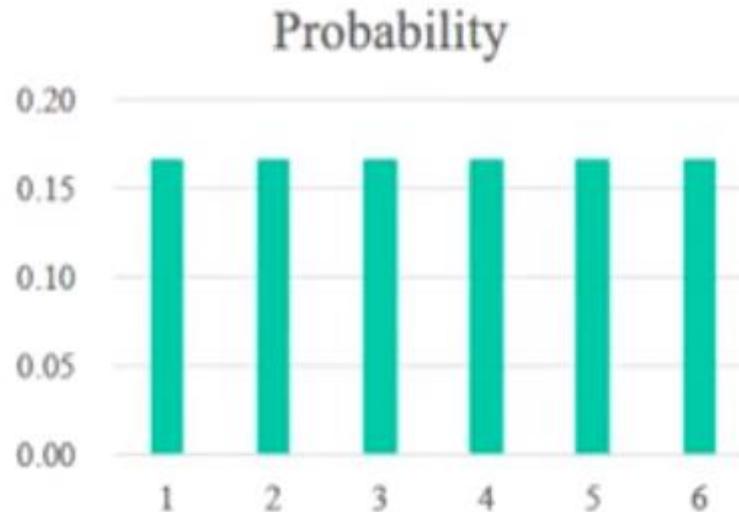
	D1	D2	D3
two	2	0	0
tea	2	2	0
me	0	1	2
you	0	1	2
for	2	2	2
and	1	1	1

# What is “Uncertainty”?

	D1	D2	D3
two	2	0	0
tea	2	2	0
me	0	1	2
you	0	1	2
for	2	2	2
and	1	1	1

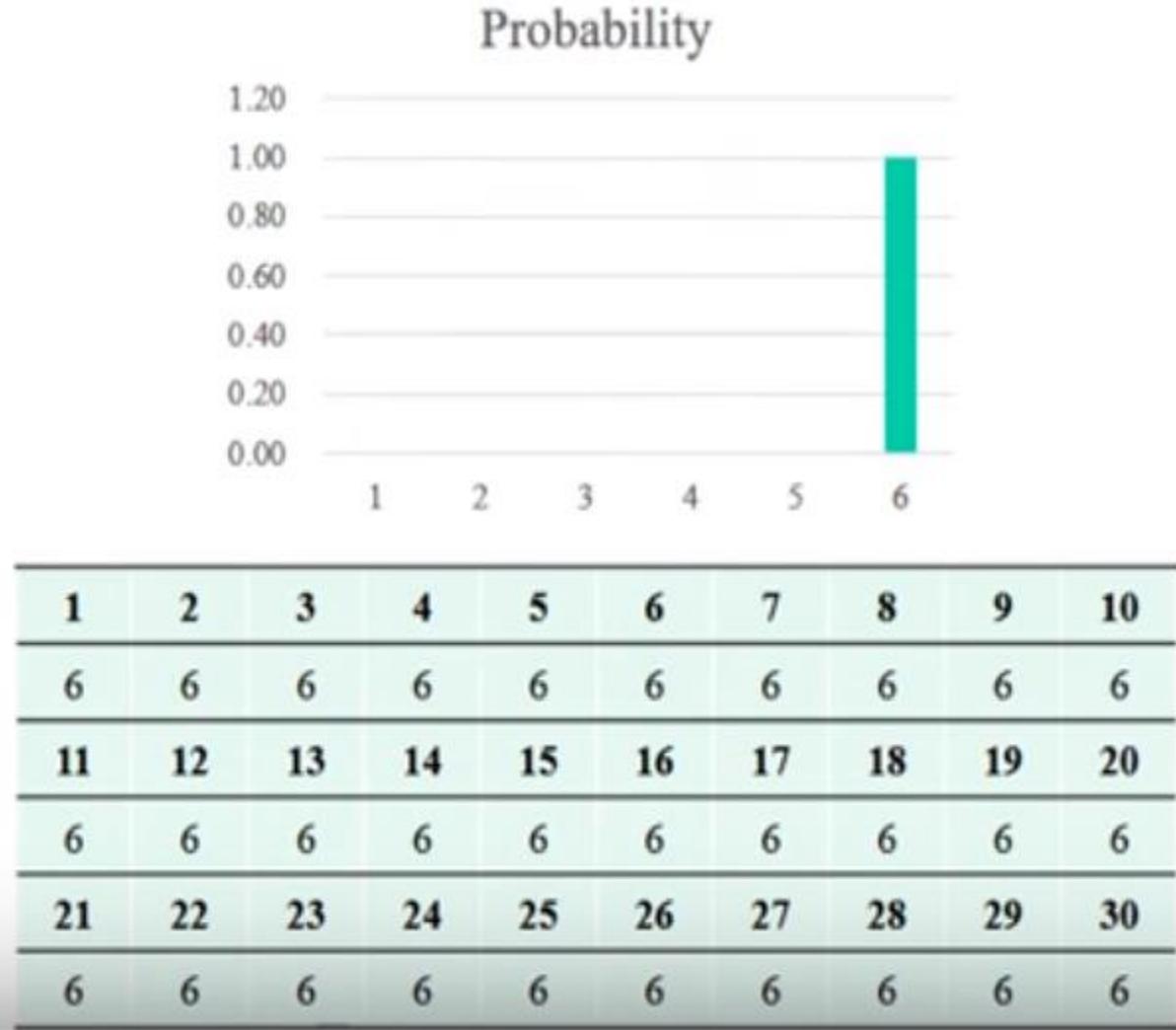


# What is “Uncertainty”?

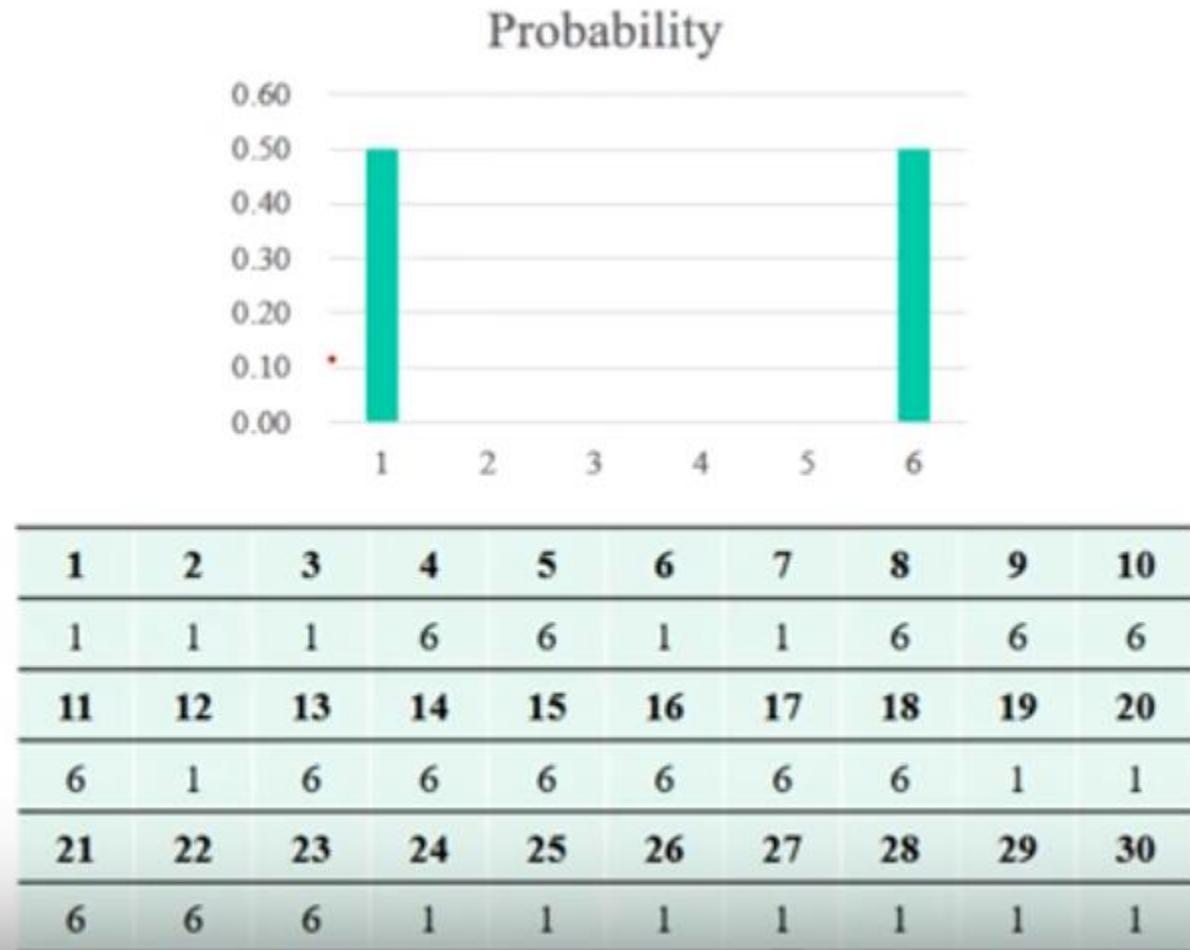


1	2	3	4	5	6	7	8	9	10
3	4	4	2	4	4	6	6	3	2
11	12	13	14	15	16	17	18	19	20
5	6	1	1	2	5	3	4	2	2
21	22	23	24	25	26	27	28	29	30
1	2	2	4	3	3	5*	3	5	3

# What is “Uncertainty”?



# What is “Uncertainty”?

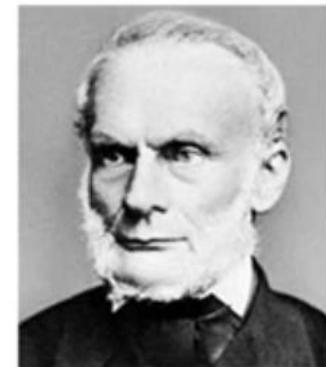


# Entropy(熵/亂度)

- “熵(entropy)”是統計力學名詞
  - 系統無序性的一種度量。
  - 即用來衡量亂的程度
  - 凌”亂”的程”度”，也可稱為亂度。

熵越大 → 越亂  
熵越小 → 越不亂

$$H = - \sum_{j=1}^m p_j * \ln p_j , \quad \sum_{j=1}^m p_j = 1$$

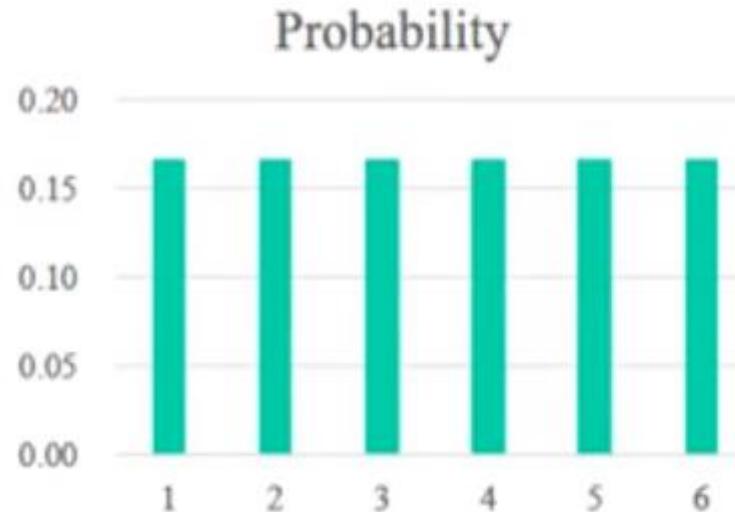


魯道夫·克勞修斯——最早提出「熵」這個概念的物理學家

<https://zh.wikipedia.org/wiki/%E9%AD%AF%E9%81%93%E5%A4%AB%C2%B7E5%85%8B%E5%8B%9E%E4%BF%AE%E6%96%AF>

# Entropy(熵/亂度)

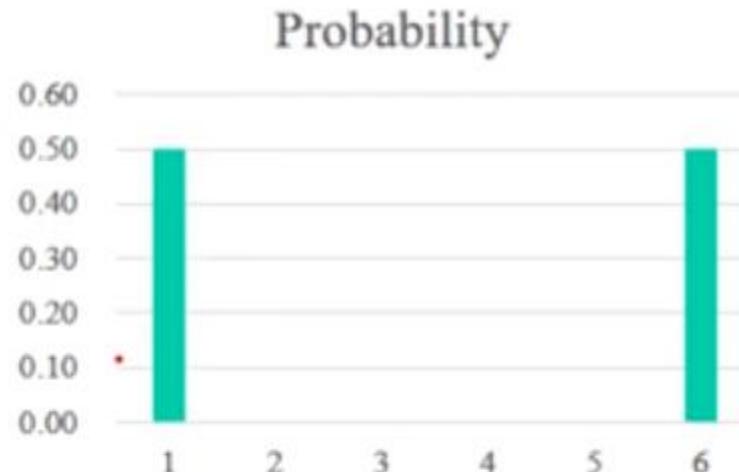
$$H = - \sum_{j=1}^m p_j * \ln p_j = 1.7918$$



1	2	3	4	5	6	7	8	9	10
3	4	4	2	4	4	6	6	3	2
11	12	13	14	15	16	17	18	19	20
5	6	1	1	2	5	3	4	2	2
21	22	23	24	25	26	27	28	29	30
1	2	2	4	3	3	5*	3	5	3

# Entropy(熵/亂度)

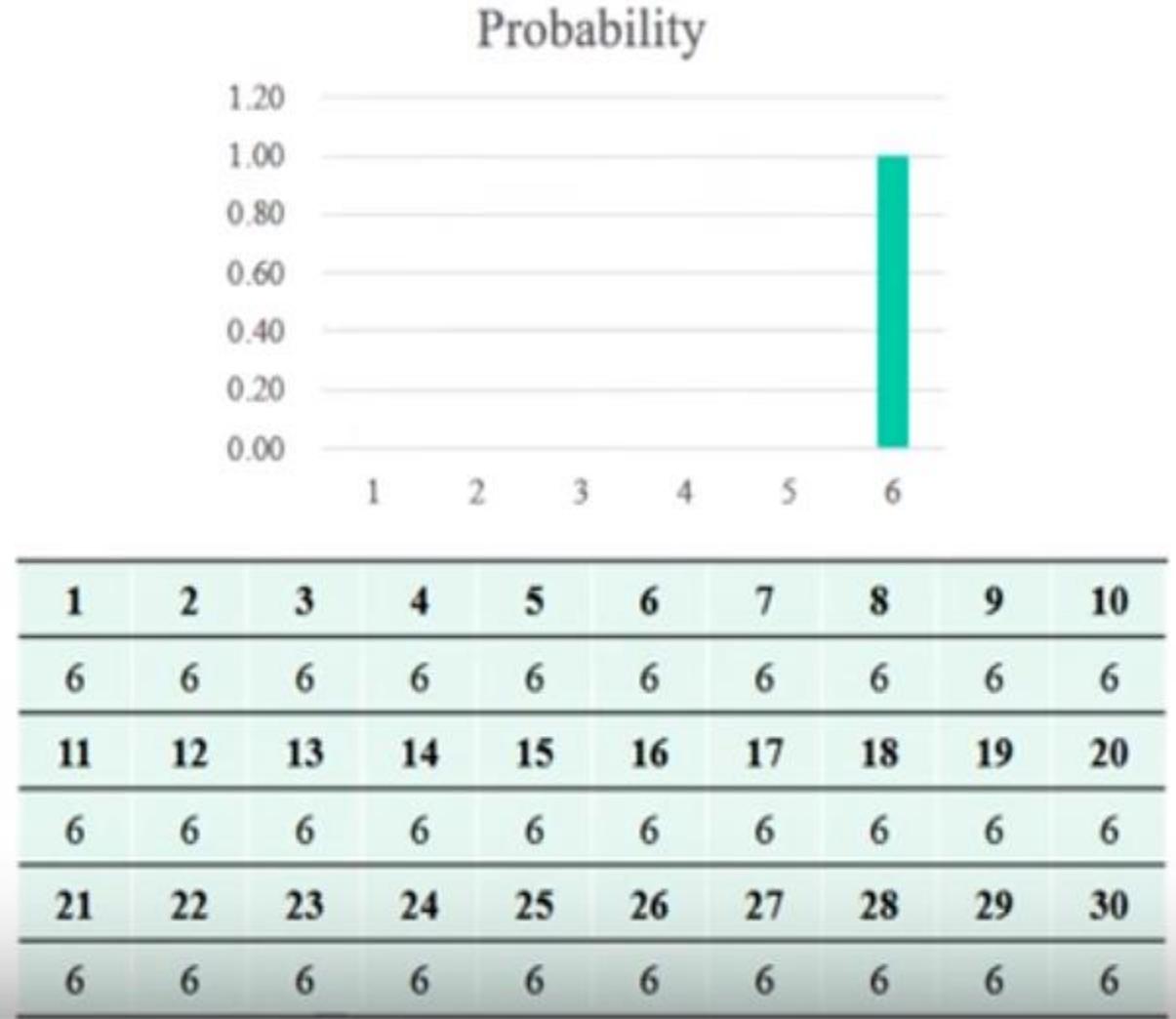
$$H = - \sum_{j=1}^m p_j * \ln p_j = 0.6931$$



1	2	3	4	5	6	7	8	9	10
1	1	1	6	6	1	1	6	6	6
<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
6	1	6	6	6	6	6	6	1	1
<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	<b>26</b>	<b>27</b>	<b>28</b>	<b>29</b>	<b>30</b>
6	6	6	1	1	1	1	1	1	1

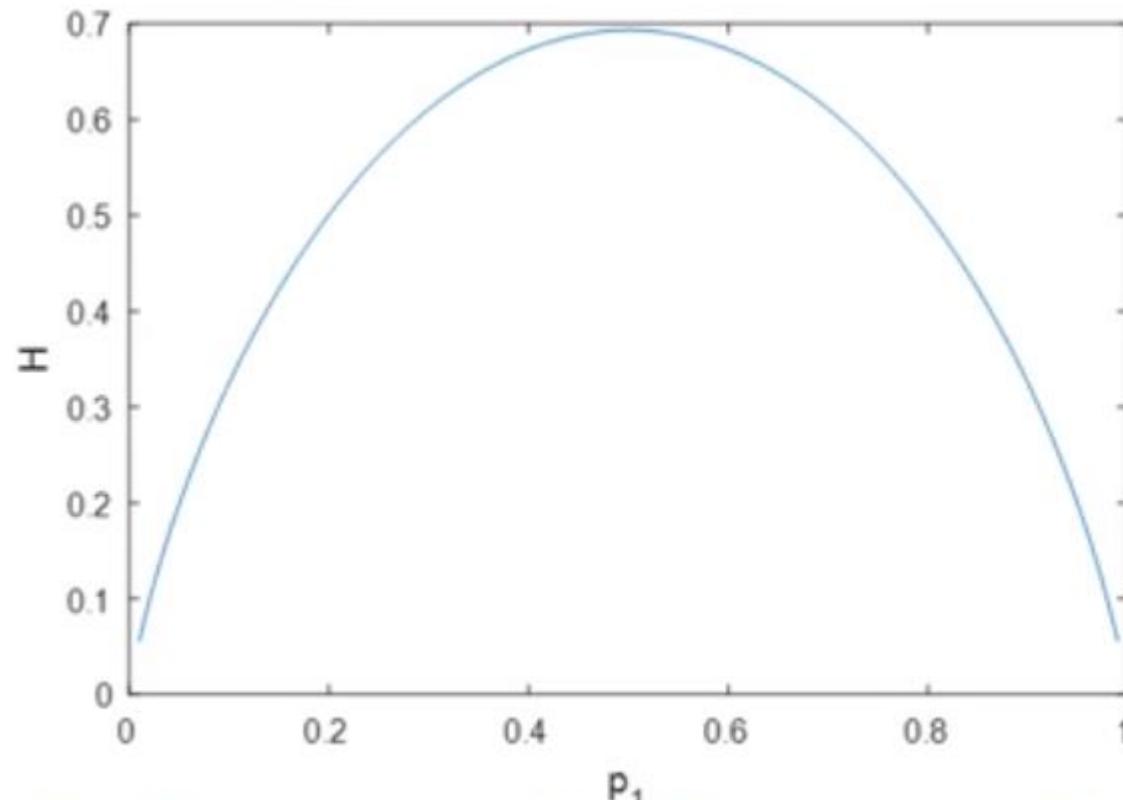
# Entropy(熵/亂度)

$$H = - \sum_{j=1}^m p_j * \ln p_j = 0$$



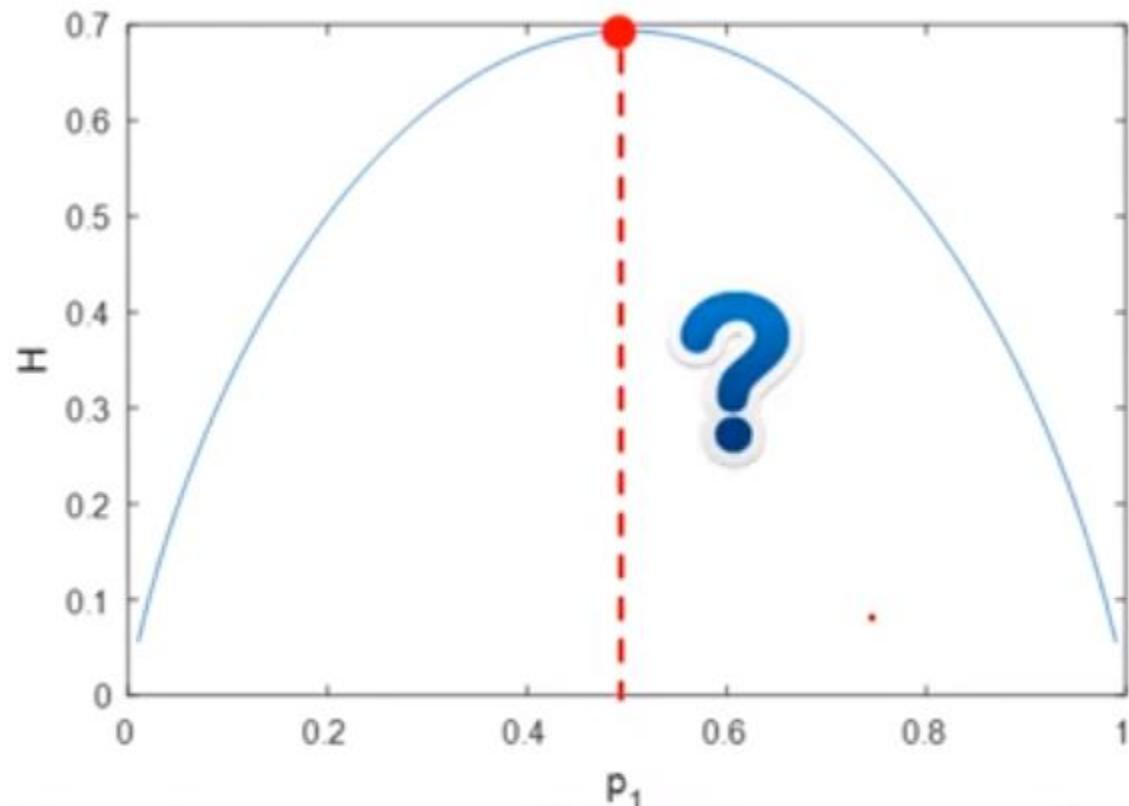
# Entropy(熵/亂度)

$$H = -(p_1 * \ln p_1 + p_2 * \ln p_2) \cdot p_1 + p_2 = 1$$



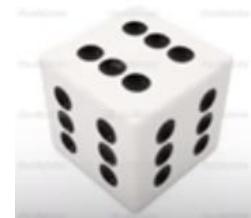
# Entropy(熵/亂度)

$$H = -(p_1 * \ln p_1 + p_2 * \ln p_2) \cdot p_1 + p_2 = 1$$



# What is “Uncertainty”?

	D1	D2	D3
two	2	0	0
tea	2	2	0
me	0	1	2
you	0	1	2
for	2	2	2
and	1	1	1



# 分類技術練習題一

客戶編號	居住區域	年紀	婚姻狀況	性別	品牌忠誠度
1	市區	<=20	已婚	女	低
2	市區	<=20	已婚	男	低
3	市郊	<=20	已婚	女	高
4	鄉鎮	21~30	已婚	女	高
5	鄉鎮	>30	未婚	女	高
6	鄉鎮	>30	未婚	男	低
7	市郊	>30	未婚	男	高
8	市區	21~30	已婚	女	低
9	市區	>30	未婚	女	高
10	鄉鎮	21~30	未婚	女	高
11	市區	21~30	未婚	男	高
12	市郊	21~30	已婚	男	高
13	市郊	<=20	未婚	女	高
14	鄉鎮	21~30	已婚	男	低

# 分類技術練習題二

編號	是否負債	性別	婚姻狀況	收入	是否還款
1	是	男	單身	低	否
2	否	女	單身	低	否
3	是	男	單身	高	是
4	否	女	結婚	低	是
5	否	男	單身	高	是
6	是	女	單身	高	否
7	否	女	結婚	低	是
8	是	男	結婚	高	否
9	否	男	單身	低	是
10	是	女	結婚	低	否
11	否	女	結婚	高	是
12	是	男	結婚	高	否

# 分類

客戶編號	年齡	婚姻	收入	購買筆記型電腦
1	<30	單身	高	否
2	<30	單身	中	否
3	>=30	單身	低	是
4	>=30	已婚	中	否
5	>=30	已婚	低	否
6	>=30	已婚	低	否
7	>=30	已婚	中	否
8	<30	單身	高	否
9	<30	已婚	低	否
10	>=30	已婚	中	否
11	<30	已婚	高	是
12	>=30	已婚	中	否
13	>=30	單身	中	是
14	>=30	已婚	低	否
15	>=30	單身	中	是
16	<30	單身	低	否