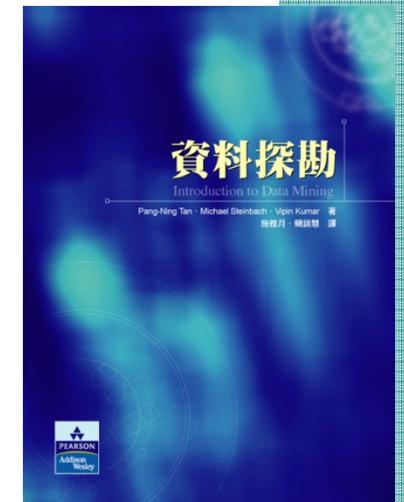


第 4 章

分類法：基本概念、 決策樹及模式的評估



© 2008 台灣培生教育出版 (Pearson Education Taiwan)

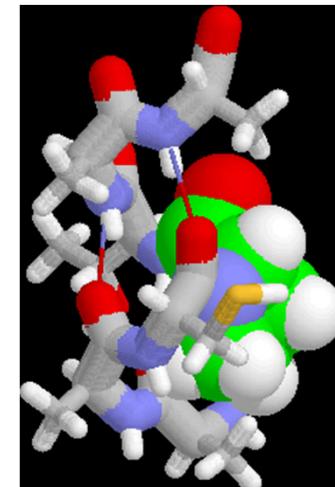
分類法：定義

- 目的是將一個**物件**指定至其中一個已**預設的分類**中
- 分類是指建立一個學習目標函數 f ，使得這個學習函數可以藉由**x屬性**對應至**y**的類別
- 適合預測**二元分類**或是**名目分類**的問題
- 具**順序特性**的類別其**效果較差**
- (例: 要分類一個人的收入情形，如高、中、低)



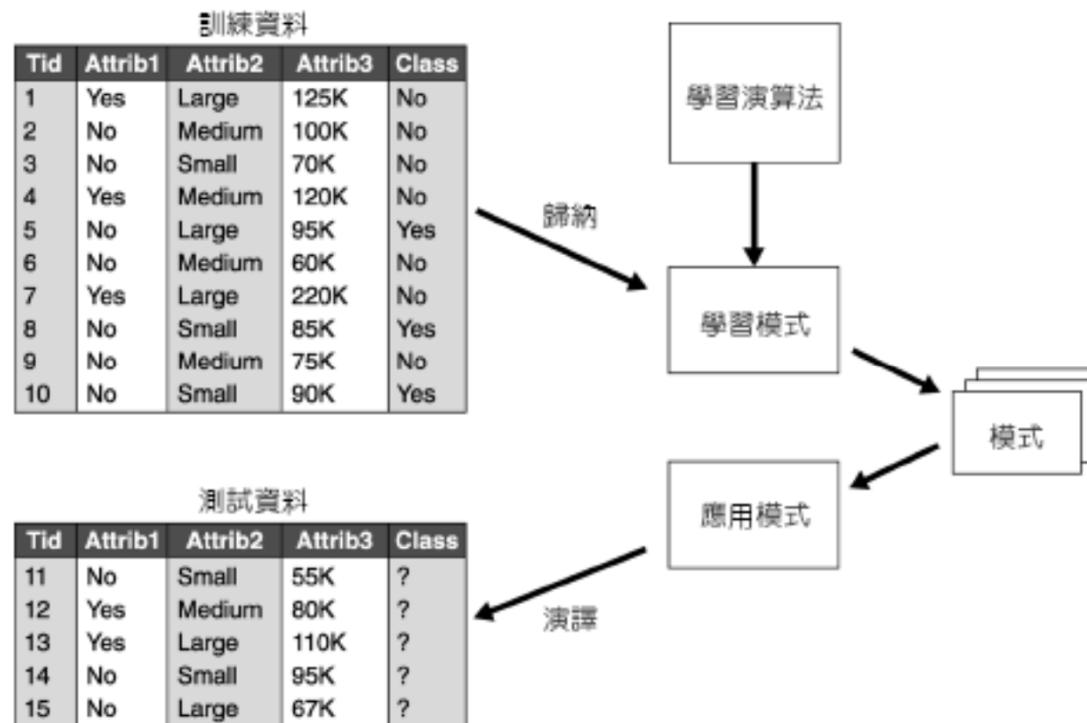
分類技術

- 從輸入資料中建立分類模式的系統化方法
 - 決策樹 (Decision tree)
 - 類神經網路 (artificial neural network)
 - 支援向量機 (support vector machines)
 - 及單純貝氏 (Naïve Bayes)



建立分類模式的做法

- 訓練資料是由一些已知類別標記的資料所組成
- 訓練資料主要是用來**建立分類模式**
- 用此模式來對**未知類別**標記的測試資料進行預測



解決分類問題

表 4.1 脊椎動物

名稱	身體溫度	外表	胎生	水生	飛行能力	有無四肢	冬眠	類別標記
人類	恆溫	毛髮	是	否	否	是	否	哺乳類
蟒蛇	冷血	鱗片	否	否	否	否	是	爬蟲類
鮭魚	冷血	鱗片	否	是	否	否	否	魚類
鯨魚	恆溫	毛髮	是	是	否	否	否	哺乳類
青蛙	冷血	無	否	一半	否	是	是	兩棲類
科摩多龍	冷血	鱗片	否	否	否	是	否	爬蟲類
蝙蝠	恆溫	毛髮	是	否	是	是	是	哺乳類
鴿子	恆溫	羽毛	否	否	是	是	否	鳥類
貓	恆溫	軟毛	是	否	否	是	否	哺乳類
豹鯊	冷血	鱗片	是	是	否	否	否	魚類
海龜	冷血	鱗片	否	一半	否	是	否	爬蟲類
企鵝	恆溫	羽毛	否	一半	否	是	否	鳥類
豪豬	恆溫	刺	是	否	否	是	是	哺乳類
鰻魚	冷血	鱗片	否	是	否	否	否	魚類
蠑螈	冷血	無	否	一半	否	是	是	兩棲類

名稱	身體溫度	外表	胎生	水生	飛行能力	有無四肢	冬眠	類別標記
大毒蜥蜴	冷血	鱗片	否	否	否	是	是	?

分類評估標準

決定於模式能夠正確預測測試資料的比例

混亂矩陣 (Confusion matrix)

		預測類別	
		Class = 1	Class = 0
實際類別	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

$$accuracy = \frac{\text{正確預測的個數}}{\text{總預測個數}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{錯誤率} = \frac{\text{錯誤預測的個數}}{\text{總預測個數}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

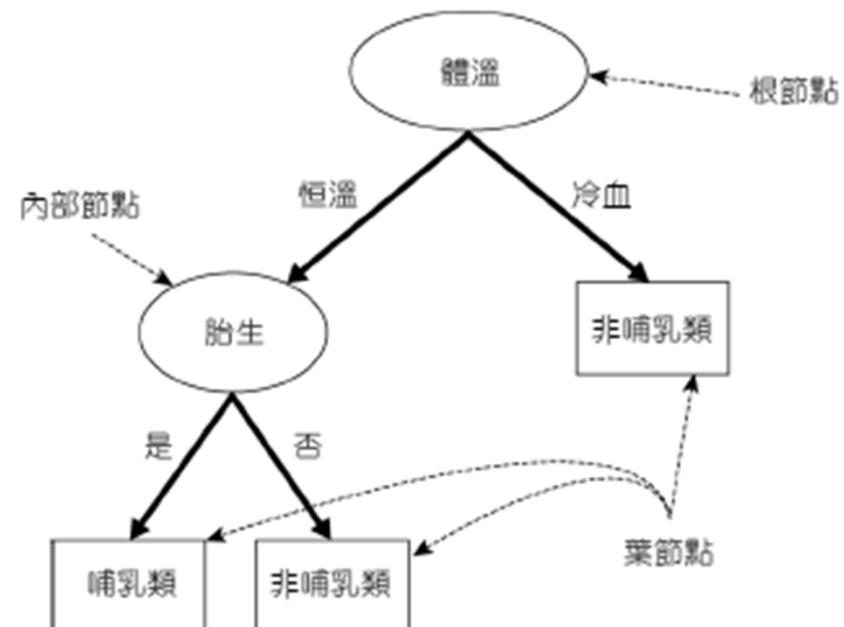
大部份的演算法都期望能夠得到最高的正確率，或是最低的錯誤率

決策樹

- 樹包含三個節點：
 - **根節點**：沒有任何進入的邊，而且有0個或是輸出的邊
 - **內部節點**：每個節點都有一個輸入的邊，以及二個或多個輸出的邊
 - **葉節點**或是**終端節點**：每個節點都有一個輸入的邊，但沒有輸出的邊

每個葉節點都是一個**類別標記**

- 決策樹範例
 - 哺乳類動物分類的問題



如何建立決策樹

- 決策樹可從已知**屬性集合**中建構起來
- 採用**貪婪策略**（greedy strategy）建立決策樹
- 決策樹演算法：其中一種是Hunt's 演算法，它是一些現存方法的基礎，包含ID3、C4.5 及CART

Hunt's 演算法

- 決策樹是用遞迴的方式不斷地將訓練資料分割至後繼的子集合中
- 假設 D_t 是與節點 t 的相關的訓練資料，而 $y = \{y_1, y_2, \dots, y_c\}$ 是類別標記
- Hunt's 演算法的遞迴定義
 - 步驟1：如果在 D_t 中的所有記錄都屬於相同類別 y_t ，那麼 t 就是一個葉節點，標記為 y_t
 - 步驟2：如果 D_t 包含一些屬於一個以上類別記錄時，則會選取一個屬性測試條件作為測試節點，以便將資料分割至較小的子集合中

Hunt's 演算法

	二元屬性	類別屬性	連續值	類別
編號	有房子	婚姻狀況	年收入	未還款
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

圖 4.6 ▶ 預測償還貸款能力的訓練資料

Hunt's 演算法 (以預測貸款者為例)

- Hunt's 演算法所產生的決策樹

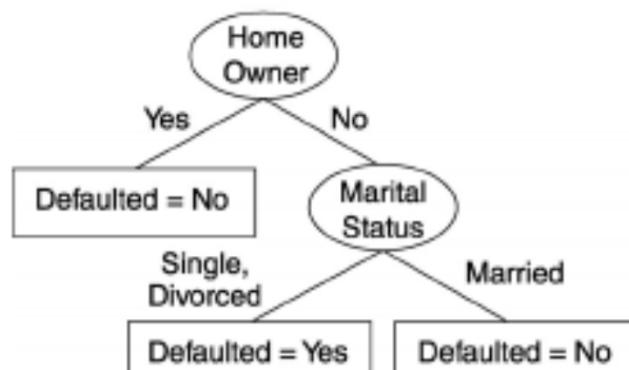
表示貸款者
有如期歸還

Defaulted = No

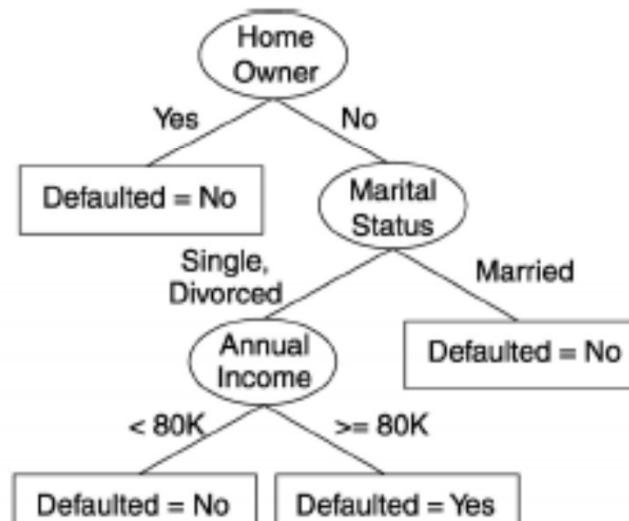
(a)



(b)



(c)



(d)

設計決策樹的問題

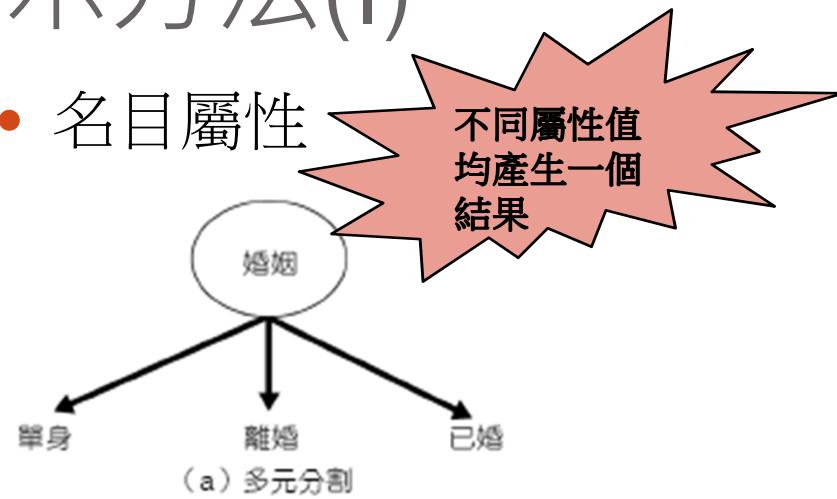
- 決策樹演算法要能處理下列問題
 - 訓練資料如何分割?
 - 演算法要提供一個方法處理不同屬性型態的測試條件
 - 分割何時停止?
 - 終止樹的成長

屬性測試條件的表示方法(I)

- 二元屬性



- 名目屬性

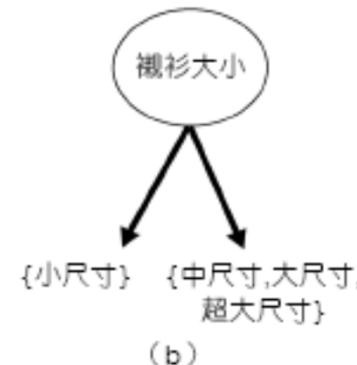
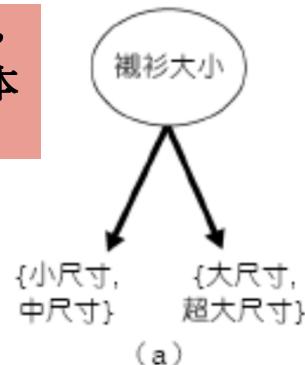


CART演算
法即屬於此
種

屬性測試條件的表示方法(II)

● 順序屬性

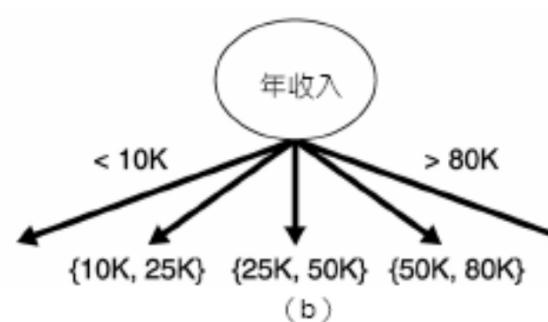
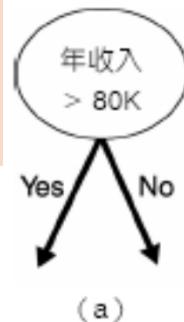
順序屬性值可以被分群，但在分群時必須保持原本屬性值之順序



此法違反了原有的屬性順序

● 連續性屬性

需考量所有可能可以進行分割的點，並找出最好的分割點
注意 仍需保持順序性



如何選擇最好的分割點(I)

- 在分割前後，由類別的分配情形來決定

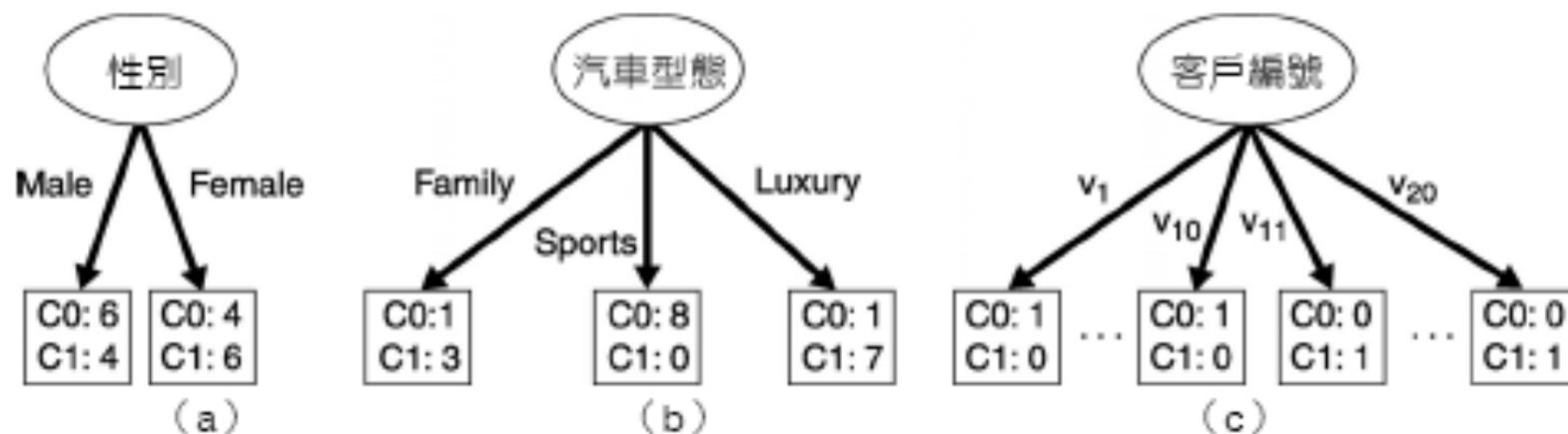


圖 多元分割與二元分割

如何選擇最好的分割點(II)

不純程度愈小，其類別分配將愈歪斜

- 不純程度 (degree of impurity) 的衡量

- 亂度 = $-\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$

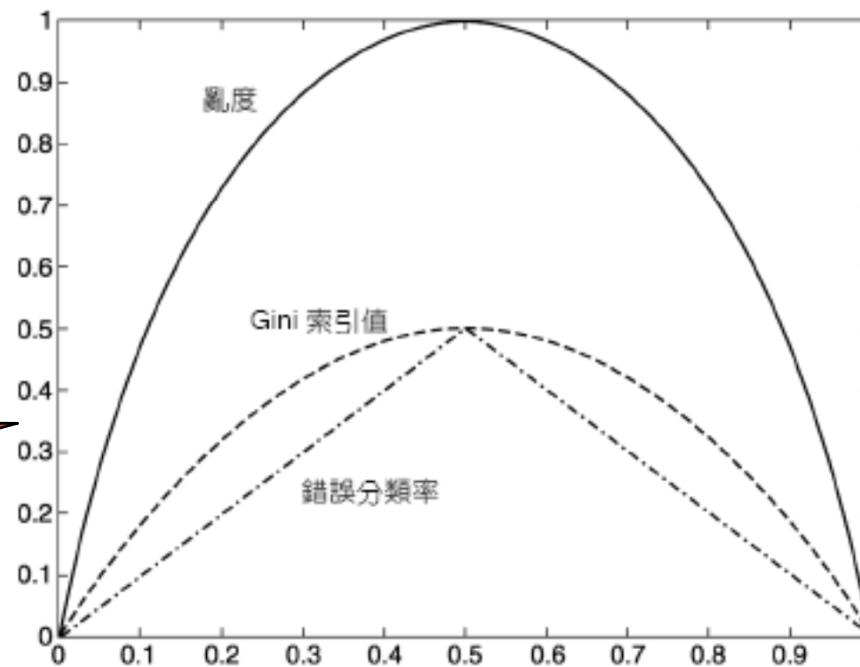
C表類別個數

- Gini 索引值 = $1 - \sum_{i=0}^{c-1} [p(i|t)]^2$

p表類別比例

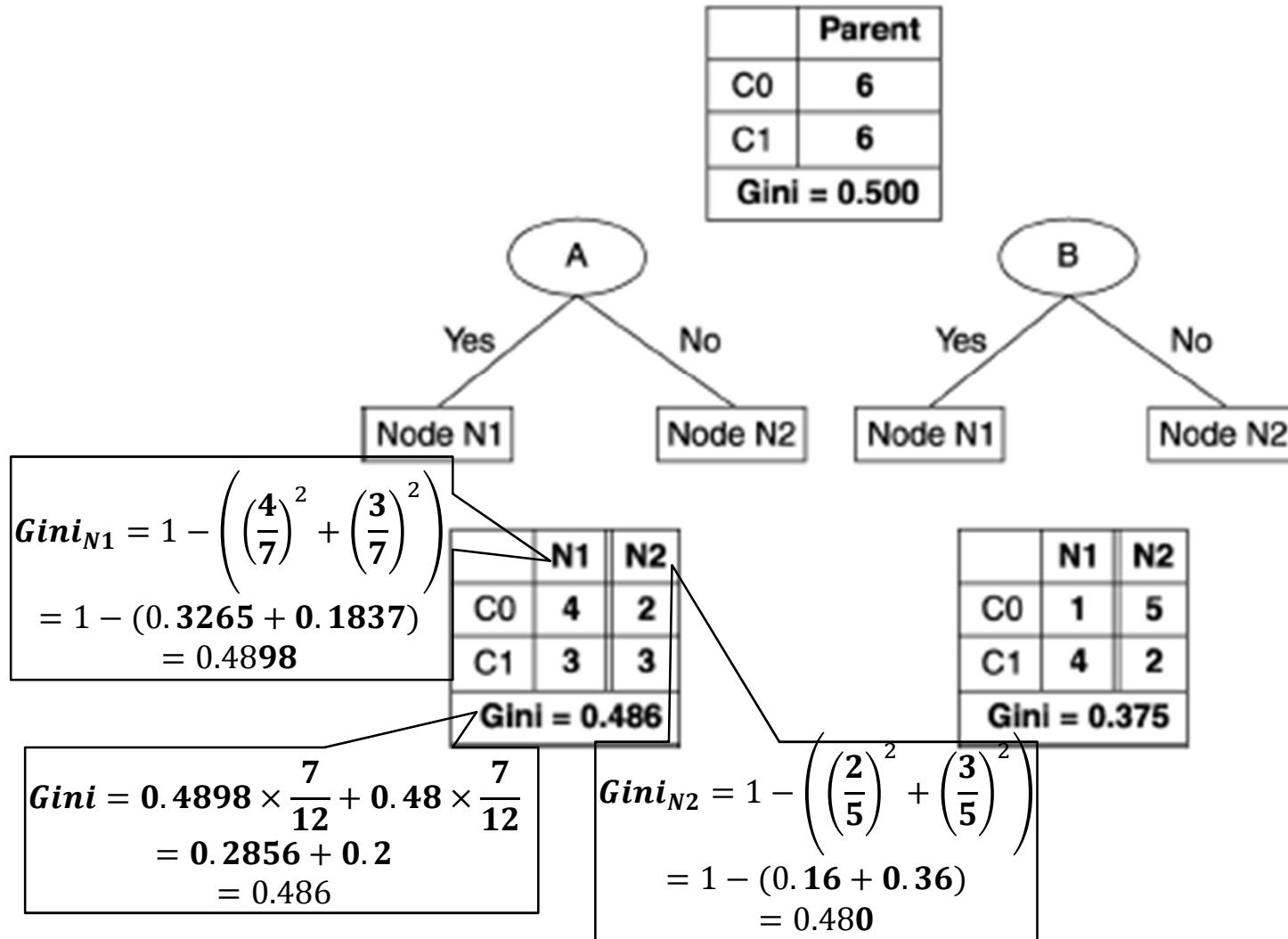
- 錯誤分類率 = $1 - \max_i [p(i|t)]$

最小化子節點不純程度

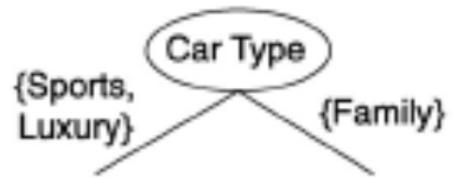


不同衡量不純程度方法比較圖

二元屬性的分割



名目屬性的分割



Car Type		
	{Sports, Luxury}	{Family}
C0	9	1
C1	7	3
Gini	0.468	

Car Type		
	{Sports}	{Family, Luxury}
C0	8	2
C1	0	10
Gini	0.167	

(a) 二元分割

Car Type			
	Family	Sports	Luxury
C0	1	8	1
C1	3	0	7
Gini	0.163		

(b) 多元分割

連續屬性的分割

Class	No	No	No	Yes	Yes	Yes	No	No	No	No	No
年收入											
排序值 →	60	70	75	85	90	95	100	120	125	220	
分割值 →	55	65	72	80	87	92	97	110	122	172	230
	<=	>	<=	>	<=	>	<=	>	<=	>	<=
Yes	0	3	0	3	0	3	1	2	2	1	3
No	0	7	1	6	2	5	3	4	3	4	4
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420

為降低運算複雜度，可將年收入進行排序再以相鄰兩值之中間值進行測試

獲利率

- 利用獲利率的分割條件來決定分割的好壞

$$\text{獲利率} = \frac{\Delta_{\text{info}}}{\text{Split Info}}$$

$$\text{Split Info} = - \sum_{i=1}^k P(v_i) \log_2 P(v_i)$$

決策樹的優點

- 決策樹很容易瞭解，而且漂亮地映射成規則的集合
- 決策樹已成功應用於實際的問題上
- 決策樹不會對於有關資料的種類做事先的假設
- 決策樹能建立資料集的模型，且包含數值的及類別的資料

資料探勘

3.2 推論出關聯法則

親和力分析（affinity analysis）是決定事物在一起與否的一般化過程。典型的應用就是市場購物籃分析，希望在購物期間，決定顧客可能購買的項目。

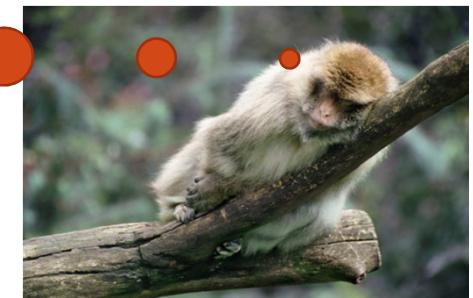
雜貨店中賣的四項產品：

- 1. 牛奶
- 2. 起司
- 3. 麵包
- 4. 蛋

可能的購
買組合

- 1. 假如顧客買牛奶，他們也會買麵包
- 2. 假如顧客買麵包，他們也會買牛奶
- 3. 顧客若買牛奶與蛋，也會買起司與麵包
- 4. 顧客若買牛奶，起司和蛋，他們也會買麵包

購買牛奶的事件會導
致購買麵包的可能性
有多大？



資料探勘

假設總共有 10,000 筆顧客交易涉及牛奶的購買，其中同時買牛奶又買麵包的交易有 5,000 筆，則**信賴度**為 $5000 / 10000 = 50\%$

假設總共有 20,000 筆顧客交易涉及麵包的購買，其中同時買麵包又買牛奶的交易有 5,000 筆，則**信賴度**為 $5000 / 20000 = 25\%$

信賴度和支持度

一個規則的信賴度不會提供另外一些重要的資訊，如一個關聯法則中所找到的屬性值，它們在所有交易中所佔的比率。所以對一條規則而言，此統計學的方法就是所謂的**支持度**（support）。簡單地說支持度就是在特定的關聯法則中包含所有項目目錄的資料庫中之最小的交易百分率。



資料探勘

- **支持度**的定義為決策變數在資料庫中所出現的比例，表現形式為 $Sup(X)$ ，也就是在整個資料庫L中出現的比例，支持度越高，越值得重視。支持度代表事件的發生機率。 $Sup(X \rightarrow Y)$ 代表同時發生X和Y兩個交易事項的機率，支持度介於0%和100%之間。

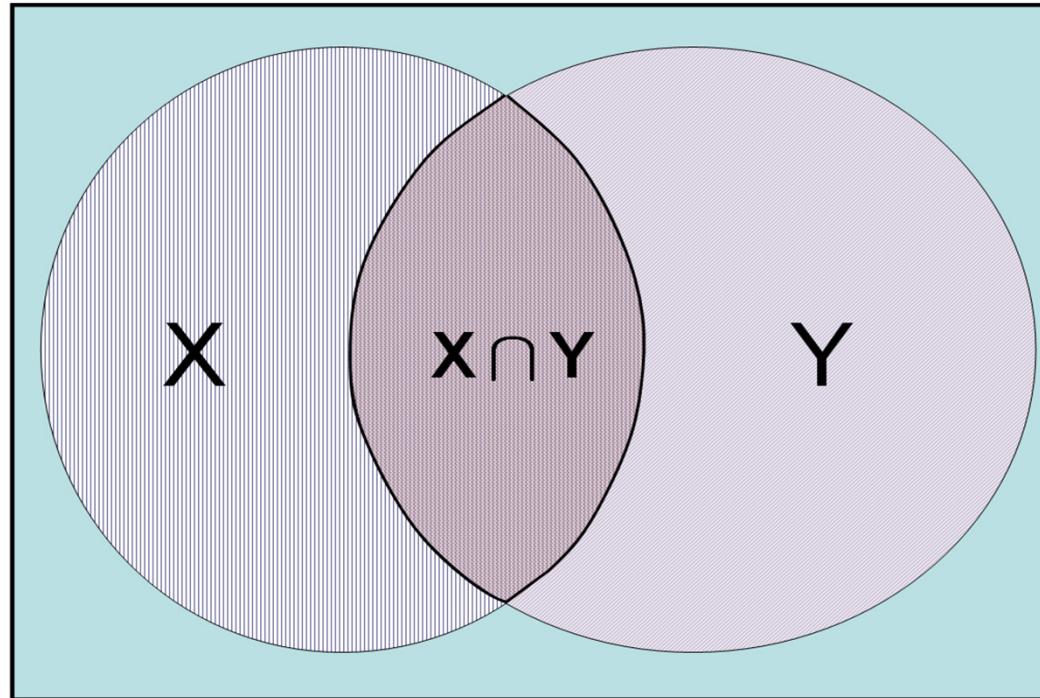
$$Sup(X) = \frac{\text{項目集合 } X \text{ 在資料庫中出現的總次 數}}{\text{資料庫中的總交易筆數}}$$

資料探勘

- 可靠度的定義此關聯性法則可信的程度，也就是某決策變數X已確知或成立時，另一決策變數Y發生或成立的機率，與統計中的條件機率相同，表現形式為 $Conf(X \rightarrow Y)$ 。 $Conf(X \rightarrow Y)$ 代表發生X的交易事項下，發生Y交易事項的機率，可靠度介於0%和100%之間。

$$Conf(X \rightarrow Y) = \frac{Sup(X \cap Y)}{Sup(X)}$$

資料探勘



- 一般而言，關聯性法則的**支持度**及**可靠度**皆必須分別大於使用者訂定的最低限制，才能據此判定其為有意義的關聯性法則。

資料探勘

表 3.3 □ 信用卡促銷資料庫的子集合

雜誌促銷	手錶促銷	壽險促銷	信用卡保險	性別
是	否	否	否	男
是	是	是	否	女
否	否	否	否	男
是	是	是	是	男
是	否	是	否	女
否	否	否	否	女
是	否	是	是	男
否	是	否	否	男
是	否	否	否	男
是	是	是	否	女

資料探勘

探勘關聯法則：一個例子

一個出色的演算法就是 *Apriori* 演算法 (Agrawal et al., 1993)。此演算法推論出所謂的**項目集合** (item set)。項目集合為屬性-數值的組成，且滿足特別的覆蓋範圍需求。這些屬性-數值的組合沒有滿足收斂的需求就會被拋棄。由於如此，此規則的推論過程可以在一個合理的時間區間中完成。

表 3.4 □ 單一項目集合

單一項目集合	項目編號
雜誌促銷 = 是	7
手錶促銷 = 是	4
手錶促銷 = 否	6
壽險促銷 = 是	5
壽險促銷 = 否	5
信用卡保險 = 否	8
性別 = 男	6
性別 = 女	4

假設收斂的信賴度
需求是 信賴度 >3

資料探勘

表 3.5 □ 二個項目集合

二個項目集合	項目編號
雜誌促銷 = 是 且 手錶促銷 = 否	4
雜誌促銷 = 是 且 壽險促銷 = 是	5
雜誌促銷 = 是 且 信用卡保險 = 否	5
雜誌促銷 = 是 且 性別 = 男	4
手錶促銷 = 否 且 壽險促銷 = 否	4
手錶促銷 = 否 且 信用卡保險 = 否	5
手錶促銷 = 否 且 性別 = 男	4
壽險促銷 = 否 且 信用卡保險 = 否	5
壽險促銷 = 否 且 性別 = 男	4
信用卡保險 = 否 且 性別 = 男	4
信用卡保險 = 否 且 性別 = 女	4

資料探勘

一般性考量

由於關聯法則有能力在大型的資料庫中找到關聯性，而且沒有選擇單一的相依變數的限制，所以它們的可見度特別高。然而，因為很多被發現的關聯是很明顯的，以致於是沒有價值的法則，所以必須在使用關聯法則前事先警告使用者。

資料探勘

3.3 K-Means 演算法

K-Means 演算法 (Lloyd, 1982) 是一個簡單而且有效的統計學分群技巧。為了幫助你較了解非監督式分群法，我們看看 K-Means 演算法如何將資料分割成不同的群集。

使用 K-Means 的例子

$$\text{距離 } (A-B) = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$$

$$\text{距離 } (C_1-1) = 0.00$$

$$\text{距離 } (C_1-2) = 3.00$$

$$\text{距離 } (C_1-3) = 1.00$$

$$\text{距離 } (C_1-4) \approx 2.24$$

$$\text{距離 } (C_1-5) \approx 2.24$$

$$\text{距離 } (C_1-6) \approx 6.02$$

$$\text{距離 } (C_2-1) = 1.00$$

$$\text{距離 } (C_2-2) \approx 3.16$$

$$\text{距離 } (C_2-3) = 0.00$$

$$\text{距離 } (C_2-4) = 2.00$$

$$\text{距離 } (C_2-5) \approx 1.41$$

$$\text{距離 } (C_2-6) \approx 5.41$$

表 3.6 □ K-Means 輸入值

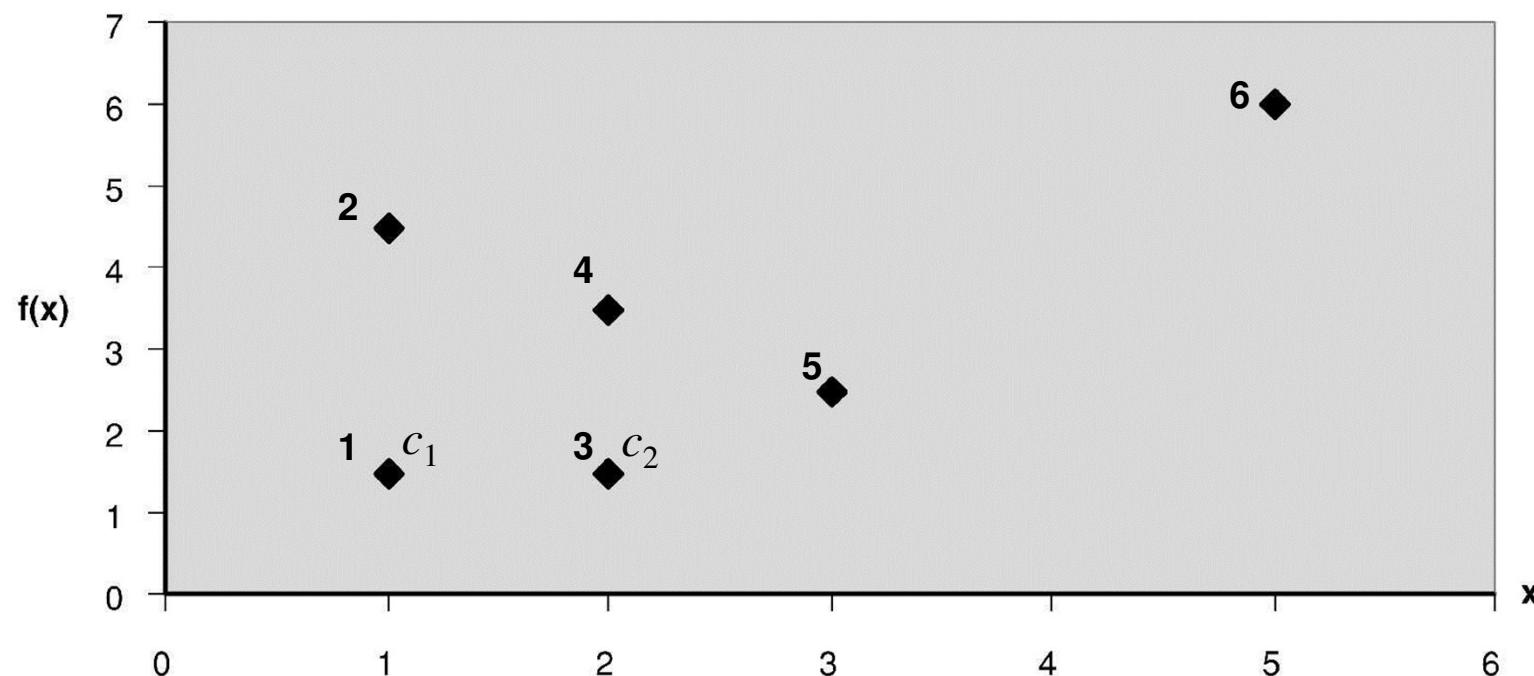
範例	X	Y
1	1.0	1.5
2	1.0	4.5
3	2.0	1.5
4	2.0	3.5
5	3.0	2.5
6	5.0	6.0

$$c_1 = (1.0, 1.5)$$

$$c_2 = (2.0, 1.5)$$

資料探勘

圖 3.6 □ 表 3.6 的座標圖



資料探勘

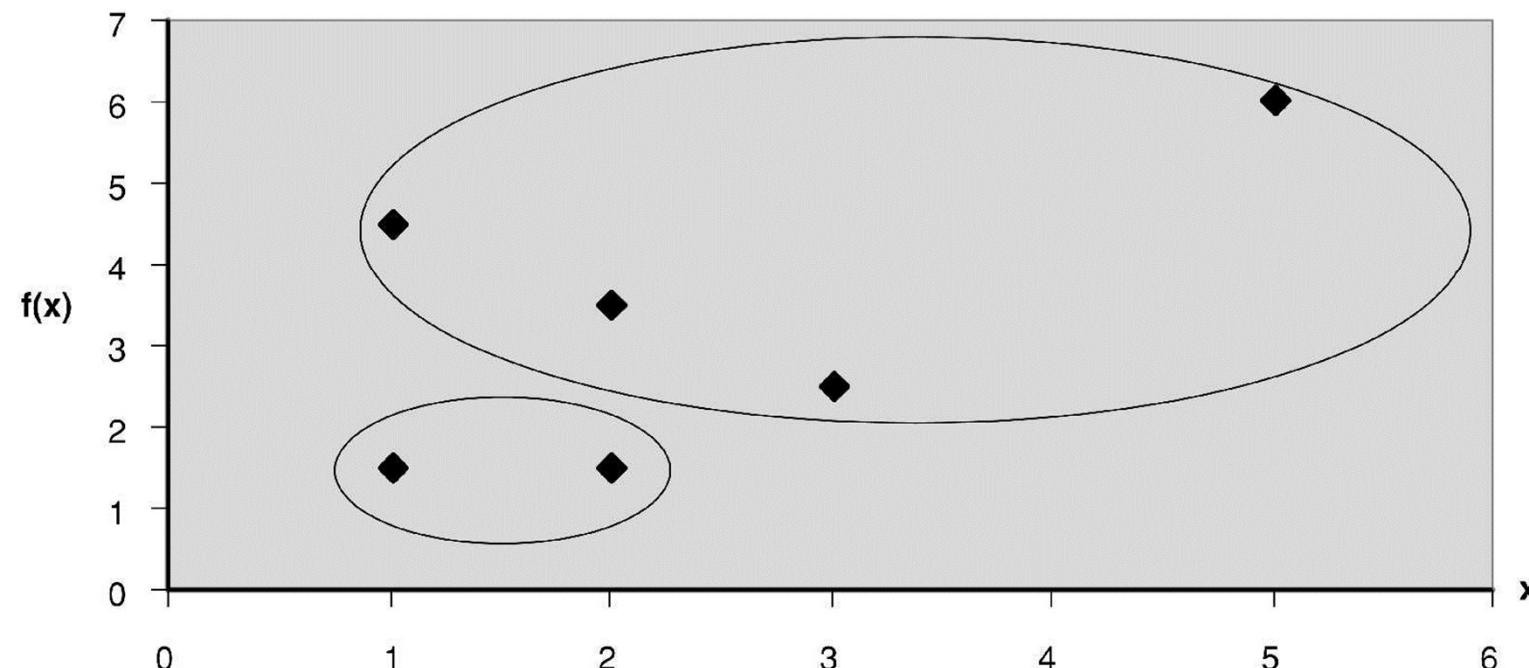
表 3.7 表示三種由 K-Means 演算法重複去應用表 3.6 的資料之後所形成的分群結果。圖 3.7 表示最經常發生的分群，此群集在表 3.7 的結果 2 中可以看到。

表 3.7 □ 一些 K-Means 演算法 (K = 2) 的應用

結果	群集中心	群集點	單位面積錯誤率
1	(2.67 , 4.67)	2, 4, 6	14.50
	(2.00 , 1.83)	1, 3, 5	
2	(1.5 , 1.5)	1, 3	15.94
	(2.75 , 4.125)	2, 4, 5, 6	
3	(1.8 , 2.7)	1, 2, 3, 4, 5	9.60
	(5 , 6)	6	

資料探勘

圖 3.7 □ 一個由表 3.6 的資料完成的 K-Means 分群結果 ($K = 2$)



一般性考量

此 K-Means 方法很容易了解且實行。

k-Means 演算法

- *k*-Means 族群推算法的步驟如下：
 1. 決定要找出多少個族群。換句話說，決定 k 值。
 2. 隨意選出 k 個資料來當做這 k 個族群的中心點。
 3. 由這 k 個資料點為起頭，建立出首輪的 k 族群。
在這個時候，每一筆資料都暫時屬於某一個族群
 4. 找出每一個族群新的中心點
 5. 重複步驟3及步驟4，直到終止條件成立

k-Means 演算法

- 最常被用到的終止條件有兩種：
 - 每個族群的**中心點**不再改變。
 - 某種**收斂標準**已經達到。
- 一種常見的收斂標準是Sum of Squared Errors (SSE)：

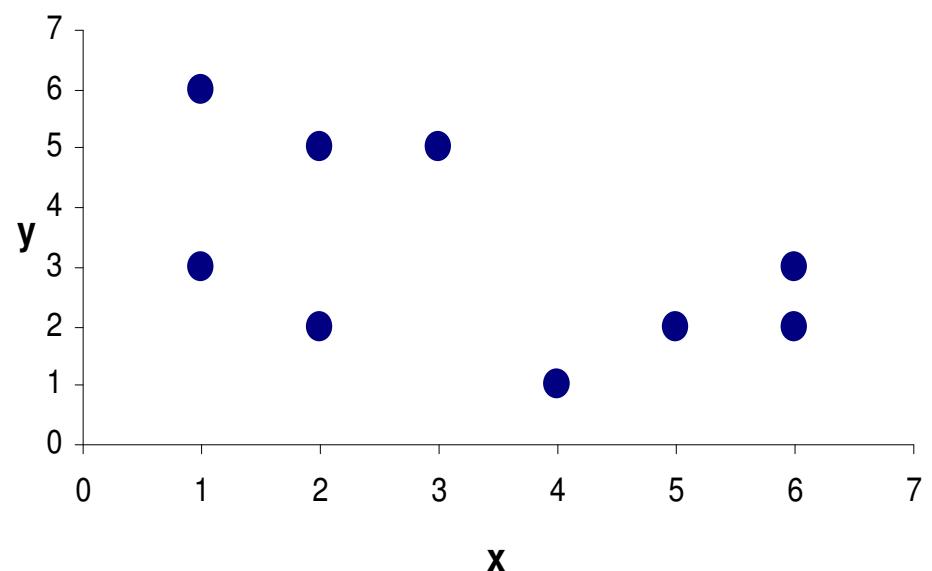
$$\text{SSE} = \sum_{i=1}^k \sum_{p \in C_i} d(p, m_i)^2$$

- $p \in C_i$ 代表在族群*i*中的每一個資料點， m_i 是族群*i*的中心點，而 $d(p, m_i)$ 代表每一個資料點和它所屬族群中心點的距離。

k -Means 演算法 -範例

9個點分成3群

a	(1,3)
b	(2,5)
c	(3,5)
d	(1,6)
e	(4,1)
f	(5,2)
g	(6,2)
h	(6,3)
i	(2,2)



k-Means 演算法-範例-續

步驟1: $k=3$

步驟2: 隨意選出3個資料來當作這3個族群的中心點

步驟3: 由這3個資料點為起點，建立出首輪的3個族群

資料點	與 m_1 的距離	與 m_2 的距離	與 m_3 的距離	族群
a	0.00	2.24	2.83	族群1
b	2.24	0.00	1.00	族群2
c	2.83	1.00	0.00	族群3
d	3.00	1.41	2.24	族群2
e	3.61	4.47	4.12	族群1
f	4.12	4.24	3.61	族群3
g	5.10	5.00	4.24	族群3
h	5.00	4.47	3.61	族群3
i	1.41	3.00	3.16	族群1

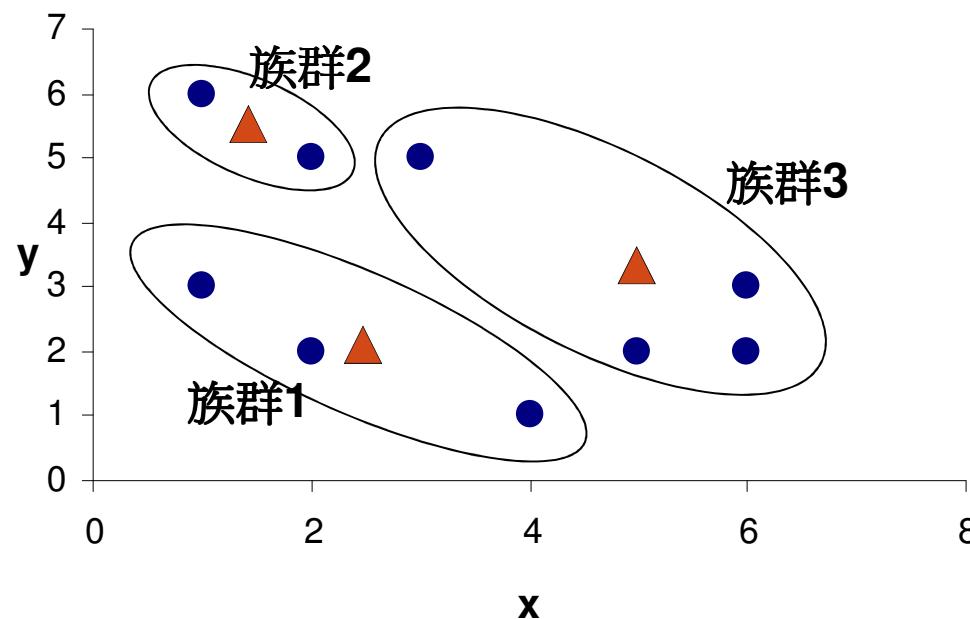
k-Means 演算法-範例-續

步驟4:找出每一個族群新的中心點

$$\text{族群1} = \{(1+4+2)/3, (3+1+2)/3\} = (2.33, 2)$$

$$\text{族群2} = \{(2+1)/2, (5+6)/2\} = (1.5, 5.5)$$

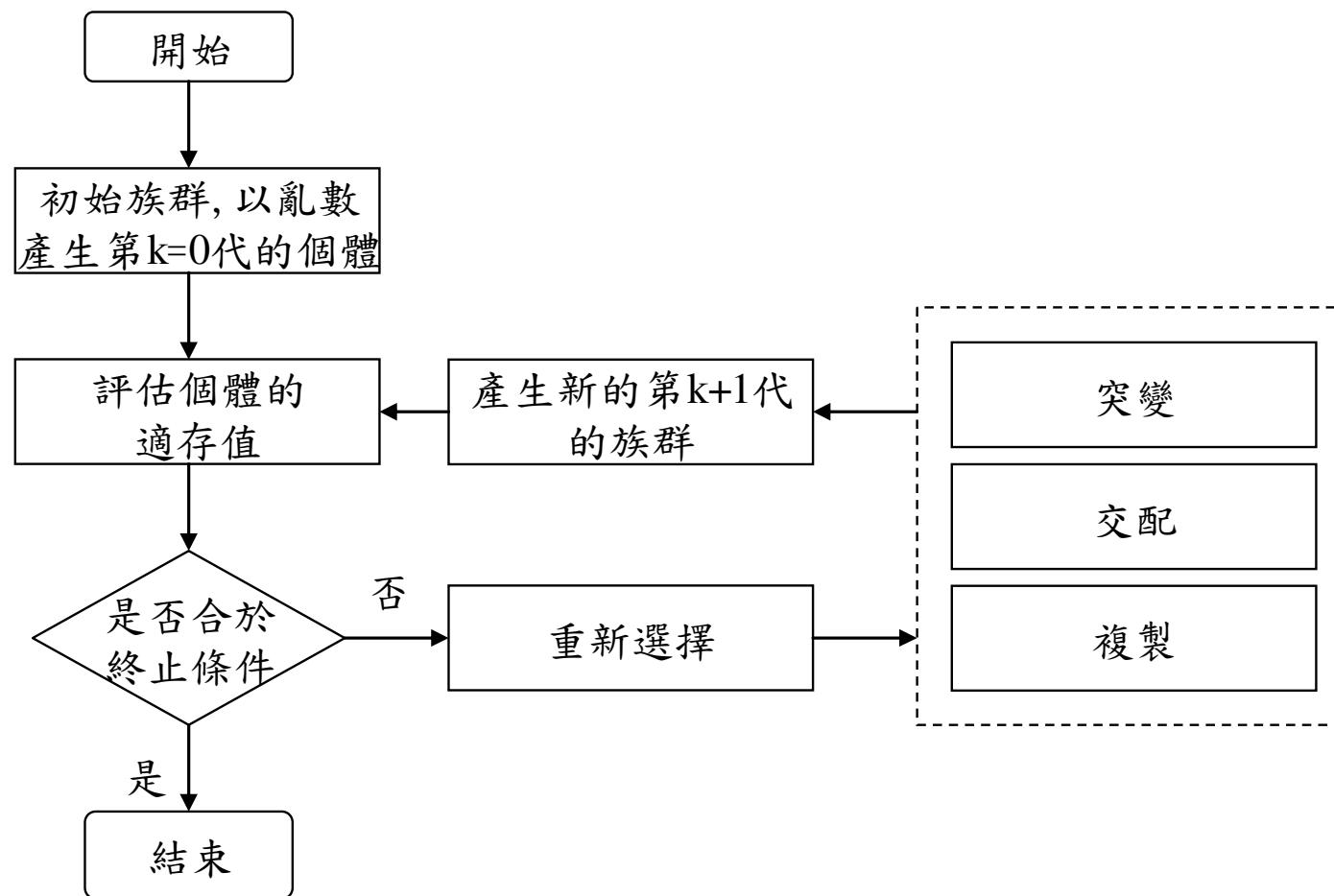
$$\text{族群3} = \{(3+5+6+6)/4, (5+2+2+3)/4\} = (5, 3)$$



遺傳學演算法(Genetic Algorithm , GA)

- 又稱為**基因演算法**
- 1975年由John Holland(1975)所提出
- 主要目的在於利用達爾文**進化論**“物競天擇，適者生存”的方式
- 近來被廣泛的應用於搜尋各類問題的可能最佳解
- 藉由生物物種的基本運算子，在每代間進行演化，終而尋得適當問題的可能最佳解

演化流程



演化流程 (cont.)

- 遺傳演算法在求解過程中，大致上可以分為七個步驟進行：
 1. 設計編碼方式(染色體個體表示法)
 2. 決定族群大小規模
 3. 設計**適合度函數**，決定染色體個體適合度的**評估標準**
 4. 決定**挑選(Selection)**與**複製**的方法
 5. 定義交配與交配率
 6. 定義突變與突變率
 7. 決定**終止條件**

演化流程 (cont.)

1. 初始族群 (*Initial population*)

- 先必須隨機的產生 q 個染色體，這 q 個染色體稱為初始族群
- 族群中的每個染色體亦稱之為一個個體(Individual)

ex: $C_1 : \underline{5, 4, 9, 4, 15, 6, 3, 3, 6, 3, 9, 3, 6, 6, 12, 6, 18, 6, 4, 4, 8, 4, 12, 4}$

$C_2 : \underline{4, 3, 8, 4, 12, 3, 4, 3, 7, 3, 10, 3, 6, 6, 10, 4, 16, 6, 6, 4, 10, 4, 14, 4}$

$C_3 : \underline{5, 5, 9, 4, 13, 4, 6, 5, 11, 5, 16, 5, 5, 4, 9, 4, 13, 4, 6, 4, 10, 4, 14, 4}$

$C_4 : \underline{3, 3, 6, 3, 9, 3, 6, 5, 11, 5, 16, 5, 6, 6, 12, 6, 18, 6, 4, 4, 8, 4, 12, 4}$

$C_5 : \underline{4, 3, 7, 3, 10, 2, 5, 3, 8, 3, 11, 3, 6, 3, 12, 3, 18, 3, 4, 4, 8, 4, 12, 4}$

演化流程 (cont.)

2. 編碼 (*Encoding*)

遺傳演算法依編碼資料型態的不同，可以分為：
整數編碼、實數編碼、二進位編碼、文字編碼以及符號編碼等

ex:

	1	2	3	4	5	6												
(a)二進位編碼	0	1	1	0	0	1	0	1	1	1	0	1	1	1	0	0	1	1
(b)實數編碼	1	2	3	4	5	6	73.5	17.3	110.9	1.2	83.5	71.4						
(c)符號編碼	1	2	3	4	5	6	F	A	Q	B	E	C						

演化流程 (cont.)

3. 染色體 (*chromosome*)

- 運算子運算的對象是染色體
- 染色體的型態是由**一串數字**串接而成的**字串**
- 每一個染色體都對應到**目標問題**的一個解
- 將每個**參數之編碼字串**串接起來就組成**染色體**
- 染色體的長度及個體都會影響到目標問題的精確度及難度
- 染色體長度越長則目標問題的分割就越精密

演化流程 (cont.)

4. 適合度函數 (*fitness*)

- 用來評估族群中每一個個體**適合度的指標**
- 進化論中提到“適者生存，不適者淘汰”的基本概念
- 適合度函數值越高表示個體的**適合度越好、競爭力越強**，相對的也就越可能將基因遺傳到下一代身上
- 可以藉由設計不同的**適合度函數**來達到**控制演化**，進而產生不同的結果

演化流程 (cont.)

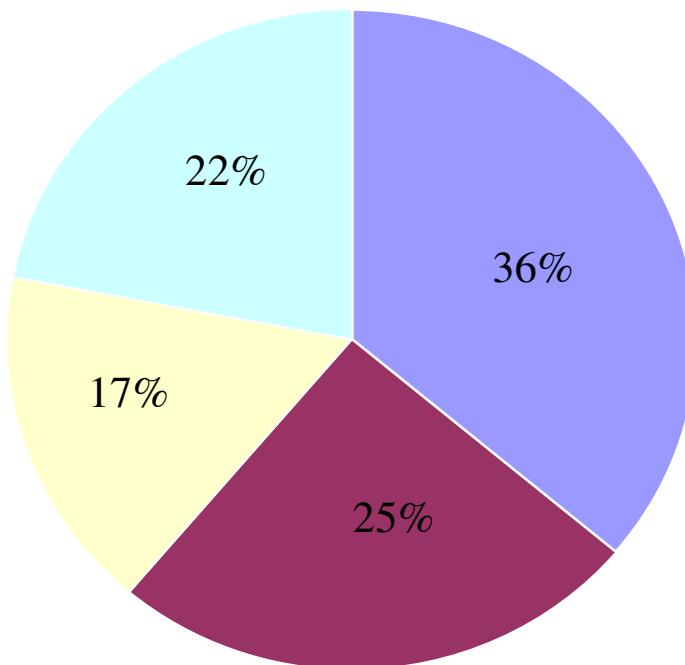
5. 複製 (*reproduction*)

- 依據每一個個體的適合度函數值之高低來決定該個體被複製的機率
- 適合度函數值越高的個體，被複製成為下一代的個體之機率就相對的越高，反之就會被淘汰
- 而選擇複製的過程，常被提及的方法有以下兩種：**輪盤式選擇法**(*Roulette Wheel Selection*)及**競爭式選擇法**(*Tournament Selection*)兩種。

演化流程 (cont.)

輪盤式選擇法(*Roulette Wheel Selection*)：

適合度函數值越大的話，則在輪盤上佔有的面積比例就越大



演化流程 (cont.)

競爭式選擇法(*Tournament Selection*)

- 在每一代的演化過程中首先隨機地選取兩個或更多的染色體個體(字串)
- 具有最大適合度函數值的基因即被選中送至交配池中
- 重複的選取，一直到**交配池**中染色體的個體數與族群的個體數相同
- 同樣的，適合度函數值越高的個體就越容易被選中

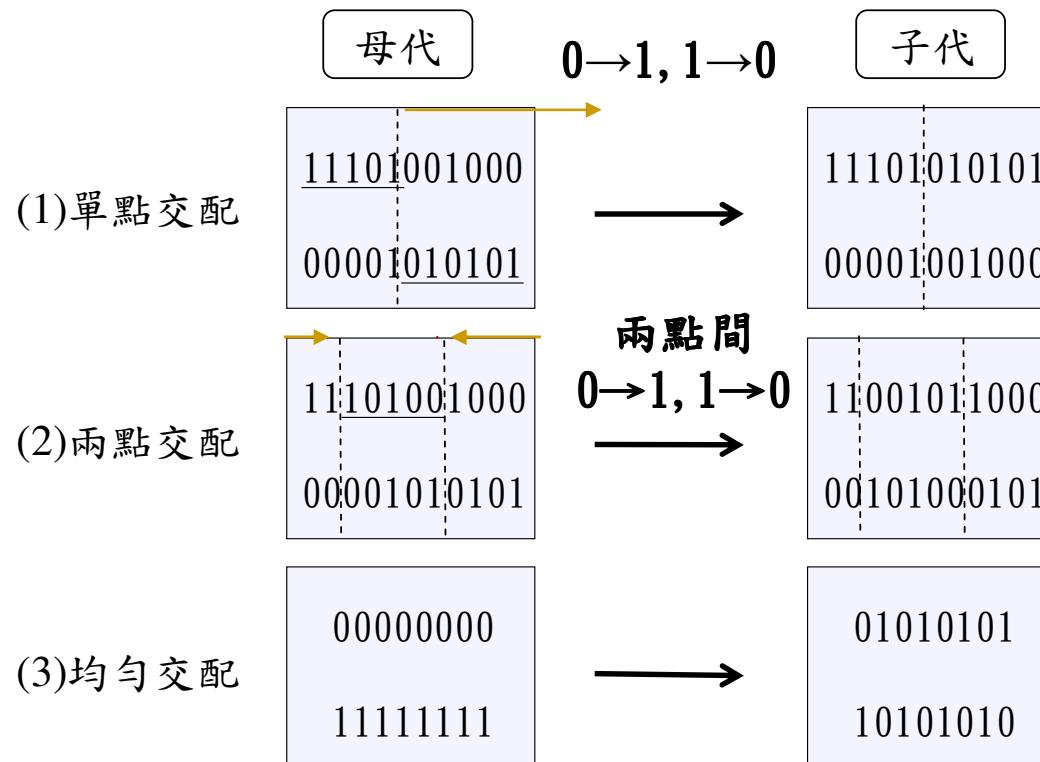
演化流程 (cont.)

6. 交配 (*crossover*)

- 交配池中的兩個母代個體，彼此交換位元資訊進而產生兩個新的個體
- 藉由累積前代較為優良的位元資訊，以期待能夠產生出更優秀的個體
- 事先設定的交配機率(probability of crossover)來決定是否要進行交配的運算
- 依據交配方式的不同，又可分為單點、兩點以及均勻三種

演化流程 (cont.)

依據交配方式的不同，又可分為單點、兩點
以及均勻三種



演化流程 (cont.)

7. 突變 (*mutation*)

- 純粹靠著**複製**與**交配**這兩個運算的話，無法創造出具有新特性的個體
- 希望透過**突變**的方式使得新的個體可以**跳脫單純交配**的**區域解**中，進而期能產生**全域最佳解**
- 主要突變方式大致上分為三種：
 - (1) 基本位突變(Simple Mutation)
 - (2) 均勻突變(Uniform Mutation)
 - (3) 非均勻突變(Non-Uniform Mutation)

演化流程 (cont.)

8.族群的取代

- 經過**複製**、**交配**以及**突變**這三個運算過程之後，在交配池中產生了新世代族群的新個體
- 族群取代的方式可以分為以下兩種：

(1)整代取代：

全部以新的個體取代舊族群中的舊個體，成為新的族群。

(2)保留菁英：

此方式最大的好處是可以確保每一個新一代族群的適合度函數值不會降低。

演化流程 (cont.)

9. 演化終止

- 通常遺傳演算法的終止條件會分為三種方式：
 - (1) 當演化流程 **達到指定的世代數目** 時；
 - (2) 當演化流程 **達到要求的目標** 時；
 - (3) 當演化流程 **停滯** 或者是 **已經達到某種飽和現象** 時。

資料探勘

一般性考量

雖然很少商業化的資料探勘產品會包含進基因學習的部分，但基因演算法會愈來愈流行。而當使用以問題解決方法為基礎的基因學習時會有一些考量：

- 基因演算法被設計來找出總體的最佳解，但此處不保證任何求出的解會是總體最佳的解。
- 適合函數決定基因演算法的計算複雜度，而適合函數所包含的計算可能會是成本很高昂的計算工作。
- 基因演算法在適合函數可理解的範圍中去解釋它們的計算結果。
- 轉換資料到一個適合的基因演算法是可以質疑其可行性的。