第二章

畫風、影像特徵與資料探勘

在這一章中，2.1 節將描述何為畫風，2.2 節描述常見的低階影像特徵，2.3 節則會介紹常見的資料探勘技術。

2.1 畫風

所謂的畫風是指大部分畫作中所呈現的共同特徵；從另一個角度而言，畫風是指與其他畫家的畫作相比，在畫作的共同特徵上之獨特性與差異性。

基於以上畫風的兩個特性，我們把繪畫影像風格的探勘包括三個議題。一、feature extraction，從美術影像中萃取低階影像特徵。二、mining frequent patterns，從所有該畫家畫作的低階影像特徵找出共同的個人畫作特徵。三、classification，找出每個畫家與別人不一樣的個人畫作特徵，就是定量描述的繪畫影像風格。

2.2 影像特徵

關於從影像中萃取低階影像特徵，我們可以利用 content-based image retrieval 與 MPEG-7 的相關研究。

2.2.1 Content-Based Image Retrieval

在 content-based image retrieval 之中，常用的 feature 有顏色（color）、形狀（shape）、材質（texture）、空間關係（spatial relationship）等。

顏色是 image retrieval 常見的影像特徵之一。使用者以顏色進行查詢時，系統會找出相近的影像，而所謂的相近，可能是所使用的顏色種類或是各個顏色所佔畫面的比例都與使用者所選的相近。常見的做法是統計各種顏色出現的次數，稱為 color histogram。如果影像共有 n 個顏色，color histogram 會表示成 n 維向量，並以此做為計算兩張影像相似度的依據。在萃取顏色特徵時，下面兩議題必須加以考慮，一是顏色的表示方式，另外是顏色的數目。

顏色的表示方式，也就是表示顏色的空間模型(color space model)。顏色的空間模型有很多種，最常見的空間模型是 RGB。RGB color space 是一個直角座標系統，維度分別是紅（red)、綠（green)、藍（blue）三原色。但是，最符合人類視覺原理的卻是 HSV。H(hue)代表色度，S(saturation)代表飽和度，V(value)代表明亮度(brightness)，色度是平常區分顏色的主要成分，例如紅、橙、黃、綠、藍等；彩度表示顏色鮮艷、飽和的程度，就是顏色中混合白色的多寡，低彩度的顏色表代混合白色愈多，例如粉紅色比紅色的彩度低；亮度就是顏色的明暗度，就是顏色中混合黑色的程度，低亮度的顏色表代混合黑色愈多，例如墨綠色比綠色的亮度低。此外，若不想讓光源影響影像擷取的效果，則可以不考慮 HSV 中的明亮度。

至於顏色的數目，顏色越多其精確度也越高，但卻會降低查詢處理的速度，因此大部分的查詢系統會在顏色數目與處理速度間折衷考慮。例如 IBM 的 QBIC 系統以 256 色的顏色分佈表示顏色特徵[15]，而哥倫比亞大學的 WebSeek 系統其顏色數目則是 166 色[33]。

顏色特徵的表示方法除了 color histogram，顏色佈局(color composition)也是常見的表示方式之一。Color histogram 因為只統計顏色整體出現的次數，並未考慮空間分佈，以圖 2.1 為例，左右兩影像的 color histogram 完全相同。顏色佈局就可以表示出這兩張圖之間的差異。

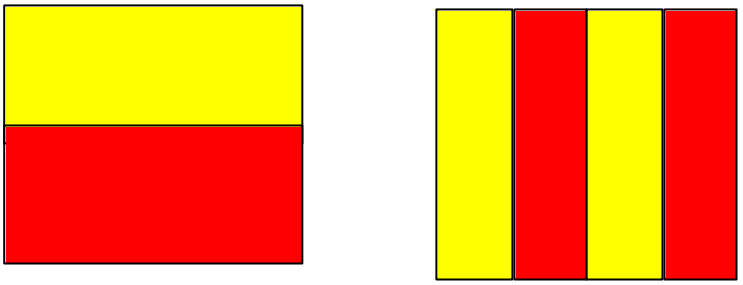


圖 2.1 左右兩不同影像，顏色分佈卻完全相同

人類在辨識影像的過程中，除了顏色之外，還會辨認影像中主要物件的相關特徵，其中包含形狀[21]。在一些 image retrieval 系統中，使用者可以藉由描述想要搜尋物體的形狀來找到相關的 image，例如使用者想找長條型的板凳。想找出影像中物體的形狀必須先經過影像處理的技術，例如先以彩色分割演算法(color-based segmentation algorithm)找出影像中之重要物體(dominant object)，再以邊線偵測演算法(edge detection algorithm)或素描演算法(outlining algorithm)找出物體的輪廓。

形狀特徵的表示可由面積、中心點、周長、方位(orientation)等度量組成。這些表示法不受位移、旋轉的影響。另外，傅立葉描述(fourier descriptors)、連鎖碼(chain code)也可表示形狀。Chain code 先將 360 度等分成八個方向，並加以編號。然後從要描述的形狀的最高一個頂點起，沿著順時針方向，以那八個方向進行描繪，直到回到起點為止。

主要物件的相關特徵，除了形狀外，也包括材質。影像處理的研究領域對於紋路已有不少的研究。IBM 的 QBIC 以 Tamura 紋路特徵表示紋路，也就是說以明暗對比度(contrast)、緊密度(coarseness)、方向(directionality)表示紋路特徵。

除了指定所要查詢的主要物件的低階特徵外，使用者在進行查詢時還可以使用主要物件間的空間關係做為條件。二維字串（2D String）是表示物件空間關係的常見作法之一，要將物件的空間關係表示成 2D String 之前要先經過圖形識別的處理，找出各個物件的最小邊界（minimum bounding rectangle），並且給予物件代號。而那個物件的參考點就是它的最小邊界的重心[6]。再分別以兩個字串記錄 X 軸方向與 Y 軸方向的物體空間關係就是二維字串。

2.2.2 MPEG-7

由於多媒體資料廣泛充斥在網際網路上，使得搜尋越來越困難。因此在 1996 年十月，MPEG (Moving Picture Experts Group)開始制定一個新的標準 MPEG-7。MPEG-7 定位在描述多媒體的內容。但是如圖 2.2 所示，這個標準中所制定的範圍，只包含了描述多媒體內容，而不包含傳統多媒體 content-based retrieval 中有關特徵萃取以及相似度計算的部分[5, 27]。MPEG-7 如何計算相似度的部分，則在 MPEG-7 標準外的 MPEG-7 XM(the eXperimentation Model) 中介紹[28]。

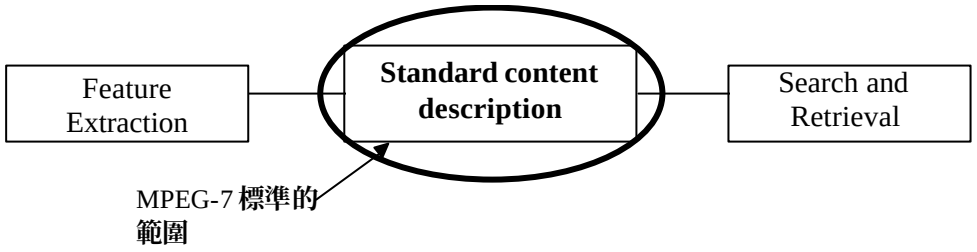


圖 2.2 MPEG-7 標準定位圖 [27]

下列是 MPEG-7 術語的介紹[5]：

- Data: 要被 MPEG-7 描述的影音資料，不特定針對某些儲存方式或是某些編碼後的資料。
- Feature: 是從 data 中找出來，可以跟別的 data 區別的特性，而且是對人有意義的。
- Descriptor: Descriptor (D)是 feature 在 MPEG-7 中的表現形式，定義出如何表示這個 feature 的語法，以及代表的語意。
- Descriptor Value: 是 descriptor 的 instantiation。舉例來說，如果有一個 descriptor 代表了整數，那麼 5 就是一個 descriptor value。
- Description Scheme: 一個 description scheme (DS)可以定義出各個元件(component)之間的語法和語意。至於一個 DS 的元件可以是 descriptor 或是另外一個 DS。以 C 程式語言的角度來說，一個 DS 就是一個 composed data type，

如 structure。而 descriptor 就是 primitive data type 如 int。

- Description: 是 description scheme 的 instantiation。
- Coded Description: 編碼過的 description，像是針對 compression efficiency，error resilience 或是 random access。
- Description Definition Language: 可以讓使用者自行定義新的 DS 或是新的 descriptor 的 MPEG-7 語言。透過 DDL，使用者也可以擴充或修改已經有的 DS。DDL、DS 與 descriptor 的關係如圖 2.3 所示：

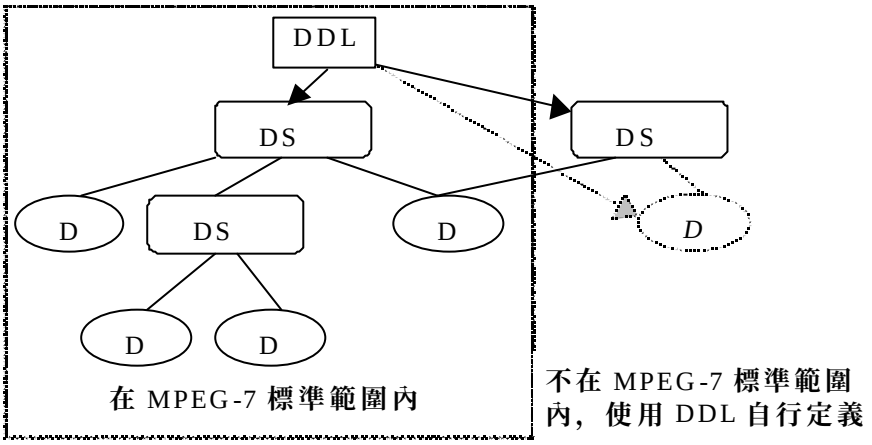


圖 2.3 DDL、DS 與 D 的關係圖

2.2.2.1 Overview

在介紹完 MPEG-7 的術語之後，我們可以知道，MPEG-7 標準的主要內容，除了一系列的 descriptor 跟 description scheme 之外，還包含了 description definition language，讓使用者自行擴充 descriptor 及 DS。最後，MPEG-7 還會包含一些 description 的編碼方式。

在 MPEG-7 標準中，包含了以下七個部分：Systems、Description Definition Language、Audio、Visual、Generic Entities and Multimedia Description Schemes、Reference Software、Conformance。其中跟我們研究最直接相關的是 Visual 部分，這

是 MPEG-7 標準中，定義跟視覺有關的部分，例如顏色，材質，形狀，以及位移。

2.2.2.2 Visual Part

因為本研究注重在 visual information 的部分，所以，接下來只針對 MPEG-7 Visual 做詳細介紹。Visual part 可以大致區分成兩個部分[32, 38]。第一個部分定義描述的基本單位，在 MPEG-7 standard 中稱為 supporting tool。第二個部分，則是針對影像特徵進行描述的主要部分。目前共區分成 color、shape、texture、motion、localization 及 other 六種。

本研究所實作的部分是影像特徵的部分。這個部分共有 color、shape、texture、motion、localization 及 other 六種，不過因為本研究只針對 color 的部分進行實作，因此將不介紹其他五種影像特徵的部分。

Visual part，所定義的 descriptor 中有關 color 的部分共有七個 descriptor，分別是 color space descriptor，color quantization descriptor，dominant color descriptor，scalable color descriptor，color layout descriptor，color structure descriptor，以及 GoF/GoP color descriptor。以下將分別介紹這些 descriptor。

Color Space Descriptor

Color space descriptor 是用來指定顏色的表示 model。到目前一共支援六種 color model，分別是 RGB，YCbCr，HSV，HMMD，Monochrome，以及一個可以將上面未支援的 color model 轉換成 RGB 的轉換矩陣(MPEG-7 稱之為 Linear Matrix)。另外，color space descriptor 只被使用在 dominant color descriptor 中。

因此，在這一個 descriptor 中，可以出現的值分別為：RGB、YcbCr、HSV、HMMD、LinearMatrix、Monochrome。

例如，一個以 text 表示的 color space 如下：

<ColorSpace type=”RGB” />

Color Quantization Descriptor

Color quantization descriptor 是用來指定在 color model 中，各別的 component 最多記錄多少種顏色，值的上限是 8192（12 bits），在三個 component 的 color model 中，最多可以表示 2^{36} 次方種顏色。跟 color space descriptor 一樣，color quantization descriptor 只被使用在 dominant color descriptor 中。

Color quantization descriptor 的例子如下，這個例子表示 YCbCr color model 中可以有 6*3*3=54 種顏色：

<ColorQuantization>

<Component>”Y”</Component>

<BinNumber>6</BinNumber>

<Component>”Cb”</Component>

<BinNumber>3</BinNumber>

<Component>”Cr”</Component>

<BinNumber>3</BinNumber>

</ColorQuantization>

Dominant Color Descriptor

Dominant color descriptor 是用來描述一塊區域中主要出現的顏色，及其相關資訊，例如顏色在空間上的緊密程度，或是每一種主要顏色佔的面積。

Dominant color descriptor 的參數可以分成兩大部分，第一部分是跟所有 dominant color 有關的參數，例如 dominant color 的個數，採用的 color space 為何，顏色數多少，空間上是否緊密。第二部分則是分別記錄每一個 dominant color 所佔的面積，顏色的 index，以及在同一個 quantization 區間中的變異量。

首先介紹第一部分的參數：

- size：指定 dominant color 的個數，最少為 1，最多為 8。
- color space：利用 color space descriptor 指定 color model。預設為 RGB。
- color quantization：利用 color quantization descriptor 指定 color model 中，每個 color component 在 quantization 後的個數。預設為 32 個。
- spatial coherency：記錄選取的 dominant color 所代表的 pixels 在空間中集中的程度，值越高的越集中。如下圖 2.4 所示，紅色的 pixel 在左邊的圖(a)中較不集中，在右邊的圖(b)中較為集中。Spatial coherency 預設值為 0。

(a) (b)

圖 2.4 (a) Low Spatial Coherency (b) High Spatial Coherency [38]

接著我們將介紹第二部分的參數，這個部分每個 dominant color 都會記錄，因此如果有 3 個 dominant color，這部分就會出現三次。然而，這裡的 dominant color，在 raw pixel 中，並不會只對應到一種顏色，因為 quantization 後的結果，會使得這裡的 dominant color 對應到一個區間，在這區間內的所有顏色都被一視同仁的計算。

- percentage：在這 dominant color 的區間內，所有的顏色佔畫面的百分比。
- color value index：經 quantization 後，這 dominant color 的 index 值。如果是三個 color component 的 color space，就會分別有三個值。
- color variance：因為在這區間內的所有顏色都被一視同仁的計算在內，可是畢竟真實出現的顏色還是不一樣，可以用 color variance 來記錄這個不同點。Color variance 的預設值為 0。

Dominant color descriptor 的例子如下：

<DominantColor size=1>

<ColorSpace type="RGB" />

<ColorQuantization>

<Component>"R"</Component><BinNumber>4</BinNumber>

<Component>"G"</Component><BinNumber>4</BinNumber>

<Component>"B"</Component><BinNumber>4</BinNumber>

</ColorQuantization>

<SpatialCoherency value=31 />

<Value>

<Percentage value=15 />

<ColorValueIndex value=41 />

<ColorVariance value=0.5 />

<ColorValueIndex value=2 />

<ColorVariance value=0.9 />

<ColorValueIndex value=10 />

<ColorVariance value=0.1 />

</Value>

</DominantColor>

Scalable Color Descriptor

Scalable color descriptor **就是記錄** color histogram**，用來統計顏色出現的次數。** Scalable color descriptor **指定用** HSV color model**，可以指定的** quantization **為** 16，32，64，128，256。Color histogram **的值會用** Haar transfrom **來編碼。**

Scalable color descriptor **的例子如下，其中** length=16 **代表這個** color histogram **一**

共有 16 個 color bins。

<ScalableColor>

<value length=16>

10 32 64 22 98 243 50 69 33 25 89 103 9 11 200 254

</value>

</ScalableColor>

Color Layout Descriptor

Color layout descriptor 是用來保留有關顏色在空間上的分佈，解決 dominant color descriptor 跟 scalable color descriptor 只記錄統計資訊的不足。Color layout descriptor 指定 YCbCr 為 color model。將每一幅畫面都分成 8x8 共 64 個區塊，這與 JPEG 在進行壓縮時的作法不同。算出每一個區塊內的平均顏色後，我們就有 64 個 YCbCr 的顏色。再分別針對 Y，Cb，Cr 的值，進行一次 8x8 的 2D discrete cosine transform。再經 quantization 跟 zigzag scan，我們就得到 Y，Cb，Cr 各 64 個係數。這時再依指定的長度選取係數儲存。在 MPEG-7 中，預設的長度是 Y 為 6，C 則為 3。Y 跟 C 可供選擇的長度有 1、3、6、10、15、21、28、64。

Color layout descriptor 的例子如下，其中 numOfYCoeffIndex 指定的是 Y 係數的個數，numOfCCoeffIndex 指定的是 Cb 與 Cr 係數的個數：

<ColorLayout numOfYCoeffIndex=3 numOfCCoeffIndex=3>

<YDCCoeff>215</YDCCoeff>

<YACCoeff>103 19</YACCoeff>

<CbDCCoeff>243</CbDCCoeff>

<CbACCoeff>201 154</CbACCoeff>

<CrDCCoeff>99</CrDCCoeff>

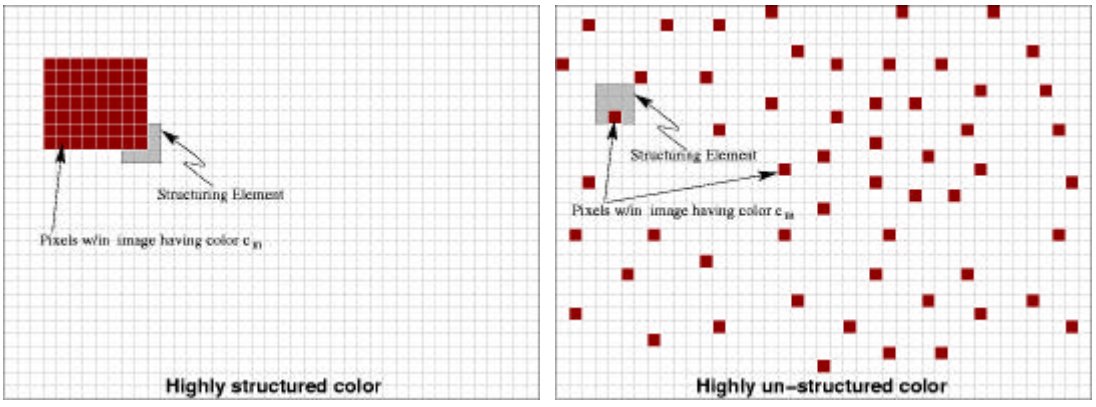
<CrACCoeff>189 78</CrACCoeff>

</ColorLayout>

Color Structure Descriptor

Color structure descriptor 和呈現顏色在空間上關係的 color layout descriptor 不一樣，它也是只記錄顏色出現的次數，只是跟 scalable color descriptor 不一樣的地方在統計的方式不同，希望能透過統計方式的不一樣，提供顏色在空間上的資訊。Color structure descriptor 利用 structure element 來統計，利用 sub-sampling 的技術，一個 structure element 只統計 64 個點。而在 structure element 中只記錄顏色有無出現。等到最後，再統計所有顏色出現過幾個 structure element 中。在 MEPG-7 中，color structure descriptor 指定用 HMMD color model，預設的 color bin 為 32 色，並提供 64，128，256 三種選項。

Color structure descriptor-與 scalable color descriptor 的差異如下圖 2.5，針對深色的點而言，scalable color 記錄的是一樣的值，而在 color structure 的記錄中，右圖（highly un-structured color）的值會比左圖（highly structured color）來的高。



(a) (b)

圖 2.5 Highly structured color 與 Highly un-structured color [38]

Color structure descriptor 的例子如下，其中 value length=32 代表這個 color structure

一共有 32 個 color bins。

<ColorStructure>

<value length=32>

10 32 64 22 98 243 50 69 33 25 89 103 9 11 200 254

153 76 38 19 15 213 177 203 123 54 219 84 67 20 25 24

</value>

</ColorStructure>

GoF/GoP Color Descriptor

GoF/GoP color descriptor 是從 scalable color descriptor 繼承下來的，它記錄的是一段 video segment 的 color histogram。而從影片中取樣的方法有三種：

1. Average：把所有 frame 的 color histogram 取平均值。
2. Median：把所有 frame 的 color histogram 的值取中位數。
3. Intersection：把所有 frame 的 color histogram 的值取最小值。

GoF/GoP color descriptor 的例子如下，其中 length=16 代表這個 GoF/GoP color 一共有 16 個 color bins。：

<GoFGoPColor aggregation="Average">

<value length=16>

153 76 38 19 15 213 177 203 123 54 219 84 67 20 25 24

</value>

</GoFGoPColor>

2.3 資料探勘

「資料探勘」（data mining）在研究從大量資料中整理出有用的資訊。常見的 data

mining 技術有 association rule，sequential pattern，clustering，以及 classification [17]。在這裡，我們介紹 data mining 領域常見的技術，包括 association rule，sequential pattern 與 classification。

2.3.1 Association Rules

Association rule 是從顧客交易資料(transaction data)中，找出購買商品(itemset)間關連的行為。例如買奶粉且買尿布的顧客中有 40%會買奶瓶。而"k-itemset"是指 item 個數為 k 的 itemset。support 是指 itemset 出現的百分比。此外使用者需指定 *minimum support* (min_sup)，而 association rule 會找出 support 超過 min_sup 的 pattern，稱為 *freqnent pattern*。如果我們將一幅繪畫影像視為一筆 transaction，而這幅繪畫影像所萃取出來的低階影像特徵，如果以三個主要顏色表示，這三種顏色就是這個 transaction 的 *itemset*，每一個顏色，就是一個 item。

讓我們以最著名的關聯規則勘測方法 Apriori 演算法探勘銷售資料庫的資料來說明 association rule 的探勘。Apriori 演算法，是由 Agrawal 與 Srikant 於 1994 年所設計的[1]。

【定義 2.1】令 $I = \{i_1, i_2, \cdots, i_m\}$ 是 literals 的集合，每一個都是一筆交易（transaction）中所購買的物品（item）。令 D 是一組交易的集合，而每一筆交易 T 都是一組物品的集合(itemset)，因此 T 是 I 的子集。令 X 是一個物品的集合。而我們稱 T 中包含 X 時，若且唯若 X 是 T 的子集。

【定義 2.2】在一個 n 筆 transaction T_i 的 database D 中，$D=\{T_1, T_2, \ldots, T_n\}$。 Itemset X 的 support 值定義為在 D 中 T_i 包含 X 的個數，記為 support(X)。

【定義 2.3】給一 itemset X，若 X 的 support 大於給定的 minimum support，且 X 的元素個數為 k，則我們稱 itemset X 為 frequent itemset，亦稱為 frequent k-itemset。

【定義 2.4】若 frequent itemset X，不存在另一 frequent itemset Y，使得 $X \subset Y$ ，則稱 X 為 maximum frequent itemset。

【定義 2.5**】一個** association rule **可以表示成下面這種形式，**X→Y**，其中** X **是** I **的子集，**Y **是** I **的子集，而且** X∩Y = ψ**。如果在** D **的所有交易中，在包含** X **的條件下，有** *c*%**的交易也包含** Y **時，則稱** rule X→Y **有** confidence *c***。如果在** D **的所有交易中，有** *s* %**的交易包含** X∪Y **時，則稱** rule X→Y **有** support *s***。**

要找出 association rule **包含了下面兩個步驟：**

1. **找出** frequent itemsets**。所謂的** frequent itemset **就是所有他的子集合** support **都超過了預先設定的門檻，**minimum supports**。**

2. **利用** frequent itemsets **來產生這個** database **的** association rule**。**

表 2.1 Database D

TID	Items
100	A C E
200	B C D
300	A B C D
400	B D

例如 transaction database D**如表** 2.1**。如果** minimum support **設定為** 5%**，也就是** 2 **筆** transaction**。要找出** frequent itemsets **的步驟如下：**

1. **先** scan **整個** database D**，找出所有** 1 **個** item **的** itemset **及其** support**。如圖** 2.6 **中的** C_1**。去掉** support **低於** minimum support **的** itemset **後，得到** frequent itemsets L_1**。把** L_1 **中的** itemsets **組合成** 2 **個** item **的** candidate itemsets**，再** scan **整個** database D**，找出他們的** supports**。去掉** support **低於** minimum support **的** itemsets **後，得到** frequent itemsets L_2**。**

2. **同樣的，把** L_2 **中的** itemsets **組合成個數為** 3 **的** candidate itemsets**。由於雖然**{A, C}**及**{B, C}**超過了** minimum support**，可是因為**{A, B}**並未超過** minimum support**，就組不出來**{A, B, C}**，因此個數為** 3 **的** itemset **只有一組**{B, C, D}**。同樣的，再** scan **整個** database D**，找出**{B, C, D}**的** supports**。沒有低於** minimum support**。因此**

frequent itemset L_3 **就只有**{B, C, D}。

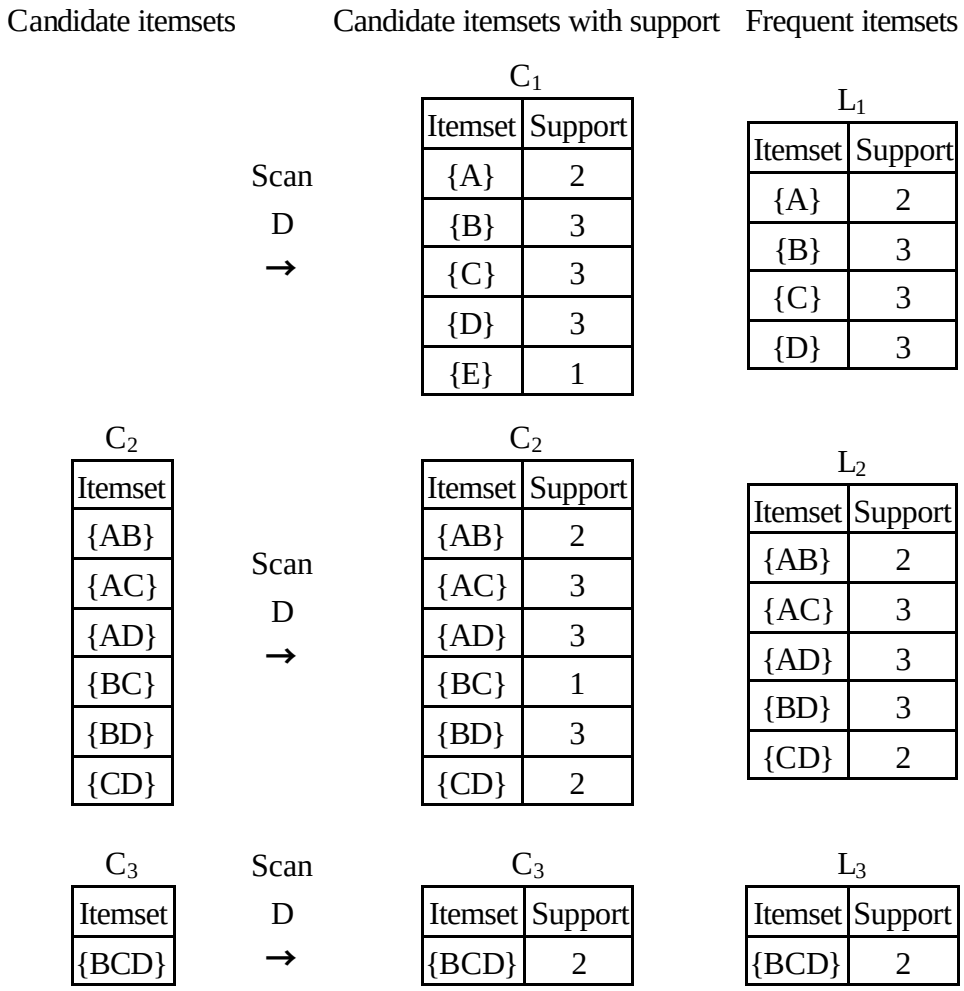


圖 2.6 Generation of candidate itemsets and frequent itemsets

因為 frequent itemset L_3 **就只有一組** itemset**，組合不出個數為** 4 **的** candidate itemset**。找出** frequent itemsets **的步驟就結束了。**

目前 association rule **的研究，除了這種基本的** association rule **以外，還有其他種類** association rule **的變形，以下將介紹** association rule **的一種變形，** multi-level association rule[18, 19]。

一般的商品通常會形成商品階層的關係，如圖 2.7 **的例子顯示，一般** mining

association rule 的方法通常只考慮同一階層之間的關係，但是，以多層的架構來 mining association rule 的話，可能會從資料中發現一些較特殊或具體關係出來，如果某部份的人喜歡買{Milk, Bread}，另外一部份的人喜歡{Soda, Bread}的組合，這時用原始的方法 mining 的話，如果這兩種 case 都沒有超過 min_sup，則 mining 不出什麼關係來，但加入 multi-level 的關念後，即可發現＜Drink＞→＜Bread＞關係，接著我們來看看如何找出這種多層商品間的關係。

Multi-level association rule 分為 uniform support 與 reduced support 兩種做法，通常愈低層的 item，support 值愈低，因此這裡我們介紹 reduced multi-level association rule。

假設商品有{A, B, C, D}四類，A 類有{A1, A2}，B 類有{B1, B2}，C 類有{C1, C2}，D 類有{D1, D2}。如圖 2.8 所示。

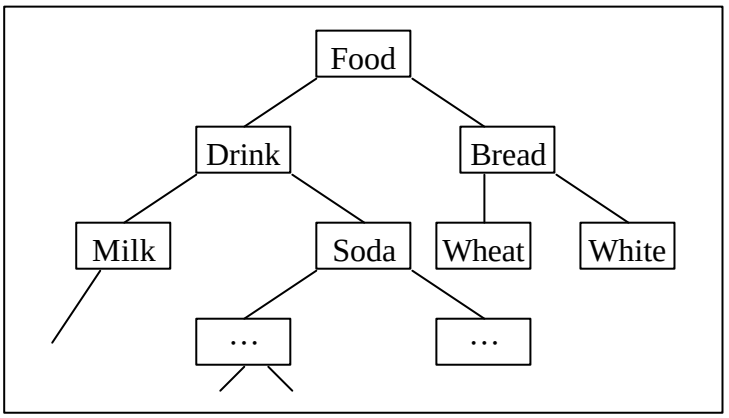


圖 2.7 商品分類架構

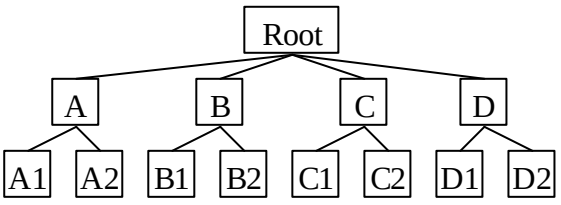


圖 2.8 分類樹

表 2.2 Database D2

TID	Items
T1	{B1,C1}
T2	{A2,B1,B2,C1,D2}

T3	{A1,B2,C2}
T4	{A2,C2}

以原本的 association rule 定 min_sup 為 50%，對表 2.2 的 Database D_2 mining 的話，將會發現{B1,C1}是 frequent itemset。然而，對 D 使用 reduced multi-level association rule，定 level-1 min_sup 75 % ，level-2 min_sup 50%的話，將會發現{A, C}、{B, C}及{B1, C1}是 frequent itemset，而其中{A, C}是原本的 association rule 所找不出來的 frequent itemset。其過程如圖 2.9 所示。

2.3.2 Sequential Patterns

探勘 sequential pattern 目的是要找出所有在同一個顧客的交易間（inter-transaction），所銷售的商品之循序性關係（sequential relationship）。也就是在同一顧客現有的各個交易中，某些商品的出現，意味著其他有循序關係的商品會在後續的交易中出現。假設探勘的結果找到某個 sequential pattern 為 “＜a, c, d＞→＜b, e ＞ [support=30%]”。這個 rule 代表資料庫中有 30%的顧客如果同時購買 a, c, d 後，會在一段時間後再同時購買 b, e。透過探勘得到的 sequential pattern，我們可以預測後續的銷售狀況或者對顧客做較合適的目標行銷。

因為在尋找 sequential pattern 的過程中，同時也必須找出所有具關聯規則的 itemset，因此包括了關聯規則的探勘，以及 itemset 間排列組合的順序關係。為了找出有興趣或者是具有足夠代表性的 pattern，使用者通常會指定 minimum support 值。探勘的結果則是列出所有其 support 值大於或等於 minimum support 的 pattern。這些 pattern 即為 frequent sequential pattern。

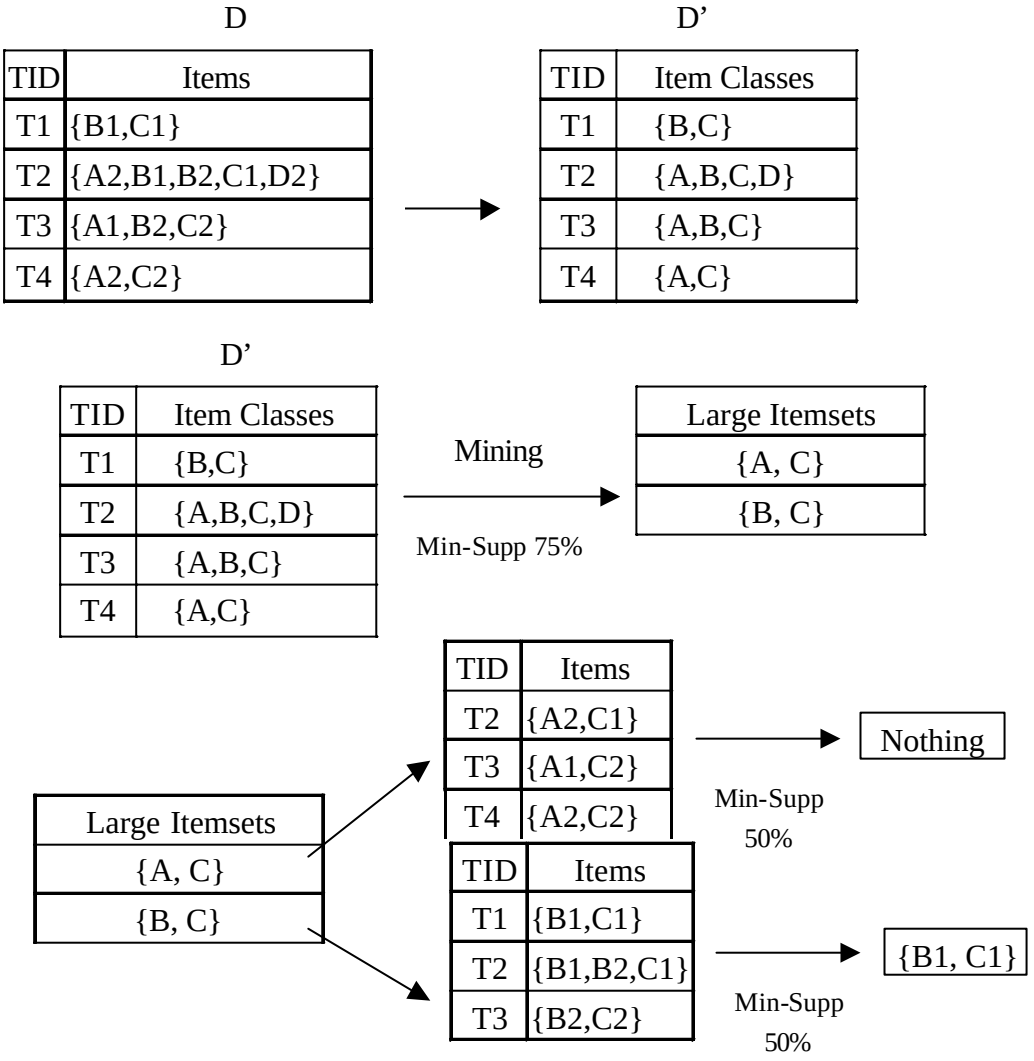


圖 2.9 Multi-Level Association Rule

一個 pattern **的** support **值是資料庫中，「包含（**contain**）」此** pattern **的** sequence **個數除以資料庫中的** sequence **總個數。所謂「包含」，舉例而言，**itemset (a, c, e)**後有** itemset (b, d)**的** sequence **（以＜**(a, c, e) (b, d)**＞表示），包含於** sequence＜(e, f) (a, c, d, e) (f) (b, d)**＞（表示** sequence **中** itemset **的出現次序是**(e, f)**，再來是**(a, c, d, e)**，然後是**(f)**，最後是**(b, d)**）。**

sequential pattern **的探勘，因為以下四個主要因素，使得探勘的難度相當地高。第**

一，序列的元素不限於單一項目，而是 itemset。第二，itemset 的組成是任一可能項目的組合，而且 sequence 的組成是任何 itemset 的可能排列。第三，組成 sequence 的 itemset 其順序不一定要連續。例如 sequence ＜(a, c, d) (b, e)＞，不僅包含於 sequence ＜(b) (a, c, d) (b, e)＞，也包含於 sequence ＜(a, c, d) (f) (b, c, e) (c)＞。第四，探勘前並不知道最長的 sequence 長度為何。

由於在探勘前無法知道哪些項目、itemset 或 sequence 是 frequent 的。因此，大多數的探勘方法是先產生可能的 sequential pattern，稱為 candidate sequence，再對資料庫中所有的 sequence 檢測，以計算各個 candidate 實際的 support 值，最後決定出合格的 sequential pattern。產生 candidate 的方式，一般而言會從上一回合的 frequent sequence 做 join 運算以延伸出下一級的 candidate sequence。目前的探勘方法大多著重於如何減少 candidate 的個數，或是想辦法減少檢視 sequence 包含哪些 candidate 所需之資料庫存取次數。每當使用者指定 minimum support 後，這些方法就開始執行其演算法以找出相對應的 sequential pattern。典型的 sequential pattern 探勘演算法如下圖 2.10 所示。

Algorithm *Mining Sequential Pattern*
Input: Transaction database *DB,* min. support *min_sup*
Output: *Frequent sequential Pattern*
1. All *1*-string are candidate *1*-sequences
2. *k = 1*
3. While the set of candidate *k*-sequences is not empty do
4. For each sequence *ds* in *DB* do
5. For each candidate *c* do
6. Increase the support of *c* if *c* is contained in *ds*
7. Endfor
8. Endfor
9. Frequent *k*-sequences = candidate *k*-sequences whose support ≥ *min_sup*
10. Generate candidate (*k+1*)- sequences from frequent *k*-sequences
11. k = k+1
12. Endwhile

圖 2.10 sequential pattern 探勘演算法圖 [2]

2.3.3 分類演算法

當資料庫中有大量資料需要分類時，可是分類的依據不明顯，我們可以用人工先分類一小部分的資料，再利用 classification 演算法，找出分類的規則，再對剩下的資料進行分類。

2.3.3.1 C4.5 Classification

例如現在有一個顧客資料表，如表 2.3，那麼我們可以根據其顧客購買電腦與否為分類，找出其分類的規則如下：

- 如果 age 是"<=30"且 student 是"no"，則 buys_computer 為"no"。
- 如果 age 是"<=30"且 student 是"yes"，則 buys_computer 為"yes"。
- 如果 age 是"31…40"，則 buys_computer 為"yes"。
- 如果 age 是">40"且 credit_rating 是" excellent"，則 buys_computer 為"no"。
- 如果 age 是">40"且 credit_rating 是" fair"，則 buys_computer 為"yes"。

C4.5 classification 的做法計算每個 attribute 的 information gain，並表示成決策樹[29]。

2.3.3.2 Associative Classification

由 B. Liu 等人提出的 associative classification 是結合 association rule 與 classification 的分類方法，分類的依據是從 relational database 中各類別間的 attribute 所探勘出來的 associative rules，再由 associative rules 建立分類器[24]。associative classification 共有二步驟，一、探勘 associative rules，二、建立分類器。而在探勘 associative rules 之前，先要將 relational database 中的資料進行轉換，轉換的格式為（attribute, integer-value），其中 attribute 是 relational database 的該欄位名稱，integer-value 則是該欄位的值所對應的代碼，以 Student 這一欄為例，(Student, 1)代表具學生身份，而(Student, 2)則是不具有學生身份。以表 3 的第一筆資料為例，轉換後的型式如下：＜(Age, 1) (Income, 3) (Student, 2) (Credit_rating. 2)＞。

表 2.3 Customer profile database

RecordID	Age	Income	Student	Credit_rating	Class: buys_computer
1	<=30	High	No	Fair	No
2	<=30	High	No	Excellent	No
3	30…40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31…40	Low	Yes	Excellent	Yes
8	<=30	Medium	No	Fair	No
9	<=30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<=30	Medium	Yes	Excellent	Yes
12	31…40	Medium	No	Excellent	Yes
13	31…40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

利用以上方法將 relational database 中的資料轉成 transactional database 後，就可以利用 Apriori 演算法分別找出不買電腦類別與會買電腦類別的 frequent itemsets。再將找出的 frequent itemsets 加上 class label 後合併起來，表示成 associative rules，例如：＜{(Age, 1) (Student, 2)}, (class, 不買電腦)＞，表示凡是出現itemset {(Age, 1) (Student, 2)}的 transaction，都會被分類成不買電腦那類。如果某一個 itemset 在不買電腦，買電腦兩邊都有出現，那麼只留下 support 大的一邊。然後再依下列公式計算其 confidence：

$$confidence = MAX\{support\_X, support\_Y\} / (support\_X + support\_Y)$$

例如＜{(Age, 1) (Student, 2)}, (class, 不買電腦)＞這 rule 只屬於不買電腦類，其 support 為 3，confidence 為 MAX{3, 0}/(3+0) = 100%

得出 confidence 後，再將合併完的 associative rules 依 confidence 及 support 的順序

排序，從高排到低，接著進行建立分類器的步驟。依排序過的 associative rules 順序，對 training set 的每一筆資料進行分類，直到每筆 training set 的每一筆資料都被分類正確為止。最後將那些進行比對且分類正確的 associative rules 組合成分類器。