# RS -- Associate Rule Mining（推薦系統 — 關聯規則探勘）

> Reference：
>
> 1. Recommendation System -- Associate Rule Mining by Dr. Tun-Wen Pai
> 2. 矩陣分解(Matrix Factorization): 交替最小平方法(Alternating least squares, ALS)和加權交替最小平方法(Alternating-least-squares with weighted-λ -regularization, ALS-WR)
> 3. Day 07：初探推薦系統(Recommendation System)

## 推薦系統

啤酒對應到尿布是一個著名的關聯規則探勘例子，透過大量的銷售資料集探勘出了啤酒跟尿布同時購買的頻率很高。

在這份筆記，我們最主要會去探討要怎麼從大量的資料找到頻繁樣式，也就是找出哪種商品的購買組合頻率最高。

從大數據來說，找到東西的關聯是非常重要的。

推薦系統支持了我們找出頻繁樣式，進而從大量的資料分析出特定的組合，就能夠找出推薦的組合，稱為推薦系統。

主要有以下幾種不同的工具可以幫我們建立推薦系統。

- Association Rule Mining（ARM），關聯規則探勘。
  - 我們可以用 Frequent pattern（頻繁樣式）來找出資料集的發生頻率。
- Alternating Least Square（ALS），交替最小平方法。
  - 對一個隱含資料（Implicit Data）使用協同過濾演算法，例如 Matrix Factorization 來建立關聯。
- Content Based Filtering
  - 以內容為基礎的過濾，比較商品的屬性，找出最相似的商品。

## 關聯規則探勘

關聯規則探勘 (Association Rule Mining)，一種非監督式學習技巧，

利用大量的資料所產生的資料集來找出頻繁樣式、關聯、相關性、因果結構等等。

其中，Association Rule Mining 使用的是類別資料（Categorical Variable）。

若我們有 $N$ 個物品 $I_i$ 組成的物品集 $I$，對於一個輸入 Association Rule Mining 的資料集，

每筆資料 $T$ 都是資料集的子集合，稱為物品集，也就是 $T \subseteq I$。

例如我們有 6 個物品所組成的 $\{A, B, C, D, E, F\}$，我們有一個資料集：

| ID | Item Set |
|---|---|
| 1 | A, B, C |
| 2 | B, D, E |
| 3 | D, E, F |
| 4 | A, C, E |
| 5 | B, E, F |

我們就能夠從這個資料集找出頻繁樣式。

## *Support, Confidence and Lift*

從大量的資料集，我們希望能夠量化出現頻率，來找到頻繁樣式，所以在 ARM 中將每個組合量化出了三種數值。

對於先發生事件 $A$ 再發生事件 $C$，我們可以量化出三種數值：

- Support：$P(A \cap C)$，量化這個組合發生的機率。
- Confidence：$P(C|A)$，量化這個組合發生 A 再發生 C 的機率。
- Lift：$P(C|A)/P(C)$，用於量化這個組合的效力，通常大於 1 就代表這個組合是有效的。

那麼使用者可以設定一個門檻，用來採納這個頻繁組合該不該適用。

若使用者設定 Support > 40%，且 Confidence > 60%，使用者就可以接受這個組合為頻繁組合。

# 先驗演算法

對於一個很大的資料庫來說，我們可能同時會有很多個不同的頻繁組合。

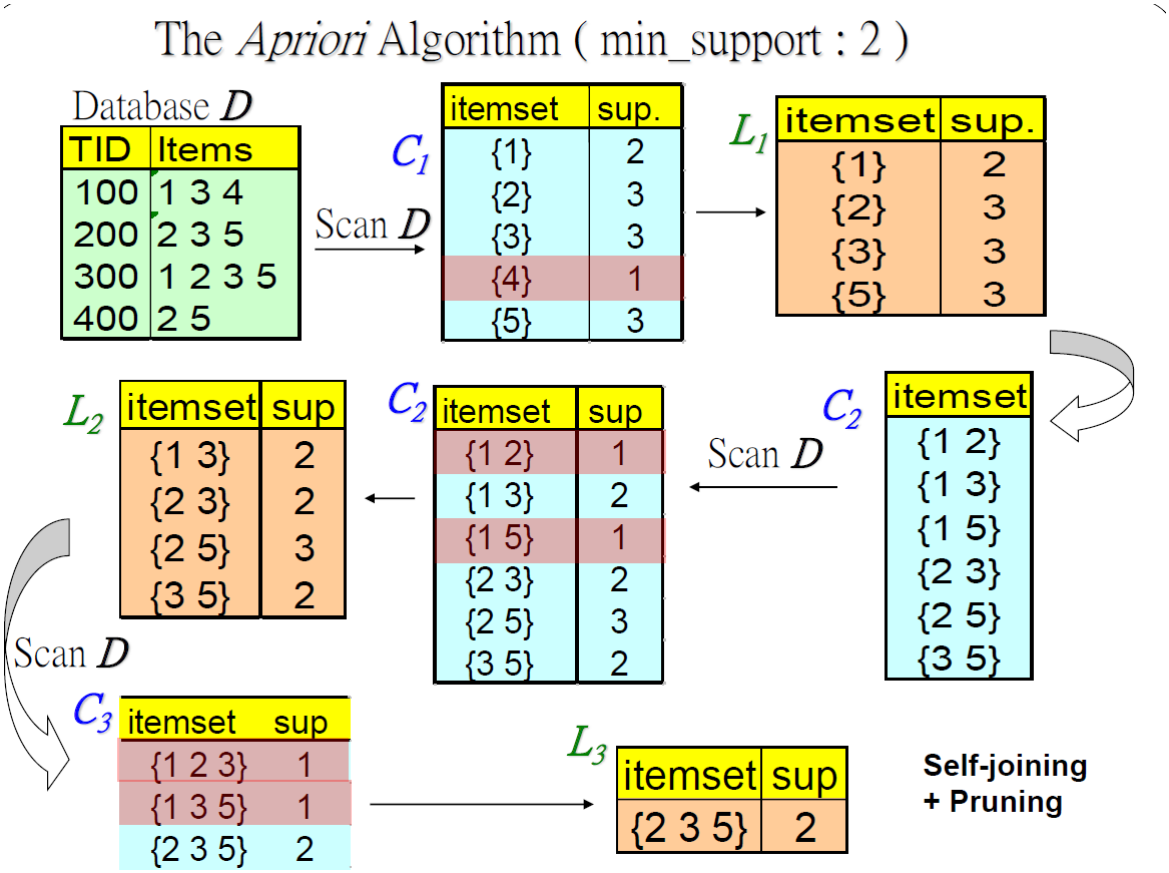
我們希望可以制定頻繁組合必須要至少 Support 大於 k，來淘汰掉一些不太適用的組合。

先驗演算法的一些原則：

1. 一個組合的任意子集都在資料庫內，例如組合為 {a, b, c}，則 {a}, {b}, {c}, {ab}, {ac}, {bc}, {abc} 都應發生在資料庫內。
2. Apriori pruning principle：當我們用任意兩種組合產生出另一個新組合，則這個新組合
   也應該要符合第一點的原則，否則不該產生。
   例如：透過 {a, c, e} 與 {a, e, f} 產生出了 {a, c, e, f} 這個組合，
   但若 {c, e, f} 這個組合並沒有發生在資料庫內，則 {a, c, e, f} 這個組合無效。

下圖為演算法的產生過程。

1. 從原先的資料庫產生出了 1-item-set 的組合，並且刪去只有 support = 1 的組合（因為我們要求 min_support = 2）
2. 從 1-item-set 的任意兩種組合聯集，產生出了 2-item-set 的組合，並且刪去只有 support = 1 的組合
3. 從 i-item-set 產生出了 (i+1)-item-set 的組合...，直到能夠被分到只有一個。




## *先驗演算法的缺點*

對於大資料集來說，**為了產生候選組合，時間與效能耗費過大**。

## FP-Tree

樹狀結構很讚，能夠高效的處理大量資料。

FP-Tree 可以幫助我們建立出頻繁樣式，雖然高壓縮但頻繁樣式完整。

## *frequent item table*

若我們有左邊的資料表，由上至下的 ID 為 1、2、3、4、5。

首先我們應先設定一個 support 為門檻，淘汰掉不需要的資料。

以左邊為例，我們設定 min_support = 3。

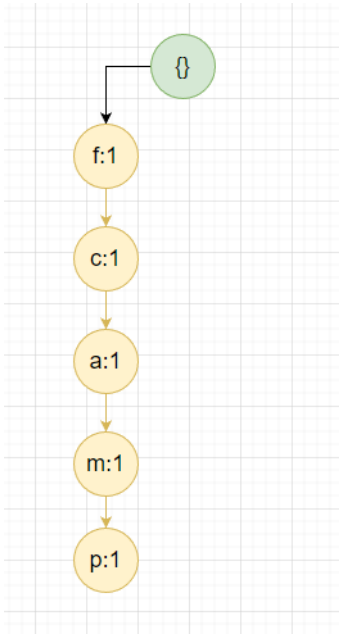
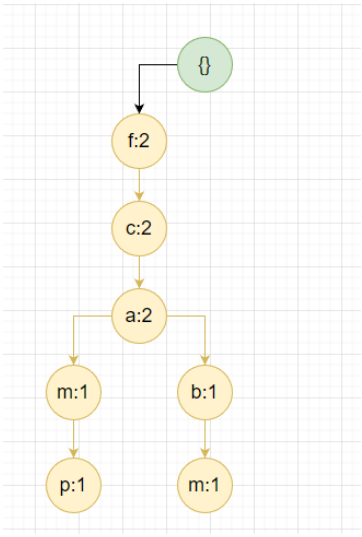
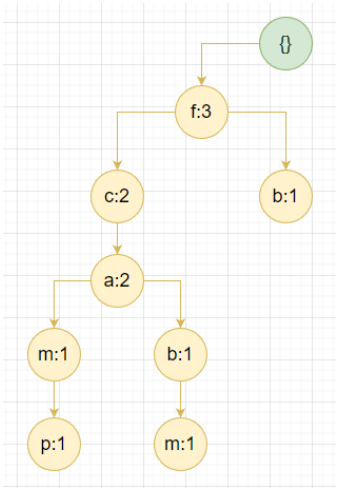
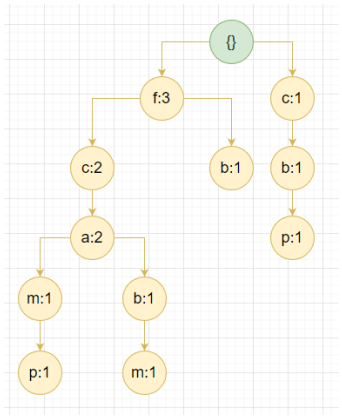
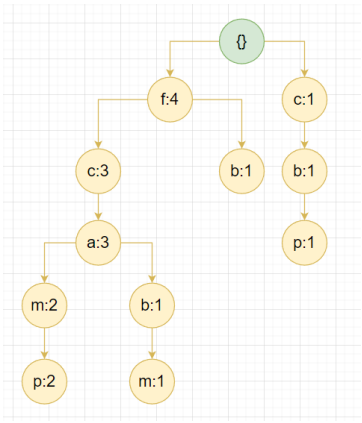
經過淘汰掉不在門檻上的資料，得到右邊的 frequent item table。

| Database | Frequent item table (unordered) |
| --- | --- |
| {f, a, c, d, g, i, m, p} | {f, a, c, m, p} |
| {a, b, c, f, l, m, o} | {a, b, c, f, m} |
| {b, f, h, j, o} | {b, f} |
| {b, c, k, s, p} | {b, c, p} |
| {a, f, c, e, l, p, m, n} | {a, c, f, m, p} |

我們需要對這個 table 做排序，根據物品出現的頻率由多至少排序，左圖呈現物品次數所建構的次數表，右圖呈現排序過後的表格。

| Header Table | Frequent item table (ordered) |
| --- | --- |
| f = 4 | {f, c, a, m, p} |
| c = 4 | {f, c, a, b, m} |
| a = 3 | {f, b} |
| b = 3 | {c, b, p} |
| m = 3 | {f, c, a, m, p} |
| p = 3 | |

接著只需要使用 Frequent item table，對於每一項逐一建樹即可。

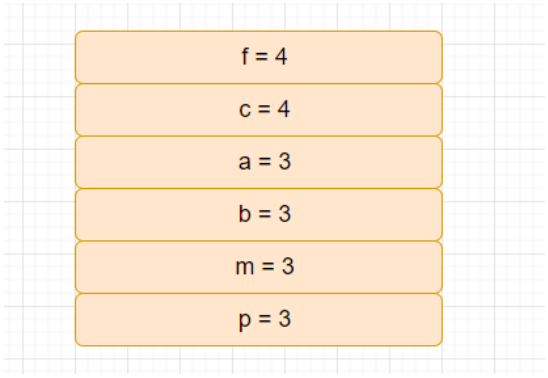
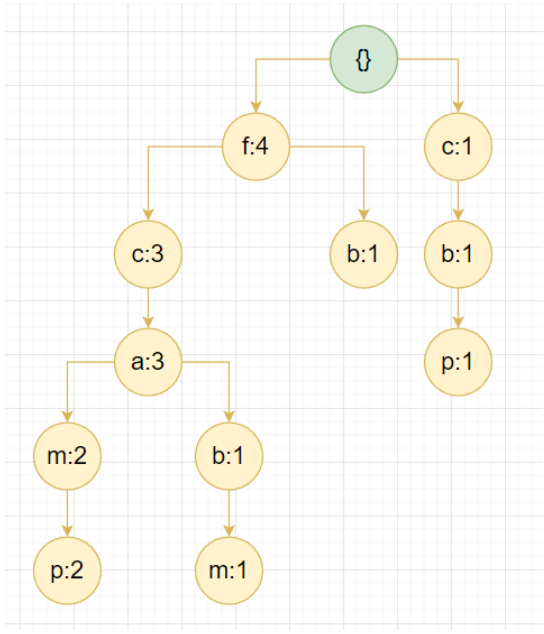
| Data1 | Data2 | Data3 |
| --- | --- | --- |
|  |  |  |
| **Data 4** | **Data 5** | |
|  |  | |

藉此我們就能建構完整的 FP-Tree。

## *conditional pattern base*

對於一棵 FP-Tree，我們可以非常快速的查詢到樣式發生的次數。

以下圖為例，我們可以根據 Header Table 來逐一查詢。

| Header Table | Frequent item table |
| --- | --- |
|  |  |

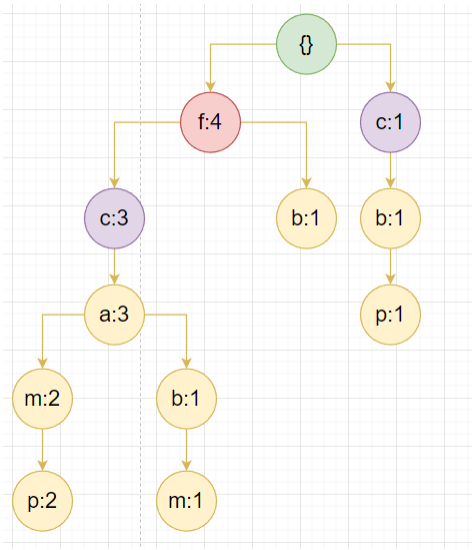
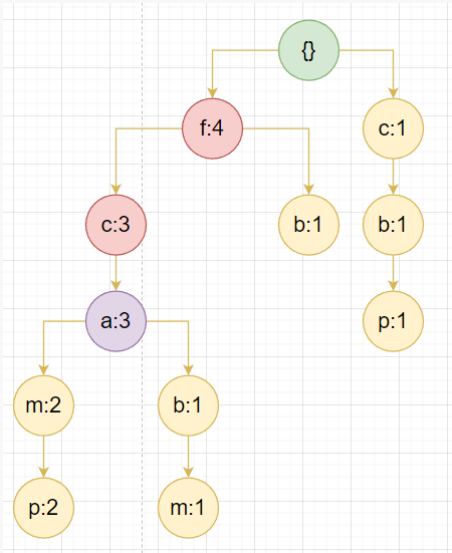
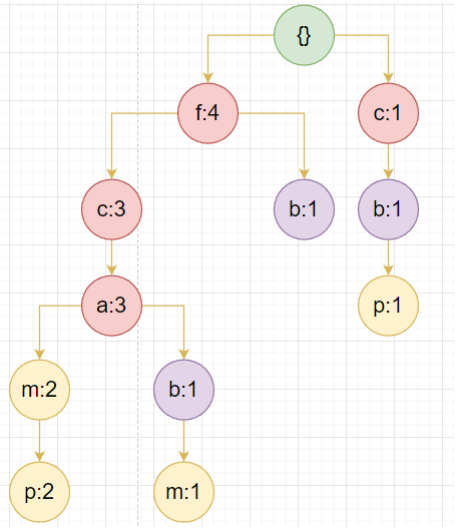


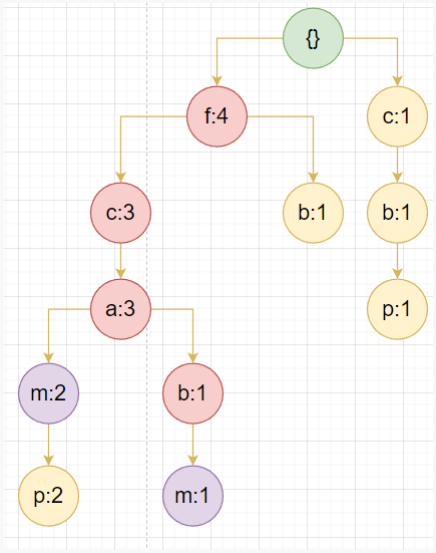
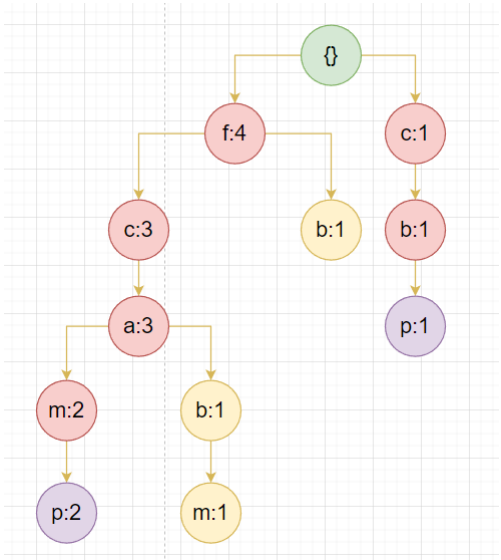


由於具有 BFS 的特性，所以我們可以很快速的查詢到相對的樣式。

我們會從 Header Table 第二項到第六項開始做向上尋訪（紅色節點）即可得到 Conditional Pattern Base

發生次數即為紫色節點的數字。

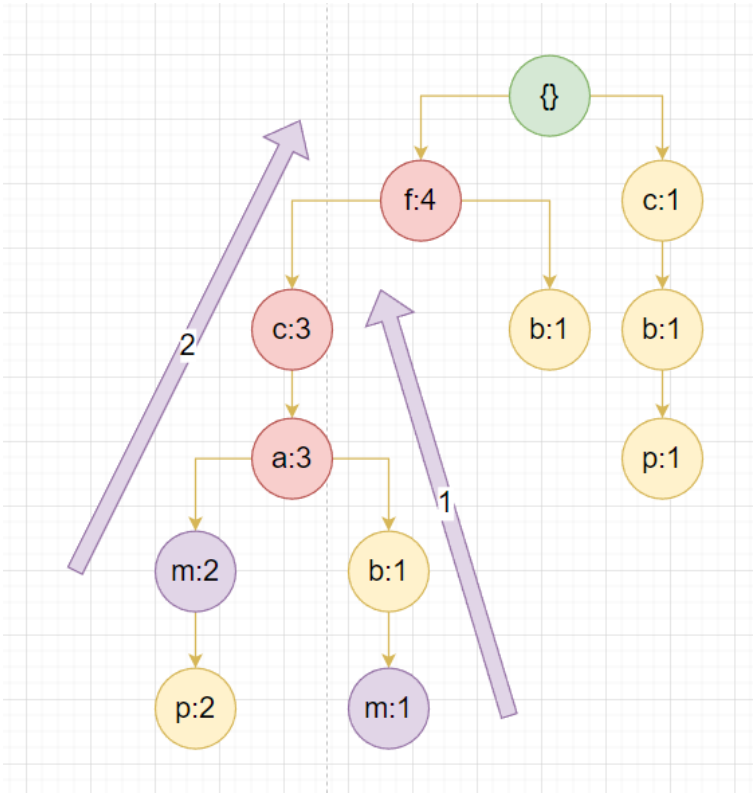
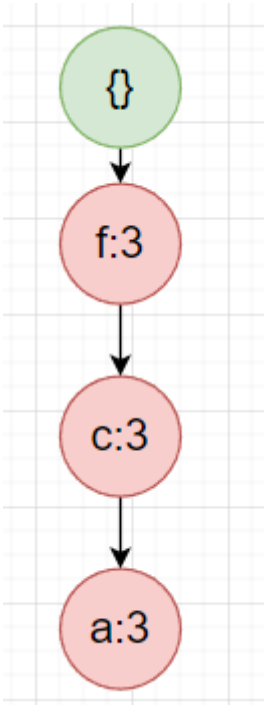
| Item | Image | Conditional Pattern base |
| --- | --- | --- |
| c |  | f:3 |
| a |  | fc:3 |
| b |  | fca: 1<br>f:1<br>c:1 |

| Item | Image | Conditional Pattern base |
| --- | --- | --- |
| m |  | fca:2<br>fcab:1 |
| p |  | fcam:2<br>cb:1 |

## *conditional FP-Tree*

從前面的 conditional pattern base 之後，我們可以根據某一物品 $i$ 的所有 conditional pattern base 所建立出的分支取交集。

以 m 為例，我們已知 `fca:2, fcam:1`，即為兩個不同分支，我們可以將這些分支取交集合併，得到一個新的 conditional FP-Tree。

| FP-Tree when item=m | conditional FP-Tree |
| --- | --- |
|  |  |



我們就能根據這個 conditional FP-Tree 窮舉出所有含有 m 的樣式，也就是`m, fm, cm, am, fcm, fam, cam, fcam`

我們還能根據 conditional FP-Tree 以遞迴的形式找出 am, cm, cam... 的 conditional FP-Tree，與上述同理。

# Collaborative Filtering

我們可以使用 Collaborative Filtering 來找出兩個相似的人或事物，分成以 User-Based 為主與以 Item-Based 為主。

我們會根據使用者或物品的 rating 來評估推薦

但也因此會有 cold start 的問題，也就是新東西或新的人進來之後會不知道要推薦什麼，因為沒有任何的評分紀錄。

要找出相似的人事物，我們需要算出相似度：

- Jaccard similarity measure $\text{sim}(x,y) = \frac{|r_x \cap r_y|}{|r_x| + |r_y| - |r_x \cap r_y|}$，其中 $v_x, v_y$ 為 User rating set。
- Cosine similarity：$\text{sim}(x,y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
- Pearson correlation coefficient：$\text{sim}(x,y) = \frac{\sum_{s \in s_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in s_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in s_{xy}} (r_{ys} - \overline{r_y})^2}}$，$s_{xy}$ 為 x 與 y 對某物的評價。

## User-based collaborative filtering

我們可以使用 user-item rating matrix，找出 user 與 user 之間的關聯性。

只要找出兩個 user 有高相似度，就能把類似的行為推薦給 user。

## Item-based collaborative filtering

我們可以使用 user-item rating matrix，找出 item 與 item 之間的關聯性。

只要找出兩個 item 有高相似度，就能把類似的行為推薦給 user。

## Jaccard similarity measure

第一個提到的相似度計算即為 Jaccard similarity measure。

Jaccard similarity measure 不管評分，只管該 index 有沒有評分，以下面的例子為例。

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| x | 1 | | | 1 | 3 |
| y | 1 | | 2 | 2 | |

我們可以得到 User rating set $r_x = \{1, 4, 5\}$，$r_y = \{1, 3, 4\}$，因為 $x$ 只有在 1、4、5 做評分，$y$ 只有在 1、3、4 做評分。

由於 $r_x$ 與 $r_y$ 有交集的數量只有 $2$，聯集則是 $4$，故 $\text{sim}(x, y) = \dfrac{2}{4} = 0.5$

缺點則是評分上是有意義的，故這樣做會忽略掉評分的意義。

## Cosine similarity

第二個提到的相似度計算即為 Cosine similarity。

以下面的例子為例。

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| x | 1 | | | 1 | 3 |
| y | 1 | | 2 | 2 | |

我們可以將 $r_x, r_y$ 視為點，則可以得到

$$r_x =< 1, 0, 0, 1, 3 >$$
$$r_y =< 1, 0, 2, 2, 0 >$$

故其 Cosine similarity 為 $\dfrac{1+0+0+2+0}{\sqrt{1^2+1^2+3^2}\sqrt{1^2+2^2+2^2}} \approx 30.2\%$

## *Pearson correlation coefficient*

定義上為 $\text{sim}(x,y)=\dfrac{\sum_{s\in s_{xy}}(r_{xs}-\overline{r_x})(r_{ys}-\overline{r_y})}{\sqrt{\sum_{s\in s_{xy}}(r_{xs}-\overline{r_x})^2}\sqrt{\sum_{s\in s_{xy}}(r_{ys}-\overline{r_y})^2}}$，$s_{xy}$ 為 x 與 y 對某物的評價。

雖然定義上是這樣，但，其實你只要算出列/行的平均，行/列存在的元素全部減去平均後砸 Cosine Similarity 也能做出來

~~安子我的算式恐懼症，這麼可怕的式子是啥鬼嗚嗚~~

以下面的例子為例。

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| x | 2 | | | 1 | 3 |
| y | 2 | | 2 | 2 | |

我們可以將 $r_x, r_y$ 視為點，則可以得到

$$r_x =< 1,0,0,1,3 >$$
$$r_y =< 1,0,2,2,0 >$$

平均 $\mu_x=2$，$\mu_y=2$，則

$$r'_x =< -1,0,0,-1,1 >$$
$$r'_y =< -1,0,0,0,0 >$$

我們可以算出 Pearson correlation coefficient

$$\text{sim}(x,y)=\frac{1}{((-1)^2+(-1)^2+1^2)((-1)^2)}=\frac{1}{3}\approx 0.333$$

## *Prediction Function*

若我們有一個物件與其他 $N$ 個物件相似，則我們可以用以下的式子來表達其預測值

$$r_{xi}=\frac{\sum_{j\in N(i;x)} s_{ij}\cdot r_{xj}}{\sum_{j\in N(i;x)} s_{ij}}$$

見以下範例。

## *Item-Item Collaborative Filtering*

假設我們有一筆這樣的資料，欄為 User 且列為 Movie，表格上每個數字為 User 對 Movie 的評價

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | 3 | | ? | 5 | | | 5 | | 4 | |
| 2 | | | 5 | 4 | | | 4 | | | 2 | 1 | 3 |
| 3 | 2 | 4 | | 1 | 2 | | 3 | | 4 | 3 | 5 | |
| 4 | | 2 | 4 | | 5 | | | 4 | | | 2 | |
| 5 | | | 4 | 3 | 4 | 2 | | | | | 2 | |
| 6 | 1 | | 3 | | 3 | | | 2 | | | 4 | |

可以看到 $D(1,5)$ 的資料是問號，即為我們想要推演的資料。

我們使用 Pearson correlation coefficient 來**計算電影的相似度**，先算出每個電影的平均值，以及該列減去平均值的向量。

$\mu_1 = (1+3+5+5+4)/5 = 3.6$，$v_1' =< -2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0 >$

$\mu_2 = (5+4+4+2+1+3)/6 = 3.167$，$v_2' =< 0, 0, 1.83, 0.83, 0, 0, 0.83, 0, 0, -1.17, -2.17, -0.17 >$

$\mu_3 = (2+4+1+2+3+4+3+5)/8 = 3$，$v_3' =< -1, 1, 0, -2, -1, 0, 0, 0, 1, 0, 2, 0 >$

$\mu_4 = (2+4+5+4+2)/5 = 3.4$，$v_4' =< 0, -1.4, 0.6, 0, 1.6, 0, 0, 0.6, 0, 0, -1.4, 0 >$

$\mu_5 = (4+3+4+2+2)/5 = 3$，$v_5' =< 0, 0, 1, 0, 1, -1, 0, 0, 0, 0, -1, 0 >$

$\mu_6 = (1+3+3+2+4)/5 = 2.6$，$v_6' =< -1.6, 0, 0.4, 0, 0.4, 0, 0, -0.6, 0, 0, 1.4. 0 >$

我們可以算出 Pearson correlation coefficient，得到

$\text{sim}(1,2) = -0.17$，$\text{sim}(1,3) = 0.41$，$\text{sim}(1,4) = -0.1$，$\text{sim}(1,5) = -0.36$，$\text{sim}(1,6) = 0.59$

故我們選擇電影 3、6 與電影 1 有較高的關聯性。

因此我們可以套用預測函數，得到：

$$\frac{0.41 \times 2 + 0.59 \times 3}{0.41 + 0.59} = 2.6$$

故 $D(1,5)$ 可以被預測成 2.6 分。

### *User-User Collaborative Filtering*

與 Item-Item Collaborative Filtering 同理，跳過。

## Content-based Filtering

使用內容來進行評估，因為使用內容，所以不會有 cold start 的問題。

# TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency)，用來量化詞彙的重要性，但會隨著詞彙出現的頻率高而出現反比影響。

其定義為：

$$TF_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$$

$$IDF_i = \log(\frac{N}{n_i})$$

$$\text{TF-IDF}_{ij} = TF_{ij} \times IDF_i$$

其中 TF 的 $n_{ij}$ 為該詞彙在文件中的次數，分母則是在所有文件中該詞彙出現的次數，$N$ 為文件的總詞彙量。