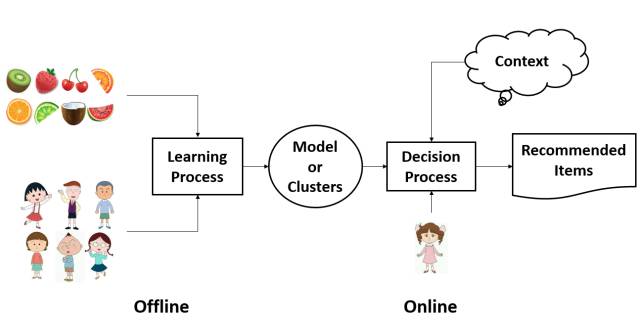
**推薦演算法**

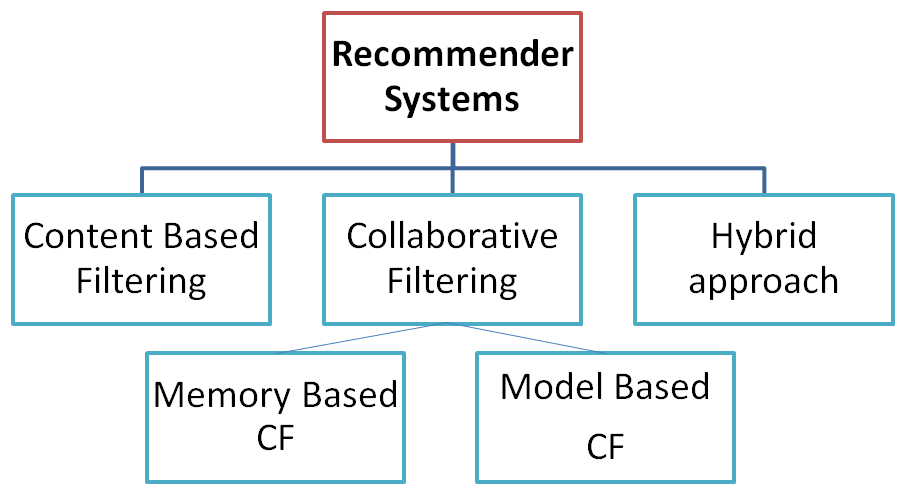
推薦演算法是一種預測系統，預測受推薦的用戶(User)對被推薦內容(Item)的喜好長度或偏好程度。推薦系統利用內容過濾（以你之前流覽過的內容來模擬你的偏好），或是協同過濾（根據你的造訪紀錄或同好），來達到個性化推薦的目的。

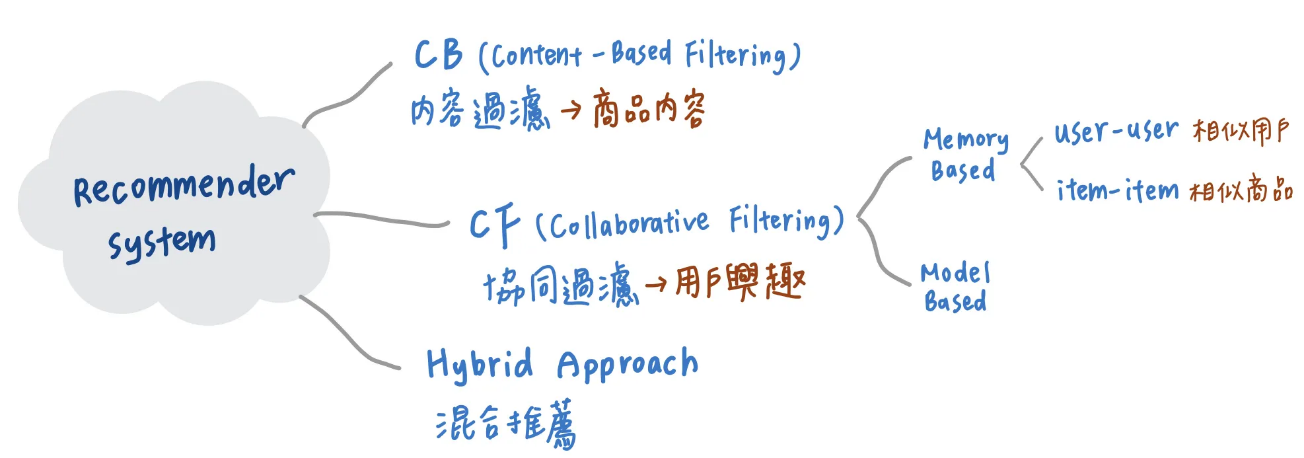
**推薦系統的結構（Structure）**



幾乎所有的推薦系統的結構都是類似的，都由線上和線下兩部分組成。線下部分包括後臺的日誌系統和推薦算法系統，線上部分就是我們看到的前臺頁面展示。線下部分通過學習使用者資料和行為日誌建立模型，在新的上下文背景之下，計算相應的推薦內容，呈現於線上頁面中。

**推薦系統分類：**





**一、以內容為基礎的過濾(Content Based Filtering, CBF)**

比較商品的屬性，找出最相似的商品。而它的前題假設是「如果小明喜歡A，那麼小明應該也會喜歡與A類似的物品」。

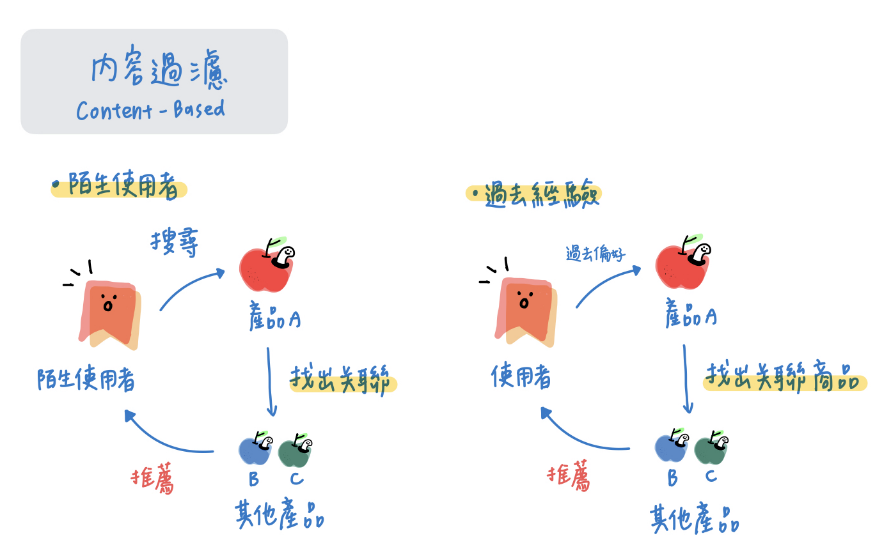
利用物品屬性進行推薦，而物品的屬性可以是音樂的類型或電影的風格等等。此方法算是最早開始使用的，它會根據你過去喜歡的產品，去推薦過去你喜歡的相似產品給你。例如，一個網路書店會根據你之前找很多英文相關書籍就推薦你此類產品。

CBF 演算法 ：

第一步（item representation）：為每一個物品（item）選出其特徵來表示此物品；

第二步（profile learning）：利用使用者過去對一個物品喜好程度的數據來學習該使用者的偏好；

第三步（Recommendation generation）：透過前兩者的比較而得出，並為使用者推薦一組相關性最大的物品。



舉一個例子來說明，每個人對於不同類別文章的喜愛程度都有所不同。今天就以個性化閱讀來說，一篇文章就是一個物品，而在

第一步（item representation）：找出此篇文章的特徵，也就是從文章內容理解它屬於哪個類別。

第二步（profile learning）：根據使用者過去喜歡什麼文章來找出該用戶的偏好，比如某位使用者經常關注泰國旅遊的文章，那系統所找出的用戶偏好中，泰國旅遊就會比其他種類的文章還要多。

第三步（Recommendation generation）：透過第一步和第二步的數據，系統就可以計算兩者的相關度並對使用者推薦文章了，也就是把所有物品中與使用者相關性最高的推薦給他。

**以內容為基礎的過濾又分為兩種：**

* 1. 依據一件瀏覽或已購買的商品，推薦屬性相似的商品。

如果使用者之前看過幾部電影，發現他比較看漫威電影，那就應該推薦他看『復仇者聯盟』，而不是恐怖片

適用時機：沒有使用者回饋的資料，只有商品屬性資料時使用。

缺點：只會推銷屬性相近的商品，不考慮商品暢銷與否，不考慮使用者的偏好。

* 1. 結合使用者評價與商品屬性，推薦使用者偏好的商品。

如果我們有使用者評價資料，那就可以結合商品屬性資料，那就可以作矩陣相乘，產生使用者喜歡的商品屬性，之後，就可以按第一種作法進行相似性計算或進行統計，找出推薦的商品，結果出來後，記得刪除使用者已購買過的商品。

適用時機：有目前使用者的偏好，可結合商品屬性資料時使用。

缺點：若是新顧客，會造成無法推薦，這就是所謂的『冷啟動』(Cold Start)的問題，遇到這種狀況，可以回歸第一種作法。

內容過濾(Content-based systems)有一個缺點，就是系統很難發現使用者不熟悉卻有潛在興趣的商品。就像如果小明每天都聽Rap，系統可能會不停地推薦小明很多好聽的Rap，但是不代表小明不喜歡別種類型的音樂啊！

**二、協同過濾(Collaborative Filtering, CF)**

以「用戶的興趣」為基礎，以團體的行為來做決策。協同就是集合眾人的意見協同合作，進而篩選或推薦商品，作法與購物籃分析類似，一樣是以銷售記錄進行分析，不同的是，並不進行商品組合分析，而是將銷售記錄轉成『使用者/商品對應的矩陣』(User-Item Matrix)，如下圖，記錄哪些使用者買過哪些商品，計算顧客間或商品間的相似度，再推薦相似顧客曾買過的商品，或推薦與目前商品最相似的其他商品，進行 cross selling。

**協同過濾演算法**

第一步：建立使用者－項目矩陣（User-Item Matrix）



矩陣的一個維度是每一個使用者，而另一個維度則是每一個項目，在矩陣中的值，即為某一使用者對另一個項目的偏好程度。

第二步：計算相似度，在這個矩陣中，User b並未對Item 2、User c也並未對Item 3表示過任何的偏好，但是透過User a, b, c對項目偏好的相近程度，系統可以推論出User b對Item 2及User c對Item 3的偏好程度。

第三步：推薦，當系統有能力評估出此值時，便能夠依據該值是否超過某一個門檻，來決定是否將Item 2推薦給User b，以及將Item 3推薦給User c。



**協同過濾演算法分類**

Memory-based以記憶為基礎的協同過濾

Model-based以模型為基礎的協同過濾

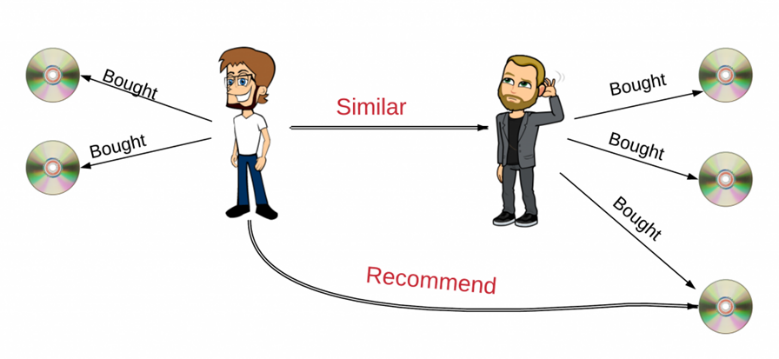
**Memory-based以記憶為基礎的協同過濾又分為**

User-based Collaborative Algorithm以用戶為基礎的協同過濾

Item-based Collaborative Filtering Algorithm 以項目為基礎的協同過濾

**2.1.1 User-based Collaborative Algorithm以用戶為基礎的協同過濾**

或稱為基於鄰居的協同過濾(Neighbor-based Collaborative Filtering)。

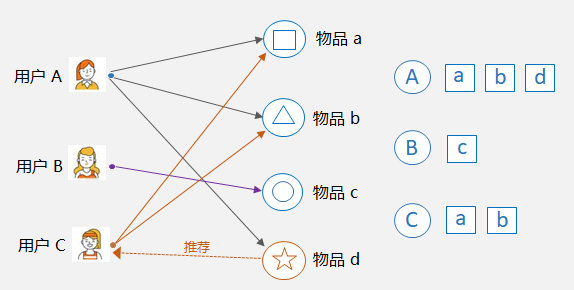


User-based的推薦邏輯是「推薦給相似的人」



計算使用者之間的相似程度，將相似用戶有興趣的商品推薦給使用者 → 需要蒐集使用者資訊，例如評價、購買紀錄等

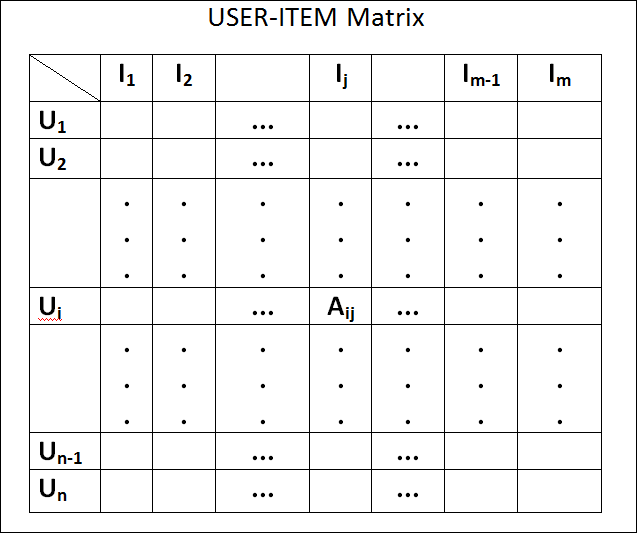
User-CF，核心是找相似的人。比如下圖中，使用者 A 和使用者 C 都購買過物品 a 和物品 b，那麼可以認為 A 和 C 是相似的，因為他們共同喜歡的物品多。這樣，就可以將使用者 A 購買過的物品 d 推薦給使用者 C 。



User-based Collaborative Filtering演算法：

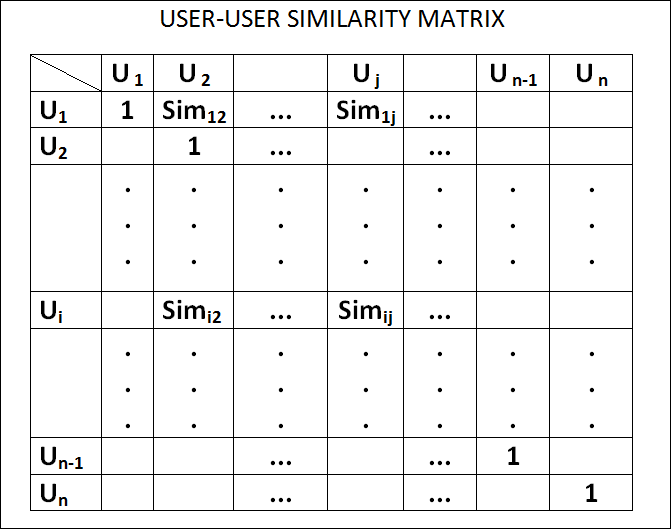
(1.) 收集用戶信息

收集可以代表用戶興趣的信息。一般的網站系統使用評分的方式或是給予評價，這種方式被稱為“主動評分”。另外一種是“被動評分”，是根據用戶的行為模式由系統代替用戶完成評價，不需要用戶直接打分或輸入評價數據。電子商務網站在被動評分的數據獲取上有其優勢，用戶購買的商品記錄是相當有用的數據。

  
圖. User-Item Matrix

(2.) 最近鄰搜尋(Nearest neighbor search, NNS)

以用戶為基礎（User-based）的協同過濾的出發點是與用戶興趣愛好相同的另一組用戶，就是計算兩個用戶的相似度。例如：查找n個和A有相似興趣用戶，把他們對M的評分作為A對M的評分預測。一般會根據數據的不同選擇不同的算法，目前較多使用的相似度算法有Pearson Correlation Coefficient、Cosine-based Similarity、Adjusted Cosine Similarity。

  
圖. USER-USER Similarity Matrix

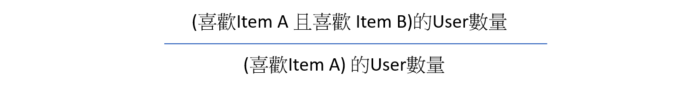
(3.) 產生推薦結果

有了最近鄰集合，就可以對目標用戶的興趣進行預測，產生推薦結果。依據推薦目的的不同進行不同形式的推薦，較常見的推薦結果有Top-N 推薦和關係推薦。Top-N 推薦是針對個體用戶產生，對每個人產生不一樣的結果，例如：通過對A用戶的最近鄰用戶進行統計，選擇出現頻率高且在A用戶的評分項目中不存在的，作為推薦結果。關係推薦是對最近鄰用戶的記錄進行關係規則(association rules)挖掘。

**2.1.2 Item-based Collaborative Filtering Algorithm 以項目為基礎的協同過濾演算法**

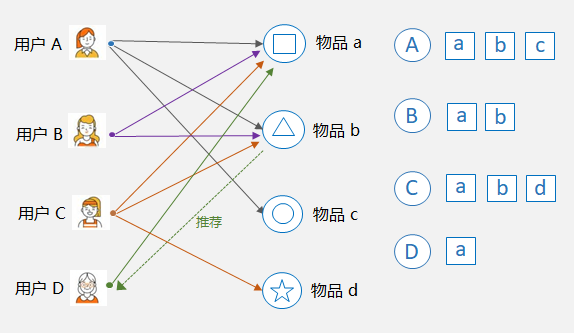
以用戶為基礎的協同推薦算法隨著用戶數量的增多，計算的時間就會變長，所以在2001年Sarwar提出了基於項目的協同過濾推薦算法(Item-based Collaborative Filtering Algorithms)。以項目為基礎的協同過濾方法有一個基本的假設“能夠引起用戶興趣的項目，必定與其之前評分高的項目相似”，通過計算項目之間的相似性來代替用戶之間的相似性。

item-based協同過濾，通過計算物品和物品的相似度找到跟物品1相似的物品2, 3, 4...再把這些物品推薦給看過物品1的使用者們。Item-based的推薦邏輯則是「推薦相似的東西」。



**Item-based**的協同過濾則是指計算物品間的相似程度，也就是喜歡物品A的使用者也喜歡物品B的有多少個，整個過程與前者類似，只不過此種方式不須考慮到使用者的差異，所以精準度的效果也會不太一樣。

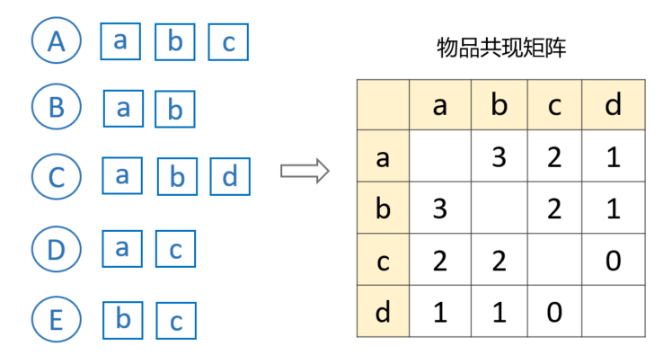
Item-CF，核心是找相似的物品。比如下圖中，物品 a 和物品 b 同時被使用者 A，B，C 購買了，那麼物品 a 和 物品 b 被認為是相似的，因為它們的共現次數很高。這樣，如果使用者 D 購買了物品 a，則可以將和物品 a 最相似的物品 b 推薦給使用者 D。



Item-based Collaborative Filtering Algorithm**演算法：**

**第一步：整理物品的共現矩陣**

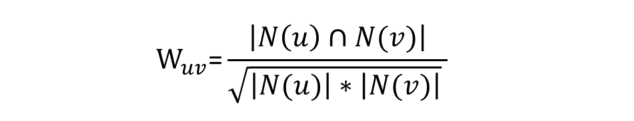
假設有 A、B、C、D、E 5個使用者，其中使用者 A 喜歡物品 a、b、c，使用者 B 喜歡物品 a、b等等。



所謂共現，即：兩個物品被同一個使用者喜歡了。比如物品 a 和 b，由於他們同時被使用者 A、B、C 喜歡，所以 a 和 b 的共現次數是3，採用這種統計方法就可以快速構建出共現矩陣。

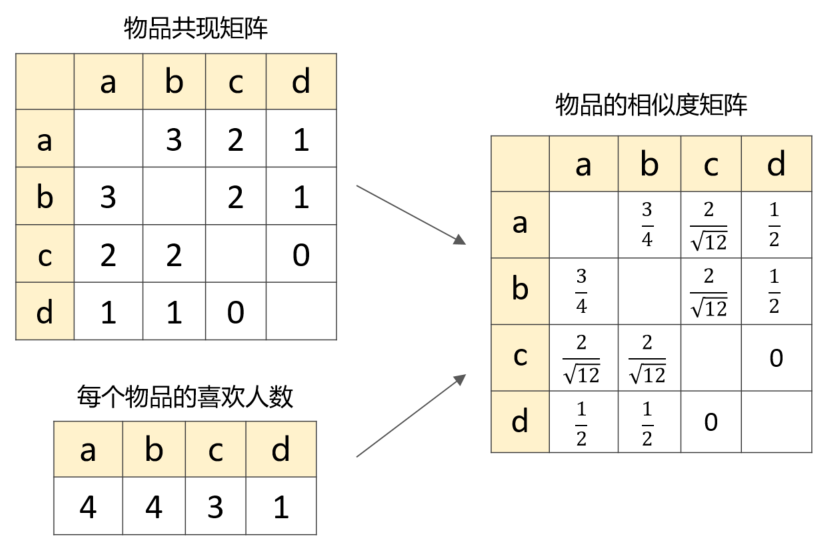
**第二步：計算物品的相似度矩陣**

對於 Item-CF 演算法來說，一般不採用前面提到的餘弦距離來衡量物品的相似度，而是採用下面的公式 ：



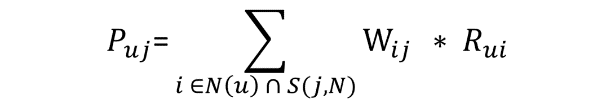
其中，N(u) 表示喜歡物品 u 的使用者數，N(v) 表示喜歡物品 v 的使用者數，兩者的交集表示同時喜歡物品 u 和物品 v 的使用者數。很顯然，如果兩個物品同時被很多人喜歡，那麼這兩個物品越相似。

基於第1步計算出來的共現矩陣以及每個物品的喜歡人數，便可以構造出物品的相似度矩陣：



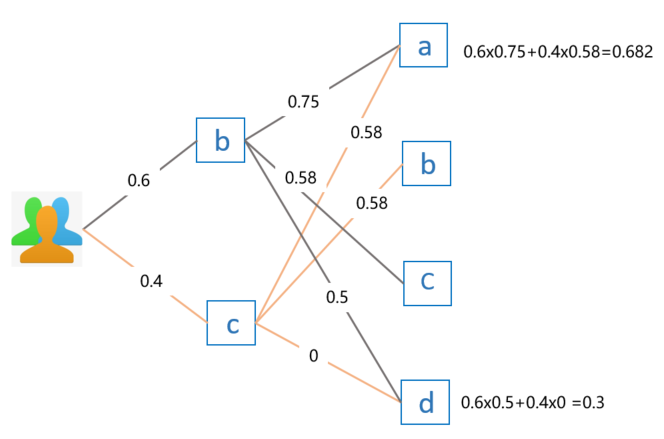
**第三步：推薦物品**

最後一步，便可以基於相似度矩陣推薦物品了，公式如下：



其中，P uj 表示使用者 u 對物品 j 的感興趣程度，值越大，越值得被推薦。N(u) 表示使用者 u 感興趣的物品集合，S(j,N) 表示和物品 j 最相似的前 N 個物品，W ij 表示物品 i 和物品 j 的相似度，R ui 表示使用者 u 對物品 i 的興趣度。

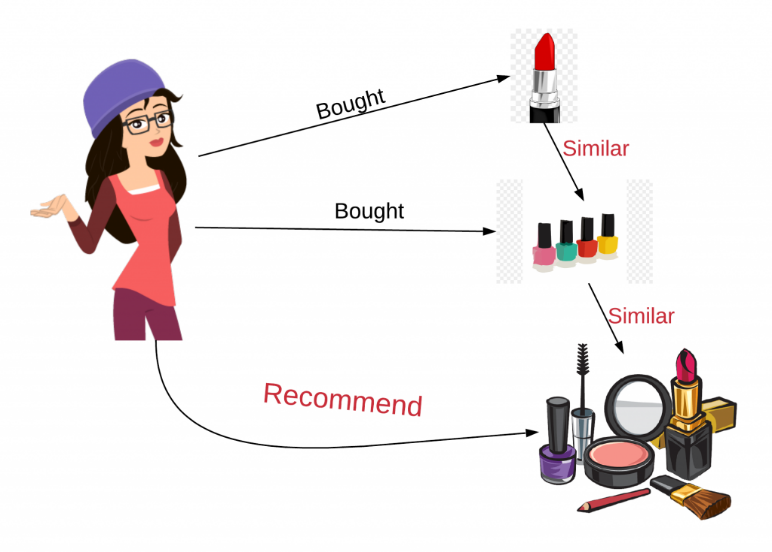
上面的公式有點抽象，直接看例子更容易理解，假設我要給使用者 E 推薦物品，前面我們已經知道使用者 E 喜歡物品 b 和物品 c，喜歡程度假設分別為 0.6 和 0.4。那麼，利用上面的公式計算出來的推薦結果如下：

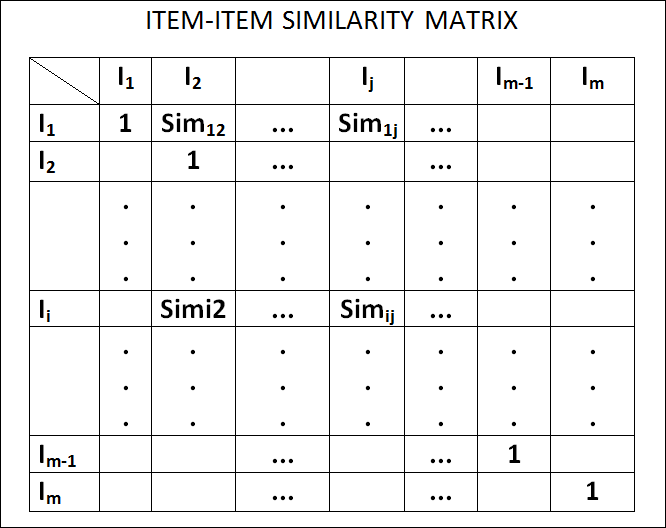


因為物品 b 和物品 c 已經被使用者 E 喜歡過了，所以不再重複推薦。最終對比使用者 E 對物品 a 和物品 d 的 感興 趣程度，因為 0.682 > 0.3，因此選擇推薦物品 a。

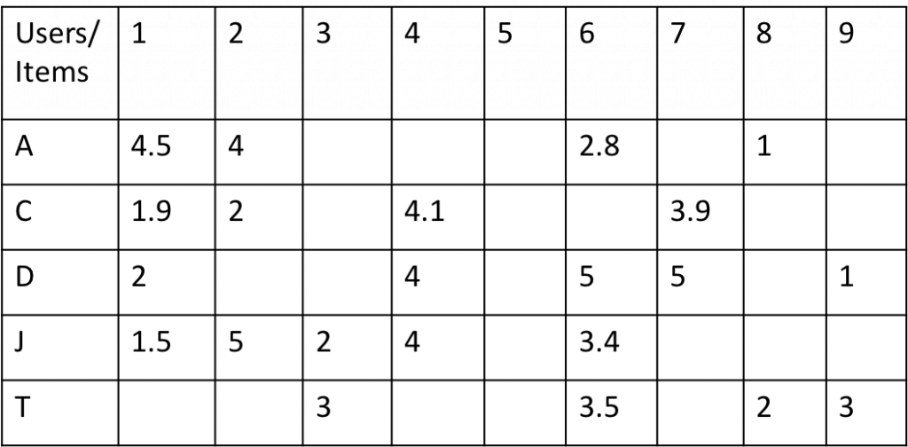
以記憶為基礎的協同過濾技術共同有的缺點就是資料稀疏問題，因為難有較大量的支援資料而影響其分析結果，於是便發展出以模型為基礎的(Model-based )協同過濾，此方法是預先用歷史資料得到一個模型，再透過它做預測。

**Item-Based :商品之間的相似程度計算**

1. ITEM-ITEM 協同過濾：找出與目前瀏覽的商品最相似的商品族群(ITEM-ITEM Similarity Matrix)，推薦給顧客。  
   

  
圖. ITEM-ITEM Similarity Matrix

如何從User-Item Matrix轉換為 USER-USER Similarity Matrix 或 ITEM-ITEM Similarity Matrix，請看以下說明。



User- Based 與 Item-Based 優缺點

1. User-Based的協同過濾系統通常會推薦最熱門(點擊率最高)的商品，然而，最熱賣的商品不見得是消費者最有興趣的。
2. Item-Based 的協同過濾系統比較容易推薦長尾(long-tail) 的商品，所謂長尾商品就是指那些「購買聲量不高，但消費者會持續購買的商品」
3. 以電商相關的部門來說，因為擁有大量的使用者資料(既有客戶、潛在客戶、沉睡客戶等)，如果使用 User-Based 的模式去計算用戶的相似程度，會需要耗費很大量的時間，因此會建議使用 Item-Based 的方法。
4. 對於一些以提供內容為主的網站來說(如：部落格)，當他們在推薦適合的內容給讀者的時候，由於文章會一直更新，就比較適合使用User-Based的協同過濾系統進行推薦。

註：Chris Anderson在 2004年提出**長尾效應**，指出只要規模夠大，非主流、需求量小的商品總銷售量也能夠和主流、需求量大的商品銷量平起平坐。

**協同過濾的優點**：

優點：協同過濾是集合眾人的意見進行推薦，比較有可信度。

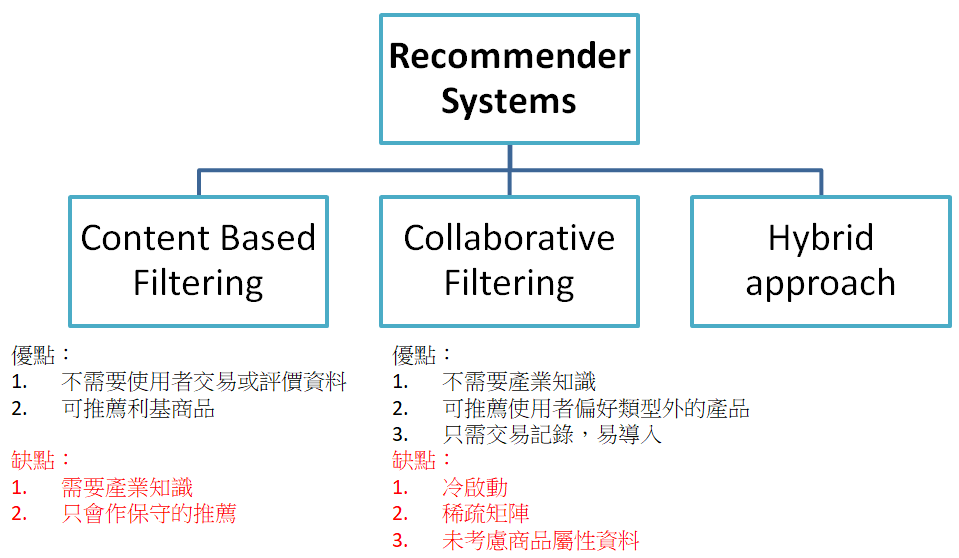
**協同過濾的缺點**，最典型的有兩點：

**冷啟動問題(Cold Start)：**所謂的冷啟動就是說我們沒有足夠的使用者與物品間的訊息而造成推薦系統效果不佳。採用USER-USER 協同過濾法，遇到新顧客無購買或瀏覽記錄，無法計算與其他顧客的相似度。比方說今天有一個新開發的電子商務網站，但因為他才剛成立所以並沒有使用者的資料，所以這就造成推薦系統無法精確的將某商品推薦給消費者，而必須等到有足夠大量的使用者相關資料才能夠有效的做推薦。

ITEM-ITEM 協同過濾法也是一樣，新商品也無交易記錄，無法計算與其他商品的相似度。這時可以考慮使用『Content Based Filtering』。

「推薦系統因為不了解新用戶的喜好而無法進行推薦」、「新項目因為沒有評分記錄而沒辦法被推薦」這兩個令人頭痛的冷啟動問題唷！

**稀疏性問題(Sparsity)：**隨著使用者和商品的數目增長，推薦系統的規模也必須擴大，但這也表示商品與用戶的重疊性降低了，也就是說使用者幾乎是沒辦法把一個網頁上的所有商品都瀏覽過。試想一下像淘寶這種規模非常大的電子商務網站所擁有的商品數（號稱8億），平均來說一位使用者所瀏覽的商品有可能是總商品數的百萬分之一(800件)嗎？幾乎是不可能吧，由此可知其稀疏度應該非常的高了。因此資料規模越大，稀疏性問題就更嚴重，如果能夠有效處理這種問題的演算法就可以說是很有前途的



**2.2. 以模型為基礎（Model- based）的協同過濾**

以用戶為基礎（User-based）的協同過濾和以項目為基礎（Item-based）的協同過濾都是以過去的記憶為基礎，所以統稱為以記憶為基礎（Memory based）的協同過濾技術，他們共有的缺點是數據稀疏，難以處理大數據量影響即時結果，因此發展出以模型為基礎的協同過濾技術。

Model Based 的協同過濾方法則是以模型為基礎，透過歷史資料得到一個模型，再藉由那個模型去預測其他的資料。Model-based Collaborative Filtering是先用歷史數據得到一個模型，再用此模型進行預測。

Model Based 的協同過濾使用各種機器學習、深度學習(Deep Learning, RL)或強化學習(Reinforcement learning, DL)的演算法構築推薦系統。主流的方法可以分為：矩陣分解，關聯演算法，聚類演算法，分類演算法，迴歸演算法，神經網路。

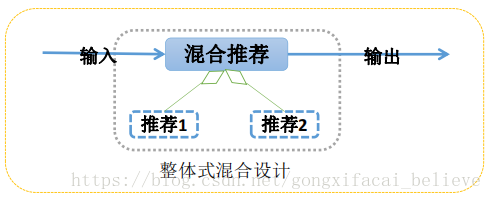
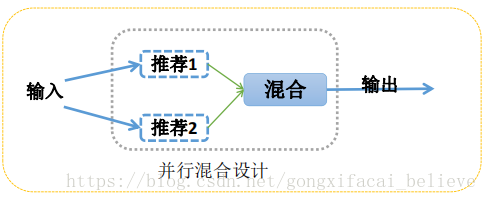
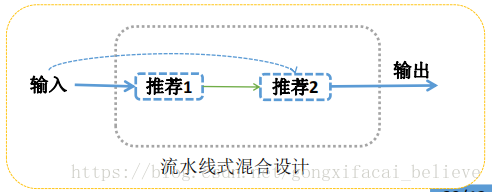
**三、Hybrid Recommendation 混合推薦**

「混合推薦」顧名思義就是結合上述的內容推薦與協同過濾等方法來建立推薦系統，以增進推薦的效率。

由於資料的複雜程度，以及每種商品都有其適合的推薦方式，業界許多推薦系統的應用都會混合不同的方法甚至結合機器學習演算法來做推薦系統。

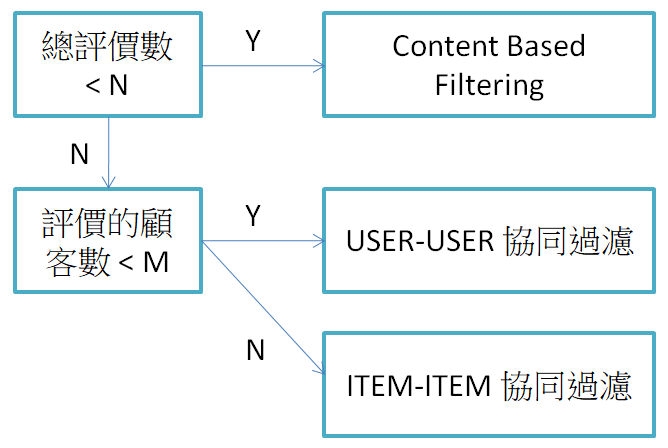
我們可以把推薦系統看成一個黑盒子，可以將輸入的資料轉換成為一個有序的物品列表再進行輸出，輸入的資料主要包含使用者記錄、上下文資訊、產品特徵、知識模型等等，但是沒有任何一個推薦演算法可以利用到這所有的輸入資料，所以可以考慮將多個推薦系統模型的結果混合到一起來作為最終的推薦結果。

混合推薦系統的設計結構主要分為三大類，分別是：整體式混合設計、並行式混  
合設計、流水線式混合設計。

**混合的推薦模型例一**

結合兩種方法，混合使用，兼容並蓄，那就天下無敵了。混合使用原則如下圖：



其中的N是網站的總評價數，M是單項商品的總評價數，可自訂門檻。

**其他考量**

推薦系統在實務上會碰到的問題還蠻多的，列舉以下幾點：

1. 交易資料過大，Pandas 一次載入，記憶體爆掉：

* 最簡單的方法就是只取一段時間的資料加以分析。
* Pandas 有分塊(Chunk)讀取的功能。

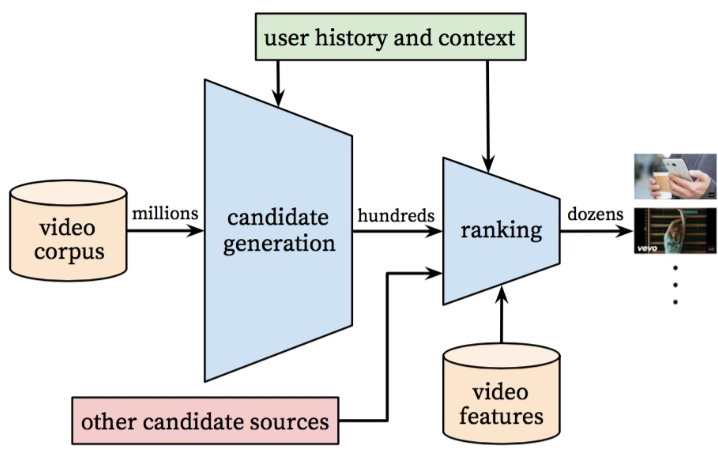
以上程式節錄自[『3 simple ways to handle large data with Pandas』](https://towardsdatascience.com/3-simple-ways-to-handle-large-data-with-pandas-d9164a3c02c1)

* 改用 Spark 處理，利用 RDD/MapReduce 分散處理機制。
* TensorFlow 也有解方，類似 iterator 機制。

1. 有些隱藏性的特徵(Latent Features)可能也是推薦系統重要的變數，例如，觀看影片長度、停留在網頁的時間，前者可進一步確認使用者是真的有看完影片，確定喜歡這部片子，而不是preview而已。
2. 情境相關的推薦系統(Context-aware recommendation systems, CARS)：額外考慮使用者的情境，例如觀賞地點(在家或點影院)、獨自觀賞/闔家觀賞、觀賞時間(平日或假日)等等。

**混合的推薦模型例二**

YouTube 在 2016 年有發表一個完整的架構，它主要是以兩個神經網路(Neural Networks)建構而成，有興趣的讀者可參閱[『Deep Neural Networks for YouTube Recommendations』](https://static.googleusercontent.com/media/research.google.com/zh-TW/pubs/archive/45530.pdf)。



**--------------------------------------------------**

***推薦系統的相關特性***

**推薦系統的缺點與限制**

看似完美的推薦系統仍然有存在的限制與缺點，主要分為兩個方向：

1. Cold start 冷啟動
2. Data sparsity 資料稀疏

**Cold Start (冷啟動)的問題**

在推薦系統中，冷啟動的問題是指「沒有充足的訊息」而造成推薦效果不佳。以電商平台來說，冷啟動的問題可以歸類成兩個面向：產品與使用者。新的電商平台可能沒有足夠的產品上架或是會員資料，導致推薦系統的訊息不足，無法有效運作，進而影響推薦效果。

此外冷啟動在協同過濾系統中一直是很大的問題。對於曝光度低的冷門產品(long-tail item)，無論是哪一種協同過濾系統(User-based / Item-based)，都不會把商品推薦給使用者，使得那個產品會一直處在冷門的狀態。

**Data sparsity 資料稀疏的問題**

然而，當電商平台的資訊量過多的時候，也可能造成所謂「資料稀疏」的問題。舉例來說，如果平台上的某個品類有太多相似的產品，當使用者瀏覽頁面的時候無法看過大部分產品，就會造成蒐集資料上的稀疏。

資料的規模越大，資料稀疏的問題**就越嚴重，這也是目前許多學者們們想解決的問題。**

***推薦系統相關技巧***

相似性(Similarity)有三種統計量：

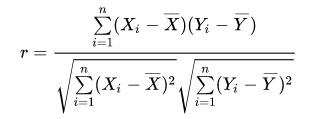
1. Jaccard Similarity：

(購買A **且** 購買B 的交易筆數) / (購買A **或** 購買B 的交易筆數)

Jaccard Similarity 雅卡爾相似度，就是對於所有的聯集中，有多少比例是交集。

1. Pearson Similarity：兩兩計算其『皮爾森係數』(Pearson coefficient)。

相對於Cosine Similarity更好的方式，是對於所有的個別項都減掉各自的平均，就變成皮爾森相關係數（Pearson Correlation Coefficient）。以下就是皮爾森相關係數的公式，是不是其實只比餘弦相似度多了減掉平均的項？



為了讓評分能夠各自區分出喜歡與不喜歡，因此我們個別將每一項減去其平均。且通常用戶僅會給予少部分的電影評分，因此將沒有值的地方先簡單補上 0，避免相似度無法計算。

1. Cosine Similarity：最常用的方法。  
   不管我們是採用哪一種協同過濾，都是將矩陣中每一欄或列視為一向量，兩兩計算其統計量，得到一個相似性的值。例如Cosine Similarity，當Cosθ=1時，表示夾角=0°，兩者最相似，反之，Cosθ=-1時，表示夾角=180°，兩者最不相似。

Cosine Similarity 餘弦相似度

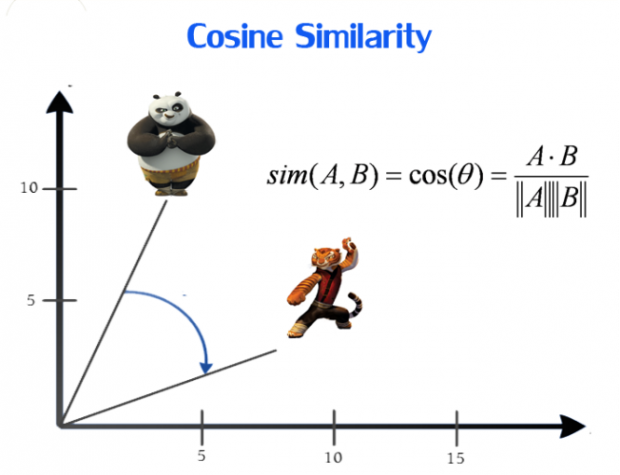
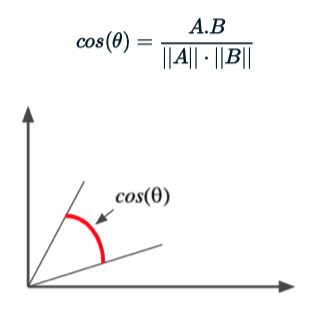
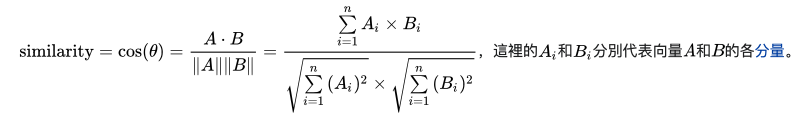


圖. Cosine Similarity

其實餘弦相似度是來自於兩個向量之間的餘弦值，能夠透過歐基理得點積公式求出，公式如下：

https://miro.medium.com/max/498/1*PpN9_SMh6nRDzlstXmUVhw.png

可以看到只要簡單地將兩邊同除兩向量的純量乘積後，就和第一個的餘弦相似度公式完全一模一樣。

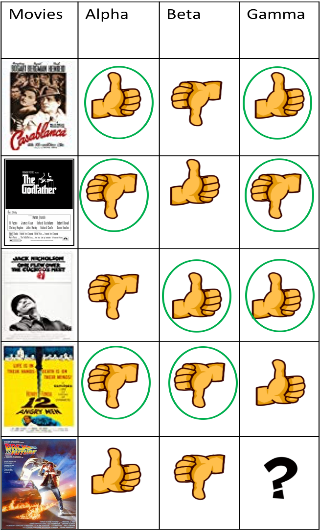


餘弦值的值會從 -1~1 之間，其特性是：如果兩個向量的方向很接近，那他就會接近 1；完全反方向則為 -1；0 通常代表這兩者之間是獨立的。而在分母除以分量的過程，可以幫他當作是對於各文件間正規化的方法。

同樣地，我們也不需要自己來計算餘弦相似度，Sklearn 一樣幫我們算好，我們只要把套件引入就完事。

**範例**

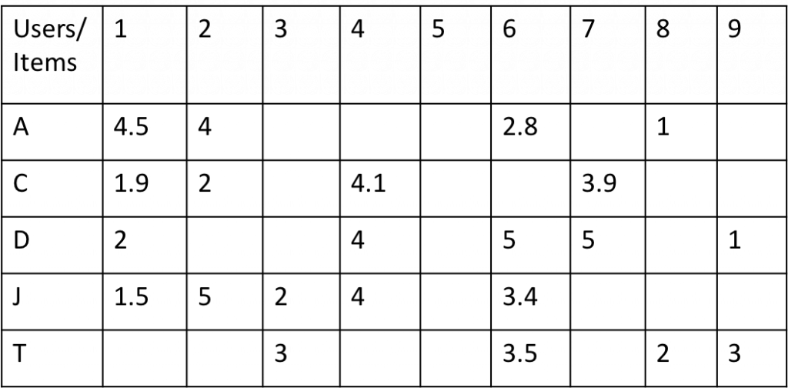
先看一個簡單的例子，假設有5部電影、3個觀眾評論，要猜第3個觀眾是否喜歡第5部電影：  
以下圖片來自：[Recommendation Systems by Akanksha](https://medium.com/x8-the-ai-community/recommendation-system-db51c868f13d)

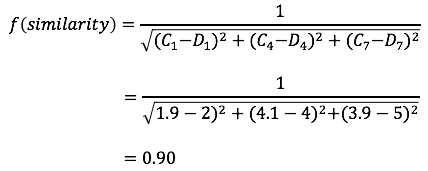


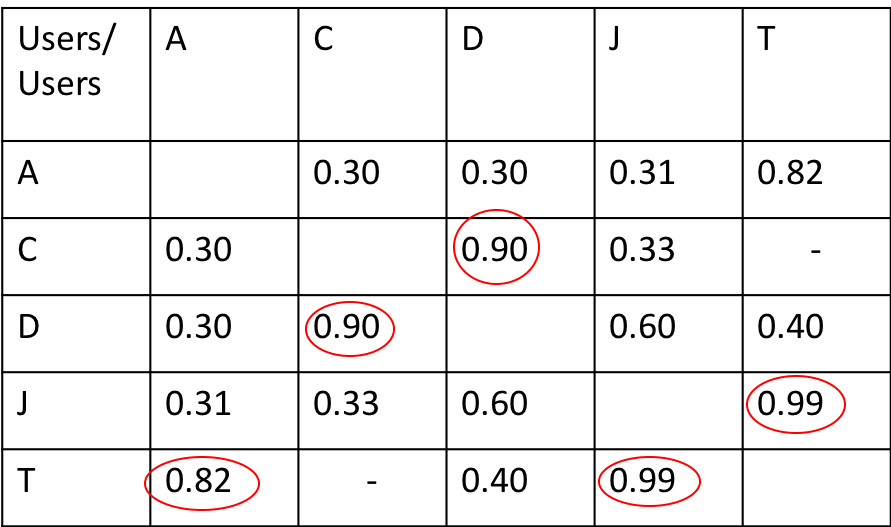
解法很簡單：

1. 第3個觀眾與第1個觀眾相同意見有2個，即 2/4 = 0.5
2. 第3個觀眾與第2個觀眾相同意見有1個，即 1/4 = 0.25  
   所以，第3個觀眾與第1個觀眾意見較相近，故猜第3個觀眾與第1個觀眾都喜歡第5部電影。

以上只有喜歡/不喜歡兩種，若觀眾評論改為分數，要如何計算? 例如觀眾評分如下：



兩兩使用者計算其反歐基里德距離(inverse Euclidean distance)，例如 C 與 D 的Cosine值計算如下：  
https://ithelp.ithome.com.tw/upload/images/20190917/20001976NuXN8MTKAA.png  


全部算完後，如下表：  


結論如下：

1. 與A最相似的是T。
2. 與C最相似的是D。
3. 與J最相似的是T。

ITEM-ITEM Similarity Matrix 產生的方式以列(Item)作比較，算法完全一樣。

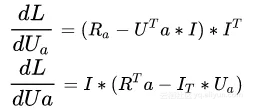
**矩陣分解**

矩陣分解 (decomposition, factorization)是將矩陣拆解為數個矩陣的乘積。比如豆瓣電影有m個使用者，n個電影。那麼使用者對電影的評分可以形成一個m行n列的矩陣R，我們可以找到一個m行k列的矩陣U，和一個k行n列的矩陣I，通過U \* I來得到矩陣R。

**ALS**

如果想通過矩陣分解的方法實現基於模型的協同過濾，ALS是一個不錯的選擇，其英文全稱是Alternating Least Square，翻譯過來是交替最小二乘法。假設使用者為a，物品為b，評分矩陣為R(m, n)，可分解為使用者矩陣U(k, m)和物品矩陣I(k, n)，其中m, n, k代表矩陣的維度。前方小段數學公式低能預警：

根據矩陣分解的定義，有  
image  
用MSE作為損失函式，為了方便化簡，加法符號左側的常數改為-1/2  
image

1. 對損失函式求U\_a的一階偏導數，那  
   
2. 令一階偏導數等於0  
   image
3. 同理，可證  
   image

求解使用者矩陣U和物品矩陣I

矩陣R是已知的，我們隨機生成使用者矩陣U，

1. 利用1.5中的式5、R和U求出I
2. 利用1.5中的式6、R和I求出U

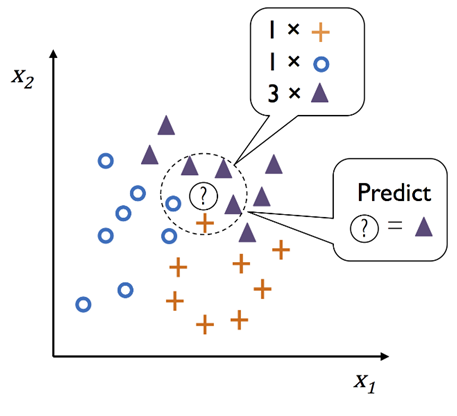
如此交替地執行步驟1和步驟2，直到演算法收斂或者迭代次數超過了最大限制，最終我們用RMSE來評價模型的好壞。

Python實現了ALS演算法:

<https://www.itread01.com/content/1546500020.html>

**最近鄰(k-Nearest Neighbors, KNN)**

KNN是監督式學習演算法中最簡單的方法，它計算所有樣本點與預測點的距離，可以取1、3、5...點，然後以多數決（Majority Voting）決定預測點是哪一類，如下圖：



1. 尋找距離預測值最近的 5 個樣本點。
2. 以多數決決定所屬分類，5 個樣本點有3個三角形，故認為測試資料為三角形的可能性最大。

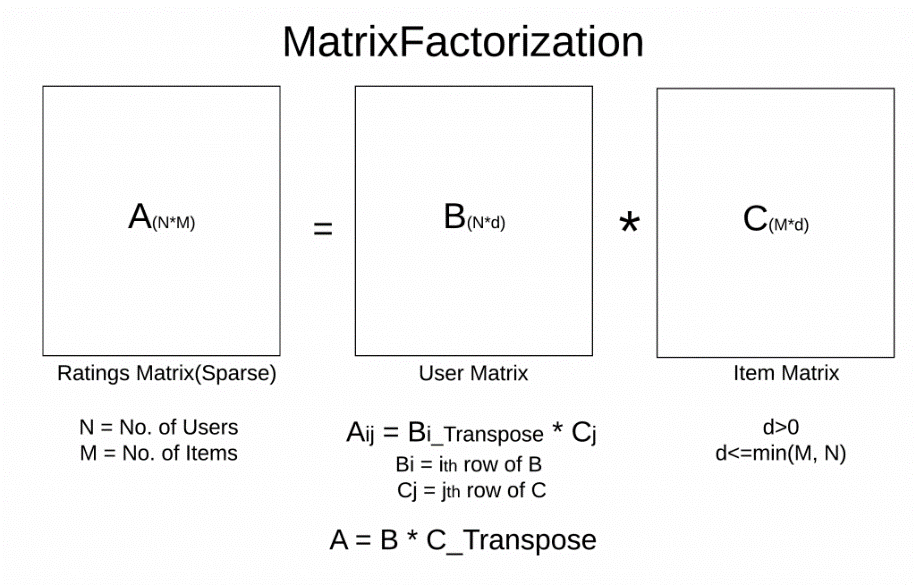
KNN 實作

<https://ithelp.ithome.com.tw/articles/10220479>

**矩陣分解(Matrix Factorization)**

NetFlix 很久之前辦過一個比賽 [Netflix Prize](https://en.wikipedia.org/wiki/Netflix_Prize)，目標是希望有新的演算法能作幫該公司作電影推薦，比賽結果是『矩陣分解』(Matrix Factorization)演算法脫穎而出，其中又以奇異值分解(Singular Value Decomposition, SVD)為代表。

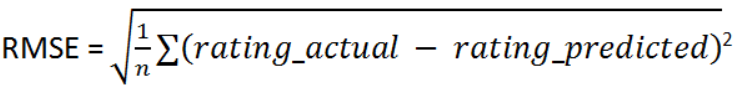
通常 User-Item Matrix 會是一個很稀疏的矩陣(Sparse Matrix)，因為，每個使用者通常只買過網站中很少種類的產品，故對照表矩陣中應該大部分是空值(Null Value)，因此，我們可以利用特徵向量(Eigenvector)或SVD作矩陣轉換，將稀疏的矩陣降維，變成比較小的矩陣，那計算就比較省時了。



資料來源：[How to Build a Recommender System](https://medium.com/datadriveninvestor/how-to-built-a-recommender-system-rs-616c988d64b2" \t "_blank)，以下圖形也都來自此篇文章。

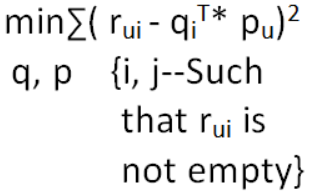
**原理**

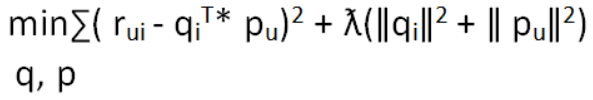
以下牽涉數學較多，沒興趣的讀者就跳過吧。

依最小平方法(OLS)定義的損失函數(Loss Function)或稱目標函數(Objective Function)如下:  


以上A/B/C矩陣對照商品推薦的資料如下：

1. A：rui -- 使用者(u)對商品(i)的評價(r)。
2. B：qi -- 商品(i)的向量
3. C：pu -- 使用者(u)的向量

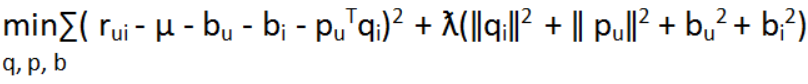
損失函數就可改寫如下：  


為避免過度擬合(overfitting)，損失函數加上 L2 正則化(Regularization)，如下：  


定義好損失函數之後，可以使用兩種優化方法(Optimizer)，求出商品(i)及使用者(u)的向量：

1. 梯度下降法(Stochastic Gradient Descent, SGD)：這是神經網路求解的方式，分別對qi、pu作偏微分，逐步調整權重，直到損失函數收斂(converge)為止。
2. 交替最小平方法(Alternating Least Squares, ALS)：先固定pu，再對qi作偏微分，之後，再固定qi，再對pu作偏微分，如此交替，直到收斂為止。

ALS方法計算比較快，通常，會使用ALS。

之後作者發現一個問題，此算法不能保證可以最小化RMSE，因此，作者再加三個參數：bu, bi, µ，再修改損失函數如下：  


bu：估計每個使用者的偏差項。  
bi：估計每個商品的偏差項。  
µ ：所有使用者及所有商品的平均值。

搞了這麼多，總之，就是利用上述的損失函數，優化求解，有興趣的讀者可參考[『Day N+1：進一步理解『梯度下降』(Gradient Descent)』](https://ithelp.ithome.com.tw/articles/10198147)。

**參考資料**

Adomavicius, G., Tuzhilin, A. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. "IEEE Transactions on Knowledge and Data Engineering 17(6), pp.734–749, 2005.

Basu, Chumki, Haym Hirsh, and William Cohen. "Recommendation as classification: Using social and content-based information in recommendation." Aaai/iaai, pp. 714-720, 1998.

Maria Temming, "Solving problems by computer just got a lot faster", Science News, July 16, 2018.

<https://www.itread01.com/content/1546329264.html>

[How to Build a Recommender System](https://medium.com/datadriveninvestor/how-to-built-a-recommender-system-rs-616c988d64b2)

推薦系同原理細說

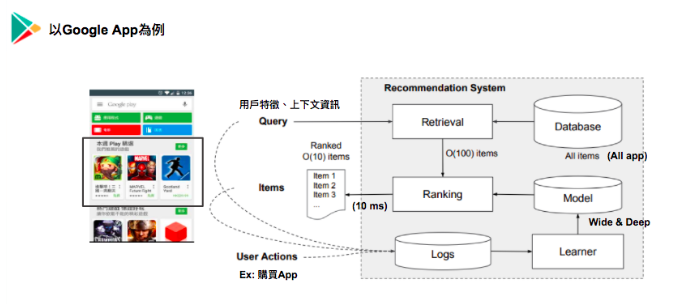
https://ithelp.ithome.com.tw/articles/10195751

https://blog.gcp.expert/recommendation-system-tensorflow-overview/

SVC

https://laoweizz.blogspot.com/2018/12/svms.html

專利資料檢索可參考模式

針對Google所提出的Recommendation System的架構為下圖：  
  
當使用者進入 Google Play，系統會針對使用者特徵以及頁面資訊對DB做檢索，從DB選出Top item針對Model 預測出來的結果做排序，並推薦產品。而Model所training的資料則是從使用者的行為資料所萃取出來。

**\*\*\* 本講義無法保證其正確性，必須自行確認正確性 \*\*\***