DATA MINING PROJECT1 報告
P76064732 曹何謙

環境 python 3.6 windows 10

程式執行方式: 參照 README

實做演算法

(1) Apriori : 利用 Apriori 和 level-wise 的方式 join/prune candidate itemset。
由於純暴力法電腦跑太久所以使用這個演算法作為 fp growth 的 baseline。
(2) Fp growth

資料集

利用 IBM Quest Data Generator 產生不同參數組合的 transaction file 之後,使用以下這五個檔案作為資料集。(*.tl 代表經過預處理的檔案)

file name	transaction number	avg item number per transaction	item number
131.tl	500	3	1000
151.tl	800	5	1000
531.tl	3000	3	1000
555.tl	5000	5	5000
test2.tl	10	3	5

觀察以上兩個演算法在不同資料集的表現(執行時間):

file name	apriori (sec)	fp growth (sec)	min support	min confidence	generated rule number
131.tl	78	0.04	0.005	0.5	0
131.tl	153	0.06	0.003	0.5	15
151.tl	94	0.2	0.005	0.5	1
531.tl	45	0.05	0.003	0.5	1
555.tl	32	0.09	0.004	0.5	1
555.tl	333	0.2	0.003	0.5	0
test2.tl	10	0.00099	0.00099	0.2	0.5

由以上結果可以觀察到當 support 設的愈低,所需要的執行時間越久,Ariori 和 Fp growth 的執行時間的差距越大,而 test2 由於檔案很小兩者執行時間差不多。

Confidence 都設一樣的原因是因為產生 rule 的時間遠小於產生 frequent itemset 的時間,由於不是影響到執行時間的主要原因所以設成一樣的不納入討論內。
使用這兩個演算法設定 support 和 confidence(尤其是 support)特別的重要,音位設太小的話會執行太久(apriori)而且會產生太多 frequent itemset,設太大的話幾乎產生不了什麼規則。
另外,比起 transaction number,support 的設定會更影響到執行的時間(比較 131.tl 和 555.tl),另外由於資料是隨機產生的 avg item number per transaction 愈大 item number 愈小的情況下,越容易產生高於 min support 的 pattern,這個時候就會執行比較久。
以上的比較其實不夠精準,因為實際上計算 itemset 的次數是去 database 裡面 scan,而我的程式是先從檔案讀入後之後都在記憶體內算次數,儘管如此執行的效率就差那麼多了,如果實際上要去 database 裡面算次數一定差距更大。