Trường Đại học Khoa học Tự nhiên TP.HCM

Khai thác dữ liệu và ứng dụng - 19_21

Lab 02 - Frequent Itemset Mining

Họ tên: Bùi Quang Bảo

MSSV: 19120454

Mục lục:

1. Thuật toán Apriori:

Apriori là một trong những thuật toán sử dụng cho khai thác tập phổ biến.

1.1. Ý tưởng:

Giả sử ta có bộ dữ liệu như sau:

4 3 2
1 2 3 5
2 6 3
2 6 3
6 1 3
5 1 2 4
3 4 2
1 3
4 1 2 3
6 4 2 3

Mỗi dòng là 1 “transaction” với các item được ngăn cách bởi dấu cách.

Chúng ta sẽ tìm tập phổ biến của dữ liệu trên bằng thuật toán **Apriori**, với **minimum support count là 3**.

Đầu tiên, với itemset chứa 1 item:

C1:

Itemset	Support Count
1	5
2	8
3	9
4	5
5	2 (< minimum support count)

6	4

L1 (Loại bỏ các itemset có support count < minimum support count):

Itemset	Support Count
1	5
2	8
3	9
4	5
6	4

Với itemset chứa 2 items:

C2 (được tạo thành từ việc join các itemset trong L1):

Itemset	Support Count
1 2	3
1 3	4
1 4	2 (< minimum support count)
1 6	1 (< minimum support count)
2 3	7
2 4	5
2 6	3
3 4	4

3 6	4
4 6	1 (< minimum support count)

L2 (Loại bỏ các itemset có support count < minimum support count):

Itemset	Support Count
1 2	3
1 3	4
2 3	7
2 4	5
2 6	3
3 4	4
3 6	4

Với itemset chứa 3 items:

C3 (được tạo thành từ việc join các itemset trong L2):

Itemset	Support Count
1 2 3	2 (< minimum support count)
1 2 4	2 (< minimum support count)
1 2 6	0 (< minimum support count)
1 3 4	1 (< minimum support count)
1 3 6	1 (< minimum support count)

2 3 4	4
2 3 6	3
2 4 6	1 (< minimum support count)

L3 (Loại bỏ các itemset có support count < minimum support count):

Itemset	Support Count
2 3 4	4
2 3 6	3

Với itemset chứa 4 items:

C4 (được tạo thành từ việc join các itemset trong L3):

Itemset	Support Count
2 3 4 6	1 (< minimum support count)

L4 (Loại bỏ các itemset có support count < minimum support count):

Tới đây L4 không còn chứa itemset nào, **dừng thuật toán Apriori**.

Sau khi thực hiện thuật toán Apriori với minimum support count = 3, ta có được kết quả như sau:

Itemset	Support Count
1	5
2	8
3	9
4	5

6	4
1 2	3
1 3	4
2 3	7
2 4	5
2 6	3
3 4	4
3 6	4
2 3 4	4
2 3 6	3

Với kết quả này, ta hoàn toàn có thể tính ra được confidence của các association rules sinh ra từ 1 itemset.
Công thức: confidence(A→B) = support_count(A∪B) / support_count(A)

Ví dụ: Với Itemset [2, 3, 4] có support count = 4:
Confidence([2] → [3, 4]) = 4 / 8 = 0.5
Confidence([3] → [2, 4]) = 4 / 9 = 0.44
Confidence([4] → [2, 3]) = 4 / 5 = 0.8 (*)
Confidence([2, 3] → [4]) = 4 / 7 = 0.57
Confidence([2, 4] → [3]) = 4 / 5 = 0.8 (*)
Confidence([3, 4] → [2]) = 4 / 4 = 1.0 (*)

Nếu ta đặt minimum confidence = 0.8, thì những rule (*) là những rule mạnh.

Chúng ta có thể làm điều tương tự đối với tất cả các itemset còn lại trong tập phổ biến và có được danh sách những association rules thoả confidence >= minimum confidence).

1.2. Thuật toán Apriori:

Apriori (hàm **apriori**):

- Tìm L1 (hàm **apriori_data_to_L1**)
- k = 2
- Tìm L[k] từ L[k-1] cho đến khi nào L[k-1] rỗng:
    - Tìm C[k] từ L[k-1]:
        - Tìm tất cả các candidate itemset cho C[k] từ L[k-1] (trong cài đặt và code chúng ta sẽ gọi là C_items) (hàm **apriori_L_to_C_items**)
        - Từ C_items, đếm support count dựa trên dữ liệu để tạo C[k] (hàm **apriori_C_items_to_C**)
    - Tìm L[k] từ C[k]:
        - Từ C[k], chỉ chọn những itemsets có support_count >= min_support_count (hàm **apriori_C_to_L**)
- Trả về kết quả

Tính toán Confidence (hàm **apriori_confidence**):
confidence(A→B) = support_count(A∪B) / support_count(A)

Association Rules (hàm **rules_gen**):
Tạo các association rules từ 1 itemset, tính toán confidence và in ra những rules có confidence >= min_confidence

1.3. Cài đặt thuật toán Apriori bằng ngôn ngữ Julia:

Sử dụng thư viện: Không

```
# ======================== Apriori Algorithm ========================
function apriori(data, min_support_count)
    L = []
    # Initialize L with L1
    push!(L, apriori_data_to_L1(data, min_support_count))
    k = 2
    # Find all L[k] from L[k-1] until L[k-1] is empty
    while !isempty(L[k-1])
        print("> Processing: L[$(k-1)] to L[$(k)]: ")
        # Find L[k] from L[k-1]
        # Use all the sub-function for Apriori
        Lk = apriori_C_to_L(
            apriori_C_items_to_C(
                data, apriori_L_to_C_items(L[k-1])
            ), min_support_count
        )
        # Add L[k] to L
        push!(L, Lk)
        k += 1
    end
    # Just remove the last is an empty list
    pop!(L)
    # Just print the apriori result to the console
    apriori_print(L)
    return L
end
# ===================================================================
```

Các hàm bổ trợ cho thuật toán Apriori hay các hàm phục vụ cho tính toán khác đều đã được comment đầy đủ trong source code, file "**apriori.jl**".

1.4. Kiểm thử:

Chúng ta sẽ sử dụng chính bộ dữ liệu được nêu ra trong phần “1.1. Ý tưởng” được đặt trong file “**test.txt**” để kiểm tra:

```
4 3 2
1 2 3 5
2 6 3
2 6 3
6 1 3
5 1 2 4
3 4 2
1 3
4 1 2 3
6 4 2 3
```

Với cài đặt ở đầu file “apriori.jl” như sau:

```
input_file_name = "data/test.txt"
min_support_count = 3
min_confidence = 0.8
```

Có nghĩa là:

- Sử dụng input file “test.txt”
- Thực hiện thuật toán Apriori với minimum support count = 3
- Tạo các association rules từ itemset [2, 3, 4] và tính toán confidence, chỉ lấy những rules thoả mãn confidence >= 0.5

Chúng ta có console output như sau:

```
PS D:\Google Drive\University\Data Mining\Lab02 - Frequent Itemset Mining> julia apriori.jl
====================== Apriori Processing ========================
> Read file successfully
> Turn data into proper format successfully
> Processing: L[1] to L[2]: L -> C's items | C's items -> C | C -> L
> Processing: L[2] to L[3]: L -> C's items | C's items -> C | C -> L
> Processing: L[3] to L[4]: L -> C's items | C's items -> C | C -> L
============================ Result ==============================
Itemset -> Support Count
[4] -> 5
[3] -> 9
[2] -> 8
[1] -> 5
[6] -> 4
[3, 4] -> 4
[2, 4] -> 5
[2, 3] -> 7
[1, 3] -> 4
[3, 6] -> 4
[1, 2] -> 3
[2, 6] -> 3
[2, 3, 4] -> 4
[2, 3, 6] -> 3
==================================================================
All Association Rules with confidence >= 0.8:
Confidence([4] => [3]) = 0.8
Confidence([4] => [2]) = 1.0
Confidence([2] => [3]) = 0.88
Confidence([1] => [3]) = 0.8
Confidence([6] => [3]) = 1.0
Confidence([4] => [2, 3]) = 0.8
Confidence([2, 4] => [3]) = 0.8
Confidence([3, 4] => [2]) = 1.0
Confidence([2, 6] => [3]) = 1.0
==================================================================
PS D:\Google Drive\University\Data Mining\Lab02 - Frequent Itemset Mining>
```

Thuật toán chạy thành công và cho ra kết quả như mong đợi.

1.5. Ứng dụng khai thác tập phổ biến:

Bộ dữ liệu được lấy từ [philippe-fournier-viger.com]

1.5.1. Tập dữ liệu Mushrooms.txt:

Cài đặt ở đầu file “apriori.jl”:

```
input_file_name = "data/mushrooms.txt"
min_support_count = 4000
min_confidence = 0.97
```

Output:

```
Confidence([36, 38, 41, 94] => [90]) = 1.0
Confidence([38, 41, 90, 94] => [36]) = 0.99
Confidence([38, 97] => [36, 90, 94]) = 0.97
Confidence([36, 38, 97] => [90, 94]) = 1.0
Confidence([38, 90, 97] => [36, 94]) = 0.97
Confidence([36, 38, 90, 97] => [94]) = 1.0
Confidence([38, 94, 97] => [36, 90]) = 1.0
Confidence([36, 38, 94, 97] => [90]) = 1.0
Confidence([38, 90, 94, 97] => [36]) = 1.0
Confidence([24, 36, 97] => [90, 94]) = 1.0
Confidence([24, 36, 90, 97] => [94]) = 1.0
Confidence([24, 94, 97] => [36, 90]) = 1.0
Confidence([24, 36, 94, 97] => [90]) = 1.0
Confidence([24, 90, 94, 97] => [36]) = 1.0
Confidence([36, 38, 41, 97] => [90]) = 1.0
Confidence([36, 38, 41, 97] => [94]) = 1.0
Confidence([38, 41, 94, 97] => [36]) = 1.0
Confidence([38, 41, 94, 97] => [90]) = 1.0
Confidence([36, 38, 41, 97] => [90, 94]) = 1.0
Confidence([36, 38, 41, 90, 97] => [94]) = 1.0
Confidence([38, 41, 94, 97] => [36, 90]) = 1.0
Confidence([36, 38, 41, 94, 97] => [90]) = 1.0
Confidence([38, 41, 90, 94, 97] => [36]) = 1.0
================================================================
PS D:\Google Drive\University\Data Mining\Lab02 - Frequent Itemset Mining>
```

(Vì khá dài nên em chỉ screenshot đoạn cuối console, thầy/cô có thể coi đầy đủ ở file “**output_apriori_mushrooms_4000.txt**”)

1.5.2. Tập dữ liệu Chess.txt:

Cài đặt ở đầu file “apriori.jl”:

```
input_file_name = "data/chess.txt"
min_support_count = 3000
min_confidence = 0.98
```

Output:

```
Confidence([36, 40, 58, 60] => [52]) = 1.0
Confidence([36, 52, 58, 60] => [40]) = 0.99
Confidence([29, 36, 40] => [52, 58, 60]) = 0.98
Confidence([36, 40, 52] => [29, 58, 60]) = 0.98
Confidence([29, 36, 40, 52] => [58, 60]) = 0.99
Confidence([29, 36, 40, 58] => [52, 60]) = 0.98
Confidence([36, 40, 52, 58] => [29, 60]) = 0.98
Confidence([29, 36, 40, 52, 58] => [60]) = 0.99
Confidence([36, 60] => [29, 40, 52, 58]) = 0.98
Confidence([29, 36, 60] => [40, 52, 58]) = 0.99
Confidence([36, 40, 60] => [29, 52, 58]) = 0.99
Confidence([29, 36, 40, 60] => [52, 58]) = 1.0
Confidence([36, 52, 60] => [29, 40, 58]) = 0.99
Confidence([29, 36, 52, 60] => [40, 58]) = 0.99
Confidence([36, 40, 52, 60] => [29, 58]) = 1.0
Confidence([29, 36, 40, 52, 60] => [58]) = 1.0
Confidence([36, 58, 60] => [29, 40, 52]) = 0.98
Confidence([29, 36, 58, 60] => [40, 52]) = 0.99
Confidence([36, 40, 58, 60] => [29, 52]) = 0.99
Confidence([29, 36, 40, 58, 60] => [52]) = 1.0
Confidence([36, 52, 58, 60] => [29, 40]) = 0.99
Confidence([29, 36, 52, 58, 60] => [40]) = 0.99
Confidence([36, 40, 52, 58, 60] => [29]) = 1.0
================================================================
PS D:\Google Drive\University\Data Mining\Lab02 - Frequent Itemset Mining>
```

(Thầy/cô có thể coi đầy đủ ở file “**output_apriori_chess_3000.txt**”)

1.5.3. Tập dữ liệu FoodmartFIM.txt:

Cài đặt ở đầu file “apriori.jl”:

```
input_file_name = "data/foodmartFIM.txt"
min_support_count = 15
min_confidence = 0
```

Output:

```
[522] -> 16
[1103] -> 15
[1386] -> 16
[839] -> 16
[1094] -> 15
[1343] -> 15
[528] -> 19
[377] -> 17
[520] -> 16
[863] -> 15
[518] -> 15
[246] -> 19
[1292] -> 23
[1081] -> 16
[402] -> 15
[1547] -> 17
[989] -> 15
[448] -> 19
[20] -> 17
[83] -> 16
[1308] -> 18
================================================================
All Association Rules with confidence >= 0:
================================================================
PS D:\Google Drive\University\Data Mining\Lab02 - Frequent Itemset Mining>
```

(Thầy/cô có thể coi đầy đủ ở file “**output_apriori_foodmartFIM_15.txt**”)

Có vẻ như với bộ dữ liệu “FoodmartFIM.txt”, chúng ta thật sự “khó” tìm được tập phổ biến, thậm chí không có itemset nào chứa 2 item có support_count >= 15.

Nếu tiếp tục giảm min_support_count xuống thấp hơn (ví dụ như 5 hay 10 chẳng hạn), có thể chúng ta sẽ nhận được vài itemset chứa nhiều hơn 1 item thoả mãn. Tuy nhiên điều đó có còn có ý nghĩa thực tiễn hay không đối với 1 tập dữ liệu hơn 4000 dòng, nhưng itemset chỉ có support_count là 5 hay 10?

So với “Mushrooms.txt” (với min_support_count = 4000, tìm được 1 itemset chứa tới 6 items) và “Chess.txt” (với min_support_count = 3000, cũng tìm được 1 itemset chứa tới 6 items) thì “FoodmartFIM.txt” hầu như rất khó tìm tập phổ biến.

1.5.4. Tập dữ liệu Retail.txt:

Cài đặt ở đầu file “apriori.jl”:

```
input_file_name = "data/retail.txt"
min_support_count = 10000
min_confidence = 0.5
```

Output:

```
PS D:\Google Drive\University\Data Mining\Lab02 - Frequent Itemset Mining> julia apriori.jl
====================== Apriori Processing ========================
> Read file successfully
> Turn data into proper format successfully
> Processing: L[1] to L[2]: L -> C's items | C's items -> C | C -> L
> Processing: L[2] to L[3]: L -> C's items | C's items -> C | C -> L
============================ Result ==============================
Itemset -> Support Count
[33] -> 15167
[39] -> 15596
[40] -> 50675
[42] -> 14945
[49] -> 42135
[39, 40] -> 10345
[40, 42] -> 11414
[40, 49] -> 29142
==================================================================
All Association Rules with confidence >= 0.5:
Confidence([39] => [40]) = 0.66
Confidence([42] => [40]) = 0.76
Confidence([40] => [49]) = 0.58
Confidence([49] => [40]) = 0.69
==================================================================
PS D:\Google Drive\University\Data Mining\Lab02 - Frequent Itemset Mining>
```

(Chạy rất lâu, output lần này khá ngắn, tuy nhiên thầy/cô vẫn có thể coi đầy đủ ở file “**output_apriori_retail_10000.txt**”)

Với min_support_count = 10000, min_confidence = 0.5, chúng ta tìm được một số frequent itemset và association rules như trên.

1.6. Hạn chế của thuật toán Apriori:

Hạn chế của thuật toán Apriori: Chậm, thời gian hoàn thành lâu đối với những bộ dữ liệu lớn (bộ dữ liệu “retail.txt” là một ví dụ, với hơn 88000 dòng).

Lí do:

- Mỗi lần tạo một k-itemset mới thì phải tạo 1 candidates set (chính là C[k]).
- Mỗi lần tạo L[k] từ C[k], thuật toán phải “quét” lại bộ dữ liệu một lần để đếm support count.
- Đối với những bộ dữ liệu lớn + nhiều loại items khác nhau (có thể tạo ra nhiều itemset khác nhau), thực hiện thuật toán Apriori có thể không thành công do giới hạn phần cứng như RAM.

2. Thuật toán Tree Projection:

Tree Projection là một trong những thuật toán sử dụng cho khai thác tập phổ biến.

2.1. Ý tưởng:

Chúng ta sẽ sử dụng lại chính bộ dữ liệu “test.txt” đã sử dụng cho Apriori để có sự so sánh giữa 2 thuật toán:

4 3 2
1 2 3 5
2 6 3
2 6 3
6 1 3
5 1 2 4
3 4 2
1 3
4 1 2 3
6 4 2 3

min_support _count = 3

Đầu tiên chúng ta tìm “Frequent Pattern Set” (Ở thuật toán Apriori, đây chính là L1):

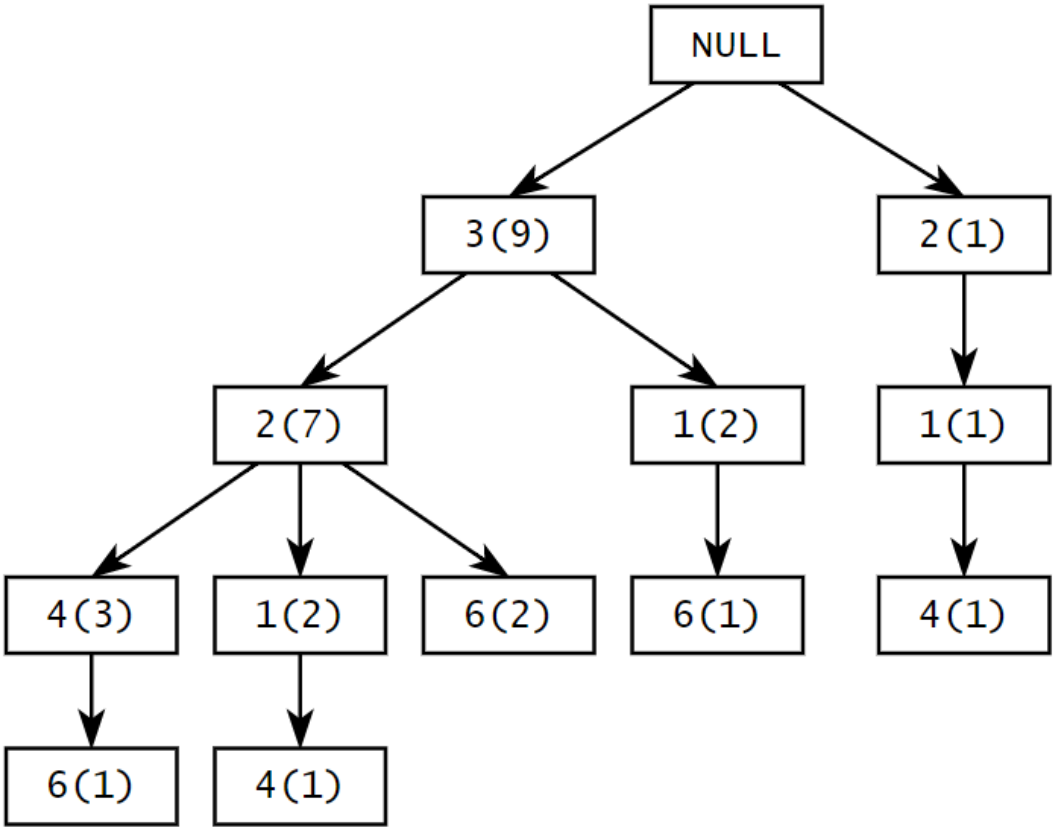
Itemset	Support Count (descending order)
3	9
2	8
1	5
4	5
6	4

(Lưu ý rằng khác với L1 ở thuật toán Apriori, FP Set của thuật toán Tree Projection sẽ được sắp xếp giảm dần theo support count.)

Tiếp theo là “Ordered-Item Set”, được tạo ra bằng cách sắp xếp lại các items trên 1 dòng dữ liệu theo thứ tự của FP Set mà chúng ta vừa thực hiện ở trên.

Transaction	Ordered-Item Set
4 3 2	3 2 4
1 2 3 5	3 2 1
2 6 3	3 2 6
2 6 3	3 2 6
6 1 3	3 1 6
5 1 2 4	2 1 4
3 4 2	3 2 4
1 3	3 1
4 1 2 3	3 2 1 4
6 4 2 3	3 2 4 6

Xây dựng FP Tree dựa trên Ordered-Item Set (Mỗi node biểu diễn 1 item cùng số lần xuất hiện của node đó, mỗi item có thể được biểu diễn bởi 1 hay nhiều node khác nhau):



Xây dựng “Conditional Pattern Base” → “Conditional Frequent Pattern Tree” → “Frequent Itemsets”:

- Items: được sắp xếp tăng dần theo support count
- Conditional Pattern Base: Tất cả những paths dẫn đến item (không gồm item)
- Conditional FP Tree: Những itemsets có mặt ở nhiều paths (>= 2 paths), chỉ chọn những itemsets nào có support count >= minimum support count
- Frequent Itemsets: Thêm item vào các itemsets trong Conditional FP Tree

Item	Conditional Pattern Base	Conditional FP Tree	Frequent Itemsets
6	[3,2,4]:1; [3,2]:2; [3,1]:1	[3]:4, [2]:3; [3,2]:3	[3,6]:4, [2,6]:3, [3,2,6]:3
4	[3,2,1]:1; [3,2]:3; [2,1]:1	[3]:4, [2]:5, ~~[1]:2~~; [3,2]:4; ~~[2,1]:2~~	[3,4]:4, [2,4]:5, [3,2,4]:4
1	[3,2]:2; [3]:2; [2]:1	[3]:4, [2]:3	[3,1]:4, [2,1]:3
2	[3]:7	3:7	[3,2]:7
3			

Tới đây, chúng ta có được danh sách các tập phổ biến (và support count). Đối chiếu và so sánh với thuật toán Apriori, thuật toán Tree Projection đã cho ra kết quả tương tự. Việc tính toán confidence tương tự như ở Apriori.

2.2. Thuật toán Tree Projection:

Tree Projection:

- Tìm Frequent Pattern Set (giống L1 của Apriori nhưng có sắp xếp giảm dần theo support count)
- Tìm Ordered-Item Set
- Xây dựng FP Tree
    - Xét từng dòng trong Ordered-Item Set và insert vào FP Tree
- Xây dựng Frequent Itemsets
    - Tìm Conditional Pattern Base
    - Tìm Conditional FP Tree
    - Tìm Frequent Itemsets

2.3. Cài đặt thuật toán Tree Projection bằng ngôn ngữ Julia:

Chưa hoàn thành

3. So sánh thuật toán Apriori và Tree Projection:

	Apriori	Tree Projection
Cấu trúc dữ liệu	Array-based	Tree-based
Xây dựng itemsets	Lưu dưới dạng k-itemset, tất cả các tập candidates (C) và k-itemset (L) đều được lưu trong bộ nhớ.	Lưu dưới dạng FP Tree, không cần tạo các tập candidates, itemset được tạo sau cùng. Gọn và tiết kiệm hơn Apriori.
Quét toàn bộ dữ liệu	Mỗi lần tạo 1 k-itemset.	Chỉ quét đúng 2 lần (để tạo Frequent Pattern Set và Ordered-Item Set).
Hiệu năng	Chậm hơn, thời gian thực thi lâu hơn.	Nhanh hơn, thời gian thực thi ngắn hơn.

Báo cáo đồ án kết thúc tại đây.

MSSV: 19120454

Họ tên: Bùi Quang Bảo