



# **Unsupervised Learning: Association Rule Learning**

**AAA-Python Edition**



# Plan

- 1- Association Rules Learning
- 2- Apriori
- 3- Apriori illustration
- 4- Apriori Application
- 5- Eclat
- 6- fim library



## 1- Association Rule Learning

### Concept and terminology

- The learning concerns identifying **associations** between data **attributes**.
- This association is expressed by : **associations rules** in the form:
  - If X then Y Or in the form:  $X \Rightarrow Y$
  - Where X and Y are subsets of attributes. These attributes are generally called: **items**. And the subsets of attributes are called: **itemsets**.
  - The data samples described by these attributes are called **transactions**.
- These rules are obtained by identifying the **frequent itemsets**.
- The rules permit to **predict** the presence of items knowing the existence of other ones.



## 1- Association Rule Learning

### Measures

- The following measures are used in the association rules identification process: (where  $X$  and  $Y$  are itemsets from the transactions set  $T$ , and form the rule  $X \Rightarrow Y$ )
- **Support\_count (X):** = frequency of  $X$  in  $T$  = Number of occurrences of  $X$  in  $T$
- **Support (X):**  $= \frac{\text{frequency of } X \text{ in } T}{\text{size of } T}$
- **Support (X,Y)**  $= \frac{\text{frequency of } X \text{ and } Y \text{ together in } T}{\text{size of } T} = \text{Support}(X \Rightarrow Y)$
- **Confidence(  $X \Rightarrow Y$ ):**  $= \frac{\text{frequency of } X \text{ and } Y \text{ together in } T}{\text{frequency of } X}$
- **Lift :**  $= \frac{\text{Support}(X, Y)}{\text{Support}(X) \times \text{Support}(Y)}$



## 1- Association Rule Learning

### Frequent itemsets and association rules

- An **itemset A** is **frequent** if :
  - $\text{Support}(A) \geq \text{minimum support threshold}$
- An association rule ( $X \Rightarrow Y$ ) is generated as follow:
  - Select **All** itemsets that are **frequent**
  - **Split each frequent** itemset in all possible subsets: X and Y that satisfy the condition:
    - $\text{Confidence}(X \Rightarrow Y) \geq \text{minimum confidence threshold}$
- In association rules **learning**, we apply **specific algorithms** on the **transactions dataset** (the training data) to identify the **frequent itemsets** in order to generate the **association rules**.
- Some of these algorithms:
  - **Apriori** ( Breath First Search )
  - **FP-growth** (Frequent Pattern Growth)
  - **Eclat** ( Depth First Search )



### Naive Apriori: frequent itemsets

- There is the “**naive**” approach, which describe the original **apriori** algorithm. Some improvements were introduced to this algorithm, which lead to different versions. We are going to describe the steps of the naive algorithm described in [\[Agrawal et al., 1994\]](#)
- The steps of frequent itemsets generation:
  - Define the  $L_1 = \{\text{frequent 1-itemset}\}$  ( $k$ -itemset = itemset with  $k$  items).
  - For ( $k=2, L_{k-1} \neq \emptyset, k++$ )
    - ➔  $C_k$  = from  $L_{k-1}$  generate-all candidates  $k$ -itemsets (using only frequent itemsets)
    - ➔ For all transactions  $t \in T$  ( $T$  is the set of all transactions)
      - Increment the count of all itemsets in  $C_k$  and contained in  $t$
    - ➔  $L_k$  = frequent itemsets in  $C_k$  ( itemsets with count  $\geq$  min support threshold)
  - The final frequent itemsets is  $\cup_k L_k$



### Naive apriori: rules generation

- The steps of rules generation are:
  - For all frequent itemsets  $l_k$ ,  $k \geq 2$ 
    - ➔ Generate all valid rules  $\bar{a} \rightarrow (l_k - \bar{a})$  for each  $\bar{a} \subset l_k$   
a valid rule is the one that have **confidence  $\geq$  min confidence threshold**
- To generate all valid rules  $\bar{a} \rightarrow (l_k - \bar{a})$  for each  $\bar{a} \subset l_k$ 
  - 1- Set  $a_m = l_k$
  - 2-  $A =$  all  $a_{m-1}$  itemsets that are subsets of  $a_m$
  - 3- For each  $a_{m-1} \in A$ 
    - ➔ Compute confidence of the rule  $r = (a_{m-1} \Rightarrow l_k - a_{m-1}) = \frac{\text{support}(l_k)}{\text{support}(a_{m-1})}$
    - ➔ If (confidence (r)  $\geq$  **min confidence threshold**) then
      - select r as a valid rule
      - If  $(m-1 > 1)$  set  $a_m = a_{m-1}$  and go to 2.



## 2- Apriori

### Remarks

- When we generate the  $C_k$  candidates, we eliminate all the ones created by subsets that are not frequent. We call it the **prune** step.
- As an improvement, we can consider only transactions that contain frequent itemsets
- The min support and min confidence thresholds must be chosen wisely:
  - A small threshold will lead to more iterations of the algorithm
  - A high threshold can eliminate rare items.





### 3- Apriori illustration

#### The data

- We will run the algorithm on the example cited in [Gollapudi, 2016] (after correction)
- We suppose we have the dataset T of transactions that represent the items bought together in each purchase.

T=

1	A, B,E
2	B, D
3	B, C
4	A, B , D
5	A, D
6	B, C
7	A, D
8	A, B, C, E
9	A, B,C

- Where each letter represents an item:
  - A = iPad
  - B = iPad case
  - C = iPad scratch guard
  - D = Apple care
  - E = iPhone
- The numbers in left column represent the TID: transaction identifier



### 3- Apriori: illustration

## Frequent itemsets

- We suppose that the **minimum support count = 2, (min support threshold (2/9))**

**L<sub>1</sub>**

Itemset	Support count $\geq 2$
A	6
B	7
C	4
D	4
E	2

**C<sub>2</sub>**

Itemset	Support count
A,B	4
A,C	2
A,D	3
A,E	2
B,C	4
B,D	2
B,E	2
C,D	0
C,E	1
D,E	0

**L<sub>2</sub>**

Itemset	Support count $\geq 2$
A,B	4
A,C	2
A,D	3
A,E	2
B,C	4
B,D	2
B,E	2

**T**

1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, D
6	B, C
7	A, D
8	A, B, C, E
9	A, B, C



### 3- Apriori: illustration

#### Frequent items sets: prune steps

$L_2$

Itemset	SC $\geq 2$
A,B	4
A,C	2
A,D	3
A,E	2
B,C	4
B,D	2
B,E	2

Not found in  $L_2$

$C_3$

Itemset	Subsets
A,B,C	AB, AC, BC
A,B,D	AB, AD, BD
A,B,E	AB, AE, BE
A,C,D	AC, AD, <b>CD</b>
A,C,E	AC, AE, <b>CE</b>
A,D,E	AD, AE, <b>DE</b>
B,C,D	BC, BD, BE
B,C,E	BC, BE, <b>CE</b>
B,D,E	BD, BE, <b>DE</b>

Prune

$C_3$

Itemset	Support count
A,B,C	2
A,B,D	1
A,B,E	2
B,C,D	0

Itemset	SC $\geq 2$
A,B,C	2
A,B,E	2

$L_3$

$L_4$

Itemset	SC
---------	----

$C_4$

Itemset	SC
---------	----

We stop here, because if we want to prune  $C_4$ , we will eliminate the generated subsets (ABCD, ABCE) since they contain subsets of 3 items that are not in  $L_3$



## 4- Apriori Application

### MI-xtend The data

- **mlxtend** library implements the **apriori** algorithm.
- But, before applying the algorithm on the previous example, we have to create the corresponding data.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

# the previous data
data = [["A", "B", "E"], ["B", "D"], ["B", "C"], ["A", "B", "D"], ["A", "D"], ["B", "C"], ["A", "D"], ["A", "B", "C"],

# encode the data, so we can apply the algorithm
TE = TransactionEncoder()
dataEnc = TE.fit(data).transform(data)
df = pd.DataFrame(dataEnc, columns=TE.columns_)
```

	A	B	C	D	E
0	True	True	False	False	True
1	False	True	False	True	False
2	False	True	True	False	False
3	True	True	False	True	False
4	True	False	False	True	False
5	False	True	True	False	False
6	True	False	False	True	False
7	True	True	True	False	True
8	True	True	True	False	False



## 4- Apriori Application

### MI-extend frequent itemsets

```
1 from mlxtend.frequent_patterns import apriori
2
3 frequent_itemsets= apriori(df, min support=0.22, use_colnames=True)
```

	support	itemsets		support	itemsets
0	0.666667	(A)	10	0.222222	(B, D)
1	0.777778	(B)	11	0.222222	(E, B)
2	0.444444	(C)	12	0.222222	(C, B, A)
3	0.444444	(D)	13	0.222222	(E, B, A)
4	0.222222	(E)			
5	0.444444	(B, A)			
6	0.222222	(C, A)			
7	0.333333	(D, A)			
8	0.222222	(E, A)			
9	0.444444	(C, B)			

The result is the union of all previous  $L_i$  we found ( $L_1 \cup L_2 \cup L_3$ )



## Mlxtend: generated rules

- we will generate the **association rules** corresponding to the found **frequent itemsets**

```
1 from mlxtend.frequent_patterns import association_rules
2
3 AR= association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
4 AR.iloc[:, [0,1,4,5,6]]
```

	antecedents	consequents	support	confidence	lift
0	(D)	(A)	0.333333	0.75	1.125000
1	(E)	(A)	0.222222	1.00	1.500000
2	(C)	(B)	0.444444	1.00	1.285714
3	(E)	(B)	0.222222	1.00	1.285714
4	(C, A)	(B)	0.222222	1.00	1.285714
5	(B, E)	(A)	0.222222	1.00	1.500000
6	(E, A)	(B)	0.222222	1.00	1.285714
7	(E)	(B, A)	0.222222	1.00	2.250000

Min  
confidence  
threshold

Correspond to  
the rule:  $E \rightarrow B, A$



### Concept

- It is based on the relationship between subset inclusion and support values.
- In this algorithm, an item is represented by the list of transactions it belongs to. They are the TidLists
- The support is computed from the intersection of these TidLists.
- If  $N$  is the size of  $T$  (the transactions dataset), then:

$$\text{support}(X, Y) = \frac{|Tid(X) \cap Tid(Y)|}{N} = \frac{\text{frequency of } X \text{ and } Y \text{ together}}{N}$$

- It also relies on the concept that:
  - If  $X \subseteq Y$ , and  $\text{support}(Y) = S \Rightarrow \text{support}(X) \geq S$
  - If  $Y \subseteq X$ , and  $\text{support}(X) \leq \text{min\_s} \Rightarrow \text{support}(Y) < \text{min\_s}$



## 5- Eclat

### Algorithm

- The algorithm is defined by a recursive function **eclat**
- A recursive function is a function that calls itself directly or indirectly. It stops when a certain condition is met.
- The steps are:
  - set  $p = \{\}$ ,  $Items = \{\text{all items}\}$
  - Call  $Eclat(P, Items)$
- Eclat (P,I) definition:
  - $F = \{\}$ ,  $C_{it} = \{\}$
  - If  $Items = \{\}$  return F
  - else
    - ➔ C = for each i in Items and not in P generate  $(P \cup i, i)$  tuples
    - ➔ Filter out C so it will contain only frequent  $P \cup i$  itemsets
    - ➔ For each  $(P \cup i, i)$  in c add i to  $C_{it}$
    - ➔ For each  $(X, i)$  in C:
      - $C_{it} = C_{it} - \{i\}$
      - $F = F \cup X \cup Eclat(X, C_{it})$
    - ➔ Return F





## 5- Eclat

### Illustration

- We will run the algorithm on the example cited in [Eclat] ( in the reference it is actually run for threshold=2 and not 3):
- $I = \{a, c, b, e, d, f\}$ ,  $\text{min\_support\_count\_threshold} = 3$
- $T = [[a, b, c], [a, c, d, e, f], [a, b, c], [d, e]]$
- $P = \{\}$  ,  $\text{Items} = \{a, b, c, d, e\}$
- $\text{Eclat}(P = \{\}, \text{Items} = \{a, b, c, d, e\})$  -----(1)
  - $\text{Eclat}(P = \{\}, \text{Items} = \{a, b, c, d, e\})$  (from 1)
    - ➔  $F = \{\}$ ;  $C_{it} = \{\}$
    - ➔  $C = \{(a, a), (b, b), (c, c), (d, d), (e, e), (f, f)\}$
    - ➔  $C = \{(a, a), (c, c)\}$ ,  $C_{it} = \{a, c\}$ 
      - 1)  $X = a$ ,  $i = a$
      - $C_{it} = \{c\}$
      - $F = \{a\} \cup \text{Eclat}(P = \{a\}, \text{Items} = \{c\})$  ----- (11)
      - $\text{Eclat}(P = \{a\}, \text{Items} = \{c\})$  (from 11)
        - $F = \{\}$ ,  $C_{it} = \{\}$
        - $C = \{(ac, c)\}$



### Illustration (suite)

- $C = \{(ac, c)\}, C_{it} = \{c\}$ 
  - $X = \{ac\}, i = c$
  - $Items = \{\}$ 
    - $F = \{ac\} \cup Eclat(P = \{ac\}, Items = \{\})$  ---- (111)
    - $Eclat(P = \{ac\}, Items = \{\})$  (from 111)
      - $F = \{\}, C_{it} = \{\}$
      - $Items = \{\},$  return F (return to 111)
  - $F = \{ac\}$
- Return F (return to ---- (11))
- $F = \{a, ac\}$
- 2)  $X = \{c\}, i = c$
- $C_{it} = \{a\}$
- $F = \{a, ac, c\} \cup Eclat(P = \{c\}, Items = \{a\})$  ----- (12)
  - $Eclat(P = \{c\}, Items = \{a\})$  (from 1)



## 5- Eclat

### Illustration (suite)

- $\text{Eclat}(P=\{c\}, \text{Items}=a)$  (from 12)
  - $F = \{\}, C_{it} = \{\}$
  - $C = \{(ca, a)\}$
  - $C_{it} = \{a\}$ 
    - $X = \{ca\}, i=a$
    - $C_{it} = \{\}$
    - $F = \{ca\} \cup \text{Eclat}(P=\{ca\}, \text{Items}=\{\})$ ---- (121)
    - $\text{Eclat}(P=\{ca\}, \text{Items}=\{\})$  (from 121)
      - $F = \{\}, C_{it} = \{\}$
      - $\text{Items} = \{\},$  return  $F$  (return to 121)
    - $F = \{ca\}$
  - Return  $F$  (return to ---- (12))
- $F = \{a, ac, c\}$ 
  - Return  $F$  (return to (1))
- Final  $F = \{a, ac, c\}$



### Library and data

- fim is a library comprised of a module that implements a set of functions dedicated to frequent itemset mining.
- The functions are related to the algorithm they implement.
- For example, the library implements “Eclat”, “apriori”, et “fpgrowth” functions.
- To have a list of all the the represented algorithms, take a look at its homepage ([PyFIM - Frequent Item Set Mining for Python](#)).
- To install the library, just use the **pip** command:
- Concerning the data, we do not need to do any transformation. So, we will use the transactions of the example, as we defined them (list of lists):

```
1 !pip install fim
```

```
T = [[ "a", "b", "c"], [ "a", "c", "d", "e", "f"], [ "a", "b", "c"], [ "d", "e"]]
```



## Eclat application

```
import fim as fim
from fim import eclat
fis = eclat(T, supp=75)
```

- Supp =75 means Support = 0.75 ( $\frac{3}{4}$ )
- By default it prints support\_count values

```
[(('c',), 3), (('a', 'c'), 3), (('a',), 3)]
```

“S” to print the support as fractions

```
1 fis_p = eclat(T, supp = 75, report = "s")
2 fis_p
```

```
[(('c',), 0.75), (('a', 'c'), 0.75), (('a',), 0.75)]
```

The same frequent itemsets we found earlier in the illustration



## Apriori application

- We will use the “apriori” function on the previous example we saw in apriori section.

```
from fim import apriori  
fis_a = apriori(data, supp=22, report="s")
```

Support\_count = 2 is  
equivalent to  
support = 2/9

```
[(('E', 'A', 'B'), 0.2222222222222222),  
 (('E', 'A'), 0.2222222222222222),  
 (('E', 'B'), 0.2222222222222222),  
 (('E', ), 0.2222222222222222),  
 (('D', ), 0.4444444444444444),  
 (('D', 'A'), 0.3333333333333333),  
 (('D', 'B'), 0.2222222222222222),  
 (('C', 'B'), 0.4444444444444444),  
 (('C', ), 0.4444444444444444),  
 (('C', 'A', 'B'), 0.2222222222222222),  
 (('C', 'A'), 0.2222222222222222),  
 (('A', ), 0.6666666666666666),  
 (('A', 'B'), 0.4444444444444444),  
 (('B', ), 0.7777777777777778)]
```



# References

- [Agrawal et al., 1994] Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB, volume 1215, pages 487–499.
- [Alexey, 2014] Alexey, G. (2014). Eclat. On-line at <http://mlwiki.org/index.php/Eclat>. Accessed on 30-12-2018.
- [Gollapudi, 2016] Gollapudi, S. (2016). Practical Machine Learning. Community experience distilled. Packt Publishing.
- [Sebastian, ] Sebastian, R. mlxtend's documentation. On-line at <http://rasbt.github.io/mlxtend/>. Accessed on 30-12-2018.



# Thank you!

FOR ALL YOUR TIME