

Unsupervised Learning: Association Rule Learning

AAA-Python Edition



Plan

- 1- Association Rules Learning
- 2- Apriori
- 3- apriori optimization
- 4- apriori implementation
- 5- apriori implementation part 2
- 6- apriori





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 ==> 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.



arning

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 $\mathbf{X} ==> \mathbf{Y}$)
- **Support_count (X):** = frequency of X in T = Number of occurences of X in T
- Support (X): = $\frac{\text{frequency of X in T}}{\text{size of T}}$
- Support (X,Y) = $=\frac{\text{frequency of X and Y together in T}}{\text{size of T}} = Support(X ==> Y)$
- Confidence(X==>Y): = $\frac{\text{frequency of X an Y together in T}}{\text{frequency of X}}$
- Lift: $=\frac{\text{Support}(X,Y)}{Support(X) \times Support(Y)}$





Frequent itemsets and association rules

- An itemset A is frequent if :
 - > Support(A) >= minimum support threshold
- An association rule (X ==> Y) is generated as follow:
 - Select All itemsets that are frequents
 - Split each frequent itemset in all possible subsets:X and Y that satisfy the condition:
 - → Confidence (X ==Y) >= 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)

[By Amina Delali]



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 ={frequent 1-itemset} (k-itemset = itemset with k items).
 - For $(k=2, L_{k-1}\neq\emptyset, k++)$
 - \rightarrow C_k = from L_{k-1} generate-all candidates k-itemsets (using only frequent itemsets)
 - \rightarrow 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
 - \rightarrow L_k = frequent itemsets in C_k (itemsets with count >= min support threshold)
 - \rightarrow 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>=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** >= **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 = I_k
 - > 2- A = all a_{m-1} itemsets that are subsets of a_m
 - \rightarrow 3- For each $a_{m-1} \in A$
 - → Compute confidence of the rule $r = (a_{m-1} = > l_k a_{m-1}) = = support (l_k) / support(a_{m-1})$
 - → If (confidence (r) >= min confidence threshold) then
 - select r as a valid rule
 - If (m-1 > 1) set $a_m = a_{m-1}$ and go to 2.



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: Frequent itemsets

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.

 $\mathsf{T} =$

- 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

A, B,E
B, D
B, C
A, B , D
A, D
B, C
A, D
A, B, C, E
A, B,C



[By Amina Delali]



Frequent itemsets

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

Itemset	Support count >= 2
А	6
В	7
С	4
D	4
Е	2

	2			
Items	et	Supp	ort t >= 2	
A,B		4		
A,C		2		
A,D		3		
A,E		2		
В,С		4		
B,D		2		
B.F		2		

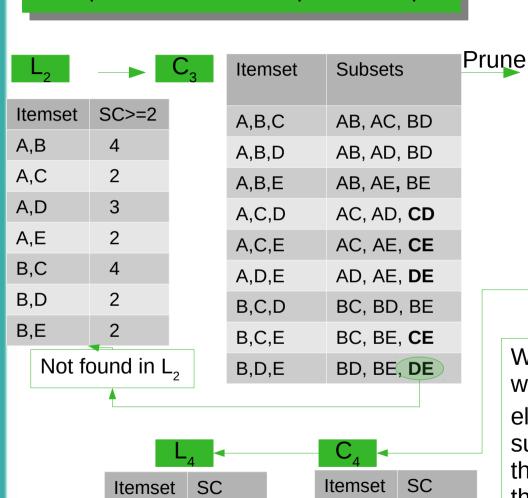
Itemset	Support count
A,B	4
A,C	2
A,D	3
A,E	2
В,С	4
B,D	2
B,E	2
C,D	0
C,E	1
D,E	0

1	A, B,E
2	B, D
3	В, С
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

[By Amina Delali]

Frequent items sets: prune steps



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

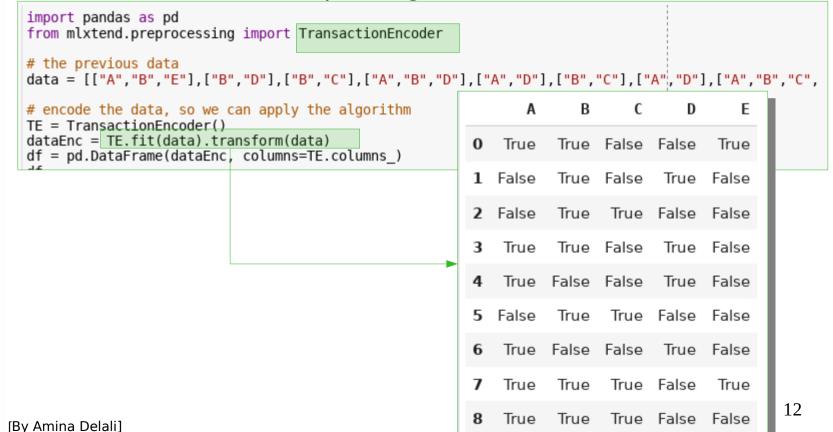
	Itemset	SC>=2
	A,B,C	2
L_3	A,B,E	2
	4	

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



MI-xtend The data

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







MI-extend frequent itemsets

1 from mlxtend.frequent_patterns import apriori

(D, A)

(E, A)

(C, B)

3 frequent itemsets= apriori(df, min support=0.22, use colnames=True)

	itemsets	support	
10	(A)	0.666667	0
11	(B)	0.777778	1
12	(C)	0.44444	2
13	(D)	0.44444	3
	(E)	0.222222	4
	(B, A)	0.44444	5
	(C, A)	0.22222	6

s		support	itemsets
()	10	0.222222	(B, D)
3)	11	0.222222	(E, B)
()	12	0.222222	(C, B, A)
)	13	0.222222	(E, B, A)

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

0.333333

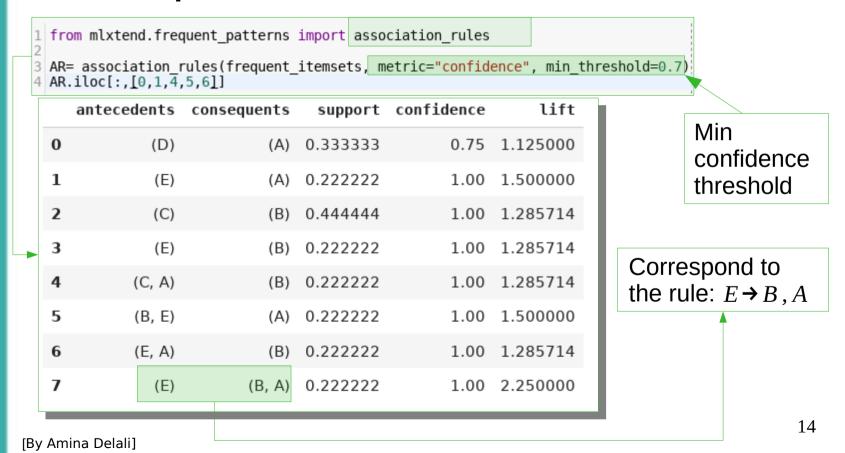
0.222222

0.444444



Mixtend: generated rules

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





Concept

- It 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:

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

- It also relies on the concept that:
 - \succ If $X \subseteq Y$, and support $(Y) = S \Rightarrow$ support $(X) \ge S$
 - ightharpoonup If $Y \subseteq X$, and $support(X) \le min_s \Rightarrow support(Y) < min_s$



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 ={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 U i, i) tuples
 - → Filter out C so it will contain only frequent P U i itemsets
 - → For each (P U i, i) in c add i to C_i
 - → For each (X, i) in C:
 - $C_{it} = C_{it} \{i\}$
 - F = F U X U Eclat(X, C_{it})
 - Return F





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}, min support count threshold= 3
- T = [[a,b,c],[a,c,d,e,f], [a,b,c],[d,e]]
- P = {} , Items = {a,b,c,d,e}
- Eclat(P= {}, Items={a,b,c,d,e}) -----(1)
 - > Eclat(P= {}, Items={a,b,c,d,e}) (from 1)
 - → F= {}; C_{i+}={}
 - → C= {(a,a),(b,b),(c,c),(d,d),(e,e),(f,f)}
 - $\rightarrow C = \{(a,a),(c,c)\}, C_{i} = \{a,c\}$
 - 1) X=a, i= a
 - $C_{i+} = \{c\}$
 - F = {a} U Eclat(P={a}, Items={c}) ----- (11)
 - Eclat(P={a}, Items= {c}) (from 11)
 - $F = \{\}, C_{i+} = \{\}$
 - $C = \{(ac,c)\}$



Illustration (suite)

- $C = \{(ac,c)\}, C_{i+} = \{c\}$
 - X={ac}, i= c
 - Items={}
 - F = {ac } U 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_{i+}={a}
- F = {a,ac,c} U Eclat(P={c}, Items={a}) ----- (12)
 - Eclat(P={c}, Items= {a}) (from 1)



Illustration (suite)

- Eclat(P={c}, Items= a}) (from 12)
 - F= {},C_{it=}{}
 - $C = \{(ca,a)\}$
 - $C_{i+} = \{a\}$
 - X= {ca} ,i=a
 - C_{i+}= {}
 - $F = \{ca\} \cup Eclat(P = \{ca\}, Items = \{\}) ---- (121)$
 - Eclat (P={ca}, Items={}) (from 121)
 - $F= \{\}, C_{it} = \{\}$
 - 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):

```
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(%)
- By default it prints support coun t 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")
                                 [(('E', 'A', 'B'), 0.2222222222222),
                                  (('E', 'A'), 0.2222222222222),
                                  (('E', 'B'), 0.22222222222222).
  Support cou
                                  (('E',), 0.2222222222222),
  nt = 2 is
                                  (('D',), 0.444444444444444),
  equivalent to
                                  (('D', 'A'), 0.33333333333333333),
                                  (('D', 'B'), 0.2222222222222),
  support = 2/9
                                  (('C', 'B'), 0.444444444444444),
                                  (('C',), 0.444444444444444),
                                  (('C', 'A', 'B'), 0.2222222222222),
                                  (('C', 'A'), 0.2222222222222),
                                  (('A',), 0.66666666666666),
                                  (('A', 'B'), 0.44444444444444).
                                  (('B',), 0.77777777777778)]
```



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