Clustering and Association Rule Learning

AI/ML Foundation Course with Python

Association Rule Learning

- A data mining tool used extensively by data mining and database community.
- Assumes all data are categorical
- Does not work well with numeric data
- Popularly used in Market Basket Analysis by e-commerce giants to find how items purchased by customers are related.

Association Rule Learning

- Support : Support is the basic probability of an event to occur. If we have an event to buy product A, Support(A) is the number of transactions which includes A divided by total number of transactions.

- Confidence: The confidence of an event is the conditional probability of the occurrence; the chances of A happening given B has already happened.

Association Rule Learning

Transaction	Item 1	Item 2	Item 3
1	Milk	Sugar	Coffee Powder
2	Milk	Sugar	Coffee Powder
3	Milk	Sugar	Coffee Powder
4	Milk	Sugar	
5	Milk	Sugar	

- Rule 1 : If Milk is purchased, then Sugar is also purchased.
- Rule 2: If Sugar is purchased, then Milk is also purchased.
- Rule 3: If Milk and Sugar are purchased, Then Coffee powder is also purchased

The model: data

- $I = \{i1, i2, \ldots, im\}$: a set of *items*.
- Transaction t :
  - t a set of items, and t I.
- Transaction Database T: a set of transactions $T = \{t1, t2, \ldots, tn\}$.

Transaction data: supermarket data

- Market basket transactions:

  t1: {bread, cheese, milk}

  t2: {apple, eggs, salt, yogurt}

  … …

  tn: {biscuit, eggs, milk}

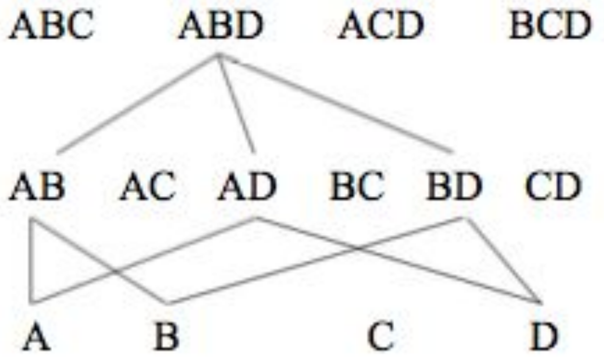
- Concepts:
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
- Completeness: find all rules.
- No target item(s) on the right-hand-side
- Mining with data on hard disk (not in memory)

Step 1: Mining all frequent itemsets

- A frequent *itemset* is an itemset whose support is ≥ minsup.
- Key idea: The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets



Details: the algorithm

```
Algorithm Apriori(T)
  C1  init-pass(T);
  F1  {f | f  C1, f.count/n  minsup};   // n: no. of transactions in T
  for (k = 2; Fk-1  ; k++) do
     Ck  candidate-gen(Fk-1);
     for each transaction t  T do
        for each candidate c  Ck do
        if c is contained in t then
          c.count++;
        end
     end
       Fk  {c  Ck | c.count/n  minsup}
  end
return F  k Fk;
```

Candidate-gen function

```
Function candidate-gen(Fk-1)
  Ck ;
  forall f1, f2  Fk-1
     with f1 = {i1, … , ik-2, ik-1}
     and f2 = {i1, … , ik-2, i'k-1}
     and ik-1 < i'k-1 do
     c  {i1, …, ik-1, i'k-1};        // join f1 and f2
     Ck  Ck  {c};
     for each (k-1)-subset s of c do
     if (s  Fk-1) then
        delete c from Ck;        // prune
     end
  end
  return Ck;
```

Step 2: Generating rules from frequent itemsets

- Frequent itemsets association rules
- One more step is needed to generate association rules
- For each frequent itemset *X*,

  For each proper nonempty subset *A* of *X*,

  - Let *B* = X - *A*
  - A B is an association rule if
  - Confidence(A B) ≥ minconf,

    support(A B) = support(AB) = support(X)

    confidence(A B) = support(A B) / support(A)

Apriori Algorithm

- Algorithm for Mining frequent itemsets for Boolean association rules

- Apriori uses bottom up approach .

- Frequent subsets are extended one item at a time which is known as candidate generation and groups are tested against the data

Finding All Frequent Itemsets

- Apriori Algorithm
- Candidate Generation
- Horizontal Search
- Support and Confidence both needed

- FP Growth
- Candidate Generation
- Vertical Search/DFS
- Only Support needed

Minsup and minconf

Minimum Support =50 %

(50/100)*4 = 2

Minimum Confidence = 50%

TRANSACTIONS	ITEMSET
T1	A,B,C
T2	A,C
T3	A,D
T4	B,E,F

Apriori Algorithm : Step 1- Candidate Generation

C1

Minimum Support =50 %

(50/100)*4 = 2

Minimum Confidence = 50%

ITEMS	SUPPORT
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

Apriori Algorithm : C1 (Candidate 1)

Minimum Support =50 %

(50/100)*4 = 2

ITEMS	SUPPORT
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

Apriori Algorithm : L1 (Frequent Itemset)

Minimum Support =50 %

(50/100)*4 = 2

ITEMS	SUPPORT
{A}	3
{B}	2
{C}	2

Apriori Algorithm : C1

Minimum Support =50 %

(50/100)*4 = 2

ITEMS *(sets of 2)*	**SUPPORT**
{A,B}	1
{B,C}	1
{C,A}	2

Apriori Algorithm : C2

Minimum Support =50 %

(50/100)*4 = 2

ITEMS *(sets of 2)*	**SUPPORT**
{A,B}	1
{B,C}	1
{C,A}	2

Apriori Algorithm : L2

ITEMS *(sets of 2)*	**SUPPORT**
{C,A}	2

Minimum Support =50 %

(50/100)*4 = 2

Apriori Algorithm : Step 2 - Generating Association Rules

ITEMS *(sets of 2)*	SUPPORT
{C,A}	2

TRANSACTIONS	ITEMSET
T1	A,B,C
T2	A,C
T3	A,D
T4	B,E,F

Association Rules	Support	Confidence	Confidence %
A → C	2	2/3	66 %
C → A	2	2/2	100 %

Confidence (A → C) = Support (A → C) /Occurrence of A

Since both entries have min conf > 50%,
So, Final Rule is : A → C and C → A

Example : Design association rules using Apriori Algo

T_ID	ITEMSETS
T_1000	M,O,N,K,E,Y
T_1001	D,O,N,K,E,Y
T_1002	M,A,K,E
T_1003	M,U,C,K,Y
T_1004	C,O,K,E

Database has 5 transactions

Minsup =60%

Minconf = 80%

Example : CANDIDATE C1

ITEMSET	SUPPORT
{M}	3
{O}	3
{N}	2
{K}	5
{E}	4
{Y}	3
{D}	1
{A}	1
{U}	1
{C}	2

CALCULATING SUPPORT FOR EACH ITEMSET

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : L1

ITEMSET	SUPPORT
{M}	3
{O}	3
{N}	2
{K}	5
{E}	4
{Y}	3
{D}	1
{A}	1
{U}	1
{C}	2

REJECTING SUPPORT LESS THAN MIINSUP

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : L1 (New transaction table)

ITEMSET	SUPPORT
{M}	3
{O}	3
{K}	5
{E}	4
{Y}	3

REJECTING SUPPORT LESS THAN MIINSUP

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : CANDIDATE C2

ITEMSET	SUPPORT
{M, O}	1
{M,K}	3
{M,E}	2
{M,Y}	2
{O,K}	3
{O,E}	3
{O,Y}	2
{K,E}	4
{K,Y}	3
{E,Y}	2

CALCULATING SUPPORT FOR EACH ITEMSET

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : L2

ITEMSET	SUPPORT
{M, O}	1
{M,K}	3
{M,E}	2
{M.Y}	2
{O,K}	3
{O,E}	3
{O,Y}	2
{K,E}	4
{K,Y}	3
{E,Y}	2

REJECTING TABLE

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : L2 (New Transaction Table for C3)

ITEMSET	SUPPORT
{M,K}	3
{O,K}	3
{O,E}	3
{K,E}	4
{K,Y}	3

REJECTING TABLE

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : CANDIDATE C3

ITEMSET	SUPPORT
{M, O, K}	1
{M,K, E}	2
{M,K,Y}	2
{O,K,E}	3
{O,K,Y}	2
{K,E,Y}	2

CALCULATING SUPPORT FOR EACH ITEMSET

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : L3

ITEMSET	SUPPORT
{M, O, K}	1
{M,K, E}	2
{M,K,Y}	2
{O,K,E}	3
{O,K,Y}	2
{K,E,Y}	2

Rejecting...

Minsup =60%
60/100*5 = 3

Minconf = 80%

Example : L3

ITEMSET	SUPPORT
{O,K,E}	3

Rejecting...

Minsup =60%
60/100*5 = 3

Minconf = 80%

- **Association rules**

Association Rules	Support	Confidence	Confidence %
$O \rightarrow K$^E	3	3/3 = 1	100
$K \rightarrow O$^E	3	3/5 = 0.6	60
$E \rightarrow K$^O	3	3 / 4 = 0.75	75

Example : L3

ITEMSET	SUPPORT
{O,K,E}	3

Rejecting...

Minsup =60%
60/100*5 = 3

Minconf = 80%

Accepted Rule : O → K^E

- Association rules

Association Rules	Support	Confidence	Confidence %
O → K^E	3	3/3 = 1	100
K → O^E	3	3/5 = 0.6	60
E → K^O	3	3 / 4 = 0.75	75

ECLAT
ASSOCIATION RULE

Mining Association Rules

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support $\geq$ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset,
- Frequent itemset generation is computationally **expensive**

ECLAT Algorithm

- **Equivalence Class Clustering and bottom up Lattice Traversal- ECLAT**
- Method for Frequent Itemset Generation
- Searches in a DFS manner.
- Represent the data in **vertical format.**

To Improve the Efficiency of Apriori : (Scalable Algorithms)

- ✓ FPGrowth
- ✓ ECLAT
- ✓ Mining Close Frequent Patterns and Maxpatterns

Eclat: algorithm

1. Get tidlist for each item (DB scan)
2. Tidlist of {a} is exactly the list of transactions containing {a}
3. Intersect tidlist of {a} with the tidlists of all other items, resulting in tidlists of {a,b}, {a,c}, {a,d}, …
   = {a}-conditional database (if {a} removed)
4. Repeat from 1 on {a}-conditional database
5. Repeat for all other items

- Both Apriori and FP-growth use **horizontal data format**
- Alternatively data can also be represented in **vertical format**

TID	Items
1	Bread,Butter,Jam
2	Butter,Coke
3	Butter,Milk
4	Bread,Butter,Coke
5	Bread,Milk
6	Butter,Milk
7	Bread,Milk
8	Bread,Butter,Milk,Jam
9	Bread,Butter,Milk

Item Set	TID set
Bread	1,4,5,7,8,9
Butter	1,2,3,4,6,8,9
Milk	3,5,6,7,8,9
Coke	2,4
Jam	1,8

Frequent 1-itemsets

min_sup=2

Item Set	TID Set
Bread	1,4,5,7,8,9
Butter	1,2,3,4,6,8,9
Milk	3,5,6,7,8,9
Coke	2,4
Jam	1,8

Frequent 2-itemsets

Item Set	TID set
{Bread,Butter}	1,4,8,9
{Bread,Milk}	5,7,8,9
{Bread,Coke}	4
{Bread,Jam}	1,8
{Butter,Milk}	3,6,8,9
{Butter,Coke}	2,4
{Butter,Jam}	1,8
{Milk,Jam}	8

Frequent 3-itemsets

Item Set	TID Set
{Bread,Butter,Milk}	8,9
{Bread,Butter,Jam}	1,8

- This process repeats, with k incremented by 1 each time, until no frequent items or no candidate itemsets can be found.

ADVANTAGES

- Depth-first search reduces memory requirements
- Usually (considerably) faster than Apriori
- No need to scan the database to find the support of (k+1) itemsets, for k>=1

DISADVANTAGES

The TID-sets can be quite long, hence expensive to manipulate


THANK YOU