SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution
Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF MCA

19CAT703 – MACHINE LEARNING

II YEAR III SEM

UNIT IV – TREE AND RULE MODELS

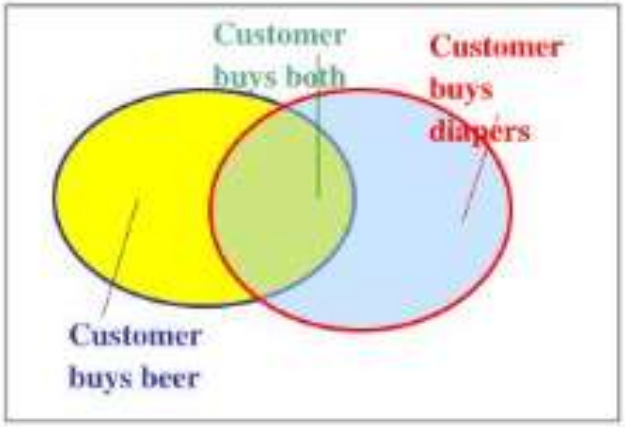
TOPIC 38 – Association Rule Mining

Association - Introduction

- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
  - Frequent pattern: pattern (set of items, sequence, etc.) that occurs frequently in a database
- Motivation: finding regularities in data
  - What products were often purchased together? — Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

Association Rule Mining-Introduction

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F



- Itemset $X=\{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with min confidence and support
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*.

Let min_support = 50%, min_conf = 50%:

$A \rightarrow C$ (50%, 66.7%)

$C \rightarrow A$ (50%, 100%)

Example

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 50%
Min. confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule $A \Rightarrow C$:

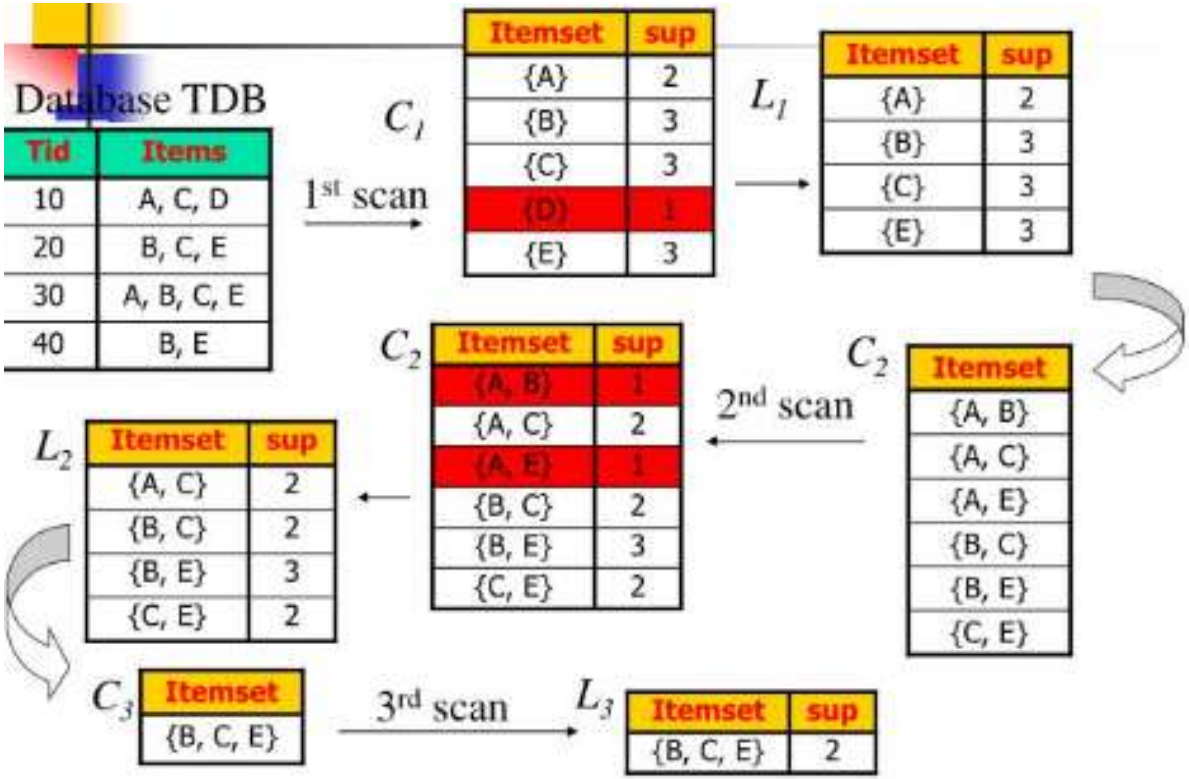
support = support($\{A\} \cup \{C\}$) = 50%

confidence = support($\{A\} \cup \{C\}$)/support($\{A\}$) = 66.6%

Apriori algorithm

- Any subset of a frequent itemset must be frequent
  - if **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - generate length (k+1) candidate itemsets from length k frequent itemsets, and
  - test the candidates against DB
- The performance studies show its efficiency and scalability

Example:Market Basket Analysis



Apriori: Steps

- Pseudo-code:

  C_k: Candidate itemset of size k
  L_k : frequent itemset of size k

  L_1 = {frequent items};
  for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
  increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
  return $\cup_k L_k$;

Meta Learning

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4={*abcd*}

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  from $L_{k-1}\ p, L_{k-1}\ q$

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning

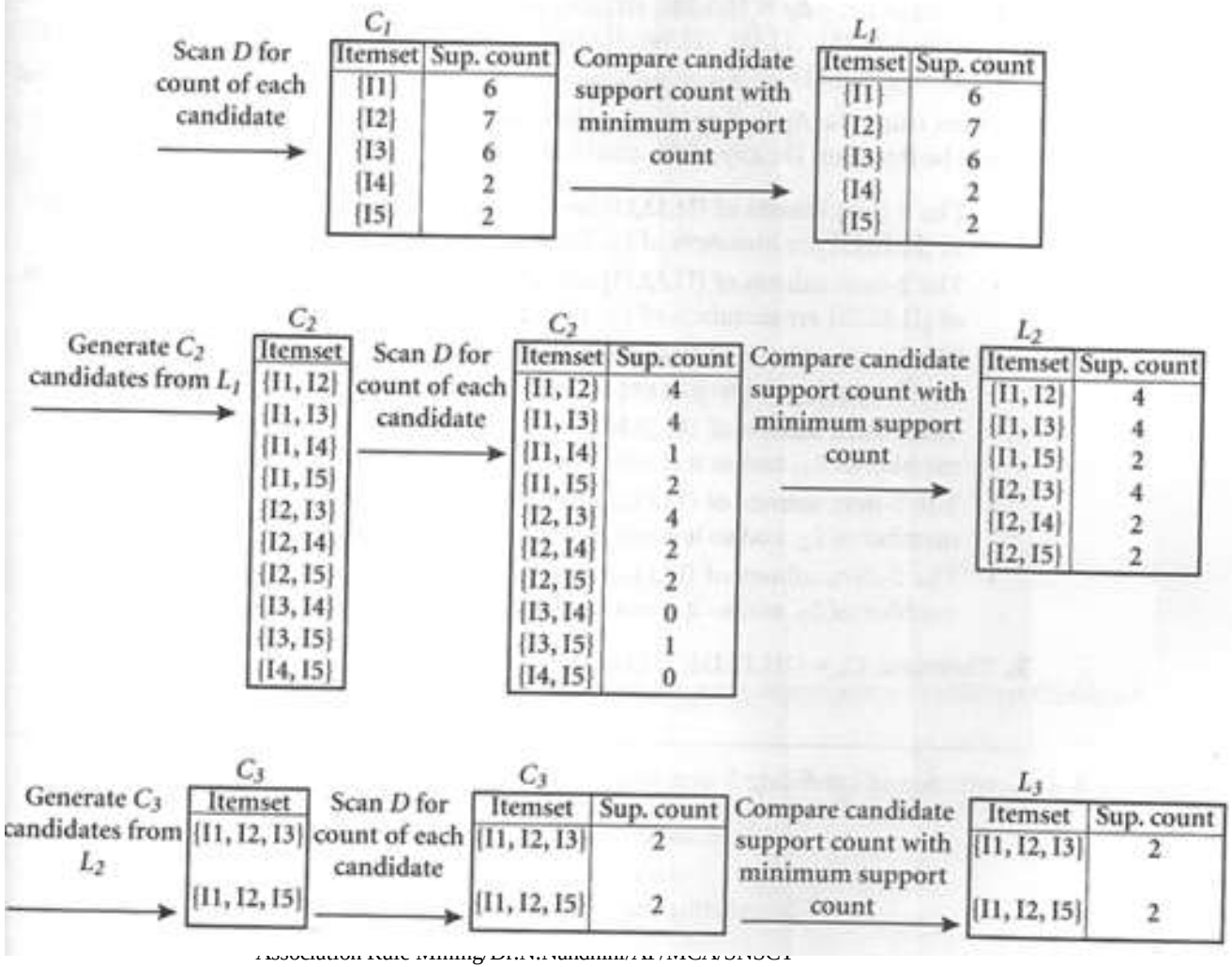
  forall ***itemsets c in*** C_k do

  forall ***(k-1)-subsets s of c*** do

  if (*s is not in* L_{k-1}) **then delete** *c* **from** C_k

Example

TID	List of Items ID
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3


Scan D for count of each candidate
C1
Itemset | Sup. count
{I1} 6
{I2} 7
{I3} 6
{I4} 2
{I5} 2
Compare candidate support count with minimum support count
L1
Itemset | Sup. count
{I1} 6
{I2} 7
{I3} 6
{I4} 2
{I5} 2
Generate C2 candidates from L1
C2
Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}
Scan D for count of each candidate
C2
Itemset | Sup. count
{I1, I2} 4
{I1, I3} 4
{I1, I4} 1
{I1, I5} 2
{I2, I3} 4
{I2, I4} 2
{I2, I5} 2
{I3, I4} 0
{I3, I5} 1
{I4, I5} 0
Compare candidate support count with minimum support count
L2
Itemset | Sup. count
{I1, I2} 4
{I1, I3} 4
{I1, I5} 2
{I2, I3} 4
{I2, I4} 2
{I2, I5} 2
Generate C3 candidates from L2
C3
Itemset
{I1, I2, I3}
{I1, I2, I5}
Scan D for count of each candidate
C3
Itemset | Sup. count
{I1, I2, I3} 2
{I1, I2, I5} 2
Compare candidate support count with minimum support count
L3
Itemset | Sup. count
{I1, I2, I3} 2
{I1, I2, I5} 2

REFERENCE

1. https://medium.com/ml-research-lab/
2. https://en.wikipedia.org/wiki
3. https://medium.com/abacus-ai/a-beginners-guide-to-meta-learning-73bb027007a