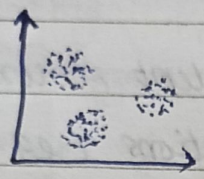
11.08.'24 MODULE 3 AGAIN

Clustering

K-means clustering algorithm

We ~~need~~ divide data into groups - each group is a cluster.



} 3 clusters

spherical types of clusters - identified by K-means

each cluster represented by a data object - cluster representative / cluster ~~centre~~ centre

For K-means,

Representative - mean value of data objects in the cluster

K - total number of clusters

K = 2 ⇝ 2 clusters ; K = 3 ⇝ 3 clusters

should be given as input to the algo.

K ⟶ range : 1 to n-1

Here, 1 to 3.

Let K be 2. Data objects : (1,1), (2,1), (4,3), (5,4)

Iteration 1 :

	(1,1) Centroid 1	(2,1) centroid 2	cluster
(1,1)	$\sqrt{(1-1)^2+(1-1)^2}$ = 0	$\sqrt{(1-2)^2+(1-1)^2}$ = 1	1
(2,1)	$\sqrt{(2-1)^2+(1-1)^2}$ = 1	$\sqrt{(2-2)^2+(1-1)^2}$ = 0	2
(4,3)	$\sqrt{(4-1)^2+(3-1)^2}$ = $\sqrt{13}$ = 3.6	$\sqrt{(4-2)^2+(3-1)^2}$ = $\sqrt{8}$ = 2.8	2
(5,4)	$\sqrt{(5-1)^2+(4-1)^2}$ = $\sqrt{16+9}$ = 5	$\sqrt{(5-2)^2+(4-1)^2}$ = $\sqrt{9+9}$ = 4.2	2

Find proximity using Euclidean Distance Measure.

Newer one - cluster (in each row)

cluster 1 = {(1,1)}

cluster 2 = {(2,1), (4,3), (5,4)}

Iteration 2: $\left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = (3.67, 2.67)$

	Centroid 1 (1,1)	Centroid 2	cluster
(1,1)	0	$\sqrt{(1-3.6)^2+(1-2.6)^2} = 3.05$	1
(2,1)	1	$\sqrt{(2-3.6)^2+(1-2.6)^2} = 2.26$	1
(4,3)	3.6	$\sqrt{(4-3.6)^2+(3-2.6)^2} = 0.57$	2
(5,4)	5	$\sqrt{(5-3.6)^2+(4-2.6)^2} = 1.98$	2

Since only one element in cluster 1 keep it as that. Calculate mean of the other cluster and take that as centroid. If more than one element in one cluster, take the mean.

Cluster 1 = {(1,1), (2,1)}

Cluster 2 = {(4,3), (5,4)}

Centroid 1 = (1,1) Centroid 2 (3.6, 2.6)

If in the next iteration, the cluster values change, we might have to continue further.

Iteration 3:

	Centroid 1 (1.5, 1)	Centroid 2 (4.5, 3.5)	cluster
(1,1)	$\sqrt{(1.5-1)^2+(1-1)^2} = 0.5$	$\sqrt{(4.5-1)^2+(3.5-1)^2} = 4.3$	1
(2,1)	$\sqrt{(2-1.5)^2+(1-1)^2} = 0.5$	$\sqrt{(4.5-2)^2+(3.5-1)^2} = 3.5$	1
(4,3)	$\sqrt{(4-1.5)^2+(3-1)^2} = 3.2$	$\sqrt{(4.5-4)^2+(3.5-3)^2} = 0.7$	2
(5,4)	$\sqrt{(5-1.5)^2+(4-1)^2} = 4.6$	$\sqrt{(4.5-5)^2+(3.5-4)^2} = 0.7$	2

Cluster 1 and Cluster 2 remains the same.

Centroid of cluster 1 = (1.5, 1)

Centroid of cluster 2 = (4.5, 3.5)

Since Iteration 2 and 3 have the same output, we can stop.

(otherwise, continue with next)

Partitioning clustering Algorithm
Iterative clustering Algorithm

In k-means, iterative relocation of data objects happens.
Disadvantage:

- the value of k is decided by users
- sensitive to outliers in dataset

~~Algorithm for k-means:~~

~~Repeat~~

~~(re) assign each data object to the cluster to which the object is most similar~~

Algorithm for K-means:

Input: K : no of clusters
D : no of data objects

Output: k clusters

Arbitarily choose K number of data objects from D as initial cluster centers.

Repeat :

(re) assign each data object to the cluster to which the object is most similar

update the cluster means

until no change in the clusters

Q. Data objects of dataset D:

(1,2), (6,4), (4,9), (2,10), (2,5), (8,4), (5,8), (7,5)

Find the clusters with $k=3$.

Ans. Iteration 1

	(1,2) Centroid 1	(5,8) Centroid 2	(8,4) Centroid 3	Cluster
(1,2)	0	7.21	7.28	1
(6,4)	5.39	4.12	3	3
(4,9)	7.62	1.41	6.40	2
(2,10)	8.06	3.61	8.49	2
(2,5)	3.16	4.24	6.08	1
(8,4)	7.28	5	0	3
(5,8)	7.21	0	5	2
(7,5)	6.71	3.61	1.41	3

cluster 1 = {(1,2), (2,5)}

cluster 2 = {(4,9), (2,10), (5,8)}

cluster 3 = {(6,4), (8,4), (7,5)}

Iteration 2

Q. Dataset: 1, 2, 3, 20, 21, 22

Ans. Iteration 1

	Medoid 1 (1)	Medoid 2 (2)	Cluster (min of other two col)
1	0	1	0
2	1	0	0
3	2	1	1
20	19	18	18
21	20	19	19
22	21	20	20

sum

Cost: 58

	Medoid [1,3]	min	Medoid [1,20]	min	Medoid [1,21]	min	Medoid [1,22]	min
1	2	1	19	1	20	1	21	1
2	1	0	18	0	19	0	20	0
3	0	0	17	1	18	1	19	1
20	17	17	0	0	1	1	2	2
21	18	18	1	1	0	0	1	1
22	19	19	2	2	1	1	0	0
new cost		55		5		4		5
new - old		-3		-53		-54		-53

replace 1 with 21

-54 is the suitable cost. Therefore replace 1 with 21.

	Medoid (2,3)		Medoid (2,20)		Medoid (2,21)		Medoid (2,22)	
1	2	0	19	0	20	0	21	0
2	1	1	18	1	19	1	20	1
3	0	0	17	2	18	2	19	2
20	17	17	0	0	1	1	2	2
21	18	18	1	1	0	0	1	1
22	19	19	2	2	1	1	0	0
		55		6		5		6
		-3		-52		-53		-52

new medoid (2,21)

disadvantage : inputs the value of k.

PAM is not suitable for high dimensional data.

Hierarchical Clustering

hierarchical decomposition of DB

2 types :

bottom up

top down

Iterative splitting of DB until some termination condition is satisfied.

Density based clustering Algorithm

DBSCAN → Density Based Spatial clustering of Applications with Noise

advantage : can find arbitarily shaped clusters

requires two parameters :

1. eps → radius - defines the neighbourhood around a data point.
2. MinPts. - minimum number of neighbours.
   minimum number of points in a circle with epsilon radius.

Algorithm DBSCAN:

Input :

D : a data set containing

ϵ : the redius parameter, and

MinPts : the neighbourhood density threshold

Output : A set of density based clusters

(1) mark all objects as unvisited.
(2) do
(3) randomly select an unvisited object p;
(4) mark p as visited;
(5) if the ϵ-neighborhood of p has at least MinPts objects
(6) create a new cluster C, and add p to C;
(7) let N be the set of objects in the ϵ-neighborhood of p;
(8) for each point p' in N
(9) if p' is unvisited
(10) mark p' as visited;
(11) if the ϵ-neighborhood of p' has at least MinPts points, add those points to N;
(12) if p' is not yet a member of any cluster, add p' to C;
(13) end for
(14) output C;

core object
iterative process
noise or outlier
ϵ must be ~~chosen~~ chosen carefully.

~~In DBSCAN,~~

~~- core point~~
~~- data points~~

In DBSCAN Algorithm, three types of data objects.

- Core Point
- Border Point
- noise / outlier

Direct Density Reachable (DDR): A point p is DDR from point q if q is the core point and p is the neighbour of q.

Density Reachable (DR) :- Two points are DR, if there is a is a chain of DDR points that link these two points.

ROCK - Robust Clustering using links

hierarchical clustering algorithm that analyse the concept of links (the number of common neighbours among two objects) for data with categorical attributes.

Hierarchical Clustering Algo → Agglomerative (eg: ROCK)
→ ~~Clustering~~ Divisive

Ways to find the proximity between clusters:

- single linkage (Euclidean distance between the closer pair of objects)
- complete linkage (distance between the two clusters = euclidean distance of the farthest pair of objects.
- average linkage (distance is defined as the average of distances between all pairs of objects r and s, where r and s belong to different clusters.)

Find the proximity using single linkage.

(15, 11, 5, 10) Proximity = 0

(5, 1, 6, 2) $\sqrt{(5-5)^2}$

	5	1	6	2
15	10	14	9	13
11	6	10	5	9
5	(0)	4	1	3
10	3	9	4	8

ROCK algorithm

1. obtaining a random sample of the data.
2. Performing clustering on the sampled data using link agglomerative ~~cluster~~ approach where the goodness measure helps to identify the best pair of clusters to be merged.
3. Assign the remaining data points of the disk to the generated clusters.

Major definitions

- Similarity function
- Neighbors
- Links
- criterion function
- goodness measure

26.03.24 MODULE 4

TID	List of item_IDs
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

Association Rule Mining

(Functional dependency vs. Association Rules)

↓

changes not FDs

3 Types of Frequent Pattern →
- Frequent Itemset ~ set of items which are frequently purchased by the customer
  Market Basket Analysis
- Frequent Structured Pattern
  (subgraph that occurs frequently in a graph)
- Frequent Sequential Pattern
  ~~(List of items~~

Only interesting patterns give knowledge.

Measures for Rule Creation:

1. support ~~> The support for $X \to Y$ is the probability
2. confidence of both X and Y appearing together, that is $P(X \cup Y)$

Absolute Support $\overset{norm}{\wedge}$ (total no of transactions in which item appears)

Relative support norm = $\frac{\text{"}}{\text{total no of transactions}}$

{Bread, Chees} 2, 2/5 = 2, 0.4 // 40%.

Confidence of $X \rightarrow Y$ is the conditional probability of Y appearing given that X exists. It is written as P(Y|X) and read as P of Y given X.

$$\text{confidence}(A \Longrightarrow B) = P(B|A)$$

$$= \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)}$$

Two steps for mining association rules.

1) Find all frequent itemsets.
2) Generate strong association rules from the frequent datasets.

1. Join
2. Pruning ⇝ done based on a property called Apriori property.

Apriori Property:
All subsets of a frequent itemset must be frequent.

1) Find frequent itemsets L_1.
2) $L_2 = L_1 \bowtie L_1$ join pruning
3) $L_3 = L_2 \bowtie L_2$ "
4) $L_4 = L_3 \bowtie L_3$

Find Association rules from the following data using Apriori algorithm with minimum support count is 2 and confidence 70%.

C1

itemset	sup.count
I1	6
I2	7
I3	6
I4	2
I5	2

L1 ← Compare support count with min. sc.
if support_count ≥ min sc
eliminate

L1

itemset	sup.count
I1	6
I2	7
I3	6
I4	2
I5	2

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

L2

itemset	sup.count
I1, I2	4
I1, I3	4
I1, I5	2
I2, I3	4
I2, I4	2
I2, I5	2

01.04.'24

L2 ⋈ L2

C3

itemset	sup.count
I1, I2, I3	2
I1, I3, I5	1
I1, I2, I5	2
I2, I3, I4	0
I2, I3, I5	1

I2, I4, I5 → 0

L3

I1, I2, I3	2
I1, I2, I5	2

i.e., C_3

itemset	sup count
I_1, I_2, I_3	2
I_1, I_2, I_5	2
I_1, I_3, I_5	1 ✗ prune
I_2, I_3, I_4	0 ✗ prune
I_2, I_3, I_5	1 ✗ prune
I_2, I_4, I_5	0 ✗ prune

$\Longrightarrow$

L_3

itemset	s.c.
I_1, I_2, I_3	2
I_1, I_2, I_5	2

subset not in L_2.

Form the association rules

Subsets of the frequent set $\Longrightarrow$ $l - S$ (remaining set)

$\rightarrow$ > 70 % $\rightsquigarrow$ strong association rule

$I_1 \Longrightarrow \{I_2, I_3\}$ confidence $= \frac{sup(I_1, I_2, I_3)}{sup(I_1)} = \frac{2}{6} = 33.3\%$

$I_2 \Longrightarrow \{I_1, I_3\}$ confidence $= \frac{2}{7} = 28.6\%$

$I_3 \Longrightarrow \{I_1, I_2\}$ confidence $= \frac{2}{6} = 33.3\%$

$\{I_1, I_2\} \Longrightarrow I_3$ confidence $= \frac{2}{4} = 50\%$

$\{I_1, I_3\} \Longrightarrow I_2$ confidence $= \frac{2}{4} = 50\%$

$\{I_2, I_3\} \Longrightarrow I_1$ confidence $= \frac{2}{4} = 50\%$

$I_1 \Longrightarrow \{I_2, I_5\}$ confidence $= \frac{2}{6} = 33.3\%$

$I_2 \Longrightarrow \{I_1, I_5\}$ confidence $= \frac{2}{7} = 28.6\%$

$I_5 \Longrightarrow \{I_1, I_2\}$ confidence $= \frac{2}{2} = 100\% \rightarrow$ } strong

$\{I_1, I_2\} \Longrightarrow I_5$ confidence $= \frac{2}{4} = 50\%$

$\{I_1, I_5\} \Longrightarrow I_2$ confidence $= \frac{2}{2} = 100\% \rightarrow$ } strong

$\{I_2, I_5\} \Longrightarrow I_1$ confidence $= \frac{2}{2} = 100\% \rightarrow$ } strong

Apriori Example

Transaction ID	Items
T1	Hot Dogs (H), Buns (B), Ketchup (K)
T2	Hot Dogs (H), Buns (B)
T3	Hot Dogs (H), Coke (Co), Chips (Ch)
T4	Chips (Ch), Coke (Co)
T5	Chips (Ch), Ketchup (K)
T6	Hot Dogs (H), Coke (Co), Chips (Ch)

$\frac{33.33 \times 6}{100} = 2$

→ min. supcount → 2

minimum support threshold : 33.33 Minimum confidence threshold : 60

	Transaction ID	Items
~~Hot dogs~~ H	T1	B, H, K
~~Buns~~ B	T2	B, H
~~Ketchup~~ K	T3	Ch, Co, H
~~Coke~~ Co	T4	Ch, Co
~~Chips~~ Ch	T5	Ch, K
	T6	Ch, Co, H

C1 :

B	2
Ch	4
Co	3
H	4
K	2

⟹

L1 :

B	2
Ch	4
Co	3
H	4
K	2

C2

{B, Ch}	0	{B, K}	1	{Ch, K}	1	{H, K}	↓ 1
{B, Co}	0	{Ch, Co}	3	{Co, H}	2		
{B, H}	2	{Ch, H}	2	{Co, K}	0		

L2 :

B, H	2
Ch, Co	3
Ch, H	2
Co, H	2

C3 :

Ch, Co, H	2

L3 :

Ch, Co, H	2

$Ch \Rightarrow \{Co, H\}$ confidence $= \frac{2}{4} = 50\%$

$Co \Rightarrow \{Ch, H\}$ confidence $= \frac{2}{3} = 66.66\%$ ⇝

$H \Rightarrow \{Ch, Co\}$ confidence $= \frac{2}{4} = 50\%$

$\{Ch, Co\} \Rightarrow H$ confidence $= \frac{2}{3} = 66.66\%$ ⇝

$\{Co, H\} \Rightarrow Ch$ confidence $= \frac{2}{2} = 100\%$ ⇝

$\{Ch, H\} \Rightarrow Co$ confidence $= \frac{2}{2} = 100\%$ ⇝

} strong

3.4.'24

Pincer-Search Algorithm

Improvement over Apriori algorithm.

Algorithm:

$L_0 = \phi$; $k = 1$; $C_1 = \{\{i\} \mid i \in I\}$; $S_0 = \phi$;

MFCS := $\{\{1, 2, \ldots, n\}\}$; MFS = ϕ;

do until $C_k = \phi$ and $S_{k-1} = \phi$

read database and count supports for C_k and MFCS;

MFS := MFS $\cup$ {frequent itemsets in MFCS};

S_k := {infrequent itemsets in C_k};

call MFCS-gen algorithm if $S_k \neq \phi$;

call MFS-pruning procedure;

generate candidates C_{k+1} from C_k; (similar to a priori's generate & prune)

If any frequent itemset in C_k is removed in MFS-pruning procedure

call the recovery procedure to recover candidates to C_{k+1};

call MFCS prune procedure to prune candidates in C_{k+1};

$k = k + 1$;

return MFS;

Q abc def

abcg minimum support = 2

abdh

bcde

abc

S represents the infrequent itemsets from C_k.
L - frequent itemsets
C_k - candidate itemsets

MFCS → Maximal Frequent candidate item set
MFCS = {a, b, c, d, e, f, g, h}
MFS → Maximal Frequent set
contains the frequent itemsets from MFCS

1) From C_1:

Itemset	Sup_count
a	4
b	5
c	4
d	3
e	2
f	1
g	1
h	1

MFCS sup_count = 0
not frequent
MFS = ϕ
S_k = {{f}, {g}, {h}}

MFCS_gen
for all itemsets s ∈ S_k
  for all itemsets m ∈ MFCS
    · if s is a subset of m
      MFCS = MFCS \ {m};
      for all items e ∈ itemset s
        if m \ {e} is not a subset of any itemset in MFCS
          MFCS = MFCS ∪ {m \ {e}};
return MFCS

L_1	Itemset	Sup-count
	a	4
	b	5
	c	4
	d	3
	e	2

considering {f} MFCS = {a, b, c, d, e, g, h}
considering {g} MFCS = {a, b, c, d, e, h}
considering {h} MFCS = {a, b, c, d, e}

MFS - Prune

for all itemsets c in C_k
  if c is a subset of any itemset is the current MFS
  delete c from C_k;

MFCS - Prune

for all itemsets c in C_{k+1}
  if c is not a subset of any itemset in the current MFS
  delete c from C_{k+1};

C_2	itemset	Sup-count
	a, b	4
	a, c	3
	a, d	2
	a, e	1

b, c	4
b, d	3
b, e	2
c, d	2
c, e	2
d, e	2

Recovery

for all itemsets $I \in C_k$

for all itemsets $m \in MFS$

if the first $k-1$ items in I are also in m

/* suppose $m.item = I.item_{k-1}$ */

for i from $j+1$ to $|m|$

$C_{k+1} = C_{k+1} \cup \{\{I.item_1, I.item_2, \ldots, I.item_k, m.item_i\}\}$

8.04.24 $k = 2$;

$MFCS = \{a, b, c, d, e\}$

$MFS = \phi \cup \phi = \phi$

$S_2 = \{\{a, e\}\}$

$L_2 = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}, \{d, e\}\}$

Call MFCS-gen algo:

$s \in \{\{a, e\}\}$

$m \in \{\{a, b, c, d, e\}\}$

$S \subseteq m$

$MFCS = \phi$

$e \in \{\{a, e\}\}$

$m | \{e\} \Rightarrow m = \{\{a, d, e\}, \{a, b, c, d\}$

$MFCS = \phi \cup \{\{b, c, d, e\}, \{a, b, c, d\}\}$

~~MFS = {a, b, c, d, e}~~

$MFCS = \{\{b, c, d, e\}, \{a, b, c, d\}\}$

$MFS = \phi \cup \phi = \phi$

$MFCS = \{\{b, c, d, e\}, \{a, b, c, d\}\}$

MFS - prune $\Rightarrow$ no action

$C_3 = \{\{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}\}$

call recovery $\Rightarrow$ no action

call MFCS - prune $\Rightarrow$ no action

$k = 3$

C_3:

itemset	sup_count
a, b, c	3
a, b, d	2
a, c, d	1
b, c, d	2
b, c, e	2
b, d, e	2
c, d, e	2

MFCS = {{b,c,d,e} = 2, {a,b,c,d} = 1}

MFS = $\phi \cup \{b,c,d,e\}$ = {{b,c,d,e}}

S_3 = {{a,c,d}}

L_3 = {{a,b,c}, {a,b,d}, {b,c,d}, {b,c,e}, {b,d,e}, {c,d,e}}

MFCS-gen $s \in$ {{a,c,d}}

$m \in$ {{b,c,d,e}, {a,b,c,d}}

$s \subseteq m$ ✓

MFCS = {b,c,d,e}

$e \in S_3$

$m\{e\}$ = {b,c,d}. {a,b,d} {a,b,c} $\notin$ MFCS

MFCS = {{b,c,d,e}, {b,c,d}, {a,b,d}, {a,b,c}}

MFS = "

MFCS = MFCS − MFS = ϕ

MFS prune

$L_3 = \phi$

all itemsets are deleted from L_3

generate C_4 ⇒ no action as $L_3 = \phi$

call recovery ⇒ no action

call MFCS-prune ⇒ no action

stop.

Find the frequent itemsets using Pincer Search

Sup_count = 2

TID	List
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

Compare Apriori and FP Growth (FP vs A)

1) Pattern generation
   constructs
   FP tree, pairing items into singletons, pairs, triplets
   ^
2) Candidate generation
   no, yes
3) Memory usage
   compact version of database is saved, candidate combinations are saved in memory.
4) Proces

Partition Algorithm

modification

2 Phases :

Phase 1 – entire database / dataset is partitioned into disjoint sets – frequent itemsets (local large itemsets)

Phase 2 – merged to ← candidate set

Algorithm:

C_k^p set of local candidate k-itemsets in partition p

L_k^p set of local large k-itemsets in partition p

L^p set of all local large itemsets in partition p

C_k^G set of global candidate k-itemsets

C^G set of all global candidate itemsets

L_k^G set of global large k-itemsets.

1) P = partition_database(D)
2) n = number of partitions
3) for i = 1 to n begin
4) read_in_partition ($p_i \in P$)
5) L^i = gen_large_itemsets (p_i)
6) end
7) for ($i=2$; $L_i^3 \neq \theta$, $j = 1, 2, \ldots, n$; i++) do
8) $C_i^G = U_{j=1,2,\ldots,n} L_i^3$ // Merge Phase
9) for i = 1 to n begin // Phase II
10) read_in_partition ($p_i \in P$)
11) for all candidates $c \in C^G$ gen_count(c, p_i)
12) end
13) $L^G = \{c \mid c = C^G \wedge count \geq minSup\}$

partition algorithm

- Initially the database D is logically partitioned into n partitions.
- Phase I of the algorithm takes n iterations.
- During ~~partition~~ iteration i only partition p_i is considered.
- The function gen-large-itemsets takes a partition p_i as input and generates local large itemsets of all lengths, $L_1^i, L_2^i, \ldots, L_i^i$ as the output.
- In the merge phase, the local large itemsets of same lengths from all n partitions are combined to generate the global candidate itemsets.
- In phase II, the algorithm sets up counters for each global candidate itemsets, and count their support for the entire database and generates the global large itemsets.

Problem

Transaction	Itemset
T1	12, 15
T2	12, 14
T3	14, 15
T4	12, 13
T5	15
T6	12, 13, 14

Partitions: {T1, T2}, {T3, T4}, {T5, T6}

First Scan min_sup_count = 1

{T1, T2} I1 - 1, I2 - 1, I4 - 1, I5 - 1

{I1, I5} = 1, {I2, I4} - 1

{T3, T4} I4 - 1, I5 - 1, I2 - 1, I3 - 1

{I4, I5} - 1 {I2, I3} - 1

{T5, T6} I2 - 1, I3 - 1, I4 - 1, I5 - 1 {I2, I3, I4} - 1

{I2, I3} - 1 {I3, I4} - 1 {I2, I4} - 1

Second Scan min_sup_count = 2

I1 - 1 (X) I3 - 2 I5 - 2

I2 - 3 I4 - 3

{I1, I5} - 1 (X) {I2, I4} - 2 {I4, I5} - 1 (X)

{I2, I3} - 2 {I3 - I4} - 1 (X) {I2, I3, I4} - 1 (X)

Shortlisted: I2, I3, I4, I5, {I2, I3}, {I2, I4}

Dynamic Itemset Counting Algorithm

Alternative to Apriori Itemset Generation (incremental approach)

Itemsets are dynamically added and deleted as transactions are read.

solid box □ } confirmed frequent itemset (finished counting, > minsup)

solid circle ○ } confirmed infrequent itemset (fc., < minsup)

dashed box ⬚ } suspected frequent itemset (still counting, > minsup.)

dashed circle ◌ } suspected infrequent itemset (s.c., < minsup)

DIC Algorithm

1. Mark the empty itemset with a solid square. Mark all the 1-itemsets with dashed circles. Leave all other itemsets unmarked.
2. While any dashed itemsets remain:
   1. Read M transactions (if we reach the end of the transaction file, continue from the beginning). For each transaction, increment the respective counters for the itemsets that appear in the transaction and are marked with dashes.
   2. If a dashed circle's count exceeds minsupp, turn it into a dashed square. If any immediate superset of it has all of its subsets as solid or dashed squares, add a new counter for it and make it a dashed circle.

Example:

minsupp = 25% (25% × 4 = 1 → minsupp)

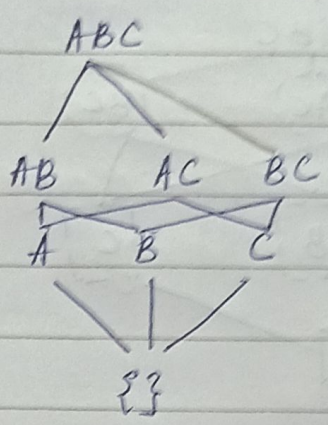
M = 2

TID	A	B	C	
T1	1	1	0	AB
T2	1	0	0	A
T3	0	1	1	BC
T4	0	0	0	

Transaction Database

Itemset lattices : an itemset lattice contains all the possible itemsets for a transaction database.

1- itemsets A, B, C.



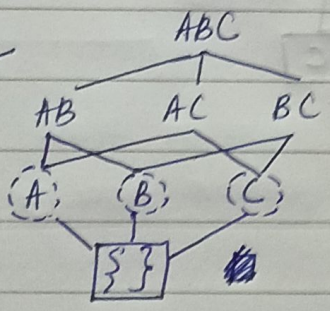
AB ⋈ AC
n-1

AB ⋈ BC ✗

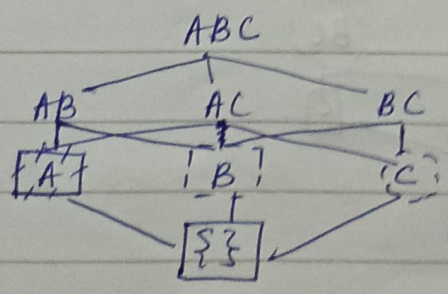
algorithm

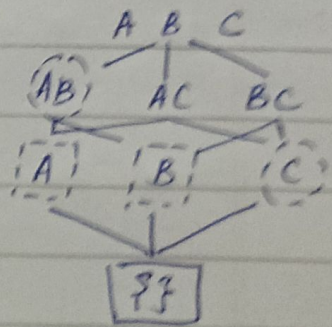
3. Once a dashed itemset has been counted through all the transactions make it solid and stop counting it.

Step 1 :



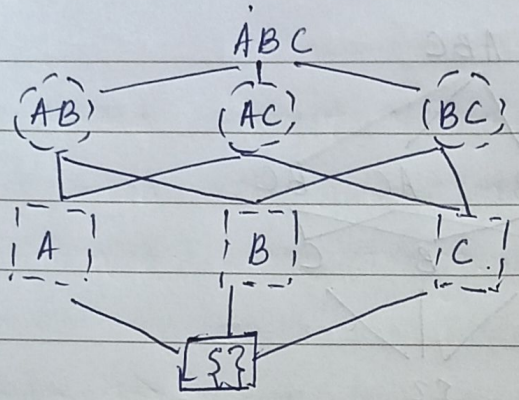
M = 2	A	B	C
T1 & T2	2	1	0

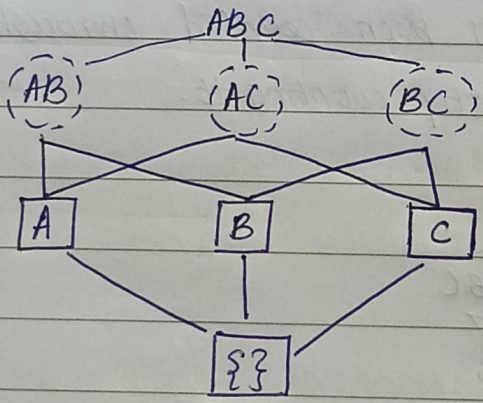




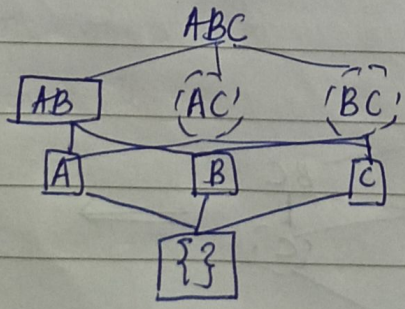
	A	B	C
M = 4	2	2	1

T3 & T4

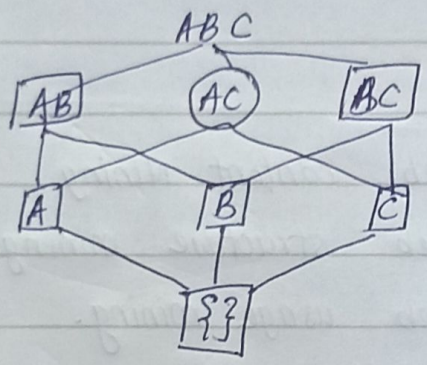




M = 6 (3M)



After 4M transactions,

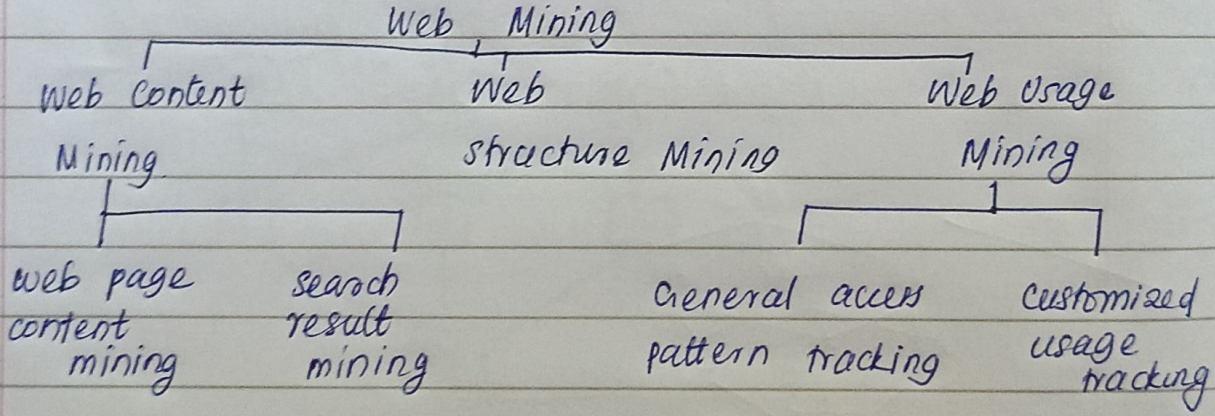


Web Mining

Mining of data related to the world wide web.
Used to discover patterns, structures and knowledge from the web.

3 areas : Web content Mining
Web structure mining
Web usage mining.

Web Mining Taxonomy



Q. T1 m, o, n, k, e, y
T2 d, o, n, k, e, y
T3 c, o, o, k, i, e
T4 m, u, c, k, y

Find the frequent-path using FP-growth.

Ans. T1 {e, k, m, n, o, y}

T2 {d, e, k, n, o, y}

T3 {c, e, i, k, o, o}

T4 {c, k, m, u, y}

L1

Itemset	Sup_count
c	2
d	1
e	3
i	1
k	4
m	2
n	2
o	3
u	1
y	3

arrange in decreasing order of support count ⟹

Itemset	SC
k	4
e	3
o	3
y	3
e	2
m	2
n	2

Transactions rearrange:

T1 {k, e, o, y, m, n}

T2 ~~{k, d, e, n, o, y}~~ {k, e, o, y, n, d?}

T3 ~~{k, c, e, i, o, o}~~ {k, e, o, o, c, i?}

T4 ~~{k, c, m, u, y}~~ {k, y, c, m, u?}