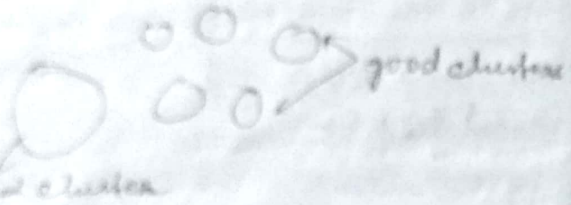
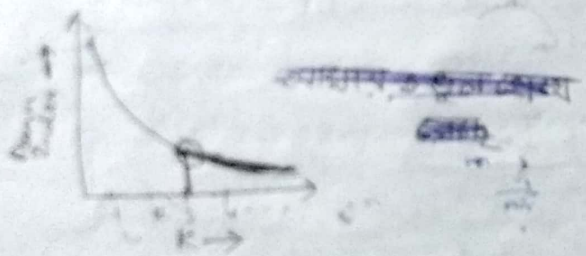
→ If DN(c) is very high, then clusters are very good. Very DN(c) high means N^2 is high which means separation is more and D^2 is less so comparison is more.

→ Our obj is to maximize DN(c)



Di index :-

① Instead of max in D^1 & D^2, we can use avg value also when all clusters are good, if one is bad then if we take max value then it will give badresult.



21/3/19

Cluster Validation Indices

② Davies - Bouldin (DB) Index :-

$$\frac{\Delta(C_1)+\Delta(C_2)}{\delta(C_1,C_2)} \rightarrow \frac{\Delta(C_1)+\Delta(C_3)}{\delta(C_1,C_3)} \quad \text{upto } m$$

① ② ③ ④ ... ⓜ

② In the worst case,

$$\frac{1}{m}\sum_{i=1}^{m} \max_{\substack{j=1\ldots m \\ i\neq j}} \left\{ \frac{\Delta(C_i)+\Delta(C_j)}{\delta(C_i,C_j)} \right\} = DB(c)$$

→ For better clusters, DB index must be low.

→ Need to compute the K value in K means

for K = 2, 3, 4, 5, ...

↓

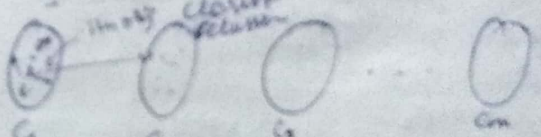
compute DB index

② The lowest value of K ⓚ that gives better DB index, is the no. of clusters.

→ If we don't know the actual requirement then we have to compute all the index values

→ If inter cluster & intra cluster criteria then we can use DB index.

Silhoutte Index (SL)



we can calculate:

SL(i) → for sample i

SL(C_i) → for cluster i

SL(c) → for all clusters

$a(i)$ = avg. dissimilarity of ith obj with the cluster in which it lies

→ Then we have to find the closest cluster from ith object.

This can be done by:

→ taking the centroid of other clusters

or → calculating dist. with all object and take their avg.

$b(i)$ = avg. dissimilarity of ith obj with all objects in its closest cluster.

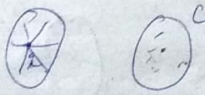
$$SL(i) = \frac{b(i) - a(i)}{\max\{b(i), a(i)\}}$$

$$-1 \leq SL(i) \leq 1$$

→ $a(i)$ must be low and $b(i)$ must be high.

if $a(i) > b(i)$ ⟹ compactness is less.

i.e.



C_1 C_2

if no. of objs in C_2 is very less than C_1 then $a(i) > b(i)$

↳ more values.

→ For good cluster, value must be nearer to 1.

If $SL(i) \approx -1$,

$$\Rightarrow b(i) - a(i) \approx \max\{b(i), a(i)\}$$

$$= -a(i)$$

$$b(i) \approx 0$$

That means ith obj is not properly placed.

If $SL(i) \approx 1$

$$b(i) - a(i) \approx b(i)$$

$$\Rightarrow a(i) \approx 0$$ (ie obj is in actual cluster)

If $SL(i) \approx 0$

$$b(i) = a(i)$$

→ Obj may be in any of the clusters.

$$SL(C_k) = \frac{1}{|C_k|} \sum_{\forall i} SL(i)$$

$$SL(C) = \frac{1}{m} \sum_{k=1}^{m} SL(C_k)$$

↓ no. of clusters

→ If $SL(C)$ or $SL(C_k)$ is higher, the cluster is better cluster

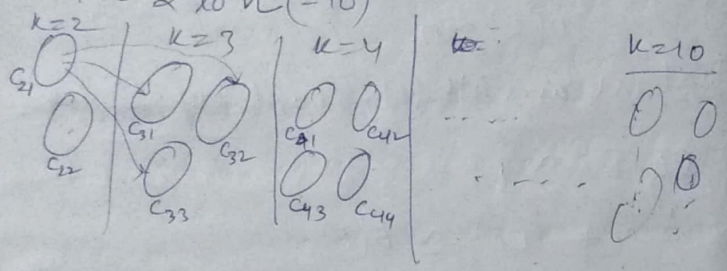
Requirement

→ Dissimilarity of obj. with centroid of other cluster or objects of other cluster

Family and Link

→ Checks whether a object is stable within a cluster or not. If yes → keep, no → discard

$K = 2$ to $n (=10)$



$$\max\left\{\frac{C_{21} \cap C_{31}}{C_{21}}, \frac{C_{21} \cap C_{32}}{C_{21}}, \frac{C_{21} \cap C_{33}}{C_{21}}\right\} = m_1$$

Similarly with $K = 4 \ldots 10$

$m_2, m_3 \ldots m_8$

Then take min of all

i.e.

$$S(C_{21}) = \min\{m_1, m_2 \ldots m_8\}$$

↓

Stability of C_{21}

$$S_{21} = \min_{i=3 \text{ to } n}\left\{\max_{j=1 \text{ to } i}\left\{\frac{|C_{21} \cap C_{ij}|}{|C_{21}|}\right\}\right\}$$

$$S_{cl} = \min_{i=2 \text{ to } n}\left\{\max_{j=1 \text{ to } i}\left\{\frac{|C_{cl} \cap C_{ij}|}{|C_{cl}|}\right\}\right\}$$

General Stability for a particular 'k'

$$GS_C = \frac{1}{C}\sum_{i=1}^{C} S_{cl}$$

Classification

→ Model ~~testing~~ training (model construction)
→ Testing (Prediction and Validation)

Issues regarding classification and prediction

- Regression (for continuous or ordinal variables)
- Classification (for discrete or nominal values)

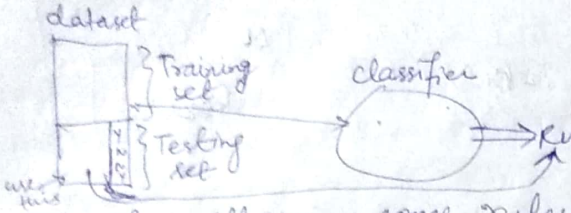
→ Prediction of class value is called classification

Steps :-

① Data preparation (All preprocessing ~~&~~ tasks)
② Performance evaluation

Performance Evaluation :-

(i) Predictive Accuracy :



→ classifier will give us some rules/decision trees

→ The output is used to predict the accuracy of the test dataset.

→ Predictive accuracy corresponds to particular test set.

(ii) Speed :-

basically time complexity

(iii) Robustness :-

→ If noises / missing values are there, then how much our model gives accuracy ? ⇒ concerns with predictive accuracy

(iv) Scalability :-

→ If dataset becomes large, what is the efficiency. concerns with speed.

(v) Interpretability :-

→ If the requirements change, we can easily modify and the algo must be easily understandable.

Accuracy :-

(i) Holdout Method :

Training :- 2/3 of the dataset (randomly select)

Test set :- 1/3 of the dataset → check the accuracy.

→ If accuracy is high, we can rely on the training.

Drawbacks

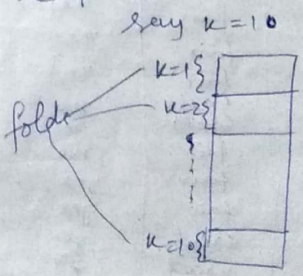
If there are not sufficient objects available in a class, then training will not be correct and hence misclassification will occur

class → C_1 ⇒ { sufficient data }

C_2 ⇒ { o_1, o_2, o_3 }

C_3 ⇒ { Suffident }

(ii) K-fold cross validation :



1st iteration :-

Training data : f_2 to f_k

Test set : f_1

2nd iteration :-

Tr. Set = f_1, f_3 - f_k

Te. set = f_2

→ Take avg. of all k' accuracies ; and that is taken as the accuracy of the classifier.

(iii) Bootstraping:-

→ Used when dataset is very small.
→ Say test data contains only 1 or 2 objects.
→ Repeat this process (for ∑no. of objs)
→ Take the first one, in next iteration the other one, then finally avg

Overfitting of Classifier:-

→ Training gives ~~test~~ good ~~result~~ accuracy, but test data will not give good accuracy.

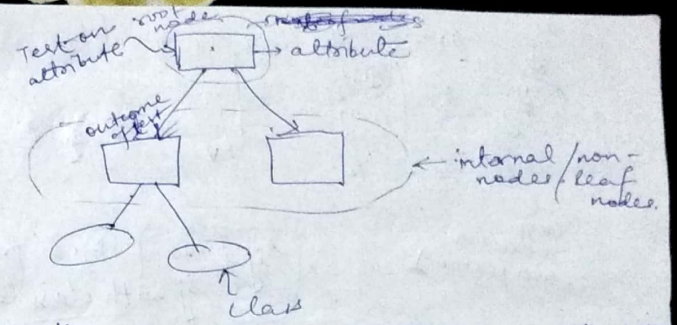
This is because:

① There are some noisy data in training set.
② Insufficient no. of features
③ Irrelevant features

27/3/18

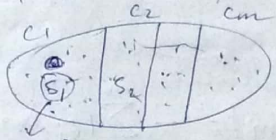
Decision Tree based classifier (Rule based)

	age	income	student	credit rating	buys comp. (class)
1	≤30	high	No	fair	no
2	≤30	high	No	Excellent	no
3	31...40	high	No	fair	yes
4	>40	medium	No	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excell.	no
7	31...40	low	yes	excel.	yes
8	≤30	medium	no	fair	no
9	≤30	low	yes	fair	yes
10	>40	med	yes	fair	yes
11	≤30	med	yes	excel.	yes
12	31...40	med	no	excel	yes
13	31...40	high	yes	fair	yes
14	>40	med	no	excel.	no



→ Information gain for each attribute is calculated

$C_1, C_2 \dots C_m \Rightarrow$ m-classes.



$S = \sum_{i=1}^{m} S_i$

No. of objs of C_i

Prob. of object being in cl C_i $= \frac{S_i}{S} = P_i$

$$I(S_1, S_2 \dots S_m) = -\sum p_i \log_2 p_i$$

expected info to classify a sample

information will be in bits.

eg

no	yes
$S_1 = 5$	$S_2 = 9$

$$I(S_1, S_2) = \frac{-5}{14}\log_2\frac{5}{14} - \frac{9}{14}\log_2\frac{9}{14}$$

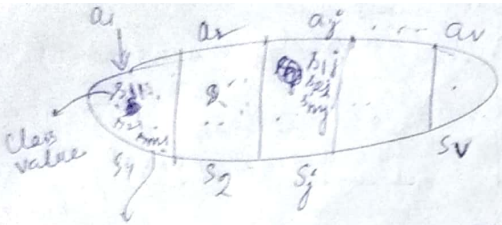
$$= 0.940$$

∴ Consider a conditional attr. A

→ Check how many distinct values are there corresp to A.

$A = \{a_1, a_2 \dots a_v\}$

discretization



s_{ij} = No. of objs in partition s_j with class label C_i

$I(s_{1j}, s_{2j}, \ldots s_{mj}) = ?$

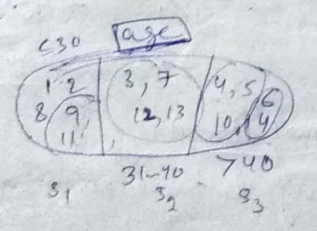
$$p_{ij} = \frac{s_{ij}}{|s_j|}$$

$$I(s_{1j}, s_{2j} \ldots s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log p_{ij}$$

$$E(A) = \sum_{j=1}^{v} \frac{|s_j|}{s} I(s_{1j}, s_{2j} \ldots s_{mj})$$

$$\text{Information gain}(A) = I(s_1, s_2 \ldots s_m) - E(A)$$

→ Corresp. to each attrib. calculate IG and take the attrib with max IG as root.



(i) $-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.971$

(ii) $-\frac{4}{4}\log\frac{4}{4} = 0$

(iii) $-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.971$

$$E(\text{age}) = \frac{5}{14} * 0.971 + 0 + \frac{5}{14} 0.971$$

$$= \frac{5}{7} * 0.971 = 0.694$$

$$\text{gain}(\text{Age}) = 0.940 - 0.694$$

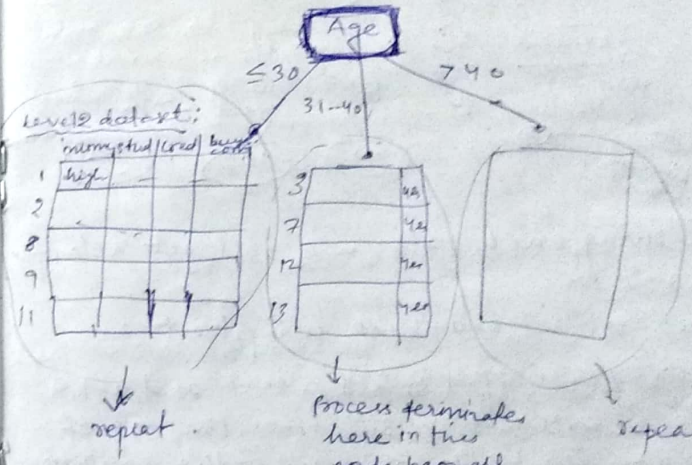
$$= 0.246$$

compute other attributes:

gain(income) = 0.029

gain(student) = 0.151

gain(credit_rating) = 0.048

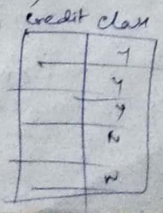


→ It is greedy approach because when max inf. gain is there we are following top down approach

Terminate:

→ we have splitted based on all attribs, we don't have any other attrib to split. (only one condition attrib, i.e. class label)

→ eg

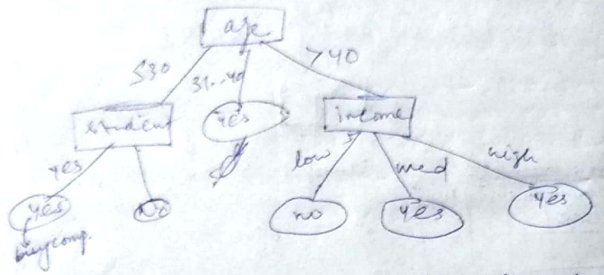


corresp to 3 values f, t, e

classified leaf node

→ No more samples are there, ∴ we can terminate.
→ If objs are in same class.

Say decision tree is:



→ Following each path, we will get a classification rule
→ Take test set and check using the tree.
→ Using k fold, if 1 fold is test and k-1 is training set, we will get K diff trees, the avg of accuracy of all K trees is the accuracy of the classifier.

Decision Tree Classifier :- 23/3/18

↓

Overfitting problem

↓

corrp to any tree, training accuracy is 98%. (say) and another 70% Testing accuracy ⇒ overfitted.

another

T_2 → 90% } not overfitted.
→ 85% }

→ due to outliers.
→ due to irrelevant, noisy ~~etc~~ etc data.

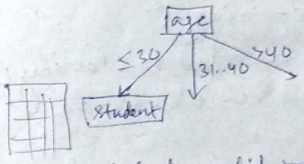
solving Overfitting :-

⇒ Remove some branches from tree.
↳ faster classification
↳ More accuracy
→ This is called Pruning.

PRUNING

PRE PUNING

before splitting check whether splitting is reqd. or not (done by checking some strength of splitting)



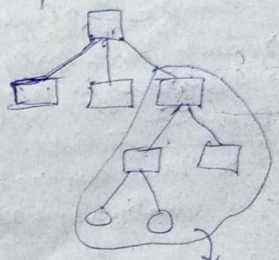
income, student, credit rating, buys_computer

Calculate IG and compare with threshold. If less, stop.
eg. If for student IG is high but < than threshold. Then stop, don't classify

Demerit

POST PUNING

construct decision tree first then check strength of each non leaf node and determine whether to prune or not.



Take away the whole subtree and replace by a leaf node.

→ Leaf node construction is based on the table for the branch.

If 15 samples
10 → Yes
5 → No
⇒ Replace By Yes

→ If error is more, Leaf then don't prune this node.

→ Go to next node.

→ Now we have several pruned trees, from all these pruned classifiers take the best classifier. And that can be done by taking a test set which is independent of train set. The one which gives more accuracy is the ~~decision tree~~ best pruned decision tree.

Merits

→ Learning technique is easy.
→ Simple: Divide and conquer.
→ Each time only IG is needed to classify.
→ Accuracy is comparable to other classifiers.

Demerit:-

→ Without discretisation, can't be classified for continuous ~~attributes, intervals~~ dataset.

Naive Bayes Classifier :-

→ Statistical classifier where prob. of different samples, classes, etc are to be calculated.

X : Object
H : Hypothesis

→ Say H says : X is in class C and X is the object whose class level is not given.

Say attributes be : colour and shape
and class : fruit type

colour	shape	fruit (v-class)
Red	round	apple

X : (Red, Round)
H : X is in class C

P(X) : Prob that X is either ~~red~~ red and round.

P(H) : Prob that ~~the~~ X is in class apple.

~~P(H/X) : Prob~~

P(X/H) : What is the prob. that X is red & round given X belongs to class apple

$$P(H/X) = \frac{P(X/H)\,P(H)}{P(X)}$$

$X = \{x_1, x_2, x_3, x_4\}$
different ~~states~~ attributes.
(student, age, name ...)

$H = \{C_1, C_2\}$
class (Yes, No)

Training set:

	A_1	A_2	...	A_k	...	A_n	class
X	x_1	x_2	...	x_k	...	x_n	C_i
	x_1	x_2	...	x_k	...	x_n	C_i
	x_1	x_2	...	x_k	...	x_n	

$X = \{x_1, x_2 \dots x_k \dots x_n\}$

$C = \{C_1, C_2 \dots C_m\}$

The object X that is to be taken should be in set X

$$P(C_i/X) = \frac{P(X/C_i)\,P(C_i)}{P(X)}$$

↳ compute $\forall i$ $i = 1$ to m.

If $P(C_i/X) > P(C_j/X)\ \forall j = 1 \dots m, i \neq j$

$\Rightarrow X$ is in class C_i

If there are 's' samples, ~~are~~ and 's' corresp. to s_i values X occurs.

i.e. $X = \{x_1, x_2 \dots x_k \dots x_n\}$

$$P(X) = \frac{s_1}{s}$$

→ X is a ~~tuple~~ in the dataset.

→ If p prob. of different classes are not given, then take $P(C_i)$ as equally likely.

Now, $P(C_i/X) = K \cdot P(X/C_i)\,P(C_i)$

$$K = \frac{1}{P(X)}$$ (For all, so ~~we~~ we can ignore.

$$P(C_i/X) = P(X/C_i) \cdot P(C_i)$$

$P(C_i) = \frac{s_i}{S}$, s_i ⇒ no. of sample belong to class C_i

S ⇒ total no. of samples.

Assumption :-

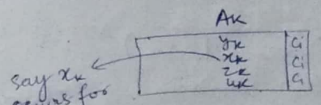
① Attributes are class conditional independent
→ i.e the attribute are not dependent on each other. (Not always true)

$$P(X/C_i) = \prod_{k=1}^{n} P(x_k/C_i)$$

→ This is possible when attributes are discrete values.

→ Cases :

Case I : If attributes (A_k) values are ~~distinct~~ discrete.



s_{ik} = no. of samples in class C_i with A_k value x_k

$$P(x_k/C_i) = s_{ik}/s_i$$

↳ No. of sample in class C_i

Case II :- If A_k values are continous.

→ we assume the values follow normal or Gaussian distribution.

$$P(x_k/C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\,\sigma_{C_i}}\, e^{-\frac{(x_k - \mu_{C_i})^2}{2\sigma_{C_i}^2}}$$

→ Real values/ordinal values.

⇒ By this compute $P(C_i/X)$ for all classes and take the max value.

Example

$X = \langle$ age = "<=30", income = medium, student = "yes", Credit rating = fair $\rangle$ ← unknown dataset → compute class

C_1 = Buys computer = 9 Yes
C_2 = 5 No

$P(C_1/X) = ?$, $P(C_2/X) = ?$

$P(C_1/X) = P(X/C_1) P(C_1)$

$P(C_2/X) = P(X/C_2) P(C_2)$

$P(C_1) = \frac{9}{14}$ (From prev dataset)

$P(C_2) = \frac{5}{14}$

$P(X/C_1) = P(x_1/C_1) P(x_2/C_1) P(x_3/C_1) P(x_4/C_1)$

Here $X = \{x_1, x_2, x_3, x_4\}$

= [ ] 9 samples with the class yes
x_1 = "age ≤ 30"
↳ 2 samples

$\therefore P(x_1/C_1) = \frac{2}{9}$

$P(x_2/C_1) = 4/9$

$P(x_3/C_1) = 6/9$

$P(x_4/C_1) = 6/9$

$\therefore P(X/C_1) = \frac{288}{94} = 0.044$

$P(C_1/X) = P(X/C_1) \cdot P(C_1)$

$= 0.044 \times \frac{1}{14} = 0.028$

Similarly,

$P(C_2/X) = 0.007$

$\therefore P(C_1/X) > P(C_2/X)$

So, the class value for the dataset is 'Yes'

Demerit:

→ It is expected to give good result because it has theoritical background. But we have assumed that attributes are independent.

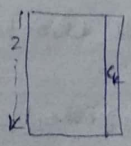
K- nearest Neighbour Classifier :-

→ Density based classifier.

every object is n-dimensional and belongs to space (assumed)

To get the class value of object (X)

→ Measure euclidean distance of x with all objects and take k nearest neighbours.



→ If neighbours belong to class C_k, then X's class value is C_k (max of the neighbours)

→ It is called the lazy learner.

→ Training time less, classification time is high.

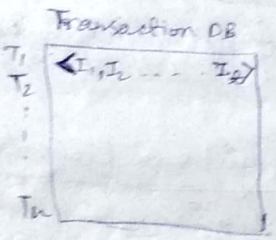
→ whenever we are taking euclidean dist, we aren't giving importance to different attributes. (All attributes are of equal weightage)

so this may effect the classifier
→ For each sample calculate the classifier.
→ Take k = ~~even~~ odd, because if same value comes in two parties, the vote goes to any one

4/4/18

Association Rule Mining :-

→ Relationship among the items in a database.
→ Useful keeps.



Market Basket Analysis:
→ ~~to~~ These items ~~not~~ will be in the basket which the customer has the tendency to buy
→ Based on Trans. DB.

$I_1 \wedge I_3 = I_5$

→ If items I_1 and I_3 are consumed by a customer, then I_5 will also be consumed by him.

Measures of association rules:-

1. $A \Rightarrow B$
2. Support $(A \Rightarrow B)$
3. Confidence $(A \Rightarrow B)$

(1) $support(A \Rightarrow B) = P(A \cup B) = \frac{\#\text{ of transactions containing both A \& B}}{\text{Total \# of transactions}}$

Eg

$<I_1, I_2, I_3>$
$<I_1, I_3, I_7>$
$<I_2, I_5>$
$<I_2, I_3>$

$Support(I_1 \Rightarrow I_3) = \frac{2}{4} = \frac{1}{2}$

→ ~~If A ∪ B ...~~
→ If no of transactions containing both A and B is very less than no. of transaction, then support $(A \Rightarrow B)$ will be very small value which means it is noise.
⇒ some threshold must be set i.e. min support threshold.

(2) $Conf(A \Rightarrow B) = P(B/A) = \frac{\#\text{ of trans. containing A \& B}}{\#\text{ of trans. containing A only}}$

→ frequent item sets are within same transaction

~~Apriori Algo~~

Apriori Algorithm

Step 1 : Join step
Step 2 : Pruning step

Trans. DB

T_{100}	$I_1\ I_2\ I_5$
T_{200}	$I_2\ I_4$
T_{300}	$I_2\ I_3$
T_{400}	$I_1\ I_2,\ I_4$
T_{500}	$I_1\ I_3$
T_{600}	$I_2\ I_3$
T_{700}	$I_1\ I_3$
T_{800}	$I_1\ I_2\ I_3\ I_5$
T_{900}	$I_1\ I_2\ I_3$

Find no. of candidate items : I_1, I_2, I_3, I_4, I_5 by scanning the DB.

Scan the database D to count each candidate

C_1

Items	support count
I_1	6
I_2	7
I_3	6
I_4	2
I_5	2

L_1 ← frequent occuring element (remove elements which are less than min^m support count)

I_1
I_2
I_3
I_4
I_5

min-support count will be given

$min\_sup = \frac{2}{9} \times 100\% = 22\%$

C_2 = candidate 2 itemset

	support count
$I_1\ I_2$	4
$I_1\ I_3$	4
$I_1\ I_4$	1
$I_1\ I_5$	2
$I_2\ I_3$	4
$I_2\ I_4$	2
$I_2\ I_5$	2
$I_3\ I_4$	0
$I_3\ I_5$	1
$I_4\ I_5$	0

$C_2 = L_1 \bowtie L_1$

frequent item set

L_2

$I_1\ I_2$
$I_1\ I_3$
$I_1\ I_5$
$I_2\ I_3$
$I_2\ I_4$
$I_2\ I_5$

L_2 discard

C_3

$I_1\ I_2\ I_3$	2
$I_1\ I_2\ I_5$	2
$I_1\ I_3\ I_5$	0
$I_2\ I_3\ I_4$	0
$I_2\ I_4\ I_5$	0
$I_2\ I_3\ I_5$	1

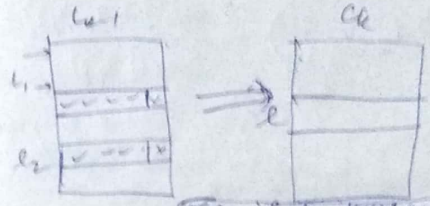
$C_3 = L_2 \bowtie L_2$

L_3

$I_1\ I_2\ I_3$
$I_1\ I_2\ I_5$

In general, $C_k = L_{k-1} \bowtie L_{k-1}$

Assume lexicographic order



e = joining operation on l_1 and l_2

Join if all items are same and last item is different

$(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k_0 - 2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$

$\{I_1\ I_2\ I_3\ I_4\ I_5\} \times \{I_1\ I_2\ I_3\ I_4\ I_5\}$

$\bowtie L_3$ → C_4

$I_1\ I_2\ I_3$	$I_1\ I_2\ I_3\ I_4$
$I_1\ I_2\ I_5$	

→ so L_3 is considered as the 3-frequent items sets (2 tuples)

→ Joining step is used to find the candidate items and pruning step is used to remove the ~~items~~ required tuples.

Now

take $\{I_1, I_2, I_3\}$

Subset of this

$\{I_1\}\ \{I_2\}\ \{I_3\}$, $\{I_1, I_2\}\ \{I_1, I_3\}\ \{I_2, I_3\}$

Generate the rules:

$\{I_1\} \Rightarrow \{I_2, I_3\}$ ✗

$\{I_2\} \Rightarrow \{I_1, I_3\}$ ✗

$I_3 \Rightarrow \{I_1, I_2\}$

$\{I_1, I_2\} \Rightarrow \{I_3\}$

$\{I_1, I_3\} \Rightarrow \{I_2\}$

$\{I_2, I_3\} \Rightarrow \{I_1\}$

$$conf(\{I_1\} \Rightarrow \{I_2, I_3\}) = \frac{sup(\{I_1, I_2, I_3\})}{sup(\{I_1\})}$$

(No. of trans. where I_1, I_2, I_3 occur / No. of trans. where I_1 occurs)

$= \frac{2}{6} = \frac{1}{3}$

$= 33\% \simeq$ Not interesting ↓ Discard

$conf(\{I_2\} \Rightarrow \{I_1, I_3\}) = \frac{2}{7} < 70\%$

$conf(\{I_3\} \Rightarrow \{I_1, I_2\}) = \frac{2}{6}$ ✗

$conf(\{I_1, I_2\} \Rightarrow \{I_3\}) = \frac{2}{4} = 50\%$

$conf(\{I_1, I_3\} \Rightarrow \{I_2\}) = 100\%$ ✓

$conf(\{I_2, I_3\} \Rightarrow \{I_1\}) = 100\%$ ✓

None is relevant

Go for $< I_1, I_2, I_5 >$

$conf(\{I_1\} \Rightarrow \{I_2, I_5\}) = \frac{2}{6}$

$conf(\{I_2\} \Rightarrow \{I_1, I_5\}) = \frac{2}{7}$

$conf(\{I_5\} \Rightarrow \{I_1, I_2\}) = \frac{2}{2} = 100\% > 70\%$ ✓

$conf(\{I_1, I_5\} \Rightarrow \{I_2\}) = 100\%$ ✓

$conf(\{I_2, I_5\} \Rightarrow \{I_1\}) = 100\%$ ✓

Interesting Measures of a rules :-

1. Simplicity :- The structure defining the rule must be as much simple as possible. Eg. in decision tree if more leafs are there then rule length is more.
2. Certainity :- Confidence value must be high
3. Utility :- Usefulness of the rule. It is denoted by support value. High support value ⇒ more utility.
4. ~~Novelty :- If the generated rule is big but we aren't getting any info~~
5. Novelty :- If any rule is giving some info but is redundant. (It must be eligible to be eliminated) But we keep it.

eg. $loc(X, "WB") \Rightarrow Buys(X, "SONY-TV")$ [8%, 70%]

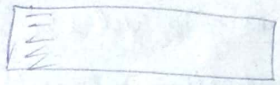
$loc(X, "KOL") \Rightarrow Buys(X, "SONY-TV")$ [2%, 72%]

↳ Redundant.

⇒ Used in finding concept hierarchy rule

Drawbacks of Apriori algo :-

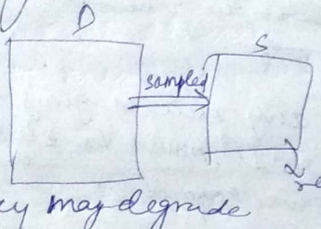
- If DB size is big, everytime scanning the DB is time consuming.

L_{k-1} 

$C_k \Rightarrow \{ I_1, I_2 \ldots I_k \}$

↳ If all the subsets of this are frequent, then it is frequent otherwise not frequent., so we don't need to ~~check~~ scan the database. So we reduce the database

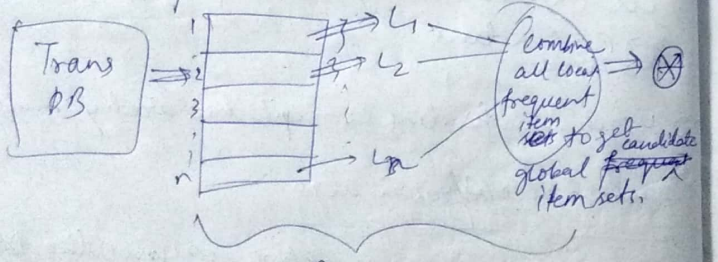
(2) Sampling :-



→ Accuracy may degrade

↳ Obj - increase accuracy

(3) Partioning :-



Phase I

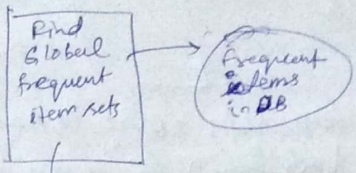
↓

we are scanning database once.

→ Partitioning reduces the dataset so that it can perfectly fit in the MM.

→ ~~The dataset size must be~~

Phase II



(4) Hash Table Technique :-

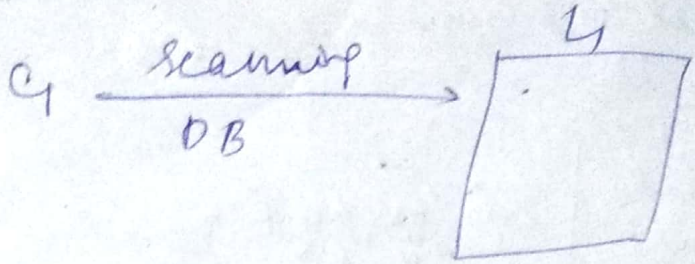
(5) Transaction Reduction :-

→ If a transaction (a tuple) is not giving ~~k~~ frequent item sets then it won't give (k+1) frequent item set for that transaction. Mark this transaction.

(4) Hash Table Technique :-

First tuple $\{ I_1, I_2, I_5 \}$

Bucket address	0	1	2	3	4	5	6
Bucket count	2	1	1	1	1	2	
Bucket contents	$\{I_1, I_4\}$, $\{I_3, I_5\}$	$\{I_1, I_5\}$	$\{I_2, I_3\}$	$\{I_2, I_4\}$	$\{I_2, I_5\}$	$\{I_1, I_2\}$, $\{I_1, I_3\}$	



Generate two item sets corresp. to each tuple and generate hash funcⁿ for those items.

eg. $\{I_1, I_2\}$

$h(\{I_1, I_2\}) = (1 \times 10 + 2) \bmod 7$

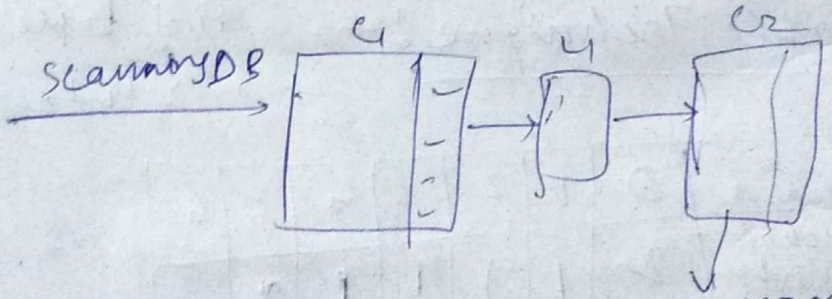
Consider each element and check the ~~Element~~ minⁿ support value.

↓

more ✓ (take for L_2)

less ✗

→ In the next step (C_2) ⇒ increase the minⁿ support value by 1.



No need to scan the DB, from the hash table we will get the L_2 and C_2 sets.