12-8-20

Date:
Page No.

⇒ Data Systems Evolution

- Traditional Database Systems
- → Indexing - to make the database systems fast
- → Query languages - to make convenient to query the database
- → Query Optimization - how it is optimally executed
- → Transaction → happen according to ACID semantics. A → Atomicity
  C → Consistency
  I → Isolation
  D → Durability
  (I → Isolation: also allows multiple transactions to occur at the same time)
- → Recovery - If database is corrupted/power goes off, how will the database recover.

Relational Database Mode → most famous
[Query Language → SQL
↳ structured in nature.

Graph Databases → Entities - Nodes
Relationships - Edges.

Functionalities:

i) Distributed & Heterogenous
↓ distributed across various system, over cloud etc. (not on single computer system)
↓ different systems with store data in different way for eg: Excel sheets, JSON etc

And such systems should have interoperability (able to communicate etc.)

Date:
Page No.

Data Analytics

Data Warehousing

i) Data is stored in one central place called as data warehouse.

ii) Data needs to be collected from different servers in many different platforms & put it into data warehouse.

iii) Data warehouse is not updated on a daily basis. Maybe weekly or monthly.

iv) Purpose: Store all the data in your enterprise across many years to extract patters

v) Data analysis is majorly done manually. using for example: OLAP.

Data Mining

i) Data can be in the form of csv, data warehouse, etc.

ii) Data is dumped into the data mining algorithms i.e. automated pattern discovery happens basically

Data Science Life Cycle → KDD Life Cycle (Knowledge Discovery in databases.)

Date:
Page No.

⇒ Types of Patterns

(1) Associations Eg: B/w ~~Sugar &~~ Coffee & Sugar

(2) Clustering Eg: Within a cluster, data or records are very similar to each other. & across the clusters they are very different.
(here we do not give any label)

(3) Classification Eg: Predict / Estimate a class or label.

Association Rules.

↳ defined when we see something happening very frequently.

Eg: Tomato, Potato → Onion

If a person buys T & P, he is very likely to buy O.

Confidence ~~Support~~ = $\frac{2}{3}$ → from below, how many buy O from them
3 → who buy T & P

= 66%.

$\frac{\frac{2}{4}}{\frac{3}{4}} = \frac{2}{4} \times \frac{4}{3} = \frac{2}{3}$

Support = $\frac{2}{4}$ = 50%.

↳ Probability of RHS given LHS
i.e. P(P|L).

↳ Probability of the entire item set occurring ~~at all~~ in the whole data set.

User gives min confidence & min support & based upon this, we need

Date:
Page No.

to define such rules which gives more than the specified min % of confidence & support.

Association Rule Application

① E-commerce : Recommendation system based upon associations. (eg: if we liked/bought this, we are likely to buy that.)

⇒ Types of Association Rules

① Boolean Association rules
↳ if an item is present or not
Eg: Potato is present or not, Tomato is present or not (eg: of 'association rules slide')

② Hierarchical Rules → Doubt

③ Quantitative & Categorical Rules.

And other rules also - All of these can be represented as Boolean Association Rules with some tweaks

---

⇒ Classification Problem ⇒ Supervised Learning

⇒ Clustering Problem ⇒ Unsupervised Learning
↓
One way is to find Euclidean distance and categorise based on the distance
Eg Outlook | Temp | Humidity | Windy

Date:
Page No.

We can treat Temp & Humidity as x & y coordinates & find Euc. distance and hence clustering will be done.

Clustering is great for exploration of dataset.

⇒ Classification

All the algorithms have a different kind of output. Thus the metrics to be considered is Accuracy, training time etc. and not only response time or memory usage.

Accuracy → main metric

Accuracy → Probability of success.
Mean value = p. (p^0)
Variance = $p(1-p)$

• Classification can be treated as bernoulli process.

Discrete

series of events where they are classified correctly (p) or "incorrectly" (1-p) — random variable values.

Classified correctly

$X = 1$ ( )
$X = 0$ (X)

How X can be 1, 2, 3 --

Doubt

- If there are n records, there are n independent & identically distributed bernoulli trials X_i where $i = 1, \dots, n$.

Then the random variable $X = \sum_{i=1 \dots n} X_i$ is said to follow a binomial distribution

Discrete

Simply means Total numbers of success at 1 is for success.

$$P(X = k) = {}^{n}C_{k}\, p^{k} (1-p)^{n-k}$$

Probability that number of correctly classified records are k.

- Normal distribution → Continuous

3rd Lecture →

Date:
Page No.

Estimating Accuracy

Hold out	k-fold cross validation
Here a record may never come in training or test (whatever be the case.	Here each record is treated as training or test atleast once

⇒ Classifiers are combined for many reasons:

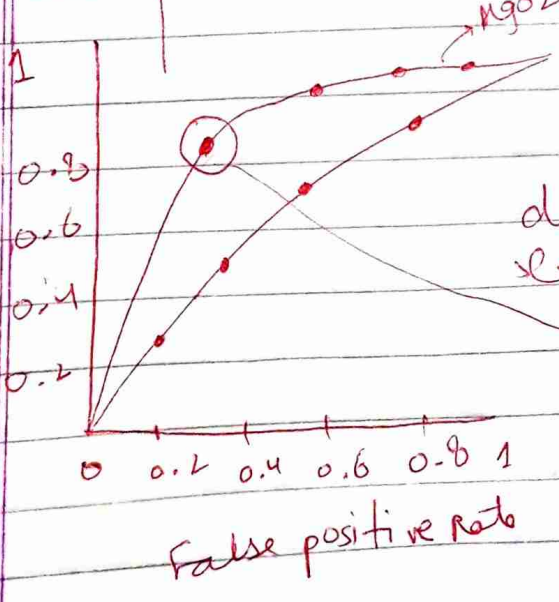
i) Different strengths
ii) Some are good for sparse data & some are good for closely related data.

No free lunch Theorem – There is no algorithm which can do the best on all datasets. An algo. may perform good one one dataset & bad on another.

24-8-20

ROC Curve (Receiver Operating Characteristic)

each point basically is a run of the algorithm with different parameter value.



Algo 1 Ran 2 algo with different parameter settings

Best parameter setting here it is closest to 1 for true positive Rate (which we wanna achieve)

Date:
Page No.

Thus accuracy can be taken as area under the curve. More the area under curve, more accurate the algo.

Ensemble Classifiers - We take a particular classification algorithm and make tweaks in it & generate say k different classifers (underlying algo is same. Just the different tweaks to same algo.)

eg: randomization

Now each classifer classifies the given record. And based on the k results, we decide which is the correct class.

Homogenous classifiers - classifiers emerging from the same kind like decision tree etc. with small tweaks.

Heterogenous classifiers - mean different types

In Bagging or Boosting, we have k number of classifiers, → they can be homo or hetero.

democracy

Bagging - The class (label) which gets the majority votes by different ensemble classifiers is chosen

Boosting - Give ~~majo~~ weight

meritocracy

Outliers - points which are very far away from others.

Date:
Page No.

⇒ Drawback of KNN

- Time Consuming because whole training data is the model itself. And when given a record, we need to search the whole training data to get K nearest neighbours.

⇒ Reverse Nearest Neighbours - used specifically when there are outliers.

Doubt

⇒
- K is typically chosen as an odd number
- K should be small.

⟹ Decision Tree

- Pure Node - where the class gets identified.
- Impure Node - where there are instances of different classes.

Q How to choose attributes for splitting?

(1) ID3 Algorithm/method is used.

↳ Use Info gain for the best split.

~~Info Gain = Entropy of the dataset - $\sum P(D_i)$~~

how large is this partition wrt to the dataset

$$\text{Info Gain} = H(D) - \sum_{i=1,\dots n} P(D_i) H(D_i)$$

where

$$H(D) = H(p_1, p_2, \dots p_m) = -\sum_{i=1,\dots m} p_i \log p_i$$

Summation over. probabilities

entropy of the whole dataset

Date:
Page No.

entropy - measure of information.

$H(D)$ → entropy before split

$-\sum_{i=1,2,\ldots n} P(D_i) H(D_i)$ → expected entropy after split

Thus info gain means how much info we are getting after the split. Thus, we need to maximize the info gain. Thus we calculate for which attribute we are getting the max. info gain and take that for splitting.

$\log 0 = -\infty$.
$0 \log 0$ = undefined
] usual case

But in the entropy formula $0 \log 0 = 0$.

- 20-question game:
  Is it a living thing - higher entropy, probability ~ 1
  Is it dusty - small entropy value, prob ~ 0.
  ∴ living thing ✓

(2) CART → split into 2 children always

$P(C_i/D_1), P(C_i/D_2)$ - Probability of particular class in partition 1 & 2 respectively.

If there difference is good enough → then good split.

Date:
Page No.

lec 5
doubt

Shannon's Entropy

③ C4.5

- Handles missing data → can put average of the nearby/related values in case of numerical attributes; and can put majority vote/ answer from neighbouring value in case of categorical attributes → just like KNN

- Handles numerical attribute split
  Sort the values. And then use binary search on the sorted attribute values to find the best split point.

- Pruning

* Prepruning – Helps to avoid overfitting of data. The split stops when the nodes become almost pure. To avoid the decision tree from doing split based on specific attributes like name, roll number etc, prepruning is done.

Date:
Page No.

If it is not done, accuracy will be very high on training set but bad on testing

* Post pruning - In this, the Decision Tree is made ~~then~~ until we achieve all the pure nodes. ~~Then~~ Here, the dataset is divided into training, validation & test. DT is made from Training. Now, accuracy/error rate is checked before removing a pure node & after removing it. If error rate/acc. doesn't change much, it is removed. And it continues until accuracy is good.

Time consuming

- Rules - Instead of DT, rules shall be made for each path (from root of DT to all the leaf nodes) in words/text.

- Goodness of split
  - Info gain favours attributes with many values. So might split based on roll number, names etc. Thus ~~info~~ gain-ratio should be considered.

DT + Bagging/Boosting → one of the best classifiers.

→ next page

Date:
Page No.

Q How to get ensemble DT classifiers:
Incorporate randomness (for eg:) in the following ways:

i) Instead of picking up the best attribute for split, choose a random " & " " among the good attributes (based upon some value like if info gain > x, then those attributes are good for split)

ii) Take smaller random dataset from entire dataset & repeat the process to get many DTs.

Above points will helps us get a numbers of DTs with small tweaks — Random forest.

And now Bagging/Boosting can be applied.

DT Complexity : $O(N \times f \times d)$

N = Number of samples
f = number of features
d = ~~depths of tree~~ no. of interior nodes basically

for checking split we traverse through all the features. And for getting each nodes probabilities/division of classes, we go through all the samples. And this repeats till the depth by the tree (until we get pure/almost pure nodes).

very expensive.

31-8-20

Dec. 6:

Doubt

⇒ SLIP

Suppose keyword is "free". It will have some prob. in spam & some prob in not spam

But Prob(free/spam) > Prob(free/not spam).

And more such words. And hence overall prob. for a ~~to~~ P(Ci|X) gets affected accordingly.

Bayesian Methods

Naive Bayes

$X = (x_1, x_2, \ldots, x_m)$

consider the classification as spam & not spam email.

Let x_1, x_2 -- be words in the email like free, congratulations etc.

$P(C_i|X)$ ⇒ Probability of a being a class (spam or not spam) given a record X.

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

→ will be same for every class. It such doesn't depend upon class so remove for proportional

$P(C_i|X) \propto P(X|C_i)\,P(C_i)$

$P(C_i|X) \propto P(x_1|C_i)\,P(x_2|C_i) \ldots P(x_m|C_i)\,P(C_i)$

probability of a keyword in a particular class.

we can multiply like this on the assumption that all the keywords $x_1, x_2 \ldots x_m$ occur independently of each other.

Naive Bayes is supposed to work nicely for text classification.

calculations might be wrong due to the above assumption but classification is good.

Date:
Page No.

~~because of new event~~

The assumption of keywords being independent is bad. It gives approximated prob. values.

eg: University & degree are 2 different words yet dependent.

To overcome this, we have

⇒ Bayesian Belief Networks → mathematically robust.

↳ exponential problem → ~~for~~ every keyword dependent on every other keyword (N^N)

We create conditional Probability Tables for the ~~nodes~~ nodes ~~to~~ which we have incoming edges.

The table is filled in 2 ways:

i) Either ~~take for~~ calculate using the given dataset (if it is proper)

ii) Domain experts can give values and we can use them directly.

$$P(X) = P(x_1 \mid Parents(x_1))\ P(x_2 \mid Parents(x_2)) \dots P(x_m \mid Parents(x_m))$$

~~P(X)~~ (Eg: P(Positive XRay, Dyspnea)

x_1 x_2.

Doubt Lec: 2-9-20

when we get P(X), we ~~consider~~ then calculate $P(c_i \mid X)$ as calculated in Naive Bayes Method.

what is this here?

Lec: 2-9-20.

Doubt ⇒

Date:

Page No.

Maximum Entropy Approach

Page No.

CLUSTERING

↳ needs a similarity function

Measurement of similarity

→ Categorical variables

$d(x, y) = 1 - \frac{m}{n}$ → Jaccard coefficient / Jaccard similarity.

x a record, y some other record:

$d(x, y)$ = similarity b/w x & y.

Jaccard similarity - Given 2 sets. Take intersection of the 2 sets divided by union of the two sets i.e. $\frac{A \cap B}{A \cup B}$

$A \cap B$ tells overlap.

$A \cup B$ how many total variables in 2 features

→ Numeric variables - Euclidean distance can be used $\cdot D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Manhatten dis = $|y2 - y1| + |x2 - x1|$

Minkowski dis = n^{th} root $\left(|y2 - y1|^n + |x2 - x1|^n \right)$

if $n = 1 \Rightarrow$ Manhatten distance

$n = 2 \Rightarrow$ Euclidean distance.

Ordinal variables - Where the exact values are not meaningful. Those numbers are given ranks

scores: 5, 9, 7 → ranks: 3, 1, 2

$z = \frac{\text{Rank} - 1}{M - 1}$

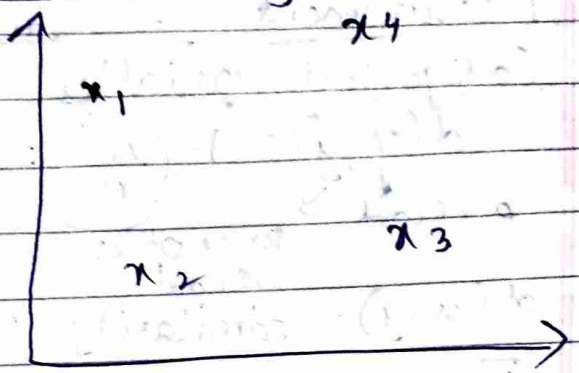
↓ normalized rank.

and then we can use Euclidean or any other dist. formula.

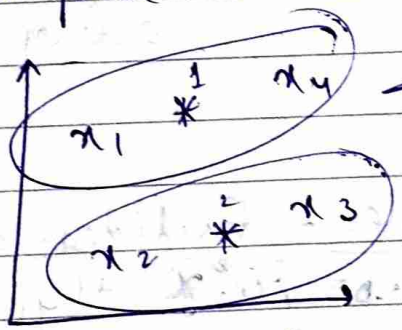
Date:
Page No.

z-score → b/w -1 to 1.

⇒ K-Means (Partitioning Based)



Choose k = 2. Initial means ⇒ x_1 & x_2.
Now find dist of x_4 with x_1 & x_2.
Now find dist of x_3 with x_1 & x_2.
Compare the distances & cluster.



→ Now recompute means as the average of all points in each cluster

Now here for all the 4 points, we will check the dist & from 1 & 2 & make 2 clusters accordingly.
If cluster formation does not change, we stop.

How to choose k? → Try different k & see what gives better clustering
How to divide initial means → Rit & try; Randomization.

There is a theorem which states the algorithm converges. But if there are too many iterations before converging,

→ Suppose there is an outlier & we force it to be a part of a cluster. Then quality will decrease as the distance of that point with the mean will be large & thus it will be squared and (to be kind of magnified.)

Date:
Page No.

we can put an upper limit to the number of iterations.

Now to know we get the best clustering? → Evaluate clustering quality.

$$E = \sum_{i=1}^{N} \sum_{x \in C_i} d(x, m_i)^2$$

squared error. → we need to minimize this for better clustering

intuitively a measure of how far away things are from their means.

K-Means works only for numerical attributes.

Work around:

i) Drop categorical & do clustering based only on numerical

ii) (Doubt) Take numerical separately & categorical separately. Calculate distance for both & then give different weights to each of them (if required). Then combine these two distances with a weighted summation.

One hot encoding → stored as Bit strings. is used for converting categorical to numerical bec ~~they~~ it are computationally very efficient. (inbuilt hardware operations etc. & are computationally very fast.)

→ 1 → records which have Red
→ 1 → " " " Green

Date
Page No.

Eg: Red: 1 1 0 1 0 0 1 1 0 0 0 1
Green: 0 1 0 1 0 0 0 1 1 0 0 1

Red & Green → have both R & G.
↘ This Bitwise operation (AND) is very fast.

This one hot encoding works nicely for smaller ~~set~~ sized datasets.
In case the datasets are large, this notation can be converted ~~into~~ into only representing 1s i.e [1, 2, 4, 7, 8, 12] records that have Red.

⇒ Hierarchical

① Agglomerative → each cluster is a part of some other cluster.

Q. How to merge 2 close clusters & ie how to find distances b/w clusters?

1. Single link- $O(N^2)$ → dist b/w 2 points that are closest in 2 clusters
2. Complete Link- $O(N^2)$ → " " farthest "
3. Average link- $O(N^2)$
4. Mean Link - $O(N)$ → ~~mean of~~ Dist b/w means of 2 clusters.

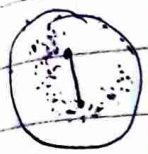
~~In agglo~~

In agglomerative or divisive, we can set thresholds to stop in b/w rather than getting a single large cluster or all single points as clusters respectively.

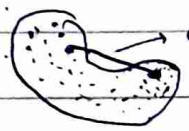
Date:
Page No.

⇒ **Density Based DBSCAN** → dense region should become a cluster.

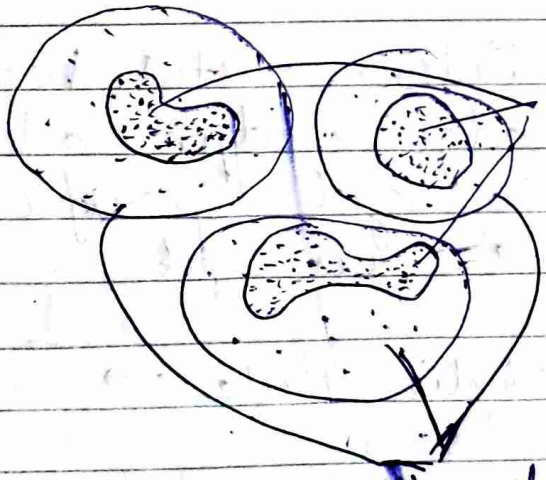
K-Means & Hierarchical clustering Algorithms gave spherical / convex clusters (the line b/w 2 points lie in the cluster itself).



Whereas DBSCAN ~~helps~~ is a way in which all clusters need not be convex.



Some enhancements to DBSCAN:
DBSCAN + Hierarchical.



⇒ **Mining Outliers using Clustering**

↳ can be found by simply running a clustering algo. If any point or a smaller cluster is left, treat it as outlier.

↳ There are techniques to mine outliers which does not require clustering.

21-9-20

Date:
Page No.

⇒ Association Rules

Naive algo ~~fro~~ for frequent itemset

↓

Keep counters. Then again scan & find support.
Exponential search → 2^n
where n = no. of items.

Doubt ⇒ KILLER APPS

- Contrained Rules – useful bcz the original algos output a lot of rules. So to get specific rules, we use this concept

- Cyclic / Periodic Rule → Time associated

we try to first satisfy the support & then try to increase the confidence.

Doubt Interest & Conviction

Date:
Page No.

Q How to choose minimum support value?

→ Choose different values say 10%, 20% -- 50% . etc. on x-axis.
And on Y-axis, output the ~~frequent~~ number of itemsets.
Low min sup → Higher num. of itemsets
High " " → Lower " "

As support gets close to 1, we get 0 number of itemsets (simply means all the ~~itemsets~~ records containing all itemsets which is almost impossible.)

A knee point in the graph can be a good minsup.

Q How to modify datasets to ~~avoid~~ maintain data privacy

Ans: One way can be to remove attributes like name, id, phone number etc. because they anyway does not help in data mining

① Generate frequent itemsets using Algos
② Generate association rules.

Date:
Page No.

⇒ Frequent Itemset Algorithms

① Apriori Algorithm

Idea: An itemset can be frequent only if all its subsets are frequent.

in slides ← D → ABC → negative borders itemsets. L their subsets are frequent itemsets but they are not

∅

A, B, C, AB, AC, BC.

subsets.

minimal infrequent itemsets. (i.e they are infrequent but could have been frequent so they also need to be checked).

proceeds level wise, so start with level 1. Generate candidates of level 2. Then make a scan for level 2 & generate for level 3

Apriori Algorithm → Bottom-up

i.e first check for singletons and then look at it supersets & so on.

Pruning step - If immediate subsets are not in F, prune Z.

F = ABC ABD

ABCD → prune bcz ACD is not in F.

Here imm. subsets of ABCD will be the subsets of length 3. We need not check for levels below 3 bcz we are already proceeding in bottom-up approach checking at every level.

Date:
Page No.

This algo is fairly efficient bcz we don't need to generate. It generates the minimal candidates.

we are not generating any candidate

While implement, at the 2nd i.e single tone level scan, the algorithm becomes very slow (pruning cannot happen) suppose F has 600 singletons. Now $600C_2$ will be new candidates generated and they cannot not be pruned.

So for Ist pass: Use a bucket/count sort technique to store the frequency of singletons. They will be pointing to a triangular matrix which is to be used for 2nd pass

in slides: White ones → freq.
violet ones → infrequent.

$A[b_i]$ ++.
↳ $\in$ t (transaction.

t' ⇒ will have all frequent singletons.

Singletons - 1D Array
2 items in set → 2D Array — for fast lookups
But for 3 items in a set like (ABC, BCD) etc.

Date:
Page No.

we cannot have a 3 dim. array bcz now that structure will be sparse enough and thus waste of memory So we use hash trees from 3rd scan onwards.

Hash Tree Algo → do from slides.

---

(2). The Partition Algorithm

- In 2 DB scans, we get frequent itemsets.
- Divide the DB into some n partitions such that each partition can fit in the main memory (along with the data structures needed to implement this algo.) Thus, there will be no disk access & hence efficient.

on getting freq. itemsets from each partition, do $F = F_1 \cup F_2 \cup F_3 \cdots$

In the end, only those F_n itemsets which are frequent in the whole database are left.

Thus this algo claims that an itemset infrequent in each partition is infrequent in entire database.

Proof : in slides.

Basic Idea.

Date:

Page No.

(3) Doubt

Sampling → gets an approximation of frequent itemsets instead of the exact freq. itemsets because we consider a random sample of the entire dataset & find F_s (freq. itemsets) & N_s (Negative borders/ minimal infrequent itemsets.)

But if we want to get the exact answers using F_s and N_s :
Here also we have 2 DB scans / Pass.

(4) Incremental Mining Technique

1 Scan Algorithm

Date:
Page No.

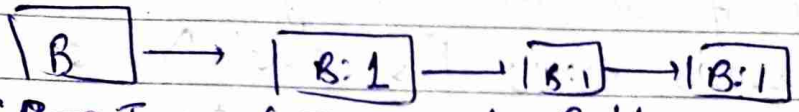
⇒ **FP-Growth** → Does data mining without candidate generation.

↳ Follows a trie kind of tree structure/ prefix tree.

The header table has 2 things:

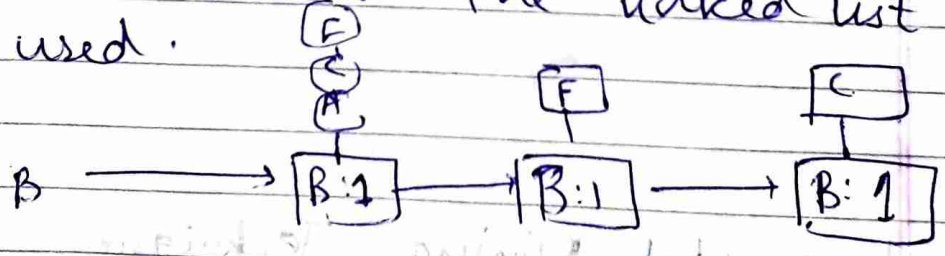
i) What is the count of each letter.

ii) Pointer to the first node and so on. ↳ Like link all the similar items.



FP-Tree (Frequent Pattern tree) → Tree + Linked list.

Conditional Pattern Base → Go through the linked list & the prefix of each node in the linked list is used.



So Pattern base of B is

FCA : 1 → values of these B.
F : 1
C : 1

~~Dense dataset~~ →

In the FP-Growth algo, we have conditional pattern Base of in levels like first for singletons. Then for pairs ~~[illegible]~~ & so on as this algo is recursive.

does counting amazingly.

Date:
Page No.

So somehow we are generating the candidate sets.

Dense dataset - When there is a ~~st~~ lot of sharing of prefixes or commonality. (Long prefixes get shared.)

we have frequent itemsets. We are just counting

⇒ ORACLE / ~~[redacted]~~

DAG structure: has 2 links from which a node is formed:

Tidlist: All the transactions that contain this itemset in the current partition.
of an itemset

Eg: for A, → AB, AC, AD, ABC -- etc

Eg: for AB → we'd take tidlist(A) ∩ tidlist(B)

if eg: AB

On intersection, we convert ~~B~~ one into tidvector (1 represents if A is present in that i^{th} record (i^{th} bit) & B is represented as a tidvector.

So the total cost = O(Tid_vector length)

Then we scan this & directly check for that i^{th} position in O(1) → Look up is O(1)

↳ Combination of Apriori, Sampling, Partition &, ~~ORACLE~~ etc.

Page No.

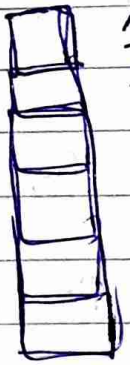
⇒ ARMOR

↳ we might add or remove items in the candidate → meaning of Update C in First Pass algo part.

We have account of frequent itemsets along with their partial support values ~~as~~ as soon as an item becomes a part of candidates set.

Why second pass? Suppose an item ~~became frequent~~ came into candidates from partition number 3. But it might be present in 1 & 2 also.

~~It wasn't added until 2 but it would be having support < minsup.~~



So now we will traverse the whole database & to get the actual count of such itemsets. And if the actual count < minsup, ~~add it to f~~ else delete it.

So C will convert into f

↓ ↓

candidates frequent itemset.

⇒ Implementation

Concise Mining

⇒ Post-mining Rule Pruning Schemes

- **Improvement**: Eg: $ABC \rightarrow D$
  $AB \rightarrow D$. → like a sub-rule of the above.

  If the confidence of these 2 are very similar (very less difference), then we need not report both of them.

⇒ Constrained Association Rules are good if the user has some constraints in mind. Else it is not good.

⇒ Maximal Frequent Itemsets

↳ If we are o/p these positive borders, then its subsets will also be frequent → so do not o/p them. But this will have drawbacks.

Suppose ABCD is max. freq itemsets and the rules are formed by breaking then say $AB \rightarrow CD$. But if we remove the subsets, then we will lose their support & could not make these rules.

⇒ Closed Itemsets

Support(A) = Support(AB) means both are appearing in the same records..

⇒ Mining Real World Data

Binning - Try to make sure that bins are equi-width & equi-depth

↓ same length range.

↓ similar frequency of values in every range.

Relational → directly suitable for clustering and classification

Transactional → directly suitable for association rules mining.

But to convert transactional to relational, we would want to normalize (3NF & ... forms) & then could be converted to relational.

For classification or clustering, we need to do joins of the various relational tables. to get a single table.

Multi-dimensional -

DATE
PAGE No

Distributed - Like relational but the data is distributed across multiple nodes. (maybe across data centres.)

- Maybe you can bring all the scattered data to a central location & do data warehousing.
- Maybe mining separately & doing analysis by getting the results to a central location. (just like partition.)

Distributed

- Homogenous — same data at all locations — Run like partition
- Heterogenous — different data at diff. locations (enterprise) — Do join & bring to central location

⇒ IR Concepts

Boolean queries → OR Query, AND Query etc.

Ranked queries → Among the retrieved query documents, which are top ranked and so on. (Trying to rank the output.)

DATE
PAGE No

⇒ Text & Web data models → semi-structured or unstructured data models.

Q How to do data mining on above data model?

Ans i) Bag of Words:

{ word 1 : count 1 , word 2 : count 2 , word 3 : count 3 ---- } .

Bag of words model

ii) (count 1, count 2, ---- count n) where words are features and their order is fixed so we need to just keep the counts.

↳ Vector space Model

Vector of n dimension.

iii) Cosine similarity - To find similarity b/w 2 documents.

$$\frac{d1 \cdot d2}{|d1||d2|}$$

dot product of 2 vectors

product of norm of 2 vectors.

dot product:

$X = (x1, x2, --)$

$Y = (y1, y2, --)$

$X \cdot Y = (x1 y1 + x2 y2 + --)$

norm $\Rightarrow |X| = \sqrt{(x1)^2 + (x2)^2 + (x3)^2 + ---}$

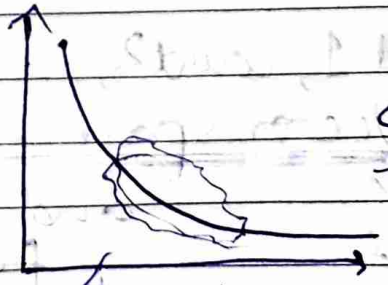
iv) Entropy - Treat count of words as probabilities

$$H = -\sum p_i \log p_i$$

v.) Stop words - words like "and", "the", "to" occur too frequently and are irrelevant in classification or clustering. so we need to remove them

Zipf Law → Most freq word → Rank 1
Second " " " → Rank 2

strong/universal law for all languages.



So now we can use some threshold based on freq & Rank to decide what all words to consider.
Mostly the middle words are used. which lie in this range maybe

vi) Stemming - Take the root word & stem out others.
Eg: Sing, sung, song, singing
Just take ~~song~~.

helps in reducing vocabulary space.

vii) TF-IDF - Tells about what are the most important/best words in the documents.

viii) Inverted Index - For each word, we store what all documents contain it.
Eg: word 1 → doc 1, doc 3, doc 4, etc.
(Inverted bcz usually we store what all words are present in each doc.)

⇒ Search Engine Architecture

i) • Crawling - Getting links from links and follow those documents recursively

Normalized Tokens

ii) • Normalization → same word can be written in different formats like U.S.A, USA, United States of America etc. So we need to normalize & get them i.e. get single token for them & use it

iii) Ranking

a) TF-IDF → Term frequency — Inverse document frequency.

mean(tf) ~~sum(tf)~~ → freq of all words.

DATE
PAGE No

TF - num. of times the word appears in the doc.

df - how many docs contain that word.

good keyword - Appear in this doc many times (tf high) but should not occur too many times in other docs. (df is low)

b) Page Rank

If the docs linked to say doc D are highly ranked, then D is highly ranked.

Talk about inlinks

Page-rank = SUM [rank(y) / |outlinks|]

- used recursively
- all the inlinks
- outlinks of that particular doc

Initially, all docs have rank = 1.