Clustering

A cluster: a collection of data objects, similar to one another within the same cluster, dissimilar to the objects in the other clusters.

Similarity between objects: Often expressed in terms of a distance measure $d(x,y)$

* Every distance function should be a metric, it should satisfy the following conditions:

a/ $d(x,y) \geq 0$ b/ $d(x,y) = 0$ iff $x = y$ c/ $d(x,y) = d(y,x)$

d/ $d(x,z) \leq d(x,y) + d(y,z)$

Similarity measures:

* Euclidean Distance: the shortest distance between two points (Pythagoras theorem)
  - $\left((x_2 - x_1)^2 + (y_2 - y_1)^2\right)^{1/2}$
* Manhatten Distance: the sum of absolute differences between points accross all dimensions.
  - $|x_2 - x_1| + |y_2 - y_1|$

  As number of dimensions ↗ it is better to use manhatten (Low computation cost)
* Minkowski Distance: the generalized form of Euclidean and Manhatten Distance.
  - $\Sigma(|x_i - y_i|^p)^{1/p}$ ⇒ For Interval-scaled variable / continuous
  - L1 norm is Manhatten, L2 norm is Euclidean, Minkowski is LP
* Cosine similarity: measures the cosine of the angle between two vectors and determines if they are moving in the same direction.
  - $\cos(x,y) = x \cdot y / \|x\| * \|y\|$
  - if $\theta = 0°$, the x and y are in the same direction $[\cos(0) = 1]$
  - If $\theta = 90°$, the x and y vectors are ⊥ to each other so they are dissimilar $[\cos(90) = 0]$

How to measure similarity for attributes:

* Binary or Nominal: Similarity = 1 if two objects are equal, 0 otherwise.
  Dissimilarity = 0 if two objects are equal, 0 otherwise.
* Interval-based: use the Normalized distance
* Ordinal: Similarity = 1 - Dissimilarity
  Dissimilarity = rank(x) - rank(y) / max. rank.

DBSCAN algorithm: A density based algorithm. It separate clusters of high density from clusters of Low density Based on a distance measurement and a minimum number of points.

* **DBSCAN** (Epsilon, Delta 'min-pt')
  - Assign id = -1 to all the points, if id = 0 then it is noise cluster
  - build the first cluster (i = 1)
  - Loop on all the data points and check if the point is -1 (not clustered)
    - Expand cluster is a recursive function:
      * If the neighborhood of p < delta and is not a core point
        → Mark p as noise (p = 0)
      * Else loop on all the neighborhood points
        → Assign the point to a cluster i
        → Check the other neighborhood points, if it finds some core point:
          - Check if it's not a noise point and not clustered
            → Assign it to the same cluster.
      * When finished return true.
  - Increase by 1.

* **Type of points:**
  - Core point: which has at least M points in its neighborhood (always belongs in a dense regin ↑ have at least minPts number of data points within their eps distance)
  - Border point: its neighborhood contains less than M points, or it is rechable from a core point. < MinPts within Eps, but in the Neighborhood of a core point
  - Outlier Point is not a core point and not close enough to be rechable from a core point. located in low density areas

* The outlier points are eliminated
* Core points that are neighbors are connected and assigned to the same cluster
* The border points are assigned to each cluster.

* **Pros:**
  - Does not require to specify number of clusters.
  - Resistant to noise
  - Clusters can take any irregular shape.
  - Can identify outliers easily.

* **Cons:**
  - Computationally complex
  - Hard to set the parameters.

Can we use Silhouette score in DBSCAN? This technique measures the separability between clusters. First an average distance is found between each point and all other points in a cluster. Then it measures the distance between each point and each point in other clusters. We substract the two average measures and divide by whichever average is Larger. (how close the points to each other in a cluster?) and (how much distance is between the clusters?) It ranges from -1 to 1.

Silhouette score of 0 suggest overlapping clusters. (on the boundary between 2 data clusters)
1: far from the neighboring cluster / -1: probably assigned to the wrong cluster

Elbow method: Used to determine the number of clusters in a data set.
It consists of plotting the explained variation as a function of the number of clusters. (The optimal K is on the elbow point on the graph)

Hierarchical clustering:

Agglomerative : from bottom to top

1. Each object is considered a single element cluster (leaf)
2. 2 similar clusters are ~~com~~ combined to a new bigger cluster (nodes)
   a) Find closest pair of clusters min $D(c_i, c_j)$
   b) Merge the clusters c_i, c_j into a new cluster c_{i+j}
   c) Remove c_i, c_j from C and add c_{i+j} instead.
3. Repeats until all points form one single big cluster (root)

* Single Linkage: min (distance (a, b)) shortest distance between pts in two clusters
* Complete Linkage: max (distance (a, b)) farthest " " "
* Average Linkage: 1/ab average of all the distances between any point of A and any point of cluster B.

Dendogram: clustering tree. leaves are the objects / internal nodes are the clusters.

KMeans Clustering:

1. Select a 'K' number of clusters
2. Randomly select 'K' distinct data points.
3. measure the distances between points and clusters and assign the nearest points to its nearest cluster.
4. Calculate the mean of each cluster
5. Repeat ~~…~~ everything and cluster using the mean values.

Hyperparameters:
- K number of clusters
- initial value 'seed'
- Distance measure.

KMEANS vs Hierarchical:
* KMeans tries to put the data into a number of clusters you tell it to.
* Hierarchical tells you pairwise what two things are most similar.

Frequent Pattern Mining:

What are the association rules? It shows the frequently occuring patterns.
An association rule has 2 parts: an antecedent (if) and a consequent (then)
"If a customer buys bread, he's more likely to buy milk"

Algorithm: Step 1: Label all points as core, border or noise points
Step 2: Remove or Eliminate all noise points
Step 3: For every core point p that has not yet been assigned to a cluster
a) Create a new cluster with the point p and
b) Add all points that are density-connected to p.
Step 4: Assign each border point to the cluster of the closest core point.

How to choose MinPoints: The minimum number of data points within the radius of a neighborhood for the neighborhood to be considered a cluster.
→ It removes the outliers.
→ It should be greater or equal to the dimensionality (d) of your dataset
→ If the dataset is noisier, we should choose a larger value of minPts.

How to choose Epsilon: Maximum Distance between two pts to be considered as a neighboring point.
→ use KNN ↗ Eps ↘ No of clusters
→ Calculate the average dist. between each point and its k-nearest neighbors
→ Plot number of pts on X and average distances on y axis
→ The resulting graph should be ↘. optimal eps is the elbow point.

Differences between DBSCAN and KMENS:

KMENS	DBSCAN
* Num. of clusters should be specified	* Can not be specified
* Can handle large datasets well	* Cannot handle large datasets well
* Does not work well will outliers and noise	* works well.
* Clusters have spherical shape and must have the same feature size.	* Clusters formed are arbitrary in shape and may not have same feature size.

Internal Evaluation of clusters: It is when the result is evaluated based on the data that was clustered itself.
- These methods usually assign the best score that produces clusters with high similarity within a cluster Silhouette.

External Evaluation of clusters: clustering results are evaluated based on data that was not used for clustering.

accuracy = TP + TN/All
precision = TP/TP + FP "How precise the model is"
recall = TP/TP + FN "How actual positive there is in the model"
Jaccard index = TP/TP + FP + TN
F1 score = ((Precision x recall) / (Precision + recall)) x 2 "balance between both"

Association rules generation: Input: Frequent Itemsets and thresholds

- Set the frequent itemset Z and the confidence threshold (params)
- Generate all nonempty subsets for each frequent itemset I
- Loop on nonempty subset and check:
  - If confidence > min threshold: & support ≥ min support threshold
    → Output the rule subset recommends (Subset - Item)
- Then we add the item to the set of candidates.

Support: popularity of an item (item / total items)

Confidence: Indicate the probability of both the antecedent and the consequent appear in the same transaction: (item A + item B / item A)

Lift: How much often the antecedent and the consequent of a rule occur together
(Conf (A → B) / Supp (B))

Item and Itemset:
* A set of items together is called an itemset.
* An itemset that occurs frequently and it satisfies a min. threshold value for support and confidence is called a frequent itemset.

How can we generate frequent itemset:

Aprion Algorithm: It is based on the concept that a subset of a frequent itemset must also be a frequent itemset.

1. Each item is taken as a 1-itemset candidate
2. Take the candidate with Support ≥ min threshold and prune the others.
3. A 2-itemset is generated by forming a group of 2 by combining items with ~~some~~ themselves.
4. The 2-itemset candidates are pruned using min-sup threshold.
5. Form 3-itemsets using join and prune step. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.
6. 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min-supp criteria.

The algorithm is stopped when the most frequent itemset is achieved.

FP-Growth: Finding frequent itemsets without candidate generation

- If the FP-Tree is a single Path or empty
- Loop on all combination of nodes:
  - report all pattern C U P ← prefix path subtrees
    conditional FP Tree
- Else loop on all the FPT elements
- Generate pattern P_i = {item} union P and report it
- Construct the conditional FPT from conditional prefix path After removing infrequent items

APPRIORI or FP-Growth: FP-Growth is more efficient.

- The frequent pattern is generated without the need for candidates generation
- Faster
- It represents the data in the form of a tree: frequent pattern Tree
- Database is stored in a compact version in memory.

Confidence-based pruning:

If a rule $X \rightarrow Y$ does not satisfy the confidence threshold, then any rule $X' \rightarrow Y - X'$ where $(X' \subset X)$ must not satisfy the confidence threshold as well.

what is Simpson's paradox:

Certain association between two variables in a population emerges, disappears or reverses when the Data is divided

A **frequent itemset** is called:

* maximal: if none of its superset are frequent.
* closed: if none of its superset has the same support

Classification

*** What is SVM hyperparameter; If we set a high "C" hyperparameter what will happen?**

The C parameter tells SVM how much you want to avoid missclassifying each training example. C ↗ : the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
C ↘ : the optimizer will look for a larger margin separating hyperplane, even if that hyperplane missclasssifies more points.

*** For KNN, how does K influence bias and variance?**

A small value of K will increase the effect of noise and a large value makes it computationally expensive.
K ↘ : makes our classifier so sensitive to noise but also may lead to the overfitting problem. (high variance)
K ↗ : High bias. risk of underfitting.

***Hyperparameter and parameter:** Parameters are the configuration model, which are internal to the model. Hyperparameters are the explicitly specified parameters that control the training process. Parameters are essential for making predictions. Hyperparameters are essential for optimizing the model.

***What is cross-validation? Why we use it?**

A Resampling procedure used to evaluate machine learning models on a limited data sample. It has a single parameter called K that refers to the number of groups that a given data sample is to be split into.
Usually called K-fold cross-validation.

- Split the dataset into K groups: training and testing
- fit the model on the training set and evaluate it on the testing set.
- Uses evaluation scores to summarize the skill of the model.

***What is the Difference between logistic regression and Linear regression?**

Logistic Regression	Linear Regression
• It provides a constant output / categorical	• It provides a continuous output / numerical
• used to describe the relationship between one dependent binary variable and indep. variables.	• Attempts to model the relationship between two variables by using a linear eq: $Y = a + bX$
• By using Sigmoid fct: $1/1-e^{-x}$	• Y is the dependent variable. X is the explanatory variable.
• It can take any value and range it between [0, 1]	

*What is regularization? A technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.
A way of finding a good bias-variance value by tuning the complexity of the model

*What is Lambda? What will happen if we increase λ?

Lambda is a hyperparameter that controls the amount of regularization applied to a model.

$$f(\theta) = \underbrace{\sum_{(x,y)\in S} (y - m^{\theta}(x))^2}_{\text{empirical error}} + \underbrace{\lambda \|\theta\|^2}_{\text{Regularization term}}$$

Increasing λ reduces overfitting → reduces variance → increases bias.

$\lambda \nearrow$: high bias → risk of underfitting (model is simple)

$\lambda \searrow$: high variance → Risk of overfitting (model is complex)

Bias variance tradeoff : Finding the right balance of values (bias & variance). A tradeoff model not too general not too specific

*What is Bias and variance?

Variance : • The difference between a random variable and its expected value.
• Determines how stable the model is.
• The amount that the estimate of the target function will change given different training data.

Bias : • The average difference between the expected prediction and the true value
• Determines How generic the model is

*What is Discriminant Function? Bayes Rule in Discriminant fct?

• A Dimensionality reduction technique.
• In case of K classes, classification can be seen as an implementation of K discriminant functions.
• Multiplication of likelihood probability and prior probability $g_1(x) = P(x|C_1)P(C_1)$

Shape of Discriminant Function in logistic Regression:
• Sigmoid S shape in range 0 and 1. Sigmoid $1/1+e^{-x}$
• discriminant $f^{ct} < 0$: we classify the instance to belong to class 0
• discriminant $f^{ct} > 0$: " " " " 1.

*Naive Bayes Classifier: A probabilistic classifier.

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

Postenior prob. probability the the instance X is labeled with class C_i

Likelihood: prob. that an arbitrary instance X belong to C_i

Prior probability: prob. than an arbitrary instance is labeled with class C_i

evidence: probab. that the instance X is seen regardless of its class

→ For X predict C_i for which $P(C_i|X)$ is maximal.

Recommendation Techniques

Implicit feedback: Information obtained about users by watching their natural interaction with the system $\Rightarrow$ No burden on the user. Boolean values 0 or 1

Explicit feedback: Rating items, provide a list of prefered items, ...

The recommendation Task: Given
- U, I and ϕ ← user feedback
- X_{user}, X_{item} caracteristics.
- Some background knowledge K
- Goal: learn a model $\hat{\phi}: U \times I \longrightarrow \mathbb{R}$ such that $acc(\hat{\phi}, \phi, T)$ is max.
- T: a set of "unseen (test) user-item pairs.
- X^{item}, X^{user}, K are often unknown.

Rating Prediction:
- From explicit feedback How Steve will rate the movie Titanic?
- $\hat{\phi}(u,i)$ - predicted rating of the user u for an item i

Item Recommendation:
- From implicit feedback
- $\hat{\phi}(u,i)$ - predicted likelihood of a "positive" implicit feedback of the user u for an item i

which movie(s) would Steve buy / see?

Types of RS:

* Knowledge-Based: based on knowledge about users's needs and preferences
- X^{item}, K, X^{user}.

* Content-based: Lean user's interests based on the features of items previously rated by the user, using supervised machine Leaning techniques
- X^{item}, ϕ

* Collaborative-filtering: recognize similarities between users according to their feedbacks and recommend objects prefered by the like-minded users.
- ϕ (also X^{item} and/or X^{user} can be utilized)

Neighborhood-based collaborative Filtering: Recommendation $\hat{\phi}(u,i)$ for user u on item i using ϕ.

* user based: $\hat{\phi}(u,i)$ computed using feedback given by K most similar users

* Item based: $\hat{\phi}(u,i)$ computed using feedback given by K most similar items.

Item Recommendation: what is the likelihood of an item i being liked by the user u?

* A simple **K-nearest-neighbor** approach
  - user-based: an average similarity of most similar users which liked the item i

$$\hat{\phi}_{ui} = \frac{\sum_{v \in N_i^{u,k}} sim(u,v)}{k}$$

  - item-based: an average similarity of most similar items liked by a user u

$$\hat{\phi}_{ui} = \frac{\sum_{j \in N_u^{i,k}} sim(i,j)}{k}$$

* only (implicit) feedback ϕ is available. Users and items represented by sparse vectors
  - Cosine-vector similarity (sim_{cv}) ← similarity measure.

Rating Prediction: How would the user rate an item?

* user's / item's ratings are **biased**: * optimistic, pessimistic users
  * items rated above or below average.
* **Mean-centered** rating prediction
  - user-based: $\hat{\phi}_{ui} = \overline{\phi}_u + \frac{\sum_{v \in N_i^{u,k}} sim(u,v) \cdot (\phi_{vi} - \overline{\phi}_v)}{\sum_{v \in N_i^{u,k}} |sim(u,v)|}$

$$\overline{\phi}_u = \frac{\sum_{i \in I_u} \phi(u,i)}{|I_u|}$$

  - item-based: $\hat{\phi}_{ui} = \overline{\phi}_i + \frac{\sum_{j \in N_u^{i,k}} sim(i,j)(\phi_{ju} - \overline{\phi}_j)}{\sum_{j \in N_u^{i,k}} |sim(i,j)|}$

$$\overline{\phi}_j = \frac{\sum_{u \in U_i} \phi(u,i)}{|U_i|}$$

* What similarity measure to use? sim_{cv} doesn't take into account the mean and variances of ratings

⇒ Pearson-correlation similarity.

$$sim_{pc}(u,v) = \frac{\sum_{i \in I_{uv}} (\phi_{ui} - \overline{\phi}_u)(\phi_{vi} - \overline{\phi}_v)}{\sqrt{\sum_{i \in I_{uv}} (\phi_{ui} - \overline{\phi}_u)^2 \sum_{i \in I_{uv}} (\phi_{vi} - \overline{\phi}_v)^2}}$$

$$sim_{pc}(i,j) = \frac{\sum_{u \in I_{ij}} (\phi_{ui} - \overline{\phi}_i)(\phi_{uj} - \overline{\phi}_j)}{\sqrt{\sum_{u \in I_{ij}} (\phi_{ui} - \phi_i)^2 \sum_{u \in I_{ij}} (\phi_{uj} - \overline{\phi}_j)^2}}$$

A latent space representation: Map users and items to a common latent space where dimensions of factors represent
- items' implicit properties
- users' interest in items' hidden properties.

Factorization models: ϕ represented as user - item matrix $\Phi^{n\times m}$ n users, m items.

Principal Component Analysis (PCA): Dimensionality reduction method, often used to reduce the dimensionality of large datasets.
- Transform data to a new coordinate system. (variances by any projection of the data lies on coordinates in ↘ order)

Singular Value Decomposition (SVD): Factorization of ϕ into 3 matrices.

$$\Phi = W^{n\times k} \Sigma^{k\times k} H^{n\times kT}$$

* column vectors of W are orthonormal eigenvalues of $\Phi \Phi^T$
* column vectors of H are " " of $\Phi^T \Phi$

Σ contains eigen values of W in ↘ order

⇒ PCA and SVD are computed algebratically.

MF- Rating Prediction:
- only $W^{n\times k}$ and $H^{m\times k}$
- k - the number of factors $\Phi^{n\times m} \approx \hat{\Phi}^{n\times m} = WH^T$
- predicted rating $\hat{\phi}_{ui} = w_u h_i^T$

loss function needs to be minimal → $err(\hat{\phi}, \phi, D) = \sum_{(u,i)\in D} e_{ui}^2 = \sum (\phi_{ui} - \hat{\phi}_{ui})^2$

$$= \sum (\phi_{ui} - w_u h_i^T)^2$$

* The objective function : regularization term λ to prevent overfitting

$$f(\hat{\phi}, \phi, D) = \sum_{(u,i)\in D} (\phi_{ui} - w_u h_i^T)^2 + \lambda (\|W\|^2 + \|H\|^2)$$

⚠ Find W and H such that, given λ, $f(\hat{\phi}, \phi, D)$ is minimal

MF with SGD: updating parameters iteratively for each data point ϕ_{ui} in the opposite direction of the gradient of the objective function at the given point until a convergence criterion is fulfilled.

$$w_u(u,i) \longleftarrow w_u - \alpha \frac{\partial f}{\partial w_u}(u,i) = w_u + \alpha (e_{ui} h_i - \lambda w_u)$$

$$h_i(u,i) \longleftarrow h_i - \alpha \frac{\partial f}{\partial h_i}(u,i) = h_i + \alpha (e_{ui} w_u - \lambda h_i)$$

α a learning rate

Hyperparameters : $\underbrace{k}_{\text{Knowledge background}}$, iters (the max number of iteration), α, $\underbrace{\lambda}_{\text{Regularization}}$, Σ^2 } eigenvalues of W in decreasing order.

Biased MF: • Much of the observed variation in the rating values are due to effects associated with either users or items. Known as biases.

• The intuition behind this is that some users give higher ratings than others, and some items received high ratings than others.

$$b_{ui} = \mu + b'_u + b''_i$$

μ: average rating across the whole D

b', b'': vectors of users and items biases

• $\hat{\phi}_{ui} = \mu + b'_u + b''_i + w_u h_i$

regularization

• $f(\hat{\phi}, \phi, D) = \sum_{(u,i) \in D} (\phi_{ui} - \mu - b'_u - b''_i - w_u h_i)^2 + \lambda (||W||^2 + ||H||^2 + b'^2 + b''^2)$

Loss function

Biased MF with SGD: • $\mu = \frac{\sum (u,i) \in D}{|D|}$ $\quad b' \leftarrow (\bar{\phi}_{u1}, \dots, \bar{\phi}_{un})$ $\quad b'' \leftarrow (\bar{\phi}_{i1}, \dots, \bar{\phi}_{im})$

• $\mu \longleftarrow \mu - \frac{\partial f}{\partial \mu}(u,i) = \mu + \alpha e_{ui}$

$b' \longleftarrow b' + \alpha (e_{ui} - \lambda b')$

$b'' \longleftarrow b'' + \alpha (e_{ui} - \lambda b'')$

MF - item recommendation: To predict a personalized ranking score $\hat{\phi}_{ui}$

* How the item i is preferred to other items for the user u.
* To find W and H such that $\hat{\Phi} = W H^T$, $\hat{\phi}_{ui} = w_u h_i^T$

problem: positive feedback only.

Bayesian Personalized Ranking: it involves pairs of items (user-specific order of two items) to come up with more personalized rankings for each user.

* The primary task of personalized ranking is to provide a user with a ranked list of items.
* Like in any Bayesian approach, we have likelihood function, prior probability and posterior probability

"The Bayesian formulation of finding the correct personalized ranking for all items $i \in I$ is to maximize the posterior probability where Θ represents the parameter vector of an arbitrary model class"

logistic Sigmoid

* Likelihood: $p(i >_u j \mid \Theta) := \sigma(\underbrace{\hat{x}_{uij}}(\Theta)) = \frac{1}{1 + e^{-x}}$

relation between user u, item i and item j

generally calculated by using an MF model

* Prior probability: $\underbrace{\sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2}_{\text{BPR-OPT}} = \arg\max_\Theta p(\Theta, >)$

BPR-OPT vs AUC: Area under the ROC curve (AUC)

* probability that the ranking of a randomly drawn pair is correct.

- $\text{AUC-OPT} = \sum_{(u,i,j) \in D_p} \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \|\theta\|^2$
- $\text{BPR-OPT} = \sum_{(u,i,j) \in D_p} \ln \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \|\theta\|^2$

The Cold start problem: Arises when not enough collaborative information is available

Context-aware recommendation: Any additional information, besides X^{user}, X^{item}, ϕ and K, that is relevant for the recommendation.

Cons and Pros:

Knowledge-based:
- pros: No cold-start, deterministic
- cons: Knowledge engineering needed, static

Content-based:
- pros: No collaborative information needed.
- Cons: content is needed, cold-start for few users

Collaborative filtering:
- pros: No user nor item attributes needed
- cons: cold-start for few users and items

Frequent Pattern Mining

If we decrease support what happens for lift ?
It will increase because Lift = confidence / support.

* Aprion Algorithm: To Generate the frequent itemsets using candidate generation. "All nonempty subsets of a frequent itemset must also be frequent"
  - Given a set of Items I and a min support threshold.
  - Initialize the cardinality K.
  - Loop on all itemsets with support >= minSupp
    * Generate candidate (frequent itemset, K+1)
    * K+1
  - Return all frequent itemsets

Support: item/total items
popularity of an item

Confidence (X → Y)
N° of transaction containing X and Y / N° of tr. cont. X
(How often items in Y appear in transaction that contain X)

Generate candidate (F: frequent itemset, K: cardinality)
- Get the items proved that they are frequent in which the union of it equal to K.
- Check if the subset is frequent and belongs to F : Monotonicity
- Return the generated candidate.

Difference between Eclat and Apriori :
- Aprion is usable with Large datasets and Eclat is better suited to small and medium datasets.
- Aprion scans the original dataset, whereas Eclat uses the transpose (vertical)

FP-Tree : A compact representation of the transactional data in a tree structure.
- The Tree is constructed by taking each itemset and mapping it to a path in the tree one at a time. frequently occuring items will have better chances of sharing items
- Get the frequent pattern by mining the tree recursively.
- Construct a conditional FP Tree, formed by count of itemsets in the path (itemsets meeting the threshold)

FP- Growth Algorithm: A recursive procedure using
- P : prefix path subtrees
- C : conditional FP Tree

- If the FP-tree is a single Path or empty
- Loop on all combinations of nodes : → report all patterns ⊆ UP
- Else Loop on all the FPT elements
  - Generate pattern $P_i = \{item\} \cup P$ and report it
  - Construct the conditional FPT from conditional prefix path after removing infrequent items

which is best apriori or FP-Growth ? FP-Growth is more efficient than apriori. It's an improvement.
- The frequent pattern is generated without the need for candidate generation.
- It is faster, the pairing of items is not done in the algorithm.
- The database is stored in a compact version in memory.

* How to generate association rules : ("if - then" relations

$A \Rightarrow C$ ← consequent $A \cap C = \emptyset$

Antecedent

Given D, I, σ and θ, the goal is to find all association rules $A \Rightarrow C$ such that $supp_D(A \Rightarrow C) \geq \sigma$

$conf_D(A \Rightarrow C) \geq \theta$

- Generate non empty subsets for each frequent itemset I
- Loop on nonempty subsets and check
  If Confidence > min threshold (θ) $Conf_D(S \Rightarrow (I-S)) \geq \theta$
  → Output the rule subset recommends (Subset_ Item)
  $\boxed{S \Rightarrow (I - S)}$

Lift $(A \Rightarrow B)$: $Conf(A \Rightarrow B) / Supp(B)$

measure How often the antecedent and the consequent occur together

Lift = 1 : No association

Lift > 1 : More likely to be associated

Lift < 1 : Unlikely " "

Simpson's Paradox: it occurs when a correlation between pairs of itemsets (antecedents and consequents of rules) appearing in different groups of data leads to being disappeared or reversed when these groups are combined.

Confidence Based Pruning:

If a rule $X \rightarrow Y$ does not satisfy the confidence threshold.

Then any rule $X' \rightarrow Y - X'$ where $(X' \subset X)$ must not satisfy the confidence Threshold.

Maximal itemset : None of its supersets are frequent.

Closed itemset : None of its supersets have the same support

* How much association rules can we generate ? $A \Rightarrow C$

$$3^{|I|} - 2^{|I|+1} + 1 = \underbrace{3^{|I|}}_{\text{Number of different splits int A and C}} - \underbrace{2(2^{|I|} - 1)}_{\text{2x rules with empty antecedents or consequents}} - \underbrace{1}_{\text{A rule with empty antecedent and consequent (empty rule)}}$$