Prediction : Supervised Learning

Prediction

Bias and Variance :

* **Variance** : ◦ The difference between a random variable and its expected value.
◦ When a model has a high variance, then it emplies that the model gets a different data points which it hasn't learnt then it cannot make right prediction as a result thus **overfitting the data** (determines how stable the model is

* **Bias** : difference between the expected prediction and the value which we are trying to predict. between our predicted value and the true value
When a model has a high bias then it implies that the model is too simple and does not capture the complexity of the data thus **underfitting the data**.
determines how generic the model is.

Tradeoff : about balancing and finding a sweet spot between errors due to bias and variance (underfitting vs overfitting dilema)
Sum : we will get the squared error we cannot calculate the actual bias and variance for a predictive modeling problem (CZ we don't know the true labeling

Overfitting and underfitting :

- Underfitting : when the models performs poorly on the training data. high bias and low variance.
- Overfitting : When the model performs well on the training data but does not perform well on the evaluation data. Low bias and high variance.

Regularization and Lambda :

* Regularization is a way of finding a good bias-variance value by tuning the complexity of the model. $f(\theta) = \sum_{(x,y)\in S_{tr}} (y - m_\theta(x))^2 + \lambda \|\theta\|^2$
* Lambda is a hyperparameter that controls the amount of regularization applied to a model.
* Increasing Lambda reduces overfitting, which reduces the variance, but it increases the bias.
* If Lambda value is too high
  - high bias → model will be simple (risk of underfitting the data)
* If Lambda value is too Low
  - high variance → model will be more complex (risk of overfitting the data)

Discriminant Function :

Bayes Rule in Discriminant Function :

- 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 $1/1+e^{-x}$
- if the descriminant function < 0 we classify the instance to belong to class 0
- if the descriminant function > 0 we classify the instance to belong to class 1.

Naive Bayes classifier:

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object

1. Convert the dataset into a frequency table (total of yes and No for each column)
2. Create likelihood table by finding the probabilities (total of yes No of row / whole total of yes N
3. Use Naive Bayesian equation to calculate the posterior probability for each class.
4. The class with the highest posterior probability is the outcome of prediction.

Equation: $\underbrace{P(C_i|x)}_{\text{Posterior probability of class}} = \underbrace{P(x|C_i)}_{\text{Likelihood}} * \underbrace{P(C_i)}_{\text{prior probability}} / \underbrace{P(x)}_{\text{Evidence of x}}$ ← dependent feature vector

Random Forest:

- Randomly select "K" features from the total features.
- calculate the node using the best split point.
- Split the node into daughter nodes.
- Repeat first 3 steps until a specific number of nodes has been reached.
- Repeat previous steps to build forest to create "n" number of trees.
- Get a prediction result from each tree.
- Calculate the votes for each predicted target.
- Select the prediction result with the most votes as the final prediction

Hyperparameters:

- n_estimators = number of trees in the forest.
- max_features = max number of features considered for splitting a node.
- max_depth = max number of levels in each decision tree.
- min_samples_split = min number of data points before the node is split.
- min_samples_leaf = min number of data points allowed in a leaf node.

Decision Tree vs Random Forest:

Random Forest consist of multiple single trees each based on a random sample of the training data. They are typically more accurate than single decision trees.

- Trees are unpruned. While a single decision tree is often pruned, a random forest tree is fully grown and unpruned, and so, naturally, the feature space is split into more and smaller regions.
- Trees are diverse. Each random forest tree is learned on a random ~~tree~~ sample, and at each node, a random set of features are considered for splitting. Both mechanisms create diversity among the trees.
- For applications in classification problems, Random Forest algorithm will avoid the overfitting problem.
- For each classification and regression task, the same random Forest algorithm can be used.

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.

K Nearest Neighbor (KNN):

- It is one of the easiest classification algorithms.
- KNN is based on similarity measurements between dataset instances, where K is a critical hyper-parameter for this algorithm, indicates how many nearest neighbors it should consider for the classification of a new instance.

How the value of K affects KNN algorithm? :

- A small value of K will increase the effect of Noise, and a Large value makes it computationally expensive.
- The smaller value for K, not only makes our classifier so sensitive to noise but also may lead to the overfitting problem.
- Large values of K may also lead to underfitting.
- If we calculate accuracy for training dataset, KNN with K = 1, we get 100% as the value are already seen by the model and a rough decision boundary is formed for K = 1 (leads to overfitting).
- When we calculate the accuracy for the unseen data, it performs really bad that is the training error would be very low but the actual error would be very high.
- So it would be better, if we chose and optimal K. To choose an optimal K, we should plot a graph between error and K value for the unseen data that is the test data.
  => we choose the value where the error is the lowest.

Logistic Regression:

- It provides a constant output | Categorical.
- used to describe the relationship between on dependent binary variable and independent variable
- by using the Logistic sigmoid function (S shape)
- Sigmoid : $1 / 1 + e^{-x}$
- It can take any value and range between [0, 1]

Linear Regression:

- It provides a continuous output | numerical
- attempts to model the relationship between two variables by using a linear equation
- $Y = a + bX$
- Y : is the dependent variable | X : is the explanatory variable.
- if $b > 0$: x and y have positive relation else has negative relation.

Lambda and Theta in linear regression :

- As Lambda tends to infinity, the coefficients will tend towards 0 and the model will be just a constant function.
- If Lambda = 0, we effectively have no regularization.
- Theta-0 and Theta-1 represent the parameters of the regression line.
- $y = ax + b$ a is slope "Theta-1", b is y-intercept "Theta-0")

C Parameter in SVC :

- The C parameter tells the SVM optimization how much we want to avoid misclassifying each training example.
- For large values of C, the optimization will choose a smaller-margin hyperplane. If the hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.
- For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

Hyperparameters vs Parameters:

* Hyperparameters cannot be learned within the estimator directly. Parameter is a more general term.
* Parameters are the configuration model, which are internal to the model.
* Hyperparameters are the explicitly specified parameters that control the training process.

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 cutomer buys bread, he's more likely to buy bread."

* If $S_1 \subset T \in D$ implies $S_2 \subset T$ then S_1 and S_2 are associated in T.

Association rules generation:

- Set the frequent itemset Z and the confidence threshold(params)
- Generate all nonempty subsets for each frequent itemset I
  - Eg For I: $\{1,3,5\}$, nonempty are $\{1\}\{3\}\{5\}\{1,3\}\{1,5\}\{3,5\}$
- Loop on nonempty subset and check:
  - If confidence > min threshold:
    - → Output the rule subset recommends (Subset - Item)
    - → Eg: $\{1,3,5\} - \{3\} = 3 \rightarrow (1,5)$
- Then we add the item to the set of candidates

Support / Confidence / Lift:

- Support: popularity of an item (item/ total items) quantitative measure
- confidence $(A \rightarrow B)$: (item A + item B / item A) qualitative measure
  - Indicate the probability of both the antecedent and the consequent appear in the same transaction.
  - an item B is bought if item A is bought.
- Lift $(A \rightarrow B)$: $(Conf(A \rightarrow B) / Sup(B))$
  - measure how much often the antecedent and the consequent of a rule occur together
  - Sale of B will increase when A is sold.
  - Lift = 1: there is no association
  - Lift > 1: more likely to be bought together
  - Lift < 1: unlikely to be bought together.

Items and itemset in pattern mining:

- A set of items together is called an itemset (2 or more items)
- An itemset that occurs frequently is called a frequent itemset.
- It's frequent if it satisfies a minimum threshold value for support and confidence.

APRIORI algorithm: Algorithm

- Provide (Set of items (I), min support threshold σ)
- Initialize the cardinality K
- Loop on all itemset with support >= σ:
  - call function **generate candidate** (F, K+1)
  - Increase K by 1.
- When finished return the frequent item K.

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

Additional: 2 steps Join & Prune.

- Set a threshold support level. Say 50% for example.
- Create a frequency table of all the items that occur in the set (Join)
- Prune to include only those items having a threshold support level over 50% (Prune)
- Make pairs of every item and repeat the previous step to eliminate the non-frequent itemset

FP-TREE:

- 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.
- Join the frequent pattern generated from the conditional FP trees.

FP-growth algorithm: A recursive procedure using:
- P: prefix path subtrees.
- C: Conditional FP Tree.

Algorithm:

- IF the FP-Tree is a single Path or empty
- Loop on all combination of nodes:
  → report all patterns C ∪ P.
- ELSE loop on all the FPT elements
- generates pattern P_i = {item} ∪ P and report it
- construct the conditional FPT from conditional prefix path after removing infrequent items.

APRIORI or FP-Growth:

FP-Growth is more efficient than APRIORI, it is an improvement to APRIORI.

- The frequent pattern is generated without the need for candidate generation.
- It is faster, the pairing of items is not done in the algorithm.
- It represents the data in the form of a tree called a frequent pattern tree.
- The database is stored in a compact version in memory.

Monotonocity:

- Each subset of frequent itemset is frequent too.
- None of the supersets of a non-frequent itemset are frequent.

Confidence-based pruning:

If a rule $X \rightarrow Y$ does not satisfy the confidence threshold. $2^{|I|}$ - 2 association rules meeting the θ threshold for each itemset $|I|$
then any rule $X' \rightarrow Y - X'$ where $(X' \subset X)$ must not satisfy the confidence threshold aswell.

How much association rules can we generate from I ?

$A \Rightarrow C$ s.th $\sup(A \Rightarrow C) \geq \sigma$
$\text{cof}(A \Rightarrow C) \geq \theta$

$$3^{|I|} - 2^{|I|+1} + 1 = 3^{|I|} - 2\left(2^{|I|} - 1\right) - 1$$

- $3^{|I|}$: number of different splits into A and C
- $2\left(2^{|I|} - 1\right)$: 2X rules with empty antecedents or consequents
- -1: A rule with empty antecedent and consequent (empty rule)

* A frequent itemset is called:

- Maximal if none of its supersets are frequent
- Closed if none of its supersets have the same support.

Clustering : Unsupervised Learning

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

Aim of clustering: to group a set of data objects into clusters.

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

* Ideally, every distance function should be a metric, it should satisfy the following conditions:
  1) $d(x,y) \geq 0$
  2) $d(x,y) = 0$ iff $x = y$
  3) $d(x,y) = d(y,x)$
  4) $d(x,z) \leq d(x,y) + d(y,z)$

Types of Data in Cluster Analysis:

* **Interval-scaled variables:**
  - Continuous measurements of a roughly linear scale
  - The measurement unit can affect the cluster analysis, to avoid dependence on the measurement unit, we should standardize the data:

$$s_f = \frac{1}{n}\left(|x_{1f} - m_f| + |x_{2f} - m_f| + \cdots + |x_{nf} - m_f|\right)$$ The mean absolute deviation

where : $m_f = \frac{1}{n}(x_{1f} + x_{2f} + \cdots + x_{nf})$

$$\Rightarrow z_{if} = \frac{x_{if} - m_f}{s_f}$$ standardized measurement (z-score)

  - One popular group of distance measures for interval-scaled variables are **Minkowski distances**

$$d(x,y) = \sqrt[p]{\sum_{i=1}^{d} (x_i - y_i)^p}$$

  - if $p = 1$, the distance measure is **Manhattan** distance $d(x,y) = \sum_{i=1}^{d} |x_i - y_i|$
  - if $p = 2$, the distance measure is **Euclidean** distance $d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$

* **Binary Variables:** - has only two states : 0 or 1.
  - A contingency table for binary data

		1	0	Sum
Y	1	a	b	a+b
	0	c	d	c+d
Sum		a+c	b+d	$m = a+b+c+d$

  - Simple matching coefficients : (invariant similarity, if the binary variable is symmetric)

$$d(x,y) = \frac{b+c}{m}$$

  - Jaccard coefficient (noninvariant similarity, the binary variable is asymmetric): $d(x,y) = \frac{b+c}{a+b+c}$

* **Nominal variables:**
  - A generalization of the binary variable, it can take more than 2 states
  - Simple matching : m: # of matches, p: total # of variables

$$d(x,y) = \frac{p-m}{p}$$

$$s(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

  - Use many binary variables : create a new binary variable for each of the M nominal states

* **Ordinal Variables:** - Can be discrete or continuous.
  - Order is important.
  - Can be treated like interval-scaled: 1. replacing x_{if} by their rank $r_{if} \in \{1, \ldots, M_f\}$
    2. map the range of each variable into $[0,1]$ by replacing the ith object in the fth variable by $z_{if} = \frac{r_{if} - 1}{M_f - 1}$
    3. compute the dissimilarity using method for interval-scaled var.

$x, y \in D$ are ranks

$$d(x,y) = \frac{|x-y|}{n-1}$$

* **Variables of Mixed Types** : - A database may contain all the 6 types of variables (binary, Nominal, Ordinal, Interval, Set, Sequence)
(qualitative, quantitative, complex)

$$- d(x,y) = \frac{\sum_{i=1}^{m} \omega_i \, d(x_i, y_i)}{\sum_{i=1}^{m} \omega_i}$$

$$\omega_i = \begin{cases} 1, & \text{if } x_i \neq NA \neq y_i \\ 0, & \text{otherwise} \end{cases}$$

- if i is binary or nominal : $d = 0$ if $x_i = y_i$, otherwise $d = 1$ / Similarity : 1 if two are equals, 0 otherwise
- if i is interval-based : use the normalized distance
- if i is ordinal : $d(x,y) = \text{rank}(x) - \text{rank}(y) \,/\, \text{max. rank}$.

* **Complex Data Types** : * All objects considered in data mining are not relational
⇒ complex types of data
* Eg. : spatial data, multimedia data, genetic data, time-series data, text data and data collected from World-Wide-Web.

Clustering Methods :

* **Partitioning Methods** : * construct a partition of a dataset D of n objects into a set of K clusters such that each cluster contains at least one object, and each object belongs to exactly one cluster.
* Given a K, find a partition of K clusters that optimizes the chosen partitioning criterion.

K-means

- each cluster is represented by the center of the cluster
- **Input** : the num. of clusters K, and a data base of n objects
- **Algorithm** :
1. partition object into K nonempty subsets/clusters.
2. compute a seed points as the **centroid** (the mean of the objects in the clusta) for each cluster in the current partition.
3. assign each object to the cluster with the neaest centroid.
4. go back to Step 2, stop when there are no more new assignments.

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

K-medoids

- each cluster is represented by one of the objects in the cluster.
- **Input** : " "
- **Algorithm**
1. arbitrarily choose K objects as the initial **medoids** (representative objects)
2. assign each remaining object to the cluster with the neaest medoid
3. select a non medoid and replace one of the medoids with it if this improves the clustering.
4. Go back to step 2, stop when there are no more new assignments.

Elbow method: The elbow method is used in determining the number of clusters in a dataset. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use.
→ The optimal K is on the elbow point of the graph.

*Clustering is distance metric dependent.
* K-means converges with every step to the minimum distortion metric

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.
* Measure properties expected in a good clustering: * Compact groups
  * Well separated groups.

Silhouette score: Calculates the average distance to elements in the same cluster with the average distance to elements in other clusters.

- Objects with a high silhouette value are considered well clustered, objects with a low value may be outliers.

1. pick range of k values.
2. plot silhouette for each value of K [ideal = 1, worst =
3. calculate silhouette coefficient for every point

$$s(i) = b(i) - a(i) / \max\{a(i), b(i)\}$$

a(i) = average distance from pt i to another pts in the same cluste
b(i): average distance from pt i to another pts in the next nearest cluster.
a(i) should be much less than b(i)
1: far away from neighboring clusters
0: boundary between 2 clusters
-1: assigned to wrong clusters.

External evaluation of clusters: clustering results are evaluated based on data tha 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"
- random index = TP + FN / All "measure the percentage of correct decisions made by the algorithm"
- Jaccard index = TP / TP + FP + TN
- F1 - score = ((Precision x Recall) / (Precision + Recall)) x 2 "balance between both"

* **Hierarchical methods**: - construct a hierarchy of clustering, not just a single partition of objects
  - The number of clusters K is not required as an input.
  - Use a **distance matrix** as clustering criteria.
  - A termination condition can be used (eg. number of clusters)
  - The hierarchy of clustering is often given as a clustering tree, also called a **dendogram**: - leaves of the tree represent the individual objects
    - internal nodes of the tree represent the clusters.

Agglomerative: (bottom-up)
- place each objects in its own cluster (a singleton)
- merge in each step the two most similar clusters until there is only one cluster left or the termination condition is satisfied.

divisive: (top-down)
- start with a big cluster containing all the objects
- divide the most distinctive cluster into small clusters and proceed until there are n clusters. or the termination condition is satisfied.

Agglomerative clustering: 1. Each object is considered as a single-element cluster (leaf)
2. 2 similar clusters are 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. Repeat until all points forms one single big cluster (root)

- Three widely used ways of defining the **inter-cluster distance**:

nearest neighbor - single linkage: min (distance (a,b)) shortest distance between pts in two clusters

furthest neighbor - complete linkage: max (distance (a,b)) farthest distance between pts in two clusters.

unweighted pair-group average - average linkage: 1 / ab

* **Density Based methods**: * Clustering methods like partitional methods or hierarchical clusters are not effective in finding clusters of arbitrary shapes. The density-based clustering method is efficient in finding the clusters of arbitrary shapes; also prevents outliers and noise.
* If clusters are created by using the density of neighborhood objects, then the **DBSCAN** algorithm is used
* If clusters are created according to a density function, then **DENCLUE**
* whereas **OPTICS** is a density-based which generates an enhanced order of the data collection structure.

DBSCAN algorithm: It separates clusters of high density from clusters of low density. Based on a distance measurement and a minimum number of points.

* Input: N objects to be clustered and global parameters ε and MinPts.
* Output: Clusters of objects.
* Algorithm: 1. Arbitrary select a point P.
  2. Retrieve all point density reachable from P wrt ε and MinPts.
  3. If P is a core point a cluster is formed.
  4. If P is a border point, then there is no point that is density-reachable and DBSCAN moves to the next point
  5. This process is continued until all the points are processed.

- The type of each point is determined:
  i. Core point which has at least M points in its neighborhood
  ii. Border point which its neighborhood contains less than M points, or it's reachable from a core point.
  iii. Outlier point is not a core point and not close enough to be reachable from a core point.

Correlation in similarity: * Correlation coefficients are used to measure how strong a relationship is between 2 var. (e.g. age and blood pressure)
* There are several types of corr coeff. Kendall, Spearman, the most popular is Pearson's. [-1, 1]
* 1 = data objects are perfectly correlated.
* -1 = Data objects are not correlated

Can we use Silhouette score in DBSCAN?: clustering isn't the same. Silhouette score assumes that all points are assigned to a cluster. It can happen that inside DBSCAN plot, you'll see that there is an outlier points which are not assigned to any cluster. In this case, single point not assigned to any cluster is enough to make a significant difference in the silhouette scores.

Metrics for measuring DBSCAN's performance:

- Almost no internal metric can handle DBSCAN results propely.
- Noise is not a cluster and almost all metrics assume the data is partioned into clusters. The result will appear much worse than it is. There is DBCV evaluation, it supposedly designed for density-based cluster evaluation.

DBSCAN parameters and how to guess them:

- Epsilon: maximum distance between two points to be considered as neighboring points.
  - Use KNN
  - calculate the average distance between each point and its K-nearest neighbors
  - Plot number of points on the X axis and average distance on the Y axis.
  - The resulting graph should be decreasing. There should be a point of the curve (elbow point) that contains the optimal epsilon.
  - Run the algorithm using different values of K and compare results.
- Minimum Points: minimum number of neighboring points to be considered as core points
  - If the dataset is large or noisy, choose a large value of MinPts.
  - MinPts should be greater than or equal to the dimensionality of the dataset.

Difference between DBSCAN and KMENS:

KMENS	DBSCAN
* Number of clusters should be specified	* Number of clusters can be not specified
* Can handle large datasets well.	* Cannot handle large datasets well.
* Does not work well with outliers and noisy datasets.	* work well with outliers and noisy datasets
* Clusters have spherical shape and must have same feature size.	* Clusters formed are arbitrary in shape and may not have same feature size.

Recommendation Techniques

Compared with other fields:

- **Information Retrieval:** * unstructed data, various topics (IR) vs. repositories focused on a single topic (RS)
  * relevant content for the query (IR) vs. relevant content for the user (RS)
- **Machine Learning:** * hardly measurable, subjective evaluation criteria (RS) besides some classic, objective evaluation measures (ML)
- **Human - Computer Interaction:** * RS should convince the user to try the recommended items.
  * Clear, transparent and trustworthy system logic.
  * Provide details about recommended items and opportunity to refine recommendations.

* Users and Items:

* Users: • age, income, education, nationality, ...
• preffered sport, hobbies, ...
• set of users U, caracteristics $X^{user}: U \to A^{user}$ (user attributes)
→ sensitive information, hard to obtain.

* Items: • set of items I
• item attributes A^{item}
• item characteristics: $X^{item}: I \to A^{item}$ movies: title, genre, year, director, ...
→ quite costly to obtain

* User feedback:

* feedback values $F \in \mathbb{R}$ observed on $D \subset U \times I$ $\quad \phi: D \to F$
* **Implicit feedback:** • information obtained about users by watching their natural interaction with the system. (view, listen, scroll, bookmark, save, purchase, ...)
  boolean value: 0 or 1
  • No burdon on the user.
* **Explicit feedback:** • rating items on a rating scale
  • scoring items
  • ranking a collection of items
  • pairwise ranking of two presented items
  • provide a list of preferred items.

* The recommendation Task:

* Given • U, I and ϕ
• X^{user}, X^{item}
• some background knowledge K.
* To learn a model $\hat{\phi}: U \times I \to \mathbb{R}$ such that $acc(\hat{\phi}, \phi, T)$ is max.
• usually $F = \{1\}$ in case of implicit feedback.
• T: a set of "unseen" (or test) user-item pairs
$$T \subseteq (U \times I) \setminus D$$
• acc is the accuracy of $\hat{\phi}$ w.r.t ϕ measured on T

* It looks like a simple prediction Task, however X^{user}, X^{item} and K are often unkno

Rating Prediction: • from explicit feedback

How will Steve rate the movie Titanic more likely?

- $\hat{\phi}(u,i)$ - 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 (ranking score) of the user u for an item i.

which movie(s) would Steve more likely buy / see ?

Types of RS: • The recommender system mainly deals with the likes and dislikes of the users.

- Its major objective is to recommend an item to a user which has a high chance of liking or is in need of a particular user based on his previous purchases.
- RS finds the similarity of the users or items from whom the recommendation has to be made with that of all the users or items which are present in the datasets.
- We find the pattern of likes and dislikes having the highest similarity
- Then we make use of that pattern to suggest whether an item or place or movie or book has to be suggested or not.

Knowledge-based: Recommendations are based on knowledge about users' needs and preferences

- X^{item}, K, X^{user}

Content-based: Learn user's interests based on the features of items previously rated by the user, using supervised machine learning techniques.

- X^{item}, ϕ

Collaborative-filtering: recognize similarities between users according to their feedbacks and recommend objects preferred by the like-minded users.

- ϕ (also X^{item} and/or X^{user} can be utilized)

Hybrid.

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**

$$\mathcal{N}_u^{i,K} = \underset{I'}{\operatorname{argmax}} \sum_{j \in I', v \neq u} sim(u,v)$$
$$I' \subseteq I_u, |I'| = K$$

$$* I_u = \{v \in U \mid \phi(v,i) \text{ is defined on } D\}$$

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

$$\mathcal{N}_u^{i,k} = \underset{I'}{\operatorname{argmax}} \sum sim(i,j) \qquad I_u = \{j \in I \mid \phi(u,j) \text{ is defined on } D\}$$

Item recommendation:

What is likelihood of an item i being liked by the user u?

- a simple **K-nearest-neighbor** approach $\phi(u,i) \rightsquigarrow \hat{\phi}_{ui}$; $I_u \cap I_v \rightsquigarrow I_{uv}$, $U_i \cap U_j \rightsquigarrow U_{ij}$
  - * 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 the user u

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

assume that 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

$$\hat{\phi}_{ui} = \bar{\phi}_u + \frac{\sum_{v \in N_i^{u,K}} sim(u,v) \cdot (\phi_{vi} - \bar{\phi}_v)}{\sum_{v \in N_i^{u,K}} |sim(u,v)|}$$

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

  - * item-based

$$\hat{\phi}_{ui} = \bar{\phi}_i + \frac{\sum_{j \in N_u^{i,K}} sim(i,j) \cdot (\phi_{uj} - \bar{\phi}_j)}{\sum_{v \in N_u^{i,K}} |sim(i,j)|}$$

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

Pearson-correlation similarity:

- sim_{cv} doesn't take into account the mean and variances of ratings.

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

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

Example: * user-based:

- $U_{Titanic} = \{Joe, Ann, Mary\}$, $\mathcal{N}^{Steve,2}_{Titanic} = \{Mary, Ann\}$
- $\bar{\phi}_{Steve} = \frac{11}{3}$, $\bar{\phi}_{Mary} = \frac{12}{4}$, $\bar{\phi}_{Ann} = \frac{13}{4} = 3.25$
- $\hat{\phi}_{ST} = \bar{\phi}_S + \frac{S_{pc}(Steve, Mary) \cdot (\phi_{MT} - \bar{\phi}_M) + S_{pc}(S, A) \cdot (\phi_{AT} - \bar{\phi}_A)}{|S_{pc}(S,M)| + |S_{pc}(S,A)|}$

* item-based:

- $I_{Steve} = \{Pulp\ Fiction, Iron\ Man, The\ Mummy\}$, $\mathcal{N}^{Titanic,2}_{Steve} = \{I, M\}$
- $\bar{\phi}_T = \frac{10}{3}$, $\bar{\phi}_I = \frac{11}{3}$, $\bar{\phi}_M = \frac{9}{3} = 3$
- $\hat{\phi}_{SI} = \bar{\phi}_I + \frac{S_{pc}(Titanic, I) \cdot (\phi_{SI} - \bar{\phi}_I) + S_{pc}(T, M) \cdot (S_{SM} - \bar{\phi}_M)}{|S_{pc}(T,I)| + |S_{pc}(T,M)|}$

Factorization Models: collaborative filtering algorithms

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

(user feedback)

Principal Component Analysis (PCA):

- Transform data to a new coordinate system.
  - * variances by any projection of the data lies on coordinates in decreasing order
  - dimensionality reduction method, often used to reduce the dimensionality of large datasets, by transforming a large set of variables into a smaller one that still contains most of the information

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

$$\Phi = \underbrace{W^{n \times k}}_{} \underbrace{\Sigma^{k \times k}}_{} \underbrace{H^{n \times k^T}}_{}$$

- $W^T W = I$, $H^T H = I$
- Column vectors of W are orthonormal eigenvectors of $\Phi\Phi^T$
- Column vectors of H are orthonormal eigenvectors of $\Phi^T\Phi$
- Σ contains eigenvalues of W in descending order.

$\Rightarrow$ PCA, SVD computed algebraically.

W

H,

$\Rightarrow$

MF - rating prediction

recommendation task:
- to find $\hat{\phi}: U \times I \to \mathbb{R}$ such that $acc(\hat{\phi}, \phi, T)$ is maximal
  - * acc is the expected accuracy on T
  - * training $\hat{\phi}$ on D such that the empirical loss $err(\hat{\phi}, \phi, D)$ is minimal

a simple, **approximative** MF model
- only $W^{n \times k}$ and $H^{m \times k}$
- k - the number of factors

$$\Phi^{n \times m} \approx \hat{\Phi}^{n \times m} = W H^T$$

- predicted rating $\hat{\phi}_{ui} = w_u h_i^T$

the Loss function $err(\hat{\phi}, \phi, D)$
- squared loss $err(\hat{\phi}, \phi, D) = \sum_{(u,i) \in D} e_{ui}^2 = \sum_{(u,i) \in D} (\phi_{ui} - \hat{\phi}_{ui})^2 = \sum_{(u,i) \in D} (\phi_{ui} - w_u h_i^T)^2$

The objective function
- regularization term $\lambda > 0$ to prevent overfitting
  - * penalizing the magnitudes of parameters

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

The task is to find parameters W and H such that, given λ, the objective function $f_\lambda(\hat{\phi}, \phi, D)$ 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.
- updating the vectors w_u and h_i for the data point $(u,i) \in D$

$$\frac{\partial f}{\partial w_u}(u,i) = -2(e_{ui} h_i - \lambda w_u)$$

$$\frac{\partial f}{\partial h_i}(u,i) = -2(e_{ui} w_u - \lambda h_i)$$

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

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

where $\alpha > 0$ is a **learning rate**

Hyperparameters: k, i_{max} (the max number of iteration), α, λ (regularization), Σ^2 ← eigenvalues of W in descending order.

knowledge background

⇒ we can find the "best" hyper-parameters combinations using e.g. grid-search

Biased MF: baseline estimate

- user - item bias: $b_{ui} = \mu + b'_u + b''_i$
  - μ - average rating across the whole D
  - b', b'' - vectors of users and item biases, respectively.
- Prediction: $\hat{\phi}_{ui} = \mu + b'_u + b''_i + w_u h_i$
- Objective function to minimize:

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

Biased MF with SGD:

- Initialize average and biases

$$\mu = \frac{\sum_{(u,i) \in D}}{|D|}$$

$$b' \leftarrow (\overline{\phi}_{u1}, \dots, \overline{\phi}_{un})$$

$$b'' \leftarrow (\overline{\phi}_{i1}, \dots \overline{\phi}_{im})$$

- Update average and biases:

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

$$b' \leftarrow b' - \frac{\partial f}{\partial b'}(u,i) = b' + \alpha (e_{ui} - \lambda b')$$

$$b'' \leftarrow b'' - \frac{\partial f}{\partial b''}(u,i) = b'' + \alpha (e_{ui} - \lambda b'')$$

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 $\hat{\Phi} = WH^T$

$$\hat{\phi}_{ui} = w_u h_i^T$$

problem: positive feedback only.

Bayesian Personalized Ranking: Formulation of the problem:
- $>$ - the unknown preference structure (ordering)
  - the derived pairwise ranking data D_p
- θ - parameters of an arbitrary prediction model
  - In case of MF, $\theta = W \cup H$

* $p(\theta) = \prod_{\theta \in \Theta} \sqrt{\frac{\lambda}{2\pi}} e^{-\frac{1}{2}\lambda \theta^2}$
* **Likelihood**: - assume users' feedbacks are independent
  - " ordering of each pair is independent
  - using the ranking scores $\hat{\phi}$

$$p(i >_u j \mid \theta) = p(\hat{\phi}_{ui} - \hat{\phi}_{uj} > 0) = \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) = \frac{1}{1 + e^{-(\hat{\phi}_{ui} - \hat{\phi}_{uj})}}$$

* Maximum a posteriori estimation of θ:

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

BPR-OPT vs ALS:

Area under the ROC curve (AUC)

- probability that the rankings of a randomly drawn pair is correct
- $AUC\text{-}OPT = \sum_{(u,i,j) \in D_p} \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \|\Theta\|^2$ smoothed AUC objective function with regularization of parameters
- $BPR\text{-}OPT = \sum_{(u,i,j) \in D_p} \ln \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \|\Theta\|^2$

The Cold-start problem: Arises when not enough collaborative information is available
- New user or new item

Possible solutions:
- recommend popular items, "predict" global average, ...
- utilize item attributes

Context-aware recommendation:

Context is any additional information, besides X^{user}, X^{item}, ϕ and K, that is relevant for the recommendation

* time, location, companion (when, where, and with whom the user wants to watch some movie).

Evaluating RS:

Offline:
- No interaction with real users, need to simulate user behaviour
- Low cost, short time

User studies:
- observing test subjects' behaviour in the system.
- questionnaries.

Online Evaluation:
- redirect a small part of the traffic to an alternative recommendation engine.
- risky.

Pros and Cons:

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,

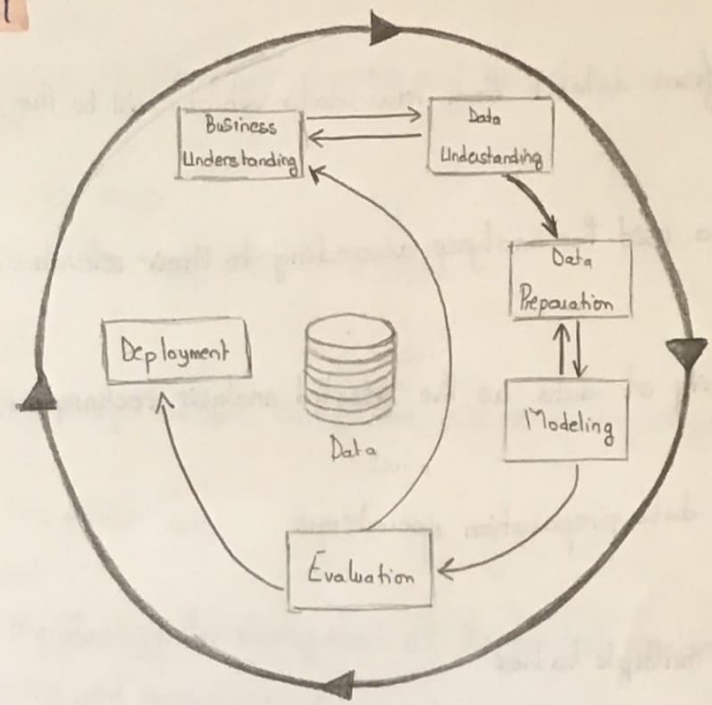
Collaborative-filtering:
- pros: No user nor item attributes needed
- Cons: cold-start for few users and items

Data P...

DM in Smart Systems

Introduction

The process model



I. Business Understanding

The aim is to understand the needs of a client, the requirements and business objectives, convert the objectives to data mining goals, uncover important factors influencing these outcomes and prepare a preliminary plan for achieving the gals.

Generic tasks of this phase are

① Determine business objectives
- understanding the client's needs from the business perspective

② Assess situation
- investigation of facts about the factors influencing the project

③ Determine data mining goals
- determine the project objectives in technical terms

④ Produce project plan
- preparation of a detailed plan to reach the project objectives

II. Data Understanding:

The aim is to collect initial data, get familiar with data and identify the quality of data as well as detect subsets interesting to form some hypotheses.

Generic tasks of this phase are:

① Collect initial data
- acquisition of data listed in resources and understanding them as well as initial data preparation steps

② Describe data
- examination of the surface properties of acquired data

③ Explore data
- querying, visualization and reporting data directly addressing the data minning goa

④ Verify data quality
- examination of the quality of data

III. Data Preparation

The aim is to construct the final dataset from raw data which will be the input for the modeling tool.

① Select data
- decision on the data used for analysis according to their relevance to the specified objectives

② Clean data
- improve the quality of data as the selected analysis techniques require

③ Construct data
- perform constructive data preparation operations

④ Integrate data
- integrate data from multiple tables

⑤ Format data
- mainly syntactic modifications of data to be suitable for modeling tools

IV. Modeling

In this phase, modeling techniques are selected and their parameters are tuned to optimal values

① Select modeling technique
- selection of the actual modeling technique

② Generate test design
- generation of a procedure to validate the model and test it's quality

③ Build model
- run the modeling technique to build models

④ Assess model
- interpretation, evaluation, comparison and ranking of models according to the evaluation criteria from a data mining perspective

Evaluation

In this phase, the model is thoroughly evaluated to be certain that it achieves the business objectives, the whole process is reviewed and next steps are determined.

Generic tasks of this phase are

① Evaluate results
- evaluate of the achievements of business objectives

② Review of process
- summarization of the whole process and detecting important factors which could be overlooked.

③ Determine next steps
- decision of the next steps to be made.

VI. Deployment:

In this phase, the knowladge gained during the process is organized, eventually, presented for the customers.

Generic tasks of this phase are

① Plan deployment
- creation of the strategy for deployment of the project results into the business

② Plan monitoring and maintenance
- preparation of the maintenance strategy

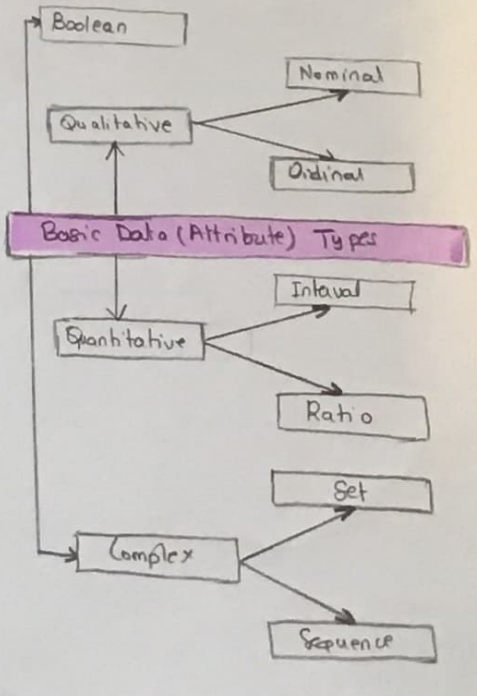
③ Produce final report
- final documentation of the project

④ Review project
- experience documentation

Data types and Attributes:

Data
- raw measurements
  symbols, signals ...
- corresponding to som attributes
  height, grade, heartbeat, ...

Attribute domain
- expresses the type of an attribute
  number, string, sequence, ...
- by the set D of admissible values
  - called the domain of the attribute
    height up to 3 m, grade from A to F, ...
- and certain operations allowed on D
  $1 < 3$, "A" $\geq$ "C", "John" $\neq$ "Jon", ...



Objects, Records, Observations

Object

- A collection of recorded measurements (attributes) representing an entity of observation (context, meaning)
  - e.g. a student represented by ID (nominal), age (quantitative), sex (boolean), English proficiency (ordinal), list of absolved courses (set), yearly scores from IQ tests (time-series), ...
- $X = (x_1, x_2, \ldots, x_m) \in D_1 \times D_2 \times \ldots \times D_m$
- Objects with mixed types of attributes can be transformed to objects having boolean or/and -quantitative attribute types
  - Be aware of the possible loss of information!
  - ~~when you propose some~~

Final Remarks:

- Domain knowledge is important
  - need to be an understanding between data scientists and domain experts and all the stakeholders.
- Choose the right tool for the job
  - often the right tool depends on the data you have (to deal with)
- ~~Talking~~

Data Preprocessing

Normalization / Standarization:

- Normalization is a scaling technique such that when it is applied the features will be rescaled so that the data will fall in the range of $[0, 1]$.
- Sandarization is a scaling technique, rescaling with propeties of a standard normal distribution with mean = 0 and standard deviation = 1 ranges between $[-1, 1]$

How to select features:

- The aim is to get rid of redundent and irelevent features.
- Strategies: embedded, filter, wrapper approaches.
- Filter approach is to select attribute with low pairwise correlation, and if two attributes are 100% correlated, we can get rid of any of them.

Encoding:

- Ordinal encoding (Label-Encoding): Each unique category value is assigned an integer value ("red" = 1, "green" is 2, bl " is 3).
- One-Hot Encoding: transform to binary values each, "1" value is placed in the binary variable for the color and "0" values for the other colors (0,0,1 = red, ..., 0, 1,0 = green)
- Dummy Variable Encoding: transform to bin .ry but using 2 bits only (red = 1,0) (green = 0, 1) (blue = 1, 1)

DataTypes:

Qualitative:

- Nominal Attributes: Categorical with no order (Black, Brown, Red)
- Ordinal Attributes: Categorical with ordu (alphabets, scale)
- Binary Attributes: (Boolean)
  - Symmetric: Both values are equally important (Gender)
  - Asymmetric: Both values are not equally important (Result).

Quantitative:

- Numeric: measurable quantity, represented in integer
- Discrete: zip code
- Continuous: height, weight, ...

- use only the attributes with values
- modify the learning alg. to accept and work with missing values

- Rmove objects: Use only the objects with values for all attributes
- Make estimates: fill with estimates based on values on the other objects

Discretization: the conversion of quantitative values into nominal or ordinal values

Normalization: A type of conversion done in order to have different attributes expressed on the same scale. (Carried out for each attribute individually)

* Standarization: subtracts the avg of the attributes values and then divides the result by the standard deviation of the values. => The values will have avg 0.0 and std deviation of 1.0
* Min-max scaling: Converts num. values to values in a given interval. For $[0.0, 1.0]$:
  Subtract the smallest value from all the values in the set
  Then divide the new values by the amplitude $[\text{new max} - \text{new min}]$

Data transformation.

* Apply Log fct to the values of a predictive attribute
* Conversion to absolute values: values magnitude is more important than its sign

Dimensionality reduction: when the N° of attributes in a dataset is very large, the data space becomes very sparse and distance between objects becomes very similar

* Attribute aggregation: we replace a group of attributes with a new attribute: a combination
  (multidimensional scaling) Project the original dataset into a new, lower-dimensional space but keeping the relevant information. **PCA**: principal componant Analysis
* Attribute Selection: select a subset of the attributes that keeps most of the information present in the original dataset.
  - Filters: Look for simple individual relations between the predictive attribute values and the target attribute. If high predictive attribute values are related with class A and Low values with class B. This predictive attributes recieves a high ranking position.
    Peauson correlation
  - Wrappers: explicitly use a classifier to guide the attribute selection process. Select the set of predictive attributes that provides the highest predictive performance for the classifier
  - Embedded: internal procedure of a predictive algorithm. Decision Tree induction algorithm can perform embedded attribute selection.

FP-Growth: It compresses the transactional data into a so-called "FP-Tree" structure

* To build an FP-Tree only two passes of the data are required.
  - In the first pass, all frequent items and their support are found
  - In the second pass, items in each transaction are processed in a decreasing order according to their support

  → Items from the transaction are added to the tree following a path with respect to their order.
* Finally a so called "header table" of items is created, with pointers to the first appearance of those items
* The support of a given item = $\sum$ counts in the linked nodes.

Maximal frequent itemset if **None** of its **supersets** is **frequent**
Closed frequent itemset if **None** of its **supersets** **has the same support.**

→ If an itemset is maximal then it is also closed

Association rules mining:

Association rule: An association rule is an implication $\underbrace{A}_{\text{antecedent}} \Rightarrow \underbrace{C}_{\text{consequent}}$, A and C itemsets that do not share common items.

How valid the rule is according the the data ?

$Supp(A \Rightarrow C) = supp(A \cup C)$ a measure of the strength of an association rule $A \Rightarrow C$

Association rule mining: Given a set of all available items I, transactional data T and threshold values min_sup and min_conf, It aims at finding those association rules generated from I for which support in T is >= min sup and confidence in T is >= min_conf.

$3^{|I|} - 2^{|I|+1} + 1$ association rule itemsets $= \underbrace{3^{|I|}}_{\text{different splits in A and C}} - \underbrace{2(2^{|I|} - 1)}_{\text{empty antecedents or consequents}} - \underbrace{1}_{\text{empty rule}}$ $\quad 2^{|I|} - 1$

→ Because the support of each rule has to meet a min_sup threshold, only those rules which the joined antecedent and consequent pairs form a frequent itemset are considered.

⇒ 1. Minining the frequent itemsets that meet the min_sup threshold
2. Generate association rules meeting the min_conf threshold from the frequent itemsets.

Monotonicity: 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.

Cross-support pattern: If a pattern contains low-support items and high-support items

Confidence: A conditional probability that a randomly selected transaction including all the items in the antecedent of the rule will include all the items in its consequent.

Lift (X => Y) = Conf (X => Y) / support (Y)

To measure the effect of the antecedent on the consequent of the rule

- Lift > 1 : - positive correlation between the antecedent and the consequent
  - The occurence of the antec. has a ⊕ effect on the occu of the conseq.
- Lift < 1 : - Negative correlation
  - " " " has a ⊖ " " "
- Lift ≃ 1 : N. correlation
  - " " " almost no effect on " "

Simpson's Paradox: Certain correlations between pairs of itemsets (antec. and conseq of rules) appearing in different groups of data may disappear or be reversed when these groups are combined

Distance Measures: an approach to associate a number with the similarity (and dissimilarity)

* Lift = Confidence / support ; Measures How often the antecedent and the consequent occur together.

  Lift = 1 : No association
  Lift > 1 : More Likely to be associated
  Lift < 1 : Unlikely to be associated

* Apriori: To generate frequent itemsets using candidates generation
  "All non empty subsets of a frequent itemset must also be frequent"

Algorithm:
- Given a set of Items I and a min supp. threshold
- Initialize the cardinality K
- Loop on all itemsets with support >= min threshold
  * Generate candidate (frequent itemset, K+1)
  * K+1
- Return all frequent itemsets

Frequent itemset mining: Given a set of all variables items I, transactional data T and a threshold value min_supp, frequent itemset mining aims at finding those itemsets called "frequent itemsets", generated from I in which support in T is at least min_sup (≤ min_sup)

Support: the ratio between the number of transactions (rows in T) in which the given itemset is present and the number of all transactions in the data.

#Setting up min_support: A hyper-parameter.

* if min_sup is very low ⟶ large number of itemsets that would be too specific to be considered "frequent". These itemsets might apply in too few cases to be useful
* If min_sup is very high ⟶ Small number of itemsets which would be too generic to be useful. The resulting information would probably not present new knowledge for the user.

Motonocity rules:
* If an itemset is frequent then each of its subsets are frequent too
* If an itemset is infrequent then none of its supersets are frequent

⇒ To avoid the generation and testing of the support of all the possible itemsets.

General principal: Traverse the itemset Lattice in order to search for those itemsets with support >= than min_sup.

Algorithm: Input T: transactional dataset

min_supp: minimum support threshold.

Initialize the cardinality K = 1 and stop = false

Repeat

Select all frequent itemsets of length K (support >= min supp)

If there are not two frequent itemsets of length K then

stop = true

else

K = K + 1

until Step

Candidate itemset: The itemset that it makes sense to generate and measure the support of in each iteration of the algorithm:

* Get the items proved that they are frequent in which the union of it equals to K.
* Check if the subset is frequent and belongs to itemset: Monotonicity
* Return the generated candidate.

Eclat:

- Main obstacle of Apriori is in every step it needs to scan the whole transactional dataset in order to count the support of candidati.itemsets.
- Counting support is computationally expensive if the database is large and does not fit into memory.

=> One way to store transactional data is "vertical format" in which, for every item, a list of transaction identifiers in which that item occurs is stored. => The support of an itemset is counted as the ratio of the cardinality of its transaction identifier set to the number of transactions in the database.

Algorithm: First, the frequent itemsets of size K = 1 are detected and stored together with their corresponding TID-sets.
Then, each frequent itemset of size K is expended by one item, resulting in an itemset of size K+1.
K is incremented and the process iterates until there are no candidates to expand.

Support of a candidate itemset is the cardinality of intersection of the TID-sets of the corresponding itemsets.

Distance Measures: an approach to associate a number with the similarity (and dissimilarity) between 2 objects ~~is to use distance measures~~

* Difference between values of common Attribute type

  * Quantitave: $d(a,b) = |a-b|$

  * Qualitative: • ordinal values: $d(a,b) = (|pos_a - pos_b|)/(n-1)$

    n: number of different values.

    • Nominal values: $d(a,b) = \begin{cases} 1, & \text{if } a \neq b \\ 0, & \text{if } a = b \end{cases}$

⚠ Qualitative attributes can be transformed into quantitative attributes. An object represented by a vector of m quantitive attributes can be mapped to an m-dimensional space. The more similar two objects are, the closer they are when mapped in this space

Distance measure for objects with quantitative attributes:

* Minkowski distance: m-dimensional objects p and q with quantitative attributes

$$d(p,q) = \sqrt[r]{\sum_{k=1}^{m} |p_k - q_k|^r}$$

* Manhatten distance

* Euclidean distance

Non-Conventional Attributes: sequences (text, biological, time series), images, sound, video

* Sequences: - Hamming distance (nº of positions at which the corresponding characters or symbols in two strings are different)

Clustering Evaluation:

Internal evaluation: looks for compactness inside each cluster and/or separation

* External: uses the extend information such as class label, if available, to define the quality of the cluster in a given partition

Jaccard external index: It evaluates how uniform the distribution of the objects in each cluster is with respect to the class label. $J = m_{11} / (m_{01} + m_{10} + m_{11})$

m_{11}: same cluster, same label; m_{01}: ≠ cluster, same label; m_{10}: same cluster, ≠ labels; m_{11}: other clusters with different labels

K-means: *A centroid: a prototype of all objects in a cluster, for example the avg of all the objects in the cluster.

* By default, Euclidean distance, assuming all attributes are quantitative.

* hyperparameters: distance measure, K number of clusters.

Algorithm:
1. Input D dataset, d distance measure, K # of clusters
2. Define K initial centroids
3. Repeat
4. | Associate each instance in D with the closest centroid according to the chosen distance measure d
5. | Recalculate each centroid using all instance from D associated with it
6. until No instances from D change of associated centroid

instances is of
g instances in the cluster lies within

hyperparameters ? Using Elbow method: the lineplot of the within-groups sum of squares for different values of k
*k optimal value is the elbow of the curve where it is approximately straight and horizontal.

DBSCAN : Also used for partitional clustering.
* Automatically defines the number of clusters.
* It define objects forming a dense region as belonging to the same cluster

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

* Core point : have at least minPts number of data points within their eps distance (∈ dense regin)
* Border point : have less than minPts in their eps distance but in the neighborhood of a core point.
* Noise point : Not a core point and not close enough to be reachable from a core point.
* MinPts : minimum number of pts within the radius of a neighborhood, for that neighborhood to be considered a cluster
* Epsilon : Maximum distance between 2 data points to be considered as neighboring pts.

Agglomerative Hierarchical A bottom-up approach.
use normalized data and Euclidean distance

Algorithm: 1. Each object is considered a single element cluster
2. 2 similar clusters are combined to a new bigger one
   a/ Find closest pair of clusters $\min d(c_i, c_j)$
   b/ Merge c_i and c_i into c_{i+j}
   c/ Remove c_i, c_j from C and add c_{i+j}
3. Repeat until all points form one single big cluster

Linkage Criterion:
* single : measures the distance between the closest instances, one from each set. Favors the appearence of a dominant cluster.
* complete : " " between most distant instance, one from each set. Favors similar clusters
* Linkage: avg distance of every pair of instance, each instance from each set.
* Ward : measures the increase within-cluster variance after meging. favors compact clusters

Dendogram: A clustering tree. leaves are the objects and internal nodes a
the clusters

How to asses and evaluate?: from the dendogram.

Hyperparameters: Distance measure and linkage criterion.