Datascience

Course brief summary

Decision Tree

Decision tree

1. ID3
2. Cart

Advantages:

1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
2. A decision tree does not require normalization of data.
3. A decision tree does not require scaling of data as well.
4. Missing values in the data also does NOT affect the process of building decision tree to any considerable extent.
5. A Decision trees model is very intuitive and easy to explain to technical teams as well as stakeholders.

Disadvantage:

1. A small change in the data can cause a large change in the structure of the decision tree causing instability.
2. For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
3. Decision tree often involves higher time to train the model.
4. Decision tree training is relatively expensive as complexity and time taken is more.
5. Decision Tree algorithm is inadequate for applying regression and predicting continuous values.

Naive Bayes

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, c_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	yes	yes	yes	?

$$P(A|M) = \frac{6}{7} \times \frac{1}{7} \times \frac{2}{7} \times \frac{5}{7} = 0.025$$

$$P(A|N) = \frac{1}{13} \times \frac{3}{13} \times \frac{3}{13} \times \frac{9}{13} = 0.0028$$

$$P(A|M)P(M) = 0.025 \times \frac{7}{20} = 0.0088$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0018$$

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

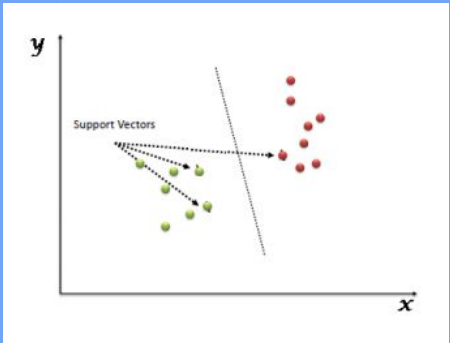
=> Mammals

Decision Trees

- Classification Problems require the prediction of a discrete target value
  - can be solved using decision tree learning
  - iteratively select the best attribute and split up the values according to this attribute
- Regression Problems require the prediction of a numerical target value
  - can be solved with regression trees and model trees
  - difference is in the models that are used at the leafs
  - are grown like decision trees, but with different splitting criteria
- Overfitting is a serious problem!
  - simpler, seemingly less accurate trees are often preferable
  - evaluation has to be done on separate test sets

SVM

SVM



The goal of a support vector machine is to find the optimal separating hyperplane which maximizes the margin of the training data Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges However, it is mostly used in classification problems

2 Linear Example – when Φ is trivial

Suppose we are given the following positively labeled data points in $\Re^2$:

$$\left\{\begin{pmatrix}3\\1\end{pmatrix},\begin{pmatrix}3\\-1\end{pmatrix},\begin{pmatrix}6\\1\end{pmatrix},\begin{pmatrix}6\\-1\end{pmatrix}\right\}$$

and the following negatively labeled data points in $\Re^2$ (see Figure 1):

$$\left\{\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}0\\-1\end{pmatrix},\begin{pmatrix}-1\\0\end{pmatrix}\right\}$$

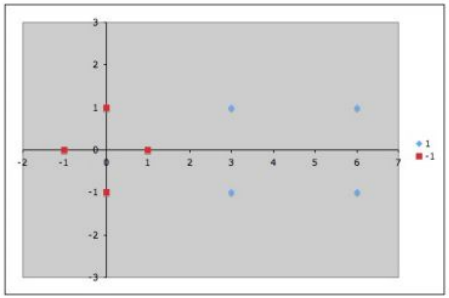


Figure 1: Sample data points in $\Re^2$. Blue diamonds are positive examples and red squares are negative examples.

We would like to discover a simple SVM that accurately discriminates the two classes. Since the data is linearly separable, we can use a linear SVM (that is, one whose mapping function $\Phi()$ is the identity function). By inspection, it should be obvious that there are three support vectors (see Figure 2):

$$\left\{s_1=\begin{pmatrix}1\\0\end{pmatrix},s_2=\begin{pmatrix}3\\1\end{pmatrix},s_3=\begin{pmatrix}3\\-1\end{pmatrix}\right\}$$

In what follows we will use vectors augmented with a 1 as a bias input, and for clarity we will differentiate these with an over-tilde. So, if $s_1 = (10)$, then $\tilde{s_1} = (101)$. Figure 3 shows the SVM architecture, and our task is to find values for the α_i such that

$$\begin{aligned}\alpha_1\Phi(s_1)\cdot\Phi(s_1)+\alpha_2\Phi(s_2)\cdot\Phi(s_1)+\alpha_3\Phi(s_3)\cdot\Phi(s_1) &= -1\\ \alpha_1\Phi(s_1)\cdot\Phi(s_2)+\alpha_2\Phi(s_2)\cdot\Phi(s_2)+\alpha_3\Phi(s_3)\cdot\Phi(s_2) &= +1\\ \alpha_1\Phi(s_1)\cdot\Phi(s_3)+\alpha_2\Phi(s_2)\cdot\Phi(s_3)+\alpha_3\Phi(s_3)\cdot\Phi(s_3) &= +1\end{aligned}$$

Since for now we have let $\Phi() = I$, this reduces to

$$\begin{aligned}\alpha_1\tilde{s_1}\cdot\tilde{s_1}+\alpha_2\tilde{s_2}\cdot\tilde{s_1}+\alpha_3\tilde{s_3}\cdot\tilde{s_1} &= -1\\ \alpha_1\tilde{s_1}\cdot\tilde{s_2}+\alpha_2\tilde{s_2}\cdot\tilde{s_2}+\alpha_3\tilde{s_3}\cdot\tilde{s_2} &= +1\\ \alpha_1\tilde{s_1}\cdot\tilde{s_3}+\alpha_2\tilde{s_2}\cdot\tilde{s_3}+\alpha_3\tilde{s_3}\cdot\tilde{s_3} &= +1\end{aligned}$$

Now, computing the dot products results in

$$\begin{aligned}2\alpha_1+4\alpha_2+4\alpha_3 &= -1\\ 4\alpha_1+11\alpha_2+9\alpha_3 &= +1\\ 4\alpha_1+9\alpha_2+11\alpha_3 &= +1\end{aligned}$$

A little algebra reveals that the solution to this system of equations is $\alpha_1 = -3.5, \alpha_2 = 0.75$ and $\alpha_3 = 0.75$.

Now, we can look at how these α values relate to the discriminating hyperplane; or, in other words, now that we have the α_i, how do we find the hyperplane that discriminates the positive from the negative examples? It turns out that

$$\begin{aligned}\tilde{w} &= \sum_i \alpha_i\tilde{s_i}\\ &= -3.5\begin{pmatrix}1\\0\\1\end{pmatrix}+0.75\begin{pmatrix}3\\1\\1\end{pmatrix}+0.75\begin{pmatrix}3\\-1\\1\end{pmatrix}\\ &= \begin{pmatrix}1\\0\\-2\end{pmatrix}\end{aligned}$$

Finally, remembering that our vectors are augmented with a bias, we can equate the last entry in $\tilde{w}$ as the hyperplane offset b and write the separating hyperplane equation $y = wx + b$ with $w = \begin{pmatrix}1\\0\end{pmatrix}$ and $b = -2$. Plotting the line gives the expected decision surface (see Figure 4).

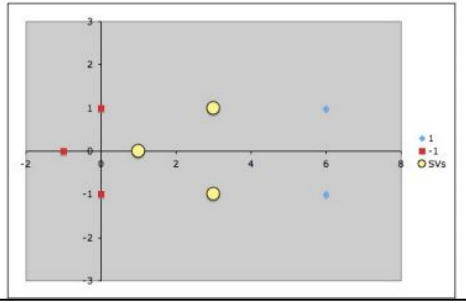


Figure 2: The three support vectors are marked as yellow circles.

SVM Advantages

- SVM's are very good when we have no idea on the data.
- Works well with even unstructured and semi structured data like text, Images and trees.
- The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
- Unlike in neural networks, SVM is not solved for local optima.
- It scales relatively well to high dimensional data.
- SVM models have generalization in practice, the risk of over-fitting is less in SVM.
- SVM is always compared with ANN. When compared to ANN models, SVMs give better results.

SVM Disadvantages

- Choosing a "good" kernel function is not easy.
- Long training time for large datasets.
- Difficult to understand and interpret the final model, variable weights and individual impact.
- Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic.
- The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact

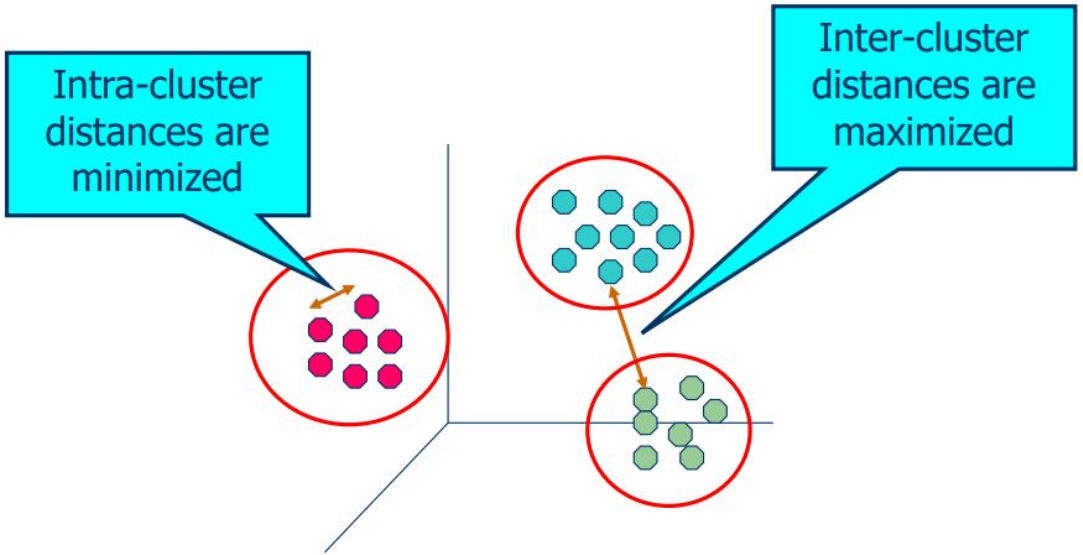
Dimensionality Reduction

PCA

Clustering

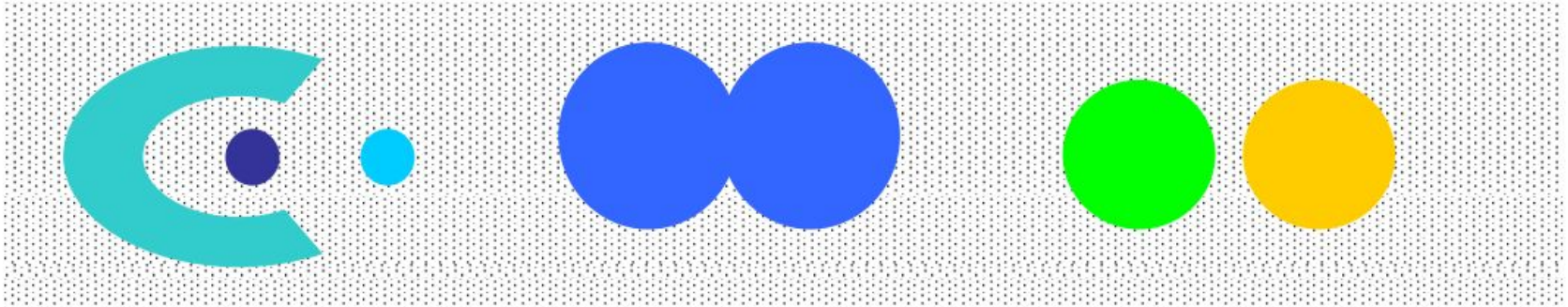
What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.



Other Distinctions Between Sets of Clusters

- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or 'border' points
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data
- Heterogeneous versus homogeneous
  - Cluster of widely different sizes, shapes, and densities

K-Means

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into *K*pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.

As a simple illustration of a k-means algorithm, consider the following data set consisting of the scores of two variables on each of seven individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This data set is to be grouped into two clusters. As a first step in finding a sensible initial partition, let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, giving:

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean. The mean vector is recalculated each time a new member is added. This leads to the following series of steps:

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Now the initial partition has changed, and the two clusters at this stage having the following characteristics:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

But we cannot yet be sure that each individual has been assigned to the right cluster. So, we compare each individual's distance to its own cluster mean and to that of the opposite cluster. And we find:

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1). In other words, each individual's distance to its own cluster mean should be smaller that the distance to the other cluster's mean (which is not the case with individual 3). Thus, individual 3 is relocated to Cluster 2 resulting in the new partition:

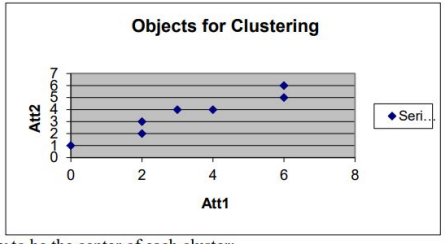
	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

The iterative relocation would now continue from this new partition until no more relocations occur. However, in this example each individual is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.

Also, it is possible that the k-means algorithm won't find a final solution. In this case it would be a good idea to consider stopping the algorithm after a pre-chosen maximum of iterations.

Question: Use k-means to cluster data from Table 1. Show all steps clearly. Use k = 2 and number of iterations = 2. Show the value of root mean square distance in each iteration

object for Clustering		
Instance	Att1	Att2
1	2	3
2	2	2
3	3	4
4	6	5
5	6	6
6	4	4
7	0	1



We want to select 2 points initially to be the center of each cluster:

initial choice of Centroids		
	initial	
	x	y
Centroid 1	2	2
Centroid 2	6	5

Calculate the distance between each points with 2 selected centroids, then can cluster by nearst points to the selected centroid:

object for Clustering (Argumanted)					
Instance	Att1	Att2	d1	d2	cluster
1	2	3	1	4.472136	1
2	2	2	0	5	1
3	3	4	2.236068	3.162278	1
4	6	5	5	0	2
5	6	6	5.656854	1	2
6	4	4	2.828427	2.236068	2
7	0	1	2.236068	7.211103	1

To calculate the centroids after first iteration by taking the average of the points that's clustered in same cluster :

Centroids after First Iteration				
	initial		after First Iteration	
	Att1	Att2	Att1	Att2
Centroid 1	2	2	1.75	2.5
Centroid 2	6	5	5.333333	5

- To calculate the first centroid RMSD:

$$\text{RMSD} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_{1,t} - x_{2,t})^2} =$$

$$\sqrt{\frac{1}{4}[(2-1.75)^2-(3-2.5)^2]+[(2-1.75)^2-(2-2.5)^2]+[(3-1.75)^2-(4-2.5)^2]+[(0-1.75)^2-(1-2.5)^2]}$$

$$=\sqrt{\frac{1}{4}[(0.25)^2-(0.5)^2]+[(0.25)^2-(-0.5)^2]+[(1.25)^2-(1.5)^2]+[(-1.75)^2-(-1.5)^2]}$$

$$=\sqrt{\frac{1}{4}[-0.25]} = -\sqrt{\frac{1}{4}[0.25]} = -0.25$$

- To calculate the second centroid RMSD:

$$\sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_{1,t} - x_{2,t})^2} = \sqrt{\frac{1}{3}[(6-5.33)^2-(5-5)^2]+[(6-5.33)^2-(6-5)^2]+[(4-5.33)^2-(4-5)^2]}$$

$$=\sqrt{\frac{1}{3}[0.4489]+[-0.5511]+[0.7689]} = \sqrt{\frac{1}{3}\,0.6667} = 0.4714163$$

Calculate the distance between each points with 2 new centroids, then can cluster by nearst points to the new centroid:

object for Clustering (Argumanted)					
Instance	Att1	Att2	d1	d2	cluster
1	2	3	0.559017	3.887301	1
2	2	2	0.559017	4.484541	1
3	3	4	1.952562	2.538591	1
4	6	5	4.930771	0.666667	2
5	6	6	5.505679	1.20185	2
6	4	4	2.704163	1.666667	2
7	0	1	2.304886	6.666667	1

Calculate the centroids after second iteration by taking the average of the points that's clustered in same cluster :

Centroids after Second Iteration						
	initial		after First Iteration		after Second Iteration	
	Att1	Att2	Att1	Att2	Att1	Att2
Centroid 1	2	2	1.75	2.5	1.75	2.5
Centroid 2	6	5	5.333333	5	5.333333	5

We notice that's the centroids no longer move because it's met it's cluster center that what we want.

- To calculate the first centroid RMSD:

$$\text{RMSD} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(x_{1,t} - x_{2,t}\right)^2} =$$

$$\sqrt{\frac{1}{4}\left[(2-1.75)^2 - (3-2.5)^2\right] + \left[(2-1.75)^2 - (2-2.5)^2\right] + \left[(3-1.75)^2 - (4-2.5)^2\right] + \left[(0-1.75)^2 - (1-2.5)^2\right]}$$

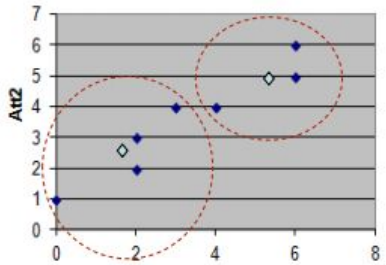
$$= \sqrt{\frac{1}{4}\left[(0.25)^2 - (0.5)^2\right] + \left[(0.25)^2 - (-0.5)^2\right] + \left[(1.25)^2 - (1.5)^2\right] + \left[(-1.75)^2 - (-1.5)^2\right]}$$

$$= \sqrt{\frac{1}{4}\left[-0.25\right]} = -\sqrt{\frac{1}{4}\left[0.25\right]} = -0.25$$

- To calculate the second centroid RMSD:

$$\sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(x_{1,t} - x_{2,t}\right)^2} = \sqrt{\frac{1}{3}\left[(6-5.33)^2 - (5-5)^2\right] + \left[(6-5.33)^2 - (6-5)^2\right] + \left[(4-5.33)^2 - (4-5)^2\right]}$$

$$= \sqrt{\frac{1}{3}\left[0.4489\right] + \left[-0.5511\right] + \left[0.7689\right]} = \sqrt{\frac{1}{3}\ 0.6667} = 0.4714163$$



K-Means Advantages :

1) If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.

2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

K-Means Disadvantages :

1) Difficult to predict K-Value.
2) With global cluster, it didn't work well.
3) Different initial partitions can result in different final clusters.
4) It does not work well with clusters (in the original data) of Different size and Different density

- **Advantages of k-means algorithm**:

1. Ease of implementation and high-speed performance
2. Measurable and efficient in large data collection

- **Disadvantages of k-means algorithm**:

1. Selection of optimal number of clusters is difficult
2. Selection of the initial centroids is random.

Hierarchical Clustering

complete-link clustering suffers from a different problem. It pays too much attention to outliers

In complete-link (or complete linkage) hierarchical clustering, we merge in each step the two clusters whose merger has the smallest diameter (or: the two clusters with the smallest **maximum** pairwise distance).

In single-link (or single linkage) hierarchical clustering, we merge in each step the two clusters whose two closest members have the smallest distance (or: the two clusters with the smallest **minimum** pairwise distance).

Question: Table 2 shows a example of a Distance Matrix. Draw Dendrogram using Hierarchical Clustering Process. Use single link clustering. Show all steps.

	a	b	c	d	E	F
a	0	13	2	7	1	3
b	13	0	6	9	10	2
c	2	6	0	14	3	4
d	7	9	14	0	4	5
e	1	10	3	4	0	16
f	3	2	4	5	16	0

We want to combine (ae) because they are closest pair with minimum distance = 1

Distance Matrix after first Merge

	ae	b	c	d	F
ae	0	10	2	4	3
b	10	0	6	9	2
c	2	6	0	14	4
d	4	9	14	0	5
f	3	2	4	5	0

We want to combine (aec) because they are closest pair with minimum distance = 2

Distance Matrix after Second Merge

	aec	b	d	f
aec	0	6	4	3
b	6	0	9	**2**
d	4	9	0	5
f	3	2	5	0

We want to combine (bf) because they are closest pair with minimum distance = 2

Distance Matrix after Third Merge

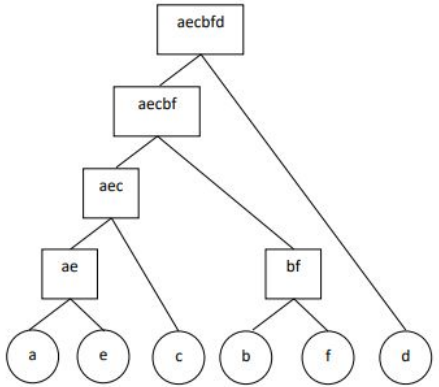
	aec	bf	d
aec	0	3	4
bf	3	0	5
d	4	5	0

We want to combine (aecbf) because they are closest pair with minimum distance = 3

Distance Matrix after Fourth Merge

	aecbf	d
aecbf	0	4
d	4	0

The final stage clusters aecbf and d are merged into a single cluster aecbfd which contains all the original six objects.
The dendrogram corresponding to this clustering process is:



Dendrogram corresponding to Hierarchical Clustering Process

Complete Link Clustering

	a	b	c	d	E	F
a	0	13	2	7	1	3
b	13	0	6	9	10	2
c	2	6	0	14	3	4
d	7	9	14	0	4	5
e	1	10	3	4	0	16
f	3	2	4	5	16	0

We want to combine (ae) because they are closest pair with minimum distance = 1

Distance Matrix after first Merge

	Ae	B	c	d	F
ae	0	13	3	7	16
b	13	0	6	9	2
c	3	6	0	14	4
d	7	9	14	0	5
f	16	2	4	5	0

Distance Matrix after 2nd Merge

	ae	bf	c	d	
ae	0	16	3	7	
bf	16	0	6	9	
c	3	6	0	14	
d	7	9	14	0	

Distance Matrix after 3rd Merge

	ace	bf		D	
ace	0	16		14	
bf	16	0		9	
d	14	9		0	

Distance Matrix after 4th merge

	Ace	bf			
ace	0	16			
bfd	16	0			

DBScan

Association Rule mining

Association rule mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Definition: Association Rule

- Association Rule
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example: {Milk, Diaper} → {Beer}
- Rule Evaluation Metrics
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association rule mining

Given a set of transactions T, the goal of association rule mining is to find all rules having – support ≥ minsup threshold – confidence ≥ minconf threshold

Mining Association Rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset: {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive

Frequent itemset methods

1. Apriori
    a. Candidate generation
2. FP-Growth
    a. FP-Tree generation

FP-Tree is better because in Apriori we have to scan all the database which gets heavy sometimes

Apriori

The Apriori Algorithm: Basics

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

Key Concepts :

- Frequent Itemsets: The sets of item which has minimum support (denoted by L_i for i^{th}-Itemset).
- Apriori Property: Any subset of frequent itemset must be frequent.
- Join Operation: To find L_k , a set of candidate k-itemsets is generated by joining L_{k-1} with itself.

The Apriori Algorithm in a Nutshell

- Find the *frequent itemsets*: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if {*AB*} is a frequent itemset, both {*A*} and {*B*} should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to *k (k*-itemset*)*
- Use the frequent itemsets to generate association rules.

Consider the following dataset and we will find frequent itemsets and generate association rules for them.

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

minimum support count is 2
minimum confidence is 60%

Step-1: K=1

(I) Create a table containing support count of each item present in dataset – Called **C1(candidate set)**

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

(II) compare candidate set item's support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items). This gives us itemset L1.

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Step-2: K=2

- Generate candidate set C2 using L1 (this is called join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.
- Check all subsets of an itemset are frequent or not and if not frequent remove that itemset. (Example subset of{I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)
- Now find support count of these itemsets by searching in dataset.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

(II) compare candidate (C2) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

Step-3:

- Generate candidate set C3 using L2 (join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common. So here, for L2, first element should match.
  So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}
- Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)
- find support count of these remaining itemset by searching in dataset.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

(II) Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Step-4:

- Generate candidate set C4 using L3 (join step). Condition of joining L_{k-1} and L_{k-1} (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.
- Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent). So no itemset in C4
- We stop here because no frequent itemsets are found further

---

Thus, we have discovered all the frequent item-sets. Now generation of strong association rule comes into picture. For that we need to calculate confidence of each rule.

Confidence –

A confidence of 60% means that 60% of the customers, who purchased milk and bread also bought butter.

$$Confidence(A->B)=Support\_count(A\cup B)/Support\_count(A)$$

So here, by taking an example of any frequent itemset, we will show the rule generation.

Itemset {I1, I2, I3} //from L3

SO rules can be

[I1^I2]=>[I3] //confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%

[I1^I3]=>[I2] //confidence = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%

[I2^I3]=>[I1] //confidence = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%

[I1]=>[I2^I3] //confidence = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%

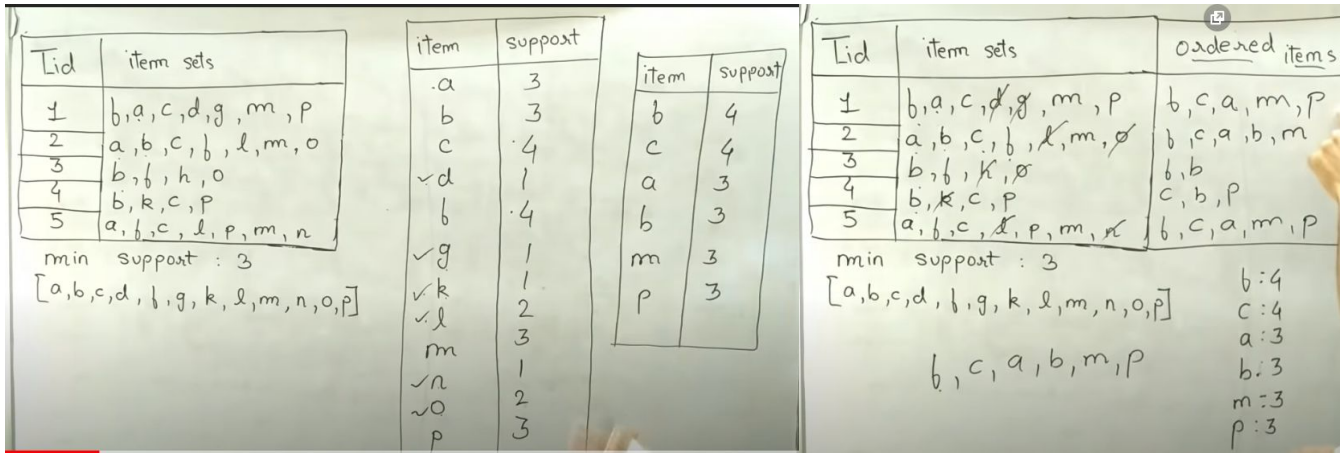
[I2]=>[I1^I3] //confidence = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%

[I3]=>[I1^I2] //confidence = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%

So if minimum confidence is 50%, then first 3 rules can be considered as strong association rules.

Limitations of Apriori Algorithm: Apriori Algorithm can be slow. The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets. For example, if there are 10^4 from frequent 1- itemsets, it need to generate more than 10^7 candidates into 2-length which in turn they will be tested and accumulate. Furthermore, to detect frequent pattern in size 100 i.e. v1, v2… v100, it have to generate 2^100 candidate itemsets that yield on costly and wasting of time of candidate generation. So, it will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions.

FP-Growth



Tid	item sets
1	b, a, c, d, g, m, p
2	a, b, c, f, l, m, o
3	b, f, h, o
4	b, k, c, p
5	a, f, c, l, p, m, n

min support : 3

[a, b, c, d, f, g, k, l, m, n, o, p]

item	support
a	3
b	3
c	4
d	1
f	4
g	1
k	1
l	2
m	3
n	1
o	2
p	3

item	support
b	4
c	4
a	3
b	3
m	3
p	3

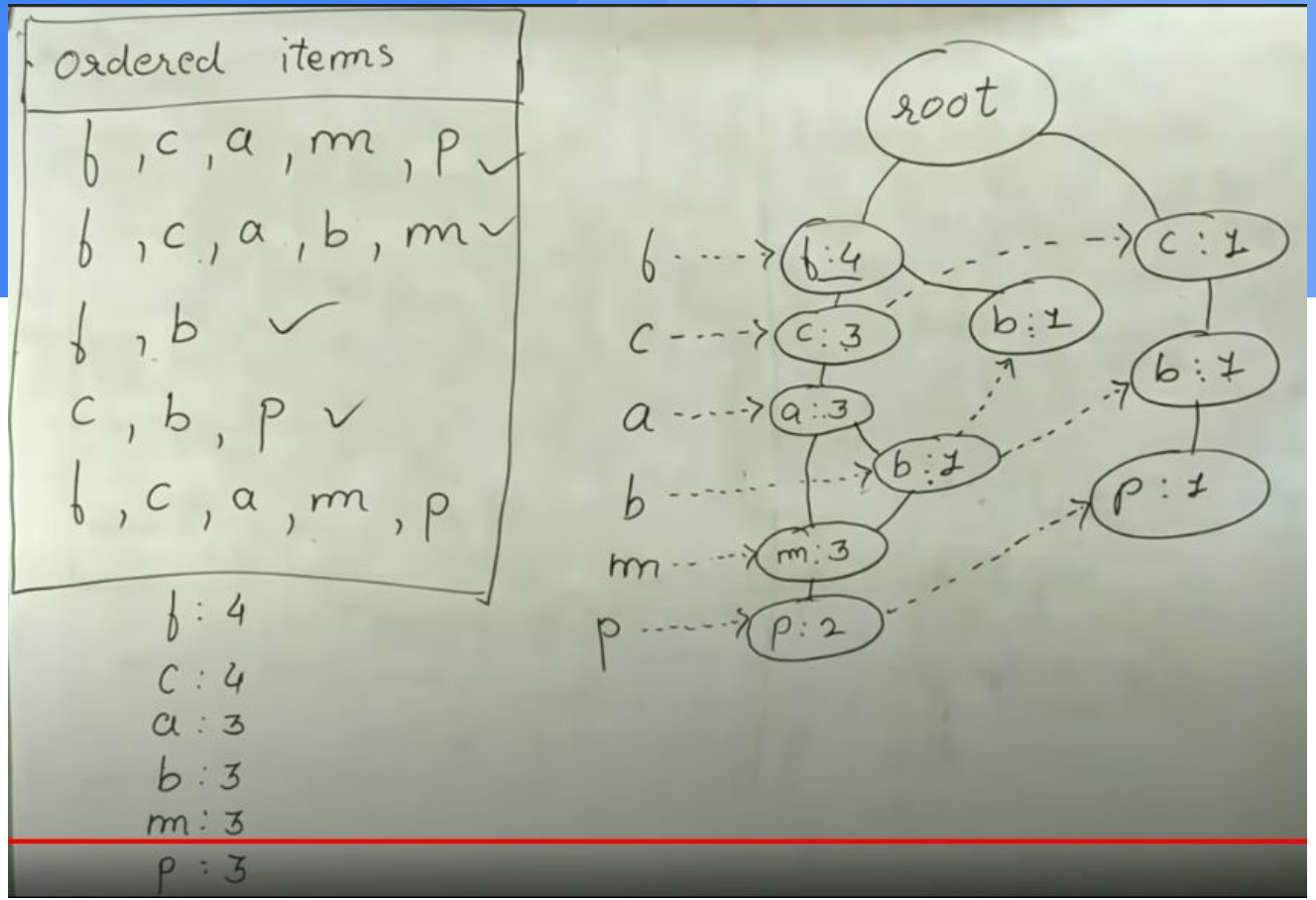
Tid	item sets	ordered items
1	b, a, c, ~~d~~, ~~g~~, m, p	b, c, a, m, p
2	a, b, c, f, ~~l~~, m, ~~o~~	b, c, a, b, m
3	b, f, ~~k~~, ~~o~~	b, b
4	b, ~~k~~, c, p	c, b, p
5	a, f, c, ~~l~~, p, m, ~~n~~	b, c, a, m, p

min support : 3

[a, b, c, d, f, g, k, l, m, n, o, p]

b, c, a, b, m, p

b : 4
c : 4
a : 3
b : 3
m : 3
p : 3


ordered items
f , c , a , m , p
f , c , a , b , m
f , b
c , b , p
f , c , a , m , p
f : 4
c : 4
a : 3
b : 3
m : 3
p : 3
root
f : 4
c : 1
c : 3
b : 1
a : 3
b : 1
b : 1
m : 3
p : 1
p : 2

FP Growth vs Apriori

FP Growth	Apriori
Pattern Generation	
FP growth generates pattern by constructing a FP tree	Apriori generates pattern by pairing the items into singletons, pairs and triplets.
Candidate Generation	
There is no candidate generation	Apriori uses candidate generation
Process	
The process is faster as compared to Apriori. The runtime of process increases linearly with increase in number of itemsets.	The process is comparatively slower than FP Growth, the runtime increases exponentially with increase in number of itemsets
Memory Usage	
A compact version of database is saved	The candidates combinations are saved in memory

Ensemble Learning

Ensemble learning

Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. **Ensemble learning** is primarily used to improve the (classification, prediction, function approximation, etc.)

Classification Fusion Techniques

- Homogenous Classifiers (Same Classifiers but different training data) e.g. Bagging, Boosting etc
- Heterogeneous Classifiers (Different Classifiers but same training data) e.g. Majority Voting, Mean etc)

Type I (abstract level):

- This is the lowest level since a classifier provides the least amount of information
- on this level, Classifier output is merely a single class label or an unordered set of candidate classes

Type II (rank level):

- Classifier output on the rank level is an ordered sequence of candidate classes, the so-called n-best list

Type III (measurement level)

- In addition to the ordered n-best lists of candidate classes on the rank level, classifier output on the measurement level has confidence values assigned to each entry of the n-best list

Voting in Ensemble Learning

- Hard Voting
  - Let Assumes that Three classifiers predicts as follow:
    - Classifier 1 predicts class A
    - Classifier 2 predicts class B
    - Classifier 3 predicts class B
  - 2/3 classifiers predict class B, so class B is the ensemble decision.
- Soft Voting
  - Let Assumes that Three classifiers predicts as follow:
    - Classifier 1 predicts class A with Prob 93%
    - Classifier 2 predicts class A with Prob 44%
    - Classifier 3 predicts class A with Prob 40%
  - On average the ensemble produces (93+44+40)/3 = 59% probability predict class A, so class A is the ensemble decision.

Others Heterogeneous Classifiers

- Weighted Majority Vote
- Naïve Bayes Combination
- Fuzzy Integral
- Decision Template
- Dempster-Shafer Combination
- Many More

Homogenous Ensemble Classifiers

Same classifier but different training data

- Bagging
- Boosting
- Random Forest
- Others

Bagging

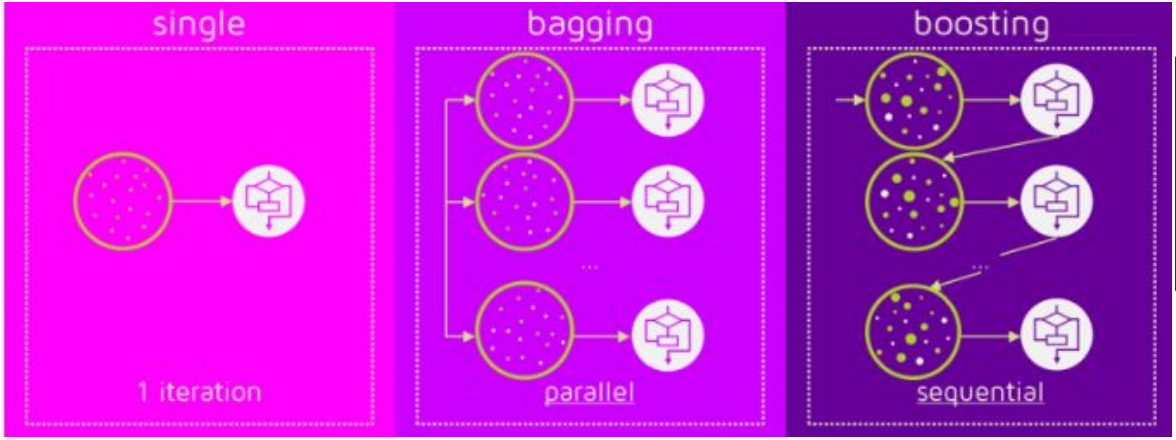
Sample several training sets of size n (instead of just having one training set of size m where m>>n) – Build a classifier for each training set – Combine the classifier's predictions.This improves performance in almost all cases if learning scheme is unstable (i.e. decision trees)

Bagging is used when the goal is to reduce the variance of a decision tree classifier. Here the **objective** is to create several subsets of data from training sample chosen randomly with replacement. Each collection of subset data is used to train their decision trees.

Bootstrap aggregating, also called bagging (from bootstrap aggregating), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting.

Boosting

The term '**Boosting**' refers to a family of algorithms which converts weak learner to strong learners. **Boosting** is an ensemble method for improving the model predictions of any given learning algorithm. The idea of **boosting** is to train weak learners sequentially, each trying to correct its predecessor.

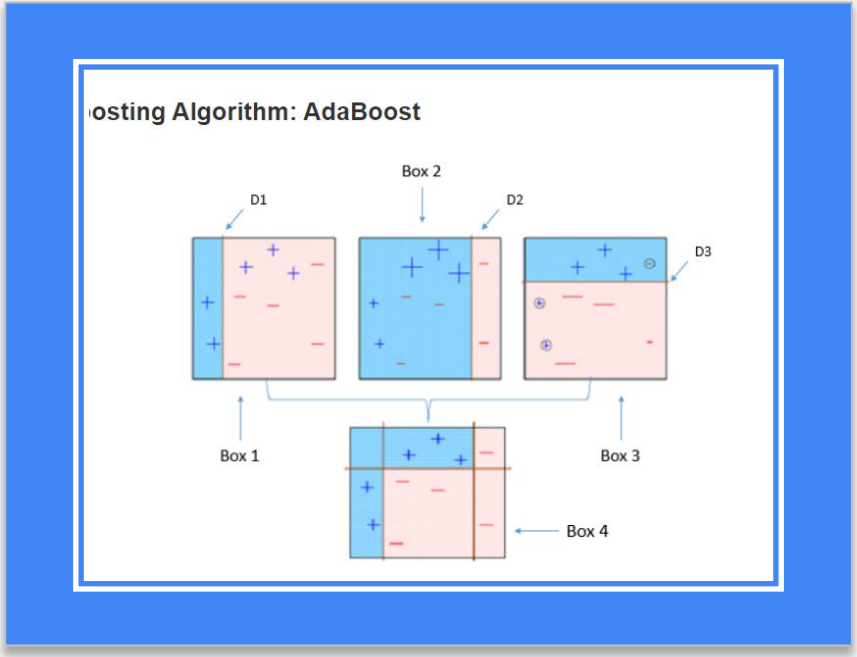


Types of boosting

1. Adaboost (Adaptive boosting)
2. Gradient boosting
3. XGBoost

Adaboost

Combines three weak learners to create the final output Box4



Gradient boosting

Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (eg. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function (*y=ax+b+e , e needs a special mention as it is the error term)*. The loss function is a measure indicating how good are model's coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimise. For example, if we are trying to predict the sales prices by using a regression, then the loss function would be based off the error between true and predicted house prices. Similarly, if our goal is to classify credit defaults, then the loss function would be a measure of how good our predictive model is at classifying bad loans. One of the biggest motivations of using gradient boosting is that it allows one to optimise a user specified cost function, instead of a loss function that usually offers less control and does not essentially correspond with real world applications.

XGBoost

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data.

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

Stacking

Stacking is an ensemble machine learning algorithm that learns how to best combine the predictions from multiple well-performing machine learning models.

Stacking or Stacked Generalization is an ensemble machine learning algorithm.

It uses a meta-learning algorithm to learn how to best combine the predictions from two or more base machine learning algorithms.

The benefit of stacking is that it can harness the capabilities of a range of well-performing models on a classification or regression task and make predictions that have better performance than any single model in the ensemble.

Some practical advices

- If the classifier is unstable (high variance), then apply bagging!
- If the classifier is stable and simple (high bias) then apply boosting!
- If the classifier is stable and complex then apply randomization injection!
- If you have many classes and a binary classifier then try error-correcting codes! If it does not work then use a complex binary classifier!

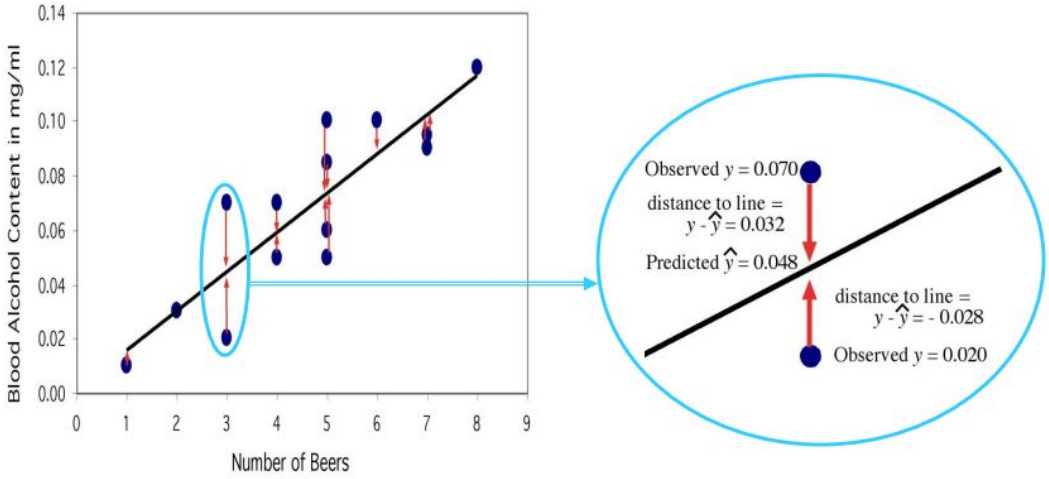
Relationships: Regression

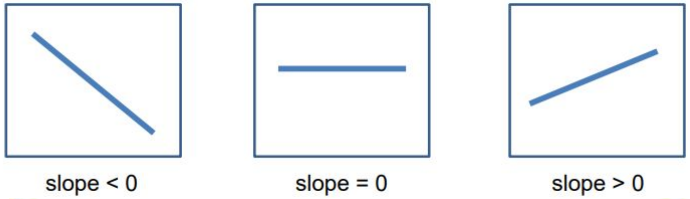
Relationships: Regression

The **least-squares regression line** is the unique line such that the sum of the **vertical distances** between the data points and the line is zero, and the sum of the squared vertical distances is the smallest possible.

$\hat{y}$ is the predicted y value on the regression line

$$\hat{y} = \text{intercept} + \text{slope}\ x \qquad \hat{y} = a + bx$$





The **slope** of the regression line describes how much we expect y to change, on average, for every unit change in x.

Relationships: Regression

The **slope of the regression line, *b***, equals: $b = r\frac{s_y}{s_x}$

r is the correlation coefficient between x and y
s_y is the standard deviation of the response variable y
s_x is the standard deviation of the explanatory variable x

The **intercept, *a***, equals: $a = \bar{y} - b\bar{x}$

$\bar{x}$ and $\bar{y}$ are the respective means of the x and y variables

r^2 represents **the fraction of the variance in *y* that can be explained by the regression model.**

The vertical distances from each point to the least-squares regression line are called **residuals**. The sum of all the residuals is by definition 0.

The Line

Our aim is to calculate the values **m** (slope) and **b** (y-intercept) in the equation of a line :

$$y = mx + b$$

Where:

- **y** = how far up
- **x** = how far along
- **m** = Slope or Gradient (how steep the line is)
- **b** = the Y Intercept (where the line crosses the Y axis)

To find the line of best fit for **N** points:

Step 1: For each (x,y) point calculate x^2 and xy

Step 2: Sum all x, y, x^2 and xy, which gives us Σx, Σy, Σx^2 and Σxy (Σ means "sum up")

Step 3: Calculate Slope **m**:

$$\mathbf{m} = \frac{N \Sigma(xy) - \Sigma x \, \Sigma y}{N \Sigma(x^2) - (\Sigma x)^2}$$

(N is the number of points.)

Step 4: Calculate Intercept **b**:

$$\mathbf{b} = \frac{\Sigma y - m \, \Sigma x}{N}$$

Step 5: Assemble the equation of a line

$$y = mx + b$$

Done!

Example: Sam found how many **hours of sunshine** vs how many **ice creams** were sold at the shop from Monday to Friday:

"x" Hours of Sunshine	"y" Ice Creams Sold
2	4
3	5
5	7
7	10
9	15

Let us find the best **m** (slope) and **b** (y-intercept) that suits that data

$$y = mx + b$$

Step 2: Sum x, y, x^2 and xy (gives us Σx, Σy, Σx^2 and Σxy):

x	y	x^2	xy
2	4	4	8
3	5	9	15
5	7	25	35
7	10	49	70
9	15	81	135
Σx: 26	**Σy: 41**	**Σx^2: 168**	**Σxy: 263**

Also **N** (number of data values) = **5**

Step 3: Calculate Slope **m**:

$$m = \frac{N\,\Sigma(xy) - \Sigma x\,\Sigma y}{N\,\Sigma(x^2) - (\Sigma x)^2}$$

$$= \frac{5 \times 263 - 26 \times 41}{5 \times 168 - 26^2}$$

$$= \frac{1315 - 1066}{840 - 676}$$

$$= \frac{249}{164} = 1.5183...$$

Step 4: Calculate Intercept **b**:

$$b = \frac{\Sigma y - m\,\Sigma x}{N}$$

$$= \frac{41 - 1.5183 \times 26}{5}$$

$$= 0.3049...$$

Step 5: Assemble the equation of a line:

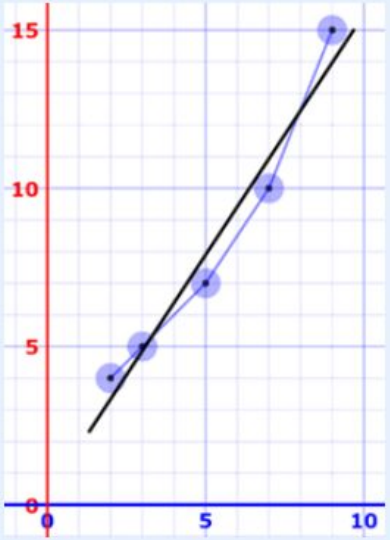
$$y = mx + b$$

$$y = 1.518x + 0.305$$

Let's see how it works out:

x	y	y = 1.518x + 0.305	error
2	4	3.34	−0.66
3	5	4.86	−0.14
5	7	7.89	0.89
7	10	10.93	0.93
9	15	13.97	−1.03

Here are the (x,y) points and the line **y = 1.518x + 0.305** on a graph:



Nice fit!

Sam hears the weather forecast which says "we expect 8 hours of sun tomorrow", so he uses the above equation to estimate that he will sell

$$y = 1.518 \times 8 + 0.305 = 12.45 \text{ Ice Creams}$$

Sam makes fresh waffle cone mixture for 14 ice creams just in case. Yum.

Lurking vs confounded variable

A lurking variable is a variable that is unknown and not controlled for; It has an important, significant effect on the variables of interest.

A **lurking variable** is a variable that connects two things that wouldn't be connected otherwise.

The classic example is # firefighters at a scene vs damage done by a fire. Logically, from those variables you can assume that the higher amount of firefighters at a scene is the reason there is more damage. However, the lurking variable here is the intensity of the fire.

Firefighters would not cause damage if there was no fire.

A **confounding variable** is something that modifies an already existent relationship. Lets say, weight gain vs food intake. Yes, more food intake will directly increase weight gain, but a confounding variable would be exercise as it can play an effect the outcome of the two other variables.

Correlations

Correlations

1

Pearson's Correlation

2

Spearman's Correlation

3

Kendall's Tau

Pearson Correlation

SUBJECT	AGE X	GLUCOSE LEVEL Y	XY	X^2	Y^2
1	43	99	4257	1849	9801
2	21	65	1365	441	4225
3	25	79	1975	625	6241
4	42	75	3150	1764	5625
5	57	87	4959	3249	7569
6	59	81	4779	3481	6561
Σ	247	486	20485	11409	40022

Step 6: *Use the following correlation coefficient formula.*

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

The answer is: **2868 / 5413.27 = 0.529809**

Spearman's correlation

An example of calculating Spearman's correlation

To calculate a Spearman rank-order correlation on data without any ties we will use the following data:

	Marks									
English	56	75	45	71	62	64	58	80	76	61
Maths	66	70	40	60	65	56	59	77	67	63

We then complete the following table:

English (mark)	Maths (mark)	Rank (English)	Rank (maths)	d	d^2
56	66	9	4	5	25
75	70	3	2	1	1
45	40	10	10	0	0
71	60	4	7	3	9
62	65	6	5	1	1
64	56	5	9	4	16
58	59	8	8	0	0
80	77	1	1	0	0
76	67	2	3	1	1
61	63	7	6	1	1

Where d = difference between ranks and d^2 = difference squared.

We then calculate the following:

$$\sum d_i^2 = 25 + 1 + 9 + 1 + 16 + 1 + 1 = 54$$

We then substitute this into the main equation with the other information as follows:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

$$\rho = 1 - \frac{6 \times 54}{10(10^2 - 1)}$$

$$\rho = 1 - \frac{324}{990}$$

$$\rho = 1 - 0.33$$

$$\rho = 0.67$$

as $n = 10$. Hence, we have a ρ (or r_s) of 0.67. This indicates a strong positive relationship between the ranks individuals obtained in the maths and English exam. That is, the higher you ranked in maths, the higher you ranked in English also, and vice versa.

Kendall's Tau

Step 4: Sum the values in the two columns:

Candidate	Interviewer 1	Interviewer 2	Concordant	Discordant
A	1	1	11	0
B	2	2	10	0
C	3	4	8	1
D	4	3	8	0
E	5	6	6	1
F	6	5	6	0
G	7	8	4	1
H	8	7	4	0
I	9	10	2	1
J	10	9	2	0
K	11	12	0	1
L	12	11		
		Totals	61	5

Step 5: Insert the totals into the formula:

Kendall's Tau = (C – D / C + D)

= (61 – 5) / (61 + 5) = 56 / 66 = .85.

The Tau coefficient is .85, suggesting a strong relationship between the rankings.