Document Clustering

(Natural Language Processing)

HND Thilini

hnd@ucsc.cmb.ac.lk

Why document clustering?

- What happens when we want to categorize texts but have no data to learn from (i.e. no labelled/annotated, training data)
  - We relied on that when we used supervised learning for this task
- Can’t we collect and annotate a dataset to learn from?
  - Easier said than done – an arduous task since we need lots of data to train on!
  - How much data? http://machinelearningmastery.com/much--training--data–required--machine--learning
- Clustering documets without ‘supervision’ of labels/annotations
  - Unsupervised learning algorithms

How can we do document clustering?

- Can we measure similarity between documents?
- Can we use distance measures to find the most similar documents?
- Can we use such measures to group similar documents?
- Can we visualize such groups to determine their veracity?

Key concepts

- Information retrieval
  - Process of fetching the most relevant documents based on a query string
  - This is what search engines do: relevancy is key
- Feature engineering
  - Extracting abstractions or representations of documents from text content
- Similarity measures
  - Ways to define and measure ‘distance’ between two words, phrases, sentences or documents
- Unsupervised machine learning
  - Primarily clustering algorithms and dimensionality reduction algorithms

Pipeline

- We use our normal preprocessing pipeline as before
- Normalizing
  - We add some words to our list of stopwords by looking at our domain
  - We also are only going to use text tokens using a regular expression – why?
- Feature extraction
  - We reuse our previous feature extractors but add the functionality provided by sklearn’s Vectorizer classes
  - We include parameters for extracting features for n–grams instead of just unigrams (default) and setting max and min counts to be considered

Text similarity

- Way to measure how similar two words, phrases, sentences etc are
- Two broad ways to define ‘similarity’
  - Lexical level: the form of the relevant word/phrase/sentence
  - Semantic level: the meaning and context of the word/phrase/sentence
- Two main types of distance metrics used
  - Term similarity: measures distance between two words/tokens
  - Document similarity: measures the distance between entire documents
- We will be using multiple distance metrics and comparing results

Term similarity

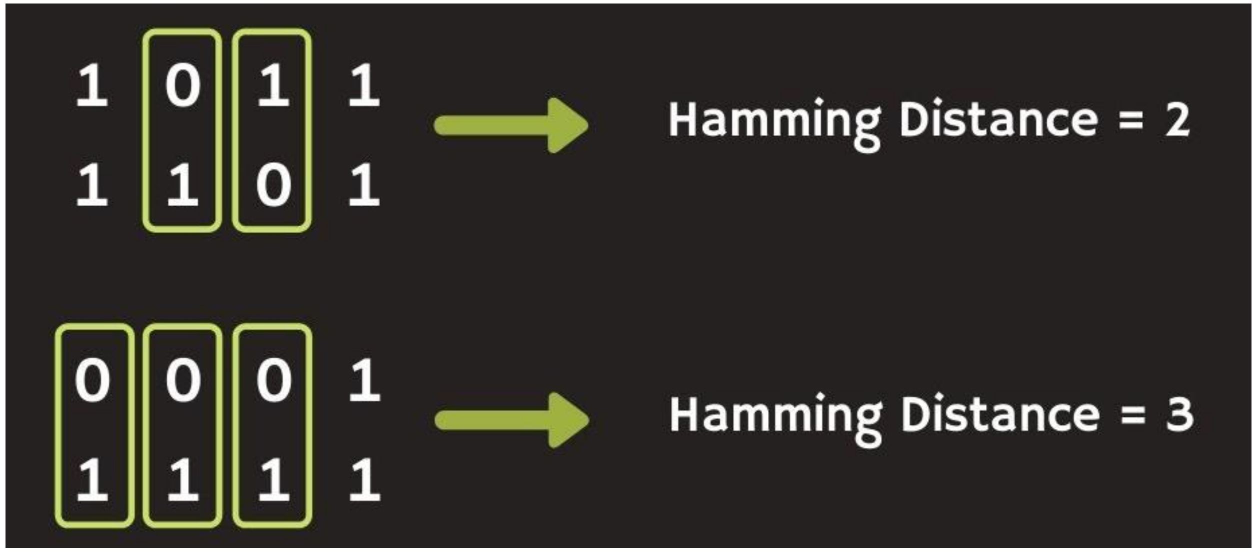
- Simple definition: how many characters match in sequence and in bag
    - i.e. a character vector and a bag of characters respectively
- We will use the most commonly used distance metrics to measure similarity
    - Hamming distance
    - Manhattan distance
    - Euclidean distance
    - Levenshtein (edit) distance
    - Cosine distance/similarity

Hamming distance

- Number of positions that have different characters or symbols between *two strings of equal length*
  - Usually normalized for the length of the string

$$hd(u,v) = \sum_{i=1}^{n} (u_i \neq v_i)$$

$$norm\_hd(u,v) = \frac{\sum_{i=1}^{n} (u_i \neq v_i)}{n}$$



Manhattan distance

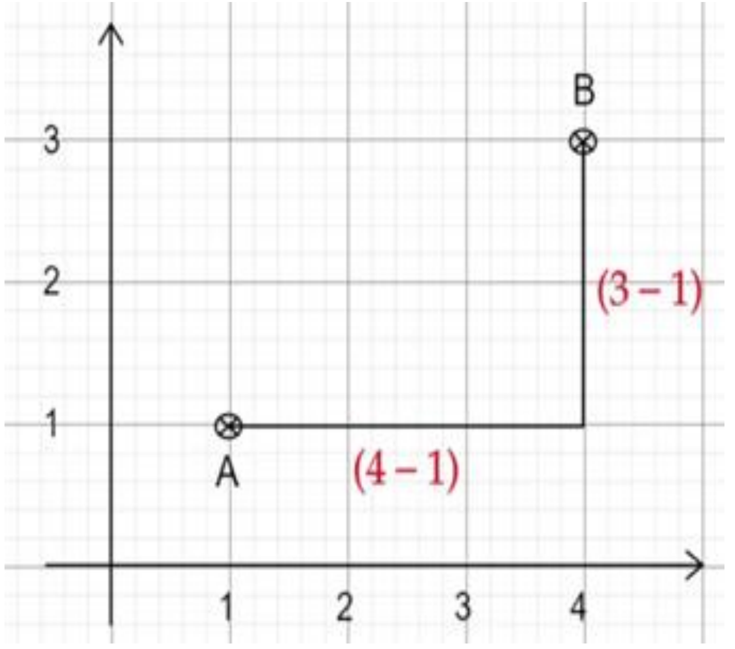
- Distance between two points in a grid based on strictly horizontal or vertical paths
  - Not using the diagonal distance usually calculated by Euclidean distance
  - Also called city block distance, taxicab metric or L1 norm
- For strings, we subtract the difference between each pair of characters at each position of the two strings
  - So requires the strings to be of equal length
  - Can be normalized for length as before

$$md(u,v) = \|u-v\|_1 = \sum_{i=1}^{n} |u_i - v_i|$$

$$norm\_md(u,v) = \frac{\|u-v\|_1}{n} = \frac{\sum_{i=1}^{n} |u_i - v_i|}{n}$$

Manhattan Disntace

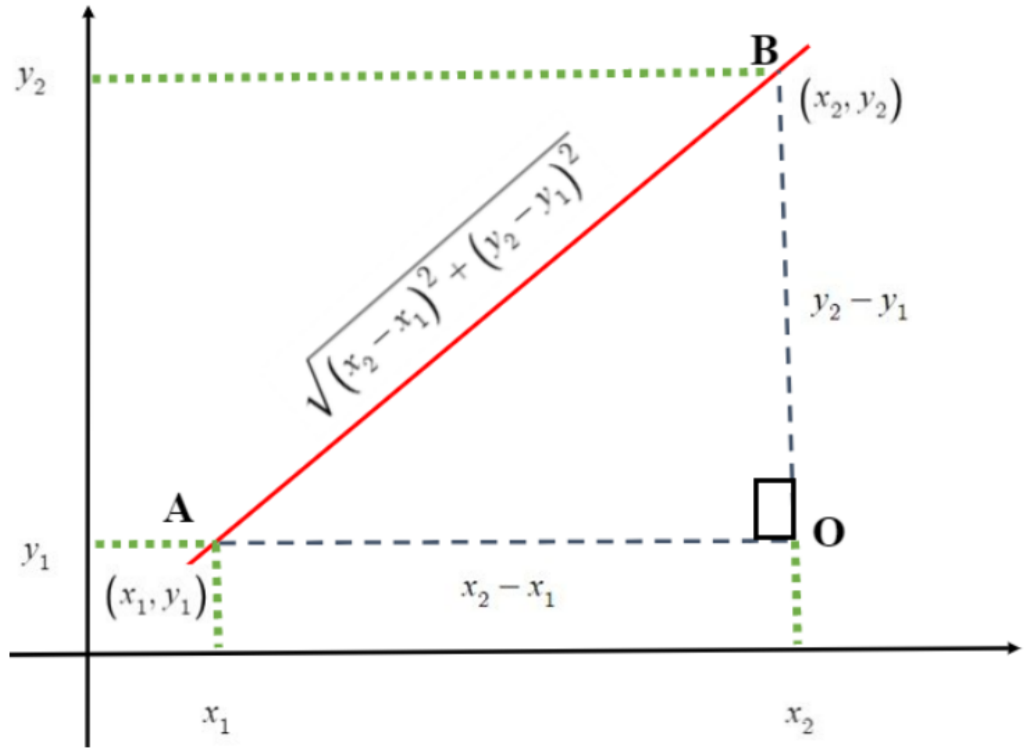
$|4-1| + |3-1| = 5$



Euclidean distance

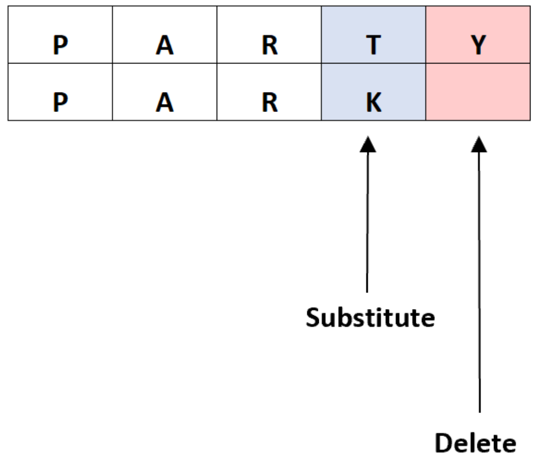
- The shortest straight--line distance between two points
  - ‘As the crow flies’
  - Also called Euclidean norm, L2 norm or L2 distance

$$ed(u,v)=\|u-v\|_2=\sqrt{\sum_{i=1}^{n}(u_i-v_i)^2}$$



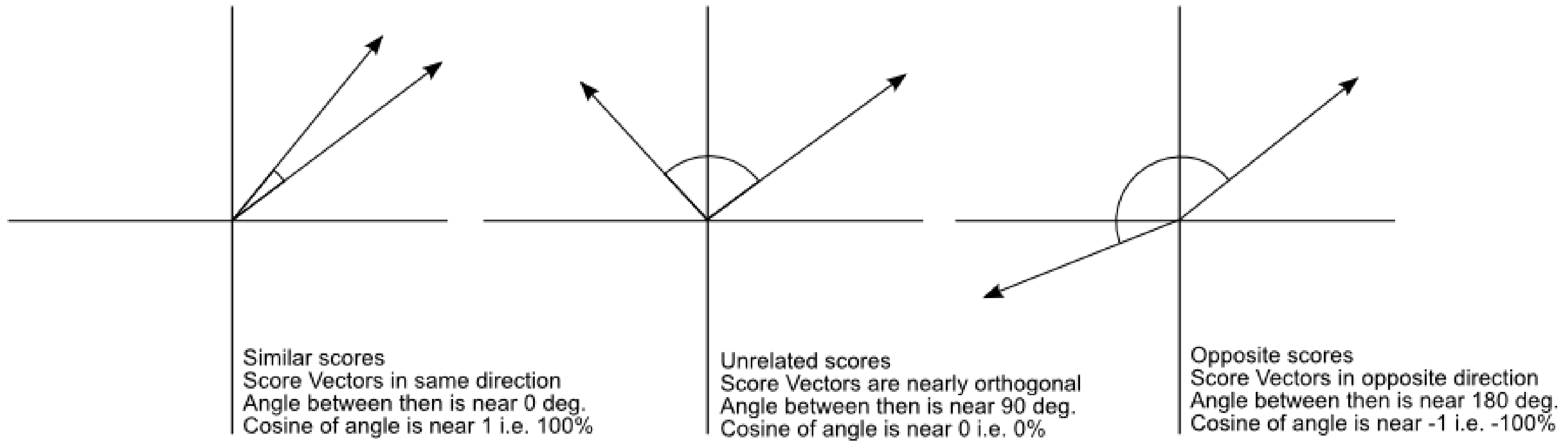
Dealing with the 'length problem'

- Very unrealistic in general to assume this!
- Solution: 'edit' distance based measures
- Levenshtein distance – the most popular
  - The minimum number of edits needed (in the form of additions, deletions, or substitutions) to change/convert one term to the other
  - Importantly, the length of the two terms need not be the same
  - Minimum = difference in length between 2 terms
  - Maximum = length of the longer term
  - If equal, distance = 0
  - Satisfies 'triangle inequality'



Cosine distance

- A measure of the cosine of the angle between two terms when represented as non--zero positive vectors in an inner product space
  - The cosine similarity (cs) value will in general lie between –1 and +1
  - For bag--of–words (or characters) based models it will be between 0 and 1 since the frequencies can never be negative
  - Cosine distance is the inverse: i.e. 1 – cs


Similar scores
Score Vectors in same direction
Angle between then is near 0 deg.
Cosine of angle is near 1 i.e. 100%
Unrelated scores
Score Vectors are nearly orthogonal
Angle between then is near 90 deg.
Cosine of angle is near 0 i.e. 0%
Opposite scores
Score Vectors in opposite direction
Angle between then is near 180 deg.
Cosine of angle is near -1 i.e. -100%

Example of cosine similarity

- If d_1 and d_2 are two document vectors, then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / ||d_1||\ ||d_2||,$$

where $\bullet$ indicates vector dot product and $|| d ||$ is the length of vector d.

- Example:

$d_1 =$ **3 2 0 5 0 0 0 2 0 0**
$d_2 =$ **1 0 0 0 0 0 0 1 0 2**

$d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$
$||d_1|| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{\mathbf{0.5}} = (42)^{\mathbf{0.5}} = 6.481$
$||d_2|| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{\mathbf{0.5}} = (6)^{\mathbf{0.5}} = 2.245$

$\cos(d_1, d_2) = .3150$

Document similarity

- We do this by vectorizing our documents using the `utils` module
  - Instead of vectorizing words in terms of characters
- Then we can use cosine similarity (not distance) to measure the similarity between two documents
  - Since we use bag-of-words models, the sentences need not be in the same word order (desirable feature)
- We will also use two other document similarity measures
  - Hellinger-Bhattacharya distance (HB-distance, Bhattacharya distance)
  - Okapi BM25 ranking (BM25)

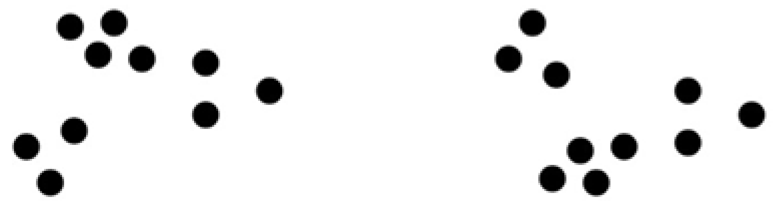
Document similarity

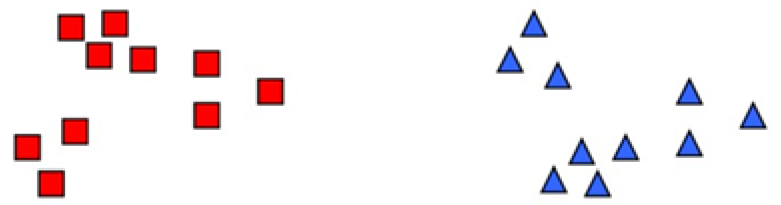
- For real--world problems dealing with large documents there are many optimized metrics that can (and should) be used
  - `cosine_similarity()` function of `sklearn.metrics.pairwise`
  - `gensim`'s `similarities` module or `cossim()` function from `gensim.matutils` module
  - `gensim`'s `hellinger()` function from the `gensim.matutils` module
  - `gensim.summarization` package has an implementation of `bm25`
- Try loading bigger corpora and test and compare the output of these functions
- Used in IR systems – e.g. in solr, Elasticsearch (on top of Lucene)
  - See https://www.elastic.co/blog/found--bm--vs--lucene--default--similarity

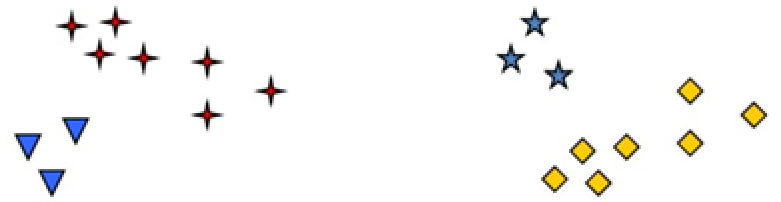
Document clustering

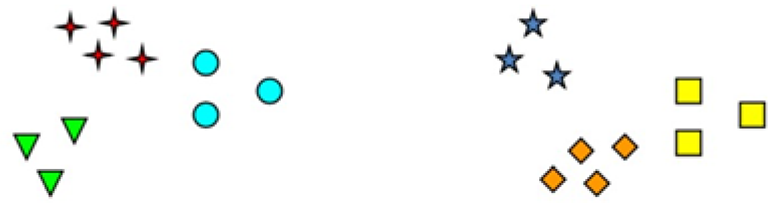
- In text/document classification, we categorize text/documents using pre--labelled ‘training data’
- What happens if we don’t have the luxury of annotated data?
- That is where document clustering is needed
    - We need unsupervised learning to solve this
- There are multiple ways to cluster any collection of data
- Many clustering algorithms usually require you to specify the number of clusters – why may this be not so bad?
- Evaluating how good a clustering scheme automatically is hard

Ambiguity of clustering

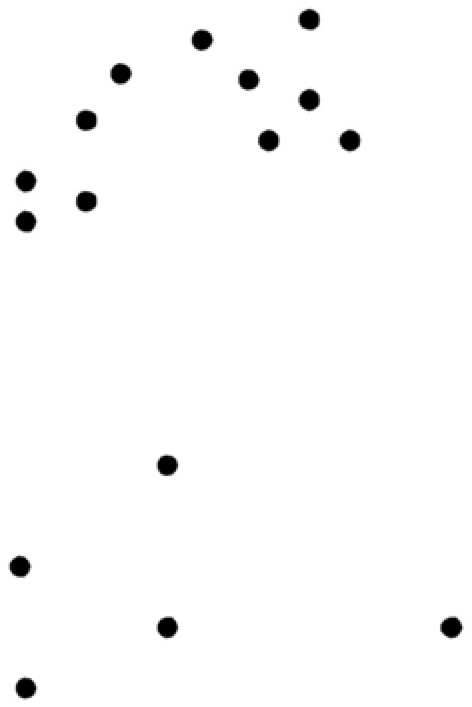

How many clusters?


Two Clusters

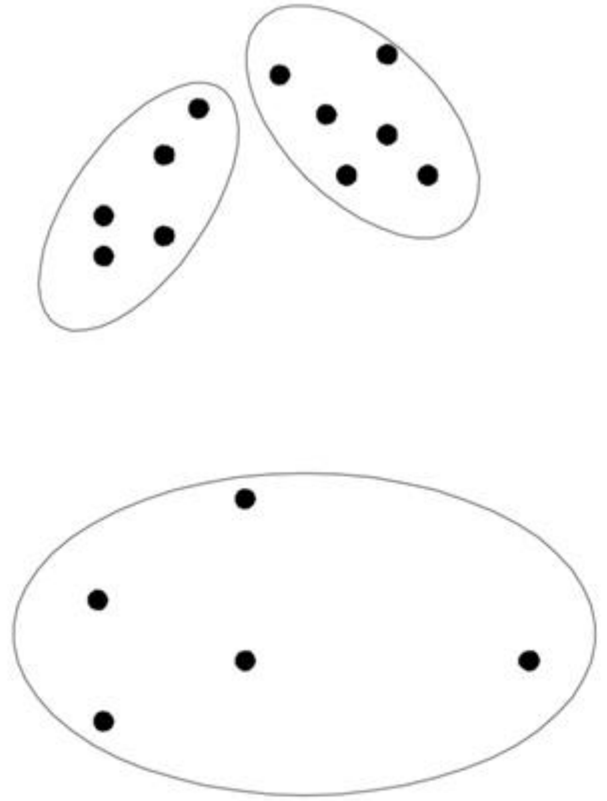

Four Clusters


Six Clusters

Ambiguity of clustering – paritional/hard

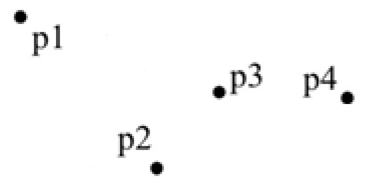


Input Data

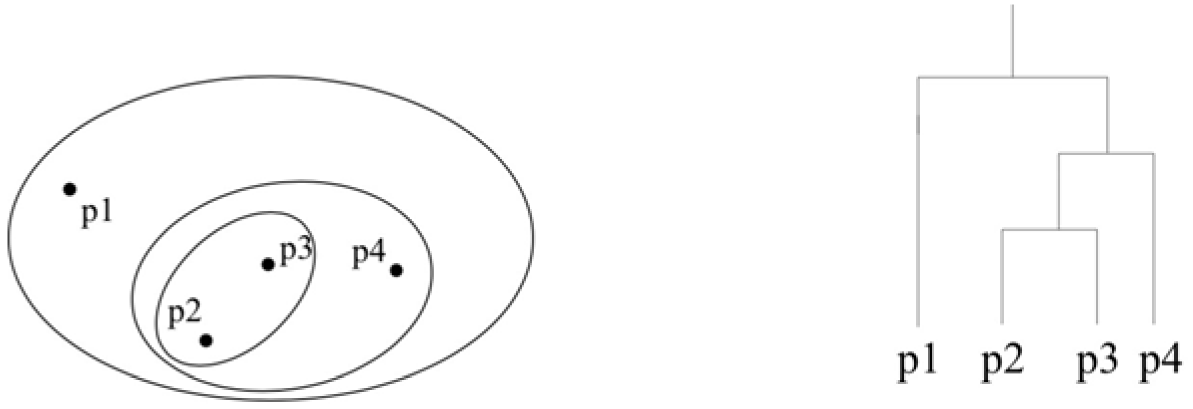


A Partitional Clustering

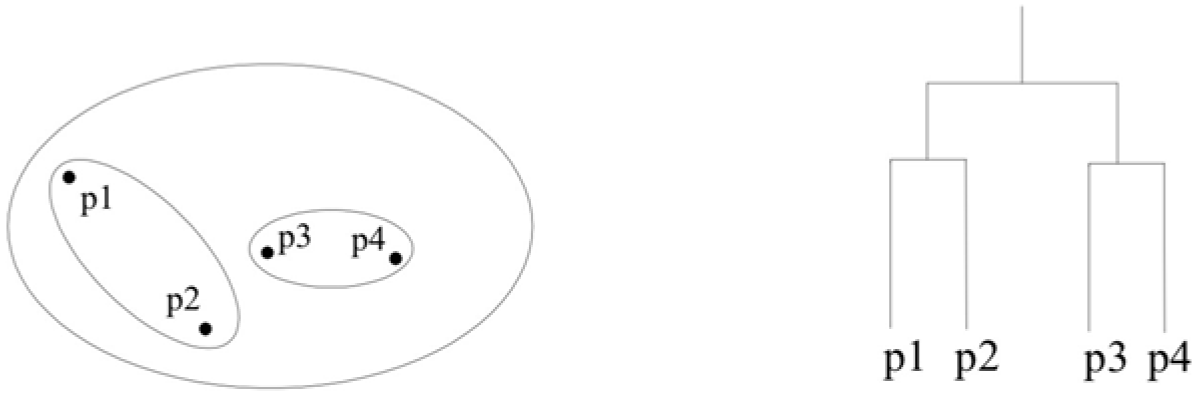
Ambiguity of clustering – hierarchical



Input Data



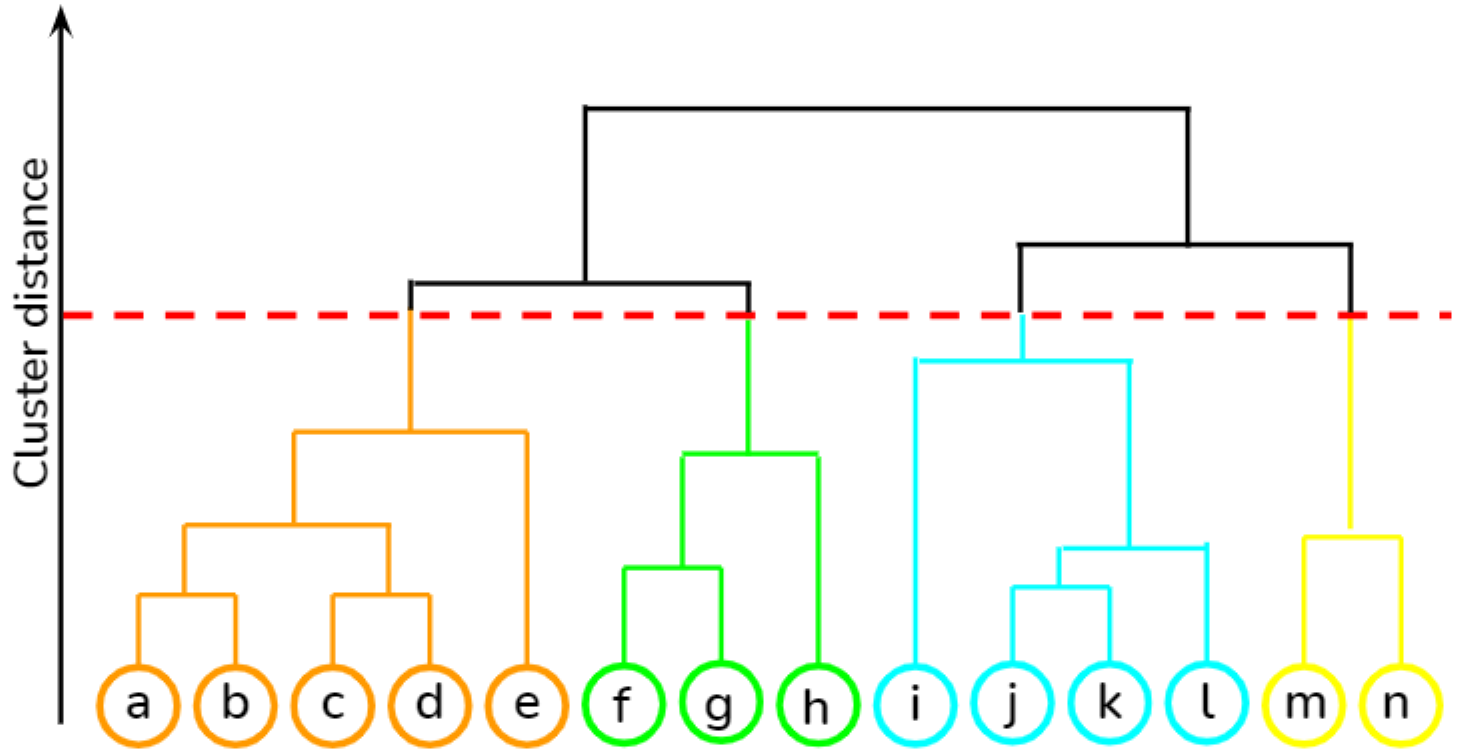
Clustering Solution 1

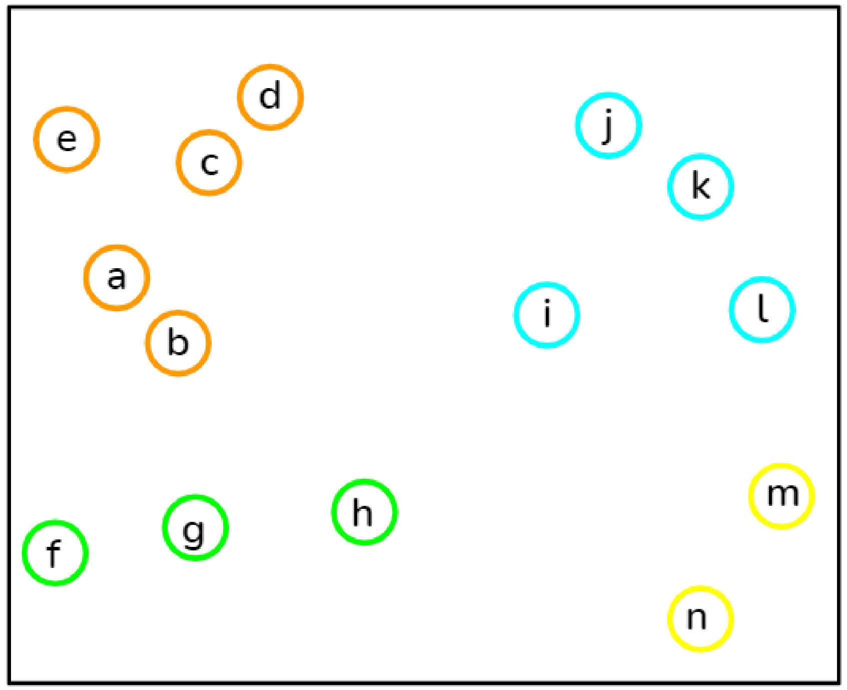


Clustering Solution 2

Clustering algorithms

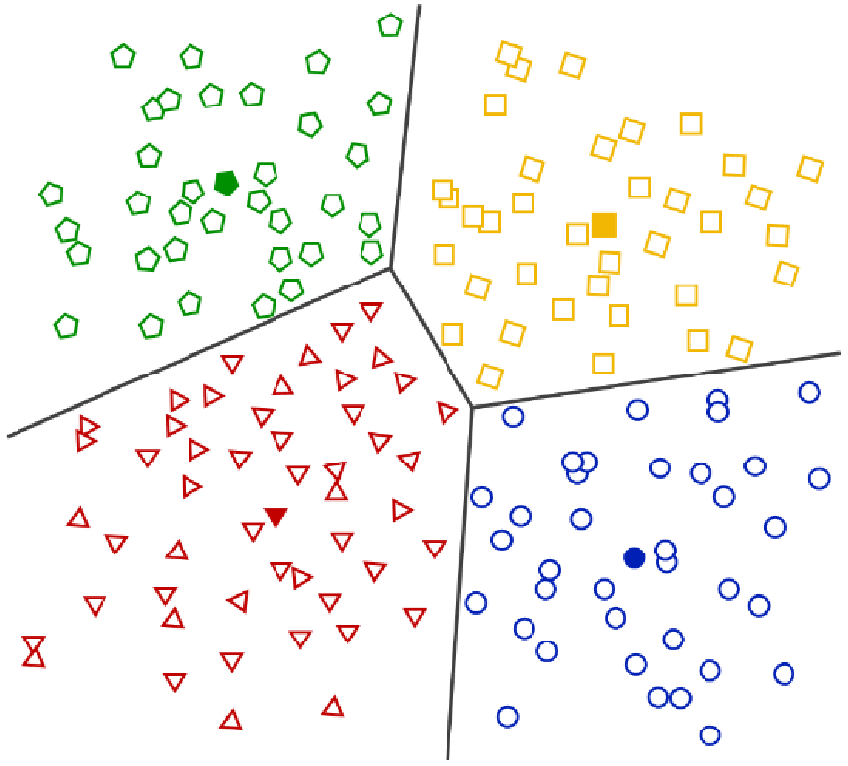
- Hierarchical clustering
  - Visualized using a *dendrogram*





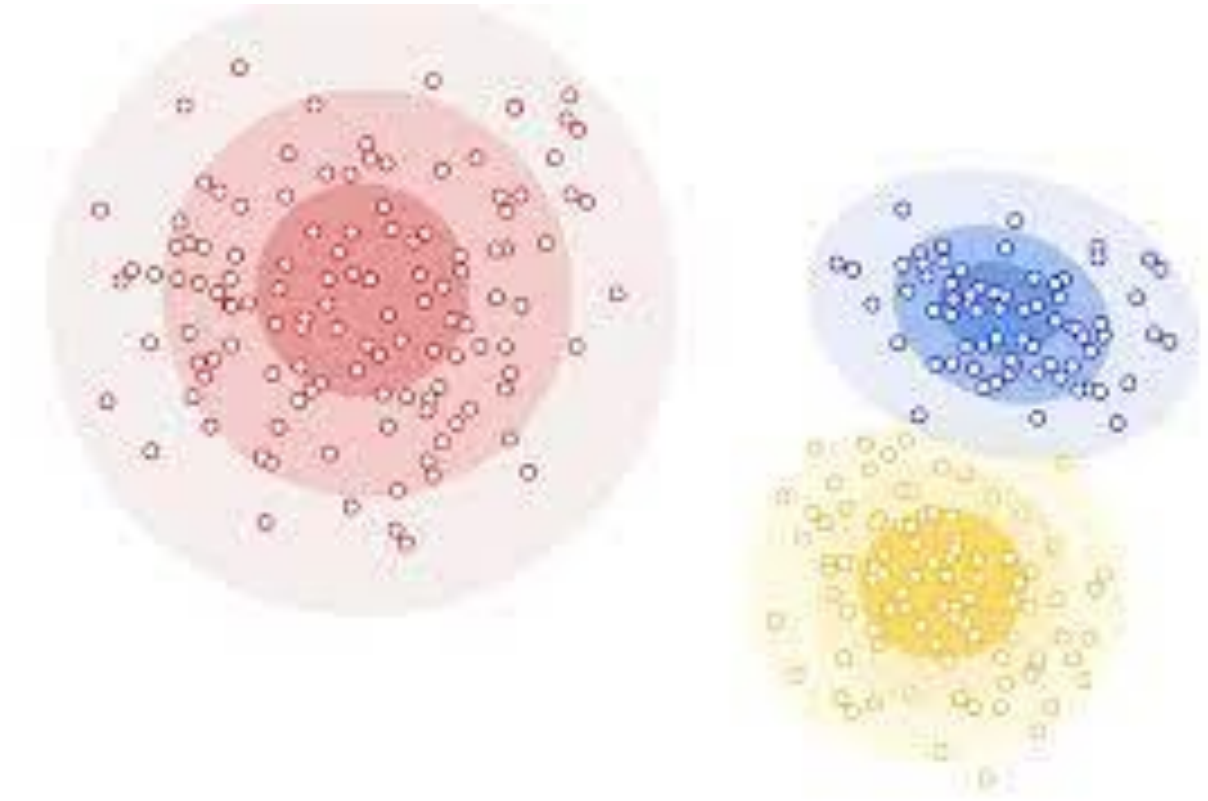
Clustering algorithms

- Centroid based clustering
  - Build clusters using central representative members per cluster (k–means)



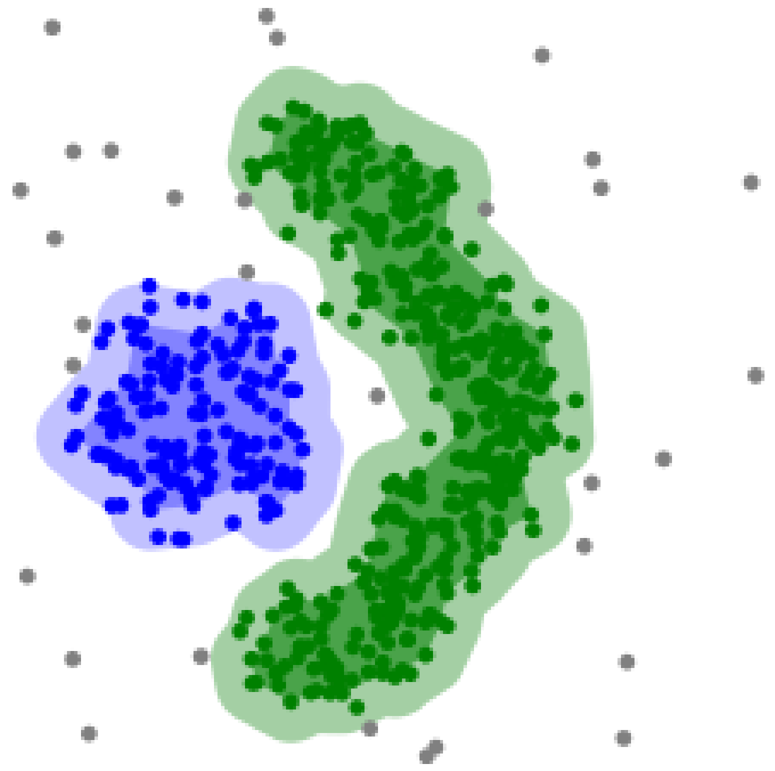
Clustering algorithms

- Distribution based clustering
  - Uses the concept that objects having similar distributions should be together



Clustering algorithms

- Density based clustering
  - Uses the high density areas in the space for clustering (DBSCAN)



Clustering algorithms

- Those interested can refer to following links for general info
  - http://scikit--learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html
  - http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html
  - https://www.toptal.com/machine--learning/clustering--algorithms

- You are encouraged to try some of the other algorithms available via scikit--learn

K–means clustering

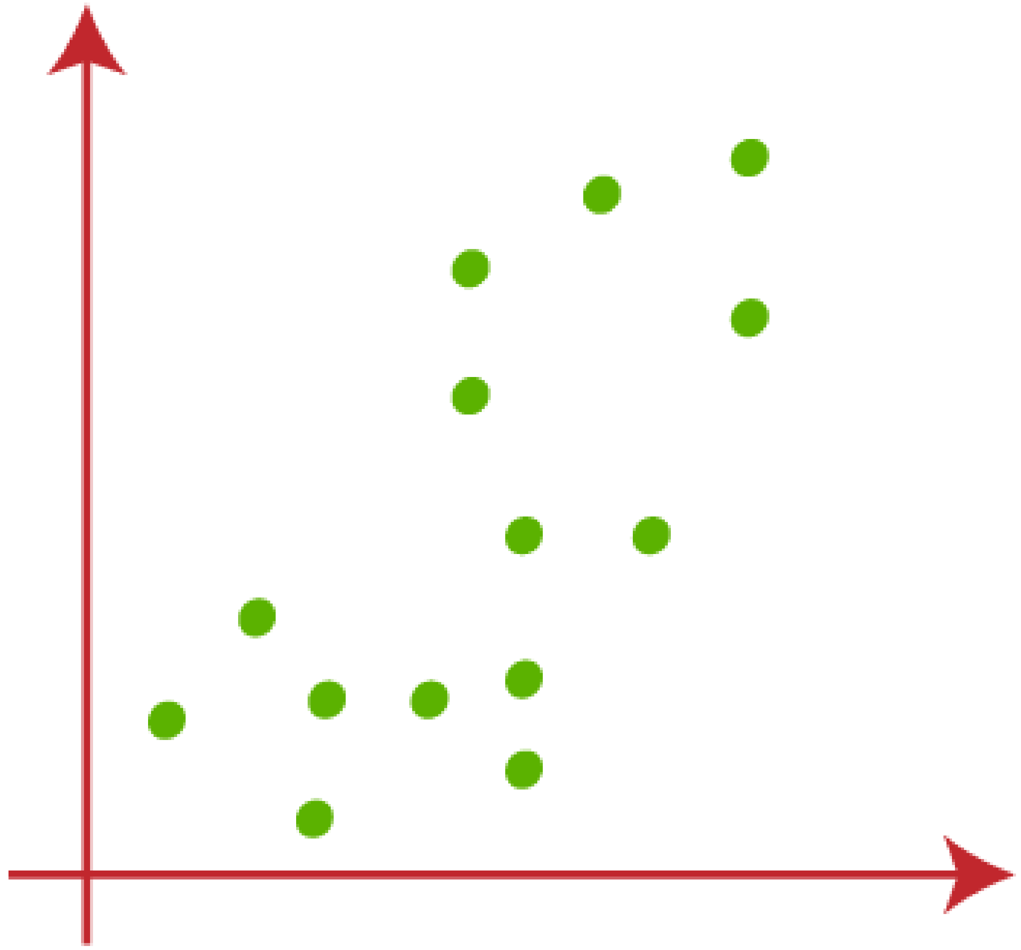
- Objective - group similar data points together and discover underlying patterns
- Tries to minimize within-cluster sum-of-squares (inertia)
- As its name implies, the number of clusters (k) needs to be specified
- This is so with all centroid--based algorithms
- Popular because it is highly scalable

K–means clustering

- For a dataset with N points
    - **Step-1:** Select the number K to decide the number of clusters.
    - **Step-2:** Select random K points or centroids.
    - **Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.
    - **Step-4:** Calculate the variance and place a new centroid of each cluster.
    - **Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
    - **Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.
    - **Step-7**: The model is ready.

K–means clustering

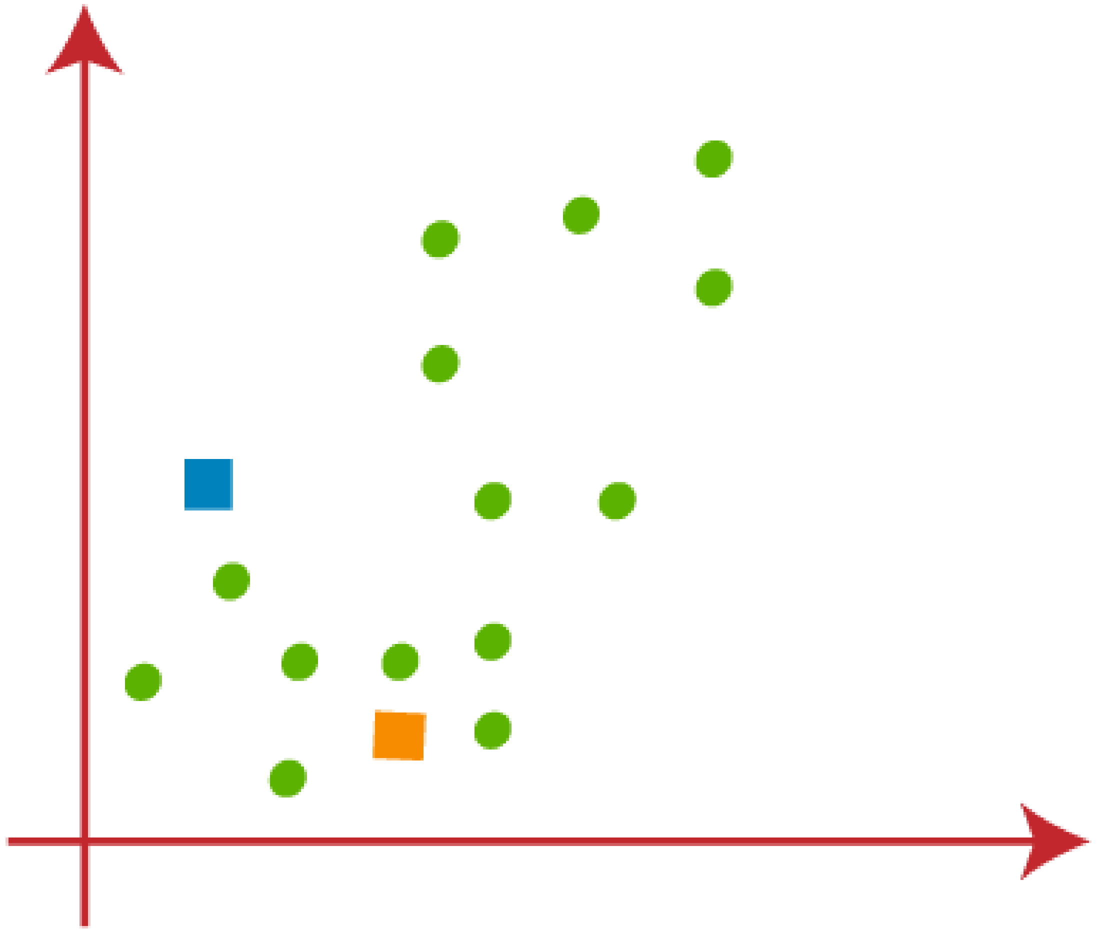
- Suppose we have two variables M1 and M2.



Suppose k = 2

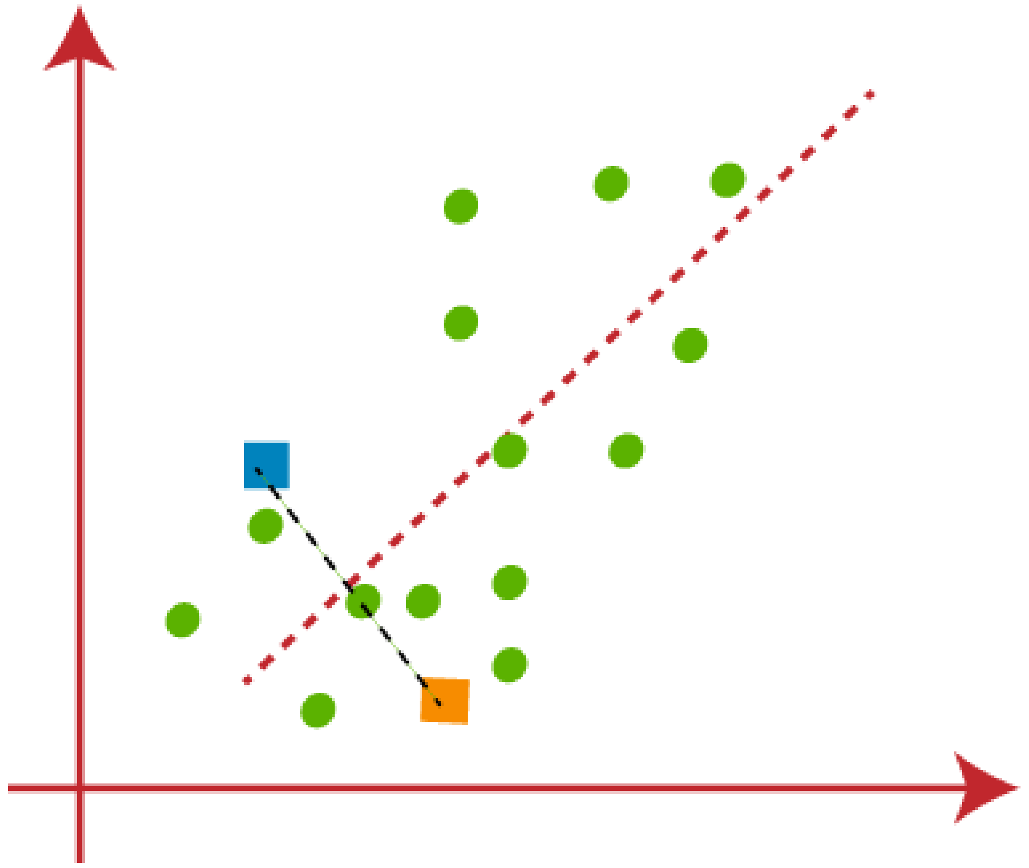
K–means clustering

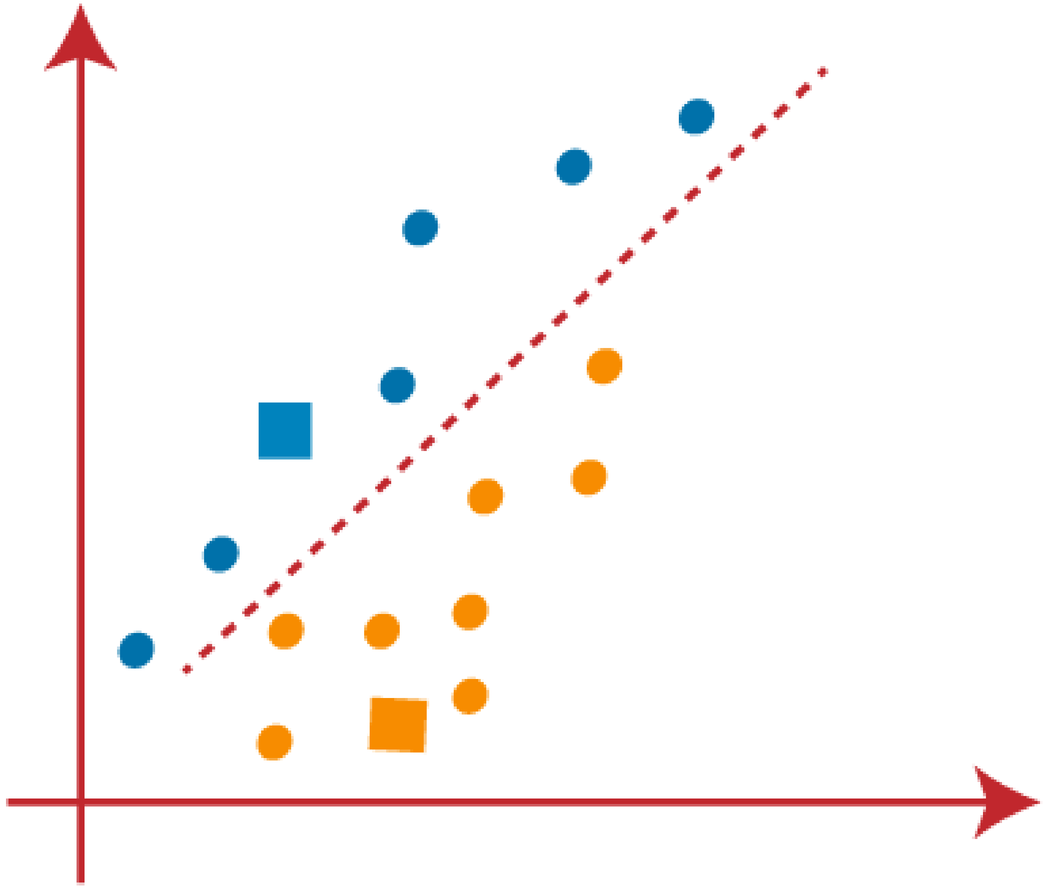
- Choose 2 points (k points) or centroid to form the cluster.
- These points can be either the points from the dataset or any other point



K–means clustering

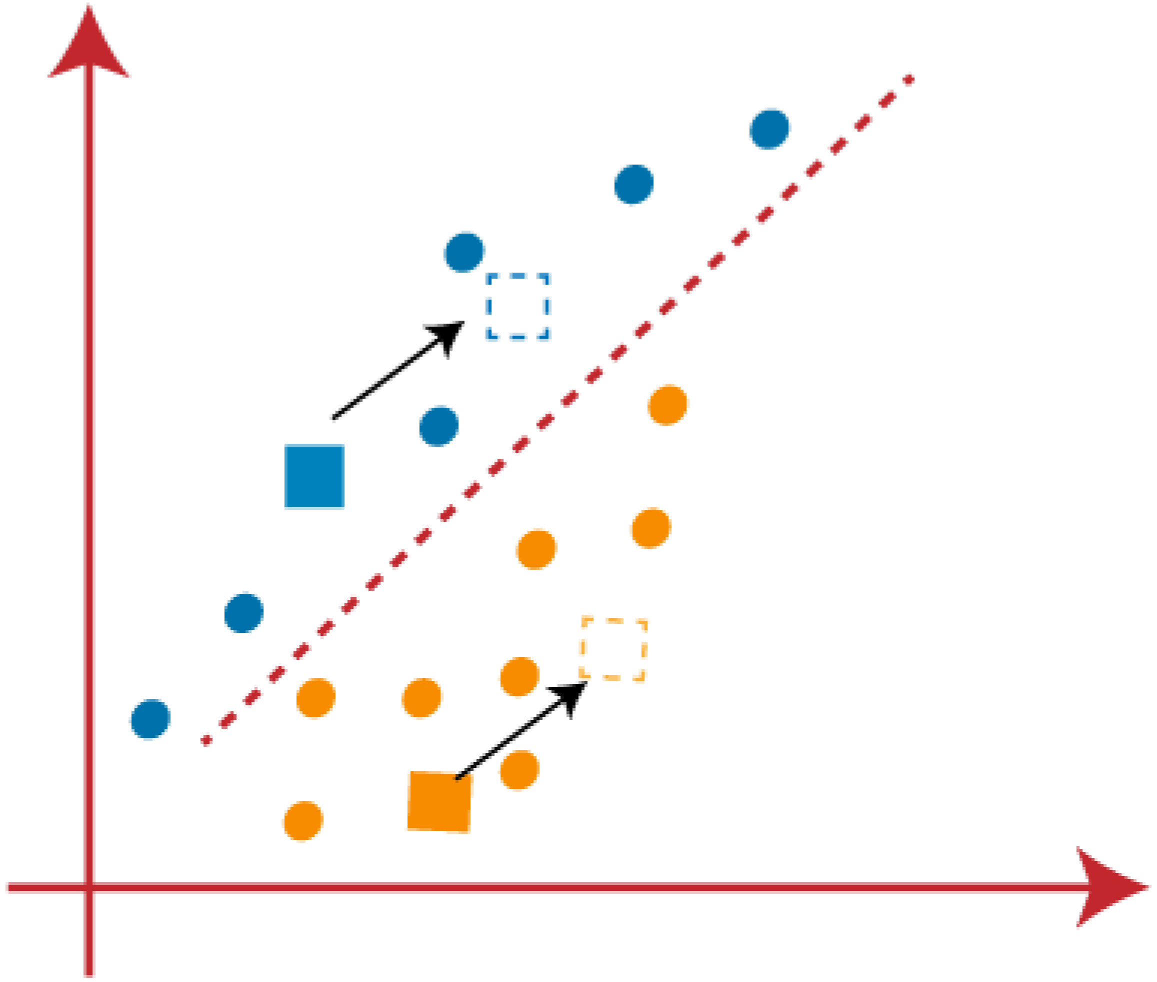
- Assign each data point of the scatter plot to its closest K-point or centroid.
- Compute it by calculating the distance between two points.
- Draw a median between both the centroids





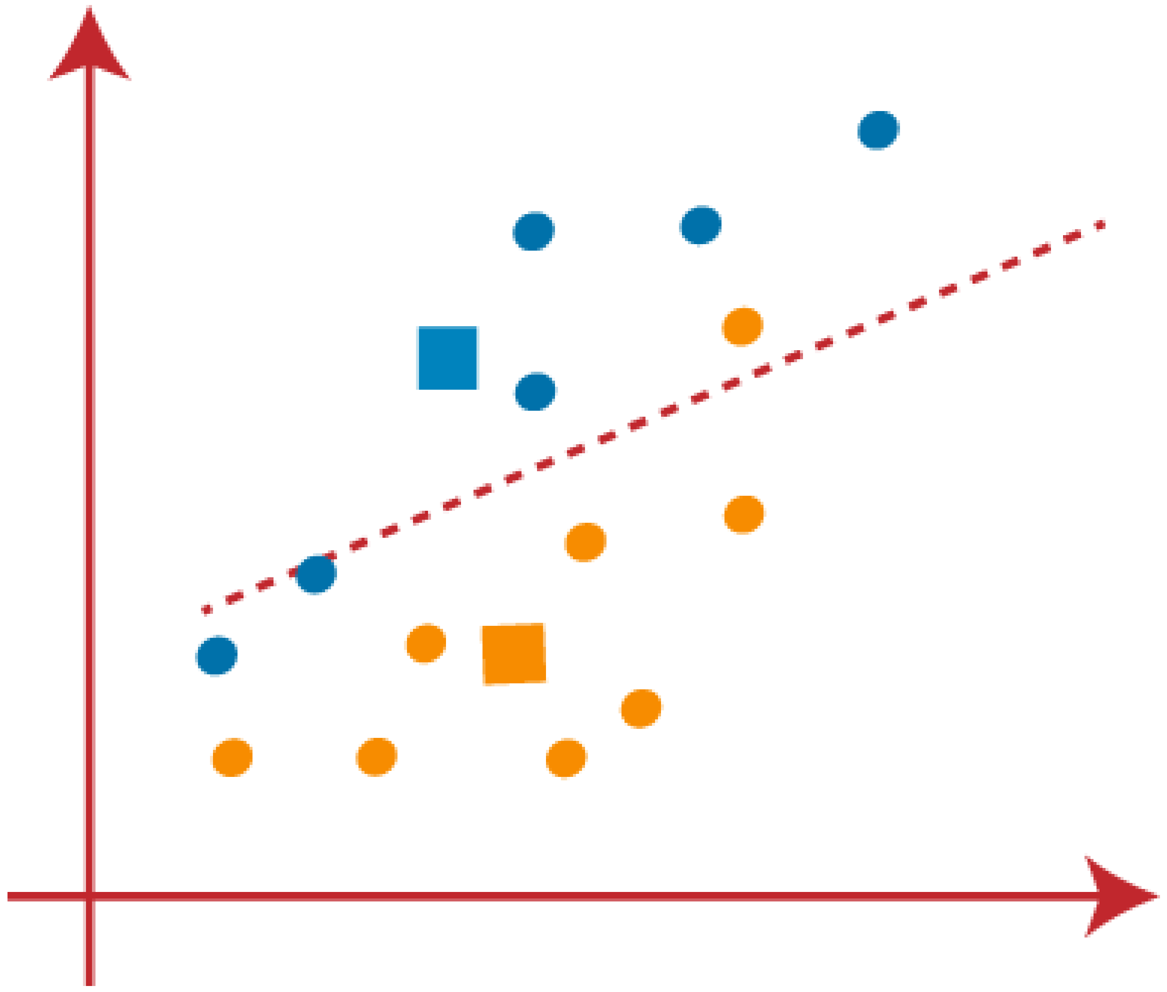
K–means clustering

- Find the closest cluster - repeat the process by choosing **a new centroid**



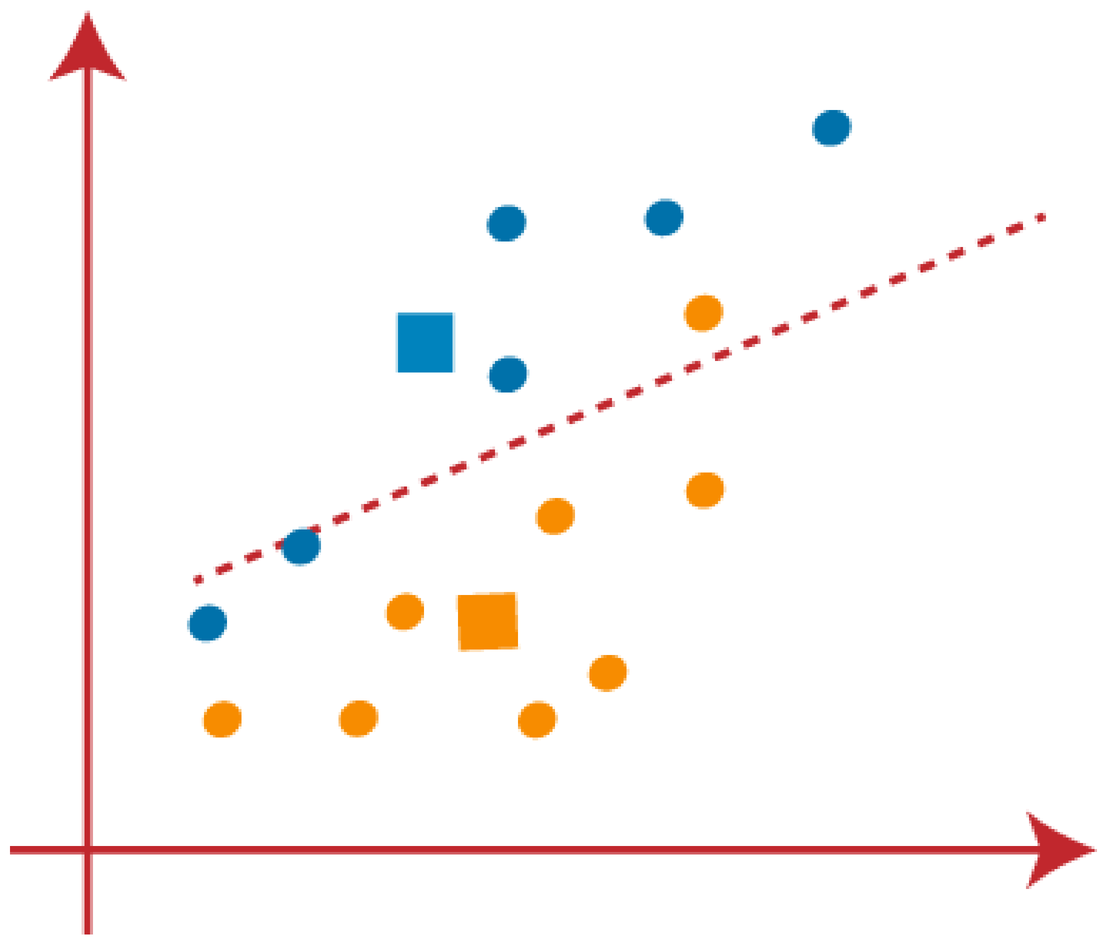
K–means clustering

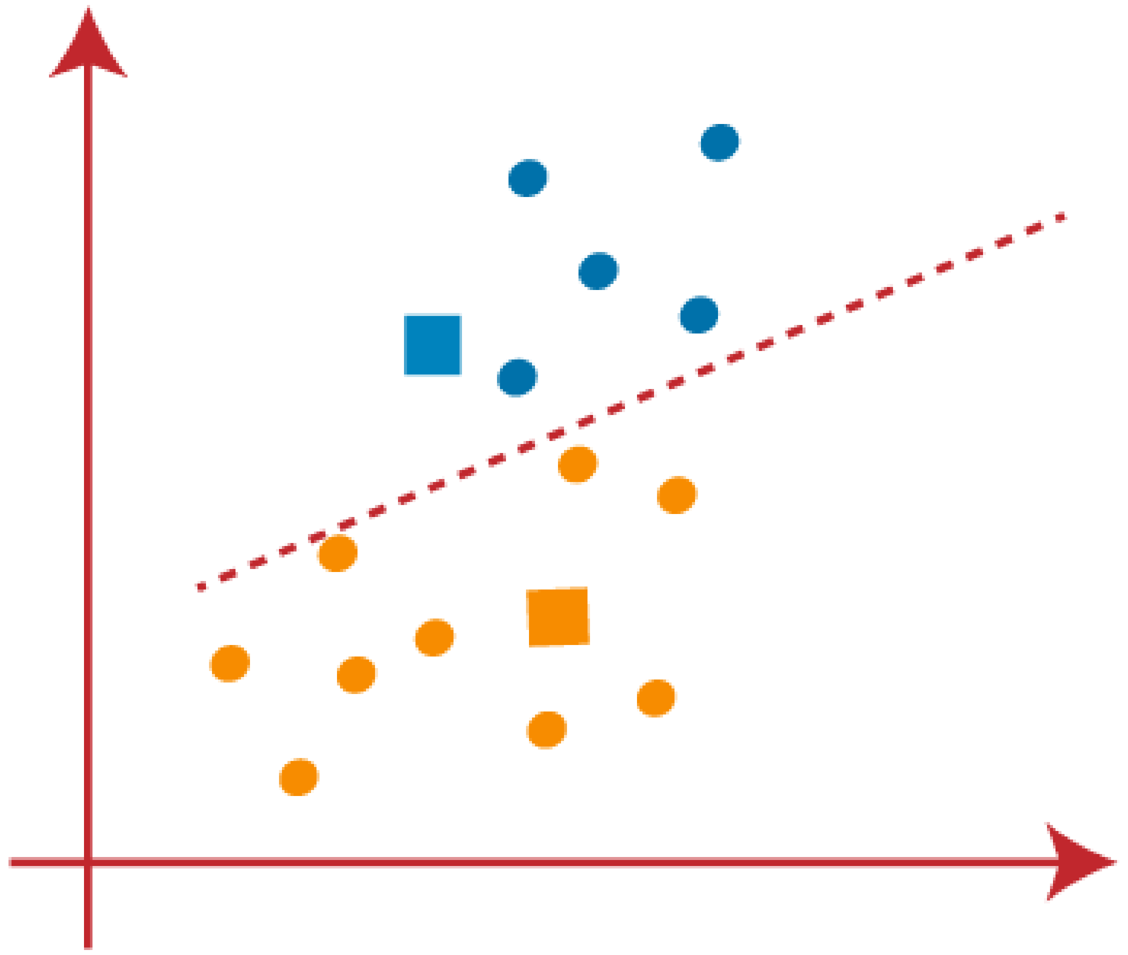
- Reassign each datapoint to the new centroid.



K–means clustering

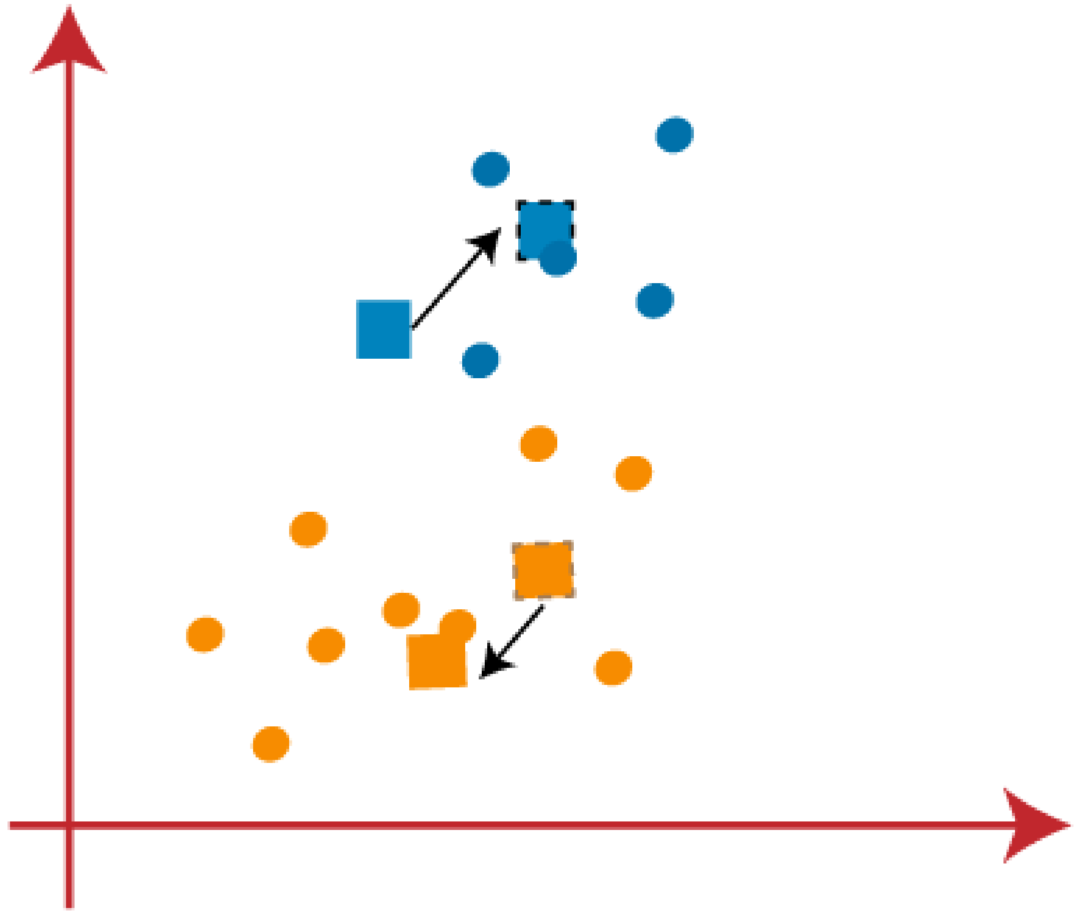
- Reassign each datapoint to the new centroid.

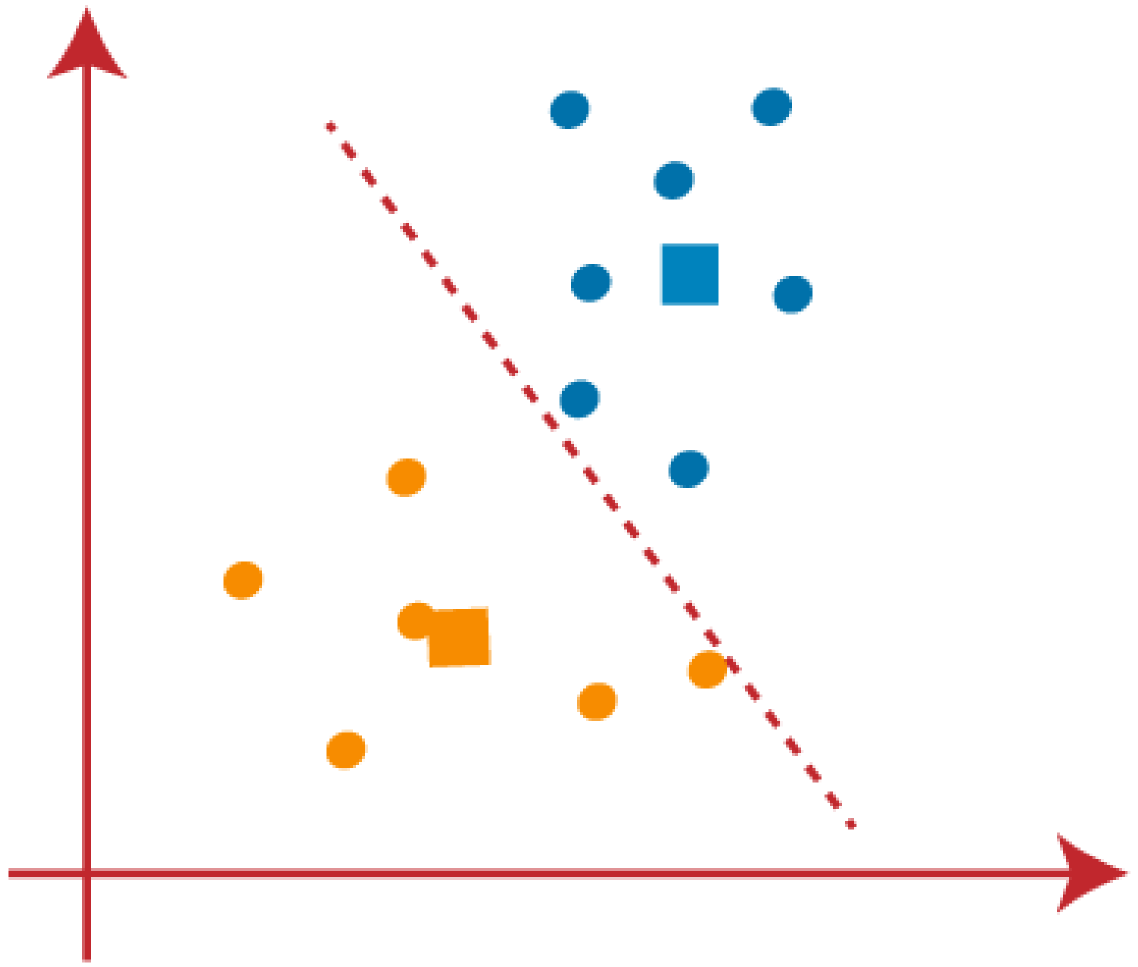




K–means clustering

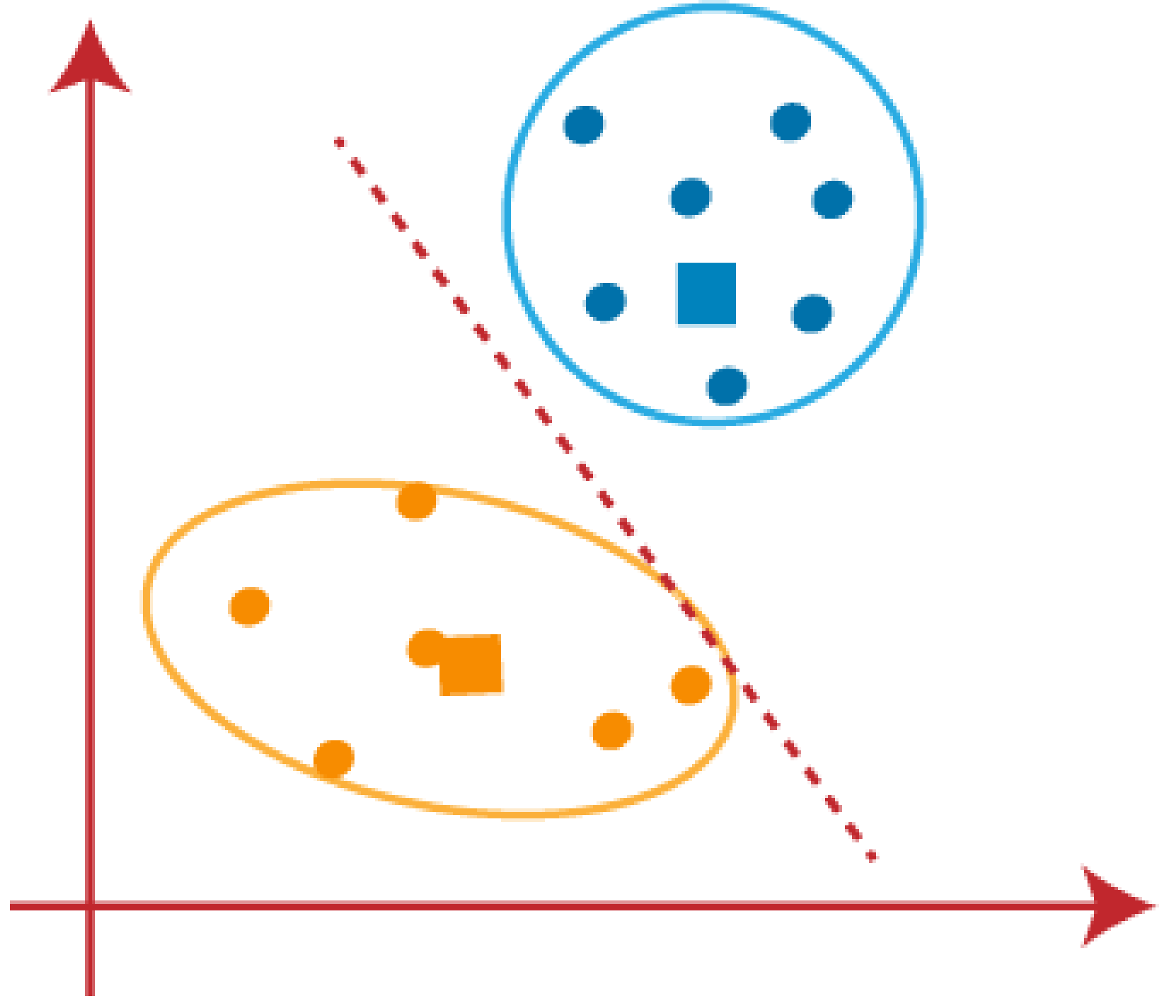
- Repeat the process





K–means clustering

- Model is formed



K–means algorithm (contd.)

- Can converge to a ‘local optimum’
- In practice, we run this algorithm multiple times with different epochs
- As common, convergence is highly dependent on initial assignment
- Two common solutions
  - Take average of multiple iterations with multiple random initializations
  - Initialize the centroids to be far apart from each other (kmeans++)
    - Implemented in scikit--learn
- Cannot visualize owing to the data generally being high--dimensional
  - We use PCA or MDS (multidimensional scaling) to view in 2 or 3 dimensions

Choosing the value of K

- Performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms
- Choosing the optimal number of clusters is a big task
- Some methods that can be used
    - The Elbow method
    - The Silhouette Method
    - …

The Elbow Method

- Probably the most well-known method for determining the optimal number of clusters
- Calculate the **Within Cluster Sum of Squares (WCSS)**

$$\text{WCSS} = \sum_{P_i \text{ in Cluster1}} \text{distance}(P_i\, C_1)^2 + \sum_{P_i \text{ in Cluster2}} \text{distance}(P_i\, C_2)^2 + \sum_{P_i \text{ in CLuster3}} \text{distance}(P_i\, C_3)^2$$

- $\sum_{P_i \text{ in Cluster1}} \text{distance}(P_i\, C_1)^2$: is the sum of the square of the distances between each data point and its centroid within a cluster1

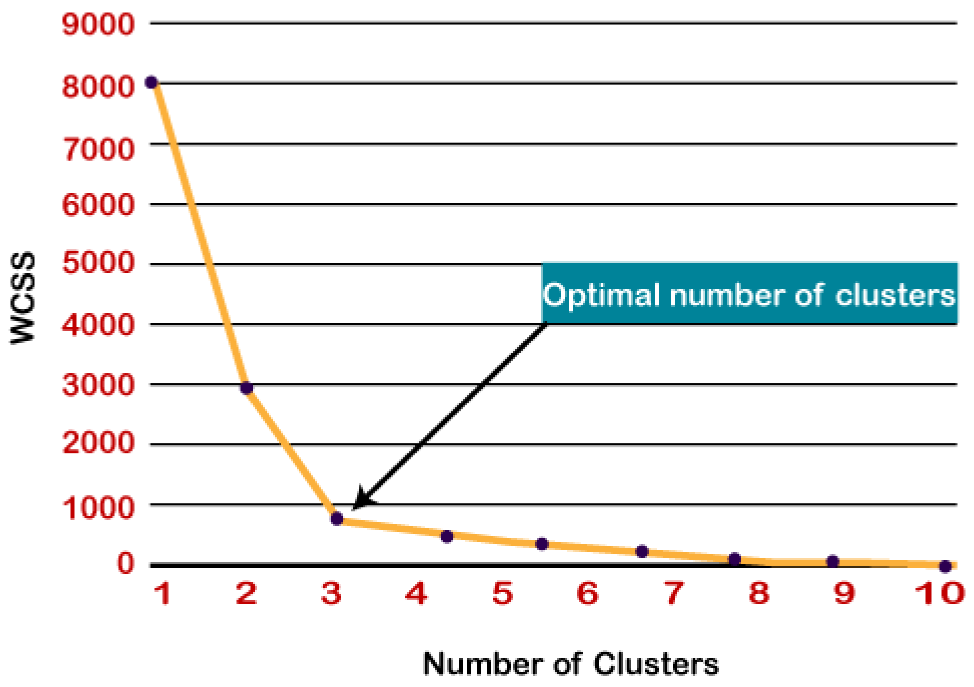
- To measure the distance between data points and centroid

  Euclidean distance, Manhattan distance, etc.

The Elbow Method

- Elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.



Affinity Propagation algorithm

- AP tries to build clusters without a pre--specified number
  - Uses the concept of ‘message passing’ between data points
- Finding *exemplars* that act as representatives of dataset
  - Based on the messages passed between each pair of data point
  - Hence, convergence takes a long time
- We use cosine similarity as the measure in our message passing
- Related to *spectral clustering*

Affinity Propagation (contd.)

- Iteratively proceed by executing the following message passing steps:
  - Send ‘responsibility’ updates between pairs of data points as a matrix R
  - Send ‘availability’ updates by as a matrix A to all data pairs
  - Keep doing these steps until convergence

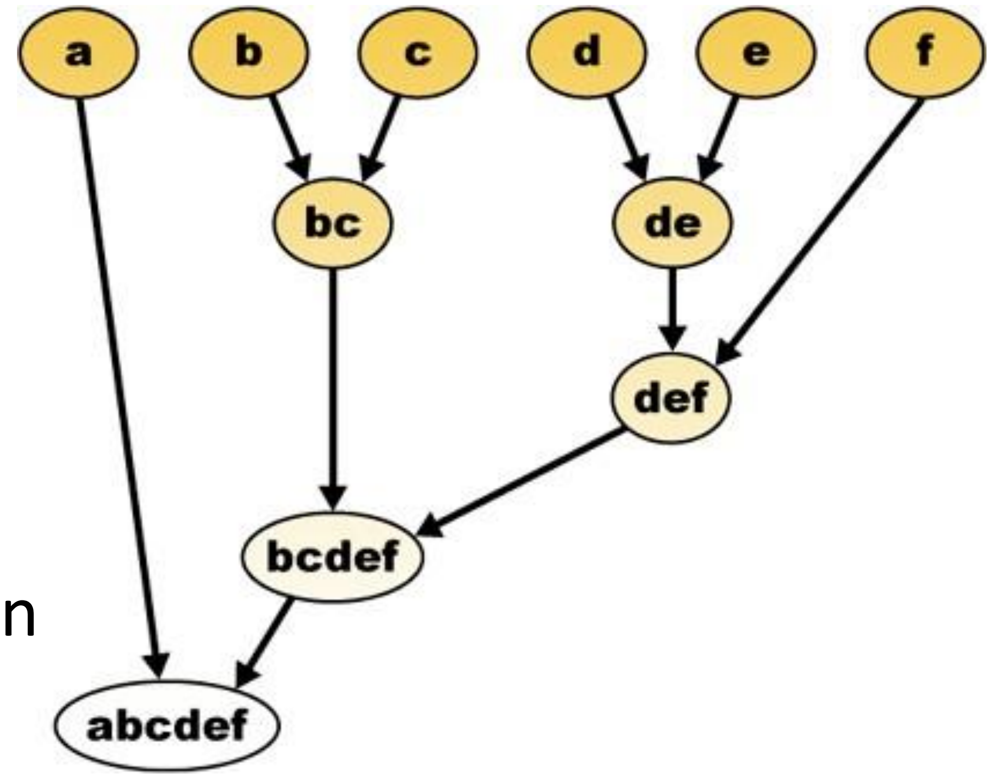
$$r(i,k) \leftarrow sim(i,k) - \max_{k' \neq k}\left\{a(i,k') + sim(i,k')\right\}$$

$$a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k))\right) \text{ for } i \neq k$$

- Task: find the best intuitive description of the AP algorithm online!

Ward’s Agglomerative Hierarchical Clustering

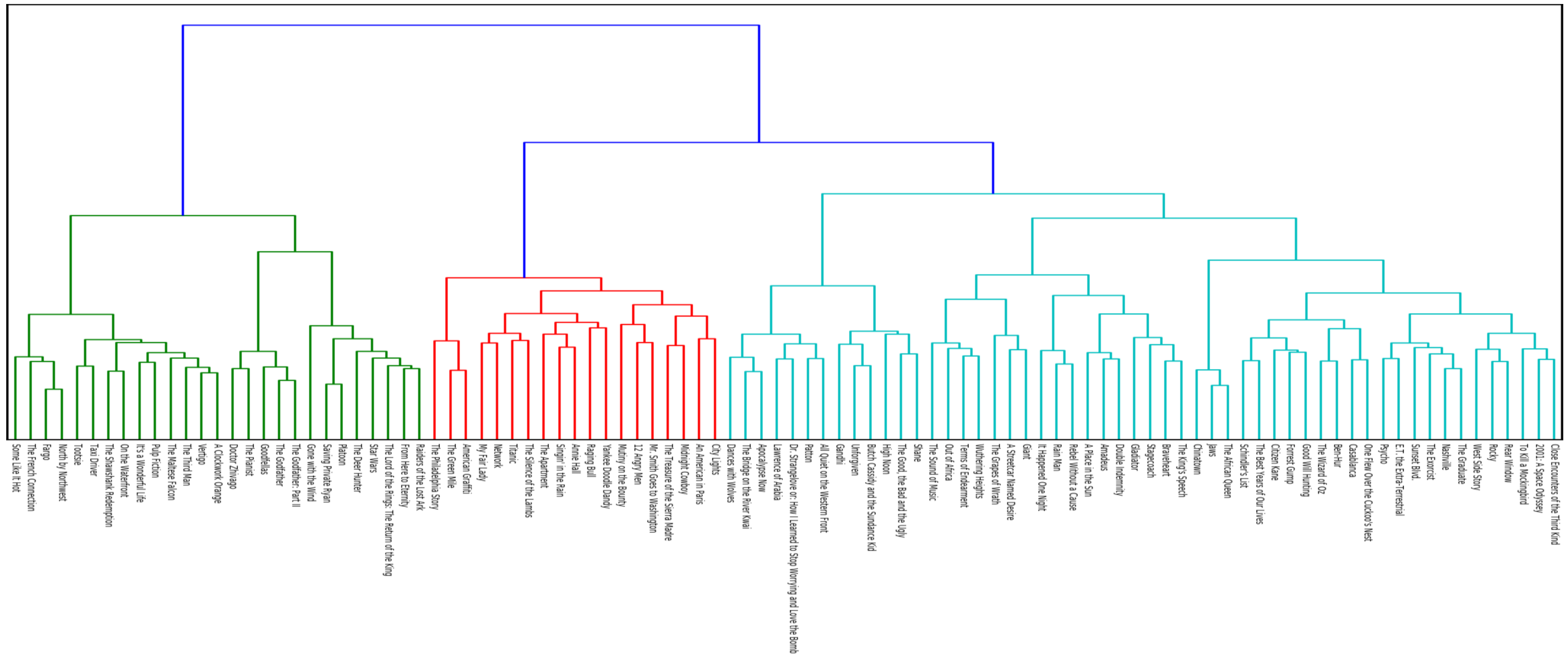
- Hierarchical clustering attempts to build nested hierarchies
- Two broad types:
  - Agglomerative (bottom--up): each point in its own class initially
  - Divisive (top--down): all points in one class initially
- Merges and splits (respectively) happen in ‘greedy’ fashion



Ward’s Agglomerative Hierarchical Clustering

- Deciding which points (or clusters) should join:
  - Based on a *distance metric* to measure the similarity or dissimilarity degree between data points (we use cosine similarity)
  - Using a *linkage criterion* that determines the metric to be used for the merging strategy of clusters (we use Ward’s method)
- Ward’s method minimized the sum of squared differences within all the clusters (variance minimization)
  - Minimize variance in each cluster using objective function such as L2 norm
- We use `matplotlib` to plot the final resulting dendrogram

Ward's Agglomerative Hierarchical Clustering



Cluster validation

- Comparing the clustering results to *ground truth* (externally known results)
  - External Index
- Evaluating the quality of clusters *without* reference to external information
  - Use only the data
  - Internal Index

Cluster validation – external

- **Notation**
  - N: number of objects in the data set
  - $P=\{P_1,\dots,P_s\}$: the set of “ground truth” clusters
  - $C=\{C_1,\dots,C_t\}$: the set of clusters reported by a clustering algorithm
- **The “incidence matrix”**
  - $N \times N$ (both rows and columns correspond to objects)
  - $P_{ij} = 1$ if O_i and O_j belong to the same “ground truth” cluster in P; $P_{ij}=0$ otherwise
  - $C_{ij} = 1$ if O_i and O_j belong to the same cluster in C; $C_{ij}=0$ otherwise

Cluster validation – external (contd.)

- A pair of data object (O_i,O_j) falls into one of the following categories
  - SS: C_{ij}=1 and P_{ij}=1; (agree)
  - DD: C_{ij}=0 and P_{ij}=0; (agree)
  - SD: C_{ij}=1 and P_{ij}=0; (disagree)
  - DS: C_{ij}=0 and P_{ij}=1; (disagree)
- **Rand index** $$Rand = \frac{|Agree|}{|Agree|+|Disagree|} = \frac{|SS|+|DD|}{|SS|+|SD|+|DS|+|DD|}$$
  - may be dominated by DD
- **Jaccard Coefficient** $$Jaccard\ \ coefficien\,t = \frac{|SS|}{|SS|+|SD|+|DS|}$$

Cluster validation – external (contd.)

Clustering

	g 1	g 2	g 3	g 4	g 5
g 1	1	1	1	0	0
g 2	1	1	1	0	0
g 3	1	1	1	0	0
g 4	0	0	0	1	1
g 5	0	0	0	1	1

Groundtruth

	g 1	g 2	g 3	g 4	g 5
g 1	1	1	0	0	0
g 2	1	1	0	0	0
g 3	0	0	1	1	1
g 4	0	0	1	1	1
g 5	0	0	1	1	1

Ground truth → Clustering

	Same Cluster	Different Cluster
Same Cluster	9	4
Different Cluster	4	8

$$Rand = \frac{|SS| + |DD|}{|SS| + |SD| + |DS| + |DD|} = \frac{17}{25}$$

$$Jaccard = \frac{|SS|}{|SS| + |SD| + |DS|} = \frac{9}{17}$$

Cluster validation – internal

- **Cohesion** is measured by the within cluster sum of squares

$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

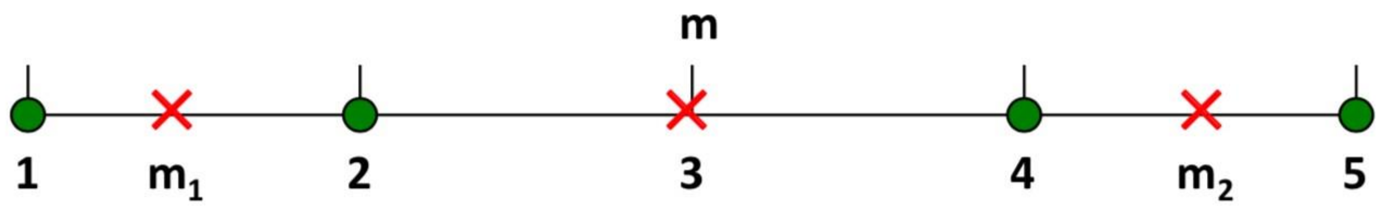
- **Separation** is measured by the between cluster sum of squares

$$BSS = \sum_i |C_i| (m - m_i)^2$$

where |*Ci*| is the size of cluster *i*, m is the centroid of the whole data set

- BSS + WSS = constant
- WSS (Cohesion) measure is called Sum of Squared Error (SSE)—a commonly used measure
- A larger number of clusters tend to result in smaller SSE

Cluster validation – internal (contd.)



K=1 :

$$WSS = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$$

$$BSS = 4 \times (3-3)^2 = 0$$

$$Total = 10 + 0 = 10$$

K=2 :

$$WSS = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$$

$$BSS = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$$

$$Total = 1 + 9 = 10$$

K=4:

$$WSS = (1-1)^2 + (2-2)^2 + (4-4)^2 + (5-5)^2 = 0$$

$$BSS = 1 \times (1-3)^2 + 1 \times (2-3)^2 + 1 \times (4-3)^2 + 1 \times (5-3)^2 = 10$$

$$Total = 0 + 10 = 10$$

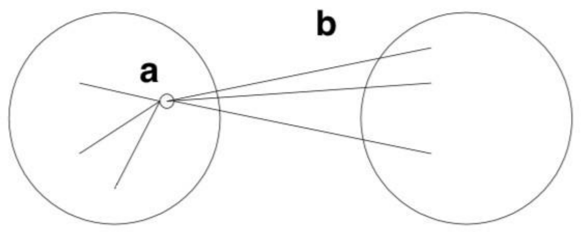
Cluster validation – internal (contd.)

- Silhouette Coefficient combines ideas of both cohesion and separation

- For an individual point, *i*
  - Calculate ***a*** = average distance of *i* to the points in its cluster
  - Calculate ***b*** = min (average distance of *i* to points in another cluster)
  - The **silhouette coefficient** for a point is then given by

    $s = 1 – a/b$ if $a < b$, ($s = b/a - 1$ if $a \geq b$, not the usual case)

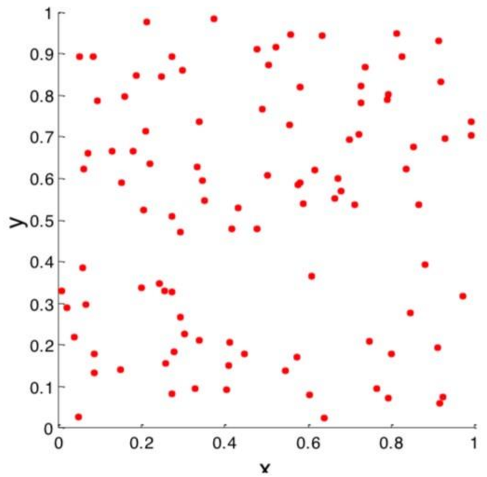
  - Typically between 0 and 1
  - The closer to 1 the better

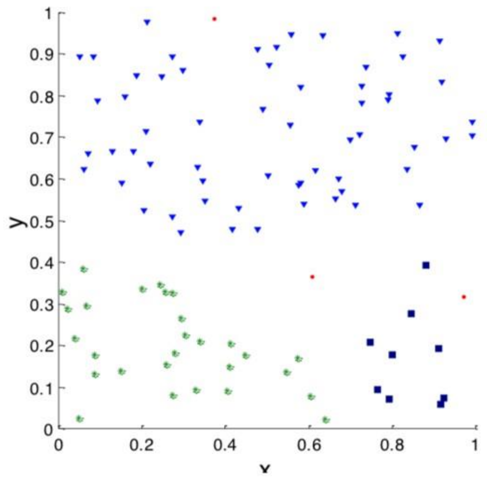


- Can calculate the Average Silhouette width for a cluster or a clustering

Are there clusters in the data?

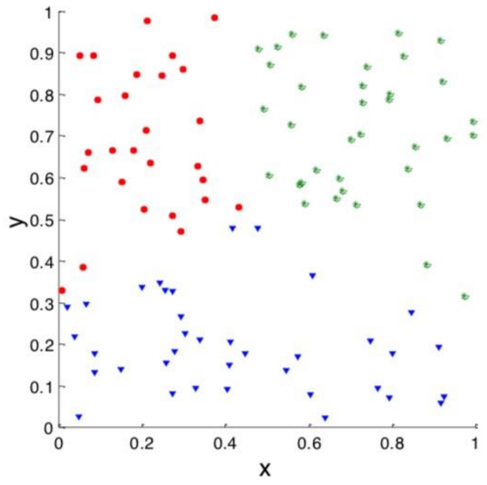
Random Points

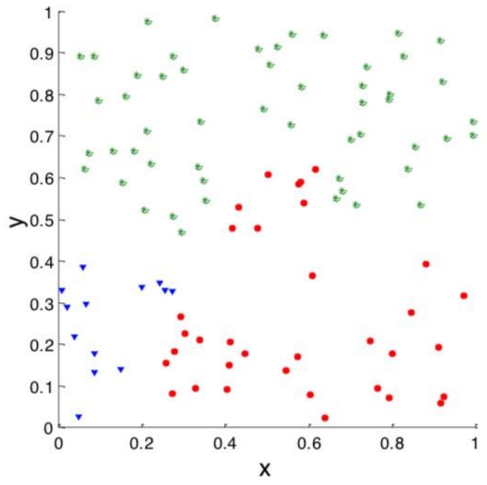




DBSCAN

K-means





Complete Link

Application: greatest movies of all time!

- IMDb contains details about movies and TV series
  - Together with reviews and ratings
- We will work with the ‘ultimate list’ (100 greatest movies)
  - http://www.imdb.com/list/ls055592025/
- We want to cluster the movies based on the title and synopsis
  - Already cleaned by various others (available as movie_data.csv)
- We load, normalize and extract features as before
  - Using bigram TF–IDF weights with min and max cutoffs
  - Then we use our clustering algorithms and compare results