Clustering

↳ grouping similar data points together into cluster based on characteristics/features

↳ · unsupervised (no target variable)

↳ Goal :-

- identify natural grouping in data
- minimize distance / dissimilarity between data points within the same cluster
- maximize distance / dissimilarity between data points in different clusters

group smaller distance together, larger distance apart

Measures for Distance

↳ Euclidean distance

- common measurement
- $\sqrt{(x-x_1)^2+(y-y_1)^2}$

↳ Manhattan distance

- absolute difference
- image / pattern recognition
- $|x_1-x_1|+|y_1-y_1|$

↳ Cosine Similarity

- based on cosine angle
- language & recommendation
- cos θ

require re-specification of number of cluster

can handle diff. type of data

good for well separated cluster

K-Means

↳ clustering algorithm that partition dataset into K-distinct clusters, where K is specified

↳ 1. Initialization :

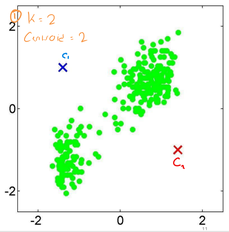
↳ select K random points as centroid

- by random selection

↳ randomly select from dataset

- by sort & split

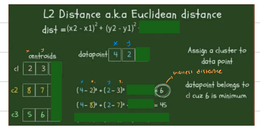
↳ sort dataset & split into K portion



2. Assignment :

↳ Assign each data points to the closest centroid cluster, by distance measure (Euclidean, manhattan...)

- Euclidean distance

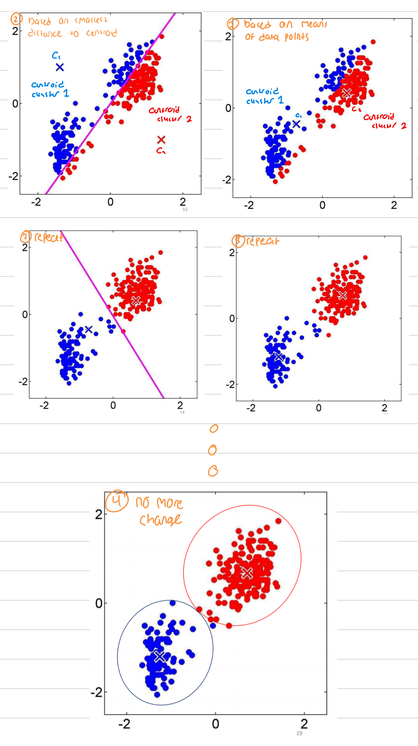


do for each data point

3. Update :

↳ recalculate centroids as the means distance of data point in the cluster,

- by simple Mean ← K-"mean"



4. Convergence :

↳ Repeat Step 2 & 3 until the change in centroids below threshold / max iteration

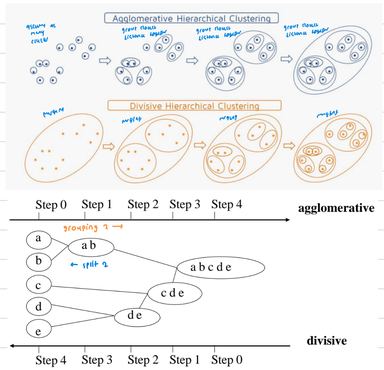
NOT in scope

Hierachal Clustering

↳ recursively merge & divide clusters based on similarity

↳ 2 types :-

- Agglomerative (Bottom-up)
  recursively merge two cluster together w/ the lowest between-cluster dissimilarity
- Divisive (Top-down)
  Split the least coherent cluster



↳ How to calculate Dissimilarity :-

- Have to measure the dissimilarity for two disjoint groups G and H, $D(G,H)$ is computed from pairwise dissimilarities $D(i,j)$ with $i \in G, j \in H$

- Single Linkage: tends to yield extended clusters. (nearest point)
  $$D_{SL}(G,H) = \min_{i \in G, j \in H} D(i,j)$$
- Complete Linkage: tends to yield round clusters. (farthest point)
  $$D_{CL}(G,H) = \max_{i \in G, j \in H} D(i,j)$$
- Group Average: tradeoff between them. Not invariant under monotone increasing transform. (ave mean point)
  $$D_{GA}(G,H) = \frac{1}{n_G n_H} \sum_{i \in G, j \in H} D(i,j)$$

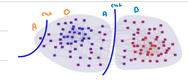
Spectral Clustering

↳ datapoints as vertices of a Graph, all vertices are connected by an Edge, all Edges have Weight

↳ weight↑, adjacent V are very similar
weight↓, adjacent V are very dissimilar

↳ Clustering using loss function

cut(A,B) = $\sum w_{u,v}$



↳ Types of cut :-

- MinCut criterion
  ↳ find partition (A,B) between that minimize cut(A,B)
  ↳ ignored subgraph size result
- Normalized cut criterion
  ↳ more balanced partitions

↳ Steps :-

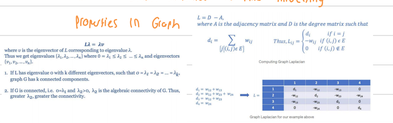
1. Compute a similarity graph

↳ i) create an undirected Graph
ii) create adjacency matrix

2. Project data to low dimensional space

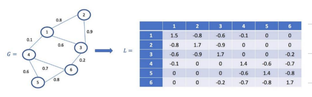
↳ transform the space so that when 2 point are close they are always in the same cluster & vice versa

↳ compute Graph laplacian to find interesting properties in Graph



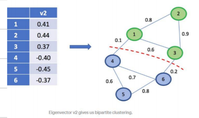
Spectrum of the Laplacian
$0 = \lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$

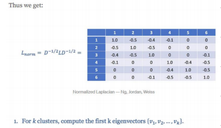
Let us consider an example with numbers:



We then compute eigenvalues and eigenvectors

3. Create clusters





Eigenvector v2 gives us bipartite clustering.

1. For k clusters, compute the first k eigenvectors $(v_1, v_2, \ldots, v_k)$.
2. Stack the vectors vertically to form a matrix with the vectors as columns.
3. Represent every node by the corresponding row of this new matrix. These rows form the feature vectors of the nodes.
4. Use K-Means Clustering to now cluster these points into k clusters $(c_1, c_2, \ldots, c_k)$.