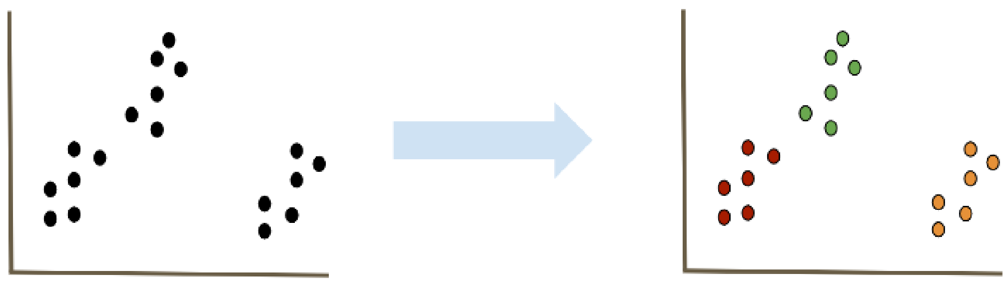
Lecture 4: Clustering - Kmeans

1) What is clustering?

a) Clustering: grouping/assignment of objects(data points) such that objects in same group cluster are:
   i) Similar to one another
   ii) Dissimilar to objects in other groups

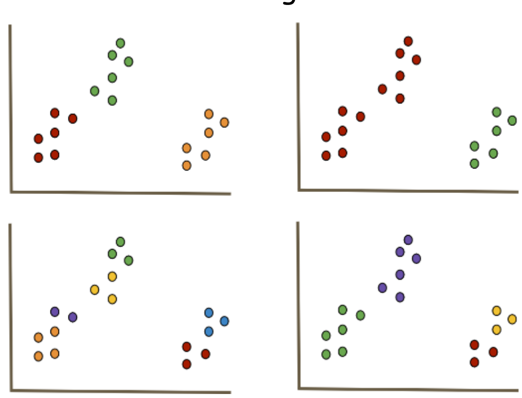


2) Applications

a) Outlier detection/anomaly detection
   i) Data cleaning/processing
b) Filling gaps in data
   i) Using same marketing strategy for similar people

3) Clustering Problem

a) Given collection of data points, find clustering such that
   i) Similar data points are in same cluster
   ii) Dissimilar data points in different clusters
b) Clusters can be ambiguous

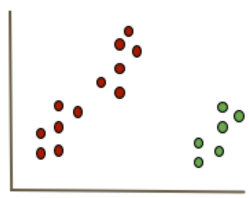


4) Types of Clusterings

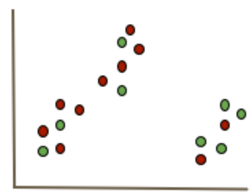
a) Partitional: Each object belongs to exactly one cluster
b) Hierarchical: A set of nested clusters organized in a tree
c) Density-Based: Defined based on the local density of points
d) Soft Clustering: Each point is assigned to every cluster with a certain probability

5) Partitional Clustering

a) Given **n** data points and a number of **k** clusters: partition **n** data points into **k** clusters



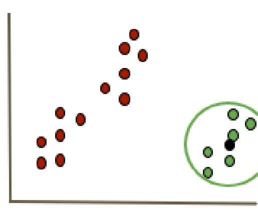
VS



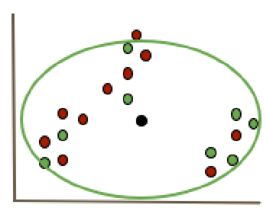
i) Clustering on left has smaller intra-cluster distances than right

ii) $$\sum_{k}^{K} \sum_{x_i, x_j \in C_k} d(x_i, x_j)$$ is smaller for the one the left

b) Given distance function **d**, we can find centroids(center of mass) for each cluster, not necessarily part of dataset that are at center of each cluster



VS



i) When d is Euclidean, centroid of m points $\{x_1, \ldots, x_2\}$ is mean/average of points

$$\sum_{k}^{K} \sum_{x_i, x_j \in C_k} d(x_i, x_j)^2 = \sum_{k}^{K} |C_k| \sum_{x_i \in C_k} d(x_i, \mu_k)^2$$

6) K-means

Given X = {x_1, …, x_n}(dataset) and k, find k points {μ_1, …, μ_k} that minimizes cost function

$$\sum_{i}^{k} \sum_{x \in C_i} d(x, \mu_i)$$

- When k = 1 and k = n this is easy
- When x_i lives in more than 2D, this is very difficult(NP-hard) problem

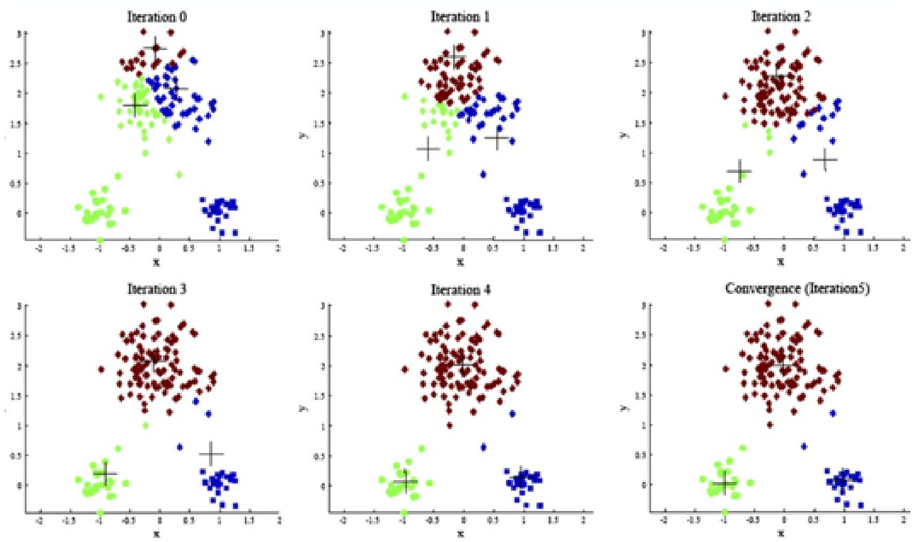
a) Lloyd's Algorithm

   i) Randomly pick k centers {μ_1, …, μ_k}

   ii) Assign each point in dataset to closest center

   iii) Compute new centers as means of each cluster

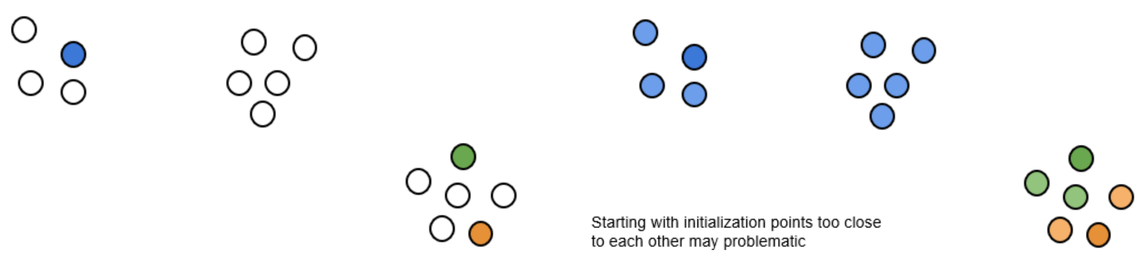
   iv) Repeat 2 and 3 until convergence



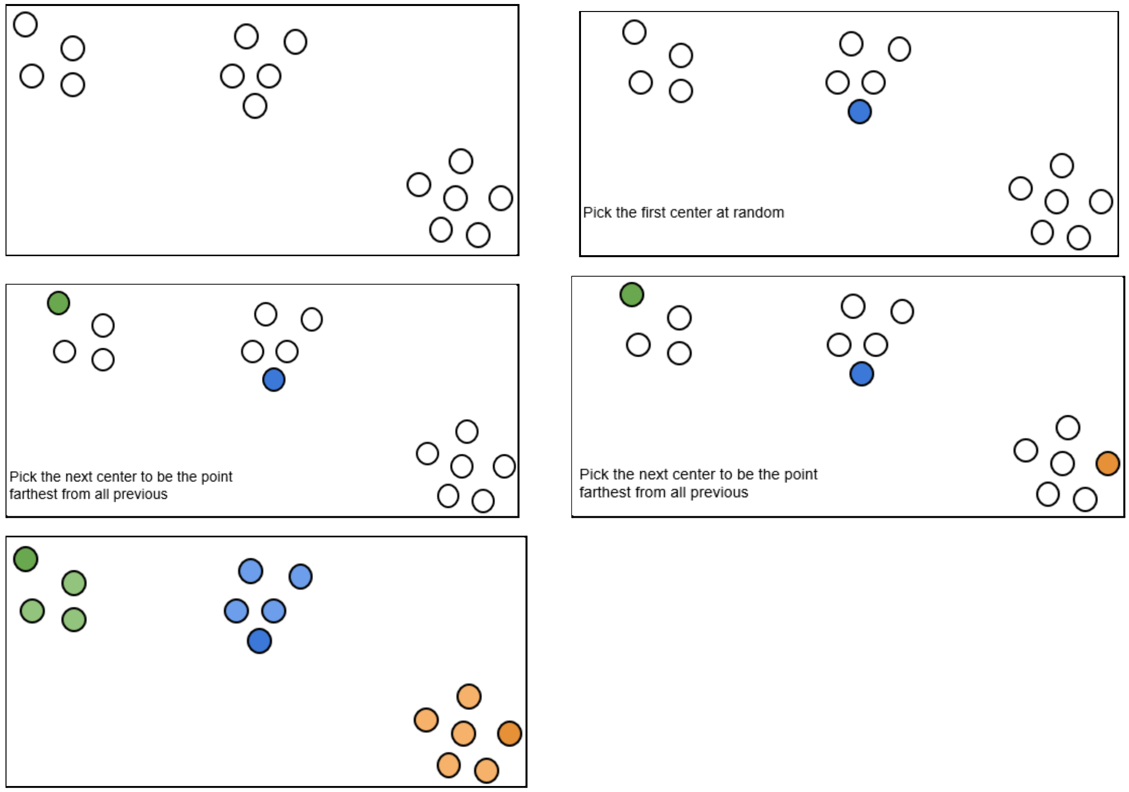
- Algorithm will always converge
- Choice of initial points has a large influence on resulting clustering
- One solution: Run Lloyd's algorithm multiple times and choose result with lowest cost
  - Can still lead to bad results because of randomness.

b) Initialization: try different initialization methods as a solution to above

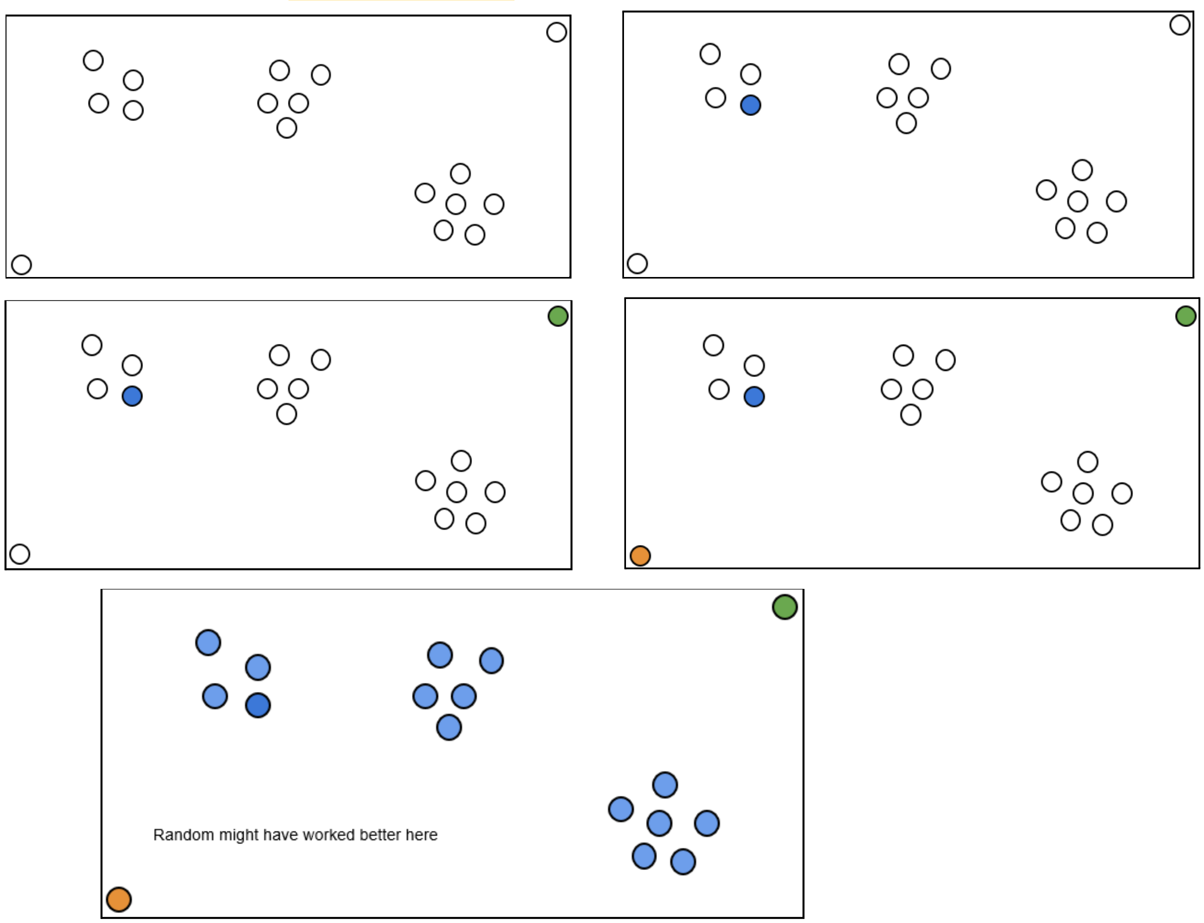
   i) Random



ii) Farthest First Traversal



iii) FFT and outliers

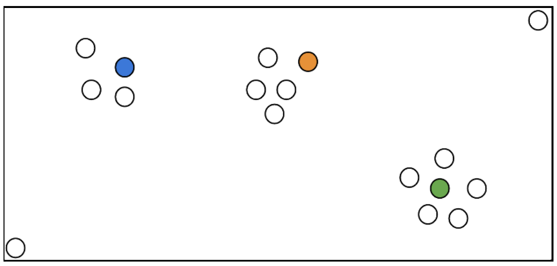


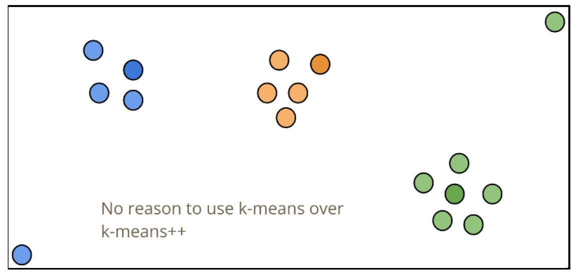
c) K-means++

i) Initialize with combination of 2 methods:

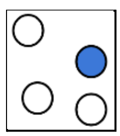
1) Start with random center
2) Let **D(x)** be distance between **x** and centers selected so far. Choose next center with probability proportion to $D(x)^a$

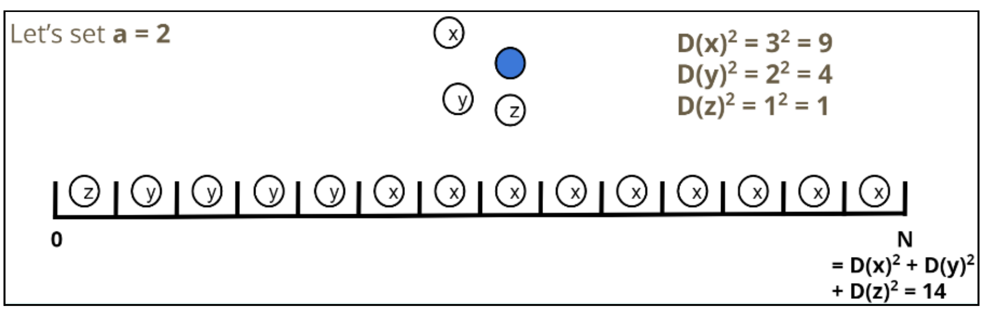
- When:
  - a = 0: random initialization(all points have equal probability)
  - a = ∞: farthest first travel
  - a = 2: K-means++





ii) Suppose we are given a black box that will generate a uniform random number between 0 and any N. How can we use black box to select points with probability proportional to $D(x)^2$?

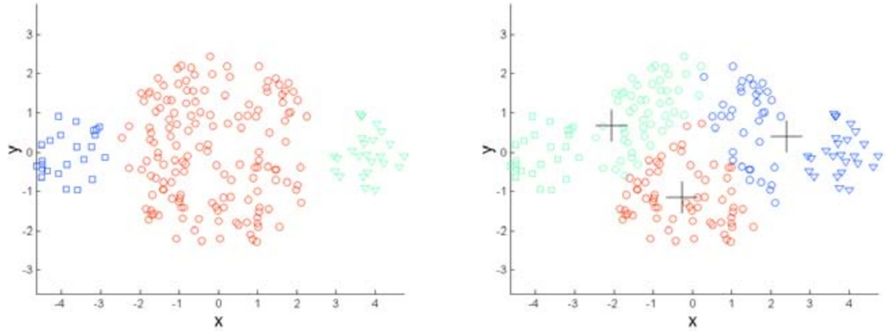




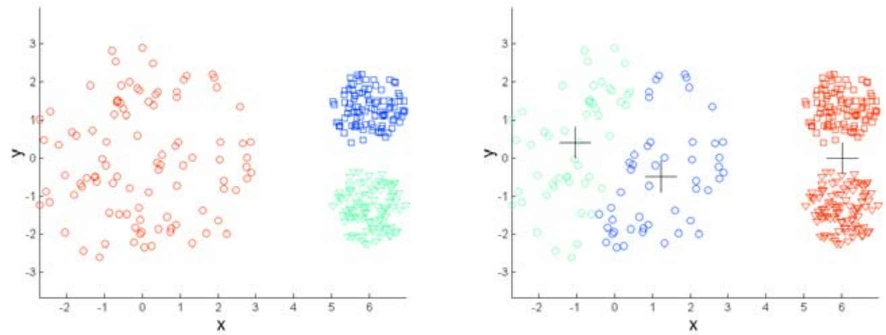
- Using black box, we can generate a number between 0 and N to determine which point to pick next. It will be chosen with probability proportional to $D(x)^2$

d) Limitations

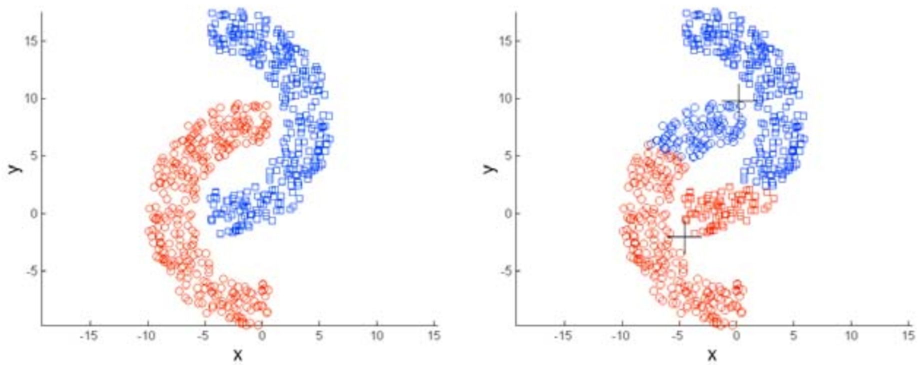
i)



ii)



iii)



e) How to choose the right k?

i) Iterate through different values of k (elbow method)

ii) Use empirical/domain-specific knowledge

   1) Example: Is there a known approximate distribution of the data? (K-means is good for spherical gaussians)

iii) Silhouette scores

f) Variations

i) K-medians: uses L1 norm/manhattan distance

ii) K-medoids: any distance function + the centers must be in the dataset

iii) Weighted K-means: each point has a different weight when computing the mean