Clustering 1

EE412: Foundation of Big Data Analytics

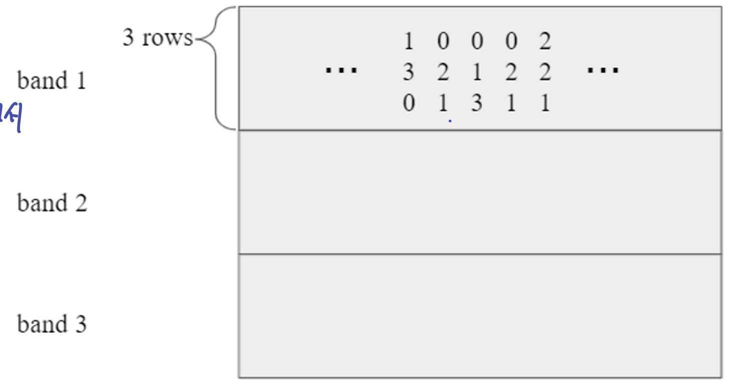
Announcements

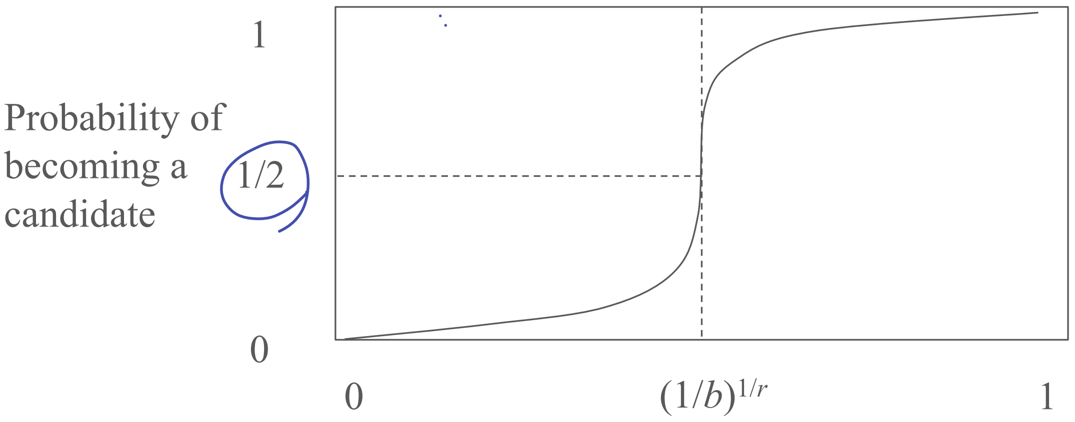
- Homeworks
    - HW0 (due: 09/21)
    - HW1 (due: 10/05)

Recap: LSH

collision => don't want => 다른 doc이 한 포트로 갈 수도 있어서

- Locality Sensitive Hashing (LSH)
- Theory of LSH
  - Locality-sensitive families
  - AND and OR constructions
- LSH for cosine distance
  - Random hyperplanes





Outline

1. **Clustering**
2. Hierarchical Clustering
3. K-means Clustering

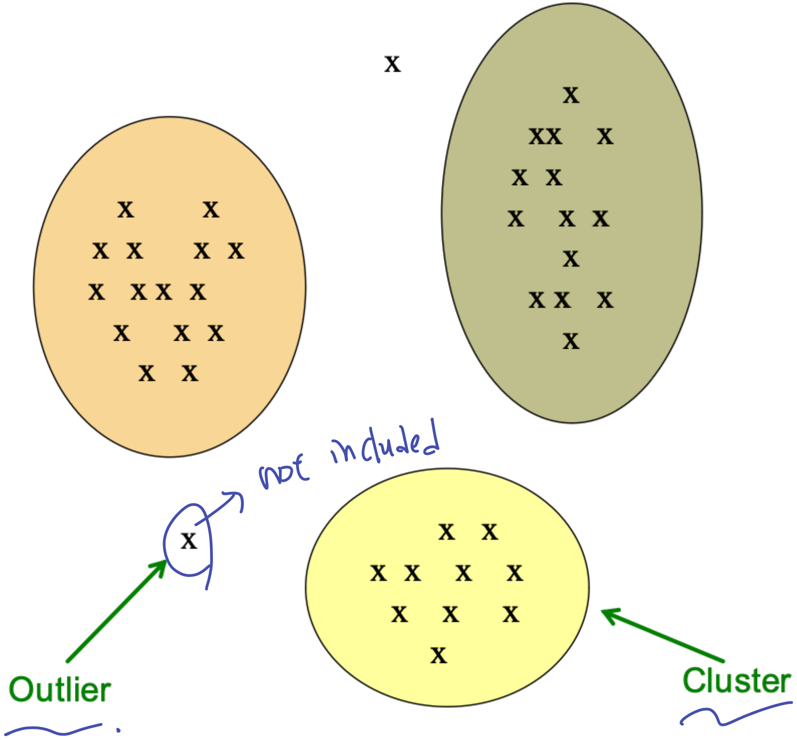
Clustering

- The process of grouping a collection of points into clusters
  - Members of the same cluster are close/similar to each other
  - Members of different clusters are dissimilar
- We focus on cases when
  - Data is very large
  - Space is high-dimensional
  - Space is non-Euclidean
  - Similarity is defined using any distance measure

Points, Spaces, and Distances

- **Distance measure** gives a distance between two points
  - Recall the 4 requirements for a distance measure
- Popular distance measures in a Euclidean space
  - Euclidean (L2) distance: $\sqrt{\sum_i (x_i - y_i)^2}$
  - Manhattan (L1) distance: $\sum_i |x_i - y_i|$
  - L0 distance: $\sum_i [1 \text{ if } x_i \neq x_y \text{ else } 0]$
  - L-Infinity distance: $\max_i |x_i - y_i|$

Example: Clusters and Outliers



Source: Stanford CS246 (2022)

Example: Documents

- **Finding topics:**
  - set of shingling
  - Represent a document by a vector $(x_1, x_2, \cdots, x_k)$
    where $x_i = 1$ iff the i-th word appears in the document
  - **Idea:** Documents with similar sets of words may be about the same topic
  - **Task:** Find clusters of documents
    - Slightly different from finding similar pairs (by LSH)

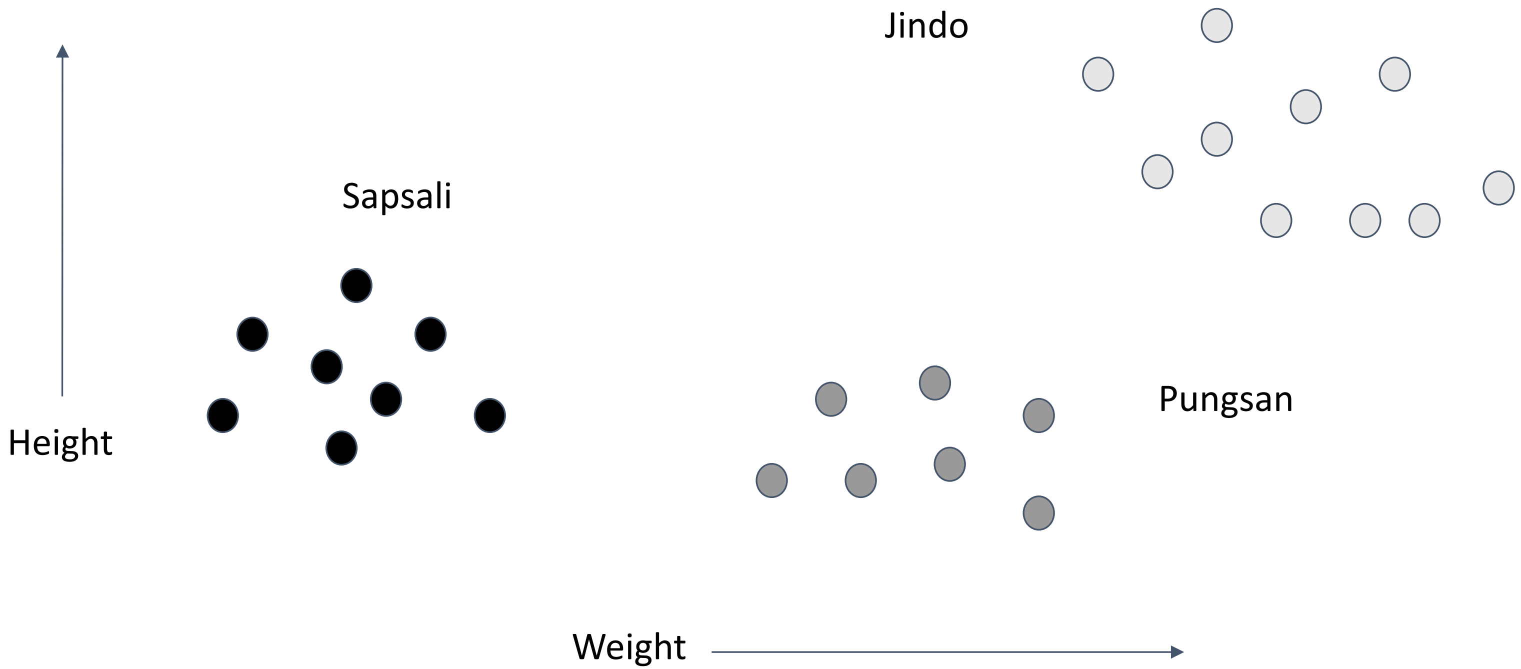
↳ Goal: good clusters, not pair

Cosine, Jaccard, and Euclidean

- We have a choice when we think of documents as sets of words:
  - **Sets as vectors:** Measure similarity by the cosine distance
  - **Sets as sets:** Measure similarity by the Jaccard distance
  - **Sets as points:** Measure similarity by Euclidean distance

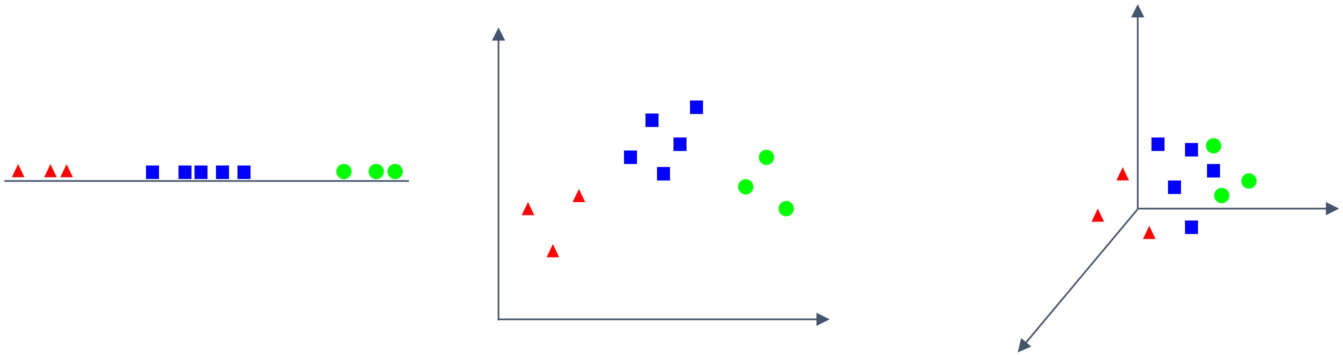
Clustering in Low-Dimensional Space

- Clustering in two dimensions looks easy



The Curse of Dimensionality

- High-dimensional spaces have the "curse of dimensionality"
  - Almost all pairs of points are equally far away from one another



The Curse of Dimensionality

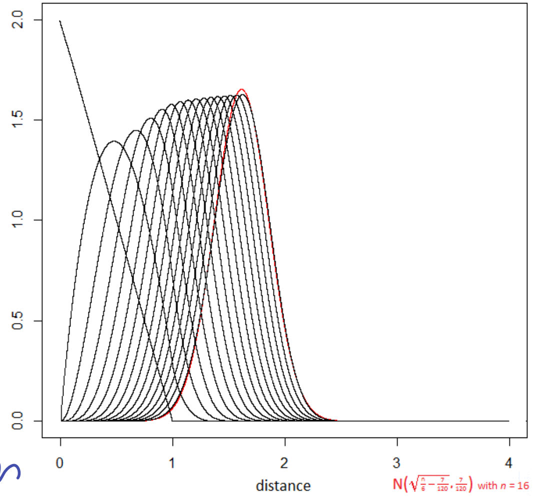
always size = 1

- Consider two random points in the n-dimensional unit hypercube
- The variance of their distance goes to a constant when $n \to \infty$
  - But the possible range is $[0, \sqrt{n})$
- E.g., when $n = 2500$
  - The distance can be anything in $[0, 50)$
  - However, most distances are in $(20, 21)$

→ although average ↑ variance is not // why curse of dim

→ good cluster is hard in high-dim

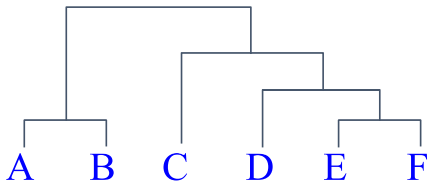
Densities of distances between random points in a unit hypercube

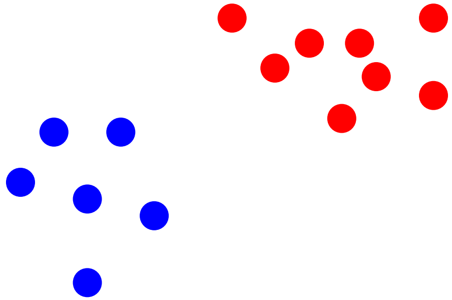


Source: https://math.stackexchange.com/questions/1976842/how-is-the-distance-of-two-random-points-in-a-unit-hypercube-distributed

Clustering Strategies (1)

- **Hierarchical**
  - **Agglomerative** (bottom up)
    - Initially, each point is a cluster
    - Repeatedly combine the "nearest" clusters into one
  - **Divisive** (top down)
    - Start with one cluster and recursively split it
- **Point assignment**
  - Maintain a set of clusters
  - Points belong to the "nearest" cluster





Clustering Strategies (2)

- **In Euclidean:**
  - Points are vectors of real numbers, i.e. coordinates
  - It is possible to summarize a collection of points as their average
    - We call it centroid
  - Distance measure: L2 norm, L1 norm
- **In non-Euclidean:**
  - There is no notion of location, and centroid
  - We summarize a collection of points differently
  - Distance measures: Jaccard, Hamming, cosine

Clustering Strategies (3)

- **Q:** Does the data fit in memory or it resides on disk?
- **In-memory clustering** is more straightforward
  - Example: K-means
- **Large-data clustering** requires loading one batch of data at a time
  - Cluster them in memory and keep summaries of clusters
  - Example: BFR, CURE

Hierarchical vs. Point-Assignment

- (Left) Point assignment is good when clusters have nice shapes
- (Right) Hierarchical can win when shapes are weird
  - Note both clusters have essentially the same centroid

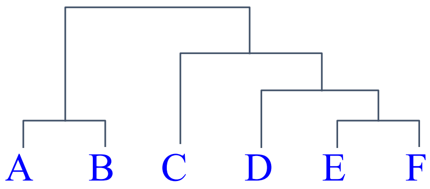
Source: Stanford CS246 (2022)

Outline

1. Clustering
2. **<u>Hierarchical Clustering</u>**
3. K-means Clustering

Hierarchical Clustering

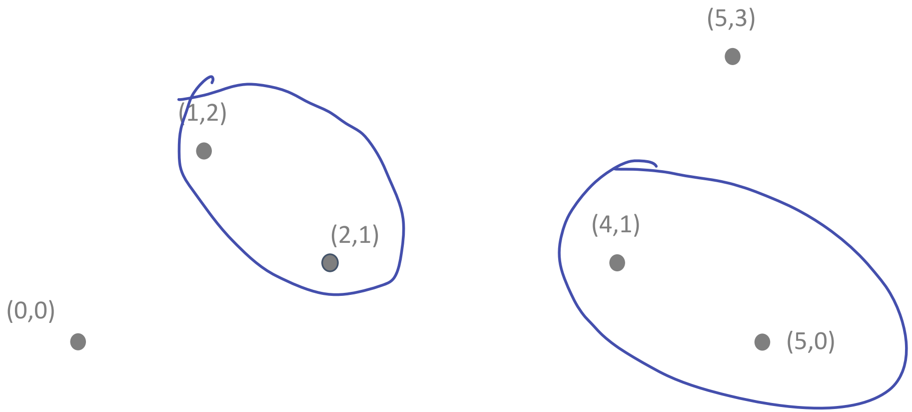
- **Key operation:** Repeatedly merge two "nearest" clusters
- **Three important questions:**
  1. How to represent a cluster?
  2. How to determine the nearness of clusters?
  3. When to stop merging clusters?



Hierarchical Clustering: Euclidean

- In a Euclidean case:
- **Q1:** How to represent a cluster of many points?
  - As we merge clusters, we represent the "location" of each cluster by
    - Its centroid = average of its (data) points
- **Q2:** How to determine the nearness of clusters?
  - Measure cluster distances by distances of centroids
  - Merge two clusters with the shortest distance

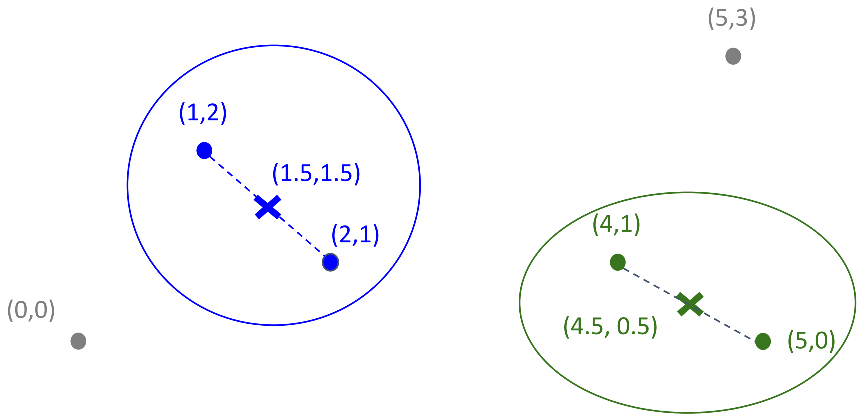
Example: Hierarchical Clustering



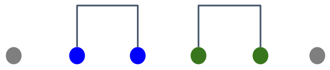
Data:
- = data point
- ✕ = centroid

Dendogram

Example: Hierarchical Clustering

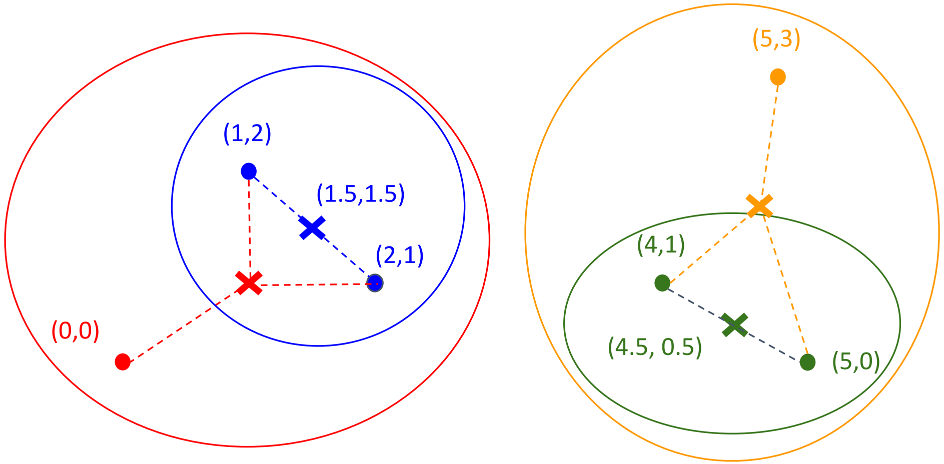


Data:
● = data point
✕ = centroid

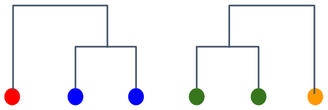


Dendogram

Example: Hierarchical Clustering

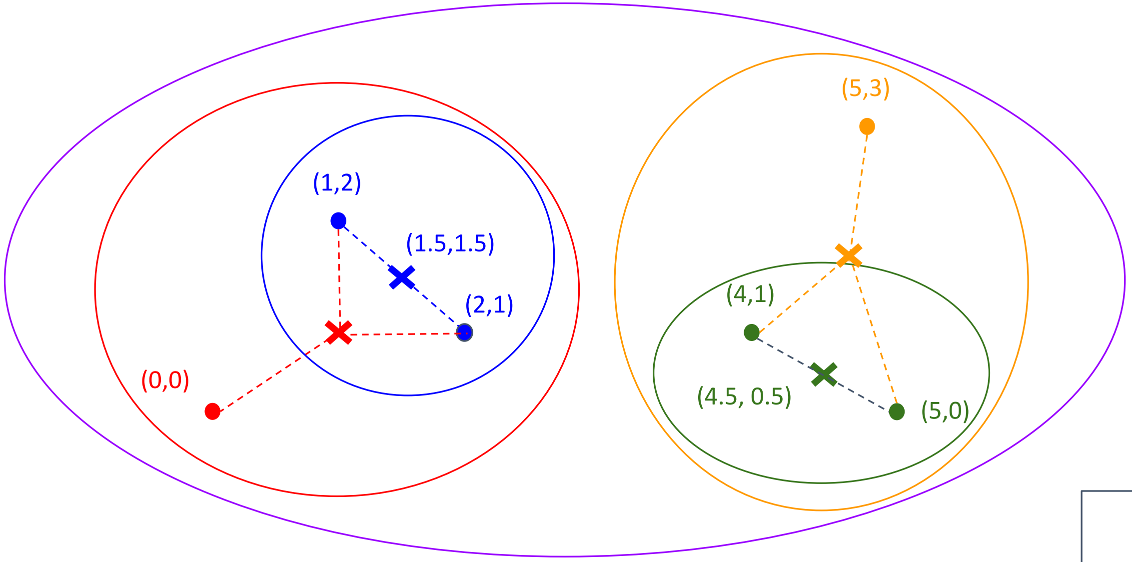


Data:
- ● = data point
- ✕ = centroid

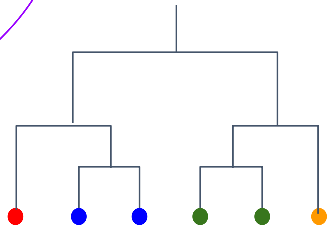


Dendogram

Example: Hierarchical Clustering



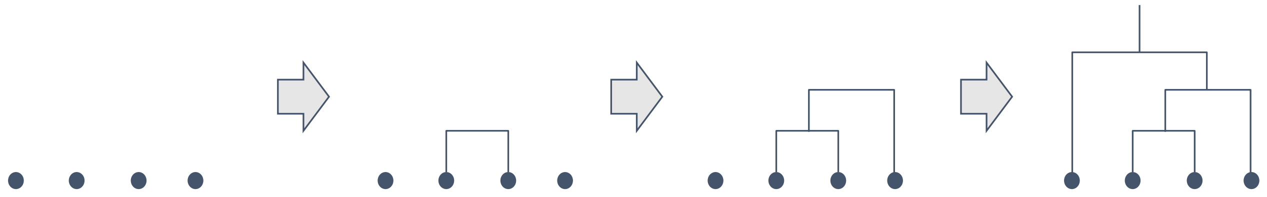
Data:
● = data point
✕ = centroid



Dendogram

Efficiency of Hierarchical Clustering

- Basic algorithm
  - At each step, compute distances between pairs of clusters
  - Running time is $O(n^3)$, since $\underbrace{n^2 + (n-1)^2 + (n-2)^2}$ $+ \cdots$?? 왜지 //



n^2 $(n-1)^2$

Efficiency of Hierarchical Clustering

- More efficient implementation is possible using a **priority queue**
  - **Priority queue** can store a collection of prioritized elements
  - The overall algorithm is $O(n^2 \log n)$, which is better than $O(n^3)$
- Why is it better?
  - We can avoid re-computing the distances computed in earlier iterations
  - See the textbook for details

Hierarchical Clustering: Non-Euclidean

- In a non-Euclidean case: $\boxed{\text{not think about centroids}}$
  - The only "locations" we can talk about are the points themselves
- **Three possible approaches** regarding the questions:
  - **Q1:** How to represent a cluster of many points?
  - **Q2:** How to determine the nearness of clusters?
- **Note:** They can also be used in a Euclidean case

Non-Euclidean Approach 1

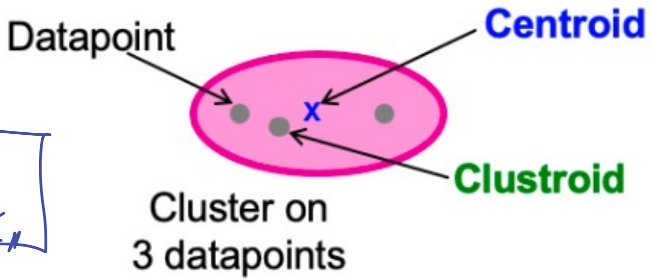
- **Q1:** How to represent a cluster of many points?
  - **Pick a clustroid = (data) point "closest" to all other points**
  - Possible meanings of "closest":
    - Smallest maximum / average distance to other points
- **Q2:** How to determine the nearness of clusters?
  - Treat clustroid as if it were centroid

Centroid vs. Clusteroid

- **Centroid** is the average of all (data) points in the cluster
  - This means centroid is an "artificial" point
- **Clustroid** is an **existing point** that is "closest" to all other points

actual data point

new generated point is quite weird for som space



Source: Stanford CS246 (2022)

Non-Euclidean Approach 2

- **Q1:** How to represent a cluster of many points?
  - As the collection of points in a cluster
- **Q2:** How to determine the nearness of clusters?
  - **Define inter-cluster distance:**
    - Minimum of the distances between any two points, one from each cluster
    - Average distance of all pairs of points, one from each cluster

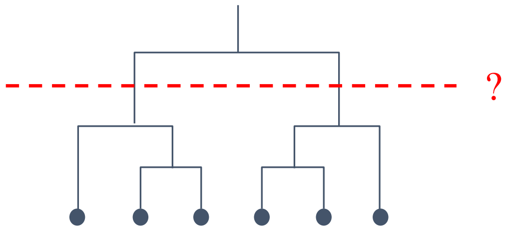
Non-Euclidean Approach 3

- **Q1:** How to represent a cluster of many points?
  - As the collection of points in a cluster
- **Q2:** How to determine the nearness of clusters?
  - **Merge clusters whose union is most cohesive** (cohision)
  - Possible notions of cohesion (the smaller, the more cohesive):
    - Diameter of the merged cluster = Maximum distance between points
    - Average distance between all points
    - Inverse density of the cluster = Diameter or average distance / # of points

how good the clustered after

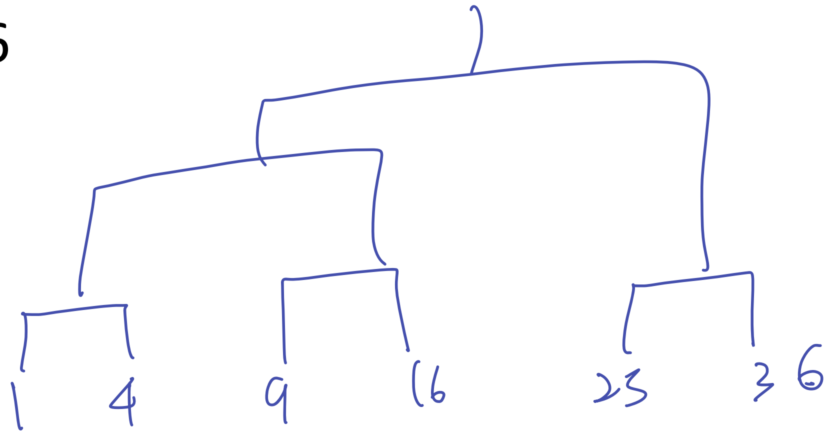
When to Stop?

- When do we stop merging clusters?
  - If the diameter of merged cluster exceeds threshold
  - If the density of merged cluster is below threshold
  - If there is evidence that the merged cluster yields a bad cluster
    - E.g., average diameter suddenly increases



Pop Quiz

- Perform a hierarchical clustering on the points below
  - The distance between two clusters is
    the maximum of the distances between any two points
- 1, 4, 9, 16, 25, 36



Outline

1. Clustering
2. Hierarchical Clustering
3. **<u>K-means Clustering</u>**

k-means Algorithm

- Best known family of clustering algorithms for point assignment
- Given Euclidean space/distance and k = number of clusters
  - Possible to deduce k by trial and error
- Find cluster centers that minimizes sum of squared distances
  - From each point to its cluster center

k-means Algorithm

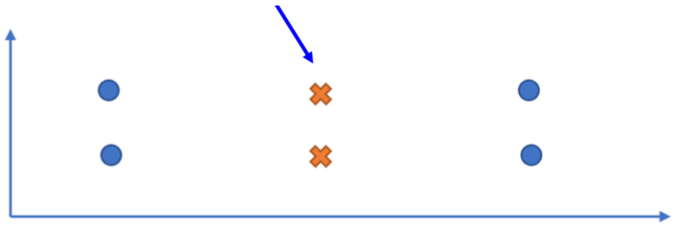
- **Initialize** clusters by picking k centers
- **Until convergence:**
  1. For each point, assign it to the cluster whose centroid is the closest
  2. After all points are assigned, update the centroids of the k clusters
     - As the average of data points within each cluster
- **Convergence** means that points don't move between clusters

Shortcoming of k-means

초기 centroid에 의존

- Convergence of k-means heavily depends on the initial centroids
- It can perform arbitrarily badly in such a case:

X guarantee good results



Source: Stanford CS246 (2022)

k-means++

- **Goal:** Pick points that are likely to lie in different clusters
- **Basic idea:**

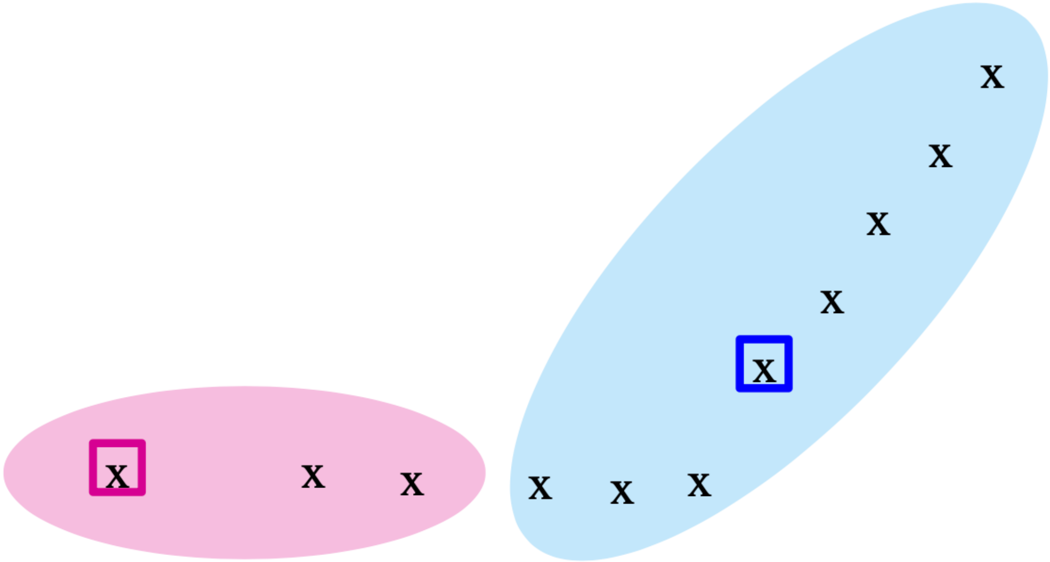
  hiarachial clustering => 초기값에 영향을 받지 않음

  - Pick a small sample of points S, cluster them, and use the centroids
    - Any algorithm can be used for the clustering
  - Sample size $|S|$ is proportional to $k \times \log n$, where n is the data size
- **How to pick sample points:**
  - Visit points in a random order x ideal
  - Add each point p to the sample with a probability proportional to $D(p)^2$.
    - $D(p)$ = Distance between p and the nearest already picked point

current point

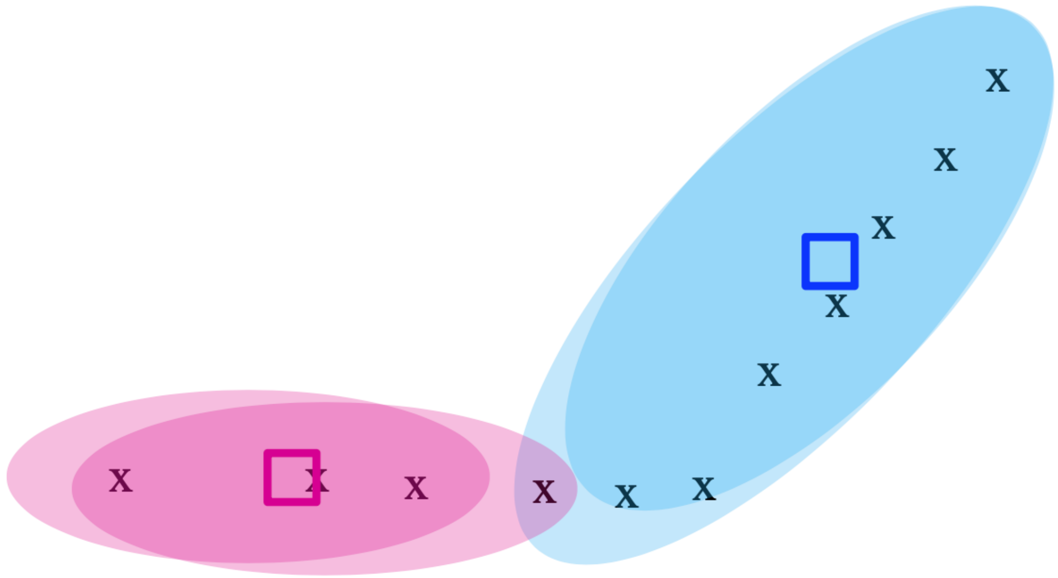
consist far points (sample covers actual space) => good initial points

Example: Assigning Clusters



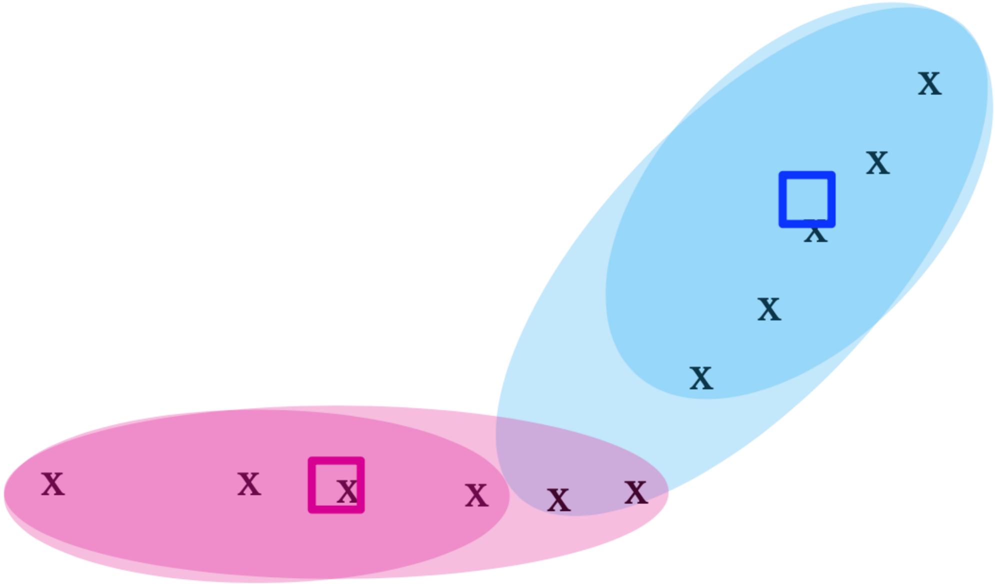
Source: Stanford CS246 (2022)

Example: Assigning Clusters



Source: Stanford CS246 (2022)

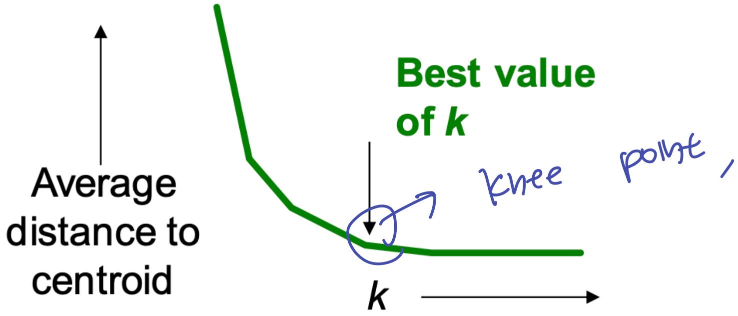
Example: Assigning Clusters



Source: Stanford CS246 (2022)

Getting the k Right

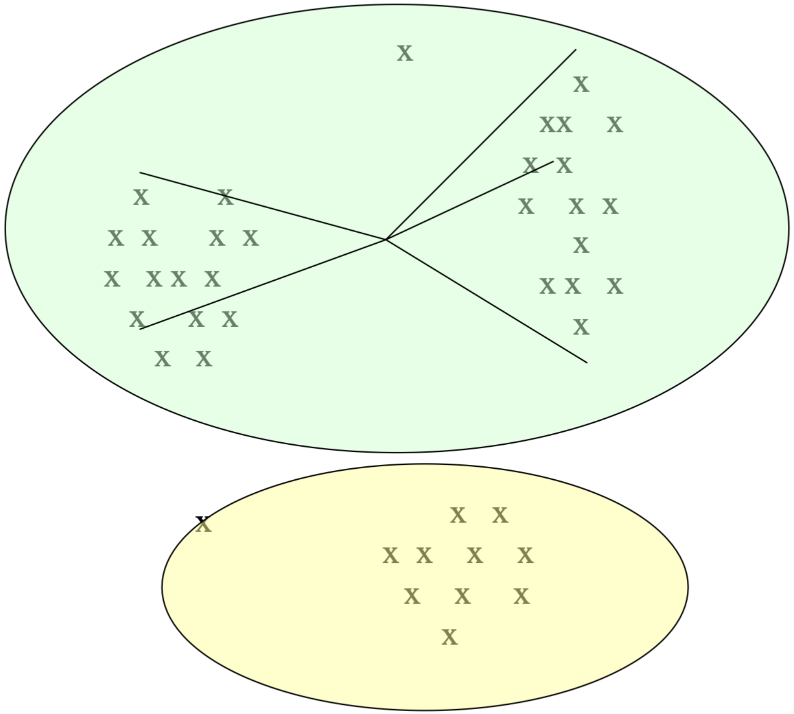
- **How to select k?**
  - Try different k
  - Look at the change in the average distance to centroid as k increases
  - Average falls rapidly until right k, then changes little



Source: Stanford CS246 (2022)

Example: Picking k

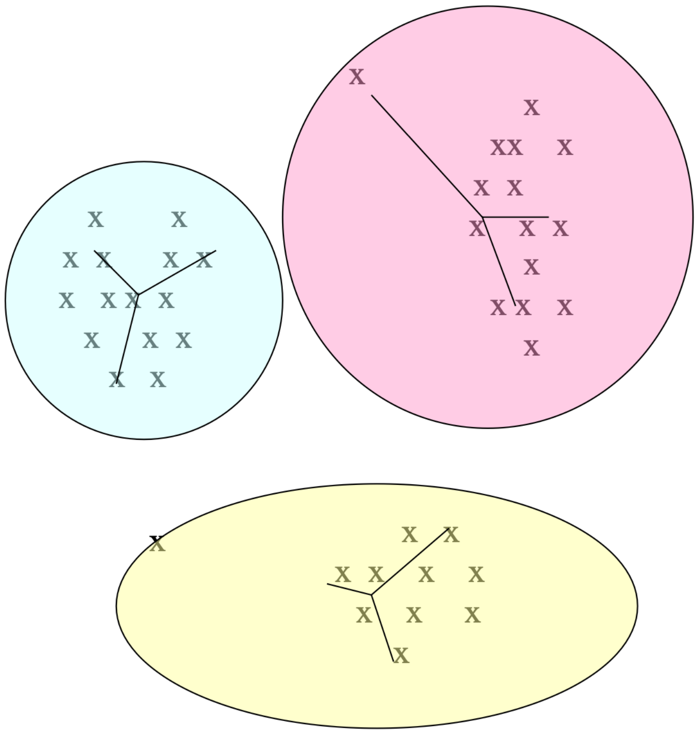
Too few;
many long distances to centroid



Source: Stanford CS246 (2022)

Example: Picking k

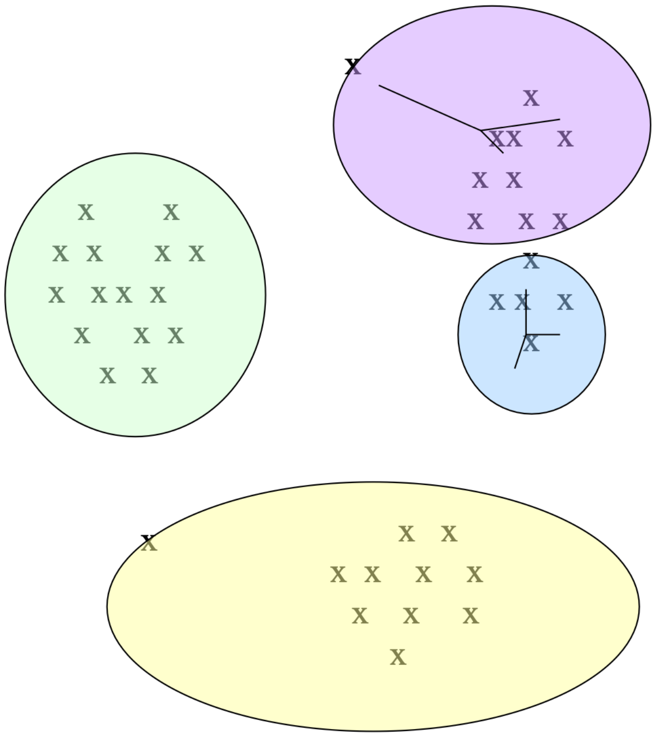
Just right;
distances
rather short



Source: Stanford CS246 (2022)

Example: Picking k

Too many;
little improvement
in average
distance



Source: Stanford CS246 (2022)

Summary

1. **<u>Clustering</u>**
    - Curse of dimensionality
    - Clustering strategies
2. Hierarchical Clustering
    - Euclidean vs. non-Euclidean
    - Centroids vs. clustroids
3. K-means Clustering
    - k-means++
    - Selection of k