

Note to other teachers and users of these slides: We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>

Clustering

Mining of Massive Datasets
Jure Leskovec, Anand Rajaraman, Jeff Ullman
Stanford University

<http://www.mmds.org>



High Dimensional Data

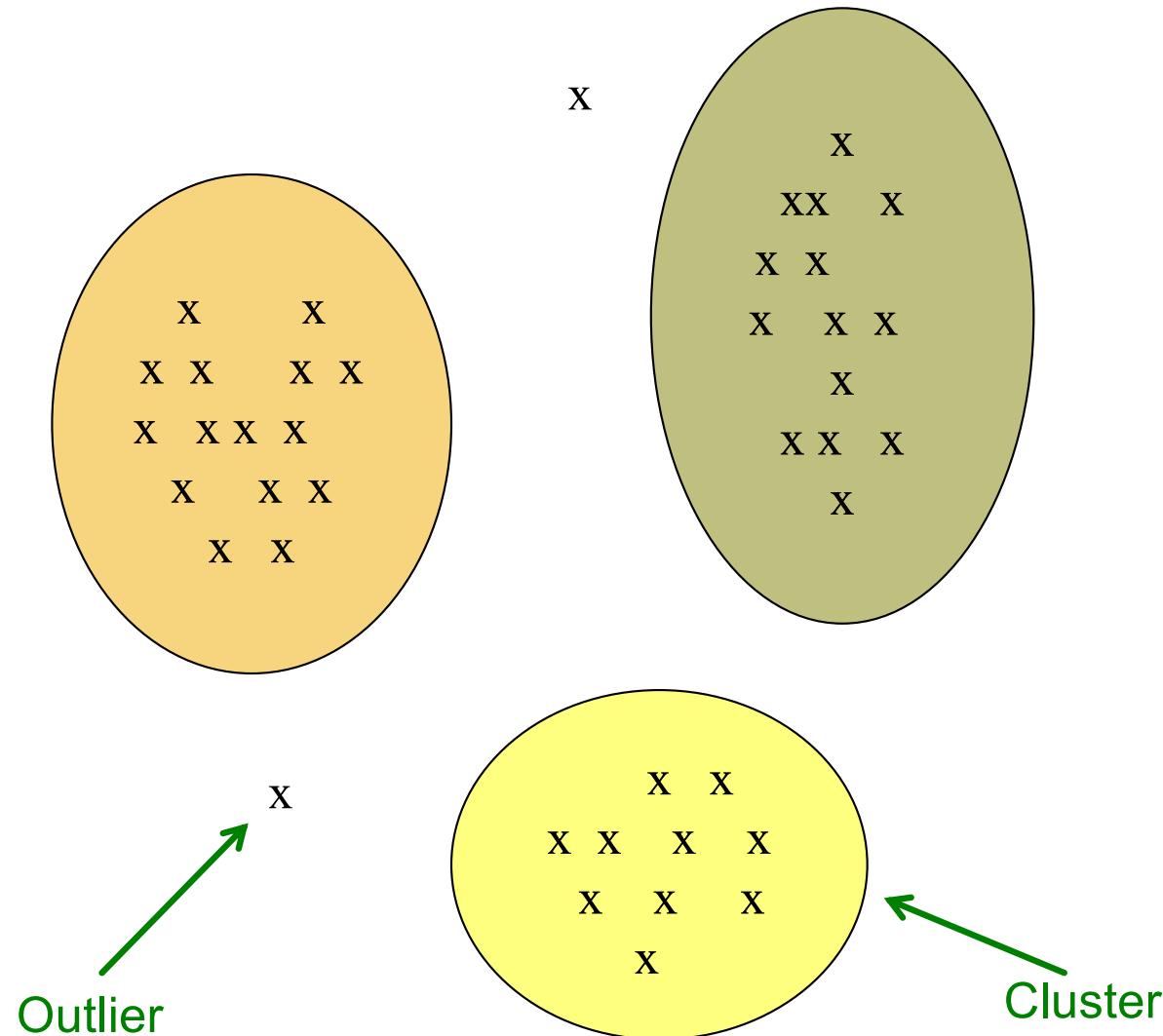
- Given a cloud of data points we want to understand its structure



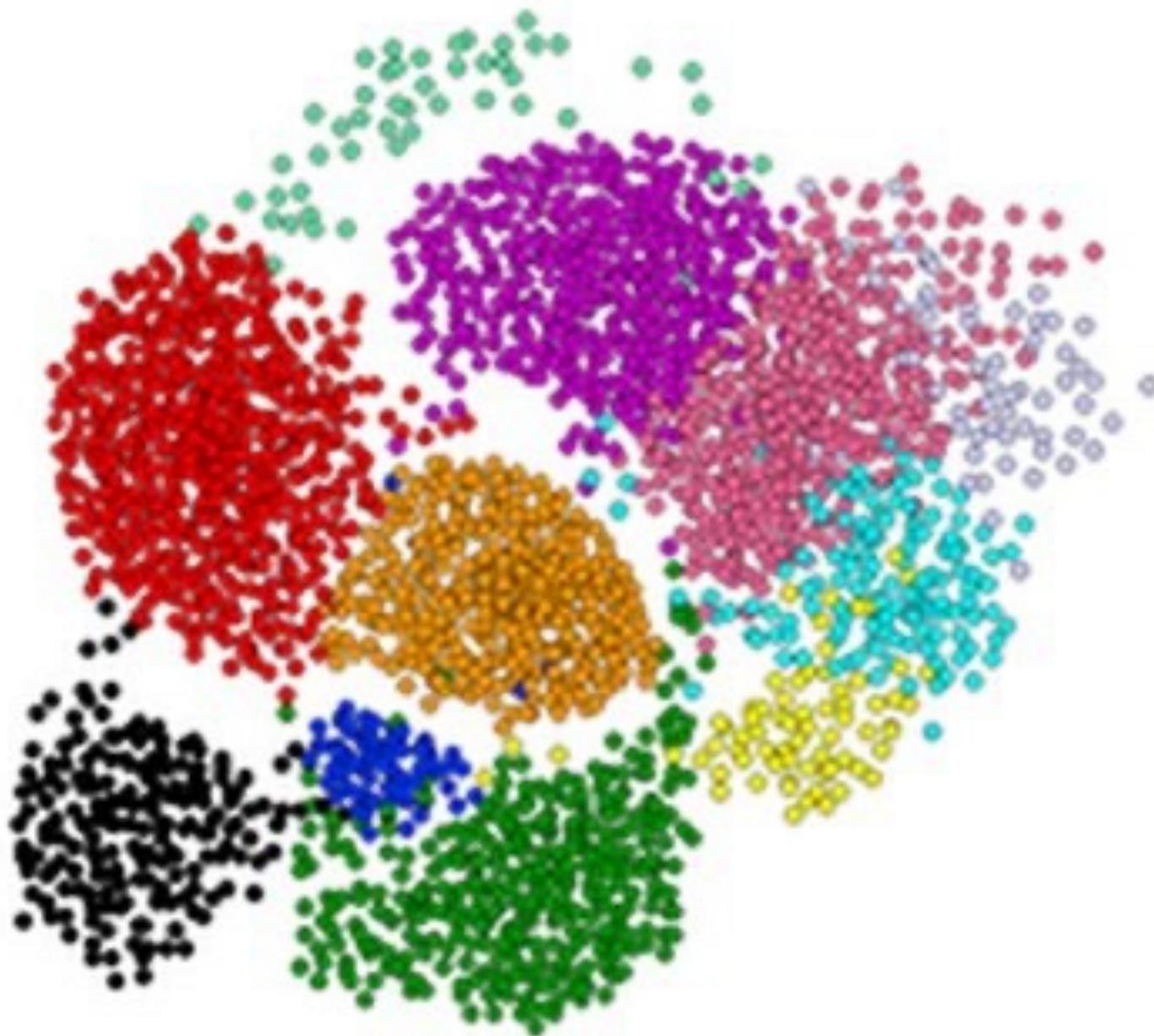
The Problem of Clustering

- Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***, so that
 - Members of a cluster are close/similar to each other
 - Members of different clusters are dissimilar
- **Usually:**
 - Points are in a high-dimensional space
 - Similarity is defined using a distance measure
 - Euclidean, Cosine, Jaccard, edit distance, ...

Example: Clusters & Outliers



Clustering is a hard problem!

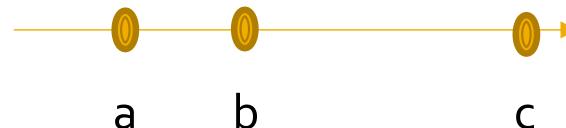


Why is it hard?

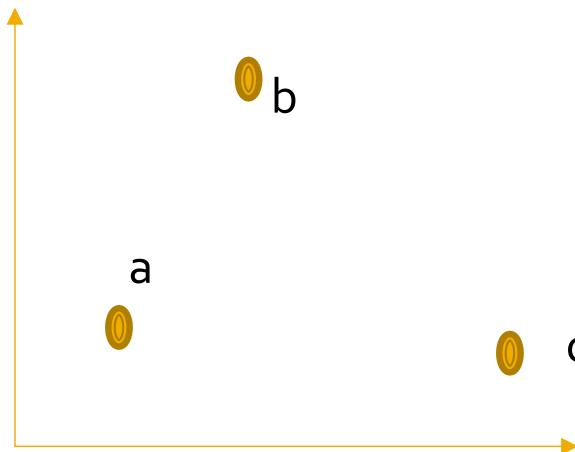
- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- Many applications involve not 2, but 10 or 10,000 dimensions
- **High-dimensional spaces look different:**
Almost all pairs of points are at about the same distance

High Dimension: Euclidean

- Consider a set of data points on a line
 - $\text{dist}(a, b) < \text{dist}(a, c)$

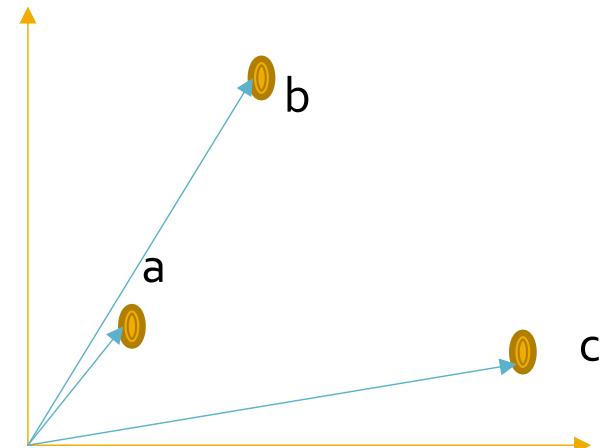


- Consider increasing the dimension by 1
 - $\text{dist}(a, b) \sim \text{dist}(a, c)$



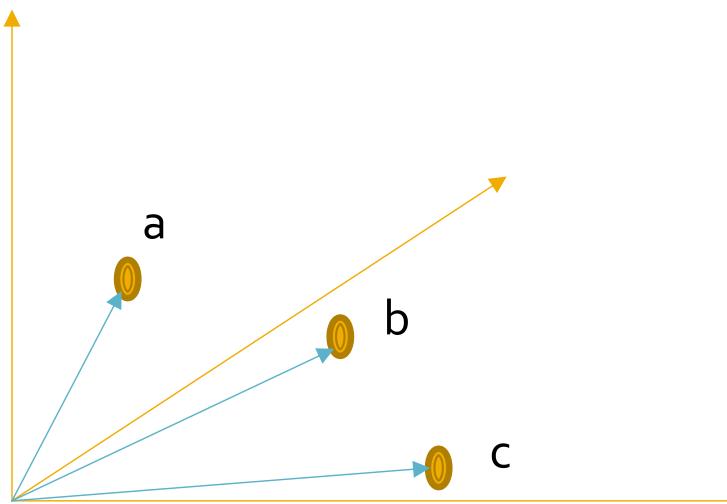
High Dimension: Cosine

- $\text{Cosine}(a, b) > \text{Cosine}(a, c)$



- Increase d to 3
 - $\text{Cosine}(a, b) \sim \text{Cosine}(a, c)$

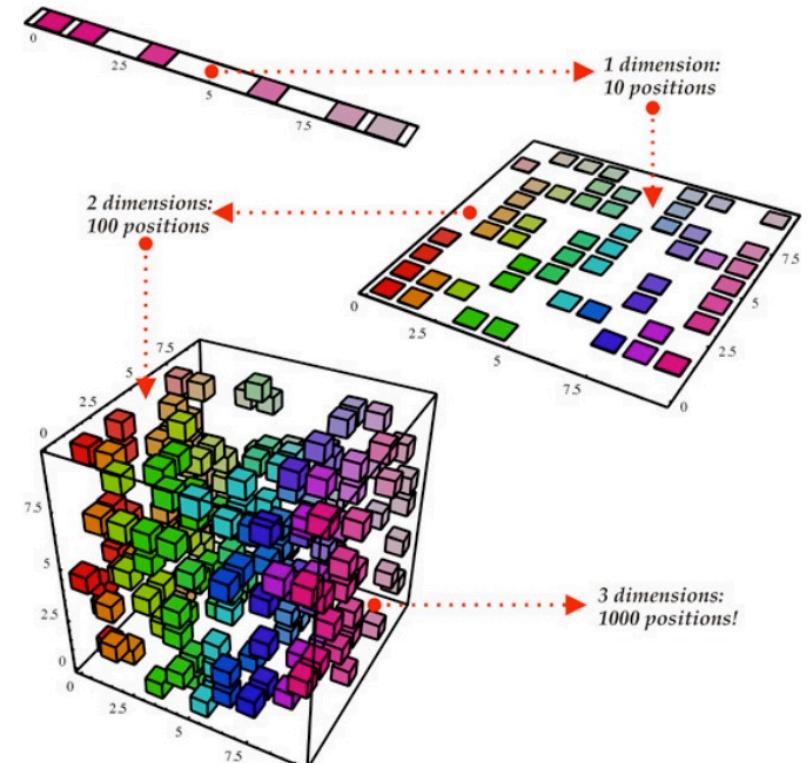
- Higher d
 - Angle $\rightarrow 90^\circ$
 - Cosine $\rightarrow 0$



Curse of Dimensionality

- Data points have similar distance btw each other
 - Euclidean distance breaks
 - almost all pairs of points are equally far away from one another

- Data vectors become orthogonal
 - Cosine function breaks
 - almost any two vectors are orthogonal



<https://bigsnarf.wordpress.com/2013/06/14/curse-of-dimensionality/>

Clustering Problem: Music CDs

- **Intuitively:** Music divides into categories, and customers prefer a few categories
 - But what are categories really?
- Represent a CD by a set of customers who bought it:
- Similar CDs have similar sets of customers, and vice-versa

Clustering Problem: Music CDs

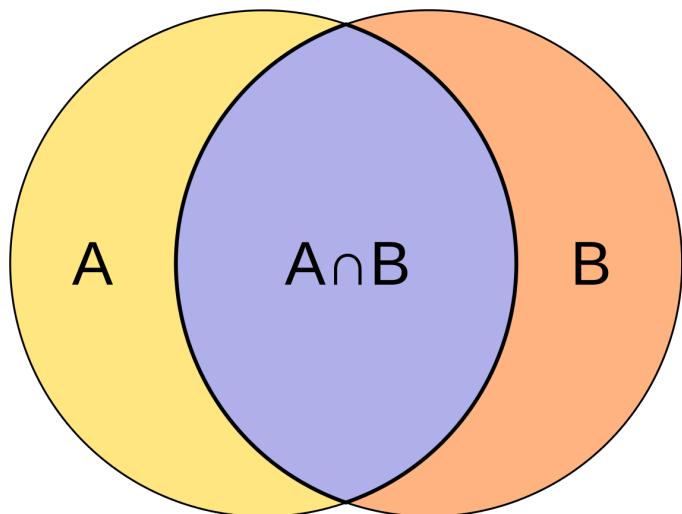
Space of all CDs:

- Think of a space with one dim. for each customer
 - Values in a dimension may be 0 or 1 only
 - A CD is a point in this space (x_1, x_2, \dots, x_k) , where $x_i = 1$ iff the i^{th} customer bought the CD
- For Amazon, the dimension is tens of millions
- **Task:** Find clusters of similar CDs

Cosine, Jaccard, and Euclidean

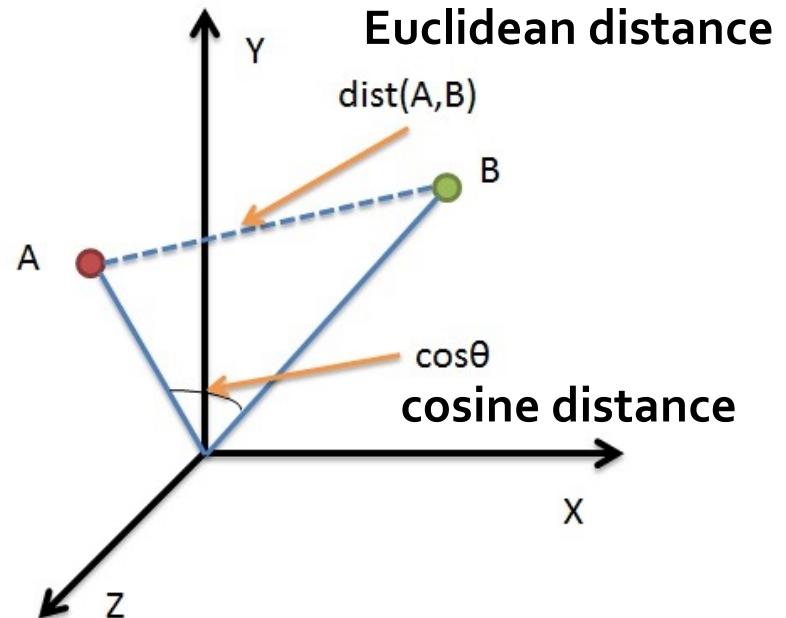
- As with CDs we have a choice when we think of documents as sets of words or shingles:
 - as vectors: Measure similarity by the cosine distance
 - as sets: Measure similarity by the Jaccard distance
 - as points: Measure similarity by Euclidean distance

Measure similarity

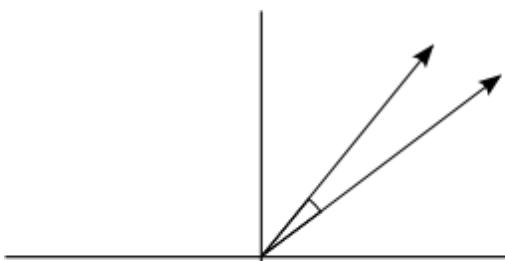


Jaccard distance

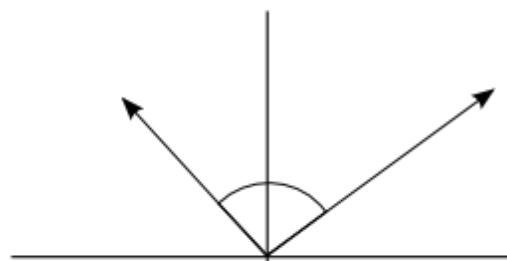
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$



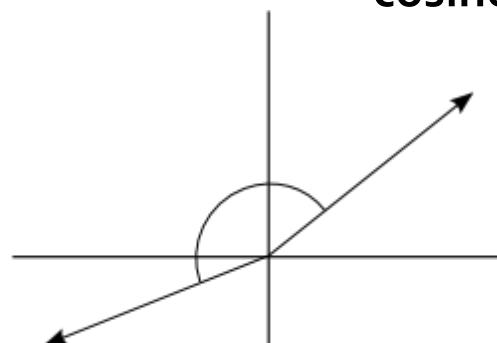
cosine distance



Similar scores
Score Vectors in same direction
Angle between them is near 0 deg.
Cosine of angle is near 1 i.e. 100%



Unrelated scores
Score Vectors are nearly orthogonal
Angle between them is near 90 deg.
Cosine of angle is near 0 i.e. 0%



Opposite scores
Score Vectors in opposite direction
Angle between them is near 180 deg.
Cosine of angle is near -1 i.e. -100%

Overview: Methods of Clustering

■ Hierarchical:

■ Agglomerative (bottom up):

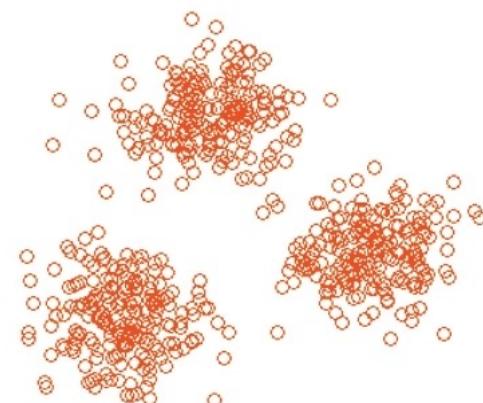
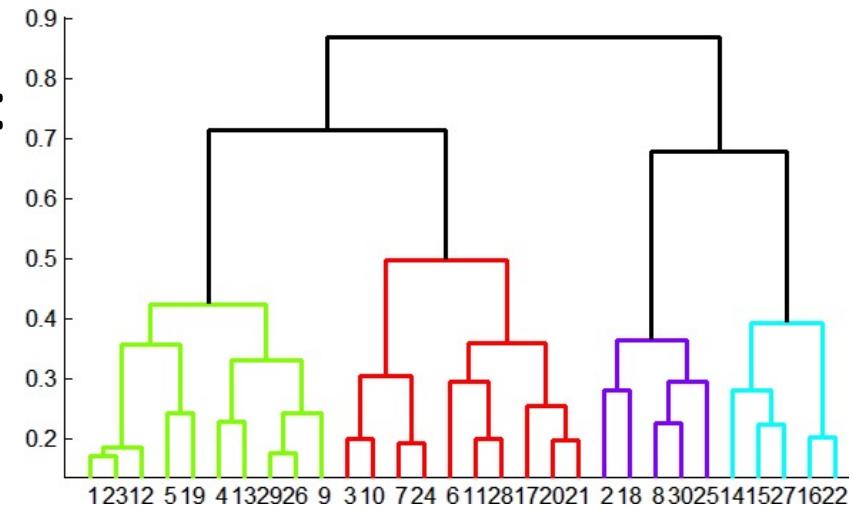
- Initially, each point is a cluster
- Repeatedly combine the two “nearest” clusters into one

■ Divisive (top down):

- Start with one cluster and recursively split it

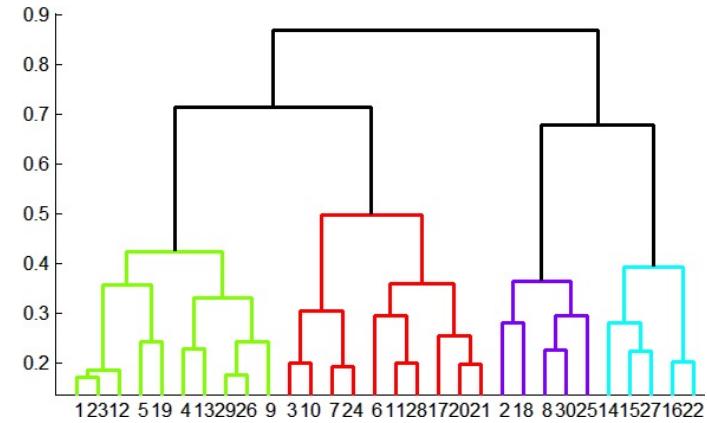
■ Point assignment:

- Maintain a set of clusters
- Points belong to “nearest” cluster



Hierarchical Clustering

- **Key operation:**
Repeatedly combine two nearest clusters

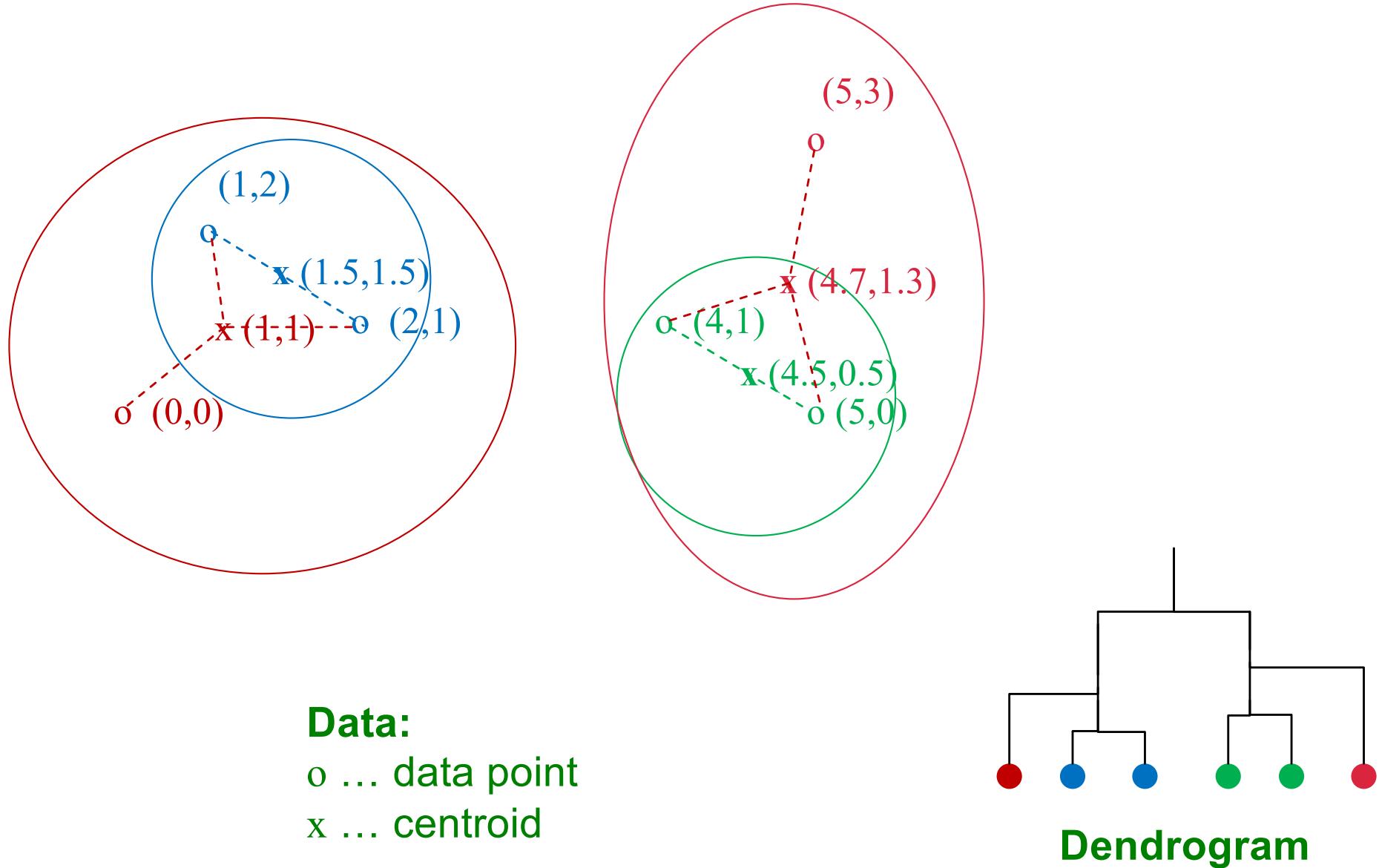


- **Three important questions:**
 - 1) How do you represent a cluster **of more than one point?**
 - 2) How do you determine the “**nearness**” of clusters?
 - 3) When to **stop combining clusters?**

Hierarchical Clustering

- **Key operation:** Repeatedly combine two nearest clusters
- **(1) How to represent a cluster of many points?**
 - **Euclidean case:** each cluster has a *centroid* = average of its (data)points
- **(2) How to determine “nearness” of clusters?**
 - Measure cluster distances by distances of centroids

Example: Hierarchical clustering



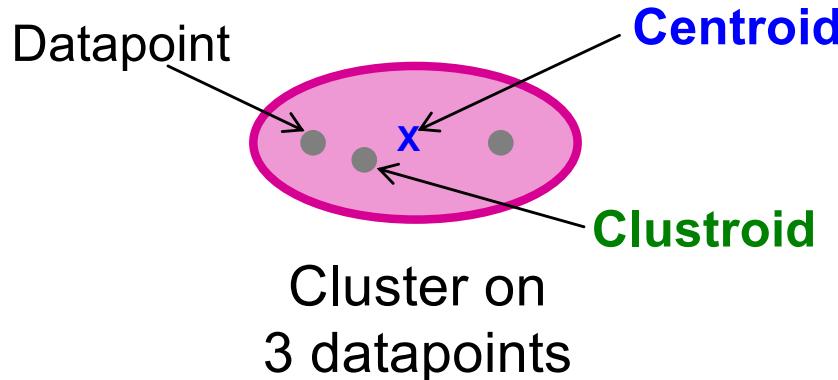
And in the Non-Euclidean Case?

What about the Non-Euclidean case?

- The only “locations” we can talk about are the points themselves
 - i.e., there is no “average” of two points
- Approach 1:
 - (1) How to represent a cluster of many points?
clustroid = (data)point “closest” to other points
 - (2) How do you determine the “nearness” of clusters? Treat clustroid as if it were centroid, when computing inter-cluster distances

“Closest” Point?

- (1) How to represent a cluster of many points?
clustroid = point “closest” to other points
- Possible meanings of “closest”:
 - Smallest **maximum** distance to other points
 - Smallest **average** distance to other points
 - Smallest **sum of squares** of distances to other points
 - For distance metric d clustroid c of cluster C is: $\min_c \sum_{x \in C} d(x, c)^2$



Centroid is the avg. of all (data)points in the cluster. This means centroid is an “artificial” point.

Clustroid is an **existing** (data)point that is “closest” to all other points in the cluster.

Defining “Nearness” of Clusters

- (2) How do you determine the “nearness” of clusters?
 - Approach 2:
Intercluster distance = minimum of the distances between any two points, one from each cluster
 - Approach 3:
Pick a notion of “**cohesion**” of clusters, e.g., maximum distance from the clustroid
 - Merge clusters whose **union** is most cohesive

Cohesion

- **Approach 3.1:** Use the **diameter** of the merged cluster = maximum distance between points in the cluster
- **Approach 3.2:** Use the **average distance** between points in the cluster
- **Approach 3.3:** Use a **density-based approach**
 - Take the diameter or avg. distance, e.g., and divide by the number of points in the cluster

Example

- Consider a cluster of 4 points:
 - abcd, aecdb, abecb, ecdab
- Their edit distances:

Insertion
Deletion
Substitution

	aecdः	abecः	ecdab
abcd	3	3	5
aecdः		2	2
abecः			4

Determine Clusteroid

- aecdb will be chosen as clusteroid
 - Located in “center” judged by all 3 measures

	aecdb	abecb	ecdab
abcd	3	3	5
aecdb		2	2
abecb			4

Point	Sum	Sum-sq	Max
abcd	11	43	5
aecdb	7	17	3
abecb	9	29	4
ecdab	11	45	5

Complexity of Hierarchical Clustering

- n data points
- At most $n - 1$ step of merging
- Naive implementation, e.g., storing pairwise cluster distances in a matrix

	C₁	C₂	C₃	C₄
C₁	0	2	3	2
C₂		0	4	5
C₃			0	3
C₄				0

Complexity of Naive Implementation

- Initially, $O(n^2)$ for creating matrix and finding pair with minimum distance
- Subsequent merge, assuming matrix: $k \times k$
 - Delete columns for old clusters: $O(k)$
 - Add new column for new cluster C' : $O(k)$
 - Compute dist. of C' with other clusters: $O(k)$
 - Find new pair of clusters with min. dist: $O(k^2)$

=> Overall complexity: $O(n^3)$

Implementation Summary

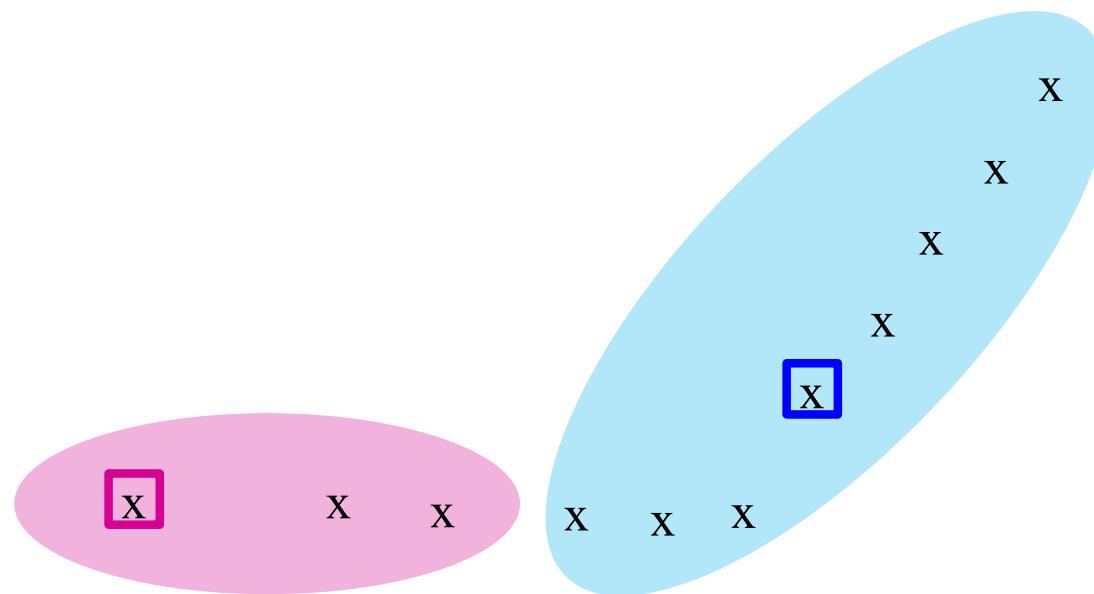
- **Naïve implementation of hierarchical clustering:**
 - At each step, compute pairwise distances between all pairs of clusters, then merge
 - $O(N^3)$
- Careful implementation using priority queue can reduce time to $O(N^2 \log N)$ (read textbook)
 - **Still too expensive for really big datasets that do not fit in memory**

k-means clustering

k -means Algorithm(s)

- Assumes Euclidean space/distance
- Start by picking k , the number of clusters
- Initialize clusters by picking one point per cluster
 - **Example:** Pick one point **at random**, then **$k-1$** other points, each **as far away as possible** from the previous points

Example: Assigning Clusters

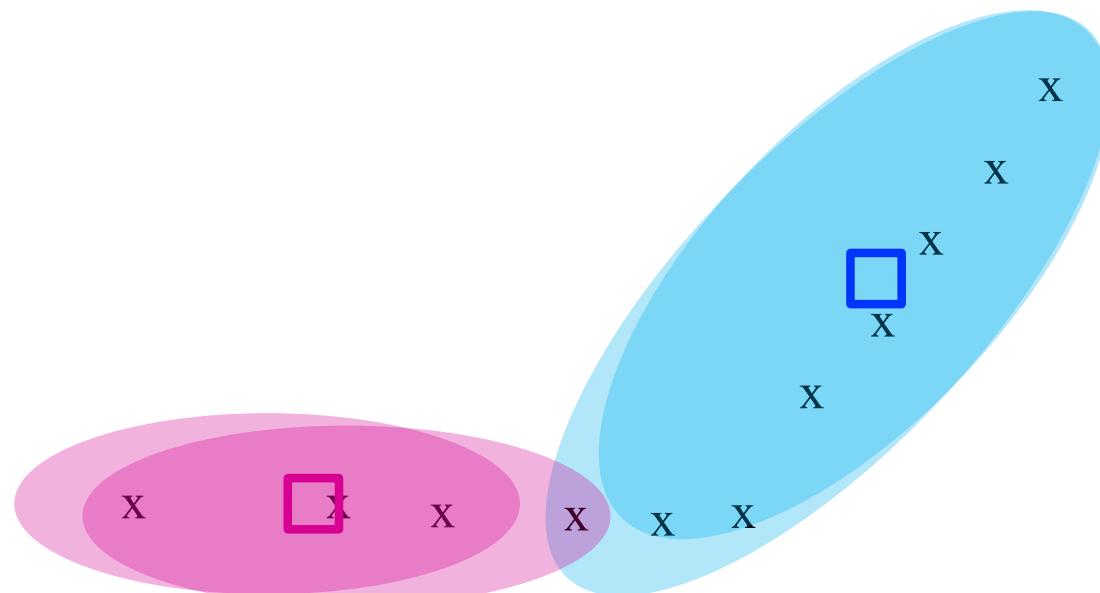


x ... data point

□ ... centroid

Clusters after round 1

Example: Assigning Clusters

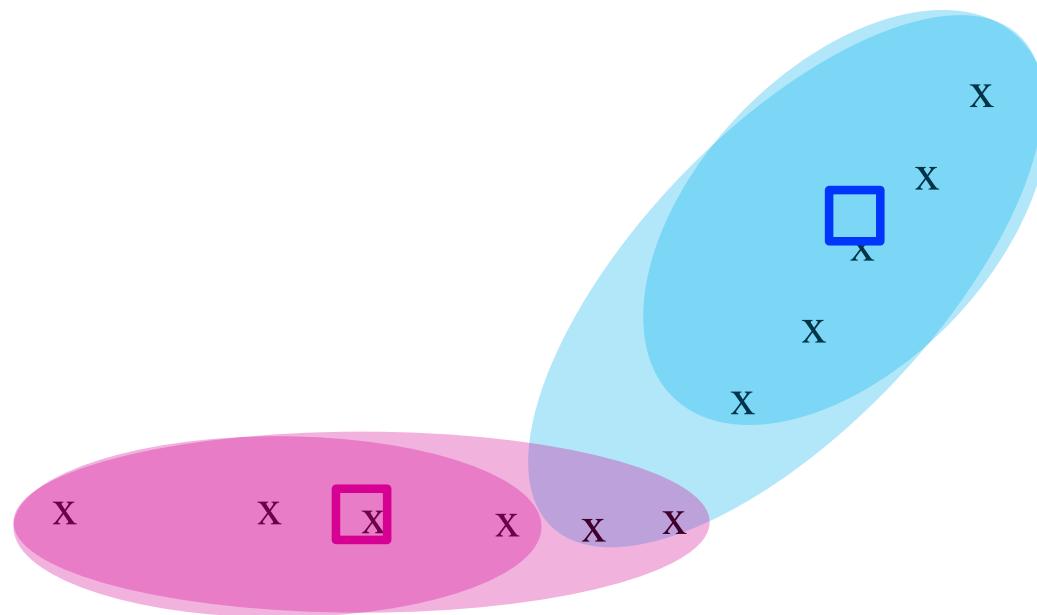


X ... data point

□ ... centroid

Clusters after round 2

Example: Assigning Clusters



X ... data point

□ ... centroid

Clusters at the end

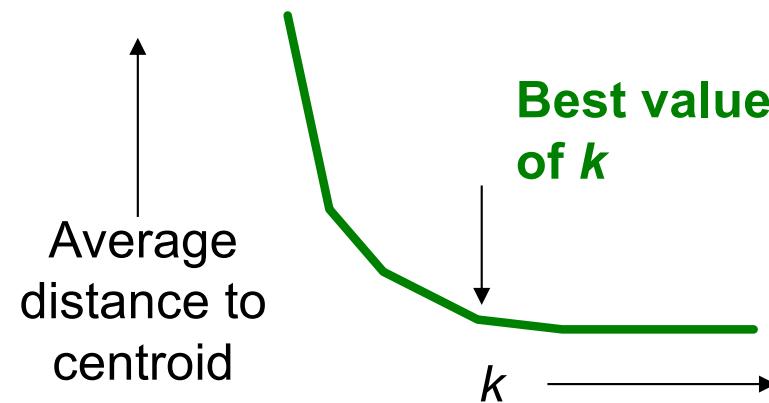
Populating Clusters

- 1) For each point, place it in the cluster whose current centroid it is nearest
- 2) After all points are assigned, update the locations of centroids of the k clusters
- 3) Reassign all points to their closest centroid
 - Sometimes moves points between clusters
- **Repeat 2 and 3 until convergence**
 - **Convergence:** Points don't move between clusters and centroids stabilize

Getting the k right

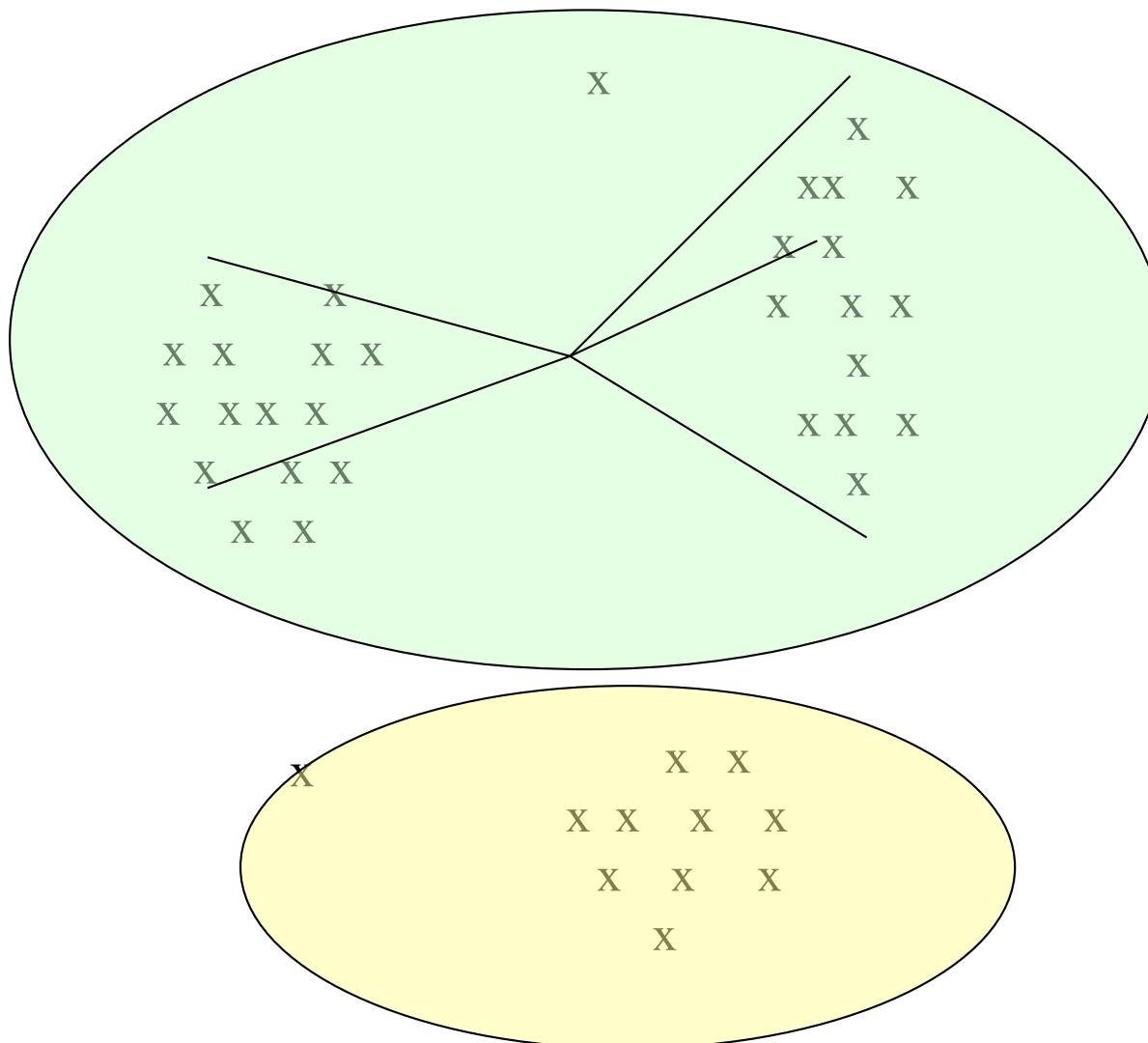
How to select k ?

- Try different k , looking at the change in the average distance to centroid as k increases
- Average falls rapidly until right k , then changes little



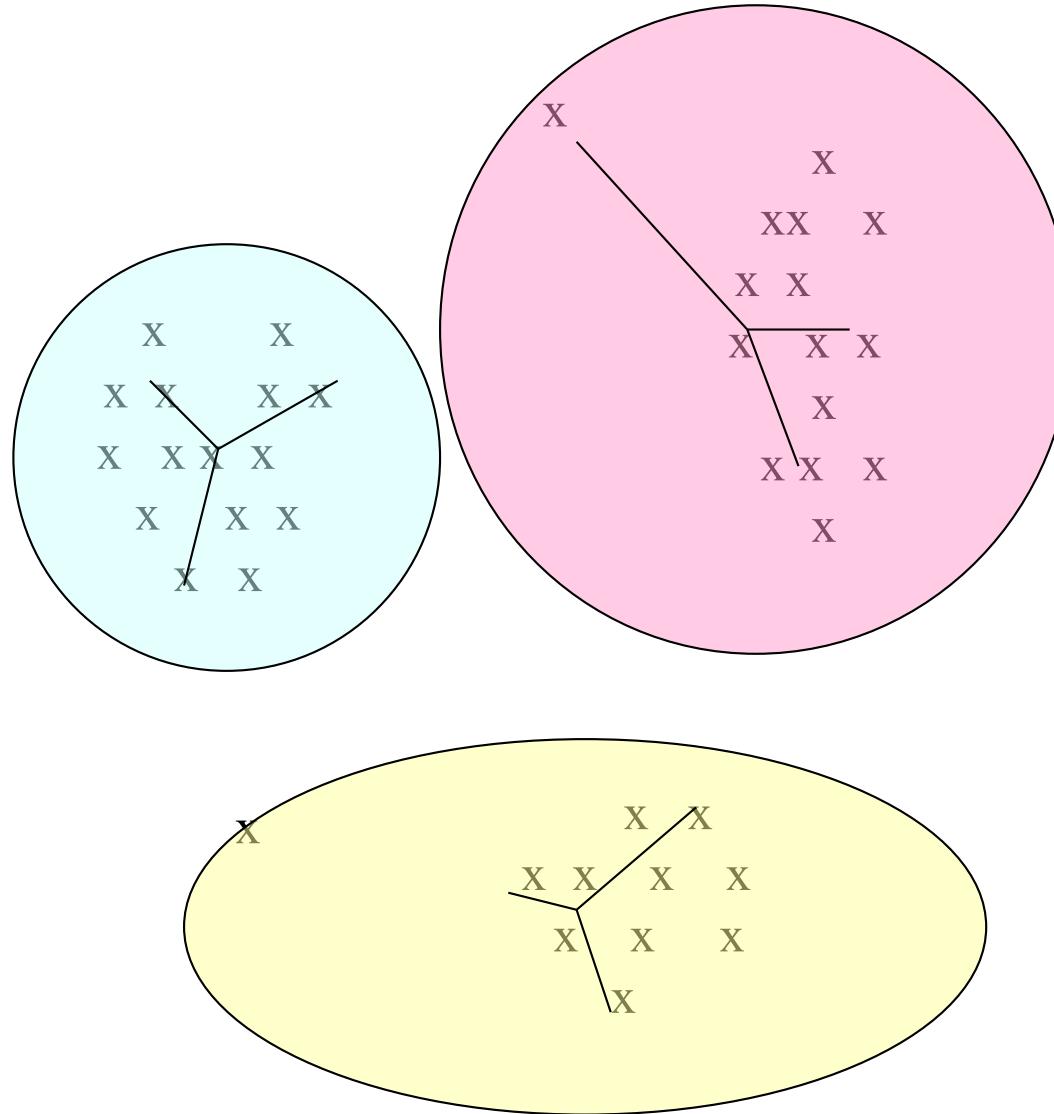
Example: Picking k

Too few;
many long
distances
to centroid.



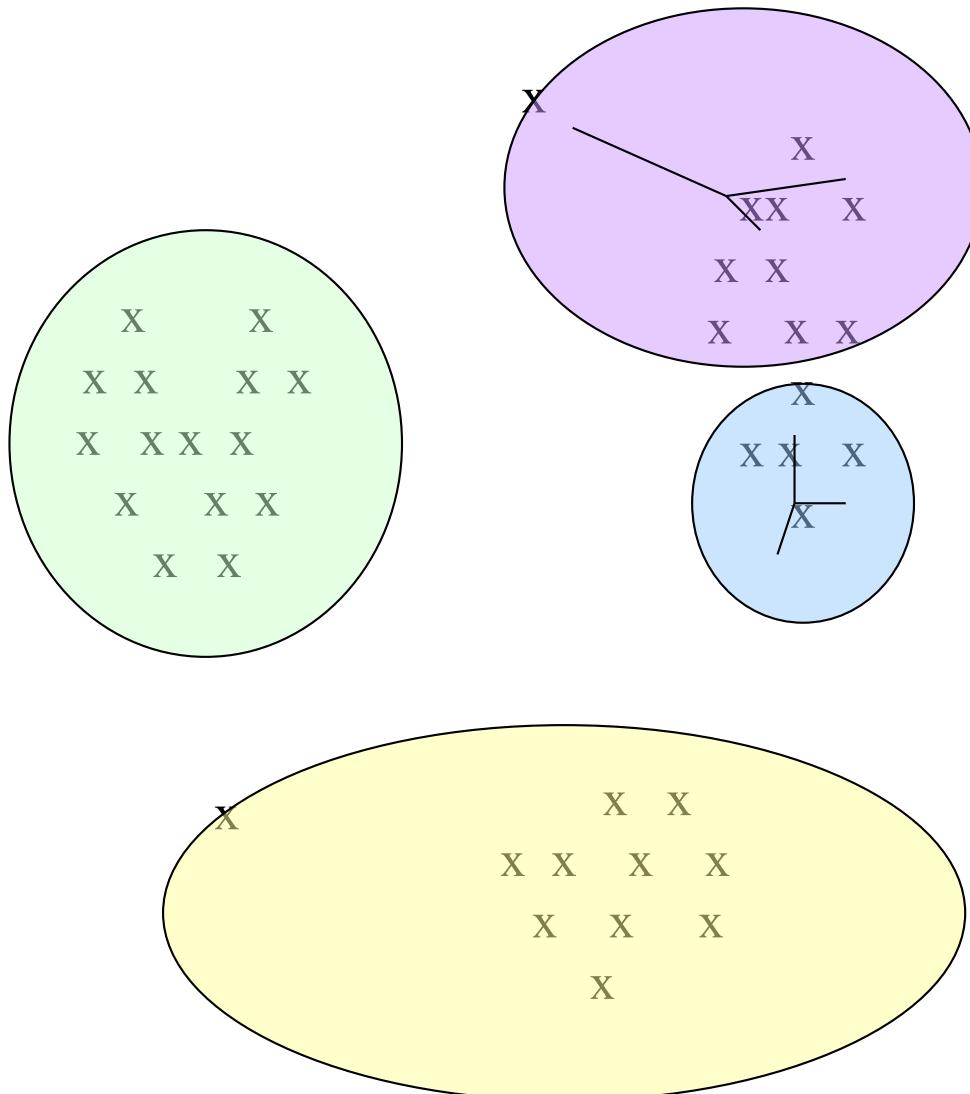
Example: Picking k

Just right;
distances
rather short.



Example: Picking k

Too many;
little improvement
in average
distance.



Summary

- **Clustering:** Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***
- **Algorithms:**
 - Agglomerative **hierarchical clustering**:
 - Centroid and clustroid
 - ***k*-means**:
 - Initialization, picking k