Welcome to

DS504/CS586: Big Data Analytics

***Big Data* Clustering**

Prof. Yanhua Li

Time: 6:00pm –8:50pm Thu
Location: AK 232
Fall 2016

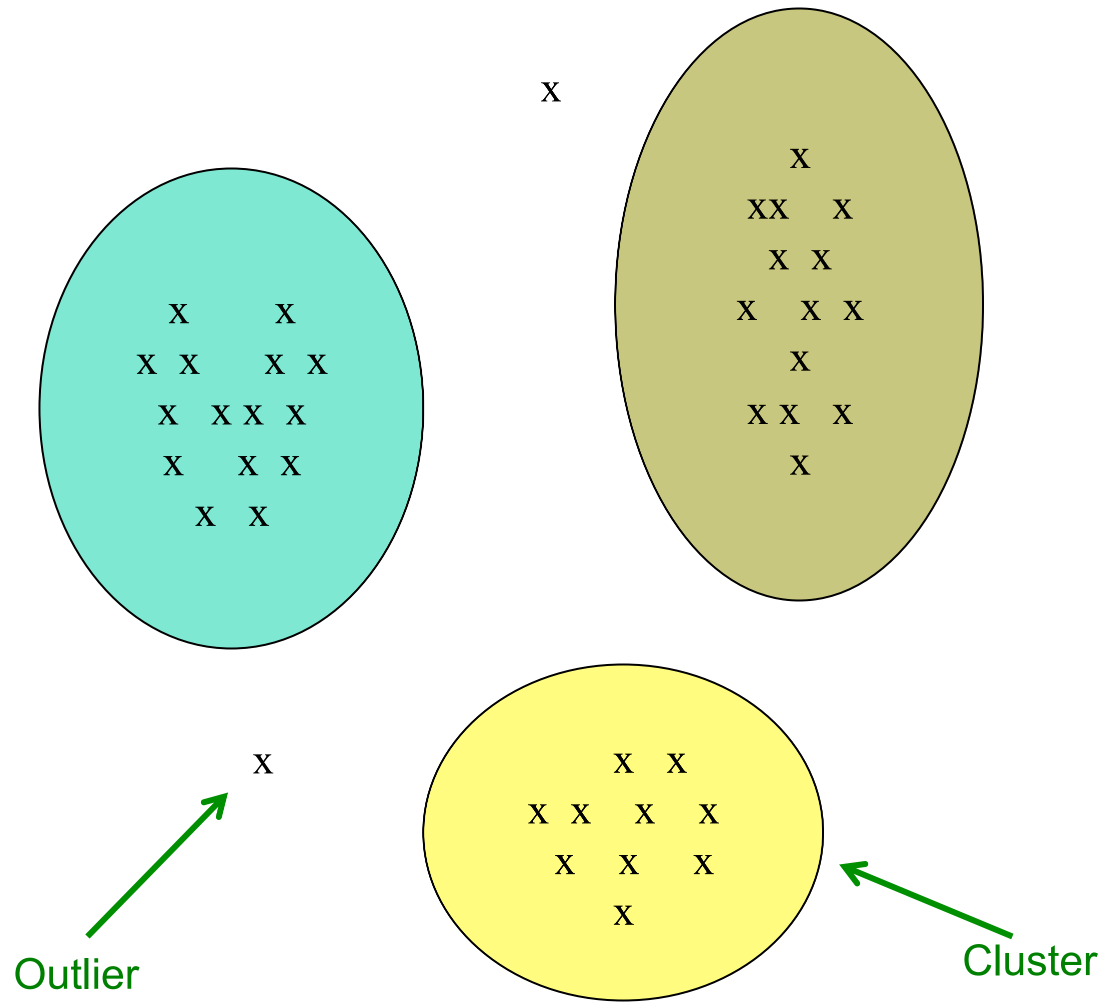
High Dimensional Data

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

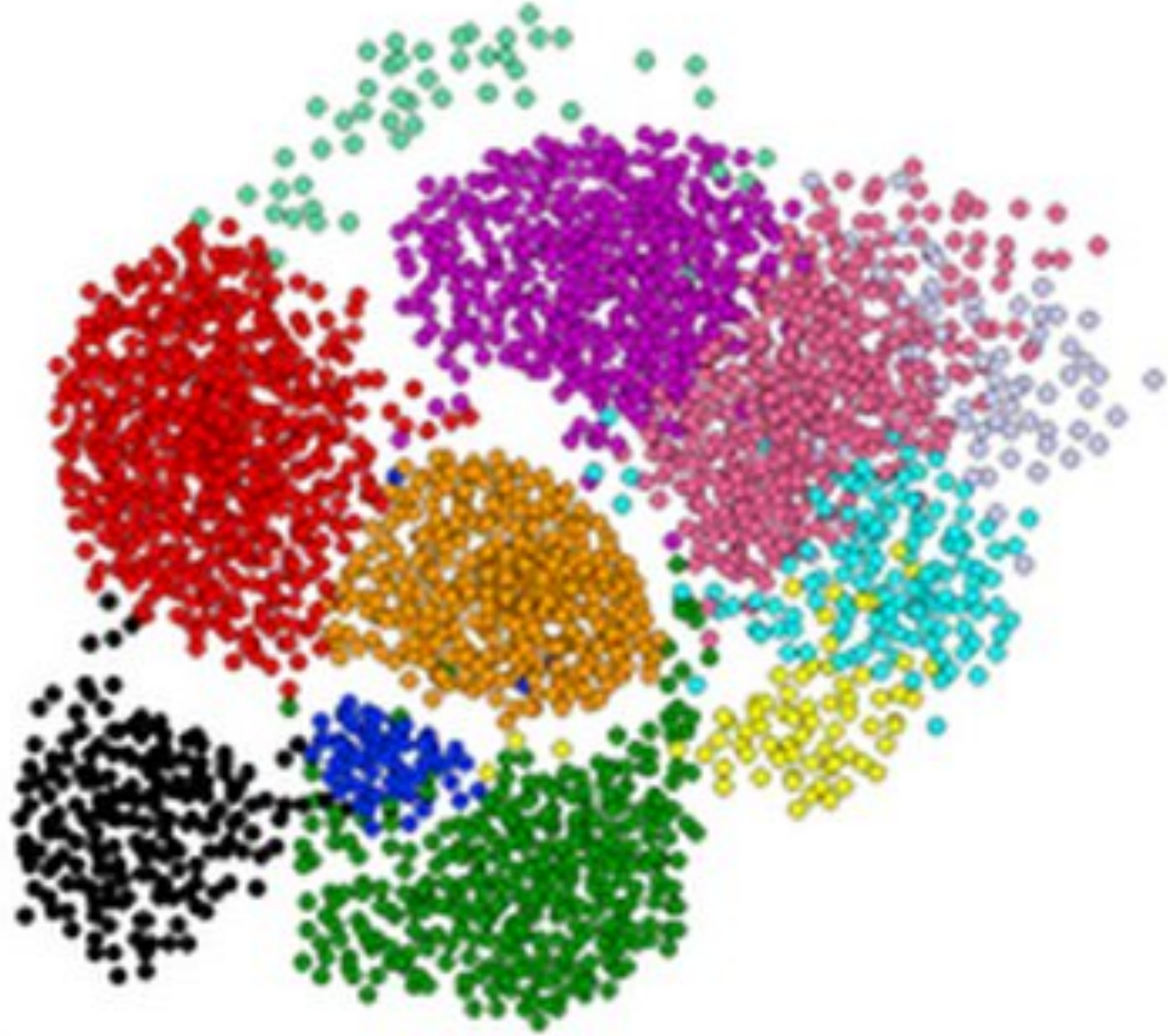
The Problem of Clustering

- 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 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
- And in most cases, looks are not deceiving

- 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

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:

- 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

Clustering Problem: Documents

Finding topics:

- Represent a document by a vector $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} word appears in the document
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words
- **Documents with similar sets of words may be about the same topic**

Cosine, Jaccard, and Euclidean

- **As with CDs we have a choice when we think of documents as sets of words:**
  - **Sets as vectors:** Measure similarity by the **cosine distance**

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

  - **Sets as sets:** Measure similarity by the **Jaccard distance**

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

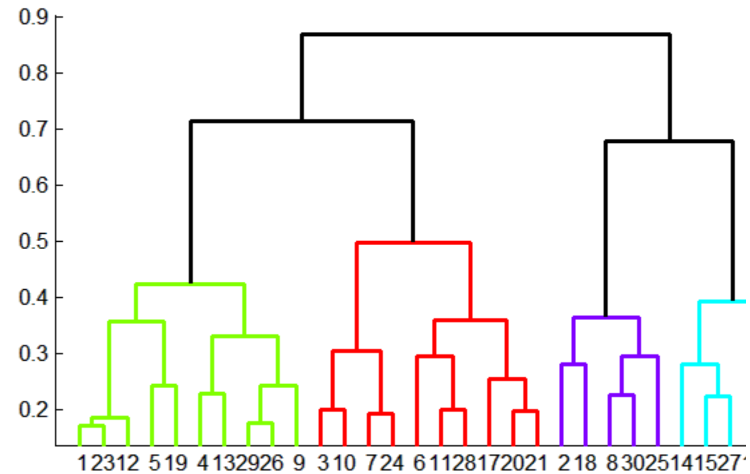
  - **Sets as points:** Measure similarity by **Euclidean distance**

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$

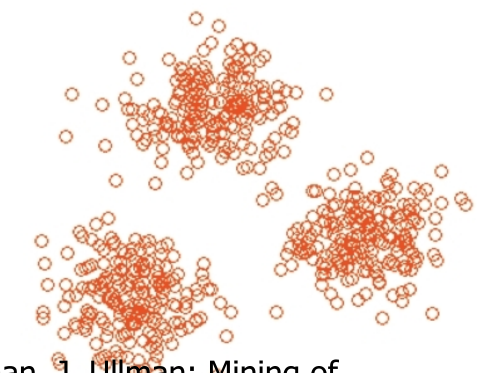
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Overview: Methods of Clustering

- **Hierarchical:**
  - (bottom up):
    - Initially, each point is a cluster
    - Repeatedly combine the two "nearest" clusters into one
  - (top down):
    - Start with one cluster and recursively split it

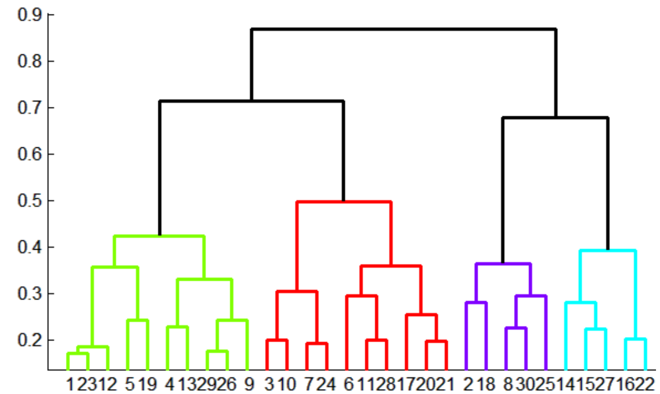


- **Point assignment:**
  - Maintain a set of clusters
  - Points belong to "nearest" cluster



Hierarchical Clustering

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

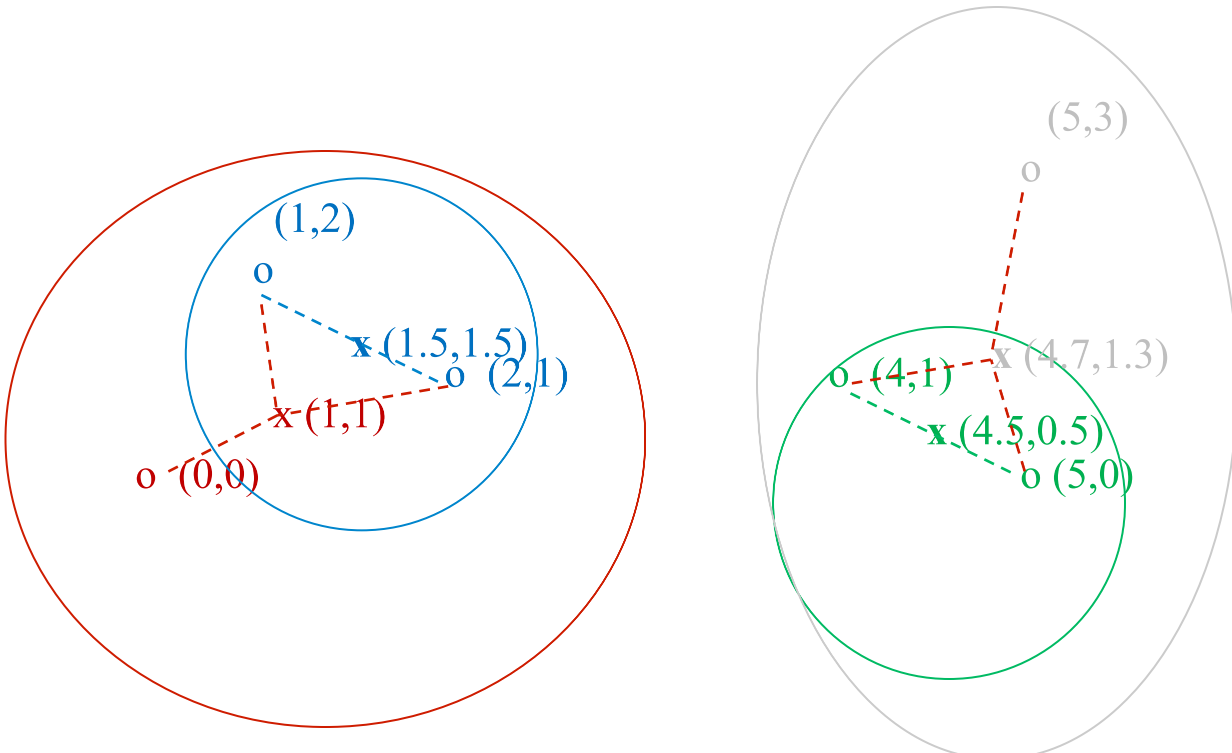


- **Three important questions:**
  - **1)** How do you represent a cluster of more than one point?
  - **2)** How do you determine the "nearness" of clusters?
  - **3)** When to stop combining clusters?

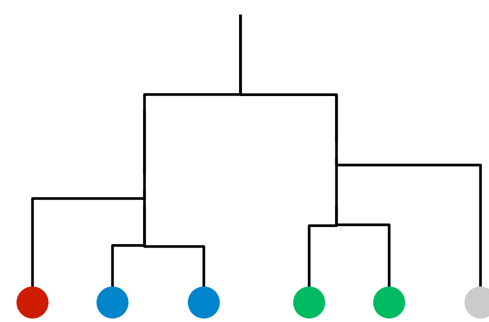
Hierarchical Clustering

- **Key operation: Repeatedly combine two nearest clusters**
- **(1) How to represent a cluster of many points?**
  - **Key problem:** As you merge clusters, how do you represent the "location" of each cluster, to tell which pair of clusters is closest?
  - **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



Data:
o … data point
x … centroid

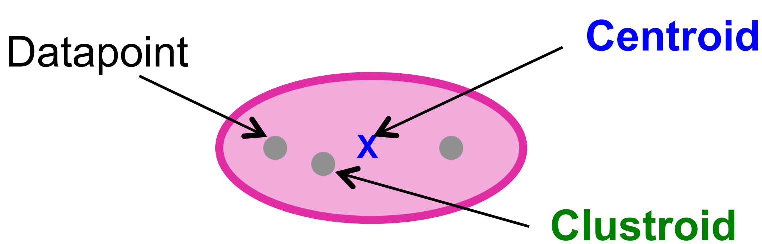


Dendrogram

"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:

$$\min_c \sum_{x \in C} d(x,c)^2$$



Cluster on
3 datapoints

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.

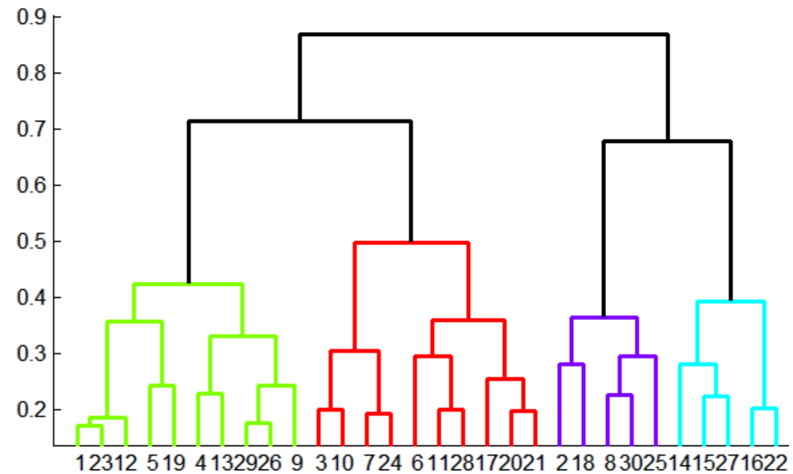
Defining "Nearness" of Clusters

- **(2) How do you determine the "nearness" of clusters?**
  - **Approach 1:**
    Intercluster distance = minimum of the distances between any two points, one from each cluster
  - **Approach 2:**
    Pick a notion of "**cohesion**" of clusters, e.g., maximum distance in the cluster
    - Merge clusters whose *union* is most cohesive

Cohesion

- **Approach 2.1:** Use the **diameter** of the merged cluster = maximum distance between points in the cluster
- **Approach 2.2:** Use the **average distance** between points in the cluster
- **Approach 2.3:** Use a **density-based approach**
  - Take the diameter or avg. distance, e.g., and divide by the number of points in the cluster

Implementation



- **Naïve implementation of hierarchical clustering:**
  - At each step, compute pairwise distances between all pairs of clusters $O(N^2)$, with up to N steps.
  - Then merge with in total $O(N^3)$

  - **Too expensive for really big datasets that do not fit in memory**

k-means clustering

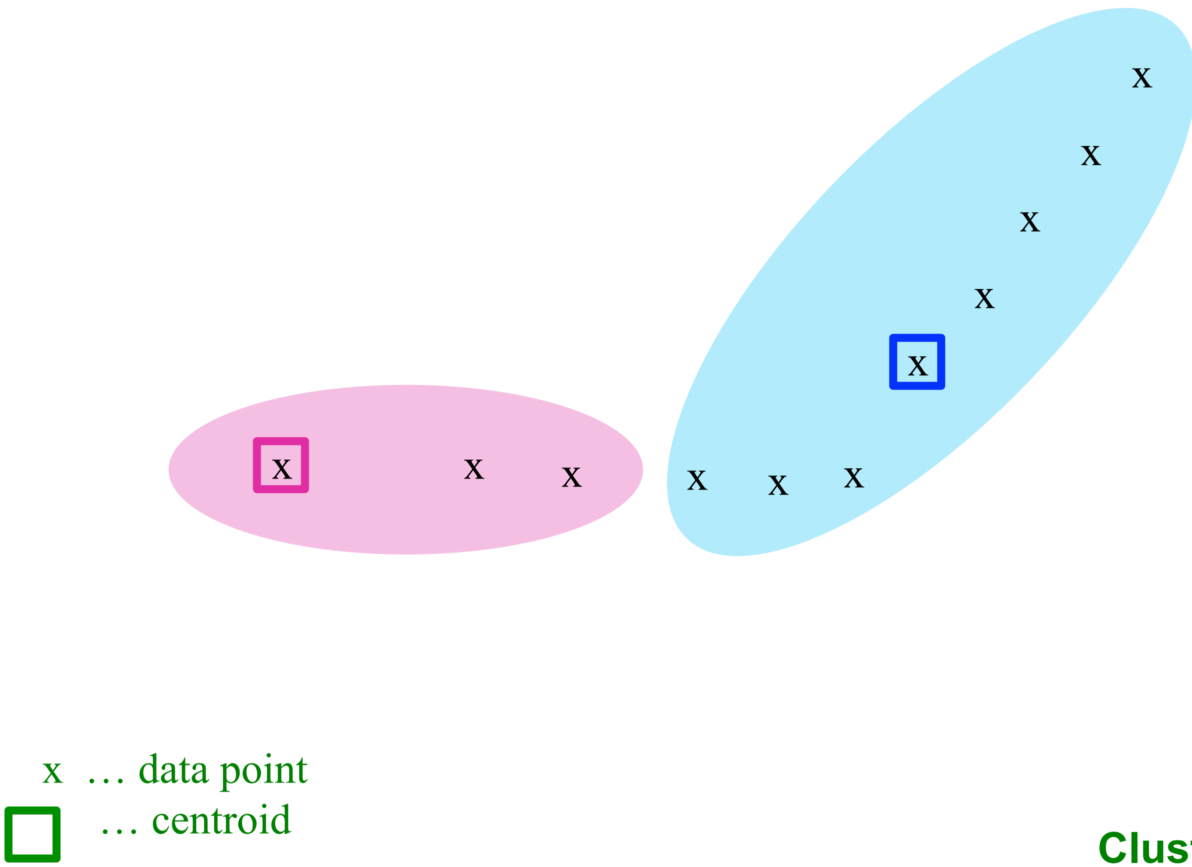
k–means Algorithm(s)

- 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

Populating Clusters

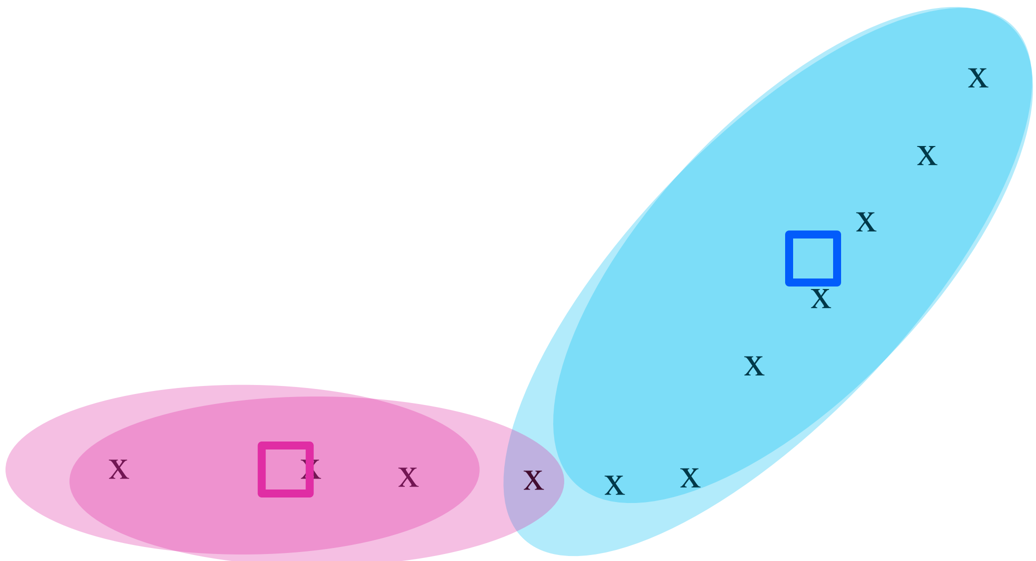
- **1)** For each point, place it in the cluster whose current centroid it is nearest
- **2)** After all points are assigned, update the locations of centroids of the ***k*** clusters
- **3)** Reassign all points to their closest centroid
  - Sometimes moves points between clusters
- **Repeat 2 and 3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

Example: Assigning Clusters



Clusters after round 1

Example: Assigning Clusters

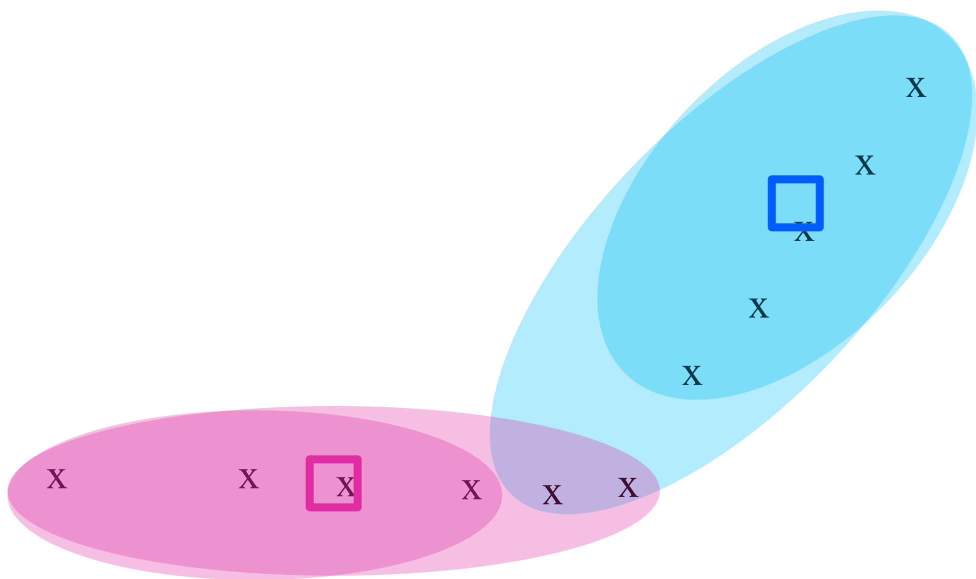


x … data point

… centroid

Clusters after round 2

Example: Assigning Clusters



x … data point

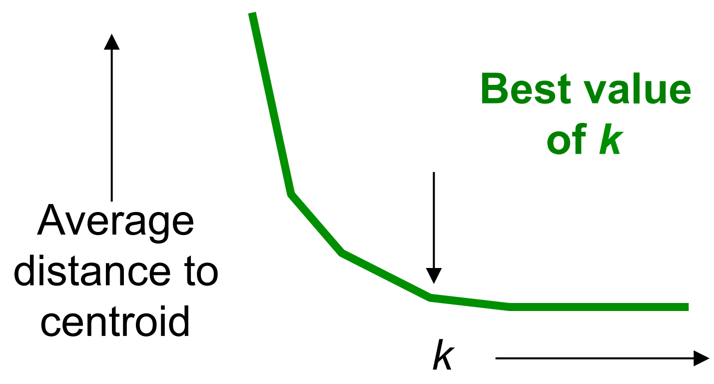
… centroid

Clusters at the end

Getting the *k* right

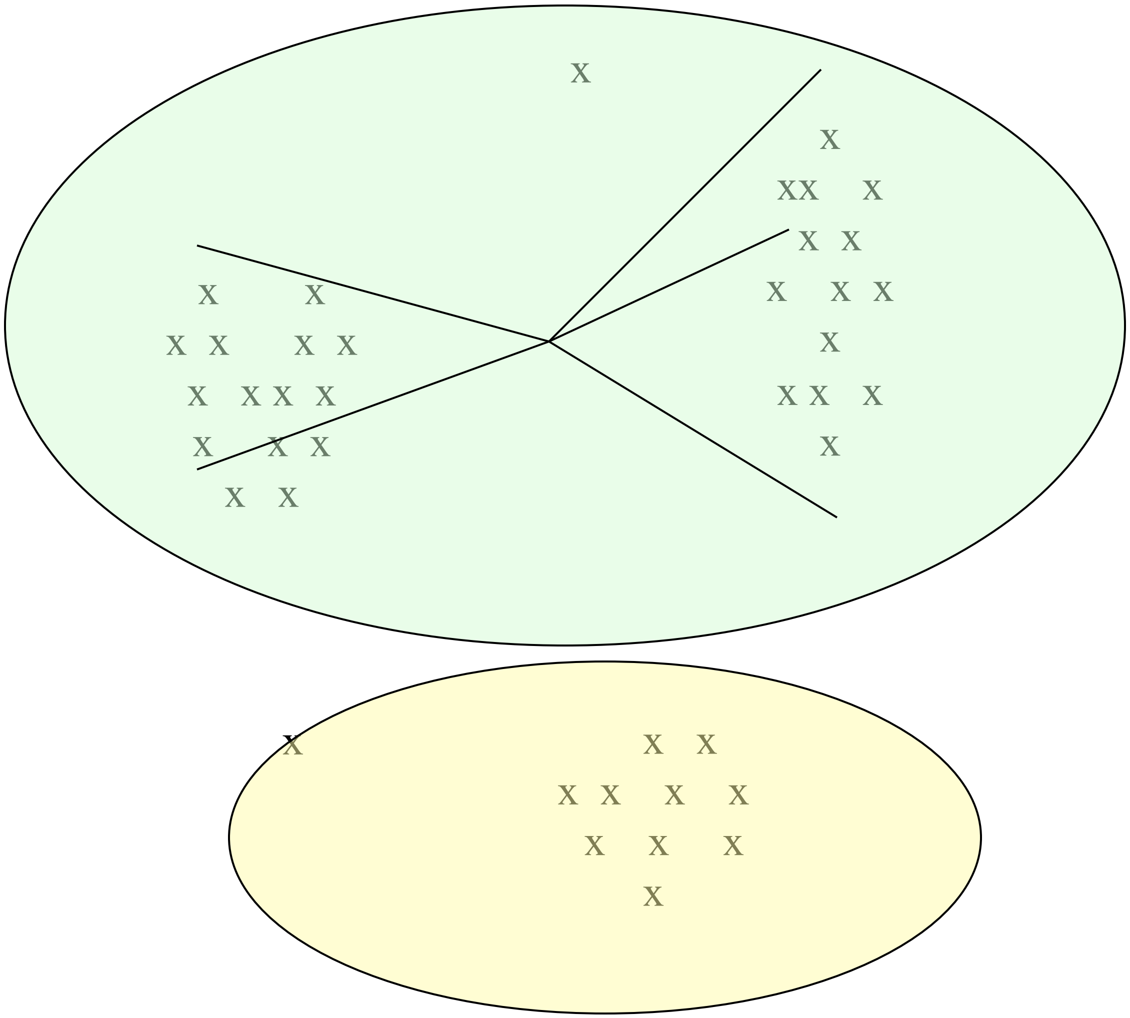
How to select *k*?

- 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



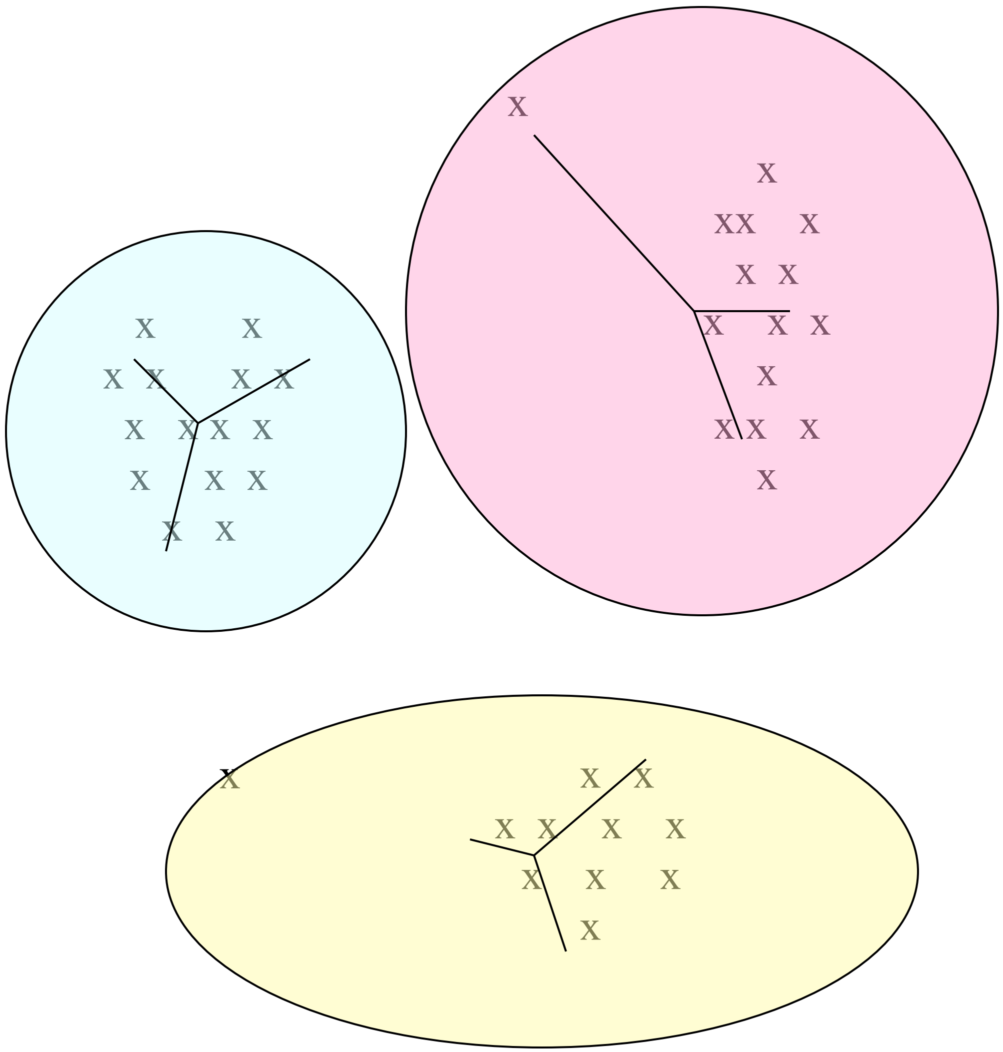
Example: Picking k=2

Too few;
many long
distances
to centroid.



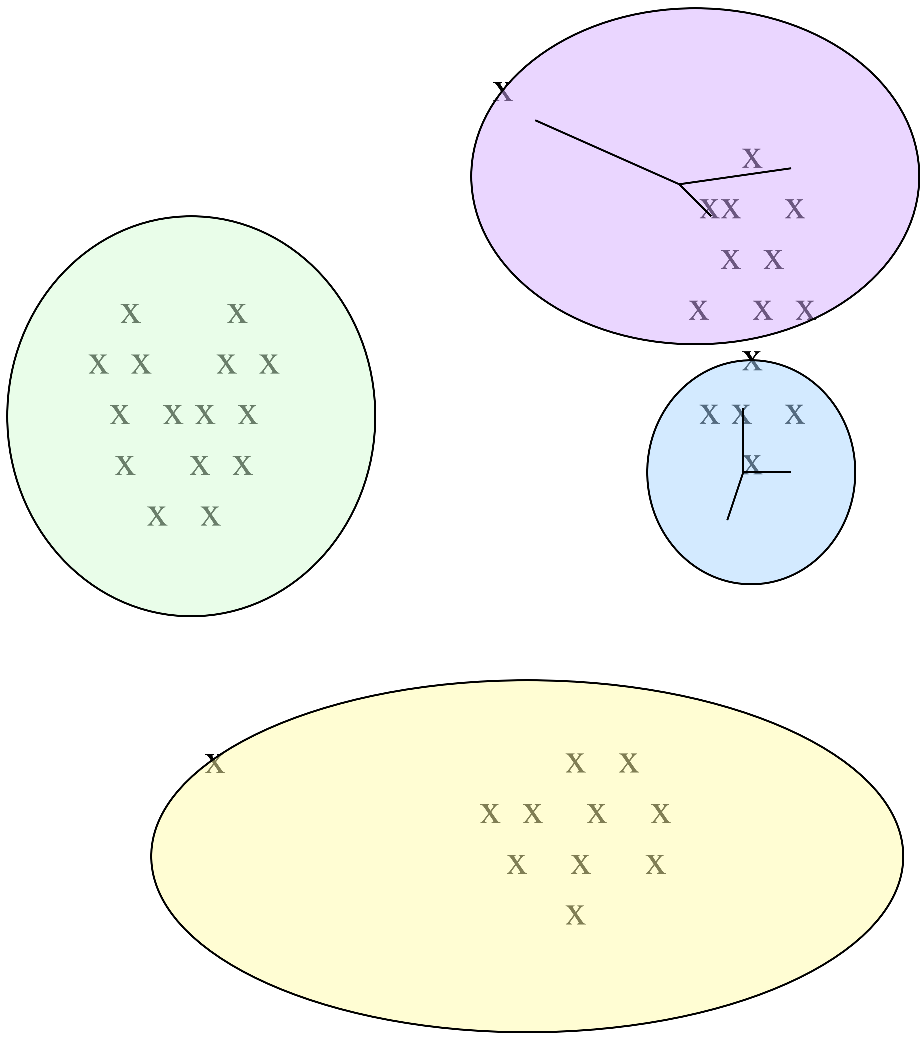
Example: Picking k=3

Just right;
distances
rather short.



Example: Picking *k*

Too many;
little improvement
in average
distance.



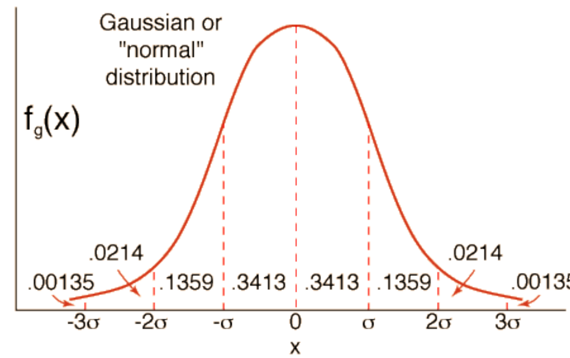
Populating Clusters

- **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

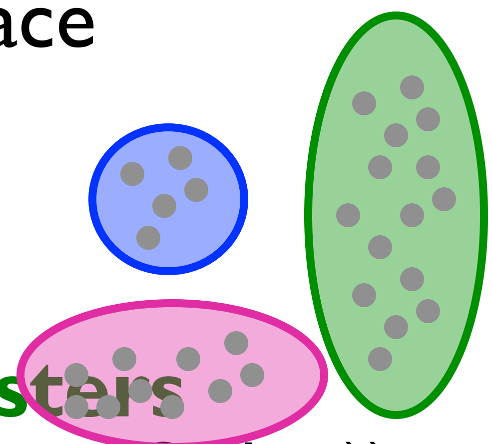
The BFR Algorithm

Extension of *k*-means to large data

BFR Algorithm



- **BFR** [Bradley-Fayyad-Reina] is a variant of *k*-means designed to handle **very large** (disk-resident) data sets
- **Assumes** that clusters are normally distributed around a centroid in a Euclidean space
  - Standard deviations in different dimensions may vary
    - Clusters are axis-aligned ellipses



- **Efficient way to summarize clusters** (want memory required O(clusters) and not O(data))

BFR Algorithm

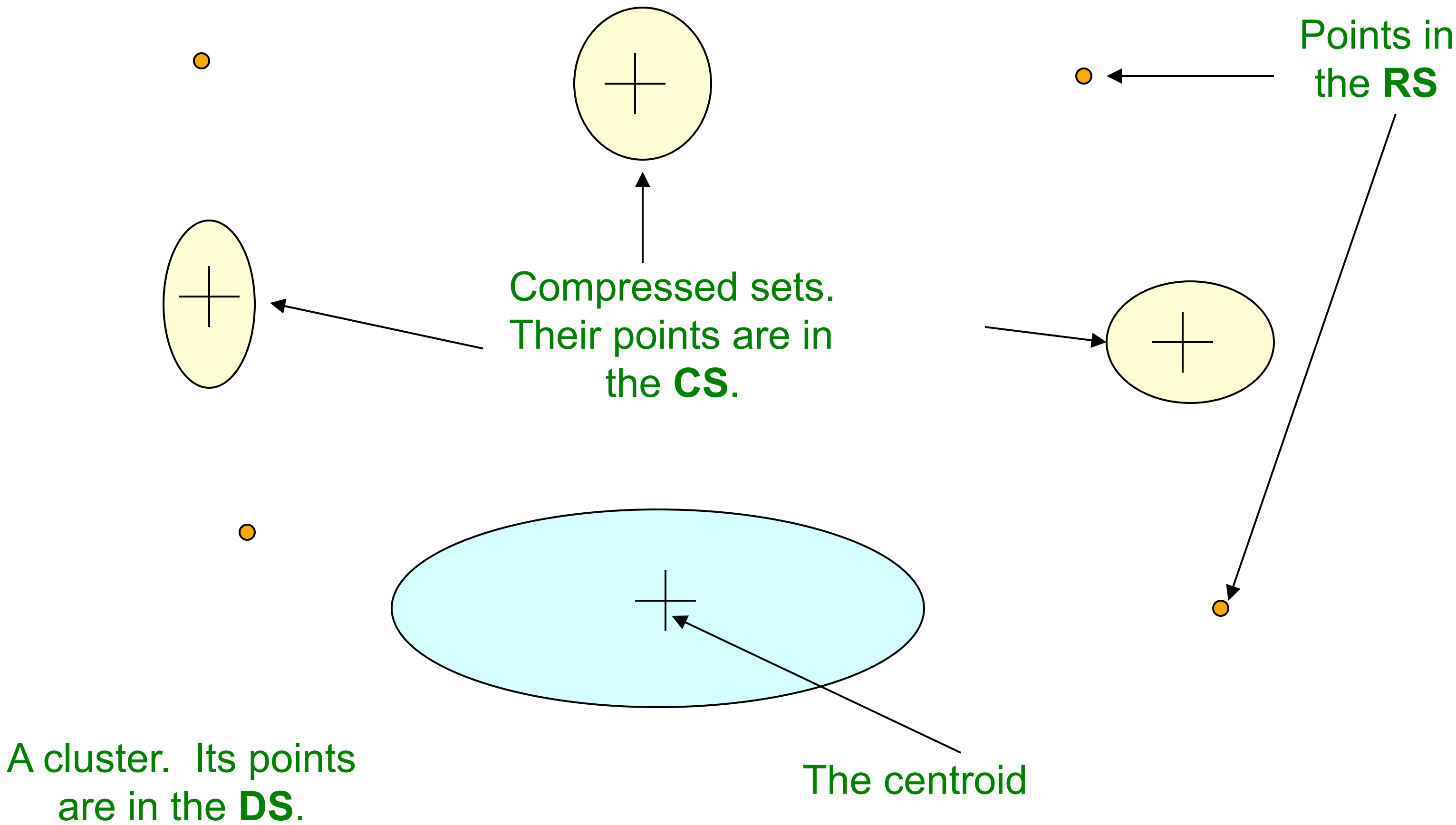
- Points are read from disk one main-memory-full at a time
- **Most points from previous memory loads are summarized by simple statistics**
- To begin, from the initial load we select the initial ***k*** centroids by some sensible approach:
  - Take ***k*** random points
  - Take a small random sample and cluster optimally
  - Take a sample; pick a random point, and then ***k–1*** more points, each as far from the previously selected points as possible

Three Classes of Points

3 sets of points which we keep track of:

- **Discard set (DS):**
  - Points close enough to a centroid to be summarized
- **Compression set (CS):**
  - Groups of points that are close together but not close to any existing centroid
  - These points are summarized, but not assigned to a cluster
- **Retained set (RS):**
  - Isolated points waiting to be assigned to a compression set

BFR: "Galaxies" Picture

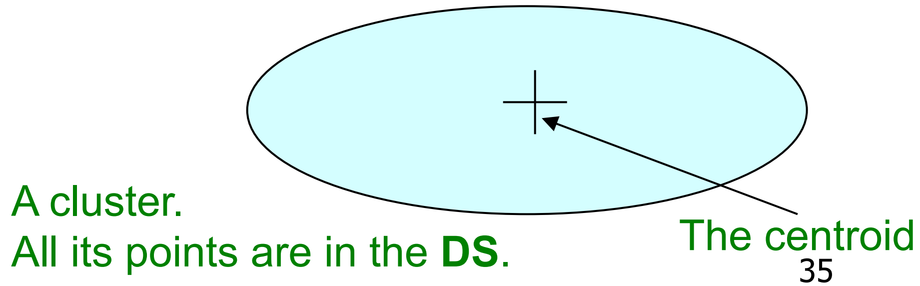


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

For each cluster, the discard set (DS) is summarized by:

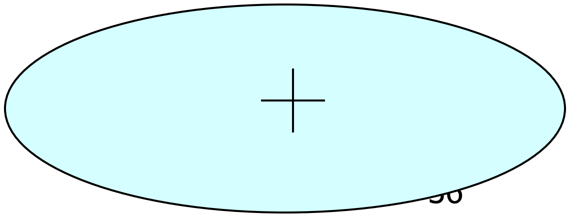
- The number of points, ***N***
- The vector ***SUM***, whose i^{th} component is the sum of the coordinates of the points in the i^{th} dimension
- The vector ***SUMSQ***: i^{th} component = sum of squares of coordinates in i^{th} dimension



Summarizing Points: Comments

- **2*d* + 1** values represent any size cluster
  - ***d*** = number of dimensions
- Average in **each dimension** (**the centroid**) can be calculated as $\mathbf{SUM}_i\,/\,N$
  - $\mathbf{SUM}_i$ = i^{th} component of SUM
- Variance of a cluster's discard set in dimension i is: $(\mathbf{SUMSQ}_i\,/\,N) - (\mathbf{SUM}_i\,/\,N)^2$
  - And standard deviation is the square root of that
- **Next step: Actual clustering**

Note: Dropping the "axis-aligned" clusters assumption would require storing full covariance matrix to summarize the cluster. So, instead of **SUMSQ** being a ***d***-dim vector, it would be a ***d x d*** matrix, which is too big!



The "Memory-Load" of Points

Processing the "Memory-Load" of points (1):

- **1)** Find those points that are "**sufficiently close**" to a cluster centroid and add those points to that cluster and the **DS**
  - These points are so close to the centroid that they can be summarized and then discarded
- **2)** Use any main-memory clustering algorithm to cluster the remaining points and the old **RS**
  - Clusters go to the **CS**; outlying points to the **RS**

Discard set (DS): Close enough to a centroid to be summarized.
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

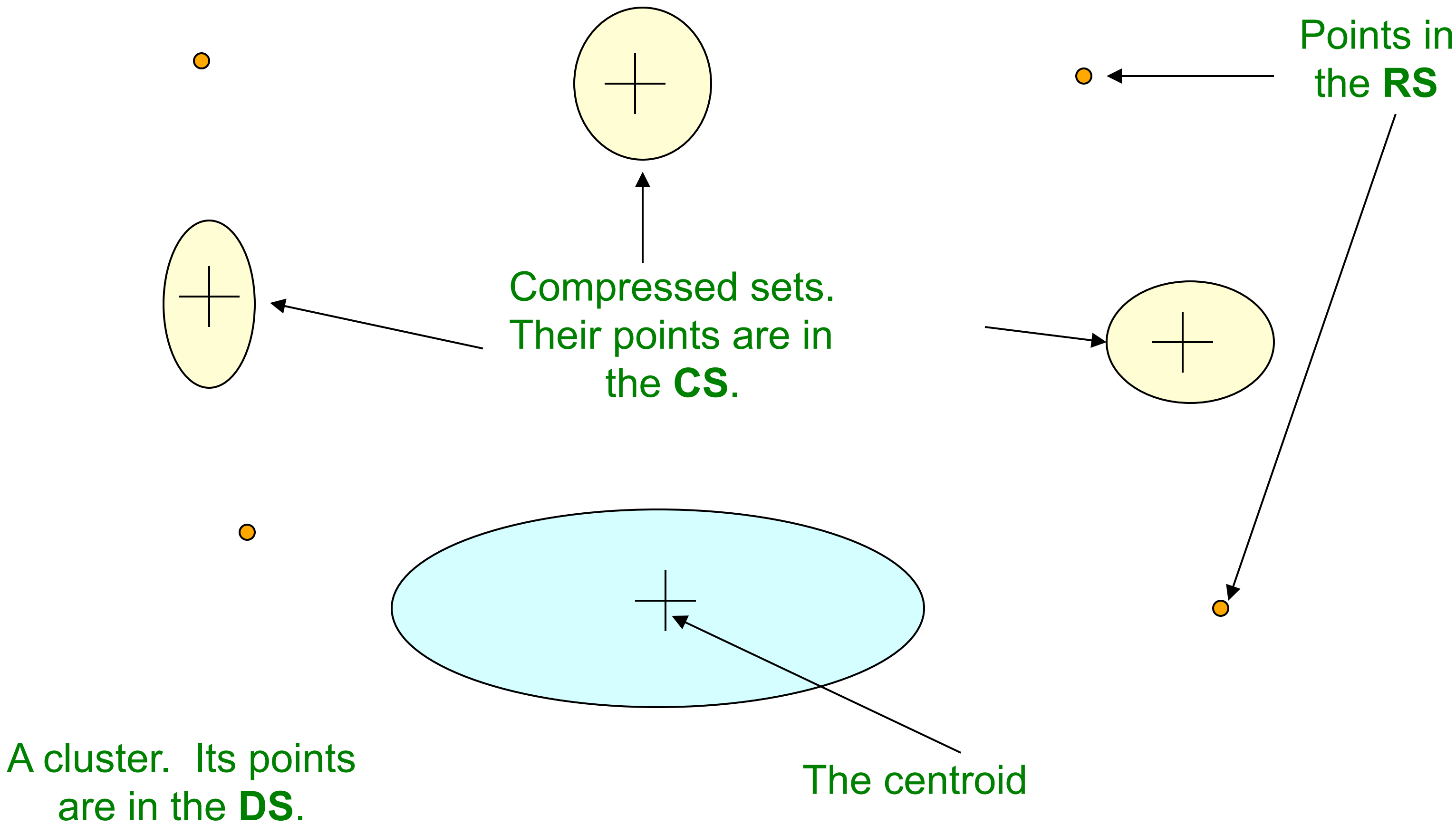
The "Memory-Load" of Points

Processing the "Memory-Load" of points (2):

- **3) DS set:** Adjust statistics of the clusters to account for the new points
  - Add ***N***s, ***SUM***s, ***SUMSQ***s
- **4)** Consider merging compressed sets in the **CS**
- **5)** If this is the last round, merge all compressed sets in the **CS** and all **RS** points into their nearest cluster

Discard set (DS): Close enough to a centroid to be summarized.
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

BFR: "Galaxies" Picture



Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

A Few Details…

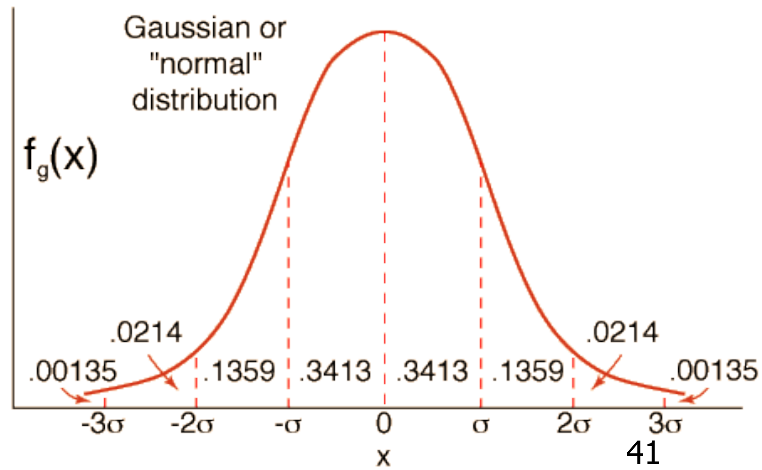
- **Q1) How do we decide if a point is "close enough" to a cluster that we will add the point to that cluster?**

- **Q2) How do we decide whether two compressed sets (CS) deserve to be combined into one?**

How Close is Close Enough?

- **Q1) We need a way to decide whether to put a new point into a cluster (and discard)**

- **BFR suggests two ways:**
  - The **Mahalanobis distance** is less than a threshold
  - **High likelihood of the point belonging to currently nearest centroid**



Mahalanobis Distance

- **Normalized Euclidean distance from centroid**

- For point $(x_1, \ldots, x_d)$ and centroid $(c_1, \ldots, c_d)$
  1. Normalize in each dimension: $y_i = (x_i - c_i) / \sigma_i$
  2. Take sum of the squares of the y_i
  3. Take the square root

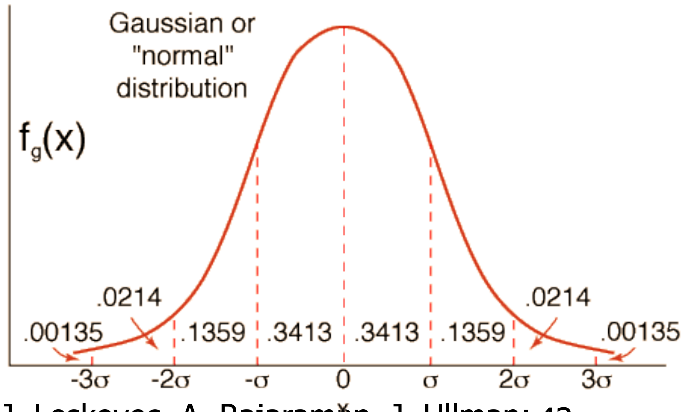
$$d(x,c) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_i - c_i}{\sigma_i}\right)^2}$$

σ_i … standard deviation of points in the cluster in the i^{th} dimension

Mahalanobis Distance

- If clusters are normally distributed in $\boldsymbol{d}$ dimensions, then after transformation, one standard deviation = $\sqrt{\boldsymbol{d}}$
  - i.e., 68% of the points of the cluster will have a Mahalanobis distance $< \sqrt{\boldsymbol{d}}$

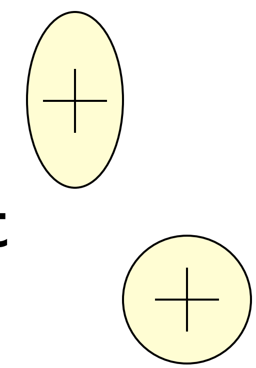
- Accept a point for a cluster if its M.D. is **<** some threshold, e.g. **2** standard deviations



Should 2 CS clusters be combined?

Q2) Should 2 CS subclusters be combined?

- Compute the variance of the combined subcluster
  - ***N***, ***SUM***, and ***SUMSQ*** allow us to make that calculation quickly
- Combine if the combined variance is below some threshold



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*
  - **BFR**

Any Questions?