Data Mining and Analysis

Clustering 2

CSCE 676 :: Fall 2019
Texas A&M University
Department of Computer Science & Engineering
Prof. James Caverlee

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, ...

Typical applications

- As a **stand-alone tool** to get insight into data distribution
- As a **preprocessing step** for other algorithms

Clustering: Examples

Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species

Information retrieval: document clustering

Land use: Identification of areas of similar land use in an earth observation database

Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs

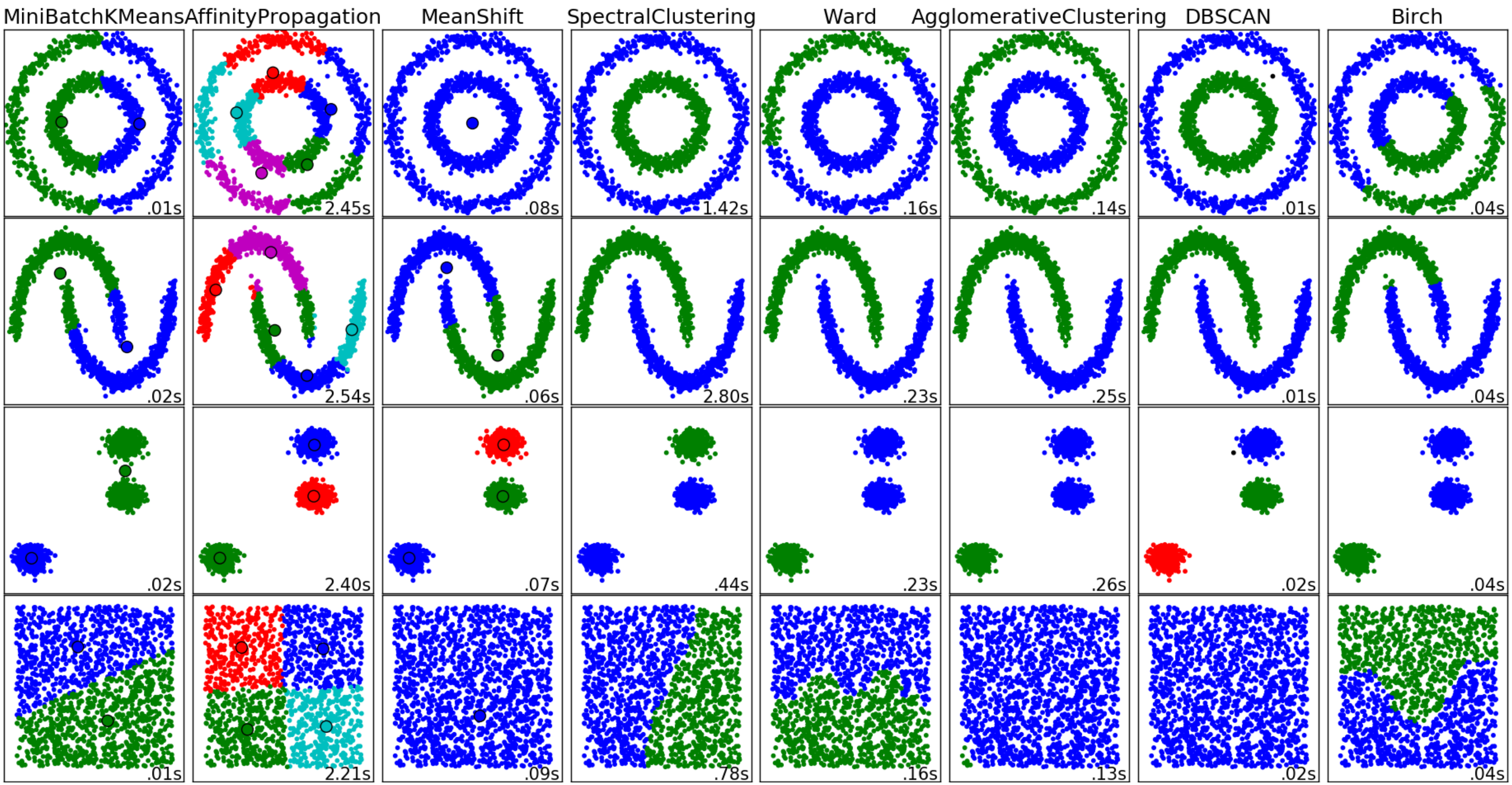
City-planning: Identifying groups of houses according to their house type, value, and geographical location

Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

Climate: understanding earth climate, find patterns of atmospheric and ocean

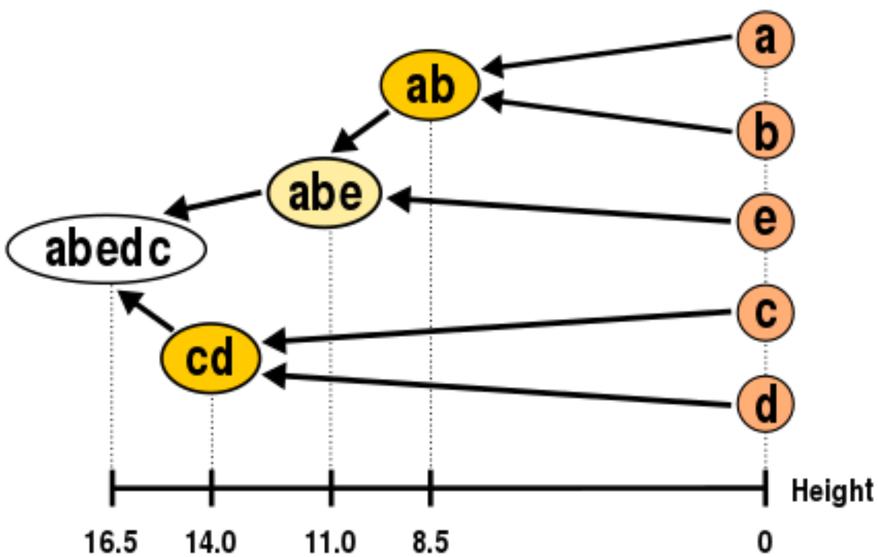
Economic Science: market research

scikit-learn: lots of options

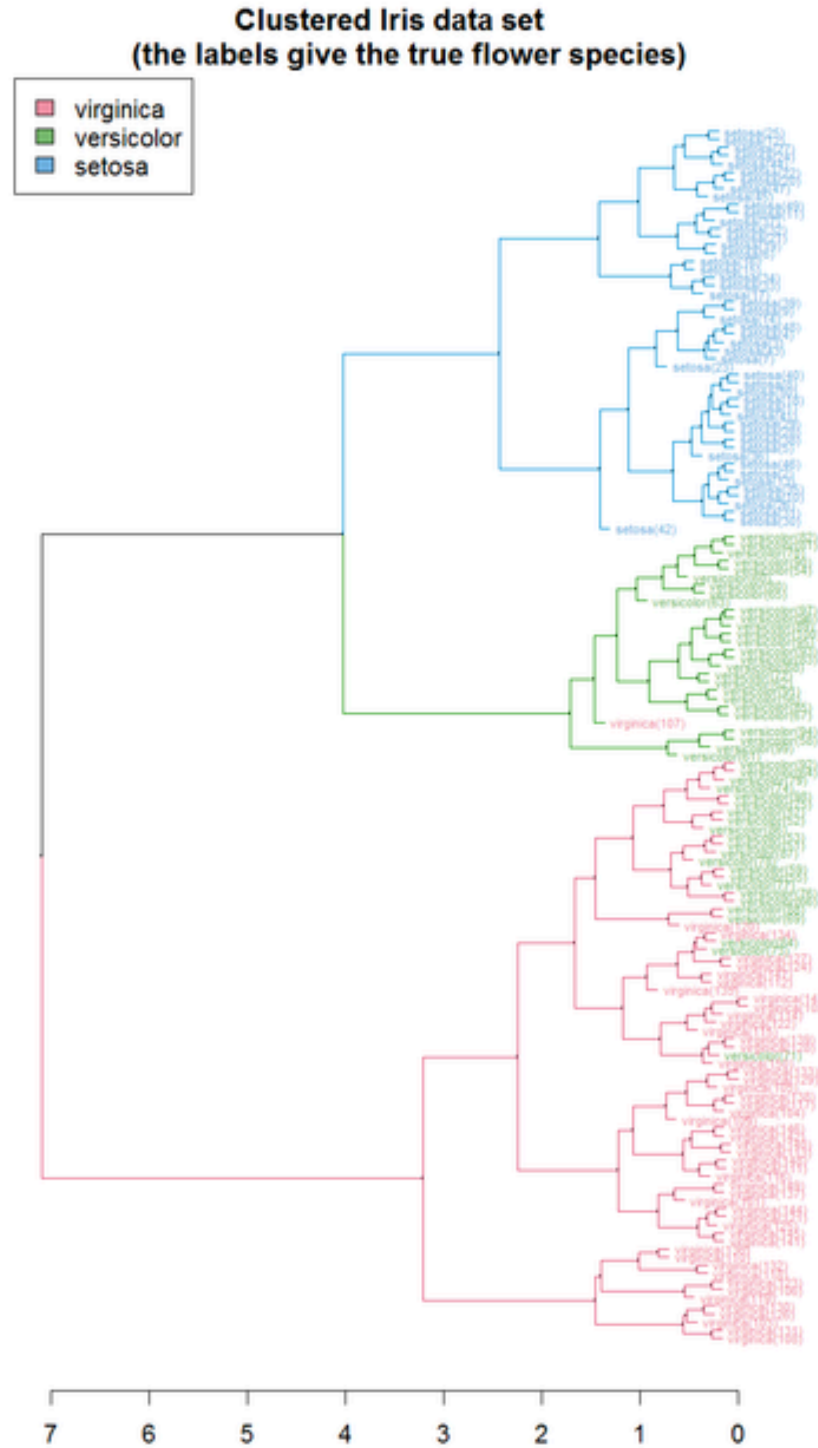


Hierarchical Clustering

Hierarchical Clustering



Dendrograms



Divisive (Top-Down) Approach: Bisecting K-means

For I=1 to k-1 do

  Pick a leaf cluster C to split

  For J=1 to ITER do

    Use K-means to split C into two sub-clusters, C1 and C2

  Choose the best of the above splits and make it permanent

Hierarchical (Bottom Up) 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?

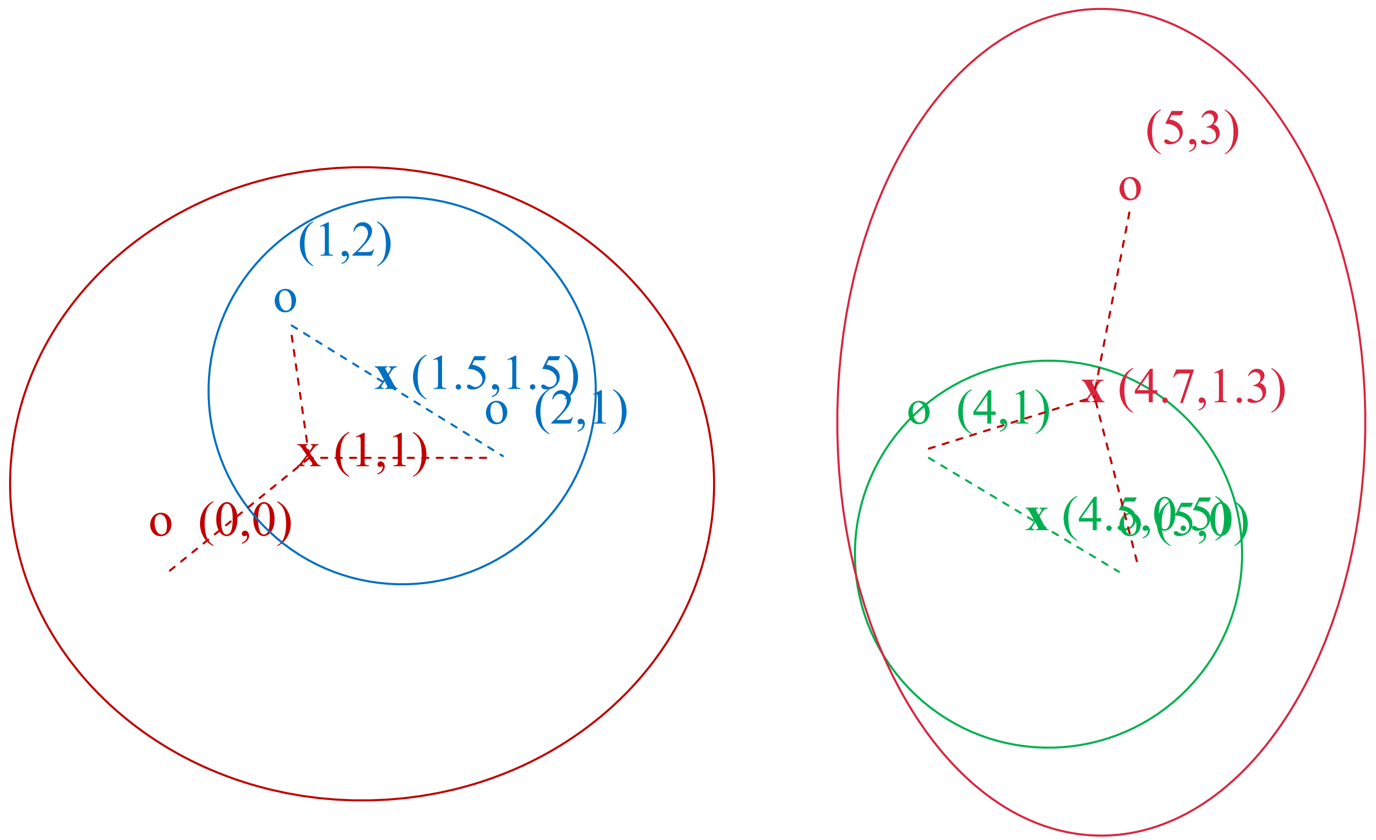
3) When to stop combining clusters?

1. How to Represent a Cluster of Many Points?

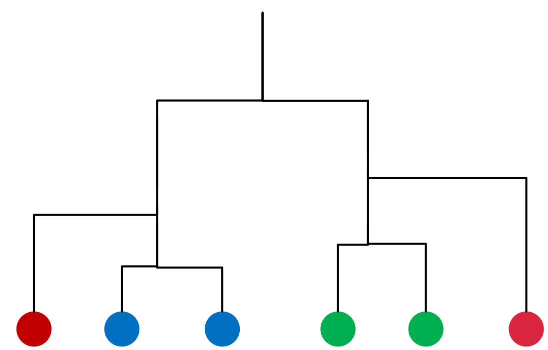
Key problem: As you merge clusters, how do you represent the “location” of each cluster, to tell which pair of clusters is closest?

Euclidean case: each cluster has a centroid = average of its (data)points

Example



Data:
o … data point
x … centroid



Dendrogram

1. How to Represent a Cluster of Many Points?

Non-Euclidean Case

The only “locations” we can talk about are the points themselves

i.e., there is no “average” of two points

1. How to Represent a Cluster of Many Points?

Non-Euclidean Case

The only “locations” we can talk about are the points themselves

i.e., there is no “average” of two points

2. Nearness of Clusters?

Single-link

Similarity of the most similar (single-link) ponts

Complete-link

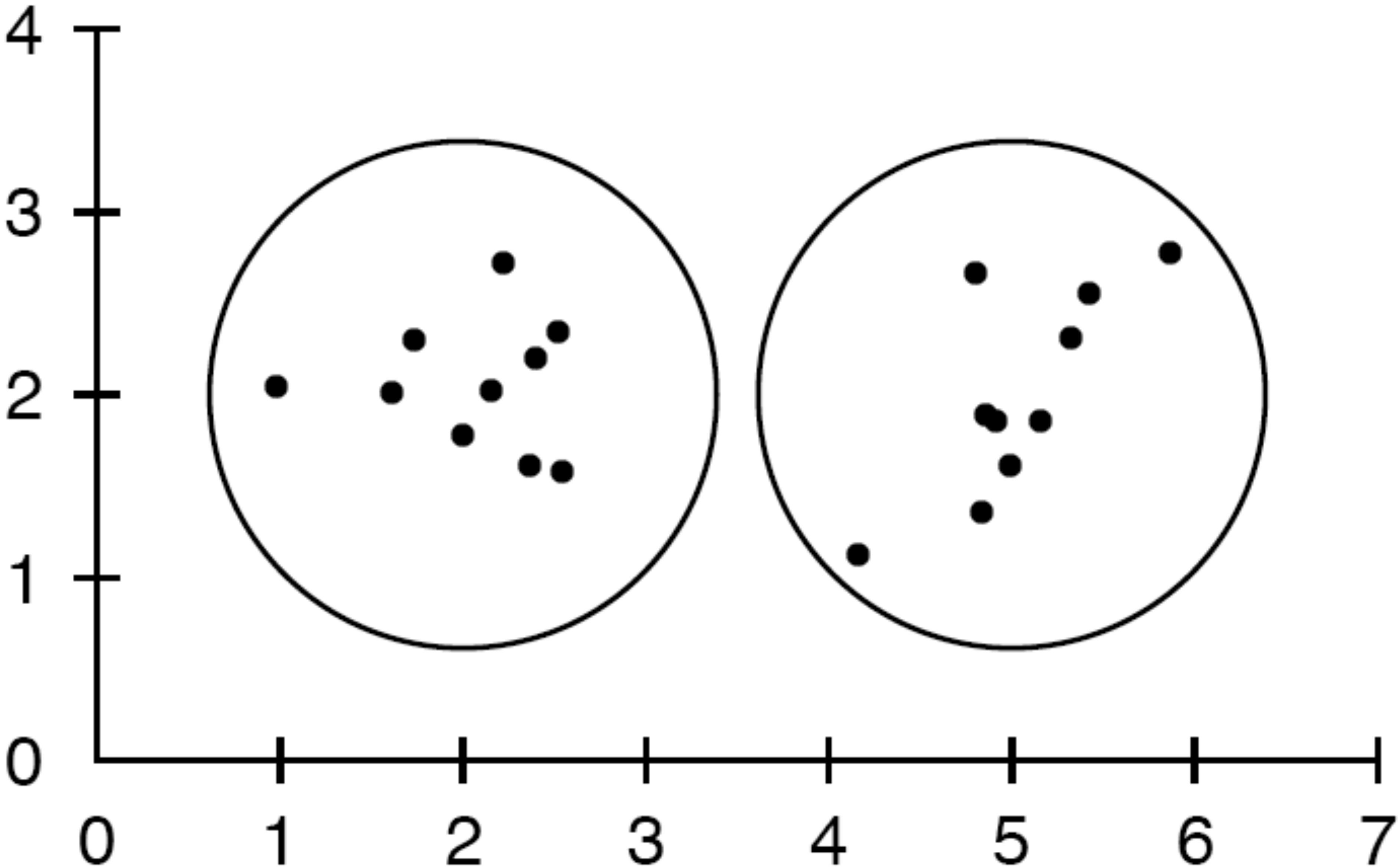
Similarity of the “furthest” points, the least similar

Centroid

Clusters whose centroids (centers of gravity) are the most similar

Average-link

Average similarity between pairs of elements



4
3
2
1
0
0
1
2
3
4
5
6
7

Example: Single Link

Use maximum similarity of pairs:

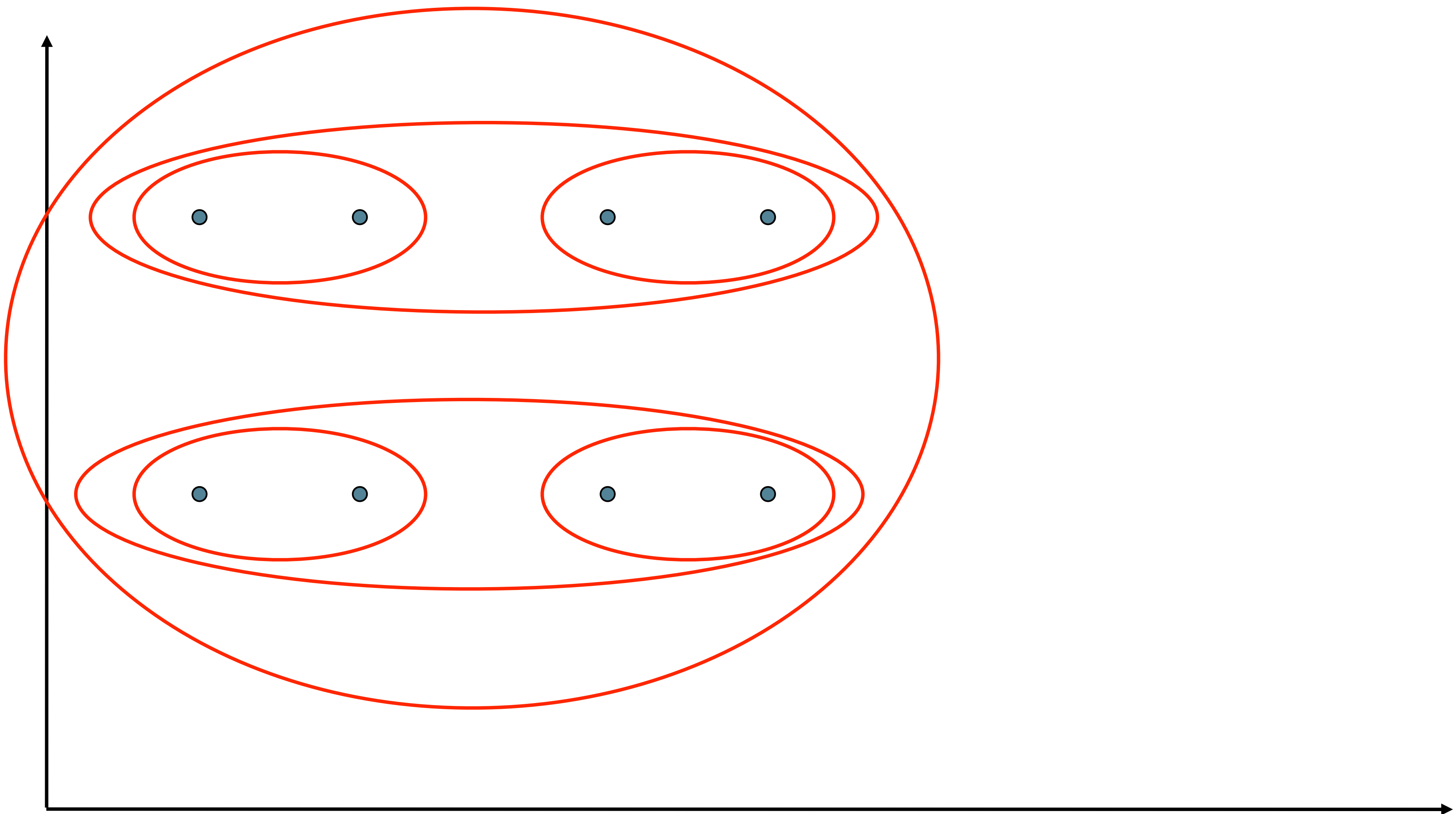
$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

Can result in "straggly" (long and thin) clusters due to chaining effect

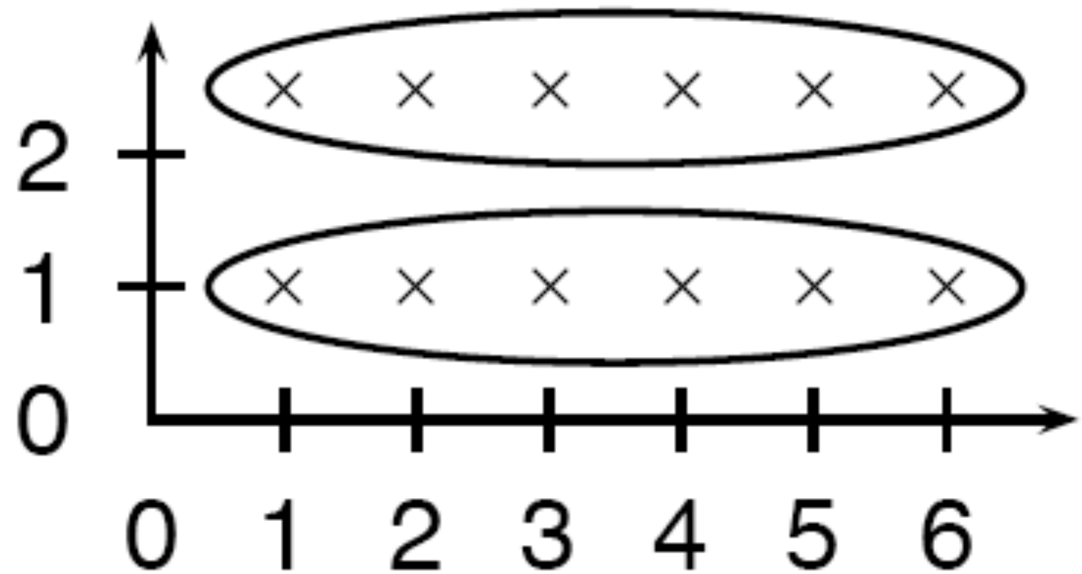
After merging two clusters, the similarity of the resulting cluster to another cluster is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Single Link Example



Single-link: Chaining



Single-link clustering often produces long, straggly clusters. For most applications, these are undesirable

Complete Link Clustering

Use minimum similarity of pairs:

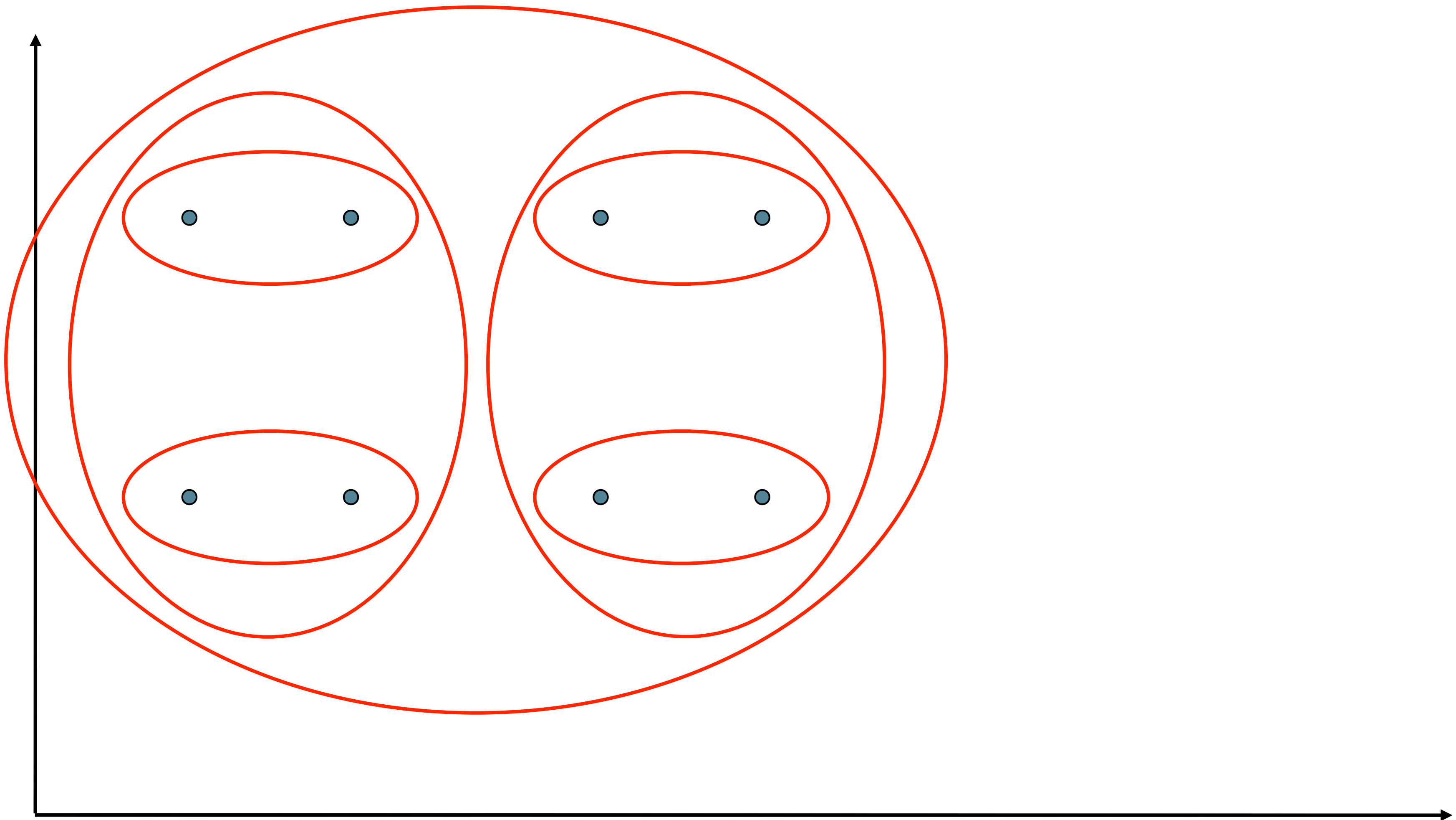
$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

Makes "tighter" spherical clusters that are typically preferable

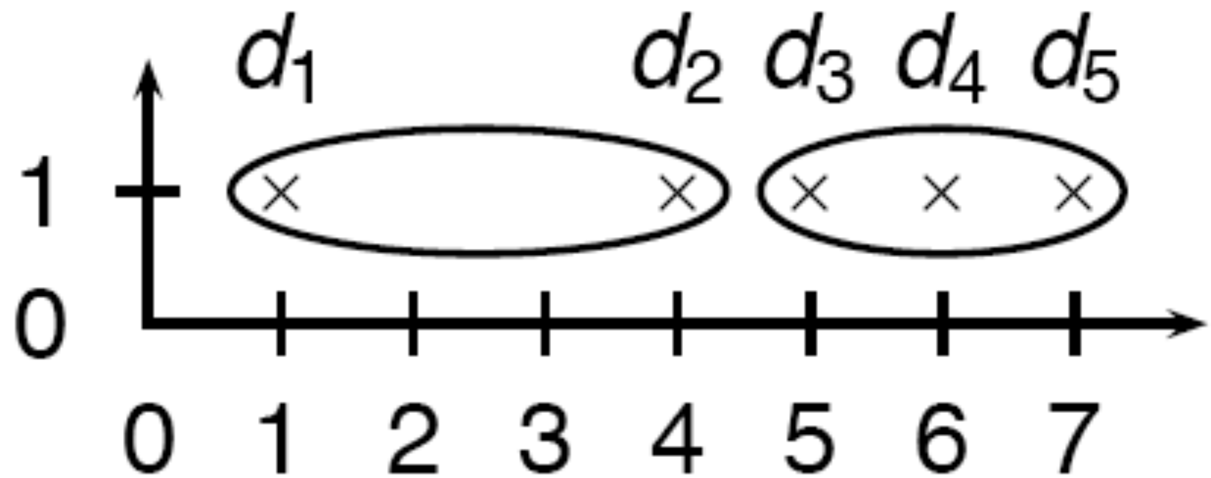
After merging two clusters, the similarity of the resulting cluster to another cluster is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

Complete Link Example



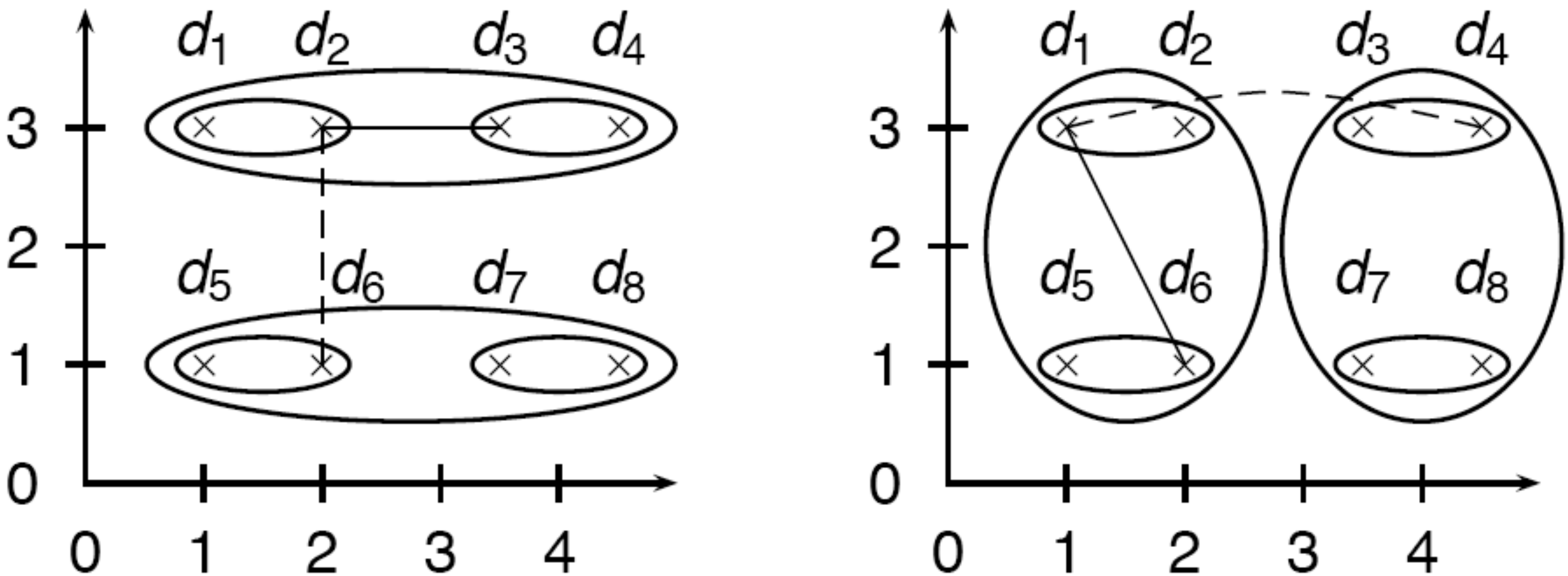
Complete-link: Sensitivity to outliers



1+2e, 4, 5+2e, 6, 7-e

A single outlier can have a large effect on the final outcome of complete-link clustering

Single vs. Complete Link



3. When to Stop Combining Clusters?

Ideas …

Computational Complexity

- In the first iteration, need to compute similarity of all pairs of N initial instances, which is $O(N^2)$.
- In each of the subsequent N–2 merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- In order to maintain an overall $O(N^2)$ performance, computing similarity to each other cluster must be done in constant time.
  - Often $O(N^3)$ if done naively or $O(N^2 \log N)$ if done more cleverly
  - Still too expensive for really big datasets that do not fit in memory

Flat or hierarchical clustering

For high efficiency, use flat clustering (or perhaps bisecting k-means)

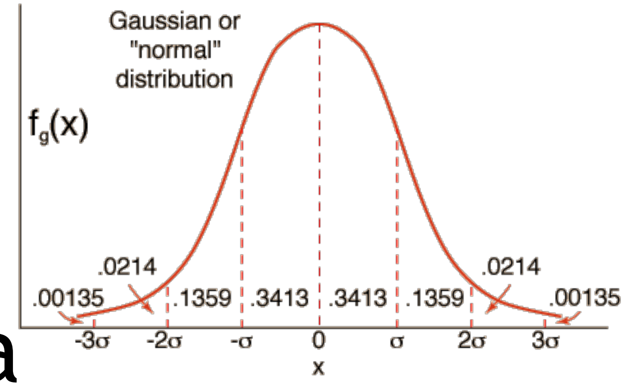
For deterministic results: HAC

When a hierarchical structure is desired: hierarchical algorithm

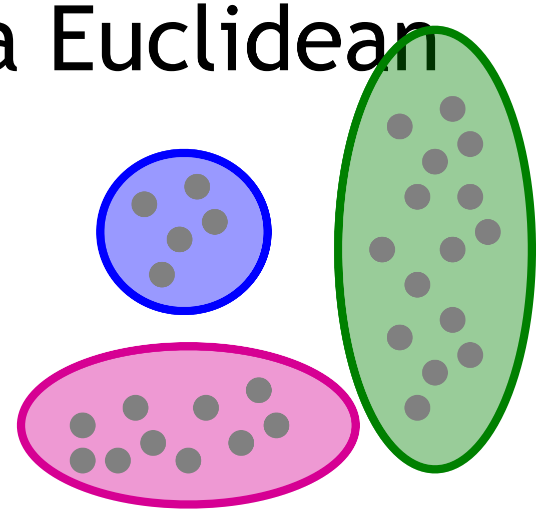
HAC also can be applied if K cannot be predetermined (can start without knowing K)

BFR: K-Means on Large Datasets

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

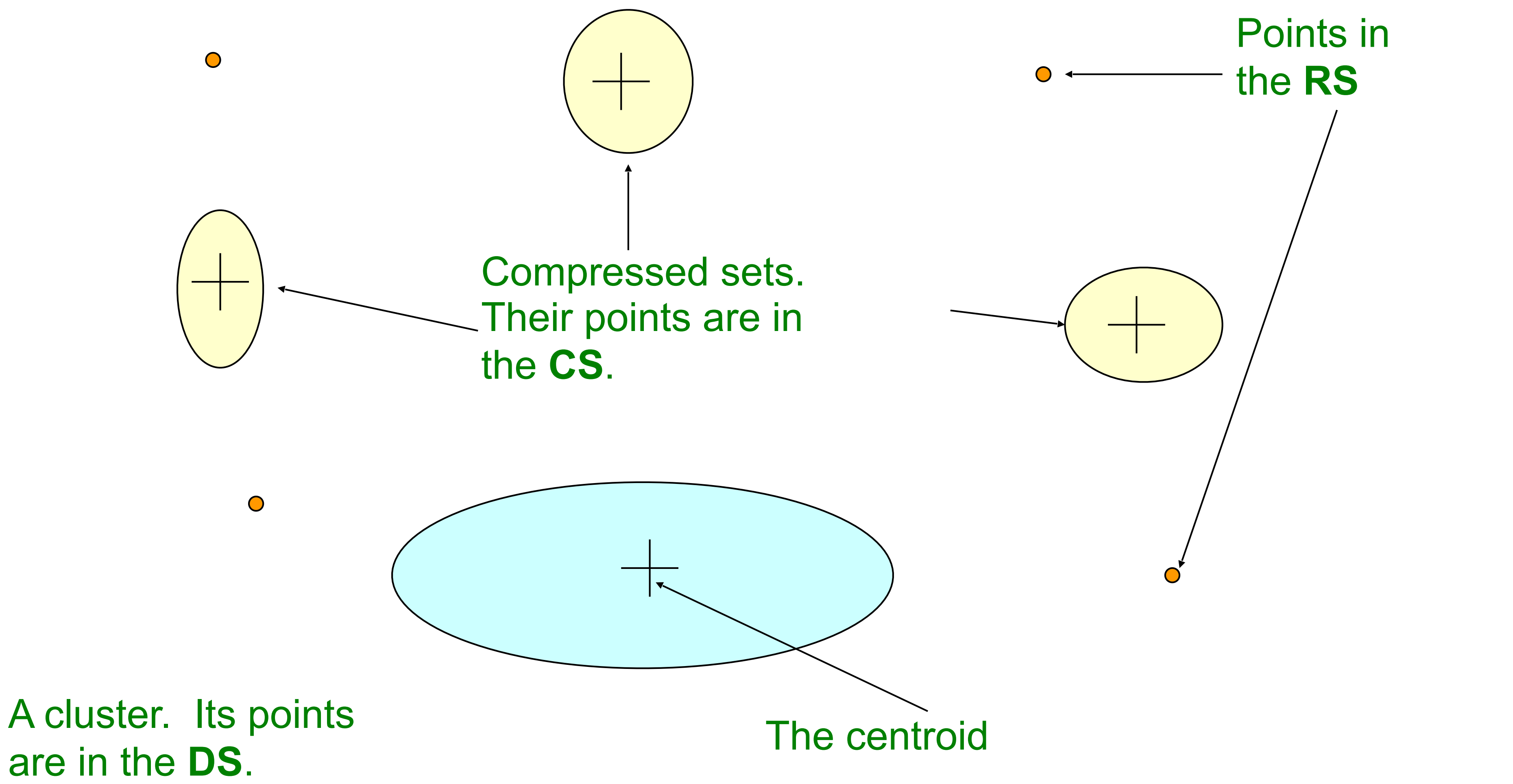
- Points are read from disk one main-memory-full at a time
- **Most points from previous memory loads are summarized by simple statistics**
- To begin, from the initial load we select the initial k centroids by some sensible approach:
  - Take k random points
  - Take a small random sample and cluster 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

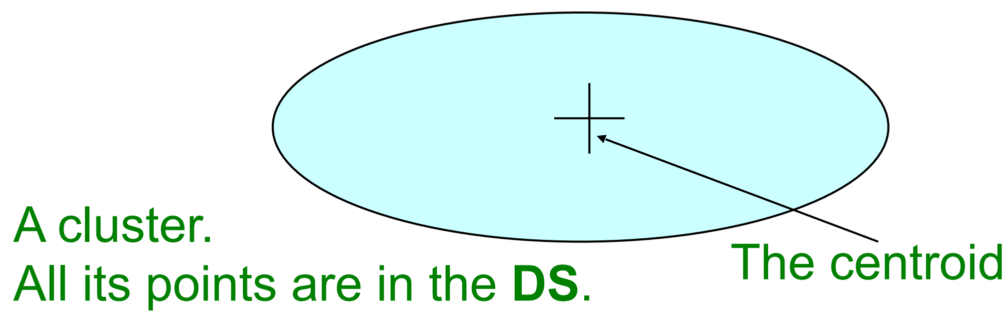


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

For each cluster, the discard set (DS) is <u>summarized</u> by:

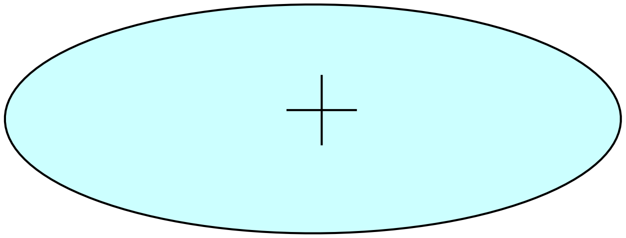
- The number of points, ***N***
- The vector ***SUM***, whose i^{th} component is the sum of the coordinates of the points in the i^{th} dimension
- The vector ***SUMSQ***: i^{th} component = sum of squares of coordinates in i^{th} dimension



Summarizing Points: Comments

- **2d + 1** values represent any size cluster
  - d = number of dimensions
- Average in **each dimension** (the centroid) can be calculated as $\textbf{SUM}_i / N$
  - $\textbf{SUM}_i$ = i^{th} component of SUM
- Variance of a cluster's discard set in dimension i is: $(\textbf{SUMSQ}_i / N) - (\textbf{SUM}_i / N)^2$
  - 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

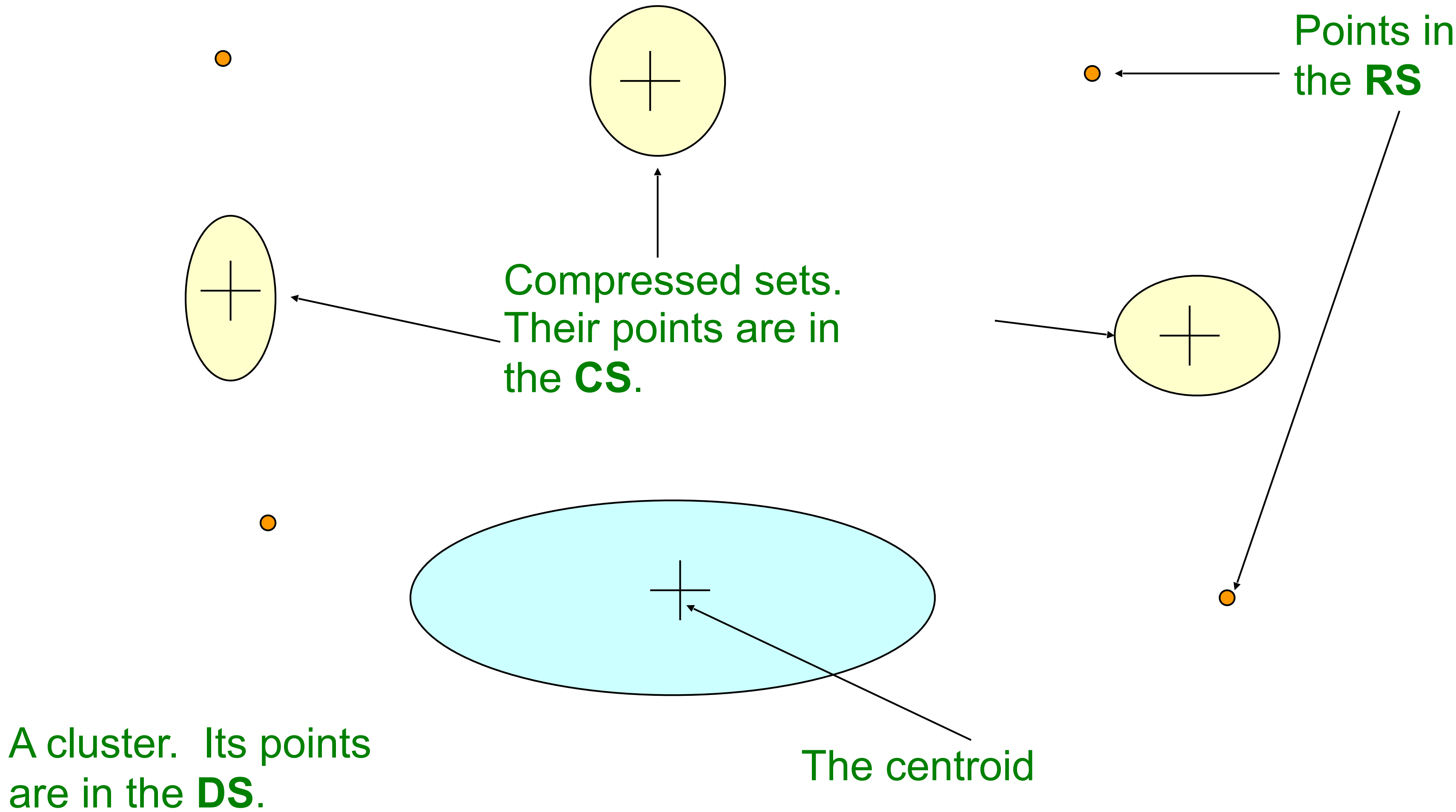
The "Memory-Load" of Points

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

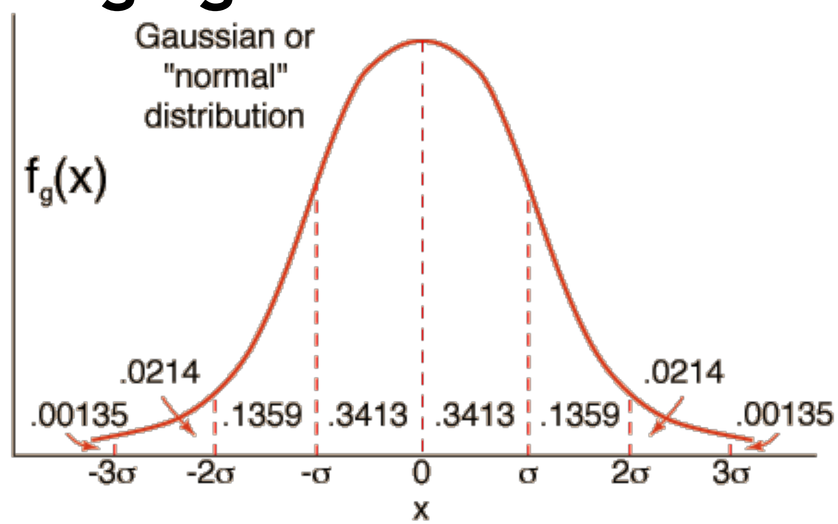
A Few Details…

- 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**

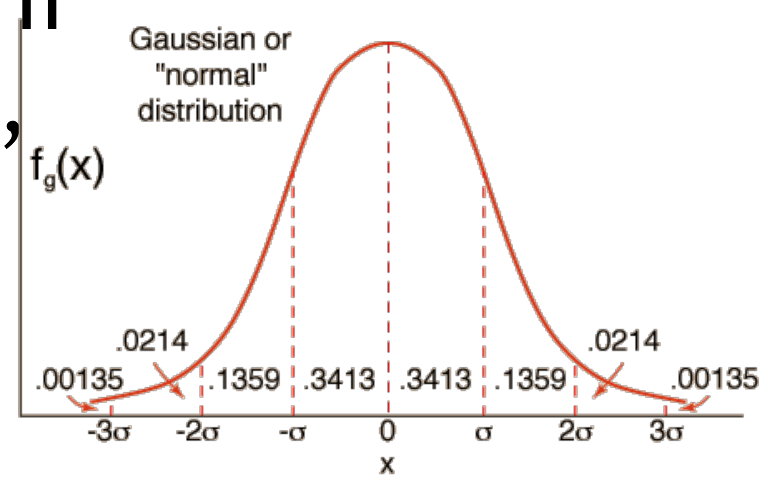
- For point $(x_1, \ldots, x_d)$ and centroid $(c_1, \ldots, c_d)$
  1. Normalize in each dimension: $y_i = (x_i - c_i) / \sigma_i$
  2. Take sum of the squares of the y_i
  3. Take the square root

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

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

Mahalanobis Distance

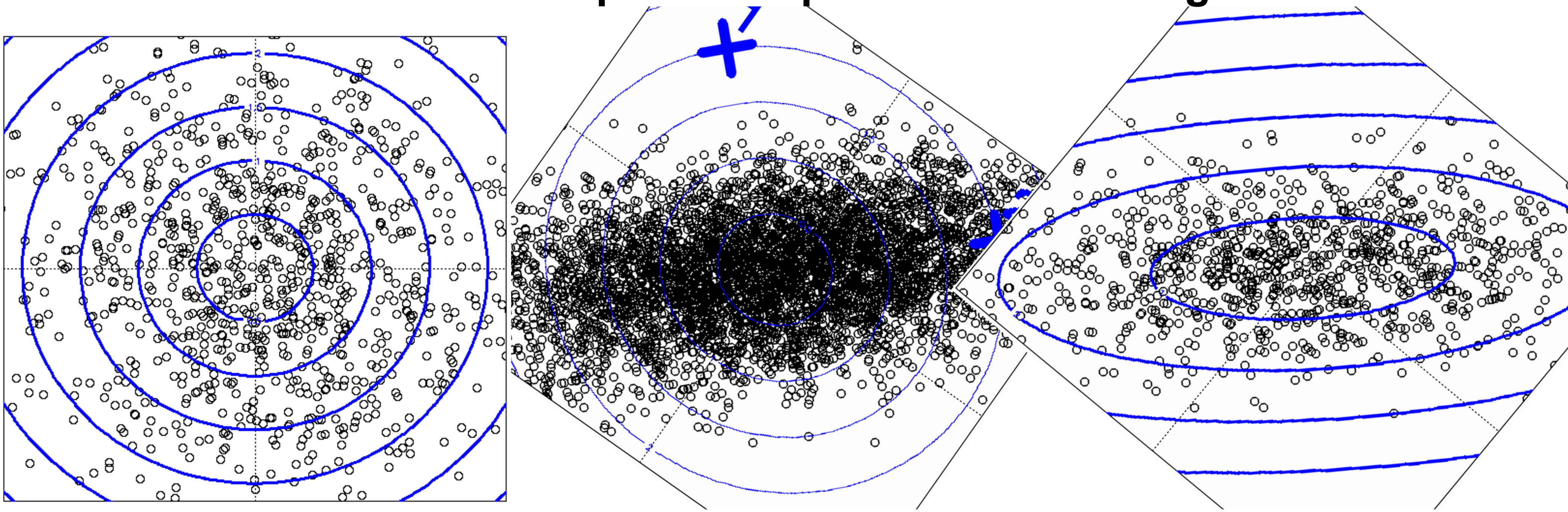
- If clusters are normally distributed in d dimensions, then after transformation, one standard deviation = $\sqrt{d}$
  - i.e., 68% of the points of the cluster will have a Mahalanobis distance $< \sqrt{d}$
- Accept a point for a cluster if its M.D. is < some threshold, e.g. 2 standard deviations



Picture: Equal M.D. Regions

- Euclidean vs. Mahalanobis distance

Contours of equidistant points from the origin



Uniformly distributed points, Euclidean distance

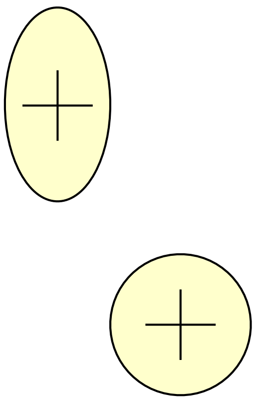
Normally distributed points, Euclidean distance

Normally distributed points, Mahalanobis distance

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



- Many alternatives: Treat dimensions differently, consider density