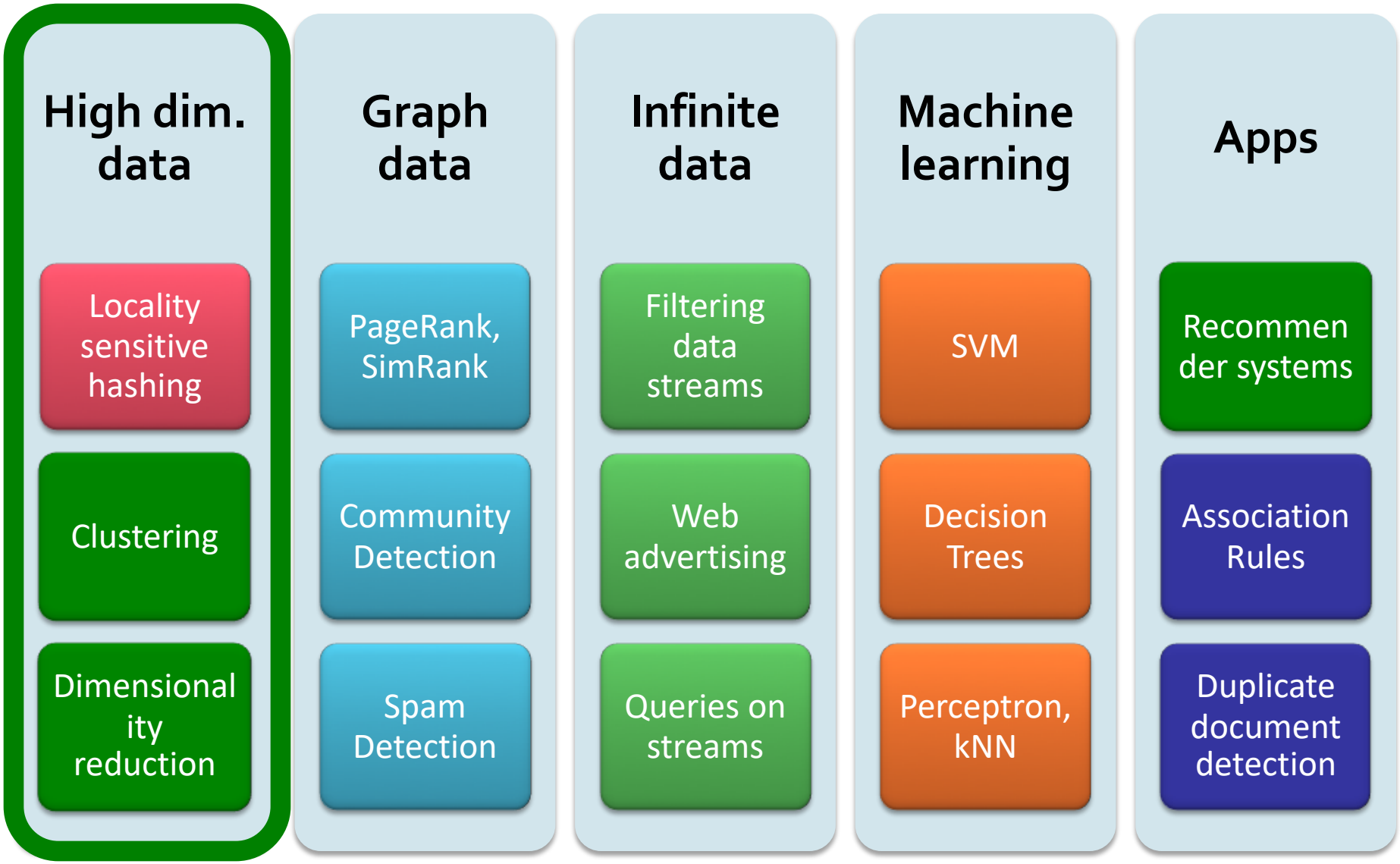
Clustering

Big Data Analytics CSCI 4030

High Dimensional Data



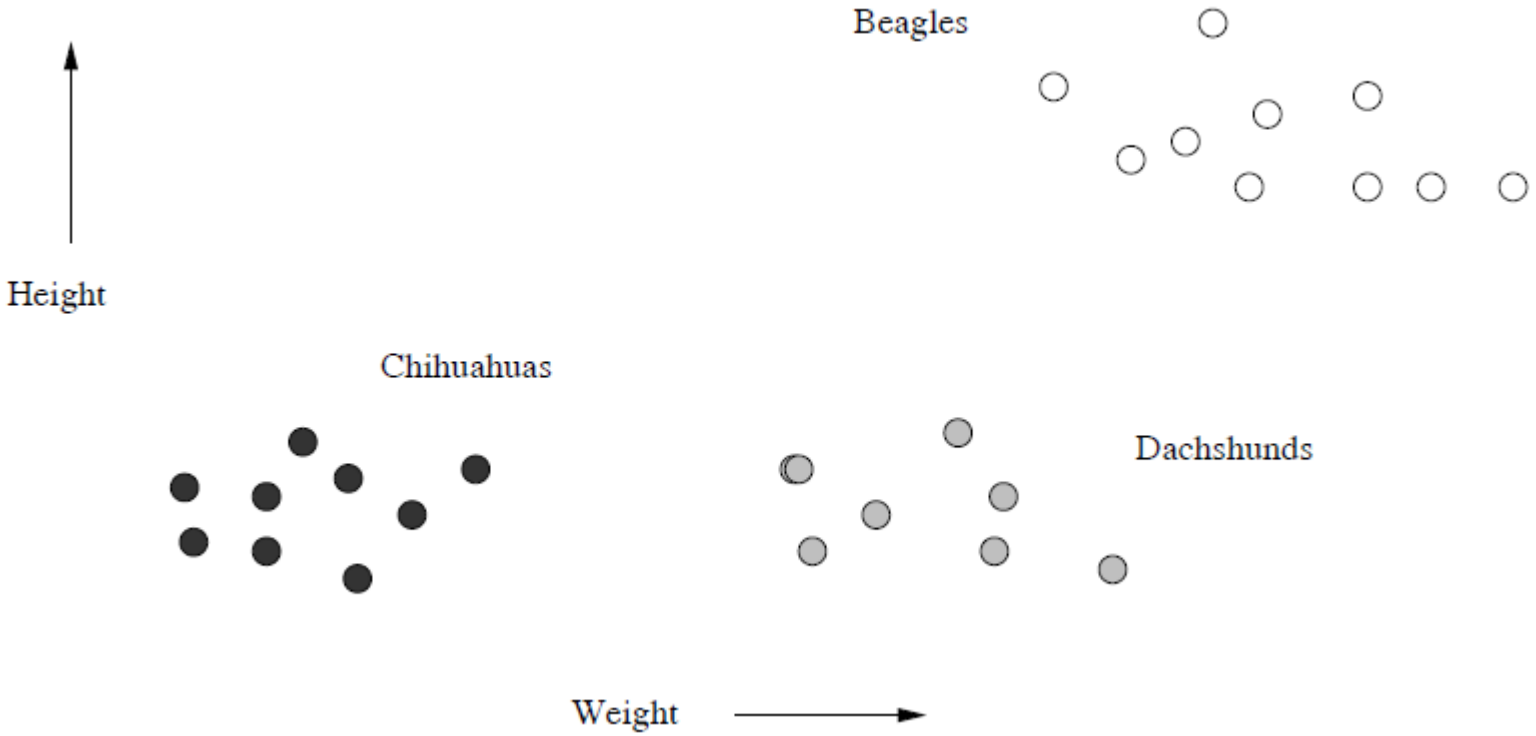
Cloud of Data

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

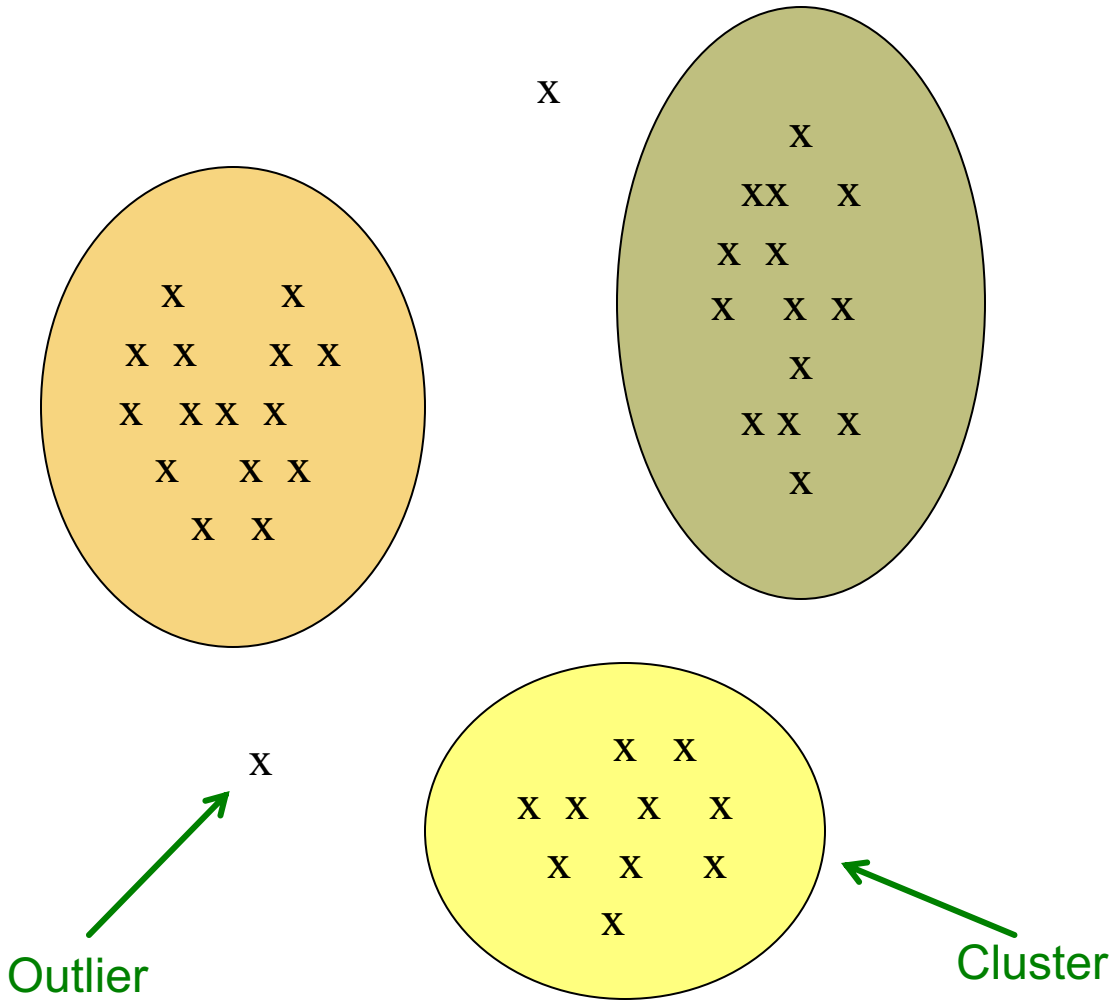
The Problem of Clustering

- 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, …

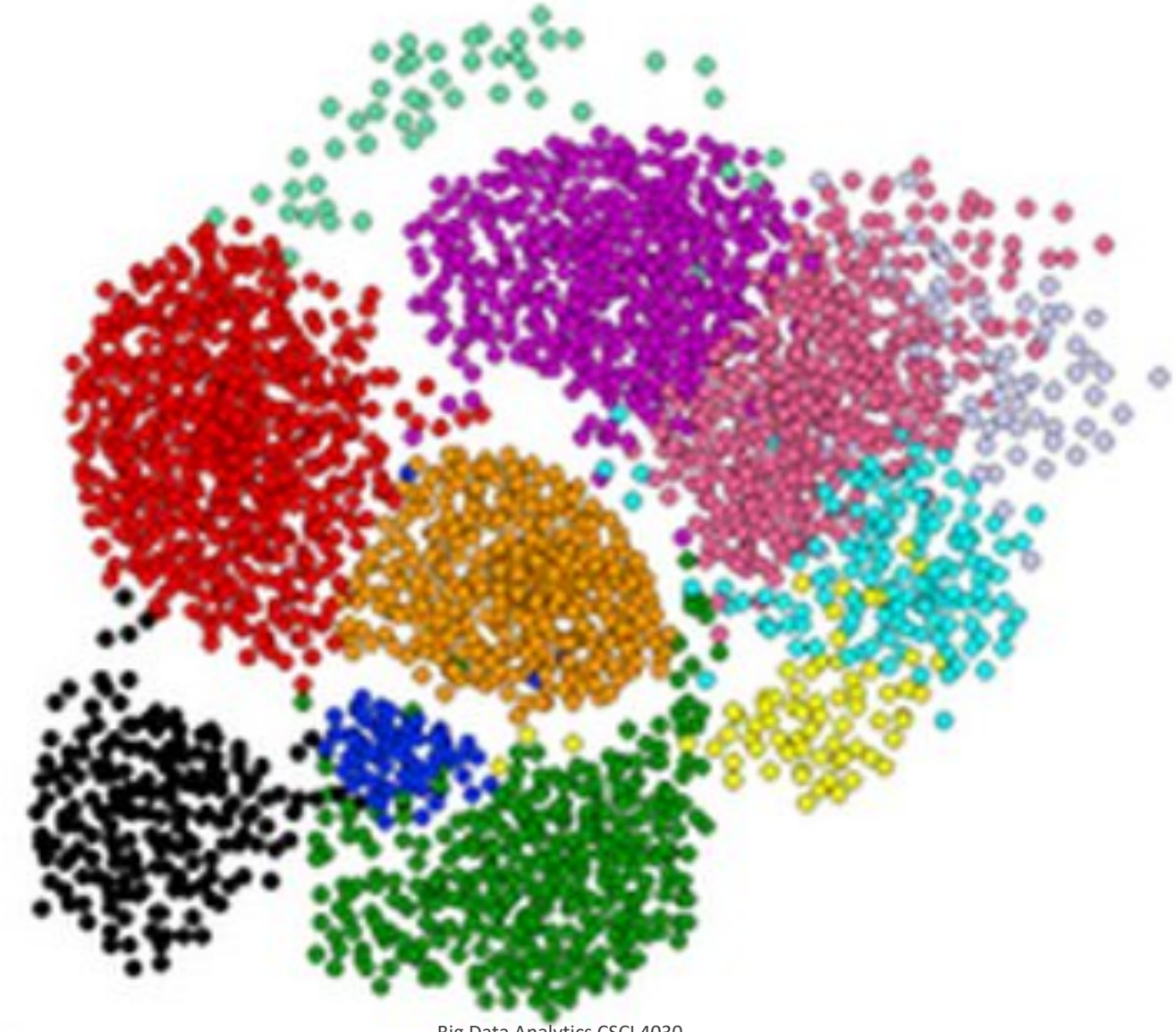
Example: Clusters



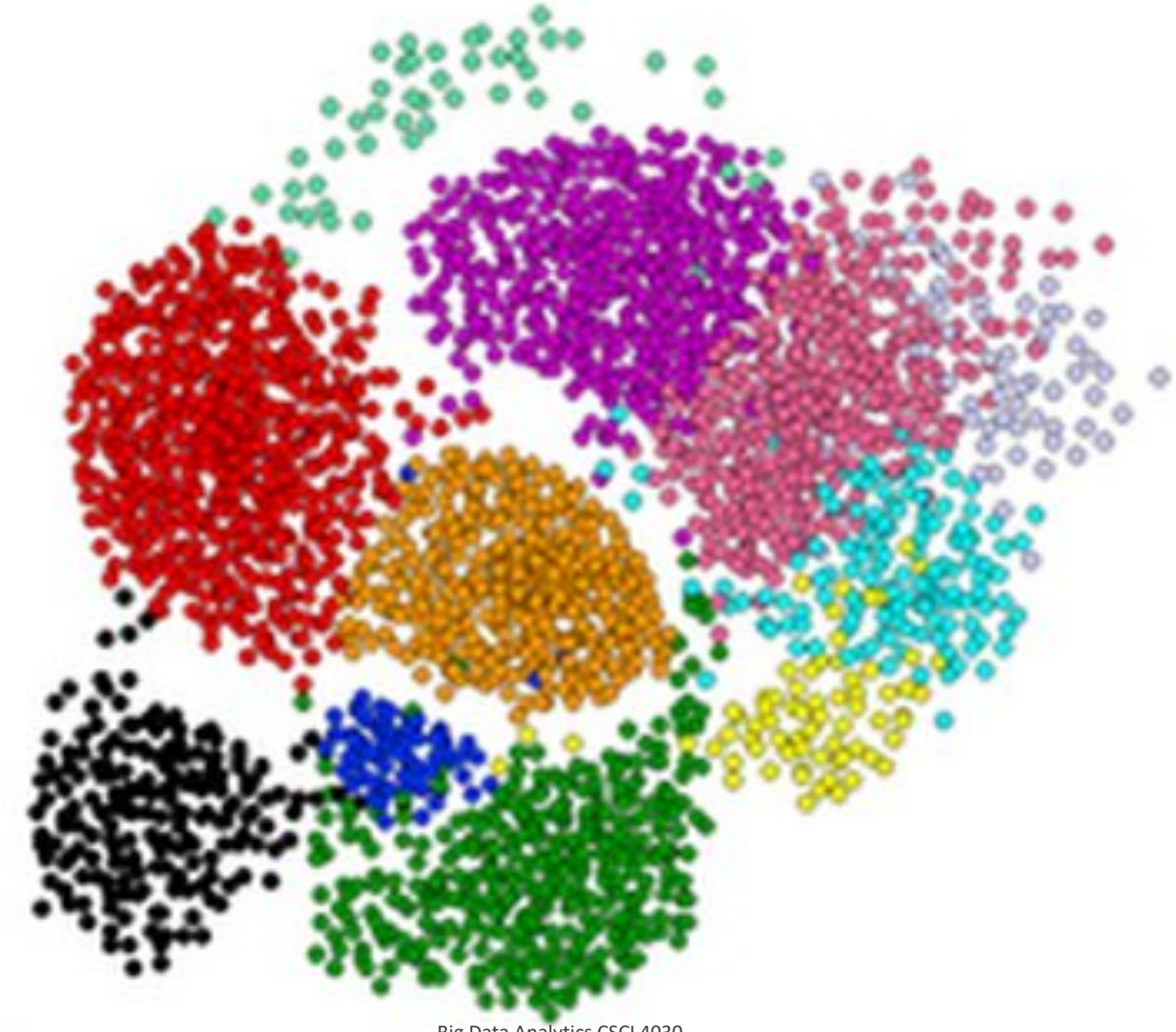
Example: Clusters & Outliers



Is clustering a hard problem?



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:** Many pairs of points are at about the same distance

Clustering Problem: Galaxies

- **A catalog of 2 billion "sky objects" represents objects by their radiation in 7 dimensions (frequency bands)**
- **Problem: Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.**

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

Clustering Problem: Music CDs

Space of all CDs:

- Values in a dimension may be 0 or 1 only
- A CD is a point in this space $(x_1, x_2, \ldots, x_k)$, 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 (in some order) appears in the document

- **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 or shingles:**
  - **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**

Cosine, Jaccard, and Euclidean

The Three Documents and Similarity Metrics

Considering only the 3 words from the above documents: 'sachin', 'dhoni', 'cricket'

Doc Sachin: Wiki page on Sachin Tendulkar
Dhoni - 10
Cricket - 50
Sachin - 200

Doc Dhoni: Wiki page on Dhoni
Dhoni - 400
Cricket - 100
Sachin - 20

Doc Dhoni_Small: Subsection of wiki on Dhoni
Dhoni - 10
Cricket - 5
Sachin - 1

Document - Term Matrix (Word Counts)

Word Counts	*"Dhoni"*	*"Cricket"*
Doc Sachin	10	50
Doc Dhoni	400	100
Doc Dhoni_Small	10	5

Similarity Metrics

Similarity or Distance Metrics	*Total Common Words*	*Euclidean distance*
Doc Sachin & *Doc Dhoni*	10 + 50 + 10 = **70**	**432.4**
Doc Dhoni & *Doc Dhoni_Small*	20 + 10 + 7 = **37**	**204.0**
Doc Sachin & *Doc Dhoni_Small*	10 + 10 + 7 = **27**	**401.85**

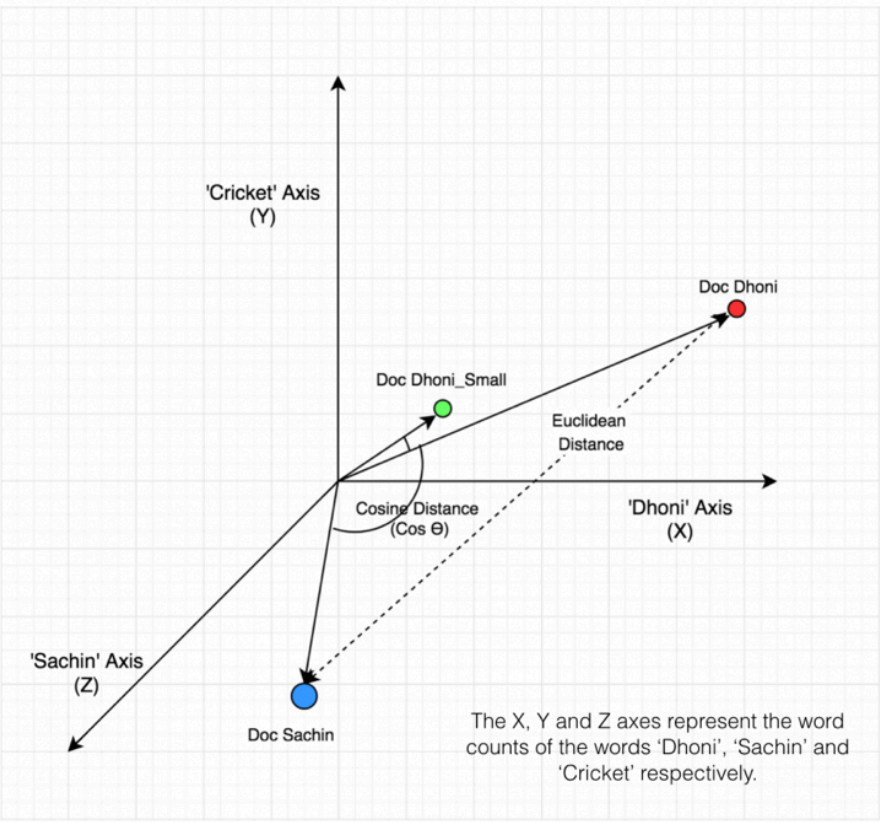
Euclidean Distance

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

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

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

Cosine Similarity

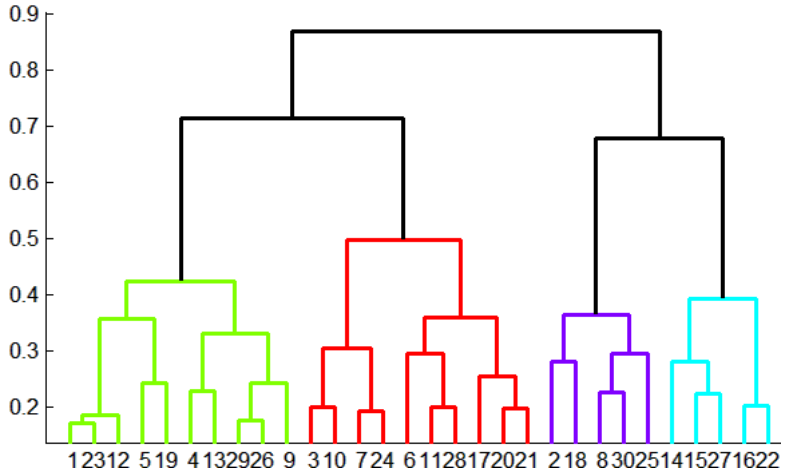


$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

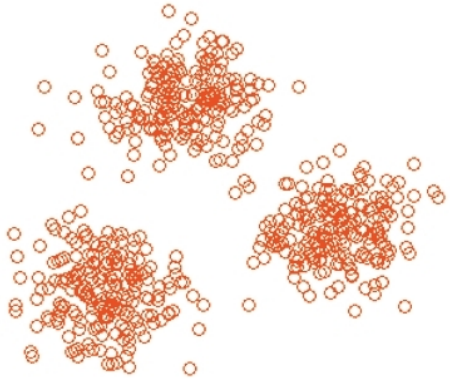
where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

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

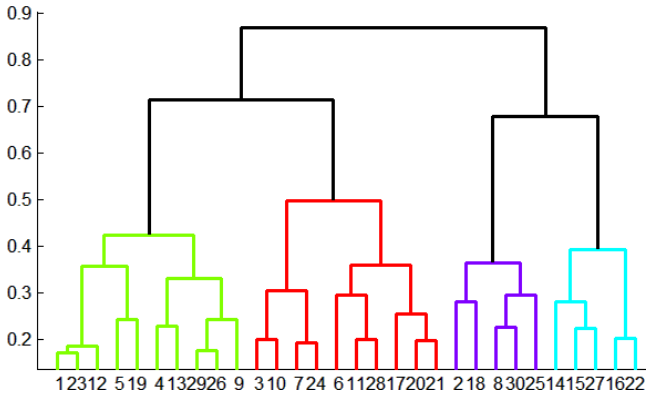


- **Point assignment:**
  - Maintain a set of clusters
  - Points belong to "nearest" cluster



Hierarchical Clustering

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

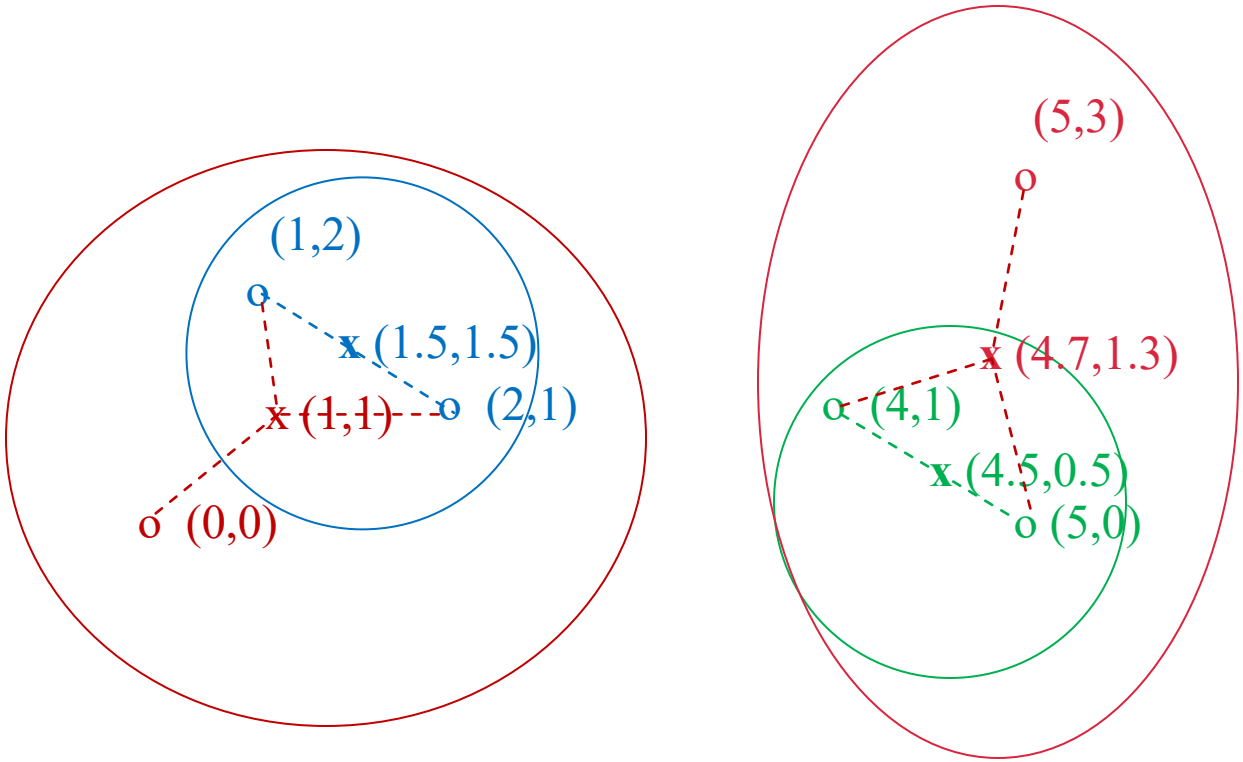


- **Three important questions:**
  - **1)** How do you represent a cluster of more than one point?
  - **2)** How do you determine the "nearness" of clusters?
  - **3)** When to stop combining clusters?

Hierarchical Clustering

- **Key operation: Repeatedly combine two nearest clusters**
- **(1) How to represent a cluster of many points?**
  - **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

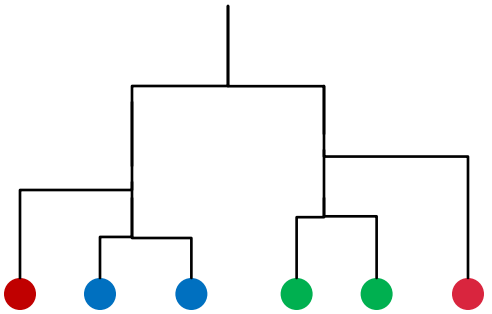
Example: Hierarchical clustering



Data:

o … data point

x … centroid



Dendrogram

And in the Non-Euclidean Case?

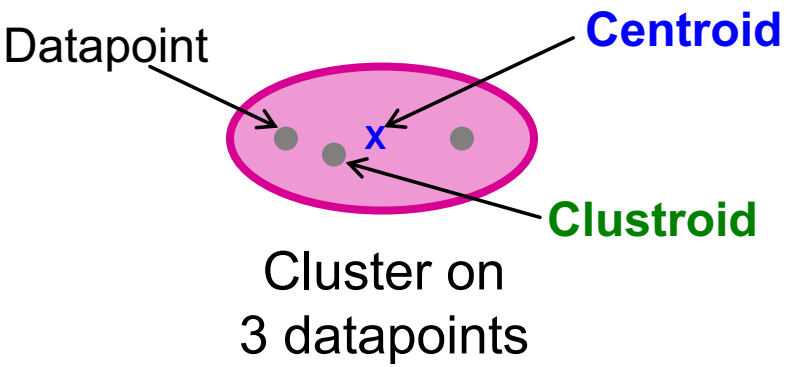
What about the Non-Euclidean case?

- 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



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

Implementation

- **Naïve implementation of hierarchical clustering:**
  - At each step, compute pairwise distances between all pairs of clusters, then merge
  - $O(N^3)$
  - **Too expensive for really big datasets that do not fit in memory**

k-means clustering

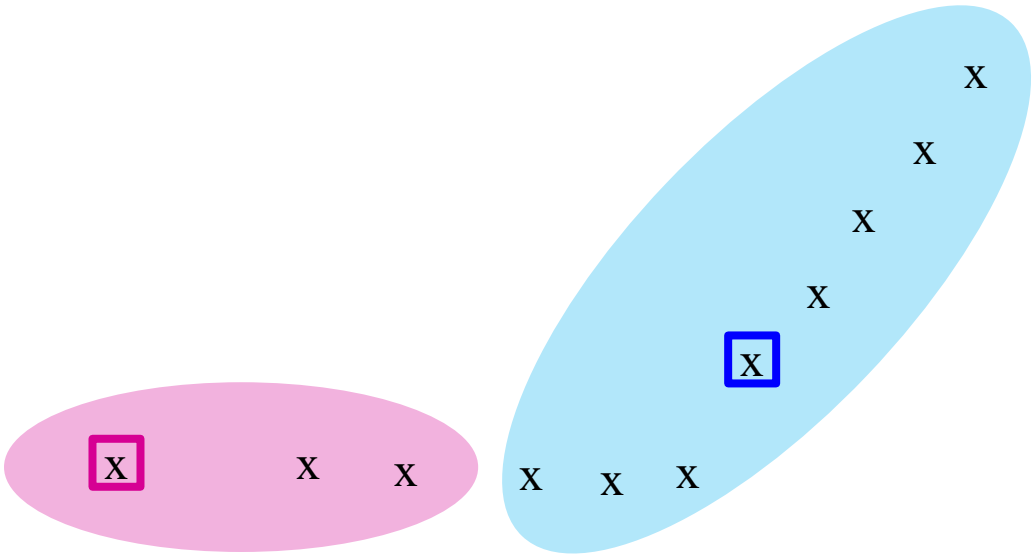
k–means Algorithm(s)

- Assumes Euclidean space/distance
- Start by picking ***k***, the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:**
    - Pick **k** points at random, or
    - 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

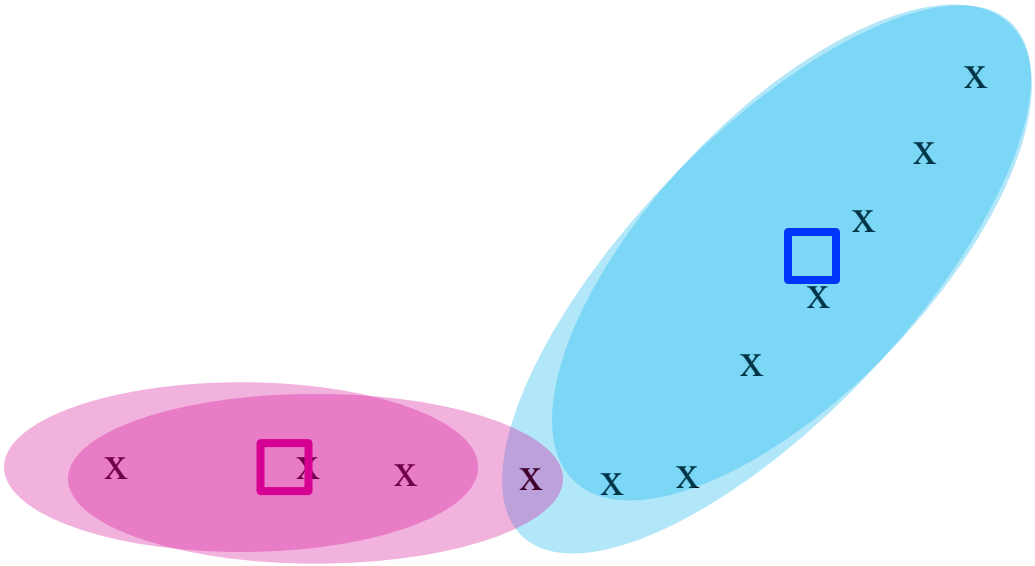
Example: Assigning Clusters



x … data point
□ … centroid

Clusters after round 1

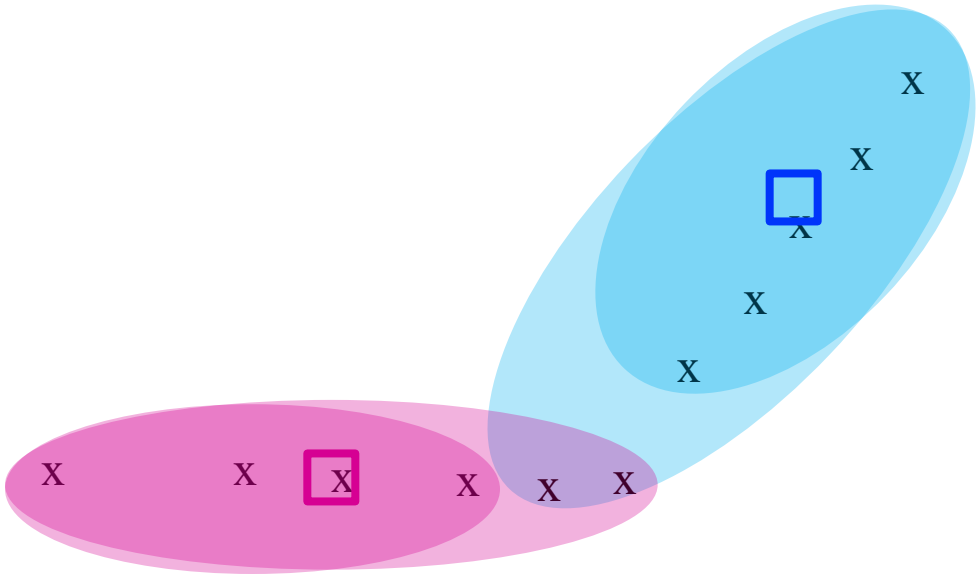
Example: Assigning Clusters



x … data point
□ … centroid

Clusters after round 2

Example: Assigning Clusters



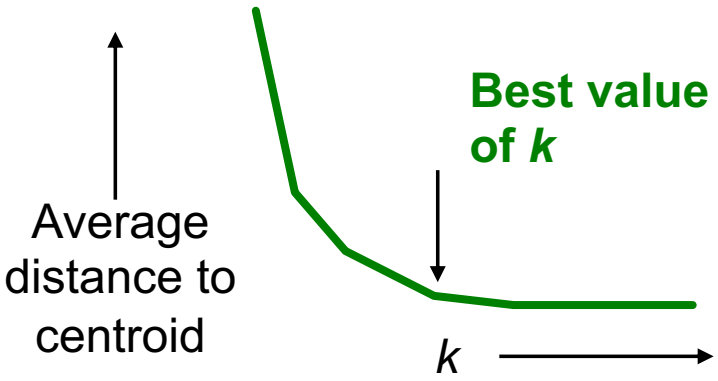
x … data point
□ … centroid

Clusters at the end

Getting the *k* right

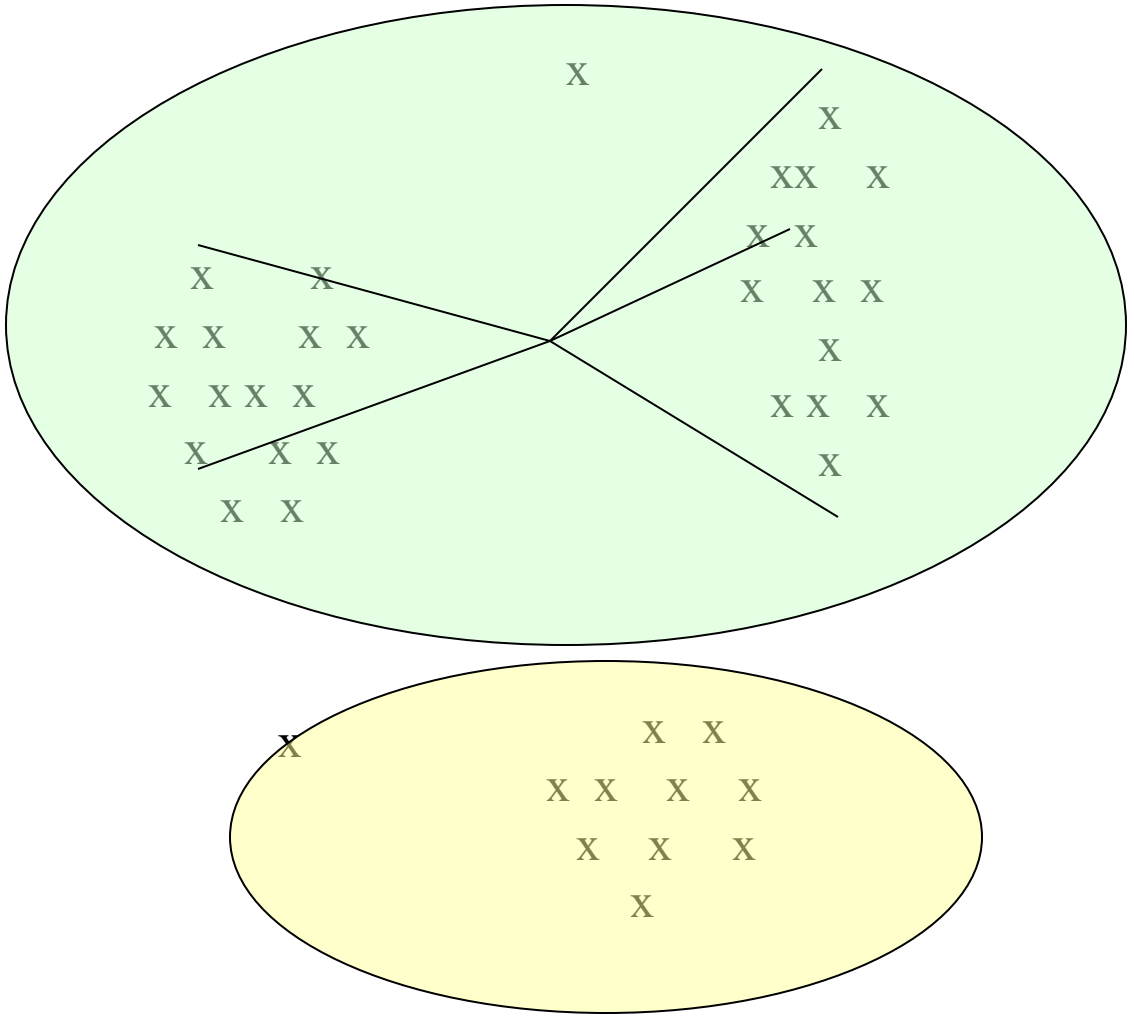
How to select *k*?

- 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



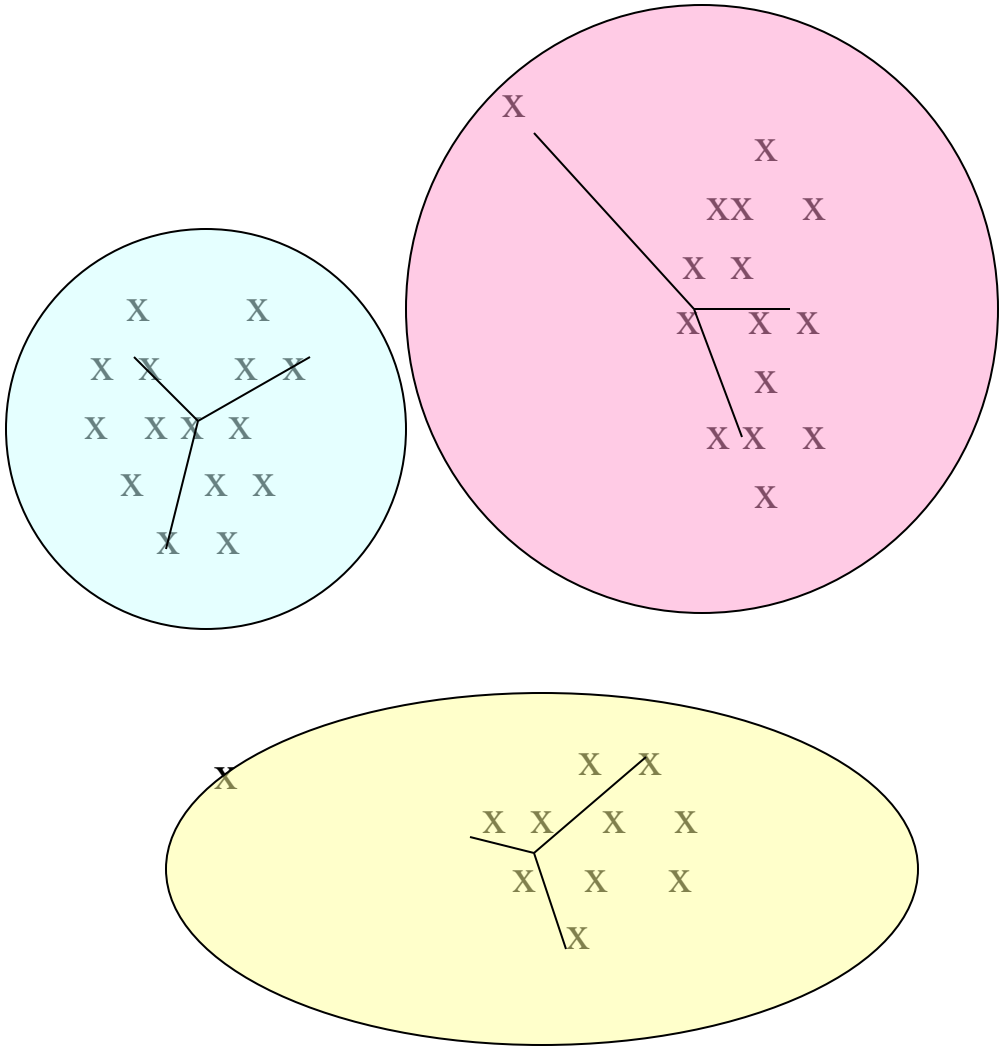
Example: Picking *k*

Too few;
many long
distances
to centroid.



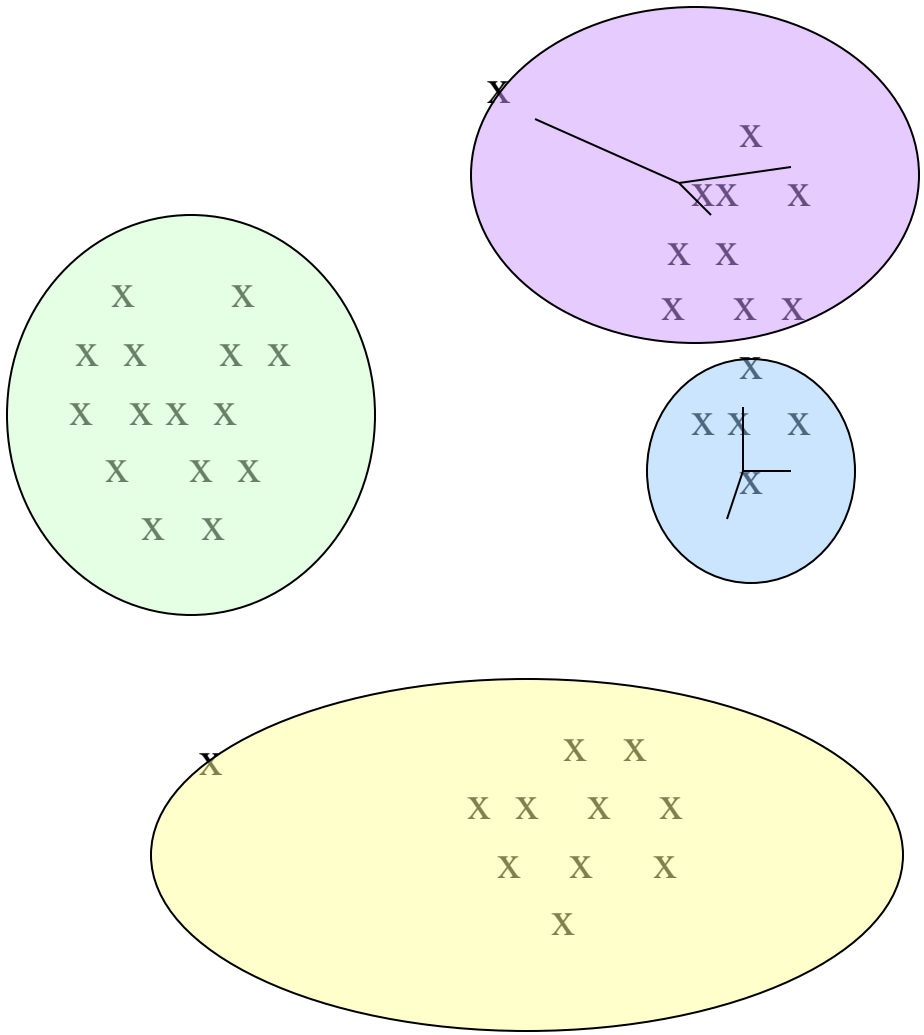
Example: Picking *k*

Just right; distances rather short.



Example: Picking *k*

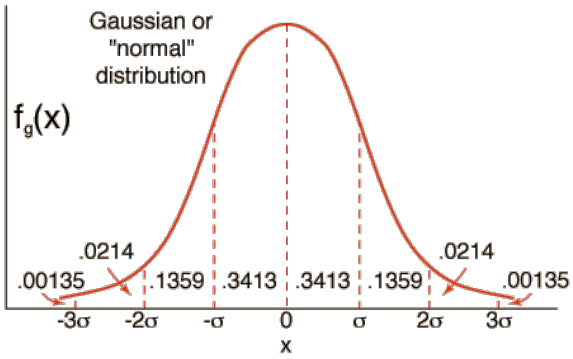
Too many;
little improvement
in average
distance.



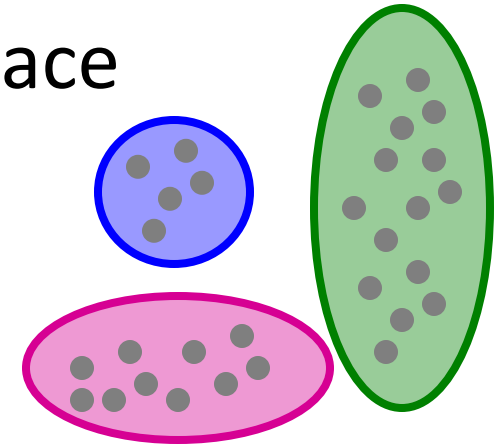
The BFR Algorithm

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
  - **Clusters are axis-aligned ellipses**



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

BFR Algorithm

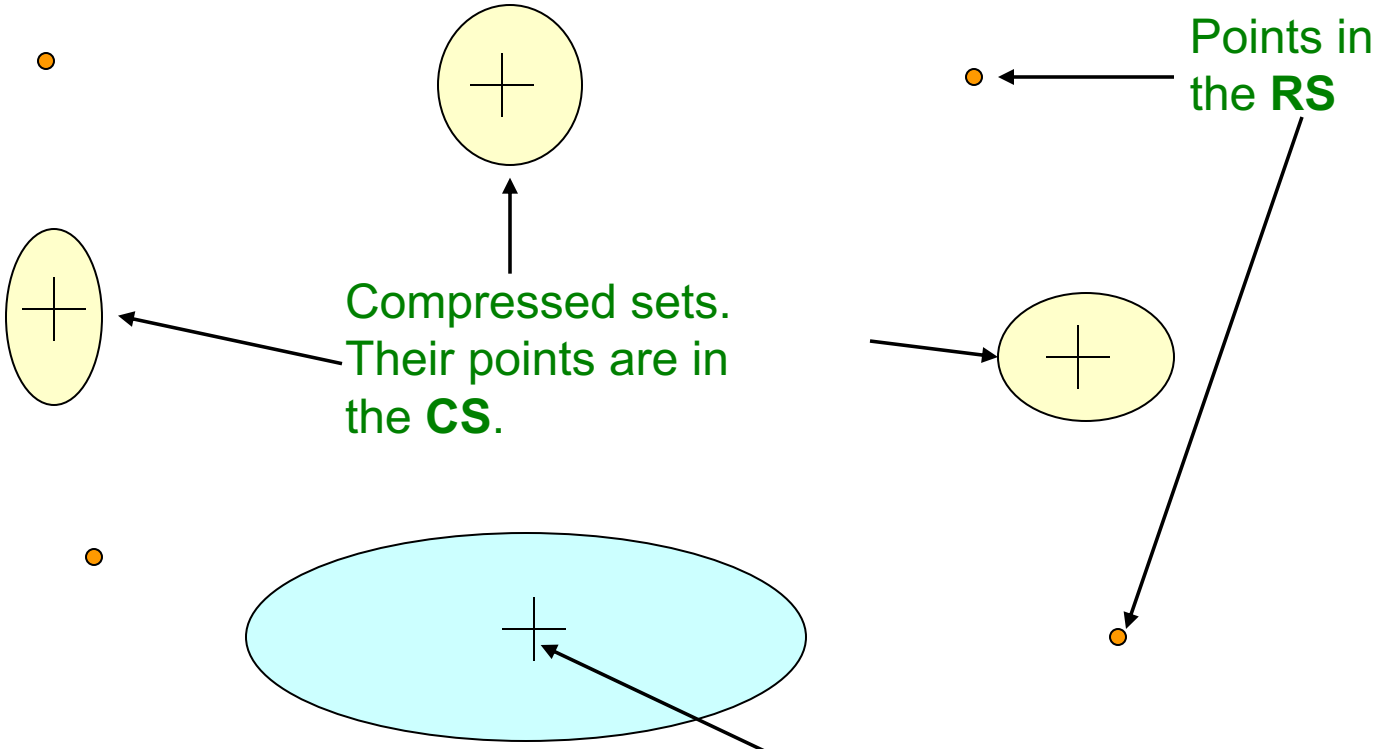
- Points are read from disk into main memory in chunks.
- **Most points from previous memory loads are summarized by simple statistics**
- To begin, from the initial load we select the initial ***k*** centroids by some sensible approach:
  - Take ***k*** random points
  - Take a small random sample and 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. Only summaries are kept in main memory.
- **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. Only summaries are kept in main memory.
- **Retained set (RS):**
  - Isolated points waiting to be assigned to a compression set. Held in main memory exactly as they are.

BFR: "Galaxies" Picture

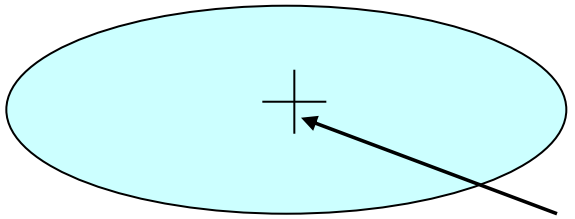


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

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

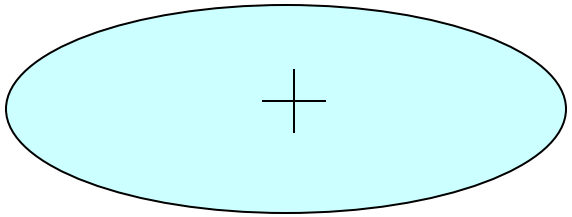


A cluster.
All its points are in the **DS**.

The centroid

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**



The "Memory-Load" of Points

Processing the "Memory-Load" of points (1), i.e. chunk of points:

- **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

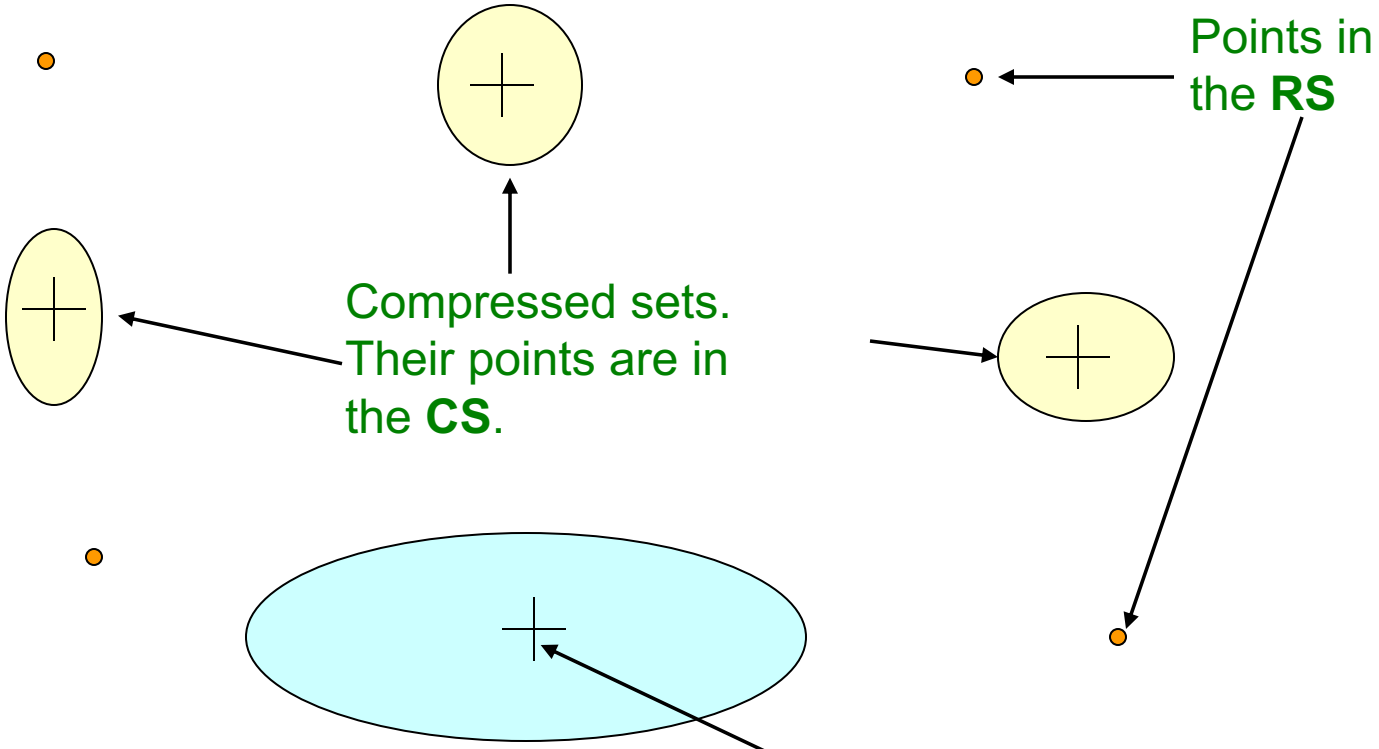
The "Memory-Load" of Points

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 (last chunk of data), merge all compressed sets in the **CS** and all **RS** points into their nearest cluster (or treat them as outliers)

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 DS 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)**
- **e.g, BFR suggests:**
  - The **Mahalanobis distance** is less than a threshold

Mahalanobis Distance

- **Normalized Euclidean distance from centroid**

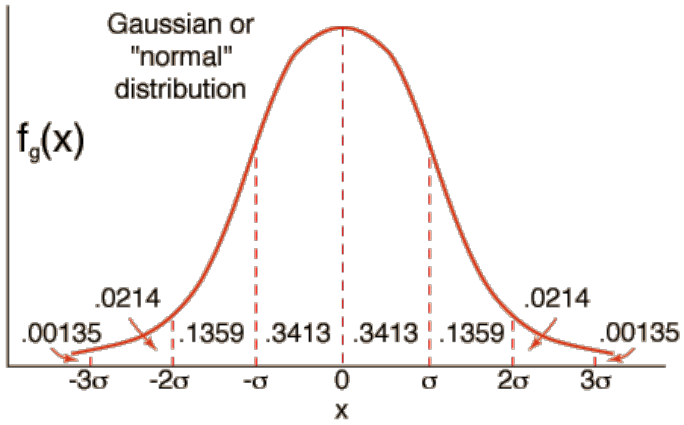
- For point $(x_1, \ldots, x_d)$ and centroid $(c_1, \ldots, c_d)$
  1. Normalize in each dimension: $y_i = (x_i - c_i) / \sigma_i$
  2. Take sum of the squares of the y_i
  3. Take the square root

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

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

Mahalanobis Distance

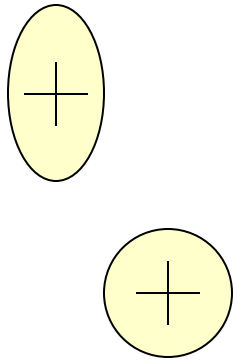
- Accept a point for a cluster if its M.D. is **<** some threshold, e.g. **2** standard deviations



Should 2 CS clusters be combined?

Q2) Should 2 CS subclusters be combined?

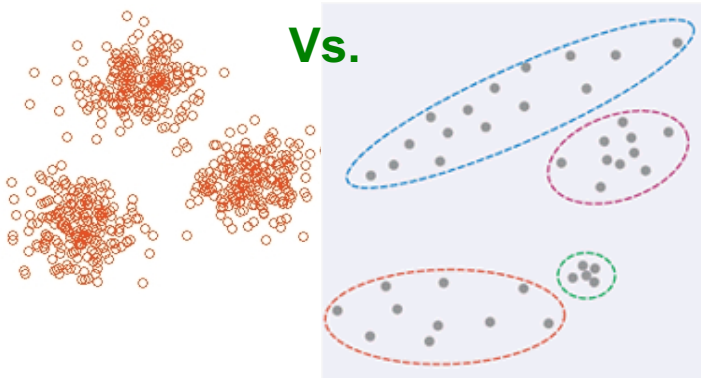
- Compute the variance of the combined subcluster
  - ***N***, ***SUM***, and ***SUMSQ*** allow us to make that calculation quickly
- Combine if the combined variance is below some threshold



The CURE Algorithm

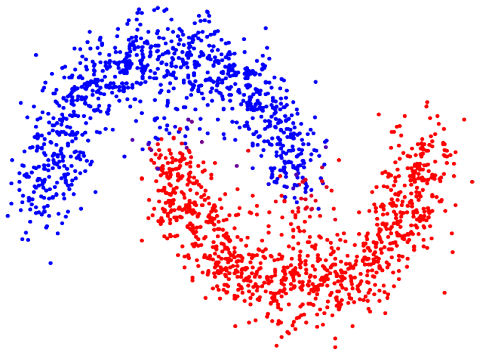
Extension of *k*-means to clusters of arbitrary shapes

The CURE Algorithm

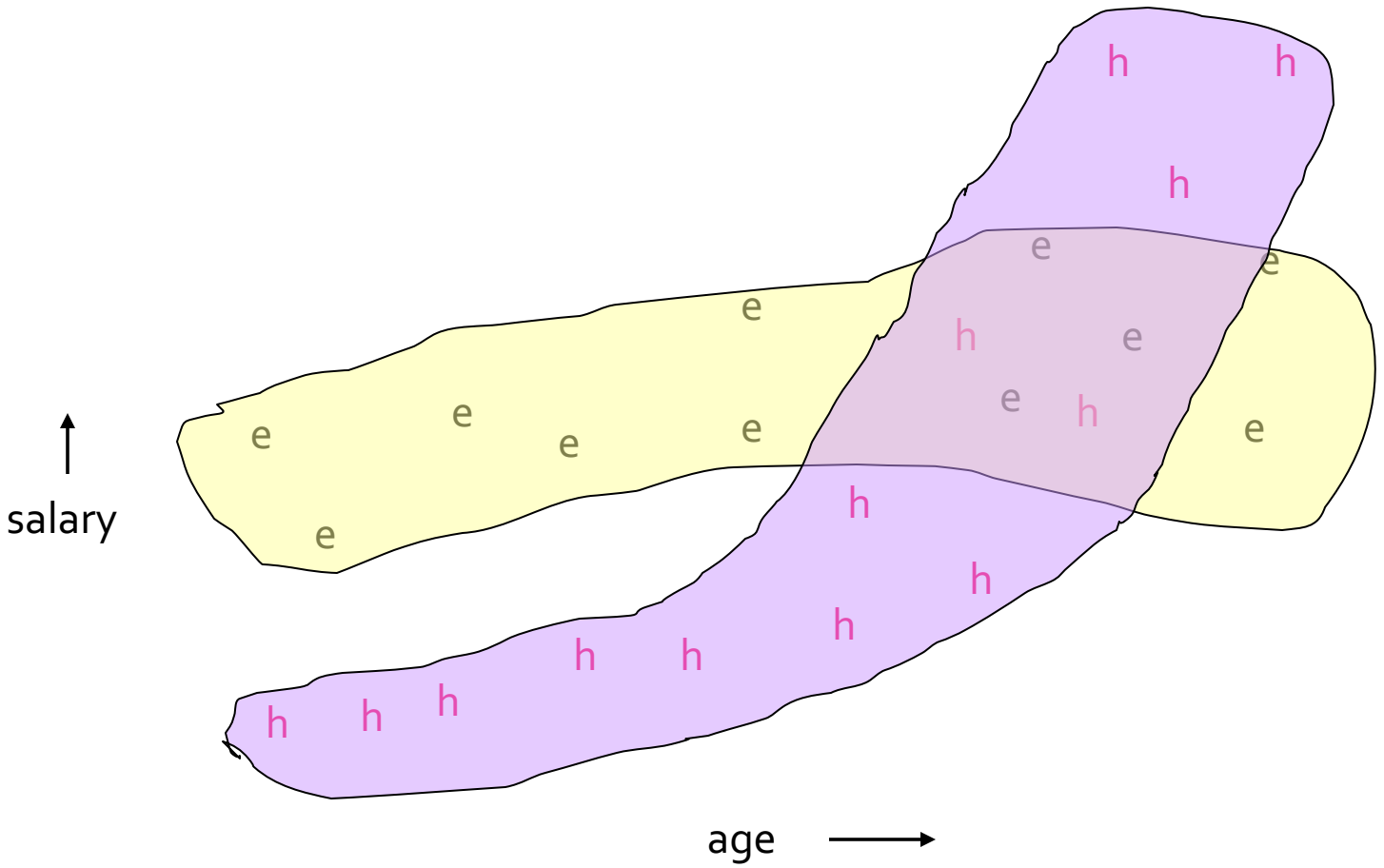


- **Problem with BFR:**
  - Assumes clusters are normally distributed in each dimension
  - And axes are fixed – ellipses at an angle are ***not OK***

- **CURE (Clustering Using REpresentatives):**
  - Assumes a Euclidean distance
  - Allows clusters to assume any shape
  - **Uses a collection of representative points to represent clusters**



Example: Salaries

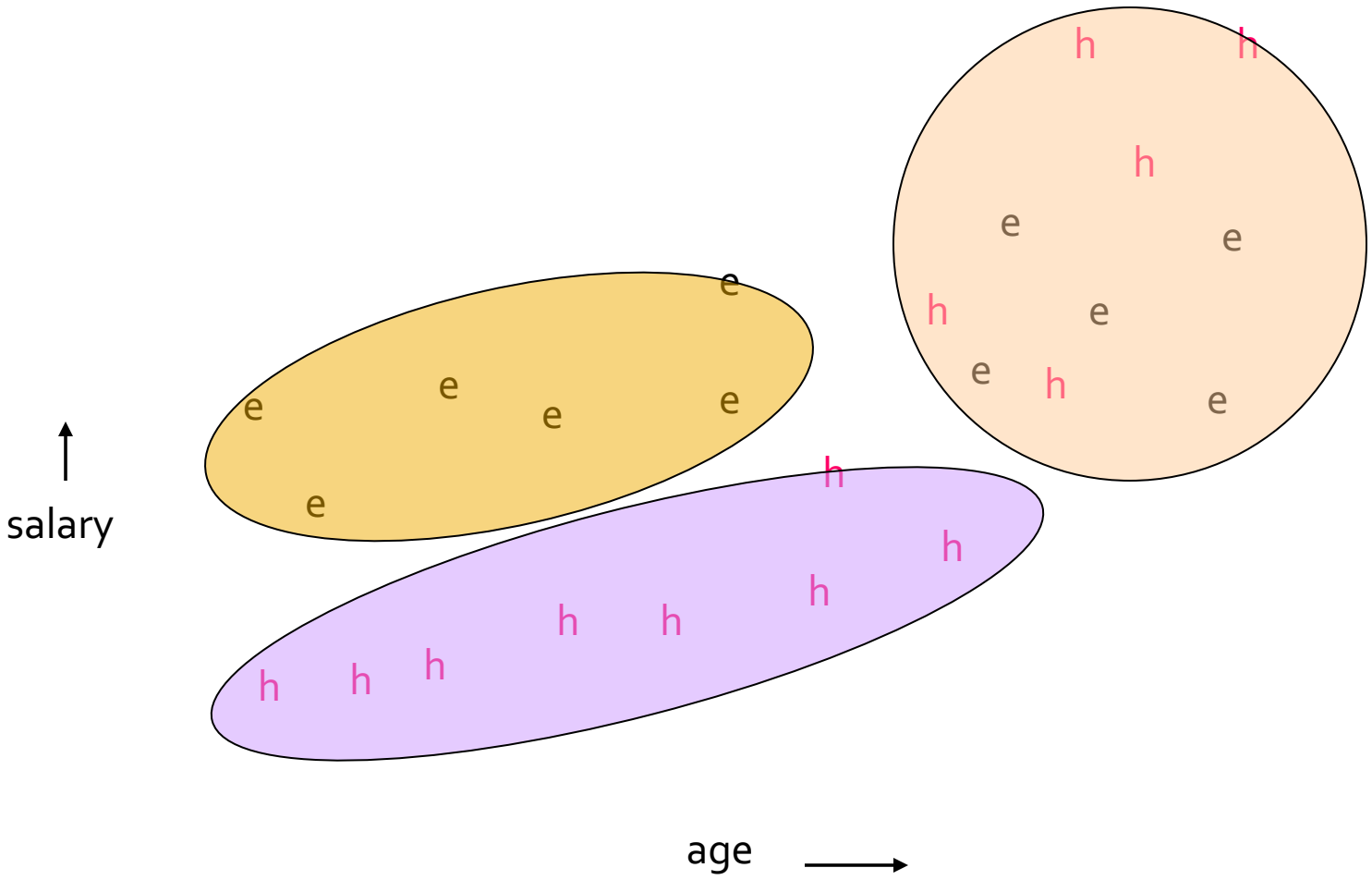


Starting CURE

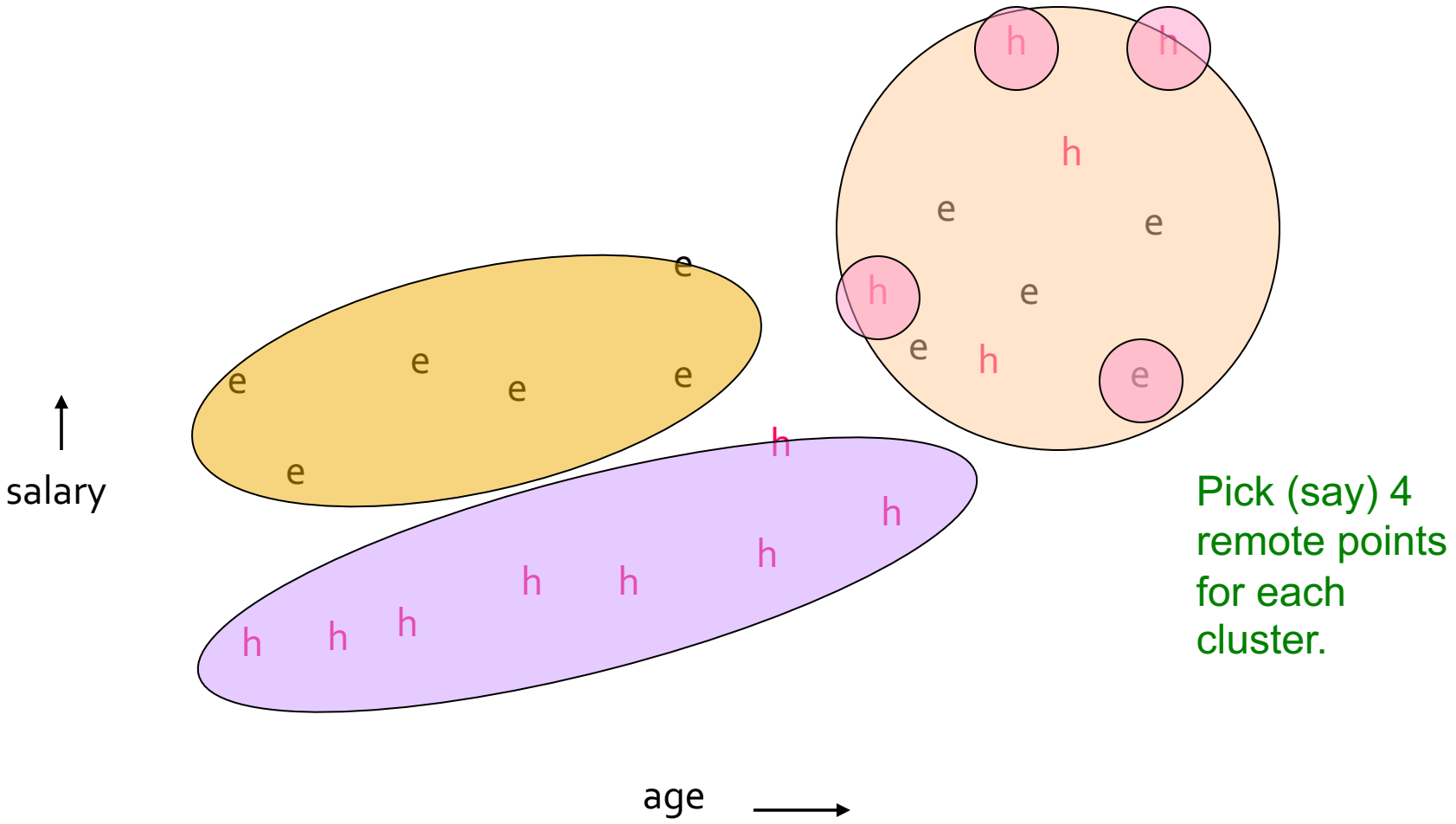
2 Pass algorithm. Pass 1:

- **0) Pick a random sample of points that fit in main memory**
- **1) Initial clusters:**
  - Cluster these points hierarchically – group nearest points/clusters
- **2) Pick representative points:**
  - For each cluster, pick a sample of points, as dispersed as possible
  - From the sample, pick representatives by moving them (say) 20% toward the centroid of the cluster

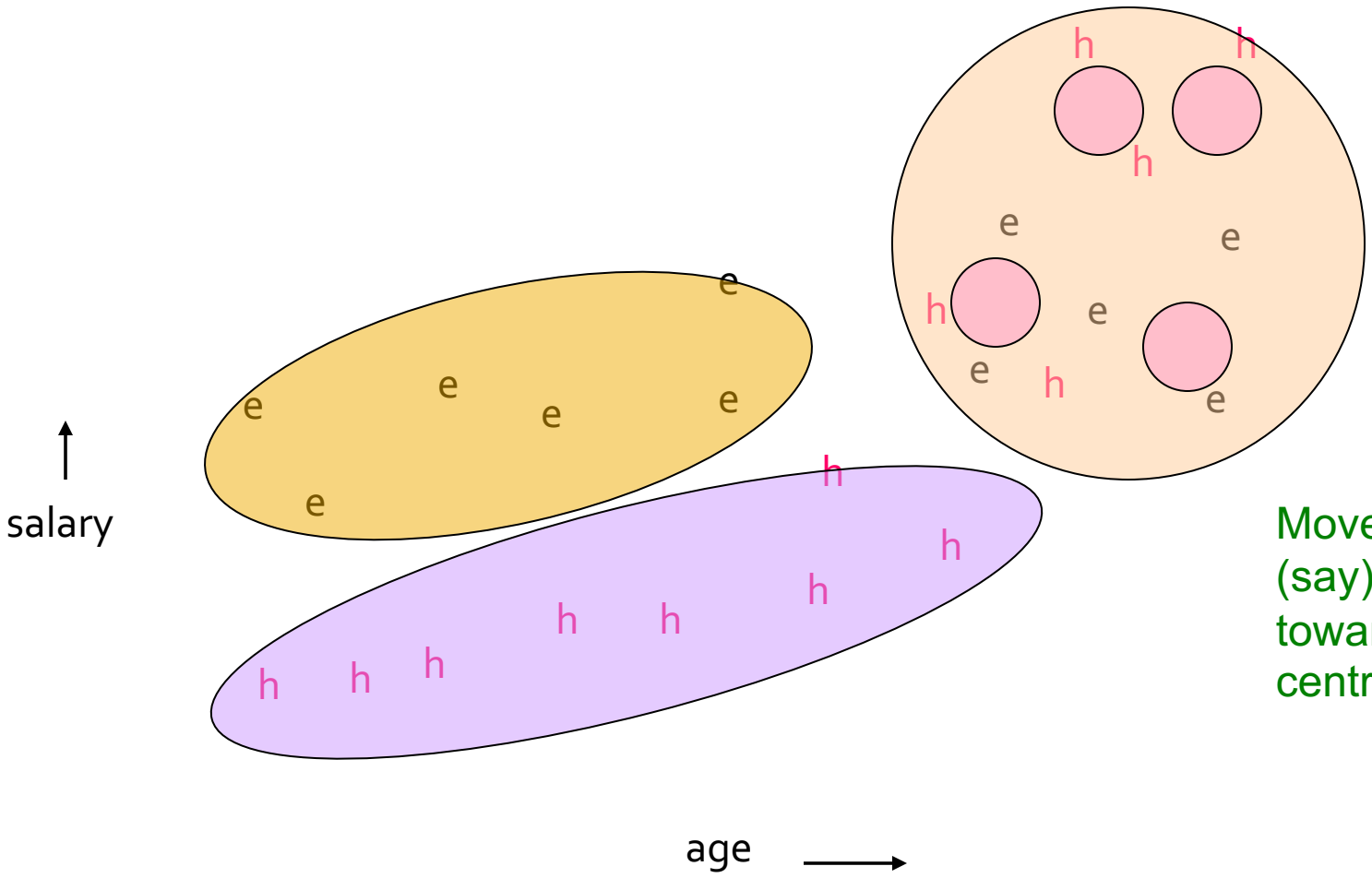
Example: Initial Clusters



Example: Pick Dispersed Points



Example: Pick Dispersed Points

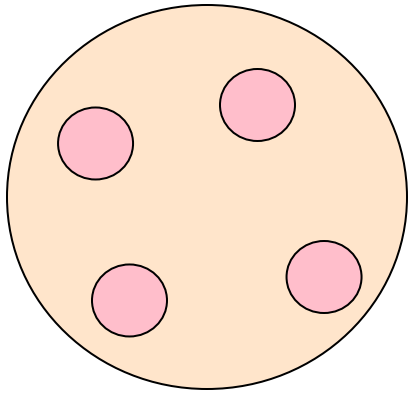


Move points (say) 20% toward the centroid.

Finishing CURE

Pass 2:

- Now, rescan the whole dataset and visit each point ***p*** in the data set



p

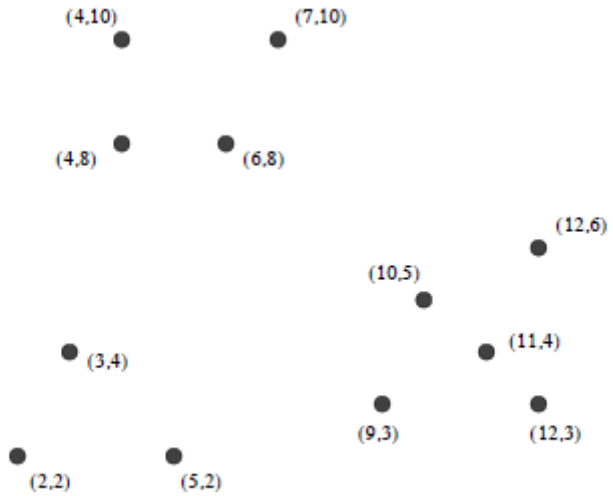
- **Place it in the "closest cluster"**
  - Normal definition of "closest": Find the closest representative to ***p*** and assign it to representative's cluster

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**
  - **CURE**

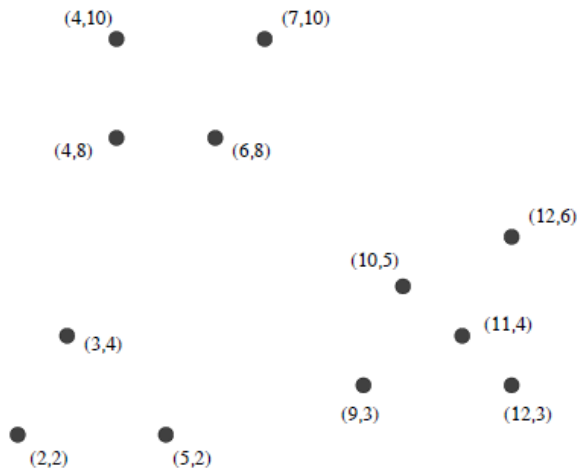
Quiz: Hierarchical Clustering

- Apply hierarchical clustering on the following data in a 2-dimensional Euclidean space.
- Assume stopping point is k = 6.
- Present the tree showing the complete grouping of the points.



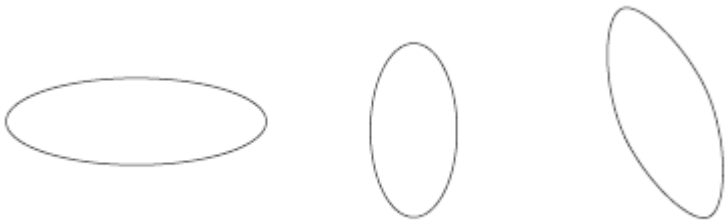
Quiz: K-Means Clustering Initialization

- Let us consider the twelve points presented below. K-Means clustering algorithm for k = 3 is used.
- Assume the initial choice of point is (6,8). What are the other two initial points? Pick points that are as far away as possible.
- Provide result of K-Means alg. after 2nd round.



Quiz: BFR Algorithm

- Describe in steps how the BFR algorithm works.
- Which of the following three shapes are allowed for clusters in BFR algorithm?

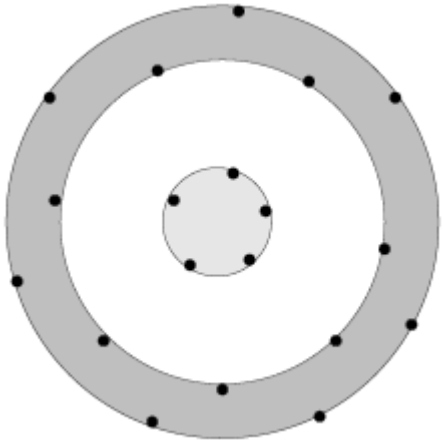


Quiz: SUM, SUMSQ, Variance

- Suppose a cluster consists of the points (5, 1), (6,−2), and (7, 0).
  - Compute each of the following statistics:
    - N
    - SUM
    - SUMSQ
    - Centroid
    - Variance
    - Standard Deviation
- Do statistics have to be computed from scratch in each round of BFR algorithm?

Quiz: Initialization in CURE

- Describe in steps how CURE algorithm works.
- Assuming following selection of representative points from each cluster move the representative points 20% of the distance to the cluster's centroid.



Quiz: Mahalanobis Distance

- Suppose a cluster of three-dimensional points has standard deviations of 2, 3, and 5, in the three dimensions, in that order. Compute the Mahalanobis distance between the origin (0, 0, 0) and the point (1,−3, 4).

Quiz: Mahalanobis Distance

- Suppose a cluster of three-dimensional points has standard deviations of 2, 3, and 5, in the three dimensions, in that order. Compute the Mahalanobis distance between the origin (0, 0, 0) and the point (1,−3, 4).
- Ans:
  - Mahalanobis distance $\sqrt{\sum_{i=1}^{d} \left(\frac{p_i - c_i}{\sigma_i}\right)^2}$
  - …
  - …

Summary

- Clustering: Clusters are often a useful summary of data that is in the form of points in some space.
  - To cluster points, we need a distance measure on that space. Ideally, points in the same cluster have small distances between them, while points in different clusters have large distances between them.
- Clustering Algorithms: Clustering algorithms generally have one of two forms.
  - Hierarchical clustering algorithms begin with all points in a cluster of their own, and nearby clusters are merged iteratively.

    Point-assignment clustering algorithms consider points in turn and assign them to the cluster in which they best fit.

Summary

- Centroids and Clustroids: In a Euclidean space, the members of a cluster can be averaged, and this average is called the centroid.
- In non-Euclidean spaces, there is no guarantee that points have an "average," so we are forced to use one of the members of the cluster as a representative or typical element of the cluster. That representative is called the clustroid.

Summary

- K-Means Algorithms: This family of algorithms is of the point-assignment type and assumes a Euclidean space.
  - It is assumed that there are exactly k clusters for some known k.
  - After picking k initial cluster centroids, the points are considered one at a time and assigned to the closest centroid.
  - The centroid of a cluster can migrate during point assignment, and an optional last step is to reassign all the points, while holding the centroids fixed at their final values obtained during the first pass.

Summary

- The BFR Algorithm: This algorithm is a version of k-means designed to handle data that is too large to fit in main memory. It assumes clusters are normally distributed about the axes.
- Processing Points in BFR: Most of the points in a main-memory load will be assigned to a nearby cluster and the parameters for that cluster will be adjusted to account for the new points.
  - Unassigned points can be formed into new miniclusters, and these miniclusters can be merged with previously discovered miniclusters or retained points.

Summary

- The CURE Algorithm: This algorithm is of the point-assignment type. It is designed for a Euclidean space, but clusters can have any shape. It handles data that is too large to fit in main memory.
- Processing Points in CURE: After creating representative points for each cluster, the entire set of points can be read from disk and assigned to a cluster. We assign a given point to the cluster of the representative point that is closest to the given point.

Actions

- Review slides!
- Read Chapter 7 from course book.
  - You can find electronic version of the book on Blackboard.