Data Mining - Unsupervised Machine Learning

Data Clustering

Prof. Yannis Velegrakis
University of Trento & Utrecht University

https://velgias.github.io

Use Cases

- Tourism: Understanding Points of Interest
  - Transportation
  - Shops / Services
  - Security / Safety
- Security: Understanding High Risk Areas
- Population Understanding: Types of citizens and Needs
- Health: High Risk Areas (The old story of London City)
- Software Usage in the Offices:
  - Identify best needed software packages
- Home Services (Gas, Water, Electricity) Consumption
- Social Media: What the citizens want/think
- Telecommunication Data: Identify Families

- Other types of Data:
  - Segments: Transportation Routes, Tourism Paths, Super Market Shopping Plans
  - Graphs: Communities
  - Streams: City Sensors

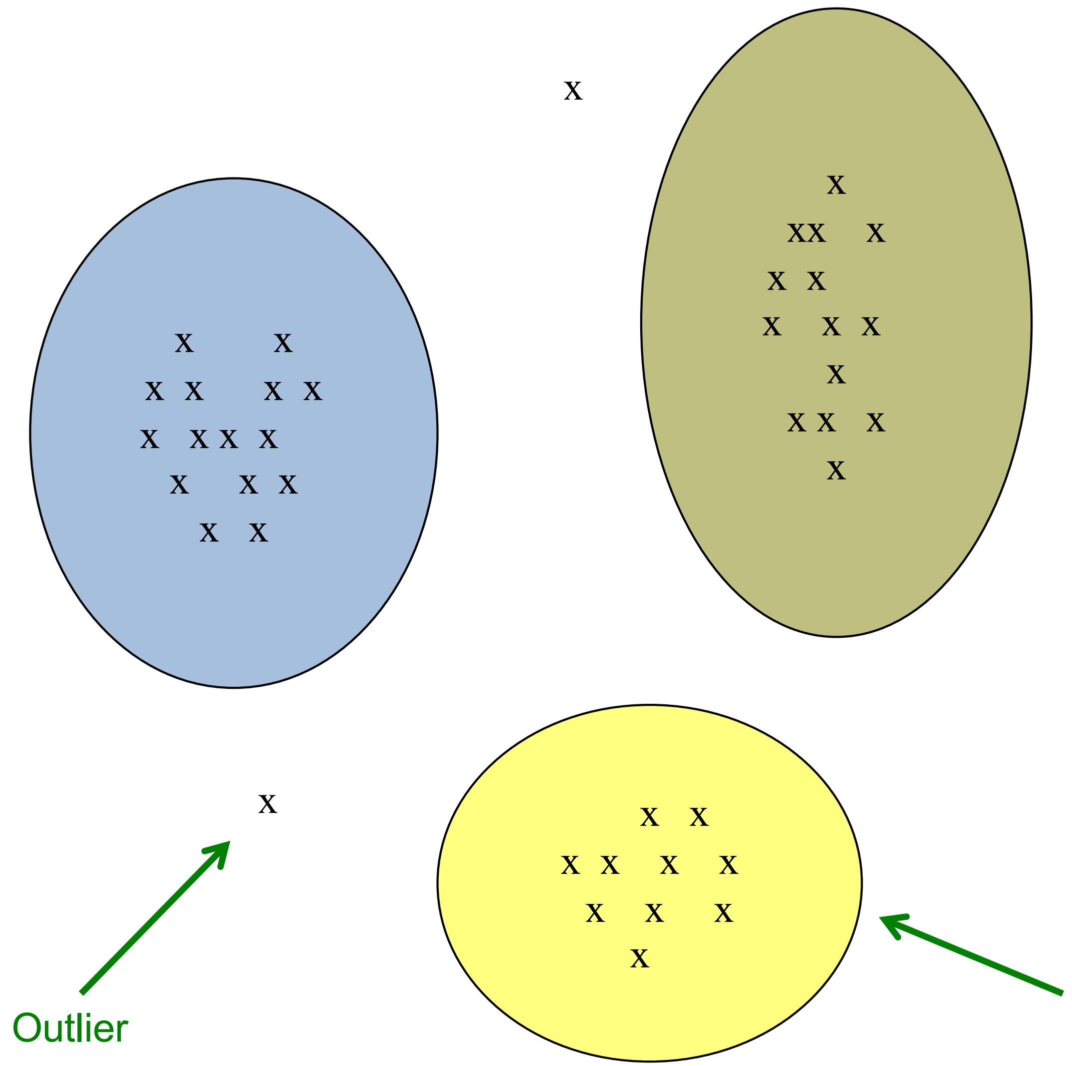
High Dimensional Data

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

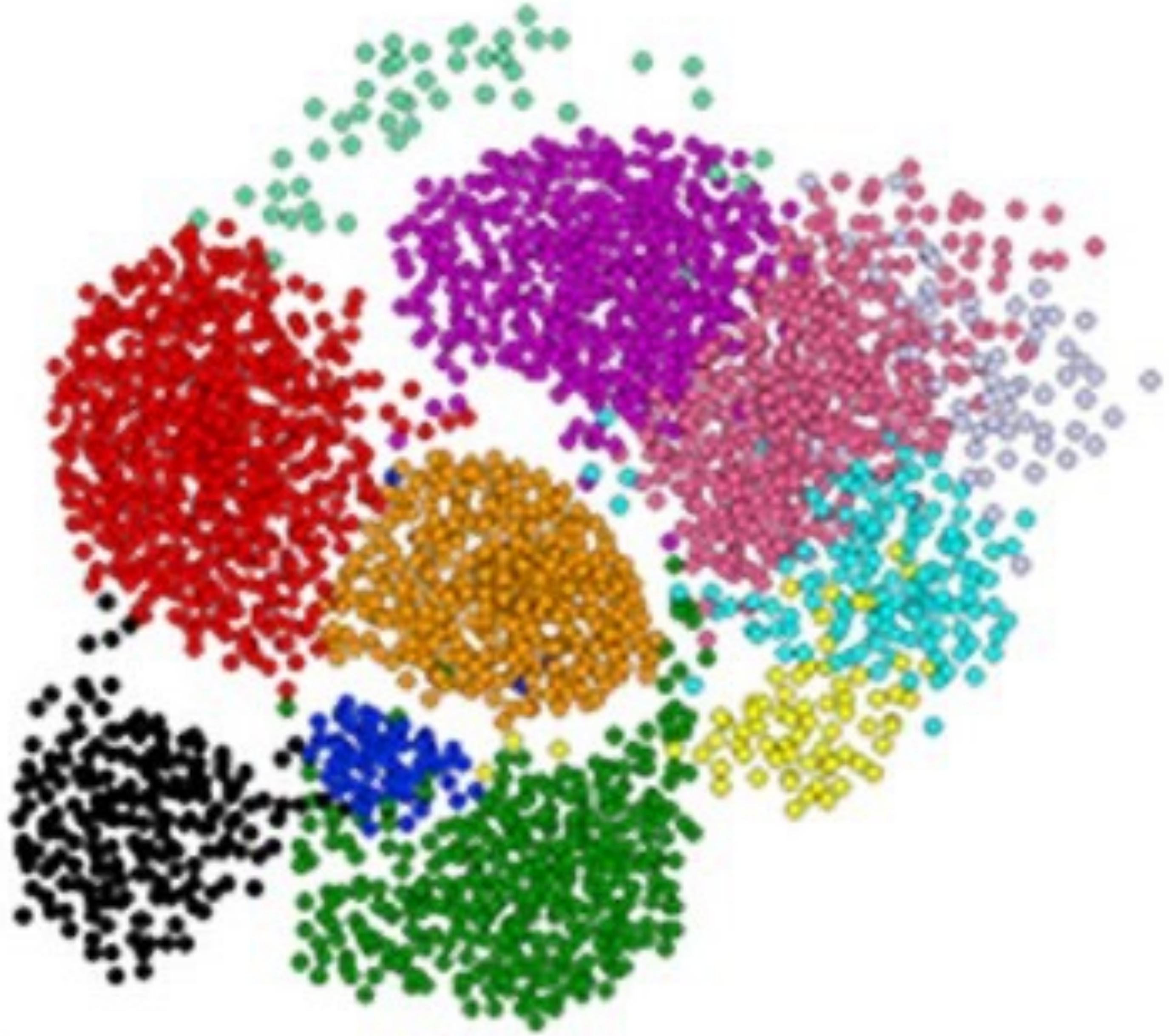
The Problem of Clustering

- 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
- 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: 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, moons, belts, clouds, etc.
- Sloan Digital Sky Survey

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 $(x_1, x_2, \ldots, x_k)$, 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
  - 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**

Clustering Problem: Topics (Dual)

Finding topics:

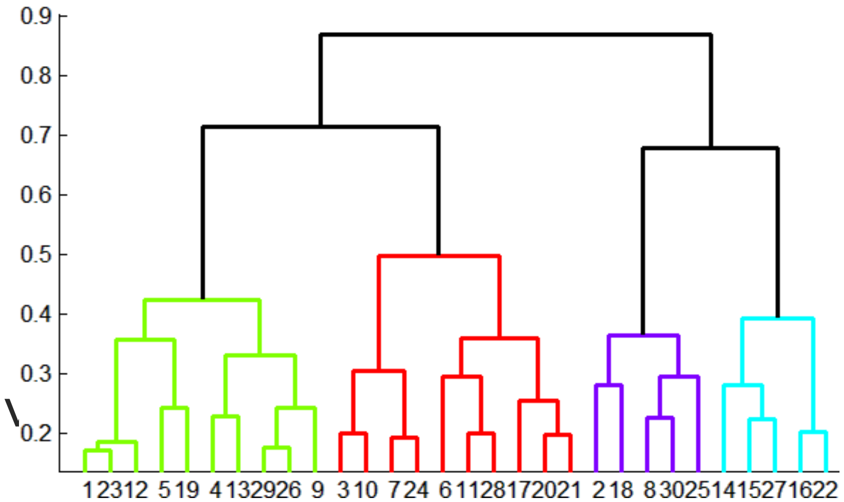
- Represent a document by a vector $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} word (in some order) appears in the document
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words
- **Topic is a set of similar documents**

Cosine, Jaccard, and Euclidean

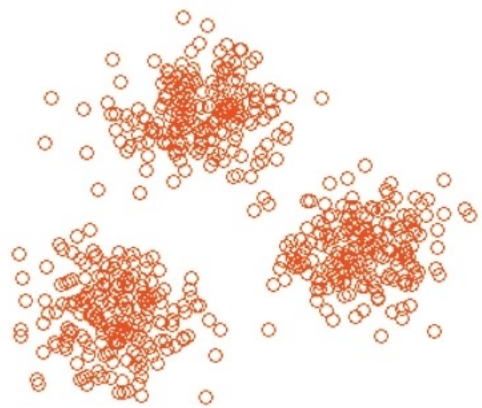
- **As with CDs we have a choice when we think of documents as sets of words:**
  - **Sets as vectors:** Measure similarity by the **cosine distance**
  - **Sets as sets:** Measure similarity by the **Jaccard distance**
  - **Sets as points:** Measure similarity by **Euclidean distance**

Overview: Methods of Clustering

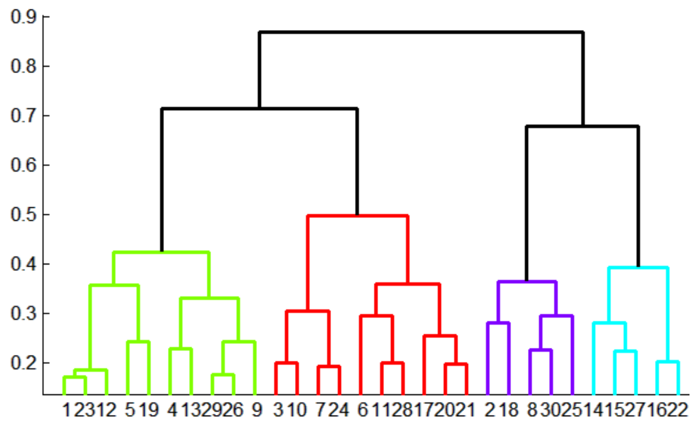
- **Hierarchical:**
  - **Agglomerative** (bottom up):
    - Initially, each point is a cluster
    - Repeatedly combine the two "nearest" clusters into one
  - **Divisive** (top down):
    - Start with one cluster and recursiv



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

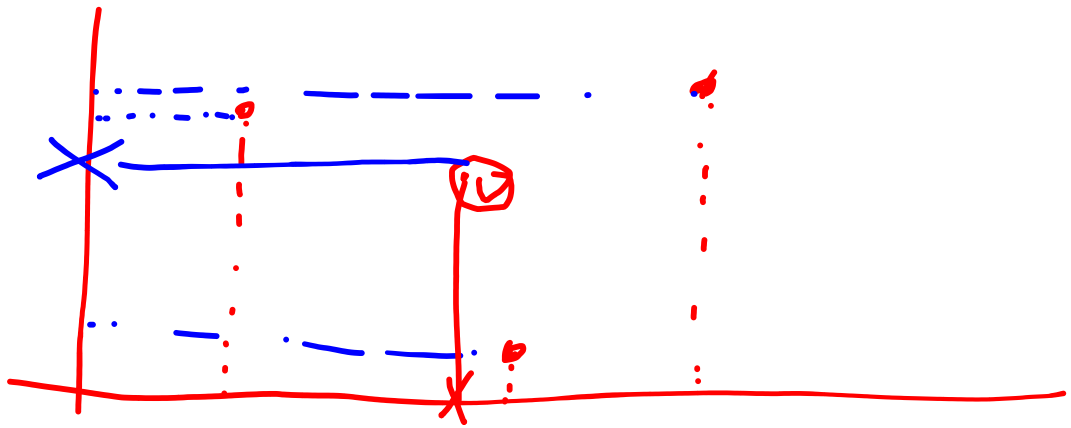


Hierarchical Clustering



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

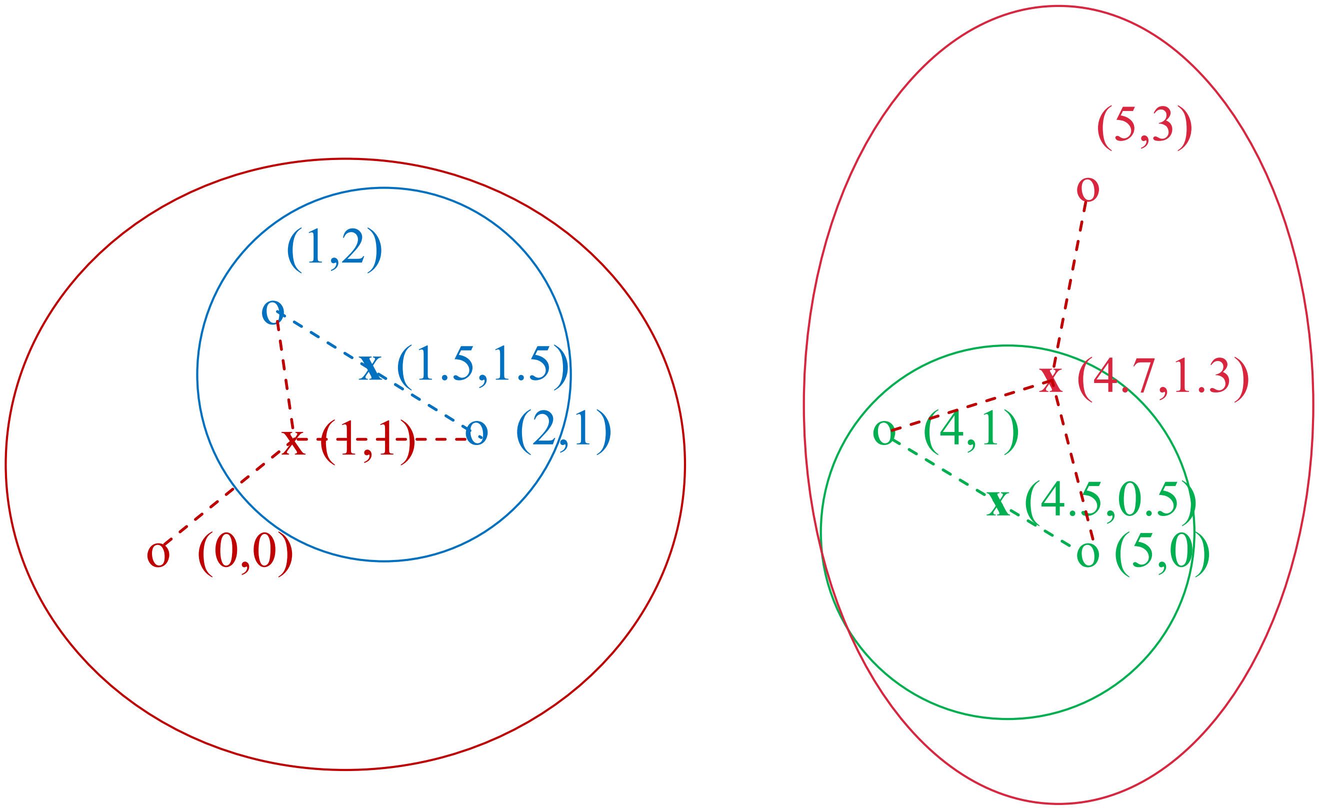
- **Three important questions:**
  - **1)** How do you represent a cluster of more than one point?
  - **2)** How do you determine the “nearness” of clusters?
  - **3)** When to stop combining clusters?



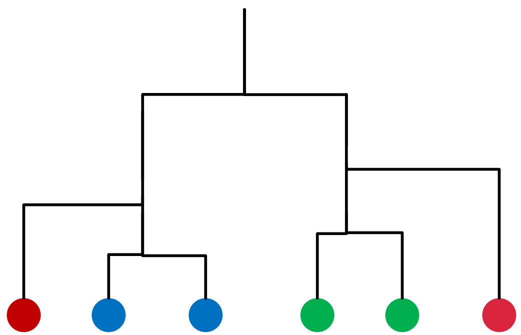
Hierarchical Clustering

- **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?

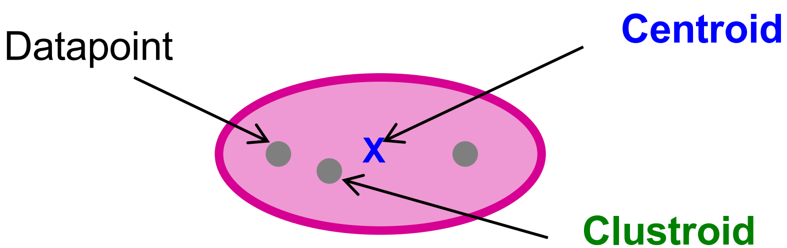
- The only "locations" we can talk about are the points themselves
  - i.e., there is no "average" of two points

- **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
  - 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 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

Termination Conditions

- When do you stop combining clusters?
- Approach 1: Pick k up front and stop when you have k
  - makes sense if we know that the data falls into k categories
- Approach 2: Stop when the next merge will create a cluster with low cohersion
  - i.e. bad cluster

Implementation

- **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 $O(N^2 \log N)$
  - **Still too expensive for really big datasets that do not fit in memory**

k-means clustering

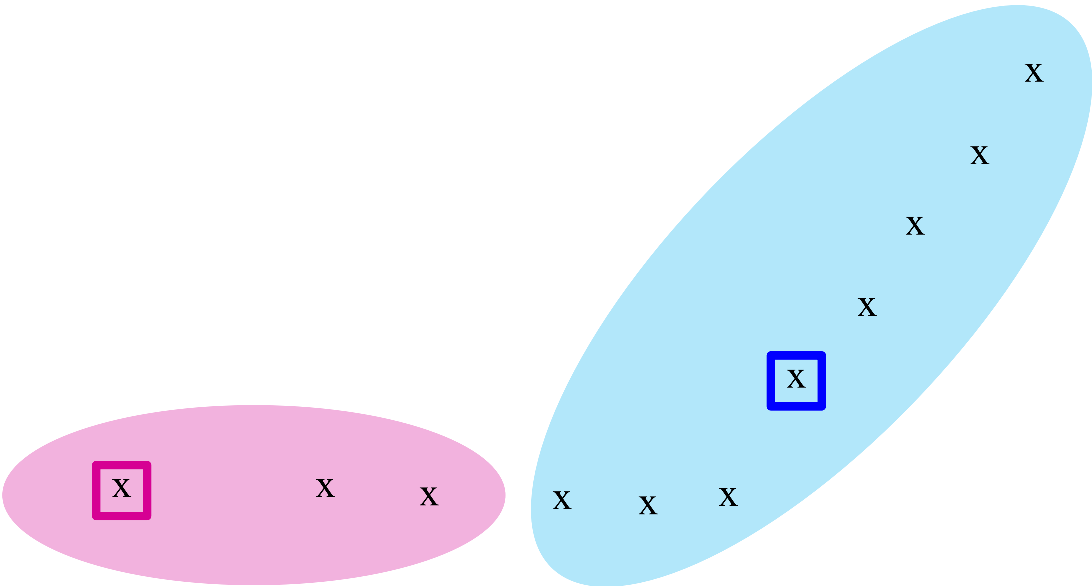
k–means Algorithm(s)

- 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

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

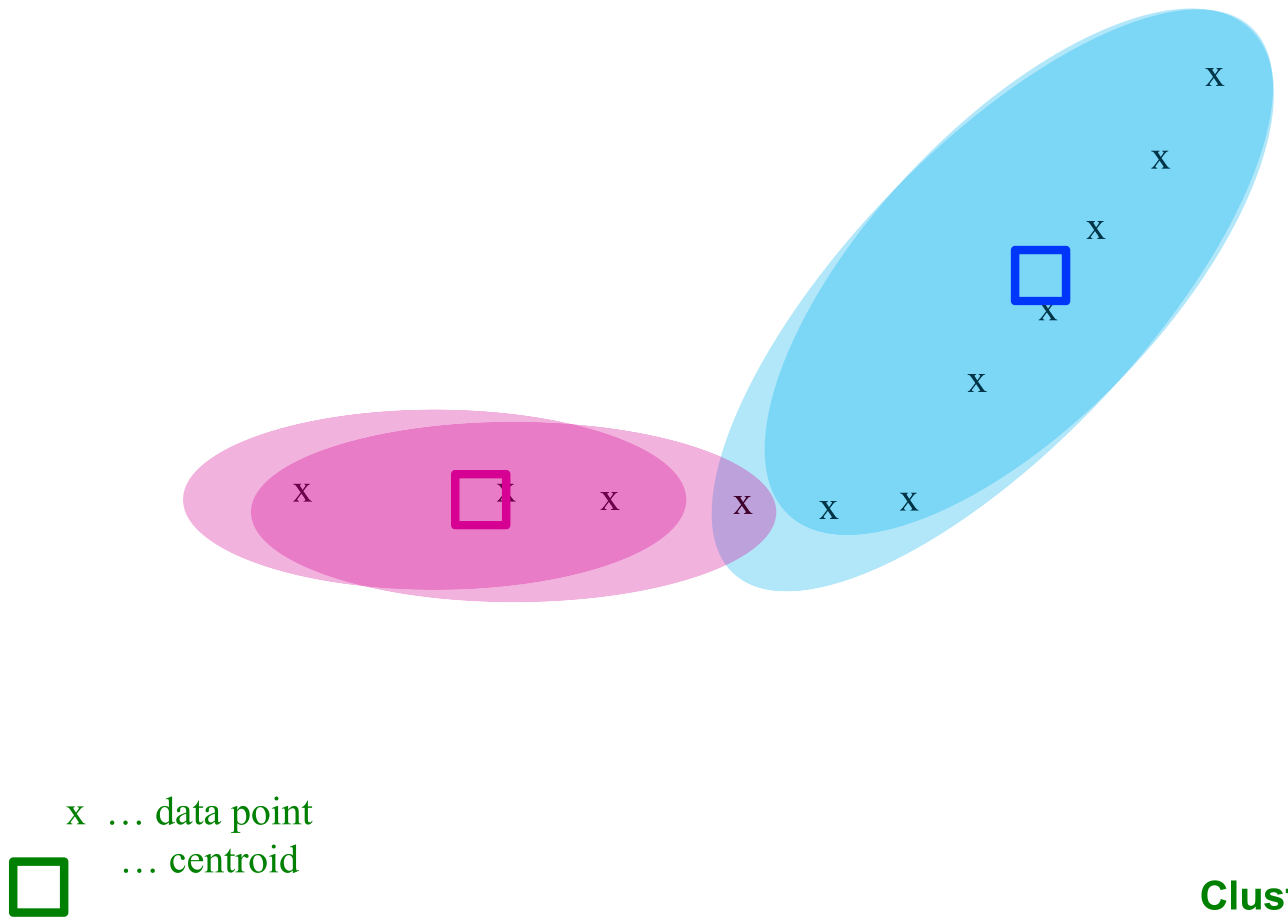
Example: Assigning Clusters



x … data point
… centroid

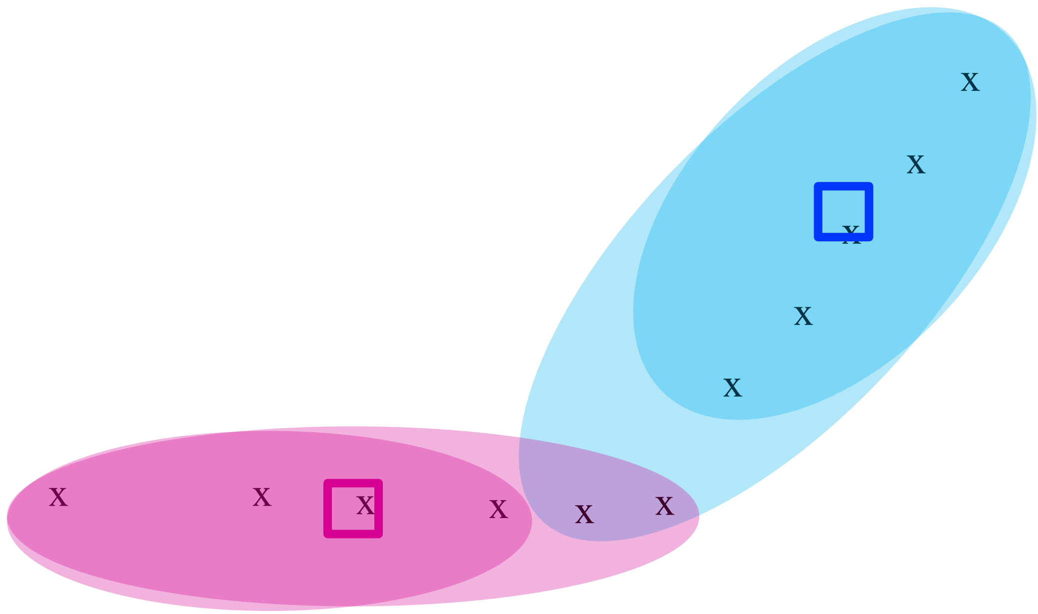
Clusters after round 1

Example: Assigning Clusters



Clusters after round 2

Example: Assigning Clusters



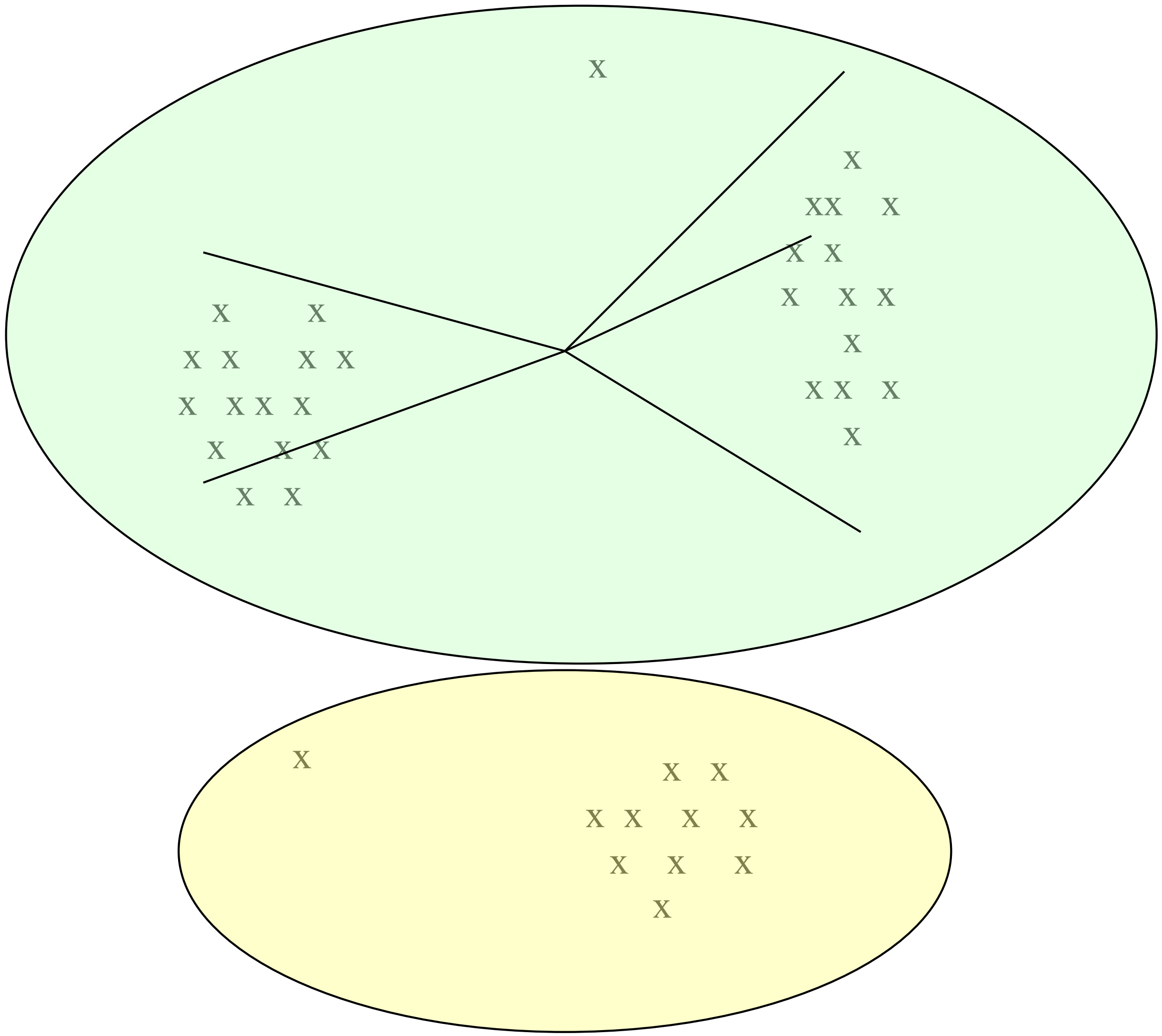
x … data point

… centroid

Clusters at the end

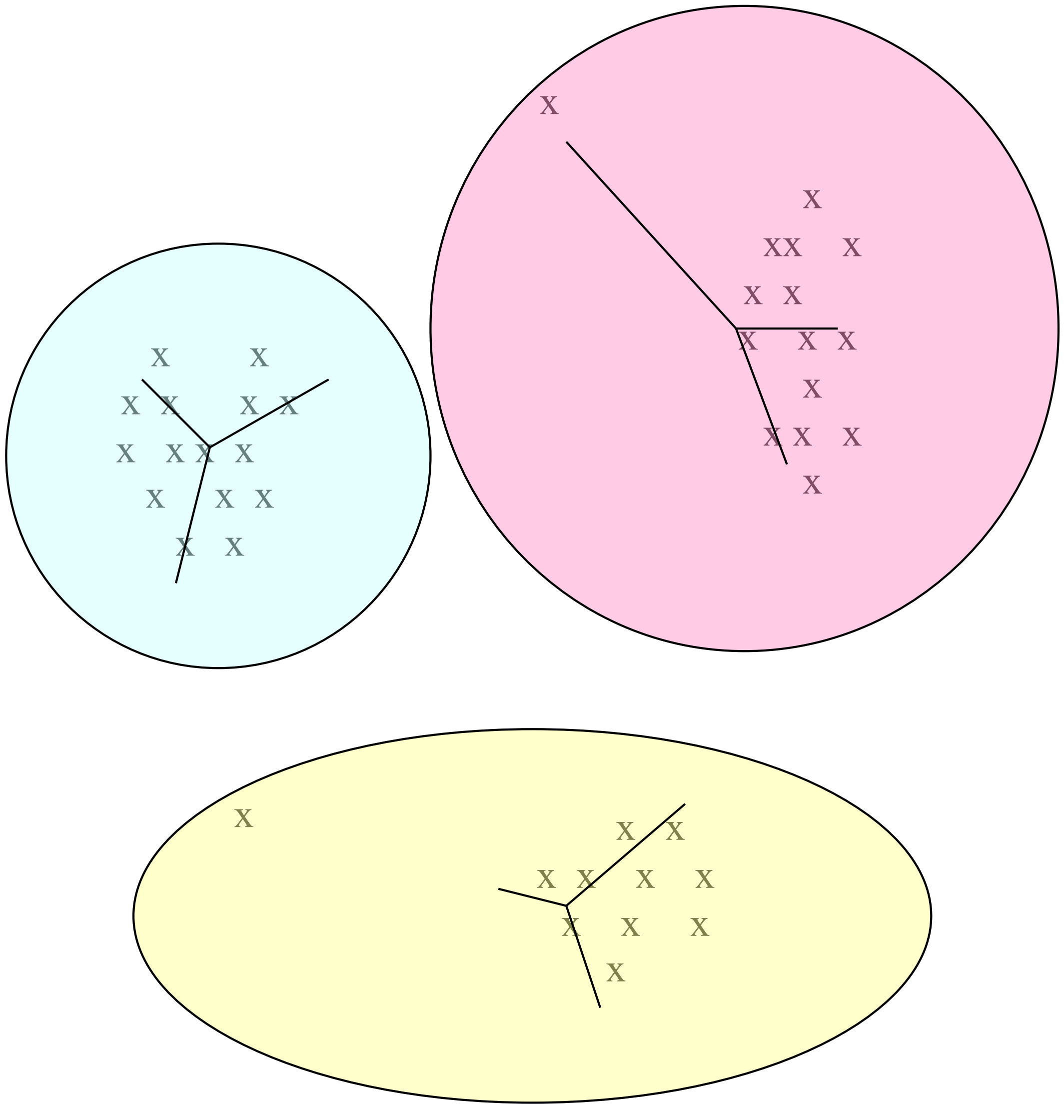
Example: Picking *k*

Too few;
many long
distances
to centroid.



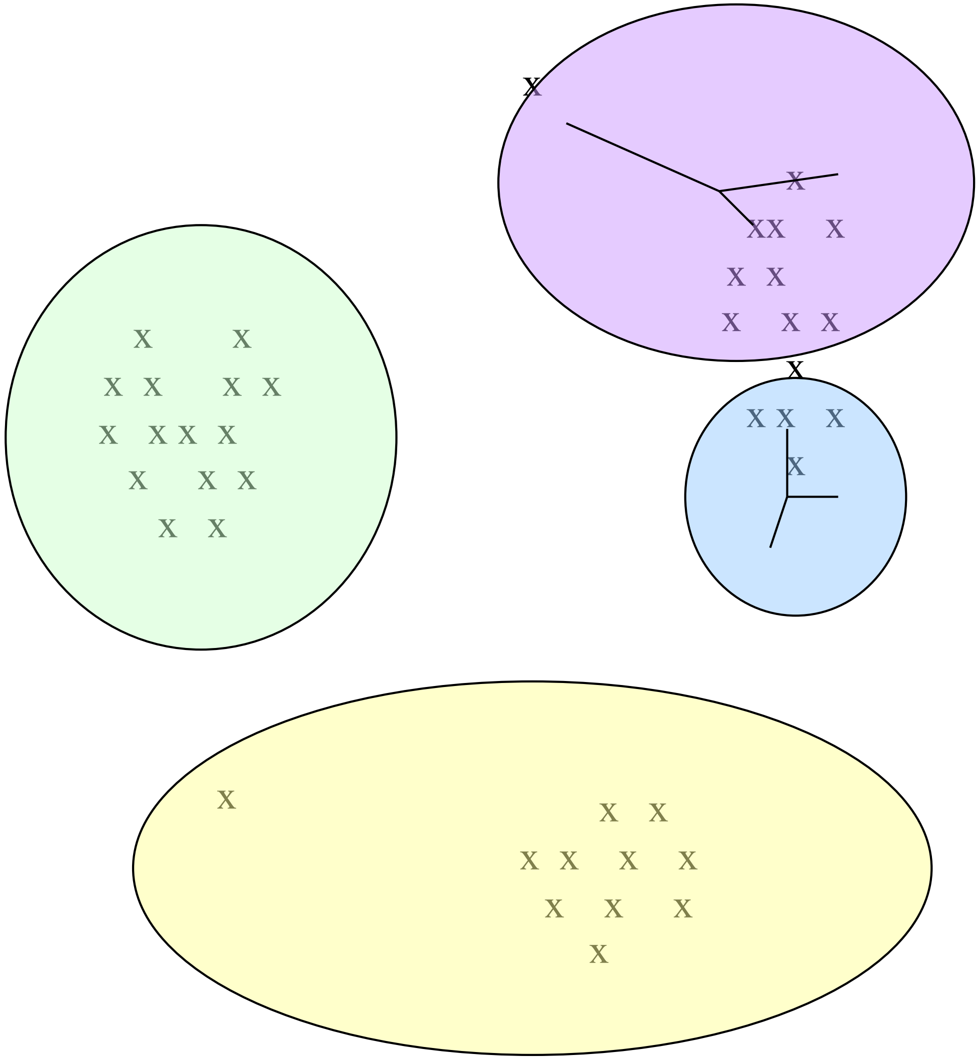
Example: Picking *k*

Just right;
distances
rather short.



Example: Picking *k*

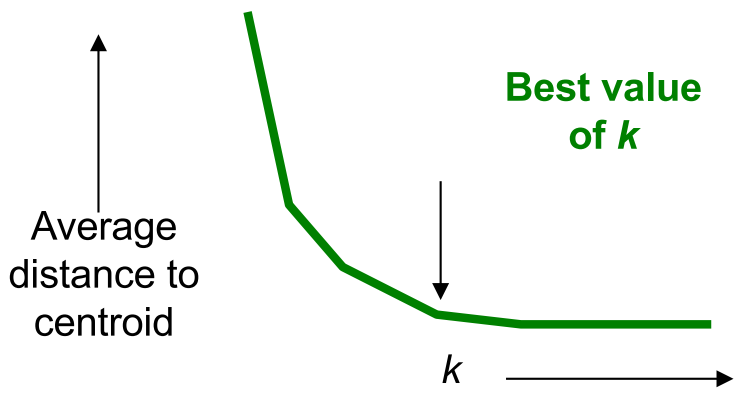
Too many;
little improvement
in average
distance.



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



Picking the initial k points

- Approach 1: Sampling
  - Cluster a sample of the data using hierarchical clustering, to obtain k clusters
  - Pick a point from each cluster (e.g. point closest to centroid)
  - Sample fits in main memory
- Approach 2: Pick "dispersed" set of points
  - Pick first point at random
  - Pick the next point to be the one whose minimum distance from the selected points is as large as possible
  - Repeat until we have *k* points

The BFR Algorithm

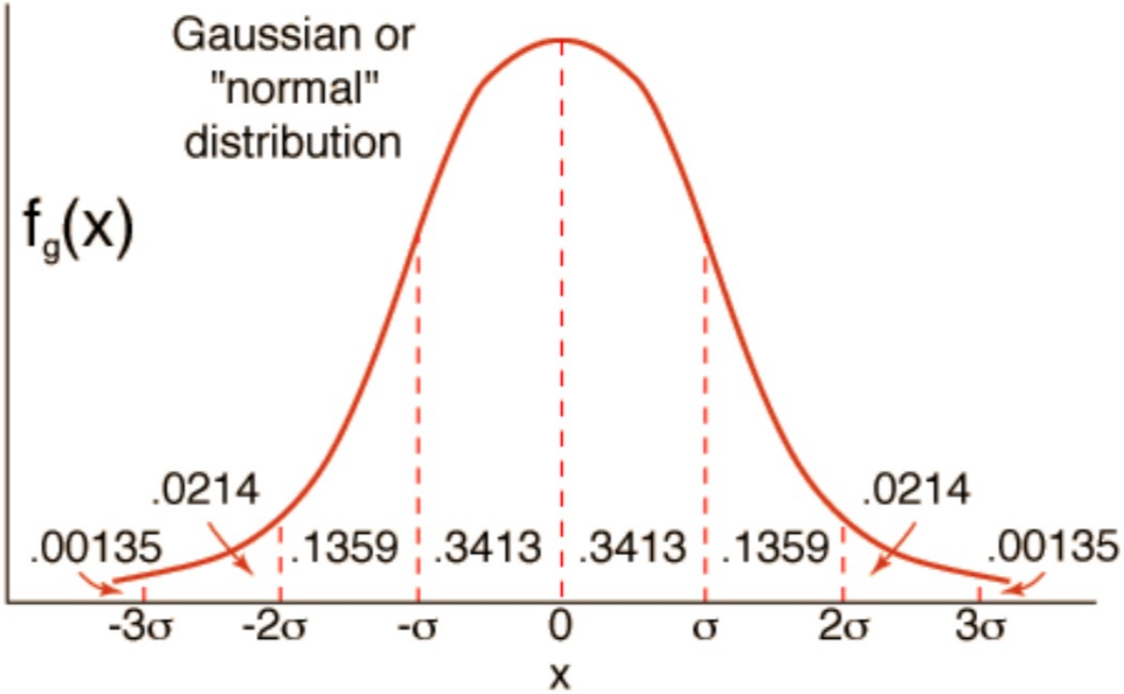
Extension of *k*-means to larger data

BFR Algorithm

- **BFR** [Bradley-Fayyad-Reina] is a variant of *k*-means for very large (disk-resident) data sets

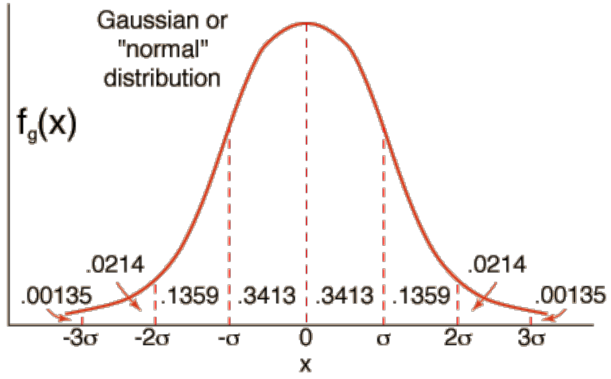
- Assumes each cluster is normally distributed around a centroid in Euclidean space

Normal Distribution



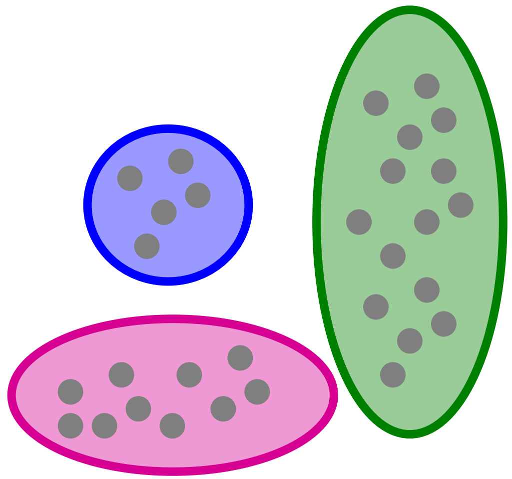
- Can quantify the likelihood of finding a point in the cluster at a given distance from the centroid along each dimension
- Standard deviations in different dimenions may vary

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** (memory required: O(clusters) and not O(data))



BFR Algorithm

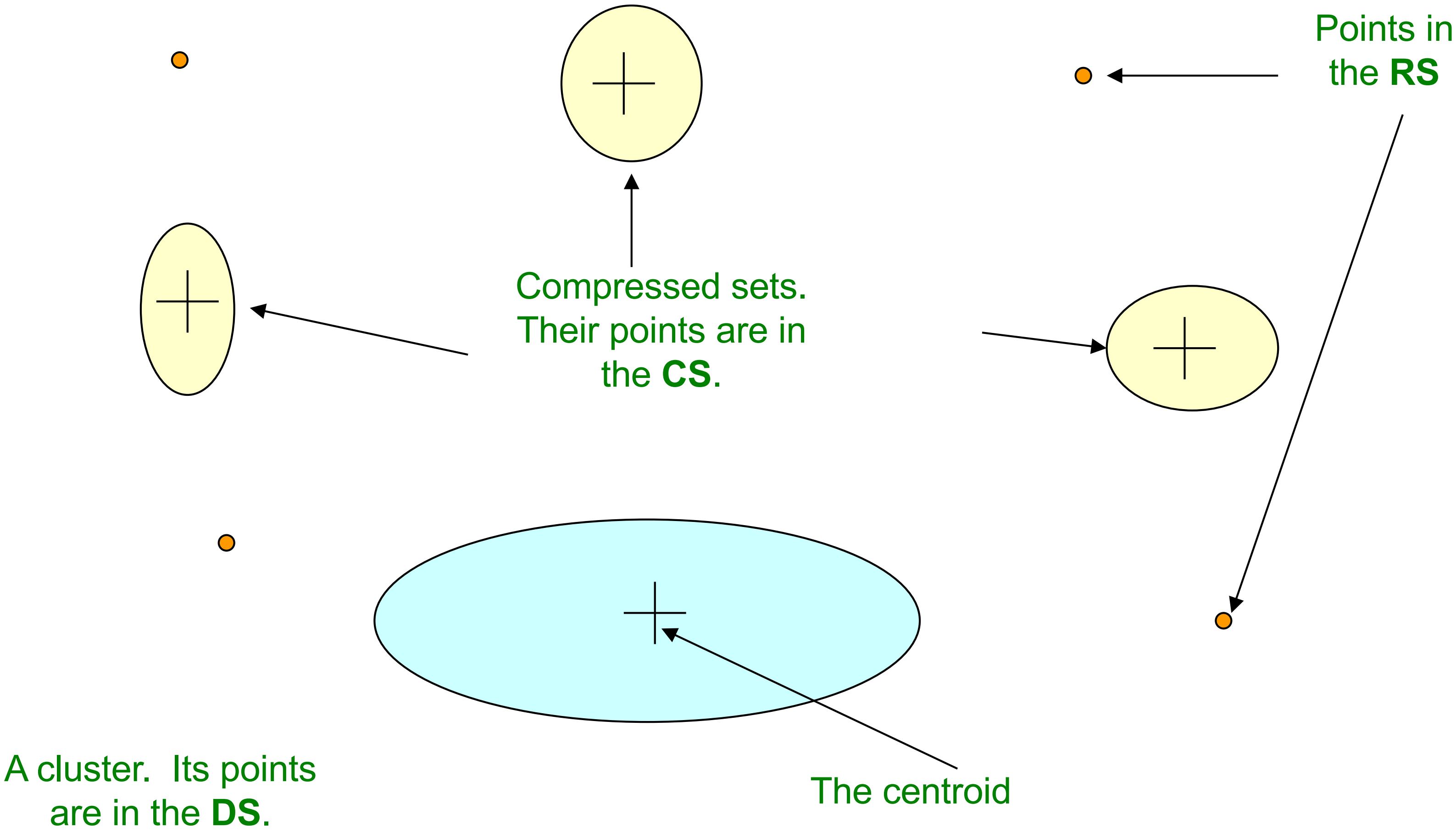
- Points are read from disk one main-memory-full at a time
- **Most points from previous memory loads are summarized by simple statistics**
- To begin, from the initial load we select the initial ***k*** centroids by some sensible approach:
  - Take ***k*** random points
  - Take a small random sample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

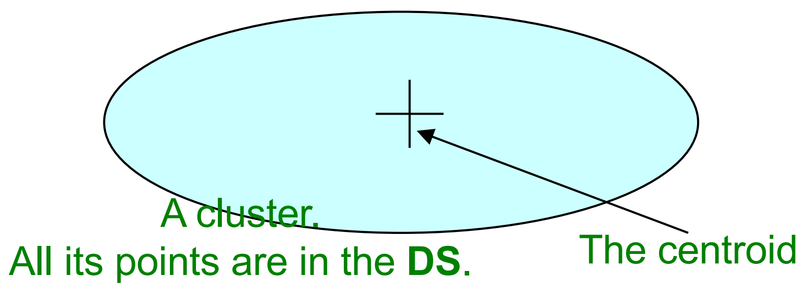


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

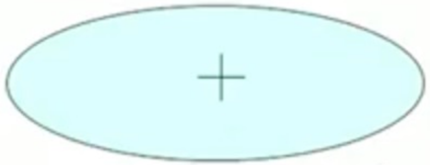
For each cluster, the discard set (DS) is summarized by:

- 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

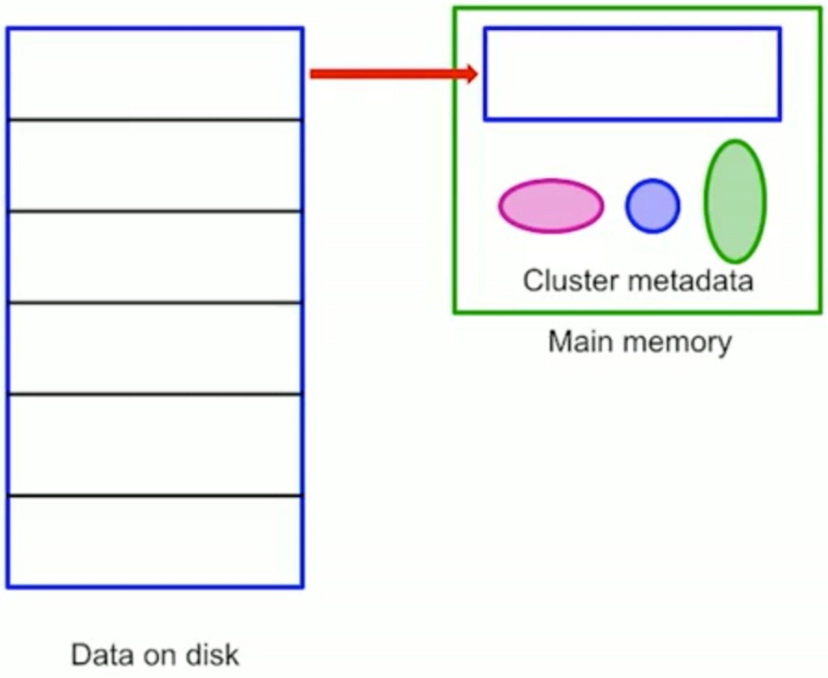


Summarizing Sets of Points

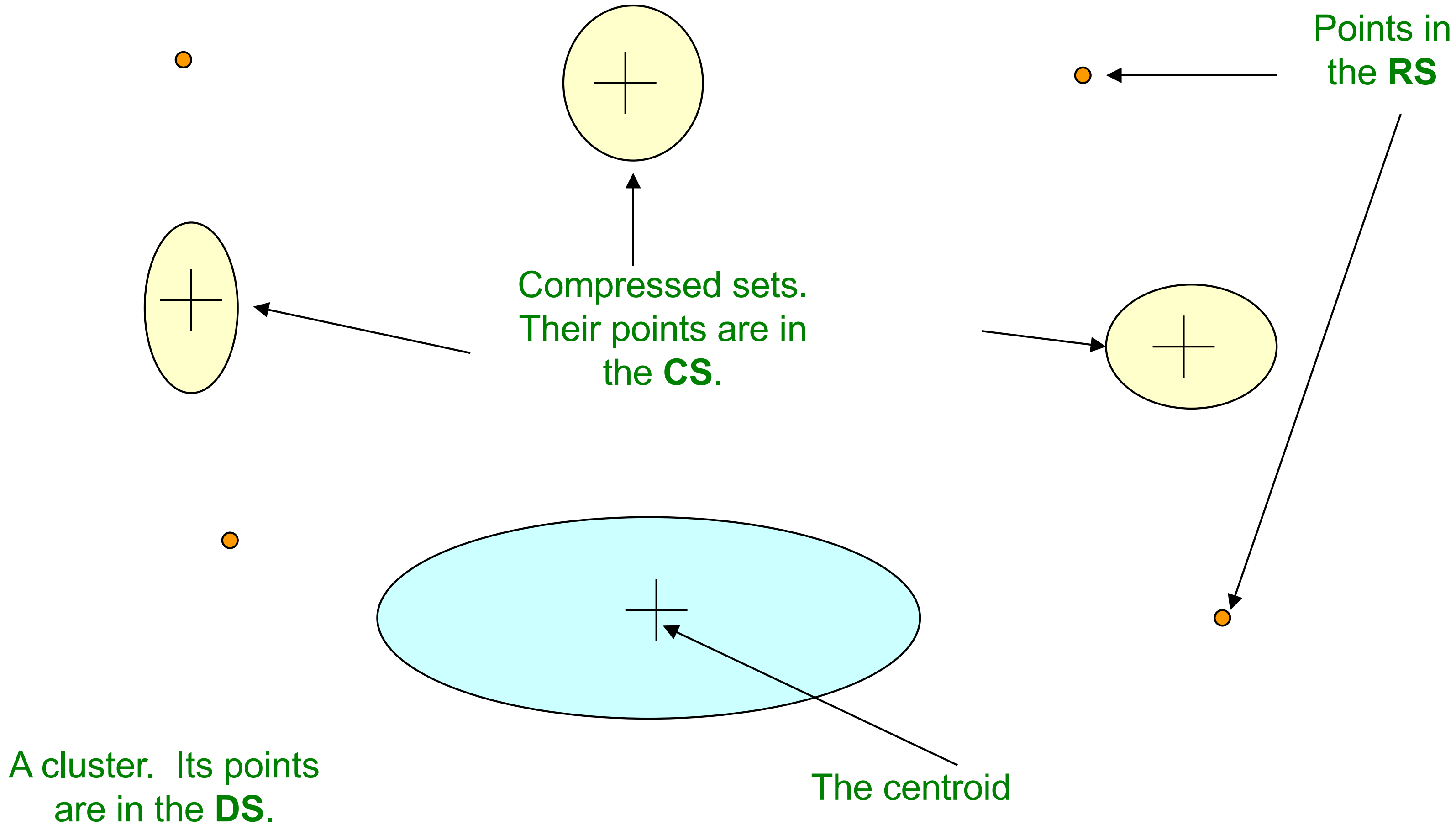
- **$2d + 1$** values represent any size cluster
  - ***d*** = number of dimensions
- Average in **each dimension** (**the centroid**) can be calculated as **SUM_i / N**
  - **SUM_i** = i^{th} component of SUM
- Variance of a cluster's discard set in dimension i is: **$(SUMSQ_i / N) - (SUM_i / N)^2$**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



BFR Overview



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

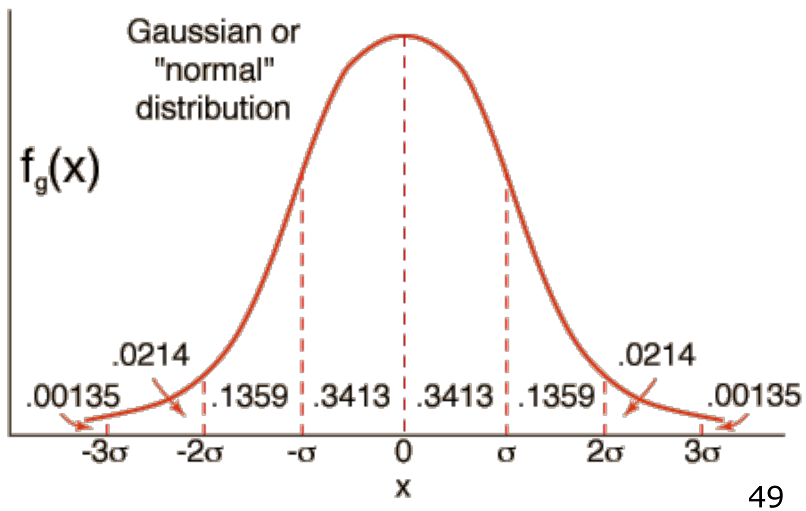
A Few Details…

- 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?
  - Compute the variance of the combined subcluster

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

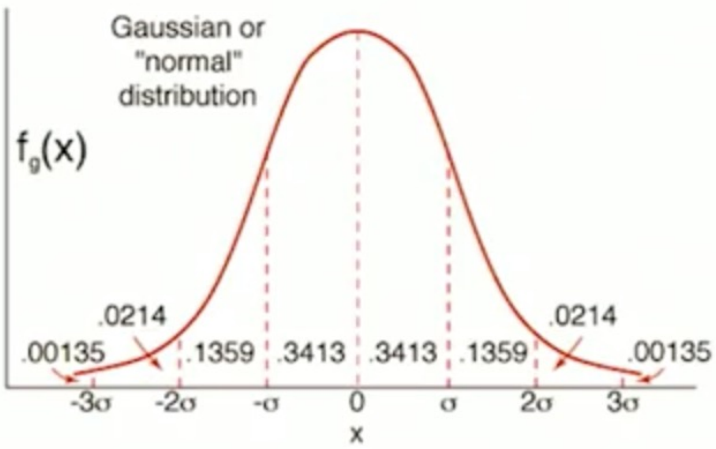
$$d(x, c) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_i - c_i}{\sigma_i}\right)^2}$$

measures the # of standard deviations from centroid across a dimension

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

Mahalanobis Acceptance Criterion

- Suppose point P is one standard deviation away from centroid in each dimension
  - Each y_i = 1 and so the MD of P is $\sqrt{d}$



68% of points have MD $\leq \sqrt{d}$

95% of points have MD $\leq 2\sqrt{d}$

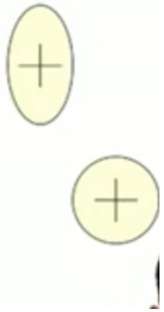
99% of points have MD $\leq 3\sqrt{d}$

Accept point P into cluster C if its MD from cluster centroid is less than a threshold e.g., $3\sqrt{d}$

Should 2 Compressed Set 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



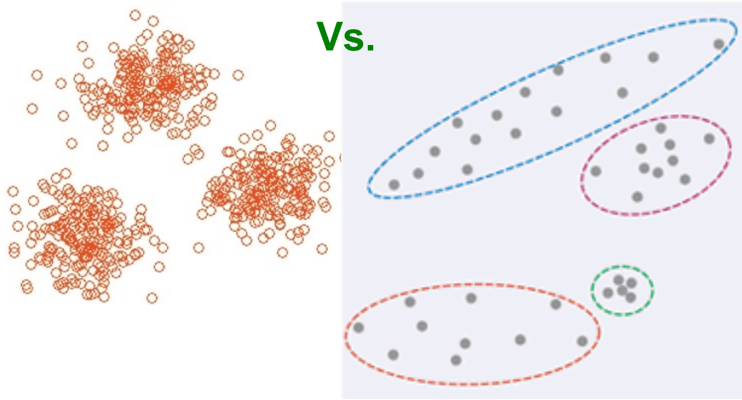
Variance = Square of Standard Deviation

The CURE Algorithm

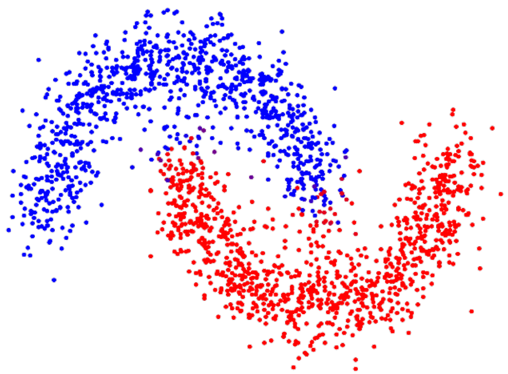
Extension of *k*-means to clusters of arbitrary shapes

The CURE Algorithm

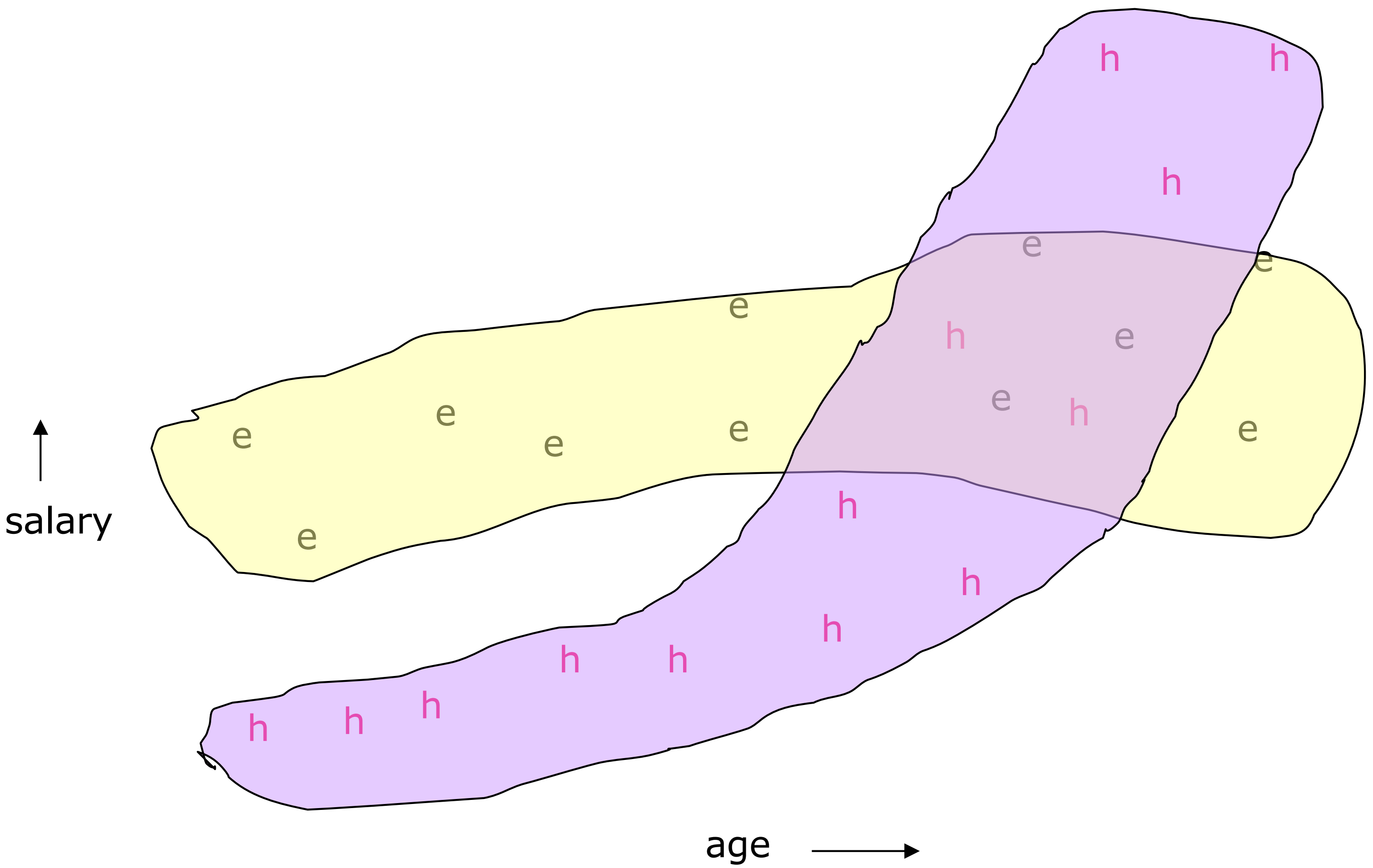
- **Problem with BFR/*k*-means:**
  - Assumes clusters are normally distributed in each dimension
  - And axes are fixed – ellipses at an angle are ***not OK***



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



Example: Professor's Salaries

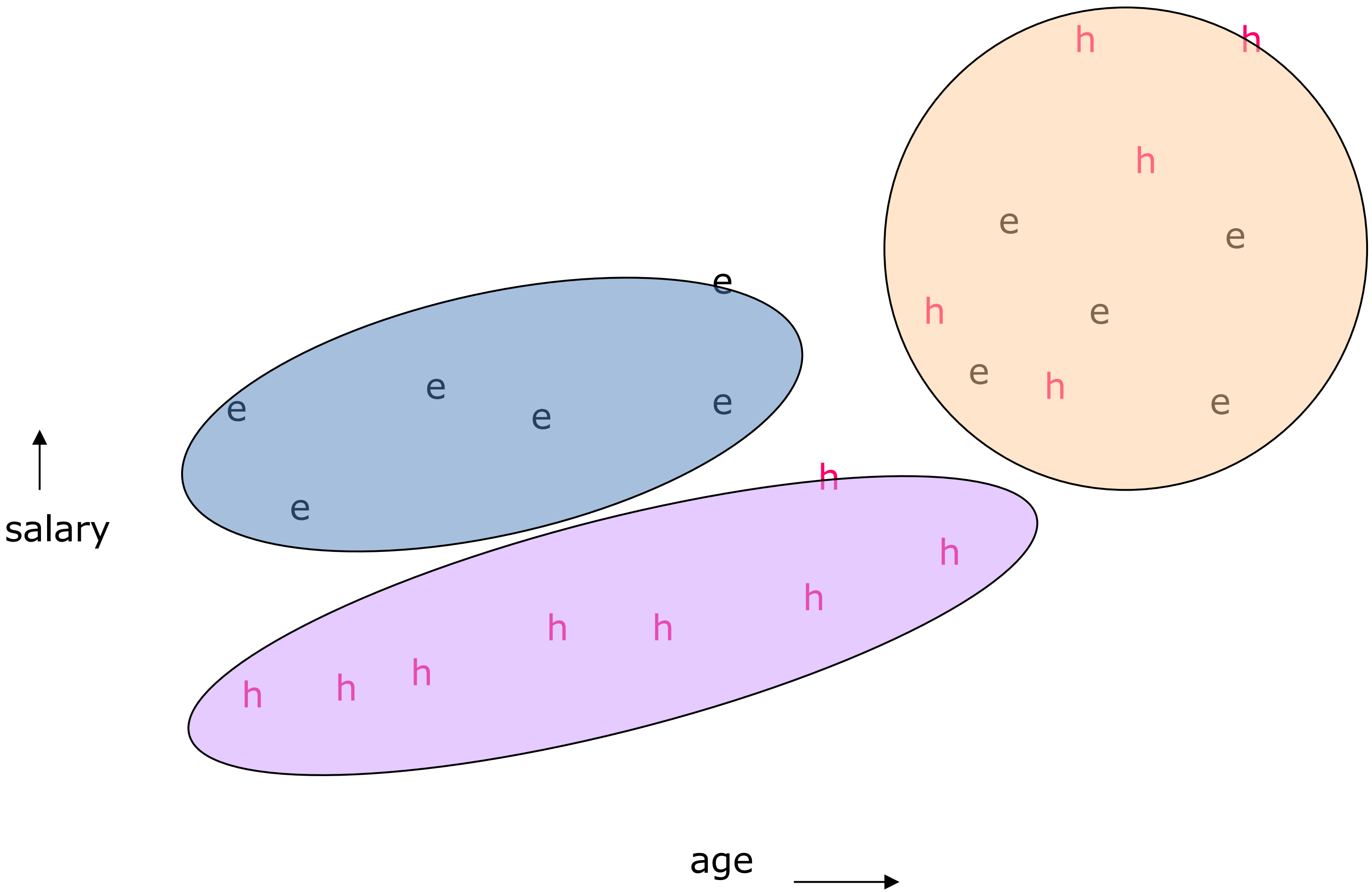


Starting CURE

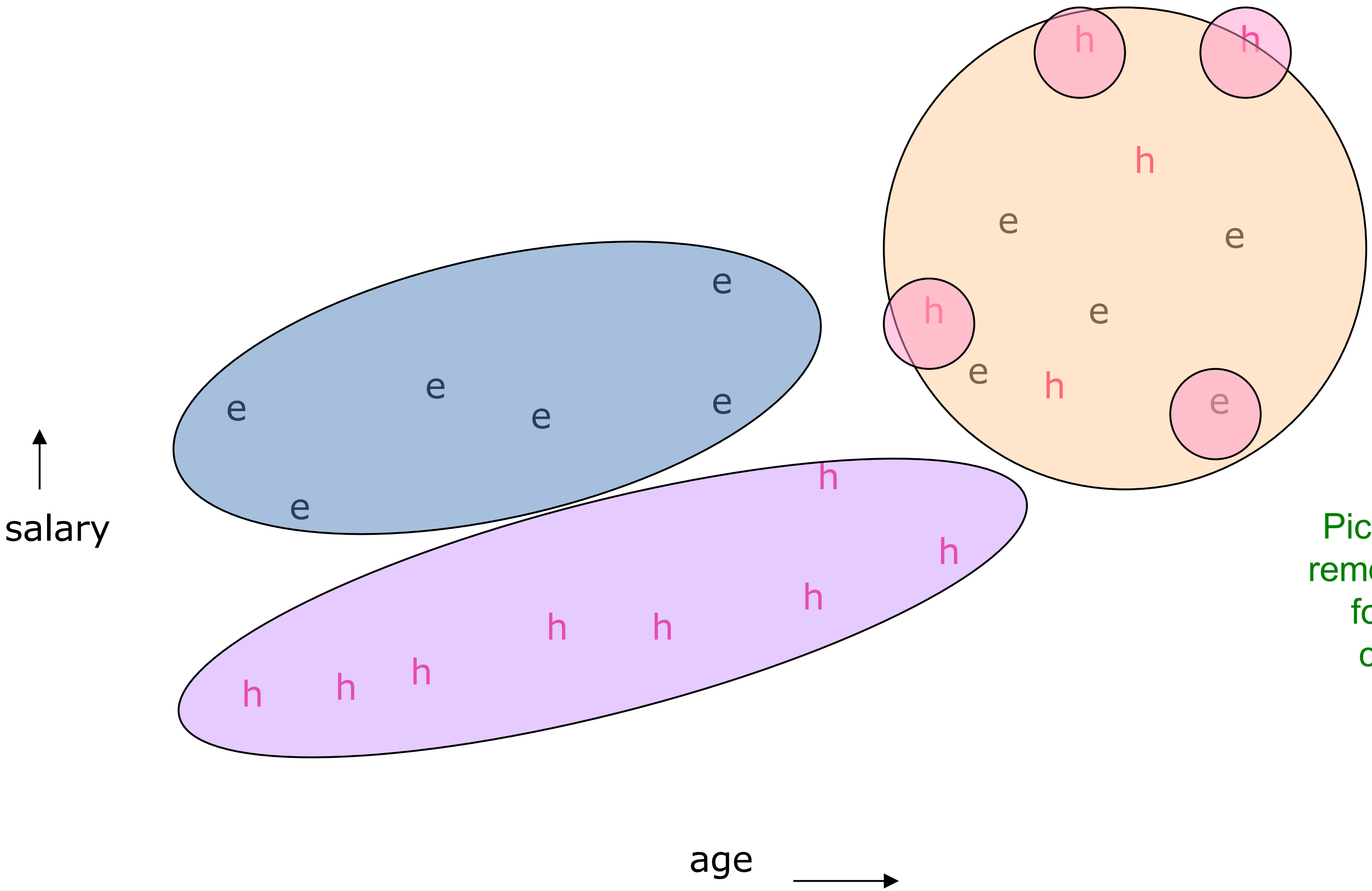
<u>2 Pass algorithm. Pass 1:</u>

- **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

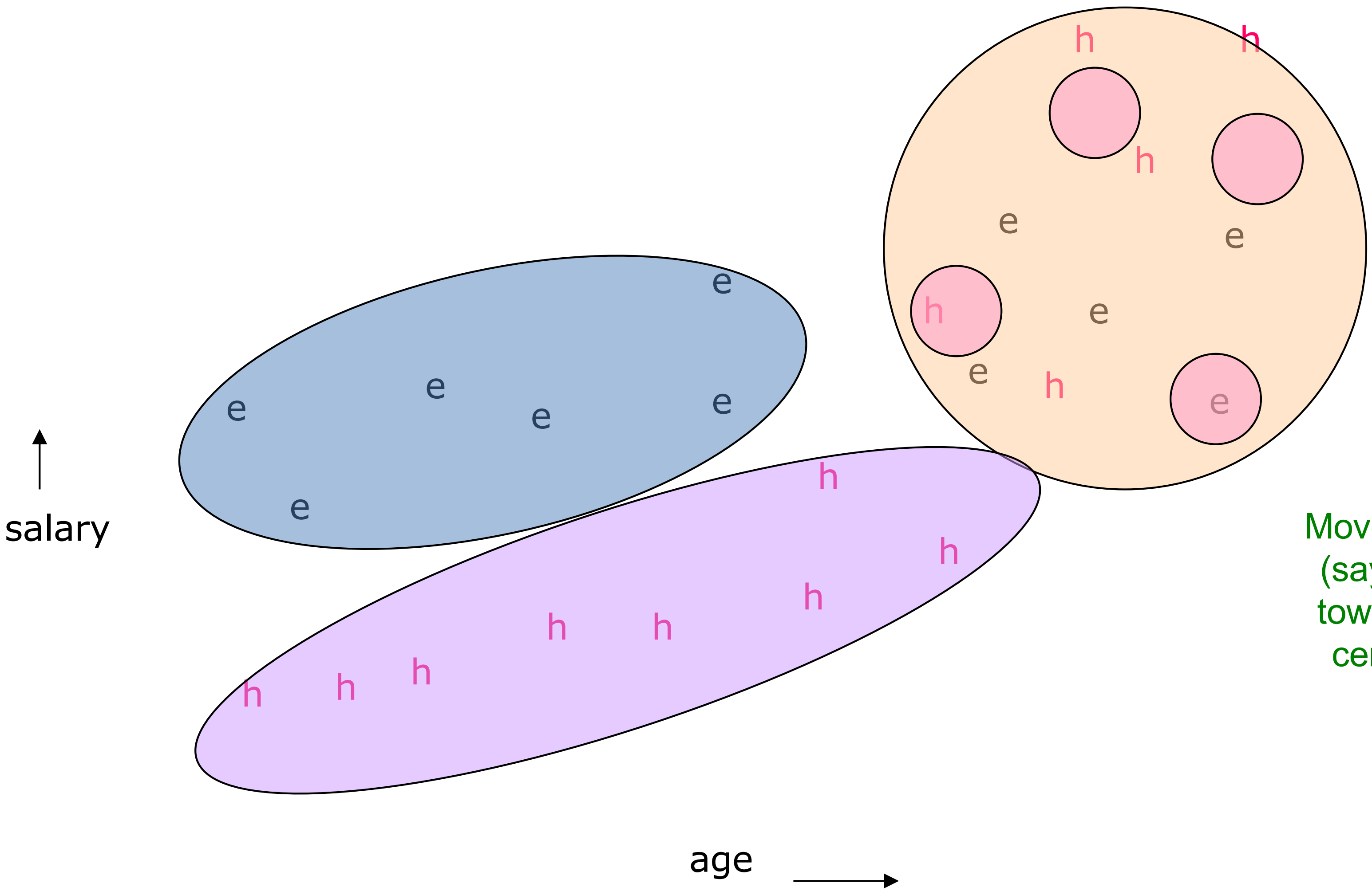


Example: Pick Dispersed Points



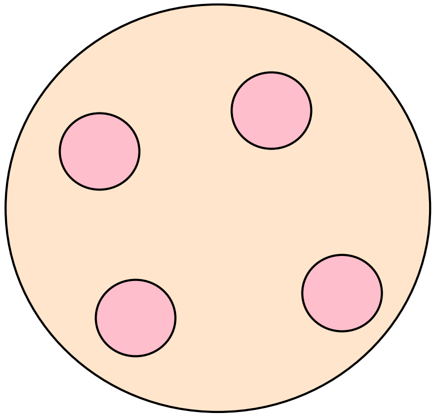
Pick (say) 4
remote points
for each
cluster.

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

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

p

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**

DBSCAN – Density-Based Spatial Clustering of Applications with Noise

DBSCAN

Density-based Clustering locates regions of high density that are separated from one another by regions of low density.

- Density = number of points within a specified radius (Eps)

- DBSCAN is a density-based algorithm.
  - A point is a core point if it has more than a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
  - A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point

DBSCAN

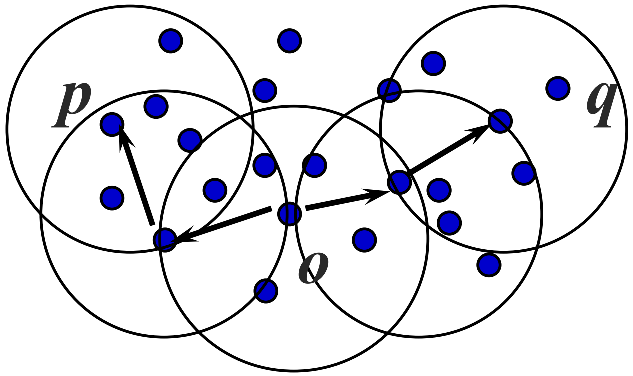
- A noise point is any point that is not a core point or a border point.
- Any two core points are close enough– within a distance Eps of one another – are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are discarded

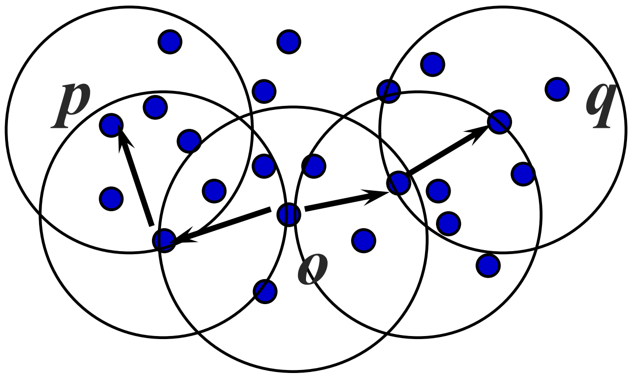
DBSCAN: The Algorithm

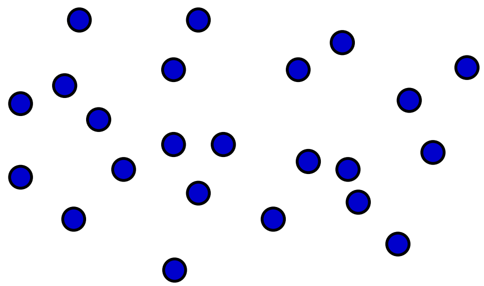
- select a point ***p***
- Retrieve all points density-reachable from ***p*** wrt ε and ***MinPts***.
- If ***p*** is a core point, a cluster is formed.
- If ***p*** is a border point, no points are density-reachable from ***p*** and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

Result is independent of the order of processing the points

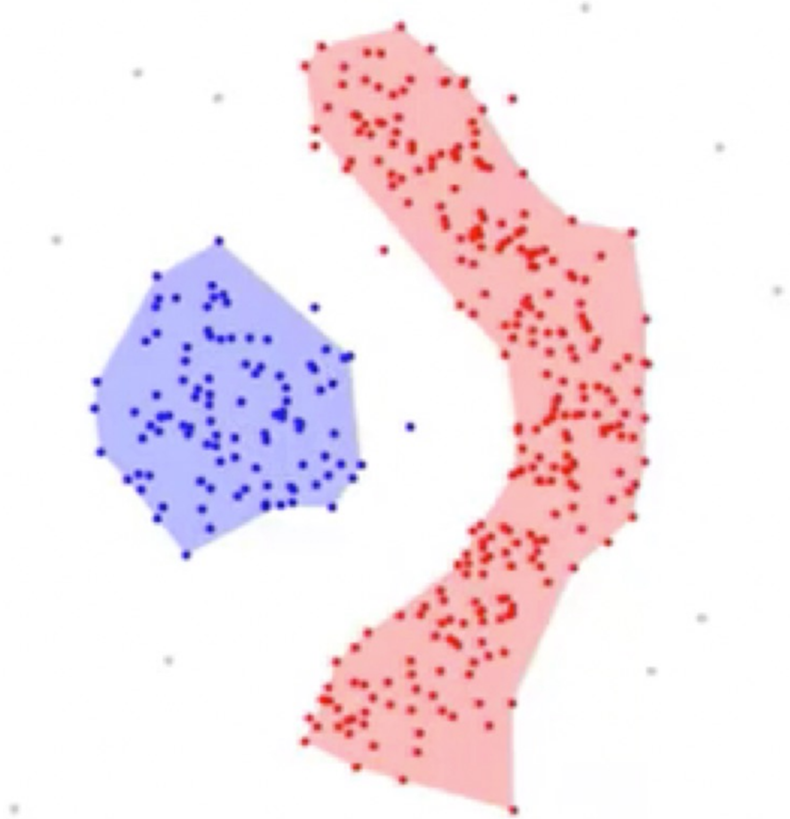
An example







DB Scan Clusters



DBSCAN Disadvantage: Irregular Density