

# Lecture 10: Clustering

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

<https://shuaili8.github.io>

<https://shuaili8.github.io/Teaching/VE445/index.html>



# Outline

- Unsupervised learning
- Clustering
- K-means
  - Algorithm
  - How to choose  $K$
  - Initialization
  - Properties
- Agglomerative clustering
- BFR algorithm

# Unsupervised Learning

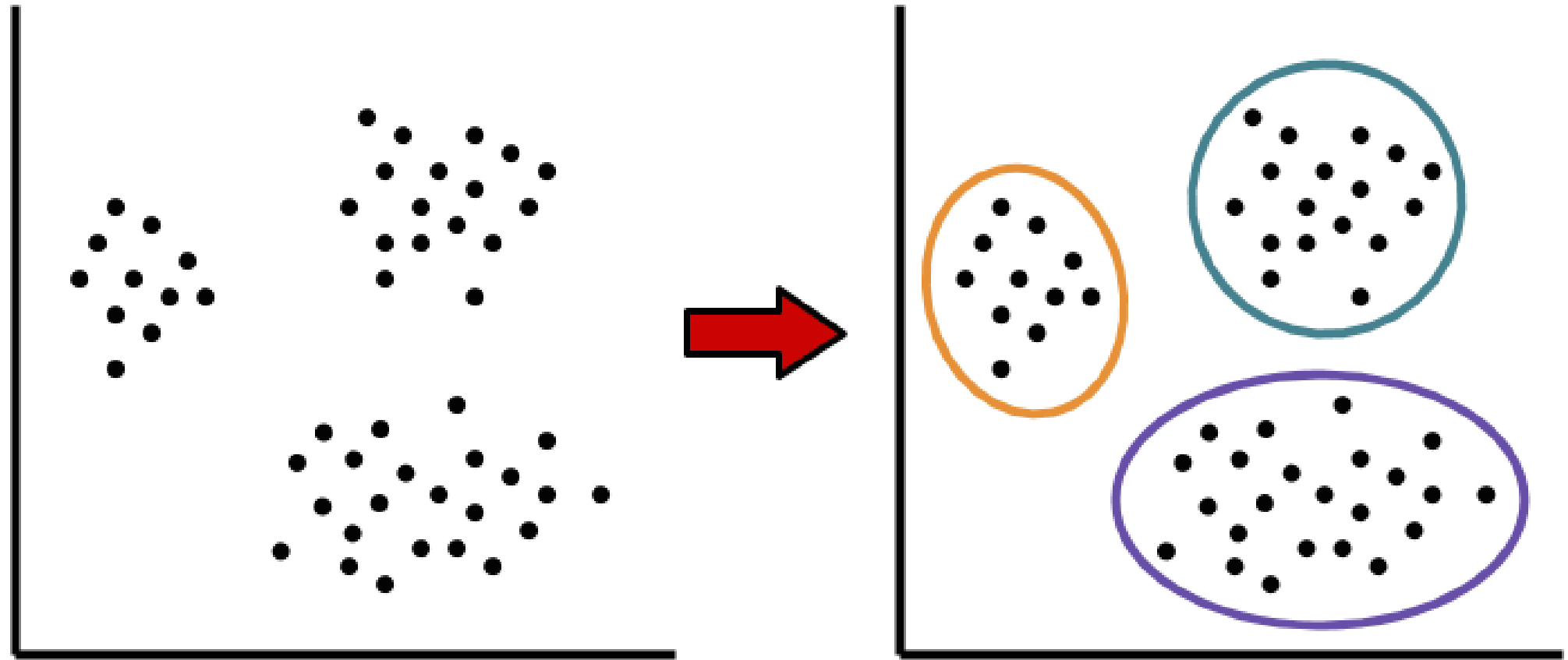
# Machine learning categories

- **Unsupervised learning**
  - No labeled data
- **Supervised learning**
  - Use labeled data to predict on unseen points
- **Semi-supervised learning**
  - Use labeled data and unlabeled data to predict on unlabeled/unseen points
- **Reinforcement learning**
  - Sequential prediction and receiving feedbacks

# Course outline

- Basics
- Supervised learning
  - Linear Regression
  - Logistic regression
  - SVM and Kernel methods
  - Decision Tree
- Deep learning
  - Neural Networks
  - Backpropagation
  - Convolutional Neural Network
  - Recurrent Neural Network
- Unsupervised learning
  - K-means, PCA, EM, GMM
- Reinforcement learning
  - Multi-armed bandits
  - MDP
  - Bellman equations
  - Q-learning
- Learning theory
  - PAC, VC-dimension, bias-variance decomposition

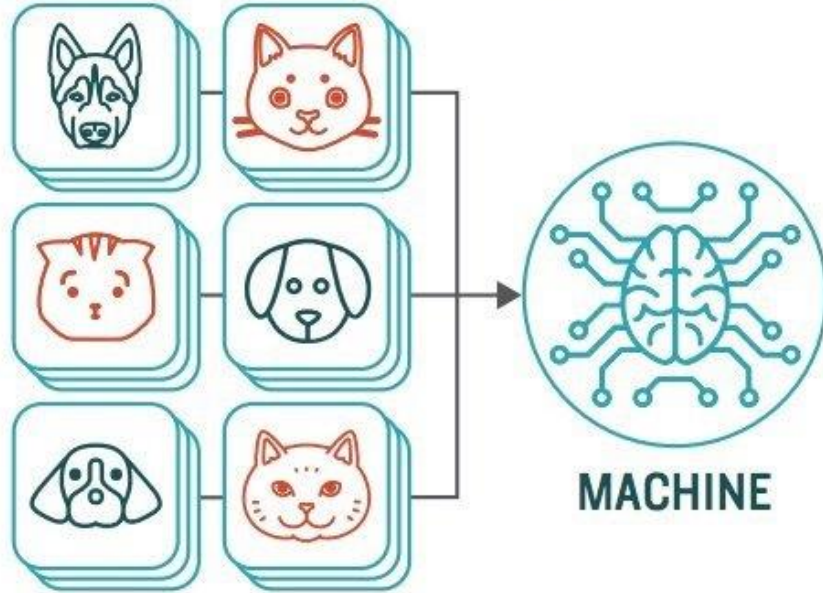
# Unsupervised learning example



# How **Unsupervised** Machine Learning Works

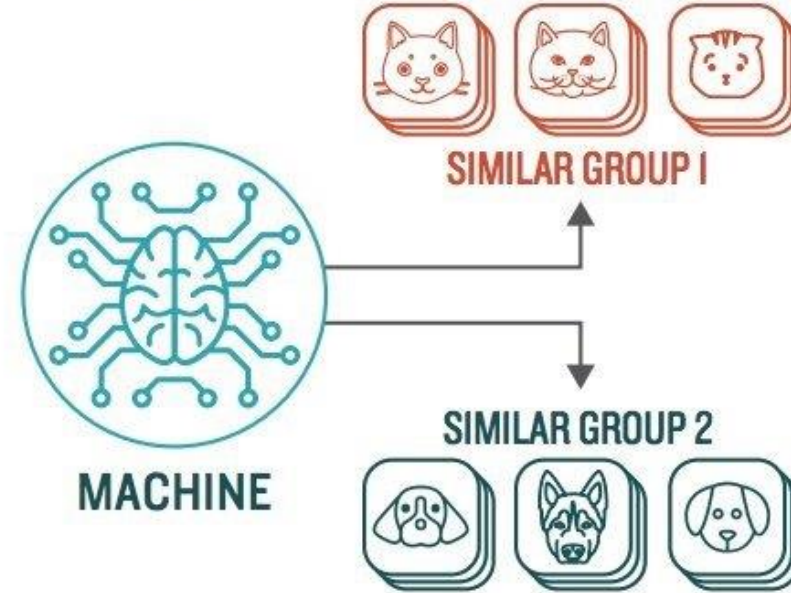
## STEP 1

Provide the machine learning algorithm uncategorized, unlabeled input data to see what patterns it finds

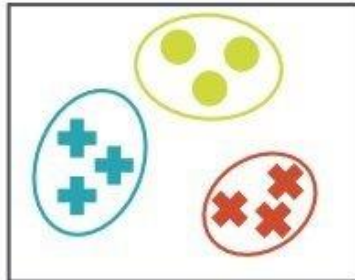


## STEP 2

Observe and learn from the patterns the machine identifies



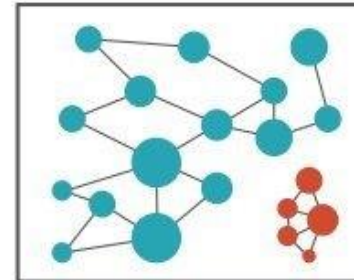
## TYPES OF PROBLEMS TO WHICH IT'S SUITED



### CLUSTERING

Identifying similarities in groups

*For Example:* Are there patterns in the data to indicate certain patients will respond better to this treatment than others?



### ANOMALY DETECTION

Identifying abnormalities in data

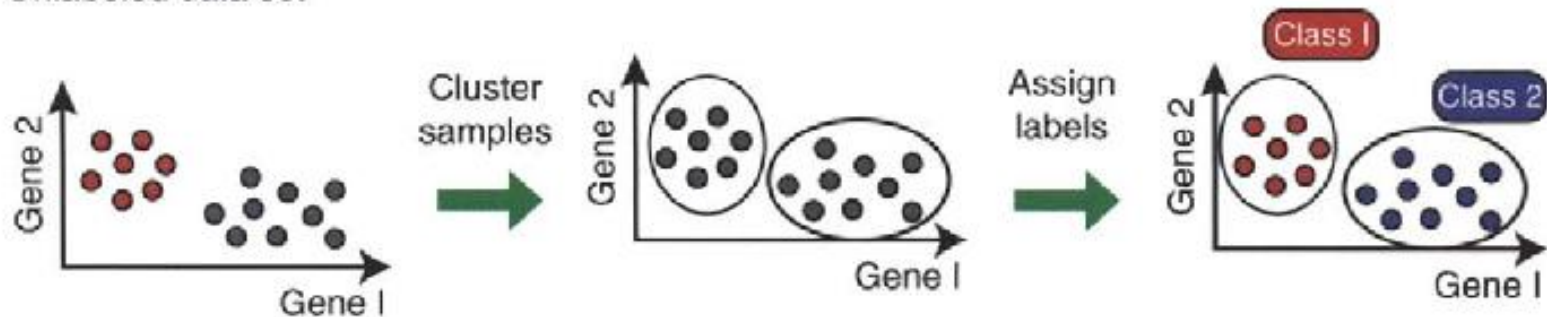
*For Example:* Is a hacker intruding in our network?

<b>Supervised Learning</b>	<b>Unsupervised Learning</b>
Input data is labelled	Input data is unlabeled
Uses training dataset	Uses just input dataset
Used for prediction	Used for analysis
Classification and regression	Clustering, density estimation and dimensionality reduction

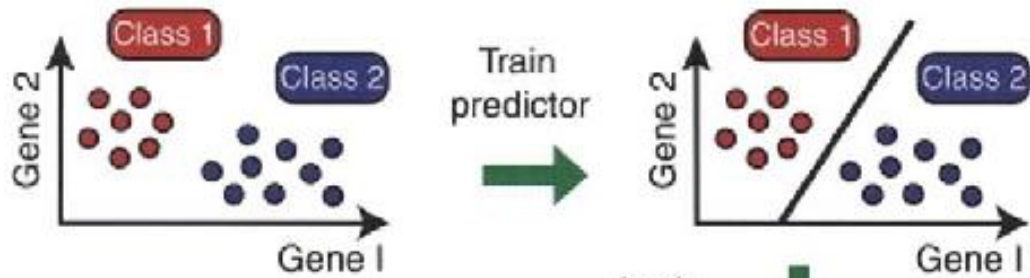


**A****Unsupervised**

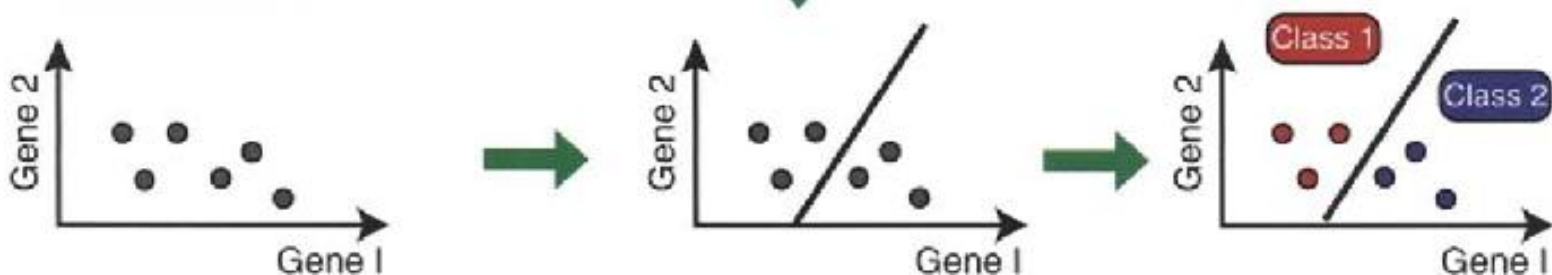
Unlabeled data set

**B****Supervised**

Labeled train set



Unlabeled test set



# Clustering

# Clustering

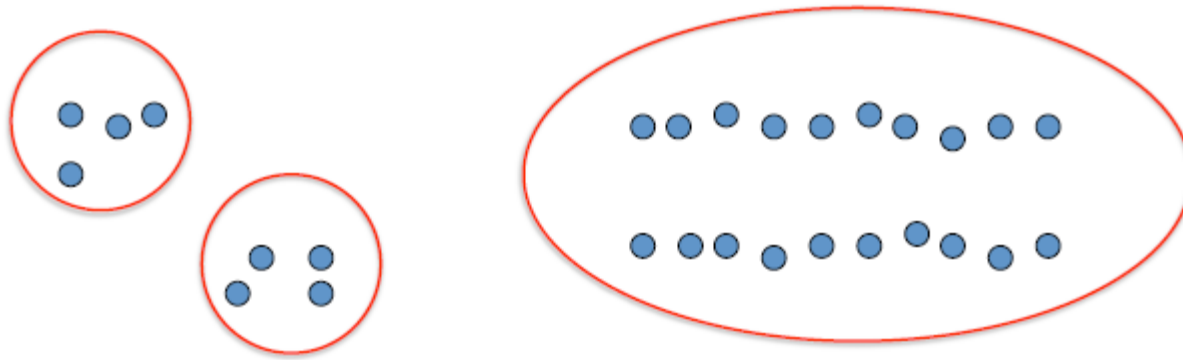
- Unsupervised learning
- Requires data, but no labels
- Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Regions of images
- Useful when don't know what you're looking for
- But: can get gibberish

# Clustering (cont.)

- Goal: Automatically segment data into groups of similar points
- Question: When and why would we want to do this?
- Useful for:
  - Automatically organizing data
  - Understanding hidden structure in some data
  - Representing high-dimensional data in a low-dimensional space
- Examples: Cluster
  - customers according to purchase histories
  - genes according to expression profile
  - search results according to topic
  - Facebook users according to interests
  - a museum catalog according to image similarity

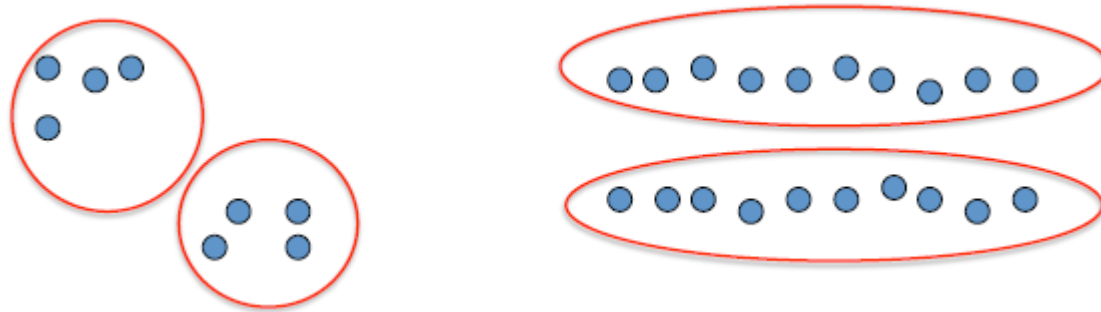
# Intuition

- Basic idea: group together similar instances
- Example: 2D point patterns



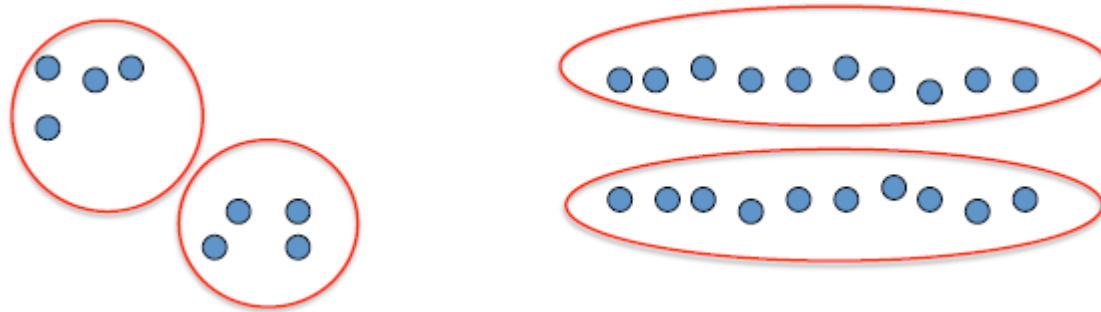
# Intuition (cont.)

- Basic idea: group together similar instances
- Example: 2D point patterns



# Intuition (cont.)

- Basic idea: group together similar instances
- Example: 2D point patterns



- What could “similar” mean?
  - – One option: small Euclidean distance (squared)
  - – Clustering results are crucially dependent on the measure of similarity (or **distance**) between “points” to be clustered

# Set-up

- Given the data:  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

- Each data point  $\mathbf{x}$  is  $p$ -dimensional:

$$\mathbf{x}_n = \langle x_{n,1}, \dots, x_{n,p} \rangle.$$

- Define a distance function between data:

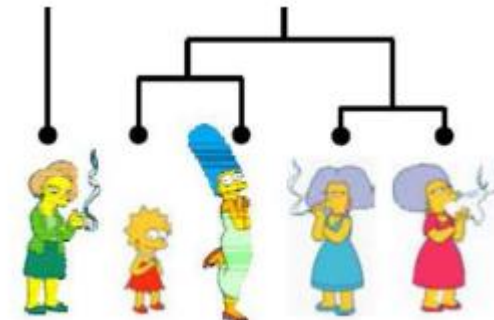
$$d(\mathbf{x}_n, \mathbf{x}_m).$$

- Goal: segment the data into  $K$  groups



# Clustering algorithms

- Partition algorithms (flat clustering)
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering
- Hierarchical algorithms
  - Bottom up-agglomerative
  - Top down-divisive



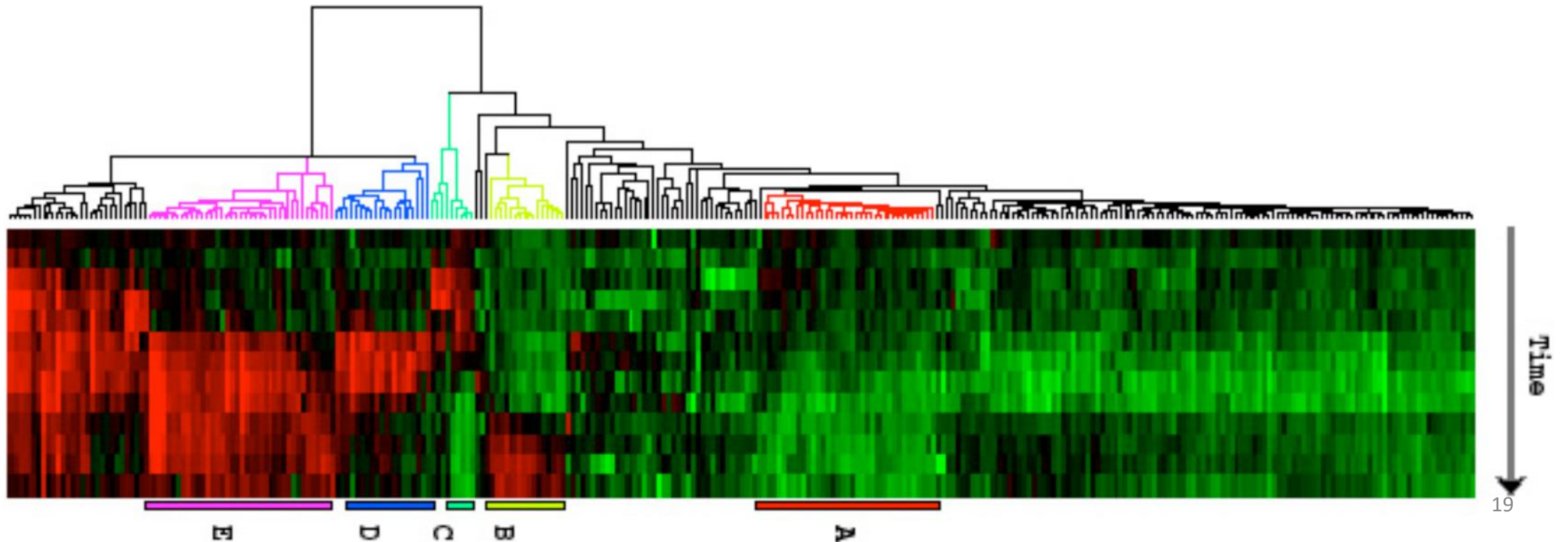
# Example

- Image segmentation
- Goal: Break up the image into meaningful or perceptually similar regions



# Example 2

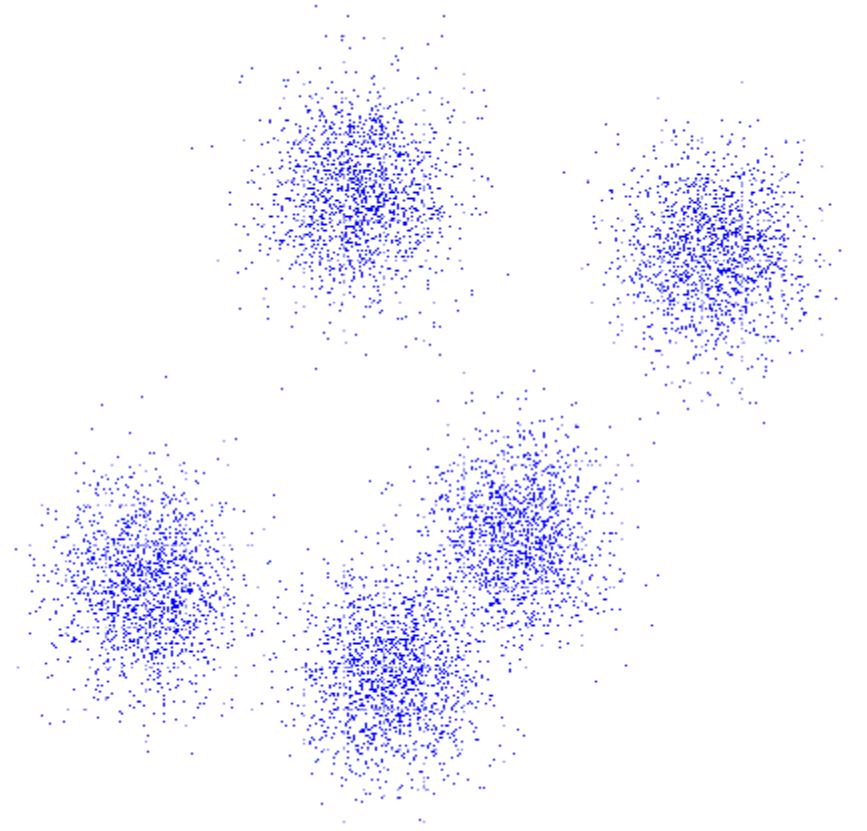
- Gene expression data clustering
  - Activity level of genes across time



# K-Means

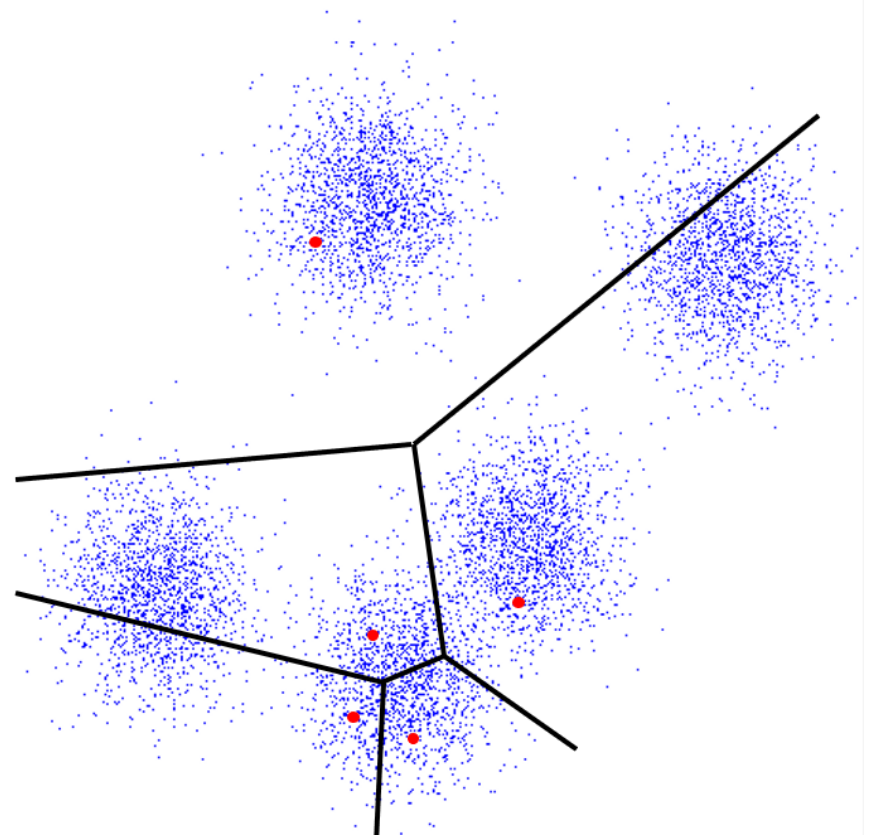
# K-Means

- An iterative clustering algorithm
- **Initialize:** Pick  $K$  **random** points as cluster centers
- **Alternate:**
  - Assign data points to closest cluster center
  - Change the cluster center to the average of its assigned points
- **Stop:** when no points' assignments change



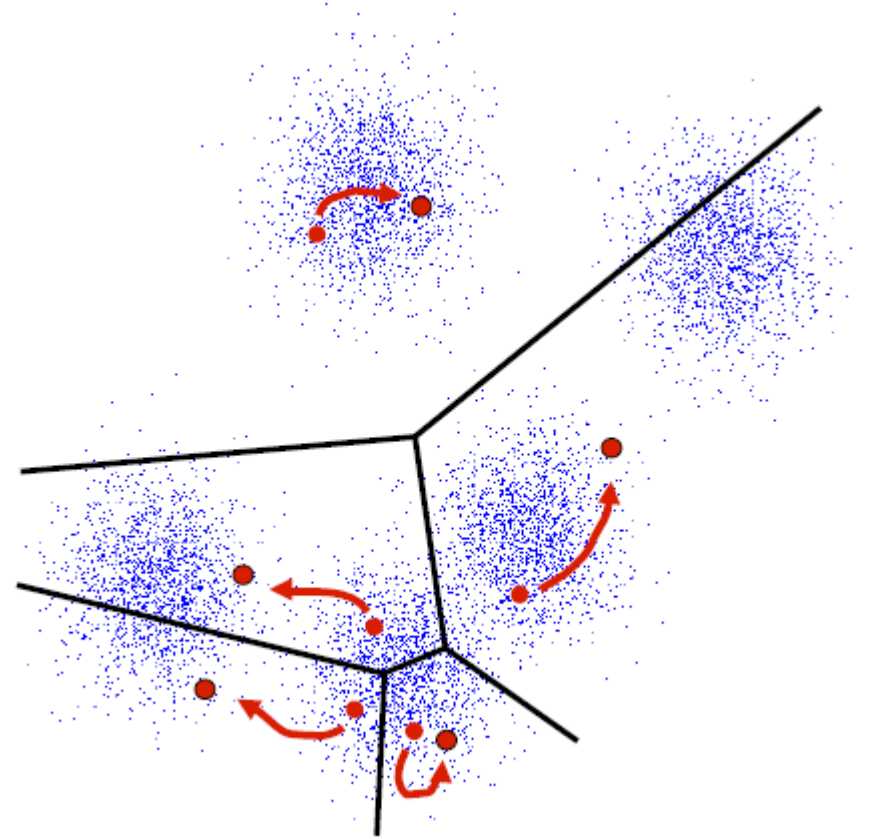
# K-Means (cont.)

- An iterative clustering algorithm
- **Initialize**: Pick  $K$  random points as cluster centers
- **Alternate**:
  - Assign data points to closest cluster center
  - Change the cluster center to the average of its assigned points
- **Stop**: when no points' assignments change



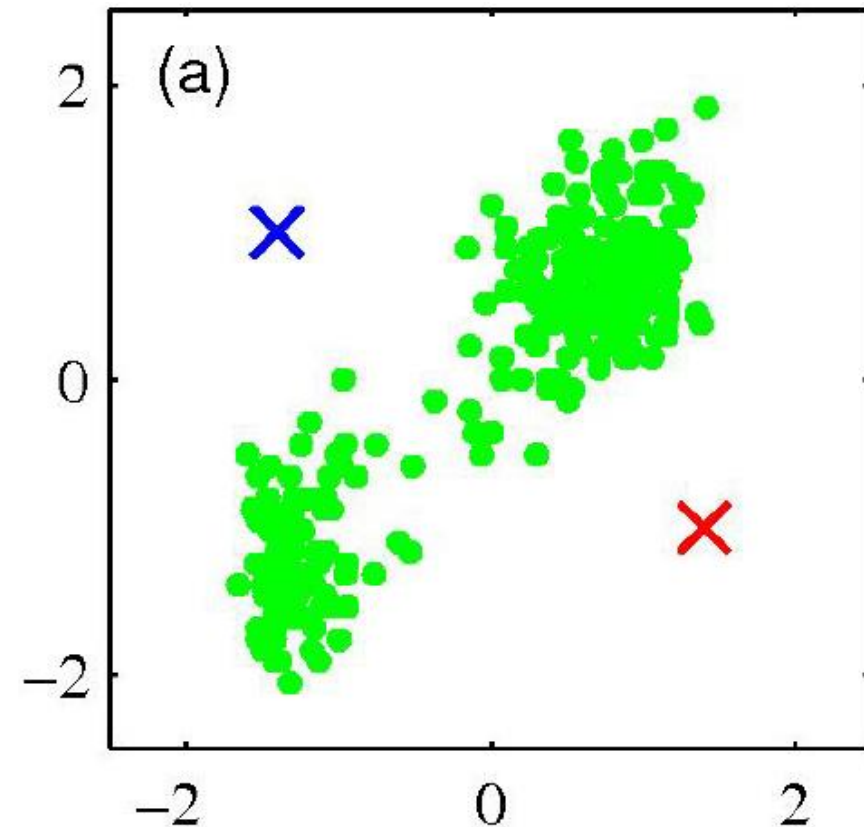
# K-Means (cont.)

- An iterative clustering algorithm
- **Initialize**: Pick  $K$  random points as cluster centers
- **Alternate**:
  - Assign data points to closest cluster center
  - Change the cluster center to the average of its assigned points
- **Stop**: when no points' assignments change



# Example

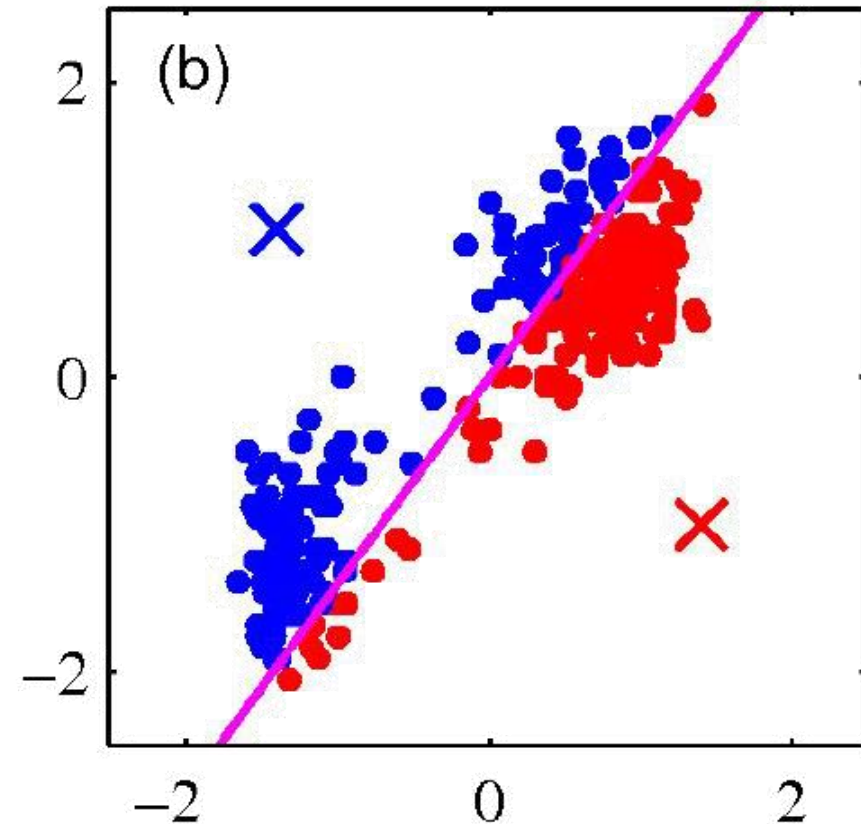
- Pick  $K$  random points as cluster centers (means)
- Shown here for  $K = 2$





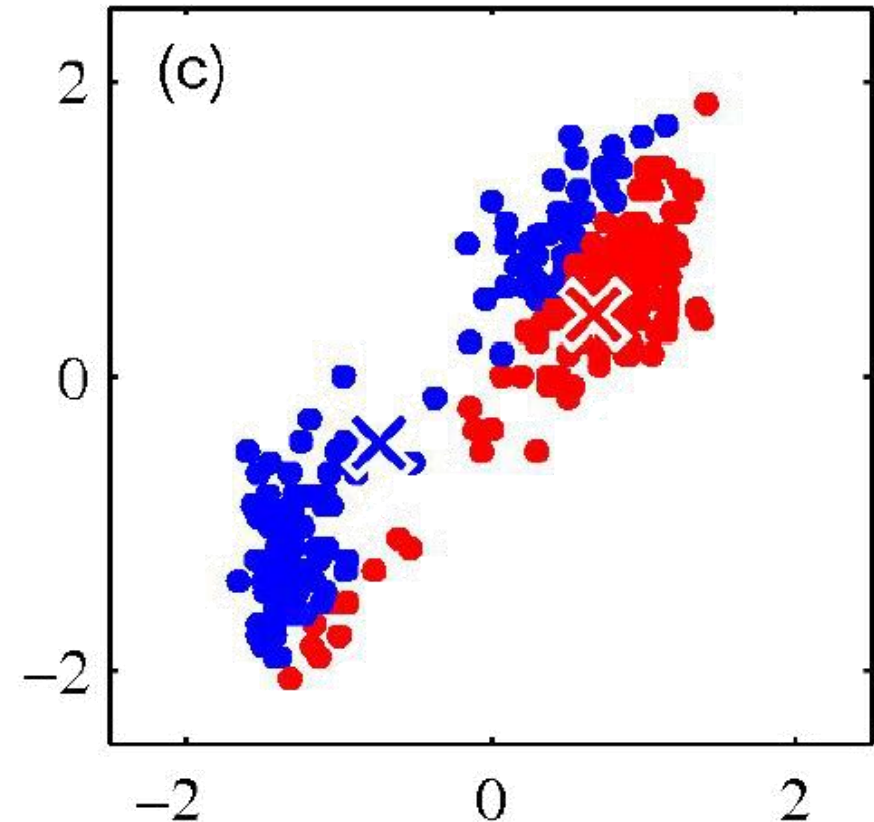
## Example (cont.)

- Iterative step 1
- Assign data points to closest cluster center



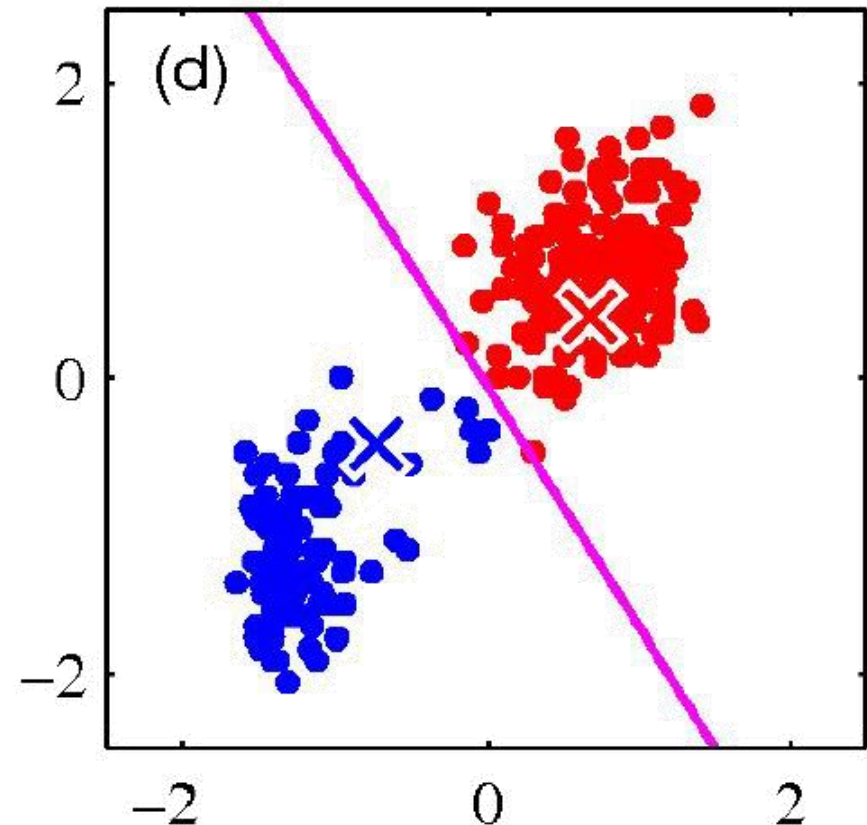
## Example (cont.)

- Iterative step 2
- Change the cluster center to the average of the assigned points



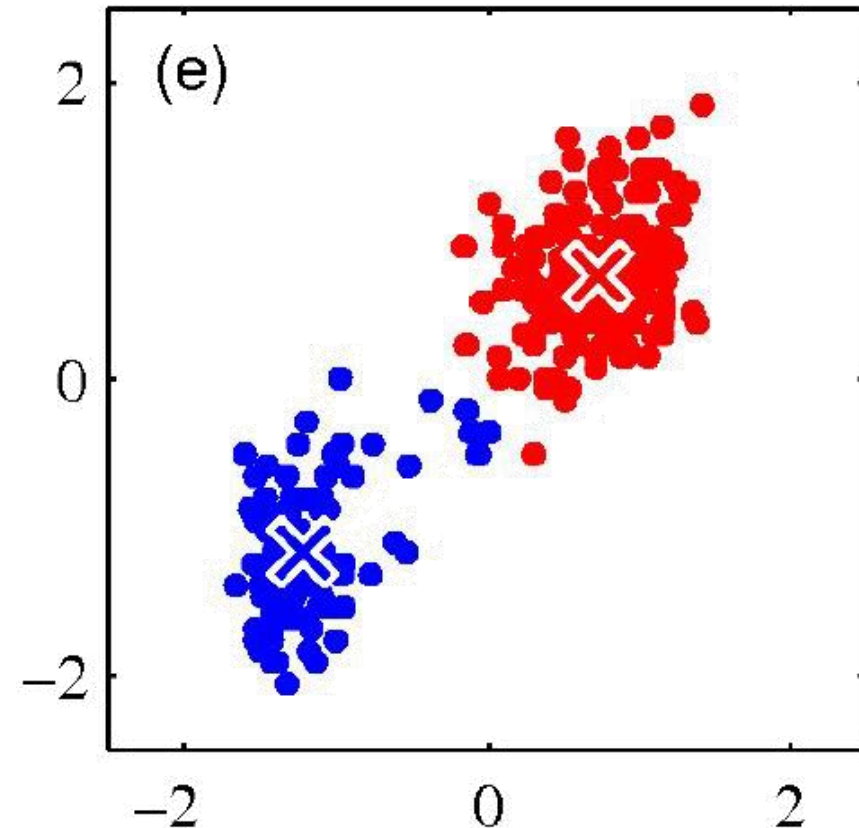
## Example (cont.)

- Repeat until convergence
- Convergence means that the differences of the center positions in two continuous loops is smaller than a threshold



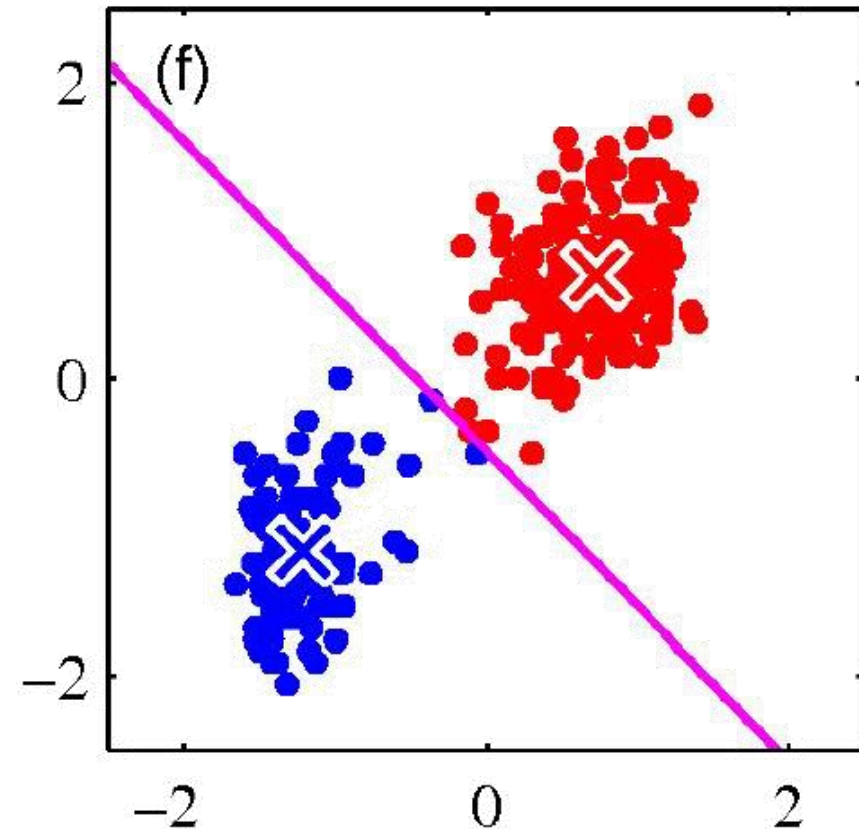
## Example (cont.)

- Repeat until convergence
- Convergence means that the differences of the center positions in two continuous loops is smaller than a threshold



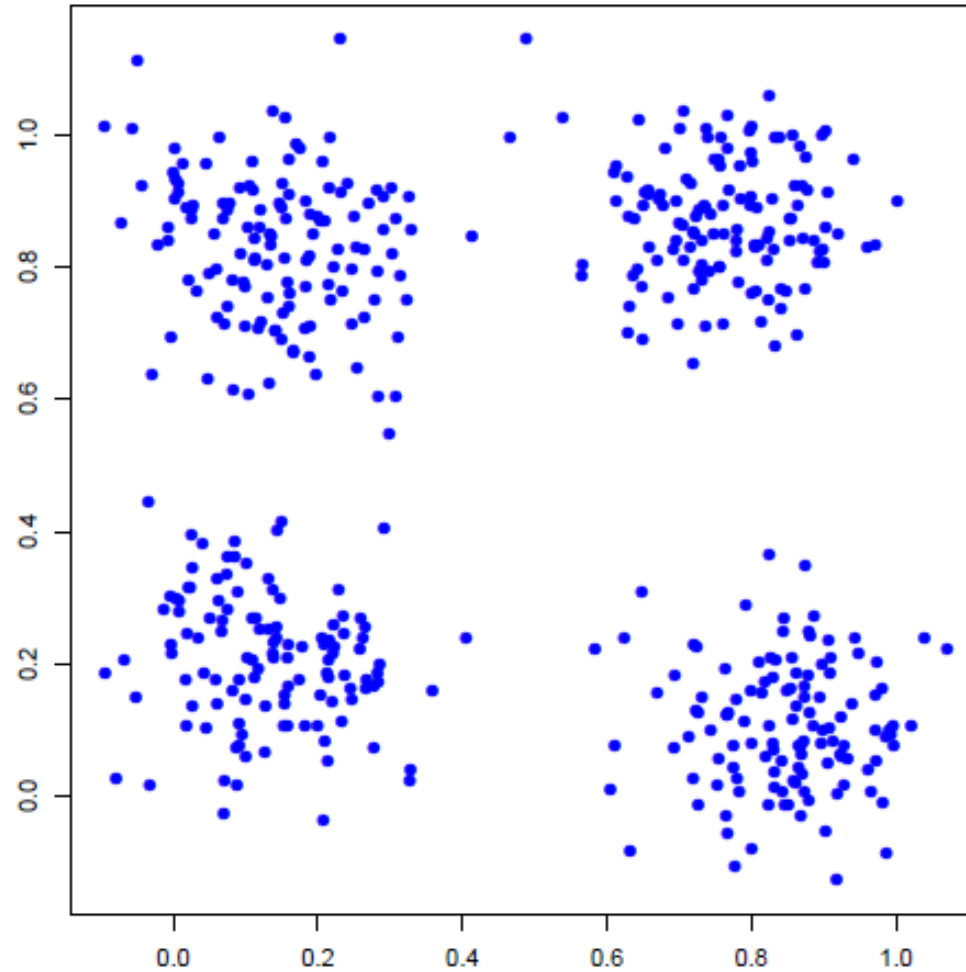
## Example (cont.)

- Repeat until convergence
- Convergence means that the differences of the center positions in two continuous loops is smaller than a threshold



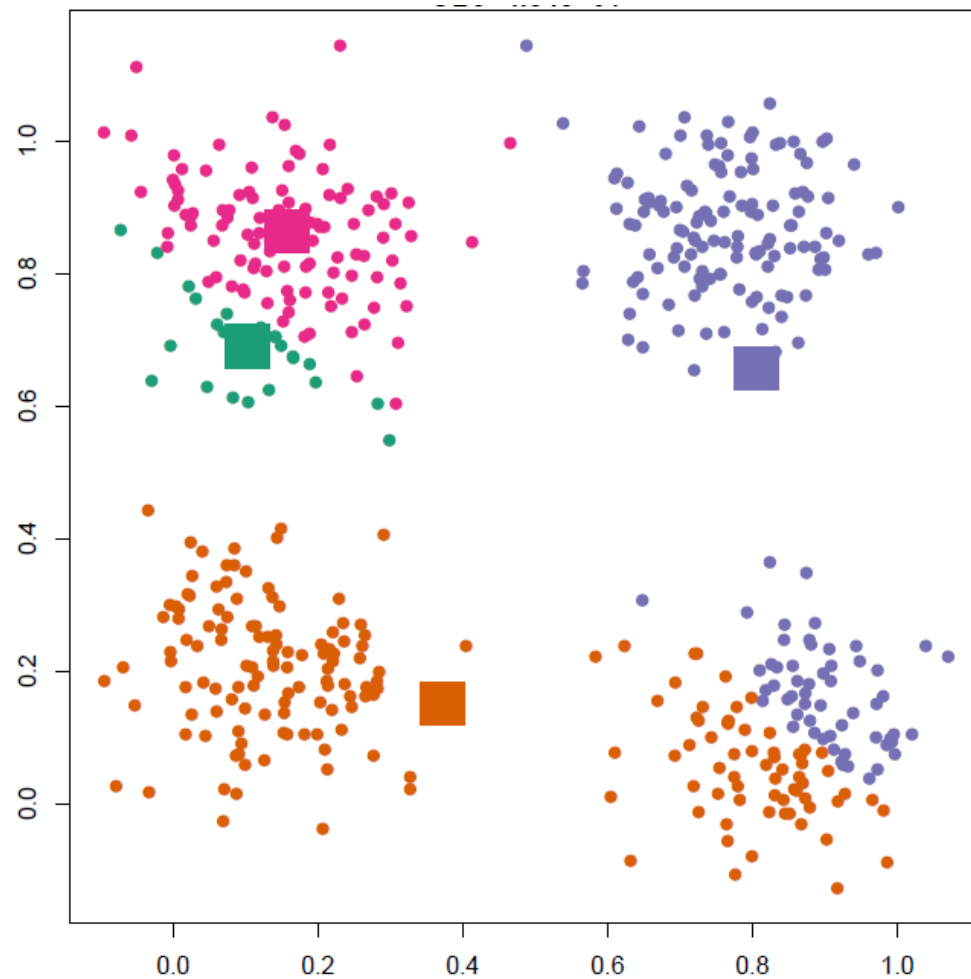
# Example 2

- $K = 4$



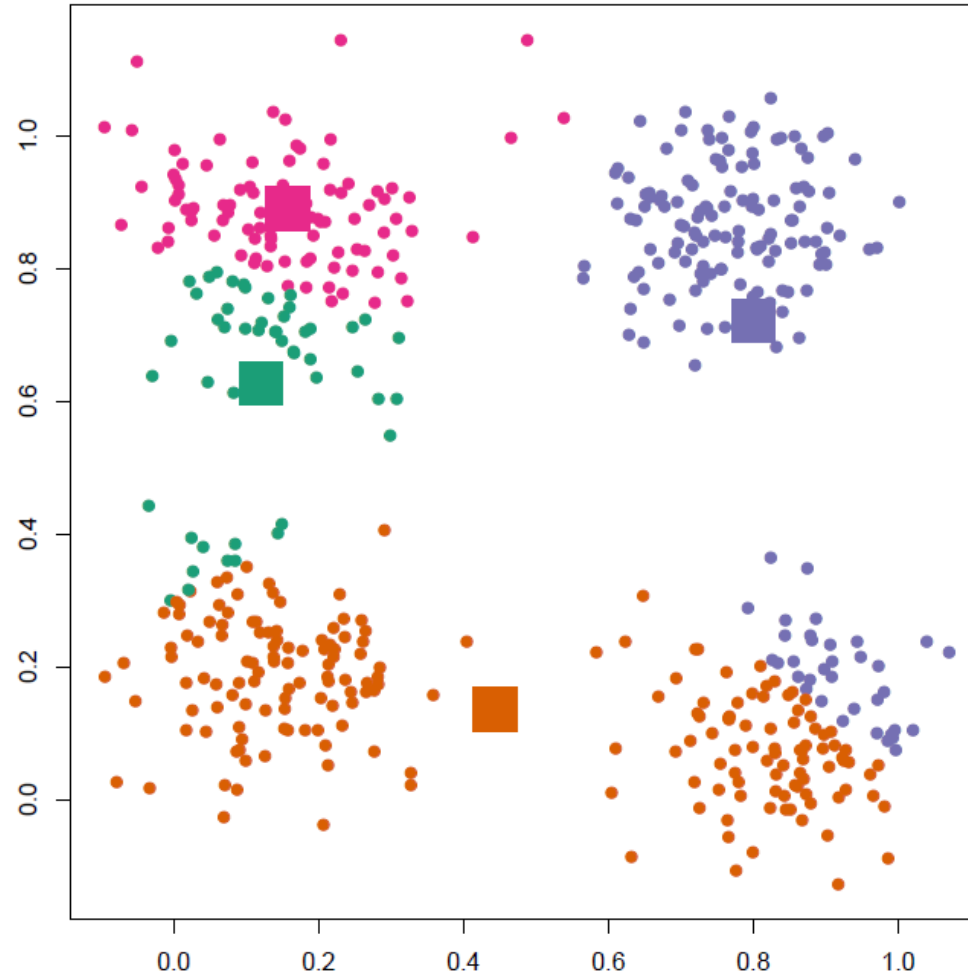
## Example 2 (cont.)

- $K = 4$



## Example 2 (cont.)

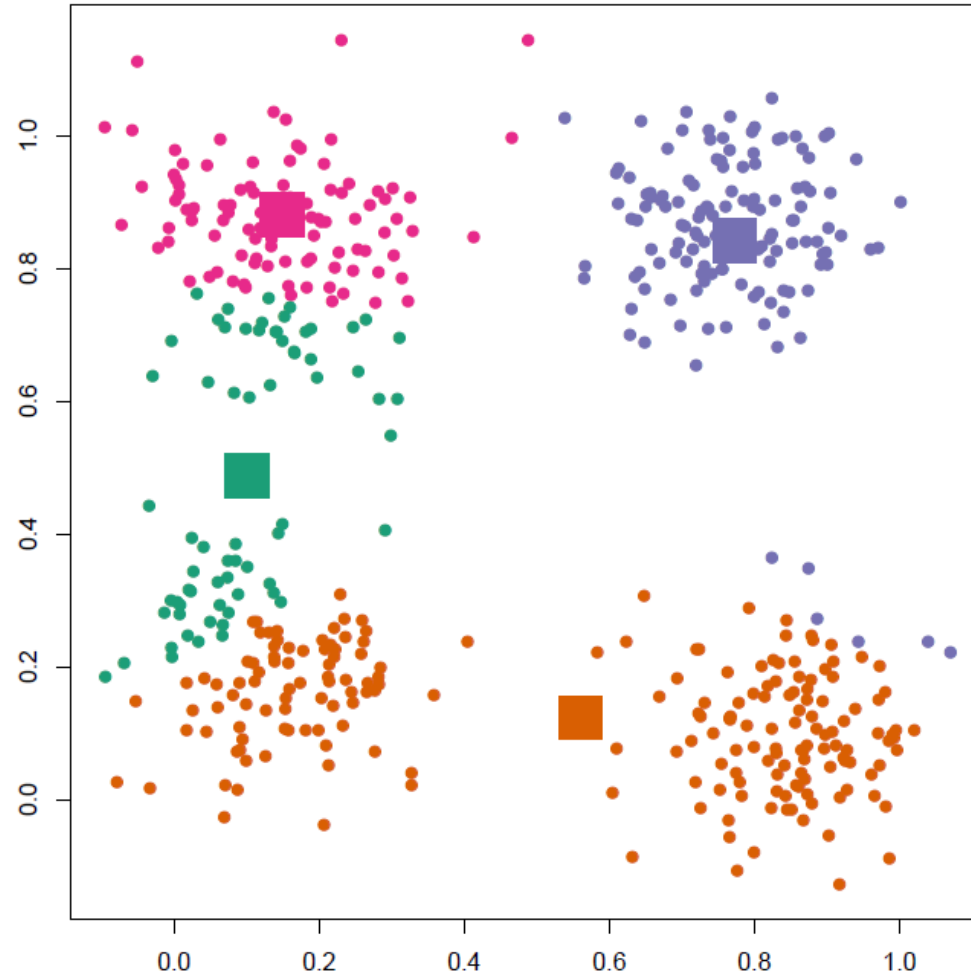
- $K = 4$





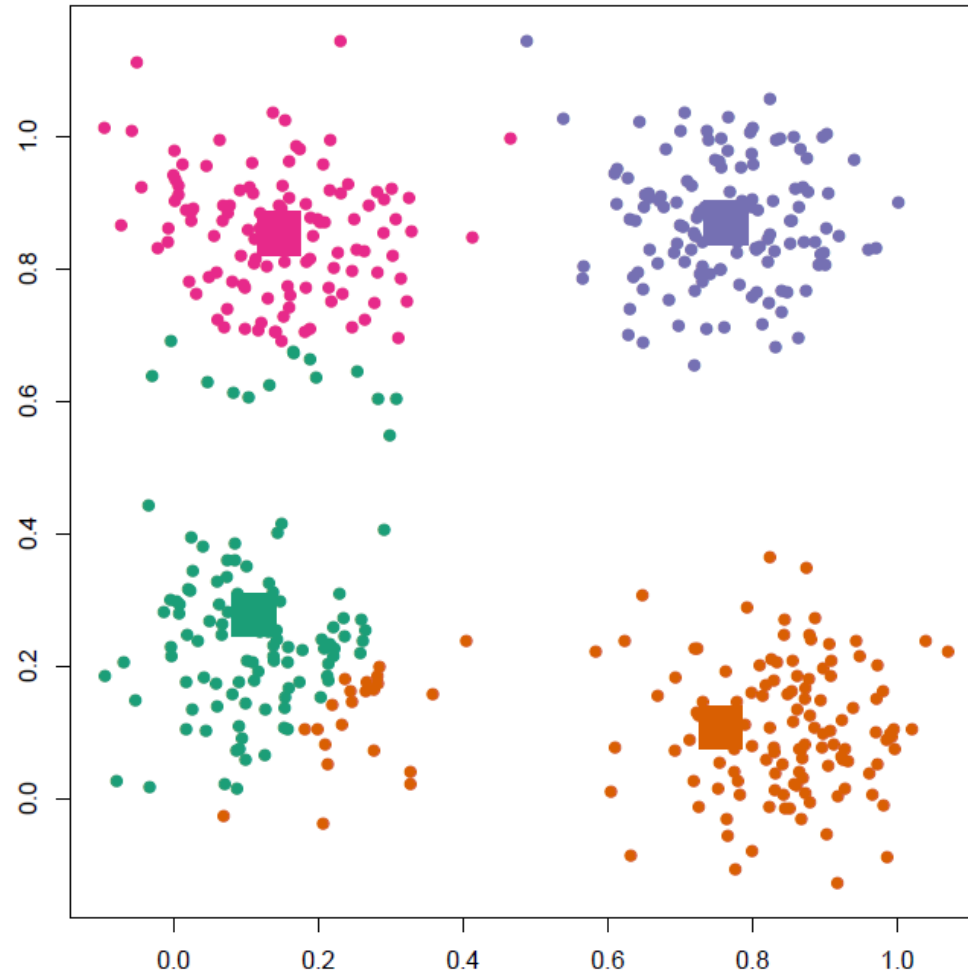
## Example 2 (cont.)

- $K = 4$



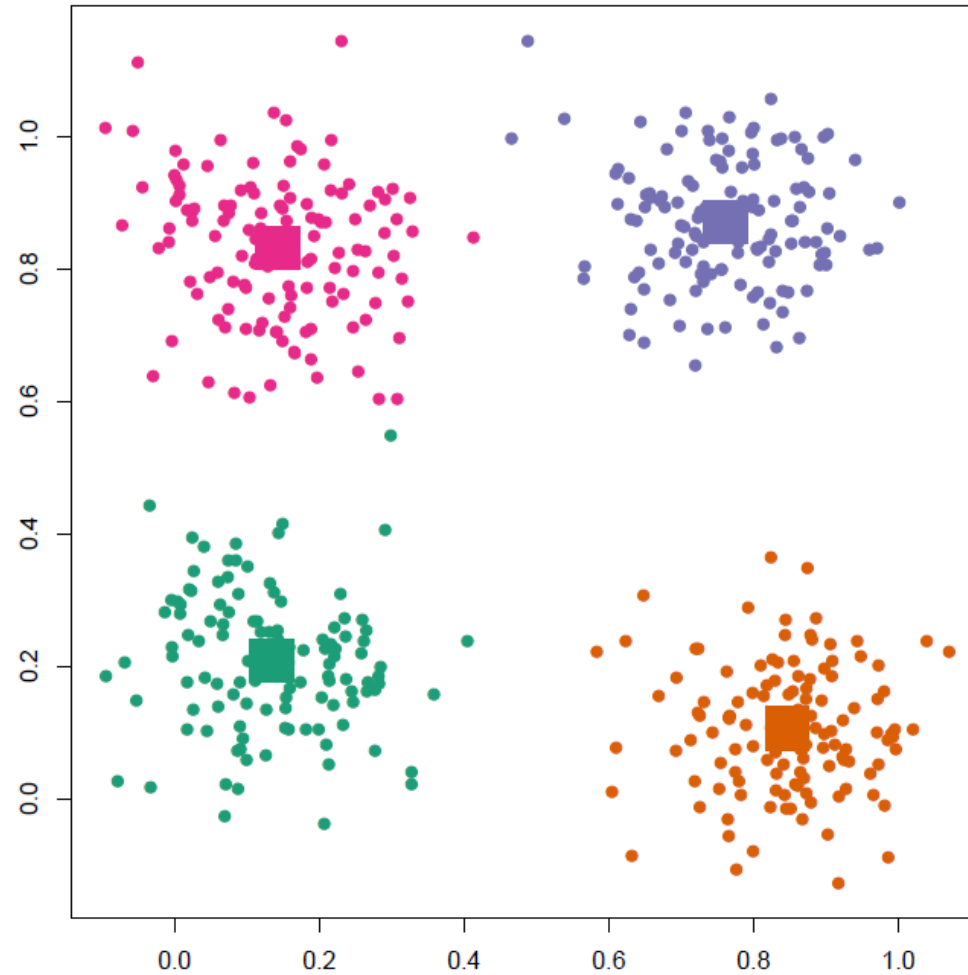
## Example 2 (cont.)

- $K = 4$



## Example 2 (cont.)

- $K = 4$



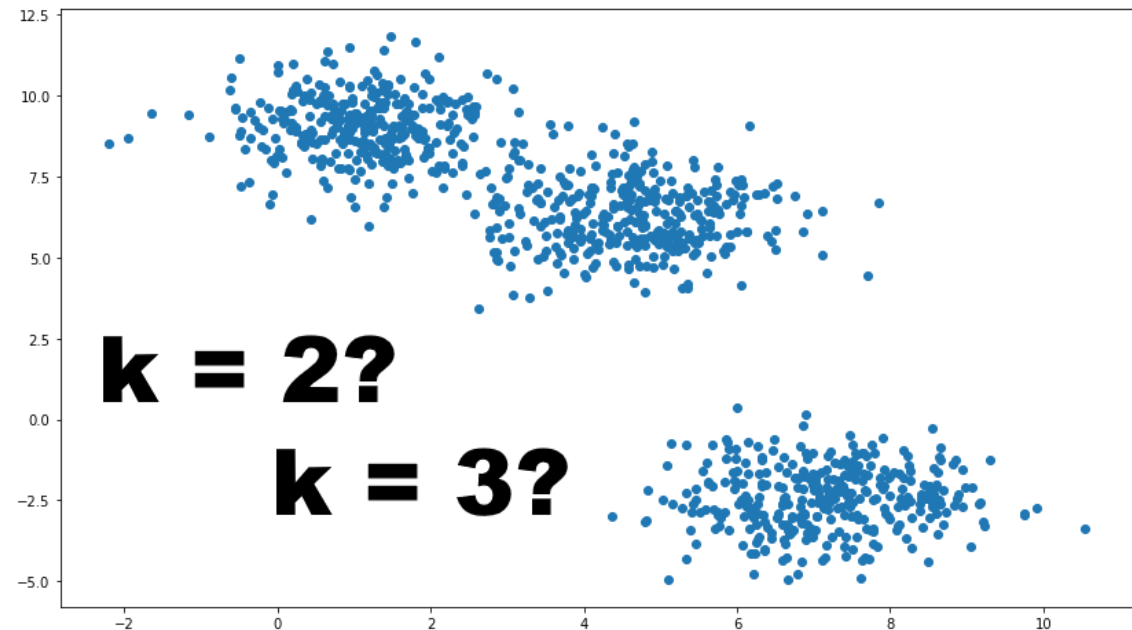
# Remained Questions in K-Means

# Remained questions in K-means

- Although the workflow of K-means is straight forward, there are some important questions that need to be discussed
- How to choose the hyper-parameter  $K$ ?
- How to initialize?

# How to choose K?

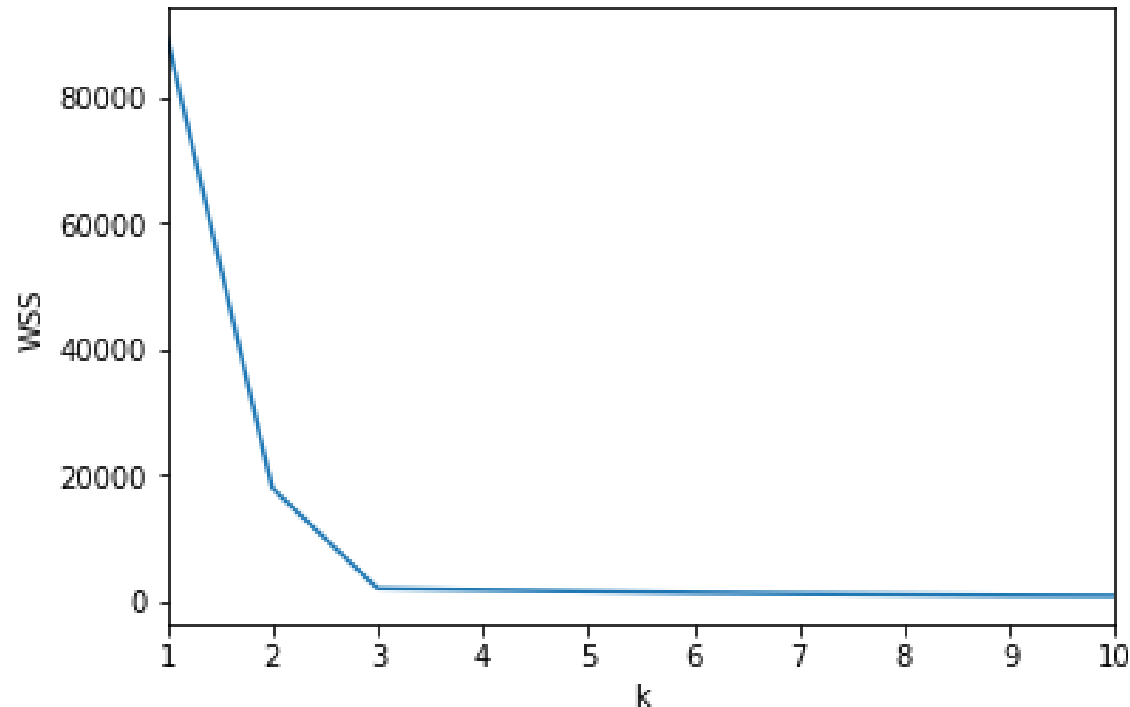
- K is the most important hyper-parameter in K-means which strongly affects its performance. In some situation, it's not an easy task to find the proper K
- The solution includes:
  - The elbow method
  - The silhouette method



# The elbow method

- Calculate the **Within-Cluster-Sum of Squared Errors (WSS)** for different values of  $K$ , and choose the  $K$  **for which WSS stops dropping significantly**. In the plot of WSS-versus-k, this is visible as an elbow

- Example:  $K = 3$



# The silhouette method

- The problem of the elbow method is that in many situations the most suitable  $K$  cannot be unambiguously identified. So we need the silhouette method
- The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation). The range of the silhouette value is between +1 and -1. A high value is desirable and indicates that the point is placed in the correct cluster



# The silhouette method (cont.)

- For each data point  $i \in C_k$ , let  $a(i)$  be its mean distance to all other points in the same cluster

$$a(i) = \frac{1}{|C_k| - 1} \sum_{j \in C_k, i \neq j} d(i, j)$$

- And let  $b(i)$  be the smallest mean distance to other clusters

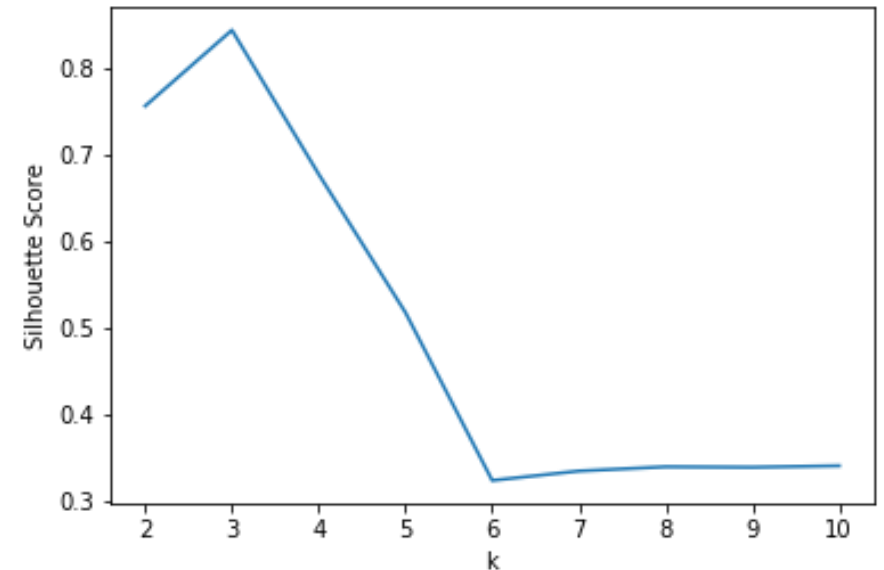
$$b(i) = \min_{l \neq k} \frac{1}{|C_l|} \sum_{j \in C_l} d(i, j)$$

# The silhouette method (cont.)

- The silhouette value of  $i \in C_k$  is defined as:

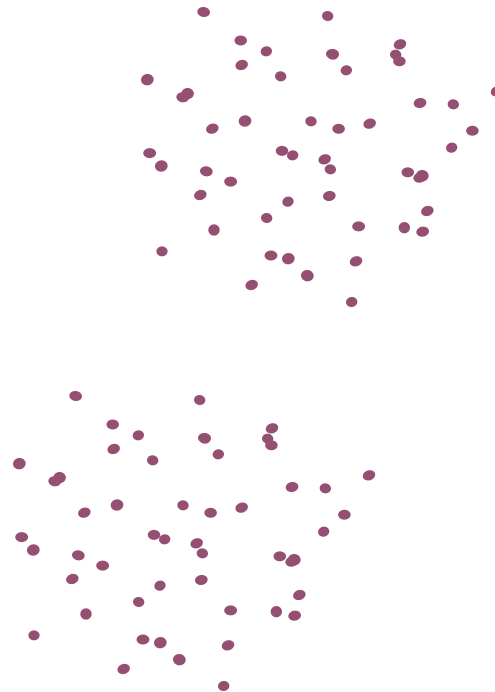
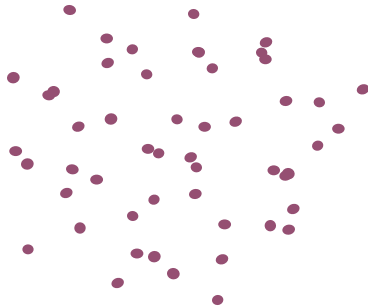
$$s(i) = \begin{cases} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, & \text{if } |C_k| > 1 \\ 0, & \text{if } |C_k| = 1 \end{cases}$$

- The silhouette score is the **average** of  $s(i)$  among all data
- Choose the  $k$  with the maximal silhouette score



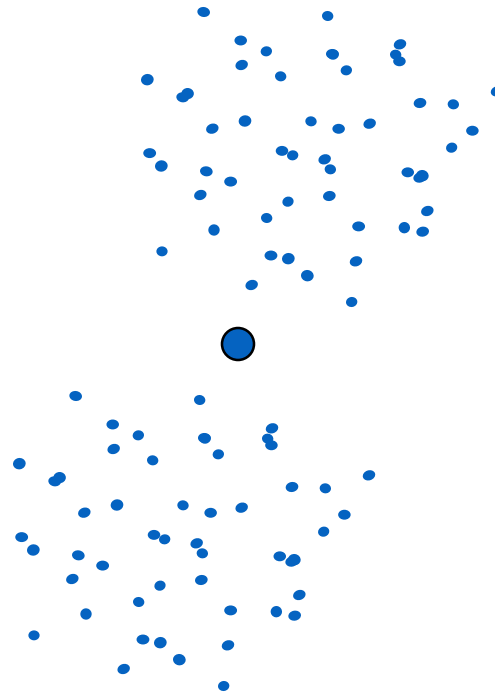
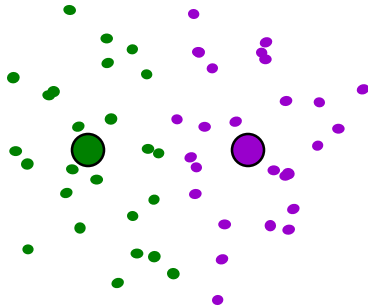
# How to initialize center positions?

- The positions of the centers in the stage of initialization are also very important in K-means algorithms. In some situations it can produce totally different clustering results
- Example:



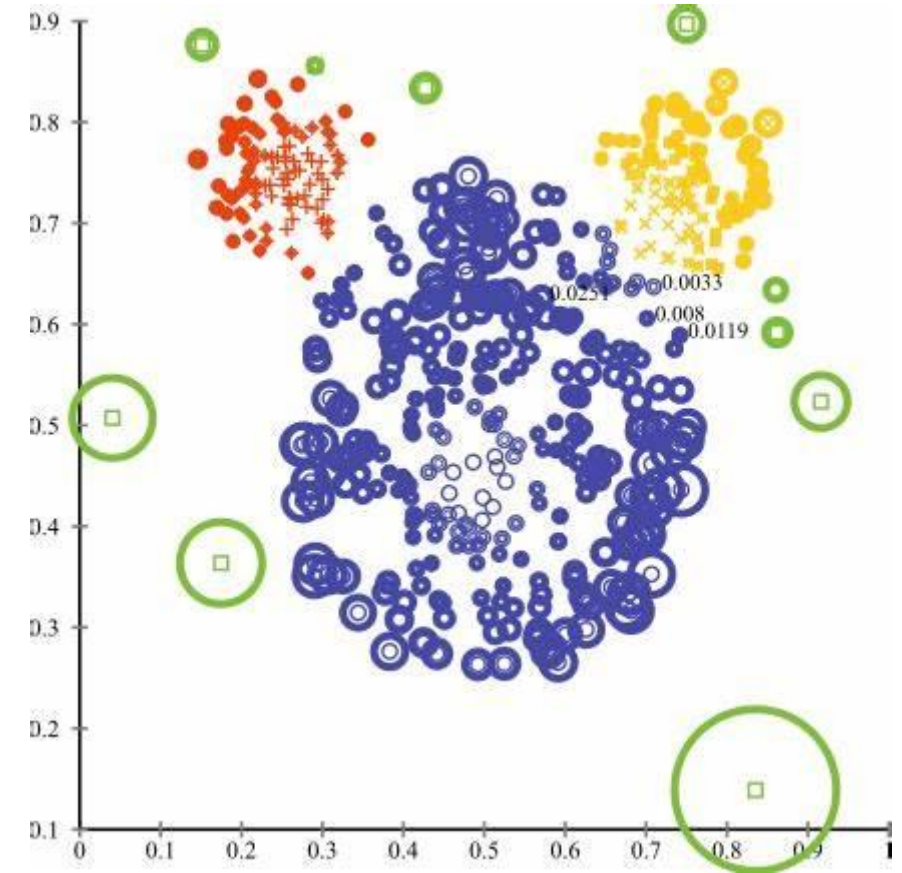
# How to initialize center positions? (cont.)

- The positions of the centers in the stage of initialization are also very important in K-means algorithms. In some situations it can produce totally different clustering results
- Example:



# A possible solution

- Pick one point at random, then  $K - 1$  other points, each as far away as possible from the previous points
  - OK, as long as there are no *outliers* (points that are far from any reasonable cluster)



# K-means++

1. The **first** centroid is chosen uniformly at random from the data points that we want to cluster. This is similar to what we do in K-Means, but instead of randomly picking all the centroids, we just pick one centroid here
2. Next, we compute the distance  $d_x$  is the nearest distance from data point  $x$  to the centroids that have already been chosen
3. Then, choose the new cluster center from the data points with the probability of  $x$  being proportional to  $d_x^2$
4. We then repeat steps 2 and 3 until  $K$  clusters have been chosen

# Example

- Suppose we have the following points and we want to make 3 clusters here:



## Example (cont.)

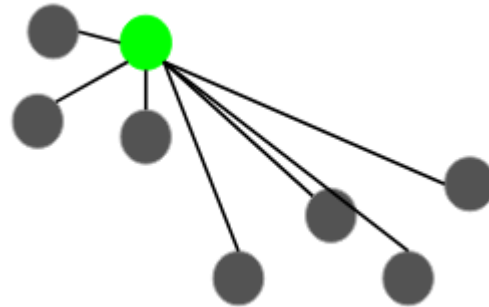
- First step is to randomly pick a data point as a cluster centroid:





## Example (cont.)

- Calculate the distance  $d_x$  of each data point with this centroid:



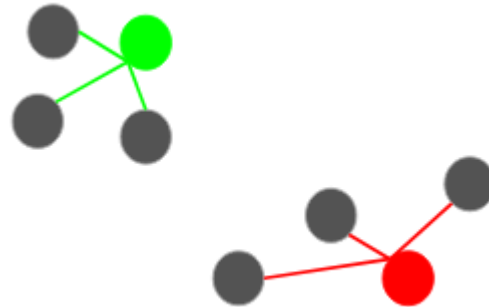
## Example (cont.)

- The next centroid will be sampled with the probability proportional to  $d_x^2$
- Say the sampled is the red one



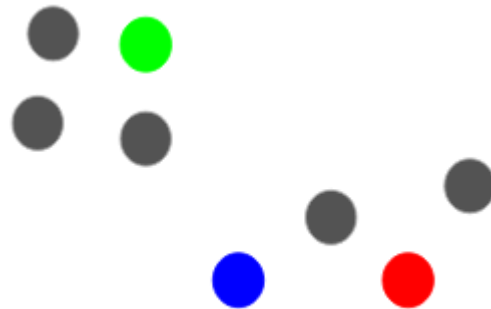
## Example (cont.)

- To select the last centroid, compute  $d_x$ , which is the distance to its closest centroid



## Example (cont.)

- Sample the one with the probability proportional to  $d_x^2$
- Say, the blue one



# Properties of K-Means

# How to measure the performance

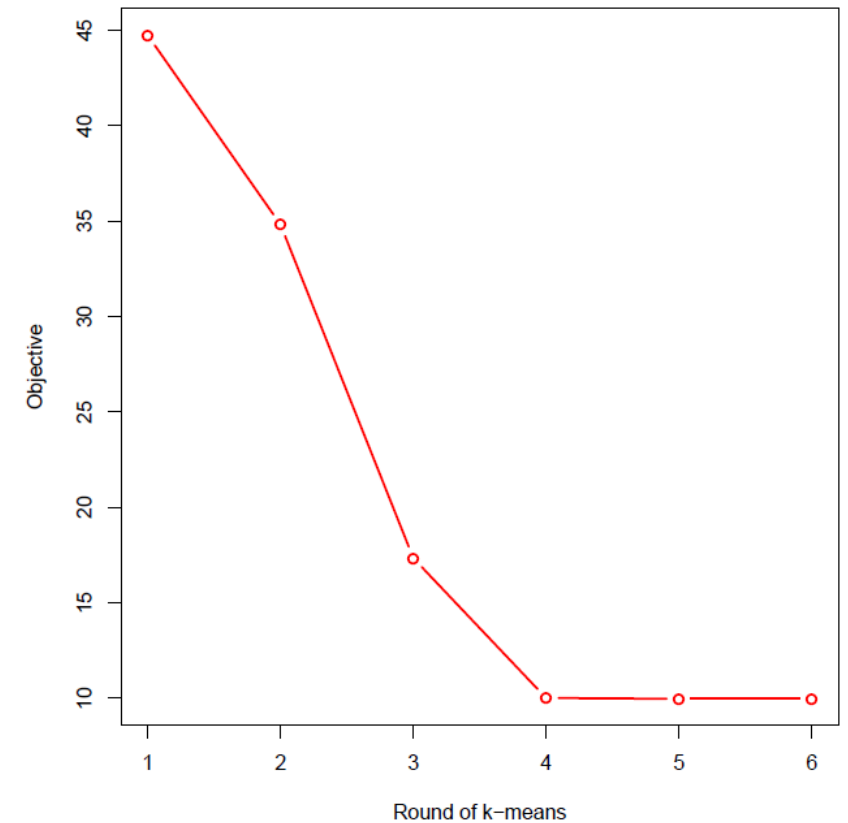
- K-means can be evaluated by the sum of distance from points to corresponding centers, or the WSS

number of clusters      number of cases      centroid for cluster  $j$

case  $i$

objective function  $\leftarrow J = \sum_{j=1}^k \sum_{i=1}^n \underbrace{\|x_i^{(j)} - c_j\|}_\text{Distance function}^2$

- The loss will approach zero when increase  $K$



# Properties of the K-means algorithm

- Guaranteed to converge in a finite number of iterations
- Running time per iteration:
  1. Assign data points to closest cluster center  
 $O(KN)$  time
  2. Change the cluster center to the average of its assigned points  
 $O(N)$

# Distance

- Distance is of crucial importance in K-means. So what kind of properties should the distance measure have?
- Symmetric
  - $D(A, B) = D(B, A)$
- Positivity, and self-similarity
  - $D(A, B) \geq 0$ , and  $D(A, B) = 0$  iff  $A = B$
- Triangle inequality
  - $D(A, B) + D(B, C) \geq D(A, C)$



# Convergence of K-means

## Objective

$$\min_{\mu} \min_C \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix  $\mu$ , optimize  $C$ :

$$\min_C \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2 = \min_c \sum_i^n |x_i - \mu_{x_i}|^2$$

**Step 1 of kmeans**

2. Fix  $C$ , optimize  $\mu$ :

$$\min_{\mu} \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

- Take partial derivative of  $\mu_i$  and set to zero, we have

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

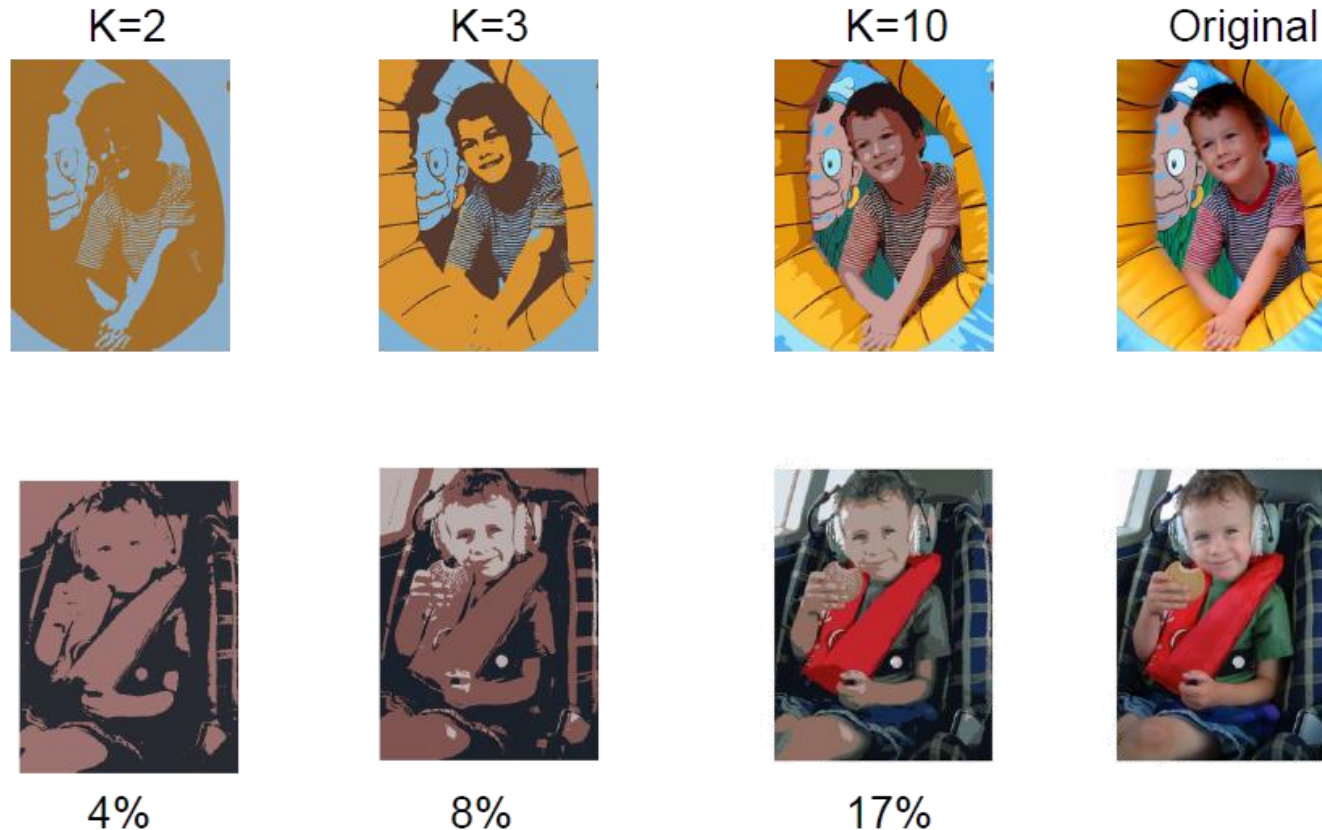
**Step 2 of kmeans**

Not guaranteed to converge to optimal

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

# Application: Segmentation

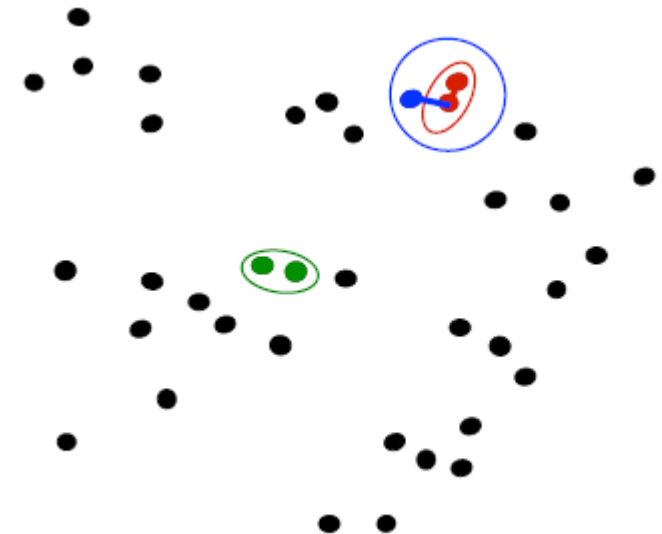
- Goal of segmentation is to partition an image into regions each of which has reasonably homogenous visual appearance
- Cluster the colors



# Agglomerative Clustering

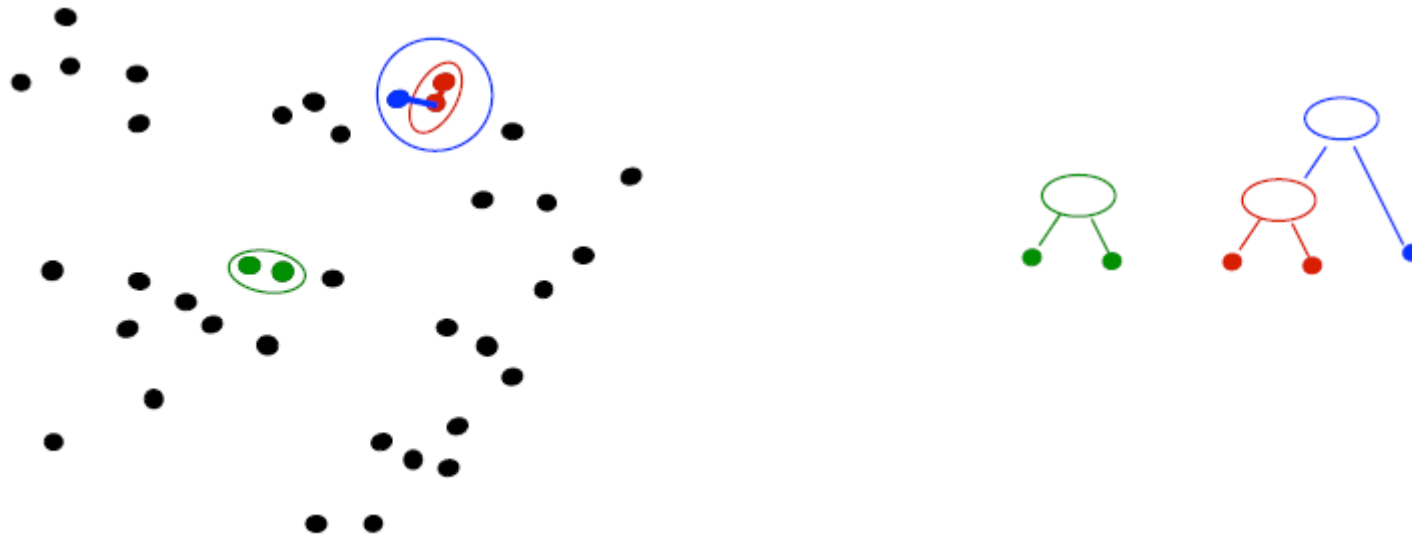
# Agglomerative clustering

- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters
- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left



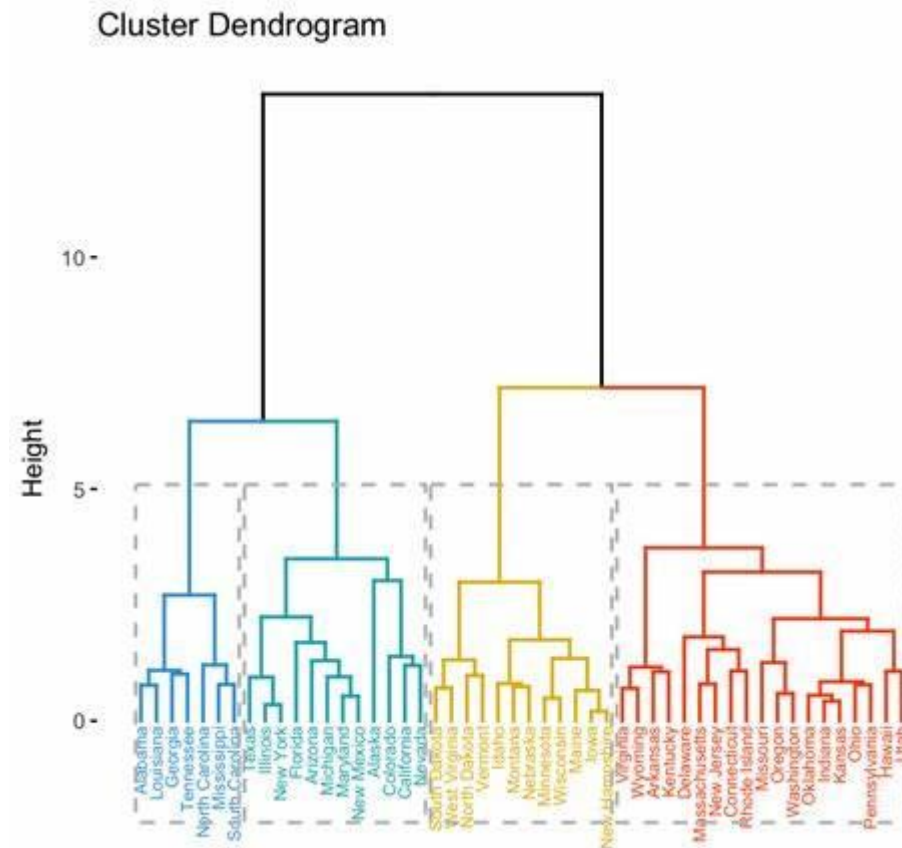
# Agglomerative clustering

- Produces not one clustering, but a family of clusterings represented by a dendrogram



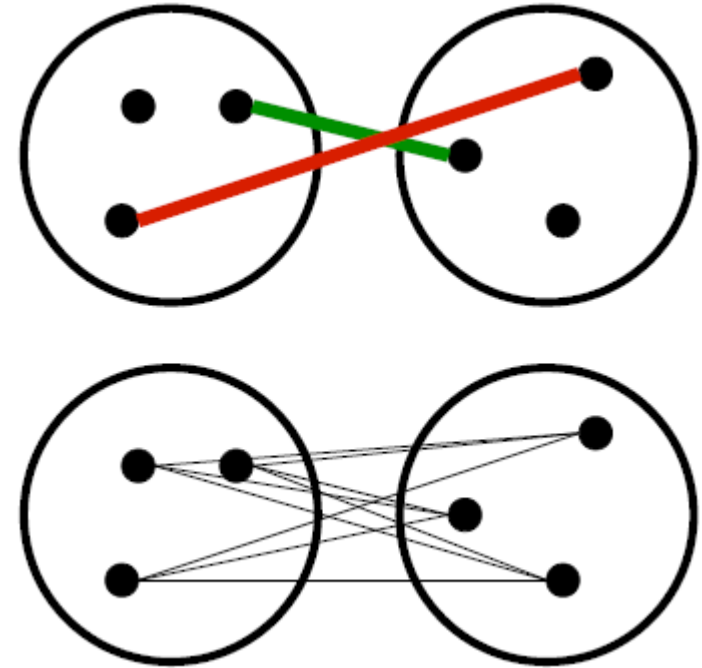
# Example

- Different heights give different clustering



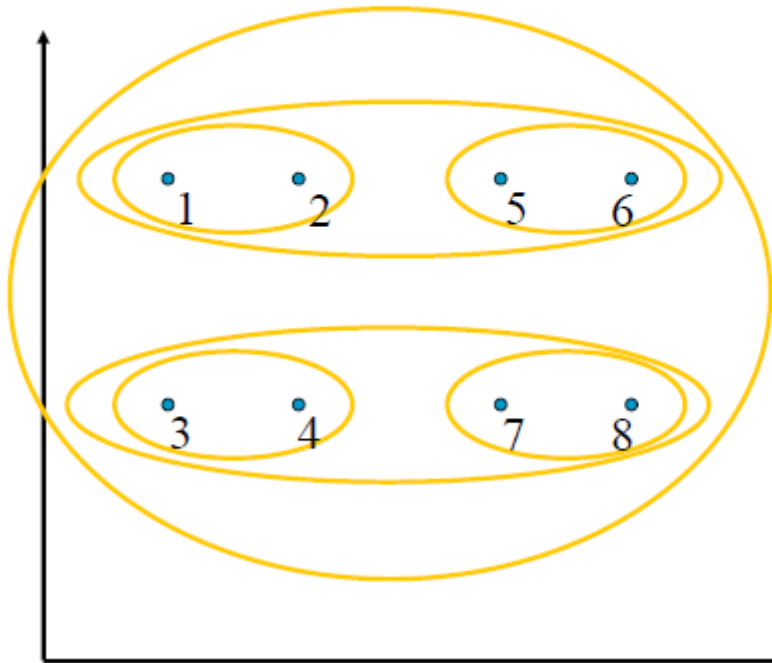
# Closeness

- How should we define “closest” for clusters with multiple elements?
- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
- Different choices create different clustering behaviors

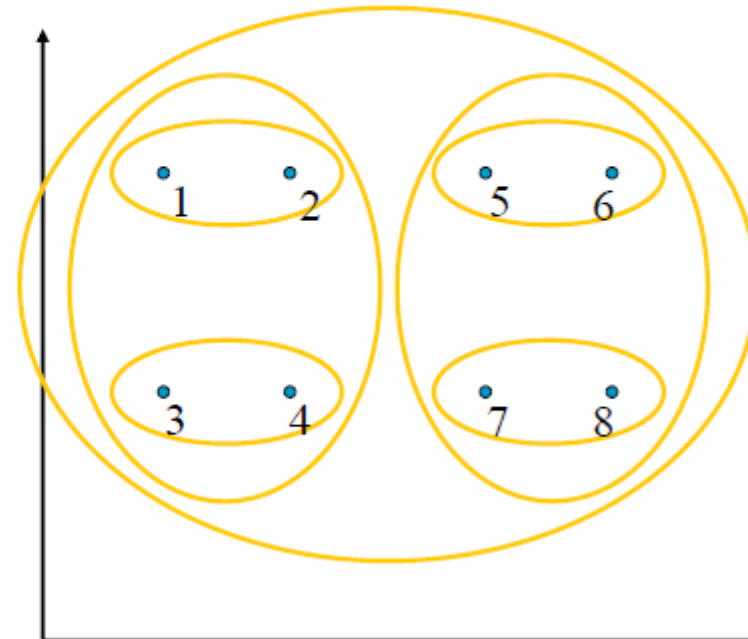


# Closeness example

Closest pair  
(single-link clustering)

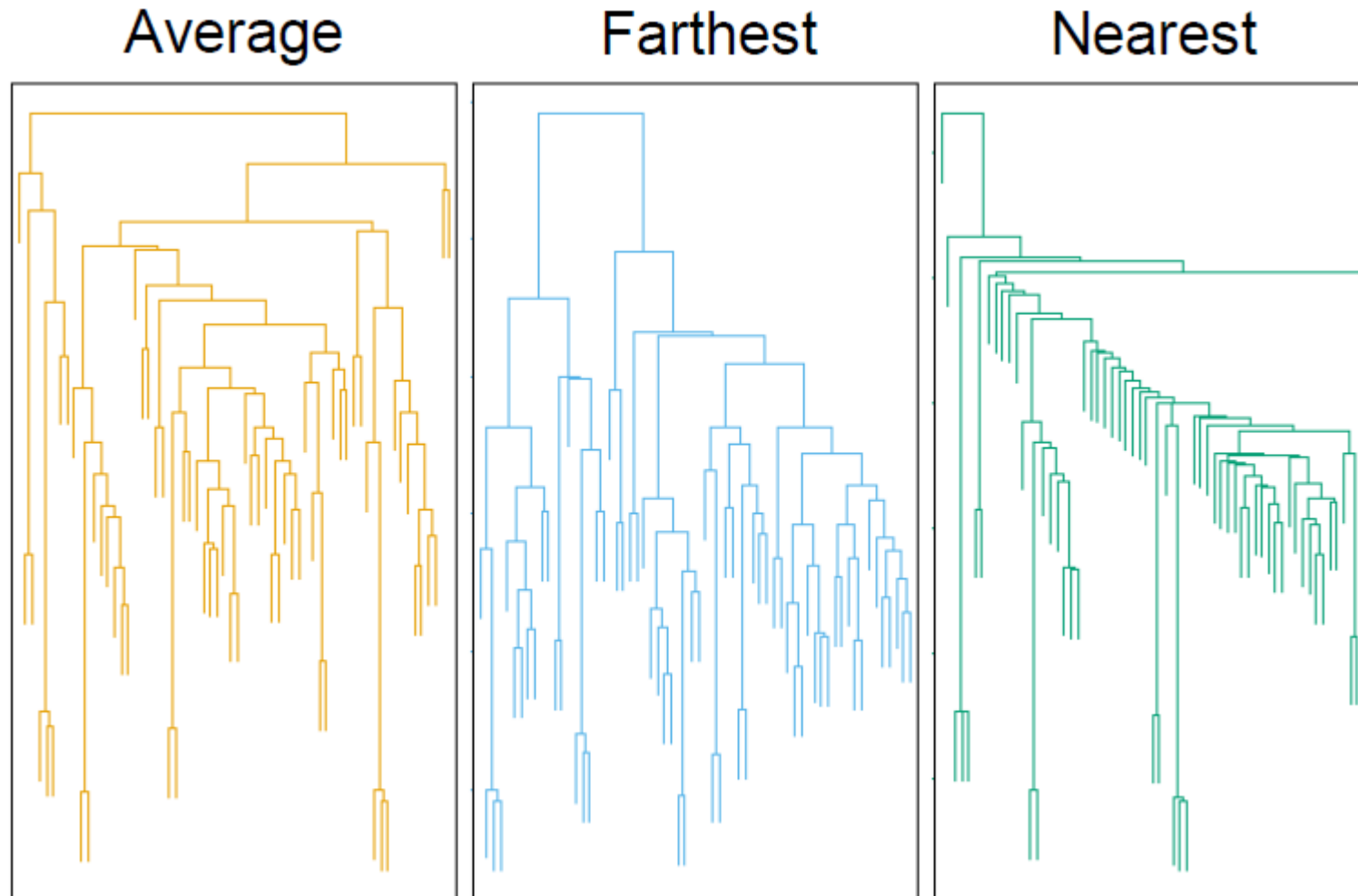


Farthest pair  
(complete-link clustering)





# Closeness example 2

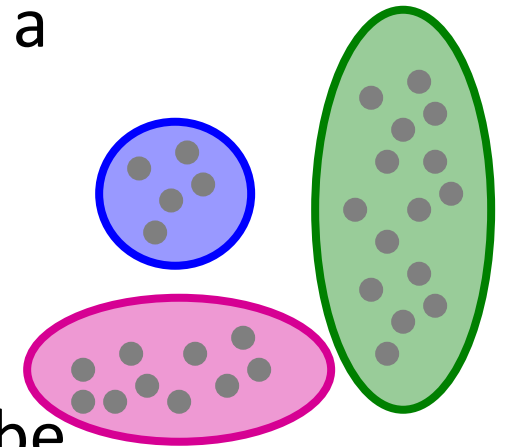
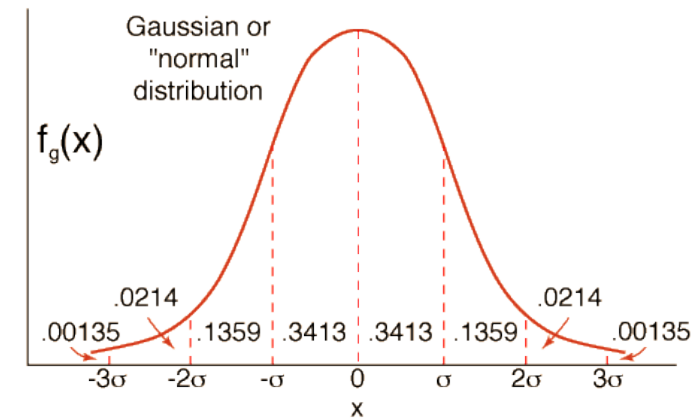


# The BFR Algorithm

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
- Goal is to find cluster centroids; point assignment can be done in a second pass through the data.



# BFR Overview

- **Efficient way to summarize clusters:** Want memory required  $O(\text{clusters})$  and not  $O(\text{data})$
- **IDEA: Rather than keeping points, BFR keeps summary statistics of groups of points**
  - 3 sets: Cluster summaries, Outliers, Points to be clustered
- **Overview of the algorithm:**
  - 1. Initialize  $K$  clusters/centroids
  - 2. Load in a bag points from disk
  - 3. Assign new points to one of the  $K$  original clusters, if they are within some distance threshold of the cluster
  - 4. Cluster the remaining points, and create new clusters
  - 5. Try to merge new clusters from step 4 with any of the existing clusters
  - 6. Repeat steps 2-5 until all points are examined

# BFR Algorithm

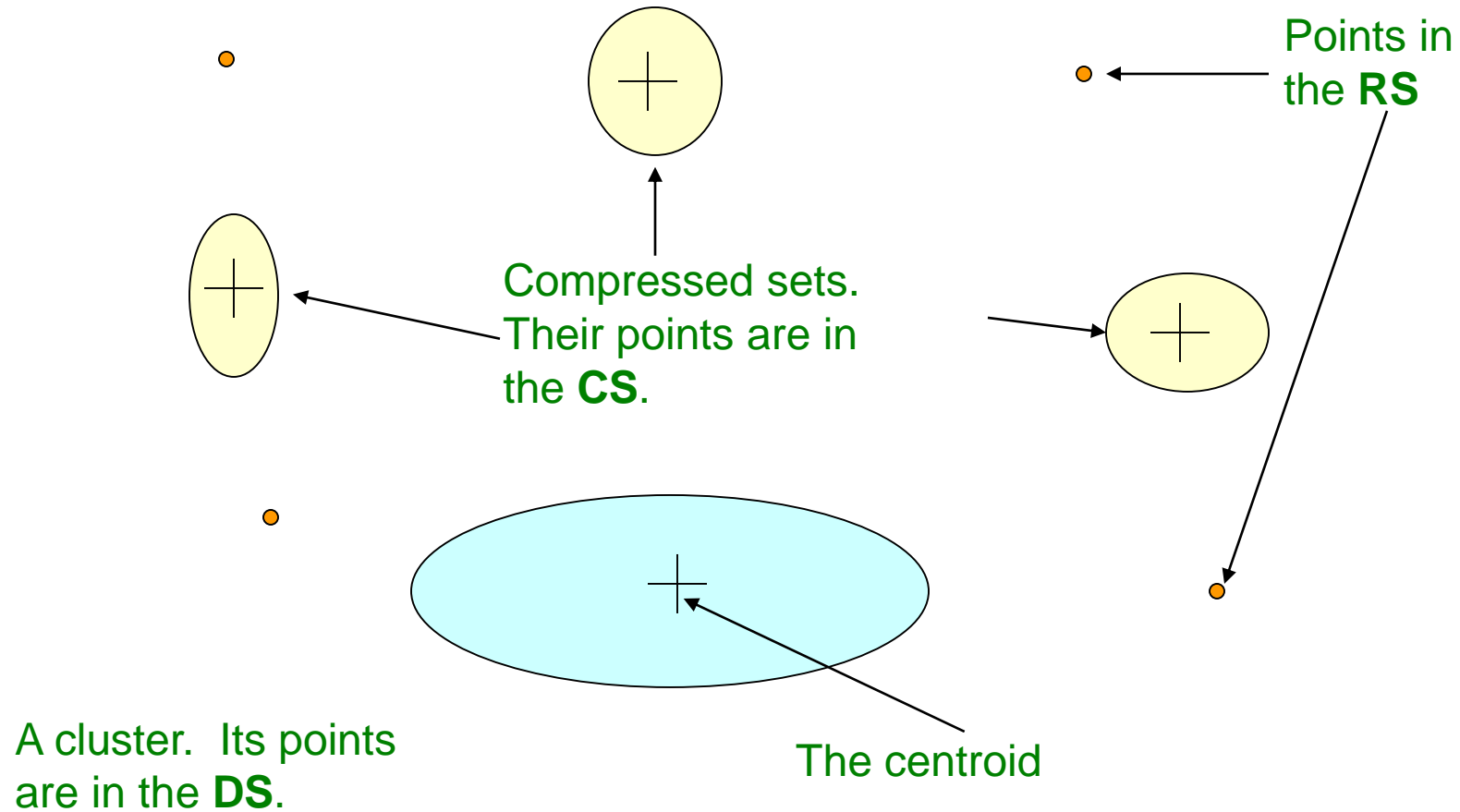
- Points are read from disk one main-memory-full at a time
- Most points from previous memory loads are summarized by simple statistics
- Step 1) From the initial load we select the initial  $k$  centroids by some sensible approach:
  - Take  $k$  random points
  - Take a small random sample and cluster 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

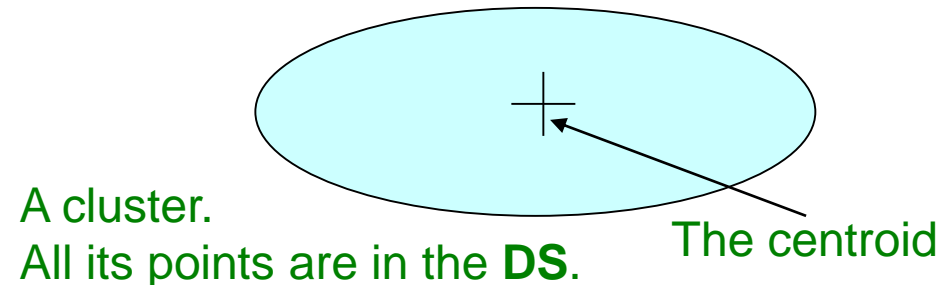


**Discard set (DS):** Close enough to a centroid to be summarized  
**Compression set (CS):** Summarized, but not assigned to a cluster  
**Retained set (RS):** Isolated points

# 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

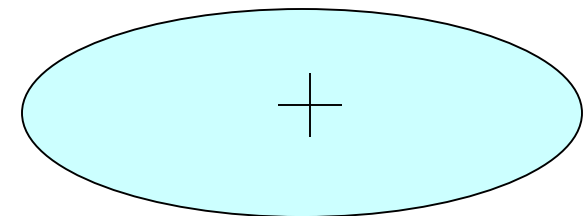




# Summarizing Points: Comments

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



# The “Memory-Load” of Points

## Steps 2-5) Processing “Memory-Load” of points:

- **Step 3)** 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
- **Step 4)** Use any in-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

## Steps 2-5) Processing “Memory-Load” of points:

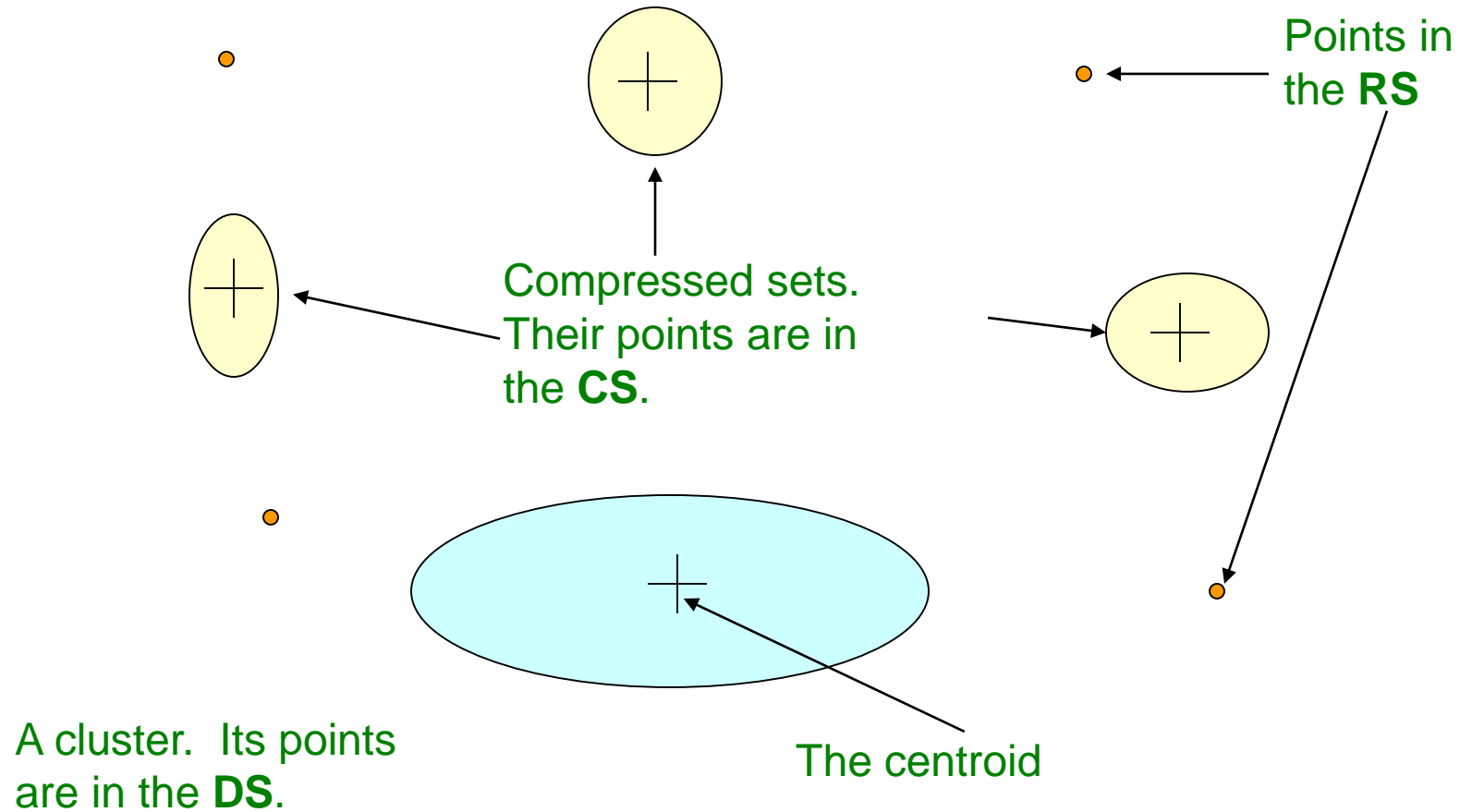
- **Step 5) DS set:** Adjust statistics of the clusters to account for the new points
  - Add  $N_s$ ,  $SUM_s$ ,  $SUMSQ_s$
  - Consider merging compressed sets in the **CS**
- **If this is the last round**, merge all compressed sets in the **CS** and all **RS** points into their nearest cluster

**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



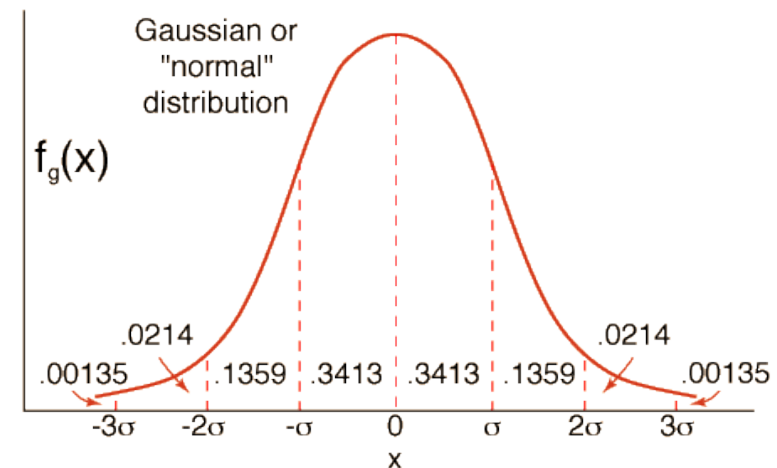
**Discard set (DS):** Close enough to a centroid to be summarized  
**Compression set (CS):** Summarized, but not assigned to a cluster  
**Retained set (RS):** Isolated points

# 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**

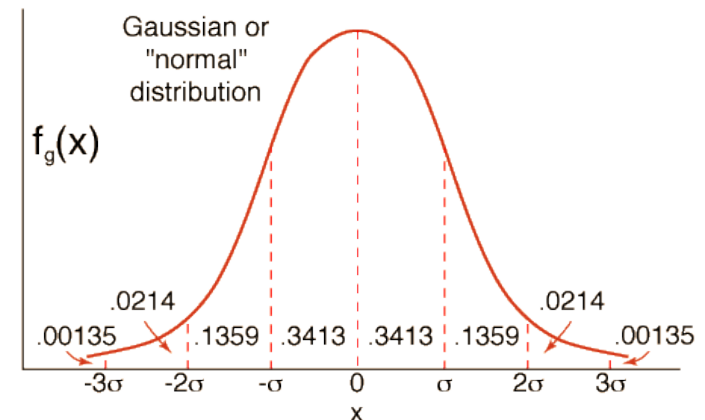
- For point  $(x_1, \dots, x_d)$  and centroid  $(c_1, \dots, c_d)$ 
  1. Normalize in each dimension:  $y_i = (x_i - c_i) / \sigma_i$
  2. Take sum of the squares of the  $y_i$
  3. Take the square root

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

$\sigma_i$  ... standard deviation of points in the cluster in the  $i^{\text{th}}$  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

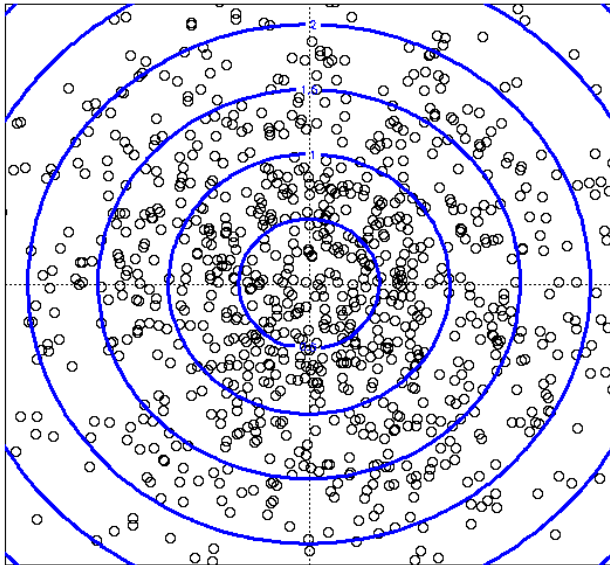




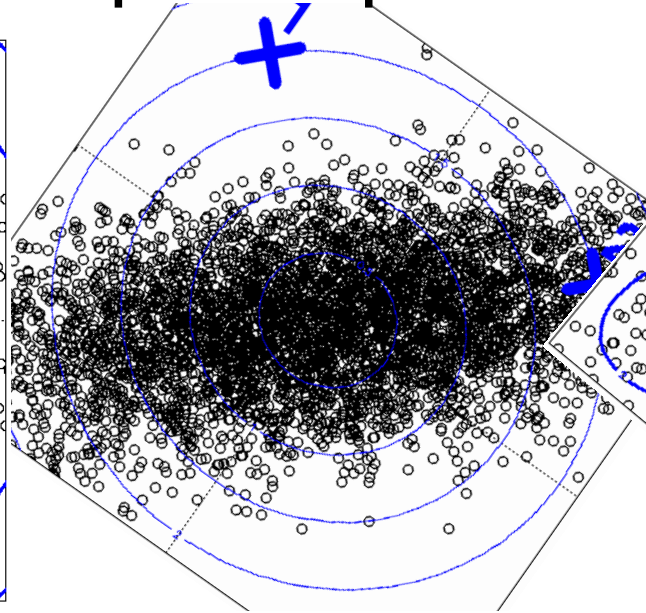
# Picture: Equal M.D. Regions

- Euclidean vs. Mahalanobis distance

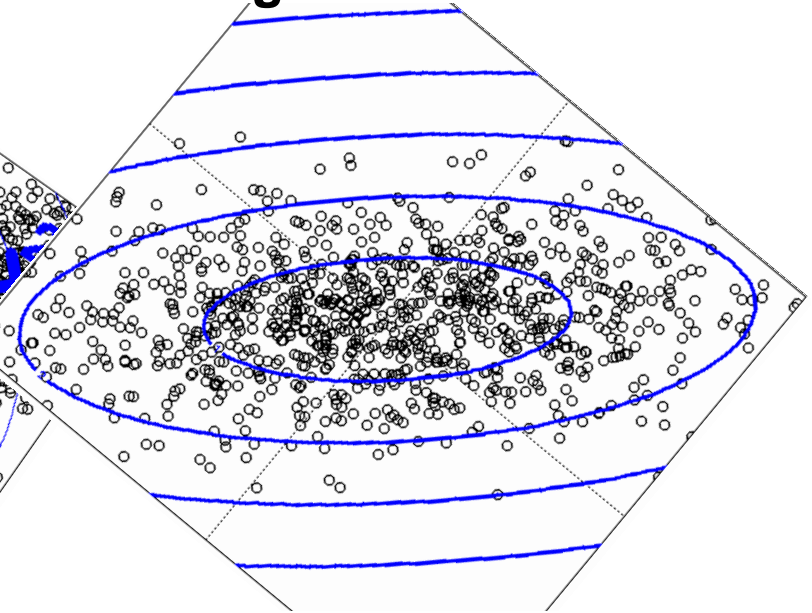
Contours of equidistant points from the origin



Uniformly distributed points,  
Euclidean distance



Normally distributed points,  
Euclidean distance

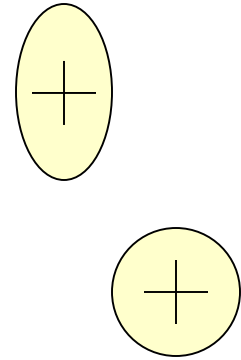


Normally distributed points,  
Mahalanobis distance

# 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



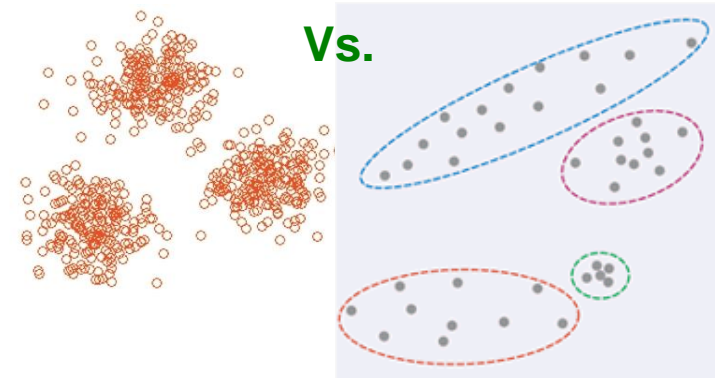
# The CURE Algorithm

Extension of k-means to clusters of arbitrary shapes

# The CURE Algorithm

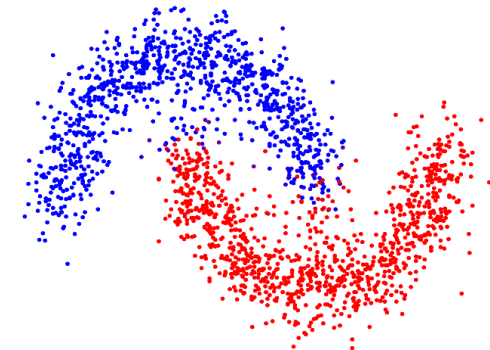
- **Problem with BFR/k-means:**

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

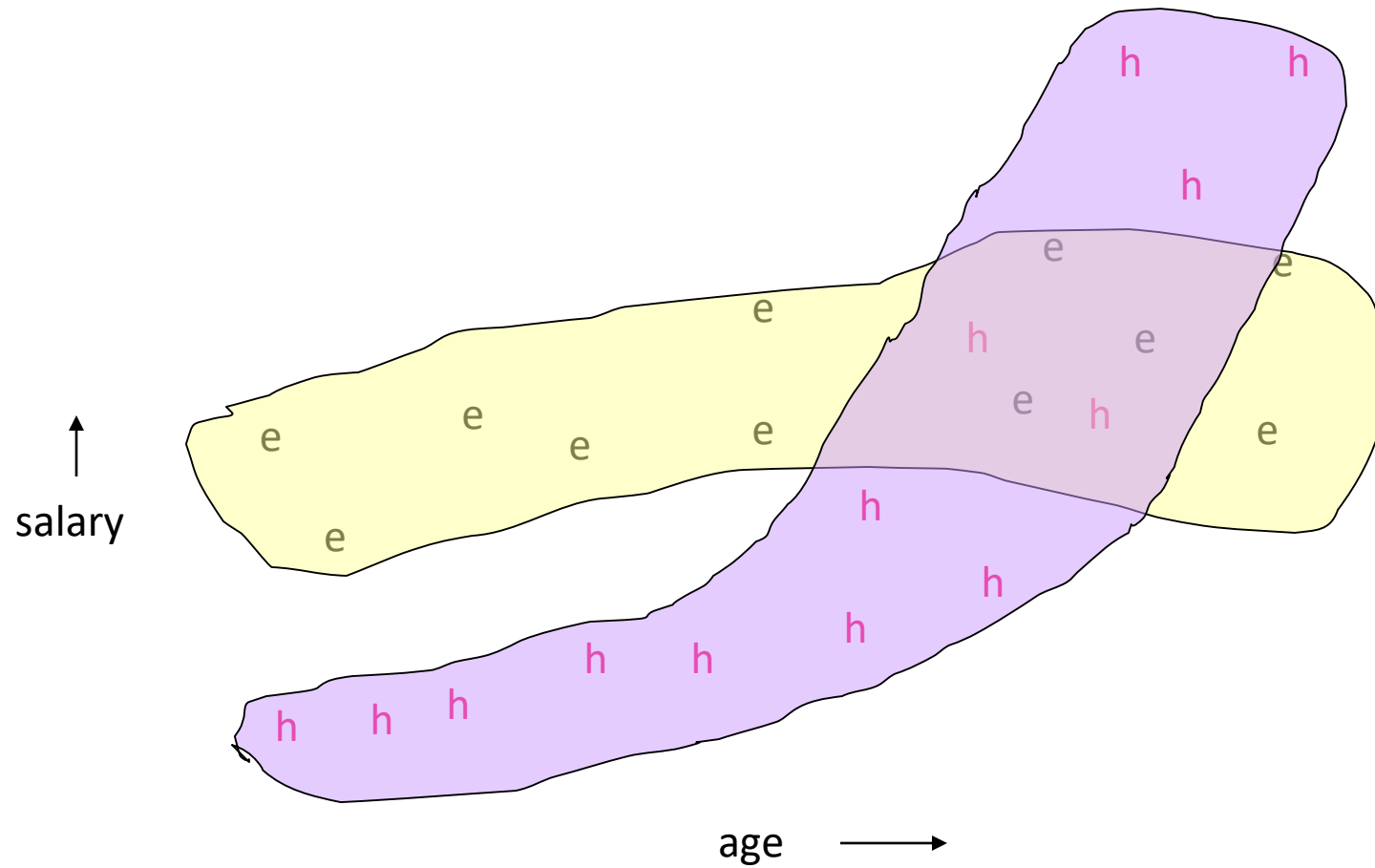


- **CURE (Clustering Using REpresentatives):**

- Assumes a Euclidean distance
- Allows clusters to assume any shape
- **Uses a collection of representative points to represent clusters**



# Example: Stanford Salaries

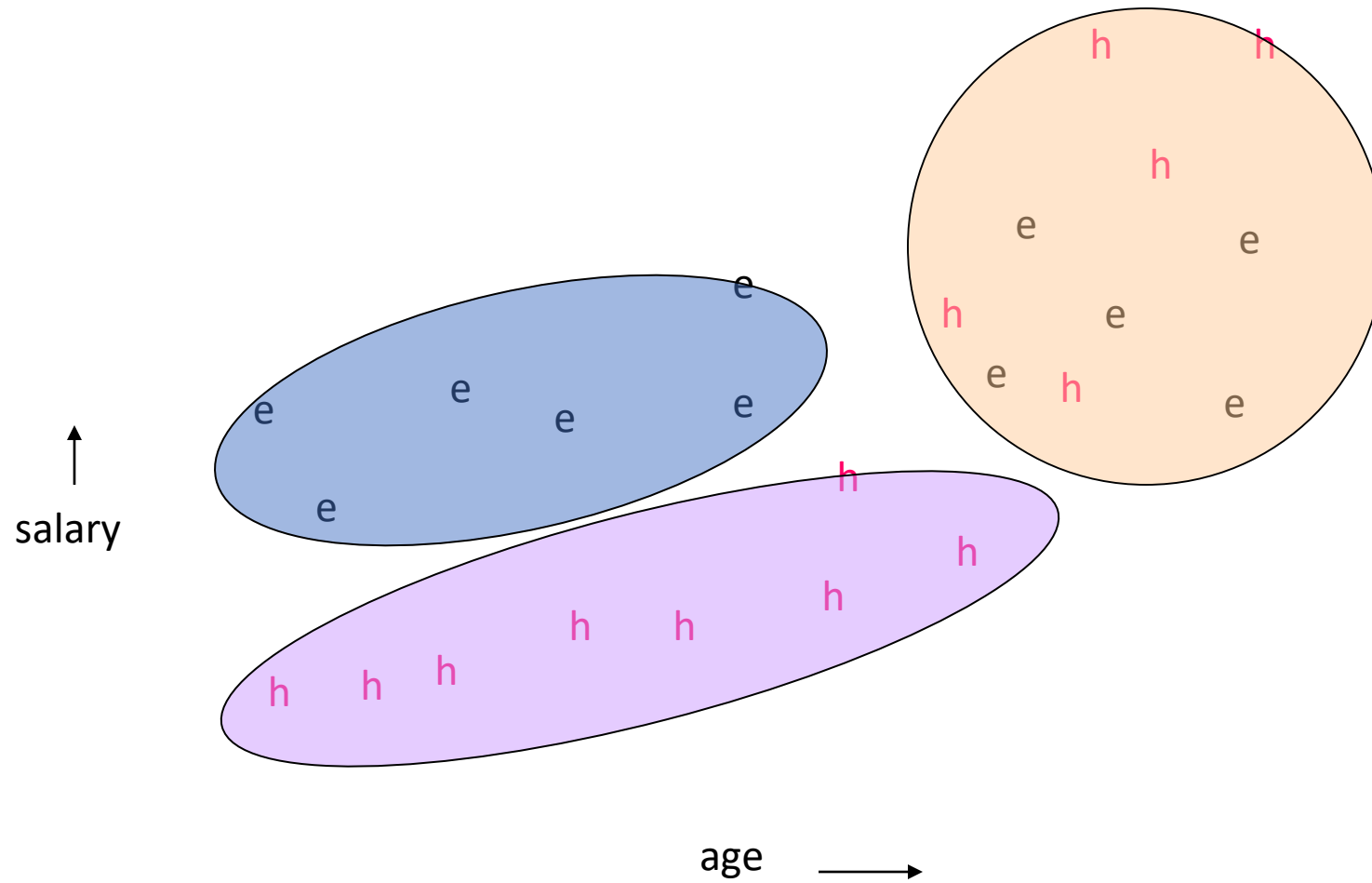


# Starting CURE

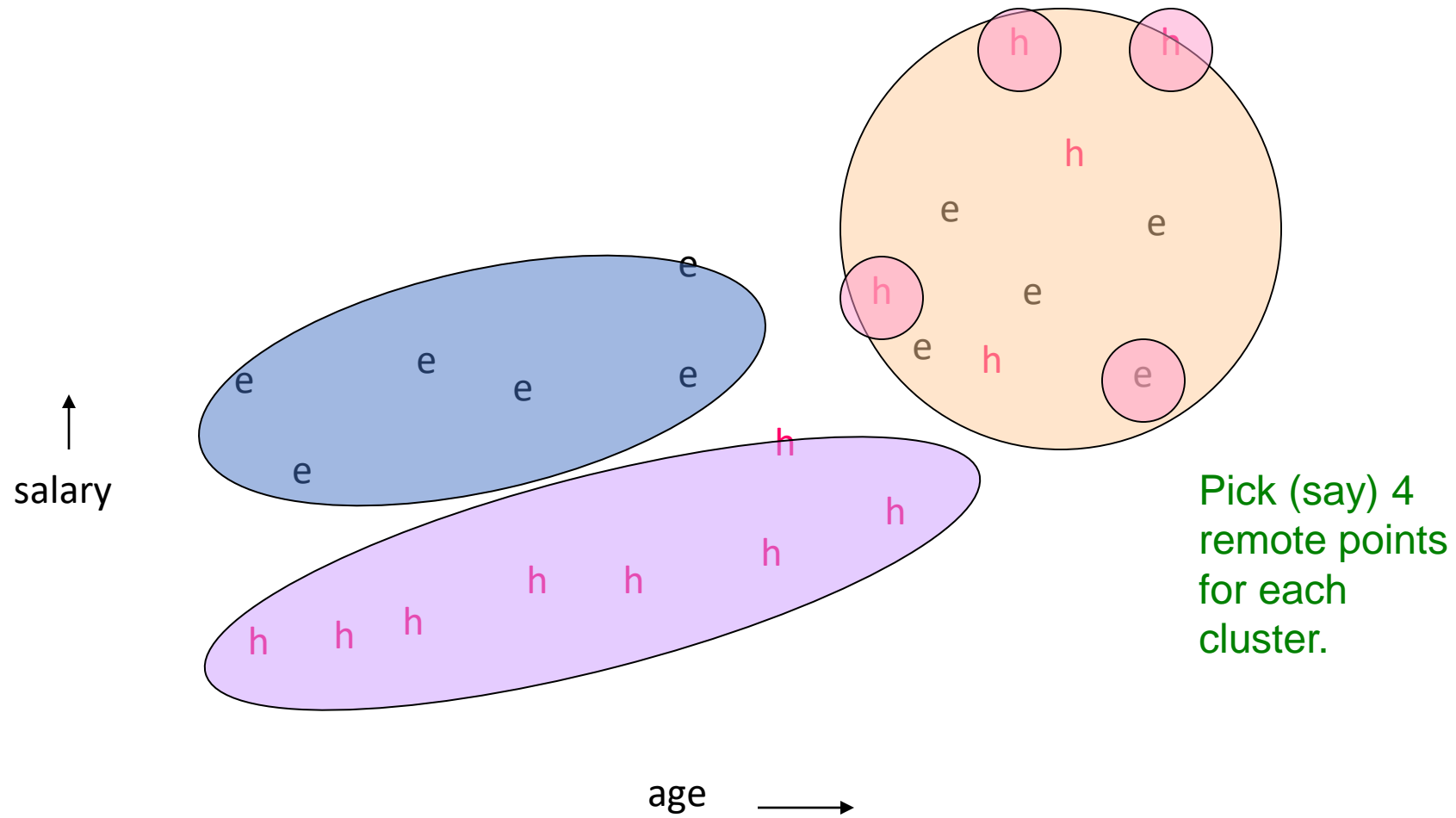
## 2 Pass algorithm. Pass 1:

- **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

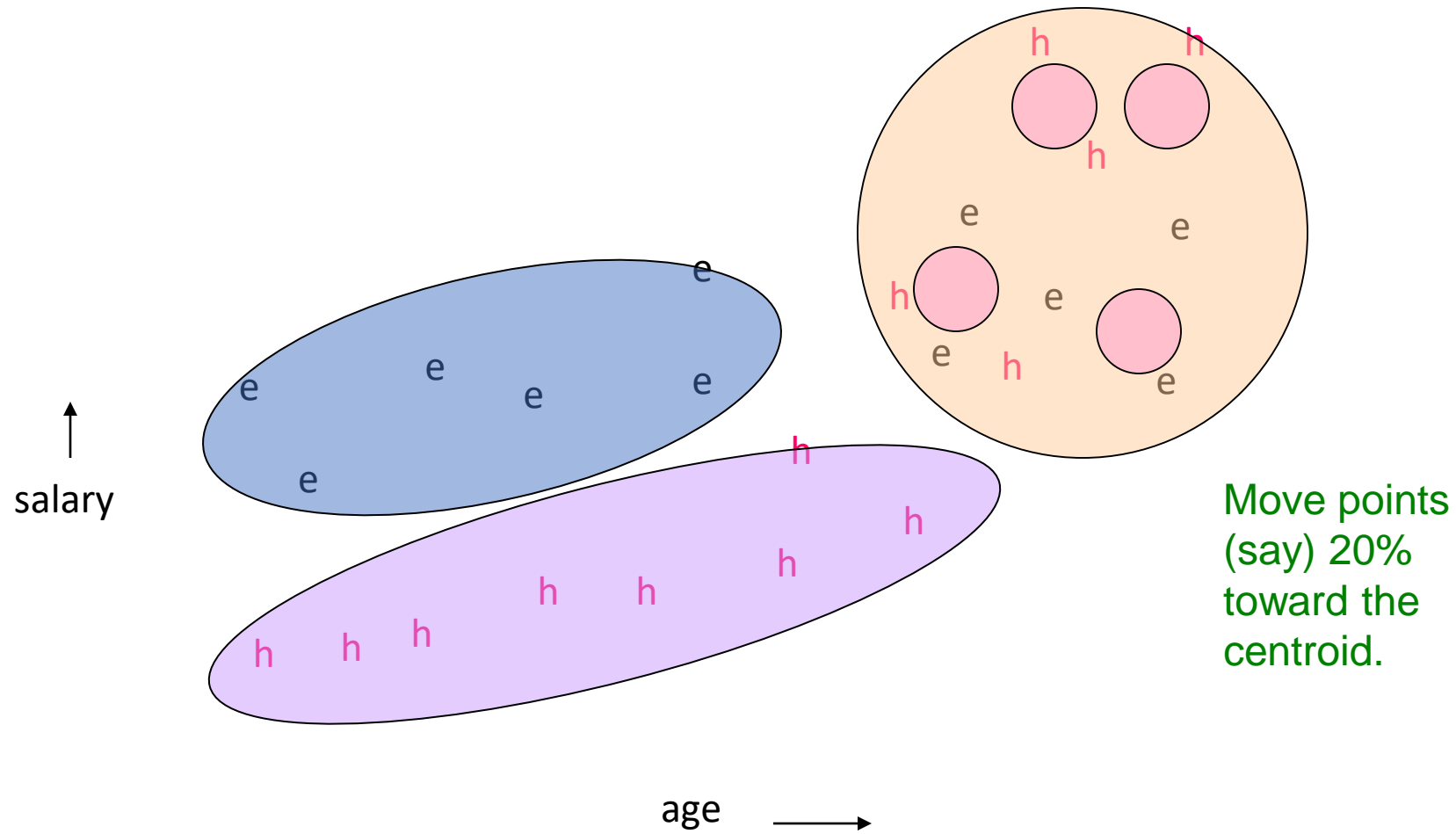


# Example: Pick Dispersed Points





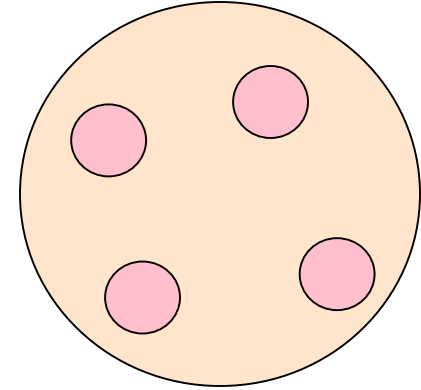
# Example: Pick Dispersed Points



# 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



$p$

# Why the 20% Move Inward?

## Intuition:

- A large, dispersed cluster will have large moves from its boundary
- A small, dense cluster will have little move.
- Favors a small, dense cluster that is near a larger dispersed cluster

