Clustering

(part 2)

INF 553:
Foundations and Applications of Data Mining

Clustering

- **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 (for Euclidean spaces)
    - Clustroid (for non-Euclidean spaces)
- **Point assignment:**
  - K-means
  - BFR: extend k-means to handle large data set
  - CURE.

k-means clustering

k–means Algorithm(s)

- **Point Assignment Algorithm**
- 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

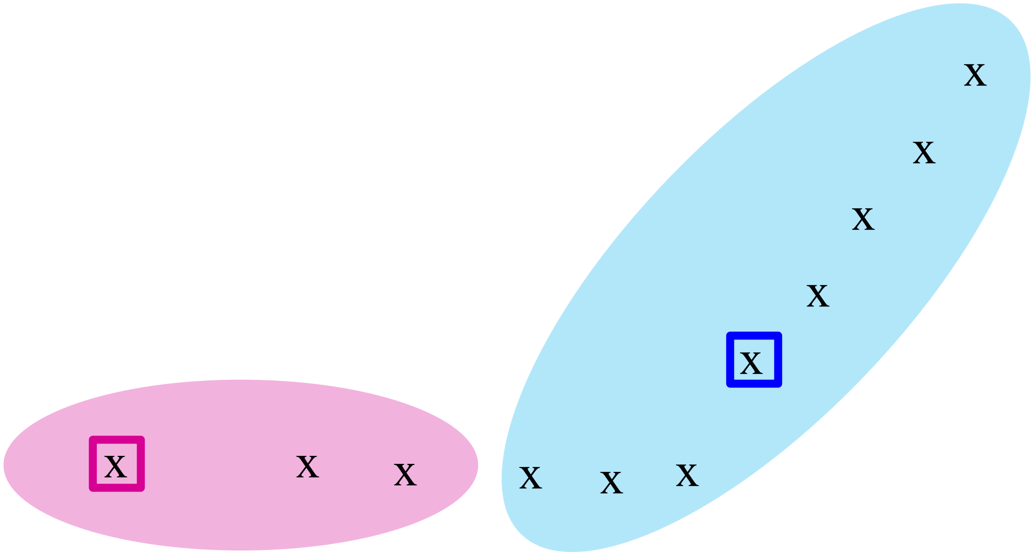
- **1)** For each point, **place it in the cluster whose current centroid it is nearest**

- **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

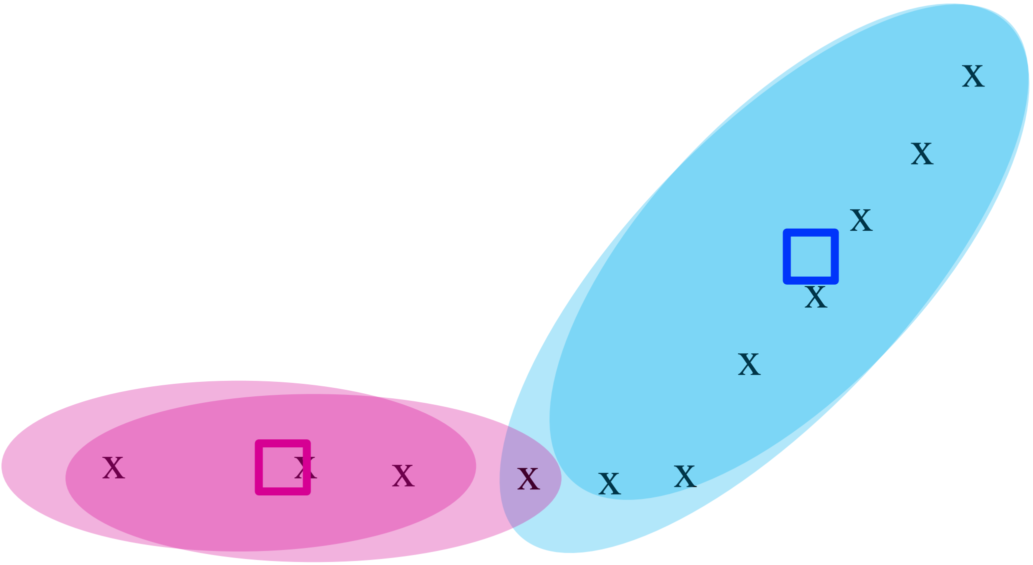


x … data point

□ … centroid

Clusters after round 1

Example: Assigning Clusters

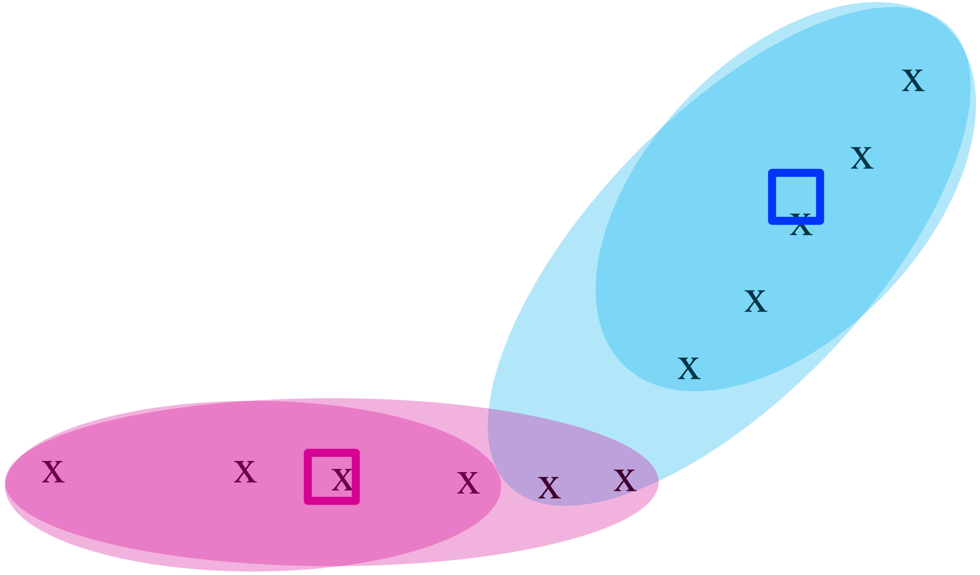


x … data point

□ … centroid

Clusters after round 2

Example: Assigning Clusters



x … data point

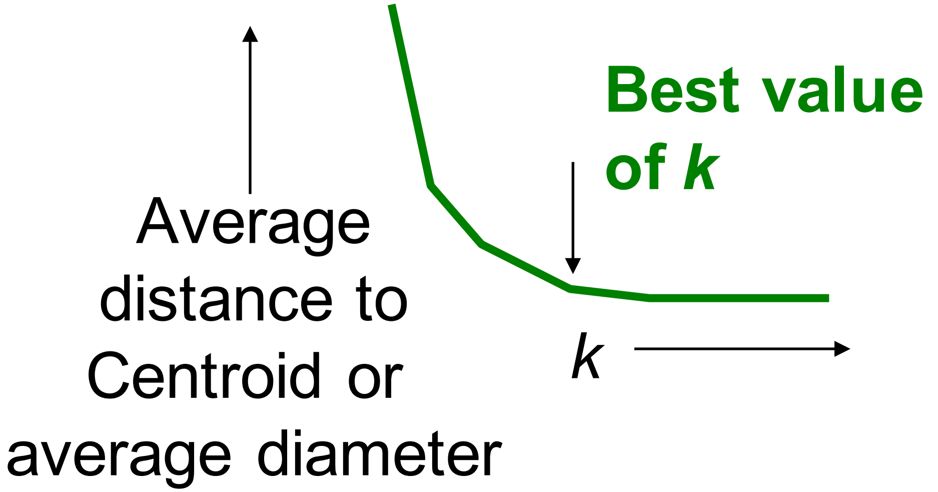
□ … centroid

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



Average distance to Centroid or average diameter

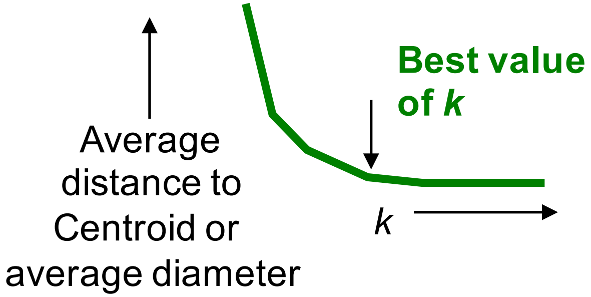
k

Best value of *k*

Intuition on Picking k

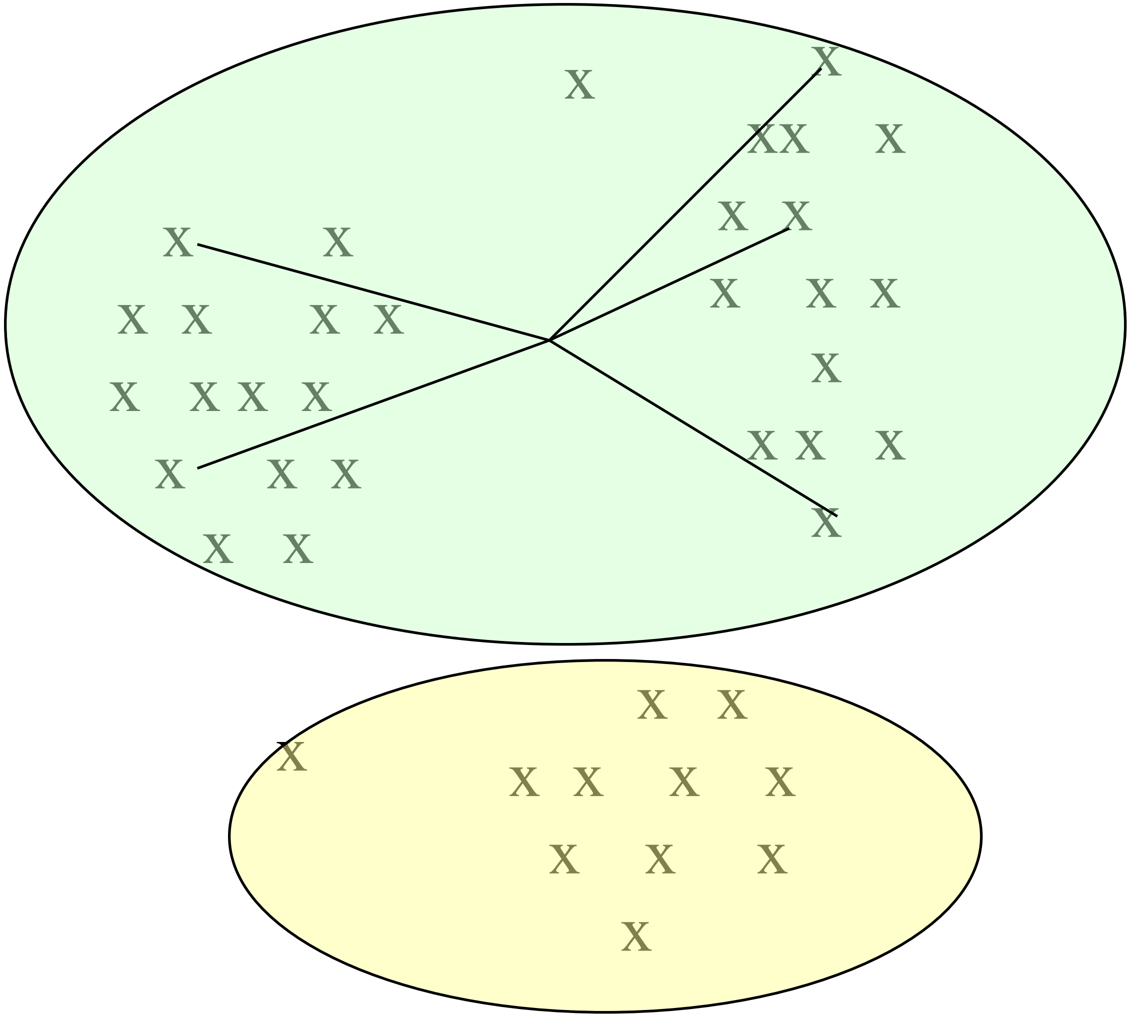
- Consider a data set with three natural clusters
- If we force a merge of two of these natural clusters, the diameter jumps quickly
  - **Diameter of cluster = maximum** distance between any **two points** of the cluster
  - **Radius of cluster = maximum** distance between all **points and the centroid**
  - Measure of **appropriateness for a cluster** (e.g., average radius or diameter) **grows slowly,** as long as the number of clusters we assume is at or above the true number of clusters.

Example: Picking *k*

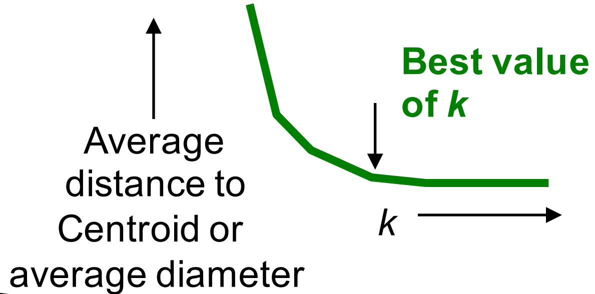


Too few;

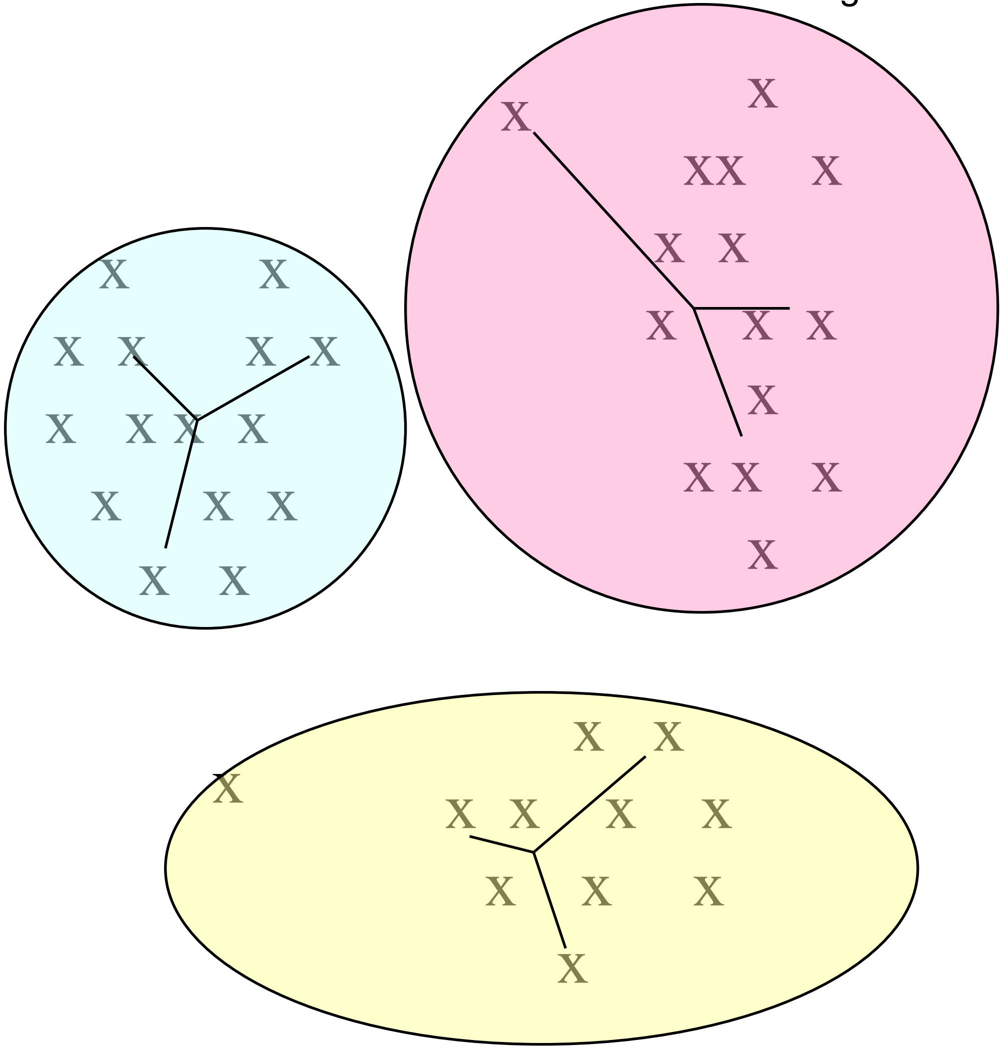
many long distances to centroid.



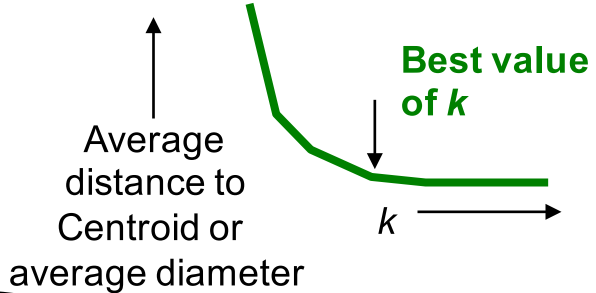
Example: Picking *k*



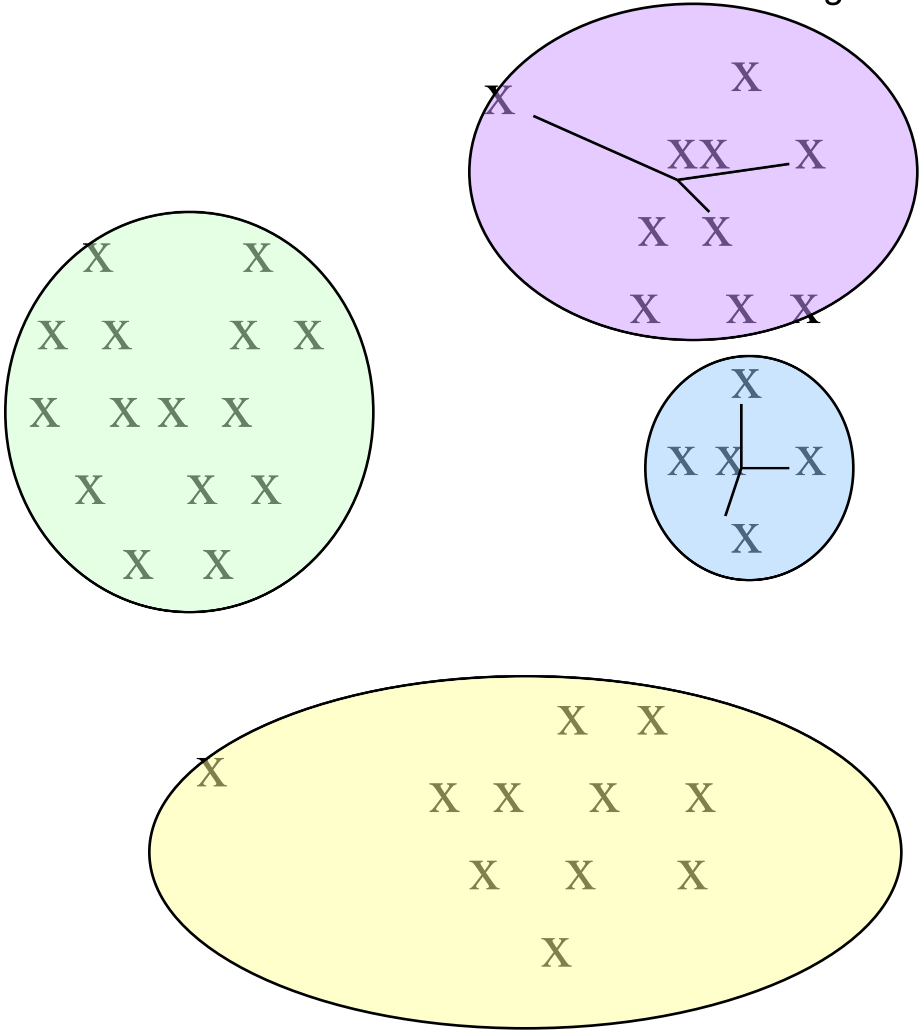
Just right;
distances
rather short.



Example: Picking *k*



Too many;
little improvement in average distance.



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.

Complexity

- In each round, we have to examine each **input point exactly once** to find **closest centroid**
- Each round is ***O(kN)*** for ***N*** **points,** ***k*** **clusters**
- But the number of rounds to convergence **can be very large!**
- Can we cluster in a single pass over the data?

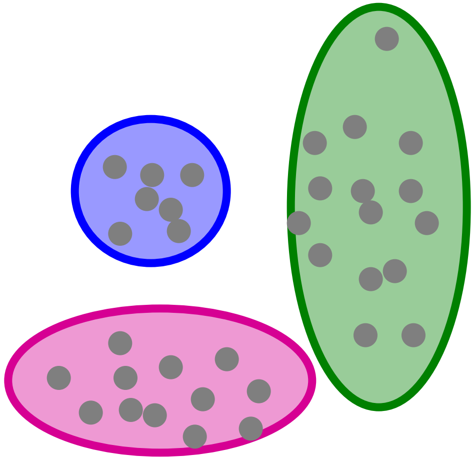
The BFR Algorithm

Extension of *k*-means to large data

BFR Algorithm

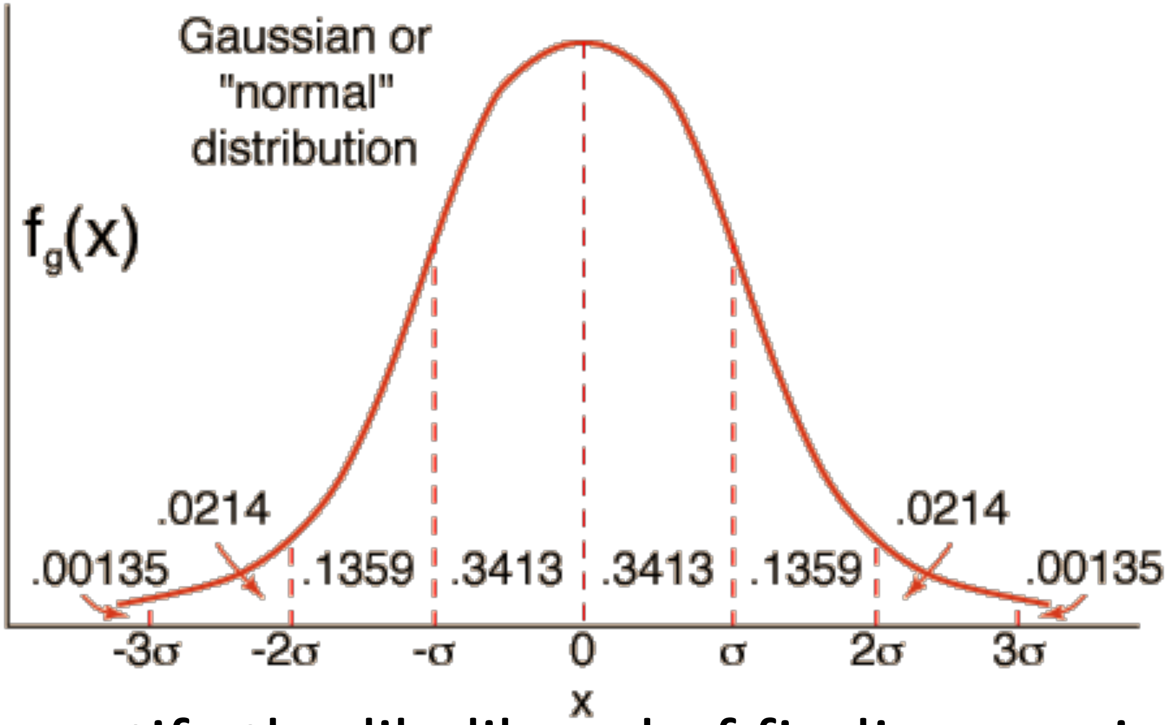
- **BFR** [Bradley-Fayyad-Reina]
- **Extension of *k*-means to large data**
- is a variant of *k*-means designed to handle **very large** (disk-resident) data sets

- **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)).

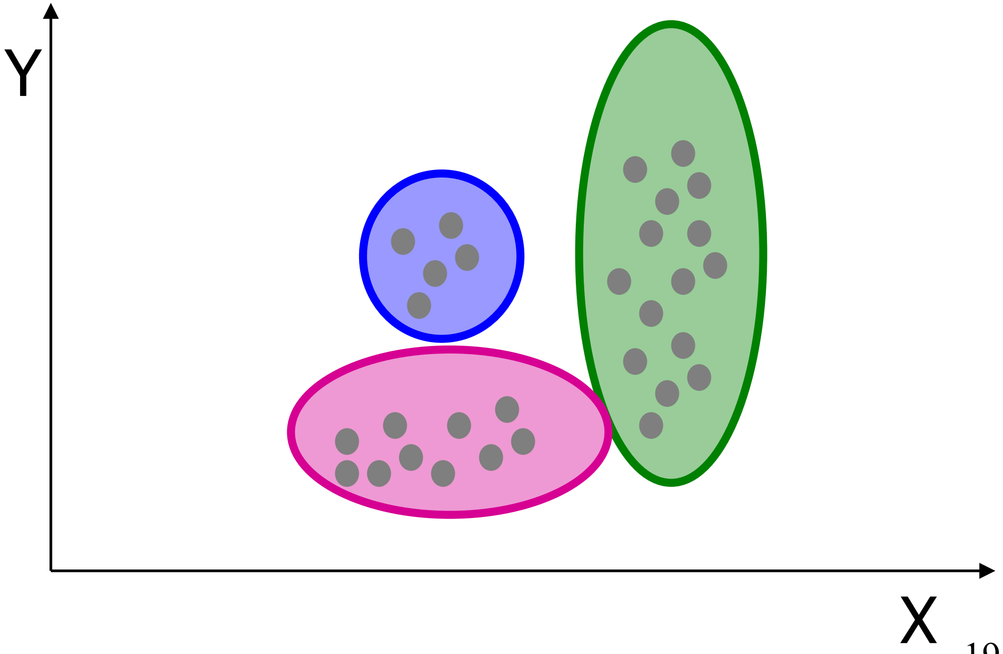
Normal Distribution



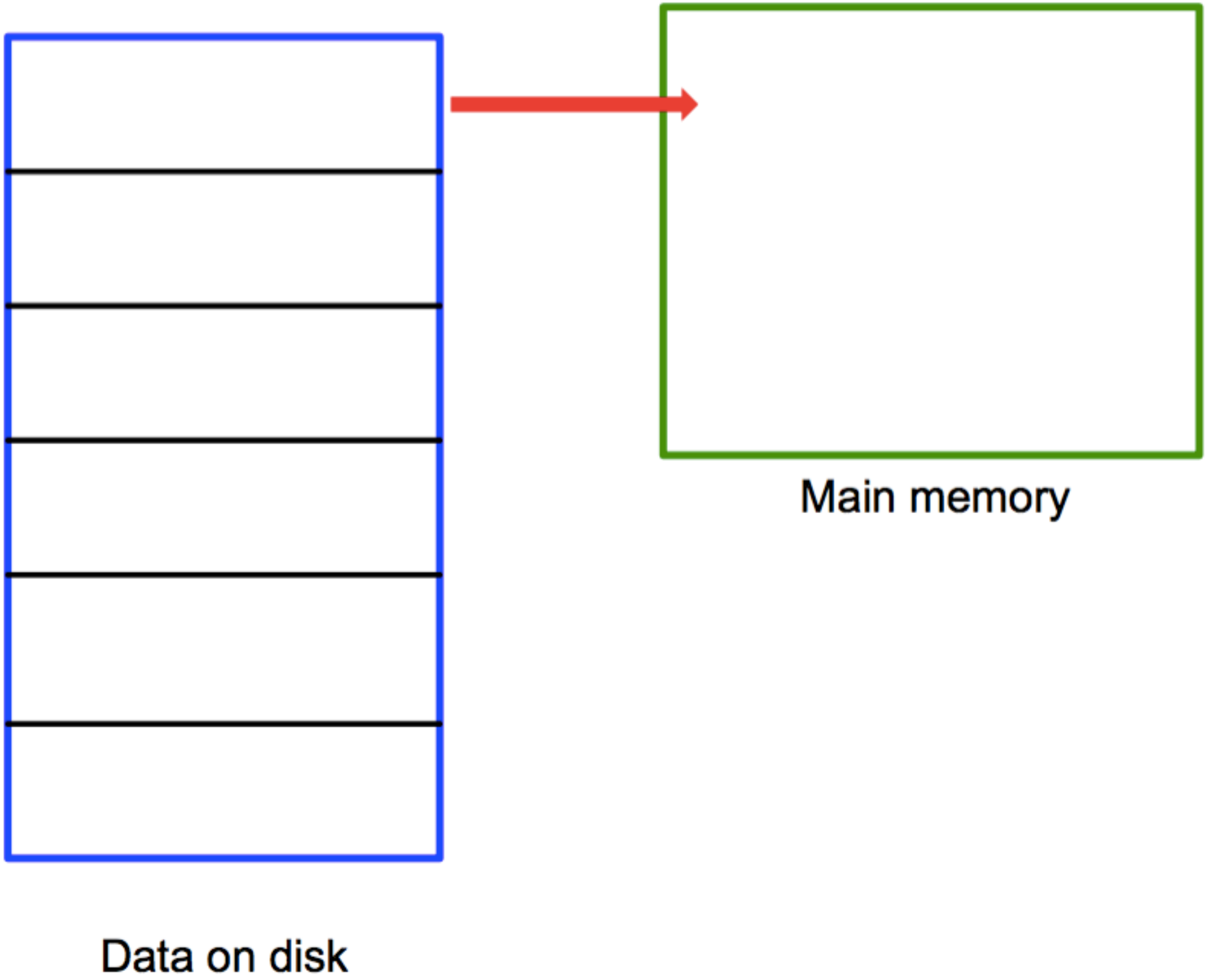
- Can quantify the likelihood of finding a point in the cluster at a given distance from the centroid along each dimension
- Standard deviation in different dimensions may vary
  - **68–95–99.7 rule:** 68.27%, 95.45% and 99.73% of the values lie within one, two and three standard deviations of the mean, respectively.

BFR Clusters

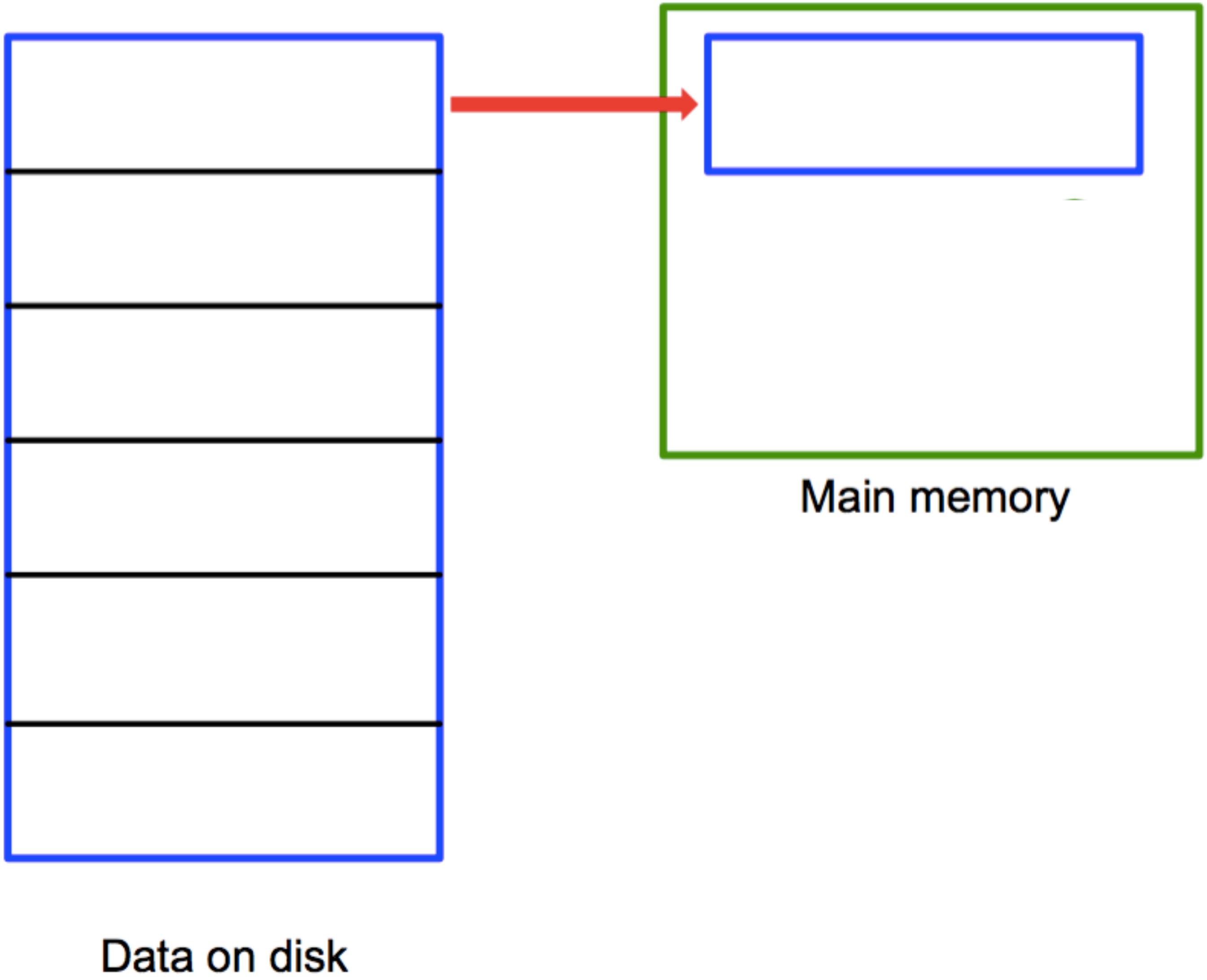
- Normal distribution assumption implies that clusters “look like” axis-aligned ellipses



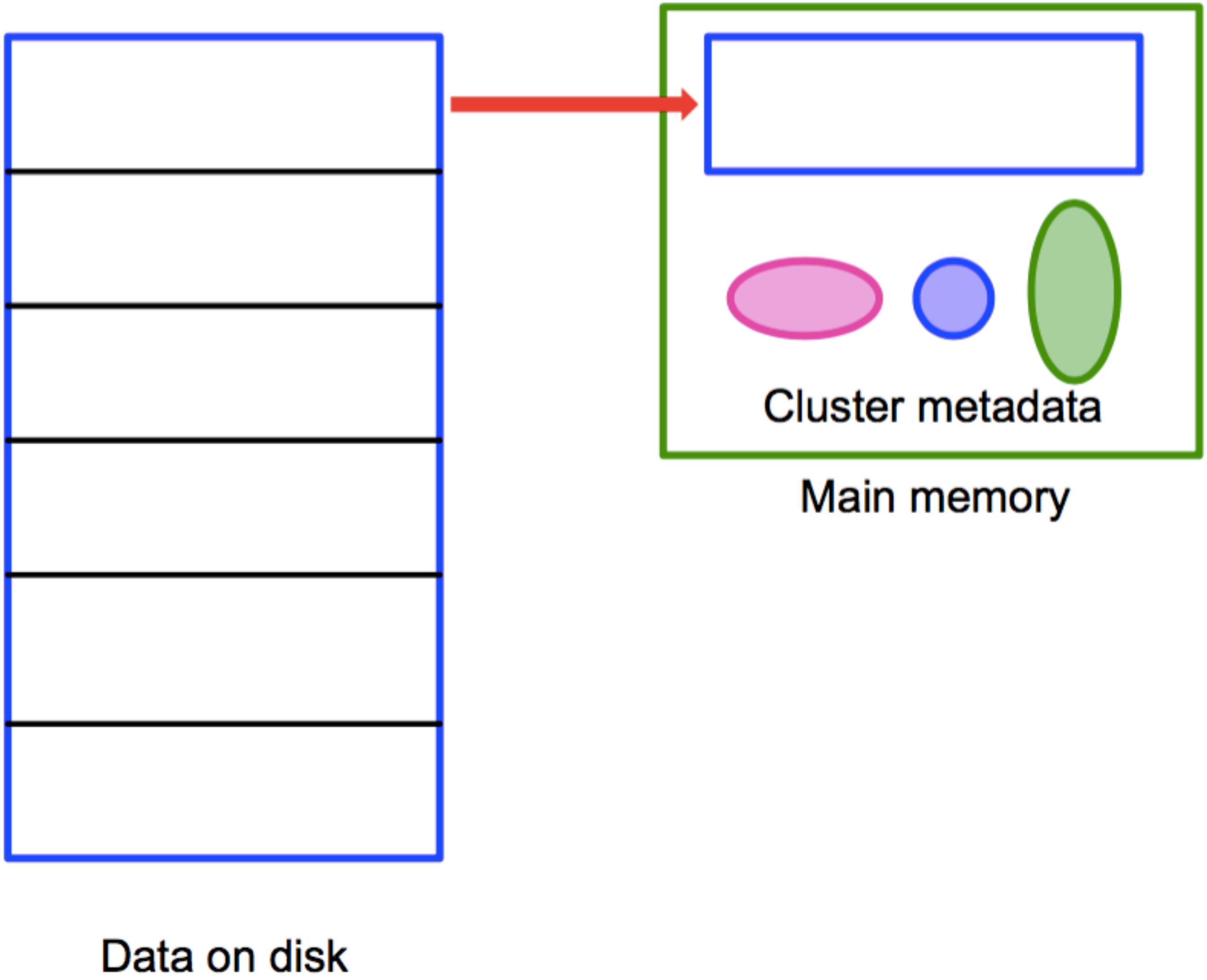
BFR Algorithm: Overview



BFR Algorithm: Overview



BFR Algorithm: Overview



BFR Algorithm

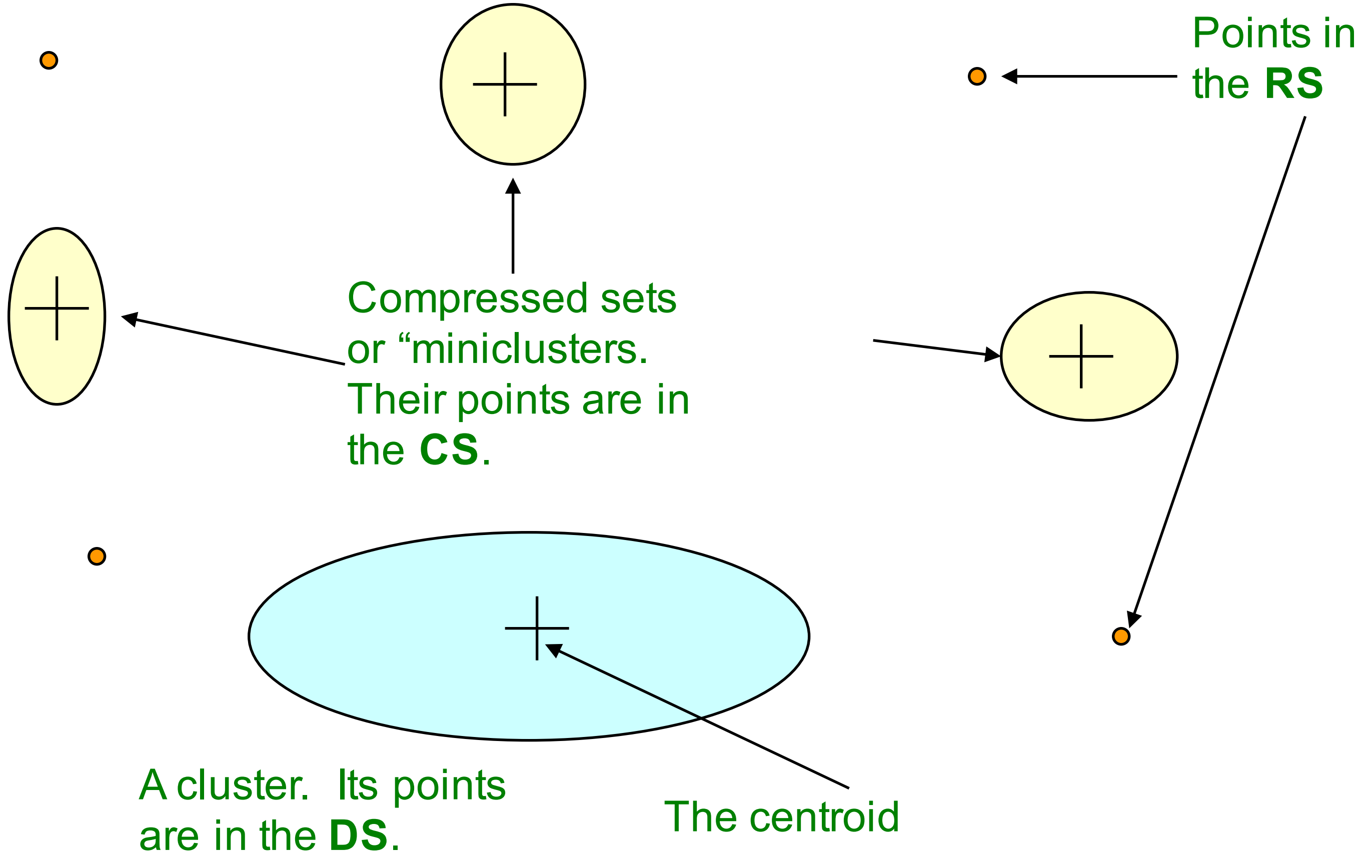
- Points are read from disk one main-memory-full at a time
- **Most points from previous memory loads are summarized by simple statistics**
- To begin, from the initial load we select the initial ***k*** centroids by some sensible approach:
  - Take ***k*** random points
  - Take a small random sample 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
  - Called a "minicluster"
- **Retained set (RS):**
  - Isolated points waiting to be assigned to a compression set.

BFR: “Galaxies” Picture



Points in the **RS**

Compressed sets or “miniclusters. Their points are in the **CS**.

A cluster. Its points are in the **DS**.

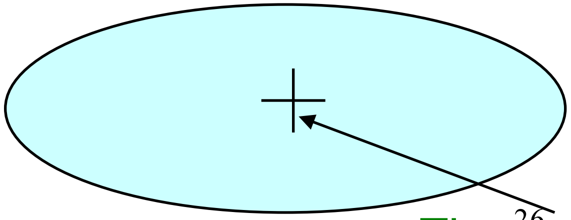
The centroid

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*** of length **d dimensions**, whose i^{th} component is **the sum of the coordinates of the points in the i^{th} dimension**
- The vector ***SUMSQ*** of length **d dimensions**, whose i^{th} component is the sum of squares of coordinates in i^{th} dimension.

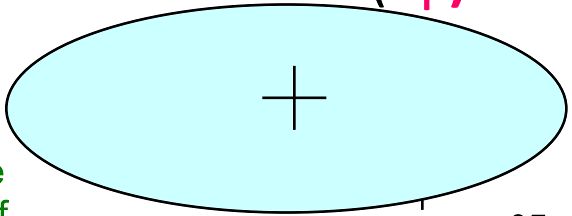


A cluster.
All its points are in the **DS**.

The centroid

Summarizing Points: Comments

- **2*d* + 1** values represent any size cluster
  - Two vectors of length ***d*** = number of dimensions
  - One integer ***N***
- 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 (σ_i)
- **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!

BFR Algorithm

Processing a "Memory-Load" of points:

- **0)** From the initial load we selected the initial ***k*** centroids
- **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**
  - **Adjust statistics of the clusters** to account for the new points
  - Add to ***N***s, ***SUM***s, ***SUMSQ***s
  - More on "sufficiently close" later…

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 Algorithm (2)

- **2)** Use any main-memory clustering algorithm to **cluster the remaining points and the old Retained Set (RS)**
  - **"Miniclusters" of more than one point are summarized, added to the CS**
    - Summarized in same way (SUM, SUMSQ, N)
  - **Singleton clusters (outlying points) become the RS**

- **3)** Consider **merging miniclusters in the Compression Set (CS)**
  - **Miniclusters from the last step**
  - Can't merge with a cluster, but **may merge with one another**
  - **Use earlier criteria** (merge clusters with smallest distance between centroids, or whose resulting cluster has smallest diameter or radius, etc.)
  - More on this later…

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 Algorithm (3)

- **4) Points that are assigned to a cluster or minicluster are written** out with their assignment **to secondary memory**
  - **(discard DS and CS from memory)**

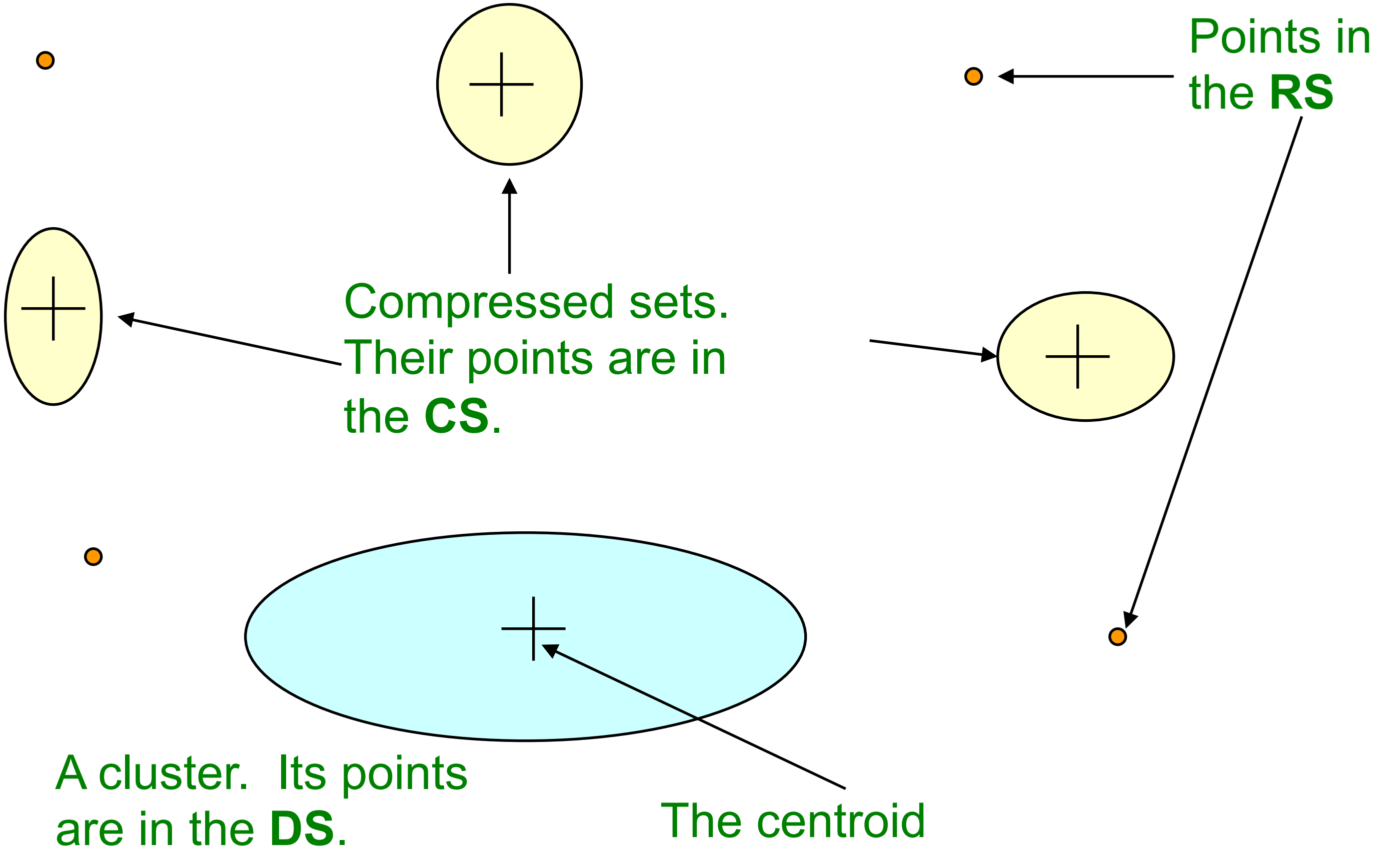
- **(Repeat steps 1 to 4 for all memory-sized chunks of input data)**

- **5)** If this is **the last chunk of input data**, **decide what to do with remaining CS and RS**
  - Can treat them as outliers and not cluster them at all Or
  - **Or Merge sets in CS and points in RS with their nearest cluster:**

  1. **Assign each point in RS to cluster with nearest centroid**
  2. **Combine each minicluster in the CS with the cluster whose centroid is closest to the centroid of the minicluster.**

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



Points in the **RS**

Compressed sets. Their points are in the **CS**.

A cluster. Its points are in the **DS**.

The centroid

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…

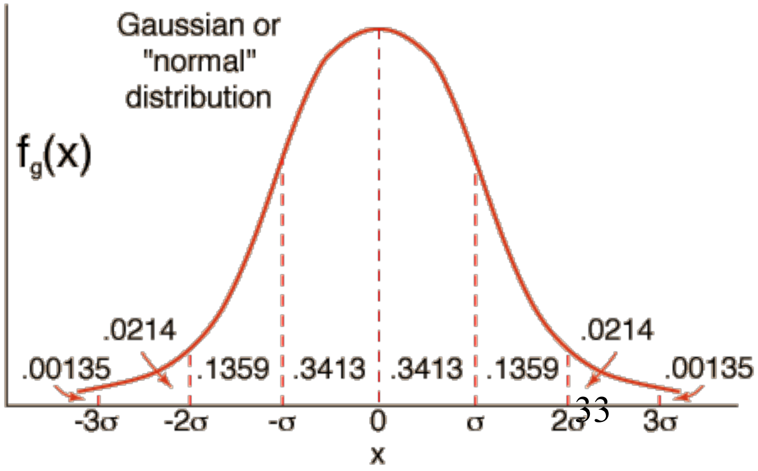
- Q1) How do we decide if a point is **“close enough”** to a cluster that we will **add the point to that cluster?**
  - New distance measure

- 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)**

- The **Mahalanobis distance** is less than a threshold
- **High likelihood of the point belonging to cluster with the currently nearest centroid.**



Mahalanobis Distance

- Normalized Euclidean distance from centroid
- Cluster *C* has centroid **(c_1, …, c_d)** and standard deviations **(σ_1, …, σ_d)**
- Point P=(p_1, …, p_d)

1. Normalize in each dimension: **$(p_i - c_i) / \sigma_i$**

   where σ_i = standard deviation of points in the cluster in the *i*th dimension

2. Take sum of squares
3. Take square root.

$$\sqrt{\sum_{i=1}^{d} (\frac{p_i - c_i}{\sigma_i})^2}$$

Example 7.9

Example 7.9
Suppose a cluster consists of the points (5,1), (6,-2), (7,0)

- Average in **each dimension** (**the centroid**) can be calculated as $\mathbf{SUM}_i \,/\, N$

- Variance of a cluster's discard set in dimension *i* is: $(\mathbf{SUMSQ}_i \,/\, N) - (\mathbf{SUM}_i \,/\, N)^2$

- Standard variation?

Example 7.9 Standard variation?

Suppose a cluster consists of the points (5,1), (6,-2), (7,0)

Then N=3

SUM= [18,-1]

SUMSQ= [110,5]

- Average in **each dimension** (**the centroid**) can be calculated as **$\mathbf{SUM}_i$ / *N***
- Variance of a cluster's discard set in dimension *i* is: **($\mathbf{SUMSQ}_i$ / *N*) – ($\mathbf{SUM}_i$ / *N*)^2**

The centroid is SUM/N= [6, -1/3]

In the first dimension, the variance is

$$110/3 - (18/3)^2 = 0.667$$

The standard variation = $\sqrt{0.667} = 0.816$

In second dimension, the variance is

$$5/3 - (-1/3)^2 = 1.56$$

So the standard variation is 1.25.

Mahalanobis Distance

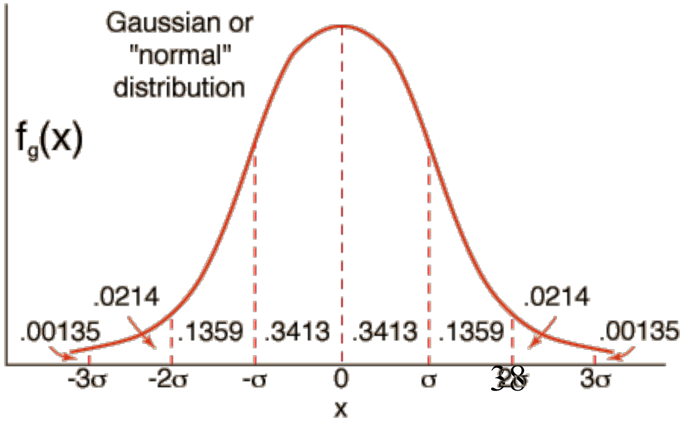
To decide whether to assign point p to a cluster:

- **Compute Mahalanobis distance between p and each of the cluster centroids**
- Choose cluster whose centroid has smallest Mahalanobis distance from p
- **If this distance is less than a threshold, add p to the cluster**
  - E.g., threshold might be 2 or 4 standard deviations from mean
  - If points are truly normally distributed, very small probability we will not include point that should be in the cluster.

Mahalanobis Distance

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

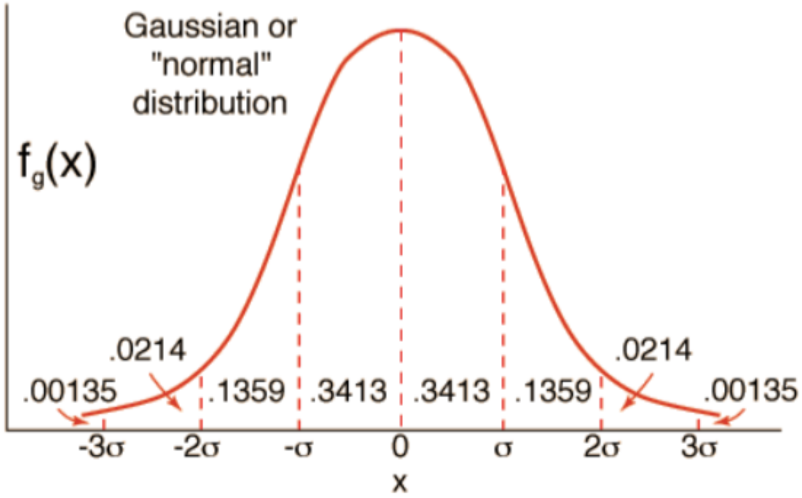
- Accept a point for a cluster if its M.D. is **<** some threshold, e.g. **2** standard deviations



Mahalanobis Acceptance Criterion

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

We have ***d*** dimensions



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

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

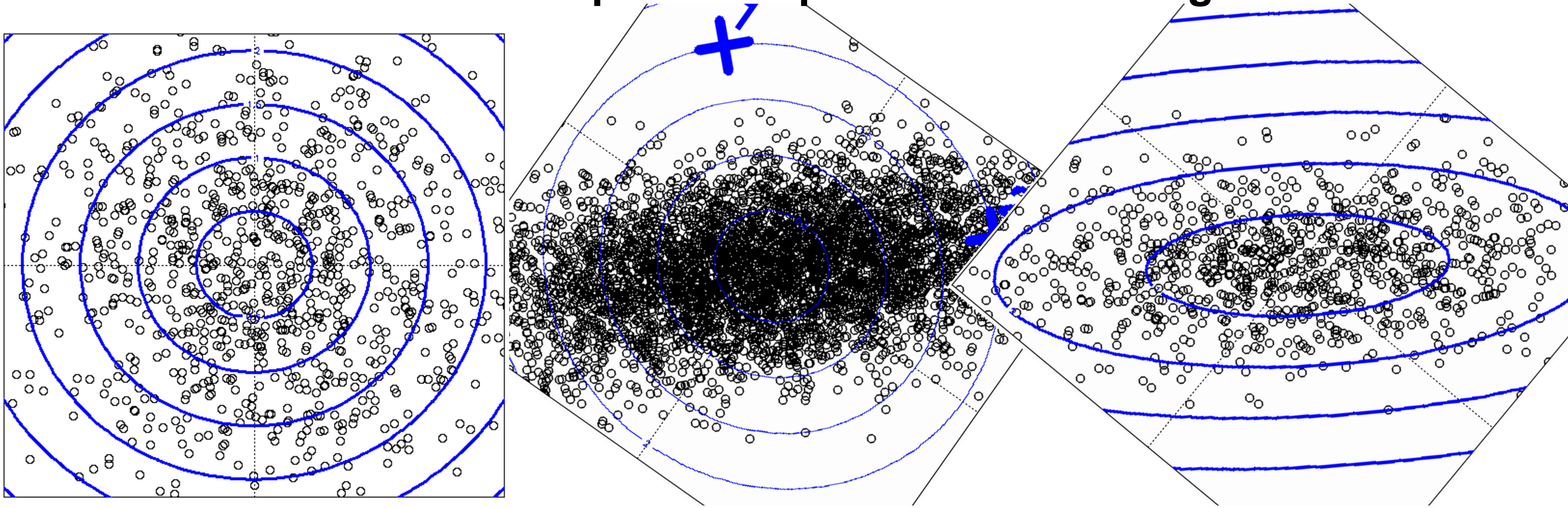
99% of points have MD $\leq 3\sqrt{d}$

Accept 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

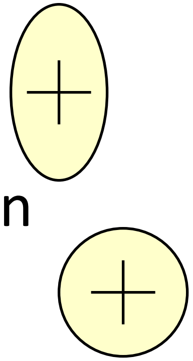
Normally distributed points, Euclidean distance

Normally distributed points, Mahalanobis distance

Should 2 CS clusters be combined?

Q2) Should 2 CS miniclusters be combined?

- Another method (in addition to earlier ones)
- Compute the variance of the combined minicluster
  - ***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.



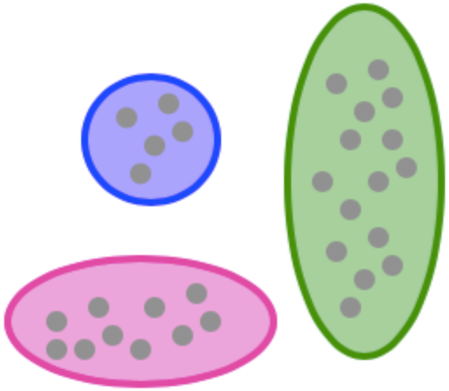
The CURE Algorithm

(Clustering Using REpresentatives)

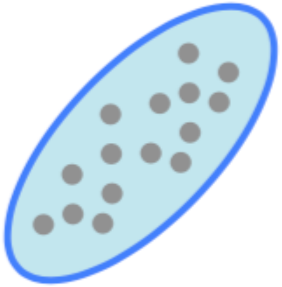
Extension of *k*-means to clusters of arbitrary shapes

Problem with BFR/*k*-means

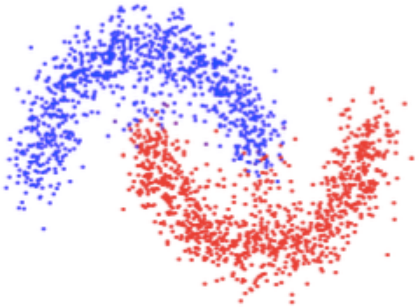
- Assumes clusters are normally distributed in each dimension
- And axes are fixed – ellipses at an angle are ***not OK***



OK

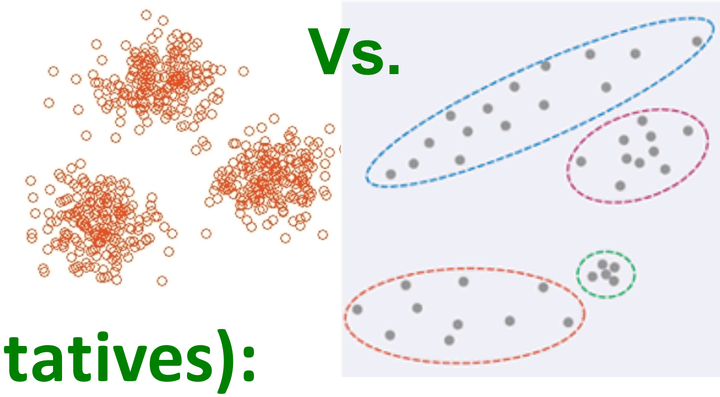


Not OK

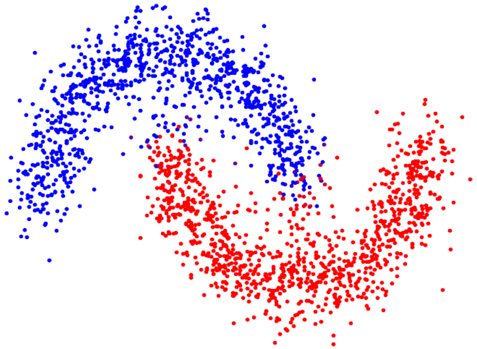


Not OK

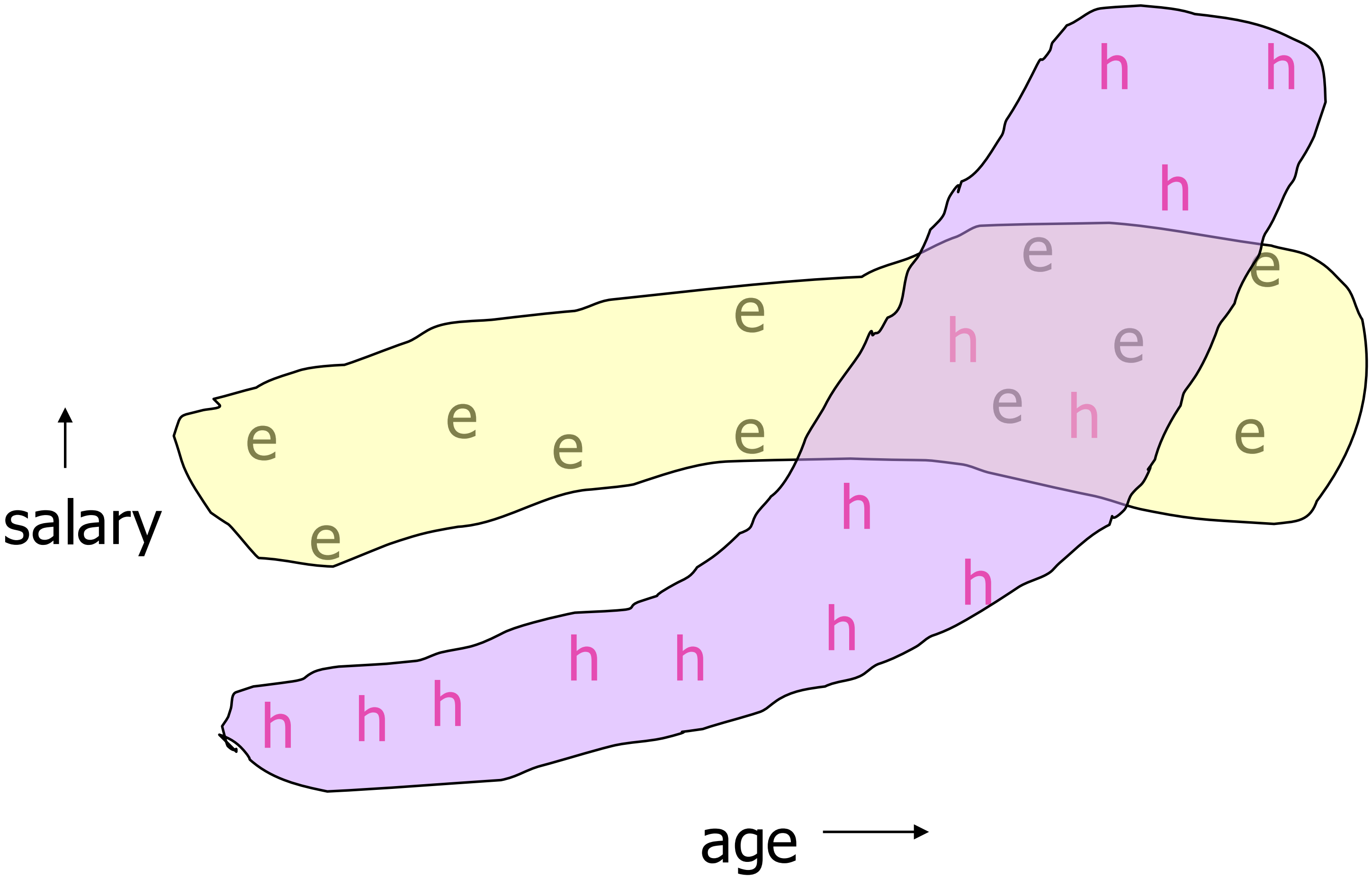
The CURE Algorithm



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



Example: University Salaries

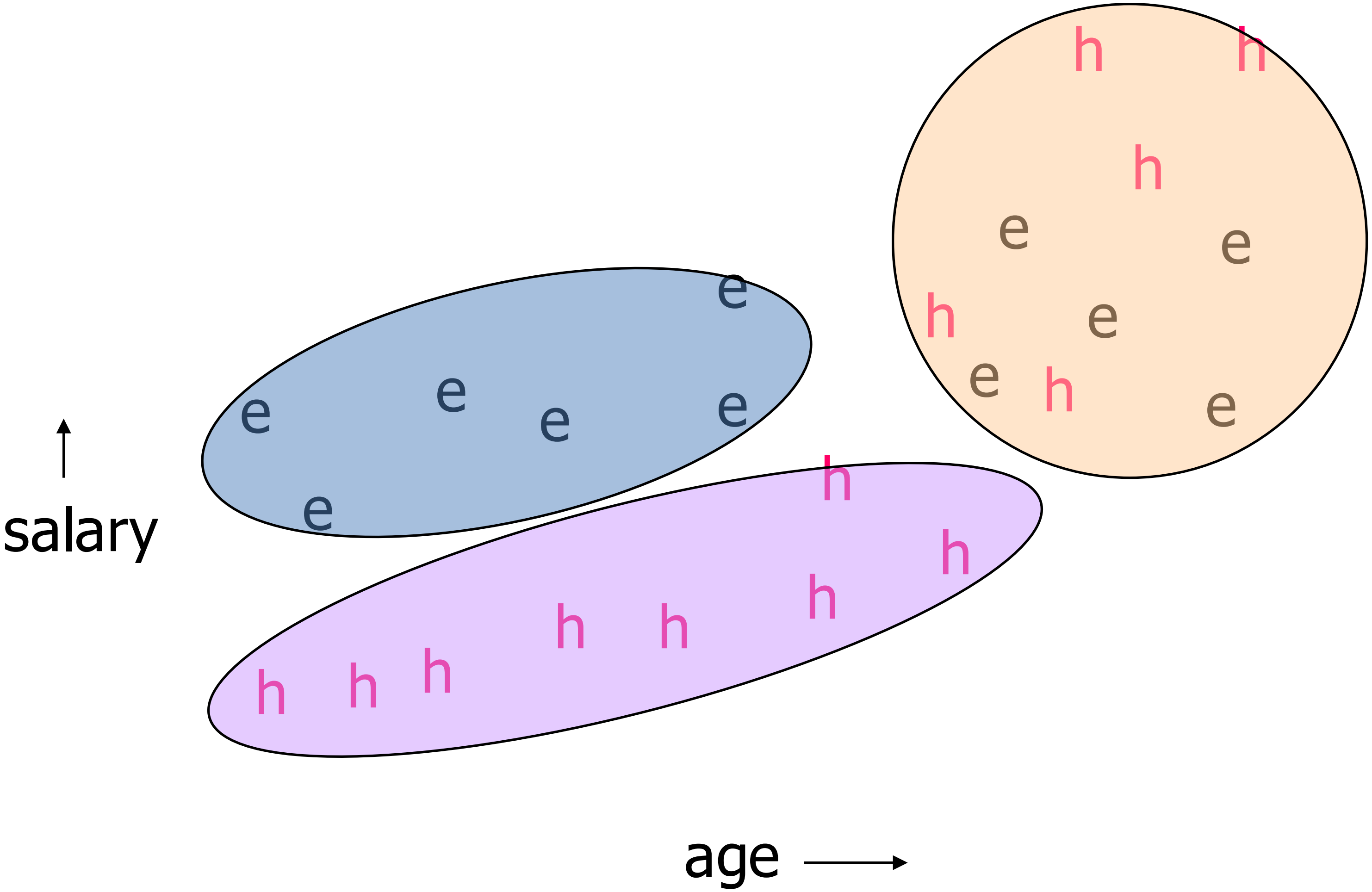


Starting CURE

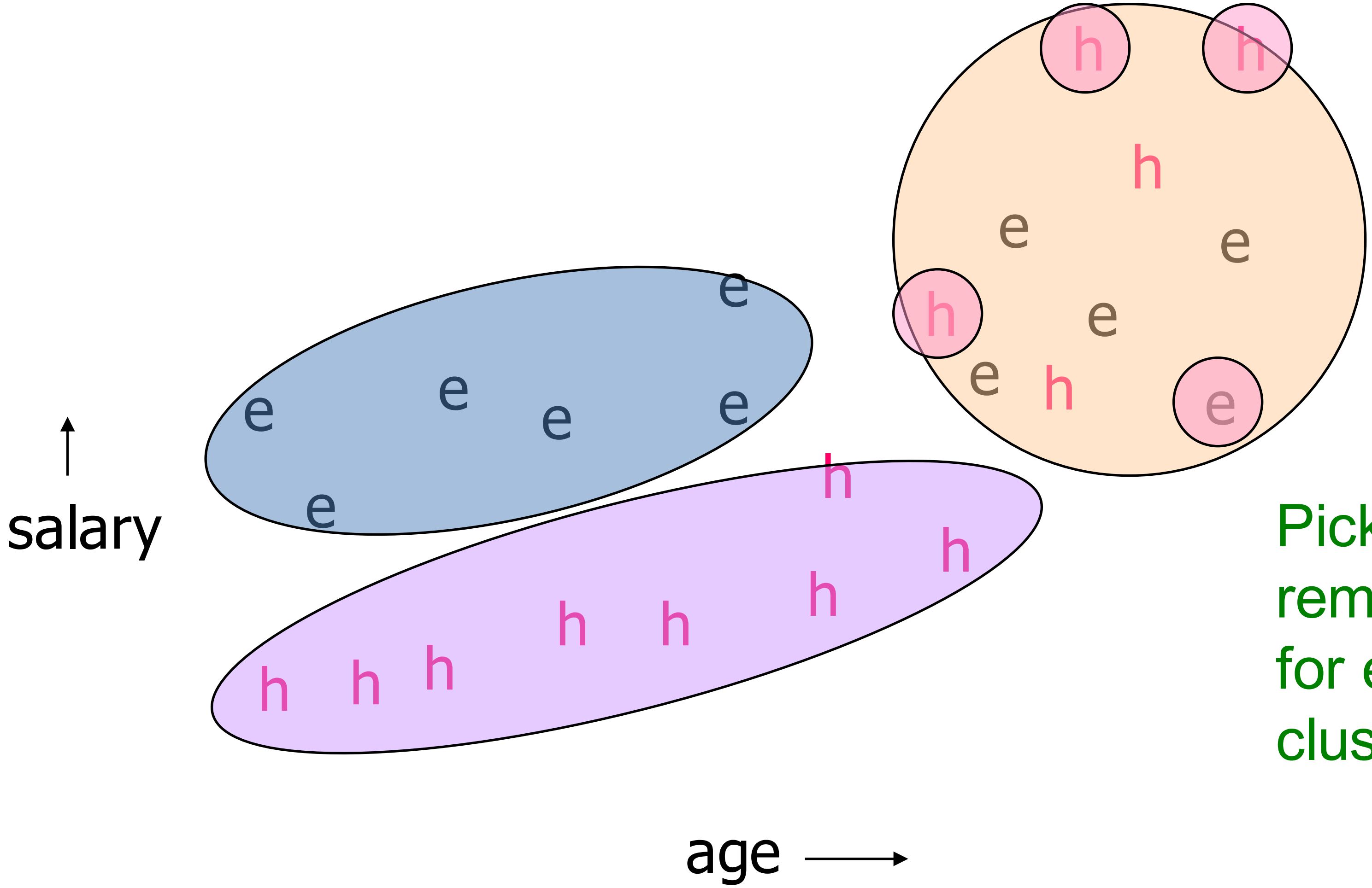
2 Pass algorithm. Pass 1:

- **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

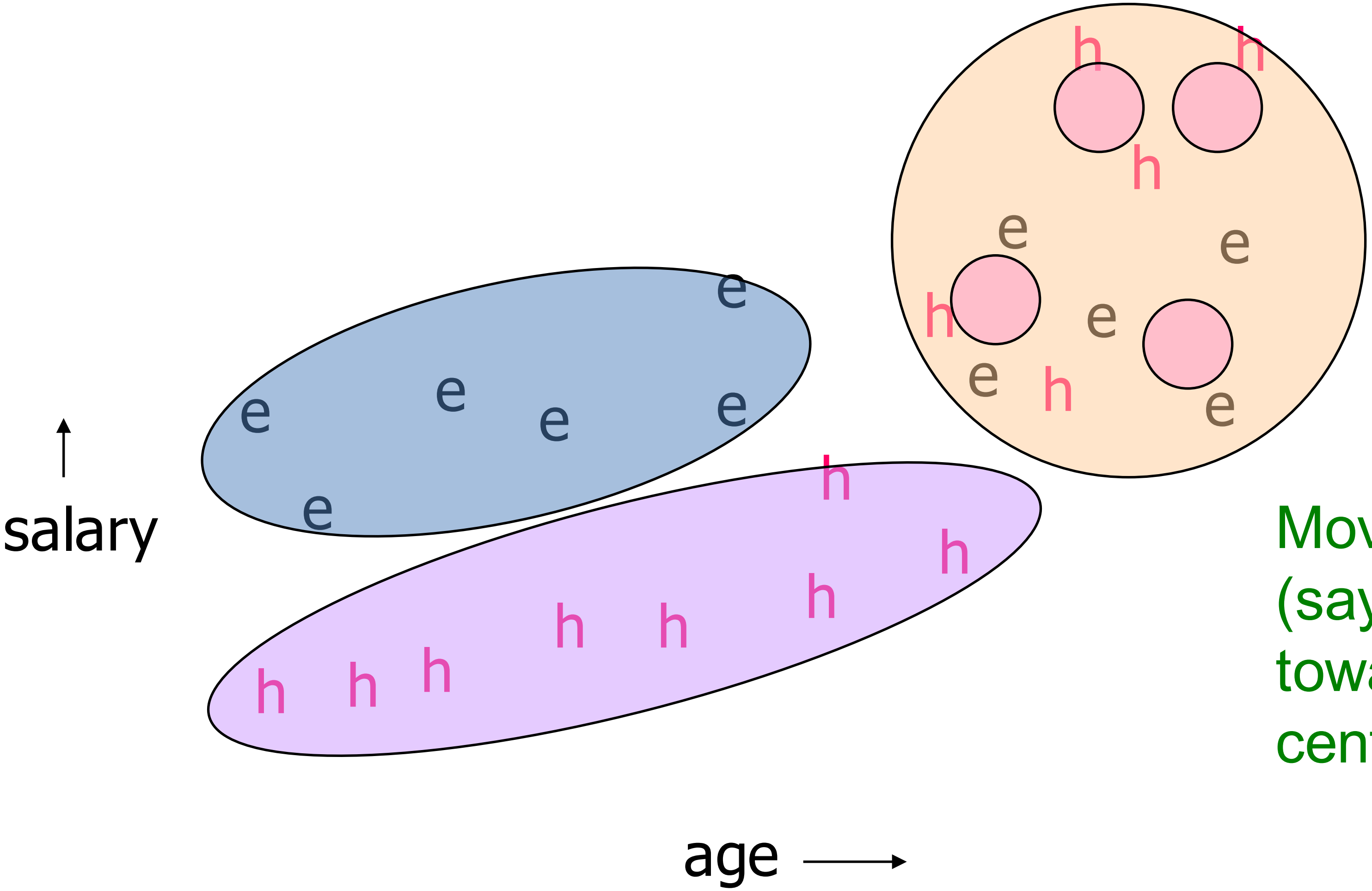


Example: Pick Dispersed Points



Pick (say) 4 remote points for each cluster.

Example: Pick Dispersed Points



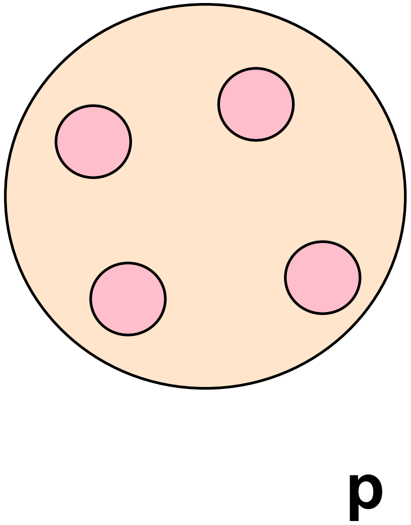
Move points (say) 20% toward the centroid.

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.



Example: A Circle and a Ring Cluster

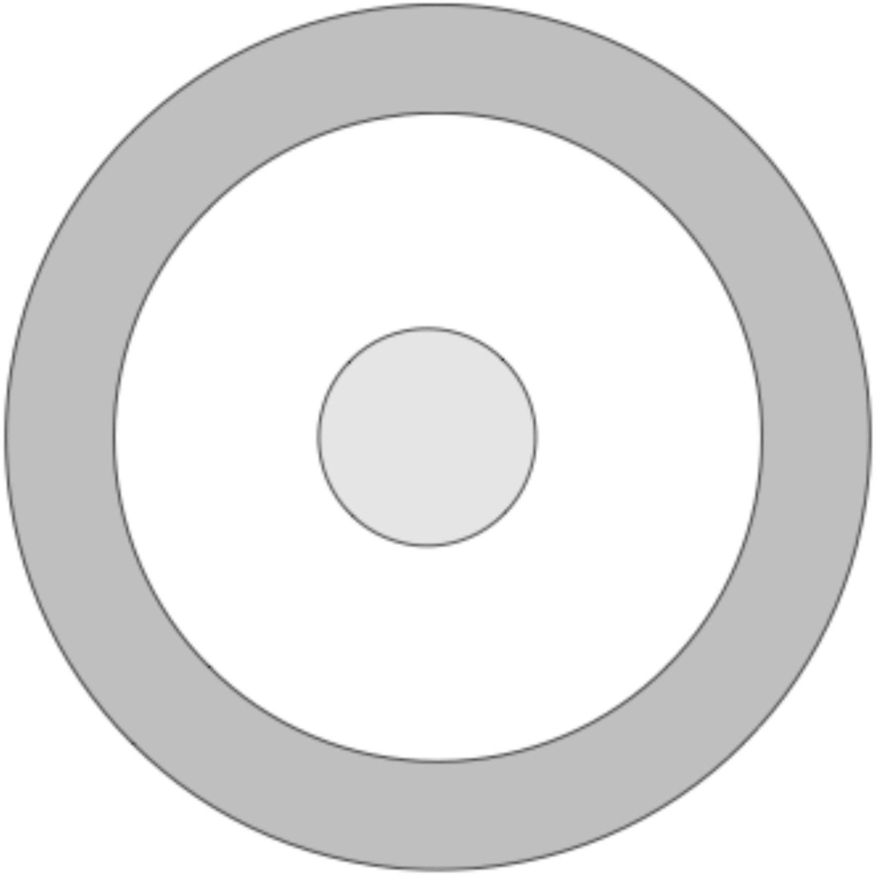


Figure 7.12: Two clusters, one surrounding the other

Starting CURE

2 Pass algorithm

Pass 1:

- **0) Pick random sample of points that fits in main memory**
- **1) Initial clusters:**
  - Cluster these points **hierarchically** – group nearest points/clusters
  - **Note: don’t use distance between centroids: centroids are same!**
- **2) Pick representative points:**
  - For each cluster, pick a sample of **representative points**, as dispersed as possible
- **3) Move representative points:**
  - From the sample, **move representatives by (say) 20% toward the centroid of the cluster**
  - Note: requires a Euclidean space.

Pick Representative Points in Clusters

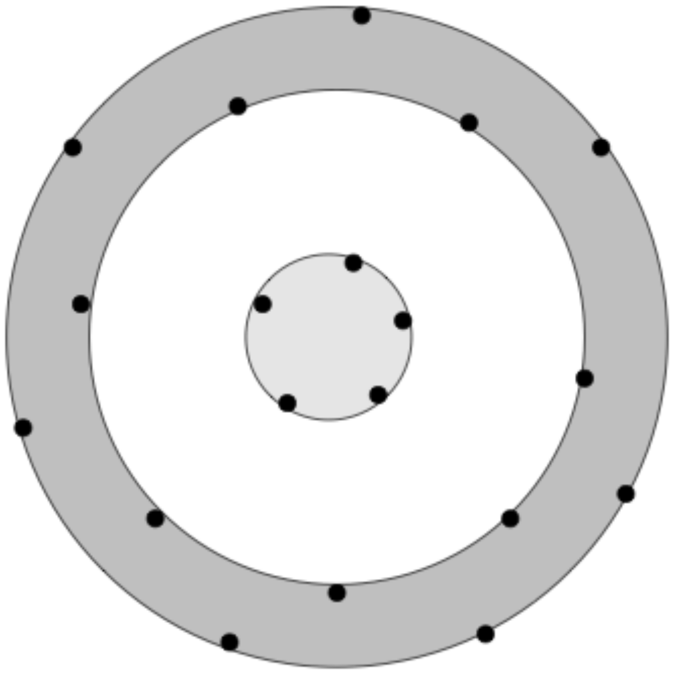


Figure 7.13: Select representative points from each cluster, as far from one another as possible

Move Representative Points 20% of the Distance to Centroid

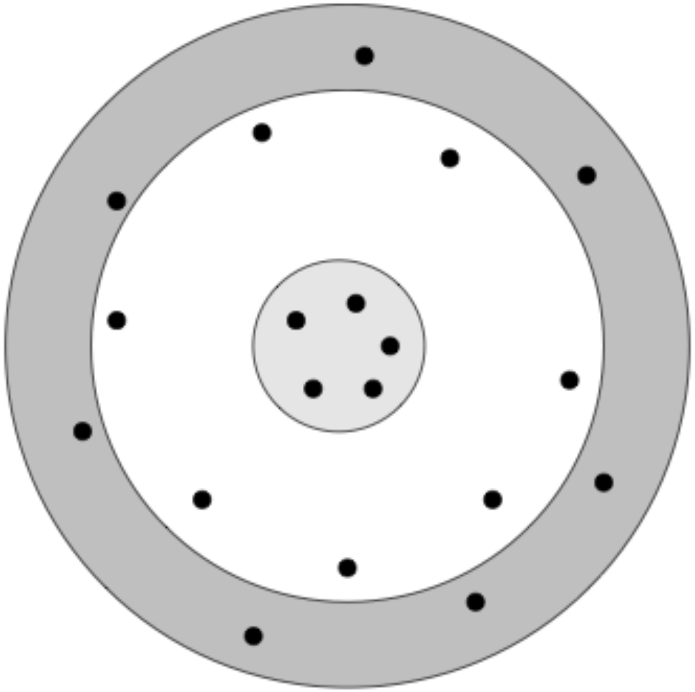


Figure 7.14: Moving the representative points 20% of the distance to the cluster's centroid

Note: both clusters have the same centroid!

Finishing CURE

Pass 2:

- Now, **rescan the whole dataset** and visit each point ***p*** in the data set
- **Place each point in the “closest cluster”**
  - Normal definition of “**closest**”: Find the **closest representative to *p*** and **assign it to the representative’s cluster**
- **End!**

Notes on this Example

- Could argue that the ring and the circle should be merged into **one cluster**, since their centroids are the same

- Whether we have one cluster or two depends on:
  - **Fraction of the distance to the centroid that we move the representative points**
  - **Choice of how far apart representative points of two clusters need to be to avoid merger.**

MapReduce for Clustering Algorithms

- Data Clustering using *MapReduce*: A Look at Various *Clustering Algorithms* Implemented with *MapReduce* Paradigm
- By Varad Meru
- http://www.slideshare.net/VaradMeru/data-clustering-using-map-reduce

MapReduce for Clustering Algorithms

- Start (as usual) with **k points** representing initial cluster centroids
- **Mappers:**
  - **Receive portion of the data points and k centroids**
  - **Do distance calculations** of each point from centroids
  - **Assign each point to its closest centroid**
  - **Emit (clusterID, datapoint)**
- **Reducers:**
  - **Work on a clusterID, list of data points in the cluster**
  - **Computes new centroid** for the cluster
  - **Writes centroid to a new centroid file** for next MapReduce phase
- **Termination:**
  - After specified **number of iterations** or **when clusters stabilize.**

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 (for Euclidean spaces)
    - Clustroid (for non-Euclidean spaces)
  - ***k*-means:**
    - Point assignment; Euclidean space; initialization, picking *k*
  - **BFR**
    - Extends k-means for data sets that don't fit in memory; assumes clusters are normally distributed about a centroid
  - **CURE**
    - Does not restrict the shape of clusters; represent clusters by a collection of representative points.