Clustering 2

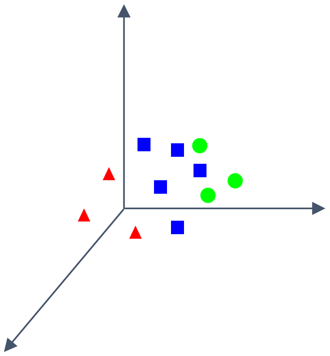
EE412: Foundation of Big Data Analytics

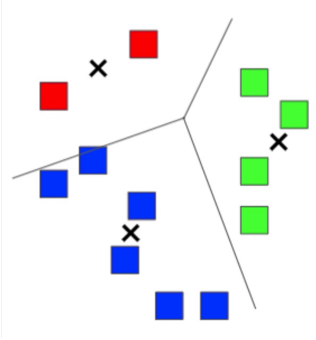
Announcements

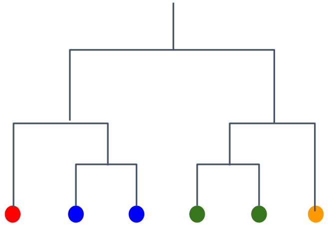
- Homeworks
  - HW1 (due: 10/05)
  - HW2 (will be posted at 10/10)

Recap

- Clustering
  - Curse of dimensionality
  - Clustering strategies
- Hierarchical Clustering
  - Euclidean vs. non-Euclidean
  - Centroids vs. clustroids
- k-means Clustering
  - k-means++
  - Selection of k





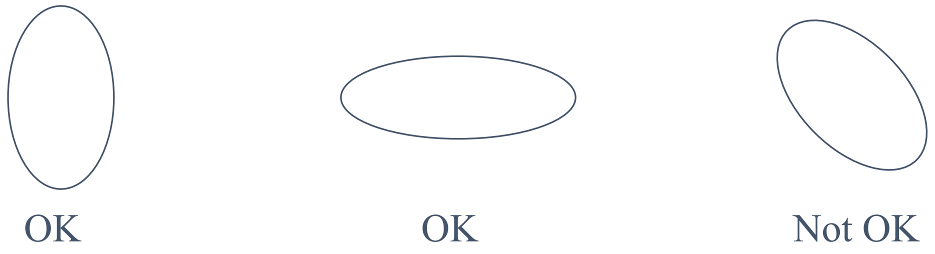


Outline

1. **BFR Algorithm**
2. BFR Algorithm: Process
3. CURE Algorithm
4. GRGPF Algorithm

BFR (Bradley, Fayyad, and Reina) Algorithm

- Variant of k-means for very large (disk-resident) data sets
- Assumes that clusters are normally distributed in a Euclidean space
  - Standard deviations in different dimensions may vary
  - Clusters are axis-aligned ellipses
- **Goal:** Find cluster centroids
  - Point assignment can be done in a second pass through the data



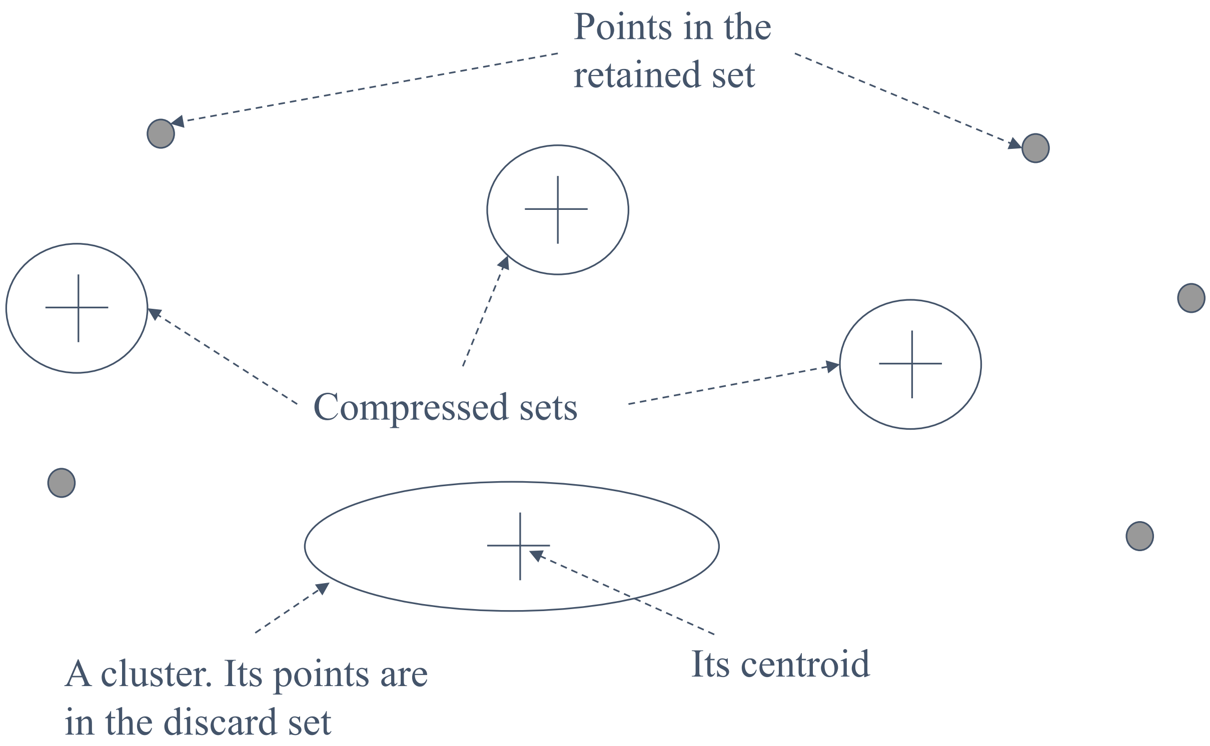
Main Idea

- **Idea:** Keep summary statistics of groups of points
  - Points are read from disk one main-memory-full at a time
  - Most points from previous memory loads are summarized
  - Changes memory requirement from $O(\text{data})$ to $O(\text{clusters})$
- **3 sets:** Discard set, Compressed set, and Retained set

Three Classes of Points

- **Discard set** → no longer use //
  - summarize 되기 충분히 가까운 애들.
  - Points close enough to a centroid to be **summarized**
- **Compressed set** => 서로 부분부분 가깝지만, existing centroid 랑 가깝지는 않은 것. (mini-clustered)
  - Groups of points that are close together but not to any existing centroid
  - These points are **summarized** but not assigned to a cluster
- **Retained set**
  - Isolated points waiting to be assigned to a compressed set

Cluster Visualization



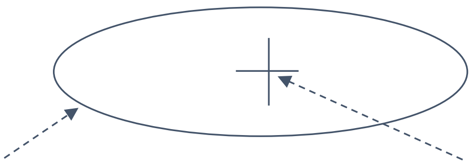
Summarizing Sets of Points

C 특정 cluster에 속한 애들 num of dimensions

- Discard or compressed set is **summarized** by $2d + 1$ values

  SUM_2 := sum of all data of dimension 2

  1개 - The number of points, N
  d개 - The vector SUM, where SUM_i is the sum of the coordinates of all points
  d개 - The vector SUMSQ, where SUMSQ_i is the sum of squared coordinates
    - That is, $\mathrm{SUM}_i = \sum_{k \in C} x_{ki}$ and $\mathrm{SUMSQ}_i = \sum_{k \in C} x_{ki}^2$



A cluster. Its points are stored in memory

Its centroid

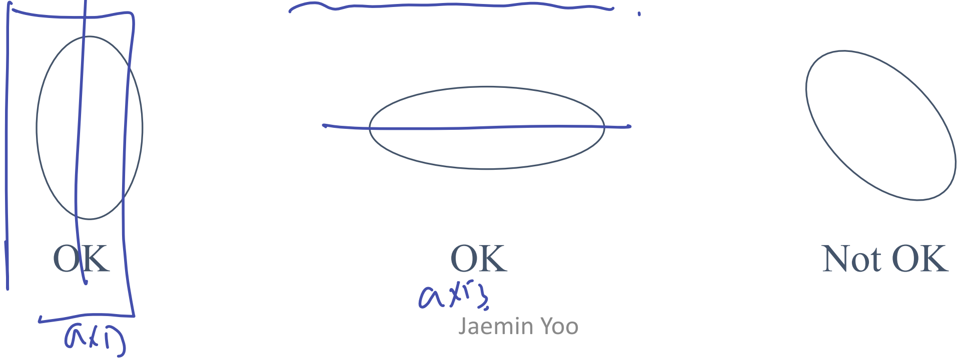
C: Set of points
d: # of dimensions

Summarizing Sets of Points

- We can compute the average and variance of a cluster
  - Average in dimension i (i.e., the **centroid**) is SUM_i/N
  - Variance in dimension i is $\mathrm{SUMSQ}_i/N - (\mathrm{SUM}_i/N)^2$
    - Because $\mathrm{Var}(X) = E[X^2] - E[X]^2$
  - Standard deviation is the square root of variance

=) elipse를 나타내는 enough info

- This is based on the assumption of "axis-aligned" clusters
  - Without it, we need to store the $d \times d$ covariance matrix



Benefits of the Representation

- Easy to add a new point to a cluster
  - Increase N by 1
  - Add the vector to SUM //
  - Add the squares of components to SUMSQ //
- Also easy to combine two sets
  - Add corresponding values of N, SUM, SUMSQ

Pop Quiz

- Represent the cluster of points (5, 1), (6, −2), and (7, 0)
  - N = ? 3
  - SUM = ? (18, -1)
  - SUMSQ = ? 25+36+49 = 110 , 1+4 = 5
- Compute the variance of the first dimension
  - Variance = ?

Outline

1. BFR Algorithm
2. **<u>BFR Algorithm: Process</u>**
3. CURE Algorithm
4. GRGPF Algorithm

compressed set => discard set으로 편입시키지는 않는다.
그냥 가만히 놔둘뿐

Overview of the Algorithm

1. Initialize k clusters/centroids (as in k-means)
2. **for** each chunk **in** a data file
3. **for** each point **in** the chunk
4. Assign it to a cluster if it is sufficiently close to the cluster
5. Cluster the remaining points, creating new clusters
6. Try to merge new clusters with any of the existing clusters

└ 이거를 CS와 합칠지 고려

가깝지 않으면 그냥 remaining으로 남기고
remaining들 끼리 모여서 CS로 만들자. 이 CS를 원래 CS와 합칠지 고민.
여기에 포함되지 않은 애들은 RS임

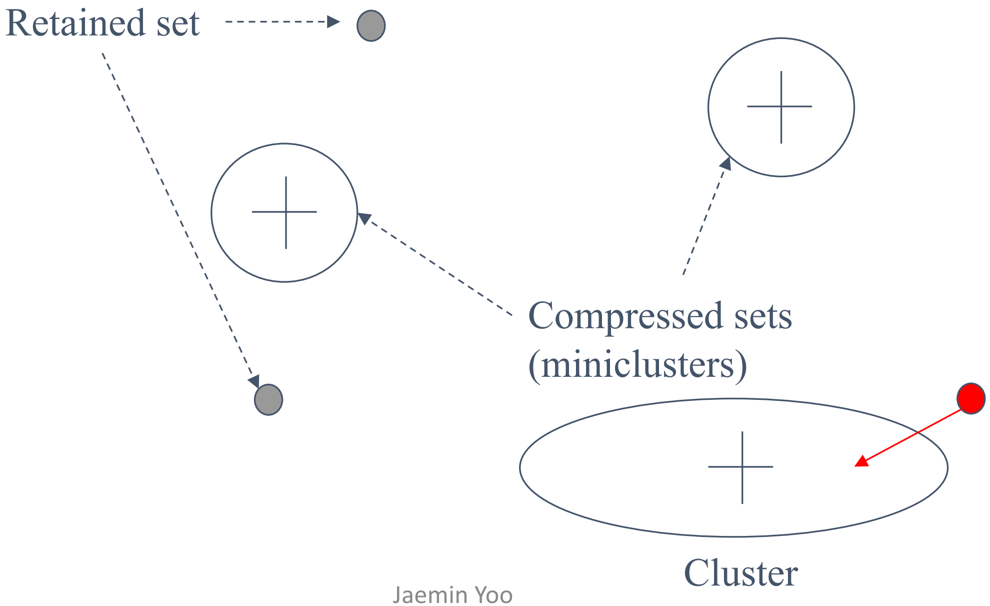
Selection of the Initial Centroids

really important.

- The k initial centroids can be selected as in k-means
  - Take k random points (not a good way, try it)
  - Take a small random sample, cluster it, and use the centroids
  - Take a sample; Pick a random point, and then k − 1 more points
    - Each as far from the previously selected points as possible

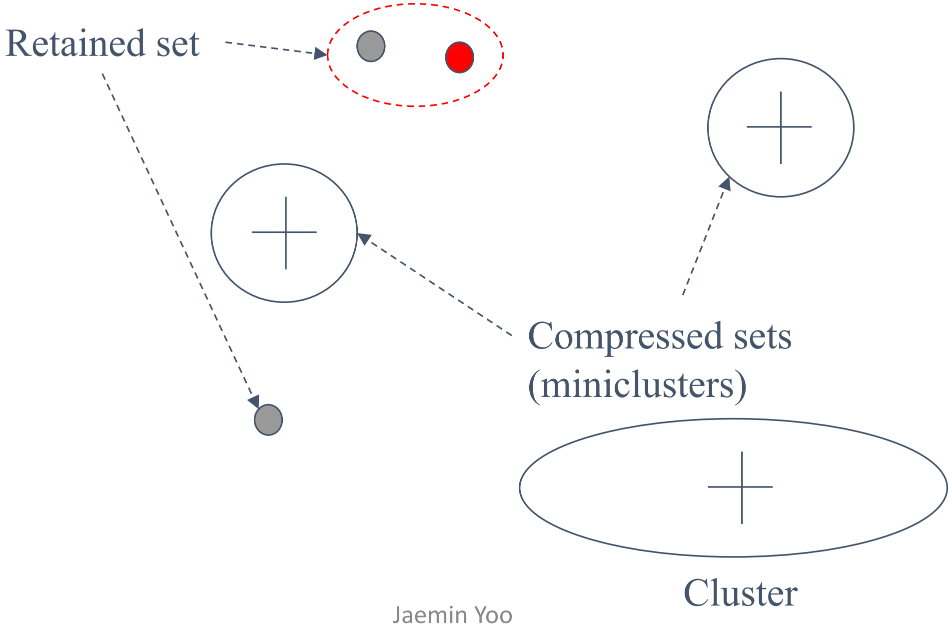
Processing a Chunk of Points (1/5)

- All points that are “sufficiently close” to the centroid of a cluster:
  - Added to that cluster



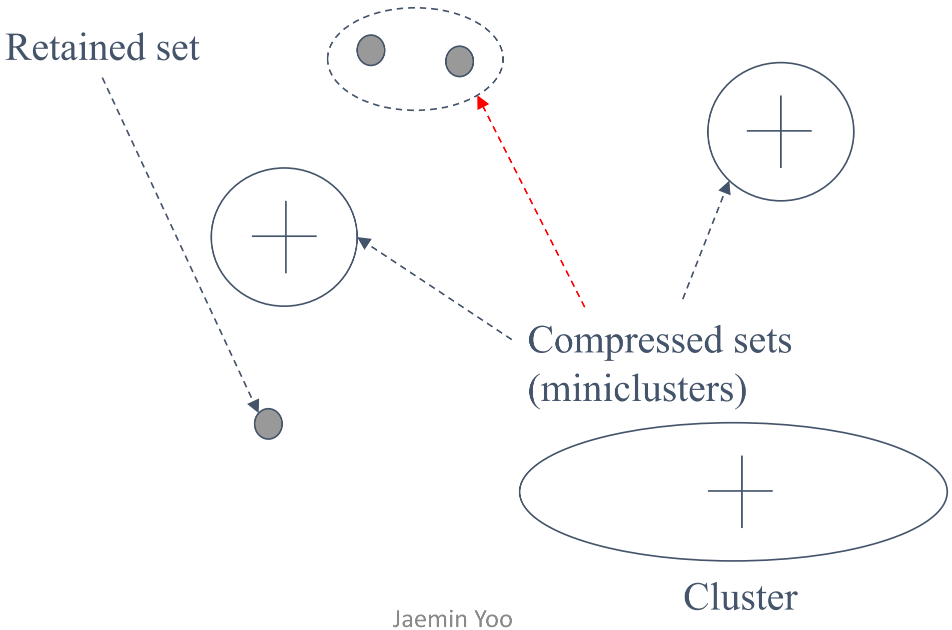
Processing a Chunk of Points (2/5)

- The points that are not sufficiently close to any centroid:
  - Clustered with the points in the retained set using any clustering algorithm



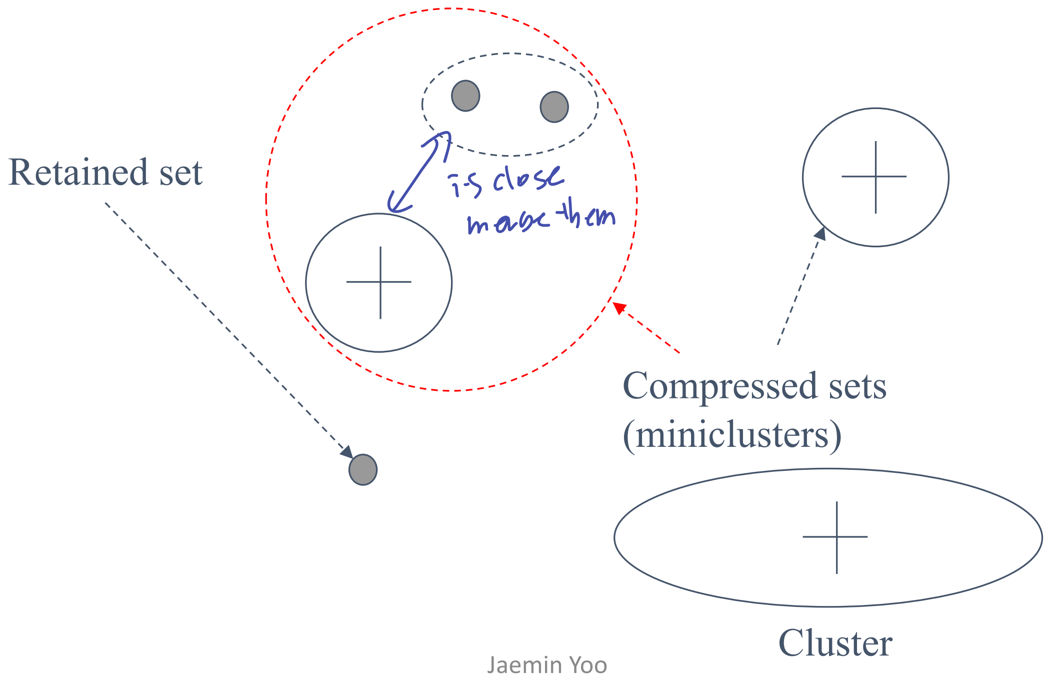
Processing a Chunk of Points (3/5)

- Clusters of $\geq$ 2 points are summarized and become miniclusters
- Singleton clusters remain in the retained set



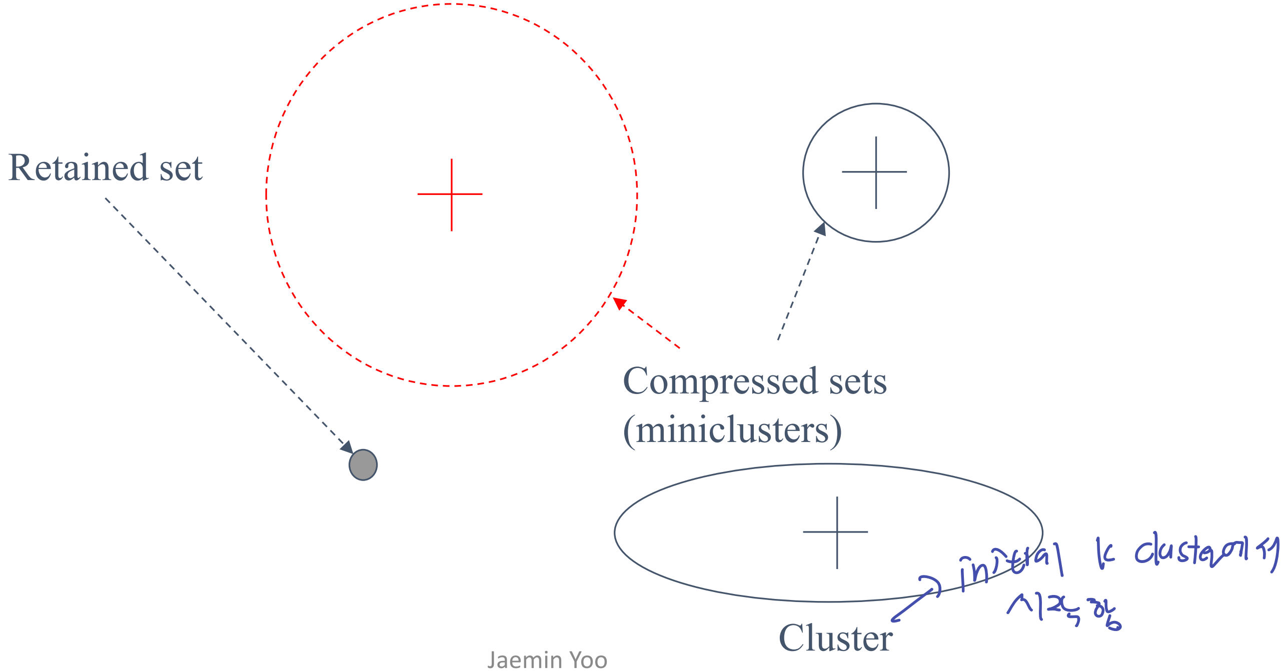
Processing a Chunk of Points (4/5)

- Cluster the new miniclusters with the old miniclusters



Processing a Chunk of Points (5/5)

- Points assigned to a cluster or a minicluster are written to disk



After Processing All Chunks

- At the last round, what to do with compressed and retained sets?
- **Option 1:** Treat them as outliers and never cluster them strong way
- **Option 2:** Assign each of them to the nearest cluster 그냥 버려버리기. (remove all outliers)
  - For the compressed set, combine each minicluster with the nearest cluster

How Close is Close Enough?

얼마나 가까워야 하냐.

- Need a way to decide whether to put a new point into a cluster

→ basically euclidean.

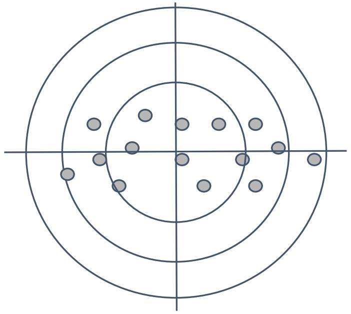
- BFR compares the **Mahalanobis distance** with a threshold
  - Exploit the assumption that points are normally distributed
  - Euclidean distance from the centroid c normalized by standard dev. σ_i

assumption (elips shape)

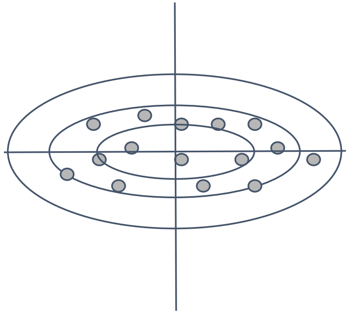
std로 정규화해서 전체 분포에서 내가 어디 있냐

- **Definition:** $\sqrt{\sum_{i=1}^{d}\left(\frac{p_i - c_i}{\sigma_i}\right)^2}$

Euclidean distance:



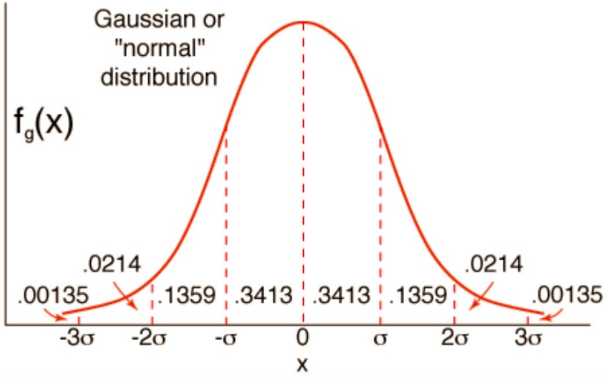
Mahalanobis distance:



Assigning a Point to a Cluster

- Choose a cluster whose centroid has the least Mahalanobis distance
- Add a point p if the distance is less than a threshold
  - E.g., if threshold = 4
  - Then, $\Pr(\text{value being 4 standard deviations from mean}) < 10^{-6}$

likelihood



Source: Stanford CS246 (2022)

When to Merge Two Clusters?

- Compute the variance of the combined subcluster
- Combine if the combined variance is below some threshold
- **Many alternatives:** E.g., considering density

Outline

1. BFR Algorithm
2. BFR Algorithm: Process
3. **<u>CURE Algorithm</u>**
4. GRGPF Algorithm

assumption 이나 여전히 strong하지만
scale을 위해 sacrifice 하는거임.
(ex, k-initial point 잡는게 super important함)

CURE (Clustering Using REpresentatives)

- CURE is a 2-pass algorithm for large disk-resident data
- No assumption about the shape of clusters
  - No need to be normally distributed in each dimension

How is it possible?

- Uses a collection of representative points to represent clusters
  - No centroids

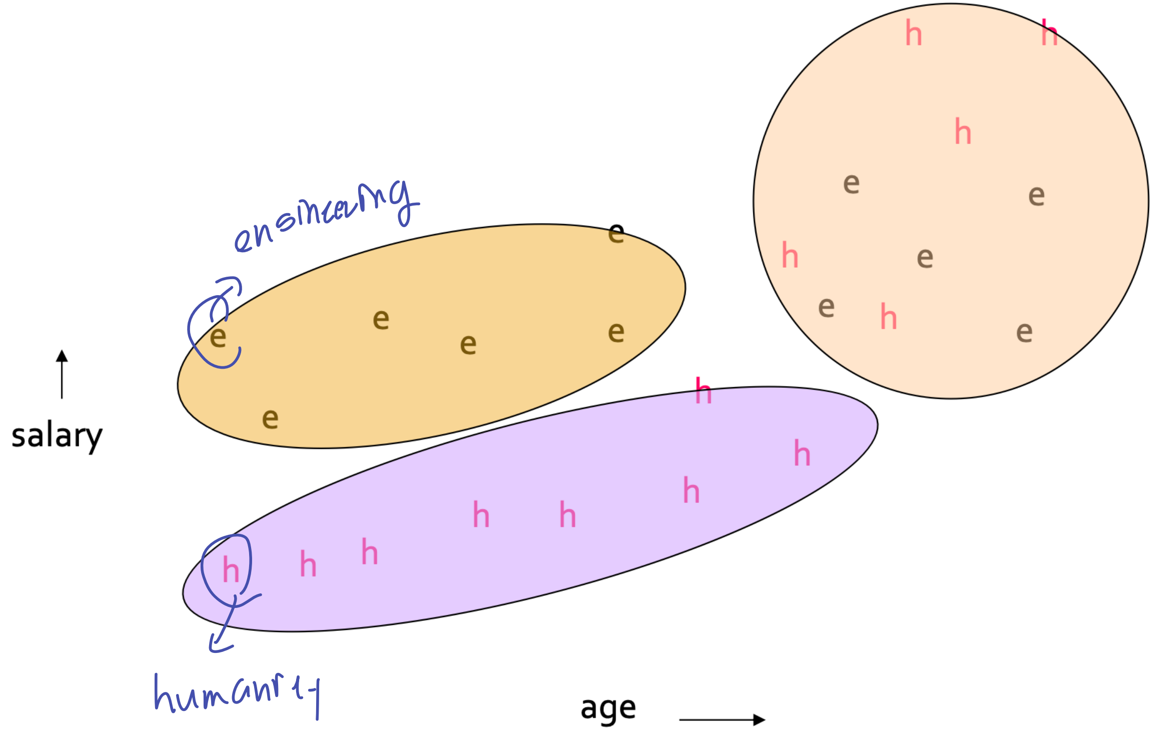
=> Not statistics
more naive way, but effective

- Assumes a Euclidean distance, with k (# of clusters) given

CURE: Pass 1

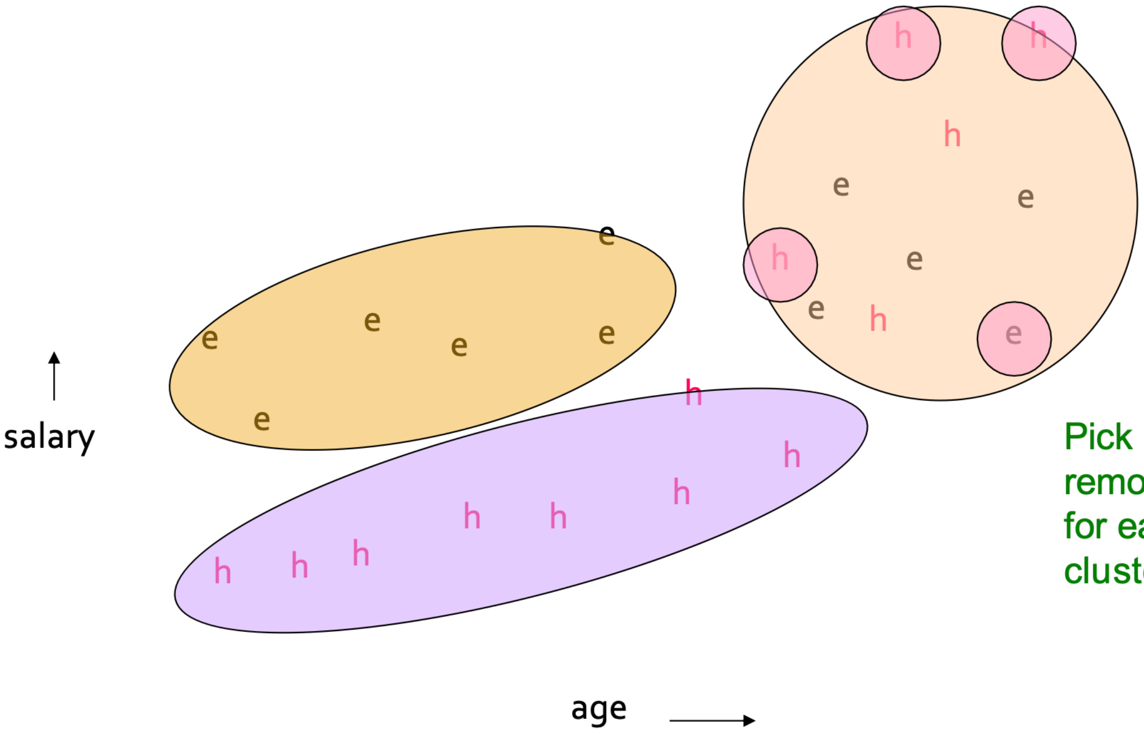
1. Pick a random sample of data   random pick
2. Cluster them in main memory using hierarchical clustering
   - Merge two clusters when they have close pairs of points   to remove shape assumption × concept of centroids
3. Pick representative points from each cluster
   - For each cluster, pick a sample of points, as dispersed as possible
     far as possible
   - Move them a fraction of distance, e.g. 20%, toward the centroid
4. Merge clusters with the closest pair of representatives

Example: Picking Dispersed Points



Source: Stanford CS246 (2022)

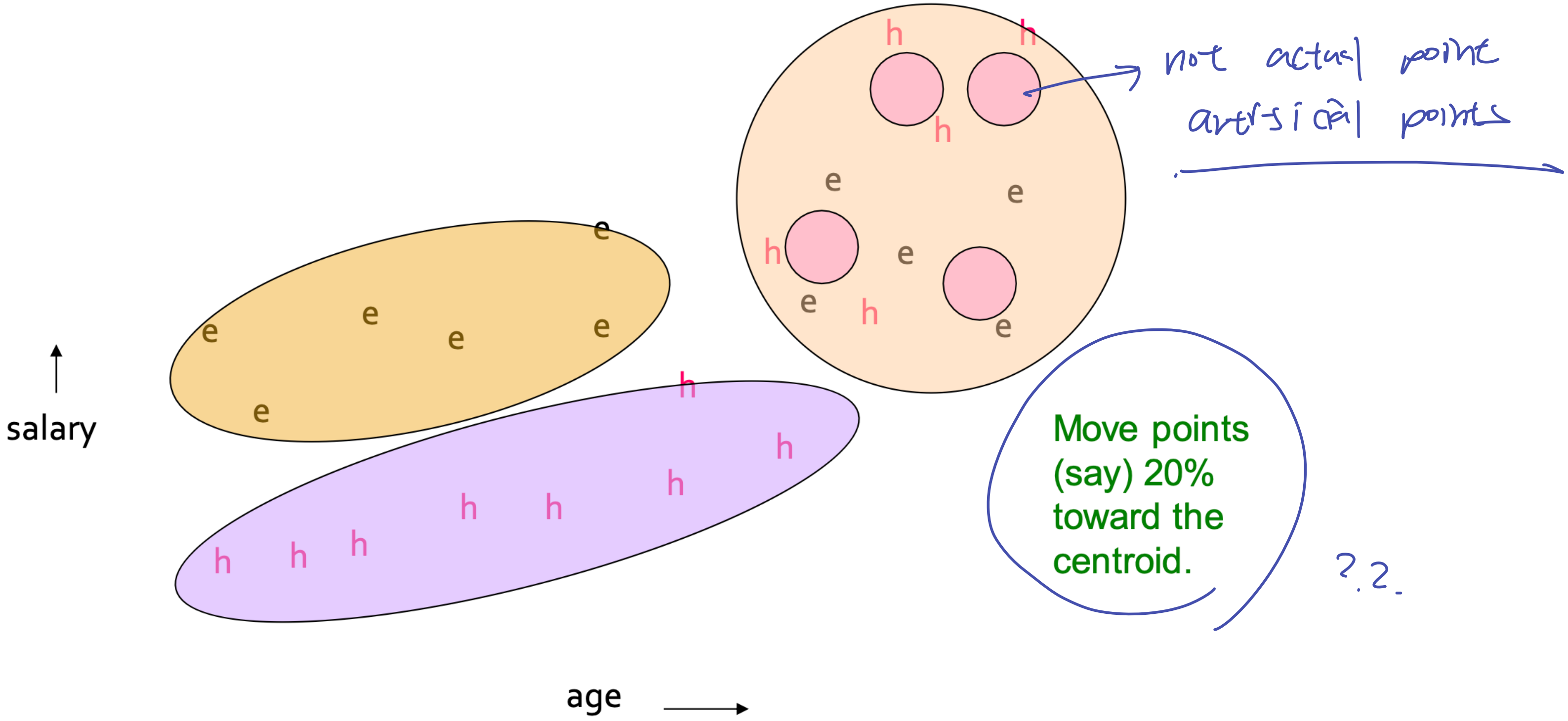
Example: Picking Dispersed Points



Pick (say) 4 remote points for each cluster.

Source: Stanford CS246 (2022)

Example: Picking Dispersed Points



Source: Stanford CS246 (2022)

CURE: Pass 2

1. Rescan the whole dataset and visit each point p in the data set
2. Place it in the "closest cluster"
   - Find the closest representative point to p
   - Assign p to the representative's cluster

Why to 20% Move Inward?

- Suppose that initial sample is large enough
- Some of the representatives will be on the boundary of clusters
  - Moving them towards the centroid
- Large, dispersed clusters will shrink more than small, dense ones
- As a result, the algorithm favors a small, dense cluster

기본적으로 그냥 진행하면
dispersed 된 친구들이 더 점들을
잘 흡수 한다.

=> dense 하고
small 인 애들이
불리함

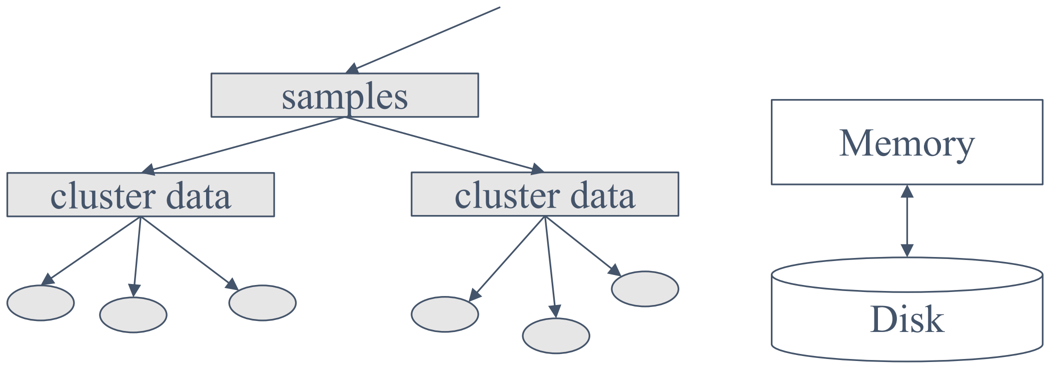
Source: Stanford CS246 (2022)

Outline

1. BFR Algorithm
2. BFR Algorithm: Process
3. CURE Algorithm
4. **GRGPF Algorithm**

GRGPF Algorithm

- Does not require a Euclidean space
- Represents clusters by **well-chosen sample points** in memory
- Organizes clusters hierarchically, as a tree (not covered today)
  - New point is assigned to a cluster by passing it down the tree

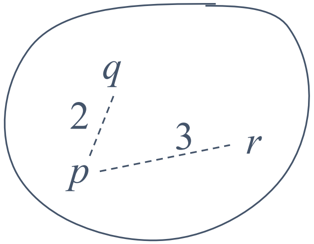


Cluster Representation

- How to represent (or summarize) a cluster in GRGPF
  - The number of points, N
  - The clustroid c x assume euclidian
  - The **rowsum** of the clustroid
    - Sum of the squares of the distances from p to each point in the cluster
  - The k points that are **nearest** to the clustroids, and their rowsums
  - The k points that are **furthest** from the clustroids, and their rowsums

x proportional to num of data

$$\textit{rowsum} \text{ of point } p \text{ in cluster } C = \sum_{c \in C} d(p, c)^2$$



Justification of the Representation

rowsum => smallest rowsum (clustroid)

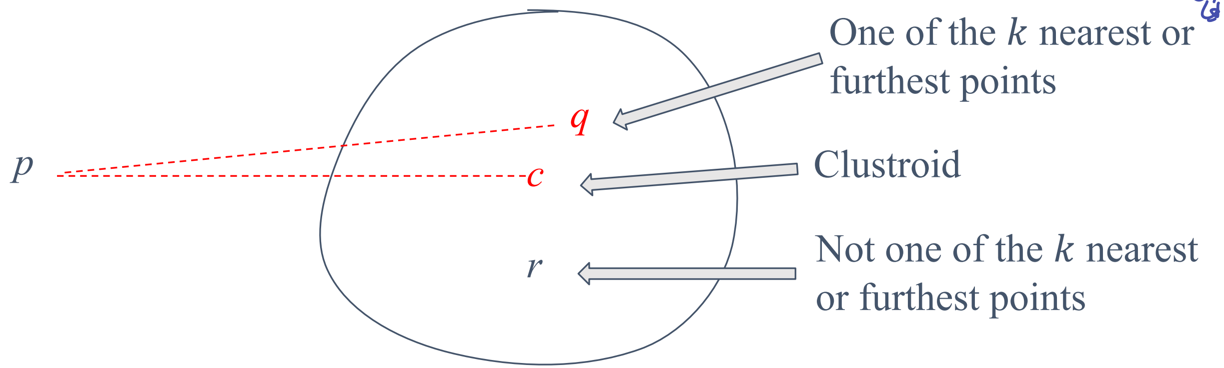
- Clustroid is the point in the cluster with the smallest rowsum
- Why the k points nearest to the clustroids?
  - If the clustroid changes, the new clustroid would be one of them
  - p becomes the new clustroid if $\mathrm{rowsum}(p) < \mathrm{rowsum}(c)$
- Why the k points farthest to the clustroids?
  - Used to determine whether two clusters are close enough to merge

??.

뭔가 족같은거

Adding a Point to a Cluster

- How can we add a point p to a cluster?
  - Add 1 to N
  - For each $q \in \{\text{clustroid}\} \cup k$ nearest points $\cup\ k$ furthest points
  - Update $\text{rowsum}(q)$ as $\text{rowsum}(q) + d^2(p, q)$ //
- What if p needs to be included in the representation?
  - We cannot compute this exactly without going to disk ⇒ 다른 점들의 정보가 없어서 p의 rowsum을 알 수가 없다.

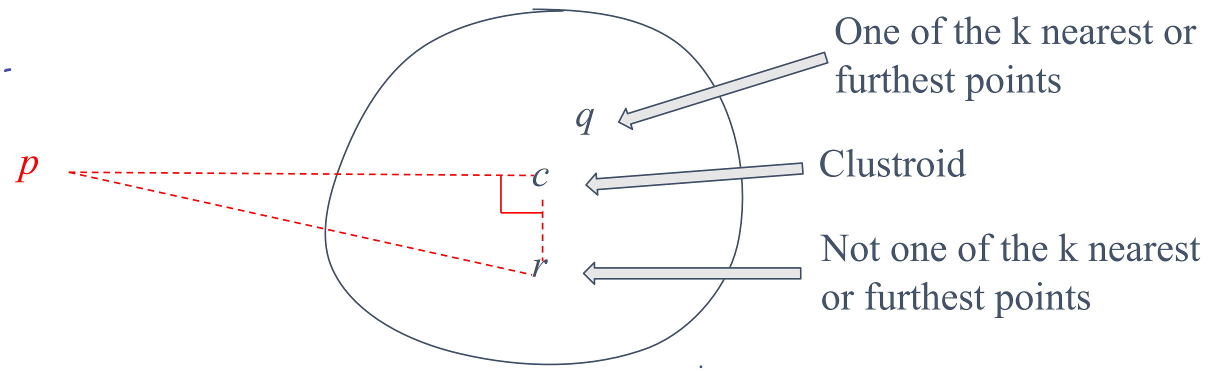


KAIST

Estimating the Rowsum

- **Estimation:** $\mathrm{rowsum}(c) + N \times d^2(p, c)$ //
- With the curse of dimensionality, almost all angles are right angles, 모든 점들 간의 각도가 90으로 감
- Thus, $d^2(p, r) \approx d^2(p, c) + d^2(c, r)$ by the Pythagorean theorem //

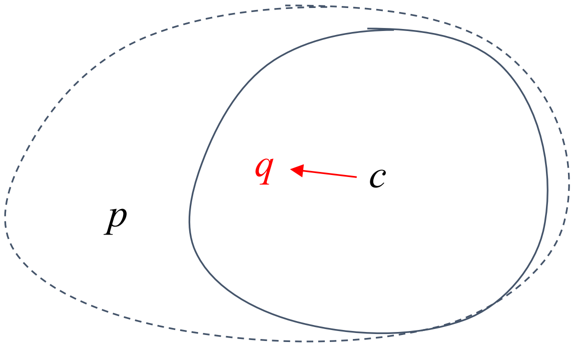
2개. 거리? angle?



Possibly Updating the Clustroid

- If $\text{rowsum}(p) < \text{rowsum}(c)$, make p the new clustroid
- Eventually, the true clustroid may not be one of the k closest points
    - Cluster representation needs to be recomputed periodically from disk

why?



Other Details of GRGPF

- See the textbook for other details:
  - How to initialize the cluster tree
  - How to use the tree for each new point
  - How to split a cluster
  - How to merge clusters (with the k furthest points)

Summary

1. BFR Algorithm
   - Cluster representation
   - Three classes of sets
2. BFR Algorithm: Process
3. CURE Algorithm
   - 2-pass algorithm
4. GRGPF Algorithm
   - Rowsum
   - Estimation of a rowsum