universidade de aveiro 

Mining Large Scale Datasets

Clustering

(Adapted from CS246@Starford.edu; http://www.mmds.org)

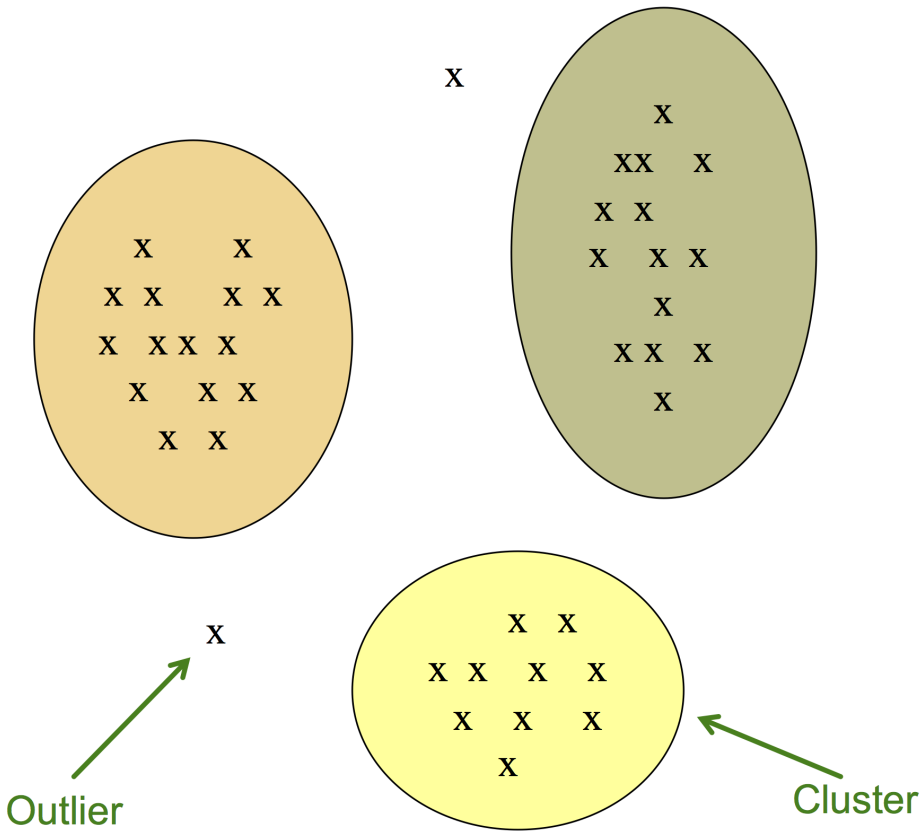
Sérgio Matos - aleixomatos@ua.pt

Clustering

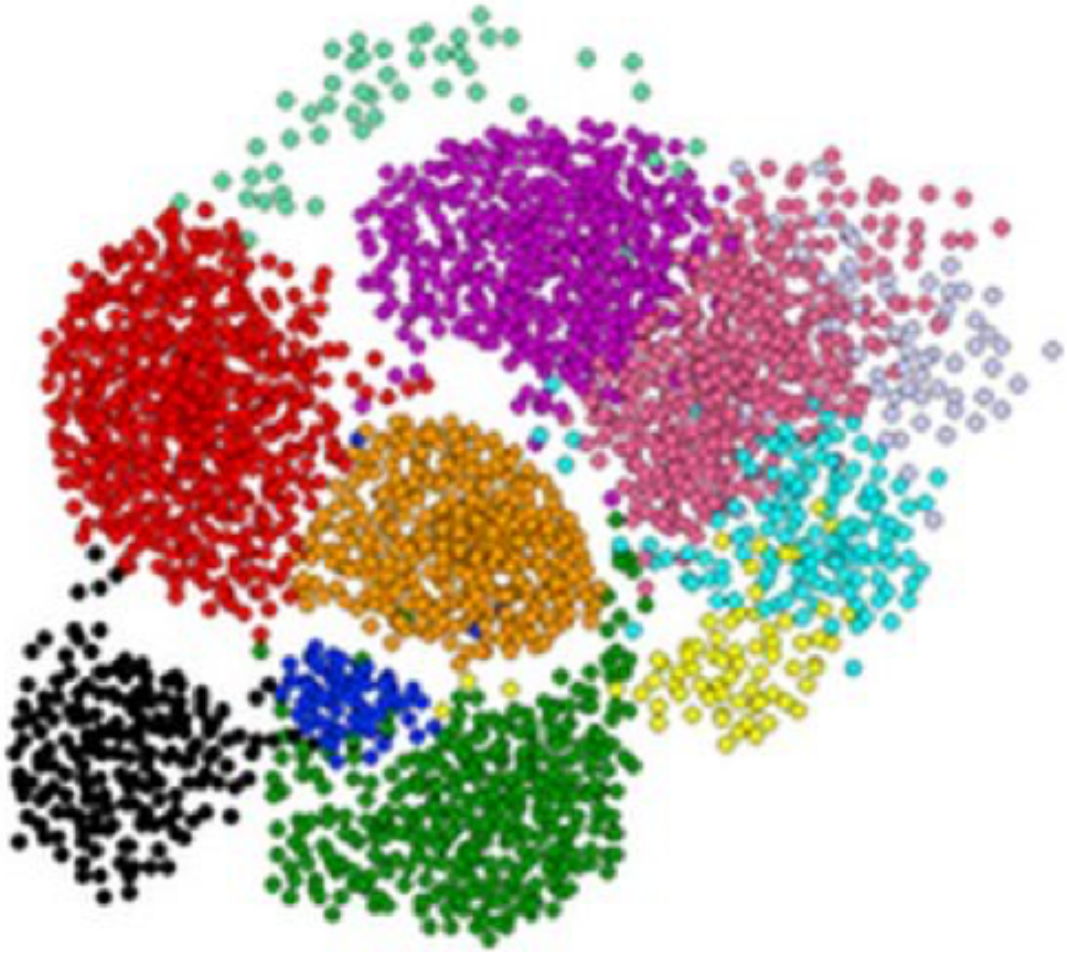
- Given a **set of points**, with a notion of **distance** between points, **group** the points into some number of **clusters**, so that
  - Members of a cluster are close/similar to each other
  - Members of different clusters are dissimilar
- Usually
  - Points are in a **high-dimensional space**
  - Similarity is defined using a distance measure

    Euclidean, Cosine, Jaccard, edit distance, ...

A simple clustering example



But... clustering is a difficult problem



What makes clustering a difficult problem?

- Clustering in two dimensions is simple
- Clustering small amounts of data is simple

- Many applications involve not 2, but 10's or 1000's dimensions
- **High-dimensional spaces behave differently**

  Almost all pairs of points are very far from each other

  $\hookrightarrow$ **Curse of Dimensionality**

Example: Music CDs

- Music can be divided into categories, and customers prefer a few genres

  But what are these categories really?

- Represent a CD by a set of customers who bought it
- Space of all CDs
  - space with one dimension for each customer
  - A CD is a "point" in this space $(x_1, x_2, ..., x_k)$, where $x_i = 1$ iff customer i bought the CD
  - For Amazon, the dimension is tens of millions
- Task: Find clusters of similar CDs

Example: Documents

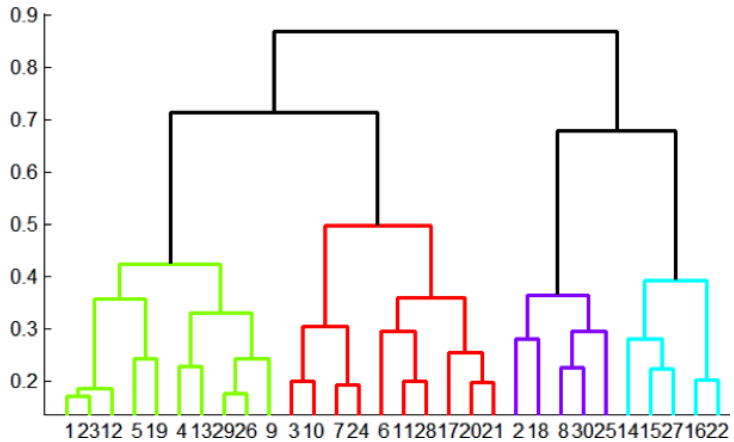
- Finding topics in documents
- Represent a document by a vector

  $(x_1, x_2, ..., x_k)$, where $x_i = 1$ iff the i^{th} word (in some order) appears in the document

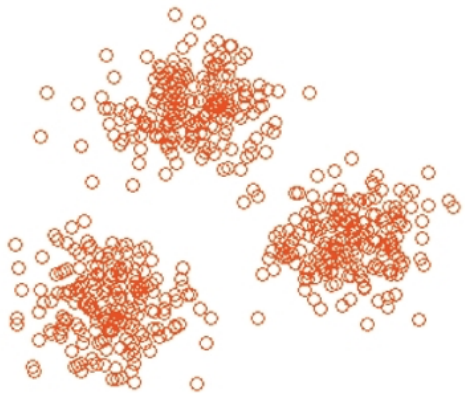
- Documents with similar sets of words may be about the same topic

Clustering methods

- Hierarchical:
  - Agglomerative (bottom up)
  - Divisive (top down)



- Point assignment:
  - Select number of clusters
  - Initialize (e.g. select random points)
  - Assign points to "nearest" cluster



Agglomerative 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?

Agglomerative hierarchical clustering

1) How to represent a cluster of many points?

   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

3) When to stop combining clusters?

   i) Stop at some predefined number of clusters

   ii) Stop if 'cohesion' drops

Agglomerative hierarchical clustering

What about **Non-Euclidean spaces**?

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

1) How to represent a cluster of many points?
   clustroid = (data)point "closest" to other points

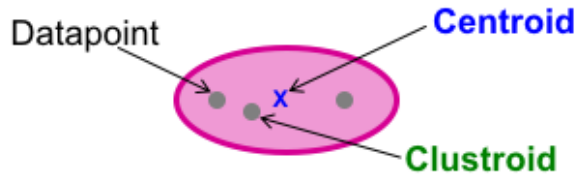
2) How to determine "nearness" of clusters?

Clustroid

- **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

Centroid is an "artificial" point
average of all (data)points in the cluster

Clustroid is an existing (data)point
"closest" to all other (data)points in the cluster



Agglomerative hierarchical clustering

What about **Non-Euclidean spaces**?

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

1) How to represent a cluster of many points?

   clustroid = (data)point "closest" to other points

2) How to determine "nearness" of clusters?

   Approach 1: Treat clustroid as if it were centroid, when computing intercluster distances

   Approach 2: No centroid, take intercluster distance as either:

   i) min distance between any two points, one from each cluster

   ii) avg distance of all pairs of points, one from each cluster

3) When to stop?

- Stop at some predefined number of clusters
- Stop if next merge creates a cluster with low cohesion
  - Radius/diameter of resulting cluster is above some threshold

    Radius: maximum distance between all the points in the cluster and the centroid (or clustroid)

    Diameter: maximum distance between any pair of points in the cluster

  - Density of resulting cluster is below some threshold

    Density is the number of points per unit volume

    Divide number of points by some power of the diameter (or radius) of cluster (e.g. d^1, r^2 or r^3)

Point assignment: k-means

- Assumes Euclidean space/distance
- Start by picking k, the number of clusters
- Initialize clusters by picking one point per cluster

  Approach 1: 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)

  Approach 2: Cluster a sample of the data into k clusters, possibly hierarchically; select one point from each cluster

k-means

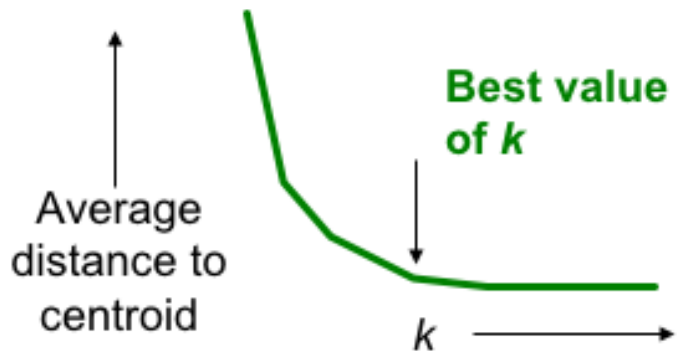
1. Initialize k clusters
2. For each point, place it in the cluster whose current centroid it is nearest
3. After all points are assigned, update the locations of centroids of the k clusters
4. Reassign all points to their closest centroid

   Sometimes moves points between clusters

Repeat 2 and 3 until convergence:
Points don't move between clusters and centroids stabilize

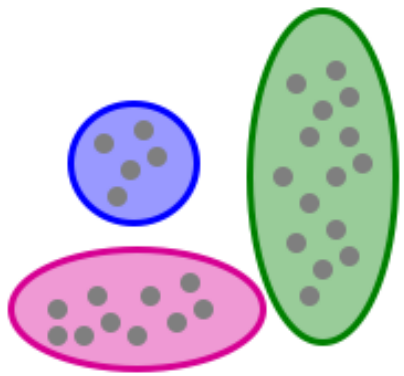
k-means: selecting k

- 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



Bradley-Fayyad-Reina (BFR) Algorithm

- 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 Algorithm

- Efficient way to summarize clusters
  - Lower memory requirement from O(data) to O(clusters)
- Rather than keeping points, keep summary statistics of groups of points
  - Cluster summaries
  - Outliers
  - Points to be clustered

BFR Algorithm

- Overview of the algorithm:
  1) Initialize K clusters/centroids
  2) Load 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

$\rightarrow$ Points are read from disk one main-memory-full at a time

$\rightarrow$ Most points from previous memory loads are summarized by simple statistics

BFR Algorithm

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
  - Take a sample; pick a random point, and then k–1 more points, each as far from the previously selected points as possible

BFR Algorithm

At each iteration keep three sets of points

- Discard set (DS)

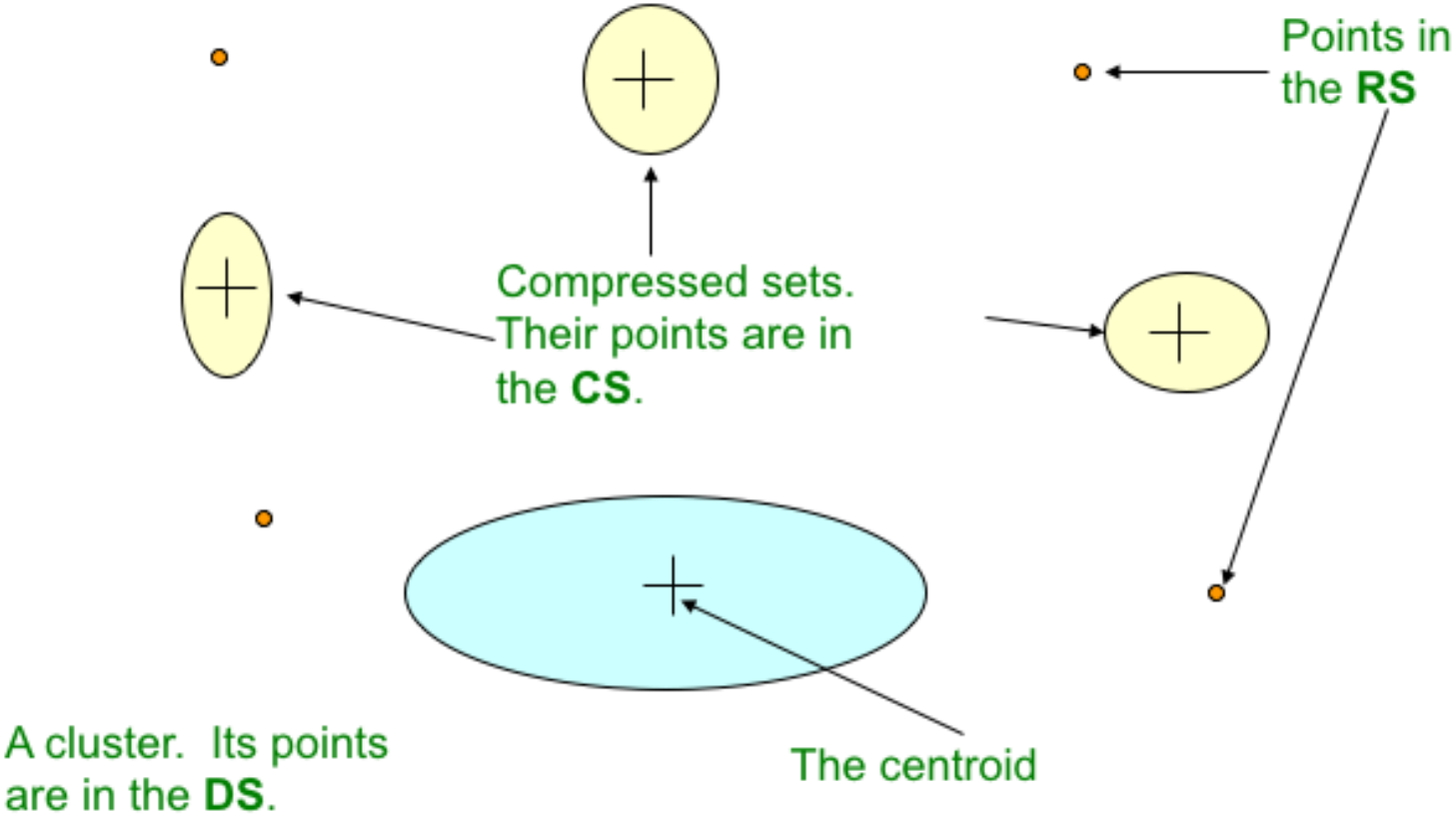
  Points close enough to a centroid to be summarized
- Compression set (CS)

  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 Algorithm



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

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

- The number of points, N
- The vector SUM

  i^{th} component is the sum of the coordinates of the points in the cluster, in the i^{th} dimension
- The vector SUMSQ

  i^{th} component is the sum of squares of the coordinates of the points in the cluster, in the i^{th} dimension

BFR Algorithm

- Uses $2d + 1$ values to represent any size cluster

  d = number of dimensions

- Average in each dimension (the centroid) can be calculated as

  SUM_i/N

- 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

BFR Algorithm

Processing loaded points

- Step 3)

  Find those points that are "sufficiently close" to a cluster centroid and add those points to that cluster and to the DS

  > These points are close to the centroid; 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

BFR Algorithm

Processing loaded points

- Step 5)

  DS set: Adjust statistics of the clusters to account for the new points

  Simply done by adding Ns, SUMs, SUMSQs

  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

  Note: all or some of these can remain as outliers

BFR Algorithm

Q1) How do we decide if a point is "close enough" to a cluster?

Q2) How do we decide whether two compressed sets (CS) should be combined into one?

BFR Algorithm

Q1) We need a way to decide whether to put a new point into a cluster (and discard)

BFR suggests

- The Mahalanobis distance is less than a threshold
- High likelihood of the point belonging to nearest centroid

Mahalanobis Distance

Mahalanobis Distance:
Normalized Euclidean distance from centroid

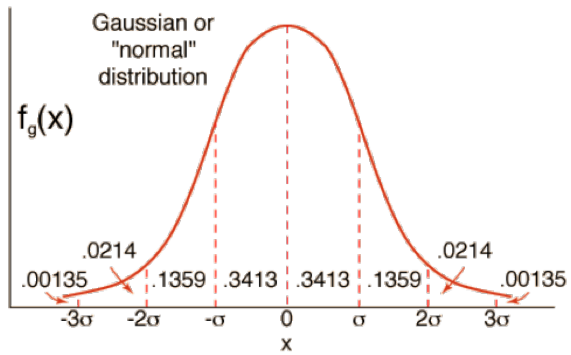
For point $(x_1, ..., x_d)$ and cluster C with centroid $(c_1, ..., c_d)$ and standard deviations $(\sigma_1, ..., \sigma_d)$

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

Normalized distance in dimension i: $y_i = \frac{x_i - c_i}{\sigma_i}$
If point is at distance σ_i in dimension i, then $y_i = 1$

BFR Algorithm

- Consider point P is one standard deviation away from centroid in each dimension; then M.D. of P is $\sqrt{d}$
- 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



BFR Algorithm

Q2) Should two CS subclusters be combined?

- Compute the variance of the combined subcluster

  N, SUM, and SUMSQ allow quick calculation
- Combine if the combined variance is below some threshold

- Many alternatives: Treat dimensions differently, consider density

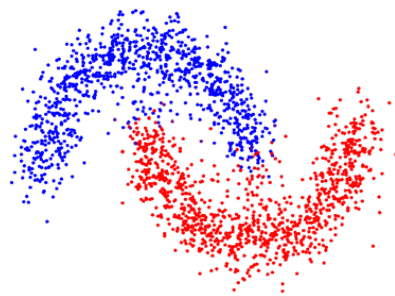
CURE algorithm

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

CURE (Clustering Using REpresentatives):

- Assumes a Euclidean distance
- Clusters can assume any shape
- Clusters represented by a collection of representative points



CURE algorithm

2 Pass algorithm: **Pass 1**

1) Initial clusters

   Pick a random sample of points that fit in main memory

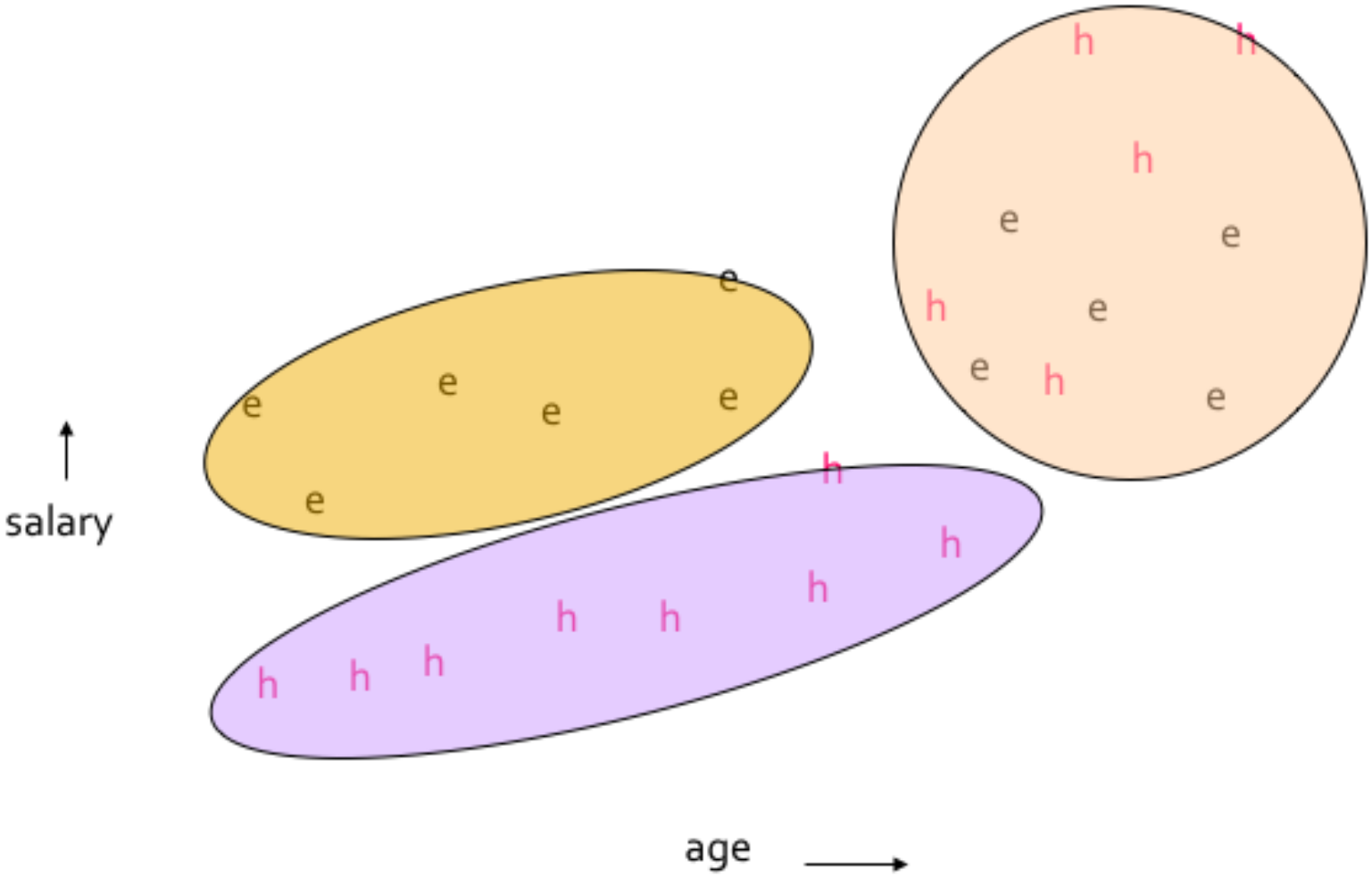
   Cluster starting points hierarchically

2) Pick representative points

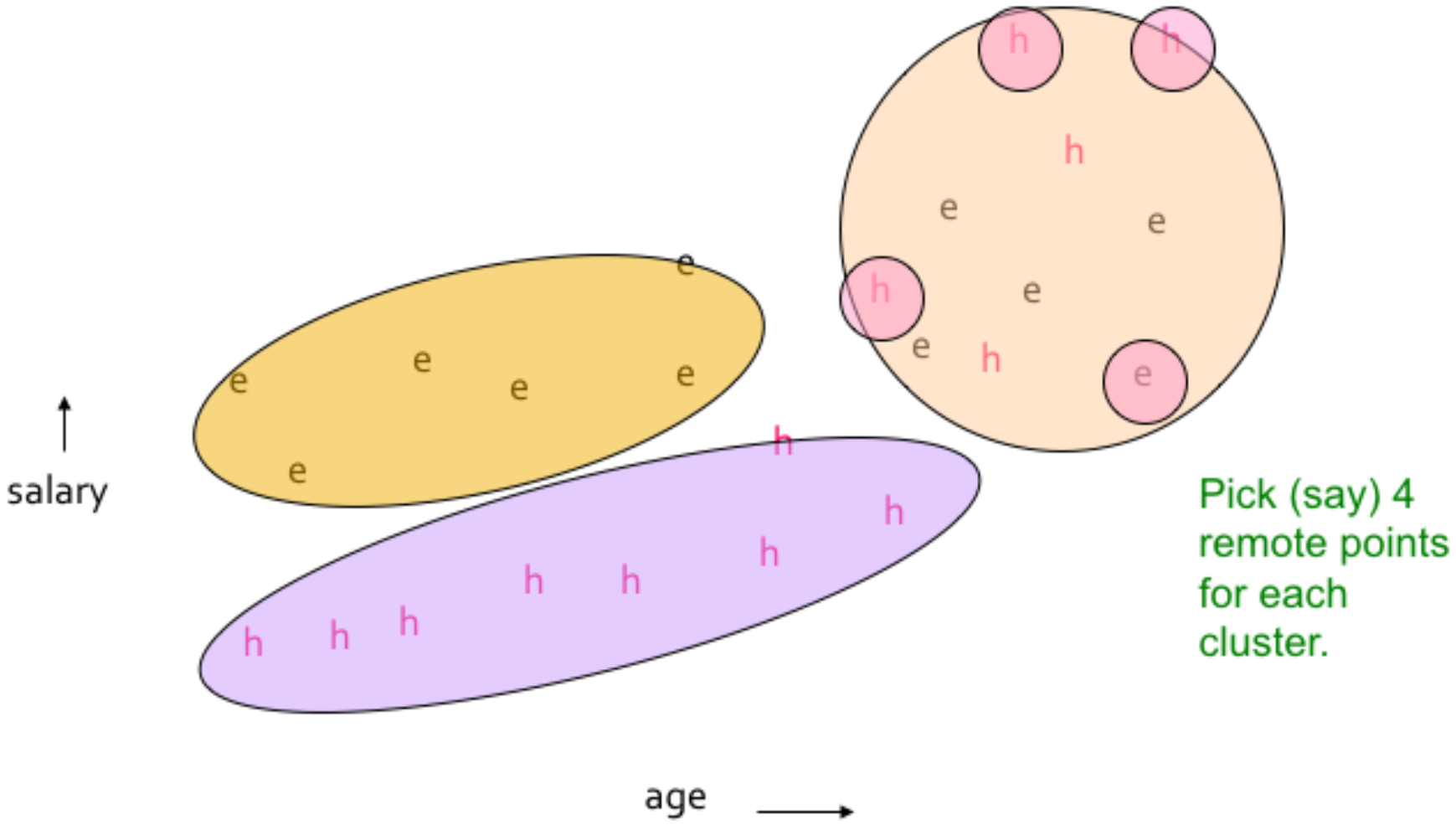
   For each cluster, pick a sample of points, as dispersed as possible

   From the sample, pick representatives by moving the points towards the centroid of the cluster (for example 20%)

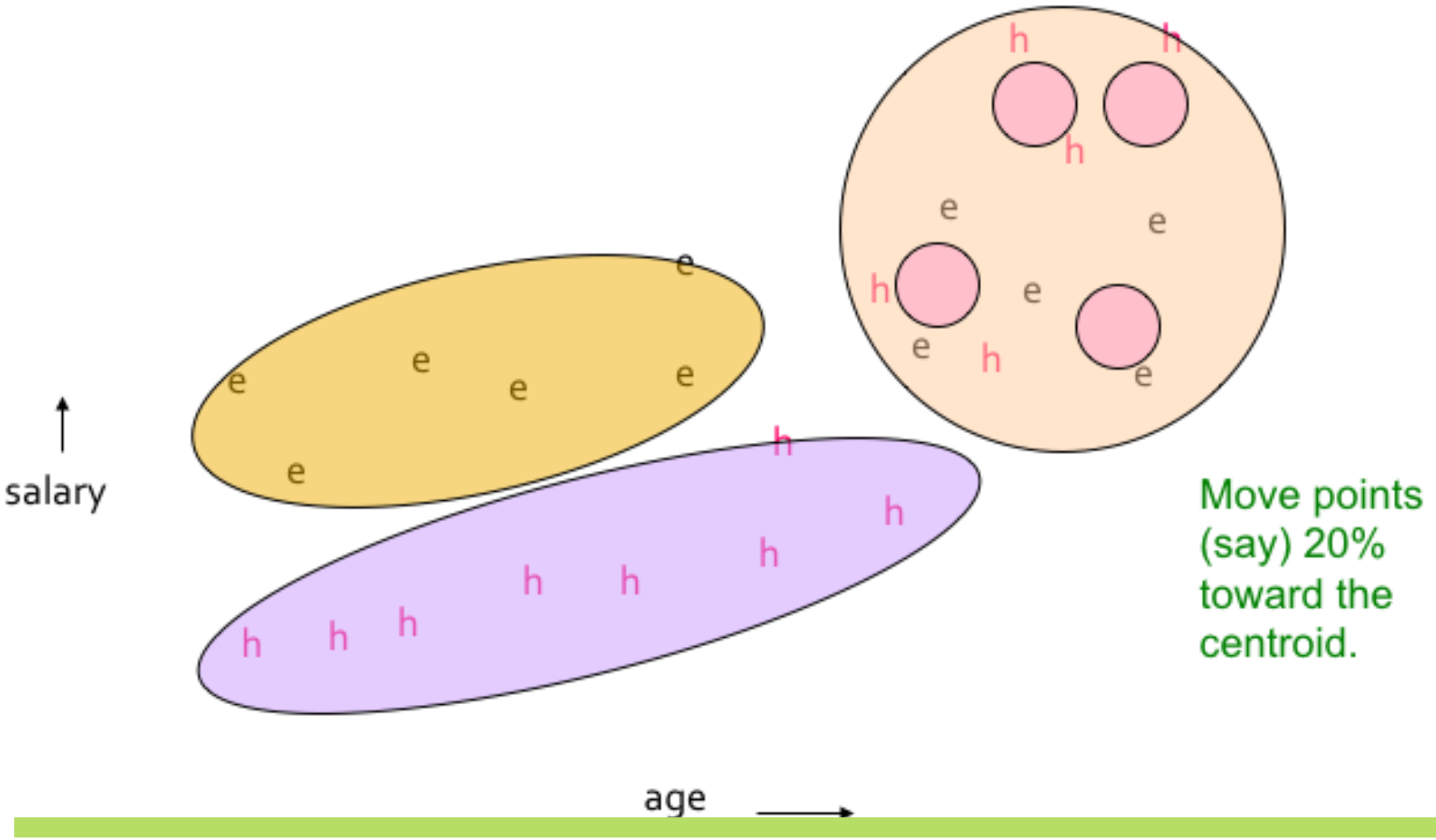
CURE algorithm



CURE algorithm



CURE algorithm



CURE algorithm

2 Pass algorithm: **Pass 2**

- Rescan the whole dataset and for each point p in the dataset
- Place it in the "closest cluster"

  Normal definition of "closest":

  Find the closest representative to p and assign it to representative's cluster

CURE algorithm

- Why the 20% Move Inward?
  - 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

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