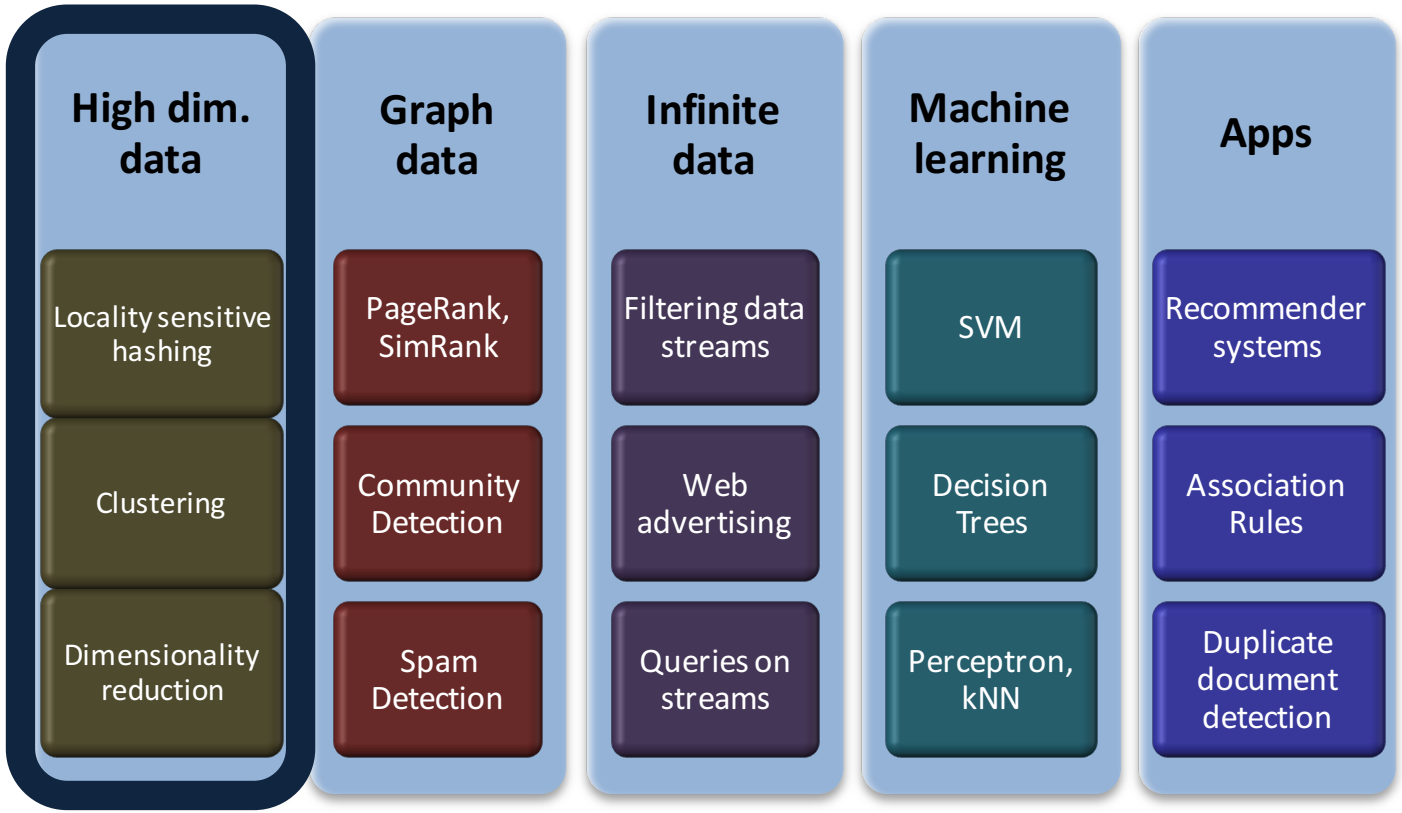
Clustering

Genoveva Vargas-Solar
http://www.vargas-solar.com/big-data-analytics
French Council of Scientific Research, LIG & LAFMIA Labs

Montevideo, 22nd November – 4th December, 2015

High Dimensional Data



High Dimensional Data

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

Clustering

Discover a set of classes given a data collection

Approaches

- As a branch of statistics, clustering analysis extensively studied focused on distance-based clustering analysis
  - AutoClass with Bayesian networks
    - P. Cheeseman, J. Stutz, Bayesian classification (AutoClass): Theory and results, In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, eds. Advances in Knowledge Discovery and Data mining, AAAI/MIT Press, 1996
  - Assume that all data point are given in advance and can be scanned frequently
- In machine learning clustering analysis
- Clustering analysis

Approaches

- As a branch of statistics
- In machine learning clustering analysis
  - Refers to unsupervised learning
    - Classes to which an object belongs to are not pre-specified
  - Conceptual clustering
    - Distance measurement may not be based on geometric distance but on that a group of objects represents a certain conceptual class
    - One needs to define a similarity between the objects and then apply it to determine the classes
    - Classes are collections of objects low interclass similarity and high intra class similarity
- Clustering analysis

Approaches

- As a branch of statistics
- In machine learning clustering analysis
- Clustering analysis
  - Probability analysis
    - Assumption that probability distributions on separate attributes are statistically independent one another (not always true)
    - The probability distribution representation of clusters → expensive clusters' updates and storage
  - Probability-based tree built to identify clusters is not height balanced
    - Increase of time and space complexity

D. Fisher, Improving inference through conceptual clustering, *In Proceedings of the AAAI Conference*, July, 1987

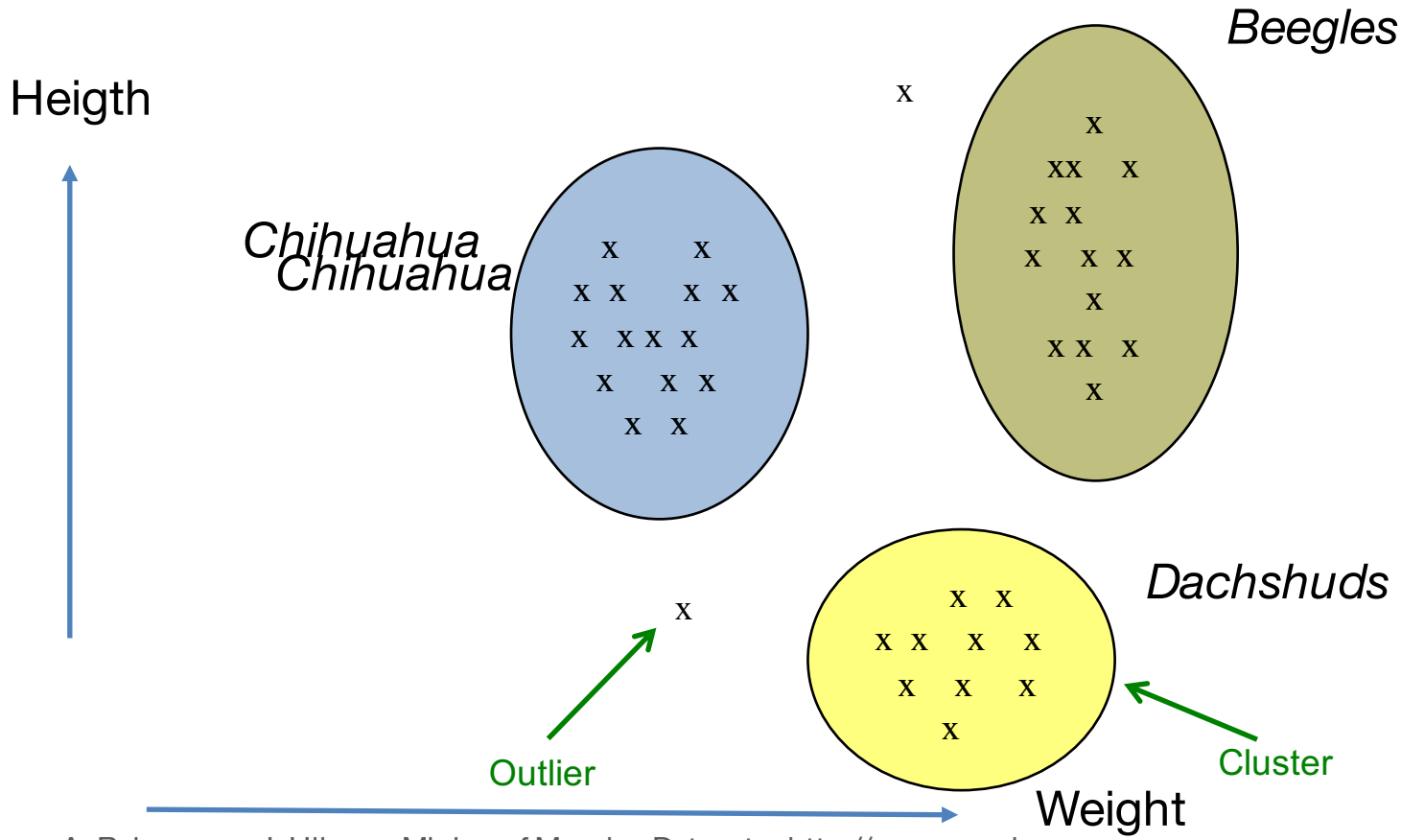
D. Fisher, Optimization and simplification of hierarchical clusterings, *In Proceedings of the 1st Conference on Knowledge Discovery and Data mining*, August, 1985

Clustering analysis

The Problem of Clustering

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

Example: Clusters & Outliers



Points and spaces

- A dataset suitable for clustering is a collection of points, which are objects belonging to some space.
- In its most general sense, a space is just a universal set of points, from which the points in the dataset are drawn.
- However, we should be mindful of the common case of a Euclidean space
  - Points are vectors of real numbers.
  - The length of the vector is the number of dimensions of the space.
  - The components of the vector are commonly called **coordinates** of the represented points
- All spaces for which we can perform a clustering **have a distance measure**, giving a distance between any two points in the space

Measuring distance

Distance measure

Suppose we have a set of points, called a space.

- A distance **measure** on this space is a **function** d(x, y) that
  - takes two points in the space as arguments and
  - produces a real number

- Satisfies the following axioms:
  1. d(x, y) ≥ 0 (no negative distances)
  2. d(x,y) = 0 if and only if x = y (distances are positive, except for the distance from a point to itself)
  3. d(x, y) = d(y, x) (distance is symmetric)
  4. d(x, y) ≤ d(x, z) + d(z, y) (the triangle inequality)

Axion 4: intuitively say that to travel from x to y, we cannot obtain any benefit if we are forced to travel via some particular third point z. It makes all distance measures behave as if distance describes the length of a shortest path from one point to another

Euclidean distance

- An **n-dimensional Euclidean space** is one where points are vectors of **n** real numbers.
- The conventional distance measure in this space, which we shall refer to as **the L2-norm**, is defined:

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Euclidean distance is a d-measure

- The Euclidean distance between two points cannot be negative (axiom 1)
- The positive square root is intended (axiom 2)
  - all squares of real numbers are nonnegative, any i such that $x_i /= y_i$ forces the distance to be strictly positive
  - if $x_i = y_i$ for all i, then the distance is clearly 0.
- Symmetry follows because $(xi - yi)^2 = (yi - xi)^2$ (axiom 3)
- The triangle inequality requires a good deal of algebra to verify
  - It is well understood to be a property of Euclidean space
  - The sum of the lengths of any two sides of a triangle is no less than the length of the third side

Other distance measures in Euclidean spaces

- For any constant r, the **Lr-norm** is the distance measure **d** defined by:

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}$$

- The case r = 2 is the usual L2-norm

L1-norm – Manhattan distance

- The distance between two points is the
  - sum of the magnitudes of the differences in each dimension`
- *Manhattan distance*
  - The distance one would have to travel between points if one were constrained to travel along grid lines
  - As on the streets of a city such as Manhattan

L_∞ - norm distance

- The limit as r approaches infinity of the Lr-norm.
- As r gets larger, only the dimension with the largest difference matters
- **L∞-norm** is defined as the maximum of $|x_i - y_i|$ over all dimensions i

L_x - examples

- Consider the two-dimensional Euclidean space and the points (2,7) and (6,4)
- L_2-norm
- L1-norm
- L∞-norm

Jaccard distance

- The **Jaccard similarity** is a measure of how close sets are, although it is not really a distance measure
  - the closer sets are, the higher the Jaccard similarity
- **Jaccard distance:** *1 minus the Jaccard similarity* is a distance measure

$$d(x, y) = 1 - \text{SIM}(x, y).$$

- where SIM(x, y) is the probability a random minhash function maps x and y to the same value
- *1 minus the ratio of the sizes of the intersection and union of sets x and y*

Cosine distance

- **Cosine distance** between two points is the angle that the vectors to those points make.
  - The angle will be in the range 0 to 180 degrees, regardless of how many dimensions the space has
  - First compute the cosine of the angle, and then
  - Apply the arc-cosine function to translate to an angle in the 0-180 degree range.
- Given two vectors x and y, the cosine of the angle between them is the
  - **dot product** x.y

$$[x_1, x_2, \ldots, x_n].[y_1, y_2, \ldots, y_n] \text{ is } \sum_{i=1}^{n} x_i y_i.$$

  - **divided** by the **L2-norms of x and y** (i.e., their Euclidean distances from the origin)

Cosine distance example

- Let our two vectors be $x = [1, 2, -1]$ and $= [2, 1, 1]$.
- The dot product x.y $1 \times 2 + 2 \times 1 + (-1) \times 1 = 3.$
- L_2-norm of x $\sqrt{1^2 + 2^2 + (-1)^2} = \sqrt{6}.$
- The cosine of the angle between x and y

$$3/(\sqrt{6}\sqrt{6}) \text{ or } 1/2$$

- The angle whose cosine is 1/2 is 60 degrees, so that is the cosine distance between x and y.

Cosine distance: final remarks

Makes sense in spaces that have dimensions, including

- Euclidean spaces
- Discrete versions of Euclidean spaces,
  - spaces where points are vectors with integer components or Boolean (0 or 1) components
  - points may be thought of as directions.
  - There is no distinction between a vector and a multiple of that vector

Edit distance (1)

- The distance between two strings $x = x_1x_2 \cdots x_n$ and $y = y_1y_2 \cdots y_m$ is the **smallest number of insertions and deletions** of **single characters** that will **convert x to y**
- The edit distance between the strings `x = abcde` and `y = acfdeg` is 3. To convert x to y:
  - Delete b
  - Insert f after c
  - Insert g after e
- No sequence of fewer than three insertions and/or deletions will convert x to y. Thus, **`d(x, y) = 3`**

Edit distance (2)

- `edit distance d(x, y):` can be calculated as the length of x plus the length of y minus twice the length of their LCS
- Compute a *longest common subsequence* (LCS) of x and y
  - a string that is constructed by deleting positions from x and y, and
  - that is as long as any string that can be constructed that way

Edit distance example

- The strings `x = abcde` and `y = acfdeg`
  - Have a unique LCS, which is **acde**
  - We can be sure it is the longest possible, because it contains every symbol appearing in both x and y
- These common symbols appear in the same order in both strings, so we are able to use them all in an LCS
- Given the length of `x` is `5`, the length of `y` is `6`, and the length of their `LCS` is `4`
- The `edit distance length(X)+length(y) - 2(LCS):` $5+6-2\times4 = 3$

Hamming distance

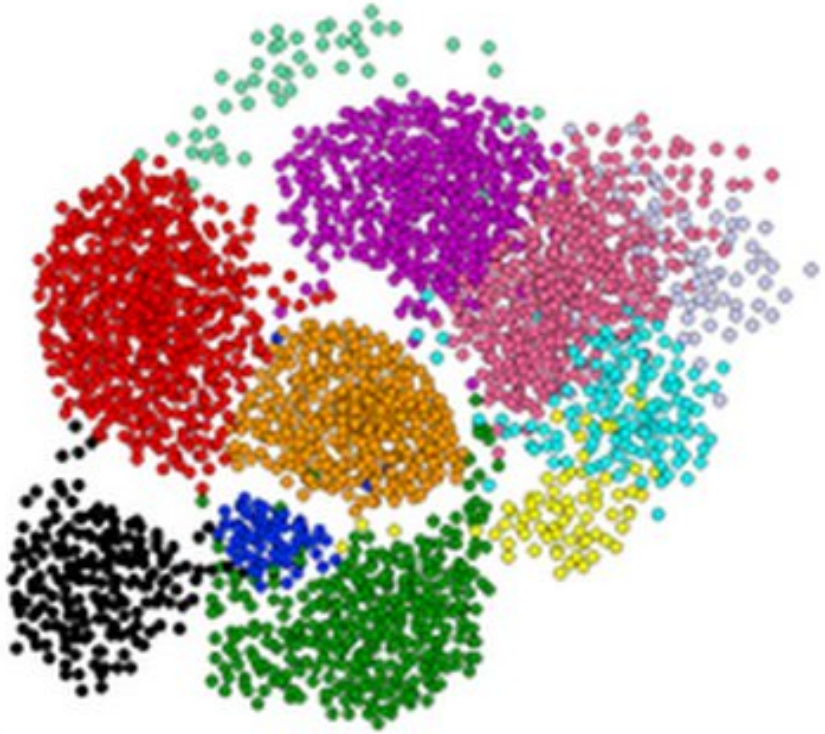
- Given a space of vectors
- `Hamming distance` between two vectors is
  - the number of components in which they differ
- Most commonly, Hamming distance is used when the vectors are Boolean
- The `Hamming distance` between the vectors `10101` and `11110` is `3`.
  - vectors differ in the second, fourth, and fifth components, while they agree in the first and third components.

Non Euclidean space

- Euclidean space property: the **average of points** in a Euclidean space **always exists** and is a **point in the space**.
- There are spaces where the notion of average makes no sense (e.g., average of strings)
- Vector spaces, may or may not be Euclidean
  - If the components of the vectors can be any real numbers, then the space is **Euclidean**
  - if we restrict components to be integers, then the space is **not Euclidean** (cannot find an average of the vectors [1, 2] and [3, 1])

A hard problem by examples

Clustering is a hard problem!



Why is it hard?

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- And in most cases, looks are not deceiving

- Many applications involve not 2, but 10 or 10,000 dimensions
- **High-dimensional spaces look different:** Almost all pairs of points are at about the same distance

Clustering Problem: Galaxies

- **A catalog of 2 billion "sky objects" represents objects by their radiation in 7 dimensions (frequency bands)**
- **Problem:** Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.
- Sloan Digital Sky Survey

Clustering Problem: Music CDs

- **Intuitively: Music divided into categories, and customers prefer a few categories**
  - But what are categories really?
- Represent a CD by a set of customers who bought it:
  - Similar CDs have similar sets of customers, and vice-versa

Clustering Problem: Music CDs

Space of all CDs:

- Think of a space with one dim. for each customer
  - Values in a dimension may be 0 or 1 only
  - A CD is a point in this space $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the ith customer bought the CD
- For Amazon, the dimension is tens of millions
- **Task:** Find clusters of similar CDs

Clustering Problem: Documents

Finding topics:

- Represent a document by a vector $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the ith word (in some order) appears in the document
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words
- **Documents with similar sets of words may be about the same topic**

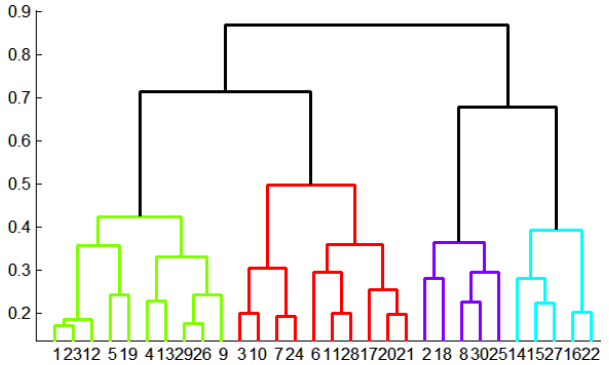
Cosine, Jaccard, and Euclidean

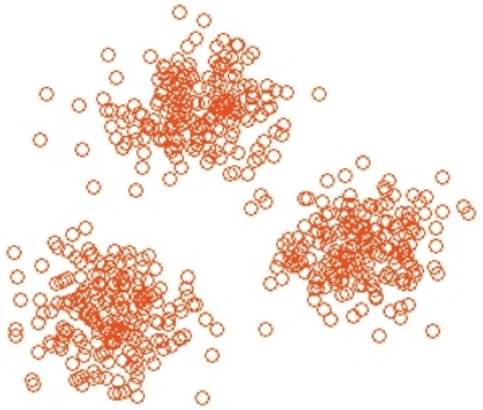
- **As with CDs we have a choice when we think of documents as sets of words or shingles:**
  - **Sets as vectors:** Measure similarity by the **cosine distance**
  - **Sets as sets:** Measure similarity by the **Jaccard distance**
  - **Sets as points:** Measure similarity by **Euclidean distance**

Clustering methods

Overview (1)

- **Agglomerative** (bottom up):
  - Initially, each point is a cluster
  - Repeatedly combine the two "nearest" clusters into one
  - **Divisive** (top down):
    - Start with one cluster and recursively split it
- **Point assignment:**
  - Maintain a set of clusters
  - Points belong to "nearest" cluster





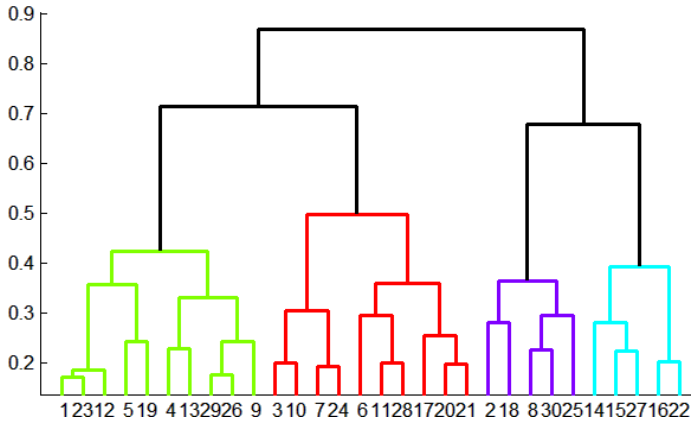
Overview (2)

- Euclidean space, or whether with arbitrary distance measure.
  - In a Euclidean space it is possible to summarize a collection of points by their centroid – the average of the points
  - In a non-Euclidean space, there is no notion of a centroid, and we are forced to develop another way to summarize clusters
- Data is small enough to fit in main memory, or data must reside in secondary memory
  - Algorithms for large amounts of data often must take shortcuts, since it is infeasible to look at all pairs of points, for example
  - It is also necessary to summarize clusters in main memory, since we cannot hold all the points of all the clusters in main memory at the same time

Hierarchical clustering

Hierarchical Clustering

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



- **Three important questions:**

  1) How do you represent a cluster of more than one point?

  2) How do you determine the "nearness" of clusters?

  3) When to stop combining clusters?

Hierarchical Clustering

Key operation: Repeatedly combine two nearest clusters

```
WHILE it is not time to stop DO
    pick the best two clusters to merge;
    combine those two clusters into one cluster;
END;
```

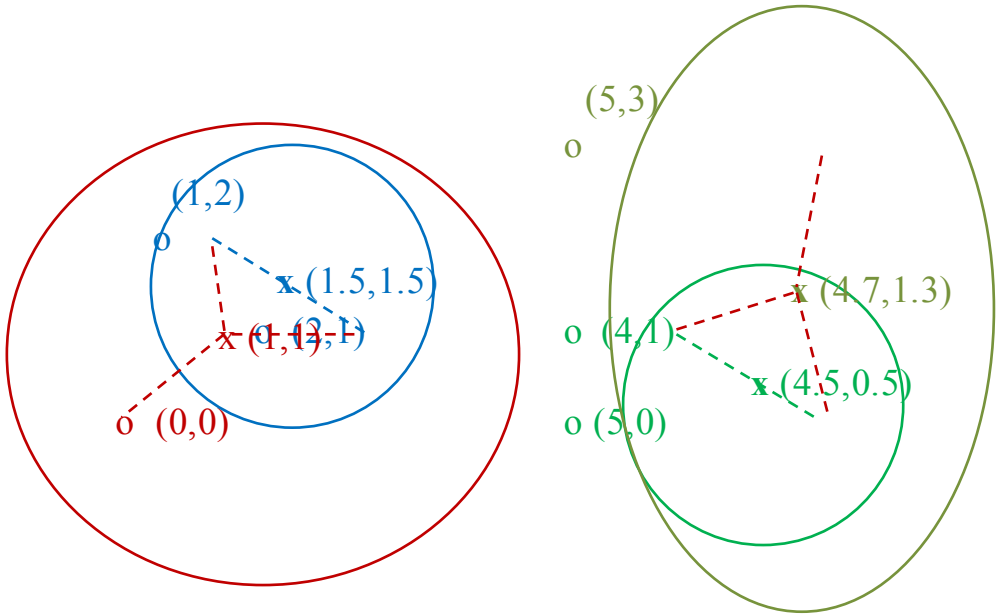
(1) How to represent a cluster of many points?

- **Key problem:** As you merge clusters, how do you represent the "location" of each cluster, to tell which pair of clusters is closest?
- **Euclidean case:** each cluster has a ***centroid*** = average of its (data)points

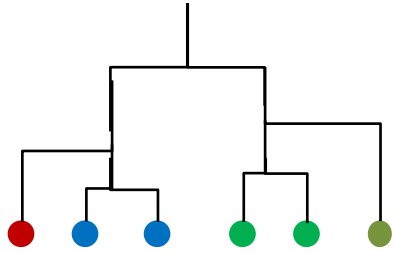
(2) How to determine "nearness" of clusters?

- Measure cluster distances by distances of centroids

Example: Hierarchical clustering



Data:
o … data point
x … centroid



Dendrogram

Stopping strategies

- Knowledge or belief, about how many clusters there are in the data
  - For example, if we are told that the data about dogs is taken from Chihuahuas, Dachshunds, and Beagles
  - Then we know to stop when there are three clusters left
- Stop combining when at some point the best combination of existing clusters produces a cluster that is "*inadequate*"
- Continue clustering until there is only one cluster.
  - meaningless to return a single cluster consisting of all the points.
  - Rather, return the tree representing the way in which all the points were combined.
  - Good sense in applications, in which the points are genomes of different species, and the distance measure reflects the difference in the genome.
  - The tree represents the evolution of these species, that is, the likely order in which two species branched from a common ancestor

And in the Non-Euclidean Case?

What about the Non-Euclidean case?

- The only "locations" we can talk about are the points themselves
  - i.e., there is no "average" of two points
  - Distances cannot be based on "location" of points

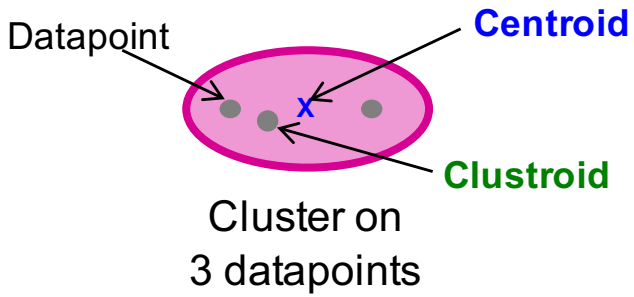
- **Approach 1:**

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

  (2) How do you determine the "nearness" of clusters? Treat clustroid as if it were centroid, when computing inter-cluster distances

"Closest" Point?

- **(1) How to represent a cluster of many points?**
  clustroid = point "***closest***" to other points
- **Possible meanings of "closest":**
  - Smallest maximum distance to other points
  - Smallest average distance to other points
  - Smallest sum of squares of distances to other points
    - For distance metric ***d*** clustroid ***c*** of cluster ***C*** is:



$$\min_{c} \sum_{x \in C} d(x,c)^2$$

Centroid is the avg. of all (data)points in the cluster. This means centroid is an "artificial" point.
Clustroid is an **existing** (data)point that is "closest" to all other points in the cluster.

Defining "Nearness" of Clusters

(2) How do you determine the "nearness" of clusters?

- **Approach 2:**
  Intercluster distance = minimum of the distances between any two points, one from each cluster
- **Approach 3:**
  Pick a notion of "**cohesion**" of clusters, *e.g.*, maximum distance from the clustroid
  - Merge clusters whose *union* is most cohesive

Cohesion

- **Approach 3.1:** Use the **diameter** of the merged cluster = maximum distance between points in the cluster
- **Approach 3.2:** Use the **average distance** between points in the cluster
- **Approach 3.3:** Use a **density-based approach**
  - Take the diameter or avg. distance, e.g., and divide by the number of points in the cluster

Implementation

- **Naïve implementation of hierarchical clustering:**
  - At each step, compute pairwise distances between all pairs of clusters, then merge
  - $O(N^3)$
- Careful implementation using priority queue can reduce time to $O(N^2 \log N)$
  - **Still too expensive for really big datasets that do not fit in memory**

K-means clustering

```
Initially choose k points that are likely to be in
    different clusters;
Make these points the centroids of their clusters;
FOR each remaining point p DO
    find the centroid to which p is closest;
    Add p to the cluster of that centroid;
    Adjust the centroid of that cluster to account for p;
END;
```

k–means Algorithm(s)

- Assumes Euclidean space/distance
- Start by picking ***k***, the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:** Pick one point at random, then ***k*-1** other points, each as far away as possible from the previous points

Populating Clusters

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

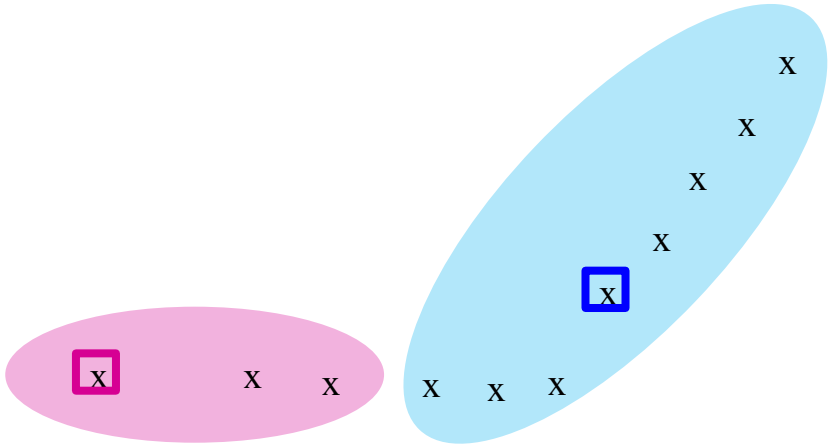
2) After all points are assigned, update the locations of centroids of the ***k*** clusters

3) Reassign all points to their closest centroid

- Sometimes moves points between clusters

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

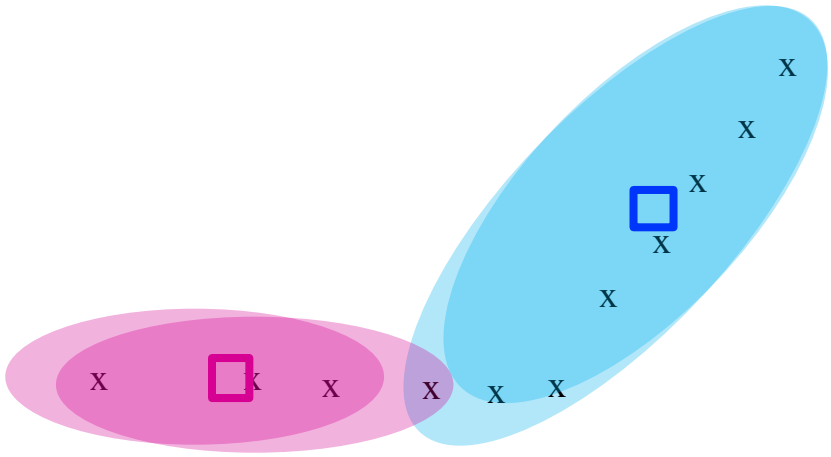
Example: Assigning Clusters



x … data point
□ … centroid

Clusters after round 1

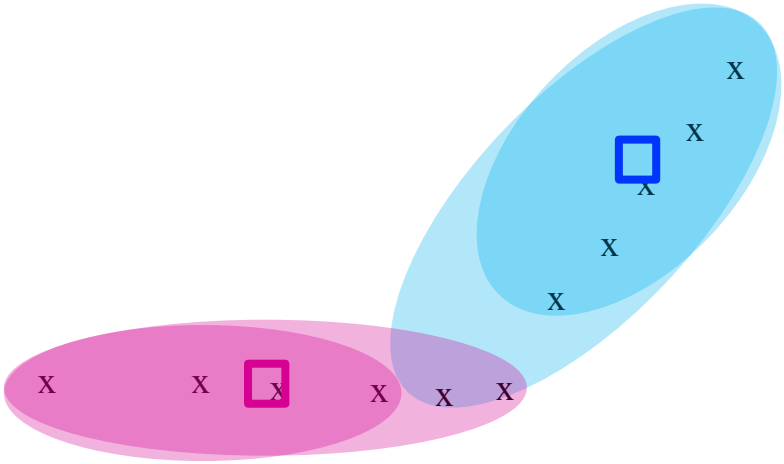
Example: Assigning Clusters



x … data point
□ … centroid

Clusters after round 2

Example: Assigning Clusters



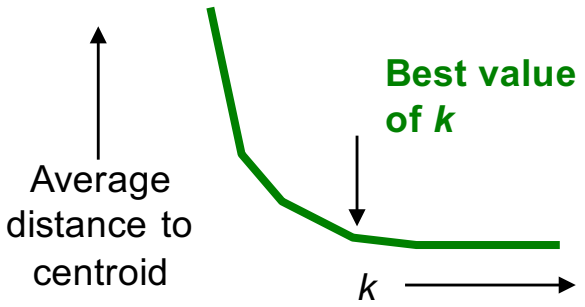
x … data point
□ … centroid

Clusters at the end

Getting the *k* right

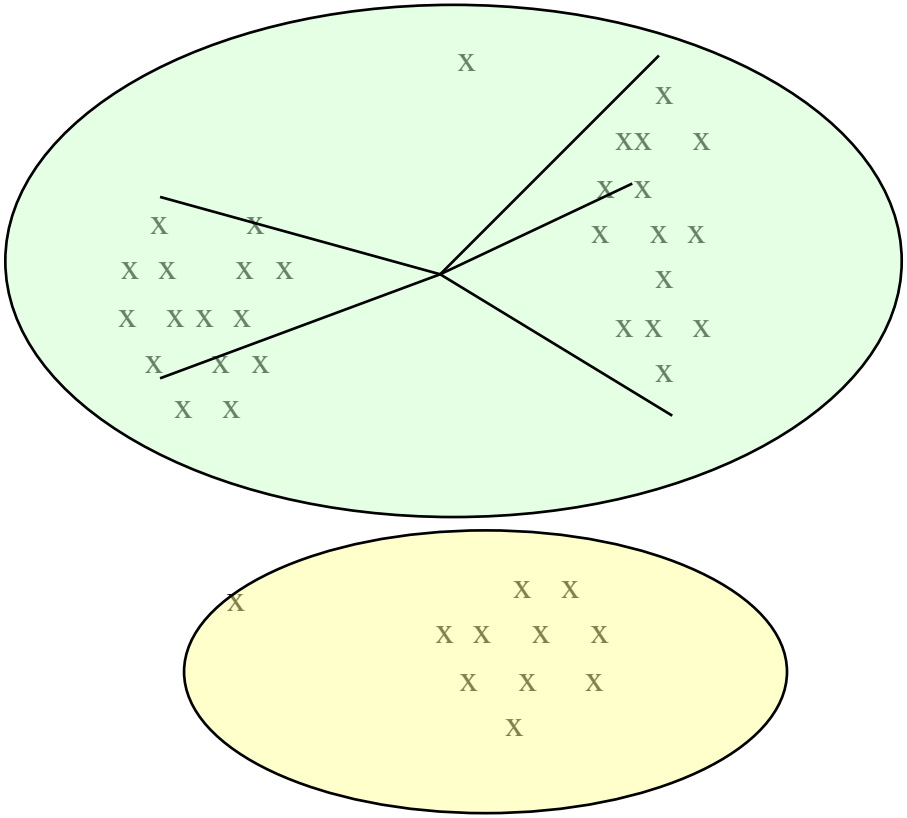
How to select *k*?

- 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



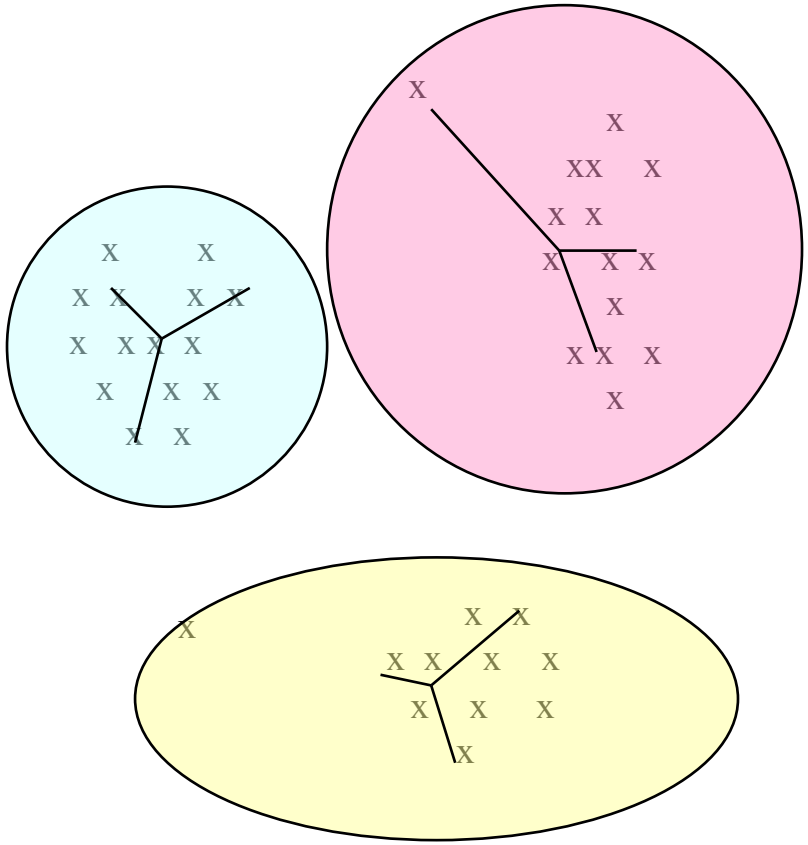
Example: Picking *k*

Too few;
many long
distances
to centroid.



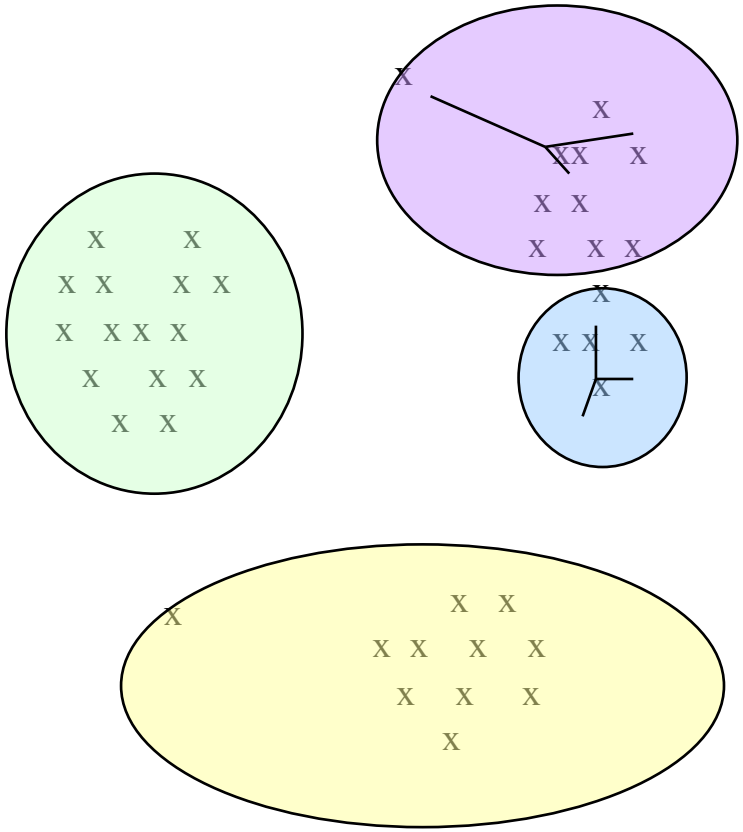
Example: Picking *k*

Just right;
distances
rather short.



Example: Picking *k*

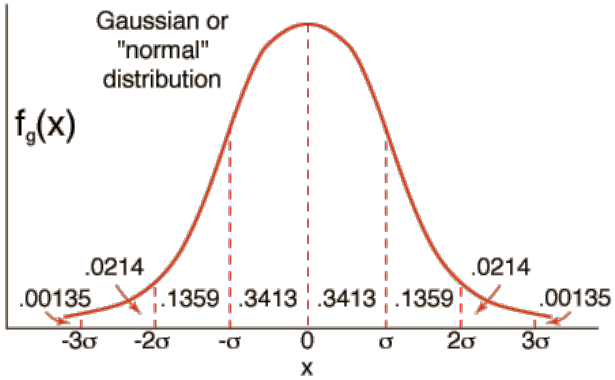
Too many;
little improvement
in average
distance.



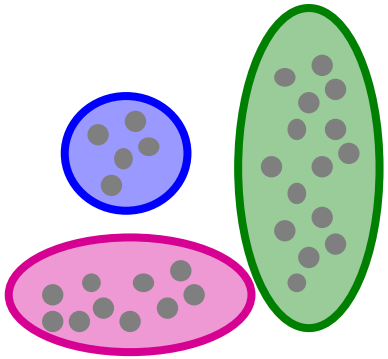
The BFR algorithm

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
- **Efficient way to summarize clusters** (want memory required O(clusters) and not O(data))



BFR Algorithm

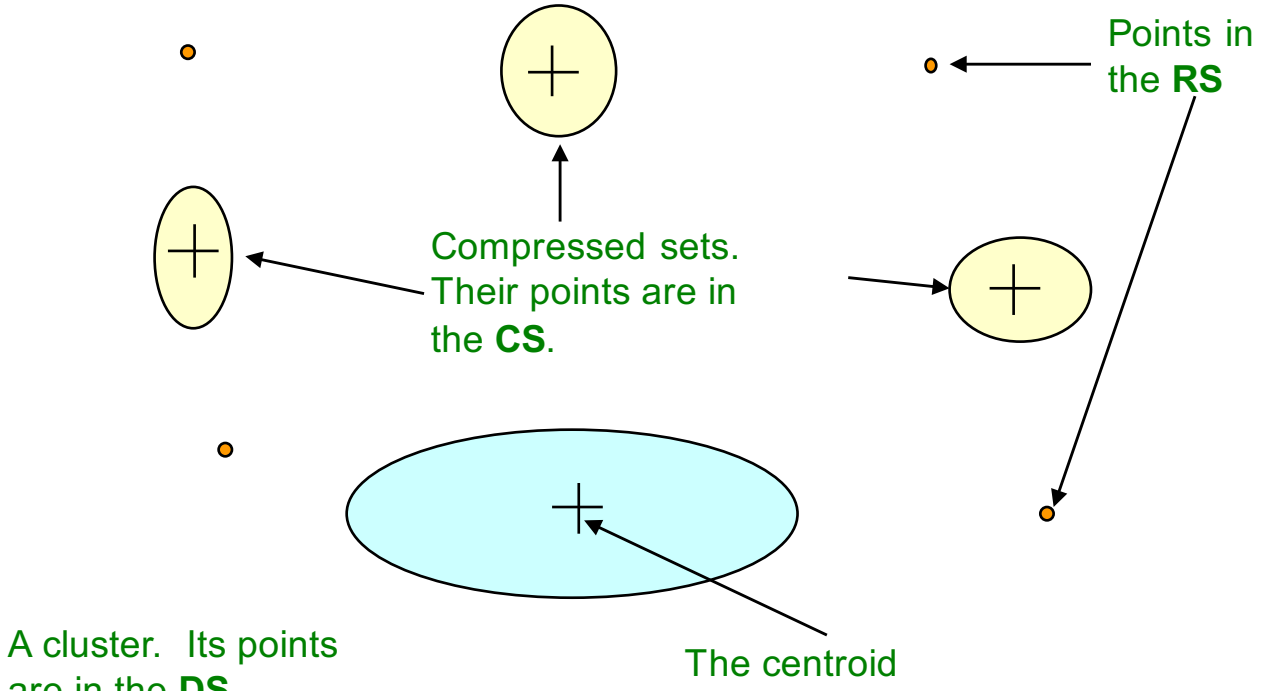
- Points are read from disk one main-memory-full at a time
- **Most points from previous memory loads are summarized by simple statistics**
- To begin, from the initial load we select the initial ***k*** centroids by some sensible approach:
  - Take ***k*** random points
  - Take a small random sample and cluster optimally
  - Take a sample; pick a random point, and then ***k–1*** more points, each as far from the previously selected points as possible

Three Classes of Points

3 sets of points which we keep track of:

- **Discard set (DS):**
  - Points close enough to a centroid to be summarized
- **Compression set (CS):**
  - Groups of points that are close together but not close to any existing centroid
  - These points are summarized, but not assigned to a cluster
- **Retained set (RS):**
  - Isolated points waiting to be assigned to a compression set

BFR: "Galaxies" Picture

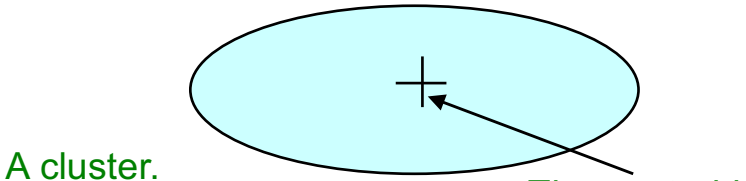


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

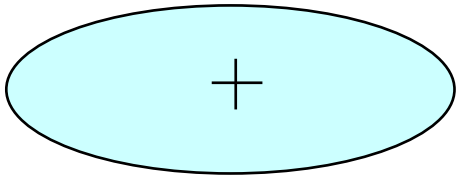
For each cluster, the discard set (DS) is summarized by:

- The number of points, ***N***
- The vector ***SUM***, whose i^{th} component is the sum of the coordinates of the points in the ith 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 $\mathbf{SUM}_i / N$
  - $\mathbf{SUM}_i$ = 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
- **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!

The "Memory-Load" of Points

Processing the "Memory-Load" of points (1):

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

2) Use any main-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

Processing the "Memory-Load" of points (2):

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

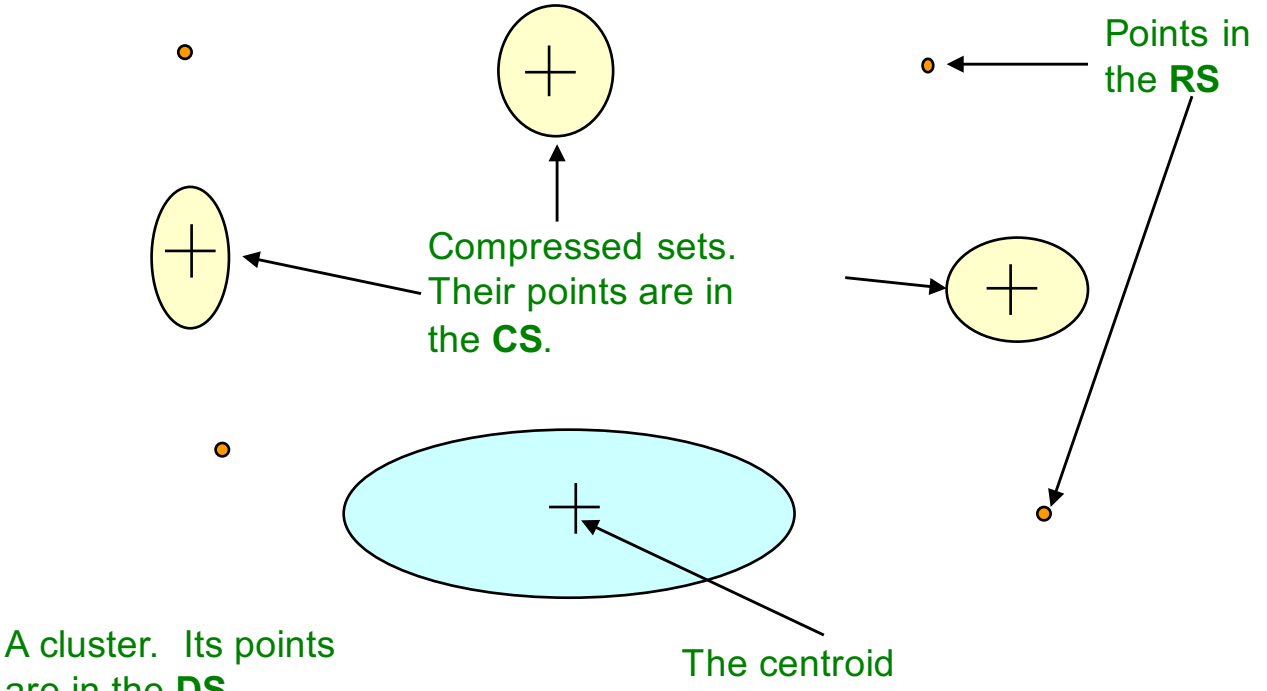
- Add ***N***s, ***SUM***s, ***SUMSQ***s

4) Consider merging compressed sets in the **CS**

5) 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

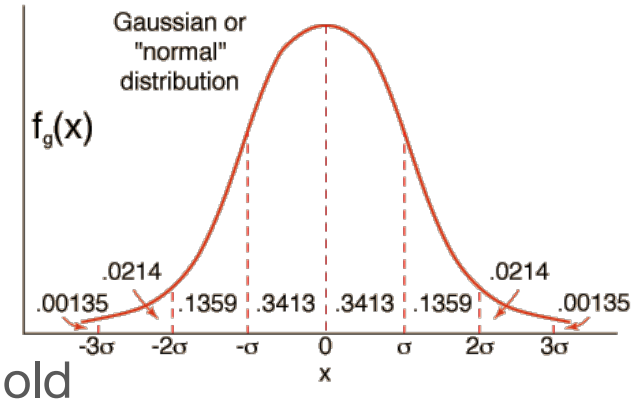
A Few Details…

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

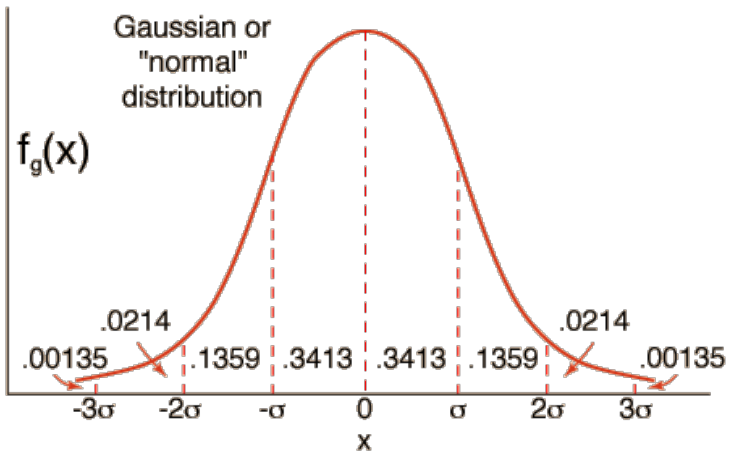
$$d(x, c) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_i - c_i}{\sigma_i}\right)^2}$$

σ_i … standard deviation of points in the cluster in the ith 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 $< \sqrt{d}$

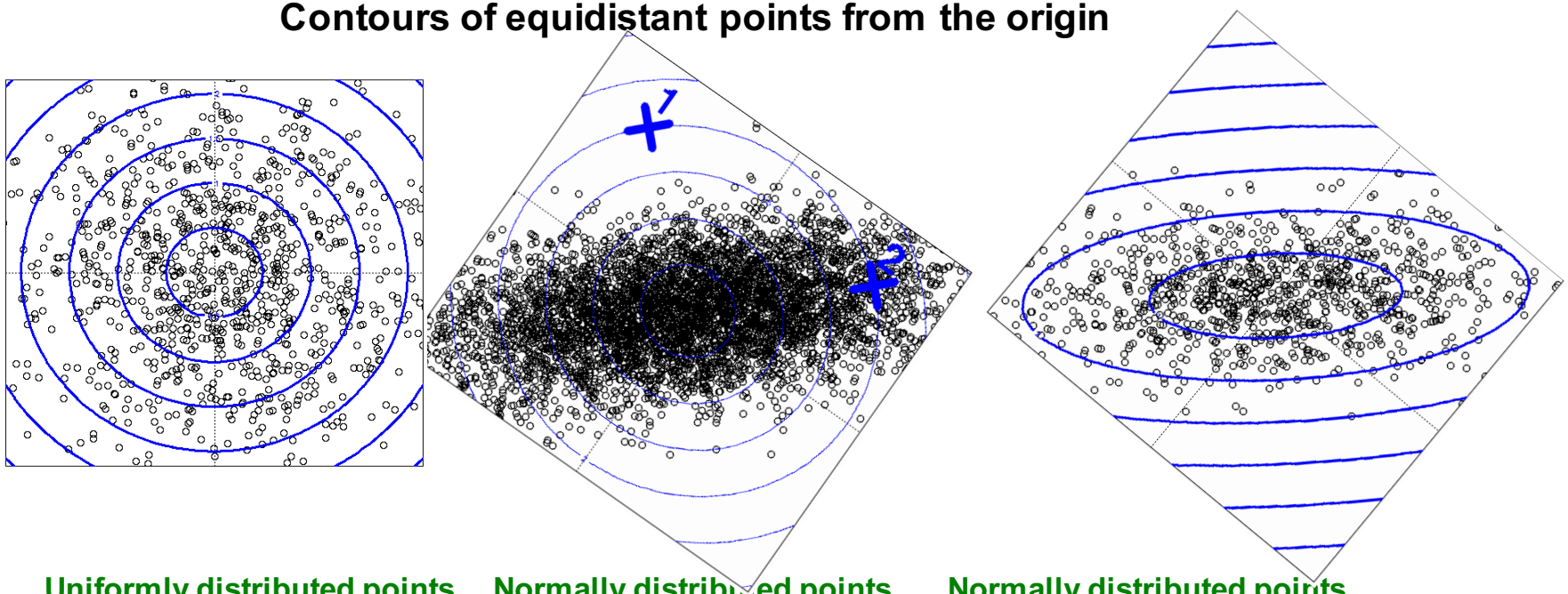
- Accept a point for a cluster if its M.D. is < some threshold, e.g. **2** standard deviations



Picture: Equal M.D. Regions

- **Euclidean vs. Mahalanobis distance**

Contours of equidistant points from the origin



Uniformly distributed points, Euclidean distance

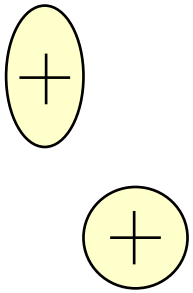
Normally distributed points, Euclidean distance

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



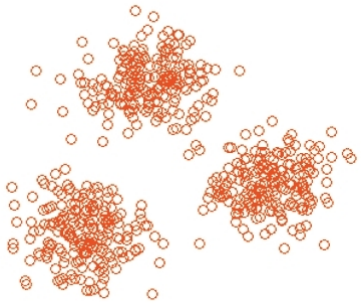
- **Many alternatives:** Treat dimensions differently, consider density

The CURE algorithm

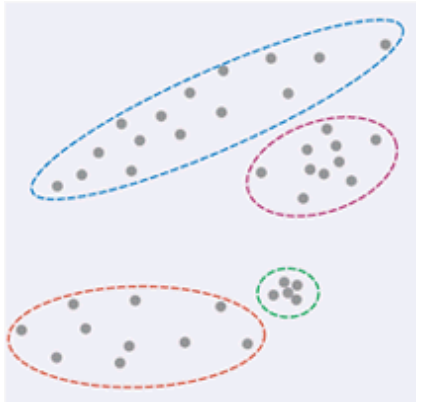
Extension of k-means to clusters of arbitrary shapes

The CURE Algorithm

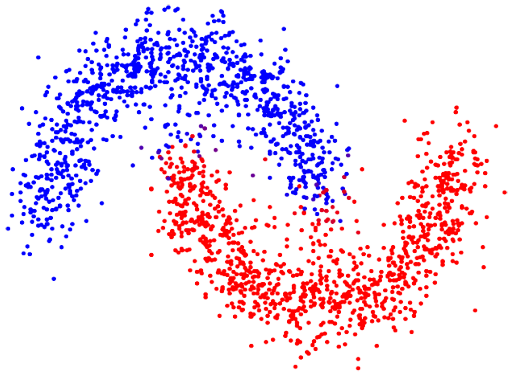
- **Problem with BFR/*k*-means:**
  - Assumes clusters are normally distributed in each dimension
  - And axes are fixed – ellipses at an angle are ***not*** ***OK***



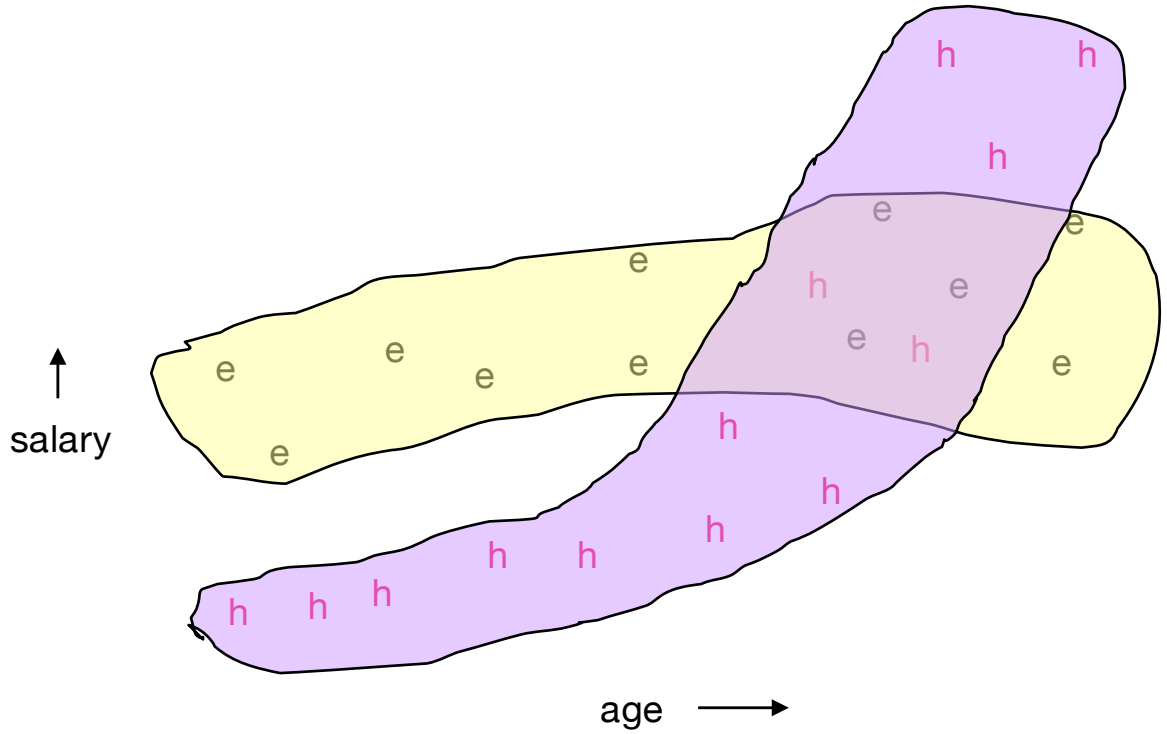
Vs.



- **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

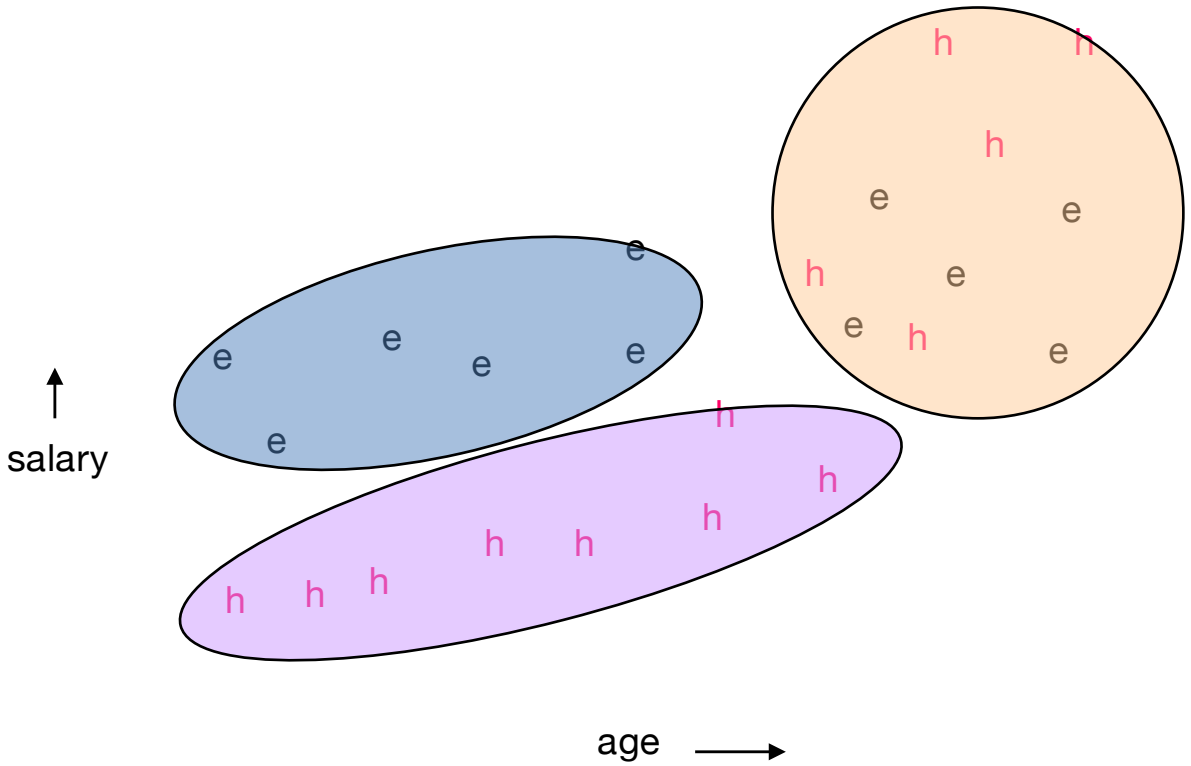
1) Initial clusters:

- Cluster these points hierarchically – group nearest points/clusters

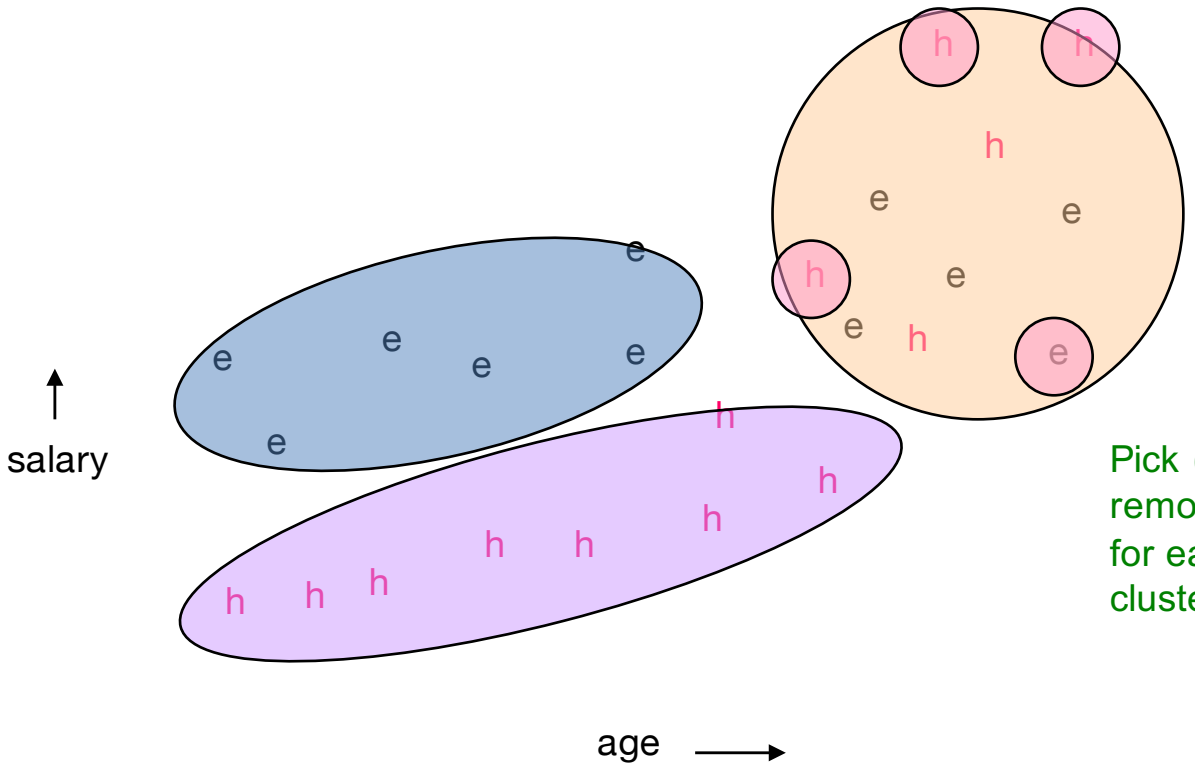
2) Pick representative points:

- 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

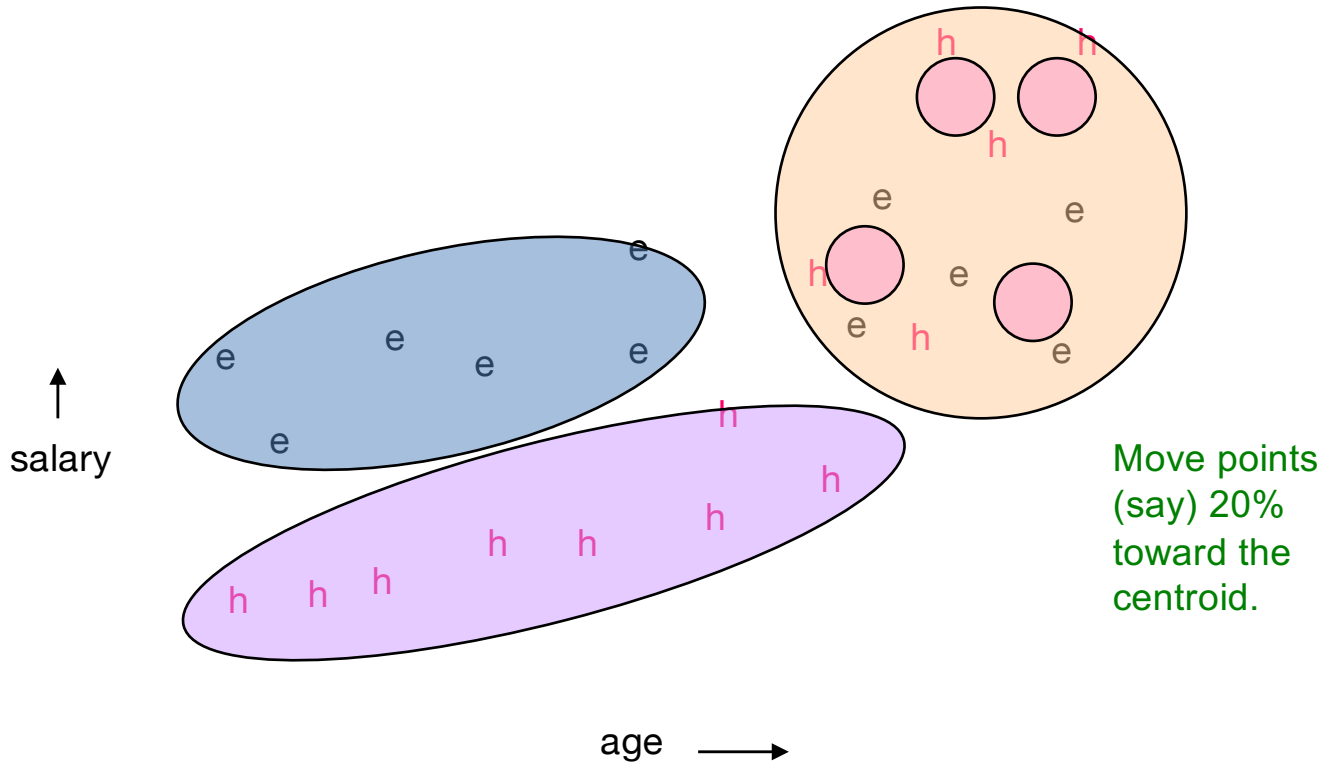


Example: Pick Dispersed Points



Pick (say) 4 remote points for each cluster.

Example: Pick Dispersed Points

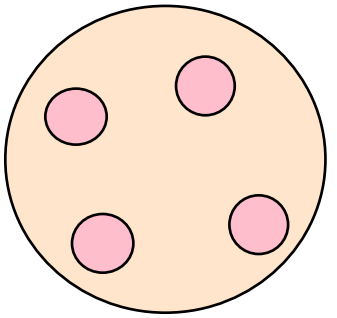


Finishing CURE

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

Final remarks

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**