Clustering

Professor Wei-Min Shen
University of Southern California

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman
Stanford University

http://www.mmds.org

High Dimensional Data

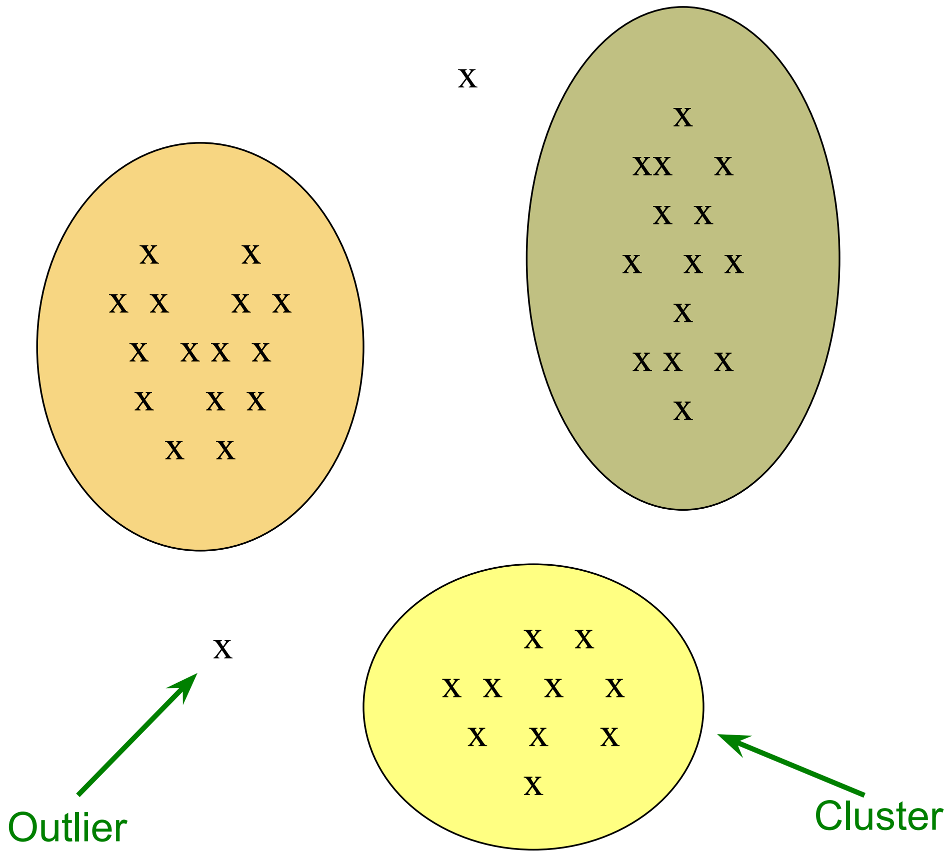
- Given a cloud of data points we want to understand its structure

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, …

⇓ So similarity becomes hard to judge.

Example: Clusters & Outliers



Clustering is a hard problem!

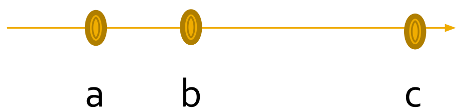
Why is it hard?

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy

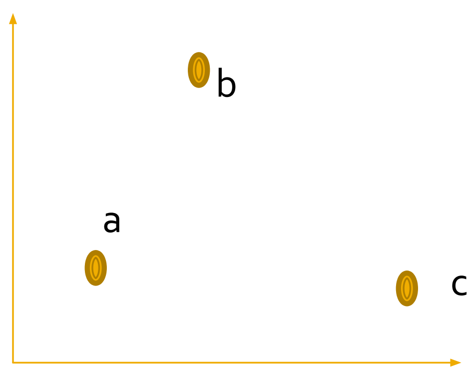
- Many applications involve not 2, but 10 or 10,000 dimensions
- **High-dimensional spaces look different:** Almost all pairs of points are at about the same distance

High Dimension: Euclidean

- Consider a set of data points on a line
  - dist(a, b) < dist(a, c)



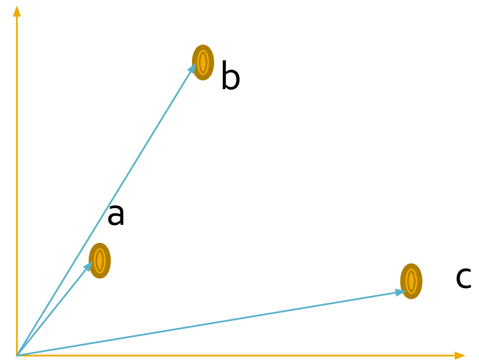
- Consider increasing the dimension by 1
  - dist(a, b) ~ dist(a, c)



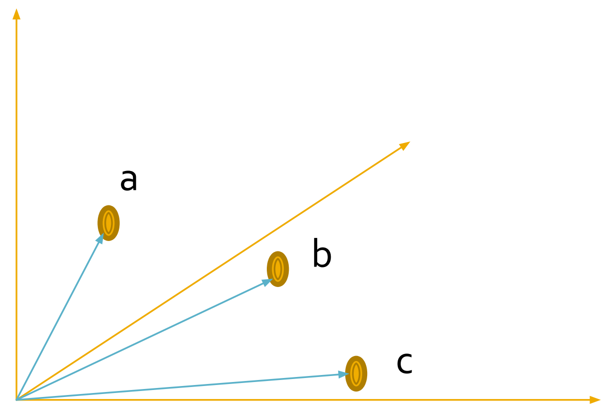
High Dimension: Cosine

- Cosine(a, b) > Cosine(a, c)

3D- vector.



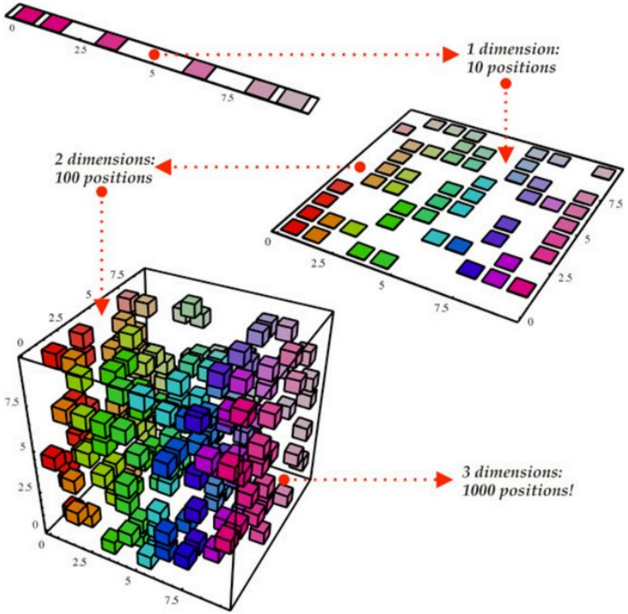
- Increase d to 3
  - Cosine(a, b) ~ Cosine(a, c)
- Higher d
  - Angle -> 90^{o}
  - Cosine -> 0



Curse of Dimensionality

- Data points have similar distance btw each other
  - Euclidean distance breaks
  - almost all pairs of points are equally far away from one another

- Data vectors become orthogonal
  - Cosine function breaks
  - almost any two vectors are orthogonal



https://bigsnarf.wordpress.com/2013/06/14/curse-of-dimensionality/

Clustering Problem: Music CDs

- **Intuitively: Music divides 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

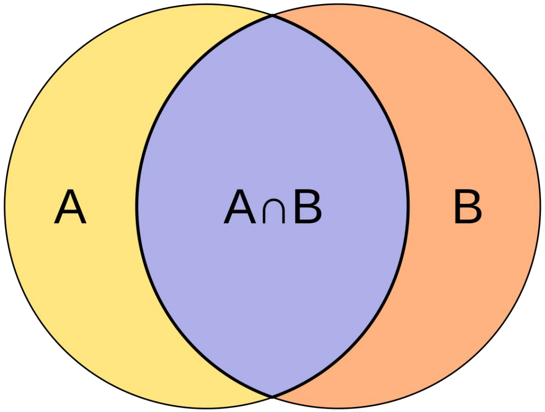
Space of all CDs:

- Think of a space with one dim. for each customer (there are many customers!)
  - 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 i^{th} customer bought the CD
- For Amazon, the dimension is tens of millions
- **Task:** Find clusters of similar CDs

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**

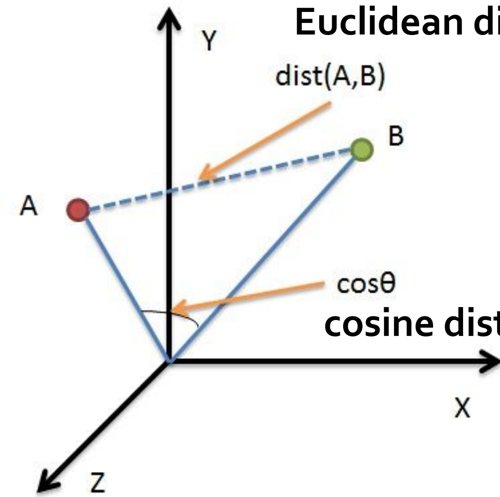
Measure similarity



Jaccard distance

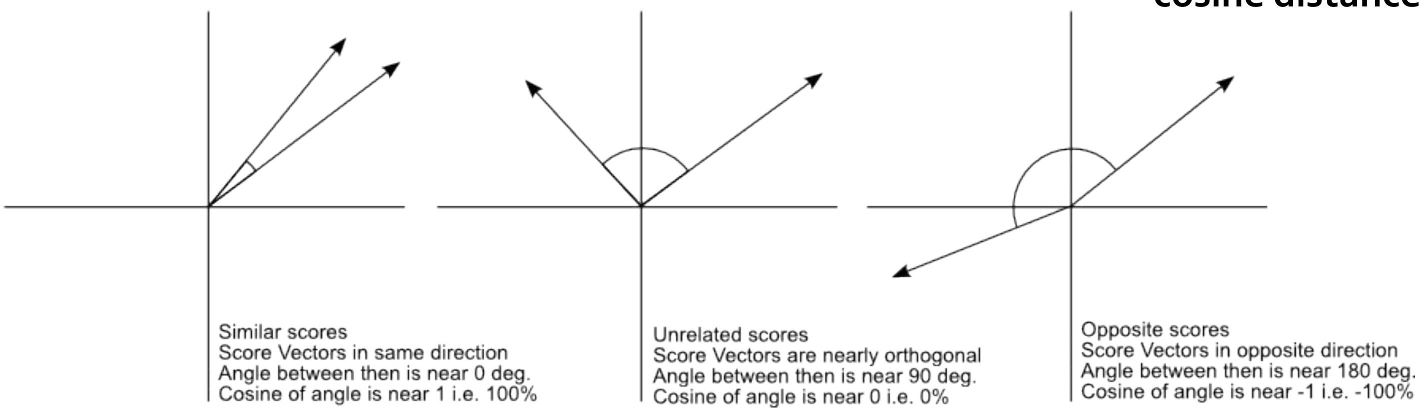
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Euclidean distance



cosine distance

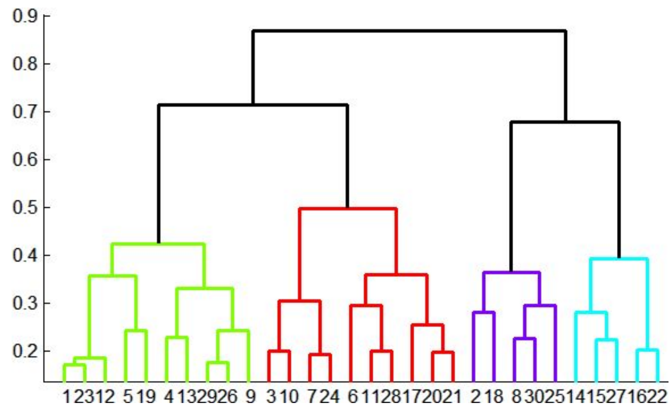
cosine distance



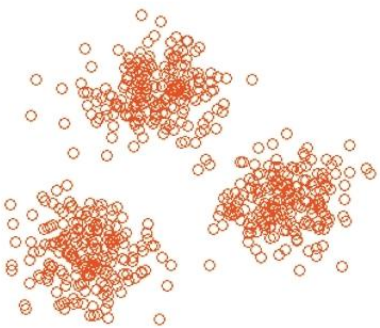
Overview: Methods of Clustering

find one then merge ...

1. **Hierarchical**
   - **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

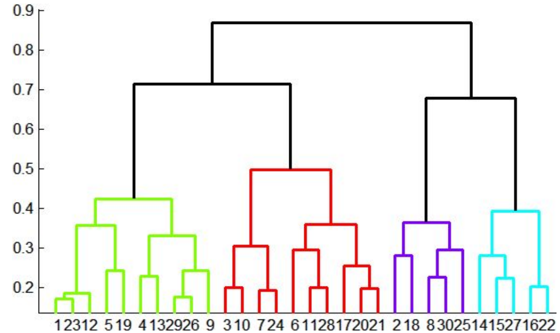


2. **Point Assignment**
   - Maintain a set of clusters
   - Points belong to "nearest" cluster



Hierarchical Clustering

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



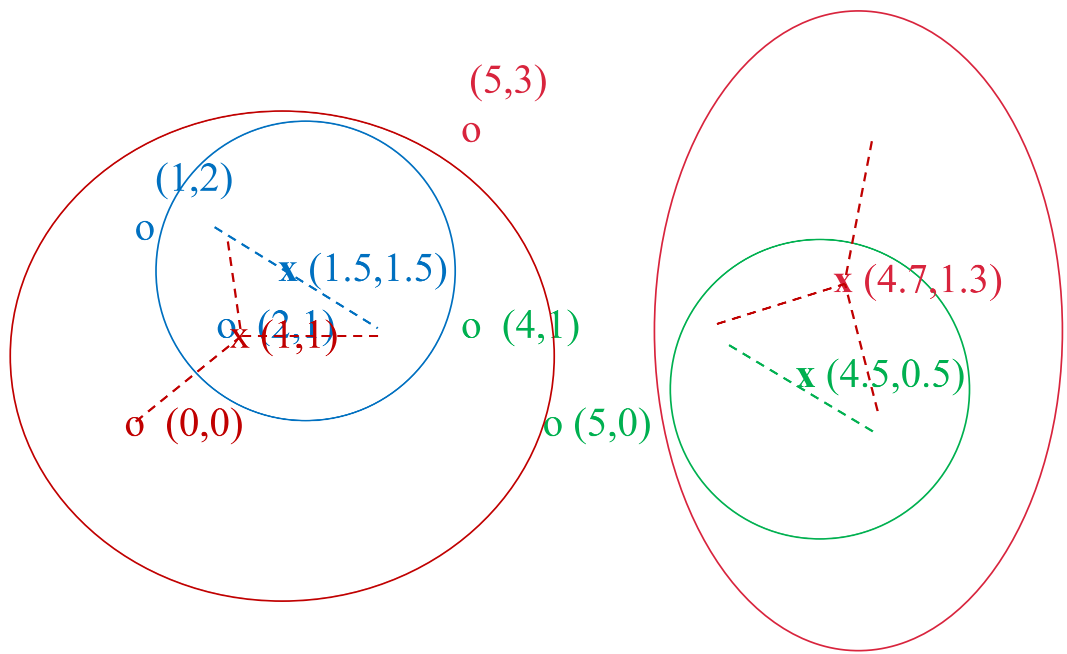
- **Three important questions:**
  - **1)** How do you represent a cluster of more than one point?
  - **2)** How do you determine the "nearness" of clusters?
  - **3)** When to stop combining clusters?

Euclidean Case?

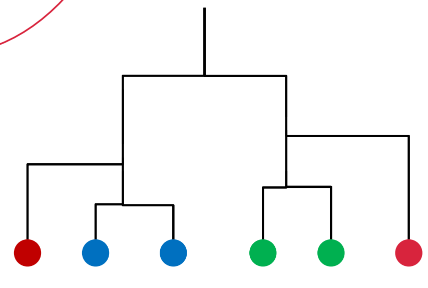
- **Key operation: Repeatedly combine two nearest clusters**
- **(1) How to represent a cluster of many points?**
  - **Euclidean case:** each cluster has a *centroid* = average of its (data)points

    isn't a data point
- **(2) How to determine "nearness" of clusters?**
  - Measure cluster distances by distances of centroids

Example: Hierarchical clustering



Data:
o … data point
x … centroid



Dendrogram

And in the Non-Euclidean Case?

What about the Non-Euclidean case?

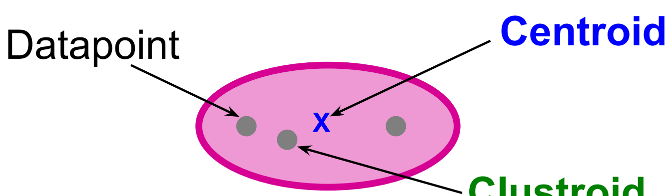
- The only "locations" we can talk about are the points themselves
  - i.e., there is no "average" of two points (e.g., students)

Approach 1:

- **(1) How to represent a cluster of many points?**
  clustroid = (data)point "***closest***" to other points
  is a data point
- **(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: $\min_{c} \sum_{x \in C} d(x,c)^2$



Cluster on
3 datapoints

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

Example

- Consider a cluster of 4 points:
  - abcd, aecdb, abecb, ecdab
- Their edit distances:

# of edit need to do to change a→b

eg : edit_d (abcd, aecdb) = 3
insert : e
remove : b
insert : b

	aecdb	abecb	ecdab
abcd	3	3	5
aecdb		2	2
abecb			4

edit_d (aecdb, abecb) = 2
insert : b
remove : b

Determine Clusteroid

- "aecdb" will be chosen as clusteroid
  - Located in "center" judged by all 3 measures

	aecdb	abecb	ecdab
abcd	3	3	5
aecdb		2	2
abecb			4

Point	Sum	Sum-sq	Max
abcd	11	43	5
aecdb	**7**	**17**	**3**
abecb	9	29	4
ecdab	11	45	5

3.

Complexity of Hierarchical Clustering

- n data points
- At most n – 1 step of merging

- Naive implementation, e.g., storing pairwise cluster distances in a matrix

	C1	C2	C3	C4
C1	0	2	3	2
C2		0	4	5
C3			0	3
C4				0

(1)

Complexity of Naive Implementation

- Initially, $O(n^2)$ for creating matrix and finding pair with minimum distance
- Subsequent merge, assuming matrix: k x k
  - Delete columns for old clusters: O(k)
  - Add new column for new cluster C’: O(k)
  - Compute dist. of C’ with other clusters: O(k)
  - Find new pair of clusters with min. dist: $O(k^2)$

=> Overall complexity: $O(n^3)$

Implementation Summary

- **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 $O(N^2 \log N)$ (read textbook)
  - **Still too expensive for really big datasets that do not fit in memory**

4. measure.

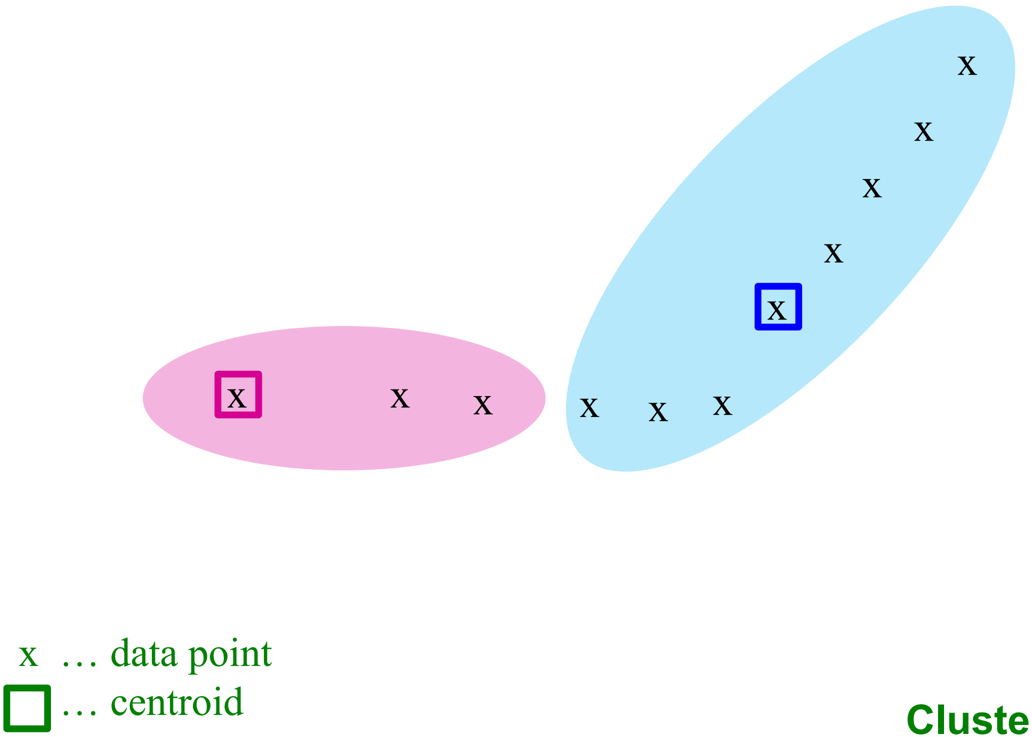
Sum Squared Error (SSE)

- SSE (sum squared error) is a common measure of the quality of a cluster.
  - It is the sum of the squares of the distances between each of the points of the cluster and the centroid.
- Sometimes, we decide to split a cluster in order to reduce the SSE.
  - Calculate the SSE before split
  - Decide the split (you can choose)
  - Calculate the sum of SSE(s) for the clusters after split
  - If SSE_after < SSE_before, then split
- Example: before split: (9,5), (2,2), and (4,8)
- how to split?
- should it be split?

k–means Algorithm(s)

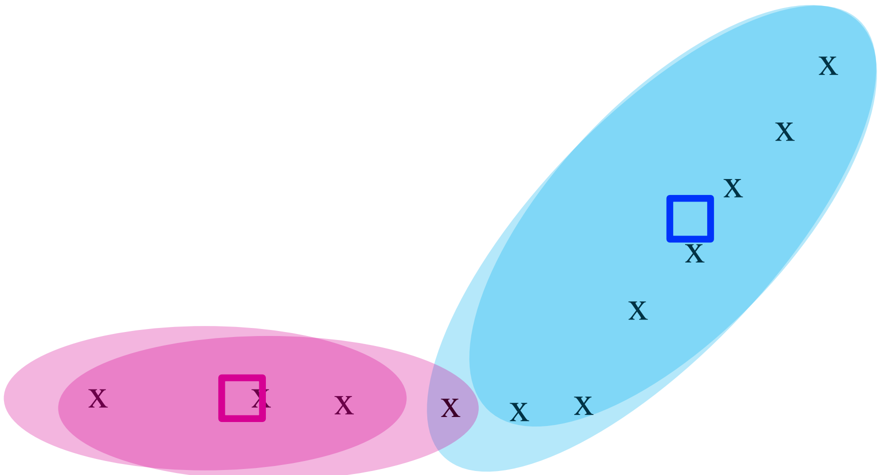
- Assumes Euclidean space/distance
- Start by picking *k*, the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:** Pick one point at random, then ***k*-1** other points, each as far away as possible from the previous points

Example: Assigning Clusters



Clusters after round 1

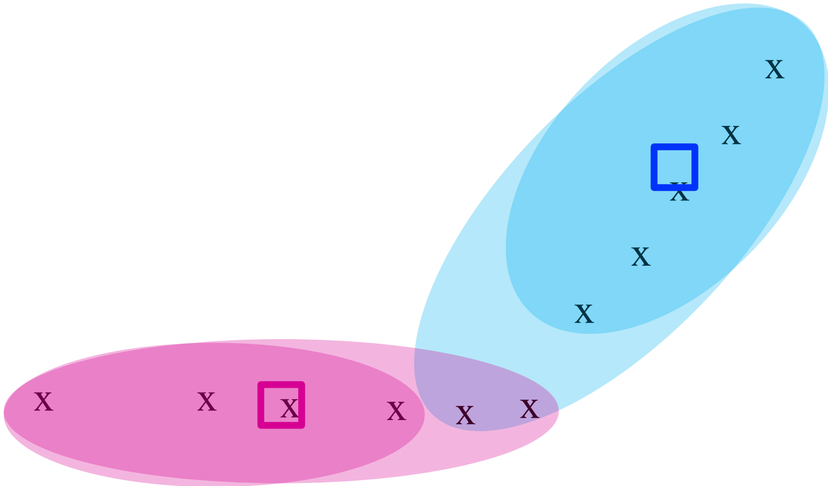
Example: Assigning Clusters



x … data point
… centroid

Clusters after round 2

Example: Assigning Clusters



x … data point
… centroid

Clusters at the end

Populating Clusters

3、step、

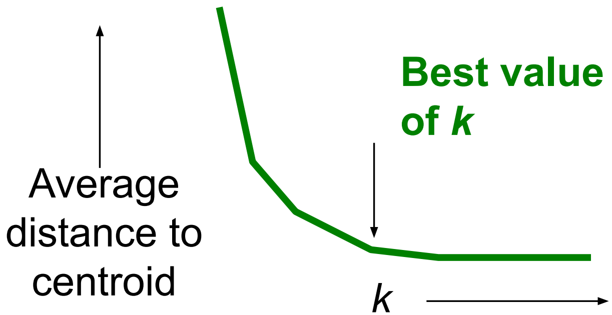
- **1)** For each point, place it in the cluster whose current centroid it is nearest
- **2)** After all points are assigned, update the locations of centroids of the ***k*** clusters
- **3)** Reassign all points to their closest centroid
  - Sometimes moves points between clusters
- **Repeat 2 and 3 until convergence**
  - **Convergence:** Points don't move between clusters and centroids stabilize

Getting the *k* right

3.

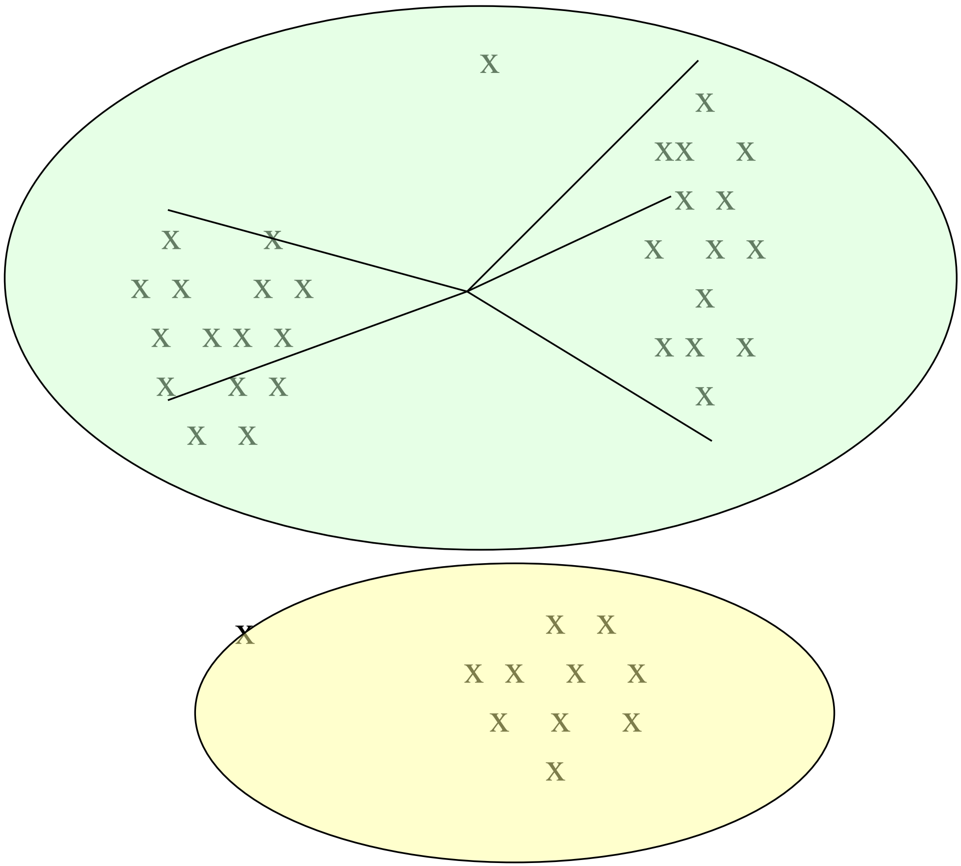
How to select *k*?

- 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



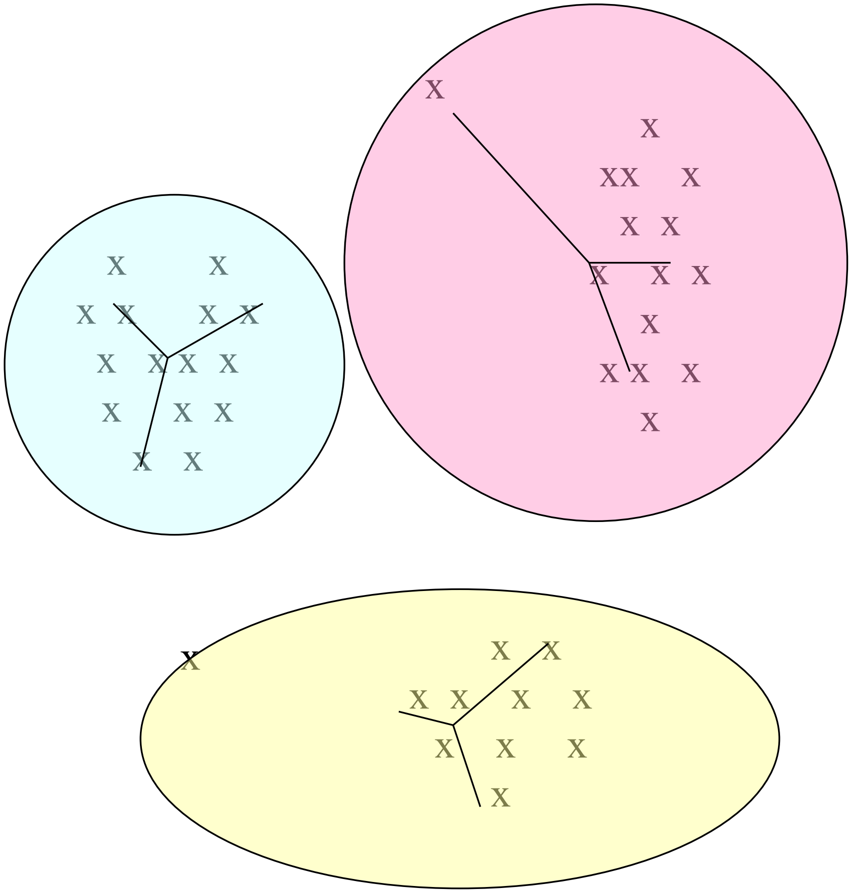
Example: Picking *k*

Too few;
many long distances to centroid.



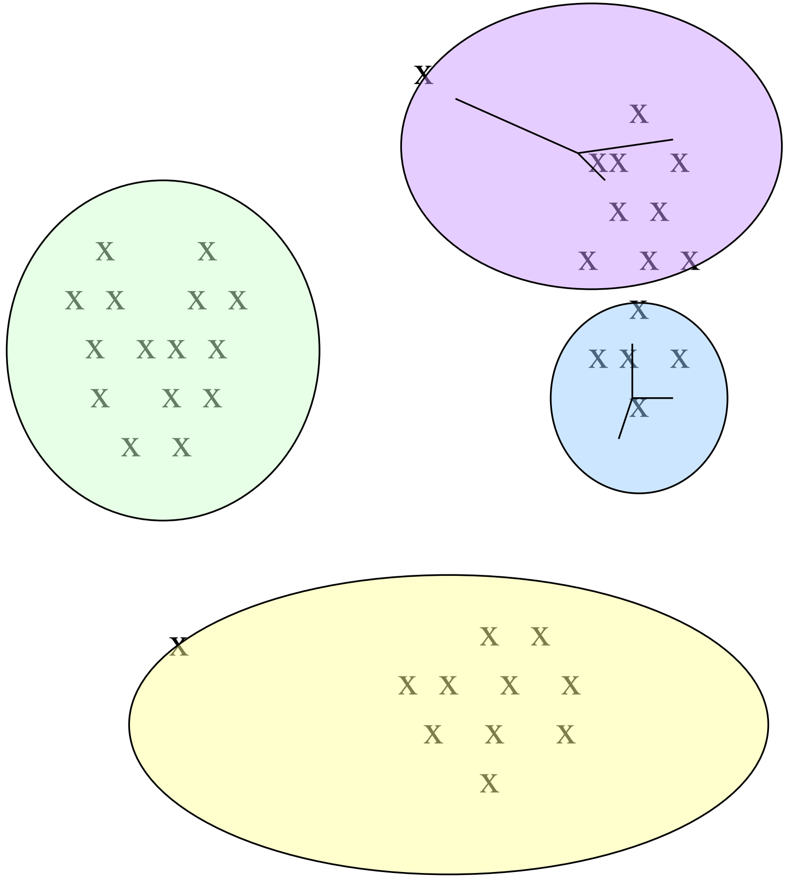
Example: Picking *k*

Just right;
distances
rather short.



Example: Picking *k*

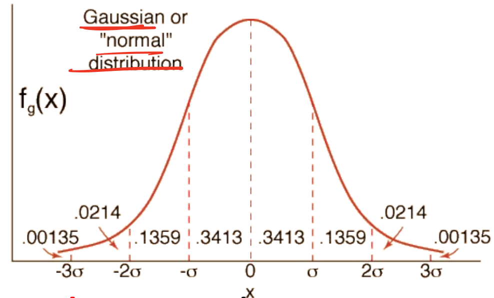
Too many;
little improvement
in average
distance.



3、 Extension of *k*-means to large data

(一) BFR Algorithm

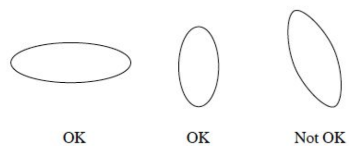
- for uniform-distributed axis-aligned shapes

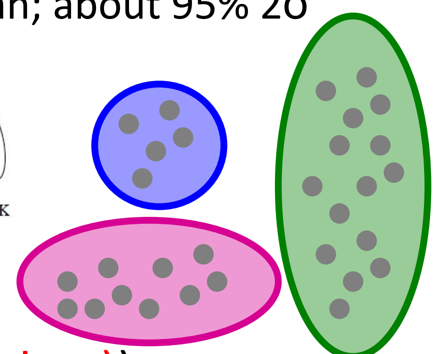


- **BFR** [Bradley-Fayyad-Reina] is a variant of *k*-means designed to handle **very large** (disk-resident) data sets

1- assumption: ①.

- Assumes that clusters are normally distributed around a centroid in a Euclidean space ② X0 of normal distribution is aligned with dimention of that
  - About 68% of values are within σ away from the mean; about 95% 2σ away; and about 99.7% 3σ away.
  - 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

2. step:

- Points are read from disk one "main-memory-full" or "memory-load" 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 first load → k cluster → remain μ and σ → second load → update μ and σ → repeat...
  - 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.

3 sets of points which we keep track of:

(1) **Discard set (DS):** don't need to remember

- Points close enough to a centroid to be summarized

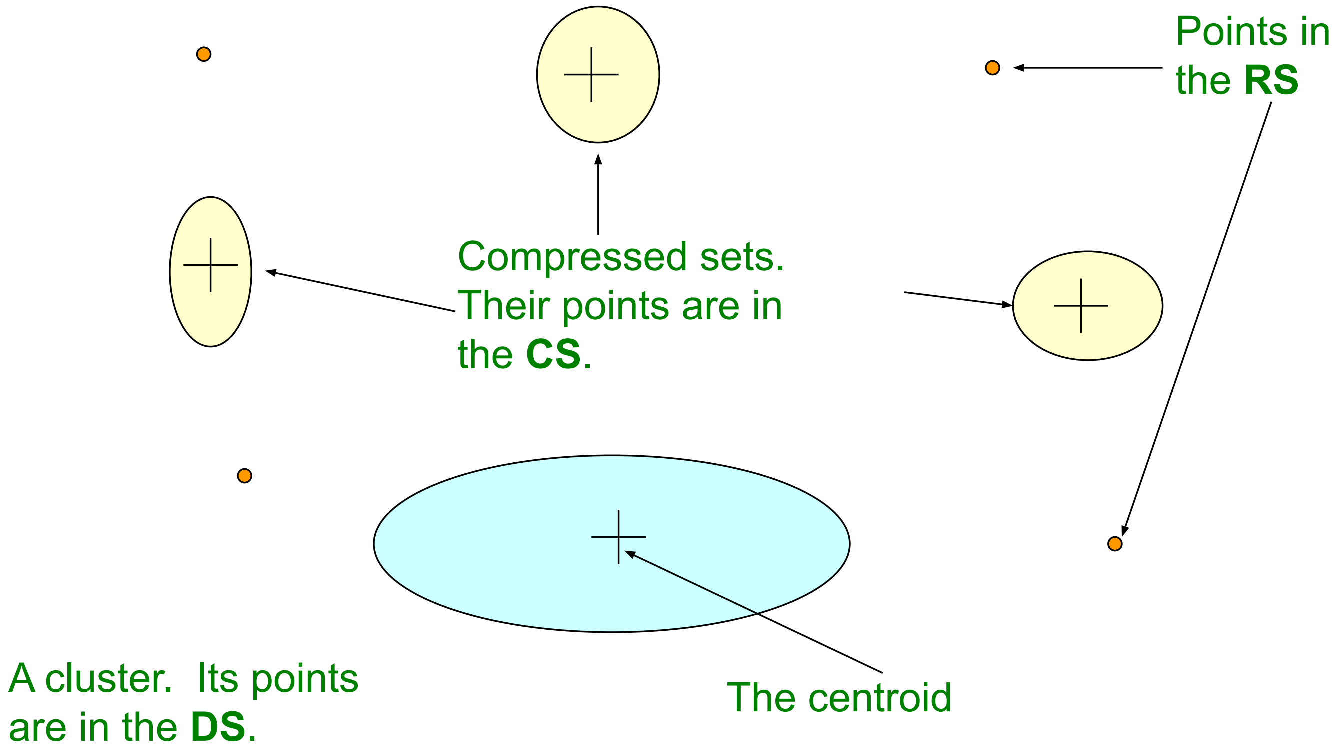
(2) **Compression set (CS):**

- Groups of points that are close together but not close to any existing centroid
- These points are summarized, but not assigned to a cluster

(3) **Retained set (RS):**

- Isolated points waiting to be assigned to a compression set

BFR: “Galaxies” Picture



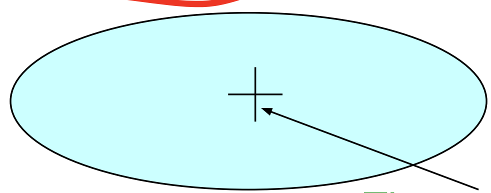
Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

4.

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

- The number of points, ***N***
- The vector ***SUM***, whose i^{th} component is the sum of the coordinates of the points in the i^{th} dimension
- The vector ***SUMSQ***: i^{th} component = sum of squares of coordinates in i^{th} dimension



Variance

$$
\begin{aligned}
Var(X) &= E\left[\left(X - E(X)\right)^2\right] \\
&= E[X^2 - 2XE[X] + (E[X])^2)] \\
&= E[X^2] - 2E[X]E[X] + (E[X])^2 \\
&= E[X^2] - (E[X])^2
\end{aligned}
$$

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \qquad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

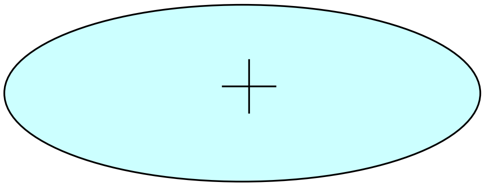
Summarizing Points: Comments

- **$2d + 1$ 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

$$\sigma^2 = \frac{\sum X^2}{N} - \mu^2$$

- Variance of a cluster's discard set in dimension i is: $\mathbf{(SUMSQ_i / N) - (SUM_i / N)^2}$
  - 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

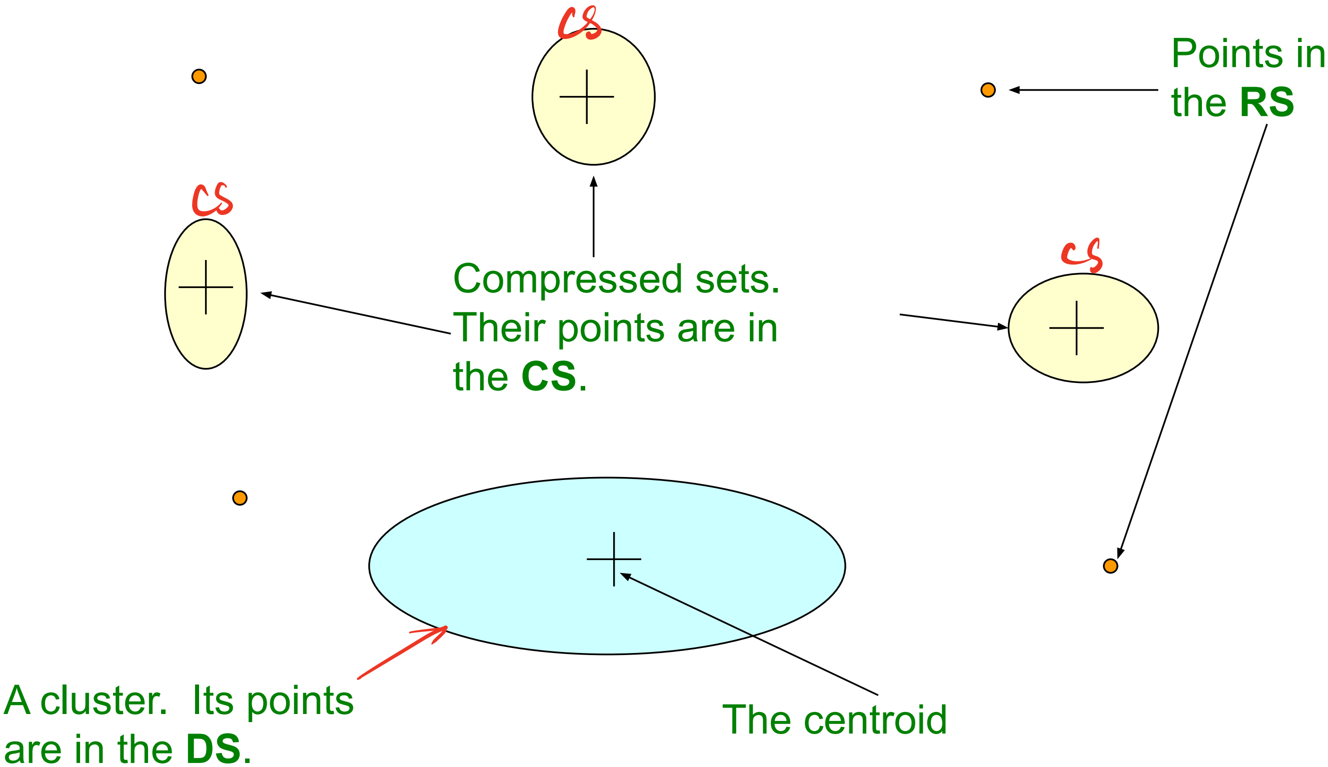
The "Memory-Load" of Points

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…

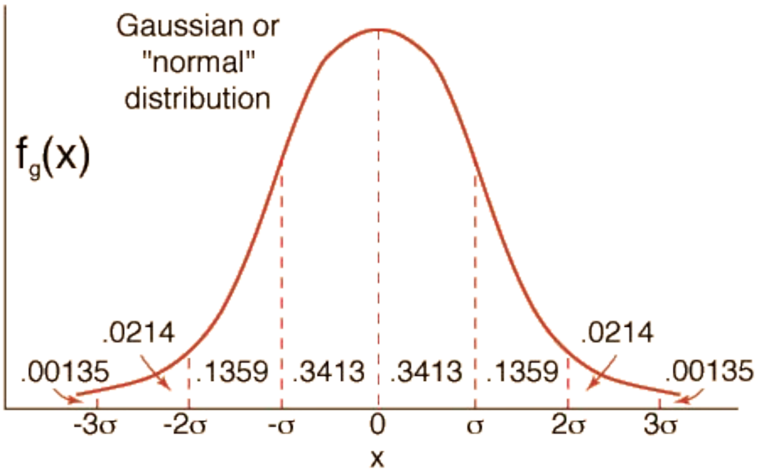
b.

- 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?

Define "Sufficiently Close"

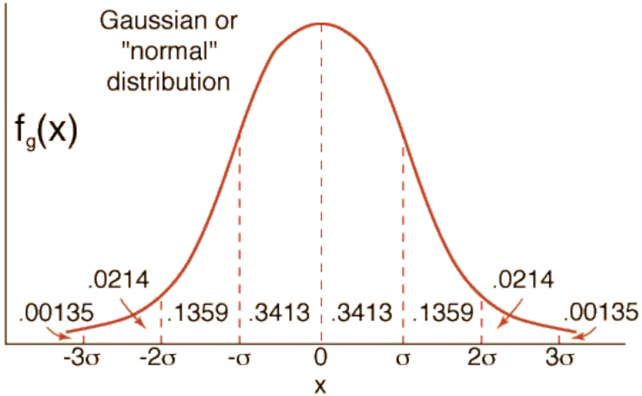
- ~68% of points: 1 σ away from mean
- ~95% of points: 2 σ away
- ~99% of points: 3 σ away



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

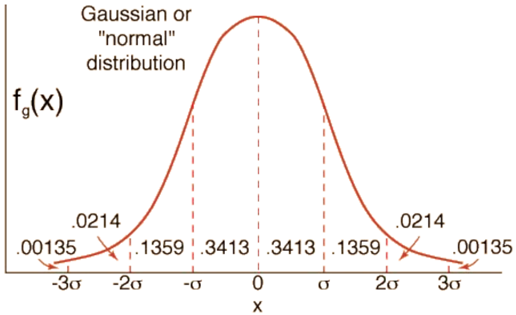
- **Normalized Euclidean distance from centroid**
- For point $(x_1, \ldots, x_d)$ and centroid $(c_1, \ldots, c_d)$
  1. Normalize by σ_i in each dimension: $y_i = (x_i - c_i) / \sigma_i$
  2. Take sum of the squares of the y_i
  3. Take the square root

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

σ_i … standard deviation of points in the cluster in the i^{th} dimension

Mahalanobis Distance

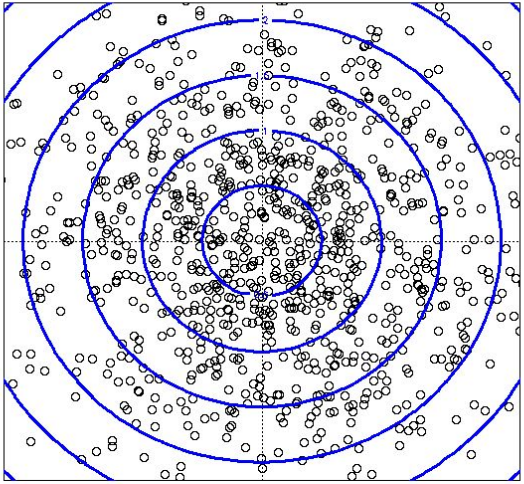
- If clusters are normally distributed in d dimensions, then after transformation, one standard deviation = $\sqrt{d}$
  - i.e., 68% of the points of the cluster will have a Mahalanobis distance $< \sqrt{d}$
- Accept a point for a cluster if its M.D. is < some threshold, e.g. **2** standard deviations
  - About 68% of values are within σ away from the mean; about 95% 2σ away; and about 99.7% 3σ away.



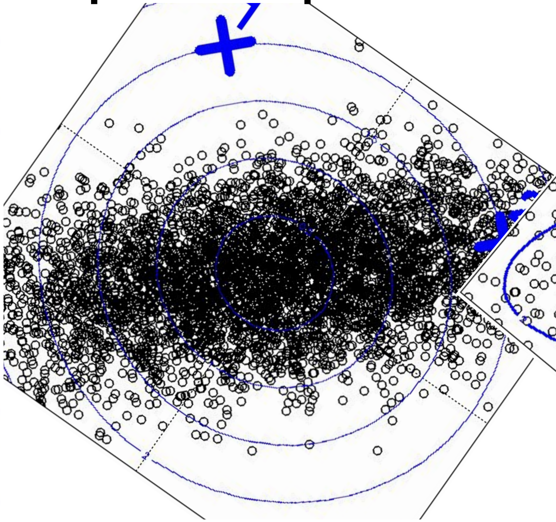
Picture: Equal M.D. Regions

- **Euclidean vs. Mahalanobis distance**

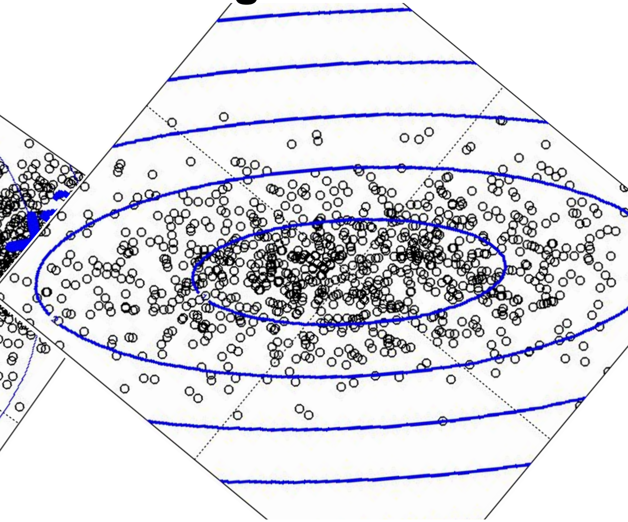
Contours of equidistant points from the origin



Uniformly distributed points, Euclidean distance



Normally distributed points, Euclidean distance



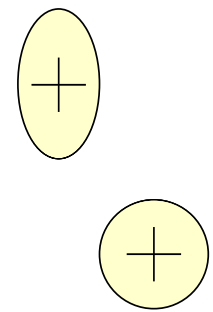
Normally distributed points, Mahalanobis distance

(2)

Should two CS clusters be combined?

Q2) Should two CS subclusters be combined?

- Compute the variance of the combined subclusters
  - ***N***, ***SUM***, and ***SUMSQ*** allow us to make that calculation quickly
- Combine if the combined variance is below some threshold

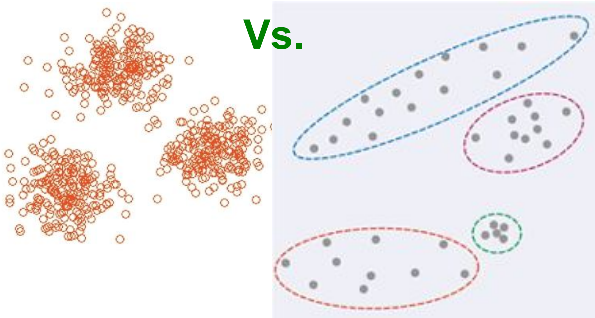


- **Many alternatives:** Treat dimensions differently, consider density

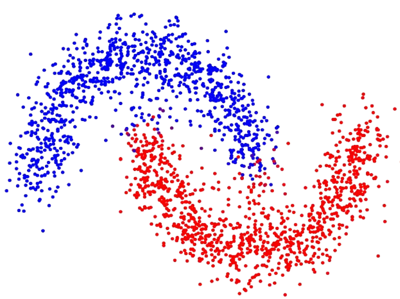
(2).

CURE: Clustering Using REpresentitives

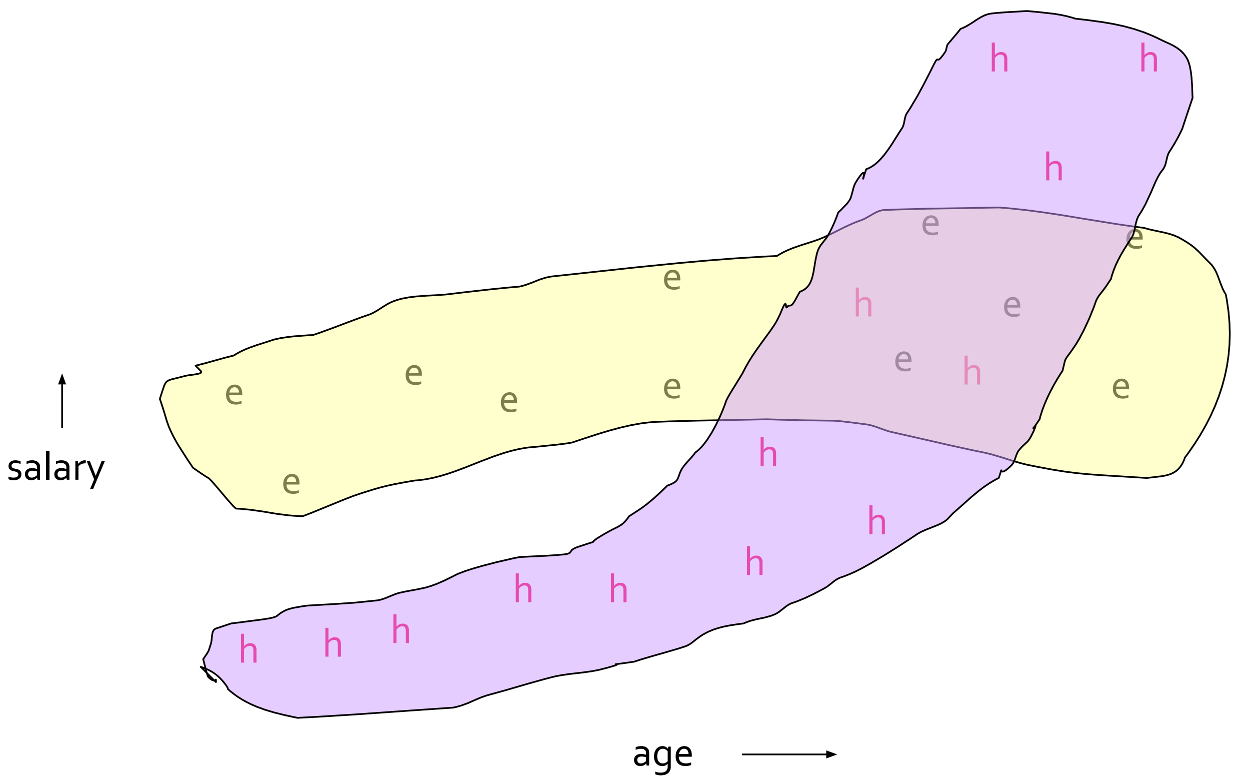
The CURE Algorithm - for arbitrary shapes



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



Example: Stanford Salaries



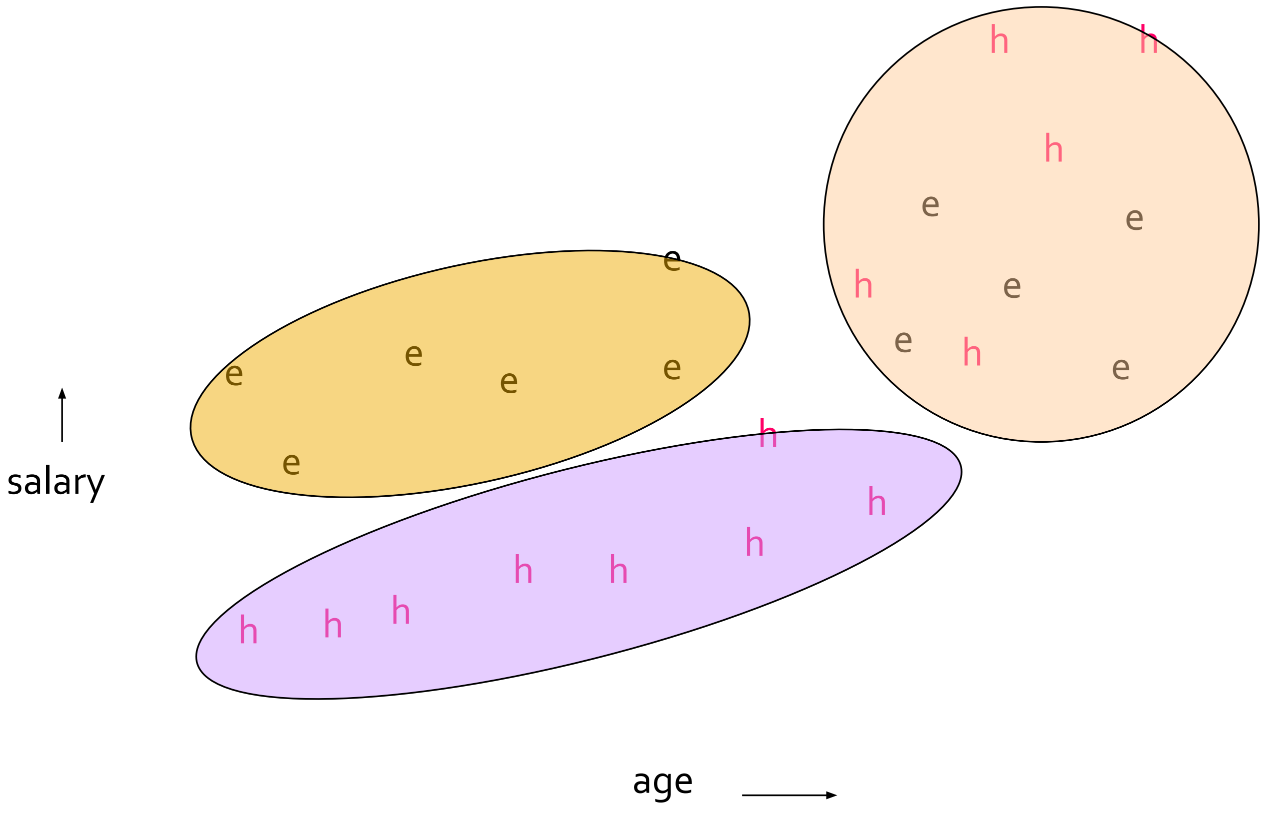
Starting CURE

2. step.

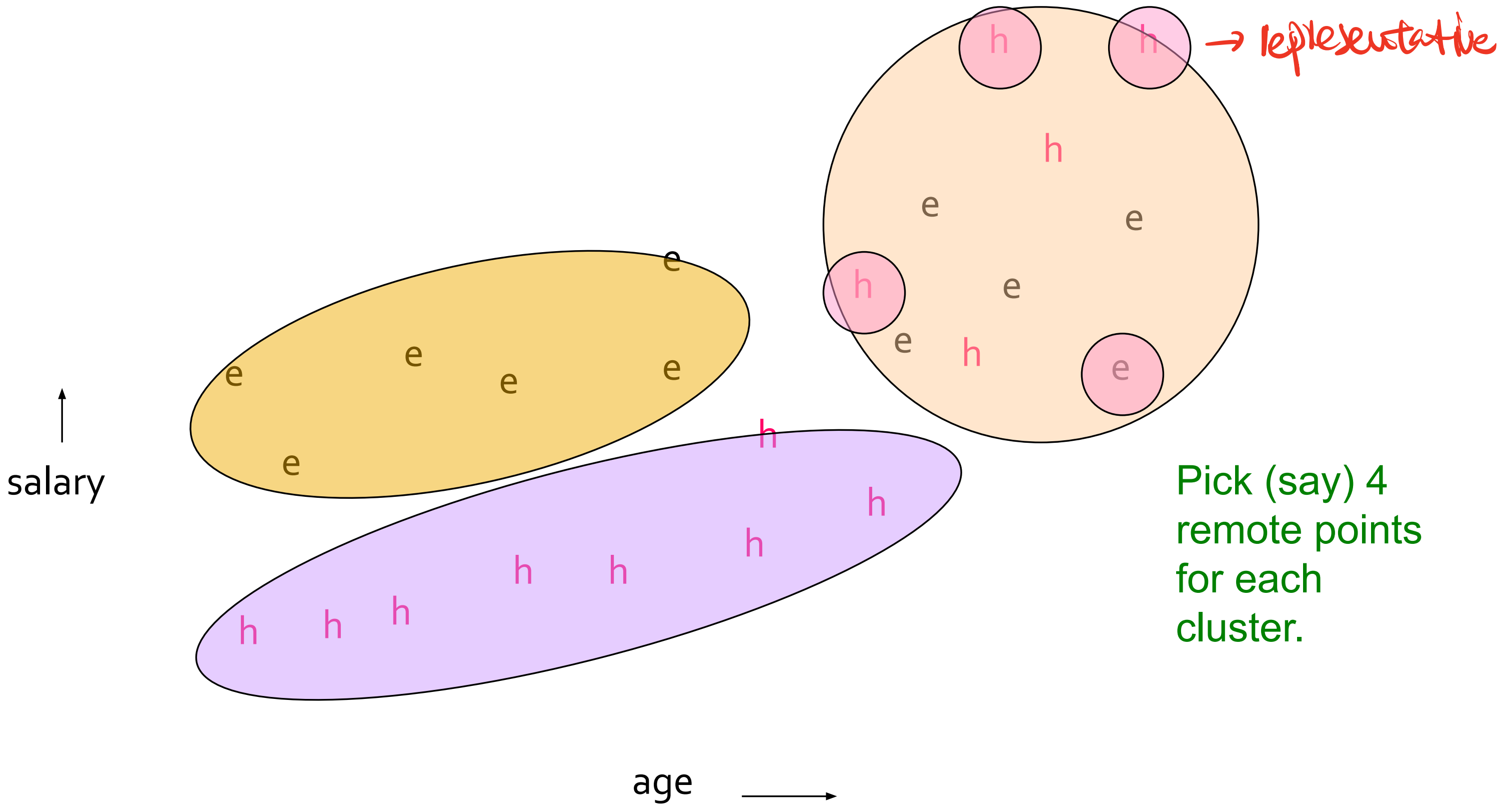
Two-Pass Algorithm: Pass 1

- 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

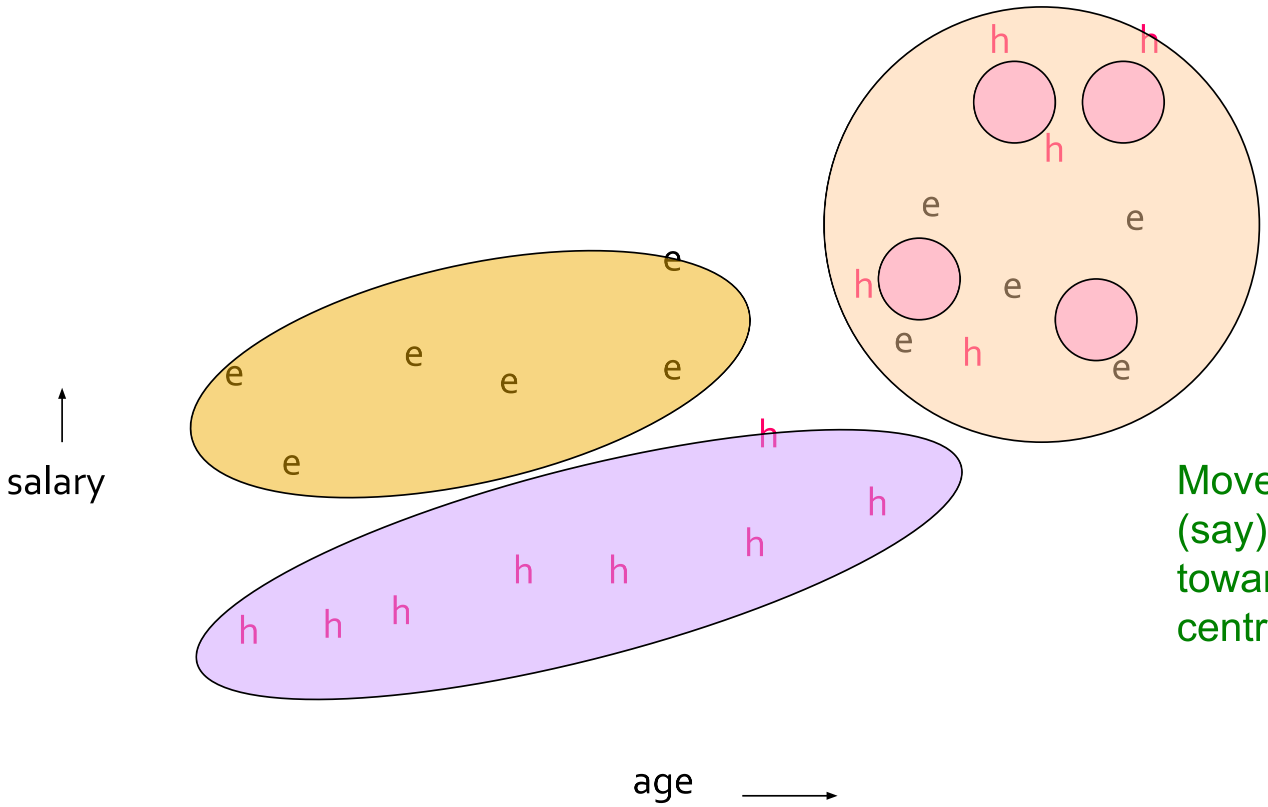


Example: Pick Dispersed Points



Pick (say) 4 remote points for each cluster.

Example: Pick Dispersed Points



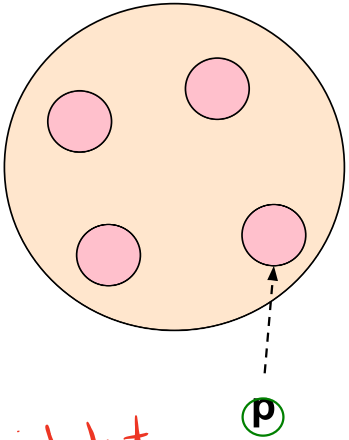
Move points (say) 20% toward the centroid.

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

not centroid but one of representatives



3) Compare CURE with BFR

- 1) Distribution of data
  - CURE: do not assume any particular distribution
  - BFR: data should be normally distributed
- 2) Representation of cluster
  - CURE: a set of representatives
  - BFR: centroid
- Common: both assume data in Euclidean space

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**

Standardization vs. Normalization

- Standardization vs. Normalization
- Incomparable units
  - "Age" vs. "GPA"
- Same units, irrelevant features
  - commuting miles vs. maximum jogging miles
- Same units, similar features
  - air quality values today vs. air quality tomorrow
  - length of right vs. left arms
- Others