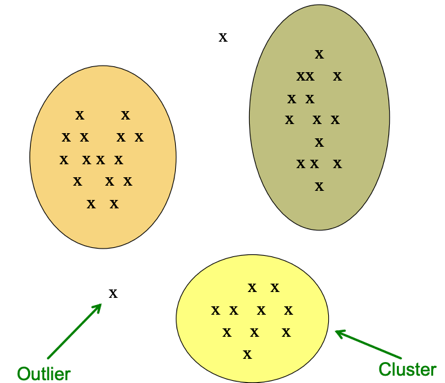
**Lecture 3 – Clustering**

Hierarchical Clustering

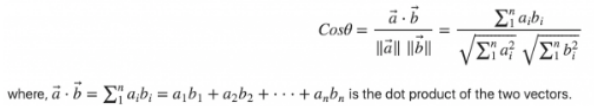
* The problem of clustering - Given a set of points, with a notion of distance between points, group the points into some number of clusters, so that;
* Members of a cluster are close/similar to each other
* Members of different clusters are dissimilar
* Example: Clusters & Outliers

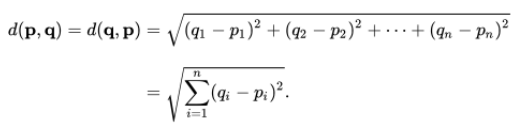


* Why is it hard?
* Clustering in two dimensions looks easy
* Clustering small amounts of data looks easy
* And in most cases, looks are not deceiving
* Many applications involve not 2, but 10 or 10,000 dimensions
* High-dimensional spaces look different: Many pairs of points are at about the same distance.
* Clustering Problem: Documents

Represent a document by a vector (x1, x2,…, x3), where xi = 1 iff the i th word (in some order) appears in the document.

* 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**
* Hierarchial
* Agglomerative (bottom up)
* Divisive (top down)
* 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 combing clusters

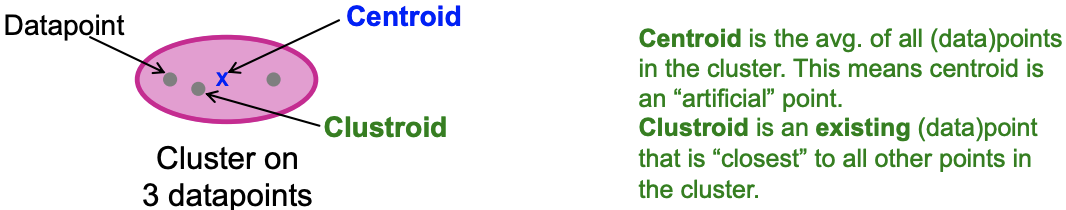
* Euclidean case:

1. How do you represent a cluster of more than one point? – **centroid** = average of its data points
2. How do you determine the “nearness” of clusters? – measure cluster distance by distances of centroids
3. When to stop combing clusters

* Non-Euclidean case:
* APPROACH 1

1. How do you represent a cluster of more than one point? – **clustroid** = data point “**closest**” to other points. Where “closest” means:

- Smallest maximum distance to other points

- Smallest average distance to other points

1. How do you determine the “nearness” of clusters? – measure cluster distance by distances of clustroids as if they were centroids.
2. When to stop combing clusters

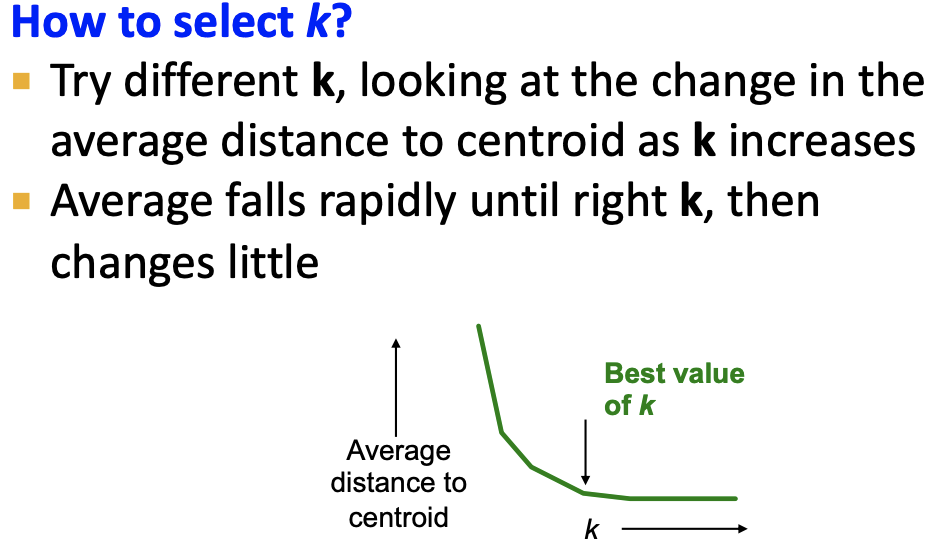
* APPROACH 2

K-Means Clustering Initialization

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

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
4. Repeat 2 and 3 until convergence. Convergence: Points don’t move between clusters and centroids stabilize



BFR Algorithm

Bradley-Fayyad-Reina is a variant of k-means designed to handle **very large** (disk-resident) data sets. Clusters are axis aligned ellipses.

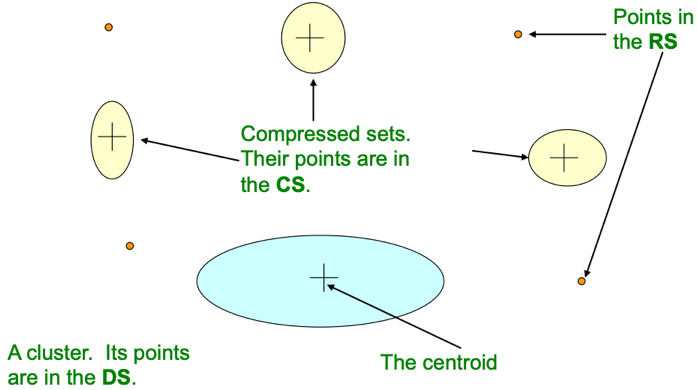
* Algorithm
* Points are read from disk into main memory in chunks.
* Most points from previous memory loads are summarized by simple statistics
* To begin, from the initial load we select the initial k centroids by some sensible approach:

1. Take k random points
2. Take a small random sample and cluster optimally
3. 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:

* 1. Discard set (DS): Close enough to a centroid to be summarized. Only summaries are kept in main memory
  2. Compression set (CS): Summarized, but not assigned to a cluster. Only summaries are kept in main memory
  3. Retained set (RS): Isolated points to be assigned to a compression set. Held in main memory exactly as they are.



**Summarizing Sets of Points**

* The number of points, **N**
* The vector **SUM**, whose ith component is the sum of the coordinates of the points in the ith dimension
* The vector **SUMSQ**: ith component = sum of squares of coordinates in ith dimension
* **2d + 1** values represent any size cluster
* **d** = number of dimensions
* Average in each dimension (the centroid) can be calculated as **SUMi / N**
* **SUMi** = ith component of SUM
* Variance of a cluster’s discard set in dimension ***i*** is: **(SUMSQi / N) – (SUMi / N)2**
* Standard deviation is the square root of variance

Initialization in CURE

The CURE Algorithm is a 2 pass algorithm.

* Problem with BFR:
* 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
* Use a collection of representative points to represent clusters

**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

**Pass 2**

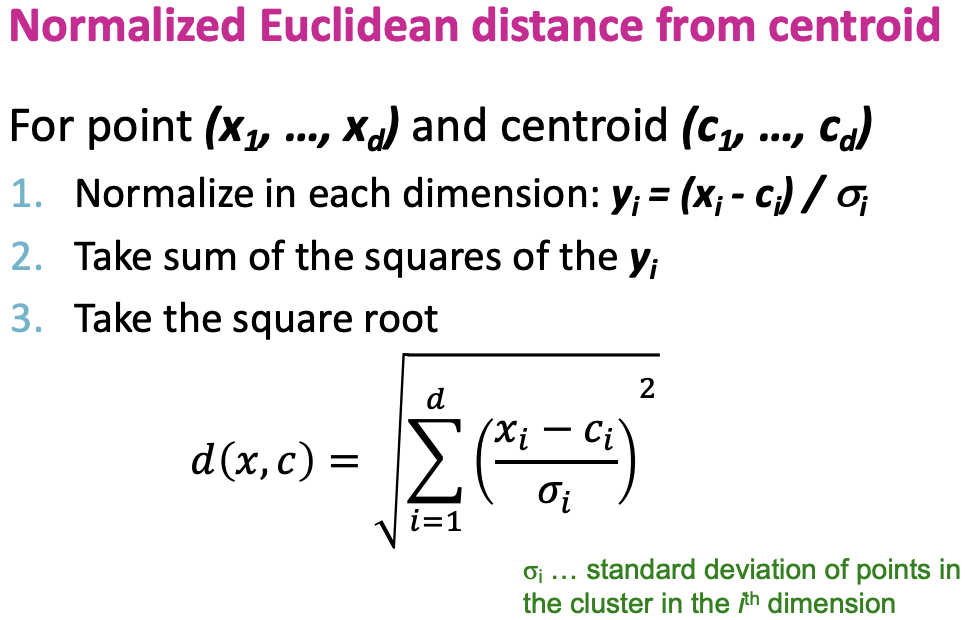
Now, rescan the whole dataset and visit each point p in the data set.

Place it in the “**closest cluster**”

* Normal definition of “closest”: Find the closest representative to **p** and assign it to representative’s cluster

Mahalanobis

* **Q1) We need a way to decide whether to put a new point into a cluster (and discard)**
* e.g, BFR suggests: The Mahalanobis distance is less than a threshold



* 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 (oval/ ellipse shape)
* Combine if the combined variance is below some threshold(circular shape)