PageRank and Beyond

Sarunas Girdzijauskas

ID2211

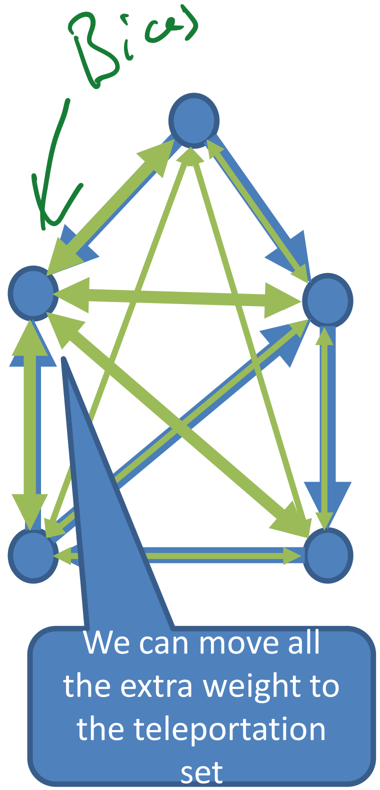
March 2019

Recap

- Transition (Random Walk) Matrix
- Convergence of Random Walk
- Graph Laplacian,
- Graph Spectra, Eigen gap,
- Structure of the Web,
- PageRank,
- Topic-Specific PageRank

Topic Specific Page Rank

- Insight: **Bias the random walk** towards ”relevant set nodes”
- Instead of teleporting to ”any node” - teleport to ”relevant pages” (**teleport set**) for topic-specific PageRank
- Once we have a biased (green) weights we recalculate PageRank in a regular fashion

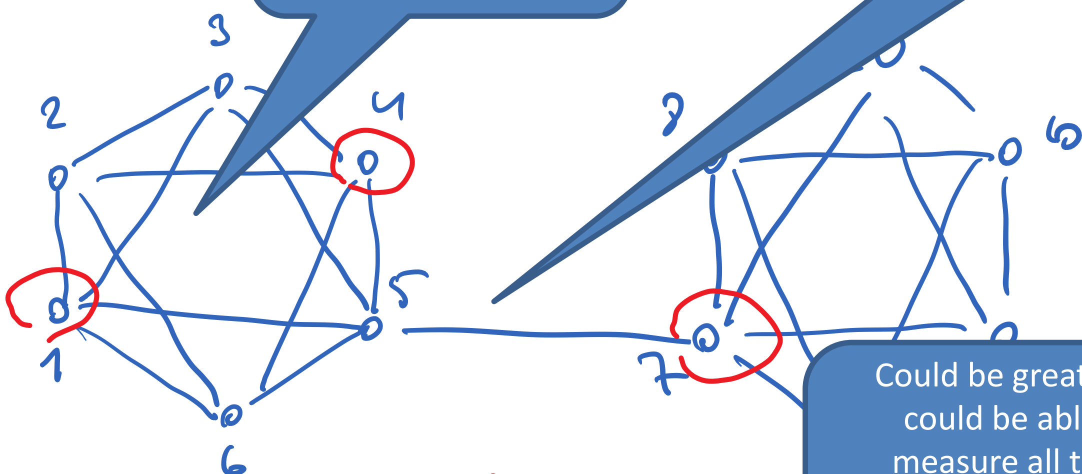


Application to Measuring/Link Prediction

Proximity in Graphs

There are multiple 2hop paths from 4 to 1 but only one 2 hop path from 1 to 7

Shortest Path might give the same distance (2)



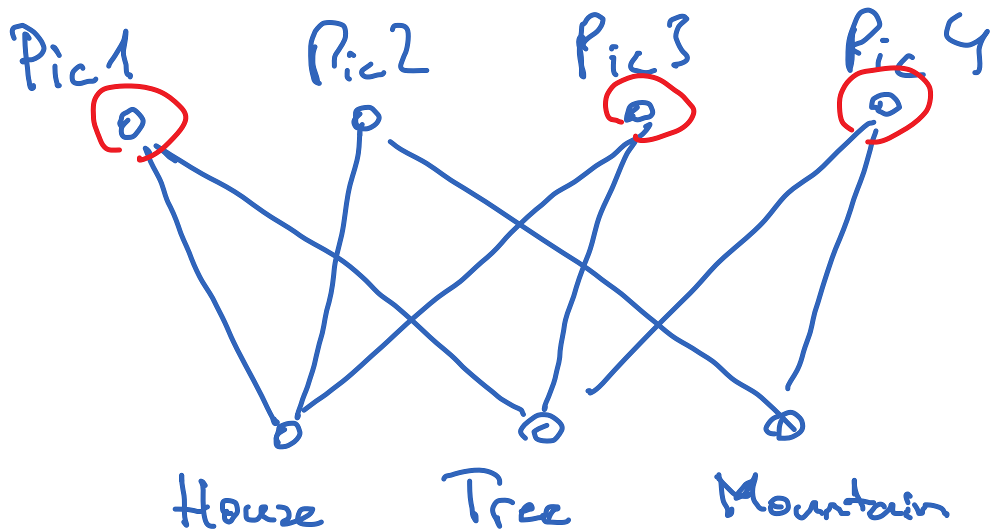
Any ideas?

Could be great if we could be able to measure all these different ways to get between two given nodes!

How close is 4 to 1?

How close is 7 to 1?

Proximity in Graphs (cont.)

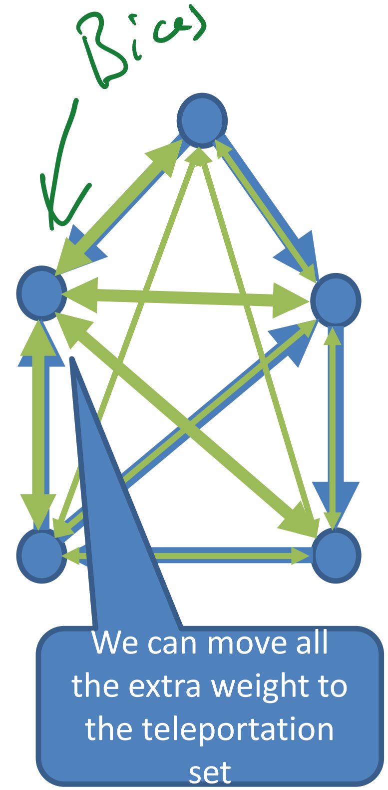


How similar is Pic 1 to Pic 4?

How similar is Pic 1 to Pic 3?

Topic Specific Page Rank

- Insight: **Bias the random walk** towards ”relevant set nodes”
- Instead of teleporting to ”any node” - teleport to ”relevant pages” (**teleport set**) for topic-specific PageRank
- Once we have a biased (green) weights we recalculate PageRank in a regular fashion
- **What happens if our teleport set is the “initial node itself”?**

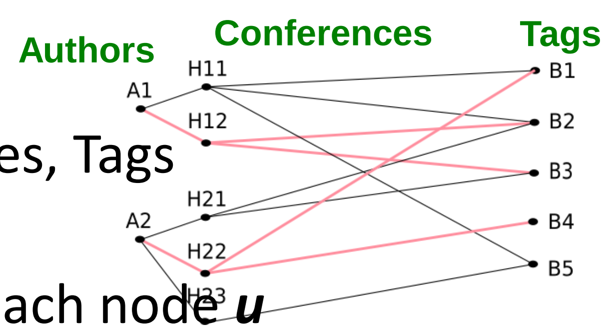


SimRank: Idea

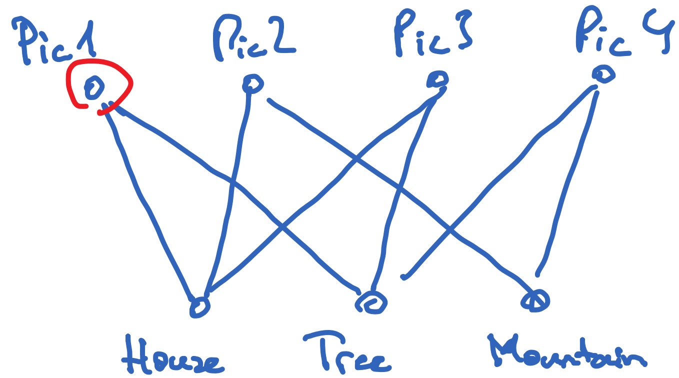
G.Jeh et al. SimRank: A Measure of Structural-Context Similarity

AKA PageRank with Restarts

- **Topic Specific PageRank** from node ***u***: **teleport set *S* = {*u*}**
  – Resulting scores measures similarity to node ***u***
- **SimRank:** Random walks from a **fixed node** on ***k***-partite graphs
- **Setting:** ***k***-partite graph with ***k*** types of nodes
  – E.g.: Authors, Conferences, Tags
- **Any problems?**
  – Must be done once for each node ***u***
  – Used in *Pinterest* for recommendation.



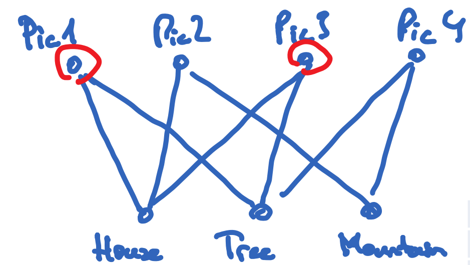
SimRank Example



- What is the most related picture to Pic1?
- Topic-Specific PageRank with teleport set ={Pic1}
  – Random Walk with restart

SimRank Example (cont.)

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0.3333 & 0.3333 & 0 \\ 0 & 0 & 0 & 0 & 0.3333 & 0 & 0.5000 \\ 0 & 0 & 0 & 0 & 0.3333 & 0.3333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.3333 & 0.5000 \\ 0.5000 & 0.5000 & 0.5000 & 0 & 0 & 0 & 0 \\ 0.5000 & 0 & 0.5000 & 0.5000 & 0 & 0 & 0 \\ 0 & 0.5000 & 0 & 0.5000 & 0 & 0 & 0 \end{pmatrix}$$



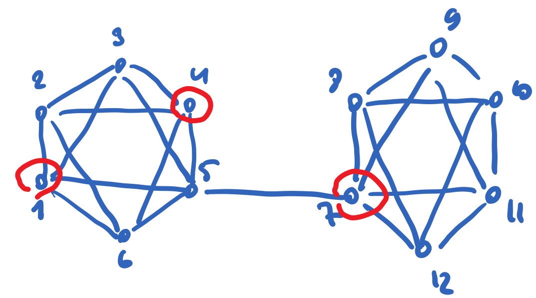
1	0.7287
2	0.3179
3	0.3102
4	0.2404
5	0.3298
6	0.3102
7	0.0744
8	0.0271
9	0.0271
10	0.0216
11	0.0271

Restart on Pic1 ($\beta = 0.8$)

$v =$

Pic1	0.3024
Pic2	0.0753
Pic3	0.1024
Pic4	0.0753
House	0.1921
Tree	0.1921
Mountain	0.0603

$$M_R = \begin{pmatrix} 0.2000 & 0.2000 & 0.2000 & 0.2000 & 0.4667 & 0.4667 & 0.2000 \\ 0 & 0 & 0 & 0 & 0.2667 & 0 & 0.4000 \\ 0 & 0 & 0 & 0 & 0.2667 & 0.2667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2667 & 0.4000 \\ 0.4000 & 0.4000 & 0.4000 & 0 & 0 & 0 & 0 \\ 0.4000 & 0 & 0.4000 & 0.4000 & 0 & 0 & 0 \\ 0 & 0.4000 & 0 & 0.4000 & 0 & 0 & 0 \end{pmatrix}$$



Calculate $Mv = v$

PageRank: Summary

- **"Normal" PageRank:**
  - Teleports uniformly at random to any node
  - All nodes have the same probability of surfer landing there
- **Topic-Specific PageRank also known as Personalized PageRank:**
  - Teleports to a topic specific set of pages
  - Nodes can have different probabilities of surfer landing there
- **Random Walk with Restarts (SimRank):**
  - Topic-Specific PageRank where teleport is always to the same node.

Clustering
ID2211

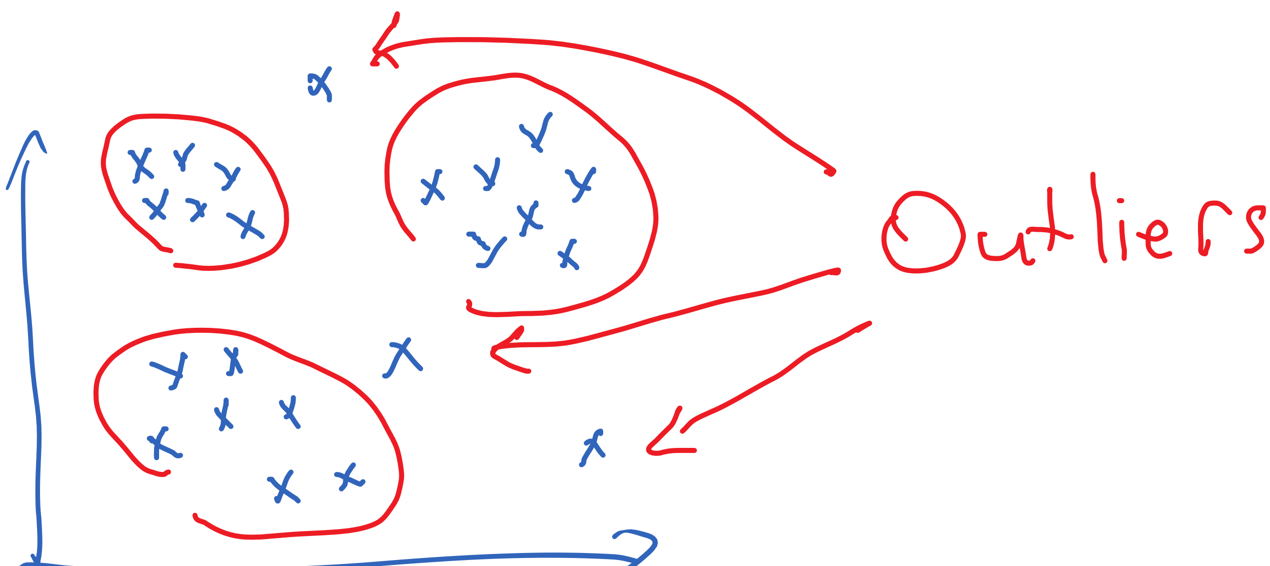
Sarunas Girdzijauskas

sarunasg@kth.se

Based on slides from http://mmds.org/

Clustering

- **What is it? Any ideas?**
- Given a cloud of data points we want to understand their stucture

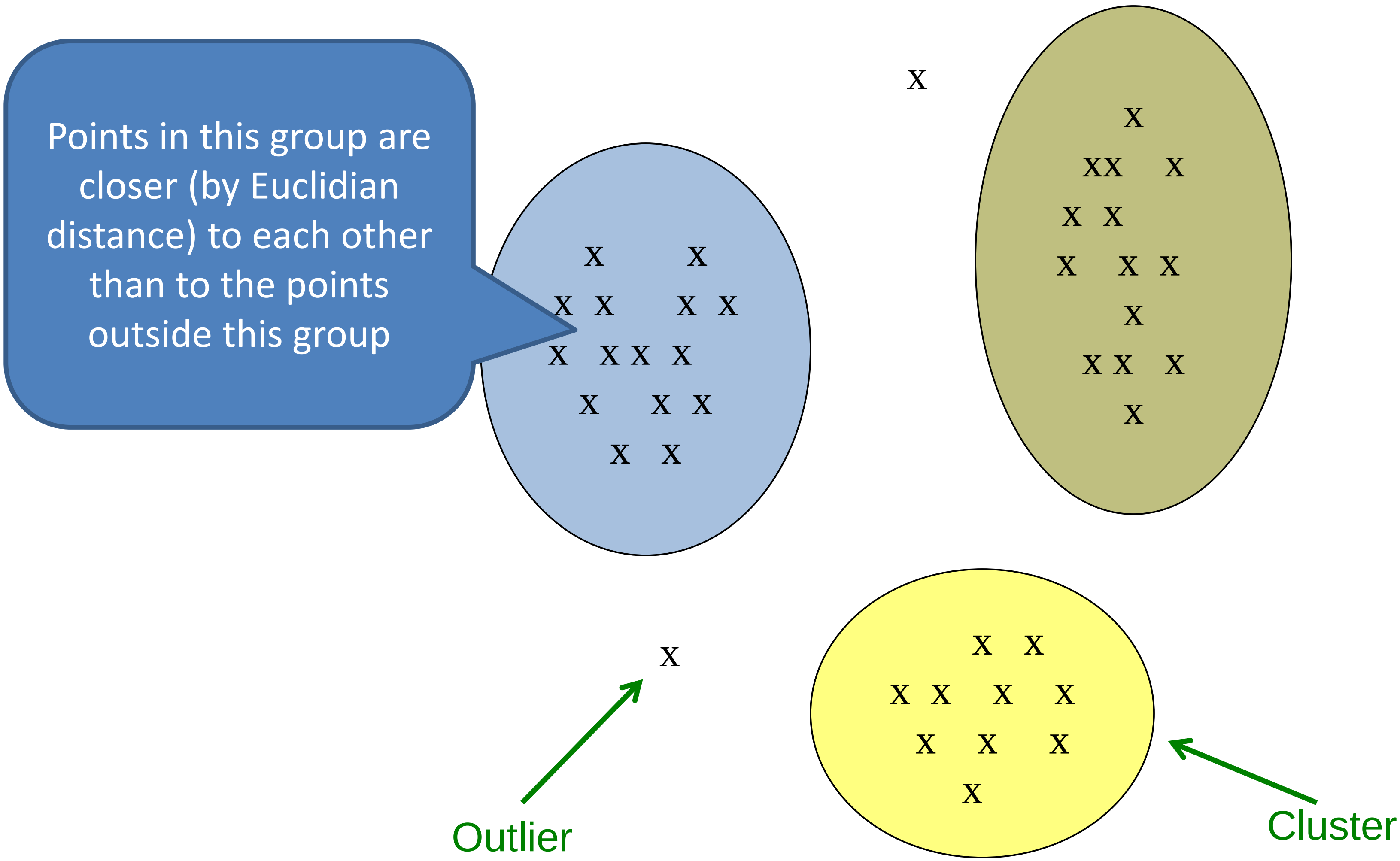


- Goal of Clustering: Finding these groups!
  - Usually in high dimensional spaces

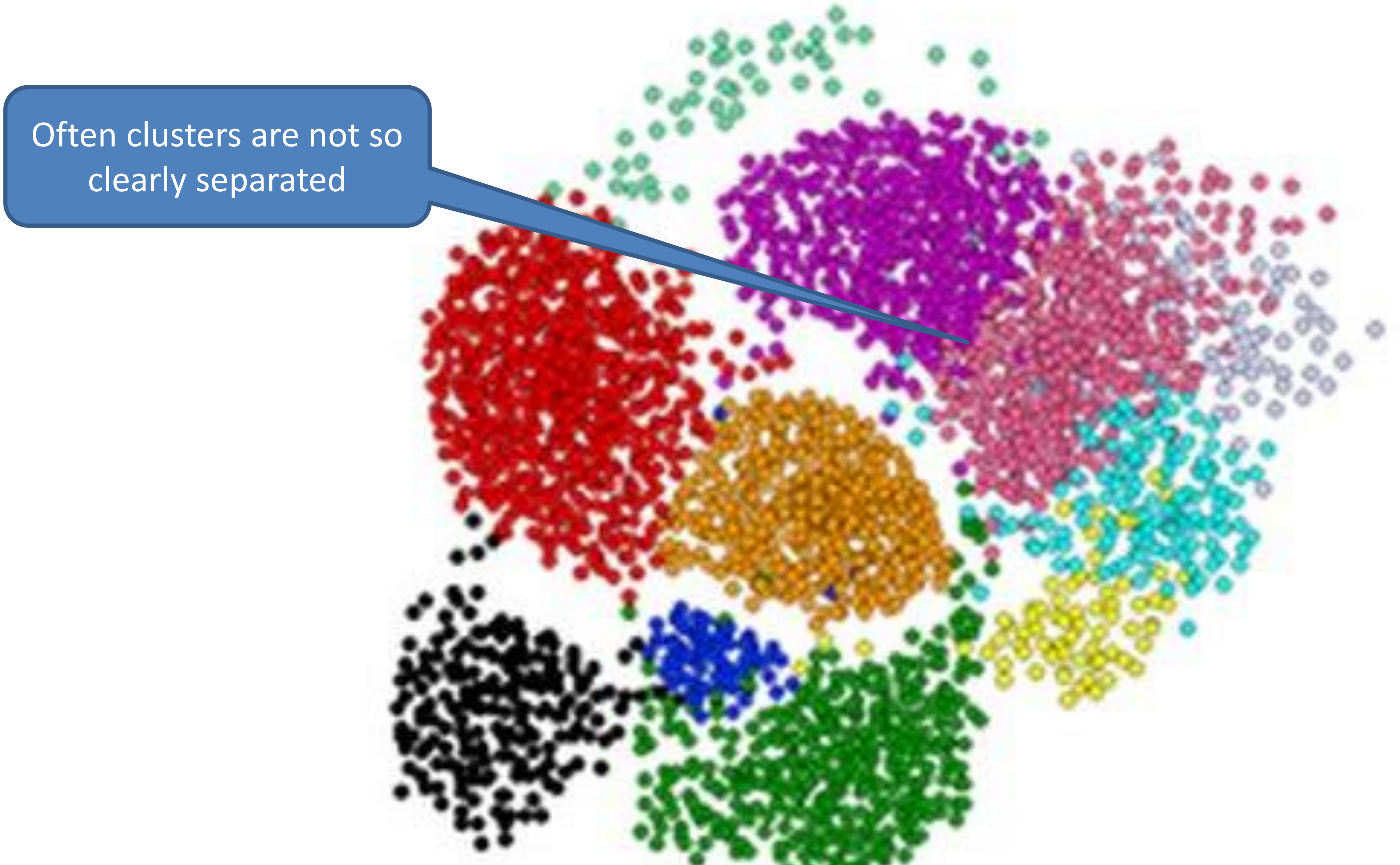
The Problem of Clustering

- 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



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

- **Problem:** Many applications involve not 2, but 10 or 10,000 dimensions
- **High-dimensional spaces look different:**
  - **Curse of Dimensionality**
  - Almost all pairs of points are at about the same distance! *So how to group them?*

Example Clustering Problem: Galaxies

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

Example 2: Music CDs

- **Intuitively: Music divides into categories, and customers prefer a few categories**
  - But what are these 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:

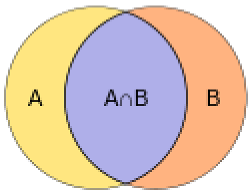
- Think of a space with one dim. for each customer
  - Values in a dimension may be 0 or 1 only
  - A CD is a point in this space $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} customer bought the CD
- For Amazon, the dimension is tens of millions
- **Task:** Find clusters of similar CDs

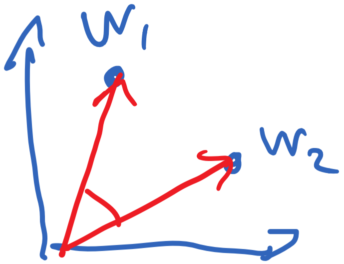
Example 3: Documents

- **Problem**: Group together documents on the same topic
- **Finding topics**:
  - represent a document by a vector $(x_1, x_2, \ldots, x_k)$, 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
- ***What is the next step once we have data representation?***
  - ***Defining Distance!***

Defining distance between data points

- ***Any ideas?***
  - **Documents as set or words:** Measure similarity by the **Jaccard distance**
  - **Documents as points in the k-dimensinal space of words:**
    - $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i th word appears in the document
    - Measure similarity by **Euclidean distance**
  - **Documents as vectors:**
    - Vector from origin to $(x_1, x_2, \ldots, x_k)$
    - Measure similarity by the angle: **cosine distance**

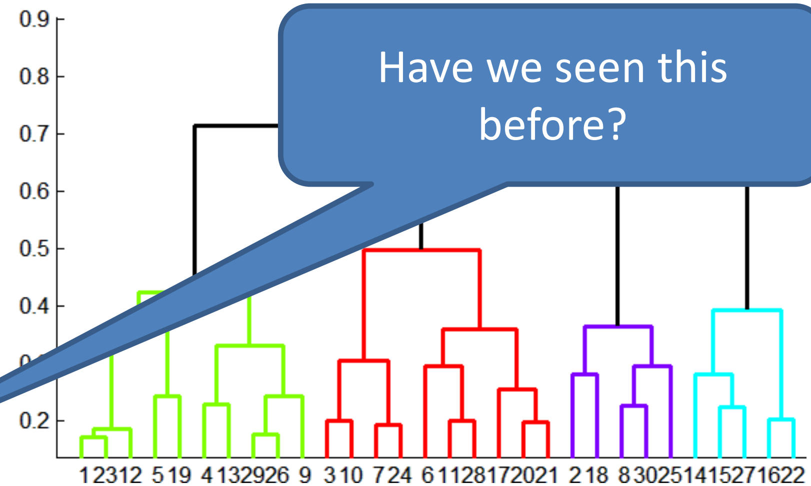




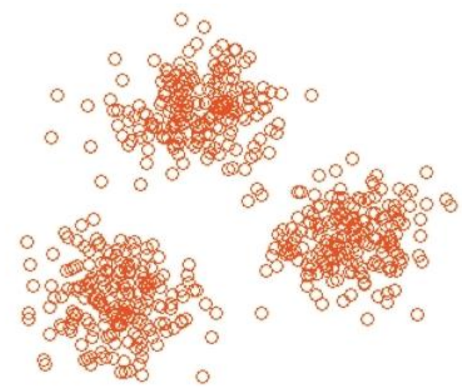
- **Depending on an application certain distance measures make more sense than others**

Overview: Methods of Clustering

Have we seen this before?



- **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

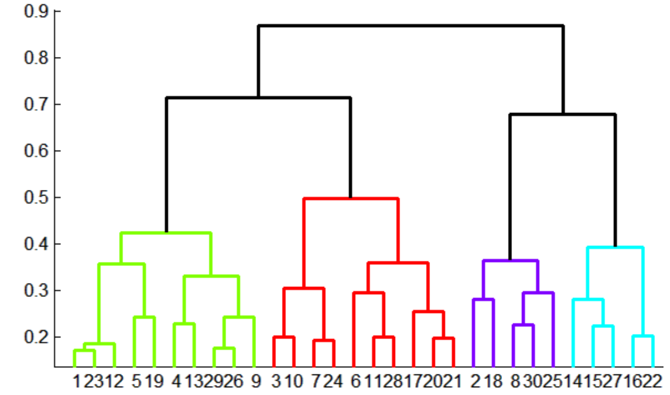


- **Point assignment:**
  - Maintain a set of clusters
  - Points belong to "nearest" cluster
    - Initial short phase of cluster estimation
    - occasional combining or splitting of clusters
    - possible unassignment of outliers (points too far from any of the current clusters).

Hierarchical Clustering

Hierarchical Agglomerative (bottom up) Clustering

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

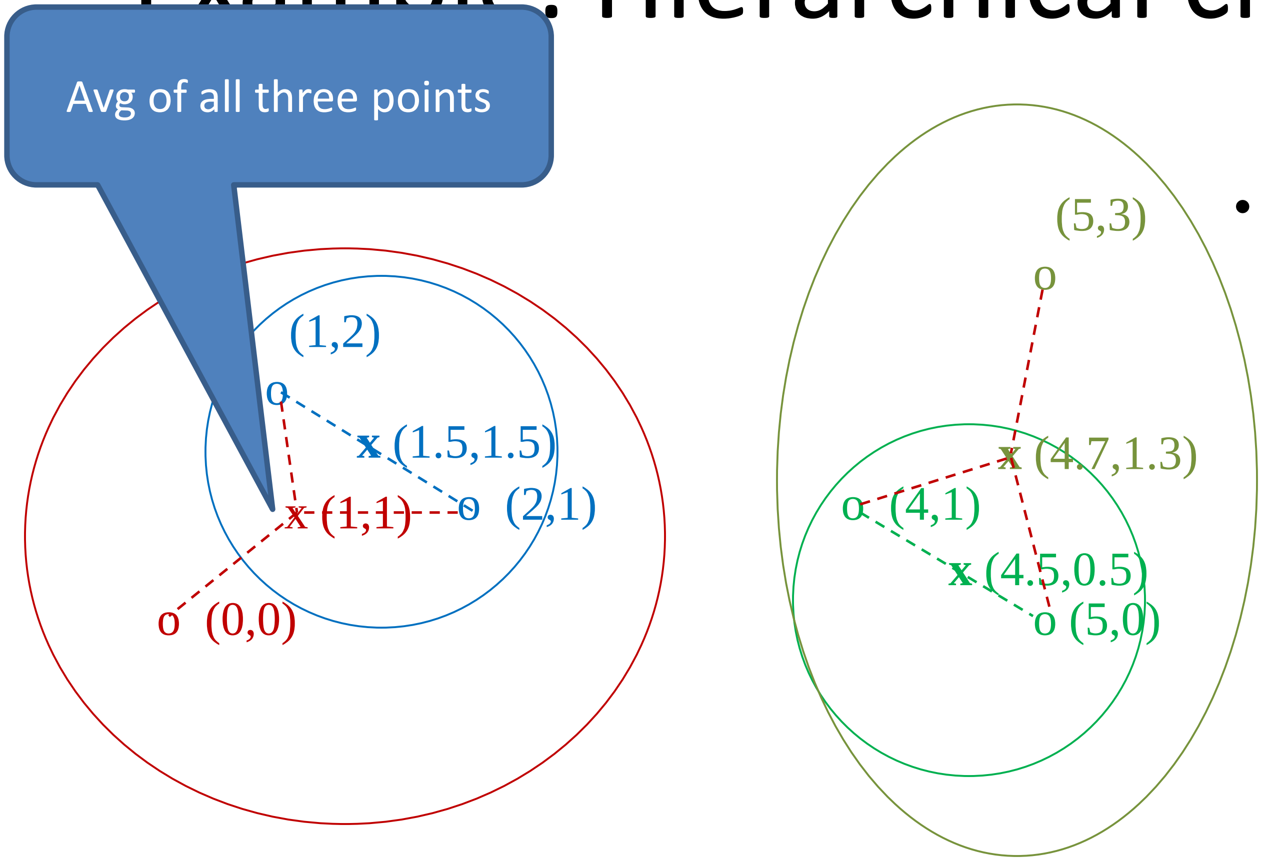


- **Three key questions:**
  - **1)** How do you represent a cluster of more than one point?
  - **2)** How do you determine the “nearness” of clusters?
  - **3)** When to stop combining clusters?

Hierarchical Clustering

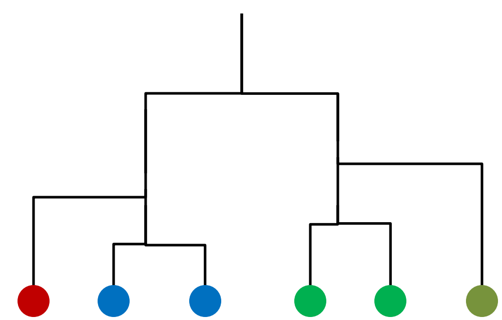
- **Key operation: Repeatedly combine two nearest clusters**
- **(1) How to represent a cluster of many points?**
  - **Key problem:** As you merge clusters, how do you represent the "location" of each cluster, to tell which pair of clusters is closest?
  - **Any ideas?**
- **Euclidean case:** each cluster 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



- **Q: How long should we continue merging?**
  - Even if we merge everything – dendogram will tell us the hierarchy.
  - E.g., Family tree in animal spiecies.

Data:
o … data point
x … centroid



Dendrogram

Non-Euclidean Case

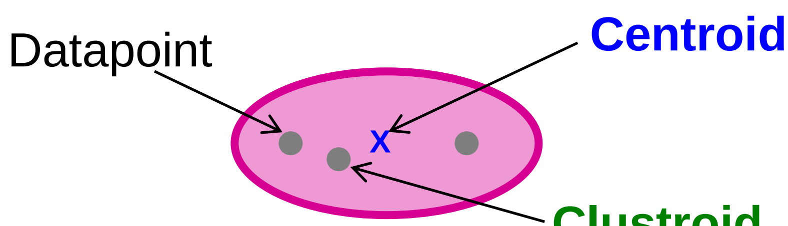
What about the Non-Euclidean case?

- **Any ideas of the problems with regard to previous example?**
  - Might not be possible to do averaging (E.g., "Edit distance" between to words)
- The only "locations" we can talk about are the points themselves
  - i.e., there is no "average" of two 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" (any ideas?):**
  - 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?**
  - Why “nearness” of previous approaches might not be good?
    - We lose a lot of information!
    - Any ideas how to fix it?
  - **Approach 2:**
    - **Intercluster distance** = minimum of the distances between any two points, one from each cluster
  - **Approach 3:**
    - Merge clusters whose *union* is most cohesive
      - Pick a notion of “**cohesion**” of clusters, *e.g.*, maximum distance from the clustroid

Termination Condition

- **(3) When do you stop combining clusters?**
  - Ideas?
- **Approach 1**: Pick a number *k* upfront, and stop once you have *k* clusters
  - Issues?
  - You have to know your data beforehand! Data should naturally fall in to k classes.
    - E.g., astronomy: galaxies and quasars,
- **Approach 2:** we don't know the number of clusters. We stop once we start getting "bad" clusters
  - Clusters with low "cohesion".

Cohesion

- **Approach 2.1:** Use the **diameter** of the merged cluster = maximum distance between points in the cluster
  - E.g.: Terminate clustering when diameter exceeds some threshold
- **Approach 2.2:** Use the **radius** as max distance of a point from centroid (or clustroid)
- **Approach 2.3:** Use the **average distance** between points in the cluster
- **Approach 2.4:** Use a **density-based approach**
  - Density=number of points per unit volume
    - E.g., divide number of points in the cluster by diameter or radius

Implementation

- **Naïve implementation of hierarchical clustering:**
  - At each step, compute pairwise distances between all pairs of clusters, then merge
    - *What's the complexity?*
    - O(N^3)
      - Might be up to N number of clusters and for each of the pair of clusters one has to compute pairwise distances between the points in these clusters O(NxN), and we might need to do O(N) mergers

- 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
  - Addressing scalability issue: k-means family of algorithms

k-means clustering

Point-assignment class of algorithms

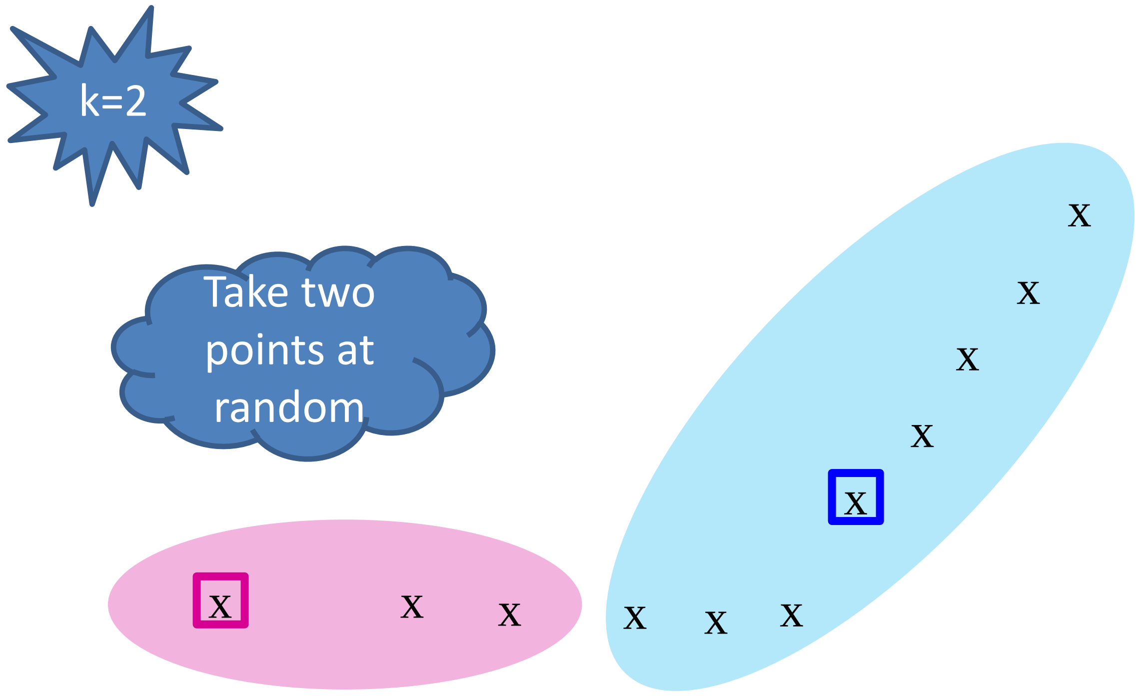
k–means Algorithm(s)

- Assumes **Euclidean** space/distance
- Start by picking ***k*,** the number of clusters
  - For now let's assume ***k*** is given
- Initialize clusters by picking one point per cluster
  - **E.g.,** Pick ***k*** points at random
    - Will discuss later in detail how to pick the initial 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 points move between clusters
- **Repeat 2 and 3 until convergence**
  - **Convergence:** Points don't move between clusters and centroids stabilize

Example: Assigning Clusters

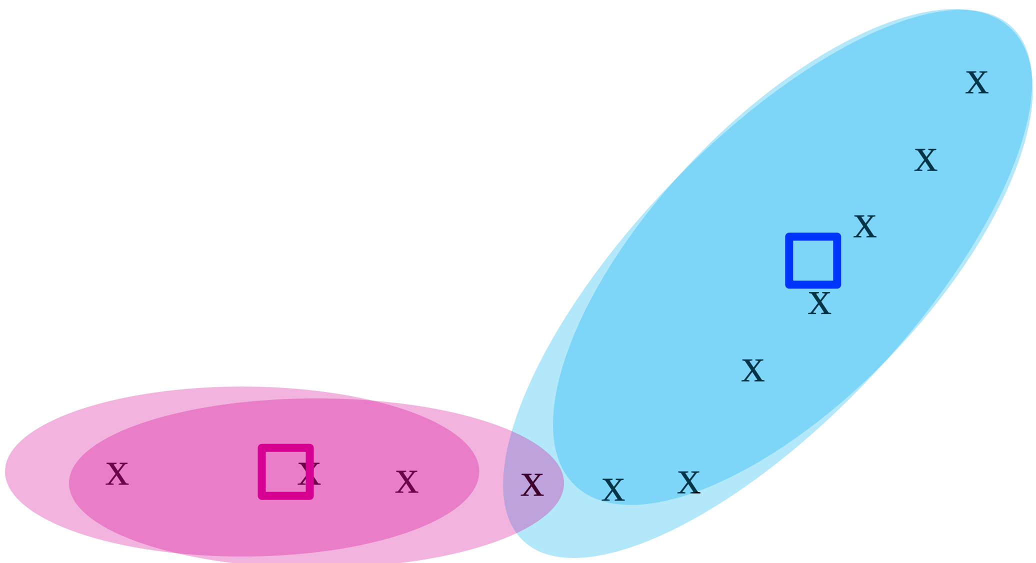


x … data point

… centroid

Clusters after round 1

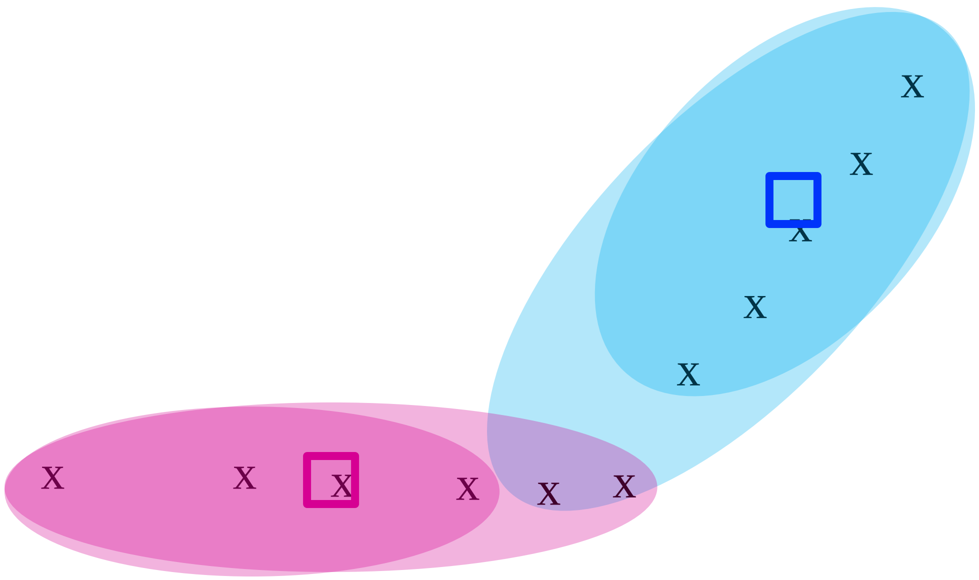
Example: Assigning Clusters



x … data point
… centroid

Clusters after round 2

Example: Assigning Clusters



x … data point
… centroid

Clusters at the end

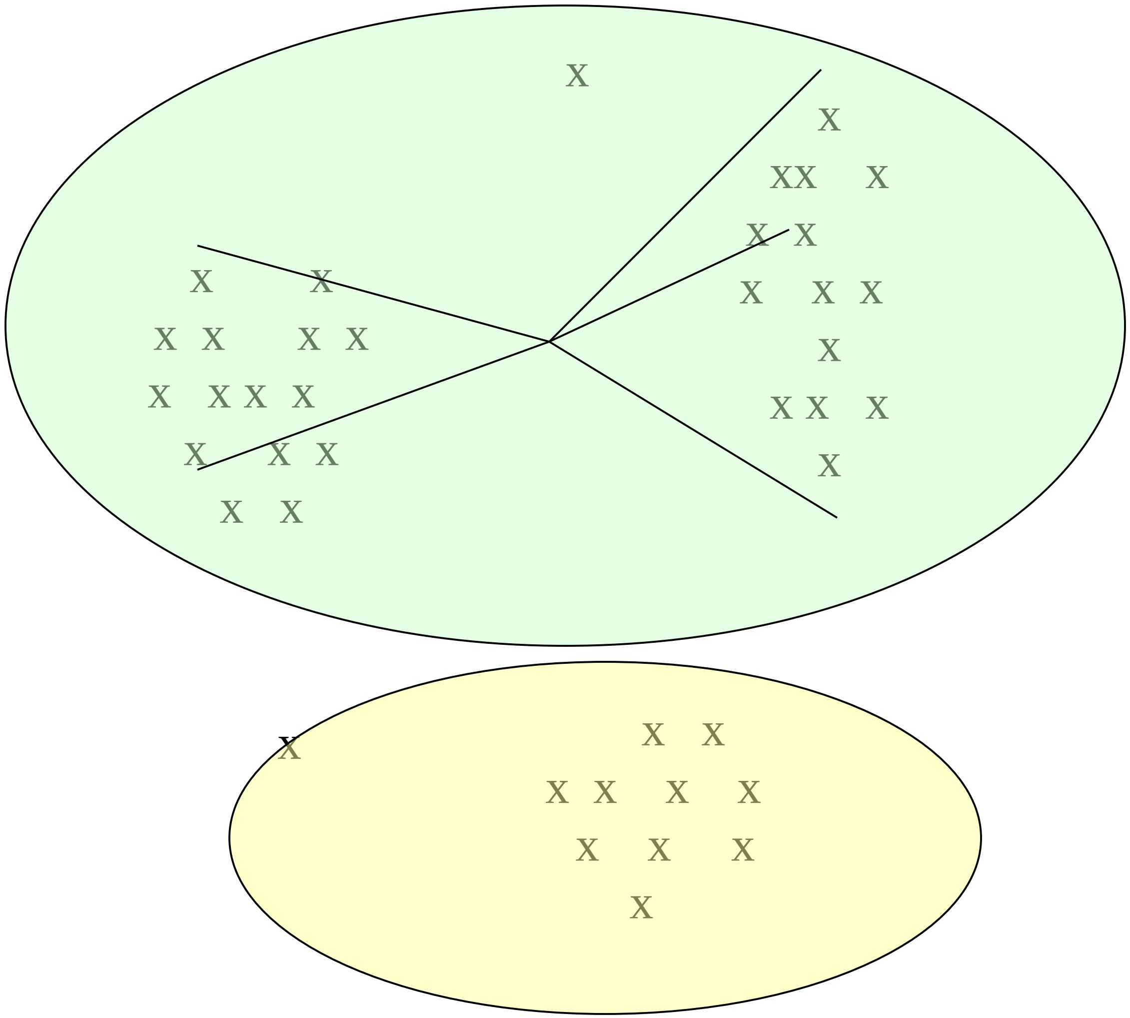
Getting the *k* right

How to select *k (any ideas)*?

- Try different **k**, looking at the change in the average distance to centroid as **k** increases
  - *What do you expect to happen to the avg. distance when k increases?*

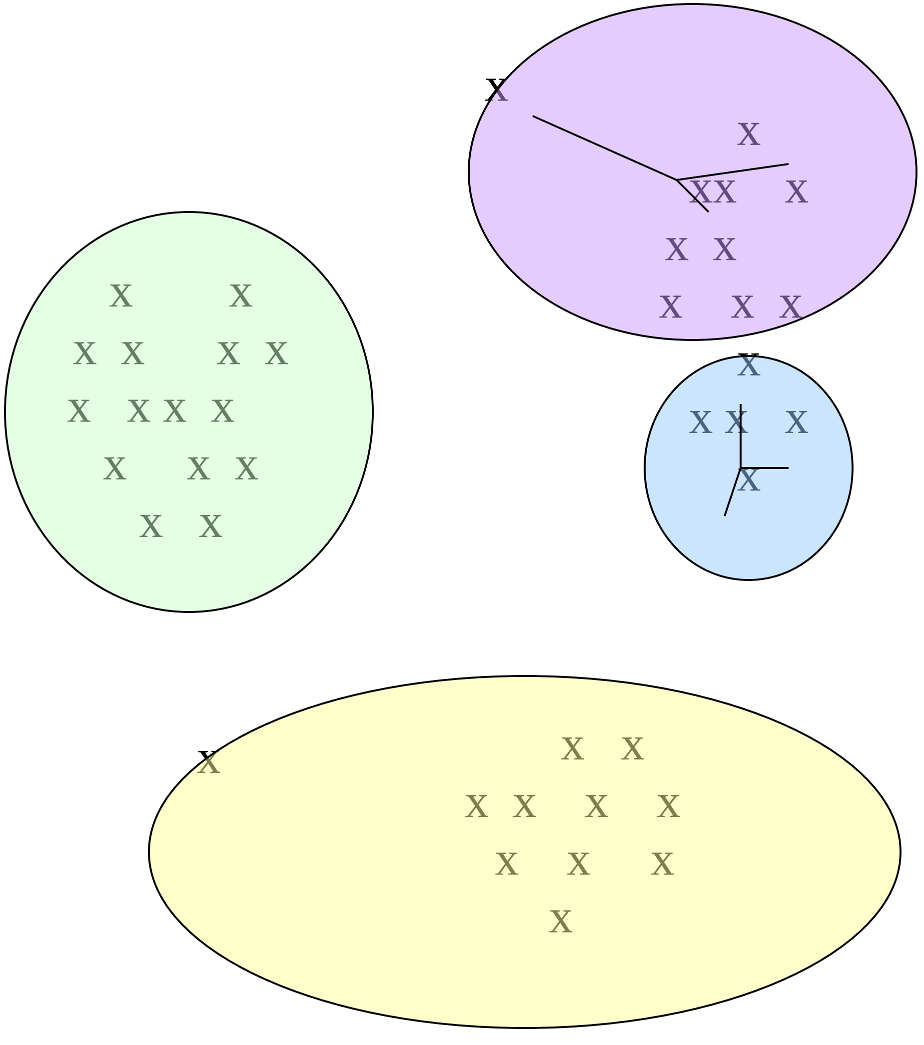
Example: Picking *k*

Too few;
many long distances to centroid.



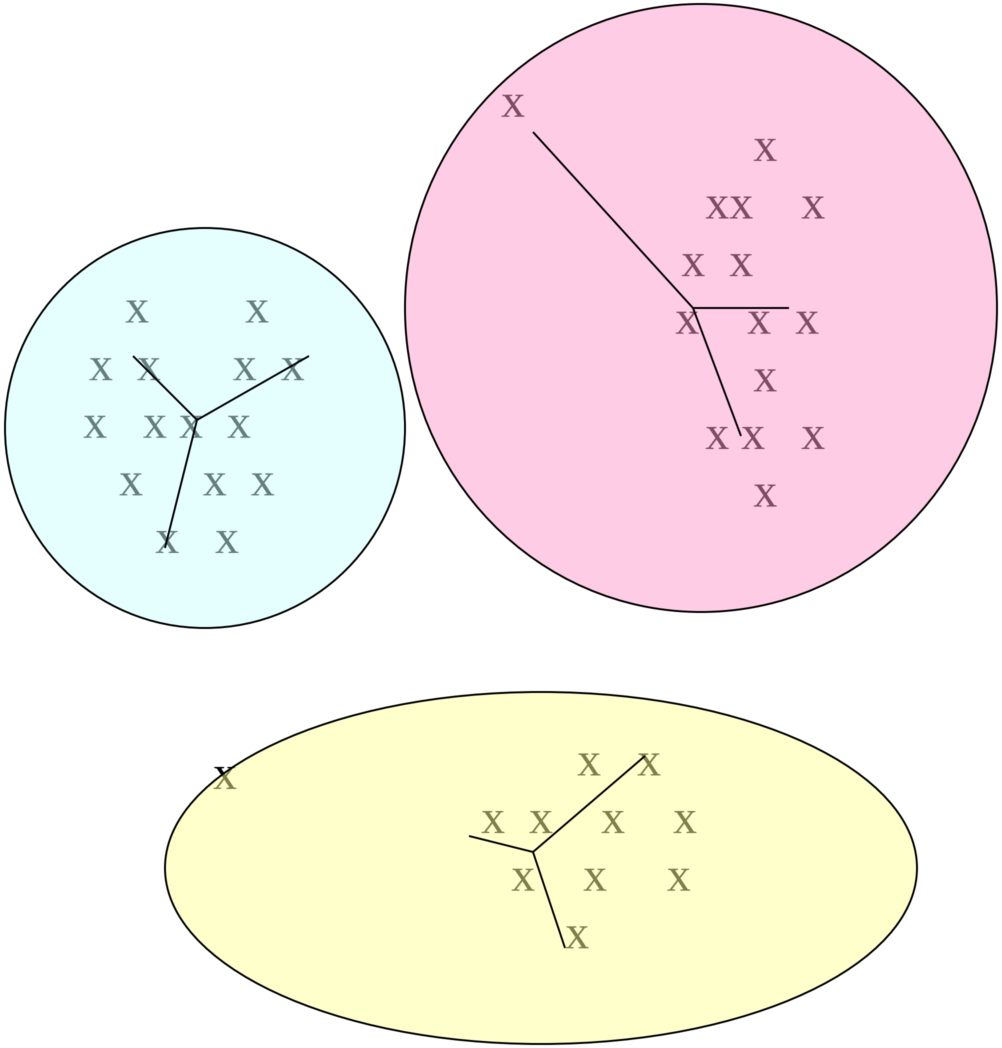
Example: Picking *k*

Too many;
little improvement in average distance.



Example: Picking *k*

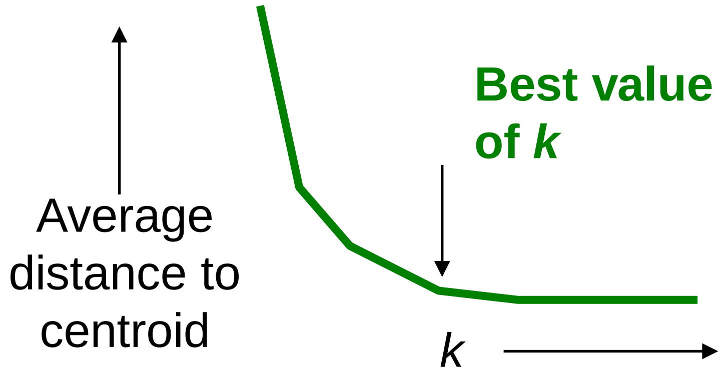
Just right;
distances
rather short.



Getting the *k* right

How to select *k*?

- Try different **k**, looking at the change in the average distance to centroid as **k** increases
  - *What do you expect to happen to the avg. distance when k increases?*
- Average falls rapidly until right **k**, then changes little



Picking the initial *k* points

- **What could be the issues? Any ideas?**
  - What if we pick all points that are in the same natural cluster?
  - Or if the initial points are outliers?
- The final clustering depends on the initial picking!
- **How to solve it? Any Ideas ?**

Picking the initial k points (cont.)

- **Approach 1: Sampling**
  - Cluster a sample of the data using hierarchical clustering to obtain k clusters
  - Pick a point from each cluster (e.g., point closest to centroid)
- **Approach 2: Pick "dispersed" set of points**
  - Pick first point at random
  - Pick the next point to be the one whose min. distance from the selected points is as large as possible
  - Repeat until we have k points.

Complexity

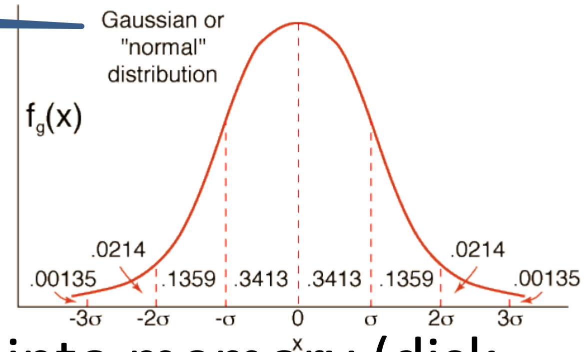
- On *each round* we have to examine *each point* once to find closest centroid
- Each round is O(kN) for N points and k clusters
  - That's not bad – linear to N.
  - But what about the number of convergence rounds?
    - Could be really large! (no theoretical limit)
- Can we cluster in a *single pass* over the data?

The BFR Algorithm

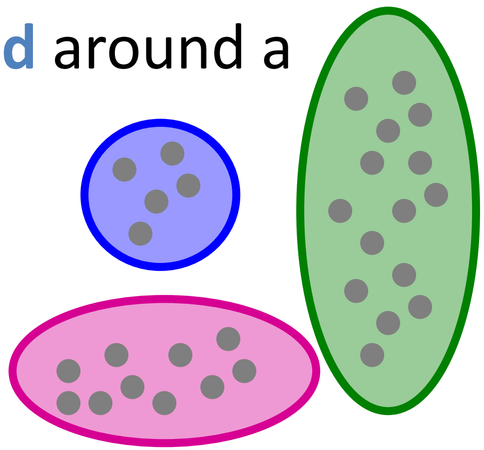
Extension of *k*-means to large data

Described by **mean** and **standard deviation**

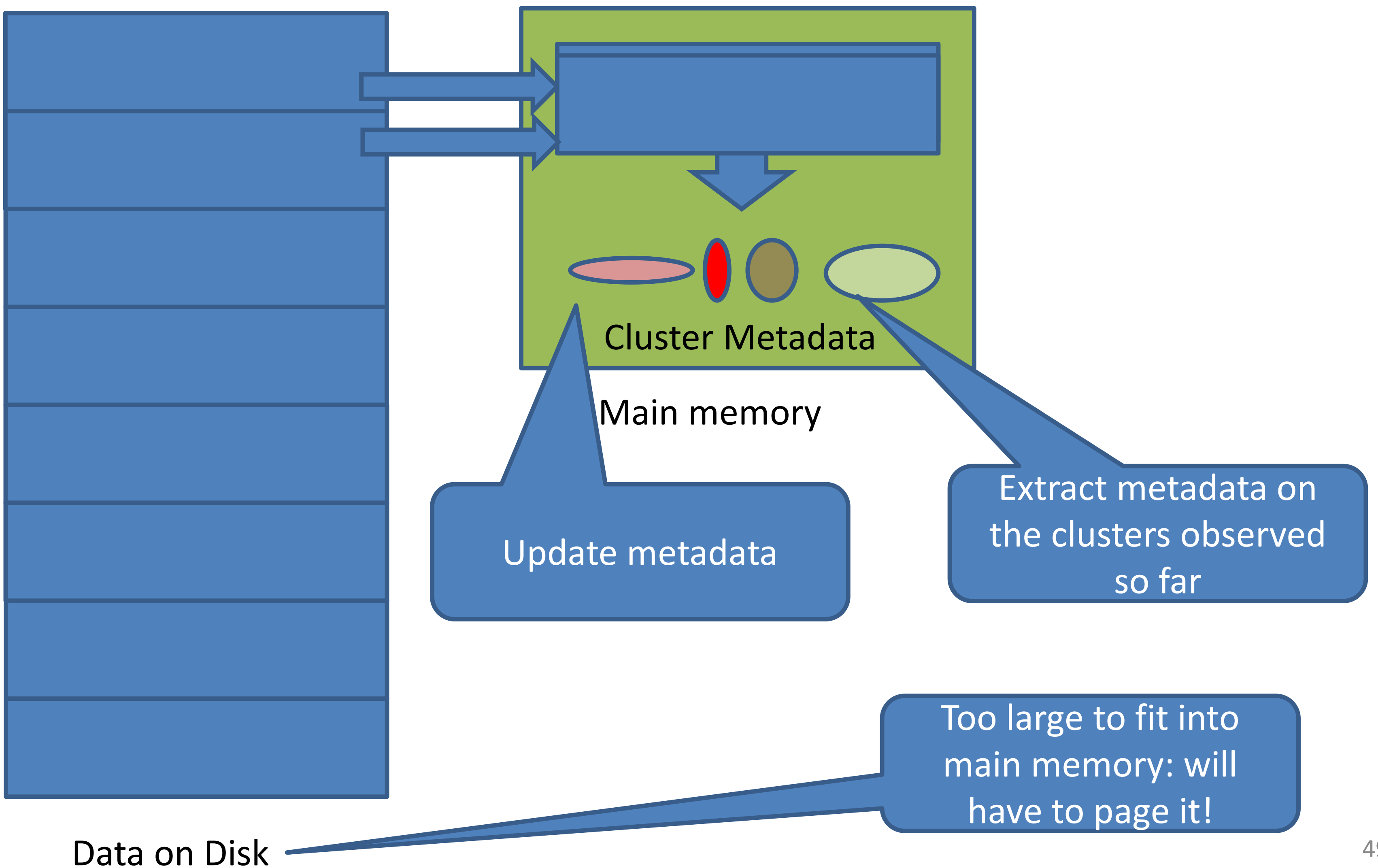
BFR Algorithm



- **BFR** [Bradley-Fayyad-Reina] is a variant of *k*-means designed to handle **very large** data sets that do not fit into memory (disk-resident)
- **However, the algo has very strong assumption:**
- **Assumes** that clusters are **normally distributed** around a centroid in a Euclidean space
  - Each dimension has its **own mean** and its **own standard deviation**
    - Standard deviations in different dimensions may vary
    - Clusters are axis-aligned ellipses
  - Can quantify the likelihood of finding a point in the cluster at a given distance from the centroid along each dimension
- **Efficient way to summarize clusters** (want memory required O(clusters) and not O(data))



BFR Algorithm: Overview



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:

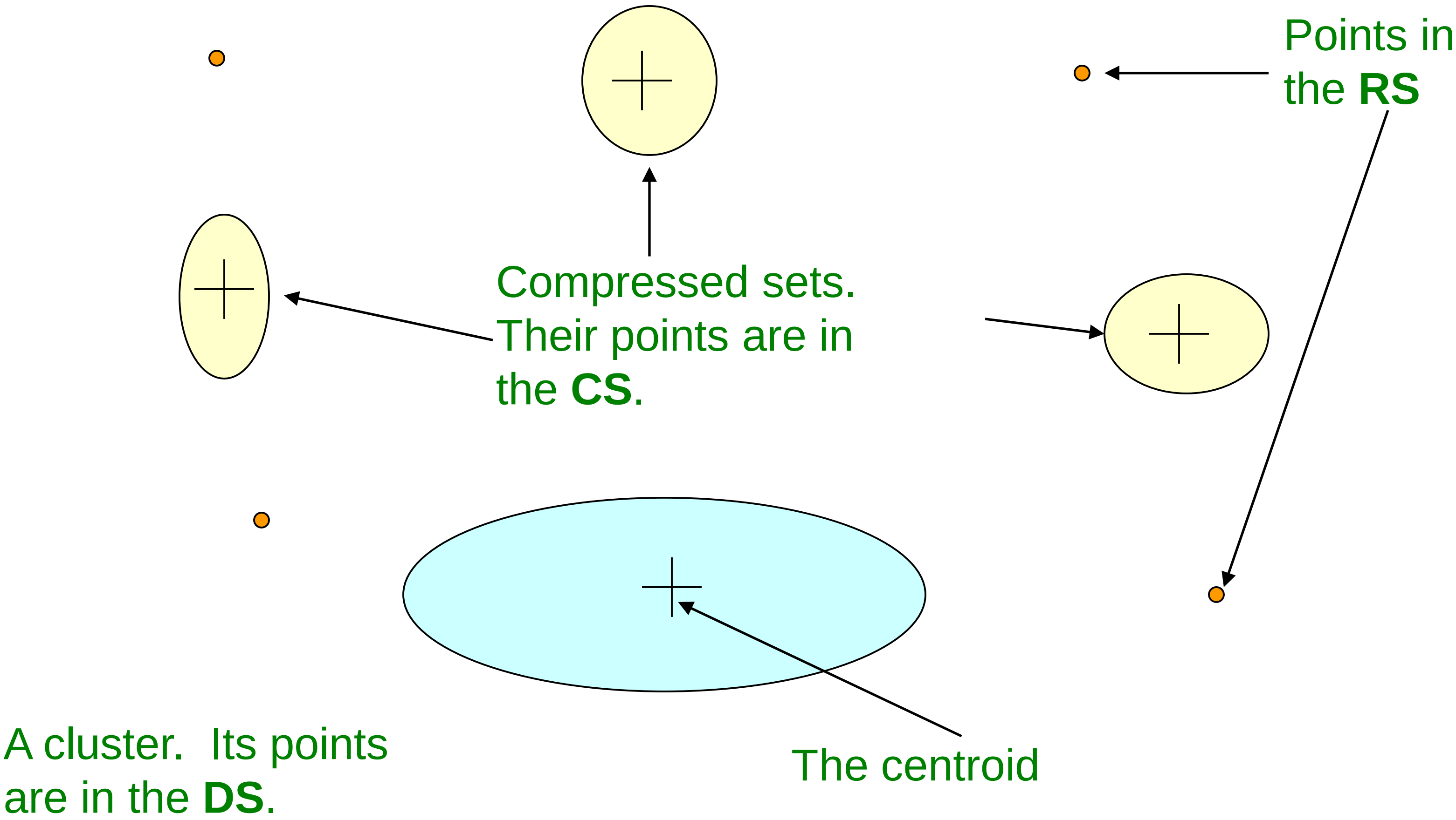
- **Discard set (DS):**
  - Points close enough to a centroid to be summarized
- **Compression set (CS):**
  - Groups of points that are close together but not close to any existing centroid
  - These points are summarized, but not assigned to a cluster
- **Retained set (RS):**
  - Isolated points waiting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

We can throw away these points and keep only metadata

We can throw away these points and keep only metadata

We have to keep these points (data) in memory

BFR: “Galaxies” Picture



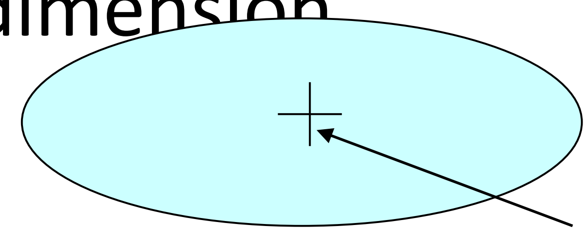
Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

For each cluster, the discard set (DS) is <u>summarized</u> by:

What info would you keep as a summary to define a cluster?

- 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



Summarizing Points: Co

Independent of number of points in the cluster!

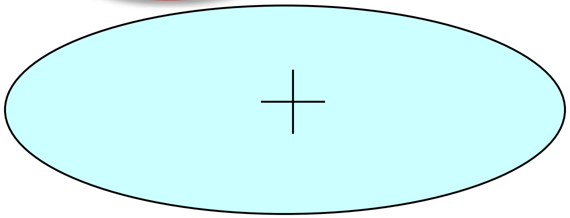
- **2d + 1** values represent any size cluster
  - d = number of dimensions
- **Average** in **each dimension** (**the centroid**) can be calculated as $\mathbf{SUM}_i / N$
  - $\mathbf{SUM}_i$ = i^{th} component of SUM

Variance is "mean of square minus square of mean"

- Variance of a cluster's discard set in dimension i is: $(\mathbf{SUMSQ}_i / N) - (\mathbf{SUM}_i / N)^2$
  - And **standard deviation** is t

Why don't we keep just **two values**: avg and std.dev?

- **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 Poi

We will discuss it in a moment: for now assume we are going to merge these 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

How do we do that?

- **2) DS set:** Adjust statistics of the clusters to account for the new points
  - Add ***N***s, ***SUM***s, ***SUMSQ***s
  - Easy to incrementally update!

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

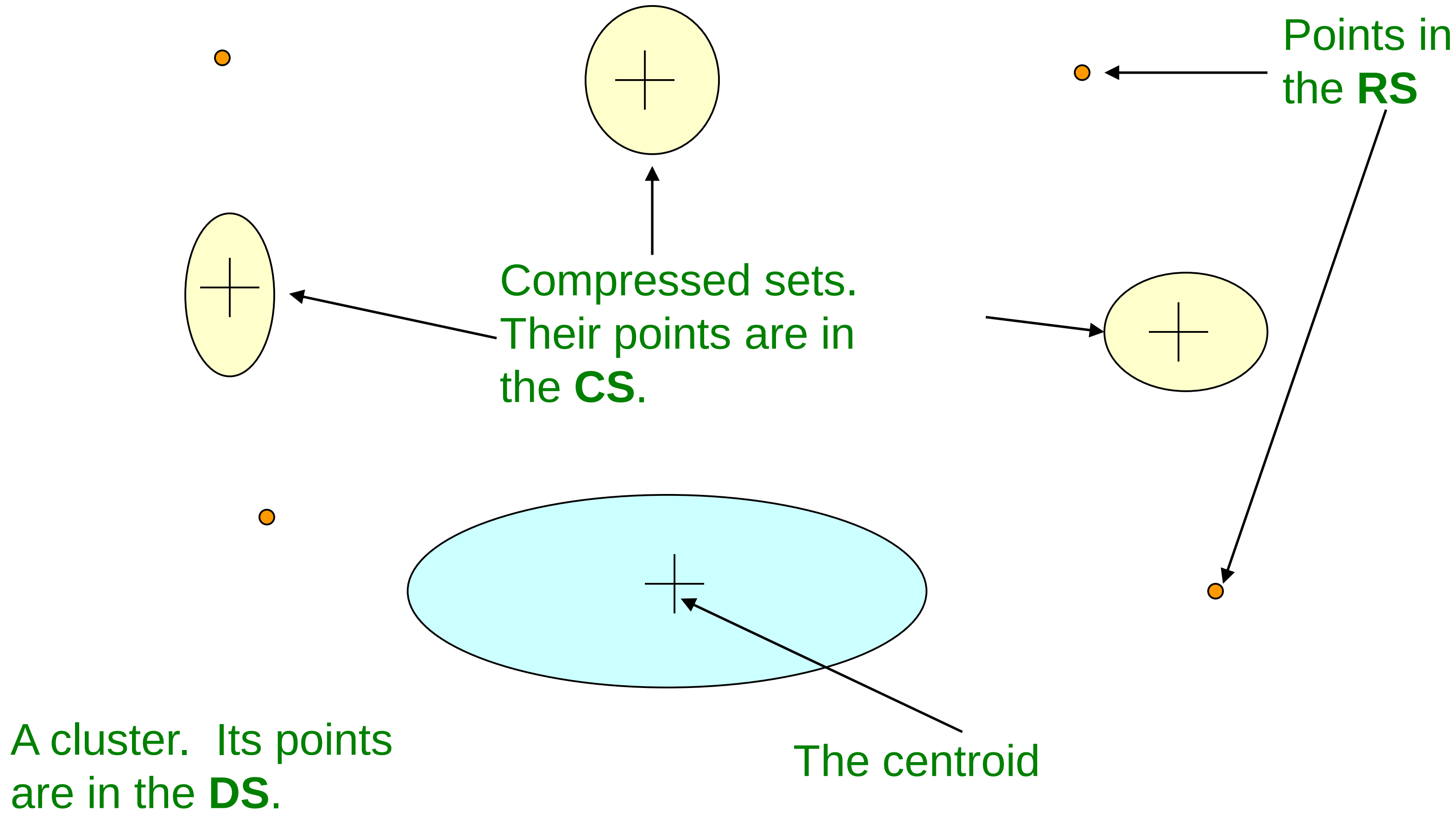
The “Memory-Load” of Points (cont.)

Processing the “Memory-Load” of points (2):

- **The remaining points are not close to any cluster**
- **3)** Use any main-memory clustering algorithm (e.g., k-means) to cluster the remaining points and the old **RS**
  - Clusters go to the **CS**; outlying points to the **RS**
- **4)** Consider merging compressed sets in the **CS**
  - We will discuss it in a moment...
- **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

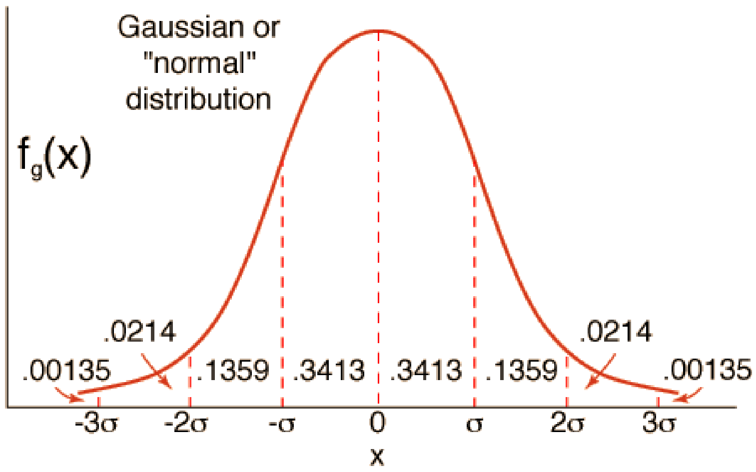
A few more details…

- Can we run the algo now? Do we need to agree on smth else? Any ideas?
- **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:**
  - The **Mahalanobis distance** is less than a threshold
  - Mahalanobis distance: is a likelihood of the point belonging to currently nearest centroid

Mahalanobis Distance (MD)

- **Normalized Euclidean distance from centroid**

- For a point $(x_1, \ldots, x_d)$ and Cluster C with centroid $(c_1, \ldots, c_d)$ and standard deviations $(\sigma_1, \ldots, \sigma_d)$
  1. Normalize in each dimension: $y_i = (x_i - c_i) / \sigma_i$
     1. Measures how many std.deviations away a point is from the centroid in that dimension
  2. Take sum of the squares of the y_i
  3. Take the square root

What happens if the point is one std. deviation away from the centroid in each dimension?

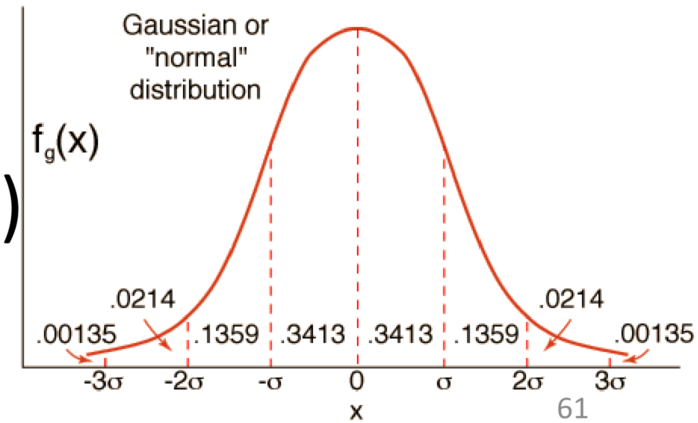
e: $d(x,c) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_i - c_i}{\sigma_i}\right)^2}$

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

Mahalanobis Distance

- If clusters are normally distributed in $\boldsymbol{d}$ dimensions, then after transformation, one standard deviation = $\sqrt{\boldsymbol{d}}$
  - i.e., 68% of the points of the cluster will have a Mahalanobis distance $\mathrm{MD} < \sqrt{\boldsymbol{d}}$
  - i.e., 99% of the points of the cluster will have a Mahalanobis distance $\mathrm{MD} < \mathbf{3}\sqrt{\boldsymbol{d}}$

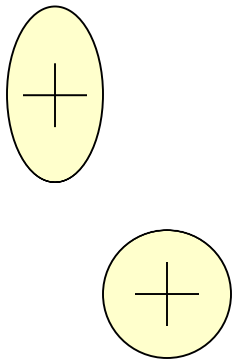
- Accept a point for a cluster if its M.D. is **<** some threshold, e.g. $\mathbf{3}\sqrt{\boldsymbol{d}}$ (3 standard deviations)



Should two CS clusters be combined?

Q2) Should two CS subclusters be combined? Any ideas?

- 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



- **There are other alternatives:**
  - Treat dimensions differently (e.g., some dimensions are “more important” than others), consider density