CLUSTERING

Yannis Kotidis

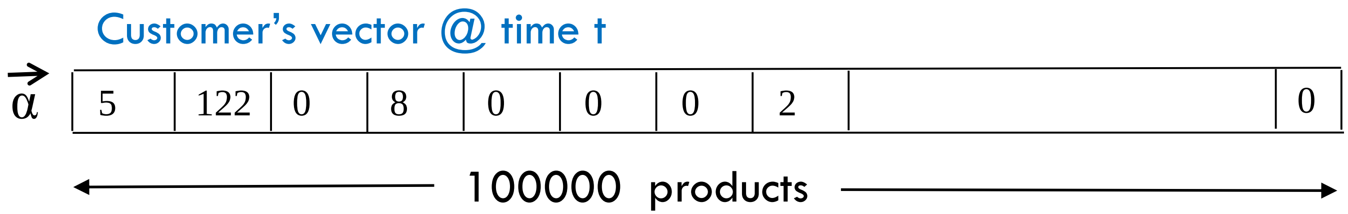
What is clustering: general idea

- Given a collection of data objects, put them into groups so that
  - members of each group are similar to each other (cohesion)
  - members of different groups are dissimilar (separation)

- Examples
  - Cluster together customers based on their purchases
    - Intuition: products explain customers habits
  - Cluster together documents that are on the same topic
    - Intuition: terms relate documents to topics

Before you start

- Choose a convenient representation
  - Example: treat your data objects as high-dim vectors/points
    - Customers represented as vectors, coords denote number of products they buy



  - Alternatively, represent a customer as a set (or bag) of products
    - Documents may also be represented as bags of words
- Choice depends on the data and the techniques used and will affect the outcome of the analysis

Need to quantify similarity

- Select an appropriate similarity/distance measure
  - Euclidian or cosine distance for customer vectors?
  - Jaccard similarity for baskets/sets/documents?

- Different distance measures lead to different cluster formations

Dimensionality curse

- In some application the number of dimensions is in the order of hundreds or thousands
  - Number of different products, customers, words etc

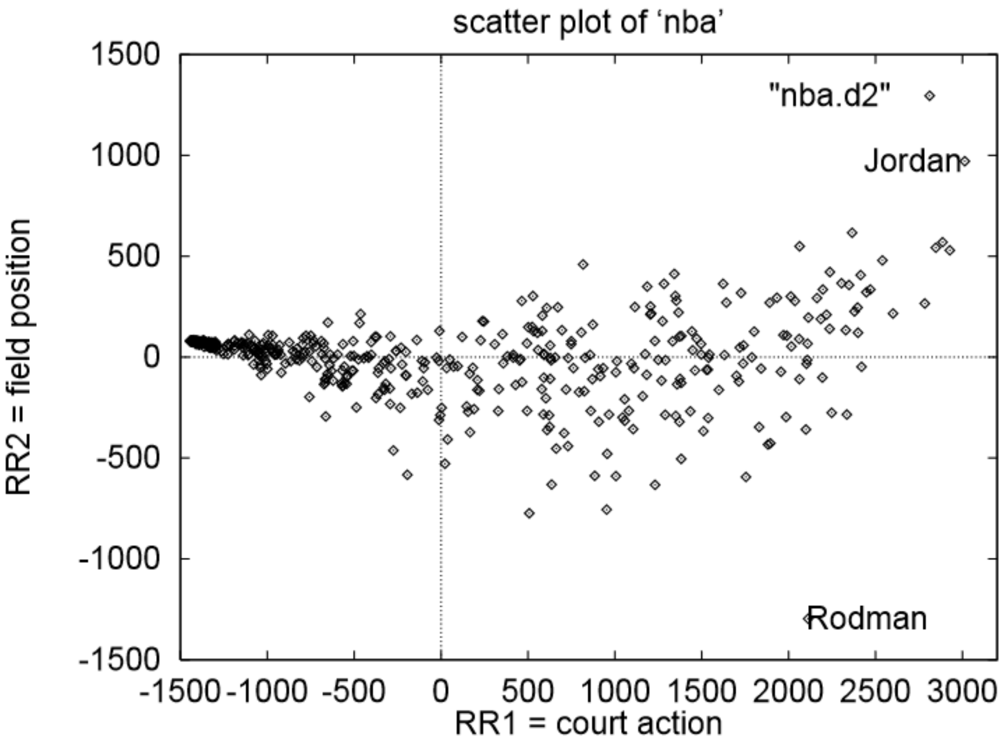
- High-dimensionality affects
  - Memory requirements, efficiency of computations
  - Quality of resulting clusters: it becomes harder to distinguish clusters
    - Also clusters are less meaningful

In high dimensions

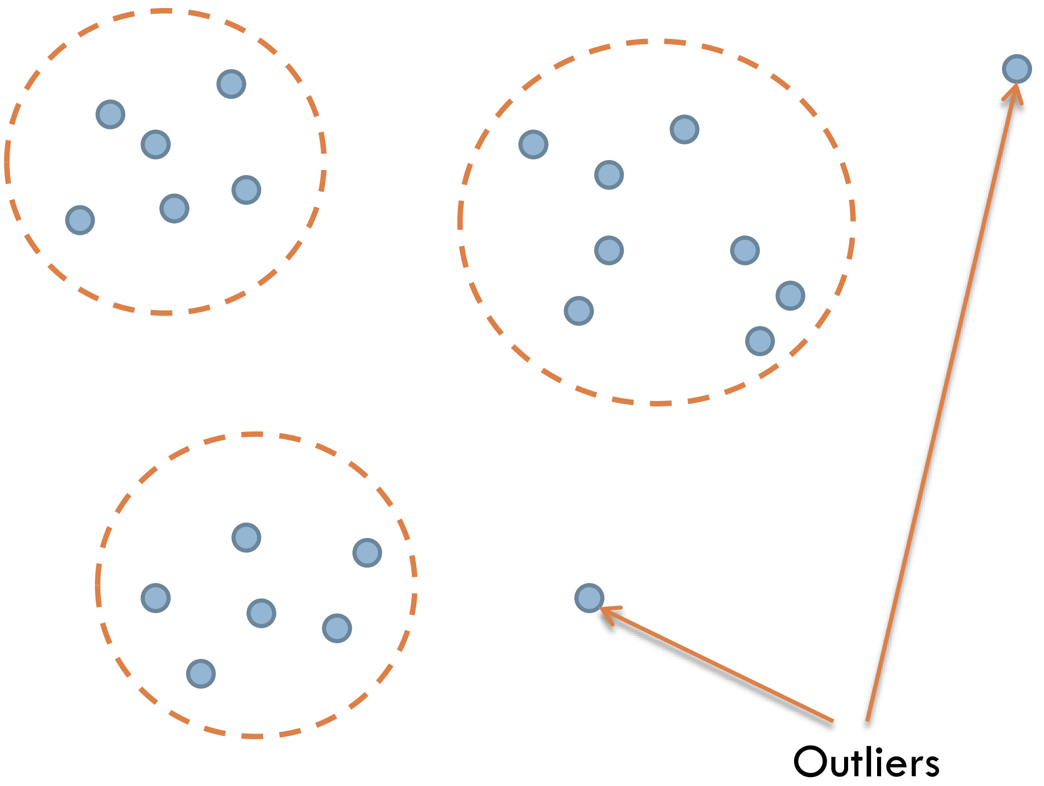
- Most pairs of points are at about the same distance from each other
- The distance to the nearest neighbor and the distance to the farthest neighbor tend to converge as dim→inf
- Nearest neighbor computations become harder and less meaningful

Dimensionality reduction/sub-space clustering

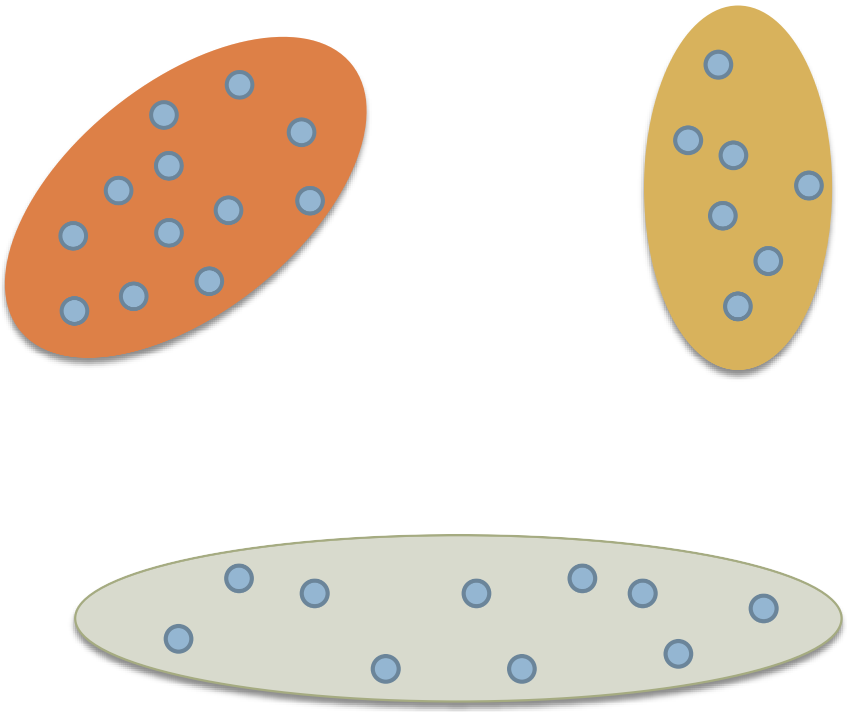
- Map points into lower-dimensionality spaces



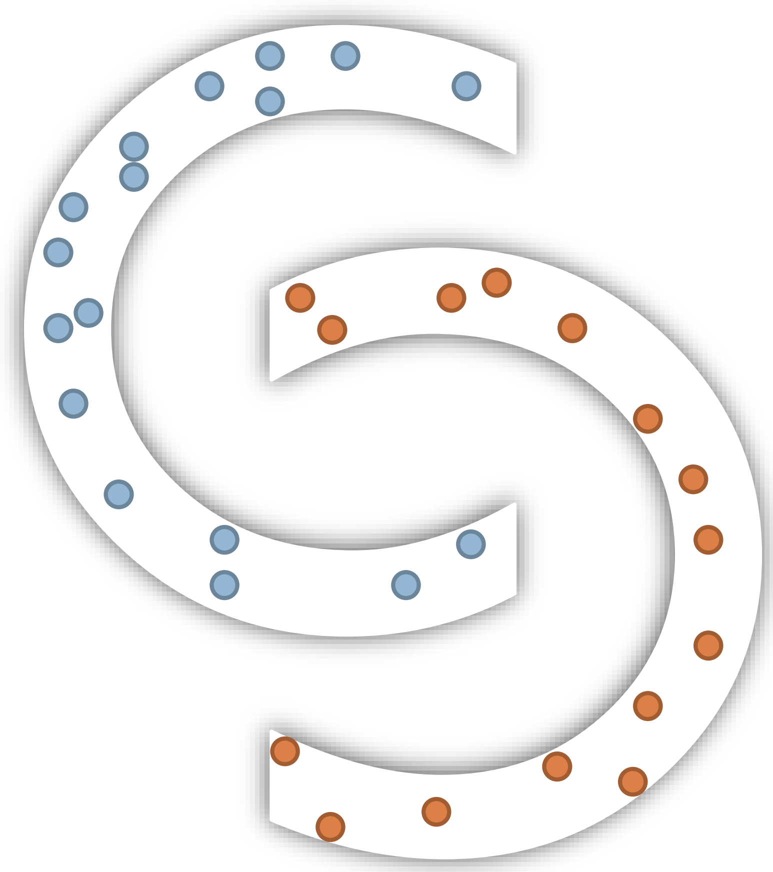
Clustering in two dimensions



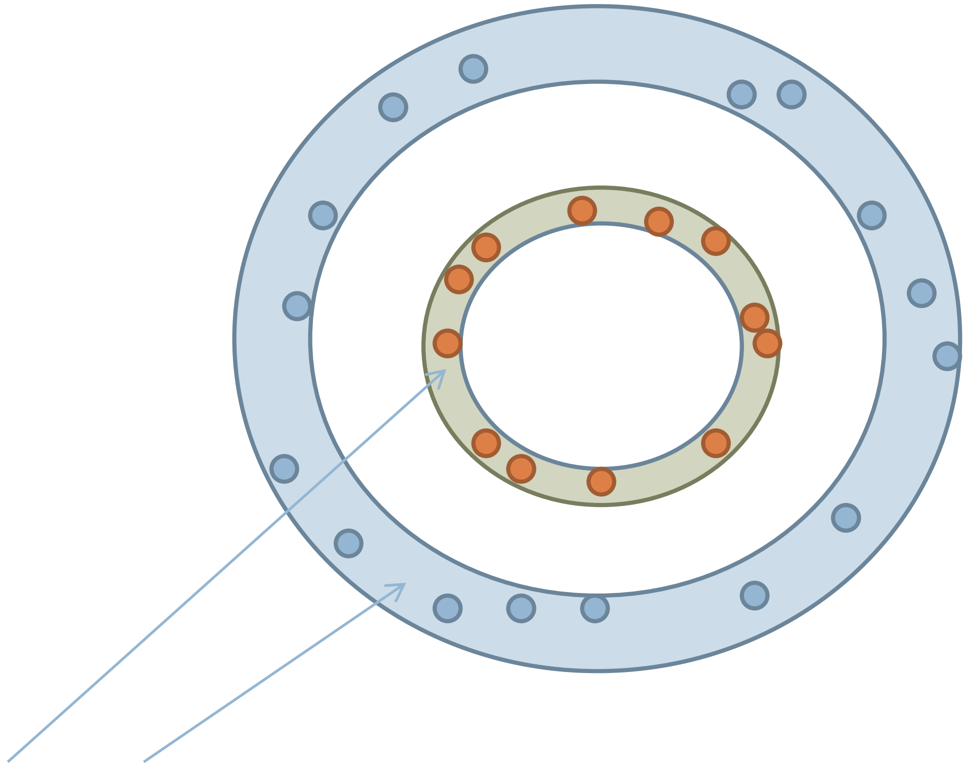
Elliptical shapes/rotated axes



Non-convex shapes



Clusters within clusters



What do they mean?

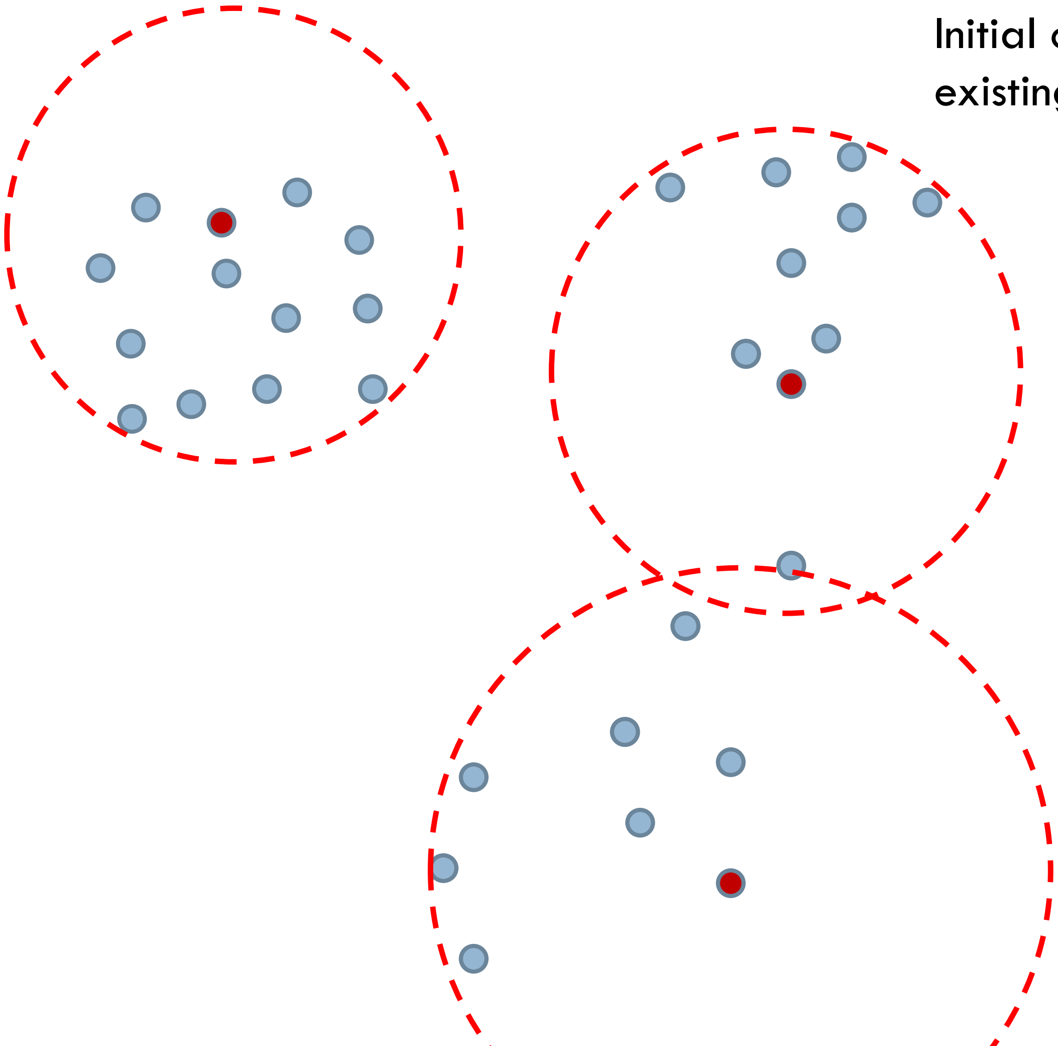
k-Means Algorithm

- Assume n points in the Euclidian space and a user-defined value of k=#clusters

1. Pick k points (centroids), one per cluster
2. Assign remaining points to closest centroid
3. In each cluster update location of its centroid
4. Reassign points, if necessary
5. Repeat steps 3-4 until clusters stabilize

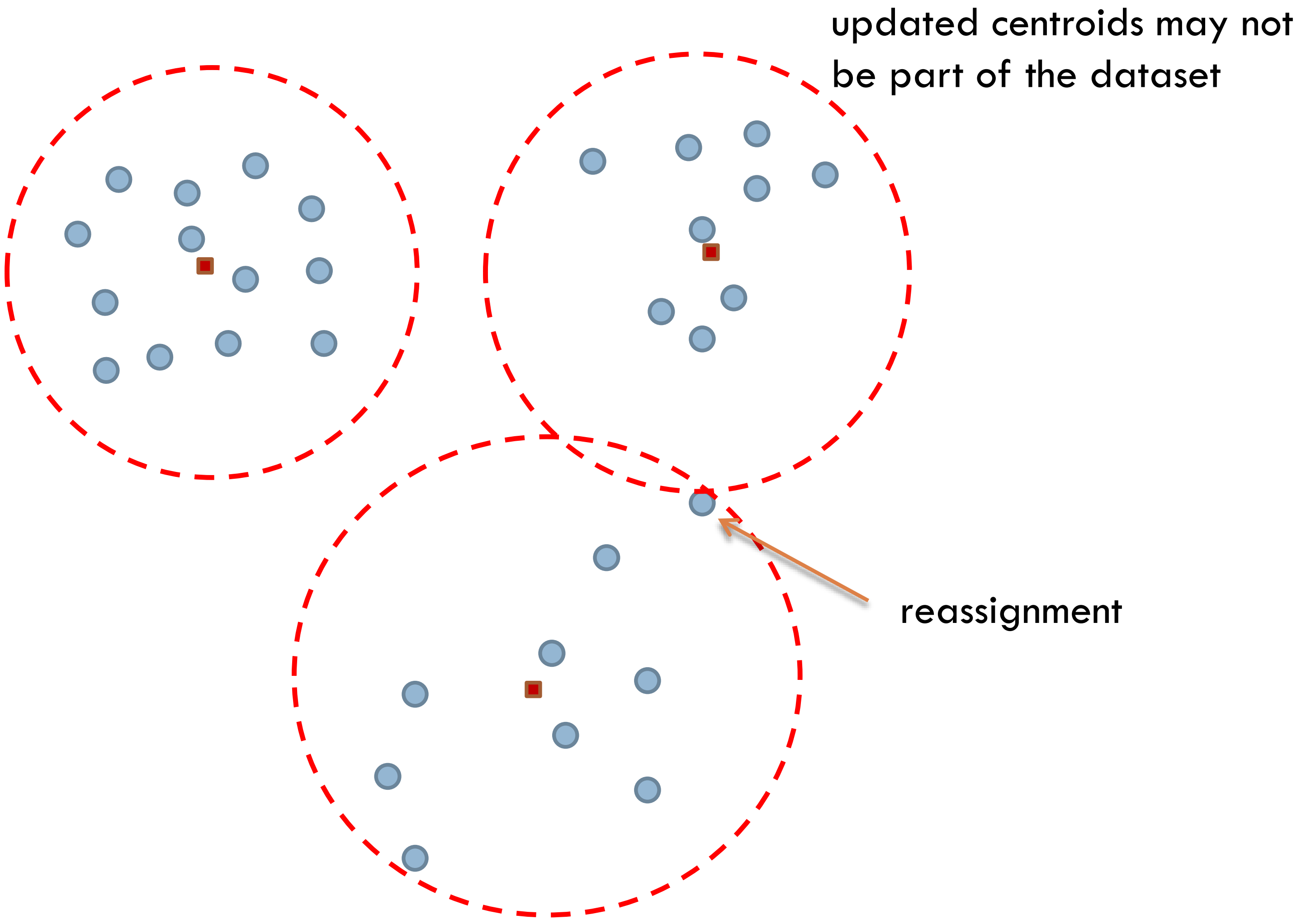
- k-Means seeks to minimize the sum of squared distances (thus the variance of the distances) from the centroids
  - the algorithm always converges to some (local) minimum solution

Example for k=3



Initial centroids are
existing dataset points

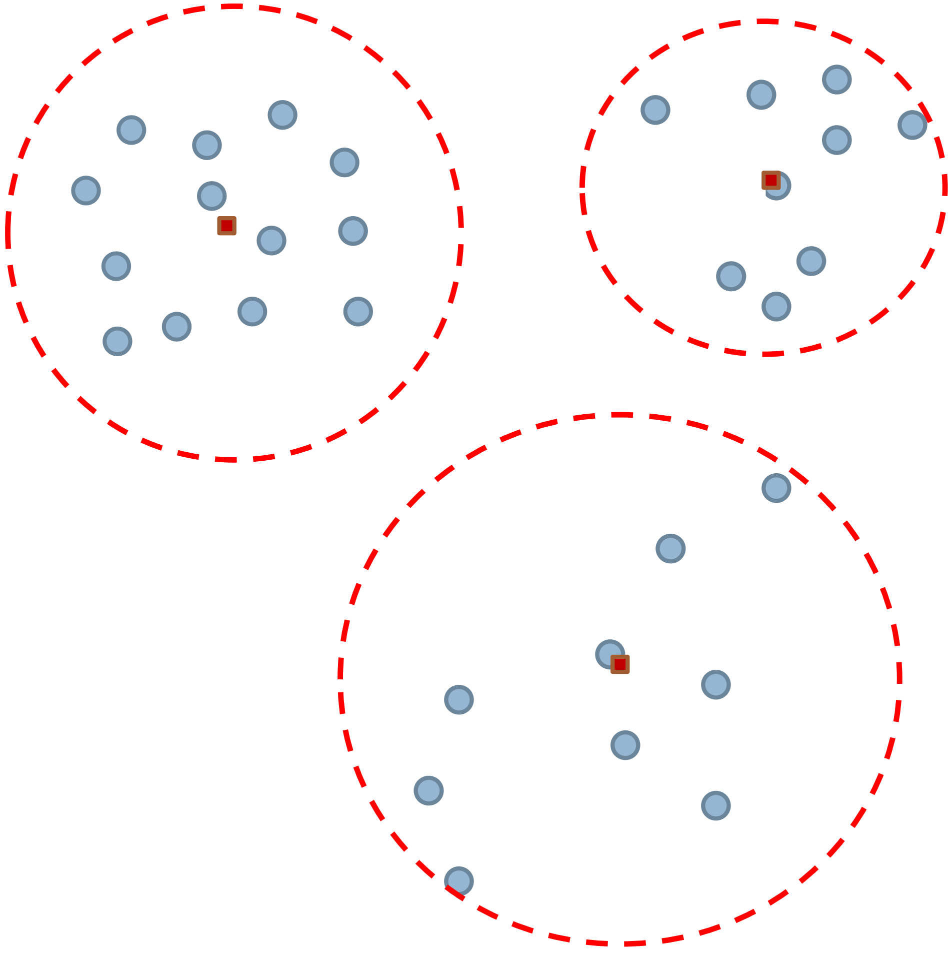
New centroids + reassignment



Performance considerations

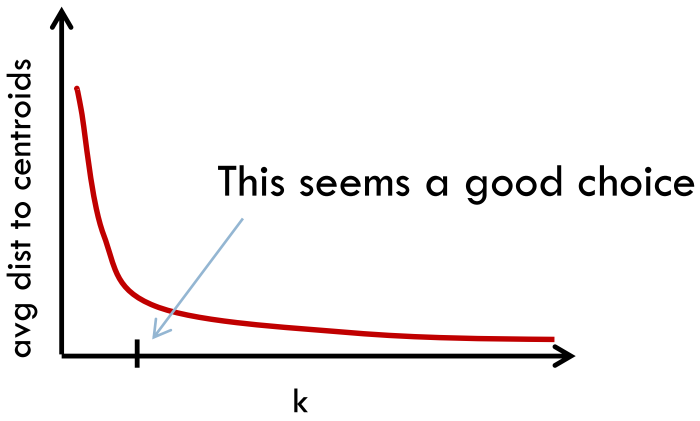
- Quality: initial selection of centroids affects cluster discovery
  - Intuition: pick points as further apart as possible
    - Pick first centroid c_1 at random
    - At step $i \leq k$, pick i^{th} centroid c_i so that the minimum distance to c_1, c_2,.. c_{i-1} is maximized
- Speed: assume m steps for convergence
  - Assume initial centroids are given
  - Each step takes O(k*N) time
  - O(k*m*N) complexity, what if m is large?

Final clusters

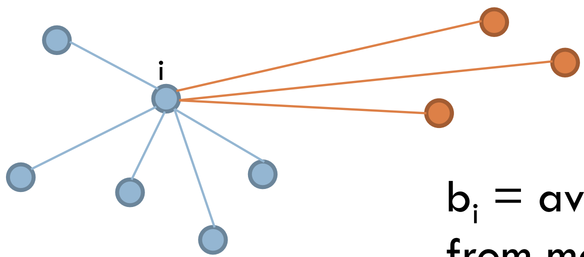


What is a good value for k?

- Small k: few large clusters with large intra-cluster distances
- Large k: many small clusters
- Solution: try different values of k
  - Plot average distance to centroids for different k



Silhouette Coefficient (e.g. combine cohesion and separation)



b_i = avg distance of i from members of another cluster (consider cluster that minimizes this value)

a_i = avg distance of i from members of its own cluster

$Silhouette_i = (b_i - a_i)/\max(a_i, b_i)$

Silhouette coefficient in [-1..+1] ⇒

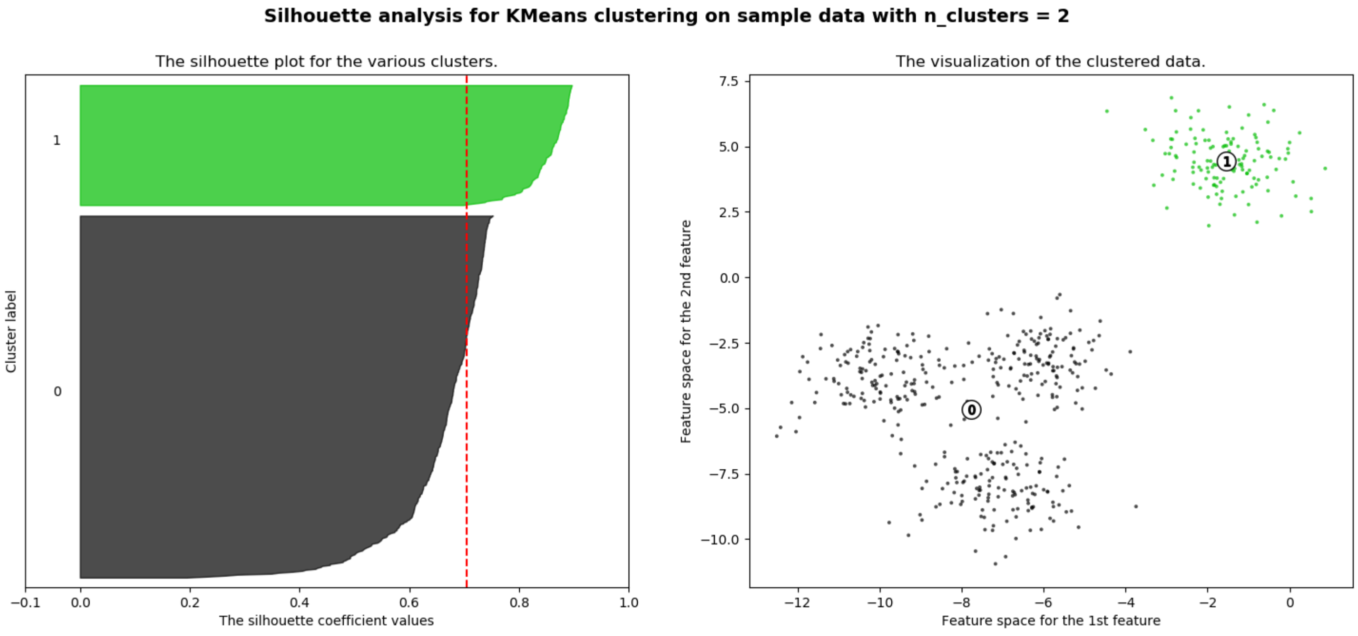
- <0 is really bad (wrong assignment)
- 0 means point is borderline
- close to 1 is best

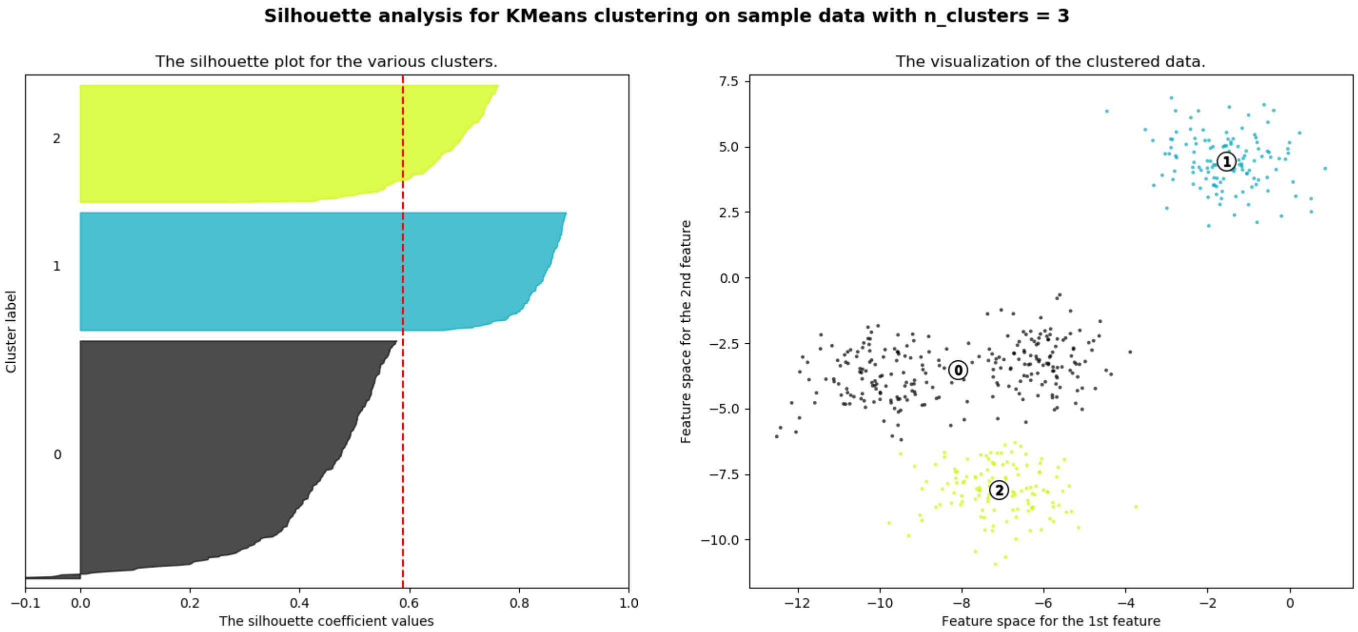
Silhouette of a cluster = avg silhouette of its points

Silhouette of a solution = avg silhouette of proposed clusters

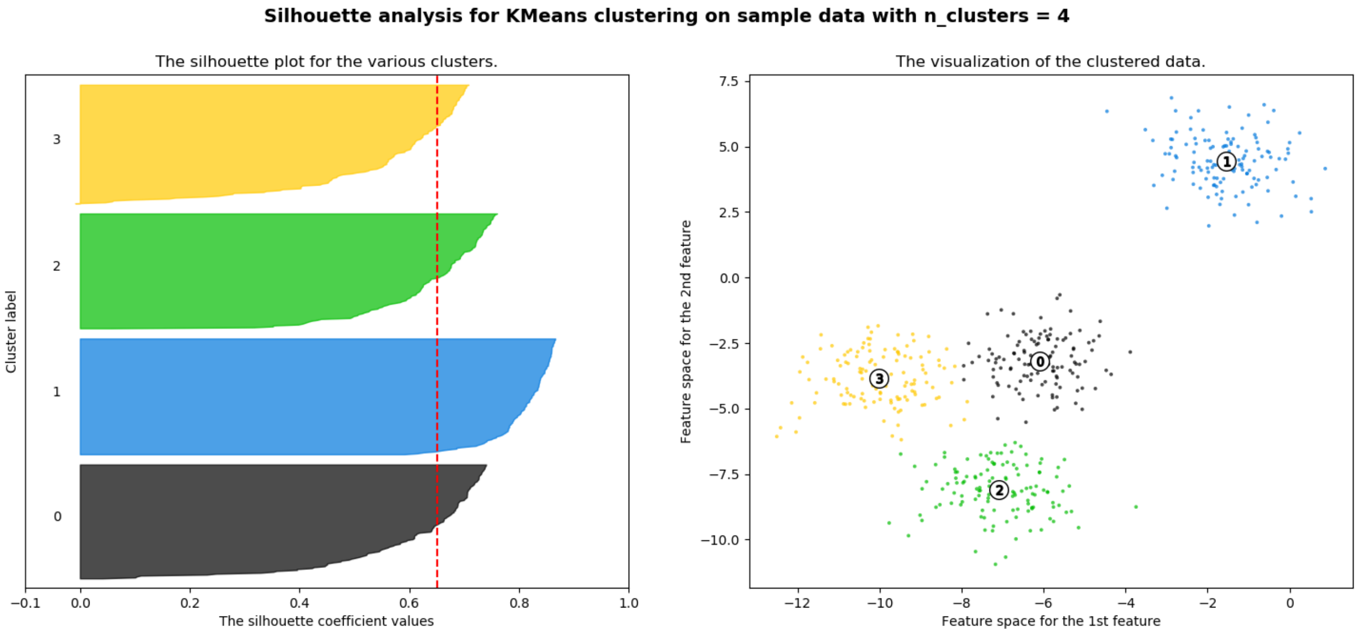
Look at the following online example (next slides)

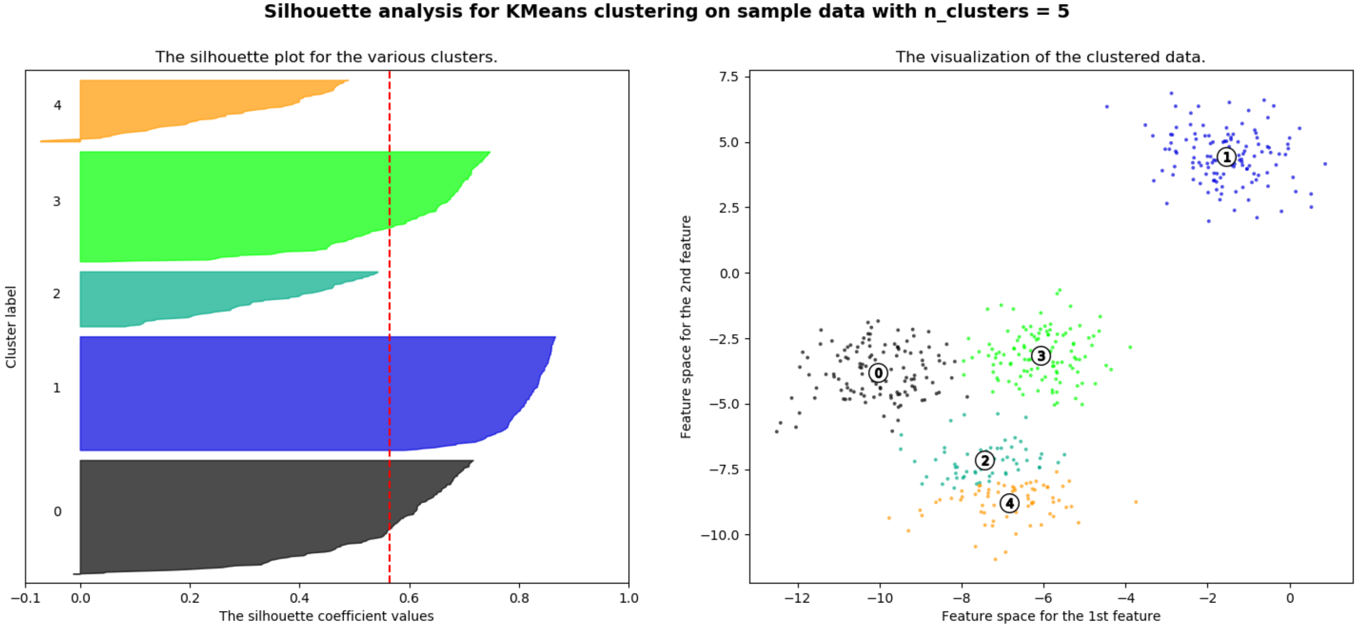
- http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html


Silhouette analysis for KMeans clustering on sample data with n_clusters = 2
The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature

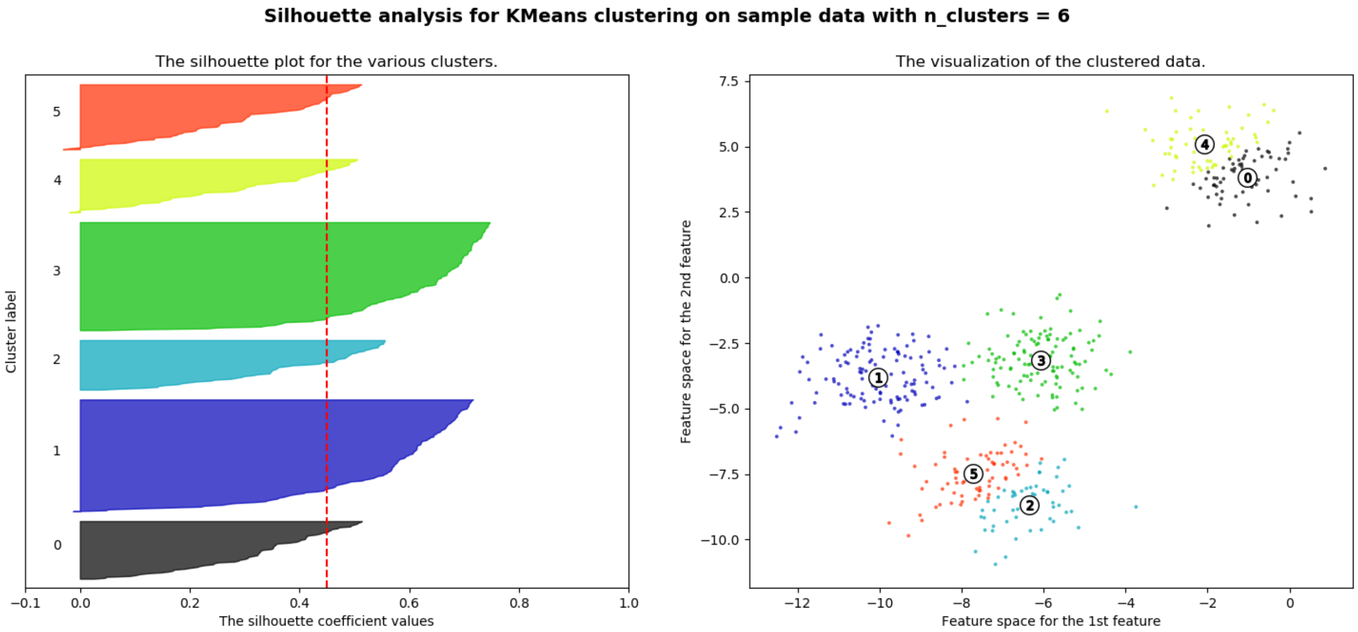


Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
Cluster label
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12
−10
−8
−6
−4
−2
0
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12
−10
−8
−6
−4
−2
0

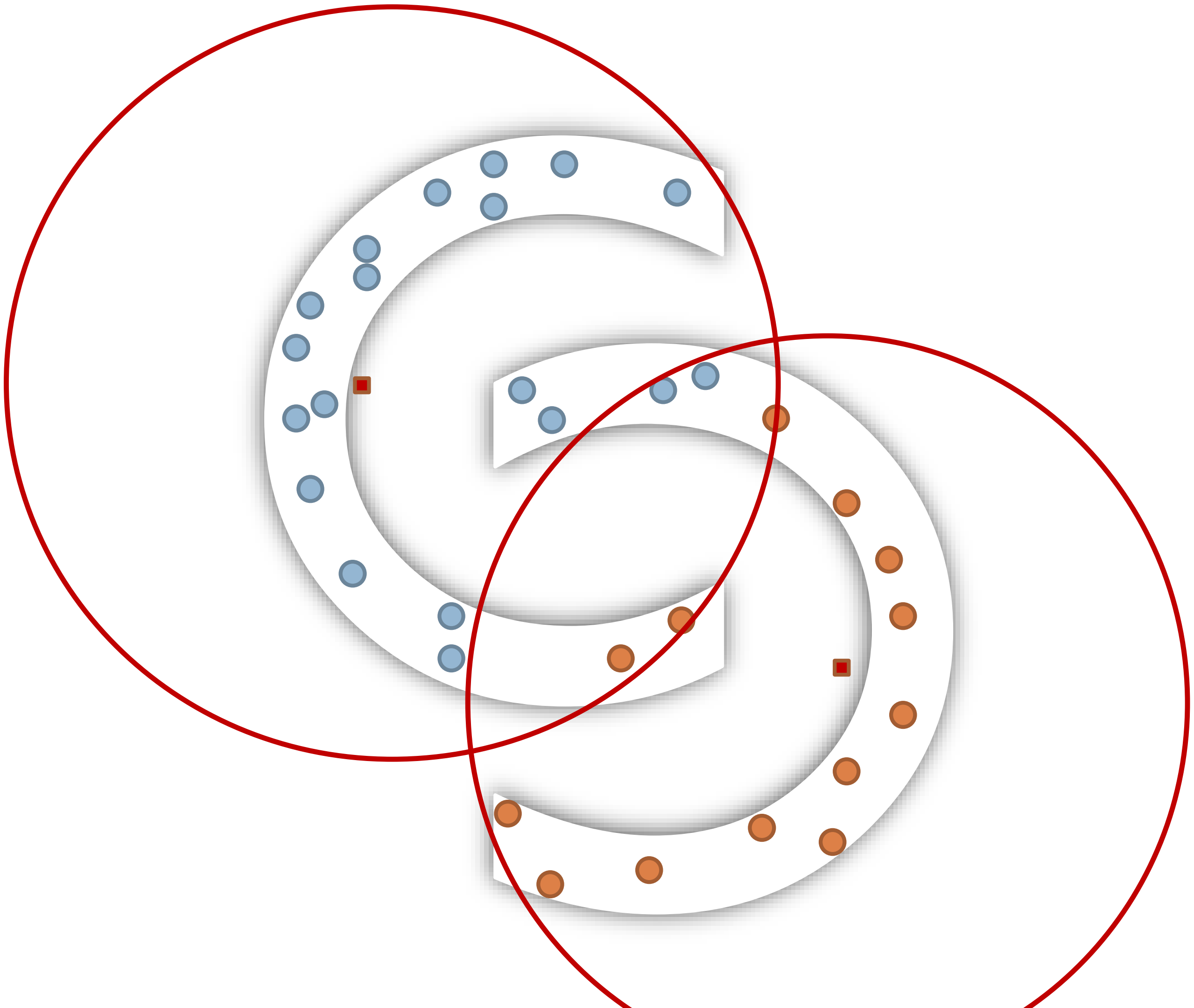


Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
The visualization of the clustered data.
Feature space for the 1st feature
Feature space for the 2nd feature
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12
−10
−8
−6
−4
−2
0



Silhouette analysis for KMeans clustering on sample data with n_clusters = 6
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature

Shape of clusters

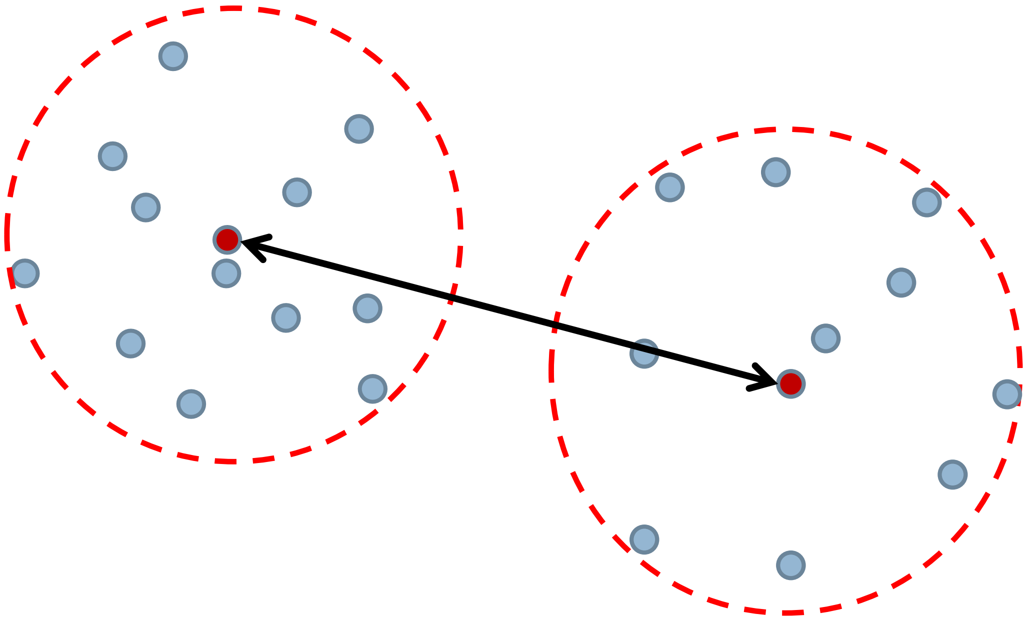


Hierarchical clustering

- Start assuming each point is a cluster
- Repeatedly merge clusters
  - Look for clusters that are "close"
  - Stop when resulting clusters are "bad"
    - Or use a pre-defined value k

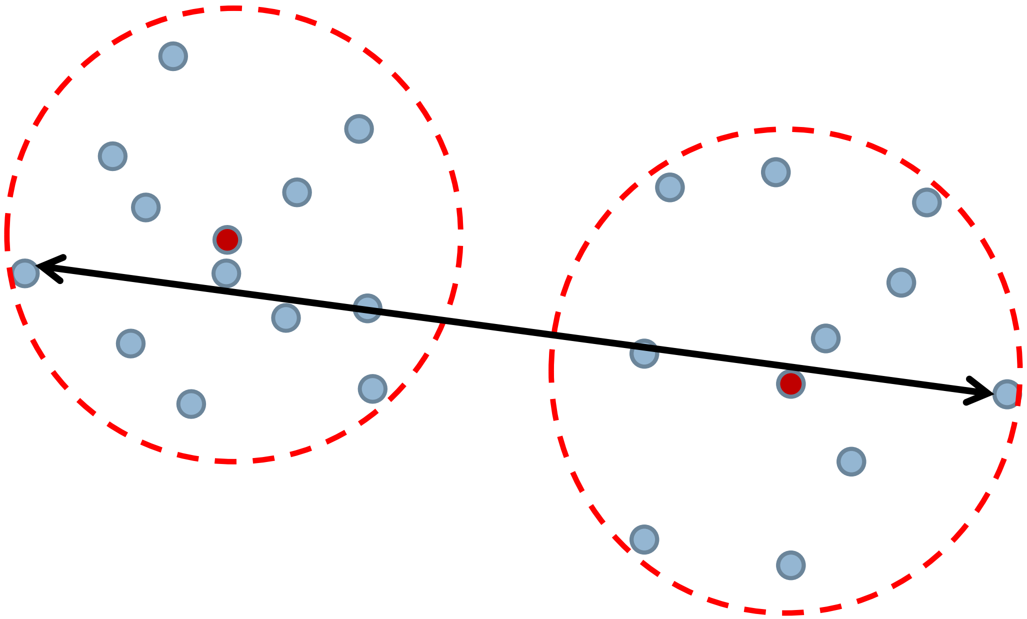
- Above method is "bottom-up" (hierarchical agglomerative clustering)
- It is possible to start from a single cluster of all points and repeatedly split it into smaller clusters
  - This "top-down" approach is often called divisive clustering

When two clusters are close?



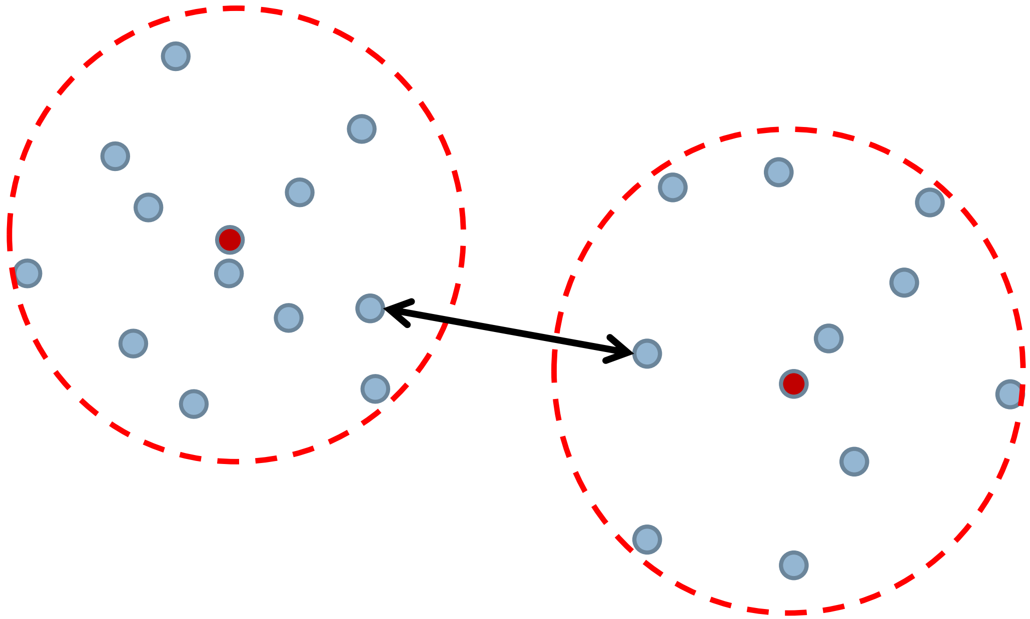
- Idea 1: measure (Euclidian) distance of their centroids

When two clusters are close?



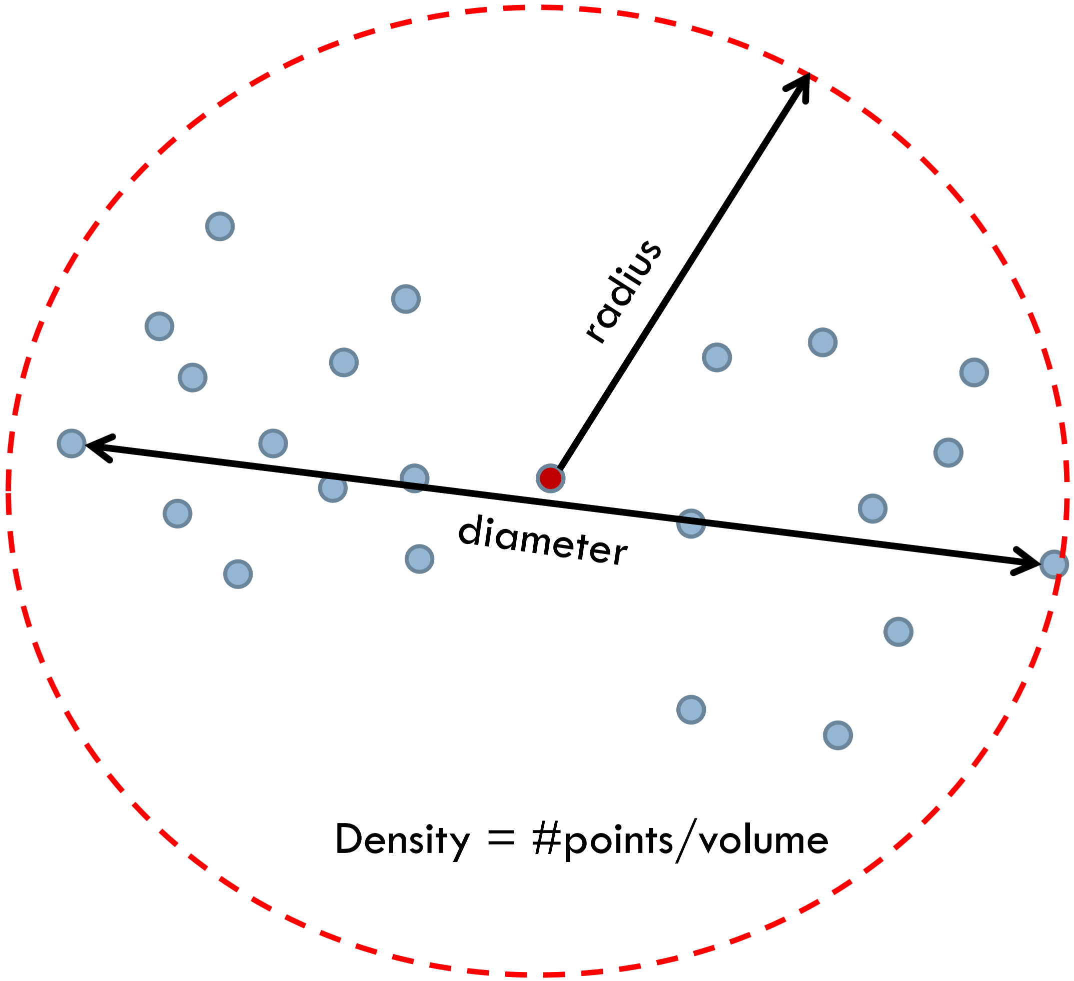
- Idea 2: measure maximum pair-wise distance
  - This will reduce the diameter of the resulting merged cluster

When two clusters are close?



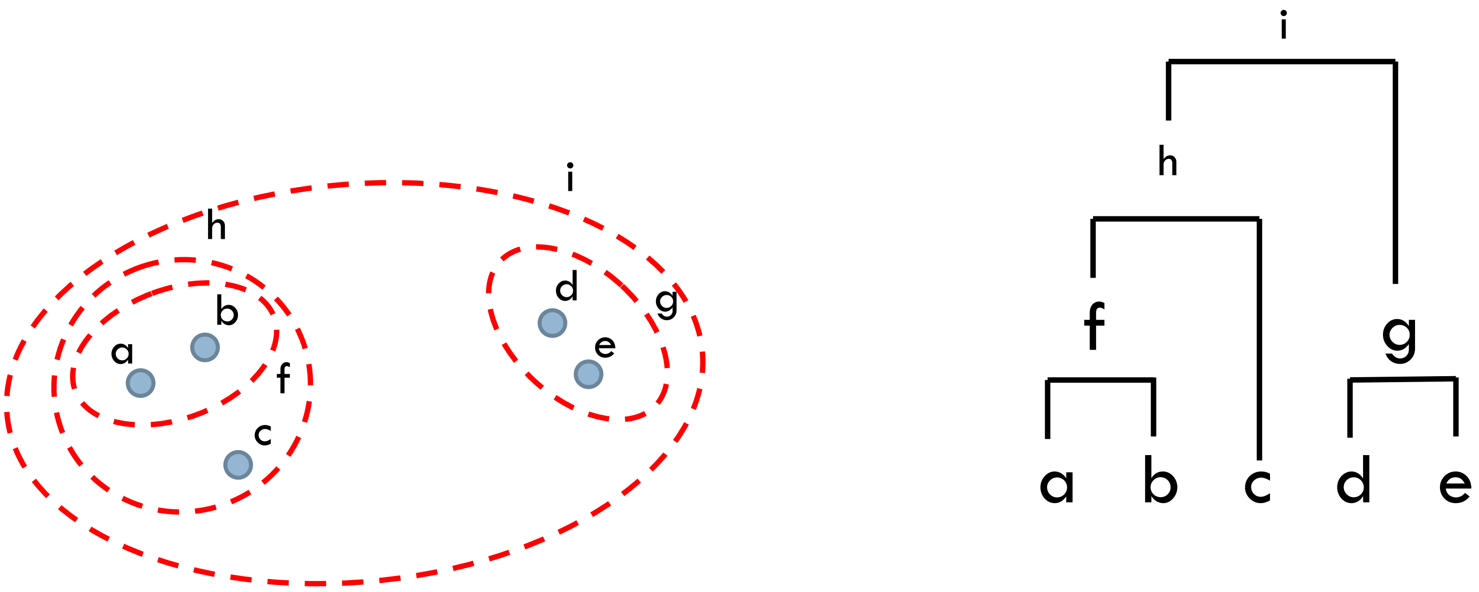
- Idea 3: measure minimum pair-wise distance
  - More ideas: average distances between points, etc

Cluster cohesion: Tell whether resulting cluster is good or bad



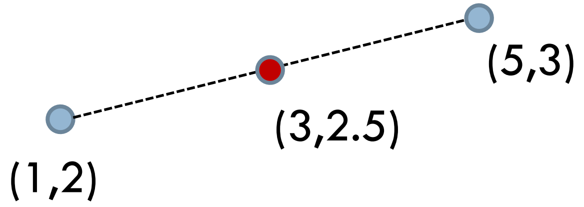
Sum of Squared Distances

HAC example



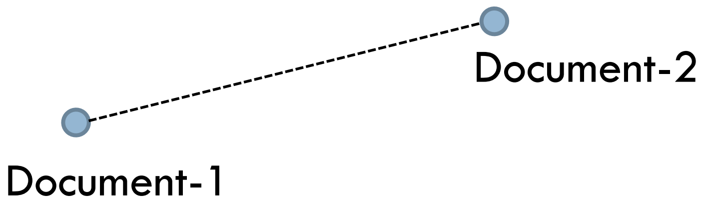
Euclidean space

- In a Euclidean space you may compute the "average" of two points, thus their "centroid"



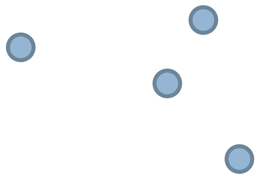
Non-Euclidean space

- In a non-Euclidean space we can not “average” two or more points
  - e.g. we can define a distance between two documents but we cannot take their average in a meaningful manner



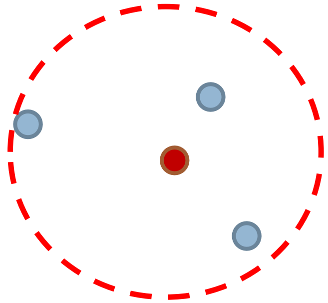
How to represent a cluster in a non-Euclidean space?

- Assume depicted points are documents



How to represent a cluster?

- Select as a representative (often termed "clustoid") the document that is closest to all other docs
  - e.g. clustoid minimizes average distance to all other docs in the cluster



Bisecting k-Means algorithm

- An example of divisive clustering
  - E.g. start from a single cluster
  - Repeatedly split clusters until k clusters are formed

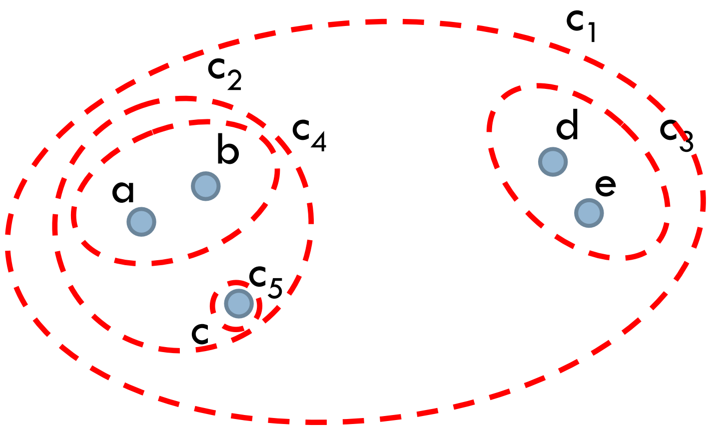
- Bisecting k-Means: Divisive step using 2-Means to split a cluster in two pieces

Algorithm

Bisecting k-Means:
Initialize set of clusters C= {c_1} // c_1 contains all points
Do
    Select a cluster c from C
    For i=1 to ITER //try different bisections of c
        Bisect c using k'-Means (k'=2)
    Pick best bisection, replace c with its sub-clusters
Until |C| = k

- Issues:
  - Which cluster to split?
    - Pick the largest?
    - Pick "worst" (less coherent?)

Bisecting k-Means (k=3)



Back to k-means

- k-means updates centroid locations at each iteration
  - New centoids are computed by taking the arithmetic mean on each dimension
  - Taking the means minimizes the sum of the squared distances from the centroids, thus the within-cluster variance

Analysis of Mean

- Mean is sensitive to outliers
  - Dataset D = {1,2,3,4,5,7,48}
  - Mean = (1+2+3+4+5+7+48)/7=10
  - Avg dist from mean = 10.9
  - Avg squared dist from mean = 244

Mean vs Median

- Mean is more sensitive to outliers
  - Dataset D = {1,2,3,4,5,7,48}
  - Mean = (1+2+3+4+5+7+48)/7=10
  - Avg dist from mean = 10.9
  - Avg squared dist from mean = 244
- Alternative idea: use median
  - Dataset D = {1,2,3,4,5,7,48}
  - Median = 4
  - Avg dist from median= 7.9
  - Avg squared dist from mean = 292.7

Mean vs Median

- Avg dist from mean = 10.9
- Avg squared dist from mean = 244
- Avg dist from median= 7.9
- Avg squared dist from mean = 292.7

k-median algorithm

- k-median algorithm uses the median on each dimension to update the centoids
  - Selection of median minimizes the sum of the distances instead of the sum of the squared distances
  - Resulting values on each dimension are from the dataset but the centroids may not exist in the original dataset (as in k-means)

- Minimizing the sum of the distances relates to the facility location problem

Facility location Problem

- Input
  - A set of demand points D
  - A set of candidate locations L where facilities can be opened
- Assumptions
  - Each demand point is serviced by the closest facility
  - Opening a facility incurs a cost f
- Goal
  - Pick a subset F of facilities to open, to minimize the sum of distances from each demand point to its nearest facility, plus the sum of opening costs of the facilities.
- Variation: pick facilities from demand points D
  - Neat online version: demand points are presented as a stream
  - Check out http://web.cs.ucla.edu/~awm/papers/ofl.pdf

Facility Location Problem for clustering

- Medians are from original point set
- No k is given, but pay f for each median
- Cost function is
  - Sum of assignment distances + (# medians) × f

Reduced when more clusters are used   Reduced when fewer clusters are used

k-Median vs. Facility Location

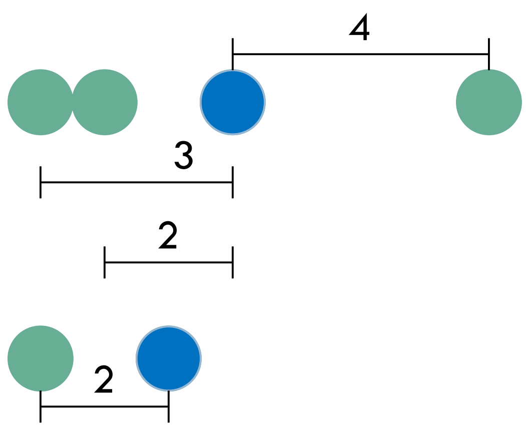
Slides from Liadan O'Callaghan: Clustering Data Streams

Demand Point | Facility Location (or centroid)

k-median:

cost = sum of distances

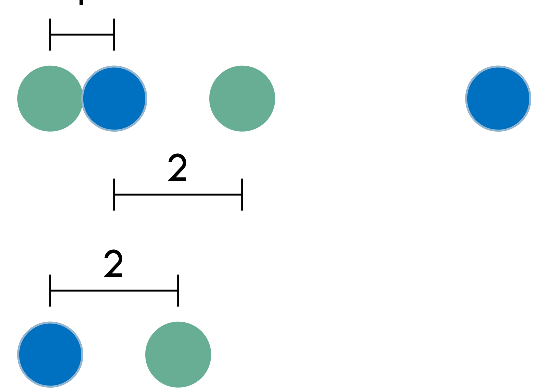
k = 2



Cost is 2+2+3+4=11

facility location: also include facility cost

Facility cost f = 1

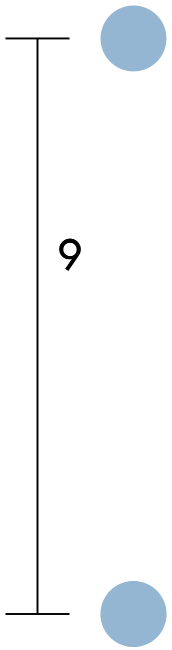


Cost is 1+2+2+(3x1)=8

Meyerson's Algorithm

- A facility location algorithm
- Let f denote facility cost
- Assumption: consider points in random order (or online)
- First point becomes a median
- If x = i^{th} point, d = distance from x to closest existing median:
  - "open" x as a median with prob. d/f
  - else assign x to nearest median

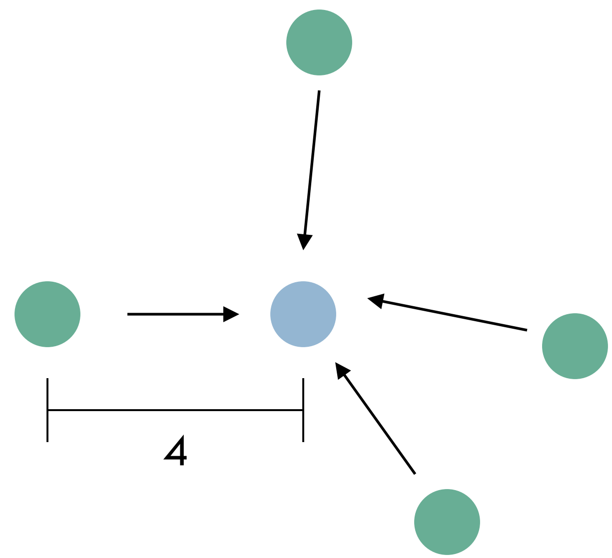
Examples



“opened” (prob .9)

Let f = 10

assigned
(prob 1 - .4 = .6)



Local Search Algorithm

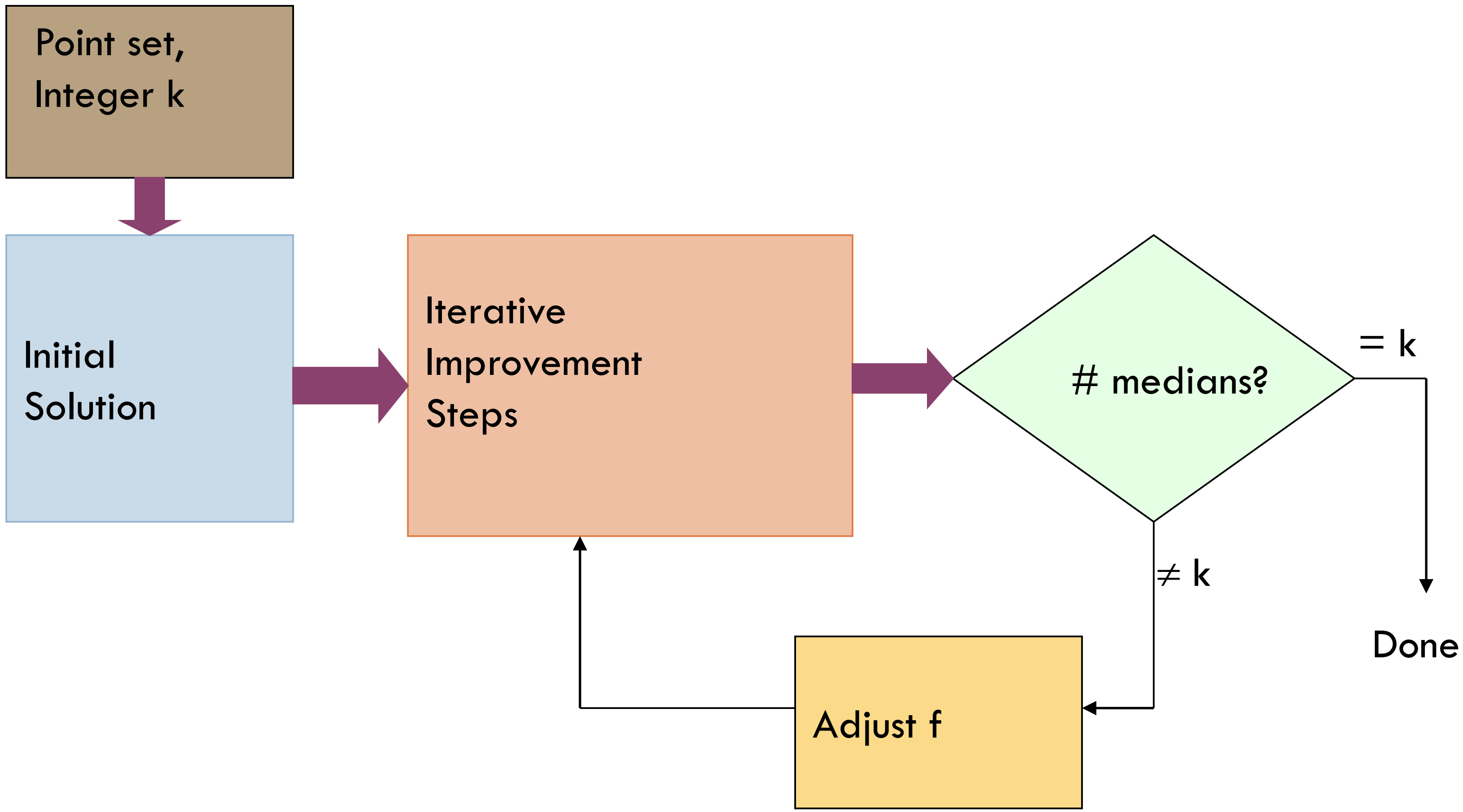
Suggested k-median algorithm will be based on local search, i.e.:

- Start with initial solution (medians + assignment function)
- Iteratively make local improvements to solution
- After some number of iterations, your solution is provably good

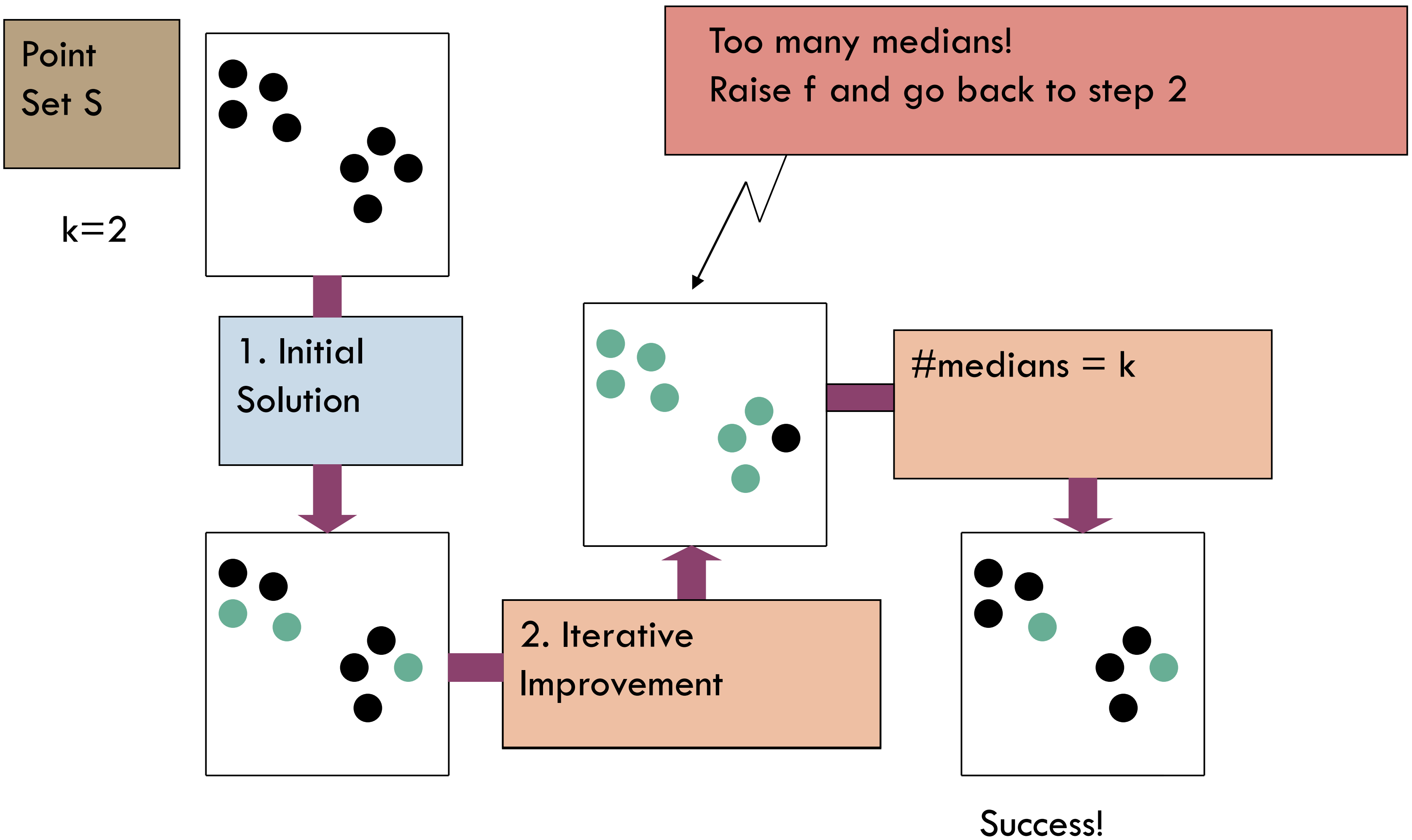
Local Search Algorithm

1. Find initial solution (Meyerson)
2. Iterative local improvement: Check each point, "opening," "closing," or reassigning so as to lower total cost
3. If #medians $\neq$ k, *adjust* facility cost and repeat step 2.
4. At the end: k medians, approx. optimal

Local Search Algorithm



Example


Point
Set S
k=2
1. Initial
Solution
2. Iterative
Improvement
Too many medians!
Raise f and go back to step 2
#medians = k
Success!

Local Search Algorithm Speedup

- Instead of considering all points as feasible facilities, take a sample at the beginning, and only let sample points be medians
- Fewer potential medians to search through
- Solution converges faster
- …And should still be good

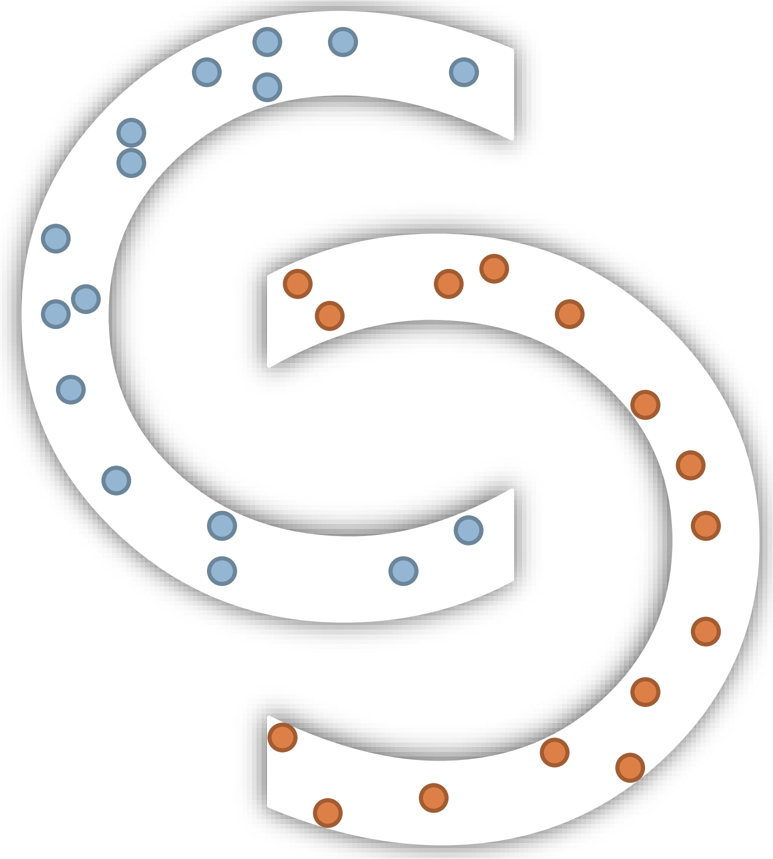
Clustering Using REpresentatives (CURE)

Sudipto Guha, Rajeev Rastogi, Kyuseok Shim:

Cure: An Efficient Clustering Algorithm for Large Databases. Inf. Syst. 26(1): 35-58 (2001)

Clustering Using REpresentatives (CURE)

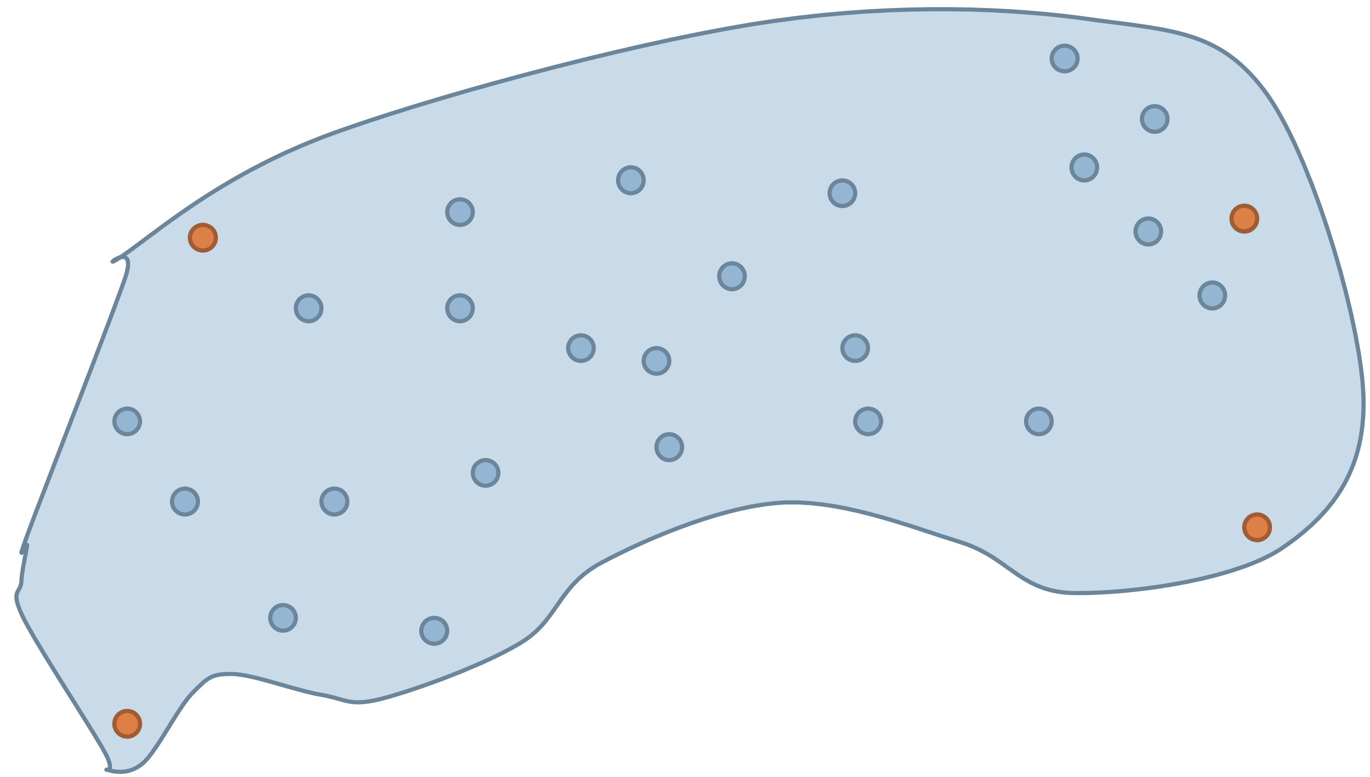
- Uses multiple representatives to represent clusters
- This allows clusters to assume complex forms
  - Also lees sensitive to outliers



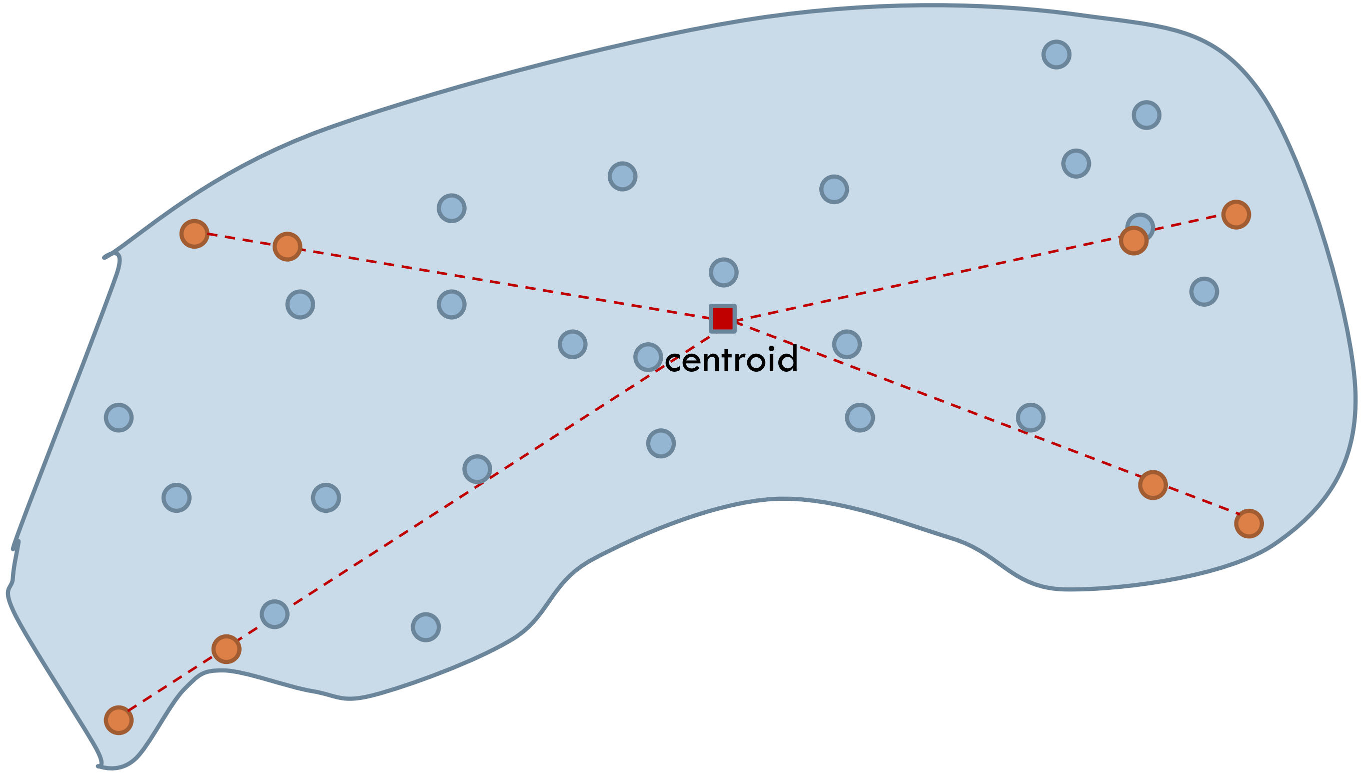
Representatives

- From each cluster select c “well scattered points” as representatives
  - Representatives are as dispersed as possible
- Move each representative points “inwards”, e.g. towards the centroid of the cluster by a fixed fraction *a*%
  - Shrinking the representatives towards the centroid (mean) by a factor *a*% helps get rid of surface abnormalities and reduces the effect of outliers

Selection of Representatives



Shrinkage



CURE uses HAC for merging clusters

- At each step pick the closest pair of clusters
  - Uses a priority queue and a k-d tree to speed up processing
- Distance between two clusters is defined as the minimum distance between their representative points

Pre-processing (for large datasets)

- Take a random sample of the data that fits in main memory
  - Partition sample, form partial clusters
  - Remove outliers, cluster partial clusters
- Use these clusters to initialize HAC

DBSCAN

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD 1996: 226-231

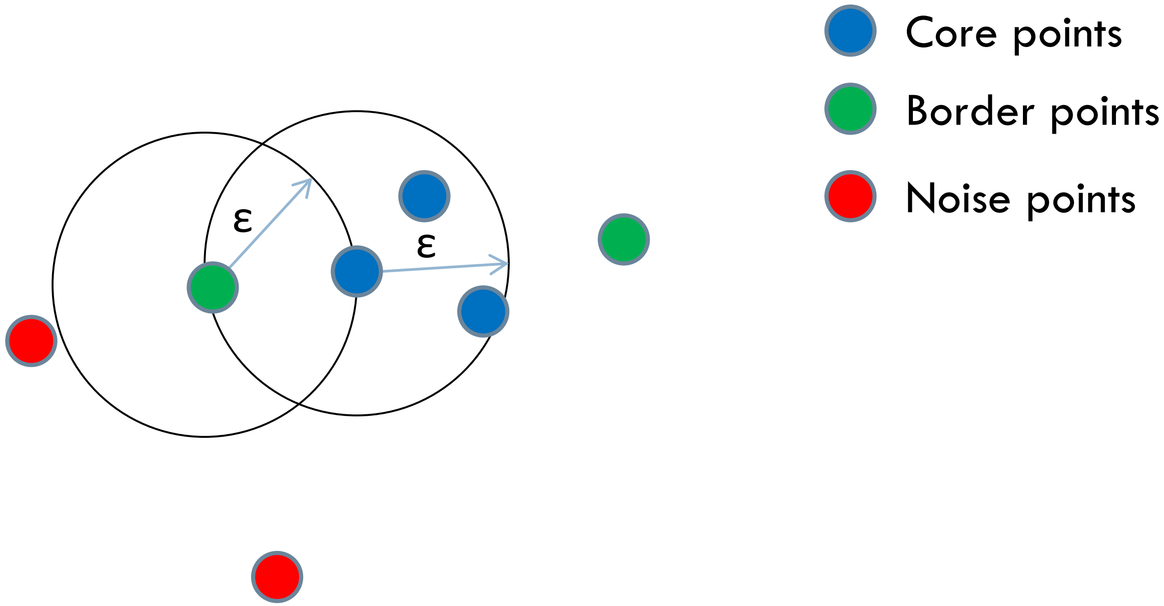
Density-based Clustering

- Intuition: clusters are formed in high density regions and are separated from one another by regions of low density.

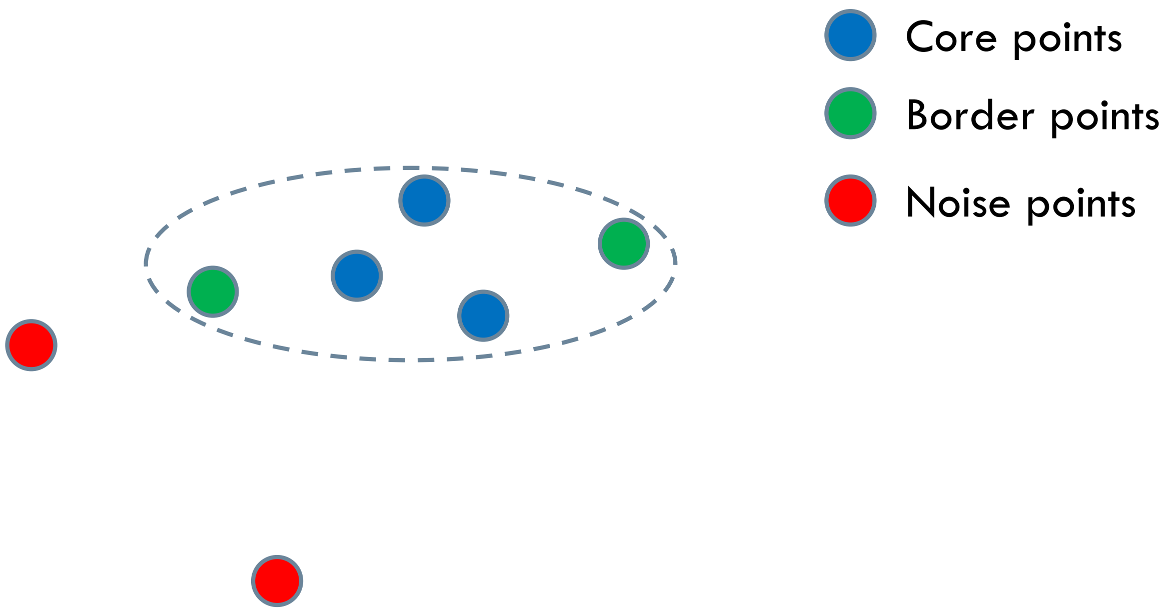
Preliminaries of DBSCAN

- A density based algorithm
  - density = number of points within a specified radius (ε)
- DBSCAN classifies points into three groups
  - A point is a core point if it has more than a specified number of points (MinPts) within distance ε
    - Core points are at the interior of a cluster
  - A border point has fewer than MinPts within distance ε, but is in the neighborhood of a core point
  - A noise point is any point that is not a core point nor a border point

Assume MinPts=3

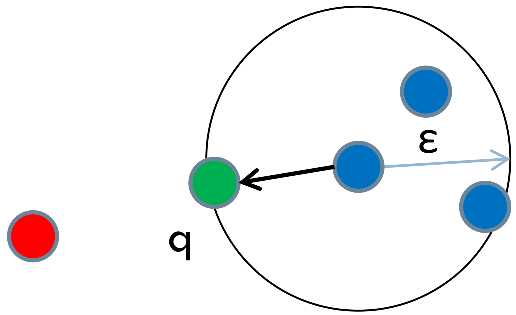


Cluster



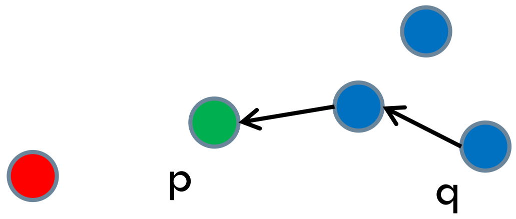
Direct Density-Reachability

- An point q is directly density-reachable from a core point p if it is within distance ε from q
  - Relationship is asymmetric (e.g. when q is a border point)



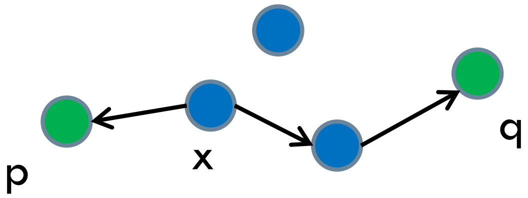
Density-reachability

- A point p is density-reachable from q if there is a chain of points $p_1,\ldots,p_n$, with $p_1=q$, $p_n=p$ such that p_{i+1} is directly density-reachable from p_i for all $1\leq i \leq n$



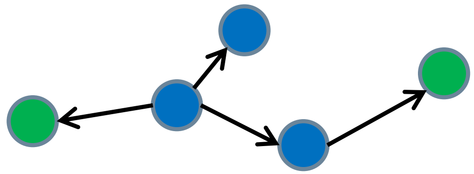
Density-connectivity

- Point p is density-connected to point q if there is an object x such that both p and q are density-reachable from x
  - Relationship is symmetric



Cluster definition

- A cluster C in a set of points satisfying
  - Maximality: For all p, q if p is in C and if q is density-reachable from p then q is also in C
  - Connectivity: for all p, q in C, p is density-connected to q

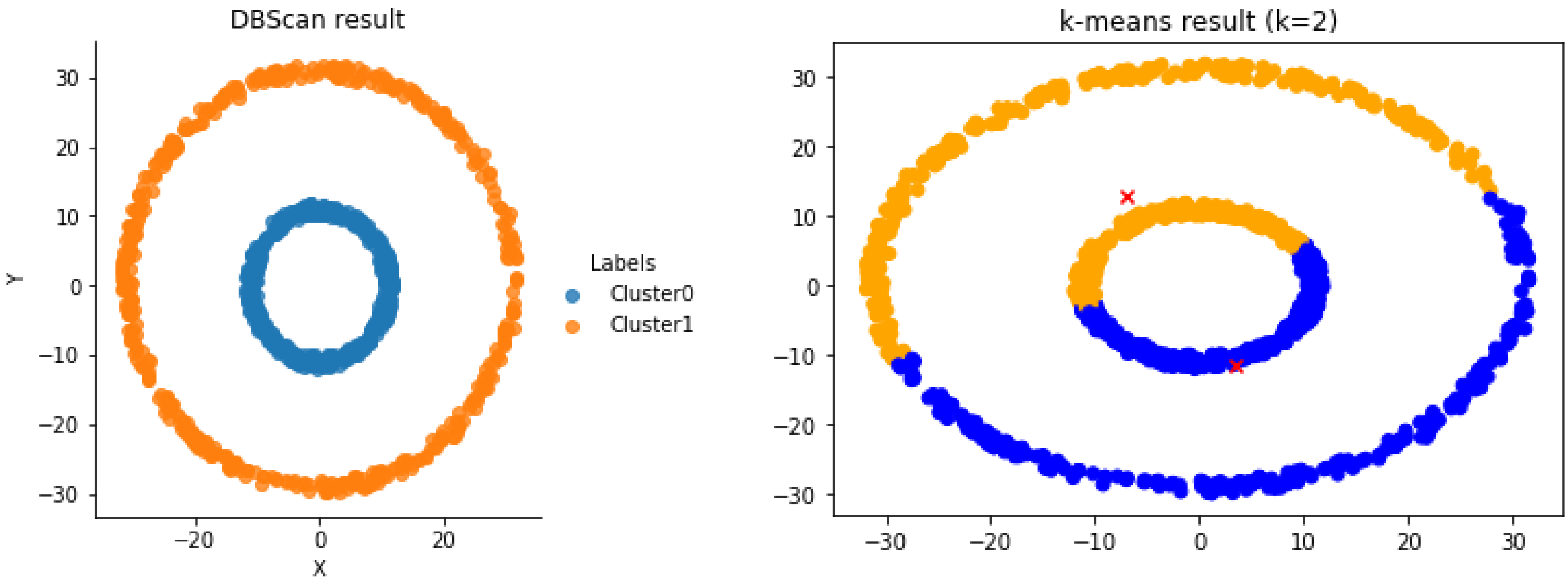


- Noise objects which are not directly density-reachable from at least one core object

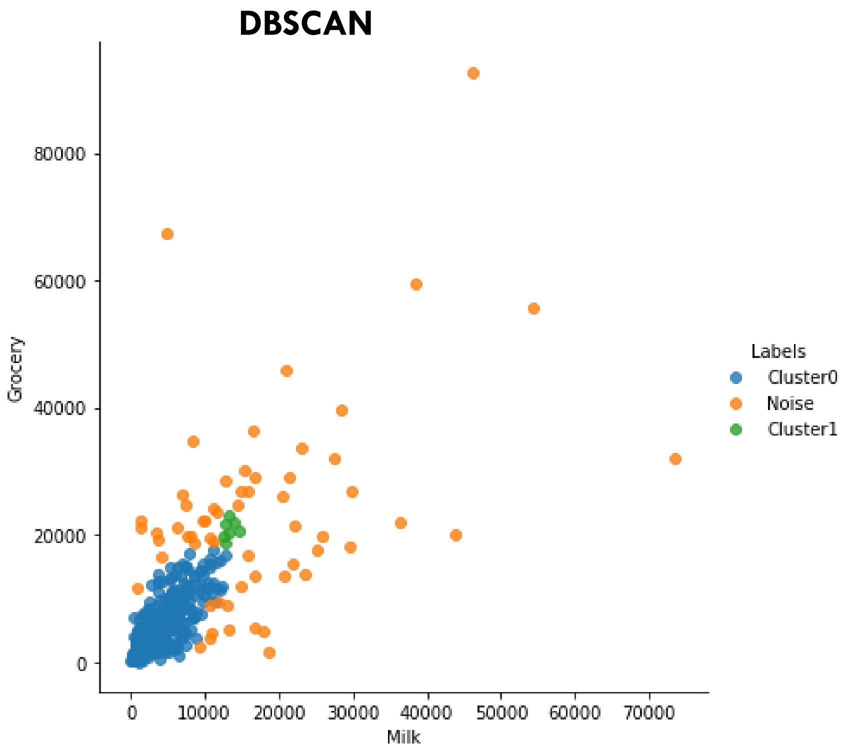
DBSCAN Overview

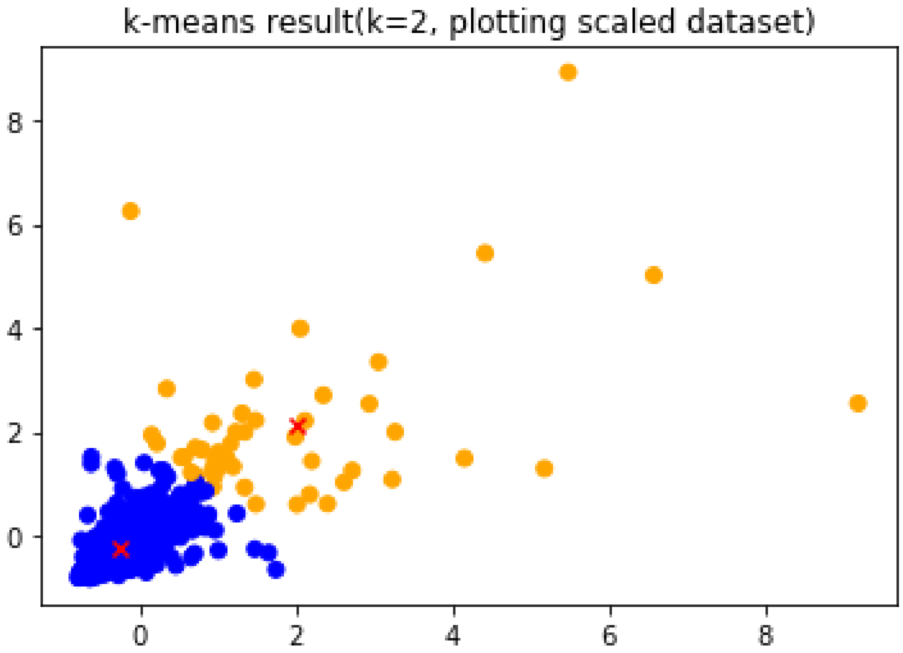
- Core points within distance ε of one another are assigned to the same cluster
- A border point that is in the neighborhood of a core point is assigned to the same cluster
- Noise points are discarded

DBSCAN vs k-Means (code available on eclass)



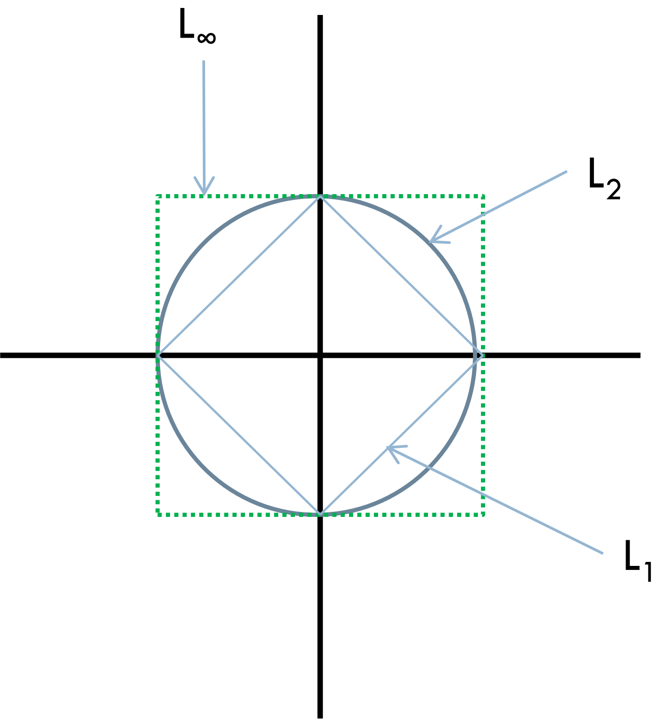
DBSCAN vs k-Means (Wholesale customers data)





How to measure distance/similarity

- Euclidean distance
- Generalization: Lp-norm



$$\|x\|_p = (\sum_{i=1}^{k} |x_i|^p)^{1/p}$$

How to measure distance/similarity

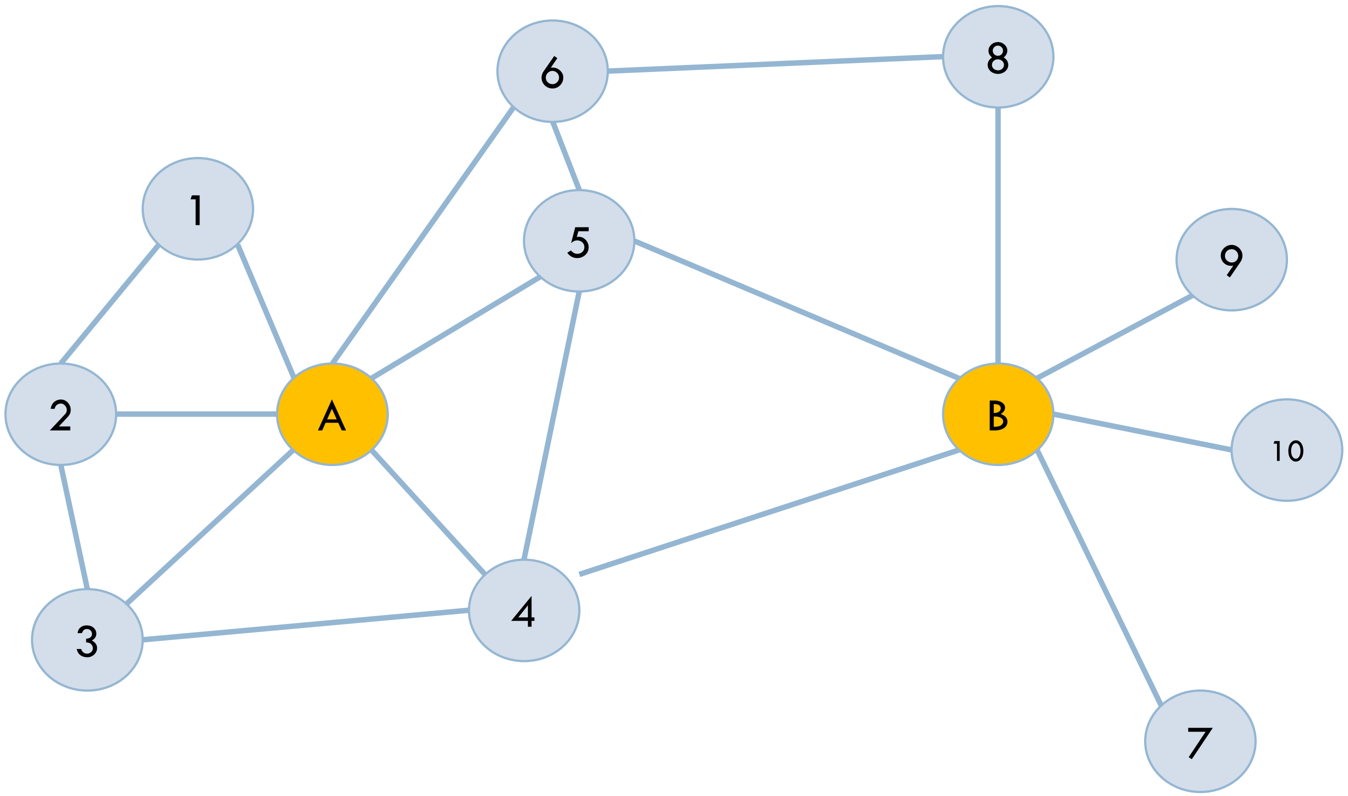
- Cosine coefficient/similarity
  - x and y are n-dimensional vectors

$$\cos(x, y) = \frac{x \bullet y}{|x||y|} = \frac{x}{|x|} \bullet \frac{y}{|y|} = \frac{\sum_{i=1}^{|n|} x_i y_i}{\sqrt{\sum_{i=1}^{|n|} x_i^2}\sqrt{\sum_{i=1}^{|n|} y_i^2}}$$

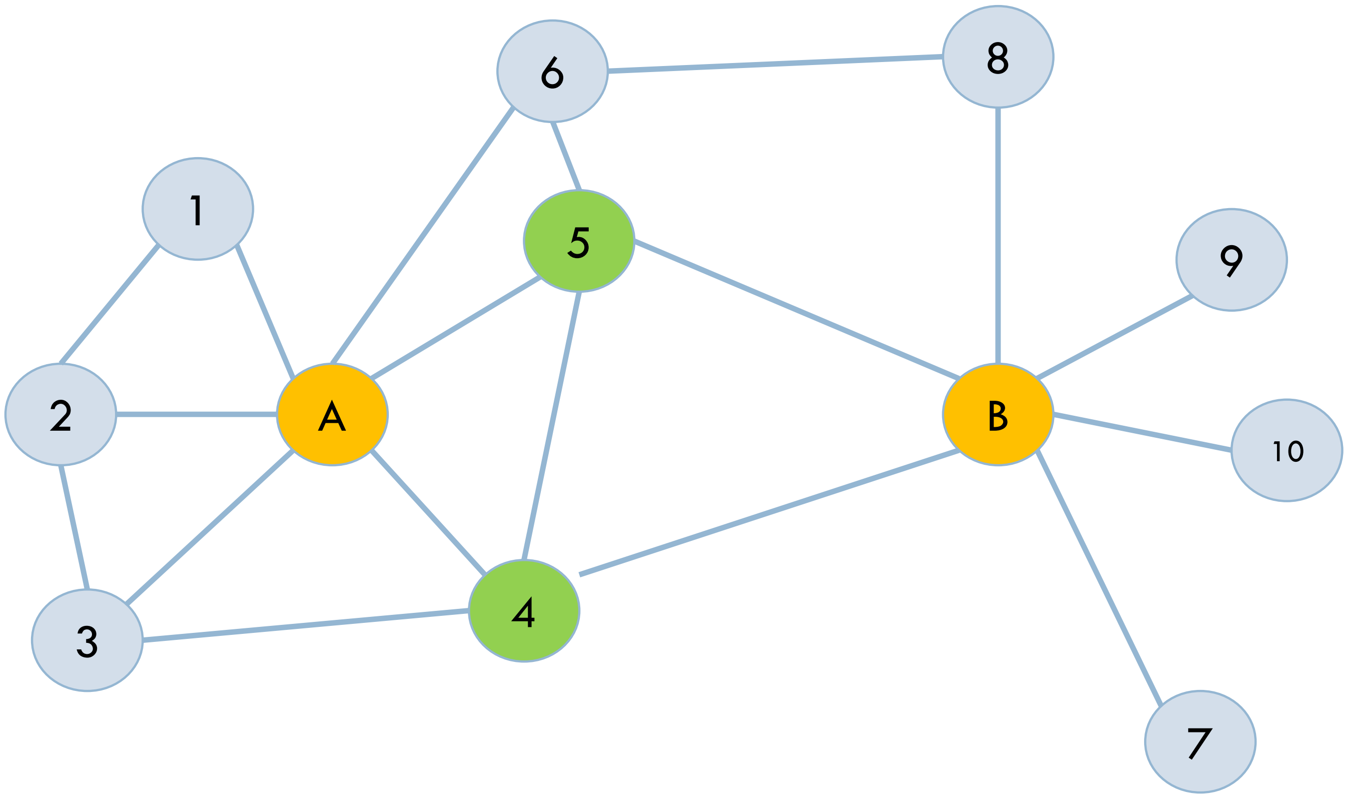
How to measure distance/similarity

- What about interconnected data?

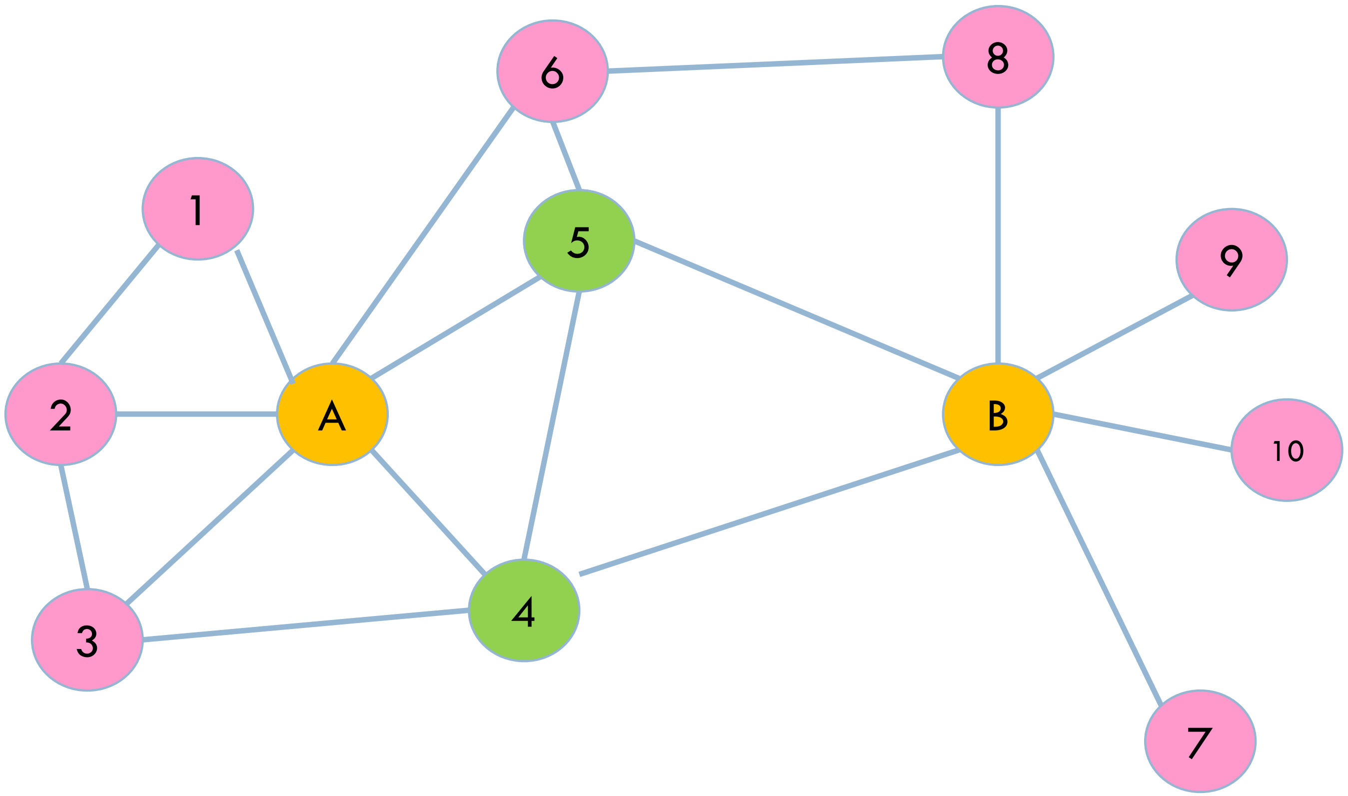
When two graph nodes are similar?



Consider neighbors in-common



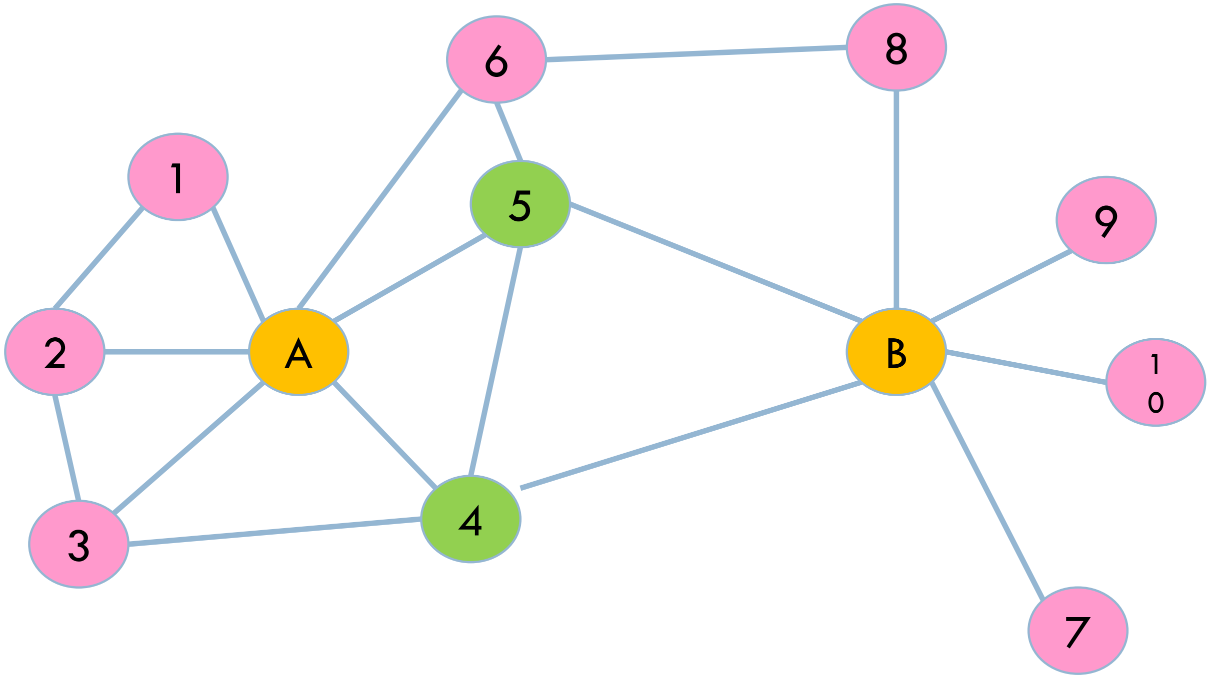
Consider neighbors not in-common



Combine using Jaccard

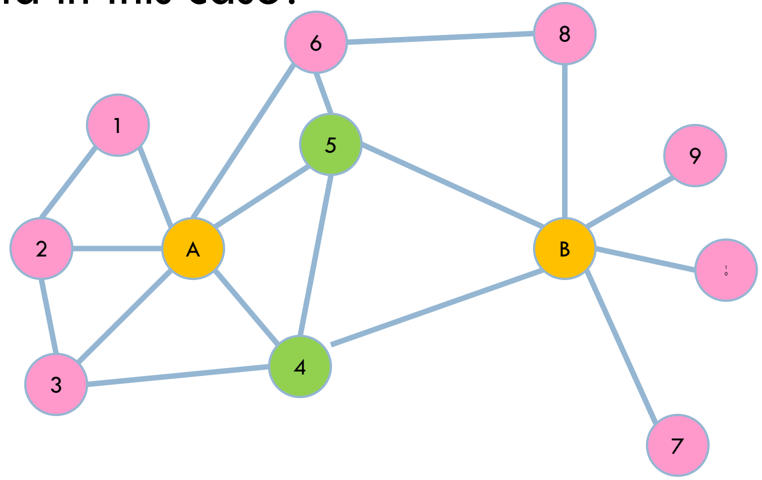
- Let N(u) = set of neighbors of node u
- sim(A,B) = Jaccard(N(A),N(B))

  $= (N(A) \cap N(B))/(N(A) \cup N(B)) = 20\%$

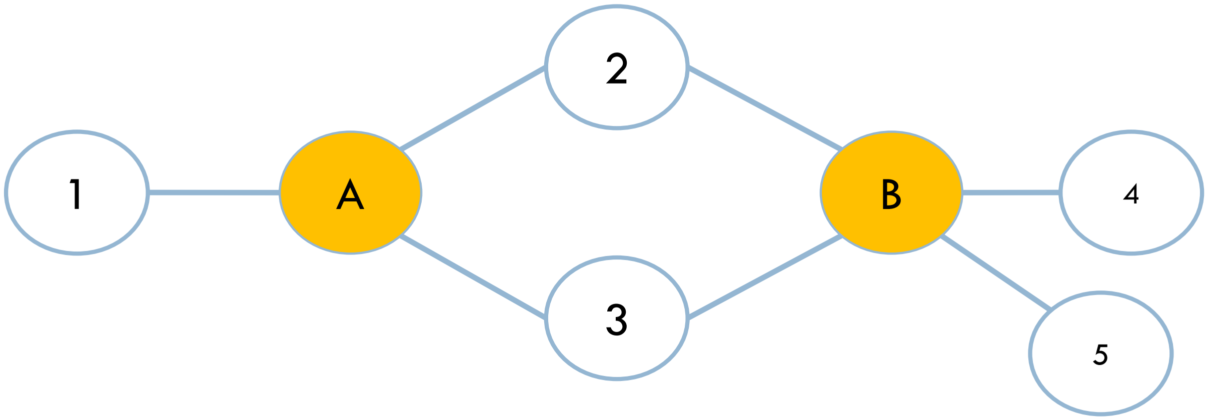


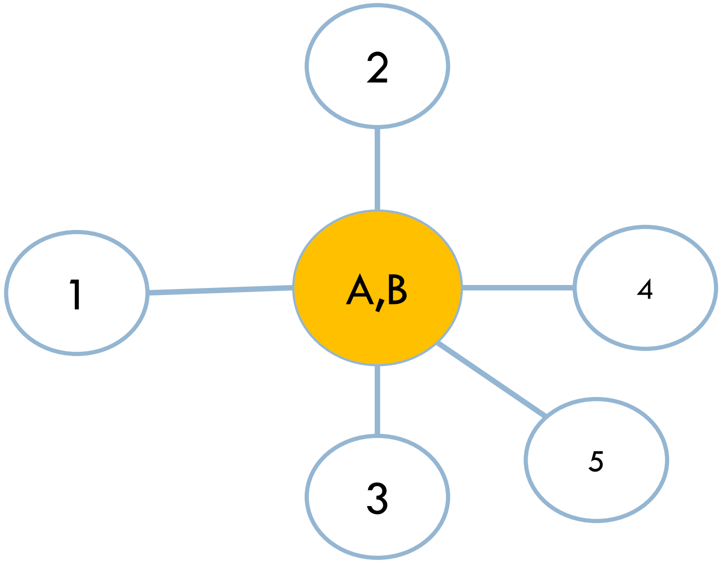
How to apply this idea for clustering

- Define a distance metric based on Jaccard similarity
  - E.g. dist(u,v)=1-Jaccard(N(u),N(v))
- Then, any hierarchical clustering method will do
  - E.g. bottom-up: merge nodes to form clusters
    - Complication: what is a clustoid in this case?

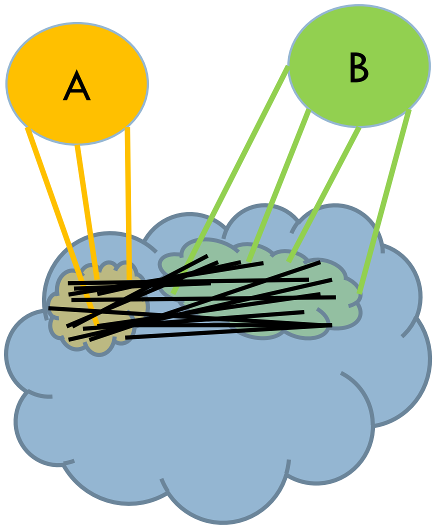


Merging of nodes



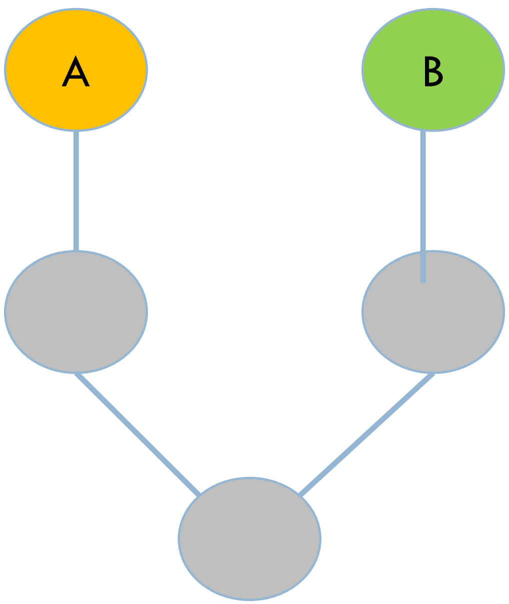


Is it always good?



sim(A,B)=0

Simpler case:
common friend-of-friend



SimRank

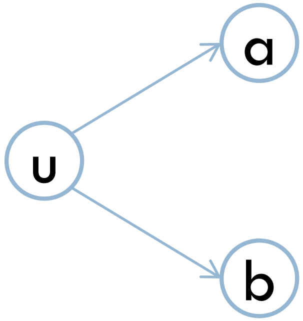
A Measure of Structural-Context Similarity

Glen Jeh and Jennifer Widom
Stanford University

ACM SIGKDD 2002

In a nutshell

- **SimRank: two objects are similar if they are referenced by similar objects**

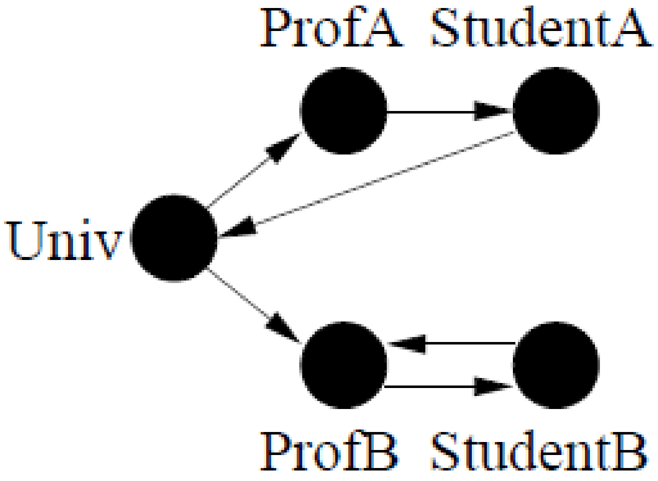


Motivation

- A similarity measure that exploits the object-to-object relationships found in many domains of interest
  - Web page X "points to" Web page Y
  - customer "buys" product
- May be used to cluster objects, such as for collaborative filtering in a recommender system

Intuition

- Concentrate on structural content
  - Can be combined with other similarity metrics that consider content similarity
- Two nodes are similar if they are referenced by similar nodes



SimRank Recursive Computation

- Initialize:
  - $s(a,b) = \begin{cases} 1, \text{ if } a=b \\ 0, \text{ otherwise} \end{cases}$

- Iteratively compute ($a \neq b$):

$$s(a,b) = \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

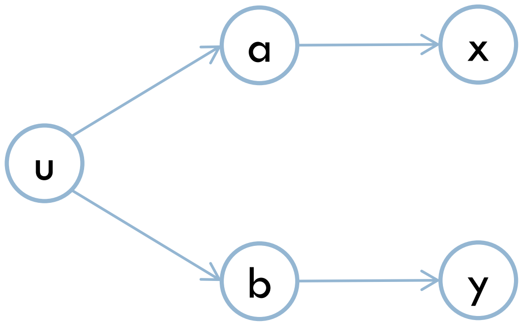
- Where
  - $I(x)$ = in-neighbors of x
  - $I_i(x)$ = i^{th} in-neighbor of x and $C<1$ (decay factor)

Explanation

$$s(a,b) = \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

- Nodes receive the average similarity of their in-neighbors multiplied by the decay factor C
- Special case: s(a,b) = 0 if |I(a)| = 0 or |I(b)|=0
  - i.e. nodes have no in-neighbors

Example

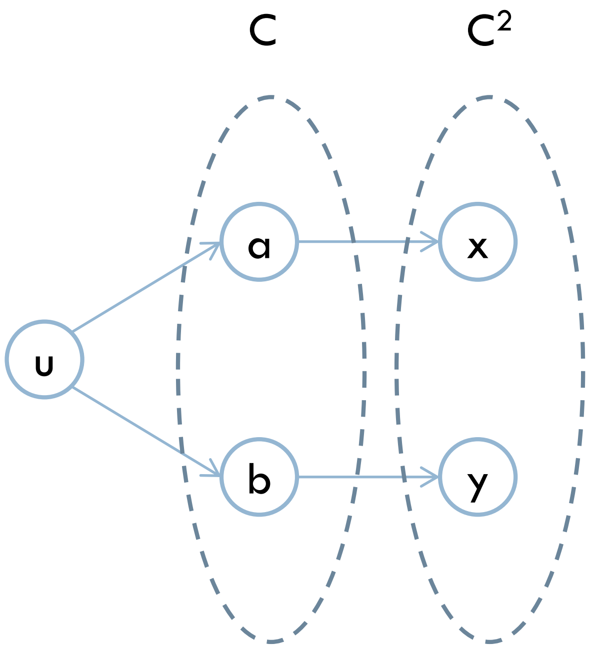


Initialization

s(u,u)=1
s(a,b)=0
s(a,x)=0
s(x,y)=0

Assume C=0.8

Iterate



Updated SimRank

s(u,u)=1

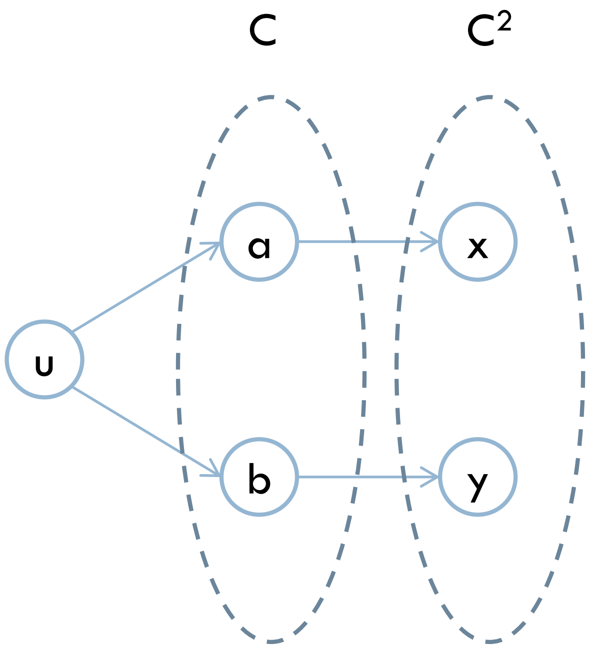
s(a,b)=0.8*s(u,u)=0.8

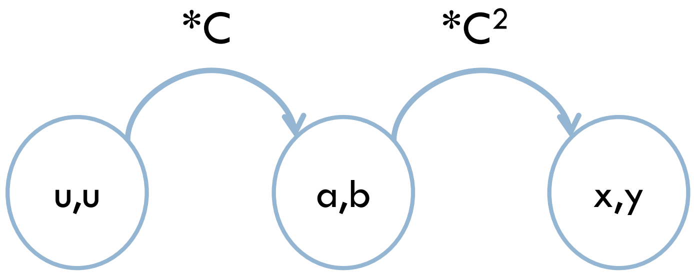
s(a,x)=0.8*s(u,a)=0

s(x,y)=0,8*s(a,b)=0,8*0,8=0,64

Assume C=0.8

SimRank propagation

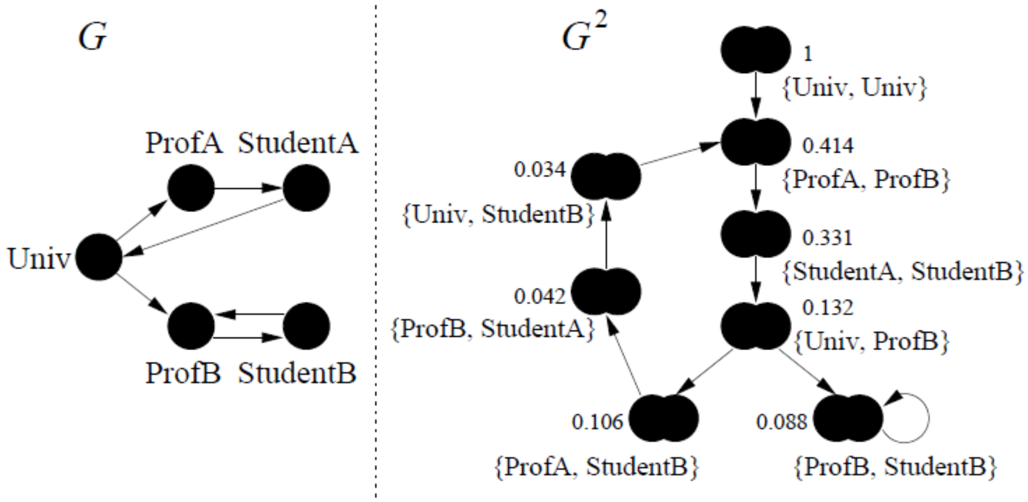




Assume C=0.8

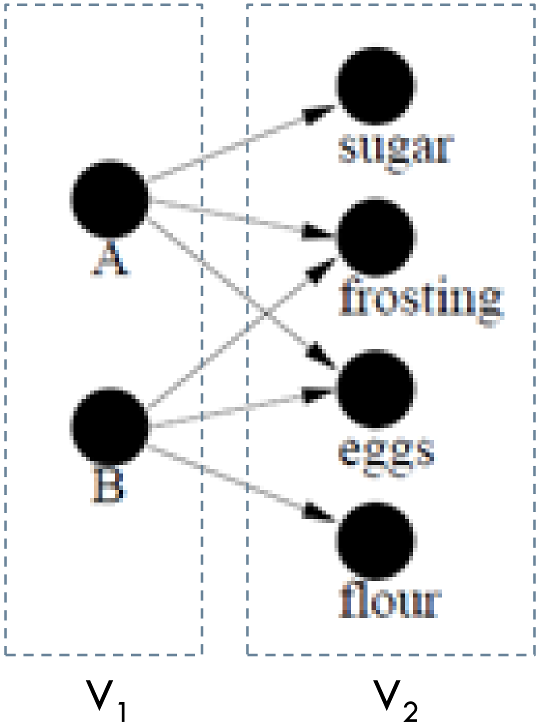
Another View

- Let $G^2=(V^2, E^2)$ with
  - $V^2=V \times V$, represents a pair (a,b) of nodes in G
  - An edge from (a,b) to (x,y) exists in E^2, iff the edges <a,x> and <b,y> exist in G
- SimRank propagates through pairs in G^2



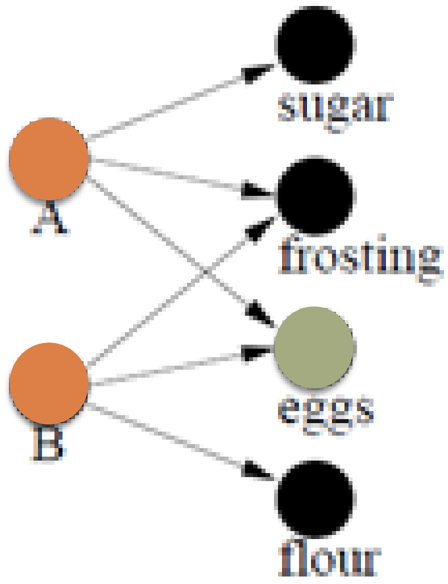
SimRank in bipartite graphs

- Bipartie graph: two disjoint classes of nodes V_1, V_2
  - e.g. V_1={customers}, V_2={items}
  - Edges only between nodes in V_1 to nodes in V_2



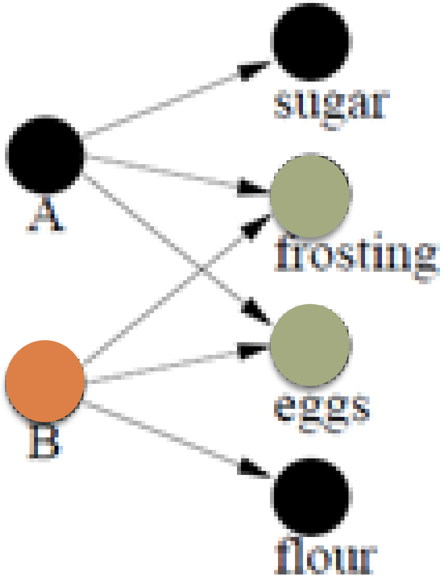
Intuition-1

- People are similar if they purchase similar objects



Intuition-2

- Items are similar if they are purchased by similar people



Bipartite SimRank

- SimRank between persons A and B, (A≠B)

$$s(A,B) = \frac{C_1}{|O(A)||O(B)|} \sum_{i=1}^{|O(A)|} \sum_{j=1}^{|O(B)|} s(O_i(A), O_j(B))$$

- SimRank between items x and y, (x≠y)

$$s(x,y) = \frac{C_2}{|I(x)||I(y)|} \sum_{i=1}^{|I(x)|} \sum_{j=1}^{|I(y)|} s(I_i(x), I_j(y))$$

- The similarity between persons A and B is the average similarity between the items they purchased
  - O(A) are the out-neighbors (items) for person A
- The similarity between items x and y is the average similarity between the people who purchased them

Modified SimRank in bipartite graphs