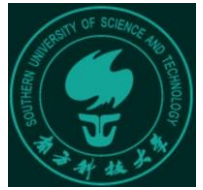
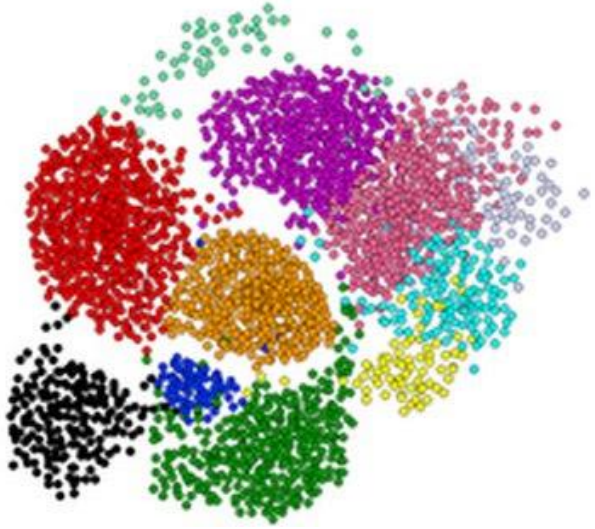
Unsupervised Learning

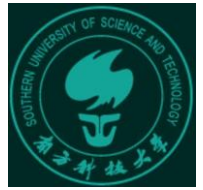


Xin Yao

Fall 2020

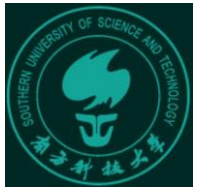


Supervised and Unsupervised Learning



- In previous lectures, we have seen some supervised learning methods. Given some labelled data (discrete or continuous), we aim at training some reliable models to predict the labels of new data that have never been seen during training.
- However, this is not always the case in real life.
- If there is no label, can we still learn something from data?
- Today, we are going to see some unsupervised learning (无监督学习) methods. The most popular one is Clustering (聚类).

Outline

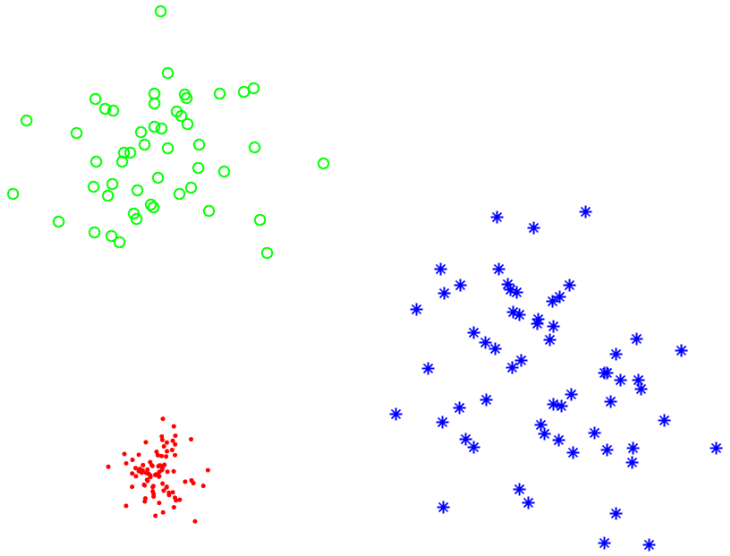


- What is Clustering?
- Hierarchical Clustering
- *k*-means and Other Cost Minimization Clustering
- Spectral Clustering
- Summary

What is Clustering

Clustering (聚类)

- Group similar objects/data together.
- It is an unsupervised learning methods, because the labels are not given.
- Try to learning the implicit pattern, properties and structures among the given unlabelled data.



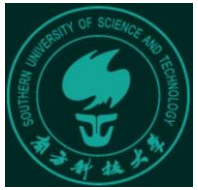
Definition of Clustering

"Given a representation of n objects, find k groups based on a measure of similarity such that the similarities between objects in the same group are high while the similarities between objects in different groups are low."

[1] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern recognition letters, 31(8), 651-666.

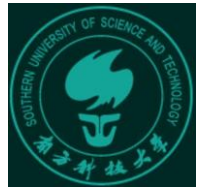
"Objects" can be numerical or not.

Why Data Clustering? [1]



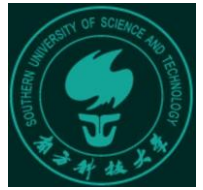
- **Underlying structure:** to gain insight into data, generate hypotheses, detect anomalies, and identify salient features.
- **Natural classification:** to identify the degree of similarity among data.
- **Compression:** as a method for organising the data and summarising it through cluster prototypes.

Data Points and Similarity/Distance



- To group the unlabelled data points to clusters, a "distance" measure is needed.
- Points that are closed to each other should be in the same cluster (*compact*), while points belong to different clusters should have larger distances (*isolated*).
- In an Euclidean space, each data point is a vector of real values (different features, e.g., price, time cost, quality, size, …).
- The vector's components are celled coordinates.
- With a distance measure, we can perform a clustering.

Properties of Legal Distance Measures



- A distance measure $d(\cdot)$ should have the following properties:
  - **Non-negative:** $\forall x_i, x_j,\ d(x_i, x_j) \geq 0.\ d(x_i, x_j) = 0$ iff $x_i = x_j$.
  - **Symmetric:** $d(x_i, x_j) = d(x_j, x_i)$.
  - **Triangle inequality:** $d(x_i, x_k) + d(x_k, x_j) \geq d(x_i, x_j)$.

Distance Measures

- **If real-valued attributes:**
  - The Minkowski distance (L_p-distance);
  - the Manhattan distance (L_1-distance);
  - the Euclidean distance (L_2-distance);
  - the L_∞-distance.
- **If ordinal categorical attributes:**
  - The Minkowski distance (L_p-distance) can be used.
- **If non-ordinal attributes:**
  - Value difference metric (VDM).
- **If mixed:**
  - Minkowski + VDM.

The Minkowski distance (L_p-distance)

- Given two examples $\boldsymbol{x}_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})$ and $\boldsymbol{x}_j = (x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(n)})$, where $x_i^{(l)}$ denotes the l^{th} coordinate of the example $\boldsymbol{x}_i$.
- The Minkowski distance is defined as:

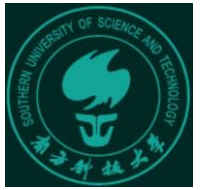
$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^p \right)^{\frac{1}{p}}.$$

The Euclidean distance (L_2-distance)

- Given two examples $\boldsymbol{x}_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})$ and $\boldsymbol{x}_j = (x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(n)})$, where $x_i^{(l)}$ denotes the l^{th} coordinate of the example $\boldsymbol{x}_i$.
- The Euclidean distance is defined as:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^2}.$$

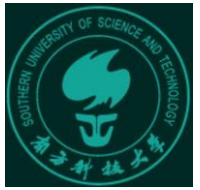
The Manhattan distance (L_1-distance)



- Given two examples $\boldsymbol{x}_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})$ and $\boldsymbol{x}_j = (x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(n)})$, where $x_i^{(l)}$ denotes the l^{th} coordinate of the example $\boldsymbol{x}_i$.
- The Manhattan distance is defined as:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|.$$

The Chebyshev distance (L_∞-distance)



- Given two examples $\boldsymbol{x}_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})$ and $\boldsymbol{x}_j = (x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(n)})$, where $x_i^{(l)}$ denotes the l^{th} coordinate of the example $\boldsymbol{x}_i$.
- The Chebyshev distance is defined as:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \max_{1 \le l \le n} |x_i^{(l)} - x_j^{(l)}|.$$

Value Difference Metric (VDM)

- Let $N(j,a)$ denote the number of examples that have value a at j^{th} coordinate and $N_i(j,a)$ denote the number of examples in the cluster i that have value a at j^{th} coordinate. k is the number of clusters.
- The VDM for j^{th} coordinate (j^{th} feature) is defined as:

$$VDM_p(a,b) = \sum_{i=1}^{k} \left| \frac{N_i(j,a)}{N(j,a)} - \frac{N_i(j,b)}{N(j,b)} \right|^p .$$

Distance Measures: More

- **Points**: Measure similarity by Euclidean distance (already explained).

- **Vectors**: Measure similarity by the cosine distance:

$$\cos(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum_{l=1}^{n} x_i^{(l)} x_j^{(l)}}{\sqrt{\sum_{l=1}^{n} \left(x_i^{(l)}\right)^2} \sqrt{\sum_{l=1}^{n} \left(x_j^{(l)}\right)^2}}.$$

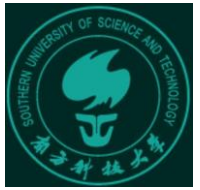
- **Sets**: Measure similarity by the Jaccard distance of sets A and B:

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

Performance Measure

- Given a clustering method or the resulted clusters, the performance can be measured as
  - Inter-cluster similarity.
    - Example: any number of distance measures, such as the distance between the centroids of the clusters.
  - Intra-cluster similarity.
    - Example: the maximal distance between any pair of elements in cluster k.
  - Stability as the generalization ability of a clustering algorithm (in PAC-Bayesian sense).

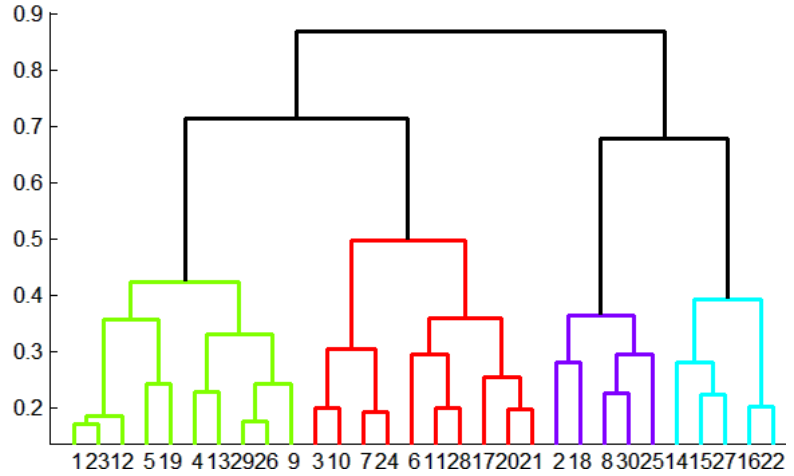
Clustering is hard!



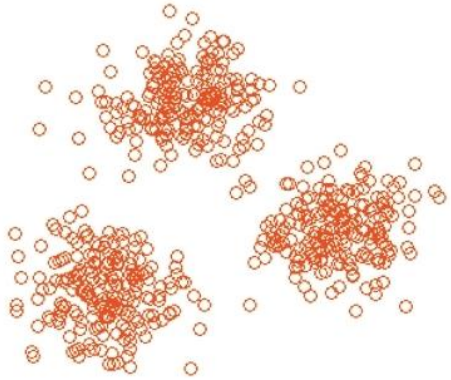
- Clustering is hard due to the scalability!
  - Clustering in two dimensions looks easy. However, many applications involve not 2, but 10 or 10,000 dimensions.
  - Clustering small amounts of data looks easy.
- In most cases, “looks” can be deceiving.
  - High-dimensional spaces look different: Almost all pairs of points are at about the same distance.
- **We need automatic algorithms for high-dimensional data.**

Overview: Methods of Clustering

- **Hierarchical:**
  - **Agglomerative (bottom up):**
    - Initially, each point is a cluster.
    - Repeatedly **merge** the two "nearest" clusters into one.
  - **Divisive (top down):**
    - Start with one cluster and recursively split it.

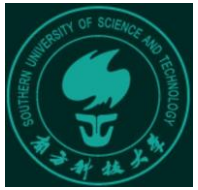


- **Partitional (point assignment):**
  - Maintain a set of clusters (partition of data).
  - Points belong to "nearest" cluster.
  - No hierarchical structure.



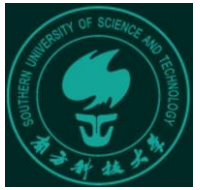
Hierarchical Clustering

Hierarchical Clustering



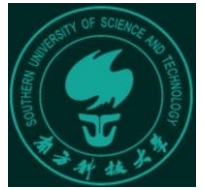
- Hierarchical clustering algorithms recursively find nested clusters
  - either in **agglomerative mode** (starting with each data point in its own cluster and merging the most similar pair of clusters successively to form a cluster hierarchy);
  - or in **divisive (top-down) mode** (starting with all the data points in one cluster and recursively dividing each cluster into smaller clusters).
- Core idea: Repeatedly merge two nearest clusters.

Input to a Hierarchical Algorithm



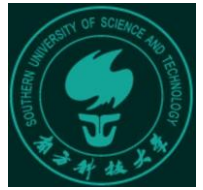
- Input to a hierarchical algorithm is an $N \times N$ similarity matrix, where N is the number of objects to be clustered.
- On the other hand, a partitional algorithm can use either an $N \times d$ pattern matrix, where N objects are embedded in a d-dimensional feature space, or an $N \times N$ similarity matrix.

Hierarchical Clustering: 3 Questions



1. How do you represent a cluster of more than one point?
2. How do you determine the “nearness/similarity” of clusters?
3. When to stop combining clusters?

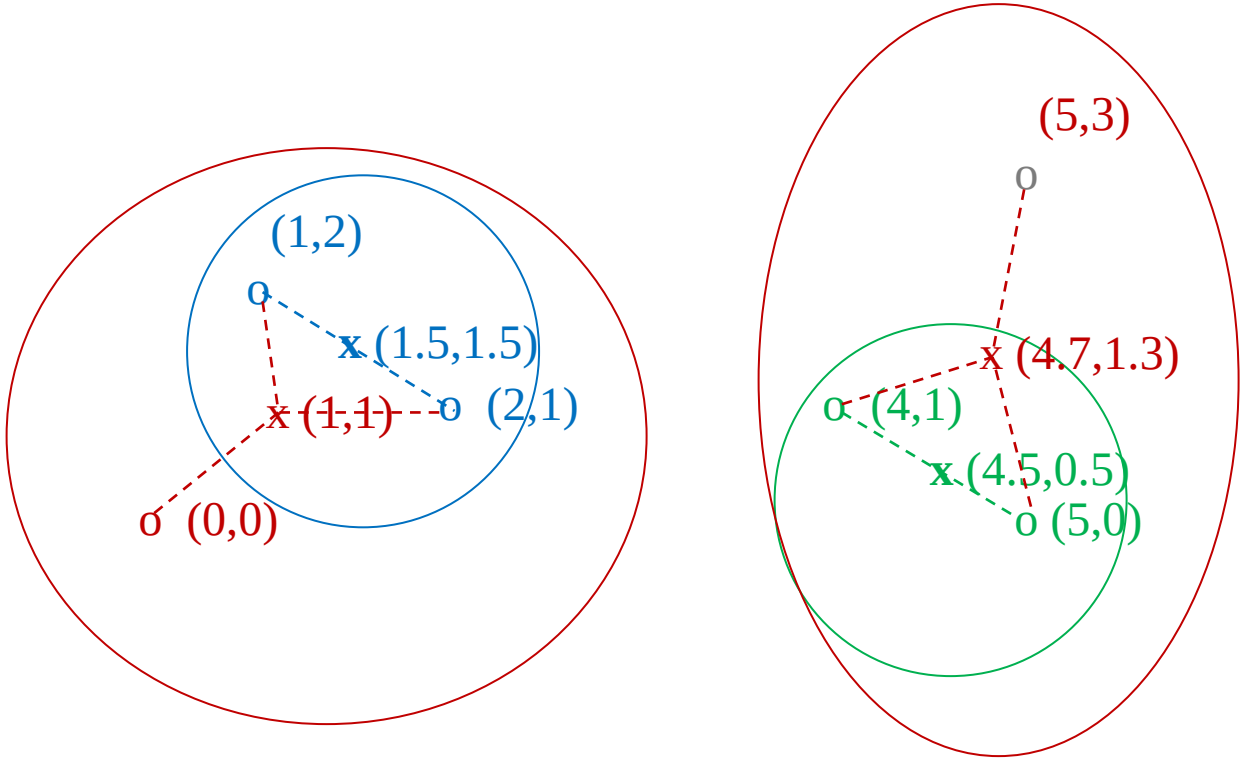
Hierarchical Clustering: 3 Questions



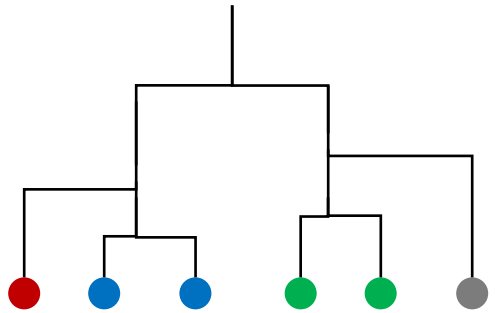
1. How do you represent a cluster of more than one point?
   - **Key problem:** As you merge clusters, how do you represent the "location" of each cluster, to tell which pair of clusters is closest?
   - **Euclidean case:** each cluster c has a ***centroid*** $\boldsymbol{\mu}_c$ = average of its (data) points:

     $\boldsymbol{\mu}_c = \frac{1}{|S_c|}\sum_{\boldsymbol{x}\in S_c}\boldsymbol{x}$, where S_c denotes the set of points in cluster c.
2. How do you determine the "nearness/similarity" of clusters?
   - Measure cluster distances by distances of centroids.
3. When to stop combining clusters?

Example: Hierarchical clustering [2]

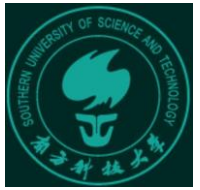


Data:
o … data point
x … centroid



Dendrogram: it illustrates the arrangement of the clusters produced by the corresponding analyses.

And in the Non-Euclidean Case?

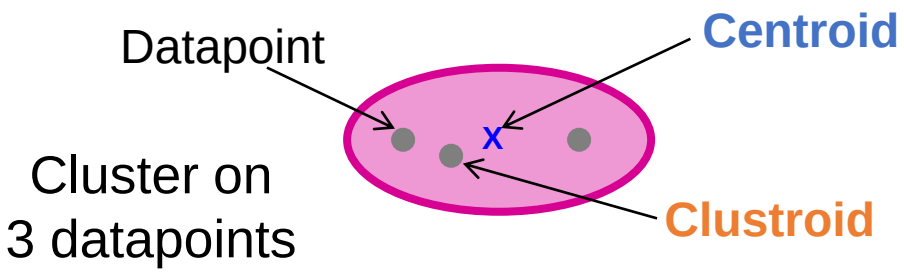


- **What about the Non-Euclidean case?**
  - The only "locations" we can talk about are the points themselves.
  - i.e., there is no "average" of two points.

- **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.

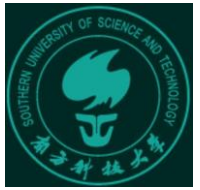
"Closest" Point? [2]

- **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 $\boldsymbol{c_p}$ of cluster ***C*** is: $\arg\min_{p_c \in C} \sum_{x \in C, x \neq c_p} d(x, p_c)^2$



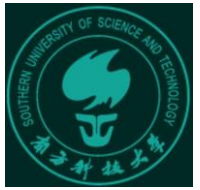
Centroid is the avg. of **all** (data) points in the cluster. This means centroid is an "artificial" point.
Clustroid is an **existing** (data) point that is "closest" to all other points in the cluster.

Defining “Nearness” of Clusters



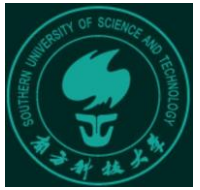
- **How do you determine the “nearness” of clusters?**
  - **Approach 2:**
    Intercluster distance = minimum of the distances between any two points, one from each cluster.
  - **Approach 3:**
    Pick a notion of “**cohesion**” of clusters, *e.g.*, maximum distance from the clustroid.
    - Merge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

Cohesion



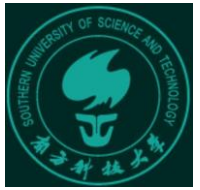
- **Approach 3.1:** Use the **diameter** of the merged cluster = maximum distance between points in the cluster.
- **Approach 3.2:** Use the **average distance** between points in the cluster.
- **Approach 3.3:** Use a **density-based approach**.
  - Take the diameter or avg. distance, e.g., and divide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

Implementation



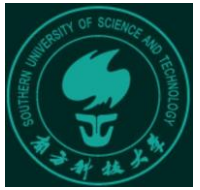
- **Naïve implementation of hierarchical clustering:**
  - At each step, compute pairwise distances between all pairs of clusters, then merge.
  - $O(N^3)$.
- Careful implementation using priority queue can reduce time to $O(N^2 \log(N))$.
  - **Still too expensive for really big datasets that do not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

Well-known Hierarchical Algorithms



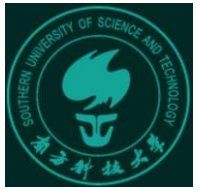
- Linkage-Based Clustering Algorithms:
  - Single-link clustering.
  - Complete-link clustering.

Single-link Clustering



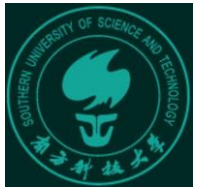
- The similarity of two clusters is the similarity of their *most similar* members (optimistic).
- We pay attention to the area where the two clusters come closest to each other. Other, more distant parts of the cluster and the clusters' overall structure are not taken into account.
  - ➢Local matters more!

Steps of Single-link Clustering



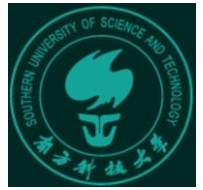
1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.
2. Find the most similar pair of clusters in the current clustering, say pair c_r, c_s, according to $d(c_r, c_s) = min\ d(c_i, c_j)$ where the minimum is over all pairs of clusters in the current clustering.
3. Increment the sequence number: $m = m + 1$. Merge clusters c_r and c_s into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d(c_r, c_s)$.
4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters c_r and c_s and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (c_r, c_s) and old cluster c_k is defined as $d\big(c_k, (c_r, c_s)\big) = min\{d(c_k, c_r), d(c_k, c_s)\}$.
5. If all objects are in one cluster, stop. Else, go to step 2.

Complete-link Clustering



- The similarity of two clusters is the similarity of their *most dissimilar* members (pessimistic).
- The entire structure of the clustering can influence merge decisions.
  - ➢Non-local!

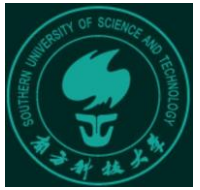
Steps of Complete-link Clustering



1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.
2. Find the most similar pair of clusters in the current clustering, say pair c_r, c_s, according to $d(c_r, c_s) = max\ d(c_i, c_j)$ where the maximum is over all pairs of clusters in the current clustering.
3. Increment the sequence number: $m = m + 1$. Merge clusters c_r and c_s into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d(c_r, c_s)$.
4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters c_r and c_s and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (c_r, c_s) and old cluster c_k is defined as $d\big(c_k, (c_r, c_s)\big) = max\{d(c_k, c_r), d(c_k, c_s)\}$.
5. If all objects are in one cluster, stop. Else, go to step 2.

k-means and Other Cost Minimization Clustering

k-means Clustering

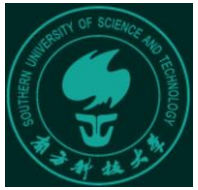


- Simple but most popular partitional algorithm.
- Assume Euclidean space.
- k clusters: $C_1, C_2, \ldots, C_k$
- Minimise the sum of squared distances to the centroid of clusters over all k clusters:

$$\min_{\{C_1, C_2, \ldots, C_k\}} \sum_{i=1}^{k} \sum_{x \in C_i} \left|\left| \boldsymbol{x} - \boldsymbol{\mu_i} \right|\right|_2^2,$$

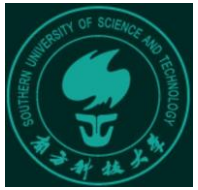
where $\boldsymbol{\mu_i} = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$ and $\{\boldsymbol{x}\}$ are d-dimensional data points.

Another objective function



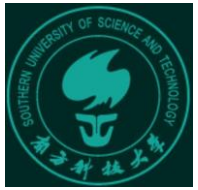
- **k-medoid [Kaufman and Rousseeuw, 2005]**
  - Clusters are represented using the median of the data instead of the mean.

k-means Clustering



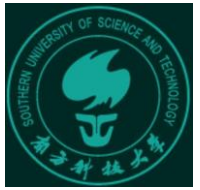
- This is a **search** problem.
  - We have a clear objective function to minimise (objective space).
  - We search for the optimal clusters (solution space).
- A NP-hard problem even for $k = 2$!
- The k-means clustering (greedy) algorithm will converge to a local optimum.

Main Procedure [Jain and Dubes, 1988]



1. Select an initial partition with k clusters.
   - **Example:**
     - Pick one point uniformly at random, then $k-1$ other points, each as far away as possible from the previous points. These k points are used as centroids.
     - For each point, place it in the cluster whose current centroid it is nearest.
     - After all points are assigned, update the locations of centroids of the k clusters.
2. Generate a new partition by assigning each pattern to its closest cluster center.
3. Compute new cluster centers.
4. **Repeat** 2 and 3 until cluster membership stablises (points don't move between clusters).

Issues of k-means Clustering



- The initialisations matter!
  - Think about the same issue of hill climbing algorithm.
  - [Question] How could you solve this?
- The distance measure.
  - Should figure out which distance measure could be more suitable for a given data set.
- The choice of cluster number k.
  - No theoretical support.

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.

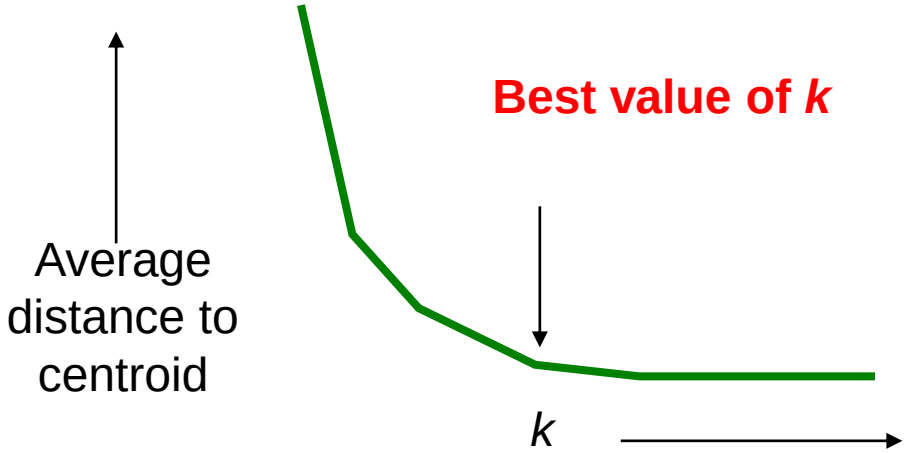
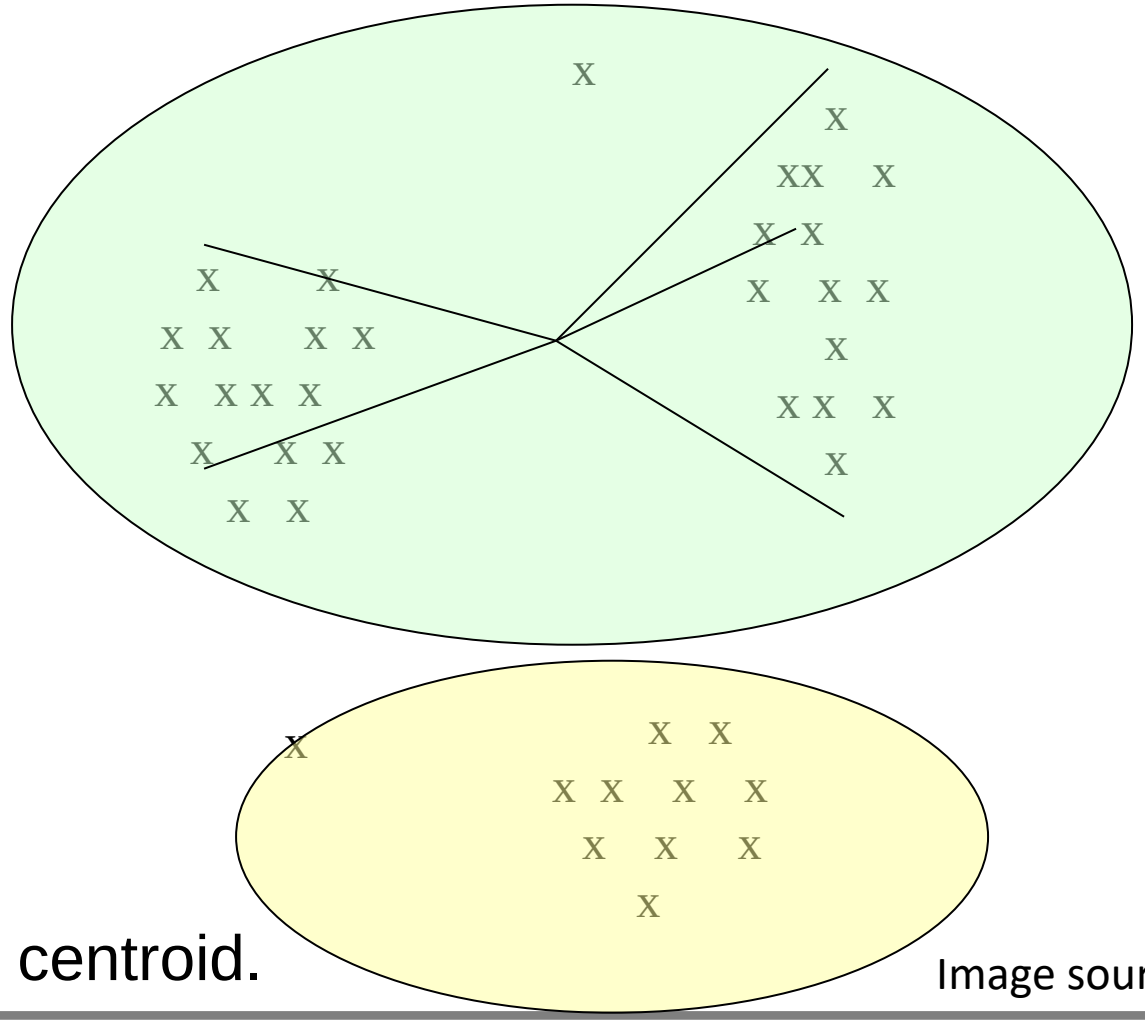


Image source: [2].

Example: Picking k



Too few; many long distances to centroid.

Image source: [2].

Example: Picking k

Just right; distances rather short.

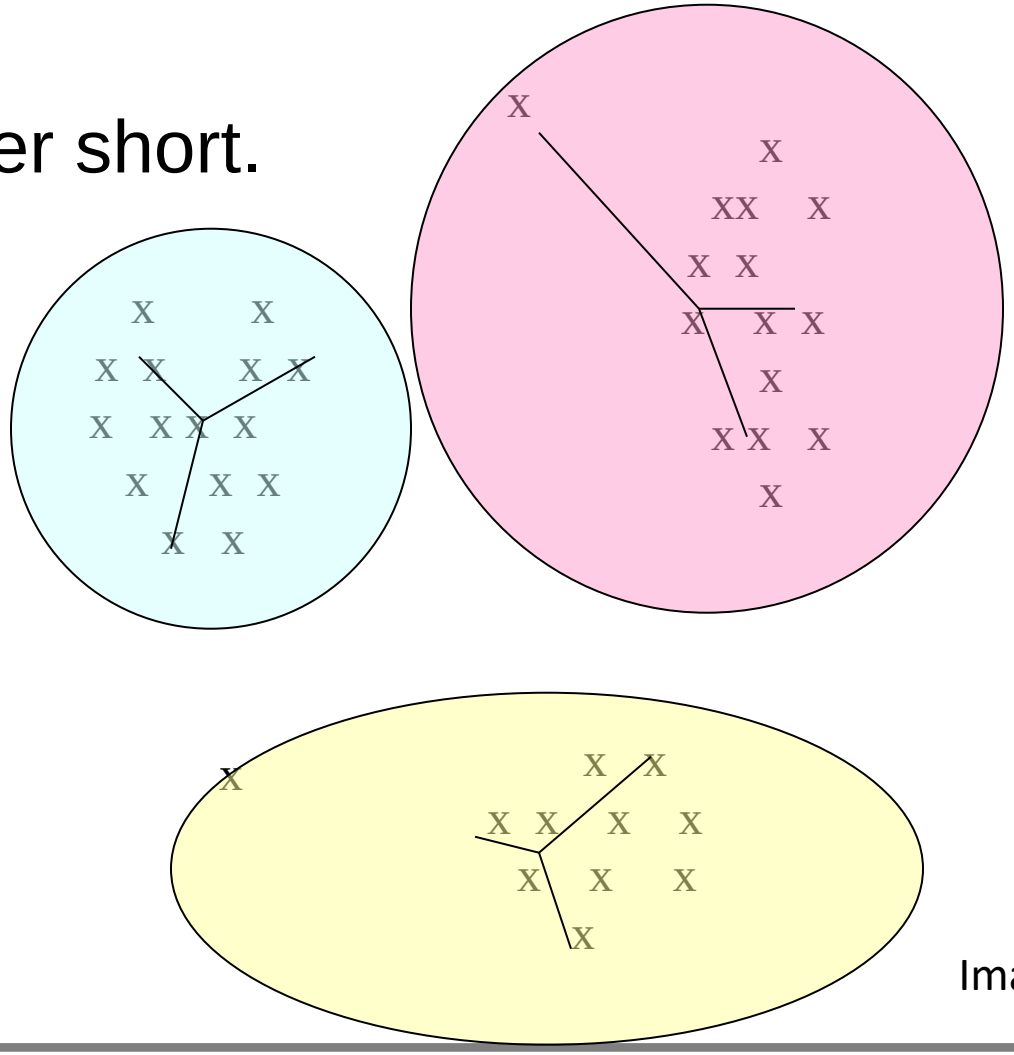


Image source: [2].

Example: Picking k

Too many; little improvement in average distance.

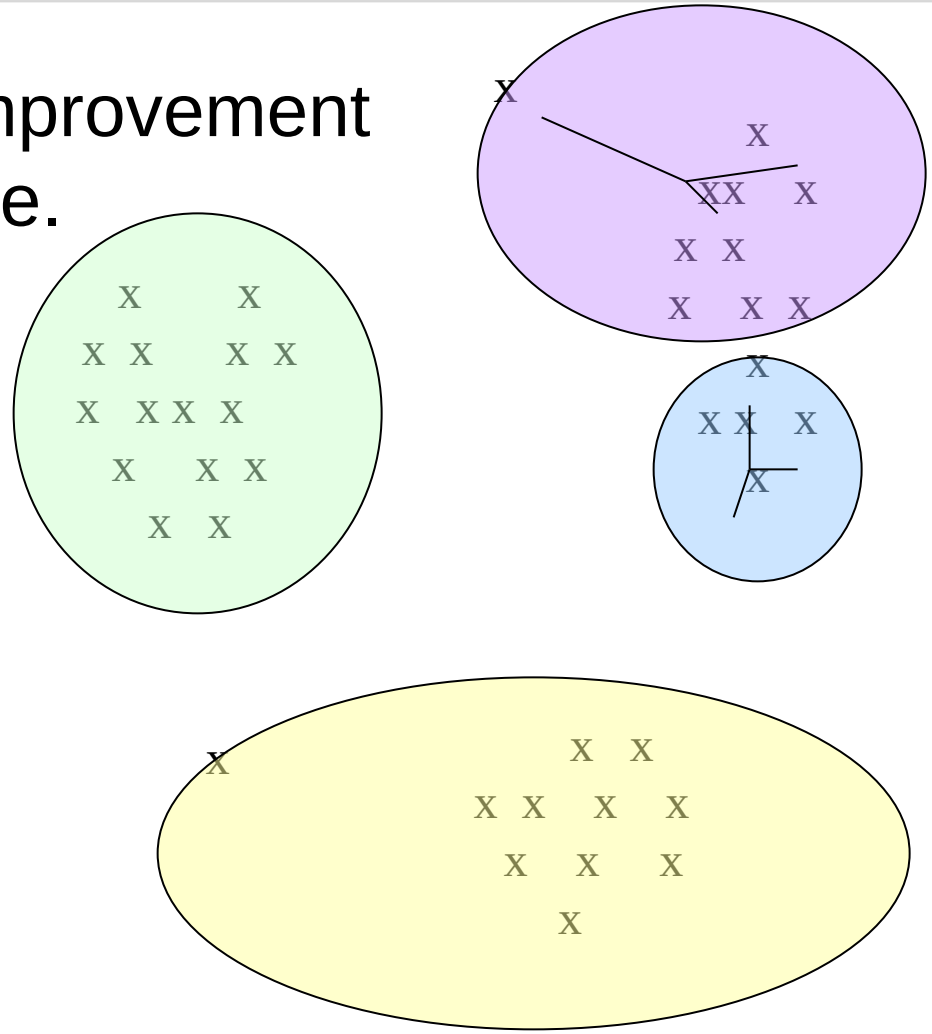


Image source: [2].

Spectral Clustering

- Graph Cut
- Minimising Normalised Cut
- Graph Laplacian

Spectral Clustering

- Sometimes called **graph theoretic clustering**.
  - Data points are represented as nodes in a weighted graph.
  - The edges are weighted by their pairwise similarity.
- Core idea: partition the nodes into two subsets A and B s.t. the cut size (the sum of the weights assigned to the edges connecting between nodes in A and B) is minimised.

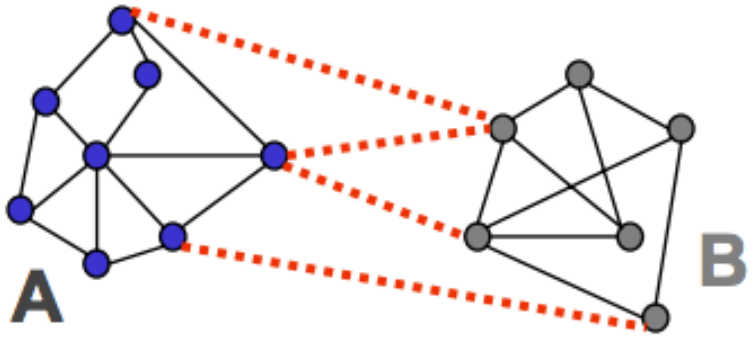
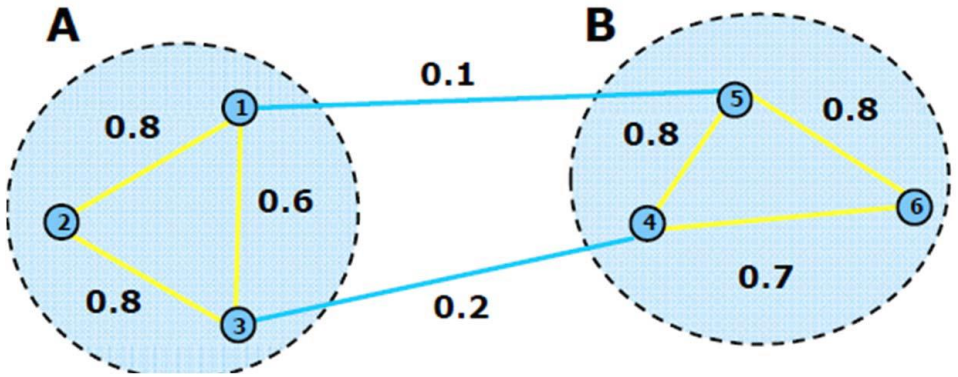


Image source: https://www.stat.washington.edu/wxs/Stat593-s03/Student-presentations/NormalizedCuts.pdf

Spectral Clustering

$$CutSize(A, B) = \sum_{x \in A} \sum_{x' \in B} weight(x, x')$$



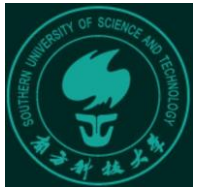
How to Create the Graph ?

- It is common to use **a Gaussian Kernel** to compute similarity between objects

$$weight(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\frac{-|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{\sigma^2}$$

- **One could create:**
  - a fully connected graph;
  - k-nearest neighbour graph (each node is only connected to its k-nearest neighbours);
  - ϵ-neighborhood graph.

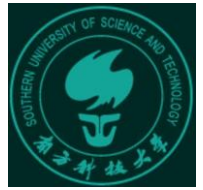
Spectral Clustering Issues (1/2)



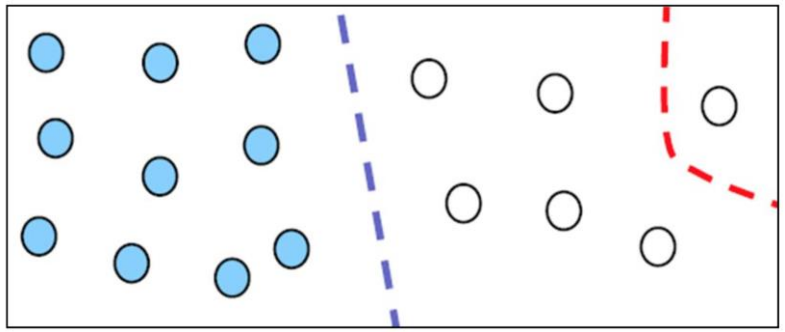
- An intuitive goal is find the partition that minimises the cut.
- Initial algorithms solved this problem using the minimum cut algorithm: minimise weight of connections between groups.

$$\min_{A \cap B = \emptyset, A \cup B = V} Cut(A, B)$$

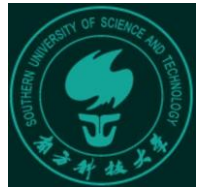
Spectral Clustering Issues (2/2)



- An intuitive goal is find the partition that minimises the cut.
- Initial algorithms solved this problem using the minimum cut algorithm, which often results in clusters of imbalanced sizes.
  - Prefer degenerate solution (e.g. the red partition).
  - Need to express preference for more balanced solution.



Normalised Cut (Ncut) [Shi and Malik, 2000]

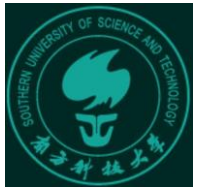


- An efficient approximate graph-cut based clustering algorithm.
- Instead of looking at the value of total edge weight connecting A and B, the proposed measure computes the cut cost as a fraction of the total edge connections. This is called **disassociation measure the normalised cut**:

$$Ncut(A,B) = \frac{CutSize(A,B)}{assoc(A,V)} + \frac{CutSize(A,B)}{assoc(B,V)},$$

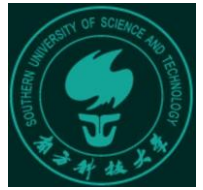
where $assoc(A,V) = \sum_{x \in A, x' \in V} weight(x,x')$ is the total connection from nodes in A to all nodes in the graph. $assoc(B,V)$ is similarly defined.

Ncut using bipartition: Main Procedure [3]



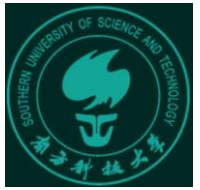
1. Given a data set, set up a weighted graph $G = (V, E)$ and set the weight on the edge connection two nodes to be a measure of the similarity between the two nodes.
2. Solve $(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{x} = \lambda \boldsymbol{D}\boldsymbol{x}$ for eigenvectors with the smallest eigenvalues.
   1. $\boldsymbol{W}$ is an $N \times N$ symmetrical matrix with $\boldsymbol{W}(i,j) = \boldsymbol{weight}(i,j)$.
   2. $\boldsymbol{D}$ is an $N \times N$ diagonal matrix with $\boldsymbol{d}$ on its diagonal, and $\boldsymbol{d}(i) = \sum_j \boldsymbol{weight}(i,j)$
3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
4. Decide if the current partition should be subdivided and recursively repartition the segmented parts if necessary.

Creating Bipartition Using 2nd Eigenvector



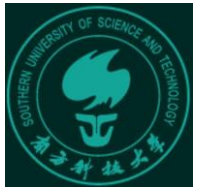
- Sometimes there is not a clear threshold to split based on the second vector since it takes continuous values
- **How to choose the splitting point?**
  I. Pick a constant value (0, or 0.5).
  II. Pick the median value as splitting point.
  III. Look for the splitting point that has the minimum *Ncut* value:
    1. Choose n possible splitting points.
    2. Compute *Ncut* value.
    3. Pick minimum.

K-way Partition?



- **Recursive bi-partitioning** (Hagen et al., 1991)
  - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner.
  - Disadvantages: Inefficient, unstable.
- **Cluster multiple eigenvectors**
  - Build a reduced space from multiple eigenvectors.
  - Commonly used in recent papers.
  - A preferable approach... its like doing dimension reduction then *k*-means.

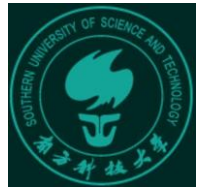
Advantages of Ncut



- Consider the connectivity between groups relative to the volume of each group.
- The cut that partitions out small isolated points will no longer have small Ncut value, since the cut value will almost certainly be a large percentage of the total connection from that small set to all other nodes.

- However, minimising normalised cut is NP-complete!

Spectral clustering (Ng, *et. al*, 2001)



- Form the affinity matrix $\boldsymbol{W}$

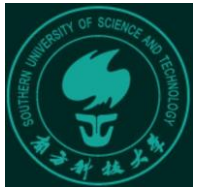
$$-\boldsymbol{W}(i,j) = \exp\left(-\frac{\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^2}{2\sigma}\right), \boldsymbol{W}(i,i) = 0$$

- Compute the degree matrix $\boldsymbol{D} = diag(\boldsymbol{W}\boldsymbol{1})$.
- Compute the normalized graph Laplacian

$$\boldsymbol{L} = \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$$

- Find the *k* largest eigenvectors, for new data matrix $\boldsymbol{X}'_{n\times k}$.
- Normalize the rows to have unit length.
- Treating each row as a data point in the *k*-d space and cluster the data into *k* clusters via k-means.

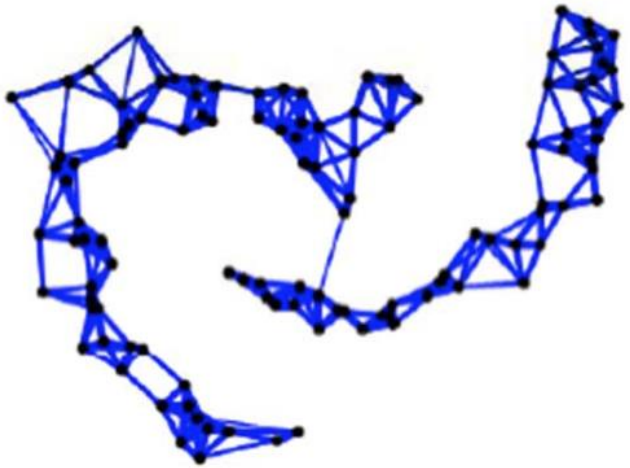
Why?



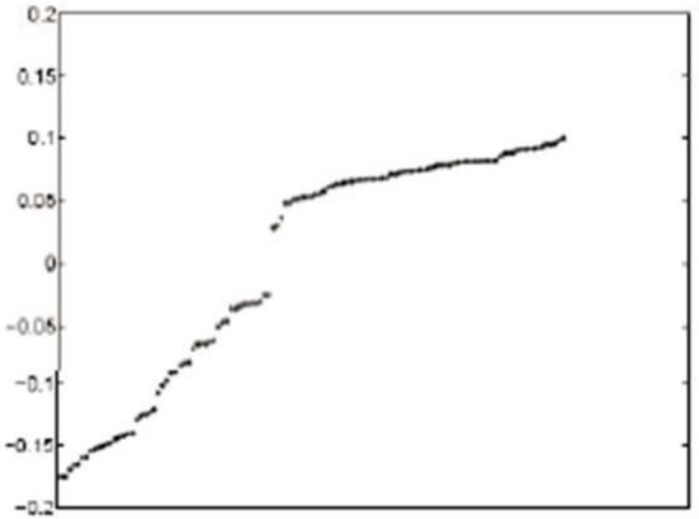
- If we eventually use *K*-means, why not just apply *K*-means to the original data?

- **This method allows us to cluster non-convex regions.**

Example



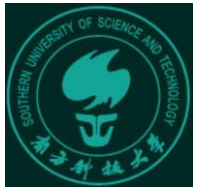
5-NN graph



2nd eigen-vector

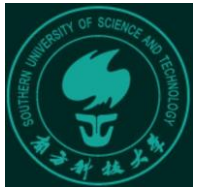
Clustering result

Summary



- **Clustering**: Given a set of data points, with a notion of **similarity** between points, group the points into some number of clusters (groups).
- **Algorithms**:
  - Agglomerative hierarchical clustering:
  - k-means:
  - Spectral Clustering
  - Minimising *Normalised* Cut

Reading Materials for This Lecture



- [1] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern recognition letters, 31(8), 651-666.
- [2] J. Leskovec, A. Rajaraman, J. Ullman, Mining of Massive Datasets (Chapter 7 Clustering), Stanford University, http://www.mmds.org
- [3] Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(8), 888-905.
- [4] Ohad Shamir and Naftali Tishby (2008). Cluster Stability for Finite Samples. NIPS, 2008.
- https://scikit-learn.org/stable/modules/clustering.html
- https://nlp.stanford.edu/IR-book/html/htmledition/single-link-and-complete-link-clustering-1.html
- **[BFR algorithm]** P.S. Bradley, U.M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," *Proc. Knowledge Discovery and Data Mining*, pp. 9–15, 1998.
- **[CURE algorithm]** S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pp. 73–84, 1998.
- **[Spectral clustering]** Von Luxburg U. A tutorial on spectral clustering. *Statistics and computing*, 2007, 17(4): 395-416.