

Clustering

Predictive Analytics

Acknowledgments

These slides draw upon and adapt selected materials by:

- **Fragkiskos D. Malliaros** (CentraleSupélec, Université Paris–Saclay, France)

Unsupervised learning (clustering)

Supervised vs. Unsupervised Learning

Supervised learning

- We have labeled examples
- Given those examples, learn a model that can generalize to unseen examples
- Key tasks:
 - Classification
 - Regression

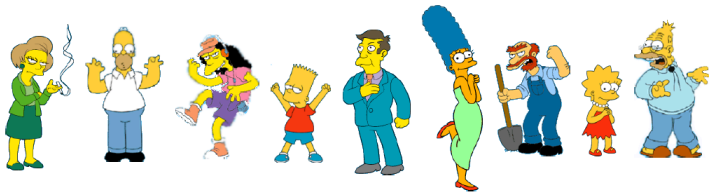
Unsupervised learning

- The data is unlabeled
- Given the data, learn a model that identifies structure in the data (and generalize to new data)
- Key task:
 - Clustering

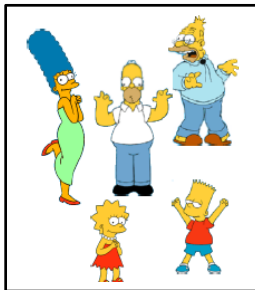
What is Cluster Analysis?

- Cluster: a collection of data objects
 - **Similar** (or related) to one another within the same group
 - **Dissimilar** (or unrelated) with objects in other groups
- Cluster analysis (or clustering)
 - Finding **similarities** between data according to the characteristics of the data and ...
 - ... **grouping** similar data objects into clusters
- Typical applications
 - As a **stand-alone tool** to get further insights about the data
 - As a **preprocessing step** for other algorithms

Any Natural Grouping?



Clustering is subjective



Simpson's family



School employees



Females



Males

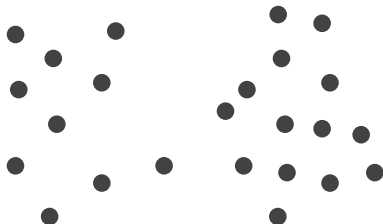
What is a Good Clustering?

- Good clusters have:
 - High **intra-cluster** similarity: **cohesive** within clusters
 - Low **inter-cluster** similarity: **distinctive** between clusters
- The quality of a clustering method depends on
 - The **similarity measure** used by the method
 - Its ability to discover some or all of the hidden patterns

Recall the distance and similarity measures covered in kNN classification

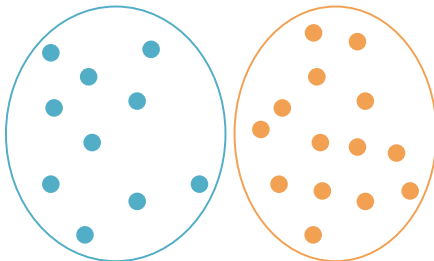
Goals of Clustering

- Group objects that are similar into **clusters**: classes that are unknown beforehand



Goals of Clustering

- Group objects that are similar into **clusters**: classes that are unknown beforehand

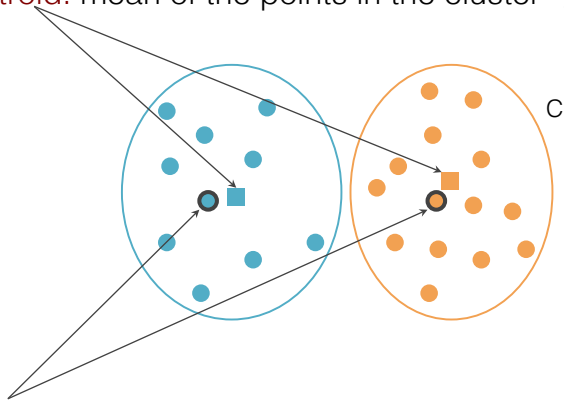


Applications of Clustering

- **Understand** general characteristics of the data
- **Visualize** the data
- **Infer** some properties of a data point based on how it relates to other data points
- Examples
 - Find subtypes of diseases
 - Visualize protein families
 - Find categories among images
 - Find patterns in financial transactions
 - Detect communities in social networks
 - Find users with similar interests (e.g., Netflix, Amazon)

Cluster Centroids and Medoids

- Centroid:** mean of the points in the cluster $\mu = \frac{1}{|C|} \sum_{x \in C} x$



- Medoid:** point in the cluster that is closest to the centroid

$$m = \arg \min_{x \in C} d(x, \mu)$$

Cluster Evaluation

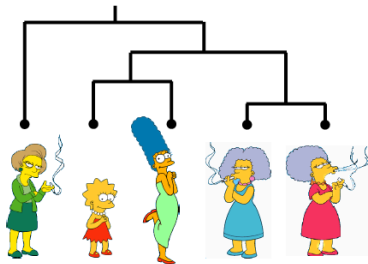
- Clustering is **unsupervised**
- There is no **ground truth**. How do we evaluate the quality of a clustering algorithm?
- Based on the **shape** of the clusters:
 - Points within the same cluster should be nearby/similar and points far from each other should belong to different clusters
- Based on the **stability** of the clusters:
 - We should get the same results if we remove some data points, add noise, etc.
- Based on **domain knowledge**
 - The clusters should “make sense”

Major Clustering Approaches

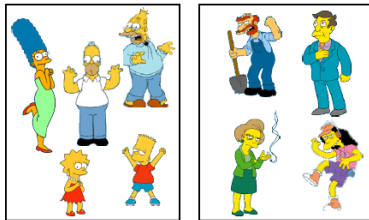
- **Partitioning approach**
 - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
 - Typical methods: k-means, k-medoids
- **Hierarchical approach**
 - Create a hierarchical decomposition of the set of data (or objects) using some criterion
 - Typical methods: Diana, Agnes, BIRCH, CHAMELEON
- **Density-based approach**
 - Based on connectivity and density functions
 - Typical methods: DBSCAN, OPTICS, DenClue
- **Grid-based approach**
 - Based on a multiple-level granularity structure
 - Typical methods: STING, WaveCluster, CLIQUE

Hierarchical vs. Partitional

Hierarchical



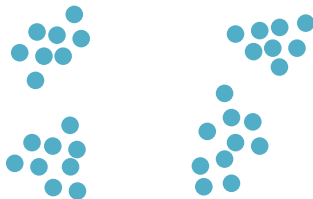
Partitional



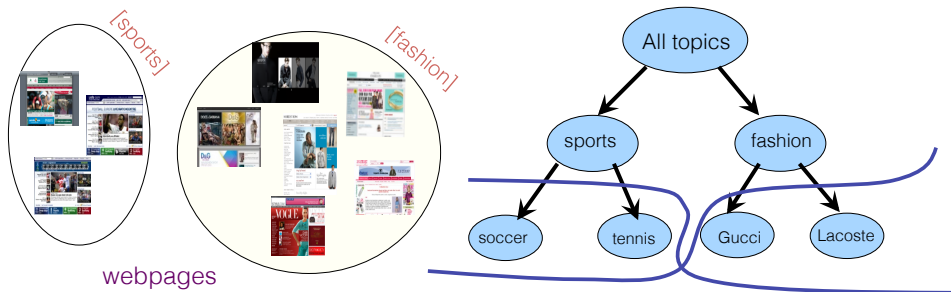
Hierarchical clustering

Hierarchical Clustering

- Group data over a variety of possible scales, in a multi-level hierarchy



Hierarchical Clustering (2/2)



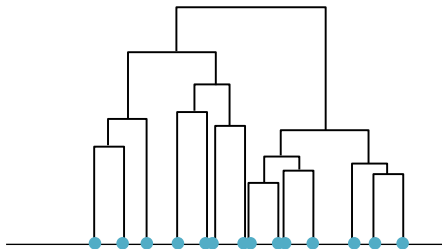
- A hierarchy might be more natural
- Different users might care about different levels of granularity or even prunings

Construction

- **Agglomerative** approach (**bottom-up**)
 - Start with each element in its own cluster
 - Iteratively **join** neighboring clusters
- **Divisive** approach (**top-down**)
 - Start with all elements in the same cluster
 - Iteratively **separate** into smaller clusters

Dendrogram

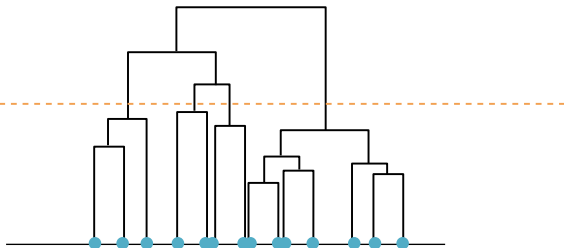
- The results of a hierarchical clustering algorithm are presented in a **dendrogram**



Dendrogram

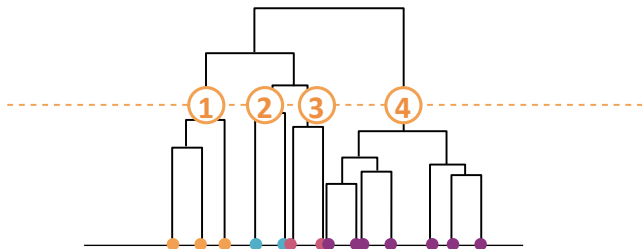
- The results of a hierarchical clustering algorithm are presented in a **dendrogram**

How many clusters do I have?



Dendrogram

- The results of a hierarchical clustering algorithm are presented in a **dendrogram**



Hierarchical Clustering

- Advantages

- No need to pre-define the number of clusters
- Interpretability

- Drawbacks

- Computational complexity
- Must decide at which level of the hierarchy to split
- Lack of robustness (unstable)

k-means clustering

k-means Algorithm – The Idea

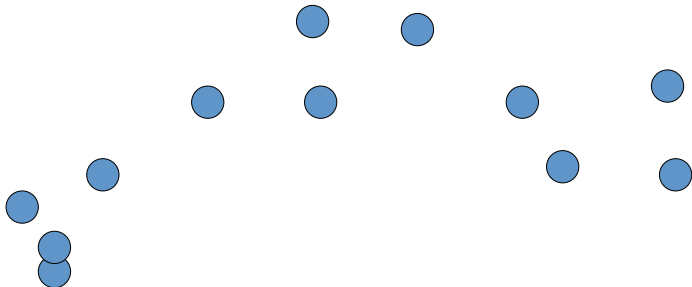
Most well-known and popular clustering algorithm:

1. Start with some initial cluster centers

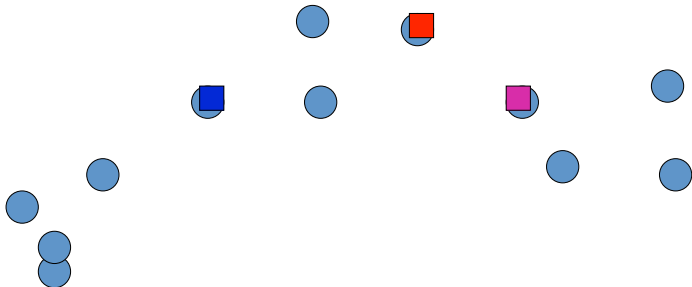
2. Iterate:

- Assign each example to closest center
- Recalculate centers as the mean of the points in a cluster

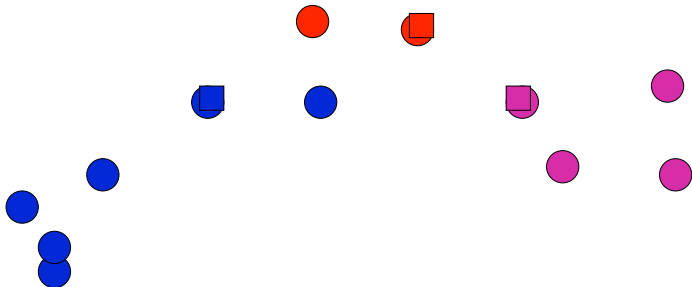
k-means: An Example



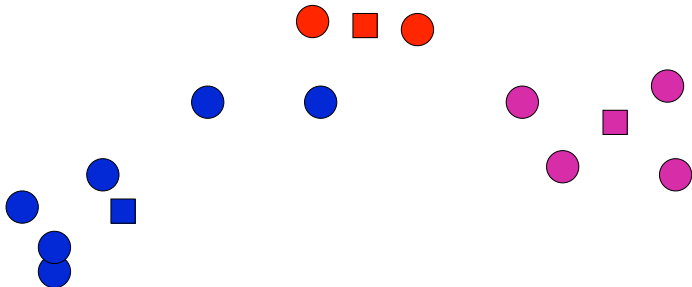
k-means: Initialize Centers Randomly



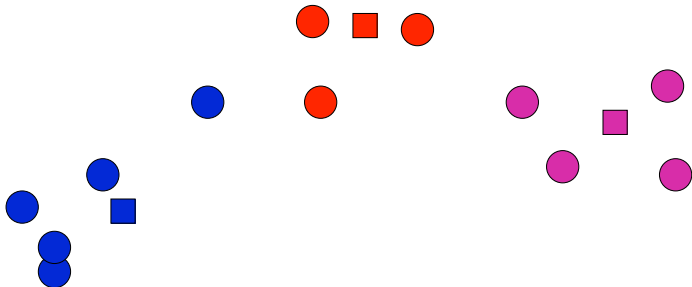
k-means: Assign Points to Nearest Center



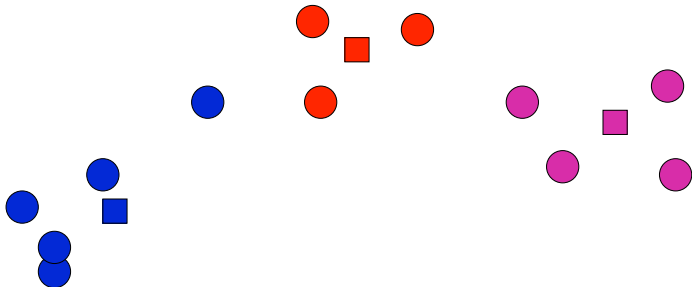
k-means: Readjust Centers



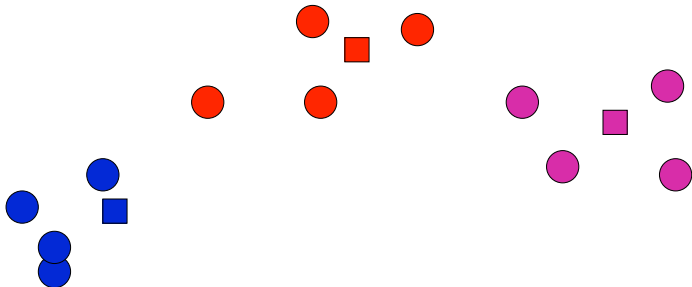
k-means: Assign Points to Nearest Center



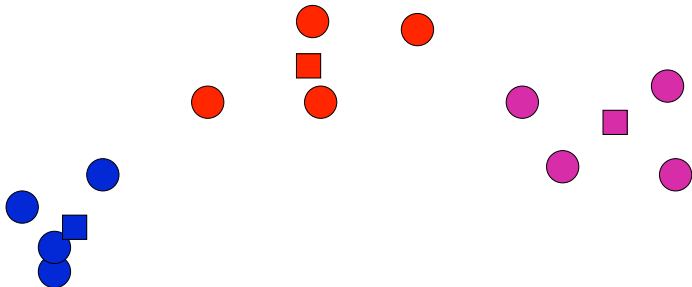
k-means: Readjust Centers



k-means: Assign Points to Nearest Center

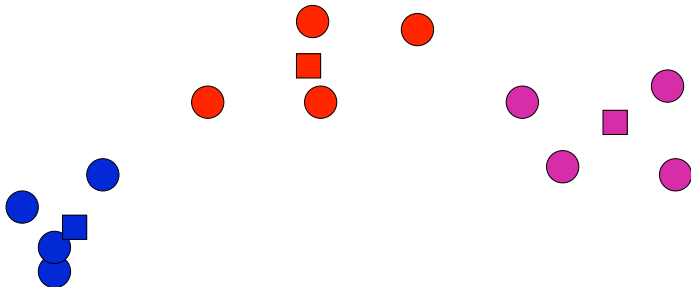


k-means: Readjust Centers



k-means: Assign Points to nearest Center

No changes: Done



Test demo at: <http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html>

k-means Clustering – Objective Function

- Minimize the **intra-cluster variance**
 - Within-cluster sum of squares

$$\text{Var}_{\text{in}}(C) = \frac{1}{|C|} \sum_{x \in C} \|x - \mu_C\|^2 \quad \text{For a cluster } C$$

$$V = \sum_{k=1}^K \sum_{x \in C_k} \frac{1}{|C_k|} \|x - \mu_{C_k}\|^2 \quad \text{For all clusters}$$

- What will this partition of the space look like?

k-means Clustering – Objective Function

- Minimize the **intra-cluster variance**
 - Within-cluster sum of squares

$$\text{Var}_{\text{in}}(C) = \frac{1}{|C|} \sum_{x \in C} \|x - \mu_C\|^2 \quad \text{For a cluster } C$$

$$V = \sum_{k=1}^K \sum_{x \in C_k} \frac{1}{|C_k|} \|x - \mu_{C_k}\|^2 \quad \text{For all clusters}$$

- For each cluster, the points in that cluster are those that are closest to its centroid than to any other centroid

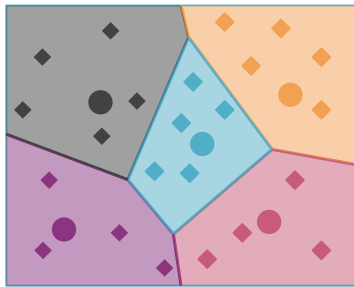
k-means Clustering – Objective Function

- Minimize the **intra-cluster variance**
 - Within-cluster sum of squares

$$\text{Var}_{\text{in}}(C) = \frac{1}{|C|} \sum_{x \in C} \|x - \mu_C\|^2 \quad \text{For a cluster } C$$

$$V = \sum_{k=1}^K \sum_{x \in C_k} \frac{1}{|C_k|} \|x - \mu_{C_k}\|^2 \quad \text{For all clusters}$$

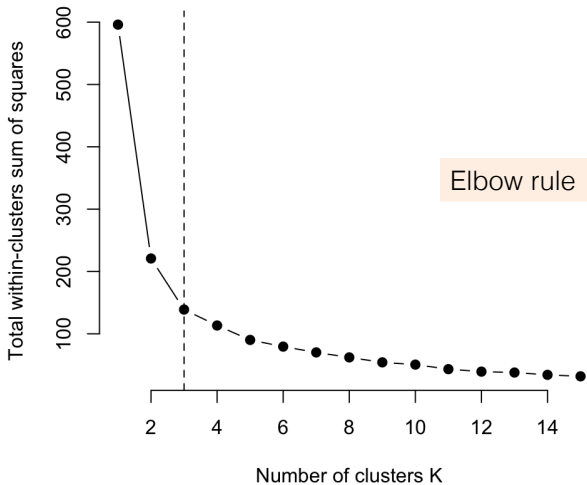
- Voronoi **tessellation**
- Optimal solution: hard problem



Lloyd's Algorithm for k-means

- k-means cannot be easily optimized
- We adopt a **greedy strategy** (Lloyd's Algorithm)
 - Partition the data into **k** clusters at random
 - Compute the centroid of each cluster
 - Assign each point to the cluster of the centroid
 - Repeat until cluster membership converges

How to Select k?



Summary of k-means

- Advantages

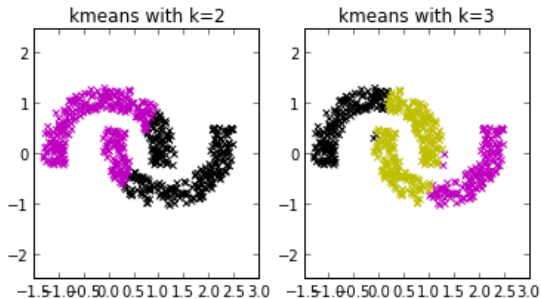
- Computational time is linear: $\mathcal{O}(npkt)$ t : number of iterations
- Easily implementable

↑
compute kn distances
in p dimensions

- Drawbacks

- Need to select k (user-defined parameter)
- Sensitivity to noise and outliers
- Non-deterministic (stochastic)
 - Different solutions with each iteration
- The clusters are forced to have “spherical” (convex) shapes

Example



Source: <https://pafnuty.wordpress.com/>

Improving k-means: k-means++

- The quality of the solution depends on the **initialization**
- Rationale behind random initialization
 - Choosing a random assignment may lead the algorithm to a good local minimum
- Another approach: **k-means++** [Arthurs and Vassilvitskii '07]
 1. Select a random point and declare it centroid \mathbf{c}_1
 2. For all remaining data points \mathbf{x}_j compute distance $d(\mathbf{x}_j, \mathbf{c}_1)$
 3. Select a random point with probability proportional to $d(\mathbf{x}_j, \mathbf{c}_1)^2$ and set it as \mathbf{c}_2
 - This will give a point far enough from \mathbf{c}_1
 4. Repeat steps 2 and 3 for all centroids \mathbf{k}

scikit-learn



<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

[http://scikit-learn.org/stable/auto_examples/cluster/
plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py](http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py)

scikit-learn

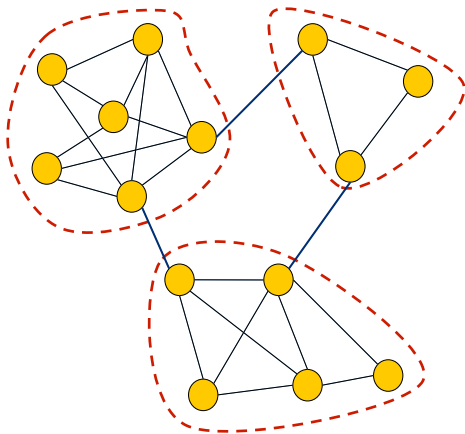


<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

[http://scikit-learn.org/stable/auto_examples/cluster/
plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py](http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py)

Spectral clustering

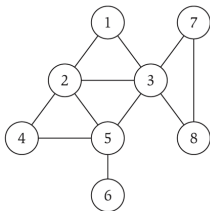
Clustering Structure in Graphs



How to discover the clustering structure?

Graph Representation: Adjacency Matrix

- A graph can be represented by the adjacency matrix A
 - Matrix of size $n \times n$, where $n=|V|$ is the number of nodes
 - $A_{ij} > 0$, if i and j are connected
 - $A_{ij} = 0$, if i and j are not connected
 - In case of **unweighted** graphs, $A_{ij} = 1$, if (i, j) is an edge of the graph
 - Space proportional to n^2



Undirected graph

Node indexing

	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Node indexing

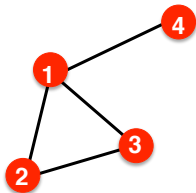
Adjacency matrix

Graph Representation: Laplacian Matrix

- Let $G = (V, E)$ be a graph. Then, the **Laplacian matrix** is defined as

$$\mathbf{L}_{ij} = \begin{cases} k_i & \text{if } i = j \\ -1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

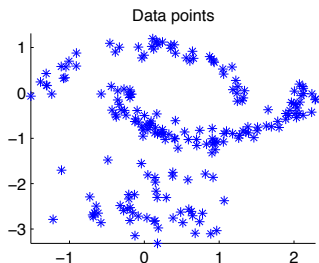
- Diagonal degree matrix \mathbf{D} , where $\mathbf{D}_{ii} = k_i$ (node degrees)



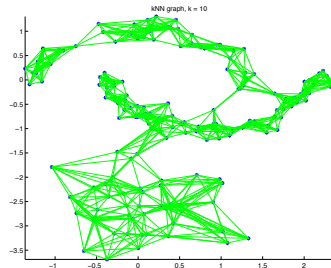
$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

Clustering Non-Graph Data

- Apply graph clustering algorithms on data with no inherent graph structure (e.g., points in a d -dimensional Euclidean space)
- **How?**
 1. Construct a **similarity graph** based on the topological relationships and distances between data points (e.g., kNN graph)
 2. Then, the problem of clustering the set of data points is transformed to a graph clustering problem

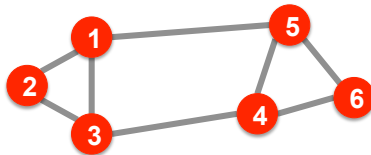


Similarity graph
(e.g., kNN)

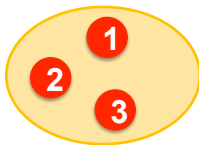


Graph Partitioning (1/2)

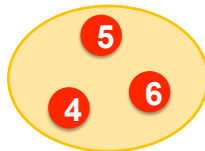
- Undirected graph $G=(V, E)$
- Bi-partitioning task:
 - Divide nodes into two disjoint groups A, B



A



B

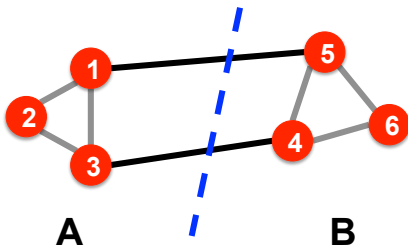


Questions:

- How can we define a **good** partition of G ?
- How can we efficiently identify such a partition?

Graph Partitioning (2/2)

- What makes a **good partition**?
 - Maximize the number of within-group connections
 - Minimize the number of between-group connections

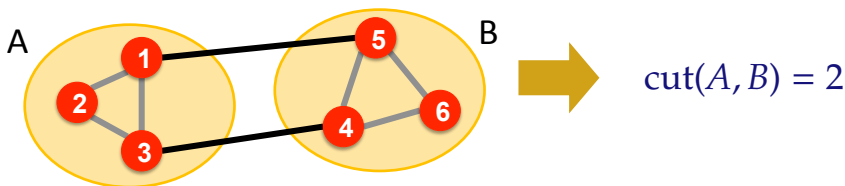


Graph Cuts

- Express partitioning objectives as a function of the **edge cut** of the partition
- Cut:** Set of edges across two groups:

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

Two partitions, A and B

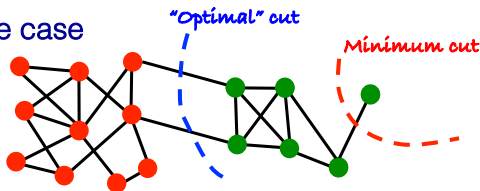


Graph Cut Criterion for Clustering

- Criterion: **Minimum-cut**
 - Minimize the weight of connections between groups

$$\arg \min_{A,B} \text{cut}(A, B)$$

- Degenerate case



Problem

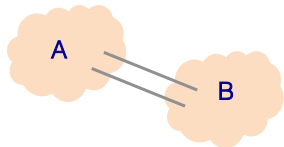
- Not satisfactory partition – often isolated nodes
- Does not consider internal cluster connectivity

Ratio Cut

Normalize cut by the **size** of the groups

$$\text{ratio-cut}(A, B) = \frac{\text{cut}(A, B)}{|A|} + \frac{\text{cut}(A, B)}{|B|}$$

Size of A and B



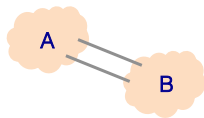
Internal group
connectivity is not
taken into account

Normalized Cut

- Criterion: **Normalized cut**
 - Connectivity between groups relative to the density of each group

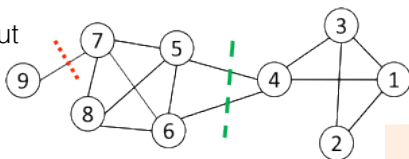
$$\text{normalized-cut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)}$$

- **Vol(A)**: total weighted degree of the nodes in **A**, i.e., $\sum_{i \in A} k_i$
- Why use this criterion?
 - It produces more **balanced partitions**
- How do we efficiently find a good partition?
 - Computing the optimal cut is **NP-hard**



Ratio Cut vs. Normalized Cut (1/2)

Red is Min-Cut



$$\text{ratio-cut}(A, B) = \frac{\text{cut}(A, B)}{|A|} + \frac{\text{cut}(A, B)}{|B|}$$

$$\text{normalized-cut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)}$$

$$\text{Ratio-Cut}(\text{Red}) = 1/1 + 1/8 = 1.125$$

$$\text{Ratio-Cut}(\text{Green}) = 2/5 + 2/4 = 0.9$$

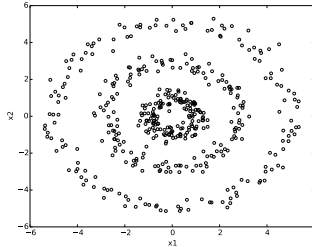
Lower value is better

$$\text{Normalized-Cut}(\text{Red}) = 1/1 + 1/26 = 1.03$$

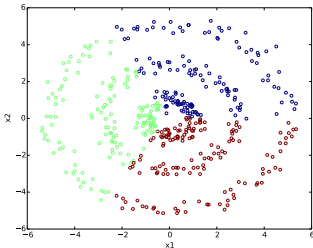
$$\text{Normalized-Cut}(\text{Green}) = 2/12 + 2/16 = 0.29$$

Normalized is even better
for **Green** due to density

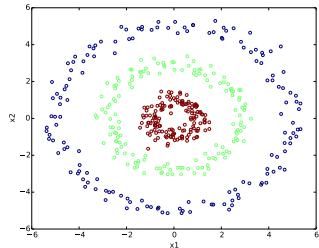
Spectral Clustering vs. k-Means



- 2-dimensional points
- Find $k=3$ clusters



k-means



spectral clustering

scikit-learn



<http://scikit-learn.org/stable/modules/clustering.html#spectral-clustering>

[http://scikit-learn.org/stable/auto_examples/cluster/
plot_cluster_comparison.html](http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html)

Comparison (scikit-learn)

