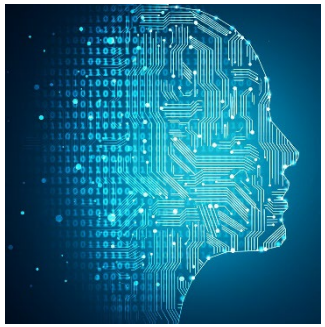
Machine Learning

Spectral Clustering

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



Clustering

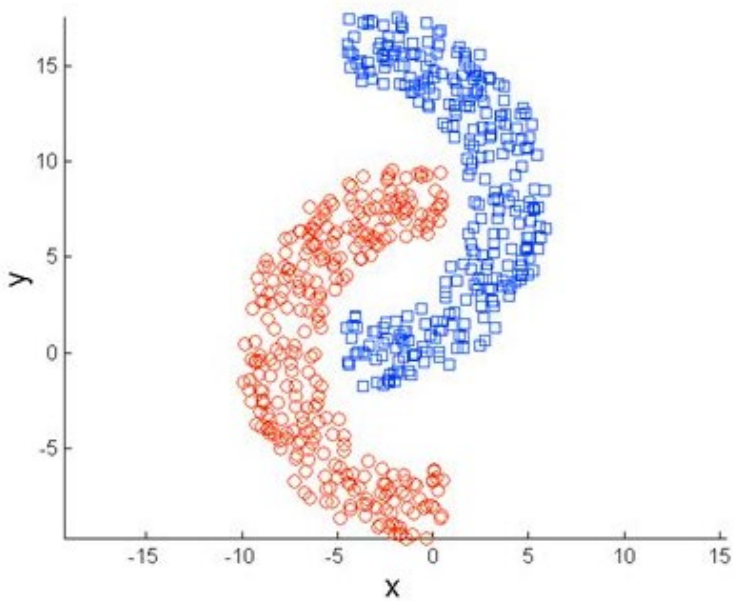
- Given data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$
  - No labels ⇒ unsupervised learning
- **Goal**: Partition the set $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ into disjoint subsets called clusters
  - Points in the same cluster are more "similar" to each other than they are to points in different clusters
  - Can be represented as a clustering map $C: \{1, \dots, n\} \to \{1, \dots, k\}$
  - The unsupervised analog to classification

# of Clusters

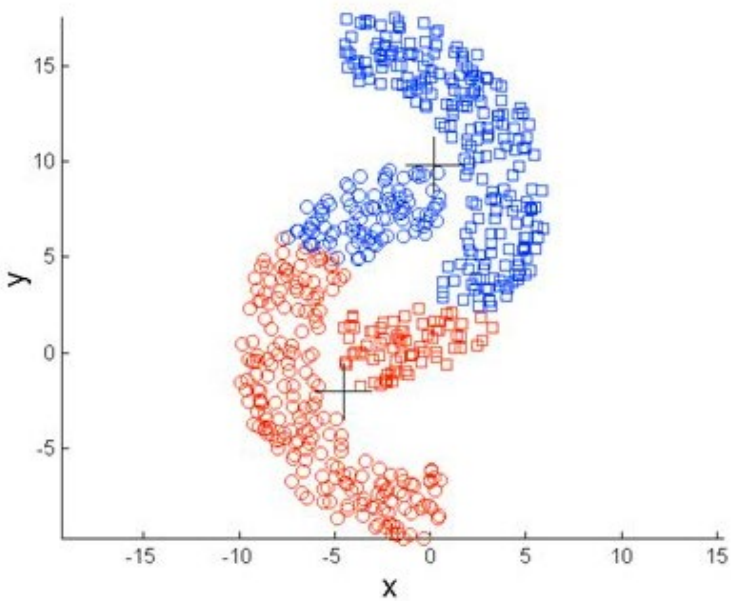
- Different definitions of "similar" lead to different algorithms

Motivation

- k-means and Gaussian Mixture Models (which we'll cover next) assume convex clusters
- Kernel k-means (k-means + the kernel trick) can find nonconvex clusters
- Spectral clustering is another method that can find nonconvex clusters



Original Points



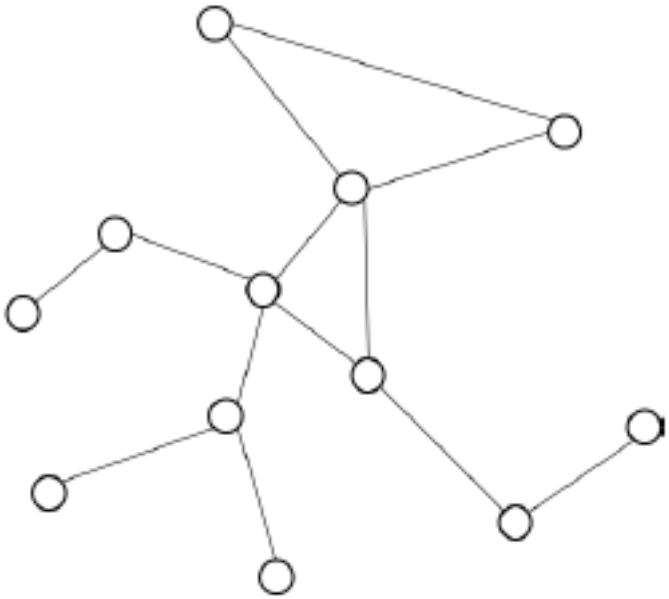
K-means (2 Clusters)

Outline

1. Graph theory
2. Similarity graphs
3. Graph Laplacian
4. Graph cuts
5. Relaxation of an optimization problem
6. Spectral clustering
7. Comparing clusterings/partitions via the ARI

Graphs

- A graph is a set of nodes/vertices V and edges E
  - $G = (V, E)$
- Vertices are often ordered for convenience
- Graphs connections described with an *adjacency matrix*
- The adjacency matrix W indicates the strength and direction of edges

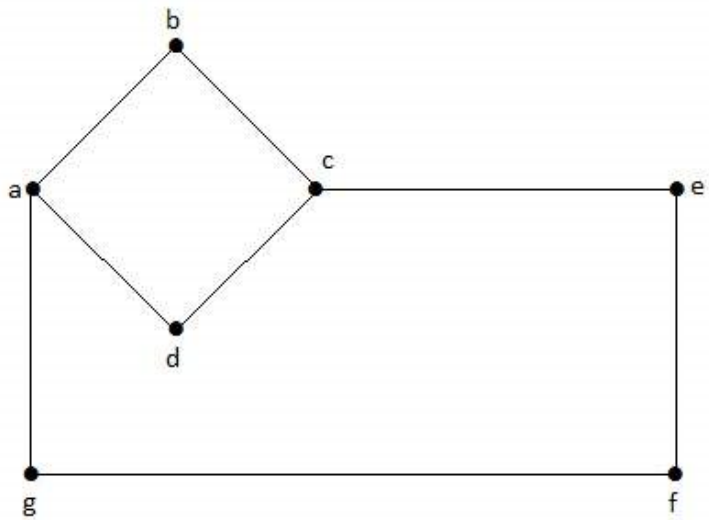


Why graphs?

- Many data are naturally described as graphs
  - Social networks, molecules, chromosome interactions, routing networks, the human body, etc.
- Most data can be represented using a *similarity graph*
  - Data points → vertices
  - Edges determined by the similarity of the data points
- Graphs have a lot of structure that can be leveraged in ML
  - E.g. the manifold assumption
- Neural networks are (directed) graphs
  - Neurons are the vertices and edges denote flow of information

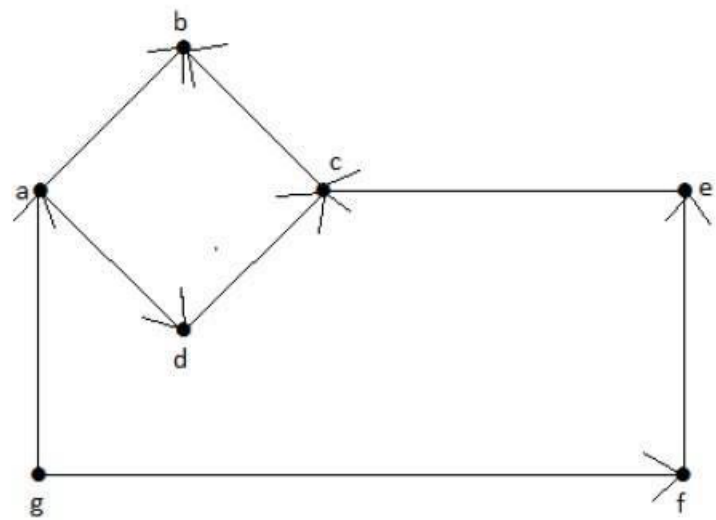
Undirected vs. directed graphs

Undirected graph



- $V = \{a, b, c, d, e, f, g\}$
- $E = \{ab, bc, cd, da, ag, gf, ef, ec\}$
- Edges are not directed
  - $ab = ba$

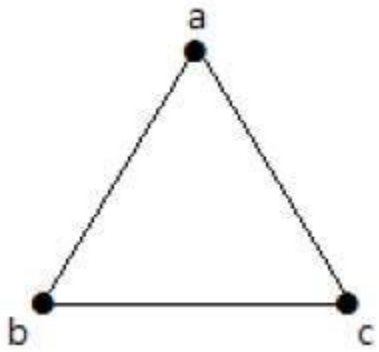
Directed graph

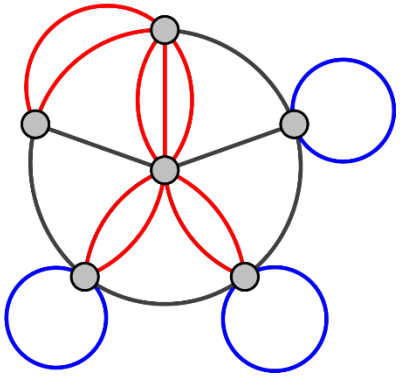


- $V = \{a, b, c, d, e, f, g\}$
- $E = \{ab, cb, dc, ad, ga, gf, fe, ec\}$
- Edges are directed
  - $ab \neq ba$

Kinds of Graphs

- Let's (mostly) consider undirected graphs for now
- We're also (for now) considering *unweighted* edges
- **Simple graph**: a graph with at most one edge between any two distinct nodes and no self-connecting edges (loops)
  - Most graphs in ML are of this type
- **Multigraph**: parallel edges and loops are allowed



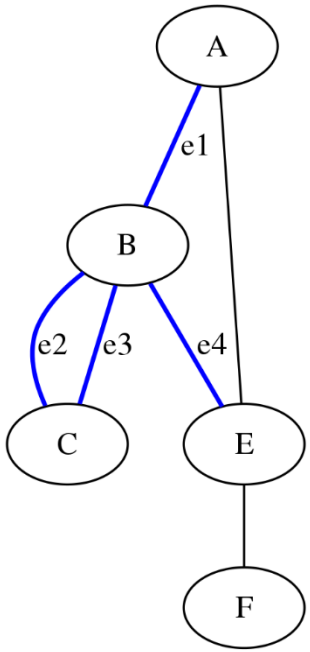


https://commons.wikimedia.org/w/index.php?curid=12247695

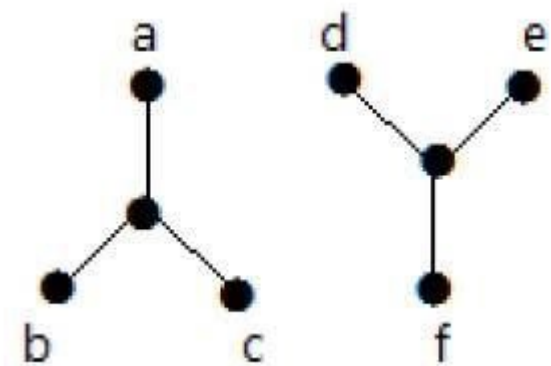
Kinds of Graphs

https://commons.wikimedia.org/w/index.php?curid=92847820

- **Walk**: a sequence of edges which joins a sequence of vertices
- **Trail**: a walk in which all edges are distinct
- **Path**: a trail in which all vertices are distinct
- **Connected graph**: there exists a path between every pair of vertices
- **Component**: an induced subgraph (a subset of vertices and all of their edges) which is connected and which is connected to no other vertices

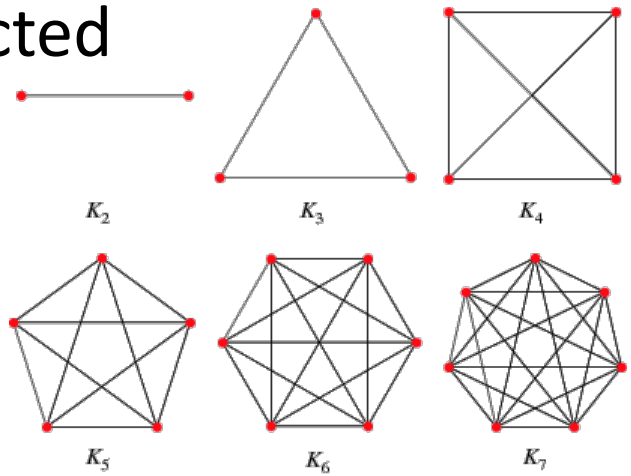


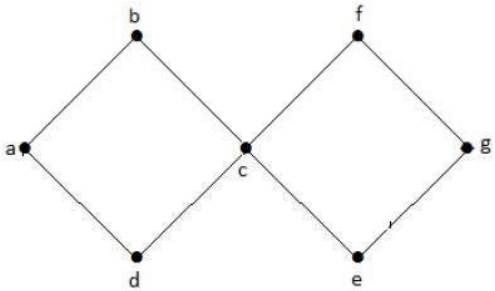
Trail from A to E, but not path (B vertex repeated)



Kinds of Graphs

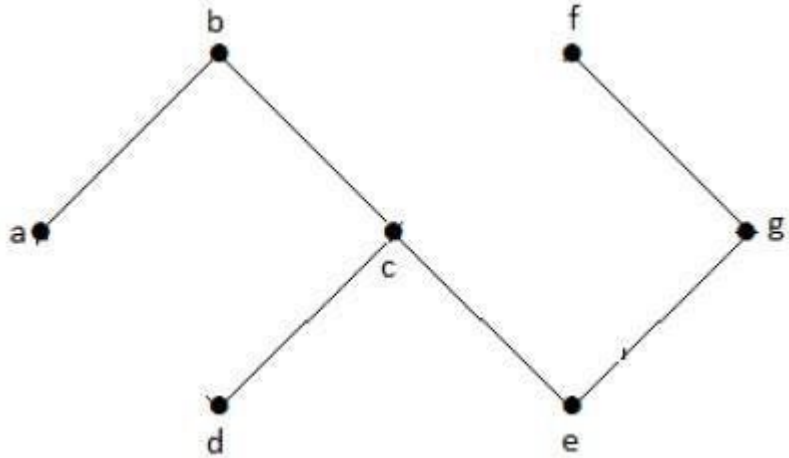
- **Degree of a vertex**: the number of edges connected to a vertex
- **Null graph:** no edges between nodes
  - Degree of each vertex is zero
- **Complete graph**: all nodes are connected to all other nodes
  - Degree of each vertex is $n - 1$
- **Cycle**: a path that starts and ends with the same vertex
- **Cyclic graph**: a graph that contains at least one cycle



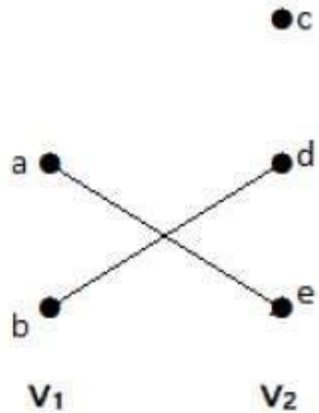


Kinds of Graphs

- **Acyclic Graph**: a graph that contains no cycles
- **Tree**: a connected, undirected acyclic graph
  - Applications include decision trees, random forests, data structures



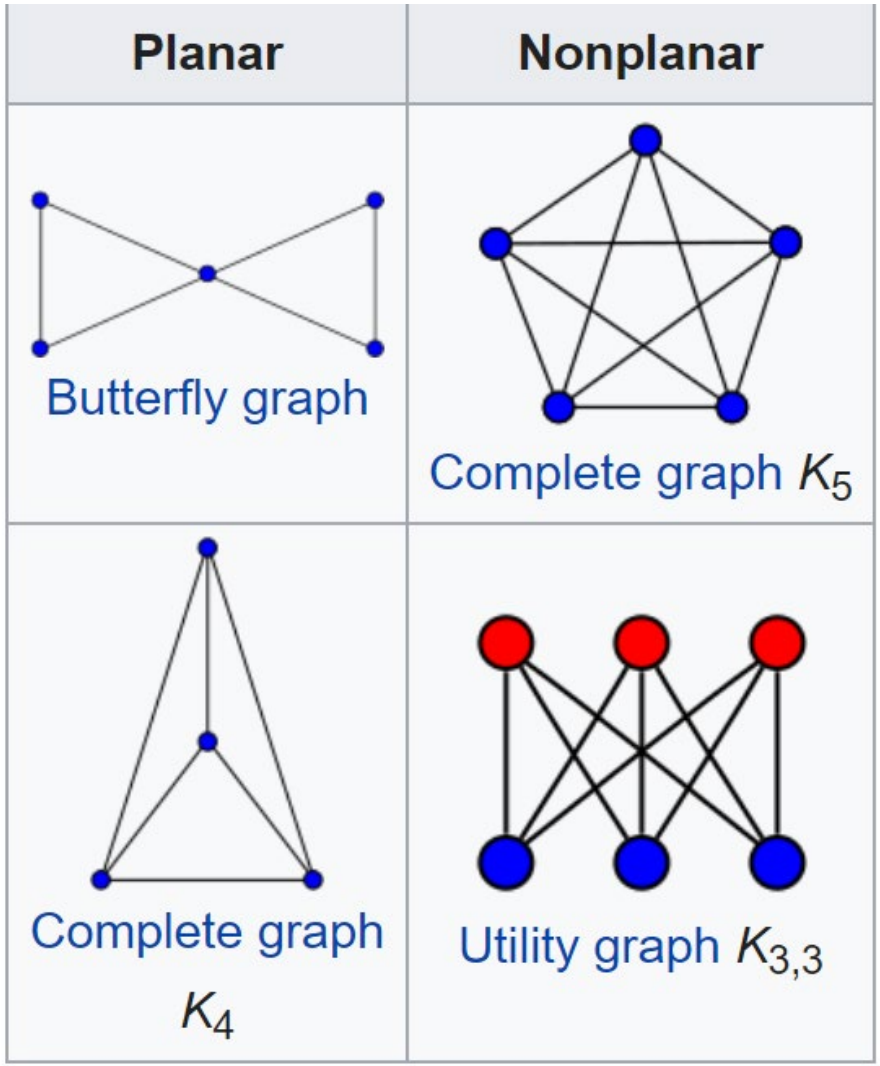
- **Bipartite graph**: a simple graph with vertex partition $V = \{V_1, V_2\}$ where every edge connects a vertex in V_1 to a vertex in V_2
  - G is a bipartite graph iff it has no cycles with odd length (hence trees are bipartite graphs)



Kinds of Graphs

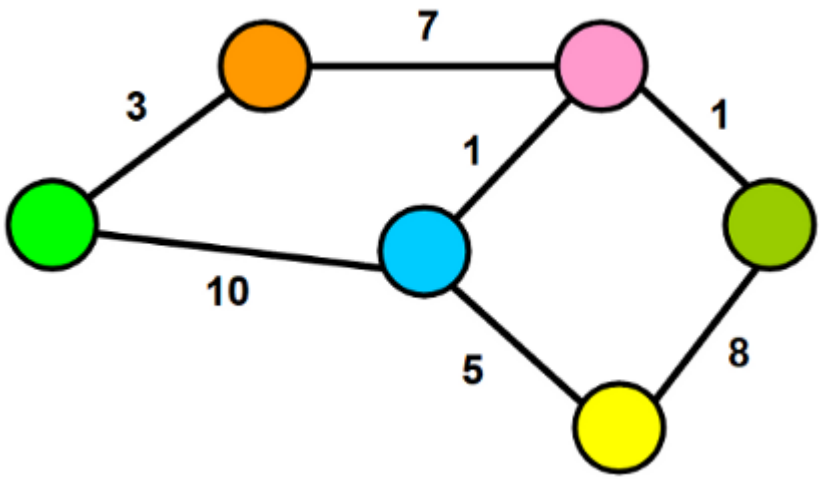
- **Planar Graph**: a graph that can be embedded on a plane without any edges intersecting each other
- **Application**: integrated circuits

Wikipedia article on planar graphs



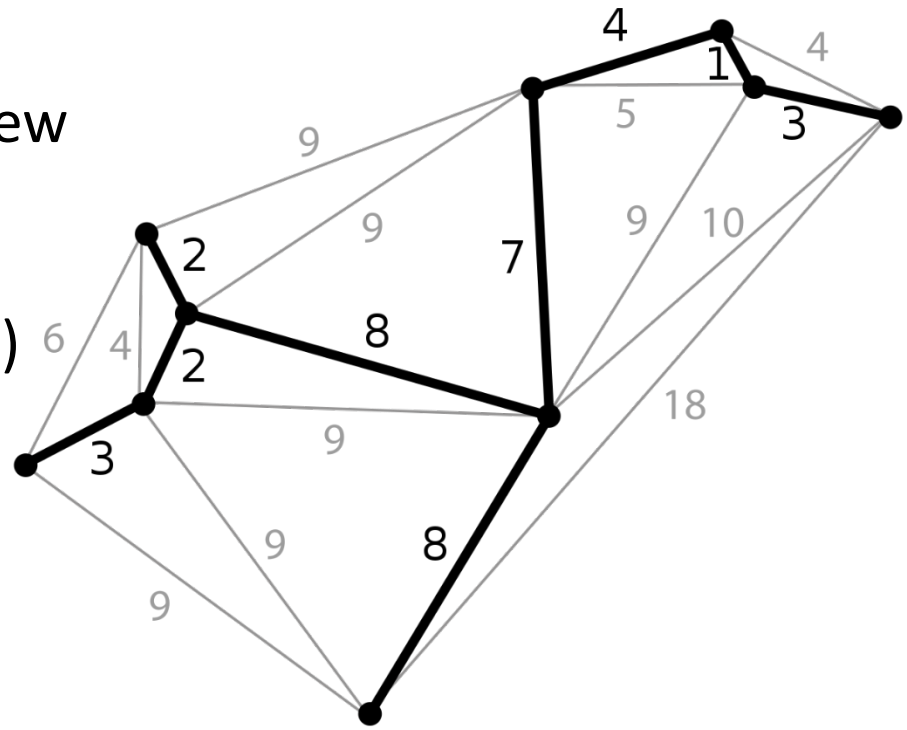
Weighted Graphs

- Each edge has a weight associated with it
- **One approach**: weights are associated with cost or distance
  - Graph structure (i.e. presence or absence of an edge) indicates viable paths
  - Weights indicate distance or cost
  - E.g. route planning, travel planning search engines, etc.
  - Shortest path algorithms take edge weights into account
  - What is the shortest path from light green to pink? From yellow to pink?



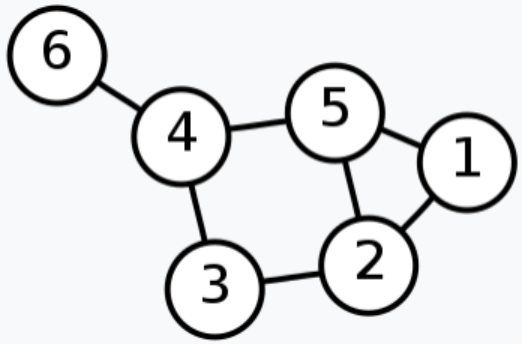
Minimal Spanning Tree

- Consider a connected, weighted graph
- Find a subset of the edges that connects all vertices without any cycles and with the minimum total possible weight
  - **Application**: laying cable in a new neighborhood
  - Graph structure represents feasible paths (e.g. along roads)
  - Weights represent costs (e.g. distance, depth required, etc.)
- Many algorithms exist

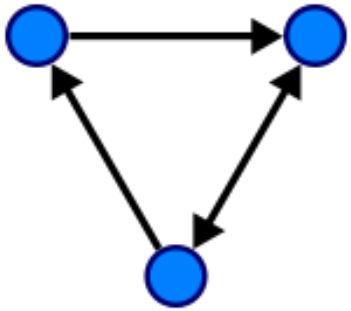


Adjacency and degree matrices

- Example

Labelled graph	Degree matrix	Adjacency matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

- For weighted graphs, replace the 1's with edge weight
- Edges can be directed (one direction) or undirected (bidirectional)



Similarity Graphs

- Unlabeled data to be clustered: $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$
- Assume relationships between data points are captured entirely by a *similarity graph*
- Similarity graphs are defined by an *affinity matrix*

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix}$$

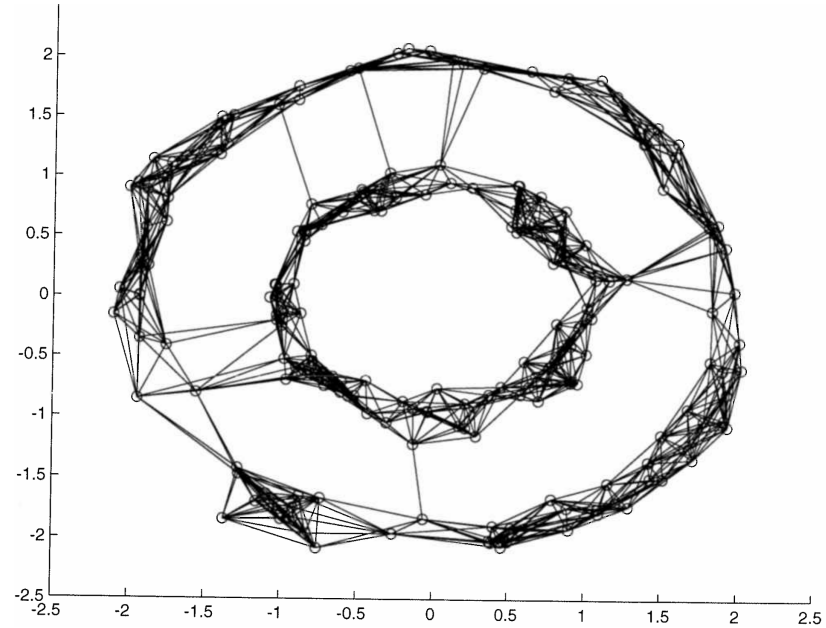
- Weights are nonnegative, symmetric:

$$w_{ij} \geq 0, \qquad w_{ij} = w_{ji}$$

- Associated graph: Nodes i and j are connected iff $w_{ij} > 0$
  - w_{ij} are the edge weights

Similarity Graphs

- Similarity graphs are defined by two things
- **Graph structure** (which edges are connected)
  - Examples
    - k-nn graph
    - ϵ −ball graph
    - Complete graph



11-nn graph

- **Weights**
  - Examples

$$w_{ij} = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right)$$

$$w_{ij} = \begin{cases} 1 & \text{if } \exists \text{ an edge between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Similarity Graphs

- Applications of k-nn graphs
  - Data compression, motion planning (e.g. of a robot), facilities location, manifold learning, density estimation
- Applications of ϵ-ball graphs
  - Similar to k-nn graphs including density estimation, spectral clustering, etc.

Graph Laplacian

- The (weighted) *degree* of a node $\boldsymbol{x}_i$ is

$$d_i := \sum_{j=1}^{n} w_{ij}$$

- The *degree matrix* is the diagonal matrix

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}$$

- The unnormalized graph Laplacian is $L = D - W$
- Note: L is independent of the self-similarity weights since

$$L_{ii} = d_i - w_{ii} = \sum_{j \neq i} w_{ij}$$

Properties of the Graph Laplacian

1. For every $\boldsymbol{f} \in \mathbb{R}^n$

$$\boldsymbol{f}^T L \boldsymbol{f} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

2. L is symmetric and PSD

3. The smallest eigenvalue of L is 0, and

$$\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^n$$

   is a corresponding eigenvector.

4. The number of connected components of the similarity graph is equal to the dimension of the 0-eigenspace (i.e., the subspace of eigenvectors for the eigenvalue zero, which is also the nullspace of L)

Proofs

1.

$$\boldsymbol{f}^T L \boldsymbol{f} = \boldsymbol{f}^T D \boldsymbol{f} - \boldsymbol{f}^T W \boldsymbol{f} = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j} w_{ij} f_i f_j$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j} w_{ij} f_i f_j + \sum_{j=1}^{n} d_j f_j^2\right)$$

$$= \frac{1}{2}\sum_{i,j} w_{ij}(f_i^2 - 2f_i f_j + f_j^2) = \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2$$

2. Follows from the first property and symmetry of W.

3.

$$L \cdot \mathbf{1} = D \cdot \mathbf{1} - W \cdot \mathbf{1} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} - \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} = \mathbf{0} = 0 \cdot \mathbf{1}$$

Normalized Graph Laplacian

- The *normalized graph Laplacian* is

$$\tilde{L} := D^{-1}L.$$

- $\tilde{L}$ is symmetric and PSD
- The smallest eigenvalue of $\tilde{L}$ is 0, and

$$\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^n$$

  is a corresponding eigenvector.

- The number of connected components of the similarity graph is equal to the dimension of the 0-eigenspace (i.e., the subspace of eigenvectors for the eigenvalue zero, which is also the nullspace of $\tilde{L}$)

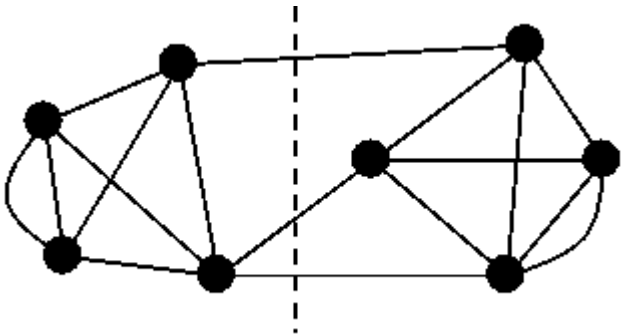
Big Picture

- Spectral clustering determines clusters from the eigenvalue (i.e., spectral) decomposition of L or $\tilde{L}$.
- There are many ways to derive spectral clustering.
- We will focus on one particular way based on graph cuts.

Graph Cuts

- Assume there are two clusters in the data
- A graph cut is a partition of the graph into disjoint sets A and B
  - $A, B \subseteq \{1, \dots, n\}$
- Idea: determine clusters by finding a good graph cut
- In particular, find a cut such that:
  - w_{ij} is large if $\boldsymbol{x}_i, \boldsymbol{x}_j$ are in the <u>same</u> cluster
  - w_{ij} is small if $\boldsymbol{x}_i, \boldsymbol{x}_j$ are in <u>different</u> clusters
- One way to quantify this:

$$\min_{A} C\left(A, \bar{A}\right)$$



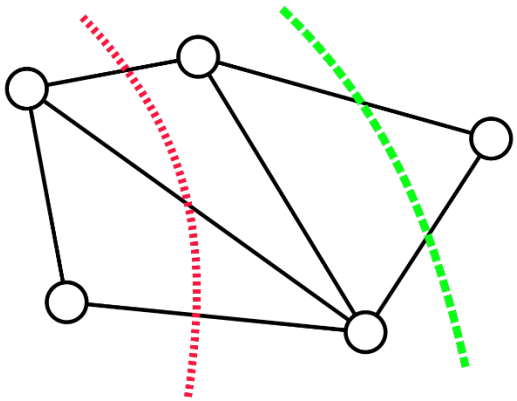
  - Where $C(A, B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$
  - $\bar{A} =$ the complement of A

The Mincut Problem

- Now assume K clusters
- Given a similarity graph/affinity matrix, we want to find a partition $A_1, \dots, A_K$ of $\{1,2,\dots,n\}$ such that:
  - w_{ij} is large if $\boldsymbol{x}_i, \boldsymbol{x}_j$ are in the <u>same</u> cluster
  - w_{ij} is small if $\boldsymbol{x}_i, \boldsymbol{x}_j$ are in <u>different</u> clusters
- The *mincut* problem aims to minimize:

$$cut(A_1, \dots, A_K) = \frac{1}{2}\sum_{k=1}^{K} C\left(A_k, \bar{A}_k\right)$$

with respect to the partition $A_1, \dots, A_K$



Other Graph Cut Problems

- Mincut unfortunately leads to small and often singleton clusters. Therefore some modifications have been proposed:

- *RatioCut* (Hagen and Kahng, 1992)

$$\text{RatioCut}(A_1, \ldots, A_K) = \frac{1}{2} \sum_{k=1}^{K} \frac{C(A_k, \overline{A_k})}{|A_k|}$$

where $|A| = \#$ of nodes in A.

- *Normalized Cut (Ncut)* (Shi and Malik, 2000)

$$\text{Ncut}(A_1, \ldots, A_K) = \frac{1}{2} \sum_{k=1}^{K} \frac{C(A_k, \overline{A_k})}{\text{vol}(A_k)}$$

where $\text{vol}(A) = \sum_{i \epsilon A} \sum_{j \epsilon V} w_{ij}$ and V is the set of all nodes (vertices) in the graph.

Relaxations

- Unfortunately, introducing these "balancing" terms causes these problems to be NP-hard.
- Therefore we will consider *relaxed* versions of these optimization problems
- A *relaxation* of a constrained optimization problem is obtained by enlarging the feasible set so as to make the problem (more) tractable.
- We will see that relaxations of RatioCut and NCut can be solved in terms of the spectral decomposition of L and $\tilde{L}$, respectively.

Relaxation of RatioCut

- Assume $K = 2$ (the argument generalizes but this case is simpler)
- We wish to minimize

$$\mathrm{RatioCut}(A, \bar{A}) = \frac{1}{2}\left[\frac{C(A, \bar{A})}{|A|} + \frac{C(A, \bar{A})}{|\bar{A}|}\right]$$

- Given $A \subseteq \{1,2, \dots, n\}$, define $\boldsymbol{f}_A = \left(f_{A_1}, \dots, f_{A_n}\right)^T \in \mathbb{R}^n$ by

$$f_{A_i} = \begin{cases} +\sqrt{|\bar{A}|/|A|} & : i \in A \\ -\sqrt{|A|/|\bar{A}|} & : i \notin A \end{cases}$$

- Example: $n = 5, A = \{3,4\} \Rightarrow$

$$f_A = \begin{bmatrix} -\sqrt{\frac{2}{3}} & -\sqrt{\frac{2}{3}} & \sqrt{\frac{3}{2}} & \sqrt{\frac{3}{2}} & -\sqrt{\frac{2}{3}} \end{bmatrix}^T$$

- Claim: For all $A \subseteq \{1,2, \dots, n\}$,

$$\boldsymbol{f}_A^T L \boldsymbol{f}_A = n \cdot \mathrm{RatioCut}(A, \bar{A})$$

Relaxation of RatioCut

- *Proof of Claim*:

$$
\begin{aligned}
\boldsymbol{f}_A^T L \boldsymbol{f}_A &= \frac{1}{2}\sum_{i,j} w_{ij}\left(f_{A_i} - f_{A_j}\right)^2 \\
&= \frac{1}{2}\sum_{i\in A, j\in \overline{A}} w_{ij}\left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}}\right)^2 + \sum_{i\in \overline{A}, j\in A} w_{ij}\left(-\sqrt{\frac{|A|}{|\overline{A}|}} - \sqrt{\frac{|\overline{A}|}{|A|}}\right)^2 \\
&= \frac{1}{2}C(A,\overline{A})\left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2\right) + \frac{1}{2}C(A,\overline{A})\left(\frac{|A|}{|\overline{A}|} + \frac{|\overline{A}|}{|A|} + 2\right) \\
&= C(A,\overline{A})\left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2\right) \\
&= C(A,\overline{A})\left(\frac{|A| + |\overline{A}|}{|\overline{A}|} + \frac{|\overline{A}| + |A|}{|A|}\right) \\
&= n\left(\frac{C(A,\overline{A})}{|\overline{A}|} + \frac{C(A,\overline{A})}{|A|}\right) = n \cdot \mathrm{RatioCut}(A,\overline{A})
\end{aligned}
$$

Relaxation of RatioCut

- Furthermore, $\boldsymbol{f}_A$ satisfies the following two properties (group exercise):

  1. $\mathbf{1}^T \boldsymbol{f}_A = 0$
  2. $\|\boldsymbol{f}_A\|^2 = n$

- Therefore, RatioCut can be written as the following optimization problem:

$$\begin{aligned} \min_{A \subset \{1,\ldots,n\}} \quad & \boldsymbol{f}_A^T L \boldsymbol{f}_A \\ s.t. \quad & \mathbf{1}^T \boldsymbol{f}_A = 0 \\ & \|\boldsymbol{f}_A\| = \sqrt{n} \end{aligned}$$

- Note that it would still be RatioCut without the above two constraints, but we include them to keep the relaxation close to the original problem.

Relaxation of RatioCut

- We can now state the relaxation
- We have written RatioCut as

$$\min_{A \subset \{1,\ldots,n\}} \quad \boldsymbol{f}_A^T L \boldsymbol{f}_A$$

$$s.t. \quad \mathbf{1}^T \boldsymbol{f}_A = 0$$

$$\|\boldsymbol{f}_A\| = \sqrt{n}$$

- A relaxation of RatioCut is

$$\min_{\boldsymbol{f} \in \mathbb{R}^n} \quad \boldsymbol{f}^T L \boldsymbol{f}$$

$$s.t. \quad \mathbf{1}^T \boldsymbol{f} = 0$$

$$\|\boldsymbol{f}\| = \sqrt{n}$$

Group Exercise

Recall: Given $A \subseteq \{1, 2, \ldots, n\}$, define $\boldsymbol{f}_A = (f_{A_1}, \ldots, f_{A_n})^T \in \mathbb{R}^n$ by

$$f_{A_i} = \begin{cases} +\sqrt{|\overline{A}| \,/\, |A|} & : i \in A \\ -\sqrt{|A| \,/\, |\overline{A}|} & : i \notin A \end{cases}$$

1. Verify that $\mathbf{1}^T \boldsymbol{f}_A = 0$ for all $A \subseteq \{1, \ldots, n\}$
2. Verify that $\|\boldsymbol{f}_A\|^2 = n$ for all $A \subseteq \{1, \ldots, n\}$
3. Determine the solution of

$$\begin{aligned} \min_{\boldsymbol{f} \in \mathbb{R}^n} \; & \boldsymbol{f}^T L \boldsymbol{f} \\ \text{s.}t. \; & \mathbf{1}^T \boldsymbol{f} = 0 \\ & \|\boldsymbol{f}\| = \sqrt{n} \end{aligned}$$

   where $L = D - W$ is the unnormalized graph Laplacian.

4. How can you recover a solution to the RatioCut problem from the solution the relaxed problem?

Remarks

- In practice the similarity graph is chosen to have only one connected component (if there is more then one connected component, just cluster each one separately). So the 0-eigenspace has dimension 1, and there is no ambiguity in the choice of $\mathbf{1}$ as the smallest eigenvector.
- A similar analysis applies to $K > 2$ and Ncut.
- The gap between the optimal value of RatioCut and the optimal value of its relaxation can be arbitrarily large.

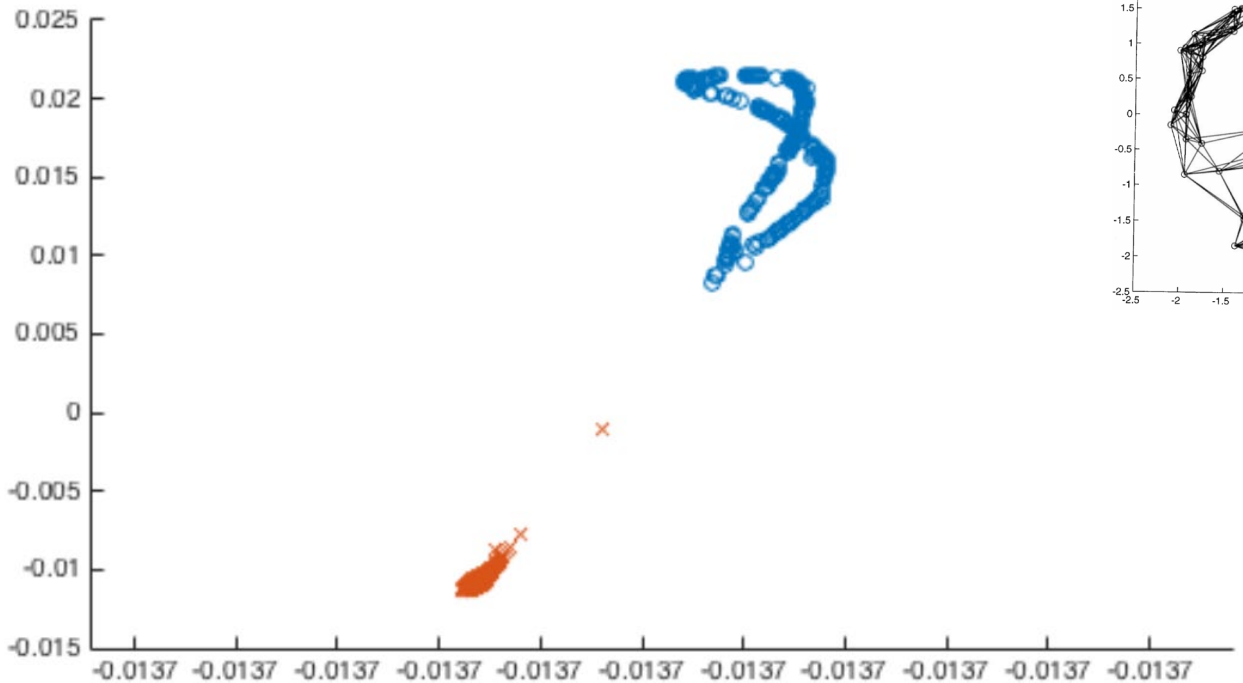
Spectral Clustering

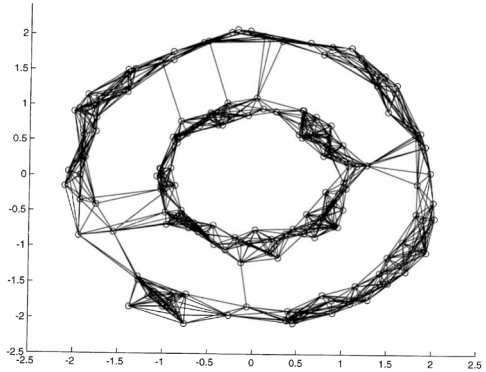
Input: $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, desired number of clusters K, parameters of similarity graph

1. Construct a similarity graph and form the graph Laplacian L (or $\tilde{L}$)
2. Determine the K smallest eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_K$ of L (or $\tilde{L}$) and corresponding eigenvectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_K \in \mathbb{R}^n$
3. Set $\boldsymbol{y}_i = \begin{bmatrix} u_1^{(i)} & u_2^{(i)} & \dots & u_K^{(i)} \end{bmatrix}^T, i \in \{1,2,\dots,n\}$
   - I.e. $\boldsymbol{y}_i$ is the ith row in the matrix $\begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_K \end{bmatrix}$
4. Cluster $\{\boldsymbol{y}_i\}_{i=1}^n$ using K-means clustering, and assign $\{\boldsymbol{x}_i\}_{i=1}^n$ to the corresponding clusters

Example

- This figure shows $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ for the circular clusters dataset shown earlier.
- The mapping $\boldsymbol{x} \mapsto \boldsymbol{y}$ is actually a form of nonlinear dimensionality reduction (called Laplacian eigenmaps) that transforms the data to a space where k-means can be successfully applied.





Final Remarks

- Model selection: Choose K such that λ_{K+1} is the first "large" eigenvalue.
- In practice $\tilde{L}$ is preferred to L.
- There is yet a third graph Laplacian defined as

$$\tilde{\tilde{L}} = D^{-\frac{1}{2}} L D^{\frac{1}{2}}$$

  It has properties similar to L and $\tilde{L}$, but the spectral clustering algorithm needs to be tweaked.
- For a very thorough tutorial on spectral clustering see U. von Luxburg, "A Tutorial on Spectral Clustering", 2007.

Comparing clusterings

- Given two clusterings or partitions of the data, how do you measure their correspondence and discrepancy?
- One way: the *Rand index*
- Given data set $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ and two partitions of X
  - $S = \{S_1, \dots, S_s\}$ is a partition of X into s subsets
  - $R = \{R_1, \dots, R_r\}$ is a partition of X into r subsets
- Define the following:
  - a = # of pairs of elements in X that are in the same subset in S and the same subset in R
  - b = # of pairs of elements in X that are in different subsets in S and different subsets in R
  - c = # of pairs of elements in X that are in the same subset in S and different subsets in R
  - d = # of pairs of elements in X that are in different subsets in S and the same subset in R

The Rand Index

- Define the following:
  - a = # of pairs of elements in X that are in the same subset in S and the same subset in R
  - b = # of pairs of elements in X that are in different subsets in S and different subsets in R
  - c = # of pairs of elements in X that are in the same subset in S and different subsets in R
  - d = # of pairs of elements in X that are in different subsets in S and the same subset in R
- The Rand index:

$$RI = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

- Intuition: $a+b$ is the number of agreements between R and S while $c+d$ is the number of disagreements
- The Rand index is the probability that R and S will agree on a randomly chosen pair

The Adjusted Rand Index

- The *adjusted Rand index* is the corrected-for-chance version of the Rand index
  - Uses a permutation model
- Can summarize the overlap between S and R with a contingency table where $n_{ij} = |S_i \cap R_j|$

S/R	R_1	R_2	$\cdots$	R_r	Sums
S_1	n_{11}	n_{12}	$\cdots$	n_{1r}	a_1
S_2	n_{21}	n_{22}	$\cdots$	n_{2r}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
S_s	n_{s1}	n_{s2}	$\cdots$	n_{sr}	a_s
Sums	b_1	b_2	$\cdots$	b_r	

$$\overbrace{ARI}^{\text{Adjusted Index}} = \frac{\overbrace{\sum_{ij}\binom{n_{ij}}{2}}^{\text{Index}} - \overbrace{\left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}^{\text{Expected Index}}}{\underbrace{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right]}_{\text{Max Index}} - \underbrace{\left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}_{\text{Expected Index}}}$$

The Adjusted Rand Index

- Rand index is between 0 and 1
  - Perfect correspondence is 1
- ARI is between -1 and 1
  - Negative values indicate the index is less than the expected index
  - Perfect correspondence is 1

Further Reading

- "A Tutorial on Spectral Clustering," 2007
- https://en.wikipedia.org/wiki/Rand_index
- ESL Section 14.5.3