

Machine Learning for Graphs and Sequential Data

Graphs – Clustering

Lecturer: Prof. Dr. Stephan Günnemann

www.daml.in.tum.de

Summer Term 2020

Data Analytics and
Machine Learning 

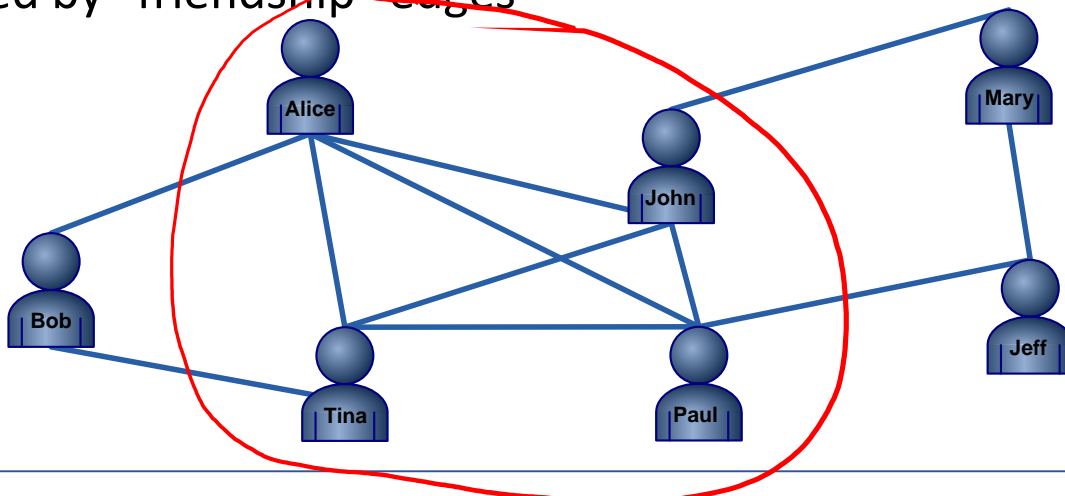
Roadmap

- **Chapter: Graphs**

1. Graphs & Networks
2. Generative Models
- 3. Clustering**
 - Introduction
 - Cuts & Spectral Clustering
 - Probabilistic Approaches
4. Node Embeddings
5. Ranking
6. Semi-Supervised Learning
7. Limitations of GNNs

Clustering in Network Data - Introduction

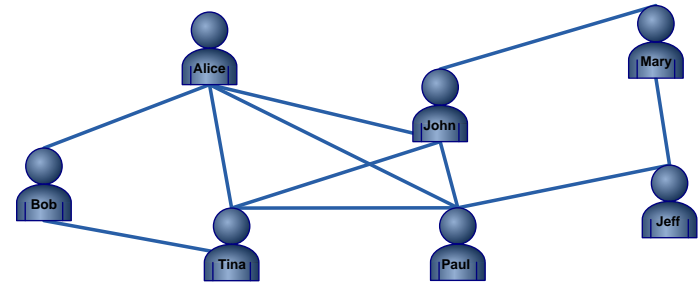
- Input is a graph $G = (V, E)$
- Aim: Find clusters of *vertices* in the graph
- Related to "traditional" clustering (e.g. *k*-Means):
 - In traditional clustering, we cluster objects based on *attribute data*
 - Here, we cluster objects based on *graph data* (information about relationships between the objects)
- Example: Given a social network, find groups of people that are densely connected by "friendship" edges



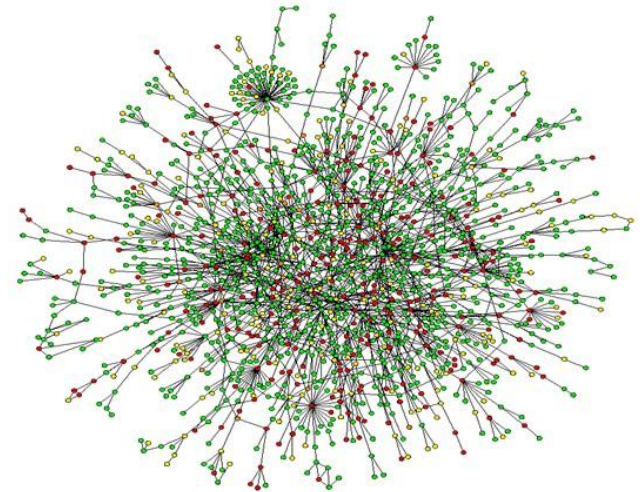
Clustering in Network Data - Introduction

Many different applications:

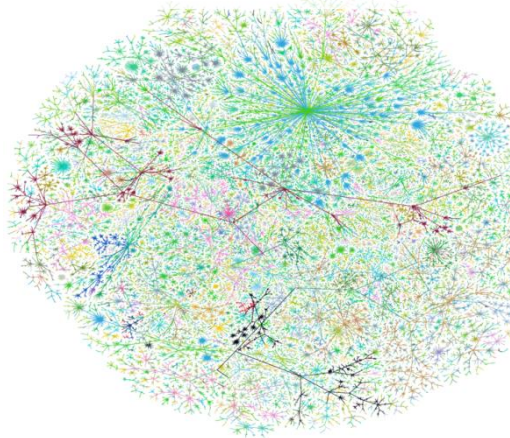
- Friendship graph: find circles of friends
- Protein-/Gene-Interaction Network: find groups of highly interacting proteins/genes
- Internet graph: Find groups of websites with similar topics



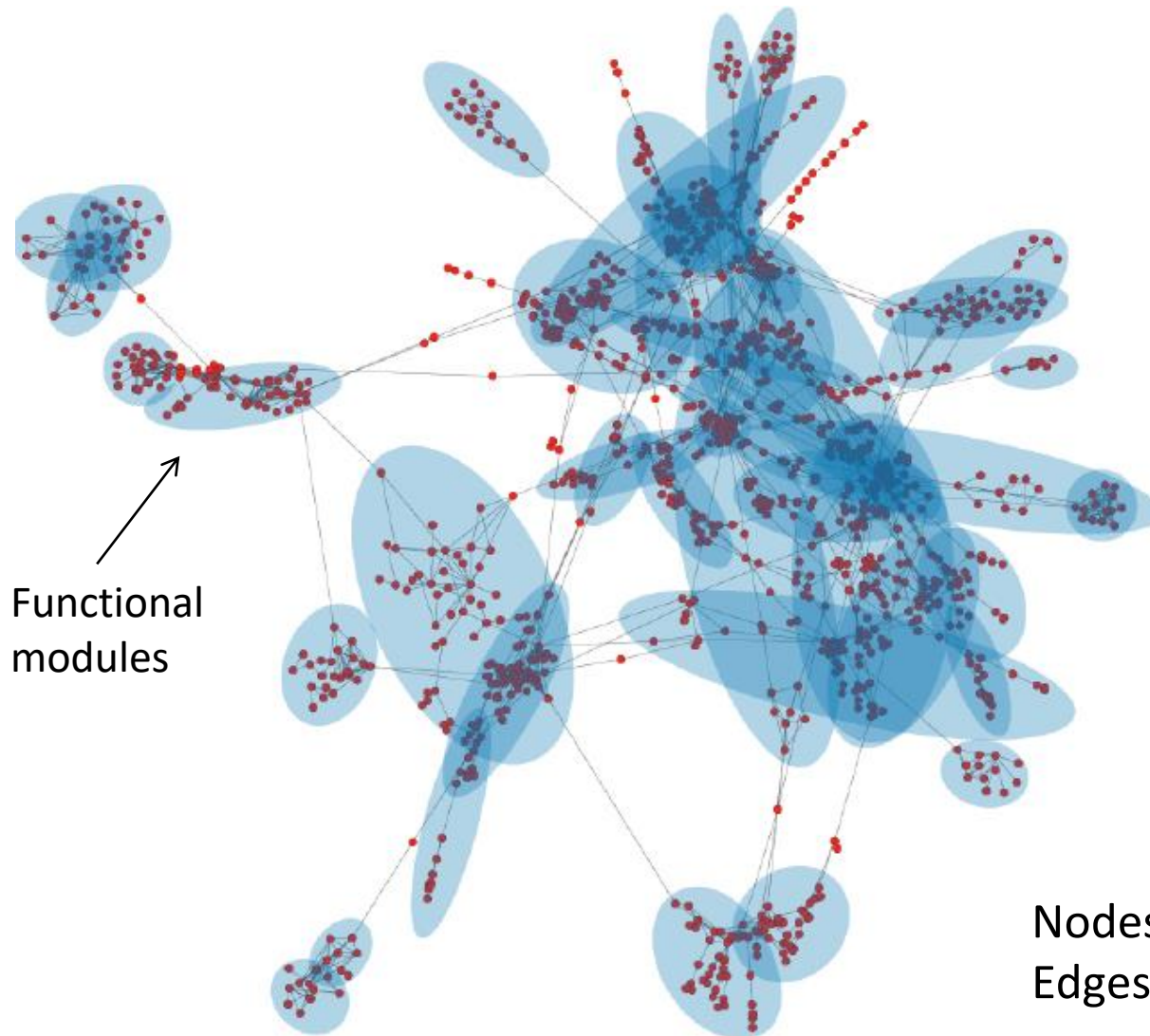
Yeast protein interaction network



Internet snapshot

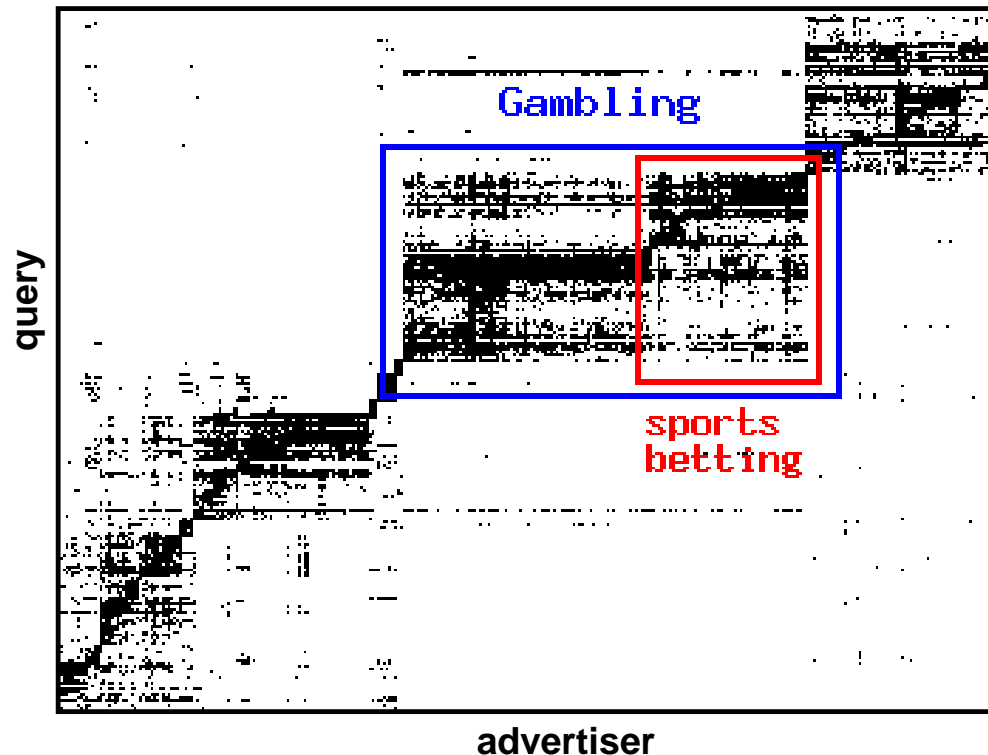


Example: Protein-Protein Interactions



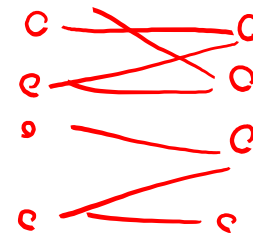
Example: Micro-Markets in Sponsored Search

- Find micro-markets by partitioning the query-to-advertiser graph:

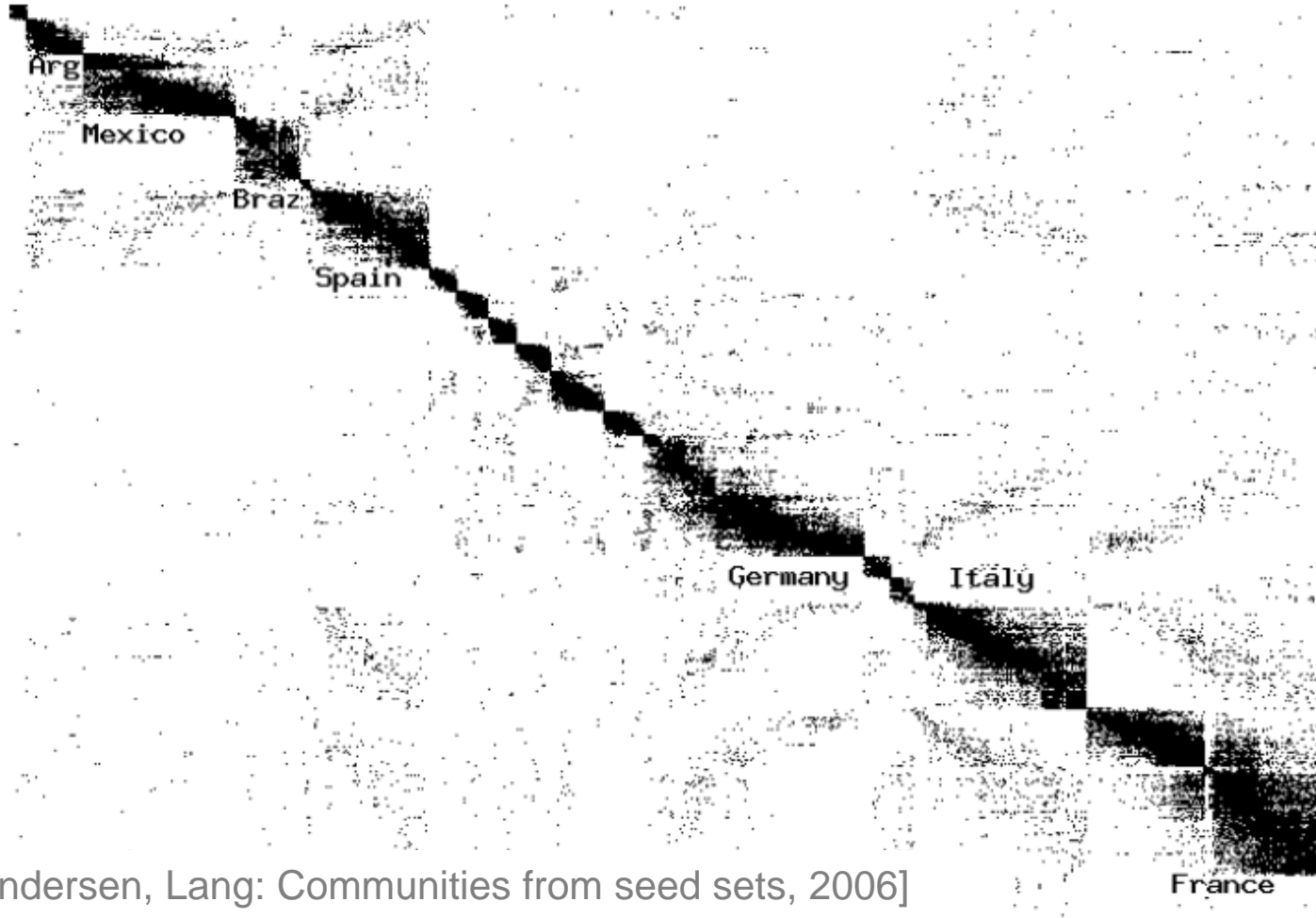


[Andersen, Lang: Communities from seed sets, 2006]

Example: Movies and Actors



- Clusters in Movies-to-Actors graph:



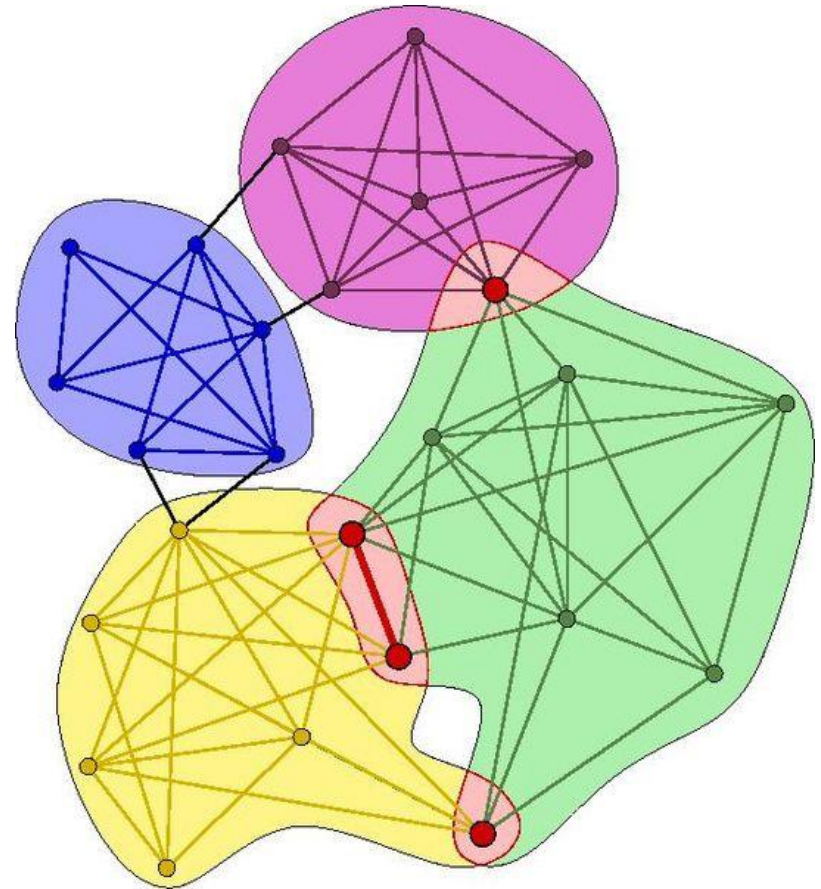
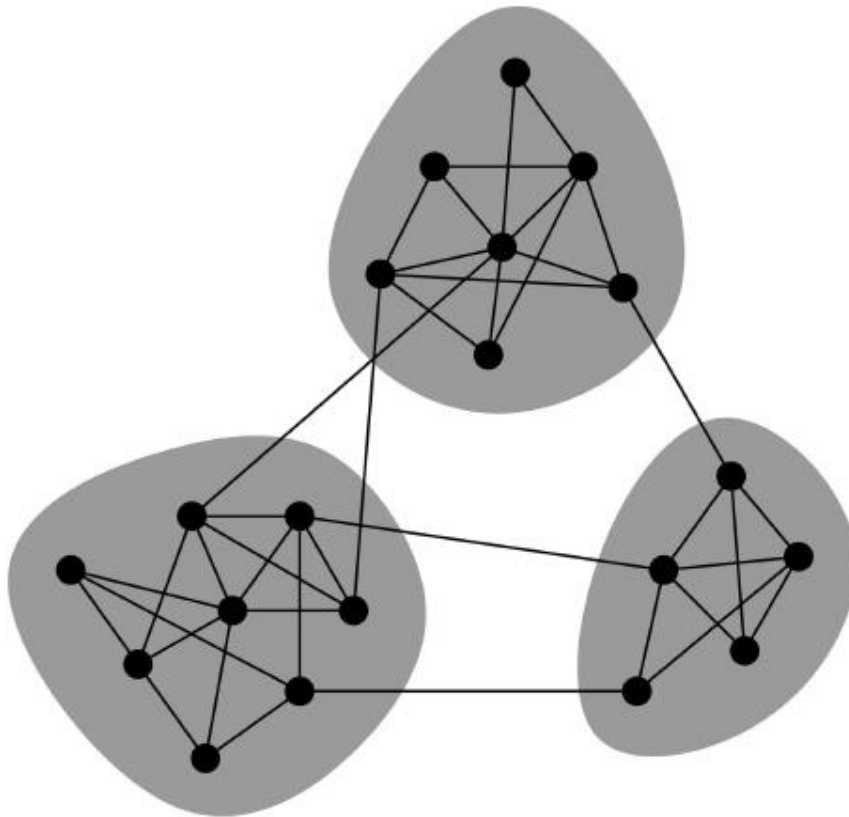
[Andersen, Lang: Communities from seed sets, 2006]

Clustering in Network Data - Introduction

- Basic aims for a good clustering:
 - Vertices in the **same cluster** should be connected by **many edges** (*intra-cluster edges*)
 - Only **few edges** between **different clusters** (*inter-cluster edges*)
- Two main categories of clustering algorithms:
 - **Partitioning approaches**: Each vertex is assigned to *exactly one* cluster
 - **Non-partitioning approaches**: Clusters can overlap, "outliers" that belong to no cluster are possible

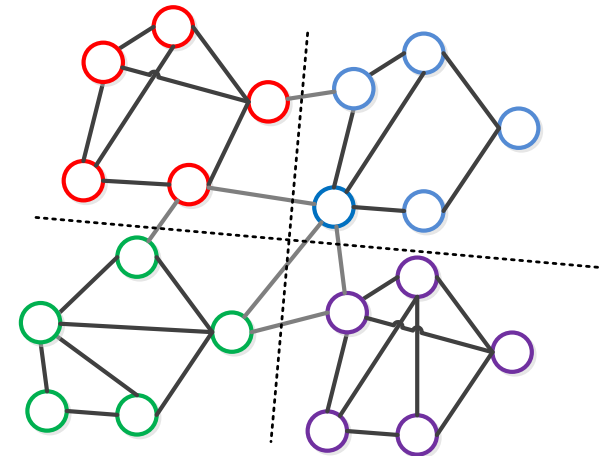
General Categorization

- Non-overlapping vs. overlapping clustering



Partitioning Approaches

- Basic idea: (Constrained) Optimization Problem
- Given a graph $G = (V, E)$, partition vertex set V into a set of clusters $\mathcal{C} = \{C_1, \dots, C_k\}$ such that
 - A given objective function $Q(\mathcal{C})$ is optimized
 - Subject to the constraints
 - $C_1 \cup \dots \cup C_k = V$ (Every vertex from V belongs to a cluster)
 - $\forall 1 \leq i < j \leq k: C_i \cap C_j = \emptyset$ (Clusters are disjoint; no overlap)
- Usually NP-hard problem since discrete optimization
 - for $k=2$ sometimes polynomial time algorithms exist
- Cf. k-Means: Objects are partitioned such that Total Distance is minimized



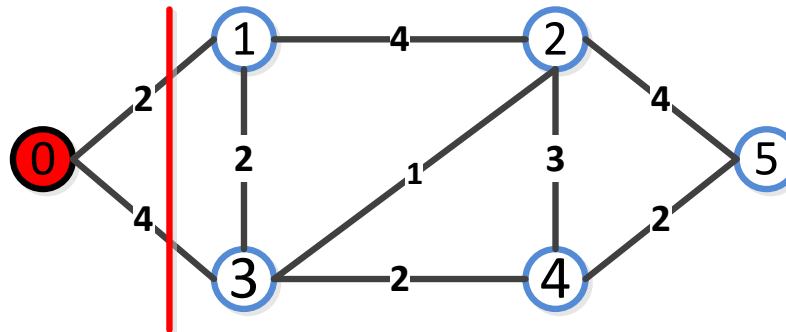
Roadmap

- **Chapter: Graphs**

1. Graphs & Networks
2. Generative Models
- 3. Clustering**
 - Introduction
 - **Cuts & Spectral Clustering**
 - Probabilistic Approaches
4. Node Embeddings
5. Ranking
6. Semi-Supervised Learning
7. Limitations of GNNs

Global Minimum Cut

- First idea: minimize the number of edges/weights between the clusters
 - small value of the cut
- Task: Partition V into two sets C_1 and C_2 , such that the sum of the inter-cluster edge weights $\text{cut}(C_1, C_2) = \sum_{v_1 \in C_1, v_2 \in C_2} w(v_1, v_2)$ is minimized
 - here: undirected edges, i.e. $w(a, b) = w(b, a)$



- Note: Computing an s-t-cut (i.e. where a source s and sink t are given) can be done in polynomial time via the algorithm of Ford and Fulkerson
 - Graph Clustering: no source/sink is given; any partitioning is possible, "global" minimum cut

Normalized Cut Criteria

Ratio: $\frac{6}{7} + \frac{6}{5}$
 $\frac{7}{3} + \frac{7}{3}$

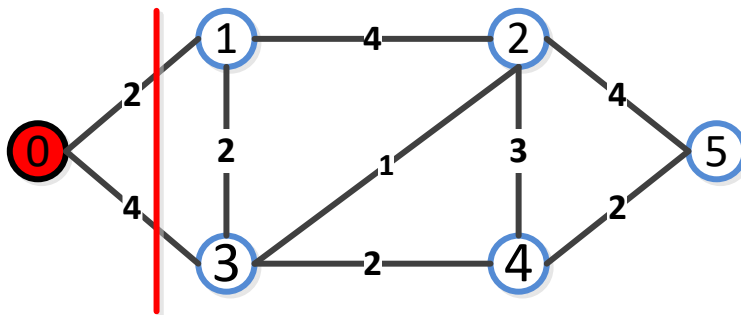
- Drawbacks of Minimum Cut:
 - Tends to cut small vertex sets from the rest of the graph
 - Considers only inter-cluster edges, no intra-cluster edges

volume of a set of nodes

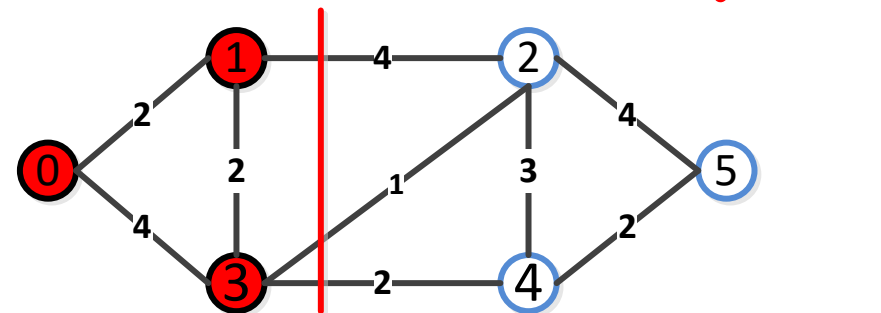
$$\begin{aligned} \text{vol}(C_i) &= \\ \text{assoc}(C_i, V) &= \\ \text{cut}(C_i, V) &= \\ \sum_{v_i \in C_i, v_j \in V} w(v_i, v_j) &= \\ \sum_{v_i \in C_i} \text{deg}(v_i) &= \end{aligned}$$

- Therefore, normalized cut criteria were introduced:

- Ratio Cut: Minimize $\frac{\text{cut}(C_1, C_2)}{|C_1|} + \frac{\text{cut}(C_2, C_1)}{|C_2|}$
- Normalized Cut: Minimize $\frac{\text{cut}(C_1, C_2)}{\text{vol}(C_1)} + \frac{\text{cut}(C_1, C_2)}{\text{vol}(C_2)}$



Minimum Cut



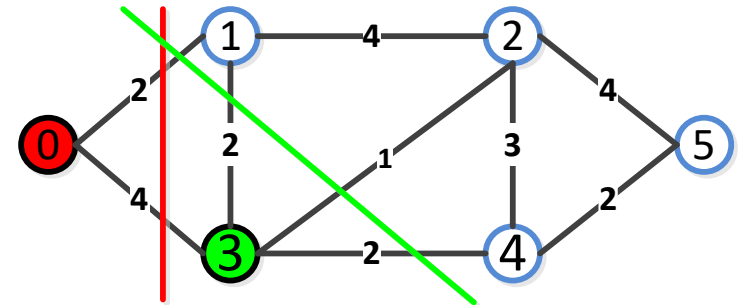
Normalized Cut

Multi-way Graph Partitioning

- Generalization to $k \geq 2$ clusters
- Partition V into disjoint clusters C_1, \dots, C_k such that

- Cut: $\min_{C_1, \dots, C_k} \sum_{i=1}^k \text{cut}(C_i, V \setminus C_i)$
- Ratio Cut: $\min_{C_1, \dots, C_k} \sum_{i=1}^k \frac{\text{cut}(C_i, V \setminus C_i)}{|C_i|}$
- Normalized Cut: $\min_{C_1, \dots, C_k} \sum_{i=1}^k \frac{\text{cut}(C_i, V \setminus C_i)}{\text{vol}(C_i)}$

$$6 + 9 + 7$$

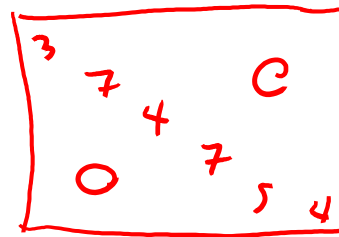


Minimum Cut for $k = 3$

- Finding the optimal solution is NP-hard
- How to compute an approximate solution efficiently?

Graph Laplacian

$$* = \sum_n \deg(n) f_n^2 - \sum_{(n,v)} W_{nv} f_n f_v$$



$$\begin{aligned} \forall n \sum_{(n,v) \in E} W_{nv} \\ = \deg(n) \end{aligned}$$

- Definition: **Laplacian matrix** $L = D - W$
 - W = (weighted) adjacency matrix, D = degree matrix

$$\text{i.e. } L_{uv} = \begin{cases} -W_{uv}, & \text{if } (u,v) \in E \\ \deg(v), & \text{if } u = v \\ 0, & \text{otherwise} \end{cases}$$

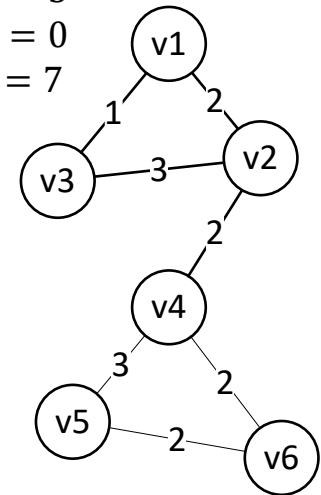
- Observation 1:** For any vector f we have

$$f^T \cdot L \cdot f = \frac{1}{2} \cdot \sum_{(u,v) \in E} W_{uv} (f_u - f_v)^2$$

$$= \frac{1}{2} \left(\sum_{(n,v)} W_{nv} f_n^2 - \underbrace{2 W_{nv} f_n f_v} + \underbrace{W_{nv} f_v^2} \right)$$

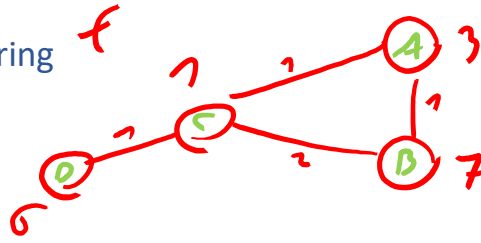
$$= \frac{1}{2} \left(\sum_n \deg(n) f_n^2 + \underbrace{\sum_v \deg(v) f_v^2} - \sum_{(n,v) \in E} 2 W_{nv} f_n f_v \right)$$

$$\begin{aligned} W[2,3] &= 3 \\ W[2,5] &= 0 \\ D[2,2] &= 7 \end{aligned}$$



Graph Laplacian

$$a = \begin{bmatrix} 3 \\ 7 \\ 1 \\ 6 \end{bmatrix}$$

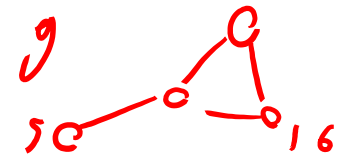


$$1 \cdot 4 + 2 \cdot 6 = 16$$

$$g(B) = 16$$

- The Laplacian is a discrete analogue of the Laplacian $\sum_i \frac{\partial^2 f}{\partial x_i^2}$, it measures to what extent a function differs at a point from its values at nearby points
 - Laplace operator often denoted as Δ
- The (discrete) Laplacian is an operator on the n-dimensional vector space of functions $f : V \rightarrow \mathbb{R}$
 - Think about the function f as assigning a "number" to each node
- The Laplacian "transforms" f to another function g , i.e. $\Delta f = g$
 - $g(v) = (\Delta f)(v) = \sum_{(u,v) \in E} W_{uv} \cdot [f(v) - f(u)]$
 - For finite-dimensional graphs, you can simply represent f and g as vectors, e.g. a and $b \Rightarrow b = L \cdot a$

$$b = \begin{bmatrix} 16 \\ 5 \end{bmatrix}$$



Properties of the Graph Laplacian (I)

- L is symmetric and positive semi-definite w.r.t. 0

 - Symmetric since D, and W are symmetric (for undirected graphs)
 - From Observation 1 we get PSD: $x^T L x \geq 0$, for any x , since it's a quadratic

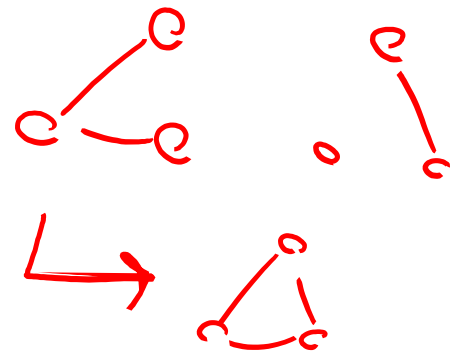
- The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\vec{1}$, (or any $c \cdot \vec{1}$, for any scalar c) $x^T x = 1$

 - Recall $L x = \lambda x \Rightarrow x^T L x = \lambda x^T x \Rightarrow \frac{1}{2} \sum_{(u,v) \in E} W_{uv} (x_u - x_v)^2 = \lambda$ $= x^T \lambda x$
 - Trivial solutions set $x = c \cdot \vec{1}$ and $\lambda = 0$

- L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$

 - In general, any symmetric matrix has real-valued eigenvalues
 - From above $\lambda_1 = 0$ is the smallest eigenvalues \Rightarrow the rest must be larger

Properties of the Graph Laplacian (II)



- Laplacian is additive: $L_{G \cup H} = L_G + L_H$
 - For two graphs G and H on the same vertex set with disjoint edge sets

- Laplacian of an edge, i.e. graph with a single edge (u, v) :

$$L_{(u,v)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- Laplacian of a graph is a sum of Laplacians for each edge:
 - $L_G = \sum_{(u,v) \in E} L(u, v)$
 - Allows us to prove properties for a single edge and add them up

Properties of the Graph Laplacian (III)

- **Observation 2:** The number of eigenvectors of L with eigenvalue 0 corresponds to the number of **connected components**.

0	1	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	0	0	1	0	2	0	0	0
0	0	0	1	2	0	0	0	0
0	0	0	0	0	0	0	3	1
0	0	0	0	0	0	3	0	1
0	0	0	0	0	0	1	1	0

Adjacency matrix W

2	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	3	0	0	0	0	0
0	0	0	0	3	0	0	0	0
0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	2

Degree matrix D

2	-1	-1	0	0	0	0	0	0
-1	2	-1	0	0	0	0	0	0
-1	-1	2	0	0	0	0	0	0
0	0	0	3	-1	-1	0	0	0
0	0	0	-1	3	-2	0	0	0
0	0	0	-1	-2	3	0	0	0
0	0	0	0	0	0	4	-3	-1
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	0	-1	-1	2

Laplacian matrix L

- Let C_k be the set of nodes corresponding to the k -th connected component and Let $f_{C_k}[i] = 1$ if $v_i \in C_k$, else $f_{C_k}[i] = 0$
 - e.g. $f_{C_1} = [1,1,1,0,0,0,0,0,0]$, $f_{C_2} = [0,0,0,1,1,1,0,0,0]$ and $f_{C_3} = [0,0,0,0,0,0,1,1,1]$
 - From Observation 1: $f_{C_k}^T L f_{C_k} = 0, \forall k \Rightarrow f_{C_k}$ are the 'smallest' eigenvectors of L
- Corollary: If the graph is connected $\lambda_2(L) > 0$

$f_{C_2} + f_{C_3}$

Properties of the Graph Laplacian (IV)

- **Algebraic connectivity** of a graph is $\lambda_2(L)$
 - Also known as Fiedler (eigen)value
 - The magnitude reflects how well connected the graph overall is
 - e.g. $\lambda_2(K_n) = n$, where K_n is a complete graph with n nodes

- For every $S \subset V$ we have $\theta(S) = \frac{\text{cut}(S, \bar{S})}{|S|} \geq \lambda_2 \left(1 - \frac{|S|}{|V|}\right)$
 - $\theta(S)$ is called the isoperimetric ratio of S
 - $\theta_G \stackrel{\text{def}}{=} \min_{|S| \leq \frac{|V|}{2}} \theta(S) \geq \lambda_2/2$
 - θ_G is called the Cheeger constant of a graph, the conductance of a graph, etc.

- The inequality implies that if λ_2 is big the graph is very well connected
 - the boundary of each *small* set of vertices is at least λ_2 times a value close to 1

Properties of the Graph Laplacian (V)

- Conclusion: The spectrum of L encodes useful information about the graph
- Unfortunately, there exist co-spectral graphs
 - Graphs that are not isomorphic but share the same spectrum
 - Implies that the spectrum doesn't completely characterize the graph
- Co-spectrality can also be useful
 - Find a sparse version of a graph with (approximately) the same spectrum
 - Yields computational benefits for large graphs

The Graph Laplacian and the Minimum Cut

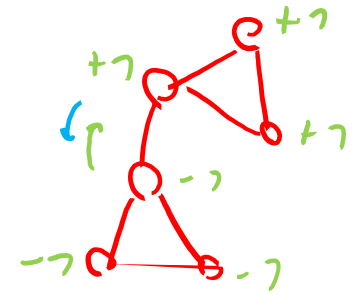
- For $k = 2$ clusters let $f \in \{+1, -1\}^n$ be an indicator vector

$$f_{C_1}[i] = \begin{cases} +1 & \text{if } v_i \in C_1 \\ -1 & \text{else, } v_i \in V \setminus C_1 = C_2 \end{cases}$$

- Then you can verify $f_{C_1}^T L f_{C_1} = 4 \cdot \text{cut}(C_1, C_2)$
- Thus we can minimize $f_{C_1}^T L f_{C_1}$ to minimize the cut

$$\min_f f^T L f$$

- However, we established that minimum cut is not ideal
- Can we specify a different indicator that leads to the ratio cut?



$$\begin{aligned} & (-1 - 1)^2 \\ & + (+1 - (-1))^2 \\ & = (-2)^2 + (+2)^2 \\ & = 4 + 4 = 4 \cdot 2 \end{aligned}$$

Spectral Clustering: 2 Clusters

- Let's focus on minimizing the **ratio cut** for $k=2$ clusters

$$- \min_{C_1 \subset V} \frac{\text{cut}(C_1, C_2)}{|C_1|} + \frac{\text{cut}(C_2, C_1)}{|C_2|} \quad \text{where } C_2 = V \setminus C_1 =: \overline{C_1}$$

- Consider the **indicator vector** f_{C_1} for the cluster C_1 , i.e.

$$f_{C_1}[i] = \begin{cases} +\sqrt{\frac{|C_1|}{|C_2|}} & \text{if } v_i \in C_1 \\ -\sqrt{\frac{|C_1|}{|C_2|}} & \text{else} \end{cases}$$

$$1. \sum_i f_{C_1}[i] = 0$$

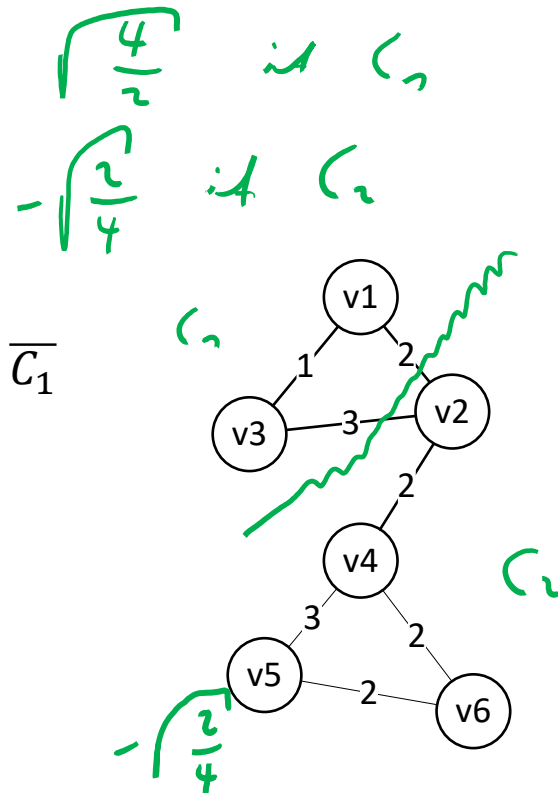
→ f_{C_1} is orthogonal to vector $\vec{1}$: $f_{C_1} \perp \vec{1}$

$$2. f_{C_1}^T \cdot f_{C_1} = \|f_{C_1}\|_2^2 = |V|$$

→ length is constant

$$3. f_{C_1}^T \cdot L \cdot f_{C_1} = \dots = |V| \cdot \left[\frac{\text{cut}(C_1, C_2)}{|C_1|} + \frac{\text{cut}(C_1, C_2)}{|\overline{C_1}|} \right]$$

→ $|V| \cdot \text{ratio cut}$



Spectral Clustering: 2 Clusters

- Minimizing the ratio cut is equivalent to

$$\min_{C_1 \subset V} \{f_{C_1}^T \cdot L \cdot f_{C_1}\} \text{ subject to } f_{C_1} \perp \vec{1} \text{ and } \|f_{C_1}\|_2 = \sqrt{|V|} \text{ and } f_{C_1} \text{ as defined before}$$

- Discrete optimization problem → still NP-hard in general
- Idea: Constraint relaxation**
 - drop the discreteness condition (i.e. f_{C_1} can take any values)

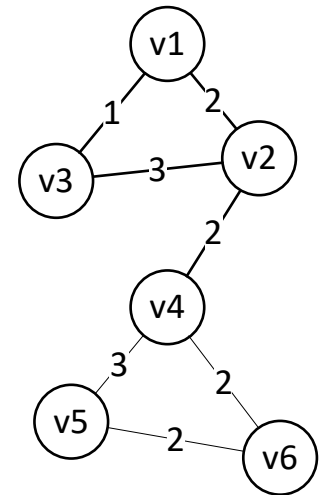
- Result: $\min_{f_{C_1} \in \mathbb{R}^{|V|}} \{f_{C_1}^T \cdot L \cdot f_{C_1}\}$ subject to $f_{C_1} \perp \vec{1}$ and $\|f_{C_1}\|_2 = \sqrt{|V|}$

Spectral Clustering: 2 Clusters

- Result: $\min_{f_{C_1} \in \mathbb{R}^{|V|}} \{f_{C_1}^T \cdot L \cdot f_{C_1}\}$ subject to $f_{C_1} \perp \vec{1}$ and $\|f_{C_1}\|_2 = \sqrt{|V|}$
- What is the solution to this problem?
- We have for any symmetric matrix L :
 - $\lambda_1 = \min_{\|x_1\|=1} x_1^T L x_1 \quad \lambda_2 = \min_{\|x_2\|=1, x_2 \perp x_1} x_2^T L x_2 \quad \lambda_3 = \dots$
- **f_{C_1} is the second smallest eigenvector of L !**
 - recall: $\vec{1}$ is the smallest eigenvector
 - $L \cdot f_{C_1} = f_{C_1} \cdot \lambda_2 \Leftrightarrow f_{C_1}^T \cdot L \cdot f_{C_1} = |V| \cdot \lambda_2$
- Example:
 - $f_{C_1} = [1.1841 \quad 0.6883 \quad 1.0620 \quad -0.6917 \quad -1.0827 \quad -1.1600]^T$

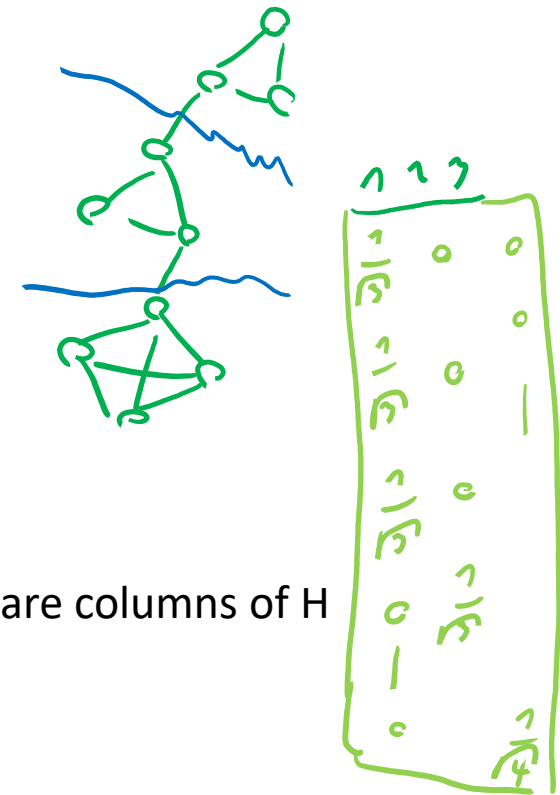
Spectral Clustering: 2 Clusters

- Solution: f_{C_1} is the second smallest eigenvector of L !
- Example:
 - $f_{C_1} = [1.1841 \quad 0.6883 \quad 1.0620 \quad -0.6917 \quad -1.0827 \quad -1.1600]^T$
- How to get the actual clustering?
 - simple case: consider the sign of the values in f_{C_1}
 - in general: perform k-means clustering of the values in f_{C_1}



Spectral Clustering: General Case

- General case ($k > 2$): clusters C_1, \dots, C_k
- Define indicator vector:
$$h_C[i] = \begin{cases} \frac{1}{\sqrt{|C|}} & \text{if } v_i \in C \\ 0 & \text{else} \end{cases}$$
 - let $H = [h_{C_1}; h_{C_2}; \dots; h_{C_k}]$ // indicator vectors are columns of H



Observations:

- $H^T H = Id$ // orthonormal matrix
 - $h_{C_i}^T \cdot L \cdot h_{C_i} = \frac{cut(C_i, V \setminus C_i)}{|C_i|}$ and $h_{C_i}^T \cdot L \cdot h_{C_i} = (H^T L H)_{ii}$
- $RatioCut(C_1, \dots, C_k) = \sum_{i=1}^k \frac{cut(C_i, V \setminus C_i)}{|C_i|} = \sum_{i=1}^k (H^T L H)_{ii} = trace(H^T L H)$

Spectral Clustering: General Case

- Minimizing ratio-cut (general case) is equivalent to

$$\min_{C_1, \dots, C_k} \text{trace}(H^T L H) \text{ subject to } H^T H = Id \text{ and } H \text{ as defined before}$$

- Constraint relaxation: allow arbitrary values for H**

➤ Result: $\min_{H \in \mathbb{R}^{V \times k}} \text{trace}(H^T L H) \text{ subject to } H^T H = Id$

- standard trace minimization problem
- **optimal H = first k smallest eigenvectors of L** // see relation to PCA/SVD

Spectral Clustering: Example

0	1	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	1	0	1	0	2	0	0	0
0	0	0	1	2	0	0	0	1
0	0	0	0	0	0	0	3	1
0	0	0	0	0	0	3	0	1
0	0	0	0	0	1	1	1	0

Adjacency matrix W

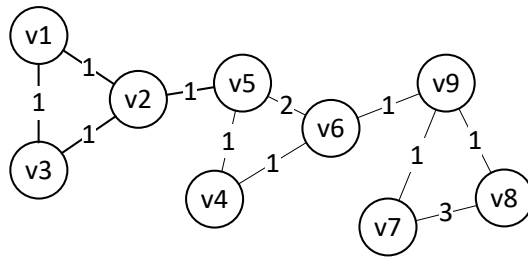
2	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	3

Degree matrix D

2	-1	-1	0	0	0	0	0	0
-1	3	-1	0	0	-1	0	0	0
-1	-1	2	0	0	0	0	0	0
0	0	0	2	-1	-1	0	0	0
0	-1	0	-1	4	-2	0	0	0
0	0	0	-1	-2	4	0	0	-1
0	0	0	0	0	0	4	-3	-1
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	-1	-1	-1	3

Laplacian matrix L

- Smallest eigenvalues of L : 0 ; 0.23 ; 0.7

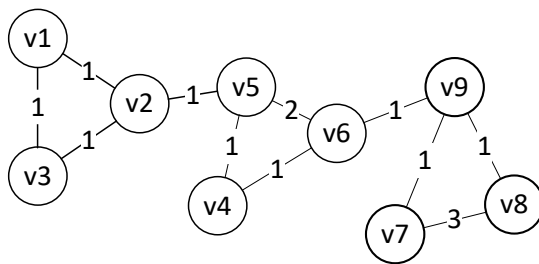


-0.3333	-0.4376	0.2939
-0.3333	-0.3370	0.0890
-0.3333	-0.4376	0.2939
-0.3333	0.0000	-0.5878
-0.3333	-0.0584	-0.3829
-0.3333	0.0584	-0.3829
-0.3333	0.4376	0.2939
-0.3333	0.4376	0.2939
-0.3333	0.3370	0.0890

*Eigenvectors
of L*

Spectral Clustering: Example

- How to find the clusters based on the eigenvectors?
- Represent each vertex by a vector of its corresponding components in the eigenvectors → **spectral embeddings** in the next lecture
- Clustering (e.g. k-Means) in the **embedding space** yields the final result



Representation of vertex v9:
 $(-0.333, 0.337, 0.0890)$

k first eigenvectors

-0.3333	-0.4376	0.2939
-0.3333	-0.3370	0.0890
-0.3333	-0.4376	0.2939
-0.3333	0.0000	-0.5878
-0.3333	-0.0584	-0.3829
-0.3333	0.0584	-0.3829
-0.3333	0.4376	0.2939
-0.3333	0.4376	0.2939
-0.3333	0.3370	0.0890

result of k-Means

first eigenvector can be ignored
 since constant anyway

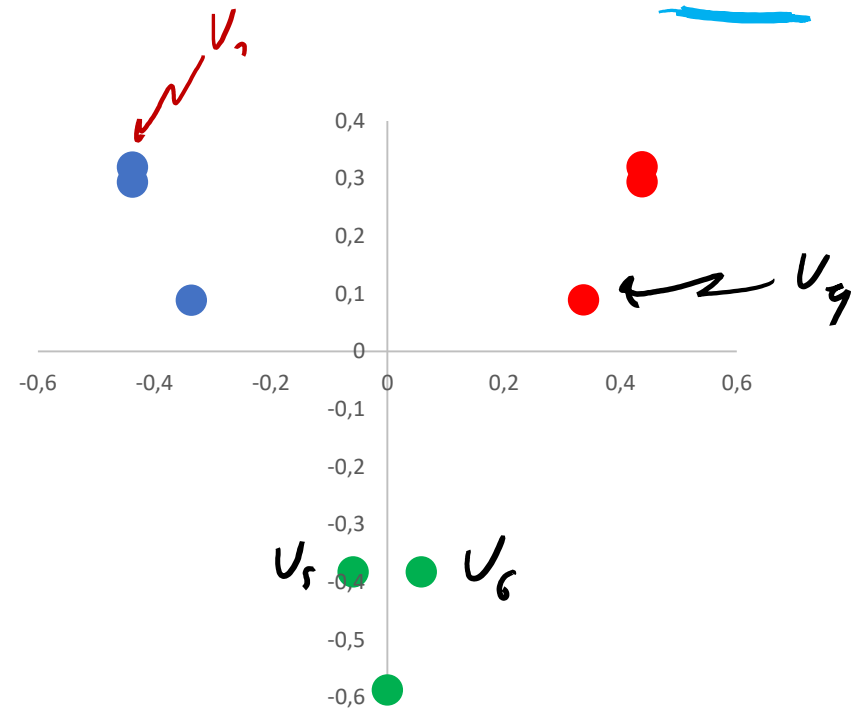
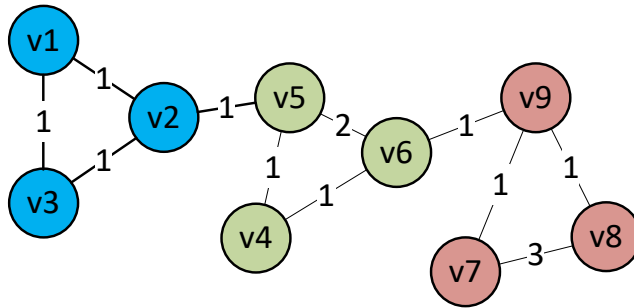
Spectral Clustering: Example of Spectral Embedding

- Also known as **Spectral Layout**

$$X - \lambda_{max} \bar{I} = \tilde{X}$$

$\lambda: 0, 1, 5, 20$

$\hat{\lambda} = -20, -19, -15, 0$



$v_1 \rightarrow$

x axis	y axis
-0.3333	-0.4376
-0.3333	-0.3370
-0.3333	-0.4376
-0.3333	0.0000
-0.3333	-0.0584
-0.3333	0.0584
-0.3333	0.4376
-0.3333	0.4376
-0.3333	0.3370

$v_9 \rightarrow$

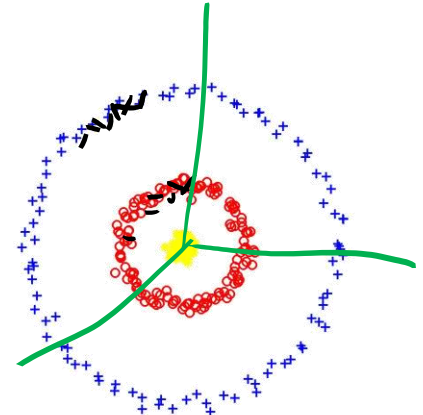
Spectral Clustering: Remarks

- Spectral clustering using **unnormalized Laplacian** $L = D - W$
 - approximation of ratio cut
- Spectral clustering using **normalized Laplacian** $L_{sym} = D^{-1/2} L D^{-1/2}$
 - approximation of normalized cut
- Drawback: No guarantee to achieve a result close to optimal cut!
 - but often performs very well in practice

Spectral Clustering for Numerical Data



- Spectral clustering is also used for other data types, e.g. numerical data
- **Step 1: Construct similarity graph**
 - requires similarity function between data instances
 - frequently used: Gaussian radial basis function kernel $\text{sim}(x, x') = e^{-\gamma \cdot \|x - x'\|^2}$
 - different variants of similarity graphs possible:
 - k-NN graph: connect two points if at least one of them is k-NN of the other
i.e. $(u, v) \in E \Leftrightarrow u \in NN_v(k) \vee v \in NN_u(k)$
 - neighborhood graph: connects all points whose distance is in specific range
i.e. $(u, v) \in E \Leftrightarrow \text{sim}(u, v) \geq \delta$
- **Step 2: Apply spectral clustering on similarity graph**
- Strong advantage of spectral clustering:
able to detect clusters of complex shapes



Roadmap

- **Chapter: Graphs**

1. Graphs & Networks
2. Generative Models
- 3. Clustering**
 - Introduction
 - Cuts & Spectral Clustering
 - **Probabilistic Approaches**
4. Node Embeddings
5. Ranking
6. Semi-Supervised Learning
7. Limitations of GNNs

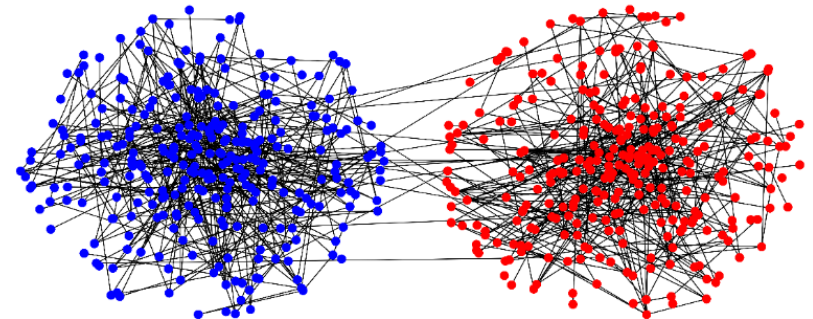
Probabilistic Community Detection

- Optimization view on graph clustering
 - find a community assignment that **optimizes** some criterion (e.g. minimum cut, maximum modularity)
- Alternatively: Probabilistic view
 - consider the graph as a realization (**sample**) drawn from a generative model
 - the generative process is controlled by a set of **parameters**
 - **communities** are explicitly modeled within the generative process
 - seen before: PPM, SBM
 - finding a “good” community assignment \Leftrightarrow performing **inference** in the model
- Advantages of the probabilistic view
 - capture uncertainty
 - handle missing / noisy data
 - generate new data (e.g. link prediction)

Recap: Planted Partition Model (PPM)

- We start with a set of nodes V , partitioned into 2 communities C_1, C_2
 - denote community assignment of node i as $z_i \in \{-1, 1\}$ – latent variables
- We generate an edge between every pair of nodes with probability

$$Pr(A_{ij} = 1 | z_i, z_j) = \begin{cases} p & \text{if } z_i = z_j \\ q & \text{if } z_i \neq z_j \end{cases}$$



Graph generated by a PPM with
 $N = 600, p = 6/600, q = 0.1/600$
 $z_i = -1$ for blue nodes, $z_i = 1$ for red nodes

Inference in PPM

- The likelihood of a community assignment $\mathbf{z} \in \{-1, 1\}^N$ for an observed symmetric adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$ is

$$\Pr(\mathbf{A}|\mathbf{z}) = \prod_{i < j} [p^{A_{ij}}(1-p)^{1-A_{ij}}]^{\mathbb{I}(z_i = z_j)} [q^{A_{ij}}(1-q)^{1-A_{ij}}]^{\mathbb{I}(z_i \neq z_j)}$$

- Assume that p and q are known.
- Given an observed \mathbf{A} , what is the most likely community assignment \mathbf{z}^* that produced it?
 - Community detection problem \Leftrightarrow probabilistic inference problem!
 - Simplest solution – maximum likelihood estimation

$$\mathbf{z}^* = \arg \max_{\mathbf{z} \in \{-1, 1\}^N} \Pr(\mathbf{A}|\mathbf{z})$$

PPM and Spectral Clustering (I)

- How do we find the ML estimate of \mathbf{z} for the planted partition model?
 - no closed-form solution
 - \mathbf{z} is discrete \Rightarrow gradient descent doesn't work
- Let's assume that the communities are balanced: $|C_1| = |C_2| = \frac{N}{2}$
 - equivalent to the constraint $\sum_i z_i = 0$
- Let's denote the number of edges whose endpoints are in different communities as

$$E_{cut}(\mathbf{z}) = |\{(i, j) \in E \text{ s.t. } z_i \neq z_j\}| = \sum_{(i, j) \in E} \mathbb{I}(z_i \neq z_j)$$

- We can show that the likelihood of the PPM is proportional to

$$\Pr(\mathbf{A}|\mathbf{z}) \propto \left(\frac{(1-p)q}{(1-q)p} \right)^{E_{cut}(\mathbf{z})}$$

PPM and Spectral Clustering (II)

$$\frac{0.01}{0.1} < 1$$

- We can show that the likelihood of the PPM is proportional to

$$\Pr(A|z) \propto \left(\frac{(1-p)q}{(1-q)p} \right)^{E_{cut}(z)}$$

↑ max
↑ min
↑ min

- Since $p > q$, maximizing the likelihood is equivalent to finding a minimum balanced cut!
 - How can we solve that? ⇒ Spectral clustering!
- MLE in PPM (under some assumptions) ⇔ minimum cut ⇔ spectral clustering

Recap: Stochastic Block Model (SBM)

- **Stochastic block model** generalizes the PPM to graphs with arbitrary numbers and sizes of communities, and varying edge densities.

- Random variables:

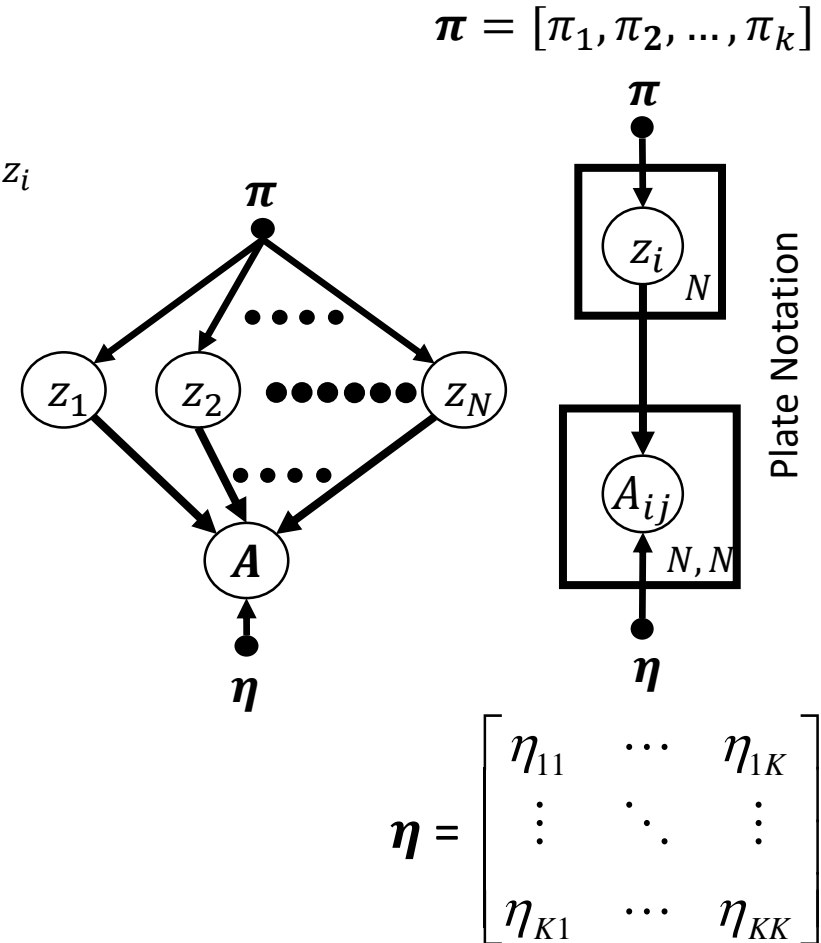
- $z_i \in \{1, \dots, K\}$: node i belongs to block/community z_i
- $A \in \{0,1\}^{N \times N}$: adjacency matrix

- Model parameters:

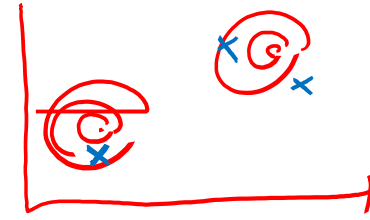
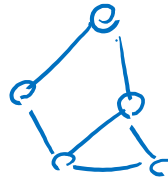
- $\pi = [\pi_1, \dots, \pi_K]$: community proportions
- η_{uv} : edge probability between two nodes that are in communities u and v .

- Conditional distributions:

- $\Pr(z_i = k) = \pi_k$
- $\Pr(A_{ij} | z_i, z_j) = \text{Bernoulli}(\eta_{z_i z_j})$



Inference in SBM



- Assume that η and π are known. What is the distribution of \mathbf{z} given \mathbf{A} ?

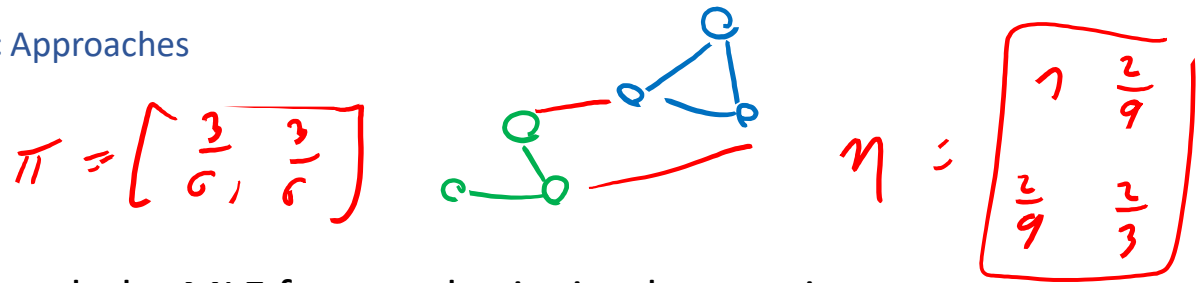
$$\Pr(\mathbf{z}|\mathbf{A}, \eta, \pi) = \frac{\Pr(\mathbf{A}|\mathbf{z}, \eta, \pi) \Pr(\mathbf{z}|\pi)}{\Pr(\mathbf{A}|\eta, \pi)}$$

- The normalizing constant $\Pr(\mathbf{A}|\eta, \pi)$ is intractable and requires $O(K^N)$ operations to compute.

- We can use, e.g., variational inference to find an approximate solution
 - find a distribution $q(\mathbf{z})$ that is similar to the true posterior $\Pr(\mathbf{z}|\mathbf{A}, \eta, \pi)$ (e.g. with a mean-field assumption)

$$\Pr(\mathbf{z}|\mathbf{A}, \eta, \pi) \approx q(\mathbf{z}|\Psi) = \prod_{i=1}^N q(\mathbf{z}_i|\psi_i)$$

Learning in SBM



- If both \mathbf{z} and \mathbf{A} are observed, the MLE for $\boldsymbol{\eta}$ and $\boldsymbol{\pi}$ is simple counting

$$\pi_k^{MLE} = \frac{\text{\# nodes in cluster } k}{N} =: \frac{N_k}{N}$$

$$\eta_{uv}^{MLE} = \frac{\text{observed \# edges between } u \text{ and } v}{\text{possible \# edges between } u \text{ and } v} = \frac{\sum_{(i,j) \in E} \mathbb{I}(z_i = u) \mathbb{I}(z_j = v)}{P_{uv}}$$

where $P_{uv} = \begin{cases} \binom{N_u}{2} & \text{if } u = v \\ N_u \cdot N_v & \text{if } u \neq v \end{cases}$ is the number of possible edges between clusters u and v

- If only \mathbf{A} is observed, we can use again variational inference
 - i.e. optimize the ELBO w.r.t. $q(\mathbf{z})$ as well as $\boldsymbol{\eta}$ and $\boldsymbol{\pi}$

Summary

- Graph Laplacian and its spectrum captures the connectivity structure of the graph
- Spectral clustering finds a partition of the graph that minimizes the number of edges between different parts, a minimum cut.
 - relax an NP-hard problem to a continuous trace minimization problem
 - optimal solution is given by the eigenvectors of the graph Laplacian
- Clustering can be framed as inference in a generative model such as the Stochastic Block Model or its special case PPM.
 - advantages: handles uncertainty, more robust against noise, finds a generative model
 - disadvantages: inference is intractable, doesn't model all known patterns

Questions

- How can you find the connected components of a graph from its Laplacian?
- Consider a graph with n arbitrarily connected nodes and k disconnected nodes. What are the first $k + 1$ clusters that spectral clustering finds? Why?

Reading Material

- Aggarwal, C., Wang, H.: Managing and Mining Graph Data, Chapter 9 and 10
- Fortunato, S.: Community detection in graphs, in Physics Reports, 2010
- Tutorial and overview about different spectral clustering approaches: "Von Luxburg, U.: A tutorial on spectral clustering"