Graphs in Machine Learning

Daniele Calandriello

DeepMind Paris, France

Collaborators: Achraf Azize,
Michal Valko

Partially based on material by: Ulrike von Luxburg, Gary Miller, Doyle & Schnell, Daniel Spielman

Previous lecture

- where do the graphs come from?
  - social, information, utility, and biological networks
  - we create them from the flat data
  - random graph models
- specific applications and concepts
  - maximizing influence on a graph **gossip propagation, submodularity**, proof of the approximation guarantee
  - Google pagerank **random surfer process, steady state vector, sparsity**
  - online semi-supervised learning **label propagation, backbone graph, online learning, combinatorial sparsification, stability analysis**
  - Erdős number project, real-world graphs, **heavy tails, small world** – when did this happen?
- similarity graphs
  - different types
  - construction
  - practical considerations

This Lecture

- spectral graph theory
- Laplacians and their properties
  - symmetric and asymmetric normalization
  - random walks
- geometry of the data and the connectivity
- spectral clustering
- manifold learning with Laplacians eigenmaps

Next Class: Lab Session

- 30. 01. 2022 by Achraf
- Short written report (graded)
- All homeworks together make up the final grade
- Content
  - Graph Construction
  - Test sensitivity to parameters: σ, k, ε
  - Spectral Clustering
  - Spectral Clustering vs. k-means
  - Image Segmentation

Similarity Graphs

Input:

$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_N$

- raw data
- flat data
- vectorial data

Similarity Graphs

Similarity graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ — **(un)weighted**

Task 1: For each pair i, j: define a **similarity function** s_{ij}

Task 2: Decide which edges to include

Similarity Graphs

Similarity graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ — **(un)weighted**

Task 1: For each pair i, j: define a **similarity function** s_{ij}
Task 2: Decide which edges to include

fully connected graphs - consider everything

ε-neighborhood graphs – connect the points with the distances smaller than ε

k-NN neighborhood graphs – take k nearest neighbors

Similarity Graphs

Similarity graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ — **(un)weighted**

Task 1: For each pair i, j: define a **similarity function** s_{ij}
Task 2: Decide which edges to include

fully connected graphs - consider everything

ε-neighborhood graphs – connect the points with the distances smaller than ε

k-NN neighborhood graphs – take k nearest neighbors

This is art (not much theory exists).

`http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/publications/Luxburg07_tutorial.pdf`

Similarity Graphs: Fully connected graphs

Edges connect everything.

Similarity Graphs: Fully connected graphs

Edges connect everything.

- choose a "meaningful" similarity function s
- default choice:

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

- why the exponential decay with the distance?

Similarity Graphs: Fully connected graphs

Edges connect everything.

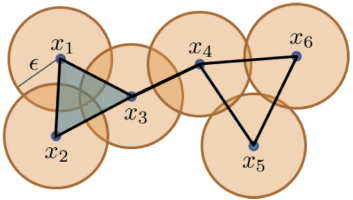
- choose a "meaningful" similarity function s
- default choice:

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

- why the exponential decay with the distance?
- σ controls the width of the neighborhoods
  - **a** practical rule of thumb: 10% of the average empirical std
  - possibility: learn σ_i for each feature independently
- metric learning (a whole field of ML)

Similarity Graphs: ε-neighborhood graphs

Edges connect the points with the distances smaller than ε.



Similarity Graphs: ε-neighborhood graphs

Edges connect the points with the distances smaller than ε.

- distances are roughly on the same scale (ε)
- weights may not bring additional info $\rightarrow$ unweighted
- equivalent to: similarity function is at least ε
- theory [Penrose, 1999]: $\varepsilon = ((\log N)/N)^d$ to guarantee connectivity N nodes, d dimension
- practice: choose ε as the length of the longest edge in the MST - minimum spanning tree

What could be the problem with this MST approach?

Similarity Graphs: ε-neighborhood graphs

Edges connect the points with the distances smaller than ε.

- distances are roughly on the same scale (ε)
- weights may not bring additional info $\rightarrow$ unweighted
- equivalent to: similarity function is at least ε
- theory [Penrose, 1999]: $\varepsilon = ((\log N)/N)^d$ to guarantee connectivity N nodes, d dimension
- practice: choose ε as the length of the longest edge in the MST - minimum spanning tree

What could be the problem with this MST approach?

Anomalies can make ε too large.

Similarity Graphs: k-nearest neighbors graphs

Edges connect each node to its k-nearest neighbors.

Similarity Graphs: k-nearest neighbors graphs

Edges connect each node to its k-nearest neighbors.

- asymmetric (or directed graph)
  - option OR: ignore the direction
  - option AND: include if we have both direction (mutual k-NN)
- how to choose k?
- $k \approx \log N$ - suggested by asymptotics (practice: up to $\sqrt{N}$)
- for mutual k-NN we need to take larger k
- mutual k-NN does not connect regions with different density
- why don't we take $k = N - 1$?
  - space and time
  - manifold considerations (preserving local properties)

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN
    - **LSH** - Locally Sensitive Hashing

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN
    - **LSH** - Locally Sensitive Hashing
    - **CoverTrees**

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN
    - **LSH** - Locally Sensitive Hashing
    - **CoverTrees**
  - sometime we may not need the graph (just the final results)

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN
    - **LSH** - Locally Sensitive Hashing
    - **CoverTrees**
  - sometime we may not need the graph (just the final results)
  - yet another story: when we start with a large graph and want to make it sparse (later in the course)

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN
    - **LSH** - Locally Sensitive Hashing
    - **CoverTrees**
  - sometime we may not need the graph (just the final results)
  - yet another story: when we start with a large graph and want to make it sparse (later in the course)
- these rules have little theoretical underpinning

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($N \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN
    - **LSH** - Locally Sensitive Hashing
    - **CoverTrees**
  - sometime we may not need the graph (just the final results)
  - yet another story: when we start with a large graph and want to make it sparse (later in the course)
- these rules have little theoretical underpinning
- similarity is very data-dependent

$$L = D - W$$

graph Laplacian

...the only matrix that matters

Graph Laplacian

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - with a set of **nodes** $\mathcal{V}$ and a set of **edges** $\mathcal{E}$

Graph Laplacian

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - with a set of **nodes** $\mathcal{V}$ and a set of **edges** $\mathcal{E}$

A adjacency matrix

Graph Laplacian

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - with a set of **nodes** $\mathcal{V}$ and a set of **edges** $\mathcal{E}$

A	adjacency matrix
W	weight matrix

Graph Laplacian

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - with a set of **nodes** $\mathcal{V}$ and a set of **edges** $\mathcal{E}$

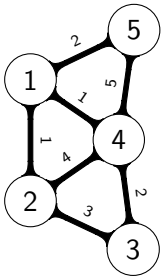
A	adjacency matrix
W	weight matrix
D	(diagonal) degree matrix

Graph Laplacian

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - with a set of **nodes** $\mathcal{V}$ and a set of **edges** $\mathcal{E}$

$\mathbf{A}$	adjacency matrix
$\mathbf{W}$	weight matrix
$\mathbf{D}$	(diagonal) degree matrix
$\mathbf{L} = \mathbf{D} - \mathbf{W}$	graph **Laplacian** matrix

$$\mathbf{L} = \begin{pmatrix} 4 & -1 & 0 & -1 & -2 \\ -1 & 8 & -3 & -4 & 0 \\ 0 & -3 & 5 & -2 & 0 \\ -1 & -4 & -2 & 12 & -5 \\ -2 & 0 & 0 & -5 & 7 \end{pmatrix}$$



L is SDD!

demo: `https://dominikschmidt.xyz/spectral-clustering-exp/`

Properties of Graph Laplacian

Graph function: a vector $\mathbf{f} \in \mathbb{R}^N$ assigning values to nodes:

$$\mathbf{f} : \mathcal{V}(\mathcal{G}) \to \mathbb{R}.$$

Properties of Graph Laplacian

Graph function: a vector $\mathbf{f} \in \mathbb{R}^N$ assigning values to nodes:

$$\mathbf{f} : \mathcal{V}(\mathcal{G}) \to \mathbb{R}.$$

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j \leq N} w_{i,j} (f_i - f_j)^2$$

Properties of Graph Laplacian

Graph function: a vector $\mathbf{f} \in \mathbb{R}^N$ assigning values to nodes:

$$\mathbf{f} : \mathcal{V}(\mathcal{G}) \to \mathbb{R}.$$

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2 = S_G(\mathbf{f})$$

Properties of Graph Laplacian

Graph function: a vector $\mathbf{f} \in \mathbb{R}^N$ assigning values to nodes:

$$\mathbf{f} : \mathcal{V}(\mathcal{G}) \to \mathbb{R}.$$

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j \leq N} w_{i,j} (f_i - f_j)^2 = S_G(\mathbf{f})$$

Proof:

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \sum_{i,j \leq N} \mathbf{L}_{i,j} f_i f_j = \sum_{i,j \leq N} \mathbf{D}_{i,j} f_i f_j - \sum_{i,j \leq N} \mathbf{W}_{i,j} f_i f_j = \sum_{i=1}^{N} d_i f_i^2 - \sum_{i,j \leq N} w_{i,j} f_i f_j$$

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

Recap: Eigenwerte und Eigenvektoren

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

Recap: Eigenwerte und Eigenvektoren

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

If $(\lambda_1, \mathbf{v}_1)$ are $(\lambda_2, \mathbf{v}_2)$ **eigenpairs** for symmetric $\mathbf{M}$ with $\lambda_1 \neq \lambda_2$ then $\mathbf{v}_1 \perp \mathbf{v}_2$, i.e., $\mathbf{v}_1^\intercal \mathbf{v}_2 = 0$.

Recap: Eigenwerte und Eigenvektoren

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

If $(\lambda_1, \mathbf{v}_1)$ are $(\lambda_2, \mathbf{v}_2)$ **eigenpairs** for symmetric $\mathbf{M}$ with $\lambda_1 \neq \lambda_2$ then $\mathbf{v}_1 \perp \mathbf{v}_2$, i.e., $\mathbf{v}_1^\intercal \mathbf{v}_2 = 0$.

Proof: $\lambda_1 \mathbf{v}_1^\intercal \mathbf{v}_2 = \mathbf{v}_1^\intercal \mathbf{M} \mathbf{v}_2 = \mathbf{v}_1^\intercal \lambda_2 \mathbf{v}_2 = \lambda_2 \mathbf{v}_1^\intercal \mathbf{v}_2 \implies \mathbf{v}_1^\intercal \mathbf{v}_2 = 0$

Recap: Eigenwerte und Eigenvektoren

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

If $(\lambda_1, \mathbf{v}_1)$ are $(\lambda_2, \mathbf{v}_2)$ **eigenpairs** for symmetric $\mathbf{M}$ with $\lambda_1 \neq \lambda_2$ then $\mathbf{v}_1 \perp \mathbf{v}_2$, i.e., $\mathbf{v}_1^\intercal \mathbf{v}_2 = 0$.

Proof: $\lambda_1 \mathbf{v}_1^\intercal \mathbf{v}_2 = \mathbf{v}_1^\intercal \mathbf{M} \mathbf{v}_2 = \mathbf{v}_1^\intercal \lambda_2 \mathbf{v}_2 = \lambda_2 \mathbf{v}_1^\intercal \mathbf{v}_2 \implies \mathbf{v}_1^\intercal \mathbf{v}_2 = 0$

If $(\lambda, \mathbf{v}_1)$, $(\lambda, \mathbf{v}_2)$ are eigenpairs for $\mathbf{M}$ then $(\lambda, \mathbf{v}_1 + \mathbf{v}_2)$ is as well.

Recap: Eigenwerte und Eigenvektoren

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

If $(\lambda_1, \mathbf{v}_1)$ are $(\lambda_2, \mathbf{v}_2)$ **eigenpairs** for symmetric $\mathbf{M}$ with $\lambda_1 \neq \lambda_2$ then $\mathbf{v}_1 \perp \mathbf{v}_2$, i.e., $\mathbf{v}_1^\intercal \mathbf{v}_2 = 0$.

Proof: $\lambda_1 \mathbf{v}_1^\intercal \mathbf{v}_2 = \mathbf{v}_1^\intercal \mathbf{M} \mathbf{v}_2 = \mathbf{v}_1^\intercal \lambda_2 \mathbf{v}_2 = \lambda_2 \mathbf{v}_1^\intercal \mathbf{v}_2 \implies \mathbf{v}_1^\intercal \mathbf{v}_2 = 0$

If $(\lambda, \mathbf{v}_1)$, $(\lambda, \mathbf{v}_2)$ are eigenpairs for $\mathbf{M}$ then $(\lambda, \mathbf{v}_1 + \mathbf{v}_2)$ is as well.

For symmetric $\mathbf{M}$, the **multiplicity** of λ is the dimension of the space of eigenvectors corresponding to λ.

$N \times N$ symmetric matrix has N eigenvalues (w/ multiplicities).

Eigenvalues, Eigenvectors, and Eigendecomposition

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

Vectors $\{\mathbf{v}_i\}_i$ form an **orthonormal** basis with $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_N$.

$$\forall i \quad \mathbf{M}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

Eigenvalues, Eigenvectors, and Eigendecomposition

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

Vectors $\{\mathbf{v}_i\}_i$ form an **orthonormal** basis with $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_N$.

$$\forall i \quad \mathbf{M}\mathbf{v}_i = \lambda_i \mathbf{v}_i \qquad \equiv \qquad \mathbf{M}\mathbf{Q} = \mathbf{Q}\Lambda$$

$\mathbf{Q}$ has eigenvectors in columns and Λ has eigenvalues on its diagonal.

Eigenvalues, Eigenvectors, and Eigendecomposition

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** λ

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

Vectors $\{\mathbf{v}_i\}_i$ form an **orthonormal** basis with $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_N$.

$$\forall i \quad \mathbf{M}\mathbf{v}_i = \lambda_i \mathbf{v}_i \qquad \equiv \qquad \mathbf{MQ} = \mathbf{Q}\Lambda$$

$\mathbf{Q}$ has eigenvectors in columns and Λ has eigenvalues on its diagonal.

Right-multiplying $\mathbf{MQ} = \mathbf{Q}\Lambda$ by $\mathbf{Q}^\intercal$ we get the **eigendecomposition** of $\mathbf{M}$:

$$\mathbf{M} = \mathbf{MQQ}^\intercal = \mathbf{Q}\Lambda\mathbf{Q}^\intercal = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^\intercal$$

M = L: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

M = L: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

L is symmetric

$\mathbf{M} = \mathbf{L}$: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

L is symmetric

L positive semi-definite

M = L: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

L is symmetric

L positive semi-definite $\leftarrow \mathbf{f}^\intercal \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$

$\mathbf{M} = \mathbf{L}$: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

L is symmetric

L positive semi-definite $\leftarrow \mathbf{f}^\intercal \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$

Recall: If $\mathbf{L}\mathbf{f} = \lambda \mathbf{f}$ then λ is an **eigenvalue** (of the Laplacian).

M = L: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

L is symmetric

L positive semi-definite $\leftarrow \mathbf{f}^\intercal \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$

Recall: If $\mathbf{L}\mathbf{f} = \lambda \mathbf{f}$ then λ is an **eigenvalue** (of the Laplacian).

The smallest eigenvalue of **L** is 0. Corresponding eigenvector: $\mathbf{1}_N$.

M = L: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

L is symmetric

L positive semi-definite $\leftarrow \mathbf{f}^\intercal \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j\leq N} w_{i,j}(f_i - f_j)^2$

Recall: If $\mathbf{L}\mathbf{f} = \lambda\mathbf{f}$ then λ is an **eigenvalue** (of the Laplacian).

The smallest eigenvalue of **L** is 0. Corresponding eigenvector: $\mathbf{1}_N$.

All eigenvalues are non-negative reals $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$.

M = L: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

L is symmetric

L positive semi-definite $\leftarrow \mathbf{f}^\intercal \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$

Recall: If $\mathbf{L}\mathbf{f} = \lambda \mathbf{f}$ then λ is an **eigenvalue** (of the Laplacian).

The smallest eigenvalue of **L** is 0. Corresponding eigenvector: $\mathbf{1}_N$.

All eigenvalues are non-negative reals $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$.

Self-edges do not change the value of **L**.

Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of **L** equals to the number of connected components.
The eigenspace of 0 is spanned by the components' indicators.

Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of $\mathbf{L}$ equals to the number of connected components. The eigenspace of 0 is spanned by the components' indicators.

Proof: If $(0, \mathbf{f})$ is an eigenpair then $0 = \frac{1}{2} \sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$.

Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of $\mathbf{L}$ equals to the number of connected components. The eigenspace of 0 is spanned by the components' indicators.

Proof: If $(0, \mathbf{f})$ is an eigenpair then $0 = \frac{1}{2} \sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$. Therefore, $\mathbf{f}$ is constant on each connected component.

Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of $\mathbf{L}$ equals to the number of connected components. The eigenspace of 0 is spanned by the components' indicators.

Proof: If $(0, \mathbf{f})$ is an eigenpair then $0 = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$. Therefore, $\mathbf{f}$ is constant on each connected component. If there are k components, then $\mathbf{L}$ is k-block-diagonal:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_k \end{bmatrix}$$

Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of $\mathbf{L}$ equals to the number of connected components. The eigenspace of 0 is spanned by the components' indicators.

Proof: If $(0, \mathbf{f})$ is an eigenpair then $0 = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$. Therefore, $\mathbf{f}$ is constant on each connected component. If there are k components, then $\mathbf{L}$ is k-block-diagonal:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_k \end{bmatrix}$$

For block-diagonal matrices: the spectrum is the union of the spectra of $\mathbf{L}_i$

Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of $\mathbf{L}$ equals to the number of connected components. The eigenspace of 0 is spanned by the components' indicators.

Proof: If $(0, \mathbf{f})$ is an eigenpair then $0 = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$. Therefore, $\mathbf{f}$ is constant on each connected component. If there are k components, then $\mathbf{L}$ is k-block-diagonal:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_k \end{bmatrix}$$

For block-diagonal matrices: the spectrum is the union of the spectra of $\mathbf{L}_i$ (eigenvectors of $\mathbf{L}_i$ padded with zeros elsewhere).

Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of $\mathbf{L}$ equals to the number of connected components. The eigenspace of 0 is spanned by the components' indicators.

Proof: If $(0, \mathbf{f})$ is an eigenpair then $0 = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2$. Therefore, $\mathbf{f}$ is constant on each connected component. If there are k components, then $\mathbf{L}$ is k-block-diagonal:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_k \end{bmatrix}$$

For block-diagonal matrices: the spectrum is the union of the spectra of $\mathbf{L}_i$ (eigenvectors of $\mathbf{L}_i$ padded with zeros elsewhere).

For $\mathbf{L}_i$ $(0, \mathbf{1}_{|V_i|})$ is an eigenpair, hence the claim.

Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{Q}\Lambda\mathbf{Q}^\top \mathbf{f} = \boldsymbol{\alpha}^\top \Lambda \boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{Q}\Lambda\mathbf{Q}^\top \mathbf{f} = \boldsymbol{\alpha}^\top \Lambda \boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{Q}\Lambda\mathbf{Q}^\top \mathbf{f} = \boldsymbol{\alpha}^\top \Lambda \boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

Spectral coordinates of eigenvector $\mathbf{v}_k$: $\mathbf{Q}^\top \mathbf{v}_k =$

Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{Q}\Lambda\mathbf{Q}^\top \mathbf{f} = \boldsymbol{\alpha}^\top \Lambda \boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

Spectral coordinates of eigenvector $\mathbf{v}_k$: $\mathbf{Q}^\top \mathbf{v}_k = \mathbf{e}_k$

Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{Q}\Lambda\mathbf{Q}^\top\mathbf{f} = \boldsymbol{\alpha}^\top\Lambda\boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

Spectral coordinates of eigenvector $\mathbf{v}_k$: $\mathbf{Q}^\top\mathbf{v}_k = \mathbf{e}_k$

$$S_G(\mathbf{v}_k) = \mathbf{v}_k^\top \mathbf{L}\mathbf{v}_k = \mathbf{v}_k^\top \mathbf{Q}\Lambda\mathbf{Q}^\top\mathbf{v}_k = \mathbf{e}_k^\top\Lambda\mathbf{e}_k = \|\mathbf{e}_k\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i (\mathbf{e}_k)_i^2$$

Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{Q}\Lambda\mathbf{Q}^\top\mathbf{f} = \boldsymbol{\alpha}^\top\Lambda\boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

Spectral coordinates of eigenvector $\mathbf{v}_k$: $\mathbf{Q}^\top\mathbf{v}_k = \mathbf{e}_k$

$$S_G(\mathbf{v}_k) = \mathbf{v}_k^\top \mathbf{L}\mathbf{v}_k = \mathbf{v}_k^\top \mathbf{Q}\Lambda\mathbf{Q}^\top\mathbf{v}_k = \mathbf{e}_k^\top\Lambda\mathbf{e}_k = \|\mathbf{e}_k\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i (\mathbf{e}_k)_i^2 = \lambda_k$$

Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{Q}\Lambda\mathbf{Q}^\top \mathbf{f} = \boldsymbol{\alpha}^\top \Lambda \boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

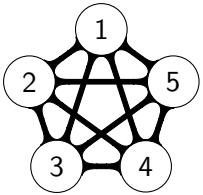
Spectral coordinates of eigenvector $\mathbf{v}_k$: $\mathbf{Q}^\top \mathbf{v}_k = \mathbf{e}_k$

$$S_G(\mathbf{v}_k) = \mathbf{v}_k^\top \mathbf{L}\mathbf{v}_k = \mathbf{v}_k^\top \mathbf{Q}\Lambda\mathbf{Q}^\top \mathbf{v}_k = \mathbf{e}_k^\top \Lambda \mathbf{e}_k = \|\mathbf{e}_k\|_\Lambda^2 = \sum_{i=1}^{N} \lambda_i (\mathbf{e}_k)_i^2 = \lambda_k$$

The smoothness of k-th eigenvector is the k-th eigenvalue.

Laplacian of the Complete Graph K_N

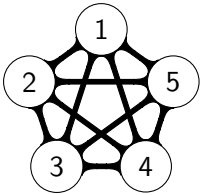
What is the eigenspectrum of $\mathbf{L}_{K_N}$?



$$\mathbf{L}_{K_N} = \begin{pmatrix} N-1 & -1 & -1 & -1 & -1 \\ -1 & N-1 & -1 & -1 & -1 \\ -1 & -1 & N-1 & -1 & -1 \\ -1 & -1 & -1 & N-1 & -1 \\ -1 & -1 & -1 & -1 & N-1 \end{pmatrix}$$

Laplacian of the Complete Graph K_N

What is the eigenspectrum of $\mathbf{L}_{K_N}$?



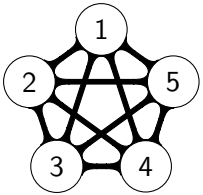
$$\mathbf{L}_{K_N} = \begin{pmatrix} N-1 & -1 & -1 & -1 & -1 \\ -1 & N-1 & -1 & -1 & -1 \\ -1 & -1 & N-1 & -1 & -1 \\ -1 & -1 & -1 & N-1 & -1 \\ -1 & -1 & -1 & -1 & N-1 \end{pmatrix}$$

From before: we know that $(0, \mathbf{1}_N)$ is an eigenpair.

If $\mathbf{v} \neq 0_N$ and $\mathbf{v} \perp \mathbf{1}_N \implies \sum_i \mathbf{v}_i = 0$.

Laplacian of the Complete Graph K_N

What is the eigenspectrum of $\mathbf{L}_{K_N}$?



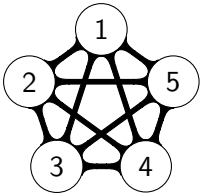
$$\mathbf{L}_{K_N} = \begin{pmatrix} N-1 & -1 & -1 & -1 & -1 \\ -1 & N-1 & -1 & -1 & -1 \\ -1 & -1 & N-1 & -1 & -1 \\ -1 & -1 & -1 & N-1 & -1 \\ -1 & -1 & -1 & -1 & N-1 \end{pmatrix}$$

From before: we know that $(0, \mathbf{1}_N)$ is an eigenpair.

If $\mathbf{v} \neq 0_N$ and $\mathbf{v} \perp \mathbf{1}_N \implies \sum_i \mathbf{v}_i = 0$. To get the other eigenvalues, we compute $(\mathbf{L}_{K_N}\mathbf{v})_1$ and divide by $\mathbf{v}_1$ (wlog $\mathbf{v}_1 \neq 0$).

Laplacian of the Complete Graph K_N

What is the eigenspectrum of $\mathbf{L}_{K_N}$?



$$\mathbf{L}_{K_N} = \begin{pmatrix} N-1 & -1 & -1 & -1 & -1 \\ -1 & N-1 & -1 & -1 & -1 \\ -1 & -1 & N-1 & -1 & -1 \\ -1 & -1 & -1 & N-1 & -1 \\ -1 & -1 & -1 & -1 & N-1 \end{pmatrix}$$

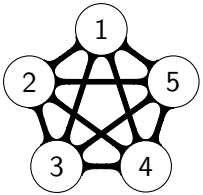
From before: we know that $(0, \mathbf{1}_N)$ is an eigenpair.

If $\mathbf{v} \neq 0_N$ and $\mathbf{v} \perp \mathbf{1}_N \implies \sum_i \mathbf{v}_i = 0$. To get the other eigenvalues, we compute $(\mathbf{L}_{K_N}\mathbf{v})_1$ and divide by $\mathbf{v}_1$ (wlog $\mathbf{v}_1 \neq 0$).

$$(\mathbf{L}_{K_N}\mathbf{v})_1 = (N-1)\mathbf{v}_1 - \sum_{i=2}^{N} \mathbf{v}_i = N\mathbf{v}_1.$$

Laplacian of the Complete Graph K_N

What is the eigenspectrum of $\mathbf{L}_{K_N}$?



$$\mathbf{L}_{K_N} = \begin{pmatrix} N-1 & -1 & -1 & -1 & -1 \\ -1 & N-1 & -1 & -1 & -1 \\ -1 & -1 & N-1 & -1 & -1 \\ -1 & -1 & -1 & N-1 & -1 \\ -1 & -1 & -1 & -1 & N-1 \end{pmatrix}$$

From before: we know that $(0, \mathbf{1}_N)$ is an eigenpair.

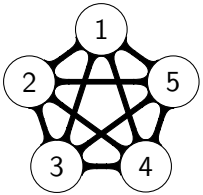
If $\mathbf{v} \neq 0_N$ and $\mathbf{v} \perp \mathbf{1}_N \implies \sum_i \mathbf{v}_i = 0$. To get the other eigenvalues, we compute $(\mathbf{L}_{K_N}\mathbf{v})_1$ and divide by $\mathbf{v}_1$ (wlog $\mathbf{v}_1 \neq 0$).

$$(\mathbf{L}_{K_N}\mathbf{v})_1 = (N-1)\mathbf{v}_1 - \sum_{i=2}^{N} \mathbf{v}_i = N\mathbf{v}_1.$$

What are the remaining eigenvalues/vectors?

Laplacian of the Complete Graph K_N

What is the eigenspectrum of $\mathbf{L}_{K_N}$?



$$\mathbf{L}_{K_N} = \begin{pmatrix} N-1 & -1 & -1 & -1 & -1 \\ -1 & N-1 & -1 & -1 & -1 \\ -1 & -1 & N-1 & -1 & -1 \\ -1 & -1 & -1 & N-1 & -1 \\ -1 & -1 & -1 & -1 & N-1 \end{pmatrix}$$

From before: we know that $(0, \mathbf{1}_N)$ is an eigenpair.

If $\mathbf{v} \neq 0_N$ and $\mathbf{v} \perp \mathbf{1}_N \implies \sum_i \mathbf{v}_i = 0$. To get the other eigenvalues, we compute $(\mathbf{L}_{K_N}\mathbf{v})_1$ and divide by $\mathbf{v}_1$ (wlog $\mathbf{v}_1 \neq 0$).

$$(\mathbf{L}_{K_N}\mathbf{v})_1 = (N-1)\mathbf{v}_1 - \sum_{i=2}^{N} \mathbf{v}_i = N\mathbf{v}_1.$$

What are the remaining eigenvalues/vectors?

Answer: $N-1$ eigenvectors $\perp \mathbf{1}_N$ for eigenvalue N with multiplicity $N-1$.

Normalized Laplacians

$$\mathbf{L}_{un} = \mathbf{D} - \mathbf{W}$$

Normalized Laplacians

$$\mathbf{L}_{un} = \mathbf{D} - \mathbf{W}$$
$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

Normalized Laplacians

$$\mathbf{L}_{un} = \mathbf{D} - \mathbf{W}$$
$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$
$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$$

Normalized Laplacians

$$\mathbf{L}_{un} = \mathbf{D} - \mathbf{W}$$
$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$
$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$$

$$\mathbf{f}^{\mathsf{T}}\mathbf{L}_{sym}\mathbf{f} = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

Normalized Laplacians

$$\mathbf{L}_{un} = \mathbf{D} - \mathbf{W}$$
$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$
$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$$

$$\mathbf{f}^{\mathsf{T}}\mathbf{L}_{sym}\mathbf{f} = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{D}^{1/2}\mathbf{u})$ is an eigenpair for $\mathbf{L}_{sym}$

Normalized Laplacians

$\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ are PSD with non-negative real eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{u})$ solve the generalized eigenproblem $\mathbf{Lu} = \lambda \mathbf{Du}$.

Normalized Laplacians

$\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ are PSD with non-negative real eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{u})$ solve the generalized eigenproblem $\mathbf{Lu} = \lambda \mathbf{Du}$.

$(0, \mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{rw}$.

Normalized Laplacians

$\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ are PSD with non-negative real eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{u})$ solve the generalized eigenproblem $\mathbf{Lu} = \lambda \mathbf{Du}$.

$(0, \mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{rw}$.

$(0, \mathbf{D}^{1/2}\mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{sym}$.

Normalized Laplacians

$\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ are PSD with non-negative real eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{u})$ solve the generalized eigenproblem $\mathbf{Lu} = \lambda \mathbf{Du}$.

$(0, \mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{rw}$.

$(0, \mathbf{D}^{1/2}\mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{sym}$.

Multiplicity of eigenvalue 0 of $\mathbf{L}_{rw}$ or $\mathbf{L}_{sym}$ equals to the number of connected components.

Normalized Laplacians

$\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ are PSD with non-negative real eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{u})$ solve the generalized eigenproblem $\mathbf{Lu} = \lambda \mathbf{Du}$.

$(0, \mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{rw}$.

$(0, \mathbf{D}^{1/2}\mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{sym}$.

Multiplicity of eigenvalue 0 of $\mathbf{L}_{rw}$ or $\mathbf{L}_{sym}$ equals to the number of connected components.

Proof: As for $\mathbf{L}$.

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)
- if G is connected and non-bipartite $\to$ unique **stationary distribution** $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ where $\pi_i = d_i/\mathrm{vol}(V)$

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)
- if G is connected and non-bipartite $\to$ unique **stationary distribution** $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ where $\pi_i = d_i/\mathrm{vol}(V)$
  - $\mathrm{vol}(G) = \mathrm{vol}(V) = \mathrm{vol}(\mathbf{W}) \stackrel{\text{def}}{=} \sum_i d_i = \sum_{i,j} w_{ij}$

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)
- if G is connected and non-bipartite $\to$ unique **stationary distribution** $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ where $\pi_i = d_i/\mathrm{vol}(V)$
  - $\mathrm{vol}(G) = \mathrm{vol}(V) = \mathrm{vol}(\mathbf{W}) \stackrel{\text{def}}{=} \sum_i d_i = \sum_{i,j} w_{ij}$
- $\pi = \frac{\mathbf{1}^\top \mathbf{W}}{\mathrm{vol}(\mathbf{W})}$ verifies $\pi\mathbf{P} = \pi$ as:

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)
- if G is connected and non-bipartite $\to$ unique **stationary distribution** $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ where $\pi_i = d_i/\mathrm{vol}(V)$
  - $\mathrm{vol}(G) = \mathrm{vol}(V) = \mathrm{vol}(\mathbf{W}) \stackrel{\text{def}}{=} \sum_i d_i = \sum_{i,j} w_{ij}$
- $\pi = \frac{\mathbf{1}^\mathsf{T}\mathbf{W}}{\mathrm{vol}(\mathbf{W})}$ verifies $\pi\mathbf{P} = \pi$ as:

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)
- if G is connected and non-bipartite $\to$ unique **stationary distribution** $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ where $\pi_i = d_i/\mathrm{vol}(V)$
  - $\mathrm{vol}(G) = \mathrm{vol}(V) = \mathrm{vol}(\mathbf{W}) \stackrel{\text{def}}{=} \sum_i d_i = \sum_{i,j} w_{ij}$
- $\pi = \frac{\mathbf{1}^\intercal \mathbf{W}}{\mathrm{vol}(\mathbf{W})}$ verifies $\pi\mathbf{P} = \pi$ as:

$$\pi\mathbf{P} = \frac{\mathbf{1}^\intercal\mathbf{W}\mathbf{P}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\intercal\mathbf{D}\mathbf{P}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\intercal\mathbf{D}\mathbf{D}^{-1}\mathbf{W}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\intercal\mathbf{W}}{\mathrm{vol}(\mathbf{W})} = \pi$$

Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)
- if G is connected and non-bipartite $\to$ unique **stationary distribution** $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ where $\pi_i = d_i/\mathrm{vol}(V)$
  - $\mathrm{vol}(G) = \mathrm{vol}(V) = \mathrm{vol}(\mathbf{W}) \stackrel{\text{def}}{=} \sum_i d_i = \sum_{i,j} w_{ij}$
- $\pi = \frac{\mathbf{1}^\intercal \mathbf{W}}{\mathrm{vol}(\mathbf{W})}$ verifies $\pi\mathbf{P} = \pi$ as:

$$\pi\mathbf{P} = \frac{\mathbf{1}^\intercal\mathbf{W}\mathbf{P}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\intercal\mathbf{D}\mathbf{P}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\intercal\mathbf{D}\mathbf{D}^{-1}\mathbf{W}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\intercal\mathbf{W}}{\mathrm{vol}(\mathbf{W})} = \pi$$

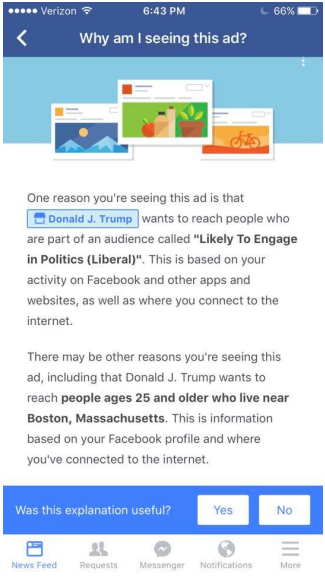
What's the difference from the PageRank™?

$$\mathrm{cut}(A, B) = \frac{1}{2}\mathbf{f}^\top \mathbf{L}\mathbf{f}$$

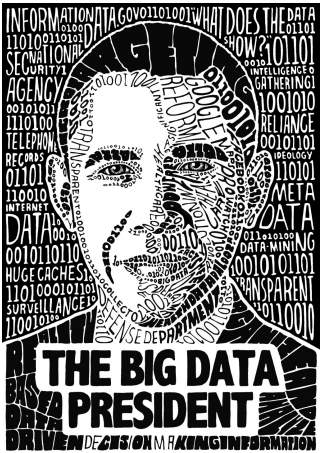
spectral clustering

...with connectivity beyond compactness

How to rule the world?



How to rule the world: "AI" is here

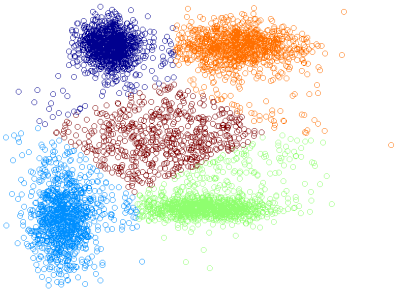


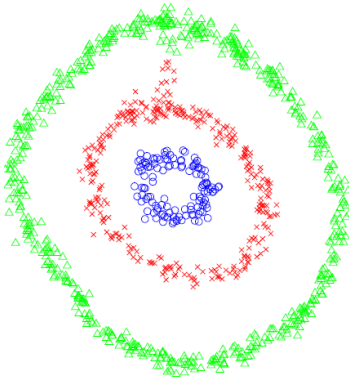
https://www.washingtonpost.com/opinions/obama-the-big-data-president/2013/06/14/1d71fe2e-d391-11e2-b05f-3ea3f0e7bb5a_story.html

https://www.technologyreview.com/s/509026/how-obamas-team-used-big-data-to-rally-voters/

Talk of Rayid Ghaniy: https://www.youtube.com/watch?v=gDM1GuszM_U

Application of Graphs for ML: Clustering





Application: Clustering - Recap

- What do we know about the **clustering** in general?

Application: Clustering - Recap

- What do we know about the **clustering** in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)

Application: Clustering - Recap

- What do we know about the clustering in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"

Application: Clustering - Recap

- What do we know about the clustering in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency

Application: Clustering - Recap

- What do we know about the clustering in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency
  - number of clusters k need often be known

Application: Clustering - Recap

- What do we know about the clustering in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency
  - number of clusters k need often be known
  - difficult to evaluate

Application: Clustering - Recap

- What do we know about the **clustering** in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency
  - number of clusters k need often be known
  - difficult to evaluate

- What do we know about k-**means**?

Application: Clustering - Recap

- What do we know about the **clustering** in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency
  - number of clusters k need often be known
  - difficult to evaluate

- What do we know about k-**means**?
  - "hard" version of EM clustering

Application: Clustering - Recap

- What do we know about the **clustering** in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency
  - number of clusters k need often be known
  - difficult to evaluate

- What do we know about k-**means**?
  - "hard" version of EM clustering
  - sensitive to initialization

Application: Clustering - Recap

- What do we know about the **clustering** in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency
  - number of clusters k need often be known
  - difficult to evaluate

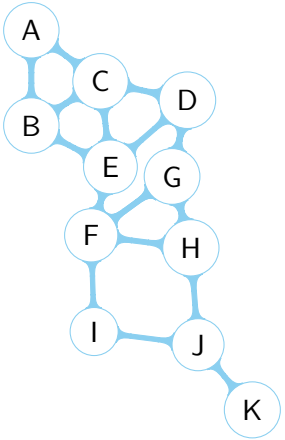
- What do we know about k-**means**?
  - "hard" version of EM clustering
  - sensitive to initialization
  - optimizes for **compactness**

Application: Clustering - Recap

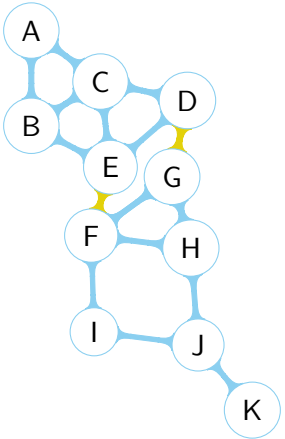
- What do we know about the **clustering** in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)
    - scale-invariance, richness, consistency
  - number of clusters k need often be known
  - difficult to evaluate

- What do we know about k-**means**?
  - "hard" version of EM clustering
  - sensitive to initialization
  - optimizes for **compactness**
  - yet: algorithm-to-go

Spectral Clustering: Cuts on graphs

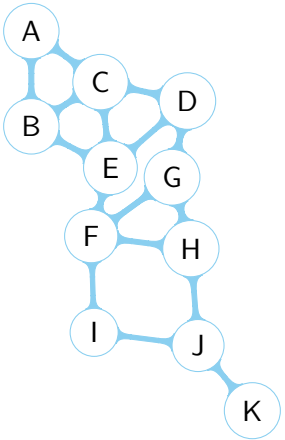


Spectral Clustering: Cuts on graphs



Defining the cut objective we get the clustering!

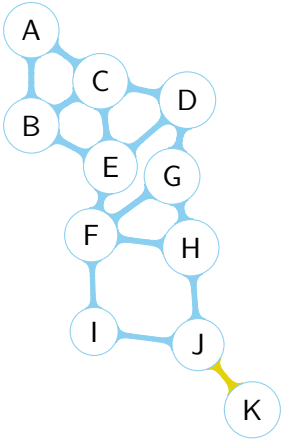
Spectral Clustering: Cuts on graphs



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

Are we done?

Spectral Clustering: Cuts on graphs



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

Are we done?

Can be solved efficiently, but maybe not what we want . . . .

Spectral Clustering: Balanced Cuts

Let's balance the cuts!

Spectral Clustering: Balanced Cuts

Let's balance the cuts!

MinCut

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

Spectral Clustering: Balanced Cuts

Let's balance the cuts!

MinCut

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

RatioCut

$$\text{RatioCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Spectral Clustering: Balanced Cuts

Let's balance the cuts!

MinCut

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

RatioCut

$$\mathrm{RatioCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Normalized Cut

$$\mathrm{NCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right)$$

Spectral Clustering: Balanced Cuts

$$\mathrm{RatioCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$$

$$\mathrm{NCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$

Easily generalizable to $k \geq 2$

Spectral Clustering: Balanced Cuts

$$\mathrm{RatioCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$$

$$\mathrm{NCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$

Easily generalizable to $k \geq 2$

Can we compute this?

Spectral Clustering: Balanced Cuts

$$\mathrm{RatioCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$$

$$\mathrm{NCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$

Easily generalizable to $k \geq 2$

Can we compute this? RatioCut and NCut are NP hard :(

Spectral Clustering: Balanced Cuts

$$\mathrm{RatioCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$$

$$\mathrm{NCut}(A, B) = \mathrm{cut}(A, B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$

Easily generalizable to $k \geq 2$

Can we compute this? RatioCut and NCut are NP hard :(Approximate!

Spectral Clustering: Relaxing Balanced Cuts

Relaxation for (simple) balanced cuts for 2 sets

$$\min_{A,B} \operatorname{cut}(A, B) \quad \text{s.t.} \quad |A| = |B|$$

Spectral Clustering: Relaxing Balanced Cuts

Relaxation for (simple) balanced cuts for 2 sets

$$\min_{A,B} \operatorname{cut}(A, B) \quad \text{s.t.} \quad |A| = |B|$$

Graph function **f** for cluster membership

Spectral Clustering: Relaxing Balanced Cuts

Relaxation for (simple) balanced cuts for 2 sets

$$\min_{A,B} \operatorname{cut}(A, B) \quad \text{s.t.} \quad |A| = |B|$$

Graph function **f** for cluster membership: $f_i = \begin{cases} 1 & \text{if } V_i \in A, \\ -1 & \text{if } V_i \in B. \end{cases}$

Spectral Clustering: Relaxing Balanced Cuts

Relaxation for (simple) balanced cuts for 2 sets

$$\min_{A,B} \mathrm{cut}(A,B) \quad \text{s.t.} \quad |A| = |B|$$

Graph function **f** for cluster membership: $f_i = \begin{cases} 1 & \text{if } V_i \in A, \\ -1 & \text{if } V_i \in B. \end{cases}$

What it is the cut value with this definition?

$$\mathrm{cut}(A,B) =$$

Spectral Clustering: Relaxing Balanced Cuts

Relaxation for (simple) balanced cuts for 2 sets

$$\min_{A,B} \operatorname{cut}(A, B) \quad \text{s.t.} \quad |A| = |B|$$

Graph function $\mathbf{f}$ for cluster membership: $f_i = \begin{cases} 1 & \text{if } V_i \in A, \\ -1 & \text{if } V_i \in B. \end{cases}$

What it is the cut value with this definition?

$$\operatorname{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\top \mathbf{L} \mathbf{f}$$

Spectral Clustering: Relaxing Balanced Cuts

Relaxation for (simple) balanced cuts for 2 sets

$$\min_{A,B} \operatorname{cut}(A,B) \quad \text{s.t.} \quad |A| = |B|$$

Graph function $\mathbf{f}$ for cluster membership: $f_i = \begin{cases} 1 & \text{if } V_i \in A, \\ -1 & \text{if } V_i \in B. \end{cases}$

What it is the cut value with this definition?

$$\operatorname{cut}(A,B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j}(f_i - f_j)^2 = \frac{1}{2}\mathbf{f}^\top \mathbf{L}\mathbf{f}$$

What is the relationship with the **smoothness** of a graph function?

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\intercal \mathbf{L} \mathbf{f}$$

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\intercal \mathbf{L} \mathbf{f}$$

$|A| = |B| \implies \sum_i f_i =$

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\intercal \mathbf{L} \mathbf{f}$$

$|A| = |B| \implies \sum_i f_i = 0 \implies$

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\intercal \mathbf{L} \mathbf{f}$$

$|A| = |B| \implies \sum_i f_i = 0 \implies \mathbf{f} \perp \mathbf{1}_N$

$\|\mathbf{f}\| =$

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\intercal \mathbf{L} \mathbf{f}$$

$$|A| = |B| \implies \sum_i f_i = 0 \implies \mathbf{f} \perp \mathbf{1}_N$$

$$\|\mathbf{f}\| = \sqrt{N}$$

Spectral Clustering: Relaxing Balanced Cuts

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\top \mathbf{L} \mathbf{f}$$

$$|A| = |B| \implies \sum_i f_i = 0 \implies \mathbf{f} \perp \mathbf{1}_N$$

$$\|\mathbf{f}\| = \sqrt{N}$$

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L} \mathbf{f} \quad \text{s.t.} \quad f_i = \pm 1, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\top \mathbf{L} \mathbf{f}$$

$$|A| = |B| \implies \sum_i f_i = 0 \implies \mathbf{f} \perp \mathbf{1}_N$$

$$\|\mathbf{f}\| = \sqrt{N}$$

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L} \mathbf{f} \quad \text{s.t.} \quad f_i = \pm 1, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Still NP hard :(

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\top \mathbf{L} \mathbf{f}$$

$$|A| = |B| \implies \sum_i f_i = 0 \implies \mathbf{f} \perp \mathbf{1}_N$$

$$\|\mathbf{f}\| = \sqrt{N}$$

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L} \mathbf{f} \quad \text{s.t.} \quad f_i = \pm 1, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Still NP hard :($\rightarrow$ Relax even further!

Spectral Clustering: Relaxing Balanced Cuts

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j} = \frac{1}{4} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = \frac{1}{2} \mathbf{f}^\top \mathbf{L} \mathbf{f}$$

$$|A| = |B| \implies \sum_i f_i = 0 \implies \mathbf{f} \perp \mathbf{1}_N$$

$$\|\mathbf{f}\| = \sqrt{N}$$

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L} \mathbf{f} \quad \text{s.t.} \quad f_i = \pm 1, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Still NP hard :($\rightarrow$ Relax even further!

~~$f_i = \pm 1$~~ $\rightarrow$ $f_i \in \mathbb{R}$

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Rayleigh-Ritz theorem

If $\lambda_1 \leq \cdots \leq \lambda_N$ are the eigenvalues of real symmetric $\mathbf{L}$ then

$$\lambda_1 = \min_{\mathbf{x} \neq 0} \frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \min_{\mathbf{x}^\top \mathbf{x} = 1} \mathbf{x}^\top \mathbf{L}\mathbf{x}$$

$$\lambda_N = \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \max_{\mathbf{x}^\top \mathbf{x} = 1} \mathbf{x}^\top \mathbf{L}\mathbf{x}$$

$\frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \equiv$ **Rayleigh quotient**

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Rayleigh-Ritz theorem

If $\lambda_1 \leq \cdots \leq \lambda_N$ are the eigenvalues of real symmetric $\mathbf{L}$ then

$$\lambda_1 = \min_{\mathbf{x} \neq 0} \frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \min_{\mathbf{x}^\top \mathbf{x} = 1} \mathbf{x}^\top \mathbf{L}\mathbf{x}$$

$$\lambda_N = \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \max_{\mathbf{x}^\top \mathbf{x} = 1} \mathbf{x}^\top \mathbf{L}\mathbf{x}$$

$\frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \equiv$ **Rayleigh quotient**

How can we use it?

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Generalized Rayleigh-Ritz theorem (Courant-Fischer-Weyl)

If $\lambda_1 \leq \cdots \leq \lambda_N$ are the eigenvalues of real symmetric $\mathbf{L}$ and $\mathbf{v}_1, \ldots, \mathbf{v}_N$ the corresponding orthogonal eigenvectors, then for $k = 1 : N-1$

$$\lambda_{k+1} = \min_{\mathbf{x} \neq 0, \mathbf{x} \perp \mathbf{v}_1, \ldots \mathbf{v}_k} \frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \min_{\mathbf{x}^\top \mathbf{x} = 1, \mathbf{x} \perp \mathbf{v}_1, \ldots \mathbf{v}_k} \mathbf{x}^\top \mathbf{L}\mathbf{x}$$

$$\lambda_{N-k} = \max_{\mathbf{x} \neq 0, \mathbf{x} \perp \mathbf{v}_n, \ldots \mathbf{v}_{N-k+1}} \frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \max_{\mathbf{x}^\top \mathbf{x} = 1, \mathbf{x} \perp \mathbf{v}_N, \ldots \mathbf{v}_{N-k+1}} \mathbf{x}^\top \mathbf{L}\mathbf{x}$$

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Solution:

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Solution: **second eigenvector** How do we get the clustering?

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Solution: **second eigenvector** How do we get the clustering?

The solution may not be integral. What to do?

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Solution: **second eigenvector** How do we get the clustering?

The solution may not be integral. What to do?

$$\text{cluster}_i = \begin{cases} 1 & \text{if } f_i \geq 0, \\ -1 & \text{if } f_i < 0. \end{cases}$$

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Solution: **second eigenvector** How do we get the clustering?

The solution may not be integral. What to do?

$$\text{cluster}_i = \begin{cases} 1 & \text{if } f_i \geq 0, \\ -1 & \text{if } f_i < 0. \end{cases}$$

Works but this heuristics is often too simple.

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Solution: **second eigenvector** How do we get the clustering?

The solution may not be integral. What to do?

$$\text{cluster}_i = \begin{cases} 1 & \text{if } f_i \geq 0, \\ -1 & \text{if } f_i < 0. \end{cases}$$

Works but this heuristics is often too simple. In practice, cluster $\mathbf{f}$ using k-means to get $\{C_i\}_i$ and assign:

Spectral Clustering: Relaxing Balanced Cuts

objective function of spectral clustering

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Solution: **second eigenvector** How do we get the clustering?

The solution may not be integral. What to do?

$$\text{cluster}_i = \begin{cases} 1 & \text{if } f_i \geq 0, \\ -1 & \text{if } f_i < 0. \end{cases}$$

Works but this heuristics is often too simple. In practice, cluster $\mathbf{f}$ using k-means to get $\{C_i\}_i$ and assign:

$$\text{cluster}_i = \begin{cases} 1 & \text{if } i \in C_1, \\ -1 & \text{if } i \in C_{-1}. \end{cases}$$

Spectral Clustering: Approximating RatioCut

Wait, but we did not care about approximating mincut!

Spectral Clustering: Approximating RatioCut

Wait, but we did not care about approximating mincut!

RatioCut

$$\mathrm{RatioCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Spectral Clustering: Approximating RatioCut

Wait, but we did not care about approximating mincut!

RatioCut

$$\mathrm{RatioCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of RatioCut

Spectral Clustering: Approximating RatioCut

Wait, but we did not care about approximating mincut!

RatioCut

$$\mathrm{RatioCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

Spectral Clustering: Approximating RatioCut

Wait, but we did not care about approximating mincut!

RatioCut

$$\mathrm{RatioCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

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

Spectral Clustering: Approximating RatioCut

Wait, but we did not care about approximating mincut!

RatioCut

$$\mathrm{RatioCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

$$\mathbf{f}^T \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j} w_{i,j} (f_i - f_j)^2 = (|A| + |B|) \mathrm{RatioCut}(A, B)$$

Spectral Clustering: Approximating RatioCut

Define graph function $\mathbf{f}$ for cluster membership of RatioCut

Spectral Clustering: Approximating RatioCut

Define graph function **f** for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

Spectral Clustering: Approximating RatioCut

Define graph function **f** for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

$$\sum_i f_i =$$

Spectral Clustering: Approximating RatioCut

Define graph function **f** for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

$$\sum_i f_i = 0$$

$$\sum_i f_i^2 =$$

Spectral Clustering: Approximating RatioCut

Define graph function **f** for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

$$\sum_i f_i = 0$$

$$\sum_i f_i^2 = N$$

Spectral Clustering: Approximating RatioCut

Define graph function $\mathbf{f}$ for cluster membership of RatioCut:

$$f_i = \begin{cases} \sqrt{\frac{|B|}{|A|}} & \text{if } V_i \in A, \\ -\sqrt{\frac{|A|}{|B|}} & \text{if } V_i \in B. \end{cases}$$

$$\sum_i f_i = 0$$

$$\sum_i f_i^2 = N$$

objective function of spectral clustering (same - it's magic!)

$$\min_{\mathbf{f}} \mathbf{f}^T \mathbf{L} \mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f} \perp \mathbf{1}_N, \quad \|\mathbf{f}\| = \sqrt{N}$$

Spectral Clustering: Approximating NCut

Normalized Cut

$$\mathrm{NCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right)$$

Spectral Clustering: Approximating NCut

Normalized Cut

$$\mathrm{NCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right)$$

Define graph function **f** for cluster membership of NCut:

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(B)}{\mathrm{vol}(A)}} & \text{if } V_i \in A, \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(B)}} & \text{if } V_i \in B. \end{cases}$$

Spectral Clustering: Approximating NCut

Normalized Cut

$$\mathrm{NCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of NCut:

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(B)}{\mathrm{vol}(A)}} & \text{if } V_i \in A, \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(B)}} & \text{if } V_i \in B. \end{cases}$$

$$(\mathbf{Df})^\top \mathbf{1}_n = 0$$

Spectral Clustering: Approximating NCut

Normalized Cut

$$\mathrm{NCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of NCut:

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(B)}{\mathrm{vol}(A)}} & \text{if } V_i \in A, \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(B)}} & \text{if } V_i \in B. \end{cases}$$

$$(\mathbf{Df})^\mathsf{T} \mathbf{1}_n = 0 \qquad \mathbf{f}^\mathsf{T} \mathbf{Df} = \mathrm{vol}(\mathcal{V})$$

Spectral Clustering: Approximating NCut

Normalized Cut

$$\mathrm{NCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of NCut:

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(B)}{\mathrm{vol}(A)}} & \text{if } V_i \in A, \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(B)}} & \text{if } V_i \in B. \end{cases}$$

$$(\mathbf{Df})^\mathsf{T} \mathbf{1}_n = 0 \qquad \mathbf{f}^\mathsf{T} \mathbf{Df} = \mathrm{vol}(\mathcal{V}) \qquad \mathbf{f}^\mathsf{T} \mathbf{Lf} = \mathrm{vol}(\mathcal{V}) \mathrm{NCut}(A, B)$$

Spectral Clustering: Approximating NCut

Normalized Cut

$$\mathrm{NCut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right)$$

Define graph function $\mathbf{f}$ for cluster membership of NCut:

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(B)}{\mathrm{vol}(A)}} & \text{if } V_i \in A, \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(B)}} & \text{if } V_i \in B. \end{cases}$$

$$(\mathbf{Df})^\mathsf{T} \mathbf{1}_n = 0 \qquad \mathbf{f}^\mathsf{T} \mathbf{Df} = \mathrm{vol}(\mathcal{V}) \qquad \mathbf{f}^\mathsf{T} \mathbf{Lf} = \mathrm{vol}(\mathcal{V}) \mathrm{NCut}(A, B)$$

objective function of spectral clustering (NCut)

$$\min_{\mathbf{f}} \mathbf{f}^\mathsf{T} \mathbf{Lf} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{Df} \perp \mathbf{1}_N, \quad \mathbf{f}^\mathsf{T} \mathbf{Df} = \mathrm{vol}(\mathcal{V})$$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{D}\mathbf{f} \perp \mathbf{1}_N, \quad \mathbf{f}^\top \mathbf{D}\mathbf{f} = \text{vol}(\mathcal{V})$$

Can we apply Rayleigh-Ritz now?

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{f}} \mathbf{f}^\intercal \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{D}\mathbf{f} \perp \mathbf{1}_N, \quad \mathbf{f}^\intercal \mathbf{D}\mathbf{f} = \text{vol}(\mathcal{V})$$

Can we apply Rayleigh-Ritz now? Define $\mathbf{w} = \mathbf{D}^{1/2}\mathbf{f}$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{D}\mathbf{f} \perp \mathbf{1}_N, \quad \mathbf{f}^\top \mathbf{D}\mathbf{f} = \text{vol}(\mathcal{V})$$

Can we apply Rayleigh-Ritz now? Define $\mathbf{w} = \mathbf{D}^{1/2}\mathbf{f}$

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\top \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \mathbf{w} \perp \mathbf{D}^{1/2}\mathbf{1}_N, \|\mathbf{w}\|^2 = \text{vol}(\mathcal{V})$$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{D}\mathbf{f} \perp \mathbf{1}_N, \quad \mathbf{f}^\top \mathbf{D}\mathbf{f} = \text{vol}(\mathcal{V})$$

Can we apply Rayleigh-Ritz now? Define $\mathbf{w} = \mathbf{D}^{1/2}\mathbf{f}$

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\top \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \mathbf{w} \perp \mathbf{D}^{1/2}\mathbf{1}_N, \|\mathbf{w}\|^2 = \text{vol}(\mathcal{V})$$

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\top \mathbf{L}_{\text{sym}}\mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\|^2 = \text{vol}(\mathcal{V})$$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Solution by Rayleigh-Ritz?

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Solution by Rayleigh-Ritz? $\mathbf{w} = \mathbf{v}_{2,\mathbf{L}_{\text{sym}}}$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Solution by Rayleigh-Ritz? $\mathbf{w} = \mathbf{v}_{2,\mathbf{L}_{\text{sym}}}$ $\mathbf{f} =$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Solution by Rayleigh-Ritz? $\mathbf{w} = \mathbf{v}_{2,\mathbf{L}_{\text{sym}}}$ $\mathbf{f} = \mathbf{D}^{-1/2}\mathbf{w}$

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Solution by Rayleigh-Ritz? $\mathbf{w} = \mathbf{v}_{2,\mathbf{L}_{\text{sym}}}$ $\mathbf{f} = \mathbf{D}^{-1/2}\mathbf{w}$

$\mathbf{f}$ is a the second eigenvector of $\mathbf{L}_{\text{rw}}$!

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Solution by Rayleigh-Ritz? $\mathbf{w} = \mathbf{v}_{2,\mathbf{L}_{\text{sym}}}$ $\mathbf{f} = \mathbf{D}^{-1/2}\mathbf{w}$

$\mathbf{f}$ is a the second eigenvector of $\mathbf{L}_{\text{rw}}$!

tl;dr: Get the second eigenvector of $\mathbf{L}/\mathbf{L}_{\text{rw}}$ for RatioCut/NCut.

Spectral Clustering: Approximating NCut

objective function of spectral clustering (NCut)

$$\min_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{L}_{\text{sym}} \mathbf{w} \quad \text{s.t.} \quad w_i \in \mathbb{R}, \quad \mathbf{w} \perp \mathbf{v}_{1,\mathbf{L}_{\text{sym}}}, \quad \|\mathbf{w}\| = \text{vol}(\mathcal{V})$$

Solution by Rayleigh-Ritz? $\mathbf{w} = \mathbf{v}_{2,\mathbf{L}_{\text{sym}}}$ $\mathbf{f} = \mathbf{D}^{-1/2}\mathbf{w}$

$\mathbf{f}$ is a the second eigenvector of $\mathbf{L}_{\text{rw}}$!

tl;dr: Get the second eigenvector of $\mathbf{L}/\mathbf{L}_{\text{rw}}$ for RatioCut/NCut.

demo: `https://dominikschmidt.xyz/spectral-clustering-exp/`

Spectral Clustering: Approximation

These are all approximations. How bad can they be?

Spectral Clustering: Approximation

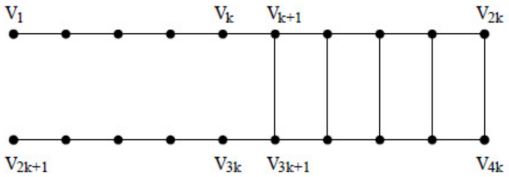
These are all approximations. How bad can they be?

Pretty bad.

Spectral Clustering: Approximation

These are all approximations. How bad can they be?

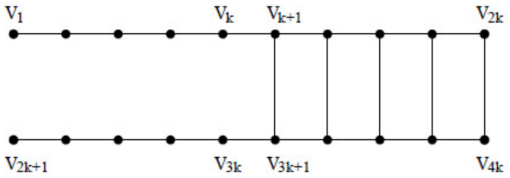
Pretty bad. Example: cockroach graphs



Spectral Clustering: Approximation

These are all approximations. How bad can they be?

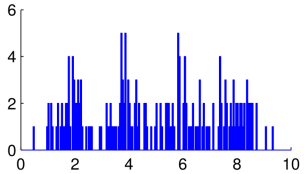
Pretty bad. Example: cockroach graphs



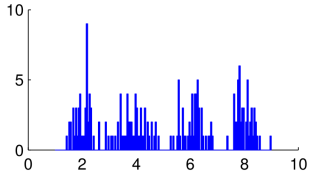
No efficient approximation exist. Other relaxations possible.

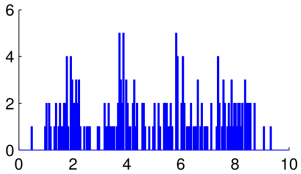
https://www.cs.cmu.edu/~glmiller/Publications/Papers/GuMi95.pdf

Spectral Clustering: 1D Example

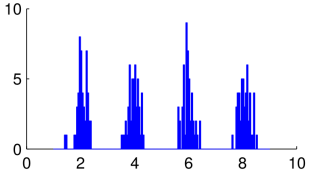


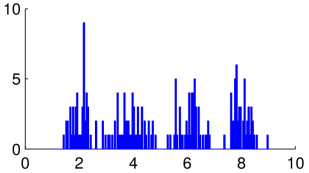
Spectral Clustering: 1D Example

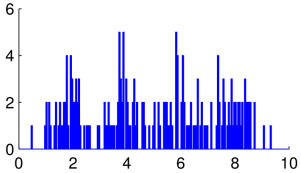




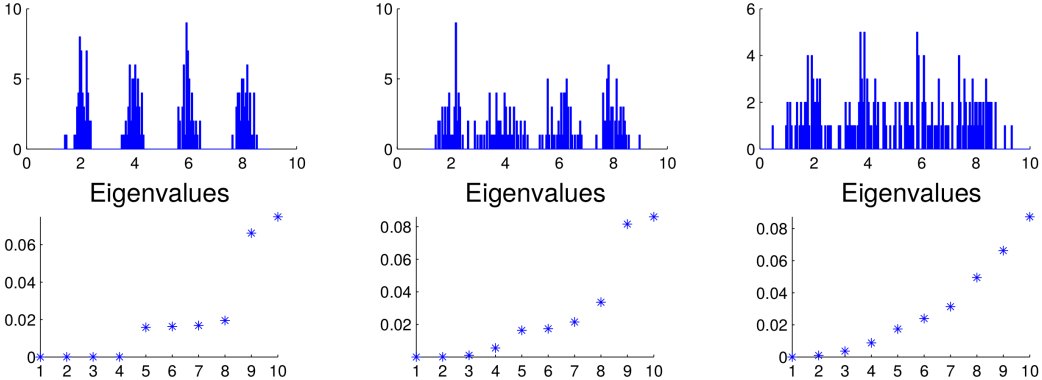
Spectral Clustering: 1D Example





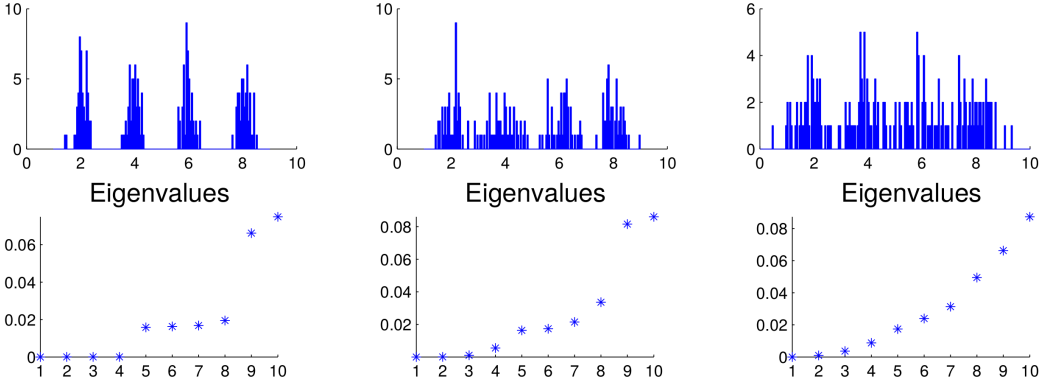


Spectral Clustering: 1D Example



Spectral Clustering: 1D Example

Elbow rule/EigenGap heuristic for number of clusters

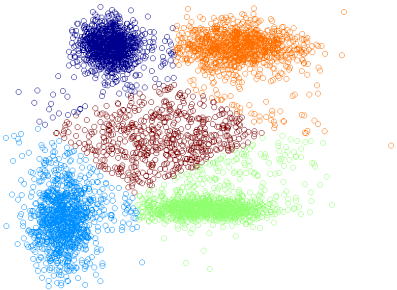


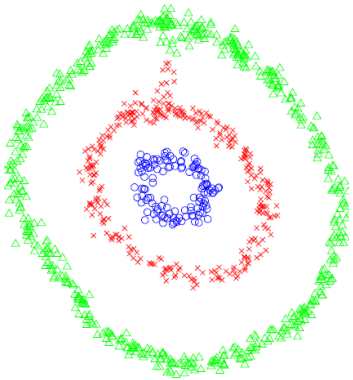
Spectral Clustering: Understanding

Compactness vs. **Connectivity**

Spectral Clustering: Understanding

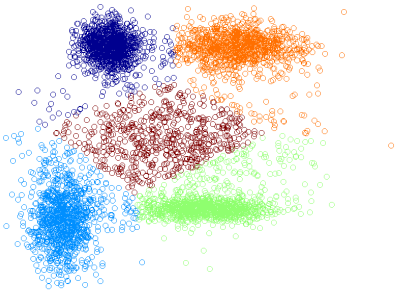
Compactness vs. **Connectivity**

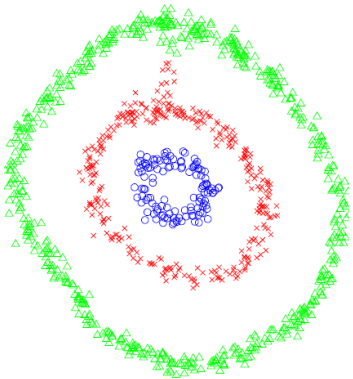




Spectral Clustering: Understanding

Compactness vs. **Connectivity**

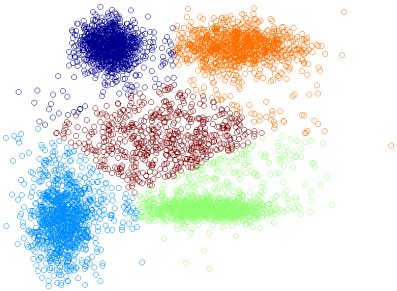


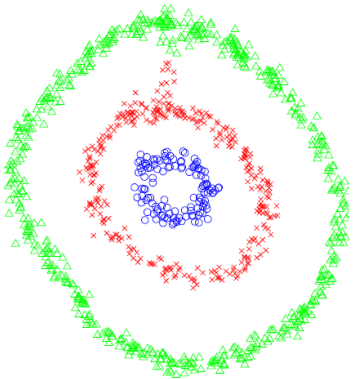


For which kind of data we can use one vs. the other?

Spectral Clustering: Understanding

Compactness vs. **Connectivity**

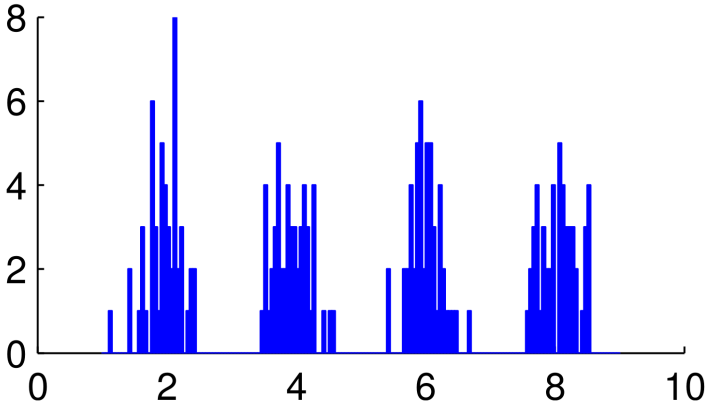




For which kind of data we can use one vs. the other?

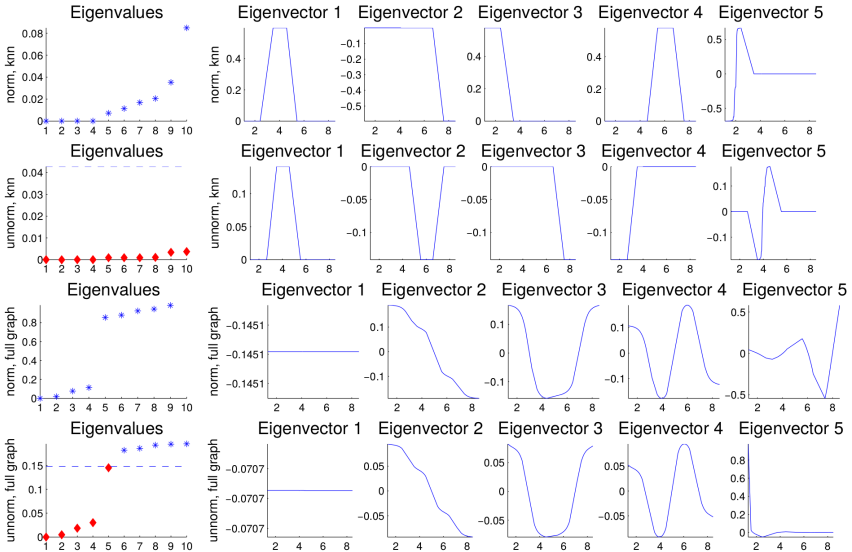
Any disadvantages of spectral clustering?

Spectral Clustering: 1D Example - Histogram



http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/publications/Luxburg07_tutorial.pdf

Spectral Clustering: 1D Example - Eigenvectors



Spectral Clustering: Bibliography

- M. Meila et al. "A random walks view of spectral segmentation". In: *International Conference on Artificial Intelligence and Statistics* (2001)
- $\mathbf{L}_{\text{sym}}$ Andrew Y Ng, Michael I Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm". In: *Neural Information Processing Systems*. 2001
- $\mathbf{L}_{\text{rm}}$ J Shi and J Malik. "Normalized Cuts and Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), pp. 888–905
- Things can go wrong with the relaxation: Daniel A. Spielman and Shang H. Teng. "Spectral partitioning works: Planar graphs and finite element meshes". In: *Linear Algebra and Its Applications* 421 (2007), pp. 284–305

$$\mathbb{R}^d \longrightarrow \mathbb{R}^m$$

manifold learning

...discworld

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^{N}$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^{N}$ in $\mathbb{R}^m$, where $m \ll d$.

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map
  - often assuming $\mathcal{M} \subset \mathbb{R}^d$

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map
  - often assuming $\mathcal{M} \subset \mathbb{R}^d$
  - feature extraction

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map
  - often assuming $\mathcal{M} \subset \mathbb{R}^d$
  - feature extraction
  - linear vs. nonlinear dimensionality reduction

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map
  - often assuming $\mathcal{M} \subset \mathbb{R}^d$
  - feature extraction
  - linear vs. nonlinear dimensionality reduction
- What do we know about linear vs. nonlinear methods?

Manifold Learning: Recap

problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map
  - often assuming $\mathcal{M} \subset \mathbb{R}^d$
  - feature extraction
  - linear vs. nonlinear dimensionality reduction
- What do we know about linear vs. nonlinear methods?
  - linear: ICA, PCA, SVD, ...

Manifold Learning: Recap

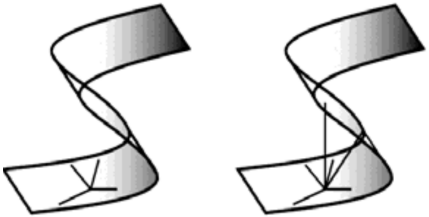
problem: definition reduction/manifold learning

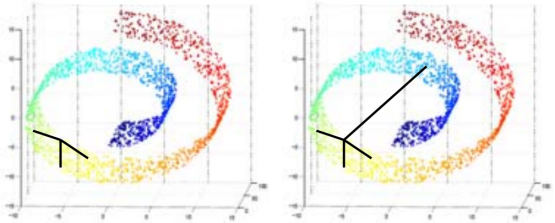
Given $\{\mathbf{x}_i\}_{i=1}^N$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^N$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map
  - often assuming $\mathcal{M} \subset \mathbb{R}^d$
  - feature extraction
  - linear vs. nonlinear dimensionality reduction
- What do we know about linear vs. nonlinear methods?
  - linear: ICA, PCA, SVD, ...
  - nonlinear often preserve only **local** distances

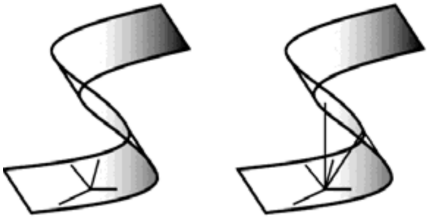
Manifold Learning: Linear vs. Non-linear

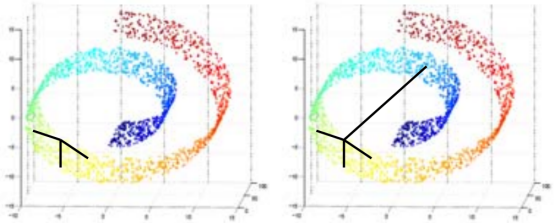
Manifold Learning: Preserving (just) local distances





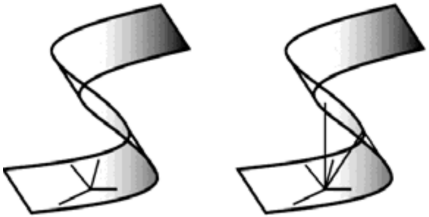
Manifold Learning: Preserving (just) local distances

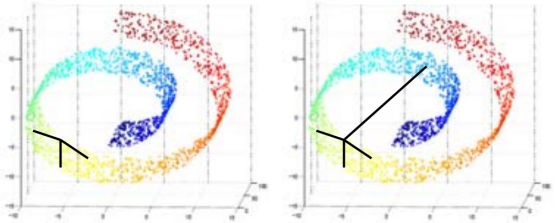




$$d(\mathbf{y}_i, \mathbf{y}_j) = d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{only if} \quad d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{is small}$$

Manifold Learning: Preserving (just) local distances

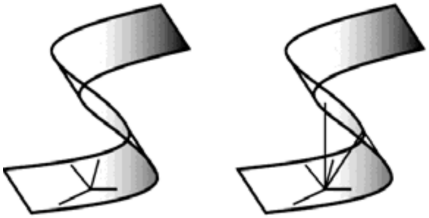


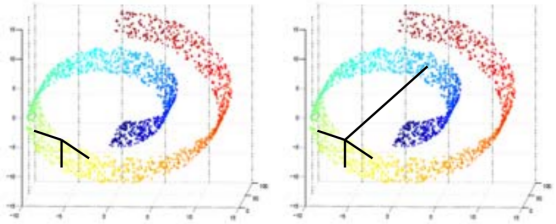


$$d(\mathbf{y}_i, \mathbf{y}_j) = d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{only if} \quad d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{is small}$$

$$\text{1-D} \qquad \min_y \sum_{ij} w_{ij}(y_i - y_j)^2$$

Manifold Learning: Preserving (just) local distances



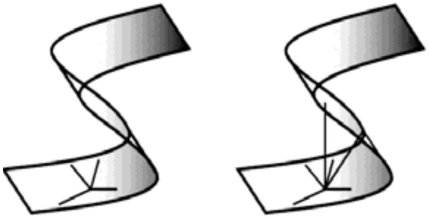


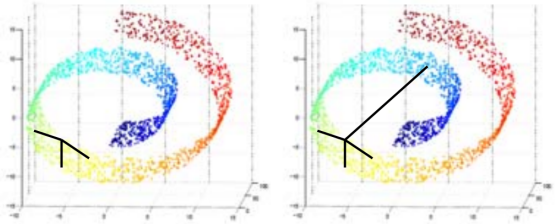
$$d(\mathbf{y}_i, \mathbf{y}_j) = d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{only if} \quad d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{is small}$$

$$\text{1-D} \qquad \min_{y} \sum_{ij} w_{ij}(y_i - y_j)^2$$

$$m\text{-D} \qquad \min_{\mathbf{y}} \sum_{ij} w_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2$$

Manifold Learning: Preserving (just) local distances





$$d(\mathbf{y}_i, \mathbf{y}_j) = d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{only if} \quad d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{is small}$$

$$\text{1-D} \qquad \min_{y} \sum_{ij} w_{ij}(y_i - y_j)^2$$

$$m\text{-D} \qquad \min_{\mathbf{y}} \sum_{ij} w_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2$$

Looks familiar?

Manifold Learning: Laplacian Eigenmaps

Step 1: Solve generalized eigenproblem:

$$\mathbf{Lf} = \lambda \mathbf{Df}$$

Manifold Learning: Laplacian Eigenmaps

Step 1: Solve generalized eigenproblem:

$$\mathbf{L}\mathbf{f} = \lambda \mathbf{D}\mathbf{f}$$

Step 2: Assign m new coordinates:

$$\mathbf{x}_i \mapsto \left(f_2\left(i\right), \ldots, f_{m+1}\left(i\right)\right)$$

Manifold Learning: Laplacian Eigenmaps

Step 1: Solve generalized eigenproblem:

$$\mathbf{L}\mathbf{f} = \lambda \mathbf{D}\mathbf{f}$$

Step 2: Assign m new coordinates:

$$\mathbf{x}_i \mapsto \left(f_2\left(i\right), \ldots, f_{m+1}\left(i\right)\right)$$

Note$_1$: we need to get $m+1$ smallest eigenvectors

Manifold Learning: Laplacian Eigenmaps

Step 1: Solve generalized eigenproblem:

$$\mathbf{Lf} = \lambda \mathbf{Df}$$

Step 2: Assign m new coordinates:

$$\mathbf{x}_i \mapsto \left(f_2\left(i\right), \ldots, f_{m+1}\left(i\right)\right)$$

Note$_1$: we need to get $m+1$ smallest eigenvectors
Note$_2$: $\mathbf{f}_1$ is useless

`http://web.cse.ohio-state.edu/~mbelkin/papers/LEM_NC_03.pdf`

Manifold Learning: Laplacian Eigenmaps to 1D

Laplacian Eigenmaps 1D objective

$$\min_{\mathbf{f}} \mathbf{f}^T \mathbf{L} \mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f}^T \mathbf{D} \mathbf{1} = 0, \quad \mathbf{f}^T \mathbf{D} \mathbf{f} = \mathbf{1}$$

Manifold Learning: Laplacian Eigenmaps to 1D

Laplacian Eigenmaps 1D objective

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f}^\top \mathbf{D}\mathbf{1} = 0, \quad \mathbf{f}^\top \mathbf{D}\mathbf{f} = \mathbf{1}$$

The meaning of the constraints is similar as for spectral clustering:

$\mathbf{f}^\top \mathbf{D}\mathbf{f} = \mathbf{1}$ is for

Manifold Learning: Laplacian Eigenmaps to 1D

Laplacian Eigenmaps 1D objective

$$\min_{\mathbf{f}} \mathbf{f}^T \mathbf{L} \mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f}^T \mathbf{D} \mathbf{1} = 0, \quad \mathbf{f}^T \mathbf{D} \mathbf{f} = \mathbf{1}$$

The meaning of the constraints is similar as for spectral clustering:

$\mathbf{f}^T \mathbf{D} \mathbf{f} = \mathbf{1}$ is for scaling

Manifold Learning: Laplacian Eigenmaps to 1D

Laplacian Eigenmaps 1D objective

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f}^\top \mathbf{D}\mathbf{1} = 0, \quad \mathbf{f}^\top \mathbf{D}\mathbf{f} = \mathbf{1}$$

The meaning of the constraints is similar as for spectral clustering:

$\mathbf{f}^\top \mathbf{D}\mathbf{f} = \mathbf{1}$ is for scaling

$\mathbf{f}^\top \mathbf{D}\mathbf{1} = 0$ is to

Manifold Learning: Laplacian Eigenmaps to 1D

Laplacian Eigenmaps 1D objective

$$\min_{\mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f}^\top \mathbf{D}\mathbf{1} = 0, \quad \mathbf{f}^\top \mathbf{D}\mathbf{f} = \mathbf{1}$$

The meaning of the constraints is similar as for spectral clustering:

$\mathbf{f}^\top \mathbf{D}\mathbf{f} = \mathbf{1}$ is for scaling

$\mathbf{f}^\top \mathbf{D}\mathbf{1} = 0$ is to not get $\mathbf{v}_1$

Manifold Learning: Laplacian Eigenmaps to 1D

Laplacian Eigenmaps 1D objective

$$\min_{\mathbf{f}} \mathbf{f}^T\mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f}^T\mathbf{D}\mathbf{1} = 0, \quad \mathbf{f}^T\mathbf{D}\mathbf{f} = \mathbf{1}$$

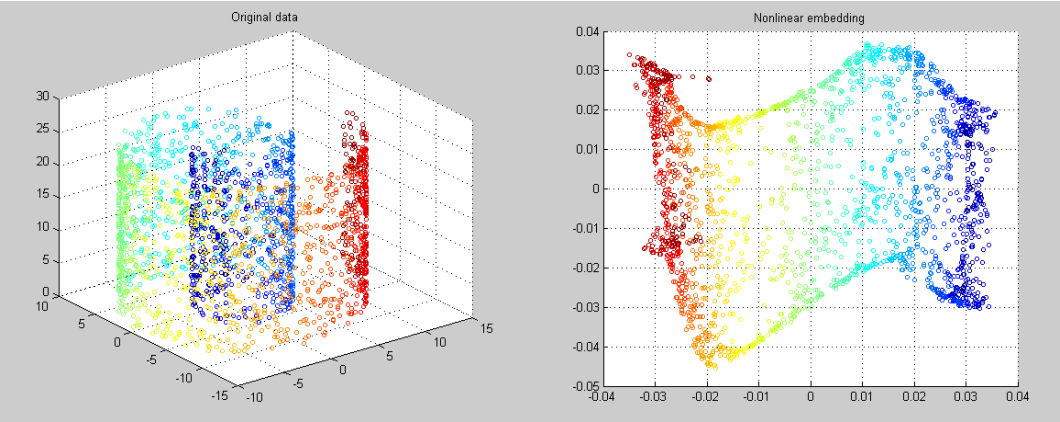
The meaning of the constraints is similar as for spectral clustering:

$\mathbf{f}^T\mathbf{D}\mathbf{f} = \mathbf{1}$ is for scaling

$\mathbf{f}^T\mathbf{D}\mathbf{1} = 0$ is to not get $\mathbf{v}_1$

What is the solution?

Manifold Learning: Example



http://www.mathworks.com/matlabcentral/fileexchange/

36141-laplacian-eigenmap--diffusion-map--manifold-learning

Daniele Calandriello
dcalandriello@google.com
ENS Paris-Saclay, MVA 2022/2023
https://sites.google.com/view/daniele-calandriello/