

Chapter 6

Spectral Methods

Part I: Spectral clustering



Roadmap

- In (partitioning) **clustering**, we want to partition the data points in clusters such that the points in the same cluster are maximally similar and the points in different clusters are maximally dissimilar
- In **spectral clustering**, we express the similarity structure with a graph, and model the clustering as a multi-way cut of the similarity graph

Background: Eigendecompositions

Eigenvectors and values

- Let $\mathbf{A} \in \mathbb{R}^{n \times n}$
 - $\mathbf{v} \in \mathbb{R}^n$ is an **eigenvector** of \mathbf{A} if $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$
 - If $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, λ is an **eigenvalue** associated to \mathbf{v}
 - If there are k eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$ s.t. $\mathbf{A}\mathbf{v}_i = \lambda\mathbf{v}_i$ for all i , then λ has (algebraic) **multiplicity** of k
 - n -by- n matrix has n eigenvectors and n eigenvalues (counting the multiplicity)
 - Some can be complex

Spectrum of a matrix

- The **characteristic polynomial** of $\mathbf{A} \in \mathbb{R}^{n \times n}$ is $p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I})$
- The roots of $p_{\mathbf{A}}(\lambda)$ are the eigenvalues of \mathbf{A}
 - I.e. $p_{\mathbf{A}}(\lambda) = 0 \Leftrightarrow \lambda$ is an eigenvalue of \mathbf{A}
- The collection of the eigenvalues of \mathbf{A} is called the **spectrum** of \mathbf{A}

Eigendecomposition

- The **eigendecomposition** of symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ is $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$
 - \mathbf{Q} is orthogonal and has the eigenvectors as its columns
 - $\mathbf{\Lambda}$ is diagonal with the eigenvalues
 - The symmetry of \mathbf{A} is sufficient but not necessary

Properties of eigendecomposition

- $\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T$
- $\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$
- If $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ then $\text{trace}(\mathbf{A}) = \text{trace}(\mathbf{\Lambda}) = \sum_i \lambda_i$
- The rank of \mathbf{A} is the number of non-zero eigenvalues

Positive semidefinite matrix

- Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is **positive semidefinite** if $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$ for any $\mathbf{x} \in \mathbb{R}^n$
 - \mathbf{A} is **positive definite** if the inequality is strict for any non-zero \mathbf{x}
- If $\mathbf{A} = \mathbf{B} \mathbf{B}^T$ for some $\mathbf{B} \in \mathbb{R}^{n \times m}$, \mathbf{A} is positive semidefinite
- If \mathbf{A} is positive semidefinite, all its eigenvalues are non-negative

Graphs and matrices

Fundamental matrices of graphs

- A graph $G = (V, E)$ can be represented by its **adjacency matrix A**
 - $a_{ij} = 1$ if $\{v_i, v_j\} \in E$, o/w 0
- Or by its **incidence matrix P**
 - $p_{ij} = 1$ if $e_j \in E$ starts from v_i , $p_{ij} = -1$ if $e_j \in E$ ends in v_i , and 0 o/w
 - Edges in undirected graphs can be oriented arbitrarily
- $a_{ii} = 0$ (no self-loops)

Similarity matrix

- The **similarity matrix S** of n elements is n -by- n symmetric nonnegative matrix
 - s_{ij} is the similarity between i and j
 - 0s at diagonal
- Can be interpret as a weighted adjacency matrix of a (complete) **similarity graph**

Which similarity?

- Any distance metric (suitable for data) can be used as the similarity measure
 - $\text{sim}(\mathbf{x}, \mathbf{y}) = M - \|\mathbf{x} - \mathbf{y}\|$ where M is the maximum distance
 - Euclidean, Hamming, Jaccard, mutual information, Hellinger, ...
- Often the similarities are scaled to emphasize high similarity (and de-emphasize low similarity)
 - **Gaussian kernel** is common:
$$K_{\text{sim}}(\mathbf{x}, \mathbf{y}) = \exp\{-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2)\}$$

Sparsifying similarity graphs

- Similarity graphs are complete
 - But often we only need pairwise similarities of quite similar elements
- To sparsify the similarity graph, we can remove edges between dissimilar pairs
 - This sets the corresponding values in the matrix to 0

Getting non-complete graphs

- How to decide when vertices are too dissimilar?
- In **ϵ -neighbour graphs** we add an edge between two vertices that are within distance ϵ to each other
 - Usually the resulting graph is considered unweighted as all weights would be roughly similar
- In **k -nearest neighbour graphs** we connect two vertices if one is within the k nearest neighbours of the other
 - In **mutual k -nearest neighbour graph** we only connect two vertices if they're both in each other's k nearest neighbours

Which similarity graph?

- With ϵ -graphs choosing the parameter is hard
 - No single correct answer if different clusters have different internal similarities
- k -nearest neighbours can connect points with different similarities
 - But far-away high density regions become unconnected
- The mutual k -nearest neighbours is somewhat in between
 - Good for detecting clusters with different densities
- General recommendation: start with k -NN
 - Others if data supports that

Even More Matrices

- The (weighted) adjacency matrix \mathbf{A} has the weight of edge (i, j) at position a_{ij}
- The **degree matrix** $\mathbf{\Delta}$ of a graph is a diagonal n -by- n matrix with the (weighted) degree of vertex i at position $\Delta_{ii} = d_i$
 - $\Delta_{ii} = d_i = \sum_j a_{ij}$
- The **normalized adjacency matrix** \mathbf{M} is the adjacency matrix where in every row i all values are divided by d_i
 - Every row sums up to 1
 - $\mathbf{M} = \mathbf{\Delta}^{-1} \mathbf{A}$

Graph Laplacians

- The **Laplacian matrix** \mathbf{L} of a graph is the adjacency matrix subtracted from the degree matrix

$$\mathbf{L} = \mathbf{\Delta} - \mathbf{A} = \begin{pmatrix} \sum_{j \neq 1} a_{1,j} & -a_{1,2} & \cdots & -a_{1,n} \\ -a_{2,1} & \sum_{j \neq 2} a_{2,j} & \cdots & -a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n,1} & -a_{n,2} & \cdots & \sum_{j \neq n} a_{n,j} \end{pmatrix}$$

- The Laplacian is symmetric and positive semi-definite
 - Undirected graphs
 - Has n real, non-negative, orthogonal eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_n \geq 0$$

The normalized, symmetric Laplacian

- The **normalized, symmetric Laplacian matrix** L^s of a graph is defined as

$$\mathbf{\Delta}^{-1/2} \mathbf{L} \mathbf{\Delta}^{-1/2} = \mathbf{I} - \mathbf{\Delta}^{-1/2} \mathbf{A} \mathbf{\Delta}^{-1/2} = \begin{pmatrix} \frac{\sum_{j \neq 1} a_{1,j}}{\sqrt{d_1 d_1}} & -\frac{a_{1,2}}{\sqrt{d_1 d_2}} & \dots & -\frac{a_{1,n}}{\sqrt{d_1 d_n}} \\ -\frac{a_{2,1}}{\sqrt{d_2 d_1}} & \frac{\sum_{j \neq 2} a_{2,j}}{\sqrt{d_2 d_2}} & \dots & -\frac{a_{2,n}}{\sqrt{d_2 d_n}} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n,1}}{\sqrt{d_n d_1}} & -\frac{a_{n,2}}{\sqrt{d_n d_2}} & \dots & \frac{\sum_{j \neq n} a_{n,j}}{\sqrt{d_n d_n}} \end{pmatrix}$$

- Also positive semi-definite
- The **normalized, asymmetric Laplacian** L^a (a.k.a **random walk Laplacian**) is $L^a = \mathbf{\Delta}^{-1} \mathbf{L}$

Spectral clustering

Clustering as Graph Cuts

- A **cut** of a connected graph $G = (V, E)$ divides the set of vertices into two partitions S and $V \setminus S$ and removes the edges between them
 - Cut can be expressed by giving the set S
 - Or by giving the cut set $F = \{(v, u) \in E : |\{v, u\} \cap S| = 1\}$
- Graph cut clusters graph's vertices into two clusters
- A **k -way** cut cuts the graph into k disjoint set of vertices C_1, C_2, \dots, C_k and removes the edges between them

What is a good cut?

- Just any cut won't cut it
- In **minimum cut** the goal is to find any set of vertices such that cutting them from the rest of the graph requires removing the least number of edges
 - Least sum of weights for weighted graphs
- The minimum cut can be found in polynomial time
 - The max-flow min-cut theorem
- But minimum cut isn't very good for clustering purposes

What cuts would cut it? (1)

- We want a cut that penalizes imbalanced cluster sizes
- In **ratio cut**, the goal is to minimize the ratio of the weight of the edges in the cut set and the size of the clusters C_i

- Let $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$

- w_{ij} is the weight of edge (i, j)

$$\text{RatioCut} = \sum_{i=1}^k \frac{W(C_i, V \setminus C_i)}{|C_i|}$$

What cuts would cut it? (2)

- The **volume** of a set of vertices A is the weight of all edges connected to A
- $vol(A) = W(A, V) = \sum_{i \in A, j \in V} w_{ij}$
- In **normalized cut** we measure the size of C_i not by $|C_i|$ but by $vol(C_i)$

$$\text{NormalizedCut} = \sum_{i=1}^k \frac{W(C_i, V \setminus C_i)}{vol(C_i)}$$

Finding optimal RatioCut or NormalizedCut is NP-hard

Clusterings and matrices redux

- Recall that we can express a clustering using a binary cluster assignment matrix
- Let the i -th column of this matrix be \mathbf{c}_i
 - Clusters are disjoint so $\mathbf{c}_i^T \mathbf{c}_j = 0$
 - Cluster has $\mathbf{c}_i^T \mathbf{c}_i = \|\mathbf{c}_i\|^2$ elements
- We can get the $vol(C_i)$ and $W(C_i, V)$ using \mathbf{c}_i 's
 - $vol(C_i) = \sum_{j \in C_i} d_j = \sum_{r=1}^n \sum_{s=1}^n (\mathbf{c}_i)_r \Delta_{rs} (\mathbf{c}_i)_s = \mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i$
 - $W(C_i, C_i) = \sum_{r \in C_i} \sum_{s \in C_i} a_{rs} = \mathbf{c}_i^T \mathbf{A} \mathbf{c}_i$
 - $W(C_i, V \setminus C_i) = W(C_i, V) - W(C_i, C_i) = \mathbf{c}_i^T (\mathbf{\Delta} - \mathbf{A}) \mathbf{c}_i = \mathbf{c}_i^T \mathbf{L} \mathbf{c}_i$

Cuts using matrices

$$\text{RatioCut} = \sum_{i=1}^k \frac{W(C_i, V \setminus C_i)}{|C_i|} = \sum_{i=1}^k \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\|\mathbf{c}_i\|^2}$$

$$\text{NormalizedCut} = \sum_{i=1}^k \frac{W(C_i, V \setminus C_i)}{\text{vol}(C_i)} = \sum_{i=1}^k \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i}$$

Finding approximate cuts

- Re-writing the objective functions doesn't make them any easier
 - The complexity comes from the binary clustering assignments
- Relax!
 - Let \mathbf{c}_i 's take any real value
- Relaxed RatioCut:

$$J_{rc}(\mathcal{C}) = \sum_{i=1}^k \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\|\mathbf{c}_i\|^2} = \sum_{i=1}^k \left(\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|} \right)^T \mathbf{L} \left(\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|} \right) = \sum_{i=1}^k \mathbf{u}_i^T \mathbf{L} \mathbf{u}_i$$

- $\mathbf{u}_i = \mathbf{c}_i / \|\mathbf{c}_i\|$ i.e. the unit vector in the direction of \mathbf{c}_i

Solving the relaxed version

- We want to minimize the function J_{rc} over \mathbf{u}_i 's
 - We have a constraint that $\mathbf{u}_i^T \mathbf{u}_i = 1$
- To solve, derive w.r.t. \mathbf{u}_i 's and find the roots
 - Add Lagrange multipliers to incorporate the constraints:

$$\frac{\partial}{\partial \mathbf{u}_i} \left(\sum_{i=1}^k \mathbf{u}_i^T \mathbf{L} \mathbf{u}_i + \sum_{i=1}^k \lambda_i (1 - \mathbf{u}_i^T \mathbf{u}_i) \right) = 0$$

- Hence, $\mathbf{L} \mathbf{u}_i = \lambda_i \mathbf{u}_i$
 - \mathbf{u}_i is an eigenvector of \mathbf{L} corresponding to the eigenvalue λ_i

Which eigenvectors to choose

- We know that $\mathbf{L}\mathbf{u}_i = \lambda_i\mathbf{u}_i$
 - Hence $\lambda_i = \mathbf{u}_i^T \mathbf{L}\mathbf{u}_i$
- As we're minimizing the sum of $\mathbf{u}_i^T \mathbf{L}\mathbf{u}_i$'s we should choose the \mathbf{u}_i 's corresponding to the k smallest eigenvalues
 - They are our relaxed cluster indicators
- Note that we know that $\lambda_n = 0$ and that the corresponding eigenvector is $(n^{-1/2}, n^{-1/2}, \dots, n^{-1/2})$ (the graph is connected!)
 - No help on clustering...

The Fiedler vector and value

- The **Fiedler value** f of graph $G = (V, E)$ is the second-smallest eigenvalue λ_{n-1} of \mathbf{L}_G
 - The **Fiedler vector** is the corresponding eigenvector
- If we want to remove minimum number of vertices s.t. we cut the graph, we have to remove at least f vertices
- The **edge boundary** ∂U of subset $U \subseteq V$ is
$$\partial U = \{(u, v) \in E : u \in U, v \notin U\}$$
 - $|\partial U| \geq f|U||V \setminus U|/n$

$W(U, V \setminus U)$ for unweighted graphs

Normalized cut and choice of Laplacians

- For normalized cut similar procedure shows that we should select the k smallest eigenvectors of \mathbf{L}^s (or \mathbf{L}^a) instead of \mathbf{L}
- Which one we should choose?
 - Both ratio and normalized cut aim at minimizing intra-cluster similarity
 - But only normalized cut considers inter-cluster similarity
⇒ Either \mathbf{L}^s or \mathbf{L}^a
- The asymmetric Laplacian is better
 - With symmetric one further normalization is needed

Spectral clustering

- To do the clustering, we need to move our real-valued eigenvectors \mathbf{u}_i to binary cluster indicator vectors
- First, create a matrix \mathbf{U} with \mathbf{u}_i 's as its columns
 - Optionally, normalize the rows to sum up to 1 (esp. if using \mathbf{L}^s)
- Cluster the rows of this matrix using k -means (or any other clustering method)

Computational complexity


- Solving the eigenvectors is $O(n^3)$ in general or $O(n^2)$ if the similarity graph has as many edges as vertices
- The k -means on the \mathbf{U} matrix takes $O(tnk^2)$
 - t is the number of iterations in k -means

Spectral clustering pseudo-code

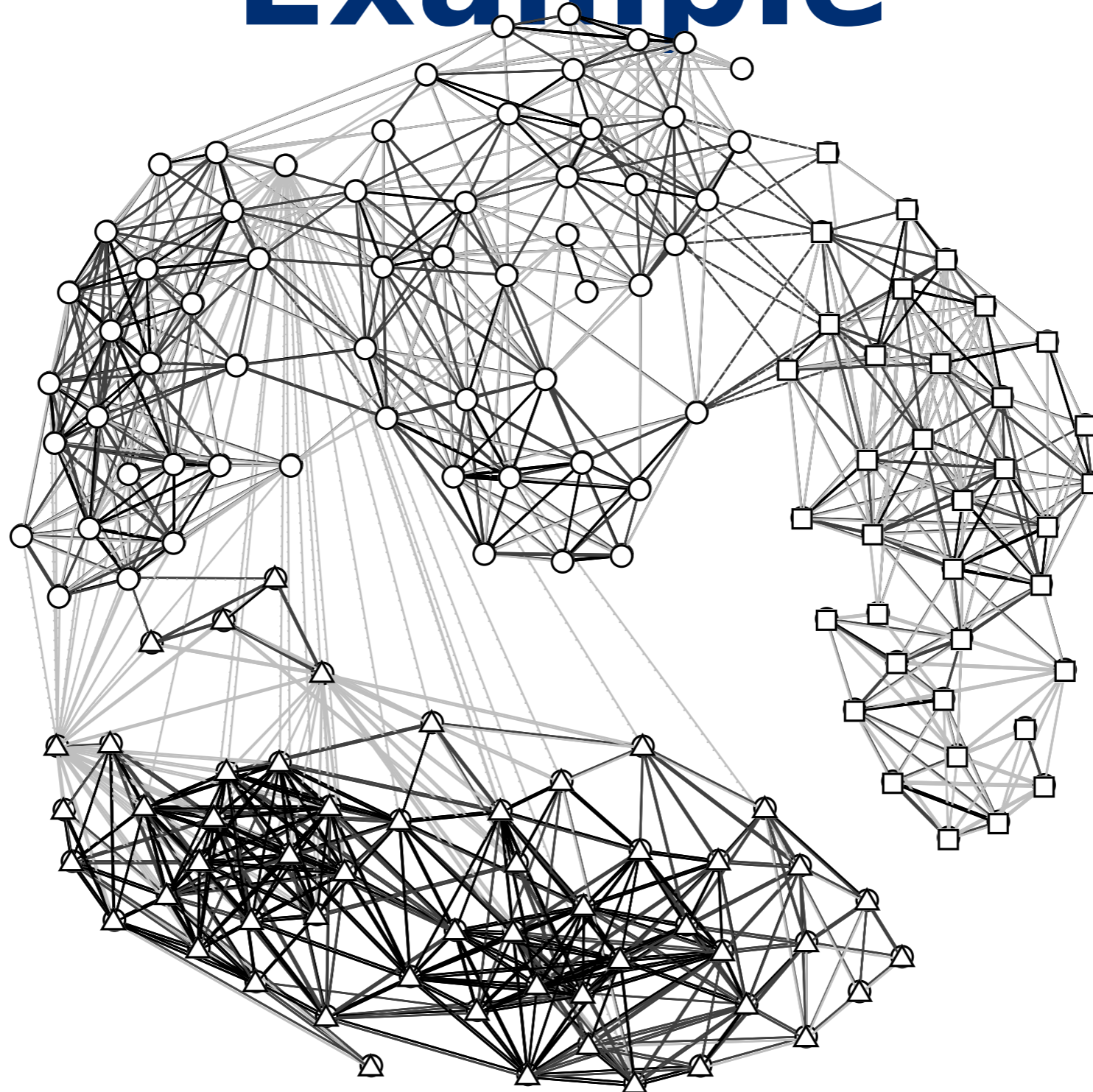
Assume connected graph

Algorithm 16.1: Spectral Clustering Algorithm

SPECTRAL CLUSTERING (\mathbf{D}, k):

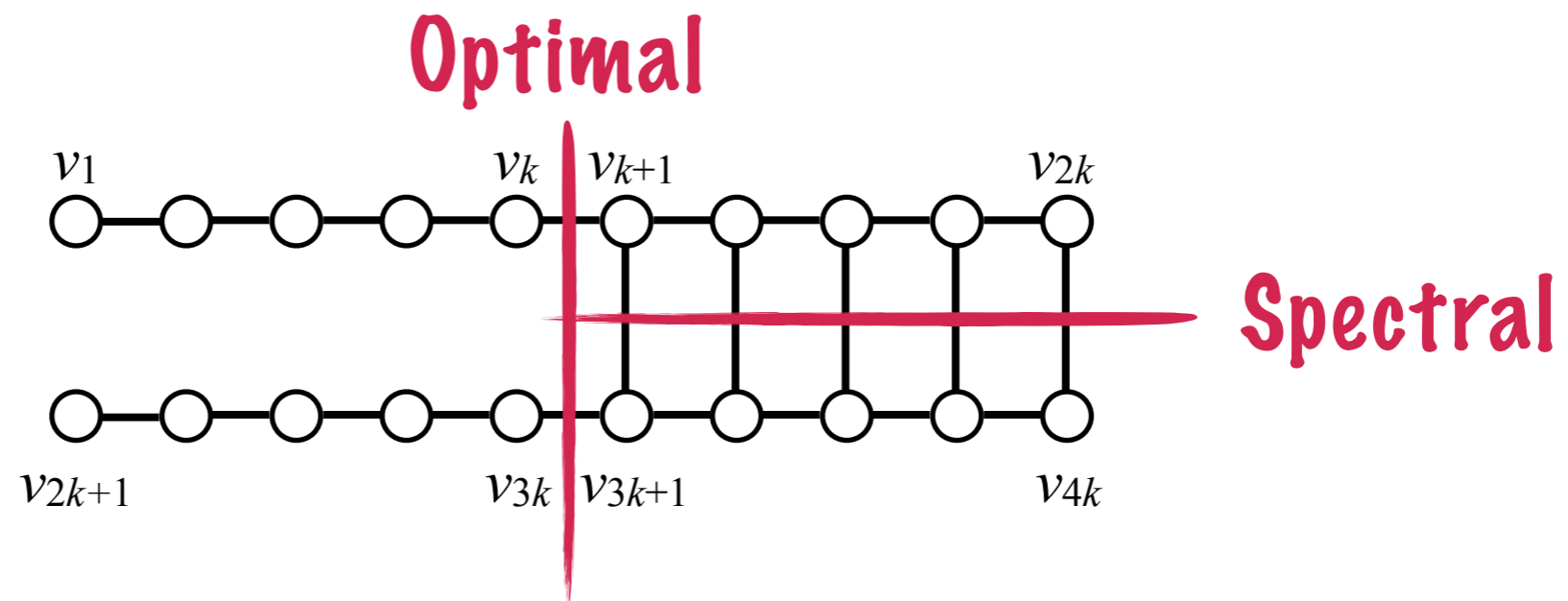
- 1 Compute the similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$  Sparsify if needed
 - 2 if *ratio cut* then $\mathbf{B} \leftarrow \mathbf{L}$
 - 3 else if *normalized cut* then $\mathbf{B} \leftarrow \mathbf{L}^s$ or \mathbf{L}^a
 - 4 Solve $\mathbf{B}\mathbf{u}_i = \lambda_i\mathbf{u}_i$ for $i = n, \dots, n - k + 1$, where $\lambda_n \leq \lambda_{n-1} \leq \dots \leq \lambda_{n-k+1}$
 - 5 $\mathbf{U} \leftarrow (\mathbf{u}_n \quad \mathbf{u}_{n-1} \quad \dots \quad \mathbf{u}_{n-k+1})$
 - 6 $\mathbf{Y} \leftarrow$ normalize rows of \mathbf{U} using (16.19)
 - 7 $\mathcal{C} \leftarrow \{C_1, \dots, C_k\}$ via K-means on \mathbf{Y}
-

Example



Is spectral clustering optimal?

- Spectral clustering is not always a good approximation of the graph cuts
- In cockroach graphs, spectral clustering cuts always horizontally, when optimal is to cut vertically
- Approximation ratio of $O(n)$



Summary

- Spectral clustering uses the linear algebra to do a very combinatorial task
 - Very deep insights between the combinatorial structure and the spectrum of the Laplacian
- Is not just one method, but a family of methods
 - Which cut, which Laplacian, how to sparsify, which clustering algorithm, ...

Next week

- Last lecture
- More on spectral stuff
- Wrap-up of the course
- Check your (current) homework marks/bonus points
- Ask-me-anything
 - (though I don't promise to answer to all questions)
 - Answers are more thought-out if you send them ahead of time