Lecture 12: Graph clustering and beyond

John Sylvester Nicolás Rivera Luca Zanetti Thomas Sauerwald

Lent 2019

UNIVERSITY OF CAMBRIDGE

Plan

In the last lecture we have introduced

- Graph clustering
- The notion of conductance
- Cheeger's inequality: $1 - \lambda_2 \lesssim \phi(G) \lesssim \sqrt{1 - \lambda_2}$.

In this lecture we will see

- How to formalise the notion of multiple clusters in a graph
- How to partition a graph in $k \geq 2$ clusters
- Applications (if time permits)

Multiway partitioning

Let $G = (V, E, w)$. Recall the notion of conductance of a set $S \subset V$:

$$\phi(S) = \frac{w(S, V \setminus S)}{\min\{\mathrm{vol}(S), \mathrm{vol}(V \setminus S)\}} = \max\left\{ \frac{w(S, V \setminus S)}{\mathrm{vol}(S)}, \frac{w(S, V \setminus S)}{\mathrm{vol}(V \setminus S)} \right\}$$

k-way partition: $\{S_1, \ldots, S_k\}$ s.t. $\emptyset \neq S_i \subset V$, $S_i \cap S_j = \emptyset$, $\bigcup_{i=1}^k S_i = V$

$$\phi_k(S_1, \ldots, S_k) = \max_{i=1,\ldots,k} \frac{w(S_i, V \setminus S_i)}{\mathrm{vol}(S_i)}$$

We can define the k-way conductance of G as

$$\phi_k(G) = \min_{\substack{\{S_1,\ldots,S_k\} \\ k\text{-way partition}}} \phi_k(S_1, \ldots, S_k)$$

Yet another variational characterisation

Let P be the transition matrix of a lazy walk on $G = (V, E, w)$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$.

Let $f_1, \ldots, f_k \colon V \to \mathbb{R}$, $\quad \mathrm{Span}(\{f_1, \ldots, f_k\}) \triangleq \left\{ \sum_{i=1}^{k} \alpha_i f_i \colon \alpha_1, \ldots, \alpha_k \in \mathbb{R} \right\}$

Let $f \colon V \to \mathbb{R}$, $\quad \mathcal{R}_G(f) \triangleq \frac{\sum_{\{u,v\} \in E} w(u,v)(f(u)-f(v))^2}{2 \sum_{u \in V} d(u) f(u)^2}$

Our new best friend

$$1 - \lambda_k = \min_{\substack{f_1, \ldots, f_k \neq 0 \\ f_i \perp f_j}} \max \left\{ \mathcal{R}_G(f) \colon f \in \mathrm{Span}(\{f_1, \ldots, f_k\}) \right\}$$

and the minimum is achieved by the eigenvectors for $\lambda_1, \ldots, \lambda_k$

Corollary

Let $f_1, \ldots, f_k \colon V \to \mathbb{R}$ be disjointly supported. Then,

$$\frac{1 - \lambda_k}{2} \leq \max_{i=1,\ldots,k} \mathcal{R}_G(f_i)$$

Eigenvalues and k-way conductance

Let $\{S_1, \ldots, S_k\}$ be a k-way partitioning of $G = (V, E, w)$ achieving $\phi_k(G)$

Define the indicator function $\mathbb{1}_{S_i} : V \to \{0, 1\}$ s.t. $\mathbb{1}_{S_i}(u) = 1 \iff u \in S_i$

Notice that $\mathbb{1}_{S_i}$'s are disjointly supported

$$\mathcal{R}_G(\mathbb{1}_{S_i}) = \frac{\sum_{\{u,v\}\in E} w(u,v)(\mathbb{1}_{S_i}(u) - \mathbb{1}_{S_i}(v))^2}{2\sum_{u\in V} d(u)\mathbb{1}_{S_i}(u)^2} = \frac{w(S_i, V \setminus S_i)}{2\,\mathrm{vol}(S_i)}$$

By the previous corollary,

$$\frac{1-\lambda_k}{2} \leq \max_{i=1,\ldots,k} \frac{w(S_i, V\setminus S_i)}{2\,\mathrm{vol}(S_i)} = \frac{1}{2}\phi_k(S_1, \ldots, S_k) = \frac{1}{2}\phi_k(G)$$

Higher-order Cheeger inequality

$$1 - \lambda_k \leq \phi_k(G) \leq O(k^2)\sqrt{\lambda_k}$$

(Easy consequence: $\lambda_k = 1$ iff at least k connected components in G)

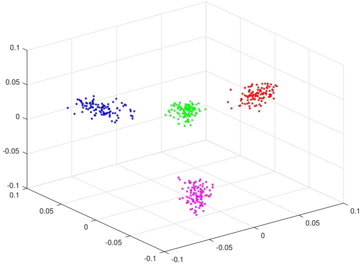
Example

Random graph $G = (V, E)$ where $V = S_1 \cup S_2 \cup S_3 \cup S_4$

$$\mathbf{P}[\,\{u, v\} \in E\,] = \begin{cases} 0.3 & u, v \in S_i \\ 0.03 & u \in S_i, v \in S_j \end{cases}$$

	f_1	f_2	f_3	f_4
S_1	$+1$	$\approx +1$	≈ -1	≈ -1
S_2	$+1$	$\approx +1$	$\approx +1$	$\approx +1$
S_3	$+1$	≈ -1	$\approx +1$	≈ -1
S_4	$+1$	≈ -1	≈ -1	$\approx +1$

- each eigenvector doesn't give us enough info by itself
- using all eigenvectors together, however, we can recover the clusters
- IDEA: map each vertex u to $F(u) = (f_1(u), f_2(u), f_3(u), f_4(u))^T$



How do we cluster points in $\mathbb{R}^k$?

Enter k-means clustering

INPUT:

- a set of n points $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$
- the number of clusters $k \geq 2$

GOAL:

- assign the points to k clusters so as to minimise the intra-cluster variance:

$$\min_{S_1,\ldots,S_k \text{ partition of } X} \sum_{i=1}^{k} \sum_{y \in S_i} \|y - c(S_i)\|^2$$

where

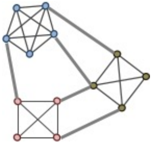
$$c(S_i) = \frac{1}{|S_i|} \sum_{y \in S_i} y \quad \text{is the center of } S_i$$

- k-means clustering is `NP-hard`!
- there are good approximation algorithms
- simple heuristics (usually!) work well in practice

Spectral clustering

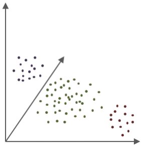
Goal: Partition $G = (V, E, w)$ in $k \geq 2$ well-separated clusters

$f_1, \ldots, f_k$ top eigenvectors of the random walk matrix of G

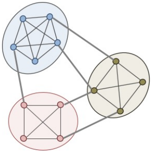


(1) Compute the spectral embedding $F\colon V \to \mathbb{R}^k$

$$F(u) = (f_1(u), \ldots, f_k(u))^T$$



(2) Solve k-means on $\{F(u)\}_{u \in V}$



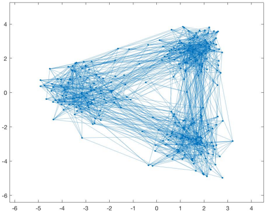
(3) Partition G according to the output of k-means

Example: Stochastic Block Models

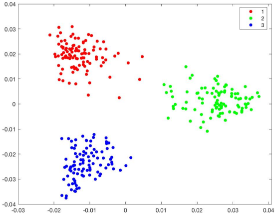
Graph $G = (V, E)$ with clusters
$S_1, S_2, S_3 \subset V; \quad 0 \le q < p \le 1$

$$\mathbf{P}[\, u \sim v \,] = \begin{cases} p & u, v \in S_i \\ q & u \in S_i, v \in S_j, i \neq j \end{cases}$$

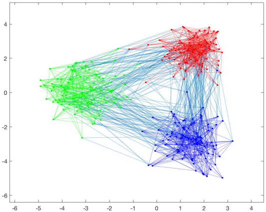
$|V| = 300, |S_i| = 100$
$p = 0.08, q = 0.01.$



Spectral embedding

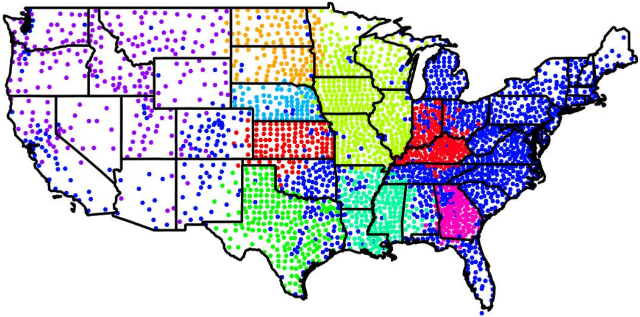


Output of Spectral Clustering



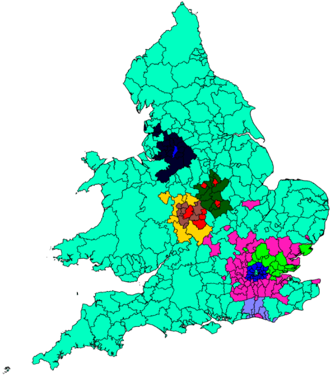
Example: US migration data

- Consider a dataset regarding internal migration in the US.
- For each pair of counties (i, j), $M(i, j)$ represents the number of people who migrated from i to j in the timeframe 2000-2010.
- This can be seen as a weighted directed graph, where each node is a county and M is its weighted adjacency matrix.
- We first make this graph undirected: compute $M + M^T$
- We compute the corresponding random walk matrix and apply Spectral Clustering ($k = 10$)



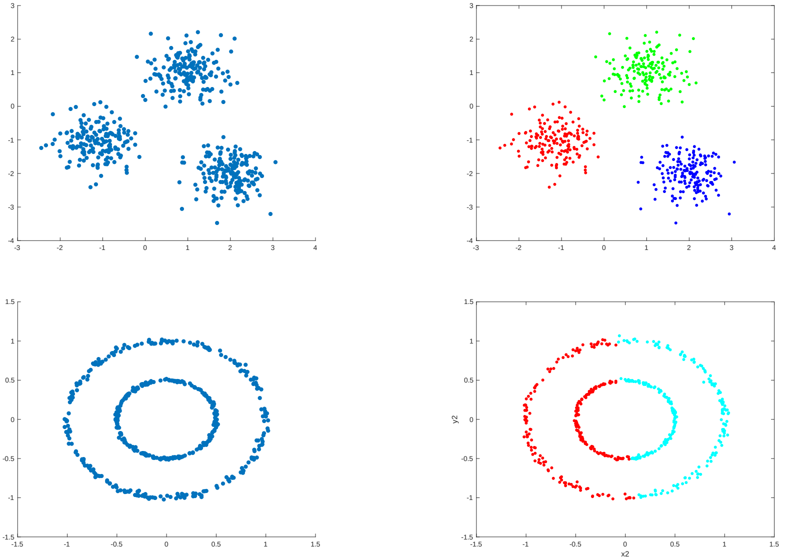
Example: England+Wales migration data

- For each pair of local authorities (i, j), $M(i, j)$ represents the number of people who migrated from i to j in the timeframe 2012-2017.
- We first make the graph undirected: $M + M^T$, and then compute its random walk matrix
- We apply Spectral Clustering ($k = 8$)



Spectral clustering beyond graphs

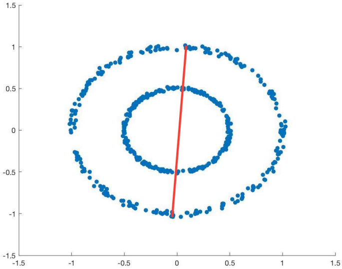
k-means clustering (examples)



Why k-means fail

k-means is able to recover only <u>convex</u> clusters:

- it divides the space in k regions with the following property: if we connect two points belonging to the same region, we never intersect any other region



Similarity graph

Given $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$, construct $G = (V, E, w)$:

- $x_i \in X \mapsto v_i \in V$
- $E = \binom{V}{2}$
- $w(v_i, v_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ (Gaussian similarity function)

Remarks:

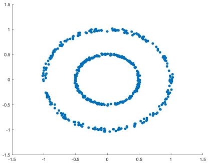
- $w(v_i, v_j)$ is large if x_i is close to x_j
- value of $\sigma \geq 0$ depends on the application (choose it by trial and error)
- large σ if, on average, pairwise nearest neighbours are far apart

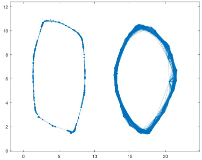
Problem: Since G is complete, from $\Theta(dn)$ to $\Theta(n^2)$ space.

Possible solution: r-nearest neighbour graph ($v_i \sim v_j$ iff x_j is one of the r-nearest neighbours of x_i or vice versa)

From geometric to graph clustering!

Example

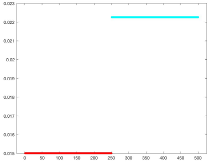


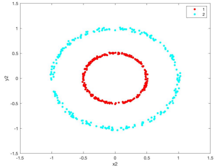


Similarity graph: Gaussian with $\sigma = 0.1$. Only edges with weight ≥ 0.01 shown.

Spectral partitioning:

1. Compute the eigenvector f_2 corresponding to λ_2
2. Order the vertices so that $f_2(u_1) \leq f_2(u_2) \leq \cdots \leq f_2(u_n)$
3. Choose "sweep" cut $(\{u_1, \ldots, u_i\}, \{u_{i+1}, \ldots, u_n\})$ with smallest conductance





References

Ulrike von Luxburg
A tutorial on spectral clustering
Statistics and computing (2007)

Santo Fortunato
Community detection in graphs
Physics reports (2010)

Daniel A. Spielman
Spectral Partitiong in a Stochastic Block Model
Lecture notes for Spectral Graph Theory (2015)
http://www.cs.yale.edu/homes/spielman/561/lect21-15.pdf

Appendix A: image segmentation

GOAL: identify different objects in an image

Construct similarity graph as follows:

- A pixel p is characterised by its position in the image and by its RGB value
- map pixel p in position (x, y) to a vector $v_p = (x, y, r, g, b)$
- construct similarity graph as explained earlier

Original image

Output SC (Gaussian, $\sigma = 10$)

Appendix B: Lloyd's algorithm for k-means

INPUT: $X \subset \mathbb{R}^d, k \geq 2$

GOAL:

$$\min_{S_1,\ldots,S_k \text{ partition of } X} \sum_{i=1}^{k} \sum_{y \in S_i} \|y - c(S_i)\|^2 \qquad \text{where} \quad c(S_i) = \frac{1}{|S_i|} \sum_{y \in S_i} y$$

Algorithm:

1. choose k random candidate centres $c_1, \ldots, c_k \in \mathbb{R}^d$
2. form clusters $S_1, \ldots, S_k$ by assigning each $y \in X$ to its nearest centre c_j:
   $S_j = \{y \in X : j = \text{argmin}_{1 \leq i \leq k} \|y - c_i\|^2\}$
3. compute the new centres of the clusters: $c_j = \frac{1}{|S_j|} \sum_{y \in S_j} y$
4. Repeat steps 2-3 until clusters don't change anymore.

- work usually well in practice, but
- exponential time to converge in the worst case
- no approximation guarantee
- by cleverly choosing the initial centres, we can obtain a $O(\log k)$-approximation algorithm (k-means++)