Optimization and Computational Linear Algebra for Data Science
Lecture 8: Graphs and Linear Algebra

Léo Miolane · leo.miolane@gmail.com

October 21, 2020

Warning: *This material is not meant to be lecture notes. It only gathers the main concepts and results from the lecture, without any additional explanation, motivation, examples, figures...*

1 Graphs

We start by a formal definition of a (simple non-oriented) graph:

Definition 1.1 (*Graph*)

A graph G is defined as a pair V_G, E_G where $V = V_G$ is the set of vertices of G and $E = E_G$ is the set of edges of G which is a subset of $V \times V$. Two vertices i, j are connected by an edge if $\{i, j\} \in E$. In such case we write $i \sim j$ and say that i and j are neighbors.

We say that G is connected if it is possible to go from every vertex in the graph to every other vertex through a series of edges, called a path.

Definition 1.2

The degree $\deg(i)$ of a node $i \in V$ is the number of its neighbors.

In this lecture we will only consider finite graphs, where V is finite. We let $n = \#V$. One can assume (up to renaming the vertices) that $V = \{1, \dots, n\}$.

Definition 1.3

We define the adjacency matrix $A \in \mathbb{R}^{n \times n}$ of the graph G by

$$A_{i,j} = \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{otherwise.} \end{cases}$$

The degree matrix of G is defined by $D = \mathrm{Diag}(\deg(1), \dots, \deg(n))$.

Notice that A is a symmetric matrix.

2 Graph Laplacian

Definition 2.1 (*Graph Laplacian*)

The Laplacian matrix of G is defined as

$$L = D - A.$$

Remark 2.1. *There exists a "normalized Laplacian" matrix $L_{\mathrm{norm}} = D^{-1/2} L D^{-1/2} = \mathrm{Id} - D^{-1/2} A D^{-1/2}$. Both L and L_{norm} enjoy similar properties. While we focus for simplicity on L in this lecture, several arguments advocate for using L_{norm} instead of L for clustering applications, see the discussion in [4, Section 8.4].*

Proposition 2.1

The matrix L satisfies the following properties:

1. *L is symmetric and positive semi-definite.*
2. *The smallest eigenvalue of L is 0 and a corresponding eigenvector is the constant one vector $\mathbb{1} \overset{\text{def}}{=} (1, 1, \ldots, 1)$.*
3. *L has n non-negative eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.*

The proposition above follows from the following key identity: for all $x \in \mathbb{R}^n$,

$$x^\mathsf{T} L x = \sum_{i \sim j} (x_i - x_j)^2, \tag{1}$$

where the sum is over all the edges $i \sim j$ of the graph. Consequently we have $x^\mathsf{T} L x = 0$ if and only if all the terms of the sum (1) are equal to zero, that is $x_i = x_j$ for all i and j that are connected by an edge. We conclude that

The eigenvectors of L associated with the eigenvalue 0 are the (non-zero) vectors that remains constant on the connected components of the graph G, i.e. the vectors x such that $x_i = x_j$ for all i, j in the same connected component of G.

We deduce:

Proposition 2.2

The graph G is connected if and only if $\lambda_2 > 0$. More generally, the multiplicity of the eigenvalue 0 of L (i.e. the number of i such that $\lambda_i = 0$) is equal to the number of connected components of L.

The eigenvalue λ_2 is called the algebraic connectivity or the Fiedler eigenvalue of G. Its magnitude reflects how well connected the graph is. Exercise: show that λ_2 increases when one adds edges to G.

Definition 2.2

For a graph G we define

- *The vertex connectivity $v(G)$ of G as the minimum number of nodes whose removal would result in losing connectivity of the graph.*
- *The edge connectivity $e(G)$ of G as the minimum number of edges whose removal would result in losing connectivity of the graph.*

Proposition 2.3 (*Fiedler [1]*)

$$\lambda_2 \leq v(G) \leq e(G).$$

We will only prove the first inequality, the second is a standard fact from graph theory.

Lemma 2.1

Let G^- be a graph obtained from G by removing one vertex (and all the edges linked to it). Then $\lambda_2(G^-) \geq \lambda_2(G) - 1$.

Using Lemma 2.1 one deduces Proposition 2.3. By definition of $v(G)$ one can disconnect G by removing $v(G)$ vertices. Let G' be the obtained graph. G' is disconnected so $\lambda_2(G') = 0$ by Proposition 2.2. We conclude using Lemma 2.1:

$$\lambda_2(G) \leq \lambda_2(G') + v(G) = v(G).$$

Proof of Lemma 2.1. Without loss of generalities, on can assume that the node n has been removed from G. Let G^+ be the graph obtained from G by connecting the node n to all the other nodes. Its Laplacian is given by

$$L_{G^+} = \begin{pmatrix} L_{G^-} + \mathrm{Id} & -\mathbb{1} \\ -\mathbb{1}^\mathsf{T} & n-1 \end{pmatrix}$$

where $\mathbb{1}$ denotes the all-one vector. Let x be an eigenvector of L_{G^-} associated with $\lambda_2(G^-)$ and orthogonal to $\mathbb{1}$. Compute

$$L_{G^+}\begin{pmatrix} x \\ 0 \end{pmatrix} = \begin{pmatrix} L_{G^-} + \mathrm{Id} & -\mathbb{1} \\ -\mathbb{1}^\mathsf{T} & n-1 \end{pmatrix}\begin{pmatrix} x \\ 0 \end{pmatrix} = \begin{pmatrix} L_{G^-}x + x \\ -\mathbb{1}^\mathsf{T}x \end{pmatrix} = (\lambda_2(G^-)+1)\begin{pmatrix} x \\ 0 \end{pmatrix}$$

because $\mathbb{1}^\mathsf{T}x = 0$. Hence $\lambda_2(G^-)+1$ is an eigenvalue of G^+ different from 0:

$$\lambda_2(G^+) \leq \lambda_2(G^-) + 1.$$

Remember that we obtained G^+ by adding edges to G: $\lambda_2(G^+) \geq \lambda_2(G)$. This concludes the proof. □

3 Spectral clustering with the graph Laplacian

From now, we assume that G is connected. We aim at clustering the nodes of G into k groups of nodes with as many intra-group edges as possible and as few inter-group edges as possible.

Algorithm 1 Spectral clustering with the Laplacian

Input: Graph Laplacian L, number of clusters k

1: Compute the first k eigenvectors $v_1, \ldots, v_k$ of the Laplacian matrix L.
2: Associate to each node i the vector $x_i = (v_2(i), \ldots, v_k(i)) \in \mathbb{R}^{k-1}$.
3: Cluster the points $x_1, \ldots, x_n$ with (for instance) the k-means algorithm.

The spectral clustering algorithm uses the eigenvectors of the Laplacian matrix to embed the nodes of the graph in the Euclidean space $\mathbb{R}^{k-1}$.

In the sequel we will focus on the case of two clusters ($k = 2$) for simplicity. In that case, the spectral clustering algorithm amounts of computing v_2 the second eigenvector of L (sometimes called the Fiedler eigenvector) and to cluster the nodes in

$$S = \{i \,|\, v_2(i) \geq \delta\} \qquad \text{and} \qquad S^c = \{i \,|\, v_2(i) < \delta\},$$

for some $\delta \in \mathbb{R}$. The next Proposition tells us that for $\delta = 0$, the cluster S is connected.

Proposition 3.1

Assume that G is connected. Let v_2 be an eigenvector associated to λ_2, the second smallest eigenvalue of L. Let

$$S = \{i \,|\, v_2(i) \geq 0\}.$$

Then the subgraph induced by S is connected.

See [3] for a proof.

4 Spectral clustering as a relaxation

For a set of nodes $S \subset V$ we define the cut of S by:

$$\text{cut}(S) = \text{« number of edges between } S \text{ and } S^c \text{ »} = \sum_{i \in S, j \in S^c} A_{i,j},$$

where A denotes the adjacency matrix of G. If we encode S in the vector x by

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{otherwise} \end{cases}$$

one can rewrite using (1)

$$\text{cut}(S) = \frac{1}{4} \sum_{i \sim j} (x_i - x_j)^2 = \frac{1}{4} x^\mathsf{T} L x.$$

However, $\text{cut}(S)$ is minimal for $S = \emptyset$ or $S = \{1, \dots, n\}$. Hence the cut is not a good metric, one should add some constraints on S. We can for instance force S to be balanced ($\#S = \#S^c = n/2$, assuming here that n is even) which is equivalent to say that x is orthogonal to $\mathbb{1}$ the vector of all ones. Thus

$$\text{minimize } \text{cut}(S) \text{ subject to } S \subset \{1, \dots, S\},\ S \text{ balanced},$$

is equivalent to

$$\text{minimize } x^\mathsf{T} L x \text{ subject to } x \in \{-1, 1\}^n,\ x \perp \mathbb{1}.$$

This problem is called the "minimum bisection" problem and is known to be NP hard. This comes from the fact that optimizing over the hypercube $\{\pm 1\}^n$ makes the problem combinatorial and difficult to solve. A way around is to relax the constraint by only forcing x to belong on the sphere of radius $\sqrt{n}$ and to solve:

$$\min_{\|x\| = \sqrt{n},\ x \perp \mathbb{1}} x^\mathsf{T} L x.$$

By Proposition 1.1 from Lecture 7, we know that the minimum is achieved by the Fiedler eigenvector v_2 (recall that $v_1 = \mathbb{1}$ is the first eigenvector of the Laplacian).

However v_2 may not belong to $\{\pm 1\}^n$. So in order to obtain a partition (S, S^c) from v_2, we finally perform a "rounding procedure":

$$S = \{i \,|\, v_2(i) \geq 0\}.$$

This is exactly the spectral clustering studied in the previous sections.

5 Spectral clustering beyond graphs

A natural generalization is to consider weighted graphs, where each edge has a weight indicating how close two connected neighbors are. This amounts to allow the entries of the adjacency matrix A to take different values from 0 and 1: $A_{i,j}$ quantify how "close" i and j are. We call such matrices "similarity matrices".

We can straightforwardly extend all the object defined above in the particular case of adjacency matrices to the case of similarity matrices. For instance $\deg(i) = \sum_j S_{i,j}$.

Further reading

The very interesting paper [2] uses the (normalized) Laplacian to cluster points in $\mathbb{R}^d$. See [4] for a very nice introduction to spectral clustering and [3] for lecture notes on spectral graph theory.

References

[1] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[2] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[3] Daniel Spielman. Spectral graph theory. *Lecture Notes, Yale University,* http://www.cs.yale.edu/homes/spielman/561/2012/, 2012.

[4] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.