Clustering methods

Vincent Divol

In previous chapters, we focused on the problem of **supervised learning**. For instance, in the image classification problem, we are given n training images $\mathbf{x_1}, \dots, \mathbf{x_n}$ that represent a car ($\mathbf{y_i} = +1$) or a plane ($\mathbf{y_i} = -1$). The goal is then to use these examples to predict whether a new image represents a car or a plane. We focus in this chapter on a different setting, called **unsupervised learning**. In this setting, we only have access to the inputs $\mathbf{x_1}, \dots, \mathbf{x_n}$, and not to the outputs. The goal is to create groups of inputs (called clusters) such that the inputs are similar to each other in each cluster, whereas the inputs in different clusters are dissimilar. The process of creating those different clusters is called **clustering**. In the object identification example, a clustering method aims at identifying two groups having different features in the set of observations $\mathbf{x_1}, \dots, \mathbf{x_n}$ (corresponding to cars and planes) without having access to any label. We present two different clustering methods: the k-means method, that can be applied to observations in $\mathbb{R}^d$, and spectral clustering, that take as an input a graph of similarities between data points.

1 The k-means problem

Let $\mathbb{X} = (x_1, \dots, x_n)$ be a collection of n points in $\mathbb{R}^d$. Assume that we want to summarize this set of n points with just one point x^*. What should be this point? A natural idea is to choose

$$x^* = \frac{x_1 + \dots + x_n}{n}, \tag{1}$$

the average of the n points. The average x^* is actually the minimizer of the function

$$y \in \mathbb{R}^d \mapsto \frac{1}{n} \sum_{i=1}^{n} \|y - x_i\|^2. \tag{2}$$

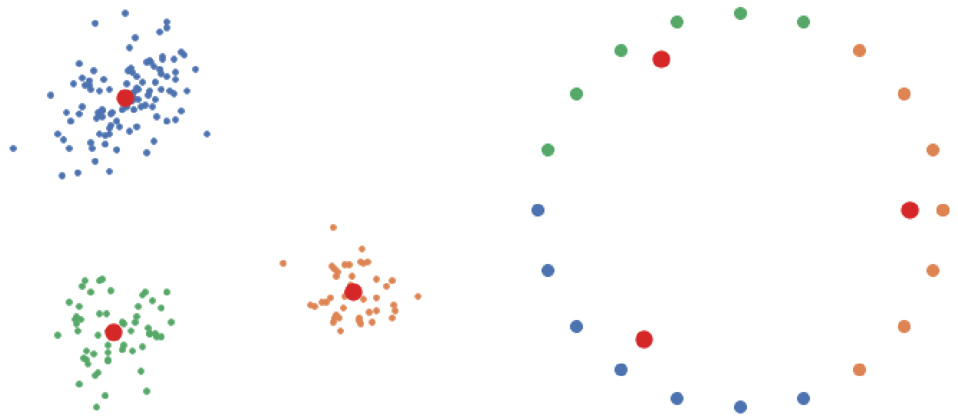


Figure 1: The k-means of two sets of points, for $k = 3$. For the second set of points, there is no uniqueness of the k-means.

This can for instance be seen by computing for the gradient of the above function, which is zero for $y = x^*$. Assume now that we want to summarize the set using k points. A generalization of (2) consists in minimizing

$$F_{k,\mathbb{X}} : (y_1, \dots, y_k) \in (\mathbb{R}^d)^k \mapsto \frac{1}{n} \sum_{i=1}^{n} \min_{l=1,\dots,k} \|y_l - x_i\|^2, \tag{3}$$

that is we are looking for k representatives (called centroids) such that the sum of the squared distances between each x_i and its closest centroid is minimal. One call the minimizer $F_{k,\mathbb{X}}$ the set of k-means of $\mathbb{X}$, that we denote by $(y_1^*, \dots, y_k^*)$. Note that the function $F_{k,\mathbb{X}}$ is non-convex in general (see Figure 2). Therefore, there might be several minimizers of $F_{k,\mathbb{X}}$, so we should in theory say 'a' set of k-means rather than 'the' set of k-means. We give in Figure 1 an example of data points $x_1, \dots, x_n$ where several k-means exist.

The k-means $(y_1^*, \dots, y_k^*)$ of the set of points $\mathbb{X}$ divide $\mathbb{X}$ into k clusters, by assigning each x_i to the lth cluster if y_l^* is centroid the closest to x_i. The simplest algorithm to compute the set of k-means is called Lloyd's algorithm, presented in Algorithm 1.

Each step of Lloyd's algorithm is made of two substeps. First, we assign every point x_i to a cluster: the cluster l is chosen if x_i is the closest to y_l^t. Second, we update the centroid by defining y_l^{t+1} as the average of the points

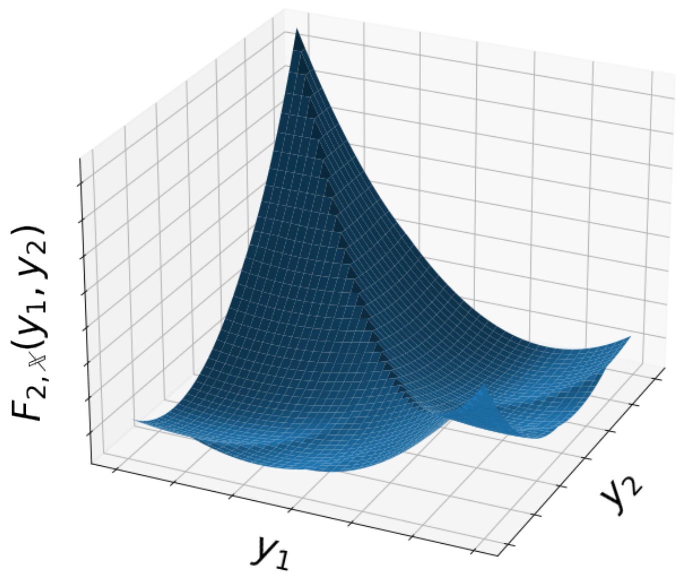


Figure 2: Graph of the function $F_{2,\mathbb{X}} : (\mathbb{R}^d)^2 \to \mathbb{R}$ for some set of points $\mathbb{X} \subset \mathbb{R}^d$, with $d = 1$. The function is not convex.

Algorithm 1: Lloyd's algorithm

```
1  Initialization: centroids $y_1^0, \dots, y_k^0 \in \mathbb{R}^d$;
2  for $t = 0, \dots, T-1$ do
3      for $l = 1, \dots, k$ do
4          let $I_l^t = \{i \in \{1, \dots, n\},\ y_l^t$ is the centroid the closest to $x_i\}$;
5          let $n_l^t$ be the number of elements in $I_l^t$;
6      end
7      for $l = 1, \dots, k$ do
8          let $y_l^{t+1} = \frac{1}{n_l^t} \sum_{i \in I_l^t} x_i$;
9      end
10 end
11 Output: $y_1^T, \dots, y_k^T \in \mathbb{R}^d$;
```

x_i of the cluster l.

Proposition 1. *Lloyd's algorithm coincides with Newton's method applied to the function $F_{k,\mathbb{X}}$.*

Proof. The function $F_{k,\mathbb{X}}$ is twice differentiable at $(y_1, \ldots, y_k)$ if there are no points of the set $\mathbb{X}$ that are equidistant to some centroid y_l. We will assume that this condition is always satisfied. In this case, defining the sets I_ls and the numbers n_ls as in Algorithm 1, we can compute $F_{k,\mathbb{X}}$:

$$\begin{aligned} F_{k,\mathbb{X}}(y_1, \ldots, y_k) &= \frac{1}{n} \sum_{i=1}^{n} \min_{l=1,\ldots,k} \|y_l - x_i\|^2 \\ &= \frac{1}{n} \sum_{l=1}^{k} \sum_{i \in I_l} \|x_i - y_l\|^2. \end{aligned} \tag{4}$$

The gradient of $\nabla F_{k,\mathbb{X}}(y_1, \ldots, y_k)$ is a vector of size dn that we write as

$$\begin{pmatrix} \nabla_{y_1} F_{k,\mathbb{X}}(y_1, \ldots, y_k) \\ \vdots \\ \nabla_{y_k} F_{k,\mathbb{X}}(y_1, \ldots, y_k) \end{pmatrix},$$

where $\nabla_{y_l} F_{k,\mathbb{X}}(y_1, \ldots, y_k)$ is the partial gradient of $F_{k,\mathbb{X}}$ with respect to y_l (that is a vector in $\mathbb{R}^d$). Let us compute $\nabla_{y_1} F_{k,\mathbb{X}}(y_1, \ldots, y_k)$. In (4), only the first term of the sum depends on y_1. Therefore,

$$\nabla_{y_1} F_{k,\mathbb{X}}(y_1, \ldots, y_k) = \frac{2}{n} \sum_{i \in I_1} (y_l - x_i) = \frac{2n_1}{n} \left(y_1 - \frac{1}{n_1} \sum_{i \in I_1} x_i \right). \tag{5}$$

Therefore,

$$\nabla F_{k,\mathbb{X}}(y_1, \ldots, y_k) = \frac{2}{n} \begin{pmatrix} n_1 \left(y_1 - \frac{1}{n_1} \sum_{i \in I_1} x_i \right) \\ \vdots \\ n_k \left(y_k - \frac{1}{n_k} \sum_{i \in I_k} x_i \right) \end{pmatrix}.$$

The gradient $\nabla F_{k,\mathbb{X}}(y_1, \ldots, y_k)$ is decomposed into k blocks, where the lth block depends linearly on y_l. The Hessian $\nabla^2 F_{k,\mathbb{X}}(y_1, \ldots, y_k)$ is therefore a diagonal matrix equal to

$$\frac{2}{n}\begin{pmatrix} n_1 \mathrm{Id}_d & & \\ & \ddots & \\ & & n_k \mathrm{Id}_k \end{pmatrix} \tag{6}$$

By definition, an iterate of Newton's method is given by

$$\begin{aligned}
\begin{pmatrix} y'_1 \\ \vdots \\ y'_k \end{pmatrix} &= \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} - \nabla^2 F_{k,\mathbb{X}}(y_1, \dots, y_k)^{-1} \nabla F_{k,\mathbb{X}}(y_1, \dots, y_k) \\
&= \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} - \begin{pmatrix} \frac{1}{n_1}\mathrm{Id}_d & & \\ & \ddots & \\ & & \frac{1}{n_k}\mathrm{Id}_k \end{pmatrix} \begin{pmatrix} n_1\left(y_1 - \frac{1}{n_1}\sum_{i \in I_1} x_i\right) \\ \vdots \\ n_k\left(y_k - \frac{1}{n_k}\sum_{i \in I_k} x_i\right) \end{pmatrix} \\
&= \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} - \begin{pmatrix} y_1 - \frac{1}{n_1}\sum_{i \in I_1} x_i \\ \vdots \\ y_k - \frac{1}{n_k}\sum_{i \in I_k} x_i \end{pmatrix} \\
&= \begin{pmatrix} \frac{1}{n_1}\sum_{i \in I_1} x_i \\ \vdots \\ \frac{1}{n_k}\sum_{i \in I_k} x_i \end{pmatrix}.
\end{aligned}$$

Those iterates are exactly the one given by Lloyd's algorithm. □

When using Lloyd's algorithm, we are applying Newton's method on the function $F_{k,\mathbb{X}}$, that is in general not convex. It is therefore not surprising that the performance of Lloyd's algorithm will crucially depend on the initialization. If the initialization $(y_1^0, \dots, y_k^0)$ is close enough to the minimizers $(y_1^*, \dots, y_k^*)$, then Lloyd's algorithm will converge very quickly to the minimizer (as expected for Newton's method). However, with bad initialization, the iterates of Lloyd's algorithm will remain stuck in local minima that correspond to configurations far from being optimal, see Figure 3. The `k-mean++` algorithm gives a procedure to find a good initialization for Lloyd's algorithm [Arthur and Vassilvitskii, 2006]. It is the default initialization method in the Python `scikit-learn` library, and shows good performance in practice.

Example 2. Add example.

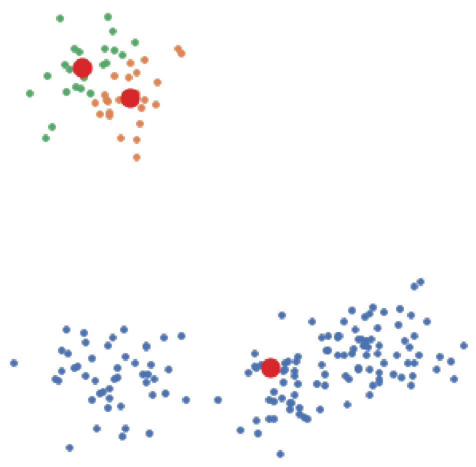


Figure 3: With bad initialization, Lloyd's algorithm may converge to a (very) bad configuration. The three centroids computed by Lloyd's algorithm with a bad initialization are displayed in red, and the associated clusters in respectively green, orange, and blue.

2 Spectral clustering

Spectral clustering is in many ways an improvement upon k-means. Unlike k-means, that can only be applied to observations in $\mathbb{R}^d$, spectral clustering can be applied to any set of observations, as long as a notion of similarity between the observations is defined.

Definition 3. *A weighted graph $\mathcal{G} := \mathcal{G}(W)$ with n vertices is described by a $n \times n$ matrix of weights $W = (W_{ij})_{1 \le i,j \le n}$, where the weights W_{ij} are nonnegative and symmetric (that is $W_{ij} = W_{ji}$).*

Some examples of weighted graphs include:

1. The weight matrix W only contains 0 and 1. In this case, we think of $\mathcal{G}$ as representing a non-weighted graph: if $W_{ij} = 1$, then there is an edge between the vertex i and the vertix j, and if $W_{ij} = 0$ then there is no edge.

2. A particular example of non-weighted graph is the ε-neighborhood graph. Let $x_1, \dots, x_n \in \mathbb{R}^d$ and let $\varepsilon > 0$. We define $W_{ij} = 1$ if

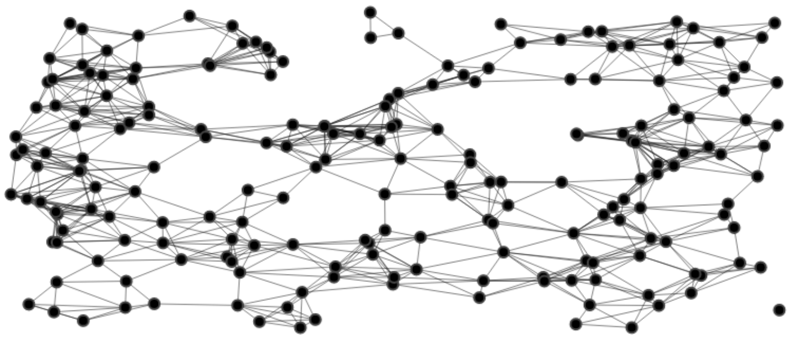


Figure 4: The ε-neighborhood graph of a set of points in $\mathbb{R}^2$.

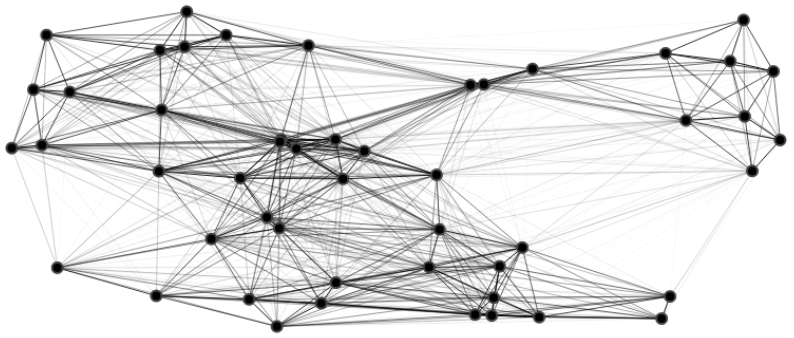


Figure 5: The σ-gaussian graph of a set of points in $\mathbb{R}^2$. Thicker edges indicate higher weights.

$\|x_i - x_j\| \leq \varepsilon$, and 0 otherwise: we connect two points by an edge if and only if they are at distance less than ε.

3. A variation of this construction is the gaussian graph, where the weights W_{ij} are given by $\exp(-\|x_i - x_j\|^2/(2\sigma^2))$ for some parameter $\sigma > 0$. Two nearby points are assigned a large weight, whereas if two points x_i and x_j are far away, then the weight W_{ij} is small.

Spectral clustering allows one to detect the presence of clusters in a weighted graph. Informally, a cluster is a set of vertices such that the similarity between two points of the cluster is high, whereas the similarity between a point of a cluster and a point outside the cluster is small. In the graph setting, the similarity between vertices is given by the weight matrix W.

Definition 4. *Let $\mathcal{G}$ be a weighted graph with weight matrix W. Let $i \in \{1, \dots, n\}$.*

1. *The **neighbors** of i are the vertices j such that $W_{ij} > 0$. We then write $i \sim_{\mathcal{G}} j$.*
2. *The **degree** D_i of a vertex i is defined as $D_i = \sum_{j=1}^n W_{ij}$. We let D be the $n \times n$ diagonal matrix with entries D_i on the diagonal. We call D the **degree matrix**.*
3. *We say that two vertices i and j are **connected** if there exists a path $i = i_1, i_2, \dots, i_{k-1}, i_k = j$ such that $i_l \sim_{\mathcal{G}} i_{l+1}$ for every $1 \leq l \leq k-1$.*
4. *A **connected component** in $\mathcal{G}$ is a set $\mathcal{C}$ of vertices in $\mathcal{G}$ such that all pairs of vertices in $\mathcal{C}$ are connected, but no vertices in $\mathcal{C}$ is connected to a vertex not in $\mathcal{C}$.*

A first natural idea is to define the clusters in our graph as the connected components. By construction, similarity is positive between points in a connected component, but zero between two points in different clusters. However, this notion is not robust to noise, see Figure 6. Indeed, the top graph in Figure 6 has two connected component, but it suffices to add one edge (bottom graph) to create a graph with a single connected component. We still want to think about the bottom graph as containing two clusters.

To built upon this first idea, we need to make the transform the notion of "being connected" into a quantitative notion, that is we need to make sense

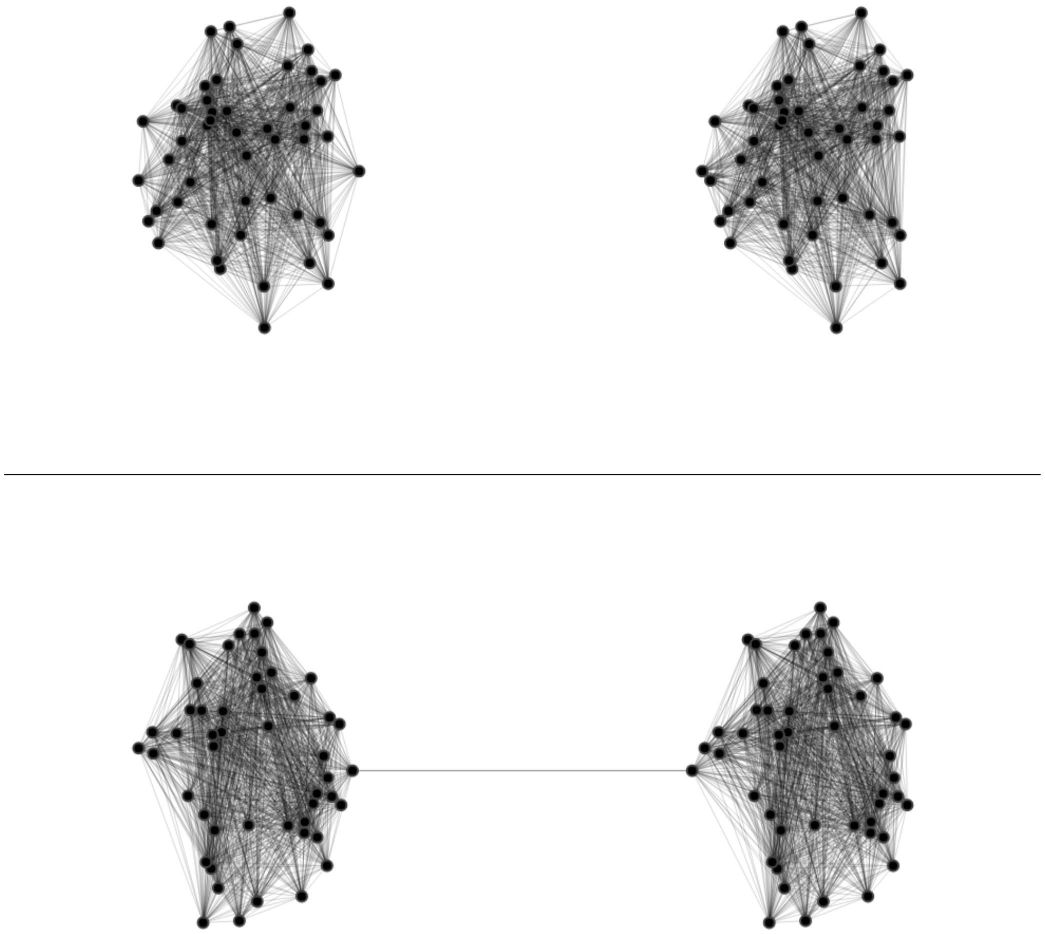


Figure 6: Top: a graph with two connected components. Bottom: a graph with a single connected component, but two clusters.

of the sentence "How connected are two vertices?". Here is a very elegant way to do so. Let i and j be two vertices. Consider a particle p starting at i and that will randomly walk around the graph: at step t, if the particle p is at vertex k, then it will move to one of the neighbors l of k with probability equal to $Q_{kl} = W_{kl}/D_k$. Informally, if the particle p takes a lot of time to go from vertex i to vertex j, then it means that the two vertices are not well-connected. On the opposite, the random particle p goes on average from i to j in a short amount of time, then i and j can be considered closed.

Definition 5. *We define the probability transition matrix Q as $D^{-1}W$, its entries are given by $Q_{ij} = W_{ij}/D_i$. The (random walk)* ***Laplacian***[1] *of the graph $\mathcal{G}$ is the matrix $L = \mathrm{Id}_n - Q$.*

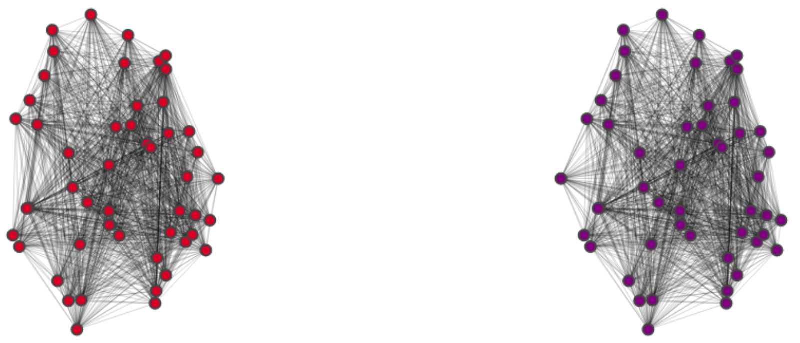
The spectral properties of the Laplacian (namely the eigenvalues and the eigenvectors of the matrix L) contain relevant information on the geometry of the graph $\mathcal{G}$, see Figure 7. Let $A \subset \{1, \dots, n\}$. We let e_A be the vector in $\mathbb{R}^n$ with entries $(e_A)_i = 1$ if $i \in A$ and $(e_A)_i = 0$ otherwise.

Proposition 6. *Let $\mathcal{G}$ be a weighted graph with associated Laplacian L.*

1. *All the eigenvalues of L are nonnegative.*
2. *The matrix L has always 0 as an eigenvalue.*
3. *The multiplicity of 0 as an eigenvalue is the number k of connected components of the graph $\mathcal{G}$. An orthogonal basis of the eigenspace associated to the eigenvalue 0 is given by $(e_{\mathcal{C}_1}, \dots, e_{\mathcal{C}_k})$ where $\mathcal{C}_1, \dots, \mathcal{C}_k$ are the connected components of $\mathcal{G}$.*

Proof. Introduce the matrix $L' = D^{1/2} L D^{-1/2}$. The eigenvalues and eigenvectors of L and L' are related: for $\lambda \in \mathbb{R}$ and $u \in \mathbb{R}^n$, it holds that $Lu = \lambda u$ if and only if $L'v = \lambda v$, where $v = D^{1/2}u$. In particular, L and L' have the same eigenvalues, and the eigenvectors of L and L' are related by the simple relation $v = D^{1/2}u$.

[1]The name Laplacian is also used in calculus to refer to a differential operator. The interested reader may find connections between the two concepts in the Appendix.



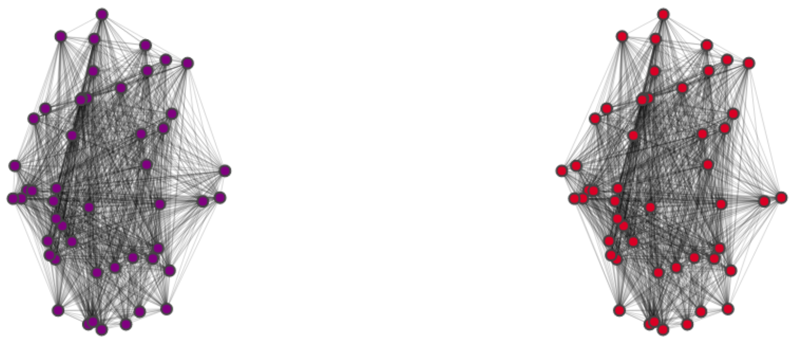


Figure 7: A basis of two eigenvectors of the eigenspace of L corresponding to the eigenvalue 0. The color of the vertex i can go from purple (the ith entry u_i of the eigenvector u is equal to 0) to red (u_i is equal to 1).

1. It suffices to show that L' is positive semi-definite. Let $v \in \mathbb{R}^n$. It holds that

$$\begin{aligned}
v^\top L' v &= v^\top v - v^\top D^{-1/2} W D^{-1/2} v \\
&= \sum_{i=1}^{n} v_i^2 - \sum_{1\le i,j\le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} \\
&= \sum_{i=1}^{n} \frac{D_i}{D_i} v_i^2 - \sum_{1\le i,j\le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} \\
&= \sum_{1\le i,j\le n} W_{ij} \frac{v_i^2}{D_i} - \sum_{1\le i,j\le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} \quad \text{(by definition of } D_i\text{)} \\
&= \frac{1}{2} \left(\sum_{1\le i,j\le n} W_{ij} \frac{v_i^2}{D_i} - 2 \sum_{1\le i,j\le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} + \sum_{1\le i,j\le n} W_{ij} \frac{v_i^2}{D_i} \right) \\
&= \frac{1}{2} \left(\sum_{1\le i,j\le n} W_{ij} \frac{v_i^2}{D_i} - 2 \sum_{1\le i,j\le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} + \sum_{1\le i,j\le n} W_{ij} \frac{v_j^2}{D_j} \right) \\
&= \frac{1}{2} \sum_{1\le i,j\le n} W_{ij} \left(\frac{v_i}{\sqrt{D_i}} - \frac{v_j}{\sqrt{D_j}} \right)^2 \ge 0, \qquad (7)
\end{aligned}$$

where we switch the roles of the dummy variables i and j in the last sum at the second to last line. All in all, this implies that L' is positive semi-definite, and therefore so is L.

2. It is clear from (7) that the vector $v = (\sqrt{D_1}, \ldots, \sqrt{D_n})$ is an eigenvector of L' with associated eigenvalue 0. Therefore 0 is also an eigenvalue of L.

3. Note also that the $D^{-1/2}v$ is an eigenvector of L, which is the vector with 1 in all of its entries. Assume that $k = 1$ (there is only one connected component), and let us show that this is the only eigenvector of L' (up to a constant). Having $v^\top L' v = 0$ is equivalent to having $v_i/\sqrt{D_i} = v_j/\sqrt{D_j}$ for every vertices i, j with $W_{ij} > 0$. As all the vertex are connected (because $k = 1$), we can always find a path from any vertex i_1 to some other vertex i_m. Going from vertex to vertex in this path, we see that the quantity $v_i/\sqrt{D_i}$ stays constant along that path. In particular, this implies that the vector v must satisfy

$v_i/\sqrt{D_i} = \text{cst}$ for every vertex i. This implies that $v = (\sqrt{D_1}, \dots, \sqrt{D_n})$ is the only eigenvector of L' associated with 0 (up to a multiplicative constant). Therefore, $D^{-1/2}v$ is the only eigenvector of L associated with 0 (up to a multiplicative constant).

Let us now treat the case $k > 1$. In this case, up to relabelling the indexes, we can write the matrix L as a block diagonal matrix

$$L = \begin{pmatrix} L_1 & & \\ & \vdots & \\ & & L_k \end{pmatrix}$$

where the block L_l is the Laplace matrix associated with the subgraph given by the connected component $\mathcal{C}_l$. As 0 is an eigenvalue with multiplicity 1 of each of the matrix L_l, it is an eigenvalue of L with multiplicity k. Furthermore, a basis of the eigenspace of L associated with 0 is given by one eigenvector of each of the block L_l with 0 eigenvalue. According to the case $k = 1$, the vectors $e_{\mathcal{C}_l}$ are such eigenvectors.

$\square$

To put it simply, the multiplicity of 0 as an eigenvalue of the Laplacian L gives the number of connected components, while the associated eigenvectors exactly give the said connected components: all the relevant information on the connected components is given by the spectral properties of the Laplacian matrix. However, the Laplacian matrix is a much richer object, that is more robust to perturbation. Indeed, if we consider a graph with two large connected components, and add a single edge between those two components, then only one single connected component remain, as said earlier. The multiplicity of 0 in the Laplacian matrix has moved from 2 to 1. However, one can show that there is now a nonzero eigenvalue λ in the spectrum of L that is very small. Also, an eigenvector corresponding to this eigenvalue will be approximately 0 on one connected component and approximately 1 on the other, that is to say the eigenvector is an approximation of the eigenvector $e_{\mathcal{C}_1}$ given by Proposition 6. This phenomenon is showcased in Figure 8.

The spectral clustering method relies on this phenomenon, see Algorithm 2. For $i \in \{1, \dots, n\}$, let $e_i = e_{\{i\}}$ be the vector in $\mathbb{R}^n$ representing the vertex i. In the case where $\mathcal{G}$ contains exactly k connected components, one has $\langle e_i, e_{\mathcal{C}_l} \rangle = 1$ if $i \in \mathcal{C}_l$ and 0 otherwise. Therefore, in this "ideal" situation,

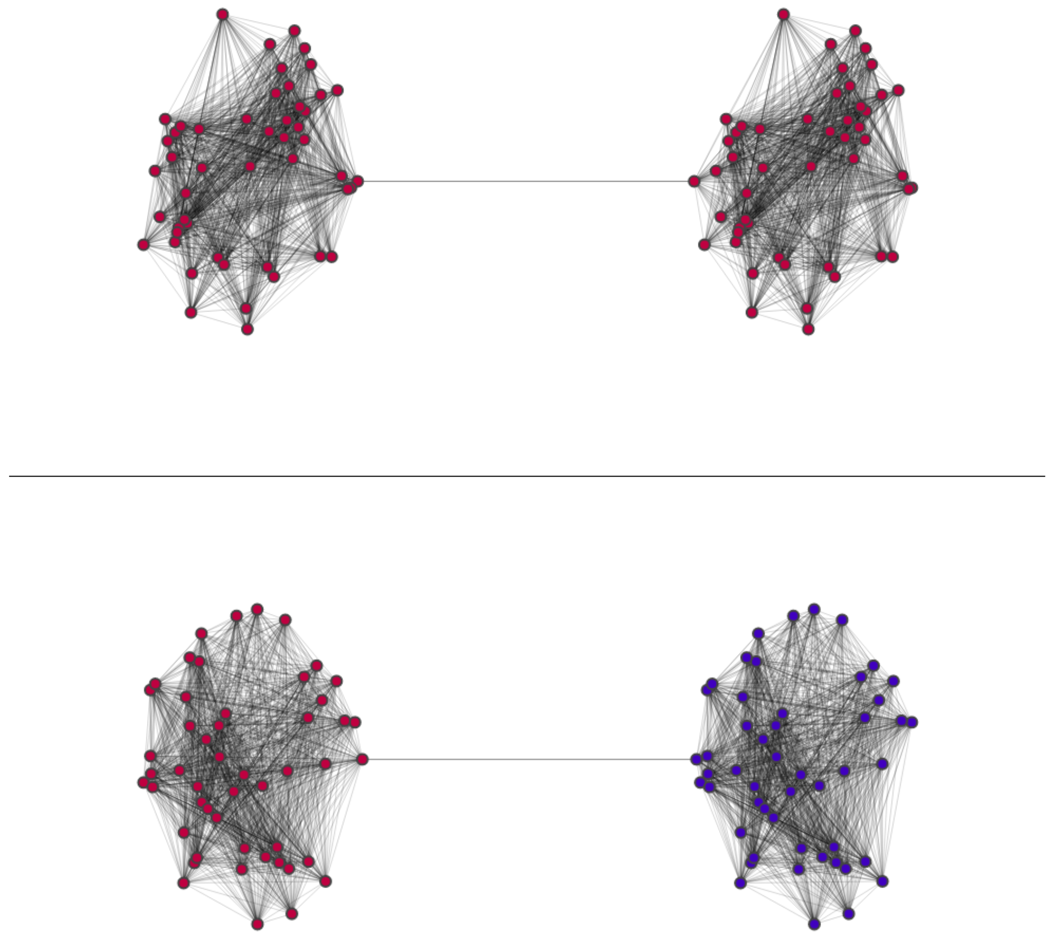


Figure 8: Top: the eigenvector of L corresponding to the eigenvalue 0. Bottom: the eigenvector of L corresponding to the smallest positive eigenvalue, equal to $\simeq 10^{-3}$.

the points $a_1, \ldots, a_n \in \mathbb{R}^k$ are all on one of the k axes of $\mathbb{R}^k$ (in its canonical basis). In particular, Lloyd's algorithm will have no trouble clustering the points $a_1, \ldots, a_k$. In the more realistic setting where only "approximate" connected components exist, one can show that the points a_i corresponding to the "approximate" lth connected component will stay close to the lth axis of $\mathbb{R}^k$. Therefore, Lloyd's algorithm will still be able to identify the clusters.

Algorithm 2: Spectral clustering

```
1 Input: weighted graph $\mathcal{G}$ with weight matrix $W$; number of clusters
    $k$;
2 compute the Laplace matrix $L$;
3 compute the eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and the associated
    eigenvectors $u_1, \ldots, u_n$.
4 for $i = 1, \cdots, n$ do
5   | let $a_i = (\langle u_1, e_i \rangle, \ldots, \langle u_k, e_i \rangle) \in \mathbb{R}^k$;
6 end
7 apply Lloyd's algorithm to $a_1, \ldots, a_n \in \mathbb{R}^k$;
```

Appendix - the Laplace matrix and the heat equation

The reader familiar with calculus may have already heard about the Laplacian in another context. If f is a twice differentiable function from $\mathbb{R}^d$ to $\mathbb{R}$, the Laplacian of f is defined by

$$\Delta f = -\sum_{j=1}^{d} \frac{\partial^2 f}{\partial x_j^2}, \tag{8}$$

(note the sign convention that we use here). How is this operator related to the Laplacian on the graph that we have defined in this chapter? In 1822, Joseph Fourier published his treatise *Théorie analytique de la chaleur*, where he considered the following problem. Consider a metal rod (that we identify with $[0, 1]$) whose temperature at both extremities is fixed at 1 throughout the whole experiment. Assume that at time $t = 0$ a certain distribution of heat f_0 is given in the rod (mathematically, a function $f_0 \geq 0$ with $\int f_0 =$

1). How will the distribution of heat evolve through time? Our intuition dictates that as time t progresses, the distribution of heat f_t at time t will become smoother, until the temperature is uniform in the rod at $t = \infty$, corresponding to $f_\infty = 1$. The equation governing the distribution of heat is the **heat equation**

$$\frac{\partial f_t}{\partial t} + \Delta f_t = 0. \tag{9}$$

Solving this equation indeed shows that the distribution of heat f_t will converge exponentially fast to a uniform temperature. A microscopic vision of the heat diffusion process consists in picturing a large number of small particles randomly moving in the rod. At each time t, the particle will with probability 1/2 infinitesimally moves to the left, or to the right. The initial distribution f_0 represents exactly the original distribution of those particles in the rod (if the heat is originally high at a point x in the rod, then f_0 is high, meaning that there are originally a lot of particles oscillating around x). For a very small value of t, we can approximate the time derivative $\partial f_t/\partial t$ by $(f_t - f_0)/t$. The heat equation (9) then becomes

$$f_t \simeq f_0 - t\Delta f_0 = (\mathrm{Id} - t\Delta) f_0.$$

This means that the distribution of heat at a small time t is approximately given by the operator $(\mathrm{Id} - t\Delta)$.

Let us now go back to the discrete world. Consider the graph $\mathcal{G}_n$ with n vertices, and weights

$$W_{ij} = \begin{cases} 1 & \text{if } |i-j| = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The graph $\mathcal{G}_n$ is a "line" graph, that is a discrete analogue of the metal rod. If we start with a particle at position i, that moves at random, then after one step, the probabilities of the position of i are given by the vector Q_i (where $Q = D^{-1}W$ is the probability transition matrix). Let us now consider a large number of particles on the graph, distributed according to some distribution $u_0 \in \mathbb{R}^n$ (with nonnegative entries and such that $\sum_{i=1}^{n}(u_0)_i = 1$). We make all those particles take one random step on the graph. After this step, the distribution of the particles is given by $Qu_1 = (\mathrm{Id}_n - L)u_0$ (by definition of the graph Laplacian). This is exactly the discrete time analogue of the equation $f_t \simeq (\mathrm{Id} - t\Delta) f_0$.

Therefore, the graph Laplacian and the Laplace operator from calculus share the same physical meaning: they both represent how particles moving at random behave on average on a small scale of time.

References

[Arthur and Vassilvitskii, 2006] Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical report, Stanford.