PDEs and Graph Based Learning

Summer School on Random Structures in Optimizations and Related Applications

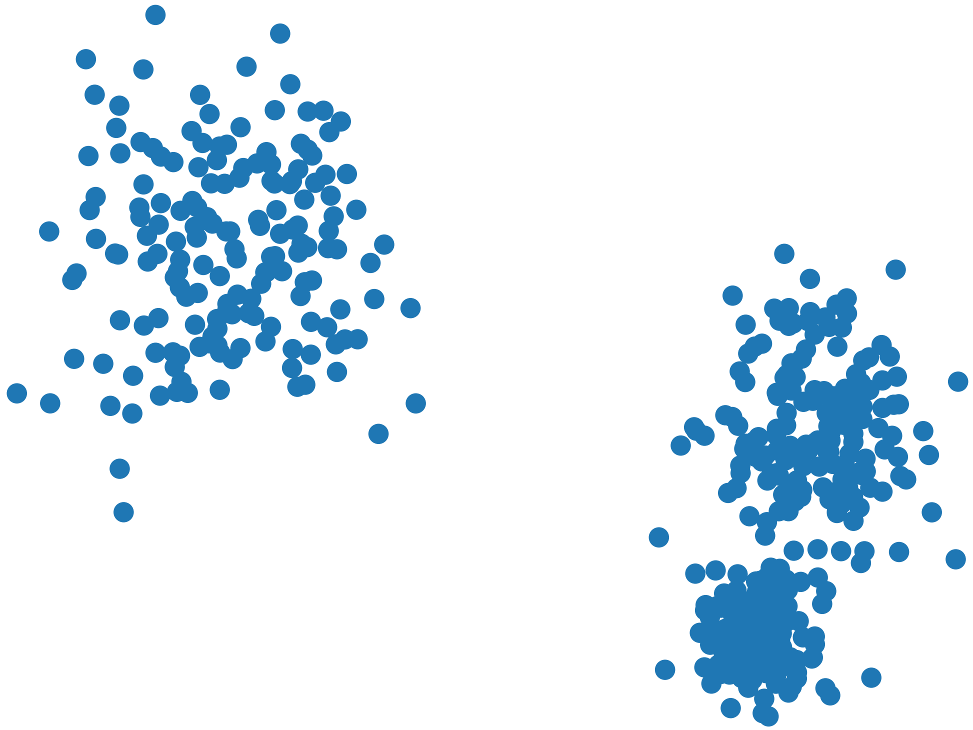
Lecture 1: k-means and spectral clustering

Instructor: Jeff Calder (jcalder@umn.edu)

Web: http://www-users.math.umn.edu/~jwcalder

Lecture Notes: http://www-users.math.umn.edu/~jwcalder/5467S21

Clustering

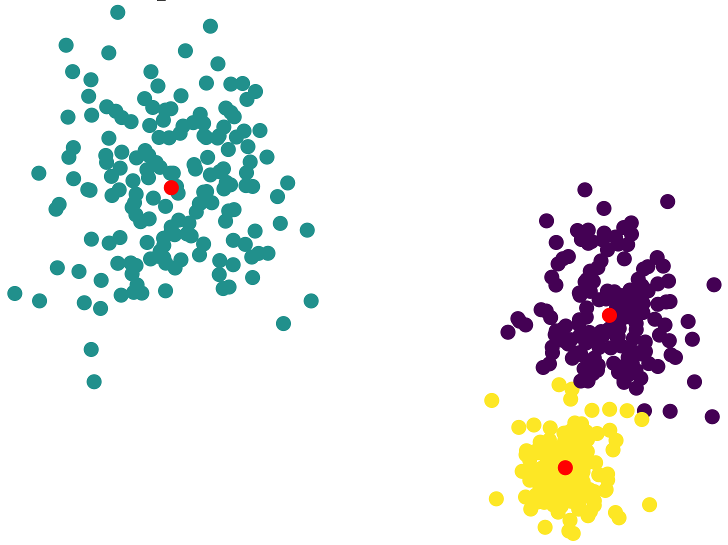


k-means clustering

Let $x_1, x_2, \ldots, x_m$ be datapoints in $\mathbb{R}^n$. The k-means algorithm is guided by the task of minimizing the energy over the choice of cluster centers c_i

$$E(c_1, c_2, \ldots, c_k) = \sum_{i=1}^{m} \min_{1 \leq j \leq k} \|x_i - c_j\|^2.$$

Minimizing E is an NP-hard problem.



k-means clustering

k-means algorithm: We start with some randomized initial values for the means $c_1^0, c_2^0, \ldots, c_k^0$, and iterate the steps below until convergence.

1. Update the clusters

$$\Omega_j^t = \left\{ x_i \, : \, \|x_i - c_j^t\|^2 = \min_{1 \le \ell \le k} \|x_i - c_\ell^t\|^2 \right\}. \tag{1}$$

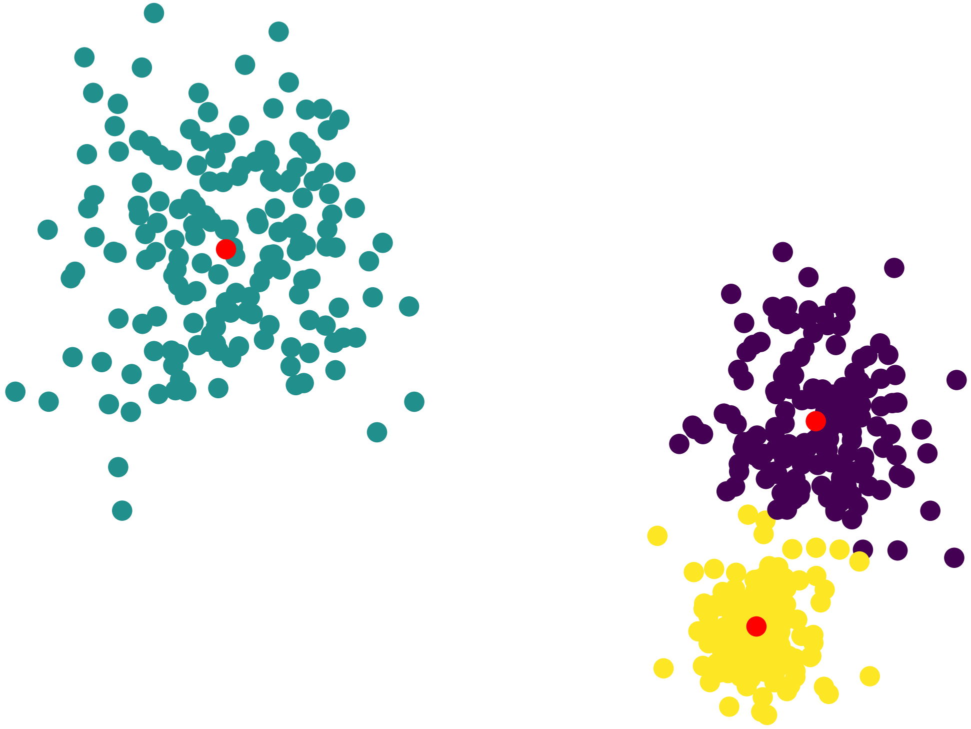
2. Update the cluster centers (means)

$$c_j^{t+1} = \frac{1}{\#\Omega_j^t} \sum_{x \in \Omega_j^t} x. \tag{2}$$

The algorithm converges when $c_j^{t+1} = c_j^t$ for all j.

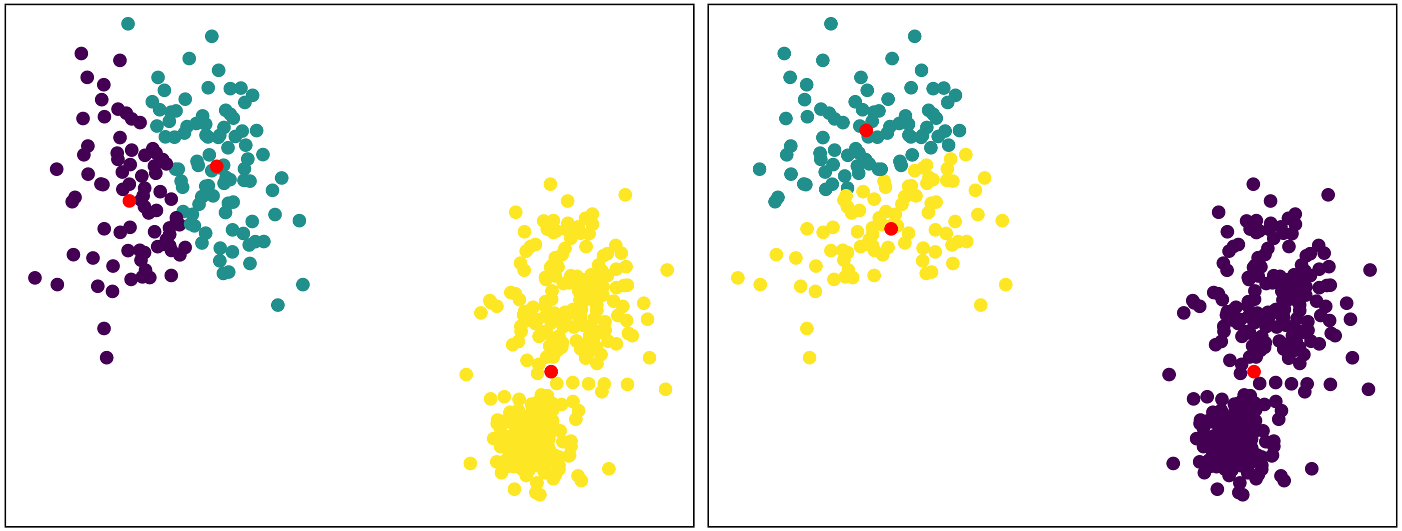
Code Demo

k-means clustering result



Poor clustering by k-means

Clustering depends on the random initialization.



k-means clustering in Python (.ipynb)

Lemma on centroids

Lemma 1. *Let $y_1, y_2, \ldots, y_m$ be points in $\mathbb{R}^n$, and define the function $f : \mathbb{R}^n \to \mathbb{R}$ by*

$$f(x) = \sum_{i=1}^{m} \|y_i - x\|^2.$$

Then the unique minimizer of f is the centroid

$$c = \frac{1}{m} \sum_{i=1}^{m} y_i.$$

In particular, $f(c) < f(x)$ if $x \neq c$.

Proof:

$$f(c) = \sum_{i=1}^{m} \|y_i - c\|^2$$

$$= \sum_{i=1}^{m} \left(\|y_i\|^2 - 2c^T y_i + \|c\|^2 \right)$$

$$= \sum_{i=1}^{m} \|y_i\|^2 - 2c^T \underbrace{\sum_{i=1}^{m} y_i}_{=mc} + \sum_{i=1}^{m} \|c\|^2$$

$$= \sum_{i=1}^{m} \|y_i\|^2 - 2\underbrace{c^T m c}_{c^Tc = \|c\|^2} + m\|c\|^2$$

$$= \sum_{i=1}^{m} \|y_i\|^2 - m\|c\|^2$$

$$\boxed{f(c) = \sum_{i=1}^{m} \left(\|y_i\|^2 - \|c\|^2\right)}$$

$$f(x) = \sum_{i=1}^{m} \|y_i - x\|^2$$

$$= \sum_{i=1}^{m} \left(\|y_i\|^2 - 2x^T y_i + \|x\|^2\right)$$

$$= \sum_{i=1}^{m} \left(\|y_i\|^2 - \|c\|^2\right) + \sum_{i=1}^{m} \left(\|c\|^2 - 2x^T y_i + \|x\|^2\right)$$

$$= f(c) + m\|c\|^2 - 2x^T mc + m\|x\|^2$$

$$= f(c) + m\left(\|c\|^2 - 2x^T c + \|x\|^2\right)$$

$$= f(c) + m\|c-x\|^2$$

$$\boxed{f(x) = f(c) + m\|c-x\|^2}$$

Alt. Proof $\quad Df(x) = 2\sum_{i=1}^{m} (y_i - x) = 0$

$$\therefore \; y_i = c$$

Convergence of k-means

Recall the k-means enegy

$$E(c_1, c_2, \ldots, c_k) = \sum_{i=1}^{m} \min_{1 \le j \le k} \|x_i - c_j\|^2. \tag{3}$$

Theorem 2. *The k-means algorithm descends on the energy (3), that is*

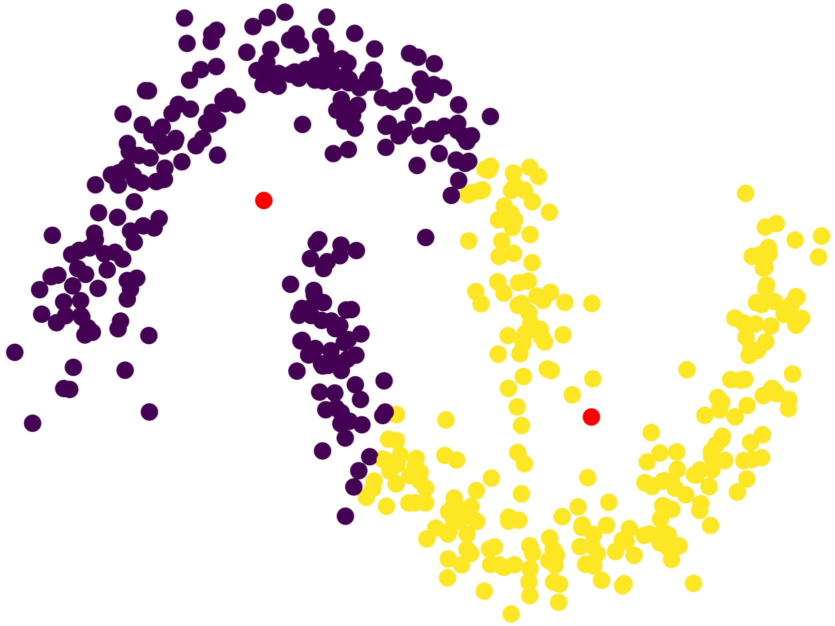
$$E(c_1^{t+1}, c_2^{t+1}, \ldots, c_k^{t+1}) \le E(c_1^t, c_2^t, \ldots, c_k^t). \tag{4}$$

Furthermore, we have equality in (4) if and only if $c_j^{t+1} = c_j^t$ for $j = 1, \ldots, k$, and hence the k-means algorithm converges in a finite number of iterations.

Note:

- k-means does **not** in general find a global minimum of E.
- It is useful because it is fast, guaranteed to converge, and often finds good clustering.

k-means on two-moons



- Sometimes a single point is not a good representative of a cluster, in Euclidean distance.
- Instead, we can try to cluster points so that nearby points are assigned to the same cluster, without specifying cluster centers.

Weight matrix

Let $x_1, x_2, \ldots, x_m$ be points in $\mathbb{R}^n$. To encode which points are nearby, we construct a weight matrix W, which is an $m \times m$ symmetric matrix where $W(i,j)$ represents the similarity between datapoints x_i and x_j. A common choice for the weight matrix is Gaussian weights

$$W(i,j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \tag{5}$$

where the σ is a free parameter that controls the scale at which points are connected.

Graph cuts for binary clustering

A graph-cut approach to clustering minimizes the graph cut energy

$$E(z) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} W(i,j)|z(i)-z(j)|^2 \tag{6}$$

over label vectors $z \in \{0,1\}^m$.

Notes:

- The value $z(i) \in \{0,1\}$ indicates which cluster x_i belongs to.
- The graph-cut energy is the sum of the edge weights $W(i,j)$ that must be **cut** to separate the dataset into two clusters.

Balanced graph cuts for binary clustering

Minimizing the graph cut energy

$$E(z) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} W(i,j)|z(i)-z(j)|^2$$

can lead to very unbalanced clusters (e.g., one cluster can have just a single point).

A more useful approach is to minimize a balanced graph cut energy

$$E_{balanced}(z) = \frac{\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} W(i,j)|z(i)-z(j)|^2}{\sum_{i=1}^{n} z(i)\sum_{j=1}^{n}(1-z(j))}. \tag{7}$$

The denominator is the product of the number of points in each cluster, which is maximized when the clusters are balanced.

Balanced graph-cut problems are NP hard.

Relaxing the graph cut problem

To relax the graph-cut problem, we consider minimizing the graph cut energy

$$E(z) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} W(i,j)|z(i) - z(j)|^2$$

over all real-vectors $z \in \mathbb{R}^m$. We still have a balancing issue (here $z = 0$ is a minimizer), so we impose the balancing constraints

$$(*) \qquad \mathbf{1}^T z = \sum_{i=1}^{m} \underbrace{\boxed{z_i}}_{z(i)} = 0 \quad \text{and} \quad \|z\|^2 = \sum_{i=1}^{m} z(i)^2 = 1.$$

Definition 3. The *binary spectral clustering problem* is

$$\text{Minimize } E(z) \text{ over } z \in \mathbb{R}^m, \text{ subject to } \mathbf{1}^T z = 0 \text{ and } \|z\|^2 = 1.$$

The resulting clusters are $C_1 = \{x_i : \underline{z(i) > 0}\}$ and $C_2 = \{x_i : \underline{z(i) \le 0}\}$.

The graph Laplacian and Fiedler vector

Let W be a symmetric $m \times m$ matrix with nonnegative entries.

Definition 4. The *graph Laplacian* matrix L is the $m \times m$ matrix

$$L = D - W \tag{8}$$

where D is the diagonal matrix with diagonal entries

$$D(i,i) = \sum_{j=1}^{m} W(i,j). \quad = \text{degree of } i$$

Lemma 5. *Then the graph cut energy can be expressed as*

$$E(z) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} W(i,j)|z(i) - z(j)|^2 = z^T L z,$$

where L is the graph Laplacian.

Proof: $$E(z) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} w(i,j)\left(z(i)^2 - 2z(i)z(j) + z(j)^2\right)$$

$$= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} w(i,j)z(i)^2 + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} w(i,j)z(j)^2 - \sum_{i=1}^{m}\sum_{j=1}^{m} w(i,j)z(i)z(j)$$

$$= \sum_{i=1}^{m}\underbrace{\sum_{j=1}^{m} w(i,j)}_{D(i,i)}z(i)^2 - \sum_{i=1}^{m}\underbrace{\sum_{j=1}^{m} w(i,j)z(j)}_{(Wz)(i)}\, z(i)$$

$$= \sum_{i=1}^{m} D(i,i)z(i)^2 - \sum_{i=1}^{m}(Wz)(i)\,z(i)$$

$$= z^T D z - z^T W z = z^T L z$$

since $L = D - W$. ▨

Properties of the graph Laplacian

Lemma 6. *Let $L = D - W$ be the graph Laplacian corresponding to a symmetric matrix W with nonnegative entries. The following properties hold.*

(i) *L is symmetric.*

(ii) *L is positive semi-definite (i.e., $z^T Lz \geq 0$ for all $z \in \mathbb{R}^m$).*

(iii) *All eigenvalues of L are nonnegative, and the constant vector $z = \mathbf{1}$ is an eigenvector of L with eigenvalue $\lambda = 0$.*

$$L\mathbf{1} = D\mathbf{1} - W\mathbf{1}$$

$$(W\mathbf{1})(i) = \sum_{j=1}^{m} W(i,j)\,1 = D(i,i)$$

$$\Rightarrow \quad W\mathbf{1} = D\mathbf{1}.$$

Fiedler vector

Let $v_1, v_2, \dots, v_m$ be the eigenvectors of the graph Laplacian, with corresponding eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m.$$

Definition 7. The second eigenvector v_2 of the graph Laplacian L is called the *Fiedler vector.*

Theorem 8. *The Fiedler vector $z = v_2$ solves the binary spectral clustering problem*

$$\textit{Minimize } E(z) \textit{ over } z \in \mathbb{R}^m, \ \textit{subject to } \underline{\mathbf{1}^T z = 0} \textit{ and } \underline{\|z\|^2 = 1}.$$

$\underline{\text{Proof}}$: Let z be a minimizer

Write $z = a_1 v_1 + a_2 v_2 + \cdots + a_m v_m$

$$\left(v_1 = \mathbf{1}/\sqrt{m}, \quad \|v_i\| = 1 \right)$$

$$0 = \mathbf{1}^T z = a_1 \mathbf{1}^T v_1 + a_2 \cancelto{0}{\mathbf{1}^T v_2} + \cdots + a_m \cancelto{0}{\mathbf{1}^T v_m}$$

since $v_i^T v_j = 0 \; , \; i \neq j$

$$= a_1 \frac{m}{\sqrt{m}} \implies a_1 = 0$$

$$1 = \|z\|^2 = \sum_{i=1}^{m} a_i^2 = a_2^2 + a_3^2 + \cdots + a_m^2$$

$$E(z) = z^T L z = z^T L \sum_{i=2}^{m} a_i v_i$$

$$= z^T \sum_{i=2}^{m} a_i L v_i$$

$$= z^T \sum_{i=2}^{m} a_i \lambda_i v_i$$

$$= \sum_{i=2}^{m} a_i \lambda_i \underbrace{z^T v_i}_{= a_i}$$

$$= \sum_{i=2}^{m} \lambda_i a_i^2$$

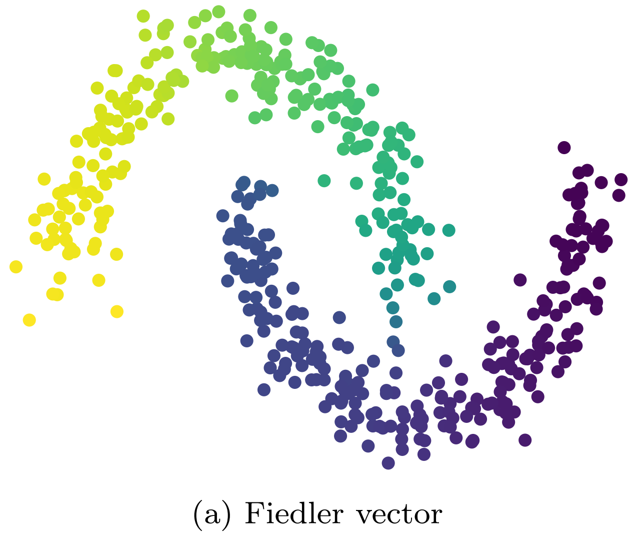
$$E(z) = \sum_{i=2}^{m} \lambda_i a_i^2 \geq \lambda_2 \sum_{i=2}^{m} a_i^2 = \lambda_2$$

$$\lambda_2 \leq \lambda_3 \leq \lambda_4 \leq \cdots \leq \lambda_m$$

Settin $z = v_2$ $(a_2 = 1,\ a_3 = 0, a_4 = 0, \ldots)$

yields $E(z) = \lambda_2$ ▨

Example


(a) Fiedler vector

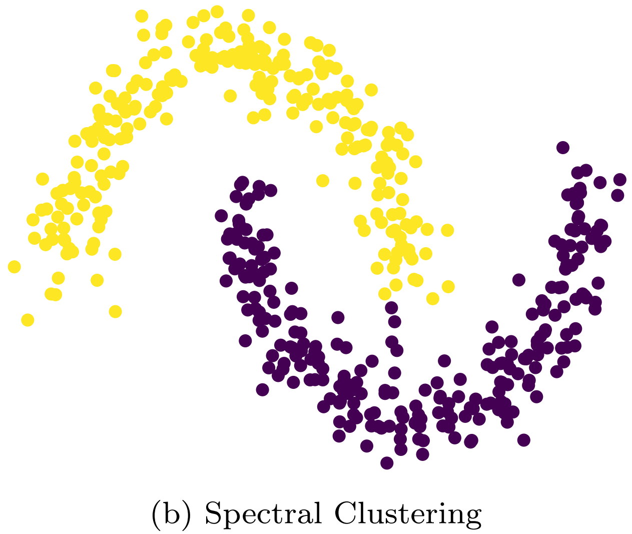

(b) Spectral Clustering

Figure 1: (a) The Fiedler vector and (b) spectral clustering on the 2-moons dataset.

Spectral embeddings

Let $v_1, v_2, v_3, \ldots$ be the normalized eigenvectors of L, in order of increasing eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots$. The spectral embedding corresponding to L is the map $\Phi : I_m \to \mathbb{R}^k$ (recall $I_m = \{1, 2, \ldots, m\}$ are the indices of our datapoints) given by

$$\Phi(i) = (v_1(i), v_2(i), \ldots, v_k(i)). \tag{9}$$

Since the first eigenvector v_1 is the trivial constant eigenvector, it is also common to omit this to obtain the embedding

$$\Phi(i) = (v_2(i), v_3(i), \ldots, v_{k+1}(i)).$$

There are other normalizations of the graph Laplacian that are commonly used, such as the symmetric normalization $L = D^{-1/2}(D - W)D^{-1/2}$, and the spectral embedding for a normalized Laplacian is defined analagously.

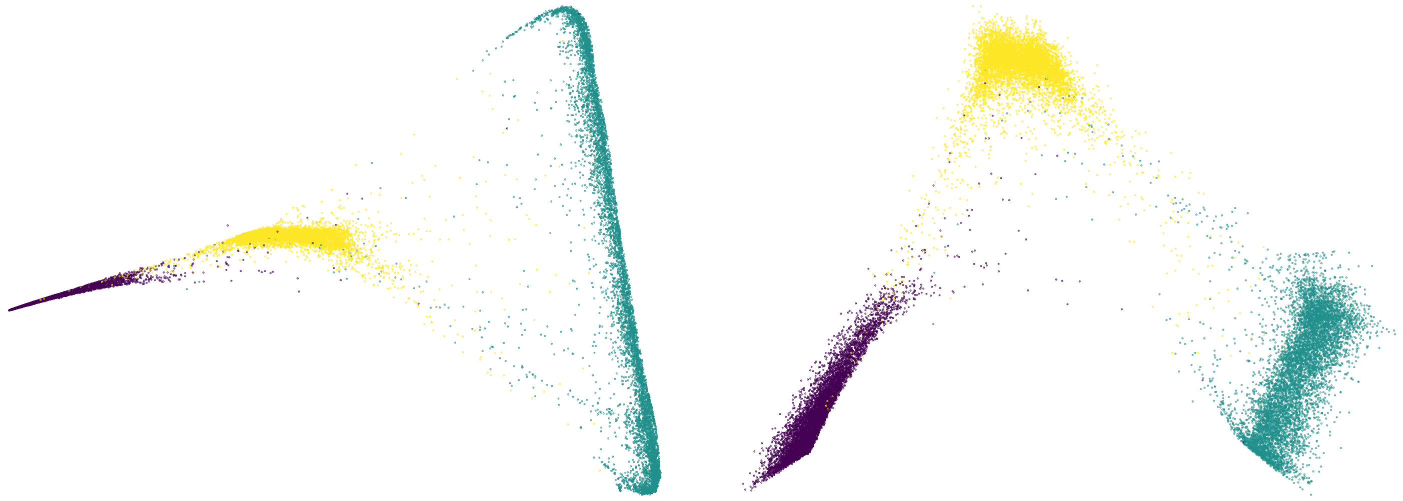
Spectral embeddings

The intuition behind the spectral embedding is encapsulated in the following simple result.

Proposition 9. *If $A \subset I_m$ is a disconected component of the graph, which means that $W(i,j) = 0$ for all $i \in A$ and $j \in I_m \setminus A$, then the indicator function of A, denoted u_A, satisfies*

$$Lu_A = (D - W)u_A = 0.$$

Spectral embedding of MNIST



(a) Unnormalized (b) Normalized

Figure 2: Example of spectral embeddings in the plane $k = 2$ of the 0, 1, and 2 digits of the MNIST dataset using the unnormalized $L = D - W$ and symmetric normalized $L = D^{-1/2}(D - W)D^{-1/2}$ graph Laplacians.

Spectral clustering with more than 2 classes

Suppose we want to cluster the graph into k clusters. Spectral clustering proceeds as follows:

1. Perform a spectral embedding of the graph into $\mathbb{R}^k$ (or sometimes $\mathbb{R}^d$ where $d \approx k$).

2. Run your favorite clustering algorithm in the embedding space $\mathbb{R}^d$, such as k-means.

k-nearest neighbor graph

The Gaussian weights

$$W(i,j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right),$$

are not always useful in practice, since the matrix W is dense (all entries are non-zero), and the connectivity length σ is the same across the whole graph.

It is more common to use a k-nearest neighbor graph. Let $d_{k,i}$ denote the Euclidean distance between x_i and its k^{th} nearest Euclidean neighboring point from $x_1, \ldots, x_m$. A k-nearest neighbor graph uses the weights

$$W(i,j) = \begin{cases} 1, & \text{if } \|x_i - x_j\| \leq \max\{d_{k,i}, d_{k,j}\} \\ 0, & \text{otherwise.} \end{cases}$$

The weights need not be binary, and can depend on $\|x_i - x_j\|$, similar to the Gaussian weights. The k-nearest neighbor graph weight matrix W is very sparse (most entries are zero), so it can be stored and computed with efficiently.

Exercises

The following proofs were omitted and are left to review as an exercise.

1. Proof of Theorem 2 (Convergence of k-means, Theorem 4.2 in lecture notes)
2. Proof of Lemma 6 (Properties of the graph Laplacian, Lemma 4.8 in lecture notes)
3. Proof of Proposition 9 (Spectral embeddings, Proposition 8.4 in lecture notes)

Spectral clustering in Python (.ipynb)