3.2.3 Spectral Clustering

https://www.youtube.com/watch?v=zkgm0i77jQ8

The last clustering method we will discuss is called spectral clustering. In contrast to the previous two methods, we require data in the form of a weighted graph for this method. However, it is capable of clustering significantly more complex datasets (e.g. two concentric circles)
A weighted graph is a tuple $G = (V, E, W)$ with a set of vertices V, a set of edges $E \subset V \times V$, and a weight function $W : E \to [0, \infty]$. We use the data points as the vertex set, i.e. $V = \{x_i\}_{i=1}^N \subset \mathbb{R}^d$ (**Explanation** the vertices are the data points in $\mathbb{R}^d$, meaning each data point is a node in the graph). Edges are drawn between "similar" data points. For this, we choose a weight function of the form $W(x, y) := \eta(||x - y||)$ which takes large values when $x, y \in V$ are similar (i.e., $||x - y||$ is small) and small values when they are dissimilar.

Explanation: If $||x - y||$ (Euclidian distance) is small, the similarity should be **large**, meaning the points are closely related.
If $||x - y||$ is **large**, the similarity should be **small**, meaning the points are far apart and less related
Popular choices are $\eta(s) = 1_{0 \leq s \leq \epsilon}$ for $\epsilon > 0$

Explanation: This means an edge is drawn between two points if their distance is within ϵ or $\eta(s) = \exp(-s^2/\sigma^2)$ for $\sigma > 0$

Example
If $\epsilon = 3$ and $||x - y|| = 2$ then $W(x, y) = \eta(2)$ and $\eta(2) = 1$ because $0 \leq 1 \leq 3$

Explanation: This function smoothly decreases similarity as distance increases. σ controls the decay; larger σ means more points are considered similar. We can define the edge set as $E = \{(x, y) \in V \times V | W(x, y) > 0\}$.

Explanation: This means that an edge exists between two points if the weight is greater than zero.
Since $W(x, y) = W(y, x)$, the resulting graph is undirected or symmetric, i.e., $(x, y) \in E \iff (y, x) \in E$. We can identify the graph G with its adjacency or weight matrix $W \in \mathbb{R}^{N \times N}$, where $W_{ij} := W(x_i, x_j)$ for $i, j = 1, \ldots, N$. This matrix is also symmetric. The interesting case for clustering is when $(x_{i_1}, x_{i_2}, \ldots, x_{i_m})$ (this represents a sequence of data points (or path in the graph starting from x_{i_1})) with $m \in \mathbb{N}$ and $i_j \in \{1, \ldots, N\}$ (this represents an index in the dataset) for $j = 1, \ldots, m$ such that $x_{i_1} = x, x_{i_m} = y$ (this means that the first point

in the sequence is x and the last point is y indicating a connection between them through intermediate points), and also $(x_{i_j}, x_{i_{j+1}}) \in E$ for all $j = 1, \ldots, m-1$

Explanation
$(x_{i_j}, x_{i_{j+1}}) \in E$ ensures that every consecutive (one after another) pair of points in the sequence is directly connected by an edge in the graph

Example
If we are given $x_{i_1} = x_2$ and $x_{i_m} = x_5$ it means that we are checking if there's a path from x_2 to x_5 through intermediate connections
The subscripts $i_1, i_2, \ldots, i_m$ indicate indices of data points forming a path in the graph

We first consider the case where we want to find $K = 2$ clusters. To do this, we want to assign a **value** to each graph **node**, with **similar nodes receiving similar values**.

Explanation: If two nodes are connected by a high-weight edge (indicating strong similarity) their assigned values should be close to each other.
We denote the collection of these values by $u \in \mathbb{R}^N$. A sensible assumption is that the mean of u is zero, i.e. $\sum_{i=1}^N u_i = 0$. If we define the vector $\mathbf{1} := (1, \ldots, 1) \in \mathbb{R}^N$, this is equivalent to $\langle u, \mathbf{1} \rangle = 0$ $(= u_1 \cdot 1 + u_2 \cdot 1 + \ldots = \sum_{i=1}^N u_i$, now we assumed above that this is $0)$
To enforce that similar nodes receive similar values, we use a minimization approach: We try to determine u such that

3.2.16

$$E(u) := \frac{1}{2} \sum_{i,j=1}^{N} W_{ij} |u_i - u_j|^2$$

is small, while also satisfying $\langle u, \mathbf{1} \rangle = 0$ (in geometric sense, if u is orthogonal to $\mathbf{1}$ then the dot product of both is 0). However, this problem has an uninteresting solution, namely $u = 0$.
Explanation
$E(u)$ measures the total squared difference between nodes x_i and x_j. The weights W_{ij} capture the similarity between nodes x_i and x_j. The goal is to minimize this function which means minimizing differences in u_i and u_j for strongly connected nodes (with large W_{ij})
To ensure that u takes positive and negative values (and not trivially 0), we require the Euclidian norm of the vector u equals 1, i.e., $||u||^2 := \sum_{i=1}^N u_i = 1$
(This is done to prevent trivial solution and force optimization to produce meaningful partition of nodes)
Thus we obtain the optimization problem (which we want to **solve**)

3.2.17

$$\min\{E(u) \mid u \in \mathbb{R}^N, \ ||u|| = 1, \ \langle u, 1 \rangle = 0\}$$

$$(= \min_{||u||_2=1} \langle u, Lu \rangle = \lambda_2)$$

Explanation
Minimize function $E(u)$ ensuring that similar nodes have similar values. Ensure that the sum of values is zero and ensure that the vector u has unit length (normalization constraint)

It turns out that the optimization problem 3.2.17 is equivalent to determining a certain eigenvector of the so-called graph Laplacian matrix. This is defined as

3.2.18

$$L := D - W \in \mathbb{R}^{N \times N}$$

where W is the **weight** matrix and D is the so-called degree matrix of the graph
This is a diagonal matrix with

3.2.19

$$D_{ij} := \begin{cases} \sum_{k=1}^{N} W_{ik} & \text{if } i = j, \\ 0 & \text{if } i \neq j \end{cases}$$

(diagonal elements D_{ii} of D are computed as the sum of the entries in the corresponding row i of the weight matrix W. This sum represents the **degree** (total connection strength) of node i)
Interpretation: You're summing across all nodes k that are connected to node i, which gives the **total weight** of all edges incident to node i. This forms the diagonal element D_{ii}, capturing the **degree (sum of edge weights)** of node i

Example
Consider following weight matrix W

$$W = \begin{bmatrix} 0 & 2 & 1 \\ 2 & 0 & 3 \\ 1 & 3 & 0 \end{bmatrix}$$

Row 1 sum: $0 + 2 + 1 = 3$
Row 2 sum: $2 + 0 + 3 = 5$
Row 3 sum $1 + 3 + 0 = 4$
So the degree matrix D is

$$D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

We can now compute the Laplacian matrix L:

$$L = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix} - \begin{bmatrix} 0 & 2 & 1 \\ 2 & 0 & 3 \\ 1 & 3 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & -2 & -1 \\ -2 & 5 & -3 \\ -1 & -3 & 4 \end{bmatrix}$$

Using the graph Laplacian matrix, we can express the function E as an inner product:

Proposition 3.2.2

It holds that $E(u) = \langle Lu, u \rangle$ for all $u \in \mathbb{R}^N$
Proof: It holds that $\langle Lu, u \rangle = \sum_{i=1}^{N} (Lu)_i u_i$ and also

$$(Lu)_i = \sum_{j=1}^{N} L_{ij} u_j$$

L is Laplacian matrix, which represents the **graph structure**, the vector $u \in \mathbb{R}^N$ contains values assigned to each node in the graph. The subscript i on $(Lu)_i$ means the i-th entry of the **vector** obtained by multiplying the Laplacian matrix L with the vector u.
Expanding this we get:

$$(Lu)_i = \sum_{j=1}^{N} L_{ij} u_j$$

This is the matrix vector multiplication of L and u, calculated as the dot product of the i-th row of L with the entire vector u. Each entry in the resulting vector Lu is a weighted sum of the components of u, where the weights come from the matrix L

Thus, we obtain

$$\langle Lu, u \rangle = \sum_{i=1}^{N} (Lu)_i u_i$$

Substituting $(Lu)_i$ in from above we get:

$$\langle Lu, u \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} L_{ij} u_j u_i$$

This equation represents a double summation over all pairs (i, j) where each term multiplies the weight L_{ij} with the corresponding values u_j and u_i.

Now we know that $L = D - W$ and $D_{ii} = \sum_{k=1}^{N} W_{ik}$ so we can replace L with $D - W$:

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} D_{ij} u_j u_i - \sum_{i=1}^{N} \sum_{j=1}^{N} W_{ij} u_j u_i$$

We also know that D is diagonal, so the off-diagonal terms vanish, leaving:

$$= \sum_{i=1}^{N} D_{ii} u_i^2 - \sum_{i=1}^{N} \sum_{j=1}^{N} W_{ij} u_j u_i$$

Exkurs
Now using the definition of $D_{ii} = \sum_{j=1}^{N} W_{ij}$ we can rewrite:

$$\sum_{i=1}^{N} D_{ii} u_i^2 = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} W_{ij} \right) u_i^2$$

Rearranging gets us to

$$\sum_{i=1}^{N} \sum_{j=1}^{N} W_{ij} u_i^2$$

Now **back to the actual proof**:

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} W_{ij} u_i^2 - \sum_{i=1}^{N} \sum_{j=1}^{N} W_{ij} u_j u_i$$

We can now factor out the common term W_{ij}. This is done by factoring out the common term in both terms of the sum (W_{ij}). The terms u_i^2 and $u_j u_i$ inside the parentheses are distinct components!

$$\sum_{i,j=1}^{N} = W_{ij} (u_i^2 - u_j u_i)$$

We factor our u_i from u_i^2

$$= \sum_{i,j=1}^{N} W_{ij} u_i (u_i - u_j)$$

Exkurs
Recall the squared difference identity: $|u_i - u_j|^2 = (u_i - u_j)^2 = u_i^2 - 2u_i u_j + u_j^2$
Now we observe that we don't have this form and only have $(u_i^2 - u_j) = u_i(u_i - u_j)$.
So we observe that the squared difference has an extra u_j^2 term, which is missing from our current expression. We add and subtract u_j^2 to balance the equation:

$$u_i^2 - u_i u_j = u_i^2 - 2u_i u_j + u_j^2 - u_j^2 + u_i u_j$$

(If you are still not sure: notice that $+u_j^2 - u_j^2$ would cancel out and $+u_i u_j$ would add to $-2u_i u_j$.

But we don't want to do this here. We want to rearrange)
We can now group the terms:

$$(u_i^2 - 2u_iu_j + u_j^2) + (-u_j^2 + u_iu_j)$$

In the first parantheses we recognize: $(u_i^2 - 2u_iu_j + u_j^2) = (u_i - u_j)^2$
Thus, the expression simplifies to $(u_i - u_j)^2 + (-u_j^2 + u_iu_j)$
Now we can factor out the remaining term $-u_j^2 + u_iu_j$

$$(-u_j^2 + u_iu_j) = u_j(-u_j + u_i) = u_j(u_i - u_j)$$

So we **finally** have

$$(u_i - u_j)^2 + u_j(u_i - u_j)$$

Now **back to the actual proof**:
Using the expansion in the sum

$$\sum_{i,j=1}^{N} W_{ij}u_i(u_i - u_j)$$

becomes

$$\sum_{i,j=1}^{N} W_{ij}((u_i - u_j)^2 + u_j(u_i - u_j))$$

Now we can split the summation into two separate sums:

$$\sum_{i,j=1}^{N} W_{ij}(u_i - u_j)^2 + \sum_{i,j=1} W_{ij}u_j(u_i - u_j)$$

This effectively is the same as

$$= \sum_{i,j=1}^{N} W_{ij}|u_i - u_j|^2 + \sum_{i,j=1}^{N} W_{ij}u_j(u_i - u_j)$$

The change to absolute value is simply a notational way to emphasize that we are always dealing with positive values here!
The first term measures the squared differences between values of u, weighted by the corresponding graph weight W_{ij}. The second term balances the contribution of the neighboring node's value u_j.
Now recall that $E(u)$ is defined as

$$E(u) := \frac{1}{2}\sum_{i,j=1}^{N} W_{ij}|u_i - u_j|^2$$

So we can just do $2E(u)$ to eliminate the $\frac{1}{2}$ and then get the equation

$$= 2E(u) - \sum_{i,j=1}^{N} W_{ij} u_i (u_i - u_j)$$

and since

$$\langle Lu, u \rangle = \sum_{i,j=1}^{N} W_{ij} u_i (u_i - u_j)$$

(if you're wondering why, just look at what we started with! It was $\langle Lu, u \rangle$ the whole time!!!) So we get

$$= 2E(u) - \langle Lu, u \rangle$$

where in the penultimate step, we used the symmetry $W_{ij} = W_{ji}$ of the weight matrix. By rearranging, we obtain $\langle Lu, u \rangle = E(u)$

END OF PROOF

Now we collect some important properties of the graph Laplacian matrix, which mostly follow from Proposition 3.2.2

Proposition 3.2.3

The graph Laplacian matrix L has the following properties

1. It is symmetric, i.e., $L^T = L$
   - Proof from exercise:
     - Let $u \neq 0$ and let the Graph be connected. $\Leftrightarrow \forall i, j = \{1, \ldots, N\}$, there exists $(i_1, \ldots, i_l)$ s.t. $i = i_1$ and $j = i_l$ and $\forall p \in \{1, \ldots, l-1\}, W_{seeexercise} > 0$
     - Let $(i_1, \ldots, i_l)$ be a path from x_1 to x_j for $j \in \{1, \ldots, N\}$
     - Then $0 \leq \frac{1}{2} \sum_{k=1}^{l}$
2. It is positive semidefinite, meaning all eigenvalues are non-negative
3. Its smallest eigenvalue is $\lambda = 0$, and if the graph is connected, all corresponding eigenvectors are of the form $u = c\mathbf{1}$ with $c \neq 0$
   - Consider the vector $\mathbf{1} = (1, 1, \ldots, 1) \in \mathbb{R}^N$
   - Applying L to $\mathbf{1}$
     - $L\mathbf{1} = (D - W)\mathbf{1} = D\mathbf{1} - W\mathbf{1}$
     - Since D is the degree matrix (and its diagonal! meaning if you sum a row, theres only one value $\neq 0$!), $D\mathbf{1}$ gives a vector which is the degree of the corresponding node

    - $W\mathbf{1}$ will give the same vector, because each row of W represents the weights of edges connected to a node, and summing those weights will give the same result (matrix-vector multiplication!)
- Therefore $L\mathbf{1} = 0$. This shows that $\mathbf{1}$ is an eigenvector corresponding to the eigenvalue $\lambda = 0$
- Now to the connected graph thing (meaning that there is a path between any pair of nodes in the garph)
    - For a connected graph, the corresponding eigenvector for the eigenvalue $\lambda = 0$ is always a **constant** vector $u = c\mathbf{1}$, where c is some constant (typically non-zero). This is because the Laplacian matrix measures the difference between a node and its neighbors. A constant vector, where all elements are the same, has no difference between any node and its neighbors, which means it corresponds to an eigenvalue of zero
        - The constant eigenvector $u = c\mathbf{1}$ corresponds to the fact that if all nodes are assigned the same value, the difference between connected nodes is zero, which minimizes the energy function and satisfies the eigenvalue equation $Lu = 0$

4. Let $u, v \in \mathbb{R}^N$ be eigenvectors corresponding to two different eigenvalues λ and μ. Then $\langle u, v \rangle = 0$
   - The Laplacian matrix L satisfies the eigenvalue problem $Lu = \lambda u$
   - Similarly for a different eigenvector v we have $Lv = \mu v$
   - Since the Laplacian is symmetric it's eigenvectors corresponding to distinct eigenvalues are orthogonal. (due to symmetry)
   - For example, consider this matrix where each column vector is an eigenvector
   -

$$\begin{bmatrix} -0.44 & -0.71 & 0.56 & 0.0 \\ -0.56 & 0.0 & -0.44 & -0.71 \\ -0.56 & 0.0 & -0.44 & 0.71 \\ -0.44 & 0.71 & 0.56 & 0 \end{bmatrix}$$

   - and consider this matrix where each value along the diagonal is an eigenvalue
   -

$$\begin{bmatrix} 2.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1.6 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

   - Now if you you take any different eigenvalue λ and μ, $\langle u, v \rangle$ will be 0
   - Example:

-

$$\begin{pmatrix} -0.44 & -0.56 & -0.56 & -0.44 \end{pmatrix} \begin{pmatrix} -0.71 \\ 0.0 \\ 0.0 \\ 0.71 \end{pmatrix}$$

-

$$\langle u, v \rangle = (-0.44 \cdot -0.71) + (-0.56 \cdot 0.0) + (-0.56 \cdot 0.0) + (-0.44 \cdot 0.71)$$

-

$$= 0.3124 + 0 + 0 - 0.3124 = 0$$

From Proposition 3.2.3 we conclude that the first eigenvalue of L is equal to zero. For connected graphs, the dimension of the corresponding eigenspace is one, and thus the second eigenvalue of L is positive. It can be shown relatively easy that the minimum value in 3.2.17 is exactly the second eigenvalue. Furthermore, the corresponding eigenvectors are minimizers.

Thus, spectral clustering consists of determining a minimizer of 3.2.17, or equivalently, an eigenvector corresponding to the second eigenvalue of the graph Laplacian matrix L.

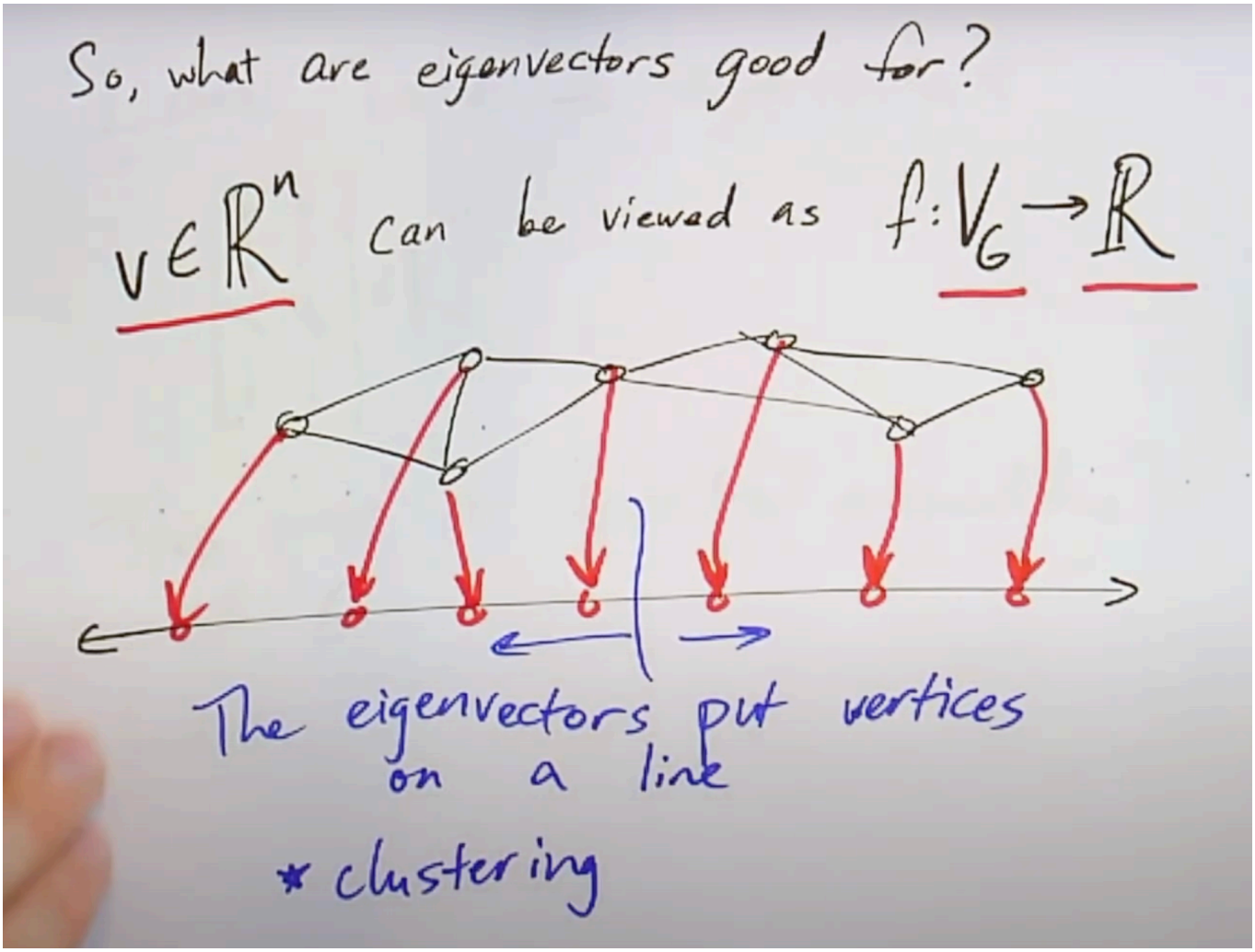
We obtain a trivial clustering for $K = 1$ by using the first eigenvector, e.g., $u^{(1)} = \mathbf{1}$. A clustering for $K = 2$ is obtained by calculating a second eigenvector $u^{(2)}$ (which is orthogonal to $u^{(1)}$, i.e. $\langle u^{(2)}, u^{(1)} \rangle = 0$). The cluster assignment for the i-th data point is then determined by the sign of $u_i^{(2)}$ (for example if node i has $u_i^{(2)} > 0$ it goes to Cluster A, else it goes to Cluster B)

To deal with larger numbers of clusters, we compute the so-called spectral embedding. This replaces the data matrix $X \in \mathbb{R}^{N \times d}$, where the i-th row of X is given by x_i, with the spectral embedding $Z \in \mathbb{R}^{N \times m}$, where the j-th column of Z contains the j-th eigenvector of L for $j = 1, \ldots, m$ with $m \leq N$. On these transformed data Z, we can then apply a clustering method such as K-means or EM again

!Missing Contents!

Need to know what y_i is computed for

Spectral clustering



Now from here you can measure the distance of those points?