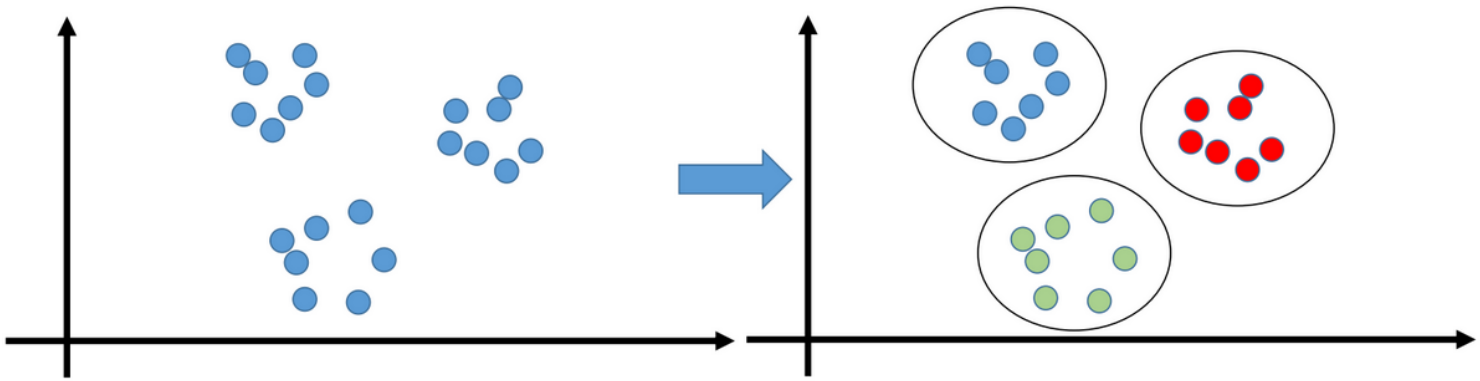
Lecture 17: Eigenvectors / Eigenvalues - Image Segmentation

July 24, 2025

Outline

Introduction

- In this lecture we will take a look at the application of eigenvalue problems in **image segmentation**. First we will given some definitions and discuss the graph Laplacian. Then we will make use of the graph Laplacian in spectral clustering.
- **Motivation:** Divide and conquer, say for image de-noising.
- **Spectral clustering** is a family of techniques that use the eigendecomposition of a matrix to identify clusters/groups of "similar" or related elements in a dataset. See for example the figure below.



Introduction

Segmentation tasks (i.e., identifying distinct parts of an image or shape) can rely on clustering.

- **Image segmentation** tries to group similar and nearby pixels.
- **Shape segmentation** tries to identify distinct parts of an object.

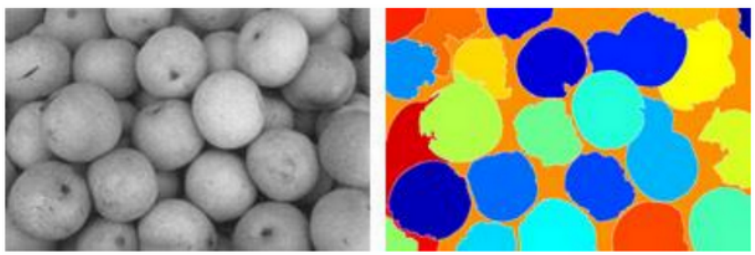
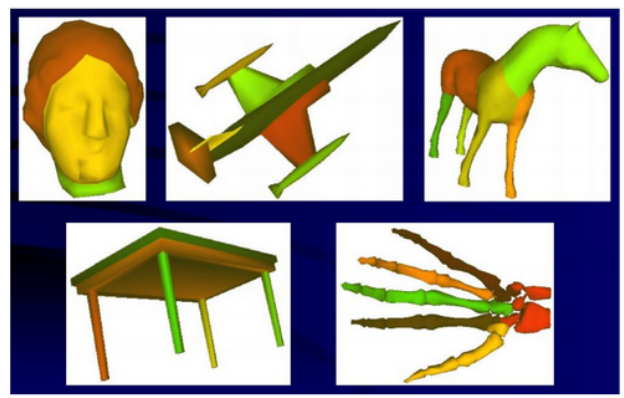


Image Segmentation



Shape Segmentation

Definitions

Consider an undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is a set of vertices and $E = \{e_{ij}\}$ is a set of edges. That is, the edge between vertices v_i and v_j is denoted e_{ij}.

Definition 1.1

The graph G is a **weighted** *graph if each edge e_{ij} has an associated weight $w_{ij} \geq 0$. We denote by $W = w_{ij}$ the* **weighted adjacency matrix** *of the graph.*

Figure: Example graph $G(V, E)$ (left) and the same graph with weighted edges (right).

Figure 1 gives an example of an undirected graph and a weighted version of the graph.

Definitions

- What is the weight matrix W for this graph in Figure 1 (right)?
- The matrix W has zeros in entries (i, j) that do not have edges joining v_i to v_j.
- There are nonzeros equal to the weights for any (i, j) that does have an edge, thus

$$W = \begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 1 & 0 & 0 & 4 & 0 & 2 \\ 2 & 3 & 4 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}.$$

- This W is **different** from our earlier adjacency matrix, where diagonal entries could be non-zero.

Definitions

The definition of the vertex degree must now be altered to include the weights of edges.

Definition 1.2

The **degree** *of a vertex v_i is given by*

$$d_i = \sum_{j=1}^{n} w_{ij},$$

where $D = diag(d_i)$ is the **degree matrix**.

This definition of degree is different from our earlier definition of degree, from matrix re-ordering.

Definitions

- For the example in Figure 1 what is the degree of v_4?
- It is just the sum of the fourth column of W, so $\deg(v_4) = d_4 = 11$.
- You can do this for all the vertices and construct the degree matrix as

$$D = \begin{bmatrix} 4 & & & & & \\ & 4 & & & & \\ & & 7 & & & \\ & & & 11 & & \\ & & & & 2 & \\ & & & & & 2 \end{bmatrix}.$$

Definitions

The indicator vector is useful when working with a subset of the graph.

Definition 1.3

Given a subset $A \subset V$, we define the **indicator vector** $\mathbf{1}_A = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^T$ *such that*

$$x_i = \begin{cases} 1 \text{ if } v_i \in A, \\ 0 \text{ if } v_i \notin A. \end{cases}$$

Definition 1.4

Given two subsets A, B, we define $W(A, B)$ to be the total weight of all the edges starting in A and ending in B, i.e.,

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}.$$

Definitions

For example, consider the two subsets of vertices, $A = \{v_1, v_2, v_6\}$ and $B = \{v_3, v_4, v_5\}$, in Figure 1. What are the indicator vectors $\mathbf{1}_A$ and $\mathbf{1}_B$? They are given by

$$\mathbf{1}_A = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{1}_B = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

What is $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$ for this example? We have

$$W(A, B) = w_{13} + w_{14} + w_{15} + w_{23} + w_{24} + w_{25} + w_{63} + w_{64} + w_{65} = 8.$$

Definitions

We will consider two ways to measure the **size** of a subset $A \subset V$.

Definition 1.5

This size of a subset is can be defined in terms of the number of vertices

$$|A| = \textit{number of vertices in } A,$$

or the degrees of the vertices

$$\textit{vol}(A) = \sum_{i \in A} d_i = \textit{sum of (weighted) degrees of vertices in } A.$$

For example, consider again the graph in Figure 1 and the two subsets of vertices $A = \{v_1, v_2, v_6\}$ and $B = \{v_3, v_4, v_5\}$. We have that $|A| = 3$ and $|B| = 3$. Furthermore, we have that $\text{vol}(A) = 10$ and $\text{vol}(B) = 20$.

Graph Laplacians

The **graph Laplacian** is a generalization of our finite difference discrete Laplacian operator to arbitrary graphs. We will consider two variants: the **unnormalized** graph Laplacian

$$L = D - W,$$

and the **normalized** graph Laplacian

$$\hat{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}.$$

Graph Laplacians

For example, with the graph in Figure 1 we have

$$
\begin{aligned}
L &= D - W \\
&= \begin{bmatrix} 4 & & & & & \\ & 4 & & & & \\ & & 7 & & & \\ & & & 11 & & \\ & & & & 2 & \\ & & & & & 2 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 1 & 0 & 0 & 4 & 0 & 2 \\ 2 & 3 & 4 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 4 & -1 & -1 & -2 & 0 & 0 \\ -1 & 4 & 0 & -3 & 0 & 0 \\ -1 & 0 & 7 & -4 & 0 & -2 \\ -2 & -3 & -4 & 11 & -2 & 0 \\ 0 & 0 & 0 & -2 & 2 & 0 \\ 0 & 0 & -2 & 0 & 0 & 2 \end{bmatrix}.
\end{aligned}
$$

Graph Laplacians

- Note that the general matrix pattern is similar to the finite difference discrete Laplacian.
- The diagonal entries are all positive, while off-diagonals are negative.
- Moreover, the sum of every row in L is zero.

Unnormalized Graph Laplacian

The following theorem gives some properties of the unnormalized graph Laplacian.

Theorem 1

The unnormalized graph Laplacian L satisfies:

1. *For any vector x,*

$$x^T L x = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2,$$

2. *L is symmetric and positive* **semi***-definite,*
3. *L has n non-negative eigenvalues* $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n$,
4. *The smallest eigenvalue of L is 0, with corresponding eigenvector being the constant one vector* $\mathbf{1} = [1, 1, \ldots, 1]^T$.

Proof.

See Lecture Notes. □

Normalized Graph Laplacian

The next theorem gives properties of the normalized graph Laplacian.

Theorem 2

The normalized graph Laplacian $\hat{L}$ satisfies:

1. *For any vector x,*

$$x^T \hat{L} x = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2,$$

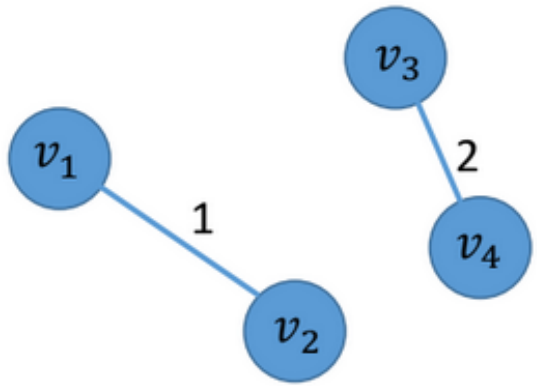
2. *$\hat{L}$ is symmetric and positive* **semi**-*definite,*
3. *$\hat{L}$ has n non-negative eigenvalues* $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n$,
4. *The smallest eigenvalue of $\hat{L}$ is 0 and the corresponding eigenvector is $D^{\frac{1}{2}}\mathbf{1}$.*

Proof.

Exercise.

Normalized Graph Laplacian

- The **multiplicity** k of the eigenvalue 0 for both L and $\hat{L}$ equals the **number of connected components** $A_1, \ldots, A_k$ in the graph.
- For example, the L and Λ for the following graph are



$$L = \begin{bmatrix} 1 & -1 & & \\ -1 & 1 & & \\ & & 2 & -2 \\ & & -2 & 2 \end{bmatrix}, \quad \text{and} \quad \Lambda = \{0, 0, 2, 4\}.$$

Normalized Graph Laplacian

Q & A

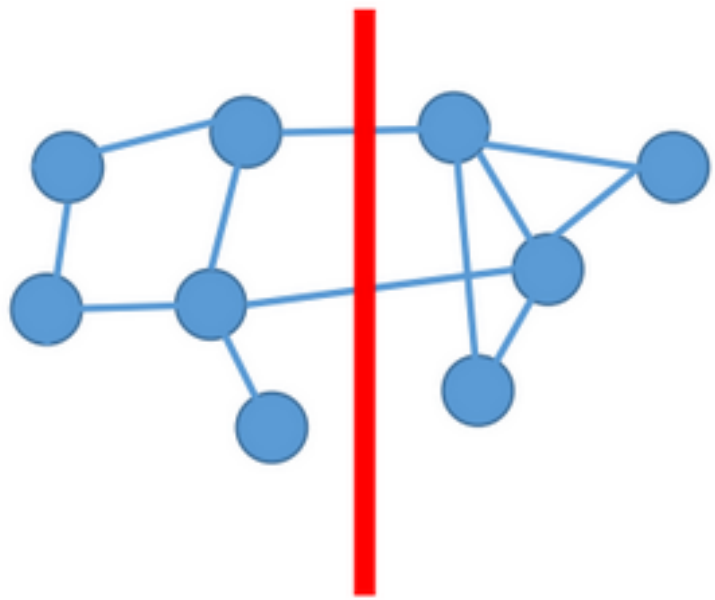
1. What do we do if we have an isolated vertex, whose degree will be 0, making it impossible to compute $D^{-\frac{1}{2}}$ later on?
   A: Segmentation is not needed for an isolated vertex: an isolated vertex is already segmented.

Normalized Graph Laplacian

- A final fact compares the graph Laplacian with the finite difference Laplacian.
- Suppose the graph is a 2D grid (e.g., representing an image) and we use weights $w_{ij} = 1$.
- Then, the unnormalized graph Laplacian L is (a scalar multiple of) the usual 2D finite difference Laplacian.
- We will now explore how graph Laplacians are used for clustering data.
- Consider the problem of finding minimally weighted **cuts** that divide the graph into parts.
- This generally leads to NP-hard problems, so we "relax" the problem, yielding our spectral clustering algorithms.

Normalized Graph Laplacian

The problem statement is as follows. Given a graph G with the weight matrix W, find a partition of G such that the edges between the partitions have very low weight. See the figure below for an example of what we want with $k = 2$ subsets.



e. g. , $k = 2$ subsets

Clustering using Graph Laplacians

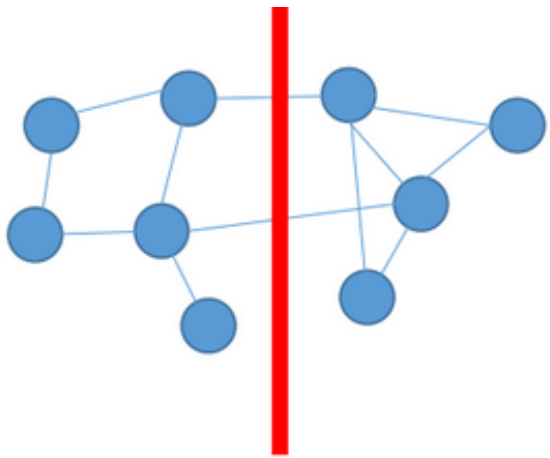
One approach is called **MinCut**, which finds a partition $A_1, \ldots, A_k$ that minimizes

$$\text{cut}(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A_i}).$$

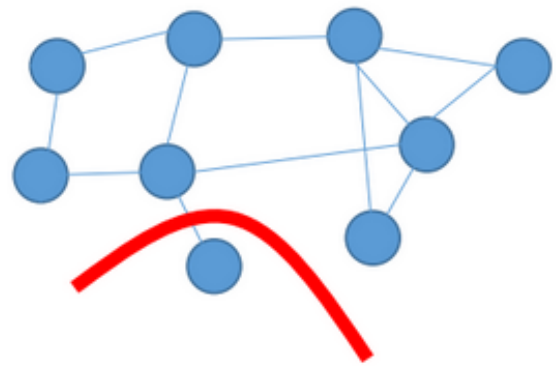
The notation $\overline{A_i}$ denotes the complement of A_i (i.e., vertices not in A_i).

Clustering using Graph Laplacians

The basic MinCut is fairly easy to solve, but it does not give **useful** results. The minimal solution often separates out individual vertices, rather than finding large subsets of nodes with low weight between them. For example, we would prefer the graph cut on the left below, but MinCut will normally compute the right cut.



Preferred cut



Computed cut

Better cuts in the graph would encourage the size of partitions to be larger, or more "balanced". Therefore, we should divide weights by the **size** of the subset ($|A_i|$ or $\text{vol}(A_i)$). This motivates the following definitions.

Clustering using Graph Laplacians

Definition 3.1

The RatioCut and Ncut (aka normalized cut) minimize the following, respectively:

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{|A_i|},$$

and

$$Ncut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{vol(A_i)}.$$

Recall that $|A_i|$ is the number of vertices in the set and $vol(A_i)$ is the sum of degrees of vertices in the set.

Clustering using Graph Laplacians - Relaxation of RatioCut via Graph Laplacian

- Sadly, minimizing for the RatioCut and Ncut is NP-hard.
- However, we can relax the minimization problem once we rewrite it in terms of the graph Laplacian.
- First, we consider the case of partitioning into 2 subsets, A and $\overline{A}$, subject to

$$\min_{A} \text{RatioCut}(A, \overline{A}).$$

- We can rewrite this in terms of the graph Laplacian as follows.

Clustering using Graph Laplacians - Relaxation of RatioCut via Graph Laplacian

Given a subset $A \subset V$, define $x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^T$, where

$$x_i = \begin{cases} +\sqrt{\frac{|\overline{A}|}{|A|}} \text{ if } v_i \in A, \\ -\sqrt{\frac{|A|}{|\overline{A}|}} \text{ if } v_i \in \overline{A}. \end{cases}$$

We can show the following three results to rewrite the minimization problem:

1. $x^T L x = |V| \cdot \text{RatioCut}(A, \overline{A})$,
2. $\sum_{i=1}^{n} x_i = 0$, i.e., $x^T \mathbf{1} = 0$,
3. $\|x\|^2 = n$.

Proof.

See Lecture Notes. □

Clustering using Graph Laplacians - Relaxation of RatioCut via Graph Laplacian

So the minimization problem becomes

$$\min_{A \subset V} x^T L x,$$
$$\text{subject to } x \perp \mathbf{1} \text{ and } \|x\| = \sqrt{n}.$$

But this minimization problem is still discrete and NP-hard! The vertices are strictly only in A or $\overline{A}$. However, we can **relax** the problem by allowing x to consist of arbitrary **real** numbers

$$\min_{x \in \mathbb{R}^n} x^T L x,$$
$$\text{subject to } x \perp \mathbf{1} \text{ and } \|x\| = \sqrt{n}.$$

The solution to this relaxed minimization problem turns out to be the eigenvector, x, of L corresponding to the 2^{nd} smallest eigenvalue (aka "Fielder vector").

Clustering using Graph Laplacians - Relaxation of RatioCut via Graph Laplacian

We can recover the separation into 2 clusters by thresholding x

$$v_i \in A \text{ if } x_i \geq 0,$$

$$v_i \in \overline{A} \text{ if } x_i < 0,$$

where x_i are the components of the Fielder vector.

Clustering using Graph Laplacians - Relaxation of Ncut via Graph Laplacian

Applying the same idea for Ncut, but with a different measure of set size, we start from

$$\min_A \mathrm{Ncut}(A, \overline{A}).$$

We can rewrite this using the normalized graph Laplacian. Given a subset $A \subset V$, define $x = \{x_1, \ldots, x_n\}$ where

$$x_i = \begin{cases} +\sqrt{\frac{\mathrm{vol}(\overline{A})}{\mathrm{vol}(A)}} & \text{if } v_i \in A, \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\overline{A})}} & \text{if } v_i \in \overline{A}. \end{cases}$$

Clustering using Graph Laplacians - Relaxation of Ncut via Graph Laplacian

We can then show that

1. $x^T L x = \text{vol}(V) \cdot \text{Ncut}(A, \overline{A})$,
2. $\sum_{i=1}^{n} d_i x_i = 0$, i.e., $(Dx)^T \mathbf{1} = 0$,
3. $x^T D x = \text{vol}(V)$.

So the minimization problem becomes

$$\min_{A \subset V} \; x^T L x, \text{ subject to } Dx \perp \mathbf{1} \text{ and } x^T D x = \text{vol}(V).$$

Clustering using Graph Laplacians - Relaxation of Ncut via Graph Laplacian

Again, this is still discrete and NP-hard to minimize. However, when we relax the minimization problem we instead solve

$$\min_{x \in \mathbb{R}^n} \; x^T L x, \text{ subject to } Dx \perp \mathbf{1} \text{ and } x^T D x = \text{vol}(V).$$

Defining $y = D^{\frac{1}{2}} x$, the relaxed problem becomes

$$\min_{y \in \mathbb{R}^n} \; y^T \hat{L} y, \text{ subject to } y \perp D^{\frac{1}{2}} \mathbf{1} \text{ and } \|y\|^2 = \text{vol}(V).$$

(Note: We use the same sign convention for the co-ordinates y_j of y that we used for the components x_j of x earlier.)
The solution again becomes the Fielder vector, but for $\hat{L}$ instead.
So we threshold y_i at zero in order to determine the two clusters.
Notation:

- $|V|$ is un-normed.
- $\text{vol}(V)$ is normed, using the weights.

K-means Clustering

- The above works for 2 clusters, but what about $k > 2$ clusters?
- For more clusters we can not simply threshold to zero, since we have more than 2 groups.
- Instead, we will make use of **k-means clustering** on data drawn from several eigenvectors.
- First, let us consider the basic k-means algorithm.
- Given a set of n data points/vectors $\{ p_j \}$, find the partition of the points $A_1, \ldots, A_k$ such that each point is assigned to the set whose mean μ_i is closest to it.
- K-means aims to solve the problem

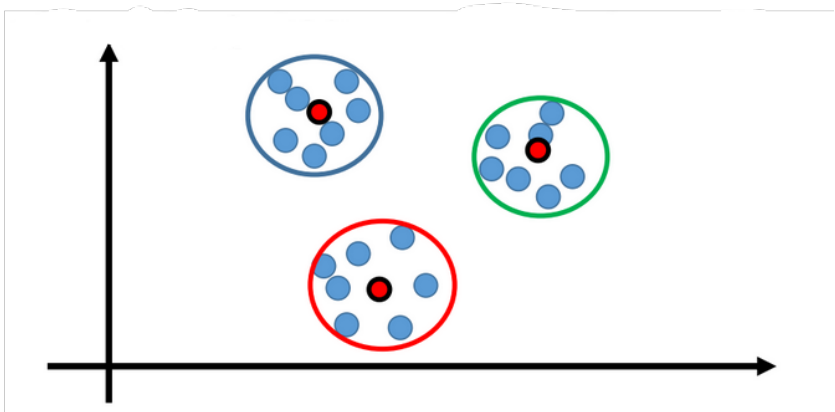
$$\min_{A_i} \sum_{i=1}^{k} \sum_{p_j \in A_i} \|p_j - \mu_i\|^2.$$

K-means Clustering

There are two factors at play in this minimization problem:

- assignment of points to sets,
- distance of each point to the mean of its set.

Consider the example in the figure below. Given blue (2D) data points try to find $k = 3$ (how to choose k: later) means **and** an assignment of points to the corresponding 3 clusters.



We would expect something like the red points as the means of the 3 clusters. The data points would belong to the cluster whose mean is closest to them (as shown with blue, green, and red encompassing circles).

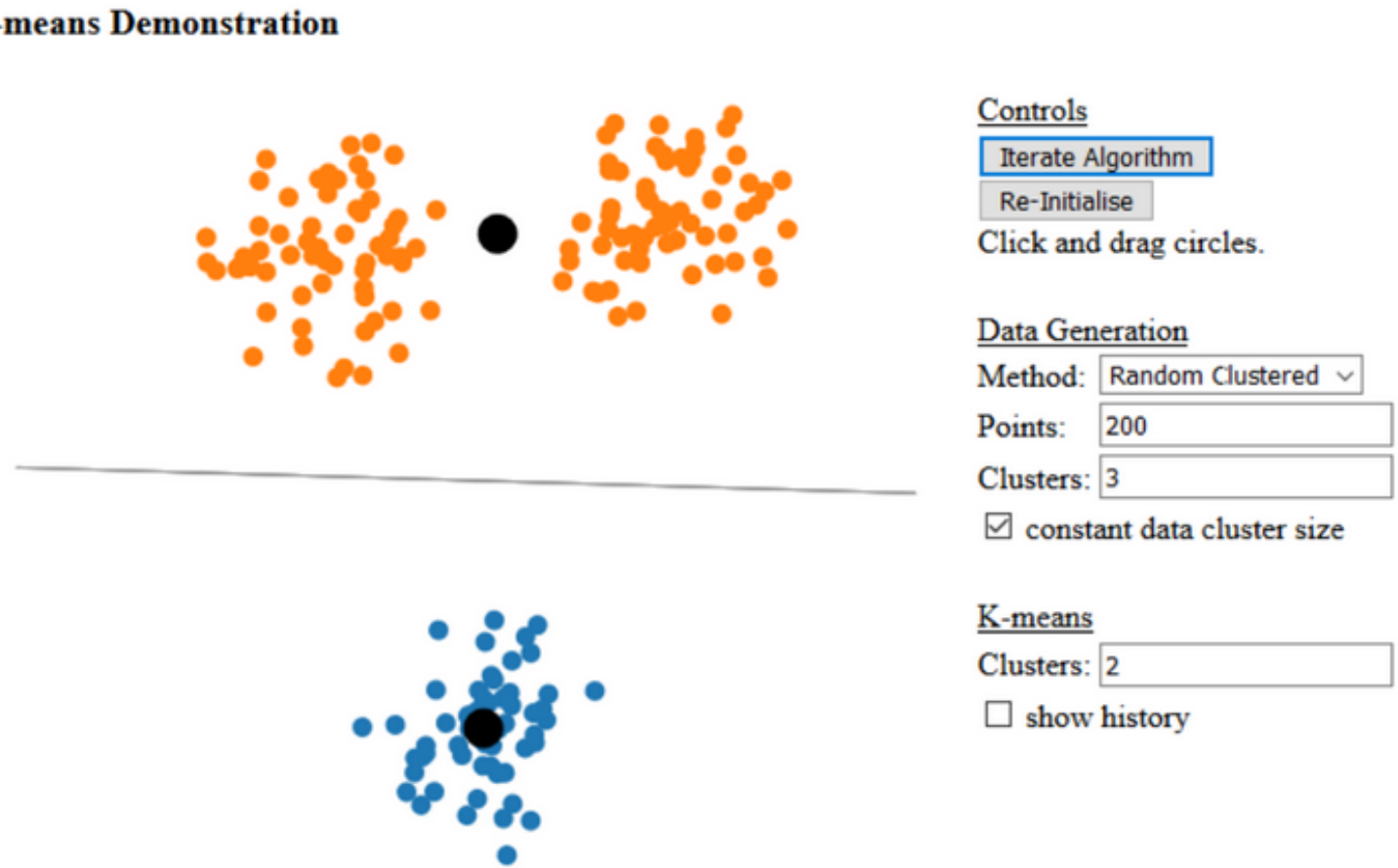
K-means Clustering

The **k-means algorithm** performs the following steps:

1. Start with some initial guesses for the k means $\{\mu_i\}$,
2. Assign each point p to the cluster A_i if p is closer to μ_i than any of the other k means,
3. Re-compute new means $\{\mu_i\}$ for all partitions $\{A_i\}$,
4. Repeat.

K-means Clustering

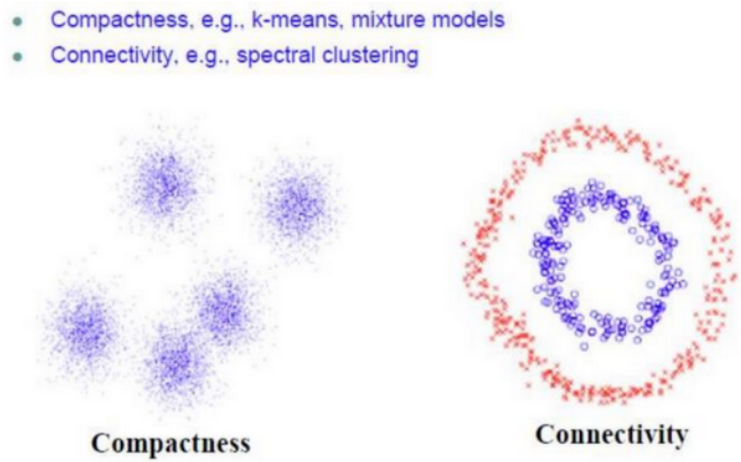
There is a nice interactive demo of k-means available online. Please visit `http://alekseynp.com/viz/k-means.html` to try it out!



Spectral Clustering: Cuts and K-means Together

If k-means can do clustering why do we not just apply it to our problem? **Spectral clustering** allows for:

- More general weights/measures of similarity (i.e., not just Euclidean distance),
- Non-convex clusters.



We can apply k-means for the $k = 2$ case instead of thresholding at zero, to assign points into the two clusters. Specifically, considering the entries of the eigenvector $\{x_i\}$ as n data points in $\mathbb{R}$, then apply k-means with $k = 2$. The advantage of this is that it can be extended to $k > 2$ clusters.

Spectral Clustering: Cuts and K-means Together

Intuitively, spectral clustering consists of the following (also depicted in Figure 2):

1. Interpret our input data as a graph and choose weights W to indicate our notions of similarity,
2. Use the eigenvectors of the graph Laplacian to convert vertices into data points in $\mathbb{R}^k$,
3. Apply k-means to cluster the points in Euclidean space ($\mathbb{R}^k$).

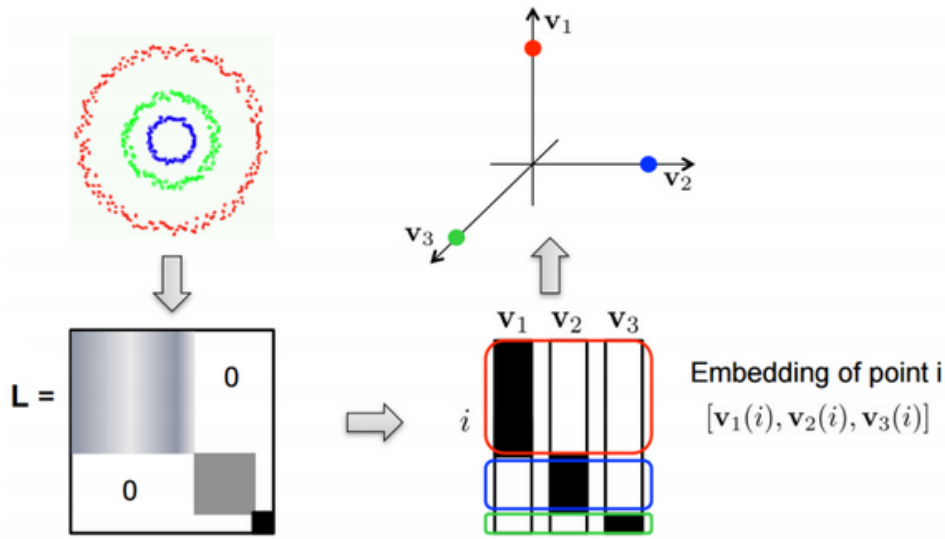


Figure: Steps of spectral clustering.

Spectral Clustering: Cuts and K-means Together

In more detail, the **unnormalized spectral clustering** algorithm is:

1. Construct the unnormalized graph Laplacian L,
2. Compute the first k eigenvectors $q_1, \ldots, q_k$ of L (corresponding to **smallest** magnitude eigenvalues),
3. Consider $Q_k = [q_1, \ldots, q_k]$. Let $p_i \in \mathbb{R}^k$ be the vector given by row i of Q_k (i.e., $p_i = Q_k(i, :)$ in Matlab notation),
4. For the resulting n points $\{p_i\}$ in $\mathbb{R}^k$, apply k-means to cluster them into k groups $\{A_1, \ldots, A_k\}$.

The **normalized** version of the **spectral clustering algorithm** requires two changes.

1. First, we use $\hat{L}$ instead of L.
2. Second, instead of p_i, we use normalized rows for points, $p_i^n = \frac{p_i}{\|p_i\|}$.

Spectral Clustering: Cuts and K-means Together

Why We Choose the p_i In $\mathbb{R}^k$:

- To create k clusters, we select the first k eigenvectors.
- We will be working in the k-dimensional subspace of $\mathbb{R}^n$ spanned by these k eigenvectors.
- It turns out to be more convenient to simply work in $\mathbb{R}^k$ itself.

Why We Choose the Smallest Magnitude Eigenvalues, Not the Largest:

- Idea: The unnormalized graph Laplacian, $L = D - W$, is already an "error".
- For segmentation purposes, we want the smallest "error" possible.

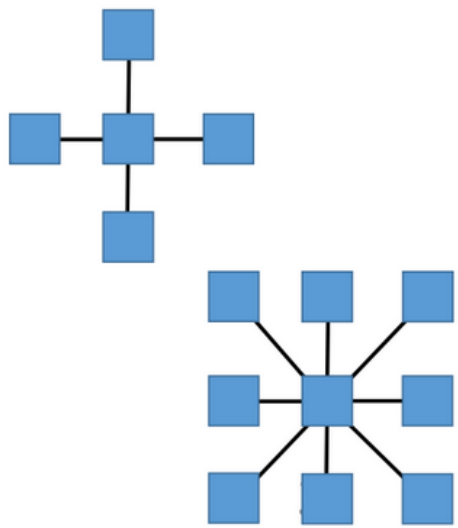
Spectral Clustering: Cuts and K-means Together - Choosing Weights W

- The choice of the weight matrix W is meant to measure similarity between vertices of the graph.
- This means W is **problem dependent**.
- Usually, we want non-zero weights only between a small set of local graph neighbours, (e.g., within graph distance 1 or 2).
- The more neighbours we include creates more non-zero entries in W.
- If we include fewer neighbours we create a sparser graph Laplacian (having a lower cost of eigendecomposition).

Spectral Clustering: Cuts and K-means Together - Choosing Weights W

For an image segmentation task we view pixels as graph vertices. We connect adjacent or nearby pixels with graph edges which form our graph. For example,

- including the 4 adjacent pixels gives W with non-zero structure similar to usual finite difference Laplacian,
- including the 4 diagonal neighbours also would give 8 neighbours, so W has at most 8 non-zeros per row.



Spectral Clustering: Cuts and K-means Together - Choosing Weights W

We will set w_{ij} to measure similarity between pixels i and j using two factors:

1. Euclidean distance between pixels i and j,
2. intensity difference between pixels i and j.

For $i \neq j$ we will use

$$w_{ij} = \left(e^{-\frac{\|x_i - x_j\|^2}{\sigma^2_{dist}}} \right) \left(e^{-\frac{|I_i - I_j|^2}{\sigma^2_{int}}} \right),$$

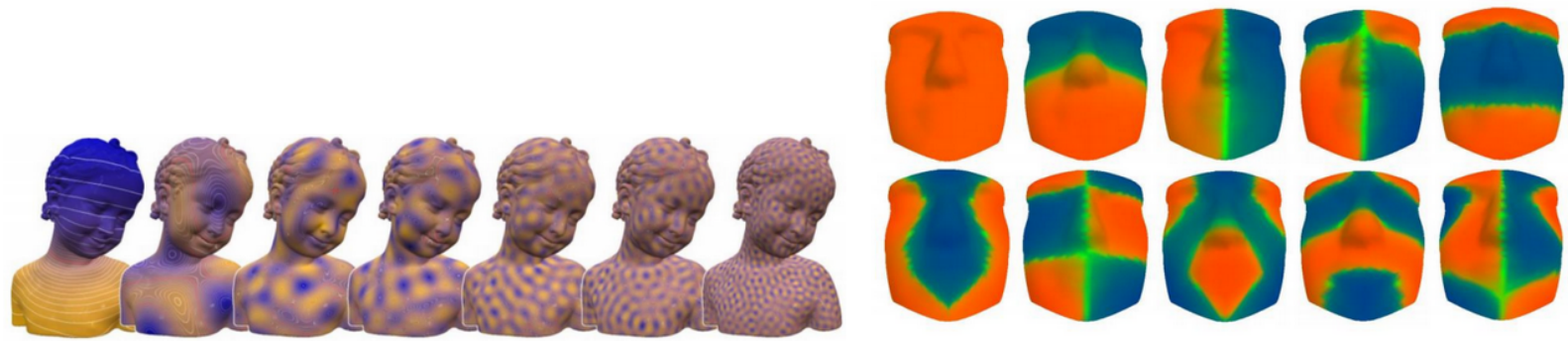
where pixel i is at position x_i with intensity I_i and likewise for pixel j. We define positions as $x_i = (r, c)$ if pixel i is at row r, column c. The parameters σ^2_{dist} and σ^2_{int} can be varied to adjust the relative importance of the terms.

Other Applications

Spectral approaches find many other applications in the field of graphics processing.

Other Applications - Geometric Mesh Processing

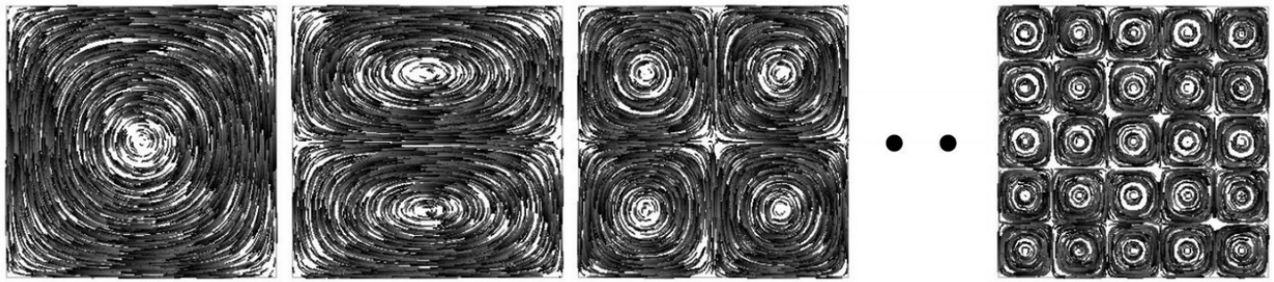
The following are visualizations of several eigenvectors associated to Laplacians defined on 3D triangle meshes.



source: `https://www.cs.sfu.ca/~haoz/pubs/zhang_eg07star_spectral.pdf`.

Other Applications - Motion Analysis

Extracting dominant motion "modes" in solids or fluids for analysis and efficient simulation.



$\longrightarrow$ increasing eigenvalue magnitude

The above images show the basis of divergent-free fields that are eigenfunctions of the vector Laplacian. Basis fields have correspondence with spatial scales of vorticity. Their coefficients form a discrete spectrum. Another example of motion analysis involves vibrating membranes, see `https://en.wikipedia.org/wiki/Vibrations_of_a_circular_membrane`.
To learn more, check out "A Tutorial on Spectral Clustering", by Ulrike von Luxburg, `http://www.tml.cs.uni-tuebingen.de/team/luxburg/publications/Luxburg07_tutorial.pdf`.