Foundations of Data Science, Fall 2020

15. Clustering

Prof. Dan Olteanu

DaST Data•(Systems+Theory)

University of Zurich[UZH]

December 1, 2020

https://lms.uzh.ch/url/RepositoryEntry/16830890400

https://uzh.zoom.us/j/96690150974?pwd=cnZmMTduWUtCeWoxYW85Z3RMYnpTZz09

Outline

- Clustering objective function
- k-Means formulation for clustering
- Transforming input formats
  - Dissimilarities from Euclidean Embeddings
  - Multidimensional Scaling
- Hierarchical clustering
- Spectral clustering

How To Group or Cluster Articles?

England pushed towards Test defeat by India

France election: Socialists scramble to avoid split after Fillon win

Giants Add to the Winless Browns' Misery

Strictly Come Dancing: Ed Balls leaves programme

Trump Claims, With No Evidence, That 'Millions of People' Voted Illegally

Vive 'La Binoche', the reigning queen of French cinema

How To Group or Cluster Articles?

Sports	England pushed towards Test defeat by India
Politics	France election: Socialists scramble to avoid split after Fillon win
Sports	Giants Add to the Winless Browns' Misery
Film&TV	Strictly Come Dancing: Ed Balls leaves programme
Politics	Trump Claims, With No Evidence, That 'Millions of People' Voted Illegally
Film&TV	Vive 'La Binoche', the reigning queen of French cinema

How To Group or Cluster Articles?

England	England pushed towards Test defeat by India
France	France election: Socialists scramble to avoid split after Fillon win
USA	Giants Add to the Winless Browns' Misery
England	Strictly Come Dancing: Ed Balls leaves programme
USA	Trump Claims, With No Evidence, That 'Millions of People' Voted Illegally
France	Vive 'La Binoche', the reigning queen of French cinema

Clustering

Often data can be grouped together into subsets that are coherent. However, this grouping may be subjective. It is hard to define a general framework.

Two types of clustering algorithms

1. Feature-based - Points are represented as vectors in $\mathbb{R}^D$
2. (Dis)similarity-based - Only know pairwise (dis)similarities

Two types of clustering methods

1. Flat - Partition the data into k clusters
2. Hierarchical - Organise data as clusters, clusters of clusters, and so on

k-Means Formulation of Clustering

Partition-Based Clustering

Goal: Partition the data into subsets $C_1, \ldots, C_k$, where k is fixed in advance

Quality of a partition defined by

$$W(C) = \frac{1}{2} \sum_{j=1}^{k} \frac{1}{|C_j|} \sum_{i,i' \in C_j} d(\mathbf{x}_i, \mathbf{x}_{i'})$$

If we use $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$, then

$$W(C) = \sum_{j=1}^{k} \sum_{i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2^2$$

where $\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}_i$

Objective: minimise the sum of squares of distances to the mean within each cluster

The k-Means Objective

Minimise jointly over partitions $C_1, \ldots, C_k$ and $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$

$$W(C) = \sum_{j=1}^{k} \sum_{i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2^2$$

This problem is NP-hard even for $k = 2$ for points in $\mathbb{R}^D$

If we fix means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, finding a partition $(C_j)_{j=1}^k$ that minimises W is easy

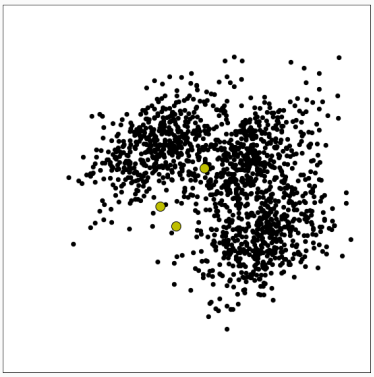
$$C_j = \{i \mid \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2 = \min_{j'} \|\mathbf{x}_i - \boldsymbol{\mu}_{j'}\|_2\}$$

If we fix the clusters $C_1, \ldots, C_k$, minimising W with respect to $(\boldsymbol{\mu}_j)_{j=1}^k$ is easy

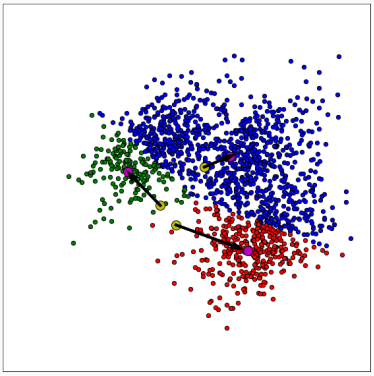
$$\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}_i$$

Alternating minimisation: Iteratively run these assignment and update steps

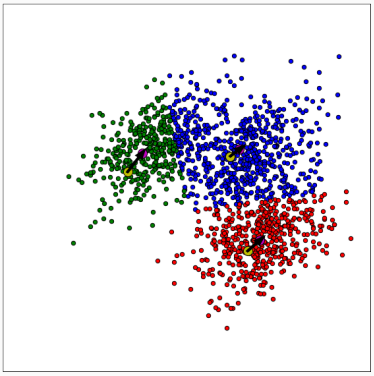
k-Means in Action



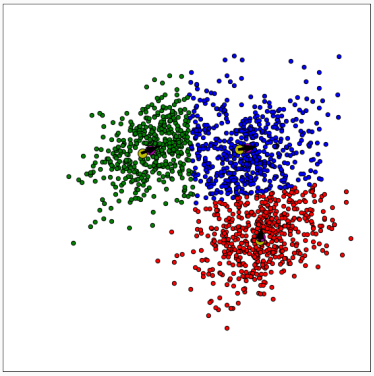
k-Means in Action



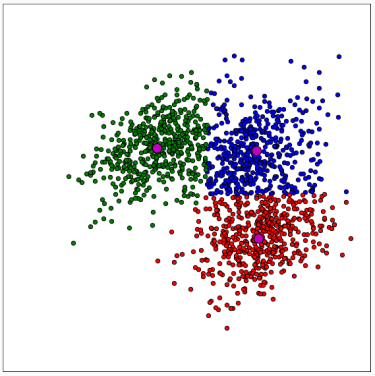
k-Means in Action



k-Means in Action

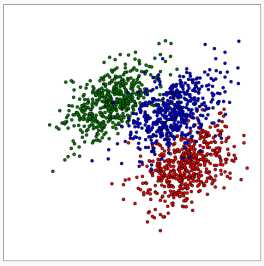


k-Means in Action

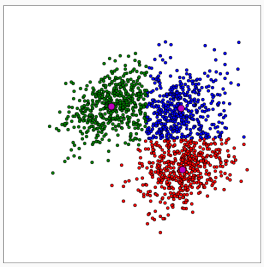


Ground Truth Clusters vs k-Means Clusters

Ground Truth Clusters



k-Means Clusters (k = 3)



The k-Means Algorithm

1. Initialise means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ "randomly"
2. Repeat until convergence:
   a. Construct clusters $C_1, \ldots, C_k$ by assigning the data to clusters represented by their means:
   $$C_j = \{i \mid j = \operatorname*{argmin}_{j'} \|\mathbf{x}_i - \boldsymbol{\mu}_{j'}\|_2^2\}$$
   b. Update means using the current cluster assignments:
   $$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}_i$$

Note 1: Ties can be broken arbitrarily

Note 2: Choosing k random datapoints to be the initial k-means is a good idea

Note 3: This algorithm is also called the Lloyd's algorithm

Convergence of k-Means

Does the algorithm always converge?

Yes, because the objective function W decreases every time a new partition is used; there are only finitely many partitions

$$W(C) = \sum_{j=1}^{k} \sum_{i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2^2$$

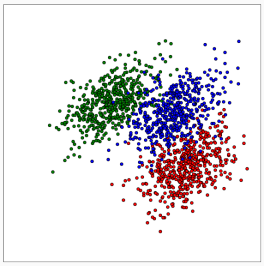
Convergence may be very slow in the worst-case, but typically fast on real-world instances (Exercise: how many iterations are necessary for convergence?)

Convergence is likely to a local minimum.

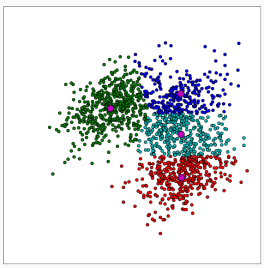
Run multiple times with random initialisation.

Which Value for k?

Ground Truth Clusters

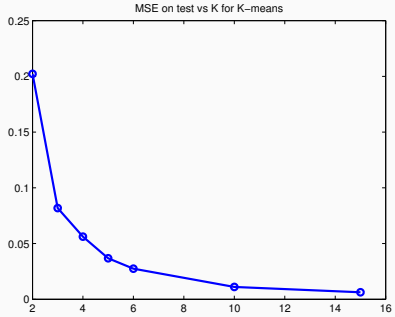


k-Means Clusters ($k = 4$)



Selecting the right k value is not easy: plot W against k and identify an "elbow"

Choosing the number of clusters k



- Larger k lowers the objective though might give poorer clustering
- Choose suitable k by identifying a "kink" or "elbow" in the curve

(Source: Kevin Murphy, Chap 11)

Beyond k-Means

k-medoids

- Actual data points as cluster means and not Euclidean averages
- Any distance between points beyond the squared Euclidean distance

  In particular, we can use the Euclidean distance or any other ℓ_p norm

k-center

- Objective: maximum over all dissimilarities between data and anchor

  For k-means: sum of all the dissimilarities in each cluster
- Anchors or centres take the role of means

Further clustering methods: hierarchical clustering, spectral clustering, etc.

Transforming Input Formats

Input Formats

Clustering algorithms may work on different data formats

- Data given in Euclidean space
- Data given as matrix of (dis)similarities

Question 1: How to transform Euclidean distance to dissimilarity?

Question 2: How to transform dissimilarity to Euclidean distance?

Dissimilarity

- Weighted dissimilarity between (real-valued) attributes

$$d(\mathbf{x}, \mathbf{x}') = f\left(\sum_{i=1}^{D} w_i d_i(x_i, x_i')\right)$$

- Simplest setting: $w_i = 1$ and $d_i(x_i, x_i') = (x_i - x_i')^2$ and $f(z) = \sqrt{z}$

  This corresponds to the Euclidean distance
- Weights allow us to emphasise features differently
- If features are ordinal or categorical then define distance suitably

  Natural choice: $d_i(x_i, x_i') = 1$ if $x_i = x_i'$ and 0 otherwise

Multidimensional Scaling (MDS)

It may be easier to define (dis)similarity between objects than embed them in Euclidean space:

- DNA sequences: Use Hamming distance as measure of dissimilarity
- Text data: Use cosine kernel as measure of similarity

Algorithms such as k-means require however points to be in Euclidean space

We need to transform (dis)similarity measures into Euclidean space

Multidimensional Scaling gives a way to find an embedding of the data in Euclidean space that (approximately) respects the original distance/similarity

Recovering the Data Points from Dissimilarity Matrix

Assume ideal case: We are given the dissimilarity matrix $\mathbf{D}$ with pairwise Euclidean distances of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$

$$D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \text{ for } i, j \in [N]$$

Can we reconstruct $\mathbf{x}_1, \ldots, \mathbf{x}_N$ from $\mathbf{D}$?

Distances are preserved under translation, rotation, reflection, etc.

We cannot recover $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ exactly

We can aim to determine $\mathbf{X}$ up to these transformations

Recovering the Data Points from Dissimilarity Matrix

If D_{ij} is the distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$, then

$$\begin{aligned} D_{ij} &= \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \\ &= \mathbf{x}_i^\mathsf{T}\mathbf{x}_i - 2\mathbf{x}_i^\mathsf{T}\mathbf{x}_j + \mathbf{x}_j^\mathsf{T}\mathbf{x}_j \\ &= M_{ii} - 2M_{ij} + M_{jj} \end{aligned}$$

Here $\mathbf{M} = \mathbf{X}\mathbf{X}^\mathsf{T}$ is the $N \times N$ matrix of dot products: $M_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$

Assuming $\sum_i \mathbf{x}_i = \mathbf{0}$, $\mathbf{M}$ can be uniquely recovered from $\mathbf{D}$

Otherwise, $\mathbf{M}$ can be recovered from $\mathbf{D}$ up to translation

To obtain points $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N$ from $\mathbf{M}$, we use Singular Value Decomposition of $\mathbf{M}$

Singular Value Decomposition (SVD)

The Singular Value Decomposition of a matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ is $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$, where

- Thin SVD:
  - $\mathbf{U} \in \mathbb{R}^{N \times D}$ and $\mathbf{V} \in \mathbb{R}^{D \times D}$ are orthonormal matrices
  - $\mathbf{\Sigma} \in \mathbb{R}^{D \times D}$ is diagonal with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_D \geq 0$
  - $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I}_D$
- Full SVD: The matrices $\mathbf{U}$ and $\mathbf{\Sigma}$ are larger
  - $\mathbf{U} \in \mathbb{R}^{N \times N}$
  - $\mathbf{\Sigma} \in \mathbb{R}^{N \times D}$
  - $\mathbf{V} \in \mathbb{R}^{D \times D}$

Eigendecomposition: $\mathbf{M}$ is square and $\mathbf{V} = \mathbf{U}$: $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\mathsf{T}$

Recovering the Data Points from Dissimilarity Matrix using SVD

$\mathbf{M}$ is square, symmetric and positive semi-definite

Starting from $\mathbf{M}$, we can reconstruct $\tilde{\mathbf{X}}$ using the eigendecomposition of $\mathbf{M}$

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\mathsf{T}$$

$\mathbf{\Sigma}$ has non-negative diagonal entries, so we can take its square root

By letting $\tilde{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}^{1/2}$, we obtain

$$\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{T} = \mathbf{U}\mathbf{\Sigma}^{1/2}(\mathbf{U}\mathbf{\Sigma}^{1/2})^\mathsf{T} = \mathbf{U}\mathbf{\Sigma}^{1/2}\mathbf{\Sigma}^{1/2}\mathbf{U}^\mathsf{T} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\mathsf{T} = \mathbf{M}$$

We thus find the points $\tilde{\mathbf{x}}_i$ as rows of $\mathbf{U}\mathbf{\Sigma}^{1/2}$.

General recipe:

- Use Mercer kernel to define the similarity
- The matrix $\mathbf{M}$ is positive semi-definite and the above derivation works

Recovering the Data Points from Dissimilarity Matrix in General

Solve the optimisation problem with (non-convex) objective:

$$\underset{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N}{\operatorname{argmin}} \sum_{i \neq j} \left(\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2 - D_{ij} \right)^2$$

This objective function is called the stress function.

It gives the degree to which it is not possible to find a Euclidean embedding for $\mathbf{D}$

No closed-form solution possible. Local optimum via gradient descent

Hierarchical Clustering

Hierarchical Clustering

Task: Build 3 clusters out of four news articles (Physics, Maths, Ski, Football)

- Possible clustering: "maths", "physics", "sports"
- Another clustering: "science", "ski", "football"
- We want to emphasise closer relationships: Maths-Physics and Ski-Football
- We could first build smaller clusters and then cluster of clusters

Hierarchical structured data exists all around us

- Measurements of different species and individuals within species
- Top-level and low-level categories in news articles
- Country, canton, town level data

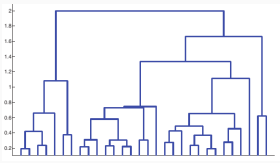
Hierarchical Clustering

Two Algorithmic Strategies for Clustering

- Agglomerative: Bottom-up, clusters formed by merging smaller clusters

  Most popular agglomerative algorithms: Linkage algorithms
- Divisive: Top-down, clusters formed by splitting larger clusters

Visualise the clustering as a dendrogram or binary tree



Cutting the dendrogram at some level gives a partition of data

Measuring Dissimilarity at Cluster Level

To find hierarchical clusters we need to define dissimilarity at cluster level, not just at datapoints

Suppose we have dissimilarity at datapoint level, *e.g.*, $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$

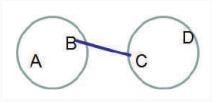
Different ways to define dissimilarity at cluster level, say C and C'

- Single Linkage
$$D(C, C') = \min_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$
- Complete Linkage
$$D(C, C') = \max_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$
- Average Linkage
$$D(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

Measuring Dissimilarity at Cluster Level

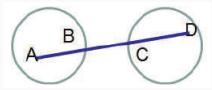
Single Linkage

$$D(C, C') = \min_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$



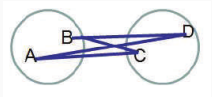
Complete Linkage

$$D(C, C') = \max_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$



Average Linkage

$$D(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

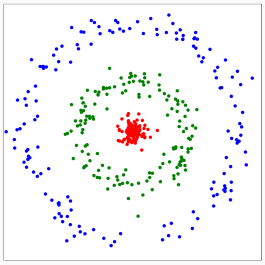


Linkage-based Clustering Algorithm

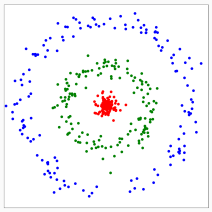
1. Initialise clusters as singletons $C_i = \{i\}$
2. Initialise clusters available for merging $S = \{1, \ldots, N\}$
3. Repeat
   a. Pick 2 most similar clusters, $(j, k) = \operatorname{argmin}_{j,k \in S} D(j, k)$
   b. Let $C_l = C_j \cup C_k$ for some new index l
   c. If $C_l = \{1, \ldots, N\}$, break;
   d. Update $S := (S \setminus \{j, k\}) \cup \{l\}$
   e. Update $D(i, l)$ for all $i \in S$ (using desired linkage property)

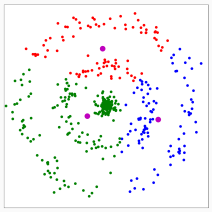
Spectral Clustering

How to Find Non-Convex Clusters?



Limitations of k-Means





k-means will typically form clusters that are spherical, elliptical, convex

Spectral clustering is a (related) alternative that often works better

Spectral Clustering

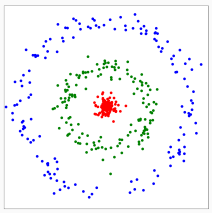
- Construct K-nearest neighbour graph from data
  - One node for every point in dataset
  - (i, j) is an edge if either i is among the K nearest neighbours of j or vice versa
  - The weight of edge (i, j), if it exists, is given by similarity measure $s_{i,j}$

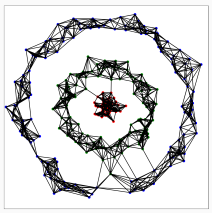
$$s_{i,j} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma)$$

- Use graph partitioning algorithms, one particular approach:
  - Use eigenvectors of the Laplacian matrix of the graph as new non-linear features

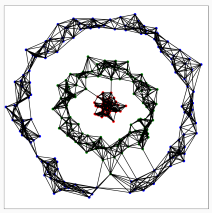
    We effectively perform non-linear dimensionality reduction
  - Apply k-means in the lower dimension

Spectral Clustering: Graph Construction





Spectral Clustering: How to Partition?



Use graph partitioning algorithms

Mincut can give bad cuts (only one node on one side of the cut)

Multi-way cuts, balanced cuts, are typically NP-hard to compute

Relaxations of these problems give eigenvectors of Laplacian

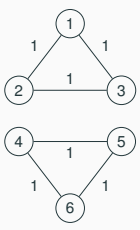
$\mathbf{W}$ is the weighted adjacency matrix

$\mathbf{D}$ is (diagonal) degree matrix: $D_{ii} = \sum_j W_{ij}$

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

Normalised Laplacian: $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$

Example 1: Computing the Laplacian



Suppose all edge weights are 1 (0 for missing edges)

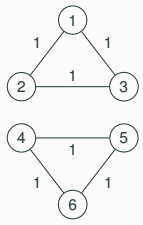
The weighted adjacency matrix, the degree matrix and the Laplacian are given by

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

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$\mathbf{L} = \mathbf{D} - \mathbf{W} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

Example 1: Computing the Eigenvectors of the Laplacian



Suppose all edge weights are 1 (0 for missing edges)

Let us consider some eigenvectors of $\mathbf{L}$

$$\mathbf{L} = \mathbf{D} - \mathbf{W} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

$\mathbf{v}_1 = [1, 1, 1, 1, 1, 1]^\mathsf{T}$ is an eigenvector with eigenvalue 0

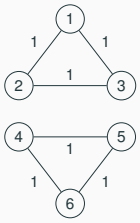
$\mathbf{v}_2 = [1, 1, 1, -1, -1, -1]^\mathsf{T}$ is also an eigenvector with eigenvalue 0

$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2$ for any α_1, α_2 is also an eigenvector with eigenvalue 0

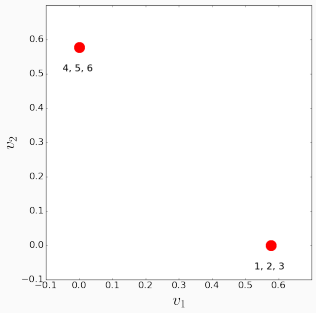
We can use the matrix $[\mathbf{v}_1, \mathbf{v}_2]$ as the $N \times 2$ feature matrix, i.e., two-dimensional embedding of the data

We apply k-means on this feature matrix

Example 1: Clustering using k-Means on Data Embedding

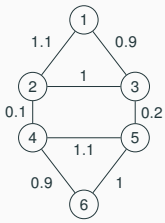


Suppose all edge weights are 1 (0 for missing edges)



Eigenvector computation gives two orthonormal vectors that span the same linear subspace as our eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ computed by hand

Example 2: Laplacian, Eigenvectors, Data Embedding



Suppose all edge weights are 1 (0 for missing edges)

Let us consider some eigenvectors of $\mathbf{L}$

$$\mathbf{L} = \mathbf{D} - \mathbf{W} = \begin{bmatrix} 2 & -1.1 & -0.9 & 0 & 0 & 0 \\ -1.1 & 2.2 & -1 & -0.1 & 0 & 0 \\ -0.9 & -1 & 2.1 & 0 & -0.2 & 0 \\ 0 & -0.1 & 0 & 2.1 & -1.1 & -0.9 \\ 0 & 0 & -0.2 & -1.1 & 2.3 & -1 \\ 0 & 0 & 0 & -0.9 & -1 & 1.9 \end{bmatrix}$$

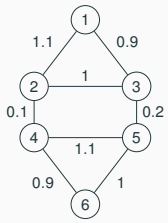
When the weights are slightly perturbed, $\mathbf{v}_1 = [1, \ldots, 1]^\mathsf{T}$ is still an eigenvector with eigenvalue 0

We can't compute the second eigenvector $\mathbf{v}_2$ by hand

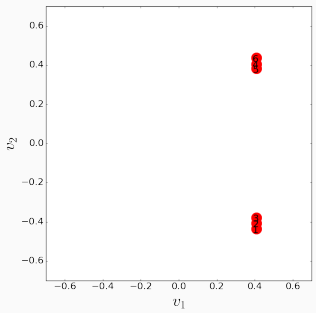
Nevertheless, we expect that the eigenspace corresponding to similar eigenvalues is relatively stable

We can still use the matrix $[\mathbf{v}_1, \mathbf{v}_2]$ as the $N \times 2$ feature matrix and perform k-means

Example 2: Clustering using k-Means on Data Embedding



Suppose all edge weights are 1 (0 for missing edges)



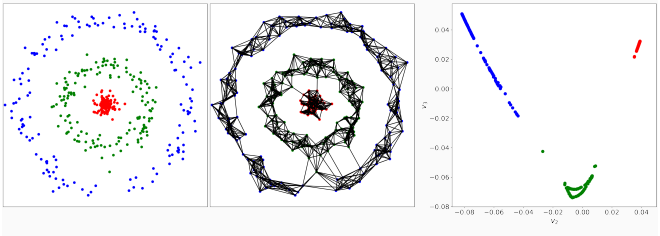
Spectral Clustering Algorithm

Input: Weighted graph with weighted adjacency matrix $\mathbf{W}$

1. Construct Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$
2. Find $\mathbf{v}_1 = \mathbf{1}, \mathbf{v}_2, \ldots, \mathbf{v}_{l+1}$ the k-eigenvectors
3. Construct the $N \times l$ feature matrix $\mathbf{V}_l = [\mathbf{v}_2, \cdots, \mathbf{v}_l]$
4. Apply clustering algorithm using $\mathbf{V}_l$ as features, *e.g.*, k-means

Note: If the degrees of nodes are not balanced, using the normalised Laplacian, $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ may be a better idea

Spectral Clustering: Scatterplot for the Clustered 2D Data Embedding



Summary: Clustering

Clustering is grouping together similar data in a larger collection of heterogeneous data

Definition of good clusters often user-dependent

Clustering algorithms in feature space, *e.g.*, k-Means

Clustering algorithms that only use (dis)similarities: k-Medoids, hierarchical clustering

Spectral clustering when clusters may be non-convex