Machine Learning (AIMS) - MT 2018
2. Clustering

Varun Kanade

University of Oxford

November 6, 2018

Outline

This week, we will study some approaches to clustering

- Defining an objective function for clustering
- k-Means formulation for clustering
- Multidimensional Scaling
- Hierarchical clustering
- Spectral clustering

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

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

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

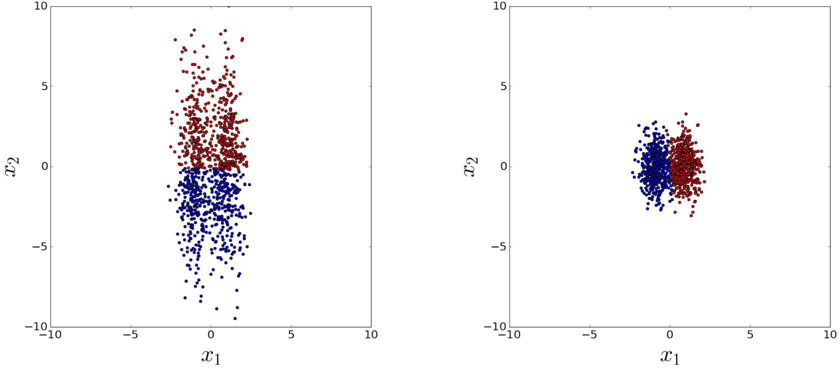
Defining Dissimilarity

- Weighted dissimilarity between (real-valued) attributes

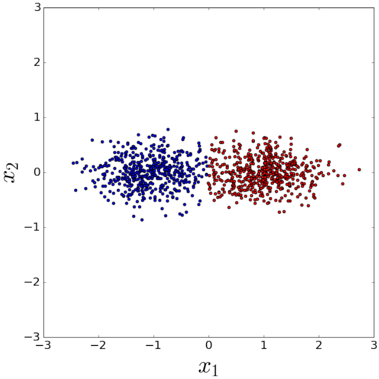
$$d(\mathbf{x}, \mathbf{x}') = f\left(\sum_{i=1}^{D} w_i d_i(x_i, x_i')\right)$$

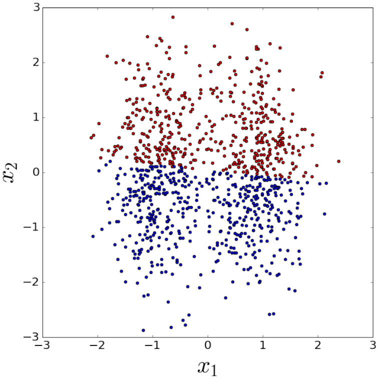
- In the simplest setting $w_i = 1$ and $d_i(x_i, x_i') = (x_i - x_i')^2$ and $f(z) = z$, which corresponds to the squared Euclidean distance
- Weights allow us to emphasise features differently
- If features are ordinal or categorical then define distance suitably
- Standardisation (mean 0, variance 1) may or may not help

Helpful Standardisation



Unhelpful Standardisation





Partition Based Clustering

Want to partition the data into subsets $C_1, \ldots, C_k$, where k is fixed in advance

Define quality of a partition by

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

Partition Based Clustering

Want to partition the data into subsets $C_1, \dots, C_k$, where k is fixed in advance

Define quality of a partition 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 $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$, then

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

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

The objective is minimising the sum of squares of distances to the mean within each cluster

Outline

Clustering Objective

k-Means Formulation of Clustering

Multidimensional Scaling

Hierarchical Clustering

Spectral Clustering

Partition Based Clustering : k-Means Objective

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

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

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

Partition Based Clustering : k-Means Objective

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

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

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

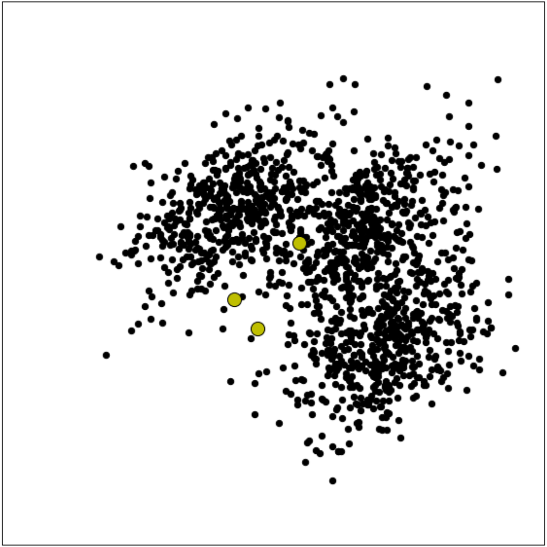
If we fix $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_j$, finding a partition $(C_j)_{j=1}^k$ that minimises W is easy

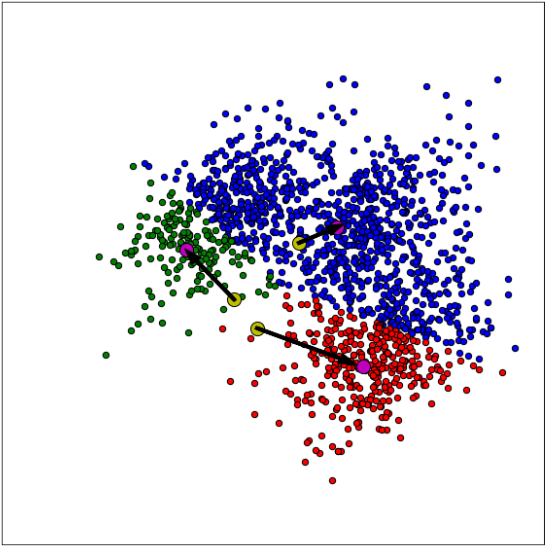
$$C_j = \{i \mid \|\mathbf{x}_i - \boldsymbol{\mu}_j\| = \min_{j'} \|\mathbf{x}_i - \boldsymbol{\mu}_{j'}\|\}$$

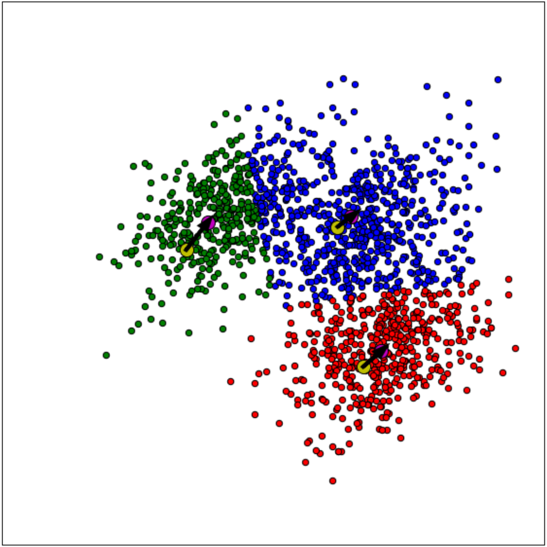
If we fix the clusters $C_1, \ldots, C_k$ minimising W with respect to $(\boldsymbol{\mu}_j)_{j=1}^k$ is easy

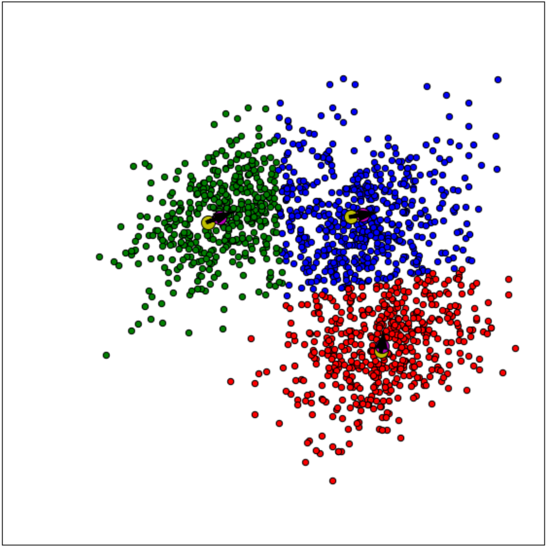
$$\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}_i$$

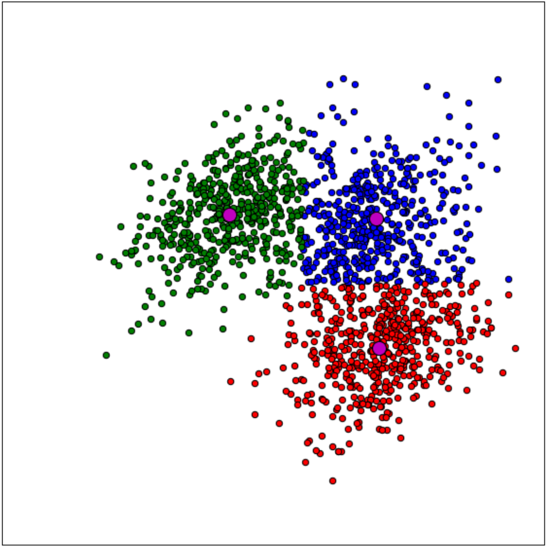
Iteratively run these two steps - assignment and update



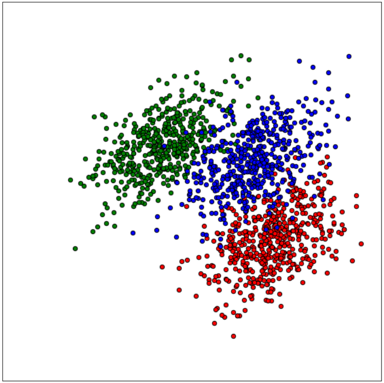




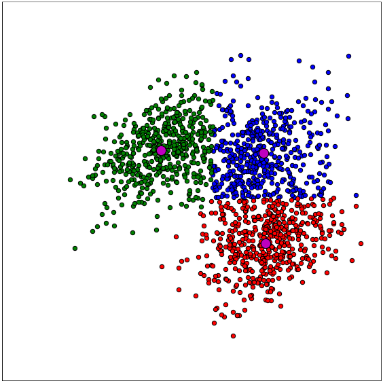




Ground Truth Clusters



k-Means Clusters (k = 3)



The k-Means Algorithm

1. Intialise means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ “randomly”

2. Repeat until convergence:

   a. Find assignments of data to clusters represented by the mean that is closest to obtain, $C_1, \ldots, C_k$:

   $$C_j = \{i \mid j = \operatorname*{argmin}_{j'} \left\| \mathbf{x}_i - \boldsymbol{\mu}_{j'} \right\|_2^2 \}$$

   b. Update means using the current cluster assignments:

   $$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}_i$$

The k-Means Algorithm

1. Intialise means $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ "randomly"

2. Repeat until convergence:

   a. Find assignments of data to clusters represented by the mean that is closest to obtain, $C_1, \dots, C_k$:

   $$C_j = \{i \mid j = \underset{j'}{\operatorname{argmin}} \left\| \mathbf{x}_i - \boldsymbol{\mu}_{j'} \right\|_2^2 \}$$

   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

The k-Means Algorithm

Does the algorithm always converge?

The k-Means Algorithm

Does the algorithm always converge?

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

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

Convergence may be very slow in the worst-case, but typically fast on real-world instances

The k-Means Algorithm

Does the algorithm always converge?

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

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

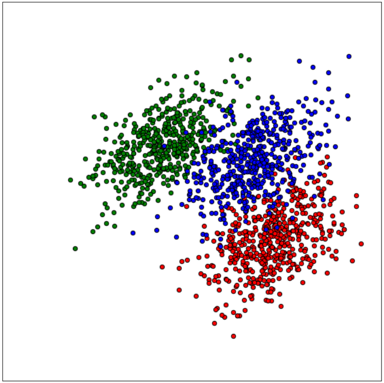
Convergence may be very slow in the worst-case, but typically fast on real-world instances

Convergence is probably to a local minimum. Run multiple times with random initialisation.

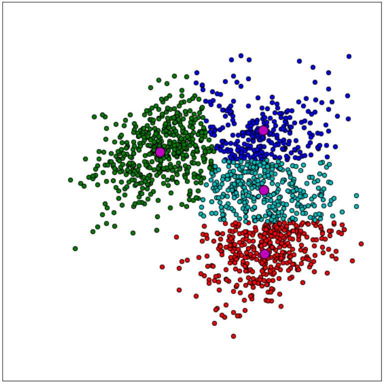
Can use other criteria: k-medoids, k-centres, etc.

Selecting the right k is not easy: plot W against k and identify a "kink"

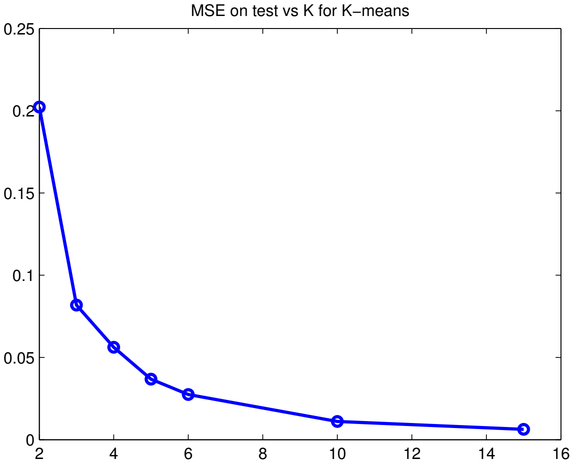
Ground Truth Clusters



k-Means Clusters ($k = 4$)



Choosing the number of clusters k



- As in the case of PCA, larger k will give better value of the objective
- Choose suitable k by identifying a "kink" or "elbow" in the curve

(Source: Kevin Murphy, Chap 11)

Outline

Clustering Objective

k-Means Formulation of Clustering

Multidimensional Scaling

Hierarchical Clustering

Spectral Clustering

Multidimensional Scaling (MDS)

In certain cases, it may be easier to define (dis)similarity between objects than embed them in Euclidean space

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

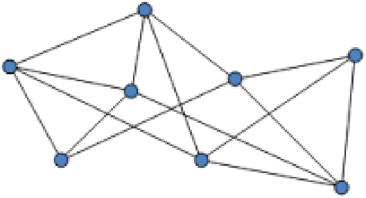
Multidimensional Scaling (MDS)

In certain cases, it may be easier to define (dis)similarity between objects than embed them in Euclidean space

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

Ideal Setting: Suppose for some N points in $\mathbb{R}^D$ we are given all pairwise Euclidean distances in a matrix $\mathbf{D}$

Can we reconstruct $\mathbf{x}_1, \ldots, \mathbf{x}_N$, i.e., all of $\mathbf{X}$?



Multidimensional Scaling

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

We cannot recover $\mathbf{X}$ exactly; we can aim to determine $\mathbf{X}$ up to these transformations

Multidimensional Scaling

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

We cannot recover $\mathbf{X}$ exactly; we can aim to determine $\mathbf{X}$ up to these transformations

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

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

Here $\mathbf{M} = \mathbf{X}\mathbf{X}^\mathsf{T}$ is the $N \times N$ matrix of dot products

Exercise: Show that assuming $\sum_i \mathbf{x}_i = \mathbf{0}$, $\mathbf{M}$ can be recovered from $\mathbf{D}$

Multidimensional Scaling

Consider the (full) SVD: $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$

We can write $\mathbf{M}$ as

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

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

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

Because, $\mathbf{M}$ is symmetric and positive semi-definite, $\mathbf{U}^\mathsf{T} = \mathbf{U}^{-1}$ and all entries of (diagonal matrix) $\mathbf{\Lambda}$ are non-negative

Let $\widetilde{\mathbf{X}} = \mathbf{U}\mathbf{\Lambda}^{1/2}$

If we are satisfied with approximate reconstruction, we can use truncated eigendecomposition

Multidimensional Scaling: Additional Comments

In general if you define (dis)similarities on objects such as text documents, genetic sequences, *etc.*, we cannot be sure that the generated similarity matrix $\mathbf{M}$ will be positive semi-definite or that the dissimilarity matrix $\mathbf{D}$ is a valid squared Euclidean distance

If such cases, we cannot always find a Euclidean embeddding that recovers the (dis)similarities exactly

Minimize stress function: Find $\mathbf{z}_1, \ldots, \mathbf{z}_N$ that minimizes

$$S(\mathbf{Z}) = \sum_{i \neq j} (D_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|)^2$$

Several other types of stress functions can be used

Multidimensional Scaling: Summary

- In certain applications, it may be easier to define pairwise similarities or distances, rather than construct a Euclidean embedding of discrete objects, *e.g.*, genetic data, text data, *etc*.

- Many machine learning algorithms require (or are more naturally expressed with) data in some Euclidean space

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

Clustering

Unsupervised Learning by Program Synthesis

Classifying MNIST digits using Logistic Regression

Why Man Utd’s title hopes are over

England pushed towards Test defeat by India

Outline

Clustering Objective

k-Means Formulation of Clustering

Multidimensional Scaling

Hierarchical Clustering

Spectral Clustering

Hierarchical Clustering

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, county, town level data

Two Algorithmic Strategies for Clustering

- Agglomerative: Bottom-up, clusters formed by merging smaller clusters
- Divisive: Top-down, clusters formed by splitting larger clusters

Visualise this as a dendrogram or tree

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

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

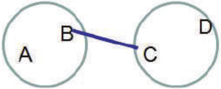
- Average Linkage

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

Measuring Dissimilarity at Cluster Level

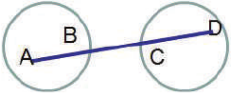
- Single Linkage

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



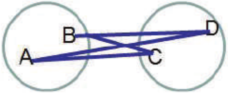
- Complete Linkage

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



- 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, $(j, k) = \underset{j,k \in S}{\operatorname{argmin}} D(j, k)$
   b. Let $C_l = C_j \cup C_k$
   c. If $C_l = \{1, \ldots, N\}$, break;
   d. Set $S = (S \setminus \{j, k\}) \cup \{l\}$
   e. Update $D(i, l)$ for all $i \in S$ (using desired linkage property)

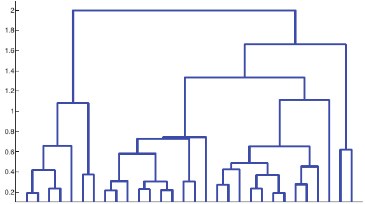
Hierarchical Clustering: Dendrogram

Outputs of hierarchical clustering algorithms are typically represented using dendrograms

A dendrogram is a binary tree, representing clusters as they were merged

The height of a node represents dissimilarity

Cutting the dendrogram at some level gives a partition of data



Outline

Clustering Objective

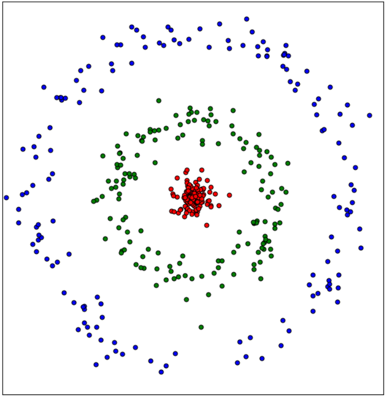
k-Means Formulation of Clustering

Multidimensional Scaling

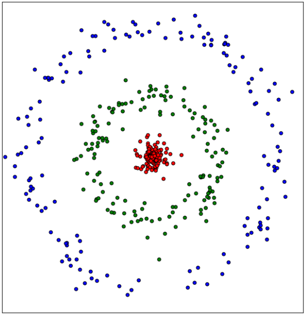
Hierarchical Clustering

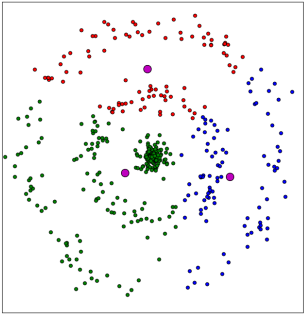
Spectral Clustering

Spectral Clustering



Spectral Clustering: Limitations of k-Means





Limitations of k-means

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

Kernel PCA followed by k-means can result in better clusters

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

Spectral Clustering

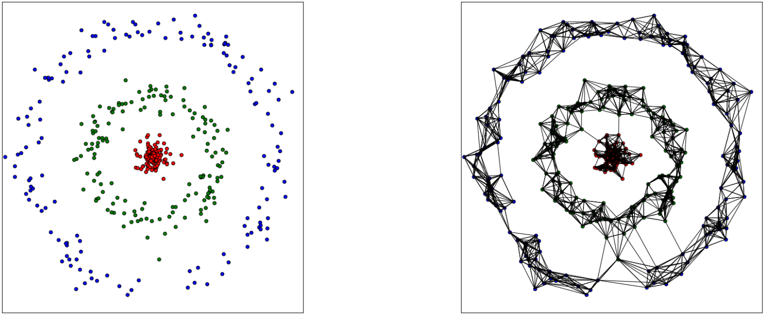
Construct a graph from data; one node for every point in dataset

Use similarity measure, ***e.g.,*** $s_{i,j} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma)$

Construct mutual K-nearest neighbour graph, ***i.e.,*** (i, j) is an edge if either i is among the K nearest neighbours of j or vice versa

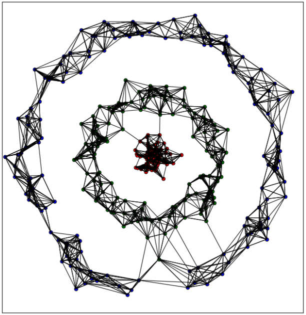
The weight of edge (i, j), if it exists is $s_{i,j}$

Spectral Clustering

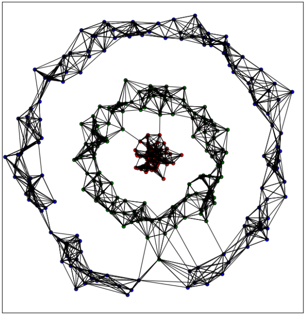


Spectral Clustering

Use graph partitioning algorithms



Spectral Clustering



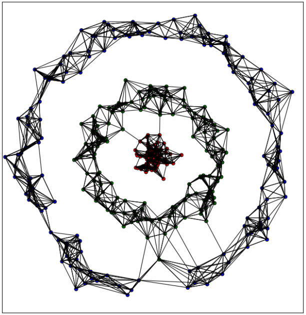
Use graph partitioning algorithms

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

Spectral Clustering



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

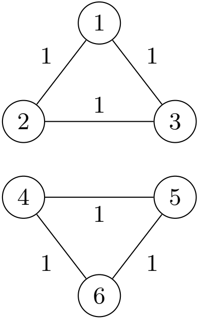
$\mathbf{W}$ is the weighted adjacency matrix

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

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

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

Spectral Clustering: Simple Example



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

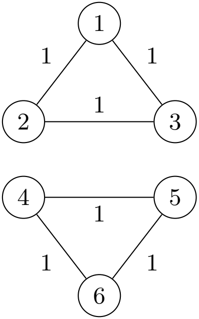
The weighted adjacency matrix, the degree matrix and the Laplacian are given by

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

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

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

Spectral Clustering: Simple Example



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

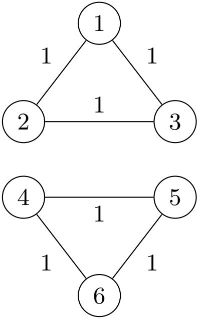
eigenvalue 0

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

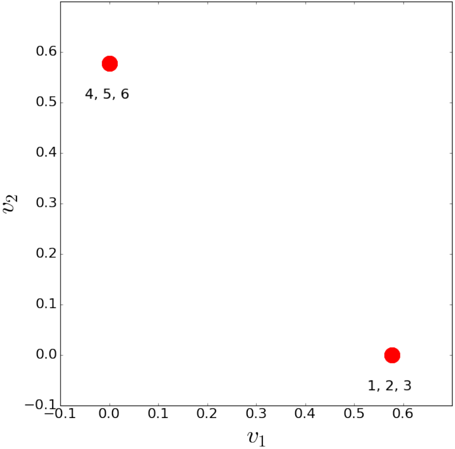
$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2$ for any α_1, α_2 is also an eigenvector with eigenvalue 0

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

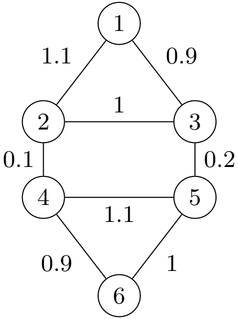
Spectral Clustering: Simple Example



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



Spectral Clustering: Simple Example



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

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

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

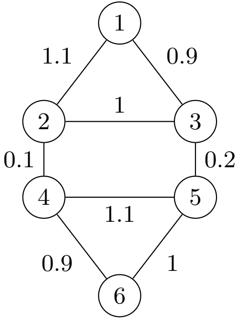
When the weights are slightly perturbed, $\mathbf{v}_1 = [1, \ldots, 1]^\mathsf{T}$ is still an eigenvector with eigenvalue 1

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

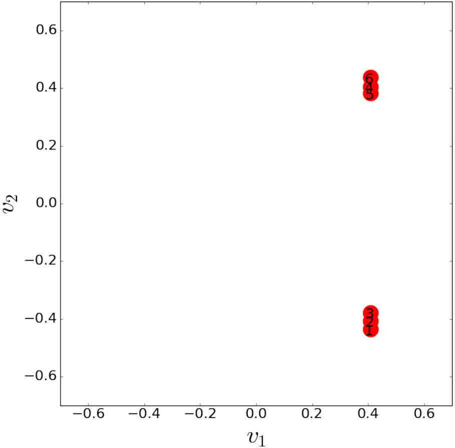
Nevertheless, we expect that the eigenspace corresponding to similar eigenvalues is relatively stable

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

Spectral Clustering: Simple Example



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



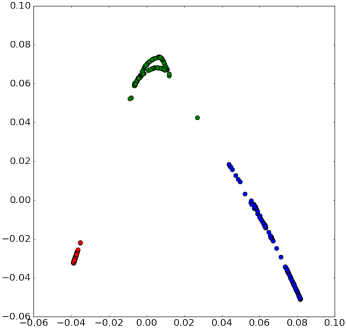
Spectral Clustering Algorithm

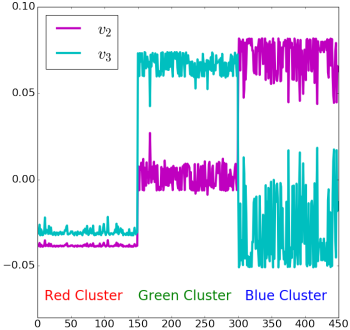
Input: Weighted graph with weighted adjacency matrix $\mathbf{W}$

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, $\widetilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ may be a better idea

Spectral Clustering





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