MATH 250: Mathematical Data Visualization

Applications: Clustering

Peter A. Gao

2024-04-15

San José State University

Outline

- Review of Diffusion Maps/Laplacian Eigenmaps
- Spectral clustering and graph partitioning (based on lecture by Guangliang Chen)
- Image segmentation
- Using other dimension reduction approaches for clustering

Other readings

- A Tutorial on Spectral Clustering
- Spectral Clustering: Theory and Implementation
- Normalized Cuts and Image Segmentation

Diffusion mapping and Laplacian eigenmaps: Review

Recall that diffusion mapping and Laplacian eigenmaps are manifold learning techniques that focus on preserving local geometry.

Both methods rely on identifying the eigenvectors of the graph Laplacian (potentially with rescaling).

Graph Laplacian

Let $d_i = \sum_{j=1}^{n} k_{ij}$ be the **degree** of node i (equal to the ith row sum of K).

Let D be the diagonal matrix with entries $d_1, \dots, d_n$.

The **graph Laplacian** matrix is defined as

$$L = D - K$$

Graph Laplacian

Suppose we have a real-valued function f on the graph (so f maps every node $\mathbf{x}_i$ to a real number).

We can think of f as a vector $\mathbf{f} \in \mathbb{R}^n$ with $L\mathbf{f}$ is a vector with ith element (**exercise**):

$$(L\mathbf{f})_i = \sum_j k_{ij}(f_i - f_j)$$

Moreover (**exercise**):

$$\mathbf{f}^\top L\mathbf{f} = \frac{1}{2}\sum_i \sum_j k_{ij}(f_i - f_j)^2$$

What does it mean when $\mathbf{f}^\top L\mathbf{f}$ is large? When $\mathbf{f}^\top L\mathbf{f}$ is small?

Properties of the graph Laplacian

Observe that $L \in \mathbb{R}^{n\times n}$ and:

- L is symmetric
- All the rows and columns of L sum to $\mathbb{0}$, meaning that $\mathbb{1}$ is an eigenvector of L with eigenvalue 0.
- For any vector $\mathbf{f} \in \mathbb{R}^n$,

  $$\mathbf{f}^\top L\mathbf{f} = \frac{1}{2}\sum_i \sum_j k_{ij}(f_i - f_j)^2$$

  meaning that L is positive semidefinite. Thus, its eigenvalues are nonnegative.

Graph Laplacian

Recall that the algebraic multiplicity of the eigenvalue `0` is the number of connected components of the graph.

Suppose graph has k connected components. Assume that the nodes are ordered according to their conencted components. Then K and L will be block diagonal:

$$\begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{bmatrix}$$

Each L_i is a graph Laplacian for the corresponding subgraph, so the vector with ones for each node in component i and zeroes elsewhere will be an eigenvector for L with eigenvalue zero.

Diffusion mapping in one dimension

The problem

$$\min_{\mathbf{f}\neq\mathbf{0},\mathbf{f}^\top\mathbf{1}=0} \frac{\mathbf{f}^\top L\mathbf{f}}{\mathbf{f}^\top\mathbf{f}}$$

is a (constrained) Rayleigh quotient problem.

Without the constraint $\mathbf{f}^\top\mathbf{1} = 0$ the minimizing $\mathbf{f}$ is the eigenvector of L corresponding to the smallest eigenvalue $\lambda_1 = 0$.

In particular the eigenvector is $\mathbf{v}_1 = \mathbf{1}$.

Diffusion mapping in one dimension

Using the constraint, we choose $\mathbf{f}$ to be orthogonal to $\mathbf{1}$.

The minimizer of the constrained Rayleigh quotient is given by the eigenvector corresponding to the second smallest eigenvalue λ_2:

$$\mathbf{f} = \mathbf{v}_2$$

Random-walk normalized Graph Laplacian

In practice, L is generally normalized to account for the sampling density of the data.

Laplacian Eigenmaps: Belkin and Niyogi (2003) proposed the use of a normalized version of L:

$$L^{rw} = D^{-1}L = I - D^{-1}K$$

Observe that if the degrees of the nodes are constant $L = L^{rw}$. However, Coifman and Lafon (2006) claim that using L^{rw} yields eigenvectors that are influenced by the graph density.

L^{rw} is called the **random-walk normalized graph Laplacian**, since the rows of $D^{-1}K$ are length one, making $D^{-1}K$ a row stochastic matrix.

Properties of L^{rw}

Note that λ is an eigenvalue of L^{rw} with corresponding eigenvector $\mathbf{v}$ if and only if λ and $\mathbf{v}$ are a solution to the generalized eigenvalue problem

$$L\mathbf{v} = \lambda D\mathbf{v}$$

Proof:

$$\begin{aligned} L^{rw}\mathbf{v} &= \lambda\mathbf{v} \\ D^{-1}L\mathbf{v} &= \lambda\mathbf{v} \\ L\mathbf{v} &= \lambda D\mathbf{v} \end{aligned}$$

Generalized Rayleigh quotient representation

Solving this generalized eigenvalue problem is equivalent to the following constrained optimization problem:

$$\min_{\mathbf{f}\neq\mathbf{0},\mathbf{f}^\top D\mathbf{1}=0} \frac{\mathbf{f}^\top L\mathbf{f}}{\mathbf{f}^\top D\mathbf{f}} = \min_{\mathbf{f}\neq\mathbf{0},\mathbf{f}^\top D\mathbf{1}=0} \frac{\sum_i \sum_j k_{ij}(f_i - f_j)^2}{\sum_i d_{ii} f_i^2}$$

Note that the denominator ensures that the components of $\mathbf{f}$ account for variation in the degree of the graph.

In addition, the condition $\mathbf{f}^\top D\mathbf{1} = 0$ accounts for the translational invariance, ensuring a non-trivial solution.

What happens when the degree is uniform across the graph?

Clustering

Objective: Given data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$, output clusters $C_1, \ldots, C_k$ where C_i contains the indices of data points in cluster i.

Note that clustering tasks begin with **unlabeled data**; that is, we do not know the labels for our original data.

Spectral clustering

Spectral clustering proceeds by constructing a low-dimensional embedding $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^k$ of the original data $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

The k-means algorithm is then used to cluster the embedded observations $\mathbf{y}_1, \ldots, \mathbf{y}_n$ into clusters.

Spectral clustering

In general, spectral cluster comprises a family of clustering methods that use the spectral decomposition of a similarity matrix

$$K = (k_{ij}) \in \mathbb{R}^{n \times n}, \quad k_{ij} = \begin{cases} k(\mathbf{x}_i, \mathbf{x}_j) & i \neq j \\ 0 & i = j \end{cases}$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function such as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||_2}{h^2}\right)$$

or cosine similarity.

k-means clustering

1. Assume there are k cluster centers $\mu_l^{(t)} \in \mathbb{R}^q$. Assign each data point $\mathbf{y}_i \in \mathbb{R}^q$ to the closest μ_l:

$$z_i^{(t)} = \arg\min_l ||\mathbf{y}_i - \mu_l||_2^2$$

2. Update the cluster centers by computing the average value of all points in the cluster:

$$\mu_l^{(t+1)} = \frac{1}{n_l} \sum_{i:z_i^{(t)}=l} \mathbf{y}_i$$

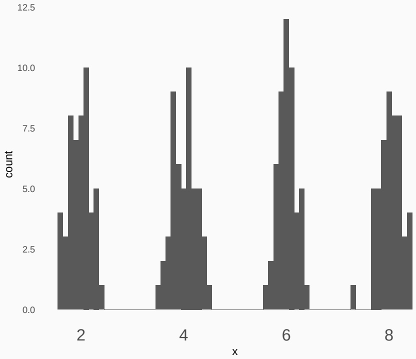
Iterate until convergence is achieved.

Normalized Cut spectral clustering (Shi and Malik 2000)

1. **Input**: Similarity matrix $K \in \mathbf{R}^{n \times n}$ and number of clusters k.
2. Compute the unnormalized graph Laplacian matrix L.
3. Compute the k generalized eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ for the generalized eigenvalue problem $L\mathbf{v} = \lambda D\mathbf{v}$ corresponding to the k smallest non-zero generalized eigenvalues.
4. Let V be the $n \times k$ matrix with columns $\mathbf{v}_1, \dots, \mathbf{v}_k$.
5. Let $\mathbf{y}_i$ be the vector corresponding to the ith row of V for $i = 1, \dots, n$.
6. Cluster the points into k clusters $C_1, \dots, C_k$ using the k-means algorithm.

Example: Mixture of Gaussians (von Luxburg 2007)

Suppose we generate $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}$ from a mixture of Gaussians:



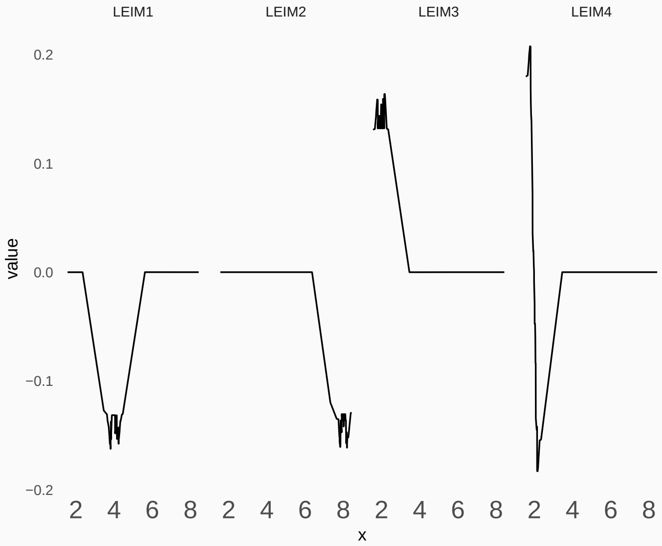
Example: Mixture of Gaussians Code

```r
set.seed(417)
mix_dat <-
  data.frame(x = c(rnorm(50, mean = 2, sd = .2),
                   rnorm(50, mean = 4, sd = .2),
                   rnorm(50, mean = 6, sd = .2),
                   rnorm(50, mean = 8, sd = .2)),
             label = rep(1:4, each = 50))
library(tidyverse)
ggplot(mix_dat, aes(x = x)) +
  geom_histogram(binwidth = .1) +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x = element_text(size = 16))
```

Example: Mixture of Gaussians (von Luxburg 2007)

```r
library(dimRed)
emb <- embed(mix_dat$x, "LaplacianEigenmaps",
             ndim = 4, knn = 10, t = 1)
```

Example: Mixture of Gaussians (von Luxburg 2007)

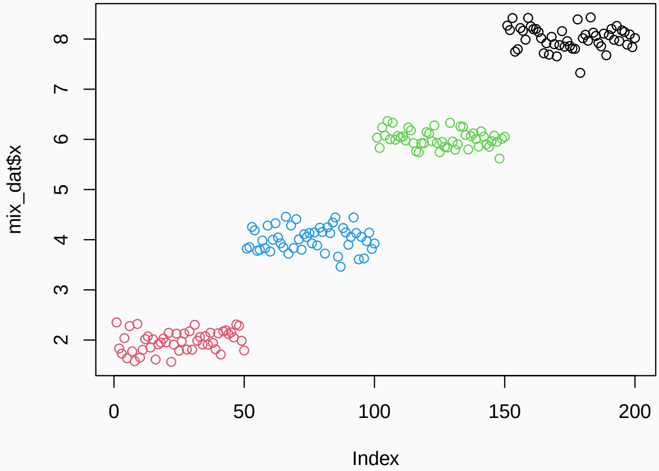


Example: Mixture of Gaussians Code

```r
plot_dat <-
  mix_dat |>
  bind_cols(emb@data@data) |>
  arrange(x) |>
  pivot_longer(contains("LEIM"), names_to = "eigenvector",
ggplot(plot_dat, aes(x = x, y = value)) +
  geom_line() +
  facet_grid(cols = vars(eigenvector)) +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x = element_text(size = 16))
```

Example: Mixture of Gaussians (von Luxburg 2007)

```r
kmeans_res <- kmeans(emb@data@data, centers = 4)
plot(mix_dat$x, col = kmeans_res$cluster)
```



Example: Mixture of Gaussians (von Luxburg 2007)

```r
options(width = 75)
emb@data@data[1, ] # y_1
```

```
        LEIM1         LEIM2         LEIM3         LEIM4 
-3.284272e-05  4.982912e-08  1.312193e-01 -1.538791e-01 
```

```r
emb@data@data[51, ] # y_2
```

```
        LEIM1         LEIM2         LEIM3         LEIM4 
-1.534113e-01  1.381505e-06 -3.839758e-05 -2.846851e-14 
```

```r
emb@data@data[101, ] # y_3
```

```
        LEIM1         LEIM2         LEIM3         LEIM4 
 1.041825e-05  3.403808e-07 -5.138115e-06  1.285431e-14 
```

```r
emb@data@data[151, ] # y_4
```

```
        LEIM1         LEIM2         LEIM3         LEIM4 
-1.367311e-06 -1.518367e-01  5.730203e-08 -1.373807e-14 
```

Graph partitioning

In the process of spectral clustering, we represent the data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ using a graph.

Suppose we wish to form two clusters. This is equivalent to **partitioning** the graph into two groups.

Typically, we will want a partition such that the edges **between** the groups have low weight and the edges **within** a group have high weight.

It turns out that spectral clustering produces an approximate solution to a graph partitioning problem.

Graph partitioning

Given a subset of vertices A, let $\mathbf{1}_A$ be the indicator vector for A:

$$\mathbf{1}_A = (a_1, \dots, a_n)^\top, \quad a_i = 1 \text{ if i} \in \text{A and 0 otherwise}$$

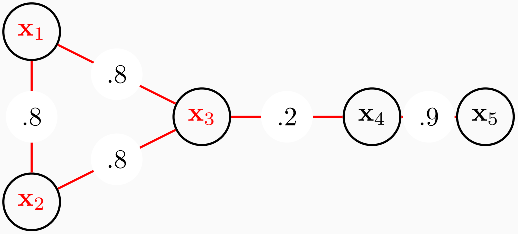
There are two typical ways for measuring the size of $A \subset V$:

$$|A| := \text{ the number of vertices in } A$$

$$\text{vol}(A) := \sum_{i \in A} d_i$$

The volume sums the weights of all edges connected to notes in A.

Graph partitioning



The three nodes on the left form a subgraph A with $|A| = 3$ and $\mathrm{vol}(A) = 1.6 + 1.6 + 1.8 = 5$.

Graph cut

For two subsets $A, B \in V$, we define

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

$W(A, B)$ quantifies the strength of the connection between nodes in A and nodes in B.

Let $\overline{A}$ denote the complement of A. For a finite number k of subsets, let

$$\mathrm{cut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i)$$

The **cut** quantifies the separation between the subsets $A_1, \dots, A_k$ Note that the scaling $\frac{1}{2}$ is only for notational consistency.

Mincut

As a result,

$$\operatorname{cut}(A, \overline{A}) := \sum_{i \in A, j \in \overline{A}} k_{ij}$$

One approach to graph partitioning is just to choose a partition $A_1, \dots, A_k$ that minimizes $\operatorname{cut}(A_1, \dots, A_k)$.

However, in practice, this may not yield reasonable partitions. Often, an individual outlier may be separated from the rest of the graph.

We can define an alternative objective function that ensures that the clusters $A_1, \dots, A_k$ are reasonably sized.

Normalized cut

One such objective function is Ncut:

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

For $k = 2$, this becomes

$$\mathrm{Ncut}(A_i, \overline{A}_i) = W(A_i, \overline{A}_i) \left(\frac{1}{\mathrm{vol}(A_i)} + \frac{1}{\mathrm{vol}(\overline{A}_i)} \right)$$

Minimizing Ncut requires optimizing over all possible partitions, which is NP-hard.

Relaxing Ncut leads to spectral clustering.

Approximating Ncut

For ease of notation, consider a partition $V = A \cup B$, where $B = \overline{A}$. Let $\mathrm{vol}(A) = a$ and $\mathrm{vol}(B) = b$.

Let

$$\mathbf{f} = \frac{1}{a}\mathbf{1}_A - \frac{1}{b}\mathbf{1}_B$$

so

$$f_i = \begin{cases} \frac{1}{a}, & i \in A \\ -\frac{1}{b}, & i \in B \end{cases}$$

Approximating Ncut

Then

$$\begin{aligned}
\mathbf{f}^\top L\mathbf{f} &= \sum_i \sum_j k_{ij}(f_i - f_j)^2 \\
&= \sum_{i\in A, j\in B} k_{ij} \left(\frac{1}{a} + \frac{1}{b}\right)^2 \\
&= \mathrm{cut}(A,B) \left(\frac{1}{a} + \frac{1}{b}\right)^2
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{f}^\top D\mathbf{f} &= \sum_i d_{ii} f_i^2 \\
&= \sum_{i\in A} \frac{1}{a^2} d_{ii} + \sum_{j\in B} \frac{1}{b^2} d_{ii} \\
&= \frac{1}{a^2}\mathrm{vol}(A) + \frac{1}{b^2}\mathrm{vol}(B) = \frac{1}{a} + \frac{1}{b}
\end{aligned}$$

Approximating Ncut

As a result,

$$\frac{\mathbf{f}^\top L\mathbf{f}}{\mathbf{f}^\top D\mathbf{f}} = \mathrm{cut}(A,B)\left(\frac{1}{a}+\frac{1}{b}\right) = \mathrm{Ncut}(A,B)$$

Moreover, observe that

$$\mathbf{f}^\top D\mathbf{1} = \sum_i f_i d_{ii} = \frac{1}{a}\mathrm{vol}(A) - \frac{1}{b}\mathrm{vol}(B) = 0$$

Approximating Ncut

As a result, the following two problems are equivalent:

$$\min_{A \cap B = \emptyset} \mathrm{Ncut}(A, B) \iff \min_{\mathbf{f} \in \{\alpha, -\beta\}^n,\ \mathbf{f}^\top D \mathbf{1} = 0} \frac{\mathbf{f}^\top L \mathbf{f}}{\mathbf{f}^\top D \mathbf{f}}$$

Here, the condition $\mathbf{f} \in \{\alpha, -\beta\}^n$ requires that $\mathbf{f}$ only takes two values. This condition means that this optimization problem is **not yet equivalent** to the normalized spectral clustering problem.

Approximating Ncut

Removing the condition $\mathbf{f} \in \{\alpha, -\beta\}^n$ yields the relaxed problem

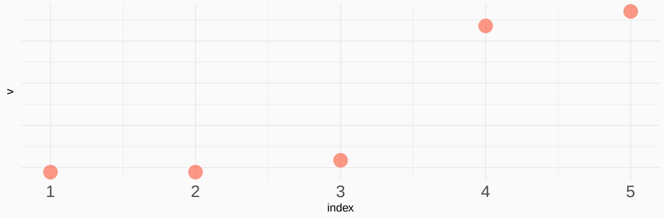
$$\min_{\mathbf{f} \neq \mathbf{0}, \mathbf{f}^\top D \mathbf{1} = 0} \frac{\mathbf{f}^\top L \mathbf{f}}{\mathbf{f}^\top D \mathbf{f}}$$

which is the generalized Rayleigh quotient problem used in Laplacian Eigenmaps.

As such, the first eigenvector $\mathbf{v}$ corresponding to a non-zero eigenvalue yields an approximate solution to the normalized cut problem.

1D Example

The eigenvector corresponding to the second smallest eigenvector is $(-0.277, -0.277, -0.207, 0.590, 0.676)$

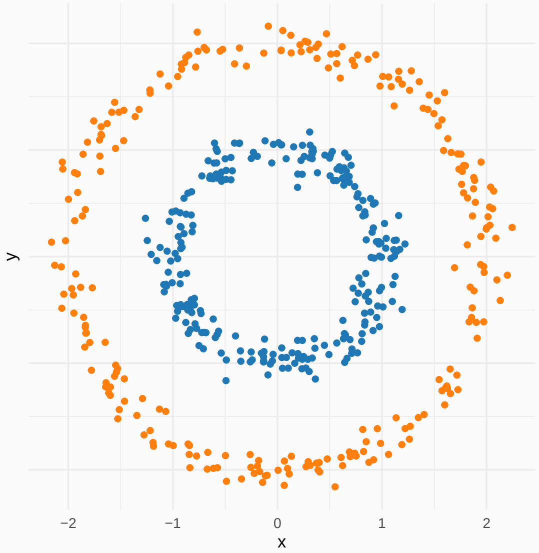


Spectral clustering

Along these lines, for k clusters, spectral clustering with L_{rw} yields an approximate solution to the Ncut problem.

However, there is no guarantee on how close this solution will be the exact solution.

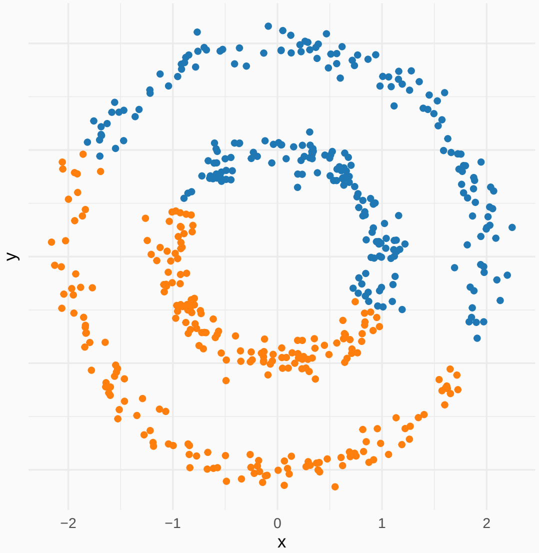
Additional examples: Concentric rings



Additional examples: Concentric rings

```r
library(ggthemes)
library(ggsci)
set.seed(123)
n <- 500
theta <- runif(n, min = 0, max = 2 * pi)
dat <- data.frame(cluster = rep(1:2, each = n / 2),
                  theta = theta) |>
  mutate(x = cluster * cos(theta) + rnorm(n, sd = .1),
         y = cluster * sin(theta) + rnorm(n, sd = .1))
ggplot(dat, aes(x = x, y = y, color = as.factor(cluster))) +
  geom_point() +
  scale_color_d3(name = "digit", scale_name = "category10") +
  theme_minimal() +
  theme(aspect.ratio = 1,
        legend.position = "none",
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

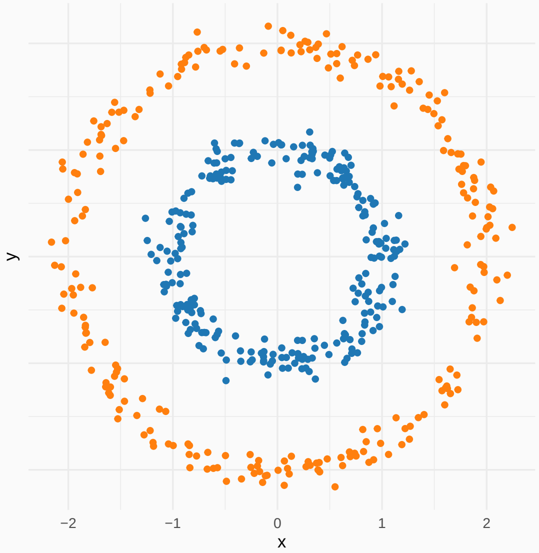
k-means applied to original data



k-means applied to original data

```r
km_res <- kmeans(as.matrix(dat[, c("x", "y")]), centers = 2)
ggplot(dat, aes(x = x, y = y, color = as.factor(km_res$cluster))) +
  geom_point() +
  scale_color_d3(name = "digit", scale_name = "category10") +
  theme_minimal() +
  theme(aspect.ratio = 1,
        legend.position = "none",
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

Spectral clustering



Spectral clustering

```r
emb <- embed(as.matrix(dat[, c("x", "y")]), "LaplacianEigenmaps",
             ndim = 2, knn = 10, t = 1)
km_res <- kmeans(emb@data@data, centers = 2)
ggplot(dat, aes(x = x, y = y, color = as.factor(km_res$cluster))) +
  geom_point() +
  scale_color_d3(name = "digit", scale_name = "category10") +
  theme_minimal() +
  theme(aspect.ratio = 1,
        legend.position = "none",
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

Image segmentation

Image segmentation is the problem of converting a single image into multiple segments. These segments typically represent different objects in the image, which can then be classified.

Image segmentation aims to convert a complex image into simpler, classifiable parts, and used for object recognition, facial recognition, and other machine learning tasks.

The original application for the normalized cuts problem was image segmentation, as seen in the following example.

Image segmentation (Shi and Malik 2000)

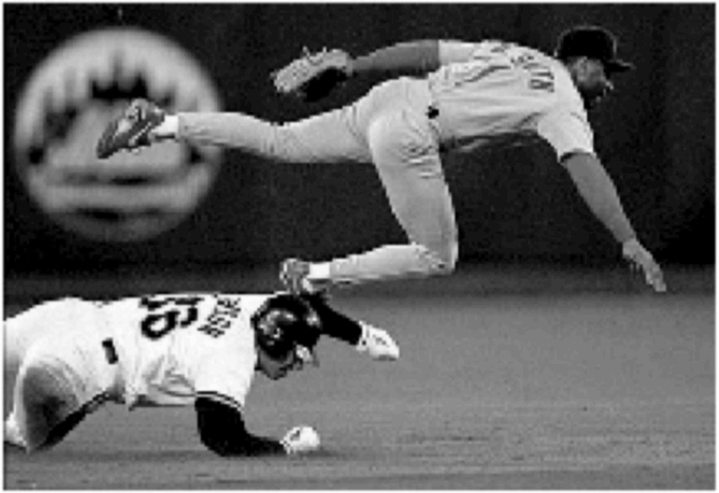


Fig. 2. A gray level image of a baseball game.

Image segmentation (Shi and Malik 2000)

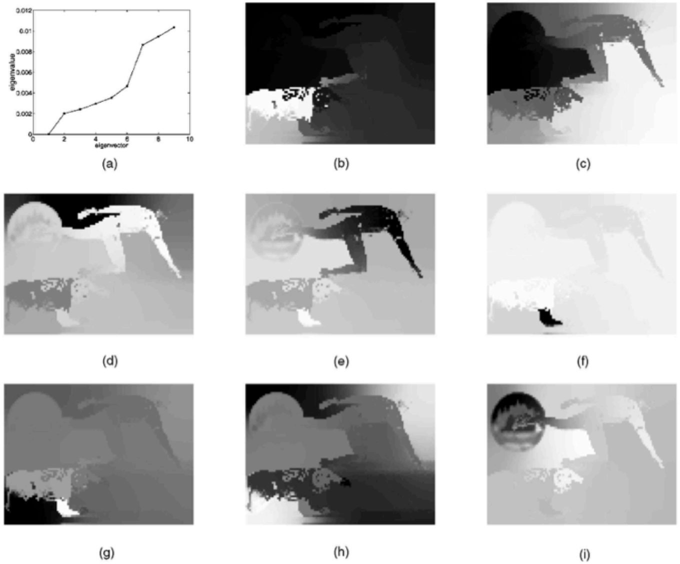


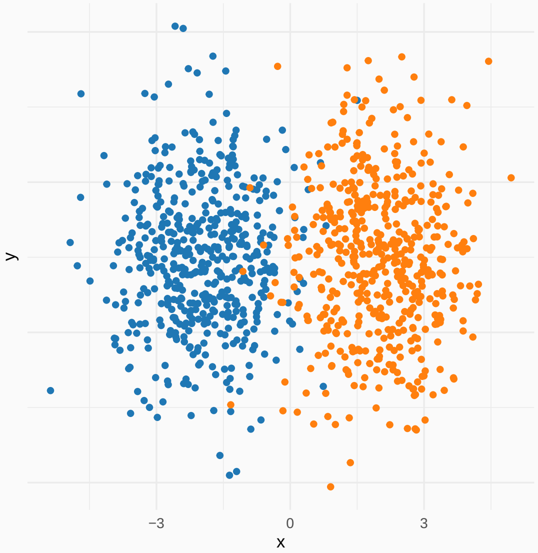
Fig. 3. Subplot (a) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplots (b)-(i) show the eigenvectors corresponding the second smallest to the ninth smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

Using other dimension reduction approaches for clustering

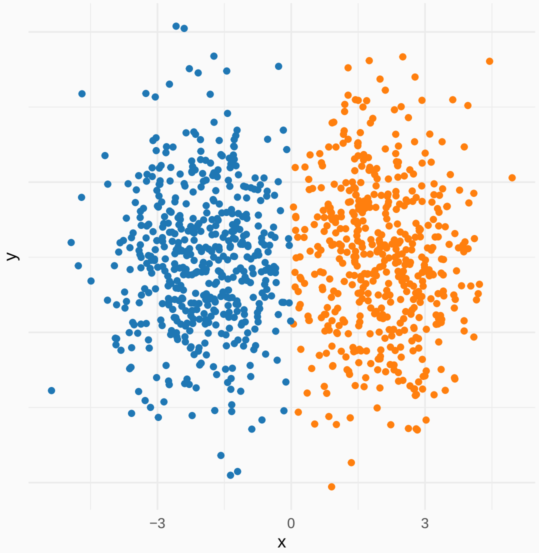
In general, applying dimension reduction techniques before using a clustering algorithm like k-means is not advisable.

Especially for attraction-repulsion algorithms like t-SNE or UMAP, the embedded data may seem to form clusters that are not reflective of the original data.

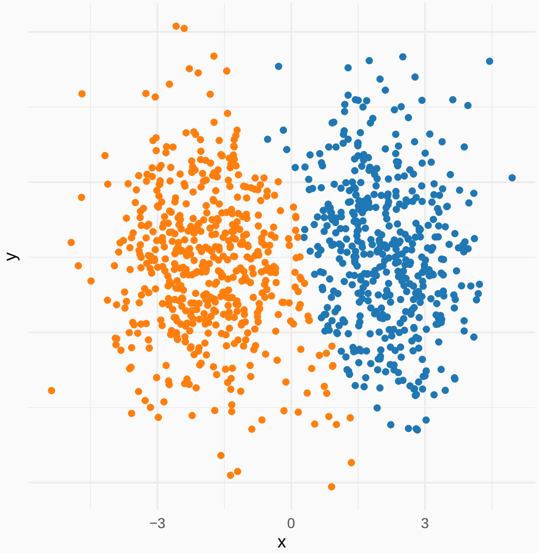
An “easy” clustering problem



An “easy” clustering problem: k-means



An “easy” clustering problem: t-SNE with low perplexity



An “easy” clustering problem: t-SNE with low perplexity

Increasing perplexity will yield better clustering results, but it is difficult to assess how good the results are with real data, when we don’t know what the true clusters look like.

References

Belkin, Mikhail, and Partha Niyogi. 2003. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural Computation* 15 (6): 1373–96. https://doi.org/10.1162/089976603321780317.

Coifman, Ronald R., and Stéphane Lafon. 2006. "Diffusion Maps." *Applied and Computational Harmonic Analysis*, Special Issue: Diffusion Maps and Wavelets, 21 (1): 5–30. https://doi.org/10.1016/j.acha.2006.04.006.