Learning From Data
Lecture 9: Unsupervised Learning III

Yang Li   yangli@sz.tsinghua.edu.cn

November 19, 2020

Front Matter

Today's Lecture

Correlation Analysis

- Review: CCA
- HGR maximal correlation

Spectral Graph Theory

- Similarity graphs
- Spectral clustering

Review: CCA Algorithm

Goal: Learn (linear) dependence between two sets of variables.
Input: Covariance matrices for centered data X and Y:

- Σ_{XY} , invertible Σ_{XX} and Σ_{YY}
- Dimension $k \leq \min(n_1, n_2)$

Output: CCA projection matrices A_k and B_k:

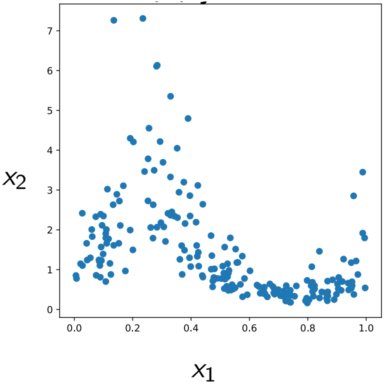
- Compute $\Omega = \Sigma_{XX}^{-\frac{1}{2}} \Sigma_{XY} \Sigma_{YY}^{-\frac{1}{2}}$
- Compute SVD decomposition of Ω

$$\Omega = \begin{bmatrix} | & \dots & | \\ c_1 & \dots & c_{n_1} \\ | & \dots & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \\ & 0 & \end{bmatrix} \begin{bmatrix} -d_1^T- \\ \vdots \\ -d_{n_2}^T- \end{bmatrix}$$

- $A_k = \Sigma_{XX}^{-\frac{1}{2}}[c_1, \dots, c_k]$ and $B_k = \Sigma_{YY}^{-\frac{1}{2}}[d_1, \dots, d_k]$

Review: Discussion of CCA

- Applications:
  - Co-clustering
  - Multi-view regression
- CCA only measures linear dependencies
- Non-linear generalizations:
  - Kernel CCA (KCCA)
  - Deep CCA (DCCA)
  - Maximal HGR Correlation



Non-linear dependency between x_1 and x_2

Maximal HGR Correlation Analysis

A Non-linear Measure of Dependence

Hirschfeld-Gebelein-Renyi (HGR) maximal correlation

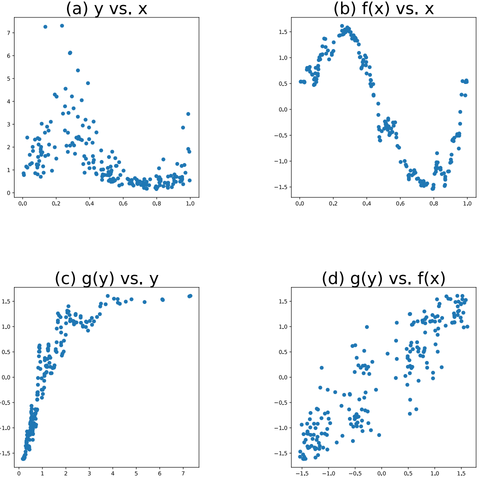
Given random variables X, Y, the HGR maximal correlation is

$$\rho(X;Y) = \max_{f(X),g(Y)} \mathbb{E}[f(X)g(Y)]$$
$$s.t. \mathbb{E}[f(X)] = \mathbb{E}[g(Y)] = 0$$
$$\mathbb{E}[f^2(X)] = \mathbb{E}[g^2(Y)] = 1$$

where $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{Y} \to \mathbb{R}$ are real-valued functions

Example of HGR maximal correlation

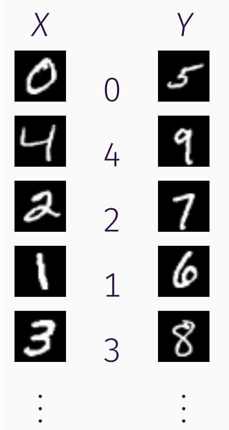
Synthesized data:$y^{(i)} = \exp\left(\sin\left(2\pi x^{(i)} + \frac{\epsilon^{(i)}}{2}\right)\right)$, $e^{(i)} \approx \mathcal{N}(0,1)$
for $i = 1, \ldots, 200$



$$\rho(X; Y) = 0.902$$

Example of HGR maximal correlation

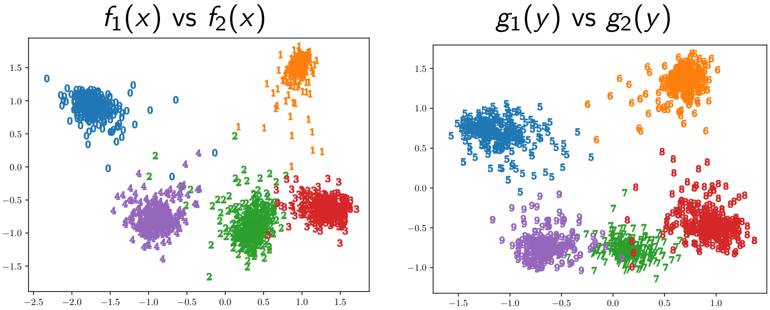
Use multi-dimensional HGR maximal correlation to learn unsupervised features from MNIST.



$$\text{“}\mathbf{y^{(i)}} = \mathbf{x^{(i)}} + \mathbf{4}\text{”}$$

Example of HGR maximal correlation

Use multi-dimensional HGR maximal correlation to learn unsupervised features from MNIST.



How to solve it?

Assume X and Y are both discrete with alphabet $\mathcal{X}$, $\mathcal{Y}$.

$$\mathbb{E}[f(x)g(y)] = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{X,Y}(x,y)f(x)g(y)$$

Define $\phi(x) \triangleq \sqrt{P_X(x)}f(x)$, $\psi(y) \triangleq \sqrt{P_Y(y)}g(y)$, then

$$\mathbb{E}[f(x)g(y)] = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} \frac{P_{X,Y}(x,y)}{\sqrt{P_X(x)P_Y(y)}}\phi(x)\psi(y) = \psi^T B\phi$$

- Matrix $B \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{X}|}$, where $B(y,x) \triangleq \frac{P_{X,Y}(x,y)}{\sqrt{P_X(x)P_Y(y)}}$
- Vectors $\phi \in \mathbb{R}^{|\mathcal{X}|}, \psi \in \mathbb{R}^{|\mathcal{Y}|}$

How to represent the constraints using ϕ and ψ?

How to solve it?

Given $\phi(x) = \sqrt{P_X(x)} f(x)$, $\psi(y) = \sqrt{P_Y(y)} g(y)$

Unit-variance constraints

- $\mathbb{E}[f(x)^2] = 1 \implies$
  $\sum_x P_X(x) \left(\frac{\phi(x)}{\sqrt{P_X(x)}} \right)^2 = \sum_x \phi(x)^2 = ||\phi||^2 = 1$
- Similarly, $\mathbb{E}[g(y)^2] = 1 \implies ||\psi||^2 = 1$

Zero-mean constraints

- $\mathbb{E}[f(x)] = 0 \implies$
  $\sum_x P_X(x) \frac{\phi(x)}{\sqrt{P_X(x)}} = \sum_x \phi(x) \sqrt{P_X(x)} = \langle \phi, \sqrt{P_X} \rangle = 0$, i.e.
  $(\phi \perp \sqrt{P_X})$
- Similarly, $\mathbb{E}[g(y)] = 0 \implies \langle \psi, \sqrt{P_Y} \rangle = 0$, i.e. $(\psi \perp \sqrt{P_Y})$

HGR Maximal Correlation as an SVD problem

Alternative definition for HGR Maximal Correlation

$$\rho(X, Y) = \max_{\phi \in \mathbb{R}^{|\mathcal{X}|}, \psi \in \mathbb{R}^{|\mathcal{Y}|}} \psi^T B \phi$$
$$s.t. ||\phi||^2 = ||\psi||^2 = 1$$
$$\phi \perp \sqrt{P_X}, \psi \perp \sqrt{P_Y}$$

Proposition 1

$(u_1, v_1) = \text{argmax}_{||u||=||v||=1} u^T B v$ *are the largest left and right singular vector of* B.

Proposition 2

The largest left and right singular vectors are $\sqrt{P_Y}$ *and* $\sqrt{P_X}$

Proposition 3

ψ^* *and* ϕ^* *are the 2nd largest left and right singular vectors of* B, *respectively.*

Alternating Condition Expectation (ACE)

A generalization of power iteration for finding singular vectors:

ACE algorithm for 1d data [Breiman & Friedman 1985]

Data: Discrete data samples $x^{(1)}, \ldots, x^{(m)}$
Result: compute $f^*(x), g^*(y)$
Randomly choose $g(y), y \in \mathcal{Y}$ such that $\mathbb{E}[g(Y)] = 0$;
while σ *not converged* **do**
    $f(x) \leftarrow \mathbb{E}_m[g(Y)|X = x]$
    Normalize $f(x)$ $\forall x \in \mathcal{X}$;
    $g(y) \leftarrow \mathbb{E}_m[f(X)|Y = y]$;
    Normalize $g(y)$ $\forall y \in \mathcal{Y}$;
    $\sigma \leftarrow \mathbb{E}_m[f(X)g(Y)]$;
end

Breiman, L. and Friedman, J. H. Estimating optimal transformations for multiple regression and correlation. J. Am. Stat. Assoc., 80(391),1985b

Alternating Condition Expectation (ACE)

A generalization of power iteration for finding singular vectors:

ACE algorithm for 1d data [Breiman & Friedman 1985]

Data: Discrete data samples $x^{(1)}, \dots, x^{(m)}$
Result: compute $f^*(x), g^*(y)$
Randomly choose $g(y), y \in \mathcal{Y}$ such that $\mathbb{E}[g(Y)] = 0$;
while σ *not converged* **do**
    $f(x) \leftarrow \mathbb{E}_m[g(Y)|X = x]$ // $\mathbb{E}_m[\cdot]$: sample expectation ;
    Normalize $f(x)$ $\forall x \in \mathcal{X}$;
    $g(y) \leftarrow \mathbb{E}_m[f(X)|Y = y]$;
    Normalize $g(y)$ $\forall y \in \mathcal{Y}$;
    $\sigma \leftarrow \mathbb{E}_m[f(X)g(Y)]$;
end

Breiman, L. and Friedman, J. H. Estimating optimal transformations for multiple regression and correlation. J. Am. Stat. Assoc., 80(391),1985b

Extension to high dimension case

k-dimensional HGR Maximal Correlation

$$\rho(X;Y) = \max_{\substack{f:\mathcal{X}\to\mathbb{R}^k \\ g:\mathcal{Y}\to\mathbb{R}^k}} \mathbb{E}[f(X)^T g(Y)] \leftarrow \text{ optimize k values in parallel}$$

$$s.t.\ \mathbb{E}[f_i(X)] = \mathbb{E}[g_i(Y)] = 0,\ \forall i = 1,\ldots,k$$

$$\mathbb{E}[f_i(X)^T f_j(X)] = \mathbb{E}[g_i(Y)^T g_j(Y)] = \mathbf{1}\{i=j\},\ \forall i,j = 1,\ldots,k$$

ACE algorithm for k-d data

Data: Discrete data samples $x^{(1)},\ldots,x^{(m)}$
Result: compute $f^*(x), g^*(y)$
Randomly choose $g(y), y \in \mathcal{Y}$ such that $\mathbb{E}[g(Y)] = 0$;
while *σ not converged* **do**
  $f(x) \leftarrow \mathbb{E}_m[g(Y)|X=x]$;
  Normalize $f(x)$ $\forall x \in \mathcal{X}$;
  $g(y) \leftarrow \mathbb{E}_m[f(X)|Y=y]$;
  Normalize $g(y)$ $\forall y \in \mathcal{Y}$;
  $\sigma \leftarrow \mathbb{E}_m[f(X)^T g(Y)]$;
end

Normalize k-d feature: for all $x \in \mathcal{X}$,

- $f(x) \leftarrow f(x) - \mathbb{E}_m[f(X)]$
- $f(x) \leftarrow f(x)\mathbb{E}_m[f(X)f(X)^T]^{-\frac{1}{2}}$

$g(y)$ is normalized similarly for all $y \in \mathcal{Y}$.

Discussion on HGR Maximal Correlation

- Useful for modal estimation from data
- ACE in Python: `https://github.com/mace-cream/xyace` (limited to discrete X and Y)
- Extension to continuous case: a deep neural network implementation of HGR maximal correlation [Wang et. al. 2018]

$$Loss(f,g) = -\hat{\mathbb{E}}[f(X)^T g(Y)] + \frac{1}{2} tr(Cov(f(X))Cov(g(Y)))$$

An Efficient Approach to Informative Feature Extraction from Multimodal Data, Wang, Lichen, et al. AAAI (2018).

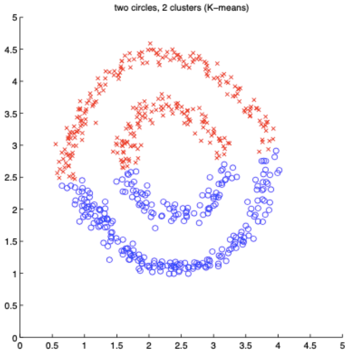
Spectral Graph Theory

Graph Terminologies and Similarity Graphs

Spectral Clustering

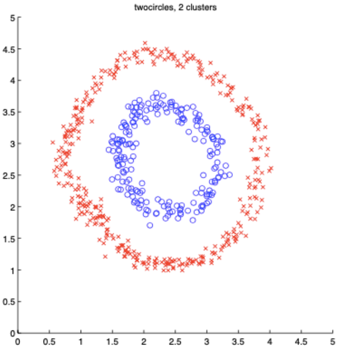
K-Means vs Spectral Clustering

K-Means



(i)

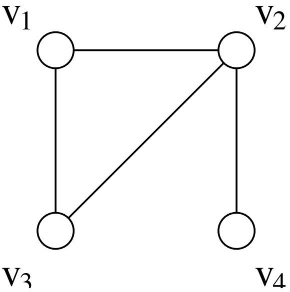
Spectral Clustering



(e)

[Shi & Malik 00; Ng, Jordan, Weiss NIPS 01]

Graph Terminologies

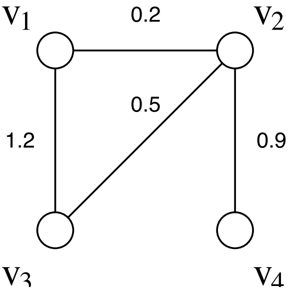


$$W = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- An **undirect graph** $G = (V, E)$ consists of nodes $V = \{v_1, \dots, v_n\}$ and edges $E = \{e_1, \dots, e_m\}$
- Edge e_{ij} connects v_i and v_j if they are **adjacent** or neighbors.
- Adjacency matrix
  $$W_{ij} = \begin{cases} 1 & \text{if there is an edge } e_{ij} \\ 0 & \text{otherwise} \end{cases}$$
- **Degree** d_i of node v_i is the number of neighbors of v_i.

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

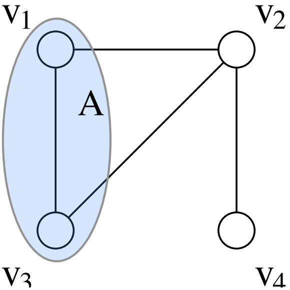
Graph Terminologies



$$W = \begin{bmatrix} 0 & 0.2 & 1.2 & 0 \\ 0.2 & 0 & 0.5 & 0.9 \\ 1.2 & 0.5 & 0 & 0 \\ 0 & 0.9 & 0 & 0 \end{bmatrix}$$

- **Weigthed undirect graph** $G = (V, E, W)$
- Edge weight $w_{ij} \in \mathbb{R}$ between v_i and v_j
- **Weighted adjacency matrix** $W = [w_{ij}]$
- Vertex degree $d_i = \sum_{j=1}^{n} w_{ij}$
- **Degree matrix** $D = diag(d_1, \ldots, d_n)$

Graph Terminologies



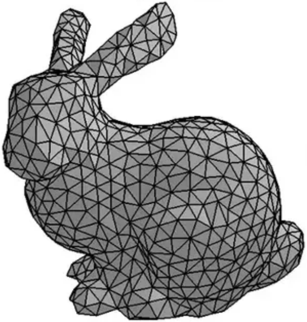
- Given vertex subset $A \subset V$, let $\bar{A} = V \backslash A$ be the complement of A in the graph
- Subset indicator function $\mathbf{1}_A \in \mathbb{R}^n$:

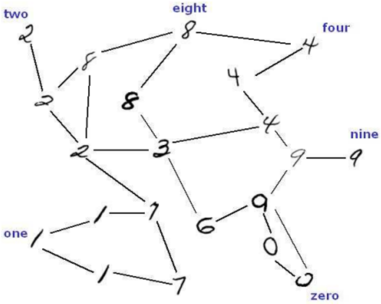
$$1_A\{i\} = \begin{cases} 1 & \text{if } v_i \in A \\ 0 & \text{if } v_i \notin A \end{cases}$$

- Sets $A_1, \ldots, A_k$ form a **partition** of the graph if $A_i \cap A_j = \emptyset$ for all $i \neq j$ and $A_1 \cup \ldots \cup A_k = V$

Represent data using a graph

Some data are naturally represented by a graph e.g. social networks, 3D mesh etc





Use graph to represent similarity in data

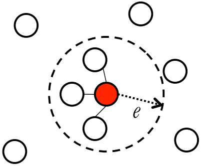
Clustering from a graph point of view

- Given data points $x^{(1)}, \ldots, x^{(n)}$ and **similarity measure** $s_{ij} \geq 0$ for all $x^{(i)}, x^{(j)}$
- A typical **similarity graph** $G = (V, E)$ is
  - $v_i \leftrightarrow x^{(i)}$
  - v_i and v_j are connected if $s_{ij} \geq \delta$ for some threshold δ
- **Clustering**: Divide data into groups such that points in the same group are similar and points in different groups are dissimilar
- **Spectral Clustering (informal)**: *Find a partition of G such that edges between the same group have high weight and edges between different groups have very low weight.*

Building similarity graphs from data

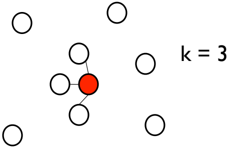
ϵ-neighborhood

Add edges to all points inside a ball of radius f centered at v



k-Nearest Neighbors

Add edges between v's k-nearest neighbors.



Fully connected graph

Often, Gaussian similarity is used

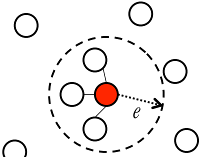
$$W_{i,j} = \exp\left(-\frac{||x^{(i)} - x^{(j)}||_2^2}{2\sigma^2}\right) \text{ for } i, j = 1, \ldots, m$$

Building similarity graphs from data

ϵ-neighborhood

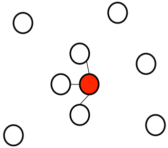
Add edges to all points inside a ball of radius f centered at v

Drawbacks: sensitive to e, edge weights are on similar scale



k-Nearest Neighbors

Add edges between v's k-nearest neighbors.



k = 3

Fully connected graph

Often, Gaussian similarity is used

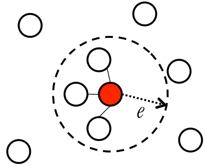
$$W_{i,j} = \exp\left(-\frac{||x^{(i)} - x^{(j)}||_2^2}{2\sigma^2}\right) \text{ for } i, j = 1, \ldots, m$$

Building similarity graphs from data

ϵ-neighborhood

Add edges to all points inside a ball of radius f centered at v

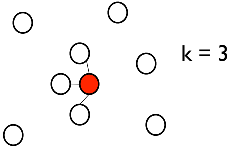
Drawbacks: sensitive to e, edge weights are on similar scale



k-Nearest Neighbors

Add edges between v's k-nearest neighbors.

Drawbacks: may result in asymmetric and irregular graph



Fully connected graph

Often, Gaussian similarity is used

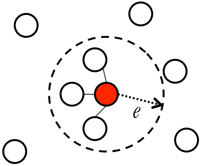
$$W_{i,j} = \exp\left(-\frac{||x^{(i)} - x^{(j)}||_2^2}{2\sigma^2}\right) \text{ for } i, j = 1, \ldots, m$$

Building similarity graphs from data

ϵ-neighborhood

Add edges to all points inside a ball of radius f centered at v

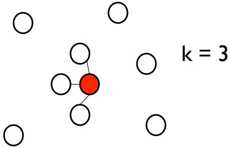
Drawbacks: sensitive to e, edge weights are on similar scale



k-Nearest Neighbors

Add edges between v's k-nearest neighbors.

Drawbacks: may result in asymmetric and irregular graph



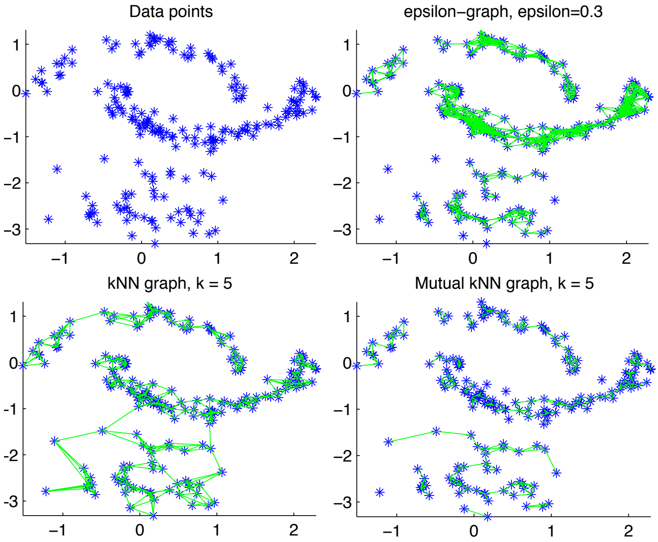
Fully connected graph

Often, Gaussian similarity is used

$$W_{i,j} = \exp\left(-\frac{||x^{(i)} - x^{(j)}||_2^2}{2\sigma^2}\right) \text{ for } i, j = 1, \ldots, m$$

Drawbacks: W is not sparse

Similarity graphs examples



Graph Laplacian

Unnormalized graph laplacian matrix:

$$L = D - W$$

Properties of L

- For every $f \in \mathbb{R}^n$, $f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$
- L is symmetric and positive semi-definite
- The smallest eigenvalue of L is 0 with eigenvector $\mathbf{1}$
- L has n real eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

Graph Laplacian

Proposition 4

Let G be an undirected graph with non-negative weights W, the multiplicity k if eigenvalue 0 of L is the number of connected components $A_1, \ldots, A_k$ in G.
The eigenspace of eigenvalue 0 is spanned by vectors $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$

(Normalized) Graph Laplacian

Normalized graph laplacian (Chung 1997) [1]:

$$L_{rw} = D^{-1}L = I - D^{-1}W$$

Properties of L_{rw}

- λ is an eigenvalue of L_{rw} with eigenvector v if and only if λ, v solve the generalized eigenproblem $Lv = \lambda Dv$
- 0 is an eigenvalue of L with eigenvector $\mathbf{1}$
- L_{rw} is positive semi-definite and has n non-negative eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

[1] Another definition of normalized graph Laplacian is $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$

(Normalized) Graph Laplacian

Normalized graph laplacian (Chung 1997) [1]:

$$L_{rw} = D^{-1}L = I - D^{-1}W$$

Properties of L_{rw}

- λ is an eigenvalue of L_{rw} with eigenvector v if and only if λ, v solve the generalized eigenproblem $Lv = \lambda Dv$
- 0 is an eigenvalue of L with eigenvector $\mathbf{1}$
- L_{rw} is positive semi-definite and has n non-negative eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

Proposition 5

Let G be an undirected graph with non-negative weights W, the multiplicity k of eigenvalue 0 of L_{rw} is the number of connected components $A_1, \ldots, A_k$ in G.
The eigenspace of eigenvalue 0 is spanned by vectors $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$

[1] Another definition of normalized graph Laplacian is $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$

Spectral Clustering Algorithm

Unormalized spectral clustering

Input: data points $x^{(1)}, \ldots, x^{(n)}$ and cluster size k

- Build a graph connecting $x^{(1)}, \ldots, x^{(n)}$ with weight W
- Compute first k eigenvectors $V = [v_1, \ldots, v_k]$ of L
- Define $y_i \in \mathbb{R}^k$ as the ith row of V, cluser $y_1, \ldots, y_n$ into k clusters $C_1, \ldots, C_k$ using k-means

Output: $A_1, \ldots, A_k$ where $A_i = \{j | y_j = C_i\}$

Spectral Clustering Algorithm

Normalized spectral clustering (Ng, Shi and Malik 2000)

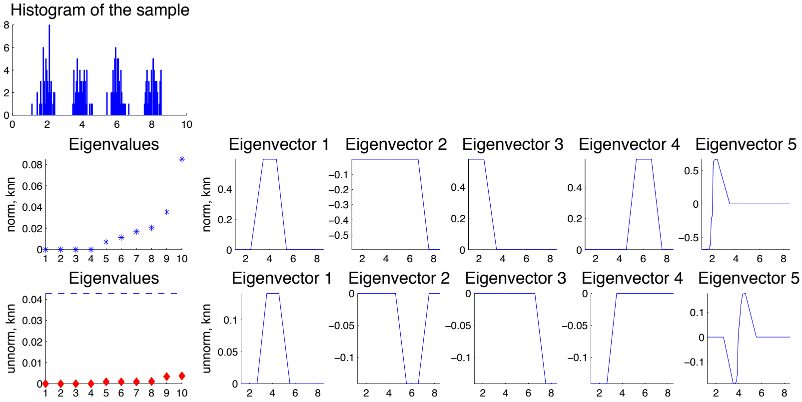
Input: data points $x^{(1)}, \ldots, x^{(n)}$ and cluster size k

- Build a graph connecting $x^{(1)}, \ldots, x^{(n)}$ with weight W
- Compute first k eigenvectors $V = [v_1, \ldots, v_k]$ of generalized eigen problem $Lv = \lambda Dv$
- Define $y_i \in \mathbb{R}^k$ as the ith row of V, cluser $y_1, \ldots, y_n$ into k clusters $C_1, \ldots, C_k$ using k-means

Output: $A_1, \ldots, A_k$ where $A_i = \{j | y_j = C_i\}$

Toy Example

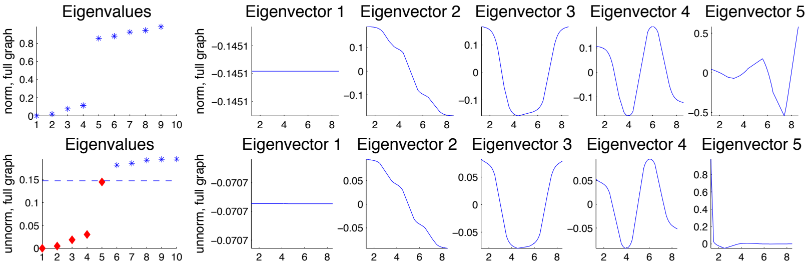
- 200 data points sampled from 4 Gaussian distributions
- KNN similarity graph ($k = 10$)



First 4 eigenvalues are 0 with eigenvectors 1_{A_i} , $i = 1, \ldots, 4$

Toy Example

- Fully connected graph with Gaussian similarity graph ($\sigma = 1$)



First eigenvector is **1** since the graph has only 1 connected component