

Machine Learning for Data Science (CS4786)

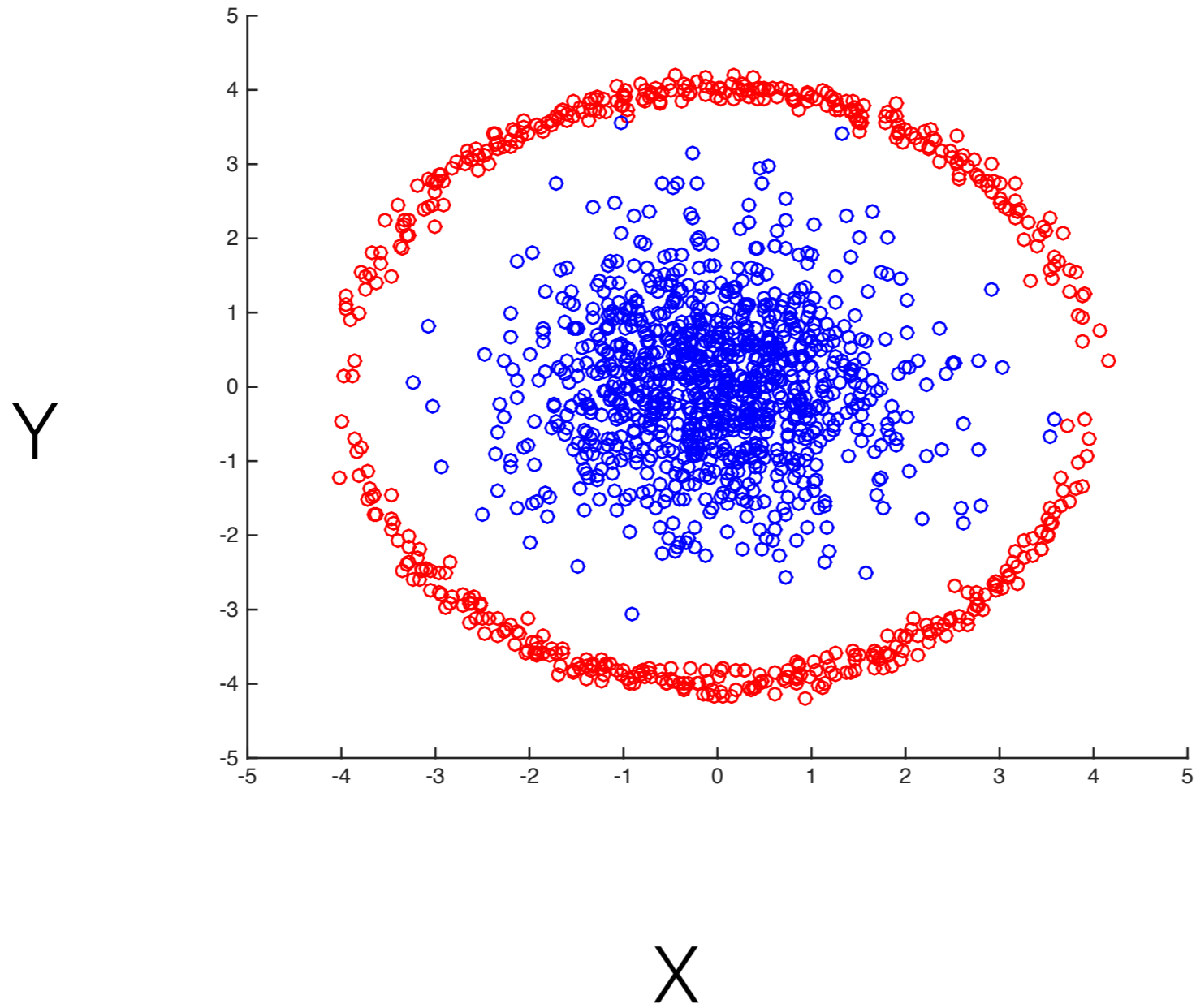
Lecture 13

Kernel PCA & Spectral Clustering

Course Webpage :

<http://www.cs.cornell.edu/Courses/cs4786/2017fa/>

EXAMPLE



A FIRST CUT

- Given $\mathbf{x}_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$\Phi(\mathbf{x}_t) = (\mathbf{x}_t[1], \dots, \mathbf{x}_t[d], \mathbf{x}_t[1] \cdot \mathbf{x}_t[1], \mathbf{x}_t[1] \cdot \mathbf{x}_t[2], \dots, \mathbf{x}_t[d] \cdot \mathbf{x}_t[d], \dots)^\top$$

A FIRST CUT

- Given $\mathbf{x}_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$\Phi(\mathbf{x}_t) = (\mathbf{x}_t[1], \dots, \mathbf{x}_t[d], \mathbf{x}_t[1] \cdot \mathbf{x}_t[1], \mathbf{x}_t[1] \cdot \mathbf{x}_t[2], \dots, \mathbf{x}_t[d] \cdot \mathbf{x}_t[d], \dots)^\top$$

- Enumerating products up to order K (ie. products of at most K coordinates) we can get degree K polynomials.

A FIRST CUT

- Given $\mathbf{x}_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$\Phi(\mathbf{x}_t) = (\mathbf{x}_t[1], \dots, \mathbf{x}_t[d], \mathbf{x}_t[1] \cdot \mathbf{x}_t[1], \mathbf{x}_t[1] \cdot \mathbf{x}_t[2], \dots, \mathbf{x}_t[d] \cdot \mathbf{x}_t[d], \dots)^\top$$

- Enumerating products up to order K (ie. products of at most K coordinates) we can get degree K polynomials.
- However dimension blows up as d^K

A FIRST CUT

- Given $\mathbf{x}_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$\Phi(\mathbf{x}_t) = (\mathbf{x}_t[1], \dots, \mathbf{x}_t[d], \mathbf{x}_t[1] \cdot \mathbf{x}_t[1], \mathbf{x}_t[1] \cdot \mathbf{x}_t[2], \dots, \mathbf{x}_t[d] \cdot \mathbf{x}_t[d], \dots)^\top$$

- Enumerating products up to order K (ie. products of at most K coordinates) we can get degree K polynomials.
- However dimension blows up as d^K
- Is there a way to do this without enumerating Φ ?

KERNEL TRICK

KERNEL TRICK

- Essence of Kernel trick:
 - If we can write down an algorithm only in terms of $\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ for data points \mathbf{x}_t and \mathbf{x}_s

KERNEL TRICK

- Essence of Kernel trick:
 - If we can write down an algorithm only in terms of $\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ for data points \mathbf{x}_t and \mathbf{x}_s
 - Then we don't need to explicitly enumerate $\Phi(\mathbf{x}_t)$'s but instead, compute $k(\mathbf{x}_t, \mathbf{x}_s) = \Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ (even if Φ maps to infinite dimensional space)

KERNEL TRICK

- Essence of Kernel trick:
 - If we can write down an algorithm only in terms of $\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ for data points \mathbf{x}_t and \mathbf{x}_s
 - Then we don't need to explicitly enumerate $\Phi(\mathbf{x}_t)$'s but instead, compute $k(\mathbf{x}_t, \mathbf{x}_s) = \Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ (even if Φ maps to infinite dimensional space)
- Example: RBF kernel $k(\mathbf{x}_t, \mathbf{x}_s) = \exp(-\sigma \|\mathbf{x}_t - \mathbf{x}_s\|_2^2)$, polynomial kernel $k(\mathbf{x}_t, \mathbf{x}_s) = (\mathbf{x}_t^\top \mathbf{y}_t)^p$

KERNEL TRICK

- Essence of Kernel trick:
 - If we can write down an algorithm only in terms of $\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ for data points \mathbf{x}_t and \mathbf{x}_s
 - Then we don't need to explicitly enumerate $\Phi(\mathbf{x}_t)$'s but instead, compute $k(\mathbf{x}_t, \mathbf{x}_s) = \Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ (even if Φ maps to infinite dimensional space)
- Example: RBF kernel $k(\mathbf{x}_t, \mathbf{x}_s) = \exp(-\sigma \|\mathbf{x}_t - \mathbf{x}_s\|_2^2)$, polynomial kernel $k(\mathbf{x}_t, \mathbf{x}_s) = (\mathbf{x}_t^\top \mathbf{y}_t)^p$
- Kernel function measures similarity between points.

LETS REWRITE PCA

- k^{th} column of W is eigenvector of covariance matrix

LETS REWRITE PCA

- k^{th} column of W is eigenvector of covariance matrix
That is, $\lambda_k W_k = \Sigma W_k$. Rewriting, for centered X

LETS REWRITE PCA

- k^{th} column of W is eigenvector of covariance matrix
That is, $\lambda_k W_k = \Sigma W_k$. Rewriting, for centered X

$$\lambda_k W_k = \frac{1}{n} \left(\sum_{t=1}^n \mathbf{x}_t \mathbf{x}_t^{\top} \right) W_k = \frac{1}{n} \sum_{t=1}^n (\mathbf{x}_t^{\top} W_k) \mathbf{x}_t$$

LETS REWRITE PCA

- k^{th} column of W is eigenvector of covariance matrix
That is, $\lambda_k W_k = \Sigma W_k$. Rewriting, for centered X

$$\lambda_k W_k = \frac{1}{n} \left(\sum_{t=1}^n \mathbf{x}_t \mathbf{x}_t^{\top} \right) W_k = \frac{1}{n} \sum_{t=1}^n (\mathbf{x}_t^{\top} W_k) \mathbf{x}_t$$

W_k 's can be written as linear combination of \mathbf{x}_t 's, as

$$W_k = \sum_{t=1}^n \alpha_k[t] \mathbf{x}_t$$

where $\alpha_k[t] = \frac{1}{\lambda_k n} (\mathbf{x}_t^{\top} W_k)$

LETS REWRITE PCA

- We have that $W_k = \sum_{s=1}^n \alpha_k[s] \mathbf{x}_s$ and that $\alpha_k[t] = \frac{1}{\lambda_k n} (\mathbf{x}_t^\top W_k)$.

LETS REWRITE PCA

- We have that $W_k = \sum_{s=1}^n \alpha_k[s] \mathbf{x}_s$ and that $\alpha_k[t] = \frac{1}{\lambda_k n} (\mathbf{x}_t^\top W_k)$.
- Hence:

$$\alpha_k[t] = \frac{1}{\lambda_k n} \left(\mathbf{x}_t^\top \left(\sum_{s=1}^n \alpha_k[s] \mathbf{x}_s \right) \right) = \frac{1}{\lambda_k n} \sum_{s=1}^n \alpha_k[s] \mathbf{x}_t^\top \mathbf{x}_s$$

LETS REWRITE PCA

- We have that $W_k = \sum_{s=1}^n \alpha_k[s] \mathbf{x}_s$ and that $\alpha_k[t] = \frac{1}{\lambda_k n} (\mathbf{x}_t^\top W_k)$.
- Hence:

$$\alpha_k[t] = \frac{1}{\lambda_k n} \left(\mathbf{x}_t^\top \left(\sum_{s=1}^n \alpha_k[s] \mathbf{x}_s \right) \right) = \frac{1}{\lambda_k n} \sum_{s=1}^n \alpha_k[s] \mathbf{x}_t^\top \mathbf{x}_s$$

- Let \tilde{K} be a matrix such that $\tilde{K}_{s,t} = \mathbf{x}_t^\top \mathbf{x}_s$. Hence, $\alpha_k[t] = \frac{1}{\lambda_k n} \alpha_k^\top \tilde{K}_t$ and

$$\alpha_k = \frac{1}{\lambda_k n} \tilde{K} \alpha_k$$

where \tilde{K}_t is the t 'th column of \tilde{K} .

LETS REWRITE PCA

- We have that $W_k = \sum_{s=1}^n \alpha_k[s] \mathbf{x}_s$ and that $\alpha_k[t] = \frac{1}{\lambda_k n} (\mathbf{x}_t^\top W_k)$.
- Hence:

$$\alpha_k[t] = \frac{1}{\lambda_k n} \left(\mathbf{x}_t^\top \left(\sum_{s=1}^n \alpha_k[s] \mathbf{x}_s \right) \right) = \frac{1}{\lambda_k n} \sum_{s=1}^n \alpha_k[s] \mathbf{x}_t^\top \mathbf{x}_s$$

- Let \tilde{K} be a matrix such that $\tilde{K}_{s,t} = \mathbf{x}_t^\top \mathbf{x}_s$. Hence, $\alpha_k[t] = \frac{1}{\lambda_k n} \alpha_k^\top \tilde{K}_t$ and

$$\alpha_k = \frac{1}{\lambda_k n} \tilde{K} \alpha_k$$

where \tilde{K}_t is the t 'th column of \tilde{K} .

- Hence α_k is in the direction of eigen vector of \tilde{K}

LETS REWRITE PCA

- Further, since W_k is unit norm,

$$1 = \|W_k\|_2^2 = \left(\sum_{t=1}^n \alpha_k[t] \mathbf{x}_t \right)^\top \left(\sum_{s=1}^n \alpha_k[s] \mathbf{x}_s \right) = \alpha_k^\top \tilde{K} \alpha_k = n \gamma_k \alpha_k^\top \alpha_k$$

Hence $\|\alpha_k\|^2 = \frac{1}{n \gamma_k}$ where γ_k is the k 'th eigen value of matrix \tilde{K}

LETS REWRITE PCA

- However W_k itself is in feature space and has the same dimensionality of $\Phi(x)$ (which is possibly infinite)!

LETS REWRITE PCA

- However W_k itself is in feature space and has the same dimensionality of $\Phi(x)$ (which is possibly infinite)!
- However, the projections are in K dimensions and we can hope to directly compute these as:

$$y_i[k] = \mathbf{x}_i^\top W_k = \sum_{t=1}^n \alpha_k[t] \tilde{K}_{t,i}$$

REWRITING PCA

- We assumed centered data, what if its not,

$$\begin{aligned}\tilde{K}_{s,t} &= \left(\mathbf{x}_t - \frac{1}{n} \sum_{u=1}^n \mathbf{x}_u \right)^\top \left(\mathbf{x}_s - \frac{1}{n} \sum_{u=1}^n \mathbf{x}_u \right) \\ &= \mathbf{x}_t^\top \mathbf{x}_s - \left(\frac{1}{n} \sum_{u=1}^n \mathbf{x}_u \right)^\top \mathbf{x}_s - \left(\frac{1}{n} \sum_{u=1}^n \mathbf{x}_u \right)^\top \mathbf{x}_t \\ &\quad + \frac{1}{n^2} \left(\sum_{u=1}^n \mathbf{x}_u \right)^\top \left(\sum_{v=1}^n \mathbf{x}_v \right) \\ &= \mathbf{x}_t^\top \mathbf{x}_s - \frac{1}{n} \sum_{u=1}^n \mathbf{x}_u^\top \mathbf{x}_s - \frac{1}{n} \sum_{u=1}^n \mathbf{x}_u^\top \mathbf{x}_t + \frac{1}{n^2} \sum_{u=1}^n \sum_{v=1}^n \mathbf{x}_u^\top \mathbf{x}_v\end{aligned}$$

REWRITING PCA

- Equivalently, if **Kern** is the matrix ($\text{Kern}_{t,s} = x_t^\top x_s$),

$$\tilde{K} = \text{Kern} - \frac{(\mathbf{1}_{n \times n} \times \text{Kern})}{n} - \frac{(\text{Kern} \times \mathbf{1}_{n \times n})}{n} + \frac{(\mathbf{1}_{n \times n} \times \text{Kern} \times \mathbf{1}_{n \times n})}{n^2}$$

PCA REWRITTEN

- Compute $\tilde{K} = \text{Kern} - \mathbf{1} \text{Kern}/n - \text{Kern} \mathbf{1}/n + \mathbf{1} \text{Kern} \mathbf{1}/n^2$

PCA REWRITTEN

- Compute $\tilde{K} = \text{Kern} - \mathbf{1} \text{ Kern}/n - \text{Kern} \mathbf{1}/n + \mathbf{1} \text{ Kern} \mathbf{1}/n^2$
- Compute top K eigen vectors P_1, \dots, P_K along with eigen values $\gamma_1, \dots, \gamma_K$ for the matrix \tilde{K}

PCA REWRITTEN

- Compute $\tilde{K} = \text{Kern} - \mathbf{1} \text{ Kern}/n - \text{Kern} \mathbf{1}/n + \mathbf{1} \text{ Kern} \mathbf{1}/n^2$
- Compute top K eigen vectors P_1, \dots, P_K along with eigen values $\gamma_1, \dots, \gamma_K$ for the matrix \tilde{K}
- Rescale each P_k by the inverse of the square-root of corresponding eigen values ie. $\alpha_k = P_k / \sqrt{n\gamma_k}$

PCA REWRITTEN

- Compute $\tilde{K} = \text{Kern} - \mathbf{1} \text{ Kern}/n - \text{Kern} \mathbf{1}/n + \mathbf{1} \text{ Kern} \mathbf{1}/n^2$
- Compute top K eigen vectors P_1, \dots, P_K along with eigen values $\gamma_1, \dots, \gamma_K$ for the matrix \tilde{K}
- Rescale each P_k by the inverse of the square-root of corresponding eigen values ie. $\alpha_k = P_k / \sqrt{n\gamma_k}$
- Compute projections by setting

$$y_i[k] = \sum_{t=1}^n \alpha_k[t] \tilde{K}_{t,i}$$

or in other words $Y = \tilde{K} \times [\alpha_1, \dots, \alpha_K]$

KERNEL PCA

KERNEL PCA

All we need to be able to compute, to perform PCA are $\mathbf{x}_t^T \mathbf{x}_s$

KERNEL PCA

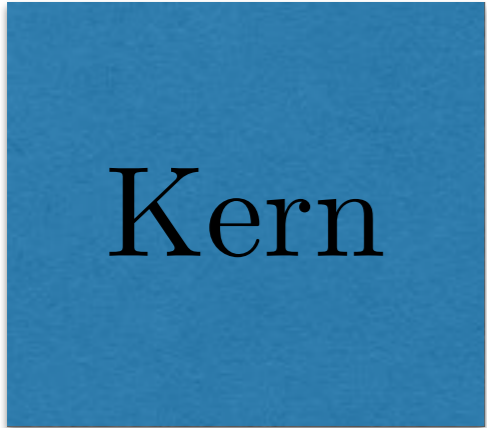
All we need to be able to compute, to perform PCA are $\mathbf{x}_t^\top \mathbf{x}_s$

Replace $\mathbf{x}_t^\top \mathbf{x}_s$ with $\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s) = k(x_t, x_s)$ to perform PCA
in feature space

KERNEL PCA

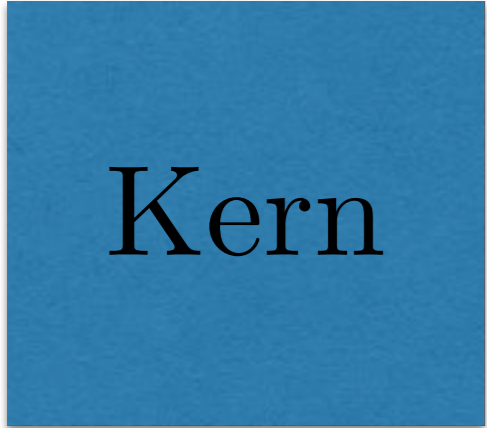
KERNEL PCA

1.

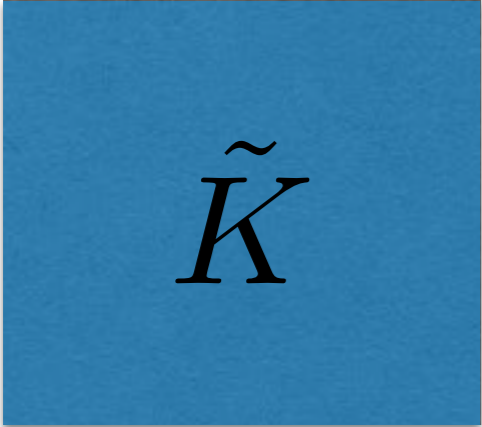

$$= \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_n) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ k(x_{n-1}, x_1) & k(x_{n-1}, x_2) & \dots & k(x_{n-1}, x_n) \\ k(x_n, x_1) & k(x_n, x_2) & \dots & k(x_n, x_n) \end{bmatrix}$$

KERNEL PCA

1.


$$= \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_n) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ k(x_{n-1}, x_1) & k(x_{n-1}, x_2) & \dots & k(x_{n-1}, x_n) \\ k(x_n, x_1) & k(x_n, x_2) & \dots & k(x_n, x_n) \end{bmatrix}$$

2.


$$= \text{Kern} - \frac{1}{n} (\mathbf{1} \text{ Kern} + \text{Kern} \mathbf{1}) + \frac{1}{n^2} \mathbf{1} \text{ Kern} \mathbf{1}$$

KERNEL PCA

KERNEL PCA

$$3. \left[\begin{array}{c} n \\ \mathbf{P} \\ K \end{array} , \gamma \right] = \text{eigs} \left(\begin{array}{c} \tilde{K} \\ K \end{array} \right)$$

KERNEL PCA

$$3. \left[\begin{array}{c} n \\ \mathbf{P} \\ K \end{array} ; \gamma \right] = \text{eigs} \left(\begin{array}{c} \tilde{K} \\ K \end{array} \right)$$

$$4. \begin{array}{c} n \\ \mathbf{a} \\ K \end{array} = n \begin{array}{c} \vdots \\ \vdots \\ \frac{P_1 \dots P_K}{\sqrt{n\gamma_1} \sqrt{n\gamma_K}} \\ \vdots \\ \vdots \\ K \end{array}$$

KERNEL PCA

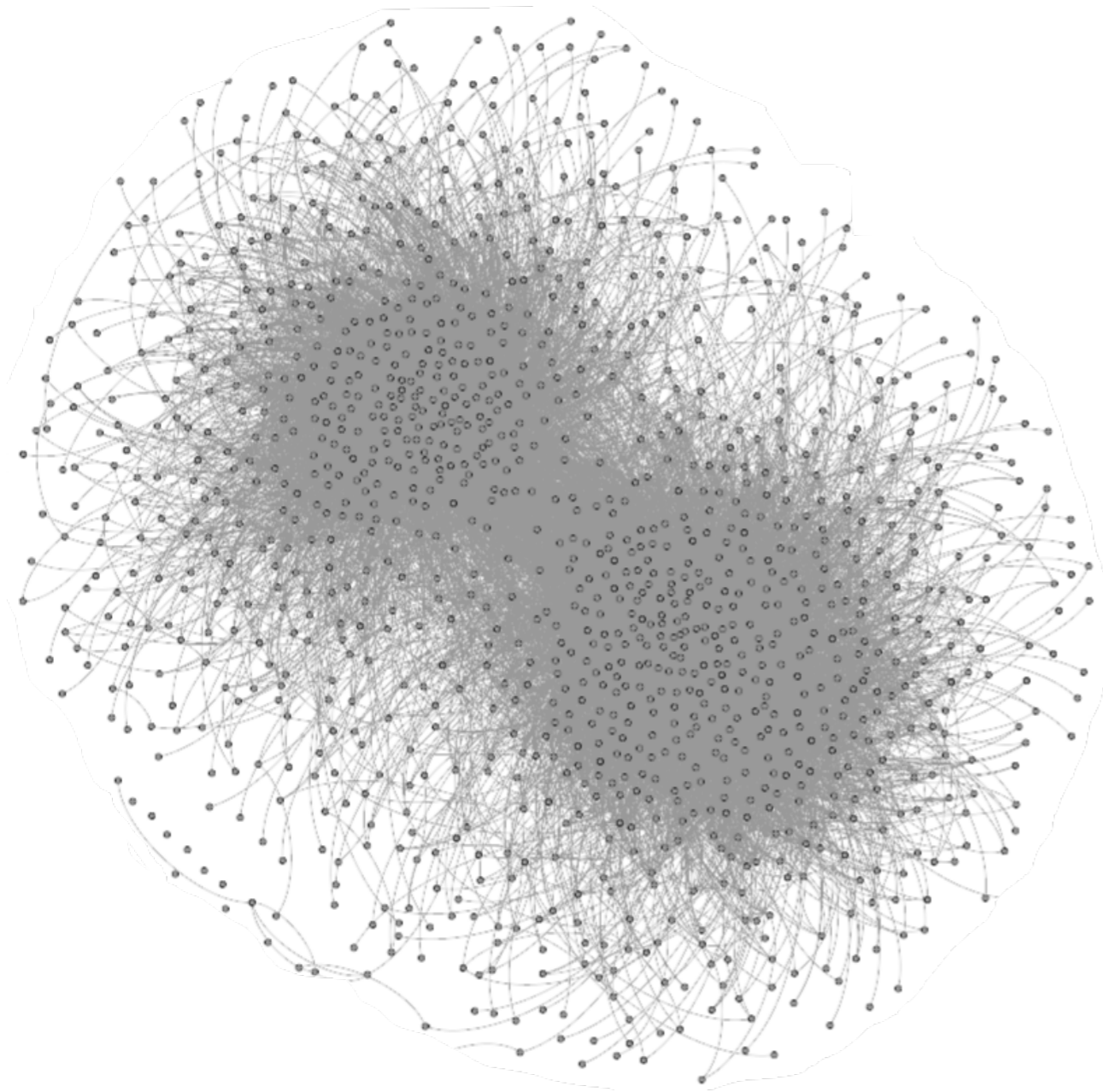
$$3. \begin{bmatrix} n \\ P \\ K, \gamma \end{bmatrix} = \text{eigs} \left(\begin{bmatrix} \tilde{K} \\ K \end{bmatrix} \right)$$

$$4. \begin{bmatrix} n \\ \alpha \\ K \end{bmatrix} = n \begin{bmatrix} P_1 \dots P_K \\ \sqrt{n\gamma_1} \dots \sqrt{n\gamma_K} \\ K \end{bmatrix}$$

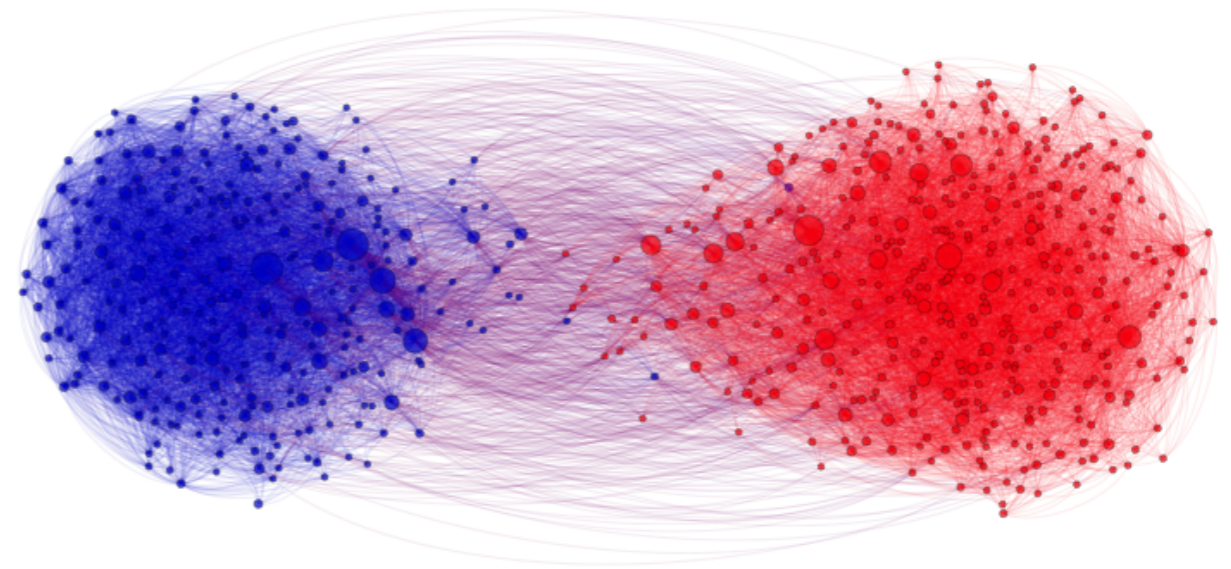
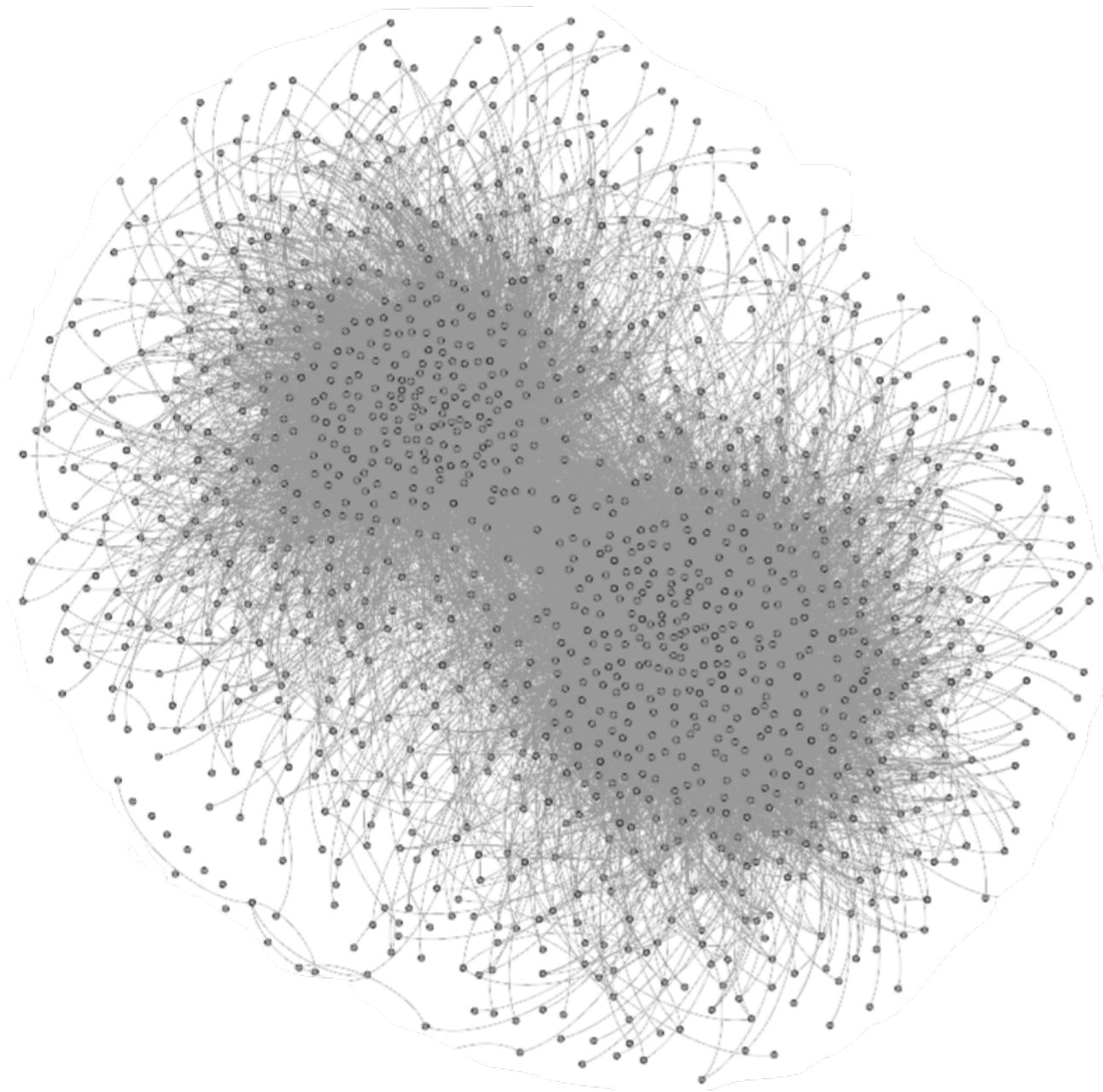
$$5. \begin{bmatrix} n \\ Y \\ K \end{bmatrix} = n \begin{bmatrix} \tilde{K} \\ n \end{bmatrix} \times \begin{bmatrix} n \\ \alpha \\ K \end{bmatrix}$$

Demo

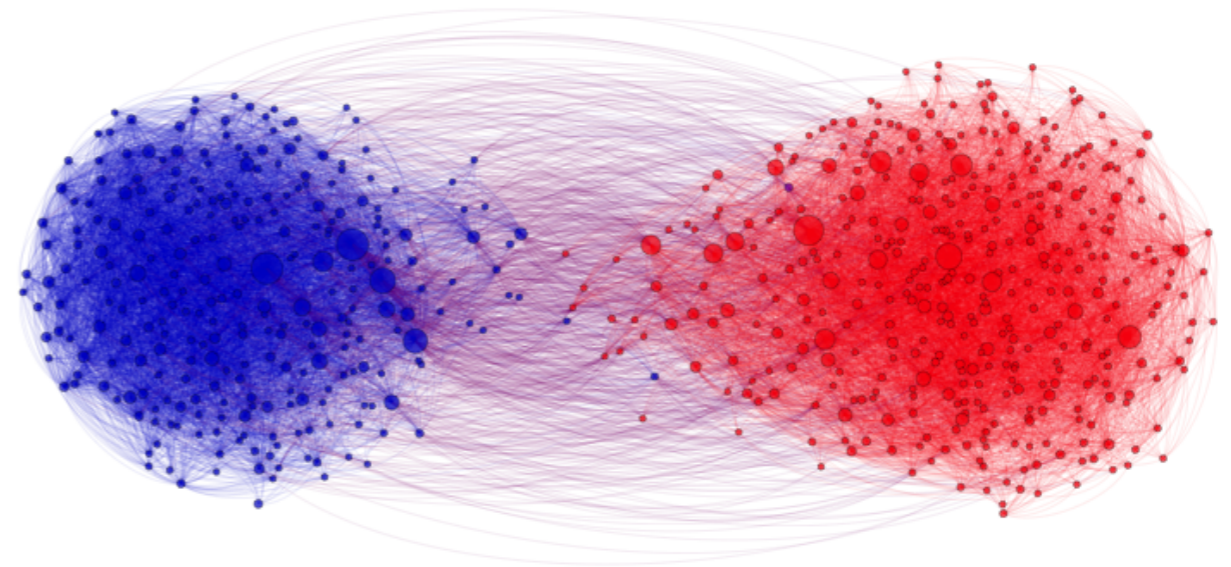
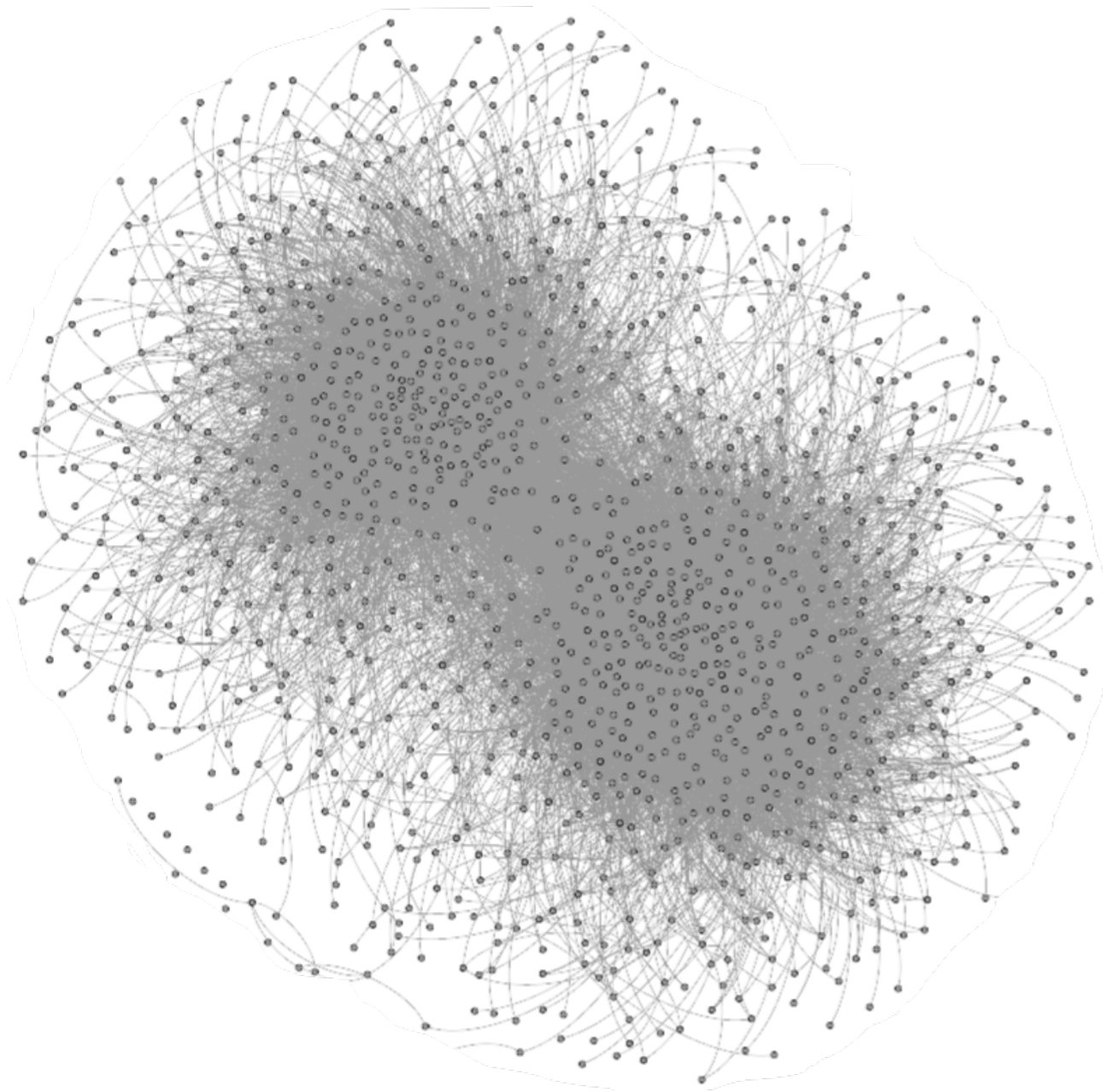
SPECTRAL CLUSTERING



SPECTRAL CLUSTERING



SPECTRAL CLUSTERING

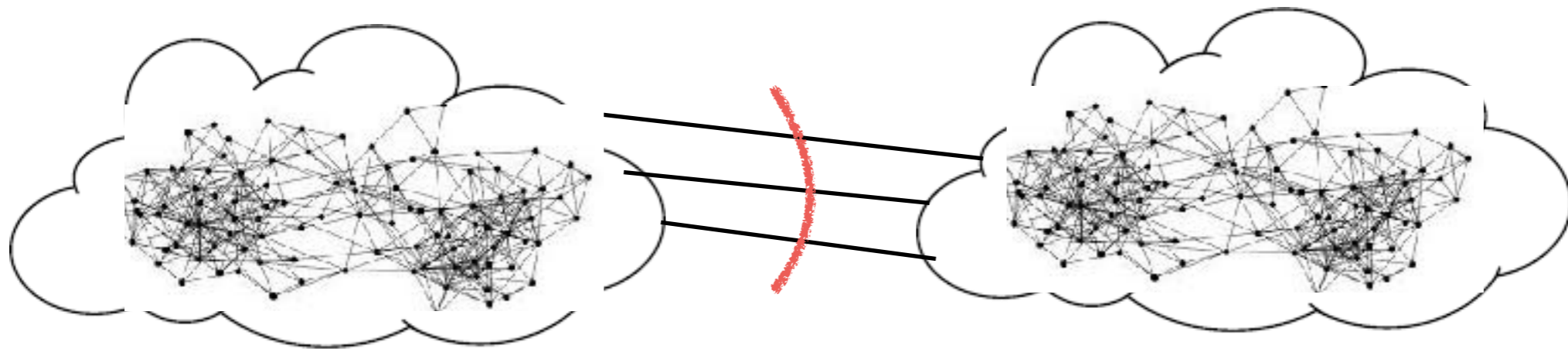


- Cluster nodes in a graph.
- Analysis of social network data.

Steps

- Map nodes to K dimensional space
 - Spectral embedding
- Use clustering on the K dimensional space

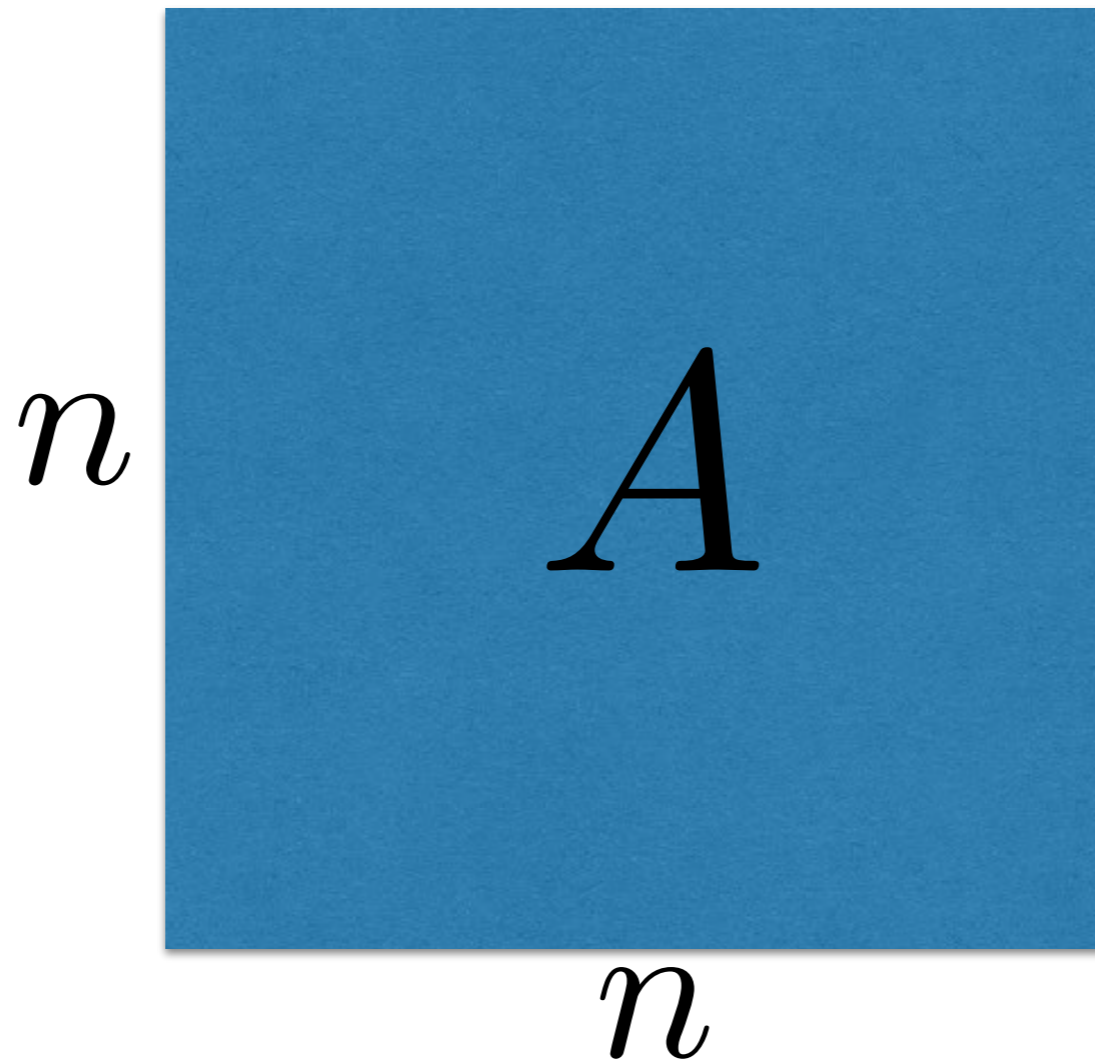
What is the Embedding?



- Map each node in V to \mathbb{R}^k
- Nodes linked to each other are close
- Disconnected groups of nodes are far from each other

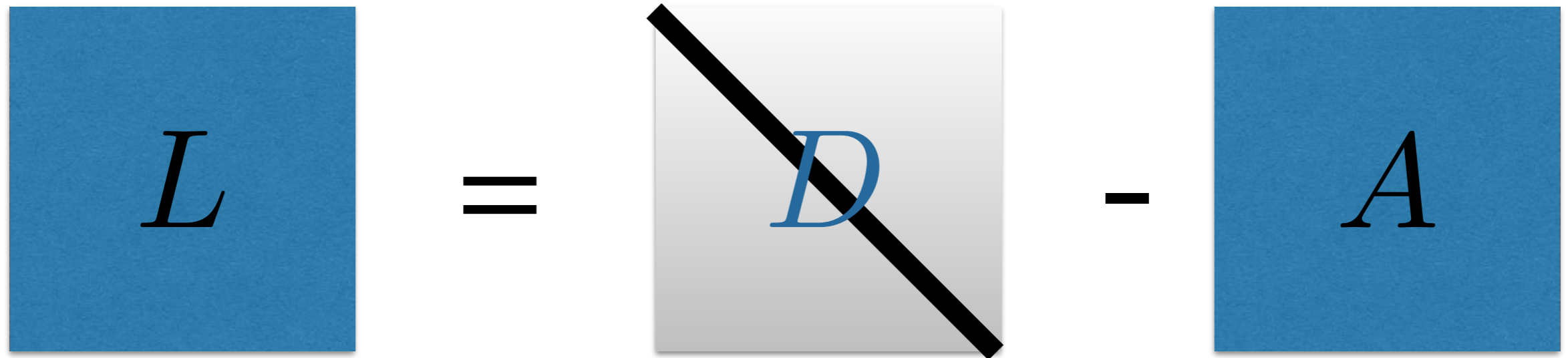
SPECTRAL CLUSTERING

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$



A is adjacency matrix of a graph

SPECTRAL CLUSTERING

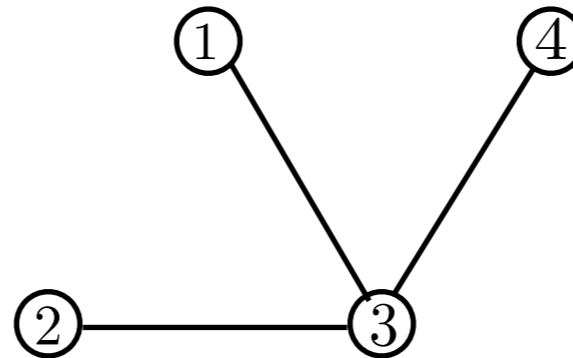
$$L = D - A$$
The diagram illustrates the equation $L = D - A$ using three square boxes. The first box on the left is blue and contains the letter L . To its right is an equals sign. The second box is gray and contains the letter D , but it is crossed out with a thick black diagonal line from the top-left to the bottom-right. To the right of this box is a minus sign. The final box on the right is blue and contains the letter A .

SPECTRAL CLUSTERING

The diagram illustrates the relationship between the Laplacian matrix L , the degree matrix D , and the adjacency matrix A . On the left is a blue square containing the letter L . This is followed by an equals sign. In the center is a light gray square containing the letter D , which is crossed out with a thick black diagonal line. To the right of this is a minus sign, followed by a blue square containing the letter A .

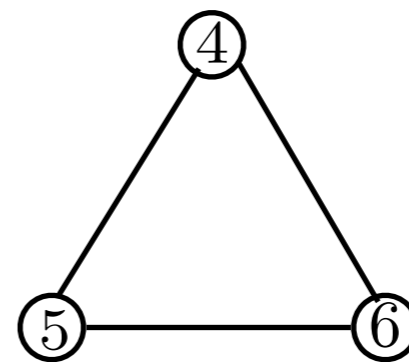
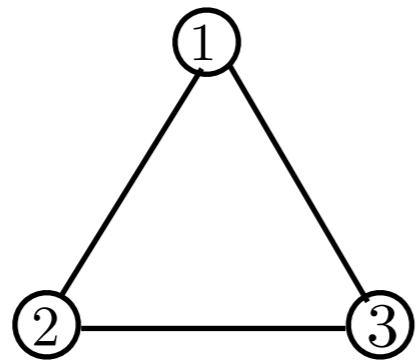
$$D_{i,i} = \sum_{j=1}^n A_{i,j}$$

GRAPH CLUSTERING



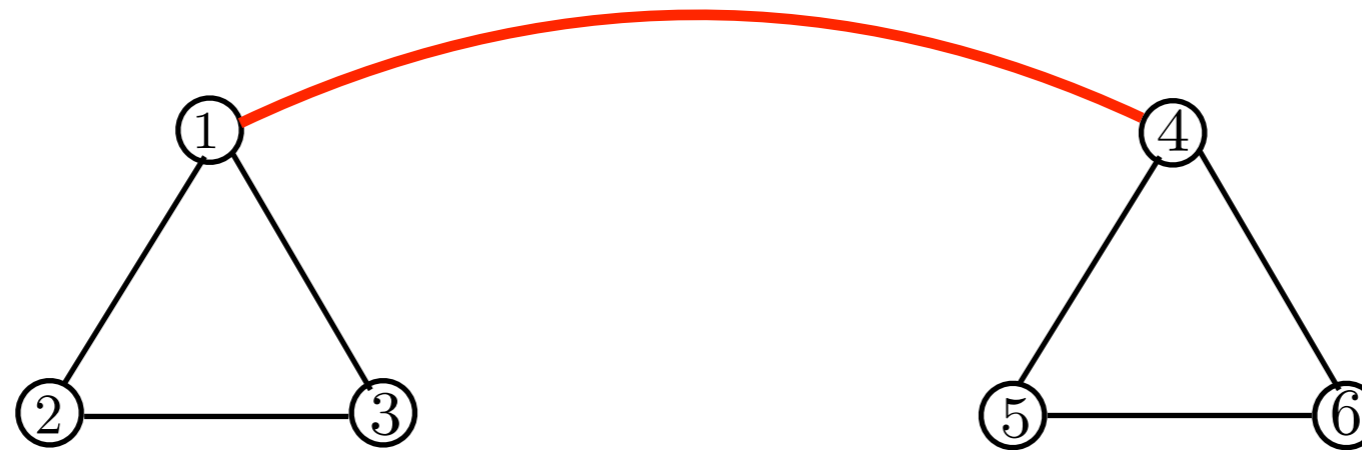
- Fact: For a connected graph, exactly one, the smallest of eigenvalues is 0 , corresponding eigenvector is $\mathbf{1} = (1, \dots, 1)^\top$
Proof: Sum of each row of L is 0 because $D_{i,i} = \sum_{j=1}^n A_{i,j}$ and $L = D - A$

GRAPH CLUSTERING



- Fact: For general graph, number of 0 eigenvalues correspond to number of connected components. The corresponding eigenvectors are all 1's on the nodes of connected components
Proof: L is block diagonal. Use connected graph result on each component.

GRAPH CLUSTERING



- Fact: For general graph, number of 0 eigenvalues correspond to number of connected components. The corresponding eigenvectors are all 1's on the nodes of connected components
Proof: L is block diagonal. Use connected graph result on each component.

Spectral Embedding

- Nodes linked to each other are close
- What has this got to do with Laplacian matrix?

CUTS AND LAPLACIAN

$$\text{Obj}(c) = \frac{1}{2} \sum_{(i,j) \in E} (c_i - c_j)^2$$

CUTS AND LAPLACIAN

$$\begin{aligned}\text{Obj}(c) &= \frac{1}{2} \sum_{(i,j) \in E} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i - c_j)^2\end{aligned}$$

CUTS AND LAPLACIAN

$$\begin{aligned}\text{Obj}(c) &= \frac{1}{2} \sum_{(i,j) \in E} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i^2 + c_j^2 - 2c_i c_j)\end{aligned}$$

CUTS AND LAPLACIAN

$$\begin{aligned}\text{Obj}(c) &= \frac{1}{2} \sum_{(i,j) \in E} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i^2 + c_j^2 - 2c_i c_j) \\ &= \frac{1}{2} \sum_{i=1}^n \left(\sum_{j=1}^n A_{i,j} \right) c_i^2 + \frac{1}{2} \sum_{j=1}^n \left(\sum_{i=1}^n A_{i,j} \right) c_j^2 - \sum_{i=1}^n \sum_{j=1}^n A_{i,j} c_i c_j\end{aligned}$$

CUTS AND LAPLACIAN

$$\begin{aligned}\text{Obj}(c) &= \frac{1}{2} \sum_{(i,j) \in E} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i^2 + c_j^2 - 2c_i c_j) \\ &= \frac{1}{2} \sum_{i=1}^n \left(\sum_{j=1}^n A_{i,j} \right) c_i^2 + \frac{1}{2} \sum_{j=1}^n \left(\sum_{i=1}^n A_{i,j} \right) c_j^2 - \sum_{i=1}^n \sum_{j=1}^n A_{i,j} c_i c_j \\ &= \frac{1}{2} \sum_{i=1}^n D_{i,i} c_i^2 + \frac{1}{2} \sum_{j=1}^n D_{j,j} c_j^2 - \sum_{i=1}^n \sum_{j=1}^n A_{i,j} c_i c_j\end{aligned}$$

CUTS AND LAPLACIAN

$$\begin{aligned}\text{Obj}(c) &= \frac{1}{2} \sum_{(i,j) \in E} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i - c_j)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} (c_i^2 + c_j^2 - 2c_i c_j) \\ &= \frac{1}{2} \sum_{i=1}^n \left(\sum_{j=1}^n A_{i,j} \right) c_i^2 + \frac{1}{2} \sum_{j=1}^n \left(\sum_{i=1}^n A_{i,j} \right) c_j^2 - \sum_{i=1}^n \sum_{j=1}^n A_{i,j} c_i c_j \\ &= \frac{1}{2} \sum_{i=1}^n D_{i,i} c_i^2 + \frac{1}{2} \sum_{j=1}^n D_{j,j} c_j^2 - \sum_{i=1}^n \sum_{j=1}^n A_{i,j} c_i c_j \\ &= c^\top D c - c^\top A c = c^\top L c\end{aligned}$$

SPECTRAL CLUSTERING, $K = 1$

Hence to find the solution we need to solve for

$$\text{Minimize } c^T L c \quad \text{s.t.} \quad \|c\| = 1$$

SPECTRAL CLUSTERING, $K = 1$

Hence to find the solution we need to solve for

$$\text{Minimize } c^T L c \quad \text{s.t.} \quad \|c\| = 1$$

Hence solution c to above is an Eigen vector, first smallest one is the all 1's vector (for connected graph), second smallest one is our solution

To get clustering assignment we simply threshold at 0

SPECTRAL CLUSTERING, $K > 1$

- Solution obtained by considering the second smallest up to K^{th} smallest eigenvectors

$$\text{Obj}(c) = \sum_{k=1}^K c^k \top L c^k$$

c^k 's are orthogonal to each other and the all ones vector

SPECTRAL CLUSTERING ALGORITHM (UNNORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the Laplacian matrix $L = D - A$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of L (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

SPECTRAL CLUSTERING ALGORITHM (UNNORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the Laplacian matrix $L = D - A$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of L (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

$\mathbf{y}_1, \dots, \mathbf{y}_n$ are called spectral embedding

SPECTRAL CLUSTERING ALGORITHM (UNNORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the Laplacian matrix $L = D - A$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of L (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

$\mathbf{y}_1, \dots, \mathbf{y}_n$ are called spectral embedding

Embeds the n nodes into $K-1$ dimensional vectors