Kernel PCA & Spectral Clustering

Course Webpage:
http://www.cs.cornell.edu/Courses/cs4786/2017fa/
Given \( x_t \in \mathbb{R}^d \), the feature space vector is given by mapping

\[
\Phi(x_t) = (x_t[1], \ldots, x_t[d], x_t[1] \cdot x_t[1], x_t[1] \cdot x_t[2], \ldots, x_t[d] \cdot x_t[d], \ldots)^\top
\]
Given $x_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$
\Phi(x_t) = (x_t[1], \ldots, x_t[d], x_t[1] \cdot x_t[1], x_t[1] \cdot x_t[2], \ldots, x_t[d] \cdot x_t[d], \ldots)^	op
$$

Enumerating products up to order $K$ (ie. products of at most $K$ coordinates) we can get degree $K$ polynomials.
Given \( x_t \in \mathbb{R}^d \), the feature space vector is given by mapping

\[
\Phi(x_t) = (x_t[1], \ldots, x_t[d], x_t[1] \cdot x_t[1], x_t[1] \cdot x_t[2], \ldots, x_t[d] \cdot x_t[d], \ldots)^T
\]

Enumerating products up to order \( K \) (i.e., products of at most \( K \) coordinates) we can get degree \( K \) polynomials.

However dimension blows up as \( d^K \)
Given $x_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$\Phi(x_t) = (x_t[1], \ldots, x_t[d], x_t[1] \cdot x_t[1], x_t[1] \cdot x_t[2], \ldots, x_t[d] \cdot x_t[d], \ldots)^\top$$

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 $\Phi$?
Essence of Kernel trick:

If we can write down an algorithm only in terms of $(x_t)^T(x_s)$ for data points $x_t$ and $x_s$, then we don't need to explicitly enumerate $(x_t)$'s but instead, compute $k(x_t, x_s) = (x_t)^T(x_s)$ (even if $x$ maps to infinite dimensional space).

Example: RBF kernel

$$k(x_t, x_s) = \exp\left(-\frac{(x_t - x_s)^2}{2}\right),$$

degree polynomial kernel

$$k(x_t, x_s) = \langle x_t, y_t \rangle^p$$

Kernel function measures similarity between points.
**Kernel Trick**

- **Essence of Kernel trick:**
  - **If** we can write down an algorithm only in terms of \( \Phi(x_t)^\top \Phi(x_s) \)
    for data points \( x_t \) and \( x_s \)

Example: RBF kernel

\[
\text{RBF kernel: } k(x_t, x_s) = \exp(-\|x_t - x_s\|^2)
\]

Polynomial kernel

\[
\text{Polynomial kernel: } k(x_t, x_s) = (\langle x_t, x_s \rangle)^p
\]
Kernel Trick

- Essence of Kernel trick:
  - **If** we can write down an algorithm only in terms of \( \Phi(x_t) \top \Phi(x_s) \) for data points \( x_t \) and \( x_s \)
  - **Then** we don’t need to explicitly enumerate \( \Phi(x_t) \)’s but instead, compute \( k(x_t, x_s) = \Phi(x_t) \top \Phi(x_s) \) (even if \( \Phi \) maps to infinite dimensional space)

- Example: RBF kernel
  \[ k(x_t, x_s) = \exp\left(-\frac{(x_t - x_s)^2}{2}ight) \]

- Polynomial kernel
  \[ k(x_t, x_s) = x_t \top y_t \]
Essence of Kernel trick:

- **If** we can write down an algorithm only in terms of $\Phi(x_t)^\top \Phi(x_s)$ for data points $x_t$ and $x_s$

- **Then** we don’t need to explicitly enumerate $\Phi(x_t)$’s but instead, compute $k(x_t, x_s) = \Phi(x_t)^\top \Phi(x_s)$ (even if $\Phi$ maps to infinite dimensional space)

Example: RBF kernel $k(x_t, x_s) = \exp(-\sigma \|x_t - x_s\|_2^2)$, polynomial kernel $k(x_t, x_s) = (x_t^\top y_t)^p$
Essence of Kernel trick:

- **If** we can write down an algorithm only in terms of $\Phi(x_t)^\top \Phi(x_s)$ for data points $x_t$ and $x_s$

- **Then** we don’t need to explicitly enumerate $\Phi(x_t)$’s but instead, compute $k(x_t, x_s) = \Phi(x_t)^\top \Phi(x_s)$ (even if $\Phi$ maps to infinite dimensional space)

- Example: RBF kernel $k(x_t, x_s) = \exp(-\sigma \|x_t - x_s\|_2^2)$, polynomial kernel $k(x_t, x_s) = (x_t^\top y_t)^p$

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

$$\lambda_k W_k = \frac{1}{n} \left( \sum_{t=1}^{n} x_t x_t^\top \right) W_k = \frac{1}{n} \sum_{t=1}^{n} (x_t^\top W_k) x_t$$
$k^{th}$ column of $W$ is eigenvector of covariance matrix
That is, $\lambda_k W_k = \Sigma W_k$. Rewriting, for centered $X$

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

$W_k$’s can be written as linear combination of $x_t$’s, as

$$W_k = \sum_{t=1}^{n} \alpha_k[t] x_t$$

where $\alpha_k[t] = \frac{1}{\lambda_k n} (x_t^\top W_k)$
We have that $W_k = \sum_{s=1}^{n} \alpha_k[s]x_s$ and that $\alpha_k[t] = \frac{1}{\lambda_{kn}} (x_t^\top W_k)$.
We have that \( W_k = \sum_{s=1}^{n} \alpha_k[s] x_s \) and that \( \alpha_k[t] = \frac{1}{\lambda_k n} (x_t^\top W_k) \).

Hence:

\[
\alpha_k[t] = \frac{1}{\lambda_k n} \left( x_t^\top \left( \sum_{s=1}^{n} \alpha_k[s] x_s \right) \right) = \frac{1}{\lambda_k n} \sum_{s=1}^{n} \alpha_k[s] x_t^\top x_s
\]
We have that $\mathbf{W}_k = \sum_{s=1}^{n} \alpha_k[s] \mathbf{x}_s$ and that $\alpha_k[t] = \frac{1}{\lambda_k n} (\mathbf{x}_t^\top \mathbf{W}_k)$.

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

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

where $\tilde{\mathbf{K}}_t$ is the $t$’th column of $\tilde{\mathbf{K}}$. 
We have that $W_k = \sum_{s=1}^{n} \alpha_k[s]x_s$ and that $\alpha_k[t] = \frac{1}{\lambda_k n} (x_t^\top W_k)$.

Hence:

$$\alpha_k[t] = \frac{1}{\lambda_k n} \left( x_t^\top \left( \sum_{s=1}^{n} \alpha_k[s]x_s \right) \right) = \frac{1}{\lambda_k n} \sum_{s=1}^{n} \alpha_k[s]x_t^\top x_s$$

Let $\tilde{K}$ be a matrix such that $\tilde{K}_{s,t} = x_t^\top x_s$. 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 $\alpha_k$ is in the direction of eigen vector of $\tilde{K}$.
Further, since $W_k$ is unit norm,

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

Hence $\| \alpha_k \|_2^2 = \frac{1}{n \gamma_k}$ where $\gamma_k$ is the $k$'th eigen value of matrix $\tilde{K}$
However $W_k$ itself is in feature space and has the same dimensionality of $\Phi(x)$ (which is possibly infinite)!
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:

\[
y_i[k] = x_i^\top W_k = \sum_{t=1}^{n} \alpha_k[t] \tilde{K}_{t,i}
\]
We assumed centered data, what if it’s not,

\[
\tilde{K}_{s,t} = \left( x_t - \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top \left( x_s - \frac{1}{n} \sum_{u=1}^{n} x_u \right) \\
= x_t^\top x_s - \left( \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top x_s - \left( \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top x_t \\
+ \frac{1}{n^2} \left( \sum_{u=1}^{n} x_u \right)^\top \left( \sum_{v=1}^{n} x_v \right) \\
= x_t^\top x_s - \frac{1}{n} \sum_{u=1}^{n} x_u^\top x_s - \frac{1}{n} \sum_{u=1}^{n} x_u^\top x_t + \frac{1}{n^2} \sum_{u=1}^{n} \sum_{v=1}^{n} x_u^\top x_v
\]
Equivalently, if \( \text{Kern} \) is the matrix (\( \text{Kern}_{t,s} = x_t^T x_s \)),

\[
\tilde{K} = \text{Kern} - \frac{(1_{n\times n} \times \text{Kern})}{n} - \frac{(	ext{Kern} \times 1_{n\times n})}{n} + \frac{(1_{n\times n} \times \text{Kern} \times 1_{n\times n})}{n^2}
\]
If we only need to compute projections of data points, it's enough to have access to matrix $\tilde{K}$ (an $n \times n$ matrix).

1. Compute the top $K$ eigen vectors $P_1, \ldots, P_K$ along with eigen values $1, \ldots, K$ for the matrix $\tilde{K}$.
2. Rescale each $P_k$ by the inverse of the square-root of corresponding eigen values: $P_k = P_k / \sqrt{n}$.
3. Compute projections by setting $y_i[k] = \sum_{t=1}^{n} P_k[t] \tilde{K}_{t,i}$, or in other words $Y = \tilde{K} \times [1, \ldots, 1]$.

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

*PCA Rewritten*
PCA Rewritten

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

- Compute top \( K \) eigen vectors \( P_1, \ldots, P_K \) along with eigen values \( \gamma_1, \ldots, \gamma_K \) for the matrix \( \tilde{K} \)
PCA Rewritten

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

- Compute top $K$ eigen vectors $P_1, \ldots, P_K$ along with eigen values $\gamma_1, \ldots, \gamma_K$ for the matrix $\tilde{K}$

- Rescale each $P_k$ by the inverse of the square-root of corresponding eigen values i.e. $\alpha_k = P_k/\sqrt{n\gamma_k}$
Compute $\tilde{K} = \text{Kern} - 1 \ \text{Kern}/n - \text{Kern} 1/n + 1 \ \text{Kern} 1/n^2$

Compute top $K$ eigen vectors $P_1, \ldots, P_K$ along with eigen values $\gamma_1, \ldots, \gamma_K$ 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 $Y = \tilde{K} \times [\alpha_1, \ldots, \alpha_K]$
All we need to be able to compute, to perform PCA are $x^t x_s$. Replace $x^t x_s$ with $(x^t) (x_s) = k(x^t, x_s)$ to perform PCA in feature space.
All we need to be able to compute, to perform PCA are $x_t^\top x_s$
All we need to be able to compute, to perform PCA are $x_t^\top x_s$

Replace $x_t^\top x_s$ with $\Phi(x_t)^\top \Phi(x_s) = k(x_t, x_s)$ to perform PCA in feature space
If we want to port PCA to kernel PCA, we need to be able to write $\tilde{K}$ in terms of kernel functions. We assumed centered data, so

$$\tilde{K}(s, t) = \sum_{u=1}^{n} (x_u^s - \frac{1}{n} \sum_{v=1}^{n} x_v^s) (x_u^t - \frac{1}{n} \sum_{v=1}^{n} x_v^t) + \frac{1}{n^2} \sum_{u=1}^{n} \sum_{v=1}^{n} k(x_u^s, x_v^t).$$

Knowing kernel function, we can perform Kernel PCA even when it maps to infinite dimensional space!
If we want to port PCA to kernel PCA, we need to be able to write $\tilde{K}$ in terms of kernel functions. We assumed centered data, so

$$\tilde{K}_{st},t=\left(\chi_t-\frac{1}{n}\sum_{u=1}^{n}\chi_u\right)\left(\chi_s-\frac{1}{n}\sum_{u=1}^{n}\chi_u\right) - \frac{1}{n^2}\sum_{u=1}^{n}\sum_{v=1}^{n}\left(\chi_u-\frac{1}{n}\sum_{u'=1}^{n}\chi_{u'}\right)\left(\chi_v-\frac{1}{n}\sum_{v'=1}^{n}\chi_{v'}\right) = k(\chi_t,\chi_s)$$

Knowing kernel function, we can perform Kernel PCA even when maps to infinite dimensional space!
If we want to port PCA to kernel PCA, we need to be able to write \( \tilde{K} \) in terms of kernel functions.

We assumed centered data, so

\[
\tilde{K}_{s,t} = \left( x_t - \frac{1}{n} \sum_{\mu=1}^{n} x_{\mu} \right) \left( x_s - \frac{1}{n} \sum_{\mu=1}^{n} x_{\mu} \right) - \frac{1}{n^2} \left( \sum_{\mu=1}^{n} x_{\mu} \right) \left( \sum_{\nu=1}^{n} x_{\nu} \right)
\]

Knowing kernel function, we can perform Kernel PCA even when maps to infinite dimensional space!
KERNEL PCA
3. $\begin{bmatrix} P \mid K \end{bmatrix} = \text{eigs}(\tilde{K}, K)$
Kernel PCA

3. \[ \begin{bmatrix} P \end{bmatrix}_{K} \gamma = \text{eigs} \left( \hat{K}, K \right) \]

4. \[ \begin{bmatrix} \alpha \end{bmatrix}_{K} = \begin{bmatrix} \frac{P_1}{\sqrt{n\gamma_1}}, \ldots, \frac{P_K}{\sqrt{n\gamma_K}} \end{bmatrix}_{K} \]
Kernel PCA

3. \[
\begin{bmatrix} P \end{bmatrix}^{n \times K} = \text{eigs} \left( \begin{bmatrix} \tilde{K} \end{bmatrix} + \begin{bmatrix} K \end{bmatrix} \right)
\]

4. \[
\begin{bmatrix} \alpha \end{bmatrix}^{n \times K} = \begin{bmatrix} \frac{P_1}{\sqrt{n \gamma_1}} \ldots \frac{P_K}{\sqrt{n \gamma_K}} \end{bmatrix}
\]

5. \[
\begin{bmatrix} Y \end{bmatrix}^{n \times K} = \begin{bmatrix} \tilde{K} \end{bmatrix}^{n \times n} \times \begin{bmatrix} \alpha \end{bmatrix}^{n \times K}
\]
Demo
Spectral Clustering

Cluster nodes in a graph.

Analysis of social network data.

**Spectral Clustering**

Input: Similarity matrix $A_{i,j} = A_{j,i} > 0$ indicates similarity between elements $x_i$ and $x_j$.

Example:

$A_{i,j} = \exp(-d(x_i, x_j))$.

$A$ is adjacency matrix of a graph.
Spectral Clustering

Cluster nodes in a graph.

Analysis of social network data.

Spectral Clustering

Input: Similarity matrix $A_{ij}, A_{ji} > 0$ indicates similarity between elements $x_i$ and $x_j$.

Example:

$A_{ij} = \exp(-d(x_i, x_j))$

$A$ is adjacency matrix of a graph
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

Input: Similarity matrix

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

\( A \) is adjacency matrix of a graph
Spectral Clustering

Input: Similarity matrix $A_{i,j} = A_{j,i} > 0$ indicates similarity between elements $x_i$ and $x_j$.

Example:

$$A_{i,j} = \exp(-d(x_i, x_j))$$

$A$ is the adjacency matrix of a graph.

$$L = D - A$$
Spectral Clustering

Input: Similarity matrix $A_{i,j} = A_{j,i} > 0$ indicates similarity between elements $x_i$ and $x_j$

Example:

$A_{i,j} = \exp(-d(x_i, x_j))$

$L = D - A$

$D_{i,i} = \sum_{j=1}^{n} A_{i,j}$
Fact: For a connected graph, exactly one, the smallest of eigenvalues is $0$, corresponding eigenvector is $\mathbf{1} = (1, \ldots, 1)^\top$

Proof: Sum of each row of $L$ is $0$ because $D_{i,i} = \sum_{j=1}^{n} A_{i,j}$ and $L = D - A$
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.
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

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

\[ = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i - c_j)^2 \]
Cuts and Laplacian

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

\[ = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i - c_j)^2 \]

\[ = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i^2 + c_j^2 - 2c_ic_j) \]
Cuts and Laplacian

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

\[
= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i - c_j)^2
\]

\[
= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i^2 + c_j^2 - 2c_i c_j)
\]

\[
= \frac{1}{2} \sum_{i=1}^{n} \left( \sum_{j=1}^{n} A_{i,j} \right) c_i^2 + \frac{1}{2} \sum_{j=1}^{n} \left( \sum_{i=1}^{n} A_{i,j} \right) c_j^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} c_i c_j
\]
Cuts and Laplacian

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

\[ = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i - c_j)^2 \]

\[ = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i^2 + c_j^2 - 2c_i c_j) \]

\[ = \frac{1}{2} \sum_{i=1}^{n} \left( \sum_{j=1}^{n} A_{i,j} \right) c_i^2 + \frac{1}{2} \sum_{j=1}^{n} \left( \sum_{i=1}^{n} A_{i,j} \right) c_j^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} c_i c_j \]

\[ = \frac{1}{2} \sum_{i=1}^{n} D_{i,i} c_i^2 + \frac{1}{2} \sum_{j=1}^{n} D_{j,j} c_j^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} c_i c_j \]
Cuts and Laplacian

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

\[ = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i - c_j)^2 \]

\[ = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} (c_i^2 + c_j^2 - 2c_i c_j) \]

\[ = \frac{1}{2} \sum_{i=1}^{n} \left( \sum_{j=1}^{n} A_{i,j} \right) c_i^2 + \frac{1}{2} \sum_{j=1}^{n} \left( \sum_{i=1}^{n} A_{i,j} \right) c_j^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} c_i c_j \]

\[ = \frac{1}{2} \sum_{i=1}^{n} D_{i,i} c_i^2 + \frac{1}{2} \sum_{j=1}^{n} D_{j,j} c_j^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} c_i c_j \]

\[ = c^\top D c - c^\top A c = c^\top L c \]
Hence to find the solution we need to solve for

\[
\text{Minimize } c^\top Lc \quad \text{s.t. } \|c\| = 1
\]
Hence to find the solution we need to solve for

\[ \text{Minimize } c^\top L c \quad \text{s.t. } \| c \| = 1 \]

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.
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 $v_1, \ldots, v_n$ of $L$ (ascending order of eigenvalues)

4. Pick the $K$ eigenvectors with smallest eigenvalues to get $y_1, \ldots, y_n \in \mathbb{R}^K$

5. Use K-means clustering algorithm on $y_1, \ldots, y_n$
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 $v_1, \ldots, v_n$ of $L$ (ascending order of eigenvalues)

4. Pick the $K$ eigenvectors with smallest eigenvalues to get $y_1, \ldots, y_n \in \mathbb{R}^K$

5. Use K-means clustering algorithm on $y_1, \ldots, y_n$

$y_1, \ldots, y_n$ 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 $v_1, \ldots, v_n$ of $L$ (ascending order of eigenvalues)
4. Pick the $K$ eigenvectors with smallest eigenvalues to get $y_1, \ldots, y_n \in \mathbb{R}^K$
5. Use K-means clustering algorithm on $y_1, \ldots, y_n$

$y_1, \ldots, y_n$ are called spectral embedding

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