Machine Learning for Data Science (CS4786)
Lecture 15

Review + Probabilistic Modeling

Course Webpage:
http://www.cs.cornell.edu/Courses/cs4786/2017fa/
Announcements

- In-class Kaggle link is up
  - only one registration per group
  - 5 submissions per day allowed
  - Start early so you get more submissions
- Survey: Participation 95.44% ! Kudos!
Lecture Speed

- Too fast
- Somewhat fast
- Right speed
- Somewhat slow
- Too slow
Lecture Style/Clarity

- Too Mathy
- Vague
- Clear and Easy
- Too easy/boring
Assignment Load

- 1-2 hours
- 2-5 hours
- 5-9 hours
- > 9 hours
Tell me who your friends are...

Cluster nodes in 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 adjacency matrix of a graph.
TELL ME WHO YOUR FRIENDS ARE

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

\[ n \]

\[ A \]
Steps

Spectral Embedding

Cluster(\text{Y})
Spectral Clustering

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

Example:

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

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

$A$ is adjacency matrix of a graph.

$L = D - A$

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

- Nodes linked to each other are close in embedded space
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
Normalized Cut

- Why cut is perhaps not a good measure?
Why cut is perhaps not a good measure?
Why cut is perhaps not a good measure?

Normalized Cut

\[
\text{NCUT} = \sum_{j} \frac{\text{Edges}(C_j)}{\text{Edges}(C) + \text{Edges}(C_j)}
\]

Example

\[
\text{CUT}(C_1, C_2) \leq 1
\]

\[
\text{Minimize } \text{CUT}(C_1, C_2) \text{ s.t. } \text{Edges}(C_1) = \text{Edges}(C_2)
\]
Why cut is perhaps not a good measure?

Normalized Cut

\[ \text{normalized cut: Minimize sum of ratio of number of edges cut per cluster and number of edges within cluster} \]

\[ \text{NCUT} = \sum_j \text{CUT}(C_j) \]

Example

\( K = 2 \)

\[ \text{CUT}(C_1, C_2) \]

\[ \text{Minimize} \text{CUT}(C_1, C_2) \text{s.t.} \]

\[ \text{Edges}(C_1) = \text{Edges}(C_2) \]
Normalized cut: Minimize sum of ratio of number of edges cut per cluster and number of edges within cluster

$$\text{NCUT} = \sum_j \frac{\text{CUT}(C_j)}{\text{Edges}(C_j)}$$

This is an NP hard problem! So relax
As before, we want to minimize \( \sum_{(i,j) \in E} (c_i - c_j)^2 = c^T Lc \)
As before, we want to minimize $\sum_{(i,j) \in E} (c_i - c_j)^2 = c^T L c$

But we also want to weight the values of $c_i$'s based on degree. We want high degree nodes to have larger $c$ magnitude.
As before, we want to minimize $\sum_{(i,j) \in E} (c_i - c_j)^2 = c^\top L c$

But we also want to weight the values of $c_i$'s based on degree. We want high degree nodes to have larger $c$ magnitude.

That is we want to simultaneously maximize $\sum_{i=1}^{n} c_i^2 D_{i,i} = c^\top D c$
As before, we want to minimize \( \sum_{(i,j) \in E} (c_i - c_j)^2 = c^\top Lc \)

But we also want to weight the values of \( c_i \)'s based on degree. We want high degree nodes to have larger \( c \) magnitude

That is we want to simultaneously maximize \( \sum_{i=1}^{n} c_i^2 D_{i,i}^2 = c^\top Dc \)

Find \( c \) so as to:

\[
\begin{align*}
\text{minimize} \quad & \frac{c^\top Lc}{c^\top Dc} \\
\equiv \text{minimize} \quad & c^\top Lc \text{ subject to } c^\top Dc = 1
\end{align*}
\]
As before, we want to minimize \( \sum_{(i,j) \in E} (c_i - c_j)^2 = c^\top Lc \)

But we also want to weight the values of \( c_i \)'s based on degree. We want high degree nodes to have larger \( c \) magnitude.

That is we want to simultaneously maximize \( \sum_{i=1}^{n} c_i^2 D_{i,i} = c^\top Dc \)

Find \( c \) so as to:

\[
\begin{align*}
\text{minimize} & \quad \frac{c^\top Lc}{c^\top Dc} \\
\equiv & \quad \text{minimize} \quad c^\top Lc \text{ subject to } c^\top Dc = 1 \\
\equiv & \quad \text{minimize} \quad u^\top D^{-1/2}LD^{-1/2}u \text{ subject to } \|u\| = 1
\end{align*}
\]
Spectral Clustering

Minimize $c^\top \tilde{L} c$ s.t. $c \perp 1$
Spectral Clustering

Minimize $c^\top \tilde{L} c$ s.t. $c \perp 1$

Approximately Minimize normalized cut!
Minimize $c^\top \tilde{L} c$ s.t. $c \perp \mathbf{1}$

Approximately Minimize normalized cut!

Solution: Find second smallest eigenvectors of $\tilde{L} = I - D^{-1/2} A D^{-1/2}$
Spectral Clustering Algorithm (Normalized)

1. Given matrix $A$ calculate diagonal matrix $D$ s.t. $D_{i,i} = \sum_{j=1}^{n} A_{i,j}$

2. Calculate the normalized Laplacian matrix $\tilde{L} = I - D^{-1/2} A D^{-1/2}$

3. Find eigen vectors $v_1, \ldots, v_n$ of $\tilde{L}$ (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 $y_1, \ldots, y_n$
Review
K-means

• K-means algorithm: (wishful thinking)
  • Fix parameters (the k means) and compute new cluster assignments (or probabilities) for every point
  • Fix cluster assignment for all data points and re-evaluate parameters (the k-means)
Single-Link Clustering

- Start with all points being their own clusters
- Until we get K-clusters, merge the closest two clusters
When to Use Single Link

- When we have dense sampling of points within each cluster
- When not to use: when we might have outliers
When to use K-means

- When we have nice spherical round equal size clusters or cluster masses are far apart
- Handles outliers better
Principal Component Analysis

1. \[ \Sigma = \text{cov}(X) \]

2. \[ W = \text{eigs}(\Sigma, K) \]

3. \[ Y = (X - \mu) \times W \]
When to use PCA

• Great when data is truly low dimensional (on a hyperplane (linear))

• Or approximately low dimensional (almost lie on plane Eg. very flat ellipsoid)

• Eg. Dimensionality reduction for face images, for multiple biometric applications as preprocessing…
CCA Algorithm

1. \[ X = \begin{pmatrix} n & X_1 \\ d_1 & X_2 \end{pmatrix} \]

2. \[ \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \text{cov}(X) \]

3. \[ W_1 = \text{eigs}\left( \begin{pmatrix} \Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} \end{pmatrix}, K \right) \]

4. \[ Y_1 = X_1 - \mu_1 \times W_1 \]
When to use CCA?

- CCA applies for problems where data can be split into 2 views $X = [X_1, X_2]$
- CCA picks directions of projection (in each view) where data is maximally correlated
- Maximizes correlation coefficient and not just covariance so is **scale free**
When to use CCA

• Scenario 1: You have two feature extraction techniques.
  • One provides excellent features for dogs Vs cats and noise on other classes
  • Other method provides excellent features for cars Vs bikes and noise for other classes

• What do we do?
  A. Use CCA to find one common representation
  B. Concatenate the two features extracted
When to use CCA

- Scenario 2: You have two cameras capturing images of the same objects from different angles.

- You have a feature extraction technique that provides feature vectors from each camera.

- You want to extract good features for recognizing the object from the two cameras

- What do we do?
  
  A. Use CCA to find one common representation
  
  B. Concatenate features provides excellent features for
$\mathbf{Y} = \mathbf{X} \times \begin{bmatrix}
  +1 & \ldots & -1 \\
  -1 & \ldots & +1 \\
  +1 & \ldots & -1 \\
  \vdots & \ddots & \vdots \\
  +1 & \ldots & -1 \\
  K & & -1
\end{bmatrix} \frac{d}{\sqrt{K}}$
When to use RP?

• When data is huge and very large dimensional

• For PCA, CCA typically you think of K (no. of dimensions we reduce to) in double digits

• For RP think of K typically in 3-4 digit numbers

• RP guarantees preservation of inter-point distances.

• RP unlike PCA and CCA does not project using unit vectors. (What does this mean?)
1. $\tilde{K} \\
\quad n = \text{compute}(k, X)$

2. $\begin{bmatrix} A, \lambda \end{bmatrix} = \text{eigs}(\tilde{K}, K)$

3. $P = \frac{A_1}{\sqrt{\lambda_1}} \cdot \frac{A_K}{\sqrt{\lambda_K}}$
4. \[ Y = \tilde{K} \times P \]
When to use Kernel PCA

- When data lies on some non-linear, low dimensional subspace

- Kernel function matters. (Eg. RBF kernel, only points close to a given point have non-negligible kernel evaluation)
Spectral Clustering

• You want to cluster nodes of a graph into groups based on connectivity

• Unnormalized spectral clustering: divide into groups where as few edges between groups are cut
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 Embedding

Use K-means on Y
Normalized Spectral Clustering

• Unnormalized spectral embedding encourages loner nodes to be pushed far away from rest

• This is indeed the min-cut solution to cut off loners

• Instead form clusters that minimize ratio of edges cut to number of edges each cluster has
  • (busy groups tend to form clusters)

• Algorithm, replace Laplacian matrix by normalized one
**Spectral Clustering Algorithm (Normalized)**

1. Given matrix $A$ calculate diagonal matrix $D$ s.t. $D_{i,i} = \sum_{j=1}^{n} A_{i,j}$

2. Calculate the normalized Laplacian matrix $\tilde{L} = I - D^{-1/2} A D^{-1/2}$

3. Find eigen vectors $v_1, \ldots, v_n$ of $\tilde{L}$ (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 $y_1, \ldots, y_n$
When to use Spectral Clustering

• First, even works with weighted graph, where weight of edge represents similarity

• When knowledge about how clusters should be formed is solely decided by similarity between points, there is no underlying prior knowledge
Probabilistic Modeling
Data: $x_1, \ldots, x_n$
Data: $x_1, \ldots, x_n$
Probabilistic Model
Probabilistic Model

\[
\pi_1 = 0.5 \\
\Sigma_1 \\
\pi_2 = 0.25 \\
\Sigma_2 \\
\pi_3 = 0.25 \\
\Sigma_3
\]
Probabilistic Model

\[
\begin{align*}
\pi_1 &= 0.5 \\
\pi_2 &= 0.25 \\
\pi_3 &= 0.25 \\
\end{align*}
\]
Set of models $\Theta$ consists of parameters s.t. $P_\theta$ for each $\theta \in \Theta$ is a distribution over data.

Learning: Estimate $\theta^* \in \Theta$ that best models given data.
Pick $\theta \in \Theta$ that maximizes probability of observation
Pick $\theta \in \Theta$ that maximizes probability of observation

Reasoning:
- One of the models in $\Theta$ is the correct one
Pick $\theta \in \Theta$ that maximizes probability of observation

Reasoning:
- One of the models in $\Theta$ is the correct one
- Given data we pick the one that best explains the observed data
Pick $\theta \in \Theta$ that maximizes probability of observation

Reasoning:
- One of the models in $\Theta$ is the correct one
- Given data we pick the one that best explains the observed data
- Equivalently pick the maximum likelihood estimator,

$$
\theta_{MLE} = \arg \max_{\theta \in \Theta} \log P_\theta(x_1, \ldots, x_n)
$$
Pick $\theta \in \Theta$ that maximizes probability of observation

**Reasoning:**
- One of the models in $\Theta$ is the correct one
- Given data we pick the one that best explains the observed data
- Equivalently pick the maximum likelihood estimator,
  \[
  \theta_{MLE} = \arg\max_{\theta \in \Theta} \log P_\theta(x_1, \ldots, x_n)
  \]

Often referred to as frequentist view
Maximum Likelihood Principal

Pick \( \theta \in \Theta \) that maximizes probability of observation

\[
\theta_{MLE} = \arg\max_{\theta \in \Theta} \log P_{\theta}(x_1, \ldots, x_n)
\]

A priori all models are equally good, data could have been generated by any one of them
Say you had a prior belief about models provided by $P(\theta)$.

Pick $\theta \in \Theta$ that is most likely given data.
Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:
- Models are abstractions that capture our belief
Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:
- Models are abstractions that capture our belief
- We update our belief based on observed data
Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:

- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most
Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:
- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most
- Pick $\theta$ that maximizes $\log P(\theta|x_1, \ldots, x_n)$
Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:
- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most
- Pick $\theta$ that maximizes $\log P(\theta|x_1, \ldots, x_n)$

I want to say: Often referred to as Bayesian view
Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:
- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most
- Pick $\theta$ that maximizes $\log P(\theta|x_1, \ldots, x_n)$

I want to say: Often referred to as Bayesian view

There are Bayesian and there Bayesians
Maximum A Posteriori

Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\theta_{MAP} = \arg\max_{\theta \in \Theta} P(\theta|x_1, \ldots, x_n)$$
Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\theta_{MAP} = \underset{\theta \in \Theta}{\operatorname{argmax}} P(\theta | x_1, \ldots, x_n)$$

$$= \underset{\theta \in \Theta}{\operatorname{argmax}} \frac{P(x_1, \ldots, x_n | \theta)P(\theta)}{P(x_1, \ldots, x_n)}$$
Maximum A Posteriori

Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\theta_{MAP} = \arg\max_{\theta \in \Theta} P(\theta|x_1, \ldots, x_n)$$

$$= \arg\max_{\theta \in \Theta} \frac{P(x_1, \ldots, x_n|\theta)P(\theta)}{P(x_1, \ldots, x_n)}$$

$$= \arg\max_{\theta \in \Theta} \left[ \frac{P(x_1, \ldots, x_n|\theta)}{P(x_1, \ldots, x_n)} \right] \frac{P(\theta)}{P(\theta)}$$

$$= \arg\max_{\theta \in \Theta} \left[ \frac{P(x_1, \ldots, x_n|\theta)}{P(\theta)} \right]$$

likelihood prior
Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\theta_{MAP} = \arg\max_{\theta \in \Theta} P(\theta | x_1, \ldots, x_n)$$

$$= \arg\max_{\theta \in \Theta} \frac{P(x_1, \ldots, x_n | \theta)P(\theta)}{P(x_1, \ldots, x_n)}$$

$$= \arg\max_{\theta \in \Theta} \underbrace{P(x_1, \ldots, x_n | \theta)}_{\text{likelihood}} \underbrace{P(\theta)}_{\text{prior}}$$

$$= \arg\max_{\theta \in \Theta} \log P(x_1, \ldots, x_n | \theta) + \log P(\theta)$$
The Bayesian Choice

Don’t pick any $\theta^* \in \Theta$

- Model is simply an abstraction
- We have a posteriori distribution over models, why pick one $\theta$?

\[
P(X|\text{data}) = \sum_{\theta \in \Theta} P(X, \theta|\text{data}) = \sum_{\theta \in \Theta} P(X|\theta)P(\theta|\text{data})
\]