Advanced Introduction to Machine Learning

10715, Fall 2014

Spectral Clustering

Eric Xing
Lecture 20, November 17, 2014

Reading:
© Eric Xing @ CMU, 2014
Data Clustering
Data Clustering

- Two different criteria
  - Compactness, e.g., k-means, mixture models
  - Connectivity, e.g., spectral clustering

Compactness

Connectivity
Spectral Clustering

Data

Similarities

© Eric Xing @ CMU, 2014
Some graph terminology
- Objects (e.g., pixels, data points): \( i \in I = \text{vertices of graph } G \)
- Edges \((ij)\) = pixel pairs with \( W_{ij} > 0 \)
- Similarity matrix \( W = [ W_{ij} ] \)

Degree:
- \( d_i = \sum_{j \in G} S_{ij} \)
- \( d_A = \sum_{i \in A} d_i \) degree of \( A \subseteq G \)

Assoc\((A, B)\) = \( \sum_{i \in A} \sum_{j \in B} W_{ij} \)
Cuts in a Graph

- (edge) cut = set of edges whose removal makes a graph disconnected

- weight of a cut:
  \[ \text{cut}(A, B) = \sum_{i \in A} \sum_{j \in B} W_{ij} = \text{Assoc}(A, B) \]

- Normalized Cut criteria: minimum cut(A, \(\overline{A}\))

  \[ \text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{d_A} + \frac{\text{cut}(A, B)}{d_B} \]

More generally:

\[ \text{Ncut}(A_1, A_2 \ldots A_k) = \sum_{r=1}^{k} \left( \frac{\sum_{i \in A_r, j \in V \setminus A_r} W_{ij}}{\sum_{i \in A_r, j \in V} W_{ij}} \right) = \sum_{r=1}^{k} \left( \frac{\text{cut}(A_r, \overline{A_r})}{d_{A_r}} \right) \]
Graph-based Clustering

- Data Grouping
- Image segmentation

Let $G = \{V, E\}$ be a graph, where $V$ is the set of vertices and $E$ is the set of edges. The weight of an edge $W_{ij} = f(d(x_i, x_j))$.

- Affinity matrix: $W = [w_{i,j}]$
- Degree matrix: $D = \text{diag}(d_i)$
- Laplacian matrix: $L = D - W$
- (bipartite) partition vector:

$$x = [x_1, \ldots, x_N] = [1, 1, \ldots, 1, -1, -1, \ldots, -1]$$
Affinity Function

\[ W_{i,j} = e^{-\frac{\|X_i - X_j\|^2_2}{\sigma^2}} \]

- Affinities grow as \( \sigma \) grows →
- How the choice of \( \sigma \) value affects the results?
- What would be the optimal choice for \( \sigma \)?

© Eric Xing @ CMU, 2014
Clustering via Optimizing Normalized Cut

- The normalized cut:

\[
Ncut(A, B) = \frac{\text{cut}(A, B)}{d_A} + \frac{\text{cut}(A, B)}{d_B}
\]

- Computing an optimal normalized cut over all possible \( y \) (i.e., partition) is NP hard

- Transform Ncut equation to a matrix form (Shi & Malik 2000):

\[
\begin{align*}
\min_x Ncut(x) &= \min_y y^T (D - W) y \\
\text{Subject to:} & \quad y \in \{1, -1\}^n \\
y^T D1 &= 0
\end{align*}
\]

- Still an NP hard problem

\[
Ncut(A, B) = \frac{\text{cut}(A, B)}{\text{deg}(A)} + \frac{\text{cut}(A, B)}{\text{deg}(B)}
\]

\[
= \frac{(1 + x)^T (D - S)(1 + x)}{k1^T D1} + \frac{(1 - x)^T (D - S)(1 - x)}{(1 - k)1^T D1}; \quad k = \frac{\sum_{i=1}^{n} D(i, i)}{\sum_{i=1}^{n} D(i, i)}
\]
Relaxation

\[
\min_x \text{Ncut}(x) = \min_y \frac{y^T (D - W) y}{y^T Dy}
\]

Subject to: \( y \in \{1, -b\}^n \)
\[
y^T D 1 = 0
\]

• Instead, relax into the continuous domain by solving generalized eigenvalue system:
\[
\min_y y^T (D - W) y, \quad \text{s.t.} \quad y^T Dy = 1
\]

• Which gives:
\[
(D - W) y = \lambda Dy
\]

• Note that \((D - W) 1 = 0\) so, the first eigenvector is \(y_0 = 1\) with eigenvalue 0.

• The second smallest eigenvector is the real valued solution to this problem!!
Algorithm

1. Define a similarity function between 2 nodes. i.e.:
   \[ w_{i,j} = e^{-\frac{\|x(i) - x(j)\|^2}{\sigma^2}} \]
2. Compute affinity matrix (W) and degree matrix (D).
3. Solve
   \[ (D - W)y = \lambda Dy \]
   • Do singular value decomposition (SVD) of the graph Laplacian \( L = D - W \)
     \[ L = V^T \Lambda V \Rightarrow y^* \]
4. Use the eigenvector with the second smallest eigenvalue, \( y^* \), to bipartition the graph.
   • For each threshold \( k \),
     \[ A_k = \{ i \mid y_i \text{ among } k \text{ largest element of } y^* \} \]
     \[ B_k = \{ i \mid y_i \text{ among } n-k \text{ smallest element of } y^* \} \]
   • Compute \( \text{Ncut}(A_k, B_k) \)
   • Output \( k^* = \arg \max Ncut(A_k, B_k) \) and \( A_{k^*}, B_{k^*} \)
Ideally …

\[ Ncut(A, B) = \frac{y^T (D - S)y}{y^T Dy}, \text{ with } y_i \in \{1, -b\}, y^T D1 = 0. \]

\[ (D - S)y = \lambda Dy \]
Example

input affinity matrix

affinity matrix reordered according to solution vector

the partition according to the solution vector
Poor features can lead to poor outcome (xing et al 2002)
Cluster vs. block matrix

\[ \text{Ncut}(A,B) = \frac{\text{cut}(A,B)}{d_A} + \frac{\text{cut}(A,B)}{d_B} \]

\[ \text{Degree}(A) = \sum_{i \in A, j \in V} W_{i,j} \]
Compare to Minimum cut

- Criterion for partition:

\[
\min \text{cut}(A, B) = \min_{A,B} \sum_{i \in A, j \in B} W_{i,j}
\]

Problem!
Weight of cut is directly proportional to the number of edges in the cut.

First proposed by Wu and Leahy

Cuts with lesser weight than the ideal cut
Superior performance?

- K-means and Gaussian mixture methods are biased toward convex clusters
Ncut is superior in certain cases
Why?
General Spectral Clustering

Data

Similarities

© Eric Xing @ CMU, 2014
Representation

- Partition matrix $X$:
  \[ X = [X_1, \ldots, X_K] \]

- Pair-wise similarity matrix $W$: $W(i, j) = \text{aff}(i, j)$

- Degree matrix $D$:
  \[ D(i, i) = \sum_j w_{i,j} \]

- Laplacian matrix $L$:
  \[ L = D - W \]

© Eric Xing @ CMU, 2014
**Eigenvectors and blocks**

- **Block matrices have block eigenvectors:**

  \[
  \begin{pmatrix}
  1 & 1 & 0 & 0 \\
  1 & 1 & 0 & 0 \\
  0 & 0 & 1 & 1 \\
  0 & 0 & 1 & 1 \\
  \end{pmatrix}
  \]

  \[
  \lambda_1 = 2, \quad \lambda_2 = 2, \quad \lambda_3 = 0, \quad \lambda_4 = 0
  \]

  eigensolver

- **Near-block matrices have near-block eigenvectors:**

  \[
  \begin{pmatrix}
  1 & 1 & .2 & 0 \\
  1 & 1 & 0 & -.2 \\
  .2 & 0 & 1 & 1 \\
  0 & -.2 & 1 & 1 \\
  \end{pmatrix}
  \]

  \[
  \lambda_1 = 2.02, \quad \lambda_2 = 2.02, \quad \lambda_3 = -0.02, \quad \lambda_4 = -0.02
  \]

  eigensolver
Spectral Space

- Can put items into blocks by eigenvectors:

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>.2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-.2</td>
</tr>
<tr>
<td>.2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>-.2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Clusters clear regardless of row ordering:

<table>
<thead>
<tr>
<th>1</th>
<th>.2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-.2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-.2</td>
<td>1</td>
</tr>
</tbody>
</table>
Spectral Clustering

- Algorithms that cluster points using eigenvectors of matrices derived from the data
- Obtain data representation in the low-dimensional space that can be easily clustered
- Variety of methods that use the eigenvectors differently (we have seen an example)
- Empirically very successful
- Authors disagree:
  - Which eigenvectors to use
  - How to derive clusters from these eigenvectors
- Two general methods
Method #1

- Partition using only one eigenvector at a time
- Use procedure recursively
- Example: Image Segmentation
  - Uses 2nd (smallest) eigenvector to define optimal cut
  - Recursively generates two clusters with each cut
Method #2

- Use k eigenvectors (k chosen by user)
- Directly compute k-way partitioning
- Experimentally has been seen to be “better”
Given a set of points $S=\{s_1, \ldots, s_n\}$

- Form the affinity matrix $w_{i,j} = e^{-\frac{||s_i - s_j||^2}{2\sigma^2}}$, $\forall i \neq j$, $w_{i,i} = 0$
- Define diagonal matrix $D_{ii} = \Sigma_{k} a_{ik}$
- Form the matrix $L = D^{-1/2}WD^{-1/2}$
- Stack the $k$ largest eigenvectors of $L$ to form the columns of the new matrix $X$:

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_k \end{bmatrix}$$

- Renormalize each of $X$’s rows to have unit length and get new matrix $Y$. Cluster rows of $Y$ as points in $R^k$
SC vs Kmeans
Why it works?

- K-means in the spectrum space!
More formally ...

- Recall generalized Ncut

\[
\text{Ncut}(A_1, A_2 \ldots A_k) = \sum_{r=1}^{k} \left( \frac{\sum_{i \in A_r, j \in V \setminus A_r} W_{ij}}{\sum_{i \in A_r, j \in V} W_{ij}} \right) = \sum_{r=1}^{k} \left( \frac{\text{cut}(A_r, \overline{A_r})}{d_{A_r}} \right)
\]

- Minimizing this is equivalent to spectral clustering

\[
\min \text{ Ncut}(A_1, A_2 \ldots A_k) = \sum_{r=1}^{k} \left( \frac{\text{cut}(A_r, \overline{A_r})}{d_{A_r}} \right)
\]

\[
\min Y^T D^{-1/2} W D^{-1/2} Y
\]

s.t. \( Y^T Y = I \)
Toy examples

Images from Matthew Brand (TR-2002-42)
User’s Prerogative

- Choice of $k$, the number of clusters

- Choice of scaling factor
  - Realistically, search over $\sigma^2$ and pick value that gives the tightest clusters

- Choice of clustering method: k-way or recursive bipartite

- Kernel affinity matrix

$$w_{i,j} = K(S_i, S_j)$$
Conclusions

● Good news:
  ● Simple and powerful methods to segment images.
  ● Flexible and easy to apply to other clustering problems.

● Bad news:
  ● High memory requirements (use sparse matrices).
  ● Very dependant on the scale factor for a specific problem.

\[
W(i, j) = e^{-\frac{-\|X(i) - X(j)\|^2_2}{\sigma_X^2}}
\]