Spectral Clustering

Aarti Singh

Machine Learning 10-701/15-781
Apr 7, 2010

Slides Courtesy: Eric Xing, M. Hein & U.V. Luxburg
Data Clustering

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

**Goal:** Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

**Similarity Graph:** $G(V,E)$
- $V$ – Vertices (Data points, pixels)
- $E$ – Edge if similarity $> 0$, Edge weights = similarities

Partition the graph so that edges within a group have large weights and edges across groups have small weights.
Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

\[ G(V,E) \]
\[ V - \text{Vertices (Data points, pixels)} \]

(1) \( E \) – Edge if similarity > 0, Edge weights = similarities \( w(x_i,x_j) \)

E.g. Gaussian kernel similarity function

\[ W_{ij} = e^{-\frac{||x_i-x_j||^2}{2\sigma^2}} \]

Controls size of neighborhood

Data clustering

\[ G = \{V,E\} \]
Similarity Graph construction

Similarity Graphs: Model local neighborhood relations between data points

\[ G(V,E) \quad V - \text{Vertices (Data points, pixels)} \]

(2) \( E - \text{Edge if } \varepsilon\text{-NN } ||x_i - x_j|| \leq \varepsilon \), \quad \text{Edge weights } = 1 \quad (\varepsilon\text{-NN } \sim \text{equi-distant})
Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

G(V,E)  V – Vertices (Data points, pixels)

(2) E – Edge if $\varepsilon$-NN $||xi - xj|| \leq \varepsilon$, Edge weights = 1  ($\varepsilon$-NN ~ equi-distant)

(3) E – Edge if $k$-NN, Edge weights = similarities $w(x_i, x_j)$

yields directed graph

connect A with B if  A $\rightarrow$ B  OR  A $\leftarrow$ B  (symmetric kNN graph)
connect A with B if  A $\rightarrow$ B  AND  A $\leftarrow$ B  (mutual kNN graph)
Some Graph Notation

- $W = (w_{ij})$ adjacency matrix of the graph
- $d_i = \sum_j w_{ij}$ degree of a vertex
- $D = \text{diag}(d_1, \ldots, d_n)$ degree matrix
- $|A| = \text{number of vertices in } A$
- $\text{vol}(A) = \sum_{i \in A} d_i$
Partitioning a graph into two clusters

**Min-cut:** Partition graph into two sets $A$ and $B$ such that weight of edges connecting vertices in $A$ to vertices in $B$ is minimum.

$$\text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij}$$

- Easy to solve $O(VE)$ algorithm
- Not satisfactory partition – often isolates vertices
Partitioning a graph into two clusters

Partition graph into two sets $A$ and $B$ such that weight of edges connecting vertices in $A$ to vertices in $B$ is minimum & size of $A$ and $B$ are very similar.

$$\text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij}$$

Balanced Min-cut:

$$\min_{A, B} \text{cut}(A, B) \text{ s.t. } |A| = |B|$$

Ratio cut:

$$\text{RatioCut}(A, B) := \text{cut}(A, B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$$

Normalized cut:

$$\text{Ncut}(A, B) := \text{cut}(A, B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$

But NP-hard to solve!!

Spectral clustering is a relaxation of these.
Graph cut

\[ \text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij} \]

Choose \( f = (f_1, ..., f_n)' \) with \( f_i = \begin{cases} 
1 & \text{if } X_i \in A \\
-1 & \text{if } X_i \in B
\end{cases} \)

\[ \text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij} = \frac{1}{4} \sum_{i,j} w_{ij} (f_i - f_j)^2 = f^T (D-W) f \]

\[ \text{RHS} = f^T (D-W) f = f^T D f - f^T W f = \sum_i d_i f_i^2 - \sum_{i,j} f_i f_j w_{ij} \]

\[ = \frac{1}{2} \left( \sum_i (\sum_j w_{ij} f_i^2) - 2 \sum_{i,j} f_i f_j w_{ij} + \sum_j (\sum_i w_{ij} f_j^2) \right) \]

\[ = \frac{1}{2} \sum_{ij} w_{ij} (f_i - f_j)^2 = \text{LHS} \]
Graph cut and Graph Laplacian

\[
\text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij} = f^T (D-W) f = f^T L f
\]

\[
L = D - W
\]

Unnormalized Graph Laplacian

Spectral properties of \( L \):

- Smallest eigenvalue of \( L \) is 0, corresponding eigenvector is \( \mathbf{1} \)
- Thus eigenvalues \( 0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \).

\[
L \mathbf{1} = D \mathbf{1} - W \mathbf{1} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} - \begin{bmatrix} \sum_j w_{1j} \\ \sum_j w_{2j} \\ \vdots \\ \sum_j w_{nj} \end{bmatrix} = 0
\]
Balanced min-cut

\[ \min_{A,B} \text{cut}(A, B) \text{ s.t. } |A| = |B| \]

\[ \min \quad f^T L f \quad \text{s.t.} \quad f^T 1 = 0 \]
\[ f \in \{-1, 1\}^n \]

(since \( \sum f_i = \sum 1_{i \in A} - 1_{i \in B} = 0 \))

Above formulation is still NP-Hard, so we relax \( f \) not to be binary:

\[ \min_{f \in \mathbb{R}^n} \quad f^T L f \quad \text{s.t.} \quad f^T 1 = 0, \quad f^T f = n \]

\[ \min_{f \in \mathbb{R}^n} \quad \frac{f^T L f}{f^T f} \quad \text{s.t.} \quad f^T 1 = 0 \]
Relaxation of Balanced min-cut

\[
\min_{f \in \mathbb{R}^n} \frac{f^T L f}{f^T f} \quad \text{s.t.} \quad f^T 1 = 0
\]

\[\lambda_{\min}(L) \quad \text{smallest eigenvalue of } L \quad \text{(Rayleigh-Ritz theorem)}\]

If \( f \) is eigenvector of \( L \), then

\[
\frac{f^T L f}{f^T f} = \frac{f^T \lambda f}{f^T f} = \lambda
\]

Recall that smallest eigenvalue of \( L \) is 0 with corresponding eigenvector \( 1 \)
But \( f \) can’t be \( 1 \) according to constraint \( f^T 1 = 0 \)

Therefore, solution \( f \) is the eigenvector of \( L \) corresponding to second smallest eigenvalue, aka second eigenvector.
Approximation of Balanced min-cut

$$\min_{A, B} \text{cut}(A, B) \text{ s.t. } |A| = |B|$$

Let $f$ be the second eigenvector of the unnormalized graph Laplacian $L$.

Recover binary partition as follows:

$$i \in A \quad \text{if} \quad f_i \geq 0$$

$$i \in B \quad \text{if} \quad f_i < 0$$

Similar relaxations work for other cut problems:

RatioCut - second eigenvector of unnormalized graph Laplacian $L = D - W$

Normalized cut – second eigenvector of normalized Laplacian $L' = I - D^{-1}W$
Example

Xing et al 2001

input affinity matrix

affinity matrix reordered according to solution vector

the partition according to the solution vector
How to partition a graph into k clusters?
Spectral Clustering Algorithm

Input: Similarity matrix \( W \), number \( k \) of clusters to construct

- Build similarity graph
- Compute the first \( k \) eigenvectors \( v_1, \ldots, v_k \) of the matrix

\[
\begin{cases} 
  L & \text{for unnormalized spectral clustering} \\
  L' & \text{for normalized spectral clustering} 
\end{cases}
\]

- Build the matrix \( V \in \mathbb{R}^{n \times k} \) with the eigenvectors as columns
- Interpret the rows of \( V \) as new data points \( Z_i \in \mathbb{R}^k \)

\[
\begin{array}{c|ccc}
  & v_1 & v_2 & v_3 \\
\hline
  Z_1 & v_{11} & v_{12} & v_{13} \\
  \vdots & \vdots & \vdots & \vdots \\
  Z_n & v_{n1} & v_{n2} & v_{n3} \\
\end{array}
\]

- Cluster the points \( Z_i \) with the \( k \)-means algorithm in \( \mathbb{R}^k \).

Dimensionality Reduction
\( n \times n \rightarrow n \times k \)
Eigenvectors of Graph Laplacian

- 1\textsuperscript{st} Eigenvector is the all ones vector $1$
- 2\textsuperscript{nd} Eigenvector thresholded at 0 separates first two clusters from last two
- k-means clustering of the 4 eigenvectors identifies all clusters
Why does it work?

Data are projected into a lower-dimensional space (the spectral/eigenvector domain) where they are easily separable, say using k-means.
Why does it work?

- Block matrices have block eigenvectors:
  $\lambda_1 = 2$
  $\lambda_2 = 2$
  $\lambda_3 = 0$
  $\lambda_4 = 0$

- Near-block matrices have near-block eigenvectors:
  $\lambda_1 = 2.02$
  $\lambda_2 = 2.02$
  $\lambda_3 = -0.02$
  $\lambda_4 = -0.02$
Why does it work?

- Can put items into blocks by eigenvectors:

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

- Clusters clear regardless of row ordering:

\[
\begin{array}{cccc}
1 & .2 & 1 & 0 \\
.2 & 1 & 0 & 1 \\
1 & 0 & 1 & -.2 \\
0 & 1 & -.2 & 1 \\
\end{array}
\]
Applying k-means to laplacian eigenvectors allows us to find clusters with non-convex boundaries.

Both perform the same, but spectral clustering is superior.
Applying k-means to laplacian eigenvectors allows us to find clusters with non-convex boundaries.
k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to find clusters with non-convex boundaries.
Examples (Choice of k)

Ng et al 2001
Some Issues

- Choice of number of clusters $k$
  Most stable clustering is usually given by the value of $k$ that maximizes the eigengap (difference between consecutive eigenvalues)

$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$
Some Issues

- Choice of number of clusters k
- Choice of similarity
  - choice of kernel
  - for Gaussian kernels, choice of $\sigma$

Good similarity measure

Poor similarity measure
Some Issues

- Choice of number of clusters $k$

- Choice of similarity
  - choice of kernel
    - for Gaussian kernels, choice of $\sigma$

- Choice of clustering method – k-way vs. recursive bipartite
Spectral clustering summary

- Algorithms that cluster points using eigenvectors of matrices derived from the data
- Useful in hard non-convex clustering problems
- Obtain data representation in the low-dimensional space that can be easily clustered
- Variety of methods that use eigenvectors of unnormalized or normalized Laplacian, different, how to derive clusters from eigenvectors, k-way vs repeated 2-way
- Empirically very successful
## Comparison Chart

<table>
<thead>
<tr>
<th></th>
<th>Decision Trees</th>
<th>K-NN</th>
<th>Naïve Bayes</th>
<th>Logistic regression</th>
<th>SVM</th>
<th>Boosting</th>
<th>Neural Nwks</th>
<th>HMM</th>
<th>Bayes Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen/Disc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loss functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision boundary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assumptions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convergence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Loss functions

\[ y_i = 1 \]

square loss

log loss

hinge loss \((1 - yf(x))_+\)

0/1 loss

\[ f(x_i) \]