Spectral Clustering

Shannon Quinn

(with thanks to William Cohen of Carnegie Mellon University, and J. Leskovec, A. Rajaraman, and J. Ullman of Stanford University)
Graph Partitioning

• Undirected graph

• **Bi-partitioning task:**
  – Divide vertices into two disjoint groups

• **Questions:**
  – How can we define a “good” partition of \( G \)?
  – How can we efficiently identify such a partition?
Graph Partitioning

• What makes a good partition?
  – Maximize the number of within-group connections
  – Minimize the number of between-group connections
Graph Cuts

• Express partitioning objectives as a function of the “edge cut” of the partition

• Cut: Set of edges with only one vertex in a group:

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

\[
cut(A, B) = 2
\]
Graph Cut Criterion

• **Criterion: Minimum-cut**
  - Minimize weight of connections between groups

\[
\arg \min_{A,B} \text{cut}(A,B)
\]

• **Degenerate case:**

• **Problem:**
  – Only considers external cluster connections
  – Does not consider internal cluster connectivity
Graph Cut Criteria

• Criterion: **Normalized-cut** [Shi-Malik, ’97]
  
  – Connectivity between groups relative to the density of each group
  
  \[
  n\text{cut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)}
  \]

  : total weight of the edges with at least one endpoint in :

  ■ Why use this criterion?
  
  ■ Produces more balanced partitions

• How do we efficiently find a good partition?
  
  – **Problem:** Computing optimal cut is NP-hard
Spectral Graph Partitioning

• $A$: adjacency matrix of undirected $G$
  – $A_{ij} = 1$ if is an edge, else 0
• $x$ is a vector in $\mathbb{R}^n$ with components
  – Think of it as a label/value of each node of
• What is the meaning of $A \cdot x$?

$$
\begin{bmatrix}
  a_{11} & \ldots & a_{1n} \\
  \vdots & & \vdots \\
  a_{n1} & \ldots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix}
= 
\begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n
\end{bmatrix}
$$

• Entry $y_i$ is a sum of labels $x_j$ of neighbors of $i$

$y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j$
What is the meaning of $Ax$?

- **$j^{th}$ coordinate of $A \cdot x$**:
  - Sum of the $x$-values of neighbors of $j$.
  - Make this a new value at node $j$.

- **Spectral Graph Theory**:
  - Analyze the “spectrum” of matrix representing.
  - **Spectrum**: Eigenvectors of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues:

\[
\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}
\]

\[
\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n
\]

Matrix Representations

- **Adjacency matrix** \((A)\):
  - \(n \times n\) matrix
  - \(A = [a_{ij}], a_{ij} = 1\) if edge between node \(i\) and \(j\)

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
2 & 1 & 0 & 1 & 0 & 0 & 0 \\
3 & 1 & 1 & 0 & 1 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 & 1 & 1 \\
5 & 1 & 0 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 1 & 1 & 0 \\
\end{array}
\]

- **Important properties:**
  - Symmetric matrix
  - Eigenvectors are real and orthogonal
Matrix Representations

• **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D = [d_{ii}]$, $d_{ii}$ = degree of node $i$
Matrix Representations

- Laplacian matrix (L):
  - $n \times n$ symmetric matrix

- What is trivial eigenpair?

- Important properties:
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real and orthogonal

\[
L = D - A
\]
Spectral Clustering: Graph = Matrix
\[ W v_1 = v_2 \] “propogates weights from neighbors”

[Shi & Meila, 2002]
Spectral Clustering: Graph = Matrix

\( W^* v_1 = v_2 \) “propagates weights from neighbors”

\[ W \cdot v = \lambda v : v \text{ is an eigenvector with eigenvalue } \lambda \]

If \( W \) is connected but roughly block diagonal with \( k \) blocks then
  - the top eigenvector is a constant vector
  - the next \( k \) eigenvectors are roughly piecewise constant with “pieces” corresponding to blocks
Spectral Clustering: Graph = Matrix

$Wv_1 = v_2$ “propagates weights from neighbors”

$W \cdot v = \lambda v : v$ is an eigenvector with eigenvalue $\lambda$

If $W$ is connected but roughly block diagonal with $k$ blocks then
- the “top” eigenvector is a constant vector
- the next $k$ eigenvectors are roughly piecewise constant with “pieces” corresponding to blocks

Spectral clustering:
- Find the top $k+1$ eigenvectors $v_1, \ldots, v_{k+1}$
- Discard the “top” one
- Replace every node $a$ with $k$-dimensional vector $x_a = \langle v_2(a), \ldots, v_{k+1}(a) \rangle$
- Cluster with $k$-means
Spectral Clustering: Graph = Matrix

\[ W^*v_1 = v_2 \text{ “propogates weights from neighbors”} \]

\[ W \cdot v = \lambda v : \text{v is an eigenvector with eigenvalue } \lambda \]

- smallest eigenvects of D-A are largest eigenvects of A
- smallest eigenvects of I-W are largest eigenvects of W

Suppose each \( y(i) = +1 \) or \(-1\):
- Then \( y \) is a cluster indicator that splits the nodes into two
- what is \( y^T(D-A)y \) ?
\[ y^T (D - A)y = y^T Dy - y^T Ay = \sum_i d_i y_i^2 - \sum_{i,j} a_{i,j} y_i y_j \]

\[
= \frac{1}{2} \left[ 2 \sum_i d_i y_i^2 - 2 \sum_{i,j} a_{i,j} y_i y_j \right]
\]

\[
= \frac{1}{2} \left[ \sum_i \left( \sum_j a_{ij} \right) y_i^2 \right] + \sum_j \left( \sum_i a_{ij} \right) y_j^2 - 2 \sum_{i,j} a_{i,j} y_i y_j
\]

\[
= \frac{1}{2} \left[ \sum_{i,j} a_{ij} y_i^2 + \sum_{i,j} a_{ij} y_j^2 - 2 \sum_{i,j} a_{i,j} y_i y_j \right]
\]

\[
= \frac{1}{2} \left[ \sum_{i,j} a_{i,j} (y_i - y_j)^2 \right] \quad = \text{size of } \text{CUT}(y)
\]

\[ y^T (I - W)y = \text{size of } \text{NCUT}(y) \]

NCUT: roughly minimize ratio of transitions between classes vs transitions within classes
So far...

• **How to define a “good” partition of a graph?**
  – Minimize a given graph cut criterion

• **How to efficiently identify such a partition?**
  – Approximate using information provided by the eigenvalues and eigenvectors of a graph

• **Spectral Clustering**
Spectral Clustering Algorithms

• Three basic stages:
  – 1) **Pre-processing**
    • Construct a matrix representation of the graph
  – 2) **Decomposition**
    • Compute eigenvalues and eigenvectors of the matrix
    • Map each point to a lower-dimensional representation based on one or more eigenvectors
  – 3) **Grouping**
    • Assign points to two or more clusters, based on the new representation
Example: Spectral Partitioning
Example: Spectral Partitioning

Components of $x_2$

- Value of $x_2$
- Rank in $x_2$
Example: Spectral partitioning
k-Way Spectral Clustering

• How do we partition a graph into $k$ clusters?

• Two basic approaches:
  – Recursive bi-partitioning [Hagen et al., ’92]
    • Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    • Disadvantages: Inefficient, unstable
  – Cluster multiple eigenvectors [Shi-Malik, ’00]
    • Build a reduced space from multiple eigenvectors
    • Commonly used in recent papers
    • A preferable approach...
Why use multiple eigenvectors?

• Approximates the optimal cut [Shi-Malik, ’00]
  – Can be used to approximate optimal $k$-way normalized cut
• Emphasizes cohesive clusters
  – Increases the unevenness in the distribution of the data
  – Associations between similar points are amplified, associations between dissimilar points are attenuated
  – The data begins to “approximate a clustering”
• Well-separated space
  – Transforms data to a new “embedded space”, consisting of $k$ orthogonal basis vectors
• Multiple eigenvectors prevent instability due to information loss
Spectral Clustering: Graph = Matrix

\[ W \cdot v = \lambda v : v \text{ is an eigenvector with eigenvalue } \lambda \]

- smallest eigenvecs of \( D - A \) are largest eigenvecs of \( A \)
- smallest eigenvecs of \( I - W \) are largest eigenvecs of \( W \)

Q: How do I pick \( v \) to be an eigenvector for a block-stochastic matrix?
Spectral Clustering: Graph = Matrix

$W \cdot v = \lambda v$ : $v$ is an eigenvector with eigenvalue $\lambda$

How do I pick $v$ to be an eigenvector for a block-stochastic matrix?
Spectral Clustering: Graph = Matrix

\(Wv_1 = v_2\) “propogates weights from neighbors”

\[W \cdot v = \lambda v : v\ \text{is an eigenvector with eigenvalue } \lambda\]

- Smallest eigenvectors of \(D-A\) are largest eigenvectors of \(A\)
- Smallest eigenvectors of \(I-W\) are largest eigenvectors of \(W\)

Suppose each \(y(i) = +1\) or \(-1\):

- Then \(y\) is a cluster indicator that cuts the nodes into two
- What is \(y^T(D-A)y\)? The cost of the graph cut defined by \(y\)
- What is \(y^T(I-W)y\)? Also a cost of a graph cut defined by \(y\)
- How to minimize it?
  - Turns out: to minimize \(y^T X y \div (y^Ty)\) find *smallest* eigenvector of \(X\)
  - But: this will not be +1/-1, so it’s a “relaxed” solution
Spectral Clustering: Graph = Matrix

$W \cdot v = \lambda v : v$ is an eigenvector with eigenvalue $\lambda$

“eigengap” $\lambda_4$

$\lambda_1, \lambda_2, \lambda_3, \lambda_5, \lambda_6, \lambda_7, \ldots$

[Shi & Meila, 2002]
Some more terms

- If $A$ is an adjacency matrix (maybe weighted) and $D$ is a (diagonal) matrix giving the degree of each node
  - Then $D-A$ is the \textit{(unnormalized) Laplacian}
  - $W=AD^{-1}$ is a \textit{probabilistic adjacency matrix}
  - $I-W$ is the \textit{(normalized or random-walk) Laplacian}
  - etc....

- The largest eigenvectors of $W$ correspond to the smallest eigenvectors of $I-W$
  - So sometimes people talk about \textit{“bottom eigenvectors of the Laplacian”}
Figure 1: Toy example for spectral clustering where the data points have been drawn from a mixture of four Gaussians on $\mathbb{R}$. Left upper corner: histogram of the data. First and second row: eigenvalues and eigenvectors of $L_{rw}$ and $L$ based on the $k$-nearest neighbor graph. Third and fourth row: eigenvalues and eigenvectors of $L_{rw}$ and $L$ based on the fully connected graph. For all plots, we used the Gaussian kernel with $\sigma = 1$ as similarity function. See text for more details.
Figure 4: Three data sets, and the smallest 10 eigenvalues of $L_{rw}$. See text for more details.
Figure 3: Different similarity graphs, see text for details.
Spectral Clustering: Pros and Cons

• Elegant, and well-founded mathematically
• Works quite well when relations are approximately transitive (like similarity)
• Very noisy datasets cause problems
  – “Informative” eigenvectors need not be in top few
  – Performance can drop suddenly from good to terrible
• Expensive for very large datasets
  – Computing eigenvectors is the bottleneck
Use cases and runtimes

- K-Means
  - FAST
  - "Embarrassingly parallel"
  - Not very useful on anisotropic data
- Spectral clustering
  - Excellent quality under many different data forms
  - Much slower than K-Means
Further Reading

• Spectral Clustering Tutorial: http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/publications/Luxburg07_tutorial.pdf