Normalized Cuts Without Eigenvectors: A Multilevel Approach

Inderjit S. Dhillon
The University of Texas at Austin

SIAM Conference on Parallel Processing
February 24, 2006

Joint work with Yiqiang Guan and Brian Kulis
Clustering

Partitioning data into clusters arises in various applications in data mining & machine learning.

Examples:

- Bioinformatics: Identifying similar genes
- Text Mining: Organizing document collections
- Image/Audio Analysis: Image and Speech segmentation
- Web Search: Clustering web search results
- Social Network Analysis: Identifying social groups
- Other: Load balancing and circuit partitioning
In many applications, the goal is to partition/cluster the nodes of a graph:

High School Friendship Network

In many applications, the goal is to partition/cluster the nodes of a graph:

The Internet

[The Internet Mapping Project, Hal Burch and Bill Cheswick, Lumeta Corp, 1999]
Graph Clustering Objectives

How do we measure the quality of a graph clustering?
Graph Clustering Objectives

- How do we measure the quality of a graph clustering?
  - Could simply minimize the *edge-cut* in the graph
    - Can lead to clusters that are highly unbalanced in size
Graph Clustering Objectives

- How do we measure the quality of a graph clustering?

- Could simply minimize the *edge-cut* in the graph
  - Can lead to clusters that are highly unbalanced in size

- Could minimize the *edge-cut* in the graph while constraining the clusters to be equal in size
  - Not a natural restriction in data analysis
Graph Clustering Objectives

- How do we measure the quality of a graph clustering?

- Could simply minimize the edge-cut in the graph
  - Can lead to clusters that are highly unbalanced in size

- Could minimize the edge-cut in the graph while constraining the clusters to be equal in size
  - Not a natural restriction in data analysis

- Popular objectives include normalized cut, ratio cut and ratio association

  Normalized Cut: minimize \( \sum_{i=1}^{c} \frac{\text{links}(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{\text{degree}(\mathcal{V}_i)} \)

  Ratio Cut: minimize \( \sum_{i=1}^{c} \frac{\text{links}(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{|\mathcal{V}_i|} \)

[Shi & Malik, IEEE Pattern Analysis & Machine Intelligence, 2000]
[Chan, Schlag & Zien, IEEE Integrated Circuits & Systems, 1994]
Examples

Normalized Cut

Ratio Cut
Spectral Clustering

- Take a real relaxation of the clustering objective
Spectral Clustering

- Take a real relaxation of the clustering objective
- Globally optimal solution of the relaxed problem is given by eigenvectors
  - For ratio cut: compute smallest eigenvectors of the Laplacian $L = D - A$
  - For normalized cut: compute smallest eigenvectors of the normalized Laplacian $I - D^{-1/2} A D^{-1/2}$
- Post-process eigenvectors to obtain a discrete clustering
Spectral Clustering

- Take a real relaxation of the clustering objective

- Globally optimal solution of the relaxed problem is given by eigenvectors
  
  For ratio cut: compute smallest eigenvectors of the Laplacian $L = D - A$
  
  For normalized cut: compute smallest eigenvectors of the normalized Laplacian $I - D^{-1/2}AD^{-1/2}$
  
- Post-process eigenvectors to obtain a discrete clustering

- **Problem:** Can be expensive if many eigenvectors of a very large graph are to be computed
The $k$-means Algorithm

- Given a set of vectors and an initial clustering, alternate between computing cluster means and assigning points to the closest mean.

1. Initialize clusters $\pi_c$ and cluster means $m_c$ for all clusters $c$.
2. For every vector $a_i$ and all clusters $c$, compute

$$d(a_i, c) = \|a_i - m_c\|^2$$

and

$$c^*(a_i) = \arg\min_c d(a_i, c)$$

3. Update clusters: $\pi_c = \{a : c^*(a_i) = c\}$.
4. Update means: $m_c = \frac{1}{|\pi_c|} \sum_{a_i \in \pi_c} a_i$

5. If not converged, go to Step 2. Otherwise, output final clustering.
From $k$-means to Weighted Kernel $k$-means

- Introduce weights $w_i$ for each point $a_i$: use the weighted mean instead
From $k$-means to Weighted Kernel $k$-means

Introduce weights $w_i$ for each point $a_i$: use the weighted mean instead.

Expanding the distance computation yields:

$$\|a_i - m_c\|^2 = a_i \cdot a_i - \frac{2 \sum_{a_j \in \pi_c} w_j a_i \cdot a_j}{\sum_{a_i \in \pi_c} w_j} + \frac{\sum_{a_i, a_j \in \pi_c} w_j w_i a_j \cdot a_l}{(\sum_{a_j \in \pi_c} w_j)^2}$$
From $k$-means to Weighted Kernel $k$-means

- Introduce weights $w_i$ for each point $a_i$: use the weighted mean instead
- Expanding the distance computation yields:

$$
\|a_i - m_c\|^2 = a_i \cdot a_i - \frac{2 \sum_{a_j \in \pi_c} w_j a_i \cdot a_j}{\sum a_i \in \pi_c w_j} + \frac{\sum a_i, a_j \in \pi_c w_j w_i a_j \cdot a_l}{(\sum a_j \in \pi_c w_j)^2}
$$

- Computation can be done only using inner products of data points
From $k$-means to Weighted Kernel $k$-means

- Introduce weights $w_i$ for each point $a_i$: use the weighted mean instead
- Expanding the distance computation yields:

\[
\|a_i - m_c\|^2 = a_i \cdot a_i - \frac{2 \sum_{a_j \in \pi_c} w_j a_i \cdot a_j}{\sum_{a_i \in \pi_c} w_j} + \frac{\sum_{a_i, a_j \in \pi_c} w_j w_i a_j \cdot a_l}{(\sum_{a_j \in \pi_c} w_j)^2}
\]

- Computation can be done only using inner products of data points
- Given a *kernel* matrix $K$ that gives inner products in feature space, can compute distances using the above formula
From $k$-means to Weighted Kernel $k$-means

- Introduce weights $w_i$ for each point $a_i$: use the weighted mean instead
- Expanding the distance computation yields:

$$\|a_i - m_c\|^2 = a_i \cdot a_i - \frac{2 \sum_{j \in \pi_c} w_j a_i \cdot a_j}{\sum_{i \in \pi_c} w_j} + \frac{\sum_{i, j \in \pi_c} w_j w_i a_j \cdot a_l}{(\sum_{j \in \pi_c} w_j)^2}$$

- Computation can be done only using inner products of data points
- Given a kernel matrix $K$ that gives inner products in feature space, can compute distances using the above formula
- Objective function for weighted kernel $k$-means:

$$\text{Minimize } D(\{\pi_c^{k_{c=1}}\}) = \sum_{c=1}^{k} \sum_{a_i \in \pi_c} w_i \|\varphi(a_i) - m_c\|^2$$

where $m_c = \frac{\sum_{a_i \in \pi_c} w_i \varphi(a_i)}{\sum_{a_i \in \pi_c} w_i}$
The Weighted Kernel $k$-means Algorithm

Given a kernel matrix (positive semi-definite similarity matrix), run $k$-means in the feature space

1. Initialize clusters $\pi_c$
2. For every vector $\mathbf{a}_i$ and all clusters $c$, compute

$$d(\mathbf{a}_i, c) = K_{ii} - \frac{2 \sum_{\mathbf{a}_j \in \pi_c} w_j K_{ij}}{\sum_{\mathbf{a}_i \in \pi_c} w_j} + \frac{\sum_{\mathbf{a}_i, \mathbf{a}_j \in \pi_c} w_j w_l K_{jl}}{\left(\sum_{\mathbf{a}_j \in \pi_c} w_j\right)^2}$$

and

$$c^*(\mathbf{a}_i) = \arg\min_c d(\mathbf{a}_i, c)$$

3. Update clusters: $\pi_c = \{\mathbf{a} : c^*(\mathbf{a}_i) = c\}$.
4. If not converged, go to Step 2. Otherwise, output final clustering.
Equivalence to Graph Clustering

Surprising Theoretical Equivalence:

- Weighted graph clustering objective is *mathematically identical* to the weighted kernel $k$-means objective
Equivalence to Graph Clustering

Surprising Theoretical Equivalence:
- Weighted graph clustering objective is *mathematically identical* to the weighted kernel $k$-means objective
- Follows by rewriting both objectives as trace maximization problems
### Equivalence to Graph Clustering

- **Surprising Theoretical Equivalence:**
  - Weighted graph clustering objective is *mathematically identical* to the weighted kernel $k$-means objective.

- Follows by rewriting both objectives as trace maximization problems.

- Popular graph clustering objectives and corresponding weights and kernels for weighted kernel $k$-means given affinity matrix $A$:

<table>
<thead>
<tr>
<th>Objective</th>
<th>Node Weight</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio Association</td>
<td>1 for each node</td>
<td>$K = \sigma I + A$</td>
</tr>
<tr>
<td>Ratio Cut</td>
<td>1 for each node</td>
<td>$K = \sigma I - L$</td>
</tr>
<tr>
<td>Kernighan-Lin</td>
<td>1 for each node</td>
<td>$K = \sigma I - L$</td>
</tr>
<tr>
<td>Normalized Cut</td>
<td>Degree of the node</td>
<td>$K = \sigma D^{-1} + D^{-1} AD^{-1}$</td>
</tr>
</tbody>
</table>
Equivalence to Graph Clustering

Surprising Theoretical Equivalence:
- Weighted graph clustering objective is mathematically identical to the weighted kernel $k$-means objective.
- Follows by rewriting both objectives as trace maximization problems.

Popular graph clustering objectives and corresponding weights and kernels for weighted kernel $k$-means given affinity matrix $A$:

<table>
<thead>
<tr>
<th>Objective</th>
<th>Node Weight</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio Association</td>
<td>1 for each node</td>
<td>$K = \sigma I + A$</td>
</tr>
<tr>
<td>Ratio Cut</td>
<td>1 for each node</td>
<td>$K = \sigma I - L$</td>
</tr>
<tr>
<td>Kernighan-Lin</td>
<td>1 for each node</td>
<td>$K = \sigma I - L$</td>
</tr>
<tr>
<td>Normalized Cut</td>
<td>Degree of the node</td>
<td>$K = \sigma D^{-1} + D^{-1} AD^{-1}$</td>
</tr>
</tbody>
</table>

Implication: Can minimize graph cuts such as normalized cut and ratio cut without any eigenvector computation.
The Multilevel Approach

Overview of the approach

Input Graph

Final Clustering

Coarsening

Initial Clustering

Refining

[CHACO, Hendrickson & Leland, 1994]
[METIS, Karypis & Kumar, 1999]
The Multilevel Approach

- **Phase I: Coarsening**
  - Coarsen the graph by merging nodes together to form smaller and smaller graphs
  - Use a simple greedy heuristic specialized to each graph cut objective function
The Multilevel Approach

- **Phase I: Coarsening**
  - Coarsen the graph by merging nodes together to form smaller and smaller graphs
  - Use a simple greedy heuristic specialized to each graph cut objective function

- **Phase II: Base Clustering**
  - Once the graph is small enough, perform a base clustering
  - Variety of techniques possible for this step
The Multilevel Approach

- **Phase I: Coarsening**
  - Coarsen the graph by merging nodes together to form smaller and smaller graphs
  - Use a simple greedy heuristic specialized to each graph cut objective function

- **Phase II: Base Clustering**
  - Once the graph is small enough, perform a base clustering
  - Variety of techniques possible for this step

- **Phase III: Refining**
  - Uncoarsen the graph, level by level
  - Use weighted kernel \( k \)-means to refine the clusterings at each level
  - Input clustering to weighted kernel \( k \)-means is the clustering from the previous level
**Experiments: gene network**


- Normalized cut values generated by Graclus and the spectral method

<table>
<thead>
<tr>
<th># clusters</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graclus</td>
<td>0</td>
<td>.009</td>
<td>.018</td>
<td>.53824</td>
<td>3.1013</td>
<td>18.735</td>
</tr>
<tr>
<td>Spectral</td>
<td>0</td>
<td>.036556</td>
<td>.1259</td>
<td>.92395</td>
<td>5.3647</td>
<td>25.463</td>
</tr>
</tbody>
</table>
Experiments: gene network


Spy plots of the functional linkage matrix before and after clustering (128 clusters)—each dot indicates a non-zero entry

- Two example clusters: Histidine biosynthesis pathway and ATP synthase multiprotein complex

(A) Histidine biosynthesis pathway
(B) ATP synthase multiprotein complex
Experiments: IMDB movie data set

The IMDB contains 1.4 million nodes and 4.3 million edges.

Normalized cut values and computation time for a varied number of clusters, using Graclus and the spectral method

<table>
<thead>
<tr>
<th># clusters</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graclus</td>
<td>.049</td>
<td>.163</td>
<td>.456</td>
<td>1.39</td>
<td>3.72</td>
<td>9.42</td>
<td>24.13</td>
<td>64.04</td>
</tr>
<tr>
<td>Spectral</td>
<td>.00</td>
<td>.016</td>
<td>.775</td>
<td>2.34</td>
<td>5.65</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Normalized cut values—lower cut values are better

Computation time (in seconds)

<table>
<thead>
<tr>
<th></th>
<th>Graclus</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Grades</td>
<td>34.57</td>
<td>37.3</td>
<td>37.96</td>
<td>46.61</td>
<td>49.93</td>
<td>53.95</td>
<td>64.83</td>
<td>81.42</td>
</tr>
<tr>
<td>Spectral</td>
<td>261.32</td>
<td>521.69</td>
<td>597.23</td>
<td>1678.05</td>
<td>5817.96</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Experiments: IMDB movie data set

The IMDB contains 1.4 million nodes and 4.3 million edges.

- We generate 5000 clusters using Graclus, which takes 12 minutes.
- If we use the spectral method, we would have to store 5000 eigenvectors of length 1.4M; that is 24 GB main memory.

<table>
<thead>
<tr>
<th>Movies</th>
<th>Actors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry Potter and the Sorcerer’s Stone</td>
<td>Daniel Radcliffe, Rupert Grint,</td>
</tr>
<tr>
<td>Harry Potter and the Chamber of Secrets</td>
<td>Emma Watson, Peter Best,</td>
</tr>
<tr>
<td>Harry Potter and the Prisoner of Azkaban</td>
<td>Joshua Herdman, Harry Melling,</td>
</tr>
<tr>
<td>Harry Potter and the Goblet of Fire</td>
<td>Robert Pattinson, James Phelps,</td>
</tr>
<tr>
<td>Harry Potter and the Order of the Phoenix</td>
<td>Tom Felton, Devon Murray,</td>
</tr>
<tr>
<td>Harry Potter: Behind the Magic</td>
<td>Jamie Waylett, Shefali Chowdhury,</td>
</tr>
<tr>
<td>Harry Potter und die Kammer des Schreckens:</td>
<td>Stanislav Ianevski, Jamie Yeates,</td>
</tr>
<tr>
<td>Das grobe RTL Special zum Film</td>
<td>Bonnie Wright, Alfred Enoch, Scott Fern,</td>
</tr>
<tr>
<td>J.K. Rowling: Harry Potter and Me</td>
<td>Chris Rankin, Matthew Lewis, Katie Leung</td>
</tr>
<tr>
<td></td>
<td>Sean Biggerstaff, Oliver Phelps</td>
</tr>
</tbody>
</table>
Experiments: Image segmentation

- Leftmost plot is the original image and each of the 3 plots to the right of it is a component (cluster) — body, tail and background.

- Normalized cut value for this multilevel clustering is .022138, smaller than .023944 for spectral
## Experiments: Benchmark graph clustering

### Test graphs:

<table>
<thead>
<tr>
<th>Graph name</th>
<th>No. of nodes</th>
<th>No. of edges</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>copter2</td>
<td>55476</td>
<td>352238</td>
<td>helicopter mesh</td>
</tr>
<tr>
<td>memplus</td>
<td>17758</td>
<td>54196</td>
<td>memory circuit</td>
</tr>
<tr>
<td>pcrystk02</td>
<td>13965</td>
<td>477309</td>
<td>structural engineering</td>
</tr>
<tr>
<td>ramage02</td>
<td>16830</td>
<td>1424761</td>
<td>navier stokes and continuity equations</td>
</tr>
</tbody>
</table>
Experiments: Benchmark graph clustering

- Computation time:

![Graph 1: Computation time for normalized cut](image1)
![Graph 2: Computation time for ratio association](image2)
Experiments: Benchmark graph clustering

Quality (normalized cut and ratio association):

Normalized cut values scaled by those generated using spectral method

Ratio association values scaled by those generated using spectral method
Experiments: Benchmark graph clustering

Computation time comparison between Graclus and Metis
Conclusions

- Minimizing graph cuts such as the normalized cut is useful in many applications

- A mathematical equivalence between spectral graph clustering objectives and the weighted kernel $k$-means objective

- Multilevel algorithm uses kernel $k$-means in its refinement phase

- Experimental results show that the multilevel algorithm, as compared to a state-of-the-art spectral clustering algorithm:
  - Mostly outperforms spectral algorithm in terms of quality
  - Significantly faster
  - Requires much less memory