

SDP Notes

The Ratio Cut Problem

Given a fully connected, undirected similarity graph $G = (V, E)$ represented by weight matrix W , and known number of clusters/communities k , ratio cut partitions the graph by minimizing the objective function:

$$\arg \min_H \text{Tr}(H^\top L H)$$

where L is the unnormalized combinatorial Laplacian matrix $L = D - W$, D is the diagonal degree matrix $d_{ii} = \sum_j w_{ij}$, and $H \in \mathbb{R}^{n \times k}$ is restricted to the form
$$h_{ij} = \begin{cases} n_j^{-1/2} & \text{vertex } i \text{ is in cluster } j \\ 0 & \text{else} \end{cases}$$

This problem is NP-hard, which brings us to ...

Spectral Clustering

Note that $H^\top H = I$. So instead of restricting H to a cluster membership matrix, we just restrict H such that $H^\top H = I$. Then $H = [v_0 \ v_1 \ \dots \ v_{k-1}]$, or the matrix of k eigenvectors that correspond to the smallest k eigenvalues. Since this does not provide cluster memberships, the first k eigenvectors are then treated as an embedding of the graph G and k -means clustering is then applied (e.g., using Lloyd's algorithm) to obtain cluster memberships.¹

However, there isn't much theoretical justification for this approach, nor are there any guarantees that it will result in a clustering that minimizes the ratio cut objective, which brings us to ...

Semidefinite Programming Approach

Ling and Strohmer² showed that under certain conditions, the ratio cut problem can be solved exactly using semidefinite programming.

First, we note that $\text{Tr}(H^\top L H) = \text{Tr}(L H H^\top)$. Let us denote $Z = H H^\top$. We can see that $z_{ij} = n_k^{-1}$ if vertices i and j both belong to the same cluster (where n_k is the size of that cluster), otherwise it is 0. We can then note that $\text{Tr}(Z) = k$, the number of clusters, and Z is positive semidefinite of rank k . We also note that $Z e = e$ where e is a constant vector.

The SDP approach relaxes these constraints on Z but keeps more constraints than the spectral clustering approach. More formally, the SDP ratio cut algorithm is as follows:

1. Solve $\arg \min_Z \text{Tr}(L Z)$ subject to
 - Z is positive semidefinite
 - $Z \geq 0$ element-wise
 - $\text{Tr}(Z) = k$
 - $Z e = e$
2. Use the solution to the above to obtain cluster memberships.

¹This is almost (but not quite) the same as performing kernel k -means clustering using the Moore-Penrose pseudoinverse of L as a kernel matrix.

²<https://arxiv.org/abs/1806.11429>

The paper does not make it clear how to actually solve the semidefinite programming problem or how to recover cluster memberships from Z , since we are not guaranteed to obtain the “correct” form of Z using this method.

Possibly Interesting Question

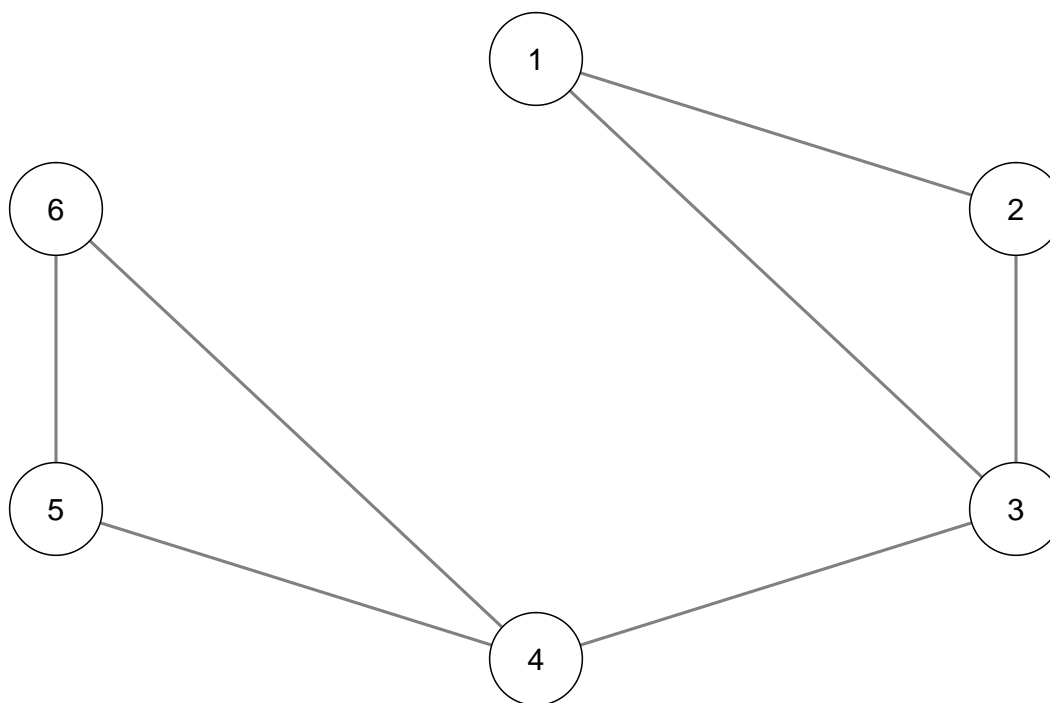
We previously noted that the spectral clustering approach to ratio cut is almost equivalent to kernel k -means using the pseudoinverse of L as a kernel matrix. One problem that we have been trying to solve is under what conditions does kernel k -means provide the optimal ratio cut clustering. Similarly, we might be interested in a kernel k -means version of the semidefinite programming approach:

Let L^\dagger be the pseudoinverse of L . Solve $\arg \max_Z \text{Tr}(L^\dagger Z)$ subject to

- Z is positive semidefinite
- $Z \geq 0$ element-wise
- $\text{Tr}(Z) = k$
- $Ze = e$

Some Examples

We will begin with a very simple graph with six vertices:



It is clear (and easy to verify) that cutting the 3 – 4 edge is optimal under the ratio cut objective. Using the semidefinite programming method:

```

$Z
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 0.333 0.333 0.333 0.000 0.000 0.000
[2,] 0.333 0.333 0.333 0.000 0.000 0.000

```

```
[3,] 0.333 0.333 0.333 0.000 0.000 0.000
[4,] 0.000 0.000 0.000 0.333 0.333 0.333
[5,] 0.000 0.000 0.000 0.333 0.333 0.333
[6,] 0.000 0.000 0.000 0.333 0.333 0.333
```

```
$clustering
```

```
[1] 2 2 2 1 1 1
```

We in fact get the exact solution to Z . The clustering here is based on a spectral decomposition of Z , which can have up to $n - 1$ nonzero eigenvectors (in this case we only have two). k -means is applied to this embedding.

We can also try the kernel k -means approach:

```
$Z
```

```
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 0.333 0.333 0.333 0.000 0.000 0.000
[2,] 0.333 0.333 0.333 0.000 0.000 0.000
[3,] 0.333 0.333 0.333 0.000 0.000 0.000
[4,] 0.000 0.000 0.000 0.333 0.333 0.333
[5,] 0.000 0.000 0.000 0.333 0.333 0.333
[6,] 0.000 0.000 0.000 0.333 0.333 0.333
```

```
$clustering
```

```
[1] 2 2 2 1 1 1
```

And in fact we get identical results as before for both Z and the clustering.

Let's try the epsilon graph example from the text. Recall that the graph Laplacian is of the form

$$\begin{bmatrix} \epsilon & -\epsilon & 0 & 0 \\ -\epsilon & 1+\epsilon & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \text{ Ratio cut and kernel } k\text{-means gave different answers when } \epsilon \in (0.6, 0.75).$$

Plugging in $\epsilon = 0.6$, the SDP method for ratio cut gives:

```
$Z
```

```
      [,1] [,2] [,3] [,4]
[1,] 0.777 0.223 0.000 0.000
[2,] 0.223 0.355 0.275 0.147
[3,] 0.000 0.275 0.370 0.355
[4,] 0.000 0.147 0.355 0.498
```

```
$clustering
```

```
[1] 1 2 2 2
```

While the SDP method for kernel k -means gives:

```
$Z
```

```
      [,1] [,2] [,3] [,4]
[1,] 0.683 0.317 0.000 0.000
[2,] 0.317 0.394 0.211 0.077
[3,] 0.000 0.211 0.394 0.394
[4,] 0.000 0.077 0.394 0.528
```

```
$clustering
```

```
[1] 2 2 1 1
```

Plugging in $\epsilon = .7$:

```
$Z
      [,1] [,2] [,3] [,4]
[1,] 0.708 0.292 0.000 0.000
[2,] 0.292 0.368 0.243 0.096
[3,] 0.000 0.243 0.389 0.368
[4,] 0.000 0.096 0.368 0.535
```

```
$clustering
[1] 1 1 2 2
```

While the SDP method for kernel k -means gives:

```
$Z
      [,1] [,2] [,3] [,4]
[1,] 0.581 0.419 0.000 0.000
[2,] 0.419 0.439 0.122 0.020
[3,] 0.000 0.122 0.439 0.439
[4,] 0.000 0.020 0.439 0.541
```

```
$clustering
[1] 1 1 2 2
```

So while we can get SDP ratio cut and SDP kernel k -means to disagree, the conditions are not identical to when this happens for the spectral clustering methods.