Notes on the SDP relaxation of k-means

The SDP relaxation of k-means

Iguchi et al.[1] formulated a semidefinite progmming approach to k-means as follows[2]:

$$\begin{aligned} \arg\max_{Z} \; & -\mathrm{Tr}(D_2 Z) \\ \text{s.t. } & \mathrm{Tr}(Z) = k \\ & Ze = e \\ & Z \geq 0 \text{ element-wise} \\ & Z \text{ is positive semidefinite} \end{aligned}$$

Where

- $D_2 = [d_{ij}] = [||x_i - x_j||^2]$
- $x_1, ..., x_n \in \mathbb{R}^q$
- The number of clusters, k, is known
- $e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^q$

Note that without the SDP relaxation, we have a rigid structure for Z where $z_{ij} = \begin{cases} n_k^{-1} & x_i, x_j \text{ in same cluster } k \\ 0 & \text{else} \end{cases}$

Equating the trace formulation of k-means to kernel k-means

We can see that the data matrix $X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix}$ is not explicitly in the objective, although squared Euclidean distances are. We can rewrite this as a kernel formulation by noting that $D_2 = \kappa(B)$ where B is a kernel matrix:

$$\begin{aligned} -\mathrm{Tr}(D_2 Z) &= -\mathrm{Tr}(\kappa(B) Z) \\ &= -\mathrm{Tr}((be^\top - 2B + eb^\top) Z) \\ &= 2\mathrm{Tr}(BZ) - \mathrm{Tr}(be^\top Z) - \mathrm{Tr}(eb^\top Z) \\ &= 2\mathrm{Tr}(BZ) - \mathrm{Tr}(be^\top) - \mathrm{Tr}(Zeb^\top) \\ &= 2\mathrm{Tr}(BZ) - \mathrm{Tr}(be^\top) - \mathrm{Tr}(eb^\top) \\ &= 2\mathrm{Tr}(BZ) - 2\mathrm{Tr}(be^\top) \\ &= 2\mathrm{Tr}(BZ) - 2\mathrm{Tr}(B) \end{aligned}$$

[1] https://arxiv.org/abs/1505.04778
[2] the notation is slightly different here

... where $b = \text{diag}(B)$, the vector of diagonal entries of B. Note that if we think of B as a weight matrix for an undirected graph, $\text{Tr}(B) = 0$. Similarly, if we impose that the diagonal entries of B are equal to 1 (e.g., B is a correlation matrix), then $\text{diag}(B) = n$. Either way, $\text{Tr}(B)$ does not depend on Z, so we can ignore it in the objective, and we can see that $\arg\max_Z \text{Tr}(D_2 Z) = \arg\max_Z \text{Tr}(BZ)$, which is just the typical kernel formulation of k-means:

$$\begin{aligned} &\arg\max_Z \text{Tr}(BZ) \\ &\quad \text{s.t. } \text{Tr}(Z) = k \\ &\qquad z_{ij} = \begin{cases} n_k^{-1} & x_i, x_j \text{ in same cluster } k \\ 0 & \text{else} \end{cases} \end{aligned}$$

Similarly, we can go from a kernel formulation of k-means to one based on squared Euclidean distances by noting that $D_2 = \tau(B)$. For simplicity of notation, we will rewrite $\arg\max_x f(x) = \arg\max_x 2f(x)$.

$$\begin{aligned} 2\text{Tr}(BZ) &= 2\text{Tr}(\tau(D_2)Z) \\ &= \text{Tr}(-PD_2PZ) \\ &= -\text{Tr}((I - n^{-1}ee^\top)D_2(I - n^{-1}ee^\top)Z) \\ &= -\text{Tr}((D_2 - n^{-1}D_2ee^\top - n^{-1}ee^\top D_2 + n^{-2}ee^\top ee^\top D_2)Z) \\ &= -\text{Tr}(D_2Z) + n^{-1}\text{Tr}(D_2ee^\top Z) + n^{-1}\text{Tr}(ee^\top D_2 Z) - n^{-2}\text{Tr}(ee^\top D_2 Z) \\ &= -\text{Tr}(D_2Z) + 2n^{-1}\text{Tr}(D_2ee^\top) - n^{-2}\text{Tr}(D_2ee^\top) \end{aligned}$$

Since the second and third terms do not depend on Z, we can discard them, and we get $\arg\max_Z \text{Tr}(BZ) = \arg\max_Z -\text{Tr}(D_2Z)$.[3]

Equating the SDP relaxation of k-means to the SDP relaxation of ratio cut

The ratio cut objective is:

$$\arg\min_Z \text{Tr}(LZ)$$

where L is the combinatorial graph Laplacian and Z has the same structure as before. If we relax the optimization problem by not enforcing Z to have this structure, we can see that:

$$\arg\min_Z \text{Tr}(LZ) = \arg\max_Z \text{Tr}(L^\dagger Z)$$

where $L^\dagger$ is the generalized inverse of L. Since $L^\dagger$ is positive semidefinite, it can be thought of as a kernel matrix, and we can apply the $\tau(\cdot)$ transformation to it to obtain D_2. In this case, D_2 is the expected commute time of the graph that generated L.

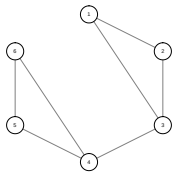
The argmin and argmax equivalence is not true in general if we force Z to have the structure that we want. It also is not true if we apply the SDP constraints (namely $Z \geq 0$ element-wise). One question of interest is under what conditions can we equate the two objectives under the SDP constraints.

[3] We can rewrite $\text{Tr}(D_2ee^\top) = \sum_{i,j} d_{ij}^2 = 2\sum_{i<j} d_{ij}^2$

Example 1

Here we look at a case where $\arg\min_Z \text{Tr}(LZ) = \arg\max_Z \text{Tr}(L^\dagger Z)$ under the SDP restrictions ($Ze = e$, $\text{Tr}(Z) = k$, Z is positive semidefinite, $Z \geq 0$ element-wise). In fact, in this example, not only do the two problems have the same solution, the solution coincides with the solution to the unrelaxed ratio cut problem.

Here we have a very simple graph with just six vertices. The "intuitive cut" for $k = 2$ is the $3 - 4$ cut, which happens to also be the solution to the (fully constrained) ratio cut problem.



The RatioCut-SDP solution (Lee and Strohmer)[4] yields:

```
rc.sdp(L, k = 2)$Z %>%
  MASS::fractions()
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 1/3  1/3  1/3    0    0    0
[2,] 1/3  1/3  1/3    0    0    0
[3,] 1/3  1/3  1/3    0    0    0
[4,]   0    0    0  1/3  1/3  1/3
[5,]   0    0    0  1/3  1/3  1/3
[6,]   0    0    0  1/3  1/3  1/3
```

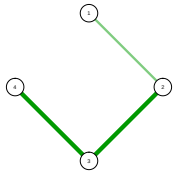
The k-means SDP solution using $L^\dagger$ as a kernel matrix (adapted from Peng and Wei)[5] yields:

```
kmeans.sdp(L.dagger, k = 2)$Z %>%
  MASS::fractions()
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 1/3  1/3  1/3    0    0    0
[2,] 1/3  1/3  1/3    0    0    0
[3,] 1/3  1/3  1/3    0    0    0
[4,]   0    0    0  1/3  1/3  1/3
[5,]   0    0    0  1/3  1/3  1/3
[6,]   0    0    0  1/3  1/3  1/3
```

Example 2

Let's try the epsilon graph example from the text. This graph has four vertices connected in series by three edges. The $2 - 3$ and $3 - 4$ edges have weight 1 but the $1 - 2$ edge has weight $\epsilon \in (0, 1)$.

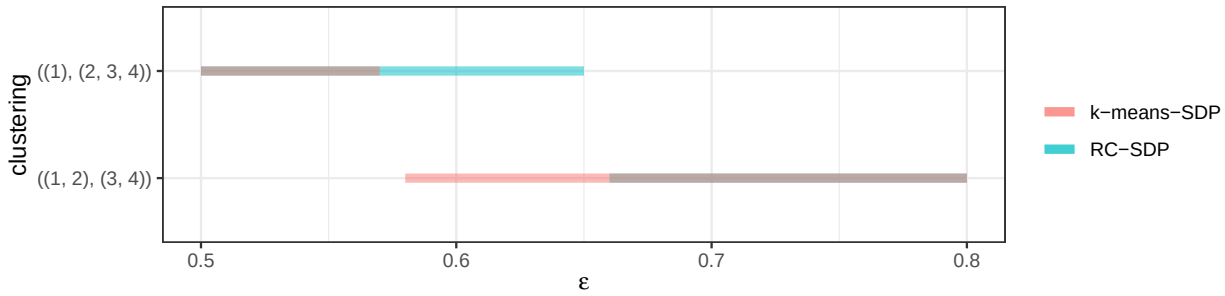


[4] https://arxiv.org/abs/1806.11429
[5] http://www.optimization-online.org/DB_FILE/2005/04/1114.pdf

Recall that the graph Laplacian is of the form

$$\begin{bmatrix} \epsilon & -\epsilon & 0 & 0 \\ -\epsilon & 1+\epsilon & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

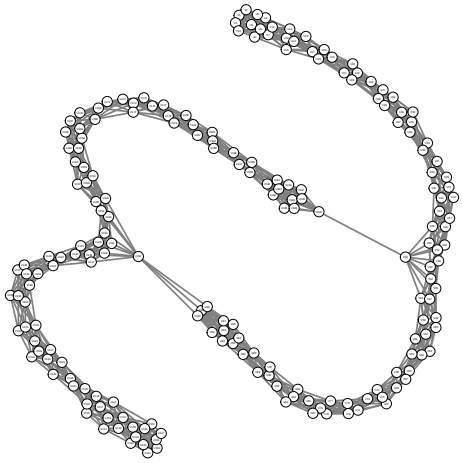
We previously showed that when $\epsilon \in (0, 0.75)$, the optimal ratio cut is the $1-2$ cut, and when $\epsilon \in (0.75, 1)$, the optimal ratio cut is the $2-3$ cut. However, for kernel k-means, the optimal clustering is $\{\{1\}, \{2,3,4\}\}$ when $\epsilon \in (0, 0.6)$ and $\{\{1,2\}, \{3,4\}\}$ when $\epsilon \in (0.6, 1)$. One thing that might be of interest is whether RatioCut-SDP and k-means-SDP yields the same results.



Interestingly, both RatioCut-SDP and k-means-SDP fail to find the correct ϵ where the optimal cut switches from $1-2$ to $2-3$.

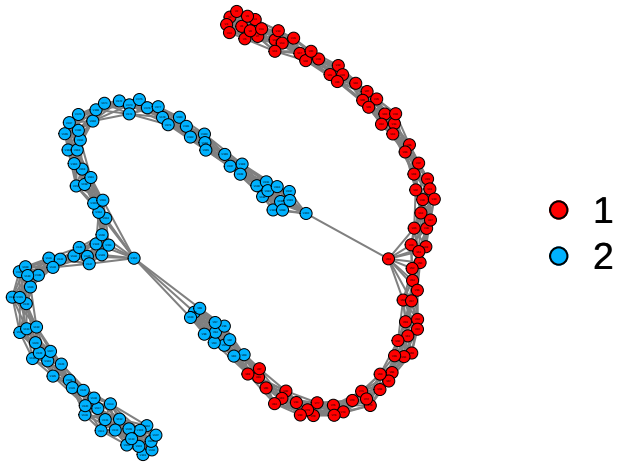
Example 3

Here we look at the "spiral graph":

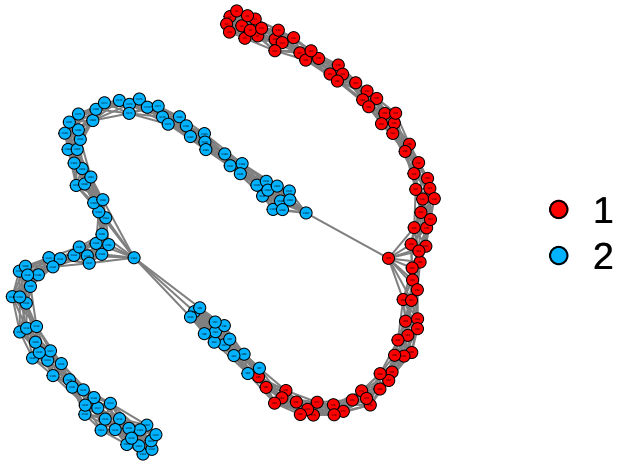


We know that the optimal ratio cut clustering is $\{\{1, ..., 100\}, \{101, ..., 200\}\}$. This also happens to coincide with the optimal k-means clustering. We also know that the spectral clustering relaxation of ratio cut fails to provide the optimal ratio cut (even when the k-means rounding step is initialized with the correct clustering).

Applying RatioCut-SDP to this graph:



Here we see that RatioCut-SDP makes the same mistake the spectral clustering relaxation of ratio cut makes. We can also try k-means SDP:



Strangely, k-means-SDP makes a similar mistake RatioCut-SDP makes, while performing k-means on the embedding of $L^\dagger$ is able to find the optimal ratio cut (although Lloyd's algorithm runs into local minima issues as we add more embedding dimensions, so it's not always practically feasible).

It's also worth noting that these optimization problems takes a while to solve.

Next steps

1. RatioCut-SDP provides $Z \in \mathbb{R}^{n \times n}$, which, under the full ratio cut constraints, should be $Z = HH^\top$. We saw that in example 1, both RatioCut-SDP and k-means-SDP were able to find this exact Z, but in examples 2 and 3, we needed a "rounding" step where we embedded Z and used Lloyd's algorithm to find a clustering. Lee and Strohmer are not clear on how this should be handled, and neither are Peng and Wei.

2. Lee and Strohmer provide some criteria under which RatioCut-SDP is guaranteed to find the optimal ratio cut clustering. So far, we haven't checked if these criteria hold before running these experiments.
3. More examples ...
4. Figure out when RatioCut-SDP and k-means SDP (using $L^\dagger$ as a kernel matrix) coincide.