Aalto University

Algorithmic Methods of Data Mining

CS-E4600 — Programming project

Spectral Graph Partitioning

Author:
Rohan Chauhan
661562 (Team 29)
rohan.chauhan@aalto.fi

Supervisor:
Aristides GIONIS

December 21, 2018

Aalto University
School of Science

Abstract

Graph Partitioning is very useful for solving problems like load balancing, VLSI layout and matrix multiplication. It is the problem of partitioning a graph G into smaller components according to the given requirements. In this project, we implement spectral graph partitioning algorithm using normalized and unnormalized laplacian and evaluate it on nine graph datasets. We compare the results of algorithm to make well balanced partitions using the competition criteria. We conclude by presenting our findings and suggesting improvements for future work.

1 Introduction

Graph Partitioning is the problem in which given a graph G(V,E,k), we need to partition the vertices of graph into k sets such that $V_1 \cup V_2 \cup V_3 ... V_k = V$. Here, V stands for vertices in the graph, E is the number of edges and k is the number of communities which should be greater than 1. Partitioning is easy to achieve by just separating k single vertex from the graph but not our desired result as it would not be useful. We want to make these partitions as balanced as possible so that vertex that are similar stay in the same partition and should be joined by the minimum number of edges to the other partition. One of the important use cases for balanced partitioning is in parallel computing. We want to partition the data between the processes such that there is minimum communication overhead [1].

In this project, we implement unnormalized spectral graph clustering algorithm [4]. Spectral graph clustering uses eigenvalues and eigenvectors to project a graph into low dimensional space where it can be easily clustered. The rest of the paper is organized as follows: In section 2, we discuss the background material for understanding spectral graph partitioning. Section 3 discusses the algorithm and implementation details of the project. Section 4 discusses the performance of the graph partitioning algorithm. Finally, conclusions are drawn in section 5.

2 Literature Review

A undirected graph G can be represented in terms of an adjacency matrix A of dimension (V,V) where V is the number of vertices in graph G. A element of adjacency matrix is given by:

$$A_{ij} = \begin{cases} 1 & \text{edge exists between } v_i \text{ and } v_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The degree of a vertex is defined as the number of edges whose endpoint lies at the vertex. A degree diagonal matrix D for graph G is defined as a diagonal matrix where it's element is given by

$$D_{ij} = \begin{cases} degree(\mathrm{v}_i) & \text{when i = j} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

A laplacian matrix can be used to study many important properties of a graph. For spectral graph clustering, we use it to project graph to a lower dimensional embedding. We calculate the eigenvectors and eigenvalues from the laplacian matrix. The smallest k eigenvectors can then be clustered easily using K-means algorithm to partition the graph into k communities.

The unnormalized laplacian matrix is given by

$$L = D - A$$

where D is the degree diagonal matrix and A is the adjacency matrix. The normalized laplacian matrix is given by

$$L = D^{\frac{-1}{2}} A D^{\frac{-1}{2}}$$

where D is the degree diagonal matrix and A is the adjacency matrix [2]. Both the unnormalized and normalized laplacian matrix are nearly the same for a Graph. However, the normalized laplacian matrix is used for Ncut and the unnormalized laplacian is used for RatioCut [3].

A cut is partition of graph into two sets where cut set denotes the edges whose one endpoint is in first set and the other endpoint is in the other set. The objective function for graph partition is to minimize the number of edges in cut set. However, this objective function can lead to the problem where graph partitions have only 1 vertex. So, we reframe the graph partition objective in terms of Ncut and RatioCut.

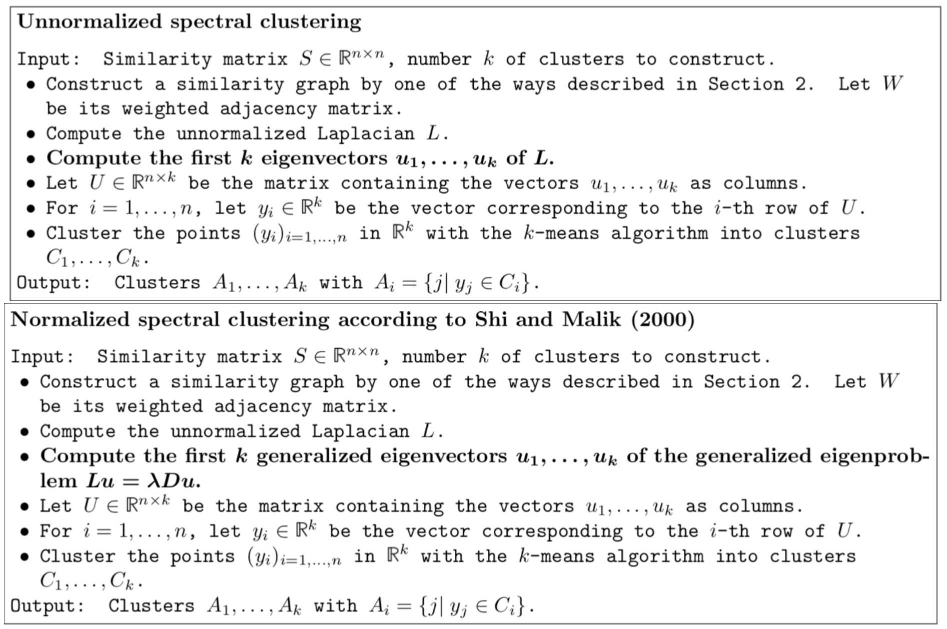
$$RatioCut = \frac{1}{2} \sum_{n=1}^{k} \frac{Cut(A_i, \overline{A_i})}{\mid A_i \mid}$$

$$Ncut = \frac{1}{2}\sum_{n=1}^{k}\frac{Cut(A_i,\overline{A_i})}{vol(A_i)}$$

where $\mid A_i \mid$ denotes the number of vertices in the partition and $vol(A_i)$ denotes the sum of the degree of vertices in the partition [4]. The value for RatioCut and Ncut will be minimum when the algorithm creates balanced partitions. However, discussion regrading the use of normalized laplacian for Ncut and unnormalized laplacian for RatioCut is beyond the scope of project.

3 Method

We have used two algorithms for spectral clustering, one with unnormalized laplacian matrix and other with normalized laplacian matrix [4]. We describe their respective algorithms below:



Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n\times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k eigenvectors $u_1, \dots, u_k$ of L.**
- Let $U \in \mathbb{R}^{n\times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
- For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \dots, C_k$.

Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n\times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k generalized eigenvectors $u_1, \dots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.**
- Let $U \in \mathbb{R}^{n\times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
- For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \dots, C_k$.

Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Figure 1: Images of both of the algorithm have been taken from A tutorial on Spectral Clustering by Ulrike von Luxburg [4]

We have provided the implementation of both of the algorithms in the code section. During implementation, initially we were using numpy for calculating the laplacian matrix and then finding the eigenvalues and eigenvectors. However, this method was not well suited for large graphs so we resorted to using networkx library to calculate the laplacian matrix. We also found out that scipy has more robust implementation for finding eigenvalues from matrix, so we used the sparse version. We also preprocessed the graph by finding the largest connected component and removing self edges.

4 Experimental Results

We present the results for normalized spectral clustering and unnormalized spectral clustering in Table 1 which were obtained after uploading the result files on google drive.

Graph Name	**Vertices**	**Edges**	**K**	**Unnormalized SC**	**Normalized SC**
ca-GrQc	4158	13428	2	0.0833	0.2286
Oregon-1	10670	22002	5	15.5	559.7
ca-HepTh	8638	24827	20	5.8	22.4
ca-CondMat	21363	91342	100	112.6667	264.25
ca-HepPh	11204	117649	25	14.5	143.4
ca-AstroPh	17903	197031	50	58	359
soc-Epinions1	75879	405740	10	4.0	4.0
web-NotreDame	325729	1117563	20	73584.0	73584.0
roadNet-CA	1957027	2760388	50	NA	NA

From the results, it is clear that, unnormalized spectral clustering obtains better score than normalized spectral clustering because the scoring criteria for the project similar to RatioCut where the change is that the denominator is replaced by the size of minimum partition. We weren't able to find eigenvalues and eigenvalues for larger graphs accurately. Eigenvectors approximated by decreasing precision gave bad results for both normalized and unnormalized spectral clustering as evident in the case of larger graphs.

We also visualize the result of unnormalized spectral clustering for ca-GrQc as it is easier to visualize due to the small k value.

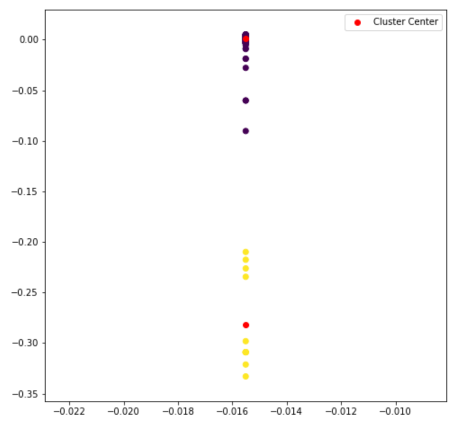


Figure 2: Spectral Graph Partitioning of ca-GrQc

5 Conclusion

We learned a lot about linear algebra and graphs while working on the project. Spectral graph partitioning is easy to implement and efficient technique for graph partitioning. One of the major drawback, we experienced was that finding eigenvectors and eigenvalues can take a lot of time for bigger networks even after approximating these values by decreasing precision. This also results in poor clustering.

There were several other variations of K-means which we could have tried to improve our performance in the competition. One of the variations is in which we choose the minimum number of points to be present in each cluster and the other variation is where clusters are of same size even though the assignment of a point to a cluster may not be perfect. It would have been nice to plot how results change on increasing exponentially the minimum number of point in each cluster from one to n/k.

References

[1] Andreev, K., and Racke, H. Balanced graph partitioning. *Theory of Computing Systems 39*, 6 (2006), 929–939.

[2] Chung, F. R. Spectral graph theory (cbms regional conference series in mathematics, no. 92).

[3] Ding, C. A tutorial on spectral clustering. In *Talk presented at ICML.(Slides available at http://crd. lbl. gov/ cding/Spectral/)* (2004).

[4] Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing 17*, 4 (2007), 395–416.

6 Code

6.1 Unnormalized Spectral Clustering

```
import networkx as nx
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from sklearn.cluster import KMeans
import scipy
import scipy.sparse as sparse

normal_graphs = ['ca-AstroPh', 'ca-CondMat', 'ca-GrQc', 'ca-HepPh', '
    ca-HepTh']
competition_graphs = ['web-NotreDame', 'roadNet-CA','Oregon-1', 'soc
    -Epinions1']

normal_graph_path = '../graphs_part_1/'
competition_graph_path = '../graphs/'
storage_path = '../ results /lap'

for graph_name in normal_graphs:

    filename = os.path.join(normal_graph_path, graph_name+'.txt')

    with open(filename,'r') as f:
        metadata = f.readline(). split ()
    graph_name = metadata[1]
    graph_vertices = int(metadata[2])
    graph_edges = metadata[3]
    graph_k = int(metadata[4])
    print(graph_name)

    G = nx.read_edgelist(filename, comments='#', delimiter=' ', nodetype
        =int, create_using=nx.MultiGraph())
    G = max(nx.connected_component_subgraphs(G), key=len)
    G.remove_edges_from(G.selfloop_edges())


    unormalized_laplacian = nx.laplacian_matrix(G)
    print('Finding Eigen')
    eigenvalues, eigenvectors = sparse.linalg.eigs(unormalized_laplacian.
        asfptype(),which='SM',k=graph_k)
    print('Starting K-means')
    klabels = KMeans(n_clusters=graph_k,random_state=0,max_iter
        =1000,n_init=50).fit(eigenvectors[:,:graph_k].real, y=None,
        sample_weight=None)
    print('Writing Partitions')
    f = open(os.path.join(storage_path, graph_name+".output"),'w+')
    f.write(" ".join(metadata)+'\n')

    vertices = list(G.nodes())
    for vertex in range(len(vertices)):
        f.write(str(vertices[vertex]) + ' ' + str(klabels.labels_[vertex
            ]) + '\n')
    f.close()
```

6.2 Normalized Spectral Clustering

```
import networkx as nx
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
```

```
from sklearn.cluster import KMeans
import scipy
import scipy.sparse as sparse

normal_graphs = ['ca−AstroPh', 'ca−CondMat', 'ca−GrQc', 'ca−HepPh', '
    ca−HepTh']
competition_graphs = ['web−NotreDame', 'roadNet−CA','Oregon−1', 'soc
    −Epinions1']

normal_graph_path = '../graphs_part_1/'
competition_graph_path = '../graphs/'
storage_path = '../ results /lap'

for graph_name in normal_graphs:

    filename = os.path.join(normal_graph_path, graph_name+'.txt')

    with open(filename,'r') as f:
        metadata = f.readline(). split ()

    graph_name = metadata[1]
    graph_vertices = int(metadata[2])
    graph_edges = metadata[3]
    graph_k = int(metadata[4])
    print(graph_name)

    G = nx.read_edgelist(filename, comments='#', delimiter=' ', nodetype
        =int, create_using=nx.MultiGraph())
    G = max(nx.connected_component_subgraphs(G), key=len)
    G.remove_edges_from(G.selfloop_edges())


    normalized_laplacian = nx.normalized_laplacian_matrix(G)
    print('Finding Eigen')
    eigenvalues, eigenvectors = sparse. linalg . eigs(normalized_laplacian.
        asfptype(),which='SM',k=graph_k)
    print('Starting K−means')
```

```
klabels = KMeans(n_clusters=graph_k,random_state=0,max_iter
    =1000,n_init=50).fit(eigenvectors[:,:graph_k].real, y=None,
    sample_weight=None)
print('Writing Partitions')
f = open(os.path.join(storage_path, graph_name+".output"),'w+')
f.write(" ".join(metadata)+'\n')

vertices = list(G.nodes())
for vertex in range(len(vertices)):
    f.write(str(vertices[vertex]) + ' ' + str(klabels.labels_[vertex
        ]) + '\n')
f.close()
```