L42: Machine Learning and Algorithms for Data Mining

Assignment 1 – Topic: Spectral Partitioning of Random Graphs

Philippos Papaphilippou (pp417), Wolfson College
Lent Term 2017

1. Introduction

A lot of research work on spectral clustering evaluates algorithms using random graphs based on the stochastic block model. The lecture notes of Spielman[1] focus on a simplified algorithm for spectral clustering to predict the partition of random graphs based on the planted bisection model.

In the following section there is an explanation of the basic principles that have been used for this work, including the planted bisection model and the presented simple spectral clustering algorithm for bisections, which makes the use of the eigenvector of the 2^{nd} largest eigenvalue to guess the two sets of the partition.

Finally, I present some experimental results that show the performance of the algorithm for different parameter values, as well as the application of the algorithm in a more general case.

2. Basic Principles

In this section there are the basic principles behind the paper's algorithm. The general idea is to apply a spectral clustering algorithm on random graphs that contain partitions using the planted partition model.

2.1 The Planted Partition Model

The Planted Partition Model is a method that we can use to construct a random graph with specific structural characteristics. By using the following general model we can generate a structured random graph from n nodes of k different classes. [3]

> Let $\psi : \{1, \ldots, n\} \rightarrow \{1, \ldots, k\}$ the partition.
> Let A the adjacency matrix of the generated graph $G(\psi, A)$.
> For all pairs (u, v), A[u,v] == A[v,u] == 1 with probability P $\psi(u)\ \psi(v)$

There are many well known models that fall into the planted partition model. A fraction of the graphs that are created with this model are planted multisections[4] where we have k classes of nodes and each intra-cluster edge appears with probability p and each inter-cluster edge with probability q, with p>q. If q is equal to 0 we have the k-coloring problem, where the clusters are not connected. We can also produce a planted clique, where we have a random graph of connectivity probability equal to p, but the edges connecting a specified subset of vertices appear with probability 1.

In the lecture notes[1], a simplified model is used and is based on the planted multisections problem, but k=2 and the number of vertices in each cluster is equal to n/2. This is also referred to as the graph bisection model.

> Let $\psi : \{1, \ldots, n/2\} \in B_1, \{n/2+1, \ldots, n\} \in B_2$
> Let A the adjacency matrix of the generated graph $G(\psi, A)$.
> For all pairs (u, v), A[u,v] == A[v,u] == 1 with probability p if $\psi(u) = \psi(v)$, q otherwise

In Figure 1 we can see some visualizations for example random graphs that I have constructed using this model for different values of q. By looking at the figure, we could predict that the closer q is to p, the more difficult the clustering is going to be. Another observation could be that the same q might behave differently for different ns in respect to the accuracy of the algorithms.

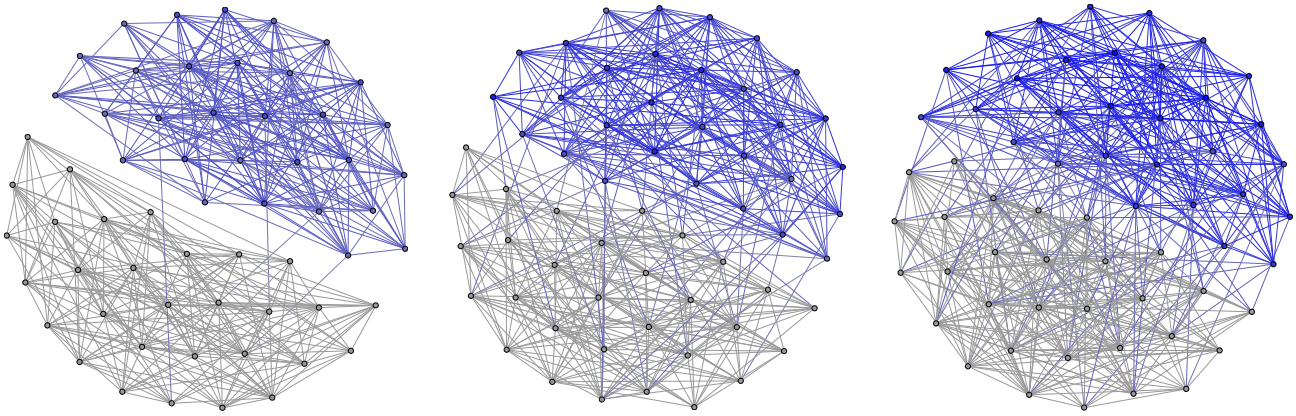

Figure 1: Visualisations of 3 random graphs using the simplified model described in 2.1. n=60 From left to right p=1/2 q=0.005, p=1/2 q=0.05, p=1/2 q=0.01. The input of these gephi visualisations has been produced by the python code that accompanies this report.

3. Spectral Clustering Algorithm

The spectral clustering algorithm on random graphs that were generated using the graph bisection model is based in the 2nd largest eigenvalue of the graph's adjacency matrix. [1]

1. Select a partition with n/2 elements in each set of vertices
2. Select the parameters of n and ϵ, where is used to calculate q, the probability of drawing each inter-cluster edge, so that p<q. $q = n - \frac{\epsilon}{\sqrt{n}}$
3. Construct the adjacency matrix A as described in the graph bisection model above
4. Find the second largest eigenvalue w_2 and calculate its corresponding eigenvector v_2
5. Set $S_1 = \{a : v_2(a) < 0\}$ and $S_2 = \{a : v_2(a) \geq 0\}$ the predicted clusters of the bisection.

4. Practical Evaluation

In this section I am going to evaluate the simple algorithm that is described in the lecture notes [1], experimentally. The source code can be divides in 3 phases:

1. Generation of a random graph based on the planted partition model
2. Shuffling of the vertex naming and the adjacency matrix accordingly. This is for validation that the prediction will not be based on the spatial localities of the values in the matrix
3. Construct the 2 sets according to the predicted partitioning
4. Compare with the original partitioning

As a basic evaluation metric I will be using the average percentage of the correctly labeled vertices. When we have only two clusters, this value is always between 50% and 100%. This is because my implementation's last step is to assign the two resulting sets to the most matching initial planted partition. This statement can be easily proven by induction. This metric is commonly refereed as percentage of accuracy.

In my python code I use the numpy library to calculate the eigenvectors, although there are quicker algorithms for this particular problem [2]. Only the second eigenvector is needed and by using a the Laplacian solver it can be approximated in near-linear time [2] (Chapter 8 Graph Partitioning IV).

Here I present the results of some sets of experiments to explore the effects of manipulating the author's defined variables and also for more general cases. The baseline configuration for all runs is n=200 (100 vertices in each cluster), p=1/2 and ϵ=4.5 $\Rightarrow q = \frac{1}{2} - \frac{4.5}{\sqrt{200}} \approx 0.182$.

When exploring many variables at the same time I am using a separate python script to work as a threadpool and accelerate the exploration procedure in multicore systems. The parameters are passes are command line arguments when executing from the threadpool. The observed percentage of the correctly labeled vertices is an average of the result of many (e.g 25 to 100) instances of the same configuration.

4.1 Varying p and ϵ

In Figure 2 we can see the relationship between p and ϵ in regards the achieved percentage of accuracy. We can observe that when ϵ is close to 1 the accuracy drops dramatically. This is because the closer ϵ is getting to 1, the closer q becomes to p. Having the very similar probability for edges between all nodes makes the planted partitioning practically indistinguishable from a uniformly random graph.

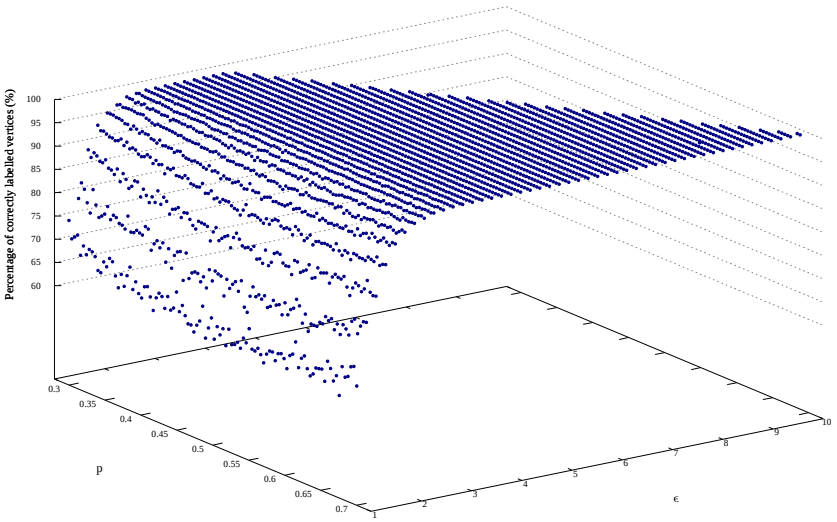

Figure 2: Relationship between p, ϵ and the achieved percentage of accuracy

The triangular shape of the data set is due to the fact that the missing points from the graph are combinations of p and ϵ that make $q \leq 0$. When $q \leq 0$ our graph is not connected. In these case, it would be easy to distinguish the two clusters with a trivial algorithm of O(n) complexity.

The Figure 3 is a similar set of experiments with the only difference being to vary the parameter q directly instead of ϵ and for wider ranges. The observation from Figure 2 is the same as Figure 1, that the algorithm performs well when q is not close to p. The noise in this graph is when q is close to 0 and the graph is not connected as this algorithm does not take into consideration this case.

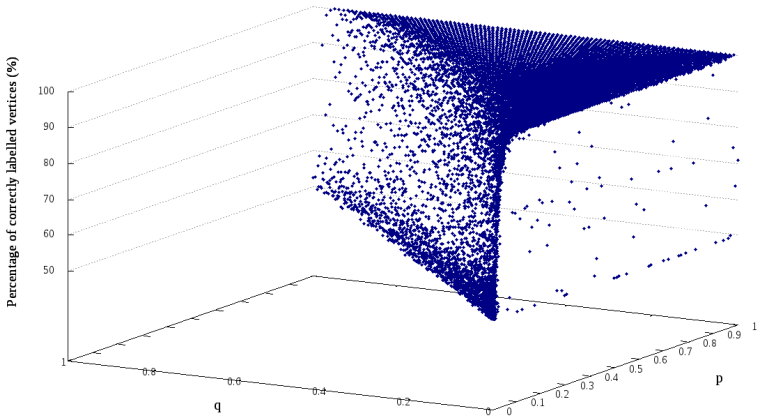

Figure 3: Relationship between p, q and the achieved percentage of accuracy

4.2 Varying the number of nodes per cluster

As suggested in the notes paper[1] , the planted partition models are used to evaluate a variety of spectral clustering algorithms, but may not be the most representative way of predicting performance on real datasets.

If we wanted to use the simplified algorithm[1] for real data, the main restriction would be that it is designed for the simple case of two clusters having n/2 nodes each. In this section I present the effects of varying the number of nodes for each cluster. In Figure 4 we can see the resulting accuracy percentage when applying the algorithm to clusters of varying size. The empty are in the graph is where q becomes less than 0.

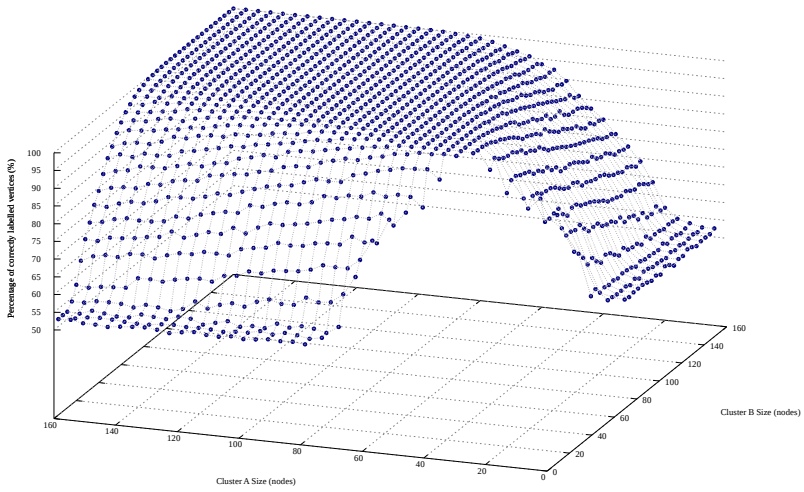

Figure 4: Relationship the size of each of the 2 clusters and the achieved percentage of accuracy,

When the number of nodes in each cluster is the same (across the x=y line) we can see that after a specific n the prediction becomes perfect. When $x \neq y$, the results are not always promising. It seems that the algorithm

can cope with a certain degree of differentiation between x and y. Visually, the size of one clusters needs to be at least the half of the other for an excellent prediction rate.

It would be interesting to show the statistical significance of the measurements to reach to more concrete conclusions.

4.3 Statistical properties of the accuracy percentage results

In this section there are 3 different experiments to measure the standard deviation of the accuracy percentage when varying different parameters. A simple approach for easy visualisation is to keep steady one of the parameters n, q(or ϵ), p and have the other two as independent variables. The error bars are shown in a two dimensional plot and therefore one of the parameters can exist as a separate set of data series.

In Figure 5, we can see how n affects the results mean and standard deviation from mean. We can observe that ϵ is indeed a better parameter to express the connectivity between the 2 clusters. This is because it seems to be independent to n and the orthogonality between n and ϵ can help our exploration by using less parameters in our design space. In practice, when exploring ϵ we can safely set the n a specific value. On the other hand, the standard deviation of the accuracy results changes substantially as n varies. This also tells as that for different ns the shape of the Figure 3 could substantially and Figure 2 is more likely to be representative for different ns.

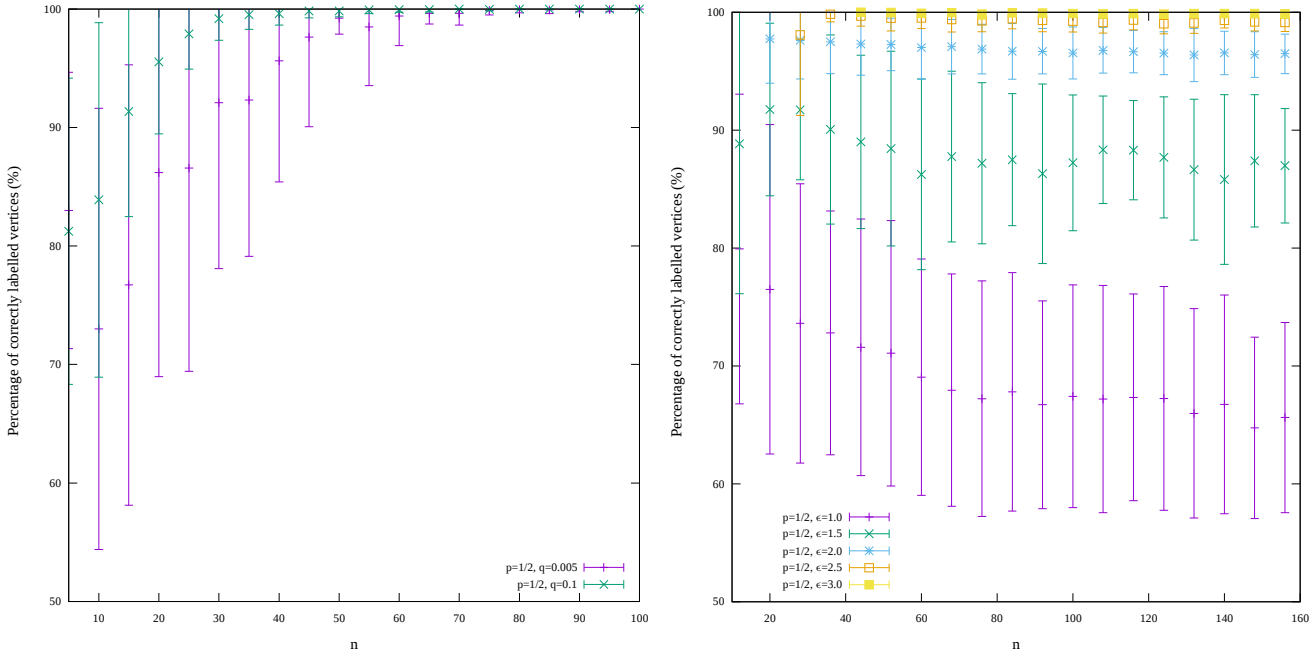

Figure 5: Relationship of n and the mean and standard deviation from mean of the achieved percentage of accuracy, for different values of q (left), and different values of ϵ (right)

Figure 5 (right) also explains what to expect for different ϵs in terms of the resulting accuracy of the dataset.

In Figure 6 we can see that this kind of orthogonality is also present between p and ϵ. There are only 5 values for ϵ but the trend is clear for larger ϵs. This is also why the x value (p) in Figure 2 does not change the shape of the surface across this axis.

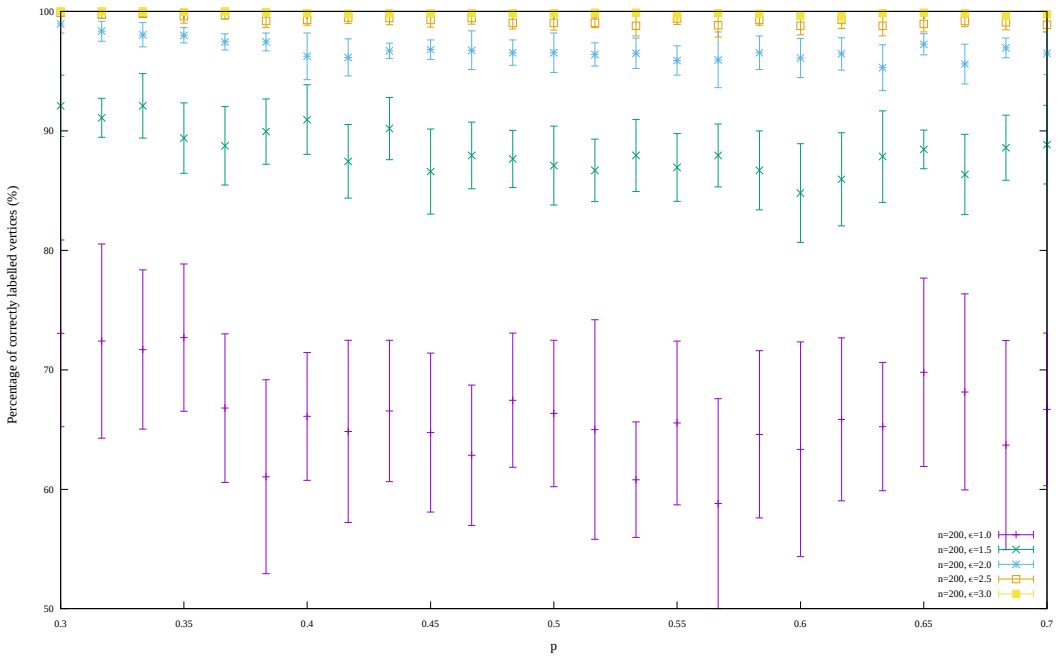

Figure 6: Relationship of p and the mean and standard deviation from mean of the achieved percentage of accuracy, for different values of ϵ

~1800 words

5. References

[1] Daniel Spielman. Spectral *Partitiong in a Stochastic Block Model* Spectral Graph Theory, Course 2016. http://www.cs.yale.edu/homes/spielman/561/lect21-15.pdf

[2] Nisheet Vishnoi. *Lx = b*, Foundations and Trends in Theoretical Computer Science, Vol. 8, Nos. 1-2 (2012), pp 1-141

[3] F. McSherry. *Spectral partitioning of random graphs*. In FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science, page 529, 2001

[4] Zhao, Yufei. *Spectral Techniques For Partitioning Planted Random Graphs* 2013