Bayesian Graph Partitioning

STAT-S 626 Final Project

John Koo

Abstract

Clustering is an important task both statistical data analysis and machine learning. Most often, clustering is done on data in $\mathbb{R}^p$. Graph partitioning aims to cluster data that are not represented as points in Euclidean space but as relationships in a graph or network. In this paper, we try a Bayesian approach and assess its "performance" against other methods.

Code available at https://github.com/johneverettkoo/stats-hw/tree/master/stat-s626

Introduction

Graph partitioning (also known as community detection) involves taking graph data and identifying partitions/clusters/communities in an unsupervized way. Let $G = (V, E)$ be an undirected graph with n vertices where the edges represent similarities (or affinities). The goal of graph partitioning is to identify $k \ll n$ partitions of V in some reasonable way. In particular, if Γ is the matrix of edge similarities of V, then $\sum\limits_{\substack{i,j \text{ in different} \\ \text{clusters}}} f(\gamma_{ij})$ should be small for some metric f. In practice, this objective function includes some regularization so that $k \ll n$ or the size of each cluster stays relatively balanced. This is by definition an ill-posed problem, and the solution can vary depending on f, k, the particular objective function, underlying assumptions on the data generating process, priors, etc.

Classical Approaches

A naive approach to this problem is the "min cut" method, where we wish to minimize $w_l = \sum\limits_{\substack{i,j \text{ in different} \\ \text{clusters}}} \gamma_{ij}$, but this often just results in one giant cluster with $n - k + 1$ nodes and $k - 1$ clusters with just one node. Two adjustments to this are common, the ratio cut method and the normalized cut method.

Ratio cut minimizes the quantity $\sum_l^k \frac{w_l}{|C_l|}$ where w_l is defined above as the min cut objective and C_l is cluster l. This penalizes clusters for being too small and aims to make clusters more evenly sized.

Normalized cut minimizes the quantity $\sum_l^k \frac{w_l}{vol(C_l)}$ where $vol(\cdot)$ is the volume function, $\sum_{i,j \in C_l} \gamma_{ij}$. This penalizes clusters that have too few within-group edges.

Both ratio and normalized cut problems are NP-hard, so they are often "solved" through a continuous relaxation. The typical approach is to embed the graph in $\mathbb{R}^p$ and then perform k-means clustering. The embedding that is most commonly used is one that approximates the pairwise expected commute distances as squared Euclidean distances. Since this involves spectral decomposition, this approach is called "spectral clustering".

Bayesian Approach[1]

In this section, we will explore unweighted graphs, i.e., graphs where each edge has a weight of 1. The similarity matrix associated with such a graph is typically called an *affinity matrix*, denoted as A.[2]

Data Model and Priors

A possible data model for graphs might be as follows: Let x_i be the data points(i.e., vertices), and let $z_i = 1, ..., k$ be the labels. Then there exists an edge between x_i and x_j with probability p_{ij}, where p_{ij} is (relatively) small if x_i and x_j belong in different clusters or (relatively) large if x_i and x_j belong to the same cluster.

Let $Q \in [0,1]^{k \times k}$ be the matrix of edge probabilities such that Q_{ij} is the probability that there exists an edge between clusters i and j. Let A be the affinity matrix for the graph. Then our data model is:

$$A_{ij} \mid z_i, z_j, Q \overset{indep}{\sim} Bernoulli(Q_{z_i, z_j})$$

No we need to set priors on z and Q. We can simply say that each z_i is multinomial:

$$z_i \mid \pi \overset{iid}{\sim} Multinomial(\pi)$$

for some $\pi \in [0,1]^k$. Then

$$\pi \sim Dirichlet(\alpha)$$

for some $\alpha \in \mathbb{R}_+^k$, which we will fix.

Since Q is a matrix of probabilities, we can say

$$Q_{rs} \overset{iid}{\sim} Beta(a, b)$$

with the constraint that $Q = Q^\top$ since our graph is undirected. a and b are fixed.

Posteriors and Gibbs Sampling

The main quantity of interest is z, the cluster assignments, with the others being nuisance parameters. We can derive the full posterior for z:

$$\begin{aligned} p(z_i|z_{-i}, A, \pi, Q) &\propto p(A|z, \pi, Q)p(z, \pi, Q) \\ &\propto p(A|z, \pi, Q)p(z|\pi, Q)p(\pi, Q) \\ &\propto p(A|z, Q)p(z|\pi) \end{aligned}$$

Since z is discrete and multinomial, we can set $h(z_i) = p(A|z, Q)p(z|\pi)$ and

[1] I also tried using a Bayesian model-based clustering method on spectral embeddings, but this often failed. My guess is that this is because it's very difficult to come up with a good prior since we have no idea what shape/distribution the embedded points are in.

[2] I was unable to derive posteriors for weighted graphs using something like a gamma or poisson distribution for the elements of the similarity matrix.

$$p(z_i|-) = \frac{h(z_i)}{\sum_j h(z_j)}$$

$p(z|\pi)$ is just the prior for z, which we set as $p(z_i|\pi) \propto \pi_{z_i}$.

Since each edge is drawn independently, we have:

$$p(A|z, Q) \propto \prod_{j \neq i} Q_{z_i,z_j}^{A_{ij}} (1 - Q_{z_i,z_j})^{1-A_{ij}} \times \prod_{k \neq i} Q_{z_k,z_i}^{A_{ki}} (1 - Q_{z_k,z_i})^{1-A_{ki}}$$

It's straightforward to see that:

$$\pi \mid z \sim Dirichlet(\alpha_1 + n_1, ..., \alpha_k + n_k)$$

where $n_l = \sum_i^n I(z_j = l)$.

It's also straightforward to see that:

$$Q_{rs} \mid A, z \sim Beta(a + S_{rs}, b + N_{rs} - S_{rs})$$

where S_{rs} is the $k \times k$ matrix of observed "successes" and N_{rs} is the $k \times k$ matrix of observed "observations". That is, S_{rs} is the number of edges between clusters r and s, while N_{rs} is the total number of possible connections between clusters r and s.[3]

The advantage with the Bayesian approach is that we get cluster assignment probabilities rather than point estimates (e.g., x_1 is in cluster 1 with probability 0.7).

Examples

Numerical issues

We can see that the entries of S and N can quickly blow up as the number of nodes increases. Q will then be drawn as follows:

$$\alpha^{(t+1)} \sim Gamma(a^{(t)}, 1)$$
$$\beta^{(t+1)} \sim Gamma(b^{(t)}, 1)$$
$$Q_{rs}^{(t+1)} = \frac{\alpha^{(t+1)}}{\alpha^{(t+1)} + \beta^{(t+1)}}$$

We can also see that drawing A involves products of many small numbers (less than 1). So $p(A|-)$ will be computed on the log scale.

[3] I actually did not fully derive this but this seems intuitively correct based on what we've seen in class. A reference listed at the end appears to be consistent with what I have here, with different notation.

Data generated from SBM

We will begin by generating data from a stochastic block model. The parameters we will use are:

- $n = 50$, the number of nodes
- $k = 2$, the number of groups
- $\alpha = (1, 1)^\top$, the *a priori* group balance
- $P(\text{edge between clusters}) = 0.05$
- $P(\text{edge within clusters}) = 0.95$
- Number of "prior observations" $= 3$

Note that while we're setting prior probabilities, we're only observing one graph.

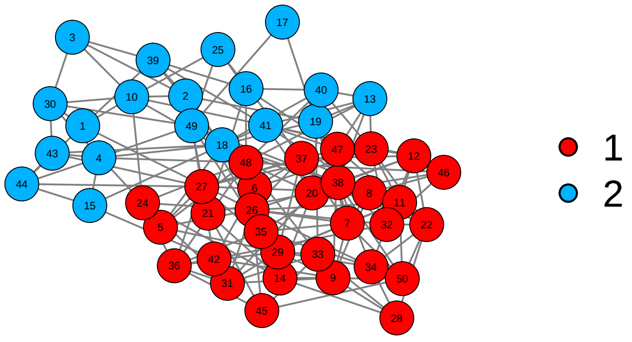

Figure 1: Graph generated from a stochastic block model

To model, we will use these priors:

- $\alpha = (1, 1)^\top$
- $a = b = 1$ for each Q_{rs}

This is an attempt at using an "uninformative" (and not entirely correct) prior for these data.

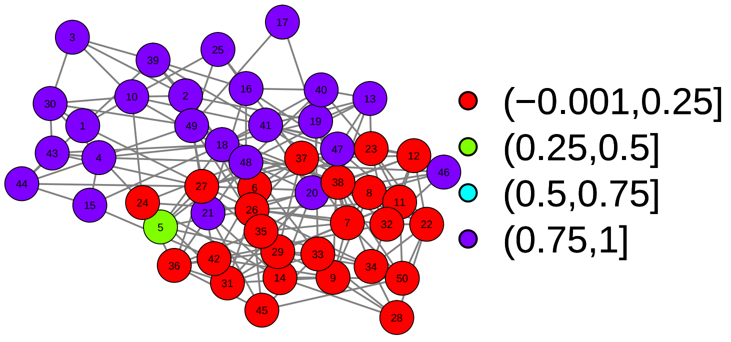

Figure 2: Averaged posterior label probabilities

The colors are based on the posterior label probability for each node, averaged out over the last 800 iterations.

The clusters look consistent with a couple inaccurate labels. As a quick measure of "performance", we take a look at the ROC curve (keeping in mind that flipped labels are just as good):

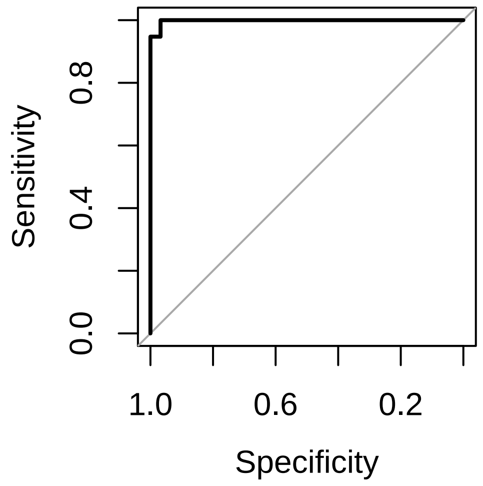

Figure 3: ROC curve between a priori and a posteriori predictions

Next, we can take a look at some trace plots to check if the chain is behaving:

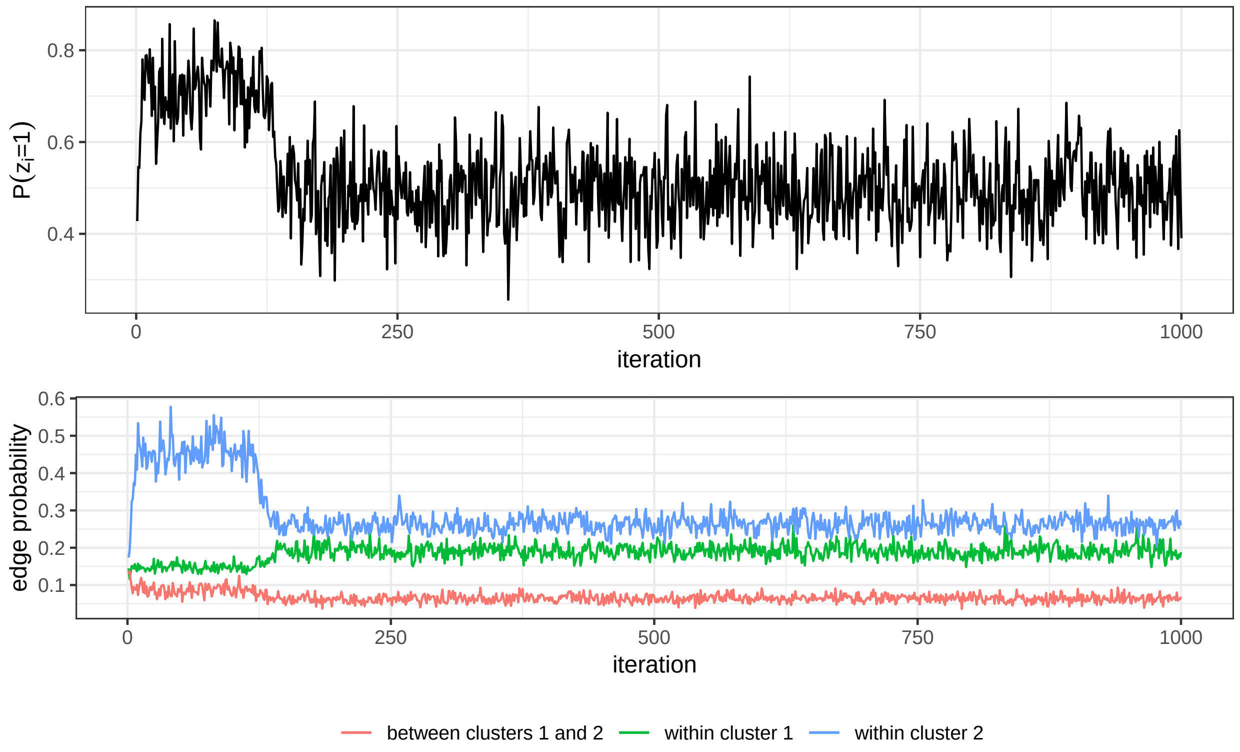

Figure 4: Trace plots

Table 1: Posterior quantiles (after removing the first 200 iterations)

parameter	2.5%	50%	97.5%
$Q_{1,1}$	0.160	0.191	0.226
$Q_{2,2}$	0.225	0.263	0.304
$Q_{1,2}$	0.047	0.064	0.084
π	0.351	0.490	0.637

It appears that after taking some time to find a mode, π and Q stabilized around the mode. The initial cluster assignments were randomly selected, so it is expected that these chains would take a few hundred iterations to stabilize.

Our posterior parameter intervals don't align completely with the priors that were used to generate the data. In particular, the within-group edge probabilities are higher than what we would expect from the prior.

Looking at the cluster assignment probabilities for each node reveals something that may be problematic: Most of these probabilities are very close to 0 or 1.

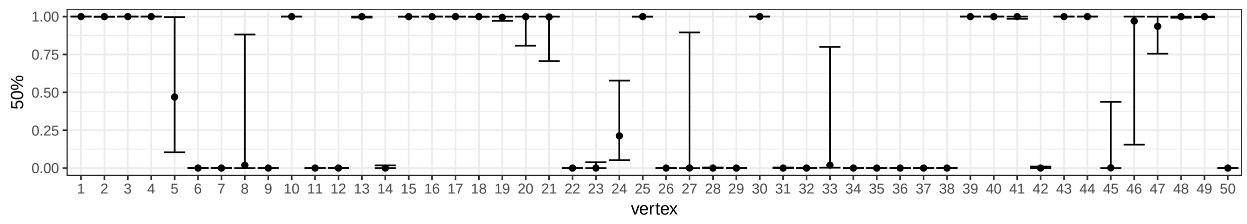

Figure 5: 95% CIs for P(z = 1)

Example: High School Network

This dataset consists of friendships among 70 students from a high school in Illinois. From this graph, we would like to identify whether there are any cliques and to which clique each student belongs.

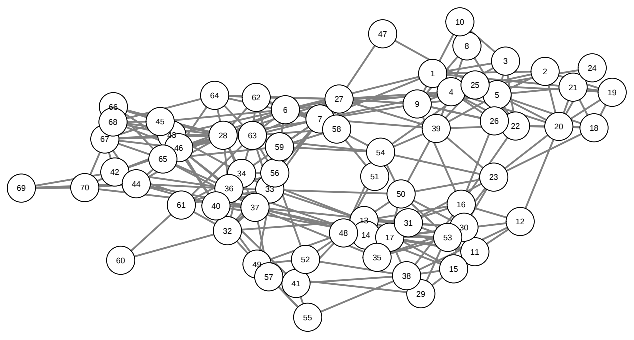

Figure 6: Graph of friendships among 70 high school students

Some reasonable guesses might be:

- There are 3 cliques/clusters/communities.
- Each person has 5 friends.

- Each clique is approximately equally sized.

Then our choice of priors might be:

- $Q_{rs} \sim Beta(1, 14)$, to reflect the probability of one person being a friend with another person
- $\pi \sim Dirichlet(5, 5, 5)$, to reinforce the belief that the cliques are approximately evenly sized

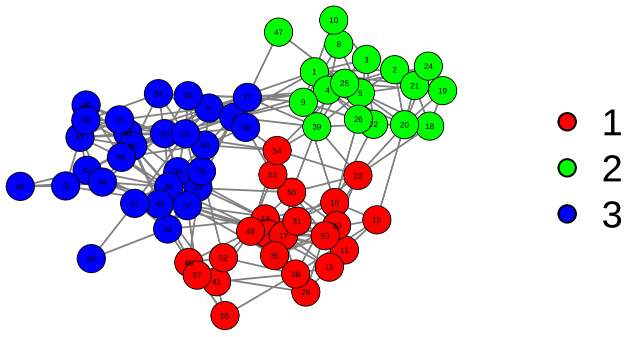

Figure 7: Most frequent cluster label from Gibbs sampler

Visually, the model appears to give us reasonable clusters. We can check some chain diagnostics:

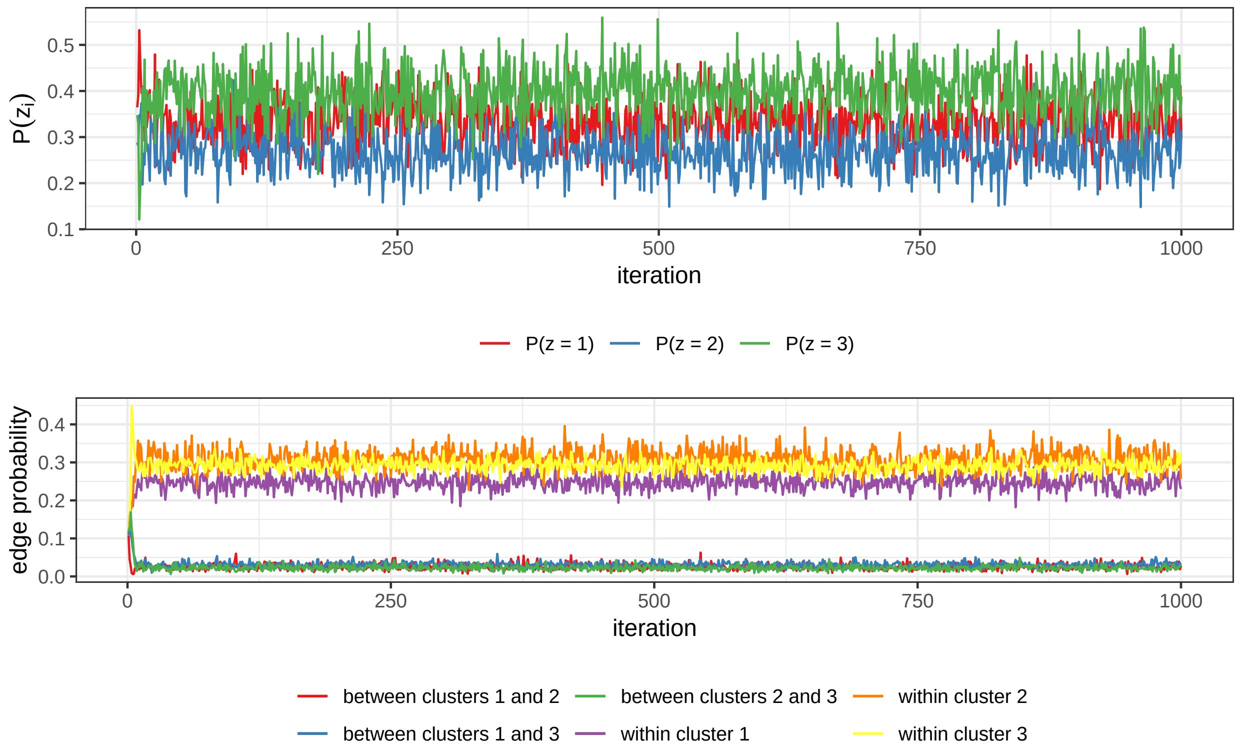

Figure 8: Trace plots

Based on this result, it appears that clique 3 is larger than clique 1, which is larger than clique 2. We can verify with interval estimates:

cluster	2.5%	50%	97.5%
1	0.2365021	0.3317393	0.4342175
2	0.1803268	0.2644461	0.3528001
3	0.3017398	0.4012055	0.5027327

We can also see that if two people are from two different cliques, there is a very low chance that they will be friends, and there is no one clique that reaches across to another clique more often than the others.

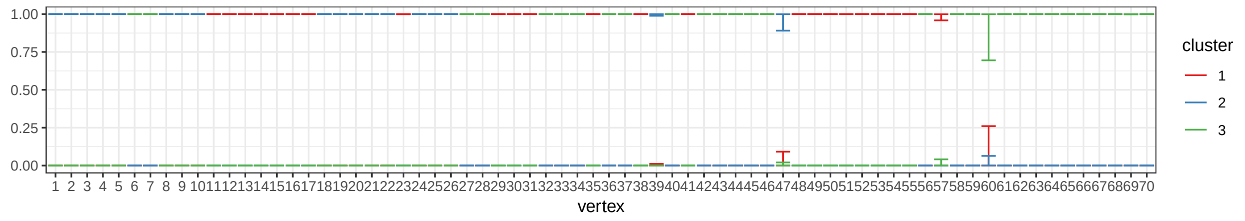

Figure 9: 95% CIs for cluster probabilities

This plot reveals that all but one node spent almost all of its time assigned to one particular cluster. We cannot get much of an idea of how uncertain we are about these cluster assignments (either that, or there is no uncertainty).

We can try comparing this against the classical spectral clustering method. For visualization purposes, we will embed the graph in $d = 2$ dimensions.

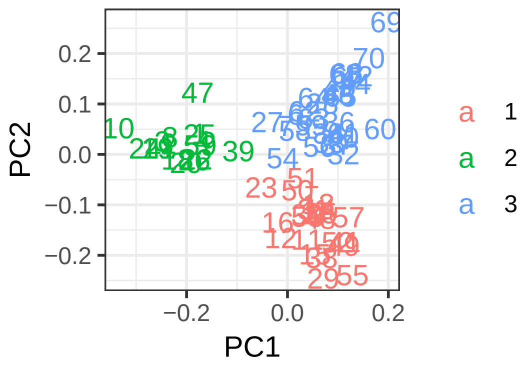

Figure 10: Spectral clustering results in 2 dimensions

```
    sbm
sc    1  2  3
  1  22  0  0
  2   0 18  0
  3   1  0 29
```

The spectral clustering results align very closely with our Gibbs sampler. It's interesting to note that single inconsistency does not correspond to the vertex with uncertainty from the MCMC sample. Instead, the high-uncertainty point in the MCMC sample corresponds to a student with only two friends, both of whom are within the same clique. From additional experimentation, it appears that vertices that are connected to few edges are associated with higher than usual uncertainty, even if they are deeply placed in a cluster.

A Problematic Example

In a previous project, we looked at a graph that consists of two long chains with the two chains linked at two places:

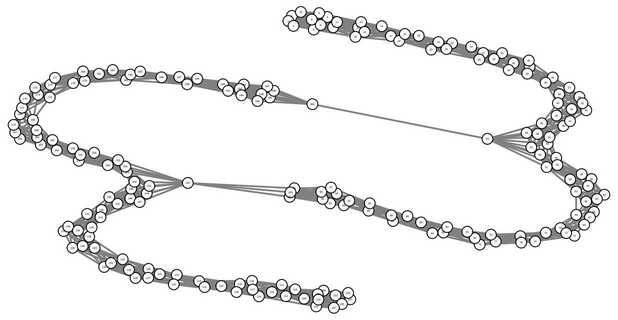

Figure 11: Graph consisting of two chains that are sparsely connected

Since this is a manufactured graph, we know some properties about it:

1. Each vertex is connected to 10 other vertices
2. Each "chain" consists of 100 vertices, with 200 vertices in total

Visually, this appears to be a very simple clustering problem. However, spectral clustering in two dimensions fails:

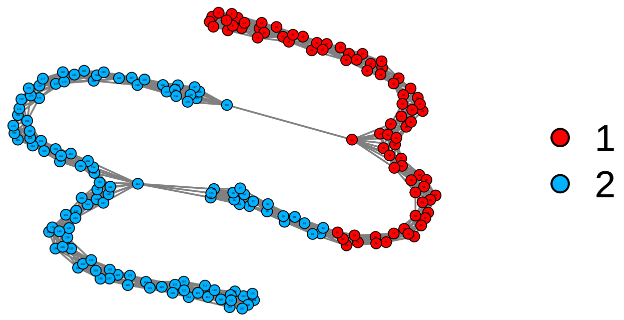

Figure 12: Cluster assignments based on spectral clustering in two dimensions

We can see why if we plot the embedding:

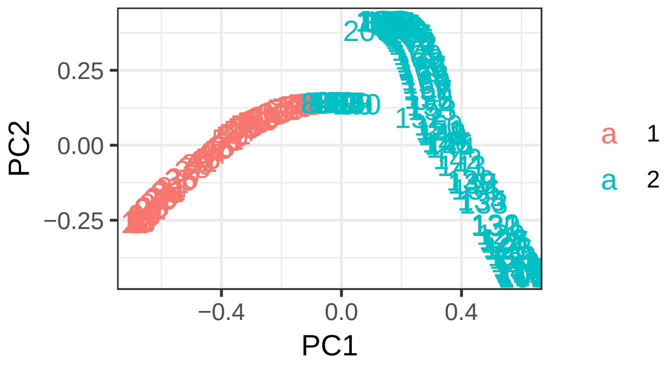

Figure 13: Embedding of the chain graph

Spectral clustering uses k-means as its final clustering step, and k-means makes several inherent assumptions: Each cluster consists of normally distributed points, clusters are evenly distributed, and the within-group variance of each cluster is the same and can be written as λI. This embedding is clearly not Gaussian, so k-means is not appropriate. If we try the full $n-1$-dimensional embedding, k-means will work, but there are many local minima and the algorithm will most likely stop before finding the global solution.[4]

We might hope that our Bayesian method will get around this, but a closer examination of the graph suggests otherwise. The stochastic block model assumes that each vertex within a cluster has equal probability of forming an edge with every other vertex within the same cluster. In this case, there clearly is an underlying pattern behind how the edges are formed, so this prior is not appropriate. Nevertheless, we can try fitting a model.

Here, our priors will be:

- $\alpha = (10, 10)^\top$—since we know that each chain consists of 100 vertices, we want to try to push the model toward evenly sized clusters
- $a = 10$ and $b = 190$, based on how this chain was constructed

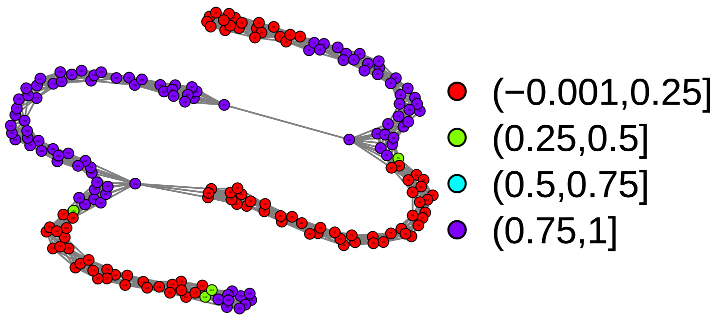

Figure 14: Clusters based on Gibbs sampler

[4]I also tried an "exchange algorithm" which resulted in clusters similar to the SBM clusters. This seems to happen because of the many local optima in this problem.

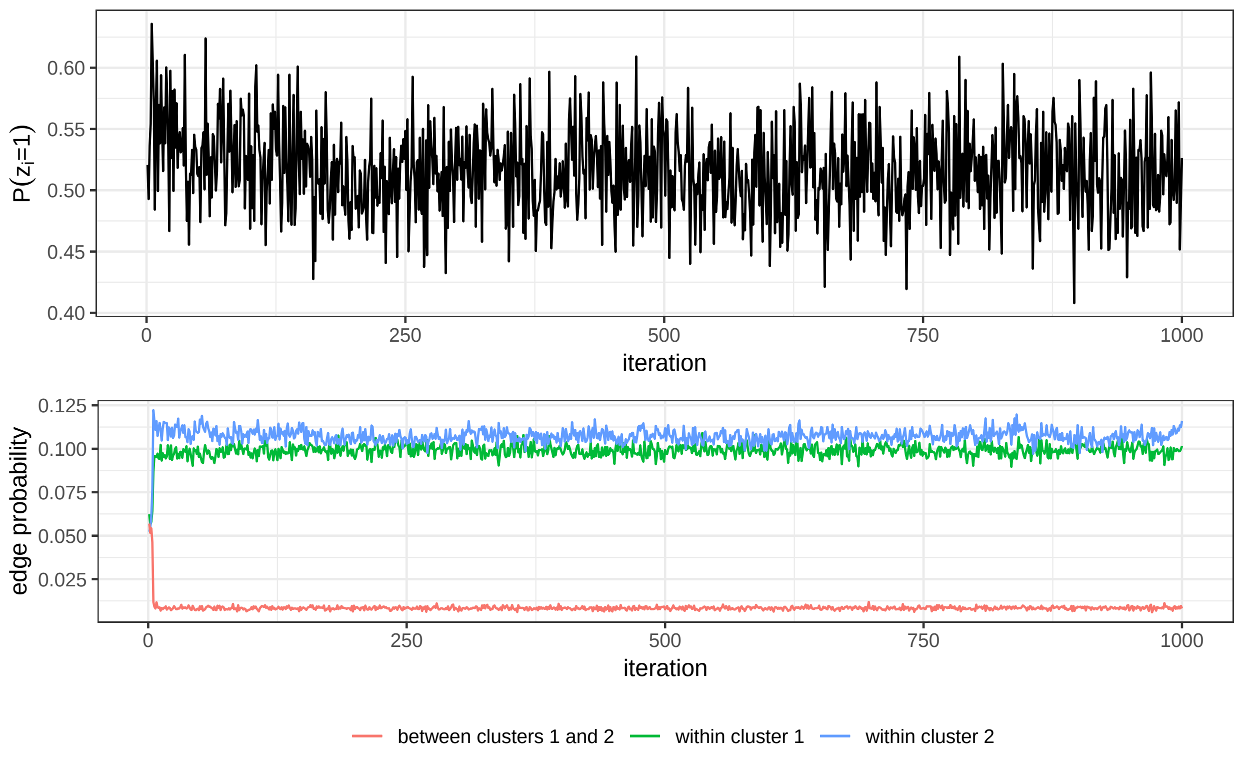

Figure 15: Trace plots

The chain appears stable and the mode that it found appears reasonable given our knowledge of how the graph was constructed, but the resulting cluster labels are not at all what we expected.

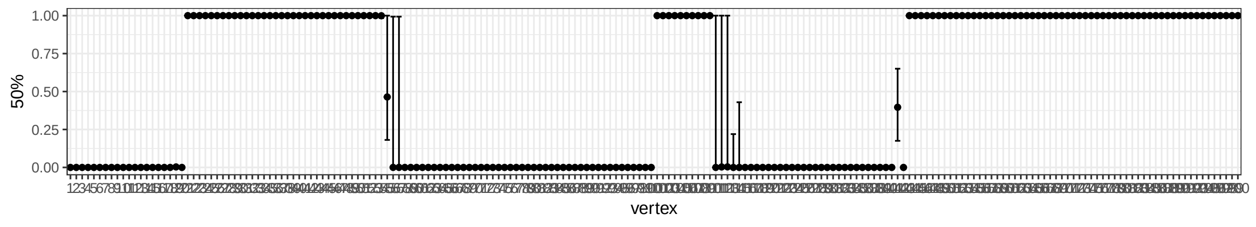

Figure 16: 95% CIs for P(z = 1)

Again, it seems like the labels settled on some configuration and stayed there for the rest of the Markov chain.

We can "fix" this by feeding the MCMC algorithm a good initial guess as to what the cluster assignments should be, but this is not always possible, and if we do this, the cluster assignments will never change and the algorithm will report zero uncertainty about them.

Conclusions and Future Work

In this paper, we implemented a Bayesian stochastic block model for graph partitioning. We derived full conditional posteriors for the parameters, implemented a Gibbs sampler, and showed that the results appear reasonable for simple problems. We also showed a failure case where our priors were incorrect.

We also showed that the Markov chain can have a tendancy to “stick” to a particular cluster configuration. This was especially present in the two examples where the data were not generated using a stochastic block model. When this happens, the labels will stay fixed for the rest of the chain as the probability of a node being a particular label collapses to 1. When this happens, it sometimes corresponds to very low variance for the other parameters. Further investigation is required to see if this is a “real” or if it is due to numerical issues and machine rounding error.

For this investigation, we fixed the number of clusters. In many cases, it is not reasonable to be able to know how many clusters there should be. An extension of this work would be to model the number of clusters (perhaps using a Poisson distribution or a stick-breaking process). We also analyzed unweighted graphs, using a Bernoulli distribution to model whether there is an edge between two vertices. This could be extended by modeling the edge weights using something like a gamma distribution.

References

1. http://networkrepository.com/moreno-highschool.php
2. http://konect.uni-koblenz.de/networks/moreno_highschool
3. https://arxiv.org/pdf/0711.0189.pdf
4. http://pages.iu.edu/~mtrosset/Courses/675/notes.pdf
5. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.9397&rep=rep1&type=pdf
6. https://projecteuclid.org/download/pdfview_1/euclid.ba/1508378465
7. https://ani.stat.fsu.edu/~debdeep/sbmBayes.pdf