Markov Cluster Algorithm

Rachel Cheung, Jinny Cho, Tegan Wilson

1. Introduction

When we think of clusters on graphs, we usually think of closely knit sub-components, similar to cliques. In other words, we would like vertices in the same cluster to have many edges between each other, and for the graph as a whole to have very few edges between clusters. The Markov Clustering Algorithm (MCL) applies the ideas of random walks and Markov chains to find these clusters in graphs. If we follow some number of random walks, we would like the probability that we end in the cluster we started in to be higher than leaving.

2. The Algorithm

2.1. Algorithm

Given: an undirected graph, two parameters, e and r, where $e \in \mathbb{Z}_{\geq 2}$ and $r \in \mathbb{R}_{>1}$.

1. Create the associated link matrix, L, where entry $a_{ij} = 1 \iff$ there is an edge between vertex i and vertex j.
2. Add self loops to the matrix.
3. Normalize the matrix so that each column sums to 1.
4. Expansion step: Raise the matrix to the e^{th} power. This simulates e random walks/Markov chains.
5. Inflation: First raise each column to the r^{th} power. Then normalize the matrix again, so each column sums to 1.

2.2. Details about the algorithm

- The reason for adding self-loops: depending on the structure of the graph, it may be impossible to travel to certain nodes via an odd or even length path. For example, if we had a tree with three vertices and we started from the middle one, we could only travel to the other two by paths of odd length. However, if we add self loops, this eliminates the problem. In short, we add self-loops so that our algorithm is not affected by whether our parameter is odd or even.

- This algorithm does not take in a number of clusters, k, but instead finds it based on the e and r parameters given. And, this algorithm does not necessarily return a partitional clustering. However, if a vertex is attracted by two different vertices, then it must be equally attracted by both clusters. This only occurs when the two clusters are isomorphic to each other.[1]

- The MCL algorithm simulates the flow of random walks using two algebraic operations on matrices, expansion and inflation. Expansion coincides with a normal matrix multiplication. It models the spreading out of flow, being responsible for allowing flow to connect different regions of the graph. Inflation is a power followed by a normalization. It models the contraction of flow, which enables strengthening of strong currents and weakening of already weak currents.

- If the given graph is weighted, the first part of the algorithm is different. a_{ij} = weight instead of a_{ij} = 1 when there is an edge between vertex i and vertex j.

- We normalize the matrix because matrix in this algorithm is a probability matrix so we want each column sums up to 1.

3. Datasets

Our Markov Clustering implementation requires undirected network data. We tested our unweighted implementation on two small animal social network datasets. Both datasets were accessed on the Koblenz Network Collection. [3]

3.1. Grevy's Zebra Network

Our first dataset consisted of 28 nodes, or individual Grevys zebras, and 111 unique undirected edges, which represented at least a single interaction during the course of the study. The data was originally collected and analyzed in a paper by Sundaresan et al., in which they found three nonzero bond network components and 11 network components based on associations with significantly preferred values. [5]

3.2. Dolphin Network

Our other dataset was made up of 62 nodes, or individual bottlenose dolphins, and 159 unique undirected edges, which represented frequent associations between two dolphins. The data was collected for a paper by Lusseau et al., which identified two larger communities and four sub-communities. [6]

4. Assessment Parameters

To determine the effectiveness, we tested our clustering using three methods: modularity, conductance, and coverage. For each metric, we implemented normalized functions found in Emmons et al.s article; each returned a value from 0 to 1, with 1 as the optimum. [7]

4.1. Modularity

Measures the amount of edges found within a cluster compared to the expected number of edges based on random probability, or chance. Modularity is computed as the following:

$$\sum_k (e_{kk} - a_k^2) \tag{1}$$

e_{kk} represents the probability of intra-cluster edges, or edges between vertices within the same cluster, S_k. This is computed, for each cluster, as the number of intra-cluster edges divided by the total number of edges in the graph. a_k represents the probability of an intra-cluster edge or an inter-cluster edge (between clusters) with an start point in S_k. This is computed, for each cluster, as the sum of the number of inter-cluster edges and the number of intra-cluster edges incident to S_k divided by the total number of edges in the graph:

$$e_{kk} = \frac{|\{(i,j) : i \in S_k, j \in S_k, (i,j) \in S_k\}|}{|E|} \qquad a_k = \frac{|\{(i,j) : i \in S_k, (i,j) \in E\}|}{|E|} \tag{2}$$

A modularity score closer to 1 means that the clusters are dense relative to the connectivity between clusters. Modularity also has a resolution preference, tending to prefer smaller clusters and discriminating against larger ones.

4.2. Conductance

Graph conductance is a measure of the connectedness of a graph. For each cluster, it describes the number of inter-cluster edges for a given cluster (A_{ij}) divided by the minimum of either: the number of edges with endpoints in the cluster ($A(S_k)$), or the number of edges that do not have endpoints in the cluster ($A(\overline{S_k})$). The conductance of a graph G is the sum of cluster conductances divided by the number of clusters subtracted from 1:

$$\phi(S_k) = \frac{\sum\limits_{i \in S_k, j \notin S_k} A_{ij}}{min\{A(S_k), A(\overline{S_k})\}} \qquad\qquad \phi(G) = 1 - \frac{1}{k}\sum_k \phi(S_k) \tag{3}$$

As noted by Emmons et al., this version of conductance emphasizes inter-cluster connectivity at the cost of ignoring intra-cluster connectivity. A value closer to 1 describes a more well-connected graph, while a lower connectivity is indicative of poor connectivity between clusters.

4.3. Coverage

Our final assessment metric, coverage, measures intra-cluster density by dividing the number of intra-cluster edges in the graph by the total number of edges in the graph:

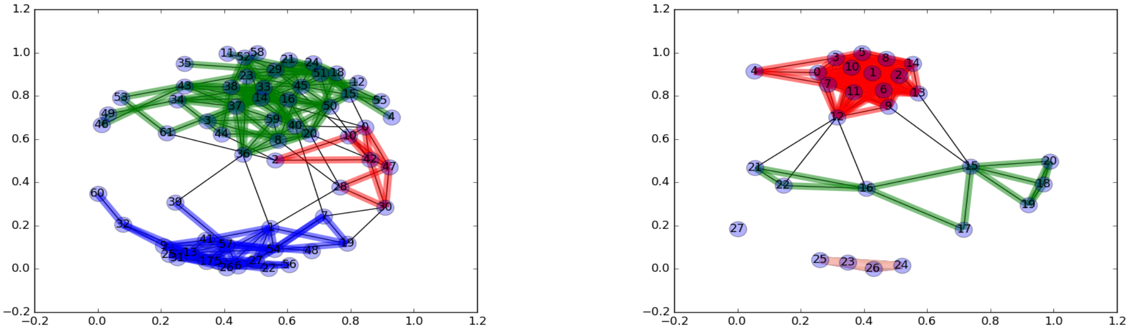
$$\frac{\sum\limits_{i,j} A_{i,j}\delta(S_i, S_j)}{\sum\limits_{i,j} A_{i,j}} \tag{4}$$

In the above equation, $\delta(S_i, S_j) = 1$ if $S_i = S_j$, or if the two points are in the same cluster, and 0 otherwise. A coverage score of 1 means nodes are highly connected within clusters, while a low coverage score indicates poorly connected clusters. Coverage alone fails to address inter-cluster connectivity and can be maximized by creating a singular cluster rather than more accurate clustering distributions.

5. Results

The algorithm we implemented can be utilized on both unweighted and weighted graphs. It takes in node-edge data, and returns a visualization of the network (with color-coded edges corresponding to clusters), the number of clusters, and values for the three assessment metrics. We ran the algorithm on our two datasets with various expansion and inflation parameters and assessed our models through visualization and a weighted average of assessment metrics (weighting modularity twice as much as conductance and coverage).

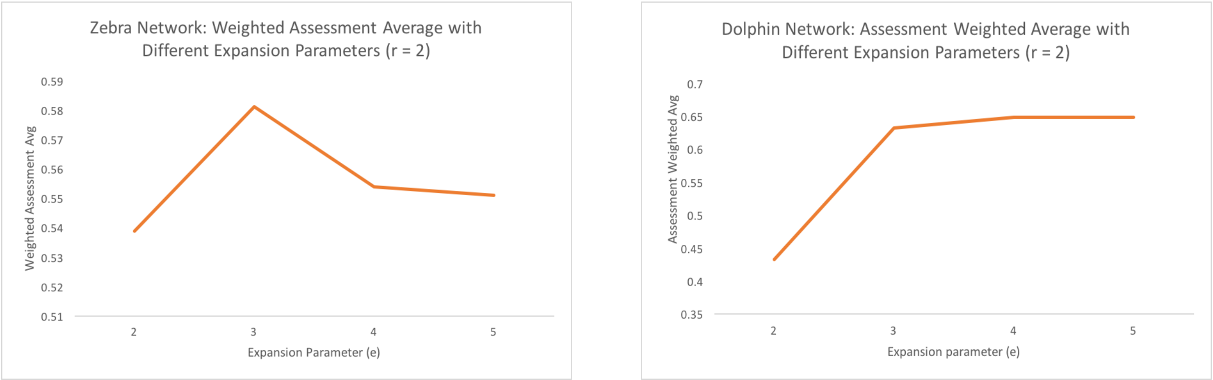
For both datasets, we found clusters that both had clear visualizations and high weighted assessment average values with inflation parameter $r = 2$. For both dolphins and zebras, an expansion parameter $e = 3$ and inflation parameter $r = 2$ yielded 3 distinct clusters (Figure 1). For zebras, $e = 3$ and $r = 2$ yielded the highest weighted assessment average of 0.581 (Figure 2), while for dolphins $e = 4$ and $r = 2$ yielded two clusters and the highest weighted assessment average of 0.648 (only marginally higher than the 0.632 with parameters $e = 3$ and $r = 2$, Figure 2). Both the three zebra clusters and the two dolphin clusters seemed approximately similar to the non-zero communities found in the Sundaresan et al. and the larger communities found in Lusseau et al. papers, respectively.



(a) Clustering of dolphins, $e = 3$ and $r = 2$

(b) Clustering of zebras, $e = 3$ and $r = 2$

Figure 1: Clusterings of different animal datasets



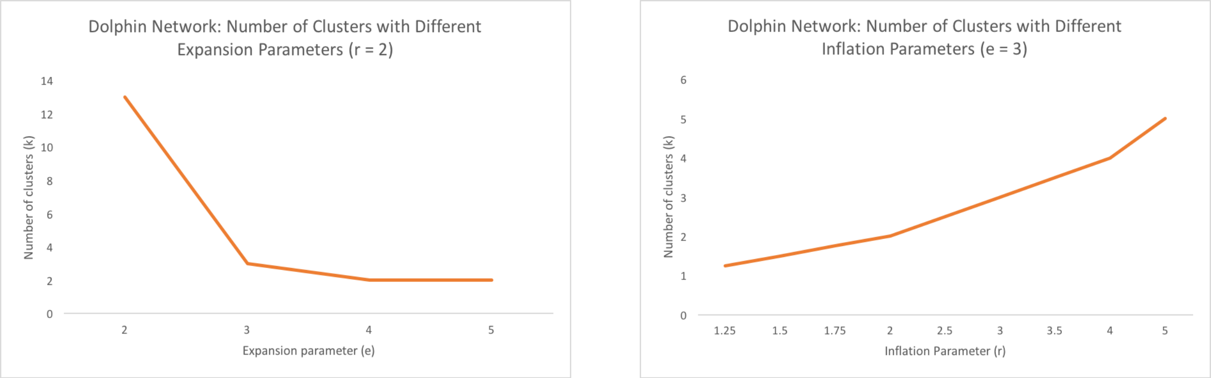
(a) Weighted assessment average with varying e

(b) Weighted assessment average with varying e

Figure 2: Tuning parameters

We found that increasing the expansion parameter decreased the number of clusters found, while increasing the inflation parameter led to more cluster granularity (more clusters of smaller sizes). (Figure 3) For example, keeping the inflation parameter r at 2, but increasing e to 5 yielded 2 zebra clusters and 2 dolphin clusters. However, fixing r at 2 and decreasing e to 2 produced 4 zebra clusters and 13 dolphin clusters. Fixing e at 3 and decreasing r to 1.5 yielded 2 zebra clusters and 2 dolphin clusters, while increasing r to 3.5 yielded 4 zebra clusters and 8 dolphin clusters. This makes some intuitive sense: Increasing expansion increases the number of random walks per step, which then spreads out the "flow" from strong attractor vertices. This in turn creates larger (and therefore fewer) clusters. Increasing inflation strengthens influence from nearby vertices, which produces smaller (but more numerous) clusters.

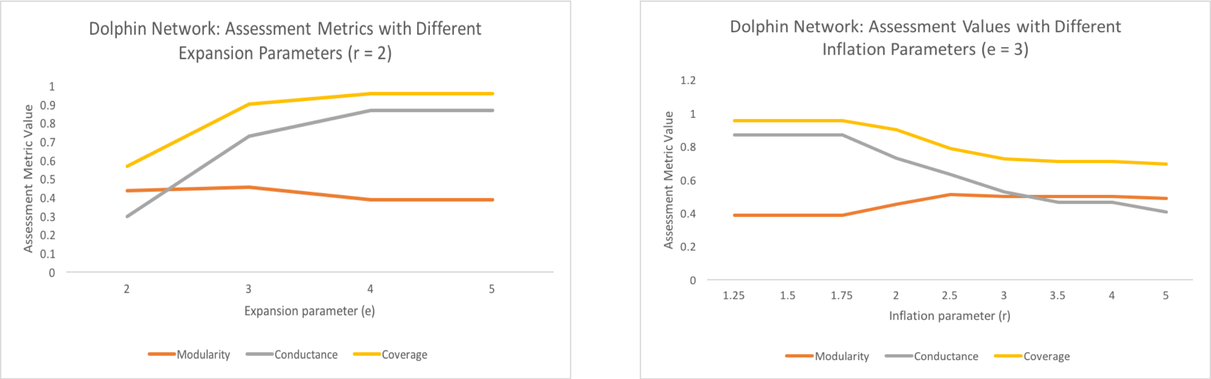
We also plotted the three assessment values against different expansion and inflation parameters. (Figure 4) For our two cases, it appears that as the expansion parameter increases, conductance and convergence have general upward trends (with some fluctuation), while modularity has a general downward trend. Conductance and coverage tend to increase to 1 as expansion increases, since both inter- and intra-cluster connectivity



(a) Number of clusters with varying e

(b) Number of clusters with varying r

Figure 3: Number of clusters with varying e and r



(a) Different assessment measures with varying e

(b) Different assessment measures with varying r

Figure 4: Modularity, conductance, and coverage with varying e and r

increase with increased random walks per step. Modularity decreases as cluster become less dense. The opposite occurs as inflation increases. Conductance and coverage decrease as the clusters become more granular and more spread out, while modularity increases as clusters become smaller and denser.

6. Future Extensions

We would like to continue exploring weighted graph applications and their associated assessments. The weighted animal datasets we had access to included kangaroos and cows, but both had very small sample sizes and produced tight-knit singular communities. Since MCL can be applied to a wide range of network data to find communities, we would like to explore the runtime and how it scales with other, larger datasets. Another interesting analysis could be done by comparing accuracy and efficiency performances of MCL and other network clustering algorithms. Finally, since MCL can only be used on undirected graphs, we would like to explore network clustering algorithms that work for directed graphs.

References

[1] Official Markov Clustering website: `https://micans.org/mcl/`

[2] Presentation slides on Markov Clustering: `https://www.cs.ucsb.edu/~xyan/classes/CS595D-2009winter/MCL_Presentation2.pdf`

[3] Dataset source: `http://konect.uni-koblenz.de/`

[4] Grevy's zebra paper: Sundaresan, Siva R., Ilya R. Fischhoff, Jonathan Dushoff, and Daniel I. Rubenstein. "Network metrics reveal differences in social organization between two fission-fusion species, Grevy's zebra and onager." Oecologia 151.1 (2007): 140-149.

[5] Dolphin paper: Lusseau, David and M.E.J. Newman. "Identifying the role that individual animals play in their social network." Proc R Soc London B (Suppl) 271 (2004): S477-S481.

[6] Assessment metrics paper: Emmons, Scott, Stephan Kobourov, Mike Gallant, and Katy Brner. "Analysis of network clustering algorithms and cluster quality metrics at scale." PLoS One 11.7 (2016).