Variance of clusterings on graphs

by

Thomas V. Mulc

In partial fulfillment of the requirements for the degree of

Bachelor of Science in Mathematics

at the

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

May 2016

Advisor: John McSweeney, Ph.D.

1 Introduction

Graphs that represent data often have structures or characteristics that can represent some relationships in the data. One of these structures is *clusters* or *community structures*. Clusters can be determined algorithmically, and most clustering algorithms for graphs are deterministic, which means they will output the same clustering each time. We investigate a few stochastic algorithms and look into the consistency of their clusterings. Lastly, we address issues with using the "true labeling" of the vertices of a graph as metric for clustering. To test the consistencies of the stochastic methods, we generated clusterings on three graphs: the Karate graph [14], an artificial dumbbell graph, the 2014 for NCAA Division 1 Football graph[9]. We use R and the iGraph package for all the computations, visualizations, and realizations of algorithms. The Football graph was generated using the data from [9].

2 Test graphs

The Karate (a.k.a "Zachary") graph came about after a karate club had two members, each of which had different views on the prices of lessons. This led to fission in the club, and members would politically associate themselves with one of the leaders so that during club votes, it was as if individuals were voting on behave of their leader/party. A group led by Wayne Zachary studied the club members and took note when members we seen together outside of karate club events; when this occurred, the observers noted friendships between pairs of individuals. A graph was created to represent the social dynamics of the club. The nodes represent people, and the edges represent friendships. There is clear community structure because those who were close to either leader tended to have similar ideological views and wouldn't socialize with people of opposing views.

The Dumbbell graph was generated in R using the iGraph package. First, two Erdős-Rényi random graphs were generated with parameters ($p = .99, |N| = 10$). Then, the graphs were connected by adding one edge from the first random graph to the second; the result was our Dumbbell graph, shown in Figure 1.

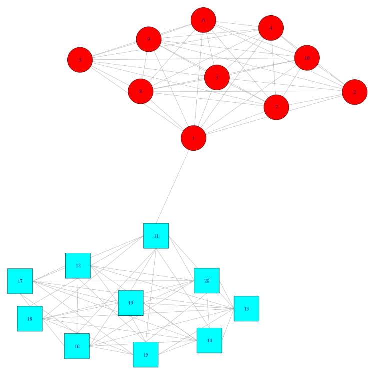


Figure 1: Artificial dumbbell graph with clear community structure.

This graph was generated to have two communities of equal size with high densities of edges within communities and low densities outside of communities.

The Football graph was generated by letting nodes represent teams and edges represent games played. Teams rarely play games outside their conference, and thus we would expect that an analysis on the Football graph would show clusters of teams, with the clusters representing conferences.

2.1 Introduction to "true labeling"

There is an inherent "true labeling" of the vertices of the Football graph, by letting each vertex be labeled by its respective conference. A clustering algorithm may output a set of labels that disagrees with the conference labels. The true labeling might represent what the league wanted as far as who plays whom during the season, but a clustering algorithm might unveil the reality of the scheduling. Determining which is better–the intended labels or an algorithm's output–is subjective, but it's important to recognize that an algorithm may be uncovering a more logical partition. On the other hand, it's equally as important to be wary of the output of an algorithm, because the algorithm may have a flaw or may have made assumptions that are unrealistic in the scenario.

3 Definitions

An undirected graph $G = (V, E)$ is defined by a set of nodes or vertices V and a set of edges $E \subseteq V \times V$. The size of the graph can be described used the number of nodes or number of edges

$$\mid V \mid = n \text{ and } \mid E \mid = m.$$

The adjacency matrix A of G is an $n \times n$ matrix where each element is defined by

$$A_{ij} = \begin{cases} 1 & e_{ij} \in E \\ 0 & e_{ij} \notin E \end{cases}$$

where e_{ij} is an edge from node i to node j.

The number of vertices incident to $i \in V$ is the degree of i defined by

$$deg(i) = \sum_{j \in V} A_{ij}.$$

A partition P of the vertices of G is defined by

$$P = \{C_1, C_2, ..., C_k\},$$

where

$$C_1 \cup C_2 \cup ... \cup C_k = V \text{ and } C_i \cap C_j = \emptyset \quad \forall i \neq j.$$

We define $\mathcal{P}$ as the set of all partitions of V.

4 Clustering

Clustering is inherently a subjective way to organize data. Mathematically, there is a general agreed upon definition of a clustering, but no agreed upon definition of what constitutes a "good" clustering. In the context of graphs, we define a clustering P on a graph a a set of groupings of the nodes. Thus, a clustering on P on a graph G is a partition of V. In general, a good clustering has **many edges within each cluster** and **few edges between clusters**.

4.1 Quantifying clustering

Although good clustering is subjective, its useful to compare different clusterings. This is done be quantify the "goodness" of a clustering.

4.1.1 Cut size

An intuitive metric is the *cut size* [4], R, defined by

$$R = \frac{1}{2} \sum_{\substack{i,j \text{ in} \\ \text{different} \\ \text{clusters}}} A_{ij}.$$

Thus, the *cut size* is the total number of edges between clusters. If two different clusterings were generated on the same graph, the better clustering would be the one with with the smaller *cut size*. The major pitfall with *cut size* is that the optimal score occurs when all nodes are in the same cluster. There are many other methods for quantifying the goodness, but perhaps the most agreed upon metric for clusterings on graphs is Modularity.

4.1.2 Modularity

The modularity score, Q can be thought of as the fraction of edges that fall within either group minus the expected number of edges from an Erdős-Rényi random graph with expected number of edges equal to m, the existing number of edges. It is calculated by

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{deg(i)deg(j)}{2m} \right] I(i,j),$$

where

$$I(i,j) = \begin{cases} 1 & i \text{ and } j \text{ in same cluster} \\ 0 & \text{else} \end{cases}$$

The modularity can be negative, and $Q \in [-\frac{1}{2}, 1]$. Typically, modularity scores of 0.3 and higher imply considerable community structure [6].

5 Clustering algorithms

5.1 SC method

Steinhauser and Chawla developed an algorithm we call SC for generating clusterings on graphs [8]. The general idea of SC is to create clusterings by taking random walks at the nodes, and later examine the walks to determine the clustering. SC can be broken up into two algorithms that operate in serial: Algorithm 1 and Algorithm 2.

In Algorithm 1, t random walks from each node are taken and form a similarity matrix S where $S[i][j]$ is the number of times node j was part of the t walks that started at node i. During each step of the walk, a neighbor is chosen uniforly at random, so that the probability of going from node i to node k is

$$Pr(i \to k) = \frac{A_{ik}}{\deg(i)}.$$

.

Algorithm 1 Community detection with random walks

```
Input: $t$, the length of the random walks
 1: **for** all nodes $i = 1, ..., n$ and $j = 1, ..., n$ **do**
 2:     $S[i][j] = 0$
 3: **end for**
 4: **for** each node $start\_node = 1, ..., n$ **do**
 5:     $i = start\_node$
 6:     $C = \{start\_node\}$
 7:     **for** number of steps $h = 1, ..., t$ **do**
 8:         randomly select $next\_node$ from $neighbors(i)$
 9:         $C = C \cup \{next\_node\}$
10:         $i = next\_node$
11:     **end for**
12:     **for** each node $i \in C$ **do**
13:         **for** each node $j \in C, i \neq j$ **do**
14:             $S[i][j] += 1$
15:         **end for**
16:     **end for**
17: **end for**
```

In Algorithm 2 the similarity matrix is used for agglomerative clustering. When attempting to replicate the algorithms from [8] we found some inconsistencies in the stopping criteria, so the algorithms presented here are our best interpretation of how the SC is supposed to behave algorithmically. Algorithm 2 starts will all nodes in different clusters. Then $[i][j] = \operatorname{argmax} S$ is the most similar pair of clusters and can be interpreted as cluster j visited cluster i's walks the most among all cluster pairs. So we merge cluster i and cluster j so that all the nodes in clusters i and j are in the same cluster. We preserve the information of the walks by average the ith and jth columns and rows and then assign the average to the newly merged cluster. This is done until there are no more clusters left to merge (which can happen if there are non-communicating classes in S).

Algorithm 2 Merging clusters into a consensus clustering representing the community structure of the network

```
 1: Let $C = \{1, 2, ..., n\}$ be the set of communities
 2: while $S[i][j] > 0$ do
 3:     $(max_i, max_j) = \operatorname{argmax} S[i][j]$
 4:     $S[max_i][max_j] = 0$
 5:     $S[max_j][max_i] = 0$
 6:     for each cluster $m = 1...dim(S)$ do
 7:         $S[max_i][m] = \text{avg}(S[max_i][m], S[max_j][m])$
 8:         $S[m][max_i] = \text{avg}(S[m][max_i], S[m][max_j])$
 9:     end for
10:     Remove column and row $max_j$ from $S$
11:     $C \cdot max_i = C \cdot max_i \cup C \cdot max_j$
12:     Remove cluster $C \cdot max_j$ from $C$
13: end while
14: return set of communities $C$
```

5.2 SC-Mod

From SC, we differentiated a new method called SC-Mod, that used modularity as its criterion for the final clustering. SC-Mod uses Algorithm 1 from SC and most of Algorithm 2 from SC but changes which clustering is outputted as the final clustering. Instead of outputting the clustering left when no more merges are possible or when the number of specified clusters had been formed, SC-Mod outputs whatever cluster has the highest modularity from all the clusterings formed during the merging algorithm. So for SC-Mod is first implements Algorithm 1 then implements the modified version of Algorithm 2

5.3 Other stochastic methods

Most other clustering algorithms are deterministic, but we found *Spinglass* [12] and *Label Propagation* [11] to be two stochastic algorithms available in the iGraph package. We won't go into the details of these algorithms, but it's important to know that their results are random. We used these two methods to compare consistency of random clustering algorithms.

5.4 Walktrap

There are many deterministic methods for graph clustering, but Walktrap is most relatable to SC. Walktrap [2] was inspired by the idea of random walks, but unlike SC, it does not actually take random walks, and is a deterministic method.

Walktrap runs in time $\mathcal{O}(mn^2)$ and space $\mathcal{O}(n^2)$, but can normally run in time $\mathcal{O}(n^2 \log n)$ and space $\mathcal{O}(n^2)$.
The general idea of this approach is to compute the transition probabilities of random walks starting from various nodes in order to determine similarity, but without actually simulating any random walks.

A transition matrix M is created where $M_{i,j}$ is the probability of going from node i to node j:

$$M_{ij} = \frac{A_{ij}}{deg(i)}.$$

A distance function r is defined by

$$r_{ij} = \sqrt{\sum_{k=1}^{n} (M_{ik} - M_{jk})^2 \frac{1}{deg(k)}}.$$

The actual clustering algorithm is agglomerative. The initial partition is $P_1 = \{\{v\}, v \in V\}$ so that each vertex is its own cluster. At each step s:

- Find $C_1, C_2 \in P_s$ s.t. they meet a minimum distance criterion

- Set $C_3 = C_1 \cup C_2$ and $P_{s+1} = (P_s \setminus \{C_1, C_2\}) \cup \{C_3\}$
- update distances between adjacent communities.

After $n-1$ steps, $P_n = \{V\}$. The final clustering of G is decided by picking one partition P_s that meets some criterion.

6 Consistency of clusterings

Since the SC algorithm is stochastic by nature, we were interested in finding the consistency of the clusterings. We want to describe the spread of the clusterings and want a partition equivalent to a variance. Describing the variance for a set of clusterings is not as straight forward as the variance of scalars because there is no additive structure for partitions. So before variance is quantified, we first quantify the similarity of two clusterings.

6.1 Difference between two clusterings

A clustering of G is a partition on V. We describe the difference between two clusterings by finding the difference between two partitions. We use a function called the Rand Index to quantify this difference [10].

6.1.1 Rand Index

Let $v_1, v_2 \in V$ and $P_i \in \mathcal{P}$, where $\mathcal{P}$ is the set of all partitions of V. We say that v_1 and v_2 *agree* in P_i if $\{v_1, v_2\} \subset C$ for some $C \in P_i$. We say v_1 and v_2 *disagree* if they don't *agree*. The Rand Index d of partitions P_i and P_j is defined by

$$d(P_i, P_j) = \frac{a+b}{\binom{n}{2}}$$

where

- $a \equiv$ number of node-pairs that agree in P_i and disagree in P_j
- $b \equiv$ number of node-pairs that disagree in P_i and agree in P_j

We have

$$0 \leq \frac{a+b}{\binom{n}{2}} \leq 1$$

so that a value of 0 means that two clusterings are identical and a value of 1 means that every possible pair of nodes disagrees. The Rand Index can be thought of as the fraction of node-pairs that disagree. This notion is nice because it allows us to make comparisons of the same clustering algorithm on different graphs. Additionally, the Rand Index works well for our application because it allows us to compare clusterings that don't have the same number of clusters. Finally, the Rand Index is mathematically nice because it's a metric on $\mathcal{P}_k$.

Claim: the Rand Index is a metric on $\mathcal{P}$

Proof. Let $P_i, P_j, P_h \in \mathcal{P}$.
Separation:
$d(P_i, P_j) = \frac{a+b}{\binom{n}{2}}$ where $a, b, \binom{n}{2} \geq 0$. Thus $d(P_i, P_j) \geq 0$.

Symmetry:
$d(P_i, P_j) = \frac{a+b}{\binom{n}{2}}$ and $d(P_j, P_i) = \frac{a^*+b^*}{\binom{n}{2}}$. Note that a is the number of nodes that agree in P_i and disagree in P_j, while b^* is the number of nodes the disagree in P_j and agree in P_i. Thus $a = b^*$. Similarly for b and a^* we have $b = a^*$. Hence, $d(P_i, P_j) = \frac{b^*+a^*}{\binom{n}{2}} = d(P_j, P_i)$.

Identity of Indiscernibles:
If $d(P_i, P_j) = 0$, then $\frac{a+b}{\binom{n}{2}} = 0$. Since $n > 0$ then $a + b = 0$. Since $a, b > 0$ then every pair of nodes agrees. Thus, $P_i = P_j$.

If $P_i = P_j$ then $a = b = 0$. Thus $d(P_i, P_j) = 0$.

Triangle Inequality: by contradiction
Assume $d(P_i, P_h) > d(P_i, P_j) + d(P_j, P_h)$.

Let $\gamma : V \times V \times \{1, ..., |\mathcal{P}|\} \times \{1, ..., |\mathcal{P}|\} \to \{0, 1\}$ be a function:

$$\gamma(v_1, v_2)_{i,j} = \begin{cases} 1 & v_1 \text{ and } v_2 \text{ agree in } P_i \text{ and disagree in } P_j \\ 1 & v_1 \text{ and } v_2 \text{ disagree in } P_i \text{ and agree in } P_j \\ 0 & \text{else} \end{cases}$$

Then for $a, b \in V$,

$$\sum_{a<b} \gamma(a,b)_{i,h} > \sum_{a<b} \gamma(a,b)_{i,j} + \sum_{a<b} \gamma(a,b)_{j,h}.$$

Thus, $\exists (o,p) \in (V \times V)$ such that $\gamma(o,p)_{i,h} = 1$ and $\gamma(o,p)_{i,j} = \gamma(o,p)_{j,h} = 0$.

Case 1: o and p disagree in P_i and agree in P_h.
Then for P_i and P_j, o and p either

1. agree in i and agree in j $\Rightarrow$ Contradiction because can't agree in P_i
2. disagree i and disagree j.

Then for P_j and P_h, o and p either

1. agree in P_j and agree in P_h
2. disagree in P_j and disagree in P_h. $\Rightarrow$ Contradiction because can't agree in P_h

Thus, it must be that o and p disagree in P_i, disagree in P_j, agree in P_j, and agree in P_h . This is a contradiction.

Case 2: o and p agree in P_i and disagree in P_h.
Then for P_i and P_j, o and p either

1. agree in P_i and agree in P_j
2. disagree in P_i and disagree in P_j. $\Rightarrow$ Contradiction because must agree in P_i

Then for P_j and P_h, o and p either

1. agree in P_j and agree in P_h $\Rightarrow$ Contradiction because must be disagree in P_h
2. disagree in P_j and disagree in P_h.

Thus, it must be that o and p agree in P_i, agree in P_j, disagree in P_j, and disagree in P_h. This is a contradiction.

Since all cases resulted in a contradiction, our assumption is false.

$$\therefore d(P_i, P_h) \leq d(P_i, P_j) + d(P_j, P_h)$$

$$\therefore \text{ The Rand Index is a metric on } \mathcal{P}.$$

$\square$

6.1.2 Variance of clusterings

We define the variance similarly to how we define the variance of a set of scalars. For some set of scalars, $X = \{x_i : i \in \{1, ..., N\}\}$, we define our sample variance as

$$\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

where $\bar{x}$ is the sample mean. Some may note that this is actually a biased estimator of the true variance, but we will continue to use this as our definition of sample variance for a set of scalars. This form can't be used to compute the variance of partitions, since there's no notion of a mean for a set of partitions. The sample variance can be written pairwise as

$$\frac{1}{N^2}\sum_{i<j}(x_i - x_j)^2.$$

Claim: $\frac{1}{N}\sum\limits_{i}(x_i - \bar{x})^2 = \frac{1}{N^2}\sum\limits_{i<j}(x_i - x_j)^2$

Proof.

$$\begin{aligned}
2N\sum_i (x_i-\bar{x})^2 &= 2\sum_j\sum_i (x_i-\bar{x})^2 \\
&= \sum_j\sum_i (x_i-\bar{x})^2 + \sum_i\sum_j (x_j-\bar{x})^2 - 2\sum_i\sum_j (x_i-\bar{x})(x_j-\bar{x}) \\
&= \sum_i\sum_j [(x_i-\bar{x})-(x_j-\bar{x})]^2 \\
&= \sum_i\sum_j (x_i-x_j)^2 \\
&= 2\sum_{i<j} (x_i-x_j)^2
\end{aligned}$$

$$\therefore \frac{1}{N}\sum_i (x_i-\bar{x})^2 = \frac{1}{N^2}\sum_{i<j}(x_i-x_j)^2$$

□

This form is convenient because it doesn't involve a mean as this would *require an additive structure*. The term $(x_i - x_j)^2$ can be interpreted as some sort of distance between two elements from X. We can use this notion to quantify the spread of a set of partitions, $\theta = \{P_1, P_2, ..., P_N\}$. The variance of θ is then defined by

$$\sigma_\theta^2 = \frac{1}{N^2}\sum_{i<j} d(P_i, P_j).$$

Note that σ_θ^2 is somewhat of an average number of pairs that disagree across all clusterings. The actual average number of pairs that disagrees is

$$\frac{1}{\binom{N}{2}}\sum_{i<j} d(P_i, P_j).$$

We use σ^2 as our working definition of variance for clusterings.

6.1.3 Run time

Let $\theta = \{P_1, P_2, ..., P_N\}$, where P_i is a partition of V. To calculate the variance of θ, the Rand Index must be calculated for every pair of partitions.

Calculating the Rand Index for a single partition pair takes $\mathcal{O}(\binom{n}{2})$ because there are $\binom{n}{2}$ pairs of nodes that are compared. Thus, the total run time is

$$\mathcal{O}\Big(\binom{N}{2}\binom{n}{2}\Big) = \mathcal{O}\big((Nn)^2\big).$$

7 Methods and results

To test clustering algorithms using the variance, we simulated SC, SC-Mod, Spinglass, and Label Propagation on the following graphs: Karate, Dumbbell, and 2014 NCAA Football. we ran 100 simulations of each algorithm ($|\theta| = 100$) on each graph. For SC all random walks had a length of $t = 4$ since that is length originally used by [8].

SC was returning a cluster where all the nodes were in the same cluster for every simulation on the Football graph. Thus, we forced SC to output the clustering where $|P| = 11$ since there were 11 conferences in 2014. Figure 2 shows one of these clusterings.

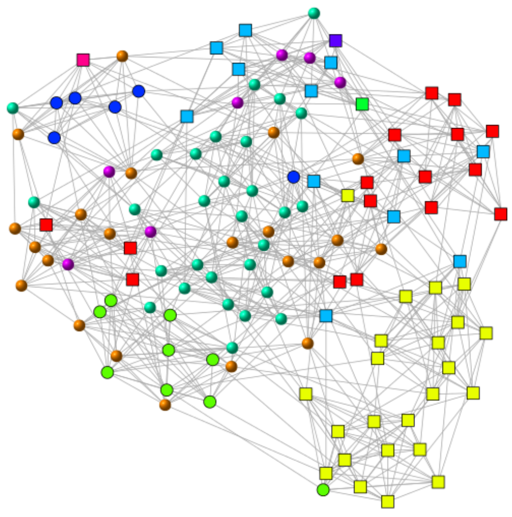
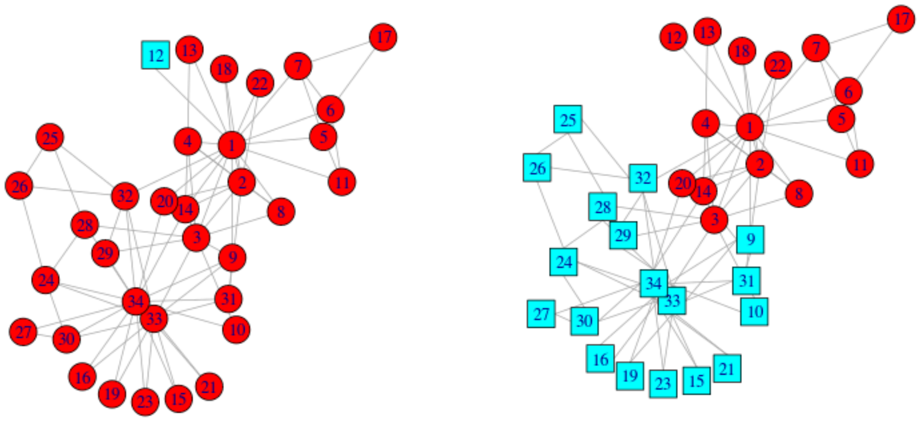


Figure 2: SC clustering results on 2014 Football graph

Additionally, we were initially getting incredibly different clusters for each implementation of SC on the Dumbbell and Karate graphs. To give SC the best possible chance to return the two communities that most algorithms agree on, we then implemented the criterion for SC to return the clustering where $|P| = 2$ for the Dumbbell and Karate graphs. Figure 3 shows two clusterings generated using the SC algorithm with these constraints on the Karate graph.



(a) One cluster only has one node

(b) Clusters with appropriate number of nodes

Figure 3: Forcing a clustering to stop at 2 clusters doesn't guarantee a good clustering

The clustering on the left is trivial, and resulted from the SC merging algorithm. To remove these clusterings from our results, we created a threshold, α, such that if

$$\min\{|C| : \forall C \in P\} \leq \alpha$$

then P was thrown out and a new clustering was generated. This process was repeated until

$$\min\{|C| : \forall C \in P_i\} > \alpha, \quad \forall P \in \theta.$$

Figure 4 shows the relationship between variance and threshold on the Karate graph.

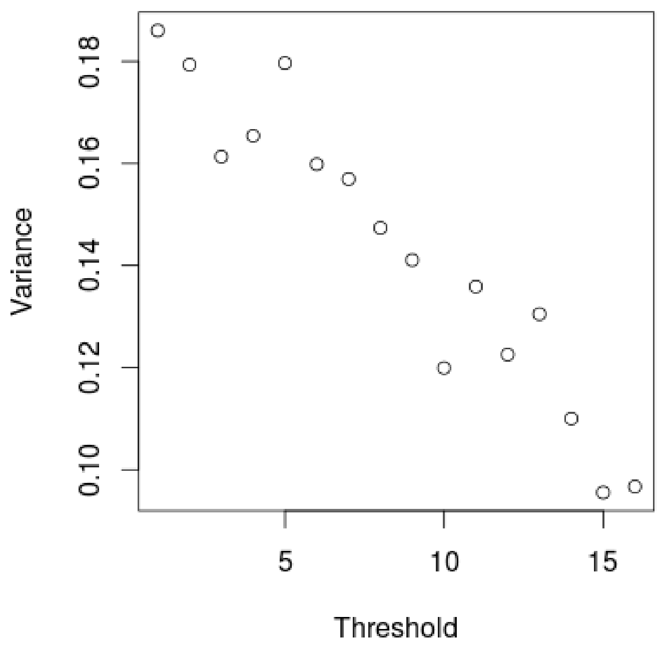


Figure 4: The relationship between the variance σ_θ^2 and threshold α for SC on Karate

It's important to note the the maximum value for α is 16 since $|V(G)| = 34$ for the Karate graph. Thus, even when we forced the clusters to be of equal sizes, there was still an average 10% of possible pairs that didn't agree between two different partitions. Table 1 show the final variances for each algorithm on all graphs.

	SC	SC-Mod	Spinglass	Label Propagation
Dumbbell	0.014	0.008	0.007	0.000
Karate	0.182	0.150	0.006	0.103
Football	0.071	0.070	0.002	0.014

Table 1: Comparison of variances for stochastic clustering algorithms

Spinglass and Label Propagation exhibited lower variances than both versions of SC on all three graphs. Further, Spinglass was the only algorithm to have extremely small variances (approximately $\leq .01$) on all three graphs.

8 "True labeling"

Steinhauser and Chawla recommended that modularity not be used as a metric for clustering, and instead, the "true labeling" of the vertices should be used as the benchmark. By true labeling we mean a labeling of the vertices, equivalently a partition of the vertices into clusters, provided a priori, which is taken to be the ground truth for the true community structure. Thus any clustering can be validated against the true labeling.

The true labeling of the vertices of graph sounds nice, because it defines an absolute truth to use in comparisons. One issue with using the true labeling of a graph when dealing with clustering is that not all graphs have a true labeling. Imagine a scenario where you need to create or adjust a clustering algorithm to provide "good" clusterings on graphs that don't have labels. You can't use labels to validate your algorithm because labels don't exist.

Another issue is that even if the graph did have a true labeling, the true labeling may be flawed. For example, an interesting issue with true labels was discovered by Miller when doing an AMS topic on Nation Football League (NFL) center and MIT graduate student, John Urschel [13]. Teams in the NFL are split among two conferences, the AFC and NFC, and within each conference a team belongs to one of four divisions (denoted by the cardinal directions North, South, East, and West). A graph of the 2015 NFL season can be generated by letting nodes represent teams and edges represent games played between two teams; this is shown in Figure 5 and was taken from [13].

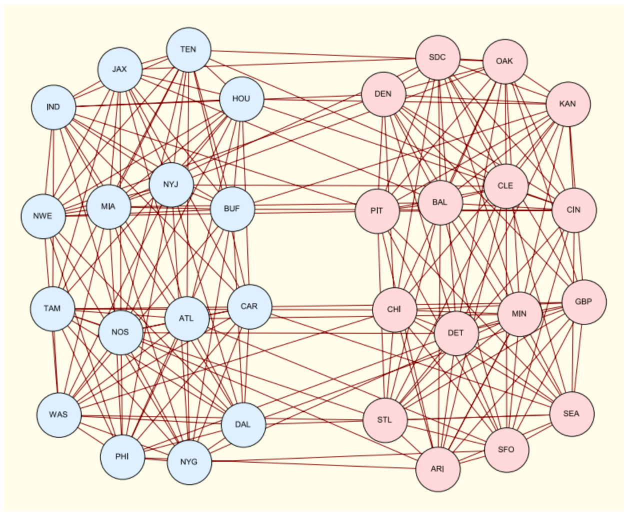


Figure 5: Spectral clustering on 2015 NFL graph

The teams on the upper-half of the graph belong to the AFC while the lower-half represents the NFC. In order to make sense of the Figure 5 its necessary to understand how the NFL schedules games.

In each season there are 192 intra-conference edges and 64 inter-conference edges. Each team will play:

- 2 games against each team in its division (12 intra-division edges per division)
- 1 game against each of the 4 teams from another division that is intra-conference (call the two divisions an intra-conference pair)
- 1 game against each of the 4 teams one division that is inter-conference (the the two divisions an inter-conference pair)
- 1 game against one team from the each of the 2 other intra-conference divisions.

It seems like since there are 96 intra-conference per conference, and only 64 inter-conference edges, naturally the best partition for splitting the NFL graph into 2 clusters would be to use each teams given conference.

However, the spectral clustering found by Miller assigned 50% of the teams to their "incorrect" respective conference. Miller went on to show that due to the structure in the 2015 schedule, an intra-division pair in the AFC was assign an inter-division pair in the NFC (this actually occurs every 3 years). Thus, there was more community structure across conferences than within conferences according to the Urschel–Zikatanov Theorem which Miller used to generate the clusters. The Urschel–Zikatanov Theorem used the eigen-values and eigen-vectors of the the Lapcian of the NFL graph to determine the clusters; this ultimately found the clustering on the NFL graph that split the teams into two clusters, where the number of edges between the clusters was minimum. In other words, the clustering shown in Figure 5 was (objectively) determined by partitioning the vertices so that the number of edges between partitions was minimized.

Now, imagine an algorithm that measures performance based on true labels. This fictitious algorithm would perform poorly–under its own standards–on the 2015 NFL graph, rather than declaring the results anomalous! This example reveals the importance graph structure has in clustering. If only the graph structure is used, then the results of clustering tell you something about the graph. If, instead, the accuracy of labeling is used, interpreting the results is less clear. If an algorithm that used true labeling validation was created and used on the NFL graph for years other than 2015, the algorithm might appear to do well during training (if the training years were 2013 and 2014), but then in 2015, you have results that you are extremely confident in but in reality they are incorrect!

This leads into the final concern with true labels. An algorithm that used true labeling as its basis for performance only has historical information as a means of credibility. This can become problematic if you'd like to test an algorithm on a graph where there has never been a true labeling assigned. For example, consider an algorithm that has performed extremely well on the standard graphs (i.e. Karate, NFL, etc.) under the true labeling validation. Now, apply the algorithm to a graph that has nothing to do with Karate clubs. How confident are you that the clustering is correct, and why are you confident?

This issue is related to the divide between unsupervised and supervised learning in Machine Learning. An unsupervised algorithm takes data that does not contains labels and organized that data points so that each data point is assigned a label. In contrast, supervised learning takes data with labels (known as training data) and trains the algorithm to gives output for new data based on the previously seen training. In Machine Learning, clustering is categorized as unsupervised learning because it should work without training data and solely on the structure of data. In general, clustering on a graphs seems to follow the unsupervised mold, because we want to give an output (i.e. clusters) only based on the structure of the graph(i.e. nodes and edges).

9 Conclusions

We defined the variance on a collection of clusterings that can be computed in $\mathcal{O}\big((Nn)^2\big)$. The variance can be used to measure the consistency of the clusterings which may be of interest with stochastic clustering methods. SC, SC-Mod, and Label Propogation all exhibited non-negligible variances on at least one of the graphs. We found that the same algorithm applied to different graphs exhibited different amounts of variance; this was true for SC, SC-Mod, and Label Propagation. Thus, we conclude that if consistency is important when generating clusterings on graphs, one should measure the amount of variance for a stochastic clusterings on a fixed graph or use deterministic methods since they have zero variance.

We investigated the notion of "true labeling" of the vertices. We discussed that although the idea of an absolute truth is convenient and has practical importance, one should be extremely cautious with using the true labeling as a benchmark for clustering algorithms.

References

[1] J. McAuley, R. Pandey, J. Leskovec, "Inferring networks of substitutable and complementary products," *Knowledge Discovery and Data Mining*, 2015

[2] P. Pons, M. Latapy, "Computing communities in large networks using random walks," *LIAFA - CNRS*, Dec. 2005.

[3] M. Clauset et al., "Finding community structure in very large networks," *Phys. Rev. E* 70, 6 Dec. 2004. doi: `10.1103/PhysRevE.70.066111`

[4] M. E. J. Newman, "Finding community structure in networks using eigenvectors of matrices," *Department of Physics and Center for the Study of Complex Systems*, Jun. 2006

[5] M. E. J. Newman, "Detecting community structure in networks," *Eur Phys. Rev. J. B.* 38,pp 321-330, Mar. 2004.

[6] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physics Review E*, vol. 69, 18 June 2004. DOI: `10.1103/PhysRevE.69.066133`

[7] K. Steinhaeuser K, N. V. Chawla, "Is Modularity the Answer to Evaluating Community Structure in Networks?" 26 June 2008.

[8] K. Steinhaeuser K, N. V. Chawla, "Identifying and evaluating community structure in complex networks" Pattern Recognition Lett. 2009, DOI:`10.1016/j.patrec.2009.11.001`

[9] Sports Reference LLC. "2014 College Football," *Sports Reference*, Oct. 2016, `http://www.sports-reference.com/cfb/years/2014-schedule.html`

[10] W. M. Rand, "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, Vol. 66, No. 336, pp.846-850, Dec. 1971

[11] U.N. Raghavan, R. Albert, S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks" *Phys Rev E 76*, 2007, `DOI:10.1103/PhysRevE.76.036106`

[12] J. Reichardt, S. Bornholdt, "Statistical Mechanics of Community Detection," 3 February 2008, DOI:`10.1103/PhysRevE.74.016110`

[13] S.D. Miller, "'I plan to be a great mathematician': An NFL Offensive Lineman Shows Hes One of Us,"Notices of the AMS, DOI: `http://dx.doi.org/10.1090/noti1331`

[14] W.W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups," *Journal of Anthropological Research*, Vol. 33, No. 4, pp. 452-473, 1977

[15] M. Matsumoto, T. Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator," 1998.