Learning Stochastic Block Models with Continuous Labels

April 20, 2022

Abstract

The Stochastic Block Model (SBM) is a method that allows us to model networks such as road systems and social media connections by classifying objects into a finite number of groups and then assuming underlying probabilistic connections between and within groups. This paper expands the scope of SBMs by allowing for multiple continuous labels for each object and also learning a function that can accurately represent the probability that two objects are related to each other. We prove that our Continuous Stochastic Block Model (CSBM) can accurately recover networks generated by the regular SBM up to a small error factor. We also analyze and prove the performance of CSBM recovery on some useful models that can be found in real life data, and test our CSBM recovery on two real-life datasets which are hypothesized to fit into the CSBM model.

Keywords: Stochastic Block Model (SBM), Random Graphs, Machine Learning, Rademacher Complexity, Log-Likelihood

Contents

1 Background

1.1 *Graph Theory*

Graph theory is used in many fields to model a set of objects in which certain pairs of objects have some connection. In a graph theory model, the underlying objects are referred to as vertices (or nodes) and pairs of vertices that have a relationship have an edge connecting them. Below are a few common scenarios that fit well into a graph theory model.

- *Transportation Networks*: This field of study looks to optimize investment in transportation infrastructure by studying graphs with cities/locations as the vertices and roads between two locations as the edges [1].
- *Disease Spread*: Graph theory can be used to model and understand the spread of diseases. A graph would be generated with virus hosts (such as people) as the vertices and interactions between pairs of hosts as the edges [2].
- *Electron Interactions*: Physics can use graph theory for approximating particle interactions. Particles like electrons form the vertices in the graph and physical interactions (such as the EM force) between pairs of electrons would be modeled as edges in the graph [3].

Definition 1.1. *A simple graph (network) G is defined as (V, E) where*

- *V is the set $\{v_1, ..., v_n\}$ of vertices with $|V| = n$ (n nodes)*
- *E is the set containing pairs of the form $\{v_i, v_j\}$. $E \subset V \times V$ with $|E| = m$ (m edges)*

Graphs can also be be expanded to include weights and directions on edges to generalize to situations. For example, in a transportation network model there might be one-way roads such that you can get from vertex v_1 to v_2 but not vice versa. In the transportation model the weight could represent the distance between v_1 and v_2.

Definition 1.2. *A weighted graph G is defined as (V, E) where*

- *V is the set $\{v_1, ..., v_n\}$ of vertices with $|V| = n$ (n nodes)*

- *E is the set $\{w_{i,j}, (i,j) \in [n] \times [n]\}$ of edges with weight $w_{i,j}$ from vertex v_i to vertex v_j with $|E| = m$ (m edges). Note that a weight $w_{i,j}$ of 0 indicates no edge.*

Thus a simple graph is just weighted graph G with all $w_{i,j} = 1$ and also contains an edge $(v_i, v_j, 1)$ iff there is an edge $(v_j, v_i, 1)$ (all edges go back and forth).

It is also useful to have an equivalent matrix representation for a graph G defined above. The matrix representation A of G is an $n \times n$ matrix that has 0 values in all coordinates $A_{i,j}$ where there is no edge between v_i and v_j and has the value $w_{i,j}$ on all other coordinates $A_{i,j}$. For example, if you take a 3 node graph $V = (A, B, C)$ with just 1 edge $E = (B, C)$ then the equivalent matrix representation of A would be the following matrix:

Figure 1: Graph Representation of A

	A	B	C
A	0	0	0
B	0	0	1
C	0	1	0

Figure 2: Matrix Representation of A

An important property of note for all matrices A representing simple weighted graphs is that the diagonal will always be 0.

1.2 *Stochastic Random Variables*

Random variables are used to represent objects whose outcome depends on some random phenomenon. Examples of random variables that are commonly used to model objects in daily life are coin-flips, the weather, and change of stock prices.

Definition 1.3. *A discrete random variable X is defined as variable whose output modeled using the probability function $f : \Omega \to [0, 1]$ where*

- *Ω is the finite sample space of distinct possible events*
- *the output $f(x \in \Omega)$ is the probability of an event occurring*

A simple relevant example of a discrete random variable is that of a weighted coin flip (also referred to as Bernoulli) random variable. This variable, X, is defined on the sample space of two events $\Omega = \{tails, heads\}$ where probability function f is defined as

$$f(x) = \begin{cases} p & x = heads, \\ 1-p & x = tails \end{cases}$$

Often we denote the *heads* event as 1 and the *tails* event as 0.

1.3 *Stochastic Block Models*

One important problem in graph theory is that of *community detection*: you are given a set of vertices and edges with the goal of partitioning vertices into clusters which are densely connected with each other. Community detection is useful in classifying groups using social networks as well as modeling information/disease spread in populations. An important model that is lends well to community detection is the stochastic block model. A stochastic block model is a randomly generated graph that can be used to simulate networks that are in real life data sets.

Definition 1.4. *Let $n \in \mathbb{N}$ be the number of vertices of set $V = \{v_1, v_2...v_n\}$ and $k \in \mathbb{N}$ represent the number of distinct communities. Let $p = (p_1, p_2...p_k)$ be the probability distribution which represents the probability that any given node v_i is in community $j \in [k]$. Let $W \in [0,1]^{k \times k}$ represent the probability matrix where W_{ij} is a value corresponding to the Bernoulli variable with probability W_{ij} of an edge between an element of community i with an element of community j. An instance of a stochastic block model generated by SBM(n, p, W) is defined as (L, G) where each L_i (the label of the node) is randomly chosen from distribution p and G is the graph on those n nodes with edges randomly generated using W.*

A simple example of a two-community (A,B) stochastic block graph is $SBM(n, p, W)$ where $p = (.5, .5)$ and W $= \begin{bmatrix} .8 & .1 \\ .1 & .8 \end{bmatrix}$. Note that W must be a symmetric matrix ($W = W^T$). Here we have a total of n vertices almost evenly partitioned into two communities (A, B) where any two nodes within the same community (either both in A or both in B) have a probability of .8 of having an edge between each other and there is a .1 probability of an

edge between any two vertices of opposite communities. Below is an example of such a graph with $n = 50$ vertices.

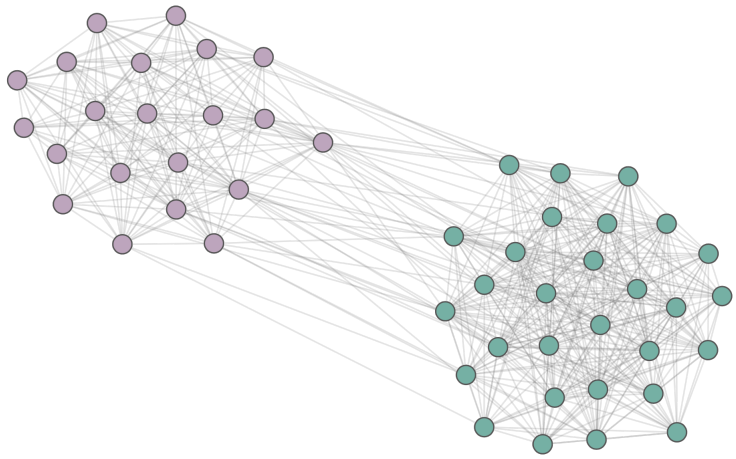

Figure 3: Example graph of 50 nodes generated by with .8 probability of an edge between nodes of the same community and .1 probability of an edge between nodes of different populations. Nodes in group A are colored purple and nodes in group B are colored green

A large segment of stochastic block model theory is focused on *recovery* of stochastic block models. The problem of stochastic block model recovery is as follows: you have access to one instance G of a stochastic random graph generated using some $SBM(n, p, W)$ and the goal is to recover V (the community membership of each node) and W with a high degree of accuracy. Since labels $i \in [k]$ are almost always interchangeable, we define the notion of *Agreement* below in order to measure the level of accuracy of various recovery models.

Definition 1.5. *Agreement between two community vectors $x, y \in [k]^n$, $A(x, y)$ is the maximal agreement between x and any permutation of relabeling of y.*

$$A(x, y) = \frac{1}{n} \max_{\pi \in S_k} \sum_{i=1}^{n} \mathbb{1}(x_i = \pi(y_i))$$

You are given an instance of (L, G) generated by $SBM(n, p, W)$. The goal is to create a function/algorithm F that takes in G and outputs $\hat{L}$ such that $\hat{L}$ and L have high agreement. Exact recovery is a regime of $SBM(n, p, W)$ that allows us to recovery all the labels correctly

with high probability and almost exact recovery is a regime of $SBM(n, p, W)$ that allows recovery of almost all the labels correctly with a high degree of accuracy.

Definition 1.6. Exact Recovery*:* $\mathbb{P}\{A(L, \hat{L}) = 1\} = 1 - o(1)$ *(for a large number n we will output the partition of labels exactly correct with probability 99%).*

Definition 1.7. Almost Exact Recovery*:* $\mathbb{P}\{A(L, \hat{L}) = 1 - o(1)\} = 1 - o(1)$ *(for a large number n we will output the partition of labels with 99% accuracy with probability 99%).*

1.4 *Proven Bounds on SBM Recovery*

Many problems look at regimes of specific types of $SBM(n, p, W)$ in which exact recovery is feasible. For example, if all entries in W are equivalent then the graph collapses into an instance of an Erdos-Renyi graph and it is impossible to partition the vertices. Similarly if a graph is disconnected it also becomes impossible to label a component region with probability more than 1 so the average degree must be $\Omega(\log(n))$. One specific set of SBMs that are studied to determine conditions where exact recovery is possible are symmetric SBMs on n communities. Symmetric SBMs are ones where the diagonal entries of W have value p and the off-diagonal entries have value q. Also, the distribution of community assignments is uniform. Thus each element has a probability $p = \frac{a \log(n)}{n}$ of an edge with an element within its community and a probability $q = \frac{b \log(n)}{n}$ of an edge otherwise. An interesting result that arises is that there is a threshold at which, if p and q are close enough to each other, it becomes impossible to implement exact recovery of the labels.

Theorem 1. *[4] Exact recovery in a symmetric SBM with two populations given by* $SBM(n, (1/2, 1/2), \frac{a \log(n)}{n}, \frac{b \log(n)}{n})$ *is efficiently solvable if* $|\sqrt{a} - \sqrt{b}| > \sqrt{2}$ *and unsolvable if* $|\sqrt{a} - \sqrt{b}| < \sqrt{2}$

1.5 *Log Likelihood Optimization*

Some efficient algorithms in solving the SBM recovery problem approach it by trying to optimizing the log-likelihood of "witnessing" the graph G over the set of possible priors of labels. Mathematically this is finding a set of parameters θ such that $\log(\mathbb{P}(G|\ \theta))$ is maximized. A simple relevant example of log-likelihood optimization is having a set of

$\theta \in [k]^n \times [0,1]^{k \times k}$ which is the combination of the set of labeling for assigning n nodes to k communities concatenated with the predicted probability matrix $\hat{W}$. Thus we can create an approximation $\hat{L} = F(\theta_{1:m})$ by solving $\text{argmax}_\theta(\log(\mathbb{P}(G|\ \theta)))$ and using F to output a set of labels. There are two caveats to this approach: first you aren't guaranteed that $A(\hat{L}, L) \approx 1$ (the closeness of L and $\hat{L}$ depends on generalization error) and the second is that it is often computationally infeasible to sample all the possible θ that can minimize the log likelihood.

1.6 *Datasets*

To test our model discusses in the results section we use the following 3 datasets.

1. *Simulated Data*: This dataset is very effective at analyzing what recovery looks like in a controlled environment where we know the underlying probability matrix X. Knowing the true dataset, we can measure the total error (in Frobenius norm or Hellinger distance) of our recovery. Simulations will include dense SBM networks, sparse SBM networks, assymmetric SBM networks, and networks generated with the CSBM model described in results.

2. *Blog Post Dataset*[5]: This dataset looks at a total of 1490 different political blog posts/websites and there are 16718 different directed edges between blogs where a blog references another blog. For the purposes of this paper we treated the blog-post graph as an undirected graph. Blog websites were classified into two groups of leaning politically left and leaning politically right.

3. *Word Adjacency Dataset*[6]: This network looked at a total of 112 of the most common nouns and adjectives in the novel "David Copperfield" by Charles Dickens. The edges between words are tallied by the number of instances that any two words are directly adjacent each other in the text. A representation of the unweighted graph of this dataset is shown in the results page.

2 Results

2.1 *Continuous Label Extension*

We begin by defining the new model that builds upon the previous stochastic block model by instead allowing for continuous labels and a function F that can assign a probability of an edge between any two labels. To our knowledge this model has not yet been attempted. The closest paper that approach our model is an experimental one that uses a neural network to assign nodes to a finite number of communities by incorporating added information including continuous attribute labels for nodes [7]. Data sets that fit this model are those where labels can be placed on a spectrum (for example political leanings) and a relation between two data points depends on some function (generally a distance) metric relating two labels.

Definition 2.1. *Let $n \in \mathbb{N}$ number of vertices of set $V = (v_1, v_2...v_n)$ such that each v_i is iid drawn from distribution Ω. Let $F : (v_i, v_j) \to [0, 1]$ be an unknown function that assigns a probability of an edge between two nodes v_i and v_j. Let matrix $X \in [0, 1]^{n\times n}$ be defined such that $X_{ij} = f(v_i, v_j)$ and X represents the probabilistic matrix representation of graph that has an edge between vertex i and j with probability X_{ij}. An instance of a continuous stochastic block model generated by CSBM(n, Ω, F) is defined as (L, G) where each node in L is independently drawn from distribution Ω and G is the graph on those n nodes with edges randomly generated using F. We also define the matrix M as the $\{0, 1\}^{n\times n}$ matrix that represents where or not there is an edge between two nodes in G.*

There are two main approaches to CSBM recovery. We will discuss the first and the main issues that arise when working with the first implementation, and why we chose to focus on the second.

2.1.1 Implementation 1: Single Label Arbitrary Function

In the first implementation we learn a set both the set of labels $\hat{L} \in \mathbb{R}^n$ and a general function $\hat{F} : \mathbb{R}^2 \to \mathbb{R}$ such that $\hat{F}(\hat{L}_i, \hat{L}_j) \approx F(L_i, L_j)$. There are several important things to note about this implementation. First it is impossible to recover either F or L_i on their own (or even get close). To see why consider two functions F and G such

that $G = cF$ and F draws labels from Ω and G draws labels from $\frac{\Omega}{c}$. It is clear that $CSBM(n, \Omega, F) = CSBM(n, \frac{\Omega}{n}, G)$ thus it would be impossible to distinguish the two.

Additionally $\hat{F}$ cannot be an arbitrary function. If $\hat{F}$ has VC dimension n^2 then it would fit $\{0,1\}^{n^2}$ points perfectly and output a probability distribution identical to M (since that would maximize the log likelihood function). Thus there needs to be a Rademacher complexity argument that can bound the amount that a function class of $\hat{F}$ could overfit.

There are other restrictions on $\hat{F}$. For obvious reasons (since we are considering undirected graphs) $\hat{F}(x, y) = \hat{F}(y, x)$. Another restriction on the $\hat{F}$ (if we want to be able to learn arbitrary SBM instances) is that $\hat{F}$ cannot be separable, ie $\hat{F}(x, y) \neq g(x) * h(y)$. The reasoning behind this is simple stochastic block matrix defined by the transition matrix

$$\begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

where A, B, C are all block matrices that give probabilities a, b, c of any two nodes within/between the communities having an edge. Then assume that $\hat{F}(x, y) = g(x) * h(y)$. We then get that

$$\hat{F}(X_a, X_a) = g(X_a) * h(X_a) = g(X_a)^2 = h(X_a)^2 = a$$

$$\hat{F}(X_b, X_b) = g(X_b) * h(X_b) = g(X_b)^2 = h(X_b)^2 = b$$

which in turn forces the value for edges c between a and b to be

$$\hat{F}(X_a, X_b) = g(X_a) * h(X_b) = \sqrt{ab}$$

but c could be arbitrary which means this model cannot capture most simple SBMs

The final complication with this model is the difficulty of *interpretability*. For example, assume in the real world the probability of connection between two people on Facebook was a function of how close they were geographically and also politically. In this model having just one label would not provide insight that there were two major features that affect an edge between two people.

2.1.2 Implementation 2: Multiple Labels

We learn a set both the set V of L labels per node $\hat{L} \in (i^{\{0,1\}}\mathbb{R})^{d \times n}$ and a function $\hat{F} : (i^{\{0,1\}}\mathbb{R})^{2d} \to \mathbb{R}$ such that $\hat{F}(\hat{L}_i, \hat{L}_j) = G(\sum_{k=1}^{d} \hat{L}_{ik} * \hat{L}_{jk}) \approx F(L_i, L_j)$. In essence this

is finding a set of k features for each node that can be used as a dot product to feed into a cumulative density function G that would output a value between [0,1]. We will discuss a natural choice for G in the correctness choice. There are several key differences between this and the previous implementation.

- *Multiple Labels*: In the original implementation there was only 1 label per node. Having multiple labels per node allows for visualization and characterization of attributes that effect community grouping
- *Fixed function definition*: As opposed to learning the function $\hat{F}$ the flexibility in this model lies in the label set and uses a fixed function (sigmoid in this paper) to output a final probability of an edge.
- *Introduction of Complex Labels*: One requirement of this model is allowing labels to take on imaginary variables. This is because labels will represent eigenvectors scaled by the square root of eigenvalues of the eigendecomposition of the true probability matrix X. Because X is symmetric we have that eigenvalues are real (but can take on negative values). Thus we allow labels to take on values in $i^{\{0,1\}}\mathbb{R}$.

The benefits of this model is that we can use linear algebraic properties and theorems to prove error bounds on the learn-ability of the true distribution of $\hat{X}$. Other benefits include clear generalizability to the SBM problem as well as understanding underlying features responsible for communities in graphs. The next step is to prove that this implementation is able to solve the continuous equivalent of *recovery* discussed in definitions 1.6 and 1.7. An important note here is that exact label recovery is still impossible in this scheme for the same reasons defined in Implementation 1.

2.2 Algorithm for Continuous Recovery and Correctness

The analogous problem of CSBM recovery focuses on determining labels that are "close" to each other iff labels are "close" in the support of Ω (this however requires F to be a metric function). An easier form of recovery (to prove and work with) focuses on outputting a probabilistic matrix $\hat{X}$ that is "close" in distance to the true probabilistic matrix X. This is the definition that we will focus on for this paper. First we define useful metrics of closeness.

Definition 2.2. *Hellinger distance d_H between two Bernoulli variables with probabilities p and q is defined as*

$$d_H^2(p,q) = (\sqrt{p} - \sqrt{q})^2 + (\sqrt{1-p} - \sqrt{1-q})^2$$

Definition 2.3. *Hellinger distance d_H between two matrices $P, Q \in [0,1]^{n \times n}$ of Bernoulli variables is the average Hellinger distance at each Hellinger coordinate*

$$d_H^2(P,Q) = \frac{1}{n^2} \sum_{i,j} d_H^2(P_{ij}, Q_{ij})$$

Definition 2.4. *Frobenius Norm of a matrix X is denoted as $||X||_F$ and is calcualted as*

$$\sqrt{\sum_{i,j} X_{i,j}^2}$$

In the problem of CSBM recovery you are given an instance of (V, G) generated by $CSBM(n, \Omega, F)$ and you must output labels $\hat{V}$ (where the domain of V_i is not necessarily the same as that of $\hat{V}_i$) and the true probabilistic matrix $\hat{X}$ for the that hold for the following definitions of recovery.

Definition 2.5. Continuous Label Recovery*: A $(\mu_1, \mu_1, \delta, \epsilon)$ continuous recovery of a CSBM is when $\mu_1(\hat{L}_i, \hat{L}_j) > \delta \rightarrow \mu_2(L_i, L_j) > \epsilon$ with probability 1-o(1)*

Definition 2.6. Probabilistic Matrix Recovery*: An δ, ϵ probabilistic matrix recovery of a CSBM is when $d_H^2(X, \hat{X}) < \delta$ with probability $1 - \epsilon$*

We proceed with this paper's proposed implementation for Probabilistic Matrix Recovery. Then we show that our method achieves strong bounds for probabilistic recovery on instances of simple SBMs as well as useful CSBMs. The key idea is that we "learn" a set of labels $L \in \mathbb{R}^{n \times r}$ of r labels for each node in V (r depends on the number of significant eigenvalues of M). Then we approximate the probability of an edge between two distinct vertices V_i and V_j by $S(L_i \cdot L_j)$ where S is the sigmoid function $S(x) = \frac{e^x}{1+e^x}$. The inverse sigmoid function is $S^{-1}(x) = \log(\frac{x}{1-x})$.

The method of recovery used to determine the values of L is gradient descent on the log-likelihood function $LL(\hat{X}, M)$ followed by projection of $\hat{X}$ onto the set of rank r matrices. The log-likelihood function of a given $\{0,1\}^{n \times n}$ output given a transition $[0,1]^{n \times n}$ matrix is defined as follows

$$LL(\hat{X}, M) = \log(\prod_{\substack{i,j \\ i \neq j}} S(\hat{X}_{ij})^{M_{ij}} \cdot S(1-\hat{X}_{ij})^{1-M_{ij}}) = \sum_{\substack{i,j \\ i \neq j}} \mathbb{1}_{M_{ij}=1} \log(S(\hat{X}_{ij})) + \mathbb{1}_{M_{ij}=0} \log(S(1-\hat{X}_{ij}))$$

Note that we don't include the log-likelihood values along the diagonal. For all intents and purposes all values along the diagonal must be 0 since we assume that G is a simple graph so a vertex cannot have an edge with itself. For simplicity in proofs we define the equivalence relation $\sim$ between matrices such that $X \sim Y \iff X_{ij} = Y_{ij} \ \forall (i,j), i \neq j$

Proposition 1. *If $M \in \{0,1\}^{n \times n}$ is drawn from the matrix distribution $X \in [0,1]^{n \times n}$ then*

$$\operatorname*{argmax}_{\hat{X}}(\mathbb{E}_M[LL(\hat{X}, M)]) = X$$

Proof. Since we can decompose $LL(\hat{X}, M))$ as the coordinate-wise sum of loss functions $L(x, i, j)$ defined below we have

$$\mathbb{E}_M[LL(\hat{X}, M)) = \sum_{\substack{i,j \\ i \neq j}} \mathbb{E}[M_{ij} \log(\hat{X_{ij}}) + (1 - M_{ij}) \log(1 - \hat{X_{ij}})]$$

$$L(x, i, j) = M_{ij} \log(x) + (1 - M_{ij}) \log(1 - x)$$

$$\operatorname*{argmax}_{x}(\mathbb{E}[L(x, i, j)]) = X_{ij}$$

$$\operatorname*{argmax}_{\hat{X}}(\mathbb{E}_M[LL(\hat{X}, M))_{ij} = X_{ij}$$

□

Theorem 2. *[8] Assume that the true probability matrix, X has rank r and $||X||_\infty < \alpha$. Let M be the probability matrix drawn from X and let $\hat{X}$ be the solution to the algorithms minimization of Log loss over the set of rank r matrices. Then with probability at least* $1 - C_1/n$

$$\frac{1}{n^2}||M - \hat{M}||_F^2 \leq 2C_\alpha \sqrt{\frac{r}{n}}$$

Where $C_\alpha = \alpha * e^\alpha * C_2$

For the proof it will be useful to work with the normalized log loss function $\overline{LL}$ and establish its concentration propery.

$$\overline{LL}(X, M) = LL(X, M) - LL(0, M)$$

The majority of this theorem relies upon the following lemma (which we will prove afterwards).

Lemma 1. *Let A be the set of $n \times n$ matrices with rank d and max coordinate value α*

$$\mathbb{P}(\sup_{X \in A} |\overline{LL}(X, M) - \mathbb{E}[\overline{LL}(X, M)]| \geq C_0 * n\alpha\sqrt{dn}) \leq \frac{C_1}{n}$$

Proof. First we show that using Markov's inequality we have

$$\mathbb{P}(\sup_{X \in A} |\overline{LL}(X, M) - \mathbb{E}[\overline{LL}(X, M)] \geq C_0 * n\alpha\sqrt{dn}) =$$

$$\mathbb{P}(\sup_{X \in A} |\overline{LL}(X, M) - \mathbb{E}[\overline{LL}(X, M)]^h \geq (C_0 * n\alpha\sqrt{dn})^h)$$

$$\leq \frac{\mathbb{E}[\sup_{X \in A} |\overline{LL}(X, M) - \mathbb{E}[\overline{LL}(X, M)]^h]}{(C_0 * n\alpha\sqrt{dn})^h}$$

Thus we want to find a bound on the quantity $\mathbb{E}[\sup_{X \in A} |\overline{LL}(X, M) - \mathbb{E}[\overline{LL}(X, M)]^h]$ and that (combined with a proper choice of h) will give the bound stated in the Lemma. Note that we can rewrite $\overline{LL}$ as

$$\overline{LL}(X, M) = 2 \sum_{\substack{i,j \\ i<j}} \mathbb{1}_{M_{ij}=1} \log\Big(\frac{S(X_{ij})}{S(0)}\Big) + \mathbb{1}_{M_{ij}=0} \log\Big(\frac{S(1 - X_{ij})}{S(0)}\Big)$$

By the symmetrization argument (lemma proven below) we have

$$\mathbb{E}\Big[\sup_{X \in A} |\overline{LL}(X, M) - \mathbb{E}[\overline{LL}(X, M)]|^h\Big]$$
$$\leq 2^h \mathbb{E}\Big[\sup_{X \in A} \Big|2 \sum_{\substack{i,j \\ i<j}} e_{i,j} \Big(\mathbb{1}_{M_{ij}=1} \log\Big(\frac{S(X_{ij})}{S(0)}\Big) + \mathbb{1}_{M_{ij}=0} \log\Big(\frac{S(1 - X_{ij})}{S(0)}\Big)\Big)\Big|^h\Big]$$

Where $e_{i,j}$ are Rademacher variables in {-1,1} and the expectation is over M and choices of $e_{i,j}$. Now we bound the latter term using the contraction principle [9] since $\log\left(\frac{S(X_{ij})}{S(0)}\right)$ and $\log\left(\frac{S(1-X_{ij})}{S(0)}\right)$ are 1-lipschitz and vanish at 0. Thus the supremum increases by at

most a factor of 2 when replacing those terms with X_{ij} and $-X_{ij}$ respectively.

$$\mathbb{E}\Big[\sup_{X\in A}|\overline{LL}(X,M)-\mathbb{E}[\overline{LL}(X,M)]|^h\Big]$$
$$\leq 4^h\mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{\substack{i,j\\i<j}}e_{i,j}\Big(\mathbb{1}_{M_{ij}=1}\log\Big(\frac{S(X_{ij})}{S(0)}\Big)+\mathbb{1}_{M_{ij}=0}\log\Big(\frac{S(1-X_{ij})}{S(0)}\Big)\Big)\Big|^h\Big]$$
$$\leq 4^h\mathbb{E}\Big[\sup_{X\in A}\Big|2\sum_{\substack{i,j\\i<j}}e_{i,j}\Big(\mathbb{1}_{M_{ij}=1}X_{ij}-\mathbb{1}_{M_{ij}=0}X_{ij})\Big)\Big|^h\Big]$$
$$= 8^h\mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{\substack{i,j\\i<j}}e_{i,j}\Big(\mathbb{1}_{M_{ij}=1}X_{ij}-\mathbb{1}_{M_{ij}=0}X_{ij})\Big)\Big|^h\Big]$$
$$= 8^h\mathbb{E}\Big[\sup_{X\in A}\Big|\frac{1}{2}\langle E\odot M',X\rangle\Big|^h\Big]$$

Where E is the symmetric random Rademacher matrix with $E_{ij}=E_{ji}=e_{i,j}$ and M' is matrix M with -1 values instead of 0s. For simplicity let the diagonal of E be 0s. Since the distribution of $E\odot M'$ is the same as that of E and the value $|\langle A,B\rangle|<||A||\ ||B||_*$ (where $||B||_*$ is defined as the nuclear norm of B) we can simplify the above expression.

$$8^h\mathbb{E}\Big[\sup_{X\in A}\Big|\frac{1}{2}\langle E\odot M',X\rangle\Big|^h\Big]=4^h\mathbb{E}\Big[\sup_{X\in A}\Big|\langle E\odot M',X\rangle\Big|^h\Big]=4^h\mathbb{E}\Big[\sup_{X\in A}\Big|\langle E,X\rangle\Big|^h\Big]$$
$$\leq 4^h\mathbb{E}\Big[\sup_{X\in A}\Big||||E||^h||X||_*^h\Big|\Big]\leq 4^h(\alpha n\sqrt{d})^h\mathbb{E}\Big[||E||^h\Big]$$

Now we analyze a bound on the expected value of the operator norm $\mathbb{E}[||E||^h]$. Observe that the entries in E have expected value of 0 and let $h\leq 2\log n$; thus we use Theorem 1.1 of [10]

$$(\mathbb{E}[||E||^h])\leq C\Big(2\mathbb{E}\Big[(\sum_{i=1}^{n-1}1)^{\frac{h}{2}}\Big]\Big)\leq 2Cn^{\frac{h}{2}}$$

Where C above is some constant. Now plugging our bounds back in we get

$$4^h(\alpha n\sqrt{d})^h\mathbb{E}\Big[||E||^h\Big]\leq(4n\alpha\sqrt{dn})^h*2C$$

Which for $C_0=4$, $h=\log(n)$, and $C_1=2C$ we prove the lemma

$$\mathbb{P}(\sup_{X\in A}|\overline{LL}(X,M)-\mathbb{E}[\overline{LL}(X,M)]|\geq C_0*n\alpha\sqrt{dn})$$

$$\leq\frac{\mathbb{E}[\sup_{X\in A}|\overline{LL}(X,M)-\mathbb{E}[\overline{LL}(X,M)]^h]}{(C_0*n\alpha\sqrt{dn})^h}$$

$$\leq \frac{(4n\alpha\sqrt{dn})^h * 2C}{(C_0 * n\alpha\sqrt{dn})^h} = \frac{C_1}{n}$$

□

We can now use this lemma to bound the Hellinger distance between $S(\hat{X})$ and $S(X)$. We introduce the notation for KL divergence $D(p||q)$ between two Bernouli variables p and q

$$D(p||q) = p\log(\frac{p}{q}) + (1-p)\log(\frac{1-p}{1-q})$$

and for two matrices P, Q we have the KL divergence defined as

$$D(P||Q) = \frac{1}{n^2}\sum_{i,j} D(P_{i,j}||Q_{i,j})$$

Lemma 2. *Symmetrization: for 0-mean variables $L_{i,X}, i \in [n]$ variables and for a sequence of independent rademacher variables $e_i, i \in [n]$*

$$\mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{i=1}^{n} L_{i,X} - \mathbb{E}[\sum_{i=1}^{n} L_{i,X}]\Big|^h\Big] \leq 2^h\mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{i=1}^{n} e_i L_{i,X}\Big|^h\Big]$$

Proof. Consider the set L'_i which is an independent copy of the the set of random variables L_i. Note that since $L_i - L'_i$ is mean 0 and symmetric, we have that the distribution of $L_i - L'_i$ is equal to that of the distribution of $e_i(L_i - L'_i)$ Now we have that using Jensen's since the supremum and absolute value functions are convex

$$\mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{i=1}^{n} L_{i,X} - \mathbb{E}[\sum_{i=1}^{n} L'_{i,X}]\Big|^h\Big] = \mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{i=1}^{n} L_{i,X} - \mathbb{E}[L'_{i,X}]\Big|^h\Big]$$

$$\leq \mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{i=1}^{n} L_{i,X} - L'_{i,X}\Big|^h\Big]$$

$$= \mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{i=1}^{n} e_i(L_{i,X} - L'_{i,X})\Big|^h\Big]$$

$$\leq 2^h\mathbb{E}\Big[\sup_{X\in A}\Big|\sum_{i=1}^{n} e_i L_{i,X}\Big|^h\Big]$$

□

Theorem 3. *[8] Assume that the true probability matrix, X has rank d and $||X||_\infty < \alpha$. Let M be an instance drawn from the probability matrix X and let $\hat{X}$ be the solution to the*

algorithms minimization of Log loss over the set of rank r matrices. Then with probability at least $1 - C_1/n$

$$d_H^2(S(X), S(\hat{X})) \ \leq 2C_\alpha \sqrt{\frac{d}{n}}$$

Where $C_\alpha = 2\alpha C_0$

Proof. We directly use Lemma 1 to help show that uniform concentration of log loss gives concentration of Hellinger distance. First we see that,

$$\begin{aligned}
\mathbb{E}[\overline{LL}(\hat{X}, M) - \overline{LL}(X, M)] &= \mathbb{E}[LL(\hat{X}, M) - LL(0, M) - (LL(X, M) - LL(0, M))] \\
&= \mathbb{E}[LL(\hat{X}, M) - LL(X, M)] \\
&= \sum_{i,j} S(X_{i,j}) \log\left(\frac{S(\hat{X}_{i,j})}{S(X_{i,j})}\right) + (1 - S(X_{i,j})) \log\left(\frac{1 - S(\hat{X}_{i,j})}{1 - S(X_{i,j})}\right) \\
&= -n^2 D(S(X)||S(\hat{X}))
\end{aligned}$$

Here the expectation is over M. Now assume $Y \in A$

$$\begin{aligned}
\overline{LL}(Y, M) - \overline{LL}(X, M) &= \mathbb{E}[\overline{LL}(Y, M) - \overline{LL}(X, M)] + (\overline{LL}(Y, M) - \mathbb{E}[\overline{LL}(Y, M)]) \\
&\quad + (\overline{LL}(X, M) - \mathbb{E}[\overline{LL}(X, M)]) \\
&\leq \mathbb{E}[\overline{LL}(Y, M) - \overline{LL}(X, M)] + 2 \sup_{Y \in A} |\overline{LL}(Y, M) - \mathbb{E}[\overline{LL}(Y, M)]| \\
&\leq -n^2 D(S(X)||S(Y)) + 2 \sup_{Y \in A} |\overline{LL}(Y, M) - \mathbb{E}[\overline{LL}(Y, M)]|
\end{aligned}$$

And since by definition $\hat{X}$ optimizes for log-loss we have that

$$0 \leq -n^2 D(S(X)||S(\hat{X})) + 2 \sup_{Y \in A} |\overline{LL}(Y, M) - \mathbb{E}[\overline{LL}(Y, M)]|$$

Lemma one gives that with probability greater than $1 - \frac{C_1}{n}$ we have that

$$0 \leq -n^2 D(S(X)||S(\hat{X})) + 2C_0 * n\alpha\sqrt{dn}$$

$$D(S(X)||S(\hat{X})) \leq 2C_0\alpha \sqrt{\frac{d}{n}}$$

We can bound KL divergence with Hellinger distance using the fact that $1 - x \leq -\log x$

$$D(p||q) = p \log(\frac{p}{q}) + (1 - p) \log(\frac{1 - p}{1 - q})$$

$$= 2\left(p\log\left(\sqrt{\frac{p}{q}}\right) + (1-p)\log\left(\sqrt{\frac{1-p}{1-q}}\right)\right)$$

$$= -2\left(p\log\left(\sqrt{\frac{q}{p}}\right) + (1-p)\log\left(\sqrt{\frac{1-q}{1-p}}\right)\right)$$

$$\geq 2p(1-\sqrt{\frac{q}{p}}) + 2(1-p)(1-\sqrt{\frac{1-q}{1-p}}))$$

$$= 2p - 2\sqrt{pq} + 2(1-p) - \sqrt{(1-p)(1-q)})$$

$$= 2 - 2\sqrt{pq} - 2\sqrt{(1-p)(1-q)}$$

$$= (\sqrt{p}-\sqrt{q})^2 + (\sqrt{1-p}-\sqrt{1-q})^2 = d_H^2(p,q)$$

Which gives that

$$d_H^2(S(M), S(\hat{M})) \ \leq C_\alpha\sqrt{\frac{d}{n}}$$

Where $C_\alpha = 2\alpha * C_0$

□

We now show how Theorem 1 directly leads to Theorem 2 using the following lemma

Lemma 3. *[8] Let* $||X||_\infty, \hat{X}_\infty \leq \alpha$ *Then*

$$d_H^2(S(X), S(\hat{X})) \geq \frac{S'(\alpha)||X-\hat{X}||_F^2}{8n^2}$$

Proof. Consider any entry (x, y) in the pair of matrices $(x, y) \in (X_{i,i}, Y_{i,j})$ The Hellinger distance between them

$$(\sqrt{S(x)}-\sqrt{S(y)})^2 + (\sqrt{1-S(x)}-\sqrt{1-S(y)})^2$$

$$\geq \frac{1}{2}\Big((\sqrt{S(x)}-\sqrt{S(y)}) - (\sqrt{1-S(x)}-\sqrt{1-S(y)})\Big)^2$$

Which we can then use Taylor's theorem by choosing a z between x and y such that

$$\frac{1}{2}\Big((\sqrt{S(x)}-\sqrt{S(y)}) - (\sqrt{1-S(x)}-\sqrt{1-S(y)})\Big)^2$$

$$\geq \frac{1}{2}\left(\left(\sqrt{S(z)}\right)'(y-x) + \left(\sqrt{1-S(z)}\right)'(y-x)\right)^2$$

$$= \frac{1}{2}\left(\frac{S'(z)}{2\sqrt{S(z)}}(y-x) + \frac{S'(z)}{2\sqrt{1-S(z)}}(y-x)\right)^2$$

$$\geq \frac{1}{8}(y-x)^2(S'(z))^2\left(\frac{1}{S(x)} + \frac{1}{1-S(x)}\right)$$

$$= \frac{(S'(z))^2}{8S(z)(1-S(z))}(y-x)^2$$
$$= \frac{S'(z)(y-x)^2}{8}$$

Thus we get the lemma by summing up across all the entries and dividing by n^2 □

2.3 Generalizability of Continuous Recovery Method

In this section we show how we can apply the theorems in the above section to prove recovery bounds for various generated matrices.

Proposition 2. *Any instance graph G of k communities generated by $SBM(n,p,W)$ has a $(\delta,\epsilon) = (2C_\alpha\sqrt{\frac{k}{n}}, \frac{C_1}{n})$ probabilistic matrix recovery where*

$$C_\alpha = 2C_0 \max_{w\in W}(\log(w) - \log(1-w))$$

Proof. Consider the probability transition matrix $S(X)$ generated by $SBM(n,p,W)$. Allow the diagonal entries to be arbitrary such that there are k distinct rows of this matrix and thus this matrix has rank k. Let $S^{-1}(S(X)) = X$ be the matrix of underlying values we are trying to learn. We have that $||X||_\infty = \max_{w\in W}(\log(w) - \log(1-w))$ and since there are still k distinct rows we have that $rank(X) \leq k$. Thus by Theorem 3 we have that with probability $1 - \frac{C_1}{n}$

$$d_H^2(S(X), S(\hat{X})) \ \leq 2C_\alpha\sqrt{\frac{k}{n}}$$

□

Now we show a generalization of the above theorem to a class of functions possible in the $CSBM$ model. First we define the degree of a 2-variable polynomial.

Definition 2.7. *Consider the 2-degree polynomial $F(x,y) = \sum_{i,j} a_{i,j}x^iy^j$. The degree of a 2-variable polynomial is defined as*

$$deg(F(x,y)) = \max_{\substack{i,j \\ a_{i,j}\neq 0}} (i+j)$$

Proposition 3. *Consider the $CSBM(n,\Omega,F)$ where $F = S \circ G$ and $deg(G) \leq d$. Then we can obtain a $(\delta,\epsilon) = ((d+1)C_\alpha\sqrt{\frac{1}{n}}, \frac{C_1}{n})$ probabilistic matrix recovery where*

$$C_\alpha = 2\sqrt{2}C_0 \max_{x,y\in\Omega}(|G(x,y)|)$$

Proof. Let $S^{-1}(F(X)) = X$ be the matrix of underlying values we are trying to learn such that $X_{i,j} = G(v_i, v_j)$ and $v_i, v_j \in \Omega$. We have that $||X||_\infty = \max_{x,y\in\Omega}(|G(x,y)|)$. Now we bound the rank of X. Since we have $deg(G) \leq d$ we know that G has at most $\frac{d(d+1)}{2}$ terms of the form $a_{i,j}x^i y^j$ (using a simple counting argument). Now let Y_{ij} represent the matrix corresponding to the function $a_{i,j}x^i y^j$. Thus $X = \sum_{i,j} Y_{i,j}$. Now we can clearly see that each of $Y_{i,j}$ have rank 1 since $Y_i, j = \alpha_{i,j}(v^i)^T * v^j$ and since rank is sub-additive we have that

$$rank(X) = rank(\sum_{i,j} Y_{i,j}) \leq \sum_{i,j} rank(Y_{i,j}) = \frac{d(d+1)}{2}$$

. Thus by Theorem 3 we have that with probability $1 - \frac{C_1}{n}$

$$d_H^2(S(X), S(\hat{X})) \ \leq (d+1)C_\alpha\sqrt{\frac{k}{n}}$$

□

This is a pretty good result because it now allows us to generalize to learning arbitrary functions by using Taylor approximation with d-degree polynomials. Thus in theory for very large graphs where $n >> d$ we can learn matrices generated by arbitrary polynomials with high precision.

2.4 *Examples on Simulated Datasets*

We can now look at the implementation of the above algorithm described in section 2.2 on some simulated datasets. We overall effectiveness of the model is measured by calculating $||X - \hat{X}||_F^2$ on the datasets (note that X and $\hat{X}$ probability matrices). However, we also examine the distribution of "continuous labels" L of the points generated by looking at the eigendecomposition of $\hat{X}$ such that $\hat{X} = Q\Lambda Q^T$ (which is made possible since $\hat{X}$ is a symmetric matrix) and the jth feature of the ith point is

$$L_{i,j} = \sqrt{\Lambda_{j,j}} * Q_{i,j}$$

2.4.1 Dataset 1

The first simulated dataset is a simple symmetric SBM on 1000 points with .8 probability of an edge within clusters and .2 probability of an edge between clusters. This graph has

a similar appearance to that of Figure 1. Below is a plot of the log-loss function over time along with the distribution of the rank-2 eigendecomposition labels of $\hat{X}$. The value of $||X - \hat{X}||_F^2 = .0006$ and the average deviation of $X_{i,j}$ from $\hat{X}_{i,j}$ is approximately .015.

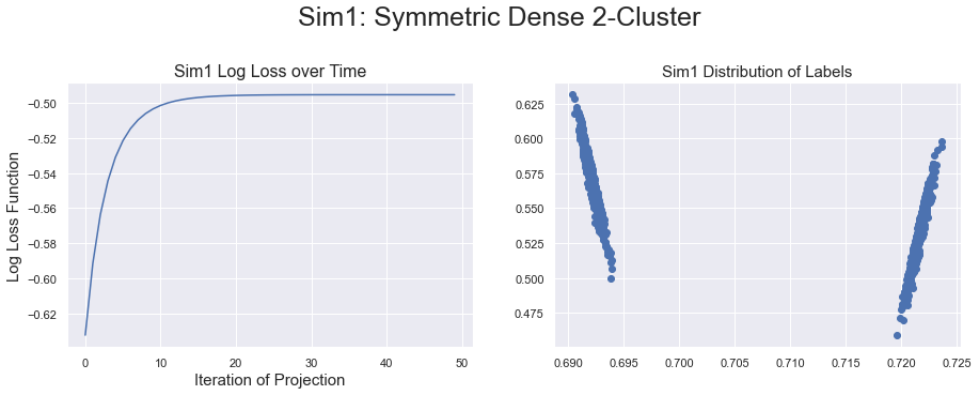

Figure 4: Log loss and clustering of dataset 1. Note that there are two distinct clusters

2.4.2 Dataset 2

The second simulated dataset is a symmetric sparse SBM on 2000 points with .01 probability of an edge within clusters and .005 probability of an edge between clusters. Below is a plot of the log-loss function over time along with the distribution of the rank-2 eigendecomposition labels of $\hat{X}$. The value of $||X - \hat{X}||_F^2 = 1.6 * 10^{-5}$ and the average deviation of $X_{i,j}$ from $\hat{X}_{i,j}$ is approximately .002. Note that due to the result of Theorem 1 and the fact that $\sqrt{2} - \sqrt{1} < \sqrt{2}$ we do not expect two completely separated clusters as in dataset 1.

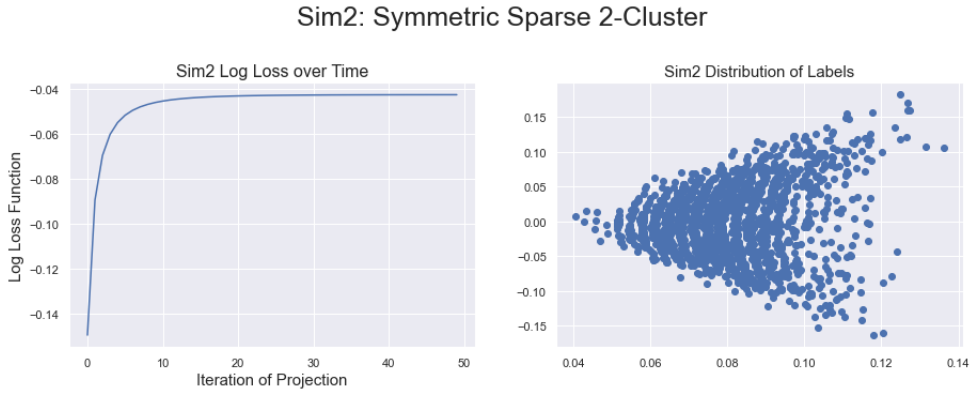

Figure 5: Log loss and clustering of dataset 2. Note that there are no distinct clusters

2.4.3 Dataset 3

The third simulated dataset is a symmetric SBM on 1000 points with .15 probability of an edge within clusters and .03 probability of an edge between clusters. Below is a plot of the log-loss function over time along with the distribution of the rank-2 eigendecomposition labels of $\hat{X}$. The value of $||X - \hat{X}||_F^2 = .00031$ and the average deviation of $X_{i,j}$ from $\hat{X}_{i,j}$ is approximately .001.

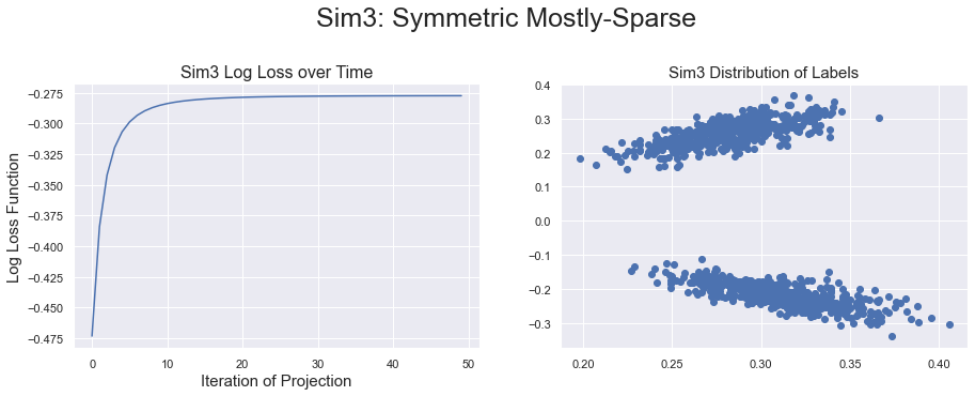

Figure 6: Log loss and clustering of dataset 3. Note that we still have distinct clusters

2.4.4 Dataset 4

The fourth simulated dataset is an assymetric SBM on 1000 points where $W = \begin{bmatrix} .8 & .2 \\ .2 & .4 \end{bmatrix}$. Below is a plot of the log-loss function over time along with the distribution of the rank-2 eigendecomposition labels of $\hat{X}$. The value of $||X - \hat{X}||_F^2 = .00073$ and the average deviation of $X_{i,j}$ from $\hat{X}_{i,j}$ is approximately .018.

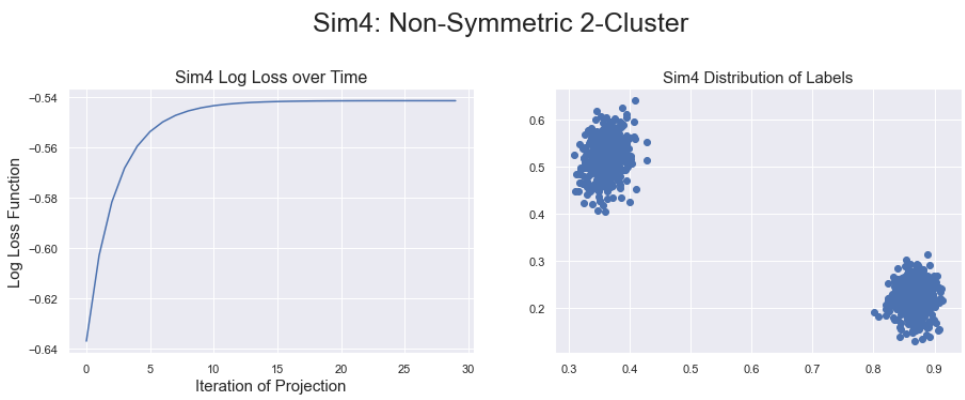

Figure 7: Log loss and clustering of dataset 4. Note that we have distinct clusters of labels

2.4.5 Dataset 5

The fifth simulated dataset is an symmetric 3-community SBM on 1000 points where

$$W = \begin{bmatrix} .8 & .2 & .2 \\ .2 & .8 & .2 \\ .2 & .2 & .8 \end{bmatrix}$$

Below is a plot of the log-loss function over time along with the distribution of the rank-3 eigendecomposition labels of $\hat{X}$. The rank-3 eigendecomposition is used due to the proposition 3. The value of $||X - \hat{X}||_F^2 = .00153$ and the average deviation of $X_{i,j}$ from $\hat{X}_{i,j}$ is approximately .026.

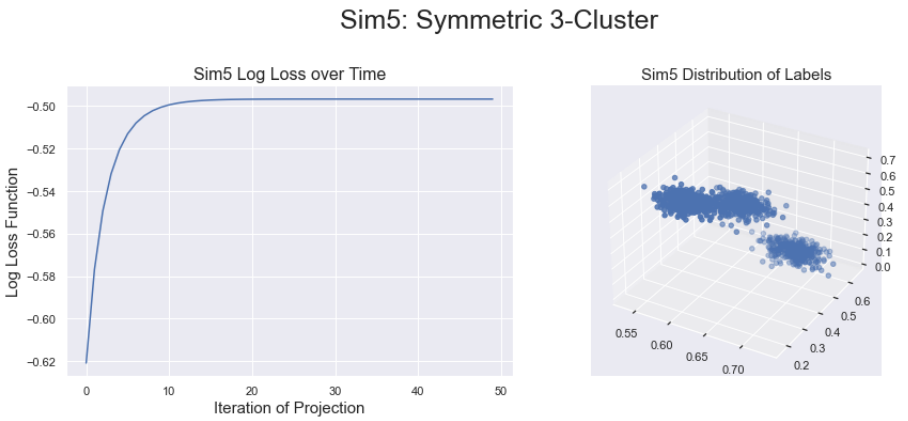

Figure 8: Log loss and clustering of dataset 5. Note that we 3 distinct clusters of labels (this is easier to verify by looking at projections into 2d space)

2.4.6 Dataset 6

The sixth and final dataset is an example of a CSBM generated by the sampling v_i uniformly from [0,1] and applying the function $G(x, y) = (x-y)^2$. Below we have the log-loss function over time along with the distribution of the rank-3 eigendecomposition labels of $\hat{X}$. The value of

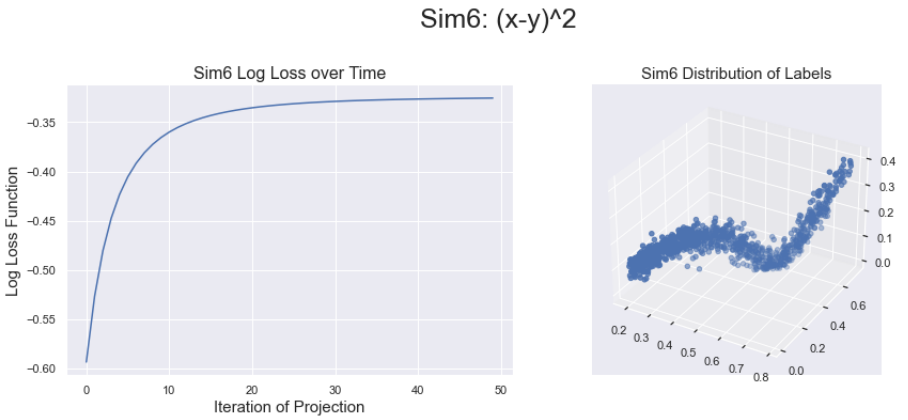

Figure 9: Log loss and clustering of dataset 6. Note the interesting distribution of labels in 3d space

2.5 *Examples on Real Life Data*

We now take a look at the effectiveness of this algorithm when applied to real-life datasets which were described in section 1.6. For these datasets we do not have access to the underlying probability matrix X that we can use to verify the correctness of our matrix recovery algorithm. Instead, we analyze whether or not our labels are able to pick out major features that should effect community formation.

2.5.1 *Political Blogs*

The political blog dataset looks at 1490 different political blog posts/websites and there are 16718 different directed edges between blogs where a blog references another blog. We treat the graph as undirected and (since we have access to a spectrum of 2 communities) we try run our algorithm using a 2-rank approximation). The results of the log-loss progression and the distribution of the labels is shown below.

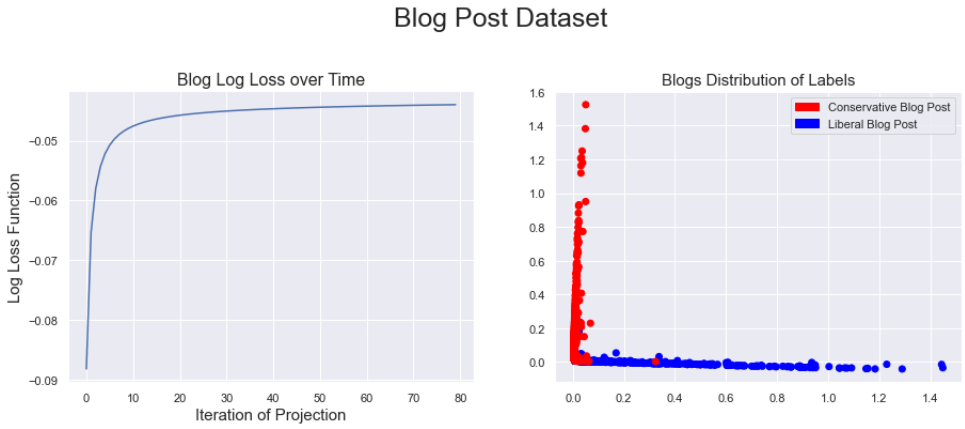

Figure 10: Log loss and clustering of Blog Post Dataset. Note that the left and right leaning distributions are well-separated

The result is interesting, not only because it can draw a distinction between right and left leaning blog posts, but also because it fits with the idea of a political spectrum where far-right posts are highly unlikely to have an edge with far-left posts and there are centrists blogs that overlap. Thus the CSBM method allows us to both categorize the political affiliation of a blog as well as determine its extent: something which is not feasible in the

simple SBM scheme.

2.5.2 *Word Adjacency Dataset*

This dataset looked at a total of 112 of the most common nouns and adjectives in the novel "David Copperfield" by Charles Dickens and weighted edges represent number of instances that any two words are directly adjacent each other in the text. We simplify our representation to have unweighted edges. A representation of the graph of this dataset is shown below.

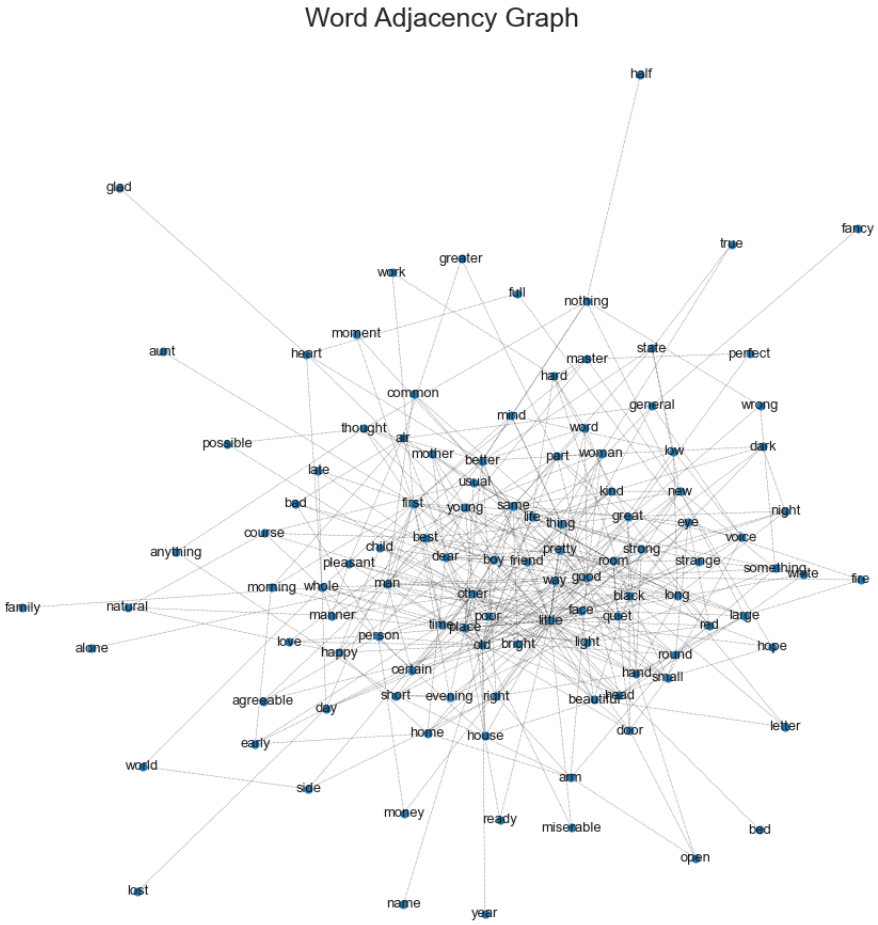

Figure 11: The graph of the adjacency dataset containing nouns and adjectives (an edge between two words means they can be found next to each other in the text)

Logically this adjacency graph should be close to a bipartite graph separating the nouns

and adjectives since both have a very small probability of appearing next to each other (much more likely for a noun to appear next to an adjective). Thus we expect some community-based seperation in the distributions of the labels. The results of the log-loss of our algorithm and the plot of the label distributions is below.

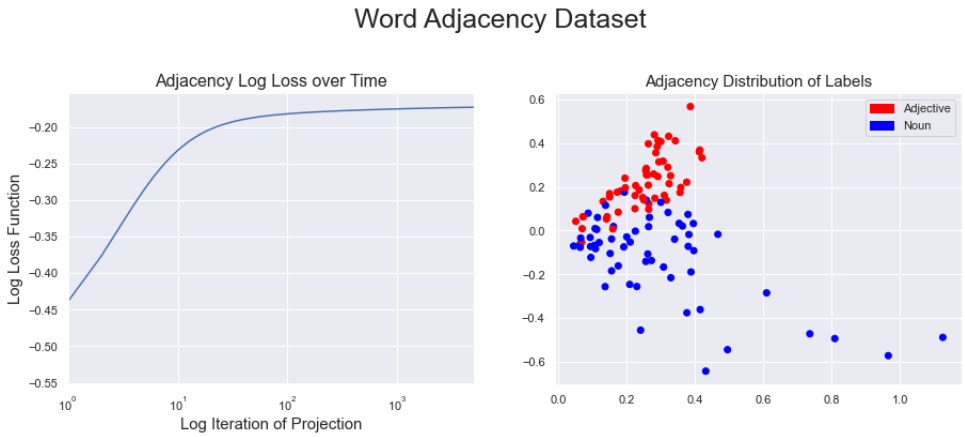

Figure 12: Log loss and clustering of Adjective/Noun Dataset. Note that the adjectives and nouns are linearly separable which suggests a community relationship that could be bipartite

3 Discussion

In this paper we have defined a new stochastic block model with continuous labels which we call the $CSBM$ model. We then went on to show that this model has a parallel definition for *recovery* to the SBM model and showed that we can implement an algorithm for Probabilistic Matrix Recovery of $CSBM$ instances generated via SBM models and $CSBM$ models with polynomial functions.

We then tested the effectiveness of the implementation on simulated datasets of sparse and dense graphs to show that the bounds hold. We then analyzed the results of our algorithm on real-life datasets modeling connections between political blogs and between noun-adjective pairs.

One interesting connection to explore in the future is the relationship between the rank-d matrix output by our algorithm and the rank-d matrix output by the top d vectors of the eigendecomposition of the original $\{0,1\}^{n\times n}$ matrix. We showed in our simulation

results that these two approaches were quite similar in terms of the output matrix $\hat{X}$ and it would be a worthwhile effort to prove that they are "close" to each other since an eigendecomposition can be more efficiently calculated than our gradient-descent algorithm.

4 Acknowledgements

I would like to thank Dr. Joe Neeman for his guidance in providing an interesting area of research and guiding me with the main proofs in this paper. I would also like to thank my parents and my friends for all of their support through my undergraduate process.

References

[1] Federico Pablo-Martí and Angel Sánchez. Improving transportation networks: Effects of population structure and decision making policies. *Scientific reports*, 7(1):1–9, 2017.

[2] Lauren Ancel Meyers, MEJ Newman, Michael Martin, and Stephanie Schrag. Applying network theory to epidemics: control measures for mycoplasma pneumoniae outbreaks. *Emerging infectious diseases*, 9(2):204, 2003.

[3] Ernesto Estrada. Graph and network theory in physics. a short introduction. Technical report, 2013.

[4] Emmanuel Abbe, Afonso S. Bandeira, and Georgina Hall. Exact recovery in the stochastic block model, 2014.

[5] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, LinkKDD '05, page 36–43, New York, NY, USA, 2005. Association for Computing Machinery.

[6] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), Sep 2006.

[7] Natalie Stanley, Thomas Bonacci, Roland Kwitt, Marc Niethammer, and Peter J. Mucha. Stochastic block models with multiple continuous attributes, 2018.

[8] Qingqing Huang, Sham M. Kakade, Weihao Kong, and Gregory Valiant. Recovering structured probability matrices, 2018.

[9] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

[10] Yoav Seginer. The expected norm of random matrices. *Combinatorics, Probability and Computing*, 9(2):149–166, 2000.