CLUSTERING ON BIPARTITE GRAPH

by

Anil Gaddam

McGill University

Project report submitted to McGill University in partial fulfillment of the requirements for the

Masters (Non-Thesis)

Department of Electrical and Computer Engineering

APRIL 2019

Approved by : _____

Prof. Mark Coates

CERTIFICATE OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified with references, and that the original work contained herein has not been performed by unspecified sources or persons.

Anil Gaddam

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Professor Mark Coates for accepting me in his group, his insightful comments, remarks, and engagement through my learning process for my Master Project. Furthermore, I would really like to thank Mathew Goonewardena (postdoc) in our Computer Networks Lab for introducing me to the topic. I am very grateful for his technical and moral support during my Master project. I would also want to thank Soumyasundar Pal (Ph.D. candidate) for proving insightful comments while writing my report. I sincerely thank them for their co-operation and helping with necessary information regarding the project.

I would like to thank the students in Computer Networks Lab for providing an enriching and motivating environment. I thank for their kindness during my journey here at McGill.

I am grateful for the Department of Electrical and Computer Engineering for awarding graduate excellence fellowship. This helped me maintain financial stability during my masters.

Finally, I would like to thank my parents who have always supported and encouraged me to pursue masters.

Table of Contents

List of Figures

List of Tables

1 Overview

1.1 Introduction

We live in a world with abundant data. Analysis of this data using statistical and machine learning methods allows us to discover useful information that can help in decision making [1]. Data analysis is performed on two types of data — labeled and unlabelled, and the associated machine learning activities are called supervised and unsupervised learning, respectively. Supervised learning is the task of learning a mapping between an input and an output based on the available example pairs of input-output responses. By contrast, unsupervised learning focuses on inferring the structure of the generative process from observed datasets [2].

Unsupervised learning has been conducted in research areas like image processing, biochemistry, bio-informatics and pattern recognition [2]. Some of the problems that arise in unsupervised learning are solved using clustering or anomaly detection [3, 4]. Example applications include object recognition [5] and pedestrian detection [6].

Clustering is essential in many data analysis tasks [7]. Clustering aims to separate a finite unlabeled data set into a finite and discrete set of "natural", hidden data groups [8]. It addresses the task of grouping similar entities together for further analysis.

Clustering algorithms that are developed to solve problems in a specialized field usually, make assumptions that are specific to the application of interest [9]. Some of the algorithms used for clustering are K-Means, Fuzzy c-means, DBSCAN, WaveCluster, ORCLUS, and hierarchical clustering [10]. Most of the clustering algorithms follow an iterative procedure to find a solution and the number of iterations depends on the complexity of the data [11]. Clustering algorithms can be grouped into two categories based on the degree of membership of a cluster assigned to observation. In *hard* clustering, a data point can belong to only one group whereas, in *soft* clustering, a data point can belong to multiple groups.

In the field of machine learning, data is available in various forms for supervised, unsupervised or semi-supervised learning. Representation of this data in the form of graphs helps capture insights to make better data-driven decisions. Graphs are

considered to be a ubiquitous data structure [12]. Recommender systems, social networks, and molecular structures can be modeled as graph data. In the graph framework, we can capture dependencies and interaction in the data through the presence of edges between nodes. Graph inference tasks of interest include classification of individual nodes or the entire graph, link prediction, and regression [12]. Some of the interesting applications based on graph-based data include predicting the role of a person in an organizational network and recommending friends on a social network [13].

Bipartite graphs consist of two different sets of nodes. An edge can only exist if the nodes are not in the same set. There are many settings where a bipartite graph is an appropriate model, including networks with plants and pollinators [14], social network users and locations [15], scientific papers and authors [16], and movie-viewer rating networks.

Clustering on graphs is usually performed based on the similarity of nodes and their associativity. The goal in such a task is to group the set of nodes based on two criteria: *homogeneity* — similarity of nodes within a cluster; and *separation* — nodes in different clusters with low similarity. In many real-world scenarios, such as communication network analysis, information is associated with a network that can be modeled as a graph. Clustering is a crucial problem to address in telecommunication network management and performance analysis. Multiple techniques have been proposed in the literature to perform this task [17, 18] but almost all algorithms have serious drawbacks, often either being too computationally demanding to scale to large graphs or identifying a poor clustering solution. Thus it becomes important to conduct research into novel graph clustering algorithms and models and contribute with methods that identify useful clusters but can also scale to large graphs. My approach is to introduce probabilistic generative models that specify the probability of cluster memberships given the observed graph topology. I then apply approximate Bayesian inference, experimenting with both Markov Chain Monte Carlo sampling techniques and variational inference.

In this project, I address the task of dividing N nodes in an unweighted bipartite graph into K clusters, where we assume that we are provided with the graph topology and we know the target number of clusters K. We propose two new algorithms: an algorithm based on posterior inference using a Mixture of Bernoulli

distributions model and an algorithm employing a new variant of a Variational Autoencoder (VAE). For the first algorithm, we use a Bayesian inference approach and perform clustering based on a posterior estimate using Markov Chain Monte Carlo (MCMC) sampling. For the second algorithm, a modified VAE [19] is used to learn latent representations of the nodes in a space of reduced dimension. Graph VAE's have been used for predicting links between nodes in graphs [20]. However, the main objective of this project is to explore the underlying distribution of the data. After constructing the latent representations using the graph VAE, a Gaussian mixture model is used to identify the clusters.

In experiments with synthetic data, the proposed algorithms outperformed several of the existing unsupervised models in the literature [21, 22]. I apply the developed techniques to the study of data from a communication network that examines the relationships between cellular base stations and mobile users. The experimental results demonstrate that the algorithms can scale to large bipartite graphs (80,000 base stations and 280,000 users) and outperform performance baselines such as autoencoders [22] and Latent Dirichlet Allocation (LDA) [21].

The pipeline for this machine learning project includes: setting up the dataset for preprocessing, identifying relevant information, building a model, training the model and finally obtaining the predictions.

1.2 Organization

Section 2 studies the related research on graph dataset inference. Section 3 defines the problem more concretely and introduces relevant notation. Section 4 develops the background for the work in the project. Section 5 provides the methodology, explaining the adopted models and presenting the developed algorithms. In Section 6 we describe the experiments and present the results. Section 7 summarizes the contributions and outcomes of the project and identifies avenues for future research.

2 Related Work

Bipartite graph inference often addresses the task of predicting links or edges between heterogeneous sets of vertices in the graph. An example in a supervised learning setting is compound-protein interaction network link prediction [23].

A variety of methods have been employed to perform community detection in bipartite graphs [14, 24, 25]. In a bipartite graph the nodes can be divided into two types (type a and type b). Zhou et al. use a one-mode projection approach to identify a separate graph for each type of node in the original bipartite graph. Traditional graph clustering and community detection methods can then be applied to these projected graphs. This approach has two disadvantages. The projection leads to the graphs that contain overlapping cliques and most community detection algorithms struggle to identify communities from graphs with such structure. The projection also leads to a loss of information about the topology of the graph.

Stochastic Block Models (SBMs) have also been used for community detection on bipartite graphs [14, 25]. The bipartite Stochastic Block Model (biSBM), presented by Larremore et al. in [14], divides nodes of both types into blocks and then specifies the likelihood of an edge as being dependent only on the block membership. Larremore et al. also introduce a degree-corrected version of this model which preserves the observed degree sequence. A greedy stochastic optimization procedure which involves swapping individual nodes from one group to another is used to search for the groupings (blocks) and model parameters that maximize the likelihood of the observed graph. This model and algorithm can perform well and can scale reasonably well to larger graphs. An important deficiency is an inability for nodes to be members of multiple communities. This could be rectified by extending mixed-membership stochastic block models to bipartite graphs.

3 Problem Statement

This project addresses the problem of clustering the nodes in large bipartite graphs. A cluster is a set of nodes of the same type. This task arises in numerous applications where we have two types of nodes and the edges between them represent the relationships between pairs of nodes. The goal of clustering is to identify clusters so that nodes within each cluster have similar behavior in terms of the edges they form with nodes of the other type. Clustering strives to identify groupings so that there is a high intracluster similarity and low intercluster similarity. Thus, the performance of the clustering algorithms can be assessed using metrics that reward the presence of multiple intracluster edges to a given node of the other type.

In this project, we focus on the specific task of clustering base stations in a telecommunication network. The process of grouping these base stations can be based on the observed patterns of interactions between the users and base stations. Given this observed data, the aim is to identify clusters of base stations based on the common users they share.

3.1 Problem Formulation

We consider a clustering problem on an unweighted, undirected bipartite graph $\mathcal{G} = (V, E)$, where the vertices (nodes) V is the union of two disjoint sets, i.e, V = $\mathcal{X} \cup \mathcal{Y}$, such that every edge in the set E connects a vertex in $\mathcal{X}$ to one in $\mathcal{Y}$. The edges on the graph $\mathcal{G}$ are unweighted. We strive to identify clusters of the nodes in $\mathcal{X}$ based on the extent to which their neighborhoods overlap (the neighborhoods consist entirely of nodes in $\mathcal{Y}$). We define a cluster as a subset of nodes in $\mathcal{X}$. We assume that each node can belong to multiple clusters and that the task involves identifying a pre-specified number of K clusters. Also, we impose the constraint that each node must belong to at least one cluster.

We use a performance measure to evaluate the performance of a clustering model and algorithm. We introduce the *cluster utilization* metric which measures the density of edges within the clusters between $\mathcal{X}$ and $\mathcal{Y}$ nodes. To identify this, we calculate a parameter κ. We find a set $\mathcal{E}$ ($\subset \mathcal{Y}$) which are connected to nodes of X in the cluster. In each cluster, κ is defined as the ratio of the number of nodes of $\mathcal{E}$ that have at most ζ edges($\leq$ number of nodes in $\mathcal{X}$) to the total number of

nodes in $\mathcal{E}$,

$$\kappa = \frac{\sum\limits_{y \in \mathcal{E}} \mathbb{I}_{\zeta}(y)}{\sum\limits_{y \in \mathcal{E}} y} \tag{1}$$

where $\mathbb{I}_{\zeta}$ is an indicator function for nodes with at most ζ edges. We evaluate with different values of ζ and pick the value of ζ which gives maximum κ. Identify the percentage of ζ over the number of nodes in $\mathcal{X}$. Cluster utilization is determined by average value of ζ over all the clusters. It is the quantity which quantifies the neighborhood affinity for nodes.

We also use the silhouette coefficient [26] which evaluates the correctness of clustering based on the distance between nodes in the cluster. *Silhouette coefficient*, s is defined as a metric which measures the proximity of nodes in the clusters

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{2}$$

Here, a is average distance between a node i and all other nodes in the same cluster(intracluster) and b is the smallest average distance between node i to all nodes in any other cluster(intercluster). It calculates intracluster and intercluster distance with the neighboring cluster to measure how tightly the points are grouped in a cluster. Clustering can be considered of good quality if the overall cluster utilization is high among all the clusters formed. Given the topology information of the graph $\mathcal{G}$, our objective is to find an optimal model m(from a set of models M) that has a high value of cluster utilization, $\underset{m \in M}{\operatorname{argmax}}\, m_{\kappa}$. We further use the silhouette coefficient s to check if it corroborates the findings from cluster utilization.

4 Background

4.1 Graph Structures

In understanding the complex structure of graphs, the modern science of detecting communities has seen significant advances [27]. Clustering nodes in graphs is of great significance in the fields of Sociology, Biology, Chemistry, and Telecommunication where data is often represented as graphs [28]. Various techniques are applied assuming that each node in a graph belongs to single or multiple communities [17, 18].

Graph data observed in many real-life applications are in unipartite or bipartite forms. In a unipartite graph, all the vertices are considered to be of the same type. Fig 1 is one such example of the graph.

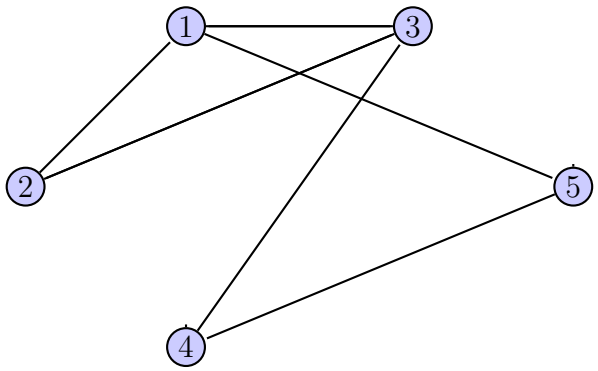


Figure 1: Unipartite graph structure

In a bipartite setting, there are two different types of vertices [14]. Only vertices of different types can be connected. The structures in the bipartite form make community detection challenging. A good example of a bipartite graph network is the user rating system. In this setup, we can see users and movies as different sets of vertices [29].

Models for unipartite graphs such as the Mixed Membership Stochastic Block Models (MMSB) [17] and the assortative MMSB (a-MMSB) [18] provide benchmarks for inferring clusters from graph-based structures. These models inspired the work on bipartite graphs based on a similar formulation in [14, 30]. In developing novel clustering algorithms for bipartite graphs, it is important to consider the issues of computation, complexity, and scalability to large graphs.

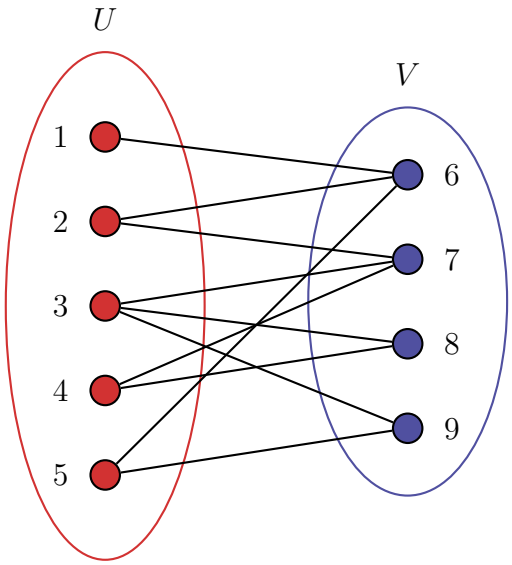


Figure 2: Two sets of vertices U and V in a bipartite graph. $\{1, 2, 3, 4, 5\} \in U$ can only be connected nodes $\{6, 7, 8, 9\} \in V$. Nodes in the same set will have no edges among themselves.

4.2 Autoencoders

Autoencoder is an unsupervised machine learning algorithm with an objective to transform input with least possible distortion [22]. It is trained with an objective to reduce the dimensionality of the input space. The architecture of an autoencoder is similar to that of neural networks. It consists of input, encoder, decoder, and output layers. The encoder and decoder layers are the hidden layers. Encoding layer tries to capture correlations in the data to get a compressed representation of the input. The decoder reconstructs the input from the encoded layer. Since we are not concerned about labels while training it is an unsupervised machine learning technique.

In Fig 3, the three partitions of an autoencoder can be seen. Code layer represents the encoded input derived from the encoder layers. The output is the reconstructed input. In the case of supervised learning, mean square error $J(x, z) = \|x - z\|^2$ is typically considered as loss function. While in the case of unsupervised learning, cross entropy $J(x, z) = \sum_k [x_k log(z_k) + (1 - x_k) log(1 - z_k)]$ can be used as loss function. This loss is minimized by backpropagation during training [22]. This type of a neural network is used majorly in reducing the dimensionality of the input space and minimize the difference for reconstructed input [31]. This results in lossy compression. Instead of trying to learn a function like in autoencoders, there is variational autoencoder which tries to learn the parameters of a probability

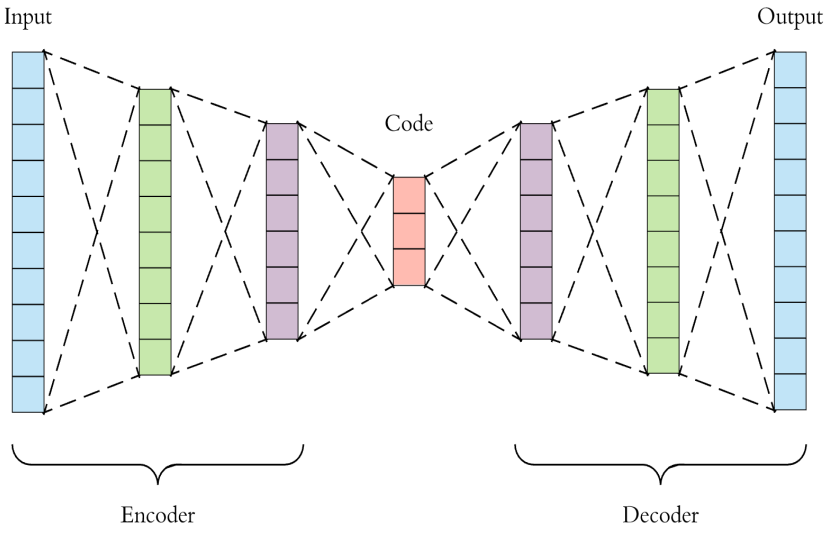


Figure 3: Encoder and Decoder layers along with the reduced dimensioned code is presented. Image reproduced from http://towardsdatascience.com

distribution representing the data. These are used as generative models which makes it possible to sample from a data distribution similar to the one from which the input might have been sampled.

4.3 Variational Autoencoder

Generative modeling is an area in machine learning which deals with understanding the distribution of data $X, P(X)$. In many real-life scenarios, these distributions are very difficult to comprehend. Advantages of generative modeling come into picture when we can use this knowledge of data to generate new data. Variational Autoencoder (VAE) is one such technique based on autoencoder architecture used to infer the distribution of data. VAE has been successful in generating different kinds of complicated data like faces [19], handwritten digits images [32], the physical model of scenes [33] and many more. Learning the model using VAE is seen to be fast because of its weak assumptions and training with back-propagation [34].

To infer the underlying distribution of data X, in VAE the main objective is to maximize $P(X)$. This is given by equation considering a latent variable z

$$P(X) = \int P(X|z;\theta)P(z)dz \tag{3}$$

For the model to be representative of dataset, some setting for the latent variables exists which causes the model to generate data similar to X. Consider a vector of latent variables z in a high-dimensional space Z sampled from probability density

function (PDF) P(z) defined over Z. θ is the parameter for a deterministic function $f(z;\theta)$ which is used to construct $P(X|z;\theta)$ in eq 3 . In VAE, Gaussian is considered as a standard choice of distribution i.e., $P(X|z;\theta) = \mathcal{N}(X|f(z;\theta), \sigma^2 * I)$

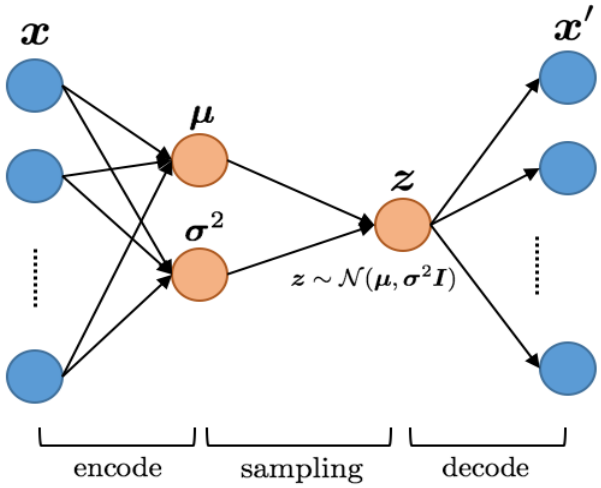


Figure 4: Variational Autoencoder. Encoding layer has input x and other hidden layers. Sampling is from the obtained layers of parameters μ and σ^2. Decoding layer used to reconstruct the input x which is represented as x'.

In Fig 4 the parameters of μ and σ^2 are parameters of the latent variable z.

Random samples are drawn using these parameters and fed to the decoder to get the reconstructed input. The sampled z should be such that it increases the probability $P(x)$. This is done by considering a variational distribution $Q(z)$ which is an approximation to $P(z|X)$. This relates to the probability of z when X is fed into the system. Then the objective is to minimize this KL divergence between $P(z|x)$ and $Q(z)$. During backpropagation, reconstruction loss(cross entropy in case of unsupervised setting) and KL divergence are minimized. KL divergence forces the learned distribution to be approximately standard Gaussian. There is a reparameterization trick [19] involved during backpropagation which makes the gradient calculation possible.

Instead of

$$x = sample(\mathcal{N}(\mu, \sigma^2)) \tag{4}$$

we rewrite it as

$$x = \mu + \sigma * sample(\mathcal{N}(0, 1)) \tag{5}$$

4.4 Gaussian Mixture Model

Gaussian Mixture Model(GMM) is a probabilistic generative model which assumes that all the data points are generated from a countable number of Gaussian distributions with unknown parameters. Considering a mixture of M Gaussians, the probability of a data point is calculated as

$$p(x_n) = \Sigma_m p(x_n|z_n = m)p(z_n = m)$$

$$= \Sigma_m \mathcal{N}(x_n|\mu_m, \Sigma_m)\pi_m \tag{6}$$

where μ_m, Σ_m are the mean and variance parameters for mth Gaussian distribution. π_m is the mixture proportion. z_n is considered a latent variable for cluster label. In a clustering problem using GMM, data points are assigned to the multivariate normal components that maximize the component posterior probability given the data.

5 Methodology

We approach the problem using two different methods. In the first method, we consider the nodes within the same cluster to have a high degree of a common neighborhood in $\mathcal{Y}$. In the second method, we are not assuming anything on the connectedness of the nodes in the cluster. The sections below provide an explanation for the models considered for the problem.

5.1 Bernoulli Mixture Model

This section introduces the idea of Bernoulli mixture model. It is a generative probabilistic model that models group structure in a network. This can be considered as a subclass of mixture models. The mixture of Bernoulli distribution is used to model binary random vectors [35]. We will be referring clusters as communities for better understanding in this scenario. The two types of nodes on this bipartite graphs are: *BTS* and *users*. The basic idea is that the edges between BTS and users are generated from a mixture of Bernoulli distributions.

With K communities, Bayesian mixture of Bernoulli distribution is modeled with B-dimensional binary input space (graph in the form of binary edges).

The algorithm for the generative process for each user node in the graph is explained as follows:

1. Draw distribution over communities, $\pi \sim Dirichlet_K(\alpha)$
2. Draw cluster membership for each user node, $\theta \sim Categorical(\pi)$
3. Draw distribution of clusters over the base stations, $\phi \sim Dirichlet(\beta)$
4. Draw link $r_{ij} \sim Bernoulli(p)$, where $p = \phi_\theta$

θ can be well explained as the distribution of category of the users and ϕ as clusters to which BTS belongs. We initialize α and β with K dimensional vector of ones.

Posterior inference is an essential computational problem for this model. It is about computing posterior distribution of latent variables given the observations. For given set of parameters α and β, the posterior is $p(\theta, \phi, \pi \,|\alpha, \beta, r)$. Analytically, computing the posterior is intractable, we resort to using MCMC sampling to do inference in the model.

The joint distribution of community membership π, cluster membership θ, cluster distribution over BTS ϕ and set of observed links is given by:

$$p(\theta, \phi, \pi, r \,|\alpha, \beta) = p(\pi \,|\alpha) \prod_{i=1}^{B} \prod_{j=1}^{U} p(r_{ij} \,|\phi, \theta) \prod_{n=1}^{K} p(\theta_n \,|\pi) \prod_{i=1}^{B} \prod_{k=1}^{K} p(\phi_{ik} \,|\beta) \quad (7)$$

There is a strong underlying assumption here that the probability of link between users and BTS within a cluster is high. Consider a user i belonging to community $\theta_i = k$, then for the cluster k, ϕ_k is probability distribution over all the BTS's.

By drawing from a Bernoulli distribution with parameter $p = \phi_\theta$ for $\phi = \phi_k$ and $\theta = \theta_i$, the model effectively says that in cluster θ_i, the user is linked to set of BTS's based on the distribution of $\{\phi_k\}_{\theta_i}$ i.e, over mixture of Bernoulli distribution.

Plate Notation

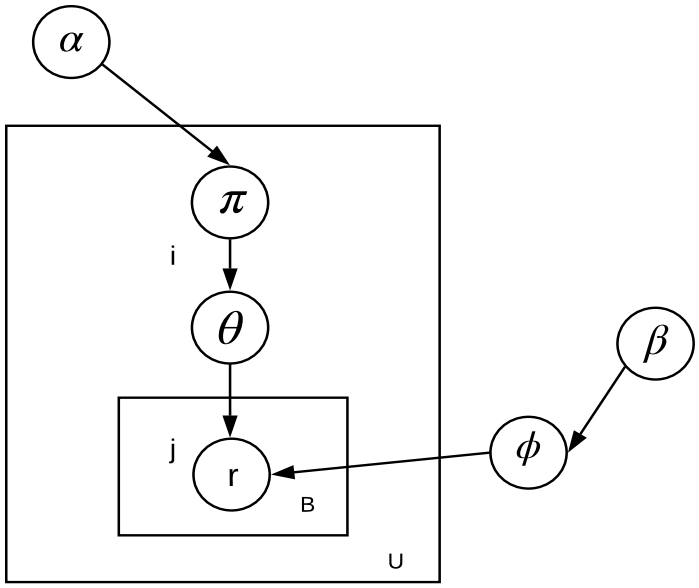


Figure 5: Generative model for mixture of Bernoulli distributions. The outer plate represents users, while the inner represents the link for BTS's with the users.

Graphical probabilistic model representation can be seen as in Fig 5. The hyperparameters α, β and the parameters θ, ϕ, π are the same as explained in the algorithm.

In order to find the estimates for the parameters, we sample the posterior using MCMC sampling. Once the posterior estimate converges we use estimation for the parameter ϕ to assign the cluster for the BTS. The clusters are assigned by taking the average of these samples. We use No-U-turn sampler [36] for sampling π and ϕ, while categorical Gibbs sampler [37] for θ.

5.2 Variational Autoencoder

VAE is a generative model [19] and works as explained in section 4.3. The original graph is used as the input to the model. The encoder tries to learn the latent variable which is Gaussian distributed with mean μ and variance Σ. The decoder samples from the given Gaussian estimating the likelihood of $p(\mathcal{G}|z)$, where z is the latent variable. While training the network, hidden layers are considered for the parameters μ and Σ . These parameters are used to sample from a Gaussian distribution which is used at the decoder to obtain the reconstructed input i.e, output.

During backpropagation in VAE, minimization of reconstruction loss and KL divergence of the posterior with a standard Normal distribution $\mathcal{N}(0, I)$ is considered. In our case, we add an extra loss term accounting for the hamming distance between output and input pairs. Hamming distance between two vectors is the number of coefficients in which they differ. The idea is to minimize the hamming loss between reconstructed and the original input pairs. The architecture of the VAE model is as shown in Fig 6.

The loss in our case would be

$$J = \underbrace{\sum_{k \in X} [x_k log(x'_k) + (1 - x_k) log(1 - x'_k)]}_{\text{cross entropy loss}} + \underbrace{\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) || \mathcal{N}(0, I)]}_{\text{KL divergence}} + \underbrace{\sum_{x_i, x_j \in X x'_i, x'_j \in X'} (hamming(x_i, x_j) - hamming(x'_i, x'_j))}_{\text{Hamming loss}} \tag{8}$$

VAE learns the latent space representation of the graph. We then throw away the decoder and use the learned encoded graph and fit GMM to this for clustering.

In Figure 6, the red boxes represent the error blocks associated with reconstruction and KL divergence. Hamming loss is added as shown in the block next to X′ . This is done to accommodate the error in hamming distance.

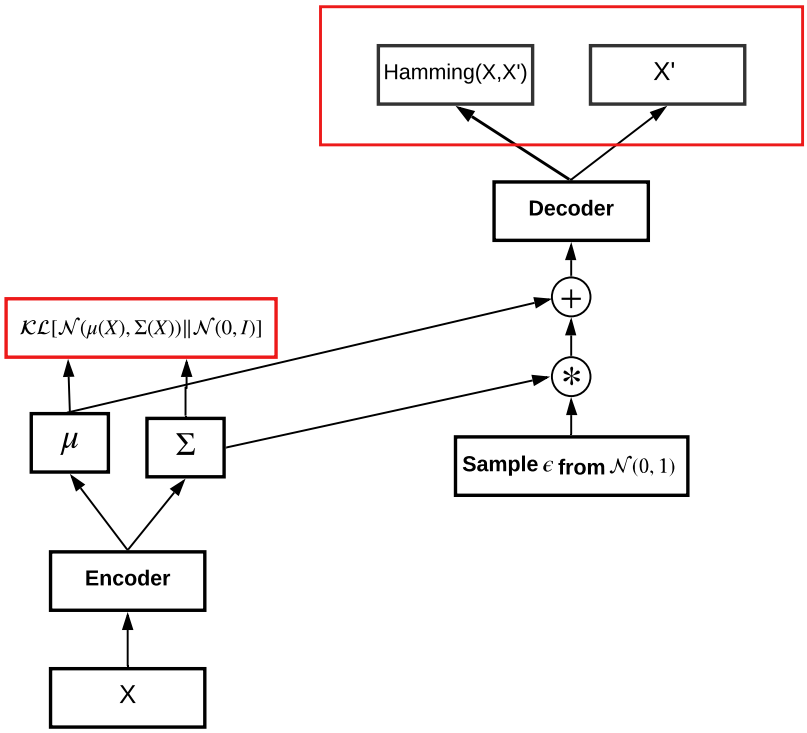


Figure 6: Block diagram of Variational Autoencoders

5.3 LDA

Latent Dirichlet Allocation [21] is a topic model used for discovering abstract topics from a set of documents. It is a generative probabilistic model for collections of the discrete dataset such as text corpora. Fig 7 shows the plate model using the below notations:

- M is the corpus of documents.
- Each document has N words
- There are K topics and z being the topic assigned.
- θ is the distribution of topics over the documents and β is the distribution of words over the topics.
- α is the parameter for the prior θ.
- The boxes in the model represent repeated sampling.
- w is a particular word in the document.

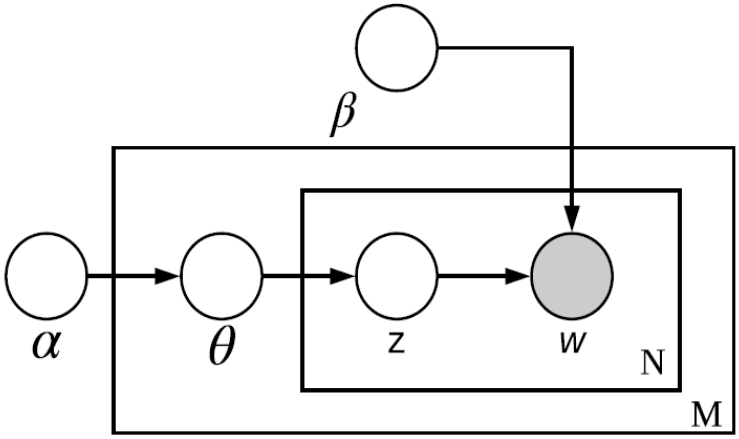


Figure 7: Graphical model representation of Latent Dirichlet Allocation reproduced from [21]

The base station list of a user can be thought of as a bag of words. Thus the clustering problem can be translated to the LDA model. A document is a user and it is characterized by a (unordered) sequence of BTSs $\mathbf{w} = (w_1, w_2, \ldots, w_N)$. A corpus is a collection of M users and is denoted by $\mathcal{D} = \{\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_M}\}$. A topic is a distribution over base stations which we identify as a cluster. We consider K topics (clusters).

Generative process can be explained as:

1. Sample a distribution over the clusters $\theta \sim \text{Dir}(\alpha)$
2. For each of the BTSs of the user:
    - sample a cluster $z_n \sim \text{Multinomial}(\theta)$
    - sample a BTS w_n from $p(w_n \mid z_n, \beta)$

6 Experiments and Results

6.1 Dataset description

The dataset used for the experiments in this project is from Ericsson's customers dated 26^{th} January 2017. It consists of IMSI(International Mobile Subscriber Identity) with the base stations(BTSs) assigned to it on the same date.

- Total number of base stations, $\mathcal{B}$: 87805
- Total number of users, $\mathcal{U}$: 283884
- Average number of users using a BTS: 152
- Maximum number of users using a BTS: 12467
- Minimum number of users using a BTS: 1

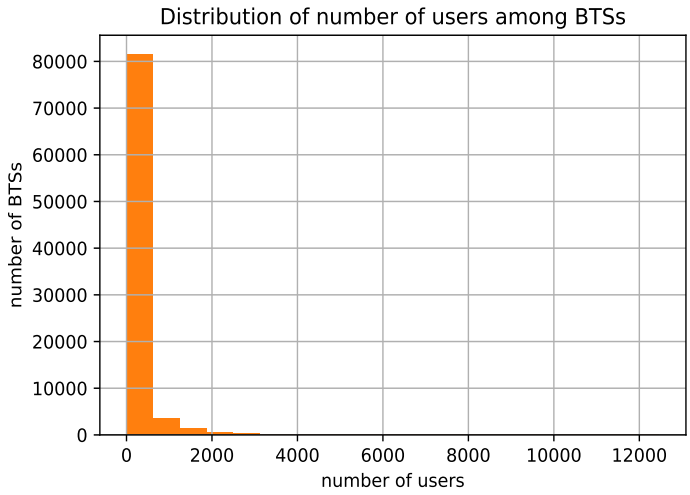


Figure 8: Distribution of number of users among BTSs

Figure 8 shows the distribution of users by the usage of BTSs. There is a small portion of users who make use of most of the base stations.

To use high usage BTS's from the complete dataset only those BTS's were selected which had the number of users above some minimum value. This minimum value was chosen as 2000 for Bernoulli mixture model and 100 for VAE and LDA. After filtering, the Bernoulli mixture model uses 1233 base stations to cluster while, VAE and LDA use 10,695. The next step involves transformation of dataset into a bi-adjacency matrix with $r \in [0, 1]$ link for user-BTS connection.

Metric used for the performance of the model

To evaluate the model, we propose a metric called "cluster utilization".

Cluster utilization : The formed clusters are analyzed based on its usage by having a measure on how the clusters are used among the users. Keeping this in mind, we calculate cluster utilization. The utilization for each cluster is calculated as below:

- Identify all the base stations in a cluster, B.
- Collect all the users using at least one base station in the cluster, U.
- Calculate the percentage of users u in U for using x% $\in [0, 100]$ base stations in the cluster.

So,

$$\kappa = u/U \tag{9}$$

For example, there are 10 BTS in a cluster and 7 users using at least one of those 10 BTS. To find what percentage of users make 50% utilization for the cluster, we check how many of those 7 users use 5 BTS (50% of 10). So $\kappa = 3/7 = 0.428$ (e.g 3 users out of 7 using 5 BTS).

This measure evaluates how efficiently the clusters are being utilized among the users. The tabular format of the results provided in this section is explained as follows:

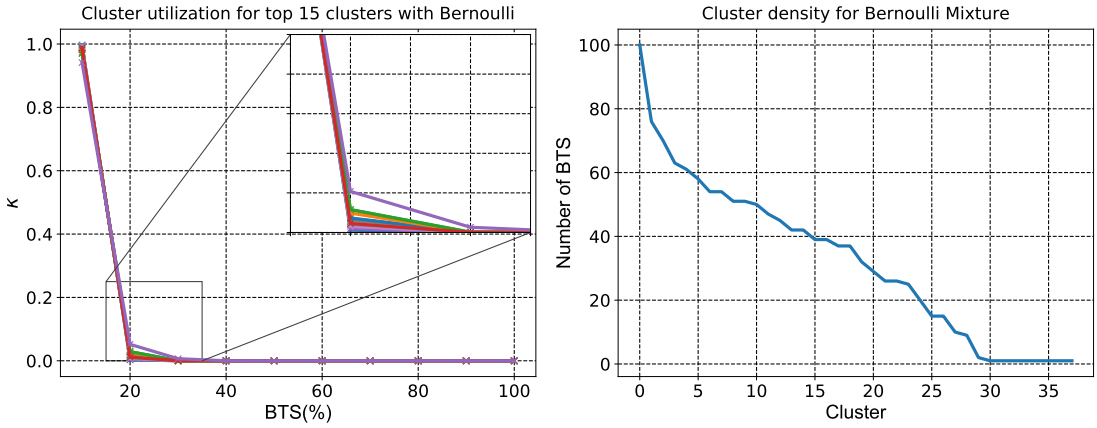
- For each cluster, every cell has users (in terms of %) who make use of the cluster.
- Analysis is done on a cumulative basis.

From these values of κ, we can infer the cluster utilization for each cluster. The column after which κ doesn't change by some small value δ is considered as the cluster utilization of that cluster.

6.2 Bernoulli Mixture Model

The data was preprocessed with minimum users per BTS as 2000. The number of clusters was chosen as 50. The hyperparameter values of α and β were initialized with K dimensional vector of ones. MCMC estimate for the posterior sampled using 2000 samples.

The plots show κ against BTS(in %) which provides a view of how BTS's in the cluster are being utilized. The plots in Fig 9 show that the number of BTS's in clusters is unevenly distributed. In Fig 9a, the utilization of 15 large clusters is captured. The utilization for most of these clusters is around 20%. However, it is slightly higher than 20% for a few clusters.



(a) Clusters Utilization (b) Number of BTS's in cluster

Figure 9: Results with Bernoulli mixture model

Table 1 shows the results for ultization of clusters. The value in each cell denotes users (in %) using the cluster. For example, referring to Table 1, in cluster 12, 77.95% of common users for the BTSs clustered make use of 10% ($\sim$ 1) or less of total base stations in that cluster. Then the next cell tells us that 99.32% users use 20%($\sim$ 3) or less base stations. This shows that utilization for this cluster is about 20%. Looking at table 1, we see that utilization is 30% for some clusters. However there is fairly smaller increase in the percentage of users using BTS's in those particular clusters.

PyMC3 package is used in implementing this model. It is a useful tool for Bayesian statistical modeling. The MCMC estimate for the posterior took approximately 1

day to converge with 2000 samples. Because of memory restrictions, the filtering criterion of minimum number of users was considered as 2000.

In table 1, columns K is the cluster number, BTS is the count of base stations in the corresponding cluster.

Table 1: Mixture of Bernoulli Cluster Utilization

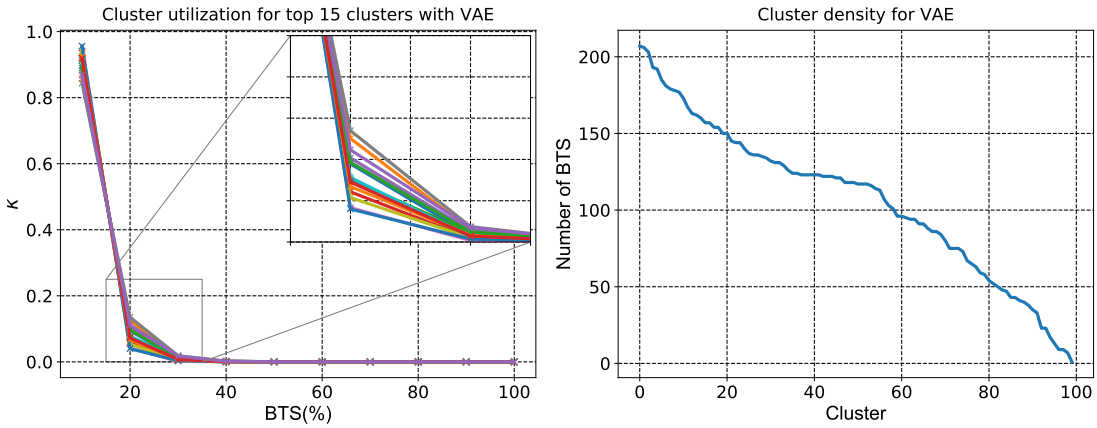
K	BTS	10.00%	20.00%	30.00%	40.00%	50.00%	60.00%	70.00%	80.00%	90.00%	100.00%
32	100	99.7	99.99	100	100	100	100	100	100	100	100
7	76	98.68	99.99	100	100	100	100	100	100	100	100
16	70	97.07	99.96	100	100	100	100	100	100	100	100
6	63	98.63	99.99	100	100	100	100	100	100	100	100
17	61	99.52	100	100	100	100	100	100	100	100	100
18	58	98.23	99.99	100	100	100	100	100	100	100	100
23	54	99.22	99.99	100	100	100	100	100	100	100	100
35	54	98.54	99.99	100	100	100	100	100	100	100	100
3	51	98.9	99.99	100	100	100	100	100	100	100	100
15	51	98.95	99.99	100	100	100	100	100	100	100	100
46	50	98.15	99.98	100	100	100	100	100	100	100	100
39	47	97.53	99.98	100	100	100	100	100	100	100	100
41	45	97.09	99.97	99.99	100	100	100	100	100	100	100
33	42	98.86	99.99	100	100	100	100	100	100	100	100
37	42	94.14	99.33	99.99	100	100	100	100	100	100	100
5	39	96.2	99.97	99.99	100	100	100	100	100	100	100
25	39	95.57	99.96	100	100	100	100	100	100	100	100
24	37	94.26	99.93	100	100	100	100	100	100	100	100
44	37	96.02	99.97	100	100	100	100	100	100	100	100
28	32	96.49	99.92	100	100	100	100	100	100	100	100
19	29	88.66	99.76	99.99	100	100	100	100	100	100	100
29	26	94.63	99.96	100	100	100	100	100	100	100	100
45	26	90.41	99.84	99.99	100	100	100	100	100	100	100
31	25	81.74	98.89	99.88	100	100	100	100	100	100	100
10	20	92.73	99.54	99.97	100	100	100	100	100	100	100
4	15	80.13	99.3	99.86	100	100	100	100	100	100	100
12	15	77.95	99.32	99.89	99.99	100	100	100	100	100	100
34	10	87.9	98.73	99.9	100	100	100	100	100	100	100
21	9	0	87.9	98.6	99.94	99.99	100	100	100	100	100
8	2	0	0	0	0	98.4	98.4	98.4	98.4	98.4	100
0	1	0	0	0	0	0	0	0	0	0	100
9	1	0	0	0	0	0	0	0	0	0	100
11	1	0	0	0	0	0	0	0	0	0	100
22	1	0	0	0	0	0	0	0	0	0	100
27	1	0	0	0	0	0	0	0	0	0	100
36	1	0	0	0	0	0	0	0	0	0	100
43	1	0	0	0	0	0	0	0	0	0	100
49	1	0	0	0	0	0	0	0	0	0	100

6.3 Variational Autoencoder

Using the variant of VAE with the hamming loss we make use of the encoded lower dimensional representation of the input graph. After fitting a GMM model to the embeddings of the graph, we assess the results. The number of clusters was chosen as 100. In Fig 10a, dense lines represent top 15 clusters (based on the number of base stations). The slope in the graph shows how the usage varies in those clusters. It is seen that the value gets very close to 0 after 30% so the utilization of the cluster is around 30%. This means that 30% of the cluster is in use among the common users. Also, Fig 10b shows that BTS's in many of these clusters are uniformly distributed. Table 2 reflects the results that the utilization overall is higher compared to the previous model.

In the training phase, VAE consists of fully connected encoder layer. It is followed by fully connected μ, σ and a decoder layer. The following settings were used for this experiment:

- Encoded dimensionality of 1740($\sim \frac{users}{20}$)
- Latent dimension of 250, batch size of 500
- Using Glorot weight initialization [38] and Adam optimizer
- Fully connected ReLU
- Ran for 50 epochs



(a) Clusters utilization (b) Number of BTS's in clusters

Figure 10: Results with VAE

The result table is presented in a similar way as done with the previous model.

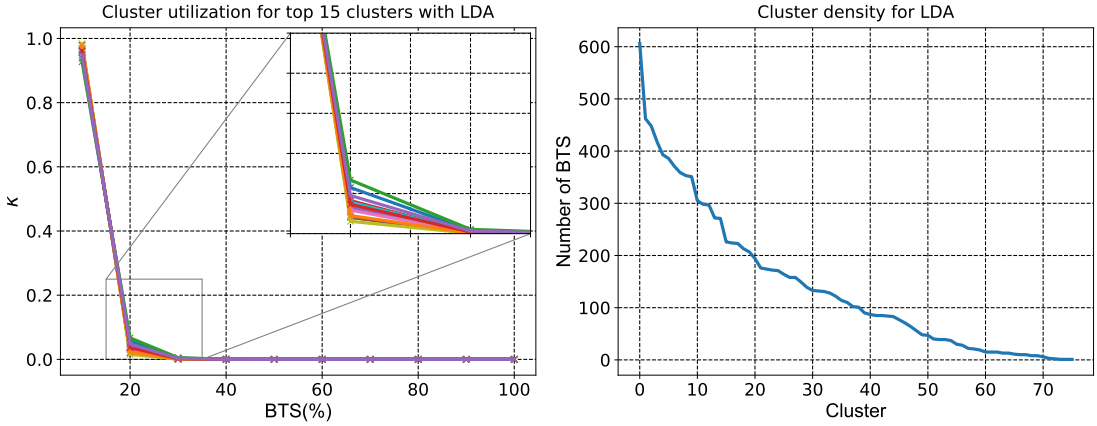
Table 2: Cluster utilization VAE with hamming loss

K	BTS	10.00%	20.00%	30.00%	40.00%	50.00%	60.00%	70.00%	80.00%	90.00%	100.00%
19	207	89.57	99.03	99.89	99.99	100	100	100	100	100	100
60	206	85.75	98.28	99.8	99.97	100	100	100	100	100	100
29	203	91.19	98.76	99.84	99.97	100	100	100	100	100	100
15	193	93.44	99.52	99.97	100	100	100	100	100	100	100
85	192	88.26	98.44	99.83	99.99	100	100	100	100	100	100
77	185	92	99.48	99.97	100	100	100	100	100	100	100
48	181	95.62	99.79	99.96	99.99	100	100	100	100	100	100
27	179	84.39	97.88	99.75	99.95	99.99	100	100	100	100	100
13	178	94.06	99.44	99.93	99.99	100	100	100	100	100	100
64	177	91.44	99.25	99.9	99.99	100	100	100	100	100	100
32	173	95.6	99.6	99.95	100	100	100	100	100	100	100
72	167	92.39	99.1	99.91	99.97	99.99	100	100	100	100	100
18	163	88.91	98.56	99.79	99.96	99.99	100	100	100	100	100
45	162	91.94	99.19	99.91	99.98	100	100	100	100	100	100
51	160	86.85	98.02	99.78	99.97	100	100	100	100	100	100
4	157	86.83	98.1	99.76	99.96	99.99	100	100	100	100	100
66	157	88.92	98.6	99.79	99.98	100	100	100	100	100	100
20	154	91.73	98.85	99.8	99.96	99.99	100	100	100	100	100
67	154	91.07	99.11	99.91	99.98	100	100	100	100	100	100
1	150	93.72	99.07	99.86	99.98	99.99	100	100	100	100	100
11	150	89.93	98.93	99.88	99.96	99.98	99.9	100	100	100	100
81	145	87.42	96.98	99.29	99.91	99.99	100	100	100	100	100
2	144	91.54	98.85	99.81	99.97	100	100	100	100	100	100
37	144	95.75	99.78	99.98	100	100	100	100	100	100	100
35	140	85.18	96.28	99.33	99.93	99.99	100	100	100	100	100
42	137	86.66	97.15	99.55	99.93	99.99	99.9	100	100	100	100
26	136	82.21	97.36	99.55	99.91	99.99	100	100	100	100	100
74	136	82.53	97.59	99.69	99.94	99.99	99.9	100	100	100	100
93	135	84.57	96.37	98.97	99.83	99.95	99.9	100	100	100	100
22	134	86.62	98.16	99.75	99.96	99.99	100	100	100	100	100
49	132	83.29	96.95	99.54	99.93	100	100	100	100	100	100
88	131	89.32	99.03	99.87	99.99	100	100	100	100	100	100
95	131	90.41	99.19	99.97	100	100	100	100	100	100	100
86	129	71.6	94.05	99.16	99.93	99.98	99.9	100	100	100	100
41	126	81.3	96.65	99.53	99.97	100	100	100	100	100	100
7	124	84.53	97.71	99.75	99.97	100	100	100	100	100	100
40	124	86.75	97.05	99.57	99.92	100	100	100	100	100	100
6	123	90.34	98.74	99.81	99.97	100	100	100	100	100	100
21	123	77.49	95.32	99.33	99.93	99.99	100	100	100	100	100
24	123	88.19	98.28	99.75	99.95	100	100	100	100	100	100
28	123	84.22	97.38	99.58	99.96	100	100	100	100	100	100
58	123	87.77	98.02	99.64	99.94	99.98	99.9	100	100	100	100
38	122	87.18	98.47	99.85	99.97	100	100	100	100	100	100
39	122	82.06	93.56	96.93	98.66	99.64	99.9	99.9	100	100	100
44	122	85.84	97.53	99.54	99.94	100	99.9	100	100	100	100
57	121	90.67	97.24	99.2	99.8	99.95	99.9	100	100	100	100
80	121	82.95	96.61	99.67	99.97	100	100	100	100	100	100
14	118	80.51	96.78	99.57	99.97	99.99	100	100	100	100	100
31	118	82.19	95.18	98.89	99.82	99.99	100	100	100	100	100

K	BTS	10.00%	20.00%	30.00%	40.00%	50.00%	60.00%	70.00%	80.00%	90.00%	100.00%
70	118	87.6	98.05	99.8	99.98	100	100	100	100	100	100
63	117	88.43	98.43	99.82	99.98	100	100	100	100	100	100
76	117	87.22	98.21	99.7	99.98	100	100	100	100	100	100
89	117	84.75	96.92	99.4	99.81	99.96	99.9	100	100	100	100
65	116	82.98	96.53	99.29	99.9	99.99	100	100	100	100	100
3	114	78.81	94.94	99.09	99.83	99.97	99.9	100	100	100	100
25	113	83.21	95.43	98.82	99.75	99.97	99.9	99.9	100	100	100
46	107	84.72	97.92	99.72	99.97	100	100	100	100	100	100
47	103	86.44	95.64	98.6	99.6	99.89	100	100	100	100	100
16	101	75.26	93.68	98.56	99.7	99.92	99.97	99.99	100	100	100
75	96	72.32	93.67	98.9	99.92	100	100	100	100	100	100
87	96	86.89	97.75	99.53	99.94	100	100	100	100	100	100
36	95	70.16	88.31	95.46	98.67	99.62	99.95	100	100	100	100
56	94	69.37	89.76	97.14	99.26	99.81	99.91	99.98	100	100	100
73	94	74.15	93.63	99.03	99.87	100	100	100	100	100	100
30	91	85.57	97.3	99.57	99.95	100	100	100	100	100	100
43	91	72.82	90.81	97.52	99.42	99.94	99.97	100	100	100	100
54	88	81.06	95.47	99.12	99.83	99.99	100	100	100	100	100
83	86	70.38	91.33	97.6	99.48	99.91	99.99	100	100	100	100
84	86	82.84	96.44	99.21	99.88	99.98	100	100	100	100	100
79	84	93.49	99.65	100	100	100	100	100	100	100	100
10	80	69.7	90.06	97.86	99.48	99.94	99.99	100	100	100	100
50	75	78.12	94.53	98.65	99.72	99.93	99.99	100	100	100	100
59	75	54.41	74.36	84.44	92.07	97.08	99.75	100	100	100	100
61	75	78.75	93.65	98.51	99.67	99.91	100	100	100	100	100
98	73	79.94	95.22	99.13	99.83	99.98	100	100	100	100	100
17	67	76.27	91.94	97.73	99.49	99.93	99.99	100	100	100	100
55	65	82.57	95.46	98.52	99.71	99.94	100	100	100	100	100
0	63	67.6	88.63	97.03	99.57	99.93	100	100	100	100	100
12	59	67.51	88.65	96.73	99.48	99.92	99.99	100	100	100	100
92	58	65.95	89.21	97.17	99.34	99.86	99.95	99.99	100	100	100
91	54	67.59	81.36	93.21	98.04	99.68	99.96	100	100	100	100
71	52	59.73	82.72	94.67	98.62	99.78	99.98	100	100	100	100
96	50	73.14	89.66	95.76	98.54	99.62	99.9	99.97	100	100	100
53	48	63.82	85.8	93.57	96.45	98.36	99.3	99.86	99.99	100	100
99	47	59.91	81.97	92.07	96.7	99.17	99.83	99.95	99.99	100	100
69	43	67.11	74.77	81.07	87.19	91.17	94.53	98.41	99.78	100	100
90	43	57.06	72.69	81.17	89.18	93.69	96.45	98.75	99.63	99.94	100
9	41	72.31	84.71	90.22	94.21	98.04	99.51	99.94	100	100	100
62	40	60.02	78.53	88.39	93.79	97.4	99.27	99.91	99.99	100	100
23	38	63.98	85.03	94.16	98.28	99.61	99.91	99.98	100	100	100
68	35	50.7	75.73	87.96	96.35	98.85	99.7	99.93	99.99	100	100
82	33	51.51	71.2	80.87	89.34	93.73	96.94	99.2	99.83	99.99	100
5	23	44.2	70.36	82.2	90.35	93.76	96.57	99.2	99.75	99.96	100
94	23	59.53	72.08	79.95	88.04	92.17	95.55	98.8	99.62	99.9	100
52	17	19.98	47.97	62.24	66.99	74.41	81.45	85.06	92.48	98.23	100
34	13	21.9	38.68	47.53	59.73	64.44	69.19	80.15	86.54	94.06	100
8	9	0	38.96	53.52	64.29	72.78	79.92	87.67	94.42	98.51	100
33	9	0	17.12	35.25	45.37	53.81	62.33	70.16	80.62	87.67	100
97	7	0	19.95	44.5	44.5	60.32	73.27	73.27	85.98	95.76	100
78	1	0	0	0	0	0	0	0	0	0	100

6.4 LDA

Topic modeling using LDA is used to assign topics i.e, clusters to the BTS. The number of clusters was chosen as 100 for this model.



(a) Clusters utilization

(b) Number of BTS's in cluster

Figure 11: Results with LDA

The plots in Fig 11 above shows the utilization and distribution of base stations in clusters. We see in Fig 11a that slope of the curve falls quickly. It also gives an impression that cluster utilization is a little over 20% for 15 of large clusters. Number of BTSs in clusters are distributed in a monotonically decreasing pattern. We have considered the LDA model as a baseline for this project. Table 3 reports κ for all the clusters. We use sklearn package for implementing LDA after preprocessing the dataset keeping the minimum number of users per BTS as 100.

Table 3: Cluster utilization LDA

K	BTS	10.00%	20.00%	30.00%	40.00%	50.00%	60.00%	70.00%	80.00%	90.00%	100.00%
98	606	96.7	99.86	100	100	100	100	100	100	100	100
64	462	97.9	99.93	100	100	100	100	100	100	100	100
97	448	96.61	99.9	100	100	100	100	100	100	100	100
61	418	95.69	99.81	99.99	100	100	100	100	100	100	100
82	393	96.59	99.84	100	100	100	100	100	100	100	100
55	386	97.89	99.93	100	100	100	100	100	100	100	100
47	371	96.94	99.87	99.99	100	100	100	100	100	100	100
45	359	96	99.87	99.98	100	100	100	100	100	100	100
22	353	98.41	99.95	100	100	100	100	100	100	100	100
59	351	95.86	99.78	99.99	100	100	100	100	100	100	100
52	305	93.82	99.57	99.97	100	100	100	100	100	100	100
7	298	97.66	99.92	100	100	100	100	100	100	100	100
29	297	92.74	99.43	99.97	100	100	100	100	100	100	100
71	272	96.13	99.78	99.99	100	100	100	100	100	100	100
95	271	94.9	99.66	99.99	100	100	100	100	100	100	100
58	226	93.09	98.69	99.77	99.97	100	100	100	100	100	100
83	224	95.76	99.74	99.99	100	100	100	100	100	100	100
72	223	97.19	99.89	99.99	100	100	100	100	100	100	100
91	213	88	98.67	99.9	100	100	100	100	100	100	100
35	207	93.35	99.49	99.95	99.99	100	100	100	100	100	100
5	194	91.45	99.27	99.95	100	100	100	100	100	100	100
51	176	90.08	99	99.92	99.99	100	100	100	100	100	100
99	174	88.25	98.82	99.9	99.99	100	100	100	100	100	100
86	172	90.72	99.36	99.95	100	100	100	100	100	100	100
24	171	88.48	98.52	99.87	99.99	100	100	100	100	100	100
0	164	89.37	98.62	99.85	99.99	100	100	100	100	100	100
27	158	83.95	96.8	99.76	99.99	100	100	100	100	100	100
77	158	80.05	96.87	99.62	99.96	99.99	100	100	100	100	100
4	149	88.79	97.62	99.56	99.94	100	100	100	100	100	100
12	139	87.46	95.56	98.06	99.28	99.85	99.97	100	100	100	100
60	133	83.02	96.56	99.47	99.9	99.99	100	100	100	100	100
41	132	89.94	98.92	99.95	100	100	100	100	100	100	100
11	131	92.88	99.57	99.99	100	100	100	100	100	100	100
69	128	86.08	96.55	99.16	99.8	99.95	99.99	100	100	100	100
43	122	80.01	95.44	99.13	99.86	99.99	100	100	100	100	100
32	114	82.81	96.03	99.38	99.92	100	100	100	100	100	100
8	110	88.72	98.45	99.8	99.99	100	100	100	100	100	100

K	BTS	10.00%	20.00%	30.00%	40.00%	50.00%	60.00%	70.00%	80.00%	90.00%	100.00%
14	102	93.53	99.08	99.89	99.97	99.99	100	100	100	100	100
13	101	88.41	98.13	99.75	99.92	99.98	99.99	100	100	100	100
2	90	96.48	99.63	99.96	99.99	100	100	100	100	100	100
25	87	79.96	93.7	98.26	99.51	99.93	99.98	99.99	100	100	100
23	85	82.75	96.77	99.34	99.9	99.99	100	100	100	100	100
50	85	77.41	93.32	98.79	99.91	100	100	100	100	100	100
18	84	82.31	94.17	98.34	99.45	99.88	99.99	100	100	100	100
84	83	93.07	98.98	99.93	99.99	100	100	100	100	100	100
63	77	89.14	97.64	99.35	99.89	99.95	100	100	100	100	100
9	71	54.78	71.97	82.59	90.26	95.72	99.32	99.93	100	100	100
78	64	84.71	97.15	99.73	99.97	100	100	100	100	100	100
46	56	87.9	98.56	99.71	99.96	100	100	100	100	100	100
42	48	91.78	98.7	99.81	99.95	99.98	100	100	100	100	100
53	47	85.73	96.69	99.08	99.7	99.86	99.94	99.97	99.98	100	100
88	40	75.69	90.61	96.36	98.73	99.67	99.9	99.96	100	100	100
75	39	73.58	89.32	95.84	98.68	99.59	99.94	99.99	100	100	100
76	39	88.95	98.64	99.89	100	100	100	100	100	100	100
33	37	63.55	90.97	97.89	99.38	99.87	99.99	100	100	100	100
81	30	74.62	88.41	95.02	98.14	99.3	99.87	99.98	100	100	100
74	28	66.18	89.67	96.5	99.03	99.81	99.84	99.93	99.99	100	100
21	22	66.76	83.93	90.84	94.57	97.75	98.91	99.63	99.89	99.98	100
66	21	77.62	92.17	96.92	98.89	99.76	99.96	100	100	100	100
36	19	53.78	83.97	93.87	98.12	99.59	99.9	100	100	100	100
3	15	79.09	98.63	99.81	100	100	100	100	100	100	100
48	15	42.18	86.24	91.51	96.42	98.39	99.61	99.92	100	100	100
79	15	38.73	75.74	83.67	93.51	95.85	98.61	99.23	99.83	99.93	100
37	13	57.98	76.47	86.98	95.27	97.46	98.5	99.79	99.91	100	100
93	13	68.31	87.45	95.17	99.44	99.82	99.96	100	100	100	100
16	11	52.46	74.06	85.8	92.17	95.87	98.34	99.22	99.81	100	100
67	10	43.24	56.31	64.53	69.9	75.34	82.01	88.48	94.24	97.73	100
80	10	66.41	87.58	96.88	99.36	99.77	99.96	99.97	100	100	100
34	8	0	66.93	89.69	96.98	99.32	99.32	99.76	99.96	100	100
62	8	0	21.83	43.71	61.94	75.43	75.43	88.96	96.36	99.32	100
57	6	0	56.9	56.9	78.83	90.11	90.11	96.61	96.61	99.52	100
68	3	0	0	0	87.65	87.65	87.65	98.37	98.37	98.37	100
44	2	0	0	0	0	99	99	99	99	99	100
10	1	0	0	0	0	0	0	0	0	0	100
15	1	0	0	0	0	0	0	0	0	0	100
89	1	0	0	0	0	0	0	0	0	0	100

6.5 Summarizing the results

To summarize the results, Table 4 shows the number of clusters with utilization for all the different models implemented for this project. Autoencoder and the VAE [19] are implemented in the same setting as the model described in section 4.3 to make the results comparable. It is evident that the clusters formed on the encoded graph using a variant of VAE with GMM gave better utilization compared to other models from the table below. Cluster utilization was one of the requirements specified by Ericsson to assess the models. However, I also looked into the silhouette coefficient as an assessment criterion.

Table 4: Summary of results

Model	Number of clusters with $\geq 30\%$ cluster utilization	Number of Clusters considered, K
Bernoulli Mixture	11	50
LDA	67	100
Autoencoder	65	100
VAE	46	100
VAE with hamming loss	**84**	**100**

Silhouette coefficient: It is a measure used to assess the similarity between nodes in a cluster [26]. It measures the separateness of the clusters. Coefficient value is in the range -1 and 1. Silhouette coefficient is calculated using the formula,

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

where b is the smallest average cluster distance of i to all nodes in neighboring cluster. and a is the average intracluster distance from each node. The coefficient value of 1 describes that the clusters are distinct. The higher the silhouette coefficient it is better. A value close to 0 indicates that the clusters formed are overlapping. Table 5 shows the coefficient values for the models implemented. The values for all the models are very small and negative. The variant of VAE has a better coefficient value among all the models. In comparison, other models have a lower coefficient value which gives the impression that the number of the nodes

that are wrongly classified is high in those models. The clusters formed by all the models are seen overlapping. Using these values it is difficult to say with conviction that any of these models works best for the given dataset. However, the variant of VAE can be considered to give comparatively better clusters in terms of number of nodes which might be wrongly classified.

Table 5: Silhouette Coefficient

Model	Silhouette Coefficient
Bernoulli Mixture	-0.128
LDA	-0.162
Autoencoder	-0.117
VAE	-0.091
VAE with hamming loss	**-0.086**

7 Summary and Conclusion

In this project, we implemented clustering techniques on graph networks using two new models. We have proposed a novel algorithm based on Bernoulli mixtures and developed a variant of VAE with a loss function using hamming distance. We evaluate our model for unsupervised clustering tasks using a specific dataset from Ericsson. These models helped in the exploratory analysis of the data used in the project for clustering base stations. The majority of the work was split between the two models. Initial results with Bernoulli Mixture Model were not satisfactory so we looked into VAE, which gave slightly better results.

Classical approaches in the literature may use maximum likelihood methods to estimate the parameters of the Bernoulli Mixture model and capture the most-likely clusters of the graph. The Bayesian approach provides an added flexibility in their ability to incorporate prior information and prevent overfitting but faces computational difficulties due to potentially slow MCMC convergence.

It is observed from the results in Table 4 and 5 that the addition of hamming loss to VAE's reconstruction and KL divergence loss impacts clustering in a significant manner. This can also help us lead to other loss functions which can prove useful for future work. One of the advantages of using a generative model like VAE is that it helps in exploring the distribution and modeling data that fits a well-studied machine learning model like GMM in this case. It is also crucial to address optimization challenges associated with VAE's with the increasing number of layers. The results obtained from the models implemented in the project show that it could prove useful on other datasets of similar nature. In the future, we can look into other metrics for finding the relation between the vectors. We could also collect useful additional feature like the amount of time a user is connected to a base station to incorporate in the model. It would also be an interesting task to find how these models work on different graph dataset.

References

[1] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.

[2] M. E. Celebi and K. Aydin, *Unsupervised learning algorithms*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, July 2009.

[4] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data." *PLoS ONE*, vol. 11, no. 4, pp. 1–31, Apr. 2016.

[5] M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, Minneapolis, USA, June 2007.

[6] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. Int. Conf. Comput. Vision and Pattern Recognition*, Portland, USA, June 2013.

[7] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?" in *Proc. Int. Conf. Database Theory*, London, UK, Jan. 1999.

[8] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *J. Amer. Statist. Assoc.*, vol. 97, no. 458, pp. 611–631, June 2002.

[9] S. Tiruveedhula, C. Sheela Rani, and K. Narayana, "A survey on clustering techniques for big data mining," *Indian J. Sci. and Technol.*, vol. 9, no. 3, pp. 1–12, Feb. 2016.

[10] D. J. Bora and A. K. Gupta, "A comparative study between fuzzy clustering algorithm and hard clustering algorithm," *Int. J. Comput. Trends and Technol.*, vol. 10, no. 2, pp. 108–113, Apr. 2014.

[11] S. Rajagopal, "Customer data clustering using data mining technique," *Int. J. Database Manage. Syst.*, vol. 3, no. 4, June 2011.

[12] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 1, pp. 52–74, Sept. 2017.

[13] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proc. ACM Int. Conf. Web Search and Data Mining*, New York, USA, Feb. 2011.

[14] D. B. Larremore, A. Clauset, and A. Z. Jacobs, "Efficiently inferring community structure in bipartite networks," *Phys. Review. E, Statist., nonlinear, and soft matter physics*, vol. 90, no. 1, p. 012805, July 2014.

[15] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proc. Int. ACM SIGIR Conf. Res. and Development in Inform. Retrieval*, New York, USA, July 2011.

[16] M. E. J. Newman, "Clustering and preferential attachment in growing networks," *Phys. Review E*, vol. 64, no. 2, p. 025102, Aug. 2001.

[17] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," in *Proc. Advances in Neural Inform. Process. Syst.*, Vancouver,Canada, Dec. 2009.

[18] P. K. Gopalan, S. Gerrish, M. Freedman, D. M. Blei, and D. M. Mimno, "Scalable inference of overlapping communities," in *Proc. Advances in Neural Inform. Process. Syst.*, Stateline, USA, Dec. 2012.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learning Representations*, Banff, Canada, Apr. 2014.

[20] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop Bayesian Deep Learning*, Barcelona, Spain, Dec. 2016.

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learning Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[22] P. Baldi, "Autoencoders, unsupervised learning and deep architectures," in *Proc. Int. Conf. Unsupervised and Transfer Learning Workshop*, Washington, USA, Apr. 2011.

[23] Y. Yamanishi, "Supervised bipartite graph inference," in *Proc. Int. Conf. Neural Inform. Process. Syst.*, Vancouver, Canada, Dec. 2008.

[24] T. Zhou, J. Ren, M. c. v. Medo, and Y.-C. Zhang, "Bipartite network projection and personal recommendation," *Phys. Rev. E*, vol. 76, no. 4, p. 046115, Oct. 2007.

[25] T. P. Peixoto, "Parsimonious module inference in large networks," *Phys. Review Letters*, vol. 110, no. 14, p. 148701, Apr. 2013.

[26] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, pp. 53–65, Nov. 1987.

[27] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, Jan. 2010.

[28] J. D. Ser, J. L. Lobo, E. Villar-Rodriguez, M. N. Bilbao, and C. Perfecto, "Community detection in graphs based on surprise maximization using firefly heuristics," in *Proc. IEEE Congr. Evolutionary Computation*, Vancouver, Canada, July 2016.

[29] J. Wyse, N. Friel, and P. Latouche, "Inferring structure in bipartite networks using the latent blockmodel and exact ICL," *Network Sci.*, vol. 5, no. 1, p. 45–69, Feb. 2017.

[30] Z. S. Razaee, A. A. Amini, and J. J. Li, "Matched bipartite block model with covariates," *CoRR*, vol. abs/1703.04943, Mar. 2017.

[31] B. Kiran, D. Thomas, and R. Parakkal, "An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos," *J. Imaging*, vol. 4, no. 2, p. 36, Feb. 2018.

[32] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Advances in Neural Inform. Process. Syst.*, Montreal, 2014, pp. 3581–3589.

[33] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Proc. Advances in Neural Inform. Process. Syst.*, Montreal, 2015, pp. 2539–2547.

[34] C. Doersch, "Tutorial on variational autoencoders," *CoRR*, vol. abs/1606.05908, June 2016.

[35] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, Mar. 1994.

[36] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo." *J. Mach. Learning Res.*, vol. 15, no. 1, pp. 1593–1623, Jan. 2014.

[37] A. F. Smith and G. O. Roberts, "Bayesian computation via the gibbs sampler and related markov chain monte carlo methods," *J. Royal Statist. Soc.*, vol. 55, no. 1, pp. 3–23, Jan. 1993.

[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *AISTATS*, Sardinia, 2010, pp. 249–256.