Community Detection in Social Networks using Deep Learning

A Dissertation Submitted to the University of Hyderabad
in Partial Fulfillment of the Degree of

Master of Technology
in
Artificial Intelligence

by

Dhilber M
17MCMI15

School of Computer and Information Sciences
University of Hyderabad
Gachibowli, Hyderabad - 500 046
Telangana, India

June 25, 2019

CERTIFICATE

This is to certify that the dissertation titled, "**Community Detection in Social Networks using Deep Learning**" submitted by **Dhilber M**, bearing Regd. No. **17MCMI15**, in partial fulfillment of the requirements for the award of Master of Technology in Artificial Intelligence is a bonafide work carried out by him under my supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. S Durga Bhavani	**Dr. K. Narayana Murthy**
Project Supervisor	Dean
School of Computer and	School of Computer and
Information Sciences	Information Sciences
University of Hyderabad.	University of Hyderabad.

DECLARATION

I, **Dhilber M**, hereby declare that this dissertation titled, "**Community Detection in Social Networks using Deep Learning**", submitted by me under the guidance and supervision of Dr. S Durga Bhavani is a bonafide work which is also free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma. I hereby agree that my dissertation can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Librarian is enclosed.

Date: **Name:** Dhilber M

Regd. No. 17MCMI15

//Countersigned//

Signature of the Supervisor(s):

(Dr. S Durga Bhavani)

Acknowledgements

It gives me a great pleasure to thank all the people who have helped me in making this project a real learning experience.

First and foremost, I would like to thank and express my heartfelt gratitude towards Dr. S Durga Bhavani, for her valuable guidance. Without her excellent guidance, this project would not have been possible. It was a real privilege and great learning to work with her.

Also, I would like to thank **School of Computer and Information Sciences, University of Hyderabad** for providing excellent lab facilities.

Last but not the least, I would like to express my heartfelt wishes to my beloved parents and my dear friends whose gracious solicitude helped me to complete this project successfully.

Dhilber M

Abstract

Community structure is found everywhere from simple networks to real world complex networks. Detecting communities in such networks is always a challenging problem in the area of network theory. The task of community detection has a wide variety of applications ranging from recommendation systems, advertising, marketing, epidemic spreading, cancer detection etc. We propose a deep learning architecture to be applied for the community detection problem on unipartite networks and as an extension we applied it's modified version on bipartite networks. For unipartite networks, two mainly existing approaches for community detection, namely stochastic model and modularity maximization model focus on building a low dimensional network embedding to reconstruct the original network structure. However the mapping to low dimensional space in these methods is purely linear. Understanding the fact that real world networks will have non-linear structure in abundance, aforementioned methods become less practical for real world networks.

Considering the non-linear representation power of deep neural networks, several solutions based on autoencoders are being proposed. In our work for unipartite networks we propose a new method wherein we stack multiple autoencoders and apply parameter sharing. This method of training autoencoders has been successfully applied for problems of link prediction and node classification in recent literature. For bipartite networks we adopt an existing representation model specifically built for them. We used those representations and applied our autoencoder approach on them. Our enhanced model with modified architecture for unipartite network produced better results compared to many other existing methods, while our approach for bipartite networks is not showing satisfactory results which is to be further investigated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Community detection is the process of identifying densely interacting vertices in a network based on its structural properties. Detecting community structures in networks is a vital task in network theory. There have been a fair amount of approaches in achieving this task in the literature [9, 28]. In this increasingly interlinked world, studying and analyzing relationships and patterns in networks is becoming inevitable.

Classical approaches in community detection use traditional statistics to extract network information. Newman [25] proposed a measure called modularity which indicates the difference between the number of edges within communities and expected number of edges in overall network. This is a widely accepted measure and studied deeply in literature. He himself reformulates modularity measure based on eigen vectors to form spectral algorithm. Another useful measure considered for community detection is betweenness centrality. In this method, edges are iteratively removed based on one of the betweenness measures. In short these classical methods use network statistics measures like centrality, connectivity etc to find communities.

Recently there has been a lot of work that incorporates non-linear capabilities of deep neural networks [7] in achieving community detection and related tasks [31, 30, 15]. Perozzi et al [26] use random walks to learn embedding by considering random walks as equivalent to sentences for language representation problem. Our primary work on unipartite networks is closely related to [31], a new method of community detection in which network embeddings are used to detect communities. However we followed a different architecture and training scheme for deep neural network which gave us better results. For bipartite network, we adopted (BiNE) for network representation, which uses random walks specifically devised for bipartite networks.

Node embeddings can also be applied for other tasks like link prediction and node classification, which has wide applications. For example link prediction can be used to predict interaction between molecules in a chemical compound network or friends in a friendship network and node classification can be used to identify the importance of individuals in a social network. We stick to the problem of community detection. It also covers major applications which include, detecting suspicious groups in a network, recommending products or services, targeted marketing, epidemic spread control, cancer detection etc.

Remaining of this dissertation is arranged as follows: In Chapter 2 we discuss related literature and similar works. In Chapter 3 we go in detail on unipartite networks followed by bipartite networks in Chapter 4 and finally conclude in Chapter 5.

Chapter 2

Background

Machine learning on graphs is one of the pivotal work that has applications ranging to multiple domains. The interconnectivity of nodes in graphs, that has great resemblance with real world makes graphs ubiquitous in several applications. One of the primary challenges in this field is to transform graphs to formats which can be then directly used for downstream machine learning tasks. Based on this, there are several approaches including Matrix Factorization based, Random Walk based and Graph Neural Networks. The classical methods use traditional network statistical measures like centrality, closeness etc and network embedding based methods use non-linear algorithms like deep neural networks.

Classical approaches for community detection are mostly based on 'modularity' measure proposed by Newman. Blondel et al [4] proposed a method which is an optimization of modularity and gives good results on large networks. Newman himself proposed modularity based algorithm called Spectral Algorithm [8], which is considered as best among classic methods.

2.1 Embedding Based Methods

Recently there is a hype in network embedding based approaches in solving community detection and related problems. In the following sections we briefly review two of the main works, Deepwalk and node2vec.

2.1.1 Deepwalk

Advent of Deepwalk in 2014 has given a new face to graph representation learning. In Deepwalk, Perozzi et al compare word sequences to random walks from a vertex. These random walks are used to learn network topology information. This inspired a lot of other works later. This was a key work that joined language modelling and vertex representation. They model small random walks in graph vertices to learn representation. These learned representations include neighbourhood information, community membership and vertex resemblence. This information is encoded into a continuous vector which can be later used for multiple machine learning tasks. In short Deepwalk model takes a network as input and gives a representation vector with its information encoded. There are multiple advantages in using random walks. It gives the flexibility to explore different parts of the network and makes the model more adaptive to small changes without overall recomputation. Deepwalk uses skipgram as language model to calculate occurence probability of words within a frame. Algorithm is also scalable to large dataset and performs exceptionally in sparse datasets.

2.1.2 Node2vec

An improved approach of Deepwalk called Node2vec [13] implements a biased random walk which explores diverse neighborhoods. They claim that a much richer representation can be learned by making the random walks biased. The key idea behind node2vec is learn topological structure based on vertex network neighborhood. This is achieved by carefully organizing a bunch of biased random walks. This provides the flexibility of exploring diverse neighborhoods which in turn results in better representation.

Kipf et al [20] apply convolutional neural networks(CNN) directly on graph structured data to encode features. Jia et al [17] make use of Generative Adversarial Networks to obtain representations with membership strength of vertices in communities and solves overlapping community detection problem. Phi et al [30] build autoencoders with tied weights for multitask learning of link prediction and node classification. Hamilton et al [15] introduced an inductive framework to generate node embedding for unseen networks by modelling a neighborhood aggregation function. Our work on unipartite networks is closely related to [31] which uses autoencoders to obtain latent representations. We used representation of BiNE [11] for bipartite work, which will be explained in detail in Chapter 4.

Another task in this area is overlapping community detection, which is not well explored in literature. The multiple community membership of nodes makes the problem a bit complex to solve. Despite that, [28] tries to solve overlapping community detection problem by using Bayesian Nonnegative Matrix factorization problem. There were always challenges in defining communities in bipartite networks, as a result solutions particular to bipartite networks are limited. We will discuss related works in bipartite networks in section 4.1.

Chapter 3

Unipartite Networks

In this Chapter, we go in depth with unipartite networks. Unipartite networks are graphs with homogeneous types of nodes. They are the most common types of graphs generally considered for studies in network theory. Since all the vertices are of same type, any algorithm applied will in effect reaches the whole network. In addition, common graph properties like centrality, connectivity etc can be applied and information can be extracted. Heterogeneous network will have great restrictions on all these mentioned factors.

In the following sections, we first define the problem statement and discuss Liang's solution. Then we go in detail with our proposed method and experiments.

3.1 Community Detection Problem

Solutions to the problem of community detection can be broadly classified into Modularity maximization model [9] and Stochastic model [16, 19, 7]. Modularity maximization model introduced by Newman in 2006 to maximize the modularity function Q, where modularity Q is defined as the difference between the number of edges within community and the expected number of edges over all pairs of vertices. Let a network $G = (V, E)$ be given with V being set of vertices and E the set of edges between pairs of vertices with —V— = N and —E— = m. G is represented as an adjacency matrix A with $A(i, j) = 1$ if there is an edge between

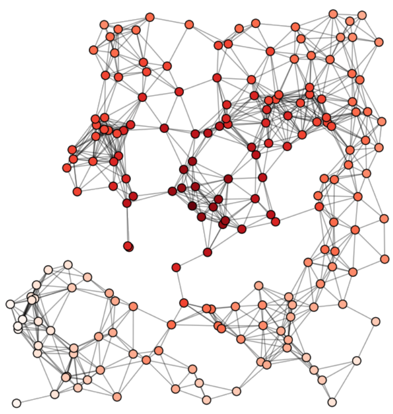

Figure 3.1: Unipartite network example (source: networkx.github.io).

vertices i and j and 0 otherwise.

$$\begin{aligned} Q &= \frac{1}{4m}\sum_{i,j}(A(i,j) - \frac{k_i k_j}{2m})(h_i h_j) \\ &= \frac{1}{4m} h^T B h \end{aligned}$$

Here h is the community membership vector with $h_i = 1$ or -1 to denote the two different communities, k_i denotes the degree of node i, B is the modularity matrix and m is the total number of edges in the network.

In Stochastic model, community detection is formulated as Non Negative Matrix Factorization problem (NMF). This approach aims to find a non-negative membership matrix H to reconstruct adjacency matrix A. Liang et al [31] proved that both Modularity maximization and Stochastic model can be interpreted as finding low dimensional representations to best reconstruct new structure. They further investigate that mapping of networks to lower dimensions is purely linear, which makes them less practical for real world networks and hence propose a non-linear solution using autoencoders [7].

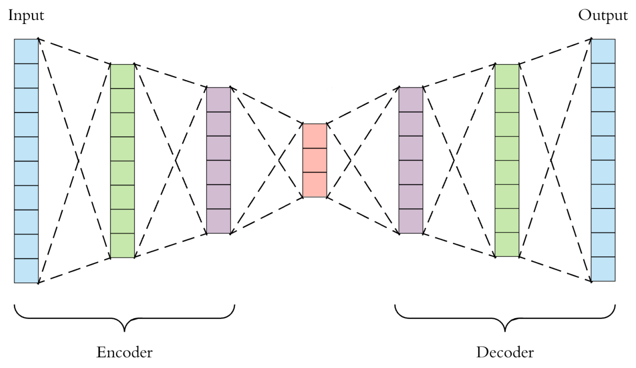

Figure 3.2: Autoencoder architecture (source: towardsdatascience.com).

3.2 Reconstruction Model

The basic idea adopted in the literature [30] is to substitute factorization based reconstruction methods with non-linear models like autoencoders. This approach solves the problem of non-linearity. The underlying nonlinear information hidden in real world networks can be preserved. In the following sections we introduce autoencoder and then go in detail with the models.

3.2.1 Autoencoder

An autoencoder is a deep neural network architecture composed of encoder and decoder. The encoder part compresses the data to a lower dimension trying to preserve maximum information. On the other side decoder expands the compressed code to reproduce output with maximum similarity as input. The main application of autoencoders is in dimensionality reduction and embedding learning. Here we use autoencoders as our non-linear reconstruction model. Figure 3.2 depicts a typical autoencoder architecture.

Autoencoders use backpropagation algorithm to learn the weights. For reconstruction, target values are put as same as input values and backpropagates the error. Considering input modularity matrix B = $[b_{ij}]$, b_i will represent vertex at column i. At encoder side for the encoded embedding E we have,

$$e_i = a(W_E b_i + d_E)$$

where W_E and d_E are the neural network weights(parameters) to be learnt, b_i

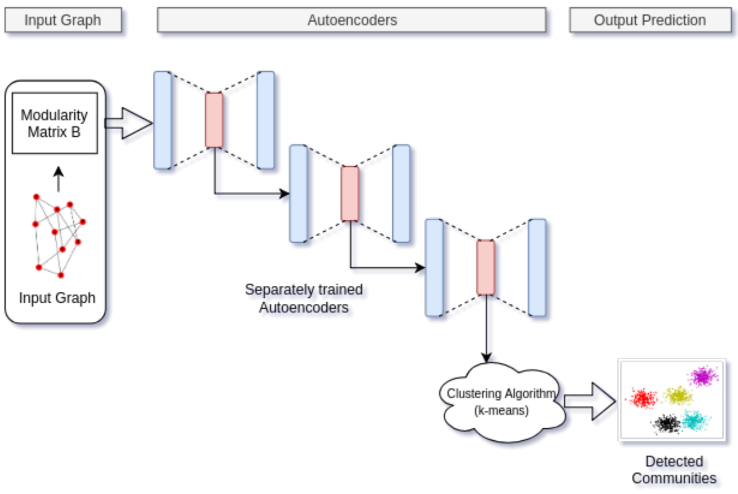

Figure 3.3: Liang's proposed model.

is the input and a is the activation function.

At the decoder side for the decoded embedding D we have,

$$d_i = a(W_D e_i + d_D)$$

where W_D and d_D are the parameters, e_i is the encoded input and a is another activation function.

3.3 Liang's Model

Idea of reconstruction in both modularity maximization model and stochastic model was well explored by Liang et al. They chose autoencoder as a substitute tool for reconstruction mechanism required here. Liang et al [31] are among the first researchers to apply deep learning(DL) approach to the community detection problem. They construct a DL model by connecting autoencoders in series to obtain parameter optimization. They train the first autoencoder to reduce reconstruction error, take new representation obtained from the first autoencoder and give it to the next one and so on. Figure 3.3 depicts layer by layer training scheme model of Liang et al. Inorder to obtain communities in the final step, K-means algorithm is used to cluster the vertex embeddings.

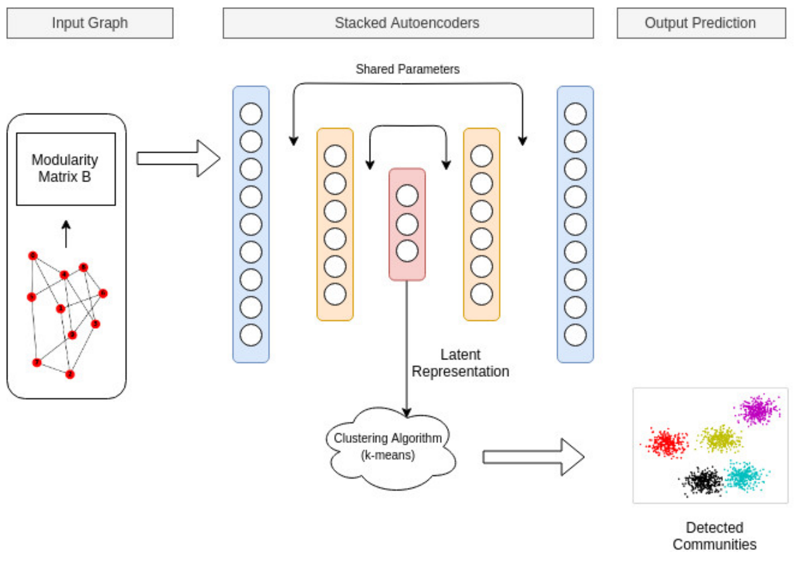

Figure 3.4: Architecture of the proposed model.

3.4 Proposed Model

We follow a similar approach as Liang at al described above. The key differences between our model and [31] are, first, we stack the autoencoder layers together, not in series and perform common training to all the layers instead of training each autoencoder separately. Secondly, we apply parameter sharing across layers to control the parameter growth and obtain optimization. This method of weight tying was successfully applied by [30] in solving link prediction and node classification problems simultaneously.

3.4.1 Method

The method of [31] with minor changes is implemented which is discussed in this section. First an autoencoder model with a common 2 layer architecture is built with varied layer configurations depending on datasets. We train the autoencoder for multiple iterations to reduce the reconstruction error. The modularity matrix of each network is given as input to the autoencoder and a low dimensional representation of the corresponding network is obtained. Community detection is carried out by applying clustering algorithms (K-means in this case) to the obtained representations. The intuition behind such an idea is, since we could reconstruct the entire network with this low dimensional representation, any downstream machine learning task applied to this low dimensional representation is equivalent in applying to the entire network.

3.4.2 Experimental setup

For implementation[1] we used python 3.7 and chose keras as deep learning library. We used keras custom layer feature to build layers capable of weight tying. We considered *adam* as optimizer and *relu* as activation function. As loss function we chose *sigmoid-cross entropy* and added 20% dropout to each layer. We trained each autoencoder to atmost 50,000 iterations to reduce the reconstruction error. For each network, a layer configuration that fits in 2 layer architecture has been chosen, which in turn gave flexibility of adopting our model to networks of different ranges. For a particular network, a one-step training of stacked layer autoencoder is done instead of layerwise training of multiple autoencoders as done by [31]. This in fact reduces training time and effort. Figure 3.5 depicts the gradual decrease of loss with respect to the number of epochs for all datasets except *Polblogs*, in which the loss decreases exponentially.

3.4.3 Evaluation Measure

Normalized Mutual Information(NMI) is used as the evaluation measure here. NMI is the quality measure value between 0 and 1 which indicates how much two clusters are correlated. A 0 value indicates no mutual information and 1 indicates perfect correlation. NMI is defined as,

$$NMI(Y,C) = \frac{2 \times I(Y;C)}{H(Y) + H(C)}$$

where Y is the class label, C is the clustering label, H() is the entropy and I(Y;C) is the mutual information between Y and C. In our case, each of the output cluster of our model is compared with ground truth value of the network.

[1]https://github.com/dilberdillu/community-detection-DL

Dataset	Layer Configuration	N	m	K	SP	FUA	FN	DNR	This Model
Polbooks	105-64-32	105	441	3	0.561	0.574	0.531	0.582	**0.600**
Polblogs	1490-256-128	1490	16718	2	0.511	0.375	0.499	0.517	**0.533**
Dolphin	62-32-16	62	159	4	0.753	0.516	0.572	0.818	**0.830**
Football	115-64-32	115	613	12	0.334	0.890	0.698	**0.914**	0.904

Table 3.1: Normalized Mutual Information of models on real world networks

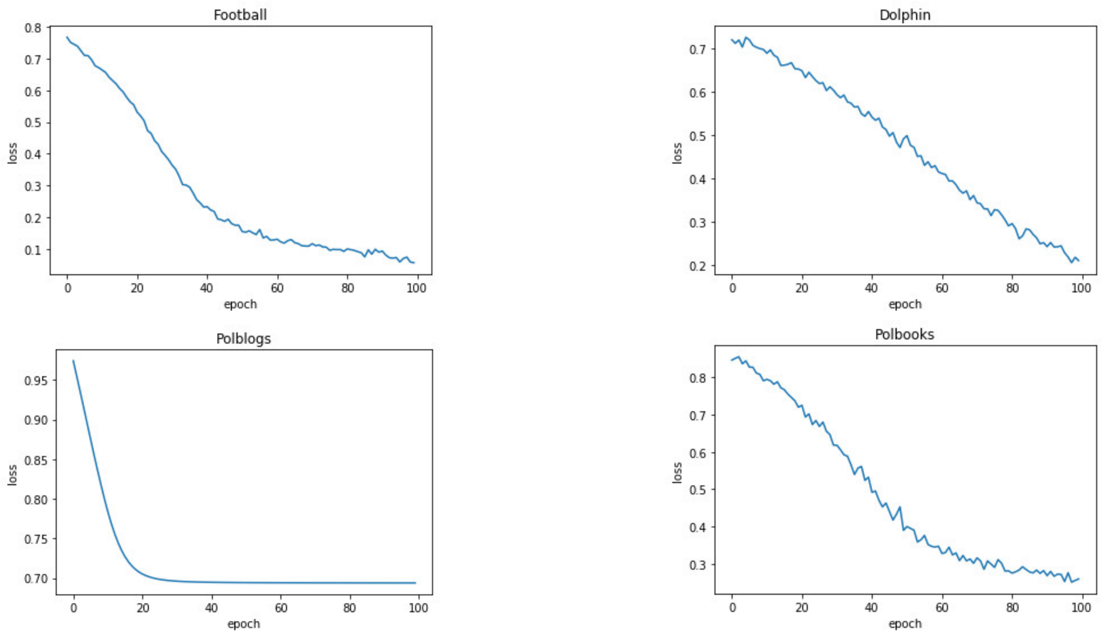

Figure 3.5: Comparison of loss with respect to the number of epochs for different datasets

3.5 Results and Evaluation

We compared our model with DNR [31], a deep learning approach, and the other existing community detection methods like SP [9], FUA [4], FN [8]. Our model was tested on 4 benchmark datasets [10, 1, 12, 22], and found out that it has improved results in 3 of them while a competing result with fourth one. We used NMI as quality measure to understand obtained cluster correlation. In Table 3.1, second column contains layer configuration of corresponding datasets, N and m refer to the number of nodes, edges respectively and K is the ground truth number of communities that we have used in the experiment. From Table 3.1, it can be seen that for the datasets of *Polbooks*, *Polblogs* and *Dolphin* networks, the proposed model performs with slightly better improved results and we have a close margin on the *Football* network with DNR[31].

We believe that the reason for this improved result is essentially credited to optimized parameter sharing. With the above experiments we conclude that our model with a varied architecture has shown better results in the majority of datasets tested and a comparable result for the rest. We try to extend this model to bipartite networks which will be explained in the next chapter.

Chapter 4

Bipartite Networks

Bipartite networks are graphs with two types of nodes. Extracting community structure from bipartite networks has great application especially in biological networks. Protein-protein interaction and gene-drug interactions are some of the examples. User-product interactions in recommendation systems is another application. Despite these facts, it is surprising that very little research is going on in this field. Here we try to solve the problem of community detection in bipartite networks by incorporating an existing idea.

4.1 Related works

It is a practice of using unipartite algorithms for bipartite networks by projection or ignoring the type information. This is less effective due to loss of information in nodes. Another challenge in this field was to formulate modularity concept, by considering type information. Guimera et al [14] proposed a bipartite modularity as the difference from random expectation of the number of edges between vertex members of same bipartite modularity. It has a weakness that it only considers connectivity from one vertex. Barber et al [2] added both connectivity information but had one to one constraints for communities on both sides. Murata's [24] method overcomes these limitations and is considered to be an apt measure for bipartite modularity. We use murata's modularity in our approach.

Despite the limitations, Newman's Leading Eigen Vector algorithm [9] gives good results for community detection in many bipartite networks. Barber pro-

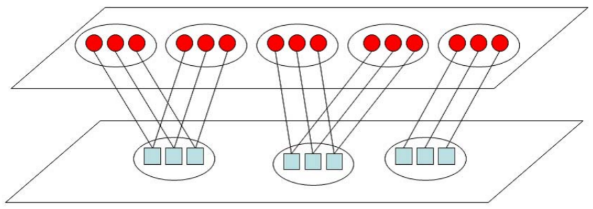

Figure 4.1: Example of bipartite network (source: Tsuyoshi Murata, Detecting communities from Bipartite Networks based on Bipartite Modularities)

posed two modularity measures called BRIM and Adaptive BRIM [2] in 2007, in which the number of communities c varies based on the modularity gain. Later in 2009 Murata et al proposed a method which is a combination of Label Propagation(LP) and BRIM [21]. It uses LP to figure out best community configuration and uses BRIM to optimize it. This method was extended by Beckett et al [3] recently in a tool called DIRTLPAwb+. Pesantez et al [27] conducted a detailed study on biological bipartite networks and proposed a model based on louvain community detection concept.

This Chapter is arranged as follows. First we introduce Murata's modularity in next section , then we quickly walk through BiNE [11] in section 4.3, a method we adopted for vertex learning. Then we explain our approach of using BiNE to formulate our model in Section 4.4 followed by experiments in section 4.5.

4.2 Murata's Modularity

Even though Newman's modularity measure can be used to calculate the division of unipartite networks, it cannot be directly applied for bipartite networks. Suppressing vertex type information leads to loss of valuable information in real world networks. As a result, Guimera et al [14] and Barber et al [2] proposed their own measure which comes with its own limitations as discussed in section 4.1. Murata overcomes these major limitations and proposed his modularity measure. We chose Murata's modularity measure for final evaluation of the detected communities in bipartite networks. An example of Bipartite networks is shown in figure 4.1.

Let the network have two types of nodes where $V = V_x \Cup V_y$, $|V_x| = n_1$ and

$|V_y| = n_2$. Suppose community division is given for V_x and V_y. Let C_i and D_j be communities within V_x and V_y respectively. A be the adjacency matrix of the network whose position value (k,l) equals to 1, if there exist an edge between k and l and equals to 0 otherwise.

Then

$$e_{ij} = \frac{1}{2m} \sum_{k \in C_i} \sum_{l \in D_j} A(k,l)$$

$$a_i = \sum_{l in D_j} e_{il} = \frac{1}{2m} \sum_{k \in C_i} \sum_{l \in V_y} A(k,l)$$

Murata's modularity gives values based on the interaction between communities across vertex types. Murata's modularity depends on computing the maximum number of edges that exist by pairing V_x communities with V_y communities. In the following section define BiNE which we chose as the vertex representation for our approach.

4.3 BiNE: Bipartite Network Embedding

BiNE is a method for vertex representation uniquely developed for bipartite networks. There are a lot of work in network embedding for unipartite networks recently [26, 17, 31] as we discussed before. However very less research have been done uniquely for bipartite networks. Even though, general embedding methods like [13] can be used for bipartite networks it is proved to be not optimal [11] due to loss of vertex type information.

Reconstruction of networks without information loss is a primary function in representation learning. The interconnectivity of nodes in homogeneous networks makes the process of reconstruction a bit easier, since all the relations are explicit. This is not the case in bipartite networks. No two vertices of the same node type have links between them, it does not mean there are no interactions to be considered between them. Figuring out these implicit relations between nodes of same type is one challenging task in bipartite vertex learning.

BiNE [11] has a well modelled algorithm that takes implicit and explicit relations into consideration. The vertex representation of both vertex sets are obtained by joint optimization of explicit and implicit relations.

4.3.1 Algorithm

The entire process in BiNE can be summarized into following steps

Step 1: Modelling Explicit Relations.

Explicit structural information includes the edges between two vertex sets. To model this relation, empirical distribution of vertex co-occuring probability and reconstructed distribution of embedding is calculated. Difference between them is measured using KL divergence and minimized.

Step 2: Modelling Implicit Relations.

If there exist a path between two vertices, some implicit relation exist between them. The path length and number of path determines the strength of relation. Since this is impractical to calculate for large bipartite datasets, solution from Deepwalk is adopted. That is, bipartite network is converted to two corpora of vertex sequences using random walks. Embeddings are learned from this corpora.

Step 3: Apply Skipgram.

Apply Skipgram [23] to obtained corpora to learn vertex embeddings. The idea is to project vertices occured in the same context as similar ones in vertex embedding.

Step 4: Negative Sampling.

To reduce the complexity of learning, negative sampling is used.

Step 5: Joint Optimization.

In the final step, to preserve implicit and explicit relations, their objective functions are joined to form a joint optimization. Stochastic Gradient Descent is used to optimize this joint model.

4.4 Our Approach

We give input network to BiNE algorithm to obtain network matrix representation for both vertex types. We reduce this representation to a lower dimension using autoencoder as discussed in Chapter 3.

Dataset	n_1	n_2	Edges
Kevan	30	114	312
Kato	91	679	1206
Memmott	25	79	299

Table 4.1: Dataset statistics

Dataset	Our Model	Leading-Eigen-vector	LPBrim	AdaptiveBrim
Kevan	0.3048	0.4334	0.4428	0.4301
Kato	0.3069	0.6029	0.5785	0.5541
Memmott	0.3215	0.3150	0.3307	0.3358

Table 4.2: Bipartite modularity against baseline methods.

Our intuition is that these reduced representations will make vertices more separable in the latent space. To understand the vertex interactions across two vertex sets and define the division, we calculate bipartite modularity (Murata's). Before going into it, the implicit interaction within vertex sets has to be defined. For that we chose K-Means clustering. Now that we got communities in both vertex sets, we estimate the quality of community interaction across vertex sets by calculating Murata's modularity.

4.5 Experiments and Results

We tested our model with benchmark datasets and compared with popular baseline methods. Table 4.1 shows dataset statistics. Here n_1 corresponds to number of nodes in first vertex set and n_2 in second vertex set. Initially we chose layer configuration as 128-100-64. Batch size is taken as 128 for each dataset and trained the autoencoder for 10,000 epochs. We deliberately removed dropout due to small size in dataset, and later we tested to check improvement.

Table 4.2 depicts comparison of results of our model with other methods(modularity). From Table 4.2, it is clear that our model does not perform to the expectation. To understand the reason behind, we chose one dataset and varied n_1 and n_2 values (with layer configuration 128-64-32) and tested extensively as depicted in Table 4.3. For *kevan* dataset, since n_1 is smaller compared to n_2 we vary the values of number of communities accordingly as in Table 4.3. We also tried different layer

n_1	2	2	2	2	3	3	4	4	4
n_2	2	3	4	6	4	7	6	8	10
Q	0.2633	0.2645	0.2417	0.2188	0.2147	0.1991	0.1768	0.1590	0.1569

Table 4.3: Bipartite modularity Q against different values of n_1 and n_2 for dataset $Kevan$

configurations like 128-64-32, 128-64-16 and 128-64-8. Since results did not show improvement we are not depicting them.

We concluded that this unexpected result might be as a result of inappropriate clustering in individual vertex sets. A better clustering method based on node interaction between two vertex sets might improve our results. We put that as our future work.

Chapter 5

Conclusion

In this work, we proposed an improved method for the problem of community detection. For unipartite networks, unlike existing methods that use autoencoders, we used shared weights across layers and followed a common 2 layer architecture with one step layer stacked training. Experiments on multiple datasets have shown that this proposed model for unipartite networks performs better than the existing state of the art methods in community detection. Our work delivers a convenient framework with a flexibility of adopting the model to networks of different sizes with reduced training time for community detection.

We extended the same architecture to bipartite networks by adopting a vertex representation model solely built for bipartite networks from literature. Proposed model for bipartite network is only upto the base minimum performance. After extensive experimentation, our intuition is that, principal clustering mechanism might be the reason behind unexpected results. Modification of this approach especially with regard to clustering within vertex sets in order to obtain bipartite communities needs to be investigated further.

Bibliography

[1] Lada A. Adamic. **The Political Blogosphere and the 2004 U.S. Election: Divided They Blog**. *Proceedings of the 3rd International Workshop on Link Discovery*, 04 2005.

[2] Michael J Barber. **Modularity and community detection in bipartite networks**. *Physical Review E*, **76**(6):066102, 2007.

[3] Stephen J Beckett. **Improved community detection in weighted bipartite networks**. *Royal Society open science*, **3**(1):140536, 2016.

[4] Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. **Fast Unfolding of Communities in Large Networks**. *Journal of Statistical Mechanics Theory and Experiment*, **2008**, 04 2008.

[5] Imre Derényi, Gergely Palla, and Tamás Vicsek. **Clique Percolation in Random Networks**. *Phys. Rev. Lett.*, **94**:160202, Apr 2005.

[6] J. Duch and Alexandre Arenas. **Community detection in complex networks using extremal optimization.** *Physical review. E, Statistical, nonlinear, and soft matter physics*, **72 2 Pt 2**:027104, 2005.

[7] Geoffrey E. Hinton and Richard S. Zemel. **Autoencoders, Minimum Description Length and Helmholtz Free Energy**. *Advances in Neural Information Processing Systems*, **6**, 02 1994.

[8] M E J Newman. **Fast algorithm for detecting community structure in networks.** *Physical review. E, Statistical, nonlinear, and soft matter physics*, **69**:066133, 07 2004.

[9] Mark E.J. Newman. **Modularity and community structure in networks.** *Proceedings of the National Academy of Sciences of the United States of America*, **103**:8577–82, 07 2006.

[10] Mark E.J. Newman. **Modularity and community structure in networks.** *Proceedings of the National Academy of Sciences of the United States of America*, **103**:8577–82, 07 2006.

[11] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. **BiNE: Bipartite Network Embedding.** In *SIGIR*, pages 715–724, 2018.

[12] Michelle Girvan and Mark E.J. Newman. **Community structure in social and biological networks**. *proc natl acad sci*, **99**:7821–7826, 11 2001.

[13] Aditya Grover and Jure Leskovec. **Node2Vec: Scalable Feature Learning for Networks**. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA, 2016. ACM.

[14] Roger Guimerà, Marta Sales-Pardo, and Luís A Nunes Amaral. **Module identification in bipartite and directed networks**. *Physical Review E*, **76**(3):036102, 2007.

[15] William L. Hamilton, Zhitao Ying, and Jure Leskovec. **Representation Learning on Graphs: Methods and Applications**. *IEEE Data Eng. Bull.*, **40**:52–74, 2017.

[16] Dongxiao He, Dayou Liu, Di Jin, and Weixiong Zhang. **A Stochastic Model for Detecting Heterogeneous Link Communities in Complex Networks**. In *AAAI*, 2015.

[17] Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. **CommunityGAN: Community Detection with Generative Adversarial Nets**. 01 2019.

[18] Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. **CommunityGAN: Community Detection with Generative Adversarial Nets**. 01 2019.

[19] Di Jin, Zheng Chen, Dongxiao He, and Weixiong Zhang. **Modeling with Node Degree Preservation Can Accurately Find Communities**. In *AAAI*, 2015.

[20] Thomas N. Kipf and Max Welling. **Semi-Supervised Classification with Graph Convolutional Networks**. *CoRR*, **abs/1609.02907**, 2016.

[21] Xin Liu and Tsuyoshi Murata. **Community detection in large-scale bipartite networks**. *Transactions of the Japanese Society for Artificial Intelligence*, **25**(1):16–24, 2010.

[22] David Lusseau and Micaleah Newman. **Identifying the role that animals play in their social networks.** *Proceedings. Biological sciences*, **271 Suppl 6**:S477–81, 2004.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. **Distributed representations of words and phrases and their compositionality**. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[24] Tsuyoshi Murata. **Detecting communities from bipartite networks based on bipartite modularities**. In *2009 International Conference on Computational Science and Engineering*, **4**, pages 50–57. IEEE, 2009.

[25] Mark Newman. *Networks: an introduction.* Oxford university press, 2010.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. **DeepWalk: Online Learning of Social Representations**. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 03 2014.

[27] Paola Gabriela Pesantez and Ananth Kalyanaraman. **Efficient Detection of Communities in Biological Bipartite Networks**. *IEEE/ACM transactions on computational biology and bioinformatics*, 2017.

[28] Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon. **Overlapping community detection using Bayesian nonnegative matrix factorization**. *Physical review. E, Statistical, nonlinear, and soft matter physics*, **83**:066114, 06 2011.

[29] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. **Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion**. *Journal of Machine Learning Research*, **11**:3371–3408, 2010.

[30] Phi Vu Tran. **Learning to Make Predictions on Graphs with Autoencoders**. pages 237–245, 10 2018.

[31] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. **Modularity based community detection with deep learning**. 01 2016.

[32] Fanghua Ye, Chuan Chen, and Zibin Zheng. **Deep Autoencoder-like Nonnegative Matrix Factorization for Community Detection**. In *CIKM*, 2018.

[33] Wayne Zachary. **An Information Flow Model for Conflict and Fission in Small Groups1**. *Journal of anthropological research*, **33**, 11 1976.