24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Integrating Network Embedding and Community Outlier Detection via Multiclass Graph Description

Sambaran Bandyopadhyay[1] and **Saley Vishal Vivek**[2] and **M. N. Murty**[3]

Abstract. Network (or graph) embedding is the task to map the nodes of a graph to a lower dimensional vector space, such that it preserves the graph properties and facilitates the downstream network mining tasks. Real world networks often come with (community) outlier nodes, which behave differently from the regular nodes of the community. These outlier nodes can affect the embedding of the regular nodes, if not handled carefully. In this paper, we propose a novel unsupervised graph embedding approach (called DMGD) which integrates outlier and community detection with node embedding. We extend the idea of deep support vector data description to the framework of graph embedding when there are multiple communities present in the given network, and an outlier is characterized relative to its community. We also show the theoretical bounds on the number of outliers detected by DMGD. Our formulation boils down to an interesting minimax game between the outliers, community assignments and the node embedding function. We also propose an efficient algorithm to solve this optimization framework. Experimental results on both synthetic and real world networks show the merit of our approach compared to state-of-the-arts.

1 INTRODUCTION

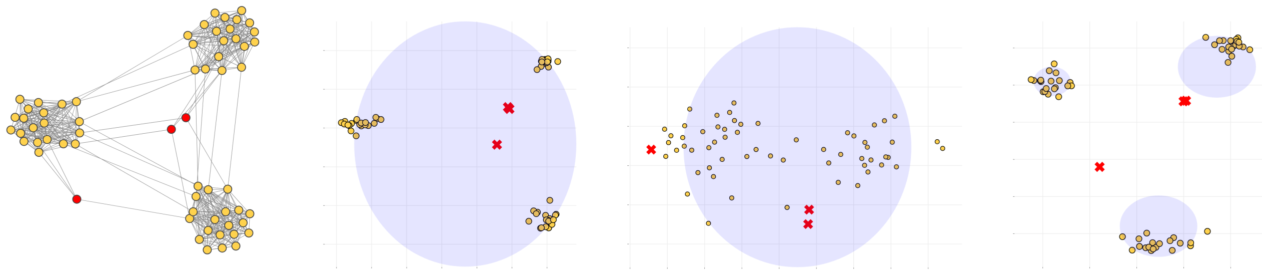
Graphs are popularly used to model structured objects such as social and information networks. Given a graph $G = (V, E)$ with N nodes, network embedding (also known as graph embedding or network representation learning) [23, 16] is the task to learn a function $f : V \to \mathbb{R}^M$, i.e., which maps each node of the graph to a vector of dimension $M < N$. The goal of graph embedding is to preserve the underlying graph structure in the embedding vector space. Typically, the quality of embedding is validated on several downstream graph mining tasks such as node classification, community detection (node clustering), etc. Different types of graph embedding techniques exist in the literature, such as random walk based embedding [23, 10], graph reconstruction based embedding [32, 8], graph neural network based embedding [16, 11, 31], etc. Fundamentally, many of these algorithms work on the assumption of homophily [21] property and the community structures that most of the networks exhibit. These properties ensure that nodes which are directly connected or closer to each other in the graph, tend to be similar to each other in attributes and form a community in the graph.

Most of the above graph embedding algorithms perform good when the nodes behave as expected. But real world networks often contain nodes which are outliers in nature. These outlier nodes behave differently in terms of their connections to other nodes and attribute values, compared to most of the nodes in their respective communities (that's why they are often called community outliers). For example, an outlier node can almost be uniformly connected to nodes from different communities, thus violating the community structure of the network. In this work, we have used the phrases *outlier* and *community outlier* interchangeably. Detection of community outliers has been studied in [9, 7]. But these outlier nodes, though are smaller in number typically, can significantly affect the embedding of the normal nodes, if not treated specially while generating the embeddings. It has been observed that mere post processing of the embeddings cannot filter out the outliers as the embeddings are already affected by them [3]. Recently, [19] proposed a semi-supervised algorithm based on reconstruction loss, combined with node classification error of an autoencoder and [3] proposed an unsupervised matrix factorization based approach to deal with outliers in network embedding, but strictly for attributed networks.

In general, the concept of outlier detection (a.k.a. one class classification) is a widely studied problem in machine learning [27, 29, 18, 33]. In this paper, we would focus on a recent approach, called deep support vector data description (deep SVDD) [25]. Deep SVDD is a deep learning based extension of SVDD [29]. In SVDD, regular data points are mapped to be within a sphere, while outliers stay outside. Deep SVDD uses deep neural networks to learn this mapping. Though Deep SVDD can be applied for detecting outliers in graphs, it is not suitable when the graph has multiple communities or clusters. As shown in Fig. 1, an outlier between two communities can actually be marked as a non-outlier by an approach similar to Deep SVDD or SVDD, where they form only one sphere for the whole graph. Also this type of mapping is not suitable for representation learning on graphs, as there is no explicit way to preserve other characteristics of the graph while detecting the outliers.

Contributions: We propose a novel deep learning based unsupervised algorithm (referred as DMGD - **D**eep **M**ulticlass **G**raph **D**escription) which extends the idea of support vector data description to jointly learn community outliers and node embedding by minimizing the effect of outliers in the embedding space. Our approach meets the requirements when the graph has multiple communities and an outlier is a node not being enclosed by any community. DMGD unifies node representation, outlier detection and community detection in graphs through a single optimization framework which boils down to an interesting minimax game. We have shown the ***theoretical bounds*** on the number of outliers detected by DMGD. Experimental results depict the merit of DMGD on both synthetic and real life network datasets for various downstream network mining tasks. Source code of DMGD can be found at `https://github.com/vasco95/DMGD` to ease the reproducibility of the results.

[1] IBM Research & IISc, Bangalore, email: samb.bandyo@gmail.com
[2] Indian Institute of Science, Bangalore, email: vishalsaley@iisc.ac.in
[3] Indian Institute of Science, Bangalore, email: mnm@iisc.ac.in



(a) Input Graph (b) SVDD (c) Deep SVDD (d) DMGD

Figure 1: We motivate and compare our proposed graph embedding algorithm DMGD with SVDD and Deep SVDD on a small synthetic network with 150 regular nodes divided into 3 communities. There are also 3 community outliers (marked in red) as they do not adhere the community structure of the network. We use Eq. 3 to generate the node embeddings, and then feed them to SVDD and Deep SVDD. We keep the embedding dimension as 2 to plot the node embeddings. As expected, SVDD and Deep SVDD do not respect the community structure of the network and hence mostly include all the outliers within the spheres they form. But DMGD (proposed algo.) detects all the outliers by keeping them outside of the learned community boundaries.

2 RELATED WORK

Detailed surveys on graph representation can be found in [12, 34]. We briefly discuss some important graph embedding techniques here. The concept of representing words in a corpus by vectors [22] in NLP literature influenced some early work in network representation learning. DeepWalk [23], node2vec [10] use random walk on the graph to capture nodes similar to a node and generate similar embeddings for the nodes which are close and frequently reachable from each other. struc2vec [24] is another random walk based technique where structurally similar nodes are assigned similar embeddings, even if they are far from each other in the graph. There are deep autoencoder based graph embedding techniques such as SDNE [32] and DNGR [5] which preserve different orders of proximities of the graph in the embedding space. TADW [36], AANE [13] and DANE [8] use complimentary information from the attributes associated with the nodes in the reconstruction of the graph properties (via matrix factorization and deep autoencoders) to generate node embeddings. Along with node proximity, global node ranking of the graph is preserved in the embedding space in [17].

Graph neural networks [26] gained significant importance in the recent literature. A semi-supervised graph convolutional network (GCN) is proposed by recursively aggregating attribute information from the neighborhood of each node in [16]. GraphSAGE [11] is an inductive representation technique which proposes to aggregate different types of neighborhood aggregation functions in GCN. A scalable and faster version of GCN via neighborhood subsampling technique is proposed in [6]. Attention mechanisms for graph embedding are proposed in GAT [31] and in Graph Attention [1].

None of the above techniques explicitly minimize the effect of outlier nodes in graph embeddings. However, real life social networks often come with outlier nodes, which can affect the embedding of the other nodes of the graph. Recently, [19] proposes a semi-supervised approach, SEANO, which learns outliers in network embedding framework. An unsupervised approach, ONE, is proposed in [3] to minimize the effect of outliers by weighted matrix factorization for attributed network embedding. Extending the idea of ONE, two deep neural architectures are proposed [4] to minimize the effect of outliers on the node embeddings, again for the attributed networks. These three approaches are based on the attributes present in the nodes and exploit the inconsistency between link structures and node attributes in the graph to detect the outliers. In this paper, we propose an integrated unsupervised approach by extending the idea of SVDD from one cluster to multiple clusters, and pose it as a representation learning problem for outlier and community detection in graphs. Contrary to the existing literature, our proposed algorithm DMGD can work by focusing only on the network link structure to detect and minimize the effect of outliers in node embeddings.

3 PRELIMINARIES OF SVDD AND DEEP SVDD

SVDD [29], inspired by support vector classifier, obtains a spherically shaped boundary around the regular points of the given dataset, characterizing outliers as the points which stay outside of the sphere. More formally, for a set of points $x_i \in \mathbb{R}^N$, SVDD aims to find the smallest hypersphere with center at $c \in \mathcal{F}$ (in feature space) and radius $R > 0$ which encloses most of the points, as follows.

$$\begin{aligned} \min_{R,c,\xi} \quad & R^2 + \alpha \sum_i \xi_i \\ \text{such that,} \quad & ||\phi(x_i) - c||_{\mathcal{F}} \leq R^2 + \xi_i, \quad \xi_i \geq 0 \ \ \forall i \end{aligned} \tag{1}$$

$\phi(x_i)$ is a function that maps the data points to a feature space $\mathcal{F}$. $\alpha > 0$ is a weight parameter and ξ_i are the slack variables. Points for which $\xi_i > 0$ stay outside of the sphere and are considered as outliers. Recently, Deep SVDD [25] extends SVDD via deep learning. The (soft-boundary) Deep SVDD objective function is shown below:

$$\begin{aligned} \min_{R,\mathcal{W}} \quad & R^2 + \alpha \sum_i max\{0, ||\phi(x_i;\mathcal{W}) - c||^2 - R^2\} \\ & + \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||_F^2 \end{aligned} \tag{2}$$

Deep SVDD replaces the function ϕ of SVDD with a deep neural network, with the set of parameters $\mathcal{W}$ which includes L layers. Please note the second term of Eq. 2 is equivalent to having the slack variables in Eq. 1. To avoid trivial solution of the optimization problem, authors of [25] do not include the center c as an optimization variable. Rather, they fix it using some preprocessing. Deep SVDD and SVDD suffer from a serious problem. They cannot distinguish outliers from the data when the given dataset has community structure and outliers can reside between communities, but not on the outskirt of the whole dataset, as shown in Fig. 1. Besides as mentioned in Section 1, they are also not suitable for representation learning, as they do not ensure other important properties of the data objects to be preserved in the embedding (feature) space.

4 OUR APPROACH

Here we discuss the proposed algorithm DMGD, which integrates graph embedding with outlier and community detection, in an unsupervised way. Given the graph $G = (V, E)$ with $|V| = N$, DMGD learns a map $f : V \rightarrow \mathbb{R}^M$, where $M < N, D$. We also assume that there are K unknown communities present in the graph. Our goal is to map N vertices of the graph to K communities, such that each regular point stays close to at least one of the communities, whereas, outliers stay outside of those communities. Along with that, we also want to preserve other graph properties in the embedding space, so that it facilitates the downstream graph mining tasks. For notational convenience, we define: $[N] = \{1, 2, \cdots, N\}$, and similarly, $[K] = \{1, 2, \cdots, K\}$.

First, we use a deep autoencoder to generate the initial graph embeddings. For each node v_i, the encoder function $f(a_i; \mathcal{W})$ maps the input structure vector to an M dimensional space. We use rows $a_i \in \mathbb{R}^N$, $\forall i \in [N]$ of the adjacency matrix of the graph G as the structural vector. One can even use page rank vectors [5] to capture higher order proximities of the nodes or additional attributes (if available) to replace the structural vector. There is also a decoder function $g(f(a_i), \mathcal{W})$ which maps the embedding of the node back to $\mathbb{R}^N$ space to reconstruct the input [4]. $\mathcal{W} = \{W^1, \cdots, W^L\}$ contains parameters for L layers of the autoencoder. We assume both encoder and decoder contain equal number of hidden layers. The autoencoder minimizes the reconstruction loss defined as: $\sum\limits_{i=1}^{N} ||a_i - g(f(a_i))||_2^2$ with respect to the parameters $\mathcal{W}$ of the neural network. We also use the homophily property [21] of an information network, which ensures two nodes which are directly connected by an edge to behave similarly. So, we minimize the L2 distance of the two embeddings where the corresponding nodes are connected by an edge: $\sum\limits_{(i,j)\in E} ||f(a_i) - f(a_j)||_2^2$. Thus, the total loss minimized to preserve these two properties is:

$$\min_{\mathcal{W}} \quad \sum_{i=1}^{N} ||a_i - g(f(a_i))||_2^2 + \sum_{(i,j)\in E} ||f(a_i) - f(a_j)||_2^2 \qquad (3)$$

It is important to note that this formulation is very generic and can be replaced easily with alternate unsupervised techniques that are based on graph convolution autoencoders [15] or random walks [10].

Next, we integrate outlier and community detection with the graph embedding objective. In the process, we would also reduce the effect of outliers on the embedding of other regular nodes. Given, there are K unknown communities in the input graph, we seek to obtain the centers of these K communities. Let, C contains these centers as, $C = \{c_1, \cdots, c_K\} \subset \mathbb{R}^M$. For each community, we like to find the smallest hypersphere [29, 18] which encloses majority of the embeddings from that community. Let, $R_k > 0$ be the radius of the k^{th} community, with $\mathcal{R} = \{R_1, \cdots, R_K\}$. For any node v_i, its community is determined by the sphere which encloses it (anyone if there are multiple such spheres) or by the smallest distance of the periphery of the spheres when it is outside of all the spheres (i.e., outliers). So the **community index** for the node v_i is $\underset{k}{\text{argmin}} \max\{||f(a_i) - c_k||_2^2 - R_k^2,\ 0\}$. So, given the embeddings of the nodes as $f(a_i), \forall i$, we optimize the following quantity:

$$\begin{aligned} \min_{\mathcal{R},\mathcal{C},\xi} \quad & \sum_{k=1}^{K} R_k^2 + \alpha \sum_{i=1}^{N} \xi_i \\ \text{such that} \quad & \min_{k\in\{1,\cdots,K\}} \{||f(a_i) - c_k||_2^2 - R_k^2\} \leq \xi_i \ \ \forall i \in [N] \\ & \xi_i \geq 0,\ \forall i \in [N] \ \text{ and } \ R_k \geq 0,\ \forall k \in [K] \end{aligned} \qquad (4)$$

Here $\xi_i \geq 0$ is the slack variable corresponding to the ith node of the graph. With respect to this formulation, nodes can be divided into three categories as follows. A **regular node** is one which stays strictly inside a community, and thus for it: $\min\limits_{k} \{||f(a_i) - c_k||_2^2 - R_k^2\} < 0$. A **boundary node** is one which lies exactly on the boundary of its community, so for it: $\min\limits_{k} \{||f(a_i) - c_k||_2^2 - R_k^2\} = 0$. An **outlier node** stays outside of all the communities, and thus for it: $\xi_i > 0$ (strictly positive). These slack variables ensure soft spherical boundaries of the communities, and outlier nodes stay outside of the community. When the weight parameter α is very small, an optimizer would mainly focus to minimize the first term in the cost function in Eq. 4, leading to very small (in terms of radius) spherical communities and many nodes will be treated as outliers. Whereas, a higher value of α ensures lesser number of outliers with larger communities. Our approach implicitly assumes that the communities in the graph are spherical in the embedding space, which is not a hard requirement. Because of the soft boundaries of the spheres and the characterization of the outliers, it can also handle communities which are not exactly spherical. The nonlinear properties of the original network are captured by neural networks for mapping into the embedding space.

Eq. 3 ensures that generic graph properties are preserved in the embedding space, while Eq. 4 ensures that the community structure is maintained in the embedding space, while separating outliers from the other nodes. So the combined objective of DMGD is given below:

$$\begin{aligned} \min_{\mathcal{R},\mathcal{C},\xi,\mathcal{W}} \quad & \sum_{k=1}^{K} R_k^2 + \alpha \sum_{i=1}^{N} \xi_i + \beta \sum_{i=1}^{N} ||a_i - g(f(a_i))||_2^2 \\ & + \gamma \sum_{(i,j)\in E} ||f(a_i) - f(a_j)||_2^2 \\ \text{such that,} \quad & \min_{k\in\{1,\cdots,K\}} \{||f(a_i) - c_k||_2^2 - R_k^2\} \leq \xi_i \ \ \forall i \in [N] \\ & \xi_i \geq 0,\ \forall i \in [N] \ \text{ and } \ R_k \geq 0,\ \forall k \in [K] \end{aligned} \qquad (5)$$

Following lemmas show the formal connection between the number of outliers and the parameter α.

Lemma 1 ***Number of outlier nodes*** ($|\{v_i : \xi_i > 0\}|$) ***detected by DMGD is upper bounded by*** $\frac{K}{\alpha}$.

Proof 1 *Suppose, the communities detected by DMGD are denoted by C_k, $k = 1, \cdots, K$. The community index for node v_i is argmin max$\{||f(a_i) - c_k||_2^2 - R_k^2,\ 0\}$ (under argmin: k). We use ν-property as stated in [27] to prove the claim. Partial objective function of DMGD can be written as:*

$$\sum_{k=1}^{K} R_k^2 + \alpha \sum_{i=1}^{N} \xi_i \ = \ \sum_{k=1}^{K} \Big(R_k^2 + \frac{1}{\frac{1}{\alpha}} \sum_{i\in C_k} \xi_i \Big)$$

[4] We omit the parameter set $\mathcal{W}$ from the function definitions of f and g when there is no ambiguity

So, when the communities are fixed, the objective for each community is exactly same as SVDD and thus ν-property ensures that the number of outliers from each community is upper bounded by $\frac{1}{\alpha}$. Hence total number of outlier nodes detected by DMGD is upper bounded by $\frac{K}{\alpha}$.

Lemma 2 ***The sum of boundary nodes (nodes which lie exactly on the boundaries of their respective communities) and outlier nodes detected by DMGD is lower bounded by $\frac{K}{\alpha}$.***

This can also be proved similarly using the ν-property. Lemmas 1 and 2 together show that the number of outlier nodes detected by DMGD is actually bounded tightly, assuming not many nodes lie exactly on the community boundaries.

4.1 Optimization and Training

Optimizing Eq. 5 is difficult because of multiple reasons. One primary reason is the presence of the constraints of type $\min_k \{||f(a_i) - c_k||_2^2 - R_k^2\} \leq \xi_i$. So we use the following trick to replace them with some other variables, as follows:

$$\begin{aligned}
\min_{\mathcal{R},\mathcal{C},\xi,\mathcal{W},\Theta} \quad & P = \sum_{k=1}^{K} R_k^2 + \alpha \sum_{i=1}^{N} \xi_i + \beta \sum_{i=1}^{N} ||a_i - g(f(a_i))||_2^2 \\
& + \gamma \sum_{(i,j)\in E} ||f(a_i) - f(a_j)||_2^2 \\
\text{such that,} \quad & \sum_{k=1}^{K} \theta_{ik}(||f(a_i) - c_k||_2^2 - R_k^2) \leq \xi_i \ \ \forall i \in [N], \\
& \sum_{k=1}^{K} \theta_{ik} = 1, \quad \theta_{ik} \geq 0, \ \ \forall i \in [N], \ \forall k \in [K], \\
& \xi_i \geq 0, \ \ \forall i \in [N] \ \text{ and } \ R_k \geq 0, \ \ \forall k \in [K]
\end{aligned} \tag{6}$$

We assume, $\Theta = \{\theta_{ik} \mid \forall i, k\}$. Optimizing Eq. 6 is simpler compared to Eq. 5, as we replaced the minimum over some functions by a functions which are linear in Θ.

Lemma 3 ***If $(\mathcal{R}^*, \mathcal{C}^*, \xi^*, \mathcal{W}^*, \Theta^*)$ is a minimum of Eq. 6, then $(\mathcal{R}^*, \mathcal{C}^*, \xi^*, \mathcal{W}^*)$ is a minimum of Eq. 5.***

Proof 2 *The feasible set of the optimization problem in Eq. 6 is a super set of that in Eq. 5. Hence the minimum value of Eq. 6 is always less than or equal to the minimum value of Eq. 5. Also, the loss function of both the optimization problems are the same. Let us denote the term: $r_{ik} = ||f(a_i) - c_k||_2^2 - R_k^2$. For any i, define $k^*(i)$ as the community index for which r_{ik} is minimum, i.e., $k^*(i) = \underset{k}{argmin}\ r_{ik}$. We will write $k^*(i)$ as just k^* when there is no ambiguity about i.*

To prove the claim: First, let us prove, for any given point $(\mathcal{R}, \mathcal{C}, \xi, \mathcal{W}, \Theta)$, there exists a set of non-negative slack variables $\bar{\xi}$ for which $P(\mathcal{R}, \mathcal{C}, \bar{\xi}, \mathcal{W}, \bar{\Theta}) \leq P(\mathcal{R}, \mathcal{C}, \xi, \mathcal{W}, \Theta)$, where P is the cost function in Eq. 6 and in $\bar{\Theta}$, for each i, $\theta_{ik^*} = 1$, and $\theta_{ik} = 0$, $\forall k \neq k^*$. Now, for any $i \in [N]$, $\sum_{k\in[K]} \theta_{ik} r_{ik} \geq r_{ik^*}$, as $r_{ik^*} \leq r_{ik}$ and $\theta_{ik} \geq 0$, $\forall k$ with $\sum_k \theta_{ik} = 1$. For each i, set $\bar{\xi}_i = \xi_i - (\sum_{k\in[K]} \theta_{ik} r_{ik} - r_{ik^*}) \geq 0$. Clearly, $\bar{\xi}_i \leq \xi_i$, and $\sum_i \bar{\xi}_i \leq \sum_i \xi_i$. Hence, $P(\mathcal{R}, \mathcal{C}, \bar{\xi}, \mathcal{W}, \bar{\Theta}) \leq P(\mathcal{R}, \mathcal{C}, \xi, \mathcal{W}, \Theta)$.

Thus if $(\mathcal{R}^*, \mathcal{C}^*, \xi^*, \mathcal{W}^*, \Theta^*)$ is already a minimum point of Eq. 6, then $\bar{\xi} = \xi$. Hence, $(\mathcal{R}^*, \mathcal{C}^*, \xi^*, \mathcal{W}^*, \bar{\Theta}^*)$ is a minimum of Eq. 6 and also because of the definition of $\bar{\Theta}$, $(\mathcal{R}, \mathcal{C}, \bar{\xi}, \mathcal{W})$ belongs to the feasible set of Eq. 5 and is a minimum of the same.

Lemma 3 shows solving the optimization problem in Eq. 6 is somewhat equivalent[5] to solving the optimization problem in Eq. 5. Hence, we will focus to solve Eq. 6 now onward. Let us first derive the partial Lagrangian dual of the same with respect to the variables $\mathcal{R}, \mathcal{C}, \xi, \mathcal{W}, \Theta$, assuming $\mathcal{W}$ (parameters of the neural network) as constant. We will later update $\mathcal{W}$ by backpropagation.

$$\begin{aligned}
\mathcal{L} = \sum_{k=1}^{K} R_k^2 + \alpha \sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \lambda_i \sum_{k=1}^{K} \Big[\theta_{ik}\big(f(a_i) - c_k||^2 - R_k^2\big) \\
- \xi_i \Big] - \sum_{i=1}^{N} \eta_i \xi_i
\end{aligned} \tag{7}$$

Here, $\lambda_i, \eta_i, \forall i \in [N]$ are non negative Lagrangian constants. Equating the partial derivatives of $\mathcal{L}$ with respect to R_k, c_k, and ξ_i to zero and using some algebraic manipulation, we get the following:

$$\begin{aligned}
\sum_{i=1}^{N} \lambda_I \theta_{ik} = 1, \ \forall k, \quad c_k = \sum_{i=1}^{N} \lambda_i \theta_{ik} f(a_i), \ \forall k \in [K], \\
0 \leq \lambda_i \leq \alpha, \ \forall i \in [N]
\end{aligned} \tag{8}$$

Using the above constraints under KKT conditions, the (partial) Lagrangian dual can be written as:

$$\begin{aligned}
\mathcal{D} = & \sum_{i=1}^{N} \sum_{k=1}^{K} \lambda_i \theta_{ik} f(a_i)^T f(a_i) \\
& - \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \lambda_i \lambda_j \theta_{ik} \theta_{jk} f(a_i)^T f(a_j)
\end{aligned} \tag{9}$$

Clearly we want to maximize the above w.r.t. Λ (where, $\Lambda = (\lambda_1, \cdots, \lambda_N)^T \in \mathbb{R}^N$) and minimize with respect to Θ. Hence, including the terms associated with the neural network parameters and the set of constraints, we seek to solve the following for DMGD.

$$\begin{aligned}
& \underset{\Theta,\mathcal{W}}{min} \ \underset{\Lambda}{max} \ \sum_{i=1}^{N} \lambda_i f(a_i)^T f(a_i) \\
& - \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \lambda_i \lambda_j \theta_{ik} \theta_{jk} f(a_i)^T f(a_j) \\
& + \beta \sum_{i=1}^{N} ||a_i - g(f(a_i))||_2^2 + \gamma \sum_{(i,j)\in E} ||f(a_i) - f(a_j)||_2^2 \qquad (10) \\
& \text{such that,} \ \sum_{i=1}^{N} \lambda_i \theta_{ik} = 1, \quad 0 \leq \lambda_i \leq \alpha, \\
& \sum_{k=1}^{K} \theta_{ik} = 1, \quad \theta_{ik} \geq 0, \quad \forall i \in [N], \ \forall k \in [K]
\end{aligned}$$

Interpretation of the Optimization: The above formulation is a very interesting minimax game, where the objective is to minimize the cost w.r.t. the community assignment variables Θ and neural network parameters $\mathcal{W}$; and to maximize the same w.r.t. the Lagrangian constants Λ. From KKT complementary slackness, it can be shown that, $0 \leq \lambda_i \leq \alpha$ for the boundary nodes and $\lambda_i = \alpha$

[5] There can other issues related to different local minima of the cost functions.

for the outlier nodes. For regular nodes, $\lambda_i = 0$. Thus, only the third and fourth terms of the cost function in Eq. 10 play role to generate the embedding of the regular points. But for a boundary or an outlier point, first two terms also contribute. If we focus on the second term $\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\sum\limits_{k=1}^{K}\lambda_i\lambda_j\theta_{ik}\theta_{jk}f(a_i)^Tf(a_j)$ which needs to be minimized w.r.t. Λ, λ_i would be high for a node which is less similar (small values of $f(a_i)^Tf(a_j)$) to all other nodes in the community. This is another way of interpreting outliers in this framework. Whereas, the embedding function f, by maximizing the above, tries to keep the nodes in the same cluster close to each other. Minimizing the first term w.r.t. the neural network parameters is equivalent to having an L2 regularizer on the embeddings of the nodes. The nodes with very high L2 norms are more likely to be outliers because of this term. Rewriting the second term we get $\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\lambda_i\lambda_jf(a_i)^Tf(a_j)(\sum\limits_{k=1}^{K}\theta_{ik}\theta_{jk})$. Maximization of this term wrt community assignments $(\theta_{i1},\cdots,\theta_{iK})$ will be such that the term $\sum\limits_{k=1}^{K}\theta_{ik}\theta_{jk}$ will be higher when product $\lambda_i\lambda_jf(a_i)^Tf(a_j)$ is higher (node similar other outlier or boundary node). In other words, a node will be assigned a community which is most similar to it.

Parameter Initialization and Training of DMGD: Like in any other deep learning technique, parameter initialization plays an important role in the convergence of our algorithm. First, we run few epochs of the autoencoder (optimizing just Eq. 3) without considering the other terms. This gives generic embeddings $f(a_i)$ for each node $v_i \in V$. Then we run k-means++ algorithm [2] (with number of clusters equal to K) on these embeddings to get the initial hard community assignments (i.e., values of θ_{ik}). Lemma 3 shows a minimum always corresponds to a hard community assignment.

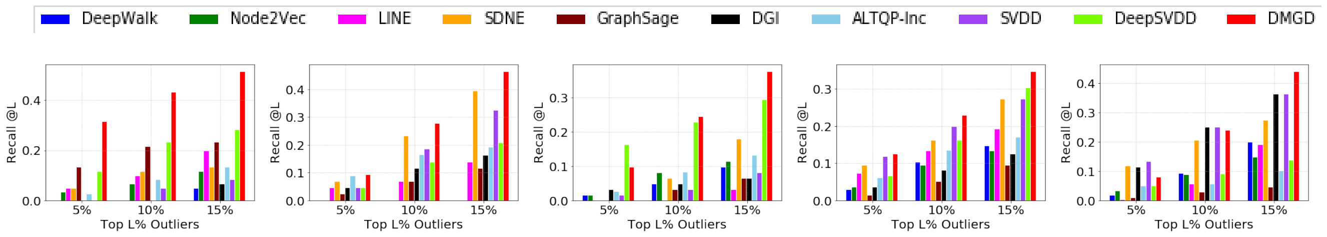
Once we have the initial embeddings and the community assignments, **(i)** we use standard quadratic solver CVXOPT qpsolver [6] to update the values of Λ (Eq. 10 is a constrained quadratic w.r.t. Λ when other variables are fixed). **(ii)** Then we calculate the center of each community $k \in [K]$ by $c_k = \sum\limits_{i=1}^{N}\lambda_i\theta_{ik}f(a_i)$ from Eq. 8. **(iii)** For each community $k \in [K]$, we calculate the radius R_k as $\min\limits_{i}\{||f(a_i) - c_k||_2^2 \mid \theta_{ik} = 1\ ,\ 0 < \lambda_i < \alpha\}$. Experimentally we consider all the nodes for which $\lambda_i > 0$ and $\lambda_i < \alpha - 10^{-6}$ to compensate for numerical errors. **(iv)** We reassign the community of a node v_i, $\forall i \in [N]$ with the updated embeddings and centers by updating $\theta_{ik} = \underset{k}{\text{argmin}}\ \max\{||f(a_i) - c_k||_2^2 - R_k^2,\ 0\}$. **(v)** Finally, with all other updated variables fixed, we compute the partial gradients of Eq. 10 w.r.t. the neural network parameters $\mathcal{W}$ and update them using backpropagation with ADAM optimizer. It is to be noted that, exact computation of the partial gradient w.r.t. each node takes $O(NK)$ time due to the second term in Eq. 10. To avoid this, we sub-sample a fixed number of nodes from the same community to approximate that term. Similarly, we also sub-sample a fixed number of nodes from the neighborhood of any node [11] to approximate the fourth term (homophily loss) Eq. 10. We use the standard parameterization of the ADAM optimizer as in [14]. Steps (i) to (v) are run sequentially within a loop and the loop is iterated for a fixed number of times or until the loss (in Eq. 10) converges. λ_i is considered as the outlier score of the node $v_i \in V$. Please refer Algorithm 1 for the pseudocode of DMGD.

Time and Space Complexity Analysis: Initial cluster assign-

Algorithm 1 DMGD - Deep Multiclass Graph Description

```
Input: The graph $G = (V, E)$, $|V| = N$, Given or generated feature vector $a_i \in \mathbb{R}^D$ for each node $v_i \in V$, $M$: Dimension of the embedding space, $K$: Number of communities in the graph
Output: The node embeddings $f(a_i)$, $i \in [N]$ of the graph $G$, Outlier score $\lambda_i$, $i \in [n]$ of the nodes, Community Centers $c_k \in \mathbb{R}^M$ for each community $k \in [K]$, Community assignment variables $\theta_{i1}, \cdots, \theta_{iK}$ for each node $i \in [N]$.
Run few epochs of the autoencoder (optimizing just Eq. 3 of the main paper) without considering the other terms to initialize the node embeddings
Run k-means++ algorithm (with number of clusters equals to $K$) on these embeddings to get the initial hard community assignments (i.e., values of $\theta_{ik}$).
for $iter \in \{1, 2, \cdots, T\}$ do
    Use standard quadratic solver CVXOPT qpsolver to update the values of $\Lambda$ (refer to Eq. 10)
    Calculate the center of each community $k \in [K]$ by $c_k = \sum_{i=1}^{N}\lambda_i\theta_{ik}f(a_i)$ from Eq. 8 of the main paper.
    for $k \in [K]$ do
        Calculate the radius $R_k$ as $\min_{i}\{||f(a_i) - c_k||_2^2 \mid \theta_{ik} = 1\ ,\ 0 < \lambda_i < \alpha\}$
    end for
    for $i \in [N]$ do
        Reassign the community of the node $v_i$ with the updated embeddings and centers by updating $\theta_{ik} = \underset{k}{\text{argmin}}\ \max\{||f(a_i) - c_k||_2^2 - R_k^2,\ 0\}$.
    end for
    Retrain the autoencoder (Section 3 of the main paper) by computing the partial gradients of Eq. 10 of the main paper w.r.t. the neural network parameters $\mathcal{W}$ and update them using backpropagation with ADAM optimizer
end for
```

[6] https://cvxopt.org/examples/tutorial/qp.html



(a) LFR Benchmark (b) WebKB (c) Polblogs (d) Cora (e) Pubmed

Figure 2: Recall at top L% from the ranked list of outliers by different Embedding algorithms.

ments using k-means++ algorithm takes $O(NMK)$ time, assuming number of iterations needed is a constant, where M is the embedding dimension. CVXOPT has run time of $O(NlogN)$ to updates Λ. Steps (ii) to (iv) take a total time of $O(NK)$. Followed by that, updates of the neural network again take $O(NK)$ time, thanks to the sub-sampling techniques which also make the time complexity independent of the number of edges in the network, without any significant drop in performance. Hence the major bottleneck is the computation of Λ using the quadratic solver. To overcome this, one can use the bounds on the number of points having non-zero λ_i from Lemmas 1 and 2. An improved solution based on this can be addressed in the future. The space complexity of DMGD is $O(|V| + |E|)$. The graph can be stored as an adjacency list on the disk. DMGD does not need the whole $O(N^2)$ expensive adjacency matrix to work with.

5 EXPERIMENTAL EVALUATION

5.1 Datasets Used and Seeding Outlier

One primary goal of DMGD is to handle and detect outliers while generating the embeddings. To the best of our knowledge, there is no publicly available standard network dataset with ground truth outliers. Also for many of the network datasets, underlying community structure does not match the label of the nodes, i.e., two nodes having same ground truth label may not belong to the same community and vice versa. So to validate our algorithm, we use a combination of synthetic (LFR Benchmark: `https://bit.ly/2Xx4EJh`) and real world network datasets, described in Table 1. We also seed 5% outliers into each dataset by perturbing nodes as follows. To perturb each outlier, we select a node randomly from the dataset. We find the top 20% nodes from the dataset which are at the farthest distance from the selected node. Finally we randomly sample an equal number of nodes as the degree of the node in the network from neighbors of those 20% nodes and the neighbors of the selected nodes. Most of the neighbors of the selected node belong to the same community, and most of the farthest nodes belong to different communities. Thus our perturbed outliers have edges to nodes from multiple communities, and satisfy the conditions of a community outlier. Please note, labels of outliers have **not** been considered for calculating the accuracy of any downstream task.

Table 1: Summary of the datasets, after planting outliers.

Dataset	#Nodes	#Edges	#Labels
LFR Benchmark	1200	5277	10
WebKB	877	2897	5
Polblogs	1224	19025	2
Cora	2708	5429	7
Pubmed	19717	44338	3

5.2 Baseline Algorithms and Experimental Setup

The proposed algorithm DMGD is unsupervised in nature. So as baselines, we choose only well-known unsupervised embedding algorithms which can work even without attributes of the nodes. The baselines are DeepWalk [23], node2vec [10], LINE [28], SDNE [32], GraphSAGE [11] (unsupervised version) and DGI [31] for all the downstream tasks. We use the publicly available implementation of these algorithms, with default hyper parameter settings. We do not consider node attributes as the focus is to exploit only the network structure. Additionally, we consider SVDD and Deep SVDD only for the experiments on outlier recall, as these two algorithms are not meant for network embedding. We generate node embeddings using Eq. 3 before applying SVDD. We have also used following two purely graph based algorithms (not for node embedding): AltQP-Inc [30] for outlier detection and SBMF [37] for community detection, and included the results for the respective tasks.

Embedding dimension is fixed at 16 for all the algorithms, on all the datasets, except Pubmed. For Pubmed, the embedding dimension is 32 as it is larger in size. We tried with increased embedding dimensions also (for e.g., 128), but there is no significant improvement in results. Encoder and decoder in DMGD contain two layers each for all the datasets. We use leaky-ReLU activation for non-linearity in all the layers except the last one, which has ReLU activation. We train autoencoder using ADAM [14] optimizer with default parameters.

5.3 Setting the hyper parameters of DMGD

Like many other ML algorithms, we also assume to know the number of communities K of a network. We theoretically show the relation between the hyper parameter α and the number of outliers detected by DMGD. If the expected number of outliers is known a priori, α can be set accordingly. The ratio $\frac{\gamma}{\beta}$, after taking β as common from the last two components of Eq. 5 of the paper, weights the autoencoder reconstruction loss and the homophily loss and can be set as shown in [32]. β balances the outlier and community detection parts of DMGD with the embedding generation part. Increasing value of β would give more importance to generating the generic node embeddings, where decreasing it would be to minimize the effect of outliers and give importance on the community structure of the network. To set the value of β, we use the standard grid search which minimizes both total loss and the individual component losses of the DMGD cost function.

5.4 Outlier Detection, Community Detection and Node Classification

Outlier detection is extremely important for network embedding, as discussed in Section 1. We use λ_i as the outlier score of the node

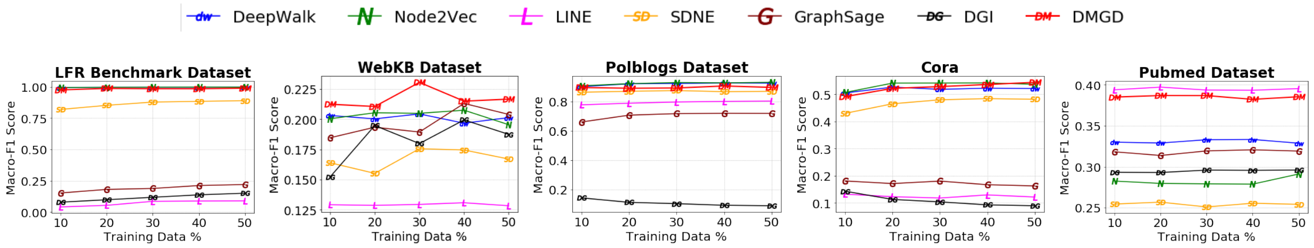


Figure 3: Accuracy of Node classification with Logistic Regression

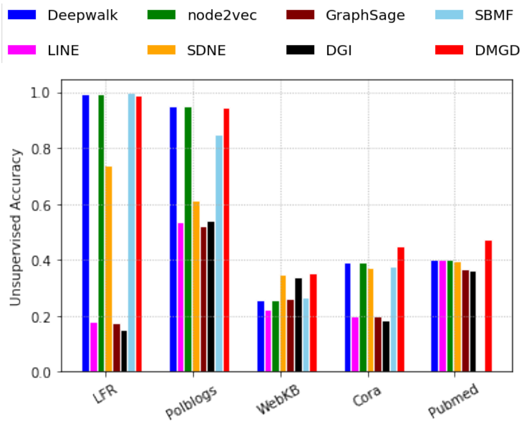


Figure 4: Unsupervised accuracy for community detection

$v_i \in V$ for DMGD (Sec. 4.1). SVDD and Deep SVDD also produce outlier scores of the nodes directly. For other baselines, we use isolation forest algorithm [20] on the generated embeddings to get outlier scores of the nodes. Each seeded dataset has 5% outliers. So we plot the outlier recall from the top 5% to 25% of the nodes in the ranked list (L) with respect to the seeded outliers in Fig. 2. Clearly, DMGD outperforms all the baseline algorithms for outlier recall. As it adheres multiple community structure of the network, it is able to detect outliers which lie between the communities. Deep SVDD, due to its high non-linear nature, turns out to be more consistent than other baselines. Most of the standard graph embedding algorithms like node2vec and GraphSAGE suffer as they do not process outliers while generating the embeddings.

DMGD also outputs the community assignment of the nodes in the graph though the set of variables Θ (Sec. 4.1). Here we check the quality of the communities produced by DMGD, with respect to the ground truth labeling of the datasets. For the baseline algorithms (except for SBMF), we give the node embeddings as input to KMeans++ [2]. To judge the quality of clustering, we use unsupervised clustering accuracy [35]. Figure 4 shows that DMGD performs better or almost as good as the best of the baselines for community detection. As DMGD integrates community detection with graph embedding and outlier detection, the output communities are more optimal than most of the baselines which finds communities by post-processing the embeddings. We could not present the result of SBMF on Pubmed as the runtime exceeds more than 3 days.

To compare the quality of node embeddings generated by DMGD to that of the baselines, we also consider node classification. We vary the training size from 10% to 50%. We train a logistic regression classifier on the training set of embeddings (along with the class labels) and check the performance on the test set by using Macro F1 score. Figure 3 shows the performance of node classification for all the publicly available datasets we used. Though the optimization of DMGD explicitly handles communities and outliers in node embeddings, its performance for node classification is highly competitive to state-of-the-art network embedding algorithms. On WebKB, DMGD turns out to be the best performer, while for other datasets, it is always very close to the best of the baselines. DeepWalk, node2vec, SDNE and LINE also perform good for node classification task depending on the datasets.

Table 2: Classification (macro & micro F1 with training size 30%) and clustering (unsupervised accuracy) performance on the unseeded and seeded versions of Cora.

		node2vec	LINE	GraphSAGE	DGI	DMGD
Macro-F1 (%)	unseeded	**55.99**	16.02	20.55	10.84	53.05
	seeded	**54.11**	11.80	18.03	10.38	52.80
Micro-F1 (%)	unseeded	**60.72**	35.32	34.86	31.54	56.32
	seeded	55.99	29.16	31.65	27.85	**57.24**
Clustering (%)	unseeded	39.51	27.51	22.34	20.08	**43.83**
	seeded	39.31	19.77	20.12	18.45	**44.86**

5.5 Influence of Outliers on Node Embeddings

Here, we empirically show the negative influence of outliers on the node embeddings and thus motivate the problem again. To show the effect of outliers, we run DMGD and some of the better performing and diverse baseline algorithms (due to limitation of page) on both the unseeded (original) and seeded (with outliers) versions of Cora dataset in Table 2. We use node classification and clustering as the downstream tasks to show the effect. Clearly, most of the baseline algorithms are affected because of the presence of the community outliers. The performance of an algorithm is generally better in the unseeded version of the dataset than the seeded one. This can be seen by comparing two consecutive rows for a metric in Table 2. For DMGD, the adverse effect is less as it is resistant to outliers. In some of the cases, there is some marginal improvement in the performance of DMGD on the seeded dataset.

6 DISCUSSION AND FUTURE WORK

In this work, we proposed an unsupervised algorithm DMGD, which integrates node embedding, minimizing the effect of outliers and community detection into a single optimization framework. We also show the theoretical bounds on the number of outliers detected by it. One shortcoming of DMGD is that, it depends heavily on the community structure of the network, which may not be so prominent in some real network. In future, we would like to address this issue. We would also like to conduct experiments on networks with overlapping communities to check the performance of DMGD in that case.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi, 'Watch your step: Learning node embeddings via graph attention', in *Advances in Neural Information Processing Systems*, pp. 9180–9190, (2018).

[2] David Arthur and Sergei Vassilvitskii, 'k-means++: The advantages of careful seeding', in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics, (2007).

[3] Sambaran Bandyopadhyay, N. Lokesh, and M. Narasimha Murty, 'Outlier aware network embedding for attributed networks', in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, Hawaii, USA*, (2019).

[4] Sambaran Bandyopadhyay, Saley Vishal Vivek, and MN Murty, 'Outlier resistant unsupervised deep architectures for attributed network embedding', in *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 25–33, (2020).

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu, 'Deep neural networks for learning graph representations', in *Thirtieth AAAI Conference on Artificial Intelligence*, (2016).

[6] Jie Chen, Tengfei Ma, and Cao Xiao, 'FastGCN: Fast learning with graph convolutional networks via importance sampling', in *International Conference on Learning Representations*, (2018).

[7] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu, 'Deep anomaly detection on attributed networks', (2019).

[8] Hongchang Gao and Heng Huang, 'Deep attributed network embedding.', in *IJCAI*, pp. 3364–3370, (2018).

[9] Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han, 'On community outliers and their efficient detection in information networks', in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 813–822. ACM, (2010).

[10] 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*, pp. 855–864. ACM, (2016).

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec, 'Inductive representation learning on large graphs', in *Advances in Neural Information Processing Systems*, pp. 1025–1035, (2017).

[12] William L Hamilton, Rex Ying, and Jure Leskovec, 'Representation learning on graphs: Methods and applications', *arXiv preprint arXiv:1709.05584*, (2017).

[13] Xiao Huang, Jundong Li, and Xia Hu, 'Accelerated attributed network embedding', in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 633–641. SIAM, (2017).

[14] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, (2014).

[15] Thomas N Kipf and Max Welling, 'Variational graph auto-encoders', *arXiv preprint arXiv:1611.07308*, (2016).

[16] Thomas N Kipf and Max Welling, 'Semi-supervised classification with graph convolutional networks', in *International Conference on Learning Representations*, (2017).

[17] Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh, and Shou-De Lin, 'Prune: Preserving proximity and global ranking for network embedding', in *Advances in neural information processing systems*, pp. 5257–5266, (2017).

[18] Trung Le, Dat Tran, Phuoc Nguyen, Wanli Ma, and Dharmendra Sharma, 'Multiple distribution data description learning method for novelty detection', in *The 2011 International Joint Conference on Neural Networks*, pp. 2321–2326. IEEE, (2011).

[19] Jiongqian Liang, Peter Jacobs, Jiankai Sun, and Srinivasan Parthasarathy, 'Semi-supervised embedding in attributed networks with outliers', in *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 153–161. SIAM, (2018).

[20] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, 'Isolation forest', in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422. IEEE, (2008).

[21] Miller McPherson, Lynn Smith-Lovin, and James M Cook, 'Birds of a feather: Homophily in social networks', *Annual review of sociology*, **27**(1), 415–444, (2001).

[22] 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*, pp. 3111–3119, (2013).

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena, 'Deepwalk: Online learning of social representations', in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, (2014).

[24] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo, 'struc2vec: Learning node representations from structural identity', in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 385–394. ACM, (2017).

[25] Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft, 'Deep one-class classification', in *International Conference on Machine Learning*, pp. 4390–4399, (2018).

[26] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini, 'The graph neural network model', *IEEE Transactions on Neural Networks*, **20**(1), 61–80, (2009).

[27] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson, 'Estimating the support of a high-dimensional distribution', *Neural computation*, **13**(7), 1443–1471, (2001).

[28] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei, 'Line: Large-scale information network embedding', in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077. International World Wide Web Conferences Steering Committee, (2015).

[29] David MJ Tax and Robert PW Duin, 'Support vector data description', *Machine learning*, **54**(1), 45–66, (2004).

[30] Hanghang Tong and Ching-Yung Lin, 'Non-negative residual matrix factorization with application to graph anomaly detection', in *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 143–153. SIAM, (2011).

[31] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm, 'Deep graph infomax', in *International Conference on Learning Representations*, (2019).

[32] Daixin Wang, Peng Cui, and Wenwu Zhu, 'Structural deep network embedding', in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234. ACM, (2016).

[33] Mingrui Wu and Jieping Ye, 'A small sphere and large margin approach for novelty detection using training data with outliers', *IEEE transactions on pattern analysis and machine intelligence*, **31**(11), 2088–2092, (2009).

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu, 'A comprehensive survey on graph neural networks', *arXiv preprint arXiv:1901.00596*, (2019).

[35] Junyuan Xie, Ross Girshick, and Ali Farhadi, 'Unsupervised deep embedding for clustering analysis', in *International conference on machine learning*, pp. 478–487, (2016).

[36] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang, 'Network representation learning with rich text information.', in *IJCAI*, pp. 2111–2117, (2015).

[37] Zhong-Yuan Zhang, Yong Wang, and Yong-Yeol Ahn, 'Overlapping community detection in complex networks using symmetric binary matrix factorization', *Physical Review E*, **87**(6), 062803, (2013).