FraudNE: a Joint Embedding Approach for Fraud Detection

Mengyu Zheng*†, Chuan Zhou*†, Jia Wu‡, Shirui Pan§, Jinqiao Shi*†, and Li Guo*†
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
‡Department of Computing, Faculty of Science and Engineering, Macquarie University, Sydney, Australia
§Centre for Artificial Intelligence, University of Technology Sydney, Australia
{zhengmengyu, zhouchuan, shijinqiao, guoli}@iie.ac.cn, shirui.pan@uts.edu.au, jia.wu@mq.edu.au

***Abstract*—Detecting fraudsters is a meaningful problem for both users and e-commerce platform. Existing graph-based approaches mainly adopt shallow models, which cannot capture the highly non-linear relationship between vertexes in a bipartite graph composed of users and items. To address this issue, in this paper we propose a joint deep structure embedding approach *FraudNE* for fraud detection that (a) can preserve the highly non-linear structural information of networks, (b) is robust to sparse networks, (c) embeds different types of vertexes jointly in the same latent space. It is worth mentioning that we can detect multiple fraudulent groups without the number of groups as a priori. Compared with baselines, our method achieved significant accuracy improvement.**

***Index Terms*—Dense Block Detection, Network Embedding, Deep Structure Learning**

I. Introduction

Online user feedback for the target items, such as reviews and ratings, usually incurs considerable influence for potential buyers. It is on this very basis that the fraudulent behavior has become more and more widespread. For example, one-third of consumer reviews and rates on the Amazon, and more than one-fifth of reviews on Yelp are estimated to be fake [1]. Users are easily cheated by these fake feedbacks and purchase items with poor quality. The user churn of e-commerce platform is then following. Hence, how to detect fraudsters and corresponding target items is a serious problem for both users and e-commerce platform.

Generally speaking, fraudsters always act in lockstep to increase total impact of target items. The fraud detection problem can therefore be viewed as finding suspicious dense blocks in the attributed bipartite graph, where the source nodes represent user accounts, the sink nodes are items, and the directed edges stand for feedback record. The attributes on each edge can be organized as a sensor composed of timestamp, rating score, review and so on.

Most attribute-based methods exploited the user accounts/items/feedbacks related features to address the suspicious dense blocks detection problem [2], [3]. However, these attributes-based methods for dense block detection are not adversarially robust. The fraudsters can fine-tune their text and behavior to make their features unsuspicious. For example, fraudsters can manipulate login times, their location, internet providers and IPs via large pools. What's more, some attributes

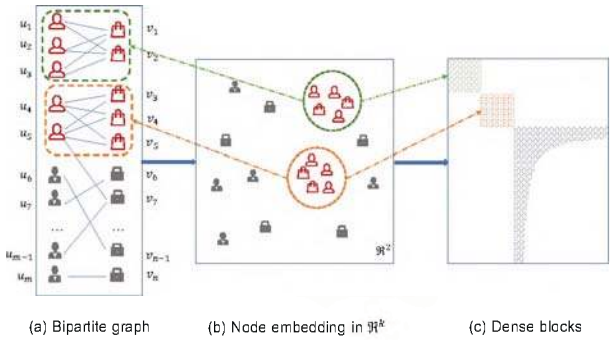


Fig. 1: The rough idea of joint bipartite graph embedding for Fraud Detection.

are not readily available or unaffordable to obtain in reality. All these make it difficult for these attributes-tensor based methods to be widely used. In contrast, the topology information of the corresponding bipartite graph is relatively easy to obtain and cannot be easily camouflaged. Therefore, the problem we focus on is: whether and to what extent we can address the fraud detection problem, if only graph topology is available.

Although there have been some dense block detection methods involving topology information only, such as designing kinds of density metrics and maximizing the arithmetic or geometric degree [4], [5], these methods mainly adopted shallow models and therefore they were particularly sensitive to topology structure. Besides, existing methods can not detect multiple dense blocks unless the block number is predefined [6], [7], while it is usually unrealistic for us to know the number of dense blocks beforehand.

To address these issues, in this paper we propose an unsupervised method named *FraudNE* to detect abnormal users and items through deep joint network embedding. Network embedding aims to learn the node latent representation, which encode network structural information in a continuous vector space and preserve long-range and global structure. The proposed *FraudNE* makes full use of bipartite graph topology information and embeds all users and items into the same low dimension space simultaneously. *FraudNE* aims to make the representations of the users and items in the same dense block as close as possible, while the presentations of the normal

users or items distribute uniformly in the latent space. We then cluster groups of fraudsters and their corresponding abnormal items without the number of dense blocks as a priori. The main idea is shown in Fig. 1.

In summary, our contributions are:

- We propose a novel *FraudNE* model to detect fraudulent dense blocks through deep structure learning on the attributed bipartite graph.
- *FraudNE* can automatically recognize the number of fraudulent blocks in bipartite graph without the block number predefined.
- Experimental results show that *FraudNE* achieves higher accuracy than state-of-the-art methods on several datasets.

The rest of paper is organized as follows. Section II is devoted to the formulation of fraud detection. In Section III we proposed *FraudNE* to detect fraud with the help of network embedding. We present the experimental results in Section IV. In Section V we review the related work. Finally, this paper is closed with Section VI.

II. Problem Definition and Challenges

We define the fraud detection problem in bipartite network with user proximity, item proximity and local proximity.

Definition 1. *(Fraud Detection) Given a bipartite and directed graph $G = (U, V, E)$, where U is the set of source nodes, V is the set of sink nodes, and E is the set of directed edges from U to V. In graph G, each $u \in U$ represents a user, each $v \in V$ represents an item, and each $e_{i,j} \in E$ is associated with a weight $w_{i,j}$ representing the number of reviews from u_i to v_j. In graph G, we hope to find multiple dense blocks for suspicious users and items detection.*

Fraudulent groups are assumed to share some common characteristics in general [7]. Fraudsters always give as much review as possible to the suspicious items, while few normal users connect to the suspicious items.

Definition 2. *(Joint Network Embedding) Given a bipartite network $G = (U, V, E)$, the problem of joint network embedding is to embed all source nodes in U and all sink nodes in V into a low-dimensional space R^d simultaneously, where d is a parameter specifying the representation dimensions. Generally, $d \ll min(|U|, |V|)$.*

Network embedding aims to map the graph data into a low-dimensional latent space. To conduct the embedding appropriate for fraud detection, the network structures should be preserved. Then we define three proximities between the different types of vertexes.

Definition 3. *(User Proximity) The user proximity between a pair of users u_i, u_j in a network is the similarity between their reviewed objects. We use binary vector $\mathbf{s}_i = (w_{i,1}, \ldots, w_{i,n})$ to present the sink nodes that source node u_i reviews. The user proximity between u_i and u_j is determined by the similarity between $\mathbf{s}_i$ and $\mathbf{s}_j$. If user u_i and user u_j have the similar review behavior, they would have high proximity. If two users review absolutely different objects, their proximity is zero.*

Definition 4. *(Item Proximity) For each $v_i \in V$, we use binary vector $\mathbf{m}_i = (w_{1,i}, \ldots, w_{m,i})$ to present the source nodes linked with sink node v_i. The item proximity between v_i and v_j is determined by the similarity between $\mathbf{m}_i$ and $\mathbf{m}_j$. If the similar users give similar attention to item v_i and v_j, they should have high item proximity.*

Definition 5. *(Local Proximity) The local proximity in a network is the local pairwise proximity between source node u and sink node v. For each pair of source-sink nodes linked by weighted edge, the number of reviews $w_{u,v}$ indicates the local proximity between user u and object v. If no edge is observed between source-sink nodes (u_1 and v_6 in Fig. 1), it means their local proximity is zero.*

Our proposed model aims to embed the fraudulent users and items in the same dense block as close as possible, while the presentations of the normal users and items distribute uniformly in the low-dimensional latent space. Hence, it is necessary to preserve the user proximity, item proximity and local proximity in the joint network embedding process.

III. The *FraudNE* Model

In order to detect fraudulent groups, we first propose a deep joint network embedding framework, named *FraudNE*, to embed source nodes $u \in U$ and sink nodes $v \in V$ jointly in one latent space. And then we make use of a clustering algorithm to find a high-density area in the latent space.

Deep neural networks have achieved satisfactory performance in homogeneous network embedding. *SDNE* [8] employs a semi-supervised deep model to construct the node representations. Because of the preservation of highly nonlinear network structure, they achieved better effectiveness in some tasks. Our network embedding framework is compose of deep autoencoder as well. There are two distinct differences between our joint network embedding framework and *SDNE*.

- *SDNE* is a network embedding framework only for homogeneous networks and there is only one autodecoder. It means that *SDNE* is unable to cope with bipartite networks. The goal of our framework is embedding different types of vertexes jointly in one latent space and our framework include two autoencoders which handle source nodes and sink nodes in networks, respectively. Therefore, our approach can solve the problem of bipartite network representations.
- The opposite of *SDNE* is a task-independent representation method, *FraudNE* is an embedding method for fraud detection. Although both of the *SDNE* as well as *FraudNE* are optimized by exploiting stochastic gradient decent, the betch selection of two types of vertexes is not completely random due to the low ratio of suspicious vertexes. As a result, we firstly choose certain number of source vertexes randomly. Then, we weighted randomly extract vertexes from V, where the weights are indegrees of each sink vertexes from chosen source vertexes.

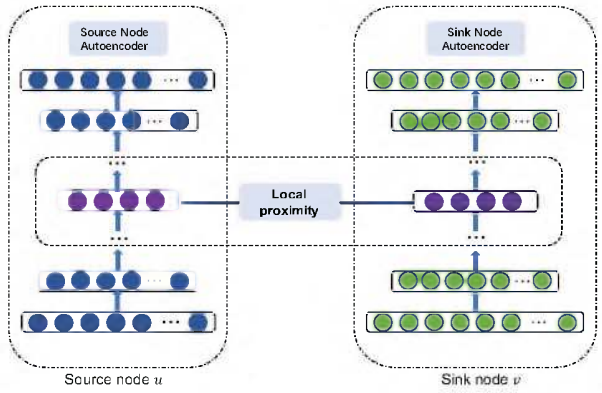


Fig. 2: The framework of *FraudNE*

A. Framework

We proposed a deep model to capture the highly non-linear structure information between the vertexes. As shown in Fig. 2, the network embedding framework consists of two unsurprised components, i.e., source node representation part and sink node representation part.

To reconstruct the node representations, we propose a deep architecture, which is composed of multiple non-linear functions to map the input into a latent space. Source node representation part and sink node representation part are completely different. These two parts can have different neural network structures, parameters and non-linear activation functions. To preserve the structural information between source nodes and sink nodes, we add constraints to refine the representations in latent space. Meanwhile, the model is robust to sparse networks. We detail this model in the following section.

B. Loss Function

We define some notations and terms in Table I. We first introduce the input of two autoencoder parts. Given a network $G = (U, V, E)$, we can obtain its source-sink interaction matrix S, which contains m instances $\mathbf{s}_1, \ldots, \mathbf{s}_m$. For each instance in the matrix $\mathbf{s}_i = \{s_{i,j}\}_{j=1}^{n}$, $s_{i,j} > 0$ if and only if there exists a weighted edge between u_i and v_j. Therefore, each instance $\mathbf{s}_i$ describes the reviews from u_i to all items $v \in V$. Matrix S preserves structural information of each source vertex in network, and $\mathbf{s}_i$ is the input vector of autoencoder in source node represent part.

At the same time, matrix S also preserves structural information of each sink vertex. We can represent S^{T} with $S^{\mathrm{T}} = \{\mathbf{m}_j, \ldots, \mathbf{m}_n\}$. For each instance $\mathbf{m}_j = \{m_{i,j}\}_{i=1}^{n}$. $m_{i,j} > 0$ if and only if there exists links between u_i and v_j. Therefore, each instance $\mathbf{m}_j$ describes the reviewing source of v_j.

Both source node representation part and sink node representation part are compose of deep autoencoder, an artificial neural network used for unsupervised learning composed with the purpose of reconstructing its own inputs. It consist with two parts, encoder part and decoder part. Both parts have an input layer, an output layer and one or more hidden layers with non-linear activation function connecting them. Besides, the nodes in output layer are as much as the nodes in input layer.

TABLE I: Notations and terms

Symbol	Definition
m	number of source vertexes
n	number of sink vertexes
K_u	number of layers in user part
K_v	number of layers in itwm part
$S = \{\mathbf{s_1}, \ldots, \mathbf{s_m}\}$	adjacency matrix for network
$X = \{\mathbf{x_i}\}_{i=1}^{m}, \hat{X} = \{\hat{\mathbf{x}}_\mathbf{i}\}_{i=1}^{m}$	input and reconstruct data in user part
$Y = \{\mathbf{y_i}\}_{i=1}^{n}, \hat{Y} = \{\hat{\mathbf{y}}_\mathbf{i}\}_{i=1}^{n}$	input and reconstruct data in item part
$Z^{(k_u)} = \{\mathbf{z}_i^{(k_u)}\}_{i=1}^{m}$	k_u-th layer hidden representations in user part
$H^{(k_v)} = \{\mathbf{h}_i^{(k_v)}\}_{i=1}^{n}$	k_v-th layer hidden representatons in item part
$W_u^{(k_u)}, \hat{W}_u^{(k_u)}$	k_u-th layer weight matrix in user part
$W_v^{(k_v)}, \hat{W}_v^{(k_v)}$	k_v-th layer weight matrix in item part
$\mathbf{b}_u^{(k_u)}, \hat{\mathbf{b}}_u^{(k_u)}$	the k_u-th layer biases in user part
$\mathbf{b}_v^{(k_v)}, \hat{\mathbf{b}}_v^{(k_v)}$	the k_v-th layer biases in item part
$\theta = \{W, \hat{W}, \mathbf{b}, \hat{\mathbf{b}}\}$	the overall parameters

Then in the source node representation part, the encoder stage of an autoencoder takes the input $\mathbf{s}_i \in R^n$ and maps it to hidden layers. The hidden representations of each layer are [1]:

$$\begin{aligned} \mathbf{z}_i^{(1)} &= \sigma(\mathbf{W_u^{(1)}}\mathbf{s_i} + \mathbf{b}_u^{(1)}) \\ \mathbf{z}_i^{(k_u)} &= \sigma(\mathbf{W_u^{(k_u)}}\mathbf{z_i^{(k_u-1)}} + \mathbf{b}_u^{(k_u)}) \end{aligned} \tag{1}$$

After getting $\mathbf{z}_i^{(K_u)}$, the decoder stage of the autoencoder maps it to the reconstruction $\hat{\mathbf{s}}_i$ of the same shape of $\mathbf{s}_i$. The goal of the autoencoder is to minimise reconstruction errors (such as squared errors):

$$\mathcal{L}_u = \sum_{i=1}^{m} \|\hat{\mathbf{s}}_i - \mathbf{s}_i\|_2^2 \tag{2}$$

Obviously, in sink node representation part, taking the input $\mathbf{m}_j \in R^m$, the encoder maps it to hidden layers. The hidden representations of each layer are:

$$\begin{aligned} \mathbf{h}_j^{(1)} &= \sigma(\mathbf{W_v^{(1)}}\mathbf{m_j} + \mathbf{b}_v^{(1)}) \\ \mathbf{h}_j^{(k_v)} &= \sigma(\mathbf{W_v^{(k_v)}}\mathbf{h_j^{(k_v-1)}} + \mathbf{b}_v^{(k_v)}) \end{aligned} \tag{3}$$

And the loss function is shown as follow:

$$\mathcal{L}_v = \sum_{i=1}^{n} \|\hat{\mathbf{m}}_i - \mathbf{m}_i\|_2^2 \tag{4}$$

The reconstruction criterion can preserve the similarity between samples through capturing the data manifolds smoothly [9], though minimizing the reconstruction loss does not be manifestly concerned about the similarity. Therefore, the latent representations would be similar when the vertexes have similar reviewed items or receive reviews from similar users.

However, due to the sparsity of the input vector, the number of non-zero elements is much smaller than that of zero elements. That means the autoencoder will tend to reconstruct the

[1] we use sigmoid function $\sigma(x) = \frac{1}{1+exp(-x)}$ as the non-linear function in every hidden layer in two parts

zero elements rather than non-zero ones, which is incompatible with our purpose. To increase the distinction, we give different weights to different elements, and redefine Eq. (2) and Eq. (4) as follows:

$$\mathcal{L}_u = \sum_{i=1}^{m} \|(\hat{\mathbf{X}} - \mathbf{X}) \odot D^u\|_2^2 \tag{5}$$

$$\mathcal{L}_v = \sum_{i=1}^{n} \|(\hat{\mathbf{Y}} - \mathbf{Y}) \odot D^v\|_2^2 \tag{6}$$

where $\odot$ means Hadamard product, D^u and D^v are weight vertors, $D^u = \{\mathbf{d}_i^u\} = \{d_{i,j}^u\}_{j=1}^n$, if $\mathbf{s}_{i,j} = 0, d_{i,j}^u = 1$, else $d_{i,j}^u = \beta > 1$. $D^v = \{\mathbf{d}_j^v\} = \{d_{i,j}^u\}_{i=1}^m$, if $m_{i,j} = 0, d_{i,j}^v = 1$, else $d_{i,j}^v = \beta > 1$.

We design the third part of framework to preserve the local proximity. The proximity can be regarded as the link information to restrict the latent representation of a pair of source node and sink node. Eq. (7) borrows the idea of Laplacian Eigenmaps [10]. A penalty will be introduced when similar vertexes are mapped far away from each other in the latent space. We incorporate the idea in our deep model to resist the two different types of vertexes linked by edge to be mapped near in latent space. As a result, the loss function for this part is shown as:

$$\mathcal{L}_{local} = \sum_{i=1}^{m}\sum_{j=1}^{n} s_{i,j} \|\mathbf{z}_i^{(K_u)} - \mathbf{h}_j^{(K_v)}\|_2^2 \tag{7}$$

To preserve three proximities at the same time, we propose a deep model which combines Eq. (5), Eq. (6) and Eq. (7) and joint minimizes the following objective function:

$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_v + \alpha\mathcal{L}_{local} + \eta\mathcal{L}_{reg} \tag{8}$$

Here, we introduce two hyper-parameters α and η to balance the weights of different parts. Moreover, $\mathcal{L}_{reg}$ is an $\mathcal{L}2$-norm regularizer to prevent overfitting, which is defined as:

$$\begin{aligned}\mathcal{L}_{reg} = &\frac{1}{2}\sum_{k=1}^{K_u}(\|W_u^{(k)}\|_F^2 + \|\hat{W}_u^{(k)}\|_F^2) \\ &+ \frac{1}{2}\sum_{k=1}^{K_v}(\|W_v^{(k)}\|_F^2 + \|\hat{W}_v^{(k)}\|_F^2)\end{aligned}$$

C. Optimization

Our goal is to minimize $\mathcal{L}$ which is the proposed objective function. Besides, the key step is to compute the partial derivative of $\partial\mathcal{L}/\partial\hat{W}_u^{(k)}$ and $\partial\mathcal{L}/\partial W_u^{(k)}$. Since the training principle of two parts in our framework are the same, we take source node representation part as example:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial W_u^{(k)}} &= \frac{\partial\mathcal{L}_u}{\partial W_u^{(k)}} + \alpha\frac{\partial\mathcal{L}_{local}}{\partial W_u^{(k)}} + \eta\frac{\partial\mathcal{L}_{reg}}{\partial W_u^{(k)}} \\ \frac{\partial\mathcal{L}}{\partial\hat{W}_u^{(k)}} &= \frac{\partial\mathcal{L}_u}{\partial\hat{W}_u^{(k)}} + \eta\frac{\partial\mathcal{L}_{reg}}{\partial\hat{W}_u^{(k)}}\end{aligned} \tag{9}$$

Algorithm 1 Training Algorithm for deep model of *FraudNE*

Input: bipartite network $G = (U, V, E)$ with user-item matrix S , the parameters α and η .
Output: latent representations and updated parameters: θ.

1: Pretrain the model through deep belief network [11] to initialize parameters θ.
2: $X = S$, $Y = S^{\mathrm{T}}$
3: **repeat**
4: Based on X, Y and θ, apply Eq. 1 and Eq. 3 to obtain $\hat{X}$, $\hat{Y}$ and $L = L^{K_u}$, $H = H^{K_v}$.
5: Apply Eq. 8 to compute objective function
6: Based on Eq. 9, use $\partial\mathcal{L}/\partial\theta$ to back-propagate through the entire network to get updated parameters θ.
7: **until** converge
8: Obtain the latent representation P.

where $\partial\mathcal{L}_u/\partial\hat{W}_u^{(k)}$ rephrased as follows:

$$\frac{\partial\mathcal{L}_u}{\partial\hat{W}_u^{(k)}} = \frac{\partial\mathcal{L}_u}{\partial X} \cdot \frac{\partial X}{\partial\hat{W}_u^{(k)}} \tag{10}$$

According to Eq. 5, we can compute:

$$\frac{\partial\mathcal{L}_u}{\partial X} = 2(X - \hat{X}) \odot B \tag{11}$$

The calculation of $\partial X/\partial\hat{W}_u^{(k)}$ is accessible. Based on back-propagation, we can iteratively calculate $\partial\mathcal{L}_u/\partial\hat{W}_u^{(k)}$, $k = 1, \ldots, K-1$ and $\partial\mathcal{L}_u/\partial W_u^{(k)}$, $k = 1, \ldots, K-1$.

The calculation of the partial derivative of $\partial\mathcal{L}_{local}/\partial W_u^{(k)}$ is accessible, since the loss function of $\mathcal{L}_{local}$ is:

$$\mathcal{L}_{local} = \sum_{i=1}^{m}\sum_{j=1}^{n} s_{i,j}\|\mathbf{z}_i^{(K_u)} - \mathbf{h}_j^{(K_v)}\|_2^2 = 2tr(P^{\mathrm{T}}LP) \tag{12}$$

where $P = Z + H$[2], and $L = D - A$, $D \in \mathbb{R}^{(m+n)\times(m+n)}$ is a diagonal matrix, $D_{i,i} = \sum_j S_{i,j}$. Besides, A is adjacency matrix. $A \in \mathbb{R}^{(m+n)\times(m+n)}$. Therefore, $\partial\mathcal{L}_{local}/\partial W_u^{(k)}$ is accessible by using back-propagation. The full algorithm is presented in Alg. 1.

D. Clustering

FraudNE can learn the representations of the user and item nodes in the same fraud block as close as possible, while the presentations of the normal nodes are far away from these fraudulent nodes. After obtaining representations, we choose *DBSCAN*, a clustering algorithm, to complete fraud detection. We briefly review the fundamental idea of *DBSCAN*. It is a density-based method grouping together points that are closely packed together. Besides *DBSCAN* marks as outliers that lie alone in low-density regions. It has a notion of noise and does not require one to specify the number of clusters in the data in advance.

[2]we denote network representations $Z = \{z_i^{K_u}\}_{i=1}^n$ as $Z^{(K_u)} = \{z_i^{K_u}\}_{i=1}^n$, and $H = \{h_i^{K_u}\}_{i=1}^n$ as $H^{(K_u)} = \{h_i^{K_u}\}_{i=1}^n$

Algorithm 2 clustering fraudulent groups

Input: latent representations of vertexes in networks $P = L + H$, the paramaters ϵ and $minPts$.

Output: fraudulent groups F including users and items.

1: Find ϵ neighbors of every point, and identify n core points with more than $minPts$ neighbors.
2: Find the connected components of core points on the neighbor graph, ignoring all non-core points.
3: Assign each non-core point to a nearby cluster if the cluster is an ϵ neighbor, otherwise assign it to noise.
4: Calculate the average degree $aver$ of the whole dataset.
5: **While** $i < n$
6: Calculate the average degree $aver_i$ of every cluster.
7: **if** $aver_i < aver$: cluster $i \in F$.
8: **end while**
9: Obtain fraudulent groups F.

After obtaining n clusters, we calculate average degree $aver_i$ in each cluster and average degree $aver$ in the whole dataset. If $aver_i > aver$, cluster i is a fraudulent group. The full algorithm is presented in Alg. 2.

E. Analysis and Discussion

New Vetexs coming: How to learn representation first for a new source node or sink node before clustering the fraudulent groups is a practical issue. For a new source vertex u_k, if its review information is known, we can obtain its vector $\mathbf{x} = s_{k,1}, \ldots, s_{k,n}$. Then we feed $\mathbf{x}$ into our model and get the representation for u_k with the help of trained parameters θ. However, if there is no edge from u_k to sink vertexes, the proposed method and state-of-the-art methods will fail to learn its representation. Our goal is detecting the fraudulent groups in the network. Obviously, if the new source vertex is not linked to any existing sink nodes, it is impossible to be a fraudulent user. Therefore, it does not need to get its latent representation.

Complexity: It is obviously that the training complexity of framework is $O((m + n)dCI)$, where m is the number of source vertexes and n is the number of sink vertexes, d is the maximum dimension of the hidden layer that might be related to the dimension of embedding vectors instead of number of vertexes, C means the average degree of network and is always a constant in real-world applications, and I is the number of iterations, independent of m or n. Therefore, dCI is be foreign to both m and n. In the nutshell, the training complexity is linear to the number of vertexes in the network. And as we all know, *DBSCAN* executes exactly one such query for each point, and if an indexing structure is used that executes a neighborhood query in $O(logn)$. An overall average runtime complexity of $O(nlogn)$ is obtained.

IV. Experiments

A. Datasets

Table II details the statistics of three datasets which are publicly available for academic research. We mimic the behaviors of fraudsters and randomly select a certain number of objects as target items in our experiments. Since unpopular items are inclined to hire fake reviews, indegree of the object we select is less than 50. We can also uniformly select a certain number of users from the whole user set as the fraudulent users, since the fraudulent accounts may be from the hijacked user accounts. To test the ability to detect multiple density blocks, the number of fraudulent groups ranges from 1 to 3. Each fraudulent group includes 400 fraudsters and 200 target items. Those fraudsters as a whole randomly rate each of the target item for 200 times. At the same time, we also create some camouflage on other products.

TABLE II: Satsitics of the dataset

Data Name	#nodes	#edges
Zomato	5.3K ×1.0K	36K
MovieLens	6.04K ×3.90K	1M
BookCrossing	77.8K ×55.6K	434K

B. Baseline Algorithms

We use the following three methods as baselines. The first two are dense block detection methods [4], [7]. We select a network embedding method [12] as the last comparison algorithm, which is aiming to prove the necessity of proposing a new embedding method for network fraud detection. For fair comparison, only topology information can be used for all the methods in our experiments.

- **HoloScope**[3]: This is a graph topology-based weighting scheme dynamically reweights objects according to beliefs about which users are suspicious.
- **FRAUDAR**[4]: This is an edge weighting scheme based on inverse logarithm of objects' degrees, which inspired by IDF.
- **DeepWalk**[5]: This approach adopts random walk and skip-gram model to generate learn d-dimensional representations. We will use the same clustering algorithm after obtaining the vertexes representations.

C. Parameter Settings

TABLE III: Neural Network Structure

Data Name	components	#nodes in each layer
Zomato	source nodes part	1013-500-100
	sink nodes part	5337-600-100
MovieLens	source nodes part	3900-600-100
	sink nodes part	6040-700-100
BookCrossing	source nodes part	9225-3500-800-100
	sink nodes part	14219-5000-1000-100

The neural network structure in this paper varies with different datasets. The structure of autoencoder in two components can be completely different as long as the number of nodes

[3]https://github.com/shenghua-liu/HoloScope
[4]www.andrew.cmu.edu/user/bhooi/camo.zip
[5]https://github.com/phanein/deepwalk

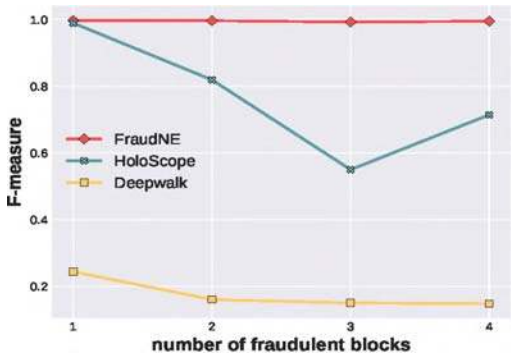


Fig. 3: F_1-measure of fraud detection on Zomato dataset

in the last encoder layer is the same. The dimension of each layer is listed in Table III.

The parameter α, η, β in deep model and the parameter ϵ, minPts in clustering algorithm are tuned to be optimal for every dataset. Besides, the parameters for baselines are tuned to be optimal. For *HoloScope*, we set base b as 32. For *DeepWalk*, we set window size as 10, walk length as 40, walks per vertex as 40.

D. Detecting multiple fraudulent groups

We propose to use F_1-measure in order to give a comparison on three datasets with different number of injected blocks. we apply Eq. 13 to compute F_1-measure.

$$F_1 = \frac{2 \times (precision \times recall)}{precision + recall} \tag{13}$$

where $precision$ is the number of correct fraudulent nodes divided by the number of all fraudulent nodes returned by Alg. 2, and $recall$ is the number of correct fraudulent nodes results divided by the number of all fraudulent nodes.

Table IV reports the fraud detection results of the proposed method *FraudNE* and three state-of-the-art methods on three datasets. We separately measure the performance on source nodes and sink nodes. As we find that, *FraudNE* achieves the best F_1-measure among the competitors in most tasks. In particular, *FraudNE* achieves considerable improvement over other methods when the number of blocks is more than one. As MovieLens is much denser than other, *DeepWalk* gets higher F_1-measure on this dataset. In contrast, *FRAUDAR* achieves higher F_1-measure on sparse datasets. *HoloScope* obtains excellent performance on detecting one block. However, when the number of blocks increases, even with the number of blocks as a priori, *HoloScope* cannot achieve high performance on account of detecting most fraudulent nodes as one group. Table V quantitatively demonstrate our methods' ability of detecting multiple blocks.

Fig. 3 shows the result of *FraudNE*, *Holoscope* and *DeepWalk* on Zomato dataset. When the number of fraudulent blocks increased, the proposed method can still maintain high F_1-measure. Most importantly, *FraudNE* does not need to know the number of fraudulent blocks as a priori to achieve a better performance while competitors do.

E. Parameter Sensitivity

Our method involves several parameters, we examine how different numbers of the embedding dimension and different values of hyper-parameter α affect the performance on Zomato dataset with two injected blocks of *FraudNE*. Fig.4 (a) reports the performance of the *FraudNE* w.r.t. the dimension d. We can find that, at the beginning, the performance raises as the dimension d increases. However, the performance drops when the dimension d becomes too large. The main reason is that dimension d is so large that introduced noise would influence the performance. The value of α can balance three proximities. Fig.4 (b) shows how the value of α influences the performance. When $\alpha = 0.02$, we can obtain the best performance.

Fig.4 (c) reports the performance obtained by our method with regard to the hyper-parameter of *DBSCAN*, the minimum number of points required to form a dense region ϵ. When the hyper-parameter varies within a certain scope, the performance will not be influenced significantly, since FraudNE makes a large distinction between normal nodes and fraudulent nodes.

V. Related Work

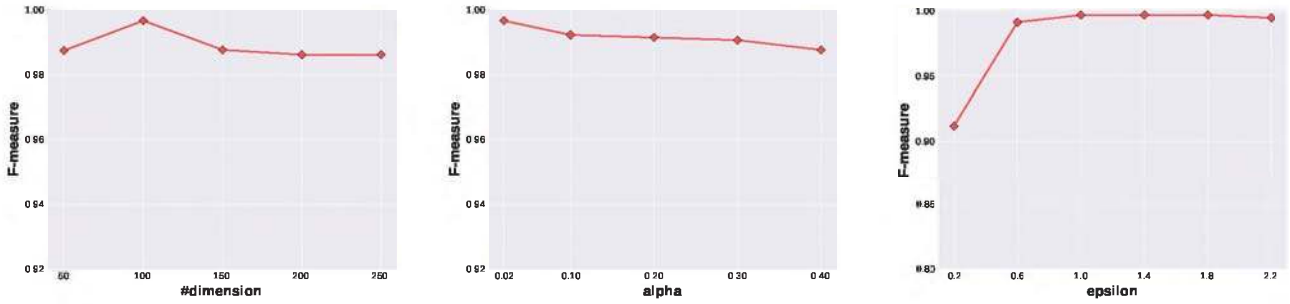
Anomaly detection. Since the groundbreaking work of [2], opinion spam has been the focus of research for more than ten years. Many existing methods aim to detect fraud through features. Various features have been proposed to characterize the differences between fraudsters and normal users. For example, deviation-based features [13], [14] are commonly used to capture the differences between the ratings given by fraudsters and the remaining users. Besides, review text can provide us with rich features, some existing method utilize text features such as content similarity, n-gram and the review length [15]. However, these approaches are not robust since the fraudsters constantly improve the quality of texts in reviews even without knowledge of detection system. It is unreliable to extract features and detect fraudsters through texts in reviews. Though the attributes of users or items can be manipulated by fraudsters, the connectivity structure of the bipartite network is still reliable and informative.

Graph-based fraud detection methods often detect unexpectedly dense regions of the networks. Since spammers unavoidably generated edges in networks when they create fake reviews, these methods are hard to evade. Most existing works study anomaly detection based on the density of blocks within adjacency matrices [16], [17] or multi-way tensors [18]. [19] builds on singular value decomposition (SVD) and focus on detecting attacks missed by spectral techniques. *FRAUDAR* [4] proposed a method to weight edge's suspiciousness by the inverse logarithm of objects' indegrees to discount popular objects. However, these methods adopt shallow models and cannot consider long-range, global structural information of networks.

Network embedding. As an emerging area from graph mining [20]–[22], network embedding aims to map the net-

TABLE IV: Experimental results on real data with injected labels

Data Name	F1 score of source nodes				F1 score of sink nodes			
	DeepWalk	FRAUDAR	HoloScope	FraudNE	DeepWalk	FRAUDAR	HoloScope	FraudNE
Zomato with block #1	0.1596	**0.9985**	0.9857	0.9975	0.9771	0.9965	**0.9985**	0.9975
Zomato with block #2	0.1306	-	0.8311	**0.9963**	0.3875	-	0.8075	**0.9975**
Zomato with block #3	0.1228	-	0.6634	**0.9958**	0.2723	-	0.3433	**0.9865**
Movielens with block #1	0.8555	0.1436	0.9546	**0.9840**	0.9875	0.1104	0.9625	**0.9954**
Movielens with block #2	0.8236	-	0.5422	**0.9824**	0.9817	-	0.50	**0.9918**
Movielens with block #3	0.8406	-	0.4484	**0.9826**	0.9889	-	0.3333	**0.9958**
BookCrossing with block #1	0.2036	0.9957	**0.9958**	0.9950	0.6703	**0.9986**	0.9926	0.9858
BookCrossing with block #2	01381	-	0.6789	**0.9933**	0.1715	-	0.50	**0.9667**
BookCrossing with block #3	0.1194	-	0.5765	**0.9975**	0.0620	-	0.2425	**0.9343**



(a) #dimension (b) α (c) ϵ

Fig. 4: Sensitivity w.r.t. dimension d, the value of α and ϵ

TABLE V: Precision, recall and F_1 score of each block

Data Name	blocks	precision	recall	F_1 score
Zomato	fraudulent group 1	0.9983	0.9850	0.9916
	fraudulent group 2	0.9932	0.9883	0.9908
	fraudulent group 3	0.9973	0.9916	0.9958
MovieLens	fraudulent group 1	0.9957	0.9850	0.9924
	fraudulent group 2	0.9986	0.9767	0.9882
	fraudulent group 3	0.9976	0.9700	0.9848
BookCrossing	fraudulent group 1	0.9979	0.9350	0.9717
	fraudulent group 2	0.9979	0.9350	0.9717
	fraudulent group 3	0.9968	0.9567	0.9796

work into low-dimensional representations for a wide range of network analysis applications [23]–[28]. Most of them are task-independent. Some earlier works like *IsoMap* [29], Local Linear Embedding [30] first constructed the affinity graph and then solved the leading eigenvectors as the network representations. Many methods have been proposed for graph embedding later, including multidimensional scaling [31], stochastic neighbor embedding [32] and spectral methods [33]. These methods are proposed to compress graph while preserving certain properties. More recently, *DeepWalk* [12] combines random walk and skip-gram to learn network representations and successfully preserve the structural information but it lacks a clear objective function. *LINE* [34] first learns the representations of networks based on preserving the first-order and second-order proximity. All of these methods are shallow models and can not capture the highly non-linear structure in the bipartite network. [8] proposed a deep network embedding method, to preserve highly non-linear structural information in network. However, it can only play a big role in homogeneous networks. There are also some methods aiming to embed heterogeneous network, such as metapath2vec [35] and [36]. But both of them can not be directly used to detect fraudsters. Besides, few existing methods embed different types of vertexes jointly in the same latent space.

Network embedding based anomaly detection. There are some previous works utilizing network embedding for anomaly detection [37], [38]. [37] proposed a method to use the spectral embedding to reveal anomalous community structure across multiple sources. [38] adopts a new measure to evaluate the level of anomalousness and detects structural inconsistencies based on network embedding. Different from these works, the proposed method focus on detecting the groups of fraudsters and their target objects in bipartite networks.

VI. Conclusion

In this paper, we present a joint embedding approach to detecting fraud groups in bipartite networks. Specifically, we design a unsupervised deep model with multiple layers of nonlinear function to capture the highly non-linear fraudulent structural information. Learning latent representations of vertexes can be robust to sparse networks and easy to capture the structure of the behavior graph. Moreover, after obtaining latent representations, we utilize one of the most

popular clustering algorithms to cluster multiple fraudulent groups in latent space without the number of groups as a priori. Empirically, our extensive experimental results have demonstrated the effectiveness of our approach to multiple fraud detection compared with state-of-the-art methods. And our method achieves high accuracy whether on sparse or dense dataset.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China (No.2016YFB0801003), the NSFC (No. 61502479), the Youth Innovation Promotion Association CAS (No. 2017210), the MQNS Grant (No. 9201701203), and the MQ Enterprise Partnership Scheme Pilot Res. Grant (No. 9201701455). C. Zhou is the corresponding author.

REFERENCES

[1] J. Ye and L. Akoglu, "Discovering opinion spammer groups by network footprints," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2015, pp. 97–97.

[2] N. Jindal and B. Liu, "Opinion spam and analysis," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 2008, pp. 219–230.

[3] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 309–319.

[4] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 895–904.

[5] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[6] K. Shin, B. Hooi, and C. Faloutsos, "M-zoom: Fast dense-block detection in tensors with quality guarantees," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 264–280.

[7] S. Liu, B. Hooi, and C. Faloutsos, "Holoscope: Topology-and-spike aware fraud detection," in *ACM*, 2017, pp. 1539–1548.

[8] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu, "Structural deep embedding for hyper-networks," *CoRR*, vol. abs/1711.10146, 2017. [Online]. Available: http://arxiv.org/abs/1711.10146

[9] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[10] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[12] B. Perozzi, R. Alrfou, and S. Skiena, "Deepwalk: online learning of social representations," pp. 701–710, 2014.

[13] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, "Detecting product review spammers using rating behaviors," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 939–948.

[14] A. Mukherjee, B. Liu, and N. Glance, "Spotting fake reviewer groups in consumer reviews," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 191–200.

[15] F. Li, M. Huang, Y. Yang, and X. Zhu, "Learning to identify review spam," in *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 2011, pp. 2488–2493. [Online]. Available: https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-414

[16] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Inferring strange behavior from connectivity pattern in social networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2014, pp. 126–138.

[17] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos, "Eigenspokes: Surprising patterns and scalable community chipping in large graphs," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 435–448.

[18] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, "D-cube: Dense-block detection in terabyte-scale tensors," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, 2017, pp. 681–689. [Online]. Available: http://dl.acm.org/citation.cfm?id=3018676

[19] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos, "Spotting suspicious link behavior with fbox: An adversarial perspective," in *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 959–964.

[20] J. Wu, S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Multiple structure-view learning for graph classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–16, 2018.

[21] J. Wu, Z. Hong, S. Pan, X. Zhu, Z. Cai, and C. Zhang, "Multi-graph-view learning for graph classification," in *2014 IEEE International Conference on Data Mining*, Dec 2014, pp. 590–599.

[22] S. Pan, J. Wu, X. Zhu, G. Long, and C. Zhang, "Finding the best not the most: regularized loss minimization subgraph selection for graph classification," *Pattern Recognition*, vol. 48, no. 11, pp. 3783–3796, 2015.

[23] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 1895–1901.

[24] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *ACM on Conference on Information and Knowledge Management*, 2017, pp. 889–898.

[25] L. Gao, J. Wu, C. Zhou, and Y. Hu, "Collaborative dynamic sparse topic regression with user profile evolution for item recommendation." in *AAAI*, 2017, pp. 1316–1322.

[26] C.-Y. Liu, C. Zhou, J. Wu, H. Xie, Y. Hu, and L. Guo, "Cpmf: A collective pairwise matrix factorization model for upcoming event recommendation," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1532–1539.

[27] C.-Y. Liu, C. Zhou, J. Wu, Y. Hu, and L. Guo, "Social recommendation with an essential preference space," in *AAAI*, 2018.

[28] C. Zhou, W. Lu, P. Zhang, J. Wu, Y. Hu, and L. Guo, "On the minimum differentially resolving set problem for diffusion source inference in networks." in *AAAI*, 2016, pp. 79–86.

[29] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[30] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[31] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[32] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Advances in neural information processing systems*, 2003, pp. 857–864.

[33] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[34] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," vol. 2, pp. 1067–1077, 2015.

[35] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 135–144.

[36] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128.

[37] J. Gao, W. Fan, D. Turaga, S. Parthasarathy, and J. Han, "A spectral framework for detecting inconsistency across multi-source object relationships," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 1050–1055.

[38] R. Hu, C. C. Aggarwal, S. Ma, and J. Huai, "An embedding approach to anomaly detection," in *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 2016, pp. 385–396.