Learning Stable Graphs from Multiple Environments with Selection Bias

Yue He
Tsinghua University
heyue18@mails.tsinghua.edu.cn

Peng Cui
Tsinghua University
cuip@tsinghua.edu.cn

Jianxin Ma
Tsinghua University
jason.mjx@alibaba-inc.com

Hao Zou
Tsinghua University
ahio@163.com

Xiaowei Wang
Alibaba
daemon.wxw@alibaba-inc.com

Hongxia Yang
Alibaba
yang.yhx@alibaba-inc.com

Philip S. Yu
University of Illinois at Chicago
psyu@cs.uic.edu

ABSTRACT

Nowadays graph has become a general and powerful representation to describe the rich relationships among different kinds of entities via the underlying patterns encoded in its structure. The knowledge (more generally) accumulated in graph is expected to be able to cross populations from one to another and the past to future. However the data collection process of graph generation is full of known or unknown sample selection biases, leading to spurious correlations among entities, especially in the non-stationary and heterogeneous environments. In this paper, we target the problem of learning stable graphs from multiple environments with selection bias. We purpose a Stable Graph Learning (SGL) framework to learn a graph that can capture general relational patterns which are irrelevant with the selection bias in an unsupervised way. Extensive experimental results from both simulation and real data demonstrate that our method could significantly benefit the generalization capacity of graph structure.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**;

KEYWORDS

Graph Structure, Stability, Multiple Environments, Selection Bias

ACM Reference Format:
Yue He, Peng Cui, Jianxin Ma, Hao Zou, Xiaowei Wang, Hongxia Yang, and Philip S. Yu. 2020. Learning Stable Graphs from Multiple Environments with Selection Bias. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403270

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403270

1 INTRODUCTION

Graph is a general and powerful representation to describe the rich relationships among different kinds of entities. The relational patterns (or knowledge more generally) it encodes are often exploited to facilitate various tasks. Therefore, many application fields are enthusiastic in generating application-specific graphs. In a recommendation system, for example, a graph is usually constructed and accumulated to reflect the purchasing patterns of products based on historical user purchasing behaviors, and this graph can be exploited in identifying product collections, predicting future purchasing behaviors, making promotion strategies and so on. In such cases, the quality of the graph is of huge impact on the subsequent tasks.

In addition to the traditional way of manually defining a graph by domain experts, the recent trend is to derive a graph from a collection of data in a data-driven manner. Among others, the most commonly used is to measure the co-occurrence frequency among entities [23] and heuristically build edges linking highly co-occurred entities. Some more advanced methods take the subsequent task into the loop and exploit the supervised information from the end task to further refine the generated graph through a learning framework[14]. No matter which strands, all these data-driven methods implicitly assume that the data collection for graph generation is sufficiently representative with respect to the whole data population, or at least, the two data collections for generating and exploiting the graph respectively should be independent and identically distributed (I.I.D.). Only under this assumption, the relational patterns encoded in the generated graph may still hold in the subsequent tasks.

In practice, however, this assumption can hardly be satisfied. The data collection process is usually full of known or unknown sample selection biases, leading to spurious correlations among entities. Take recommendation system as an example. In constructing a product co-purchasing graph, the collected data may consist of much more female user behaviors than male. Then the resulted graph generated by existing methods will induce much bias in predicting male purchasing behaviors (if we suppose a large discrepancy between female and male purchasing behavior patterns). This problem is more serious in non-stationary and heterogeneous environments

such as different geographical regions, diverse demographic populations, or long time-span. As we usually ignore the original data distribution where we derive the graph when exploiting it, it is highly demanded to scrutinize the stability of the generated graph across different environments.

In recent years, stable (or invariant) learning[11, 17, 26] arouses considerable research interests. These methods aim to learn a stable function from input variables to the output variable which is consistent across unknown distribution shifts, but few of them study the problem in the context of graph structure. In this paper, we target the problem of learning stable graphs from multiple environments with selection bias. We suppose the data availability from multiple environments with different sample selection biases, and aim to learn a graph that capture general relational patterns that are irrelevant with these selection biases. We expect that the learned graph can be applied into multiple, even unknown environments and help to produce stable performances with subsequent tasks. Meanwhile, the stable graph should be more explainable as it contains less bias and thus less spurious correlations than conventional generated graphs. In addition, as we do not target any specific subsequent tasks, the learning of stable graphs should be in an unsupervised way.

Here we propose a Stable Graph Learning (SGL) framework to accomplish these goals. For more generality, we design the framework for set data, where a data sample is represented by a set (e.g. a shopping basket in recommendation systems) consisting of several elements (e.g. products in recommendation systems). The framework is composed by three modules, including a graph construction module to generate a graph, a graph convolutional network (GCN) to learn embeddings for elements and transform a set into a pooled embedding vector (i.e. a set vector), as well as an element-wise variational auto-encoder (E-VAE) to reconstruct the original input set.

The parameters in the three modules are jointly optimized in a iterative way, but, conceptually, we can decompose the training process into two phases. In the first phase, we fix the graphs constructed in each environment, and update parameters in GCN and E-VAE, enabling the two modules with the capability to generate the set data in the corresponding environment from the constructed graph. In the second phase, we fix the parameters in GCN and E-VAE, and refine the graph structure to make the probability of generating a set equals to the average probability of generating the set over all environments. Assuming the environments are randomly selected with different selection bias, the resulted graph structure contains less bias than every single environment, and consequentially possess the superiority in producing stable performance across different environments in subsequent tasks.

The main technological contributions in this paper are summarized as below:

(1) We investigate the problem of learning stable graphs from multiple environments with selection bias, which is of paramount significance in both research and applications but rarely studied in literature.
(2) We propose a simple yet effective framework SGL for learning stable graphs from set data, which is composed by a GCN module and a subtly designed element-wise VAE (E-VAE) for high-dimensional sparse set data.
(3) Extensive experiments are conducted in both simulation and real data, and the results show that the graph learned by SGL can indeed perform better than other graphs generated by conventional methods in terms of generalization across different environments.

The rest of this paper is organized as follows. Section 2 reviews the works of the related fields. Section 3 introduces our Stable Graph Learning framework. Section 4 gives the experimental results to show the effectiveness of our model. Finally, we concludes this work at the end of paper.

2 RELATED WORKS

In this section, we review the works of some related fields with this work, including Graph Convolutional Networks, the deep generative models and the graph generation task.

In the past few years, Graph Convolutional Networks (GCN)[8, 16] have become the major technology to catch patterns encoded in the graph because of its powerful capacity via deep learning. Like Convolutional Neural Networks (CNN)[20], the fundamental models for processing images and texts, GCNs introduce the convolution operation into graph data by using the graph Laplacian matrix in the spectral domain[8]. Through stacking the convolution layers, the information would be propagated and aggregated among the nodes to capture the structural characteristics. To improve the performance of GCNs, further works are proposed, e.g. [28, 29] add attention mechanism when aggregating node neighborhoods; [21] aims to learn node embeddings with disentangled semantics; [5, 6] use sampling strategy to improve the computational efficiency of algorithms. For applications, a variety of GCNs are designed to handle with specific tasks, including the node classification[2], graph generation[7], network embedding[33] and etc.

Another technological line is the deep generative models that are used to learn the joint distribution of observed data. Generative Adversarial Networks (GAN)[10] optimize the generator together with a discriminator in a thought of zero-sum game. In constrast, AutoEncoder (AE)[13] learns the implicit distribution in a more direct way: an encoder could compress the raw data into a latent vector, then a decoder is to recover the data from the latent vector conversely through a neural network. The two modules would be optimized under a reconstruction loss between the real input and the recovered output. To solve the monotonicity of generated samples in AE, the Variational AutoEncoder (VAE)[15] is proposed to replace the latent vector with a latent distribution that should be approximate to a prior distribution and fed a sampled vector to the decoder. In this paper, we subtly design an element-wise VAE (E-VAE) for high-dimensional sparse set data generation.

On the other hand, to generate graphs to capture rich relationships in real-world has been studied in a flood of literature. Trained on a representative set of graphs from particular applications, MolGAN[7] utilizes a permutation-invariant discriminator to learn the implicit distribution of molecular graphs in the GAN framework with a reward network to gain the desired chemical properties; [22] proposes a regularized VAE to encourage to generate semantically valid graphs satisfied with constraint formulations;

GraphRNN[30] could generate variable-sized graph by adding new nodes and edges in a BFS ordering sequence recurrently. Given partial node labels, GLCN[14] jointly optimizes the refined graph structure and the learnt graph convolution; GAT[28] employs attention mechanism to directly calculate the connective strengths between nodes via the supervised information in a specific task. But in the more general scenes, none of the graph examples and the specific tasks are provided, only the collection of frequent occurrence, such as set data, is available. To deal with these forms of data, NetGAN[3] requires the learnt graph to output random walks that could not be distinguished with the real samples in the GAN architecture. The methods based on co-occurrence construction like Adamic/Adar[1], are still the most widely used in industry yet because of their effectiveness and efficiency. However, all the graph generative models above are based on the I.I.D. assumption, and ignore the selection bias on the data collection, causing their generated graphs always fail in distribution-shifted environments.

Although a paucity of researchers try to address the problem in the framework of causal inference and causal graphs are exploited to guarantee stable performance across environments due to the invariance of causality[18, 25, 27, 31], the high complexity of causal discovery and causal graph generation makes it infeasible for high-dimensional situations in practice. Moreover, the resulted causal graphs are thought to be directed acyclic graphs[4, 32], that cannot cover the common graph types with complex cyclic structure in real world.

3 PROBLEM AND OUR METHOD

In this section, we introduce the problem we investigate and our proposed method in detail. Note that we assume the initial graph for each environment has been constructed by an I.I.D. graph generative model (co-occurrence based method in this paper) before the Stable Graph Learning framework.

3.1 Notations and Problem

The major notations used in this paper are standardized in Table1. Given a graph and set data $\{(\boldsymbol{G}^{(m)}, \mathbf{S}^{(m)})\}_{m=1}^{M}$ from M different environments, the task of Stable Graph Learning is to learn a graph $\boldsymbol{G}_S$ that represents the unbiased connective structure, e.g. the weighted adjacency matrix, over the K elements across all environments.

Because $\boldsymbol{G}^{(m)}$ is generated from $\mathbf{S}^{(m)}$, if the collection process of $\mathbf{S}^{(m)}$ comes from an environment with selection bias, the spurious correlations among the elements would cause $\boldsymbol{G}^{(m)}$ to performance poorly in other environments. We could explain this via the reverse generation process from $\boldsymbol{G}^{(m)}$ to $\mathbf{S}^{(m)}$.

The generation process of a set data can be described as adding the elements into the current set one-by-one, which is similar to the sequence generation in the literature of nature language process[19]. Beginning from an empty set, a steady-state set would be achieved eventually after rounds of transfer (one step of addition) according to the relational patterns encoded in graph. We can formalize the one transfer as a function (Eq.(1)) of the probability $P^{(m)}(\cdot|\cdot)$ for each element I_k to be selected conditioned on the current set $\mathbf{s}$ on the basis of graph $\boldsymbol{G}^{(m)}$, where h is an inherent function and m indicates the index of environment.

Table 1: Notation and Definitions

Notation	Annotation
K	the number of elements (nodes).
M	the number of environments.
I_k	the k_{th} element in the environment.
$\boldsymbol{G}^{(m)}$	the relative graph in the m_{th} environment.
$\mathbf{s}_i^{(m)} \in \{0,1\}^{1\times K}$	the i_{th} set data in the m_{th} environment. All the positions with value 1 represent the corresponding elements are in the set.
$\mathbf{X}^{(m)} \in \mathbb{R}^{K\times F}$	the embedding matrix in m_{th} environment.
F	the dimension of node embedding.
L	the size of batch data.

$$P^{(m)}(I_k|\mathbf{s}) = h(\boldsymbol{G}^{(m)}, I_k, \mathbf{s}). \tag{1}$$

Due to the distribution shift, with access to $\{(\boldsymbol{G}^{(m)}, \mathbf{S}^{(m)})\}_{m=1}^{M}$ from M environments, $P^{(m)}(I_k|\mathbf{s})$ always changes from $m = 1$ to $m = M$, implying the unstable connective structure in graph. However, we could infer the unbiased probability $P_S(I_k|\mathbf{s})$ if the provided environments are randomly chose as following :

$$P_S(I_k|\mathbf{s}) = \sum_{m=1}^{M} \frac{P^{(m)}(I_k|\mathbf{s})}{M}. \tag{2}$$

Obviously, a graph $\boldsymbol{G}_S$ is said to be unbiased if it can satisfy the unbiased generative probability. Although it's quite difficult to correct the graph structure on original parameter space (e.g. the weighted adjacency matrix) due to the underlying complicated correlations, Eq.(2) enlightens us to do it indirectly via eliminating the bias on probability space conveniently after transforming the graph structure to the generative probability. To achieve this goal, we propose an unsupervised SGL framework to learn the stable graph from multiple environments with selection bias, which can be decomposed into two steps including graph-based set generation and stable graph learning. These two steps are jointly optimized in our SGL framework. But for ease of understanding, we introduce them separately in following sections.

3.2 Graph Based Set Generation

Given an environment, we first model the function h, an inherent high-order and non-linear generative function, from the graph $\boldsymbol{G}^{(m)}$ to the sparse set data $\mathbf{S}^{(m)}$. To capture the complex structural patterns among the nodes, a GCN module is applied on the graph to embed the local structural information of a node into its representation. The representation $\mathbf{x}_k^{(l+1)}$ of the node k in the $(l+1)^{th}$ layer is transformed from the l^{th} convolution layer as

$$\mathbf{x}_k^{(l+1)} = \sigma\Big(\sum_{j\in N_k} \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{x}_j^{(l)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}\Big), \tag{3}$$

where σ is a non-linear activation function, $\hat{\mathbf{A}}$ is the adjacency matrix with self loop in graph, $\hat{\mathbf{D}}$ is the degree matrix of $\hat{\mathbf{A}}$, N_k represents all the neighbours of node k (including node k itself), and $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the parameters of the l^{th} convolution layer.

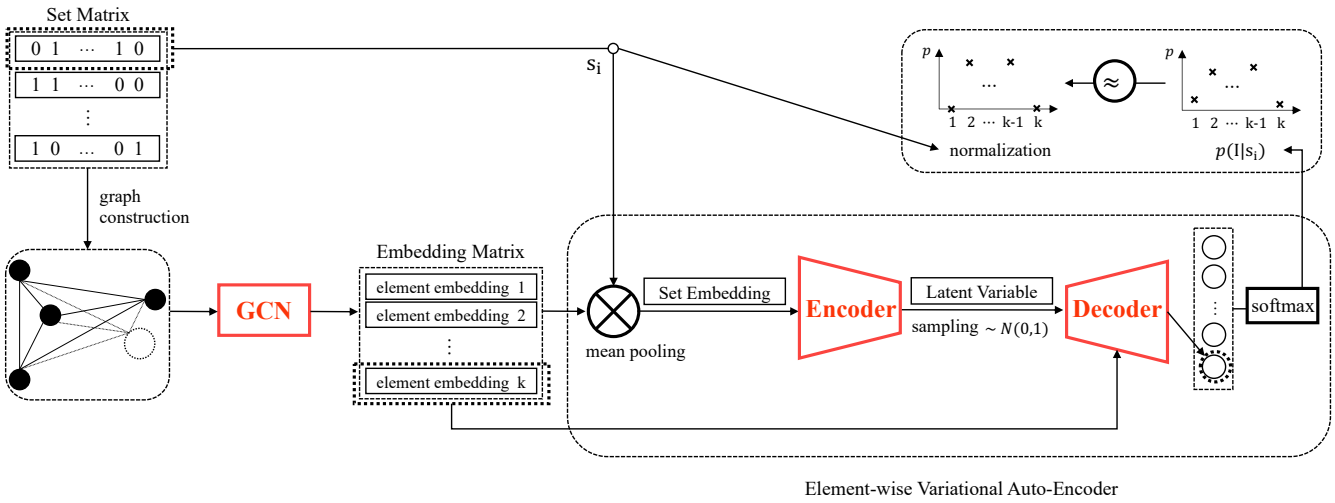


Figure 1: The illustration of graph based set generation. The set vector is the output of a mean pooling operation of the element embedding matrix learnt by GCN from constructed graph and the real set data. Each element embedding concatenates with the sampled latent vector and is inputted into the decoder separately. Combining all the decoder's outputs ("selective intensity") together, we obtain the conditional probability space over the all elements after a softmax layer. By maximizing likelihood of observed data, we can optimize parameters of GCN, encoder and decoder in every single environment.

Through the stacked multiple convolution layers, we can obtain the node embedding matrix $\mathbf{X}^{(m)}$ that would contain the relationships among nodes in graph $\boldsymbol{G}^{(m)}$.

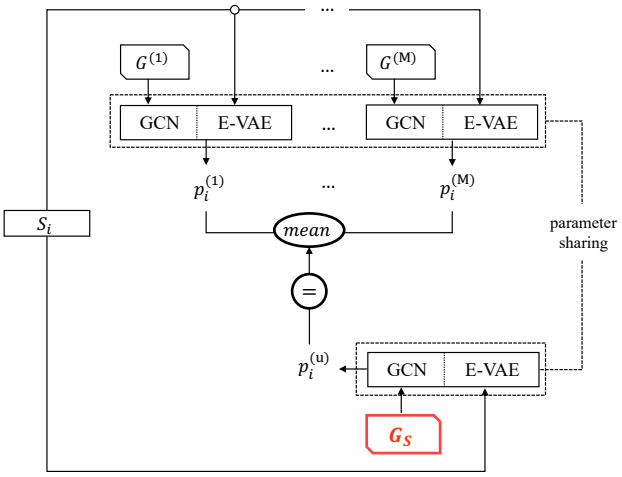


Figure 2: The illustration of stale graph learning process. Sharing the same GCN and E-VAE across M environments, we can fix their parameters after optimization and learn the stable graph G_S afterwards. Sampling arbitrary set S_i, we then obtain the generative probabilities $\{P_i^{(m)}\}_{m=1}^{M}$ ($P_i^{(m)}$ is short for $P^{(m)}(\mathrm{I}|\mathrm{s}_i)$) on the basis of different graph structures in all the environments, as well as $P_i^{(u)}$ underlain in G_S from a virtualized unbiased environment. The G_S would be updated so that $P_i^{(u)}$ can be equal to the mean of $\{P_i^{(m)}\}_{m=1}^{M}$.

Based on the derived $\mathbf{X}^{(m)}$, we then learn the generation mechanism of set data, i.e. the conditional probability $P^{(m)}(I_k|\mathbf{s})$, in the way of reconstructing the distribution of real $\mathrm{S}^{(m)}$. In practice, the set data is often high-dimensional and sparse. In recommendation system, for example, a shopping basket only include very small number of products compared with the whole product pool. Therefore, a set in such cases is in a very high dimensionality and only several elements are with value 1. These high-dimensional sparse sets impose tremendous challenge to a generative model like VAE, as the commonly used norm-based reconstruction loss functions are not sensitive enough to effectively differentiate similar or dissimilar pair of high-dimensional sparse vectors. Here we subtly propose an element-wise variational autoencoder (E-VAE) to address this challenge, which reconstructs $\mathrm{S}^{(m)}$ from the embeddings of the elements in the set in an element-by-element way. The detailed framework of graph based set generation is shown in Figure1.

For each optimization iteration, we sample a real set $\mathbf{s}_i^{(m)}$ from $\mathrm{S}^{(m)}$. To better express the $\mathbf{s}_i^{(m)}$, we project it into the element embedding space $\mathbf{X}^{(m)}$ (learnt from $\boldsymbol{G}^{(m)}$) by mean pooling over the embeddings of the elements that are in the set, i.e. $\frac{\mathbf{s}_i^{(m)} \cdot \mathbf{X}^{(m)}}{\mathbf{s}_i^{(m)} \cdot \mathbf{1}}$, and input it into the encoder of E-VAE. The encoder finally learns a mean value vector $\boldsymbol{\mu}$ and a variance vector $\boldsymbol{\varphi}$ to represent a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\varphi})$ of the latent variable $\mathcal{Z}_i$.

For decoder, we sample a vector $\mathbf{z}_i$ from the normal distribution of $\mathcal{Z}_i$ firstly. Due to the non-sequential characteristics of set generation, we input the concatenation of the latent variable $\mathbf{z}_i$ and the embedding of each node into the decoder separately to obtain the intensity of each element to be selected by $\mathbf{z}_i$. Then the intensity values of all elements are combined together into a vector which is further fed into a softmax layer. The k_{th} output of softmax layer

is eventually $P^{(m)}(I_k|\mathbf{s}_i^{(m)})$, the probability of k_{th} element to be selected conditioned on the real set $\mathbf{s}_i^{(m)}$.

Because $\mathbf{s}_i^{(m)}$ is a piece of real data, we could suppose it is steady-state. That says the next element most likely to be selected should be among the ones that have appeared in $\mathbf{s}_i^{(m)}$, the input set should be restored after one transfer round of element addition. Hence, we aim to maximize the likelihood defined by

$$Maximize \prod_{I_k \in \mathbf{s}_i^{(m)}} P^{(m)}(I_k|\mathbf{s}_i^{(m)}). \tag{4}$$

To solve the non-sensitive problem of discriminating sparse high-dimensional vectors, we adopt the negative log likelihood loss. Referring to [12], the objective function can be written as below :

$$\mathcal{L}_{rc} = -\mathbf{s}_i^{(m)} \cdot log[P^{(m)}(I_k|\mathbf{s}_i^{(m)})]. \tag{5}$$

Besides the reconstruction loss $\mathcal{L}_{rc}$, the latent variable $\mathcal{Z}_i$ should be also constrained to satisfy the standard normal distribution. The loss function $\mathcal{L}_{kl}$ is to minimize the KL divergence between the output $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\varphi})$ and $\mathcal{N}(\mathbf{0}, \mathbf{1})$ as widely used in VAE models.

$$\mathcal{L}_{kl} = -0.5 * (1 + log(\varphi) - \mu^2 - \varphi). \tag{6}$$

3.3 Stable Graph Learning

Given $\{(\boldsymbol{G}^{(m)}, \mathrm{S}^{(m)})\}_{m=1}^{M}$ from M environments, we share the same parameters of GCN and E-VAE for set generation across all the environments, because the generative function h is inherent and independent of all the environments. Such design could also prevent the overfitting problem of model learning. By reconstructing the real set data in each environment based on its graph structure, we could achieve the generative probability $\{P^{(m)}(I_k|\mathbf{s})\}_{m=1}^{M}$ of every element in all environments conditioned on arbitrary set $\mathbf{s}$, as well as the parameters of GCN and E-VAE.

Up to now, we have transformed the biased graph structures from raw parameter space into different generative probability spaces of M environments. With access to these, we can indirectly learn an stable graph $\boldsymbol{G}_S$ to meet the constraint in Eq.(2) much more conveniently. In detail, we first virtualize an environment without selection bias and initial the graph structure (weighted adjacency matrix) of $\boldsymbol{G}_S$. Then we alternately choose m_{th} environment and randomly sample a $\mathbf{s}_i^{(m)}$ from it. The real set $\mathbf{s}_i^{(m)}$ together with each of $\{\boldsymbol{G}^{(m)}\}_{m=1}^{M}$ and $\boldsymbol{G}_S$ are inputted into the architecture of GCN and E-VAE to obtain the $\{P^{(j)}(\mathbf{I}|\mathbf{s}_i^{(m)})\}_{j=1}^{M}$ and $P_S(\mathbf{I}|\mathbf{s}_i^{(m)})$ respectively, where $P(\mathbf{I}|\mathbf{s}_i^{(m)})$ is the output vector of softmax layer that represents the conditional probability space over the all elements. Finally $\boldsymbol{G}_S$ could be directly optimized by minimizing the refining objective loss function $\mathcal{L}_{rf}$ in the Eq.(7).

$$\mathcal{L}_{rf} = ||P_S(\mathbf{I}|\mathbf{s}_i^{(m)}) - \sum_{j=1}^{M} \frac{P^{(j)}(\mathbf{I}|\mathbf{s}_i^{(m)})}{M}||_2 \tag{7}$$

Because the learning of GCN, E-VAE and the stable graph G_S would interact on each other closely, we joint optimize them in practice as shown in Figure2. The overall objective function is a combination of $\mathcal{L}_{rc}$ and $\mathcal{L}_{kl}$ and $\mathcal{L}_{rf}$ as in Eq.(8).

Algorithm 1 Stable Graph Learning (SGL) framework

Input: $\{(\boldsymbol{G}^{(m)}, \mathrm{S}^{(m)})\}_{m=1}^{M}$ from M different environments
Output: Stable graph structure $\boldsymbol{G}_S$ over K elements, non-linear parameters θ of GCN and φ of VAE
Initial $\boldsymbol{G}_S$, θ and φ
while not converged **do**
  Compute the element embedding pool $\{\mathbf{X}^{(m)}\}_{m=1}^{M}$ of M environments and $\mathbf{X}_S$ via $\{\boldsymbol{G}^{(m)}\}_{m=1}^{M}$, the current $\boldsymbol{G}_S$ and θ
  for $m = 1$ to M **do**
    Sample batch of set data $\{\mathbf{s}_i^{(m)}\}_{i=1}^{L}$
    Obtain $P_S(\mathbf{I})$ conditioned on $\{\mathbf{s}_i^{(m)}\}_{i=1}^{L}$ via $\mathbf{X}_S$ and φ
    for $j = 1$ to M **do**
      Obtain $P^{(j)}(\mathbf{I})$ conditioned on $\{\mathbf{s}_i^{(m)}\}_{i=1}^{L}$ via $\mathbf{X}^{(j)}$ and φ
    end for
    Calculate total $\mathcal{L}^{(m)}$ in m_{th} environment as in Eq.(8)
  end for
  $\mathcal{L} = \mathcal{L}^{(1)} + ... + \mathcal{L}^{(M)}$
  Optimize $\boldsymbol{G}_S$, θ and φ to minimize $\mathcal{L}$
end while
return: $\boldsymbol{G}_S$, θ, φ

$$\mathcal{L} = \mathcal{L}_{rc} + \alpha\mathcal{L}_{kl} + \beta\mathcal{L}_{rf}, \tag{8}$$

where α and β are hyper-parameters. And we summary the whole SGL framework in the Algorithm1.

4 EXPERIMENT

To verify the disadvantage of biased graph structure and the effectiveness of our SGL framework on learning the stable graph from multiple environments with selection bias, we carry out both simulation experiments and real data experiments. All the experiments conducted in this paper, as well as the model of GCN and E-VAE, are implemented in Pytorch[24].

4.1 Baselines and Evaluation Metrics

In our study, only the set data is available, and the initial graph structure is easily obtained from data by a predefined generative model. For comparison, our baselines include the graphs $\{\boldsymbol{G}^{(m)}\}_{m=1}^{M}$ constructed using the data in single environment only, the average result of these graphs $\boldsymbol{G}_A = \sum_{m=1}^{M} \frac{\boldsymbol{G}^{(m)}}{M}$ and the graph $\boldsymbol{G}_C$ that is generated from the union of data in all environments (uniformly composed). Lastly, the $\boldsymbol{G}_S$ is the learnt graph by SGL framework.

To evaluate the stability of the learnt graph structure, we design a set prediction task based on the node embeddings. A piece of set data $\mathbf{s}$ could be split into a target element I_t and the remaining elements $\mathbf{s}-\{I_t\}$. The set prediction task is to recover $\mathbf{s}$ given $\mathbf{s}-\{I_t\}$, that is to select I_t from all the elements except for the elements in the given set $\mathbf{s} - \{I_t\}$. We define the distance of one candidate I_c to the given set as the mean **Cosine Distance** between the node embeddings of I_c and the elements in the given set as following :

$$Disc(I_c, \mathbf{s} - \{I_t\}) = \frac{1}{|\mathbf{s} - \{I_t\}|} \sum_{I_j \in \mathbf{s} - \{I_t\}} Cosine(\mathbf{x}_t, \mathbf{x}_j). \tag{9}$$

If I_c is in the top-K elements with the minimal distance to the given set, we says I_c is retrieved successfully. Then we calculate the top-K accuracy of successful retrieval based on node embeddings learnt from graphs generated by different methods. For fair comparison, we keep the same parameters of embedding learning model for all the graphs.

In the later sections, we explain the experimental setup and results on simulation data and real data in detail.

4.2 Simulation Experiment

For brevity, we assume the number of available environments is two in the following state about experiments.

4.2.1 Biased weighted random walk. Weighted random walk is to generate a path by randomly walking among the nodes based on the edge weights in the graph. The probability $\pi_{v,u}$ of moving from node v to u is defined as:

$$\pi_{v,u} = \frac{w_{v,u}}{\sum_{u' \in N_v} w_{v,u'}}, \tag{10}$$

where $w_{v,u}$ is the weight of edge (v, u) and N_v is the set of neighbours of node v. To introduce the distribution shift in two data environments, we make biased weighted random walk (*BWRW* for brevity) to simulate the generation of set data. Firstly we introduce the high-order correlation and redefine the $\pi'_{v,u}$ as in the Eq.(11), where $Q(t(v), v, u)$ is a coefficient depending on the node v, u and the front node $t(v)$ of v.

$$\pi'_{v,u} = \frac{Q(t(v), v, u) * w_{v,u}}{\sum_{u' \in N(v)} Q(t(v), v, u') * w_{v,u'}}. \tag{11}$$

Obviously, the assignment of different value to Q can cause the underlying relationships change in generated data. According to the Q function defined in Eq.(12), where $q_2 > q_1 > 1$ and $d(t(v), u)$ represents the shortest distance between $t(v)$ and u, a node prefers to move to the nodes with the same type. If two nodes of the same type in environment 1 are of different types in environment 2, it equals to make an intervention on selection bias in the two environments, leading to the shift of data distribution.

$$Q(t(v), v, u) = \begin{cases} w_{v,u} * q_2 & If\ d(t(v), u) <= 1 \quad and \\ & t(v),\ v,\ u\ are\ the\ same\ type \\ w_{v,u} * q_1 & If\ v,\ u\ are\ the\ same\ type \\ w_{v,u} & Otherwise \end{cases} \tag{12}$$

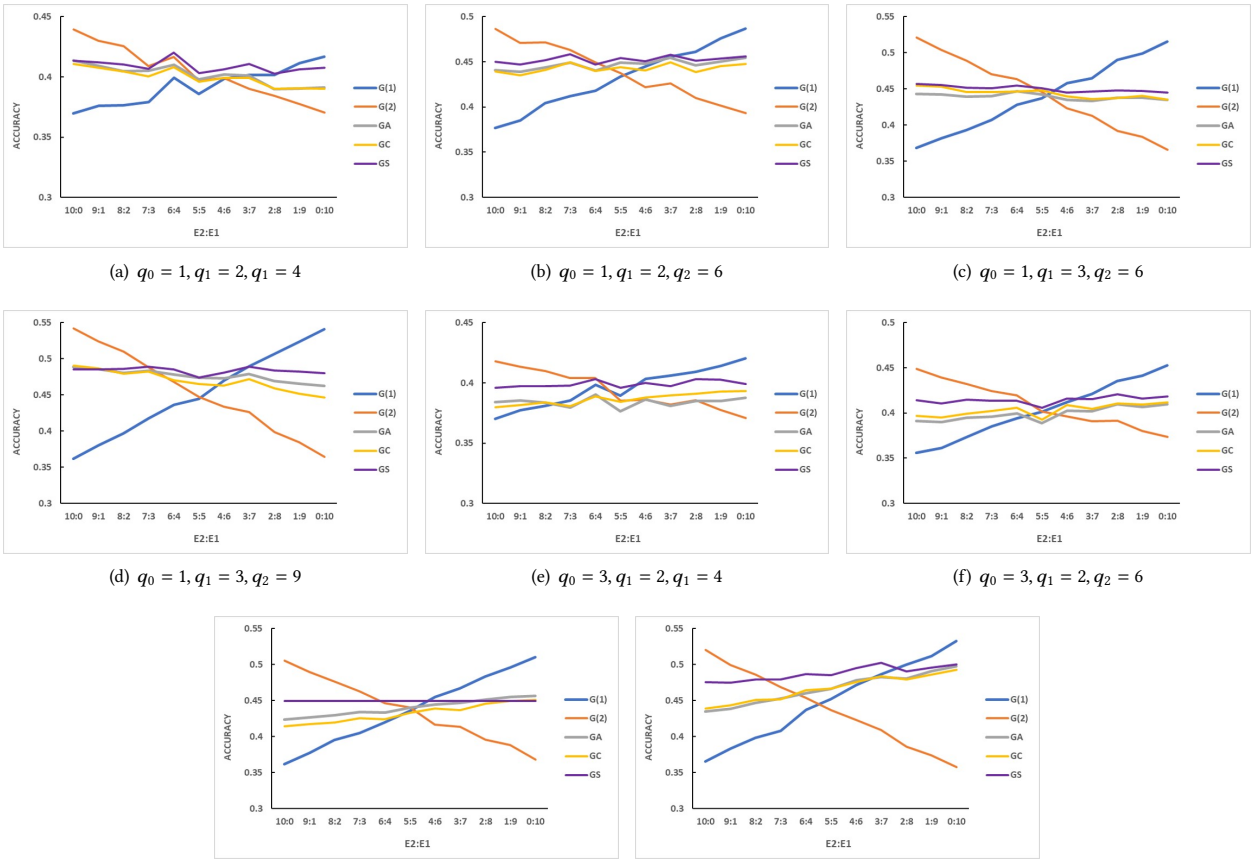
4.2.2 Experimental setup. Initially, we create an undirected graph with 100 nodes. Every node pairs is connected with 50% probability and the weight of connected edge is assigned uniformly with value between 1 and 100. All the 100 nodes are divided equally into two types in both environments, and half of them have different types in the two environments. In addition to the first-order and second-order bias controlled by q_1 and q_2, we also introduce the zero-order bias represented by q_0, that simulates the prior preference of data selection, e.g. the female users are more concerned with the skin care products than the males. For $q_0 > 1$, all the nodes always have equal probability to become a starting node in environment 1, but in environment 2 the probability of the nodes of type 1 being the starting node are q_0 times than that of the nodes of type 2.

MEAN of ACCURACY				
	$q_0 = 1$			
	$q_1 = 2, q_2 = 4$	$q_1 = 2, q_2 = 6$	$q_1 = 3, q_2 = 6$	$q_1 = 3, q_2 = 9$
$G^{(1)}$	39.24%	43.23%	44.03%	45.18%
$G^{(2)}$	40.36%	43.93%	44.29%	45.34%
G_A	40.14%	44.69%	43.94%	47.62%
G_C	39.98%	44.28%	44.38%	46.96%
G_S	**40.91%**	**45.27%**	**45.02%**	**48.38%**
	$q_0 = 3$			
	$q_1 = 2, q_2 = 4$	$q_1 = 2, q_2 = 6$	$q_1 = 3, q_2 = 6$	$q_1 = 3, q_2 = 9$
$G^{(1)}$	39.58%	40.31%	43.70%	44.97%
$G^{(2)}$	39.43%	40.89%	43.67%	43.78%
G_A	38.40%	39.92%	44.02%	46.64%
G_C	38.68%	40.34%	43.23%	46.68%
G_S	**39.90%**	**41.45%**	**44.99%**	**48.79%**
STD of ACCURACY				
	$q_0 = 1$			
	$q_1 = 2, q_2 = 4$	$q_1 = 2, q_2 = 6$	$q_1 = 3, q_2 = 6$	$q_1 = 3, q_2 = 9$
$G^{(1)}$	0.0149	0.0345	0.0473	0.0569
$G^{(2)}$	0.0214	0.0299	0.0491	0.0560
G_A	0.0079	0.0052	**0.0039**	0.0079
G_C	0.0071	0.0046	0.0064	0.0134
G_S	**0.0048**	**0.0037**	0.0040	**0.0041**
	$q_0 = 3$			
	$q_1 = 2, q_2 = 4$	$q_1 = 2, q_2 = 6$	$q_1 = 3, q_2 = 6$	$q_1 = 3, q_2 = 9$
$G^{(1)}$	0.0155	0.0310	0.0472	0.0532
$G^{(2)}$	0.0152	0.0241	0.0422	0.0509
G_A	0.0036	0.0074	0.0109	0.0203
G_C	0.0046	0.0064	0.0125	0.0175
G_S	**0.0026**	**0.0037**	**0.0076**	**0.0093**

Table 2: The mean and std of top-30 accuracy of set prediction task across 11 testing datasets for 8 parameter groups of *BWRW*. The node embeddings learnt from G_S can achieve the higher mean accuracy more stably than the baselines in almost all the experiments. And the advantage of graph G_S is more remarkable as environments' discrepancy increases.

Determined by a group of q_0, q_1 and q_2, two environments of set data with distribution shift could be acquired by *BWRW*. Taking them as training datasets, we directly build up the graph of baselines, including $\boldsymbol{G}^{(1)}$, $\boldsymbol{G}^{(2)}$, $\boldsymbol{G}_A$ and $\boldsymbol{G}_C$, based on the co-occurrence generative method, where the edge weight of two nodes is the co-occurrence frequency of them. Also we learn the graph $\boldsymbol{G}_S$ by SGL framework. For evaluation, 11 testing datasets are produced by mixing the data of two environments with a proportion from 0:10 to 10:0. For credibility, we conduct kinds of experiments about stability based on different parameter groups (q_0, q_1, q_2) of *BWRW*. In the next, we'll elaborate them with the results.

4.2.3 Experimental result. We evaluate the performances of all the models in the designed set prediction task using the node embeddings learnt by GCN from different graphs. To keep the two data environments of testing datasets be the same with the training's



(a) $q_0 = 1, q_1 = 2, q_1 = 4$ (b) $q_0 = 1, q_1 = 2, q_2 = 6$ (c) $q_0 = 1, q_1 = 3, q_2 = 6$

(d) $q_0 = 1, q_1 = 3, q_2 = 9$ (e) $q_0 = 3, q_1 = 2, q_1 = 4$ (f) $q_0 = 3, q_1 = 2, q_2 = 6$

(g) $q_0 = 3, q_1 = 3, q_2 = 6$ (h) $q_0 = 3, q_1 = 3, q_2 = 9$

Figure 3: Top-30 accuracy of set prediction task using the node embeddings learnt from different graphs. Each subfigure corresponds to an experiment with the value of parameter group of *BWRW* below the subfigure. E1:E2 represents the mixing proportion of data from the two environments and the curves point to the baselines and the learnt graph by SGL. The purple curve to G_S is always more smooth than the baselines in 8 experiments.

Table 3: The mean and std of top-30 accuracy of set prediction task when the data environments of testing datasets are of different parameter groups of *BWRW* with the environments of training datasets.

TRAIN: $q_0 = 1, q_1 = 2, q_1 = 4$			TEST: $q_0 = 1, q_1 = 3, q_2 = 9$		
	$G^{(1)}$	$G^{(2)}$	G_A	G_C	G_S
MEAN	43.65%	43.87%	43.25%	43.09%	**44.02%**
STD	0.0364	0.0379	0.0082	0.0085	**0.0032**
TRAIN: $q_0 = 3, q_1 = 2, q_1 = 6$			TEST: $q_0 = 3, q_1 = 3, q_2 = 6$		
	$G^{(1)}$	$G^{(2)}$	G_A	G_C	G_S
MEAN	42.54%	42.00%	41.99%	41.95%	**43.57%**
STD	0.0351	0.0316	0.0046	0.0054	**0.0038**
TRAIN: $q_0 = 1, q_1 = 3, q_1 = 6$			TEST: $q_0 = 3, q_1 = 3, q_2 = 9$		
	$G^{(1)}$	$G^{(2)}$	G_A	G_C	G_S
MEAN	45.01%	43.91%	45.45%	45.55%	**46.56%**
STD	0.0511	0.0462	0.0043	0.0041	**0.0039**

firstly, we experiment on 8 parameter groups of *BWRW*. It can be observed that the performance of the graph built in single environment rapidly declines as the data proportion from another environment increases in Figure3. And the G_A, G_C and G_S could keep relatively stable when the testing distribution changes. However, the performance of G_S is much more stable and achieves higher mean accuracy than all the baseline in almost all the experiments, especially when the discrepancy between the two environments is more significant, seen in Table2. That is because G_S directly learn the less biased high-order and non-linear correlations among the nodes at the level of generation probability from two environments (G_A only balances the linear part), so that it could also ignore the prior preference of data selection (controlled by q_0) and reduce the model bias of the initial graph generative method to some extent (G_C still suffers from the both biases).

Moreover we explore the situations when the testing data environments are of different parameters groups of *BWRW* with the training environments. The G_S is expected to keep its superiority even if the training datasets can not cover the testing distribution

	Mean ACC	STD	Env1:Env2=0:10	1:9	2:8	3:7	4:6	5:5	6:4	7:3	8:2	9:1	10:0
$G^{(1)}$	12.93%	**0.0050**	13.54%	13.42%	13.62%	12.62%	13.49%	12.87%	12.92%	12.81%	12.16%	12.17%	12.62%
$G^{(2)}$	15.96%	0.0485	23.12%	22.21%	20.64%	18.83%	17.91%	16.11%	14.04%	13.67%	11.62%	9.46%	7.56%
G_A	18.09%	0.0347	23.21%	22.74%	21.76%	19.68%	19.46%	17.71%	16.68%	16.87%	14.98%	13.34%	12.57%
G_C	16.15%	0.0310	20.50%	20.57%	19.23%	17.49%	17.47%	16.08%	14.79%	15.24%	13.26%	11.78%	11.23%
G_S	**18.64%**	**0.0288**	22.90%	22.44%	21.81%	19.71%	19.98%	18.53%	17.31%	17.57%	15.91%	14.68%	14.22%

Table 4: Purchasing behavior prediction with exposure bias using item embeddings learnt from commodity network. The environment 1 consists of shopping logs mainly with unpopular items and env 2 consists of logs mainly with popular items.

	Mean ACC	STD	Env1:Env2=0:10	1:9	2:8	3:7	4:6	5:5	6:4	7:3	8:2	9:1	10:0
$G^{(1)}$	17.69%	0.0148	15.85%	16.03%	16.44%	16.64%	16.94%	16.67%	18.22%	18.87%	18.87%	19.99%	20.03%
$G^{(2)}$	17.46%	**0.0063**	16.86%	16.97%	16.56%	16.87%	17.17%	18.16%	16.99%	18.07%	18.10%	18.09%	18.27%
G_A	18.51%	0.0132	16.79%	16.94%	17.24%	17.60%	17.94%	18.50%	18.24%	19.78%	19.43%	20.46%	20.70%
G_C	18.56%	0.0127	16.84%	16.97%	17.34%	17.63%	17.81%	18.68%	18.94%	19.50%	19.53%	20.33%	20.62%
G_S	**20.17%**	**0.0092**	19.09%	19.01%	19.14%	19.51%	19.77%	20.02%	20.29%	20.84%	21.02%	21.62%	21.53%

Table 5: Purchasing behavior prediction in different gender groups using item embeddings learnt from commodity network. The environment 1 consists of shopping logs of females and env 2 consists of logs of males.

shift, i.e. the improved stability learnt inside training environments should be adaptive to the agnostic environment to some extent. The results reported in Table3 can well support our claim.

4.3 Real Data Experiment

Furthermore we study the stable graph structure problem in a common real-world application of commodity recommendation.

4.3.1 Experimental setup. For better recommendation, a commodity network is always constructed according to the users' purchase history to preserve the similarities between items. Such a graph is of great business value because the recommendation system could directly promote the items that have the strongest relation to the users' purchase list. However, the shopping logs are full of selection bias, making the commodity network easy to contain spurious correlations. For example, the women are more willing to shop than men and the online system prefers to recommend popular items.

In order to study the problem in depth, we use the data from the public "Cloud Theme Click Dataset", which is an important recommendation procedure in mobile terminal of electronic business. The dataset released in [9] includes more than 4 million purchase histories of users for one month before the promotion started. A shopping behavior can be seen as the set data containing multiple items. Because such decision-making process is susceptible to noise, we use the Swing method to generate the initial graph. The Swing could filter the data noise with cooperation ability of human by the similarity metric $Sim(i, j)$ between the item i and j defined in Eq.(13), where U_i is the user set of purchasing item i and It_u is the item set purchased by user u.

$$Sim(i, j) = \sum_{u \in U_i \cap U_j} \sum_{v \in U_i \cap U_j} \frac{1}{\alpha + |It_u \cap It_v|} \tag{13}$$

Taking the $Sim(i, j)$ as the edge weight of item i and j, we can construct a commodity network in each shopping data environment respectively and then learn a stable version via the SGL framework. Also we use the set prediction task, purchasing behavior prediction here, to evaluate the stability of graph structure. Below we'll present our findings in two special cases: commodity exposure bias and user group bias.

4.3.2 commodity exposure bias. The online system prefers to recommend popular items, causing the unpopular items to get harder to obtain exposure. The exposure bias is very concerned in electronic business, because it can result in the cold-start problem of new commodities. Here we define an item is popular if its occurrence frequency over 100 in dataset, otherwise it is unpopular. If a shopping log consists of popular items mainly, it would be assigned into environment 2, or else into the environment 1. Then we select the total 1585 items which appear at least 4 times in each of the two environments. After dividing the training and testing data in a proportion of 6:4, we use the item embeddings learnt by GCN from different graphs to evaluate the prediction performances in 11 mixing testing datasets as we have done in simulation experiment.

From Table4, the followings are observed: the relations between popular items are easier to be predicted because they are apt to be associated by recommendation; it fails to learn the unpopular items' relationships solely in environment 2 due to the overexposure of popular items, vice versa in environment 1; the stable commodity network G_S can balance the correlations in both environments and reach the highest mean prediction rate more stably.

4.3.3 user group bias. Taking the gender attribute as an example, the women are more willing to shop than men, leading to the correlations in graph inclined to female user group. The detailed experimental procedure is similar to that of commodity exposure bias, except that we divide the purchase logs into environment 1 of female users and environment 2 of the males. It seems that the women tend to purchase all the related goods together at once time, so we can do better prediction for them as in Table5. And the proposed SGL framework can well make up the information loss in single environment to generate the graph G_S with the best

overall performance via learning the essential relationships within commodities.

5 CONCLUSION

In this paper, we target to solve the problem of data selection bias for graph generation, that leads to the bad performance of biased graph structure in Non-I.I.D. scene. To learn the stable graph, we propose a SGL framework, which consists of a GCN module for structure embedding and a designed E-VAE for high-dimensional sparse set generation, to balance the biased underlying relationships between nodes from different environments. Experiments in both simulation data and real-world data have proved the disadvantage of biased graph structure and our algorithm could indeed improve the generalization ability of learnt graph. The SGL framework can be easily varied to adapt to different types of graphs and collected data, e.g. sequential data, which we will exploit in the future work.

6 ACKNOWLEDGEMENTS

This work was supported in part by National Key R&D Program of China No. 2018AAA0102004, National Natural Science Foundation of China (No. U1936219, 61772304, U1611461), Beijing Academy of Artificial Intelligence (BAAI), and a grant from the Institute for Guo Qiang, Tsinghua University; NSF under grants III-1526499, III-1763325, III-1909323, and CNS-1930941. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.
[2] James Atwood and Don Towsley. [n.d.]. Diffusion-Convolutional Neural Networks. ([n. d.]).
[3] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816* (2018).
[4] Peter Bühlmann, Jonas Peters, Jan Ernest, et al. 2014. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics* 42, 6 (2014), 2526–2556.
[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* (2018).
[6] Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *International Conference on Machine Learning*. 942–950.
[7] Nicola De Cao and Thomas Kipf. [n.d.]. MolGAN: An implicit generative model for small molecular graphs. ([n. d.]).
[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.
[9] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2895–2904.
[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[11] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. 2017. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949* (2017).
[12] Kilol Gupta, Mukund Yelahanka Raghuprasad, and Pankhuri Kumar. [n.d.]. A Hybrid Variational Autoencoder for Collaborative Filtering. ([n. d.]).
[13] Hinton and G. E. [n.d.]. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 ([n. d.]), 504–507.
[14] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11313–11320.
[15] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[16] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. (2016).
[17] Kun Kuang, Peng Cui, Susan Athey, Ruoxuan Xiong, and Bo Li. 2018. Stable prediction across unknown environments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1617–1626.
[18] Kun Kuang, Peng Cui, Bo Li, Meng Jiang, and Shiqiang Yang. 2017. Estimating treatment effect in the wild via differentiated confounder balancing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 265–274.
[19] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*. 1378–1387.
[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[21] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *International Conference on Machine Learning*. 4212–4221.
[22] Tengfei Ma, Jie Chen, and Cao Xiao. 2018. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *Advances in Neural Information Processing Systems*. 7113–7124.
[23] Yutaka Matsuo and Mitsuru Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools* 13, 01 (2004), 157–169.
[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.
[25] Zheyan Shen, Peng Cui, Kun Kuang, Bo Li, and Peixuan Chen. 2018. Causally regularized learning with agnostic data selection bias. In *Proceedings of the 26th ACM international conference on Multimedia*. 411–419.
[26] Zheyan Shen, Peng Cui, Tong Zhang, and Kun Kuang. 2019. Stable Learning via Sample Reweighting. arXiv:arXiv:1911.12580
[27] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. 2006. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* 7, Oct (2006), 2003–2030.
[28] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2019. Graph Attention Networks. In *International Conference on Learning Representations*.
[29] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.
[30] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773* (2018).
[31] Kun Zhang and Aapo Hyvarinen. 2012. On the identifiability of the post-nonlinear causal model. *arXiv preprint arXiv:1205.2599* (2012).
[32] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. 2018. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*. 9472–9483.
[33] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. 2018. Deep variational network embedding in wasserstein space. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2827–2836.