Counterfactual Graph Learning for Link Prediction

Anonymous Author(s)
Affiliation
Address
email

Abstract

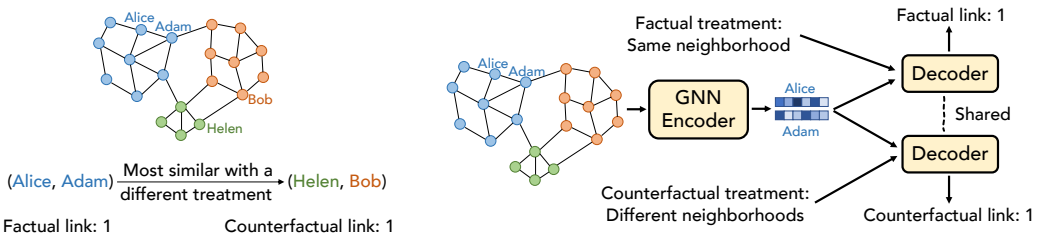
Learning to predict missing links is important for many graph-based applications. Existing methods were designed to learn the observed association between two sets of variables: (1) the observed graph structure and (2) the existence of link between a pair of nodes. However, the causal relationship between these variables was ignored and we visit the possibility of learning it by simply asking a counterfactual question: "*would the link exist or not if the observed graph structure became different?*" To answer this question by causal inference, we consider the information of the node pair as context, global graph structural properties as treatment, and link existence as outcome. In this work, we propose a novel link prediction method that enhances graph learning by the counterfactual inference. It creates counterfactual links from the observed ones, and our method learns representations from both of them. Experiments on a number of benchmark datasets show that our proposed method achieves the state-of-the-art performance on link prediction.

1 Introduction

Link prediction seeks to predict the likelihood of edge existence between node pairs based on the observed graph. Given the omnipresence of graph-structured data, link prediction has copious applications such as movie recommendation (Bennett et al., 2007), chemical interaction prediction (Stanfield et al., 2017), and knowledge graph completion (Kazemi and Poole, 2018). Graph machine learning methods have been widely applied to solve this problem. Their standard scheme is to first learn the representation vectors of nodes and then learn the *association* between the representations of a pair of nodes and the existence of the link between them. For example, graph neural networks (GNNs) use neighborhood aggregation to create the representation vectors: the representation vector of a node is computed by recursively aggregating and transforming representation vectors of its neighboring nodes (Kipf and Welling, 2016a; Hamilton et al., 2017; Wu et al., 2020). Then the vectors are fed into a binary classification model to learn the *association*. GNN methods have shown predominance in the task of link prediction (Kipf and Welling, 2016b; Zhang and Chen, 2018; Zhang et al., 2020a).

Unfortunately, the causal relationship between graph structure and link existence was largely ignored in the previous work. Existing methods that learn from association only were not able to capture essential factors to accurately predict missing links in the *test data*. Take social network as an example. Suppose Alice and Adam live in the same neighborhood and they are close friends. The association between neighborhood belonging and friend closeness could be too strong to discover the essential factors of the friendship such as common interests or family relationship which could be the cause of being living in the same neighborhood. So, our idea is to ask a *counterfactual* question: "*would Alice and Adam still be close friends if they were not living in the same neighborhood?*" If a graph learning model could learn the causal relationship from data by asking the counterfactual questions, it would improve the performance of link prediction with the novel knowledge it captured. Generally, the questions can be described as "*would the link exist or not if the graph structure became different?*"

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Find counterfactual link as the most similar node pair with a different treatment.

(b) Train a GNN-based link predictor to predict factual and counterfactual links given the corresponding treatments.

Figure 1: The proposed CFLP learns the causal relationship between the observed graph structure (e.g., neighborhood similarity, considered as treatment variable) and link existence (considered as outcome). In this example, the link predictor would be trained to estimate the individual treatment effect (ITE) as $1 - 1 = 0$ so it looks for factors other than neighborhood to predict the factual link.

As known to many, counterfactual question is a key component of causal inference and have been well defined in the literature. A counterfactual question is usually framed with three factors: context (as a data point), manipulation (e.g., treatment, intervention, action, strategy), and outcome (van der Laan and Petersen, 2007; Johansson et al., 2016). (To simplify the language, we use "treatment" to refer to the manipulation in this paper, as readers might be familiar more with the word "treatment.") Given certain data context, it asks what the outcome would have been if the treatment had not been the observed value. In the scenario of link prediction, we consider the information of a pair of nodes as context, graph structural properties as treatment, and link existence as outcome. Recall the social network example. The context is Alice and Adam, which includes their personal attributes and relationships with others on the network. The treatment is living in the same neighborhood, which can be given as one attribute or identified by community detection. And the outcome is their friendship.

In this work, we present a **c**ounter**f**actual graph learning method for **l**ink **p**rediction (CFLP) that trains graph learning models to answer the counterfactual questions. Figure 1 illustrates this two-step method. Suppose the treatment variable is defined as one type of global graph structure, e.g., the neighborhood assignment discovered by spectral clustering or community detection algorithms. We are wondering how likely the neighborhood distribution makes a difference on the link (non-)existence for each pair of nodes. So, given a pair of nodes (like Alice and Adam) and the treatment value on this pair (in the same neighborhood), we find a pair of nodes (like Helen and Bob) that satisfies two conditions: (1) it has a different treatment (in different neighborhoods) and (2) it is the most similar pair with the given pair of nodes. We call these matched pair of nodes as "counterfactual links." Note that the outcome of the counterfactual link can be either 1 or 0, depending on whether there exists an edge between the matched pair of nodes. The counterfactual link provides unobserved outcome to the given pair of nodes (Alice and Adam) under a counterfactual condition (in different neighborhoods). After counterfactual links are created for all (positive and negative) training examples, CFLP trains a link predictor (which can be GNN-based) to learn the representation vectors of nodes to predict both the observed factual links and the counterfactual links. In this Alice-Adam example, the link predictor is trained to estimate the individual treatment effect (ITE) of neighborhood assignment as $1 - 1 = 0$. So, the learner will try to discover the essential factors on the friendship between Alice and Adam. For some other examples, if the outcome of counterfactual link is different from that of the given pair of nodes, the learner will estimate the strong effect of the treatment variable. Therefore, CFLP enables graph learning models to predict missing links regarding causal relationship.

Contributions. Our main contributions can be summarized as follows. (1) This is the first work that proposes to improve link prediction by causal inference, specifically, learning to answer counterfactual questions about link existence. (2) This work introduces CFLP that trains GNN-based link predictors to predict both factual and counterfactual links. It learns the causal relationship between global graph structure and link existence. (3) CFLP outperforms competitive baseline methods on several benchmark datasets. On OGB-DDI, our CFLP achieves the state-of-the-art performance. We analyze the impact of counterfactual links as well as the choice of treatment variable. This work sheds insights for improving graph machine learning with causal analysis, which has not been extensively studied yet, when the other direction (machine learning for causal inference) has been studied for a long time.

2 Preliminary

Notations Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph of N nodes, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of observed links. We denote the adjacency matrix as $\mathbf{A} \in \{0,1\}^{N \times N}$, where $A_{i,j} = 1$ indicates nodes v_i and v_j are connected and vice versa. We denote the node feature matrix as $\mathbf{X} \in \mathbb{R}^{N \times F}$, where F is the number of node features and $\mathbf{x}_i$ (bolded) indicates the feature vector of node v_i (the i-th row of $\mathbf{X}$).

Counterfactual Learning Let $\mathcal{X}$ be the set of contexts, $\mathcal{Y}$ be the set of outcome values, and $\mathcal{T}$ be the set of treatments. For a context $x \in \mathcal{X}$ and a treatment $t \in \mathcal{T}$, we denote the outcome of x under the treatment t by $Y_t(x) \in \mathcal{Y}$. Ideally, we would need all possible outcomes of x under all kinds of treatments to study the causal relationships (Morgan and Winship, 2015). However, in reality, only one treatment was applied and thus only one outcome was observed for a given context x. When the variables are specified in data, people use Neyman–Rubin casual model (BCM) to develop statistical learning methods such as propensity score matching (PSM) for causal inference (Rubin, 1974, 2005).

In this work, we look at **link prediction** on graphs. Here we define the variables of counterfactual learning in this scenario. Given a graph G, a context is a pair of nodes $x = (v_i, v_j)$ in the graph; and thus, $\mathcal{X} = \mathcal{V} \times \mathcal{V}$. The outcome variable $Y(x)$ is naturally binary, indicating whether a link exists between the node pair x; and thus, $\mathcal{Y} = \{0, 1\}$. We study the causal effect of binary treatment variable $t \in \mathcal{T} = \{0, 1\}$, where the value of $Y_1(x) - Y_0(x)$ for a particular context x is of high interest and known as the *individualized treatment effect* (ITE) (van der Laan and Petersen, 2007; Weiss et al., 2015). The value of ITE indicates the causality relationship between the treatment and outcome on the context. And the expected ITE given the context distribution is called *averaged treatment effect* (ATE). i.e., ATE $= \mathbb{E}_{x \sim \mathcal{X}}$ ITE(x), for a particular treatment variable.

However, as aforementioned, the fact that we can only observe one potential outcome under one particular treatment prevents the ITE from being known (Johansson et al., 2016). In the problem setting of link prediction, we refer the observed adjacency matrix as the *factual* outcomes $\mathbf{A}$ and the unobserved adjacency matrix when the treatment is different as the *counterfactual* outcomes $\mathbf{A}^{CF}$. We denote $\mathbf{T} \in \{0,1\}^{N \times N}$ as the factual treatment matrix, where $T_{i,j}$ indicates the treatment of the node pair (v_i, v_j). We denote $\mathbf{T}^{CF}$ as the counterfactual treatment matrix where $T^{CF}_{i,j} = 1 - T_{i,j}$. We are interested in (1) estimating the counterfactual outcomes $\mathbf{A}^{CF}$ via observed data, (2) learning with the counterfactual adjacency matrix $\mathbf{A}^{CF}$ to enhance link prediction, and (3) learning the causal relationship between graph structural information (treatment) and link existence (outcome).

3 The Proposed Method

In this section, we introduce CFLP, a novel counterfactual graph learning method for link prediction. In Section 3.1, we define treatment variable and counterfactual outcomes/links on graph data and present how to compute them (Figure 1(a)). In Section 3.2, we introduce the graph learning model that learns from both the observed graph and the created counterfactual links (Figure 1(b)).

3.1 Defining Treatment Variable and Counterfactual Links

Treatment Previous work on graph machine learning (Velickovic et al., 2019; Park et al., 2020) showed that the graph's global structural information could improve the quality of representation vectors of nodes learned by GNNs. This is because the message passing-based GNNs aggregate local information in the algorithm of representation vector generation and the global structural information is complementary with the aggregated information. Therefore, for a pair of nodes, one option of defining the treatment variable is its global structural role in the graph. Without the loss of generality, we use Louvain (Blondel et al., 2008), an unsupervised approach that has been widely used for community detection, as an example. Louvain discovers community structure of a graph and assigns each node to one community. Then we can define the binary treatment variable as whether these two nodes in the pair belong to the same community. Let $c : \mathcal{V} \to \mathbb{N}$ be any graph mining/clustering method that outputs the index of community/cluster/neighborhood that each node belongs to. The treatment matrix $\mathbf{T}$ is defined as

$$T_{i,j} = \begin{cases} 1 & \text{, if } c(v_i) = c(v_j); \\ 0 & \text{, otherwise.} \end{cases} \tag{1}$$

For the choice of c, we suggest methods that group nodes based on global graph structural information, including but not limited to Louvain (Blondel et al., 2008), K-core (Bader and Hogue, 2003), and spectral clustering (Ng et al., 2001).

Counterfactual Links As mentioned in Section 2, for each node pair (context), the observed data contains only the factual treatment and outcome, meaning that the link existence for the given node pair with an opposite treatment is unknown. Therefore, we use the outcome from the nearest observed context as a substitute. This idea has been adopted by many methods (Johansson et al., 2016; Alaa and Van Der Schaar, 2019). That is, we want to find the nearest neighbor with the opposite treatment for each observed node pairs and use the nearest neighbor's outcome as a *counterfactual link*. Formally, $\forall (v_i, v_j) \in \mathcal{V} \times \mathcal{V}$, we want to find its counterfactual link (v_a, v_b) as below:

$$(v_a, v_b) = \arg\min_{v_a, v_b \in \mathcal{V}} \{d((v_i, v_j), (v_a, v_b)) \mid T_{a,b} = 1 - T_{i,j}\}, \tag{2}$$

where $d(\cdot, \cdot)$ is a metric of measuring the distance between a pair of node pairs (a pair of contexts). Considering that we want to find the nearest node pair based on not only the raw node features but also structural features, here we take the state-of-the-art unsupervised graph representation learning method MVGRL (Hassani and Khasahmadi, 2020) to learn the node embeddings $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times \tilde{F}}$ from the observed graph. We use $\tilde{\mathbf{X}}$ to find the nearest neighbors of node pairs. Nevertheless, finding the nearest neighbors by computing the distance between all pairs of node pairs is extremely inefficient, which takes $O(N^4)$ comparisons (as there are totally $O(N^2)$ node pairs). Hence we approximate Eq. (2) by substituting the distance between node pairs by the distance between nodes. That is, $\forall (v_i, v_j) \in \mathcal{V} \times \mathcal{V}$, we want to find its counterfactual link (v_a, v_b) as below:

$$(v_a, v_b) = \arg\min_{v_a, v_b \in \mathcal{V}} \{d(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_a) + d(\tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_b) \mid T_{a,b} = 1 - T_{i,j}, d(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_a) + d(\tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_b) < 2\gamma\}, \tag{3}$$

where $d(\cdot, \cdot)$ is specified as the Euclidean distance on the embedding space of $\tilde{\mathbf{X}}$, and γ is a hyperparameter that defines the maximum distance that two nodes are considered as similar. Note that when no node pair satisfies the above equation, we do not assign any nearest neighbor for a given node pair to ensure all the neighbors are similar enough (as substitutes) in the feature space. Therefore, the counterfactual treatment matrix $\mathbf{T}^{CF}$ and the counterfactual adjacency matrix $\mathbf{A}^{CF}$ are defined as

$$T^{CF}_{i,j}, A^{CF}_{i,j} = \begin{cases} 1 - T_{i,j}, A_{a,b} & \text{, if } \exists\ (v_a, v_b) \in \mathcal{V} \times \mathcal{V} \text{ satisfies Eq. (3);} \\ T_{i,j}, A_{i,j} & \text{, otherwise.} \end{cases} \tag{4}$$

It is worth noting that the node embeddings $\tilde{\mathbf{X}}$ and the nearest neighbors are computed only once and do not change during the learning process. $\tilde{\mathbf{X}}$ is only used for finding the nearest neighbors.

Learning from Counterfactual Distributions Let P^F be the factual distribution of the observed contexts and treatments, and P^{CF} be the counterfactual distribution that is composed of the observed contexts and opposite treatments. We define the empirical factual distribution $\hat{P}^F \sim P^F$ as $\hat{P}^F = \{(v_i, v_j, T^F_{i,j})\}^N_{i,j=1}$, and define the empirical counterfactual distribution $\hat{P}^{CF} \sim P^{CF}$ as $\hat{P}^{CF} = \{(v_i, v_j, T^{CF}_{i,j})\}^N_{i,j=1}$. Unlike traditional link prediction methods that take only $\hat{P}^F$ as input and use the observed outcomes $\mathbf{A}$ as the training target, the idea of counterfactual graph learning is to take advantage of the counterfactual distribution by having $\hat{P}^{CF}$ as a complementary input and use the counterfactual outcomes $\mathbf{A}^{CF}$ as the training target for the counterfactual data samples.

3.2 The Counterfactual Graph Learning Model

In this subsection, we present the design of our model as well as the training method. The input of the model in CFLP includes (1) the observed graph data $\mathbf{A}$ and raw feature matrix $\mathbf{X}$, (2) the factual treatments $\mathbf{T}^F$ and counterfactual treatments $\mathbf{T}^{CF}$, and (3) the counterfactual graph data $\mathbf{A}^{CF}$. The output contains link prediction logits in $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{A}}^{CF}$ for the factual and counterfactual adjacency matrices $\mathbf{A}$ and $\mathbf{A}^{CF}$, respectively.

Graph Learning Model The model consist of two trainable components: a graph encoder f and a link decoder g. The graph encoder generates representation vectors of nodes from graph data G. And the link decoder projects the representation vectors of node pairs into the link prediction logits. The

choice of the graph encoder f can be any end-to-end GNN model. Without the loss of generality, here we use the commonly used graph convolutional network (GCN) (Kipf and Welling, 2016a). Each layer of GCN is defined as

$$\mathbf{H}^{(l)} = f^{(l)}(\mathbf{A}, \mathbf{H}^{(l-1)}; \mathbf{W}^{(l)}) = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}), \tag{5}$$

where l is the layer index, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops, $\tilde{\mathbf{D}}$ is the diagonal degree matrix $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $\mathbf{H}^{(0)} = \mathbf{X}$, $\mathbf{W}^{(l)}$ is the learnable weight matrix at the l-th layer, and $\sigma(\cdot)$ denotes a nonlinear activation such as ReLU. We denote $\mathbf{Z} = f(\mathbf{A}, \mathbf{X}) \in \mathbb{R}^{N \times H}$ as the output from the encoder's last layer, i.e., the H-dimensional representation vectors of nodes. Following previous work (Zhang et al., 2020a), we compute the representation of a node pair as the Hadamard product of the vectors of the two nodes. That is, the representation for the node pair (v_i, v_j) is $\mathbf{z}_i \odot \mathbf{z}_j \in \mathbb{R}^H$, where $\odot$ stands for the Hadamard product.

For the link decoder that predicts whether a link exists between a pair of nodes, we opt for simplicity and adopt a simple decoder based on multi-layer perceptron (MLP), given the representations of node pairs and their treatments. That is, the decoder g is defined as

$$\widehat{\mathbf{A}} = g(\mathbf{Z}, \mathbf{T}), \text{ where } \widehat{A}_{i,j} = \text{MLP}([\mathbf{z}_i \odot \mathbf{z}_j, T_{i,j}]), \tag{6}$$

$$\widehat{\mathbf{A}}^{CF} = g(\mathbf{Z}, \mathbf{T}^{CF}), \text{ where } \widehat{A}^{CF}_{i,j} = \text{MLP}([\mathbf{z}_i \odot \mathbf{z}_j, T^{CF}_{i,j}]), \tag{7}$$

where $[\cdot, \cdot]$ stands for the concatenation of vectors.

During the training process, data samples from the empirical factual distribution $\hat{P}^F$ and the empirical counterfactual distribution $\hat{P}^{CF}$ are fed into decoder g and optimized towards $\mathbf{A}$ and $\mathbf{A}^{CF}$, respectively. That is, for the two distributions, the loss functions are as follows:

$$\mathcal{L}_F = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j} \cdot \log \widehat{A}_{i,j} + (1 - A_{i,j}) \cdot \log(1 - \widehat{A}_{i,j}), \tag{8}$$

$$\mathcal{L}_{CF} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} A^{CF}_{i,j} \cdot \log \widehat{A}^{CF}_{i,j} + (1 - A^{CF}_{i,j}) \cdot \log(1 - \widehat{A}^{CF}_{i,j}). \tag{9}$$

Balancing Counterfactual Learning In the training process, the above loss minimizations train the model on both the empirical factual distribution $\hat{P}^F \sim P^F$ and empirical counterfactual distribution $\hat{P}^{CF} \sim P^{CF}$ that are not necessarily equal – the training examples (node pairs) do not have to be aligned. However, at the stage of inference, the test data contains only observed (factual) samples. Such a gap between the training and test data distributions exposes the model in the risk of covariant shift, which is a common issue in counterfactual learning (Johansson et al., 2016; Assaad et al., 2021).

To force the distributions of representations of factual distributions and counterfactual distributions to be similar, we use the discrepancy distance (Mansour et al., 2009; Johansson et al., 2016) as another objective to regularize the representation learning. That is, we use the following loss term to minimize the distance between the learned representations from $\hat{P}^F$ and $\hat{P}^{CF}$:

$$\mathcal{L}_{disc} = \text{disc}(\hat{P}^F_f, \hat{P}^{CF}_f), \text{ where disc}(P, Q) = ||P - Q||_F, \tag{10}$$

where $||\cdot||_F$ denotes the Frobenius Norm, and $\hat{P}^F_f$ and $\hat{P}^{CF}_f$ denote the node pair representations learned by graph encoder f from factual distribution and counterfactual distribution, respectively.

Training During the training of CFLP, we want the model to be optimized towards three targets: (1) accurate link prediction on the observed outcomes (Eq. (8)), (2) accurate estimation on the counterfactual outcomes (Eq. (9)), and (3) regularization on the representation spaces learned from $\hat{P}^F$ and $\hat{P}^{CF}$ (Eq. (10)). Therefore, the overall training loss of our proposed CFLP is

$$\mathcal{L} = \mathcal{L}_F + \alpha \cdot \mathcal{L}_{CF} + \beta \cdot \mathcal{L}_{disc}, \tag{11}$$

where α and β are hyperparameters to control the weights of counterfactual link prediction (outcome estimation) loss and discrepancy loss.

Table 1: Statistics of datasets used in the experiments.

Dataset	CORA	CITESEER	PUBMED	FACEBOOK	OGB-DDI
# nodes	2,708	3,327	19,717	4,039	4,267
# links	5,278	4,552	44,324	88,234	1,334,889
# validation node pairs	1,054	910	8,864	17,646	235,371
# test node pairs	2,110	1,820	17,728	35,292	229,088

Summary Algorithm 1 summarizes the whole process of CFLP. The **first step** is to compute the factual and counterfactual treatments $\mathbf{T}$, $\mathbf{T}^{CF}$ as well as the counterfactual outcomes $\mathbf{A}^{CF}$. Then, the **second step** trains the graph learning model on both the observed factual data and created counterfactual data with the integrated loss function (Eq. (11)). Note that the discrepancy loss (Eq. (10)) is computed on the representations of node pairs learned by the graph encoder f, so the decoder g is trained with data from both $\hat{P}^F$ and $\hat{P}^{CF}$ without balancing the constraints. Therefore, after the model is sufficiently trained, we freeze the graph encoder f and fine-tune g with only the factual data. Finally, after the decoder is sufficiently fine-tuned, we output the link prediction logits for both the factual and counterfactual adjacency matrices.

Algorithm 1: CFLP: Counterfactual graph learning for link prediction

Input : f, g, $\mathbf{A}$, $\mathbf{X}$, n_epochs, n_epoch_ft

Compute $\mathbf{T}$ by Eq. (1) ;
Compute $\mathbf{T}^{CF}$, $\mathbf{A}^{CF}$ by Eqs. (3) and (4) ;
`/* model training */`
Initialize Θ_f in f and Θ_g in g ;
for *epoch in range(n_epochs)* **do**
  $\mathbf{Z} = f(\mathbf{A}, \mathbf{X})$;
  Get $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{A}}^{CF}$ via g with Eqs. (6) and (7) ;
  Update Θ_f and Θ_g with $\mathcal{L}$; `// (11)`
end
`/* decoder fine-tuning */`
Freeze Θ_f and re-initialize Θ_g ;
$\mathbf{Z} = f(\mathbf{A}, \mathbf{X})$;
for *epoch in range(n_epochs_ft)* **do**
  Get $\widehat{\mathbf{A}}$ via g with Eq. (6) ;
  Update Θ_g with $\mathcal{L}_F$; `// Eq. (8)`
end
`/* model inferencing */`
$\mathbf{Z} = f(\mathbf{A}, \mathbf{X})$;
Get $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{A}}^{CF}$ via g with Eqs. (6) and (7) ;

Output : $\widehat{\mathbf{A}}$ for link prediction, $\widehat{\mathbf{A}}^{CF}$

Complexity The complexity of the first step (finding counterfactual links with nearest neighbors) is proportional to the number of node pairs. When γ is set as a small value to obtain indeed similar node pairs, this step (Eq. (3)) uses constant time. Moreover, the computation in Eq. (3) can be parallelized. Therefore, the time complexity is $O(N^2/C)$ where C is the number of processes. For the complexity of the second step (training counterfactual learning model), the GNN encoder has time complexity of $O(LH^2N + LH|\mathcal{E}|)$ (Wu et al., 2020), where L is the number of GNN layers and H is the size of node representations. Given that we sample the same number of non-existing links as that of observed links during training, the complexity of a *three-layer MLP* decoder is $O(((H+1)\cdot d_h + d_h \cdot 1)|\mathcal{E}|) = O(d_h(H+2)|\mathcal{E}|)$, where d_h is the number of neurons in the hidden layer. Therefore, the second step has linear time complexity w.r.t. the sum of node and edge counts.

Limitations First, as mentioned above, the computation of finding counterfactual links has a worst-case complexity of $O(N^2)$. Second, CFLP performs counterfactual prediction with only a single treatment; however, there are quite a few kinds of graph structural information that can be considered as treatments. Future work can leverage the rich structural information by bundled treatments (Zou et al., 2020) in counterfactual graph learning.

4 Experiments

4.1 Experimental Setup

We conduct experiments on five benchmark datasets including citation networks (CORA, CITESEER, PUBMED (Yang et al., 2016)), social network (FACEBOOK (McAuley and Leskovec, 2012)), and drug-drug interaction network (OGB-DDI (Wishart et al., 2018)) from the Open Graph Benchmark

Table 2: Link prediction performances measured by Hits@20. Best performance and best baseline performance are marked with bold and underline, respectively.

	CORA	CITESEER	PUBMED	FACEBOOK	OGB-DDI
Node2Vec	49.96±2.51	47.78±1.72	39.19±1.02	24.24±3.02	23.26±2.09
MVGRL	19.53±2.64	14.07±0.79	14.19±0.85	14.43±0.33	10.02±1.01
VGAE	45.91±3.38	44.04±4.86	23.73±1.61	37.01±0.63	11.71±1.96
SEAL	51.35±2.26	40.90±3.68	28.45±3.81	40.89±5.70	30.56±3.86
LGLP	<u>62.98</u>±0.56	<u>57.43</u>±3.71	–	37.86±2.13	–
GCN	49.06±1.72	55.56±1.32	21.84±3.87	<u>53.89</u>±2.14	37.07±5.07
GSAGE	53.54±2.96	53.67±2.94	<u>39.13</u>±4.41	45.51±3.22	53.90±4.74
JKNet	48.21±3.86	55.60±2.17	25.64±4.11	52.25±1.48	<u>60.56</u>±8.69
Our proposed CFLP with different graph encoders					
CFLP w/ GCN	60.34±2.33	59.45±2.30	34.12±2.72	53.95±2.29	52.51±1.09
CFLP w/ GSAGE	57.33±1.73	53.05±2.07	43.07±2.36	47.28±3.00	75.49±4.33
CFLP w/ JKNet	**65.57**±1.05	**68.09**±1.49	**44.90**±2.00	**55.22**±1.29	**86.08**±1.98

(OGB) (Hu et al., 2020). For the first four datasets, we randomly select 10%/20% of the links and the same numbers of disconnected node pairs as validation/test samples. The links in the validation and test sets are masked off from the training graph. For OGB-DDI, we used the OGB official train/validation/test splits. Statistics for the datasets are given in Table 1 and details are in Appendix. We use K-core (Bader and Hogue, 2003) clusters as the default treatment variable. We evaluate CFLP on three commonly used GNN encoders: GCN (Kipf and Welling, 2016a), GSAGE (Hamilton et al., 2017), and JKNet (Xu et al., 2018). We compare the link prediction performance of CFLP against Node2Vec (Grover and Leskovec, 2016), MVGRL (Hassani and Khasahmadi, 2020), VGAE (Kipf and Welling, 2016b), SEAL (Zhang and Chen, 2018), LGLP (Cai et al., 2021), and GNNs with MLP decoder. We report averaged test performance and their standard deviation over 20 runs with different random parameter initializations. Other than the most commonly used of Area Under ROC Curve (AUC), we report Hits@20 (one of the primary metrics on OGB leaderboard) as a more challenging metric, as it expects models to rank positive edges higher than nearly all negative edges.

Besides performance comparison on link prediction, we will answer two questions to suggest a way of choosing a treatment variable for creating counterfactual links: (Q1) Does CFLP sufficiently learn the observed *averaged treatment effect* (ATE) derived from the counterfactual links? (Q2) What is the relationship between the estimated ATE learned in the method and the prediction performance? If the answer to Q1 is yes, then the answer to Q2 will indicate how to choose treatment based on observed ATE. To answer the Q1, we calculate the observed ATE ($\widehat{\text{ATE}}_{obs}$) by comparing the observed links in $\mathbf{A}$ and created counterfactual links $\mathbf{A}^{CF}$ that have opposite treatments. And we calculate the estimated ATE ($\widehat{\text{ATE}}_{est}$) by comparing the predicted links in $\widehat{\mathbf{A}}$ and predicted counterfactual links $\widehat{\mathbf{A}}^{CF}$. Formally, $\widehat{\text{ATE}}_{obs}$ and $\widehat{\text{ATE}}_{est}$ are defined as

$$\widehat{\text{ATE}}_{obs} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\{\mathbf{T}\odot(\mathbf{A}-\mathbf{A}^{CF})+(\mathbf{1}_{N\times N}-\mathbf{T})\odot(\mathbf{A}^{CF}-\mathbf{A})\}_{i,j}. \tag{12}$$

$$\widehat{\text{ATE}}_{est} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\{\mathbf{T}\odot(\widehat{\mathbf{A}}-\widehat{\mathbf{A}}^{CF})+(\mathbf{1}_{N\times N}-\mathbf{T})\odot(\widehat{\mathbf{A}}^{CF}-\widehat{\mathbf{A}})\}_{i,j}. \tag{13}$$

The treatment variables we will investigate are usually graph clustering or community detection methods, such as K-core (Bader and Hogue, 2003), stochastic block model (SBM) (Karrer and Newman, 2011), spectral clustering (SpecC) (Ng et al., 2001), propagation clustering (PropC) (Raghavan et al., 2007), Louvain (Blondel et al., 2008), common neighbors (CommN), Katz index, and hierarchical clustering (Ward) (Ward Jr, 1963). We use JKNet (Xu et al., 2018) as the default graph encoder.

Implementation details and supplementary experimental results (e.g., sensitivity on γ, ablation study on $\mathcal{L}_{CF}$ and $\mathcal{L}_{disc}$) can be found in Appendix. Source code is available in supplementary material.

4.2 Experimental Results

Link Prediction Tables 2 and 3 show the link prediction performance of Hits@20 and AUC by all methods. LGLP on PUBMED and OGB-DDI are missing due to the out of memory error when

Table 3: Link prediction performances measured by AUC. Best performance and best baseline performance are marked with bold and underline, respectively.

	CORA	CITESEER	PUBMED	FACEBOOK	OGB-DDI
Node2Vec	84.49±0.49	80.00±0.68	80.32±0.29	86.49±4.32	90.83±0.02
MVGRL	75.07±3.63	61.20±0.55	80.78±1.28	79.83±0.30	81.45±0.99
VGAE	88.68±0.40	85.35±0.60	95.80±0.13	98.66±0.04	93.08±0.15
SEAL	92.55±0.50	85.82±0.44	96.36±0.28	**99.60**±0.02	97.85±0.17
LGLP	91.30±0.05	89.41±0.13	–	98.51±0.01	–
GCN	90.25±0.53	71.47±1.40	96.33±0.80	99.43±0.02	99.82±0.05
GSAGE	90.24±0.34	87.38±1.39	96.78±0.11	99.29±0.04	99.93±0.02
JKNet	89.05±0.67	88.58±1.78	96.58±0.23	99.43±0.02	**99.94**±0.01
Our proposed CFLP with different graph encoders					
CFLP w/ GCN	92.55±0.50	89.65±0.20	96.99±0.08	99.38±0.01	99.44±0.05
CFLP w/ GSAGE	92.61±0.52	91.84±0.20	97.01±0.01	99.34±0.10	99.83±0.05
CFLP w/ JKNet	**93.05**±0.24	**92.12**±0.47	**97.53**±0.17	99.31±0.04	**99.94**±0.01

running the code package from the authors. We observe that our CFLP on different graph encoders achieve similar or better performances compared with baselines. The only exception is the AUC on FACEBOOK where most methods have close-to-perfect AUC. As AUC is a relatively easier metric comparing with Hits@20, most methods achieved good performance on AUC. We observe that CFLP with JKNet almost consistently achieves the best performance and outperforms baselines significantly on Hits@20. Specifically, compared with the best baseline, CFLP improves relatively by 16.4% and 0.8% on Hits@20 and AUC, respectively. It is worth noting that CFLP with JKNet achieves the state-of-the-art performance on the official leaderboard[1] of OGB-DDI.

Figure 2 shows the AUC performance of CFLP on CORA with different combinations of α and β. We observe that the performance is the poorest when $\alpha = \beta = 0$ and gradually improves and gets stable as α and β increase, showing that CFLP is robust to the hyperparameters α and β.

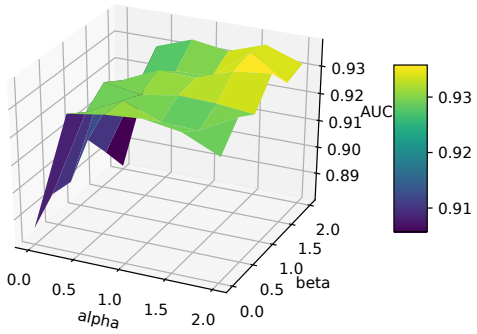


Figure 2: AUC performance of CFLP on CORA w.r.t different combanitions of α and β.

ATE with Different Treatments Tables 4 and 5 show the link prediction performance, $\widehat{\text{ATE}}_{obs}$, and $\widehat{\text{ATE}}_{est}$ of CFLP (with JKNet) when using different treatments. The treatments in Tables 4 and 5 are sorted by the Hits@20 performance. Bigger ATE indicates stronger causal relationship between the treatment and outcome, and vice versa. We observe: (1) $\widehat{\text{ATE}}_{est}$ values are generally close to $\widehat{\text{ATE}}_{obs}$, showing that CFLP was sufficiently trained to learn the causal relationship between graph structure information and link existence; (2) $\widehat{\text{ATE}}_{obs}$ and $\widehat{\text{ATE}}_{est}$ are both negatively correlated with the link prediction performance, showing that we can pick a proper treatment prior to training a model with CFLP. Using the treatment that has the weakest causal relationship with link existence is likely to train the model to capture more essential factors on the outcome, in a way similar to denoising the unrelated information from the representations.

5 Related Work

Link Prediction With its wide applications, link prediction has draw attention from many research communities including statistical machine learning and data mining. Stochastic generative methods based on stochastic block models (SBM) are developed to generate links (Mehta et al., 2019). In data mining, matrix factorization (Menon and Elkan, 2011), heuristic methods (Philip et al., 2010; Martínez et al., 2016), and graph embedding methods (Cui et al., 2018) have been applied to predict links in the graph. Heuristic methods compute the similarity score of nodes based on their neighborhoods. These

[1] https://ogb.stanford.edu/docs/leader_linkprop/#ogbl-ddi

Table 4: Results of CFLP with different treatments on CORA. (sorted by Hits@20)

	Hits@20	$\widehat{\text{ATE}}_{obs}$	$\widehat{\text{ATE}}_{est}$
K-core	65.6±1.1	0.002	0.013±0.003
SBM	64.2±1.1	0.006	0.023±0.015
CommN	62.3±1.6	0.007	0.053±0.021
PropC	61.7±1.4	0.037	0.059±0.065
Ward	61.2±2.3	0.001	0.033±0.012
SpecC	59.3±2.8	0.002	0.033±0.011
Louvain	57.6±1.8	0.025	0.138±0.091
Katz	56.6±3.4	0.740	0.802±0.041

Table 5: Results of CFLP with different treatments on CITESEER. (sorted by Hits@20)

	Hits@20	$\widehat{\text{ATE}}_{obs}$	$\widehat{\text{ATE}}_{est}$
SBM	71.6 ±1.9	0.004	0.005 ±0.001
K-core	68.1±1.5	0.002	0.010±0.002
Ward	67.0±1.7	0.003	0.037±0.009
PropC	64.6±3.6	0.141	0.232±0.113
Louvain	63.3±2.5	0.126	0.151±0.078
SpecC	59.9±1.3	0.009	0.166±0.034
Katz	57.3±0.5	0.245	0.224±0.037
CommN	56.8±4.9	0.678	0.195±0.034

methods can be generally categorized into first-order, second-order, and high-order heuristics based on the maximum distance of the neighbors. Graph embedding methods learn latent node features via embedding lookup and use them for link prediction (Perozzi et al., 2014; Tang et al., 2015; Grover and Leskovec, 2016; Wang et al., 2016).

In the past few years, GNNs have showed promising results on various graph-based tasks with their ability of learning from features and custom aggregations on structures, (Kipf and Welling, 2016a; Hamilton et al., 2017; Xu et al., 2018; Wu et al., 2020). With node pair representations and an attached MLP or inner-product decoder, GNNs can be used for link prediction (Zhang et al., 2020a). For example, VGAE used GCN to learn node representations and reconstruct the graph structure (Kipf and Welling, 2016b). SEAL extracted a local subgraph around each target node pair and then learned graph representation from local subgraph for link prediction (Zhang and Chen, 2018). Following the scheme of SEAL, Cai and Ji (2020) proposed to improve local subgraph representation learning by multi-scale graph representation learning. And LGLP inverted the local subgraphs to line graphs before learning representations (Cai et al., 2021). However, very limited work has studied to use causal inference for improving link prediction.

Counterfactual Prediction As a mean of learning the causality between treatment and outcome, counterfactual prediction has been used for a variety of applicaitons such as recommender systems (Wang et al., 2020; Xu et al., 2020), health care (Alaa and van der Schaar, 2017), vision-language tasks (Zhang et al., 2020b; Parvaneh et al., 2020), and decision making (Coston et al., 2020; Pitis et al., 2020; Kusner et al., 2017). To infer the causal relationships, previous work usually estimated the ITE via function fitting models (Gelman and Hill, 2006; Chipman et al., 2010; Wager and Athey, 2018; Assaad et al., 2021) which estimated the transductive ITE. Peysakhovich et al. (2019) and Zou et al. (2020) studied counterfactual prediction with multiple agents and bundled treatments, respectively. Pawlowski et al. (2020) proposed a deep structural causal model for tractable counterfactual inference.

Causal Inference Causal inference methods usually re-weighted samples based on propensity score (Rosenbaum and Rubin, 1983; Austin, 2011; Kuang et al., 2017a,b) to remove confounding bias from binary treatments. Recently, several works studied about learning treatment invariant representation to predict the counterfactual outcomes (Hassanpour and Greiner, 2019b,a; Shalit et al., 2017; Yao et al., 2018; Bica et al., 2020; Hassanpour and Greiner, 2019a; Li and Fu, 2017). When part of unobserved outcomes may mislead the counterfactual prediction, Louizos et al. (2017) attempted to infer the outcomes from proxies, and Hartford et al. (2017) introduced instrumental variable. SITE preserved local similarity to balance the distributions of control and treated groups (Yao et al., 2018). Yoon et al. (2018) estimated ITE with generative adversarial networks (GANs). Assaad et al. (2021) discussed the trade-off between achieving balance and predictive power.

6 Conclusion

In this work, we presented a counterfactual graph learning method for link prediction (CFLP). We introduced the idea of counterfactual prediction to improve link prediction on graphs. CFLP accurately predicted the missing links by exploring the causal relationship between global graph structure and link existence. Extensive experiments demonstrated that CFLP achieved the state-of-the-art performance on benchmark datasets.

References

Ahmed Alaa and Mihaela Van Der Schaar. Validating causal inference models via influence functions. In *International Conference on Machine Learning*, pages 191–201. PMLR, 2019.

Ahmed M Alaa and Mihaela van der Schaar. Bayesian inference of individualized treatment effects using multi-task gaussian processes. *Advances in Neural Information Processing Systems*, 2017.

Serge Assaad, Shuxi Zeng, Chenyang Tao, Shounak Datta, Nikhil Mehta, Ricardo Henao, Fan Li, and Lawrence Carin Duke. Counterfactual representation learning with balancing weights. In *International Conference on Artificial Intelligence and Statistics*, pages 1972–1980. PMLR, 2021.

Peter C Austin. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research*, 46(3):399–424, 2011.

Gary D Bader and Christopher WV Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC bioinformatics*, 4(1):1–27, 2003.

James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer, 2007.

Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. *arXiv preprint arXiv:2002.04083*, 2020.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10):P10008, 2008.

Lei Cai and Shuiwang Ji. A multi-scale approach for graph link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3308–3315, 2020.

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Amanda Coston, Edward H Kennedy, and Alexandra Chouldechova. Counterfactual predictions under runtime confounding. *Advances in Neural Information Processing Systems*, 2020.

Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2018.

Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

Jason Hartford, Greg Lewis, Kevin Leyton-Brown, and Matt Taddy. Deep iv: A flexible approach for counterfactual prediction. In *International Conference on Machine Learning*, pages 1414–1423. PMLR, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

Negar Hassanpour and Russell Greiner. Counterfactual regression with importance sampling weights. In *IJCAI*, pages 5880–5887, 2019a.

Negar Hassanpour and Russell Greiner. Learning disentangled representations for counterfactual regression. In *International Conference on Learning Representations*, 2019b.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pages 3020–3029. PMLR, 2016.

Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Kun Kuang, Peng Cui, Bo Li, Meng Jiang, and Shiqiang Yang. 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*, pages 265–274, 2017a.

Kun Kuang, Peng Cui, Bo Li, Meng Jiang, Shiqiang Yang, and Fei Wang. Treatment effect estimation with data-driven variable decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017b.

Matt J Kusner, Joshua R Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *Advances in Neural Information Processing Systems*, 2017.

Sheng Li and Yun Fu. Matching on balanced nonlinear representations for treatment effects estimation. In *Advances in Neural Information Processing Systems*, 2017.

Christos Louizos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. *arXiv preprint arXiv:1705.08821*, 2017.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016.

Julian J McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems*, volume 2012, pages 548–56, 2012.

Nikhil Mehta, Lawrence Carin Duke, and Piyush Rai. Stochastic blockmodels meet graph neural networks. In *International Conference on Machine Learning*, pages 4466–4474. PMLR, 2019.

Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

Stephen L Morgan and Christopher Winship. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856, 2001.

Chanyoung Park, Jiawei Han, and Hwanjo Yu. Deep multiplex graph infomax: Attentive multiplex network embedding using global information. *Knowledge-Based Systems*, 197:105861, 2020.

Amin Parvaneh, Ehsan Abbasnejad, Damien Teney, Qinfeng Shi, and Anton van den Hengel. Counterfactual vision-and-language navigation: Unravelling the unseen. *Advances in Neural Information Processing Systems*, 33, 2020.

Nick Pawlowski, Daniel C Castro, and Ben Glocker. Deep structural causal models for tractable counterfactual inference. *Advances in Neural Information Processing Systems*, 2020.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

Alexander Peysakhovich, Christian Kroer, and Adam Lerer. Robust multi-agent counterfactual prediction. *Advances in Neural Information Processing Systems*, 2019.

S Yu Philip, Jiawei Han, and Christos Faloutsos. *Link mining: Models, algorithms, and applications*. Springer, 2010.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. *Advances in Neural Information Processing Systems*, 2020.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pages 3076–3085. PMLR, 2017.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

Zachary Stanfield, Mustafa Coşkun, and Mehmet Koyutürk. Drug response prediction as a link prediction problem. *Scientific reports*, 7(1):1–13, 2017.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

Mark J van der Laan and Maya L Petersen. Causal effect models for realistic individualized treatment and intention to treat rules. *The international journal of biostatistics*, 3(1), 2007.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR (Poster)*, 2019.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

Zifeng Wang, Xi Chen, Rui Wen, Shao-Lun Huang, Ercan E Kuruoglu, and Yefeng Zheng. Information theoretic counterfactual learning from missing-not-at-random feedback. *Advances in Neural Information Processing Systems*, 2020.

Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

Jeremy Weiss, Finn Kuusisto, Kendrick Boyd, Jie Liu, and David Page. Machine learning for treatment assignment: Improving individualized risk attribution. In *AMIA Annual Symposium Proceedings*, volume 2015, page 1306. American Medical Informatics Association, 2015.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Adversarial counterfactual learning and evaluation for recommender system. *Advances in Neural Information Processing Systems*, 2020.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

Liuyi Yao, Sheng Li, Yaliang Li, Mengdi Huai, Jing Gao, and Aidong Zhang. Representation learning for treatment effect estimation from observational data. *Advances in Neural Information Processing Systems*, 31, 2018.

Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. Ganite: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*, 2018.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Revisiting graph neural networks for link prediction. *arXiv preprint arXiv:2010.16103*, 2020a.

Zhu Zhang, Zhou Zhao, Zhijie Lin, Xiuqiang He, et al. Counterfactual contrastive learning for weakly-supervised vision-language grounding. *Advances in Neural Information Processing Systems*, 33: 18123–18134, 2020b.

Hao Zou, Peng Cui, Bo Li, Zheyan Shen, Jianxin Ma, Hongxia Yang, and Yue He. Counterfactual prediction for bundle treatment. *Advances in Neural Information Processing Systems*, 33, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] at the end of Section 3.2
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] in Section 4.1 and Appendix

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] in Appendix

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]