Distance-Enhanced Graph Neural Network for Link Prediction

Boning Li[1] Yingce Xia[2] Shufang Xie[2] Lijun Wu[2] Tao Qin[2]

Abstract

Link prediction, which is to predict the existence of a link/edge between two vertices in a graph, is a classical problem in machine learning. Intuitively, if it takes a long distance to walk from u to v along the existing edges, there should not be a link between them, and vice versa. This motivates us to explicitly combine the distance information with graph neural networks (GNNs) to improve link prediction. Calculating the distances between any two vertices (e.g., shortest path, expectation of random walk) in training is time consuming. To overcome this difficulty, we propose an anchor-based distance: First, we randomly select K anchor vertices from the graph and then calculate the shortest distances of all vertices in the graph to them. The distance between vertices u and v is estimated as the average of their distances to the K anchor vertices. After that, we feed the distance into the GNN module. Our method brings significant improvement for link prediction with few additional parameters. We achieved state-of-the-art result on the drug-drug-interaction (i.e., DDI) and protein-protein-association (i.e., PPA) tasks of OGB (Hu et al., 2020). Our code is available at `https://github.com/lbn187/DLGNN`.

1. Introduction

Many biological and medical tasks can be formulated as the problems on graphs (Berg et al., 2017; Nguyen et al., 2019; Gilmer et al., 2017; Kearnes et al., 2016). A graph is made up of vertices and edges, which represent the entities and the relations among entities respectively. Link prediction is an important topic in graph learning, which is about to predict the properties of edges (i.e., existence, edge type, etc). Link prediction[1] can be applied to a wide range of applications like drug re-propusing (Tsubaki et al., 2019), drug-drug interaction prediction, etc.

Intuitively, if it takes a long distance to walk from vertex u to v over a graph, it is very likely that the two vertexes are not closely related and there should be no link/edge between them; in contrast, if the path between them is short over the graph, it is likely that they are close to each other and there may be a link between them.

Graph neural networks (GNNs) receive much attention for graph related problems (Hamilton et al., 2017; Kipf & Welling, 2017; Veličković et al., 2018) recently. Previous methods based on GNNs do not explicitly take the distance information between vertices into consideration. In this work, we propose to combine distance information with GNNs and propose a new method, *distance-enhanced* GNN for drug-drug interaction (briefly, DDI) and protein-protein-association (i.e., PPA), that can significantly improve the performance.

The technical challenge of combining distance information with GNNs is that in training we need to calculate the distance of many vertex pairs, which is computational costly over a large graph (e.g., with millions of nodes). To improve efficiency, we propose an anchor-based distance. Specifically, we first randomly select K_A vertices from the node set and denote them as anchor vertices. Then we calculate the shortest path starting from these anchor vertices to any vertices. After that, we use the average of the distances between the two vertices and each anchor vertex as an estimation. Compared to the shortest path, calculating anchor distances requires $O(K_A(V+E))$ time complexity only, where K_A is the number of anchor vertices.

After obtaining the distances, they are fused with standard GNN (Hamilton et al., 2017; Veličković et al., 2018; Kipf & Welling, 2017). Specifically, the GNN modules first output a representation of the graph, which is then concatenated with the obtained distances as an enriched representation and processed by the classifier for the eventual decision. To further increase the efficiency, we introduce a distance sampler for distances of negative edges, by which we do not

[1]Shanghai Jiao Tong University, Shanghai, China [2]Microsoft Research, Beijing, China. Correspondence to: Yingce Xia <yingce.xia@microsoft.com>. This work was done when Boning Li was an intern at Microsoft Research.

The 2021 ICML Workshop on Computational Biology. Copyright 2021 by the author(s).

[1]In this work, "link" and "edge" are used alternatively, which both represent the connection between two nodes.

need to obtain all distances of negative edges in advance. We conduct experiments on three link prediction tasks on the OGB-DDI dataset (Hu et al., 2020).

2. Background

In this section, we first define the notations and the mathematical problem definition of link prediction. Then we introduce the background of GNN, followed by using GNN for link prediction. We focus on the the undirected graph in this work.

Notations and problem definition: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where $\mathcal{V}$ and $\mathcal{E}$ are the collections of vertices and edges respectively. Each edge $e \in \mathcal{E}$ can be equivalently represented as a vertex pair (u_e, v_e), where $u_e, v_e \in \mathcal{V}$ and the vertices of edge e. Let $\mathcal{N}(u)$ denote the neighbors of vertex u, which is $\{v \in \mathcal{G} | (u, v) \in \mathcal{E}\}$.

$\mathcal{E}$ is split into training, validation and test sets, which are denoted as $\mathcal{E}_{\text{train}}$, $\mathcal{E}_{\text{valid}}$ and $\mathcal{E}_{\text{test}}$ respectively. For ease of reference, we name the edges in $\mathcal{E}$ positive edges. To train an edge predictor and evaluate its performance, we also need some negative edges, which corresponds to the vertex pairs without edges, i.e., $\bar{\mathcal{E}} = \mathcal{A} \backslash \mathcal{E}$, $\mathcal{A} = \{(u, v) | u, v \in \mathcal{V}\}$. For the validation and test sets, they are made up of "real" negative edges, which are verified by experts to ensure there are no edges between them. For training set, the negative edges are randomly sampled. That is, the negative edges in the training set could be the positive edges in validation and test sets.

Brief introduction of GNN: GNN is a type of neural network defined over graphs. Usually, an GNN model can output the representations of vertices and edges in a graph. A core module of GNN is about information aggregation. Let h_u^l denote the hidden representation of vertex u at the l-th layer of an GNN model, $u \in \mathcal{V}$. Generally, h_u^l can be obtained as follows:

$$h_u^l = \phi\Big(h_u^{l-1}, \psi(\{h_v^{l-1} | v \in \mathcal{N}(u)\})\Big), \tag{1}$$

where ψ is a permutation invariant function that can aggregate the information from the neighborhood, and ϕ can map the information from the bottom layer to the top layer. ψ can be implemented as a vanilla feed-forward (i.e., the MLP) layer (Kipf & Welling, 2017; Hamilton et al., 2017), attention layer (Veličković et al., 2018), gated convolutional layer (Bresson & Laurent, 2017), etc. ϕ is usually implemented as an MLP layer with non-linear activation.

Link Prediction To predict whether a link exists between node u and v, we first merge the representations of vertices of u and v into an edge representation. Let h_u and h_v denote the output of the last layer of the GNN model (i.e, for an L-layer GNN module, h_u and h_v are the h_u^L and h_v^L respectively in equation 1) Following (Hu et al., 2020), the edge presentation $h_{u,v}$ is obtained as follows:

$$\begin{aligned}
&h_{u,v}^{(0)} = h_u \odot h_v;\\
&h_{u,v}^{(k)} = \texttt{ReLU}(W_k h_{u,v}^{(k-1)} + b_k),\ k \in \{1, 2, \cdots, K-1\};\\
&h_{u,v}^{(K)} = W_k h_{u,v}^{(K-1)} + b_K, \quad h_{u,v} = h_{u,v}^{(K)}.
\end{aligned} \tag{2}$$

where $\odot$ is element-wise product, the W's and b's are learnable parameters, $h_{u,v} \in \mathbb{R}^{d_0}$, and d_0 and K are hyperparameters. That is, the vertex representations are processed by K MLP layers with ReLU activation, and we eventually get a d_0-dimension representation $h_{u,v}$.

3. Our Method

Network Architecture: The overall architecture is shown in Figure 1. For the input graph $(\mathcal{V}, \mathcal{E}_{\text{train}})$, we use the standard GNN network (denoted as $\texttt{GNN}$), which could be GCN, GAT, GraphSage, to extract features of each vertex following equation 1. Then we can obtain the representations of edges by equation 2.

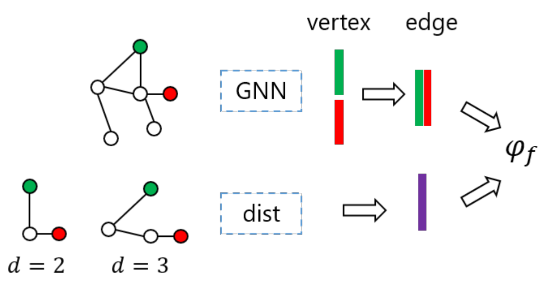

Figure 1. Model architecture. The task is to predict whether there is a link between the green vertex and the red vertex. We use GNN to extract the vertex representations and merge them as an edge feature. We then obtain the features about distances (e.g., shortest path, anchor-based distance, etc). The edge features and distances features are fused for link prediction.

The distance information is represented by a d_1-dimension vector dist, where each element represents a specific distance (e.g., shortest path, anchor-based distance, etc). We concatenate $h_{u,v}$ and dist to get a $d_0 + d_1$ representation $\tilde{h}_{u,v} = \texttt{concat}(h_{u,v}, \text{dist}_{u,v})$ with distance information explicitly incorporated. $\tilde{h}_{u,v}$ will be eventually processed by a binary classifier $\varphi_f : \mathbb{R}^{d_0+d_1} \mapsto [0, 1]$ to determine whether there exists an edge between u and v.

Anchor-based distance We propose an efficient distance metrics, anchor-based distance, for our method. We randomly pick K_A nodes $a_1, a_2, \cdots a_{K_A}$ as anchor vertices. For each anchor point a_i, we use breadth-first search to calculate the shortest path from a_i to all non anchor points in the graph. The anchor-based distance between node u and node v is defined as follows:

$$\text{dist}^a_{u,v} = \frac{1}{K_A}\sum_{i=1}^{K_A} \text{dist}(u, a_i) + \text{dist}(a_i, v). \tag{3}$$

The time complexity of obtaining anchor-based distances is $O(K_A|\mathcal{E}|)$.

Training strategy: During training time, all edges in $\mathcal{A}\backslash\mathcal{E}_{\text{train}}$ could be negative edges. Usually in edge prediction tasks, $|\mathcal{E}_{\text{train}}| \ll |\mathcal{A}|$ (Li et al., 2020). Therefore, it is extremely cost to calculate the distances for all possible negative edges. Alternatively, we use a negative edge sampler to model the distance. When a negative edge (u, v) comes, the distance is uniformly sampled from $\mathcal{D} = \{\text{dist}_{u,v}|(u, v) \in \bar{\mathcal{E}}_{\text{train}}\}$. $\text{dist}_{u,v}$ can be any kind of distance, e.g., short-path distance, anchor-based distance.

The training strategy is shown in Algorithm 1. First, we generate M negative edges[2] and build $\bar{\mathcal{E}}_{\text{train}}$ for negative distance sampling. Next, we calculate distances for all positive and negative edges $(u, v) \in \mathcal{E}_{\text{train}} \cup \bar{\mathcal{E}}_{\text{train}}$. The distances of negative edges are put in a list $\mathcal{D}$.

Algorithm 1 Training Strategy.

```
**Input**: Vertices $\mathcal{V}$, positive edges in the training set $\mathcal{E}_{\text{train}}$.
Randomly sample $|\mathcal{E}_{\text{train}}|$ negative edges from $\mathcal{A}\backslash\mathcal{E}_{\text{train}}$ and denote them as $\bar{\mathcal{E}}_{\text{train}}$.
Calculate the distances $\text{dist}_{u,v}$ for any $(u, v) \in \mathcal{E}_{\text{train}} \cup \bar{\mathcal{E}}_{\text{train}}$; specifically, define $\mathcal{D} = \{\text{dist}_{u,v}|(u, v) \in \bar{\mathcal{E}}_{\text{train}}\}$.
**while** not convergence **do**
    Calculate the vertex representations: $\{h_u\}_{u\in\mathcal{V}} = \texttt{GNN}(\mathcal{V}, \mathcal{E}_{\text{train}})$.
    Sample a batch of positive edges $\mathcal{B}_+$ from $\mathcal{E}_{\text{train}}$; randomly generate $|\mathcal{B}_+|$ negative edges and denote them as $\mathcal{B}_-$; for each edge $(u_-, v_-) \in \mathcal{B}_-$, $\text{dist}_{u_-,v_-}$ is uniformed sampled from $\mathcal{D}$.
    Calculate the edge features $h_{u,v}$ $\forall(u, v) \in \mathcal{E}_{\text{train}} \cup \bar{\mathcal{E}}_{\text{train}}$ by Eqn.(2).
    Update the models by minimizing
        $$-\frac{1}{|\mathcal{B}_+|}\sum_{(u,v)\in\mathcal{B}_+} \log(\varphi_f(\texttt{concat}(h_{u,v}, \text{dist}_{u,v}))) - \frac{1}{|\mathcal{B}_-|}\sum_{(u_-,v_-)\in\mathcal{B}_-} \log(1 - \varphi_f(\texttt{concat}(h_{u_-,v_-}, \text{dist}_{u_-,v_-}))). \tag{4}$$
**end while**
**Return** the well-trained GNN and $\varphi_f$.
```

We will keep training until convergence. At each iteration, we first calculate the vertex representation of $\mathcal{V}$. After that, we sample a minibatch of positive edges $\mathcal{B}_+$ from $\mathcal{E}_{\text{train}}$ and a minibatch of negative edges $\mathcal{B}_-$ with equal size. Note that $\mathcal{B}_-$ does not need to be a subset of the $\bar{\mathcal{E}}_{\text{train}}$ and could be sampled online. The distance of positive node are obtained in advance, while the distance of node pairs $(u, v) \in \bar{\mathcal{E}}_{\text{train}}$ are randomly sampled from $\mathcal{D}$.

We obtain the representation about vertex u and v by Eqn.(2) and the incorporate the distance information $\text{dist}_{u,v}$. We use Eqn.(4) to update the models.

4. Experiments

4.1. Settings

Datasets: We conduct experiments on the DDI and PPA datasets from the OGB benchmark (Hu et al., 2020). DDI is a dataset about predicting drug-durg interactions. Each vertex represents a drug. The edge represents the interactions between drugs and and can be interpreted as a phenomenon where the joint effect of taking the two drugs together is considerably different from the expected effect in which drugs act independently of each other. In PPA, each vertex represents a protein, and the edge indicates the biologically meaningful associations between proteins. The training, validation and test sets have been officially released by (Hu et al., 2020). More detailed statistics is at Table 1.

Configuration: The detailed settings of the experiments (including the network architectures, the number of layers, hidden dimension, training epoch, initial learning rate, the number of random trees, the number of anchor points and the number of independent runs) are summarized in Table 2.

Evaluation: The evaluation metrics for DDI and PPA are Hits@20 and Hits@100. Given a set of positive edges $\{e_{+,i}\}_{i=1}^{N_+}$ and negative edges $\{e_{-,j}\}_{j=1}^{N_-}$, let $p_{+,i}$ and $p_{-,j}$ denote the probability that the i-th positive edge and j-th negative as categorized as positive edges. Define the τ-th largest element in $\{p_{-,j}\}_{j=1}^{N_i}$ as p_τ. The Hits@τ (τ is an integer) is defined as follows:

$$\text{Hits@}\tau = \frac{\sum_{i=1}^{N_+}\mathbb{I}(p_{+,i} \geq p_\tau)}{N_+}, \tag{5}$$

where $\mathbb{I}$ is the indicator function. The script to calculate the Hits@τ is officially provided by (Hu et al., 2020).

Name	Nodes	Edges	Metric
OGBL-DDI	4267	1334889	Hits@20
OGBL-PPA	576289	30326273	Hits@100

Table 1. Statistics of OGB datasets (Hu et al., 2020).

[2] https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/utils/negative_sampling.html

	DDI	PPA
network	GraphSage	GCN
#layer	2	3
#hidden	500	200
epoch	500	200
lrate	0.003	0.002
dropout	0.3	0.05
K_A	1000	2000
#runs	30	5
embedding param	2184704	115257800
network param	1575430	1132012
total param	3760134	116389812

Table 2. Detailed settings of the experiments. From top to bottom, the rows represent the network architectures, the number of layers, hidden dimension, training epoch, initial learning rate and the number of anchor points.

4.2. Results

The results of DDI are reported in Table 3. We have the following observations:

1. After incorporating one distance into GNN, (i.e., the shortest path, Katz index, anchor-based distance), the baseline can be significantly improvement compared with the standard GraphSage baseline.
2. Using shortest path is not the best choice for DDI, either for computation efficiency and the performance. The reason is that DDI is a relatively dense graph, and the shortest paths cannot distinguish the positive and negative edges significantly.

Algorithm	Test Hits@20
GraphSage	79.32 ± 5.42
+ (S)	80.92 ± 5.92
+ (K)	81.24 ± 5.23
+ (A)	82.39 ± 4.37

Table 3. Results of DDI. Let (S), (K) and (A) denote the shortest distance, Katz index (with $\beta = 0.03$) and anchor-based distance respectively.

The results of PPA can be found in Table 4. With anchor-based distance, the baseline can be boosted from 48.23 to 49.58, which demonstrates the effectiveness of our method.

Algorithm	Validation	Test
GCN	50.71 ± 0.77	48.23 ± 2.24
+ (S)	51.69 ± 0.18	48.83 ± 1.22
+ (A)	50.92 ± 1.29	49.58 ± 1.45

Table 4. Results of PPA. Let (S) and (A) denote the shortest distance and anchor-based distance respectively.

5. Related Work

There are some metric based methods for link prediction with the intuition that the distances of two related nodes should be short. The metrics include Jaccard Index (Jaccard, 1901), the shortest path (Floyd, 1969), random walk (PEARSON & KARL, 1905), Katz Index (Katz, 1953), simRank (Jeh & Widom, 2002), pageRank (Brin & Page, 1998), etc. (Hasan & A, 2005; Liben-Nowell & Kleinberg, 2007) showed that local similarity metrics and global similarity metrics have significant effect on the link prediction task in social network.

In additional to the above manually-designed metrics, the features of nearby subgraphs can be obtained by neural networks. (Zhang & Chen, 2017) proposed Weisfeiler-Lehman Neural Machine (WLNM), which extracts enclosing subgraphs around edges as training data. (Zhang & Chen, 2018) proposed SEAL, a new link prediction framework based on GNN. SEAL learns "heuristic" suits the current network automatically by using a function mapping the sub-graph patterns to link existence. LGLP (Cai et al., 2020) improved SEAL by refactoring the graph to ensure the number of nodes and removing the pooling operation. PLACN (Ragunathan et al., 2020) incorporates information like common neighbors of nodes on target edge and a combination of heuristic features through a deep learning method like SEAL. (You et al., 2019) propose Position-aware Graph Neural Networks (P-GNNs). In P-GNNs, each vertex is attached an additional location feature, which is obtained by the distance from several anchor vertices in the graph. (Li et al., 2020) theoretically studies the distance encoding of GNN. (Cao et al., 2019) used local structural information in additional to the graph embedding, and SEMAC (Cao et al., 2018) takes the subgraphs with different depths into consideration.

6. Conclusions

In this paper, we propose a new method, distance-enhanced GNN, that explicitly incorporates distance into graph neural networks. Experiments on DDI and PPA demonstrate the effectiveness of our method. For future work, we will study more ways to leverage the distance information.

References

Berg, R. v. d., Kipf, T. N., and Welling, M. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, April 1998. ISSN 0169-7552. doi: 10.1016/S0169-7552(98)00110-X. URL http://dx.doi.org/10.1016/S0169-7552(98)00110-X.

Cai, L., Li, J., Wang, J., and Ji, S. Line graph neural networks for link prediction, 2020.

Cao, R. M., Liu, S. Y., and Xu, X. K. Network embedding for link prediction: The pitfall and improvement. *Chaos: An Interdiplinary Journal of Nonlinear ence*, 29(10):103102–, 2019.

Cao, Z., Wang, L., and de Melo, G. Link prediction via subgraph embedding-based convex matrix completion. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 2803–2810. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16442.

Floyd, R. W. Algorithm 97: Shortest path. *Comm Acm*, 5, 1969.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Hasan, M. and A, M. Link prediction using supervised learning. 01 2005.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Jaccard, P. Etude comparative de la distribution florale dans une portion des alpes et des jura. *bulletin del la societe vaudoise des sciences naturelles*, 37(142):547–579, 1901.

Jeh, G. and Widom, J. Simrank: A measure of structural-context similarity. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2002. doi: 10.1145/775047.775126.

Katz, L. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding – design provably more powerful graph neural networks for structural representation learning, 2020.

Liben-Nowell, D. and Kleinberg, J. The link-prediction problem for social networks. *Journal of the American Society for Information ence and Technology*, 58(7):1019–1031, 2007.

Nguyen, T., Le, H., and Venkatesh, S. Graphdta: prediction of drug–target binding affinity using graph convolutional networks. *BioRxiv*, pp. 684662, 2019.

PEARSON and KARL. The problem of the random walk. *Nature*, 72(1865):294, 1905.

Ragunathan, K., Selvarajah, K., and Kobti, Z. Link prediction by analyzing common neighbors based subgraphs using convolutional neural network. In Giacomo, G. D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., and Lang, J. (eds.), *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 1906–1913. IOS Press, 2020. doi: 10.3233/FAIA200308. URL https://doi.org/10.3233/FAIA200308.

Tsubaki, M., Tomii, K., and Sese, J. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35(2):309–318, 2019.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

You, J., Ying, R., and Leskovec, J. Position-aware graph neural networks. *CoRR*, abs/1906.04817, 2019. URL http://arxiv.org/abs/1906.04817.

Zhang, M. and Chen, Y. Weisfeiler-lehman neural machine for link prediction. pp. 575–583, 08 2017. doi: 10.1145/3097983.3097996.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.