Graph Convolutional Network using a Reliability-based Feature Aggregation Mechanism

Yanling Wang[1,2], Cuiping Li[1,2], Jing Zhang[1,2], Peng Ni[1,2], and Hong Chen[1,2]

[1] Key Laboratory of Data Engineering and Knowledge Engineering, Renmin University of China, Beijing, China
[2] School of Information, Renmin University of China, Beijing, China
{wangyanling, licuiping, zhang-jing, nipeng, chong}@ruc.edu.cn

Abstract. Graph convolutional networks (GCNs) have been proven extremely effective in a variety of prediction tasks. The general idea is to update the embedding of a node by recursively aggregating features from the node's neighborhood. To improve the training efficiency, modern GCNs usually sample a fixed-size set of neighbors uniformly or sample according to nodes' importance, instead of using the full neighborhood. However, both the sampling strategies ignore the reliability of a link between the target node and its neighbor, which can be implied by the graph structure and may seriously impact the performance of GCNs. To deal with this problem, we present a Graph Convolutional Network using a Reliability-based Feature Aggregation Mechanism called GraphRFA, where we sample the neighbors for each node according to different kinds of link reliability and further aggregate feature information from different reliability-specific neighborhoods by a dual feature aggregation scheme. We also theoretically prove that our aggregation scheme is permutation invariant for the graph data, and provide two simple but effective instantiations satisfying such scheme. Experimental results demonstrate the effectiveness of GraphRFA on different datasets.

Keywords: Graph convolutional network · Link reliability · Neighborhood definition · Dual feature aggregation scheme.

1 Introduction

Node embedding techniques learn low-dimensional representations for nodes in graphs and effectively preserve the structure information [6, 12]. Among related algorithms [1, 7, 8, 10, 13, 15, 21, 22, 24], graph convolutional networks (GCNs) adopt deep learning on the graph data and achieve outstanding performance on various tasks, such as node classification, community detection, link prediction and so on [13, 15, 26, 29]. The general idea behind GCNs is to update the representation of each node by aggregating the features from the node's neighborhood. The feature information of a node can be initialized by extracting the side information of the node. In traditional models, all the n-hop neighbors

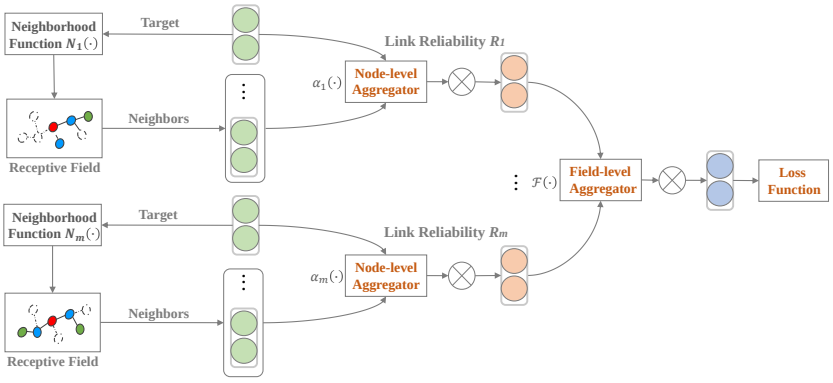


Fig. 1. The overview framework of GraphRFA.

form the neighborhood (i.e., the receptive field) of a given node, where n is less than or equal to the number of layers in the model. Since GCNs learn the node representations through recursive neighborhood aggregation, the large number of neighbors will lead to computational inefficiency. To make the memory and runtime of a single batch training controllable, modern GCNs usually sample a fixed-size set of neighbors for each node, instead of using the full neighborhood.

The uniform sampling strategy treats every node equally [4, 13]. Intuitively, not all the neighbors are useful for the target node, some non-uniform strategies are thereby designed to sample a set of neighbors according to the global node-importance such as pagerank centrality [3, 5, 25]. However, most methods do not consider the local correlations between the target node and its neighbors, making some important neighbors of the target node ignored. For example, in a social network, if one of your close friends is not an influential user in the whole social network, her/his influence on you will be weakened. Besides, a graph contained spurious connections can lead to ineffective graph convolution caused by the unreliable neighbors [27].

This paper aims to propose a link-reliability based GCN model, which pays attention to the likelihood that a link truly exists in a graph, i.e., the link reliability [11]. A higher score of link reliability indicates a more reliable relationship between two nodes. To achieve the goal, we try to answer the following questions: how to effectively measure the link reliability? And how to incorporate the receptive fields resulted by different measures together? To address these challenges, we present GraphRFA, a Graph Convolutional Network using a Reliability-based Feature Aggregation Mechanism. Figure 1 illustrates the overview framework of our model. Specifically, the model contains two main components:

Link-reliability based Neighborhood Definition: To measure the link reliability, we adopt random-walk methods to capture the characteristics of the graph structure from two views, i.e., the similarity between two nodes and the centrality of a link. Then we use a drop-bottom approach to sample neighborhood for each node, which drops the most unreliable neighbors and uniformly samples

a fixed-size set of neighbors from the remaining ones. Compared with previous strategy, we allow the moderately reliable neighbors to be sampled. Moreover, we can adjust the ratio of dropped neighbors to trade off between uniform sampling and non-uniform sampling.

Dual Feature Aggregation: Since we measure the link reliability from different views such as similarity and centrality, we design a dual feature aggregation scheme to perform convolution across different receptive fields, which are composed by the neighbors sampled from different views. More specifically, a node-level aggregator aggregates the node features from each receptive field for each node, then a field-level aggregator merges the features from different receptive fields for each node. Unlike learning over images and sentences, the neighbors of a node have no natural order [13]. So we give theoretical analysis on our aggregation scheme to guarantee the permutation invariant requirement.

The contributions of our work can be summarized as follows:

- We propose a graph convolutional network GraphRFA, which incorporates the link reliability in neighborhood definition to capture the reliable information from the graph structure and the node features.
- In GraphRFA, different measures of link reliability can be supported simultaneously, and a dual feature aggregation scheme is proposed to aggregate feature information from the different kinds of reliability-specific receptive fields. We also give the theoretical analysis to guarantee our aggregation scheme is permutation invariant for the graph data.
- We evaluate our algorithm by downstream tasks. Experimental results show the effectiveness of GraphRFA on different graphs.

2 Related Work

Graph Convolutional Networks. Recently, a large number of graph convolutional networks are proposed, which generalize the convolution operation to the permutation invariant graph data and achieve outstanding performance on various tasks over graphs. As one of the pioneer works, [15] is designed based on the full graph Laplacian, thus it is hard to generate embeddings for previously unseen nodes efficiently. GraphSAGE [13] solves the efficiency problem by a sampling-based inductive learning method. Graph Attention Network (GAT) [22] pays different attentions to different neighbors by adding a self-attention layer. Many advanced variants are proposed [18, 24]. For example, GIN [24] formally characterizes the properties of a powerful graph neural network and is as powerful as Weisfeiler-Lehman graph isomorphism test (WL test) [23].

Neighborhood Definition. The basic idea behind existing graph convolutional networks is to aggregate information from the neighborhood of the target node. Here, we summarize two kinds of approaches to define the neighborhood.

Adaptive learning approaches aim to learn the weights of different nodes in the receptive field. GAT [22] and GeniePath [18] are the representative approaches. GAT uses the self-attention mechanism to determine the importance

of each one-hop neighbor. As an extension of GAT, GeniePath uses the LSTM-like function to filter signals from the n-hop neighbors. Nevertheless, both GAT and GeniePath do not emphasize the neighborhood sampling.

Sampling-style approaches aim to reduce the size of the receptive field. Typically, existing approaches reduce the receptive field by sampling a set of neighbors for each node uniformly [13] or sampling according to the nodes' importance [3, 5, 25]. Although previous sampling-style GCNs have achieved outstanding performance, the authors of [4] said there was no theoretical guarantee on the convergence of the stochastic training algorithm. Thus, they propose a novel control variate based algorithm, denoted as CV-GCN, which achieves a similar convergence with the exact algorithm.

Discussion. In summary, neighborhood definition is a fundamental step in GCNs. To ensure the scalability, we aim to design a sampling-style algorithm. Distinct from the previous sampling-based algorithms, we consider the local correlations between target node and its neighbors from different views, hence we incorporate different kinds of link reliability in the sampling process.

3 Problem Formulation

3.1 Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where each $v \in \mathcal{V}$ denotes a node and each $e \in \mathcal{E}$ denotes a link. Notation $X_v \in \mathbb{R}^{d_0}$ indicates the initial feature vector of node v. In real datasets, there are usually noisy links contained in $\mathcal{E}$.

The problem is to learn a representation for each node v in $\mathcal{G}$, such that the representation can not only capture the characteristics of the local graph structure of v, but also emphasize the reliability of the links between target node and its neighbors. In other words, we aim to learn the embeddings $h_i, h_j \in \mathbb{R}^d$ for nodes v_i and v_j such that h_i, h_j can be well distinguished if their local neighborhoods are structurally different or the link between v_i and v_j are unreliable.

3.2 Background

In the deep graph representation learning, hidden feature of node v at the l-th layer of the model is denoted by $h_v^{(l)} \in \mathbb{R}^{d_l}$, and $h_v^{(0)} = X_v$.

Graph convolution in [15] requires to calculate the propagation matrix $\hat{A} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$. $\widetilde{A}$ is the adjacency matrix of $\mathcal{G}$ with self-connections. $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. The matrix of hidden features at the l-th layer is updated by:

$$H^{(l)} = \sigma\left(\hat{A}H^{(l-1)}W^{(l)}\right) \tag{1}$$

where the weight matrix $W^{(l)}$ is learnable, and σ is an activation function.

GraphSAGE [13] trains a series of aggregators to aggregate information from the node's neighborhood, so that the trained aggregators can generate embeddings for previously unseen nodes effectively. [24] summarizes the strategy of neighborhood aggregation by AGGREGATE and COMBINE steps, given the one-hop neighborhood $\mathcal{N}(v)$ of node v, the l-th layer updates $h_v^{(l)}$ by:

$$h_{\mathcal{N}(v)}^{(l)}=AGGREGATE^{(l)}\left(\{h_u^{(l-1)}|u\in\mathcal{N}(v)\}\right),h_v^{(l)}=COMBINE^{(l)}\left(h_{\mathcal{N}(v)}^{(l)},h_v^{(l-1)}\right) \tag{2}$$

Among different kinds of aggregators, GCN-aggregator [13] is an inductive variant of [15], and is essentially a modified mean-based function:

$$h_v^{(l)}=\sigma\left(W^{(l)}\cdot MEAN\left(h_v^{(l-1)}\cup\{h_u^{(l-1)},\forall u\in\mathcal{N}(v)\}\right)\right) \tag{3}$$

4 The Proposed Approach

In this section, we describe the structure of our model. GraphRFA consists of two major components: *neighborhood definition* and *dual feature aggregation scheme*. Distinct from existing studies on the neighborhood definition, we sample neighbors for each node based on link reliability and devise a drop-bottom strategy to trade off between uniform sampling and non-uniform sampling. To further simultaneously support different measures of link reliability, a dual feature aggregation scheme is proposed to aggregate feature information from the receptive fields resulted by different reliability measures.

4.1 Neighborhood Definition based on Link Reliability

We define the fixed-size neighborhood based on the measures of link reliability. The neighbors connected by more reliable links should propagate more useful information to the target nodes.

Random-walk based Measures for Link Reliability. Different measures of link reliability can be used in GraphRFA. To propose a general algorithm, we measure the link reliability based on graph structure without using any attribute of a node or a link. Specially, *similarity* and *centrality* can be regarded as the views of designing the link-reliability measures [20, 28].

Under the similarity hypothesis, links connecting similar nodes are supposed to have high existence-likelihoods [20]. Following such an assumption, many well-known path-dependent indices are proposed, such as CN (Common Neighbors) [16] , LP (Local Path) [30] and Katz index [14]. The number of paths between two nodes is used to quantify the link reliability. Intuitively, the highly interconnected nodes usually tend to be similar.

Under the centrality hypothesis, links with higher centrality is more reliable [28]. In other words, links acting as important bridges deserve high reliability scores, because they directly or indirectly connect a large number of nodes in a

graph. Both PA (Preferential Attachment) [2] and EB (Edge Betweenness) [9] are typical centrality-based indices.

Despite their success, most of the above measures are computational inefficient. Our goal is to adopt the random-walk based methods, as random walk is computationally efficient in terms of both space and time requirements. In this paper, we focus on the undirected graphs, although our method can be extended to other types of graphs.

Similarity-based Link Reliability: In traditional path-dependent indices such as LP and Katz, the reliability of the link (v_i, v_j) is proportional to the number of paths between v_i and v_j. From the perspective of random walk, the more paths between v_i and v_j exist, the more likely the random-walker walks between v_i and v_j. Therefore, we simulate local random walks with walk-length r starting from the initial node for t times, then compute the normalized visit-count to formulate the similarity-based link reliability:

$$R_{ij} = \frac{c_{i\to j} + c_{j\to i}}{\sum_{v_k\in\mathcal{V}} c_{i\to k} + \sum_{v_k\in\mathcal{V}} c_{j\to k}} \tag{4}$$

where $c_{i\to j}$ is the visit-count of v_j from the initial node v_i. Specially, in the neighborhood definition, we emphasize the neighbors that the target node tends to follow, so we sample neighbors according to the normalized visit-count from the target node to each of its neighbors through the local random walks.

Turn to traditional similarity-based measures, a unified framework $R_{ij} = \sum \beta^l |paths_{ij}^l|$ can formulate CN, LP and Katz, where for CN, $l = 2$, for LP, $l = 2, 3$, and for Katz, $l = 1, 2, 3, \cdots, +\infty$ [19]. $|paths_{ij}^l|$ represents the number of paths between v_i and v_j with path-length l. Free parameter β^l adjusts the weight of a path with path-length l. Particularly, the walk-length r in our method corresponds to the maximal path-length.

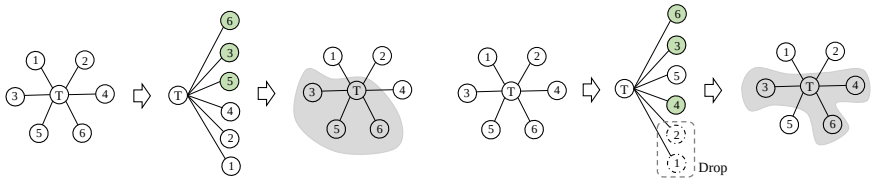
Centrality-based Link Reliability: We put a walker on the initial node v_i and simulate random walks over the graph. Let P be the transition probability matrix, the probabilities that the walker locates at any other nodes at the t-th step is computed by:

$$\pi_i(t) = P^T \pi_i(t-1) \tag{5}$$

At the stationary state, the probability that the walker locates at v_j is $\pi_{ij} = \frac{d_j}{2|E|}$, where d_j is the degree of v_j, $|E|$ is the number of links in graph [17]. In other words, the stationary probability that the random-walker starting from any node locates at node v_i is $\pi_{*i} = \frac{d_i}{2|E|}$. A higher probability π_{*i} indicates that it is more likely to locate at v_i. Thus, if both π_{*i} and π_{*j} are high probabilities, the walker can easily walk to both v_i and v_j from any node, which means a large number of nodes can be connected by the link (v_i, v_j). According to the above analysis, the centrality-based link reliability can be formulated as:

$$R_{ij} = \pi_{*i} \cdot \pi_{*j} = \frac{d_i}{2|E|} \cdot \frac{d_j}{2|E|} \propto d_i \cdot d_j \tag{6}$$

Specially, Eq. 6 precisely corresponds to the algorithm of Preferential Attachment (PA).



(a) Select-Top Approach (b) Drop-Bottom Approach

Fig. 2. Illustration of the link-reliability based neighborhood definition. In the example, the green nodes denote the sampled neighbors, the dashed nodes denote the dropped unreliable neighbors, and the grey areas indicate the resulted receptive fields.

Neighborhood Function. Given a node in a graph, the neighborhood function $\mathcal{N}(\cdot)$ draws S_l one-hop neighbors for the node at the l-th layer. We provide two approaches to define $\mathcal{N}(\cdot)$. The core ideas behind them are shown in Figure 2. Both approaches rank all one-hop neighbors for each node according to the link reliability between the node and its neighbors, while the difference is that:

Select-Top selects the top neighbors to form a fixed-size neighborhood. This straightforward approach has been utilized in previous attempts [3], which rank the one-hop neighbors according to nodes' importance rather than link reliability.

Drop-Bottom firstly drops the most unreliable $k\%$ neighbors and then uniformly samples from the remaining ones. $k\%$, named as dropping rate, is a hyper-parameter to be tuned to trade off between uniform sampling and non-uniform sampling.

4.2 Dual Feature Aggregation

Based on different measures of link reliability, we assign different receptive fields to each node. So we devise a dual feature aggregation scheme to perform convolution across different receptive fields from the node-level and the field-level.

Node-level Aggregator. The node-level aggregator acts on each receptive field. Theoretically, any candidate function satisfying the permutation invariant requirement can be applied. The default node-level aggregator in our model follows state-of-the-art model GIN [24], in which the graph convolution for node v_i at layer l is defined as:

$$h_i^{(l)} = MLP^{(l)}\left(\left(1+\epsilon^{(l)}\right)\cdot h_i^{(l-1)} + \sum\nolimits_{v_j \in \mathcal{N}(v_i)} h_j^{(l-1)}\right) \tag{7}$$

where $MLP^{(l)}$ is the multi-layer perceptron at the l-th layer, and ϵ is a learnable parameter or a fixed scalar.

Since different measures of link reliability are adopted in section 4.1, we derive a similarity-based neighborhood function $\mathcal{N}^s(\cdot)$ and a centrality-based neighborhood function $\mathcal{N}^c(\cdot)$ to produce the receptive field respectively. That means

we have two node-level aggregators $\alpha^s(\cdot)$ and $\alpha^c(\cdot)$ to compute the reliability-specific embeddings h^s, h^c for each node in a graph.

Field-level Aggregator. To further aggregate reliability-specific embeddings from different receptive fields, we develop the field-level aggregator. A basic requirement for the field-level aggregator $\mathcal{F}$ is the permutation invariant property, which is formally defined as:

Definition 1. *Let H be the feature set of nodes in graph G, and let σ be a permutation of $\{1, \cdots, n\}$. $\sigma(H)$ is a new feature set of nodes given by $[\sigma(H)]_i = H_{\sigma(i)}$. Mapping function $\mathcal{F}$ is said to be permutation invariant, if all the permutations σ of input H satisfy: $\mathcal{F}(\sigma(H)) = \sigma(\mathcal{F}(H))$.*

Associative property of permutation invariant manner has been discussed in [18]:

Remark 1. If the output of function α is invariant to the permutations of the input H, and function ρ is independent of the orders of H, $\rho \circ \alpha$ is still permutation invariant with respect to H.

Here, the function α corresponds to the node-level aggregator. Suppose we have m permutation invariant node-level aggregators, and each of them corresponds to a certain measure of link reliability, we need to prove that if we integrate different node-level aggregators, the result still satisfies the property of permutation invariance.

Theorem 1. *If the field-level aggregator $\mathcal{F}$ can be decomposed by Eq. 8, $\mathcal{F}$ is permutation invariant.*

$$\mathcal{F}(H) = \sum\nolimits_{i=1}^{m} \rho_i(\alpha_i(H)) \tag{8}$$

where the node-level aggregation function α_i is permutation invariant, and the function ρ_i is independent of the orders of input H.

Proof. We need to show that $\mathcal{F}$ in Theorem 1 satisfies the condition of Definition 1 (i.e., the permutation invariant property). In other words, for any permutation σ of input H, $\mathcal{F}(\sigma(H)) = \sigma(\mathcal{F}(H))$. Hence, we write $\mathcal{F}(\sigma(H))$ using Eq. 8 and derive the following equality:

$$\mathcal{F}(\sigma(H)) = \sum\nolimits_{i=1}^{m} \rho_i \circ \alpha_i(\sigma(H)) \tag{9}$$

The argument of α_i is invariant under the permutation σ, and ρ_i is independent of the orders of H. According to Remark 1, we can derive $\rho_i \circ \alpha_i(\sigma(H)) = \sigma(\rho_i \circ \alpha_i(H))$. Then, Eq. 9 can be rewritten as:

$$\mathcal{F}(\sigma(H)) = \sum\nolimits_{i=1}^{m} \sigma(\rho_i \circ \alpha_i(H)) = \sigma\left(\sum\nolimits_{i=1}^{m} \rho_i \circ \alpha_i(H)\right) = \sigma(\mathcal{F}(H)) \tag{10}$$

where the equality follows Definition 1. Thus, we prove that Eq. 8 is one kind of field-level aggregator that implies the permutation invariance.

In this paper, we mainly explore two simple but effective instantiations of Theorem 1.

Mean-pooling Aggregation: We take the weighted-mean of node's reliability-specific embeddings as the candidate field-level aggregator:

$$h_i = \lambda \cdot [\alpha_s(H)]_i + (1-\lambda) \cdot [\alpha_c(H)]_i \tag{11}$$

where λ is learnable or fixed. Eq. 11 is a simplest formalization of Theorem 1. [1]

Concatenation Aggregation: We concatenate the reliability-specific embeddings of node v_i, then perform a linear transformation to reduce the dimension. The weight and bias of transformation are shared across all nodes, which helps reduce over-fitting. Formally, the concatenation aggregator is defined as:

$$h_i = [[\alpha_s(H)]_i, [\alpha_c(H)]_i] \cdot W + b \tag{12}$$

Next, we show that Eq. 12 satisfies Theorem 1. Eq. 12 can be composed into the form of $([\alpha_s(H)]_i \cdot W_s + b_s) + ([\alpha_c(H)]_i \cdot W_c + b_c)$, i.e., $[\rho_s \circ \alpha_s(H) + \rho_c \circ \alpha_c(H)]_i$, where α is the reliability-specific node-level aggregator and ρ is the linear transformation.

4.3 Optimization

We try two kinds of optimizations. The first one leverages the labels of a downstream supervised task such as multi-label classification to optimize the cross-entropy loss function. The second one directly optimizes the popular graph-based loss function [13], so that the nearby nodes tend to have similar embeddings, while disparate nodes tend to have highly distinct embeddings. We optimize the loss function using Adam optimizer.

4.4 Complexity Analysis

When quantifying the link reliability, the centrality-based measure is proportional to the node degree, while we need to simulate local random walks to compute the similarity-based measure. We set r to be the walk length, and we repeat the random walks on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from each initial node for t times. The computation complexity of local random walk is $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}|rt)$ on a weighted graph while $\mathcal{O}(|\mathcal{V}|rt)$ on an unweighted graph. We rank all one-hop neighbors according to the link reliability. The computation complexity is $\mathcal{O}(|\mathcal{V}|d \cdot logd)$, where d is the average degree in $\mathcal{G}$. Due to the sparsity of most real graphs (i.e., $d \ll |\mathcal{V}|$), we simplify $\mathcal{O}(|\mathcal{V}|d \cdot logd)$ to $\mathcal{O}(|\mathcal{V}|)$. As for the graph convolution, the computation complexity is $\mathcal{O}\left(|\mathcal{V}| \cdot \prod_{l=1}^{L} S_l\right)$, where $S_l, l \in \{1, 2, 3, \cdots, L\}$ is the neighbor sampling size at the l-th layer, and $S_l \ll |\mathcal{V}|$. The convolution complexity of GraphRFA is the same as that of other sampling-style GCNs.

[1] We set $\lambda = 0.5$ in our experiments.

Table 1. Summary of the datasets used in our experiments

Dataset	#Nodes	#Edges	#Labels	#Features	#Aver Degree	#Max Degree
PPI-S	14,755	228,431	121	50	31	722
PPI	56,944	818,716	121	50	29	722
Retweet	18,470	48,053	2	79	5	786
Reddit	232,965	57,307,946	41	602	492	21657

5 Experiments

5.1 Experimental Settings

Datasets. The experiments are conducted on the following undirected graphs, detailed summary is shown in Table 1:

- **PPI**[2]: This is a protein-protein interaction network in human tissue. In this dataset, 4/5 nodes are used for training, and 1/5 nodes are used for testing (with 54% for validation). Specially, we have a smaller PPI network containing 14,755 nodes, denoted as **PPI-S**.
- **Retweet**[3]: This is a political retweet network. We treat this dataset as an undirected graph. We randomly select 2/3 nodes for training and 1/3 nodes for testing (with 30% for validation).
- **Reddit**[4]: This is a post network from Reddit made in the month of September, 2014. The data in the first 20 days is used for training, and the remaining is used for testing (with 30% for validation).

For PPI and Reddit, the train-validation-test splits and initial node features are provided by the original datasets. For the Retweet network without node attributes, we use one-hot encoding of int(degree/10) as the initial node feature.

Baselines. Among the following baselines, GraphCSC-M adopts importance based sampling to define the neighborhood for each node. The others adopt the uniform sampling. GeniePath-lazy learns the importance of different neighbors.

- **GraphSAGE**: This is a classic model for the inductive graph embedding, and it uses uniform sampling to control the size of receptive field.
- **GIN**: This is a state-of-the-art method designed for both node classification and graph classification. Emphatically, it performs as powerful as WL test.
- **GeniePath-lazy**: This algorithm is an extension of GAT and outperforms GAT shown in [18]. A path layer is designed for both breadth and depth exploration. Compared with GeniePath, GeniePath-lazy postpones the evaluations of the adaptive depth function. Through the experimental evaluation, GeniePath-lazy converges faster than GeniePath in most cases [18].

[2] http://snap.stanford.edu/graphsage/
[3] http://networkrepository.com/soc-political-retweet.php
[4] http://snap.stanford.edu/graphsage/

- **GraphCSC-M**: This is a typical GCN algorithm that defines the neighborhood according to nodes' importance, which is quantified by the node centrality in the whole graph, and this algorithm applies the attention mechanism to the graph convolution. The authors provide five possible centrality measures. Considering solvability and computation cost, we adopt degree-centrality and pagerank-centrality in the following experiments.
- **CV-GCN**: This sampling-style algorithm presents a control variate based method to achieve a similar convergence with the exact GCN [15], even using a small neighbor sampling size. CV and CVD are two estimators proposed in CV-GCN, so we choose the better result to present the model's performance.

Parameter Settings. We implement our algorithm in TensorFlow with Adam optimizer. All models are implemented under the inductive framework proposed by [13] except CV-GCN. All models adopt neighbor sampling to guarantee the scalability. We set the number of layers, neighbor sampling size, output dimension and batch size to be the same for different models. In detail, we set the number of layers to be 2, the neighbor sampling size to be $S_1 = 25, S_2 = 10$, the output dimension to be 256, the training batch size to be 512 and the validation batch size to be 256. Following [13], we subsample edges so that the maximum degree is 125. Specially, we adjust the neighbor sampling size and maximum degree when comparing with CV-GCN. Referring to the experimental setups in [24], we fix ϵ in Eq. 7 to be 0. In GIN and GraphRFA, we concatenate the node's previous layer representation with the aggregated feature vector computed by Eq. 7, and we find the concatenation operation can lead to a better prediction result. We simulate random walks with walk-length 3 starting from each initial node for 100 times to measure the similarity-based link reliability. Then we use the drop-bottom strategy to sample neighbors and perform a parameter sweep on dropping rates {0.2, 0.4, 0.6, 0.8}.

Evaluation Settings. We optimize GraphRFA under the supervised setting and unsupervised setting. Under the unsupervised setting, we put the learned embeddings to a logistic regression classifier for node classification. Specially for the Reddit, we randomly sample 10000 nodes from the training data and 6000 nodes from the testing data after finishing the unsupervised representation learning, and conduct the logistic regression based on the sampled nodes. We treat the binary classification on the Retweet as a special case of multi-class classification, and use Micro F1 score and Macro F1 score for evaluation on all datasets. Analogous trends hold for F1 score on the Retweet. Please note the theoretical guarantee of CV-GCN is proved under the supervised setting [4], so we do not evaluate CV-GCN under the unsupervised setting.

5.2 Experimental Results

Comparison with the traditional sampling-style baselines. In this section, we ran all models for 300 times. Note that GraphCSC-M proposed by [3]

Table 2. Comparison under the supervised setting. Detail testing results of the node classification are shown (epoch=300).

	Micro F1				Macro F1			
	PPI-S	PPI	Retweet	Reddit	PPI-S	PPI	Retweet	Reddit
GraphSAGE-GCN	0.5455	0.5525	0.8889	0.9255	0.3624	0.3738	0.8812	0.8865
GIN	0.7058	0.8099	0.8995	0.9460	0.6177	0.7656	0.8924	0.9195
GeniePath-lazy	0.7908	0.8808	0.8983	0.9491	0.7364	0.8559	0.8903	0.9216
GraphCSC-M*	0.5918	0.7264	0.9374	0.8823	0.4967	0.6635	0.9328	0.8229
GraphRFA (mean)	0.7357	0.8491	0.9331	0.9510	0.6633	0.8166	0.9286	0.9274
GraphRFA(concat)	0.7588	0.8624	0.9353	0.9515	0.6949	0.8338	0.9307	0.9292
GraphRFA* (concat)	0.7671	0.8883	0.9353	0.9515	0.7090	0.8662	0.9307	0.9292

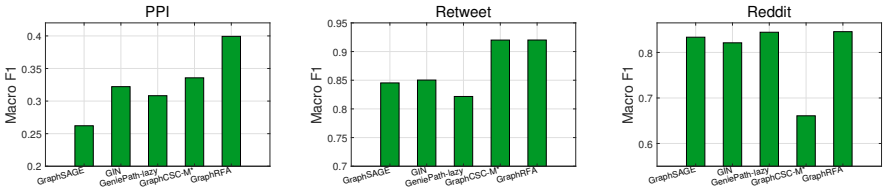


Fig. 3. Comparison under the unsupervised setting. Detail testing results of the node classification are shown (epoch=300). Here GraphSAGE denotes GraphSAGE-GCN. Similar trends hold for the Micro F1 score.

incorporates the node-centrality in loss function, however we focus on evaluating the methods of neighborhood definition and graph convolution. Therefore, we need to keep loss function the same for all models. The candidate loss functions are introduced in section 4.3. In light of this, the examined model is essentially a variant of GraphCSC-M, denoted as GraphCSC-M*.

Supervised Learning: We perform a parameter sweep on initial learning rates {0.01, 0.001, 0.0001}. Experimental results are shown in Table 2. As an extension of GIN, our model consistently outperforms GIN. Among these baselines, GeniePath-lazy learns the importance of different neighbors and aggregates node features weighted by the learned importance, however such scheme will perform better on the algorithm using node's full neighborhood [18], i.e., the neighbor sampling may affect the effectiveness of GeniePath-lazy, while using the full neighborhood will lead to a large receptive field. Besides, we can see that GeniePath-lazy obtains better performance under the supervised setting. That is to say the task labels are necessary for learning the neighbors' importance. GraphCSC-M* uses centrality-based sampling to form the receptive field. Compared with this algorithm, our model performs more stable, especially on PPI and Reddit. In our model, the concatenation aggregator works better, which can be explained by the more sufficient feature interaction, so we use GraphRFA (concat) for evaluation in the following experiments. Moreover, we permit the dropping rates about different link reliability to be distinct, de-

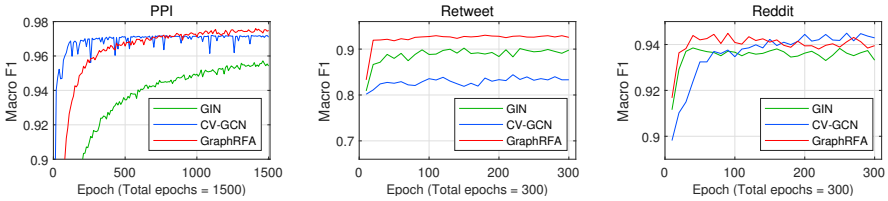


Fig. 4. Comparison with CV-GCN. Validation results of the supervised node classification are shown. (best seen in color).

noted as GraphRFA*. Compared with GraphRFA, GraphRFA* obtains better performance on the PPI.

Unsupervised Learning: We perform a parameter sweep on initial learning rates {0.01, 0.001, 0.0001}. From Figure 3, we can see that GraphCSC-M* outperforms other baselines on PPI and Retweet, while this algorithm is not very effective on the Reddit. In other words, the information of node-centrality is insufficient for the neighborhood definition. Thus, we choose to measure the correlations between target node and its neighbors. GeniePath-lazy aims to learn such correlations, while the learned correlations can not support the neighbor sampling. In addition, we observe that the uniform-sampling based algorithms (i.e., GraphSAGE, GIN and GeniePath-lazy) perform well on the Reddit, which means more diverse neighbors can provide more benefits, and that is why the drop-bottom strategy prefers a smaller dropping rate on the Reddit [5]. Conversely, a larger dropping rate leads to a better prediction result on both PPI and Retweet. More details about tuning the dropping rate will be discussed later.

Comparison with CV-GCN. CV-GCN enjoys a similar convergence with the exact algorithm even using two neighbors per node, so we set its neighbor sampling size to be 2. The initial learning rate of this model is provided in [4]. The neighbor sampling size S_l of GraphRFA is constrained within 50. In our model, we do not implement the preprocessing technique proposed in CV-GCN, which makes the first neighbor averaging exact. We observe that the supervised GraphRFA prefers a small dropping rate on both PPI and Reddit, thus we set the dropping rate to be 0 on PPI and Reddit in this experiment. Results are shown in Figure 4. GraphRFA outperforms CV-GCN on PPI and Retweet. Since we follow GIN to define the node-level feature aggregator, we also compare with GIN, and find that our model consistently outperforms GIN on every dataset. Perhaps benefiting from the powerful feature aggregator, GIN performs better than CV-GCN on the Retweet.

Parameter sensitivity: insight into how the drop-bottom works. The drop-bottom strategy trades off between uniform sampling and non-uniform sampling by adjusting the dropping rate, where the uniform sampling provides more

[5] We set the dropping rate to be 0.2 on the Reddit.

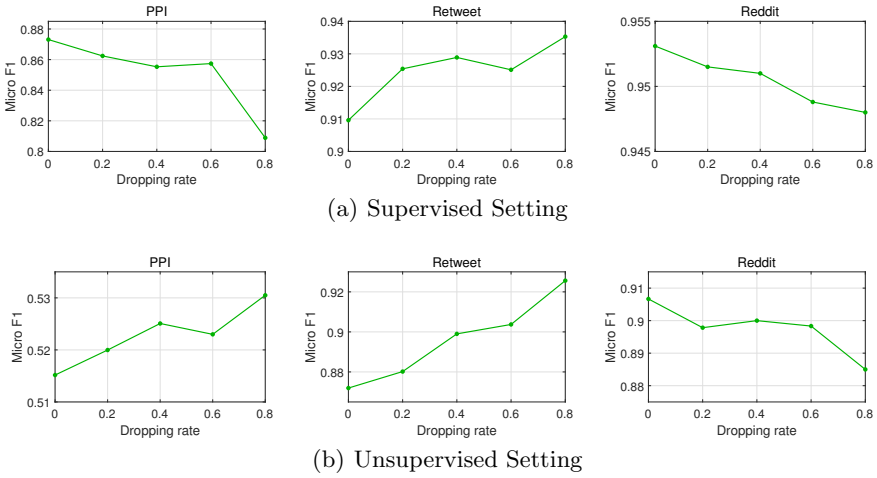


Fig. 5. Sensitivity analysis of dropping rate in the drop-bottom strategy. Testing results of supervised and unsupervised node classification are shown.

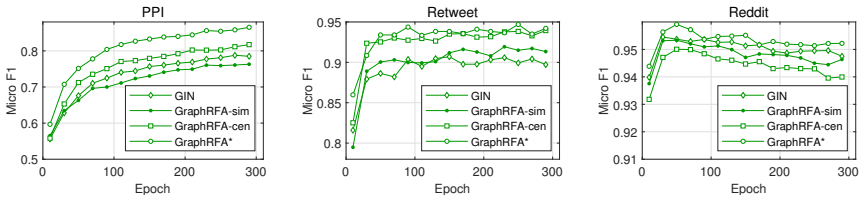


Fig. 6. Necessity of the field-level feature aggregation. Validation results of the supervised node classification are shown.

diverse node features, while the non-uniform sampling aims to capture features from the more reliable neighbors. Here we explore how different dropping rates influence the performance of GraphRFA. We run all models for 300 times.

Figure 5 shows the experimental results. Generally, we can see that supervised model prefers a smaller dropping rate than the unsupervised model. Since the supervised model is capable of learning a suitable aggregation function to weight different neighbors by the task labels, the more diverse neighbors can provide more benefits. On the contrary, the unsupervised model largely depends on the graph structure, thus highly reliable neighbors deserve more attentions due to the missing of the downstream labels. We can also observe the opposite choosing of dropping rate on Retweet and Reddit. Because in the political retweet network, users usually have clear attitude, which results in the more important role taken by the close neighbors than random neighbors on node representations. However, users in Reddit discuss various topics, and their interests are diverse. Only keeping the close neighbors may harm the performance of user representation learning.

Necessity of the field-level feature aggregation. In this section, we only consider one kind of measures (i.e., the similarity/centrality-based link reliability) then use the node-level aggregator to generate node representations, denoted as GraphRFA-sim and GraphRFA-cen. Specially, the adjusted dropping rates of GraphRFA-sim and GraphRFA-cen are directly applied to GraphRFA*. We also compare with GIN which adopts the uniform sampling. Taking the supervised setting as an example, Figure 6 shows that the centrality-based link reliability plays a key role on both PPI and Retweet, and the similarity-based link reliability is also necessary for the Retweet. GIN performs better than GraphRFA-sim and GraphRFA-cen on the Reddit, that is because uniform sampling provides more benefits on this dataset discussed in the previous section. The field-level aggregator improves the prediction result on the Reddit.

6 Conclusion

In this paper, we introduce an graph convolutional network, called GraphRFA, that captures reliable features from the node's neighborhood by a reliability-based feature aggregation mechanism. Theoretical analysis provides insight into how our model guarantees the permutation invariant property for graph data. In the experiments, we evaluate the effectiveness of GraphRFA on different datasets through supervised learning and unsupervised learning. Nonetheless, some extensions are possible too, such as learning an optimal dropping rate and extending GraphRFA to the heterogeneous graphs.

Acknowledgments. This work is supported by National Key R & D Program of China (No.2018YFB1004401) and NSFC under the grant No. 61532021, 61772537, 61772536, 61702522.

References

1. Ahmed, A., Shervashidze, N., Narayanamurthy, S.M., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: WWW (2013)
2. Barabási, A., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
3. Chen, H., Yin, H., Chen, T., Nguyen, Q.V.H., Peng, W., Li, X.: Exploiting centrality information with graph convolutions for network representation learning. In: ICDE (2019)
4. Chen, J., Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction. In: ICML (2018)
5. Chen, J., Ma, T., Xiao, C.: Fastgcn: Fast learning with graph convolutional networks via importance sampling. In: ICLR (2018)
6. Cui, P., Wang, X., Pei, J., Zhu, W.: A survey on network embedding. IEEE Trans. Knowl. Data Eng. **31**(5), 833–852 (2019)
7. Dave, V.S., Zhang, B., Chen, P., Hasan, M.A.: Neural-brane: Neural bayesian personalized ranking for attributed network embedding. Data Science and Engineering **4**(2), 119–131 (2019)

8. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NeurIPS (2016)
9. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences of the United States of America **99**(12), 7821–7826 (2002)
10. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD (2016)
11. Guimerà, R., Sales-Pardo, M.: Missing and spurious interactions and the reconstruction of complex networks. Proceedings of the National Academy of Sciences of the United States of America **106**(52), 22073–22078 (2009)
12. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. IEEE Data Eng. Bull. **40**(3), 52–74 (2017)
13. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS (2017)
14. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (1953)
15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
16. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology **58**(7), 1019–1031 (2007)
17. Liu, W., Lü, L.: Link prediction based on local random walk. EPL (Europhysics Letters) **89**(5), 58007 (2010)
18. Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., Qi, Y.: Geniepath: Graph neural networks with adaptive receptive paths. In: AAAI (2019)
19. Lü, L., Jin, C., Zhou, T.: Similarity index based on local paths for link prediction of complex networks. Physical Review E **80**(4 Pt 2), 046122 (2009)
20. Lü, L., Zhou, T.: Link prediction in complex networks: A survey. Physica A Statistical Mechanics and Its Applications **390**(6), 1150–1170 (2011)
21. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: KDD (2014)
22. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
23. Weisfeiler, B., Lehman, A.: A reduction of a graph to a canonical form and an algebra arising during this reduction. Nauchno-Technicheskaya Informatsia **2**(9), 12–16 (1968)
24. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2019)
25. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: KDD (2018)
26. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: NeurIPS (2018)
27. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph transformer networks. In: NeurIPS (2019)
28. Zeng, A., Cimini, G.: Removing spurious interactions in complex networks. Physical Review E **85**(3 Pt 2), 036101 (2012)
29. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: NeurIPS (2018)
30. Zhou, T., Lü, L., Zhang, Y.: Predicting missing links via local information. European Physical Journal B **71**(4), 623–630 (2009)