Data Mining and Knowledge Discovery (2019) 33:474–498
https://doi.org/10.1007/s10618-018-0601-y

DeepRank: improving unsupervised node ranking via link discovery

Yi-An Lai[1] · Chin-Chi Hsu[2] · Wen-Hao Chen[1] · Mi-Yen Yeh[2] · Shou-De Lin[1,3]

Received: 10 December 2017 / Accepted: 14 November 2018 / Published online: 2 January 2019

Abstract
This paper proposes an unsupervised node-ranking model that considers not only the attributes of nodes in a graph but also the incompleteness of the graph structure. We formulate the unsupervised ranking task into an optimization task and propose a deep neural network (DNN) structure to solve it. The rich representation capability of the DNN structure together with a novel design of the objectives allow the proposed model to significantly outperform the state-of-the-art ranking solutions.

Keywords Unsupervised learning · Node ranking · PageRank · Link prediction · Neural networks

1 Introduction

Evaluating the importance of nodes in a network is a fundamental research problem applicable to many real-world tasks such as spam web page detection (Gyöngyi et al.

Responsible editor: Jesse Davis, Elisa Fromont, Derek Greene, Bjorn Bringmann.

✉ Chin-Chi Hsu
chinchi@iis.sinica.edu.tw

Yi-An Lai
b99202031@ntu.edu.tw

Wen-Hao Chen
b02902023@ntu.edu.tw

Mi-Yen Yeh
miyen@iis.sinica.edu.tw

Shou-De Lin
sdlin@csie.ntu.edu.tw

1 Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

2 Institute of Information Science, Academia Sinica, Taipei, Taiwan

3 Research Center for Information Technology, Academia Sinica, Taipei, Taiwan

2004), paper impact determination (Wang et al. 2013), and link prediction (Backstrom and Leskovec 2011; Lichtenwalter et al. 2010). The difficulty in gathering ground-truth labels (i.e., whether one node should be ranked higher than another) incurs the demand for unsupervised solutions. For example, WSDM Cup 2016[1] and KDD Cup 2016[2] challenged the participants to rank the impact of papers and affiliation influence, respectively, in an academic network without providing any labeled training instances.

This paper focuses on addressing the following concerns of representative unsupervised node ranking algorithms, such as PageRank (Page et al. 1999) and HITS (Kleinberg 1999). First, these approaches generate ranking scores assuming the given network structure is complete. Such assumption might be too strong since in the real world one can hardly assume all connections in the graph are known. For instance, in a social network such as Facebook, it is very likely some friendship links are missing. Performing PageRank or HITS on the incomplete graph can over/under-estimate the importance of some nodes. Link prediction models (Backstrom and Leskovec 2011; Liben-Nowell and Kleinberg 2007; Lichtenwalter et al. 2010; Lü and Zhou 2011; Martínez et al. 2016; Menon and Elkan 2011; Newman 2001; Wang et al. 2015; Zhai and Zhang 2015) are designed to discover the missing links in a network. However, as far as we know, there are no previous works that attempt to recover the missing links to boost node ranking performance. Second, most of the existing models consider only network topology but not information associated with the nodes. In real-world networks, however, some information or attributes of a node are given, like user profiles or website metadata. It is conceivable that the quality of ranking can be boosted by leveraging such information.

A naïve two-stage solution to incorporate link prediction with unsupervised node ranking is described here. One can apply any existing link prediction model to recover links in the graph, and then perform a ranking algorithm such as PageRank on the recovered graph. There are at least three drawbacks to such strategy. First, link prediction algorithms produce a likelihood (or score) representing how likely two nodes are indeed linked. A threshold needs to be defined to make binary decision about the existence of links before sending the updated graph for the second stage. Such a threshold is hard to define in an unsupervised scenario. Second, in the first stage, the link prediction algorithm is not aware of the ultimate goal of node ranking, thus it is not tailored to optimize for such a goal. Third, such two-stage model can potentially suffer from error propagation: the error of link prediction propagates to the ranking stage and yields inferior result, and there lacks a feedback mechanism to adjust the model from the first stage. This paper proposes a joint learning model for node ranking and link prediction to address above concerns.

Utilizing node attributes provides more benefit than predicting missing links, which brings up the second challenge: how to incorporate node attributes into an unsupervised ranking model. There have been some supervised or semi-supervised ranking models (Bogolubsky et al. 2016; Gao et al. 2011; Zhukovskiy et al. 2014) that perform such task. Those approaches use training data to learn the importance of different attributes toward ranking. The only unsupervised solution we have identified is AttriRank (Hsu

[1] https://wsdmcupchallenge.azurewebsites.net/.

[2] https://kddcup2016.azurewebsites.net/.

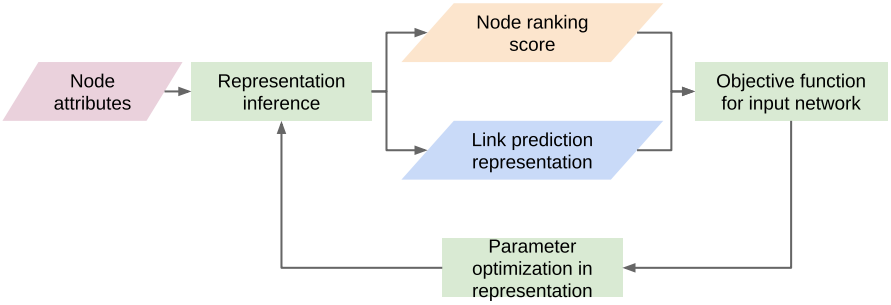

Fig. 1 DeepRank: joint framework of node ranking and link prediction

et al. 2017), which assumes the attributes are equally important as opposed to our model that learns the importance of the attributes. Furthermore, none of the above models consider the missing link information in a graph.

In this paper, we propose a model to jointly perform node ranking and link prediction, named DeepRank, to exploit relationships between network structures and node attributes. We cast the unsupervised ranking task into an optimization task. First, we build a Siamese deep neural network (Bromley et al. 1993) structure as an inference procedure to infer latent representation of nodes based on the correlations between node attributes. The representation leads to a 1-dimensional node ranking score and a multi-dimensional vector for link prediction. Different from a typical DNN model that trains on labeled data with supervised learning, or an auto-encoder that tries to recover the original data, the parameters of our deep learning structure are tuned based on three carefully designed objectives. The first objective is modified from that of PageRank, requesting source node to *propagate* its importance to the target node. The second objective regularizes the ranking scores to avoid overfitting. Finally, a link-prediction objective is realized through a factorization model. Figure 1 summarizes the model.

2 Related work

Ranking nodes in a network has been an active research topic for decades. In the earlier era, Freeman (1978) defines closeness and betweenness centrality evaluated by shortest paths. Later, HITS (Kleinberg 1999) identifies nodes with both authorities and hubs, transmitting ranking scores to each other until convergence. PageRank (Page et al. 1999) proposes a random-walk-based method to rank nodes by evaluating the probabilities of a random walker visiting nodes. However, the above models do not consider other available information such as attributes of nodes.

There are works including Adaptive PageRank (Tsoi et al. 2003) and LiftHITS (Chang et al. 2000) studying how to introduce labels to node ranking. These methods can be regarded as special cases of Semi-Supervised PageRank (SSP) as confirmed in Gao et al. (2011). Considering node attributes, edge attributes, and labels, SSP forms a PageRank-inspired objective function where attributes are weighted over the transition and reset probabilities of PageRank. Furthermore, Supervised Nested PageRank (SNP)

Table 1 Comparisons of DeepRank and representative related works: PageRank (Page et al. 1999), SSP (Gao et al. 2011), SNP (Zhukovskiy et al. 2014) and AttriRank (Hsu et al. 2017)

Model	Input data		Model design		
	Network	Attributes	Learning	Weight range	Link prediction
PageRank	✓		UL		
SSP	✓	✓	SSL	$[0, \infty)$	
SNP	✓	✓	SL	$[0, \infty)$	
AttriRank	✓	✓	UL	All equal	
DeepRank	✓	✓	UL	$\mathbb{R}$	✓

SL supervised learning, *SSL* semi-supervised learning, *UL* unsupervised learning

(Zhukovskiy et al. 2014) improves SSP by setting weights on both attributes and neighbor influence for each node. These models, however, demands labeled data to train. The latest supervised or semi-supervised ranking models all confine weights and attributes to be non-negative to construct valid probability distributions inside PageRank-derived objective functions. In contrast, DeepRank does not demand labeled data nor assumes non-negative hidden-layer weights or input-layer attributes.

AttriRank (Hsu et al. 2017) is arguably the state-of-the-art unsupervised graph ranking with node attributes. The authors declare two model assumptions about PageRank and node attributes. Then a random walk model is proposed realizing these two assumptions. AttriRank, however, possesses an unrealistic equal-weight assumption for all node attributes in its model, which is not the case in DeepRank. Instead of assuming each attribute is equally important, DeepRank learns the weights of each attribute to optimize the designed objective through multi-layer neural networks. Finally, AttriRank does not consider the missing links, but DeepRank does through jointly performing link prediction.

Link prediction asks whether two non-adjacent nodes could be implicitly connected given present network structure. Early works examine several metrics such as common neighbors (Newman 2001) to evaluate the likelihood of link existence. Then parameterized models are proposed to model more information in a complex network. For example, Backstrom and Leskovec (2011) and Lichtenwalter et al. (2010) present PageRank-like random walk methods; Menon and Elkan (2011) applies Matrix Factorization (MF) that performs well in recommender systems; Zhai and Zhang (2015) combines MF and auto-encoders to boost prediction performance. We refer readers to Liben-Nowell and Kleinberg (2007), Lü and Zhou (2011), Wang et al. (2015), Martínez et al. (2016) to review several link prediction proposals. MF-based link prediction is chosen to be integrated into DeepRank since it can seamlessly combine with our DNN structure to be trained efficiently. Table 1 summarizes the features of DeepRank and some major previous works.

Recently a network embedding work PRUNE (Lai et al. 2017) is proposed whose objective resembles that of DeepRank. Nevertheless, there are two major differences between PRUNE and DeepRank. First, the aims of these two works are distinct. PRUNE is proposed to learn an embedding vector for each node, given a single graph

without attributes. On the other hand, DeepRank improves the quality of unsupervised node ranking, given a graph with an external attribute vector for each node. Second, there are several differences between the objectives of PRUNE and DeepRank. Although both uses matrix tri-factorization, PRUNE derives the objective from proximity and community preservation. In contrast, DeepRank views it as a collaborative filtering method. For PageRank-related objective, PRUNE derives an upper-bound by simply applying Cauchy–Schwarz inequality to the PageRank recursive formulation. DeepRank, however, demonstrates a loose extension of two critical parts of PageRank, including the recursive form and the constant-sum constraint. The potential advantage of DeepRank over PRUNE is the consideration of negative edge examples. PRUNE, on the other hand, assumes certain link distribution in a graph. We believe that sampling negative examples in DeepRank can capture real link distributions, and therefore DeepRank allows to sensitively rank numerous nodes. Empirical experiments illustrate the superior performance of DeepRank.

3 DeepRank architecture

3.1 Problem definition

Suppose that we are given a directed graph $G = (V, E)$ where V denotes a set of N nodes and E a set of M directed links with different weights. Each node i contains a K-dimensional attribute vector $\boldsymbol{x}_i \in \mathbb{R}^K$ that encodes information about the node itself. Our goal is to model an unsupervised ranking process that, given the input information, provides a ranking score $\pi_i \geq 0$ for each node that matches best to the true ranking without supervised ranking labels. Algorithms such as PageRank have the same goal but do not consider missing links.

Input network G is described by an adjacency matrix $\boldsymbol{A}_{N\times N}$ where each entry a_{ij} denotes the link weight of a node pair (i, j). For the ease of explanation, in the following sections we consider only binary links, i.e., $a_{ij} \in \{1, 0\}$; however, all the derivations can be straightforwardly extended to the case of non-negative link weights $a_{ij} \in [0, \infty)$. Specifically, we see all the observed links $(i, j) \in E$ from node i to j as positive instances $a_{ij} = 1$ for link prediction. The mission of link prediction is to infer the true values of missing entries a_{ij}. For ease of reference, all the commonly used notations are listed in "Appendix A".

3.2 Overview

Figure 2 illustrates our multi-task deep neural network structure. Nodes i and j each has its own input and output layer, while sharing the same hidden layers for learning latent representation. With various parameters in hidden layers, the input for node i determines the output for node i, while the input of node j influences the output of node j. Such structure is called a Siamese neural network (Bromley et al. 1993) that is used to learn the relationship between a pair of objects (as inputs), for example, an edge (a pair of nodes) in DeepRank. We regard a node pair $(i, j) \in E \cup F$ as a data instance. F

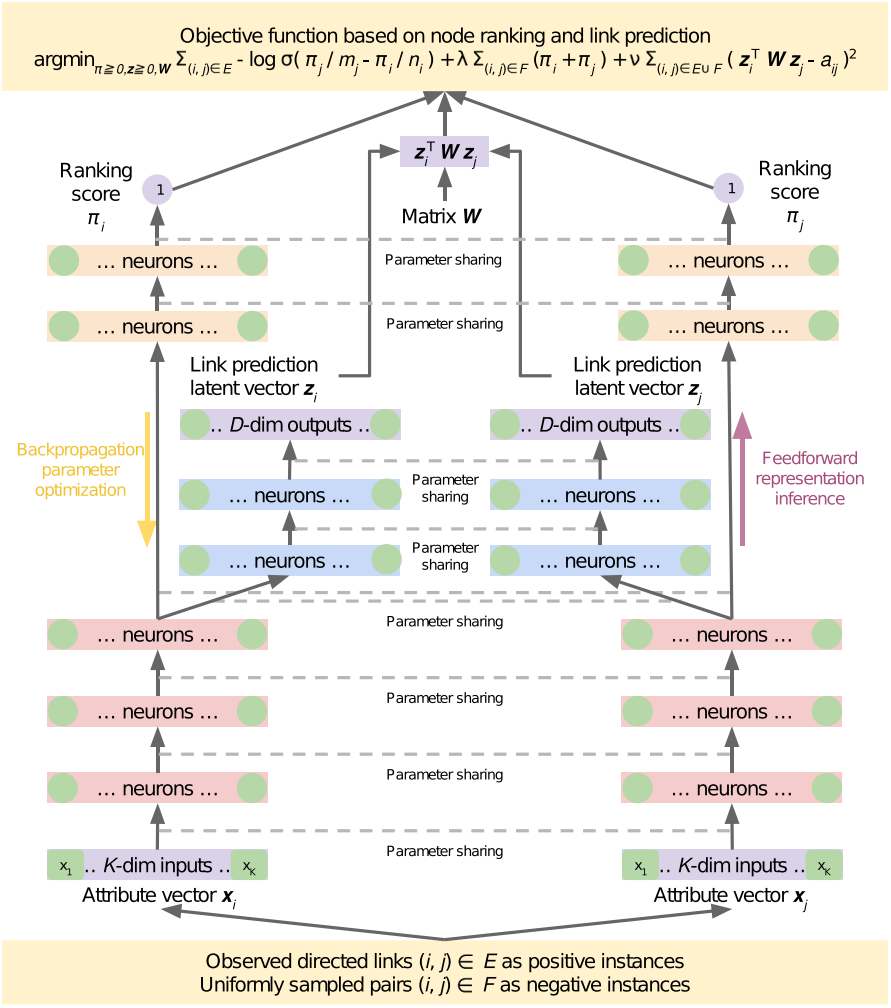

Fig. 2 Our DeepRank model. Each neuron denotes a non-linear function and each solid arrow denotes a weight parameter

denotes the set of sampled node pairs $(i, j) \in F = \{(i, j) \mid a_{ij} = 0\} \subseteq (V \times V) \backslash E$ as the negative training examples of DeepRank, compared with positive edge examples $E = \{(i, j) \mid a_{ij} = 1\}$. In our experiments we set the number of sampled negative examples $|F| = |E| = M$. The generation and analysis of F is presented in Sect. 4.2. Also, in Sect. 5.2.4 we show that $|F| = M$ is a proper choice of negative example size. In DeepRank, we learn the representation of node ranking scores for link prediction. Therefore, the objective function directly reflects the relationship between the learned properties of node pairs. Also, by learning parameters in the shared hidden layers, the implicit dependency of two different objectives (e.g., node ranking and link prediction in DeepRank) can be captured in this neural network. In particular, missing links are

predicted by our neural network, using the link prediction part in our objective function [see (3)]. As shown in Fig. 2, since hidden layers closer to the input layer are shared for both the node ranking and the link prediction objectives, the missing links serve as an implicit regularization in the DeepRank neural network. Not only does the model design prevent node ranking from totally depending on observed links like PageRank, but also alleviate error propagation from link prediction to node ranking because of the joint learning structure.

Initially, the input layer accepts the K-dimensional attribute vectors of node i and j. Above the input layer, we put several hidden layers, which gradually learns different resolutions of attribute correlations. The neurons are fully connected between any two adjacent layers. Each neuron k in the hidden layer l represents a non-linear function h,

$$y_k^{(l)} = h_k^{(l)}(\boldsymbol{y}^{(l-1)}) = \phi(\boldsymbol{\omega}_k^{(l)\top}\boldsymbol{y}^{(l-1)} + b_k^{(l)}), \tag{1}$$

where vector $\boldsymbol{\omega}_k^{(l)}$ and scalar $b_k^{(l)}$ are linear model parameters for input vector $\boldsymbol{y}^{(l-1)}$ from layer $l-1$. Activation function ϕ provides non-linearity. In the higher layers, we further separate neurons into two disjoint parts as a multitask framework. One learns the node ranking value and the other learns the latent vectors for link prediction. That is, one output is a single neuron that represents our target ranking scores π_i; the other output layer keeps a D-dimensional vector $\boldsymbol{z}_i$ to be used for auxiliary link prediction. Mathematically the L-layered neural network refers to the following function composition,

$$\begin{aligned}(\pi_i, \boldsymbol{z}_i) &= f(\boldsymbol{x}_i) = h^{(L)}(\ldots h^{(2)}(h^{(1)}(\boldsymbol{x}_i))),\\ (\pi_j, \boldsymbol{z}_j) &= f(\boldsymbol{x}_j) = h^{(L)}(\ldots h^{(2)}(h^{(1)}(\boldsymbol{x}_j))).\end{aligned} \tag{2}$$

The weight parameters in this neural network reflect our representation inference in Fig. 1. If all the parameters $\boldsymbol{\omega}$ and $\boldsymbol{b}$ are given, then for any node, Eq. (2) can transform its attribute vector $\boldsymbol{x}$ to its representation form $(\pi, \boldsymbol{z})$. This way, node ranking and link prediction are jointly modeled because they share the same deep representation of nodes as inputs.

Since we are facing an unsupervised learning task, model training has to be based on certain criteria, instead of fitting a set of ground truth. Using the representations of node pairs (i, j), we have to define an objective function to train the parameters in the model. Our objective looks like

$$\begin{aligned}\operatorname{argmin}_{\pi\geq 0, z\geq 0, \boldsymbol{W}} &\underbrace{\sum_{(i,j)\in E} -\log\sigma\left(\frac{\pi_j}{m_j} - \frac{\pi_i}{n_i}\right) + \lambda \sum_{(i,j)\in F}(\pi_i + \pi_j)}_{\text{Node ranking}}\\ &+ \underbrace{\nu \sum_{(i,j)\in E\cup F}\left(\boldsymbol{z}_i^\top \boldsymbol{W}\boldsymbol{z}_j - a_{ij}\right)^2}_{\text{Link prediction}},\end{aligned} \tag{3}$$

where parameters π and $\boldsymbol{z}$ are the neural-network function of corresponding node attributes $\boldsymbol{x}$, as shown in (2). Optimizing π and $\boldsymbol{z}$ is equivalent to optimizing all the neuron parameters in the neural network. $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, n_i is the out-degree of node i, m_j is the in-degree of node j, and parameters λ, ν control the weights of their corresponding terms. Note that for positive node pairs $(i, j) \in E$, they are involved in only the optimization of the first and the third terms of (3). On the other hand, negative node pairs $(i, j) \in F$ optimize the last two terms of (3).

The first two terms in (3) are related to the node ranking task and the last one is related to link prediction, which suggests that we should raise the likelihood of $\frac{\pi_j}{m_j}$ greater than $\frac{\pi_i}{n_i}$ for any observed links from i to j (i.e., $(i, j) \in E$). The second term indicates non-existing link samples F, implying that for lower-degree nodes their ranking shall be lower. We will show that optimizing the first two terms is highly correlated, though not equivalent, to the execution of PageRank updates. As will be described later, the final term performs link prediction, which adds a matrix $\boldsymbol{W} \in \mathbb{R}^{D\times D}$ to connect the representations of a node pair. It means that the discovery of a link a_{ij} indeed depends on several latent factors. Representation vector $\boldsymbol{z}$ records the distribution of a node influenced by these factors. Matrix $\boldsymbol{W}$ denotes the positive or negative relationships between two clusters. Although (3) contains two constraints $\pi \geq 0$ and $\boldsymbol{z} \geq 0$, both can be satisfied as long as we choose an activation function whose output is non-negative [for example, Rectified Linear Unit (ReLU) or softplus (Glorot et al. 2011)] at the output layer. All the parameters in the hidden layers need not to be non-negative. We will provide theoretical justification of Eq. (3) in Sect. 4.

We exploit (3) to obtain the optimized parameters of the deep neural network. Since DeepRank is built on deep neural networks, the optimization task is done through general neural network learning process, in particular, Stochastic Gradient Descent (SGD) is exploited. The only constraint is the non-negative latent representation in the network. Thus, advanced neural network techniques like Adam (Kingma and Ba 2015) and Dropout (Srivastava et al. 2014) can be applied straightforwardly.

3.3 Complexity analysis

We present DeepRank pseudocode in "Appendix B" to analyze its complexity. In terms of space, we have $|E| = M$ positive instances, $|F| = M$ sampled negative instances (see Sect. 4.2 for details of our negative sampling), and each of the N nodes has K-dimensional attributes. In addition, a single instance needs to optimize at most Ω^2 parameters between two adjacent layers, in which Ω is the maximum number of neurons in a layer. Besides the neural network, $D \times D$ matrix $\boldsymbol{W}$ is incorporated into the link prediction objective function. Hence if DeepRank is modeled by L hidden layers, then the neural network occupies overall $O(L\Omega^2 + D^2)$ space for all the parameters. Overall, the space complexity is $O(M + NK + L\Omega^2 + D^2)$. However, in practice $N + NK \gg L\Omega^2 + D^2$ since the size of a real-world graph is often dominant. In terms of time, we have mentioned that there are total $O(L\Omega^2 + D^2)$ parameters in DeepRank, so every epoch in SGD optimization takes $O(M(L\Omega^2 + D^2))$ time. The scalability is confirmed as both space and time complexities are linear to the number of links M and nodes N.

4 DeepRank objective analysis

Here we present the theoretical analysis for the DeepRank objective function (3). As Stochastic Gradient Descent (SGD) based solution has become popular for DNN optimization, we would like to show the designed objective function can be optimized by SGD. We also intend to provide theoretical connections of our objective to the widely used PageRank and matrix factorization.

4.1 The analysis of ranking objectives

The first term in (3) is designed to rank nodes. We analyze the similarity and difference from a general objective of PageRank. Starting from transforming the PageRank process into an optimization task: Given the network structure, PageRank reaches equality as its Markov chain converges,

$$\pi_j = \sum_{i \in P_j} \frac{\pi_i}{n_i} \text{ subject to } \sum_{j \in V} \pi_j = 1, \tag{4}$$

where P_j is the set of direct predecessors of node j. Note that for simplicity we do not consider the damping factor here, but the derivations shall remain unchanged with it. Equation (4) can be reformulated as an optimization task as follows,

$$\text{argmin}_{\pi \geq 0} \sum_{j \in V} \left(\sum_{i \in P_j} \frac{\pi_i}{n_i} - \pi_j \right)^2 + \lambda \left(\sum_{j \in V} \pi_j - s \right)^2, \tag{5}$$

where $\lambda > 0, s > 0$ to avoid the trivial solution $\pi_j = 0 \forall j$. Classical PageRank defines $s = 1$ so the ranking score π has a probability interpretation. We utilize a regularization term (i.e., soft constraint) that controls the sum without designing a specific neural network to handle hard constraints. We prefer an unconstrained objective function to maintain the flexibility of extending DNN designs in DeepRank; for example, applying the latest activation functions, using convolutional input layers for specific node attributes, and so on.

The choice of parameters λ and s may not be trivial. We have found that in practice if the input network contains a large number of nodes, then assigning s to 1 or a smaller number will lead to inferior optimization results as the ranking scores tend to approach zero. It is thus difficult for SGD to adjust its learning rate in such a small range. On the other hand, despite the soft constraint, value of λ being too large will prevent SGD from updating any ranking score because even a slight change of ranking scores would violate the constraint. Our goal is to manipulate the optimization equation, so we do not need to tune λ and s altogether. Thus, we conduct a series of relaxation steps from Eqs. (5) to (10), to obtain an objective function more suitable for SGD optimization.

As our first relaxation step, we choose to minimize the upper bound of (5):

Lemma 1 *An upper bound of* (5) *can be derived as,*

$$\text{argmin}_{\pi \geq 0} \sum_{j \in V} \left(\sum_{i \in P_j} \frac{\pi_i}{n_i} - \pi_j + \lambda s \right)^2 + \lambda \left(\sum_{j \in V} \pi_j \right)^2 + \lambda s^2 - \sum_{j \in V} \lambda^2 s^2, \quad (6)$$

and the difference between (5) *and* (6) *is* $\Delta = 2\lambda s \sum_{j \in V} \pi_j \; \forall \pi_j$.

Proof Please refer to "Appendix C".

That says, the minimum of (5) must be bounded by the minimum of (6). The minimization of the second term of (6) implies to shrink the bound difference Δ. It supports the necessity of designing regularization in our objective function: approaching to the original PageRank objective (5). The last two terms are constant such that they do not influence the optimization of π. We leave the discussion of the ranking score regularization [the second term in (6)] to Sect. 4.2. The major concern here is that we have a combination of hyper-parameters $\lambda s \geq 0$ which cannot be tuned easily without labeled data. Here we exploit an approximation to use an inequality as replacement: With $\lambda s \geq 0$ and the second term in (6) suggests π_j cannot be too large, we choose to satisfy another objective,

$$\pi_j \geq \sum_{i \in P_j} \frac{\pi_i}{n_i}, \quad (7)$$

which guarantees that the first term in (6) is small. (7) reflects an intuition for node ranking: score $\boldsymbol{\pi}_j$ of a successor node j is supposed to be larger than the sum of outdegree-weighted scores $\frac{\boldsymbol{\pi}_i}{n_i}$ of predecessor node i. By doing such we can essentially eliminate the need to tune one hyper-parameter s. We choose the maximum likelihood estimation to re-formulate the objective function representing (7).

$$\text{argmax}_{\pi \geq 0} \prod_{j \in V} \Pr \left(\pi_j \geq \sum_{i \in P_j} \frac{\pi_i}{n_i} \right) = \text{argmax}_{\pi \geq 0} \prod_{j \in V} \sigma \left(\pi_j - \sum_{i \in P_j} \frac{\pi_i}{n_i} \right), \quad (8)$$

where we assume that the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$ models the expression of probabilities. Unfortunately, the negative logarithm of (8) cannot be tackled by SGD since it requires an objective function to be the sum of sub-objective functions, each of which is relative to a training instance (i, j). We then propose an alternative instance-based objective:

$$\frac{\pi_j}{m_j} \geq \frac{\pi_i}{n_i} \; \forall \; (i, j) \in E. \quad (9)$$

By the following lemma, we show that (9) is a sufficient condition of the original objective (7).

Lemma 2 *If objective* (9) *is satisfied, then objective* (7) *must be satisfied.*

Proof Please refer to "Appendix D".

Therefore, we choose to maximize the likelihood of (9) that can also satisfy the original objective (7),

$$\text{argmax}_{\pi \geq 0} \prod_{(i,j) \in E} \Pr\left(\frac{\pi_j}{m_j} \geq \frac{\pi_i}{n_i}\right) = \text{argmax}_{\pi \geq 0} \prod_{(i,j) \in E} \sigma\left(\frac{\pi_j}{m_j} - \frac{\pi_i}{n_i}\right). \tag{10}$$

We minimize the negative logarithm of (10), as shown in the first term of (3). The above derivation shows that the DeepRank inequality (7) is a relaxation of PageRank equality (4). It inherits the spirit of the ranking propagation in PageRank and in the meantime making some adjustment to not only reduce the number of hyper-parameters but also allow the objective to be optimized by SGD.

4.2 Regularizing low-degree instances

As implied in the second term of (6), a regularization over the ranking scores is required for a tighter upper bound (6) of the original PageRank objective (5). Here the second term in (3), shown as (11), is designed for similar purpose with some modification,

$$\sum_{(i,j) \in F} (\pi_i + \pi_j). \tag{11}$$

Instead of treating all nodes equivalently in regularization as suggested in (6), here we propose to impose stronger regularization on low-degree nodes as they are supposed to obtain lower ranking values in general. We first resort to *negative sampling* to randomly sample a set of non-existing links, F. Refer to "Appendix B" for the pseudo code. Our regularization term suggests if a training instance (i, j) is labeled negative, then the ranking scores (π_i, π_j) on both nodes should be minimized. Here we prove that the simple formulation is equivalent to a weighted regularization of $\pi_i, \forall i \in V$, where lower-degree nodes are penalized with higher weights. Specifically, for each $(i, j) \in F$, the probabilities of i and j are sampled as defined below,

$$\Pr(\text{Sampling } i \text{ as the source node}) = \frac{|V \backslash S_i|}{|V \times V \backslash E|} = \frac{N - n_i}{N^2 - M},$$
$$\Pr(\text{Sampling } j \text{ as the target node}) = \frac{|V \backslash P_j|}{|V \times V \backslash E|} = \frac{N - m_j}{N^2 - M},$$

where S_i is the set of direct successors of node i. For $|F| = M$ negative examples and with respect to probability, our regularization term (11) can be written as the expected value

$$\begin{aligned}
&\operatorname{argmin}_{\pi \geq 0} M \sum_{i \in V} \sum_{j \in V} \frac{N - n_i}{N^2 - M} \frac{N - m_j}{N^2 - M} (\pi_i + \pi_j) \\
&= \operatorname{argmin}_{\pi \geq 0} \frac{M}{N^2 - M} \left(\sum_{i \in V} (N - n_i) \pi_i + \sum_{j \in V} (N - m_j) \pi_j \right) \\
&= \operatorname{argmin}_{\pi \geq 0} \frac{M}{N^2 - M} \sum_{i \in V} \underbrace{(2N - n_i - m_i)\, \pi_i}_{\text{Merge indices } i, j}. \qquad (12)
\end{aligned}$$

For each (π_i, π_j), it is weighted by the regularization term $N - n_i$ or $N - m_j$. In other words, by (12), nodes with small in-degree or out-degree are heavily regularized in DeepRank. We regard it as reasonable since these nodes reveal less information in the input network.

4.3 Link prediction as tri-factorization

The discovery of a link a_{ij} can be performed by collaborative filtering. In other words, we can infer whether two nodes i, j are adjacent in the future, by using other nodes that have similar neighbor sets to i and j. We adopt matrix factorization, a popular collaborative filtering method, in our DeepRank objective. The method not only can incorporate node attributes like what Stern et al. (2009) does, but also is compatible with current neural networks because SGD optimization can be applied. The third term in (3) implements link prediction based on matrix tri-factorization $\boldsymbol{A} \approx \boldsymbol{Z}^\top \boldsymbol{W} \boldsymbol{Z}$. The main reason to abandon the commonly used $\boldsymbol{A} \approx \boldsymbol{P}^\top \boldsymbol{Q}$ formula is that by doing such we need two latent vectors for each node, one for incoming and one for outgoing scenarios, which inevitably doubles the parameter size in DNN.

Here we provide a mathematical justification on why it is an adequate choice based on the concept of clustering. Suppose that the existence of a link is determined by the clustering distribution of its source and target nodes. Each node is associated with a distribution of belongingness to these D clusters. Then link a_{ij} can be expressed as the expected value $\mathbb{E}$,

$$\begin{aligned}
a_{ij} &= \sum_{c=1}^{D} \sum_{d=1}^{D} \Pr(i \in C_c, j \in C_d) w_{cd} \\
&= \sum_{c=1}^{D} \sum_{d=1}^{D} \Pr(i \in C_c) \Pr(j \in C_d) w_{cd} \\
&= \mathbb{E}_C^{(i,j)}(\boldsymbol{W}(C)), \qquad (13)
\end{aligned}$$

where C_d is the set of nodes in cluster d and w_{cd} denotes the adjacency tendency from cluster c to d. For example, a group of students is associated with different clubs which affect their friendship connections. If club c cooperates with another club d, then w_{cd} is highly positive; on the other hand, $w_{cd} < 0$ for two rival clubs. Here we assume the association of students to clubs are independent. Let $z_i \geq 0$ be the cluster

probability distribution of node i, and then we can rewrite (13) as

$$\sum_{c=1}^{D}\sum_{d=1}^{D} z_{ic} z_{jd} w_{cd} = \boldsymbol{z}_i^{\top} \boldsymbol{W} \boldsymbol{z}_j. \tag{14}$$

Recall that vector $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$ is learned from our neural networks. Theoretically the softmax function $\forall 1 \leq d \leq D, \phi(\boldsymbol{x})_d = \frac{\exp x_d}{\sum_{d'=1}^{D} \exp x_{d'}}$ is the natural choice of activation function at the output layer, since it can represent a distribution of D clusters. However, in real-world scenario, $\boldsymbol{z}_i$ is likely to be sparse (e.g. students only participate in a small fraction of clubs) which is difficult to model in a normal softmax function. Here we take sparsity-induced Rectified Linear Unit (ReLU), $\phi(x) = \max\{0, x\}$ (Glorot et al. 2011), for each dimension of vector $\boldsymbol{z}$ to produce sparse $\boldsymbol{z}_i$. We also notice a recently proposed sparsemax function (Martins and Astudillo 2016) that introduces sparsity to the softmax function, and will experiment with it in future work.

5 Experiment

5.1 Setup

5.1.1 Datasets

We prepare three datasets with different ranking applications to verify our proposal. (I) *Hep-Ph* (Gehrke et al. 2003)[3]: From 1993 to 2003, there are overall 34,546 papers with 421,578 citations. Wang et al. (2013) determines the number of citations after year 2000 as ground-truth labels of paper importance. Following Wang et al. (2013), we split the dataset and feed models with the citation network before 1999. (II) *Webspam*[4]: In 2008, Web Spam Challenge competition asked for designing a ranking model to rank non-spam webpages higher than the spam ones. In this competition, the network entails 114,529 webpages and 1,836,441 links with 122 webpages marked spam and 1933 webpages labeled non-spam. (III) *FB Wall Post* (Viswanath et al. 2009)[5]: Collected from Facebook (FB) in New Orleans in 2009, the network has 63,731 users and 831,401 links where a link represents one user's posting on another's wall. Heidemann et al. (2010) proposes a task to rank active users higher than inactive users where an active user posts at least one article in the next three weeks. There are 14,862 active users and 48,869 inactive ones in the network.

5.1.2 Evaluation

For *Hep-Ph* with real-value labels, following Wang et al. (2013), Spearman's Rank Correlation is selected as evaluation metric. Since labels are binary in *Webspam* and

[3] http://snap.stanford.edu/data/cit-HepPh.html.

[4] http://chato.cl/webspam/datasets/uk2007/.

[5] http://socialnetworks.mpi-sws.org/data-wosn2009.html.

FB Wall Post, the popular Area Under ROC Curve (AUC) is used for evaluation, same as the final evaluation metric in the original Web Spam Competition. Note that AUC expects positive nodes (non-spam webpages or active users) to receive higher rankings than negative ones. Moreover, to evaluate the ability to capture top-ranked nodes, NDCG@100 and Average Precision (AP) are adopted as alternative evaluation metrics. Note that AP is only applied on *Webspam* and *FB Wall Post* which contain binary labels. For the value of k in NDCG@k, small k, $k = 100 \ll N$, is chosen because for many applications only the highest ranked nodes matters, and hence we pay special attention to whether the top-ranked 100 nodes are positively labeled.

5.1.3 Competitors

We compare DeepRank with general-purpose unsupervised node ranking models: (I) *Closeness and betweenness centrality* (Freeman 1978); (II) *PageRank* (Page et al. 1999); (III) *Weighted PageRank (WPR)* (Xing and Ghorbani 2004); (IV) *Semi-Supervised PageRank (SSP)* (Gao et al. 2011) (we use the unsupervised component only); (V) *AttriRank* (Hsu et al. 2017) (state-of-the-art model). Our competitors include global node ranking models using graph topology (i.e. Centrality, PageRank and WPR) and ones that exploit node attributes in graphs (SSP and AttriRank). Note that AttriRank is considered as the state-of-the-art and its superiority over all existing models has demonstrated the usefulness of incorporation of input attributes, and therefore we do not report experiments comparing models with and without input attributes. We briefly introduce these competitors in "Appendix E".

5.1.4 DeepRank details

As shown in Fig. 2, 128, 256 and 512 neurons are used in shared hidden layers and 128 neurons are used for task-specific hidden layers of node ranking and link prediction. In the link prediction component, the dimension of parameters $\boldsymbol{z}$ and $\boldsymbol{W}$ is set to $D = 16$. All hidden layers use Exponential Linear Unit (ELU) (Clevert et al. 2016) as the activation function with 25% dropout probability (Srivastava et al. 2014) for faster and more accurate learning. The non-sparse-induced softplus (Glorot et al. 2011) activation is used as the node ranking output layer to avoid zero-score nodes. The Rectified Linear Unit (ReLU) (Glorot et al. 2011) is selected at the output layer for link prediction, as explained in Sect. 4.3. Adam (Kingma and Ba 2015) is applied for Stochastic Gradient Descent (SGD) with a batch size of 512. The convergence is reached when the objective values change less than 0.1% between epochs. The convergence takes no more than 70 epochs in our experiments. There are two pre-defined hyperparameters, λ and ν, in DeepRank objective (3). We simply fix $(\lambda, \nu) = (0.5, 10.0)$ for all experiments, and we find that DeepRank consistently outperforms all the baseline models using this setting. The theoretical justification of why this simple setting works well is left as future work. In terms of efficiency, DeepRank completes training in around 20 min on the largest *Webspam* dataset with more than 100,000 nodes and 1.8 million links using Nvidia Tesla K80 GPU.

5.1.5 Experiment settings

For the parameters of other models, we fix $d = 0.85$ for PageRank, SSP and WPR and let d follow the beta distribution Beta($\alpha = 2, \beta = 3$) for AttriRank with radial basis function kernel as the original papers suggested. From Web Spam Challenge official website, 138 transformed link-based as well as 96 content-based attributes are available and shown to be useful from participants' reports. For *HepPh* and *FB Wall Post* which have no external attributes, we then refer to the settings in Hsu et al. (2017): For each node, we generate 13-dimensional attribute vector directly from the graph[6] generated from input networks. The competitors SSP and AttriRank share the same attributes. We show that even though an input network contains no external node attributes, DeepRank is still useful given internal attributes of the input network. For models that entail random initializations, we repeat each experiment five times and use one-sample t-test to compare with models producing deterministic rankings. To compare two models that both need random initializations, Welch's t-test is adopted for statistical testing.

5.2 Results

The goal of our experiments is to verify the following hypotheses.

5.2.1 Does DeepRank outperform the competitors in terms of node ranking on different evaluation metrics?

The results in Table 2 show that DeepRank significantly outperforms the other competitors across different applications with various evaluation metrics, with 2nd-best model being AttriRank (p value < 0.01). The results show that comparing to other attribute-aware methods such as SSP and AttriRank, DeepRank does exploit the attributes more effectively with the help of DNN and link prediction. Actually, AttriRank is specifically designed as an attributed augmented random walk algorithm, as described in Sect. 2. Compared with AttriRank, we confirm that the idea of introducing link discovery is beneficial.

5.2.2 Is our joint learning method better than a two-stage model?

In Sect. 1, we describe the benefits of jointly learning node ranking and link prediction to overcome the drawbacks of a two-stage model. Here we would like to verify such statement empirically. We use matrix factorization as the link prediction component in a two-stage model and PageRank as the ranking model in the second stage. The results in Table 3 show that the joint learning model does outperform the two-stage model significantly.

[6] The attributes are the logarithm of: (1) degree divided by average degree of neighbors; (2) in-degree; (3) out-degree; (4, 5) the sum and mean of in-degrees of direct successors; (6, 7) the sum and mean of out-degree of direct predecessors; (8, 9, 10) the number of successors at distance $k \in \{2, 3, 4\}$; (11, 12, 13) the number of successors at distance k divided by the number of successors at distance $k - 1$.

Table 2 Model performance comparison across datasets

Dataset	Evaluation	Closeness	Betweenness	PageRank	WPR	SSP	AttriRank	DeepRank
Hep-Ph	Rank Corr.	0.286	0.445	0.434	0.406	0.252	0.605	**0.625**†
	NDCG@100	0.341	0.405	0.246	0.352	0.151	0.326	**0.422**†
Webspam	AUC	0.577	0.556	0.553	0.509	0.559	0.666	**0.692**†
	NDCG@100	0.090	0.080	0.132	0.125	0.063	0.118	**0.191**†
	AP	0.057	0.058	0.099	0.090	0.110	0.101	**0.135**†
*FB Wall Post	AUC	0.755	0.765	0.775	0.765	0.786	0.810	**0.833**†
	NDCG@100	0.831	0.865	0.921	0.922	0.282	0.935	**0.943**†
	AP	0.442	0.524	0.529	0.519	0.367	0.561	**0.580**†

Bold values indicate the best performance among the experimented models, in terms of the evaluation metric
†Outperforms 2nd-best with p value < 0.01)

Table 3 Performance comparison with two-stage model (T.S.), DeepRank without link prediction ($\nu = 0$), and DeepRank with PRUNE objective (D+P)

Dataset	Evaluation	T.S.	$\nu = 0$	D+P	DeepRank
Hep-Ph	Rank Corr.	0.258	0.608	0.582	**0.625**
	NDCG@100	0.030	0.351	0.384	**0.422**
Webspam	AUC	0.545	0.582	0.597	**0.692**
	NDCG@100	0.053	0.094	0.126	**0.191**
	AP	0.052	0.067	0.068	**0.135**
FB Wall Post	AUC	0.731	0.796	0.788	**0.833**
	NDCG@100	0.473	0.925	0.820	**0.943**
	AP	0.407	0.556	0.491	**0.580**

Bold values indicate the best performance among the experimented models, in terms of the evaluation metric

5.2.3 Can the link prediction component boost the quality of node ranking in DeepRank?

To justify this hypothesis, we set $\nu = 0$ for DeepRank to drop the influence of link prediction, and then compare the results with the original DeepRank. The experiment results in Table 3 demonstrate that taking link prediction into consideration brings significant improvement. On the other hand, we observe that DeepRank with $\nu = 0$ does not outperform the strongest baseline AttriRank in Table 2. It implies that straightforwardly relying on deep neural network structures cannot achieve start-of-the-art ranking performance. The observation supports our motivation of designing an auxiliary link prediction objective for DeepRank.

5.2.4 Is the performance of DeepRank sensitive to the pre-defined hyper-parameters?

DeepRank objective contains two pre-defined parameters λ and ν, which determine the weights of ranking regularization as well as the link prediction components. We examine performance sensitivity with respect to λ and ν. In Figs. 3, 4 and 5, we vary the values of λ and ν up to 40% from the default setting. The figure shows promising outcome that even though the parameters have certain level of impact on the performance, DeepRank is still better across different setups comparing to the state-of-the-art approach, AttriRank.

On the other hand, in Sect. 4.2, we have stated how to sample a set of negative examples F in DeepRank. Figure 6 illustrates the sensitivity experiments for $|F| = k|E| = kM$ where k is one of hyper-parameters of DeepRank. Overall, DeepRank is more sensitive to k, but $k = 1$ achieves the best performance across all three datasets. We believe that choosing $|F| = M$ is reasonable not only due to the experiment results, but also because balancing the sizes between positive and negative examples can alleviate biased learning in DeepRank. In particular, the first and the second term of DeepRank objective function (3) adopt the set E and F respectively. If the sizes of two sets are highly unbalanced, then one of the two terms dominates the learning of

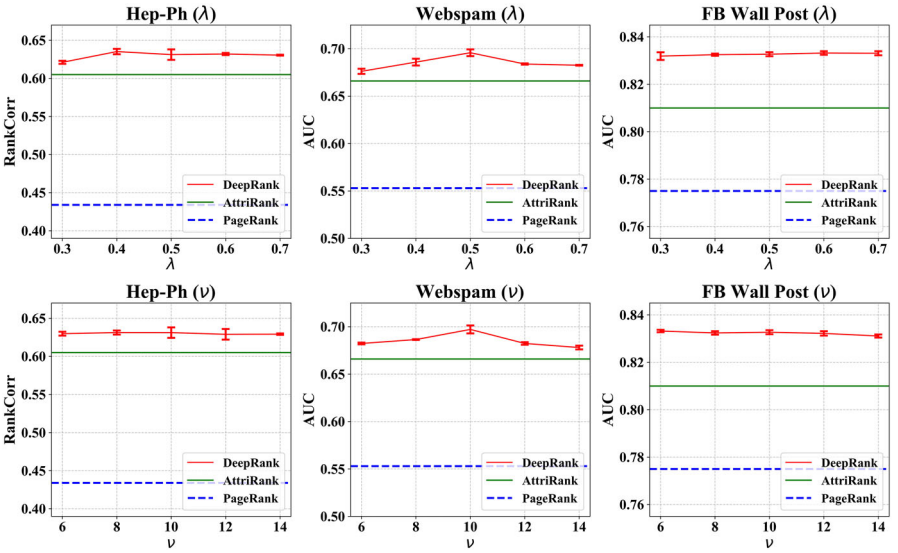

Fig. 3 Spearman's rank correlation coefficient and area under ROC curve (AUC) with respect to parameters λ and ν in its objective function

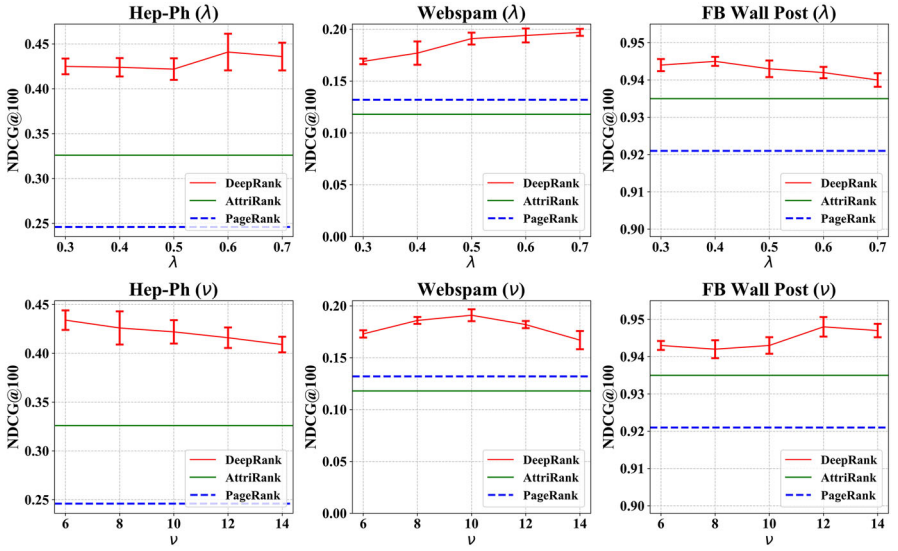

Fig. 4 Normalized discounted cumulative gain (NDCG) with respect to parameters λ and ν in its objective function

neural network parameters, such that DeepRank fails to fit both terms coming from PageRank relaxation (Sect. 4.1).

Note that our negative sampling method has been shown effectiveness in existing collaborative filtering works (He et al. 2017). However different from previous works, the negative sampling in DeepRank can be explained by a weighted regularization

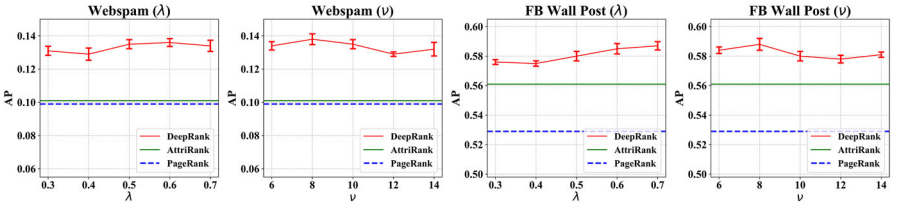

Fig. 5 Average precision (AP) with respect to parameters λ and ν in its objective function

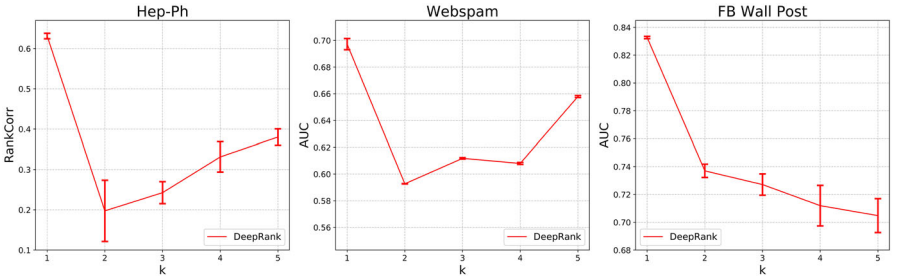

Fig. 6 Spearman's rank correlation coefficient and area under ROC curve (AUC) with respect to parameters k where $|F| = k|E|$

objective in Sect. 4.2. One of our future work is to use and analyze non-uniform negative sampling methods (Gantner et al. 2011; Rendle and Freudenthaler 2014) in our DeepRank.

5.2.5 Is the DeepRank objective function a better choice for node ranking comparing with that of PRUNE?

As commented in Sect. 2, both DeepRank and PRUNE (Lai et al. 2017) consider node ranking scenarios but propose different objective functions. The objective function of PRUNE can be adopted in our model to rank nodes, which allows us to conduct an experiment to compare the effectiveness. We replace the DeepRank objective with that of PRUNE,

$$\begin{aligned}\text{argmin}_{\pi\geq 0, z\geq 0, W\geq 0} & \underbrace{\sum_{(i,j)\in E}\left(z_i^\top W z_j - \max\left\{0, \log\left(\frac{M}{\alpha m_j n_i}\right)\right\}\right)^2}_{\text{replace DeepRank link prediction objective}} \\ + & \underbrace{\lambda \sum_{(i,j)\in E} m_j \left(\frac{\pi_j}{m_j} - \frac{\pi_i}{n_i}\right)^2}_{\text{replace DeepRank node ranking objective}} , \end{aligned} \tag{15}$$

and examine their ranking outputs π with finely tuned hyper-parameters α and λ.

The results are reported in Table 3. Clearly our model significantly outperforms the competitor that uses the PRUNE objective inside DeepRank. Note that the learning-

to-rank experiments in the publication of PRUNE is supervised learning. But in our unsupervised ranking experiments, PRUNE cannot attain state-of-the-art level.

6 Conclusion

We believe the success of DeepRank comes from several aspects. It is apparent that the DNN structure brings a richer representation capability to our model. However, without an adequate objective function and suitable parameter tuning framework, an unsupervised deep learning structure is highly prone to fitting in the wrong direction. Thus, the key really lies in the useful insight behind each of the three objectives, coupled with carefully-trimmed hyper-parameter space and the adjustment on the objective to fit better in the SGD training scheme. Future works include deeper analysis on the activation functions (e.g. Sparsemax Martins and Astudillo 2016), considering convolution input structures for other kinds of attributes such as image, and incorporating additional edge attributes.

Acknowledgements This material is based upon work supported by the Air Force Office of Scientific Research, AOARD under Award Number FA2386-17-1-4038, and Taiwan Ministry of Science and Technology (MOST) under Grant Number 106-2218-E-002-042.

Appendices

Appendix A: Notations

See Table 4.

Table 4 Commonly used notations

Notation	Description
$G = (V, E = \{(i, j)\|a_{ij} = 1\})$	Input network (or graph) with node set V and directed link set E
$\boldsymbol{x}_i \in \mathbb{R}^K$	Attribute vector of node i
$N = \|V\|$	Number of nodes (or vertices)
$M = \|E\|$	Number of links (or edges)
$\boldsymbol{A} = [a_{ij}] \in \{1, 0\}^{N \times N}$	Adjacency matrix of G
$F = \{(i, j)\|a_{ij} = 0\} \subseteq (V \times V) \backslash E$	Set of sampled negative instances
S_i	Set of direct successors of node i
P_i	Set of direct predecessors of node i
$n_i = \|S_i\|$	Out-degree of node i
$m_i = \|P_i\|$	In-degree of node i
$\pi_i \geq 0$	Ranking score of node i
$\boldsymbol{z} \in [0, \infty)^D$	Vector parameters of link prediction
$\boldsymbol{W} \in \mathbb{R}^{D \times D}$	Matrix parameters of link prediction

Appendix B: DeepRank pseudo code

Algorithm 1 DeepRank

Require: Network $G = (V, E)$ where each node $i \in V$ has attribute vector $\boldsymbol{x}_i$, pre-defined parameters λ, ν and the number of neurons in the neural network.
Ensure: Ranking score $\pi_i = f(\boldsymbol{x}_i)$ for each node i.
1: Sample negative instance set F as shown in Algorithm 2.
2: **while** updates not convergent **do**
3: **for** minibatch sampled from $E \cup F$ **do**
4: Update matrix $\boldsymbol{W}$ and all the parameters $\boldsymbol{\omega}$, b at the hidden layers using their minibatch gradients of our objective function.

Algorithm 2 Sampling the set F of M negative instances from population $V \times V \backslash E$ uniformly at random

1: **while** $|F| < M$ **do**
2: Sample a node pair (i, j) from adjacency matrix $\boldsymbol{A}$ uniformly at random.
3: **if** $(i, j) \notin E$ and $(i, j) \notin F$ **then**
4: Add (i, j) to F.

Appendix C: Derivation of an upper bound of PageRank objective function

$$\begin{aligned}
&\sum_{j\in V}\left(\sum_{i\in P_j}\frac{\pi_i}{n_i}-\pi_j\right)^2+\lambda\left(\sum_{j\in V}\pi_j-s\right)^2\\
&=\sum_{j\in V}\left(\sum_{i\in P_j}\frac{\pi_i}{n_i}-\pi_j\right)^2+\lambda\left(\sum_{j\in V}\pi_j\right)^2-2\lambda s\sum_{j\in V}\pi_j+\lambda s^2\\
&\le\sum_{j\in V}\left(\sum_{i\in P_j}\frac{\pi_i}{n_i}-\pi_j\right)^2+2\lambda s\sum_{j\in V}\left(\sum_{i\in P_j}\frac{\pi_i}{n_i}-\pi_j\right)+\lambda\left(\sum_{j\in V}\pi_j\right)^2+\lambda s^2.
\end{aligned} \tag{16}$$

On the right-hand side of the inequality, we add a term $0 \le \Delta = 2\lambda s \sum_{j\in V}\sum_{i\in P_j}\frac{\pi_i}{n_i}$. The inequality holds due to non-negative λ, s, $\pi_i \forall i$ and $n_i \forall i$. Completing the square to the upper bound (16), we have:

$$\sum_{j\in V}\left(\sum_{i\in P_j}\frac{\pi_i}{n_i}-\pi_j+\lambda s\right)^2+\lambda\left(\sum_{j\in V}\pi_j\right)^2+\lambda s^2-\sum_{j\in V}\lambda^2 s^2. \tag{17}$$

Definitely Δ is the difference between the upper bound and the original objective function. For ease of presentation, we use matrix representations to derive Δ. Let vector $\boldsymbol{\pi} \in [0, \infty)^N$ be the ranking score vector for all N nodes, while $\mathbf{1}$ represents a

N-dimensional constant vector of all 1's. Matrix $\boldsymbol{Q} \in [0, \infty)^{N \times N}$ denotes a transition matrix where each entry $q_{ij} = n_i^{-1}$ for row i and column j. By the definition of $\boldsymbol{Q}$, the entry sum of each row of $\boldsymbol{Q}$ is exactly 1, that is, $\boldsymbol{Q}\mathbf{1} = \mathbf{1}$. Having the matrix representations, we derive Δ as follows,

$$\Delta = 2\lambda s \sum_{j \in V} \sum_{i \in P_j} \frac{\pi_i}{n_i} = 2\lambda s \mathbf{1}^\top \left(\boldsymbol{Q}^\top \boldsymbol{\pi} \right) = 2\lambda s \left(\mathbf{1}^\top \boldsymbol{Q}^\top \right) \boldsymbol{\pi} = 2\lambda s \mathbf{1}^\top \boldsymbol{\pi} = 2\lambda s \sum_{j \in V} \pi_j. \tag{18}$$

Appendix D: Proof for the reduction of node ranking

Suppose that for all link $(i, j) \in E$, the inequality $\frac{\pi_j}{m_j} \geq \frac{\pi_i}{n_i}$ holds. Then for any arbitrary node j, $\frac{\pi_j}{m_j} \geq \frac{\pi_i}{n_i}$ for all direct predecessors i of j. That is, $\frac{\pi_j}{m_j}$ must be no less than the average of $\frac{\pi_i}{n_i}$ of all direct predecessors $i \in P_j$. Given $m_j = |P_j|$, we have:

$$\frac{\pi_j}{m_j} \geq \frac{\pi_i}{n_i} \; \forall \, i \in P_j \implies \frac{\pi_j}{m_j} \geq \underbrace{\frac{1}{m_j} \sum_{i \in P_j} \frac{\pi_i}{n_i}}_{\text{Average}} \implies \pi_j \geq \sum_{i \in P_j} \frac{\pi_i}{n_i}.$$

Appendix E: Introduction of competitors

E.0.6 Closeness and betweenness centrality (Freeman 1978)

In social network analysis, centrality methods find the most important nodes based on current network structure. We choose two common centrality definitions in Freeman (1978), closeness and betweenness centralities. Closeness centrality claims that nodes with shorter path length to others are more important. Betweenness centrality claims the more important nodes are part of more shortest paths in the network.

E.0.7 PageRank (Page et al. 1999)

It is a well-known node ranking algorithm without using node attributes. Under Markov Chain framework, we repeatedly update ranking scores using the following rule until convergence,

$$\boldsymbol{\pi}^{(t+1)} = (1 - d) \frac{1}{N} \mathbf{1} + d \, \boldsymbol{Q} \boldsymbol{\pi}^{(t)}, \tag{19}$$

where vector $\boldsymbol{\pi}$ is the ranking score set of all N nodes, $\boldsymbol{Q} = [q_{ij} = \frac{1}{n_j}]$ is the transition matrix, $\mathbf{1}$ is a vector of all 1's, and d is the damping factor normally set to 0.85.

E.0.8 Weighted PageRank (WPR) (Xing and Ghorbani 2004)

This variant of PageRank modifies the transition matrix from uniformly distributed weights to a weight distribution proportional to in-degree and out-degree of pointed nodes. Its update rule is

$$\boldsymbol{\pi}^{(t+1)} = (1-d)\frac{1}{N}\mathbf{1} + d\,\boldsymbol{Q}\boldsymbol{\pi}^{(t)} \text{ and } q_{ij} = \frac{1}{\zeta_j}\frac{m_i}{\sum_{k\in S_j} m_k}\frac{n_i}{\sum_{k\in S_j} n_k}, \tag{20}$$

where $\zeta_j = \sum_i q_{ij}$ for $\boldsymbol{Q} = [q_{ij}]$.

E.0.9 Semi-supervised PageRank (SSP) (Gao et al. 2011)

It is the state-of-the-art semi-supervised solution to node ranking. It is composed of a supervised part and an unsupervised part. We adopt only its unsupervised component with node attributes. The objective function of its unsupervised component is simplified as below,

$$\arg\min_{\boldsymbol{\pi}\geq 0, \boldsymbol{\omega}\geq 0} \|(1-d)\boldsymbol{X}^\top\boldsymbol{\omega} + d\,\boldsymbol{Q}\boldsymbol{\pi} - \boldsymbol{\pi}\|_2^2, \tag{21}$$

where matrix $\boldsymbol{X} = [\boldsymbol{x}_1\boldsymbol{x}_2\ldots\boldsymbol{x}_N]$ denotes the collection of node attributes, $\boldsymbol{Q} = [q_{ij} = \frac{1}{n_j}]$ represents the transition matrix, and $\boldsymbol{\omega}$ refers to the weight vector. Note that weights $\boldsymbol{\omega}$ and attributes $\boldsymbol{X}$ should be non-negative. (21) is optimized using projected gradient descent in Gao et al. (2011).

E.0.10 AttriRank (Hsu et al. 2017)

It is the state-of-the-art unsupervised general approach to node ranking with node attributes. We follow the setup written in the original paper for parameter setting and selection. The update equation is as below,

$$\boldsymbol{\pi}^{(t+1)} = (1-d)\frac{1}{N}\boldsymbol{r} + d\,\boldsymbol{Q}\boldsymbol{\pi}^{(t)} \text{ and } r_i = \frac{1}{\zeta}\sum_{j\in V} e^{-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2^2}, \tag{22}$$

where vector $\boldsymbol{r} = [r_i]$ encodes the information of node attributes using the sum of Radial Basis Function (RBF) kernels, $\zeta = \sum_i r_i$, and γ is the parameter of RBF kernel.

References

Backstrom L, Leskovec J (2011) Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the fourth ACM international conference on web search and data mining (WSDM'11). ACM, New York, NY, USA, pp 635–644. https://doi.org/10.1145/1935826.1935914

Bogolubsky L, Dvurechensky P, Gasnikov A, Gusev G, Nesterov Y, Raigorodskii AM, Tikhonov A, Zhukovskii M (2016) Learning supervised pagerank with gradient-based and gradient-free optimization methods. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds) Advances in neural information processing systems 29. Curran Associates, Inc., pp 4914–4922. http://papers.nips.cc/paper/6565-learning-supervised-pagerank-with-gradient-based-and-gradient-free-optimization-methods.pdf

Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R (1993) Signature verification using a "siamese" time delay neural network. In: Proceedings of the 6th international conference on neural information processing systems (NIPS'93). Morgan Kaufmann, San Francisco, CA, USA, pp 737–744. http://dl.acm.org/citation.cfm?id=2987189.2987282

Chang H, Cohn D, McCallum A (2000) Learning to create customized authority lists. In: Langley P (ed) Proceedings of the seventeenth international conference on machine learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29–July 2, 2000. Morgan Kaufmann, pp 127–134

Clevert D, Unterthiner T, Hochreiter S (2016) Fast and accurate deep network learning by exponential linear units (elus). In: International conference on learning representations arXiv:1511.07289

Freeman LC (1978) Centrality in social networks conceptual clarification. Soc Netw 1(3):215–239. https://doi.org/10.1016/0378-8733(78)90021-7

Gantner Z, Drumond L, Freudenthaler C, Schmidt-Thieme L (2011) Personalized ranking for non-uniformly sampled items. In: Proceedings of the 2011 international conference on KDD Cup 2011—Volume 18, JMLR.org (KDDCUP'11), pp 231–247. http://dl.acm.org/citation.cfm?id=3000375.3000390

Gao B, Liu TY, Wei W, Wang T, Li H (2011) Semi-supervised ranking on very large graphs with rich metadata. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'11). ACM, New York, NY, USA, pp 96–104. https://doi.org/10.1145/2020408.2020430

Gehrke J, Ginsparg P, Kleinberg J (2003) Overview of the 2003 KDD cup. SIGKDD Explor Newsl 5(2):149–151. https://doi.org/10.1145/980972.980992

Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Gordon G, Dunson D, Dudk M (eds) Proceedings of the fourteenth international conference on artificial intelligence and statistics, PMLR, Fort Lauderdale, FL, USA, proceedings of machine learning research, vol 15, pp 315–323. http://proceedings.mlr.press/v15/glorot11a.html

Gyöngyi Z, Garcia-Molina H, Pedersen J (2004) Combating web spam with trustrank. In: Proceedings of the thirtieth international conference on very large data bases—volume 30, VLDB Endowment (VLDB'04), pp 576–587. http://dl.acm.org/citation.cfm?id=1316689.1316740

He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on World Wide Web, International World Wide Web Conferences Steering Committee (WWW'17), Republic and Canton of Geneva, Switzerland, pp 173–182. https://doi.org/10.1145/3038912.3052569

Heidemann J, Klier M, Probst F (2010) Identifying key users in online social networks: a pagerank based approach. In: Sabherwal R, Sumner M (eds) Proceedings of the international conference on information systems (ICIS 2010), Saint Louis, Missouri, USA, December 12–15, 2010. Association for Information Systems, p 79. http://aisel.aisnet.org/icis2010_submissions/79

Hsu CC, Lai YA, Chen WH, Feng MH, Lin SD (2017) Unsupervised ranking using graph structures and node attributes. In: Proceedings of the tenth ACM international conference on web search and data mining (WSDM'17). ACM, New York, NY, USA, pp 771–779. https://doi.org/10.1145/3018661.3018668

Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: International conference on learning representations arXiv:1412.6980

Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. J ACM 46(5):604–632. https://doi.org/10.1145/324133.324140

Lai YA, Hsu CC, Chen WH, Yeh MY, Lin SD (2017) Prune: preserving proximity and global ranking for network embedding. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30. Curran Associates, Inc., pp 5257–5266. http://papers.nips.cc/paper/7110-prune-preserving-proximity-and-global-ranking-for-network-embedding.pdf

Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031. https://doi.org/10.1002/asi.v58:7

Lichtenwalter RN, Lussier JT, Chawla NV (2010) New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, NY, USA (KDD'10), pp 243–252. https://doi.org/10.1145/1835804.1835837

Lü L, Zhou T (2011) Link prediction in complex networks: a survey. Phys A Stat Mech Its Appl 390(6):1150–1170. https://doi.org/10.1016/j.physa.2010.11.027

Martínez V, Berzal F, Cubero JC (2016) A survey of link prediction in complex networks. ACM Comput Surv 49(4):69:1–69:33. https://doi.org/10.1145/3012704

Martins AFT, Astudillo RF (2016) From softmax to sparsemax: a sparse model of attention and multi-label classification. In: Proceedings of the 33rd international conference on international conference on machine learning—volume 48, JMLR.org, ICML'16, pp 1614–1623. http://dl.acm.org/citation.cfm?id=3045390.3045561

Menon AK, Elkan C (2011) Link prediction via matrix factorization. In: Proceedings of the 2011 European conference on machine learning and knowledge discovery in databases—volume part II (ECML PKDD'11). Springer, Berlin, pp 437–452. http://dl.acm.org/citation.cfm?id=2034117.2034146

Newman MEJ (2001) Clustering and preferential attachment in growing networks. Phys Rev E 64:025102. https://doi.org/10.1103/PhysRevE.64.025102

Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the web. Technical Report 1999-66, Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/, previous number = SIDL-WP-1999-0120

Rendle S, Freudenthaler C (2014) Improving pairwise learning for item recommendation from implicit feedback. In: Proceedings of the 7th ACM international conference on web search and data mining (WSDM'14). ACM, New York, NY, USA, pp 273–282. https://doi.org/10.1145/2556195.2556248

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Stern DH, Herbrich R, Graepel T (2009) Matchbox: Large scale online bayesian recommendations. In: Proceedings of the 18th international conference on World Wide Web (WWW'09). ACM, New York, NY, USA, pp 111–120. https://doi.org/10.1145/1526709.1526725

Tsoi AC, Morini G, Scarselli F, Hagenbuchner M, Maggini M (2003) Adaptive ranking of web pages. In: Proceedings of the 12th international conference on World Wide Web (WWW'03). ACM, New York, NY, USA, pp 356–365. https://doi.org/10.1145/775152.775203

Viswanath B, Mislove A, Cha M, Gummadi KP (2009) On the evolution of user interaction in facebook. In: Proceedings of the 2Nd ACM workshop on online social networks (WOSN'09). ACM, New York, NY, USA, pp 37–42. https://doi.org/10.1145/1592665.1592675

Wang Y, Tong Y, Zeng M (2013) Ranking scientific articles by exploiting citations, authors, journals, and time information. In: desJardins M, Littman ML (eds) Proceedings of the twenty-seventh AAAI conference on artificial intelligence, July 14–18, 2013, Bellevue, Washington, USA. AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6363

Wang P, Xu B, Wu Y, Zhou X (2015) Link prediction in social networks: the state-of-the-art. Sci China Inf Sci 58(1):1–38. https://doi.org/10.1007/s11432-014-5237-y

Xing W, Ghorbani A (2004) Weighted pagerank algorithm. In: Proceedings. In: Second annual conference on communication networks and services research, 2004, pp 305–314. https://doi.org/10.1109/DNSR.2004.1344743

Zhai S, Zhang ZM (2015) Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs. In: Venkatasubramanian S, Ye J (eds) Proceedings of the 2015 SIAM international conference on data mining, Vancouver, BC, Canada, April 30–May 2, 2015. SIAM, pp 451–459. https://doi.org/10.1137/1.9781611974010.51

Zhukovskiy M, Gusev G, Serdyukov P (2014) Supervised nested pagerank. In: Proceedings of the 23rd ACM international conference on information and knowledge management (CIKM'14). ACM, New York, NY, USA, pp 1059–1068. https://doi.org/10.1145/2661829.2661969