AFRL-AFOSR-JP-TR-2019-0031

Can discovering boost learning? - Improving the quality of a machine learning model through discovering hidden structure among data

Shou-de Lin
NATIONAL TAIWAN UNIVERSITY
1, ROOSEVELT RD., SEC. 4
TAIPEI CITY, 10617
TW

05/02/2019
Final Report

Air Force Research Laboratory
Air Force Office of Scientific Research
Asian Office of Aerospace Research and Development
Unit 45002, APO AP 96338-5002

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
02-05-2019	Final	02 May 2017 to 01 May 2019

4. TITLE AND SUBTITLE
Can discovering boost learning? - Improving the quality of a machine learning model through discovering hidden structure among data

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA2386-17-1-4038

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Shou-de Lin

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NATIONAL TAIWAN UNIVERSITY
1, ROOSEVELT RD., SEC. 4
TAIPEI CITY, 10617 TW

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AOARD
UNIT 45002
APO AP 96338-5002

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR IOA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-JP-TR-2019-0031

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The PI has successfully completed the research proposed. In this project the PIs studied how the performance of a machine learning model can be boosted through incorporating a discovery engine aiming at finding hidden/missing structure or information. The team used an unsupervised node-ranking model that considers not only the attributes of nodes in a graph but also the incompleteness of the graph structure. They showed that by discovering the structure of networks they were able to perform better node ranking. They then used deep neural network (DNN)(based solution as a ranking model. The rich representation capability of the DNN structure together with a novel design of the discover objectives allow the proposed model to significantly outperform the state-of-the-art ranking solutions. There were 3 peer reviewed publications as a direct result of this grant award.

15. SUBJECT TERMS
Hidden structure, Non-negative probabilistic matrix factorization, Knowledge discovery, Latent variables, Implicit relationships, Stochastic Gradient Descent

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			LIN, ALAN
Unclassified	Unclassified	Unclassified	SAR		19b. TELEPHONE NUMBER *(Include area code)* 315-227-7009

Final Report for AOARD Grant FA2386-17-1-4038

"Can Discovering Boost Learning? Improving the quality of a Machine Learning Model through Discovering Hidden Structure From Data"

4/30/2019

Name of Principal Investigators (PI and Co-PIs): Shou-De Lin

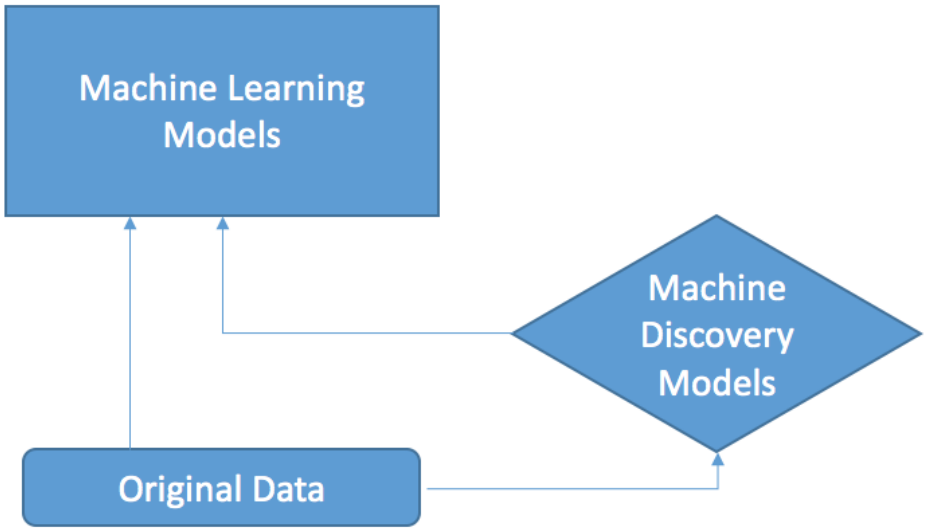
- e-mail address : sdlin@csie.ntu.edu.tw
- Institution : National Taiwan University
- Mailing Address : CSIE Department, National Taiwan University, No. 1, Roosevelt Rd., Sec. 4, Taipei City 10617 Taiwan
- Phone :+ 886233664888
- Fax : N/A

Period of Performance: 5/2/2017– 5/1/2019

Abstract:
In this project we would like to study how the performance of a machine learning model can be boosted through incorporating a discovery engine aiming at finding hidden/missing structure or information. As an example, we demonstrate 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 argue that by discovering the structure of networks we can perform better node ranking. We further exploit deep neural network based solution as our ranking model. The rich representation capability of the DNN structure together with a novel design of the discover objectives allow the proposed model to significantly outperform the state-of-the-art ranking solutions.

Background:

In some real-world tasks, we do not have complete information to generate a high-quality learning model. In this project, we have proposed a framework to utilize the results of discovery to boost the performance of learning. Below is a general structure of our framework. The discover model tries to find some hidden information or identify hidden structure from data, and such information can be used by a learning model to boost the learning capability. Thanks to the powerfulness of deep neural network, now it is possible to perform both discovery and learning tasks in one single framework using neural network and back propagation.
Given such framework, we are allowed to perform learning and discovery altogether to boost the performance of a machine learning model. For instance, we have proposed a solution to boost the quality of recommendation through discovering implicit social information among people (see the listed publication 2 "A General Framework for Implicit and Explicit Social Recommendation."). We have also propose a graph embedding model that relies on the discovery of missing links (see the listed publication 3 "PRUNE: Preserving Proximity and Global Ranking for Network Embedding."). In this report, we will focus on describing a recent work of us that utilizes the discovery model to boost the node-ranking quality.



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 [8], paper impact determination [23], and link prediction [1, 13]. 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] asked the participants to rank the impact of papers and affiliation influence, respectively, in an academic network without providing any labeled training instances.

This work focuses on addressing the following concerns of the representative unsupervised node ranking algorithms, such as PageRank [19] and HITS [11]. First, these approaches generate ranking scores assuming the given network structure is complete. Such assumption might be too strong since in the real-world situation 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 [1, 12–15, 17, 18, 22, 25] 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. Second, most of the existing models consider only network topology but not information associated with the nodes. In real-world networks, however, there are some information or attributes of a node given, like user profiles or website metadata. It is conceivable that the quality of ranking can be boosted by leveraging such information.

A naive two-stage solution to incorporate link prediction with unsupervised node ranking is described here. One can apply an 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 of such strategy. First, link prediction algorithms produce a likelihood (or score) representing how likely two nodes are indeed linked. We need to define a threshold to make binary decision about the existence of links before sending the updated graph for the second stage. Such threshold is hard to define given 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 such goal. Third, such two-stage model can potentially suffer from error propagation: the error of link prediction propagates to the ranking stage to yield inferior result, and there lacks a feedback mechanism to adjust the first-stage model. This work proposes a jointly learning model for node ranking and link prediction to address such 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 [2, 6, 26] performing such technique. Those approaches use training data to learn the importance of different attributes toward ranking. The only unsupervised solution we have identified is AttriRank [10], 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 work, we propose a model to jointly perform both node ranking and link prediction, named DeepRank, to exploit interactions between network structures and node attributes. We cast the unsupervised ranking task into an optimization task. First, we build a deep neural network 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; and finally, a link-prediction objective is realized through a factorization model. Figure 1 summarizes the model.

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

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

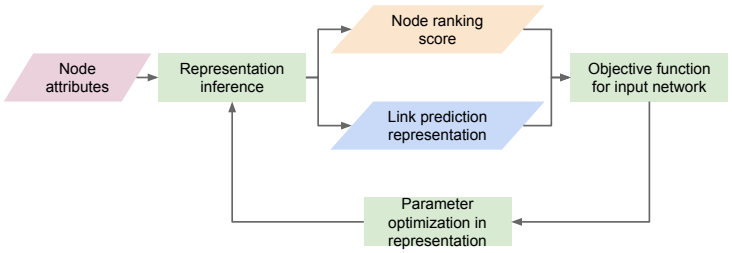


Figure 1: DeepRank: joint framework of node ranking and link prediction.

2 Related work

Ranking nodes in a network have been an active research topic for decades. In the earlier days, [5] defines closeness and betweenness centrality evaluated by shortest paths. Later, HITS [11] identifies nodes with both authorities and hubs, transmitting ranking scores to each other until convergence. PageRank [19] 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 extra available information such as attributes of nodes.

There are works including Adaptive PageRank [21], LiftHITS [3] studying how to introduce labels to node ranking. These methods can be regarded as special cases of Semi-Supervised PageRank (SSP) as confirmed in [6]. 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) [26] improves SSP by setting weights on both attributes and neighbor influence for each node. These models, however, demands label data to train. The latest supervised or semi-supervised ranking models all confine weights and attributes to non-negative to construct valid probability distributions inside PageRank-derived objective functions. On the other hand, DeepRank does not demand labelled data nor assume non-negative hidden-layer weights or input-layer attributes.

AttriRank [10] 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 to all node attributes in its model, which is not the case in DeepRank. Instead of assuming each attribute is equally important, DeepRank learns weights to optimize the designed objective through multi-layer neural networks. Finally, AttriRank does not consider the missing link, while DeepRank does through jointly performing link prediction.

Link prediction asks whether two non-adjacent nodes are connected given present network structure. Earlier researchers examine several metrics such as common neighbors [18] to evaluate the likelihood of link existence. Then parameterized models are proposed to model more information in a complex network. For example, [1] and [13] present PageRank-like random walk methods; [17] applies Matrix Factorization (MF) that is well performed in recommender systems; [25] combines MF and auto-encoders to boost prediction performance. We refer readers to [12, 14, 15, 22] that review various 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.

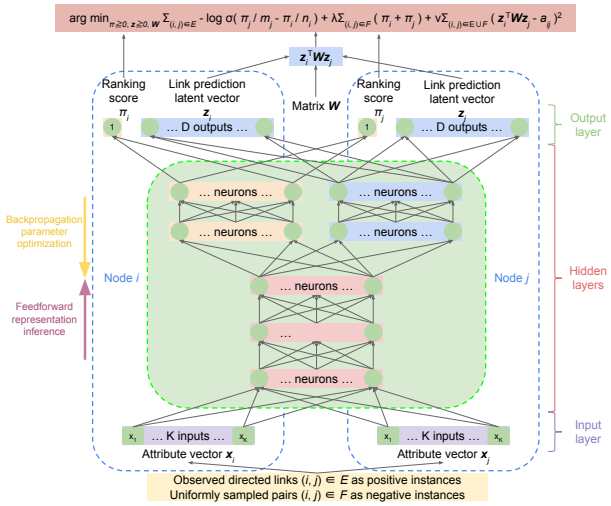


Figure 2: Our DeepRank model. Each neuron denotes a non-linear function and each arrow denotes a weight parameter.

Table 1: Comparisons of DeepRank and representative related works (G: Network; $\boldsymbol{x}$: attributes; SL: supervised learning; SSL: semi-supervised learning; UL: unsupervised learning).

Model	Input data		Model design		
	G	$\boldsymbol{x}$	Learning	Weight range	Link prediction
PageRank [19]	✓		UL		
SSP [6]	✓	✓	SSL	$[0, \infty)$	
SNP [26]	✓	✓	SL	$[0, \infty)$	
AttriRank [10]	✓	✓	UL	All equal	
DeepRank	✓	✓	UL	$\mathbb{R}$	✓

3 DeepRank architecture

3.1 Problem definition

Given a direct 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 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. 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

neurons …

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 of our published journal 1.

3.2 Overview

Figure 2 illustrates our multi-task deep neural network structure. We regard a node pair $(i, j) \in E \cup F$ as a data instance. Nodes i and j each has its own input layers and output layers, but share the same hidden layers for learning latent representation. Initially, the input layer accepts the K-dimensional attribute vector 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 near 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:

$$(\pi_i, \boldsymbol{z}_i) = f(\boldsymbol{x}_i) = h^{(L)}(\dots h^{(2)}(h^{(1)}(\boldsymbol{x}_i))). \tag{2}$$

The weight parameters in this neural network reflect our representation inference in Figure 1. If all the parameters $\boldsymbol{\omega}$, and $\boldsymbol{b}$ are given, then for any node, Equation (2) can transform its attribute vector $\boldsymbol{x}$ to its representation form $(\pi, \boldsymbol{z})$. In this way, node ranking and link prediction are jointly modeled since 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, rather than 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:

$$\arg\min_{\pi\geq 0, \boldsymbol{z}\geq 0, \boldsymbol{W}} \sum_{(i,j)\in E} -\log\sigma\left(\frac{\pi_j}{m_j} - \frac{\pi_i}{n_i}\right) + \lambda \sum_{(i,j)\in F} (\pi_i + \pi_j) + \nu \sum_{(i,j)\in E\cup F} \left(\boldsymbol{z}_i^\top \boldsymbol{W} \boldsymbol{z}_j - a_{ij}\right)^2, \tag{3}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ 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. 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 $\pi \geq 0, \boldsymbol{z} \geq 0$, both are satisfied as long as we choose an activation function whose output is non-negative (for example, Rectified Linear Unit (ReLU) or softplus [7]) at the output layer. All the parameters in the hidden layers need not to be non-negative. We will provide theoretical justification of equation (3) in Section 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, backpropagation Stochastic Gradient Descent (SGD) is exploited. The only constraint is the non-negative latent representation in the network. Thus advanced neural network techniques like RMSProp [9], Dropout [20], etc. can be applied straightforwardly.

3.3 Complexity analysis

We present DeepRank pseudocode (please refer to the pseudocode (Appendix B) in our published journal 1) to analyze the complexity. In terms of space, we have M positive instances, M negative instances, and each of the N nodes has K-dimensional attributes. Therefore the space complexity is $O(M + NK)$. In terms of time, a single instance needs to optimize Ω^2 parameters between two consecutive layers, in which Ω is the maximum number of neurons in a layer. Hence if DeepRank is modeled by L hidden layers, then every epoch in SGD optimization takes $O(ML\Omega^2)$ time. The scalability is confirmed as both space and time complexity are linear to the number of links M or nodes N.

4 DeepRank objective analysis

Here we present the theoretical analysis for the DeepRank objective function. As the most popular optimization technique suitable for DNN is the Stochastic Gradient Descent (SGD) based solution, we prefer an unconstrained objective function to maintain the flexibility of the DNN representation.

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:

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

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 have 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.

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 can violate the constraint. Our goal is to adjust the optimization equation so we do not need to adjust λ and s altogether. Instead of (5), we choose to minimize its upper bound:

Lemma 4.1. *We can derive an upper bound of (5) as follows:*

$$\arg\min_{\pi \geq 0} \sum_{j \in V} \left(\sum_{i \in P_j} \frac{\pi_i}{n_i} - \pi_j + \lambda s \right)^2 + \lambda \left(\sum_{j \in V} \pi_j \right)^2 + \lambda s^2 - \sum_{j \in V} \lambda^2 s^2 \tag{6}$$

Proof. Please refer to Appenix C in our published journal 1. □

The last two terms are constant such that they do not influence the optimized solution of π. 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 a replacement: As $\lambda s \geq 0$ and the second term in (6) suggests π cannot be too large, we choose to satisfy

$$\pi_j \geq \sum_{i \in P_j} \frac{\pi_i}{n_i}. \tag{7}$$

By doing such we can essentially eliminate the need to tune the hyper-parameter pairs λs. We then choose the maximum likelihood estimation to re-formulate the objective function representing (7).

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

where the sigmoid function σ 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:

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

By the following lemma, we show that the original assumption (7) can be reduced to (9).

Lemma 4.2. *If objective (9) is satisfied, then objective (7) must be satisfied.*

Proof. Please refer to Appendix D in our published journal paper 1. □

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

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

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 tries to inherit the spirit of the ranking propagation in PageRank and in the meantime making some adjustment to satisfy the SGD condition and reducing hyper-parameters.

4.2 Regularizing low-degree instances

As also mentioned in (6), a regularization over the ranking scored is preferable. Here the second term in (3), shown as (11), is designed for similar purpose with some maneuver.

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

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 the technology of *negative sampling* to randomly sample a set of non-existing links (please refer to Appendix B in our published journal for the pseudo code). Let F denotes such set of negative examples. 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 in F as defined below:

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

where S_i is the set of direct successors of node i. With respect to probability, our regularization term is the expected value:

$$\arg\min_{\pi \geq 0} \sum_{i\in V} \sum_{j\in V} \frac{N - n_i}{N^2 - M} \frac{N - m_j}{N^2 - M} (\pi_i + \pi_j) = \frac{1}{N^2 - M} \left(\sum_{i\in V} (N - n_i)\pi_i + \sum_{j\in V} (N - m_j)\pi_j \right). \tag{12}$$

For each (π_i, π_j), it is the regularization weighted by $N - n_i$ 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 available information in the input network.

4.3 Link prediction as tri-factorization

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

Table 2: Model performance comparison across datasets (†: outperforms 2nd-best with p-value < 0.01).

Dataset	Evaluation	Closeness	Betweenness	PageRank	WPR	SSP	AttriRank
Webspam	AUC	0.577	0.556	0.553	0.509	0.559	0.666
FB Wall Post	AUC	0.755	0.765	0.775	0.765	0.786	0.810

Dataset	Evaluation	Two-Stage	DeepRank($\nu = 0$)	DeepRank
Webspam	AUC	0.541	0.582	**0.671**†
FB Wall Post	AUC	0.510	0.777	**0.828**†

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 about 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 belonging to these D clusters. Then link a_{ij} can be expressed as the expected value $\mathbb{E}$:

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

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} could be highly positive; otherwise, $w_{cd} < 0$ for two rival clubs. Here we assume the association of students to clubs are independent. Let $\boldsymbol{z}_i \geq 0$ be the cluster probability distribution of node i, and thus 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) $\phi(x) = \max\{0, x\}$ [7] for each dimension of vector $\boldsymbol{z}$ to produce sparse $\boldsymbol{z}_i$. We also notice a recently proposed sparsemax function [16] that introduces sparsity to the softmax, and will experiment this choice in our future work.

5 Experiment

Datasets. We prepare two datasets with different ranking applications to verify our proposal: (I) *Webspam* [3] ($114,529$ webpages, $1,836,441$ links, 122 webpages marked spam while $1,933$ labeled non-spam); (II) *FB Wall Post* [4]: ($63,731$ users, $831,401$ links, $14,862$ marked active users and $48,869$ labeled inactive users). See details in section 5 of our published journal 1.

Evaluation. Since the labels are binary in Webspam and FB Wall Post datasets, we use the popular Area Under ROC Curve (AUC) as the evaluation metric.

Competitors. We compare DeepRank with general-purpose unsupervised node ranking models: (I) *Closeness and betweenness centrality* [5]; (II) *PageRank* [19]; (III) *Weighted PageRank (WPR)* [24]; (IV) *Semi-Supervised PageRank (SSP)* [6] (we use its unsupervised component only); (V) *AttriRank* [10] (state-of-the-art model). See details in our supplementary material section 6.

DeepRank Details. As shown in Figure 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.

[3] `http://chato.cl/webspam/datasets/uk2007/`

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

All hidden layers use Exponential Linear Unit (ELU) [4] as the activation function with Dropout [20] of 75% keep probability for faster and more accurate learning. The non-sparse-induced softplus [7] activation is used as the node ranking output layer to avoid zero-score nodes. The Rectified Linear Unit (ReLU) [7] is selected at the output layer of link prediction part, as explained in Section 4.3. We fix $(\lambda, \nu) = (0.1, 10.0)$ for all experiments. RMSProp [9] (decay ratio 0.9 and momentum 0.95) is applied for Stochastic Gradient Descent (SGD) with a batch size of 256. The convergence is reached when the objective values change less than 0.1% between epochs. The convergence costs no more than 50 epochs in our experiments.

Experiment Settings. For the parameters of experimented 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 Webspam 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 the FB Wall Post dataset that have no external attributes, we then referring to the settings in [10]: for each node, we generate 13-dimensional attribute vector directly from the graph [5] 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.

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

(I) Does DeepRank outperform the competitors in terms of node ranking?

The results in Table 2 show that DeepRank significantly outperforms the other competitors across different applications, 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.

(II) Is our joint learning method better than a two-stage model?

In Section 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 2 show that the joint learning model does outperform the two-stage model significantly.

(III) Can the link prediction component boost 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 2 demonstrate that taking link prediction into consideration brings significant improvement.

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), considering convolution input structures for other kinds of attributes such as image, and incorporating additional attributes relevant to edges.

References

[1] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. WSDM '11, pages 635–644, New York, NY, USA, 2011. ACM.

[5] 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 the mean of in-degrees of direct successors; (6, 7) the sum and the 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 $k - 1$.

[2] Lev Bogolubsky, Pavel Dvurechensky, Alexander Gasnikov, Gleb Gusev, Yurii Nesterov, Andrei M Raigorodskii, Aleksey Tikhonov, and Maksim Zhukovskii. Learning supervised pagerank with gradient-based and gradient-free optimization methods. NIPS'16.

[3] Huan Chang, David Cohn, and Andrew McCallum. Learning to create customized authority lists. ICML '00, pages 127–134, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[4] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ArXiv e-prints*, November 2015.

[5] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215 – 239, 1978-1979.

[6] Bin Gao, Tie-Yan Liu, Wei Wei, Taifeng Wang, and Hang Li. Semi-supervised ranking on very large graphs with rich metadata. KDD '11, pages 96–104, New York, NY, USA, 2011. ACM.

[7] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. AISTATS'11.

[8] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. VLDB '04, pages 576–587. VLDB Endowment, 2004.

[9] Geoffrey Hinton. Neural networks for machine learning, coursera video lectures, 2012.

[10] Chin-Chi Hsu, Yi-An Lai, Wen-Hao Chen, Ming-Han Feng, and Shou-De Lin. Unsupervised ranking using graph structures and node attributes. WSDM '17, pages 771–779, New York, NY, USA, 2017. ACM.

[11] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 1999.

[12] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, May 2007.

[13] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. KDD '10, pages 243–252, New York, NY, USA, 2010. ACM.

[14] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[15] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Comput. Surv.*, 49(4):69:1–69:33, December 2016.

[16] André F. T. Martins and Ramón F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. ICML'16, pages 1614–1623. JMLR.org, 2016.

[17] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. ECML PKDD'11, pages 437–452, Berlin, Heidelberg, 2011. Springer-Verlag.

[18] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64:025102, Jul 2001.

[19] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting.

[21] Ah Chung Tsoi, Gianni Morini, Franco Scarselli, Markus Hagenbuchner, and Marco Maggini. Adaptive ranking of web pages. WWW '03, pages 356–365, New York, NY, USA, 2003. ACM.

[22] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[23] Yujing Wang, Yunhai Tong, and Ming Zeng. Ranking scientific articles by exploiting citations, authors, journals, and time information, 2013.

[24] W. Xing and A. Ghorbani. Weighted pagerank algorithm. In *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.*, pages 305–314, May 2004.

[25] Shuangfei Zhai and Zhongfei (Mark) Zhang. Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs. SDM'15, 2015.

[26] Maxim Zhukovskiy, Gleb Gusev, and Pavel Serdyukov. Supervised nested pagerank. CIKM '14, pages 1059–1068, New York, NY, USA, 2014. ACM.

List of Publications and Significant Collaborations that resulted from your AOARD supported project: In standard format showing authors, title, journal, issue, pages, and date, for each category list the following:

1. Yi-An Lai, Chin-Chi Hsu, Wen-Hao Chen, Mi-Yen Yeh, and Shou-De Lin, "DeepRank: improving unsupervised node ranking via link discovery," Data Mining and Knowledge Discovery, volume 33, number 2, pages 474–498, March 2019.
2. Chin-Chi Hsu, Mi-Yen Yeh, Shou-De Lin: A General Framework for Implicit and Explicit Social Recommendation. IEEE Trans. Knowl. Data Eng. 30(12): 2228-2241 (2018)
3. Yi-An Lai, Chin-Chi Hsu, Wen-Hao Chen, Mi-Yen Yeh, Shou-De Lin: PRUNE: Preserving Proximity and Global Ranking for Network Embedding. NIPS 2017: 5263-5272