Homogeneous Network Embedding for Massive Graphs via Reweighted Personalized PageRank

Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, Sourav Saha Bhowmick

August 2020

NANYANG TECHNOLOGICAL UNIVERSITY

THE HONG KONG POLYTECHNIC UNIVERSITY

NUS

School of Computing

HAMAD BIN KHALIFA UNIVERSITY
Outline

• Problem Definition & Applications
• Existing Work & Motivations
• Proposed solution: NRP
• Experiments
Homogeneous Network Embedding (HNE)

\[ G = (V, E) \]
\[ n = |V|, \ m = |E| \]

- **Link Prediction**
  - [Backstrom et al., WSDM’2011]
  - [Gupta et al., KDD’2013]

- **Graph Reconstruction**
  - [Radivojac et al., Nature methods’2004]

- **Node Classification**
  - [Perozzi et al., KDD’2014]
  - [Ribeiro et al., KDD’2017]
Existing Work

• Learning-based HNE methods
  – with random walks
    • truncated random walks: Deepwalk [Perozzi et al. KDD14],
    • biased random walks: Node2vec [Grover et al. KDD16],
    • Personalized PageRank (PPR): VERS [Tsitsulin et al. WWW18], APP
      [Zhou et al. AAAI]

\[ X_u \cdot X_v \sim \Pr[u \rightarrow v] \]

  – without random walks
    • Auto-encoders, graph neural networks (GNN), generative adversarial networks (GAN), long short-term memory networks (LSTM)

Expensive training courses!
Existing Work

• Factorization-based HNE methods
  – Construct an $n \times n$ proximity matrix $\mathbf{M}$
    • Katz score, AROPE [Zhang et al. KDD18]
    • PPR, STRAP [Yin et al. KDD 2019]
  – Factorize $\mathbf{M} = \mathbf{X} \cdot \mathbf{Y}^T$ (e.g., SVD, NMF)

\[ O(n^2)! \]
Motivations: Efficiency

Exact PPR

\[ \Pi = \sum_{t=0}^{\infty} \alpha(1 - \alpha)^t P^t \]

\[ M = \sum_{t=1}^{l_1} \alpha(1 - \alpha)^t P^t \]

\[ \mathcal{O}(n^2) \] too dense! \hspace{1cm} \mathcal{O}(n^3) \] too slow!

\[ \mathcal{O}(m) \]

\[ \mathcal{O}(mk\log(n)) \]

\[ \text{SVD} \]

How to refine?
Motivations: Effectiveness

Table 1: PPR for $v_2$ and $v_9$ in Fig. 1 ($\alpha = 0.15$).

| $v_i$  | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ | $v_9$ |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| $\pi(v_2, v_i)$ | 0.15  | 0.269 | 0.188 | 0.118 | 0.17  | 0.048 | 0.029 | 0.019 | 0.008 |
| $\pi(v_4, v_i)$ | 0.15  | 0.118 | 0.188 | 0.269 | 0.17  | 0.048 | 0.029 | 0.019 | 0.008 |
| $\pi(v_7, v_i)$ | 0.036 | 0.043 | 0.056 | 0.043 | 0.093 | 0.137 | 0.29  | 0.187 | 0.12  |
| $\pi(v_9, v_i)$ | 0.02  | 0.024 | 0.031 | 0.024 | 0.056 | 0.083 | 0.168 | 0.311 | 0.282 |

3 common neighbours

Potential link: $(v_2, v_4) > (v_7, v_9)$

1 common neighbour

Why?

Neglect node global importance

How?

Reweight nodes with weights!

$\pi(v_2, v_4) + \pi(v_4, v_2) = 0.236$

$\pi(v_9, v_7) + \pi(v_7, v_9) = 0.288$
Proposed solution: NRP

• Basic idea: \( \forall v \in V \)
  - A forward embedding \( X_v \)
  - A backward embedding \( Y_v \)
  - A forward weight \( \bar{w}_v \)
  - A backward weight \( \bar{w}_v \)

  \[
  X_u \cdot Y_v^T \approx \bar{w}_u \cdot \pi (u, v) \cdot \bar{w}_v
  \]

  \[
  X_u \cdot Y_v^T \neq X_v \cdot Y_u^T
  \]

  Preserve node global importance
  Preserve node proximity
  Preserve edge direction

• Challenges
  - Approximate PPR \( \pi (u, v) \) for all \( (u, v) \) pairs efficiently
  - Learn \( \bar{w}_v / \bar{w}_v \) reflecting node importance
NRP: Step 1: Approximate PPR

\[
P \approx X_1 \cdot Y^T
\]

\[
P \approx X_1 \cdot Y^T
\]

\[
X = \begin{bmatrix}
X_{v_1} \\
X_{v_2} \\
X_{v_3} \\
X_{v_4} \\
X_{v_5} \\
X_{v_6} \\
X_{v_7} \\
X_{v_8} \\
X_{v_9}
\end{bmatrix} = \sum_{t=1}^{t_1} \alpha(1 - \alpha)^t P^{t-1} X_1
\]

\[
M \approx X \cdot Y^T
\]

\[
O(mk \log(n))
\]

\[
O(mk t_1)
\]
NRP: Step 2: Node Reweighting

- **Intuition:**
  1. total strength of connections from other nodes to \( u = \) in-degree of \( u \)
  2. total strength of connections from \( u \) to other nodes = out-degree of \( u \)

- **Objective function:**

    \[
    O = \min_{\hat{w}, \hat{w}'} \sum_v \left( \sum_{u \neq v} \left( \hat{w}_u \mathbf{X}_u \mathbf{Y}_v^\top \hat{w}'_v \right) - d_{in}(v) \right) \quad \text{(1)}
    \]

    \[
    + \sum_u \left( \sum_{u \neq v} \left( \hat{w}_u \mathbf{X}_u \mathbf{Y}_v^\top \hat{w}_v \right) - d_{out}(u) \right) \quad \text{(2)}
    \]

    \[
    + \lambda \sum_u \left( \| \hat{w}_u \|_2 + \| \hat{w}'_u \|_2 \right),
    \]

    subject to \( \forall u \in V, \hat{w}_u, \hat{w}'_u \geq \frac{1}{n} \).

- **Output:** \( \forall v \in V \)

    \[
    \mathbf{X}_v \leftarrow \hat{w}'_v \cdot \mathbf{X}_v
    \]

    \[
    \mathbf{Y}_v \leftarrow \hat{w}'_v \cdot \mathbf{Y}_v
    \]
Experiments: Settings

Table 2. Data Sets

| Name        | |V|   | |E|   | Type      | #labels |
|-------------|---|-----|---|-----|----------|---------|
| Wiki        | 4.78K | 184.81K | directed | 40 |
| BlogCatalog | 10.31K | 333.98K | undirected | 39 |
| Youtube     | 1.13M | 2.99M | undirected | 47 |
| TWeeibo     | 2.32M | 50.65M | directed | 100 |
| Orkut       | 3.1M | 234M | undirected | 100 |
| Twitter     | 41.6M | 1.2B | directed | - |
| Friendster  | 65.6M | 1.8B | undirected | - |

- NRP: \( k = 128, \, \iota_1 = 20, \, \iota_2 = 10, \, \alpha = 0.15 \)
- ApproxPPR: \( k = 128, \, \iota_1 = 20, \, \alpha = 0.15 \) (without reweighting)
- an Intel Xeon(R) E5-2650 v2@2.60GHz CPU and 96GB RAM
Experiments: Link Prediction

(a) Wiki

(b) BlogCatalog

(c) TWeibo

(d) Orkut

(e) Twitter

(f) Friendster
Experiments: Efficiency

- NRP
- VERSE
- GA
- APP
- AROPE
- PBG
- STRAP
- RandNE
- node2vec
- ProNE
- DeepWalk
- LINE
- DNNGR
- DRNE
- RaRE
- NetHiex
- GraphWave
- GraphGAN
- NetSMF
- ApproxPPR

(a) Wiki
(b) BlogCatalog
(c) TWeibo
(d) Orkut
(e) Twitter
(f) Friendster
Thanks

Q & A
Why NRP Works

NRP preserves

• Multi-hop proximity between nodes (PPR)
• The global importance of nodes (Reweighting)
• Edge directions (forward/backward embeddings)
  – For example

Me (Nobody) ➔ Follow ➔ Donald J. Trump ➔ Follow? ➔ Not interested!
Competitors

• Factorization-based
  – AROPE, RandNE, NetSMF, ProNE, STRAP

• Random-walk-based
  – DeepWalk, LINE, node2vec, PBG, APP, VERSE

• Neural-network-based
  – DNGR, DRNE, GraphGAN, GA

• Other
  – RaRE, NetHiex, GraphWave
Experiments: Graph Reconstruction

Figure 5: Graph reconstruction results vs. $K$ (best viewed in color).
Experiments: Node Classification

Figure 6: Node classification results (best viewed in color).
Experiments: Parameter Analysis

Figure 8. Link prediction results
Experiments: Link Prediction on Dynamic Graphs

| Name | $|V|$ | $|E|$ | $|E_{old}|$ | $|E_{new}|$ | Type |
|------|------|------|----------|----------|------|
| VK   | 78.59K | 5.35M | 2.68M    | 2.67M    | undirected |
| Digg | 279.63K | 1.73M | 1.03M    | 701.59K  | directed |

Table 4: Dataset statistics ($K = 10^3$, $M = 10^6$).

Figure 9: Link prediction performance on dynamic graphs (best viewed in color).
Experiments: Efficiency

Figure 10: Scalability tests.

Figure 11: Running time with varying parameters (best viewed in color).
Algorithm 1: ApproxPPR

Input: $A, D^{-1}, P, \alpha, k', \ell_1, \epsilon.$
Output: $X, Y.$

1. $[U, \Sigma, V] \leftarrow \text{BKSVD}(A, k', \epsilon);$  
2. $X_1 \leftarrow D^{-1}U\sqrt{\Sigma}, \quad Y \leftarrow V\sqrt{\Sigma};$
3. for $i \leftarrow 2$ to $\ell_1$ do
4. $X_i \leftarrow (1 - \alpha)PX_{i-1} + X_1;$
5. $X \leftarrow \alpha(1 - \alpha)X_{\ell_1};$
6. return $X, Y;$


Node Reweighting

Algorithm 2: updateBwdWeights

```
Input: \( G, k', \vec{w}, \hat{w}, X, Y \).
Output: \( \hat{w} \)
1 Compute \( \xi, \chi, \rho_1, \rho_2, \Lambda, \) and \( \Phi \) based on Eq. (9), (10), and (13);
2 for \( r \leftarrow 1 \) to \( k' \) do
3 \( \phi[r] = \sum_u \hat{w}^2_u X_u[r]^2 \);
4 for \( v^* \in V \) in random order do
5 \( \text{Compute } a_1, a_2, a_3, b_1, b_2 \) by Eq. (9), (10), and (14);
6 \( \vec{w}'_{v^*} = \hat{w}_{v^*} \);
7 \( \hat{w}_{v^*} = \max \left\{ \frac{1}{n}, \frac{a_1+a_2-a_3}{b_1+b_2+\lambda} \right\} \);
8 \( \rho_1 = \rho_1 + (\vec{w}_{v^*} - \hat{w}'_{v^*}) Y_{v^*} \);
9 \( \rho_2 = \rho_2 + (\vec{w}_{v^*} - \hat{w}'_{v^*}) \hat{w}^2_{v^*} (X_{v^*} Y_{v^*}^\top) X_{v^*} \);
10 return \( \hat{w} \);
```

Algorithm 4: updateFwdWeights

```
Input: \( G, k', \vec{w}, \hat{w}, X, Y \).
Output: \( \vec{w} \)
1 Compute \( \xi, \chi, \rho_1, \rho_2, \Lambda \) based on Eq. (24), (25);
2 for \( r \leftarrow 1 \) to \( k' \) do
3 \( \phi[r] = \sum_v \hat{w}^2_v Y_v[r]^2 \);
4 for \( u^* \in V \) in random order do
5 \( \text{Compute } a'_1, a'_2, a'_3, b'_1, b'_2 \) by Eq. (24), (25), and (29);
6 \( \vec{w}'_{u^*} = \hat{w}_{u^*} \);
7 \( \hat{w}_{u^*} = \max \left\{ \frac{1}{n}, \frac{a'_1+a'_2-a'_3}{b'_1+b'_2+\lambda} \right\} \);
8 \( \rho_1 = \rho_1 + (\vec{w}_{u^*} - \hat{w}'_{u^*}) X_{u^*} \);
9 \( \rho_2 = \rho_2 + (\vec{w}_{u^*} - \hat{w}'_{u^*}) \hat{w}^2_{u^*} (X_{u^*} Y_{u^*}^\top) Y_{u^*} \);
10 return \( \vec{w} \);
```

NANYANG TECHNOLOGICAL UNIVERSITY
Algorithm 3: NRP

Input: Graph $G$, embedding dimensionality $k$, thresholds $\ell_1, \ell_2$, random walk decay factor $\alpha$ and error threshold $\epsilon$

Output: Embedding matrices $X$ and $Y$.

1. $k' \leftarrow k/2$;
2. $[X, Y] \leftarrow \text{ApproxPPR}(A, D^{-1}, P, \alpha, k', \ell_1, \epsilon)$;
3. for $v \in V$ do
4. \hspace{1em} $\overline{w}_v = d_{out}(v), \overline{w}_v = 1$;
5. for $l \leftarrow 1$ to $\ell_2$ do
6. \hspace{1em} $\overline{w} = \text{updateBwdWeights}(G, k', \overrightarrow{w}, \overleftarrow{w}, X, Y)$;
7. \hspace{1em} $\overrightarrow{w} = \text{updateFwdWeights}(G, k', \overrightarrow{w}, \overleftarrow{w}, X, Y)$;
8. for $v \in V$ do
9. \hspace{1em} $X_v = \overrightarrow{w}_v \cdot X_v, \ Y_v = \overleftarrow{w}_v \cdot Y_v$;
10. return $X, Y$;