NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization

Jiezhong Qiu

Tsinghua University

June 17, 2019

Joint work with Yuxiao Dong (MSR), Hao Ma (Facebook AI), Jian Li (IIIS, Tsinghua), Chi Wang (MSR), Kuansan Wang (MSR), and Jie Tang (DCST, Tsinghua)

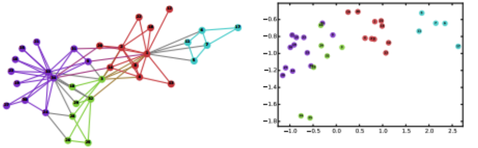
Motivation and Problem Formulation

Problem Formulation

Give a network $G = (V, E)$, aim to learn a function $f : V \to \mathbb{R}^p$ to capture neighborhood similarity and community membership.

Applications:

- link prediction
- community detection
- label classification



(a) Input: Karate Graph (b) Output: Representation

Figure 1: A toy example (Figure from DeepWalk).

Two Genres of Network Embedding Algorithm

- Local Context Methods:
  - LINE, DeepWalk, node2vec, metapath2vec.
  - Usually be formulated as a skip-gram-like problem, and optimized by SGD.
- Global Matrix Factorization Methods.
  - NetMF, GraRep, HOPE.
  - Leverage global statistics of the input networks.
  - Not necessarily a gradient-based optimization problem.
  - Usually requires explicit construction of the matrix to be factorized.

Notations

Consider an undirected weighted graph $G = (V, E)$, where $|V| = n$ and $|E| = m$.

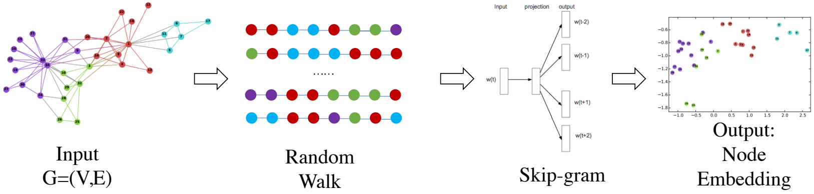
- Adjacency matrix $A \in \mathbb{R}_{+}^{n \times n}$:

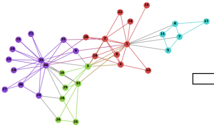
$$A_{i,j} = \begin{cases} a_{i,j} > 0 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}.$$

- Degree matrix $D = \mathrm{diag}(d_1, \cdots, d_n)$, where d_i is the generalized degree of vertex i.
- Volume of the graph G: $\mathrm{vol}(G) = \sum_i \sum_j A_{i,j}$.

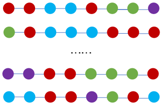
Contents

DeepWalk and NetMF

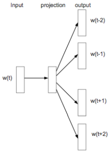




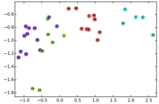
Input
G=(V,E)



Random
Walk



Skip-gram



Output:
Node
Embedding

Levy & Goldberg (NIPS 14)

$$\log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

b Number of negative samples

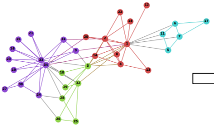
$\#(w,c)$ Co-occurrence of w and c

$\#(w)$ Occurrence of word w

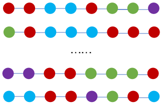
$|D|$ Total number of word-context pairs

$\#(c)$ Occurrence of context c

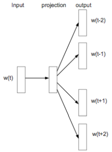
DeepWalk and NetMF



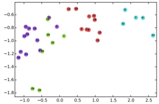
Input
G=(V,E)



Random
Walk



Skip-gram



Output:
Node
Embedding

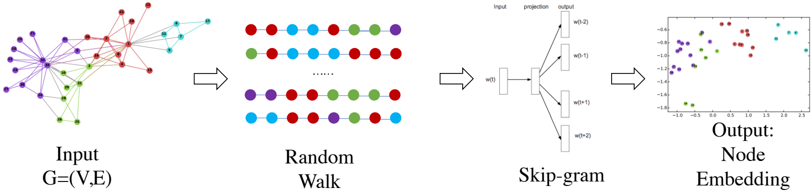
Levy & Goldberg (NIPS 14)

?

$$\log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

b	Number of negative samples
$\#(w,c)$	Co-occurrence of w and c
$\#(w)$	Occurrence of word w
$\lvert D\rvert$	Total number of word-context pairs
$\#(c)$	Occurrence of context c

DeepWalk and NetMF

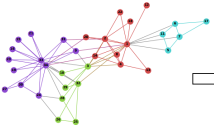


Levy & Goldberg (NIPS 14)

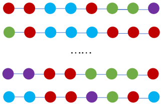
$$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right) \Longleftarrow \log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

$\boldsymbol{A}$	Adjacency matrix	$\mathrm{vol}(G)=\sum_i\sum_j \boldsymbol{A}_{i,j}$	
$\boldsymbol{D}$	Degree matrix	b	Number of negative samples

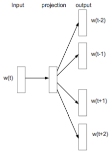
DeepWalk and NetMF



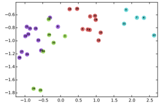
Input
G=(V,E)



Random
Walk



Skip-gram



Output:
Node
Embedding

Levy & Goldberg (NIPS 14)

$$\log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

$$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

Matrix
Factorization

Contents

Contents

Computation Challanges of NetMF

For small world networks,

$$\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\underbrace{\left(D\quad{}^{1}A\right)^{r}}_{\text{matrix power}}\right)D\quad{}^{1} \text{ is always a dense matrix .}$$

Computation Challanges of NetMF

For small world networks,

$$\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\underbrace{\left(D^{-1}A\right)^{r}}_{\text{matrix power}}\right)D^{-1} \text{ is always a dense matrix .}$$

Why?

- In small world networks, each pair of vertices (i, j) can reach each other in a small number of hops.
- Make the corresponding matrix entry a positive value.

Computation Challanges of NetMF

For small world networks,

$$\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\underbrace{\left(D^{-1}A\right)^{r}}_{\text{matrix power}}\right)D^{-1} \text{ is always a dense matrix .}$$

Why?

- In small world networks, each pair of vertices (i, j) can reach each other in a small number of hops.
- Make the corresponding matrix entry a positive value.

Idea

- Sparse matrix is easier to handle.
- Can we achieve a matrix sparse but 'good enough' matrix.

Observation

Definition
For $\sum_{r=1}^{T} \;\; _r = 1$ and $\;\; _r$ non-negative,

$$L = D \quad \sum_{r=1}^{T} \;\; _r D \left(D \quad ^1 A\right)^r \tag{1}$$

is a T-degree random-walk matrix polynomial.

Observation
For $\;\; _1 = \qquad = \;\; _T = \frac{1}{T}$:

$$
\begin{aligned}
& \log \quad \left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(D \quad ^1 A\right)^r\right) D \quad ^1\right) \\
= & \log \quad \left(\frac{\mathrm{vol}(G)}{b} D \quad ^1 (D \qquad L) D \quad ^1\right) \\
& \log \quad \left(\frac{\mathrm{vol}(G)}{b} D \quad ^1 (D \qquad \widetilde{L}) D \quad ^1\right)
\end{aligned}
$$

Random-walk Matrix Polynomial Sparsification

Theorem

[CCL[+]15] For random-walk matrix polynomial $L = D \quad \sum_{r=1}^{T} \quad {}_{r} D \left(D \quad {}^{1} A \right)^{r}$, one can construct, in time $O(T^2 m \quad {}^2 \log^2 n)$, a $(1 + \quad)$-spectral sparsi er, $\widetilde{L}$, with $O(n \log n \quad {}^2)$ non-zeros. For unweighted graphs, the time complexity can be reduced to $O(T^2 m \quad {}^2 \log n)$.

NetSMF—Algorithm

The proposed NetSMF algorithm consists of three steps:

- Construct a random walk matrix polynomial sparsifier, $\widetilde{L}$, by calling PathSampling algorithm proposed in [CCL+15].
- Construct a NetMF matrix sparsifier.

$$\mathrm{trunc\_log} \left(\frac{\mathrm{vol}(G)}{b} D^{-1} (D - \widetilde{L}) D^{-1} \right)$$

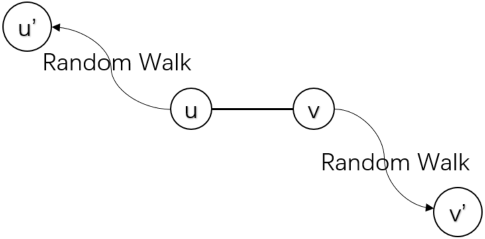
- Truncated randomized singular value decomposition.

Detailed Algorithm

Algorithm Details

PathSampling:

- Sample an edge $(u; v)$ from edge set.
- Start very short random walk from u and arrive u^0.
- Start very short random walk from v and arrive v^0.
- Record vertex pair $(u^0; v^0)$.



Randomized SVD:

- Project origin matrix to low dimensional space by Gaussian random matrix.
- Deal with the projected small matrix.

NetSMF — System Design

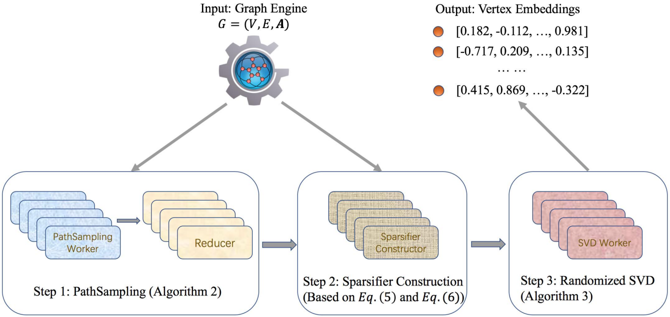


Figure 2: The System Design of NetSMF.

Contents

Setup

Label Classification:

- BlogCatelog, PPI, Flickr, YouTube, OAG.
- Logistic Regression
- NetSMF ($T = 10$), NetMF ($T = 10$), DeepWalk, LINE.

Table 1: Statistics of Datasets.

Dataset	**BlogCatalog**	**PPI**	**Flickr**	**YouTube**	**OAG**
$\|V\|$	10,312	3,890	80,513	1,138,499	67,768,244
$\|E\|$	333,983	76,584	5,899,882	2,990,443	895,368,962
#Labels	39	50	195	47	19

Experimental Results

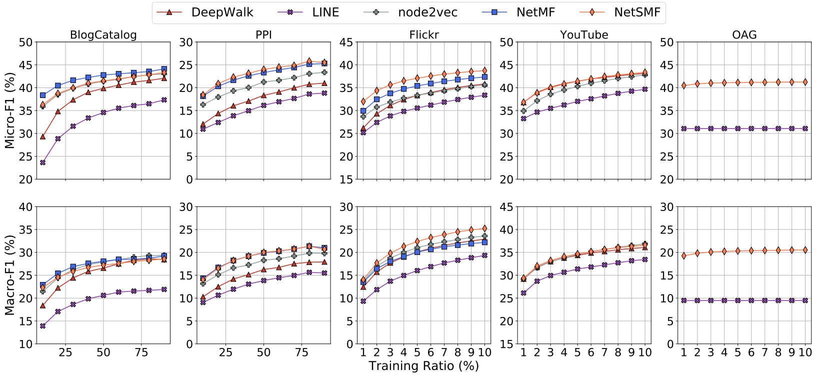


Figure 3: Predictive performance on varying the ratio of training data. The x-axis represents the ratio of labeled data (%), and the y-axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores respectively.

Running Time

Table 2: Running Time

	LINE	DeepWalk	node2vec	NetMF	NetSMF
BlogCatalog	40 mins	12 mins	56 mins	2 mins	13 mins
PPI	41 mins	4 mins	4 mins	16 secs	10 secs
Flickr	42 mins	2.2 hours	21 hours	2 hours	48 mins
YouTube	46 mins	1 day	4 days		4.1 hours
OAG	2.6 hours	–	–		24 hours

Conclusion and Future Work

We propose NetSMF, a scalable, efficient, and effective network embedding algorithm.

Future Work

- A distributed-memory implementation.
- Extension to directed, dynamic, heterogeneous graphs.

Thanks.

- Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec (WSDM '18)
- NetSMF: Network Embedding as Sparse Matrix Factorization (WebConf '19)

Code for NetMF available at `github.com/xptree/NetMF`
Code for NetSMF available at `github.com/xptree/NetSMF`
Q&A

On the Large-dimensionality Assumption of [LG14]

Recall the objective of skip-gram model:

$$\min_{X,Y} \mathcal{L}(X, Y)$$

where

$$\mathcal{L}(X, Y) = |\mathcal{D}| \sum_{w} \sum_{c} \left(\frac{\#(w,c)}{|\mathcal{D}|} \log g(x_w^\top y_c) + b \frac{\#(w)}{|\mathcal{D}|} \frac{\#(c)}{|\mathcal{D}|} \log g(-x_w^\top y_c) \right)$$

Theorem

For DeepWalk, when the length of random walk $L \to \infty$,

$$\frac{\#(w,c)}{|\mathcal{D}|} \xrightarrow{p} \frac{1}{2T} \sum_{r=1}^{T} \left(\frac{d_w}{\mathrm{vol}(G)} (P^r)_{w,c} + \frac{d_c}{\mathrm{vol}(G)} (P^r)_{c,w} \right).$$

$$\frac{\#(w)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_w}{\mathrm{vol}(G)} \text{ and } \frac{\#(c)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_c}{\mathrm{vol}(G)}.$$

Back

NetSMF — Approximation Error

Denote $M = \color{red}{D^{-1}(D - L)D^{-1}}$ in

$$\operatorname{trunc\_log}^{\circ}\left(\frac{\mathrm{vol}(G)}{b}\color{red}{D^{-1}(D - \widetilde{L})D^{-1}}\right);$$

and $\widetilde{M}$ to be its sparsifier the we constructed.

Theorem

The singular value of $\widetilde{M} - M$ satisfies

$$\sigma_i(\widetilde{M} - M) \le \frac{4}{\sqrt{d_i d_{\min}}}, \forall i \in [n].$$

Theorem

Let $\|\cdot\|_F$ be the matrix Frobenius norm. Then

$$\left\|\operatorname{trunc\_log}^{\circ}\left(\frac{\mathrm{vol}(G)}{b}\widetilde{M}\right) - \operatorname{trunc\_log}^{\circ}\left(\frac{\mathrm{vol}(G)}{b}M\right)\right\|_F \le \frac{4}{b}\frac{\mathrm{vol}(G)}{\sqrt{d_{\min}}}\sqrt{\sum_{i=1}^{n}\frac{1}{d_i}}.$$

Spectrally Similar

Definition

Suppose $G = (V, E, A)$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{A})$ are two weighted undirected networks. Let $L = D_G - A$ and $\widetilde{L} = D_{\widetilde{G}} - \widetilde{A}$ be their Laplacian matrices, respectively. We define G and $\widetilde{G}$ are $(1+\epsilon)$-spectrally similar if

$$\forall x \in \mathbb{R}^n, (1-\epsilon) \cdot x^\top \widetilde{L} x \leq x^\top L x \leq (1+\epsilon) \cdot x^\top \widetilde{L} x.$$

NetSMF—Algorithm

Algorithm 1: NetSMF

Input : A social network $G = (V, E, \boldsymbol{A})$ which we want to learn network embedding; The number of non-zeros M in the sparsifier; The dimension of embedding d.

Output: An embedding matrix of size $n \times d$, each row corresponding to a vertex.

$\widetilde{G} \leftarrow (V, \emptyset, \widetilde{\boldsymbol{A}} = \boldsymbol{0})$

`/* Create an empty network with` $E = \emptyset$ `and` $\widetilde{\boldsymbol{A}} = 0$. `*/`

for $i \leftarrow 1$ **to** M **do**

  Uniformly pick an edge $e = (u, v) \in E$

  Uniformly pick an integer $r \in [T]$

  $u', v', Z \leftarrow$ `PathSampling(`e, r`)`

  Add an edge $\left(u', v', \frac{2rm}{MZ}\right)$ to $\widetilde{G}$

  `/* Parallel edges will be merged into one edge, with their weights summed up together. */`

end

Compute $\widetilde{\boldsymbol{L}}$ to be the unnormalized graph Laplacian of $\widetilde{G}$

Compute $\widetilde{\boldsymbol{M}} = \boldsymbol{D}^{-1}\left(\boldsymbol{D} - \widetilde{\boldsymbol{L}}\right)\boldsymbol{D}^{-1}$

$\boldsymbol{U}_d, \boldsymbol{\Sigma}_d, \boldsymbol{V}_d \leftarrow$ `RandomizedSVD(`trunc_log$^{\circ}\left(\frac{\mathrm{vol}(G)}{b}\widetilde{\boldsymbol{M}}\right), d$)

return $\boldsymbol{U}_d\sqrt{\boldsymbol{\Sigma}_d}$ as network embeddings

Back

Algorithm 2: PathSampling algorithm as described in [CCL+15].

Procedure `PathSampling`($e = (u; v)$, r)

Uniformly pick an integer k 2 $[r]$

Perform $(k \quad 1)$-step random walk from u to u_0

Perform $(r \quad k)$-step random walk from v to u_r

Keep track of $Z(p) = \sum_{i=1}^{r} \frac{2}{A_{u_{i-1}, u_i}}$ along the length-r path p between u_0 and u_r

return $u_0; u_r; Z(p)$

Back

Randomized SVD

Algorithm 3: Randomized SVD on NetMF Matrix Sparsifier

Procedure `RandomizedSVD`(A, d)

- Sampling Gaussian random matrix O // $O \in \mathbb{R}^{n \times d}$
- Compute sample matrix $Y = A^\top O = AO$ // $Y \in \mathbb{R}^{n \times d}$
- Orthonormalize Y
- Compute $B = AY$ // $B \in \mathbb{R}^{n \times d}$
- Sample another Gaussian random matrix P // $P \in \mathbb{R}^{d \times d}$
- Compute sample matrix of $Z = BP$ // $Z \in \mathbb{R}^{n \times d}$
- Orthonormalize Z
- Compute $C = Z^\top B$ // $C \in \mathbb{R}^{d \times d}$
- Run Jacobi SVD on $C = U \Sigma V^\top$
- **return** ZU, Σ, YV

/* Result matrices are of shape $n \times d$, $d \times d$, $n \times d$ resp. */

Time and Space Complexity

Table 3: Time and Space Complexity of NetSMF.

	Time	**Space**
Step 1	$O(MT \log n)$ for weighted networks $O(MT)$ for unweighted networks	$O(M + n + m)$
Step 2	$O(M)$	$O(M + n)$
Step 3	$O(Md + nd^2 + d^3)$	$O(M + nd)$

References I

Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng, *Spectral sparsi cation of random-walk matrix polynomials*, arXiv preprint arXiv:1502.03496 (2015).

Omer Levy and Yoav Goldberg, *Neural word embedding as implicit matrix factorization*, Advances in neural information processing systems, 2014, pp. 2177–2185.