NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization

Jiezhong Qiu[1], Yuxiao Dong[2], Hao Ma[4*], Jian Li[3], Chi Wang[2], Kuansan Wang[2], Jie Tang[1]

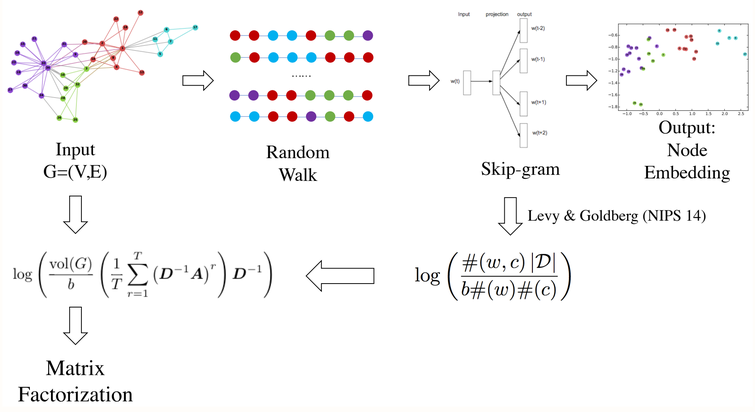
[1]Department of Computer Science and Technolegy, Tsinghua University

[2]Microsoft Research, Redmond, Washington

[3]Institute for Interdisciplinary Information Sciences, Tsinghua University

[4]Facebook AI *Work peroformed while at Microsoft Research

Revisit Skip-gram based Network Embedding Algorithms



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

Table 1: From [5], the matrices that are implicitly approximated and factorized by DeepWalk [4], LINE [6], and node2vec [3].

Algorithm	Matrix
DeepWalk	$\log^\circ\left(\mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}(\mathbf{D}^{-1}\mathbf{A})^r\right)\mathbf{D}^{-1}\right)-\log b$
LINE	$\log^\circ\left(\mathrm{vol}(G)\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\right)-\log b$
node2vec	$\log^\circ\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}(\sum_u \mathbf{X}_{w,u}\mathbf{P}^r_{c,w,u}+\sum_u \mathbf{X}_{c,u}\mathbf{P}^r_{w,c,u})}{(\sum_u \mathbf{X}_{w,u})(\sum_u \mathbf{X}_{c,u})}\right)-\log b$

NetSMF

NetMF Revisit

Normalized graph Laplacian has eigen-decomposition:

$$\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$$

where $\mathbf{U}$ is orthonormal and $\mathbf{\Lambda}=\mathrm{diag}(\lambda_1,\cdots,\lambda_n)$ with $\forall\lambda_i\in[-1,1]$. In NetMF, we approximates the NetMF matrix with the top-h eigen-paris of normalized graph Laplacian.

$$\begin{aligned}
&\log^\circ\left(\frac{\mathrm{vol}(G)}{b}\frac{1}{T}\sum_{r=1}^{T}(\mathbf{D}^{-1}\mathbf{A})^r\mathbf{D}^{-1}\right)\\
=&\log^\circ\left(\frac{\mathrm{vol}(G)}{b}\mathbf{D}^{-1/2}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^r\right)\mathbf{D}^{-1/2}\right)\\
=&\log^\circ\left(\frac{\mathrm{vol}(G)}{b}\mathbf{D}^{-1/2}\left(\mathbf{U}\left(\frac{1}{T}\sum_{r=1}^{T}\mathbf{\Lambda}^r\right)\mathbf{U}^\top\right)\mathbf{D}^{-1/2}\right)\\
\approx&\log^\circ\left(\frac{\mathrm{vol}(G)}{b}\mathbf{D}^{-1/2}\left(\mathbf{U}_h\left(\frac{1}{T}\sum_{r=1}^{T}\mathbf{\Lambda}_h^r\right)\mathbf{U}_h^\top\right)\mathbf{D}^{-1/2}\right)
\end{aligned}$$

Random-Walk Matrix-Polynomial Sparsification

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

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

Theorem [1, 2] For random-walk matrix polynomial $\mathbf{L}=\mathbf{D}-\sum_{r=1}^{T}\alpha_r\mathbf{D}\left(\mathbf{D}^{-1}\mathbf{A}\right)^r$, where $\sum_{r=1}^{T}\alpha_r=1$ and α_r non-negative, one can construct, in time $O(T^2m\epsilon^{-2}\log^2 n)$, a $(1+\epsilon)$-spectral sparsifier, $\widetilde{\mathbf{L}}$, with $O(n\log n\epsilon^{-2})$ non-zeros. For unweighted graphs, the time complexity can be reduced to $O(T^2m\epsilon^{-2}\log n)$.

NetSMF

Let $\alpha_i=\frac{1}{T},i\in[T]$ in $\mathbf{L}=\mathbf{D}-\sum_{r=1}^{T}\alpha_r\mathbf{D}\left(\mathbf{D}^{-1}\mathbf{A}\right)^r$, we can observe the tight connection between random walk matrix polynomial and NetMF matrix:

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

NetSMF — Algorithm

- Construct a random walk molynomial sparsifier.
- Construct NetMF matrix sparsifier.
- Truncated randomized singular value decomposition.

Algorithm 1: NetSMF

Input : A social network $G=(V,E,\mathbf{A})$ 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.

$\bar{G}\leftarrow(V,\emptyset,\bar{\mathbf{A}}=0)$
for $i\leftarrow 1$ ***to*** M **do**
  Uniformly pick an edge $e=(u,v)\in E$
  Uniformly pick an integer $r\in[T]$
  $u',v',Z\leftarrow\texttt{PathSampling}(e,r)$
  Add an edge $(u',v',\frac{2rm}{MZ})$ to $\bar{G}$
end
Compute $\bar{\mathbf{L}}$ to be the unnormalized graph Laplacian of $\bar{G}$
Compute $\mathbf{D}^{-1}\left(\mathbf{D}-\bar{\mathbf{L}}\right)\mathbf{D}^{-1}$
$\mathbf{U}_d,\mathbf{\Sigma}_d,\mathbf{V}_d\leftarrow\texttt{RandomizedSVD}(\mathrm{trunc\_log}^\circ\left(\frac{\mathrm{vol}(G)}{b}\mathbf{D}^{-1}\left(\mathbf{D}-\bar{\mathbf{L}}\right)\mathbf{D}^{-1}\right),d)$
return $\mathbf{U}_d\sqrt{\mathbf{\Sigma}_d}$ as network embeddings

System Design

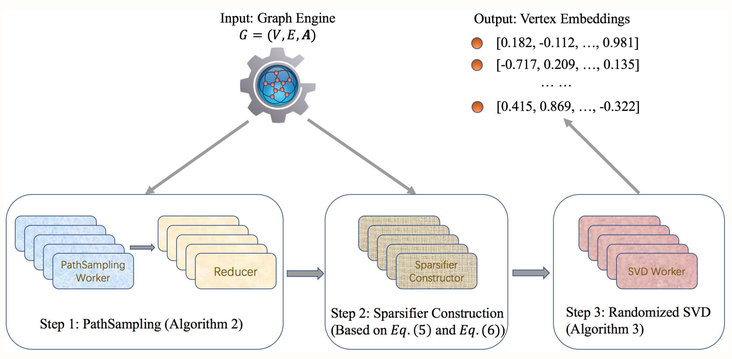


Figure 1: The System Design of NetSMF. The input comes from a graph engine which stores the network data and provides efficient APIs to graph queries. In Step 1, the system launches several PathSampling workers to handle a subset of samples. Then, a reducer is designed to aggregate the output of the PathSampling algorithm. In Step 2, the system distributes data to several sparsifier constructors to perform the transformation and the truncated element-wise matrix logarithm. In the final step, the system applies truncated randomized SVD on the constructed sparsifier and dumps the resulted embeddings to storage.

Experimental Result and Discussions

Multi-label Classification

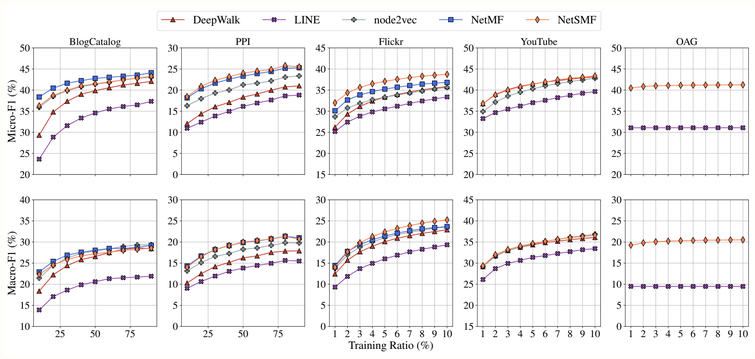


Figure 2: Predictive performance on varying the ratio of training data.

Table 2: Efficiency comparison.

	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

Table 3: The comparison between NetSMF and other popular network embedding algorithms.

	LINE	DeepWalk	node2vec	NetMF	NetSMF
Efficiency	√				√
Global context		√	√	√	√
Theoretical guarantee				√	√
High-order proximity			√		

About PathSampling and Randomized SVD

Algorithm 2: PathSampling algorithm

Procedure *PathSampling*$(e=(u,v),r)$
  Uniformly pick an integer $k\in[r]$
  Perform $(k-1)$-step random walk from u to u_0
  Perform $(r-k)$-step random walk from v to u_r
  Keep track of $Z(\mathbf{p})=\sum_{i=1}^{r}\frac{2}{\mathbf{A}_{u_{i-1},u_i}}$ along the length-r path $\mathbf{p}$ between u_0 and u_r
  return $u_0,u_r,Z(\mathbf{p})$

Algorithm 3: Randomized SVD

Procedure *RandomizedSVD*$(\mathbf{A},d)$
  Sampling Gaussian random matrix $\mathbf{O}$ // $\mathbf{O}\in\mathbb{R}^{n\times d}$
  Compute sample matrix $\mathbf{Y}=\mathbf{A}^\top\mathbf{O}=\mathbf{A}\mathbf{O}$ // $\mathbf{Y}\in\mathbb{R}^{n\times d}$
  Orthonormalize $\mathbf{Y}$
  Compute $\mathbf{B}=\mathbf{A}\mathbf{Y}$ // $\mathbf{B}\in\mathbb{R}^{n\times d}$
  Sample another Gaussian random matrix $\mathbf{P}$ // $\mathbf{P}\in\mathbb{R}^{d\times d}$
  Compute sample matrix of $\mathbf{Z}=\mathbf{B}\mathbf{P}$ // $\mathbf{Z}\in\mathbb{R}^{n\times d}$
  Orthonormalize $\mathbf{Z}$
  Compute $\mathbf{C}=\mathbf{Z}^\top\mathbf{B}$ // $\mathbf{C}\in\mathbb{R}^{d\times d}$
  Run Jacobi SVD on $\mathbf{C}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$
  return $\mathbf{Z}\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{Y}\mathbf{V}$

References

[1] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.-H. Teng. Efficient sampling for gaussian graphical models via spectral sparsification. In *COLT '15*, pages 364–390, 2015.

[2] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.-H. Teng. Spectral sparsification of random-walk matrix polynomials. *arXiv preprint arXiv:1502.03496*, 2015.

[3] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD '16*, pages 855–864. ACM, 2016.

[4] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD '14*, pages 701–710. ACM, 2014.

[5] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM '18*, pages 459–467. ACM, 2018.

[6] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW '15*, pages 1067–1077, 2015.

The Web Conference, San Francisco, May 13-17, 2019