vGraph: A Generative Model for Joint Community Detection and Node Representation Learning

Fan-Yun Sun[1,2] Meng Qu[2] Jordan Hoffmann[2,3] Chin-Wei Huang[2,4] Jian Tang[2,5,6]

[1]National Taiwan University
[2]Mila-Quebec Institute for Learning Algorithms, Canada
[3]Harvard University, USA
[4]Element AI, Canada
[5]HEC Montreal, Canada
[6]CIFAR AI Research Chair

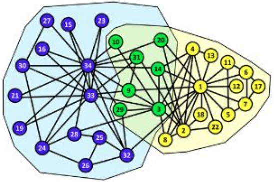
Jul 13, 2020

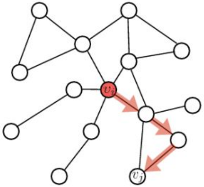
Introduction

Graphs are a general and flexible data structure to encode complex relationships among objects. Examples of real-world graphs include social networks, airline networks, protein-protein interaction networks, and traffic networks. This paper focuses on two fundamental tasks of graph analysis: **community detection** and **node representation learning**, which capture the global and local structures of graphs, respectively.

Motivation

(Non)Overlapping Community Detection

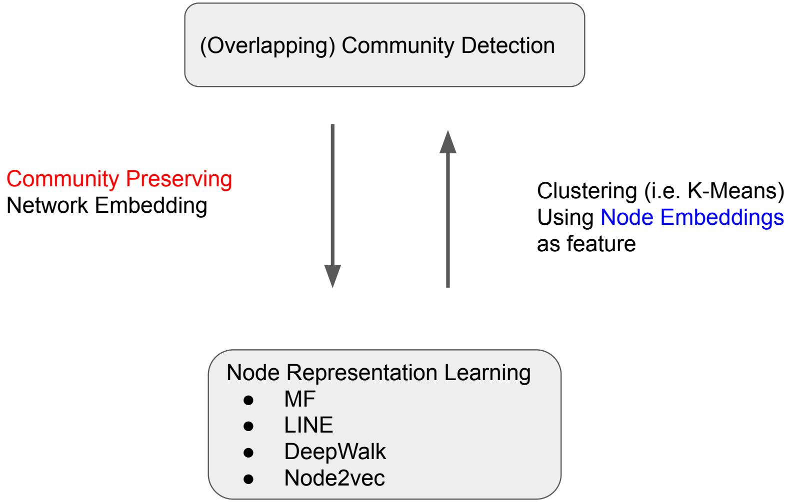




Node Representation Learning

- MF
- LINE
- DeepWalk
- Node2vec

Motivation


(Overlapping) Community Detection
Community Preserving
Network Embedding
Clustering (i.e. K-Means)
Using Node Embeddings
as feature
Node Representation Learning
MF
LINE
DeepWalk
Node2vec

vGraph - probabilistic generative model

- Generative model with Variational Inference to solve community detection and node representation learning jointly.
- Scalable: $O(d * |E| * K)$.
  - d: embedding dimension
  - $|E|$: number of edges in the graph
  - K: number of communities

Problem Definition

Given a graph $G(V, E)$ where V and E represent the set of vertices and edges respectively, we aim to jointly learn a node embedding $\phi_i \in \mathbb{R}^d$ and community affiliation $\mathcal{F}(\nu_i) \subseteq \{1, ..., K\}$ for each vertex ν_i.

Model

vGraph assumes that the edges (w, c) is generated from the following stocastic process: for node w, we first draw a community assignment $z \sim p(z|w)$ representing the social context of w during the generation process. Then, the linked neighbor c is generated based on the assignment z through $c \sim p(c|z)$. Formally, this process can be fomulated as:

$$p(c|w) = \sum_z p(c|z)p(z|w)$$

Model

$$p(c|w) = \sum_z p(c|z)p(z|w)$$

vGraph parameterizes the distributions $p(z|w)$ and $p(c|z)$ by introducing three sets of embeddings: ϕ_i, φ_i, and ψ_j. The prior distribution $p_{\phi,\psi}(z|w)$ and the node distribution conditioned on a community $p_{\psi,\varphi}(c|z)$ are parameterized by two softmax models:

$$p_{\phi,\psi}(z = j|w) = \frac{\exp(\phi_w^T \psi_j)}{\sum_{i=1}^{K} \exp(\phi_w^T \psi_i)}$$

$$p_{\psi,\varphi}(c|z = j) = \frac{\exp(\psi_j^T \varphi_c)}{\sum_{c\prime \in N} \exp(\psi_j^T \varphi_{c\prime})}$$

Variational Inference

The goal is to maximize the log-likelihood of the observed edges

$$\sum_{(c,w)\in E} \log p_{\phi,\varphi,\psi}(c|w)$$

Directly optimizing this objective is intractable for large graphs, we instead optimize the following evidence lower bound

$$\mathcal{L} = E_{z\sim q(z|c,w)}(\log p_{\psi,\varphi}(c|z)) - KL(q(z|c,w)\|p_{\phi,\psi}(z|w))$$

where $q(z|c,w)$ is a variational distribution that approximates the true posterior distribution $p(z|c,w)$, and $KL(\cdot\|\cdot)$ represents the Kullback-Leibler divergence between two distributions.

Variational Inference

$$\mathcal{L} = E_{z \sim q(z|c,w)}(\log p_{\psi,\varphi}(c|z)) - KL(q(z|c,w) \| p_{\phi,\psi}(z|w))$$

$$p_{\phi,\psi}(z = j|w) = \frac{\exp(\phi_w^T \psi_j)}{\sum_{i=1}^{K} \exp(\phi_w^T \psi_i)}$$

$$p_{\psi,\varphi}(c|z = j) = \frac{\exp(\psi_j^T \varphi_c)}{\sum_{c' \in N} \exp(\psi_j^T \varphi_{c'})}$$

Specifically, we parameterize the variational distribution using a neural network as follows ($\odot$ denotes element-wise multiplication):

$$q_{\phi,\psi}(z = j|c, w) = \frac{\exp((\phi_w \odot \phi_c)^T \psi_j)}{\sum_{i=1}^{K} \exp((\phi_w \odot \phi_c)^T \psi_i)}$$

Straight-through Gumbel-Softmax estimator

It refactors the sampling operation into a deterministic function.

$$z = \underset{i}{argmax}\{G_i + log(\pi_i)\}$$

where $G_i \sim Gumbel(0,1)$. To pass gradients back, Straight-through estimator is used, which is basically using softmax to approximate the argmax operation during back propagation.

Community-smoothness Regularization

vGraph add the following regularization term to ensure that connected nodes tend to be in the same community:

$$\mathcal{L}_{reg} = \lambda \sum_{(w,c)\in E} \alpha_{w,c} \cdot d(p(z|c), p(z|w))$$

where λ is a tunable hyperparameter, $d(\cdot,\cdot)$ is the distance measure (squared difference in the experiment). $\alpha_{w,c}$ is given by:

$$\alpha_{w,c} = \frac{|N(w) \cap N(c)|}{|N(w) \cup N(c)|}$$

where $N(w)$ is the set of neighbors of w. The intuition behind this is that $\alpha_{w,c}$ serves as a similarity measure and the Jaccard's coefficient is used for this metric.

Experiment Setup

- 20 standard graph datasets.
- 3 tasks: overlaping community detection, non-overlaping community detection, and vertex classification.

Overlapping community detection

Table 2: Evaluation (in terms of F1-Score and Jaccard Similarity) on networks with overlapping ground-truth communities. NA means the task is not completed in 24 hours. In order to evaluate the effectiveness of smoothness regularization, we show the result of our model with (vGraph+) and without the regularization.

	F1-score						Jaccard					
Dataset	Bigclam	CESNA	Circles	SVI	vGraph	vGraph+	Bigclam	CESNA	Circles	SVI	vGraph	vGraph+
facebook0	**0.2948**	0.2806	0.2860	0.2810	0.2440	0.2606	**0.1846**	0.1725	0.1862	0.1760	0.1458	0.1594
facebook107	**0.3928**	0.3733	0.2467	0.2689	0.2817	0.3178	**0.2752**	0.2695	0.1547	0.1719	0.1827	0.2170
facebook1684	0.5041	**0.5121**	0.2894	0.3591	0.4232	0.4379	0.3801	**0.3871**	0.1871	0.2467	0.2917	0.3272
facebook1912	0.3493	0.3474	0.2617	0.2804	0.2579	**0.3750**	0.2412	0.2394	0.1672	0.2010	0.1855	**0.2796**
facebook3437	0.1986	0.2009	0.1009	0.1544	0.2087	**0.2267**	0.1148	0.1165	0.0545	0.0902	0.1201	**0.1328**
facebook348	0.4964	0.5375	0.5175	0.4607	**0.5539**	0.5314	0.3586	0.4001	0.3927	0.3360	**0.4099**	0.4050
facebook3980	0.3274	0.3574	0.3203	NA	**0.4450**	0.4150	0.2426	0.2645	0.2097	NA	**0.3376**	0.2933
facebook414	0.5886	0.6007	0.4843	0.3893	0.6471	**0.6693**	0.4713	0.4732	0.3418	0.2931	0.5184	**0.5587**
facebook686	0.3825	0.3900	0.5036	0.4639	0.4775	**0.5379**	0.2504	0.2534	0.3615	0.3394	0.3272	**0.3856**
facebook698	0.5423	0.5865	0.3515	0.4031	0.5396	**0.5950**	0.4192	0.4588	0.2255	0.3002	0.4356	**0.4771**
Youtube	0.4370	0.3840	0.3600	0.4140	0.5070	**0.5220**	0.2929	0.2416	0.2207	0.2867	0.3434	**0.3480**
Amazon	0.4640	0.4680	**0.5330**	0.4730	**0.5330**	0.5320	0.3505	0.3502	0.3671	0.3643	0.3689	**0.3693**
Dblp	0.2360	0.3590	NA	NA	0.3930	**0.3990**	0.1384	0.2226	NA	NA	0.2501	**0.2505**
Coauthor-CS	0.3830	0.4200	NA	0.4070	0.4980	**0.5020**	0.2409	0.2682	NA	0.2972	**0.3517**	0.3432

Non-overlapping community detection

Table 3: Evaluation (in terms of NMI and Modularity) on networks with non-overlapping ground-truth communities.

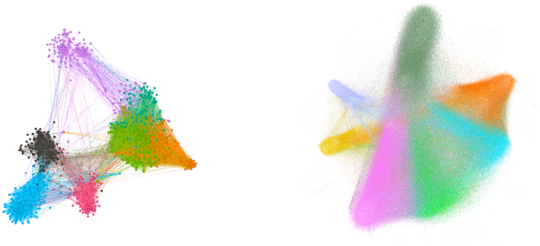
	NMI						Modularity					
Dataset	MF	deepwalk	LINE	node2vec	ComE	vGraph	MF	deepwalk	LINE	node2vec	ComE	vGraph
cornell	0.0632	0.0789	0.0697	0.0712	0.0732	**0.0803**	0.4220	0.4055	0.2372	0.4573	0.5748	**0.5792**
texas	0.0562	0.0684	**0.1289**	0.0655	0.0772	0.0809	0.2835	0.3443	0.1921	0.3926	**0.4856**	0.4636
washington	0.0599	0.0752	**0.0910**	0.0538	0.0504	0.0649	0.3679	0.1841	0.1655	0.4311	0.4862	**0.5169**
wisconsin	0.0530	0.0759	0.0680	0.0749	0.0689	**0.0852**	0.3892	0.3384	0.1651	0.5338	0.5500	**0.5706**
cora	0.2673	0.3387	0.2202	0.3157	**0.3660**	0.3445	0.6711	0.6398	0.4832	0.5392	0.7010	**0.7358**
citeseer	0.0552	0.1190	0.0340	0.1592	**0.2499**	0.1030	0.6963	0.6819	0.4014	0.4657	0.7324	**0.7711**

Vertex classification

Table 4: Results of node classification on 6 datasets.

	Macro-F1						Micro-F1					
Datasets	MF	DeepWalk	LINE	Node2Vec	ComE	vGraph	MF	DeepWalk	LINE	Node2Vec	ComE	vGraph
Cornell	13.05	22.69	21.78	20.70	19.86	**29.76**	15.25	33.05	23.73	24.58	25.42	**37.29**
Texas	8.74	21.32	16.33	14.95	15.46	**26.00**	14.03	40.35	27.19	25.44	33.33	**47.37**
Washington	15.88	18.45	13.99	21.23	15.80	**30.36**	15.94	34.06	25.36	28.99	33.33	**34.78**
Wisconsin	14.77	23.44	19.06	18.47	14.63	**29.91**	18.75	**38.75**	28.12	25.00	32.50	35.00
Cora	11.29	13.21	11.86	10.52	12.88	**16.23**	12.79	22.32	14.59	27.74	**28.04**	24.35
Citeseer	14.59	16.17	15.99	16.68	12.88	**17.88**	15.79	19.01	16.80	**20.82**	19.42	20.42

Visualization



(a) (b)

Figure 2: In panel (a) we visualize the result on the facebook107 dataset using vGraph. In panel (b) we visualize the result on Dblp-full dataset using vGraph. The coordinates of the nodes are determined by t-SNE of the node embeddings.

Conclusion

This paper presented a probabilistic method for jointly solving community detection and node representation learning. Experiments show that it performs well for both tasks.

Thank you!