The 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)
New Orleans, Louisiana, USA

GraphGAN: Graph Representation Learning with Generative Adversarial Nets

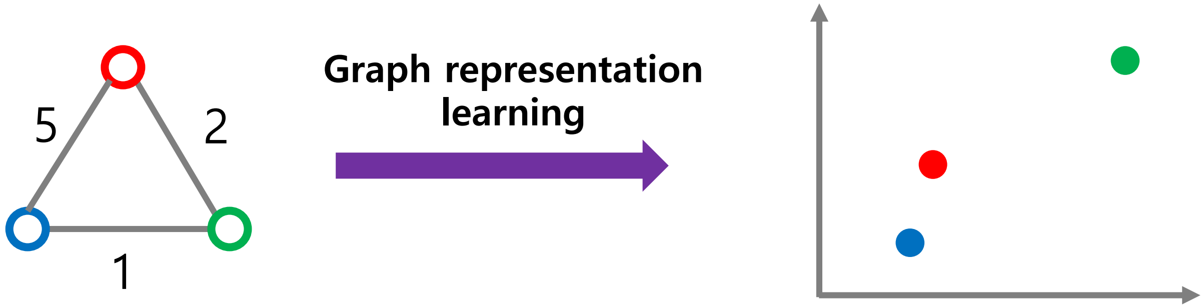
Hongwei Wang[1,2], Jia Wang[3], Jialin Wang[4], Miao Zhao[3],
Weinan Zhang[1], Fuzheng Zhang[2], Xing Xie[2], Minyi Guo[1]

[1] Shanghai Jiao Tong University, [2] Microsoft Research Asia,
[3] The Hong Kong Polytechnic University,
[4] Huazhong University of Science and Technology

February 04, 2018

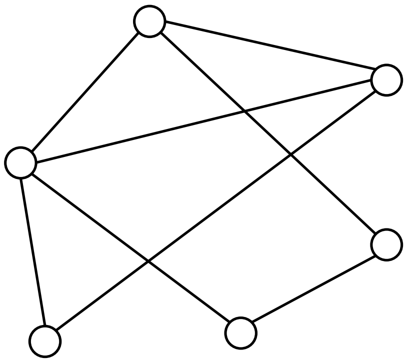
Background

- **Graph representation learning (GRL)** tries to embed each node of a graph into a low-dimensional vector space, while preserving the structural similarities among the nodes in the original graph
- a.k.a. **graph embedding / network embedding**



Motivation

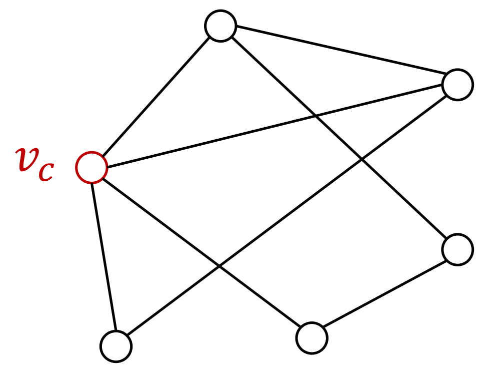
Generative Model



Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$

Motivation

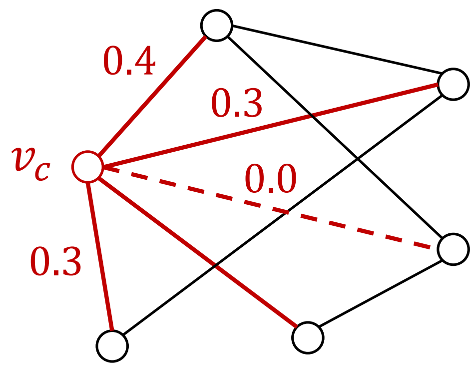
Generative Model



Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$

Motivation

Generative Model



Underlying true connectivity distribution: $p_{true}(v|v_c)$ for $v \in \mathcal{V} \backslash \{v_c\}$

Motivation

Generative Model

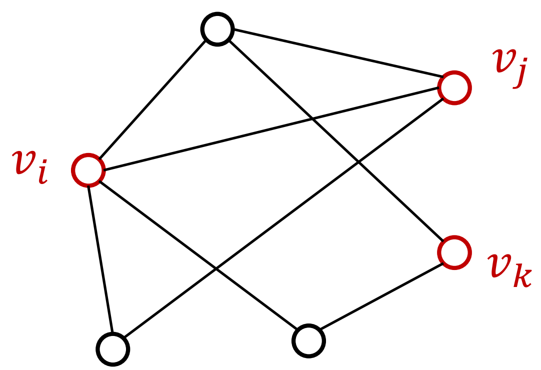
Maximize the likelihood of edges:

$$\max_{\Theta} p(v|v_c;\Theta) \text{ for } (v,v_c) \in \mathcal{E}$$

E.g., DeepWalk (KDD 2014) and node2vec (KDD 2016)

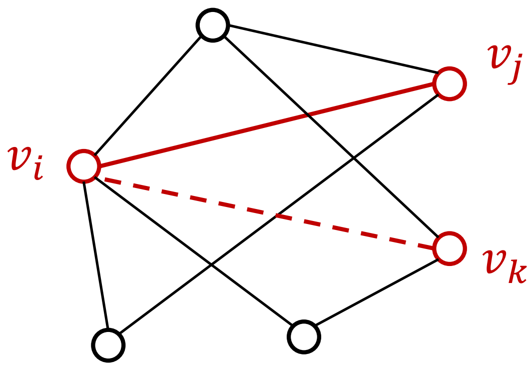
Motivation

Discriminative Model



Motivation

Discriminative Model



$$p(edge|v_i, v_j) = 0.8$$

$$p(edge|v_i, v_k) = 0.3$$

……

Motivation

Discriminative Model

Learn the classifier:

$$p(edge|v_i, v_j)$$

E.g., SDNE (KDD 2016) and PPNE (DASFAA, 2017)

Motivation

G + D ?

- Generative and discriminative models are two sides of the same coin
- Generative adversarial nets (GAN):
  - A game-theoretical minimax game to combine G and D
  - GAN applications:
    - image generation (Denton et al., NIPS 2015)
    - sequence generation (Yu et al., AAAI 2017)
    - dialogue generation (Li et al., arXiv 2017)
    - information retrieval (Wang et al., SIGIR 2017)
    - domain adaption (Zhang, Barzilay, and Jaakkola, arXiv 2017)
- **GraphGAN**: unifying generative and discriminative thinking for graph representation learning

The Minimax Game

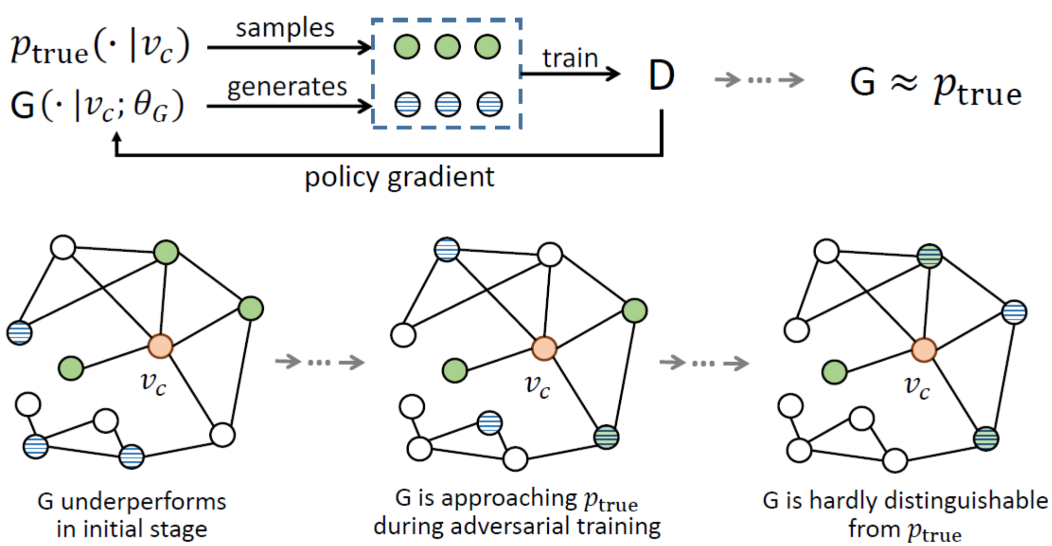
- The objective of GraphGAN is to learn the following two models:
  - $G(v|v_c;\theta_G)$: to approximate $p_{true}(v_c)$
  - $D(v,v_c;\theta_D)$: to judge the probability of edge existing between (v,v_c)

- The two-player minimax game:

$$\min_{\theta_G}\max_{\theta_D} V(G,D)=\sum_{c=1}^{V}\Big(\mathbb{E}_{v\sim p_{\text{true}}(\cdot|v_c)}\big[\log D(v,v_c;\theta_D)\big]+ \mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1-D(v,v_c;\theta_D)\big)\big]\Big)$$

GraphGAN Framework

$$\min_{\theta_G}\max_{\theta_D} V(G,D) = \sum_{c=1}^{V}\Big(\mathbb{E}_{v\sim p_{\mathrm{true}}(\cdot|v_c)}\big[\log D(v,v_c;\theta_D)\big]+ \\ \mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1-D(v,v_c;\theta_D)\big)\big]\Big)$$



Implementation & Optimization of D

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{V} \Big(\mathbb{E}_{v \sim p_{\text{true}}(\cdot|v_c)} \big[\log D(v, v_c; \theta_D) \big] + \mathbb{E}_{v \sim G(\cdot|v_c;\theta_G)} \big[\log \big(1 - D(v, v_c; \theta_D) \big) \big] \Big)$$

- Implementation of D:

$$D(v, v_c; \theta_D) = \sigma(\mathbf{d}_v^\top \mathbf{d}_{v_c}) = \frac{1}{1 + \exp(-\mathbf{d}_v^\top \mathbf{d}_{v_c})}$$

where $\mathbf{d}_v, \mathbf{d}_{v_c} \in \mathbb{R}^k$ are the k-dimensional vectors of v and v_c for D

- Gradient of $V(G, D)$ w.r.t θ_D:

$$\nabla_{\theta_D} V(G, D) = \begin{cases} \nabla_{\mathbf{d}_v, \mathbf{d}_{v_c}} \log D(v, v_c; \theta_D), & if\ v \sim p_{\text{true}}; \\ \nabla_{\mathbf{d}_v, \mathbf{d}_{v_c}} \big(1 - \log D(v, v_c; \theta_D)\big), & if\ v \sim G. \end{cases}$$

Optimization of G

$$\min_{\theta_G}\max_{\theta_D} V(G,D) = \sum_{c=1}^{V}\Big(\mathbb{E}_{v\sim p_{\text{true}}(\cdot|v_c)}\big[\log D(v,v_c;\theta_D)\big]+ \\ \mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1-D(v,v_c;\theta_D)\big)\big]\Big)$$

- Gradient of $V(G,D)$ w.r.t θ_G (policy gradient):

$$\begin{aligned}
&\nabla_{\theta_G} V(G,D) \\
=&\nabla_{\theta_G}\sum_{c=1}^{V}\mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1-D(v,v_c;\theta_D)\big)\big] \\
=&\sum_{c=1}^{V}\sum_{i=1}^{N}\nabla_{\theta_G} G(v_i|v_c;\theta_G)\log\big(1-D(v_i,v_c;\theta_D)\big) \\
=&\sum_{c=1}^{V}\sum_{i=1}^{N} G(v_i|v_c;\theta_G)\nabla_{\theta_G}\log G(v_i|v_c;\theta_G)\log\big(1-D(v_i,v_c;\theta_D)\big) \\
=&\sum_{c=1}^{V}\mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\underline{\nabla_{\theta_G}\log G(v|v_c;\theta_G)}\log\big(1-D(v,v_c;\theta_D)\big)\big].
\end{aligned}$$

Implementation of G

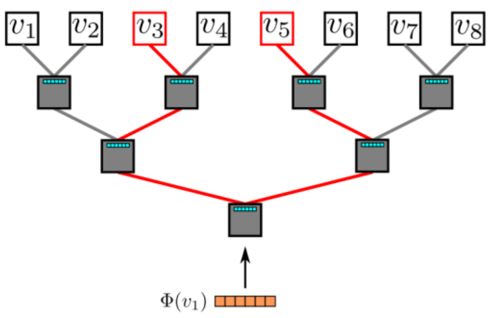
- Softmax?

$$G(v|v_c;\theta_G) = \frac{\exp(\mathbf{g}_v^\top \mathbf{g}_{v_c})}{\sum_{v \neq v_c} \exp(\mathbf{g}_v^\top \mathbf{g}_{v_c})}$$

where $\mathbf{g}_v, \mathbf{g}_{v_c} \in \mathbb{R}^k$ are the k-dimensional vectors of v and v_c for G

Computationally inefficient
Graph-structure-unaware

- Hierarchical softmax?



Graph-structure-unaware

- Negative sampling?

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

Graph-structure-unaware
Invalid distribution

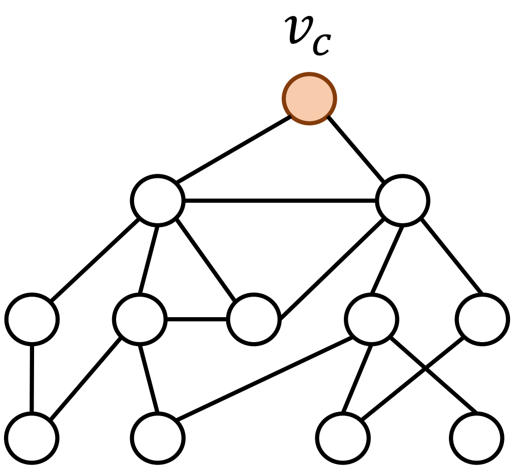
Graph Softmax

Design Objectives

- **Normalized**
  - The generator should produce a valid probability distribution, i.e., $\sum_{v \neq v_c} G(v|v_c;\theta_G) = 1$
- **Graph-structure-aware**
  - The generator should take advantage of the structural information of a graph
- **Computationally efficient**
  - The computation of $G(v|v_c;\theta_G)$ should only involve a small number of nodes in the graph

Graph Softmax

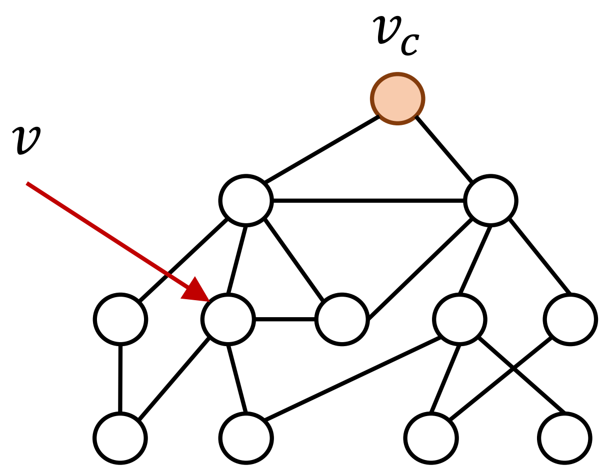
Design



Original graph $\mathcal{G}$

Graph Softmax

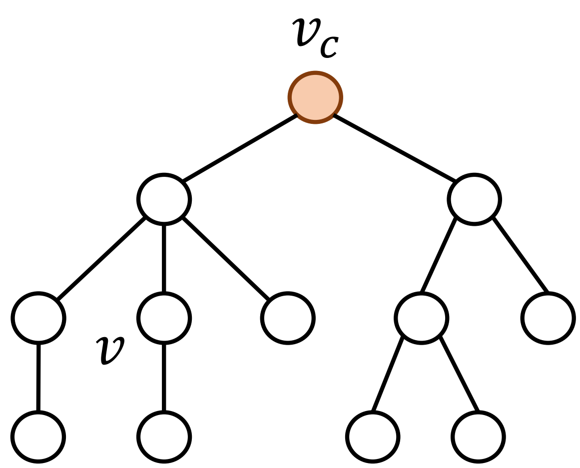
Design



Original graph $\mathcal{G}$

Graph Softmax

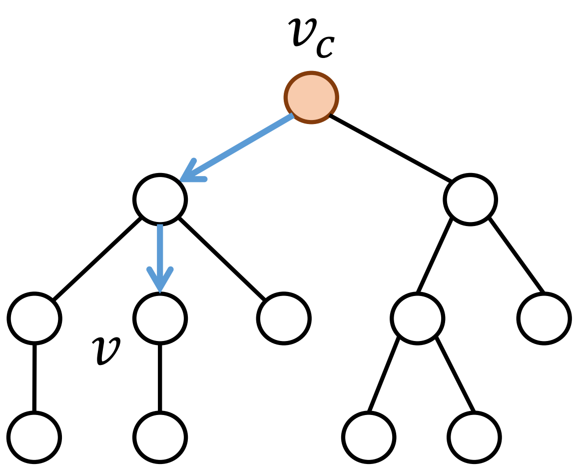
Design



Construct a BFS-tree T_c rooted at v_c

Graph Softmax

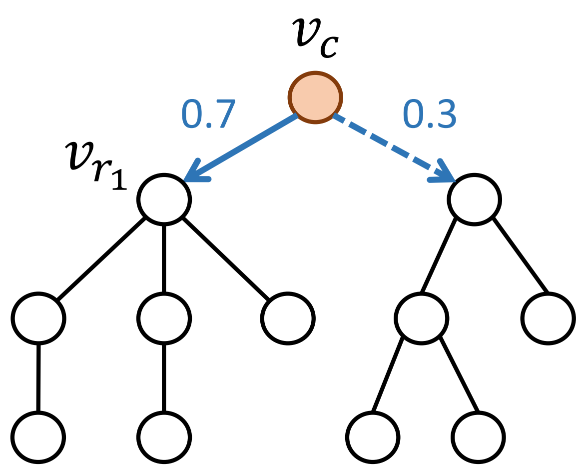
Design



Find the unique path from v_c to v in tree T_c

Graph Softmax

Design

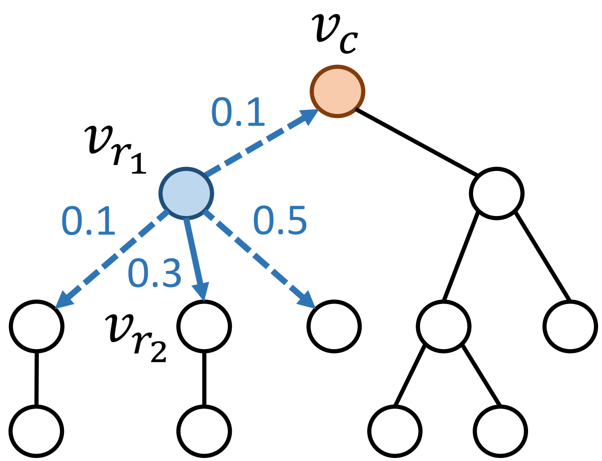


$$p_c(v_{r_1}|v_c) = 0.7$$

relevance probability: $p_c(v_i|v) = \frac{\exp(\mathbf{g}_{v_i}^\top \mathbf{g}_v)}{\sum_{v_j \in \mathcal{N}_c(v)} \exp(\mathbf{g}_{v_j}^\top \mathbf{g}_v)}$

Graph Softmax

Design

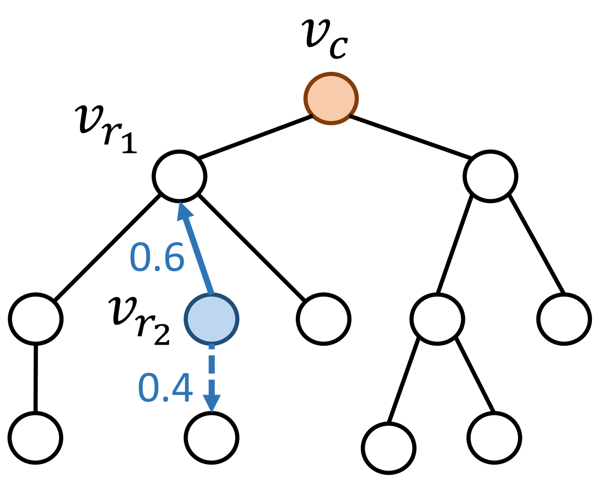


$$p_c(v_{r_1}|v_c) = 0.7$$

$$p_c(v_{r_2}|v_{r_1}) = 0.3$$

Graph Softmax

Design



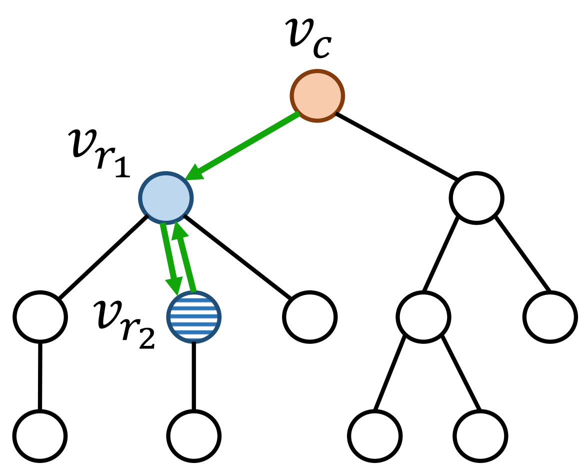
$$p_c(v_{r_1}|v_c) = 0.7$$

$$p_c(v_{r_2}|v_{r_1}) = 0.3$$

$$p_c(v_{r_1}|v_{r_2}) = 0.6$$

Graph Softmax

Design



$$G\left(v_{r_2} \middle| v_c; \theta_G\right) = 0.7 \times 0.3 \times 0.6 = 0.126$$

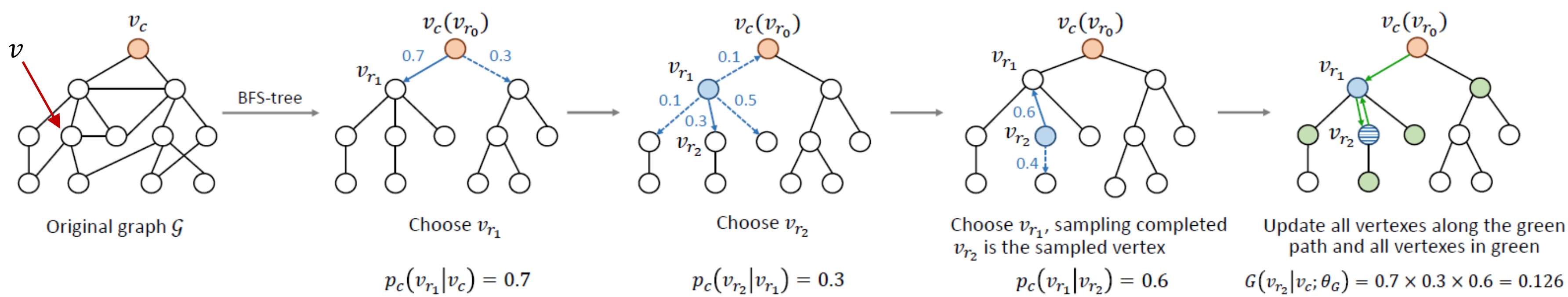
Graph Softmax

Design

❑ Graph softmax

$$G(v|v_c;\theta_G) \triangleq \left(\prod_{j=1}^{m} p_c(v_{r_j}|v_{r_{j-1}})\right) \cdot p_c(v_{r_{m-1}}|v_{r_m}),$$

given the unique path from v_c to v in tree T_c: $P_{v_c \to v} = (v_{r_0}, v_1, \ldots, v_{r_m})$, where $v_{r_0} = v_c$ and $v_{r_0} = v$



Graph Softmax

Properties

- ✓ **Normalized:** $\sum_{v \neq v_c} G(v|v_c; \theta_G) = 1$

- ✓ **Graph-structure-aware:** $G(v|v_c; \theta_G)$ decreases exponentially with the increase of the shortest distance between v and v_c in original graph $\mathcal{G}$

- ✓ **Computationally efficient:** Calculation of $G(v|v_c; \theta_G)$ depends on $O(d \log V)$ nodes, where d is average degree of nodes and V is the number of nodes in graph $\mathcal{G}$

Experiments

Datasets

- arXiv-AstroPh: 18,772 vertices and 198,110 edges
- arXiv-GrQc: 5,242 vertices and 14,496 edges
- BlogCatalog: 10,312 vertices, 333,982 edges and 39 labels
- Wikipedia: 4,777 vertices, 184,812 edges and 40 labels
- MovieLens-1M: 6,040 users and 3,706 movies

Baselines

- DeepWalk (KDD 2014)
- LINE (WWW 2015)
- Node2vec (KDD 2016)
- Struc2vec (KDD 2017)

Experiments

Link Prediction

TABLE 1: Accuracy and Macro-F1 on arXiv-AstroPh and arXiv-GrQc in link prediction.

Model	arXiv-AstroPh		arXiv-GrQc	
	Accuracy	Macro-F1	Accuracy	Macro-F1
DeepWalk	0.841	0.839	0.803	0.812
LINE	0.820	0.814	0.764	0.761
Node2vec	0.845	0.854	0.844	0.842
Struc2vec	0.821	0.810	0.780	0.776
GraphGAN	**0.855**	**0.859**	**0.849**	**0.853**

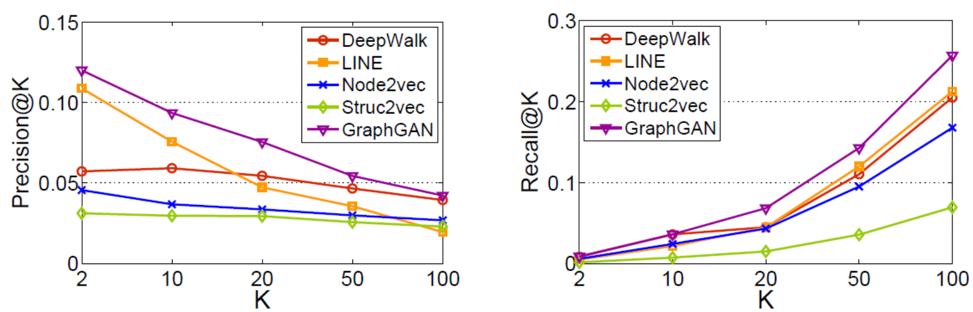
Node Classification

TABLE 2: Accuracy and Macro-F1 on BlogCatalog and Wikipedia in node classification.

Model	BlogCatalog		Wikipedia	
	Accuracy	Macro-F1	Accuracy	Macro-F1
DeepWalk	0.225	0.214	0.194	0.183
LINE	0.205	0.192	0.175	0.164
Node2vec	0.215	0.206	0.191	0.179
Struc2vec	0.228	0.216	0.211	0.190
GraphGAN	**0.232**	**0.221**	**0.213**	**0.194**

Experiments

Recommendation



(a) Precision@K (b) Recall@K

Fig. 5: Precision@K and Recall@K on MovieLens-1M in recommendation.

Summary

- We propose **GraphGAN**, a novel framework that unifies generative and discriminative models for graph representation learning

- G and D act as two players in a **minimax game**

- We propose **graph softmax** as the implementation of G

Q & A

Thanks!

Visit my homepage for the code and a more complete version of the slides: https://hwwang55.github.io