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

GraphGAN: Graph Representation Learning with Generative Adversarial Nets

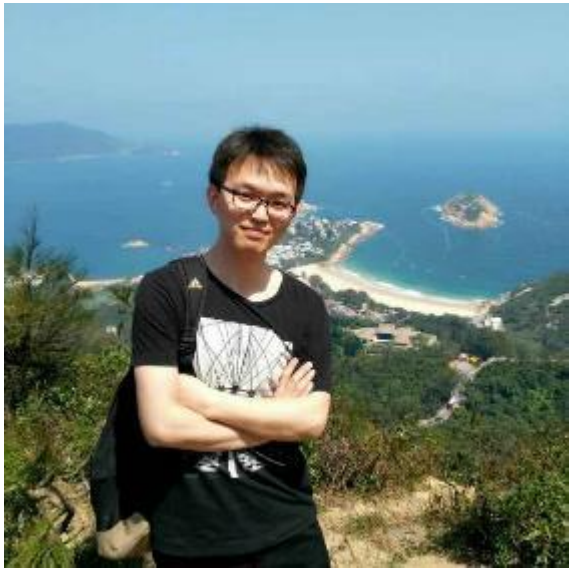
Hongwei Wang[1,2], Jia Wang[3], Jialin Wang[4], Miao Zhao[3],
Weinan Zhang[1], Fuzheng Zhang[2], Xing Xie[2], Minyi Guo[1]

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

January 10, 2018

About Me

- **Hongwei Wang (王鸿伟)**
  - Ph.D. candidate at Shanghai Jiao Tong University
  - Full-time intern at Microsoft Research Asia
- Advisor
  - Prof. Minyi Guo (SJTU)
  - Dr. Xing Xie and Dr. Baining Guo (MSRA)
- Research interests
  - Recommender systems [TCSS 2017] [RecSys 2017] [CIKM 2017] [WWW 2018]
  - Graph representation learning [WSDM 2018] [AAAI 2018a]
  - Machine learning applications [ICDCS 2017] [TPDS 2018] [AAAI 2018b]
- Homepage: https://hwwang55.github.io



Outline

- **Introduction to graph representation learning**
  - Definition and application
  - Taxonomy
  - Representative work
- **GraphGAN** [AAAI 2018]
- **GRL application**
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]

1. A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications, TKDE 2017
2. https://github.com/thunlp/NRLPapers
3. https://hwwang55.github.io/files/a_literature_review_on_rs_and_nrl.pdf (in Chinese)

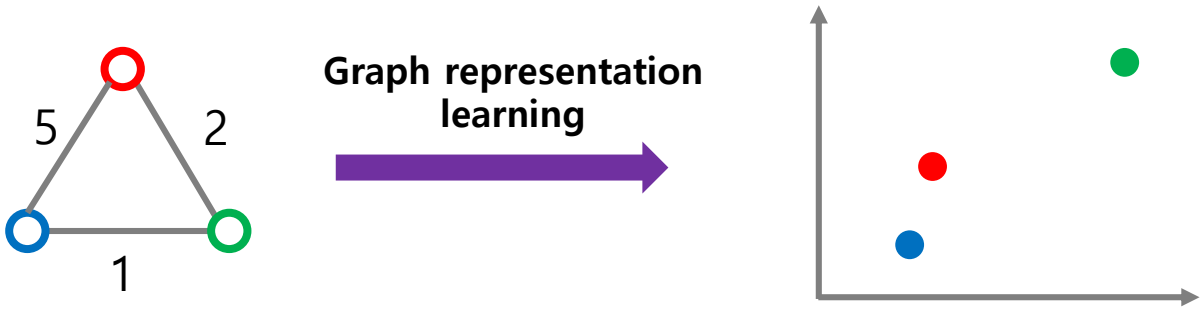
Outline

- **Introduction to graph representation learning**
  - Definition and application
  - Taxonomy
  - Representative work
- **GraphGAN** [AAAI 2018]
- **GRL application**
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]

Definition

- **Graph representation learning** tries to embed each node of a graph into a low-dimensional vector space, which preserves the structural similarities or distances among the nodes in the original graph

- Also known as **network embedding** / **graph embedding** / **network representation learning**



Application

- **Graph representation learning** can benefit a wide range of real-world applications:
  - Link prediction (Gao, Denoyer, and Gallinari, CIKM 2011)
  - Node classification (Tang, Aggarwal, and Liu, SDM 2016)
  - Recommendation (Yu et al., WSDM 2014)
  - Visualization (Maaten and Hinton, JMLR 2008)
  - Knowledge graph representation (Lin et al., AAAI 2015)
  - Clustering (Tian et al., AAAI 2014)
  - Text embedding (Tang, Qu, and Mei, KDD 2015)
  - Social network analysis (Liu et al., IJCAI 2016)

Taxonomy (1/3)

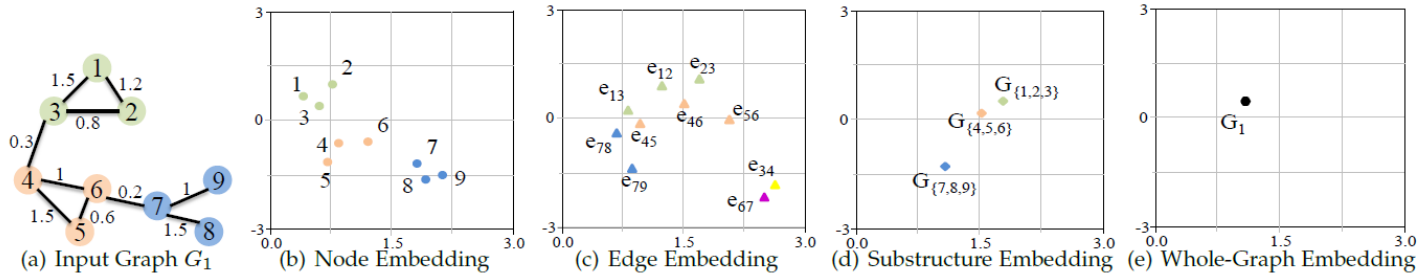
Input

- Homogeneous graph (e.g., citation network)
  - Weighted / Unweighted
  - Directed / Undirected
  - Signed / Unsigned
- Heterogeneous graph
  - Multimedia network
  - Knowledge graph
- Graph with side information
  - Node/edge label (categorical)
  - Node/edge attribute (discrete or continuous)
  - Node feature (e.g., texts)
- Graph transformed from non-relational data
  - Manifold learning

Taxonomy (2/3)

Output

- Node embedding (the most common case)
- Edge embedding
  - Relations in knowledge graph
  - Link prediction
- Sub-graph embedding
  - Substructure embedding
  - Community embedding
- Whole-graph embedding
  - Multiple small graphs, e.g., molecule, protein

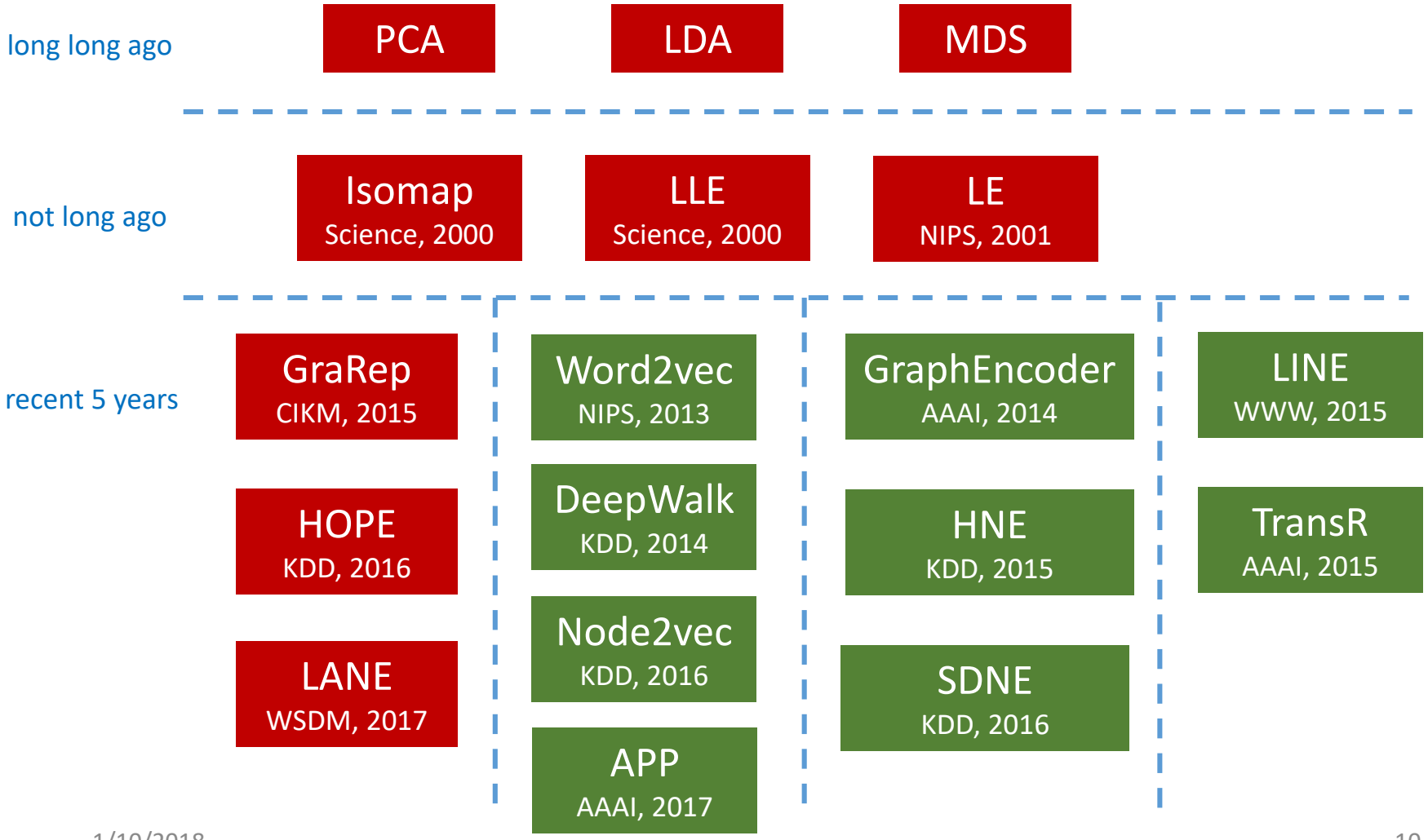


Taxonomy (3/3)

Method

- Matrix factorization
  - SVD
  - Spectral Analysis
- Random walk
- Deep learning
  - Auto-encoder
  - Convolutional neural network
- Self-defined loss
  - Maximizing edge reconstruction probability
  - Minimizing distance-based loss
  - Minimizing margin-based ranking loss

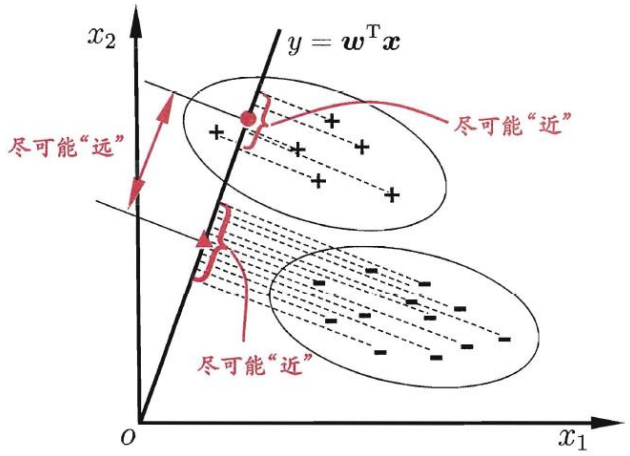
Representative Work



Linear Discriminant Analysis

- Suppose binary classification
- $D = \{(x_i, y_i)\}$, μ_i: mean of data of the *i*-th class, Σ_i: covariance matrix of data of the *i*-th class
- Make projected covariance matrix as small as possible, while make projected distance between the mean of two classes as large as possible
- maximize

$$J = \frac{\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\|_2^2}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}} = \frac{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\boldsymbol{w}}$$

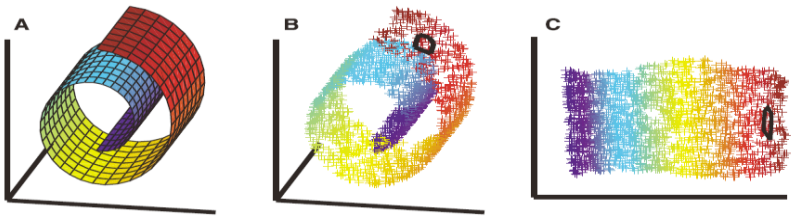


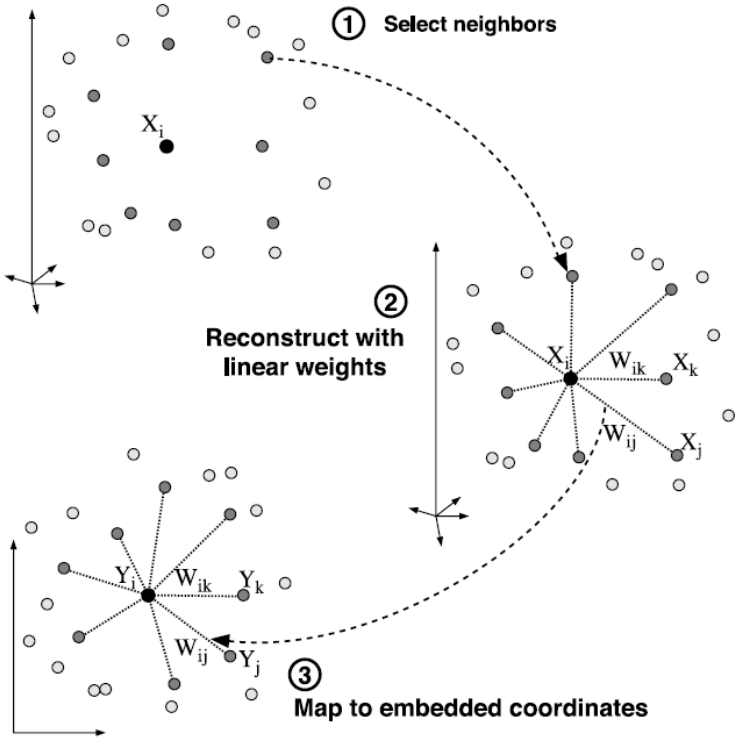
《机器学习》，周志华

Locally Linear Embedding

Introduction

- An unsupervised learning algorithm that computes low-dimensional, **neighborhood-preserving** embeddings of high-dimensional inputs
- LLE keeps linear dependency between local instances





Word2vec

- **Skip-Gram Model**
  - Find word representations that are useful for **predicting the surrounding words** in a sentence
  - Maximizing

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

where

$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top}v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_{w}}^{\top}v_{w_I}\right)}$$

  - Negative sampling:

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

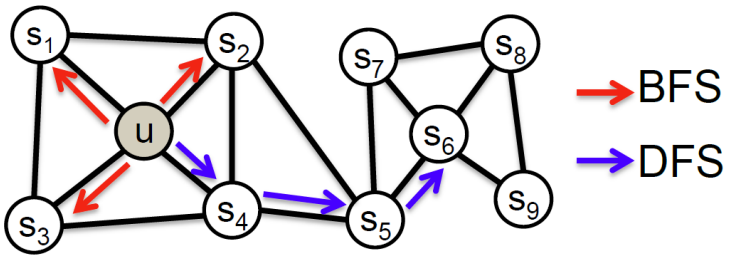
DeepWalk / Node2vec

- **DeepWalk: Random walk + Word2vec**
  - Sample the next node to be visited **uniformly** from the neighbors of current node

$$\underset{\Phi}{\text{minimize}} \quad -\log \Pr\left(\{v_{i-w}, \cdots, v_{i+w}\} \setminus v_i \mid \Phi(v_i)\right)$$

  - Hierarchical softmax

- **Node2vec: Biased random walk + Word2vec**
  - Sample the next node to be visited **with bias** from the neighbors of current node



LINE

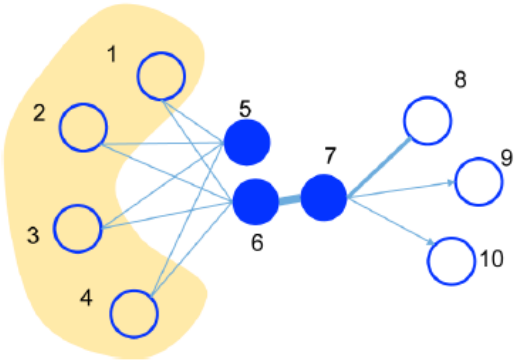
- **First-order proximity**

$$O_1 = d(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot))$$

$$p_1(v_i, v_j) = \frac{1}{1+\exp(-\vec{u}_i^T \cdot \vec{u}_j)}$$

$$\hat{p}_1(i,j) = \frac{w_{ij}}{W}$$

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$



- **Second-order proximity**

$$O_2 = \sum_{i\in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i))$$

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}_k'^T \cdot \vec{u}_i)}$$

$$\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{d_i}$$

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log p_2(v_j|v_i)$$

TransX

- Embed **knowledge graph** into a continuous vector space while preserving structural information

- TransE (NIPS 13):
  - Ensures $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when (h, r, t) holds

- TransH (AAAI 14):
  - Ensures $\mathbf{h}_\perp + \mathbf{r} \approx \mathbf{t}_\perp$ when (h, r, t) holds, where $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^{\mathrm{T}}\mathbf{h}\mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^{\mathrm{T}}\mathbf{t}\mathbf{w}_r$

- TransR (AAAI 15):
  - Score function: $f\_r\ (h,\ t)$=‖$\mathbf{h}\mathbf{M}\_r$+$\boldsymbol{r}$−$\mathbf{t}\mathbf{M}\_r$ ‖_2^2, where $\mathbf{M}\_r$ is the projection matrix for relation r.

Knowledge Graph Embedding: A Survey of Approaches and Applications, TDKE 2017

SDNE

- **Reconstruction loss term**

$$\mathcal{L}_{2nd} = \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b}_i\|_2^2$$
$$= \|(\hat{X} - X) \odot B\|_F^2$$

- **Proximity loss term**

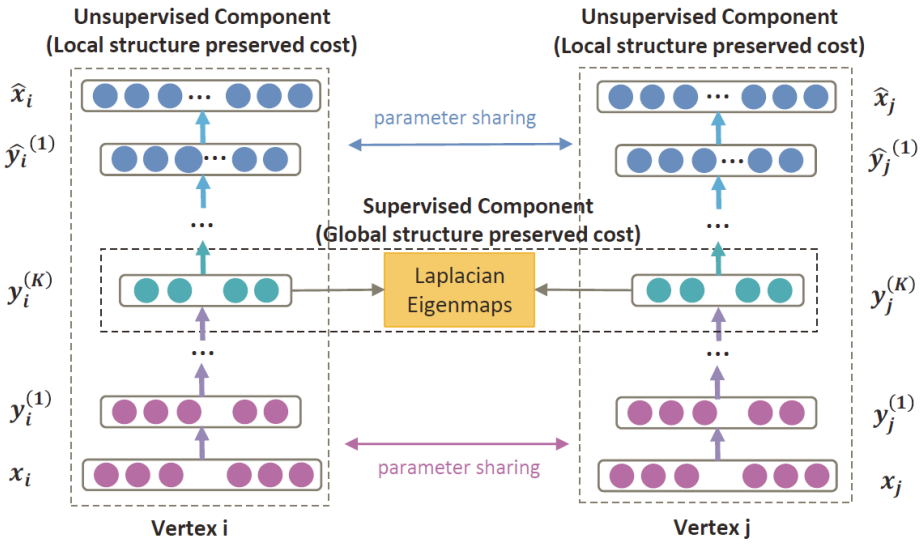
$$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)}\|_2^2$$
$$= \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2$$

- **Regularization term**

$$\mathcal{L}_{reg} = \frac{1}{2} \sum_{k=1}^{K} (\|W^{(k)}\|_F^2 + \|\hat{W}^{(k)}\|_F^2)$$

- **Loss function**

$$\mathcal{L}_{mix} = \mathcal{L}_{2nd} + \alpha \mathcal{L}_{1st} + \nu \mathcal{L}_{reg}$$



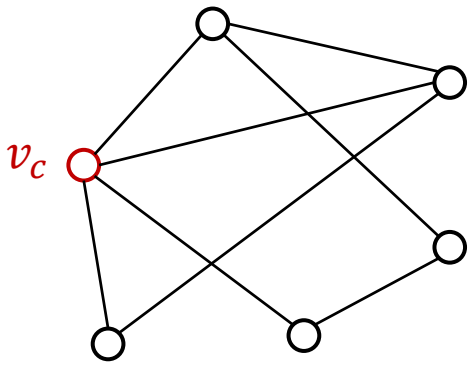
Outline

- Introduction to graph representation learning
  - Definition and application
  - Taxonomy
  - Representative work
- **GraphGAN** [AAAI 2018]
- GRL application
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]

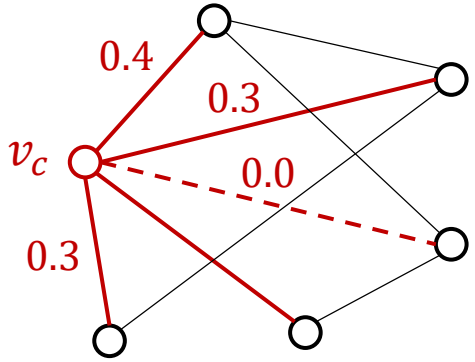
Motivation (1/3)

Generative Model

- Generative graph representation learning model assumes an **underlying true connectivity distribution** $p_{true}(v|v_c)$ for each vertex v_c
  - Similar to GMM and LDA
  - The edges can be viewed as observed samples generated by $p_{true}(v|v_c)$
  - Vertex embeddings are learned by maximizing the likelihood of edges
  - E.g., DeepWalk (KDD 2014) and node2vec (KDD 2016)



Original graph

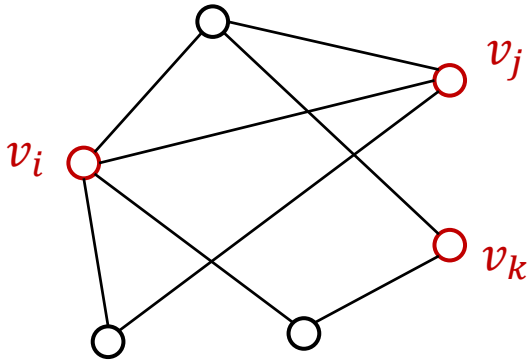


$p_{true}(v|v_c)$

Motivation (2/3)

Discriminative Model

- Discriminative graph representation learning model aim to learn a **classifier** for predicting the existence of edges directly
  - Consider two vertices v_i and v_j jointly as features
  - Predict the probability of an edge existing between them, i.e., $p(edge|v_i, v_j)$
  - E.g., SDNE (KDD 2016) and PPNE (DASFAA, 2017)



$$p(edge|v_i, v_j) = 0.8$$
$$p(edge|v_i, v_k) = 0.3$$
……

Motivation (3/3)

G + D ?

- Generative and discriminative models are two sides of the same coin
- LINE (WWW 2015) has tried to combine these two objectives
- Generative adversarial nets (GAN) have received a great deal of attention
  - GAN designs a game-theoretical minimax game to combine G and D
  - GAN achieves success in various 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)

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

The Minimax Game

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{v_1, \dots, v_V\}$, $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{V}$
- $\mathcal{N}(v_c)$: set of neighbors of v_c
- $p_{true}(v_c)$: underlying true connectivity distribution for v_c

- The objective of GraphGAN is to learn the following two models:
  - $G(v|v_c; \theta_G)$ which tries to approximate $p_{true}(v_c)$
  - $D(v, v_c; \theta_D)$ which aims to discriminate the connectivity for the vertex pair (v, v_c)

- The two-player minimax game:

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

Implementation & Optimization of D

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

- Implementation of D:

$$D(v,v_c;\theta_D) = \sigma(\mathbf{d}_v^\top \mathbf{d}_{v_c}) = \frac{1}{1+\exp(-\mathbf{d}_v^\top \mathbf{d}_{v_c})}, \tag{2}$$

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:

$$\nabla_{\theta_D} V(G,D) = \begin{cases} \nabla_{\mathbf{d}_v,\mathbf{d}_{v_c}} \log D(v,v_c;\theta_D), & if\ v\sim p_{\text{true}}; \\ \nabla_{\mathbf{d}_v,\mathbf{d}_{v_c}} \big(1-\log D(v,v_c;\theta_D)\big), & if\ v\sim G. \end{cases} \tag{3}$$

Optimization of G

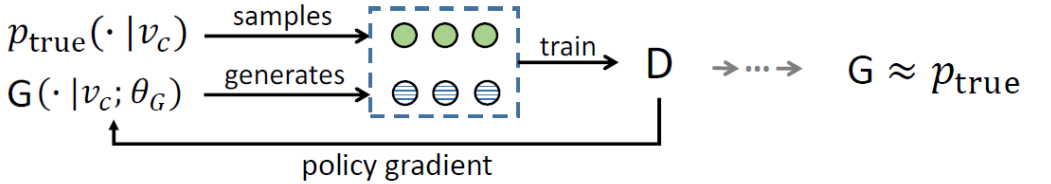
$$\min_{\theta_G}\max_{\theta_D} V(G,D) = \sum_{c=1}^{V}\Big(\mathbb{E}_{v\sim p_{\text{true}}(\cdot|v_c)}\big[\log D(v,v_c;\theta_D)\big] + \mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1-D(v,v_c;\theta_D)\big)\big]\Big) \tag{1}$$

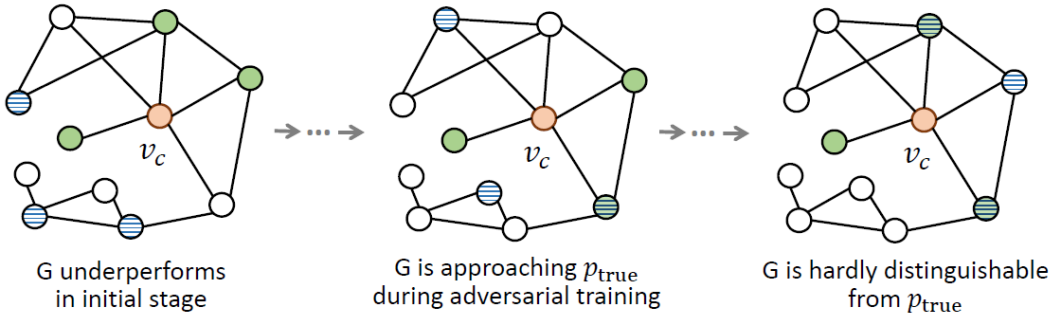
- Gradient of $V(G,D)$ w.r.t θ_G (policy gradient):

$$\begin{aligned}
&\nabla_{\theta_G} V(G,D)\\
=&\nabla_{\theta_G}\sum_{c=1}^{V}\mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1-D(v,v_c;\theta_D)\big)\big]\\
=&\sum_{c=1}^{V}\sum_{i=1}^{N}\nabla_{\theta_G} G(v_i|v_c;\theta_G)\log\big(1-D(v_i,v_c;\theta_D)\big)\\
=&\sum_{c=1}^{V}\sum_{i=1}^{N} G(v_i|v_c;\theta_G)\nabla_{\theta_G}\log G(v_i|v_c;\theta_G)\log\big(1-D(v_i,v_c;\theta_D)\big)\\
=&\sum_{c=1}^{V}\mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\nabla_{\theta_G}\log G(v|v_c;\theta_G)\log\big(1-D(v,v_c;\theta_D)\big)\big].
\end{aligned} \tag{4}$$

GraphGAN Framework

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





G underperforms in initial stage

G is approaching p_{true} during adversarial training

G is hardly distinguishable from p_{true}

Implementation of G

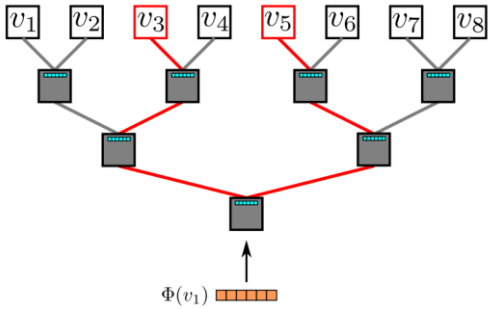
- Softmax?
  - Computationally inefficient
  - Graph-structure-unaware

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

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

- Hierarchical softmax?
  - Graph-structure-unaware

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left(\llbracket n(w,j+1) = \mathrm{ch}(n(w,j)) \rrbracket \cdot {v'_{n(w,j)}}^\top v_{w_I}\right)$$



- Negative sampling?
  - Not a valid probability distribution
  - graph-structure-unaware

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

Graph Softmax (1/5)

Objectives

- The design of graph softmax should satisfy the following three properties:
  - **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 vertices in the graph

Graph Softmax (2/5)

Design

- **Breadth First Search** (BFS) on $\mathcal{G}$ from every vertex v_c
  - **BFS-tree** T_c rooted at v_c
- For a given vertex v and one of its neighbors $v_i \in \mathcal{N}_c(v)$, the **relevance probability** of v_i given v as

$$p_c(v_i|v) = \frac{\exp(\mathbf{g}_{v_i}^\top \mathbf{g}_v)}{\sum_{v_j \in \mathcal{N}_c(v)} \exp(\mathbf{g}_{v_j}^\top \mathbf{g}_v)}, \qquad (6)$$

  where $\mathcal{N}_c(v)$ is the set of neighbors of v in T_c

- **Graph softmax**

$$G(v|v_c;\theta_G) \triangleq \left(\prod_{j=1}^{m} p_c(v_{r_j}|v_{r_{j-1}})\right) \cdot p_c(v_{r_{m-1}}|v_{r_m}), \qquad (7)$$

  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 $v_{r_0} = v$

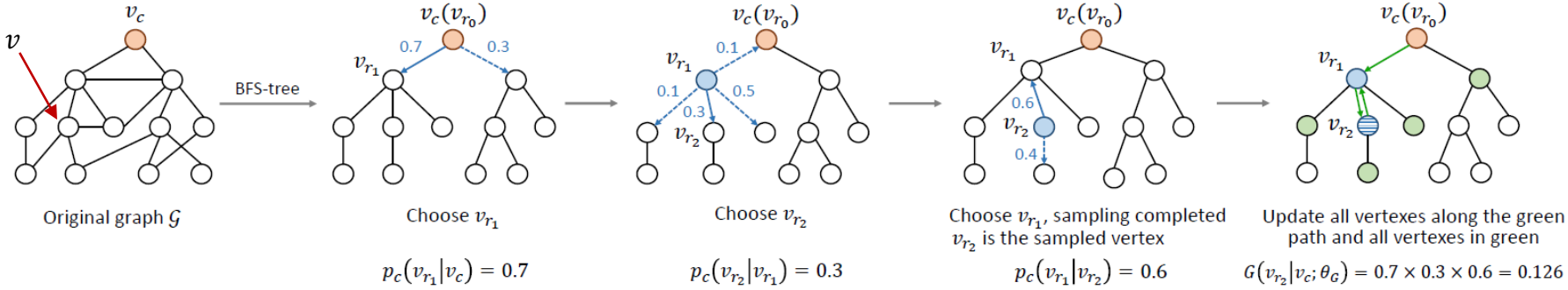
Graph Softmax (3/5)

Design

- **Graph softmax**

$$G(v|v_c;\theta_G) \triangleq \left(\prod_{j=1}^{m} p_c(v_{r_j}|v_{r_{j-1}})\right) \cdot p_c(v_{r_{m-1}}|v_{r_m}), \qquad (7)$$

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 (4/5)

Properties

- $\sum_{v \neq v_c} G(v|v_c;\theta_G) = 1$ in graph softmax

- In graph softmax, $G(v|v_c;\theta_G)$ decreases exponentially with the increase of the shortest distance between v and v_c in original graph $\mathcal{G}$

- In graph softmax, calculation of $G(v|v_c;\theta_G)$ depends on $O(d \log V)$ vertices, where d is average degree of vertices and V is the number of vertices in graph $\mathcal{G}$

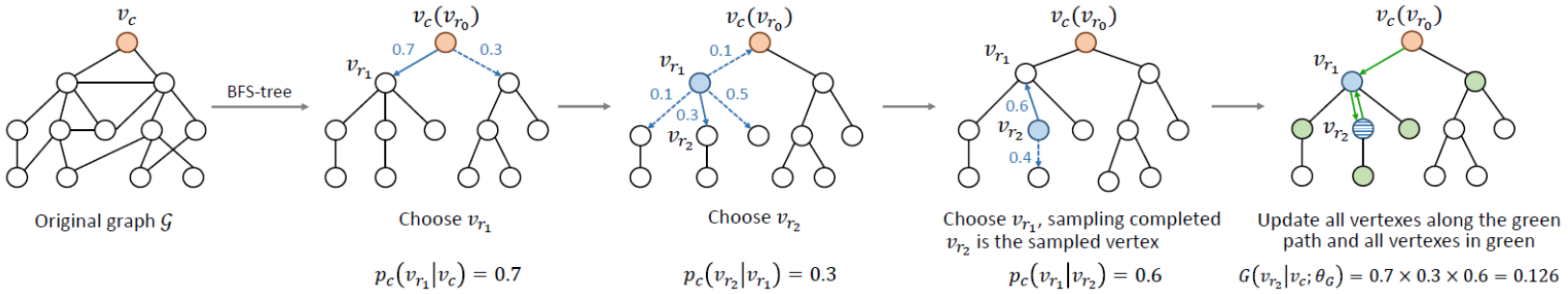
Graph Softmax (5/5)

Generating Strategy

Algorithm 1 Online generating strategy for the generator

Input: BFS-tree T_c, representation vectors $\{\mathbf{g}_i\}_{i\in\mathcal{V}}$
Output: generated sample v_{gen}

```
v_pre ← v_c, v_cur ← v_c;
while true do
    Randomly select v_i proportionally to p_c(v_i|v_cur) in Eq. (6);
    if v_i = v_pre then
        v_gen ← v_cur;
        return v_gen
    else
        v_pre ← v_cur, v_cur ← v_i;
    end if
end while
```



GraphGAN Algorithm

Algorithm 2 GraphGAN framework

Input: dimension of embedding k, size of generating samples s, size of discriminating samples t
Output: generator $G(v|v_c;\theta_G)$, discriminator $D(v,v_c;\theta_D)$

Initialize and pre-train $G(v|v_c;\theta_G)$ and $D(v,v_c;\theta_D)$;
Construct BFS-tree T_c for all $v_c \in \mathcal{V}$;
while GraphGAN not converge **do**
  for G-steps **do**
    $G(v|v_c;\theta_G)$ generates s vertices for each vertex v_c according to Algorithm 1;
    Update θ_G according to Eq. (4), (6) and (7);
  end for
  for D-steps **do**
    Sample t positive vertices from ground truth and t negative vertices from $G(v|v_c;\theta_G)$ for each vertex v_c;
    Update θ_D according to Eq. (2) and (3);
  end for
end while
return $G(v|v_c;\theta_G)$ and $D(v,v_c;\theta_D)$

Experiments (1/3)

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

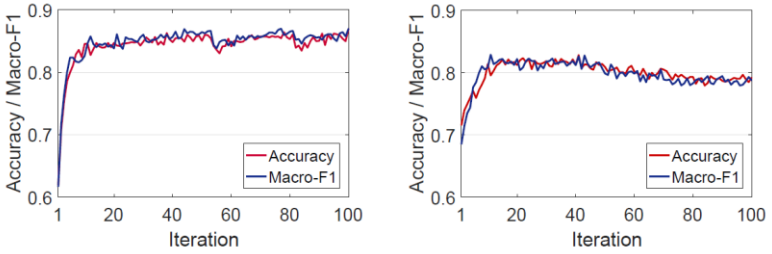
Baselines

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

Experiments (2/3)

Link Prediction

- Learning curves



(a) Generator (b) Discriminator

Fig. 4: Learning curves of the generator and the discriminator of GraphGAN on arXiv-GrQc in link prediction.

- Results

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**

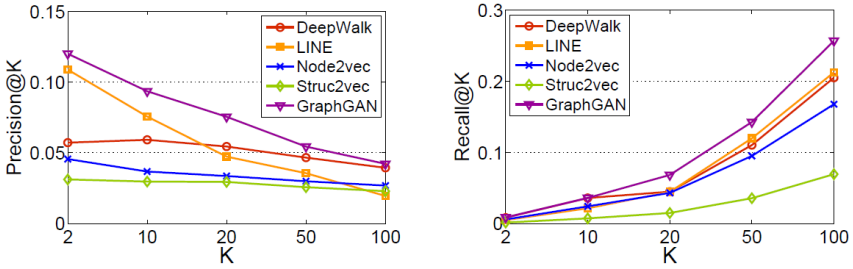
Experiments (3/3)

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**

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 thinking for graph representation learning
  - Generator $G(v|v_c)$ tries to fit $p_{true}(v|v_c)$ as much as possible
  - Discriminator $D(v, v_c)$ tries to tell whether an edge exists between v and v_c
- G and D act as two players in a **minimax game**:
  - G tries to produce the most indistinguishable "fake" vertices under guidance provided by D
  - D tries to draw a clear line between the ground truth and "counterfeits" to avoid being fooled by G
- We propose **graph softmax** as the implementation of G
  - Graph softmax overcomes the limitations of softmax and hierarchical softmax
  - Graph softmax satisfies the properties of **normalization**, **graph structure awareness** and **computational efficiency**

Outline

- **Introduction to graph representation learning**
  - Definition and application
  - Taxonomy
  - Representative work
- **GraphGAN** [AAAI 2018]
- **GRL application**
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]

DKN (WWW 2018)

- **DKN: Deep Knowledge-Aware Network for News Recommendation**
  - Learning knowledge graph representations by TransX
  - A CNN framework for combining word embedding and entity embedding
  - Attention-based CTR prediction

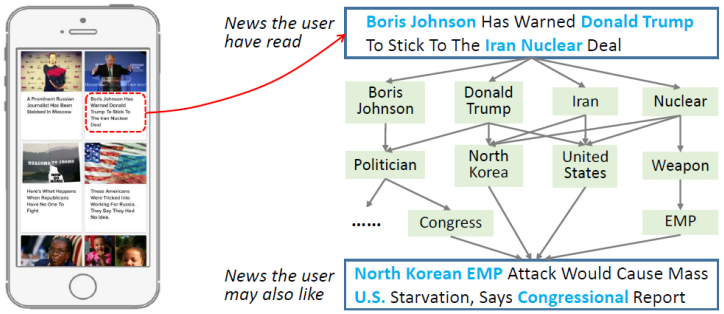


Fig. 1: Knowledge graph in news recommendation

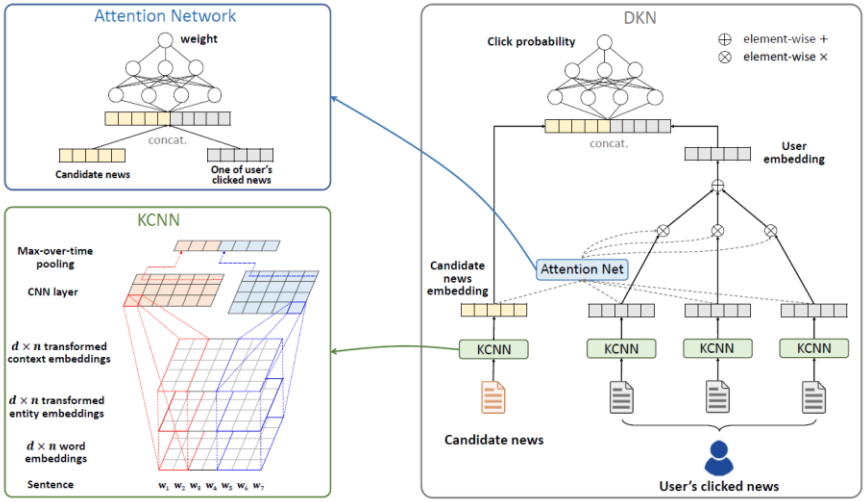


Fig. 2: DKN framework

SHINE (WSDM 2018)

- **SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction**
  - **Sign**: positive/negative sentiment link
  - **Heterogeneity**: sentiment network, social network, knowledge graph
  - Auto-encoder based framework

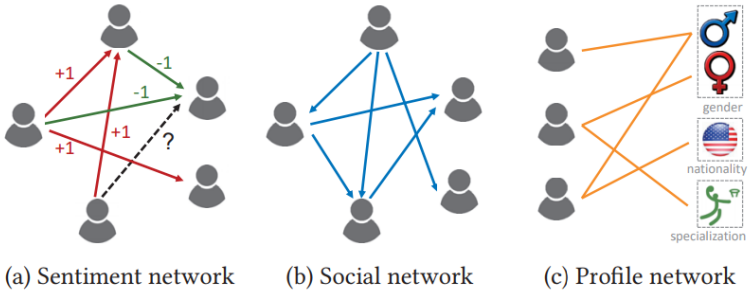


Fig. 1: Signed heterogeneous networks in sentiment prediction

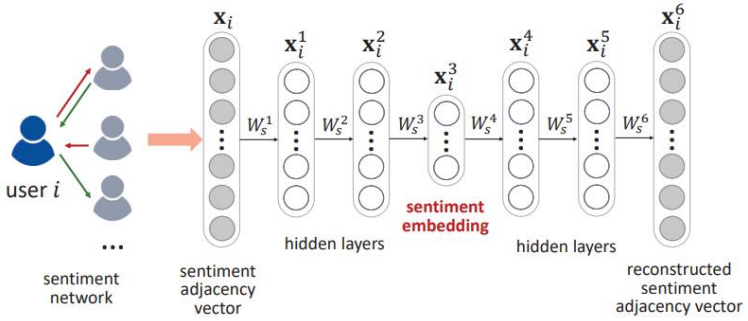


Fig. 2: Auto-encoder for sentiment network representation learning

SHINE (WSDM 2018)

- **SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction**
  - **Sign**: positive/negative sentiment link
  - **Heterogeneity**: sentiment network, social network, knowledge graph
  - Auto-encoder based framework

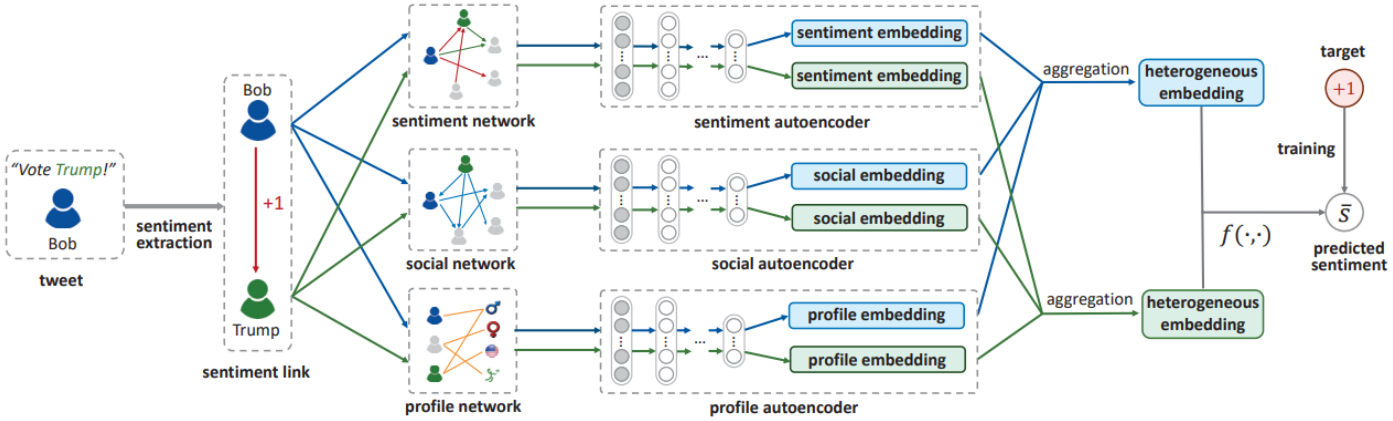


Fig. 3: SHINE framework

Q & A

Thanks!

Visit https://hwwang55.github.io for full papers, slides, code and more information