Deep Learning Applications

Graph Representation Learning

U Kang
Seoul National University

In This Lecture

- Motivation of graph representation learning
- Deepwalk
- Node2vec
- SIDE

Outline

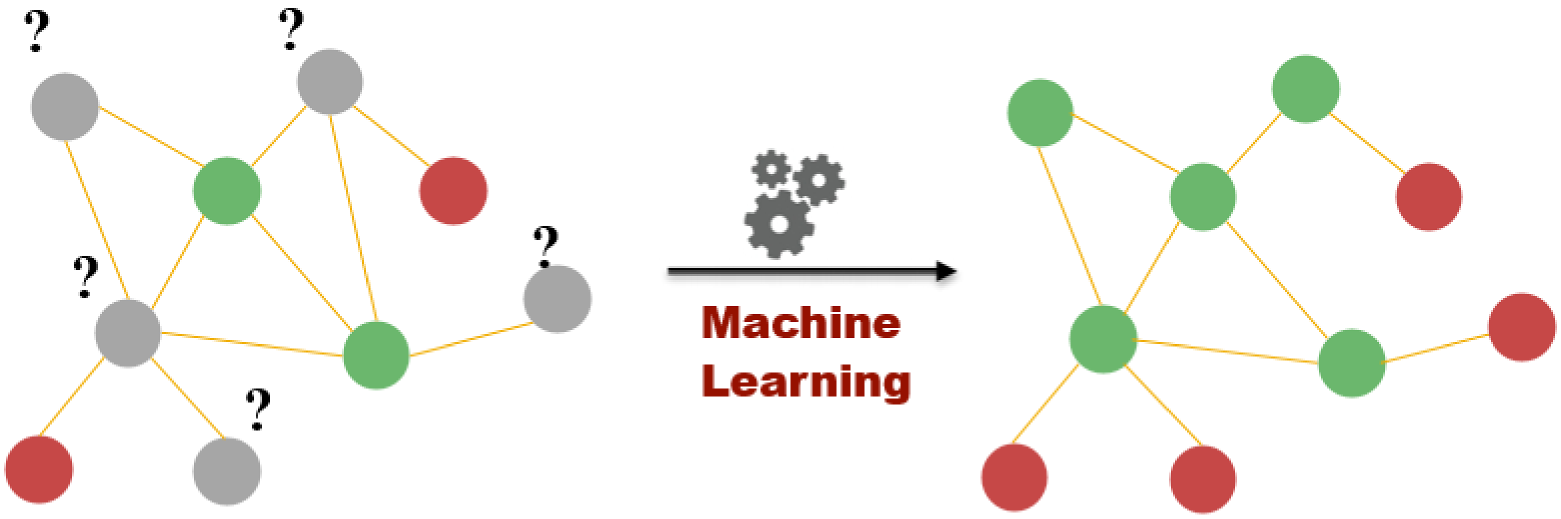
➡ □ **Motivation**

□ DeepWalk

□ Node2vec

□ SIDE

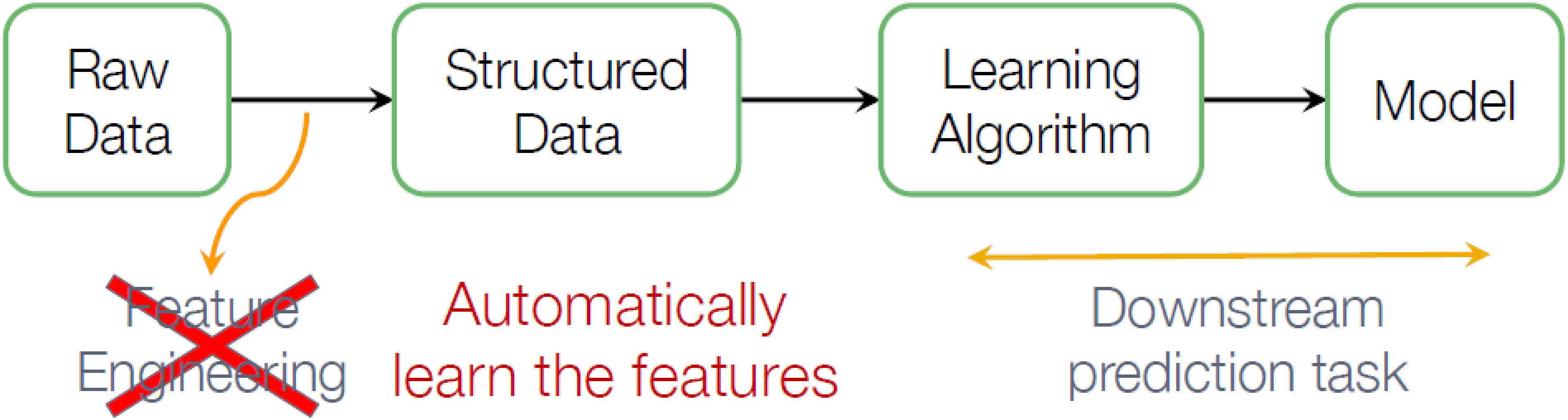
Machine Learning in Networks



Node classification

Machine Learning Lifecycle

- **(Supervised) Machine Learning Lifecycle: This feature, that feature. Every single time!**



Raw Data

Structured Data

Learning Algorithm

Model

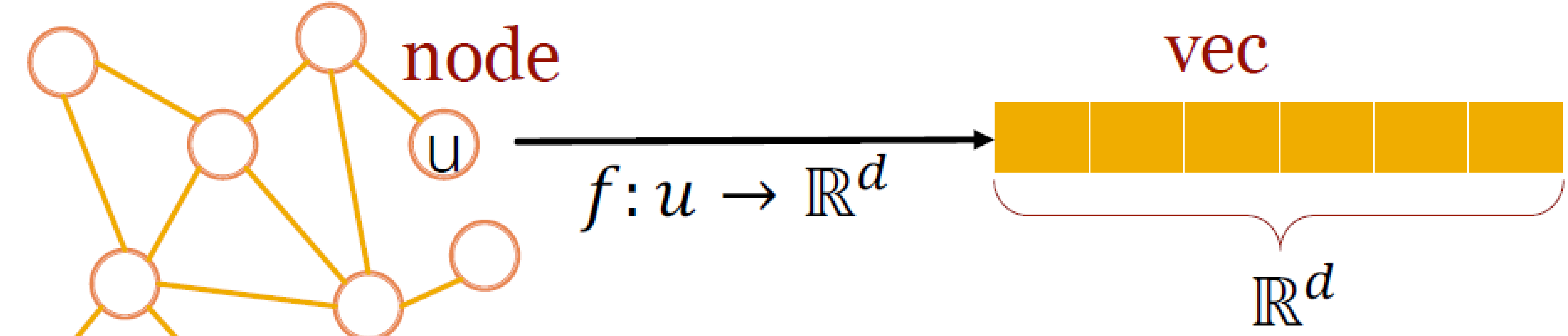
~~Feature Engineering~~

Automatically learn the features

Downstream prediction task

Feature Learning in Graph

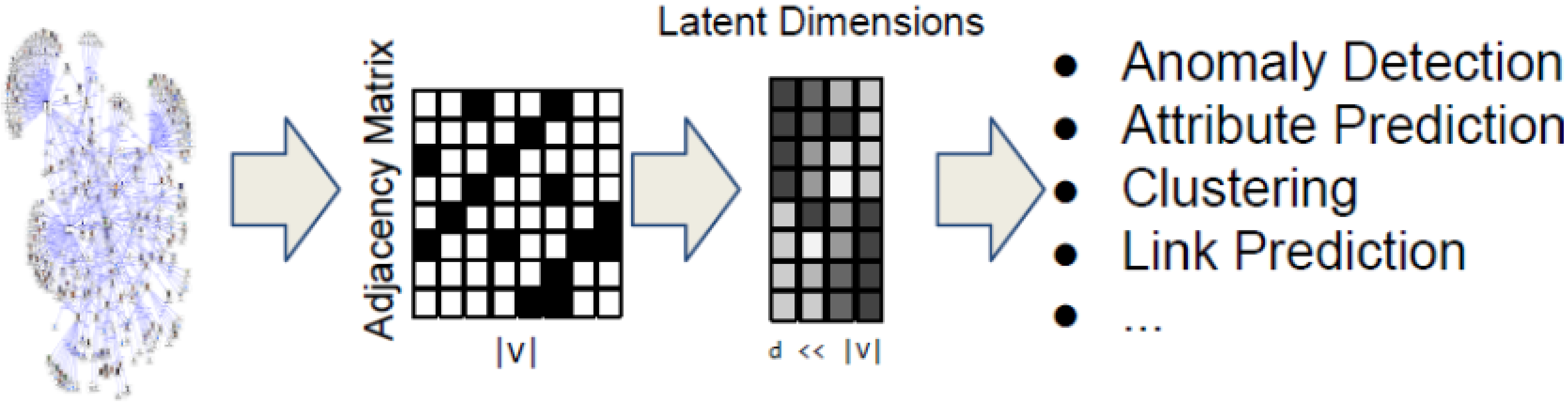
- Graph feature learning = representation learning
- Goal: efficient task-independent feature learning for machine learning in networks!



Feature representation, embedding

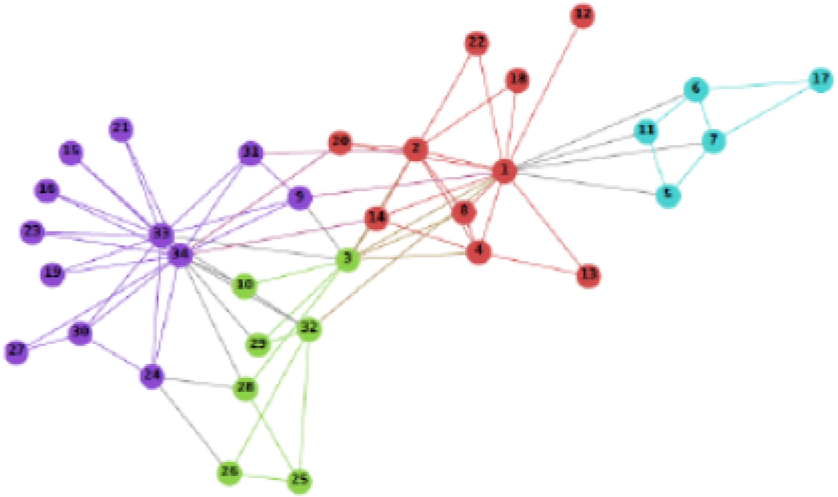
Why Network Embedding?

- We map each node in a network into a low-dimensional space
  - Distributed representation for nodes
  - Similarity between nodes indicate the link strength
  - Encode network information and generate node representation

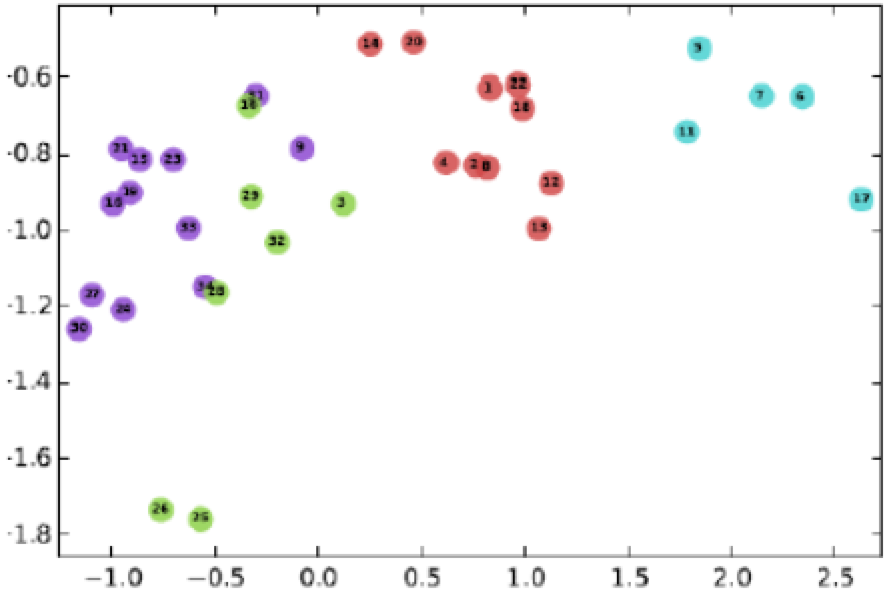


Example

- Zachary's Karate Club network



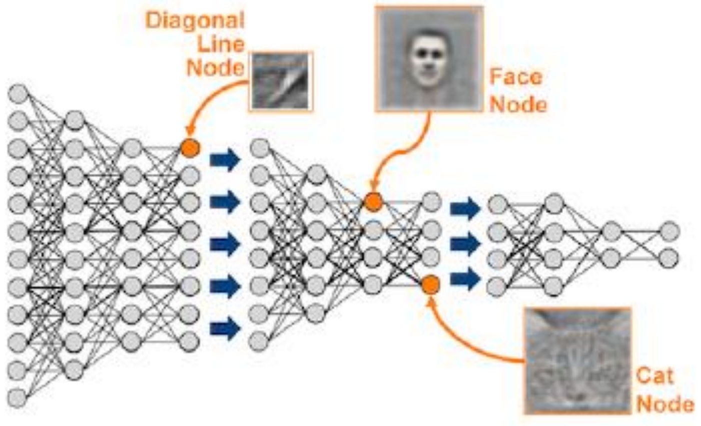
Input

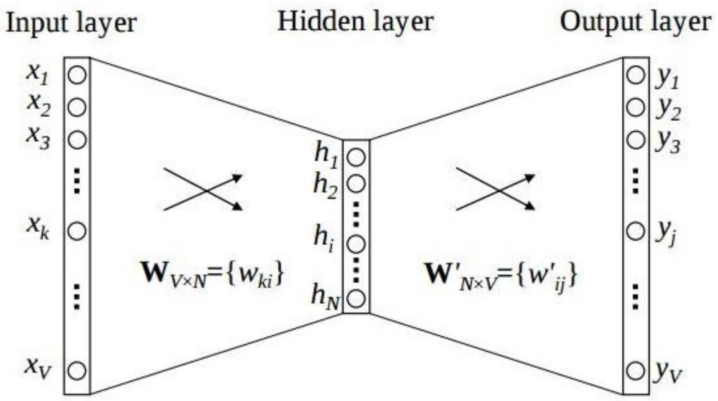


Output

Why Is It Hard?

- Graph representation learning is hard
  - Images are fixed size
    - Convolutions (CNNs)
  - Text is linear
    - Sliding window (word2vec)
  - Graphs are neither of these!
    - Node numbering is arbitrary
      - (node isomorphism problem)
    - Much more complicated structure

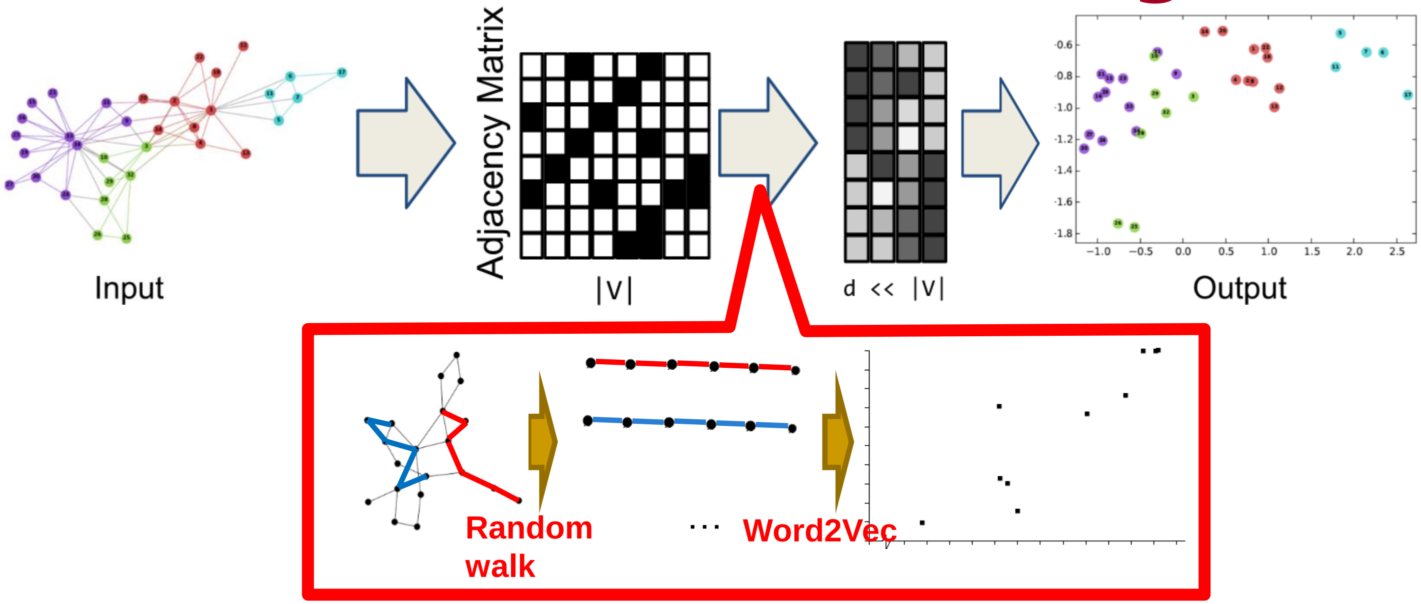




Outline

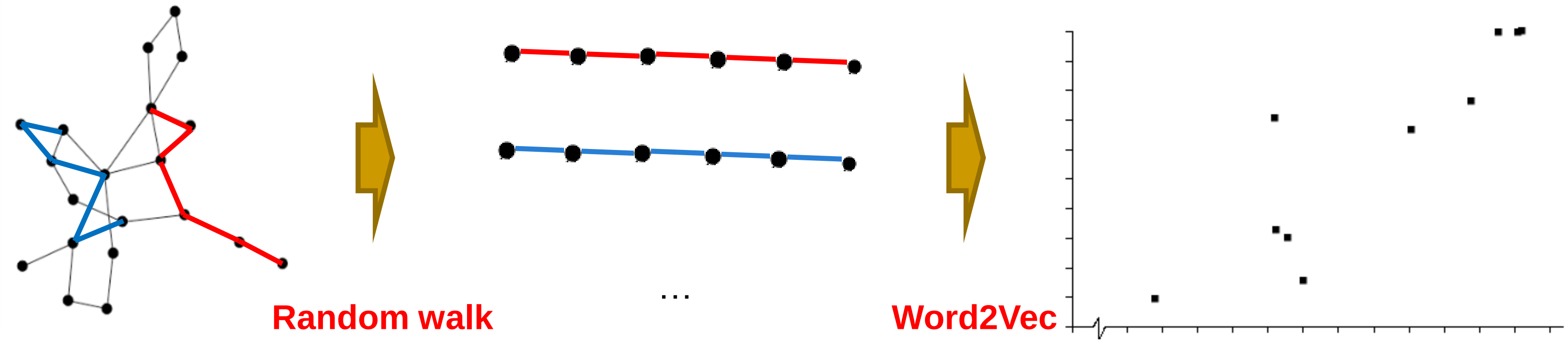
- ☑ Motivation
- ➡ ☐ **DeepWalk**
- ☐ Node2vec
- ☐ SIDE

DeepWalk: Random Walk Based Network Embedding



- Simple but effective learning process

DeepWalk Details



- Generate random walks for each vertex
- Regard random walks as sentences
- Use Word2Vec to learn representation for each vertex
  - Similar words appear in similar contexts
  - Similar nodes connect to similar neighbors
- $P(u,v) = \sigma(W_u \cdot W_v') = \frac{1}{1+\exp\left(-W_u \cdot W_v'\right)}$ where σ is a sigmoid function

Aside: Word2vec

Word Embedding in NLP

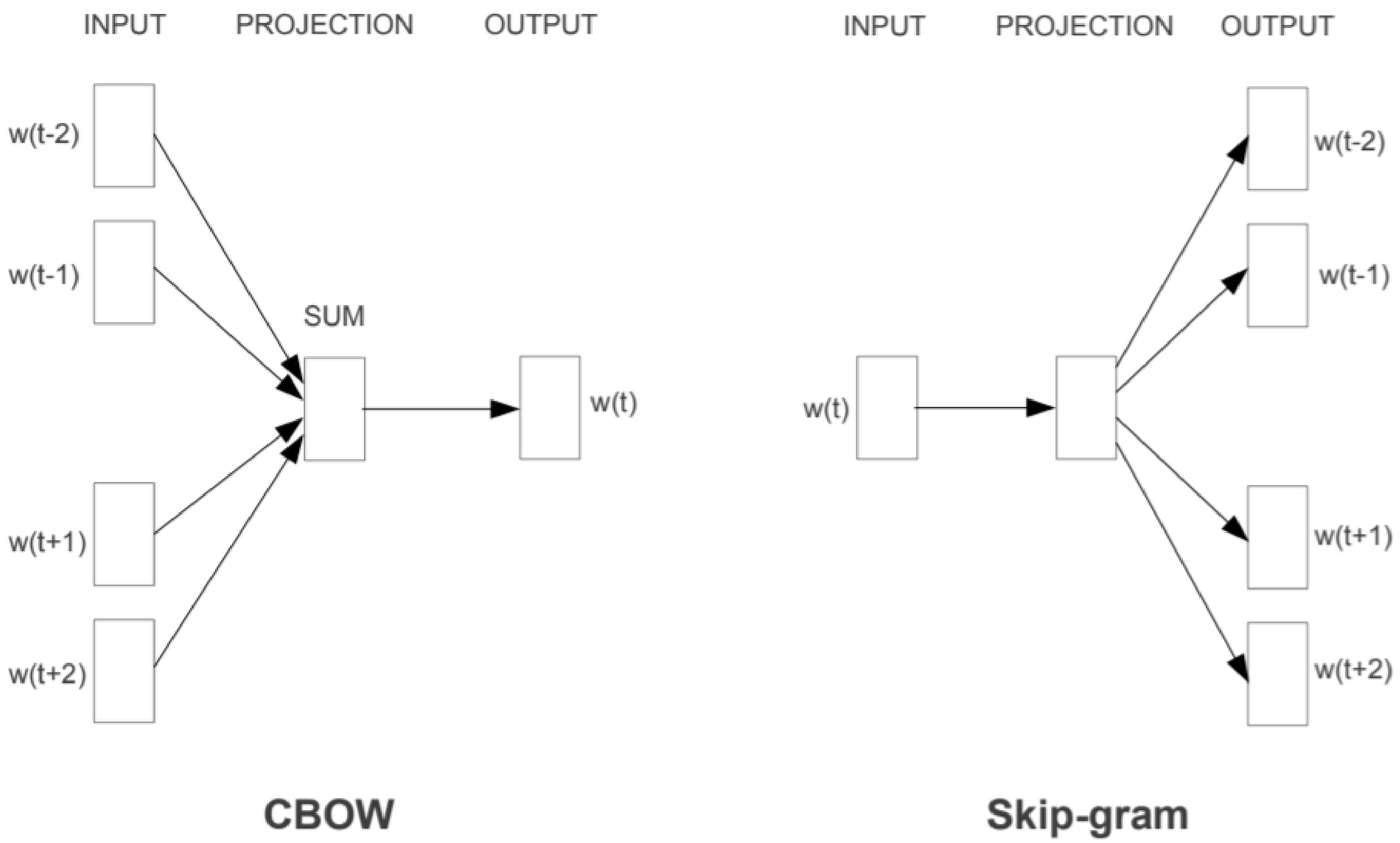
- Neural language models
  - Learns distributed representation of words (also known as word embeddings)
  - In the embedding space, words that frequently appear in similar contexts (or any pair of words sharing some features learned by the model) are close to each other. This often results in words with similar meanings being neighbors
  - Skipgram model (word2vec)
    - Given a sequence of training words $w_1, w_2, \dots, w_T$,

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

where $p(w_O|w_I) = \frac{\exp({v\prime_{w_O}}^T v_{w_I})}{\sum_{w=1}^{|V|}\exp({v\prime_w}^T v_{w_I})}$

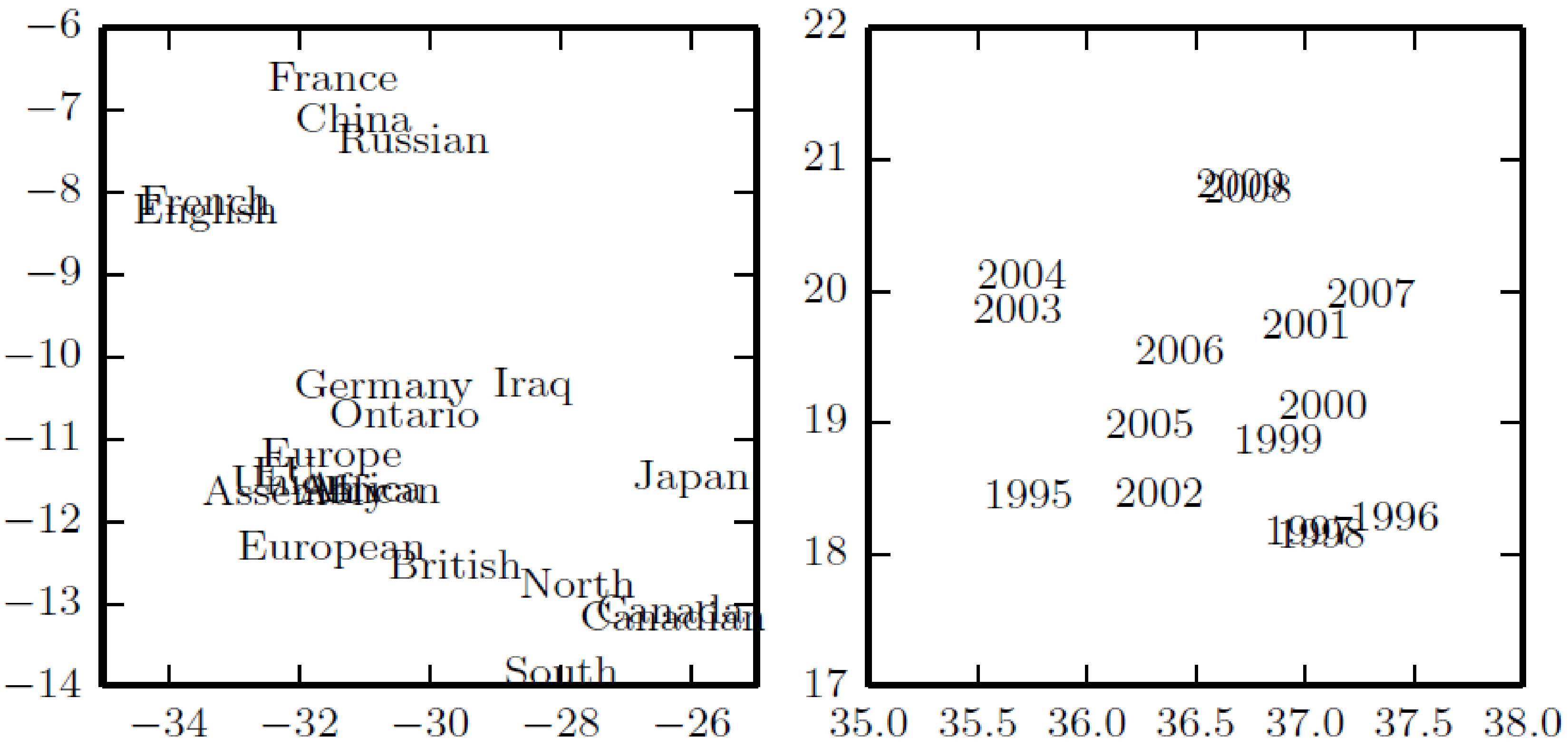
Word Embedding in NLP

- CBOW vs. Skipgram



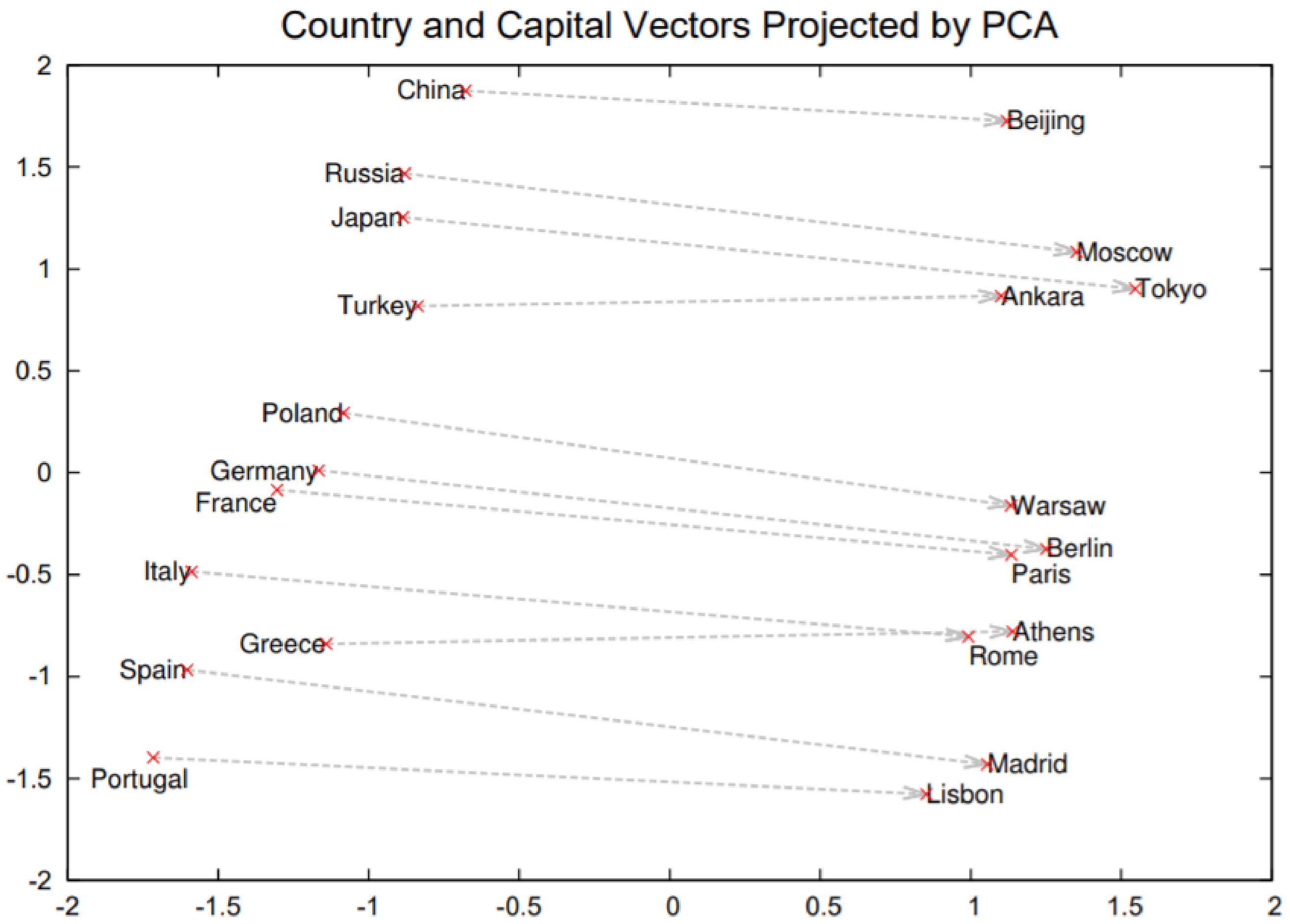
Word Embedding in NLP

- Word2vec



Word Embedding in NLP

- Word2vec



High-Dim. Output Layers for Large Vocabularies

- In many natural language applications, we want our model to produce words as the fundamental unit of the output
- The vocabulary V often contains hundreds of thousands of words
- Assume h is the top hidden layer used to predict the output probabilities y
  - $a_i = b_i + \sum_j W_{ij} h_j$
  - $y_i = \frac{e^{a_i}}{\sum_{i'=1}^{|V|} e^{a_{i'}}}$
- To compute y_i, we need to compute |V| terms in the denominator!

Negative Sampling

- In word2vec, our goal is to maximize $p(w_O|w_I) = \frac{\exp({v\prime_{w_O}}^T v_{w_I})}{\sum_{w=1}^{|V|} \exp({v\prime_w}^T v_{w_I})}$
- Evaluating the denominator requires too much computation
- Negative sampling
  - Maximize a bit different, but related objective
  - New objective: $\log \sigma({v'_{w_O}}^T v_{w_I}) + \sum_{i=1}^{k} E_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^T v_{w_I})\right]$
  - Intuition: maximize ${v'_{w_O}}^T v_{w_I}$ for similar words w_O and w_I, but minimize ${v'_{w_i}}^T v_{w_I}$ for dissimilar words w_i and w_I

End of Aside

Outline

- ☑ Motivation
- ☑ DeepWalk
- ➡ ☐ **Node2vec**
- ☐ SIDE

Overview of Node2Vec

- **Goal:** Embed nodes with similar network neighborhoods close in the feature space.
- We frame this goal as prediction-task independent maximum likelihood optimization problem.
- Key observation: Flexible notion of network neighborhood $N_S(u)$ of node u leads to rich features.
- Develop biased 2nd order random walk procedure S to generate network neighborhood $N_S(u)$ of node u.

Unsupervised Feature Learning

- Intuition: Find embedding of nodes to d-dimensions that preserves similarity

- Idea: Learn node embedding such that nearby nodes are close together

- Given a node u , how do we define nearby nodes?
  - $N_S(u)$: neighbourhood of u obtained by some strategy S

Feature Learning as Optimization

- Given G = (V, E), our goal is to learn a mapping $f: u \to R^d$

- Log-likelihood objective: $max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))$
  - where $N_S(u)$ is neighborhood of node u
- Given node u, we want to learn feature representations predictive of nodes in its neighborhood $N_S(u)$

Feature Learning as Optimization

- $max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))$
- Assumption: conditional likelihood factorizes over the set of neighbors
  - $\log \Pr(N_S(u) | f(u)) = \sum_{n_i \in N_S(u)} \log \Pr(f(n_i) | f(u))$
- Softmax parameterization
  - $\Pr(f(n_i) | f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$

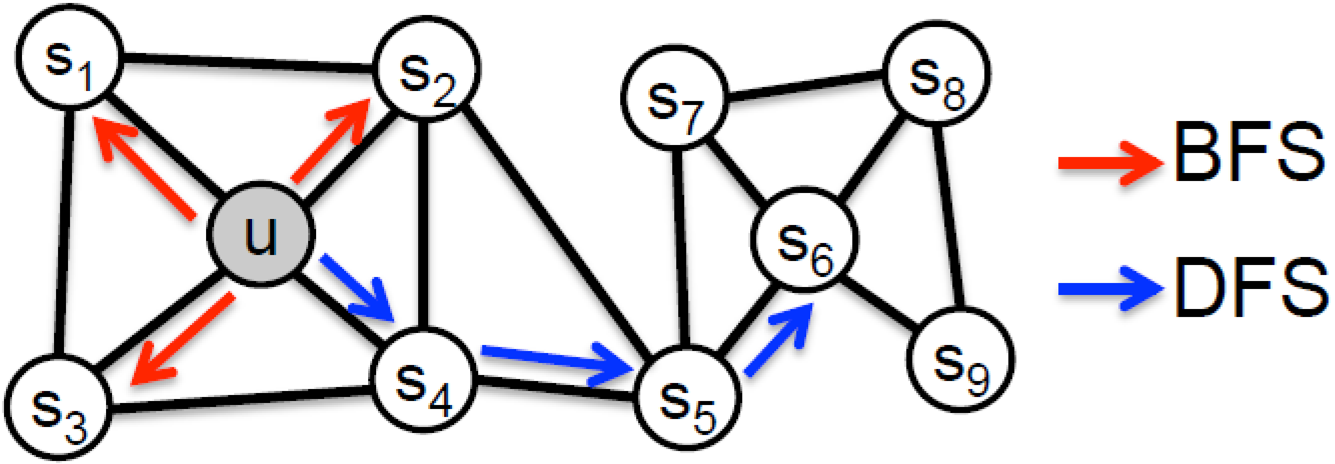
Negative Sampling

$$max_f \sum_{u \in V} \sum_{n \in N_S(u)} \log \frac{\exp(f(n) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

- Maximize the objective using stochastic gradient descent with negative sampling
  - Computing the summation is expensive
  - Idea: just sample a couple of “negative nodes”
  - This means at each iteration only embeddings of a few nodes will be updated at a time
  - Much faster training of embeddings

How to Determine $N_S(u)$

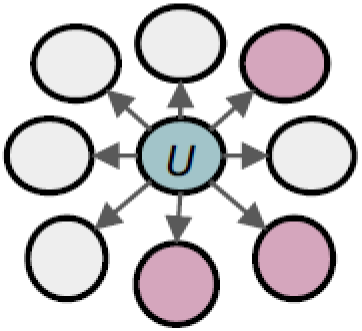
- Two classic strategies to define a neighborhood $N_S(u)$ of a given node u



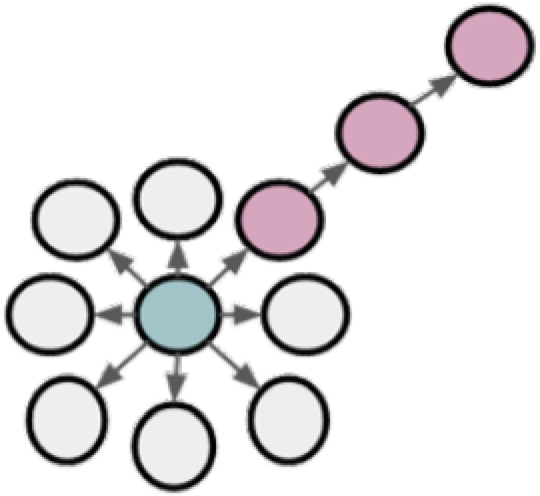
$N_{BFS}(u) = \{ s_1, s_2, s_3\}$ Local microscopic view

$N_{DFS}(u) = \{ s_4, s_5, s_6\}$ Global macroscopic view

BFS vs. DFS



BFS:
Micro-view of neighbourhood



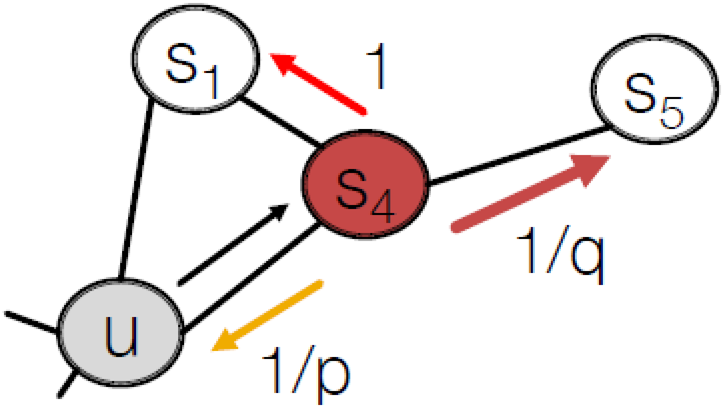
DFS:
Macro-view of neighbourhood

Interpolating BFS and DFS

- Biased random walk S that given a node u generated neighborhood $N_S(u)$
- Two parameters
  - Return parameter p: return back to the previous node
  - In-out parameter q: moving outwards (DFS) vs. inwards (BFS)

Biased Random Walks

- $N_S(u)$: Biased 2nd-order random walks explore network neighborhoods



$u \to s_4 \to ?$

u
s_1
s_5

  - BFS-like: low value of p
  - DFS-like: low value of q
- p, q can be learned in a semi-supervised way

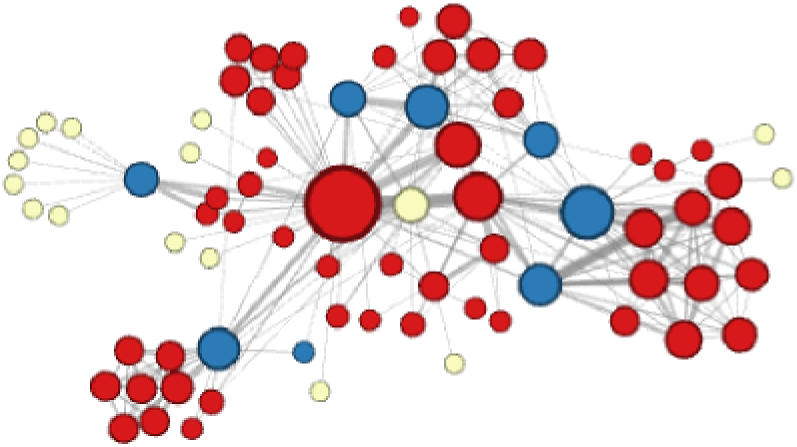
node2vec algorithm

- 1) Compute random walk probs.
- 2) Simulate r random walks of length l starting from each node u
- 3) Optimize the node2vec objective using Stochastic Gradient Descent

- Linear-time complexity
- All 3 steps are individually parallelizable

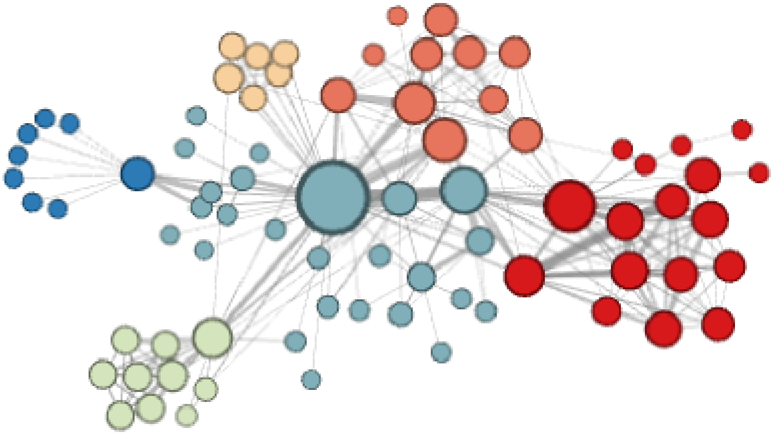
Experiments: Micro vs. Macro

- Interactions of characters in a novel



p=1, q=2

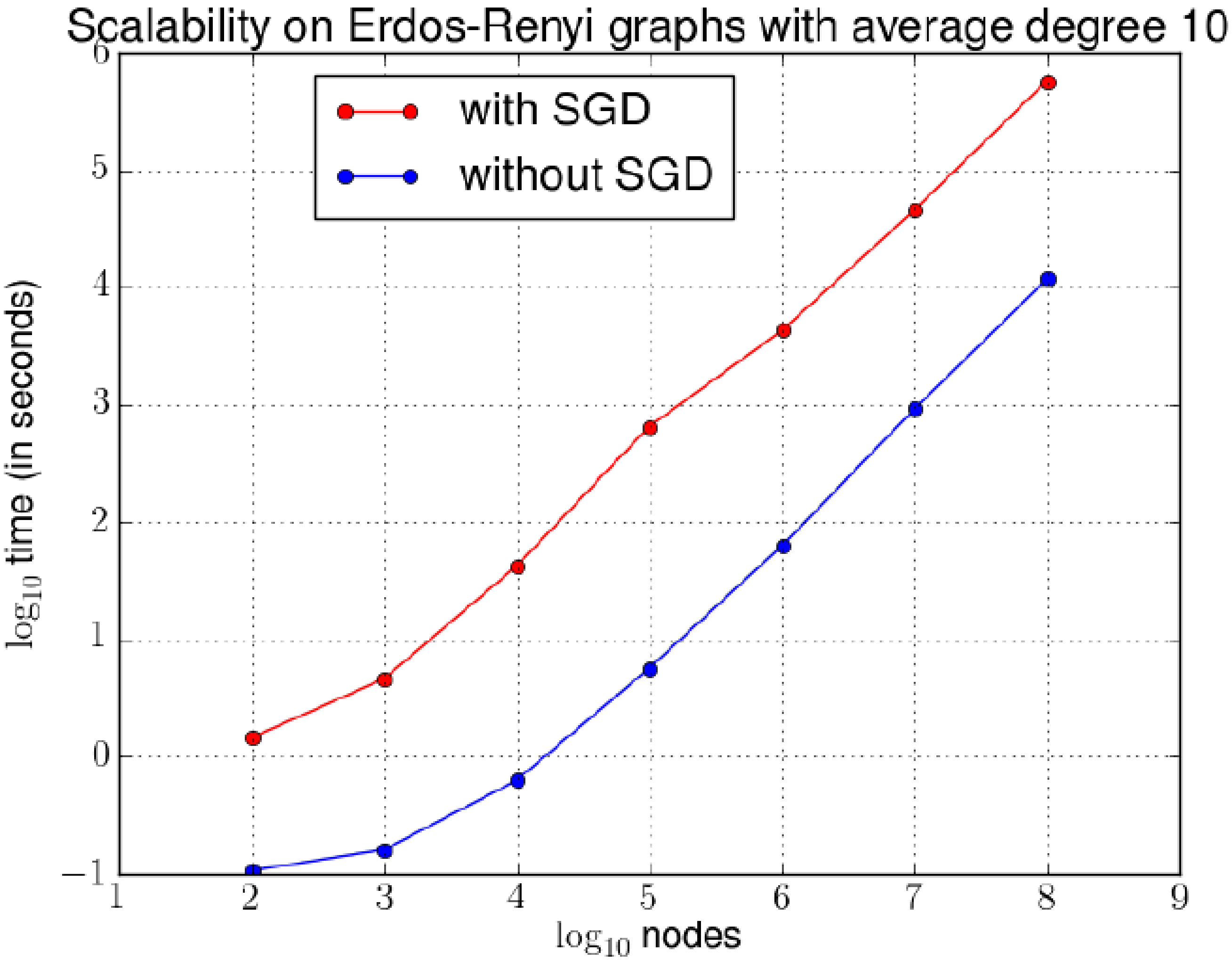
Microscopic view of the network neighbourhood



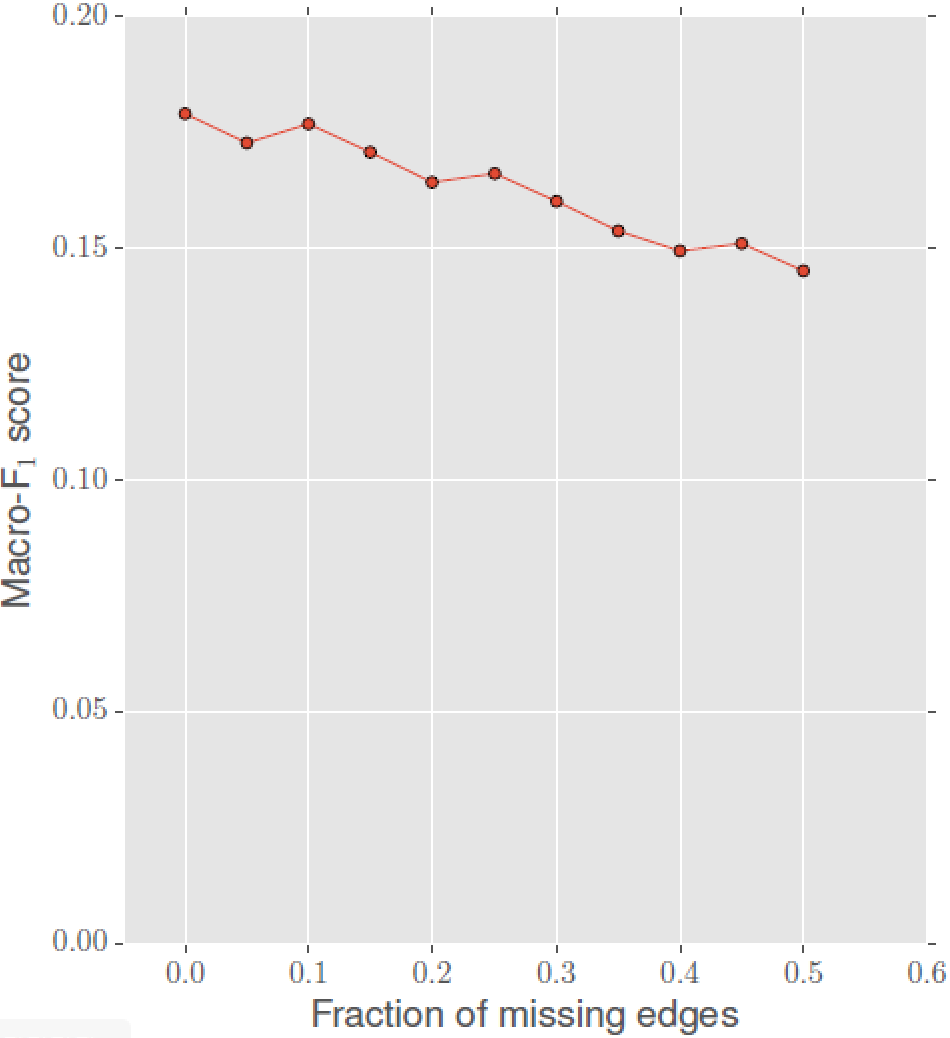
p=1, q=0.5

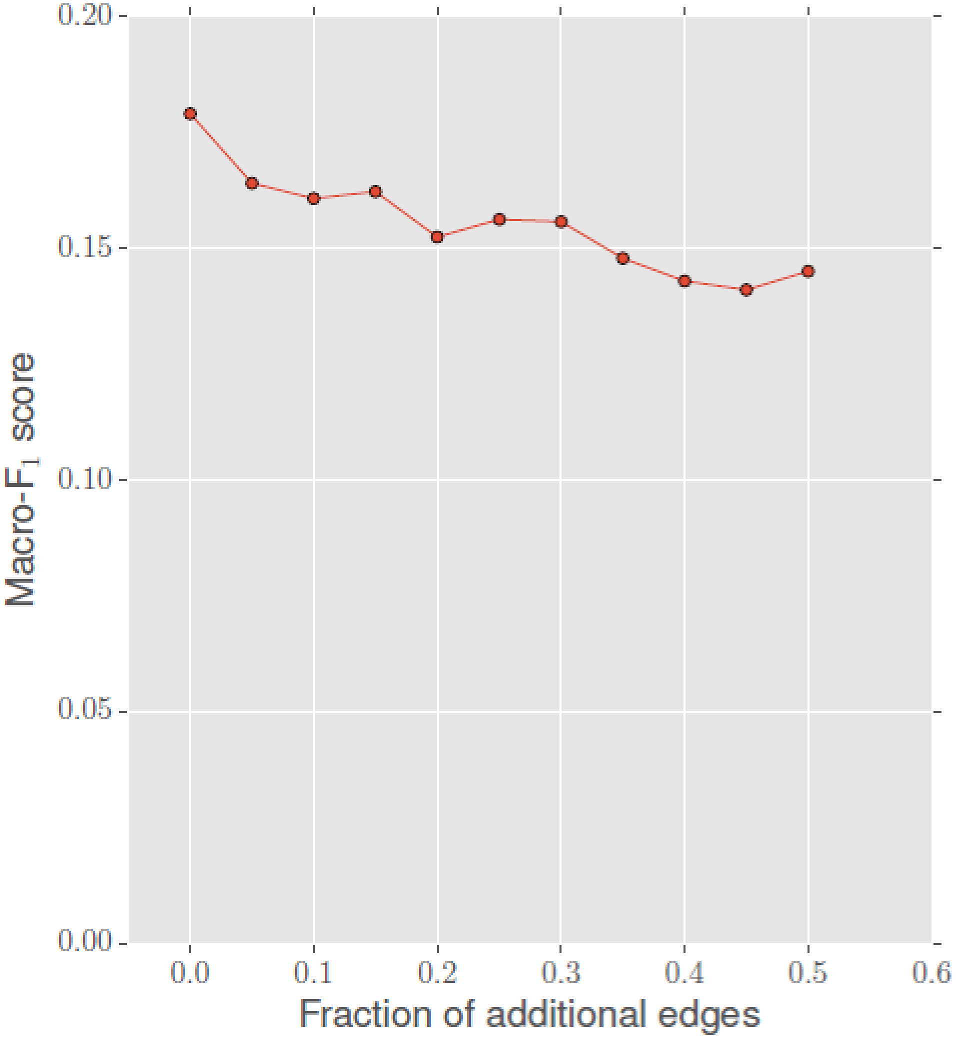
Macroscopic view of the network neighbourhood

Scalability of node2vec



Incomplete Network Data (PPI)





Node2vec: Summary

- General-purpose feature learning in networks:
  - An explicit locality preserving objective for feature learning.
  - Biased random walks capture diversity of network patterns.
  - Scalable and robust algorithm with excellent empirical performance.

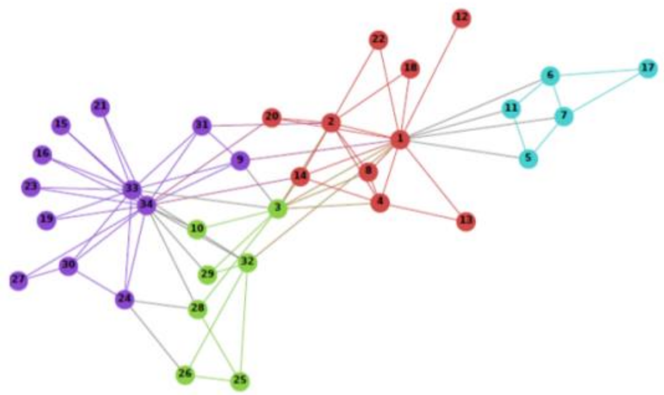
Outline

- ☑ Motivation
- ☑ DeepWalk
- ☑ Node2vec
- ➡ □ **SIDE**

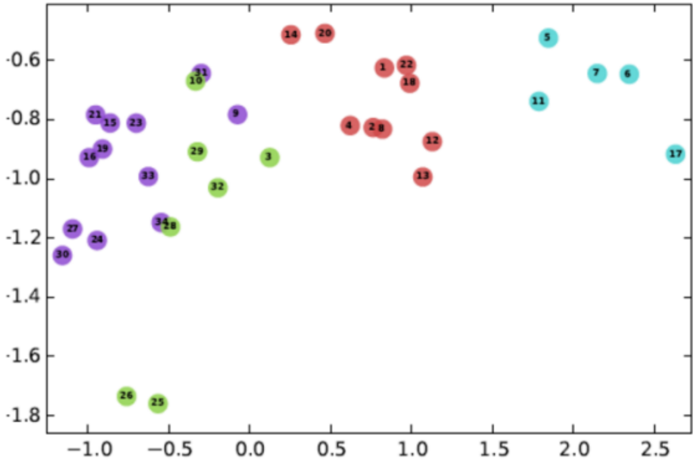
Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U Kang,
SIDE: Representation Learning in Signed Directed Networks,
The Web Conference (WWW) 2018, Lyon, France.

Introduction

- Given a social graph, how can we learn a low-dimensional representation of each node? I.e., how can we **embed** each node in the graph **into the Euclidean space**?



Input



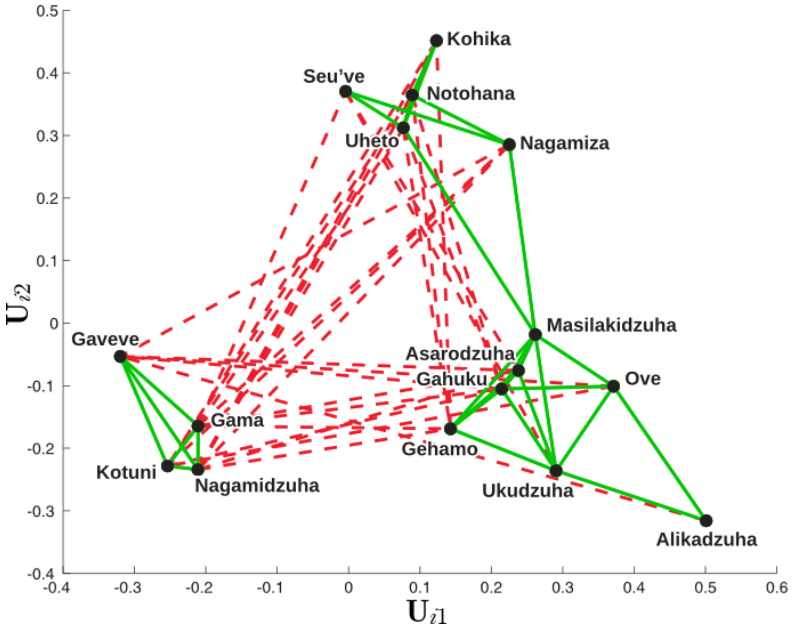
Output

Introduction

- How can we **embed** each node in the graph **into the Euclidean space**?
  - *Benefit*
    - The Euclidean distance becomes a surrogate for dissimilarity
    - Standard machine learning techniques can be applied
    - Visualization in low dimension is possible
  - Property
    - *Proximity*: nodes connected by heavily weighted edges are close
    - *Transitivity*: friends of friends are closer than stranger
  - One popular solution: neural network embedding

Problem Formulation

- How can we extend **network embedding** method to encode structure of a **directed** graph with **negative edges**?



Epinions.com

Slashdot

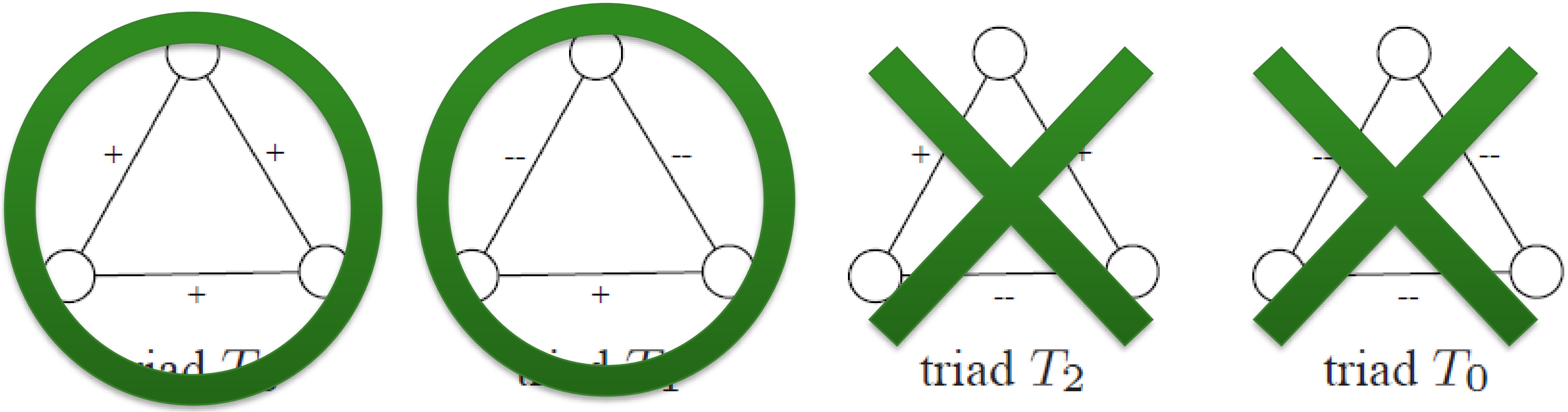
News for Nerds. Stuff that matters.

Two Reasons for Link Formation

- Similarity
  - Nodes form links to similar nodes
  - Social phenomenon: homophily
  - Properties: symmetric and transitive
  - Model: distance in embedding space
- Connectivity
  - Some nodes form more links than others do
  - Social phenomenon: preferential attachment
  - Properties: asymmetric, nontransitive and personalized
  - Model: ???

The Theory of Structural Balance

- The friend of my friend is my friend
- The friend of my enemy is my enemy
- The enemy of my friend is my enemy
- The enemy of my enemy is my friend



Main Ideas

- Challenges
  - Multi-step relationship: ambiguity of sign and direction
  - Negative edges: difference of negative and empty edges
  - Individual connectivity: representation of connectivity
- Main Ideas
  - Sign and direction aggregation: consistent with balance theory
  - Signed proximity term: interpretation of negative edges
  - Bias terms: modeling a sort of individualized social radius

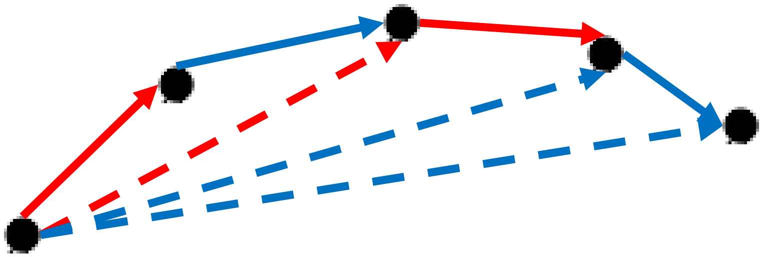
Likelihood Formation

$$J = \sum_{(u,v)\in \mathcal{D}} [-\log P(u,v) + \sum_{j=1}^{n} -\log P(u,v_j')]$$
$$+ \frac{\lambda}{2}(\|b^{in,+}\|^2 + \|b^{in,-}\|^2 + \|b^{out,+}\|^2 + \|b^{out,-}\|^2)$$

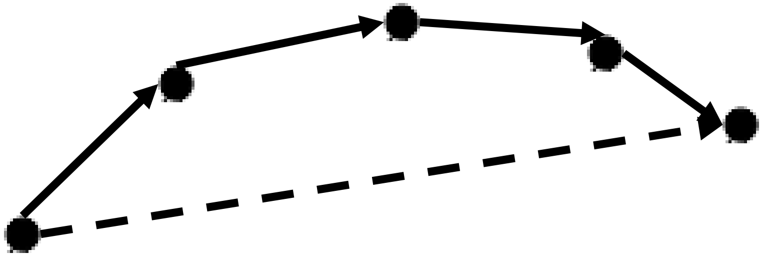
- (u,v): a node pair with aggregated sign and direction
- P(u,v): probability of forming a link u to v
- b: bias terms

Sign and Direction Aggregation

- Sign Aggregation
  - Sign defined by structural balance theory



- Direction Aggregation
  - Direction according to topological order of walk



Signed Proximity Term and Bias Terms

- Likelihood Formulation
  - $$P(u,v)=\begin{cases}\sigma\left(W_u^{out}\cdot W_v^{in}+b_u^{out,+}+b_v^{in,+}\right) & \text{if } sign(u,v)>0\\ \sigma\left(-W_u^{out}\cdot W_v^{in}+b_u^{out,-}+b_v^{in,-}\right) & \text{if } sign(u,v)<0\\ \sigma\left(-W_u^{out}\cdot W_v^{in}\right) & \text{if } v \text{ is a noise}\end{cases}$$

  where σ is a sigmoid function
  - Signed Proximity Term $\pm W_u^{out}\cdot W_v^{in}$: higher inner product term increases/decreases the probability of positive/negative connection.
  - Bias Terms $b_u^{out,\pm}, b_v^{in,\pm}$: higher bias term increases the probability of corresponding connection.

Linkage to Social Theories

- Two Reasons for Link Formation
  - Similarity: Signed Proximity Term $\pm W_u^{out} \cdot W_v^{in}$
  - Connectivity: Bias Terms $b_u^{out,\pm}, b_v^{in,\pm}$
  - Nodes with higher connectivity can connect to less similar nodes.
- Social Balance Theory
  - Signed proximity term encourages configurations allowed by social balance theory
    - Triad with all positive edges: three nodes closely placed together
    - Triad with one negative and two positive edges: incompatible with transitivity of closeness
    - Triad with one positive and two negative edges: one node far apart from the other two closely placed nodes
    - Triad with all negative edges: three nodes far apart from each other

Additional Optimization

- Deletion of nodes with degree one
  - When random walker encounters degree-one node, she should always come back to the only neighbor.
  - Because of the power law distribution of node degree, there are many degree-one nodes.

  → Delete degree-one nodes before random walk
- Subsampling of high-degree nodes
  - Links of high-degree nodes are less informative.
  - Because of the power law distribution of node degree, random walk encounters high-degree nodes frequently.

  → Pass high-degree nodes with high probability when encountered during random walk

Link Sign Prediction

- Predict unobserved link sign from embedding vector and biases
- Simple logistic regression

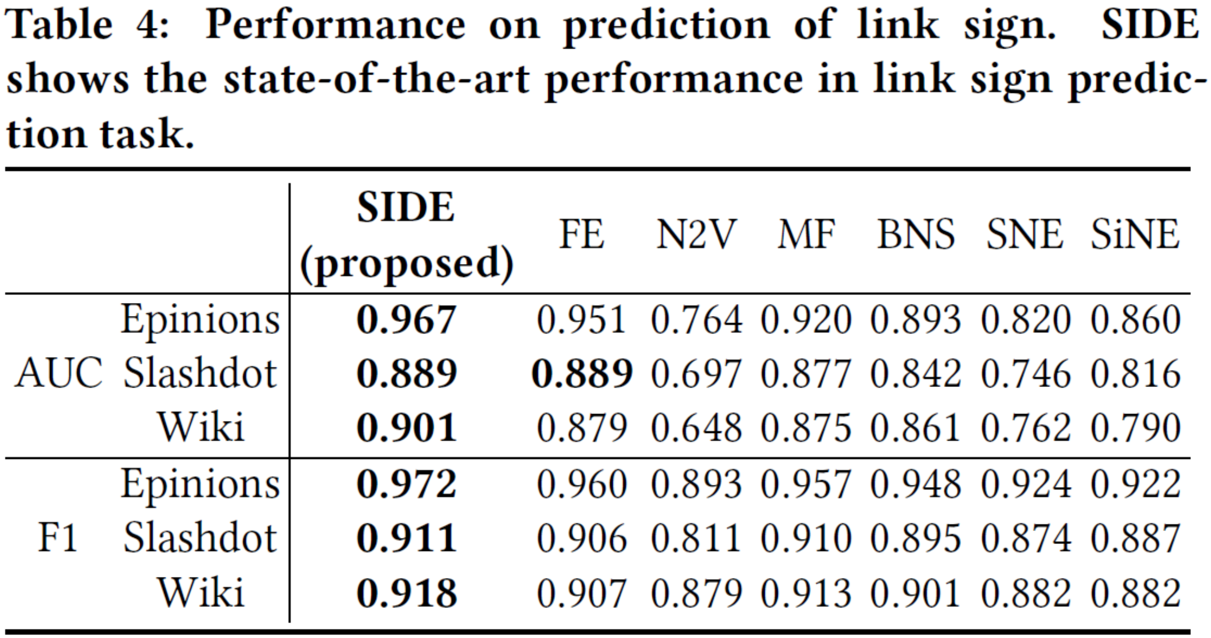


Table 4: Performance on prediction of link sign. SIDE shows the state-of-the-art performance in link sign prediction task.

		SIDE (proposed)	FE	N2V	MF	BNS	SNE	SiNE
AUC	Epinions	**0.967**	0.951	0.764	0.920	0.893	0.820	0.860
	Slashdot	**0.889**	**0.889**	0.697	0.877	0.842	0.746	0.816
	Wiki	**0.901**	0.879	0.648	0.875	0.861	0.762	0.790
F1	Epinions	**0.972**	0.960	0.893	0.957	0.948	0.924	0.922
	Slashdot	**0.911**	0.906	0.811	0.910	0.895	0.874	0.887
	Wiki	**0.918**	0.907	0.879	0.913	0.901	0.882	0.882

Embedding Analysis

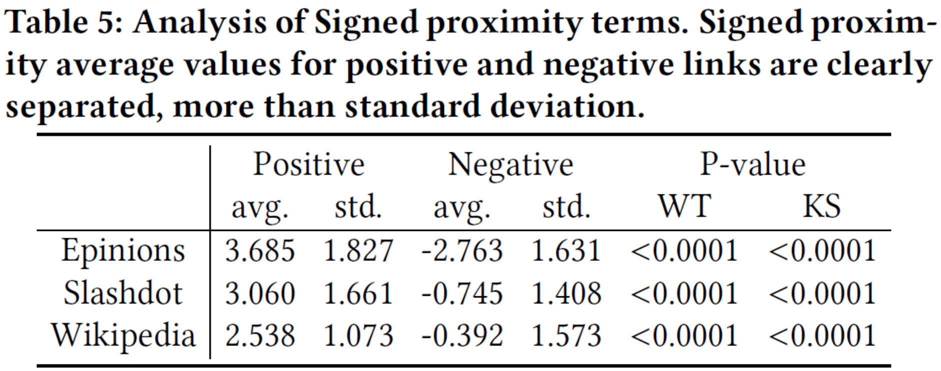


Table 5: Analysis of Signed proximity terms. Signed proximity average values for positive and negative links are clearly separated, more than standard deviation.

	Positive		Negative		P-value	
	avg.	std.	avg.	std.	WT	KS
Epinions	3.685	1.827	-2.763	1.631	<0.0001	<0.0001
Slashdot	3.060	1.661	-0.745	1.408	<0.0001	<0.0001
Wikipedia	2.538	1.073	-0.392	1.573	<0.0001	<0.0001

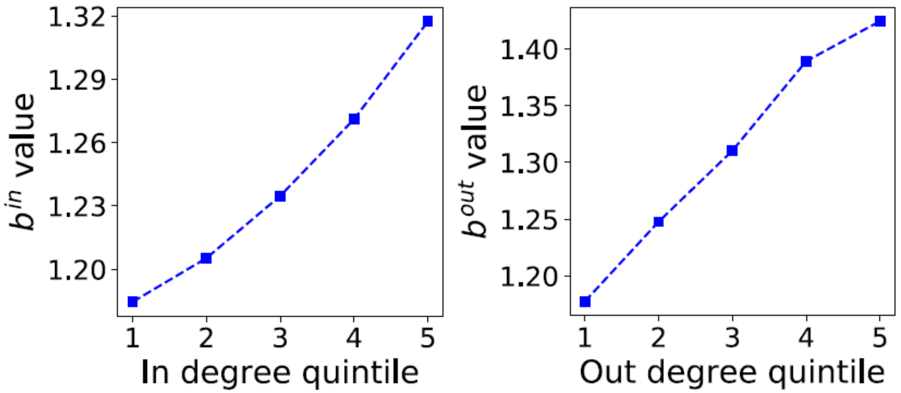
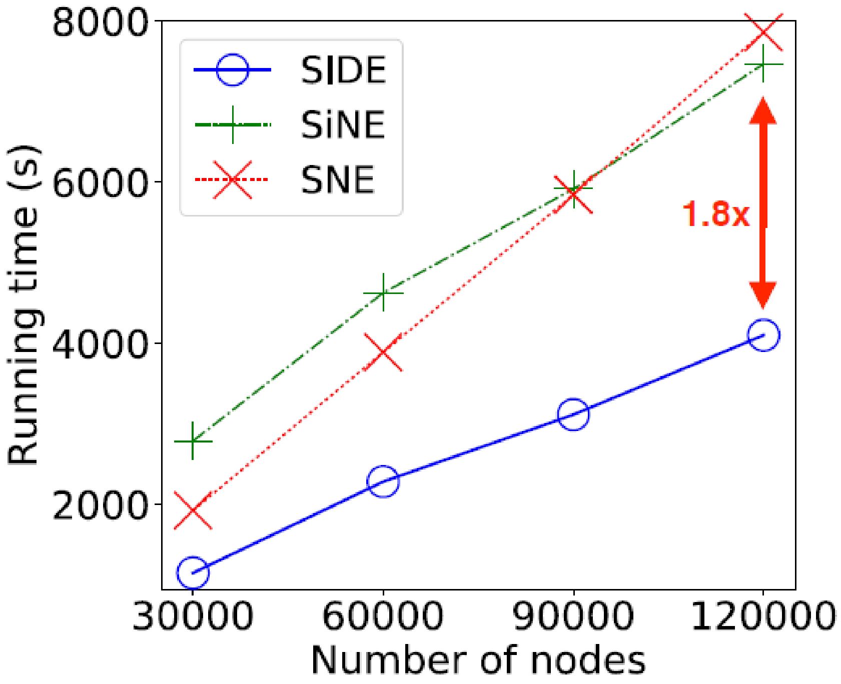


Figure 2: Correlation of bias and degree. Bias value increases as the degree quintile increases. This is consistent with the preferential attachment interpretation of bias terms.

- Verify that parameters are learned as intended
- Check whether
  - learned embedding vectors have positive/negative inner product for positive/negative links, respectively
  - learned bias terms increase as node degrees increase

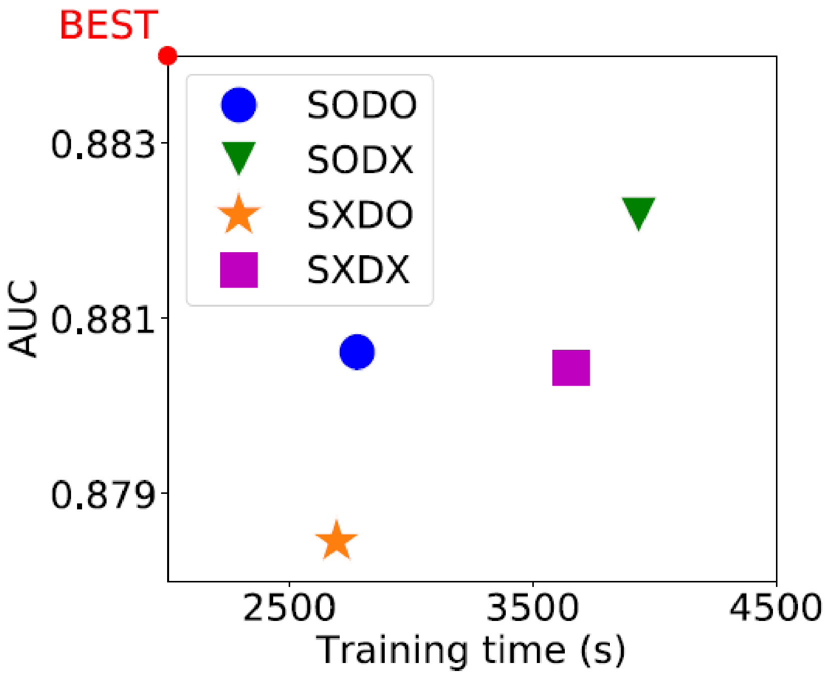
Scalability

- SIDE is linearly scalable in terms of the number of nodes.
- SIDE is at most 1.8x faster than competitors.



Effect of Optimization

- Subsampling of high-degree nodes(S) increases performance with only small increase in training time.
- Deletion of degree-one nodes(D) reduces training time by sacrificing small performance.
- Applying both techniques, better performance with smaller training time can be achieved.



SIDE: Summary

- SIDE: a fast and accurate network embedding method to represent signed directed network.
- SIDE overcomes three challenges: (1) utilization of multi-step connections, (2) consistent interpretation of both signs, and (3) representation of individual linking tendency.
- SIDE is based on the following social theories: (1) homophily and preferential attachment interpretation of link formation, and (2) social balance theory for link sign.
- SIDE achieves the state-of-the-art performance in the task of link sign prediction.

What You Need to Know

- Motivation of graph representation learning
- Deepwalk
- Node2vec
- SIDE

Questions?