

Aditya Grover, Jure Leskovec

Node2Vec: Scalable Feature Learning for Networks

2016

2021. 02. 23

Applied Statistics

Sunwoo Kim



Introduction

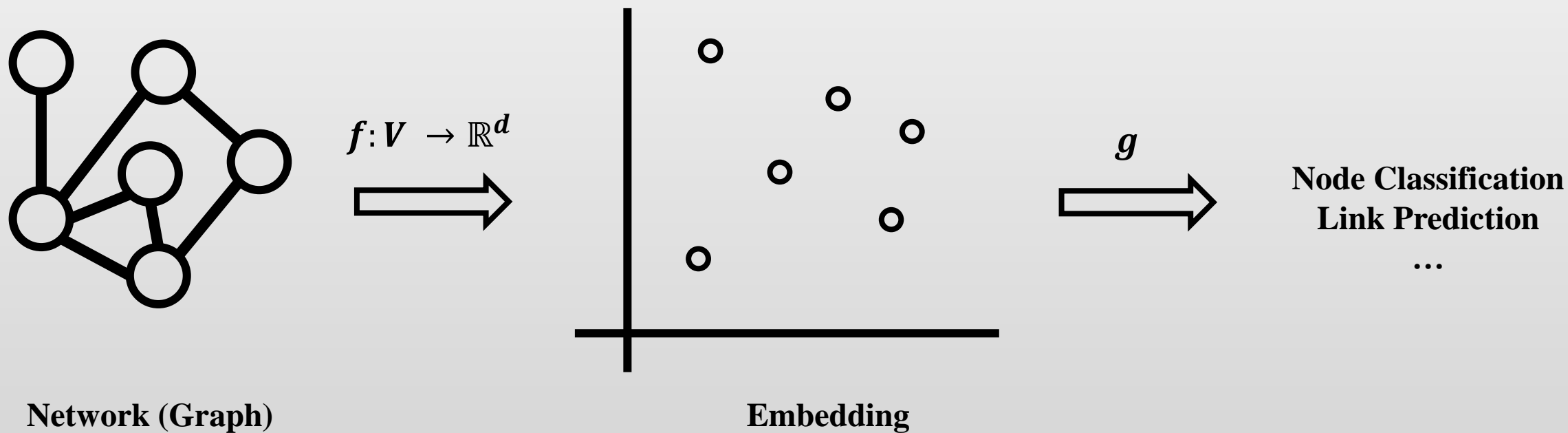


Understanding Node2Vec



Discovery & Discussion

Introduction



How to find a **GOOD** embedding?



Hand-Engineering Domain-Specific features based on expert knowledge.
But it requires one's tedious efforts and **cannot be generalized**.



Learn Feature Representations by solving an **Optimization Problem**.
But it's hard to define an **objective function**, which is closely related
to the **notion of node's neighbors**.

Deep Walk, LINE, and Node2Vec

➔ *Review. Deep Walk*

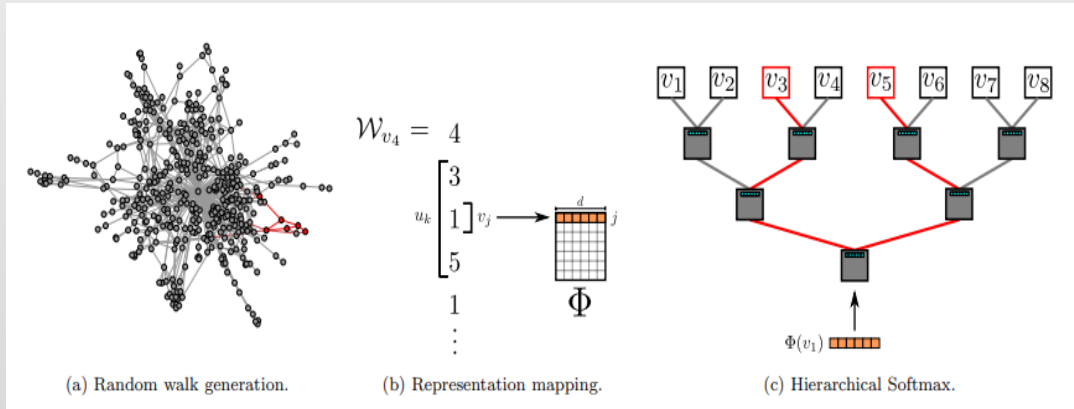


Figure from [Perozzi et al. Deep Walk. 2014]

Key Idea.

- Idea of **Skip-Gram** + Generating **Random-Walk Sequence**
- Walk length and Window Size
- For the computational efficiency, use **Hierarchical Softmax**.

$$P(u_k | \Phi(v_j)) = \prod_{l=1}^{\lceil \log |V| \rceil} P(b_l | \Phi(v_j))$$

$$P(b_l | \Phi(v_j)) = \frac{1}{1 + \exp(-\Phi(v_j) \cdot \Psi(b_l))}$$

$(b_0, b_1, \dots, b_{\lceil \log |V| \rceil})$: sequence of tree nodes

Φ : Embedding Vector / Ψ : Tree representation

➔ Review. LINE

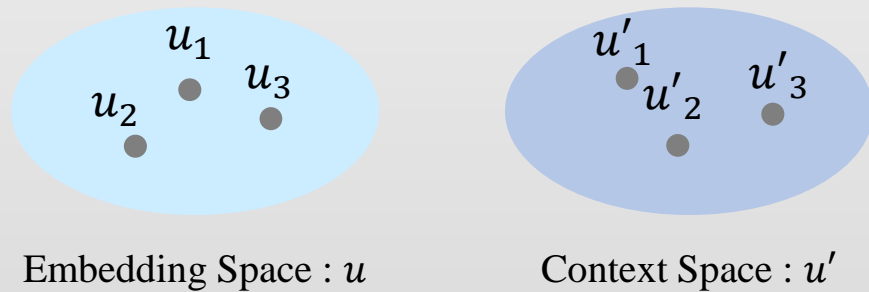


Figure from [Professor's Lecture Note]

It is important to distinguish
embedding vector & context vector

Key Idea.

- **First-order Proximity** : Connection between the nodes. (**Edges**)
- **Second-order Proximity** : **Shared Neighbors** between the nodes.
- In order to avoid a trivial solution(FOP) & decrease the computational cost(SOP), use **Negative Sampling**

FOP : Maximizing $\log \sigma(\mathbf{u}_j^T \cdot u_i) + \sum_{i=1}^K E_{v_n \sim P_{n(v)}} [\log \sigma(-\mathbf{u}_n^T \cdot u_i)]$

SOP : Maximizing $\log \sigma(\mathbf{u}_j^T \cdot u_i) + \sum_{i=1}^K E_{v_n \sim P_{n(v)}} [\log \sigma(-\mathbf{u}_n^T \cdot u_i)]$

Where $\sigma = \frac{1}{1 + \exp(-x)}$

➔ *Mentioned Drawbacks*

Authors claim that Deep Walk & LINE rely on a **rigid notion of a network neighbor**.

These Approaches are **insensitive to connectivity patterns** unique to networks

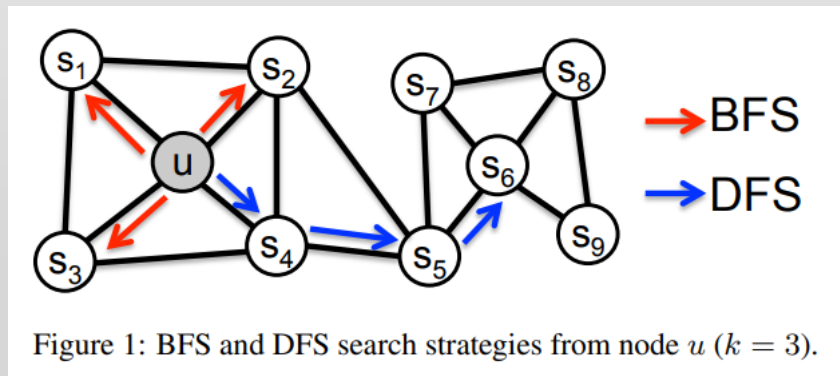


Figure from [Grover and Leskovec. node2vec. 2016]

1. Nodes could be organized based on their communities. (i.e., Homophily)
2. Nodes could be organized based on their roles. (i.e., Structural Equivalence)

Case 1 : (u, s_1) Case 2 : (u, s_6)

So, it is important to **address these roles to get a flexible algorithm!**

“By choosing an **appropriate notion of a neighborhood**,
node2vec can learn representations of case 1 and/or case 2”

Understanding Node2Vec

➔ *Definitions*

Notation	Explanation
$G = (V, E)$	Given Network
$f : V \rightarrow \mathbb{R}^d$	Mapping Function from nodes to feature representation.
f	$ V \times d$ size Matrix
$N_S(u) \subset V$ where $u \in V$	Network Neighborhood of node u , Generated through S
S	Neighborhood Sampling Strategy

➡ *Ideation*

- ① **Goal** : Find the **mapping function f** , that is **network-aware** and **neighborhood preserving**
- ① They use the architecture of the **Skip-Gram**
- ① Specifically, it is **maximum likelihood optimization** problem
- ② Then, **how can we define** aforementioned ‘Neighbors?’ → Will be covered soon in detail

➔ Optimization

$$\max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))$$

$$\max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))$$

$$= \max_f \sum_{u \in V} \log \prod_{n_i \in N_S(u)} \Pr(n_i | f(u))$$

$$= \max_f \sum_{u \in V} \sum_{n_i \in N_S(u)} \log \Pr(n_i | f(u))$$

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

$$= \max_f \sum_{u \in V} \sum_{n_i \in N_S(u)} \log [\exp(f(n_i) \cdot f(u))] - \log [\sum_{v \in V} \exp(f(v) \cdot f(u))]$$

$$= \max_f \sum_{u \in V} [-|N_S(u)| \log Z_u + \sum_{n_i \in N_S(u)} (f(n_i) \cdot f(u))], \quad z_u = \sum_{v \in V} e^{(f(u) \cdot f(v))}$$

$$\max_f \sum_{u \in V} [-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u)]$$

A. Conditional Independence

$$\Pr(N_S(u) | f(u)) = \prod_{n_i \in N_S(u)} \Pr(n_i | f(u))$$

B. Symmetry in Feature Space

$$\Pr(n_i | f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

➡ *Approximation*

$$\max_f \sum_{u \in V} [-\log \mathbf{z}_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u)]$$

$$\mathbf{z}_u = \sum_{v \in V} \exp(f(u) \cdot f(v)) \quad \text{Expensive to Compute! : } O(|V|^2)$$



Breakthrough



Negative Sampling :

$$\max_f \sum_{u \in V} \sum_{n_i \in N_S(u)} [\log \sigma(f(n_i) \cdot f(u)) + \sum_{l=1}^k \log \sigma(-f(m_l) \cdot f(u))]$$


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$


$m_l \sim P_V$: Random distribution over all nodes. / Here, sample k – negative nodes proportional to degree.

* According to CS224W Lecture. m_l can be equal to n_i

➡ Interpretation & Issues

$$\max_f \sum_{u \in V} \sum_{n_i \in N_S(u)} [\log \sigma(f(n_i) \cdot f(u)) + \sum_{l=1}^k \log \sigma(-f(m_l) \cdot f(u))]$$

 Locating neighbors close to each other,

 Repulsing

➡ Whole process reduce the computational cost

Consider basic SGD,

$$O_1 = -\log \sigma(f(n_i) \cdot f(u)) - \sum_{l=1}^k \log \sigma(-f(m_l) \cdot f(u))$$

Minimizing O_1

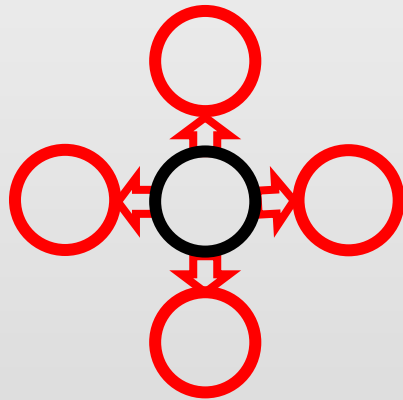
$N_S(u)$?

Unlike Skip-Gram, There is no defined sequence here.

How can we define the neighbors?

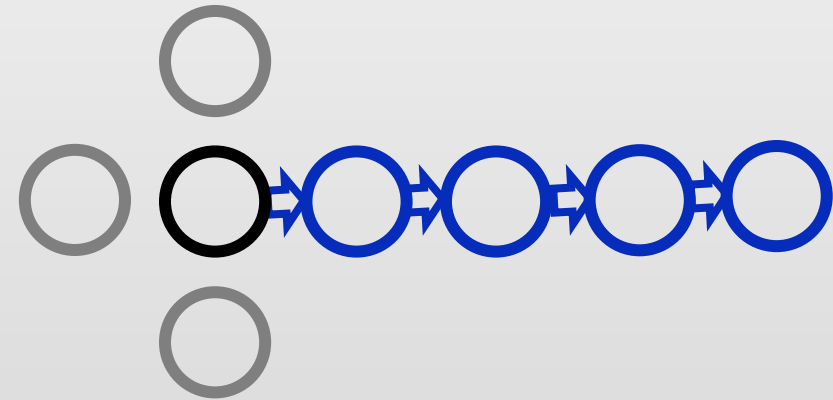
= How can we build the sampling strategy S ?

➔ *Two Similarities & Two Search Strategies*



Breadth-First Search (BFS)

- **Microscopic View** of neighbors
- Good to embed the **Structural Role**



Depth-First Search (DFS)

- **Macroscopic View** of neighbors
- Good to embed the **Community**

Got Figure Idea from CS224W. Leskovec.

➔ *Biased 2nd order Random-Walk*

Sampling Strategy that **interpolates between BFS & DFS**

- ✓ **2nd order** : Remember where it comes from
- ✓ **Bias** : Manipulating the probability in some measure

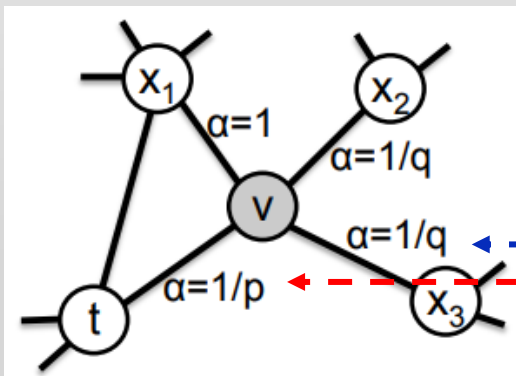


Figure from [Grover and Leskovec. node2vec. 2016]

A Walk has reached at node v after node t

Two Parameters :

Return Parameter, p ↻

- Getting back to its previous node
- High p ($> \max(q, 1)$) : Encourage to explore
- Low p ($< \min(q, 1)$) : Encourage to stay 'local'

In-Out Parameter, q ↻

- Moving Inward or Outward
- High q ($q > 1$) : Local View / **BFS like**
- Low q ($q < 1$) : Going further away / **DFS like**

Now in V

node	prob	Dist. From v
t	$1/p$	One less
x_1	1	Same
x_2 or x_3	$1/q$	One More

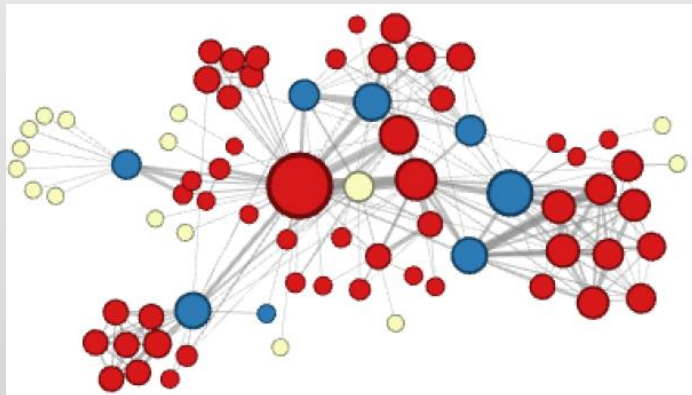
➔ *Case Study: Les Misérables network*

Relationship network between the characters in the novel Les Misérables

This can be Experimental Discovery

Embedding → K-Means Clustering

77 nodes. 254 edges. $d = 16$

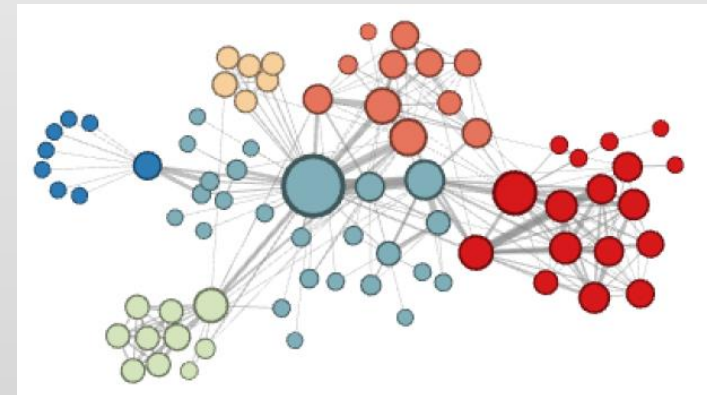


$p = 1, q = 2$

Microscopic view

Structural Equivalence (Role)

Figure from [Grover and Leskovec. node2vec. 2016]



$p = 1, q = 0.5$

Macroscopic view

Homophily (Community)

Figure from [Grover and Leskovec. node2vec. 2016]

➔ *Strict Explanation*

Starting from the source node u , generate a fixed l -length random walk.

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{l} \text{where } \pi_{vx} \text{ is unnormalized transition probability between nodes } v \text{ and } x \\ Z \text{ is the normalizing constant} \end{array}$$

Assume that a walk has reached node v after node t . Here, transition probability π_{vx} indicates a move on edge, from v to x . Then, $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, which is...

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p}, & \text{if } d_{tx} = 0 \\ 1, & \text{if } d_{tx} = 1 \\ \frac{1}{q}, & \text{if } d_{tx} = 2 \end{cases}$$

Note that d_{tx} should be one of $\{0, 1, 2\}$

We can say this as a 2nd Order Markov Chain.

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d , Walks per node r , Walk length l , Context size k , Return p , In-out q)
 $\pi = \text{PreprocessModifiedWeights}(G, p, q)$
 $G' = (V, E, \pi)$
 Initialize *walks* to Empty
 for $iter = 1$ **to** r **do**
 for all nodes $u \in V$ **do**
 $walk = \text{node2vecWalk}(G', u, l)$
 Append *walk* to *walks*
 $f = \text{StochasticGradientDescent}(k, d, \text{walks})$
 return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u , Length l)
 Initialize *walk* to $[u]$
 for $walk_iter = 1$ **to** l **do**
 $curr = \text{walk}[-1]$
 $V_{curr} = \text{GetNeighbors}(curr, G')$
 $s = \text{AliasSample}(V_{curr}, \pi)$
 Append s to *walk*
 return *walk*

Figure from [Grover and Leskovec. node2vec. 2016]

➔ *Compared to the Deep Walk & LINE*

Node2Vec

From the Idea of Skip-Gram

Using the **Flexible Sampling Strategy**

Biased 2nd Order Random-Walk

Approx. with Negative-Sampling

- *More **rich notion of neighbors***
- *Can be more **generalized***

Deep-Walk

From the Idea of Skip Gram

Normal Random-Walk (sampling of $p=q=1$ node2vec)

Approx. with Hierarchical Softmax

LINE

Idea of First / Second Order Proximity

Usage of Context Vector in Second Order Proximity

Approx. with Negative Sampling

“There is no clear
Winning Sampling Strategy
across all networks and all
Prediction tasks.”

Discovery & Discussion

➔ *Experimental Setup*

- Spectral Clustering : Matrix Factorization approach of top d eigen vectors of the normalized Laplacian matrix of graph G
 - Deep Walk : d -dimensional feature representation / $d = 128, r = 10, l = 80, k = 10$
 - LINE : $d/2$ with first order proximity + $d/2$ with second order proximity
 - Node2Vec : d -dimensional feature representation / $d = 128, r = 10, l = 80, k = 10, p, q = \{0.25, 0.5, 1, 2, 4\}$ in 10 – Fold CV
- ☑ Repeated 10 times for each model with different seeds
- Ⓢ Results are statistically significant with a p – value of less than 0.01

➡ *Multi-Label Classification*

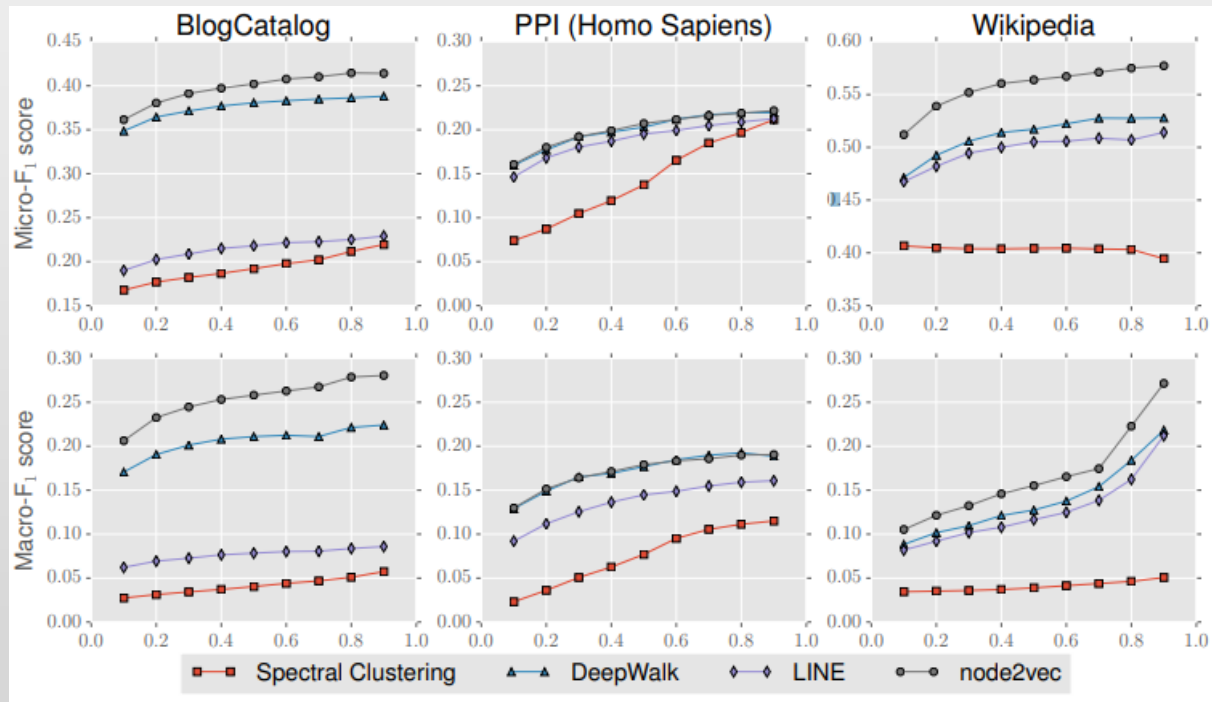


Figure from [Grover and Leskovec, node2vec, 2016]

x: fraction of labeled data used in training out of entire data

y: score

Showed good overall performance in a node classification

BlogCatalog:

Social relationships of the bloggers

10,312 nodes / 333,983 edges / 39 labels

Protein-Protein Interactions:

Subgraph of the PPI network for Homo Sapiens

3,890 nodes / 76,584 edges / 50 labels

Wikipedia:

Cooccurrence network of words with POS tags

4,777 nodes / 184,812 edges / 40 labels

They all have both equivalences

- One vs all Logistic Regression with L2 Regularization
- Using Macro F1 (Arithmetic Mean of various labels' F1)
- Using Micro F1 = Overall Accuracy

➔ *Parameter Sensitivity*

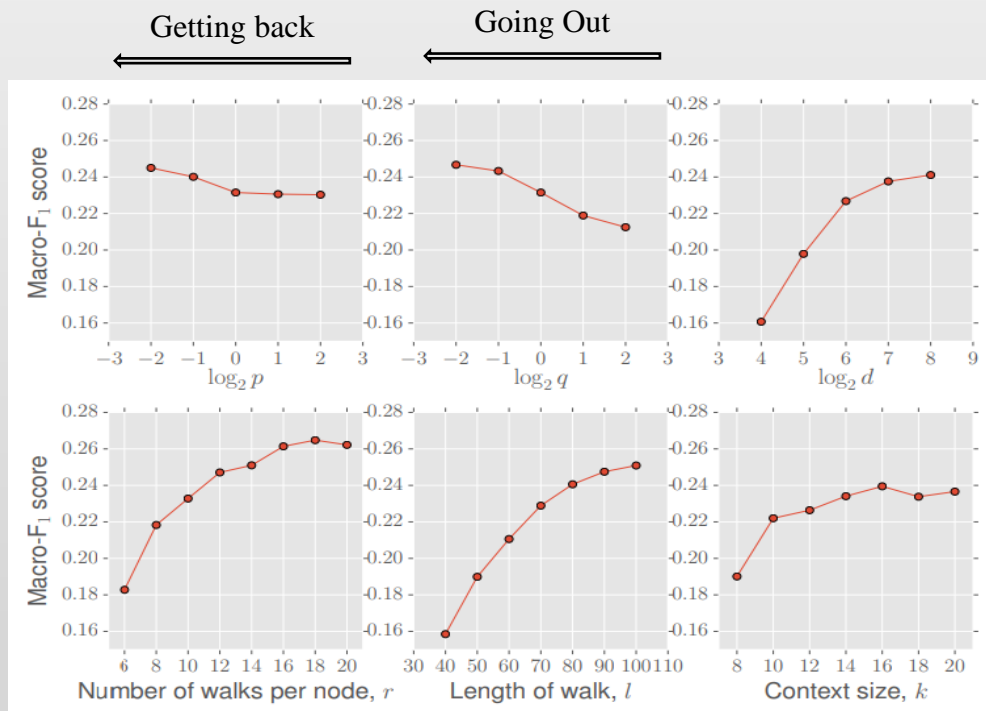


Figure from [Grover and Leskovec. node2vec. 2016]

- Performance tend to saturate once d reaches around 100
- Performance is proportional to the length of walk
- Context Size also matters, but increased optimization time
- p & q are set to be 1 at default
- BlogCatalog Dataset
- Train : Test = 1:1

➔ *Perturbation Analysis*

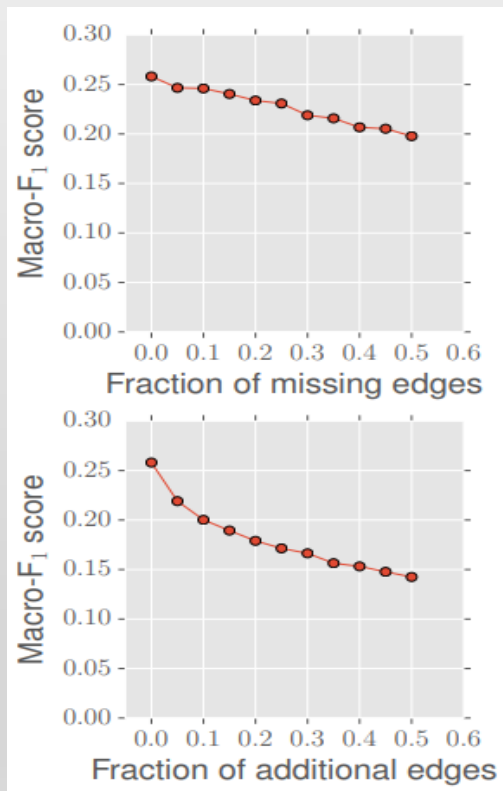


Figure from [Grover and Leskovec, node2vec, 2016]

- Implies the robustness of the model

Top Plot: Erase Edges Randomly

- When some information omitted
- Performance decrease, but it maintains to some extent

Bottom Plot: Add Noise Edges

- When some noise is added
- Performance decrease, but it maintains to some extent

- BlogCatalog Dataset

➔ *Scalability*

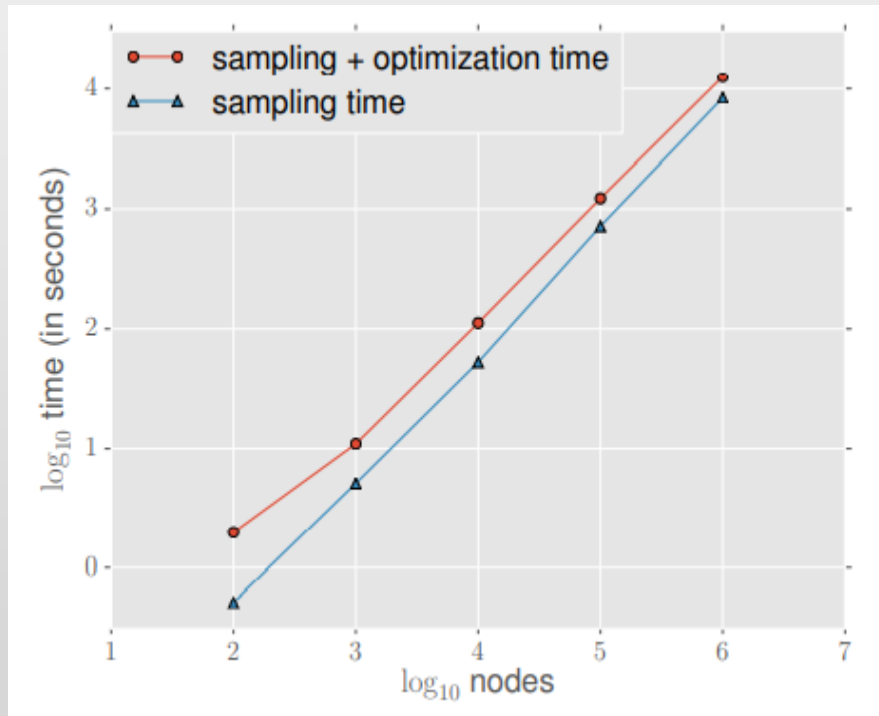


Figure from [Grover and Leskovec. node2vec. 2016]

- Measuring how much time it takes to train

Node2Vec can manage giant graph with the help of...

- Reuse of neighbors
- Negative Sampling & Asynchronous SGD
- Furthermore, we can manage best parameter searching

- Erdos-Renyi graphs

➔ Link Prediction

Op	Algorithm	Dataset		
		Facebook	PPI	arXiv
	Common Neighbors	0.8100	0.7142	0.8153
	Jaccard's Coefficient	0.8880	0.7018	0.8067
	Adamic-Adar	0.8289	0.7126	0.8315
	Pref. Attachment	0.7137	0.6670	0.6996
(a)	Spectral Clustering	0.5960	0.6588	0.5812
	DeepWalk	0.7238	0.6923	0.7066
	LINE	0.7029	0.6330	0.6516
	node2vec	0.7266	0.7543	0.7221
(b)	Spectral Clustering	0.6192	0.4920	0.5740
	DeepWalk	0.9680	0.7441	0.9340
	LINE	0.9490	0.7249	0.8902
	node2vec	0.9680	0.7719	0.9366
(c)	Spectral Clustering	0.7200	0.6356	0.7099
	DeepWalk	0.9574	0.6026	0.8282
	LINE	0.9483	0.7024	0.8809
	node2vec	0.9602	0.6292	0.8468
(d)	Spectral Clustering	0.7107	0.6026	0.6765
	DeepWalk	0.9584	0.6118	0.8305
	LINE	0.9460	0.7106	0.8862
	node2vec	0.9606	0.6236	0.8477

Table 4: Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstrapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (See Table 1 for definitions).

Operator	Symbol	Definition
Average	\boxplus	$[f(u) \boxplus f(v)]_i = \frac{f_i(u) + f_i(v)}{2}$
Hadamard	\boxtimes	$[f(u) \boxtimes f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\ \cdot\ _1$	$\ f(u) \cdot f(v)\ _1 = f_i(u) - f_i(v) $
Weighted-L2	$\ \cdot\ _2$	$\ f(u) \cdot f(v)\ _2 = f_i(u) - f_i(v) ^2$

Table 1: Choice of binary operators \circ for learning edge features. The definitions correspond to the i th component of $g(u, v)$.

Figure from [Grover and Leskovec. node2vec. 2016]

Showed good overall performance in a link prediction

Hadamard Operator works best

- Stable result
- Good Performance

Facebook

Users and their friendship relation

4,039 nodes / 88,234 edges

Protein-Protein Interactions:

Subgraph of the PPI network for Homo Sapines

19,706 nodes / 390,633 edges

arXiv ASTRO-PH:

Scientists and their collaboration

18,722 nodes / 198,110 edges

- Train 50% edges / Test 50% edges
- Randomly choose the same number of negative node pairs

Benchmark Models

Score	Definition
Common Neighbors	$ \mathcal{N}(u) \cap \mathcal{N}(v) $
Jaccard's Coefficient	$\frac{ \mathcal{N}(u) \cap \mathcal{N}(v) }{ \mathcal{N}(u) \cup \mathcal{N}(v) }$
Adamic-Adar Score	$\sum_{t \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log \mathcal{N}(t) }$
Preferential Attachment	$ \mathcal{N}(u) \cdot \mathcal{N}(v) $

Table 3: Link prediction heuristic scores for node pair (u, v) with immediate neighbor sets $\mathcal{N}(u)$ and $\mathcal{N}(v)$ respectively.

Figure from [Grover and Leskovec. node2vec. 2016]

➔ *Discussion*

Node2Vec

- ✓ Explains the trade-off between exploration-exploitation
- ✓ Provides a degree of interpretability to the embedding
- ✓ Provides a flexible and controllable search strategy
- ✓ Is Scalable and robust to perturbations
- ✓ Works well on node classification and link prediction

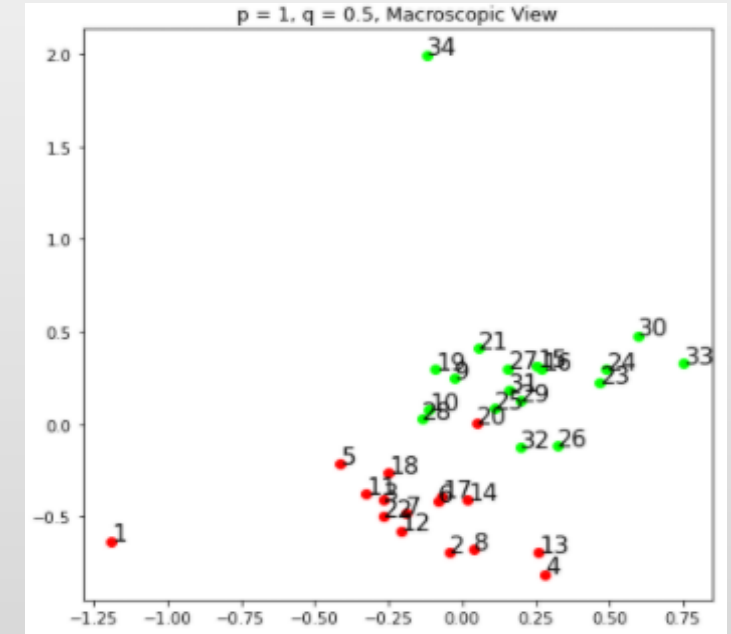
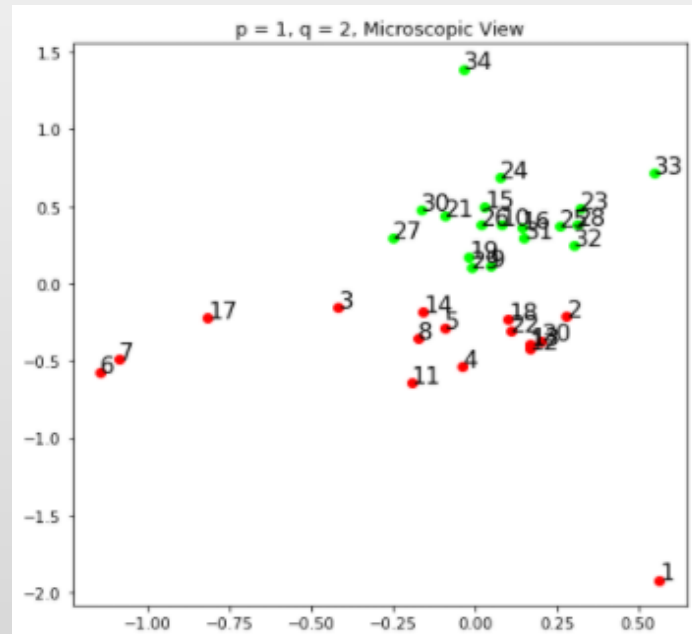
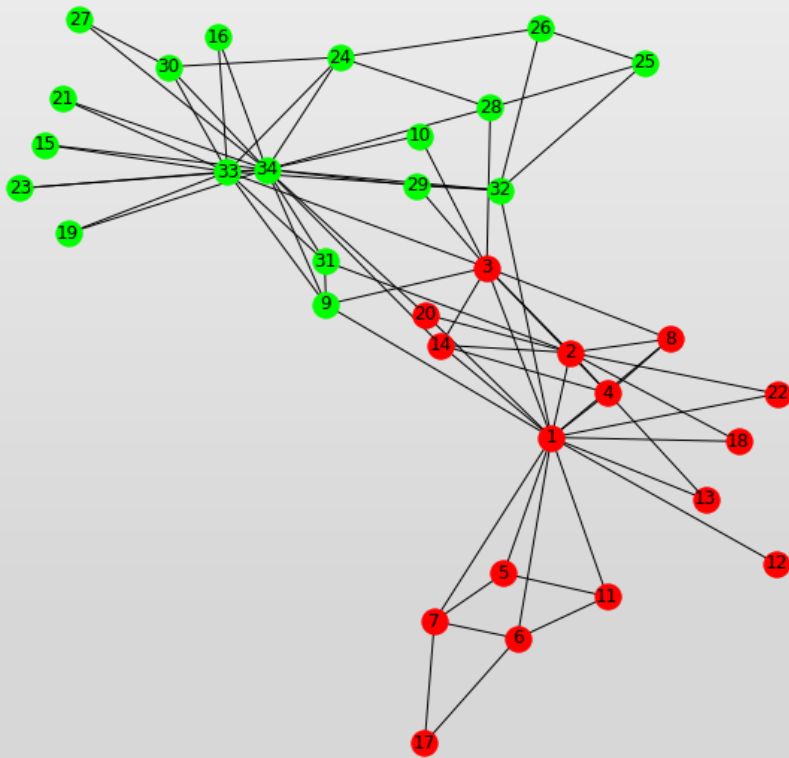
Future Work

- 📊 Reason why Hadamard operator works best
- 📊 Extension to the special structure network

*“Backbone of many deep learning algorithms,
may work as building blocks for end-to-end
deep learning on graphs!”*

Discovery & Discussion

➔ *Implementation on the Karate Club Data*



Embedding Size : 2
Initialization : $N(0, 0.1)$
Epochs : 10
Walk Length : 6
Window(Context) Size : 2
Learning Rate : 0.02
Negative Sample : 3

Thank You!

Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. 2016

Tang et al. LINE: Large-Scale Information Network Embedding. 2015

Perozzi et al. DeepWalk: Online Learning of Social Representations. 2014

Xin Rong. Word2vec Parameter Learning Explained. 2016

Professor's Lecture Note. Context Embedding Figure

Jure Leskovec. Stanford CS224W: Machine Learning with Graphs Lectures. <http://web.stanford.edu/class/cs224w/>

Breadth-first search. Wikipedia. https://en.wikipedia.org/wiki/Breadth-first_search

Depth-first search. Wikipedia. https://en.wikipedia.org/wiki/Depth-first_search

Multi-label classification. Wikipedia. https://en.wikipedia.org/wiki/Multi-label_classification