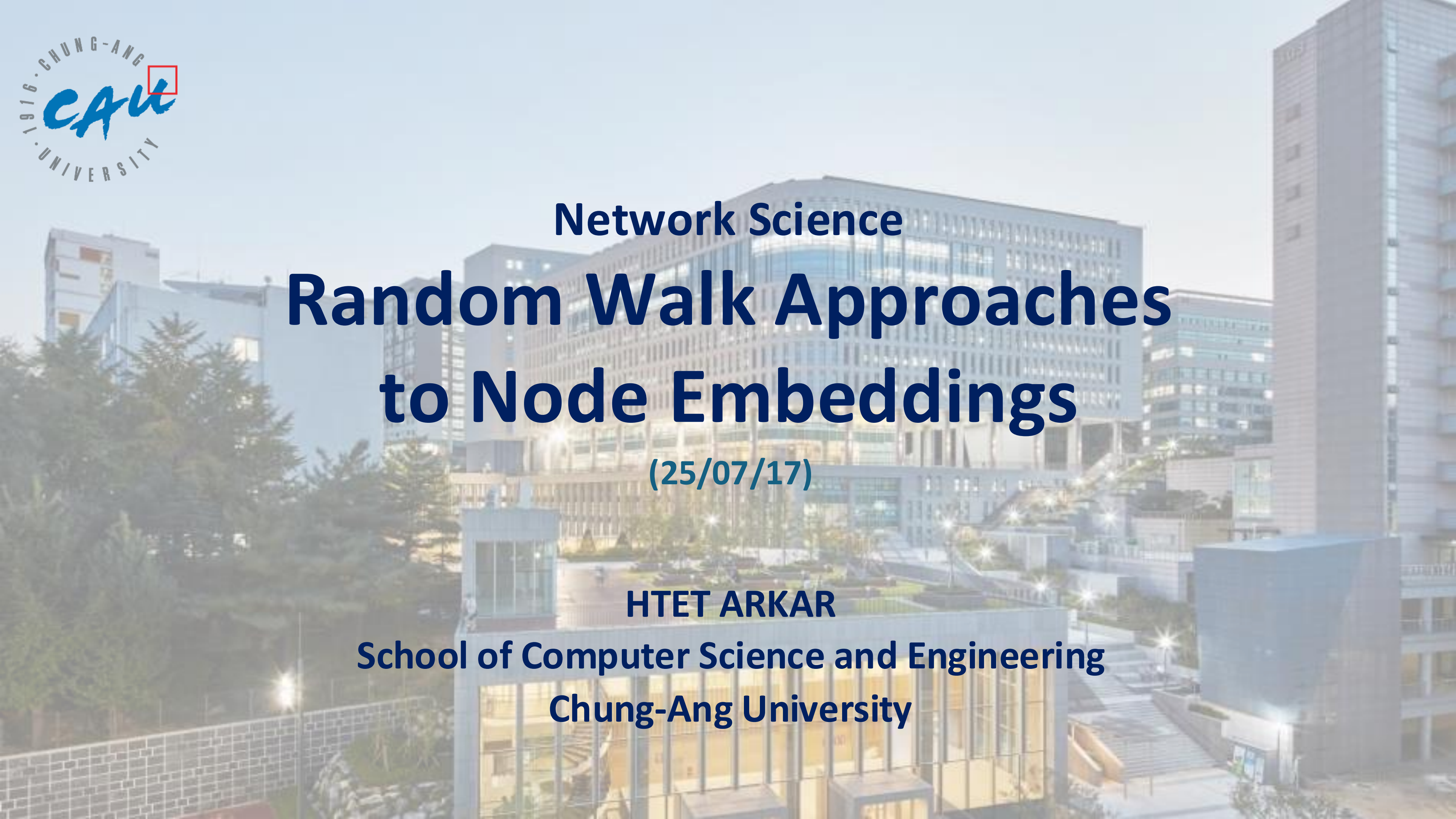


Network Science

Random Walk Approaches to Node Embeddings

(25/07/17)

HTET ARKAR
School of Computer Science and Engineering
Chung-Ang University

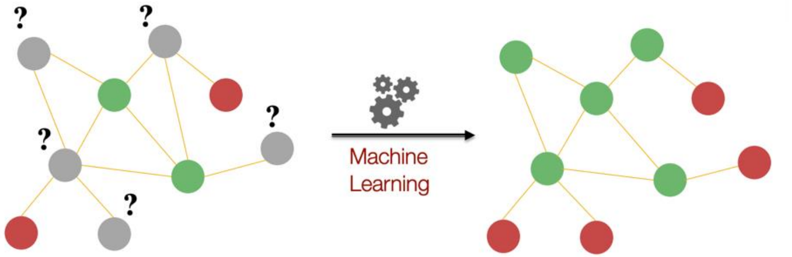
Outline

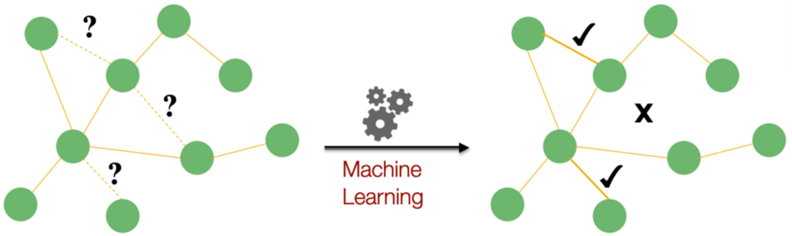
- **Machine Learning with Graphs**
- **Graph Representation Learning**
- **Learning Node Embeddings**
- **Random Walk Approaches to Node Embeddings**
    - ☐ DeepWalk
        - Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations. KDD.
    - ☐ Node2vec
        - Grover et al. 2016. node2vec: Scalable Feature Learning for Networks. *KDD.*
- **Conclusion**

Machine Learning on Graphs

Classical ML Tasks in Graphs

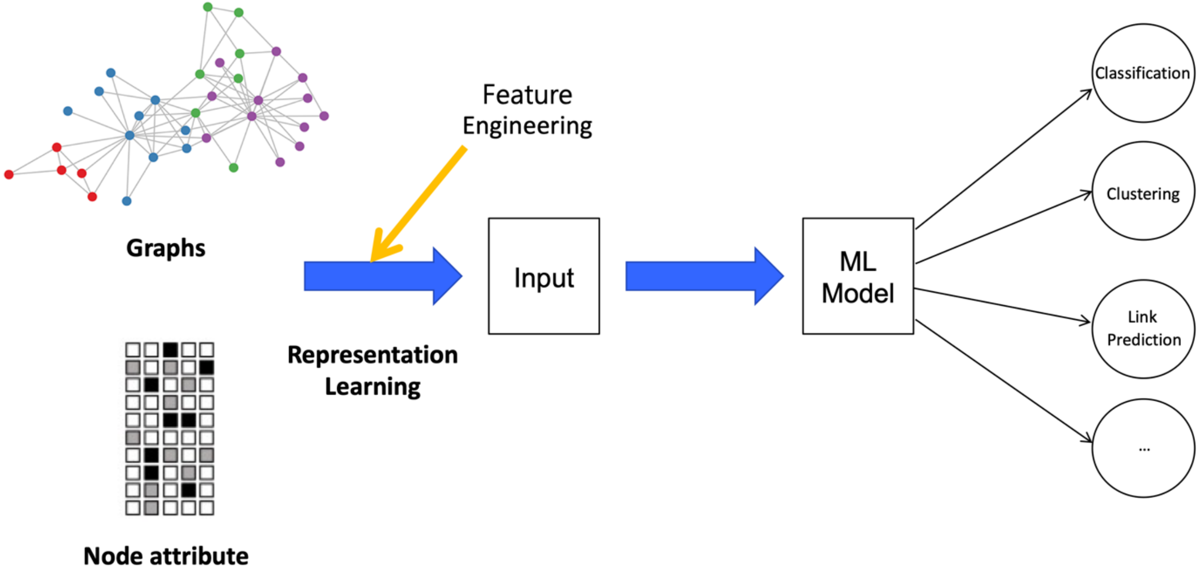
- Node classification
  - Predict the type of a given node
- Link prediction
  - Predict whether two nodes are linked
- Community detection
  - Identify densely linked clusters of nodes
- Network similarity
  - How similar are two (sub)networks





Machine Learning on Graphs

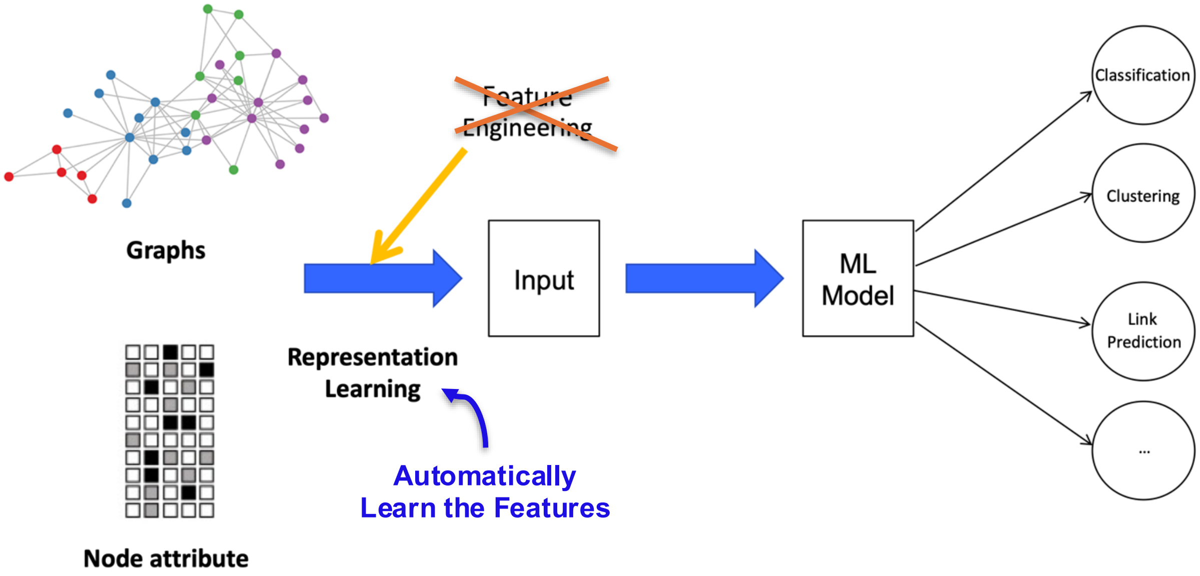
- **(Supervised) Machine Learning Lifecycle**
  - ☐ Require feature engineering every single time
  - ☐ A set of informative, discrimination, and independent features



Machine Learning on Graphs

❖ (Supervised) Machine Learning Lifecycle

- ❑ Require feature engineering every single time
- ❑ A set of informative, discrimination, and independent features



Graphs

Node attribute

~~Feature Engineering~~

Representation Learning

Automatically Learn the Features

Input

ML Model

Classification

Clustering

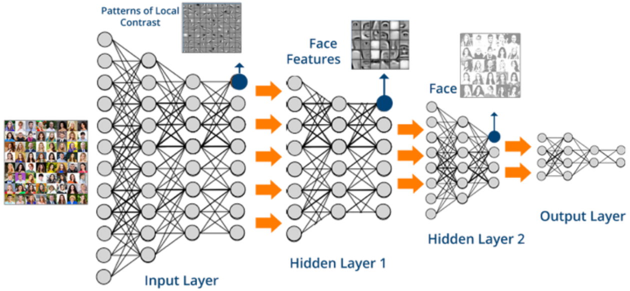
Link Prediction

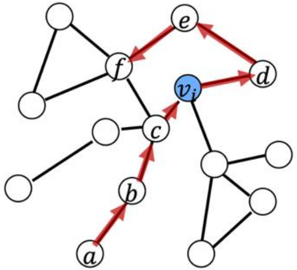
...

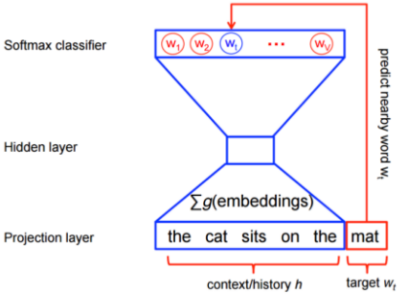
Graph Representation Learning

❖ Graph Representation Learning is hard

- ❑ Images are fixed size
    - ➢ Convolutions (CNNs)
- ❑ Text(sequence) is linear
    - ➢ Sliding window (word2vec)
- ❑ Graphs are neither of these, but
    - ➢ Complex topographic structure
    - ➢ No fixed node ordering or reference point
    - ➢ Often dynamic and have multimodal features

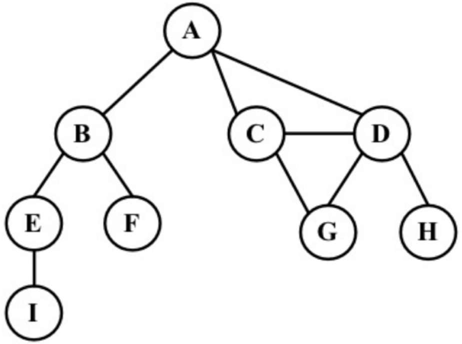






Traditional Machine Learning on Graphs

Graph Representation



$$\begin{array}{c|ccccccccc}
 & A & B & C & D & E & F & G & H & I \\
\hline
A & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
B & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
C & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
D & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
E & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
F & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
H & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
I & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}$$

Adjacency matrix

Problems

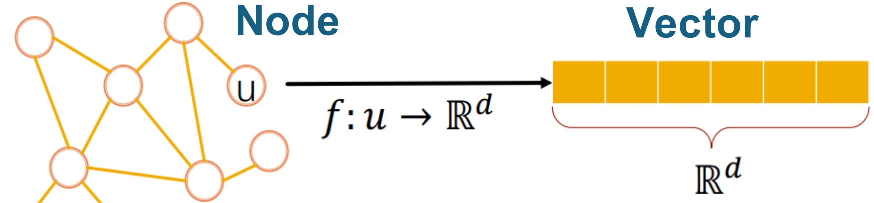
- Suffer from data sparsity
- Suffer from high dimensionality
- High complexity for computation
- Does not represent “semantics”
- …

How to effectively and efficiently represent graphs is the key!

Feature Learning in Graphs

Automatic Learning Feature Representations

- Efficient **task-independent** feature learning with graphs
- Map each node in the network into **embedding** space



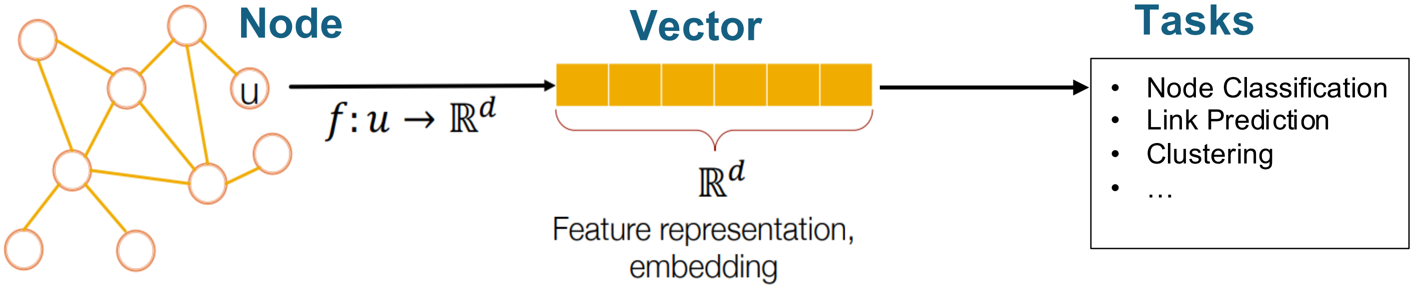
$f: u \to \mathbb{R}^d$

$\mathbb{R}^d$

Feature representation, embedding

Graph Representation Learning

- **Why embedding?**
  - **Similarity in the embedding space ≈ Similarity in the original network**
  - Similar nodes in a network have similar vector representations
  - Encode network information and generate node representation



Node

Vector

Tasks

$f: u \rightarrow \mathbb{R}^d$

$\mathbb{R}^d$

Feature representation, embedding

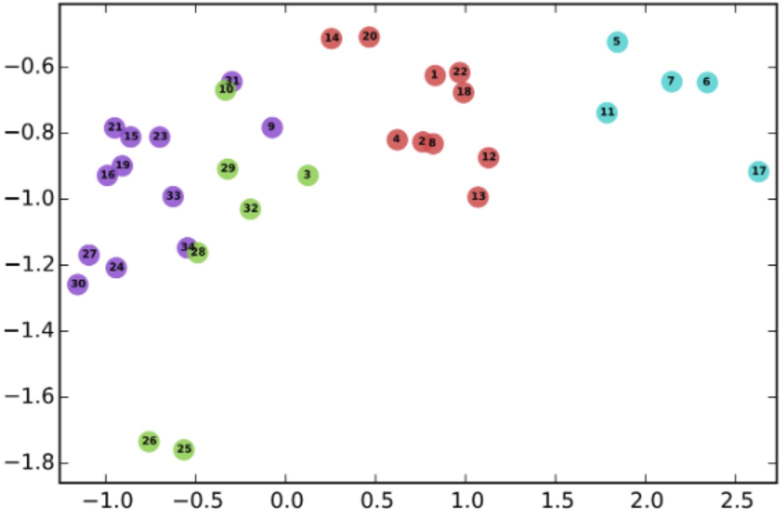
- Node Classification
- Link Prediction
- Clustering
- …

Graph Representation Learning

❖ **Example Node Embedding**

Input

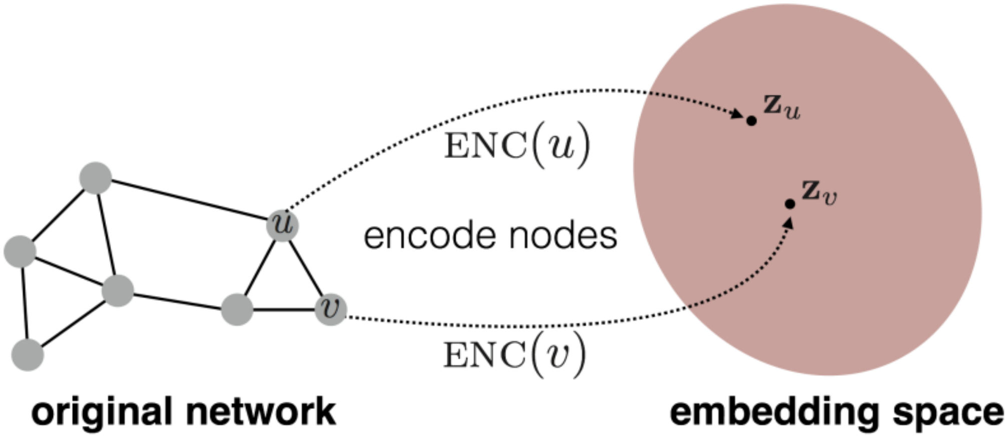
Node Embedding



Output

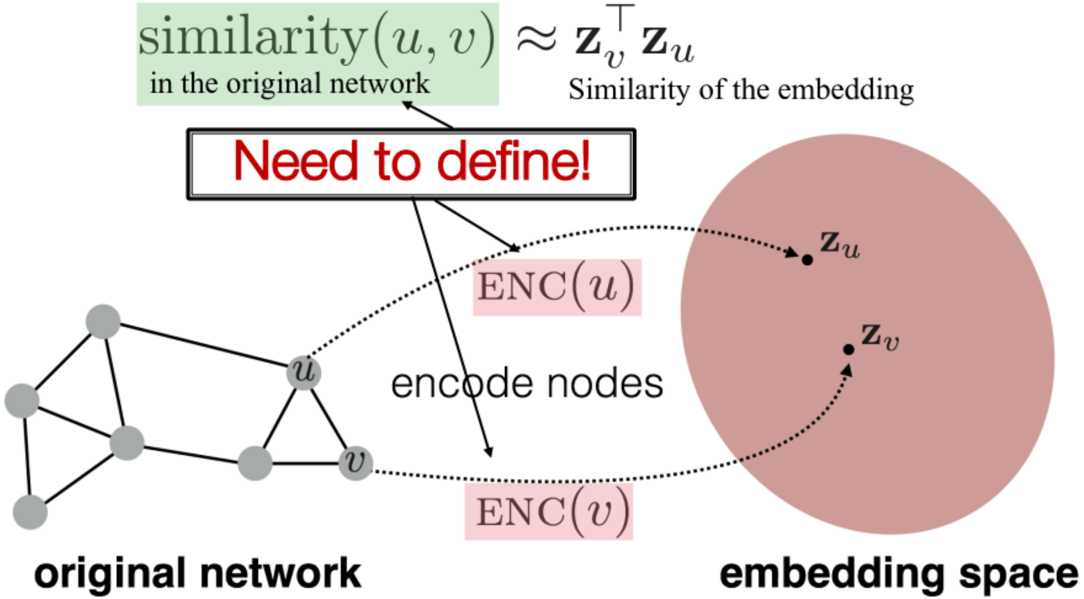
Node Embeddings

- **Encode nodes as low-dimensional vectors**
  - Project nodes into a latent space
  - Summarize their graph position and the structure of their local graph neighborhood
- **Goal:**
  - Encode nodes that similarity in the embedding space approximates similarity in the original network



Node Embeddings

- **Goal:** Encode nodes with similar network neighborhoods close in the feature space



$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

in the original network

Similarity of the embedding

Need to define!

$\text{ENC}(u)$

$\text{ENC}(v)$

encode nodes

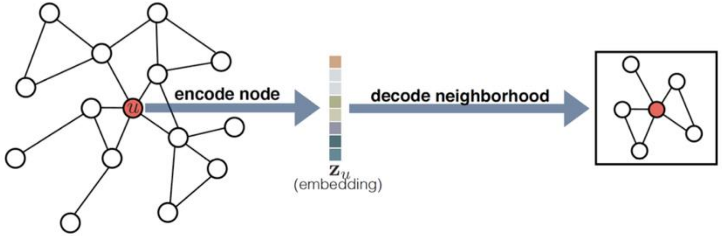
Learning Node Embeddings

- **Encoder** ⟶ **DeepWalk, node2vec, etc.**
  - ❑ A mapping from nodes to unique embeddings (low-dimensional) vector
- **Define a node similarity function**
  - ❑ A measure of similarity in the original network
- **Decoder**
  - ❑ A mapping from embeddings to the similarity score
- **Optimize the parameters of the encoder**
  - ❑ So that:

Decoder

$$\underbrace{\text{similarity}(u, v)}_{\text{in the original network}} \approx \underbrace{\boxed{\mathbf{z}_v^\top \mathbf{z}_u}}_{\text{Similarity of the embedding}}$$

?



Learning Node Embeddings

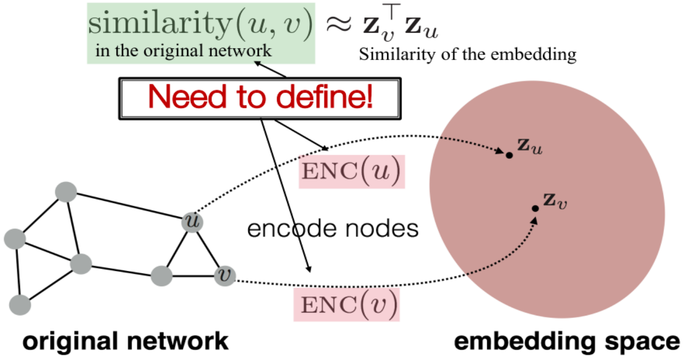
How to Define Encoder

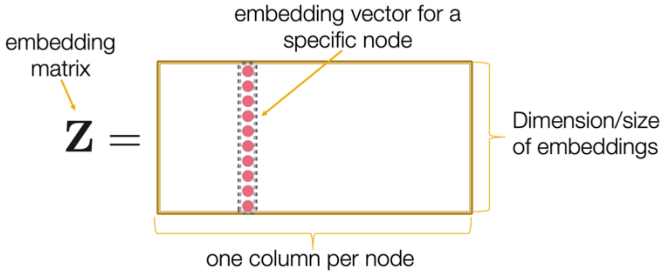
- A function that maps nodes $v \in V$ to vector embeddings $z \in R^d$

$$ENC: V \rightarrow R^d$$

- "Shallow" Encoding
  - An encoder function is simply an embedding-lookup

$$ENC(v) = Z[v]$$

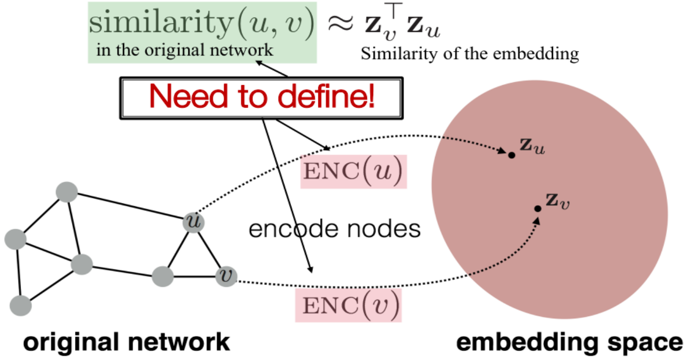




Learning Node Embeddings

❖ How to Define Node Similarity

- ❑ Should two nodes have a similar embeddings if they
  - ➢ Are connected?
  - ➢ Share neighbors?
  - ➢ Have similar "structural roles"?
  - ➢ etc.



- ❑ One of Node Similarity definitions is **random walks approach**

Node embeddings are optimized so that two nodes have similar embeddings if they tend to co-occur on short random walks over the graph

Random Walk Approaches

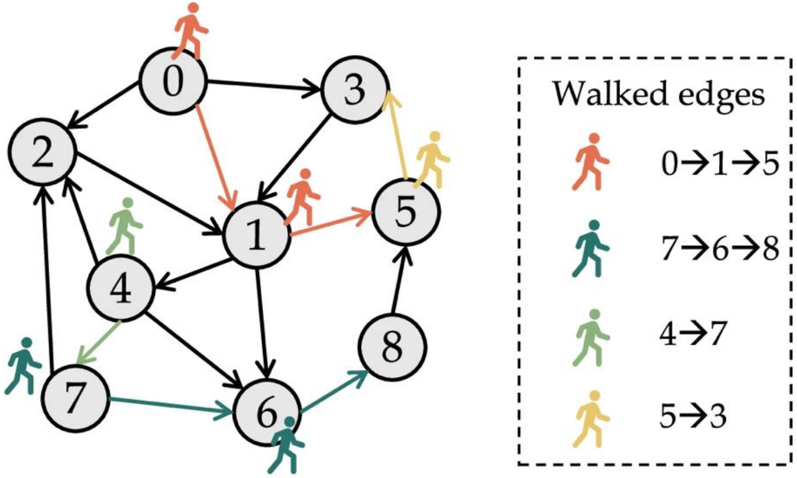
- Node Embeddings

❖ Notation

- ❑ Vector $\boldsymbol{z}_u$: **Embedding** of node u
- ❑ Probability $P(v|\boldsymbol{z}_u)$: (Predicted) **Probability** of visiting node v on random walks starting from node u

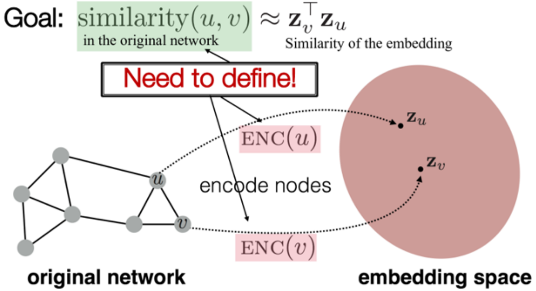
❖ Random Walk on the Graph

- ❑ Given a graph and a starting point
- ❑ Select a neighbor at random
- ❑ Move to this neighbor
- ❑ Then select a neighbor of this point at random
- ❑ And move to it, etc.
- ❑ **The (random) sequence of points visited this way**

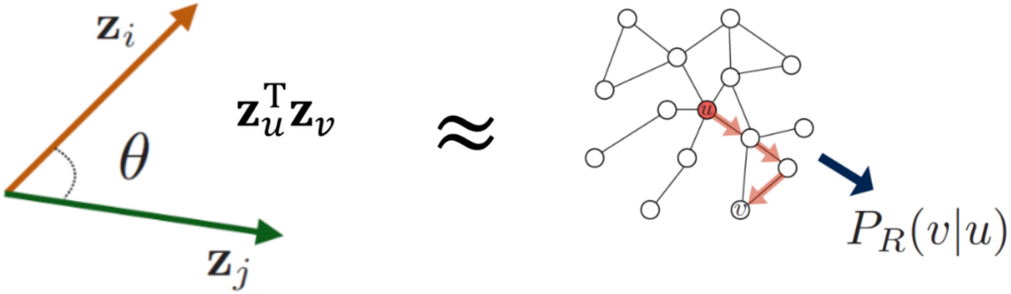


Random Walk Approaches

❖ Random Walk Embeddings



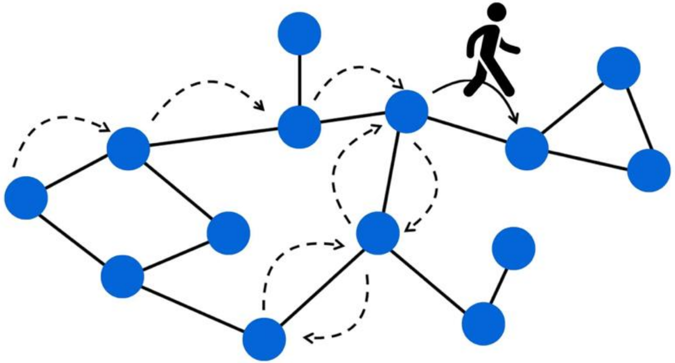
$\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v \approx$ probability that u and v co-occur on a random walk over the graph

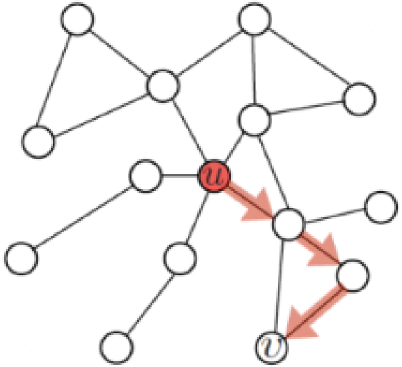


- ❑ **Similarity in embedding space encodes random walk “similarity”**

Why Using Random Walk?

- **Simply define nodes are similar if they are connected**
- **Why random walk?**
  - Expressivity: Random walk incorporates both **local** and **higher-order multi-hop** neighborhood
  - Efficiency: Do not need to consider all node pairs at training state;
    - Only need to consider node pairs on the random walks





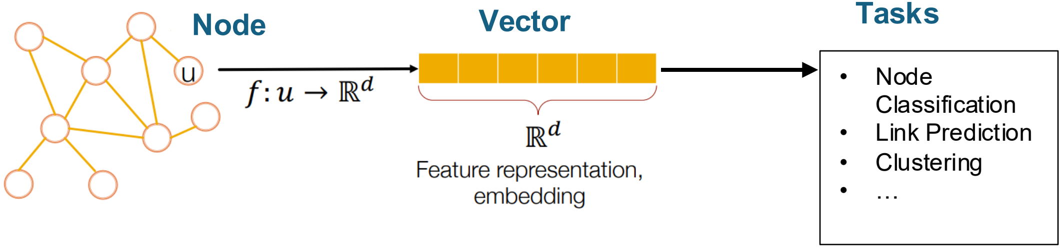
Feature Learning Framework

- **Given $G = (V, E)$**
- **Goal: Learning a mapping $f: u \rightarrow R^d$**
  - $f(u) = z_u$
- **Log-likelihood Objective:** $$\max_f \sum_{u \in V} \log \mathrm{P}(N_{\mathrm{R}}(u) | \mathbf{z}_u)$$
  - $N_R(u)$: the neighborhood of node u by strategy R
- **Equivalently, loss function:** $$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$ $$\boxed{P(v|\mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_n)}}$$
- **Given node u, Learn feature representations** $N_R(u)$
  - Predictive of the nodes in its random walk neighborhood

Feature Learning Framework

- **Unsupervised/self-supervised way of learning node embeddings**
  - Not utilizing node labels/features
  - Directly estimate a set of coordinates of a node
  - So that some aspect of the network structure is preserved

- **These embeddings are task independent**
  - Not trained for a specific task
  - But can be used for any task



Node

$f: u \rightarrow \mathbb{R}^d$

Vector

$\mathbb{R}^d$

Feature representation, embedding

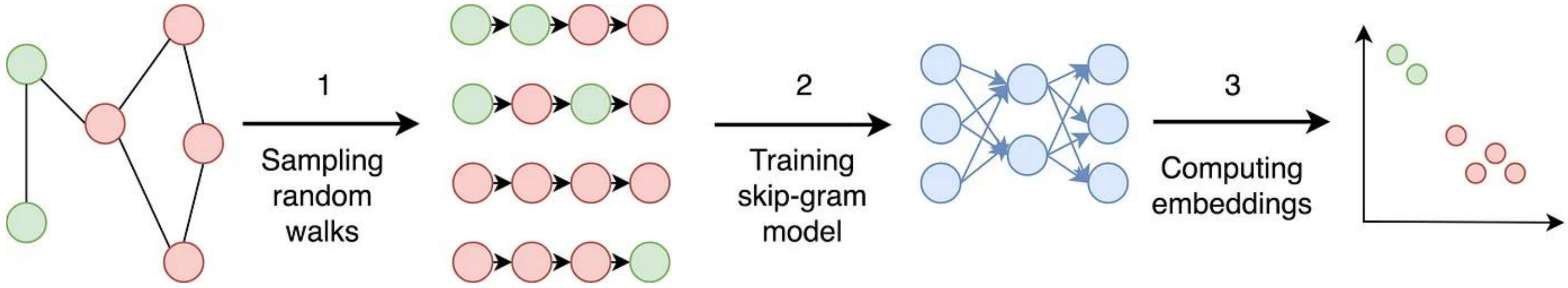
Tasks

- Node Classification
- Link Prediction
- Clustering
- …

DeepWalk

- **Learn Social Representations of a Graph's Vertices**
  - Latent features of the vertices that capture neighborhood similarity and **community membership**

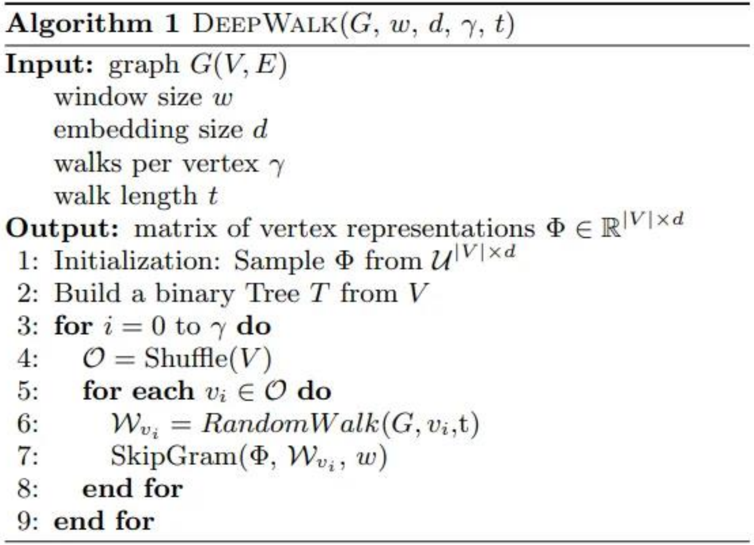
- **Run fixed-length, unbiased random walks starting from each node**



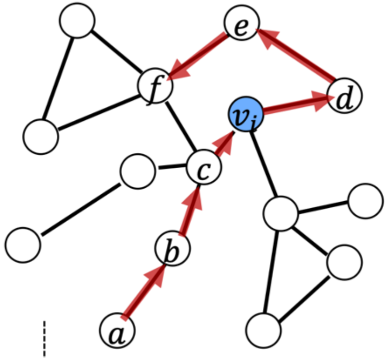
Overview of DeepWalk

DeepWalk

- **Algorithm: Two Parts**
- **(1) A Random Walk Generator**
- (2) AN Update Procedure



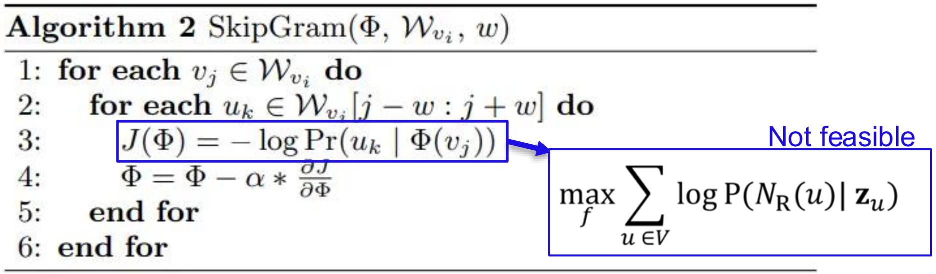
```
Algorithm 1 DeepWalk($G$, $w$, $d$, $\gamma$, $t$)
Input: graph $G(V, E)$
    window size $w$
    embedding size $d$
    walks per vertex $\gamma$
    walk length $t$
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$
Initialization: Sample $\Phi$ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree $T$ from $V$
for $i = 0$ to $\gamma$ do
    $\mathcal{O}$ = Shuffle($V$)
    for each $v_i \in \mathcal{O}$ do
        $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
        SkipGram($\Phi$, $\mathcal{W}_{v_i}$, $w$)
    end for
end for
```



Example seq	$a \to b \to c \to v_i \to d \to e \to f$
Window size=2	$a \mid\to b \to c \to v_i \to d \to e \mid\to f$
Center node	v_i
Neighborhood	$N(v_i) = b, c, d, e$
Observation o	$o = (N(v_i), v_i)$=($\{b, c, d, e\}$,v_i)

DeepWalk

- **Algorithm: Two Parts**
- (1) A Random Walk Generator
- **(2) AN Update Procedure**



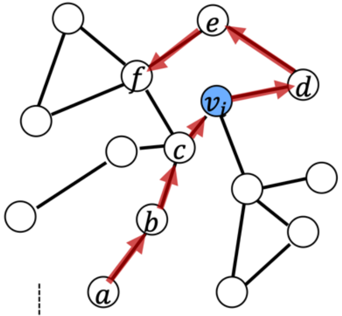
Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

```
for each $v_j \in \mathcal{W}_{v_i}$ do
    for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ do
        $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
        $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
    end for
end for
```

Not feasible

$$\max_{f} \sum_{u \in V} \log \mathrm{P}(N_{\mathrm{R}}(u) | \mathbf{z}_u)$$

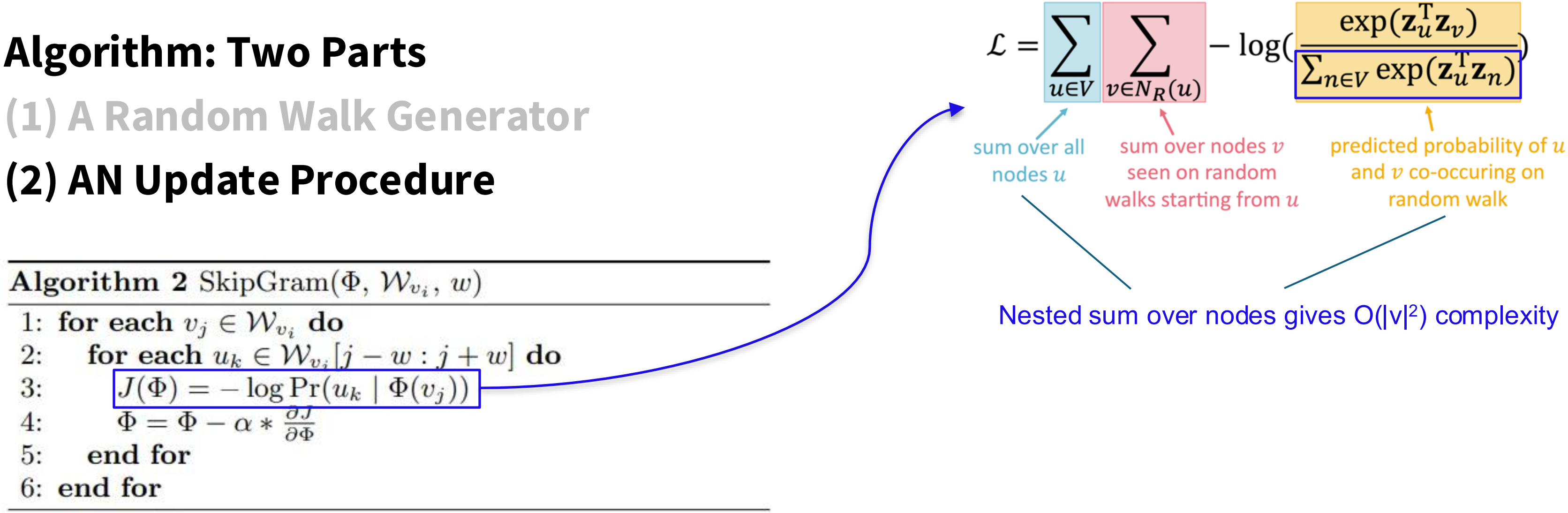
- ❑ Maximize the probability of its neighbors in the window size



Example seq	$a \to b \to c \to v_i \to d \to e \to f$
Window size=2	$a \to b \to c \to v_i \to d \to e \to f$
Center node	v_i
Neighborhood	$N(v_i) = b, c, d, e$
Observation o	$o = (N(v_i), v_i)$=($\{b, c, d, e\}$,v_i

DeepWalk

- **Algorithm: Two Parts**
- (1) A Random Walk Generator
- **(2) AN Update Procedure**



Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

```
for each $v_j \in \mathcal{W}_{v_i}$ do
    for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ do
        $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
        $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
    end for
end for
```

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\boxed{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}}\right)$$

sum over all nodes u

sum over nodes v seen on random walks starting from u

predicted probability of u and v co-occuring on random walk

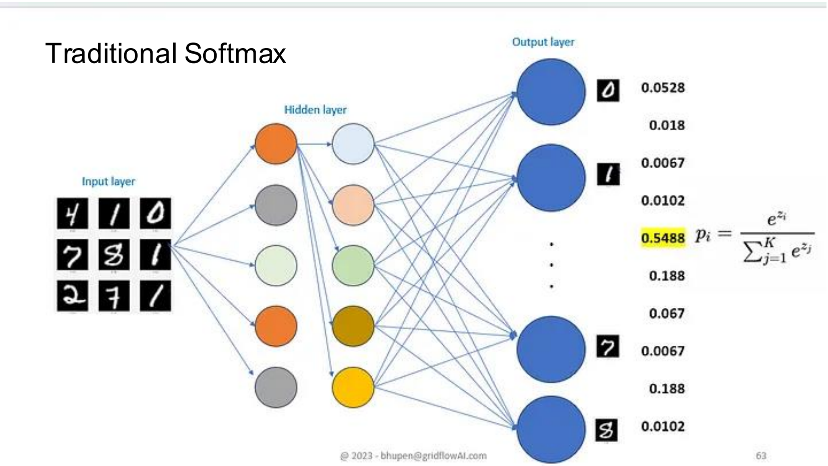
Nested sum over nodes gives O(|v|^2) complexity

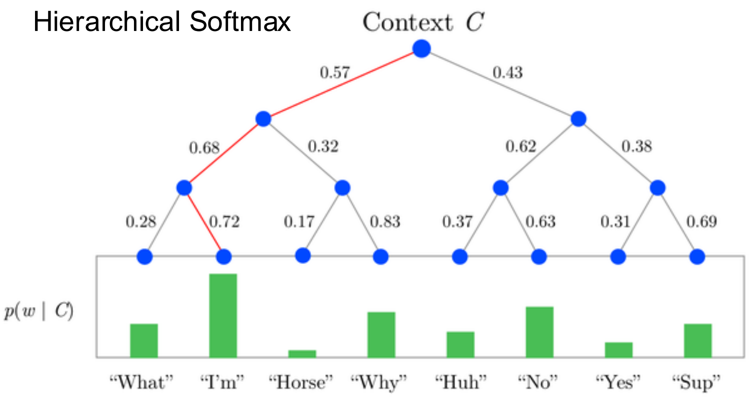
- ❑ Maximize the probability of its neighbors in the window size
- ❑ In scenarios involving large target samples, computation SoftMax probabilities becomes expensive
- ❑ Instead use the **Hierarchical SoftMax**

DeepWalk

❖ Why use Hierarchical SoftMax?

- ❑ Address this issue by organizing the output space into binary tree
- ❑ Accelerate the computation ($O(n) \ \rightarrow O(\log n)$)
- ❑ Still give the behavior and results of SoftMax function

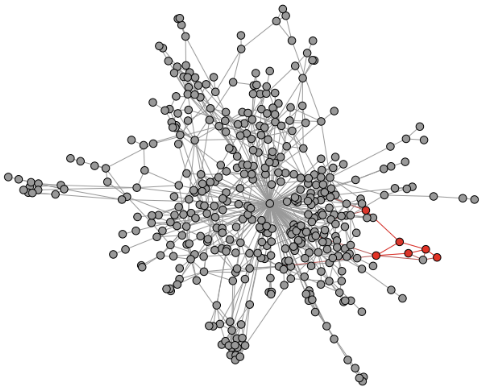




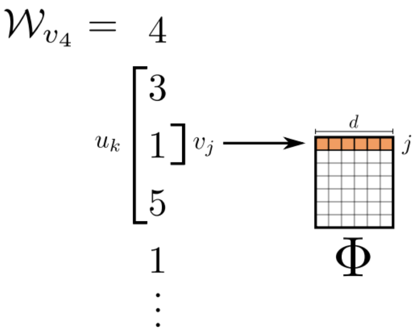
DeepWalk

❖ Hierarchical SoftMax

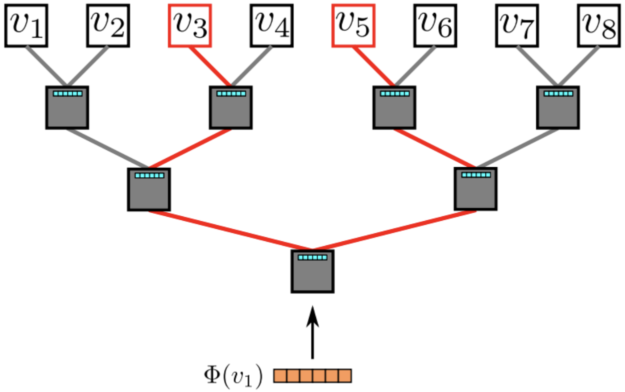
- ❑ Assign the vertices to the leaves of a binary tree
- ❑ The prediction problem turns into maximizing the probability of a specific path in the tree
- ❑ Huffman coding is used to reduce the access time of frequent elements in the tree



(a) Random walk generation.



(b) Representation mapping.

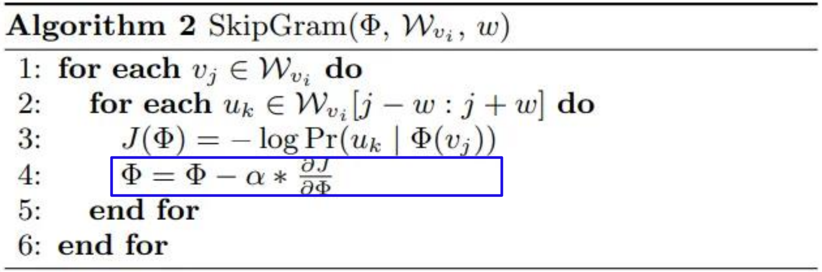


(c) Hierarchical Softmax.

DeepWalk

❖ Optimization

- ❑ Model parameter set, $\theta = \{\varphi, \psi\}$
- ❑ Use SGD to optimize these parameters (Line 4)
- ❑ Update by using back-propagation algorithm



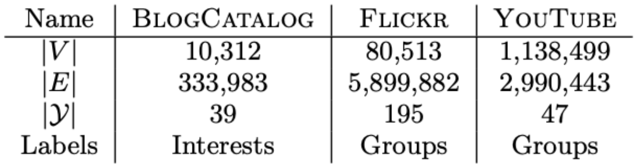
Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

```
1: for each $v_j \in \mathcal{W}_{v_i}$ do
2:     for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ do
3:         $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
4:         $\boxed{\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}}$
5:     end for
6: end for
```

DeepWalk

- Experiment

❖ Datasets



Name	BlogCatalog	Flickr	YouTube
$\lvert V \rvert$	10,312	80,513	1,138,499
$\lvert E \rvert$	333,983	5,899,882	2,990,443
$\lvert \mathcal{Y} \rvert$	39	195	47
Labels	Interests	Groups	Groups

Table 1: Graphs used in our experiments.

❖ Downstream Tasks

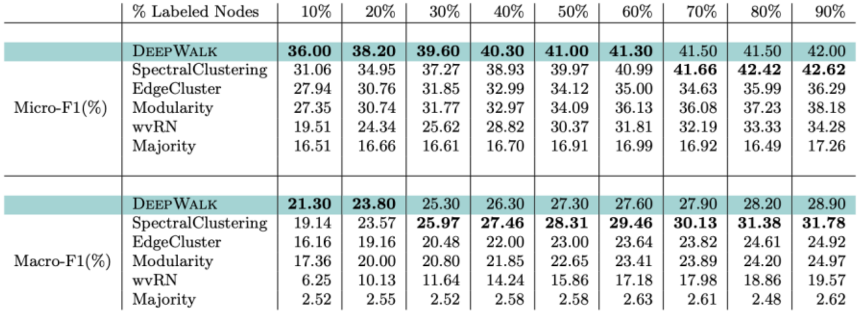
- ❑ Multi-Label Classification

DeepWalk

- Experiment

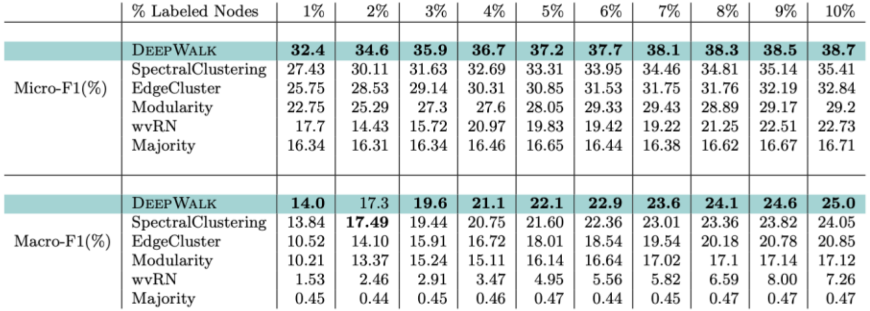
Result

- Still outperforms all baselines although labeled data is sparse
- Significantly outperforms for creating graph representation in YouTube networks



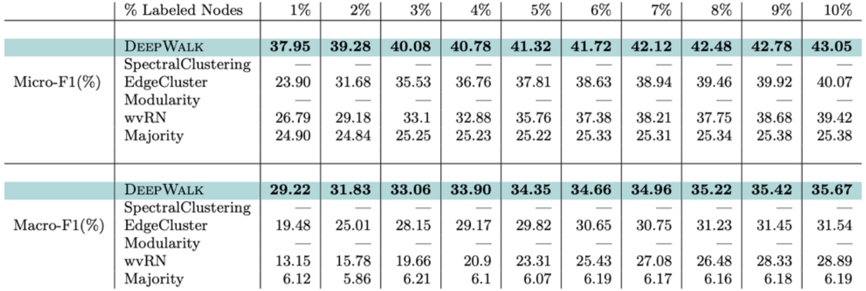
	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

Table 2: Multi-label classification results in BlogCatalog



	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
Macro-F1(%)	DeepWalk	**14.0**	17.3	**19.6**	**21.1**	**22.1**	**22.9**	**23.6**	**24.1**	**24.6**	**25.0**
	SpectralClustering	13.84	**17.49**	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
	EdgeCluster	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Modularity	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.1	17.14	17.12
	wvRN	1.53	2.46	2.91	3.47	4.95	5.56	5.82	6.59	8.00	7.26
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

Table 3: Multi-label classification results in Flickr



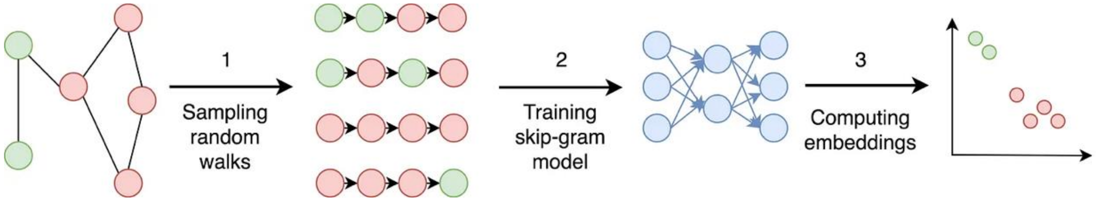
	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
Macro-F1(%)	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

Table 4: Multi-label classification results in YouTube

DeepWalk

❖ Summary

- ❑ Idea: Run short fixed-length random walks starting from each node on the graph
- ❑ For each node u collect $N\&\,(u)$, the multiset of nodes visited on random walks starting from u
- ❑ Optimize embeddings using Stochastic Gradient Descent

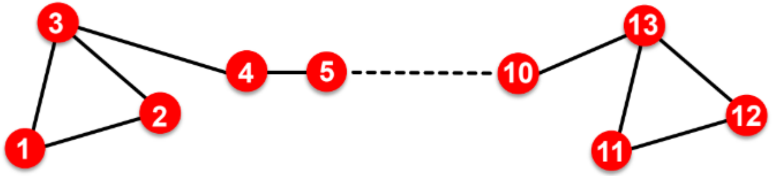


❖ Fail to offer any flexibility in sampling of nodes from a network

- ❑ Not rich enough to embed nodes from **same network community** as well as **nodes with similar structural roles**

node2vec

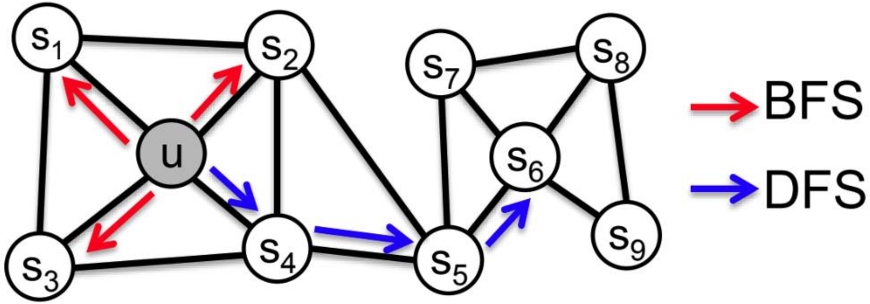
- **Goal**
  - Embed nodes with similar network neighborhoods close in the embedding space
- **Key**
  - Flexible notion of network neighborhood $N(u)$ of node u leads to rich node embeddings
- **Learning Method**
  - Maximum likelihood optimization problem



node2vec

❖ Biased Walks

- ☐ Idea: Use flexible, biased random walks that can trade off between local and global views of the network
- ☐ Two classic strategies to define a neighborhood $N_R(u)$ of a given node u



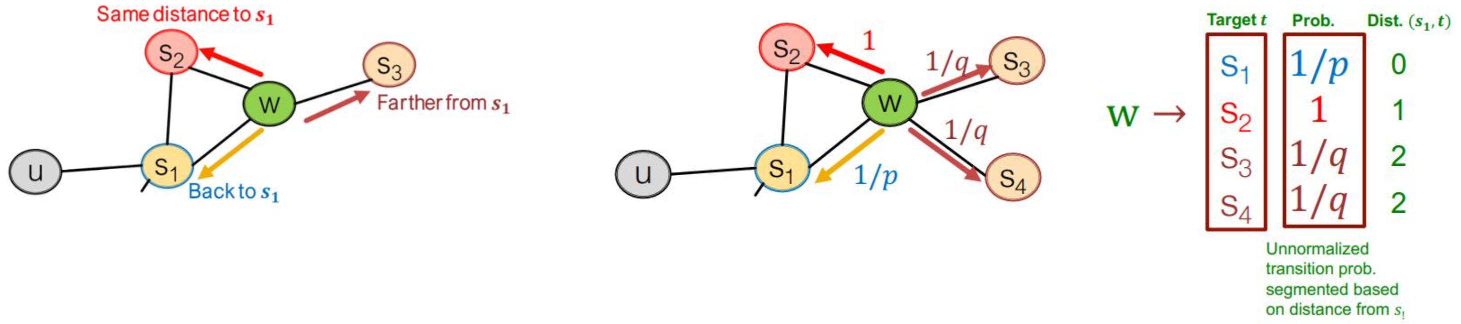
Walk of length ($N_R(u)$ of size 3):

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

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

node2vec

- **Biased 2nd-order random walks explore network neighborhood**
  - Walker just traversed edge (s1, w) and is at w, now he can go:

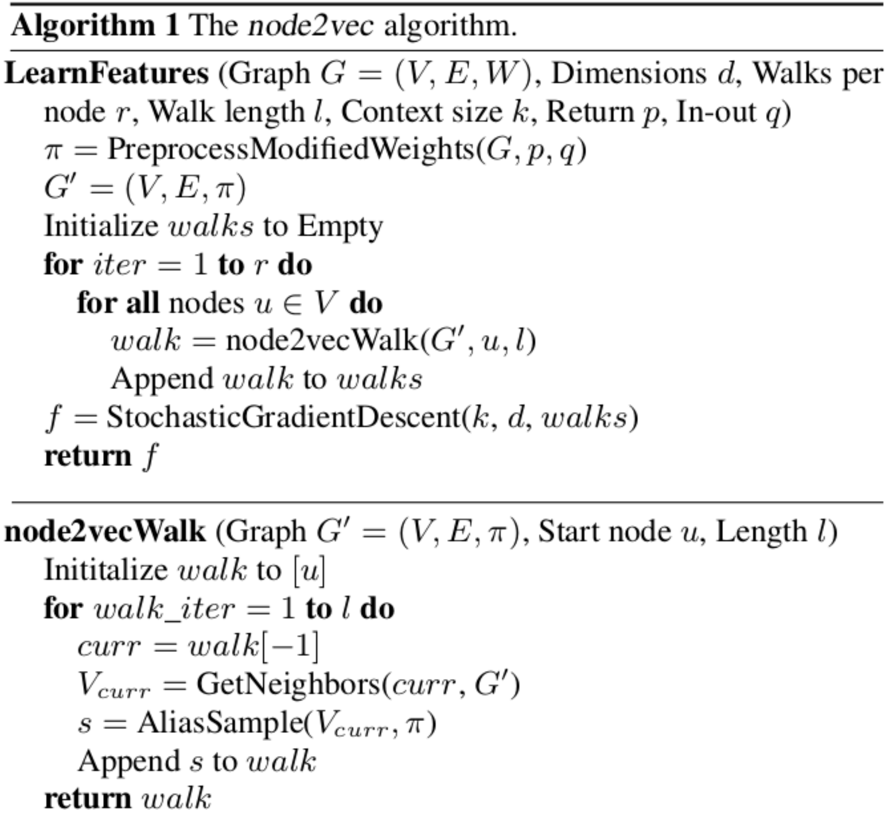


Target t	Prob.	Dist. (s_1, t)
s_1	$1/p$	0
s_2	1	1
s_3	$1/q$	2
s_4	$1/q$	2

Unnormalized transition prob. segmented based on distance from s_1

  - BFS-like walk: Low value of **p**
  - DFS-like walk: Low value of **q**
  - $\boldsymbol{N_R(u)}$: the nodes visited by the biased walk

node2vec

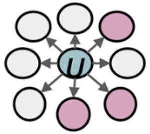


Algorithm 1 The *node2vec* algorithm.

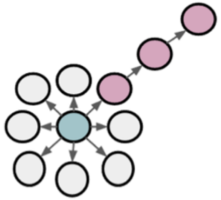
```
LearnFeatures (Graph $G = (V, E, W)$, Dimensions $d$, Walks per
    node $r$, Walk length $l$, Context size $k$, Return $p$, In-out $q$)
    $\pi$ = PreprocessModifiedWeights($G, p, q$)
    $G' = (V, E, \pi)$
    Initialize $walks$ to Empty
    for $iter = 1$ to $r$ do
        for all nodes $u \in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

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

node2vec

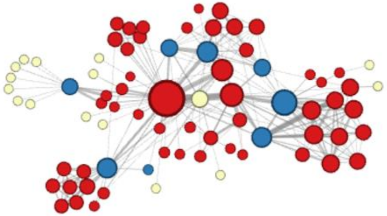
- **BFS vs. DFS**



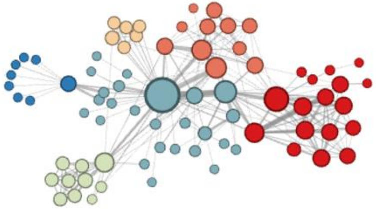
BFS:
Micro-view of neighbourhood



DFS:
Macro-view of neighbourhood



p=1, q=2
Microscopic view of the network neighbourhood



p=1, q=0.5
Macroscopic view of the network neighbourhood

Negative Sampling in Optimization

- **Problem:** Expensive in summing over nodes ($O(|V|^2)$)

$$\mathcal{L} = \sum_{\underline{u \in V}} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v)}{\underline{\sum_{n \in V}} \exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_n)}\right)$$

- **Solution:** Negative Sampling (Softmax → Sigmoid)

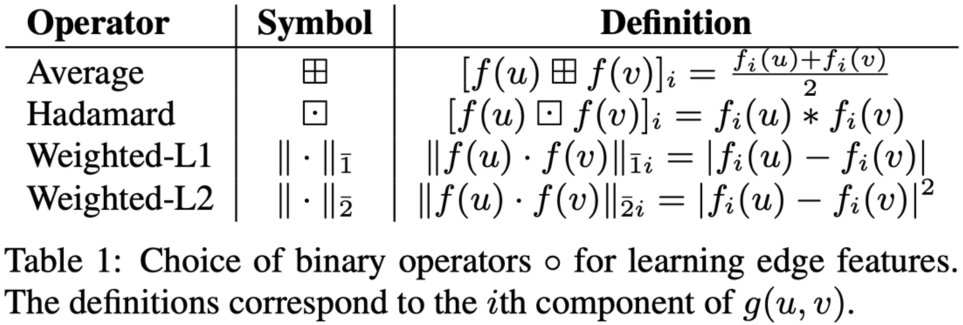
Random distribution over nodes

$$\log\left(\frac{\exp\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v\right)}{\sum_{n \in V} \exp\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_n\right)}\right) \approx \log\left(\sigma\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v\right)\right) - \sum_{i=1}^{k} \log\left(\sigma\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_{n_i}\right)\right), n_i \sim P_V$$

How to Use Node Embeddings

❖ Using embeddings of nodes

- ❑ Clustering/community detection: Cluster points z_i
- ❑ Node Classification: Predict label $f(z_i)$ of node i based on z_i
- ❑ Link Prediction: Predict edge (i, j) based on $f(z_i, z_j)$
  - ➢ Concatenate, Avg, Product, or take a difference between the embeddings



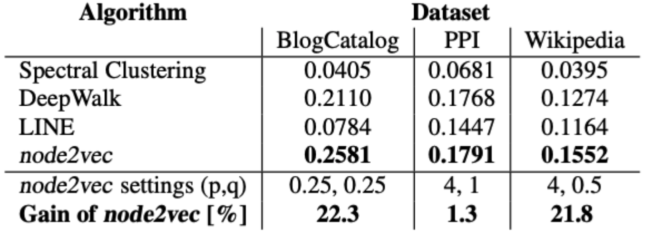
Operator	Symbol	Definition
Average	$\boxplus$	$[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\boxdot$	$[f(u) \boxdot f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\|\cdot\|_{\bar{1}}$	$\|f(u) \cdot f(v)\|_{\bar{1}i} = \lvert f_i(u) - f_i(v) \rvert$
Weighted-L2	$\|\cdot\|_{\bar{2}}$	$\|f(u) \cdot f(v)\|_{\bar{2}i} = \lvert f_i(u) - f_i(v) \rvert^2$

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

node2vec

- Experiments

❖ Result in Multi-label Classification



Algorithm	**Dataset**		
	BlogCatalog	PPI	Wikipedia
Spectral Clustering	0.0405	0.0681	0.0395
DeepWalk	0.2110	0.1768	0.1274
LINE	0.0784	0.1447	0.1164
node2vec	**0.2581**	**0.1791**	**0.1552**
node2vec settings (p,q)	0.25, 0.25	4, 1	4, 0.5
Gain of *node2vec* [%]	**22.3**	**1.3**	**21.8**

Table 2: Macro-F_1 scores for multilabel classification on BlogCatalog, PPI (Homo sapiens) and Wikipedia word cooccurrence networks with 50% of the nodes labeled for training.

❖ More Fine-grained analysis

On varying the amount of labelled data

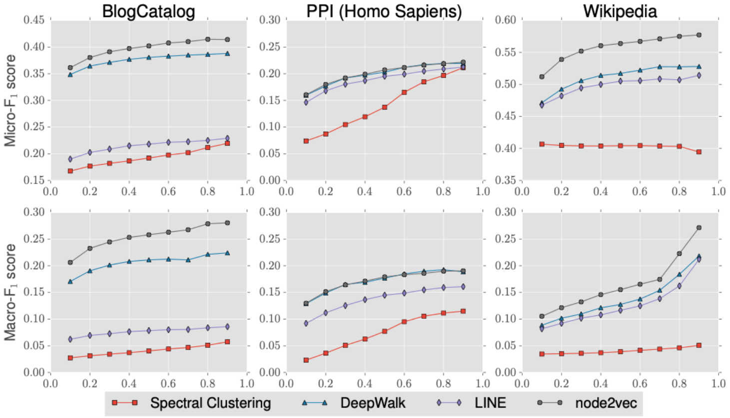


Figure 4: Performance evaluation of different benchmarks on varying the amount of labeled data used for training. The x axis denotes the fraction of labeled data, whereas the y axis in the top and bottom rows denote the Micro-F_1 and Macro-F_1 scores respectively. DeepWalk and *node2vec* give comparable performance on PPI. In all other networks, across all fractions of labeled data *node2vec* performs best.

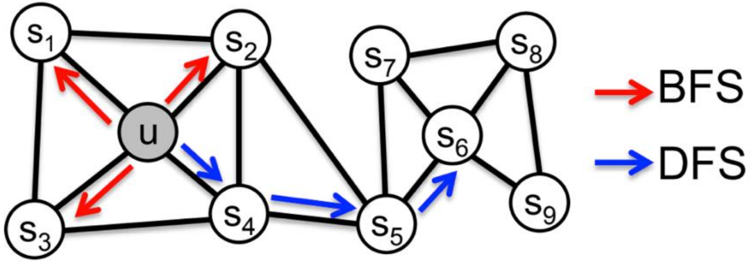
node2vec

Summary

- A flexible neighborhood sampling strategy which allows smoothly interpolate between BFS and DFS

- Developing a flexible biased 2nd order random walk procedure
  - By fine-tuning random walk hyper-parameters to encapsulate more of a homophily or structure equivalence
  - Biased random walks capture diversity of network patterns

- Used generated sequences of nodes as an input to a skip-gram with negative sampling model



Conclusion

- **Basic Idea**
  - ☐ Embed nodes so that distances in embedding space reflect node similarities in the original network

- **Different Notations of Node Similarity**
  - ☐ Adjacency-based (similar if connected)
  - ☐ Multi-hop similarity definitions
  - ☐ Random walk approaches (DeepWalk, node2vec)

- **Which one is better?**
  - ☐ No method wins in all cases
  - ☐ Random walk approaches are generally more efficient

Conclusion

- Random Walk Approaches

❖ Encoder-decoder Framework Perspective

❖ Node Embeddings

- ❑ Map the nodes to embeddings simply as an embedding lookup
- ❑ Train a unique embeddings for each node in the graph

❖ Node Similarity Measure

- ❑ Uniform random walks (DeepWalk)
- ❑ Flexible biased random walks (node2vec)

❖ Optimization Problem

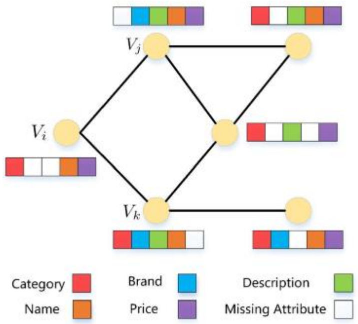
- ❑ DeepWalk: Hierarchical softmax
- ❑ node2ves: Noise contrastive approach (using negative sampling)

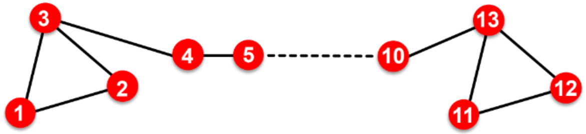
Conclusion

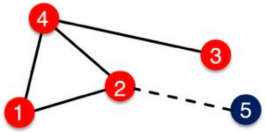
- Random Walk Approaches

❖ Shallow Embeddings

- ❑ Not share any parameters between nodes in the encoder
- ❑ Not leverage node (edge, graph) features
- ❑ Cannot capture structural similarity
  - ➢ Node 1 and 11 are structurally similar but different embeddings
- ❑ Inherently transductive
  - ➢ Not obtain embeddings for nodes if not in the training set
  - ➢ Cannot apply to new graphs







Training set

❖ To solve these limitation, Graph Neural Networks is introduced

Reference

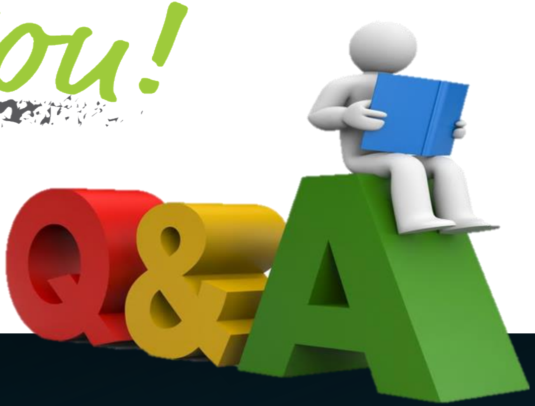
- Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '14). Association for Computing Machinery, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

- Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

- Machine Learning with Graphs Report
  - ☐ https://wandb.ai/syllogismos/machine-learning-with-graphs/reportlist

- William L. Hamilton. (2020). Graph Representation Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 14, No. 3 , Pages 1-159.

Thank You!



HTET ARKAR
hak3601@cau.ac.kr