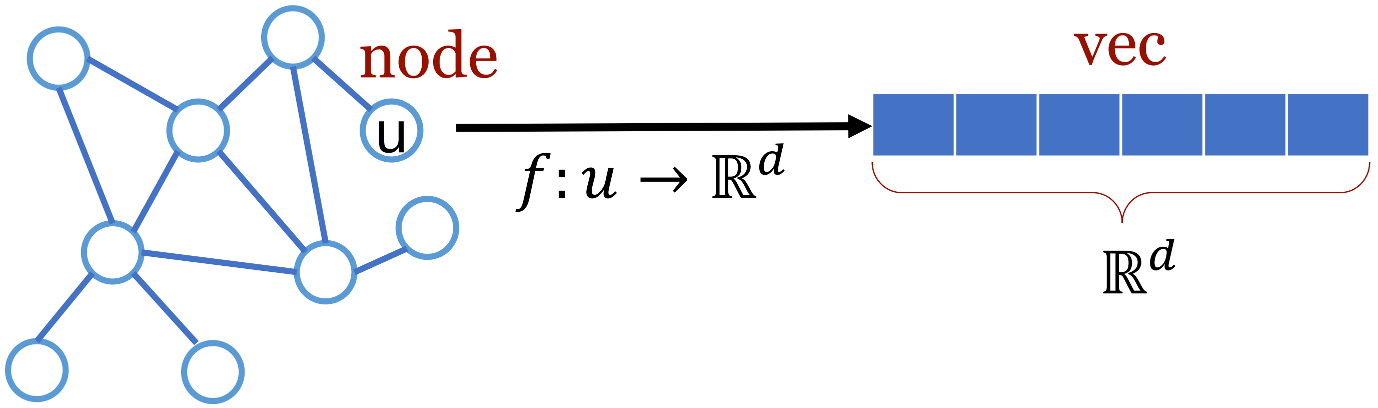
课程安排

- 1. GNN基础
- 2. 基于random walk的图嵌入算法
- 3. GCN
- 4. 研究方向说明
- 5.语言模型GraphSAGE
- 6. Attention机制与GAT
- 7. 实例讲解：HGNN
- 8. 异质图算法
- 9. VGAE和graph GAN
- 10. 实例讲解：DSTG
- 11. GNN模型的解释性
- 12. How to write
- 13. beyondGNN
- 14. 论文选会以及如何让你的论文更容易发表

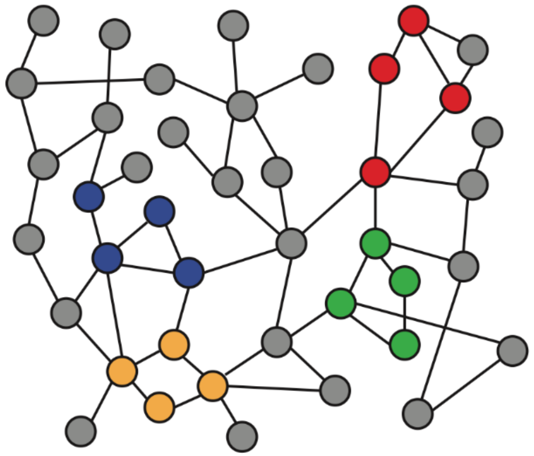
Node Embedding

Feature Learning in Graphs

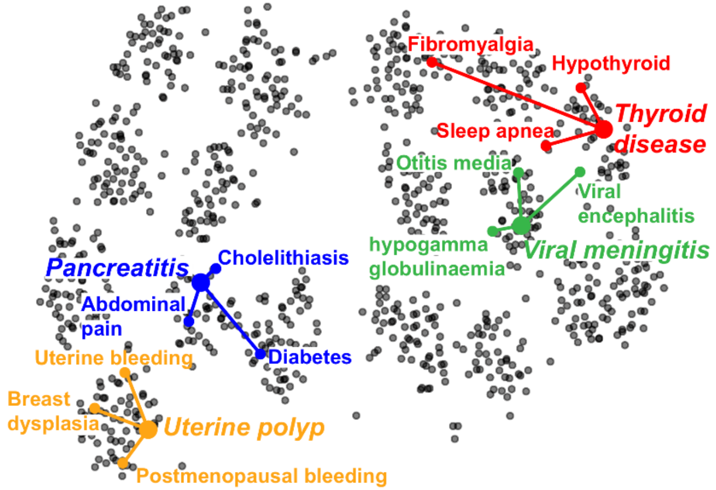
Goal: Efficient task-independent feature learning for machine learning in networks!



Embedding Nodes



Input



Output

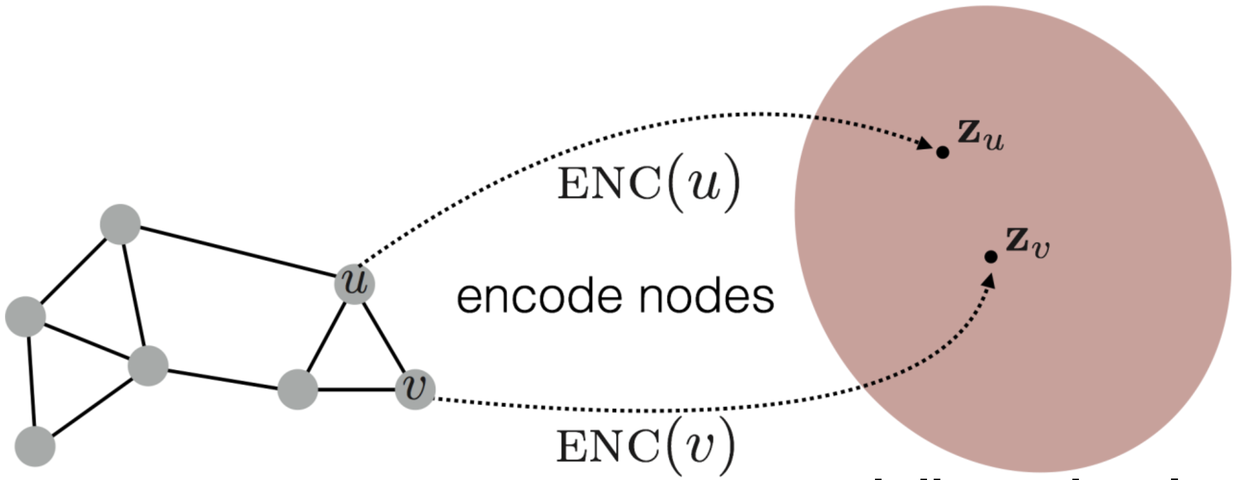
Intuition: Map nodes to d-dimensional embeddings such that similar nodes in the graph are embedded close together

Feature Learning in Graphs

Assume we have a graph $\mathbf{G}$:

- $\mathbf{V}$ is the vertex set
- $\boldsymbol{A}$ is the adjacency matrix (assume binary)
- **No node features or extra information is used!**

Goal: Map nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the network



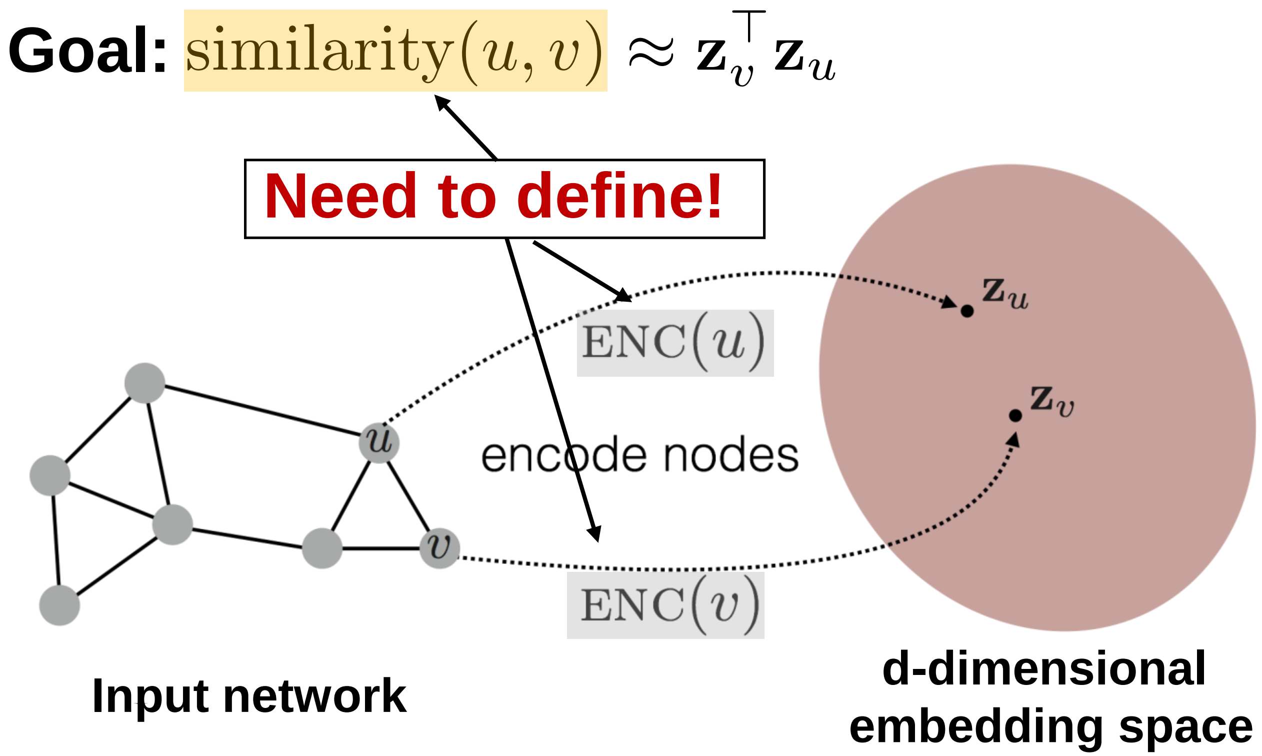
Input network

d-dimensional embedding space

Embedding Nodes

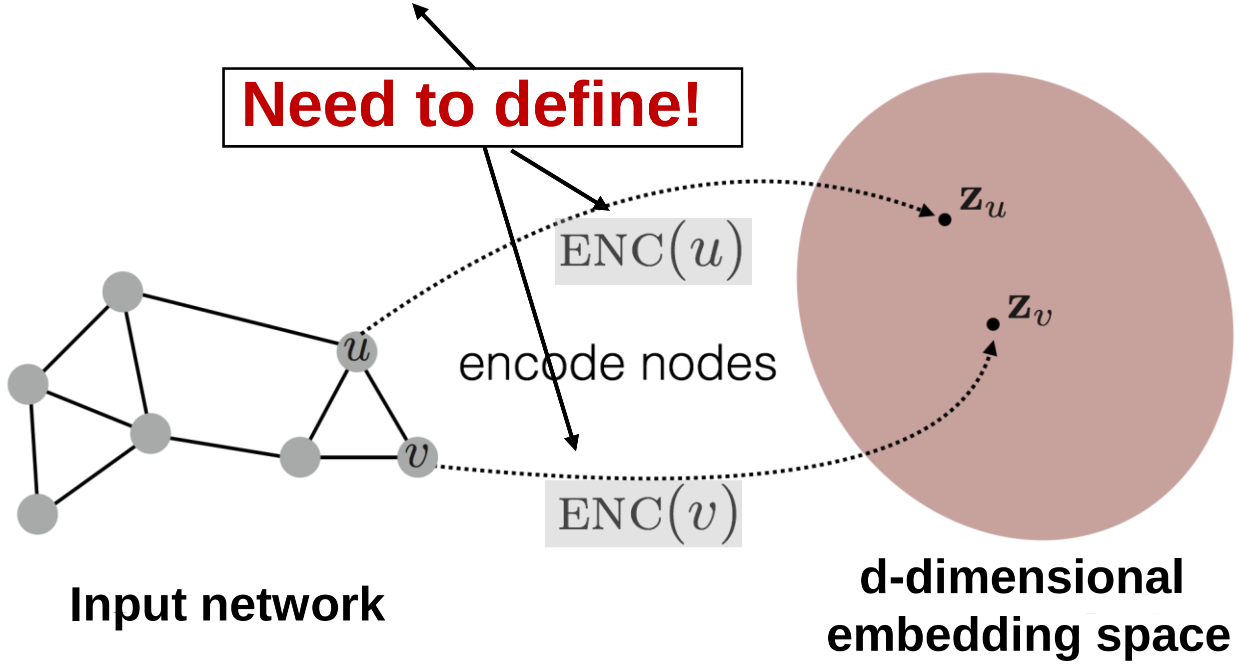
1. **Define an encoder** (a function ENC that maps node u to embedding $\boldsymbol{z}_u$)
2. **Define a node similarity function** (a measure of similarity in the input network)
3. **Optimize parameters of the encoder so that:**

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



Embedding Nodes

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



1. **Encoder** maps a node to a d-dimensional vector:

$$\text{ENC}(v) = \mathbf{z}_v$$

node in the input graph

d-dimensional embedding

2. **Similarity function** defines how relationships in the input network map to relationships in the embedding space:

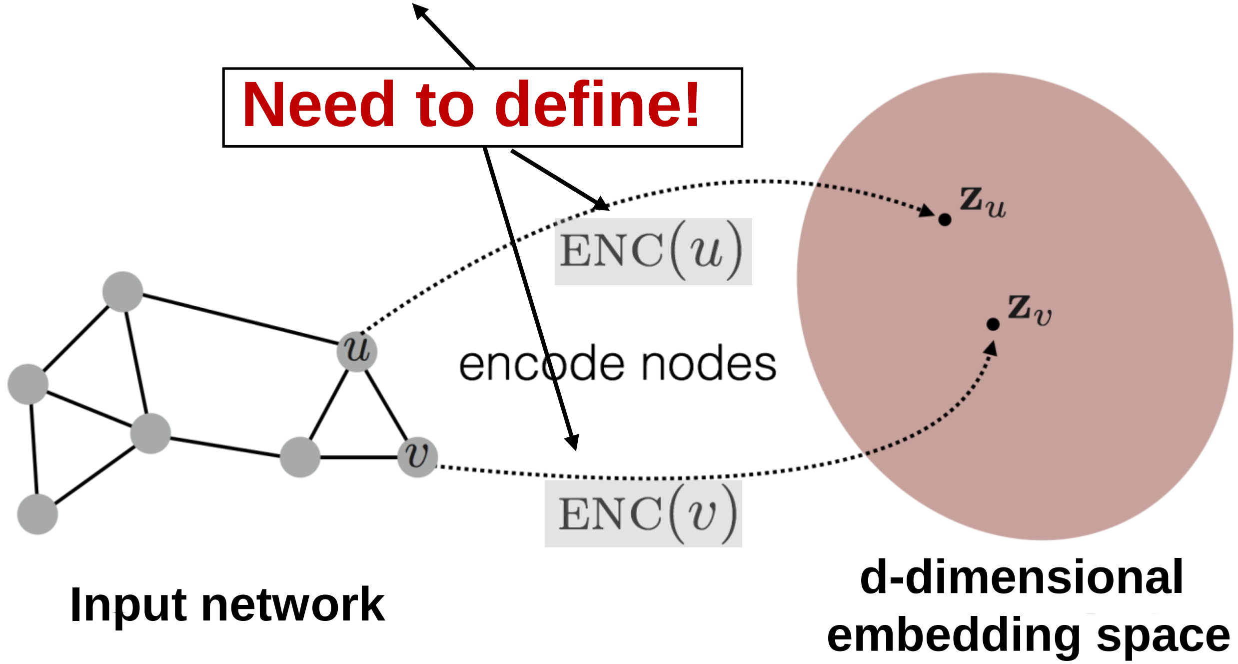
$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

Similarity of u and v in the network

dot product between node embeddings

Embedding Nodes

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



- Many methods use similar encoders:
  - node2vec, DeepWalk, LINE, struc2vec
- These methods use different notions of node similarity:
  - Two nodes have similar embeddings if:
    - they are connected?
    - they share many neighbors?
    - they have similar local network structure?
    - etc.

Adjacency-based Similarity

- **Similarity function** is the edge weight between u and v in the network
- **Intuition:** Dot products between node embeddings approximate edge existence

$$\boxed{\mathcal{L}} = \boxed{\sum_{(u,v)\in V\times V}} \| \boxed{\mathbf{z}_u^\top \mathbf{z}_v} - \boxed{\mathbf{A}}_{u,v} \|^2$$

loss (what we want to minimize)

sum over all node pairs

embedding similarity

(weighted) adjacency matrix for the graph

Adjacency-based Similarity

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \|\mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v}\|^2$$

- Find embedding matrix $\mathbf{Z} \in \mathbb{R}^{d \times |V|}$ that minimizes the loss $\mathcal{L}$:
  - Option 1: Stochastic gradient descent (SGD)
    - Highly scalable, general approach
  - Option 2: Solve matrix decomposition solvers
    - e.g., SVD or QR decompositions
    - Need to derive specialized solvers

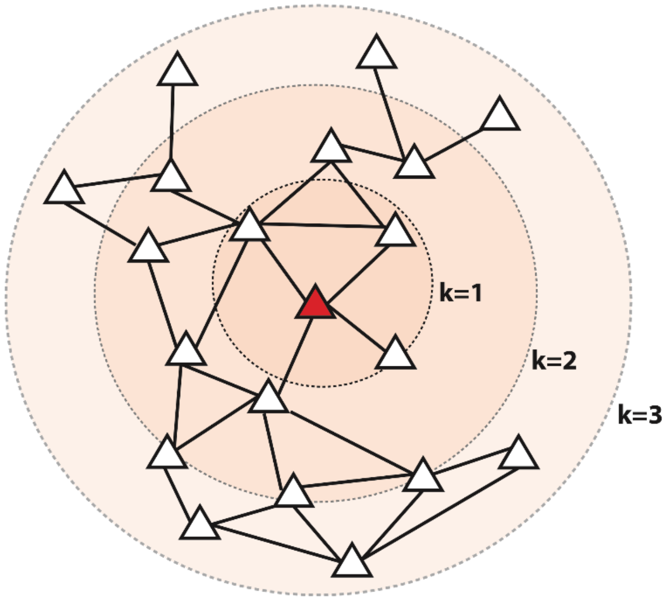
Random Walk

Material based on:

- Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations. *KDD.*
- Grover et al. 2016. node2vec: Scalable Feature Learning for Networks. *KDD.*
- Ribeiro et al. 2017. struc2vec: Learning Node Representations from Structural Identity. *KDD.*

Multi-Hop Similarity

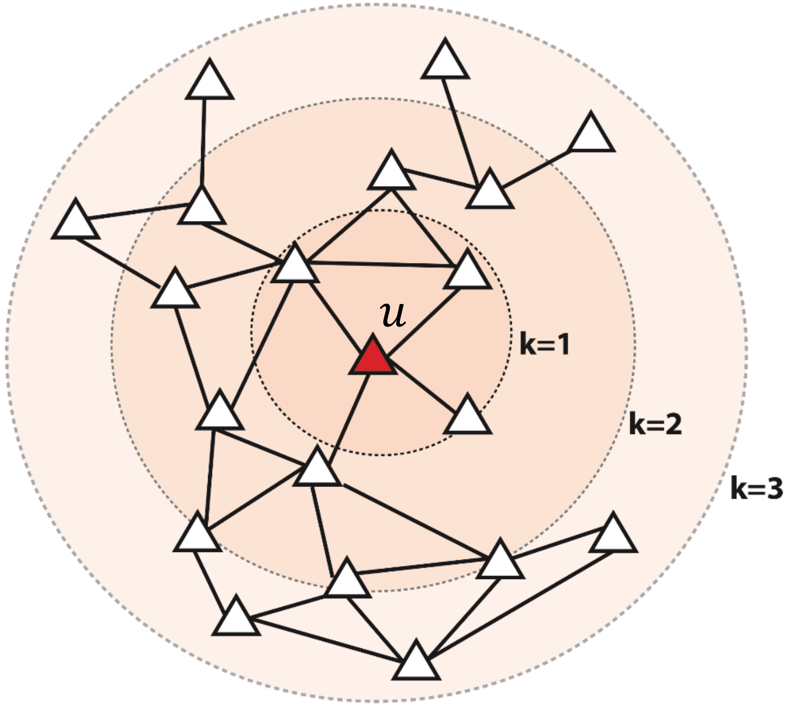
Idea: Define node similarity function based on higher-order neighborhoods



- **Red:** Target node
- **k=1:** 1-hop neighbors
  - $\boldsymbol{A}$ (i.e., adjacency matrix)
- **k= 2:** 2-hop neighbors
- **k=3:** 3-hop neighbors

How to stochastically define these higher-order neighborhoods?

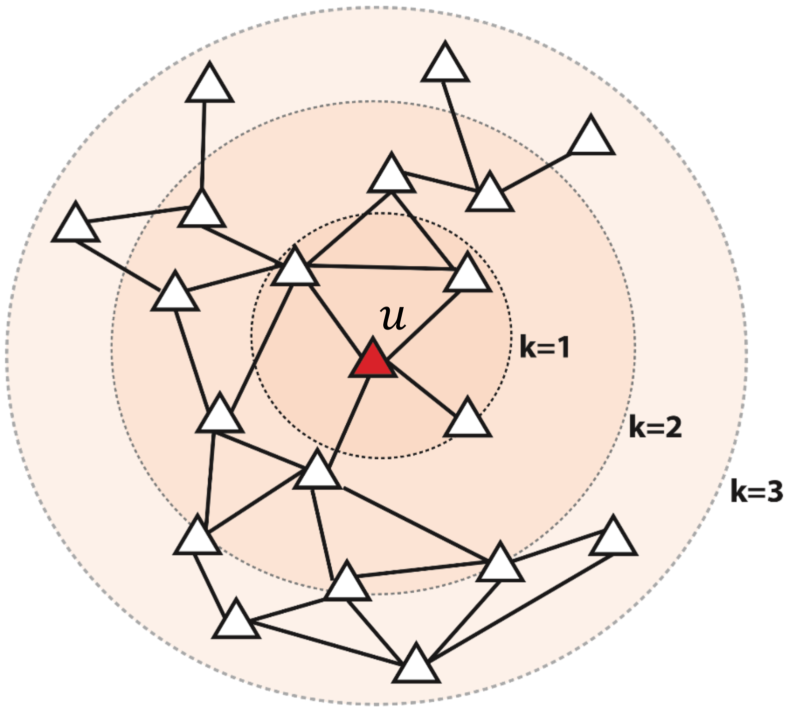
Unsupervised Feature Learning



- 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?
  - $N_R(u)$ ⋯ neighbourhood of u obtained by some strategy R

Randomwalk：
- 节点u开始形成100个路径
- 每个路径跳3次

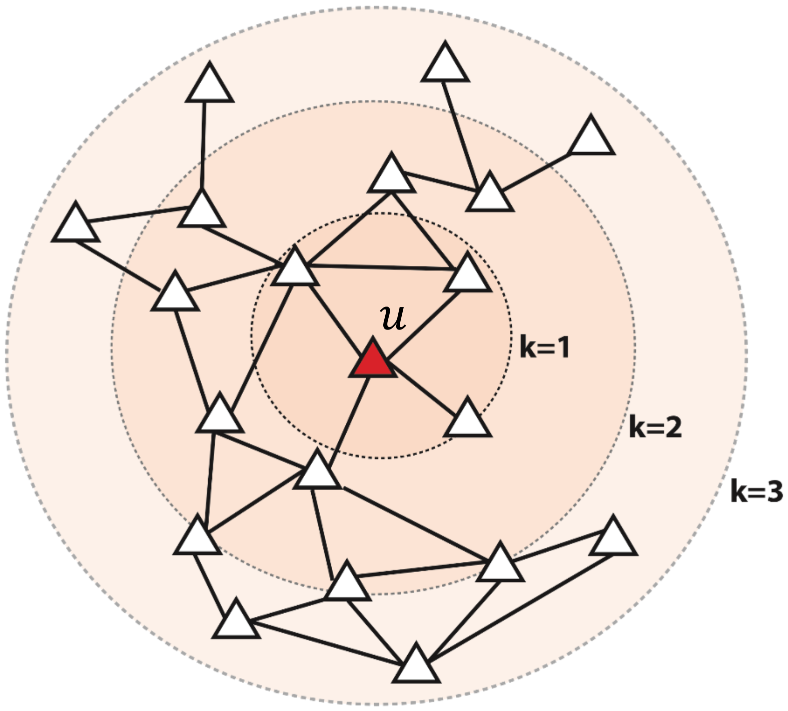
Unsupervised Feature Learning



- Given $G = (V, E)$
- Goal is to learn $f : u \to \mathbb{R}^d$
  - where f is a table lookup
    - We directly "learn" coordinates $\boldsymbol{z_u} = f(u)$ of u
- Given node u, we want to learn feature representation $f(u)$ that is predictive of nodes in u's neighborhood $N_\mathrm{R}(u)$

$$\max_f \sum_{u \in V} \log \Pr(N_\mathrm{R}(u) | \boldsymbol{z}_\mathrm{u})$$

Unsupervised Feature Learning



Goal: Find embedding $\boldsymbol{z_u}$ that predicts nearby nodes $N_R(\boldsymbol{u})$:

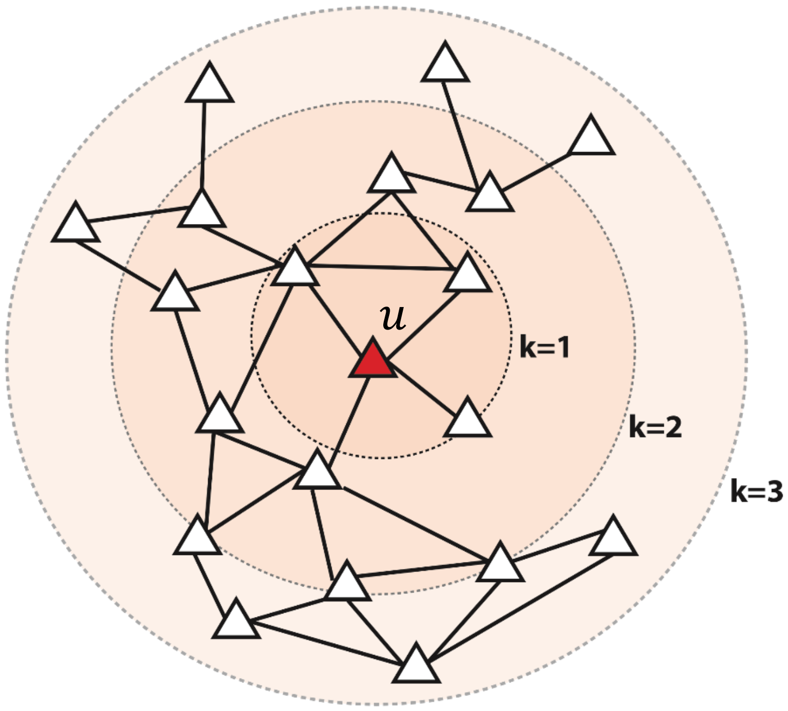
$$\sum_{v \in V} \log(P(N_R(u)|\mathbf{z}_u))$$

Assume conditional likelihood factorizes:

$$P(N_R(u)|\mathbf{z}_u) = \prod_{n_i \in N_R(u)} P(n_i|\mathbf{z}_u)$$

Random-walk Embeddings $\mathbf{z}_u^\top \mathbf{z}_v \approx$ Probability that u and v co-occur in a random walk over the network

Random Walk

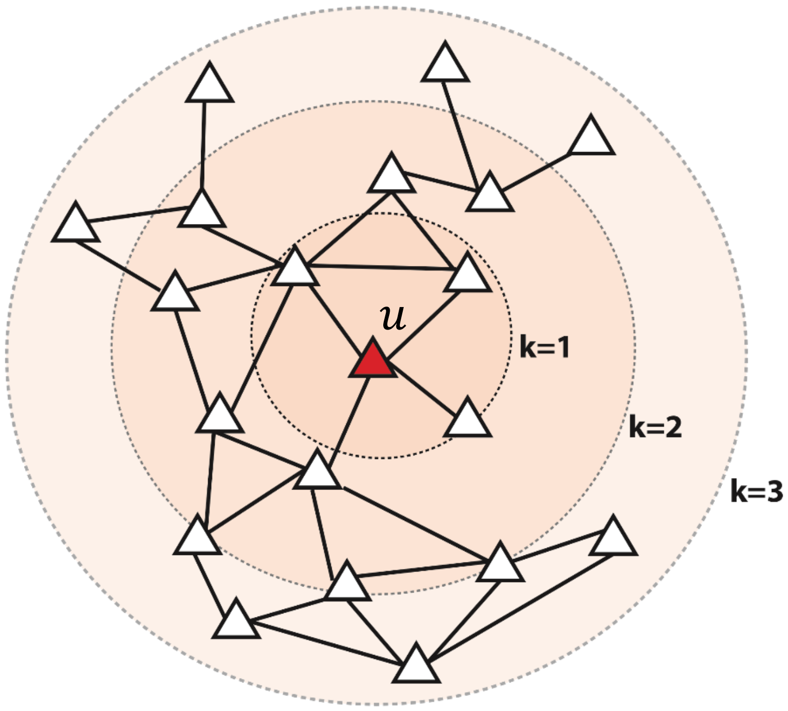


1. Simulate many short random walks starting from each node using a strategy R
2. For each node u, get $N_R(u)$ as a sequence of nodes visited by random walks starting at u
3. For each node u, learn its embedding by predicting which nodes are in $N_R(u)$:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

Random-walk Embeddings $\mathbf{z}_u^\top \mathbf{z}_v \approx$ Probability that u and v co-occur in a random walk over the network

Random Walk



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

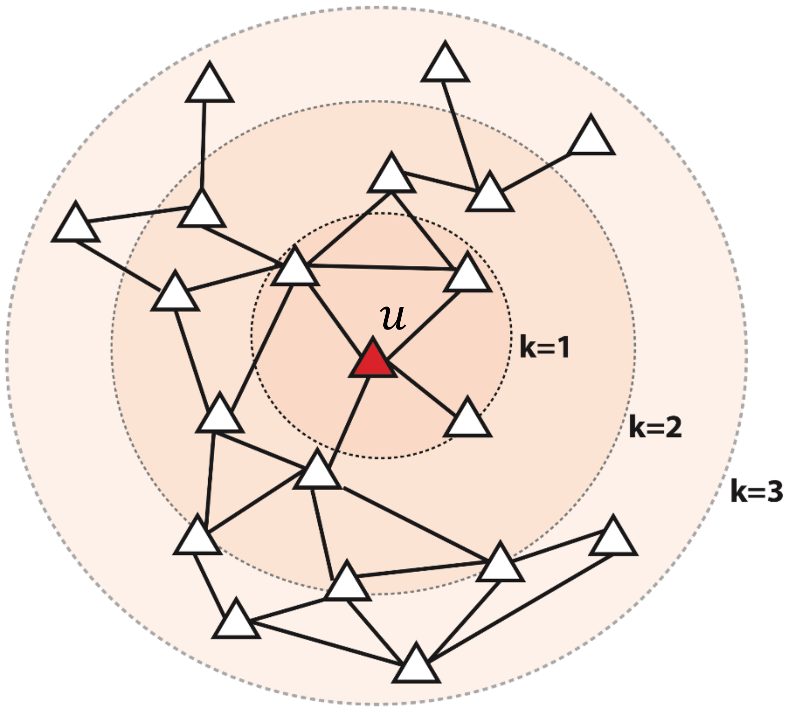
sum over all nodes u

sum over nodes v seen on random walks starting from u

predicted probability of u and v co-occuring on random walk, i.e., use softmax to parameterize $P(v|\mathbf{z}_u)$

Random walk embeddings = z_u minimizing L

Random Walk



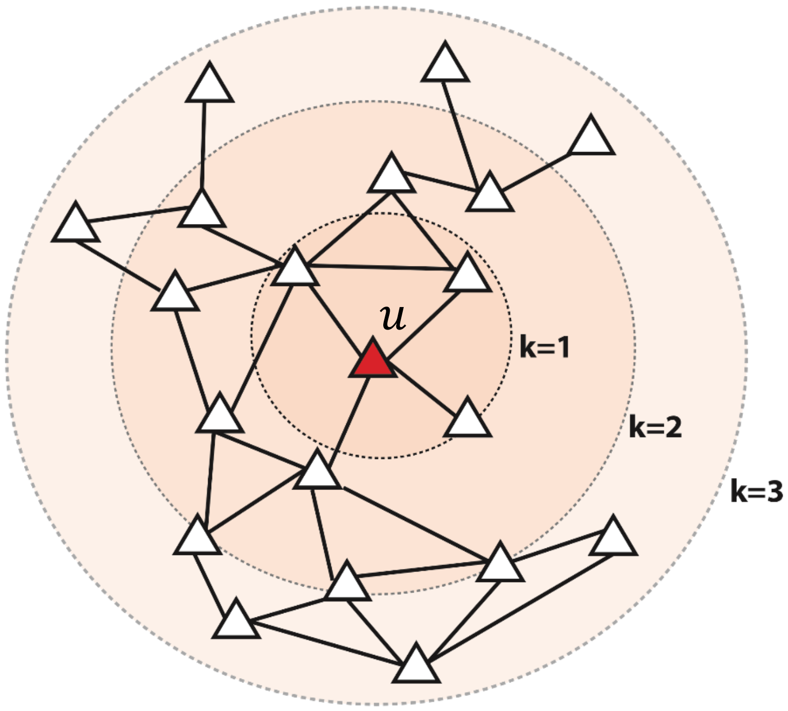
But doing this naively is too expensive!

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

Nested sum over nodes gives $O(|V|^2)$ complexity!

The problem is normalization term in the softmax function?

Random Walk



Solution: Negative sampling (Mikolov et al., 2013)

$$\log\left(\frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{n\in V}\exp(\mathbf{z}_u^\top \mathbf{z}_n)}\right)$$

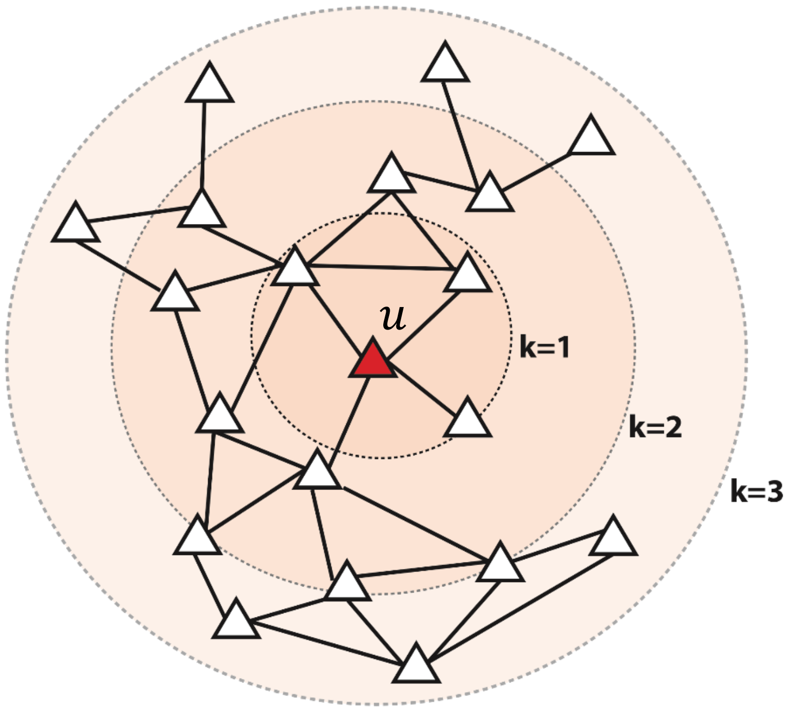
$$\approx \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - \sum_{i=1}^{k} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_{n_i})), n_i \sim P_V$$

sigmoid function

random distribution over all nodes

i.e., instead of normalizing w.r.t. all nodes, just normalize against k random **negative samples**

Random Walk

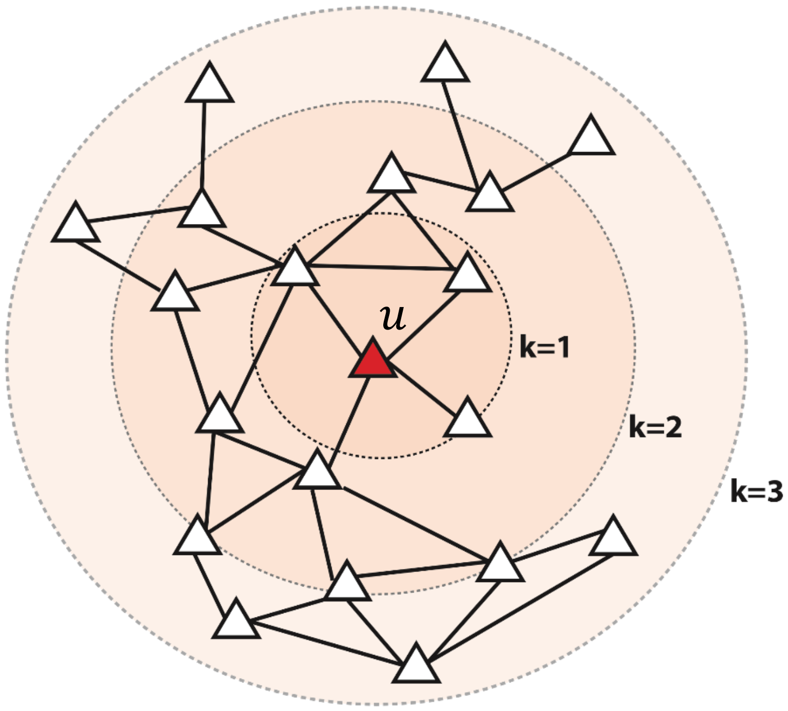


1. Simulate many short random walks starting from each node using a strategy R
2. For each node u, get $N_R(u)$ as a sequence of nodes visited by random walks starting at u
3. For each node u, learn its embedding by predicting which nodes are in $N_R(u)$:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

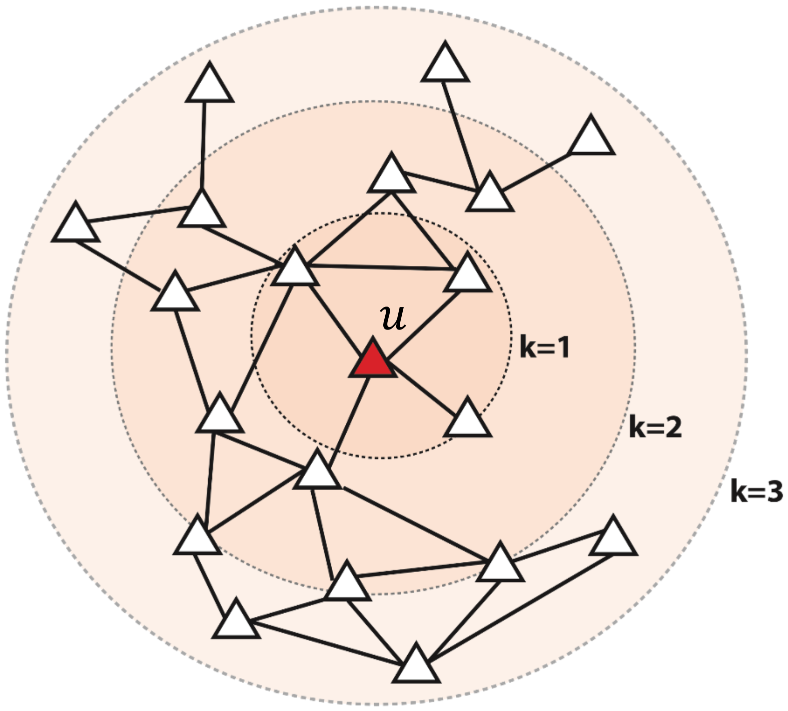
Can efficiently approximate using negative sampling

Random Walk （针对有权图如何修改）

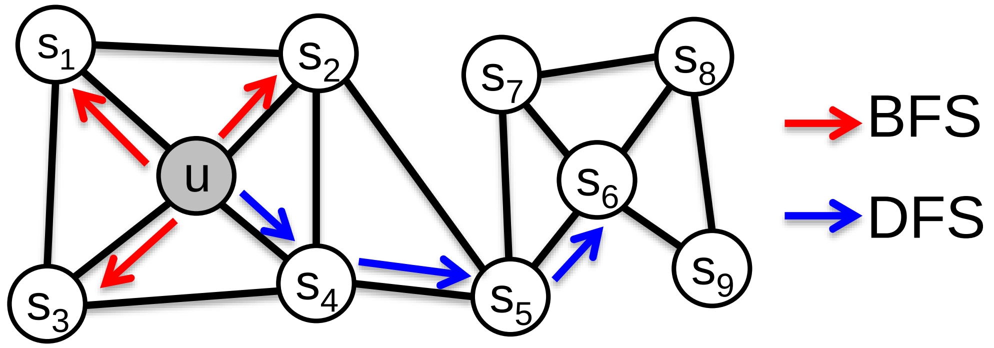


- **What strategies can we use to obtain these random walks?**
  - Simplest idea:
    - Fixed-length, unbiased random walks starting from each node (i.e., DeepWalk from Perozzi et al., 2013)
  - **Can we do better?**
    - Grover et al., 2016; Ribeiro et al., 2017; Abu-El-Haija et al., 2017 and many others

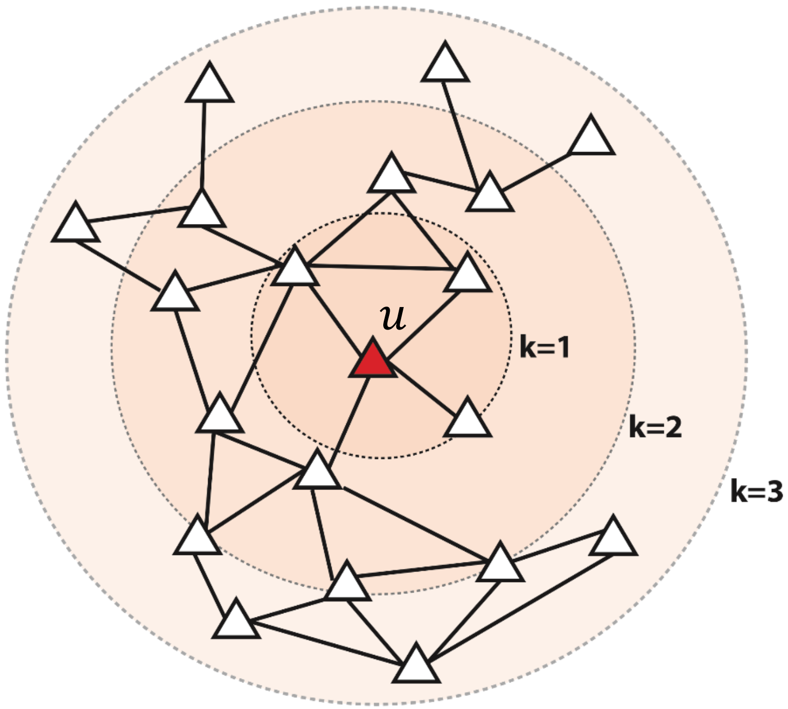
node2vec: Biased Walks



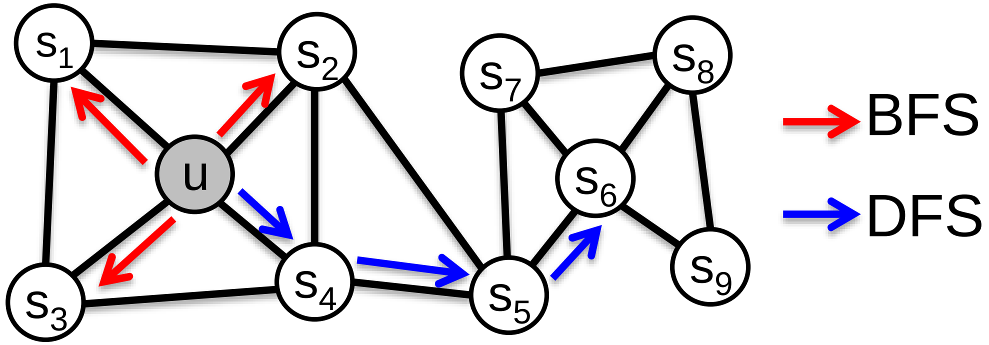
Idea: Use flexible, biased random walks that can trade off between **local** and **global** views of the network (Grover and Leskovec, 2016)



node2vec: Biased Walks



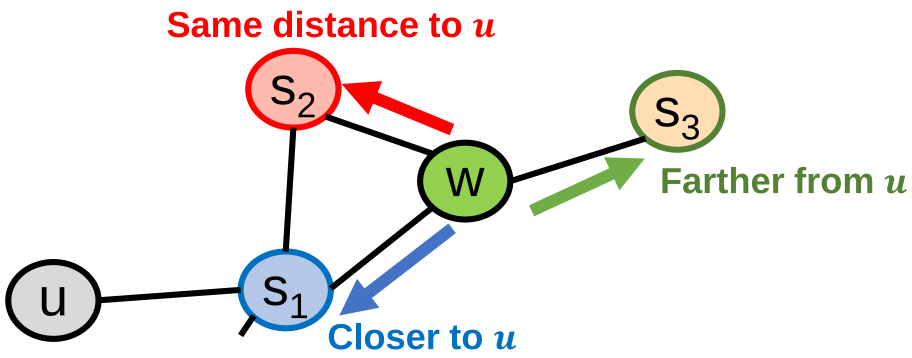
Two classic strategies to define a neighborhood $N_R(u)$ of a given node u:

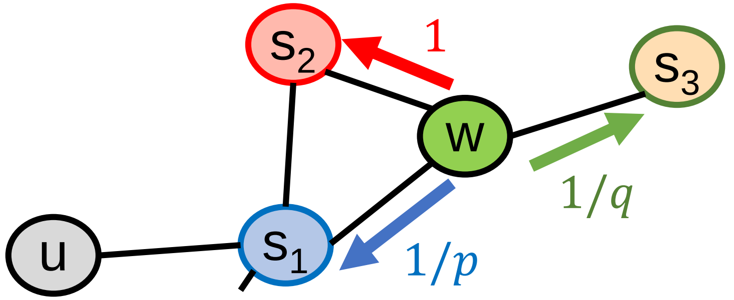


$N_{BFS}(u) = \{ s_1, s_2, s_3\}$ Local microscopic view

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

node2vec: Biased Walks





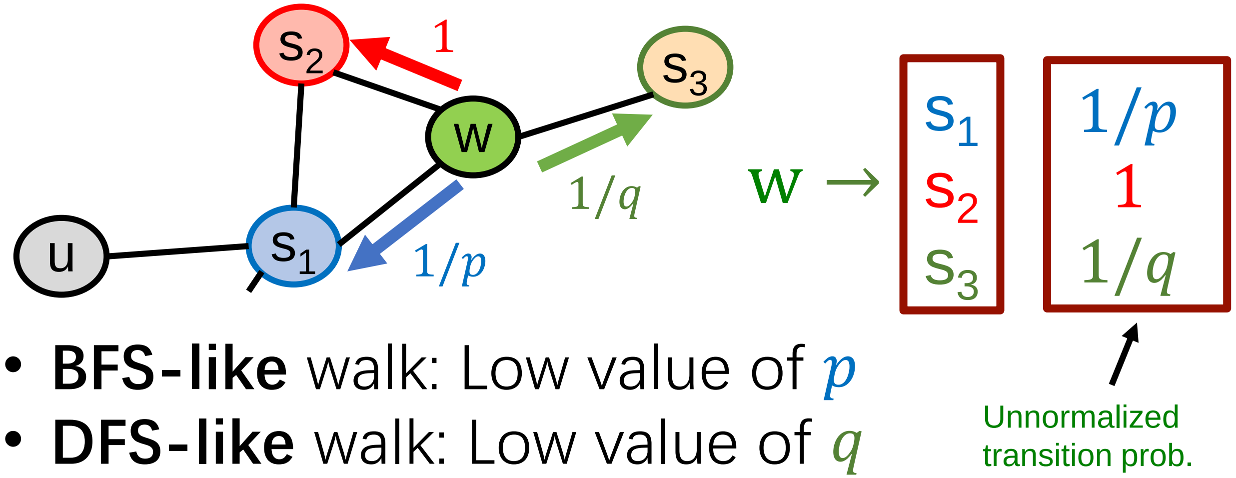
$1/p, 1/q, 1$ are unnormalized probabilities

Biased random walk R that given a node u generates neighborhood $N_R(u)$

- Two parameters:
  - Return parameter p:
    - return parameter
    - Return back to the previous node
  - In-out parameter q:
    - walk away parameter
    - Moving outwards (DFS) vs. inwards (BFS)

node2vec: Biased Walks

- Walker is at w. Where to go next?



- **BFS-like** walk: Low value of p
- **DFS-like** walk: Low value of q

$N_S(u)$ are the nodes visited by the walker

Karate Club

- 该图描述了一个空手道俱乐部会员的社交关系，以34名会员作为节点，如果两位会员在俱乐部之外仍保持社交关系，则在节点间增加一条边。
- 每个节点具有一个34维的特征向量，一共有78条边。
- 在收集数据的过程中，管理人员 John A 和 教练 Mr. Hi（化名）之间产生了冲突，会员们选择了站队，一半会员跟随 Mr. Hi 成立了新俱乐部，剩下一半会员找了新教练或退出了俱乐部。