Learning Beyond Node-level Embeddings

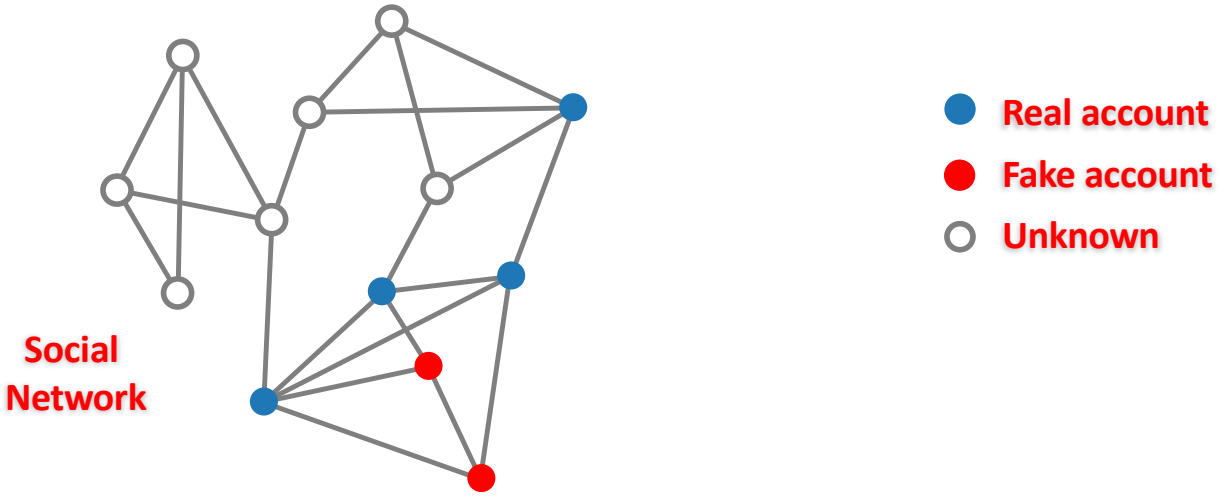
ACMS 80770: Deep Learning with Graphs

Instructor: Navid Shervani-Tabar

Department of Applied and Comp Math and Stats

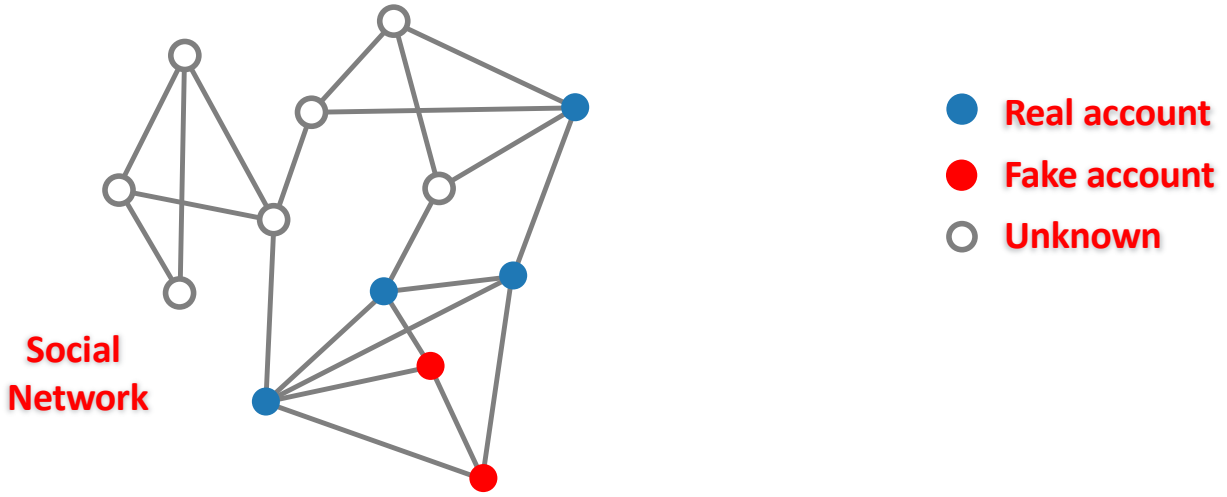
Node Embedding

- So far, we discussed learning **node embeddings** in a graph.
- These node embeddings can be used for node **classification** tasks.
  - Fake accounts problem



Node Embedding

- So far, we discussed learning **node embeddings** in a graph.
- These node embeddings can be used for node **classification** tasks.
  - Fake accounts problem



- We can use these node embeddings to construct **edge-level**, **subgraph-level**, or **graph-level** tasks.

Edge Embedding

- One can construct a feature vectors $\boldsymbol{z}_{(i,j)}$ corresponding to edges (v_i, v_j) by **integrating** pairs of node embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, to perform edge-level tasks.
- There area few approaches to integrate pair of features $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ to construct feature vectors $\boldsymbol{z}_{(i,j)}: V \times V \rightarrow \mathbb{R}^d$.

Edge Embedding

- One can construct a feature vectors $\boldsymbol{z}_{(i,j)}$ corresponding to edges (v_i, v_j) by **integrating** pairs of node embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, to perform edge-level tasks.
- There area few approaches to integrate pair of features $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ to construct feature vectors $\boldsymbol{z}_{(i,j)}: V \times V \rightarrow \mathbb{R}^d$.
  - Hadamard product

$$\mathbf{z}_{(i,j)} = \mathbf{z}_i \odot \mathbf{z}_j$$

  - Average

$$\mathbf{z}_{(i,j)} = \frac{1}{2}\left(\mathbf{z}_i + \mathbf{z}_j\right)$$

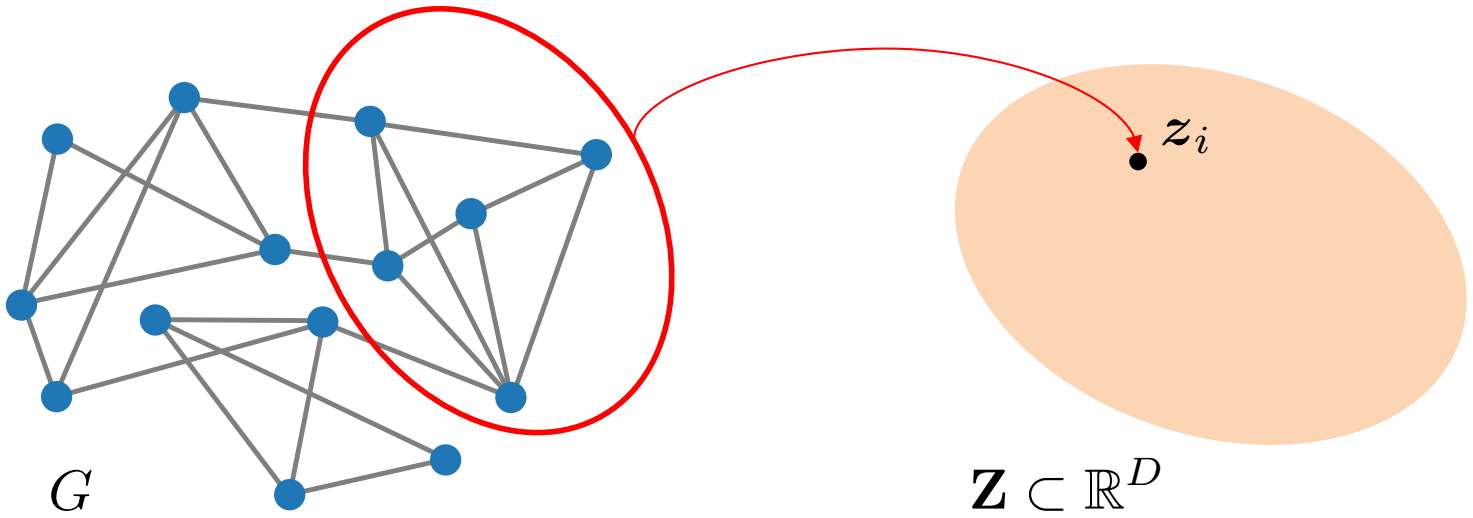
  -

$$\boldsymbol{z}_{(i,j)} = \left(\mathbf{z}_i - \mathbf{z}_j\right) \odot \left(\mathbf{z}_i - \mathbf{z}_j\right)$$

  -

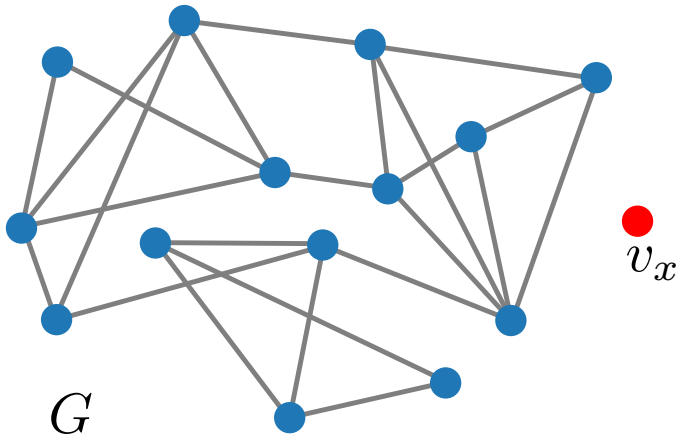
$$\boldsymbol{z}_{(i,j)} = \left|\mathbf{z}_i - \boldsymbol{z}_j\right|$$

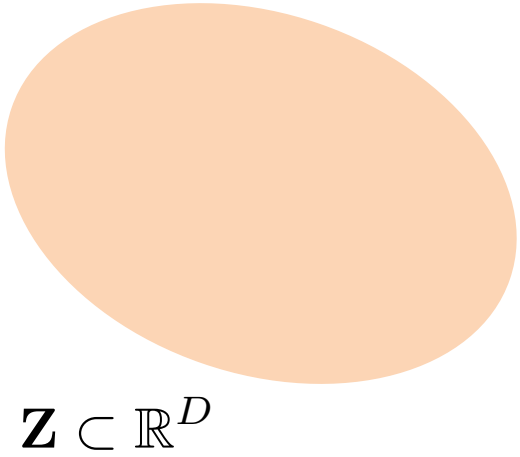
Subgraph Embedding



Subgraph Embedding

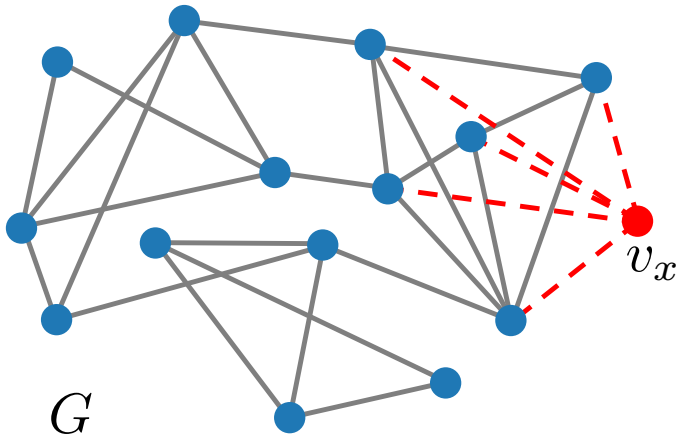
- We can define a **dummy node** for constructing subgraph-level embeddings.

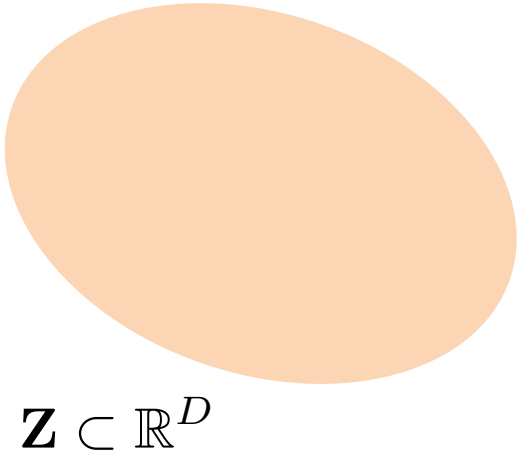




Subgraph Embedding

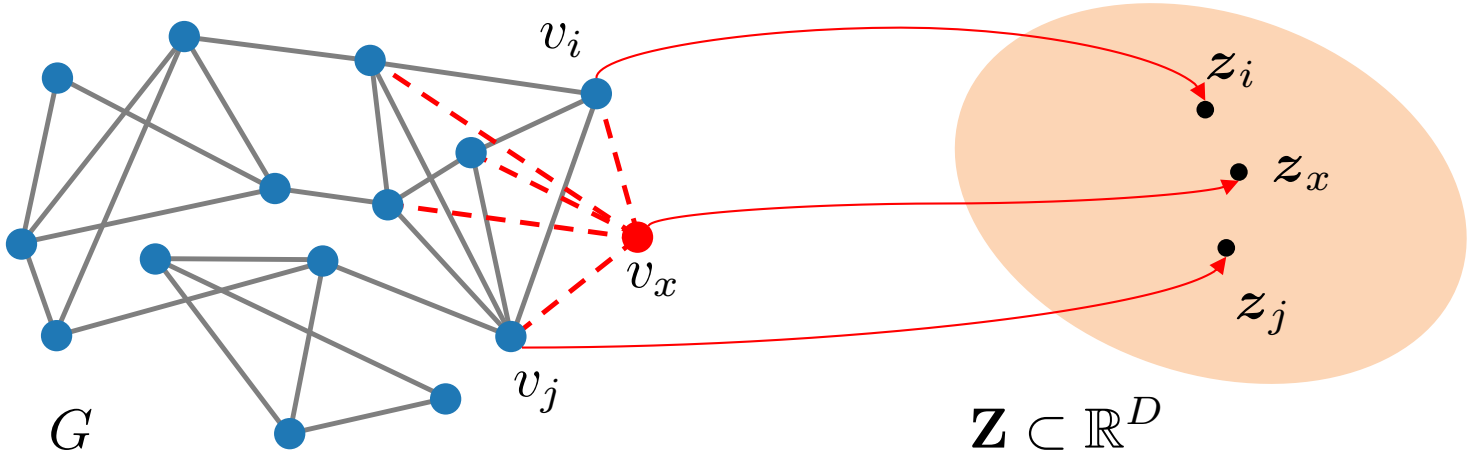
- We can define a **dummy node** for constructing subgraph-level embeddings.





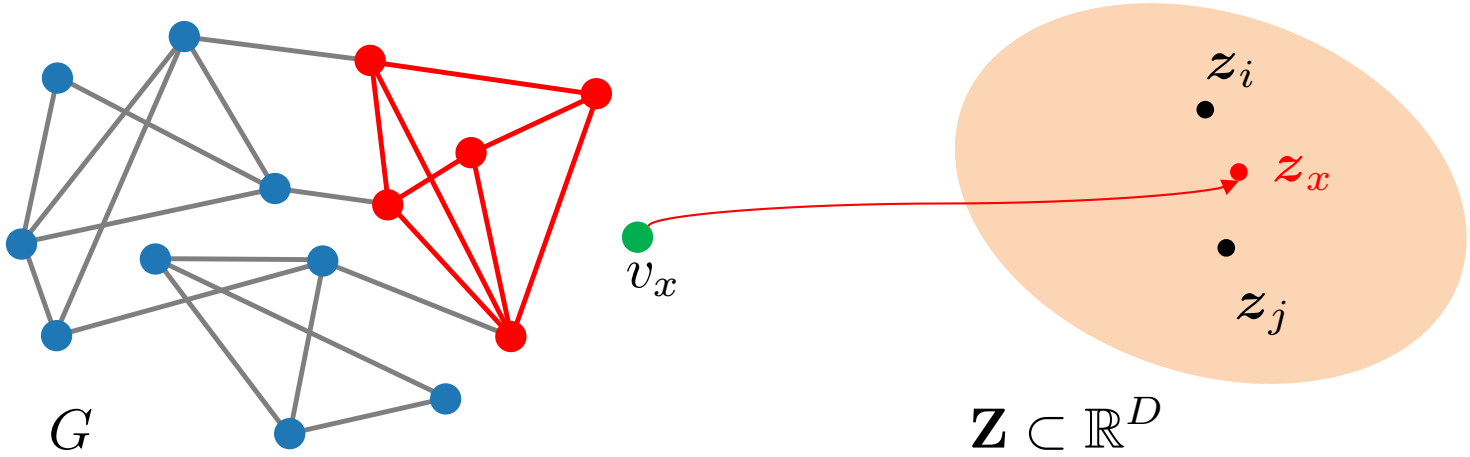
Subgraph Embedding

- We can define a **dummy node** for constructing subgraph-level embeddings.



Subgraph Embedding

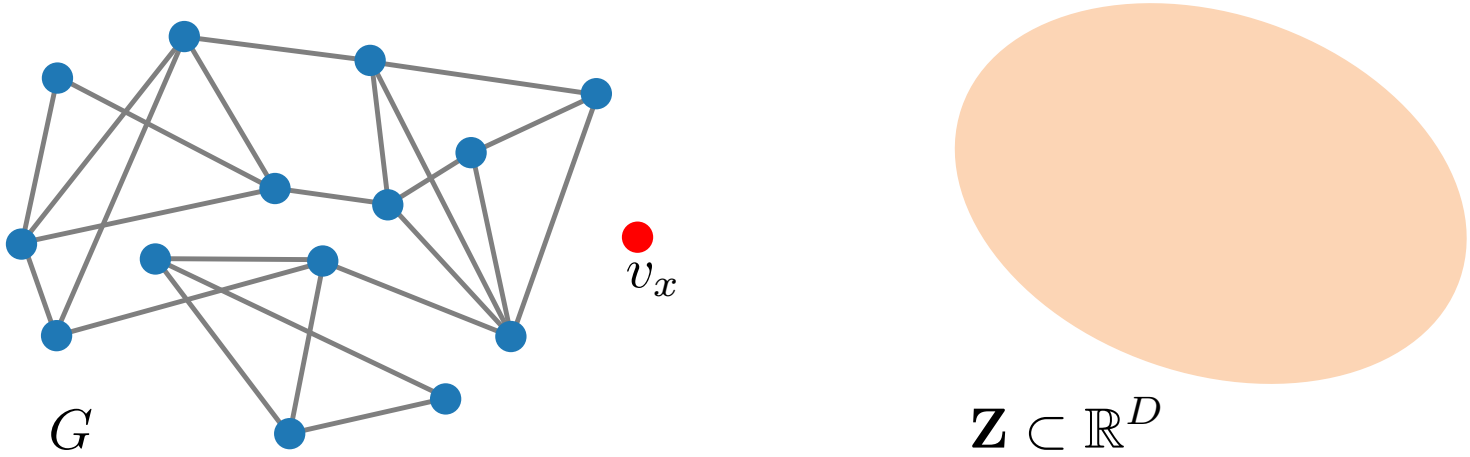
- We can define a dummy node for constructing subgraph-level embeddings.



- Learn node embedding for the nodes $v_i \in V$ and use the learned embedding for the dummy node to represent the **subgraph**.

Graph Embedding

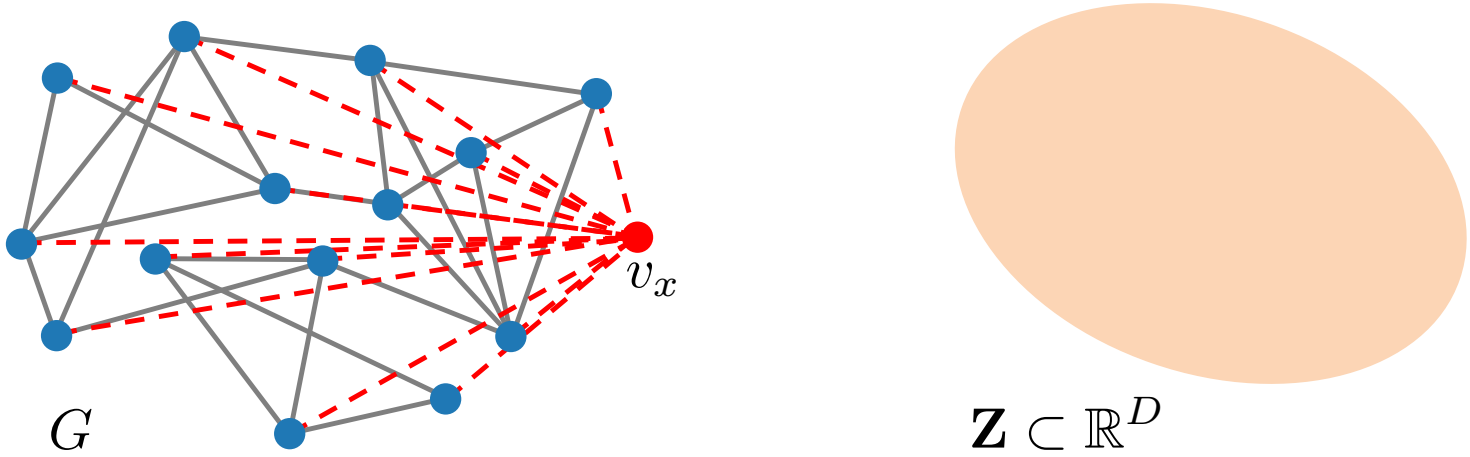
- We can define a **dummy node** for constructing subgraph-level embeddings.



- Learn node embedding for the nodes $v_i \in V$ and use the learned embedding for the dummy node to represent the **subgraph**.
- This idea can be extended to the whole **graph**.

Graph Embedding

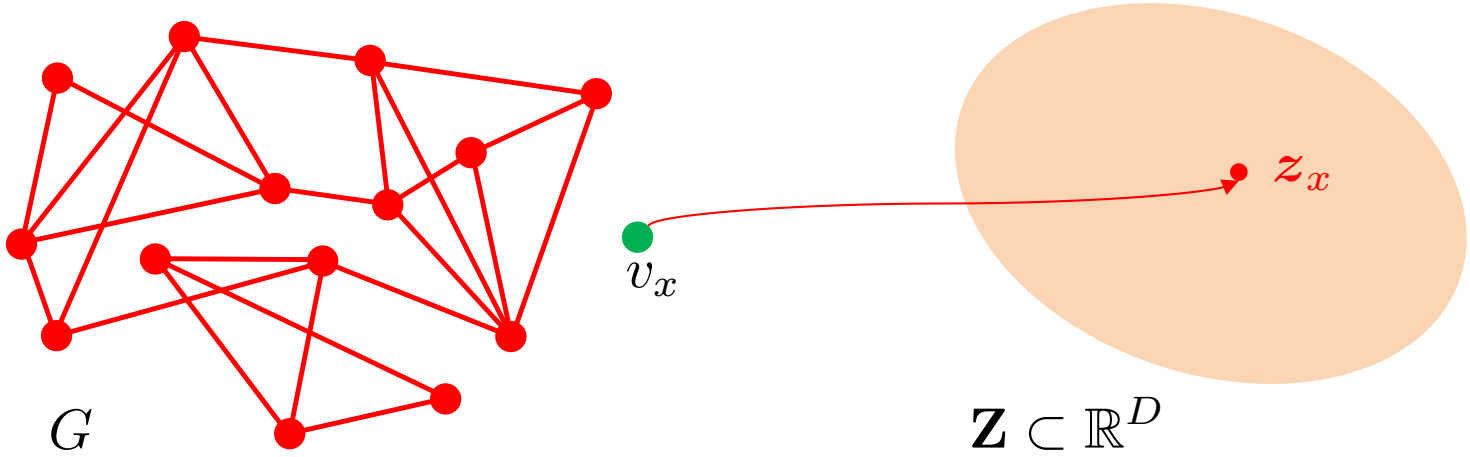
- We can define a dummy node for constructing subgraph-level embeddings.



- Learn node embedding for the nodes $v_i \in V$ and use the learned embedding for the dummy node to represent the **subgraph**.
- This idea can be extended to the whole **graph**.

Graph Embedding

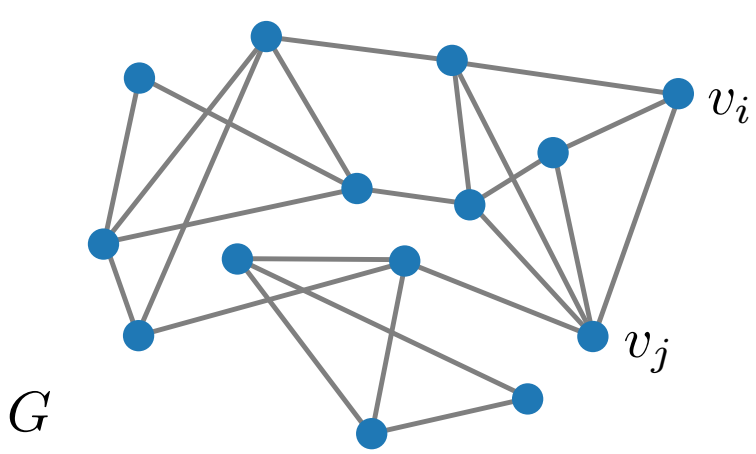
- We can define a dummy node for constructing subgraph-level embeddings.

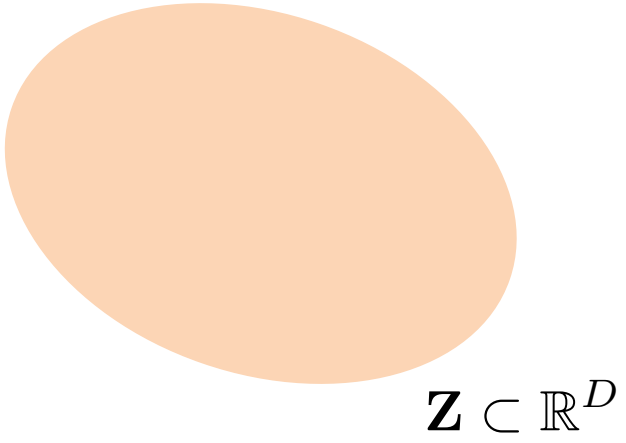


- Learn node embedding for the nodes $v_i \in V$ and use the learned embedding for the dummy node to represent the **subgraph**.
- This idea can be extended to the whole **graph**.

Graph Embedding

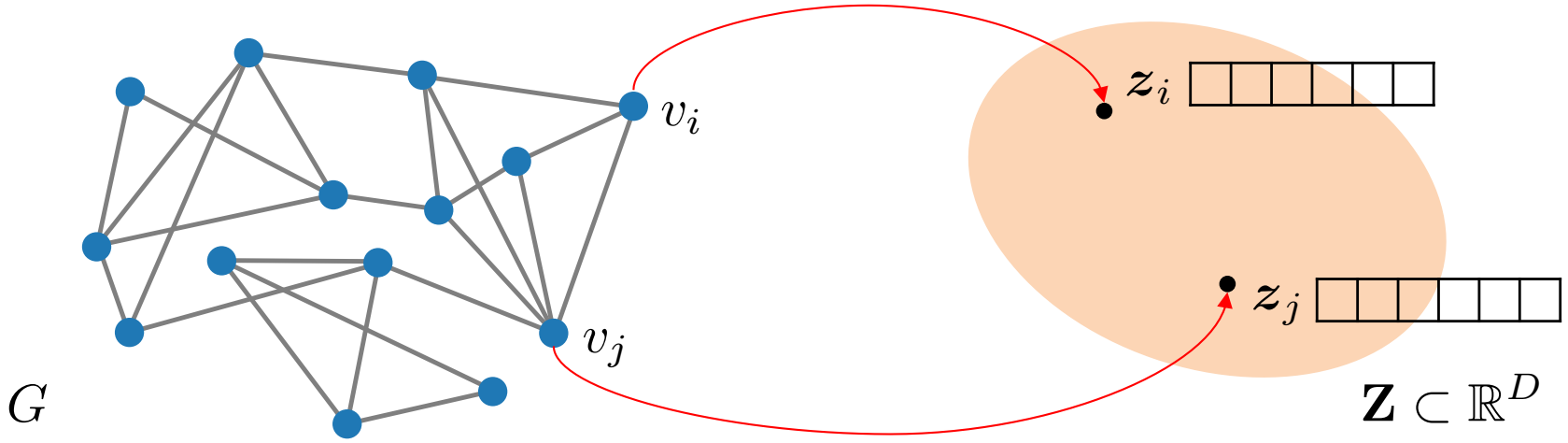
- Alternatively, we can use node embeddings to construct a **graph embedding**.





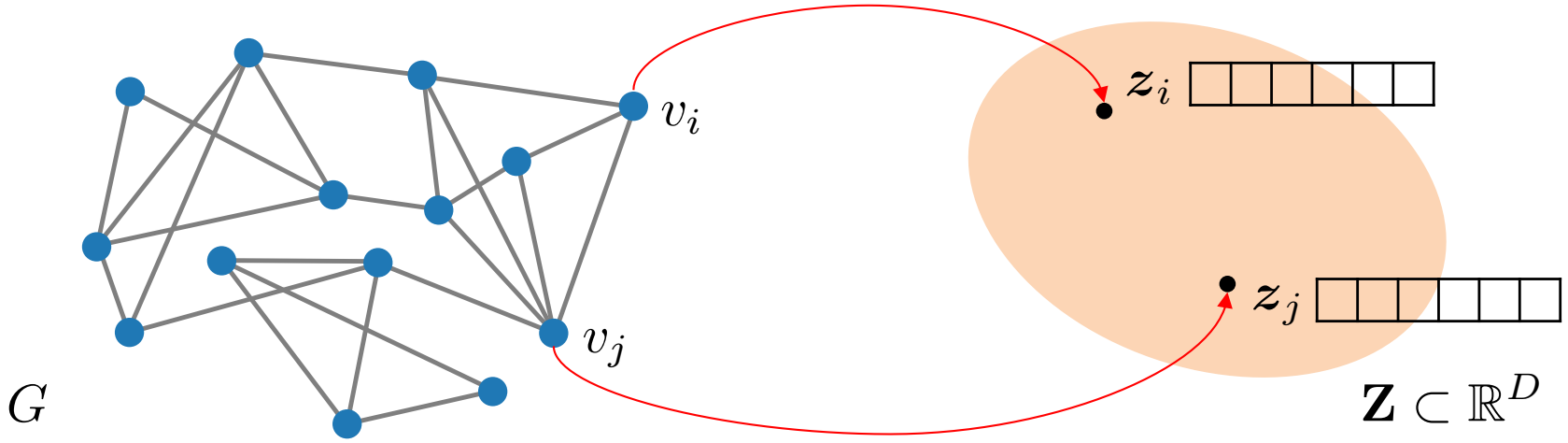
Graph Embedding

- Alternatively, we can use node embeddings to construct a **graph embedding**.



Graph Embedding

- Alternatively, we can use node embeddings to construct a **graph embedding**.



$$z_G = \sum_{v_i \in V} z_i$$

- This graph embedding can be used to perform **graph-level** machine learning tasks.

Anonymous Walk

- We can construct graph representation using **anonymous walks**.
- Consider a random walk

$$w = \{\{v^{(1)}, v^{(2)}, \ldots, v^{(k)}\}\}$$

Anonymous Walk

- We can construct graph representation using **anonymous walks**.
- Consider a random walk

$$w = \{\{v^{(1)}, v^{(2)}, \ldots, v^{(k)}\}\}$$

- The anonymous walk corresponding to w is

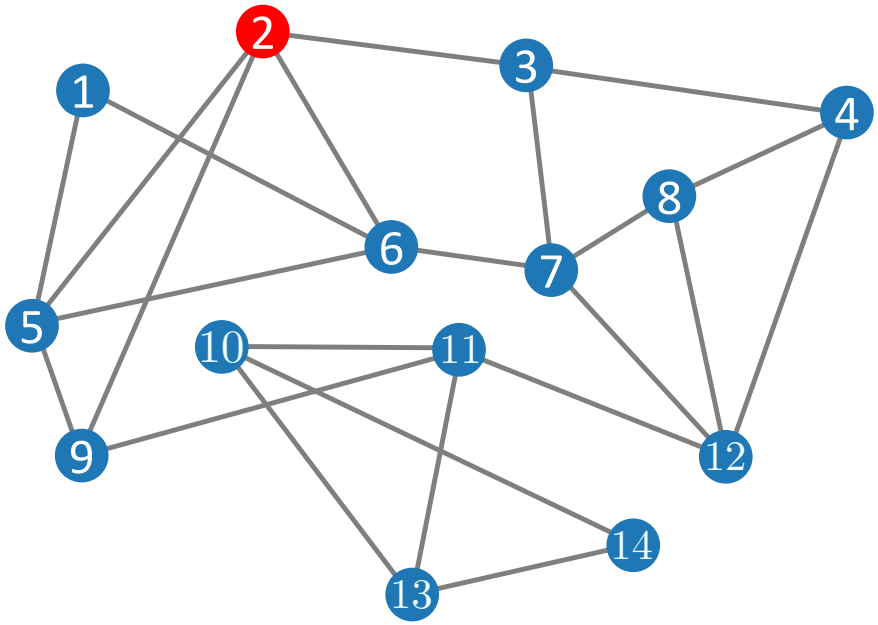
$$a = \{\{f(v^{(1)}), f(v^{(2)}), \ldots, f(v^{(k)})\}\}$$

where

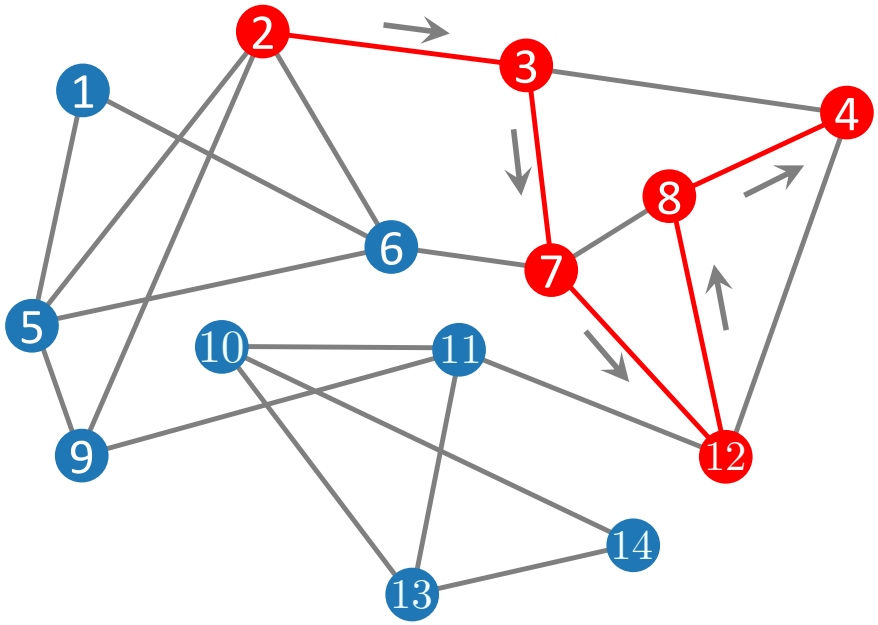
$$f\left(v^{(i)}\right) = \min pos\left(w, v^{(i)}\right)$$

with $pos(w, v^{(i)})$ the set of all positions of $v^{(i)}$ occurring in w.

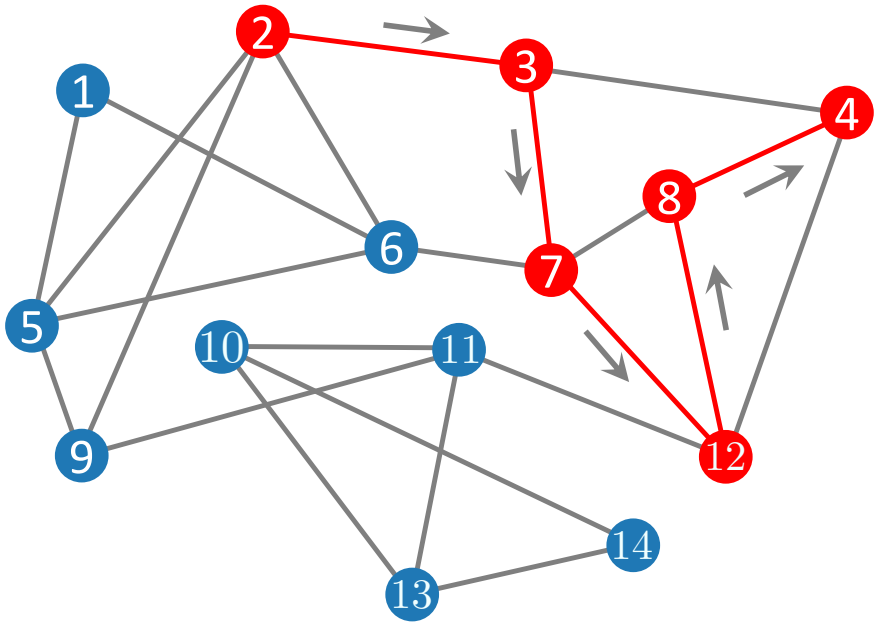
Anonymous Walk



Anonymous Walk

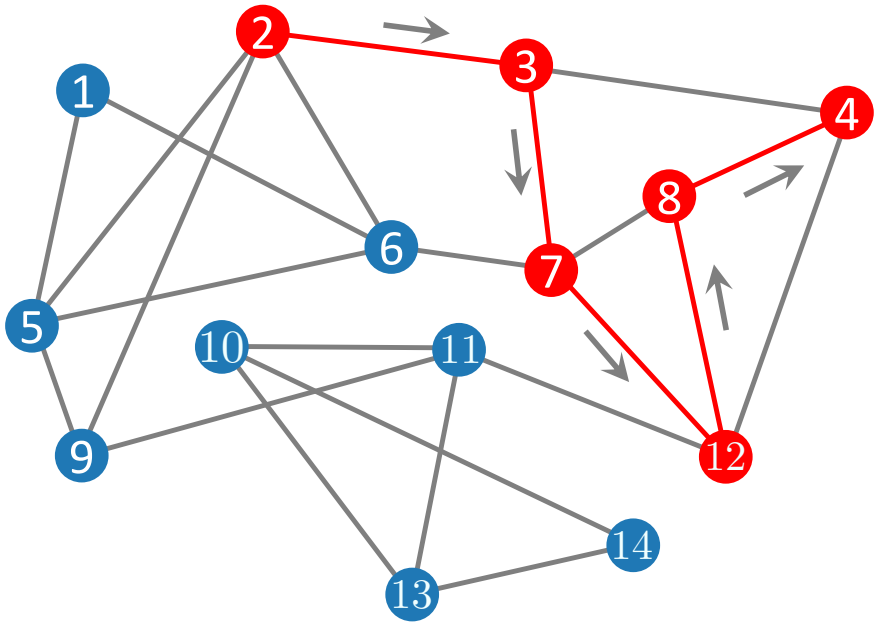


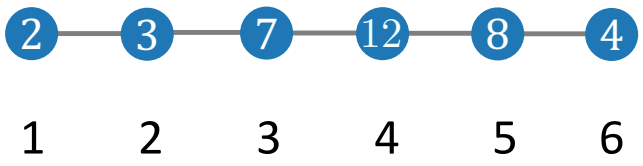
Anonymous Walk



$$w = \{\{v_2, v_3, v_7, v_{12}, v_8, v_4\}\}$$

Anonymous Walk

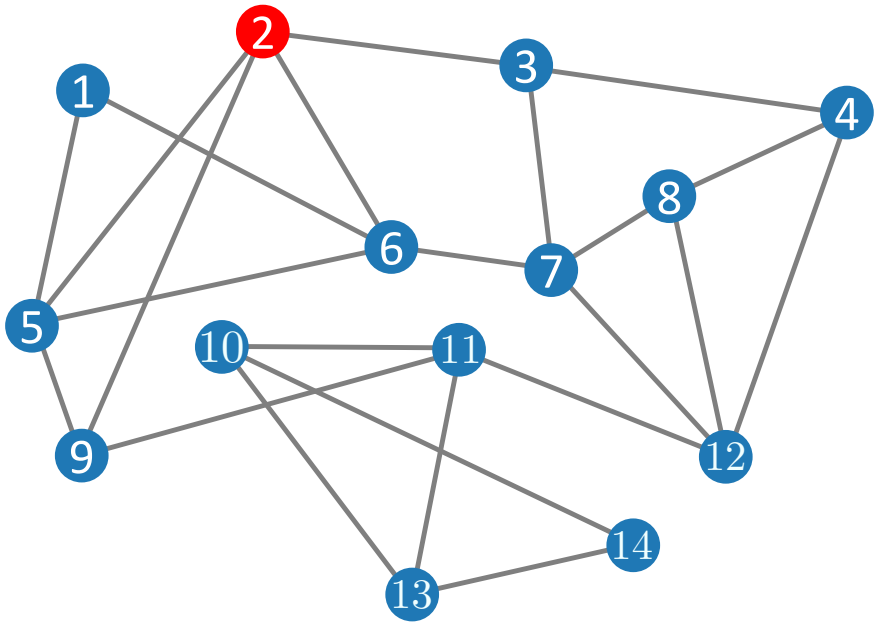




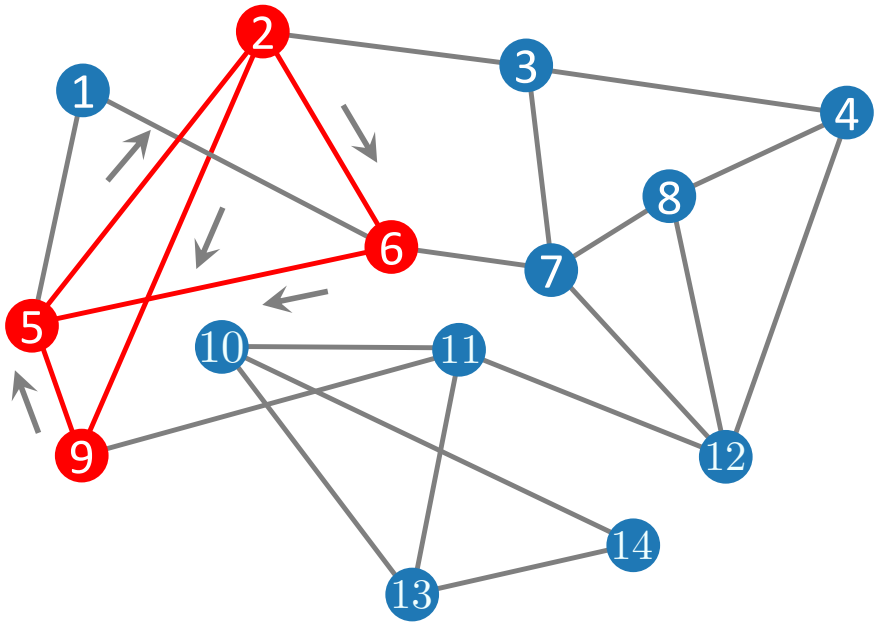
$$w = \{\{v_2, v_3, v_7, v_{12}, v_8, v_4\}\}$$

$$a = \{\{v^{(1)}, v^{(2)}, v^{(3)}, v^{(4)}, v^{(5)}, v^{(6)}\}\}$$

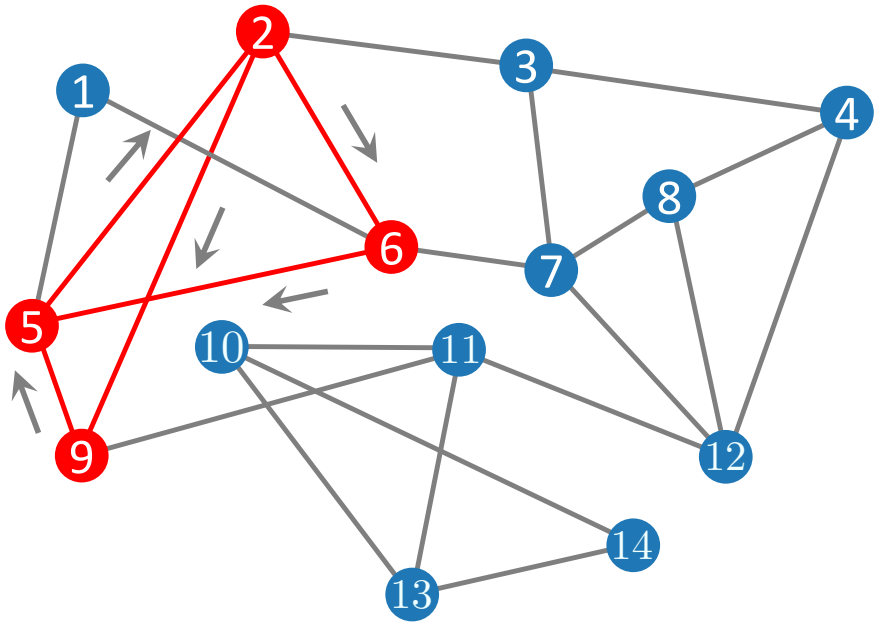
Anonymous Walk



Anonymous Walk

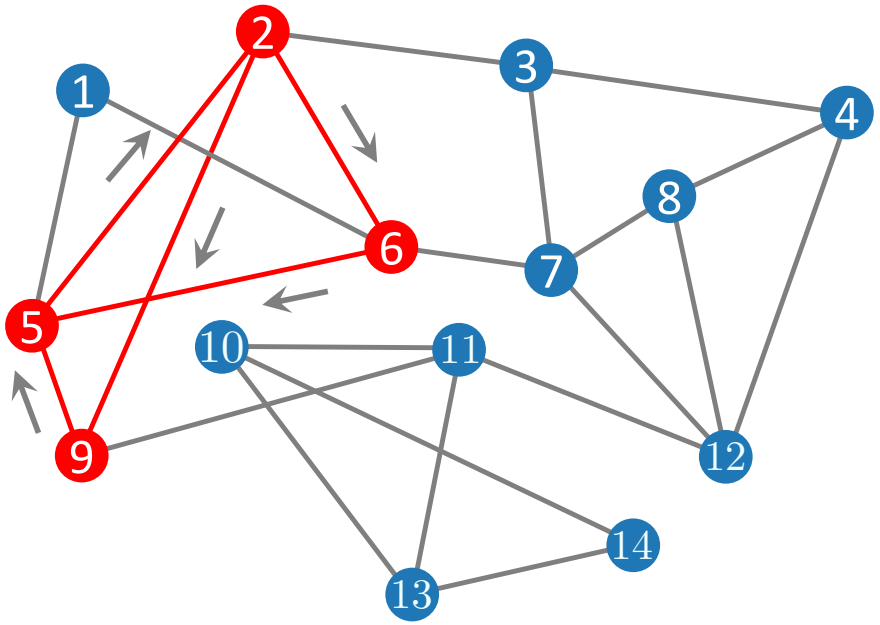


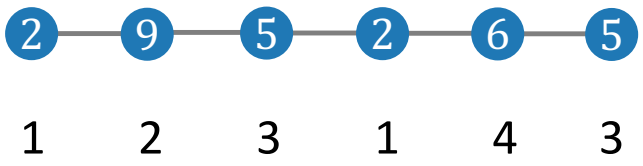
Anonymous Walk



$$w = \{\{v_2, v_9, v_5, v_2, v_6, v_5\}\}$$

Anonymous Walk

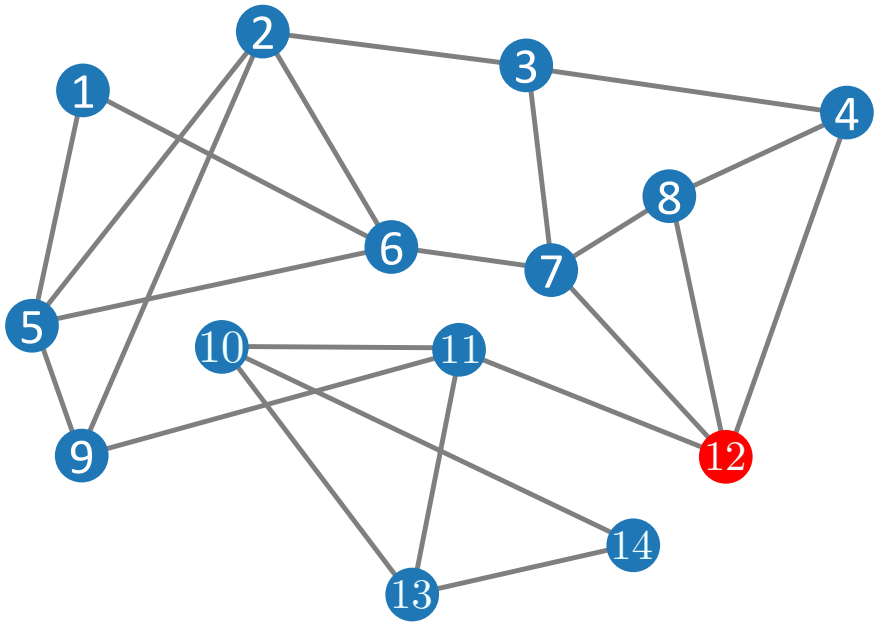




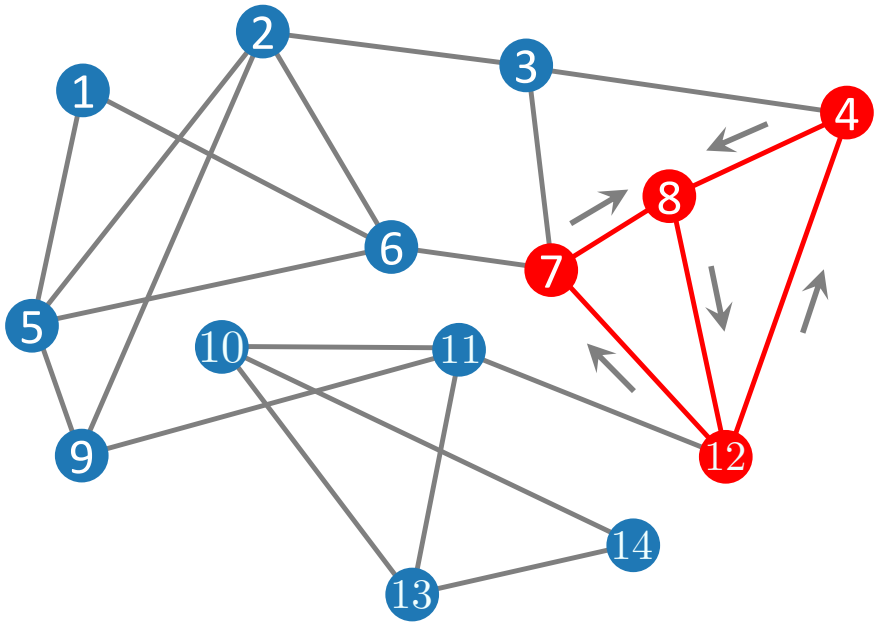
$$w = \{\{v_2, v_9, v_5, v_2, v_6, v_5\}\}$$

$$a = \{\{v^{(1)}, v^{(2)}, v^{(3)}, v^{(1)}, v^{(4)}, v^{(3)}\}\}$$

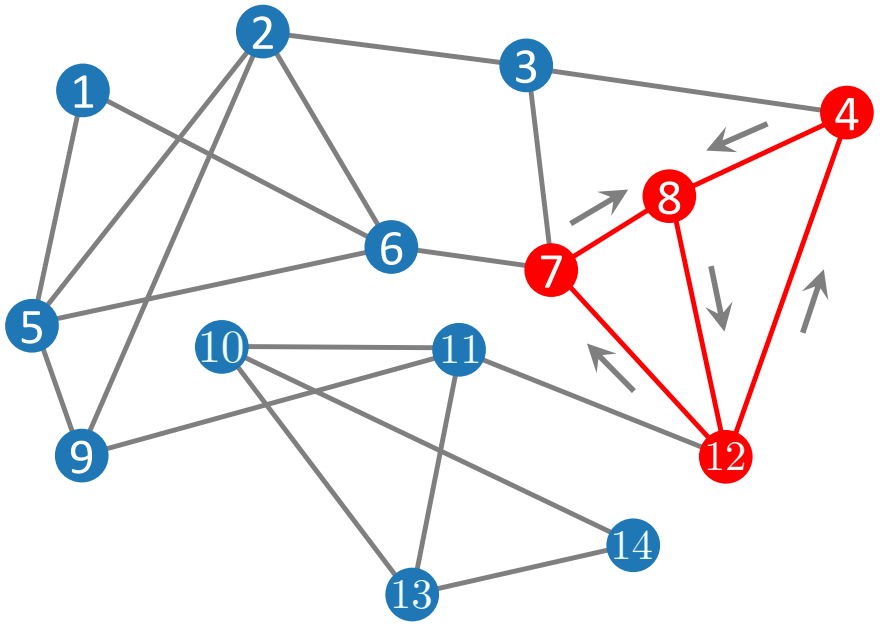
Anonymous Walk



Anonymous Walk

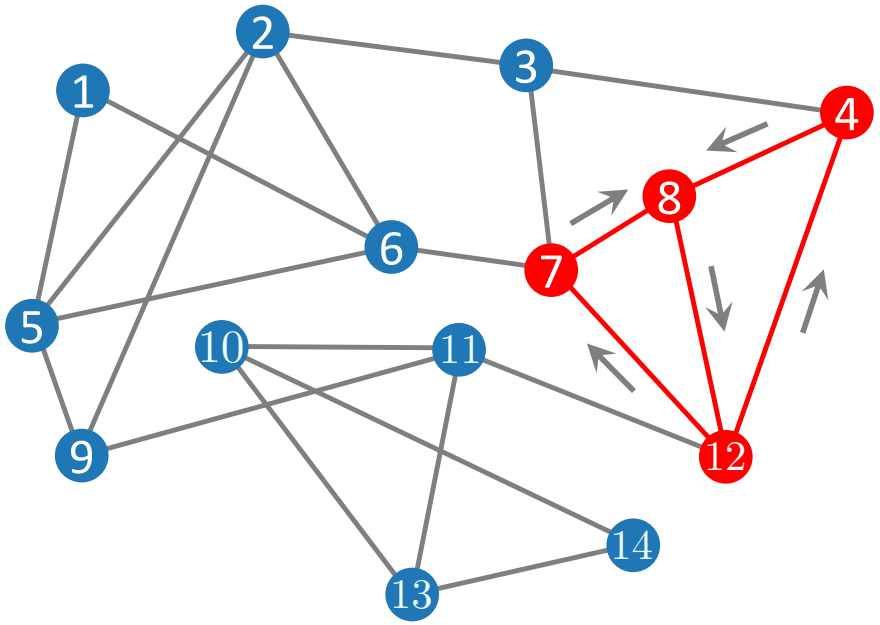


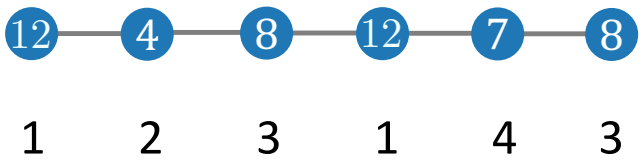
Anonymous Walk



$$w = \{\{v_{12}, v_4, v_8, v_{12}, v_7, v_8\}\}$$

Anonymous Walk

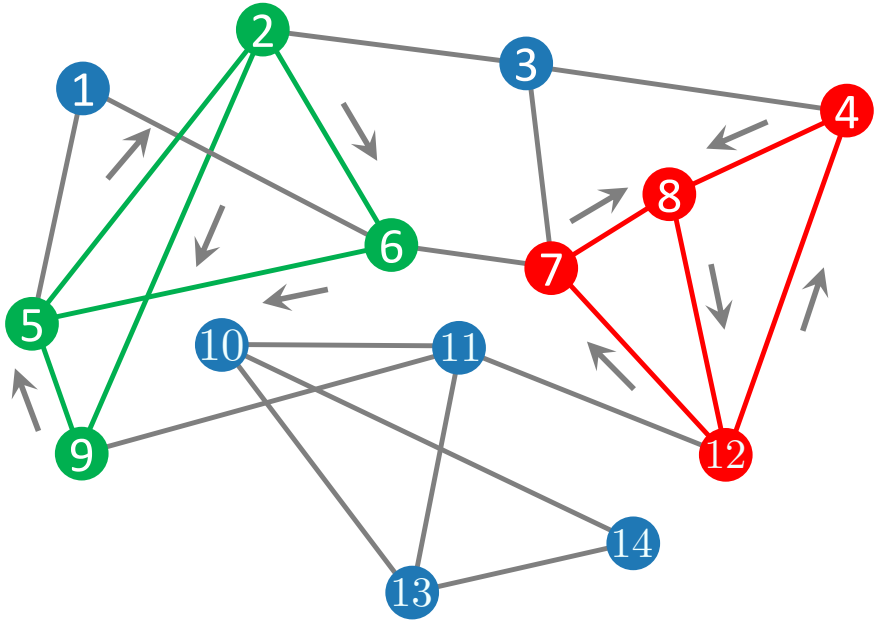




$$w = \{\{v_{12}, v_4, v_8, v_{12}, v_7, v_8\}\}$$

$$a = \{\{v^{(1)}, v^{(2)}, v^{(3)}, v^{(1)}, v^{(4)}, v^{(3)}\}\}$$

Anonymous Walk

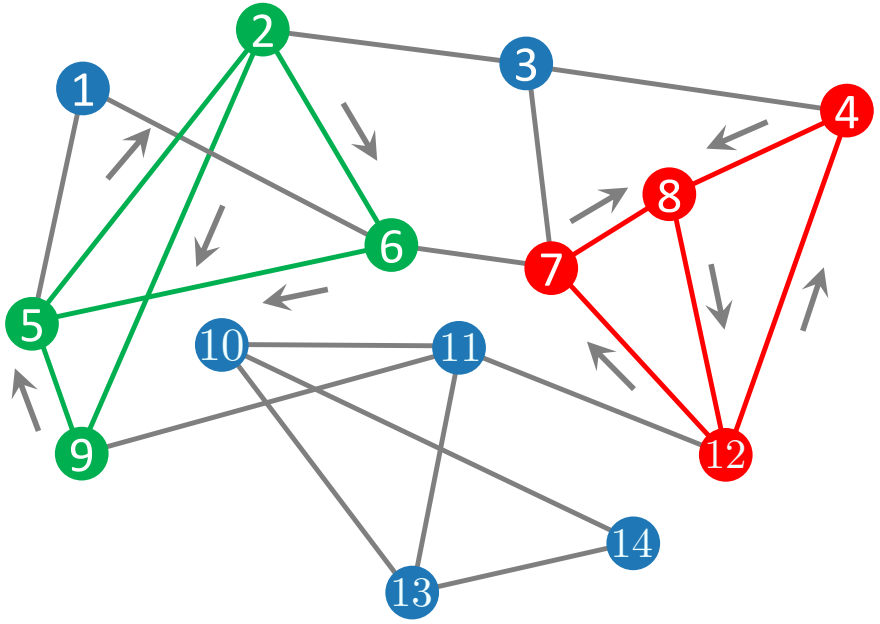


$$w = \{\{v_2, v_9, v_5, v_2, v_6, v_5\}\}$$

$$w = \{\{v_{12}, v_4, v_8, v_{12}, v_7, v_8\}\}$$

Anonymous Walk

- Different random walks may correspond to the **same** anonymous walk.



$$w = \{\{v_2, v_9, v_5, v_2, v_6, v_5\}\}$$

$$a = \{\{v^{(1)}, v^{(2)}, v^{(3)}, v^{(1)}, v^{(4)}, v^{(3)}\}\}$$

$$w = \{\{v_{12}, v_4, v_8, v_{12}, v_7, v_8\}\}$$

$$a = \{\{v^{(1)}, v^{(2)}, v^{(3)}, v^{(1)}, v^{(4)}, v^{(3)}\}\}$$

Graph Embedding

- We use the notion of anonymous walk to learn **graph embeddings**.
- There are two approaches to construct graph representations with anonymous walks
  - **Feature-based** graph embedding
  - **Data-driven** graph embedding

Anonymous Walk Embedding

- We use the notion of anonymous walk to learn **graph embeddings**.
- There are two approaches to construct graph representations with anonymous walks
  - **Feature-based** graph embedding
  - **Data-driven** graph embedding
- Consider the random walk

$$w = \{\{v_i, v_j, v_k, \dots\}\}$$

- The probability of observing random walk w on graph G can be defined using the notion of the **transition probability** matrix.

Anonymous Walk Embedding

- Let $\boldsymbol{P}$ denote **transition probability** matrix on graph G. In index notation

$$P_{ij} = \frac{A_{ij}}{d_j}$$

- Let the **probability** of observing a **random walk** w on graph G is the product of the transition probabilities

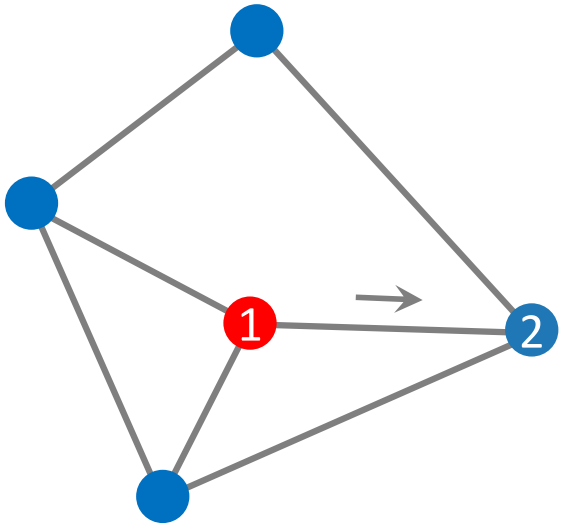
$$p(w) = \prod_{(v_i, v_j) \in w} P_{ij}$$

corresponding to the edges traversed by the random walker.

$$w = \{\{v_i, v_j, v_k, \dots\}\}$$

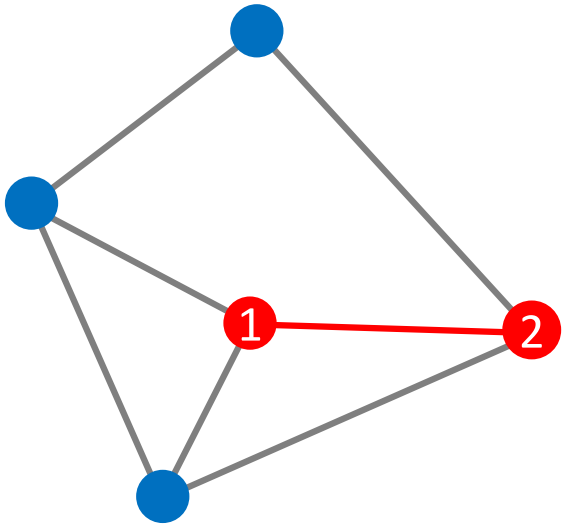
Anonymous Walk Embedding

- $l = 2$



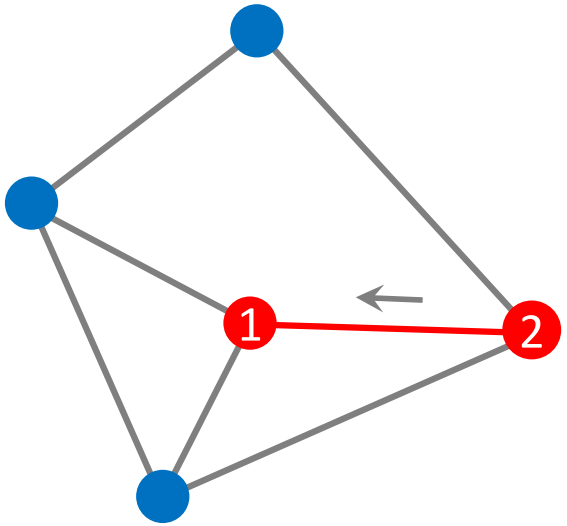
Anonymous Walk Embedding

- $l = 2$



Anonymous Walk Embedding

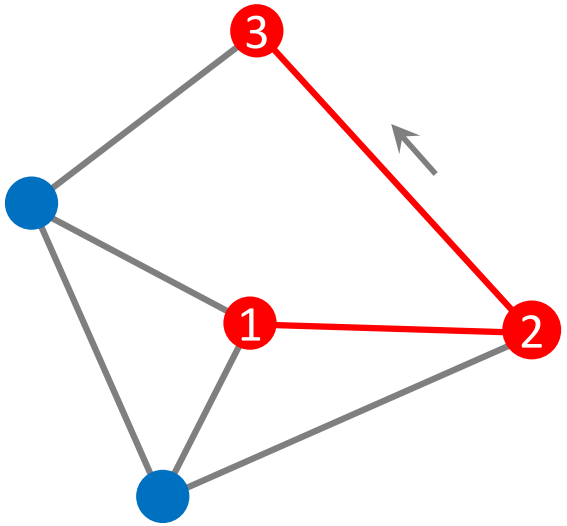
- $l = 2$



$$a_1 = \{\{1, 2, 1\}\}$$

Anonymous Walk Embedding

- $l = 2$

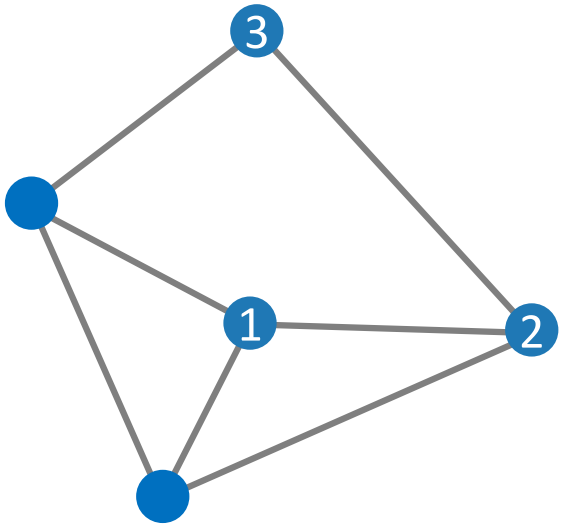


$$a_1 = \{\{1, 2, 1\}\}$$

$$a_2 = \{\{1, 2, 3\}\}$$

Anonymous Walk Embedding

- There are only a certain number of possible anonymous walks of given length l.
  - $l = 2$



$$a_1 = \{\{1, 2, 1\}\}$$

$$a_2 = \{\{1, 2, 3\}\}$$

$$\mathcal{A}_l = \{a_1, a_2\}$$

Anonymous Walk Embedding

- There are only a certain number of possible anonymous walks of given length l.
- Different random walks may correspond to the same anonymous walk.

$$w = \{\{v_2, v_9, v_5, v_2, v_6, v_5\}\} \qquad a = \{\{1, 2, 3, 1, 4, 3\}\}$$

$$w = \{\{v_{12}, v_4, v_8, v_{12}, v_7, v_8\}\} \qquad a = \{\{1, 2, 3, 1, 4, 3\}\}$$

Anonymous Walk Embedding

- There are only a certain number of possible anonymous walks of given length l.
- Different random walks may correspond to the same anonymous walk.
- The probability of seeing an anonymous walk a_i in graph G is

$$p(a_k) = \frac{1}{|V|} \sum_{v_i \in V} \sum_{w \to a_k} p(w)$$

Anonymous Walk Embedding

- There are only a certain number of possible anonymous walks of given length l.
- Different random walks may correspond to the same anonymous walk.
- The probability of seeing an anonymous walk a_i in graph G is

$$p\left(a_k\right) = \frac{1}{|V|} \sum_{v_i \in V} \sum_{w \to a_k} p\left(w\right)$$

- We can use this and define anonymous walk embedding as the vector of probabilities of different walks on the graph

$$\phi(G) = \left(p\left(a_1\right), p\left(a_2\right), \ldots, p\left(a_K\right)\right)$$

Anonymous Walk Embedding

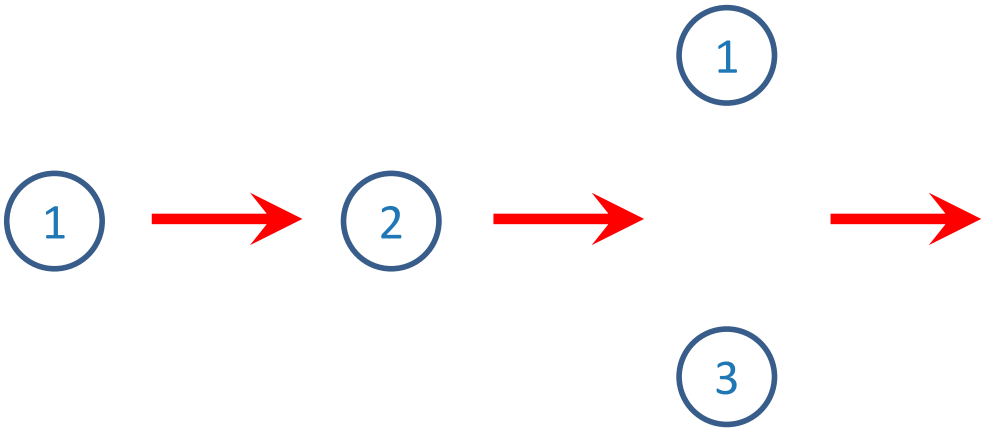
- This requires counting all the anonymous walks of type a_k that initiate from node v_i, for all nodes $v_i \in V$ in the graph G.
- However, the number of the length-l anonymous walks grows exponentially with l.
  - $l = 3$:

Anonymous Walk Embedding

- This requires counting all the anonymous walks of type a_k that initiate from node v_i, for all nodes $v_i \in V$ in the graph G.
- However, the number of the length-l anonymous walks grows exponentially with l.
  - $l = 3$:

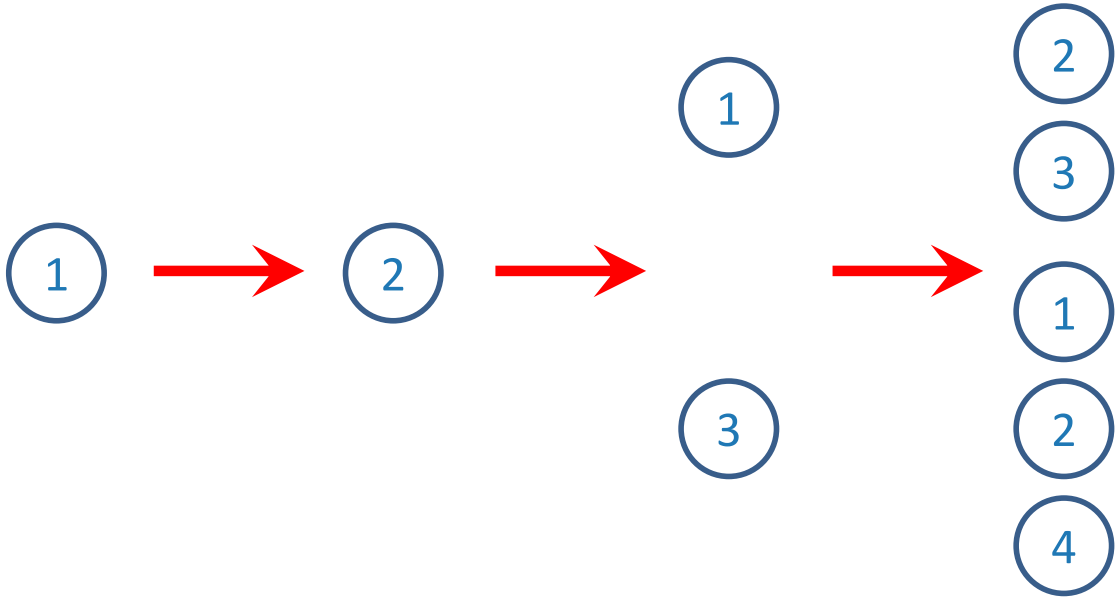
Anonymous Walk Embedding

- This requires counting all the anonymous walks of type a_k that initiate from node v_i, for all nodes $v_i \in V$ in the graph G.
- However, the number of the length-l anonymous walks grows exponentially with l.
  - $l = 3$:



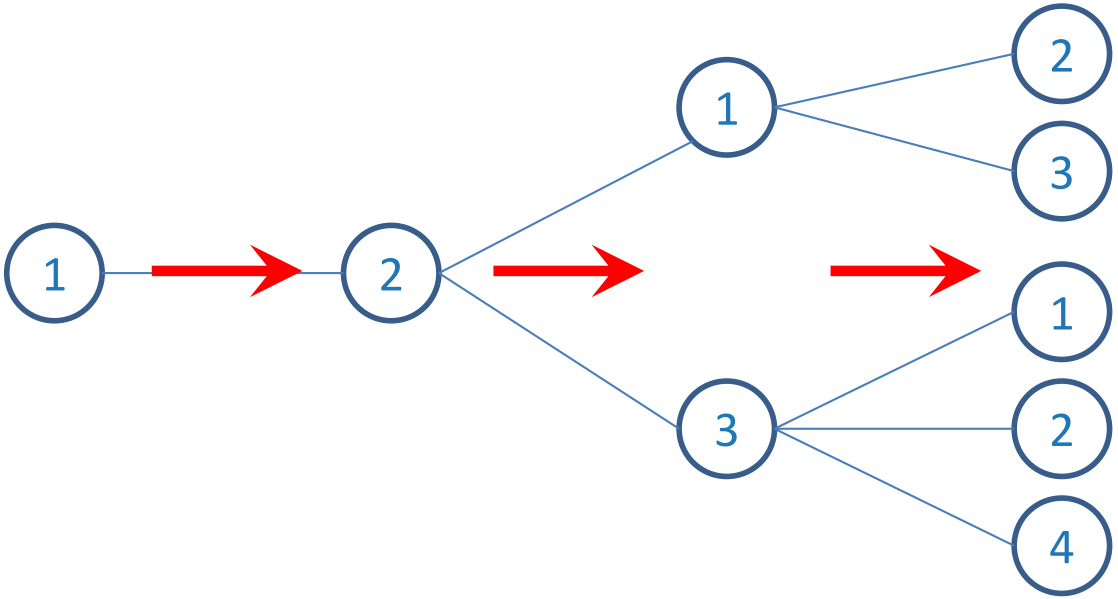
Anonymous Walk Embedding

- This requires counting all the anonymous walks of type a_k that initiate from node v_i, for all nodes $v_i \in V$ in the graph G.
- However, the number of the length-l anonymous walks grows exponentially with l.
  - $l = 3$:



Anonymous Walk Embedding

- This requires counting all the anonymous walks of type a_k that initiate from node v_i, for all nodes $v_i \in V$ in the graph G.
- However, the number of the length-l anonymous walks grows exponentially with l.
  - $l = 3$: There are 5 anonymous walks of length 3.



Anonymous Walk Embedding

- This requires counting all the anonymous walks of type a_k that initiate from node v_i, for all nodes $v_i \in V$ in the graph G.
- However, the number of the length-l anonymous walks grows exponentially with l.
  - $l = 3$: There are 5 anonymous walks of length 3.

$$\mathcal{A}_{l=3} = \{a_1, a_2, a_3, a_4, a_5\}$$

  - $l = 12$: There are $4{,}000{,}000$ anonymous walks of length 12.
- Therefore, a more feasible approach would be to use a sampling method to approximate the probabilities $p(a_k)$.

Anonymous Walk Embedding

- Let A_l be the set of all possible length-l anonymous walks.

$$\mathcal{A}_l = \{a_1, \ldots, a_K\}$$

- Let $\mathbb{D}_l$ be the true distribution over different length-l anonymous walks a_k.
- Let $\mathbb{D}_l^m$ be the empirical distribution over the set of i.i.d. anonymous walks $X^m = (X_1, \ldots, X_m)$ randomly drawn from $\mathbb{D}_l$,

$$\mathbb{D}^m(k) = \frac{1}{m} \sum_{X \sim \mathbb{D}_l}^{m} 1_{[X = a_k]}$$

- We need to find the minimum number of samples m such that $\mathbb{D}_l^m$ is an accurate approximation of $\mathbb{D}_l$.

Anonymous Walk Embedding

- To ensure that $\mathbb{D}_l^m$ accurately approximate $\mathbb{D}_l$, we need to sample at least M anonymous walks, where

$$M = \left\lceil \frac{2}{\varepsilon^2} \left(\log\left(2^K - 2\right) - \log(\delta) \right) \right\rceil$$

Anonymous Walk Embedding

- To ensure that $\mathbb{D}_l^m$ accurately approximate $\mathbb{D}_l$, we need to sample at least M anonymous walks, where

$$M = \left\lceil \frac{2}{\varepsilon^2} \left(\log\left(2^K - 2\right) - \log(\delta) \right) \right\rceil$$

- For length $l = 7$, there are $K = 877$ anonymous walks.

  - $\epsilon = 0.5, \delta = 0.05$ M=4888
  - $\epsilon = 0.1, \delta = 0.01$ M=122500

Anonymous Walk Embedding

- So far, we have studied **feature-based** approaches to represent anonymous walk embeddings for graphs.
- An alternative approach is to use a **data-driven** approach to learn graph embeddings based on anonymous random walks.
- This approach learns graph embeddings based on the notion of **co-occurring** anonymous walks.
- Let W be a set of T random walks rooted at node v_i

$$W_i = \{w_1, w_2, \ldots, w_T\}$$

- By anonymizing this set, we get a new set

$$N_a(v_i) = \{a_1, a_2, \ldots, a_T\}$$

Anonymous Walk Embedding

- The multiset of anonymous walks $N_a(v_i)$ rooted at v_i defines the notion of neighboring anonymous walks.
- Let's define **context** walks as a sequence of anonymous walks co-occurring in a window of size $2\Delta + 1$ in $N_a(v_i)$.
- Let a **target** anonymous walk for each such window be the walk occurring in the middle of the sequence.
- Given the observed data, we predict the probability of a target walk for a set of context walks.

$$p\left(a_t \mid a_{t-\Delta}, \ldots, a_{t+\Delta}, G\right)$$

Anonymous Walk Embedding

- Given observed data $N_a(v_i)$, we maximize this probability.
- We want to learn anonymous walk embedding

$$\boldsymbol{Z}_a \in \mathbb{R}^{K \times d_a}$$

and graph embedding

$$\boldsymbol{z}_G \in \mathbb{R}^{d_G}$$

- Therefore, we maximize

$$p(a_t | \boldsymbol{z}_{a_{t-\Delta}}, \ldots, \boldsymbol{z}_{a_{t+\Delta}}, \boldsymbol{z}_G)$$

- For a set of co-occurring anonymous walks rooted at v_i, we maximize

$$\prod_{i=\Delta}^{T-\Delta} p\left(a_t^i \mid \boldsymbol{z}_{a_{t-\Delta}^i}, \ldots, \boldsymbol{z}_{a_{t+\Delta}^i}, \boldsymbol{z}_G\right)$$

Anonymous Walk Embedding

- Using softmax function, we define

$$p(a_t|\boldsymbol{z}_{a_{t-\Delta}},\ldots,\boldsymbol{z}_{a_{t+\Delta}},\boldsymbol{z}_G)=\frac{\exp(f(\boldsymbol{z}_{a_t}))}{\sum_1^K \exp(f(\boldsymbol{z}_{a_k}))}$$

where

$$f\left(\boldsymbol{z}_{a_t}\right)=b+(U)\left(\left[\frac{1}{2\Delta}\sum_{k=-\Delta}^{\Delta}\boldsymbol{z}_{a_{t+k}}\middle|\boldsymbol{z}_G\right]\right)$$

where $U \in \mathbb{R}^{d_a+d_G}$ and $b \in \mathbb{R}$ are some parameters.

- Therefore, taking the log, we maximize

$$\sum_{v_i\in V}\sum_{i=\Delta}^{T-\Delta}\log p\left(a_t^i \mid \boldsymbol{z}_{a_{t-\Delta}^i},\ldots,\boldsymbol{z}_{a_{t+\Delta}^i},\boldsymbol{z}_G\right)$$

Summary

- Edge-level embeddings
- Subgraph level embedding
- Graph level embedding
  - Super node
  - Summation
  - Anonymous walk
- Anonymous walk embedding
  - Feature based
  - Data driven