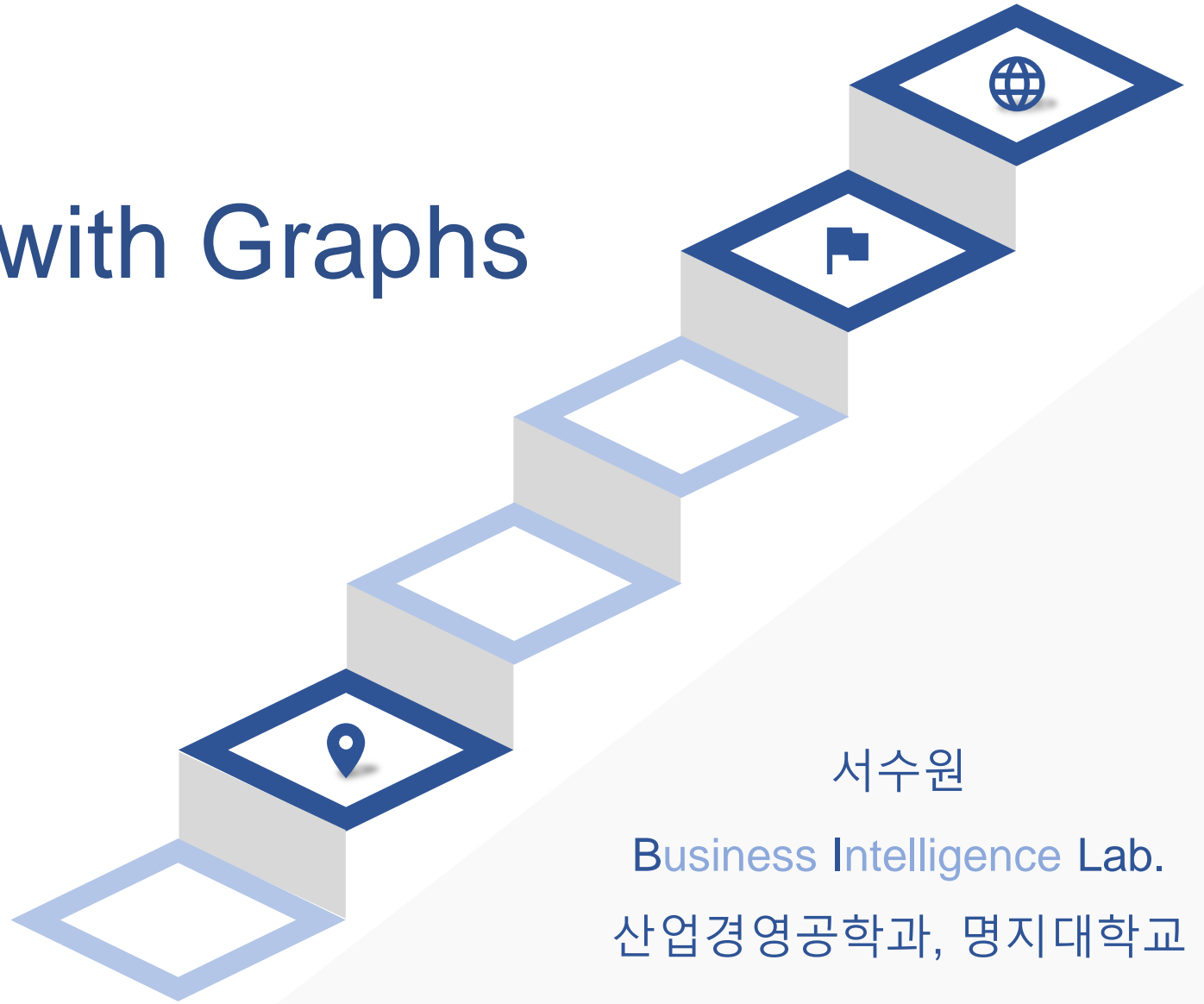




20230713

Machine Learning with Graphs



서수원

Business Intelligence Lab.
산업경영공학과, 명지대학교

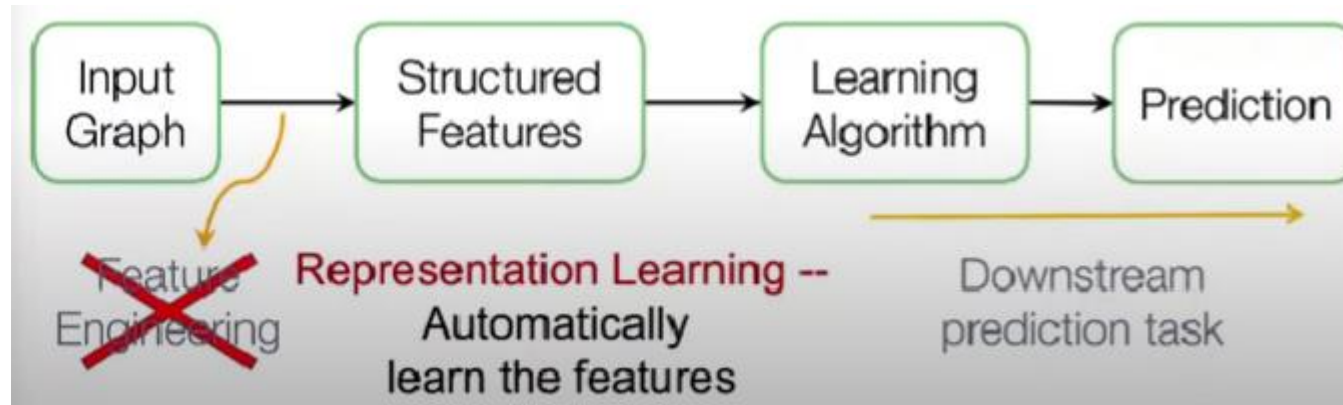
01

**Node
Embeddings**

Recap : Traditional ML for Graphs

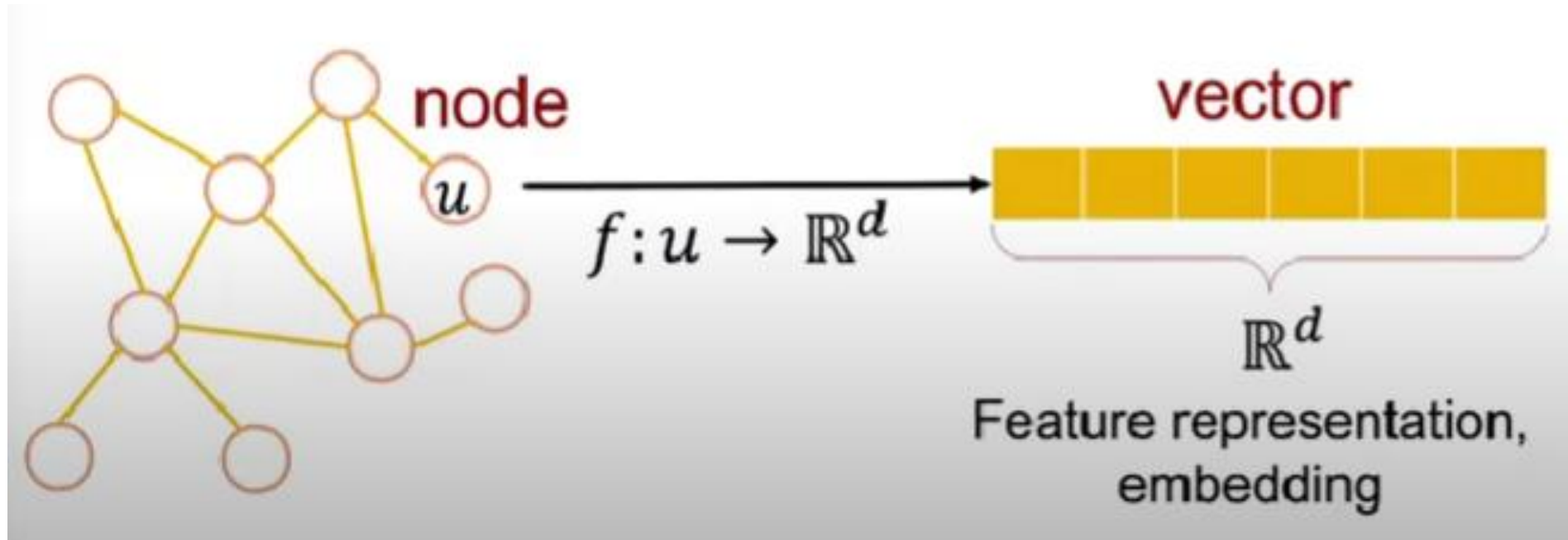
- Traditional ML for Graphs

- 이전의 전통적 연구는 Feature engineering 에 많은 시간을 할애 하게 된다.
 - ✓ 우리가 배우는 것은 Feature engineering 을 자동화 할 수 있는 것에 대해 배운다.
 - ❖ 이는 Representation Learning이라고 한다.



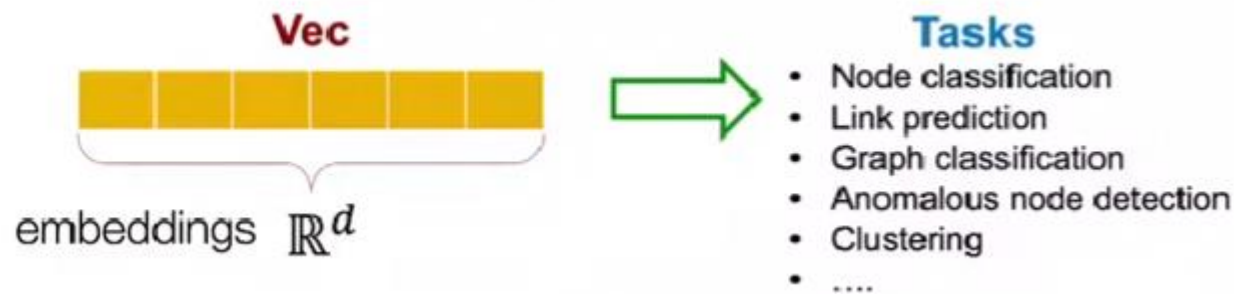
Recap : Traditional ML for Graphs

- Machine Learning with Graphs : Goal
 - Feature learning을 하지 않고 머신러닝을 사용하는 것이다.
 - 만약 개별 노드 수준의 예측이라면, d 차원의 공간으로 맵핑 하는 것을 배운다.
 - 이 맵핑이 자동적으로 된다는 것이 중요하다.



Why Embedding?

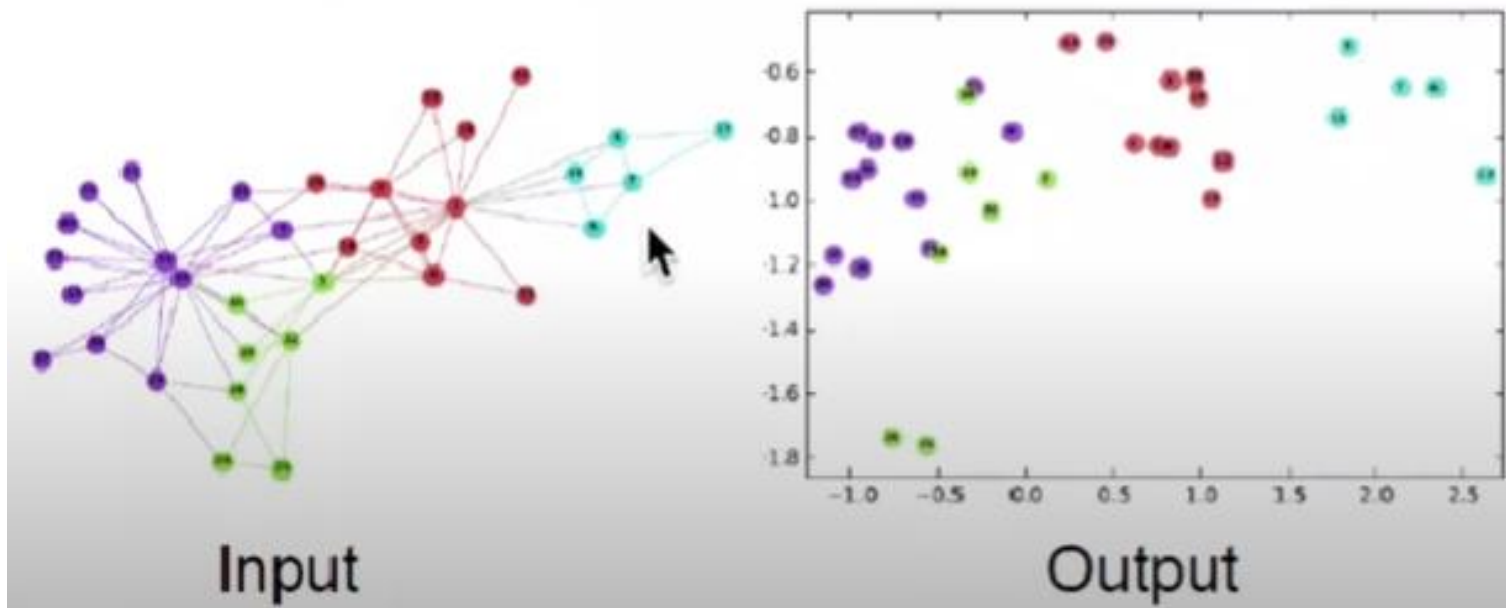
- Embedding
 - 노드간 임베딩의 유사성이 네트워크의 유사성을 나타낸다.
 - ✓ 링크 되어 있는 노드라면 서로 비슷한 공간에 임베딩 된다는 의미이다.
 - 네트워크 구조 정보를 자동으로 인코딩 할 수 있다.
 - 여러 활용이 가능하다.



Embedding : Example

- Embedding

■ 2D embedding of nodes of the Zachary's Karate Club network:



Encode Nodes

- Embedding

- 노드를 특정 차원의 공간에 임베딩 하는 것을 Encode Nodes라고 한다.

- ✓ 이때 유사성은 보존이 된다.

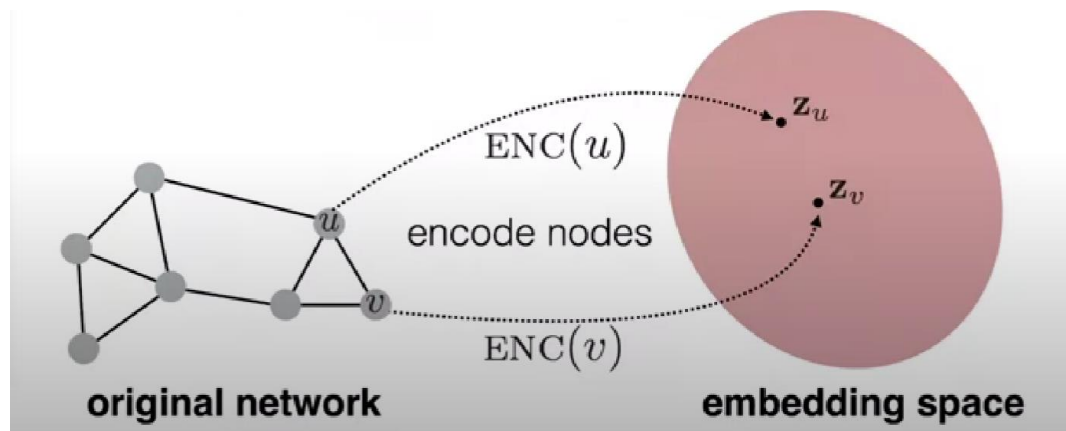
- ✓ U라는 노드를 Z공간에 임베딩 하고, V라는 노드를 Z공간에 임베딩 했을 때 둘은 링크로 연결이 되어 있으므로, 가깝게 임베딩된 것을 볼 수 있다.

- ❖ 가깝다의 기준은 내적을 통해 구할 수 있다.

- ❖ DEC, 디코더는 임베딩된 것을 유사성 함수로 맵핑한다.

$$\text{DEC}(\mathbf{z}_v^T \mathbf{z}_u)$$

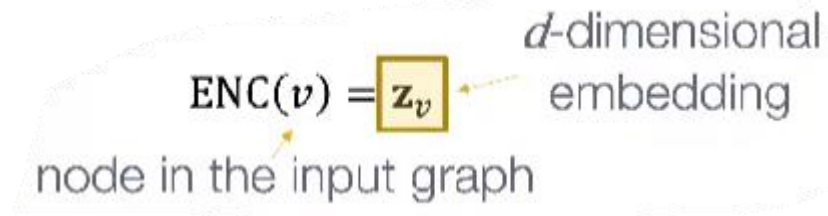
- ✓ **Similarity(u,v), 즉 네트워크에서 유사성의 개념을 정의 하고 유사성을 임베딩 하기위한 목적함수를 정의 하는게 앞으로 배울 내용이다.**



$$\text{Goal: } \underset{\text{in the original network}}{\text{similarity}(u, v)} \approx \underset{\text{Similarity of the embedding}}{\mathbf{z}_v^T \mathbf{z}_u}$$

Two Key Components

- Encoder
 - 노드를 저차원의 벡터로 맵핑 하는 것을 의미한다.



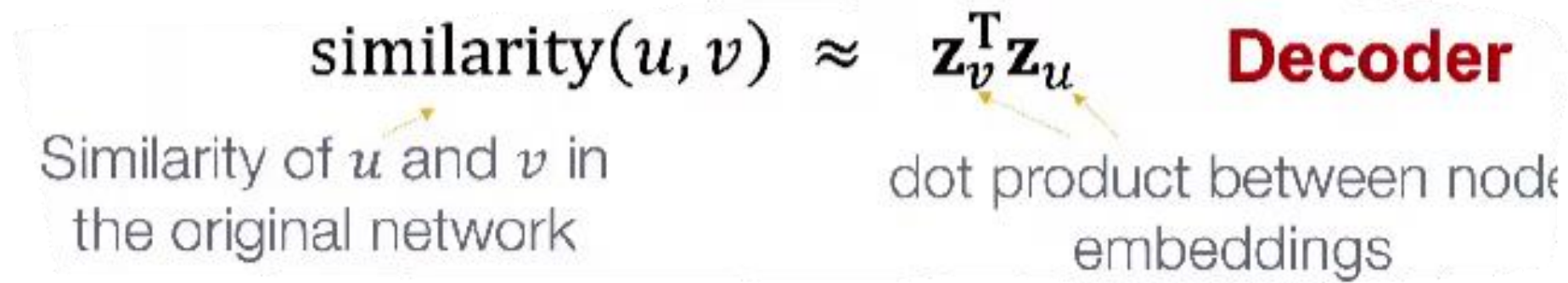
The diagram illustrates the encoder component. It shows the equation $\text{ENC}(v) = \mathbf{z}_v$. A yellow box highlights \mathbf{z}_v , with a dashed arrow pointing to the text " d -dimensional embedding". Another dashed arrow points from the text "node in the input graph" to v in the equation.

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

d -dimensional embedding

node in the input graph

- Similarity Function
 - 벡터공간에 맵핑된 노드들의 관계가 원래 네트워크의 노드들의 관계와 유사하게 맵핑 되어야 한다.
 - ✓ Random walks를 통해 정의 할 예정이다.



The diagram illustrates the similarity function. It shows the equation $\text{similarity}(u, v) \approx \mathbf{z}_v^T \mathbf{z}_u$. The word "Decoder" is written in red to the right of the equation. A dashed arrow points from the text "Similarity of u and v in the original network" to $\text{similarity}(u, v)$. Another dashed arrow points from the text "dot product between node embeddings" to $\mathbf{z}_v^T \mathbf{z}_u$.

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

Decoder

Similarity of u and v in the original network

dot product between node embeddings

Shallow Encoding

- Encoding

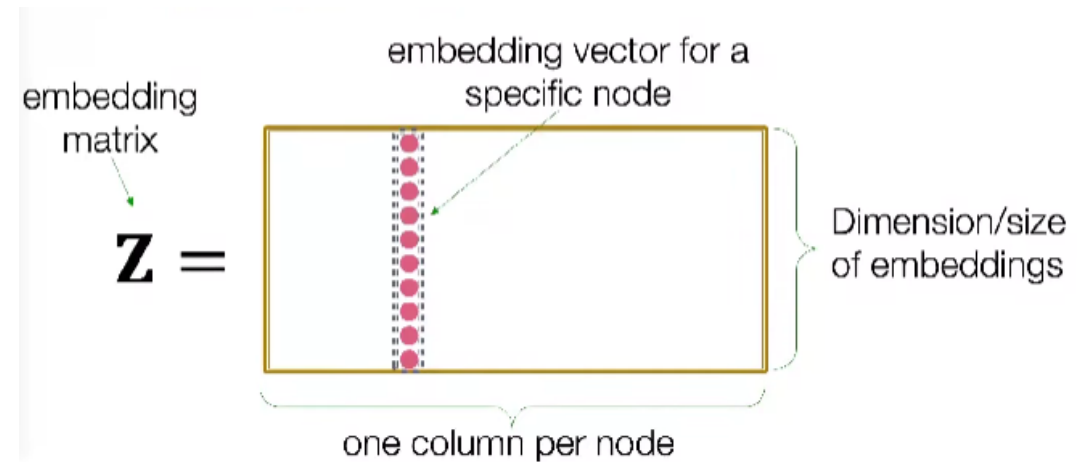
- 인코더는 embedding-lookup으로 봐도 된다.

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

- 우리의 목표는 $d \times v$ 의 크기를 가진 \mathbf{z} 를 배우는 것이다.
 - ✓ v 는 노드의 수를 의미한다. 즉 모든 노드가 할당된 공간이 있다는 것을 의미한다.

$\mathbf{Z} \in \mathbb{R}^{d \times |V|}$ matrix, each column is a node embedding [what we learn / optimize]

$\mathbf{v} \in \mathbb{I}^{|V|}$ indicator vector, all zeroes except a one in column indicating node v



02

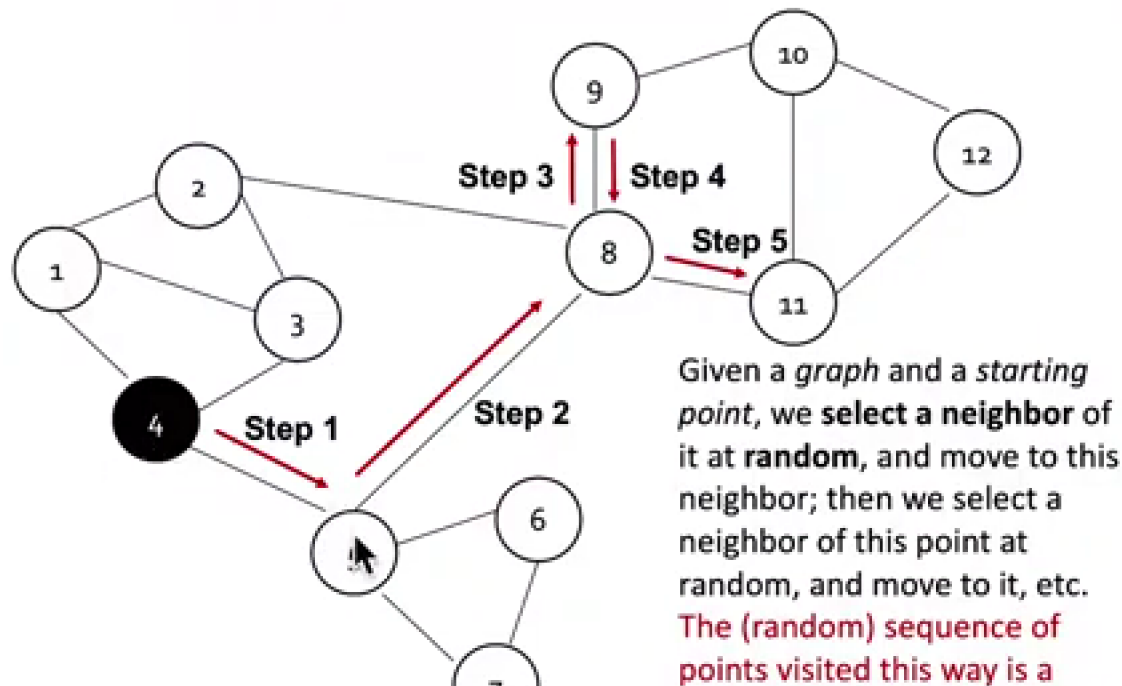
Random Walk Approaches for Node Embeddings

Random Walk Approaches for Node Embeddings

- Notation
 - Vector Z_u
 - ✓ 노드 u 의 임베딩을 의미한다.
 - $P(V|Z_u)$
 - ✓ U 에서 V 로 Random Walks 할 확률을 의미한다.
 - ✓ 확률을 예측하기 위해 비선형 함수가 사용된다.

Random Walk Approaches for Node Embeddings

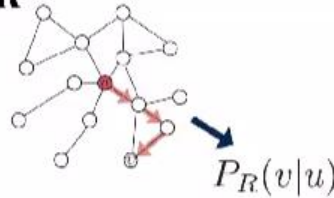
- Random Walk
 - 단순히 생각하면 연결되어 있는 노드로 뺏어나가는 것을 의미한다.



Random Walk Approaches for Node Embeddings

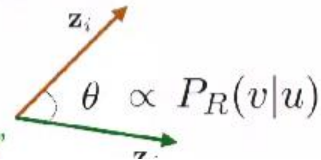
- Random Walk Embeddings
 - 먼저 v 에 갈 확률을 추정해야 한다.
 - 랜덤워크 통계치를 인코딩 하는 방식을 통해 최적화된 임베딩을 수행한다.

1. Estimate probability of visiting node v on a random walk starting from node u using some random walk strategy R



2. Optimize embeddings to encode these random walk statistics:

Similarity in embedding space (Here: dot product= $\cos(\theta)$) encodes random walk "similarity"



Random Walk Approaches for Node Embeddings

- Random Walk
 - 랜덤워크는 표현하기 좋고 효율적이다.
 - ✓ 정보와 함께 이웃과 그 사이의 유사성을 확률로 정의 할 수 있다.
 - ✓ 링크만 보기 때문에 훨씬 계산 효율적이다.

- Feature Learning as Optimization

- Given $G = (V, E)$,
- Our goal is to learn a mapping $f: u \rightarrow \mathbb{R}^d$:
 $f(u) = \mathbf{z}_u$

- Log-likelihood objective:

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

- $N_R(u)$ is the neighborhood of node u by strategy R

Random Walk Approaches for Node Embeddings

- Random Walk Optimization
 - 밑과 같이 다시 표현을 할 수 있다.

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

- 파라미터 $P(V|Z_u)$ 는 소프트맥스를 이용한다.

Parameterize $P(v|\mathbf{z}_u)$ using softmax:

$$P(v|\mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^T \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^T \mathbf{z}_n)}$$

Random Walk Approaches for Node Embeddings

- Random Walk Optimization
 - 앞선 수식을 한번에 나타내면 밑과 같이 나타낼 수 있다.
 - ✓ 우도함수가 최소가 되게 하는 Z값을 찾는게 핵심이다.

Putting it all together:

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

sum over all nodes u sum over nodes v seen on random walks starting from u predicted probability of u and v co-occurring on random walk

Random Walk Approaches for Node Embeddings

• Random Walk Optimization

- 단순히 수식대로 하면 너무 계산량이 늘어난다.
 - ✓ V^2 인 이유는 먼저 시작 노드에 대해 네트워크의 모든 노드를 합산하고, Softmax를 정규화 할 때 또 모든 노드에 대해서 합산을 하니 계산이 불필요하게 중복되고 복잡해진다.
- 이는 소프트 맥스에서 문제가 있는 것 임으로 이를(분모 구하는 것을) 근사화 하면 해결이 된다.

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

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

Random Walk Approaches for Node Embeddings

- Negative Sampling

- 전체 노드가 아닌 부분집합에 대해서만 정규화를 하자는 것 이다.
- NLP에선 타겟 단어와 연관성이 없을 것 이라고 추정되는 단어를 부분집합으로 구성한다.
- 네트워크에서는 node degree가 높은 노드들로 구성한다.
 - ✓ 이때 선택되는 부분집합의 노드의 수는 5~20개로 설정한다.

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

logistic regression (sigmoid func.) to distinguish the target node v from nodes n_i sampled from background distribution P_V .
More at <https://arxiv.org/pdf/1402.3722.pdf>

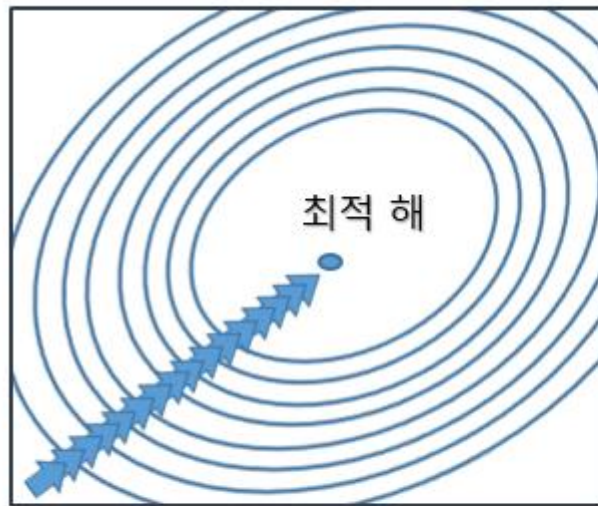
$$\approx \log\left(\sigma(\mathbf{z}_u^T \mathbf{z}_v)\right) - \sum_{i=1}^k \log\left(\sigma(\mathbf{z}_u^T \mathbf{z}_{n_i})\right), n_i \sim P_V$$

sigmoid function
(makes each term a "probability" between 0 and 1)

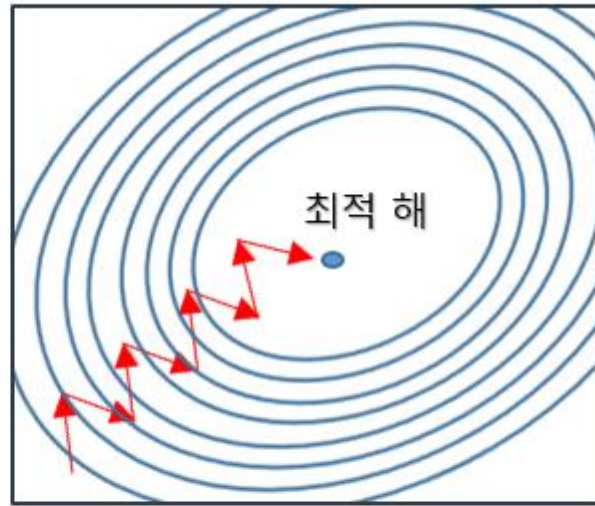
random distribution over nodes

Random Walk Approaches for Node Embeddings

- Stochastic Gradient Descent
 - 목적함수를 정의 후 최적화를 할 때 사용한다.



경사 하강법



확률적 경사 하강법

[그림 1] 경사 하강법 및 확률적 경사 하강법 [2]

Random Walk Approaches for Node Embeddings

- 요약 정리
 - 가까운 거리로 랜덤워크를 실행 한다.
 - u 에서 시작하는 랜덤워크를 통해 이웃의 정보를 수집한다.
 - 확률적 경사 하강법을 이용하여 표현을 최적화 하는 z 값을 찾는다.
 - 네거티브 샘플링을 통해 효율성을 극대화 한다.

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

We can efficiently approximate this using

Random Walk Approaches for Node Embeddings

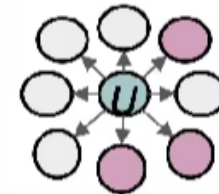
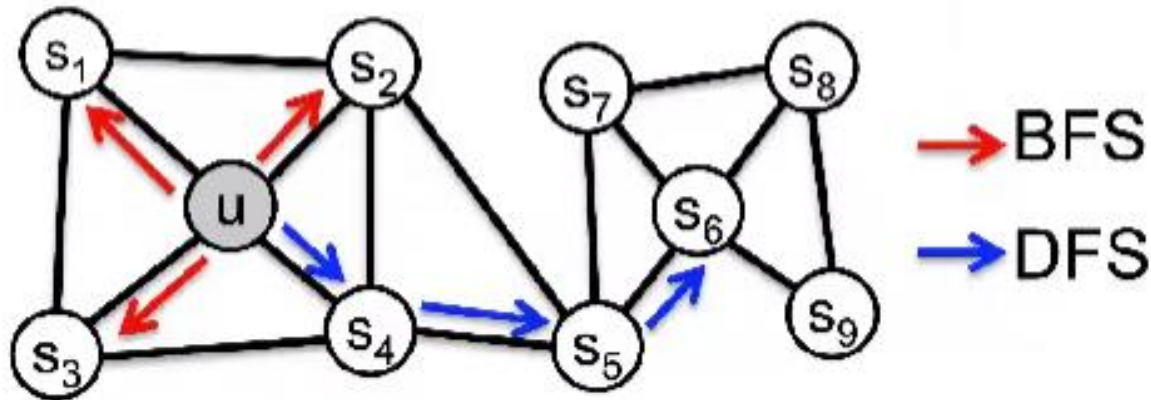
- Node2Vec
 - 더 나은 랜덤워크를 위해 사용되는 방법이다.
 - ✓ 기존 방법은 네트워크에서 파악되는 connectivity pattern에 대해 제대로 학습을 하지 못한다는 단점이 있었다.
 - 최대 우도 최적화문제를 해결하기 위해 사용된다.
 - 여기서 중요한 점은 우리가 이웃에 대한 유연한 개념을 가지고 있다는 것이다.
 - Node2Vec을 통해 랜덤워크를 생성할 수 있다.
 - Word2Vec의 일반화된 형태로도 볼 수 있다.
 - ✓ I -> am -> a -> boy, network는 양방향일 때도 있기 때문이다.

Random Walk Approaches for Node Embeddings

• Node2Vec

– BFS 와 DFS를 사용 할 수 있다.

- ✓ BFS는 비슷한 노드는 서로 공통적으로 연결되어 있는 노드가 많다는 개념을 기반으로 한다.
 - ❖ 지역적으로 탐색하고 지역에 대한 정보를 준다.
- ✓ DFS는 비슷한 노드는 네트워크에서 서로 비슷한 구조적인 위치에 존재한다는 것을 기반으로 한다.
 - ❖ Global(네트워크 전체의 구조)를 탐색하고 전체 구조에 대해 정보를 준다



BFS:
Micro-view of
neighbourhood



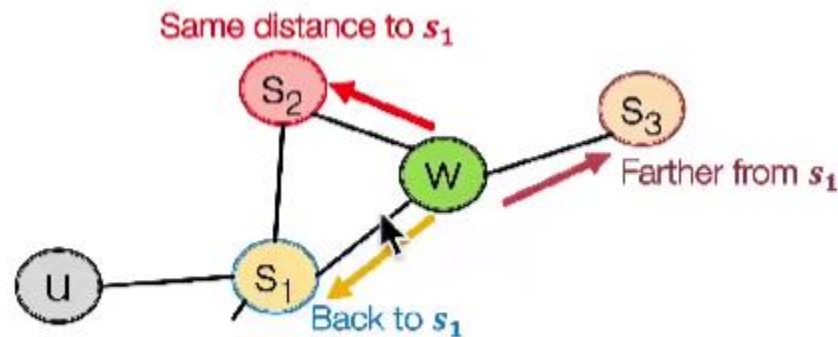
DFS:
Macro-view of
neighbourhood

Random Walk Approaches for Node Embeddings

- Interpolation BFS and DFS

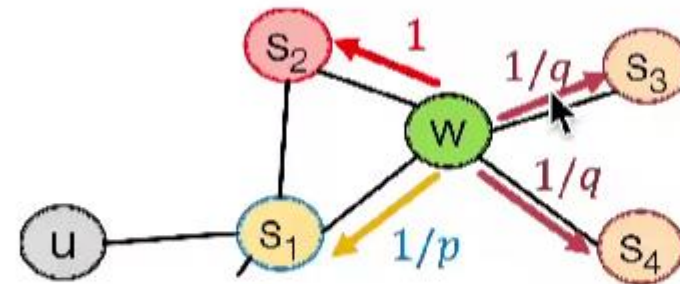
- 두개의 파라미터가 있다.

- ✓ Return parameter p
 - ❖ 이전 노드로 다시 되돌아 오는 것을 정의한다.
 - ✓ In-out parameter q
 - ❖ 직관적으로 BFS와 DFS의 비율을 의미한다.
 - ✓ 밑은 S_1 에서 W 로 간 상황을 가정한다.



Idea: Remember where the walk came from

- Walker came over edge (s_1, w) and is at w .
Where to go next?

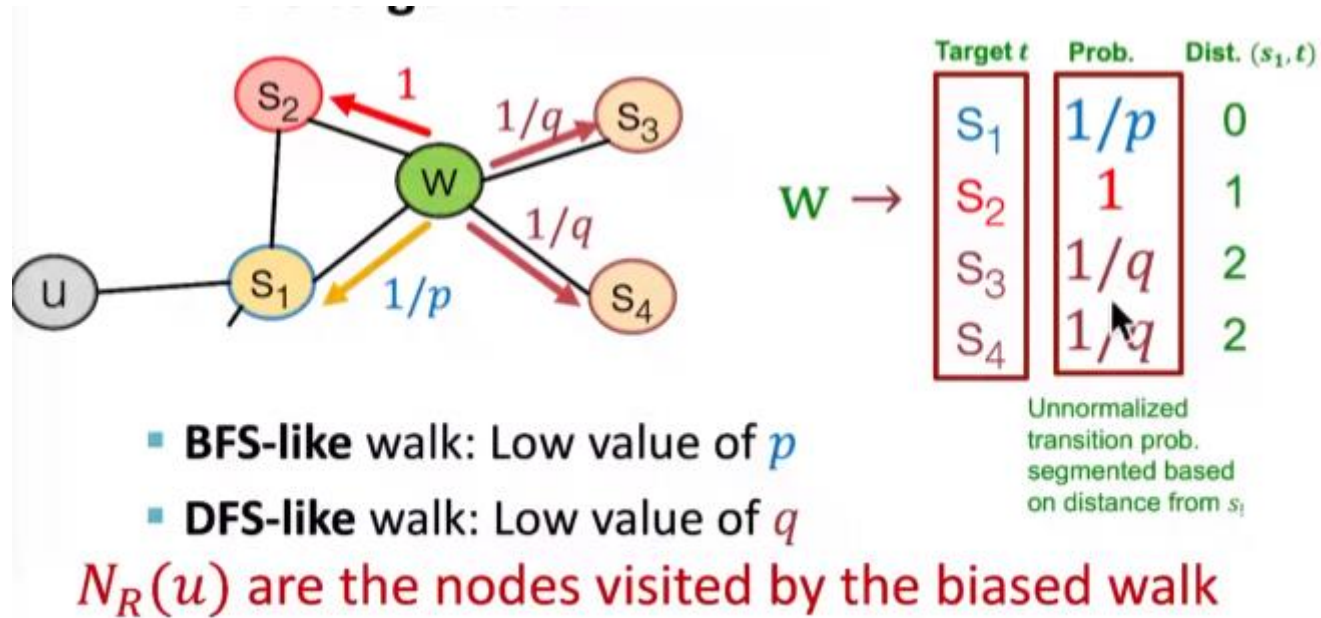


$1/p, 1/q, 1$ are
unnormalized
probabilities

- p, q model transition probabilities
 - p ... return parameter
 - q ... "walk away" parameter

Random Walk Approaches for Node Embeddings

- Biased Random Walks
 - W 에 대해 정규화 되지 않은 값들을 구할 수 있다.
 - ✓ 이 값들을 합해서 1이 되도록 정규화를 한다.
 - ✓ 4가지 값 중 하나를 고르게 p 와 q 값을 정한다.
 - ❖ 이때 일정한 확률이 아님으로 편향되었다고 표현을 하는 것 같다.



1/19/21

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

45

Random Walk Approaches for Node Embeddings

- Node2Vec Algorithm
 - 랜덤워크의 확률을 계산한다.
 - 노드 u 에서 시작하는 편향된 랜덤워크를 시뮬레이션 한다.
 - 목적함수를 최적화 한다.
 - ✓ 앞선 방식과 동일한 경사하강법을 사용 한다.
 - 링크에 수에 대해 시간이 선형으로 증가하는 장점이 있다.
 - 단점으로는 모든 노드에 대해 개별적으로 임베딩 해야 한다는 것이 있다.

03

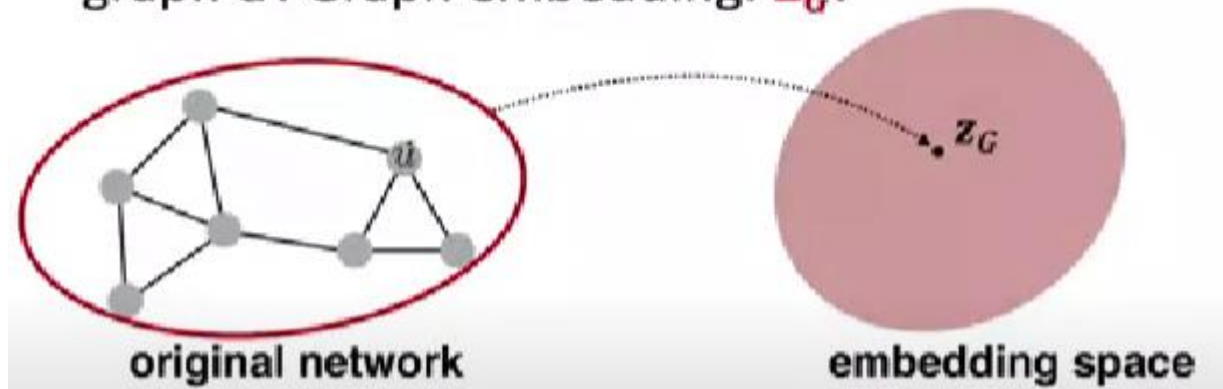
**Embedding
Entire Graphs**

Embedding Entire Graphs

- Goal

- 앞에선 노드단위 임베딩 이었다면 그래프 전체, 혹은 그래프 안의 부분집합을 임베딩 하는 것이 목표이다.
 - ✓ 예시로는 어떤 분자가 독성이 있는지, 무독한지를 확인 할 때 활용 가능하다.

- **Goal:** Want to embed a subgraph or an entire graph G . Graph embedding: \mathbf{z}_G .



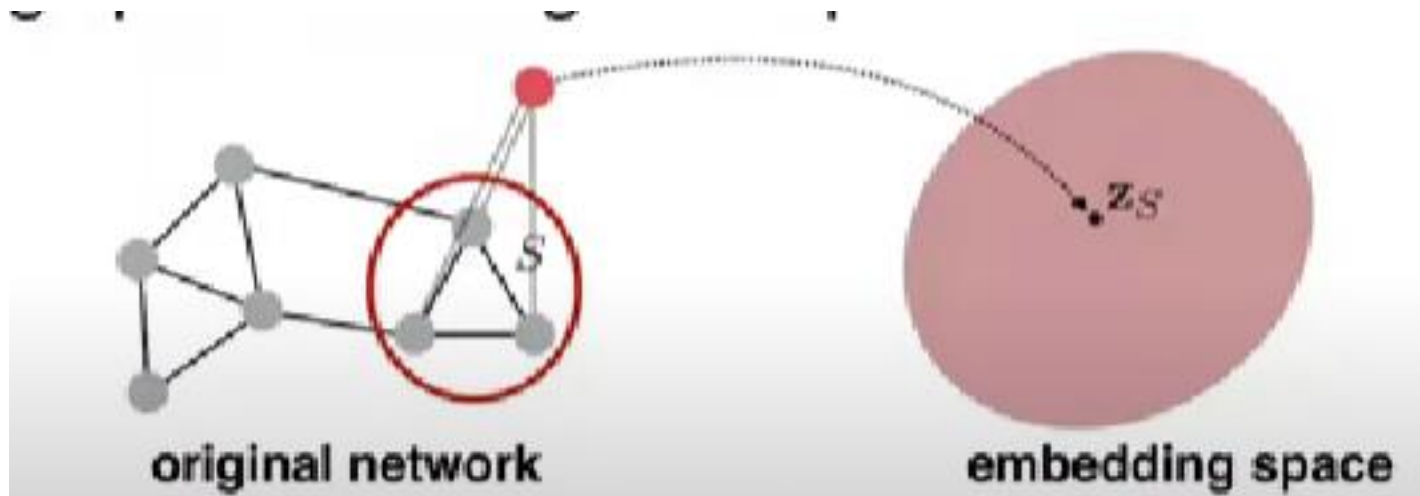
Embedding Entire Graphs

- Approach1
 - 앞서 했던 임베딩 방식을 통해 노드단위 임베딩을 하고, 그 합을 통해 그래프의 위치를 구하는 접근이다.
 - ✓ 그래프 임베딩은 단순히 그래프에 있는 노드 임베딩의 합이다.

$$\mathbf{z}_G = \sum_{v \in G} \mathbf{z}_v$$

Embedding Entire Graphs

- Approach2
 - 전체 그래프 혹은 부분 그래프를 노드들과 연결된 가상의 노드를 통해 임베딩 한다.
 - ✓ 앞서 설명했던 방법들을 활용해 임베딩 한다.



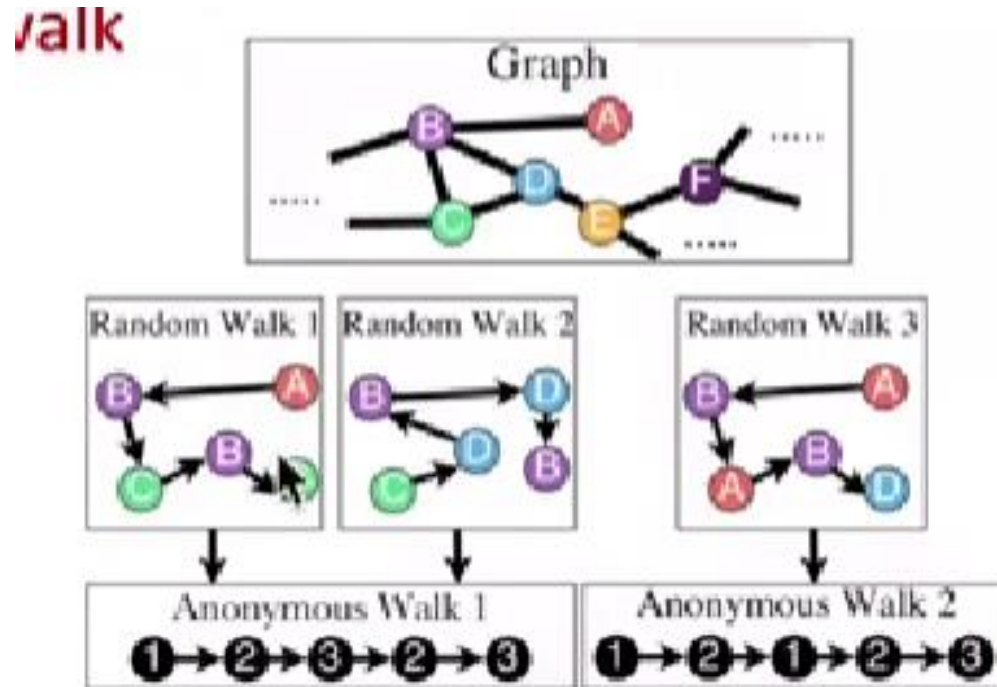
Embedding Entire Graphs

Approach3

Anonymous Walk Embeddings

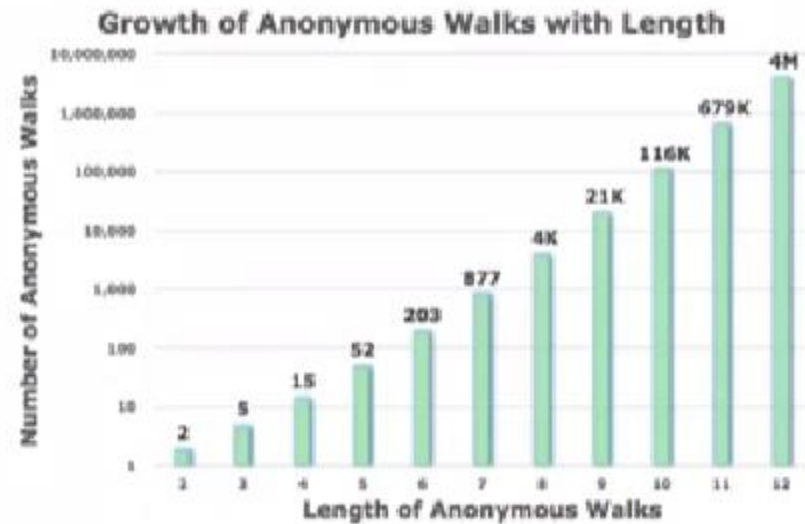
- ✓ 랜덤워크 과정 중 처음 방문하는 노드의 인덱스와 anonymous walks의 상태가 일치한다
 - ❖ 랜덤워크 과정에 따라 레이블을 다시 설정한다.
- ✓ 1 :A or C 즉 과정에 따라 레이블이 달라지기 때문에 Anonymous를 붙인 것 같다.

- 지나온 노드가 아닌 지나온 순서만을 고려 한다고 볼 수 있다.



Embedding Entire Graphs

- Anonymous Walk Embeddings
 - Anonymous Walk의 길이가 궁금하다.
 - ✓ 예) 3번 움직이는 Walks에는 5개의 Anonymous Walks가 존재한다.
 - ❖ 동일한 노드에 3번 머물 수도 있고, 한 번 움직일 수도 있고, 계속 움직일 수도 있다.



Number of anonymous walks grows exponentially:

- There are 5 anon. walks w_i of length 3:

$w_1=111$ $w_2=112$ $w_3=121$ $w_4=122$ $w_5=123$

Embedding Entire Graphs

- Anonymous Walk
 - Anonymous Walks를 시뮬레이션 했을 때 L step의 w_i 를 구한다.
 - ✓ 그래프를 확률분포로 표현 할 수 있다.
 - ✓ 그래프의 i 번째 값은 anonymous walk의 확률값이다.

■ For example:

- Set $l = 3$
- Then we can represent the graph as a 5-dim vector
 - Since there are 5 anonymous walks w_i of length 3: 111, 112, 121, 122, 123
- $\mathbf{Z}_G[i] = \text{probability of anonymous walk } w_i \text{ in } G$



- Anonymous Walk
 - Anonymous Walks의 적정 수를 구하는 방법

Embedding Entire Graphs

- Learn Walk Embeddings
 - 앞선 방법을 더 향상시킨 방법이다.
 - ✓ 각 W 가 발생 할 때 그것의 임베딩인 z 를 배울 수 있다는 것이다.
 - ❖ anonymous walk 를 한 z_i 와 함께 그래프 임베딩인 z_g 를 함께 배운다는 의미이다.

Embedding Entire Graphs

- Learn Walk Embeddings

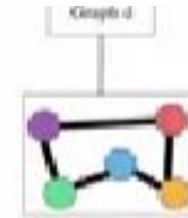
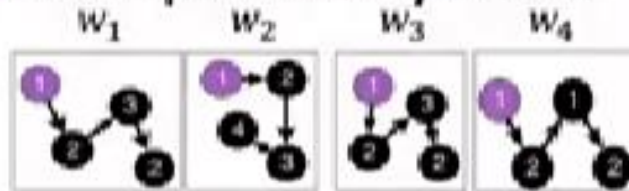
- Z_G 라는 파라미터를 얻는다.

- ✓ 학습할 전체 그래프의 임베딩을 의미한다.
 - ✓ Anonymous random walks를 한다.
 - ✓ 델타라는 윈도우 사이즈를 통해 여러 값을 가지고 예측을 한다.

- A vector parameter \mathbf{Z}_G for input graph

- The embedding of entire graph to be learned

- Starting from **node 1**: Sample anonymous random walks, e.g.



- Learn to predict walks that co-occur in Δ -size window (e.g. predict w_2 given w_1, w_3 if $\Delta = 1$)

- Objective:

$$\max \sum_{t=\Delta}^{T-\Delta} \log P(w_t | w_{t-\Delta}, \dots, w_{t+\Delta}, \mathbf{Z}_G)$$

Embedding Entire Graphs

- Learn Walk Embeddings

- 아이디어는 길이가 l인 노드 U각각에서 T개의 랜덤워크를 실행한다는 것이다.
 - ✓ 우리는 이제 이웃 노드의 집합이 아닌, anonymous walk의 집합이 된다.

$$N_R(u) = \{w_1^u, w_2^u \dots w_T^u\}$$

- ✓ 다음 anonymous walk를 예측하는 값을 최대로 하는 것이 목표이다.

Objective: $\max_{Z, d} \frac{1}{T} \sum_{t=\Delta}^{T-\Delta} \log P(w_t | \{w_{t-\Delta}, \dots, w_{t+\Delta}, \mathbf{z}_G\})$

- $P(w_t | \{w_{t-\Delta}, \dots, w_{t+\Delta}, \mathbf{z}_G\}) = \frac{\exp(y(w_t))}{\sum_{i=1}^{\eta} \exp(y(w_i))}$ ← All possible walks (require negative sampling)
- $y(w_t) = b + U \cdot \left(\text{cat}\left(\frac{1}{2\Delta} \sum_{i=-\Delta}^{\Delta} z_i, \mathbf{z}_G\right) \right)$

Embedding Entire Graphs

- Learn Walk Embeddings

- 아이디어는 길이가 l인 노드 u 각각에서 T개의 랜덤워크를 실행한다는 것이다.
 - ✓ 우리는 이제 이웃 노드의 집합이 아닌, anonymous walk의 집합이 된다.

$$N_R(u) = \{w_1^u, w_2^u \dots w_T^u\}$$

- ✓ 다음 anonymous walk를 예측하는 값을 최대로 하는 것이 목표이다.
- 그래프 분류등의 Task에 사용 된다.

Objective: $\max_{Z, d} \frac{1}{T} \sum_{t=\Delta}^{T-\Delta} \log P(w_t | \{w_{t-\Delta}, \dots, w_{t+\Delta}, \mathbf{z}_G\})$

- $P(w_t | \{w_{t-\Delta}, \dots, w_{t+\Delta}, \mathbf{z}_G\}) = \frac{\exp(y(w_t))}{\sum_{i=1}^{\eta} \exp(y(w_i))}$ ← All possible walks (require negative sampling)
- $y(w_t) = b + U \cdot \left(\text{cat}\left(\frac{1}{2\Delta} \sum_{i=-\Delta}^{\Delta} z_i, \mathbf{z}_G\right) \right)$

Embedding Entire Graphs

- How to use embeddings Z_i of nodes
 - Link 예측에는 양 노드의 embedding을 함께 활용한다.
 - 군집을 찾을 때 활용 가능하다.
 - 노드 분류에 활용 가능하다.
 - 그래프 분류에 활용 가능하다.