Node2vec
: Scalable Feature Learning for Networks

Aditya Grover, Jure Leskovec (Stanford University)

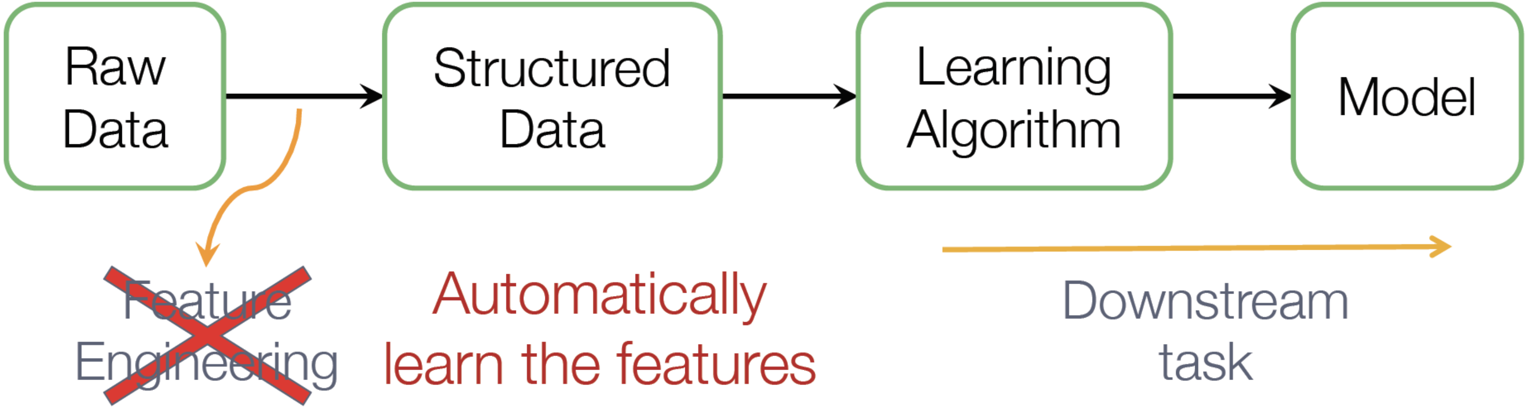
MINHO KIM

Typical feature representation solution의 한계

- □ 머신러닝 알고리즘은 **feature vector representation**를 필요로 함

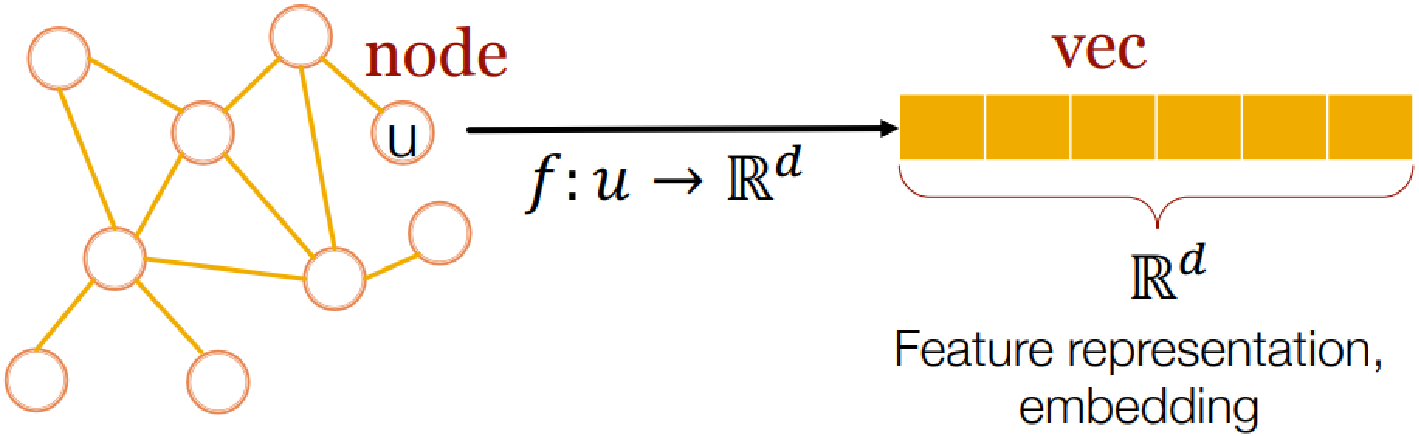
BUT

- □ 전문가 지식에 기반한 **Hand-engineering**
- □ **Task**마다 **feature engineering**이 필요함
- □ 다양한 **prediction tasks**에 일반화해서 적용할 수 없음



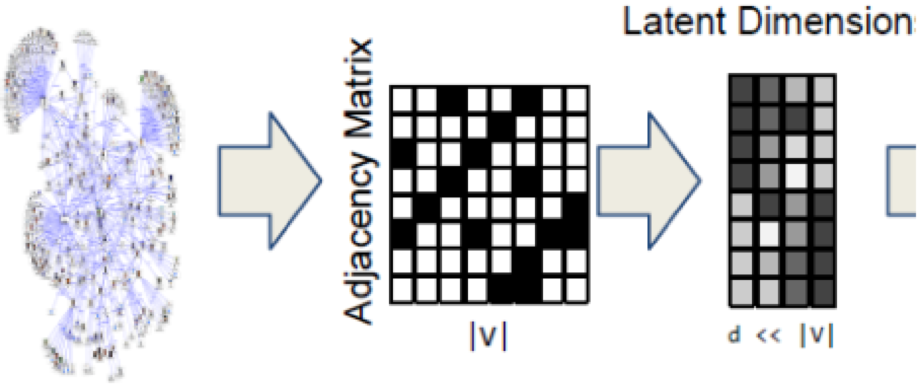
Representation Learning

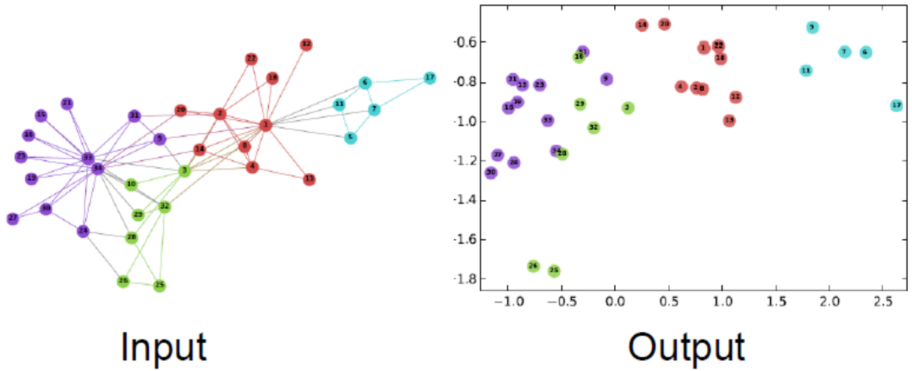
- **representation learning(feature learning)**
  - 데이터를 기반으로 특징을 자동으로 추출할 수 있도록 학습하는 과정임



Why Embedding?

- □ 그래프의 인접 행렬
  - ■ 각 column이 node를 의미 -> 크기가 매우 큼 -> computational efficiency BAD
- □ embedding Matrix
  - ■ 보다 낮은 차원으로 **embedding**하여 computational efficiency 를 얻음
  - ■ 각 column이 node를 의미하지 않고, feature를 의미
  - ■ 그래프의 정보를 담음 -> embedding 공간에서 유사하면 node끼리도 유사





Classic approaches

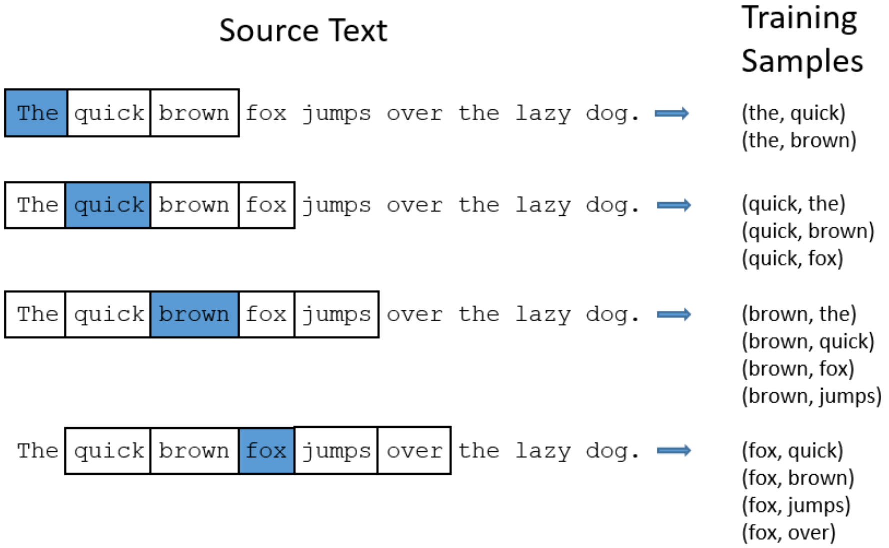
- **Matrix factorization methods**
  - 차원 축소 기술
  - PCA
  - Multi-Dimensional Scaling

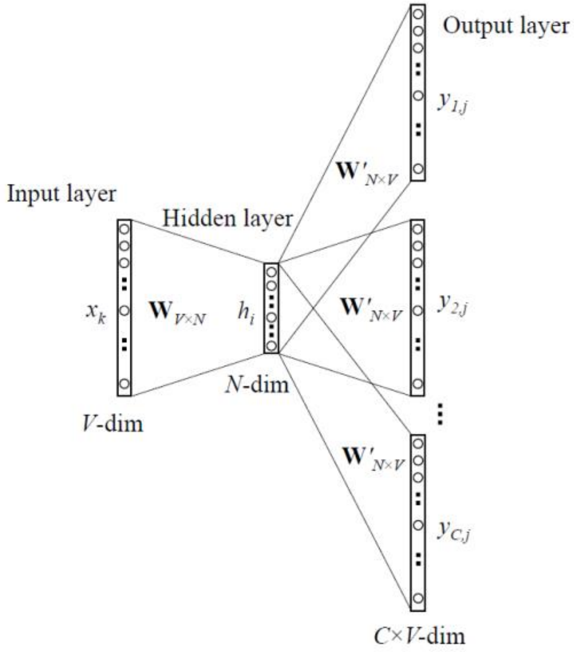
- **poor performance**
- **large real-world networks**에 적용하기 힘듬
  - 고유값 분해가 필요함(Matrix의 크기가 커질수록 연산량이 증가)

Related work

□ Skip-gram model (Word2vec)

- 중심 단어를 통해 주변단어를 예측하는 모델

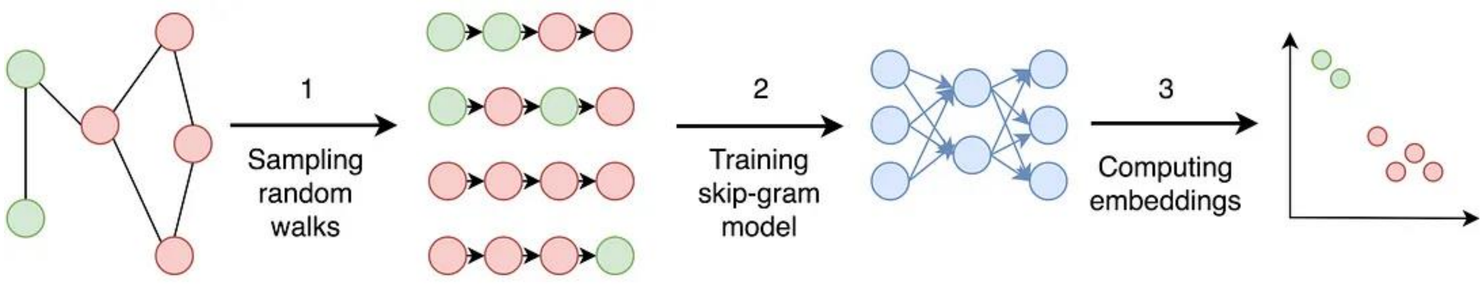




Classic approaches

□ Deep walk

- Random walk 기반 node embedding
- objective that seeks to preserve local neighborhoods of nodes
- Objective는 SGD를 통해 최적화 할 수 있음.



Phases of DeepWalk approach

□ 한계

- network neighborhood가 homophily에 의해서만 정의됨
- insensitive to connectivity patterns unique to networks

Goal

- homophily
  - 노드들이 속한 커뮤니티(가까운 거리)
  - 가까이 있는 커뮤니티에 같이 속해 있으면 비슷한 임베딩
  - Ex) u and s1
- structural equivalence
  - 네트워크에서의 구조적 역할
  - 비슷한 구조적 역할을 하면 비슷한 임베딩
  - Ex) u and s6

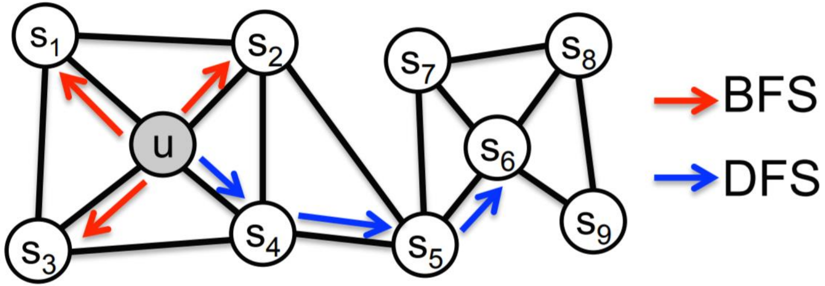
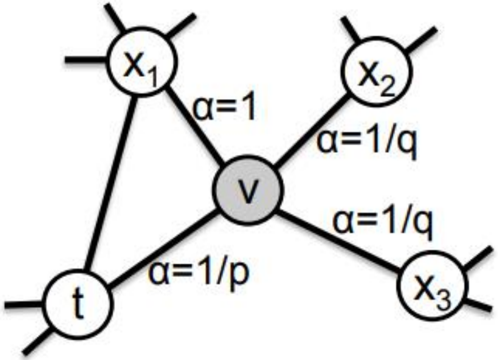


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

Sampling in node2vec

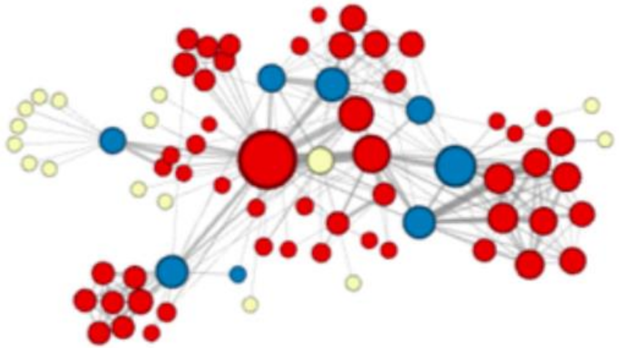
- 2nd order random walks
  - P: return parameter, controls the likelihood of immediately revisiting a node
  - q: In-out parameter, allows the search to differentiate between In and Out nodes
  - p ↑, q ↓: 새로운 노드로 가는 경향성 높아짐,멀리 있는 노드로 뻗어감
  - p ↓, q ↑ : 새로운 노드로 가는 경향성 낮아짐,근처 노드에서 맴돔

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

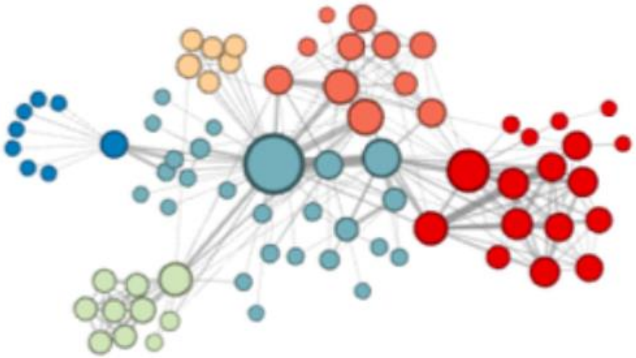


Sampling

- BFS
  - Structural equivalence
  - p ↑, q ↓



- DFS
  - Homophily
  - p ↓, q ↑



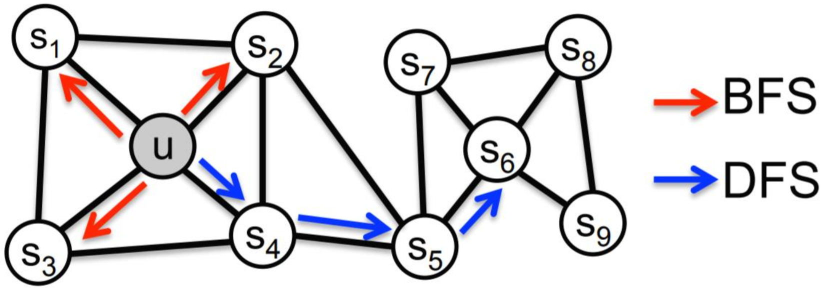


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

FEATURE LEARNING FRAMEWORK

- **objective function**

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

- **standard assumptions**
  - Conditional independence

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

  - Symmetry in feature space

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

Benefit

- sensitive to connectivity patterns unique to networks

- Flexible
  - P,q를 조절하여 관찰하려는 네트워크의 특성 조절가능

Experiment

- ☐ **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**

- ☐ **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

Coclusion

- Representation Learning
  - Task마다 feature engineering을 피하기 위해
- 기존의 **Node Embedding 방법들**
  - Matrix factorization methods
  - **Deepwalk**
- **Node2vec**
  - Homophily
  - Structural equivalence
  - 기존의 Deepwalk를 발전시켜서 flexible하게 만들기 위함.