Node2vec

: Scalable Feature Learning for Networks

이혜승

Contents

[*Introduction*]

Introduction

- 네트워크 분석
  1. **node classification**
  2. **Link prediction**

- Issue: **feature vector representation** 이 필요.
- But 현재 기술: 손수 feature 추출함. 일반화 X
  현재의 Classic 방법들: 합리적인 목적함수 최적화x. 선형/비선형 차원축소 – expensive, bad 성능.

  → Aim: feature 를 학습하는 objective function(목적함수)를 정의 & optimize it.(계산효율성과 예측 정확도의 밸런스를 맞춤)

Introduction

- 대안: nodes의 지역 이웃의 보존하고자 하는 objective function 을 설계 → optimization
- 최근의 시도들이 효율적인 알고리즘을 제안하고있으나, rigid notion에 기반함.

1. Homophily: 네트워크의 노드들은 그 노드들이 속한 Communities 기반
2. Structural equivalence: 네트워크에서 노드의 구조적 역할 기반

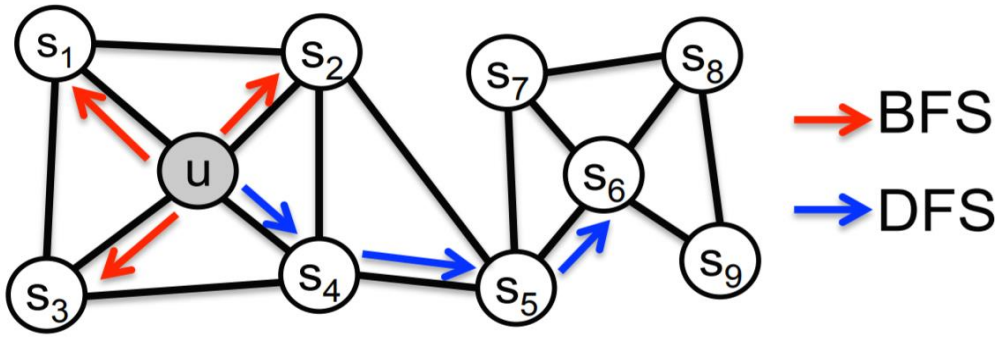


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

1. 노드 u와 노드 s1은 같은 커뮤니티에 속함.
2. 노드 u와 노드 s6은 hub node로써의 구조적 역할.

Introduction

- Real-world network: 두 종류의 equivalence(homophily & structural)의 혼합을 나타냄.

- Thus, **두 가지 원리를 모두 준수하는 node representation을 학습하는 flexible algorithm**이 필요.

→ Feature 학습 알고리즘의 일반화 가능!

→ Node2vec

Introduction

- 풀고자 하는 문제

: homophily와 structural equivalence, 두 가지 원리를 모두 준수하는, 효율적이고 자동화된 feature vector representation algorithm 의 개발.

- 관련 연구들은 이 문제를 어떻게 풀어왔는지

: feature learning approach - 선형 / 비선형 차원 축소.

- 이전 연구들과 달리, 어떻게 문제를 해결했는지

: 유연한 feature learning 알고리즘 사용

[*Feature Learning framework*]

Feature Learning framework

- 두가지 standard 가정(최적화 문제를 쉽게 다루기 위해)

1. **조건부 독립**: 노드 u가 주어졌을 때, 다른 이웃 노드들이 관측될 확률은 모두 독립적.

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

2. **Feature space에서 대칭**: 노드 사이의 관계는 상호 같음.

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

Feature Learning framework

- 두 개의 가정 하에, objective function이 간소화된다.

$$\max_f \sum_{u \in V} \log(N_S(u)|f(u)) = \boldsymbol{\max_f} \sum_{\boldsymbol{u \in V}} \left[-\boldsymbol{\log Z_u} + \sum_{\boldsymbol{n_i \in N_S(u)}} \boldsymbol{f(n_i) f(u)} \right]$$

(좌변) skip-gram 아키텍처를 네트워크로 확장.

- skip-gram이 중심단어로부터 주변단어가 나올 log-likelihood를 최대화하는 것
- node2vec 에서는 중심 노드로부터 이웃 노드들이 나올 log-likelihood 를 최대화 하는 것

(우변) 간소화된 objective function.

노드당 partition 함수인 Zu는 큰 네트워크에서 계산 비용이 많이 들고, negative samplin을 사용해서 이것을 계산함. Stochastic gradient ascen를 활용해서, 간소화된 objective func을 최적화한다.

Feature Learning framework

- Sampling strategies

1. BFS(Breath-first Sampling): 너비 우선 탐색

   Structural equivalence 관점 반영

   그림1에서 $N_S(u)$ = {s1, s2, s3} , k=3

2. DFS(Depth-first Sampling): 깊이 우선 탐색

   Homophily 관점 반영

   1에서 $N_S(u)$ = {s4, s5, s6}, k=3

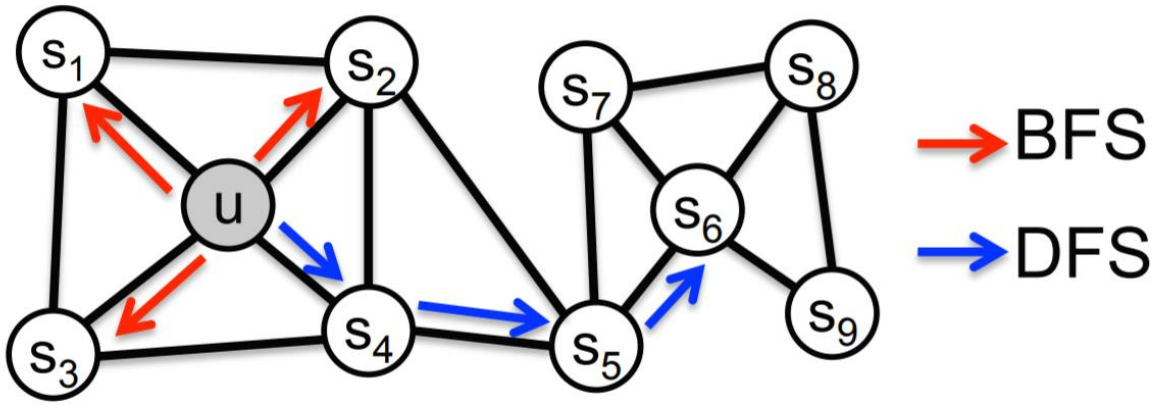


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

→ But real-world에서는 두 개의 동등성 개념이 배타적X

→ 본 논문에서는, 유연한 neighbor sampling strategies를 설계 = node2vec

[*Node2vec*]

node2vec

- 두 가지 원리를 준수하는 유연한 알고리즘
- 네트워크에서 확장가능한 feature 학습을 위한 semi-supervised 알고리즘

node2vec

- NLP representation 학습에의 발전 → 이산적인 객체들의 feature 학습이 가능해짐
- Skip-gram 모델: likelihood 목적을 보존하기위해, neigborhood를 최적화함으로써 단어의 연속적인 feature 표현을 학습하는 알고리즘.

  목표: 유사한 맥락의 단어들이 유사한 의미를 갖는 경향이 있다는 것을 가정. 즉, 비슷한 단어의 neighbor 에서 비슷한 단어가 나타나는 경향.

- 최근 연구는 네트워크를 "문서"로 표현 → 네트워크의 유추를 확립

네트워크로부터 노드 순서를 샘플링하여 네트워크를 순서가 정해진 노드로 바꿀 수 있음.

이때 샘플링 전략에 따라 결과가 많이 바뀌는데, 특정 샘플링 전략에 묶이지않고 paramete로 탐색공간을 유연하게 조절하는 node2vec을 사용.

node2vec

Random Walk

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

소스 노드 u가 주어졌을 때, 고정된 길이 ℓ 로 Random walk를 시뮬레이션.

- c_i: walk의 i-번째 노드
- π_{vx} : 노드v와 노드x간의 비정규화된 transition probability(전이확률: state가 v에서 x로 이동할 확률)
- Z: 정규화 상수

node2vec

Parameters

2^{nd} order randomwalk를 2개의 파라미터 p, q로 정의한다.

1. Return parameter, **p**
   - parameter p ↑, 새로운 노드로 가는 경향
   - parameter p ↓, 소스 노드에서 walk를 가깝게 유지

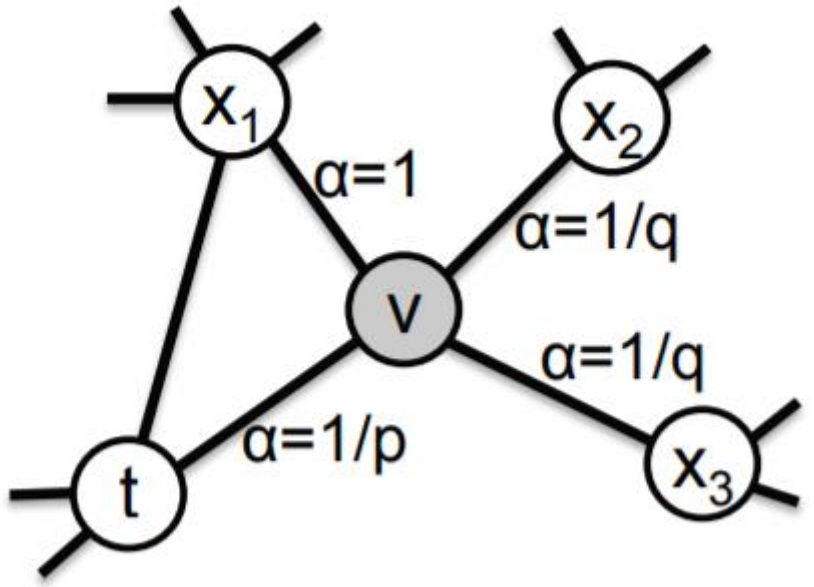
2. In-out parameter, **q**
   - q > 1, 시작 노드에 대한 그래프의 지역적인 관점 얻음(BFS 관점)
   - q < 1, 시작 노드에서 멀리 떨어진 노드를 방문하는 경향(DFS 관점)

node2vec

Parameters

t노드에서 2nd random walk진행

1. V에서 t로 전환
2. 다음 스텝으로는 t(내부 재방문), x1, x2, x3가능
   - d(t, t) = 0
   - d(t, x1) = 1
   - d(t, x2) = d(t, x3) = 2



역수로 계산되는 p가 작거나 q가 커질 경우, sampling이 근처 노드에서만 진행.
P가 크거나 q가 작아질 경우, 멀리 있는 노드로 뻗어 나가는 식으로 sampling 진행.

"p와 q의 설정에 따라 node2vec은 다른 결과를 반환"

node2vec

benefits of random walks.

1. 공간적 효율성

   : 그래프의 모든 노드의 직접적인 이웃들을 저장하는 **공간복잡도**는 O(|E|).
   2차 순서 랜덤워크 - O(a^2|V|)

2. 시간적 효율성
   : 샘플 생성 과정에서 그래프 연결을 부과함으로써, 랜덤 워크는 편리한 매커니즘을 제공.
   샘플들을 재사용 → 샘플링 속도를 효과적으로 높임.

node2vec

Node2vec algorithm

. node2vec의 세 가지 단계:

1. 전이확률을 계산하는 전처리

2.랜덤워크 시뮬레이션

3. SGD를 이용한 최적화.

: 순차적으로 수행됨.

각자 단계는 비동기적으로 병렬처리 및 실행되어 node2vec의 전체 확장성에 기여.

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)
π = 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}, π)
  Append s to $walk$
return $walk$

Introduction

- 풀고자 하는 문제

: homophily와 structural equivalence, 두 가지 원리를 모두 준수하는, 효율적이고 자동화된 feature vector representation algorithm 의 개발.

- 관련 연구들은 이 문제를 어떻게 풀어왔는지

: 차원 축소(성능 bad). BFS, DFS

- 이전 연구들과 달리, 어떻게 문제를 해결했는지

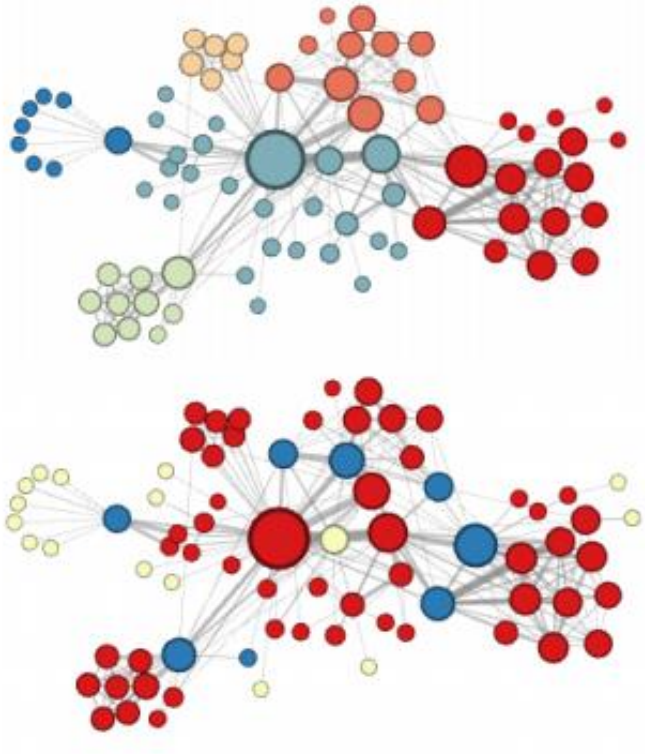
: 유연한 feature learning 알고리즘 사용, Node2vec

[*Experiments*]

Experiments

Case study 1.

- 레미제라블 소설에 '동시에' 등장하는 인물들에 대해 network를 모델링
  → node2vec 적용 → feature representation → k-means 클러스터링



Node: 캐릭터

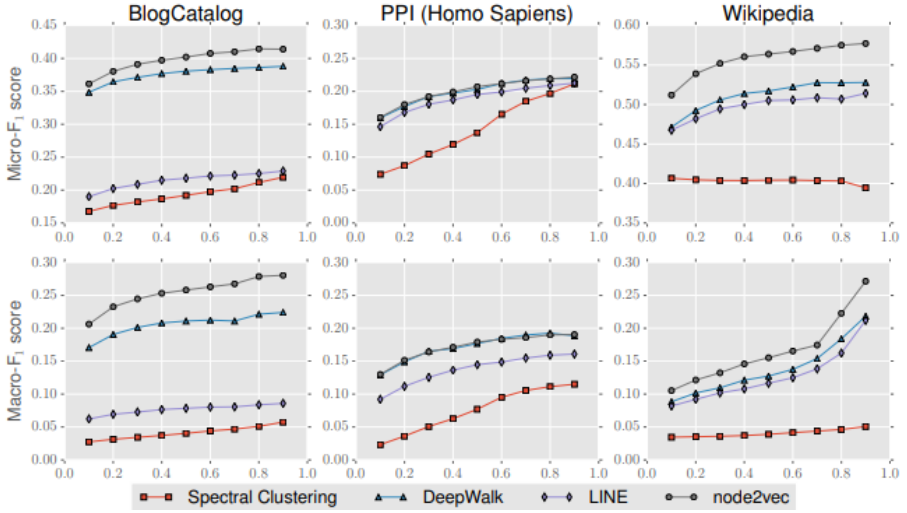
Edge: 같이 등장하면 edge로 노드 간 연결

1) p=1 q=0.5: homophily에 연관됨.
   즉, 같이 등장하는 캐릭터들끼리 즉, community 내부의 캐릭터들이 비슷하게 배정.
2) P=1 q=2: 노드의 구조적 역할을 발견할 수 있음.
   소설의 다른 plot에서 각각 맡은 역할이 비슷하면 비슷하게 배정.

Experiments

Multi-label classification

"spectral clustering, DeepWalk, LINE과 Node2vec을 multi-label classification 문제에 적용해보니, **Node2vec의 성능이 좋았음.**"



- BlogCatalog에서, train:test = 7:3인 경우 deepwalk보다 성능이 가장 크게 우수.
- PPI에서, deepwalk와 성능 차이 별로 없음. p=4(이미 방문한 노드의 재방문을 피함)로 두어, 조금 우위에 있을 뿐. LINE에 비해서는 상승률을 보임.

Experiments

Link prediction

- Setting for Link prediction : 네트워크의 connectivity를 유지한 상태로 50%의 edge를 cut.

- 데이터:

- Facebook: node-사용자, edge-친구여부
- PPI: node- 단백질, edge-생물학적 상호작용
- arXiv: node- 과학자, edge-협업 논문 여부

- 실험결과: **Node2vec best.**
  :모든 네트워크에서 Deepwalk와 LINE을 능가
  Hadamard 연산자 사용시, 가장 안정성이 뛰어남

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

Table 4: Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (See Table 1 for definitions).

Reference

- 논문 원본, https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf
- 논문 리뷰 블로그 1, https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef
- 논문 리뷰 블로그 2, https://frhyme.github.io/machine-learning/node2vec/#hyper-parameter-p-and-q
- 논문 리뷰 블로그 3, https://analysisbugs.tistory.com/m/214

$$[\ Q\ n\ A\ ]$$