그래프 분석

전종준[1]

September 6, 2022

[1]Dept. of Statistics,
University of Seoul,
S. Korea

목차

7-1. OD데이터의 처리

목차

목차

7-1. OD데이터의 처리

(1)한국스마트카드 교통카드거래실적 데이터 수집

- 서울교통통계
  - 한국스마트카드(KSCC)사의 교통카드 거래실적 데이터를 활용
  - 서울시 대중교통 버스노선별, 동별, 지하철 O/D 통계자료 제공

7-1. OD데이터의 처리

(1) 한국스마트카드 교통카드거래실적 데이터 수집

- 서울시 대중교통에서 버스, 지하철 데이터의 수집 방법
    - 버스 : 서울시 관할 운수사의 버스 노선
    - 지하철 : 서울교통공사 운영노선(1~9호선)에서 승차 또는 하차가 발생한 통행
    - 집계 case : 분당선(승차) → 1호선(하차), 1호선(승차) → 에버라인선(하차)
    - 미집계 case : 분당선(승차) → 신분당선(하차), 분당선(승차) → 분당선(하차)

7-1. OD데이터의 처리

(1) 한국스마트카드 교통카드거래실적 데이터 수집

- 대중교통 정보를 어떻게 서울시가 관리할 수 있을까?

(a) (b)

Figure 1: 한국스마트카드 (링크)와 준공영제 (링크)

7-1. OD데이터의 처리

(1) 한국스마트카드 교통카드거래실적 데이터 수집

- 데이터 수집 및 필드값 해석
- 데이터 : 노선별 OD 2021년 3월 16일

(a)

	A	B	C	D	E	F	G	H	I
1	기준일자	노선명	승차_정류장ARS	승차_정류장명	하차_정류장ARS	하차_정류장명	승차_정류장순번	하차_정류장순번	승객수
2	20210316	17	3689	청암자이아파트	3250	용산전자상가입구	1	10	1
3	20210316	17	3689	청암자이아파트	3252	신용산.지하차도	1	11	2
4	20210316	17	3689	청암자이아파트	3255	용산역	1	12	8
5	20210316	17	3689	청암자이아파트	3310	남이장군사당	1	7	2
6	20210316	17	3689	청암자이아파트	3312	새마을금고	1	8	1

(2) 노선별 OD데이터의 전처리

- 날짜 형식 데이터 처리

```
import pandas as pd

df['기준일자']=df['기준일자'].astype(str) #object로 변환
pd.to_datetime(df['기준일자'])# object를 Datetime형태로 변환
df['기준일자']=pd.to_datetime(df['기준일자'])
df['기준일자'].dt.year
df['기준일자'].dt.month
df['기준일자'].dt.day
df['기준일자'].dt.day_name()
```

	기준일자	노선명	승차_정류장ARS	승차_정류장명	하차_정류장ARS	하차_정류장명	승차_정류장순번	하차_정류장순번	승객수	날짜_date	year	month	day	day_name
0	20210316	0017	3689.0	청암자이아파트	3250.0	용산전자상가입구	1.0	10.0	1	2021-03-16	2021	3	16	Tuesday
1	20210316	0017	3689.0	청암자이아파트	3252.0	신용산.지하차도	1.0	11.0	2	2021-03-16	2021	3	16	Tuesday
2	20210316	0017	3689.0	청암자이아파트	3255.0	용산역	1.0	12.0	8	2021-03-16	2021	3	16	Tuesday

(2) 노선별 OD데이터의 전처리

- 노선명이 없는 데이터 처리

```
df.isnull().sum()
df[df['하차_정류장명'].isnull()]
df.shape #(490432, 14)
df=df.dropna(subset=['하차_정류장명'],axis=0)
df.shape #(490393, 14)
```

```
df.isnull().sum() ...
기준일자          0
노선명          163
승차_정류장ARS   983
승차_정류장명      0
하차_정류장ARS   161
하차_정류장명     39
승차_정류장순번  1007
하차_정류장순번   174
승객수            0
날짜_date         0
year              0
month             0
day               0
day_name          0
```

(2) 노선별 OD데이터의 전처리

- 정류장의 좌표정보받기
- 서울 열린데이터 광장 '서울특별시 버스정류소 위치정보' (링크)

파일내려받기

* 파일에 이상이 있는 경우 '오류신고'를 통해 운영자에게 알려주세요. 오류신고

NO	항목	파일명	용량(MB)	수정일	내려받기
1	데이터	서울시버스정류소좌표데이터(2021.01.14.).csv	0.6	2021.01.15	

(a)

	A	B	C	D	E
1	표준ID	ARS-ID	정류소명	X좌표	Y좌표
2	100000001	1001	종로2가사거리	126.9877862	37.5697641
3	100000002	1002	창경궁.서울대학교병원	126.9965202	37.5791788
4	100000003	1003	명륜3가.성대입구	126.9982902	37.5827088
5	100000004	1004	종로2가.삼일교	126.9875072	37.5685822

(b)

(2) 노선별 OD데이터의 전처리

- 데이터 병합

```
df1=pd.merge(df,location, how='left',
            left_on=['승차_정류장ARS'],right_on=['ARS-ID'])
df_loc=pd.merge(df1,location, how='left',
            left_on=['하차_정류장ARS'],right_on=['ARS-ID'])
df_loc #정류소명_x : 승차, 정류소명_y: 하차
```

- 노선의 수는?

```
len(df['노선명'].unique()) #621
```

- 정류장의 수는?

(3) 시점별, 동별 이동분석

- 특정 노선시각화
- 노선명이 0017을 대상

```
import folium
line_17=df_loc[df_loc['노선명']== '0017']
m = folium.Map(location=[line_17['Y좌표_x'][0],line_17['X좌표_x'][0]], zoom_start=14)

for name, lat,lng in zip(line_17['승차_정류장명'],line_17['Y좌표_x'],line_17['X좌표_x']):
    sub_m1=folium.Marker([lat, lng],
                         popup=name,
                         icon=(folium.Icon(color='green',icon='bus',prefix='fa')))
    sub_m1.add_to(m)

folium.LayerControl().add_to(m)
m
```

(3) 시점별, 동별 이동분석

- 특정 노선시각화
- 노선명이 0017을 대상

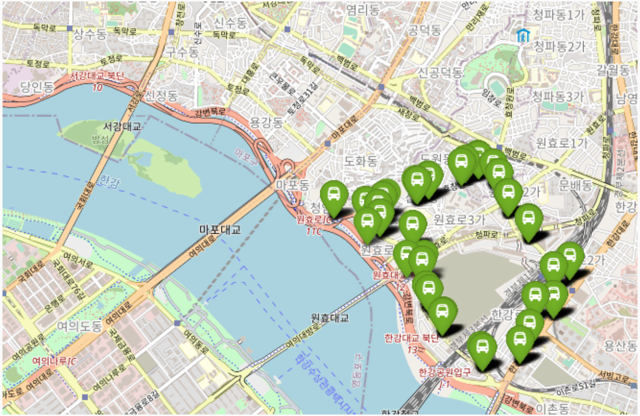


Figure 6. 특정 노선 시각

(3) 시점별, 동별 이동분석

- 서울시교통정보월별대중교통O/D현황
- 데이터: 동별 수단 2021년 3월 16일

```
Data = pd.read_csv("동별_수단OD_20210316.csv", encoding='cp949')
```

	A	B	C	D	E	F	G	H
1	기준일자	출발_구	출발_동	도착_구	도착_동	총_승객수	지하철_승객수	버스_승객수
2	20200201	가평군	가평읍	가평군	가평읍	3	3	0
3	20200201	가평군	가평읍	강남구	논현1동	4	4	0
4	20200201	가평군	가평읍	강남구	논현2동	5	5	0
5	20200201	가평군	가평읍	강남구	대치2동	2	2	0

Figure 7: 동별 수단 데이터

(3) 시점별, 동별 이동분석

- 출발구와 도착구는 어디 지역이 있을지, 혼잡할지 분석해보자

```
print(Data['출발_구'].unique())  #(a)
print(Data['도착_구'].unique())  #(b)

# 출발구에 따른 총승객수
Data['총_승객수'].groupby(Data['출발_구']).describe().head()
Data['총_승객수'].groupby(Data['도착_구']).describe().head()
#지하철 승객수
Data.sort_values(by = ['지하철_승객수'])
Data.sort_values(by = ['지하철_승객수'], ascending = False) #내림차순
```

(1) 그래프 데이터의 구조

- 그래프(Graph): 단순히 노드(N, node)와 그 노드를 연결하는 간선(E, edge)을 하나로 모아 놓은 자료 구조
- 즉, 연결되어 있는 객체 간의 관계를 표현할 수 있는 자료구조
- 각 node마다 고유의 정보들이 있고, edge는 node 간의 연결관계의 강도 및 방향성 등에 대한 정보
  Ex) 지도, 지하철 노선도의 최단 경로, 전기 회로의 소자들, 도로(교차점과 일방 통행길), 선수 과목 등

(1) 그래프 데이터의 구조

- 노드(node) 또는정점(vertex): 위치라는 개념.
- 엣지(edge): 위치 간의 관계. 즉, 노드를 연결하는 선 (link, branch 라고도 부름)
- 엣지가중치 : 그래프의 모든 엣지가 가중치(weight)를 가지고 있는 가중치 그래프에서 사용

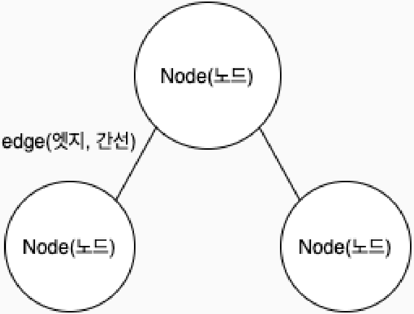


Figure 8: graph

(1) 그래프 데이터의 구조

- 특별한 그래프 (이진 트리)
    - DAG(Directed Acyclic Graph) 방향성이 있는 비순환 그래프의 한 종류
    - 각각의 노드가 최대 두 개의 자식 노드를 가지는 트리 자료 구조

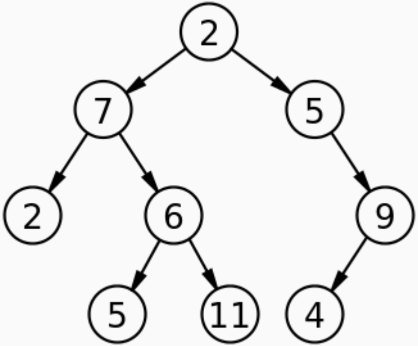
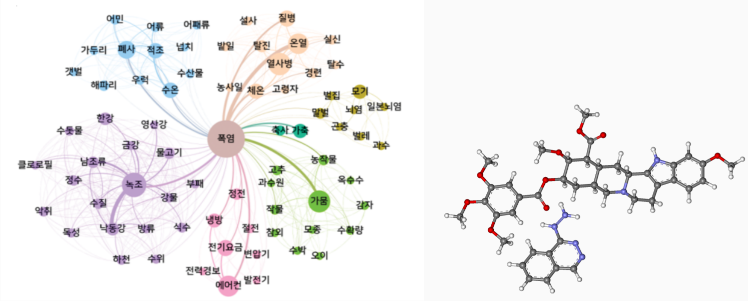


Figure 9: binary tree

(1) 그래프 데이터의 구조

- 그래프 구조의 활용 예
  - (a) 문장 데이터를 그래프로 표현
  - (b) 분자 데이터를 그래프로 표현

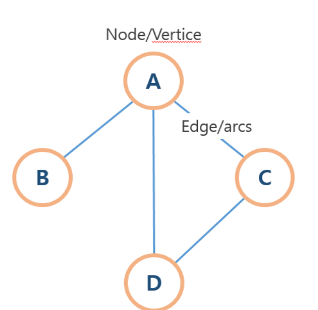


(a) (b)

Figure 10: 그래프 구조 예시

(2) 그래프 데이터의 요약통계량 1

- Degree (Node의 연결정도, "차수" 라고도 함, 정수로 표현)
  - 차수(degree): 무방향 그래프에서 하나의 정점에 인접한 정점의 수
  - 그래프에서 A의 차수는 3
  - 진출차수(out-degree)/진입차수(in-degree) : 방향그래프에서 사용되는 용어
  - 진출 차수(out-degree): 한 노드에서 외부로 향하는 간선의 수
  - 진입차수(in-degree): 외부 노드에서 들어오는 간선의 수



(2) 그래프 데이터의 요약통계량 1

- Centrality(연결 중심성, Node의 연결정도, 활동성, 비율)
    - 누가 이 네트워크 내에서 중요한 사람인가?
    - Network에서 가장 중요한 vertices를 찾는 측도임
    - Network의 vertices의 실함수로 정의됨
    - 다양한 종류의 Centrality가 있음
        1. Degree Centrality
        2. Betweenness Centrality
        3. Closeness Centrality
        4. Eigenvector Centrality
           : Network에서 vertex의 영향력을 측정하는 측도로 이용됨.
           높은 점수를 가진 vertex와의 연결이 영향력를 측정하는데 중요하게 사용됨
        5. Katz Centrality
           :직접적인 edge의 개수뿐만 아니라 연결된 모든 vertex의 개수를 계산함
           Degree Centrality와 Eigenvector Centrality의 일반화된 개념
        6. PageRank Centrality
           :Google Search에서 웹 사이트의 순위를 매기기 위해 사용됨
           Eigenvector Centrality를 더 확장한 개념

(2) 그래프 데이터의 요약통계량 1

1. Degree Centrality
   - Degree는 Network에서 vertex에 연결되어 있는 edge의 개수로 정의됨
   - vertex v에 대해서 $C(v) = deg(v)$로 표기함
2. Betweenness Centrality
   - Network에서 최단 경로의 개념을 사용함
   - 모든 두 개의 vertex에 대해서 두 vertex를 잇는 최단경로가 존재함
   - 한 vertex를 지나는 최단경로의 개수를 Betweenness Centrality로 정의함
   - 수학적 정의
     - vertex v에 대해서 σ_{st}를 s에서 t 사이의 모든 최단경로의 개수로 정의함
     - $\sigma_{st}(v)$를 그 중 v를 지나는 최단경로의 개수로 정의함
     - Betweenness Centrality $C(v) = \sum_{s \neq v \neq t} \dfrac{\sigma_{st}(v)}{\sigma_{st}}$

(2) 그래프 데이터의 요약통계량 1

3. Closeness Centrality
    - Networ에서 최단 경로의 개념을 사용함
    - vertex v에 대해서 다른 모든 vertex와의 평균 최단 경로의 길이를 계산함
    - 수학적 정의
        - vertex y,v에 $d(y,v)$대해서 를 y와 v사이의 거리로 정의함
        - Closeness Centrality $C(v) = \frac{1}{\sum_y d(y,v)}$
        - 혹은 $H(v) = \sum_{y \neq v} \frac{1}{d(y,v)}$로 정의하기도 함

(3) 그래프 데이터의 요약통계량 2

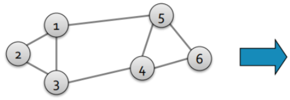
- Network Cohesion
- 결합력(Cohesion)은 social network를 구성하는 점(node)들간의 강한(strong) 연결관계를 나타냄
- 네트워크 응집력은 다양한 방법으로 측정될 수 있으며, 대부분은 Dyadic cohesion을 기반
- Clique cohesion
  - 네트워크(network)를 구성하는 점(node)들간의 결합력(Cohesion)을 바탕으로 해서 군집 구조를 파악하는 것.
  - Clique은 결합력을 가지는 최소 3개의 점(node)으로 구성되는 그룹(group)을 나타내며 모든 점(node)이 직접적으로 연결되어 있어야만 Clique이 성립
  - 그러므로 Clique은 정의상 완벽한 연관관계와 높은 밀도를 가짐

(3) 그래프 데이터의 요약통계량 2

- Laplacian matrix와 그래프의 spectral analysis
- $L = D - A$라고 정의를 하는데 D는 Degree Matrix, A는 Adjacency matrix를 나타냄.
- 이 matrix는
  1) symmetric
  2) n개(node의 개수)의 eigen values를 가짐
  3) 모든 eigenvetor가 real하고 Orthogonal

(3) 그래프 데이터의 요약통계량 2

- Laplacian matrix example



	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

(a)

	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

(b)

	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

(c)

Figure 12: (a)Adjacency matrix, (b) Degree matrix, (c) Laplacian matrix

(3) 그래프 데이터의 요약통계량 2

- for symmetric matrix M:

$$\lambda_2 = min_{x:x^T W_1 = 0} \frac{x^T M x}{x^T x}$$

- M자리에 L(Laplacian Matrix) 넣으면 변형

$$\lambda_2 = min \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2}$$

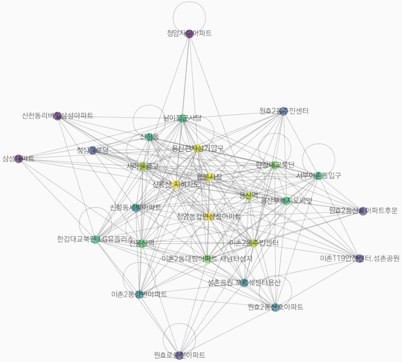
(4) 교통 OD데이터를 이용한 실습

- 데이터: 노선별 OD 2021년 3월 16일
- 네트워크 방법 1: 데이터프레임형식에서
- Graph with 27 nodes and 230 edges

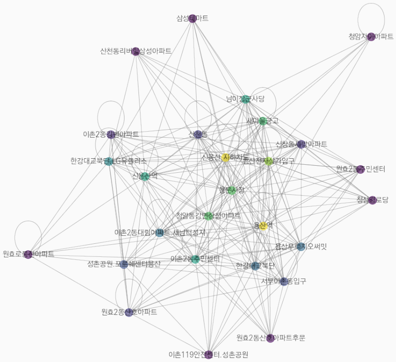
```
g = nx.Graph()
g = nx.from_pandas_edgelist(line_17,
                source = '정류소명_x', target = '정류소명_y')
print(nx.info(g))
pr=nx.degree_centrality(g)
#nx.betweenness_centrality(g)
nx.draw_networkx(g,font_family='KoPubDotum',font_size=16,
                node_color=list(pr.values()), alpha=0.6,
                edge_color='.5')
```

(4) 교통 OD데이터를 이용한 실습

- centrality에 따라 노드 색 다르게 나타남



(a)



(b)

(4) 교통 OD데이터를 이용한 실습

- 네트워크 방법 2 : node와 edge 지정
- Graph with 27 nodes and 230 edges

```
g = nx.Graph()
g.add_nodes_from(nodes['정류소명'])  #노드추가

g.add_edges_from(link_node)
print(nx.info(g))

pr=nx.degree_centrality(g)

nx.draw_networkx(g,font_family='KoPubDotum',font_size=16,
                node_color=list(pr.values()),  alpha=0.6,
                edge_color='.5' )
```

(4) 교통 OD데이터를 이용한 실습

- 데이터 : 동별수단 2021년 3월 16일
- 가장 혼잡한 곳은 어디일까?
- 네트워크 내에서 노드의 중요도를 측정하는 방법을 조사해보자.
- 시간에 따라 네트워크 노드의 중요도가 달라짐을 확인해보자.
- 지도상에서 일별 혼잡도를 시각화하고자 한다. 필요한 데이터 구조에 대해서 논의해보자.

(1) 랜덤 그래프 이론: classical random graph models

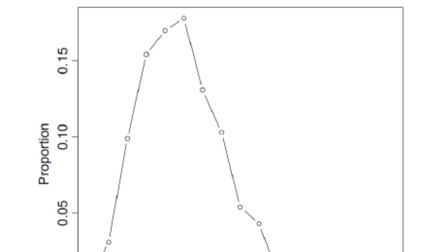
- 네트워크 자료의 모형중 하나
- $g := g_{N_v,N_e} : |V| = N_v, |E| = N_e$ 인 모든 그래프 모임
- 이 때, 임의의 $G \in g$에 대하여 $P(G) = \binom{N}{N_e}^{-1}$
  (단, $N = \binom{N_v}{2}$: N_v개의 노드 사이에 연결될 수 있는 가능한 edge의 총 갯수)
- 반면, 각 edge 별 발현 확률에 대한 모수를 부여하여 모형을 만들 수 있음
- $g := g_{N_v} : |V| = N_v$ 인 모든 그래프 모임
- 이 위에서, 임의의 edge가 연결될 확률을 p로 하는 모형을 만들면 됨
- 첫번째 모형은 node의 수와 edge의 수가 주어지면 확률 모형이 결정되고, 두번째 모형은 node의 수와 각 edge의 발현 확률이 주어지면 확률 모형이 결정
- 즉, 두 모형을 각각 $G(N_v, N_e), G(N_v, p)$와 같이 표현
- 첫번째 모형은 edge의 수도 통제되므로 활용성이 떨어지는 반면, 두번째 모형이 훨씬 사용하기 쉽고 다양한 그래프를 포함

(1) 랜덤 그래프 이론: classical random graph models

- properties
- 노드의 수 N_v가 충분히 크면, degree의 분포는 모수가 $(N_v-1)p$인 포아송 분포와 비슷한 양상

cf) 포아송 분포: 모수 λ 에 대하여, 확률변수 $X \in \{0\} \cup \mathbb{N}$가 k일 확률이 아래와 같이 표현되는 분포

$$p(X=k) = \frac{\lambda^k e^{-\lambda}}{k!}$$



(1) 랜덤 그래프 이론:classical random graph models

- 또한, p가 아주 작지 않으면 생성되는 그래프들의 component가 대부분 1개라는 사실이 알려져 있음
- 앞 페이지의 결과와 결합하여 해석해보면, 평균 degree가 꽤 작아도 전체 그래프는 하나의 component로 연결되어있다는 뜻

cf) 10000개의 노드가 있을 때 평균 degree가 10개 이상만 되어도 대부분의 랜덤 그래프는 component가 1개

- 마지막으로, 랜덤 그래프 모형을 통해 생성되는 그래프의 diameter 또한 그래프의 사이즈에 비해 비교적 작다는 특징: small-world property

(1) 랜덤 그래프 이론: generalized random graph model

- 첫번째 classical random graph model에서는 node의 수 N_v와 edge의 수 N_e를 통해 모형 구축
- 반면, 각 노드의 degree 값을 모두 고정시켜놓고 그안에서 균일한 확률을 갖는 랜덤 그래프 모형을 만들 수 있음
- 각 노드의 degree를 $d_1, ..., d_{N_v}$라 하면, N_e는 따로 정해주지 않아도 자동으로 결정
    - $\bar{d} = 2N_e/N_v$
- 즉, 위의 모형은 앞에서 소개한 두가지 classical random graph model 보다 모형의 공간이 작음 (제약조건이 많아졌기때문)

(2) 네트워크 성장이론

- 앞서 배운 random graph model은 degree의 분포가 두터운 꼬리를 가짐(heavy-tailed)
- degree의 분포가 근사적으로 포아송 분포를 따름
- Scale-Free Models (Network Growth Model)
  - 반면, 여러 네트워크 데이터는 degree의 분포가 얇은 꼬리를 가짐

  ex. 웹페이지 데이터 : 포탈 사이트 등의 소수 웹페이지는 매우 많은 웹페이지와 연결되어있지만, 대부분의 나머지 웹사이트들은 서로 연결되어있지 않음

  ex. SNS 데이터 : 대부분의 일반인들은 몇백명 수준의 팔로워를 가지지만 소수의 유명인사들은 매우 높은 팔로워 수를 가짐

(2) 네트워크 성장이론

- 대부분의 노드는 degree가 매우 높은 hub node를 통해 연결관계를 가짐
- 이러한 자료에 맞는 네트워크 모형이 필요 (연속형 자료에서 t-분포를 사용하는 이유를 생각)

(2) 네트워크 성장이론

- The Barabasi-Albert model
- Pregerential Attachment: 네트워크가 점점 확장될수록, 새로 진입하는 노드는 이미 많은 노드들과 연결되어 있는 hub node에 연결될 가능성이 높음
- rich-get-richer 현상
- 그래프 생성 모형
  1. 초기 그래프 $G^{(0)}$: $N_V^{(0)}$ 개의 node, $N_e^{(0)}$ 개의 edge
  2. t-1 시점의 그래프 $G^{(t-1)}$ 가 주어졌을 때 다음 시점 t의 그래프 $G^{(t)}$ 는 degree가 m인 새로운 노드를 추가하여 생성
  3. 새로운 노드의 degree가 m이므로 기존네트워크 $G^{(t-1)}$ 에서 m개의 노드를 선택하여 연결. 이때, $G^{(t-1)}$ 의 각 노드의 degree의 값이 클수록 선택될 확률이 높도록 설정

(2) 네트워크 성장이론

- The Barabasi-Albert model
- 위 모형을 통해 생성되는 t시점의 그래프 $G^{(t)}$는 $N_v^{(0)} + t$개의 node와 $N_e^{(0)} + tm$개의 edge를 가짐
- 새로 진입하는 노드는 기존 네트워크에서 높은 degree를 갖는 노드와 연결될 확률이 높으므로 그래프의 사이즈가 커질수록 자연스럽게 hub node가 생성

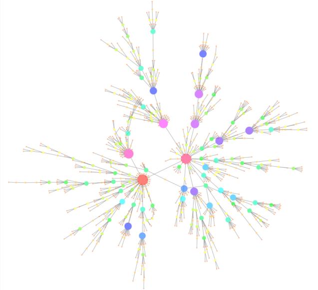
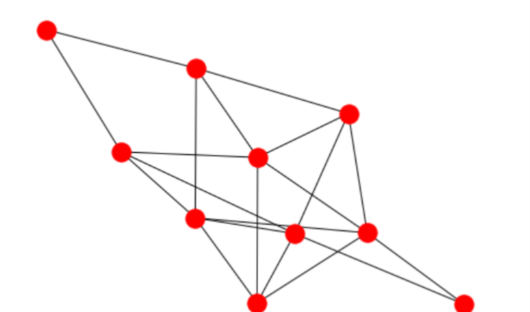


Figure 14: The Barabasi-Albert model

(3) 그래프 데이터 적합 실습

- Random Graph Models
- Erdos Renyi Graph

```
import networkx as nx
n = 10  # 10 nodes
m = 20  # 20 edges
G = nx.gnm_random_graph(n, m, seed=seed)
```



7-3. 랜덤 그래프

(3) 그래프 데이터 적합 실습

- Erdos Renyi Graph

```python
# some properties
for v in nx.nodes(G): #node degree clustering
    print(f"{v} {nx.degree(G, v)} {nx.clustering(G, v)}")
for line in nx.generate_adjlist(G): #the adjacency list
    print (some properties)
```

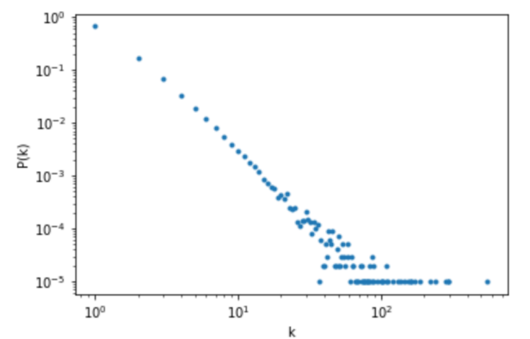
```
node degree clustering
0 4 0.3333333333333333
1 5 0.3
2 4 0.16666666666666666
3 4 0.5
4 4 0.16666666666666666
5 2 0
6 2 0
7 5 0.3
```

```
the adjacency list
0 8 2 9 1
1 2 4 9 3
2 7 6
3 9 8 7
4 7 6 8
5 8 9
6
7 9 8
```

(3) 그래프 데이터 적합 실습

- Network Growth Models
- The Barabasi-Albert model
    - degree(k) 별 node개수가 거듭제곱 분포

```
G = nx.barabasi_albert_graph(n=100000, m=1, seed=5)
k = dict(nx.degree(G))
```



(3) 그래프 데이터 적합 실습

- The Barabasi-Albert model

(a) (b) (c)

Figure 18: BA model 출력 결과