



SESSION # 12

네트워크 분석

By Team 2

@ 남정우, 신윤기

Date _ 2018. 01.09

CONTENTS

- 01 Network?
- 02 네트워크의 지표
- 03 중심성의 이론적 배경과 계산 알고리즘
- 04 네트워크 활용 사례
- 05 NetworkX와 Gephi

01 Network

네트워크 이론

- 수학의 그래프 이론에 따라, **연결 구조와 연결 강도** 등을 바탕으로 사용자의 영향력을 측정하는 방법.
- 사람, 그룹, 조직, 컴퓨터 및 데이터 등 **객체간의 관계** 및 **네트워크 특성과 구조를 분석**하고 시각화하는 분석 방법론
- 범죄 수사, 첩보, 조직 분석, 커뮤니케이션망 분석, 에이즈 확산 연구, 제약 연구 등의 분야에 활발하게 응용

01 Network

네트워크란?

- 다수의 점과 점들을 연결하는 다수의 선으로 구성된 망.
노드(Node), 엣지(Edge)로 구성됨
- 노드 (Node): 개체(사람, 조직, 사물 등) – 점으로 표시
- 엣지 (Edge): 개체들 간의 관계 – 선으로 연결

01 Network

네트워크의 종류

- **방향 네트워크 (directed network):** 정보전달, 국가간의 수출/수입 등 방향성이 있는 경우로, 송신자와 수신자가 확실함.
- **무방향 네트워크 (undirected network):** 외교관계, 혈연관계 등 액터 관계의 존재 자체를 문제로 하는 경우로, 일반적으로 방향성이 없는 관계임.

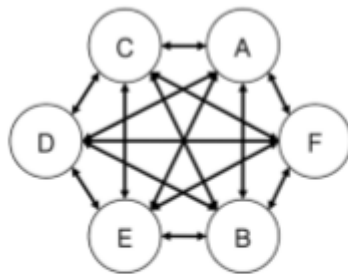
01 Network

네트워크의 종류

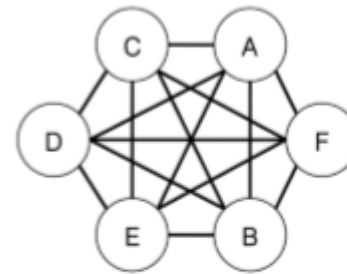
- **완전 그래프 (complete graph):** 각 정점에서 다른 모든 정점을 연결하여 가능한 최대의 연결선을 가진 그래프

정점이 n 개인 방향 그래프에서 최대 엣지의 개수: $n(n-1)$ 개

정점이 n 개인 무 방향 그래프에서 최대 엣지의 개수: $n(n-1)/2$ 개



(a) 방향 그래프
: 30개 엣지



(b) 무방향 그래프
: 15개 엣지

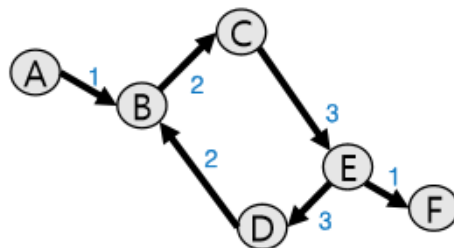
01 Network

네트워크의 종류

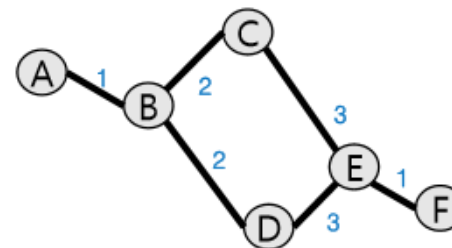
- 가중 그래프 (weighted graph): 정점을 연결하는 엣지에 가중치를 할당한 그래프

가중치는 엣지의 정도 차이를 나타냄

친구 관계를 네트워크 그래프로 나타내는 경우, 단순한 친구관계 혹은 친분이 두터운 친구 관계에 따른 관계 정도가 가중치로 나타남



(a) 방향 그래프



(b) 무방향 그래프

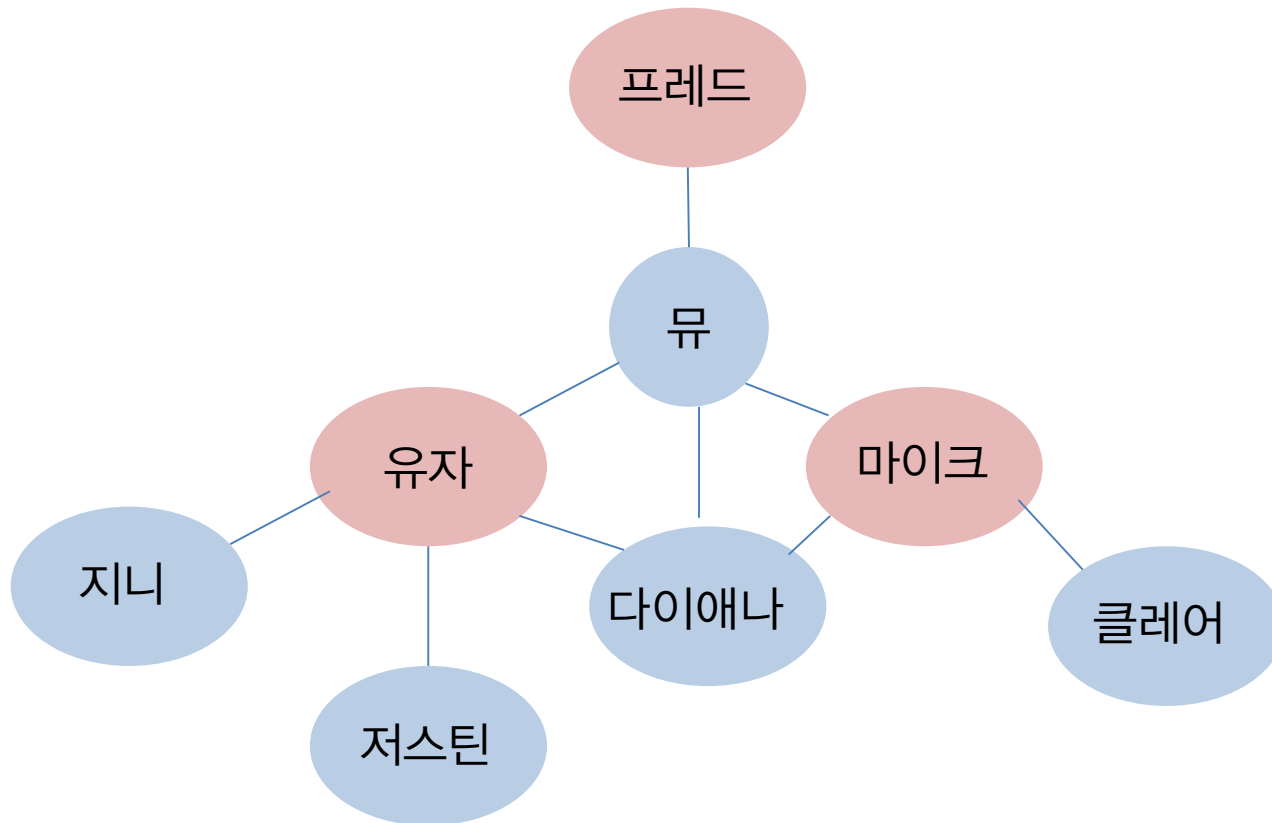
01 Network

네트워크의 예시

- 저스틴과 유자는 친구이다.
- 유자는 지니와 친구이고 다이애나와도 친구이다.
- 다이애나는 뮤와 마이크와 친구이다.
- 뮤는 유자와 친구이고 프레드, 마이크와도 친구이다.
- 마이크와 클레어는 친구이다.

01 Network

네트워크의 예시



01 Network

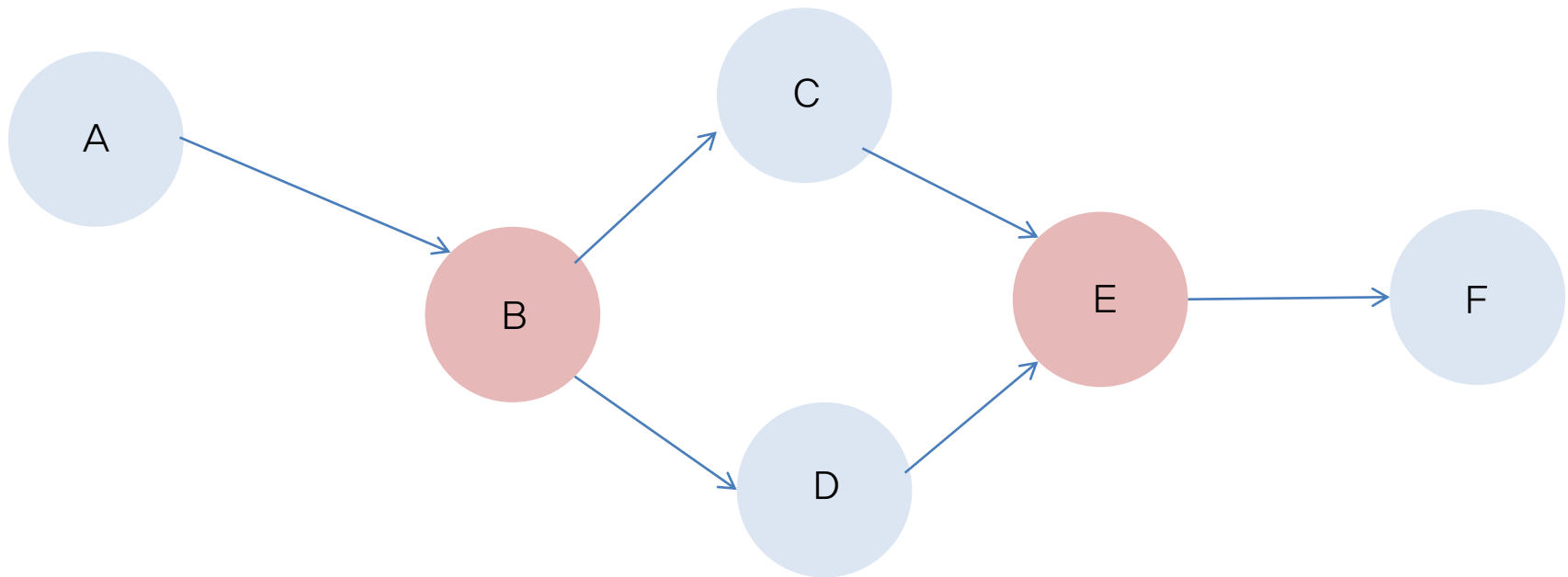
네트워크의 예시

- 그래프 G란 개체를 나타내는 정점(vertex) V와 개체를 연결하는 엣지(edge) E의 집합
- $G = (V, E)$
- $V = \{ A, B, C, D, E, F \}$
V : 노드(Node)
- $E = \{ (A, B), (B, C), (C, E), (E, D), (D, B), (E, F) \}$
E : 엣지(Edge)

01 Network

네트워크의 예시

- $G = (V, E)$



02 네트워크의 지표

■ 그래프의 특징을 나타내는 지표

- 차수(degree)와 허브(hub)
- 차수의 분포 (degree distribution)
- 밀도 (density)
- 중심성 (centrality)
 - 연결 정도 중심성 (degree centrality)
 - 근접 중심성 (closeness centrality)
 - 매개 중심성 (betweenness centrality)

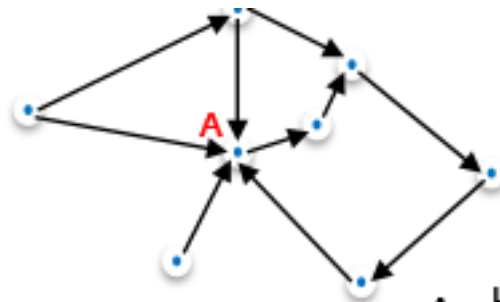
02 네트워크의 지표

차수(degree)의 정의

- 노드에 연결된 엣지들의 수로 해당 노드가 다른 노드들과 얼마나 많이 연결되어 있는가에 대한 측정지표
- 방향 그래프의 경우, 진입차수(indegree)와 진출차수(outdegree)로 구분
 - 진입차수(indegree) : 해당 노드에 들어오는 엣지들의 수
 - 진출차수(outdegree) : 해당 노드에서 나가는 엣지들의 수

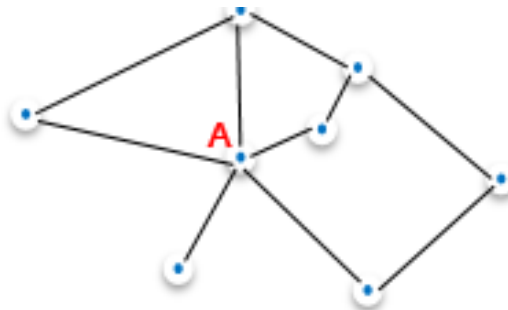
02 네트워크의 지표

차수(degree)의 정의



- 노드 A의 진입차수: $k_{in} = 4$
- 노드 A의 진출차수: $k_{out} = 1$

(a) 방향 그래프



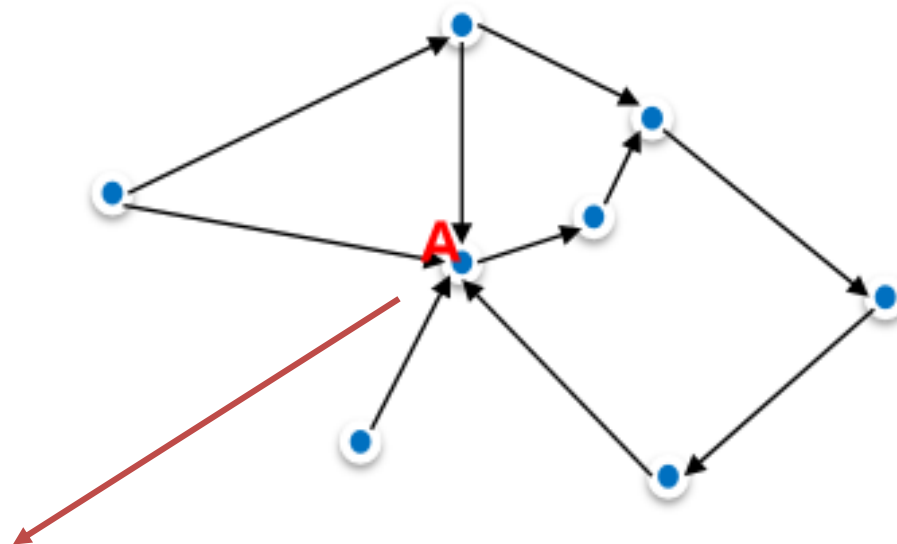
- 노드 A의 차수: $k = 5$

(b) 무방향 그래프

02 네트워크의 지표

허브(hub)의 정의

- 노드 중에서 가장 높은 차수를 가지고 있는 노드



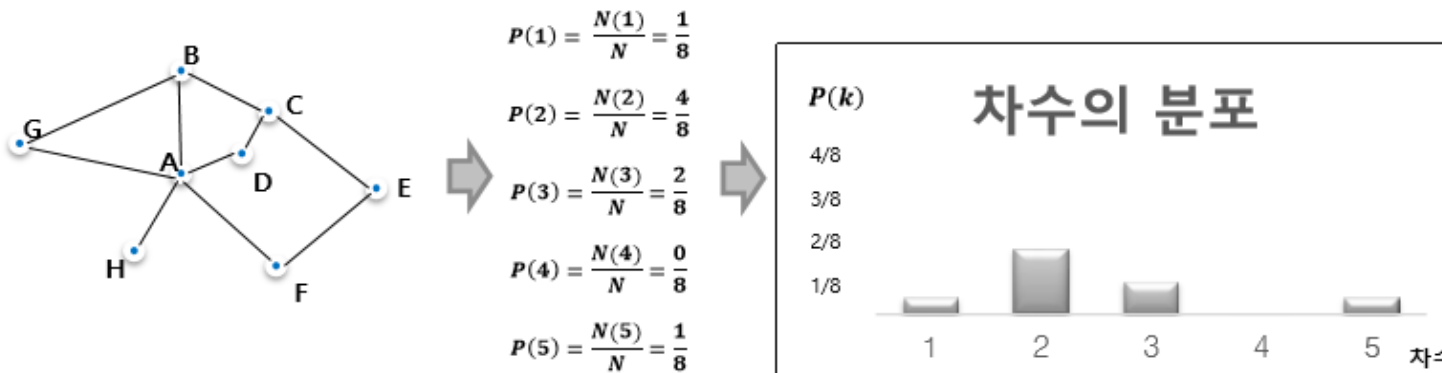
Hub: 노드 A

02 네트워크의 지표

차수의 분포 $P(k)$

- 그래프에서 차수 K 를 갖는 노드의 비율을 의미
- 차수 k 를 갖는 노드의 수를 전체 노드 수 N 으로 나눈 값

$$P(k) = \frac{N(k)}{N}, \quad k = 1, 2, \dots, n$$

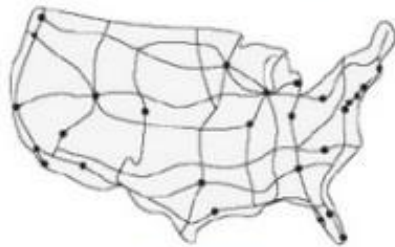
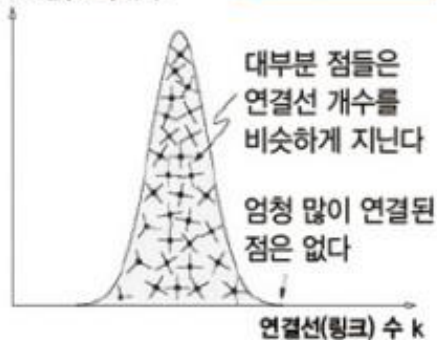


무작위 네트워크와 Scale-Free 네트워크

네트워크의 두 가지 유형

연결선 k 개를
지나는 점(노드)의 수

가우시안 분포



고속도로망

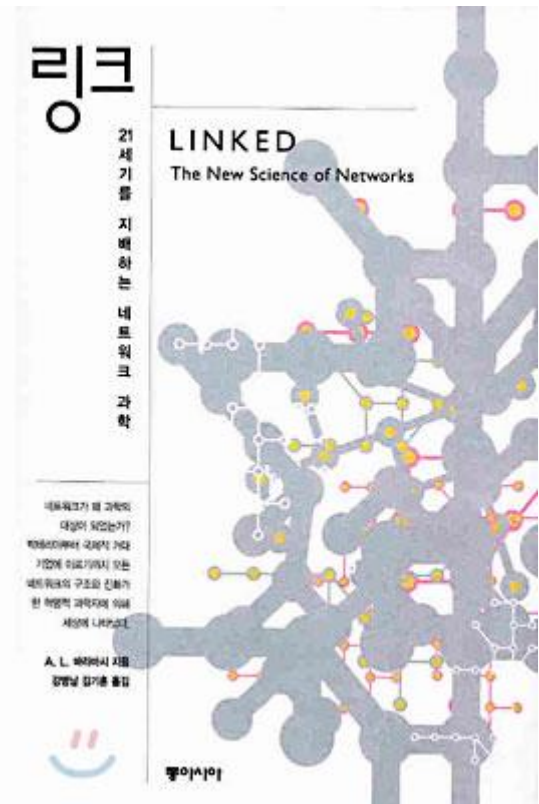
연결선 k 개를
지나는 점(노드)의 수

멱함수 분포



항공망

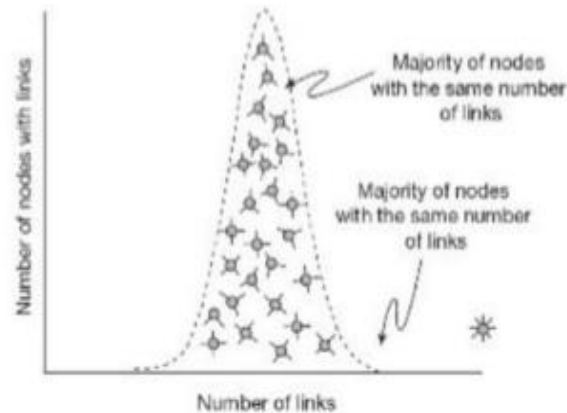
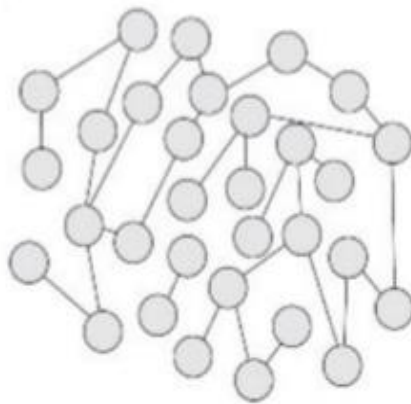
멱함수(오른쪽)는 네트워크에서 얼마나 많은 점들이 몇 개의 연결선으로 연결돼 있는지를 보여주는 연결선 분포 함수다. 우리에게 익숙한, 평균 주변에 많이 모여 있는 종 모양의 '가우시안 분포'(왼쪽)와 달리, 멱함수 분포는 많은 연결선을 지닌 허브를 비롯해 다양성이 존재함을 보여준다.



02 네트워크의 지표

무작위 네트워크(Random Graph, 랜덤 그래프)

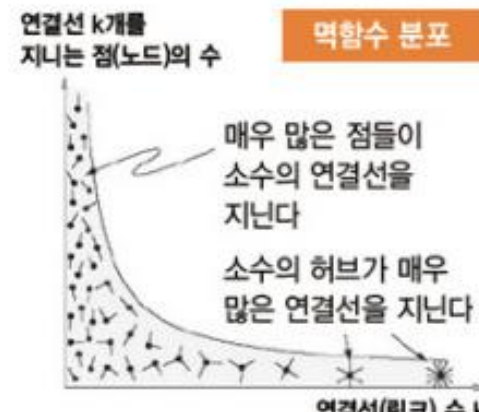
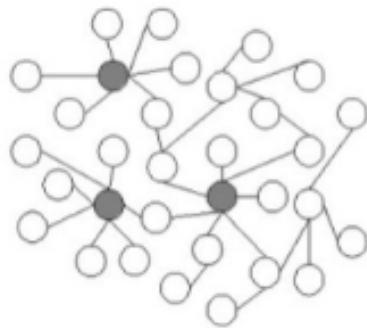
- 대다수 노드들이 유사한 수의 엣지를 갖고 있는 네트워크
- 현실에서 나타나는 네트워크의 경우 매우 높은 엣지를 가지는 노드가 나타나 무작위 네트워크로 설명이 불가능한 문제가 있음



02 네트워크의 지표

척도 없는 그래프 (scale-free graph)

- 대부분 노드들이 소수의 엣지를 갖고 있고 몇 개의 노드들이 거대한 엣지를 가지는 네트워크
- 인터넷, 소셜 네트워크와 같은 현실의 많은 네트워크에서 Scale-free graph의 모습을 보임



02 네트워크의 지표

밀도 (density)

- 최대 가능한 엣지들의 개수에 대한 실제 엣지들의 개수의 비
- (실제 네트워크에 존재하는 엣지의 개수) / (모든 노드가 전부 연결되어 있다는 가정하에 구한 총 엣지 수)
- 노드와 노드 사이에 엣지들이 얼마나 밀집되어 있는지를 판단할 수 있는 척도
 - 높은 밀도를 갖는 그래프는 낮은 밀도를 갖는 그래프에 비해 노드들 간에 더 많이 연결되어 있음.
 - 그래프 밀도는 0과 1사이의 값을 가짐
 - 그래프에서 노드들 간에 완전 연결되어 있는 경우, 밀도는 1의 값을 가짐

02 네트워크의 지표

밀도 (density)의 의미와 예시

- 네트워크에서 밀도는 네트워크 내 구성원이 서로 얼마나 많은 관계를 맺고 있는가를 표현하는 지표임
- A라는 학교에서 특정 학급 학생들 간 네트워크를 $N1$, 학교의 학생들 간 네트워크를 $N2$ 라고 할 경우, 한 학급의 학생들 간은 서로 알고 있으나, 전체 학교의 학생들 간에서 서로 모를 수도 있다는 가정 하에 $N1$ 의 밀도가, $N2$ 의 밀도보다 높다고 할 수 있음.

02 네트워크의 지표

중심성 (centrality)

- 개체가 전체 네트워크에서 얼마만큼 중심에 가까이 자리 잡고 있는지를 나타내는 지표
 - 특정한 노드가 많은 다른 노드들과 연결되어 있는 경우, 그 노드는 네트워크의 가운데 쪽으로 위치하게 됨.
 - **중심성은** 네트워크 분석에서 개체가 가지는 **영향력을 분석하는데** 많이 사용됨

02 네트워크의 지표

■ 중심성 (centrality) 지표의 종류

- 연결정도 중심성 (degree centrality)
- 근접 중심성 (closeness centrality)
- 매개 중심성 (betweenness centrality)
- 아이겐벡터 중심성 (eigenvector centrality)

02 네트워크의 지표

연결 정도 중심성 (degree centrality)

- 네트워크에서 한 노드가 다른 노드들과 **직접적으로 연결되어** 있는 지를 측정하는 지표
- 특정 노드의 연결 정도 중심성은 특정 노드와 직접 연결된 노드의 수를 특정 노드와 직, 간접적으로 연결된 모든 노드의 수로 나눈 값
 - 연결된 노드의 수가 많을 수록 연결 정도 중심성 상승
 - 이 지표는 단순히 1촌 만을 고려한 것으로 국지적인 범위의 역할만 파악 가능 → 확산된 정도는 보지 못함 (한계점)

$$D_c(i) = \frac{i\text{와 직접 연결된 노드의 수}}{i\text{와 직·간접 연결된 노드의 수}}$$

02 네트워크의 지표

연결 정도 중심성 (degree centrality)

자기 자신과 연결된 엣지 수 / ?????? [표준화를 위한 나누기]
 (표준화? - ∵ 네트워크 크기에 따라 연결된 노드 수가 같아도 그 비중이 다르기 때문)

표준화하는 방법은 많으나 책에서는 **네트워크 내 모든 엣지 수**로 나눔
 (네트워크 내 가능한 최대 수치인 (총 노드의 개수 - 1)로 나누거나
 해당 네트워크 내 가장 많은 엣지를 가진 단일 노드의 엣지 수로 나누기도 함)

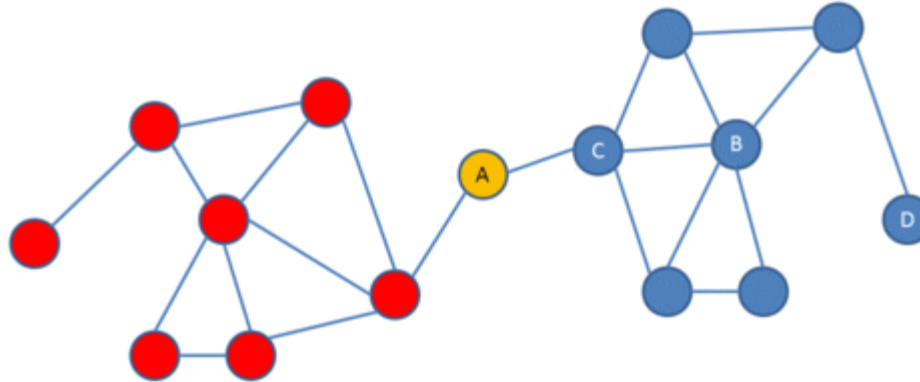
물론 방향성이 있으면 In-Degree와 Out-Degree를 따로 보아야 함

02 네트워크의 지표

■ 근접 중심성 (closeness centrality)

- 직접적으로 연결된 노드뿐만 아니라 **간접적으로 연결된 노드까지 포함**해 중심성을 측정하는 지표
- 특정 노드와 네트워크에 있는 다른 노드들 간의 최단 거리가 짧을수록 근접 중심성이 높다는 전제

$$c_c(i) = \frac{n - 1}{\sum_{j=1}^n d(i, j)}$$



노란색 노드는
꽤 중요해 보이는데도
degree centrality가 낮다.

하지만!
betweenness centrality가 출동한다면?

02 네트워크의 지표

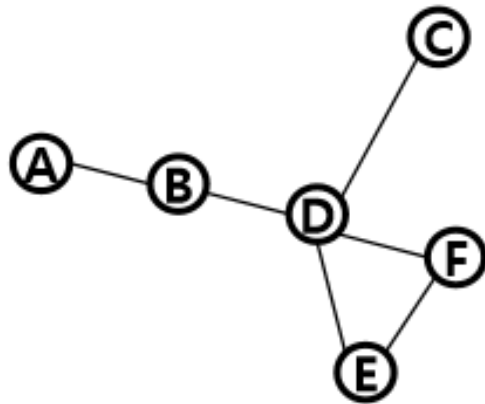
매개 중심성 (betweenness centrality)

- 해당 노드를 통과하는 최단 경로들의 개수로 정의. 경로의 끝에 있는 노드의 경우, 두 노드 간의 최단 경로가 존재하지 않으므로 매개 중심성은 0임.
 - 한 노드가 다른 노드들 간의 네트워크를 구축하는데 중계자 혹은 매개자로서의 역할 정도를 나타내는 지표
 - 전체 네트워크 내에서 얼마나 다리 역할을 하는지를 나타낼 수 있음. 즉, 상이한 집단 간을 연결하는 노드일수록 매개 중심성이 높게 나타남

$$B_c(i) = \sum_{j < k} g_{jk}(n_i)$$

02 네트워크의 지표

중심성 지표 산출 예시



■ 연결정도 중심성

A	B	C	D	E	F
$1/5=0.2$	$2/5=0.4$	$1/5=0.2$	$4/5=0.8$	$2/5=0.4$	$2/5=0.4$

■ 근접 중심성

A	B	C	D	E	F
$5/12=0.42$	$5/8=0.625$	$5/10=0.5$	$5/6=0.83$	$5/9=0.56$	$5/9=0.56$

■ 매개 중심성

A	B*	C	D**	E	F
0	4	0	8	0	0

*노드 B를 포함하는 최단경로: (A,D), (A,D,C),(A,D,E),(A,D,F)

**노드 D를 포함하는 최단경로:

(B,C),(B,E),(B,F),(A,B,C),(A,B,E),(A,B,F),(C,E),(C,F)

02 네트워크의 지표

아이겐벡터 중심성 (eigenvector centrality)

[Eigenvector centrality] is based on the intuition that a node's importance in a network is determined by **how important its neighbors are**

	A	B	C	D	b1	b2	b3	c1	c2	c3	d1	d2	d3							
A	0	1	1	1	0	0	0	0	0	0	0	0	0	\times	$=$	$\xrightarrow{\text{Normalized}}$				
B	1	0	0	0	1	1	1	0	0	0	0	0	0					3	12	0.182
C	1	0	0	0	0	0	0	1	1	1	0	0	0					4	6	0.091
D	1	0	0	0	0	0	0	0	0	0	1	1	1					4	6	0.091
b1	0	1	0	0	0	0	0	0	0	0	0	0	0					1	4	0.061
b2	0	1	0	0	0	0	0	0	0	0	0	0	0					1	4	0.061
b3	0	1	0	0	0	0	0	0	0	0	0	0	0					1	4	0.061
c1	0	0	1	0	0	0	0	0	0	0	0	0	0					1	4	0.061
c2	0	0	1	0	0	0	0	0	0	0	0	0	0					1	4	0.061
c3	0	0	1	0	0	0	0	0	0	0	0	0	0					1	4	0.061
d1	0	0	0	1	0	0	0	0	0	0	0	0	0					1	4	0.061
d2	0	0	0	1	0	0	0	0	0	0	0	0	0					1	4	0.061
d3	0	0	0	1	0	0	0	0	0	0	0	0	0					1	4	0.061

Adjacency matrix

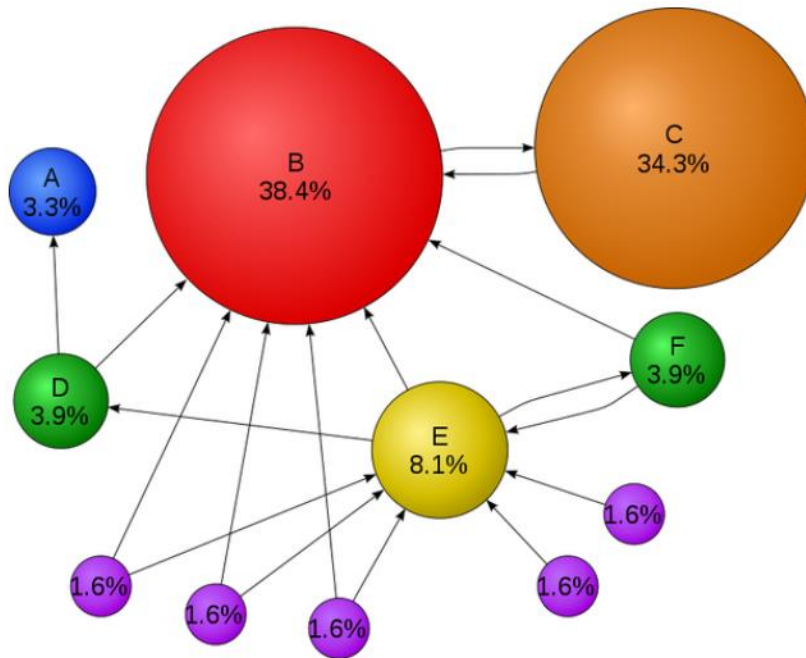
Centrality matrix
(sum of the rows)

Eigenvector Centrality

Eigenvector Centrality
(Normalized)

02 네트워크의 지표

페이지 랭크(Page Rank) = Eigenvector Centrality의 예

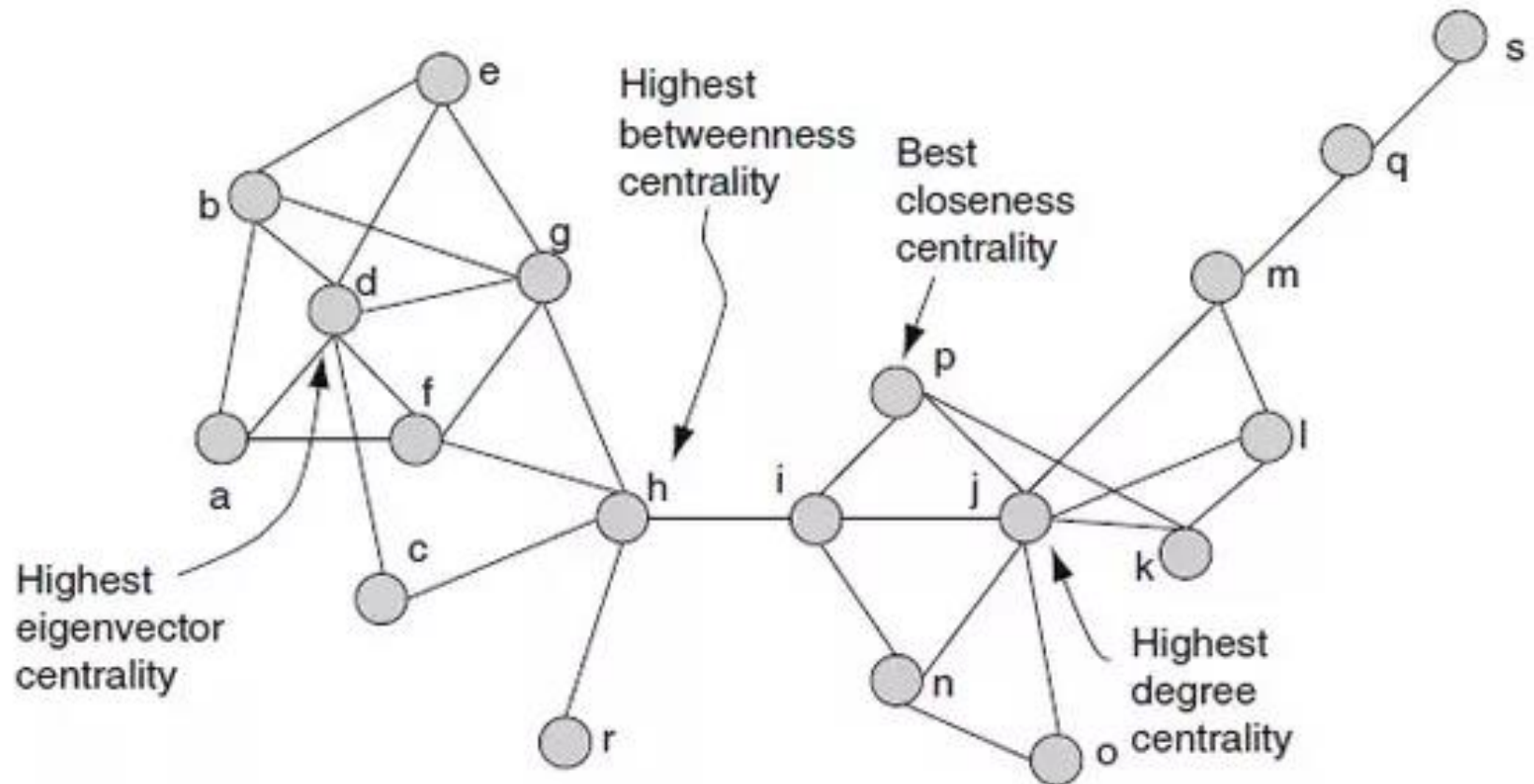


가정1: 중요한 페이지는 더 많은 다른 사이트로부터 링크를 받는다

가정2: 페이지는 임의로 방문하며 탐색한다

1. A,B,C,D라는 페이지가 있을 때 A페이지의 방문자는 A페이지를 통해 다른 페이지를 방문
2. 이러한 확률(Damping Factor)을 a 라 한다면, 다른 페이지에 $1/3 * a$ 만큼 페이지 랭크를 부여
3. 페이지 랭크는 이와 같은 방법을 통해 페이지간 페이지 랭크 값을 주고 받는 것을 반복하다보면, 전체 웹 페이지가 특정한 페이지 랭크 값을 수렴

02 네트워크의 지표



03 중심성의 이론적 배경과 계산 알고리즘

■ 중심성의 종류 (방금 했지만 개념이 아직……)

1. 연결 중심성 (degree centrality)
2. 매개 중심성 (betweenness centrality)
3. 근접 중심성 (closeness centrality)
4. 고유벡터 중심성 (eigenvector centrality)

그 외에도 많다고 합니다.

03 중심성의 이론적 배경과 계산 알고리즘

■ 본 게임 전 데이터 준비

모듈 불러오기

```
from Collections import deque
```

– 후에 자세히 설명.

```
from pprint import pprint
```

– pretty print.

– 보기 좋게 프린트해줌.

03 중심성의 이론적 배경과 계산 알고리즘

본 게임 전 데이터 준비

교재 데이터를 Python으로!

```
4 ▼ users = [  
5     { "id": 0, "name": "Hero" },  
6     { "id": 1, "name": "Dunn" },  
7     { "id": 2, "name": "Sue" },  
8     { "id": 3, "name": "Chi" },  
9     { "id": 4, "name": "Thor" },  
10    { "id": 5, "name": "Clive" },  
11    { "id": 6, "name": "Hicks" },  
12    { "id": 7, "name": "Devin" },  
13    { "id": 8, "name": "Kate" },  
14    { "id": 9, "name": "Klein" }  
15 ]  
16  
17 friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4),  
18               (4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)]  
19
```

03 중심성의 이론적 배경과 계산 알고리즘

본 게임 전 데이터 준비

```
20 # 각 유저마다 "friends"라는 Key에 빈 리스트를 Value로 할당
21 for user in users:
22     user["friends"] = []
23
24 for i, j in friendships:
25     # 각 friendship을 튜플로 갖는 리스트인 friendship에서
26     # 방향성 없는 네트워크이므로 양쪽에 서로 친구 추가
27     users[i]["friends"].append(users[j]) # j를 i의 친구로 추가
28     users[j]["friends"].append(users[i]) # i를 j의 친구로 추가
29
```

여기서 의문.

users[0]["friends"]에 users[1]을 넣기 전과 후의 users[0]는 다르다.

과연 다를까??

03 중심성의 이론적 배경과 계산 알고리즘

본 게임 전 데이터 준비

```
[{'id': 0, 'name': 'Hero', 'friends': [{'id': 1, 'name': 'Dunn', 'friends':
[...], {'id': 2, 'name': 'Sue', 'friends': [...], {'id': 3, 'name':
'Chi', 'friends': [...], {'id': 4, 'name': 'Thor', 'friends': [...],
{'id': 5, 'name': 'Clive', 'friends': [...], {'id': 6, 'name': 'Hicks',
'friends': [...], {'id': 8, 'name': 'Kate', 'friends': [...], {'id': 7,
'name': 'Devin', 'friends': [...], {...}}]}, {'id': 9, 'name': 'Klein',
'friends': [...]}]}]}, {'id': 7, 'name': 'Devin', 'friends': [...], {'id':
8, 'name': 'Kate', 'friends': [{'id': 6, 'name': 'Hicks', 'friends': [...],
...}], {...}, {'id': 9, 'name': 'Klein', 'friends': [...]}]}]}]}, {'id': 3, 'name': 'Chi', 'friends': [...], {'id': 2, 'name': 'Sue',
'friends': [...], {...}, {...}], {'id': 4, 'name': 'Thor', 'friends':
[...], {'id': 5, 'name': 'Clive', 'friends': [...], {'id': 6, 'name':
'Hicks', 'friends': [...], {'id': 8, 'name': 'Kate', 'friends': [...],
{'id': 7, 'name': 'Devin', 'friends': [...], {...}}]}, {'id': 9, 'name':
'Klein', 'friends': [...]}]}]}, {'id': 7, 'name': 'Devin', 'friends':
[...], {'id': 8, 'name': 'Kate', 'friends': [{'id': 6, 'name': 'Hicks',
'friends': [...], {...}], {...}, {'id': 9, 'name': 'Klein', 'friends':
[...]}]}]}]}]}]}, {'id': 2, 'name': 'Sue', 'friends': [...], {'id': 1,
'name': 'Dunn', 'friends': [...], {...}, {'id': 3, 'name': 'Chi', 'friends':
[...], {...}, {'id': 4, 'name': 'Thor', 'friends': [...], {'id': 5, 'name':
'Clive', 'friends': [...], {'id': 6, 'name': 'Hicks', 'friends': [...],
{'id': 8, 'name': 'Kate', 'friends': [...], {'id': 7, 'name': 'Devin',
'friends': [...], {...}}]}, {'id': 9, 'name': 'Klein', 'friends':
[...]}]}]}]}]}, {'id': 7, 'name': 'Devin', 'friends': [...], {'id': 8, 'name':
'Kate', 'friends': [{'id': 6, 'name': 'Hicks', 'friends': [...], {...}],
...}, {'id': 9, 'name': 'Klein', 'friends': [...]}]}]}]}]}]}, {'id': 3,
'name': 'Chi', 'friends': [{'id': 1, 'name': 'Dunn', 'friends': [...], {...},
...}], {...}, {'id': 4, 'name': 'Thor', 'friends': [...], {'id': 5, 'name':
'Clive', 'friends': [...], {'id': 6, 'name': 'Hicks', 'friends': [...],
{'id': 8, 'name': 'Kate', 'friends': [...], {'id': 7, 'name': 'Devin',
'friends': [...], {...}}]}, {'id': 9, 'name': 'Klein', 'friends': [...]}]}]}]}]}]}]
```

03 중심성의 이론적 배경과 계산 알고리즘

1. 연결 중심성 (degree centrality)

책에서 주어진 대로!

자기 자신과 연결된 엣지 수 / 네트워크 내 모든 엣지 수

계산 어떻게?

특정 유저의 연결 중심성을 도출하기 위해 분자와 분모 모두 계산

03 중심성의 이론적 배경과 계산 알고리즘

1. 연결 중심성 (degree centrality)

방금 준비 과정에서 각 유저는 friends 키를 가지게 되었으므로

해당 유저 사전의 friends 키에 해당하는 Value의 길이(즉, 리스트)
→ 분자

모든 유저의 friends 키에 해당하는 Value의 길이를 모두 합한 것
→ 분모

03 중심성의 이론적 배경과 계산 알고리즘

1. 연결 중심성 (degree centrality)

```
def degree_centrality(users, user_id):  
    # users에는 전체 데이터셋.  
    # user_id에는 계산하고자 하는 노드의 id.  
    # 예를 들면 1, 2  
    numerator = len(users[user_id]["friends"])  
    # 분자에 들어갈 수 계산.  
  
    denominator = 0  
    for user in users:  
        denominator += len(user["friends"])  
    # 분모에는 모든 value의 합.  
  
    return numerator / denominator
```


03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)



03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

덜 효율적이지만 이해하기 쉬운 책에 있는 알고리즘 사용


최단 경로를 구하는 알고리즘인 Breadth-First-Search (BFS)

본 세션은 BFS로 진행!

이름에서 주는 느낌처럼 옆으로 가지치기를 하는 것!

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

그래서 첫 번째로 할 수 있는 것은 임의의 두 사람이 주어졌을 때 그들 간의 최단 경로를 모두 구하는 것이다. 최단 경로를 효율적으로 구해 주는 복잡한 방법들이 많이 있지만, 이 책에서는 덜 효율적이더라도 훨씬 이해하기 쉬운 알고리즘을 사용할 것이다. 'Breadth-first search'라고도 알려져 있는 이 알고리즘은 그래도 이 책에서는 가장 복잡  중 하나이니, 찬찬히 살펴보도록 하자.

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

그래서 첫 번째로 할 수 있는 것은 임의의 두 사람이 주어졌을 때 그들 간의 최단 경로를 모두 구하는 것이다. 최단 경로를 효율적으로 구해 주는 복잡한 방법들이 많이 있지만, 이 책에서는 덜 효율적이더라도 훨씬 이해하기 쉬운 알고리즘을 사용할 것이다. 'Breadth-first search'라고도 알려져 있는 이 알고리즘은 그래도 이 책에서는 가장 복잡한 알고리즘 중 하나이니, 찬찬히 살펴보도록 하자.

1. 먼저 from user로부터 다른 모든 사용자까지의 최단 경로를 계산해 주는 최

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

책은 설명이 영 별로라..... 책 설명을 풀어서 진행

목표는 노드 간 최단 경로 구하기!

모든 노드에서부터 모든 노드까지의 최단 경로를 구해야 한다.

유저 A로부터 유저 B까지의 최단 경로를 어떻게 구할 수 있을까?

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

책은 설명이 영 별로라…… 책 설명을 풀어서 진행

목표는 노드 간 최단 경로 구하기!

모든 노드에서부터 모든 노드까지의 최단 경로를 구해야 한다.

유저 A로부터 유저 B까지의 최단 경로를 어떻게 구할 수 있을까?

A의 친구(1단계) – A의 친구의 친구(2단계) – ... – B(n단계)에서
n이 최소인 것! (아예 도달하지 못할 수도 있음.)

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

알고리즘을 한 번 상상해봅시다.

```
>>> pprint(users[0]["shortest_paths"])
{0: [[[]],
 1: [[1]],
 2: [[2]],
 3: [[1, 3], [2, 3]],
 4: [[1, 3, 4], [2, 3, 4]],
 5: [[1, 3, 4, 5], [2, 3, 4, 5]],
 6: [[1, 3, 4, 5, 6], [2, 3, 4, 5, 6]],
 7: [[1, 3, 4, 5, 7], [2, 3, 4, 5, 7]],
 8: [[1, 3, 4, 5, 6, 8],
     [2, 3, 4, 5, 6, 8],
     [1, 3, 4, 5, 7, 8],
     [2, 3, 4, 5, 7, 8]],
 9: [[1, 3, 4, 5, 6, 8, 9],
     [2, 3, 4, 5, 6, 8, 9],
     [1, 3, 4, 5, 7, 8, 9],
     [2, 3, 4, 5, 7, 8, 9]]}
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

알고리즘을 한 번 상상해봅시다.

내가 노드 A의 매개 중심성을 알고 싶으면 A의 친구를 전부 찾고,

그 친구의 친구를 전부 찾고,

그 과정에서 새롭게 출현한 노드에 도달하는 경로가 지금까지
건너간 친구의 축적

(그리고 서로 다른 경로로 온 것은 그 거리를 비교해 더 짧은 것만 최단
경로로 인정해주면 될 것 같다!)

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

이해는 가는데…… 여기서 발생하는 문제 하나.

A의 친구, A의 친구의 친구, …를 어떻게 순서대로 관리?

아까 본 지저분한 데이터 구조의 이유.

큐(queue)의 개념을 알아보시다.

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

스택 (Stack) VS 큐 (Queue)

둘 다 자료 구조에 해당

스택은 입력을 아래에서부터 위로 쌓아올리는 것
큐는 입력을 앞에서부터 뒤로 줄 세우는 것

스택은 Last In First Out

[맨 위에 자료 쌓기인 push와 맨 위 자료 빼기인 pop]

큐는 First In First Out (leftpop 가능)

[맨 뒤에 자료 줄 세우기인 enqueue / 맨 앞 자료 빼기인 dequeue]

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

from collections import deque

deque는 꼭 뒤로만 줄을 세우지 않을 수 있는 양 방향 큐!

deque에 대한 설명을 <https://goo.gl/ycosQr>에서 지금 같이 확인!

친구, 친구의 친구, ... 중 아직 경로가 잡히지 않은(즉, 처음 보는 노드)를

큐에 넣고, 큐가 빌 때까지 반복하며 관리해주면

해당 노드의 다른 노드에 대한 최단 경로를 아까 알고리즘으로 도출 가능

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):  
    #from_user는 users[0], users[1]과 같은 형식.  
    #앞으로 계속 users[0]을 변수로 넣었을 때를 가정하고 설명.  
  
    shortest_paths_to = {from_user["id"] : [[]]}  
    #shrotest_paths_to는 {0:[[]]}  
  
    frontier = deque((from_user, friend) for friend in from_user["friends"])  
    #frontier = deque((users[0],users[1]), (users[0], users[2]))
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

While문 시작!!

```
while frontier:
    # frontier에 아무것도 남아있지 않을 때 까지 반복.
    # 주석은 첫 loop를 기준으로 작성.

    prev_user, user = frontier.popleft()
    # prev_user = users[0]
    # user = users[1]

    user_id = user["id"]
    # user_id = 1

    # 큐에 사용자를 추가하는 방법을 고려해 보면
    # prev_user까지의 최단 경로를 이미 알고 있을 수도 있다
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
paths_to_prev_user = shortest_paths_to[prev_user["id"]]
# paths_to_prev_user에 shortest_paths_to라는 딕셔너리의 prev_user["id"]라는 키의
# value를 할당.
# 첫 루프에선 paths_to_prev_user에 shortest_paths_to[0]인 [[]]를 할당.
# 우리는 0에서 1로 가는 최단 경로를 보고 싶은 것.

new_paths_to_user = [path + [user_id] for path in paths_to_prev_user]
# 1로 가는 최단 경로는 0으로 가는 최단 경로에다가 1을 추가하는 의미.
# new_paths_to_user는 [[]+[1]]이 되고 [[1]]이 됨.
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
old_paths_to_user = shortest_paths_to.get(user_id, [])  
# old_paths_to_user에는 shortest_paths_to[1]의 value를 반환.  
# 하지만 shortest_paths_to에 1이라는 key가 존재하지 않을 경우는 []를 반환하겠다는 의미.  
# 아직까진 shortest_paths_to에 해당 key가 없으므로 [] 할당.
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
if old_paths_to_user:    # old_paths_to_user가 빈 리스트인지 확인.

    old_min_path_length = len(min(old_paths_to_user, key = lambda x : len(x)))
    # 빈 리스트가 아니라면, old_min_path_length는 old_paths_to_user의 element
    # 중에서, 가장 길이가 작은 것의 길이를 반환.

else:
    old_min_path_length = float("inf")
    # 빈 리스트라면, 무한대를 반환해서 무조건 대체될 수 있도록 설정.

new_min_path_length = len(min(new_paths_to_user, key = lambda x : len(x)))
# new_min_path_length에 new_paths_to_user의 element중 가장 길이가
# 작은 것의 길이를 반환.
```


03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
if new_min_path_length < old_min_path_length:
    shortest_paths_to[user_id] = [path for path in new_paths_to_user \
                                  if len(path) == new_min_path_length]
# new_min_path_length가 old_min_path_length보다 작다면,
# shortest_paths_to[1]에 해당하는 value를
# 1로 가는 최단 경로를 new_paths_to_user중에 가장 짧은 것으로 같음.

elif new_min_path_length == old_min_path_length:
    new_paths_to_user = [path for path in new_paths_to_user \
                          if len(path) == new_min_path_length \
                          and path not in old_paths_to_user]

    shortest_paths_to[user_id] = old_paths_to_user + new_paths_to_user
# 만약 둘이 같다면, new_paths_to_user 중 가장 짧은 것을 골라서 그 길이가
# old_paths_to_user 중 가장 짧은 것과 같기 때문에
# shortest_paths_to[user_id] 는 두 가지를 합친 것.

else:
    pass
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
frontier.extend((user, friend) for friend in user["friends"] \
                if friend["id"] not in shortest_paths_to)
# user, 여기서 users[1]의 친구들을 frontier의 오른쪽에 add.
# 하지만 그 친구들 중 이미 방문한 것이 있으면 건너 뛰어야함.
```

While문 끝!

```
return shortest_paths_to
```

shortest_paths_from(from_user) 끝!

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
for user in users:
    user["shortest_paths"] = shortest_paths_from(user)
# 데이터셋에 shortest_paths라는 키를 만들고, value 할당.

for user in users:
    user["betweenness_centrality"] = 0.0
# 데이터셋에 betweenness_centrality라는 키를 만들고, value 할당.
# 초기값은 0.0
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

총 최단 경로의 개수를 구하고, 그 기여를 얼마나 하였는지 확인!

```
for source in users:
    source_id = source["id"]
    # source_id에는 0,1,2 등의 값 할당.

    for target_id, paths in source["shortest_paths"].items():
        if source_id < target_id:
            # 중복되지 않게.
            # 다시말해 0에서 1로가는 것만 고려하고 1에서 0으로 가는 것은 고려X.
            num_paths = len(paths)
            # source_id에서 target_id로 가는 최단 경로의 수
            contrib = 1 / num_paths
            # 최단 경로에 노드가 포함 되어 있을 때 마다
            # betweenness_centrality에 추가되는 값.
            for path in paths:
                # 각 최단 경로에 대해
                for id in path:
                    # 포함되는 노드에 대해
                    if id != target_id:
                        # target_id와 다르다면
                        users[id]["betweenness_centrality"] += contrib
                    # 해당 노드의 betweenness_centrality에
                    # contrib을 더해준다.
```

03 중심성의 이론적 배경과 계산 알고리즘

3. 근접 중심성 (closeness centrality)

근접 중심성은 해당 사용자가 갖는 다른 모든 사용자와의 최단 거리의 합을 분모로 갖는 수치

⇒ 앞서 구한 최단 거리를 활용하여 최단 거리 합을 구하고 활용!

03 중심성의 이론적 배경과 계산 알고리즘

3. 근접 중심성 (closeness centrality)

근접 중심성은 해당 사용자가 갖는 다른 모든 사용자와의 최단 거리의 합을 분모로 갖는 수치

⇒ 앞서 구한 최단 거리를 활용하여 최단 거리 합을 구하고 활용!

```
def farness(user):  
    # user는 users[0]와 같은 형태.  
    return sum(len(paths[0]) for paths \  
                in user["shortest_paths"].values())  
    # 해당 노드가 가지는 다른 노드들 까지의 최단 경로들의  
    # 길이의 합.  
  
for user in users:  
    user["closeness_centrality"] = 1 / farness(user)
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

결국엔 인접행렬과 eigenvector를 구하는 문제

인접행렬은 friendships 리스트에 포함된 튜플을 활용해 만들기

```
adj_matrix = np.zeros((len(users),len(users)))  
# 노드의 수 만큼 정사각 영행렬 정의  
for i, j in friendships:  
    adj_matrix[i][j] = 1  
    adj_matrix[j][i] = 1
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

numpy.linalg.eig를 활용하면 eigenvector도 간단히!

numpy.linalg.eig

`numpy.linalg.eig(a)`

[\[source\]](#)

Compute the eigenvalues and right eigenvectors of a square array.

Parameters: `a` : (\dots, M, M) array

Matrices for which the eigenvalues and right eigenvectors will be computed

Returns: `w` : (\dots, M) array

The eigenvalues, each repeated according to its multiplicity. The eigenvalues are not necessarily ordered. The resulting array will be of complex type, unless the imaginary part is zero in which case it will be cast to a real type. When `a` is real the resulting eigenvalues will be real (0 imaginary part) or occur in conjugate pairs

`v` : (\dots, M, M) array

The normalized (unit "length") eigenvectors, such that the column `v[:,i]` is the eigenvector corresponding to the eigenvalue `w[i]`.

Raises: `LinAlgError`

If the eigenvalue computation does not converge.

See also:

`eigvals` eigenvalues of a non-symmetric array.

`eigh` eigenvalues and eigenvectors of a symmetric or Hermitian (conjugate symmetric) array.

`eigvalsh` eigenvalues of a symmetric or Hermitian (conjugate symmetric) array.

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

numpy.linalg.eig를 활용하면 eigenvector도 간단히!

Examples

```
>>> from numpy import linalg as LA
```

(Almost) trivial example with real e-values and e-vectors.

```
>>> w, v = LA.eig(np.diag((1, 2, 3)))
>>> w; v
array([ 1.,  2.,  3.])
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

```
from numpy import linalg as LA
```

```
w, v = LA.eig(adj_matrix) # 한 줄로 진짜 끝
```

```
>>> w[0]
2.6688229150885681
>>> v[:,0]
array([-0.38578095, -0.51479052, -0.51479052, -0.47331326, -0.23360825,
       -0.15014578, -0.08355213, -0.08355213, -0.07284006, -0.02729295])
>>> -v[:,0]
array([ 0.38578095,  0.51479052,  0.51479052,  0.47331326,  0.23360825,
        0.15014578,  0.08355213,  0.08355213,  0.07284006,  0.02729295])
>>>
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

```
w, v = LA.eig(adj_matrix)
# numpy.linalg.eig(정사각행렬)은 eigenvalue의 array와
# eigenvector의 array를 반환

for user_id, eigenvector_centrality in enumerate(-v[:,0]):
    print(user_id, eigenvector_centrality, sep = " : ")
    # 0번 인덱스에 해당하는 eigenvalue에 대응하는 eigenvector의
    # 성분을 모두 양의 실수로 변환하고 인덱스 순서대로 출력
    users[user_id]["eigenvector_centrality"] = eigenvector_centrality
    # 데이터셋에 "eigenvector_centrality"라는 키를 생성하고
    # 해당 값을 할당.
pprint(users[1]["eigenvector_centrality"])
```


네트워크와 브랜드



<https://organicmedialab.com/2012/06/23/network-is-eating-the-world-2/>

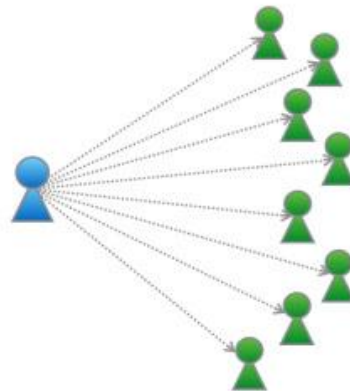
네트워크와 소비자

매개의 주체, 개인

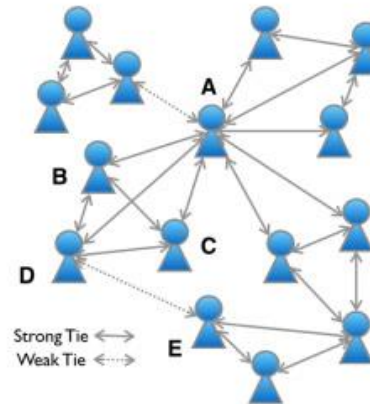
- 주는 대로 받아들이는 대중이 사라지고 스스로 미디어가 되어 네트워크를 형성한 개인(노드)들이 존재하는 상황
- 대중 매체라 불리던 TV나 신문도 이제 나와 같은 하나의 노드에 불과하다 (물론 나보다는 훨씬 목소리가 크다).

구분	산업사회	미흡하지만 4P→4C etc. 마케팅 믹스 의 변화도~	정보사회
중심 개념	생산자 중심		소비자 중심
마케팅 믹스 전략	제품 (Product)		소비자 혜택 (Customer Benefits)
	가격 (Price)		소비자 기회비용 (Cost of Customer)
	유통 (Place)		편리성 (Convenience), 커뮤니케이션 (Communication)
	촉진 (Promotion)		

매스 미디어/오가닉 미디어 네트워크 (Mass Media or Organic Media Network)



Mass Media Network



Organic Media Network

Organic Media Lab, 2015

“세상 참 좁다” – SNS와 바이럴 마케팅

‘여섯 단계의 분리’ (Six degrees of separation)

이 세상에 살고 있는 어떤 두 사람 간에 그들을 연결해 줄 경로(path)가 존재하고 심지어 이 경로가 평균 6개의 링크(지인관계)로 이루어져 있다는 주장

페이스북에서 임의의 두 사람간의 거리(경로 길이)는 평균 4.7

[정의]

- 바이럴 마케팅은 소셜 미디어를 통해 거미줄처럼 네트워크되어 있는 소비자들에게 바이러스처럼 빠르게 확산되는 새로운 마케팅 현상.
- 누리꾼이 이메일이나 SNS 등 전파 가능한 매체를 통해 자발적으로 어떤 기업 또는 제품을 홍보하도록 유도하는 마케팅 기법.

[콘텐츠 지속성과 네트워크]

- 콘텐츠가 너무 많기 때문에, 연결되지 않은 콘텐츠는 눈에 보이지 않는다.
- 콘텐츠가 지속되려면 ‘관계’를 만들어 주어야 한다. (오가닉 마케팅의 견지)

어떻게 한 사람으로부터 시작해서 네트워크 전체를 감염시킬 수 있을까?

“세상 참 좁다” – SNS와 바이럴 마케팅

EX. 네트워크 특성에 기초한 마케팅 타겟 설정

변형된 와츠와 스트로가츠 모델
(A Variation of Watts-Strogatz Model)

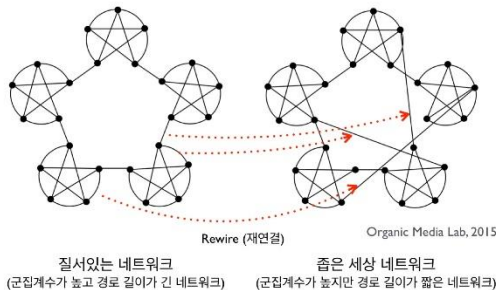
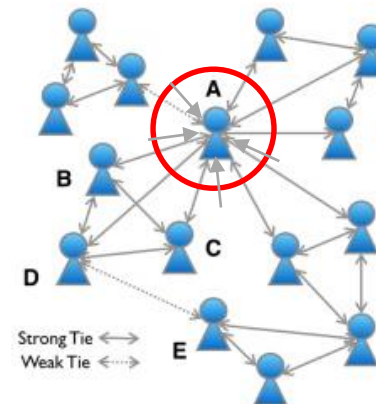


Image Source: Fang et al., "Balancing exploration and exploitation through structural design" Organization Science, 2010.

약한 연결(Weak Tie)은
메시지를 멀리 퍼지게 한다



허브(Hub)는
한번에 많은 메시지를 보낸다

소셜 네트워크는 수많은 무리들이 전세계에 흩어져 있지만
(허브에 의해) **널리** (사회적 다리에 의해) **멀리** 메시지가 전파될 수 있는 구조를 가지고 있다

타겟팅이 더 궁금하다면? <http://m.itdaily.kr/news/articleView.html?idxno=84748>

바이럴 마케팅 성공 사례



바이럴마케팅 성공사례 ② 코카콜라 댄스 자판기

영상 출처: 유튜브



유튜브 110만건 이상
폭발적인 조회수 기록

바이럴마케팅 성공사례 ① The T-Mobile Dance

영상 출처: 유튜브



바이럴마케팅 대표 성공사례
바이럴마케팅의 의미를 보여줌

바이럴마케팅 성공사례 ③ 하이네켄 광고

영상 출처: 유튜브



인상을 강렬하게 남기는
재밌고 즐거운 광고

<http://originaln.story.com/16>

상권 분석

12

한국의류학회지

Vol. 40 No. 2, 2016

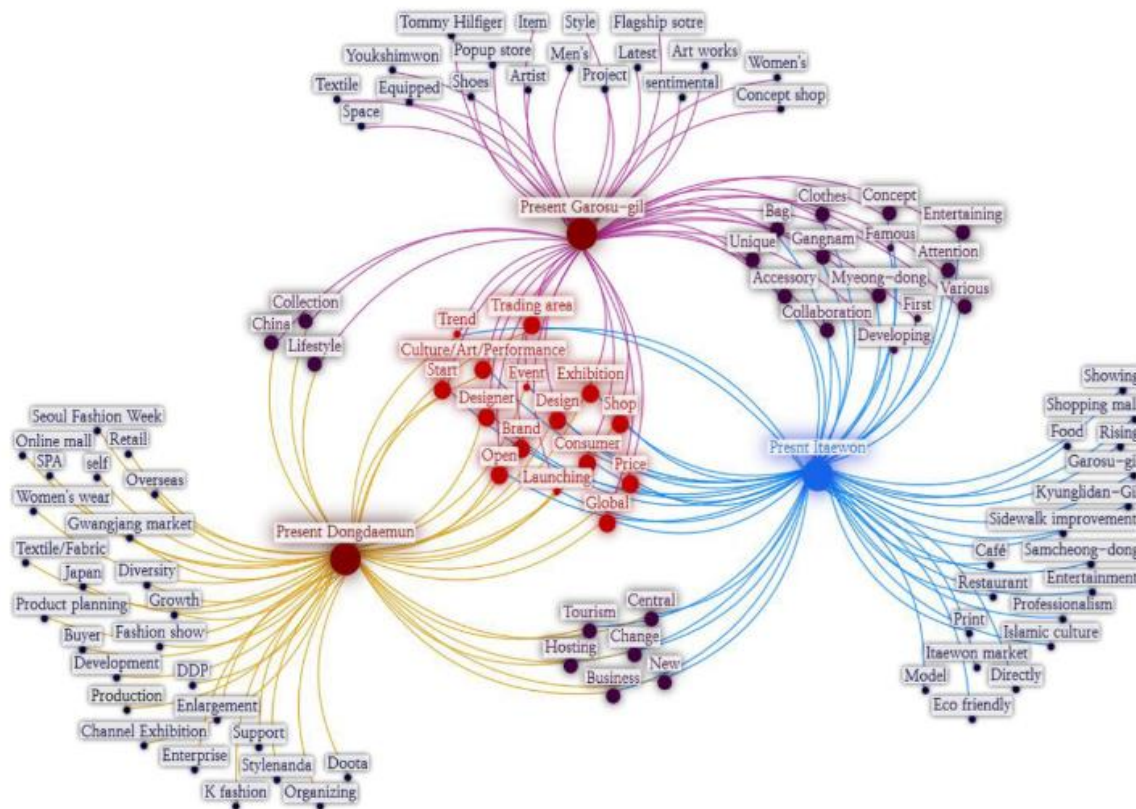


Fig. 2. Relationship between three trading areas in the present (2013-2014).

충성 고객과 이탈 고객

經營科學
第26卷 第1號
2009年 3月

183

사회 네트워크 분석을 이용한 충성고객과 이탈고객의 구매 특성 비교 연구*

김재경** · *최일영** · 김혜경** · 김남희**

Social Network Analysis to Analyze the Purchase Behavior Of
Churning Customers and Loyal Customers

Jae Kyeong Kim** · *Il Young Choi** · Hyea Kyeong Kim** · Nam Hee Kim**

충성고객 네트워크는 구매 제품의 종류가 10개 이상~15개 미만일 때 높은 연결중심성을 보였으나 이탈고객 네트워크는 구매 제품 종류의 수에 관계없이 연결중심성이 높았다. 또한 이탈고객 네트워크가 충성고객 네트워크보다 밀도가 높게 나타났다. 이는 이탈 고객들은 광고, 쿠폰 등의 마케팅 활동과 관계된 특정 제품을 구매함으로써 고객간 유사성이 높은 반면에, 충성 고객들은 자신에게 필요한 품목 중심으로 구매함으로써 고객간 유사성이 낮다고 해석할 수 있다. 또한 충성고객과 이탈고객의 구매 제품을 분석한 결과 이탈고객은 충성고객보다 메이크업 제품을 더 많이 구매함을 알 수 있었다.

제품 가치와 네트워크

Gal Oestreicher-Singer, Barak Libai, Liron Sivan, Eyal Carmi, & Ohad Yassin

The Network Value of Products

Traditionally, the value of a product has been assessed according to the direct revenues the product creates. However, products do not exist in isolation but rather influence one another's sales. Such influence is especially evident in e-commerce environments, in which products are often presented as a collection of web pages linked by recommendation hyperlinks, creating a large-scale product network. The authors present a systematic approach to estimate products' true value to a firm in such a product network. Their approach, which is in the spirit of the PageRank algorithm, uses available data from large-scale e-commerce sites and separates a product's value into its own intrinsic value, the value it receives from the network, and the value it contributes to the network. The authors demonstrate their approach using data collected from the product network of books on Amazon.com. Specifically, they show that the value of low sellers may be underestimated, whereas the value of best sellers may be overestimated. The authors explore the sources of this discrepancy and discuss the implications for managing products in the growing environment of product networks.

Keywords: product value, cross-selling, electronic commerce, recommendation systems, social networks

네트워크와 브랜드 가치

브랜드는 ‘인식’ 의 싸움이다

구매자가 그 브랜드의 상품, 유통 채널, 직원, 그리고 커뮤니케이션을 접하면서 오랜 기간에 걸쳐 형성한 긍정적 또는 부정적 인상의 결정체

-장 노엘 캐퍼러-

브랜드란 사업자가 제공하는 것이 아니라 **고객, 시장을 통해 ‘구축된다’** 는 관점
브랜드의 발달, 성장, 진화, 소멸의 사이클은 사용자의 경험에서 시작된다

04 네트워크 활용 사례

Journal – Network & Tuberculosis

Transmission Network Analysis to Complement Routine Tuberculosis Contact Investigations

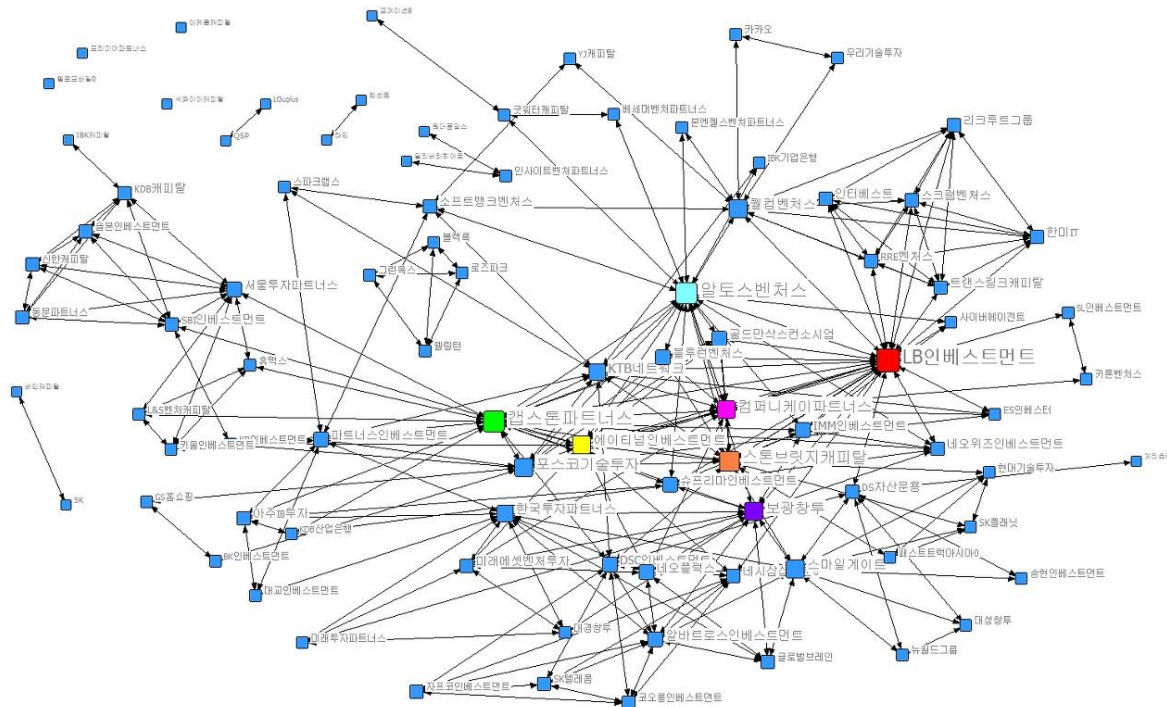
Objective. We examined the feasibility and value of network analysis to complement routine tuberculosis (TB) contact investigation procedures during an outbreak.

Methods. We reviewed hospital, health department, and jail records and interviewed TB patients. *Mycobacterium tuberculosis* isolates were genotyped. We evaluated contacts of TB patients for latent TB infection (LTBI) and TB, and analyzed routine contact investigation data, including tuberculin skin test (TST) results. Outcomes included number of contacts identified, number of contacts evaluated, and their TST status. We used network analysis visualizations and metrics (reach, degree, betweenness) to characterize the outbreak.

Results. The index patient was symptomatic for 8 months and was linked to 37 secondary TB patients and more than 1200 contacts. Genotyping detected a 21-band pattern of a strain W variant. No HIV-infected patients were diagnosed. Contacts prioritized by network analysis were more likely to have LTBI than nonprioritized contacts (odds ratio=7.8; 95% confidence interval= 1.6, 36.6). Network visualizations and metrics highlighted patients central to sustaining the outbreak and helped prioritize contacts for evaluation.

Conclusions. A network-informed approach to TB contact investigations provided a novel means to examine large quantities of data and helped focus TB control. (Am J Public Health. 2007;97:470–477. doi:10.2105/AJPH.2005.071936)

네트워크와 벤처캐피탈



- 벤처 캐피탈 (Venture Capital)
 - 벤처기업에 주식투자 형식으로 투자하는 기업 또는 기업의 자본
 - 장래성과 수익성에 주목하여 투융자

04 네트워크 활용 사례

Journal – Network & VC

Whom You Know Matters: Venture Capital Networks and Investment Performance

Yael V. Hochberg, Alexander Ljungqvist, Yang Lu

〈Abstract〉

Many financial markets are characterized by strong relationships and networks rather than arm's-length, spot-market transactions. We examine the performance consequences of this organizational choice in the context of relationships established when VCs syndicate portfolio company investments, using a comprehensive sample of U.S. based VCs over the period 1980 to 2003. VC funds whose parent firms enjoy more influential network positions have significantly better performance, as measured by the proportion of portfolio company investments that are successfully exited through an initial public offering or a sale to another company. Similarly, the portfolio companies of better networked VC firms are significantly more likely to survive to subsequent rounds of financing and to eventual exit. The magnitude of these effects is economically large, and is robust to a wide range of specifications. Once we control for network effects in our models of fund and portfolio company performance, the importance of how much investment experience a VC has is reduced, and in some specifications, eliminated. Finally, we provide initial evidence on the evolution of VC networks.

04 Journal – Network & VC

논문 소개 및 목적 – Syndicate ⇔ Networking

- Networks are widespread in many financial markets
 - VCs tend to **syndicate** their investments with other VCs, Rather than investing alone(Lerner(1994a))
 - 신디케이트 (Syndicate)
 - 일반적으로는 기업**연합**을 말하는 것이지만 증권용어로는 주식이나 공사채 등의 유가증권 발행 시 그 인수를 위하여 결성되는 인수단
 - 인수 증권의 판매력 확대와 위험분산
- EX) 양질의 딜 소싱을 하는 VC + 벤처기업 육성을 잘하는 VC
⇒ Networking이 매우 중요

04 Journal – Network & VC

논문 소개 및 목적 – 네트워킹의 장점

- 1) 미래의 호혜성
- 2) 리스크 감소 + 정보 증가
→ 더 좋은 투자처 발굴
- 3) 포트폴리오 다양화 가능성
→ 각 VC 기업이 지닌 전문가들의 협업
→ 섹터 바운더리를 확장할 수 있음
- 4) 포트폴리오 기업들의 성장 가능성 up

Network =



▪ In the VC market,
greater centricity may translate into better access to information, deal flow,
deeper pools of capital, expertise, contacts, and so on.

그러나,

네트워킹이 만연하다~ 정도의 논문은 많지만
조직화된 행동의 성과, 결과와 관련한 연구는 부족

04 Journal – Network & VC

Network Analysis Methodology

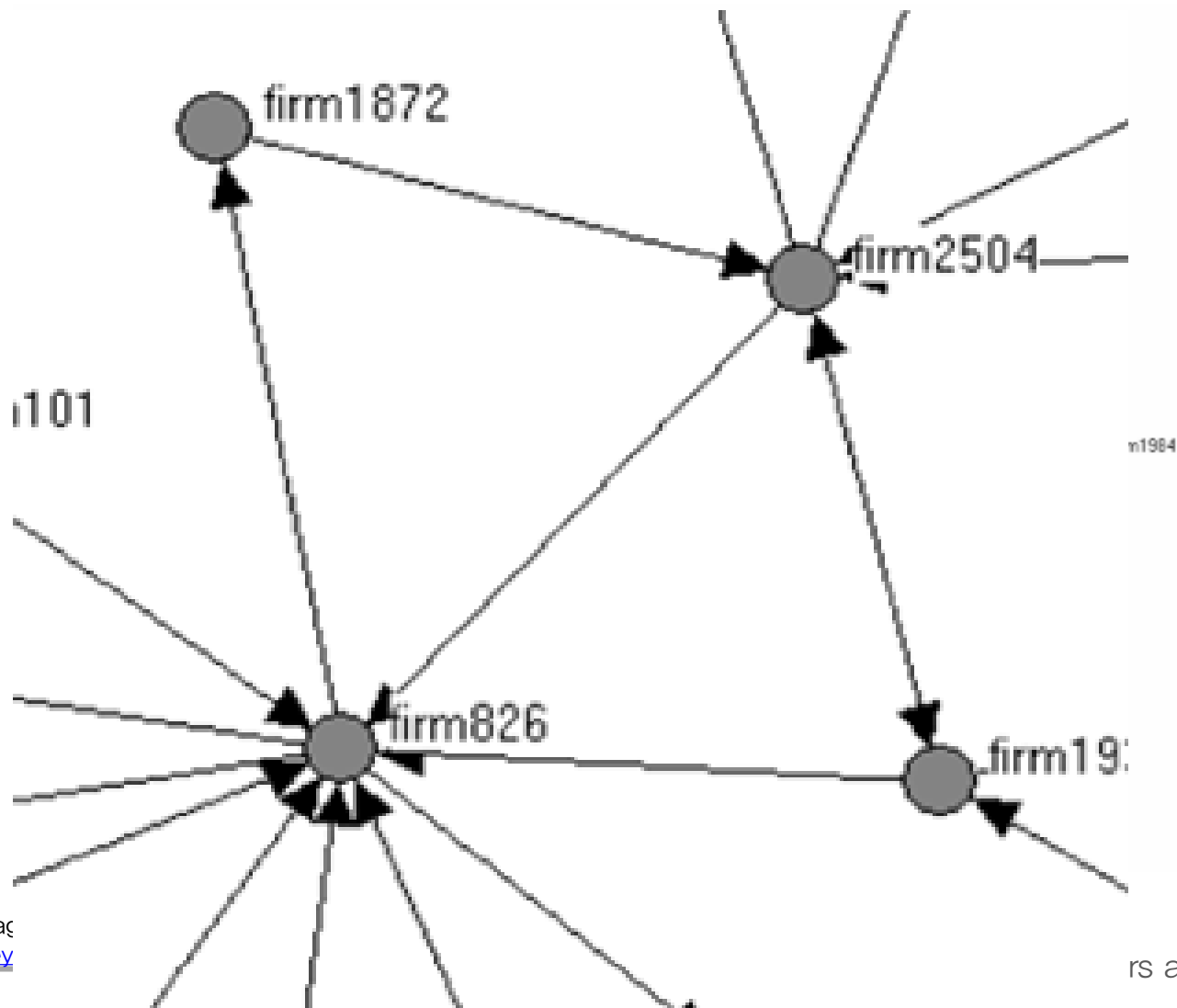
Graph Theory(그래프 이론)

- 변수 상호간의 정상적인 의미에서의 그래프를 다루는 것이 아니라, 특수한 의미의 그래프를 다루는 수학적 방법이다.
- 점과 그것들을 연결하는 선에 의해 구성된다.
선이 점과 연결되는 방식, 순환의 방식, 선이 점과 접하거나 분리하는 방식, 그래프가 하위 그래프를 갖는가 등을 다룬다.
- 점은 개인이나 집단이고 선은 그들 간의 사회적 관계의 망이 된다.

사회학사전, 2000. 10. 30., 사회문화연구소

04 Journal – Network & VC

Network of Biotech VC firms, 1990–1994



04 Journal – Network & VC

II. Sample & Data

VC fund & 포트폴리오 기업 두 가지 관점

1. Sample: US based VCs over the period 1980 to 2003

2. Fund의 라이프사이클에 따른 분류

- 벤처 투자 기본 투자 기간이 미국에선 약 10년 가량
- 1980년(벤처투자 개념 등장 시기)~1999년 까지 생성된 펀드
- 2003년부터 성과 측정 -> 최소 4년의 성장 기간 제공
- ➔ VC 산업의 유동성 반영하기 위함

II. Sample & Data

VC fund & 포트폴리오 기업 두 가지 관점

Fund characteristics

fund size (\$m)

sequence number

first fund (fraction, %)

seed or early-stage fund (fraction, %)

Fund performance

exit rate (% of portfolio companies exited)

IPO rate (% of portfolio companies sold via IPO)

M&A rate (% of portfolio companies sold via M&A)

dollar exit rate (% of invested \$ exited)

dollar IPO rate (% of invested \$ exited via IPO)

dollar M&A rate (% of invested \$ exited via M&A)

Fund parent's experience (as of vintage year)

days since parent's first investment

no. of rounds parent has participated in so far

aggregate amount parent has invested so far (\$m)

no. of portfolio companies parent has invested in so far

Network measures (as of vintage year)

outdegree

indegree

degree

betweenness

eigenvector

Competition

VC inflows in fund's vintage year (\$bn)

Investment opportunities

average P/E ratio in fund's first 3 years

average B/M ratio in fund's first 3 years

A. Fund Characteristics

B. Measuring Fund Performance

익명 혹은 통합된 IRR 대신 Exit(M&A와 IPO)으로

C. Company-Level Performance Measure

다음 라운드에서의 생존은 잠정적인 성공 시그널로 이해

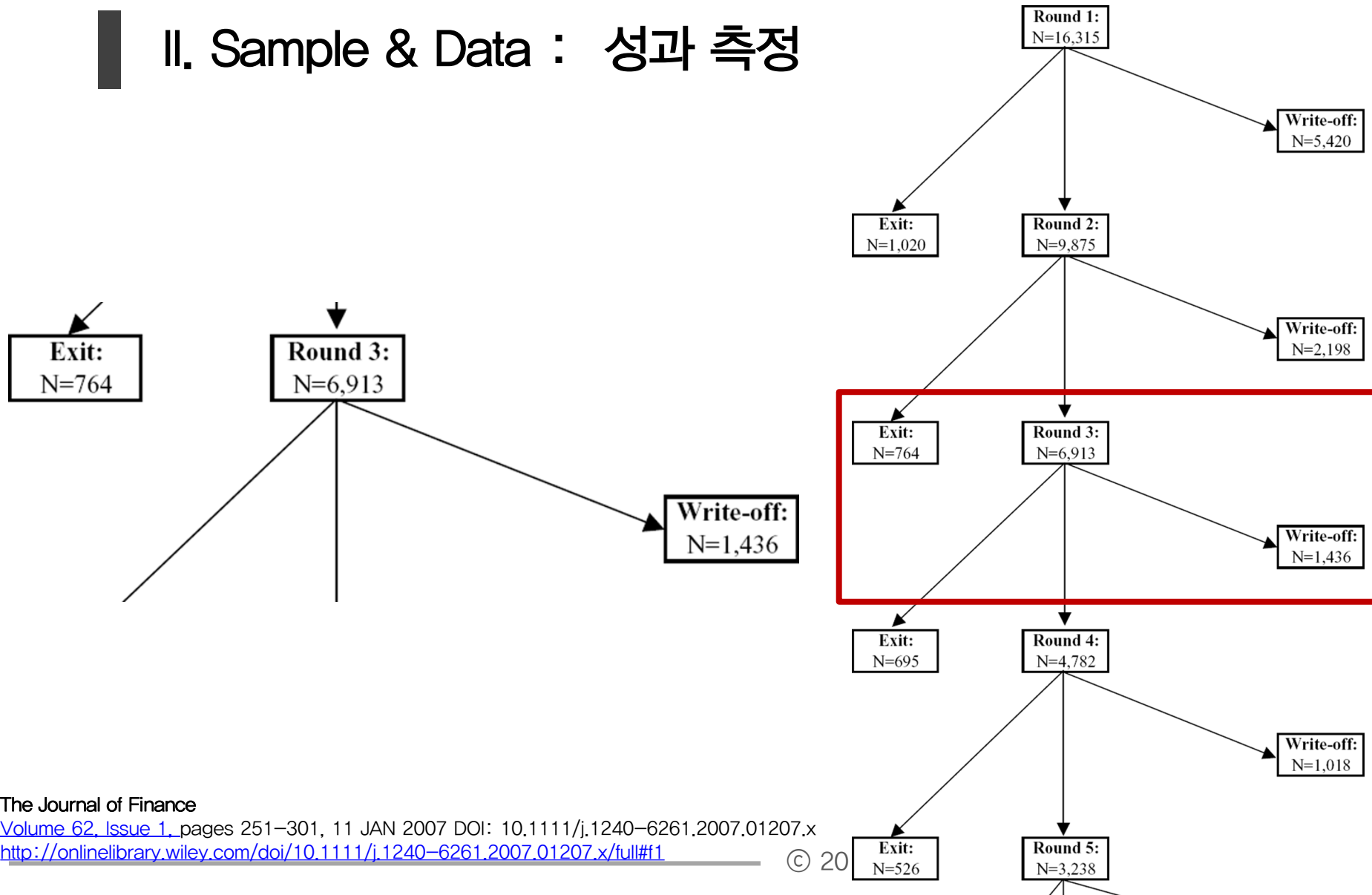
D. VC firm Experience

E. Network Measures

F. Competition for Deal Flow and Investment Opportunities

B/M(low book to market ratio): 반비례관계 P/E ratio

II. Sample & Data : 성과 측정



04 Journal – Network & VC

II. Sample & Data

E. Network Measures

Density of ties (% of theoretical max.)

undirected ties

directed ties

mean	s.d.	mean	s.d.
3.7	19.0	0.8	9.0
3.7	18.8	0.7	8.6
3.5	18.4	0.7	8.3
3.6	18.7	0.7	8.2
3.5	18.4	0.7	8.1
3.5	18.3	0.7	8.0
4.1	19.9	0.7	8.3
4.0	19.6	0.8	8.8
4.1	19.9	0.8	9.1
4.2	20.0	0.9	9.4
4.5	20.6	1.0	9.8
4.5	20.7	1.0	9.9
4.4	20.4	1.0	10.0
4.2	20.0	1.0	9.9
3.9	19.3	1.0	9.7
2.8	16.5	0.7	8.3
2.2	14.7	0.6	7.4
1.8	13.4	0.5	6.8
1.5	12.2	0.4	6.2
1.4	11.6	0.3	5.7
1.2	11.1	0.3	5.4
1.2	11.0	0.3	5.4
1.2	10.8	0.3	5.3
1.2	11.0	0.3	5.3

Table I. Descriptive Statistics (continued)

	No.	Mean	Std. dev.	Min	Median	Max
Network measures (as of vintage year)						
outdegree (Lead)	3,469	1.203	2.463	0	0.099	22.91
indegree	3,469	1.003	1.671	0	0.210	13.54
degree	3,469	4.237	6.355	0	1.245	41.29
betweenness	3,469	0.285	0.750	0	0.004	7.16
eigenvector	3,469	3.742	5.188	0	1.188	30.96
Competition						
VC inflows in fund's vintage year (\$bn)	3,469	23.842	29.349	2.295	6.474	84.632

04 Journal – Network & VC

II. Sample & Data

E. Network Measures

- Fund의 절반 이상이 신디케이트 펀드이지만 매우 낮은 연결 중심성을 보임
→ 소수의 VC들만 참여하는 배타적이고 안정적인 관계
- Centricity Variation
→ never syndicated VC (1995–1999) 존재 (10.3%, total: 1,820 VCs)

Degree / Closeness / Betweenness
중심성 종류에 따라 미묘하게 달라지는 의미를 비교해봅시다.

04 Journal – Network & VC

I. Network Analysis Methodology

1. Degree Centrality (연결중심성)

- ① 네트워크에서 액터가 관계 맺고 있는 수를 측정한다
 $\sum P_{ij}$ = 신디케이트 관계가 몇 회 있었는가
- ② 더 많은 VC들과 연결 관계를 가지면 우위를 점한다
 - 노드가 많을 수록 정보나 deal flow*에 있어서 덜 의존적
 - wider range of contacts, expertise, and pools of capital

*Deal Flow : rate at which they receive business proposals/investment offers

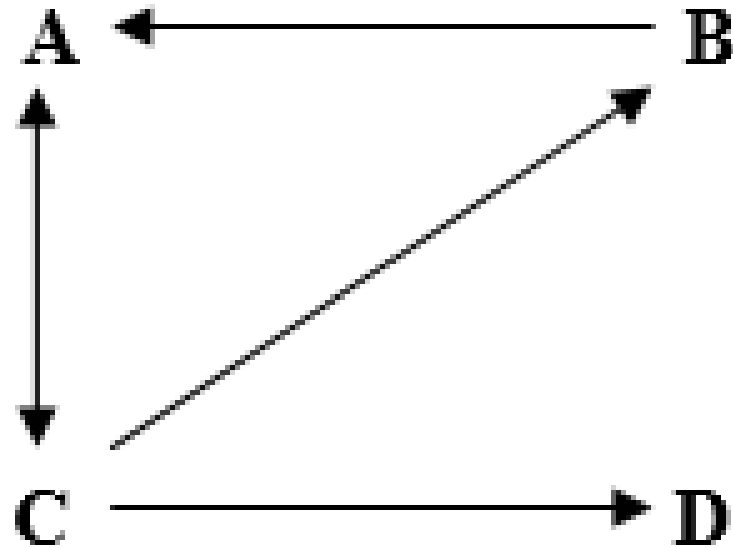
- ③ Normalization
 - 시간에 따라 변화하는 업계 특성을 반영
 - 중심성 / (n-1)
 - n-1 : maximum possible degree in an n-actor network

04 Journal – Network & VC

I. Network Analysis Methodology

1. Degree Centrality (연결중심성)

- ④ Undirected / Directed
Indegree ←
Outdegree →



04 Journal – Network & VC

Network of Biotech VC firms, 1990–1994

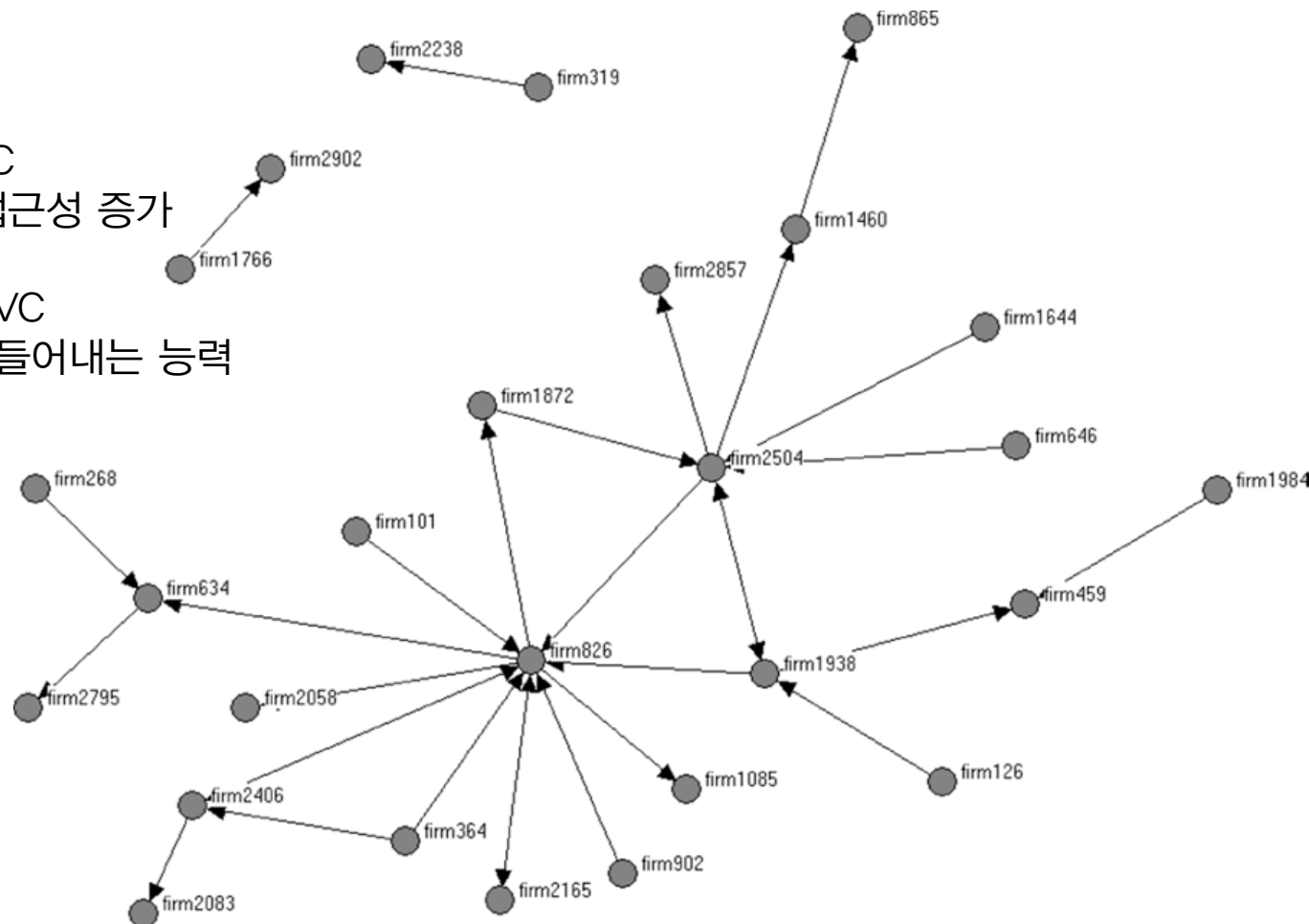
Directed Model

Indegree: 제안 받은 VC

← 정보 및 투자 기회 접근성 증가

Outdegree: 제안 하는 VC

→ 공동 투자 기회를 만들어내는 능력
측정의 수단



04 Journal – Network & VC

I. Network Analysis Methodology

2. Between Centricity (매개중심성)

“Intermediary”

능력자 VC들을 결집시키거나, 직접적으로 관련 없던 투자 기회를 제안하는 매개자 역할의 VC를 측정하는데 활용

➔ 측정 결과, 성과에 미치는 영향은 미미한 것으로 나타남

Having the ability to act as a broker between other VCs (betweenness) has a smaller effect, with a one-standard-deviation increase in this centrality measure being associated with only a one percentage point increase in fund performance. This will prove to be true throughout our analysis, suggesting that indirect relationships (those requiring intermediation) play a lesser role in the venture capital market. (p.21)

04 Journal – Network & VC

I. Network Analysis Methodology

3. Closeness

중요한 노드일수록 다른 노드까지 도달하는 경로가 짧을 것이라는 가정

Ex: 사회문화 A도시 vs B도시

➔ 액터의 Quality

4. Eigenvector Centrality

- 연결된 다른 액터들의 중심성을 **가중치**로 계산
 VC_i 의 아이겐벡터 중심성 = $\sum P_{ij} * ev_j$
- A노드와 연결된 B노드가 많은 엣지(연결관계)를 가지고 있으면
A노드의 아이겐벡터 중심성이 높아진다

연결 중심성이 높아도 연결된 각 노드의 영향력이 낮으면
(아이겐벡터 중심성이 낮으면) 영향력이 낮다

III. Fund Level Analysis

C. The effect of Firm Networks on Fund performance

Table IV. Network measures X (The Effect of Firm Experience on Fund Performance)

Diagnostics

Adjusted R^2

Test: all coefficients = 0 (F)

No. of observations

18.4 %	18.0 %	18.8 %	18.0 %
44.2***	42.7***	44.3***	42.5***
3,105	3,105	3,105	3,105

Table V. Network measures O (The Effect of Firm Networks on Fund Performance)

Network measures

outdegree

0.006***
0.002

indegree

0.013***
0.003

degree

0.003***
0.001

betweenness

0.013**
0.006

eigenvector

0.004***
0.001

Diagnostics

Adjusted R^2

Test: all coefficients = 0 (F)

No. of observations

18.9 %	19.1 %	19.1 %	18.8 %	19.1 %
43.4	44.3	44.1	42.8	44.1
3,105	3,105	3,105	3,105	3,105

결정계수 상승

04 Journal – Network & VC

III. Fund Level Analysis

C. The effect of Firm Networks on Fund performance

1) Eigen Vector>degree>indegree 순으로 경제적 효과가 큰 것으로 밝혀짐

(To illustrate, a one-standard-deviation increase in these measures is associated with a more than two percentage point increase in exit rates, all else equal.)

2) Thus a VC benefits from having many ties (degree), especially when the ties involve other well-connected VCs (eigenvector), and from being invited into many syndicates (indegree).

3) Outdegree, 즉 펀드를 리드하는 경우에는 “미래의 “ 호혜성을 기대하는 것이 적합. 상대적으로 경제적 효과 적음

04 Journal – Network & VC

IV. Company Level Analysis

C. Portfolio Company Exit

- 평균 Exit 기간 24분기
- 아이젠벡터가 가장 높은 성과를 보임
 - “Lead VC의 아이젠벡터 중심성의 std가 1 증가하면 2분기가 단축됨”
 - ➔ 영향력 높은 VC들과 관계를 맺고 있는 VC의 포트폴리오 기업들은
 - ➔ Exit 속도가 빨라진다.

Outdegree, indegree, degree 중심성이 높은 VC도 약 1분기 가량 단축

* Robust 통계량 : 이상 값에 영향을 적게 받는 통계량

ex: 5명의 마을 주민 월급 평균 13만원 + 빌게이츠가 이사옴 ➔ 이상값(outlier) 발생

➔ 평균 대신 중앙값 사용

04 Journal – Network & VC

VII. Conclusion

V. Further Robustness Tests

VI. How do VC Firms Become Networked?

의의

벤처캐피탈 산업에서 조직화된 형태로 네트워크가 가지는 경제적 가치를 분석한 첫 사례

활용 가능성

- 다른 금융 업계의 네트워킹 효과 분석
- 앞서 살펴본 다양한 마케팅 사례에도 적용 가능

한계

VCs likely benefit from personal network ties which we have not so far taken account of.

More broadly, what determines a VC' s choice whether or not to network?

What are the costs associated with becoming well-networked?

And how does one form relationships with influential VCs in the network?