Network Community Detection

- network에서 연결 밀도가 높은 집단끼리 서로 묶어 분석하는 방법
- Community: 다른 집단보다 모듈성(modularity)가 높은 집단
  - community, group, cluster 모두 같은 의미
  - 내부에서는 수많은 연결(edge)가 존재하고 외부와는 연결이 적은 node들의 집합

1. Modularity

- network가 얼마나 각 community로 얼마나 잘 나누어 졌는지에 대한 측정기준
- network에서 communities를 찾을 때 modularity를 최대화하는 것이 목표
- network를 disjoint한 그룹 s들로 나누었다고 가정했을 때

$$Q \propto \sum_{s \in S}[(\#\text{ edges within group s})-(\text{expected }\#\text{ edges within group s})]$$

  - 이를 위해서는 null model이 필요 (expected # edges 계산을 위해)

Network Community Detection

1. Modularity

Null Model : configuration model

- Null Model은 원래의 네트워크와 비교할 때 사용되는 기준 네트워크를 말함.
- N개의 node, m개의 edge로 구성된 network G를 기준으로 새로운 G' 을 만든다
- G'는 원래 네트워크G와 degree* distribution은 같고 connection을 완전히 무작위로 재배치함(uniformly random).
- G'은 multigraph**
- Node I, j 사이의 expected number of edge :
  - 각 degre는 k_i, k_j

$$k_i * \frac{k_j}{2m} = \frac{k_i k_j}{sm}$$ 무방향 네트워크일 때 s=2임.

무방향 네트워크에서 두 노드 i, j 사이에 엣지가 형성될 기대값
(분모의 2m은 전체 가능한 엣지 쌍의 수)

- Expected number of edges in G'

$$= \frac{1}{2} \sum_{i \in N} \sum_{j \in N} \frac{k_i k_j}{2m} = \frac{1}{2} \frac{1}{2m} \sum_{i \in N} k_i (\sum_{j \in N} k_j) = \frac{1}{4m} 2m 2m = m$$

2로 나누는 이유: 무방향 네트워크에서는 두 노드 사이의 연결을 양방향으로 고려할 필요가 없기 때문임. (이중 계산을 피하기 위해서)

* degree: 각 노드가 가지고 있는 연결의 수(엣지의 수)
** multigraph: 두 노드 사이에 여러 개의 엣지를 허용하는 그래프로 다중연결, 자기 루프, 가중치 및 방향성의 특징을 가짐.

Network Community Detection

1. Modularity

- Modularity: 최적의 community를 찾기 위한 평가지표 중 하나임.
  네트워크가 얼마나 잘 모듈화되어 있는지, 즉 노드들이 얼마나 강하게 클러스터화 되어 있는지를 수치화함.
  이 값이 높을수록 네트워크 내에 명확한 커뮤니티 또는 모듈이 존재함.

- 네트워크 G에 대한 모듈성을 나타내는 지표

$$Q \propto \sum_{s \in S}[(\#\text{ edges within group s}) - (\text{expected }\#\text{ edges within group s})]$$

$$Q(G,S) = \frac{1}{2m}\sum_{s\in S}\sum_{i\in s}\sum_{j\in s}\left(A_{ij} - \frac{k_i k_j}{2m}\right) = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j)$$

* 동일한 커뮤니티 내 i와 j노드가 연결되어 있을 때(Aij=1)
  i, j 노드에 붙어있는 연결선이 많음 -> 음수
  i, j 노드에 붙어있는 연결선이 적음 -> 양수

* 동일한 커뮤니티 내 i와 j노드가 끊겨 있을 때 (Aij=0)
  I, j 노드에 붙어있는 연결선이 많음 -> 큰 음수
  I, j 노드에 붙어있는 연결선이 적음 -> 작은 음수

- S: 네트워크 내의 커뮤니티 집합
- m: 네트워크 내의 모든 에지(연결)의 수
- Aij: 인접 행렬에서 노드 i와 노드 j 사이의 엣지의 유무를 나타냄. 에지가 있으면 1, 없으면 0
- ki: 노드 i의 차수(degree)로, 노드 i에 연결된 엣지의 수
- $\delta(c_i, c_j)$: 크로네커 델타 함수로, 노드 i와 j가 같은 커뮤니티에 속해 있을 경우 1, 아닐 경우 0

- Modularity의 범위 : [-1, 1]
  - 당연히 수치가 클수록 community structure가 강한 것
  - 0.3~0.7 정도면 유의미한 community structure
  - 음수면 anti community를 의미

Network Community Detection

2. community를 추출하는 방법

1) community간 link를 자르는 방법
- divisive algorithm으로 network가 분열되도록 community간 link를 잘라나가는 방법
- 대표적인 방법이 link의 betweenness 중심성을 조사하는 방법으로, Girvan-Newman algorithm이 있음.

2) 점진적으로 합쳐나가는 방법
- agglomerative algorithm으로 비슷한 node, community를 점진적으로 합쳐나가는 방법
- 주로, clustering기법을 사용하며 link community, Louvain algorithm 같은 방법이 있음.

Network Community Detection

3. Louvain Algorithm

- Modularity의 단점
  - 계산에 긴 시간이 소요됨.
  - 큰 규모의 super-communities를 생산해버리는 경향이 있음.
    (인공적으로 community구조가 없도록 만든 network에서도 거대한 규모의 community를 추출하기도 함.)
- 위 문제점을 해결하기 위해 고안된 알고리즘

- 개념
  - 한개의 node로부터 주변 node들을 흡수하며 community를 생성해 나가는 방식(Greedy algorithm*)
  - 개선된 modularity 계산법을 사용하며 network의 계층적 구조를 이용함.
  - 속도가 빨라서 크기가 큰 network에서 많이 사용됨.

* Greedy algorithm: 선택의 순간마다 당장 눈앞에 보이는 최적의 상황만을 쫓아 최종적인 해답에 도달하는 방법

- 진행 순서
  - Phase 1 : Modularity를 최적화하는 방향으로 node를 community에 할당
  - Phase 2 : community가 정해진 뒤에 super-node로 aggregate

Network Community Detection

3. Louvain Algorithm

Method

Louvain : 1st phase Partitioning

- 각 node를 특정 각 community에 할당 (one node per comunity)
- 각 node들에 대해 두가지 계산을 시행
  - node i를 이웃 j의 community로 할당한 뒤 modularity delta(ΔQ) *계산
  - 가장 큰 modularity gain를 만들어내는 community로 할당
- 더이상 modularity가 커지지 않을때 멈춤.
  (node에 대한 계산을 수행하는 순서에 따라 값이 달라지지만 큰 영향은 없으니 무시해도 된다고 함.)

$\frac{\sum_{in}+k_{i,in}}{2m}$ • 노드 i를 커뮤니티 C에 추가했을 때, 커뮤니티 C 내부의 링크 가중치 비율

$(\frac{\sum_{tot}+k_i}{2m})^2$ • 커뮤니티 C에 노드 i를 추가했을 때 커뮤니티 C의 총 링크 가중치 비율의 제곱

* modularity delta : node i를 community C로 옮겼을 때 변화량

node i가 community에 배속된 상태의 modularity − node i가 배속되지 않은 상태의 modularity

$$\Delta Q(i \to C) = [\frac{\sum_{in}+k_{i,in}}{2m} - (\frac{\sum_{tot}+k_i}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k_i}{2m})^2]$$

- $\sum_{in}$: C에 속한 node끼리의 link weight 합
- $\sum_{tot}$: C에 속한 node의 모든 link weight 합(커뮤니티 C에 연결된 전체 링크의 가중치)
- $k_{i,in}$: C와 node i사이의 link weight 합
- k_i : node i의 link weight 합

$$\frac{1}{2m}\sum_{i,j\in C}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) = \frac{1}{2m}\sum_{i,j\in C}A_{ij} - (\frac{1}{2m})^2\sum_{i,j\in C}k_i k_j = \frac{\sum_{in}+k_{i,in}}{2m} - (\frac{\sum_{tot}+k_i}{2m})^2$$

node i를 community C로 옮겼을 때
커뮤니티 내부의 연결이 랜덤 기대치보다 얼마나 높은가

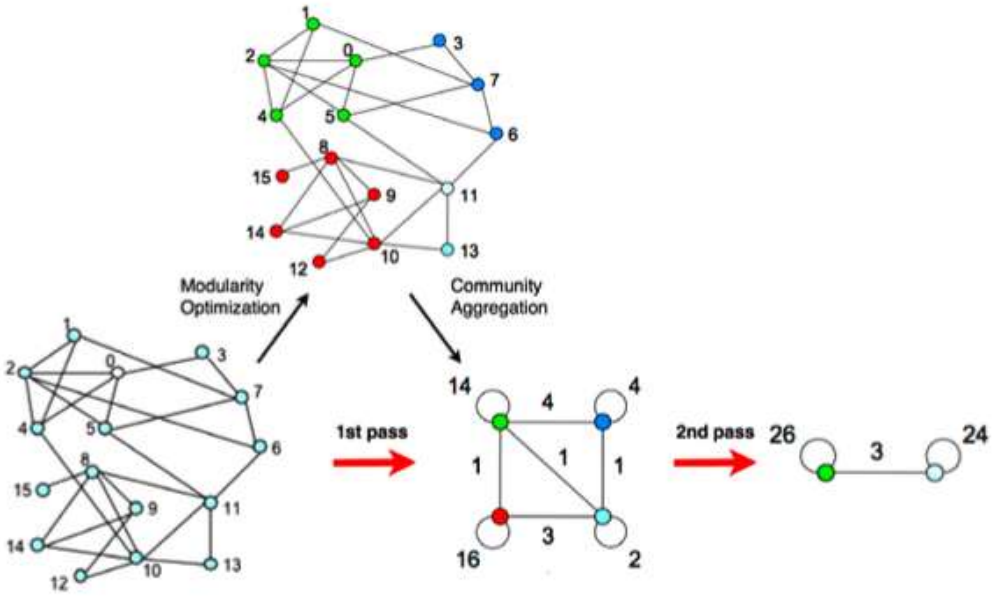
Network Community Detection

3. Louvain Algorithm

Method

Louvain : 2nd phase Restrcturing

- Phase 1에서 생성된 community를 하나의 block으로 합쳐서 node취급
- community 내부 link의 weight은 자기 회귀 상태의 link로, community간에 연결되어있던 node간의 link weight는 합쳐서 하나의 link로 생성
- 이 후, 이 새롭게 변형한 network는 다시 Phase 1의 algorithm을 통해 병합
- Phase 1을 마치면 다시 Phase 2로 돌입
- Phase 2 이후, Phase 1에서 더 이상 변화가 일어나지 않을 때 Louvain algorithm은 동작을 정지



출처

- 논문
  Fast unfolding of communities in large networks, Vincent D et al., Journal of Statistical Mechanics: Theory and Experiment(2008)

- 참고 사이트
  https://wikidocs.net/93410
  https://mons1220.tistory.com/129
  https://lsjsj92.tistory.com/587