Lab Study Report

배홍섭

INDEX

1. Introduction

01 | Introduction

- GNN (Graph Neural Networks)

Graph Neural Networks

Node-Edge로 구성된 그래프 데이터의 구조를 학습하는 딥러닝 모델

1.1. What is graph?

01 | What is graph?

Formally, a graph G = (V,E) is defined by a set of nodes V and a set of edges E between these nodes. We denote an edge going from node $u \in V$ to node $v \in V$ as $(u,v) \in E$.

01 | What is graph?

A convenient way to represent graphs is through an adjacency matrix $A \in R|V|\times|V|$.

To represent a graph with an adjacency matrix, we order the nodes in the graph so that every node indexes a particular row and column in the adjacency matrix.

01 | What is graph?

- Directed or Undirected

If the graph contains only undirected edges then A will be a symmetric matrix, but if the graph is directed (i.e., edge direction matters) then A will not necessarily be symmetric.

Some graphs can also have weighted edges, where the entries in the adjacency matrix are arbitrary real-values rather than {0,1}.

01 | What is graph?

- Multi-relational Graphs

여기에 수식을 입력하십시오.

We call such graphs multi-relational, and the entire graph can be summarized by an adjacency tensor $A \in R|V|\times|R|\times|V|$, where R is the set of relations.

Two important subsets of multi-relational graphs are often known as heterogeneous and multiplex graphs.

01 | What is graph?

- Heterogeneous graphs

meaning that we can partition the set of nodes into disjoint sets $V = V1 \cup V2 \cup ... \cup Vk$ where $Vi \cap Vj = \emptyset, \forall i \neq j$.

01 | What is graph?

- Multiplex graphs

In multiplex graphs we assume that the graph can be decomposed in a set of k layers.

Every node is assumed to belong to every layer, and each layer corresponds to a unique relation, representing the intra- layer edge type for that layer.
We also assume that inter-layer edges types can exist, which connect the same node across layers.

For instance, in a multiplex transportation network, each node might represent a city and each layer might represent a different mode of transportation (e.g., air travel or train travel).
Intra-layer edges would then represent cities that are connected by different modes of transportation, while inter-layer edges represent the possibility of switching modes of transportation within a particular city.

01 | What is graph?

- Feature Information

Lastly, in many cases we also have attribute or feature information associated with a graph.

Most often these are node-level attributes that we represent using a real-valued matrix $X \in R|V|\times m$

1.2. Machine learning on graphs

02| Machine learning on graphs

We seek to build models that can learn from data in order to solve particular tasks, and machine learning models are often categorized according to the type of task they seek to solve

In this section we provide a brief overview of the most important and well-studied machine learning tasks on graph data.

02| Machine learning on graphs

- Node classification

Suppose we are given a large social network dataset with millions of users, but we know that a significant number of these users are actually bots. Identifying these bots could be important for many reasons: a company might not want to advertise to bots or bots may actually be in violation of the social network's terms of service. Manually examining every user to determine if they are a bot would be prohibitively expensive, so ideally we would like to have a model that could classify users as a bot (or not) given only a small number of manually labeled examples.

This is a classic example of node classification, where the goal is to predict the label yu—which could be a type, category, or attribute—associated with all the nodes $u \in V$, when we are only given the true labels on a training set of nodes $V_{train} \subset V$. Node classification is perhaps the most popular machine learning task on graph data, especially in recent years.

02| Machine learning on graphs

- Node classification

At first glance, node classification appears to be a straightforward variation of standard supervised classification, but there are in fact important differences. The most important difference is that the nodes in a graph are not independent and identically distributed (i.i.d.)

Node classification completely breaks this i.i.d. assumption. Rather than modeling a set of i.i.d. datapoints, we are instead modeling an interconnected set of nodes.

02| Machine learning on graphs

- Node classification (example)

homophily, which is the tendency for nodes to share attributes with their neighbors in the graph.

For example, people tend to form friendships with others who share the same interests or demographics. Based on the notion of homophily we can build machine learning models that try to assign similar labels to neighboring nodes in a graph [Zhou et al., 2004].

Beyond homophily there are also concepts such as structural equivalence [Donnat et al., 2018], which is the idea that nodes with similar local neighborhood structures will have similar labels, as well as heterophily, which presumes that nodes will be preferentially connected to nodes with different labels.

02| Machine learning on graphs

- Relation prediction
  - Can we use machine learning to infer the edges between nodes in a graph?

Along with node classification, it is one of the more popular machine learning tasks with graph data and has countless real-world applications: recommending content to users in social platforms [Ying et al., 2018a], predicting drug side-effects [Zitnik et al., 2018], or inferring new facts in a relational databases [Bordes et al., 2013]—all of these tasks can be viewed as special cases of relation prediction.

02| Machine learning on graphs

- Relation prediction

The standard setup for relation prediction is that we are given a set of nodes V and an incomplete set of edges between these nodes Etrain ⊂ E. Our goal is to use this partial information to infer the missing edges E \ Etrain. The complexity of this task is highly dependent on the type of graph data we are examining. For instance, in simple graphs, such as social networks that only encode “friendship” relations, there are simple heuristics based on how many neighbors two nodes share that can achieve strong performance [L ̈ ̈u and Zhou, 2011]. On the other hand, in more complex multi-relational graph datasets, such as biomedical knowledge graphs that encode hundreds of different biological interactions, relation prediction can require complex reasoning and inference strategies [Nickel et al., 2016].

02| Machine learning on graphs

- Clustering and community detection

The standard setup for relation prediction is that we are given a set of nodes V and an incomplete set of edges between these nodes Etrain ⊂ E. Our goal is to use this partial information to infer the missing edges E \ Etrain. The complexity of this task is highly dependent on the type of graph data we are examining. For instance, in simple graphs, such as social networks that only encode “friendship” relations, there are simple heuristics based on how many neighbors two nodes share that can achieve strong performance [L ̈ ̈u and Zhou, 2011]. On the other hand, in more complex multi-relational graph datasets, such as biomedical knowledge graphs that encode hundreds of different biological interactions, relation prediction can require complex reasoning and inference strategies [Nickel et al., 2016].

02| Machine learning on graphs

- Graph classification, regression, and clustering
  - The final class of popular machine learning applications on graph data involve classification, regression, or clustering problems over entire graphs.

Of all the machine learning tasks on graphs, graph regression and classification are perhaps the most straightforward analogues of standard supervised learning.

In a similar way graph clustering is the straightforward extension of unsupervised clustering for graph data.

The challenge in these graph-level tasks, however, is how to define useful features that take into account the relational structure within each datapoint

2. Background and Traditional Approaches

01 Background and Traditional Approaches

제 2 장

1. 기본 그래프 통계, 커널 방법, 노드 및 그래프 분류 작업
2. 노드 이웃 간의 중첩을 측정하기 위한 다양한 접근방식 소개.
3. Graph Laplacian 을 사용한 스펙트럼 클러스터링에 대한소개.

2.1. Graph Statistics and Kernel Methods

01 | Graph Statistics and Kernel Methods

- 몇 가지 중요한 노드 수준 기능과 통계를 소개

2.1.1. Node-level statistics and features

01 Node-level statistics and features

- 이 소셜 네트워크는 Padgett와 Ansell[1993]의 작업으로 잘 알려져 있으며, 이 네트워크를 사용하여 피렌체 정치를 지배하게 된 Medici 가족 (중앙 근처에 묘사됨)의 권력 상승을 설명.
- 정치적 결혼은 메디치 시대에 권력을 공고히 하는 중요한 방법이었기 때문에 이 결혼 연결망은 이 시대의 정치 구조에 대해 많은 것을 담고 있음

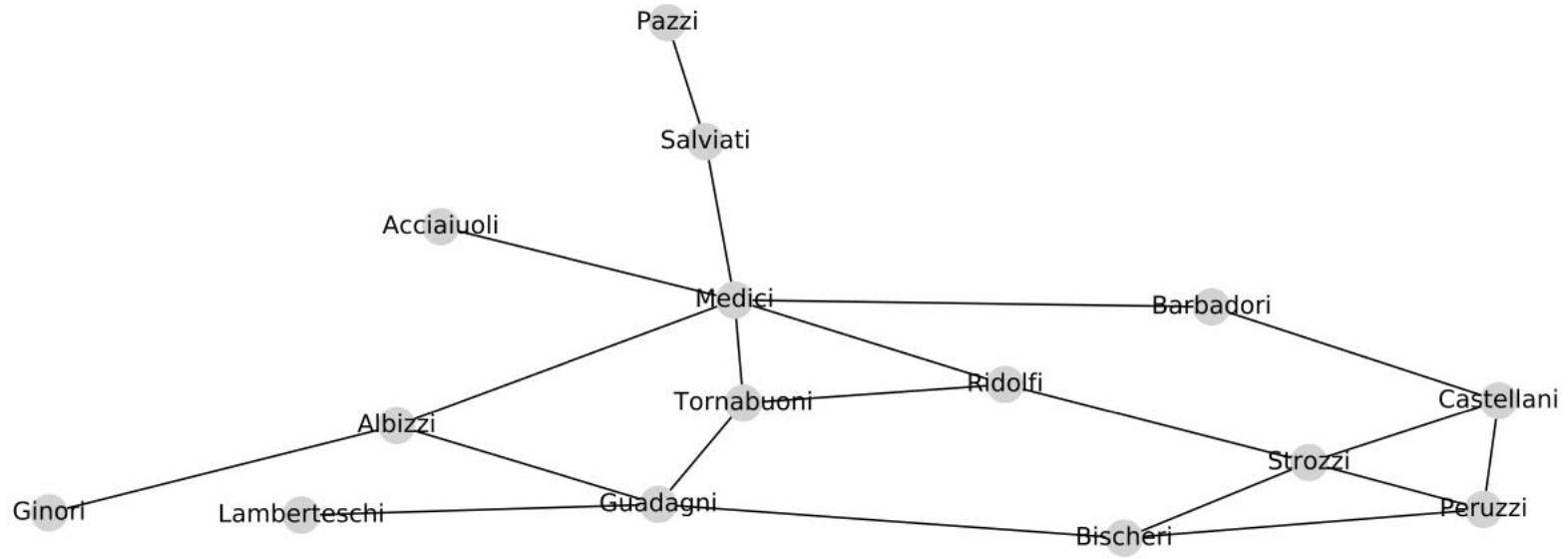


Figure 2.1: A visualization of the marriages between various different prominent families in 15th century Florence [Padgett and Ansell, 1993].

01 | Node-level statistics and features

- Node degree
  - 노드에 연결된 edges수로 정의됨.

$$d_u = \sum A[u, v], (v \in V)$$

- Node centrality
  - degree는 단순히 각 노드의 이웃 수를 측정하 는 반면 Centrality은 노드의 이웃이 얼마나 중요한지를 고려

$$e_u = \frac{1}{\lambda} \sum_{v \in V} A[u, v] e_v, \forall u \in V$$

$e : the\ vector\ of\ node\ centralities$
$\lambda : constant$
$A : adjacency\ matrix$

Rewriting

$$\lambda e = Ae$$

Eigenvector-eigenvalue equation!!

Node centrality는 인접행렬의 eigenvector에 해당한다.

01 | Node-level statistics and features

- The clustering coefficient $$c_u = \frac{|(v_1, v_2) \in \varepsilon : v_1, v_2 \in \mathrm{N}(u)|}{\frac{d_u}{2}}$$

  - 클러스터링 계수는 노드의 이웃이 얼마나 밀접하게 있는지 측정.
  - 클러스터 링 계수가 1이면 u의 모든 이웃도 서로의 이웃임을 의미.

2.1.2. Graph-level features and graph kernels

02| Graph-level features and graph kernels

- Bag of nodes
  - Graph level feature을 정의하는 가장 쉬운 방법은 node level의 통계를 집계하는 것.
  - 하지만 global 수준의 중요한 특성을 놓칠 수 있는 단점이 존재함.

- The Weisfeiler-Lehman kernel
  - interative neighborhood aggregation을 사용한 bag of nodes의 단점을 개선한 방법
  - Hash 함수를 사용하여 K번 re-labeling.

- Graphlets and path-based methods
  - 서로 다른 ‘graphlets’ 의 발생 횟수를 count(=단순 나열 enumerating)
  - challenge : counting graphlets is difficult!

- path-based methods.
  - not enumerating, but path-based methods.
  - 그래프에서 발생하는 다양한 종류의 경로를 조사
  - Graph data의 함정을 피하면서 많은 structural information 를 얻을 수 있는 방법.

2.2. Neighborhood Overlap Detection

02| Neighborhood Overlap Detection

소개된 다양한 접근방식들은 node나 graph level에서 유용하다.
그러나 node 간의 relation level(=edge)에는 한계가 있다.

$$S[u, v] = |N(u) \cap N(v)|$$

다음 수식은 두 node (u,v) 간의 공유하는 neighborhood의 수를 계산하여 관계를 수량화한 값을 나타낸다

2.2.1. Local overlap measures

02| Local overlap measures

- Node (u,v) 가 가지는 공통 neighborhood의 수, 정규화 방식에 따라 다양한 지표 존재.
- Sorensen index $$S_{sorenson}[u,v] = \frac{2|N(u) \cap N(v)|}{d_u + d_v}$$
- Salton index $$S_{Salton}[u,v] = \frac{2|N(u) \cap N(v)|}{\sqrt{d_u d_v}}$$
- Jaccard index $$S_{Jaccard}[u,v] = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

- salton 과 jaccard index는 한 node의 degree로 편향되는 것을 방지하기 위한 정규화 기법을 사용
- local neighbornood만 고려하기 때문에 제한적일 수 있음. -> global overlap 도 고려.

2.2.1. Global overlap measures

02| Global overlap measures

- Katz index $S_{Katz}[u,v] = \sum_{i=1}^{\infty} \beta^i A^i[u,v]$

  - 두 node (u,v) 사이의 모든 경로의 길이를 계산.
  - 이때 β는 경로 길이에 따라 부여되는 weight -> hyperparameter
  - $\beta < 1$이라면 long path의 중요성을 낮춘다.

그러나 node degree에 의해 편향될 수 있음. High-degree node일 수록 path 수가 많으므로 Katz index가 높게 측정된다.

- Leicht, Holme, and Newman (LHN) similarity

$$\frac{A^i}{\mathrm{E}[A^i]}$$

  - 기댓값으로 정규화를 함으로써 Katz index가 발생시킬 수 있는 bias를 회피한다.

02| Global overlap measures

- How we get the expectation score?

$$\mathrm{E}[A[u,v]] = \frac{d_u d_v}{2m}$$

  - 이때 m 은 그래프 안의 총 edge 수를 의미.

- 만약 path의 길이가 2 라면?

$$\mathrm{E}[A^2[u,v]]$$

  - node u 를 떠나는 edges. 그리고 각 edges가 node v에서 끝날 확률(=기댓값)

$$\mathrm{E}[A^2[v1,v2]]$$

$$\mathrm{E}[A^2[v_1,v_2]] = \frac{d_{v1} d_{v2}}{(2m)^2} \sum_{u \in V} (d_u - 1) d_u.$$

  - 위와 같이 계산해낼 수 있다.

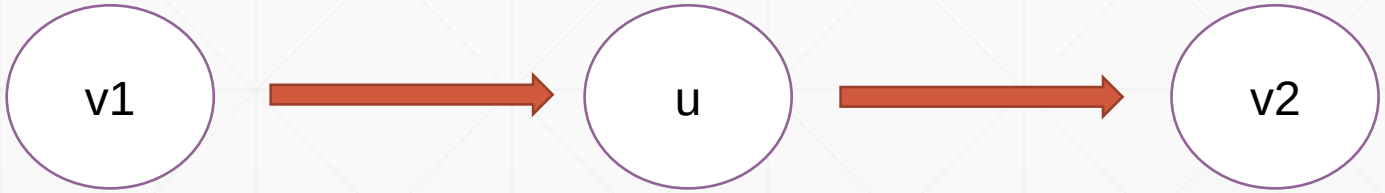
02| Global overlap measures

- Node v1을 떠나 node u를 지날 확률(기댓값) : $\frac{d_{v1}d_u}{2m}$
- Node u를 떠나 node v2에 도착할 확률(기댓값) : $\frac{d_{v2}(d_u-1)}{2m}$

- v2에 들어오는 u의 edges 중 v1에서 들어왔던거는 이미 사용했기 때문에 $d_u - 1$

둘이 동시에 일어나야하므로 곱 진행.

$$\mathrm{E}[A^2[v_1, v_2]] = \frac{d_{v1}d_{v2}}{(2m)^2} \sum_{u \in V}(d_u - 1)d_u.$$



Challenge : 3 이상의 경우 계산이 매우 복잡하다.

02| Global overlap measures

- 3이상의 거리
  - Leicht et al. [2006] 에 의하여 largest eigenvalue가 number of paths에 근사할 수 있다는 사실을 사용.

$$Ap_i = \lambda_1 p_{i-1}$$

  - 이때 p 는 node u와 다른 모든 노드 사이의 length(i) 길이의 path 수 이다.
  - λ_1은 A의 가장 큰 가장 큰 $eigenvalue$.

$$\mathrm{E}\left[A^i[u,v]\right] = \frac{d_u d_v \lambda^{i-1}}{2m}$$

  - i가 증가할 수록 λ_1만큼 증가. 길이 i 인 path의 기댓값은 λ_1 의 i-1 제곱배 만큼 증가.

02| Global overlap measures

따라서 Katz의 문제점을 정규화 함으로써 bias를 회피한 LNH index를 얻을 수 있다.

$S_{Katz}[u,v] = \sum_{i=1}^{\infty} \beta^i A^i[u,v]$ 에 A^i 대신 $\frac{A^i}{\mathrm{E}[A^i]}$ 대입

이때 $\mathrm{E}[A^i] = \mathrm{E}\left[A^i[u,v]\right] = \frac{d_u d_v \lambda^{i-1}}{2m}$

최종 식 :

$$S_{LNH}[u,v] = I[u,v] + \frac{2m}{d_u d_v} \sum_{i=0}^{\infty} \beta^i \lambda_1^{1-i} A^i[u,v]$$

2.3. Graph Laplacians and Spectral Methods

02| Graph Laplacians and Spectral Methods

- 그래프를 나타내는 데 사용할 수 있는 몇 가지 중요한 행렬의 정의
- 스펙트럼 그래프 이론의 기초에 대한 간략한 소개

2.3.1. Graph Laplacians

02| Graph Laplacians

- Laplacians 이란?
  - 인접행렬의 다양한 변환으로 표현된 행렬

- 종류 : Unnormalized Laplacian / Normalized Laplacian
  - Unnormalized Laplacian : $L = D - A$ (A is Adjacency ,D is degree matrix)
  - Normalized Laplacian : symmetric normalized Laplacian / random walk Laplacian
    - Symmetric normalized Laplacian : $L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
    - Random walk Laplacian : $L_{RW} = D^{-1}$

    두 normalized Laplacian의 차이점은 정규화 과정에서 상수가 다르다.

02| Graph Laplacians

- Laplacian matrix(L)의 특성
  1. L은 대칭행렬이다.
  2. 모든 |V|차원 벡터 x에 대해서 다음 식을 만족한다.
     - $x^T Lx = \frac{1}{2}\sum_{u\in V}\sum_{v\in V}A[u,v](x[u]-x[v])^2 = \sum_{(u,v)\in\varepsilon}(x[u]-x[v])^2$
  3. L은 0 이상의 (양수인) 고유값을 가진다.

- The Laplacian and connected components
  - Theorem 2. : Laplacian(L) 에서 0 인 고유값(eigenvalue)의 기하학적 중요도는 그래프에서 연결된 components의 수에 대응한다.
    - Eigenvalue-Eigenvector 방정식에 의해 0의 고유값에 대응하는 모든 고유벡터(e)는 다음과 같이 정의할 수 있다.

$$e^T Le = 0$$

참고 : https://darkpgmr.tistory.com/105

02| Graph Laplacians

$$e^T L e = 0$$

이때 e는 eigenvector, L은 Unnormalized Laplacian.
L = D – A 로 인해 L은 대칭행렬이며 n x n 정방행렬이다. (A가 인접행렬이므로)

$$Av = \lambda v$$

eigenvector-eigenvalue equation인데 A는 n x n 정방행렬, v는 eigenvector, λ는 eigenvalue이다.
양 변에 v의 역행렬 v^{-1}을 곱해준다.

$$v^{-1} A v = v^{-1} \lambda v$$

이때 A는 L, v는 e 대입. 이때 L은 대칭행렬이므로 v는 직교행렬이다.

$Theorem. A$ 행렬이 대칭이면 서로 다른 고유공간$(eigenspace)$에 있는 어떤 두 고유 벡터는 직교합니다.

따라서 $v^{-1} = v^T$.

$$e^T L e = e^T \lambda e$$

이때 L은 0의 고유값에 대한 행렬이므로 $\lambda = 0$

$$e^T L e = 0$$

참고 : https://darkpgmr.tistory.com/105, https://deep-learning-study.tistory.com/414

02| Graph Laplacians

- Laplacian matrix(L)의 특성
  1. L은 대칭행렬이다.
  2. 모든 |v|차원 벡터 x에 대해서 다음 식을 만족한다.
     - $x^T Lx = \frac{1}{2}\sum_{u\in V}\sum_{v\in V} A[u,v](x[u]-x[v])^2 = \sum_{(u,v)\in\varepsilon}(x[u]-x[v])^2$
  3. L은 0 이상의 (양수인) 고유값을 가진다.

$$e^T Le = 0$$

Laplacian matri의 2번째 특성에 의해 $e^T Le = \sum_{(u,v)\epsilon\varepsilon}(e[u] - e[v])^2 = 0$
이는 곧 $e[u] = e[v]$를 의미.
즉 동일한 components에 존재하는, connected된 모든 node u 는 같은 상수 값인 $eigenvector\ e[u]$를 가진다.

예시) fully-connected graph, eigenvalue = 0
모든 node가 서로 connected라면, 동일한 components 내에서는 모든 node u는 같은 상수 값인 eigenvector(one vector)를 가진다. 이 경우는 방정식의 해가 하나이다.

참고 : https://darkpgmr.tistory.com/105, https://deep-learning-study.tistory.com/414

02| Graph Laplacians

그래프가 동일한 components가 아닌 multiple connected components일 경우
- 단, 각 components와 연결된 Laplacian blocks는 서로 독립적이다.

$$L = \begin{bmatrix} L_1 & & \\ & \ddots & \\ & & L_k \end{bmatrix}$$

L 은 대각행렬이므로 L의 spectrum은 모든 L_k의 spectrum의 합집합이다.

L_k는 fully connected된 subgraph.
- 따라서 앞서 동일한 components에서 fully-connected 예시와 같이 L_k는 0 eigenvalue와 1 eigenvector를 가진다고 볼 수 있다.

L의 eigenvalue는 L_k 의 eigenvalue의 합집합이며 L의 eigenvector는 L_k의 eigenvector의 합집합이다.

참고 : https://darkpgmr.tistory.com/105, https://deep-learning-study.tistory.com/414

2.3.2. Graph Cuts and Clustering

02| Graph Cuts and Clustering

- Cut : cutvalue를 측정하여 minimizes하는 방향으로 k개의 cluste를 생성(=cut 진행)
  - Cutvalue가 최소가 되었을 때 optimal clustering

- A를 graph속 node들의 subset, $\overline{A}$ 를 A의 여집합이라 하자.

$$Let\ A \subset V\ denote\ a\ subset\ of\ the\ nodes\ in\ the\ graph\ .$$
$$Let\ A^{-}\ denote\ the\ complement\ of\ this\ set.\ \ (A \cup A^{-} = V, A \cap \overline{A} = \emptyset)$$

$A_1 A_2 \ldots A_k$를 겹치지 않는 여러 개의 subset이라고 할 때, cutvalue는 다음과 같이 정의한다.

$$cut(A_1, A_2, \ldots, A_k) = \frac{1}{2}\sum_{k=1}^{K} |(u, v) \in \varepsilon : u \in A_k, v \in \overline{A}_k|$$

02| Graph Cuts and Clustering

$$cut(A_1, A_2, \dots, A_k) = \frac{1}{2}\sum_{k=1}^{K}|(u,v) \in \varepsilon : u \in A_k, v \in \overline{A}_k|$$

정규화 기법에 따라 RatioCut / Ncut으로 나눈다.

1. RatioCut : $cut(A_1, A_2, \dots, A_k) = \frac{1}{2}\sum_{k=1}^{K}\frac{|(u,v)\in \varepsilon : u\in A_k, v\in \overline{A}_k|}{|\overline{A}_k|}$
   - 단, 여집합의 크기로 정규화를 진행하기 때문에 작은 크기의 cluster를 고를 때는 좋지 않다.
     (cluste의 크기가 작다면 여집합의 크기가 상대적으로 크기 때문에 정규화가 너무 많이 진행됨)
2. Ncut : $cut(A_1, A_2, \dots, A_k) = \frac{1}{2}\sum_{k=1}^{K}\frac{|(u,v)\in \varepsilon : u\in A_k, v\in \overline{A}_k|}{vol(A_k)}$
   - $vol(A_k) = \sum_{u\in A} d_u$

02| Approximating the cutvalue with the Laplacian spectrum

$If\ k = 2$, 목적식 : $\min_{A \in V} RatioCut(A, \overline{A})$

$$a[u] = \begin{cases} \sqrt{|\frac{\overline{A}}{A}|} & if\ u \in A \\ -\sqrt{|\frac{A}{\overline{A}}|} & if\ u \in \overline{A} \end{cases}$$ 라고 하자.

앞서 살펴본 수식에 대입하면 $a^T La = \sum_{(u,v)\epsilon\varepsilon} (a[u] - a[v])^2$

$$= \sum_{(u,v) \in \varepsilon : u \in A, v \in \overline{A}} \left(\sqrt{\frac{|\overline{A}|}{|A|}} - \left(-\sqrt{\frac{|A|}{|\overline{A}|}}\right)\right)^2$$

하지만 NP-hard problem 발생. -> Rayleigh-Ritz Theorem 사용.

02| Approximating the cutvalue with the Laplacian spectrum

Rayleigh-Ritz Theorem

- NP-hard problem의 해결을 위해 second-smallest eigenvector L을 사용한다.

$a[u]$ 값에 근거하여 node를 cluster에 할당.

$$\begin{cases} u \in A & if\ a[u] \geq 0 \\ u \in \overline{A} & if\ a[u] < 0 \end{cases}$$

따라서 second-smallest eigenvector L으로 최적의 cluster 할당을 진행.

2.3.3. Generalized spectral clustering

02| Generalized spectral clustering

앞서 if k=2 대신 K개의 cluster로 확장 가능
- if k=2 : second-smallest eigenvector
- if k=K : K-smallest eigenvector

1. Find the K smallest eigenvectors of L (excluding the smallest):
   - $e_{|v|-1}, e_{|v|-2}, e_{\ldots} e_{|v|-K}$
2. 1단계에서 찾은 eigenvector 를 column으로 하는 행렬 U를 생성.
   - $U \in \mathbb{R}^{|v| \, x \, (k-1)}$
3. 각각의 node는 U 행렬의 row로 표현됨.
   - $Z_u = U[u]$
4. Embedding Z_u 에 의해 K-mean clustering 진행.

2.4. Towards Learned Representations

2.4. Towards Learned Representations

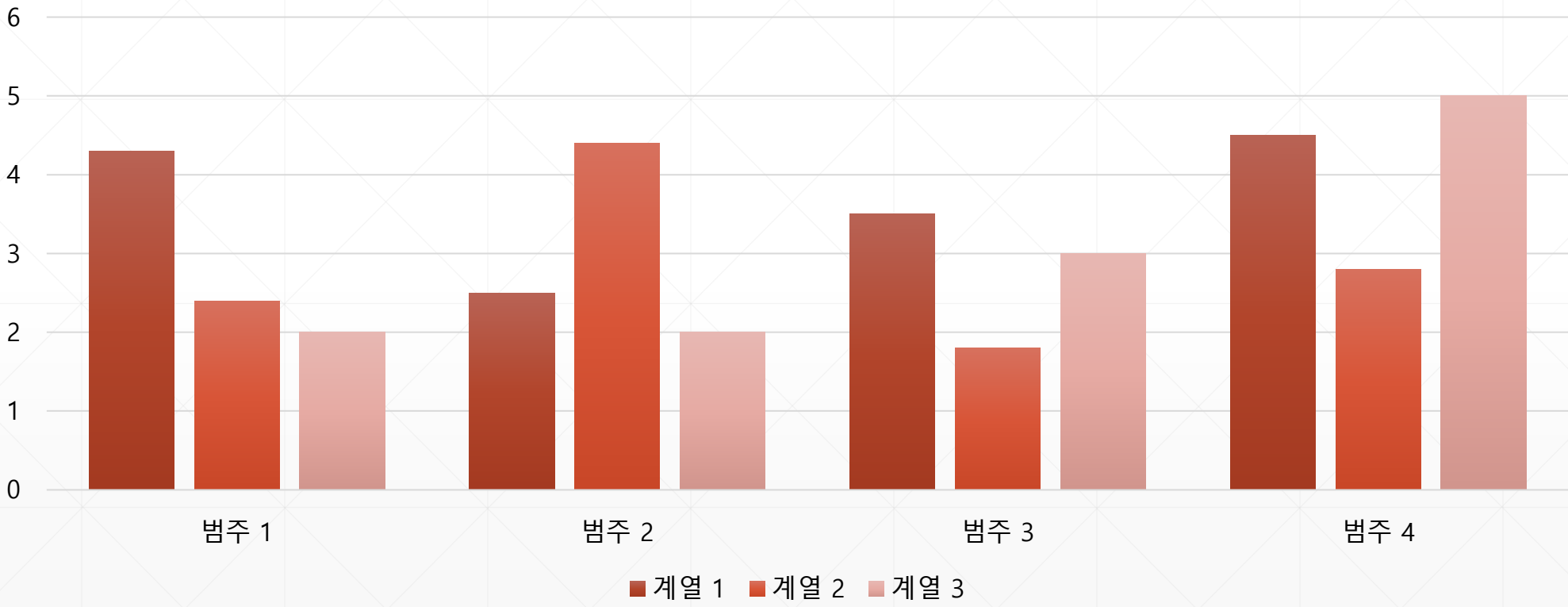
Breif conclusion : $hand-engineered\ statistics\ and\ measures.$

- ✓ 그래프의 전통적인 방식을 알아봄
- ✓ Classification task를 위해 어떻게 feature extrac하는지 알아봄
- ✓ Relation predict를 위해 neighborhood overlap를 알아봄
- ✓ K가 2이상 spectral approach에 대해 알아봄
- ✓ time-consuming and expensive process 문제점 발생

따라서 이후 챕터에서는 다른 대안 접근방식이 소개됨

- $graph\ representation\ learning.$
- not extract hand-engineered features, but ‘learn’ representation

차트를 사용한 제목 및 내용 레이아웃



표를 사용한 두 개의 내용 레이아웃

- 여기에 첫 번째 글머리 기호
- 여기에 두 번째 글머리 기호
- 여기에 세 번째 글머리 기호

클래스	그룹 1	그룹 2
클래스 1	82	95
클래스 2	76	88
클래스 3	84	90

SmartArt가 있는 제목 및 내용 레이아웃

슬라이드 제목 추가 - 3

슬라이드 제목 추가 - 4

슬라이드 제목 추가 - 5