최 종 연 구 보 고 서

라이프로그 그래프 데이터로부터 생활패턴 발견기술 연구

연세대학교

한 국 전 자 통 신 연 구 원

최 종 연 구 보 고 서

라이프로그 그래프 데이터로부터 생활패턴 발견기술 연구

연세대학교

한 국 전 자 통 신 연 구 원

제 출 문

한국전자통신연구원장 귀하

본 보고서를 라이프로그 그래프 데이터로부터 생활패턴 발견 기술 연구 의 최종연구보고서로 제출합니다.

2023년 11월 30일

위탁연구개발기관 : 연세대학교

위탁연구개발기관장 : 김지현 (인)

연 구 책 임 자 : 연세대학교 컴퓨터과학과 조성배
참 여 연 구 원 : 연세대학교 인공지능학과 박노성
참 여 연 구 원 : 연세대학교 컴퓨터과학과 문형준
참 여 연 구 원 : 연세대학교 인공지능학과 최정환
참 여 연 구 원 : 연세대학교 인공지능학과 김자영
참 여 연 구 원 : 연세대학교 인공지능학과 신예진
참 여 연 구 원 : 연세대학교 인공지능학과 위효원
참 여 연 구 원 : 연세대학교 인공지능학과 이채정
참 여 연 구 원 : 연세대학교 인공지능학과 임학수
참 여 연 구 원 : 연세대학교 인공지능학과 전진성
참 여 연 구 원 : 연세대학교 인공지능학과 조우진
참 여 연 구 원 : 연세대학교 인공지능학과 진서연
참 여 연 구 원 : 연세대학교 인공지능학과 안민혁
참 여 연 구 원 : 연세대학교 인공지능학과 임서하
참 여 연 구 원 : 연세대학교 컴퓨터과학과 Gatum Erlangga

요 약 문

1. 제목 : 라이프로그 그래프 데이터로부터 생활패턴 발견기술 연구

2. 연구의 목적 및 중요성

1) 연구의 목적

개인의 일상생활을 수집한 그래프 라이프로그로부터 일상생활 패턴을 분류/예측하는 기술 및 개인화된 그래프 필터/컨볼루션 기반으로 추천하는 기술 고도화

2) 연구의 중요성

스마트 기기의 발전과 소셜 미디어의 대중화 및 인공지능 기술의 발전으로 사용자 경험 상황에서 의미 있는 패턴을 추출하고 이를 이용한 다양한 개인 맞춤형 서비스가 이용되고 있다. 이러한 개인 맞춤형 서비스는 미래 산업으로서 대규모 시장가치 창출이 가능할 것으로 알려졌다. 기존의 데이터베이스 구조를 기반으로 하는 라이프로그 데이터를 기반으로 개인 맞춤형 서비스를 구현할 경우 데이터 간 의미추론이 불가능하여 형식적인 질의만 가능하다. 또한 데이터 간 관계를 형성하기 위해서는 테이블을 조인해야 하고 복잡한 질의를 수행하기 위해서는 많은 양의 연산이 필요하다. 따라서 계산 복잡도 측면의 어려움을 극복하기 위해 인스턴스 간의 관계를 나타낼 수 있는 그래프 형태의 데이터를 기반으로 하는 맞춤형 서비스에 대한 필요성이 증대되었고, 라이프로그 데이터를 지식 그래프, 시맨틱 그래프 구조로 나타내어 개인의 일상을 모델링하고자 하는 연구가 특히 중요하다.

그래프 구조를 이용한 사용자 맞춤 기술은 노드와 엣지로 연결된 저장 구조를 이용하여 상호연관된 정보를 추출하여 패턴을 분석할 수 있으며 사용자에게 상호작용 서비스를 제공할 수 있다. 이런 장점에도 불구하고 그래프 데이터의 특성에서 기인한 문제가 있다. 그래프 데이터의 복잡성, 가변적인 그래프의 크기, 그래프마다 다른 근접 이웃의 존재는 그래프의 상호관계 메타정보 표현의 장점을 살려주는 동시에 모델링이 어려워진다. 그리고 입력 데이터의 각 인스턴스

들이 독립적이라고 가정한 후 학습하는 기존의 인공지능 모델은 그래프 데이터의 활용을 어렵게 하므로 그래프 데이터에 최적화된 인공지능 모델과 그에 따른 알고리즘 연구가 필요하다는 것을 시사한다. 특히 그래프형 데이터를 임베딩하고 이를 인공지능 기술에 접목하는 것과 함께, 그래프 임베딩 연구를 기반으로 라이프로그 데이터에서 생활패턴을 발견하는 연구로 확장하는 것이 필요하다.

기존 그래프 임베딩 방법에서 전체 그래프에서 발견할 패턴을 찾을 수 있는 정보는 매우 제한적이기 때문에 다양한 임베딩 기술이 적용되고 있지만, 대부분 그래프 데이터의 특성에 맞는 임베딩 방법이라고 보기 어렵다. 노드, 엣지 등 임베딩 하고자 하는 정보에 따라 최적의 방법이 없기 때문에, 많은 연구자들은 언어나 이미지 처리에서 활용되는 임베딩 방법을 그래프 데이터의 특성에 맞게 적용한다. 이런 추세에 따라 서브 그래프 특징추출을 좀 더 수월하게 하기 위한 다단계 임베딩 방법과, 이를 통해 인과적으로 그래프 분류가 가능한 새로운 분류 모델을 고안한다. 또한, 기존의 추천 시스템들이 사용자의 취향을 정확히 반영하지 못하는 문제를 해결하기 위해서 그래프 신호처리 이론을 활용한 추천 방법을 개발한다. 그래프 신호처리 이론은 그래프 내에서 노드 간의 상호작용을 분석하고, 이를 주파수 영역에서 처리하여 정확한 정보를 추출할 수 있는 방법을 제공한다. 이러한 방법을 활용하여 추천 시스템을 개발하면, 기존의 시스템보다 더 정확한 추천이 가능하다. 이 방법은 대규모 그래프 데이터를 처리하는 데 있어서 높은 성능을 보장할 수 있으며, 추천 시스템 분야에서 더욱 정확하고 효과적인 결과를 제공할 수 있을 것이다.

그래프 필터 기반 추천 시스템은 사용자의 관심사를 파악하고 이에 따라 추천하는 방식으로 작동한다. 이러한 기술은 현재 많은 연구가 이루어지고 있으며, 그중에서도 주파수 이론을 이용하여 개인화된 그래프 필터 기반 추천 시스템의 연구가 중요하다. 기존의 그래프 필터 기반 추천 시스템은 대부분 주파수를 기반으로 한 필터를 사용하지만, 이는 모든 사용자에게 동일한 필터를 적용하는 것이기 때문에 각 사용자의 선호도와 같은 개인적인 특성을 고려하지 못하는 문제가 있다. 이에 따라 주파수 이론을 이용하여 개인화된 필터를 적용할 수 있는 연구가 필요하다. 주파수 이론을 이용하면 각 사용자의 선호도와 같은 개인적인 특성을 반영하여 그래프 신호를 처리할 수 있다. 이를 통해 더욱 정확한 개인화된 추천을 제공할 수 있어, 사용자 경험을 향상시킬 것이다.

3. 연구의 내용 및 범위

가. 일상생활 패턴 발견을 위한 그래프 분할 기반 그래프 컨볼루션 신경망의 고도화

- 의미 구조 반영을 위한 그래프 분할 알고리즘 고도화
- 분할 그래프 앙상블을 통한 고성능 그래프 분류 및 예측 알고리즘 개발

나. 그래프 기반 추천 알고리즘 동향 분석 및 조사

- 개인화를 위한 그래프 기반 추천 알고리즘의 체계적인 문헌 조사 및 분석
- 그래프 대조 학습 기반 추천 알고리즘의 체계적인 문헌 조사 및 분석

다. 개인화된 그래프 필터 기반 추천 알고리즘 개발

- 개인화를 위한 그래프 필터 기반 추천 알고리즘 동향 및 조사
- 그래프 대조 학습 기반 개인화된 추천 알고리즘 개발

라. 개발된 알고리즘의 유용성 평가를 위한 프로토타입 개발

- 분할 그래프로부터 의미 구조 반영 패턴 발견 시스템의 프로토타입 구현
- 다양한 시나리오와 척도에 따른 패턴 발견 성능 평가
- 개인화된 그래프 필터 기반 추천 알고리즘 모델 성능 평가

4. 연구 결과

연구결과물 목록(최초 계획한 연구결과물과 다를 경우 비고에 사유 기재 必)

결과물유형	결과물 명칭	규격	수량	제출일	비고
중간 발표회 발표자료	중간진도점검 결과	발표 자료	1	2023. 08. 22	
보고서 인쇄본	라이프로그 상세패턴 발견 및 개인화된 추천 시스템을 위한 연구 최종보고서	보고서	10	2023. 11. 30	
연구논문	라이프로그 상세패턴 발견 및 개인화된 추천 시스템을 위한 연구논문	논문	5	2023. 11. 30	
	해외 학술대회 논문	논문	5	2023. 11. 30	

※ 결과물 유형에는 지식재산권 문서, TM/TDP, 연구논문, 기고서, SW program, 회로도, 연구수행 데이터 등 도출 가능한 연구결과를 모두 기재

5. 활용에 대한 건의

가. 사용자 경험의 구조화 방법과 생활패턴의 추출을 위한 딥러닝 기술 기반 그래프 패턴 발견을 수행하는 원천기술 제공

나. 그래프 형식의 대용량 사용자 데이터 내부의 패턴 발견을 위한 원천기술 제공

다. 사용자의 발견된 패턴으로부터 하루 일과 요약 및 사용자 행동 패턴 기반 추천을 위한 원천기술 제공

6. 기대 효과

가. 그래프 임베딩 방식은 사용자의 정서, 행위, 상황에 대한 경험상황 학습모델의 구현 시 활용 가능

나. 그래프 구조의 라이프로그에 기반한 개인화 서비스를 위해서 사용자 생활패턴 정보 제공 시 활용 가능

다. 그래프 내 발견된 패턴별 상황과 추천 알고리즘 기술 성능과의 연관관계 분석 가능

라. 사용자의 행동 패턴에 기반하여 추천 모델의 성능 향상 가능

− 목 차 −

제 1 장. 서론

제 1 절. 연구의 개요

1. 상세패턴 발견을 위한 라이프로그 그래프의 분할 기술 고도화

라이프로그 데이터는 일상생활에서 개인의 행동을 스마트 기기나 센서로 수집한 정보를 의미한다. 이 데이터를 활용하여 기업들은 개인화된 추천 시스템을 개발하고 있다. 예를 들어, 넷플릭스는 사용자가 시청한 콘텐츠, 검색 기록, 평점 등의 라이프로그 데이터를 기반으로 맞춤형 추천을 제공한다. 이를 통해 사용자는 선호하는 콘텐츠를 쉽게 찾아볼 수 있으며, 기업은 고객 만족도와 구독률을 향상시킨다. 쿠팡은 사용자의 검색어, 클릭한 상품, 구매 내역 등을 수집해 개인 맞춤형 상품 추천 서비스를 제공한다. 이를 통해 사용자는 원하는 상품을 보다 효율적으로 찾아 구매율과 매출을 증가시킨다.

최근에는 스마트 기기와 소셜 미디어에서 수집한 데이터를 활용해 개인의 일상과 행동 패턴을 분석하는 빅데이터 구축이 활발하게 진행되고 있다. 특히, 채팅 기록, 접속 기록, 구매 기록 등의 라이프로그 데이터를 지식 그래프나 시멘틱 그래프 구조로 모델링하는 연구가 주목받고 있다. 이러한 그래프 형식의 데이터는 노드와 엣지를 활용해 비유클리드 공간에서의 관계와 메타정보를 표현할 수 있으며, 이는 기존 데이터베이스 구조와 다른 접근방식이다.

기존 데이터베이스 구조를 사용하는 경우, 데이터 간 의미추론이 제한적이고 형식적인 질의에만 응답한다. 또한 데이터 중복이 자주 발생하고 복잡한 질의가 어렵다. 반면 그래프로 표현된 라이프로그 데이터는 노드 간 다양한 의미 관계를 포함할 수 있어 패턴 탐지 연구에 유리하다. 이 분야에서는 RecGNNs, ConvGNNs, GAEs 등 다양한 방법으로 연구가 진행되고 있으며, 각 방법은 사용되는 모델에 따라 구분된다.

Category	Publications
RecGNNs	Graph Recurrent Neural Networks
	RecGNNs using Graph Echo State
	LSTM based RecGNNs
	Learning Steady-states of Iterative Algorithm
ConvGNNs	Locally Convolutional Neural Networks
	Deep Convolutional Neural Networks
	Contextual Convolutional Neural Networks
	Diffusion Convolutional Neural Networks
GAEs	Unsupervised Graph Embedding with Autoencoder
	Structural Deep Embedding
	Generative Model of Graphs
	Deep Generative Model of Graphs

표 1. Graph Neural Networks (GNNs) 관련 연구

그래프 데이터 처리 분야의 주요 어려움은 그래프의 고유한 특성에서 비롯된다. 복잡한 구조, 변동하는 크기, 그리고 각 그래프마다 다른 이웃 구조로 인해 전통적인 탐색 알고리즘에서 계산 복잡성이 증가한다. 기존 머신러닝 알고리즘은 대부분 인스턴스들이 독립적이라는 가정을 바탕으로 하기 때문에, 서로 연결된 그래프 데이터에 적용하기 어려운 측면이 있다. 이러한 문제를 해결하기 위해 그래프 데이터에 최적화된 AI 모델과 알고리즘 개발이 필요하다.

그래프 임베딩은 이러한 문제를 해결하는 방법 중 하나로, 그래프 데이터를 벡터 형태로 변환해 인공지능 모델 학습에 용이하게 만든다. 전통적인 그래프에서 의미 구조를 탐색하는 것이 어렵기 때문에, 임베딩을 통해 전체 그래프를 벡터 공간에서 탐색한다. 이때 중요한 패턴 정보는 종종 매우 국소적이다. 이를 해결하기 위해 다양한 해상도의 그래프를 각각 임베딩하고, 이들을 통합하여 다양한 관점에서 해석하는 접근방식이 제안된다.

그래프 임베딩은 라이프로그 데이터를 압축된 벡터로 표현하는 기술이지만, 실제 문제 해결을 위해서는 임베딩 모델과 함께 목적에 맞는 추가 모델이 필요하다. 특히, 라이프로그 데이터에 최적화된 분류 모델 개발이 중요하다. 개발될 그래프 분류 모델은 라이프로그 데이터를 입력으로 받아, 해당 데이터의 패턴 종류를 인과적 절차에 따라 분류한다.

그래프 기반 머신러닝의 발전과 함께, 그래프 신경망(Graph Neural Networks, GNNs)과 같은 새로운 기법들도 등장하고 있다. GNNs는 그래프 구조의 데이터를 처리하기 위해 설계된 신경망으로, 노드 간의 관계와 그 구조를 학습한다. 이는 그래프 데이터의 복잡한 관계와 동적인 특성을 반영할 수 있는 강력한 도구로 평가되며, 사회 네트워크 분석, 분자 구조 예측, 추천 시스템 등 다양한 분야에서 활용되고 있다. GNNs와 임베딩 기술의 결합은 그래프 데이터 처리 분야의 효율성과 정확도를 높이는 중요한 발전으로 볼 수 있다.

2. 개인화된 그래프 기반 추천 알고리즘 개발

추천 시스템은 사용자의 선호도와 관심사를 이해하고, 그에 따라 상품, 서비스, 정보 등을 추천하는 데 활용됩니다. 이러한 추천 시스템은 온라인 쇼핑, 음악 스트리밍, 영화 추천, 소셜 미디어 피드 정렬 등 다양한 영역에서 사용되고 있으며, 사용자에게 맞춤형 콘텐츠를 제공함으로써 기업과 사용자 모두에게 혜택을 제공한다.

그러나 현실 세계에서의 데이터는 복잡하며, 그래프 형태의 데이터로 표현되는 경우가 많다. 라이프로그 데이터는 다양한 사건과 상호작용의 네트워크를 나타내며, 이러한 데이터에서 추천 시스템을 구축하는 것은 도전적이다. 라이프로그 데이터는 사용자 간의 상호작용, 상품 및 서비스 간의 연결, 이벤트 시퀀스 등을 표현하고 있으며, 이러한 정보를 이용하여 사용자에게 맞춤형 추천을 제공하는 것은 더 복잡한 모델링과 분석을 요구한다.

이에 따라, 그래프 기반 추천 시스템의 연구가 라이프로그 데이터에 대한 이해를 높이고 사용자에게 더 나은 추천을 제공하는 데 필수적이다. 그래프 기반 추천 시스템은 그래프의 노드 및 엣지 정보를 활용하여 사용자와 상품 또는 서비스 간의 상호작용을 모델링하고, 이를 통해 사용자에게 맞춤형 추천을 생성할 수 있다. 이러한 연구는 라이프로그 데이터에 대한 효과적인 분석 및 예측 방법을 개발하고, 사용자 경험을 개선하며 기업의 비즈니스 성과를 향상시키는 데 이바지할 것으로 기대된다.

한편, 그래프 데이터의 다른 중요한 측면은 노드 분류이다. 노드 분류는 그래프 내의 노드를 여러 카테고리로 분류하거나 예측하는 작업을 의미하며, 예를 들어 소셜 미디어 그래프에서 사용자의 관심 주제를 예측하는 데 활용된다. 그래프 신경망은 이러한 노드 분류 문제에 효과적으로 적

용될 수 있는 도구로 간주되고 있다.

본 연구에서는 라이프로그 그래프 패턴 기반 행동 추천 시스템과 노드 분류를 위한 그래프 신경망을 고도화하며 이들을 고도화하기 위한 방법들을 논의한다. 이 연구는 다양한 응용 분야에서 라이프로그 데이터의 활용을 개선하고, 사용자 경험을 개선하며, 비즈니스 성과를 향상시키는 데 기여할 것으로 기대한다.

제 2 절. 연구개발 목표 및 내용

1. 일상생활 패턴 발견을 위한 그래프 분할 기반 그래프 컨볼루션 신경망의 고도화

그래프 임베딩은 인공지능 모델 학습을 위해 그래프 데이터를 표현형 벡터 데이터로 표현하는 것으로 정의한다. 그래프 임베딩 알고리즘의 목적은 그래프 구조를 최대한 유지하면서 저차원의 벡터로 나타내는 것으로 그래프의 노드, 엣지, 경로, 서브 그래프 혹은 전체 그래프를 저차원 벡터로 나타낼 수 있고 이 벡터는 훈련 그래프 데이터의 피처를 나타내는 동시에 다른 모델의 학습에 활용된다.

최근 인공지능 및 머신러닝 기술이 그래프 임베딩 방법론으로 떠오르게 되면서 그래프 임베딩 성능이 향상되었다. 대표적으로 합성곱 레이어(convolutional layer)를 활용한 그래프 임베딩 방법은 2차원 데이터를 처리하는 딥러닝 모델에서 많이 활용하고 있으며, 데이터로부터 입력 벡터 사이의 국소적인 상관관계를 모델링할 수 있는 필터를 학습하는 신경망 층으로 요약한다. 크기를 조절하고 이동함으로써 입력의 저차원 피처부터 고차원 피처까지 추출한다. 대표적인 사용 예로 합성곱 레이어와 풀링 레이어를 활용한 그래프 임베딩 모델이 있다. 모델의 입력으로 들어간 그래프 데이터는 여러 겹의 합성곱 레이어와 풀링 레이어를 거쳐 입력 그래프 데이터의 피처를 표현하는 저차원의 벡터로 임베딩 되며 모델의 분류 판단의 근거로 사용된다.

합성곱 레이어를 활용한 임베딩 과정은 그래프의 마스킹 사이즈에 맞는 각 서브 그래프의 피처를 추출하고 취합하는 과정이다. 각 노드의 인접 이웃이 균일하게 구성된 이미지에서는 마스킹 사이즈에 맞는 일부 이미지의 피처를 추출하고 취합하는 것이 효과적일 수 있어도 지식 그래프, 시멘틱 그래프와 같이 관계에 따라 노드 간 이웃이 달라지고 노드와 엣지의 가중치가 가변적인 특성을 띠는 동적인 데이터에서는 합성곱 레이어에서 적절한 서브 그래프를 생성하지 않는 문제를 내포한다. 따라서 그래프 임베딩 과정 중 적절한 서브 그래프 선택을 위한 다단계 및 계층적 그래프 임베딩 방식이 필요하며, 이를 통해 적절한 임베딩 벡터 생성 및 라이프로그 그래프 패턴 탐지 성능을 향상시킬 수 있다.

그래프 생활패턴 탐지 문제는 그래프 데이터베이스가 주어졌을 때 그래프의 주된 인스턴스와 다르고 드물게 나타나는 노드, 엣지, 서브 그래프 등의 그래프 오브젝트를 탐색해야 한다. 생활패

턴 탐지 연구는 패턴 파악의 기준을 모델에게 학습시키기 어렵다는 점과 생활패턴이 선제적으로 정의되어야 한다는 측면의 어려움이 있다. 또한 그래프 구조 데이터에서의 생활패턴은 그래프 각 노드 사이의 관계가 사전 정의되어 있더라도 개인적인 변산이 있다. 특히 그래프의 종류 및 데이터에 따라 탐지의 범위가 달라짐에 대비할 수 있는 메커니즘이 필요하다. 따라서 해당 문제점을 대처하기 위해, 본 과제에서는 의미 구조 반영 분할 그래프 앙상블 연산이 기반으로 되는 그래프 패턴 발견 기술 개발을 제안한다.

2. 그래프 기반 추천 알고리즘 동향 분석 및 조사

그래프 기반 추천 시스템은 최근 몇 년간 그래프 신호 처리 분야에서 높은 관심을 받고 있다. 다양한 연구에서 이 방법을 활용하여 추천 시스템의 성능을 향상시키는 방법을 연구하고 있다. 이 방법은 크게 3단계로 나눌 수 있다.

가. 그래프 신호 처리

그래프 신호 처리는 그래프 데이터에서 신호를 추출하고 처리하는 기술이다. 이를 위해서는 그래프 내 각 노드의 연결성과 상호작용을 분석하고, 이를 통해 노드 간의 유사도와 중요도를 측정한다.

나. 필터링

그래프 신호를 처리하기 위해 필터를 적용한다. 필터는 주파수 영역에서 신호를 변형시키는 작업을 수행한다. 이를 통해 그래프 신호를 저주파, 고주파로 나누어 필터링을 수행한다.

다. 추천

필터링된 그래프 신호를 이용하여 추천을 수행한다. 이를 위해서는 추천 알고리즘을 활용하여 사용자의 선호도와 상품의 특성을 분석하고, 이를 통해 새로운 추천 아이템을 도출한다.

이러한 기술들을 활용하여 현재 그래프 필터 기반 추천 시스템은 저주파 필터링을 기반으로 하는 방법과 고주파 필터링을 기반으로 하는 방법으로 나뉘어 있다. 또한, 이러한 방법들은 딥러닝과의 결합을 통해 더욱 정확하고 효과적인 추천 시스템을 구현하는 방법으로 발전하고 있다.

그러나, 기존 그래프 신경망 연구들은 그래프 신경망의 저역 통과 필터의 특성 때문에 오버스무딩 문제와 헤테로필리 노드 문제에 직면하여 한계가 있다. 이 연구에서는 라이프로그 그래프 패턴을 반응-확산 방정식을 활용하여 그래프 신경망을 고도화하여 개발하고 검증한다. 제안한 그래프 신경망은 노드 분류에 대한 벤치마크 데이터셋에 대해 최고 정확도를 달성하고 오버스무딩 및 헤테로필리 노드 문제를 해결한다.

이 연구의 의의는 기존 그래프 신경망의 한계를 극복하고 라이프로그 데이터와 같이 복잡한 네트워크를 더 효과적으로 모델링하고 분석할 수 있는 새로운 방법론을 개발하는 데 있다. 또한 이 연구를 통해 그래프 신경망을 더 효과적으로 활용하여 노드 분류 및 예측 문제를 해결함으로써 라이프로그 데이터뿐만 아니라 다양한 응용 분야에서 더 나은 결과를 얻을 수 있을 것으로 기대한다.

3. 개인화된 그래프 필터 기반 추천 알고리즘 개발

본 연구의 목표는 라이프로그 그래프 패턴 기반 행동 추천 알고리즘을 개발하는 것이며, 이를 통해 기존 추천 시스템의 한계를 극복하고 사용자 행동 패턴을 더 정확하게 고려하는 데 의의가 있다. 기존 추천 시스템은 행렬 분해부터 그래프 기반 협업 필터링까지 연구가 많이 되어왔지만, 최근 그래프 기반 협업 필터링은 저역 통과 필터의 특성을 가지고 있기에 사용자 행동 패턴을 고려하는 데 한계가 있다. 본 연구는 저역 통과 필터와 고역 통과 필터를 동시에 사용하고 연속적인 방법으로 추천 시스템을 고도화한다. 연구 결과 LightGCN 모델의 기준으로 벤치마크 데이터인 Amazon-book의 성능지표 Recall@20 기준으로 10% 향상의 목표를 달성한다.

본 연구의 의의는 사용자의 행동 패턴을 포착하는 것이 추천 시스템의 성능에 중요하다는 것이다. 추천 시스템 중 협업 필터링은 사용자의 과거 소비와 다른 사용자와의 유사성을 고려하여 예측이 이루어질 수 있다고 가정한다. 이 가정은 사용자 행동을 기반으로 예측한다. 따라서 사용자 행동 패턴을 포착하는 것은 추천 시스템의 고도화에 기여할 수 있다. 실제 산업에서는 뛰어난 추천 성능으로 사용자들의 만족도를 높일 수 있다.

4. 개발된 알고리즘의 유용성 평가를 위한 프로토타입 개발

개발된 그래프 임베딩 및 그래프 추천 알고리즘의 유용성 평가를 위해 각 개발된 방법을 검증하는 프로토타입을 개발하고 유용성을 확인한다. 일반적으로 라이프로그 데이터는 센서에 포함된 노이즈나 개인 정보 유출 등의 문제로 인해 데이터를 구성하거나, 구성하더라도 가공하여 사용하는 데에 있어 어려움이 존재하기 때문에 개발된 방법의 프로토타입을 개발하여 여러 태스크에 적용하여 유용성 평가를 수행하였다.

제 2 장. 그래프 분할기반 그래프 컨볼루션 신경망의 고도화

제 1 절. 의미 구조 반영을 위한 그래프 분할 알고리즘 고도화

1. 그래프 신경망과 그래프 분류

그래프 분류는 특정 속성에 기반하여 그래프를 분류하는 과정에 집중한다. 사회 및 분자 네트워크와 같은 데이터셋에서 이러한 방법이 널리 사용된다. 그래프 데이터에서 정보를 추출하는 것은 패턴 인식, 예측, 의사 결정 등의 분야에서 중요한 역할을 하며, 시간이 지남에 따라 다양한 그래프 분류 방법이 개발되었다.

가. 그래프 신경망

최근 몇 년 동안 그래프 신경망에 관한 연구가 확장되어, 다양한 모델이 개발되었다. 그래프 컨볼루션 네트워크(GCN)는 노드와 엣지 모두를 활용해 정보를 전파하는 모델로, 이웃 노드로부터 데이터를 수집하여 노드의 특성을 업데이트한다. 그래프 동형성 네트워크(GIN)는 그래프의 구조와 노드 특성을 모두 고려하는 신경망 모델로, 그래프의 특성을 추출하는 데 사용된다. GIN0는 노드 특성 집계 과정에서 가중치를 사용하지 않는 간소화된 버전이다. GraphSAGE는 이웃 노드로부터 정보를 집계하고, 이를 기반으로 노드 임베딩을 업데이트한다. 그래프 주의 네트워크(GAT)는 그래프 내 노드와 엣지 간 상호작용을 고려하여 중요한 정보를 강조하는 기술이다. 이 기술은 그래프의 전역 구조를 고려하여 특정 노드에 초점을 맞추는 데 유용하다. 이러한 모델들은 그래프 내 정보를 효과적으로 전파하고 업데이트하는 데 중점을 둔다. 이러한 모델들의 작동 과정은 표 2에서 개요로 제공된다.

Name	Method
GCN[1]	$X^{l+1} = \widetilde{D}^{-(\frac{1}{2})}\widetilde{A}\widetilde{D}^{(\frac{1}{2})}X^l W^l$
GIN[2]	$X^{l+1} = MLP((A + (1+\epsilon) \cdot I) \cdot X)$
GAT[3]	$X_i^{l+1} = \alpha_{i,j}\theta X_i^l + \Sigma_{j \in N(i)}\alpha_{i,j}\theta X_i^l$
	Attention coefficient $\alpha_{i,j} = \frac{\exp(\leq LeakyRelu(a^T \mid \theta X_i^l \mid \mid \theta X_j^l \mid))}{\Sigma_{k \in N(i) \cup i}\exp(\leq LeakyRelu(a^T \mid \theta X_i^l \mid \mid \theta X_j^l \mid))}$
GraphSAGE[4]	$X_i^{l+1} = W_1 X^l + W_2 \cdot mean_{j \in N(i)} X_j^l$

표 2. 여러 그래프 모델의 작동 방법에 대한 개요

이외에도 그래프 데이터의 분류와 관련하여 다양한 연구가 진행 중이다. 예를 들어, 그래프 기반 머신러닝에서는 복잡한 네트워크 구조 내의 패턴을 인식하고 이해하는 데 초점을 맞추고 있다. 이를 통해 사회 네트워크 분석, 생물정보학, 신약 개발 등 다양한 분야에서 응용 가능성을 탐구하고 있다. 또한, 빅데이터 시대에 맞춰 그래프 기반 머신러닝은 대규모 데이터셋의 처리와 분석에서 중요한 역할을 하고 있다. 이런 연구들은 그래프 데이터의 복잡성을 해결하고, 실제 문제에 적용 가능한 효율적인 솔루션을 제공하는 데 기여하고 있다.

나. 그래프 풀링

이 방법은 중요한 정보를 유지하면서 그래프 크기를 줄이는 데 초점을 맞춘다. 예를 들어, TopK는 중요한 정보만을 보존하면서 그래프 크기를 줄인다. SAGPool은 중요한 노드를 선택하고 그룹화하여 각 그룹의 대표 노드를 생성한다. EdgePool은 그래프의 엣지를 풀링하여 크기를 줄이는데, 이는 그래프 구조를 유지하면서 중요한 엣지를 선택한다. Graclus는 그래프를 작은 서브 그래프로 클러스터링하며, 이는 서브 그래프 내의 노드들이 밀접하게 상호 작용하며 정보를 공유할 수 있게 한다. 이후 그래프는 다시 확장되고 재구성된 임베딩을 생성한다.

초기 그래프 처리 방법들은 그래프의 구조와 속성을 정확하게 모델링하는 데 어려움을 겪었다. 전통적인 기계 학습 기술은 그래프 데이터의 구조와 관계를 파악하는 데 어려움을 겪었으며, 그래프의 규모와 복잡성에 적합하지 않았다. 또한 그래프의 크기나 구조적 다양성이 결과에 영향을 미

칠 때 성능의 제한이 명확해졌다.

이러한 도전 과제에 대응하여, 최신 그래프 처리 기술은 그래프의 구조적 복잡성과 다양성을 더 잘 이해하고 처리할 수 있는 방법론을 개발해 왔다. 이는 그래프 기반 머신러닝의 발전을 촉진하고, 그래프 데이터의 심층적 분석 및 응용 가능성을 확대하고 있다. 이를 통해 사회 네트워크 분석, 생명과학, 신약 발견 등 다양한 분야에서 그래프 데이터의 가치를 극대화하는 새로운 방법들이 개발되고 있다.

2. 그래프 신경망의 문제점

이러한 도전을 해결하기 위해 그래프 신경망(GNNs)이 그래프 데이터에서 구조적 정보를 추출하는 강력한 모델로 도입되었다. GNN은 노드와 엣지의 속성을 통합하여 그래프 특성에 대한 더 깊은 이해를 제공한다. 예를 들어, 주목할 만한 GNN 방법 중 하나인 그래프 컨볼루션 네트워크(GCN)의 방정식이다.

$$X^{l+1} = f(X, A) = \sigma(\widetilde{D}^{-(\frac{1}{2})}\widetilde{A}\widetilde{D}^{(\frac{1}{2})}X^{l}W^{l})$$

이 표기법에서 A는 그래프 G의 인접 행렬을 나타내며 $\widetilde{A} = A + I_N$에서 I_N는 항등 행렬이다. 여기서 특정 노드 i로부터 n개의 엣지를 지나 노드 j의 정보를 참조하려면 GCN은 최소 n개의 계층이 필요하다. 이러한 계층 간 참조 범위의 확장은 그래프 신경망에 구조적 속성을 이해하는 뛰어난 능력을 부여한다. 그 결과 그들은 그래프의 구조와 노드 간 복잡한 관계를 식별하는 데 전통적인 방법보다 뛰어나다.

그러나 GNN이 계층을 통해 발전함에 따라 그래프 전반에 걸쳐 로컬 정보의 희석이 발생하며, 이는 그래프 크기가 커질수록 더욱 중요해진다. 그래프의 전역 정보를 임베딩 하기 위해 여러 계층을 사용할 때, 노드는 같은 노드, 같은 정보를 반복적으로 참조하며 이는 로컬 정보를 희석할 수 있다. 이러한 도전은 노드 간의 효과적인 정보 업데이트를 위해 더 많은 GNN 계층이 필요한 더 큰 그래프에서 더욱 악화된다. 이렇게 널리 퍼진 도전은 GNN에서 오버-스무딩 문제로 알려져 있으며 그 영향을 완화하기 위한 상당한 연구 노력이 이루어지고 있다. 특히 노드 기능의 통합이 필요한 그래프 분류에서 이 문제가 더욱 도전적이다. 특정 노드 기능이 그래프를 지배하면 임베딩

과정에서 다른 독특한 노드 정보를 가리게 되어 구별되는 로컬 정보가 희석될 수 있다.

3. 그래프 지역 정보 보존을 위한 그래프 분할 방법

그래프 임베딩 중 전역 및 로컬 정보의 균형을 맞추고 정보 희석을 줄이기 위해, 그래프를 여러 서브 그래프로 나누는 방법이 있다. 각 서브 그래프는 독립적으로 임베딩 되는데, 이 방법은 각 서브 그래프의 독특한 특성에 초점을 맞추어 광범위하고 구체적인 세부 사항 사이의 조화를 유지하는 것을 목표로 한다. 외부 연결을 끊음으로써, 서브 그래프는 그래프 깊이가 증가하더라도 내부 세부 사항을 유지할 수 있다. 여기에서 효과적인 분류를 위해 그래프를 어떻게 나누느냐가 중요하다.

그래프를 나누는 방법은 비교적 간단한데, 몇 개의 엣지만이 두 서브 그래프를 연결한다면 구분되는 특성을 가질 것으로 예상할 수 있다. 이는 커뮤니티 탐지와 유사하기에 그래프 내에서 밀접하게 상호 작용하는 노드 그룹을 추출하고 서브 그래프를 형성하는 것을 목표로 한다. 서브 그래프를 생성할 때, 가장 작은 외부 정도를 가진 노드를 선택하고 이를 중심으로 서브 그래프를 분할하면 된다. 이는 서브 그래프의 내부 구조를 강화하고 밀접하게 관련된 노드들의 더 타이트한 그룹화를 가능하게 한다. 서브 그래프는 최종 그래프 임베딩의 일부를 임베딩하는 데 사용될 수 있으며 증가하는 계층과 관련된 정보 손실을 완화할 수 있다. 각 서브 그래프는 독립적으로 학습되어, 분할된 그래프의 임베딩 벡터가 전체 구조에 통합될 때 원래 그래프의 로컬 특성을 더 정확하게 유지할 수 있다.

본 연구에서는 제안된 방법을 그래프 분할 알고리즘과 그래프 임베딩 및 분류를 위한 프레임워크 이렇게 두 단계 구조로 설계되었다. 전체 작동은 알고리즘 1에서 설명되고 그림 1은 제안된 방법의 전체 구조를 보여주며 GNN에서의 전통적인 그래프 분류 방법에 대한 간략한 설명을 포함하고 있다.

Input: Graph G (with adjacency matrix A and node features X)
Output: Graph G, Subgraph S^1, S^2

$D =$ Degree of each node in G.
$d =$ Diameter of G.
$n =$ number of nodes in G
if d is even number **then**
    $k = \frac{d}{2} - 1$
else
    $k = \frac{(d-1)}{2}$
end
$R =$ Sum of $[A, A^2, A^3, \ldots, A^k, I]$, $I = Identity\ matrix$
Extract the subgraph S_i according to node i, using R
$E_i =$ External degree of S_i
$e = \infty$
for i from 1 to n **do**
    for j from 1 to n **do**
        if $S_i \cap S_j = \emptyset$ and $E_i + E_j < e$ **then**
            $e = E_i + E_j$
            $S^1 = S_i, S^2 = S_j$
        end
    end
end
Return Graph G, Subgraphs S^1 and S^2

알고리즘 1. 그래프 분할

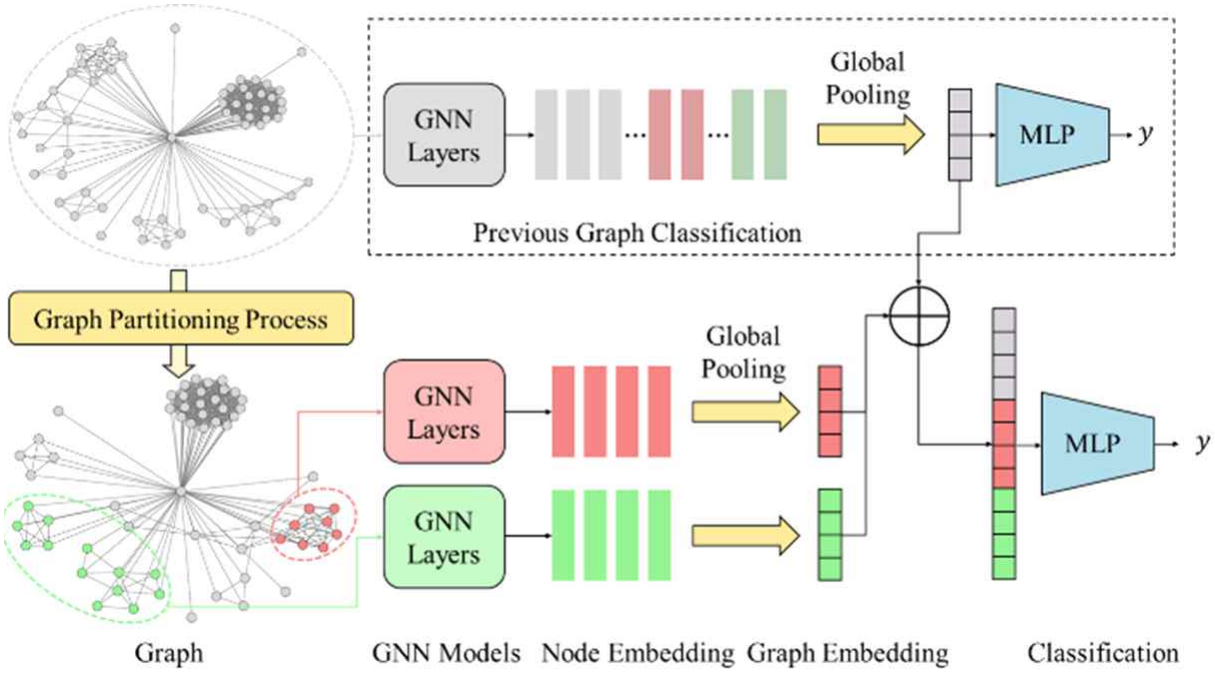


그림 1. GNN에서의 전통적인 그래프 분류 방법

가. 서브 그래프 임베딩을 위한 그래프 분할

분할 알고리즘은 각 노드 주변에서 이웃 기반 확장을 수행하여 임시 서브 그래프를 계산한다. 특정 기준에 도달하면 반복적인 이웃 기반 확장 과정이 종료되고, 각 노드 기반 서브 그래프의 외부 정도가 계산된다. 이어서 중복되지 않고 가장 작은 외부 정도를 가진 서브 그래프가 선택된다.

그래프는 G로 표시되며, 인접 행렬 A와 노드 특성 X로 구성된다. 그래프는 총 n개의 노드를 가지며 인접 행렬 A의 크기는 $n \times n$이고, 노드 특성 X의 크기는 $n \times f$이다. 여기서 f는 노드 특성의 차원성 또는 크기를 나타내며 각 노드와 관련된 특성의 수를 나타낸다.

그래프 분할 알고리즘은 다음 단계를 따른다.

(1) 이웃 기반 확장

먼저 각 노드의 정도를 계산하는데, 노드의 정도는 해당 노드에 연결된 엣지의 수와 같으며 D로 표시될 수 있다.

$$D_{i,i} = \Sigma_j A_{i,j}$$

제안된 알고리즘의 목표는 GNN이 노드 정보를 효과적으로 업데이트하고 로컬 정보를 일반화하여 특성을 임베딩 할 수 있도록 하는 것이다. 이를 달성하기 위해 분할 알고리즘은 이웃을 기반으로 반복적으로 확장되며 각 반복에서 노드가 참조할 수 있는 이웃을 선택한다.

k번 반복 후 행렬 A^k의 (i, j) 항목은 그래프에서 노드 i에서 노드 j까지의 길이 k의 경로 수를 나타낸다. 이는 노드 i에서 노드 j까지 k 거리 내에서 이동할 수 있는 경로의 수를 나타낸다. 일부의 경우에는 이전 $k-1$ 반복에서 도달 가능했던 노드가 k번째 반복에서 도달할 수 없게 될 수 있다. 이러한 과정은 다음과 같은 특별 행렬을 통해 계산한다.

$$R = H(\Sigma_k A^k + I), H(x) = [1 \text{ if } x > 0, else\, 0]$$

도달 가능한 노드를 식별하기 위해 이진 행렬 R_i를 생성한다. 여기서 도달 가능한 노드는 1로, 도달할 수 없는 노드는 0으로 표현된다. 서브 그래프 추출을 용이하게 하기 위해, 계산 과정에서 이진 행렬 R_i에 항등 행렬 I를 추가한다.

(2) 임시 서브 그래프 형성

알고리즘에서 반복을 언제 종료할 것인지 결정하는 것은 중요한 측면이다. 반복 횟수가 너무 적으면(k가 그래프 크기에 비해 작으면) 의미 있는 로컬 정보가 추출되지 않을 수 있다. 반면 반복 횟수가 너무 많으면(k가 그래프 크기에 비해 크면) 서브 그래프의 중복이나 로컬 정보의 희석 가능성이 있다. 이 문제를 해결하기 위해 그래프의 직경을 기반으로 유연한 종료 기준을 설정한다. 그래프의 직경은 그래프 내 임의의 두 노드 간의 최단 거리의 최댓값을 말한다. 따라서 다음과 같은 정리가 성립된다.

정리 1: 그래프G의 직경이 d이고 d가 2보다 크면 항상 2개 이상의 중복되지 않는 이웃 기반 서브 그래프를 생성할 수 있다.

이는 그래프 G의 직경을 구성하는 노드를 고려하여 증명할 수 있는데, 예를 들어, 직경의 3인 경우 그것을 구성하는 노드는 각각 두 노드로 서브 그래프를 형성할 수 있다. 이 이론에 의하면 k를 다음과 같이 정의할 수 있다.

- d가 짝수일 때: $k = \frac{d}{2} - 1$
- d가 홀수일 때: $k = \frac{d-1}{2}$

k의 이러한 값들을 선택하는 이유는 중복 없이 이웃 확장을 기반으로 가장 큰 서브 그래프를 얻기 위함이다. 직경이 2인 경우 가장 멀리 떨어진 두 노드를 임의로 선택하고 그들로부터 1홉 거리 내에 있는 서브 그래프를 추출한다.

(3) 외부 정도 계산

서브 그래프의 외부 정도를 계산하기 위해 다음과 같은 과정을 수행한다. 이 시점에서 원래 그래프 G에서 추출된 서브 그래프는 S로 표시된다. S의 외부 정도를 계산하기 위한 공식은 다음과 같다.

$$S_i^{external} = \Sigma_j R_{i,j} D_{j,j} - \Sigma_n \Sigma_m A_{n,m}$$

위 공식의 첫 번째 부분은 노드 i를 기반으로 생성된 서브 그래프의 총 정도를 나타내며, 두 번째 부분은 서브 그래프가 구성될 때 엣지의 수의 두 배를 나타낸다. 각 서브 그래프의 외부 정도는 노드 i를 기반으로 계산되며 이후 서브 그래프 선택 과정에서 사용된다.

(4) 서브 그래프 선택

제안된 알고리즘은 가장 낮은 외부 정도를 가진 두 그래프를 선택하여 중복되지 않도록 하고, 이를 최종 서브 그래프로 사용한다. 낮은 외부 정도는 서브 그래프에 연결된 노드 수가 적음을 나타내며 구조적 정보에 기반하여 다른 서브 그래프들과 비교해 낮은 연결성을 제안한다. 이는 낮은 연결성을 기반으로 로컬 정보를 추출하고 중복을 피하면서 가능한 많은 특성을 추출하려는 목적을 가지고 있다. 이렇게 선택된 그래프 G^{ori}, S^1, S^2로 표시된 것들은 그래프 분류 작업을 위해 그래프 신경망 모델에 입력된다.

4. 서브 그래프를 이용한 그래프 분류

그래프 분할 알고리즘 이후에는 예측을 위해 적절한 임베딩으로 그래프를 표현할 필요가 있다. f로 표기된 그래프 신경망을 사용하여 각 그래프를 벡터 형식으로 변환하고 예측을 수행한다. 임베딩 과정은 다음과 같이 표현할 수 있다.

$$X^{l+1} = f(X^l, A, \theta^l)$$

l 계층을 가진 그래프 신경망과 각 계층에서 사용되는 가중치를 나타내는 θ가 주어지면 n번의 반복을 통해 업데이트된 그래프 임베딩을 생성한다. “풀(pool)”이라는 용어는 노드 정보를 단일 표현으로 결합하는 풀링 과정을 나타낸다.

$$O = pool(\acute{X})$$

이후 그래프 예측을 위해 분할 알고리즘에서 얻은 임베딩을 단일 벡터로 변환한다. 각 그래프의 특성을 최대한 보존하기 위해 우리는 연결 연산을 사용해 그래프 임베딩을 하나로 병합한다. 마지막으로 MLP를 사용하여 그래프를 예측한다.

$$y = MLP(concat(O_G, O_{S^1}, O_{S^2}))$$

요약하자면, 제안된 방법은 그래프 신경망을 사용해 그래프 임베딩을 생성하고 분할 알고리즘에서 얻은 임베딩 벡터를 연결해 그래프 분류를 위해 MLP를 적용한다.

5. 네트워크 설명 및 실험 구성

가. 데이터셋

제안된 방법의 효과를 검증하기 위해 그래프 분류에 일반적으로 사용되는 11개의 벤치마크 데이터셋에서 실험을 수행하였다. 표 3은 이 11개의 데이터셋에 대한 자세한 정보를 제공한다.

COLLAB 데이터셋은 과학 협력 네트워크에서 추출된 것으로, 노드는 연구 논문의 저자를 나타내고 엣지는 협력 관계를 나타내며 이는 두 개의 연구 분야로 분류된다. IMDB-BINARY 및 IMDB-MULTI 데이터셋은 영화와 관련되어 있으며 노드는 배우를 나타내고 엣지는 동일한 영화에서의 공동 출연을 나타낸다. 따라서 이는 액션 및 로맨스 장르로 분류된다. MUTAG 데이터셋은 화학 구조를 나타내며 노드는 원자를 나타내고 엣지는 화학 결합을 나타낸다. PTC 및 NCI 그룹은 MUTAG와 유사한 분자 데이터셋이지만 각 그룹의 노드 특성 수에 차이가 있다. PROTEINS 데이터셋은 노드로서 단백질 아미노산을 나타내며 엣지는 아미노산 간의 상호작용을 나타낸다.

Name	Ref.	# of Graph	# of Class	Avg. Nodes	Avg. Edges	# of Node feature
IMDB-B[b]	[5]	1000	2	19.77	96.53	136[a]
IMDB-M[b]		1500	3	13.00	65.94	89[a]
COLLAB		5000	3	74.49	2457.78	492[a]
MUTAG	[6]	188	2	17.93	19.79	7
PTC_MR	[7]	344	2	14.29	14.69	18
PTC_FR		351	2	14.56	15.00	19
PTC_MM		336	2	13.97	14.32	20
PTC_FM		349	2	14.11	14.48	18
NCI1	[8], [9]	4110	2	29.87	32.30	37
NCI109		4127	2	29.68	32.13	38
PROTEINS	[10],[11]	1113	2	39.06	72.82	3

[a.]노드 라벨이 기본적으로 제공되지 않는 데이터셋의 경우, 각 노드의 차수에 따라 노드 라벨 할당
[b.]IMDB-B: IMDB-BINARY, IMDB-M: IMDB-MULTI
[c.]모든 데이터셋은 https://chrsmrrs.github.io/datasets/에서 다운로드 가능

표 3. 데이터셋에 대한 정보

나. 실험 설정

표 4에 설명된 모든 방법에 대해 실험을 수행했다. 모든 실험은 10-겹 교차 검증을 사용하며, 200 에포크, 4096의 배치 크기, Adam 옵티마이저, 초기 학습률 0.01, 학습률 감소 계수 0.5, 학습률 감소 단계 크기 50으로 수행되었다. 각 데이터셋과 방법에 대해 계층 수와 숨겨진 차원의 다양한 조합을 탐색해 최고의 성능을 기록했고 고려된 조합은 레이어 = [1, 2, 3, 4, 5]와 히든 레이어 = [16, 32, 64, 128]이다. 이후 추출된 특성에 그래프 분할 알고리즘을 적용하여 제안된 프레임워크에 동일한 과정을 따랐다. 모든 데이터셋과 방법에 대해 선택된 계층 수와 숨겨진 차원은 표3에 명시된 값과 일치한다. H는 숨겨진 차원을 의미하며 1은 16을, 2는 32를, 3은 64를, 4는 128을 나타낸다.

또한 표 5와 표 6에서 이름 뒤에 “JK”가 붙은 방법들은 점핑 지식 계층 집계를 통합하는 모듈을 나타낸다. 실험은 6 * A100 80GB GPU를 사용하여 수행되었고 사용된 백그라운드 프레임워크는 PyTorch-Geometric이다.

Method		GCN	GCNWithJK	GIN	GINWithJK	GIN0	GIN0WithJK	TopK	SAGPool	EdgePool	4Graclus	GAT	Set2Set	GraphSAGE WithJK	GraphSAGE	Overall
Dataset	MUTAG	L	5	5	2	3	3	3	5	4	5	5	5	5	5	5	4.3
		H	4	4	1	1	1	1	4	3	2	2	4	3	4	3	2.6
	NCI1	L	4	4	2	2	2	2	3	5	5	5	5	4	4	5	3.7
		H	4	4	3	3	4	4	4	3	3	3	4	4	3	3	3.5
	NCI109	L	5	4	2	2	2	3	4	5	5	5	5	3	3	4	3.7
		H	2	2	3	4	4	4	3	3	2	3	3	3	3	3	3
	PROTEINS	L	4	5	1	1	1	2	4	4	5	5	4	2	4	4	3.3
		H	4	4	4	4	3	4	3	2	4	4	4	3	4	4	3.6
	PTC_MR	L	5	3	2	1	1	2	5	5	4	5	2	5	2	1	3.1
		H	2	2	4	4	4	1	3	4	1	2	2	3	2	3	2.6
	PTC_FR	L	5	5	1	2	1	1	4	3	3	5	4	4	5	2	3.2
		H	4	4	3	4	3	3	3	3	3	4	4	4	4	2	6.4
	PTC_MM	L	5	5	3	5	4	2	5	5	4	3	1	1	3	4	3.6
		H	4	3	1	1	1	1	4	4	3	4	4	3	4	4	2.9
	IMDB_BINARY	L	5	4	1	1	1	1	5	5	1	5	4	4	2	2	2.9
		H	4	4	3	2	2	3	4	2	4	4	4	4	4	4	3.4
	IMDB_MULTI	L	4	4	1	4	1	1	4	4	1	2	2	1	2	3	2.4
		H	4	4	2	1	2	4	3	4	4	4	4	4	4	3	3.4
	COLLAB	L	1	1	2	4	1	5	1	1	1	2	1	2	1	1	1.7
		H	3	4	1	2	3	2	4	2	4	3	4	3	4	4	3.1

표 4. 데이터셋과 방법에 대해 선택된 계층 수와 숨겨진 차원 수

Method	Dataset							
	MUTAG		*NCI1*		*NCI109*		*PROTEINS*	
	Original	*Ours*	*Original*	*Ours*	*Original*	*Ours*	*Original*	*Ours*
GCN	74.0±6.1	**76.6±5.1**	69.2±2.7	**71.8±5.2**	**67.5±2.9**	64.6±9.8	71.6±3.8	**74.1±4.3**
GCNWithJK	**77.2±5.3**	75.1±5.9	70.7±2.7	**70.8±2.8**	68.2±3.0	**68.4±2.7**	72.1±3.9	**72.4±3.6**
GIN	81.0±10.2	**82.5±9.9**	**76.4±4.1**	76.0±2.8	73.6±5.2	**74.7±4.1**	71.4±4.4	**72.2±4.0**
GINWithJK	81.4±7.9	**82.9±9.1**	74.5±4.3	**75.8±3.1**	74.1±3.7	**75.9±2.1**	71.8±4.6	**72.6±2.4**
GIN0	80.9±7.5	**82.3±12.1**	74.9±3.7	**76.8±3.0**	75.0±2.9	**75.1±2.7**	70.1±4.1	**71.9±3.9**
GIN0WithJK	82.1±8.4	**83.0±9.2**	75.8±4.3	**76.6±3.6**	**75.7±3.2**	75.6±2.5	**71.6±4.9**	71.3±3.5
TopK	72.9±5.8	**78.2±5.6**	71.6±4.6	**74.9±3.9**	69.9±3.2	**71.7±2.9**	72.0±3.4	**72.1±4.2**
SAGPool	80.8±10.8	**83.0±7.4**	70.8±5.1	**74.5±7.9**	70.9±3.6	**72.0±3.2**	71.8±3.4	**72.2±5.4**
EdgePool	73.5±5.9	**78.9±9.5**	**73.1±2.5**	72.4±5.3	70.1±5.6	**70.4±2.6**	71.0±3.6	**71.3±1.4**
Graclus	77.1±5.9	**77.3±7.3**	71.3±4.4	**76.0±1.7**	70.3±2.9	**71.2±3.6**	71.9±3.3	**72.5±3.2**
GAT	75.5±8.9	**77.1±8.0**	71.5±4.8	**72.2±5.6**	**68.5±4.7**	66.6±9.0	71.8±4.0	**72.8±4.3**
Set2Set	**73.4±11.4**	70.3±12.1	70.4±3.5	**73.8±3.9**	**79.7±2.7**	78.9±7.3	**73.1±4.7**	72.2±5.4
GraphSAGE	**77.2±4.7**	74.1±8.5	71.3±3.5	**76.1±3.1**	69.6±3.1	**70.3±3.2**	**71.6±2.7**	71.4±2.9
GraphSAGWithJK	**77.8±9.4**	76.2±8.2	71.2±3.5	**72.1±2.9**	**70.2±2.7**	69.7±2.3	**72.5±4.7**	70.8±5.3
Overall	76.7±7.73	**78.4±8.4**	72.3±3.8	**74.3±3.9**	71.7±3.5	**71.8±4.1**	**71.7±4.0**	**72.1±3.8**
	PTC_MR		*PTC_FR*		*PTC_MM*		*PTC_FM*	
	Original	*Ours*	*Original*	*Ours*	*Original*	*Ours*	*Original*	*Ours*
GCN	54.9±5.0	**56.4±6.3**	**65.5±2.9**	64.9±2.5	66.6±8.4	**67.5±5.6**	**60.7±3.9**	58.5±7.3
GCNWithJK	56.4±8.3	**59.0±4.0**	**64.1±5.8**	63.5±3.0	**67.8±4.8**	67.5±5.1	**67.0±8.7**	63.0±5.9
GIN	55.6±4.1	**56.1±3.1**	**64.1±5.7**	63.8±8.0	62.2±7.7	**65.1±7.1**	**61.0±4.4**	58.5±6.8
GINWithJK	57.3±5.4	**57.4±5.4**	62.7±6.3	**63.4±4.5**	61.6±5.7	**61.8±5.7**	57.9±4.6	**58.0±5.1**
GIN0	56.4±6.6	**58.1±6.2**	64.4±7.3	**65.8±4.1**	**64.0±3.2**	59.5±6.2	**60.2±2.9**	58.2±7.5
GIN0WithJK	56.4±3.8	**56.7±4.2**	65.2±6.8	**65.2±5.6**	**64.8±6.4**	62.2±2.5	**60.2±1.9**	57.0±6.6
TopK	57.3±8.8	**57.5±8.3**	66.1±4.2	**66.7±4.7**	**66.9±4.8**	66.6±6.3	60.2±3.9	**61.3±7.6**
SAGPool	57.2±6.2	**57.9±5.0**	66.1±4.8	**67.2±3.7**	66.0±5.2	**67.5±5.3**	60.7±4.5	**61.2±6.6**
EdgePool	55.2±7.4	**56.1±6.9**	64.7±7.1	**65.2±4.6**	65.4±6.2	**67.8±5.1**	60.2±4.8	**60.7±5.6**
Graclus	52.9±9.4	**56.7±4.6**	**65.5±5.7**	65.2±4.6	64.8±6.1	**66.0±5.4**	61.0±4.4	**63.3±3.9**
GAT	53.2±7.0	**58.4±5.6**	65.2±5.9	**65.3±5.2**	67.8±6.0	**69.0±5.7**	59.6±4.0	**61.3±4.2**
Set2Set	**55.2±4.5**	53.8±6.8	**66.7±3.9**	65.5±3.6	**67.8±5.7**	66.3±6.4	59.9±4.3	59.9±1.9
GraphSAGE	54.1±6.1	**56.7±9.0**	**66.4±2.2**	63.5±6.0	63.4±5.7	**66.9±5.8**	**60.5±4.1**	59.9±8.2
GraphSAGWithJK	**58.7±8.0**	57.5±5.1	65.2±4.0	**66.4±3.1**	64.5±6.2	**66.0±7.8**	**59.6±3.7**	58.5±6.2
Overall	55.8±6.5	**57.0±5.8**	65.1±5.2	65.1±4.5	65.3±5.9	**65.7±5.7**	**60.6±4.3**	60.0±6.0

표 5. 노드 특성이 있는 데이터셋에서 10-폴드 크로스 검증을 사용한 그래프 분류의 정확도
(밑줄은 데이터셋에서 가장 높은 점수를 나타내며 굵은 글씨는 더 높은 성능을 나타냄)

Method	Dataset					
	IMDB-BINARY		*IMDB-MULTI*		*COLLAB*	
	Original	*Ours*	*Original*	*Ours*	*Original*	*Ours*
GCN	**74.6±4.8**	73.9±4.0	50.9±3.6	50.9±3.5	80.6±0.4	**82.1±0.9**
GCNWithJK	**74.1±3.6**	73.2±3.9	**49.6±3.2**	48.9±2.4	**80.6±0.7**	80.5±0.6
GIN	**72.7±5.5**	72.1±5.6	48.7±3.0	**50.6±2.9**	80.2±0.6	**81.2±0.8**
GINWithJK	72.3±5.4	**73.3±5.2**	49.8±3.1	**50.3±2.1**	78.8±3.3	**81.9±2.4**
GIN0	73.6±4.4	**73.7±5.4**	48.3±2.3	**49.9±4.3**	80.1±0.2	81.2±1.7
GIN0WithJK	72.3±3.8	**74.5±4.7**	48.6±3.7	**50.2±2.9**	**81.6±1.4**	81.0±0.4
TopK	**74.3±5.6**	73.8±5.5	**50.5±2.7**	49.4±2.5	78.8±1.4	**79.1±2.6**
SAGPool	73.4±5.3	**74.1±5.4**	**50.5±2.5**	50.0±2.6	80.3±1.2	**81.6±1.0**
EdgePool	**73.5±5.0**	73.2±4.7	**50.8±2.5**	50.5±2.4	**82.4±1.0**	80.3±1.9
Graclus	72.7±4.7	**73.2±3.1**	**50.7±2.9**	49.8±3.0	**79.4±0.5**	78.8±1.1
GAT	**73.0±4.6**	72.4±4.2	**50.0±3.2**	49.7±2.9	**80.3±0.7**	79.4±1.4
Set2Set	**72.9±4.4**	72.5±4.9	**50.9±2.9**	49.7±2.6	**77.6±0.8**	76.4±3.1
GraphSAGE	71.7±3.9	**72.8±3.6**	**51.2±3.0**	50.7±3.6	**79.6±2.0**	78.6±1.7
GraphSAGEWithJK	71.7±3.8	**72.5±4.2**	49.9±1.9	**50.9±3.5**	78.4±1.3	**80.0±1.9**
Overall	73.1±4.6	**73.2±4.6**	50.0±2.9	**50.1±2.9**	79.9±1.1	**80.2±1.5**

표 6. 노드 특성이 없는 데이터셋에서 10-폴드 크로스 검증을 사용한 그래프 분류의 정확도
(밑줄은 데이터셋에서 가장 좋은 점수를 나타내며 굵은 글씨는 더 높은 성능을 나타냄)

다. 실험 결과

(1) 비교 연구

표 5와 표 6는 잘 알려진 GNN 방법과 데이터셋에 우리의 제안된 방법을 적용하기 전후의 정확도 비교를 제시한다. 표 5에서, 우리의 방법은 8개의 데이터셋 중 6개에서 평균적으로 뛰어난 정확도를 보이며, 표 5에서는 노드 특성이 없는 실험에서 모든 데이터셋의 평균 정확도가 증가한다. 전체적으로, 수행된 154개의 실험 조합 중 95개에서 성능 향상을 관찰하여, 그래프 분할 알고리즘을 적용함으로써 정보 희석 문제를 해결하는 우리의 제안된 방법의 효과를 나타낸다.

그러나, 표 5에서 PTC_FR 및 PTC_FM에 대한 실험은 우리의 제안된 방법을 적용했을 때 개선이 없거나 심지어 증가하지 않는다는 점을 주목해야 한다. 특히, PTC_FR 데이터셋의 경우, 우리

의 방법은 SAGpool을 포함한 다른 방법들 중에서 최고의 성능을 달성하지만, 성능 차이는 미미하다. 이는 일부 실험에서 다른 실험에 비해 작은 성능 향상이 나타났고, 서브 그래프를 독립적으로 임베딩 하기 위해 우리의 방법이 요구하는 많은 수의 매개변수로 인해 동일한 조건에서 수행될 때 다른 방법들에 비해 더 큰 훈련 부담이 발생할 수 있다는 사실로 기인할 수 있다.

또한, 표 6에서의 성능 향상은 표 5에 제시된 데이터셋에 비해 낮다. 이는 원래 그래프의 모든 노드 간의 상호 연결성으로 인해 더 많은 계층에 의한 정보 희석이 최소화되었을 수 있으며, 이로 인해 제안된 방법의 효과가 감소했을 수 있음을 나타낼 수 있다. 이러한 이유를 조사하기 위해, 우리는 에포크 수와 계층 수에 대한 제안된 방법의 성능 변화를 검토하는 추가적인 분석 연구를 확장하여, 이러한 요인들에 대한 추가적인 증거를 제공하고자 한다.

(2) 분석 연구

① 계층 수 변화: 그림 2는 PROTEINS, NCI109, IMDB-BINARY, IMDB-MULTI라는 네 개의 데이터셋에서 성능이 좋지 않았던 GIN0WithJK와 GraphSAGEWithJK와 비교하여 제안된 방법을 비교한다. 우리는 동일한 실험 설정을 따르면서 계층 수에 따른 성능 변화를 조사한다. 모든 경우에서 제안된 방법은 5개의 계층으로 가장 좋은 성능을 보이며 특히 NCI109와 PROTEINS의 경우, 제안된 방법은 기존 방법들(표 5에 나타난 것과 같이)에 비해 우수한 성능을 달성한다. 주목할 점은, 기존 방법들이 성능 저하를 경험한 반면, 제안된 방법은 지속적으로 성능 향상을 보인다는 것이다. 이는 제안된 방법이 더 많은 계층을 사용하여 로컬 정보를 보존하고 전역 및 로컬 정보를 효과적으로 임베딩 할 수 있는 능력을 갖추고 있음을 나타낸다. 특히 표 6에서 지적된 것처럼, 직경이 긴 그래프에서 그러하다.

반면에, IMDB-BINARY와 IMDB-MULTI에서는 기존 방법들과 제안된 방법 모두 작은 성능 변화를 보이는데, 이는 IMDB-BINARY, IMDB-MULTI 및 COLLAB과 같이 직경이 짧은 그래프에서는 계층 수에 기반한 중요한 성능 변화가 없음을 시사한다. 그러나 최종 분류 신경망이 참조할 수 있는 추가 서브 그래프 임베딩을 제공하는 제안된 방법은 원래 그래프를 통합함으로써 기존 방법들보다 더 나은 성능을 보장한다.

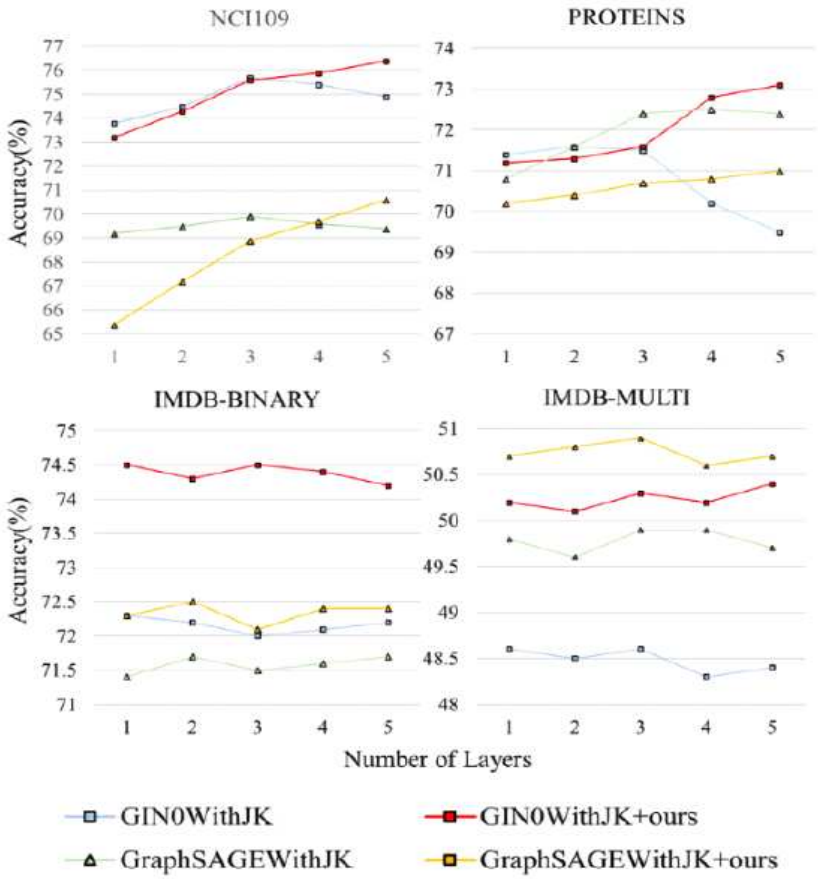


그림 2. 레이어 수에 따른 정확도 변화

② 에포크 수 변화: 계층 수에 기반한 분석과 유사하게, 우리는 제안된 방법이 훈련량에 어떻게 영향을 받는지 또한 조사한다. 그림 3에서 분명하게 나타나듯이, 제안된 방법은 PTC_MM의 Set2Set 모델을 제외하고 다른 모델들에 비해 더 높은 훈련량에서 효과적임을 보여준다. 우리는 제안된 방법이 PTC_FR에서 기존 방법을 200 에포크의 원래 설정보다 2.5배 높은 수준에서 능가한다는 것을 측정했다. 훈련해야 할 모델의 수가 세 배로 증가했다는 점을 고려할 때, 이는 합리적인 수준의 훈련량을 요구한다는 것을 나타낸다.

반면, PTC_MM의 경우에는 훈련량이 기존 방법의 요구 사항을 충족시키지 못하는 것으로 관찰된다. 이는 Set2Set 방법이 LSTM과 내용 기반 주의를 활용하는데, 이는 기존 방법보다 더 많은 훈련을 요구하기 때문이다. 따라서 제안된 방법은 최대한의 학습 잠재력을 극대화하기 위해 다른 전통적인 방법들에 비해 충분한 훈련에 대한 더 작은 요구 사항을 가진다.

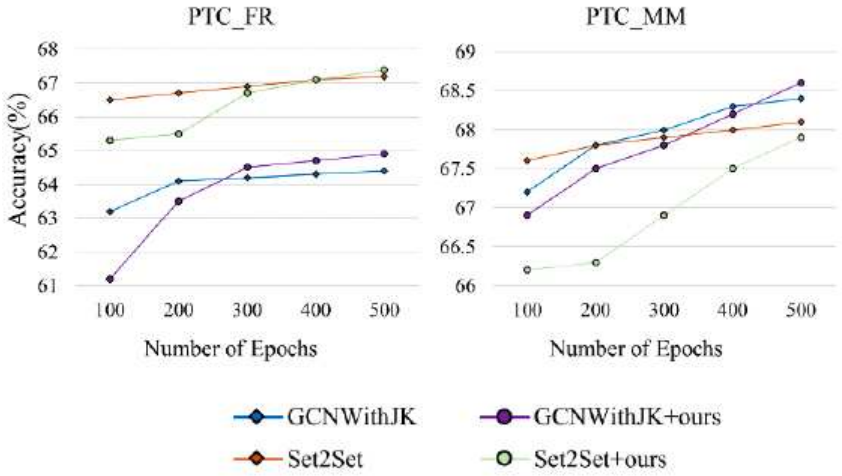


그림 3. 에포크 수에 따른 정확도 변화

③ 그래프 분할 알고리즘 결과: 그림 4는 각 데이터셋에 대한 그래프 분할 알고리즘 후의 그래프 구조를 보여준다. 각 행에서, 가장 왼쪽은 원래 그래프를 나타내며, 어두운색으로 강조된 두 노드는 서브 그래프 생성을 위한 선택된 참조점을 나타낸다. 그림 4(a)는 하나의 중앙 노드가 다른 노드들을 연결하는 특징적인 구조를 보여주며 이는 데이터 자체의 특성으로, 표 3에서 관찰될 수 있는 직경의 감소를 초래한다. GNN 모델이 이 구조를 임베딩 할 때, 중앙 노드의 존재는 전파를 통해 다른 노드들 간의 쉬운 정보 공유를 가능하게 하며, 적은 계층에서도 가능하다.

반면, 그림 4(b)는 반대 상황을 나타낸다. 여기서는 중앙 노드의 부재 또는 엣지 연결 없이 분리된 노드의 존재가 관찰될 수 있다. 이는 직경 증가의 또 다른 원인이며, 여러 계층을 사용하여 전역 정보를 업데이트할 때 로컬 특성의 희석으로 이어질 수 있다.

그래프 신경망에 입력하기 전에 그래프 분할 알고리즘을 쉽게 적용할 수 있는 강력한 방법을 제안했다. 이 방법은 서브 그래프 임베딩의 형태로 로컬 정보를 보존할 수 있게 해주고 다양한 데이터셋에서 계층 수가 증가하더라도 뛰어난 성능지표를 보여줬다. 특히, 직경이 큰 데이터셋에 적용되고 더 많은 계층을 사용할 때, 제안된 방법은 현저하게 향상된 성능을 보여주었다. 이는 로컬과 전역 정보 간의 절충을 해결하고 더 나은 분류 결과를 달성할 수 있는 능력을 나타낸다.

그러나 더 많은 계층을 사용하여 추가 서브 그래프 임베딩을 통합함으로써 매개변수의 수가 증가한다는 점을 주목할 필요가 있다. 직경이 작고 적은 수의 계층으로 충분한 예측이 가능한 데이터셋의 경우, 제안된 방법의 성능 향상은 상대적으로 더 적을 수 있다.

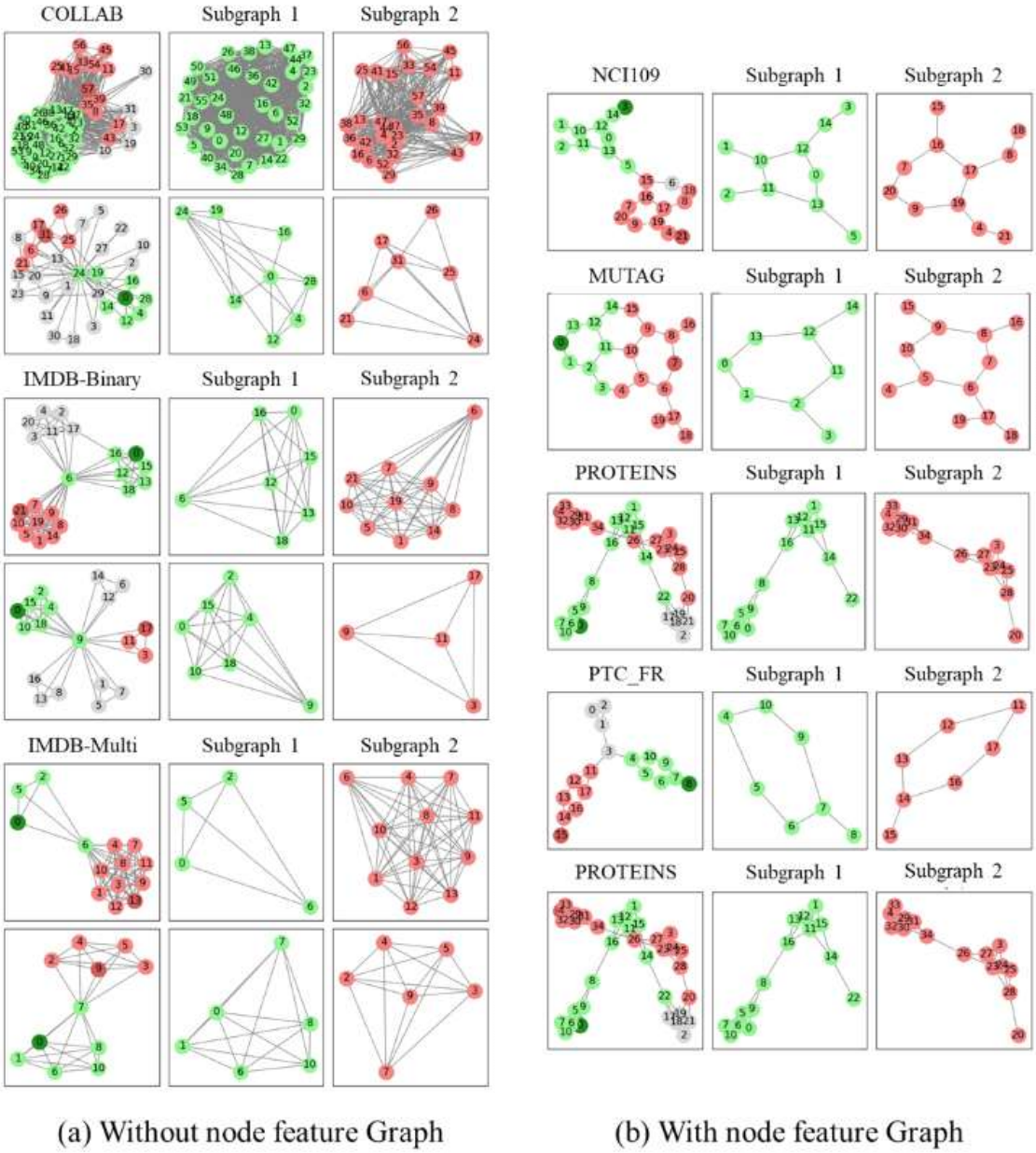


그림 4. 파티셔닝 전후의 그래프

(가장 왼쪽에 있는 그래프는 원래 그래프를 나타내며
더 어두운 색상의 노드들은 서브 그래프의 중심 노드를 대표함)

제 2 절. 분할 그래프 앙상블을 통한 고성능 그래프 분류 및 예측 알고리즘

1. 고성능 그래프 분류

그래프 분류는 노드와 에지의 속성에 기반하여 그래프를 분류하는 분야 중 하나이다. 최근 딥러닝의 발전으로, 그래프 신경망(GNNs)은 그래프를 효율적으로 분류하기 위해 광범위하게 연구되었고, GNNs는 각 노드를 그 이웃과 함께 업데이트한 후 모든 노드 정보를 통합하여 분류를 위한 단일 임베딩 벡터를 생성한다. 이 GNN 접근법은 다양한 문제에 대해 인상적인 성능을 보여주며, 그래프 분류에 매우 효과적이라고 볼 수 있다.

GNNs는 우수한 성능에도 불구하고, 노드 임베딩 과정에서 낮은 그래프 주파수를 가진 노드가 이웃 노드와 혼합되는 "스무딩(smoothing) 현상"에 시달리는데, 이는 GNN의 독특한 노드 특성을 구별하는 능력을 저하한다. 노드 임베딩을 결합하여 임베딩 벡터를 생성하는 후속 단계는 특히 그래프 분류에서 문제를 악화시킬 수 있으며, 이 문제를 해결하기 위해 최근 연구들은 그래프를 서브 그래프로 나누어 스무딩을 방지하려고 한다. 그러나 소셜 네트워크와 같은 큰 그래프에서 노드 정보가 유사한 경우, 이 전략은 그다지 효과적이지 않을 수 있으며, 그래프를 지나치게 복잡하게 만들어 방법의 장점을 감소시킬 수 있다.

이러한 문제를 극복하기 위해, 그래프를 다양한 관점에서 분할하고 그래프 임베딩 벡터를 생성할 때 임베딩 된 서브 그래프 임베딩에 가중치를 할당하는 방법을 사용할 수 있다. 이 가중치를 도입함으로써, 우리는 각 서브 그래프 임베딩이 해당 그래프의 독특한 특성을 포착하는 능력에 따라 중요성을 우선시할 수 있다. 이 방법은 그래프의 내부 속성에 기반하여 노드 씨드를 선택한 다음 그래프를 여러 서브 그래프로 분할하는 것으로 시작한다. 분할된 각각의 서브 그래프는 그래프 신경망을 통한 노드 임베딩 업데이트로 훈련되어, 각 그래프가 그 특성을 반영하도록 한다. 분류를 위해 단일 그래프 임베딩을 생성할 때, 각 그래프의 독특한 속성을 포착하는 서브 그래프 임베딩에 가중치를 할당하기 위해 어텐션을 사용하면 된다. 이 과정은 그래프 내의 로컬 특징을 효과적으로 추출하고 보존하는 방법을 보장할 수 있다.

위 방법을 검증하기 위해, 여섯 개의 대규모 소셜 그래프에서 실험을 수행했고, 모든 데이터셋에서의 실험 결과는 성능 향상을 보여주며, 특히 IMDB-M 데이터셋에서 이전의 최신 방법인

MA-GCNN에 비해 27.87%p만큼 증가한 결과를 도출했다. 또한 ablation study는 선택된 씨드 값과 어텐션 기반의 결과 출력이 결과에 미치는 성능 향상을 분석할 수 있다.

2. 관련 연구

최근의 연구들은 소셜 네트워크에서 그래프 분류와 추천 시스템을 위한 그래프 신경망의 사용을 탐구하였다. 일부 접근 방법에는 이웃 노드의 특성을 집계하여 구조적 그래프 특성을 학습하는 그래프 합성곱 신경망이 포함되며, 라플라시안 고유지도에서 가중치를 고려하여 구조적 특성과 함께 추가적인 로컬 특성을 추출하는 구조적 깊은 네트워크 임베딩이 있었다. 이러한 방법들은 분야를 발전시켰지만, 그래프에서 상세한 로컬 정보를 포착하는 데에 여전히 한계가 있다.

그래프 분류를 위한 언어 모델 적용의 초기 시도로는 그래프의 특정 지점에서 고정된 길이의 경로 샘플을 생성하여 로컬 특성을 추출하는 "딥 워크 (deep walk)" 알고리즘이 있었지만, 이 방법은 대규모 소셜 네트워크에서 제한적이며 그래프의 구조적 특성을 고려하지 않는다.

모델 개발을 넘어서, 다른 최근 방법들은 입력 그래프를 변환하여 로컬 또는 관찰되지 않은 그래프 특성을 발견하는 데 초점을 맞췄다. 예를 들어 Y. Yang의 연구는 행렬 분해를 통해 팩터 그래프를 생성하고 독립적인 GCN을 사용하여 최종 출력 그래프의 노드로 특성을 전달했다.

H. Peng이 제시한 방법은 중심 노드를 선택하기 위해 가까운 중심성에 기반한 모티프 개념을 도입하고, 비동형 서브 그래프를 찾아 구조적 그래프 정보를 보존하기 위해 결합하는 것이다. L. Cotta, P.A. Papp과 Y. Rong은 입력 그래프에서 K 노드를 사용하여 서브 그래프를 생성하거나 K 노드를 제거하거나 에지를 제거함으로써 숨겨진 로컬 정보를 찾았다. 또한 J. Yang의 최근 연구는 해석 가능한 그래프 캡슐을 사용하여 계층적 그래프 표현을 포착하는 데 중점을 두는 것이었다.

제안된 방법들은 소셜 네트워크에서 새로운 로컬 특성을 추출하기 위해 세 가지 그래프 속성을 고려함으로써 이러한 방법들과 다르다. MA-GCNN 모델은 우리가 위에서 제시한 방법과 유사하지만 소셜 네트워크에서 세 가지 속성을 활용하지 않으며 각 서브 그래프의 의미를 간과한다. 우리가 제시한 방법은 원래 그래프를 세 가지 다른 관점에서 관찰하여 그래프의 로컬 패턴을 얻는 것이다. 실험에서 제안된 방법은 소셜 네트워크 분류에서 최신 결과를 달성한 HGCN 및 MA-GCCN 모델과 비교할 수 있다.

3. 제시 방법

이 부분에서, 제안된 방법은 세 단계 구조로 설명된다. 1) 서브 그래프 파티셔닝을 위한 노드 선택 과정, 2) 서브 그래프 임베딩 과정, 그리고 3) 서브 그래프 임베딩 통합 및 분류를 위한 주의력 리드아웃. 그림 5는 제안된 방법의 전체 구조를 보여준다.

가. 노드 선택과 서브 그래프 분할 알고리즘

앞서 언급한 대로, 그래프 내의 지역 정보를 추출하고 보존하기 위해 그래프 분할을 수행한다. 이를 위해 주로 그래프 분석 방법과 그래프 클러스터링 기술에서 흔히 사용되는 연결도, 근접성, 그리고 매개 중심성을 활용한다. 이 중심성 측정치들은 그래프 분할 과정의 기초를 형성하며, 그래프 내에서 중요한 노드를 식별하고 그래프 내의 지역적인 특성을 보존하는 데 도움이 된다. 그래프 이론을 기반으로 노드 시드를 선택하는 것은 그래프의 구조적 특성을 모델링하는 것을 목표로 한다. 우리의 궁극적인 목표는 그래프의 구조적 속성을 기반으로 특정 참조 노드를 선택하여 지역 정보를 보존하는 것이다. 이러한 구조적 속성을 기반으로 그래프를 분할하기 위해 노드 선택에 대한 세 가지 기준을 채택한다.

알고리즘 2는 세 가지 그래프 속성 중 하나와 인접 행렬 정보를 입력 변수로 받는다. 그 후 선택한 속성은 각 노드에 대한 스칼라값이 있는데, 이를 통해 각각의 최댓값과 최솟값을 기반으로 두 개의 시드를 선택할 수 있다. 그래프 분할은 다음 세 가지 속성을 사용하여 수행된다.

그래프에서의 **차수 중심성**은 해당 노드의 연결 수를 나타내어 해당 노드가 몇 개의 엣지에 연결되어 있는지를 나타낸다. 차수가 높은 노드는 더 많은 연결을 가지고 있으며 그래프 내의 지역 특성을 보존하기 위한 분할의 중요한 시작점으로 간주될 수 있다.

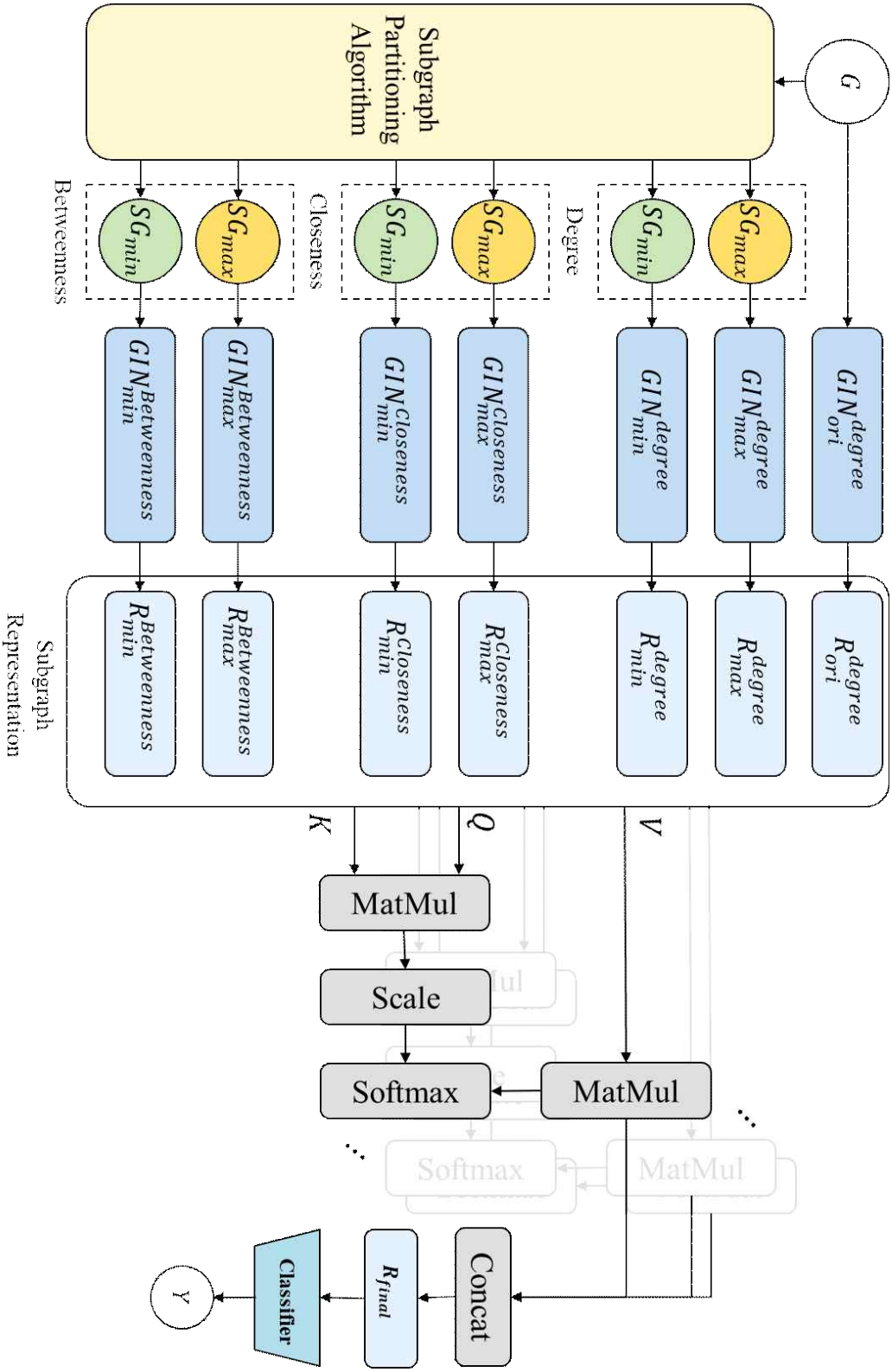


그림 5. 더 나은 분류를 위해 다양한 지역 정보를 활용하여 그래프 표현을 학습하는 제안된 방법

Input: Graph $G = (V, E)$ and Seed Feature F (ex. Degree)
Output: Subgraph S = (S_1, S_2)
1: Initialize maxV = 0, minV=0 and S = []
2: for $v \epsilon V$ do
3: Compute F(v)
4: maxV $\leftarrow \max_{v \epsilon V}$ (F (v))}
5: minV $\leftarrow \min_{v \epsilon V}$(F ($v$))
6: seed_node = [maxV, minV]
7: while True do
8: for i=0:len(seed_node) do
9: neighbor_node = G.neighbor(seed_node[i])
10: S[i].append(neighbor)
11: if G.check_all_node_tranversing(S) do
12: break
13: for i=0:len(S) do
14: S[i] = S.remove_node_duplication()
15: return S

알고리즘 2. 버블 프레임워크 기반의 그래프 분할 알고리즘

$$Degree(v_i) = \sum_{j=1}^{g} x_{i,j}, i \neq j$$

노드 v_i의 차수는 이를 연결하는 모든 엣지의 합으로 정의되며, 여기서 $i \neq j$이다. 이 차수는 노드의 연결 수준을 나타낸다. 차수가 높은 노드는 더 많은 연결을 가지고 있어 지역적인 특성이 있는 집합에서 중요한 참조 지점이 될 수 있다. 이러한 고 차수 노드는 그래프 분할에 있어서 지역 특성을 보존하는 데 중요하다. 소셜 네트워크에서 차수 중심성이 높은 노드는 그들의 다수의 직접적인 연결로 인해 강력한 영향력을 나타낼 수 있다.

그래프에서의 **근접 중심성**은 한 노드와 모든 다른 노드 간의 평균 거리를 나타낸다. 이는 간접적으로 노드의 중요성을 나타내며, 분할에 활용되어 정보 교환을 제한하고 그래프 내에서 노드 중심성을 유지하는 데 사용될 수 있다.

$$Closeness(v_i) = \frac{N-1}{\sum_{j \neq i} l(i,j)}$$

$\sum_{j \neq i} l(i,j)$은 노드 i와 다른 노드 간의 엣지 합을 나타낸다. 그래프 신경망에서의 높거나 낮은 근

접 중심성은 얼마나 자주 노드가 네트워크 레이어 간에 참조되는지 나타내어 그래프 내에서의 중요성을 시사한다. 이를 그래프 분할에 사용하면 중심성을 고려하고 정보의 희석을 줄일 수 있다. 소셜 네트워크에서 근접 중심성이 높은 노드는 다른 노드에게 큰 영향을 미친다.

그래프에서의 **매개 중심성**은 한 노드가 다른 노드 간의 최단 경로 상에 있는지를 측정한다. 높은 매개 중심성을 가진 노드는 서브 그룹 간의 통신 다리 역할을 하며, 그들을 분할할 때 이들을 제거하면 통신이 없는 독특한 서브 그래프가 생성된다.

$$Betweeness(v_i) = \frac{\sum_{s \neq v \neq t} l(s,v,t)}{l(s,t)}$$

여기서 $l(s,v,t)$는 노드 s에서 노드 t로 가는 최단 경로를 노드 v를 통과하여 계산한 것이며, 위 식은 v를 통과하는 경로의 총 경로에 대한 비율을 계산한다. 그래프에서 매개 중심성이 높은 노드는 작은 그룹을 연결하는 다리 역할을 하며, 이 노드를 제거하면 서브 그룹을 격리시켜 독특하게 만들 수 있다. 소셜 네트워크에서 매개 중심성은 희소하게 연결된 노드에 강력한 영향을 미치기보다는 다리 역할을 하는 노드를 나타낸다.

서로 다른 속성값을 가진 시드를 선택함으로써 서로 다른 특징을 가진 노드를 매핑하는 것을 보장한다. 동일한 서브 그래프의 노드는 일반적으로 유사한 속성을 가지므로 최댓값과 최솟값을 선택하는 것은 다양한 특징을 캡처하는 데 도움이 된다. 이 과정은 선택한 두 노드를 시작으로 이웃 기반 확장을 사용하여 반복적으로 확장되어 모든 노드가 횡단될 때까지 계속된다. 모든 노드가 탐색 되면 노드 확장 프로세스가 종료된다. 그런 다음 선택한 두 노드를 참조 지점으로 사용하여 그래프를 두 개의 서브 그래프로 분할한다. 이 방법은 보존된 지역 특성을 갖는 의미 있는 서브그래프를 구성하는 데 도움이 된다. 최종적으로, 구성된 여섯 개의 서브 그래프는 그래프 임베딩 단계를 통과하여 각각이 그래프를 나타내는 여섯 개의 개별 벡터로 형성된다.

나. 그래프 임베딩 네트워크

제안된 네트워크는 그래프 동형 네트워크(GIN)를 사용하여 인접 행렬과 노드 특성 행렬을 받아 이웃 노드의 특성을 집계하고 각 노드의 특성을 업데이트한다. 상세한 학습 과정은 다음과 같이 설명된다.

다음 첫 번째 식을 통해 노드 v 및 이웃 노드의 특성 벡터의 합을 얻는다. 그런 다음 두 번째 식을 통해 각 특성은 MLP를 통해 업데이트되며 이웃 노드의 특성과 기존 노드 v의 특성이 추가된다.

$$a_v^{k-1} = \sum_{\forall\, u \in N(v)} h_u^{k-1}$$

$$h_v^k = MLP^k(a_u^{k-1}, h_v^{k-1})$$

그래프 분류를 위해 모델은 각 이웃 노드로부터 데이터를 집계한 후 그래프 수준을 나타내는 통합된 벡터를 생성한다. 그래프 신경망은 이웃 노드로부터의 정보를 연결(concatenation) 연산을 사용하여 각 노드에서 결합하고, 이를 다음 식에서 볼 수 있듯이 분류에 활용한다. 리드아웃 함수는 모든 노드 정보를 하나의 값으로 결합한다.

$$H_g = Concatenation(H_v^k, \forall\, v \in V \mid k = 0, 1, ..., K)$$

제안된 방법에서는 각 서브 그래프에는 고유한 지역 정보를 학습하는 별도의 모델이 필요하다. 이는 각 기준에 대한 지역 정보가 격리되어 다른 서브 그래프와 혼합되지 않도록 보장한다. 또한 각 모델은 고유한 시각에서의 원본 그래프(G)를 입력으로 사용하여 다양한 전역 정보를 추출한다. 제안된 방법은 가중치 공유 없이 개별 모델을 통한 다양한 지역 정보 추출을 허용하면서도 다양한 관점에서의 전역 정보를 고려한다. 따라서 서브 그래프 모델에서 추출된 정보는 $h = (h_{Gp}, h_{sub1p}, h_{sub2p})$로 정의되며, 이에 따라 계산된다. 다음 식은 그래프 신경망을 사용하여 서브 그래프와 원본 그래프를 통한 노드 업데이트 과정을 설명한다.

$$h_p = GNN_p(G, G_{sub1p}, G_{sub2p}), \forall\, p \in Graph\ attributes$$

다. 분류를 위한 어텐션 기반의 리드아웃

세 개의 개별 서브 그래프 모델로부터 나온 지역 및 전역 정보는 $h = h_{Gp}, h_{sub1p}, h_{sub2p}$로 표현됩니다. 그런 다음 통합 네트워크는 각 서브 그래프의 지역 및 전역 특성을 결합한다. 지역/전역 정보(h_n)를 attention 가중치(a_{ij})와 결합하는 과정은 다음 세 개의 식에서 설명되어 있다. 두 개의 서브 그래프와 원본 그래프가 각 모델에 공급되고, 각 그래프가 해당 특성 벡터로 변환된다. 이후에는 일곱 개의 특성 벡터가 있다 ($N = 7$): 세 개의 전역 특성(입력이 동일하므로 하나만 사

용됨)과 통합 전의 여섯 개의 지역 특성이다. 특성 벡터의 크기가 d이면 추출된 특성의 차원은 $(7, d)$가 된다.

$$a_{ij} = softmax_j(h_{ij}) = \frac{\exp[score(h_i, h_j) \cdot W_v h_i]}{\sum_{n=1}^{N} \exp[score(h_i, h_j) \cdot W_v h_i]}$$

$$score(h_i, h_j) = f(W_q h_i, W_k h_j)$$

$$c_i = \sum_{n=1}^{N} a_{in} \cdot h_n$$

이 과정의 끝에서 모델은 10개의 지역 특성과 3개의 전역 특성을 집계하기 위해 각 그래프에 대한 다양한 영향 또는 가중치를 캡처하기 위해 attention score를 계산한다. 이는 다양한 그래프에 대해 네트워크의 일부를 선택적으로 활성화한다.

알고리즘 3은 서브 그래프 및 원본 그래프의 벡터를 통합하여 단일 통합 벡터를 출력하는 과정을 설명한다. 학습 단계는 알고리즘 2의 5-7 라인에서 제시된 작업을 통해 각 그래프 정보가 다른 그래프로부터 받는 영향의 정도를 고려하여 수행된다. 최종 벡터 c를 사용하여 각 정보 간의 영향 정도를 고려하고, W_V, W_Q, W_K 가중치가 손실 함수를 최소화하도록 업데이트된다.

Input $h = \left\{\left\{h^p_{max}, h^p_{min}, h^p_{graph} \ \forall p \in five\ properties\right\}\right\}$)
$\hat{C}$: $Classifier$, CE: $Cross\ Entropy$, lr: $learning\ rate$
1: Initialize W_V, W_Q, W_K, $a[len(h), len(h)]$, $c[len(h)]$
2: for each epoch **do**
3: **for** $i \epsilon h$ **do**
4: **for** $j \epsilon h$ **do**
5: $score(R_i, R_j) = f(W_q R_i, W_k R_j)$
6: $a_{ij} = \frac{exp[score(R_i, R_j) \cdot W_v R_i]}{\sum_{n=1}^{N} exp\,[score(R_i, R_n) \cdot W_v R_i]}$
7: $c_i = \sum_{n=1}^{N} a_{in} \cdot h_n$
8: $c = \sum_{i=1}^{N} c_i$
9: $y_F = \hat{C}(c)$
10: $L_D = CE(y_F, y_{GT})$
11: $W_V, W_Q, W_K = (W_V - lr\nabla L_D), (W_Q - lr\nabla L_D),$
$(W_K - lr\nabla L_D)$ // Update parameter

알고리즘 3. 셀프 어텐션을 활용한 어셈블

4. 실험 결과

가. 데이터셋

우리는 제안된 방법의 효과를 검증하기 위해 여섯 가지 소셜 네트워크 데이터셋 COLLAB, IMDB-Binary, IMDB-Multi, Dblp-ct1, Twitter-egos, Reddit Binary를 사용했다. 각 데이터셋의 세부 사항은 표 7에 나타냈다.

데이터셋	그래프 수	평균 노드 개수	평균 에지 수	클래스 수
COLLAB	5000	74.49	2457.50	3
IMDB-B	1000	19.77	96.53	2
IMDB-M	1500	13.00	65.94	3
Dblp_ct1	755	52.87	320.09	2
Twitch-egos	127094	29.67	86.59	2
Reddit Binary	200	429.63	497.75	2

표 7. 그래프 데이터셋 6개에 대한 세부 사항

나. 베이스라인 기법

실험은 Nvidia DGX station에서 수행되었으며, 해당 시스템은 2,560개의 Nvidia 텐서 코어, Ubuntu 데스크톱 리눅스 OS, 4개의 Tesla V100 (64GB), 그리고 256GB LRDIMM DDR4 메모리를 사용하였다. 소프트웨어 환경으로는 Ubuntu, Python 3.x, 그리고 TensorFlow 2.3 버전을 사용했다. COLLAB의 SOTA 성능을 달성한 그래프 동형 네트워크 (GIN-0)를 사용하여, 모델 구조 변경을 통한 성능 향상이 아닌 제안된 방법을 통한 유용성을 보여주기 위해 이를 활용했다. GIN은 두 개의 그래프 컨볼루션 레이어(filter size = 32, activation = 'relu'), 학습률이 0.001인 Adam 옵티마이저, 그리고 교차 엔트로피 손실 함수를 사용하여 Tensorflow 환경에서 구현되었다.

그래프 딥러닝 방법에 대해 다음과 같은 기준선들과 비교하였다. 그래프 컨볼루션 네트워크

(GCN), 그래프 어텐션 네트워크 (GAT), 그래프 동형 네트워크 (GIN-0), GraphSAGE 네트워크, FactorGCN, 모티프 기반 그래프 컨볼루션 신경망 (M-GCNN), 모티프 기반 어텐션 그래프 컨볼루션 신경망 (MA-GCNN), DropGNN, 그리고 계층적 그래프 캡슐 네트워크 (HGCN)를 사용하여, 제안된 방법이 소셜 및 생물 정보학 그래프 데이터셋에 대한 성능을 강조한다. 통제 실험에서는 제안된 방법의 각 구성 요소의 유효성을 확인하기 위해 서로 다른 종자 조합으로 실험을 수행했다. 추가 실험을 통해 리드아웃의 효과를 보여주기도 한다.

다. 결과 분석

(1) 소셜 네트워크 데이터셋에 대한 전반적인 비교: 실험은 잘 알려진 벤치마크 데이터셋에서 10-fold 교차 검증을 사용하여 최신의 그래프 신경망 분류 방법으로 수행되었다. 이러한 방법들은 state-of-the-art 성능지표를 달성했다는 점에서 제안된 방법의 중요성을 검증한다.

표 8은 성능의 평균 정확도와 표준편차를 보여준다. 여섯 가지 지역 정보를 서로 다른 attention 가중치로 통합하는 우리의 방법은 상당한 정확도 향상을 이루어낸다. 결과적으로 제안된 방법은 HGCN과 비교하여 COLLAB, Reddit Binary에서 각각 3.49%p 및 0.10%p를, MA-GCCN과 비교하여 IMDB-Binary 및 IMDB-Multi에서 각각 13.23%p 및 27.87%p의 향상된 정확도를 달성한다. 이 두 모델은 현재 소셜 네트워크 분류의 state-of-the-art다. 또한 Dblp-ct1 및 Twitter-egos 데이터셋의 경우, HGCN과 비교하여 각각 2.54%p 및 1.36%p의 성능이 향상되었다.

모델	COLLAB	IMDB-D	IMDB-M	Dblp_ct1	Twitch-egos	Reddit Binary
GCN	79.36±1.94	71.61±2.11	50.61±3.69	64.99±2.93	66.94±0.95	91.51±1.23
GAT	75.80±1.60	70.50±2.30	47.80±3.10	63.53±1.21	67.10±0.72	92.60±2.17
GIN-0	79.00±1.70	74.00±3.40	51.90±3.80	61.67±4.99	66.27±1.66	90.40±2.50
GraphSAGE	80.26±1.22	74.91±1.96	51.66±2.70	62.81±1.88	68.41±2.89	91.54±1.99
FactGCN	81.20±1.40	75.30±2.70	52.01±4.28	61.17±3.62	70.84±4.08	90.33±1.78
M-GCNN	80.08±2.12	75.10±3.14	52.19±2.66	62.34±1.03	71.28±1.42	88.06±1.29
MA-GCNN	83.13±3.09	77.20±2.96	53.77±3.11	63.84±2.31	73.70±1.81	90.44±2.18
Drop-GNN	79.02±3.08	75.70±4.20	51.40±2.80	61.15±4.31	75.90±2.13	80.08±2.51
HGCN	82.86±1.81	77.20±4.73	52.80±2.45	65.50±2.01	76.80±1.15	93.15±1.58
Ours	**86.35±2.95 (+3.49%)**	**90.43±4.89 (+13.23%)**	**81.64±1.48 (+27.87%)**	**68.04±1.96 (2.54%)**	**78.16±0.66 (+1.36%)**	**93.25±2.18 (0.10%)**

표 8. 여러 모델에 대한 평균 분류 정확도와 표준편차 비교
가장 높은 지표에 대해서는 굵은 글씨를, 두 번째 지표에 대해서는 밑줄 표시함

(2) 세 가지 그래프 속성에 대한 Ablation Study: 표 9는 각 데이터셋에 사용된 서로 다른 시작점 또는 '시드'에 따른 성능 변화를 보여준다. 제안된 방법은 사용 가능한 모든 정보를 사용할 때 가장 우수한 성능을 발휘한다. 이는 여러 가지 시드를 혼합하여 그래프의 상세 정보를 유지하고, 이를 통해 그래프 요약 또는 임베딩을 생성할 때 도움이 되기 때문이다. 그래프의 모든 세부 정보를 최대한 활용하면 더 나은 결과를 얻을 수 있다. 특히 두 가지 특정 유형의 시작점을 사용할 때 결과는 세 가지 유형을 함께 사용하는 것과 거의 동등하다. 이는 이 두 시작점이 각자 그래프의 다른 그러나 보완적인 세부 정보를 캡처한다는 것을 의미한다. 이러한 결합 효과가 제안된 방법에서 그래프를 분할하고 통합하는 데 실제로 결과를 높인다.

De	B	C	COLLAB	IMDB-D	IMDB-M	Dblp_ct1	Twitch-egos	Reddit Binary
✓			0.7850	0.7600	0.5540	0.6400	0.7039	0.8000
	✓		0.7750	0.7400	0.5237	0.6502	0.7035	0.8000
		✓	0.7650	0.7300	0.5015	0.6400	0.7637	0.7500
✓	✓		0.7866	0.7900	0.6021	0.6510	0.7264	0.8000
✓		✓	0.7843	0.7800	0.6018	0.6525	0.7688	0.7500
	✓	✓	0.7839	0.8000	0.6169	0.6518	0.7715	0.8000
✓	✓	✓	**0.7950**	**0.8300**	**0.6667**	**0.6533**	**0.7799**	**0.8500**

표 9. COLLAB, IMDB-B, IMDB-M, Dblp-ct1, Twitter-egos, Reddit Binary에 대한 A/B 테스트의 평균 분류 정확도 및 표준편차 비교
("De"는 차수를 나타내고, "B"는 매개 중심성을 나타내며, "C"는 근접 중심성을 나타냄
체크 마크는 해당 벡터가 분류에 사용되었음을 나타냄
가장 높은 지표에 대해서는 굵은 글씨를, 두 번째 지표에 대해서는 밑줄 표시함)

(3) Attention 기반 리드아웃에 대한 Ablation Study: 표 10은 그래프 특성의 변화 및 각 그래프가 이러한 특성을 통합하는 방법을 비교한 결과를 보여준다. 우리의 새로운 Attention 기반 방법이 다른 방법들과 비교하여 어떻게 성과를 내는지 확인하기 위해 다양한 데이터셋에서 테스트했다. 결과는 제안된 방법이 더 나은 성능을 발휘함을 보여준다. 일부 다른 방법은 분류할 때 정보를 지나치게 복잡하게 만들거나 지나치게 간단하게 만들어서 일부 지역 세부 정보를 놓칠 수 있다. 반면에 우리의 방법은 각 지역 정보를 개별적으로 주의 깊게 살펴본다. 이 방법은 단순히 모든 것을 섞는 것이 아니라 각 지역 세부 정보의 중요성을 존중하고 유지하기 때문에 더 나은 결과를 얻을 수 있다.

기법	Concatenation	Ensemble with Average	Ensemble with Element-wise Attention	**Ensemble with Channel-wise Attention (Ours)**
COLLAB	76.30±1.53	77.46±0.81	83.20±1.65	**86.30**±1.89
IMDB-D	70.07±2.16	70.48±5.26	87.15±2.16	**90.15**±3.86
IMDB-M	51.40±2.96	48.33±2.40	76.37±3.64	**81.48**±2.44
Dblp_ct1	55.54±2.81	56.90±0.92	65.15±2.10	**67.54**±1.15
Twitch-egos	70.33±0.61	71.37±5.79	74.88±4.00	**79.20**±0.80
Reddit Binary	83.71±4.73	82.39±1.13	87.70±3.85	**90.99**±2.11

표 10. 기법에 따른 평균 분류 정확도와 표준편차 비교
(가장 높은 지표에 대해서는 굵은 글씨를, 두 번째 지표에 대해서는 밑줄 표시함)

본 연구에서는 그래프를 세 가지 핵심 속성을 사용하여 분할하고, 각 속성에 대해 별도의 그래프 컨볼루션 네트워크를 사용하여 그래프의 다양한 지역 세부 정보를 캡처하는 방법을 제안한다. 이 방법은 세 가지 구별되는 측정에서 파생된 self-attention 메커니즘을 활용하여 여섯 가지 소셜 네트워크에서 기존의 최고 수준의 방법을 능가한다. 각 범주에 대한 attention distributed 그래프를 사용함으로써 어떤 세부 사항이 그래프를 분류하는 데 가장 효과적인지 확인할 수 있다. 우리의 연구 결과는 더 적은 노드를 사용하는 소셜 네트워크 데이터에서 다양한 지역 패턴을 강조하는 방법의 강점을 강조한다. 이는 특히 노드 간의 연결만을 의존하는 소셜 네트워크 분류에서 뚜렷하게 나타난다.

향후에는 시간과 공간을 줄이는 방향으로 이 방법을 개선할 계획이다. 실제 그래프 연구에서는 계산 시간이 가능한 한 짧아야 하며, 주요 초점은 가장 정확하고 의미 있는 결과를 얻는 데 있어야 한다. 이 관점에서 실제 세계 사용에 잘 맞는 효율적이고 현실적인 방법을 개발하기를 희망한다. 이 분야에서 나아감에 따라 주요 목표는 여전히 현실적인 상황에 맞는 실용적이고 영향적인 그래프 기반 솔루션을 제공하는 것이다.

제 3 장. 그래프기반 추천 알고리즘 동향 분석 및 조사

제 1 절. 라이프로그 그래프 구조를 통한 개개인 사용자의 행동 패턴별 설명

1. 라이프로그 그래프의 노드 예측

그래프의 노드 예측은 라이프로그 데이터셋에서 중요한 작업 중 하나다. 이러한 작업의 목적은 일반적으로 그래프의 미래 상태를 예측하거나, 그래프 내에 존재하지 않는 (혹은 숨겨진) 노드의 속성이나 범주를 예측하는 것이다. 그래프에서 노드 예측을 수행한다는 것은 노드의 특성(예: 사용자의 행동 유형), 노드의 그룹(예: 사용자의 소셜 그룹)을 예측하는 것을 의미할 수 있다. 그림 6에서와 같이 알려지지 않은 회색 노드의 클래스를 이웃 노드들인 빨간색과 파란색의 노드들로부터 정보를 전파받아 회색 노드의 클래스를 예측 또는 분류를 하는 것이다. 라이프로그 그래프에서의 노드 예측은 다음과 같은 예시가 있을 수 있다.

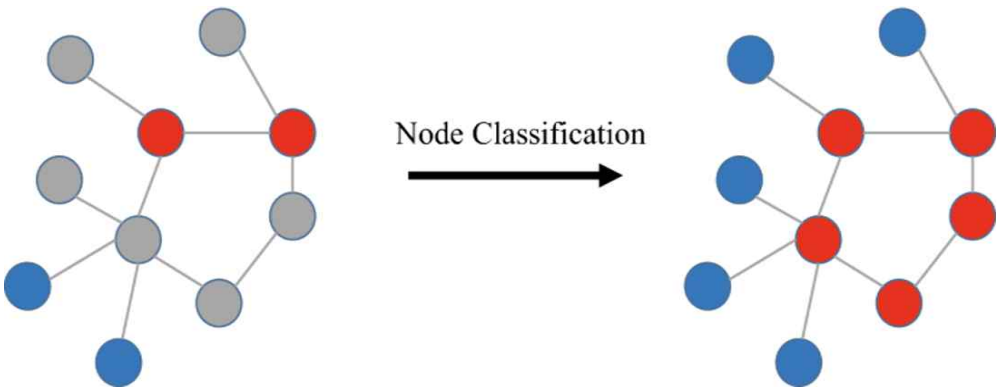


그림 6. 그래프 노드 분류에 대한 예시

가. 개인화 추천

라이프로그 데이터를 활용하여 사용자의 선호도, 관심사 등을 학습하고, 이를 바탕으로 개인화된 추천을 제공할 수 있다. 예를 들어, 사용자의 과거 위치 데이터를 바탕으로 미래에 방문할 가능성이 큰 장소를 예측할 수 있다.

나. 사회적 상호작용 예측

노드 간의 관계를 분석하여 사람들 사이의 사회적 상호작용을 예측한다. 예를 들어, 누가 언제 친구가 될 가능성이 큰지 예측할 수 있다.

다. 행동 패턴 인식

사용자의 행동 또는 활동 패턴을 식별하여, 그래프상의 노드로 표현된 개인의 일정이나 활동을 예측한다. 이는 건강 관리 앱에서 운동 또는 식습관 패턴을 분석하는 데 사용될 수 있다.

라. 비정상 행동 탐지

노드의 예측된 행동 패턴이 기존의 행동 그래프와 크게 달라질 경우, 이를 비정상적인 활동으로 간주하여 경고를 줄 수 있다. 이는 보안 또는 건강 모니터링에 유용하다.

마. 건강 모니터링

사용자의 생체 신호와 활동 데이터를 기반으로 건강 상태의 변화를 예측할 수 있다. 예를 들어, 심박수와 활동 수준을 분석하여 스트레스 수준이나 운동 부족과 같은 건강 관련 문제를 예측할 수 있다.

이처럼 그래프의 노드를 예측하는 것은 라이프로그 그래프에서 중요한 태스크 중 하나이다.

2. 노드 예측을 위한 그래프 신경망 프레임워크 동향 및 조사

그림 7과 같이 그래프 신경망에 관한 연구는 ICLR 2022년도에서 세 번째로 많은 키워드가 나올 정도로 연구가 활발하게 이루어지고 있다. 또한 그림 8에서 알 수 있듯이 논문 제목에 포함된 키워드 중 그래프는 두 번째로 가장 많은 키워드로 통계가 밝혀졌다.

그중에서도 노드 예측/분류를 위한 그래프 신경망 프레임워크 동향에 대해서 정리하면 표와 같다. 2017년도에 발표된 GCN부터 2021년도에 발표한 확산 방정식 기반의 그래프 신경망인 GRAND까지 다룬다.

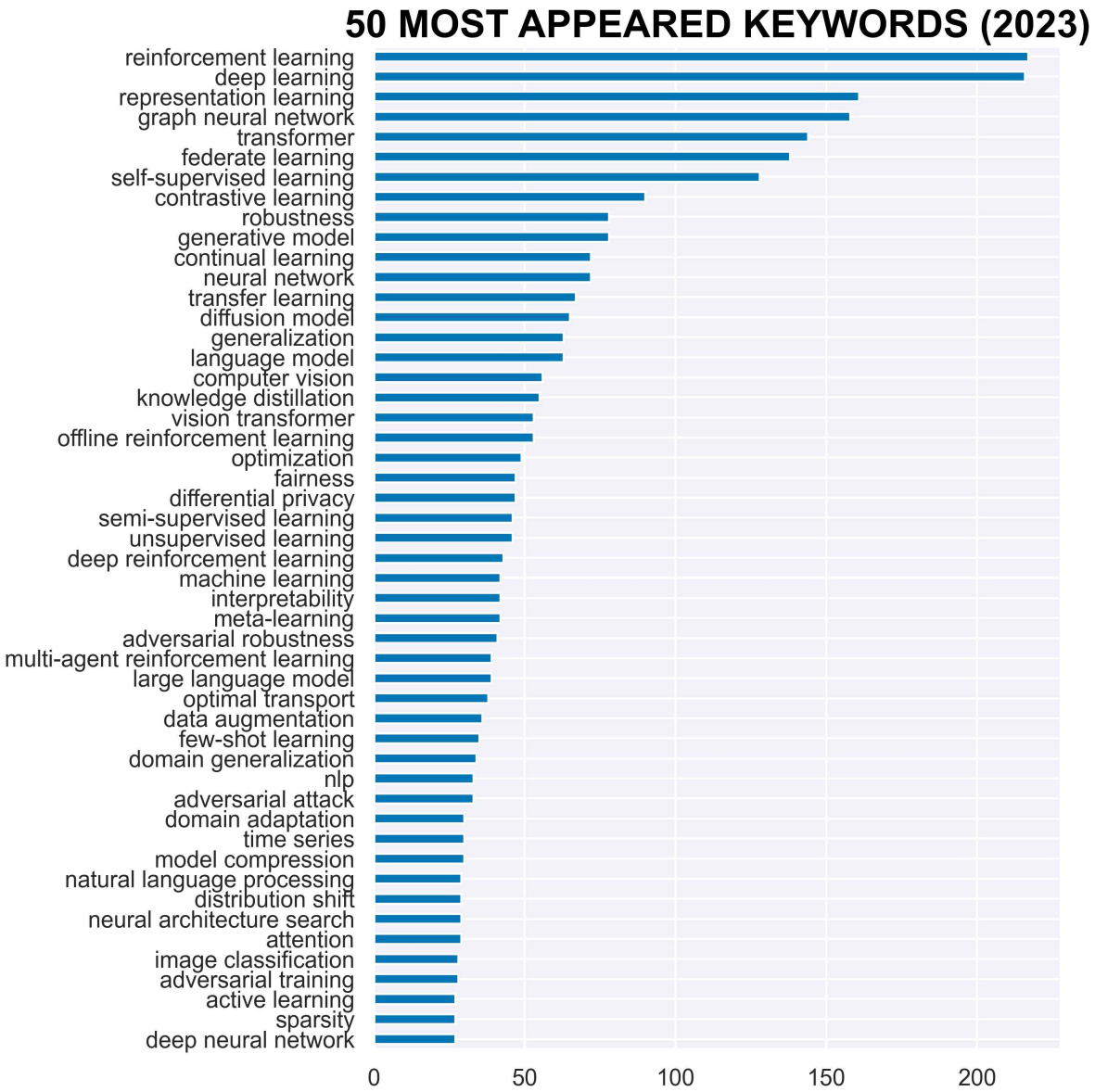


그림 7. ICLR 2023년의 제출 논문 중 상위 50개의 키워드 통계

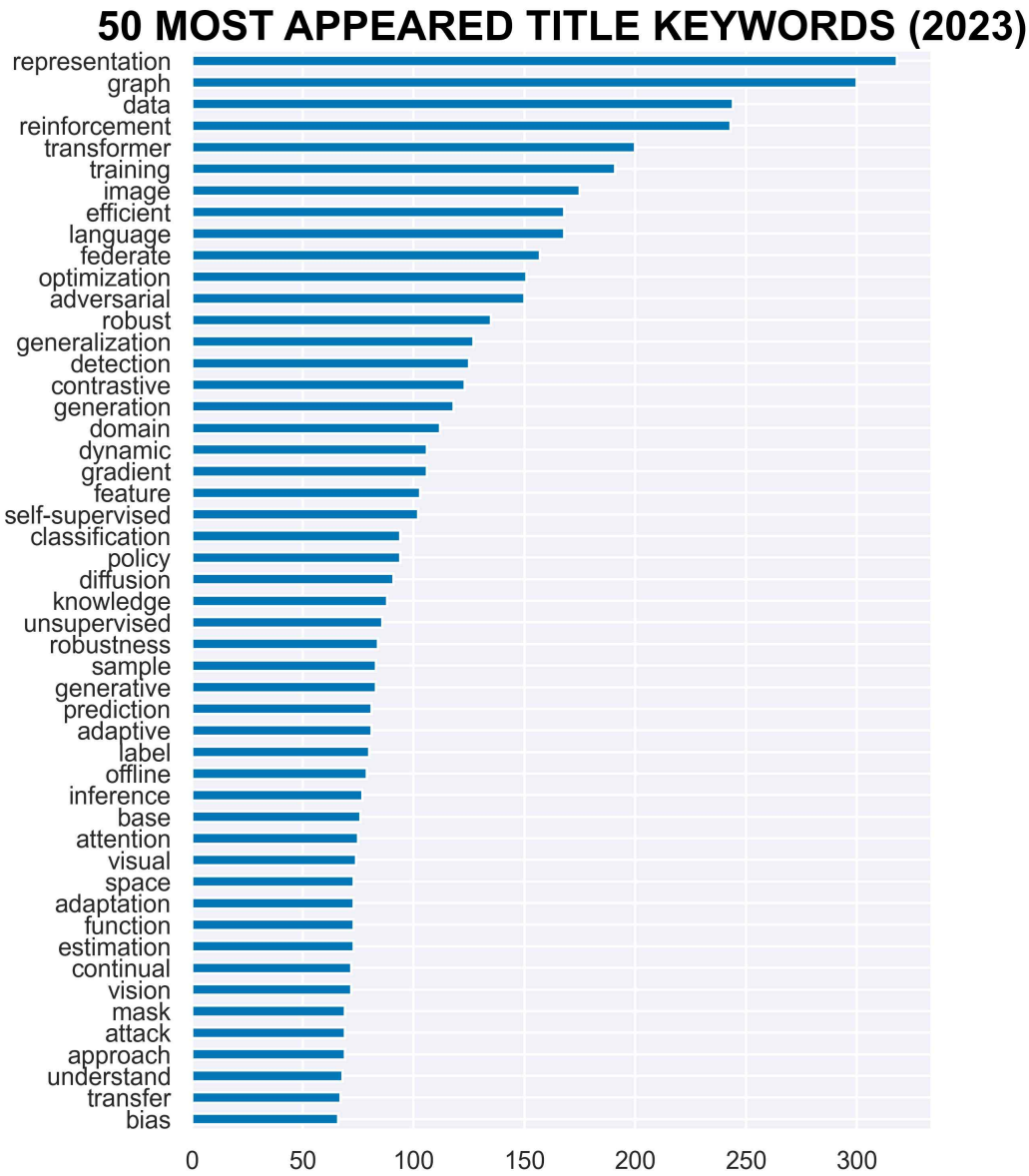


그림 8. ICLR 2023년의 제출 논문 제목에 포함된 키워드의 상위 50개 통계

먼저 그림 9의 위 행은 GCN[12]을 나타낸다. GCN은 K 레이어 전체에 걸쳐 피쳐 벡터를 반복적으로 변환한다. 그런 다음 최종 표현에 선형 분류기를 적용한다. 그리고 아래 행은 SGC[13]이며 전체 절차를 간단한 피쳐 전파 단계와 이어서 표준 로지스틱 회귀로 축소한다. SGC는 비선형성을 연속적으로 제거하고 연속 레이어 간의 가중치 행렬을 축소하여 이러한 과도한 복잡성을 줄인다. SGC는 선형 분류기가 뒤따르는 고정 저역 통과 필터에 해당한다.

표 11. 그래프 기반 추천 알고리즘 동향

논 문	연 도	학 회	모 델
Semi-Supervised Classification with Graph Convolutional Networks	2017	ICLR	GCN
Graph Attention Networks	2018	ICLR	GAT
Inductive representation learning on large graphs	2017	NeurIPS	GraphSAGE
Simplifying graph convolutional networks	2019	ICML	SGC
Simple and Deep Graph Convolutional Networks	2020	ICML	GCNII
Continuous Graph Neural Networks	2021	ICML	CGNN
Beyond Low-frequency Information in Graph Convolutional Networks	2021	AAAI	FAGNN
Adaptive universal generalized pagerank graph neural network	2021	ICLR	GPR-GNN
GRAND: Graph Neural Diffusion	2021	ICML	GRAND

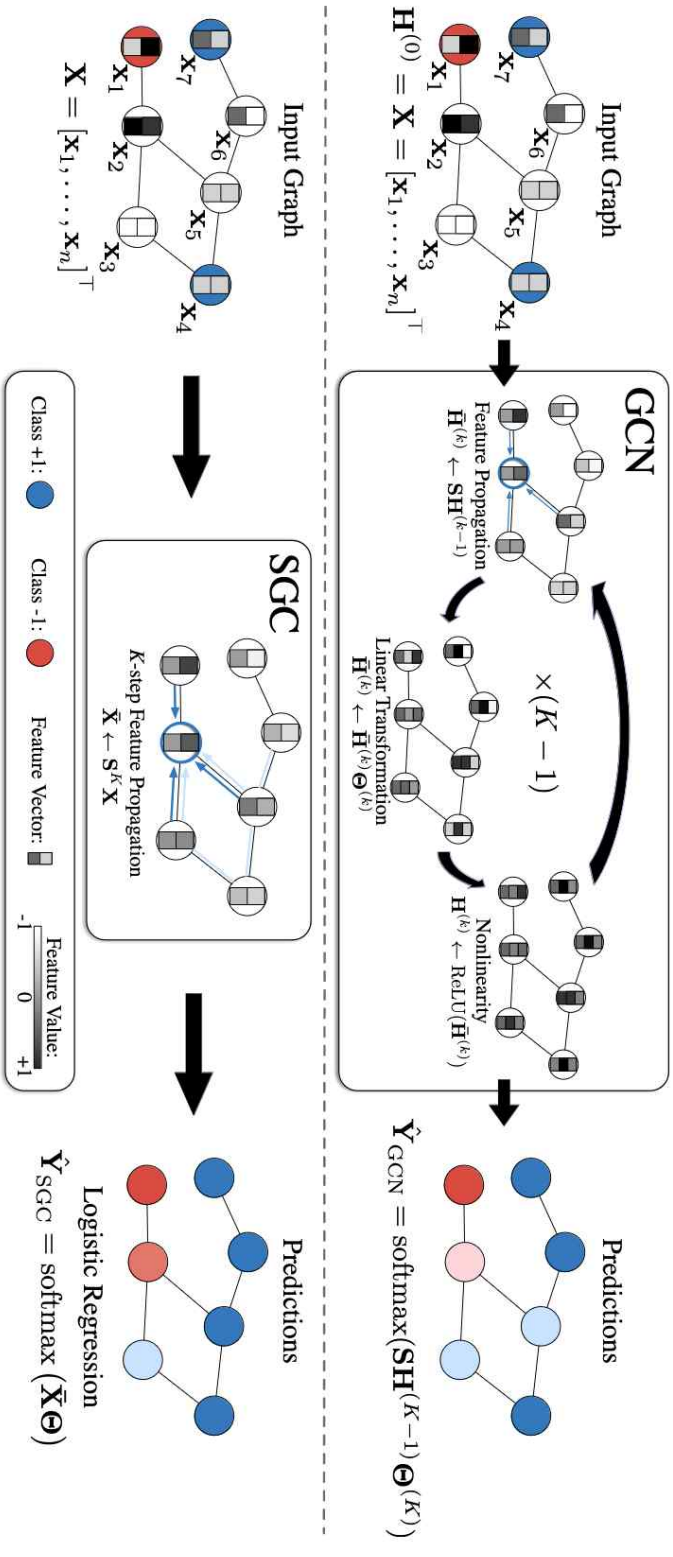


그림 9. GCN 대 SGC의 도식적 레이아웃 SGC

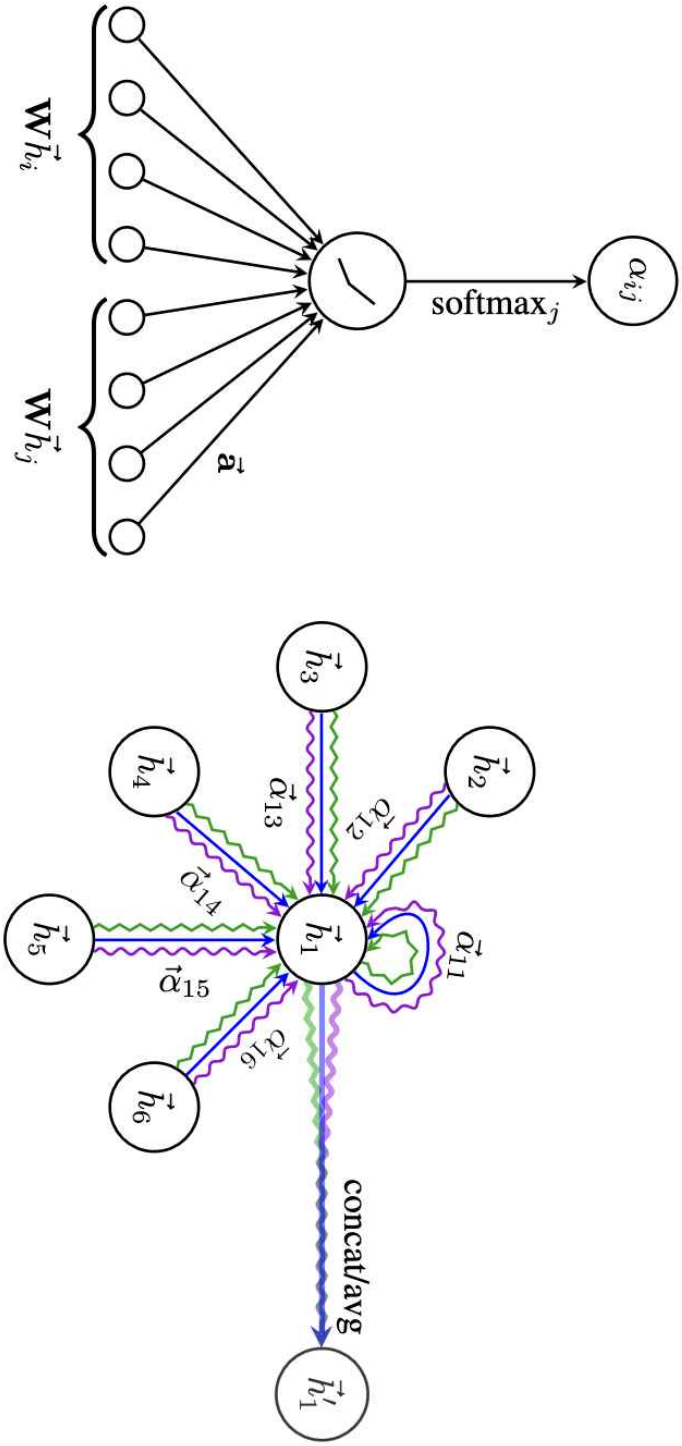


그림 10. 왼쪽 그림은 attention 메커니즘인 $a(Wh_i, Wh_j)$,
(오른쪽 그림은 노드 1과 이웃 노드들에 대한 멀티-헤드 attention)

Graph Attention Networks (GAT)[14]는 2018년도에 발표된 그래프 신경망으로 가장 인기 있는 그래프 신경망 중 하나이다. GAT는 그래프 구조화된 데이터에서 노드의 특성을 학습할 때 각 이웃의 중요도를 가중치로 계산하는 방식을 사용한다. GAT는 그래프의 각 노드에 대해 다른 이웃들이 주는 영향을 자동으로 학습하며, 각 이웃의 중요도를 동적으로 결정한다. 이를 위해 attention 메커니즘을 사용하여 각 노드에 대한 이웃들의 중요도를 계산하고, 이 정보를 바탕으로 노드의 새로운 특성을 생성한다.

GAT의 핵심은 각 노드에 대해 이웃 노드들로부터 얻은 정보를 가중치 있는 합으로 표현하는 것이다. 이 가중치는 해당 이웃 노드가 현재 노드의 새로운 특성을 계산하는 데 얼마나 중요한지를 나타낸다. 높은 가중치는 더 큰 중요도를 의미한다.

GAT의 기본 수식은 아래와 같다.

$$\text{Attention coefficients: } \alpha_{ij} = \frac{\exp(LeakyReLU(a^T[Wh_i \mid\mid Wh_j]))}{\sum_{k \in N_i} \exp(LeakyReLU(a^T[Wh_i \mid\mid Wh_k]))}$$

$$\text{Node Update: } h_i' = \sigma(\sum_{j \in N_i} \alpha_{ij} Wh_j)$$

여기서 h_i는 노드 i의 입력 피쳐 벡터, h_i' 는 업데이트된 피쳐 벡터, W는 학습 가능한 가중치 행렬, a는 attention 메커니즘의 가중치 벡터이다. α_{ij}는 노드 j 가 노드 i의 특성을 업데이트하는 데 이바지하는 정도 (attention 가중치) 그리고 σ는 활성화 함수이다.

GAT 모델은 Cora, Citeseer 및 Pubmed 인용 네트워크 데이터 세트와 단백질-단백질 상호 작용 데이터 세트 등 4개의 그래프 벤치마크에서 좋은 성능을 달성한다.

GraphSAGE[15]는 노드 피쳐 정보(예: 텍스트 속성)를 활용하여 이전에 볼 수 없었던 데이터에 대한 노드 임베딩을 효율적으로 생성하는 일반적인 귀납적 프레임워크이다. GraphSAGE는 노드 특성을 학습하기 위해 노드의 이웃으로부터 샘플링하고 집계하는 방식을 사용하는 그래프 신경망의 한 형태이다. 이 방법은 노드의 이웃을 고정된 크기로 샘플링하여 확장성 있는 학습을 가능하게 한다. GraphSAGE는 다양한 집계 함수(예: 평균, 최댓값, LSTM)를 사용하여 이웃의 정보를 합치고, 결과적으로 각 노드의 특성을 업데이트한다.

GraphSAGE는 특히 큰 그래프에 적용할 때 유용하며, 전체 이웃을 고려하는 대신 샘플링을 통

해 계산 비용을 줄인다. GraphSAGE의 수식은 아래와 같다.

$$h_{N_i}{}' = AGGREGATE_k\left(h_j, \forall j \in N_i\right)$$

$$h_i{}' = \sigma\left(W \cdot CONCAT(h_i, h_{N_i}{}')\right)$$

여기서 h_i는 노드 i의 현재 특성 벡터, h_{N_i} 는 노드 i의 이웃들의 특성 벡터를 집계한 것, AGGREGATE는 사용자가 정의한 k번째 레이어의 집계 함수, W는 학습 가능한 가중치 행렬, 그리고 σ는 활성화 함수이다. GraphSAGE는 다음과 같은 절차로 작동한다.

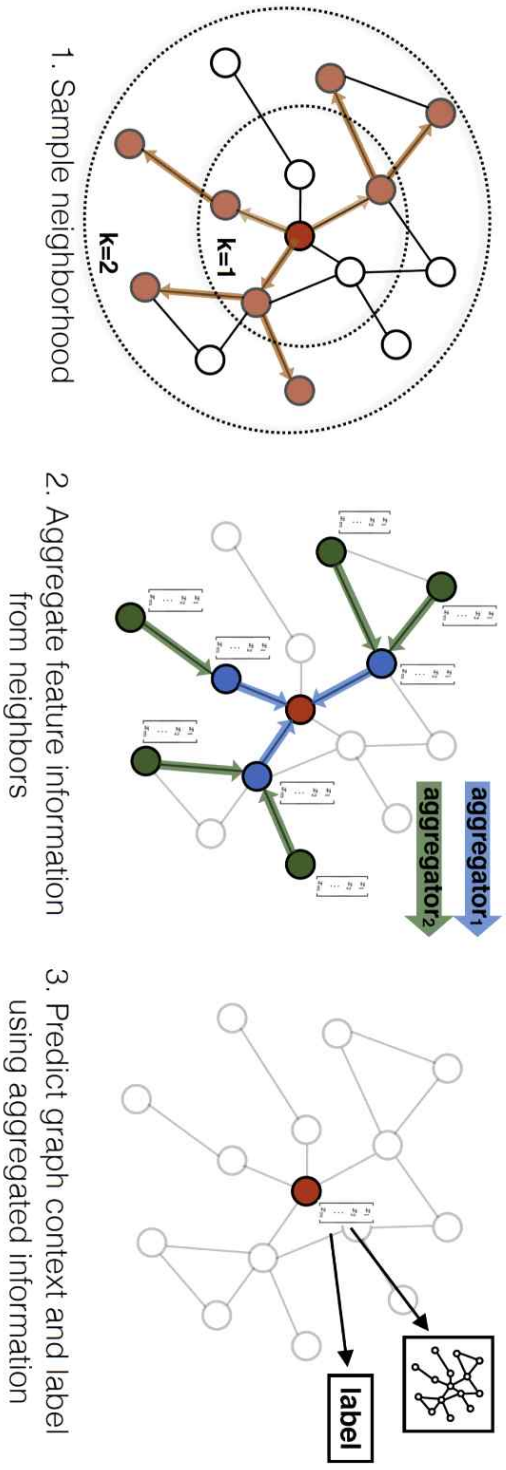


그림 11. GraphSAGE 샘플 및 집계 접근 방법

가. 각 노드에 대해 고정된 수의 이웃을 샘플링한다.

나. 샘플링된 이웃의 특성을 집계하는 집계 함수를 사용하여 이웃의 정보를 결합한다.

다. 결합된 이웃 정보와 현재 노드의 특성을 결합한다.

라. 결합된 벡터에 대해 가중치 행렬을 적용하고 비선형 활성화 함수를 통과시켜 새로운 노드의 특성을 생성한다.

라. 필요한 경우, 여러 계층을 통해 위의 과정을 반복하여 더 복잡한 특성을 학습한다.

GCNII[16]는 간단하면서도 효과적인 두 가지 기술인 초기 잔차와 Identity 매핑을 사용하여 바닐라 GCN 모델을 확장한다. GCNII의 수식은 다음과 같다.

$$H(l+1) = \sigma(((1-\alpha_l)\widehat{A}H(l) + H(0)))(((1-\beta_{l)}H(l)\,W(l)))$$

여기서 왼쪽 항이 초기 잔차 연결에 해당하고 Identity 매핑은 두 번째 항에 해당한다. α_l는 0.1 또는 0.2 로 설정되는 하이퍼파라미터이다. $\beta_l = \log(\lambda/l+1) \approx \lambda/l$에서는 λ가 하이퍼파라미터이고 0.5로 설정된다. GCNII는 두 기술이 과도하게 평활화되는 문제를 효과적으로 완화한다는 이론적, 경험적 증거를 제공한다.

그림 12[17]는 Generalized PageRank GNN을 제안한다. GPR-GNN은 기존 GNN을 괴롭히는 두 가지 문제인 오버스무딩과 이종성 (헤테로필리; heterophily) 그래프 문제를 해결하는 것을 목표로 그래프 신경망(GNN)에 Generalized PageRank (GPR)를 통합하려고 시도한다. GPR-GNN의 핵심 PageRank score를 계산하는 k번째 반복에 가중치 요소를 도입하는 것이다. GPR-GNN는 벤치마크 데이터 세트를 사용하여 광범위한 평가를 수행하고 일부 기존 GNN 방법에 대한 개선 사항을 보여준다.

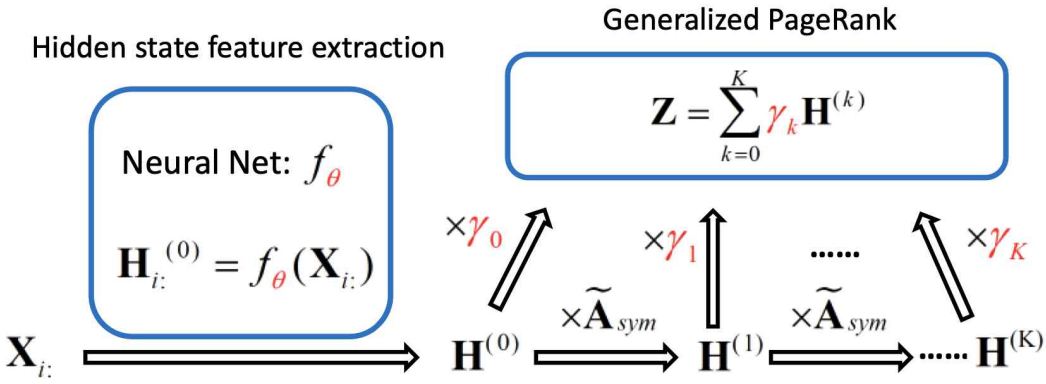


그림 12. GPR-GNN 모델의 오버뷰

CGNN[18]은 그래프 신경망과 기존 동적 시스템 간의 연결을 기반으로 한다. 이 모델은 이산 동역학을 갖춘 기존 그래프 신경망을 일반화하는 Continuous Graph Neural Network (CGNN)을 제안한다. 그래프의 기존 확산 기반 방법 (예: PageRank 및 소셜 네트워크의 전염병 모델)에서 영감을 받아 노드의 변화량을 현재 노드 표현, 이웃 표현 및 노드의 초기 값의 조합으로 정의한다. CGNN은 오버스무딩을 어느 정도 완화할 수 있어서 더 깊은 네트워크를 구축할 수 있으며, 이를 통해 노드 간의 장거리 종속성을 캡처할 수 있다. CGNN은 크게 인코더, ODE 솔버, 디코더 세 가지 부분으로 이루어져 있다. 먼저 인코더는 각 노드 피쳐를 잠재 공간으로 보내주는 역할로, 노드 피쳐 행렬로 변환해준다. 그 후 미리 준비된 ODE와 초깃값에 대한 initial value problem을 풀어주는 ODE 솔버를 거쳐, 최종 시간에 대한 노드 표현을 만든다. 마지막으로 디코더를 거쳐 노드-레이블 행렬로 변환된다.

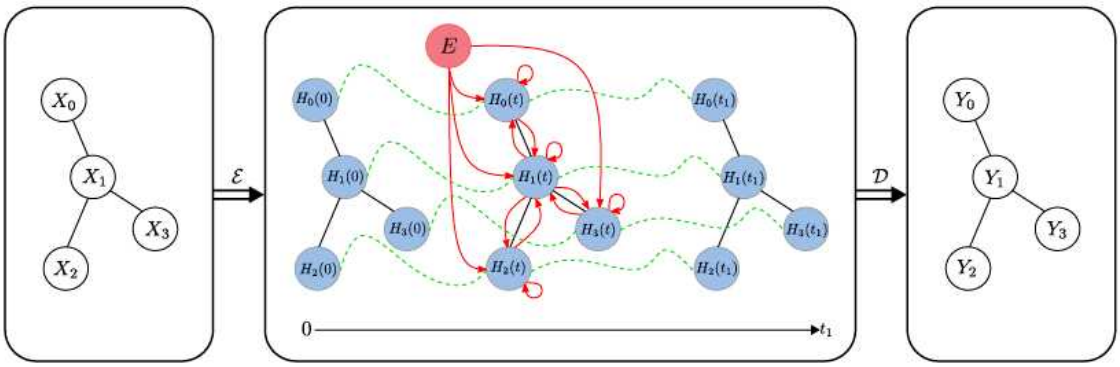
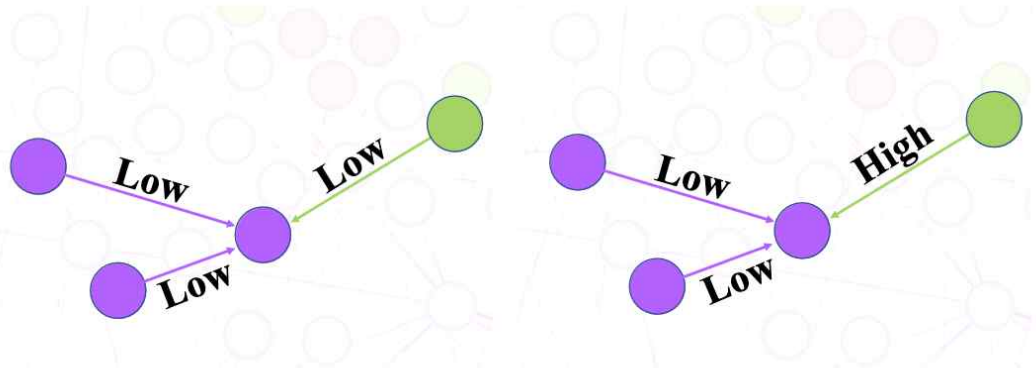


그림 13. CGNN의 오버뷰

그림 14에서와 같이 FAGCN[19]은 GNN의 저주파 정보 외에 더 많은 정보를 적응적으로 학습할 수 있다. 대부분의 기존 GNN은 일반적으로 노드 기능의 저주파 신호를 활용하는 한계가 있다. 이 문제를 해결하기 위해 메시지 전달 과정에서 다양한 신호를 적응적으로 통합할 수 있는 자체 게이팅 메커니즘을 갖춘 새로운 FAGCN (Frequency Adaptation Graph Convolutional Networks)을 제안한다. FAGCN은 과도한 스무딩 문제를 완화할 수 있다는 것을 보인다.

GRAND[20]는 그래프 신경망을 연속 확산 프로세스로 접근하고 그래프 신경망을 기본 PDE의 이산화로 처리하는 모델인 그래프 확산 신경망을 제시한다. GRAND는 오버스무딩과 병목 현상과 같은 그래프 학습 모델의 일반적인 문제를 해결할 수 있다. GRAND의 선형 및 비선형 버전은 벤치마크 데이터셋에서 경쟁력 있는 결과를 달성한다.



(b) Existing GNNs (c) FAGCN

그림 14. 기존 그래프 신경망과 FAGCN 의 비교

제 2 절. 개인화를 위한 그래프기반 추천 알고리즘의 체계적인 조사 및 분석

1. 그래프 필터와 그래프 신경망

우리 주변에는 이제 엄청난 양의 데이터가 존재한다. 건강 모니터링 장치에서 수집되는 건강 데이터, 다양한 앱에서 수집되는 개인 사용 기록 데이터, 금융 및 은행 데이터, 소셜 네트워크, 이동성과 교통 패턴, 마케팅 선호도 등 인간 삶의 거의 모든 측면에서 데이터가 기록되고 있다. 이러한 상호작용의 복잡성은 데이터가 이제 표준 구조에 의존하지 않는 불규칙하고 복잡한 구조에 상주한다는 것을 의미한다.

그래프는 그러한 데이터와 사용자들 사이의 복잡한 상호 작용을 모델링하는 기능을 제공한다. 예를 들어, 트위터의 사용자는 정점(노드)으로 모델링 될 수 있고 친구 연결은 간선(엣지)로 모델링 될 수 있다. 이러한 정점에 속성을 추가하고 그래프의 신호를 모델링하는 방법을 탐구하기 위해 연구하는 분야가 그래프 신호 처리(Graph Signal Processing, GSP)이다. 그래프의 정점들은 일반적인 간선이 아닌 그래프 신호 처리에 의해 그림 9와 같이 가중치가 주어진 간선으로 연결된다. 그래프 신호 처리를 위해서는 푸리에 변환, 필터링 및 주파수 처리와 같은 고전적인 신호 처리 도구부터 최신 딥러닝 기술까지 다양한 방법이 사용된다.

스펙트럴 그래프 필터 (Spectral Graph Filter)는 그래프 신호 처리 분야에서 최근 큰 주목을 받고 있다. 스펙트럴 (Spectral)은 행렬의 고윳값 (Eigenvalue)를 의미한다. 스펙트럴 그래프 필터는 최신 그래프 신경망의 핵심 구성 요소이다. 주어진 그래프를 스펙트럴 도메인으로 해석하는 데에는 라플라시안 행렬이 사용된다. 그림 15에는 그래프, 차수 행렬 D, 인접 행렬 A, 라플라시안 행렬 L의 관계가 나타나 있다. 인접 행렬은 그래프를 행렬 형태로 나타낸 것이다. 결국, 그래프에서 어느 꼭짓점들이 변으로 연결되었는지 나타내는 정사각 행렬이다. 차수 행렬은 각 대각 행렬을 제외하고는 0의 값을 가진다. 즉, 특정 노드의 차수(Degree) 정보를 표현하는 행렬이다. 그리고 라플라시안 행렬은 차수행렬 - 인접행렬이다. 라플라시안 행렬은 다음과 같은 특징을 갖는다. 1) 대각성분을 중심으로 양 값이 대칭적이며, 2) 대각 요소 성분을 제외하고 음의 값을 갖고 있으며, 3) 주 대각성분의 값이 행의 다른 값보다 크거나 같다. 라플라시안 행렬은 보통 정규화를 거친 후에 사용된다. 정규화된 라플라시안 행렬 L^{sym}의 경우 $L^{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$으로 구해진다.

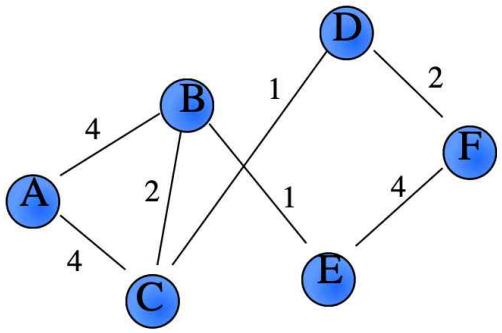
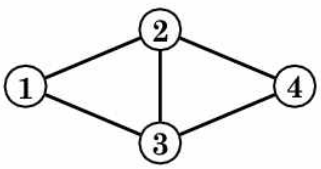


그림 15. 가중 그래프 (Weighted Graph)

이러한 라플라시안 행렬은 그래프의 평탄함 (Smoothness)를 측정하는 데에 사용된다. 한 정점의 정점 벡터를 x라고 했을 때, 라플라시안 행렬을 곱해준 Lx는 해당 정점이 이웃 정점과 얼마나 차이가 나는지 측정하는 척도가 된다. 또한, x^TLx는 각 정점과 이웃 정점 사이의 차이 제곱의 합이 된다. 따라서, 만약 이웃 정점끼리 값이 비슷하다면 x^TLx는 값이 작을 것이고, 이웃 정점끼리 값이 크게 다르다면 x^TLx는 값이 클 것이다. 이러한 성질로 라플라시안 행렬은 그래프의 부드러운 정도를 측정할 수 있다.

라플라시안 행렬은 대각성분 기준으로 좌우가 대칭인 양의 정부호 행렬 (Positive semi-definite matrix)이고, 모든 고윳값이 실수이다. 따라서 라플라시안 행렬은 다음과 같은 고윳값 분해가 가능하다. $L = U\Lambda U^T$. 이때 고윳값 행렬인 $\Lambda = diag[\lambda_1, \lambda_2, \cdots, \lambda_n]$의 고윳값을 보면, 고윳값이 작을수록 그래프에 대한 의미를 많이 갖고 있다고 해석된다. U는 고유벡터를 담고 있는 행렬이다.

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

그림 16. 그래프의 라플라시안 행렬 예시

스펙트럴 그래프 필터(Spectral Graph Filter)는 이러한 고윳값과 고유벡터를 활용한다. 라플라시안 행렬 L에 대해 고윳값 분해 $L = U\Lambda U^T$를 수행했을 때, 고유벡터 행렬을 U, 고윳값을 담은 행렬을 Λ라고 하자. 이때의 스펙트럴 그래프 필터는 다음 수식과 같다.

$$h(L) = Uh(\Lambda)U^T = Udiag[h(\lambda_1),\cdots,h(\lambda_n)]U^T$$

그래프 신호 x에 대한 스펙트럴 필터 $h(L)$를 곱해주는 것이 곧 그래프 컨볼루션(Graph Convolution)이며, 다음 수식과 같다.

$$h(L)x = Uh(\Lambda)U^Tx = Udiag[h(\lambda_1),\ldots,h(\lambda_n)]U^Tx$$

기존 신호 처리 분야에서 자주 사용되는 필터는 다음 그림 20에 나타나 있다. (a)에 나타난 저역 통과 필터(Low-pass filter)는 일정 주파수보다 낮은 주파수의 신호를 통과시키고, 일정 주파수보다 높은 주파수의 신호를 감쇠시킨다. 신호 처리 도메인에서 신호는 종종 매끄럽고 낮은 주파수로 이루어지며, 잡음은 종종 매끄럽지 않고 높은 주파수로 이루어진다. 따라서, 가장 많이 쓰이는 필터는 노이즈 제거를 위해 그래프 신호의 부드러움을 촉진하는 저역 통과 필터이다. (b) 고역 통과 필터 (High-pass filter)는 일정 주파수보다 높은 주파수의 신호를 통과시키고, 일정 주파수보다 낮은 주파수의 신호를 감쇠시킨다. (c) 대역 통과 필터(Band-pass filter)는 고역 통과 필터(High-pass filter)와 저역 통과 필터 (Low-pass filter)를 종속시킨 것이며, (d) 대역 거부 필터(Band-reject filter)는 고역 통과 필터(High-pass filter)와 저역 통과 필터(Low-pass filter)의 병렬 조합이다.

고윳값 행렬 $\Lambda = diag[\lambda_1,\lambda_2,\cdots,\lambda_n]$에서 λ가 곧 주파수를 의미한다. 신호 처리에서 사용되는 필터들을 그래프 관점에서 해석하게 되면, 다음 그림 21와 같다. 저역 통과 필터를 적용하게 되면 노드들이 서로 유사해지고, 고역 통과 필터를 적용하면 노드들의 차이가 강조된다.

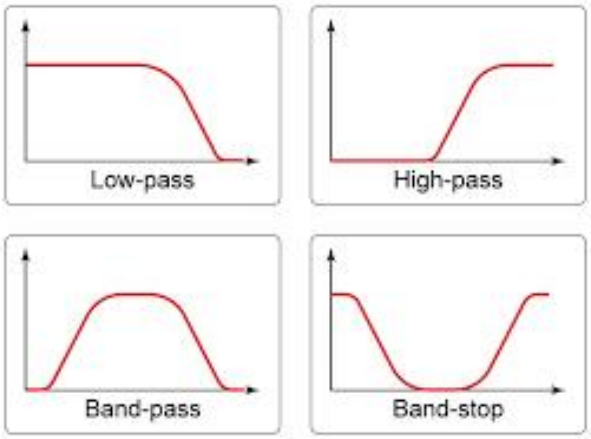


그림 17. 신호 처리 필터 종류

그래프 신경망의 핵심 연산은 주변 노드의 특징을 집계하는 것이다. 이를 위해 각 노드는 자신과 연결된 이웃 노드들로부터 정보를 수집하고, 이를 바탕으로 자신의 특징을 갱신한다. 이 과정은 인접 행렬 A를 사용하여 표현할 수 있다. 노드의 피쳐를 나타내는 행렬 H가 주어졌을 때, 인접 행렬을 이용하여 노드의 특징을 집계할 수 있다. 간단한 그래프 신경망의 한 레이어는 다음과 같이 표현될 수 있다.

$$H(l+1) = \sigma(AH(l)W(l))$$

여기서 $H(l)$는 l번째 레이어의 노드 특징을, $W(l)$는 l번째 레이어의 가중치를 나타내며, σ는 비선형 활성화 함수를 의미한다.

이 수식에서 $AH(l)$은 각 노드가 이웃 노드의 특징을 더하는 과정을 나타낸다. 만약 A가 정규화되지 않았다면, 이 과정은 노드의 차수 (연결된 이웃의 수)에 비례하여 가중치가 부여된다. 이것은 노드의 특징이 차수가 높은 노드에 의해 지배될 수 있음을 의미한다. 따라서, 일반적으로 A는 정규화 과정을 거쳐 각 노드의 특징이 동일하게 고려된다. 정규화된 인접 행렬은 보통 다음과 같이 정의된다.

$$\hat{A} = D^{-1/2}AD^{-1/2}$$

여기서 D는 대각 행렬로, 각 노드의 차수를 대각 원소로 가진다. 정규화를 통한 그래프 신경망의 한 레이어는 다음과 같이 표현될 수 있다.

$$H(l+1) = \sigma(\hat{A}H(l)W(l))$$

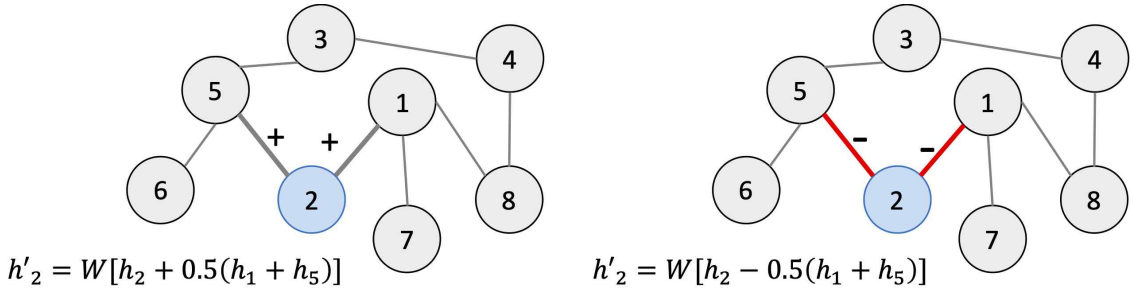


그림 18. 그래프 저역 통과 필터 (왼쪽), 고역 통과 필터 (오른쪽)

이 정규화 과정은 저역통과 필터의 특성을 부여한다. $\hat{A}$ 를 통해 각 노드는 주변 노드들과의 평균화된 특징을 받게 되며, 이는 저주파 신호(즉, 광범위한 구조적 패턴)를 강조하고 고주파 신호(즉, 노드별 미세한 변동성)를 감소시키는 역할을 한다.

결과적으로, GNN에서 레이어를 거칠수록 노드 특징들은 그래프의 광범위한 구조적 정보를 반영하는 방식으로 스무딩(smoothing) 되며, 이것은 저역통과 필터의 특성과 일치한다.

2. 반응-확산 방정식과 그래프 신경망

먼저 확산 방정식과 그래프 신경망과의 관계를 설명한다. 열 확산 방정식은 열이 물리적 매체를 통해 퍼지는 방식을 설명하는 미분 방정식이다. 가장 기본적인 형태는 다음과 같다.

$$\frac{\partial u}{\partial t} = \Delta u$$

여기서 u는 시간 t에 대한 열의 분포를 나타내고, Δ는 라플라시안(Laplacian) 연산자로써, 공간에 대한 두 번째 미분으로 열이 어떻게 분포하는지를 나타낸다. 이 두 개념 사이의 연결 고리는 바로 그래프 라플라시안 (Graph Laplacian)에 있다. 그래프 라플라시안은 그래프에서 노드 간의 연결 구조를 나타내는 데 사용되며, 열 확산 방정식에서의 라플라시안 연산자와 유사한 역할을 한다.

그래프 신경망은 위에서 설명한 것과 같이 $H(l+1) = \sigma(\hat{A}H(l)W(l))$ 으로 나타낼 수 있다. 그래프에서의 라플라시안 행렬 L은 L=D-A로 정의되며, 여기서 D는 노드들의 연결 강도(차수)에 대

한 행렬이고, A는 인접 행렬이다. 그래프 라플라시안은 열 확산 방정식에서와 같이 그래프상에서의 "확산"을 나타낸다.

GCN에서 사용하는 정규화된 그래프 라플라시안은 열이 그래프의 노드를 통해 어떻게 퍼지는지를 모사한다. 각 노드의 새로운 특성은 자신과 이웃 노드의 특성에 기반한 가중 평균으로 계산되며, 이는 물리적 세계에서 열이 물질 속을 퍼져 나가는 방식과 유사하다. 따라서, GCN은 그래프상에서 노드의 특성이 "확산"되는 방식을 학습함으로써 노드의 특성을 업데이트하는 것과 비슷하게 작동한다.

반응-확산 시스템은 화학 반응과 물질의 확산을 모델링하는 방정식으로, 생물학적 패턴 형성이나 화학 반응 같은 현상을 설명하는 데 사용된다. 반응-확산 방정식의 일반적인 형태는 다음과 같다.

$$\frac{\partial u}{\partial t} = \Delta u + R(u)$$

여기서 R은 반응 항이며 반응-확산 방정식이 사용되는 도메인마다 다르게 설계할 수 있다. 이 수식은 확산 방정식에 반응 항이 더해진 수식으로 볼 수 있다.

반응-확산 방정식과 그래프 신경망의 관계는 데이터가 노드 간의 연결을 통해 확산되는 방식과 유사하다. 그래프 신경망에서는 노드의 특성이 그래프 구조를 통해 전파되고, 이때 각 노드에서는 이웃한 노드로부터 정보를 받아 자신의 특성을 갱신한다. 확산 방정식만을 고려할 때, 이는 노드의 특성이 그래프상에서 확산되는 과정을 모델링할 수 있다. 특정 노드의 특성이 시간에 따라 인접한 노드로 확산되면서 전체적인 균형 상태를 이루게 된다. 이는 그래프 신경망에서의 전파(Propagation) 메커니즘과 유사하게, 노드의 정보가 이웃 사이를 통해 전파되는 과정을 수학적으로 표현한다.

그래프에서의 반응-확산 시스템을 예로 들자면, 노드가 화학 물질을 나타내고 엣지가 물질들이 확산될 수 있는 경로를 나타낸다고 할 수 있다. 이때, 각 노드에서의 화학 물질의 농도는 시간에 따라 변화하며, 노드 간의 확산 및 해당 노드에서의 화학 반응에 의해 영향을 받는다. 하지만 반응-확산 방정식을 그래프 신경망으로 재설계하기 위해서는 반응 항을 어떻게 설계할 수 있는지가 가장 중요한 연구다. 이에 대한 설명은 다음 목차에서 설명한다.

3. 기존 그래프 신경망의 오버스무딩 문제와 헤테로필리 그래프의 어려움

그래프 신경망은 여러 분야에서 성공적으로 사용이 되고 있지만 여전히 몇 가지 주요 문제에 직면해 있다. 여기에는 "오버스무딩(Over-smoothing)" 문제와 "헤테로필리(Heterophily)" 그래프에서의 어려움이 포함된다.

오버스무딩은 그래프 신경망이 너무 많은 층을 쌓게 되면 발생하는 문제이다. 각 층에서 노드는 이웃으로부터 정보를 통합하여 자신의 특성을 업데이트한다. 이 과정이 반복되면, 멀리 떨어진 노드들 간의 특성이 혼합되어 노드들의 특성이 균일화되는 현상이 나타난다. 즉, 네트워크의 깊이가 깊어질수록 노드 임베딩들이 구별하기 어려운, 서로 비슷한 값으로 수렴한다. 이로 인해 노드를 구별하는 능력이 떨어지고, 결과적으로 분류나 다른 다운스트림 태스크의 성능이 저하될 수 있다.

오버스무딩은 그래프 신경망이 너무 많은 층을 쌓게 되면 발생하는 문제이다. 각 층에서 노드는 이웃으로부터 정보를 통합하여 자신의 특성을 업데이트한다. 이 과정이 반복되면, 멀리 떨어진 노드들 간의 특성이 혼합되어 노드들의 특성이 균일화되는 현상이 나타난다. 즉, 네트워크의 깊이가 깊어질수록 노드 임베딩들이 구별하기 어려운, 서로 비슷한 값으로 수렴한다. 이로 인해 노드를 구별하는 능력이 떨어지고, 결과적으로 분류나 다른 다운스트림 태스크의 성능이 저하될 수 있다.

그림 19에서와 같이 GCN, GAT, GraphSAGE 모두 레이어 수가 증가할수록 디리클레 에너지와 MAD가 0에 가까워지는 경향을 보인다. 여기서 디리클레 에너지와 MAD는 오버스무딩을 측정할 수 있는 방법들이다.

특히 그래프 신경망은 저역통과 필터로 간주가 되는데, 이는 노드의 특성 벡터가 이웃 노드들과 유사해지도록 만든다. 이 과정이 여러 층에 걸쳐 반복될 때, 멀리 떨어진 노드들 사이에서도 정보가 전파되어 특성이 점점 더 유사해진다. 그리고 기존 그래프 신경망 연구들은 대부분 저역통과 필터인 문제가 있다 (표 12 참고).

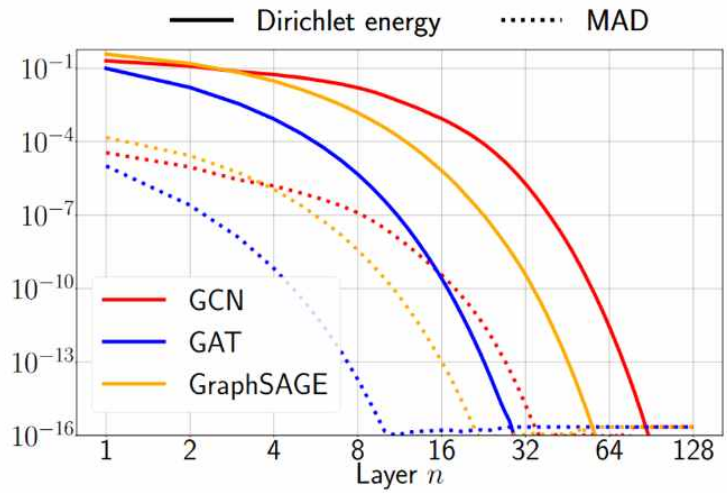


그림 19. 그래프 신경망의 레이어 증가에 따른 디리클레 에너지와 MAD

모델	저역통과필터 사용 여부	고역통과필터 사용 여부
GCN	O	X
SGC	O	X
GCNII	O	X
CGNN	O	X
FA-GCN	O	O
GPR-GNN	O	O
GRAND	O	X

표 12. 저역통과필터를 사용하는 모델들과 고역통과필터를 고려하는 모델들

헤테로필리(Heterophily)는 그래프의 노드들이 서로 다른 종류의 노드와 연결되어 있는 경우를 말한다. 이는 동질성 (노드들이 유사한 노드와 연결되는 경향)과 대비되는 개념이다. 많은 실세계 네트워크에서 노드들은 서로 다른 속성을 가진 이웃과 연결될 수 있다. 예를 들어, 비슷한 관심사를 가진 사람들이 친구가 되거나, 비슷한 기능을 가진 단백질이 상호작용하는 경우가 이에 해당한다. 그러나 대부분의 그래프 신경망 아키텍처는 동질성을 가정하고 있어, 노드들이 유사한 이웃과 연결될 때 가장 잘 작동한다. 결과적으로, 이러한 GNN은 헤테로필리 네트워크에서는 효과적으로 정보를 통합하지 못하고, 노드의 특성을 적절하게 학습하지 못하는 문제가 있다.

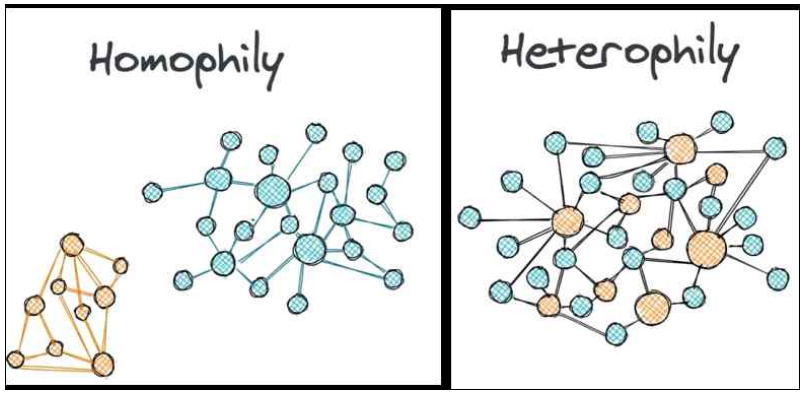


그림 20. 동질성 그래프와 헤테로필리 그래프의 예시

헤테로필리 그래프의 경우, 이웃 노드들이 다른 종류의 노드일 확률이 높기 때문에, 이러한 방식의 정보 집계가 효과적이지 않을 수 있다. 이는 다음과 같은 문제를 야기한다.

가. 정보 손실

다른 종류의 이웃 노드로부터 정보를 단순히 합치는 것은 중요한 정보를 희석시킬 수 있으며, 이는 노드를 정확히 분류하는 데 필요한 구별 가능한 신호를 약화시킨다.

나. 과적합

GNN이 특정한 동질성 패턴에 과적합 되어, 헤테로필리 패턴을 가진 데이터에 대해서는 일반화 성능이 떨어질 수 있다.

다. 노드 역할의 무시

헤테로필리 그래프에서는 종종 노드가 수행하는 역할이 중요하다. 예를 들어, 소셜 네트워크에서는 다른 커뮤니티를 연결하는 '브릿지' 역할을 하는 노드가 있을 수 있다. 동질성을 가정한 GNN은 이러한 역할 기반의 특성을 무시할 수 있다.

해결책으로는 노드 자신의 특성을 유지하면서 이웃으로부터 정보를 통합하는 방법을 개선하거나, GNN 아키텍처 자체를 헤테로필리한 그래프에 적합하도록 조정하는 연구가 진행되고 있다. 예를 들어, 이웃의 정보를 통합할 때 노드의 타입이나 이웃과의 관계 유형을 고려하는 attention 메커니즘을 도입하는 방법, 그리고 레이어 별로 서로 다른 가중치를 주어 멀리 떨어진 노드의 정보가 너무 많이 혼합되는 것을 방지하는 기술 등이 개발되고 있다.

4. 반응-확산 방정식을 이용한 그래프 신경망 고도화

그래프 신경망은 딥러닝에 대한 가장 인기 있는 연구 주제 중 하나이다. 그래프 신경망 방법은 일반적으로 그래프 신호 처리 이론을 기반으로 설계되었다. 특히, 확산 방정식은 그래프 신경망의 핵심 처리 계층을 설계하는 데 널리 사용되어 왔기 때문에 필연적으로 악명 높은 오버스무딩 문제에 취약하다. 최근 몇 편의 논문이 확산 방정식과 연계하여 연구가 이루어지고 있다. 그러나 이 논문들은 모두 제한된 형태의 반응 방정식을 고려한다. 이를 위해 우리는 우리가 설계한 하나의 특수 반응식 외에도 널리 사용되는 모든 유형의 반응식을 고려하는 반응확산 방정식 기반 그래프 신경망 방법인 GREAD를 제안한다.

노드 특징 행렬 X, 대칭 정규화 라플라시안 행렬 L, 대칭 정규화 인접 A가 포함된 그래프 G가 주어지면 GREAD의 전체 아키텍처는 다음과 같이 작성될 수 있다.

$$\text{인코딩 계층: } H(0) = e(X)$$

$$\text{Reac.-diff. 계층: } H(T) = H(0) + \int_0^T f(H(t))dt$$

$$\text{아웃풋 계층: } y = \sigma(H(T))$$

e는 노드 특징 행렬 X를 초기 숨겨진 상태 H(0)에 삽입하는 인코더이다. 그런 다음 f의 반응-확산 방정식을 통해 초기 숨겨진 상태를 H(T)로 발전시킨다. 함수 o는 노드 분류와 같은 다운스트림 작업을 위한 출력 레이어이다.

여기서 reaction-diffusion 계층의 함수는 이처럼 정의한다.

$$f(H(t)) := dH(t)dt = -\alpha LH(t) + \beta r(H(t))$$

$r(H)$는 반응 항이고, α와 β는 각 항을 (비)강조하기 위해 훈련 가능한 매개변수이다.

특히, β는 스칼라 또는 벡터 매개변수일 수 있다. 여기서 스칼라 설정은 모든 노드에 동일한 반응 프로세스를 적용하고 벡터 설정에서는 노드에 서로 다른 계수를 가진 서로 다른 반응 프로세스를 적용한다는 의미다.

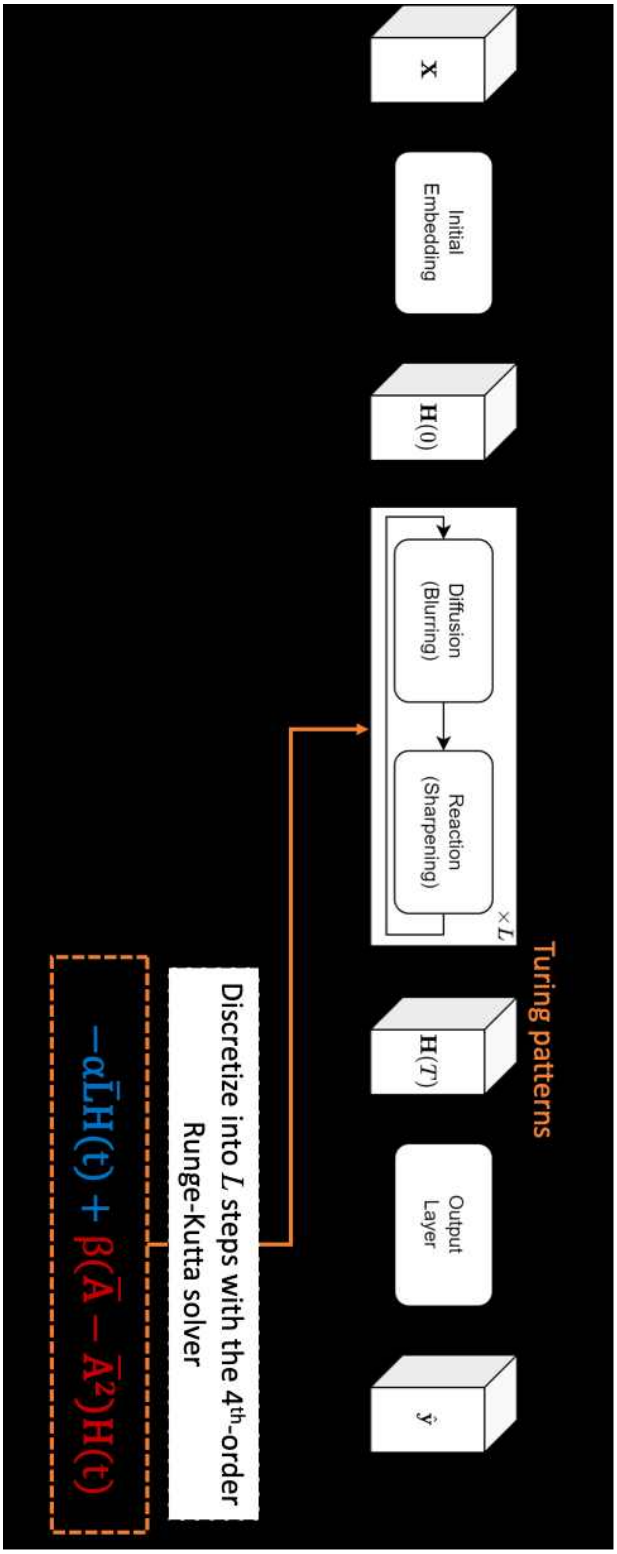


그림 21. GREAD 모델의 오버뷰

인코더 e에는 ReLU(Rectified Linear Unit) 활성화 기능을 갖춘 완전히 연결된 두 개의 레이어가 있다. 출력 레이어 o는 일반적으로 완전 연결 레이어이고, 그 뒤에는 실험에서 분류를 위한 소프트맥스 활성화가 이어진다. 특히, r에 대한 기존의 반응항을 거의 모두 고려하는데, 이는 이를 철저하게 고려하지 않는 기존 연구와는 다르다. 이러한 관점에서 우리 연구는 우리 지식에 대한 반응 확산 GNN에 대한 가장 포괄적인 연구다. 다음 하위 섹션에서 우리는 또한 반응 항의 일부 선택이 다른 유명한 모델에 해당한다는 것을 보여준다. 즉, 일부 다른 유명한 모델은 GREAD의 특별한 경우이다.

그래프 G가 주어지면 원래 대칭 정규화 인접 행렬 A ∈ [0, 1]|V|×|V| f(H(t))의 경우. 그러나 ~A ∈ [0, 1]|V|×|V|로 표시되는 소프트 인접 행렬을 생성하는 방법도 제공한다. 우리는 ~L을 사용하여 ~A의 라플라시안 대응을 나타낸다. 우리의 소프트 인접 행렬은 확산성을 학습하는 데 중요한 역할을 한다. 우리의 반응 확산 레이어는 확산성을 학습하기 위해 소프트 인접 행렬을 사용한다. 이러한 소프트 인접 행렬을 생성하기 위해 우리는 스케일 내적(scaled dot product) 방법을 사용한다.

f의 정의에서 '−LH(t)'는 G에 대한 열 확산을 설명하는 열 방정식에 해당하는 확산 항이며 다양한 GNN에서 널리 사용된다. 확산 항은 오버스무딩 문제를 일으키는 것으로 알려져 있다. 이는 확산 처리만 너무 많이 적용하면 노드의 마지막 숨겨진 상태가 너무 유사해짐을 의미한다. 최근에는 이를 방지하기 위해 경험적 방법을 사용하지만 이들의 방법은 여전히 한계가 있다.

GREAD의 경우, 반응항 r을 추가하고 오버스무딩 문제를 방지한다. 우리는 반응항 r에 대해 다음 옵션을 고려한다.

$$r(H(t)) := \begin{cases} H(t) \odot (1 - H(t)) & \text{if } Fisher\,(F) \\ H(t) \odot (1 - H(t)^{o2}) & \text{if } Allen - Cahn\,(AC) \\ H(t) \odot (H(t) - H(t)^{o2}) & \text{if } Zeldovich\,(Z) \\ (\overline{A} - \overline{A^2}) H(t) & \text{if } Blurring - Sharpening\,(BS) \\ H(0) & \text{if } Source\,Term\,(ST) \\ LH(t) & \text{if } FilterBank\,(FB) \\ LH(t) + H(t) & \text{if } FilterBank^{*}\,(FB^{*}) \end{cases}$$

여기서 '⊙'는 하다마르 곱(Hadamard product)을 의미하고, 'o2'는 하다마르 거듭제곱(Hadamard power)을 의미한다. 처음 세 가지 반응 항, 즉 F, AC 및 Z는 다양한 영역에서 널리 사용된다. 예를 들어, F는 생물학적 개체군의 확산을 설명하는 데 사용되며, AC는 질서 무질서 전이를 포함하는 다성분 합금 시스템의 상 분리 과정을 설명하는 데 사용된다. Z는 연소 이론에서 발생하는 현

상을 설명하는 방정식이다. BS는 GNN을 위해 특별히 설계되었다. ST는 초기 Hidden State를 반응항으로 추가한 경우이다. ST는 이론적으로 반응 과정은 아니지만 목표가 동일하기 때문에 (예: 악명 높은 과도 평탄화 문제 완화) 모델의 일부로 간주한다. FB는 반응 과정에 해당하는 고역 통과 필터를 의미한다. 고역 통과 필터를 추가함으로써 반응 확산층은 저역 통과 필터와 고역 통과 필터를 보유하는 필터 뱅크처럼 작동한다. FB*는 동일 채널 H(t)도 고려하는 반응항이다.

ST를 사용하는 경우 GREAD는 두 방법 모두 초기 숨겨진 상태를 주입한다는 관점에서 GCNII와 유사하다. GREAD, FA-GCN 및 GPR-GNN은 저역 통과 필터와 고역 통과 필터를 활용하는 방법이 다르다. FA-GCN은 GAT와 마찬가지로 에지 수준 집계 가중치를 학습하지만 음수 가중치를 허용한다. GPR-GNN은 특성 전파에 대해 양수 및 음수가 모두 가능한 학습 가능한 가중치를 사용한다. 이를 통해 FA-GCN 및 GPR-GNN은 이종친화성 그래프에 적응하고 그래프 신호의 고주파 및 저주파 부분을 모두 처리할 수 있다. 그러나 GREAD-BS는 개발된 반응 확산 시스템에 따라 저역 통과 필터링된 신호를 선명하게 한다. GREAD-BS는 또한 각 용어를 적응적으로 조정한다.

5. 반응-확산 방정식을 이용한 그래프 신경망의 성능 측정 결과

9개 데이터세트와 28개 기준선을 사용하여 실험한다. GREAD-BS는 Squirrel 및 Citeseer 데이터세트를 제외한 다른 방법에 비해 우수한 분류 성능을 보여준다. 기존의 다른 방법이 부족했던 안정성을 보여준다.

Dataset	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
MLP	80.81±4.75	85.29±3.31	81.89±6.40	36.53±0.70	28.77±1.56	46.21±2.99	87.16±0.37	74.02±1.90	75.69±2.00
GCN	55.14±5.16	51.76±3.06	60.54±5.30	27.32±1.10	53.43±2.01	64.82±2.24	86.98±1.27	76.50±1.36	88.42±0.50
ChebNet	78.37±6.04	79.02±3.18	75.68±6.94	34.13±1.09	36.43±1.17	58.64±1.64	85.45±1.58	75.07±1.25	89.00±0.46
GAT	52.16±6.63	49.41±4.09	61.89±5.05	27.44±0.89	40.72±1.55	60.26±2.50	86.33±0.48	76.55±1.23	87.30±1.10
GraphSAGE	82.43±6.14	81.18±5.56	75.95±5.01	34.23±0.99	41.61±0.74	58.73±1.68	86.90±1.04	76.04±1.30	88.45±0.50
SGC	58.10±4.20	55.29±4.28	60.00±3.59	27.20±1.52	33.00±1.97	42.45±3.82	86.12±1.44	76.01±1.31	86.90±1.32
MixHop	77.84±7.73	75.88±4.90	73.51±6.34	32.22±2.34	43.80±1.48	60.50±2.53	87.61±0.85	76.26±1.33	85.31±0.61
Geom-GCN	66.76±2.72	64.51±3.66	60.54±3.67	31.59±1.15	38.15±0.92	60.00±2.81	85.35±1.57	78.02±1.15	89.95±0.47
FA-GCN	82.43±6.89	82.94±7.95	79.19±9.79	34.87±1.25	42.59±0.79	55.22±3.19	87.21±1.43	76.87±1.56	87.45±0.61
GPR-GNN	78.38±4.36	82.94±4.21	80.27±8.11	34.63±1.22	31.61±1.24	46.58±1.84	87.95±1.18	77.13±1.67	87.54±0.38
H2GCN	84.86±7.23	87.65±4.98	82.70±5.28	35.70±1.00	36.48±1.86	60.11±2.15	87.87±1.20	77.11±1.57	89.49±0.38
WRGAT	83.62±5.50	86.98±3.78	81.62±3.90	36.53±0.77	48.85±0.78	65.24±0.87	88.20±2.26	76.81±1.89	89.29±0.38
GGCN	84.86±4.55	86.86±3.29	85.68±6.63	37.54±1.56	55.17±1.58	71.14±1.84	87.95±1.05	77.14±1.45	89.15±0.37
LINKX	74.60±8.37	75.49±5.72	77.84±5.81	36.10±1.55	61.81±1.80	68.42±1.38	84.64±1.13	73.19±0.99	87.86±0.77
GloGNN	84.32±4.15	87.06±3.53	83.51±4.26	37.35±1.30	57.54±1.39	69.78±2.42	88.31±1.13	77.41±1.65	89.62±0.35
ACM-GCN	87.84±4.40	88.43±3.22	85.14±6.07	36.28±1.09	54.40±1.88	66.93±1.85	87.91±0.95	77.32±1.70	90.00±0.52
PairNorm	60.27±4.34	48.43±6.14	58.92±3.15	27.40±1.24	50.44±2.04	62.74±2.82	85.79±1.01	73.59±1.47	87.53±0.44
JKNet	62.70±8.34	53.14±5.22	59.72±4.60	29.25±1.37	39.78±1.72	52.63±3.90	86.48±1.04	75.99±1.28	87.23±0.55
GCNII	77.57±3.83	80.39±3.40	77.86±3.79	37.44±1.30	38.47±1.58	63.86±3.04	88.37±1.25	77.33±1.48	90.15±0.43
GCON-GCN	85.40±4.20	87.80±3.30	84.30±4.80	34.65±0.61	33.30±1.57	48.08±2.16	87.40±1.82	76.46±1.70	87.71±0.35
GCON-GAT	82.20±4.70	85.70±3.60	83.20±7.00	35.85±0.84	34.45±1.08	48.31±1.53	86.96±1.73	76.20±2.12	87.73±0.41
CGNN	71.35±4.05	74.31±7.26	66.22±7.69	35.95±0.86	29.24±1.09	46.89±1.66	87.10±1.35	76.91±1.81	87.70±0.49
GDE	74.05±6.96	79.80±5.62	82.43±7.07	35.36±1.31	35.94±1.91	47.76±2.08	87.22±1.41	76.21±2.11	87.80±0.38
GRAND	75.68±7.25	79.41±3.64	82.16±7.09	35.62±1.01	40.05±1.50	54.67±2.54	87.36±0.96	76.46±1.77	89.02±0.51
BLEND	83.24±4.65	84.12±3.56	85.95±6.82	35.63±1.01	43.06±1.39	60.11±2.09	88.09±1.22	76.63±1.60	89.24±0.42
ACMP	86.20±0.30	86.10±0.40	85.40±0.70	34.44±4.44	52.65±2.23	52.63±2.28	86.38±3.79	76.52±1.84	87.54±0.57
Sheaf	85.05±5.51	89.41±4.74	84.86±4.71	37.81±1.15	56.34±1.32	68.04±1.58	86.90±1.13	76.70±1.57	89.49±0.40
GRAFF	88.38±4.53	87.45±2.94	83.24±6.49	36.09±0.81	54.52±1.37	71.08±1.75	87.61±0.97	76.92±1.70	88.95±0.52
GREAD-BS	88.92±3.72	89.41±3.30	86.49±7.15	37.90±1.17	59.22±1.44	71.38±1.53	88.57±0.66	77.60±1.81	90.23±0.55
GREAD-F	89.73±4.49	86.47±4.84	86.49±5.13	36.72±0.66	46.16±1.44	65.20±1.40	88.39±0.91	77.40±1.54	90.09±0.31
GREAD-AC	85.95±2.65	86.08±3.56	87.03±4.95	37.21±1.10	45.10±2.11	65.09±1.08	88.29±0.67	77.38±1.53	90.10±0.36
GREAD-Z	87.30±5.68	86.29±4.32	85.68±5.41	37.01±1.11	46.25±1.72	62.70±2.30	88.31±1.10	77.39±1.90	90.11±0.27
GREAD-ST	81.08±5.67	86.67±3.01	86.22±5.98	37.66±0.90	45.83±1.40	63.03±1.32	88.47±1.19	77.25±1.47	90.13±0.36
GREAD-FB	86.76±5.05	87.65±3.17	86.22±5.85	37.40±0.55	50.83±2.27	66.05±1.21	88.01±1.34	77.28±1.73	90.07±0.45
GREAD-FB*	87.03±3.97	88.04±1.63	85.95±5.64	37.70±0.51	50.57±1.52	65.83±1.10	88.01±0.80	77.42±1.93	90.08±0.46

표 13. 벤치마크 데이터셋에 대한 실험 결과

제 4 장. 개인화된 그래프 필터 기반 추천 알고리즘 개발

제 1 절. 개인화를 위한 그래프 필터 기반 추천 알고리즘 동향 및 조사

1. 추천 시스템 소개 및 배경지식

추천 시스템은 정보 필터링 (Information Filtering) 기술의 일종으로, 특정 사용자가 관심을 가질 만한 정보 (영화, 음악, 책, 뉴스, 이미지, 웹 페이지 등)를 추천하는 것이다. 일상에서 우리가 접하는 유튜브, 넷플릭스, 아마존 등의 다양한 서비스들은 추천 시스템을 사용하여 수익을 높이고 사용자들의 만족도를 향상시킨다. 넷플릭스 영상 시청의 80%가 추천 시스템에서 비롯되었으며 아마존의 추천 시스템은 전체 매출의 35%를 향상시켰다.

추천 시스템은 (Recommendation System)은 일반적으로 두 가지 형태로 사용된다. 사용자의 기록을 기반으로 한 개인화된 페이지를 생성하는 추천이다. 구글 플레이 스토어나 유튜브의 메인 화면을 예시로 들 수 있다. 두 번째는 유사 아이템 추천으로 특정 아이템과 유사한 아이템을 추천해주는 시스템을 의미한다. 추천 시스템에서 사용되는 용어의 설명은 다음 표 14와 같다.

아이템 (item)	아이템은 시스템이 추천하는 개체를 말한다. 예를 들어 넷플릭스의 경우에는 영화를 지칭한다.
인터랙션 (interaction)	인터랙션은 시스템이 추천을 위해 사용하는 정보이며, 사용자가 구매, 시청, 방문, 클릭, 리뷰하였던 정보를 사용한다. 즉, 사용자와 아이템간의 상호작용이라 할 수 있다.
임베딩 (embedding)	연속 값 특성으로 표현된 범주형 특성을 의미하며 일반적으로 임베딩은 고차원 벡터를 저차원 공간으로 변환한 결과이다.

표 14. 추천 시스템의 주요 용어 설명

추천 시스템은 일반적으로 협업 필터링 (Collaborative Filtering) 방식과 컨텐츠 기반 필터링 (Content-based Filtering)을 통해 추천 목록을 만든다. 협업 필터링은 사용자 행동, 활동 또는 선호도에 대한 많은 정보를 분석하고 다른 사용자와의 비슷함에 기초를 두고 사용자들이 무엇을 좋아

할 지를 예측하는 것에 기초를 둔다. 협업 필터링의 주요 장점은 정확하게 아이템 그 자체를 이해하지 않고도 영화와 같은 복잡한 아이템을 추천할 수 있다. 본 과제에서는 협업 필터링을 사용한다. 협업 필터링은 과거에 동의한 사용자들이 미래에도 동의하고 사용자들이 과거에 선호하던 것들을 선호할 것이라는 가정에 기초를 두고 있다. 사용자의 행동으로부터 데이터를 구축할 때 아래와 같이 외재적 피드백 (Explicit Feedback) 과 내재적 피드백 (Implicit Feedback)으로 구분할 수 있다. 외재적 피드백은 사용자에게 아이템을 평가하게 하도록 하는 것이다. 대표적인 예는 영화의 평점을 남기는 것이 있다. 내재적 피드백은 사용자가 구매하거나 클릭을 한 기록만을 사용한다. 내재적 피드백의 장점은 사용자가 리뷰를 남기지 않는다는 한계를 극복할 수 있다.

협업 필터링은 내재적 피드백을 포함한 인터랙션 행렬을 이분 그래프 (bipartite graph)로 나타낼 수 있다. 아래와 같이 사용자가 어떤 아이템을 구매했던 기록이 있다면 이분 그래프에서 두 노드 간의 엣지(edge)가 존재하는 구조로 만들 수 있다. 그림 22의 사용자 u_1은 $i_1, i_2 i_3$을 구매했었기 때문에 세 개의 엣지를 가진다. 그리고 추천 시스템의 목표는 사용자가 구매하지 않은 아이템을 추천해주어야 하는지를 결정하는 것이다.

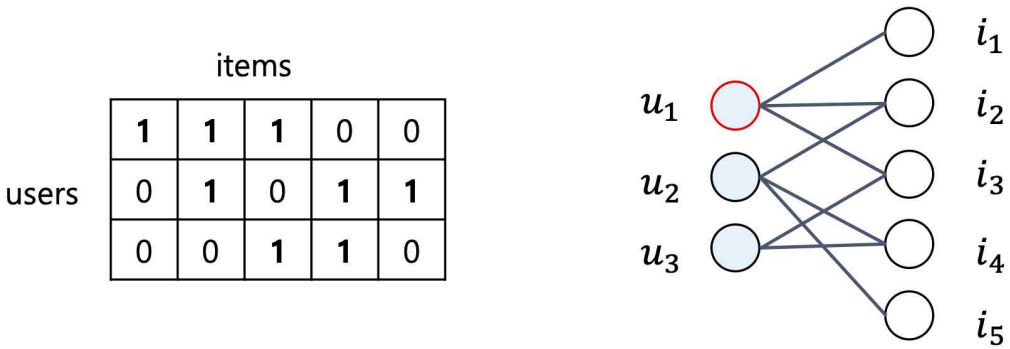


그림 22. 인터랙션 행렬과 이분 그래프

다시 정리하자면 협업 필터링은 추천 시스템에서 가장 많이 사용되는 방법론으로 사용자-아이템 상호작용 데이터를 활용하여 사용자가 소비할 상품을 예측하는 것이다. 협업 필터링의 가장 잘 알려진 예는 행렬 분해 (Matrix Factorization)이다. 행렬 분해 알고리즘은 사용자-아이템 상호작용 행렬을 2개의 낮은 차원의 직사각형 행렬의 곱으로 분해하여 작동한다[21]. 이 방법은 아이템의 인기도와 사용자의 활동성을 기반으로 잠재 요인에 다른 정규화 가중치를 할당하여 예측 결과를 개

선할 수 있다.

그림 23과 같이 인터랙션 행렬은 사용자 행렬과 아이템 행렬로 분해를 한 뒤 두 행렬을 곱하여 예측된 사용자 rating을 계산할 수 있다. 다음 절에서 딥러닝과 그래프를 이용한 행렬 분해를 소개한다.

그림 23. 행렬 분해 (Matrix Factorization) 의 예시

2. 그래프 추천 시스템의 필요성

추천 시스템은 행렬 분해 기법에 대한 연구가 활발하였다. 최근 몇 년 동안 많은 딥러닝 기술이 제안되었으며 그 중 일부는 비선형 뉴럴 네트워크를 사용해 전통적인 행렬 분해 알고리즘을 일반화한다[22]. 일반적으로 학습 가능한 협력 필터링 모델은 두 가지 주요 구성 요소가 있다. (1) 사용자 및 아이템을 표현을 변환하는 임베딩과 (2) 임베딩을 기반으로 과거 정보 상호 작용을 재구성하는 상호 작용 모델링이다. 뉴럴 협업 필터링 (Neural Collaborative Filtering) 모델은 내적으로 이루어진 행렬 분해 상호작용을 비선형 뉴럴 네트워크로 대체한다.

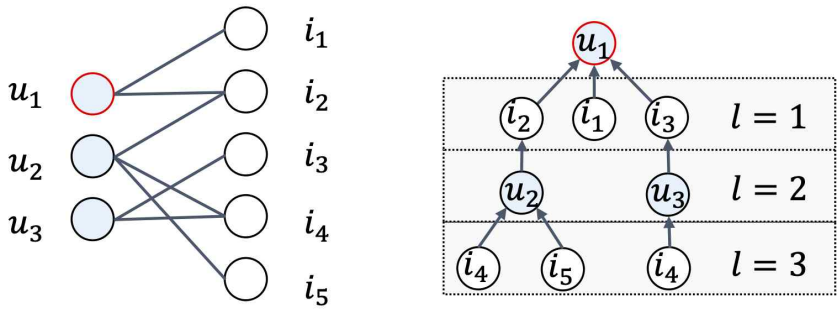


그림 24. 이분 그래프와 고차 연결성 (다중 홉)을 고려하는 예시

하지만 고차 연결성 정보를 고려하지 않는 모델들의 한계를 극복하고자 그래프 기반 추천 알고리즘의 연구들이 등장하게 된다. 그래프 기반 추천 알고리즘의 필요성은 다음과 같다. 임베딩 피쳐가 사용자 또는 아이템 간의 유사성을 나타내기 위해 사용자-아이템 상호작용에 잠재되어 있는 중요한 협력 신호 (Collaborative Filtering)의 명시적 인코딩이 부족하여 협력 필터링에 만족스러운 임베딩을 생성하는데 충분하지 않다. 좀 더 구체적으로 말하면, 사용자-아이템 상호작용을 고려하지 않고 임베딩 함수를 만든다.

사용자-아이템 상호작용을 임베딩 피쳐에 통합하는 데 기존의 상호 작용 모델링은 협업 필터링을 포착하는데 충분치 않다. 특히 상호작용의 규모는 실제 애플리케이션에서 수백만 또는 그 이상에 쉽게 도달할 수 있으므로 원하는 협업 신호를 추출하기 어렵다. 상호작용 그래프 구조에서 협업 신호를 인코딩하는 자연스러운 방법은 그림 24와 같이 사용자-아이템 상호 작용의 고차 연결

성을 활용하는 것이다. 추천에 대한 관심 사용자 u_1 이며 사용자−아이템 상호 작용 그래프의 그림에서 이중 원으로 레이블 된 노드이다. 오른쪽 그림은 u_1에서 확장된 트리 구조를 보여준다. 고차 연결은 경로 길이가 1보다 큰 노드에서 u_1에 도달하는 경로를 나타낸다. 이러한 고차 연결에는 협력 신호를 전달하는 풍부한 의미가 포함되어 있다. 예를 들어 경로 $u_1 \leftarrow i_2 \leftarrow u_2$ 는 두 사용자가 모두 i_2 와 상호 작용했기 때문에 u_1와 u_2 사이의 동작 유사성을 나타낸다. 더 긴 경로 $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$는 u_1이 i_4를 선택할 가능성이 있음을 나타낸다. 더욱이 $l=3$의 전체적인 관점에서, 아이템 i_4 는 $< i_4, u_1 >$을 연결하는 두 개의 경로가 있고 하나의 경로만이 $< i_5, u_1 >$을 연결하기 때문에 아이템 i_5보다 더 관심이 많다.

3. 그래프 추천 시스템의 관련 연구

그래프 추천 시스템의 관련 연구는 다음 표 15와 같다.

논문	연도	학회	모델	데이터셋
Neural Graph Collaborative Filtering	2019	SIGIR	GCN	Gowalla Yelp2018, Amazon-book
LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation	2020	SIGIR	Simplified GCN	Gowalla Yelp2018, Amazon-book
LT-OCF: Learnable-Time ODE-based Collaborative Filtering	2021	CIKM	Simplified GCN + ODE	Gowalla Yelp2018, Amazon-book
How Powerful is Graph Convolution for Recommendation?	2021	CIKM	Graph Filter	Gowalla Yelp2018, Amazon-book

표 15. 그래프 기반 추천 알고리즘 동향

Neural Graph Collaborative Filtering (NGCF)[23] 는 사용자-아이템 이분 그래프에서 임베딩 전파(propagation)를 통해 더 높은 연결 정보를 사용하는 협력 필터링이다. NGCF는 임베딩 함수에서 고차 연결 정보를 모델링 하는 것을 제안한다. 구현하기 복잡한 트리로 상호작용 그래프를 확장하는 대신 그래프에 재귀적으로 임베딩을 전파하는 신경망 방법을 설계한다. 구체적으로 상호 작용

한 아이템 또는 사용자가 다중 임베딩 전파 계층을 쌓음으로써 임베딩을 적용하여 고차 연결에서 협력 신호를 캡처할 수 있다.

그림 25는 NGCF의 아키텍처로 사용자와 아이템 임베딩$e_{u_1}^{(0)}$, $e_{i_4}^{(0)}$을 여러 임베딩 전파 레이어로 개선되며 최종 예측을 위해 각각의 레이어에서 임베딩이 출력된다. 레이어의 임베딩들을 가중치 합을 사용하여 최종 사용자와 아이템 임베딩$e_{u_1}^{*}$, $e_{i_4}^{*}$을 만들고 두 임베딩의 곱을 통해 예측 rating 값 $\tilde{y}_{NGCF}(u_1,i_4)$을 출력한다.

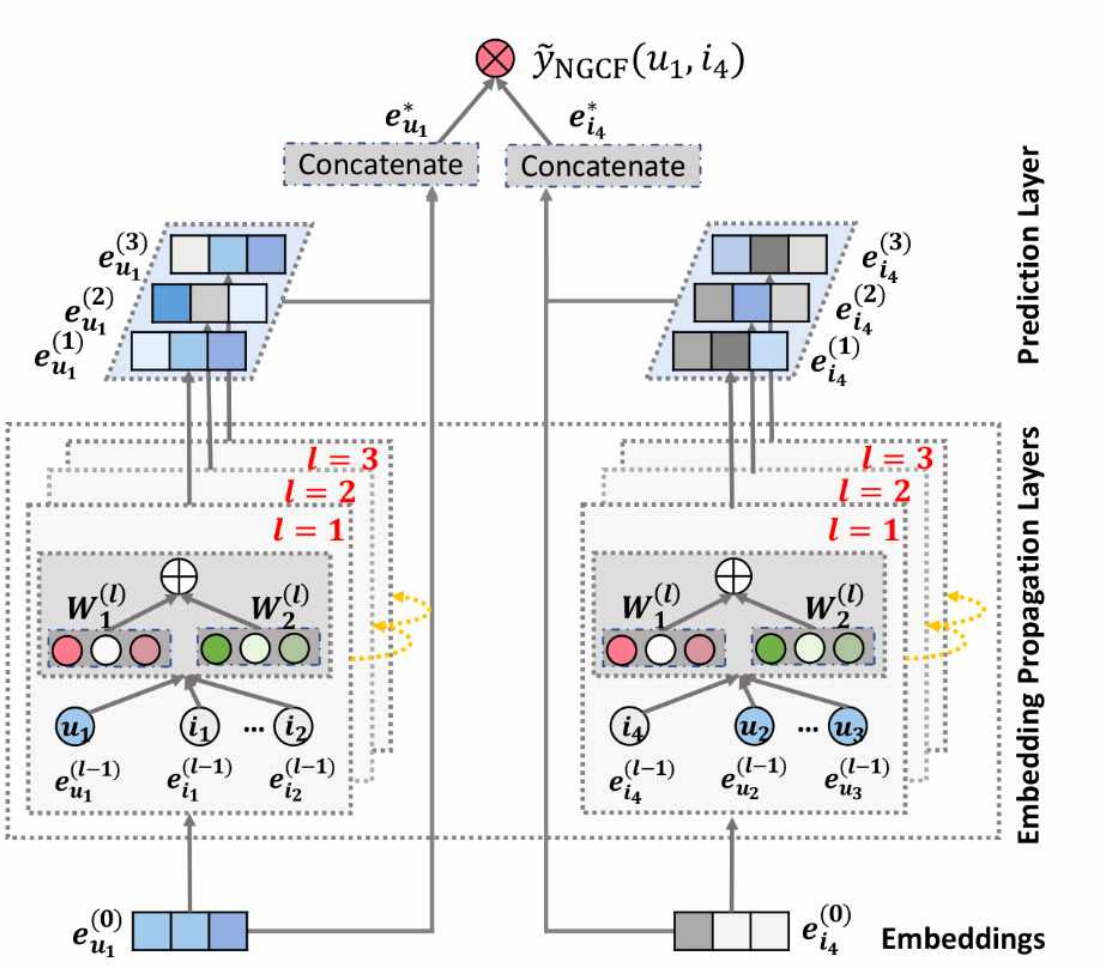


그림 25. NGCF의 아키텍처

아이템을 소비하는 사용자는 아이템의 피쳐로 취급될 수 있고 두 아이템의 협업 유사성을 측정하는 데 사용될 수 있다. 이를 기반으로 연결된 사용자와 항목 간의 임베딩 전파를 수행한다. 임베딩 전파를 수행하는 데 사용되는 두 가지 주요 기능인 메시지 구성 (message construction)과 메시지 집계 (message aggregation)를 공식화한다.

$$m_{u\leftarrow i} = \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_i + W_2(e_i \odot e_u))$$

메시지 구성 (message construction). $m_{u\leftarrow i}$은 메시지 임베딩이고 사용자와 아이템 임베딩 $\boldsymbol{e}_u$, $\boldsymbol{e}_i$ 을 입력으로 받는다. $\boldsymbol{W_1}$와 $\boldsymbol{W_2}$ 는 전파에 유용한 정보를 얻기 위한 학습 가능한 가중치 파라미터이다. $\boldsymbol{e}_i$만을 고려하는 기존의 GCN과 달리 사용자와 아이템 임베딩 간의 상호 작용을 $\boldsymbol{e_i \odot e_u}$을 통해 전달되는 메시지에 추가로 인코딩한다. ⊙는 성분별 곱(element-wise) 곱을 의미한다. N_u와 N_i는 사용자와 아이템의 첫 번째 홉 이웃을 나타낸다. 이를 사용하는 정규화는 과거 항목이 사용자 선호도에 얼마나 기여했는지를 반영하고 메시지 전달의 관점에서 전달되는 메시지가 경로 길이에 따라 감소해야 하는 점을 반영하는 것으로 해석될 수 있다.

메시지 집계 (message aggregation). u의 표현을 구체화하기 위해 u의 이웃에서 전파된 메시지를 집계한다. 집계 함수 (aggregation function)은 아래와 같이 정의한다.

$$e_u^{(l)} = LeakyReLU(m_{u\leftarrow u}^{(l)} + \Sigma_{i\in N_u} m_{u\leftarrow i}^{(l)})$$

$\boldsymbol{e}_u^{(l)}$은 임베딩 전파를 마친 사용자 임베딩을 의미한다. 활성화 함수 LeakyReLU는 메시지를 양의 신호와 작은 음의 신호로 인코딩될 수 있도록 한다. 그리고 더 많은 임베딩 전파 레이어를 스택하여 고차 연결 정보를 탐색할 수 있다. 전파되는 계층을 쌓음으로써 사용자 및 아이템은 1-홉 이웃에서 전파된 메시지를 받을 수 있다.

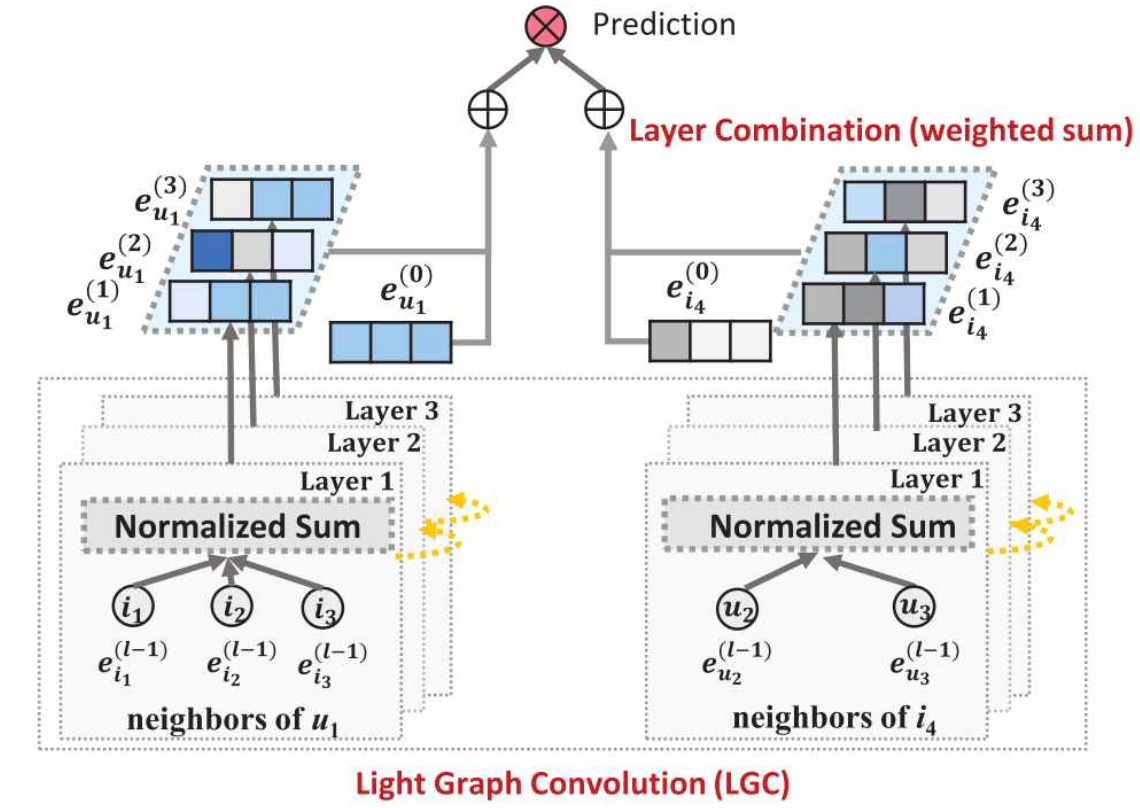


그림 26. LightGCN의 아키텍쳐

LightGCN[24]은 NGCF에서 사용하는 피쳐 변환 행렬 (학습 가능한 가중치 행렬)인 W_1와 W_2을 제거하고 비선형 활성화 함수 (LeakyReLU)를 사용하지 않는 선형 GCN을 제안한다. 학습 프레임워크는 NGCF와 마찬가지로 사용자와 아이템 임베딩 e_u, e_i 을 입력하여 협력 필터링에 적합한 임베딩으로 학습한다. 학습 시에 고차 연결을 고려하는 임베딩 전파를 사용하며 마찬가지로 각 레이어의 임베딩들을 출력하여 최종 임베딩을 구성한 후 내적을 통해 예측한다.

LightGCN은 실험적으로 비선형 GCN을 사용하는 NGCF보다 성능과 효율성 측면에서 향상되는 것을 표 19 에서 확인할 수 있다.

Dataset		Gowalla		Yelp2018		Gowalla	
Layer #	Method	recall	ndeg	recall	ndeg	recall	ndeg
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755	0.1492	0.0631	0.0515	0.0384	0.0298
		(+12.79%)	(+13.46%)	(+16.20%)	(+16.51%)	(+22.68%)	(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777	0.1524	0.0622	0.0504	0.0411	0.0315
		(+14.84%)	(+16.60%)	(+9.89%)	(+8.38%)	(+24.54%)	(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823	0.1555	0.0639	0.0525	0.0410	0.0318
		(+16.19%)	(+17.18%)	(+10.38%)	(+10.06%)	(+21.66%)	(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830	0.1550	0.0649	0.0530	0.0406	0.0313
		(+16.56%)	(+16.80%)	(+14.58%)	(+15.02%)	(+17.92%)	(+18.92%)

표 16. NGCF와 LightGCN의 성능 비교

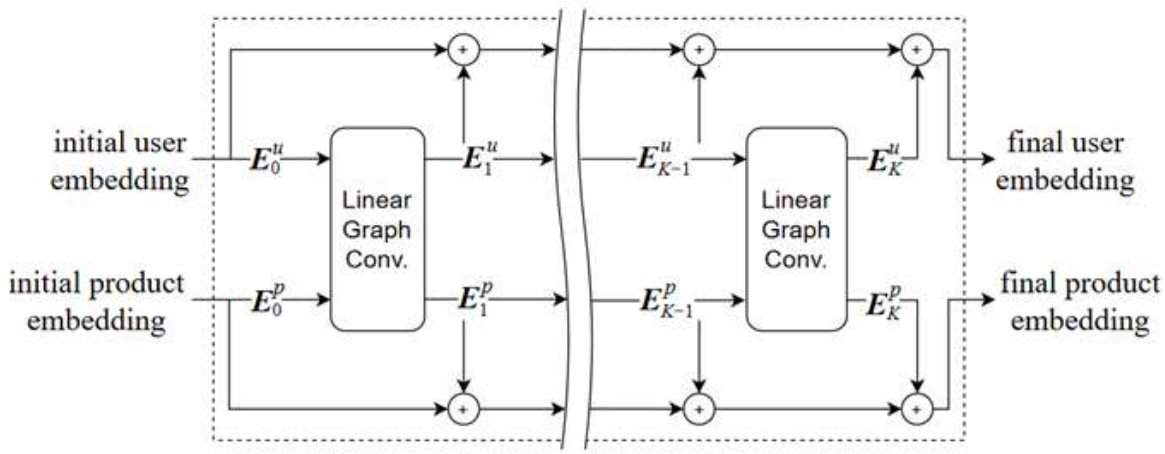


그림 27. LightGCN의 아키텍처

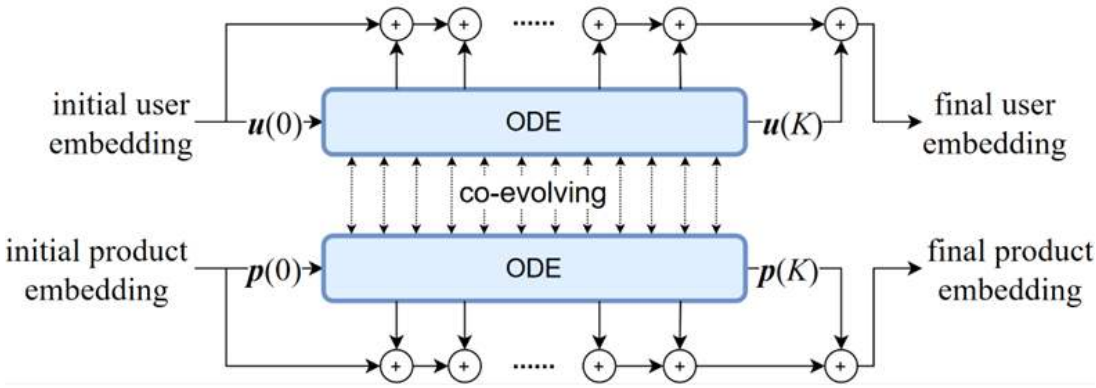


그림 28. LT-OCF의 아키텍처

LT-OCF[25] 는 선형 GCN 개념을 미분 방정식으로 해석할 수 있기 때문에 신경 상미분 방정식(NODE; Neural Ordinary Differential Equations)[26]을 기반으로 확장하고 학습 가능한 시간 ODE 기반 협업 필터링(LT-OCF) 방법을 제시한다. LT-OCF는 NODE의 패러다임 위에 선형 GCN을 재설계한 후 1) 수동으로 설계된 아키텍처에 의존하기보다 최적의 아키텍처를 학습하고 2) 협력 필터링에 적합한 것으로 간주되는 부드러운 ODE 솔루션을 학습하고 3) 다양한 ODE 솔버로 테스트한다. 연속적인 개념의 선형 GCN을 제안하며 성능상에서 3개의 추천 벤치마크 데이터셋에서 성능 향상을 보인다.

그림 30의 LightGCN은 수동으로 설계되어 임베딩 전파의 횟수에 따라 성능이 하락하거나 상승한다. 그림 31의 LT-OCF는 선형 GCN을 연속적인 모델로 재설계 함으로써 전파의 범위를 학습을 통해 결정할 수 있다.

GF-CF[27]의 이상 저역 통과 필터에는 차단(Cut-off) 주파수(Frequency)가 존재한다. 이상 필터의 경우 추천 시스템에서의 행렬 분해(Low-rank matrix factorization)와 동일하다. 행렬 분해의 경우 가장 클래식한 협업 필터링 알고리즘 중 하나로, 선형대수학의 특잇값 분해(Singular Value Decomposition, SVD)를 응용하여 나온 방법론이다. 간단히 말해 하나의 행렬을 여러 행렬로 분해한다는 관점을 빌린 것이다. 특잇값 분해는 $m \times n$ 차원의 행렬 W에 대하여 다음 수식과 같이 행렬을 분해할 수 있다는 행렬 분해(Decomposition) 방법 중 하나이다.

$$W = U \Sigma V^T$$

그중에서 가운데 행렬 Σ는 특잇값을 갖고 있는데, 특잇값은 클수록 더 많은 정보를 갖고 있다고 해석할 수 있다. 하지만, 모든 특잇값 정보를 다 사용하는 것보다는 핵심적인 정보만 추려서 사용하는 것이 효율적이다. 그래서 그림 29와 같이 특잇값이 높은 상위 K개만 추려서 사용하는데 이를 Truncated SVD라고 한다. 모든 값을 다 사용하는 것이 아니라 적은 정보만으로 A와 근사한 행렬을 도출할 수 있는 것이다. 이렇게 적은 정보의 근사 행렬을 사용하는 것이 추천에서의 Low-rank matrix factorization이다.

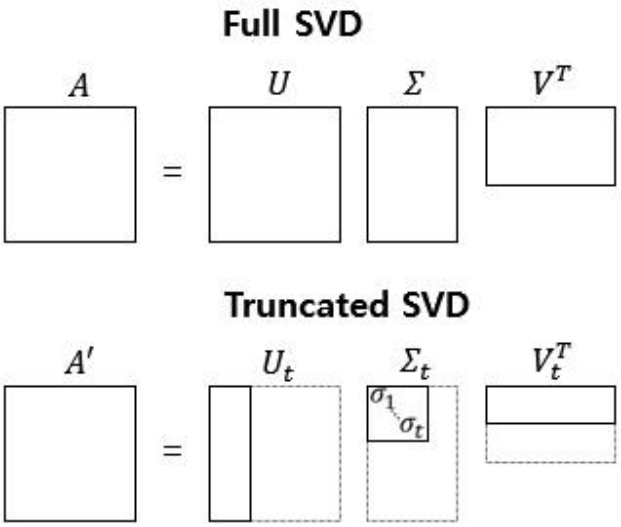


그림 29. Full SVD와 Truncated SVD

신호 처리 관점에서 보면, 선형 저역 신호 성분이 곧 분해 행렬의 기본 구성 요소에 해당한다. 이러한 관점에서 Low-rank 행렬 분해는 컨볼루션 정규화를 사용하는 무한 계층 GCN (an infinite layer GCN with convolutional normalization)과 동일하다.

GF-CF는 다음 수식 같은 선형 필터와 이상 필터를 결합한 Closed-form으로 구할 수 있다. $\tilde{R}^T\tilde{R}$이 선형 필터, $D_I^{-\frac{1}{2}}\overline{U}\overline{U}^T D_I^{\frac{1}{2}}$이 이상 필터에 해당한다. $\tilde{R}$는 정규화된 평점 행렬, D_I는 상품 차수 행렬, $\overline{U}$는 $\tilde{R}$의 top-K 특이 벡터로, Truncated SVD로부터 구해진다.

$$s_u = r_u(\tilde{R}^T\tilde{R} + \alpha D_I^{-\frac{1}{2}}\overline{U}\overline{U}^T D_I^{\frac{1}{2}})$$

제 2 절. 그래프 대조 학습 기반 개인화된 추천 알고리즘 개발

1. 기존 추천 알고리즘의 문제점

추천 도메인에서 많이 쓰이는 저역 통과 필터는 연결이 되어있는 노드들을 유사하게 필터링한다. 예를 들어, 추천 도메인에서는 같은 영화를 본 사용자들을 유사하게 필터링한다. 하지만, 같은 영화를 본 사용자 중에서 서로 다른 유형의 사용자에 대해서는 저역 통과 필터의 한계가 존재한다. 저주파 통과 필터를 적용하면 이미지의 인접한 픽셀들이 유사해지듯이 사용자 간의 정보가 유사해지기에 인기 상품들만 추천되는 문제가 존재한다.

그림 30의 긴 꼬리 현상(Long-tail problem)은 널리 알려진 추천 시스템의 현상으로, 소수의 상품에 대해 소비가 많고 나머지는 소비가 적어 데이터 불균형이 생기는 것이다. 이러한 특수성 때문에 저역 통과 필터 적용 시 추천 도메인에서는 특히 다양한 상품 추천이 어려울 수 있다.

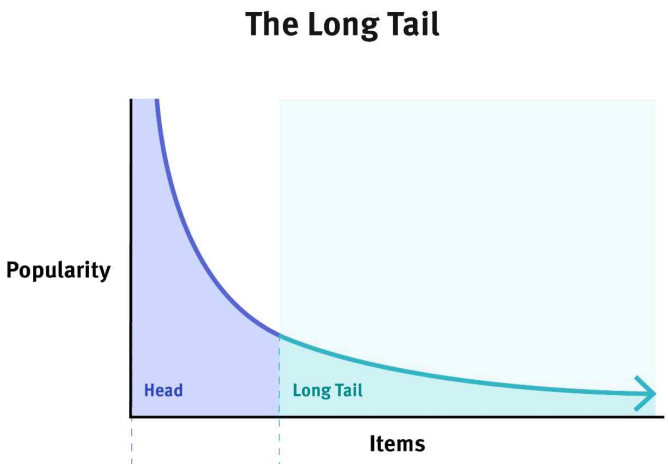


그림 30. 긴 꼬리 현상

이 현상을 확인하기 위해 Amazon-book 데이터에 대해 사용자 행동 패턴 분석을 진행했다. 소비 횟수는 상품이 전체 유저들에 의해 소비된 횟수이다. 그림 31에서 볼 수 있듯이 저주파 그래프 필터만을 사용한 GF-CF는 비교적 인기 있는 상품은 잘 예측하지만, 인기가 없는 상품에 대해서는

잘 찾아내지 못하는 한계를 발견했다. 이 경우, 성능이 아무리 좋아도 사용자의 만족감은 떨어질 수밖에 없다. 사용자들은 자신의 행동 패턴에 맞는 새롭고 신선한 상품 추천을 원하기 때문에 단순히 주요 성분을 추출해 사용하는 것이 아닌, 주요 성분에서 관측되지 않는 사용자 각각의 행동 패턴을 잘 반영하는 필터가 필요할 것이다.

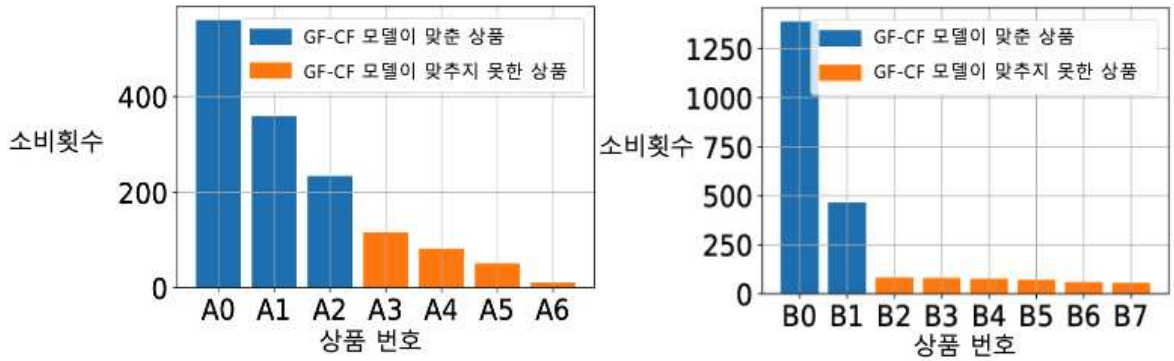


그림 31. Amazon-book 데이터에 대한 사용자 행동 패턴 분석

2. 그래프 필터 기반 추천 시스템

그래프 필터 기반 추천 시스템은 최근 몇 년간 그래프 신호 처리 분야에서 높은 관심을 받고 있다. 다양한 연구에서 이 방법을 활용하여 추천 시스템의 성능을 향상시키는 방법을 연구하고 있으며, 크게 3단계로 나눌 수 있다.

가. 그래프 신호 처리

그래프 신호 처리는 그래프 데이터에서 신호를 추출하고 처리하는 기술이다. 이를 위해서는 그래프 내 각 노드의 연결성과 상호작용을 분석하고, 이를 통해 노드 간의 유사도와 중요도를 측정한다.

나. 필터링

그래프 신호를 처리하기 위해 필터를 적용한다. 필터는 주파수 영역에서 신호를 변형시키는 작업을 수행한다. 이를 통해 그래프 신호를 저주파, 고주파로 나누어 필터링을 수행한다.

다. 추천

필터링된 그래프 신호를 이용하여 추천을 수행한다. 이를 위해서는 추천 알고리즘을 활용하여 유저의 선호도와 상품의 특성을 분석하고, 이를 통해 새로운 추천 아이템을 도출한다.

이러한 기술들을 활용하여 현재 그래프 필터 기반 추천 시스템은 저주파 필터링을 기반으로 하는 방법과 고주파 필터링을 기반으로 하는 방법으로 나뉘어 있다. 또한, 이러한 방법들은 딥러닝과의 결합을 통해 더욱 정확하고 효과적인 추천 시스템을 구현하는 방법으로 발전하고 있다.

스펙트럴 그래프 필터(Spectral Graph Filter)는 최신 그래프 신경망 GNN, GCN 등의 핵심 구성 요소이며, GNN 및 GCN이 사용되는 추천 시스템 분야에서도 많이 사용되고 있다. 추천 시스템에서 주로 사용되는 필터는 저역 통과 필터(Low—pass filter)이다. 저역 통과 필터를 사용하게 되면, 그림 32과 같이 유저들은 더욱 유사해지게 되고, 다른 유저들이 소비한 상품을 다른 유사한 유저에게 추천해주게 된다. NGCF, LightGCN, GF-CF 모두 저역 통과 필터만을 사용하며, 그 중 LightGCN이 사용하는 필터는 선형 저역 통과 필터에 해당한다.

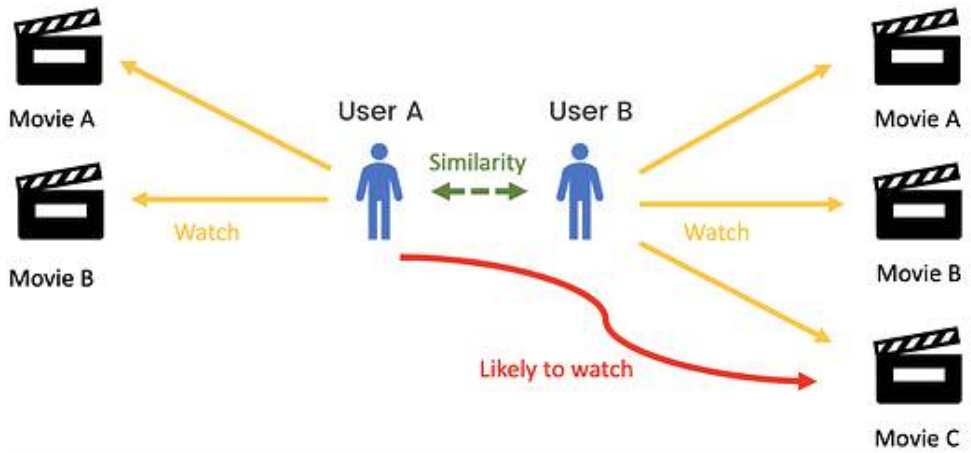


그림 32. 유사 상품 추천 과정

3. 그래프 필터를 이용한 추천 알고리즘 고도화

행렬 분해부터 그래프 컨볼루션 방법에 이르기까지 협업 필터링을 위해 다양한 방법이 제안되었다. 특히 그래프 컨볼루션 기반의 협업 필터링 방법은 높은 정확도를 보여준다. 사용자-아이템 인터랙션을 이분 그래프로 나타내고 그래프 컨볼루션 기술을 적용한다. 다양한 그래프 컨볼루션 연산 중에서도 모두 비교적 간단한 선형 또는 저역 통과 필터를 사용한다. 놀랍게도 이러한 접근 방식은 이제 다른 딥러닝 기반 방법을 능가한다. 이 연구에서는 BSPM (Blurring-sharpening process model)의 새로운 개념을 제시한다. 블러링 (blurring)과 샤프닝 (sharpening) 프로세스는 미분 방정식으로 공식화된다. 또한 기존 그래프 컨볼루션 기반 협업 필터링 방법 중 일부가 우리 모델의 특수한 경우임을 보여준다.

연도	모델	Blurring process (저주파 통과 필터)	Sharpening process (고주파 통과 필터)	학습 단계
2019	NGCF	O	X	O
2020	LightGCN	O	X	O
2021	GF-CF	O	X	X
2022	BSPM	O	O	X

표 17. 기존 그래프 기반 추천 모델과의 비교

BSPM은 단순하고 계산적으로 효율적이지만 높은 정확도를 보여주는 GF-CF의 최근 성공에 영감을 받았다. GF-CF는 사용자-아이템에 대한 임베딩 벡터를 학습하지 않고 직접 알려지지 않은 사용자-아이템 상호작용을 유도하기 위해 사용자-아이템 인터랙션 행렬을 처리한다. GF-CF가 고무적인 결과를 보여주었지만, 이미지 생성에서 널리 사용되는 교란-복구 (pertubation-recovery) 패러다임이 이를 크게 향상시킬 수 있음을 발견했다. 예를 들어 디퓨전 (diffusion) 모델 (또는 score-based 생성 모델) [28]은 이미지 생성 영역에서 최첨단 품질을 보여준다. 확산 모델에서는 특정 유형의 확률적 미분 방정식을 채택하여 순방향 및 역방향을 설명하기 위해 적용된다 (역방향 프로세스는 생성 모델로 간주 됨).

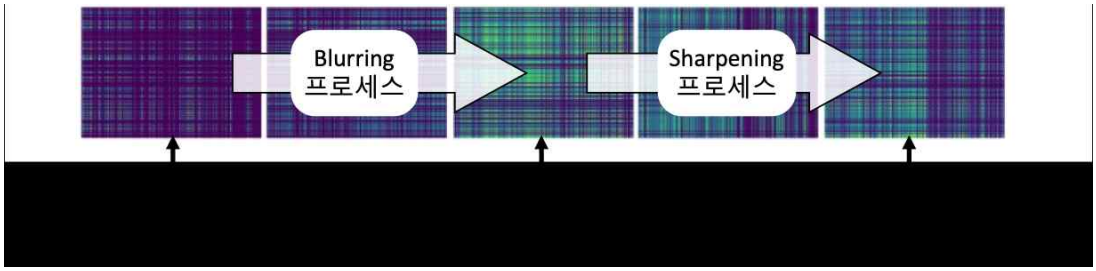


그림 33. BSPM은 두 결정적인 블러링과 샤프인 프로세스를, 디퓨전 모델은 여러 이미지를 학습하지만 BSPM은 하나의 인터랙션 행렬만 사용함

BSPM의 전체 모델 설계는 perturbation-recovery 아키텍처를 가지고 있다. 즉 블러링 프로세스는 사용자-아이템 행렬의 원래 정보를 손상 또는 교란 하고 샤프닝 프로세스는 유망한 추가 정보와 함께 원래 정보를 복구하려고 시도한다. GF-CF와 같은 기존 방법은 행렬에 특정 블러링 필터를 적용하는 반면, 우리는 처음으로 협업 필터링에 대한 Blurring-sharpening 프로세스 패러다임을 처음으로 제안한다.

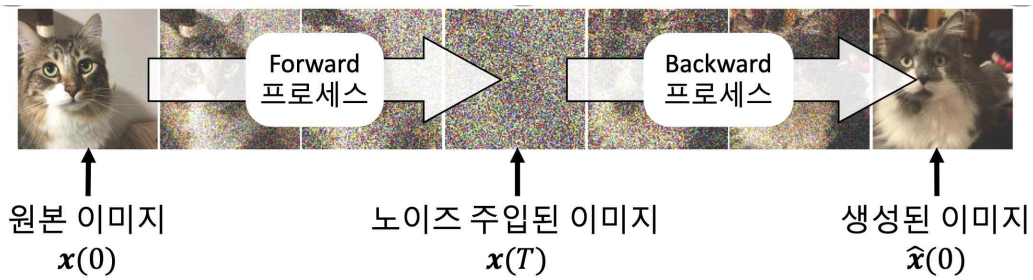


그림 34. 디퓨전 (diffusion) 모델은 두 확률 프로세스를 사용함
순방향 프로세스는 perturbation을 하고 역방향 프로세스는 recovery를 하고,
recovery 프로세스가 확률적이기 때문에 오리지널 샘플로 수렴하지 않고 다른 비슷한 샘플로 수렴함

BSPM의 전체 워크플로우는 인터랙션 행렬 R에 연속 블러링 프로세스를 적용하여 블러링 행렬을 파생시킨 다음 연속 샤프닝 프로세스를 적용하는 것이다. 이러한 과정을 거친 후 아이템을 추천할 수 있다. BSPM의 방법에는 학습한 것이 없다는 것을 분명히 한다. 이 과정에서 신경망은 전혀 사용되지 않으며 사용자-아이템 임베딩 벡터를 학습하지 않는다. 다시 말하자면 BSPM은 학습 가능한 매개변수가 없다. 따라서 프로세스가 놀라울 정도로 간단하고 전체 계산을 빠르게 수행할

수 있다.

블러링 프로세스. 블러링 프로세스는 협업 필터링의 핵심이다. 많은 그래프 기반 협업 필터링은 블러링 프로세스 혹은 저주파 통과 필터에 해당하는 그래프 컨볼루션 필터를 사용한다. 일반적으로 인기 있는 아이템은 이 과정을 거쳐 사용자에게 추천된다. 블러링 프로세스는 아래와 같이 공식화할 수 있다.

$$\boldsymbol{B}(T_b) = \boldsymbol{B}(0) + \int_0^{T_b} b(\boldsymbol{B}(t))dt$$

b는 블러링 함수이며 $\boldsymbol{B}(0)$은 인터랙션 행렬 $\boldsymbol{R}$이다. $\boldsymbol{B}(T_b)$는 블러링 된 인터랙션 행렬을 의미한다. 함수 b는 저주파 통과 필터라 할 수 있는 열 방정식 (또는 선형 필터)와 이상 저주파 통과 필터로 구성된다.

샤프닝의 의미. 샤프닝 프로세스는 블러링 프로세스의 반대이므로 사용자 특정 아이템을 찾는다. 일반 GCN의 경우 유사하게 선명하게 하는 과정이 노드 피쳐 간의 차이를 강조하는 데 사용된다. 덜 인기 있는 아이템도 인기 있는 항목과 함께 사용자에게 추천될 수 있다. 샤프닝 프로세스는 아래와 같이 공식화할 수 있다. s는 $\boldsymbol{S}(t)$에 대한 시간 변화량을 근사하는 샤프닝 함수이고 $\boldsymbol{S}(T_s)$는 $\boldsymbol{S}(0)$으로부터 샤프닝된 행렬을 의미한다.

$$\boldsymbol{S}(T_s) = \boldsymbol{S}(0) + \int_0^{T_s} s(\boldsymbol{S}(t))dt,$$

두 프로세스를 사용하는 BSPM 모델은 다음과 같이 나타낼 수 있다.

$$\hat{\boldsymbol{R}} = \begin{cases} \boldsymbol{B}(T_b) + \boldsymbol{S}(T_s), & \text{if residual} \\ \mathrm{S}(\mathrm{T_s}) & \text{,otherwise} \end{cases}$$

BSPM은 잔차 연결을 사용하는 경우와 그렇지 않은 경우로 설계할 수 있다. 잔차 연결을 사용하는 경우 블러링 된 행렬과 샤프닝 된 행렬을 모두 사용하여 최종 예측 인터랙션 행렬 $\hat{\boldsymbol{R}}$을 구성한다.

4. 그래프 필터를 이용한 추천 알고리즘의 성능 평가

데이터셋	사용자 수	아이템 수	인터랙션 수	밀집도
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-book	52,643	91,599	2,984,108	0.00062

표. 18 데이터셋 통계

기존 모델과의 성능을 비교하기 위해 문헌에서 가장 자주 사용되는 세 가지 벤치마크 데이터 세트인 Gowalla, Yelp2018 및 Amazon-book을 사용한다 (표 18 참고). Recall@20 및 NDCG@20이라는 두 가지 널리 사용되는 순위 측정 기준을 채택한다. 사용자와 상호 작용이 없는 모든 아이템은 사용자의 추천 후보이다. 이전 연구와의 비교를 공정하게 유지하기 위해 동일한 데이터 세트와 동일한 학습/테스트 분할을 사용한다.

놀랍게도 BSPM은 기존의 모든 인기 있는 협업 필터링 알고리즘을 큰 차이로 능가한다(표 19 참고).

모델	Gowalla		Yelp2018		Amazon-book	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
LightGCN	0.1830	0.1554	0.0649	0.0530	0.0411	0.0315
LT-OCF	0.1875	0.1574	0.0671	0.0549	0.0442	0.0341
GF-CF	0.1875	0.1574	0.0703	0.0579	0.0710	0.0584
BSPM	**0.1910**	**0.1581**	**0.0719**	**0.0591**	**0.0733**	**0.0609**

표 19. 모델 성능 비교

뿐만 아니라 그림 33에서 알 수 있듯이 BSPM의 샤프닝 효과는 단순한 추천 성능의 향상이 아니다. 그림 35의 (b), (d), (e)에서는 Coverage와 Novelty의 척도를 사용하여 추천 모델의 성능뿐만 아니라 추천 아이템의 다양성과 참신성을 측정한다. BSPM의 경우 샤프닝 프로세스를 추가함으로써 두 메트릭 모두 증가시킬 수 있다는 것을 모든 벤치마크 데이터셋에서 보인다.

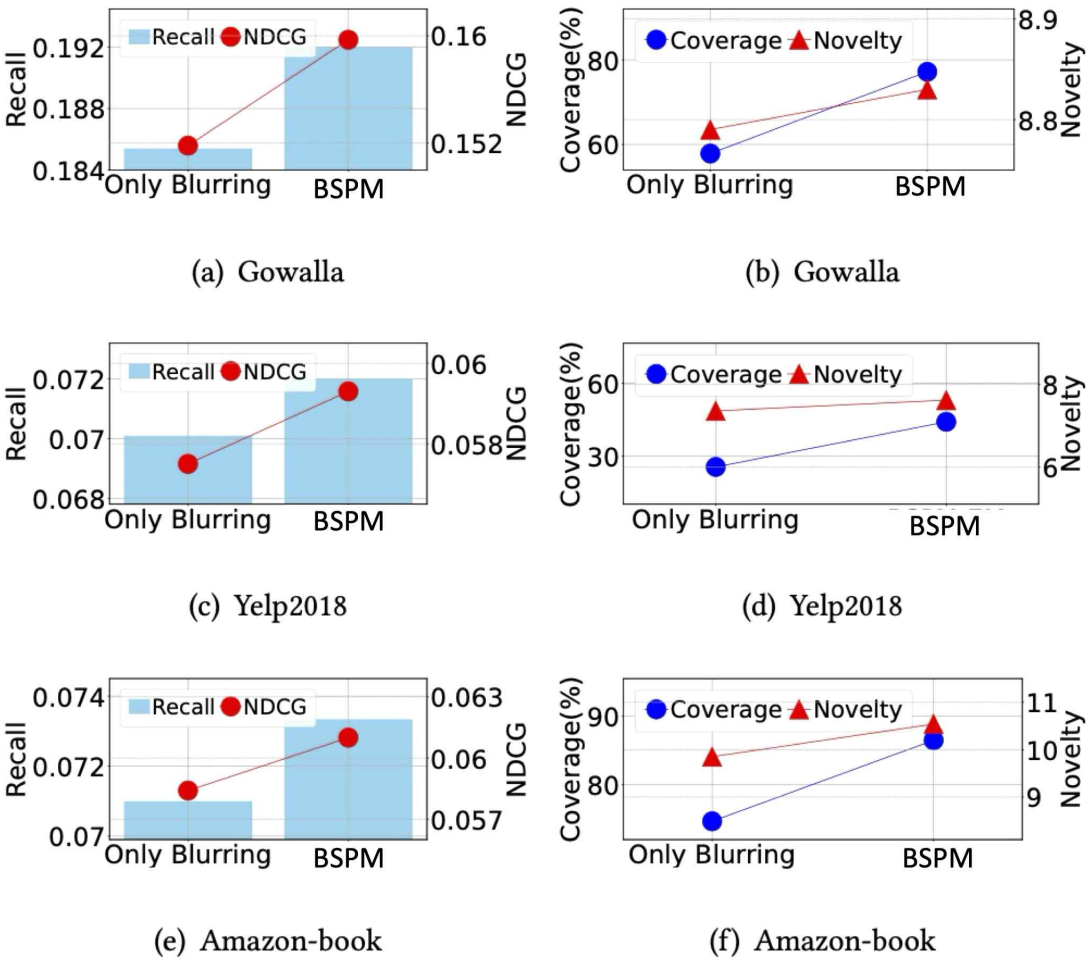


그림 35. Only Blurring과 BSPM의 정확도 및 Coverage, Novelty 메트릭 비교

5. 개인화를 위한 그래프 대조 학습 배경지식

최근 몇 년 동안 그래프 기반 추천 시스템에 관한 연구가 활발하였다. 사용자-아이템 상호작용의 고차 연결성 정보를 고려하여 협업 신호를 추출하는 기술이 발전되었다. 그러나 사용자-아이템 상호작용에 대한 문제점이 협력 신호 (Collaborative Filtering)의 추출을 방해한다는 연구가 진행되기 시작하였다. 이에 관한 문제점은 세 가지로 요약할 수 있다.

가. 사용자-아이템 상호작용이 너무 희소하여 지도 학습이 어렵다.

나. 추천 시스템에서 널리 알려진 긴 꼬리 현상(Long-tail problem)으로 인해 추천 시스템이 자주 상호작용되는 아이템, 즉 차수가 높은 아이템에 편향되어 차수가 낮은 아이템의 추천이 어렵다.

다. 사용자가 클릭하거나 보는 등의 내재적 피드백 (Implicit Feedback)의 경우 잘못 클릭하거나 소비한 후 만족하지 못한 경우와 같이 상호 작용에서 노이즈가 많기 때문에 학습에 방해가 된다.

이러한 문제점을 극복하기 위해, 자기 지도 학습(Self-Supervised Learning)을 활용한 추천 시스템이 큰 주목을 받고 있다. 자기 지도 학습은 비지도 학습(Unsupervised Learning)의 한 형태로, 라벨링 되어있지 않은 데이터에서 스스로 특징을 추출하고 학습하는 방법을 의미한다. 자기 지도 학습은 라벨이 부족하거나 라벨을 생성하기 어려운 상황에서 유용하다. 따라서 추천 시스템에서 사용되는 사용자-아이템 상호 작용에 대한 문제점을 자기 지도 학습을 통해 해결할 수 있다.

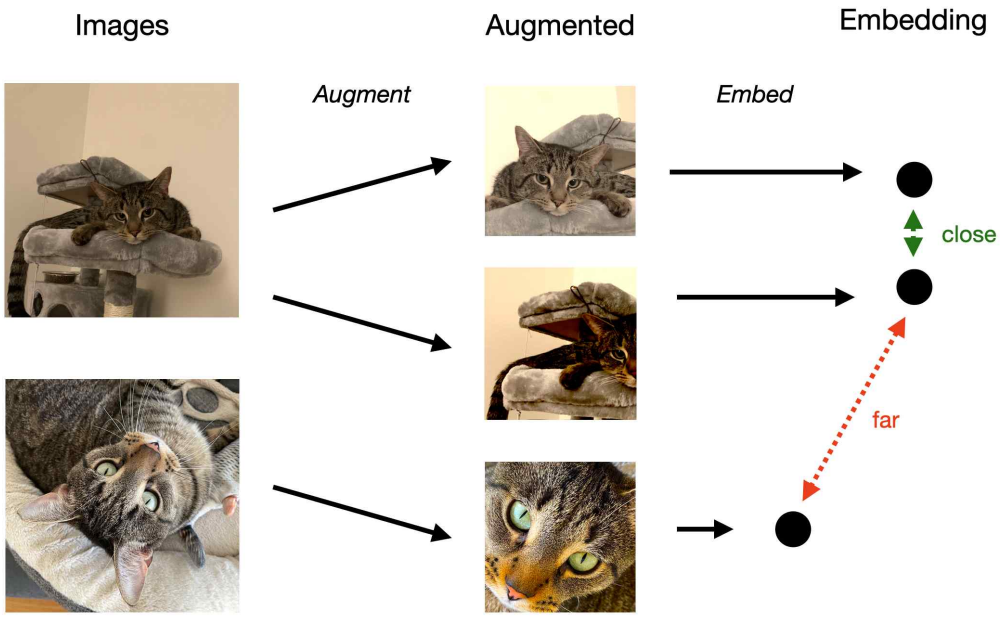


그림 36. 대조 학습 예시

대조 학습(Contrastive Learning)은 자기 지도 학습의 주된 학습 방법으로, 데이터 내의 유사성과 차이를 학습하여 특징을 추출하는 방법이다. 유사한 샘플들은 서로 가깝게, 다른 샘플들은 서로 멀리 배치함으로써 특징을 더 잘 학습하게 한다. 이때 유사한 샘플은 동일한 샘플을 다양한 방식으로 증강해 얻는 경우가 많다. 대조 학습은 이미지, 자연어 처리 분야에서 널리 사용되어 왔다. 대조 학습의 예시는 그림 36과 같다.

대조 학습은 유사한 샘플 간의 유사도를 높이고, 다른 샘플 간의 유사도를 낮추는 방식으로 진행된다. 이를 공식화한 대조 학습 손실 함수는 다음과 같다.

$$l_{i,j} = -\log \frac{\exp(\sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N_{[k \neq i]}} \exp(\sim(z_i, z_k)/\tau)}$$

6. 그래프 대조 학습 기반 추천 시스템의 관련 연구

이미지, 자연어 처리에서 데이터를 증강하는 방식은 그래프 기반 추천 시스템에 그대로 적용시킬 수 없다. 사용자-아이템 상호작용의 피쳐 (feature)는 원-핫 인코딩 (one-hot encoding)이나 다른 범주형 변수와 같이 이산형 변수 (discrete variable)이다. 따라서 이미지의 증강 방법들인 회전, 자르기, 혹은 흐리게 하기 등의 연산을 할 수 없다. 또한 각 데이터 객체를 격리된 것으로 보는 이미지, 자연어 처리와는 달리, 사용자와 아이템은 본질적으로 연결되어 있고, 서로 의존적이다. 따라서 그래프 기반 추천에 알맞은 증강 방식을 사용하는 것이 중요하다. 사용자-아이템 상호 작용은 이분 그래프 (bipartite graph)로, 협력 신호를 포함하고 있다. 첫 번째 홉 (hop) 이웃은 사용자의 아이템 구매 기록 혹은 아이템이 상호 작용한 사용자들에 대한 정보를 갖고 있다. 따라서 사용자 혹은 아이템의 기존 특징으로 볼 수 있다. 두 번째 홉 이웃은 유사한 행동을 하는 유사한 사용자, 혹은 유사한 사용자가 구매한 유사한 아이템으로 볼 수 있다. 또한, 사용자에서 아이템으로의 고차 경로는 아이템에 대한 사용자의 잠재적 관심사로 볼 수 있다. 따라서 그래프 기반 데이터 증강 방식은 그래프 구조의 고유 패턴을 활용해야 한다.

SGL[29]은 그래프 증강을 위하여 세 가지의 그래프 구조에 대한 연산을 제안한다. 1) 노드 드롭아웃 (Node Dropout, ND), 2) 엣지 드롭아웃 (Edge Dropout, ED), 3) 랜덤 워크 (Random Walk, RW). 세 연산자는 모두 동일한 방식으로 적용된다. 그래프 증강 수식은 다음과 같다.

$$Z_1^{(l)} = H(Z_1^{(l-1)}, s_1(G)), Z_2^{(l)} = H(Z_2^{(l-1)}, s_2(G)), s_1, s_2 \sim S$$

$Z_1^{(l)}$와 $Z_2^{(l)}$는 l번째 레이어에서 증강된 노드들의 뷰 (view)이다. 기존 그래프 G에 각각 증강 연산 s_1와 s_2를 적용한 후 이전 레이어에서 얻은 뷰와 집계(aggregation)하여 얻을 수 있다. 증강 연산자 s는 다음 수식들로 공식화할 수 있다.

첫 번째로 노드 드롭아웃은 다음과 같다.

$$s_1(G) = (M' \odot V, E), s_2(G) = (M'' \odot V, E)$$

각 노드는 ρ의 확률로 연결된 엣지와 함께 제거된다. M'과 M''은 0 혹은 1의 값을 갖는 마스킹 벡터 (Masking vector)로 노드 집합에 적용되어 두 개의 서브 그래프를 생성한다. 이 연산을 사용할 경우, 서로 다르게 증강된 뷰에 대해서 영향력 있는 노드들을 구분해 내도록 학습할 수 있

기 때문에, 그래프 구조에 영향을 덜 받게 된다.

두 번째로 엣지 드롭아웃은 다음 수식과 같다.

$$s_1(G) = (V, M_1 \odot E), \;\; s_2(G) = (V, M_2 \odot E)$$

각 엣지는 ρ의 확률로 제거된다. M_1과 M_2은 0 혹은 1의 값을 갖는 마스킹 벡터로 엣지 집합에 적용되어 두 개의 서브 그래프를 생성한다. 이 연산을 사용할 경우, 노드의 국소적 구조에서 유용한 패턴을 찾아낼 수 있게 해서 노이즈가 많은 상호 작용에 덜 민감하게 학습할 수 있다.

세 번째로 랜덤 워크는 다음 수식과 같다.

$$s_1(G) = (V, M_1^{(l)} \odot E), \;\; s_2(G) = (V, M_2^{(l)} \odot E)$$

노드 드롭아웃과 엣지 드롭아웃은 마스킹 벡터가 모든 레이어에서 동일하다. 랜덤 워크는 각 레이어마다 다른 서브 그래프를 생성한다. 각 레이어마다 엣지 드롭아웃을 진행한다. $M_1^{(l)}$과 $M_2^{(l)}$은 각 레이어마다 적용되는 마스킹 벡터이다.

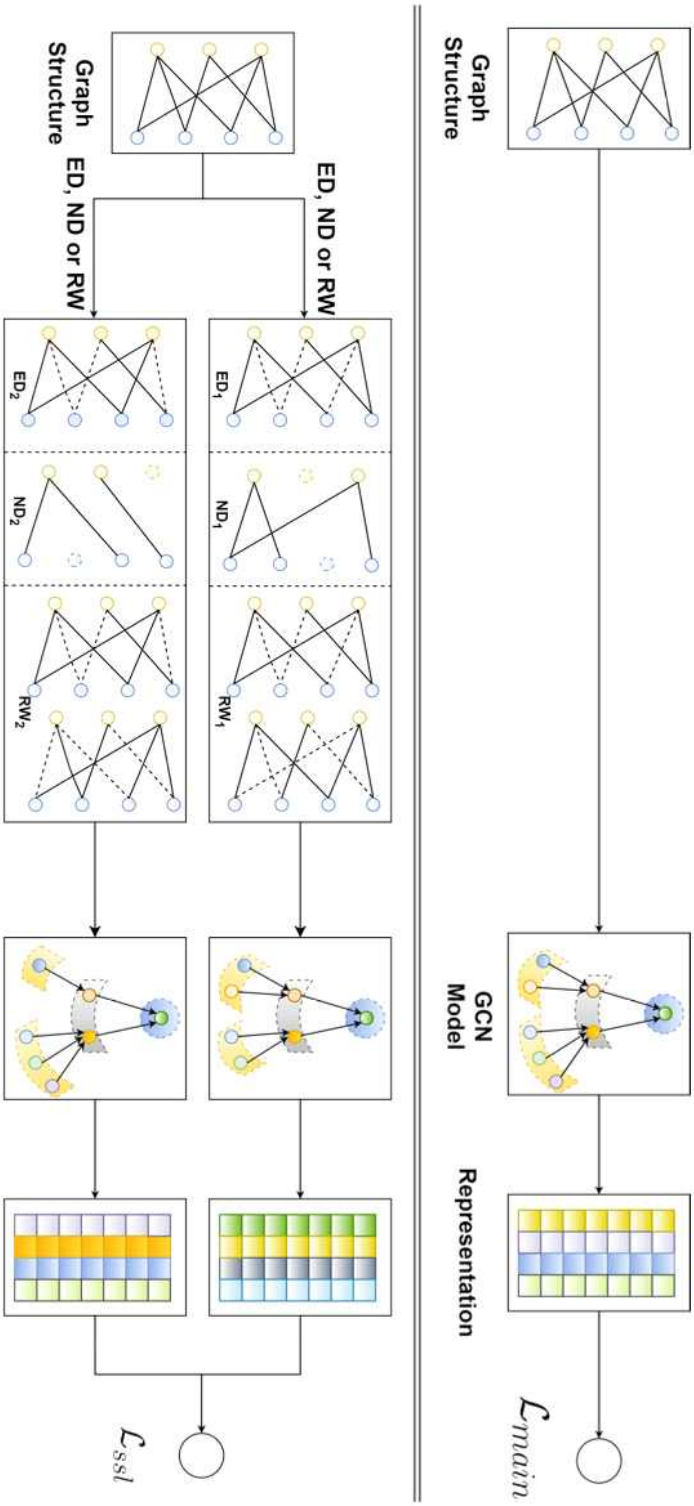


그림 37. SGL의 아키텍처

SGL은 LightGCN을 기반으로 기존과 같이 BPR 손실 함수를 통해 지도 학습하는 부분과, 증강된 뷰에 대해서 대조 함수를 통해 비지도 학습하는 부분으로 나뉜다. 비지도 학습하는 부분에서는, 그래프 증강 연산자를 통하여 비슷하지만 다른 두 개의 뷰를 만들어 낸다. 이때 동일한 노드에 대한 뷰를 양성 쌍 (Positive pair)로 보고, 서로 다른 노드에 대한 뷰를 음성 쌍 (Negative pair)으로 보고 대조 학습을 시행한다. 대조 학습에 사용되는 손실 함수는 InfoNCE[상호참조] 함수로, 양성 쌍들을 더 비슷하게 만들어 주고, 음성 쌍들을 더 다르게 만들어 준다. SGL에서 사용한 InfoNCE 함수는 다음과 같다.

$$L_{ssl}^{user} = \sum_{u \in v} -\log \frac{\exp(s(z_u', z_u''/\tau)}{\sum_{v \in U} \exp(s(z_u', z_v'')/\tau)}$$

$s(\cdot)$은 두 벡터 간의 유사도를 측정한 것이고, 코사인 유사도 (cosine similarity)가 사용되었다. τ는 소프트맥스 (softmax) 함수에 사용되는 하이퍼파라미터이다. 사용자와 아이템에 대하여 자기 지도 학습을 위한 손실 함수를 모두 계산한다.

표 20에서 실험적으로 SGL의 성능을 확인할 수 있다. 그래프 증강을 통한 대조 학습이 성능 향상에 큰 영향을 미친 것을 확인할 수 있다.

Dataset	Yelp2018		Amazon-Book		Alibaba-iFashion	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
NGCF	0.0579	0.0477	0.0344	0.0263	0.1043	0.0486
LightGCN	0.0639	0.0525	0.0411	0.0315	0.1078	0.0507
Mult-VAE	0.0584	0.0450	0.0407	0.0315	0.1041	0.0497
DNN+SSL	0.0483	0.0382	0.0438	0.0337	0.0712	0.0325
SGL-ED	**0.0675**	**0.0555**	**0.0478**	**0.0379**	**0.1126**	**0.0538**
%Improv.	5.63%	5.71%	9.13%	12.46%	4.45%	6.11%
p-value	5.92e-8	1.89e-8	5.07e-10	3.63e-10	3.34e-8	4.68e-10

표 20. SGL모델 성능 비교

SimGCL[30]은 SGL에서 사용한 그래프 증강 방식의 문제점을 지적하며 제안되었다. 노드 드롭, 엣지 드롭, 그리고 랜덤 워크 연산 방식은 시간이 오래 걸리고, 중요한 노드나 엣지를 제거하는

경우 연결된 그래프를 몇 개의 연결되지 않은 그래프로 나눌 수 있어 기존 그래프와 너무 달라져 학습이 어려워질 수 있다. 따라서 드롭아웃 기반의 방식으로 그래프를 증강하는 것이 아닌, 작은 노이즈를 더해 임베딩 공간상에서 데이터를 증강한다. 노이즈의 크기를 적절하게 조절하여 사용하기 때문에 학습을 방해하지 않고, 노이즈가 임베딩 공간의 분포를 더 균일하게 만들어 주며, 구현이 간단하다는 장점이 있다. 임베딩 공간에서 노이즈를 추가하는 수식은 다음과 같다.

$$e_i' = e_i + \Delta_i', e_i'' = e_i + \Delta_i''$$

위 수식의 e_i는 노드 i에 대한 임베딩이고, Δ_i'와 Δ_i''는 더해지는 노이즈 벡터이다. 노이즈 벡터에는 몇 가지 조건이 있는데, 첫 번째는 $\| \Delta \|_2 = \epsilon$ 로, 노이즈 벡터의 크기를 반지름이 ϵ인 원으로 제한한다는 것이다. 두 번째는 $\Delta = \overline{\Delta} \odot sign(e_i)$, $\overline{\Delta} \in \mathbb{R}^d \sim U(0,1)$로 e_i, Δ', Δ''가 같은 사분면에 있도록 해서, 노이즈를 더하는 것이 임베딩에 너무 큰 영향을 미치지는 않도록 규제한다.

그림 38은 노이즈를 더하는 것의 효과를 $\mathbb{R}^2$에서 설명한다. 노이즈를 추가함으로써 임베딩 벡터가 작지만, 충분히 회전하여 서로 다른 증강 표현을 생성해 낸다.

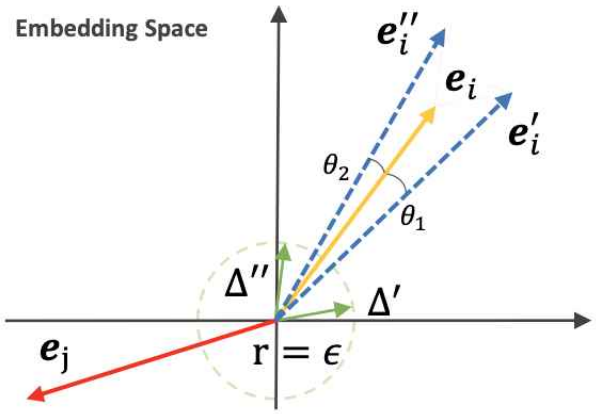


그림 38. SimGCL 노이즈 추가 그림

SimGCL이 사용한 대조 학습의 손실 함수는 다음 수식과 같다.

$$L_{cl} = \sum_{i \in B} -1/\tau + \log(\exp(1/\tau) + \sum_{j \in B/\{i\}} \exp(z_i^\top z_j/\tau))$$

이는 SGL에서 사용한 손실 함수를 변형하여 사용한 것이다. 왜냐하면 $1/\tau$가 상수이기 때문에 실제로 대조 학습 손실 함수가 최소화하는 것은 다른 노드 임베딩 간의 코사인 유사도이다. 이는 연결된 노드가 노드의 차수가 높은 허브 노드에서 멀어지도록 하여 임베딩의 분포를 더욱 균등하게 만들어 준다.

표 21에서 실험적으로 SimGCL의 성능을 확인할 수 있다. 다른 모델들 대비 성능이 크게 향상된 것을 확인할 수 있다.

Method	Douban-Book		Yelp2018		Amazon-Book	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	0.1501	0.1282	0.0639	0.0525	0.0411	0.0315
Mult-VAE	0.1310	0.1103	0.0584	0.0450	0.0407	0.0315
DNN+SSL	0.1366	0.1148	0.0483	0.0382	0.0438	0.0337
BUIR	0.1127	0.8938	0.0487	0.0404	0.026	0.0209
MixGCF	0.1731	0.1552	0.0713	0.0589	0.0485	0.0378
SimGCL	**0.1772**	**0.1583**	**0.0721**	**0.0601**	**0.0515**	**0.0410**

표 21. SimGCL 모델 성능 비교

Method	Yelp2018		Amazon-Book	
	Recall@20	NDCG@20	Recall@20	NDCG@20
LightGCN	0.0639	0.0525	0.041	0.0318
SGL-ND	0.0644	0.0528	0.044	0.0346
SGL-ED	**0.0675**	**0.0555**	**0.0478**	**0.0379**
SGL-RW	0.0667	0.0547	0.0457	0.0356
SGL-WA	0.0671	0.055	0.0466	0.0373
CL Only	0.0245	0.019	0.0314	0.0258

표 22. SimGCL, LightGCN, 그리고 SGL 간 성능 비교

그림 39와 그림 40은 각각 추천 시스템에서 자주 사용되는 데이터인 Yelp2018과 Amazon-Book 데이터에 대해서 학습된 아이템 임베딩의 분포를 나타낸 것이다.

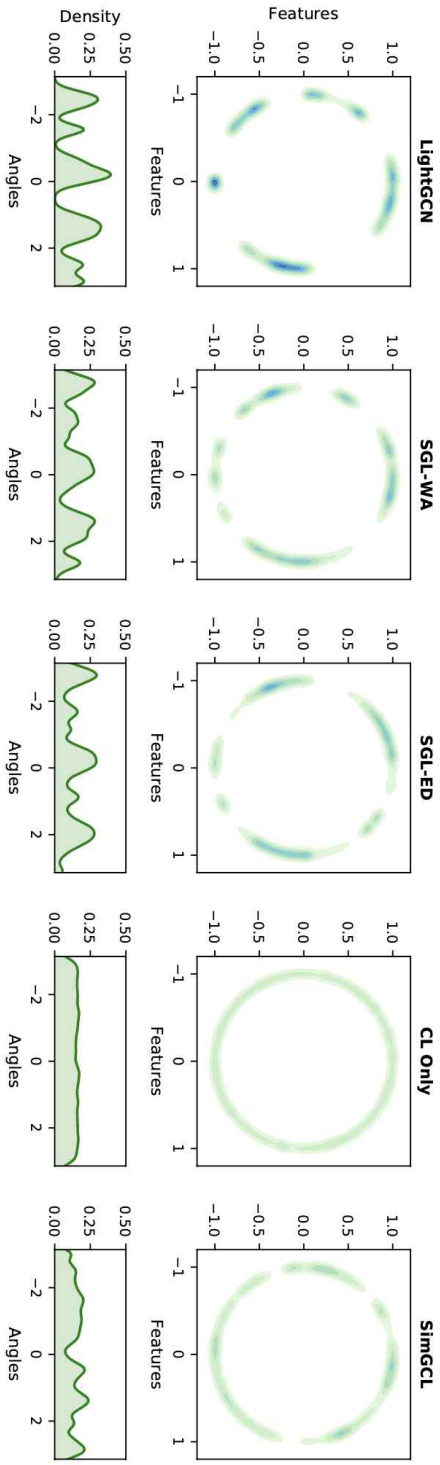


그림 39. SimGCL Yelp2018 데이터에 대한 아이템 임베딩 분포

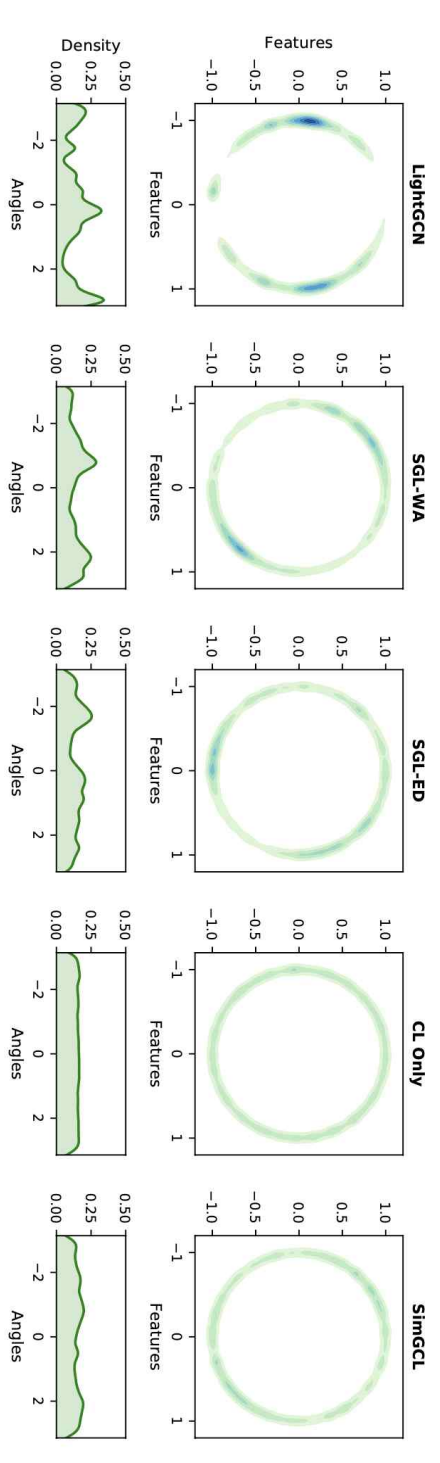


그림 40. SimGCL Amazon-Book 데이터에 대한 아이템 임베딩 분포

성능 결과와 함께 비교한다면, 임베딩 분포가 적절하게 균일한 것이 성능에 큰 영향을 미친다는 것을 알 수 있다. LightGCN, SGL-WA, SGL-ED의 경우 임베딩의 분포가 고르지 못한 반면 SimGCL은 임베딩의 분포가 고르다. 더 고른 임베딩의 분포가 노드의 고유한 특성을 보존하고, 일반화 능력을 향상시킬 수 있기 때문이다. 반면 CL Only의 경우, 임베딩의 분포가 고르지만 성능이 매우 저조하다. 이는 균일성과 성능 간의 양의 상관 관계가 제한된 범위에서만 유지된다는 것을 의미한다.

그림 41에서는 SimGCL이 편향성을 제거하는 능력이 뛰어남을 보여준다. 그림의 가로축은 아이템을 소비 횟수에 따라 분류한 것으로, Unpopular는 소비 횟수 하위 80%의 그룹이며, Popular는 소비 횟수 상위 5%, 그리고 Normal은 나머지 아이템들의 그룹이다. SimGCL은 다른 모델 대비 성능이 가장 뛰어나다. 특히 모든 데이터에 대해서 Unpopular 그룹에 대한 성능이 다른 모델 대비 더 좋은데, 이는 모델이 유명하지 않은 아이템을 추천할 수 있는 능력을 갖췄다는 것이다. 그리고 이 능력은 그림 38과 그림 39를 함께 보았을 때 베딩 분포의 균일성과 양의 상관 관계가 있다는 것을 확인할 수 있다.

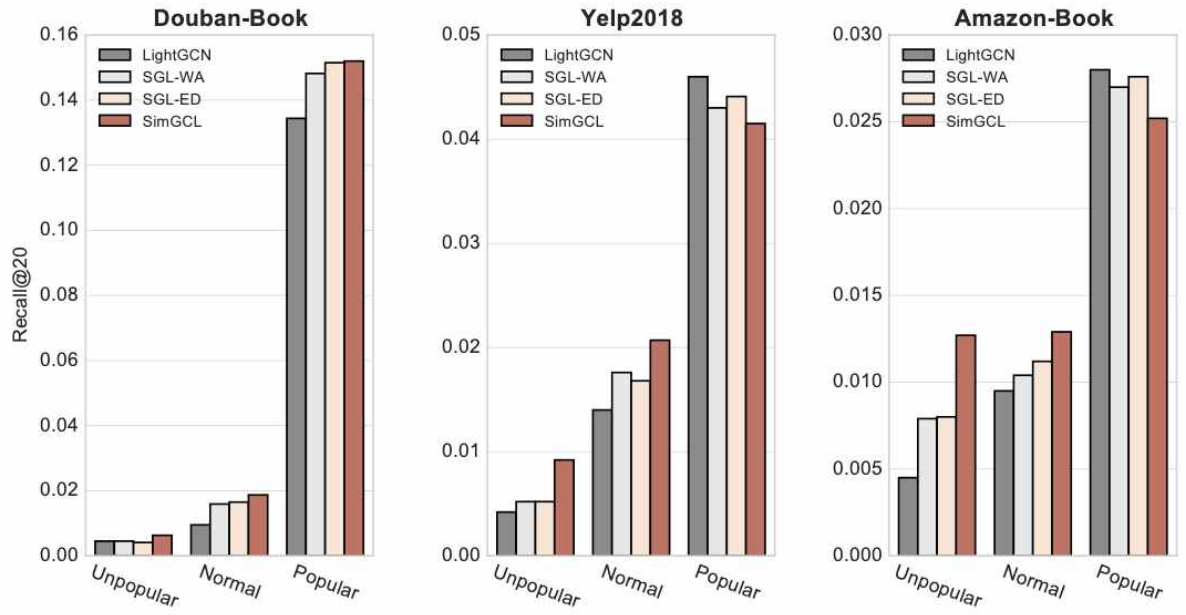


그림 41. 소비 횟수에 따른 아이템 그룹별 성능 비교

SGL은 노드 드롭아웃, 엣지 드롭아웃, 그리고 랜덤 워크 방식으로 그래프의 구조를 변경시켜 뷰를 증강시키고, SimGCL은 임베딩 공간에서 노이즈를 주입하여 뷰를 증강시키지만 여전히 이러한 방식들에 대한 문제점이 존재한다. 첫 번째는 이러한 무작위 간섭 (Random perturbation)을 통한 그래프 증강이 그래프의 유용한 구조적 정보를 손실시켜 학습을 어렵게 할 수 있다는 것이다. 두 번째는 이러한 경험적 (heuristic) 대조 학습의 성공 여부는 대부분 뷰를 생성하는 과정에 달려 있어, 모델의 일반성을 떨어트리고 노이즈가 많은 사용자에 취약하다는 점이다. 마지막으로, 최근 대조 학습을 사용한 그래프 기반 추천 시스템 대부분은 과도한 평탄함 문제 (Over-smoothing problem)를 겪고 있어, 표현들이 서로 구분되지 않게 된다.

LightGCL[31]은 이러한 문제를 해결하기 위해 간단하지만 효과적인 증강 방식을 제안한다. 기존 그래프에 특잇값 분해 (Singular Value Decomposition, SVD)를 적용해서 그래프를 증강시킴으로써, 사용자-아이템 상호 작용의 유용한 정보를 얻어내고 전역적인 협력 신호를 대조 학습에 사용할 수 있게 한다.

그림 42는 LightGCL의 전반적인 모델 구조를 나타낸 그림이다. 그림의 상단 부분은 기존 그래프에 GCN 연산을 적용하여 국소적인 그래프 정보를 얻어낸다. 그림의 하단 부분은, 특잇값 분해를 통해 재구성한 그래프에 GCN 연산을 적용하여 뷰를 증강시키고, 전역적이고 유용한 사용자-아이템 협력 신호를 추출한다.

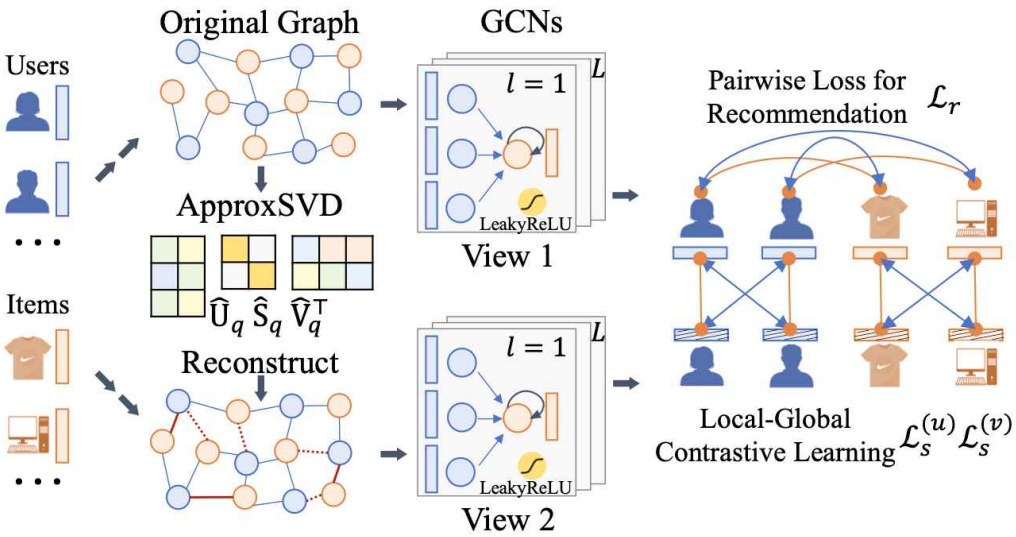


그림 42. LightGCL 아키텍처

기존 추천 시스템에서 대조 학습을 적용한 모델들은, 두 개의 증강된 노드 임베딩 뷰를 생성하여 그 둘을 대조한다. 따라서 기존 그래프에서 생성된 임베딩은 대조 학습 손실 함수에 사용되지 않는다. 이렇게 총 세 개의 뷰를 사용하는 것은 그들이 사용하는 무작위 간섭이 기존 그래프를 통해 생성한 임베딩에 안 좋은 영향을 끼치기 때문이다. 그러나 LightGCL은 그러한 무작위 간섭이 없이, 기존 그래프에서 생성한 임베딩을 국소적인 뷰로 보고, 특잇값 분해를 통해 재구성한 그래프에서 생성한 임베딩을 전역적인 뷰로 보고 둘 사이의 대조 학습을 진행할 수 있다. 다음은 LightGCL의 대조 학습에 대한 손실 함수다:

$$L_s^{(u)} = \sum_{i=0}^{I} \sum_{l=0}^{L} -\log \frac{\exp(s(z_{i,l}^{(u)}, g_{i,l}^{(u)}/\tau))}{\sum_{i'=0}^{I} \exp(s(z_{i,l}^{(u)}, g_{i',l}^{(u)})/\tau)}$$

$z_{i,l}^{(u)}$는 기존 그래프에서 생성한 뷰이고, $g_{i,l}^{(u)}$는 특잇값 분해를 통해 재구성한 그래프에서 생성한 뷰이다. 이 둘 사이의 대조 학습을 진행한다.

표 23에서 실험적으로 LightGCL의 성능을 확인할 수 있다. 다른 모델들 대비 성능이 크게 향상된 것을 확인할 수 있다.

Data	Metric	DGCF	HyRec	LightGCN	MHCN	SGL	SimGRACE	GCA	HCCF	SHT	SimGCL	LightGCL	p-val	impr.
Yelp	R@20	0.0466	0.0472	0.0482	0.0503	0.0526	0.0603	0.0621	0.0626	0.0651	0.0718	**0.0793**	7e−9	10%
	N@20	0.0395	0.0395	0.0409	0.0424	0.0444	0.0435	0.053	0.0527	0.0546	0.0615	**0.0668**	8e−9	8%
	R@40	0.0774	0.0791	0.0803	0.0826	0.0869	0.0989	0.1021	0.1040	0.1091	0.1166	**0.1292**	2e−9	10%
	N@40	0.0511	0.0522	0.0527	0.0544	0.0571	0.0656	0.0677	0.0681	0.0709	0.0778	**0.0852**	2e−9	9%
Gowalla	R@20	0.0944	0.0901	0.0985	0.0955	0.1030	0.0869	0.0896	0.107	0.1232	0.1357	**0.1578**	1e−6	16%
	N@20	0.0522	0.0498	0.0593	0.0574	0.0623	0.0528	0.0537	0.0644	0.0731	0.0818	**0.0935**	2e−6	14%
	R@40	0.1401	0.1356	0.1431	0.1393	0.1500	0.1276	0.1322	0.1535	0.1804	0.1956	**0.2245**	3e−6	14%
	N@40	0.0671	0.0660	0.0710	0.0689	0.0746	0.0637	0.0651	0.0767	0.0881	0.0975	**0.1108**	3e−6	13%
ML−10M	R@20	0.1763	0.1801	0.1789	0.1497	0.1833	0.2254	0.2145	0.2219	0.2173	0.2265	**0.2613**	1e−9	15%
	N@20	0.2101	0.2178	0.2128	0.1814	0.2205	0.2686	0.2613	0.2629	0.2573	0.2613	**0.3106**	3e−9	18%
	R@40	0.2681	0.2685	0.265	0.2250	0.2768	0.3295	0.3231	0.3265	0.3211	0.3345	**0.3799**	7e−10	13%
	N@40	0.2340	0.2340	0.2322	0.1962	0.2426	0.2939	0.2871	0.2880	0.3318	0.2880	**0.3387**	1e−0	17%
Amzazon	R@20	0.0211	0.0302	0.0319	0.0296	0.0327	0.0381	0.0309	0.0322	0.0441	0.0474	**0.0585**	2e−7	23%
	N@20	0.0154	0.0225	0.0236	0.0219	0.0249	0.0291	0.0238	0.0247	0.0328	0.0360	**0.0436**	2e−6	21%
	R@40	0.0351	0.0432	0.0499	0.0489	0.0531	0.0621	0.0498	0.0525	0.0719	0.0750	**0.0933**	1e−7	24%
	N@40	0.0201	0.0246	0.029	0.0284	0.0312	0.0371	0.0301	0.0314	0.0420	0.0451	**0.0551**	9e−7	22%
Tmall	R@20	0.0235	0.0233	0.0225	0.0203	0.0268	0.0222	0.0373	0.0314	0.0387	0.0473	**0.0528**	3e−5	11%
	N@20	0.0163	0.0160	0.0154	0.0139	0.0183	0.0152	0.0252	0.0213	0.0262	0.0328	**0.0361**	1e−4	10%
	R@40	0.0394	0.0350	0.0378	0.034	0.0446	0.0367	0.0616	0.0519	0.0645	0.0766	**0.0852**	1e−5	11%
	N@40	0.0218	0.0199	0.0208	0.0188	0.0246	0.0203	0.0337	0.0284	0.0352	0.0429	**0.0473**	7e−5	10%

표 23. LightGCL 모델 성능 비교

제 5 장. 개발된 알고리즘의 유용성 평가를 위한 프로토타입 개발

제 1절. 분할 그래프로부터 의미 구조 반영 패턴 발견 시스템의 프로토타입 구현

1. 돌연변이 예측을 위한 그래프 분할 및 임베딩

분자의 돌연변이성 예측은 치명적인 건강 영향을 미칠 수 있는 상황을 피하기 위해 필수적이다. 그러나 일반적으로 분자 화합물의 돌연변이를 예측하기 위해서는 많은 양의 실험이나 동물을 사용한 실험함으로써 시간적으로, 윤리적으로 문제가 되는 경우가 존재한다. 이를 해결하고 제안하는 의미 구조를 반영한 패턴 발견 시스템의 효과를 검증하고자 탄소 고리와 작용기로 존재하는 분자 그래프를 분할하고 분류하는 프로토타입을 개발하였다.

그림 42와 알고리즘 4는 각각 제안하는 전체 방법과 분할 알고리즘을 나타낸다. 의미 구조 반영을 위해 사용한 패턴 발견 시스템은 기존 분자 그래프의 작용기와 탄소 고리로 이루어진 사전 지식을 최대한으로 사용하기 위해 Betweenness 기반 분할 알고리즘을 사용한다. 제안하는 방법은 이를 통해 그래프를 작용기와 탄소 고리로 효율적으로 분할하고, 분할된 서브 그래프는 임베딩 되어 새로운 그래프 임베딩을 생성한다. 생성된 서브 그래프 임베딩은 통합되며 돌연변이성을 예측하는 데 사용된다.

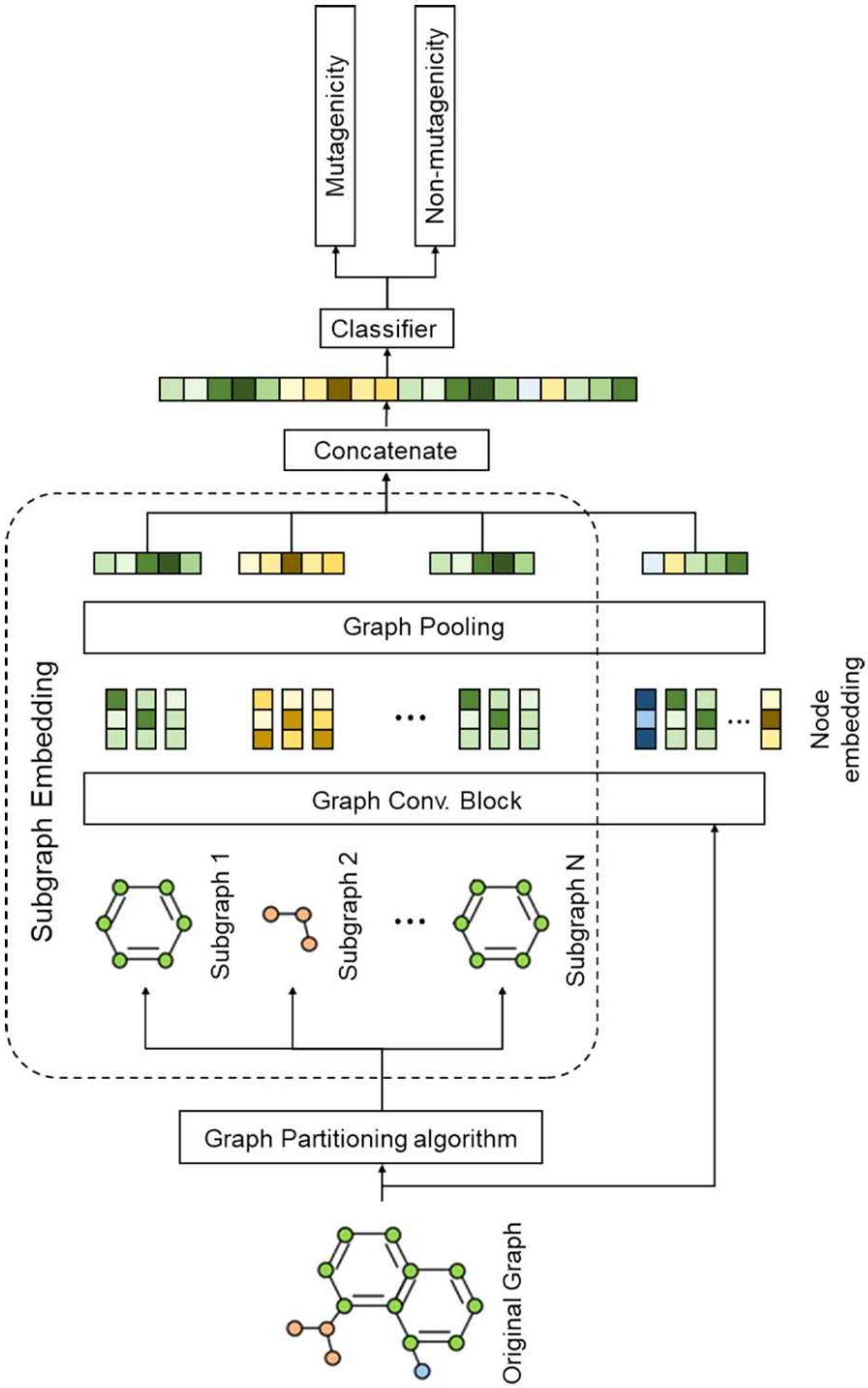


그림 43. 제안된 전체 방법

Input : *graph G , edge label* E_L

Output : *subgraph set S*

1 $e \leftarrow$ *edge in G*

2 $c_B(e) \leftarrow$ *edge betweenness centrality*

3 $V \leftarrow$ *the set of nodes in G*

4 $\sigma(s,t) \leftarrow$ *the number of shortest paths between s and t*

5 $\sigma(s,t|e) \leftarrow$ *the number the paths passing through edge e*

6 $C \leftarrow$ *the number of connected components of G*

7 $c_B(e) = \sum_{s,t \in V} \frac{\sigma(s,\ t|e)}{\sigma(s,t)}$

8 $N := C$

9 ***while*** $(N_o \leq C_o)$ ***do*** *// until G is partitioned*

10 *selected edge* $:=$ *the highest value of* $c_B(e)$

11 *remove selected edge in G*

12 $N :=$ *the number of connected components of G*

13 ***end***

14 $S :=$ *connected components of G*

15 ***return*** *S*

알고리즘 4. Girvan-Newman 그래프 분할 알고리즘

Method	Dataset		
	MUTAG	NCI-1	NCI-109
GCN	85.20±3.13	71.59±3.11	70.15±1.56
GIN	86.40±2.14	82.55±1.81	-
GAT	87.16±4.11	73.61±3.94	72.49±2.61
GraphSAGE	81.02±5.07	72.94±0.78	73.08±2.66
DGCNN	83.10±2.26	69.12±0.85	72.32±1.78
U2GNN	89.97±3.41	80.13±1.41	81.23±0.91
G-Resnet	90.64±2.84	79.94±1.84	80.11±2.34
G-inception	91.61±3.48	80.45±2.58	80.30±1.21
G-DensNet	91.47±2.46	81.17±0.78	80.72±2.15
Ours	**93.64±3.33**	**82.60±2.42**	**82.28±3.41**

표 24. 여러 방법에 대한 데이터셋 별 성능 비교

이 실험 결과는 제안된 방법이 모든 데이터셋에 대해 최신 방법보다 더 나은 성능을 달성한다는 것이다. MUTAG 데이터셋에 대해 약 2%p 성능 향상, 그리고 NCI1 및 NCI109 데이터셋에 대해 1%p 이상의 향상을 통해 분할하여 임베딩하고 통합하는 제안 그래프 분할 알고리즘이 임베딩 되면서 의미 구조를 반영한 패턴 발견 시스템의 구성 가능성을 보였다.

2. 멀웨어 분류를 위한 그래프 변환 및 임베딩 방법

사이버 보안에서 가장 중요하고 긴급한 문제 중 하나는 악성 소프트웨어 또는 멀웨어를 식별하고 분류하는 것이다. 이 분야의 주요 도전 과제 중 하나는 딥러닝 모델에 적합한 멀웨어 샘플의 표현을 찾는 것이다. 이는 멀웨어 샘플이 매우 가변적이며 딥러닝 분류기에서 포착하기 어려울 수 있는 미묘한 차이를 보일 수 있기 때문이다.

그림 44는 성능 저하의 주요 원인을 보여준다. 예를 들어, Vundo, Tracur, Obfuscator.ACY 및 Gatak과 같은 트로이 스타일 공격의 벡터 표현은 밀접하게 얽혀 있다. 그들의 서브루틴은 동일한 형태의 공격을 수행하기 때문에 유사하다.

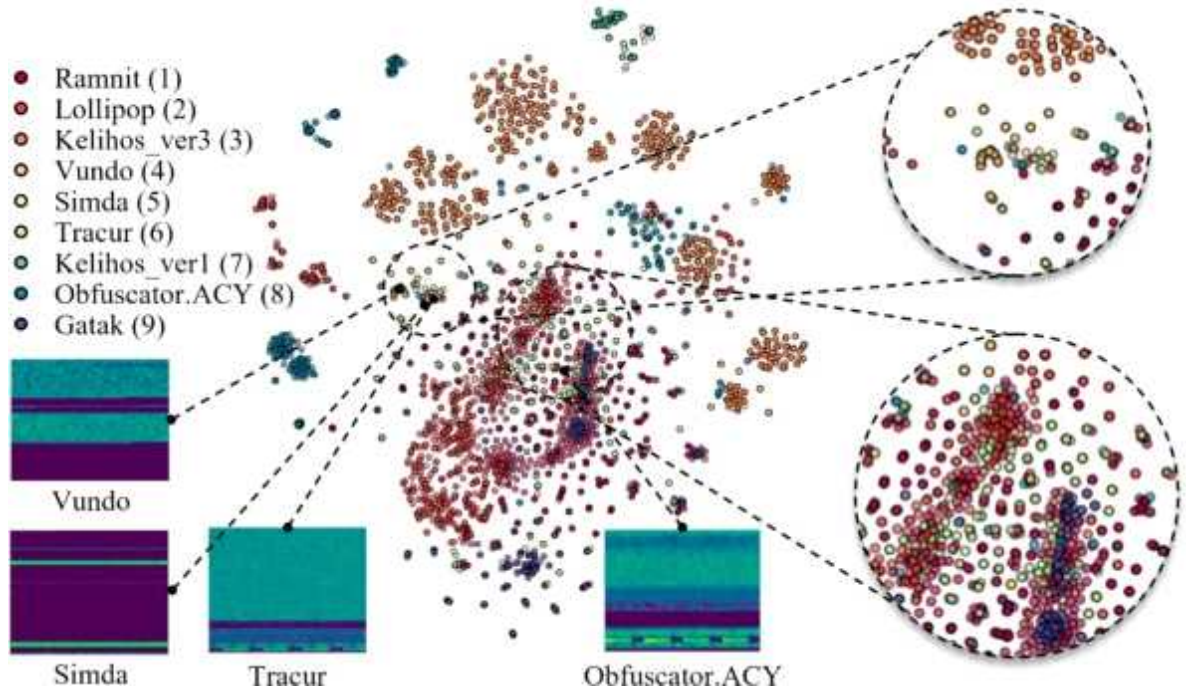


그림 44. 성능 저하의 주요 원인

높은 가변성과 유사성을 가진 멀웨어를 효과적으로 분류하는 도전을 해결하기 위해, 우리는 멀웨어 분류에서 분리된 표현 학습을 위한 진화적 Triplet 네트워크 기반의 새로운 방법을 제안한다.

본 논문에서는 API 호출, 루트킷 DLL 설치, 특정 가상메모리의 접근을 포함하는 기능적 특징을 학습하는 것으로 재현율을 개선하기 위해 제어 흐름 그래프를 명시적으로 샘플링하고 임베딩하는 방법을 제안한다.

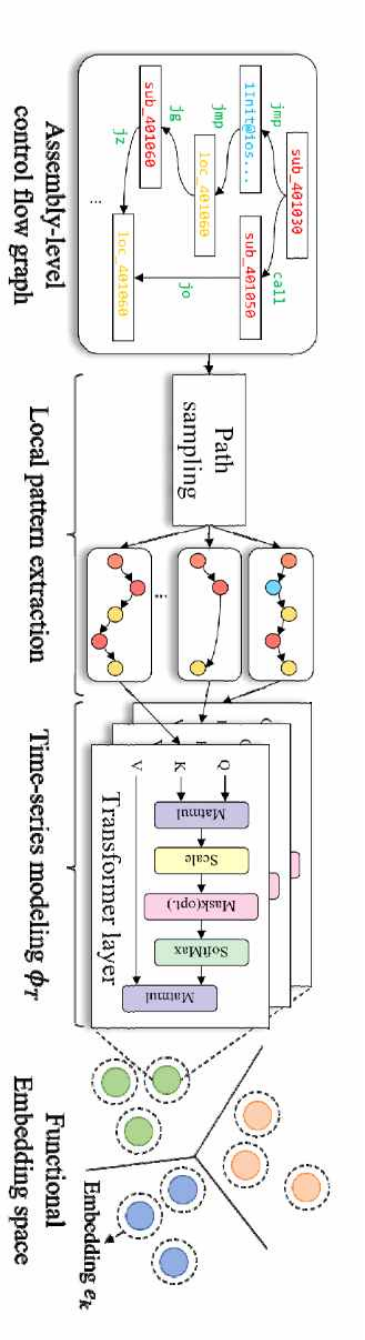


그림 45. 트랜스포머 기반의
그래프 임베딩 함수 이용한
악성코드 종류 분류 알고리즘

제어 흐름 그래프로부터 악성코드의 기능적 패턴을 모델링하기 위해 악성코드의 제어 흐름으로부터 공격 경로를 샘플링한 뒤 트랜스포머 기반의 그래프 임베딩 함수를 이용하여 악성코드 종류를 분류한다. 제안하는 방법을 입증하기 위해 실제 윈도우 악성코드로 구성된 마이크로소프트 챌린지 데이터셋을 사용하였다. 악성코드의 제어 흐름을 명시적으로 학습함으로써 최고의 재현율 97.89%를 확보하였고, 최신 및 가장 진보된 방법의 분류 정확도(97.89%)에 대비하여 크게 개선된 정확도(99.45%)를 달성하였다.

Method	Metrics	
	Acc.	Recall
Byte image-based approach		
Entropy CNN	0.987	0.9285
Byte image-CNN	0.9574	0.915
DCGAN-based Image Augmentation	0.9784	0.9448
Triplet loss-based CNN	0.9838	0.9381
Control flow graph-based approach		
Yolo-style CNN	0.9512	0.9007
Graph convolutional network	0.9825	0.9314
Path-sampling LSTM	0.9795	0.9384
Control flow graph-based prototype embedding		
Ours w/o prototype embedding	0.9882	0.9553
Ours	**0.9945**	**0.9789**

표 25. 여러 가지 방법에 대한 정확도와 재현율 비교

제 2 절. 개인화된 그래프 필터 기반 추천 알고리즘 모델 성능 평가

1. 그래프 대조 학습 기반 추천 시스템의 문제점

표 26에서 알 수 있듯이 대조 학습을 위한 뷰를 생성하는 방식으로 그래프 구조를 변경하거나, 임베딩에 노이즈를 주입하거나, 혹은 특잇값 분해를 통해 그래프를 재구성하는 등 여러 방법론이 제안됐다. 그러나 이러한 증강 방식을 기반으로 하는 협업 필터링 추천 시스템은 한계가 존재한다. 첫 번째 한계점은 그래프를 증강하는 것은 노이즈나 불필요한 정보를 만들게 되어 학습되는 노드 표현의 질을 떨어트린다는 것이다. 두 번째는 그래프 구조를 변형하는 것으로는 충분한 다양성이나 대조할 수 있는 노드 간 차이를 만들어 내지 못한다는 점이다. 마지막으로 세 번째는 이러한 대조 학습 기반 협업 필터링 추천 시스템들이 LightGCN과 같은 저주파 통과 필터만을 사용하는 모델을 기반으로 설계되었다는 점이다.

모델	뷰 생성 방식	저주파 통과 필터	고주파 통과 필터
SGL	노드/엣지 드롭아웃, 랜덤 워크	O	X
SimGCL	임베딩에 노이즈 주입	O	X
LightGCL	특잇값 분해를 통한 그래프 재구성	O	X
RDGCL	저주파, 고주파 통과 필터 적용	확산 방정식 (Diffusion eq.)	반응 방정식 (Reaction eq.)

표 26. RDGCL과 기존 대조 학습 기반 모델의 차이점 비교

2. 반응-확산 방정식을 사용하여 그래프 대조 학습 기반 추천 시스템 고도화

저주파 통과 필터만을 사용하는 모델의 경우, 과도한 평탄함 문제(Over-smoothing problem)로 인하여 노드 표현들이 서로 유사해져 구분되지 못하게 된다. RDGCL은 이러한 문제점을 해결하기 위해 반응-확산 방정식을 추천 시스템에 적용하였다.

반응-확산 방정식은 화학 반응과 물질의 확산을 모델링하는 방정식이며, 그래프 신경망에서 데이터가 노드 간의 연결을 통해 확산되는 방식과 유사하다. 이때 확산 과정은 저주파 통과 필터를 통해 진행되며, 반응 과정은 고주파 통과 필터를 통해 진행되면서 과도한 평탄함 문제를 처리할 수 있게 된다. RDGCL의 워크 플로우는 그림 46과 같다. 임베딩 E에 저주파 통과 필터를 이용한 연속 확산 과정을 적용하여 확산 행렬을 파생시킨 다음, 확산 행렬에 고주파 통과 필터를 이용한 연속 반응 과정을 적용하여 반응 행렬을 파생시킨다. 이 과정을 반복하면서 최종적으로 나오는 임베딩을 추천에 사용하며, 파생된 확산 행렬과 반응 행렬을 대조 학습에 사용한다.

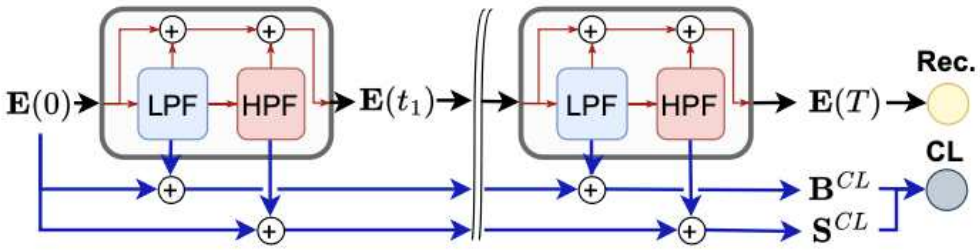


그림 46. RDGCL 아키텍처

확산 과정. 협업 필터링에서 자주 사용되는 확산 과정은 저주파 통과 필터에 해당하는 그래프 컨볼루션 필터를 사용한다. 많은 GNN이 이 확산 과정으로 일반화될 수 있다. RDGCL의 확산 과정의 공식은 다음과 같다:

$$B(t) = E(t) - \tilde{L}E(t) = E(t) + (\tilde{A} - I)E(t) = \tilde{A}E(t)$$

$B(t)$는 확산 행렬이며, $\tilde{A}$는 정규화된 사용자-아이템 상호 작용 행렬이다.

반응 과정. 반응 과정은 확산 과정의 반대 과정으로 고주파 통과 필터에 해당하는 라플라시안 행렬 (Laplacian matrix)를 사용한다. 반응 과정은 확산 과정 후에 나온 확산 행렬에 적용된다. 반응 과정의 공식은 다음과 같다.

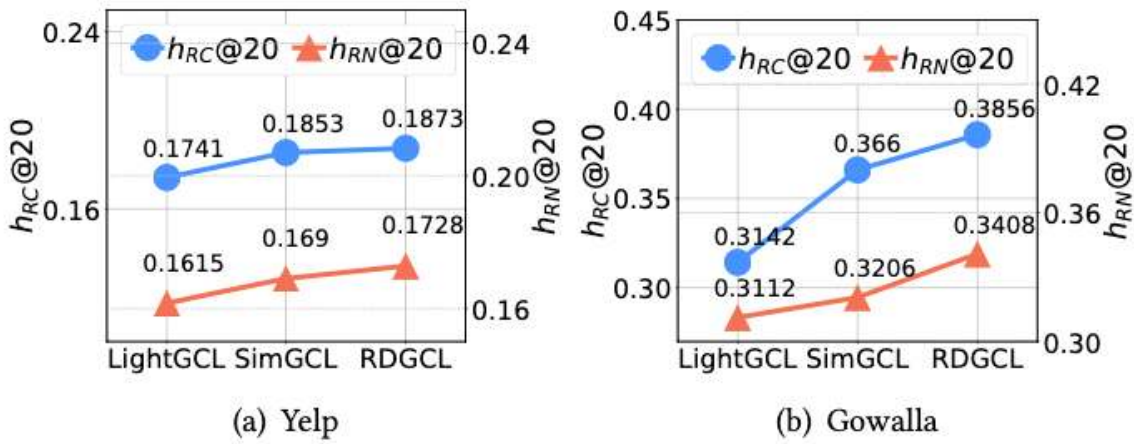
$$R(E(t)) := \tilde{L}B(t) = \tilde{L}(\tilde{A}E(t))$$

이 수식에서 $R(E(t))$는 반응 행렬이며, $\tilde{L}$은 정규화된 라플라시안 행렬이다.

3. 고도화된 그래프 대조 학습 기반 추천 시스템의 성능 평가

기존 모델과의 성능을 비교하기 위해 문헌에서 자주 사용되는 여섯 가지 벤치마크 데이터 세트인 Yelp, Gowalla, Amazon-Books, Amazon-Electronics, Amazon-CDs 및 Tmall을 사용한다. 순위 측정 기준으로 널리 사용되는 Recall@20, Recall@40, NDCG@20 및 NDCG@40을 사용한다. 사용자와 상호 작용이 없는 모든 아이템은 사용자의 추천 후보이다. 이전 연구와의 비교를 공정하게 유지하기 위해 동일한 데이터 세트와 동일한 학습/테스트 분할을 사용한다. 한 가지 경우를 제외한 모든 경우에서 RDGCL이 기존의 협업 필터링 알고리즘의 성능을 능가한다.

그림 47에서는 성능과 Coverage, Novelty 척도를 통해 추천 모델의 성능과 더불어 추천된 아이템의 다양성과 참신성을 평가한다. RDGCL은 확산-반응 과정을 통해 생성된 뷰를 대조 학습에 사용함으로써 성능 향상과 더불어 다양성과 참신성을 모두 높일 수 있다는 것을 Yelp와 Gowalla 데이터 세트를 통해 확인할 수 있다.



(a) Yelp (b) Gowalla

그림 47. LightGCL, SimGCL 그리고 RDGCL의 정확도 및 Coverage, Novelty 척도 비교

Data	Metric	Light GCN	LT-OCF	HMLET	SGL	SimGRACE	GCA	HCCF	SHT	SimGFL	XSimGCL	LightGCL	GF-CF	BSPM	RDGCL	Imp.
Yelp	Recall@20	0.0826	0.0947	0.0859	0.0967	0.0899	0.0779	0.0995	0.0853	0.1065	0.0974	0.1012	0.1043	0.1059	0.1099	3.19%
	NDCG@20	0.0690	0.0800	0.0699	0.0824	0.0775	0.0670	0.0842	0.0719	0.0912	0.0823	0.0870	0.0890	0.0913	0.0939	2.77%
	Recall@40	0.1346	0.1531	0.1388	0.1544	0.1443	0.1279	0.1578	0.1382	0.1688	0.1564	0.1591	0.1659	0.1677	0.1721	1.95%
	NDCG@40	0.0882	0.1016	0.0898	0.0906	0.1032	0.0851	0.1056	0.0913	0.1038	0.1040	0.1081	0.1115	0.1139	0.1165	2.28%
Gowalla	Recall@20	0.1294	0.2215	0.2157	0.2371	0.1519	0.1899	0.2222	0.1877	0.2304	0.2314	0.2351	0.2347	0.2455	0.2564	4.43%
	NDCG@20	0.0781	0.1287	0.1270	0.1400	0.0850	0.1100	0.1298	0.1119	0.1363	0.1380	0.1386	0.1382	0.1472	0.1549	5.19%
	Recall@40	0.1869	0.3131	0.3010	0.3269	0.2225	0.2690	0.3106	0.2671	0.3195	0.3224	0.3251	0.3266	0.3342	0.3460	3.53%
	NDCG@40	0.0932	0.1529	0.1494	0.1636	0.1034	0.1309	0.1528	0.1325	0.1597	0.1619	0.1622	0.1624	0.1707	0.1783	4.45%
Amazon-Books	Recall@20	0.0950	0.0987	0.0934	0.1333	0.0982	0.0850	0.1187	0.0834	0.1317	0.1148	0.1356	0.1392	0.1377	0.1416	1.74%
	NDCG@20	0.0704	0.0734	0.0697	0.1081	0.0769	0.0668	0.0912	0.063	0.1042	0.0887	0.1081	0.1158	0.1171	0.1141	-2.56%
	Recall@40	0.1454	0.1512	0.1417	0.1848	0.1429	0.1248	0.1728	0.1296	0.1867	0.1692	0.1917	0.1893	0.1830	0.1979	3.22%
	NDCG@40	0.0872	0.0910	0.0858	0.1249	0.0905	0.0799	0.1090	0.0782	0.1221	0.1065	0.1264	0.1321	0.1316	0.1324	0.23%
Amazon-Electronics	Recall@20	0.1251	0.1319	0.1258	0.1331	0.1236	0.1254	0.0597	0.0909	0.1371	0.1295	0.1306	0.1306	0.1311	0.1393	0.16%
	NDCG@20	0.0737	0.0792	0.0755	0.0783	0.0712	0.0727	0.0338	0.0912	0.0777	0.0748	0.0771	0.0771	0.0792	0.0809	2.15%
	Recall@40	0.1842	0.1850	0.1828	0.1913	0.1811	0.1306	0.0918	0.1409	0.1939	0.1833	0.1907	0.1907	0.1742	0.1982	2.22%
	NDCG@40	0.0896	0.0933	0.0908	0.0942	0.0868	0.0771	0.0426	0.0640	0.0934	0.0897	0.0934	0.0934	0.0912	0.0972	3.18%
Amazon-CDs	Recall@20	0.0956	0.1572	0.1438	0.1565	0.0729	0.0941	0.0658	0.1256	0.1583	0.1335	0.1460	0.1394	0.1443	0.1622	2.46%
	NDCG@20	0.0561	0.0981	0.0900	0.0997	0.0434	0.0581	0.0434	0.0756	0.1020	0.0804	0.0925	0.0912	0.0929	0.1054	3.33%
	Recall@40	0.1484	0.2196	0.2011	0.2159	0.1150	0.1346	0.1076	0.1829	0.2183	0.1952	0.2038	0.1940	0.1974	0.2225	1.92%
	NDCG@40	0.0708	0.1157	0.1061	0.1163	0.0552	0.0695	0.0542	0.0919	0.1188	0.0975	0.1086	0.1066	0.1085	0.1223	2.95%
Tmall	Recall@20	0.0717	0.0737	0.0676	0.1035	0.0705	0.0730	0.093	0.0706	0.0993	0.0880	0.1033	0.0879	0.0879	0.1040	0.44%
	NDCG@20	0.0498	0.0515	0.0464	0.0745	0.0488	0.0511	0.0663	0.0492	0.0712	0.0626	0.0626	0.0612	0.0612	0.0753	0.64%
	Recall@40	0.1126	0.1168	0.1059	0.1545	0.1104	0.1124	0.1427	0.1118	0.1505	0.1370	0.1370	0.1379	0.1380	0.1559	0.86%
	NDCG@40	0.0640	0.0665	0.0598	0.0922	0.0626	0.0648	0.0834	0.0634	0.0890	0.0795	0.0795	0.0784	0.0785	0.0933	0.85%

표 27. 모델 성능 비교

제 6 장. 결론 및 향후 연구

제 1 절. 결론

본 연구에서는 최근 지속해서 증가하는 그래프 관련 연구 동향을 정리하고 주어진 그래프 관련 문제에 적합한 그래프 모델링 기술을 사용할 수 있는 가이드라인을 제시한다. 또한, 가변적인 그래프 데이터의 특성의 한계를 극복하기 위해 그래프 내부의 특성은 유지하면서 고정된 크기의 임베딩 공간으로 전사할 수 있도록 입력 그래프를 서브 그래프로 분할하고 각각을 모델링 및 취합하여 임베딩 벡터를 추출하는 복합기술을 개발한다. 이를 통해 사용자 경험상황이 모두 담겨있는 그래프 내부에 존재하는 다양한 상세패턴들을 분석하여 서비스를 제공할 수 있다.

서브 그래프 분할 기법에서는 그래프 내부의 특성인 Degree, Closeness, Betweenness를 활용하여 그래프 시드를 설정하고 이를 기반으로 서브 그래프를 확장하는 알고리즘을 설계하였으며 내부의 정보 분석을 통해 그 성능을 입증하였다. 국소 정보별 그래프 임베딩 모델 구조화 기법에서는 서브 그래프별 내포하는 의미가 다른 점에 착안하여 가중치를 공유하지 않는 그래프 신경망을 도입하여 각 서브 그래프 그룹별로 적용하는 임베딩 구조화 기법을 개발하였다. 최종적으로 그래프 특성 기반 서브 그래프 분할 기법과 국소 정보별 그래프 임베딩 구조화 기법을 활용하여 얻어진 여러 그래프 임베딩들의 서로의 영향도를 고려하여 최종 그래프 인코딩으로 융합되었고 다양한 벤치마크 그래프 데이터셋을 이용하여 연구된 방법이 우수한 성능을 내는 것을 확인하였다.

본 연구의 라이프 그래프 패턴 기반 그래프 신경망 연구는 노드 분류를 위한 그래프 신경망의 한계들을 해결하는 것이 목표이다. 기존 그래프 신경망은 오버스무딩 문제와 헤테로필리 노드 문제에 어려움이 있기 때문에 그래프 신경망을 고도화하는 것은 필수적이다. 본 연구는 반응-확산 방정식을 기반으로 그래프 신경망을 제안하며 반응 항과 확산 항을 그래프 필터 개념으로 해석하여 설계했다. 이를 통해 그래프 신경망의 오버스무딩과 헤테로필리 문제를 해결하였으며 노드 분류 벤치마크 데이터셋에서 베이스라인들보다 우수한 성능을 보인다.

본 연구의 추천 시스템 연구는 라이프로그 데이터에서의 사용자 선호도와 관심사를 이해하고, 그에 따라 상품, 서비스, 정보 등을 추천하는 것이 목표이다. 기존의 그래프 기반 추천 알고리즘들은 저역 통과 필터의 역할을 하며 이는 사용자의 행동 패턴을 포착하는 데 한계가 있다. 저주파 통과 필터는 사용자가 많이 구매하였던 인기 있는 상품만을 추천하는 한계가 있다. 본 연구에서는

이를 극복하기 위해 개인화를 위한 그래프 필터 기반 추천 알고리즘을 고도화하고 개발했다. 이를 위해 고주파 통과를 적용하여 사용자들 간의 구분되는 패턴을 포착하여 사용자에게 개인화되고 참신한 추천 아이템을 추천할 수 있다. 고도화된 추천 알고리즘은 3개의 벤치마크 데이터셋에서 현존 최고 성능을 달성한다. 본 연구는 또한 기존 그래프 대조 학습 기반의 추천 알고리즘도 고도화하는 것을 목표로 하였다. 기존 그래프 대조 학습의 문제점들을 그래프 필터의 관점에서 해결하고자 하였다. 저주파와 고주파 통과 필터로부터 각각의 증강된 뷰를 생성하여 그래프 대조 학습의 새로운 설계를 개발하였다. 이 방법은 기존 그래프 대조 학습 기반의 추천 알고리즘보다 더 우수한 성능을 보이는 것을 확인하였다.

본 연구에서 개발한 고도화된 그래프 신경망과 추천 알고리즘은 라이프로그 데이터를 활용하는 데 있어서 다음과 같은 기대 효과가 있다.

가. 라이프로그 데이터를 활용하여 사용자의 선호도를 학습하고 개인화된 추천을 제공 가능

나. 노드 간의 관계를 분석하여 사람들 사이의 사회적 상호작용을 예측

다. 사용자의 행동 또는 활동 패턴을 식별하여, 그래프상의 노드로 표현된 개인의 일정이나 활동을 예측. 이는 건강 관리 앱에서 운동 또는 식습관 패턴을 분석하는데 사용 가능

제 2 절. 향후 연구

본 연구에서 개발한 라이프로그 상세 패턴 탐지를 위한 그래프 분할 기술에 대해, 향후 다루어야 할 이슈와 연구 방향은 다음과 같다.

(이슈 A) 추출된 상세 패턴의 해석과 의미에 대한 검증이 필요하다.
- 소셜 네트워크 데이터의 특성에 부합하는 상세 패턴의 의미를 분석하고 해석한다.
- 소셜 네트워크 기반의 다양한 척도에 대한 정당성을 검증한다.
- 특정 레이어에서 상세 패턴을 통합하는 모델링 접근의 타당성을 확인한다.

(이슈 B) 제안된 방법의 타당성을 다양한 도메인으로 확장하여 검증한다.
- 실제 라이프로그 데이터를 적용하고 검증해본다.
- 소셜 네트워크를 넘어서 바이오 도메인의 그래프 구조에도 적용하여 검증하고, 노드 속성 정보를 고려한 분할 기술을 적용한다.

(이슈 C) 사용자 행동 패턴 기반 학습 필터를 사용하여 다양한 상품 추천의 타당성을 검증한다:
- 라이프로그 데이터에 적합한 문제 정의와 검증을 수행한다.
- 개발된 기술들을 통합하는 시스템을 개발한다.

향후 연구에서는 추천 시스템의 데이터 희소성 문제와 네거티브 샘플링 방식의 문제에 초점을 맞출 예정이다. 많은 사용자들은 소수의 항목들과만 상호작용하며, 대다수의 항목들은 소수의 상호작용만을 가지는 희소한 데이터 특성을 갖는다. 이러한 희소 데이터 내 숨겨진 상호작용 패턴을 찾는 것은 어렵고, 노드 표현을 학습하는 것만으로는 부족하다. 이 문제를 해결하기 위해, 그래프의 증강된 뷰를 생성하고, 다른 뷰 간의 유사성과 차이를 강조하는 대조 학습을 활용할 계획이다. 기존의 대조 학습 기반 추천 시스템은 그래프를 무작위로 변경하여 증강 뷰를 생성하는데, 이 방법은 그래프 정보의 손실을 초래한다.

널리 사용되는 BPR 손실 함수는 관찰된 사용자 항목 쌍과 비교하면 관찰되지 않은 사용자 항목 쌍을 낮게 평가하는 경향이 있으며, 이는 아이템의 인기도를 반영하지 못하는 문제점을 가지고 있

다. 이에 대한 해결책으로, 대조 학습과 네거티브 샘플링이 공통된 핵심 요소를 공유하는 점에 착안하여, 두 방법을 통합한 샘플링 접근법을 개발할 것이다. 이 새로운 접근법은 확률론적 샘플링과 생성 모델을 기반으로 하며, 라이프로그 데이터에서 개인화된 추천 시스템을 위한 최첨단 기술을 확보하는 데 중요한 진전을 이룰 것으로 기대된다.

1. 이 보고서는 미래창조과학부(과학기술정보통신부 등)에서 시행한 한국전자통신연구원의 정보통신 • 방송 연구개발사업 위탁연구개발과제 (자율성장형 복합인공지능 원천기술연구, 23ZS1130)의 위탁연구개발과제 최종보고서입니다.

2. 이 연구개발내용을 대외적으로 발표할 때에는 반드시 한국전자통신연구원 정보통신 • 방송 연구개발사업 위탁연구개발과제 (자율성장형 복합인공지능 원천기술연구, 23ZS1130)의 위탁연구개발과제의 결과임을 밝혀야 합니다.

참고문헌

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv: 1609.02907, 2016.

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," arXiv preprint arXiv: 1810.00826, 2018.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv preprint arXiv: 1710.10903, 2017.

[4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Advances in Neural Information Processing Systems, vol. 30, 2017.

[5] P. Yanardag and S. Vishwanathan, "Deep graph kernels," Int. Conf. on Knowledge Discovery and Data Mining, pp. 1365-1374, 2015.

[6] N. Kriege and P. Mutzel, "Subgraph matching kernels for attributed graphs," arXiv preprint arXiv:1206.6483, 2012.

[7] C. Helma, R. D. King, S. Kramer, and A. Srinivasan, "The Predictive Toxicology Challenge 2000-2001," Bioinformatics, vol. 17, no. 1, pp. 107-108, 2001.

[8] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," Knowledge and Information Systems, vol. 14, pp. 347-375, 2008.

[9] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," Journal of Machine Learning Research, vol. 12, no. 9, 2011.

[10] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," Bioinformatics, vol. 21, pp. i47-i56, 2005.

[11] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," Journal of Molecular Biology, no. 4, vol. 330, pp. 771-783, 2003.

[12] T. N. Kipf, and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," In International Conference on Learning Representations (ICLR), 2017.

[13] Wu, Felix, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger.

"Simplifying graph convolutional networks." In *International conference on machine learning*, pp. 6861-6871. PMLR, 2019.

[14] Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks." In *International Conference on Learning Representations,* 2018.

[15] Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems* 30, 2017.

[16] Chen, Ming, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. "Simple and deep graph convolutional networks." In *International conference on machine learning*, pp. 1725-1735, PMLR, 2020.

[17] Chien, Eli, Jianhao Peng, Pan Li, and Olgica Milenkovic. "Adaptive Universal Generalized PageRank Graph Neural Network." In *International Conference on Learning Representations,* 2020.

[18] Xhonneux, Louis-Pascal, Meng Qu, and Jian Tang. "Continuous graph neural networks." In *International Conference on Machine Learning*, pp. 10432-10441. PMLR, 2020.

[19] Bo, Deyu, Xiao Wang, Chuan Shi, and Huawei Shen. "Beyond low-frequency information in graph convolutional networks." In *Proceedings of the AAAI Conference on Artificial Intelligence*, no. 5, vol. 35, pp. 3950-3957, 2021.

[20] Chamberlain, Ben, James Rowbottom, Maria I. Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. "Grand: Graph neural diffusion." In *International Conference on Machine Learning*, pp. 1407-1418, PMLR, 2021.

[21] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," Computer 42, no. 8: 30-37, 2009.

[22] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural Collaborative Filtering," In Proceedings of the 26th International Conference on World Wide Web, 173-182, 2017

[23] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural Graph Collaborative Filtering," In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, pp. 165-174, 2019.

[24] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," In Proceedings of the 43rd

International ACM SIGIR conference on research and development in Information Retrieval, pp. 639-648, 2020.

[25] J. Choi, J. Jeon, and N. Park, "LT-OCF: Learnable-Time ODE-based Collaborative Filtering," In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 251-260, 2021.

[26] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," Advances in Neural information Processing systems, 31, 2018.

[27] Y. Shen, Y. Wu, Y. Zhang, C. Shan, J. Zhang, K. B. Letaief, and D. Li, "How Powerful Is Graph Convolution for Recommendation?," In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1619-1629, 2021.

[28] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations. In International Conference on Learning Representations", 2020.

[29] Wu, Jiancan, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. "Self-supervised graph learning for recommendation." In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 726-735, 2021.

[30] Yu, Junliang, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. "Are graph augmentations necessary? simple graph contrastive learning for recommendation." In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pp. 1294-1303, 2022.

[31] Cai, Xuheng, Chao Huang, Lianghao Xia, and Xubin Ren. "LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation." In *The Eleventh International Conference on Learning Representations,* 2022.