2023학년도

석사학위논문

추천 시스템에서
그래프 콘볼루션 네트워크
최적화 방법

지도교수 : 김남기

경기대학교 대학원

컴퓨터과학과

이 상 민

추천 시스템에서
그래프 콘볼루션 네트워크
최적화 방법

이 논문을 석사학위논문으로 제출함

2023년 12월

경기대학교 대학원

컴퓨터과학과

이 상 민

이 상 민의 석사학위논문을 인준함

심 사 위 원 장 _____ 인

심 사 위 원 _____ 인

심 사 위 원 _____ 인

2023년 12월

경기대학교 대학원

목 차

표 목 차

그 림 목 차

감사의 글

경기대학교 김남기 교수님 연구실 소속 연구원으로 학부 연구생 2년, 석사 연구생 2년을 지내며 4년간 육체적, 정신적으로 힘든 시기도 있었지만, 당시를 다시 생각한다면 좋은 경험을 쌓았던 것 같습니다. 이러한 경험들 속에서 김남기 교수님에게 연구 관련된 것만이 아니라, 삶에 대한 귀감을 받았으며, 단단하고 유연한 사람이 되는 방법을 배웠습니다. 이병대 교수님에게 학부 연구생 시절 의료 이미지 프로젝트를 통해 연구 과정에 대한 많은 지도 편달을 수행하며 연구의 기초적인 지식을 많이 쌓을 수 있었습니다.

연구실 생활을 같이한 고준형, 최락현 선배와 동기인 채지훈, 장민호와 다른 연구실 소속인 전병욱, 문기렴, 한동현 후배도 감사한 마음을 전하고 싶습니다. 각자의 영역에서 다들 열심히 하는 모습을 보고 저 또한 열심히 할 수 있었습니다. 석사 연구생 동안 많은 귀감과 도움을 주었던 존경하는 김현빈, 최낙훈, 백지원 선배에게도 감사한 마음을 전하고 싶으며, 석사 1학년 시절 같은 연구실 소속 선배들이 모두 졸업하여 연구의 갈피를 잡지 못하는 기간 동안 선배들이 많은 경험적 도움을 주셔서 용기를 가지고 연구를 할 수 있었다고 생각하며 평생의 은인이라 생각하며 받은 은혜를 최선을 다해 갚을 수 있는 사람이 되도록 노력하겠습니다.

저는 다양한 학부 연구생 시절부터 다양한 외부 활동을 통해서 다양한 분야의 연구하시는 분들을 만날 수 있는 감사한 경험을 가졌습니다. 이러한 경험을 통해 다시 한번 범인이라는 것을 깨달았으며, 세상에는 엄청나게 훌륭하고 똑똑하신 분들이 많다는 것을 몸소 느꼈기에 석사 과정을 진행하며 연구 및 대외 활동에 더욱 정진할 수 있게 된 계기가 되었습니다. 처음으로 논문을 작성할 당시에는 훌륭하신 분들에 비해 많은 실수와 논리적인 과정의 연구를 수행하지 못하여서 많은 좌절과 절망을

느끼며, 석사 과정 동안 빛이 보이지 않는 끝없는 터널 속에 있다고 생각했습니다. 이를 극복하기 위한 역량을 키우기 위해서 석사 과정 중 꾸준히 인공지능 대회도 참여하였으며, 이러한 다양한 문제를 통해 경험이 수행 중인 연구에도 좋은 영향을 끼쳤으며, 분야에 상관없이 다양한 문제를 경험하는 것이 기계학습 엔지니어가 되기 위한 가장 큰 경험이라 생각합니다. 어떠한 문제의 해결을 위한 논리적 사고의 흐름을 계속해서 생각하며 몰입하여 정진하는 과정이 연구를 수행하기 위해서는 가장 중요하다는 것을 깨달았기에 참으로 많은 것을 얻은 석사 과정이었습니다.

졸업을 앞둔 현재, 끝없는 터널이라고 생각했던 길의 빛이 들어온다는 것을 느끼며, 석사 과정의 끝을 맺는다는 것에 감개무량합니다. 석사 과정을 통해 원하던 결과 이상의 결과를 얻게 되어 도움을 주신 모든 분에게 정말 감사히 생각하며, 석사 과정 동안 옆에서 많은 힘이 되어주고 응원해 준 혜빈이와 고난과 역경을 함께 분담해 주시고 기쁜 일에 함께 기뻐해 주신 부모님에게 항상 감사하다는 말씀을 전하고 싶으며, 모든 영광을 돌리고 싶습니다.

논 문 개 요

추천 시스템(Recommendation Systems)은 수많은 정보가 존재하는 인터넷에서 사용자가 원할 만한 콘텐츠를 빠르게 찾을 수 있도록 도움을 주며, 추천 시스템은 사용자 경험 향상과 플랫폼의 수익성을 증가시키는 중요한 역할을 하며, 이를 위해 사용자의 선호와 행동을 정확하게 분석하고 개인화된 제품이나 서비스를 제공한다. 이러한 추천 시스템을 효율적으로 구현하기 위해 그래프 콘볼루션 네트워크 (Graph Convolution Network, GCN) 알고리즘들이 많이 사용되고 있다. 이러한 알고리즘은 사용자 경험을 향상하고 플랫폼의 수익성을 증가시키는 중요한 역할을 하며, 이를 위해 사용자의 선호와 행동을 기존 방법론보다 정확하게 분석하고 개인화된 제품이나 서비스를 제공합니다. GCN은 그래프 구조를 활용하여 사용자와 아이템 간의 복잡한 상호 작용을 모형화하는데 탁월한 성능을 보여주며, 이를 통해 추천 시스템의 정확도와 효율성을 높일 수 있다. 그러나 GCN 기반 추천 시스템은 학습 과정에서 임베딩 값의 손실 문제와 깊은 Layer를 쌓지 못하는 문제에 직면하고 있으며, 이는 학습 속도와 정확도에 부정적인 영향을 미친다.

추천 시스템에서 학습 속도와 정확도는 상당히 중요하며, 추천의 품질과 사용자 경험에 직접적인 영향을 미친다. 학습 속도가 느리면 추천 시스템이 사용자의 최신 행동 패턴과 선호도 변화를 빠르게 반영하는 데 어려움을 겪을 수 있다. 이는 오래된 정보를 바탕으로 한 부정확한 추천으로 이어져 사용자 만족도를 저하할 수 있다. 또한 알고리즘의 추천 정확도가 낮으면 시스템이 사용자의 진정한 선호도를 제대로 이해하지 못하고, 이에 따라 사용자와의 관련성이 낮은 아이템을 추천할 가능성이 커진다. 따라서 본 논문은 그래프 콘볼루션 네트워크를 활용한 추천 시스템의 학습 속도와 정확도를 향상하기 위한 2가지 기법을 제안한다. 제

안하는 기법을 통해 추천 시스템의 학습 속도 및 정확도를 개선 하기 위해 본 연구에서는 Egress 초기화 기법과 가중치 전달 (Weighted Forwarding, WF) 기법을 제안한다. Egress 초기화 기법은 임베딩 값을 더 넓은 범위로 초기화하여 임베딩 값의 손실을 방지하고 학습 과정을 가속하며, 각 노드의 정보가 더 넓은 값 범위를 통해 전파하여 그래프 구조에서 더 다양한 정보를 캡처함으로써 추천 정확도를 향상했다. WF 기법은 임베딩 값에 가중치를 곱하여 다음 Layer로 전달함으로써 임베딩 값을 강화하여 GCN의 학습 속도와 추천 정확도를 개선하였다. 또한 제안하는 기법을 통한 추천 성능의 실험 결과를 분석하기 위해 임베딩 값들에 대한 통계적 분석, PCA, GKDE을 수행하였으며, 실험 결과 제안하는 2가지 기법을 통해 전반적인 학습 속도 및 정확도 개선을 보였다. 또한 기존의 알고리즘이 특정 데이터 집합에서 제대로 작동하지 않는 문제를 해결하였다. 이는 제안하는 기법들이 사용자와 아이템 간의 깊이 있는 관계의 모형화 결과를 기존 알고리즘에 더욱 효율적으로 활용할 수 있게 하고 학습 과정에서 발생하던 문제를 완화하여 정확도를 개선하였다.

제 1 장 서 론

제 1 절 연구 배경 및 목적

추천 시스템은 디지털 정보의 폭발적인 증가와 사용자의 변화된 컴퓨터 사용 습관, 개인화 추세, 그리고 인터넷 접근의 증가로 인한 사용자의 다양한 요구에 대응하기 위한 중요한 도구로서 필요성이 대두되었으며, 이러한 이유를 바탕으로 빠르게 발전했다. 추천 시스템의 발전에는 사용자의 과거 행동, 선호도 및 관심사 등의 데이터가 많은 영향을 미쳤다. 추천을 위한 사용자 데이터는 그림 1과 같은 방식으로 추천 시스템의 지속적인 발전을 가능하게 하였다. 이러한 방식으로 수집된 데이터를 분석하여 더 정확하게 개인화된 추천을 제공하여 사용자에게 더 적절하고 유용한 정보나 제품을 제시할 수 있기에 온라인 정보를 효율적으로 필터링하는 도구로 자리매김하게 되었으며, 개인화된 추천을 통해서 플랫폼에 대한 사용자 충성도와 수익성이 증가한다.

이러한 추천 시스템의 발전에 있어 데이터의 중요성 외로 기존의 기록된 데이터들을 최대한 활용하기 위해 학습 속도와 정확도 개선의 필요성이 대두되었으며, 학습 속도 개선은 대용량 데이터 처리, 실시간 추천 제공, 시스템 자원 최적화, 반복적 학습과 알고리즘 업데이트, 시장 경쟁력 유지, 그리고 개발 및 디버깅 시간 절약과 같은 여러 중요한 이유로 인해 중요하다. 현대의 추천 시스템은 방대한 양의 데이터를 처리하며, 이를 효율적으로 수행함으로써 더 높은 정확도로 추천을 제공할 수 있다. 빠른 학습 속도는 사용자의 최근 행동과 선호도를 신속하게 학습하고 반영함으로써, 실시간으로 더 정확한 추천을 제공할 수 있게 하며, 이는 플랫폼의 시장 경쟁력을 유지하고 사용자의 만족도를 높이는 데 기여한다. 또한, 학습 속도의 향상은 시스템 자원의 효율적인 활용을 가능하게 하여, 시스템 운영 비용을 줄이고 전반적인 성능을 향상할 수 있다. 지속해서 새로운 데이터를 학습하고 알고리즘을 업데이트하는 추천 시스템의 필요성은, 빠른 학습 속도가 반복적인 학습과 업데이트 프로세스를 더 빨리 수행할 수 있게 하여, 시스템이 더 높은 반응성과 정확도를 유지할 수 있게 한다. 마지막으로, 빠른 학습 속도는 개발자들이 알고리즘을 더 빨리

훈련하고, 디버깅하며, 개선할 수 있게 하며, 이는 더 나은 추천 시스템을 더 빨리 출시할 수 있게 한다. 이러한 이유로, 추천 시스템의 학습 속도를 개선하는 것은 사용자 경험의 향상, 시스템 성능의 최적화, 그리고 시장 경쟁력 유지와 같은 여러 이점을 제공하며, 추천 시스템의 전반적인 효율성과 성공에 큰 영향을 미친다.

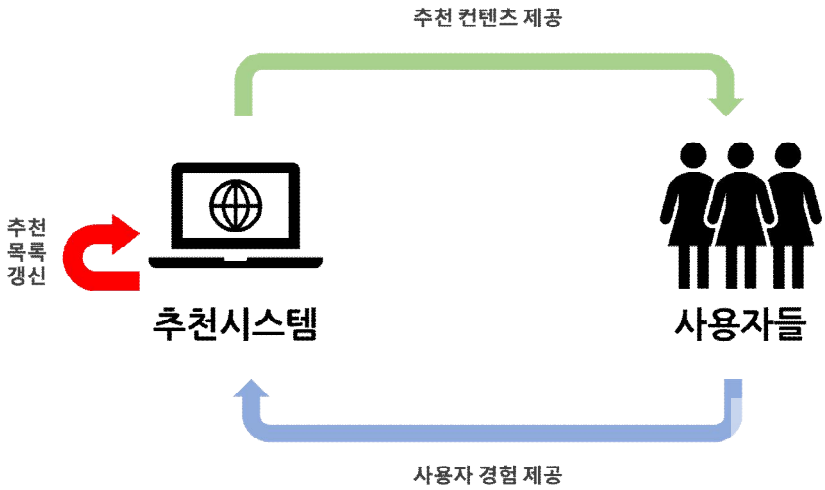
또한 추천 시스템의 정확도 개선은 사용자 만족도 향상, 플랫폼 수익성 증가, 시간 및 자원 절약, 개인화 서비스 제공, 데이터 활용 효율성 증대, 지속적인 플랫폼 개선, 그리고 경쟁 우위 확보와 같은 다양한 이유로 중요하다. 정확도가 높아질수록 사용자는 자신의 취향과 관심사에 더 잘 맞는 제품과 서비스를 제안받을 수 있어 사용자 만족도가 향상되고, 이는 장기적으로 플랫폼의 충성 고객 기반을 구축하는데 기여한다. 또한, 정확한 추천은 사용자가 관심을 가질만한 제품이나 서비스에 더 빨리 접근할 수 있게 돕는다, 이는 구매 확률을 높이고, 광고 수익을 증가시키며, 전반적으로 플랫폼의 수익성을 높일 수 있다. 사용자는 정확한 추천을 통해 원하는 것을 더 빨리 찾을 수 있으므로 시간과 자원을 절약할 수 있고, 이는 특히 대규모 온라인 쇼핑 플랫폼이나 스트리밍 서비스에서 중요하다. 개인화 서비스의 제공은 사용자 경험을 향상하며, 더 나은 데이터 활용은 플랫폼의 성능 평가와 개선에 도움을 준다. 높은 추천 정확도는 플랫폼에 경쟁 우위를 제공하며, 이는 시장에서 더 많은 사용자를 끌어들이고 장기적으로 플랫폼의 성공을 보장하는 데 기여한다. 이러한 이유로, 추천 시스템의 추천 정확도를 개선하는 것은 사용자와 플랫폼 모두에게 이익을 제공하며, 이는 추천 시스템의 발전 및 연구에서 중요한 목표로 간주한다.

이러한 다양한 이유로 추천 시스템의 학습 속도와 정확도 개선의 필요성이 대두되고 있으며, 이렇게 개선된 추천 시스템은 e-commerce, entertainment, social media 등 다양한 영역에서 활용될 수 있으며, 특히 YouTube, Netflix, Amazon과 같은 대형 플랫폼에서 사용자에게 더 나은 콘텐츠를 더 빠르게 제공하는 데 기여할 수 있다.

이러한 효율적인 최신 추천 시스템은 그래프 콘볼루션 네트워크 (Graph Convolutional Networks, GCN)이 해당 분야에서 두각을 나타내고 있는데, 이

는 주로 그래프 구조를 활용하여 사용자와 아이템 간의 복잡한 관계를 포착할 수 있는 능력 때문이다. GCN은 고차원 데이터 처리 능력으로 다양한 특성과 관계를 포착하며, 의미 임베딩을 통해 노드와 엣지의 의미 정보를 임베딩으로 캡처하여 더 나은 특성 표현을 학습하여 추천의 정확도 개선이 가능하다. 또한, 구조적 데이터를 활용하여 사용자와 아이템 간의 구조적 관계를 이해하여 이를 추천에 활용할 수 있다. 이러한 이점들은 여러 연구 및 적용 사례에서 GCN 기반 추천 시스템이 높은 성능을 보여주었으며, 이러한 점을 바탕으로 GCN의 중요성을 인식시키고 있다. GCN의 활용은 추천 시스템의 정확도와 효율성을 높이는 데 크게 기여하고 있다.

본 연구에서는 GCN 기반 추천 시스템의 학습 속도와 정확도를 개선에 중점을 두었다. 이러한 문제점을 극복하기 위해 가중치 전달 (Weight Forwarding, WF) 기법과 Egress 초기화 방법을 제안하였으며, 해당 방법은 GCN 기반 추천 시스템의 학습 속도와 정확도를 개선한다. 이러한 방법론은 추천 시스템의 학습 과정에서 embedding 값의 손실 현상을 해결하기 위한 새로운 학습 방식을 제시한다. WF 기법은 추천 시스템의 학습 과정을 가속하며, Egress 초기화 방법은 추천 시스템 알고리즘의 학습 과정을 개선하는 데 기여한다. 이러한 개선 사항을 통해 추천 시스템의 학습 속도 및 정확도를 개선한다.



<그림 1> 추천 시스템을 이용한 서비스 과정

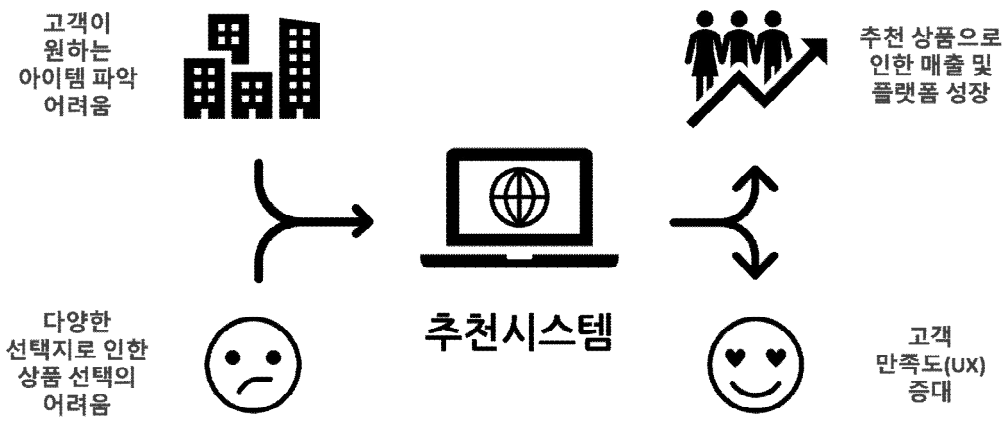
제 2 절 연구 필요성 및 기대 효과

추천 시스템은 다양한 온라인 플랫폼에서 그림 2와 같이 사용자의 선호와 행동을 분석해 고객이 원하는 개인화된 제품이나 서비스를 제시하여 매출 및 플랫폼을 성장시키기 위한 기술로, 최근 몇 년 동안 크게 주목받고 있다. 이러한 시스템은 기업에 많은 이점을 제공하며, 그중에서도 이익 증가는 가장 중요한 요인 중 하나이다. 또한 사용자에게도 추천 선택지를 통해 원하는 아이템 선택의 소비되는 시간을 줄여줌으로써 고객 만족도를 증진한다.

이러한 이점을 플랫폼에서 같이 위해서는 기업 측면에서는 효과적인 추천 시스템이 구축되어야 하며 이를 위해 다양한 방법론이 연구되었다. 그중 협업 필터링(Collaborative Filtering, CF) 방법론은 과거 사용자들이 수행한 아이템에 대한 평가를 기반으로 사용자들 간의 유사성 또는 아이템 간의 유사성을 계산하여 사용자에게 아이템 추천을 제공한다. 그런데 CF-기반 추천 시스템 알고리즘은 사용자의 아이템 구매 기록 등의 사용자-아이템 행동 데이터만을 이용하여 추천하기에 성능에 한계가 있다. 또한 사용자가 많아지면 사용자 간의 유사성을 파악하는 것에 있어 컴퓨터 계산량이 막대하게 증가하는 scalability 문제가 있다. 이러한 문제를 극복하기 위해서 GCN 알고리즘을 활용한 추천 시스템 알고리즘이 최근 활발히 연구되고 있다. 그러나 GCN 알고리즘은 여러 계층의 인접 노드 정보를 결합하면서 임베딩을 갱신하고, network가 인접한 노드의 정보를 계속해서 정규화하면서 전체적인 특징값이 작아지게 된다. 그 결과 GCN 알고리즘에서 존재하는 모든 노드는 초기부터 큰 값을 가진 임베딩 값으로 수렴하게 되어 network의 가장 큰 특징값를 가진 사용자와 상호 작용이 많은 아이템만 추천되게 되며, 상호 작용이 높지 못한 아이템의 정보가 일반화되어 사라지는 임베딩 값 소실 문제가 발생한다. 이러한 학습 과정 문제는 추천 목록의 다양성을 감소시키고 성능을 저하한다. [1-6]

추천 시스템 등 새로운 신경망으로 뜨고 있는 그래프 신경망은 깊은 Layer를 쌓지 못하고 있으며, 이러한 문제를 극복하기 위해서는 다양한 연구가 필요하다. 또한 GCN 알고리즘의 학습 과정에 사용되는 딥러닝 기법들이 그래프 구조에 최적화 되어있지 않은 방법들이 전통적으로 사용되고 있기에 그래프

구조에 맞는 기법들의 연구 필요성이 제기되고 있다. [7-9]



<그림 2> 추천 시스템을 통한 필요성 및 기대효과

제 3 절 연구 내용

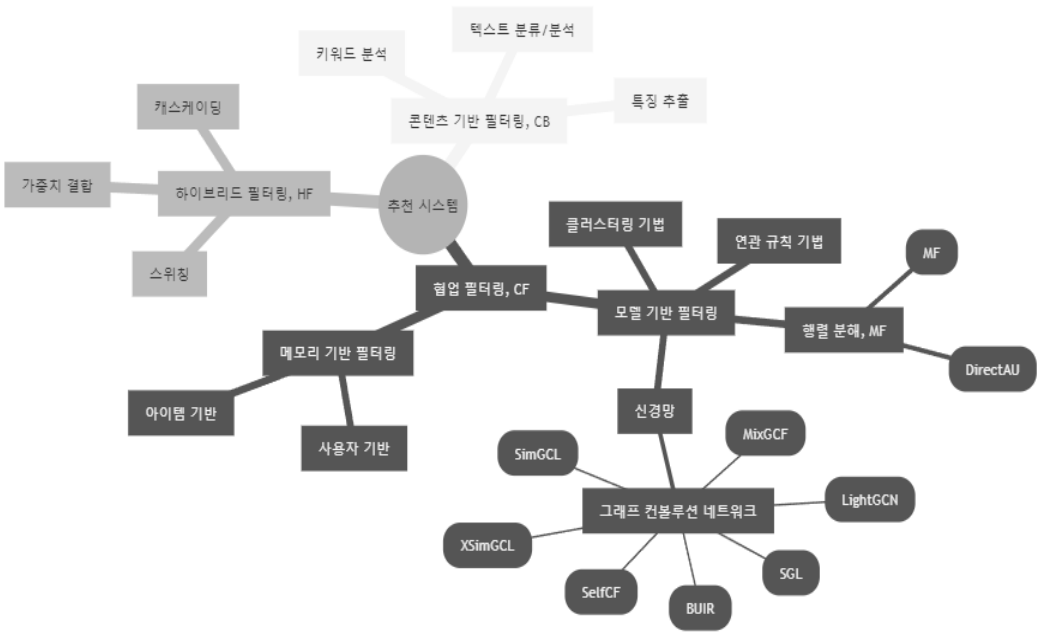
본 연구에서는 임베딩 값 소실 문제가 발생하는 GCN 알고리즘의 학습 과정을 개선하기 위한 2가지 방법을 제안한다. 우선 딥러닝의 기본적인 학습 과정은 초깃값을 반복해서 학습을 수행하며 이전에 사용된 값과 상당히 유사한 값을 다시 학습에 사용한다. 이러한 반복되는 학습 과정을 줄이기 위해서는 임베딩 값의 활용 방법에 대한 변화가 필요하다. 따라서 WF 기법은 GCN 알고리즘의 hop 계층 학습 과정 중 이전 Layer의 학습을 통해 나온 임베딩 값에 weight를 multiply 하여 다음 Layer에 전달한다. 해당 방법을 임베딩 값의 vanishing을 억제하고 적은 수의 epoch만으로도 더욱 많은 epoch를 학습하는 효과를 가져올 수 있다. 또한 딥러닝의 초깃값을 학습 과정에 상당히 많은 영향을 미친다. 기존의 GCN 알고리즘들에 저명하게 사용하던 가중치 초기화는 Xavier Uniform 초기화가 사용되고 있으며, 해당 방식은 딥러닝의 학습 과정 중 발생하는 여러 문제를 해결하기 위해 고안된 초기화 방법이다. 하지만 그래프 학습에는 더욱 최적의 초기화 방법을 사용하는 것이 학습 과정 개선을 보인다는 연구가 있었으며, 이에 따라 그래프의 임베딩 크기만을 고려한 Egress 초기화 기법을 제안한다.

제 2 장 관련 연구

제 1 절 추천 시스템 개요

추천 시스템은 크게 콘텐츠 기반 필터링(Content-Based Filtering, CB) 방법론, 하이브리드 필터링 방법론(Hybrid Filtering, HF), 협업 필터링(Collaborative Filtering, CF) 방법론으로 나누어진다. CB 방법론은 사용자가 이전에 선호했던 아이템의 내용(content)을 분석하여, 그와 유사한 아이템을 추천하는 방식이며, 해당 방법은 아이템의 특성을 분석하고, 사용자의 선호도 프로필을 구축하여 사용자에게 개인화된 추천을 제공한다. 해당 방법론의 장점은 새로운 아이템이 시스템에 추가되었을 때, 이를 사용자에게 추천할 수 있다. 하지만 시간이 지남에 따라 사용자의 선호도 변화를 해당 방법론은 과거의 선호도만을 기반으로 추천을 생성하므로, 이러한 변화를 캡처하지 못한다. 또한, 사용자 프로필 정보가 부족할 경우 사용자(콜드 스타트 문제)에 대한 추천을 생성하기 어렵다는 문제를 가지고 있다. HF 방법론은 여러 추천 기술을 결합하여 사용자에게 아이템을 추천하는 방식이며, 해당 방법론은 CB 방법론, CF 방법론 그리고 기타 추천 기술을 함께 사용하여 추천의 정확도를 향상하게 시키고 서로의 단점을 상쇄할 수 있으나, 해당 방법론은 구현 복잡성이 증가하고, 서로 다른 추천 알고리즘을 적절하게 결합하여야 한다. CF 방법론은 기존 사용자들의 데이터베이스를 바탕으로 신규 사용자와 비슷한 사용자가 좋아했던 상품을 신규 사용자도 좋아하리라는 것을 전제로 추천 콘텐츠 목록을 생성한다. CF 방법론은 크게 메모리 기반 필터링(Memory-Based Filtering) 방법론과 알고리즘 기반 필터링(Model-Based Filtering) 방법론으로 나누어지며, 메모리 기반 필터링 방법론은 CB 방법론과 유사하게 사용자들과 아이템 간의 평점이나 기타 상호 작용을 분석하여 분석된 데이터를 기반으로 추천을 제공하며, 구현이 비교적 간단하고, 실시간 추천을 제공할 수 있다. 하지만 데이터가 크고 희소할 경우, 유사도 계산이 부정확해질 수 있으며, 새로운 사용자나 아이템(콜드 스타트 문제)에 대한 추천을 생성하기 어렵다. 또한, 메모리 기반 필터링 방법론은 모든 데이터를 메모리에 유지해야 하므로, 큰

데이터 세트에 대해 스케일링이 어렵다. 알고리즘 기반 필터링 방법론은 사용자 또는 아이템을 유사한 그룹으로 클러스터링하여 추천을 생성하는 클러스터링 기법 방법론, 사용자의 구매 이력이나 아이템 간의 관계를 분석하여 추천을 생성하는 연관 규칙 기법 방법론, 사용자와 아이템 간의 상호 작용 행렬을 두 개의 저차원 행렬로 분해하여 사용자의 선호도와 아이템의 특성을 학습하는 행렬 분해 방법론, 딥 러닝 기술을 활용하여 사용자와 아이템 간의 복잡한 관계를 포착하고, 이를 기반으로 사용자에게 아이템을 추천하는 신경망 방법론으로 나누어진다. 신경망 방법론은 사용자의 아이템에 대한 평가 기록, 사용자의 세션 정보나 아이템의 카테고리 정보의 고차원적이고 비선형적인 관계 정보를 활용할 수 있어 복잡한 추천 시나리오에 유용하며, 딥러닝의 발전으로 인하여 신경망 기반 추천 시스템이 활발히 연구되고 있다. [10]



<그림 3> 추천 시스템 방법론 다이어그램

제 2 절 추천 시스템 방법론

협업 필터링의 방법론에 속하는 알고리즘 기반 필터링 방법론 중 행렬 분해 방법론과 신경망 방법론이 추천 시스템 분야에서 주로 연구되고 있으며, 사용자와 아이템 간의 복잡한 관계의 포착을 기반으로 한 효과적인 추천의 생성이 가능하기 때문이다. 행렬 분해 방법론은 사용자와 아이템을 저차원의 잠재 공간에 표현이 가능하기에 새로운 아이템 또는 사용자에 대한 추천도 생성할 수 있다. 신경망 방법론은 GPU와 분산 컴퓨팅 기술의 발전으로 대용량 데이터를 처리하는 데 더 효율적으로 되었으며, 다양한 신경망 구조를 통해 높은 표현력을 제공하며, 복잡한 상호 작용 패턴의 학습이 가능하다.

이러한 다양한 요인으로 인해 행렬 분해 방법론과 신경망 방법론이 주로 연구되고 있고, 실제 추천 시스템 애플리케이션에서도 성공적으로 적용되고 있으며, 다양한 산업 분야에서 비즈니스 가치를 창출하고 있다.

제 1 항 협업 필터링(Collaborative Filtering, CF)

협업 필터링(Collaborative Filtering, CF) 방법론이란, 타깃 사용자와 취향이 유사한 '이웃'이 선호하는 아이템을 타깃 사용자에게 추천해 주는 추천 방법을 의미한다. 기존 CF 방법론에서는 사용자-아이템 interaction 데이터를 매트릭스로 보고, 행렬 분해 방법론을 통해 원래 매트릭스를 복원하는 방식을 채택하였으며, 이러한 방법을 기반으로 하여 학습을 진행하면 아이템/사용자 간의 상호 작용이 많이 겹치는 사용자/아이템끼리 벡터가 유사해지며, 타깃 사용자와 취향이 비슷한 '이웃'이 상호작용했었으나, 타깃 사용자가 상호작용하지 않았던 아이템을 상위권 순위로 추천해 줄 수 있다. 하지만 Basic 한 CF 방법론과 MF 방법론은 사용자의 아이템에 대한 평가 기록, 사용자의 세션 정보나 아이템의 카테고리 정보와 같은 다양한 context 정보를 통합하여 활용하지 못하는 한계가 있다. 이러한 점을 극복하기 위해 최신 연구에서는 고차원적이고 비선형적인 상호 작용을 추천 시스템에 반영하여 효율적인 추천 시스템을 구축하기 위한 연구들이 활발히 진행되고 있다.

제 2 항 행렬 분해 (Matrix Factorization, MF)

행렬 분해(Matrix Factorization, MF) 방법론[11]은 추천 시스템에서 널리 사용되는 기법의 하나이다. 해당 기법은 사용자와 아이템 간의 상호 작용을 나타내는 행렬을 두 개의 저차원 행렬로 분해하는 방법을 기반으로 한다. 이 두 행렬은 각각 사용자와 아이템의 잠재 요인(latent factor)을 나타낸다. 해당 잠재 요인은 사용자의 선호나 아이템의 특성과 같은 숨겨진 특징을 반영한다. MF 방법론에서는 나와 비슷한 아이템들을 소비한 사용자의 Latent feature vector를 collaborative signal로 사용한다. MF 방법론에서의 학습 과정은 경사 하강법과 같은 최적화 알고리즘을 사용하여 원래의 행렬과 재구성된 행렬 간의 차이(오차)를 최소화한다. 두 개의 분해된 저차원 행렬을 곱하여 원래의 행렬을 근사하게 재구성한다. 이렇게 재구성된 행렬의 특정 항목은 해당 사용자가 해당 아이템에 대해 가질 것으로 예상되는 평점 또는 선호도를 나타낸다. MF 방법론의 잠재 요인을 통해 사용자와 아이템의 숨겨진 특징의 포착이 가능하므로, 새로운 아이템 또는 사용자에 대한 추천도 생성할 수 있다. 하지만, 해당 방법론은 대규모 데이터 집합에서 행렬 분해를 수행하는 것은 계산적으로 매우 복잡하다. 특히 사용자와 아이템의 수가 많을 경우, 행렬 분해 과정은 많은 시간과 컴퓨팅 자원이 필요하다. 또한 잠재 벡터의 크기가 추천 성능에 많은 영향을 끼치기 때문에 초기 설정이 매우 중요하다. 이러한 요인들로 인해 사용자의 선호나 아이템의 특성이 시간에 따라 변하는 동적인 변화를 반영하기 어렵다.

1) DirectAU

DirectAU[12]는 MF 인코더를 사용하여 DirectAU 손실을 최적화한다. 여기서 행렬 분해 인코더는 일반적으로 각 사용자와 항목을 기반으로 잠재 벡터에 대응하는 임베딩 테이블이다. 행렬 분해 인코더를 통해 L_{align}, $L_{uniform}$을 계산하며 여기서 L_{align}은 포지티브 관련 사용자 항목 쌍의 표현 간의 유사성을 측정하며, $L_{uniform}$손실은 표현이 하이퍼 스피어에 얼마나 잘 흩어져 있는지를 측정한다. 또한 $L_{uniform}$은 사용자와 아이템의 데이터 분포가 다양할 수 있기

에 별도로 계산하여 사용한다. L_{align}, $L_{uniform}$을 결합하여 DirectAU 손실로 사용하며, 이를 통해 사용자 및 아이템 쌍에 대한 등급 예측이 가능하고, 예측된 등급을 통해 사용자의 과거 행동을 활용하여 새롭거나 관찰되지 않은 항목에 대한 선호도를 예측하여 추천의 정확성과 관련성을 높일 수 있다. 또한 사용자의 선호도에 맞는 맞춤형 추천이 가능하다.

제 3 항 그래프 콘볼루션 네트워크

CF 방법론과 MF 방법론의 한계를 극복하고 사용자 경험 데이터를 활용하여 사용자-아이템의 상호 작용을 더욱 정확하게 포착하려는 그래프 신경망 방법론이 최근 주목받고 있다. 추천 시스템에서 주로 사용되는 딥러닝 알고리즘으로는 Graph Neural Network(GNN) 알고리즘[13]에 속하는 Graph Convolution Network(GCN)가 있다.

이러한 GNN과 GCN은 사용자-아이템 매트릭스를 이분 그래프와 무방향 그래프(undirected graph)로 표현하여 사용자와 아이템 간의 짧고 긴 거리 상호 작용을 포착하며, 이를 통해 기존 추천 방법론보다 더 정확한 선호 패턴을 학습이 가능하다. 특히 GNN 알고리즘의 특정한 형태인 GCN 알고리즘이 추천 시스템에서 가장 널리 사용되는데 이는 GCN 알고리즘이 추천 시스템을 구현하기에 매우 적합한 구조를 가지고 있기 때문이다. GCN 알고리즘은 그래프의 노드와 그 주변 정보를 convolution 연산을 통해 간결하게 집계하는 방법을 제공한다. 해당 방식은 GCN 알고리즘의 학습과 최적화를 용이하게 한다. GCN 알고리즘은 그래프 구조의 data를 처리하는 데 특화되어 있어, 사용자와 아이템 간의 복잡한 interaction을 잘 반영할 수 있다. 이러한 interaction을 반영하는 부분에 있어 GCN 알고리즘은 사용자와 아이템 간의 관계에 부가 정보(예를 들면 사용자의 성별, 연령, 아이템의 카테고리 등)를 통합하여 포착이 가능하기에, 기존 MF 방법론이나 CF 방법론에서 사용자와 아이템의 복잡한 interaction을 완전히 반영하기 어려웠던 한계점을 극복할 수 있다.

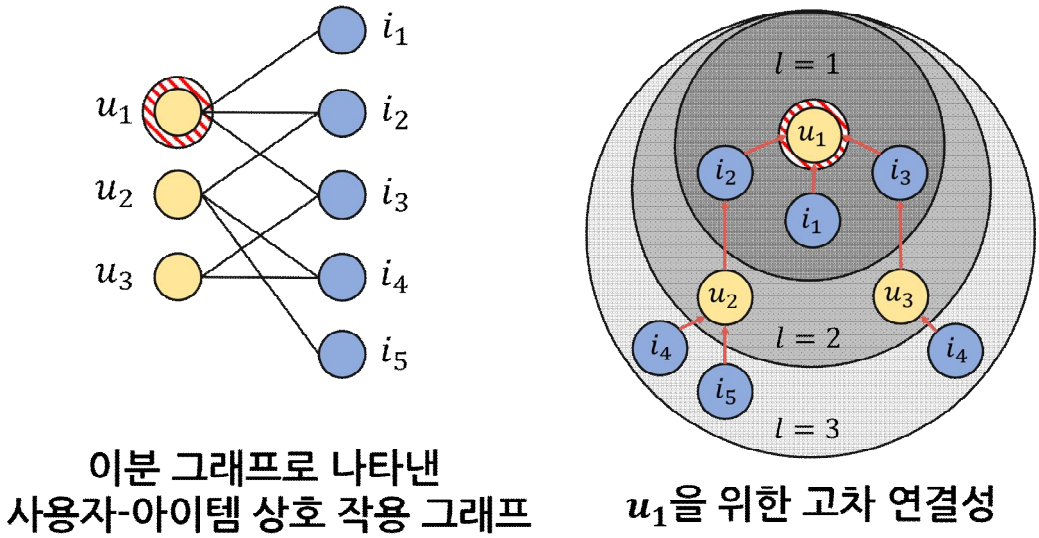
GCN 기반 추천 시스템 알고리즘은 노드와 그 주변의 정보를 집계하여 고차원의 임베딩을 생성하여 사용자와 아이템의 복잡한 feature를 반영할 수 있다. 그 과정을 단계별로 살펴보면 GCN-기반 추천 시스템 알고리즘은 먼저 사용자와 아이템 간의 interaction은 그래프로 표현하여 사용자와 아이템 간의 discrete 관계를 효과적으로 표현한다. 그다음 도출된 사용자-아이템의 협업 신호 포착의 결과를 GCN 알고리즘을 활용하여 학습하고 최종적으로 사용자와 아이템에 대한 임베딩 값을 결정한다. 추천 시스템 알고리즘은 이렇게 결정된 사용자 임베딩과 아이템 임베딩을 이용하여 사용자-아이템 간 선호도 점수를 계산하고, 그 점수를 바탕으로 콘텐츠들을 효과적으로 추천한다. GCN

알고리즘은 새로운 사용자나 아이템의 정보를 손쉽게 그래프 구조로 표현 가능하기에 Basic 한 CF 방법론이나 MF 방법론에 비해 Cold Start problem를 완화할 수 있다. 또 GCN 알고리즘은 다양한 그래프 구조와 알고리즘에 대한 연구가 활발히 진행되고 있어 다양한 추천 시나리오에 이를 적용할 수 있는 유연성도 가지고 있다. [14-16]

그래프 신경망 기반의 추천 시스템은 NGCF [17]을 기반으로 하여 개선된 LightGCN [18]이 제안 되었으며, 해당 알고리즘을 백본 네트워크로 사용한 BUIR [19], SelfCF [20], MixGCF [21], SGL [22], SimGCL [23], XSimGCL[24]이 있다.

1) NGCF

Neural Graph Collaborative Filtering(NGCF) [19]는 이분 그래프(bipartite graph, 그림 4. 왼)로 변환된 그래프에 사용자-아이템 interaction data를 통한 high-order connectivity를 반영하기 위한 방법으로 GCN을 활용하여 '협업 신호(Collaborative Signal)'을 explicit 하게 인코딩하는 방법을 제안했다. 여기서 high-order connectivity(그림 4. 오)는 이분 그래프로 나타낼 수 없는 고차 연결을 의미하며, 고차 연결이란 경로 길이가 1보다 큰 모든 노드에서 u_1에 도달하는 경로를 말한다. 이러한 고차 연결에는 협업 신호를 전달하는 풍부한 의미가 포함되어 있다. 그림 4. 오른쪽과 같이 예를 들어 $u_1 \leftarrow i_2 \leftarrow u_2$ 경로는 두 사용자 모두 i_2와 상호 작용했기 때문에 u_1과 u_2 사이의 동작 유사성을 나타낸다. $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$가 더 긴 경로를 보면 비슷한 사용자인 u_2가 이전에 i_4를 사용한 적이 있기 때문에 u_1이 i_4를 채택할 가능성이 높다는 것을 알 수 있다. i_4의 연결된 경로는 두 개이고 i_5의 연결된 경로는 하나이기에, u_1을 기준으로 $l=3, < i_4, u_1 > < i_5, u_1 > = 3$이라는 전체론적 관점에서 볼 때 i_4는 항목 i_5보다 u_1이 관심을 가질 가능성이 더 높다. 이러한 고차 연결은 사용자와 항목 간의 협업 신호를 포착하여 보다 정확한 추천을 제공하는 데 사용 가능 하 다 .



<그림 4> 이분 그래프, 그래프 고차 연결성

이러한 고차 연결 정보를 바탕으로 하는 협업 신호를 그래프 신경망에 활용하기 위한 알고리즘을 제안하였으며, 협업 신호는 타깃 사용자/아이템과 상호작용 명세가 겹치는 '이웃 사용자/아이템'의 임베딩 벡터, 내지는 그 '이웃들과 상호작용했었으나, 타깃과는 상호작용하지 않았던 아이템/사용자'의 임베딩 벡터라고 할 수 있다.

행렬 분해(MF) 방법론에서는 이러한 협업 신호를 implicit 하게 반영해 줬다면 NGCF에서는 이것을 explicit 하게 반영하였으며, 제안된 알고리즘을 사용하여 이전 방법론과 대비하여 추천 성능의 향상을 보였다. 요약하자면, 본 연구를 통해 기존 행렬로 이루어진 데이터를 통해 bipartite 그래프로 변환 후 고차 연결성의 협업 신호를 인코딩하는 GCN 기반 CF의 알고리즘을 최초로 제안하였다.

2) LightGCN

CF 기반 알고리즘인 NGCF를 단순화한 LightGCN [20]은 GCN의 propagation Layer에서 feature transformation, non-linear activation, self-connection을 제거하여 복잡한 가중치 학습을 단순화하였다. LightGCN은

노드 간의 interaction을 강화하기 위해 사용자와 아이템 간의 interaction 그래프를 이용한다. 그 결과 사용자와 아이템 간의 연결 정보만 사용하여 노드의 임베딩을 생성하여 추천 성능을 향상한다. LightGCN은 GCN 알고리즘의 가중치 학습 대신 노드 간의 interaction 그래프를 이용하여 노드 임베딩을 생성하므로, 학습 및 추론 시간을 대폭 줄일 수 있다. LightGCN는 구조가 간단하고 빨라서 다양한 추천 시나리오에서 적은 학습으로도 높은 성능을 보여 대규모 추천 시스템에서 유용하게 사용될 수 있다.

3) BUIR

단일 클래스 협업 필터링 (OCCF) 은 암시적인 피드백을 기반으로 사용자가 선호하는 항목을 정확하게 추론하기 위해 광범위하게 연구되고 있다. BUIR [21]은 단일 클래스 협업 필터링 (OCCF)을 위한 새로운 프레임워크이며, BPR 및 그래프 기반 LightGCN을 백본 네트워크로 사용한다. 또한 아직 상호 작용하지 않은 긍정적으로 관련된 사용자 항목 쌍을 식별하는 것을 목표한다. 이를 위해 BUIR은 서로 학습하는 두 개의 별개의 인코더 네트워크를 사용한다. 첫 번째 인코더(Online Encoder)는 두 번째 인코더(Target Encoder)의 출력을 목표로 예측하도록 온라인 인코더를 훈련함으로써 알고리즘은 사용자와 항목 간의 잠재 요인과 유사성을 포착한다. 이를 통해 긍정적으로 관련된 사용자와 항목을 서로 비슷하게 표현하도록 훈련되며, 두 번째 인코더는 첫 번째 인코더의 근사치를 계산한다. 이러한 Target 인코더는 계기 업데이트에 따라 업데이트되어 출력 일관성이 서서히 개선되며, 두 번째 인코더가 첫 번째 인코더를 점진적으로 근사화하면 시간이 지남에 따라 표현 품질이 향상된다. 이러한 상호 유리한 방식으로 인코더를 학습함으로써 사용자와 항목 간의 잠재적 요인과 유사성을 포착하여 보다 정확하고 차별적인 표현을 가능하게 한다. 이러한 두 개의 인코더를 활용함으로써 “양성이지만 관찰되지 않는” 쌍을 음수로 선택할 수 있는 일반적인 네거티브 샘플링이 필요하지 않다. 이러한 이점을 바탕으로 BUIR은 인코더 입력에 확률적 데이터 증강을 적용하여 OCCF의 데이터 희소성 문제를 해결한다.

확률적 데이터 증강은 기존 데이터에서 새 샘플을 생성하여 훈련 데이터의

양을 늘리도록 활용하였으며, 사용자 및 항목의 주변 정보를 사용하여 각 긍정적인 상호 작용에 대한 증강 뷰를 생성한다. 여기서 말하는 주변 정보란 일부 유사성 지표를 기반으로 특정 사용자/항목과 유사한 사용자/항목 집합을 말한다. 알고리즘이 긍정적인 상호 작용을 인코딩할 때마다 증강 뷰가 무작위로 생성되며, 즉, 알고리즘은 훈련될 때마다 상호 작용을 다르게 보므로 과적합을 줄이고 일반화를 개선한다. 이러한 확률적 데이터 증강 기반의 자체 감독 학습을 통해 추가 레이블 없이 증강 뷰를 기반으로 알고리즘을 학습하며, 긍정적인 사용자 항목 상호 작용이 극히 일부만 관찰되는 시나리오에서 데이터 희소성 문제를 극복한다.

4) SelfCF

SelfCF [22]는 긍정적으로 관찰된 상호 작용만을 기반으로 사용자와 항목의 잠재 표현을 학습하는 자체 지도 협업 필터링 프레임워크이다. SelfCF의 학습 목표는 BUIR 프레임워크와 마찬가지로 부정적인 표본을 사용하지 않고도 사용자와 항목에 대한 정보를 제공하는 표현을 학습하는 것이며, 이를 위해 BUIR 프레임워크와 동일하게 자체 지도 학습 (SSL) 접근 방식을 채택하여 음성 샘플이 없는 알고리즘을 학습한다. 하지만 SelfCF 프레임워크는 사용자/항목 ID의 원시 입력을 사용하는 대신 Siamese networks를 기본 구조로 사용하여 기본 구조에서 생성되는 잠재 임베딩을 보강합니다. 해당 접근 방식을 통해 샴 네트워크를 단순화하고 기존의 딥 러닝 기반 CF 알고리즘에 쉽게 적용할 수 있다.

또한 BUIR은 긍정적인 상호 작용만을 기반으로 사용자와 항목의 표현을 학습하므로 알고리즘의 정확도가 제한될 수 있으며, Online Encoder와 Target Encoder의 매개변수를 구분하여 서로 다른 관점을 제시하지만, 이 수정으로 인해 그래프 기반 CF 알고리즘의 기본 논리에 어긋날 수 있기에 SelfCF 프레임워크는 Encoder의 출력을 증가시켜 표현 학습을 위해 서로 다르지만, 관련이 있는 두 개의 임베딩을 생성한다. 이러한 점을 통해 Online Encoder와 Target Encoder간에 동일한 Encoder를 공유하므로 추가 Encoder에 필요한 메모리와 계산을 위한 자원이 줄어든다.

또한 SelfCF 프레임워크에서는 output perturbation을 통해 동일한 사용자 항목 임베딩의 역사 임베딩, 임베딩 드롭아웃, 에지 프루닝이라는 세 가지 출력 증강 기법을 고안하였다. 해당 3가지 기법은 프레임워크의 주요 구성 요소인 백본 네트워크의 출력을 왜곡시켜 대조적 임베딩을 생성하기 위함이며, 과거 임베딩 섭동은 이전 학습에 사용하였던 임베딩을 활용하여 현재의 임베딩을 교란한 후 결합하여 사용하고, 임베딩 드롭아웃은 현재 임베딩의 노이즈를 적용하여 교란 시킨 후 다음 임베딩으로 사용하고, 에지 프루닝 방법은 현재 임베딩에 콘볼루션 Layer를 하나 더 쌓아서 생성되는 다음 임베딩에 적용됩니다. 이러한 output perturbation은 네거티브 샘플 없이도 사용자와 항목에 대한 정보를 제공하는 표현을 학습하는 데 도움이 되며 다양한 CF 기반 알고리즘에 적용할 수 있다. 해당 프레임워크는 주어진 사용자에게 교란된 항목의 예측을 극대화하고 그 반대의 경우도 마찬가지이므로 추천 정확도가 향상되고 학습 속도가 빨라진다.

5) MixGCF

GNN 기반 CF의 네거티브 샘플링 방법을 개선하기 위해 MixGCF [23] 학습 기법은 그래프 신경망 기반 추천 시스템 훈련을 위한 새로운 네거티브 샘플링 전략을 제안하였다. MixGCF의 주요한 제안 방법은 GNN 기반 추천 시스템을 훈련하는 데 직접 사용할 수 있는 일반적인 네거티브 샘플링이며, 학습 데이터에서 원시 네거티브를 직접 샘플링하는 대신 포지티브 믹싱과 홉 믹싱 기법을 사용하여 하드 네거티브를 합성한다.

포지티브 믹싱은 사용자 항목 그래프 구조의 포지티브 정보를 활용하여 후보 네거티브 항목의 임베딩을 향상하며, 풀링 연산을 사용하여 양수 항목의 임베딩을 후보 음수 항목의 임베딩과 결합한다. 해당 방법은 인접한 포지티브 아이템의 정보를 통합하여 부정적인 아이템의 표현을 풍부하게 만든다. 홉 믹싱은 합성 네거티브 아이템을 통해 임베딩을 생성하기 위해 사용하며, 각 Layer에 대해 후보 네거티브 항목의 임베딩 세트에서 후보 네거티브 임베딩을 샘플링하며, 각 Layer에서 선택한 임베딩을 풀링 연산을 사용하여 결합하여 합성 네거티브 항목의 표현을 합성한다. MixGCF의 실험을 위해 NGCF의

연결 기반 풀링, LightGCN의 합계 기반 풀링을 활용하여 실험을 진행하였다.

6) SGL

SGL[24]은 LightGCN 알고리즘에 self-supervised 학습 알고리즘을 적용한 것이다. SGL은 그래프에 있는 다른 노드들과의 관계를 활용하여 그래프의 재구성을 위해 Structure Reconstruction (SR)을 활용한다. SGL은 SR을 통해 그래프의 노드 혹은 엣지를 무작위로 제거하고, 제거된 노드 또는 엣지를 재구성하여 학습 데이터를 증강한다. 여기서 증강된 data는 별도의 labeling 없이도 self-supervised 학습을 이용한 그래프 학습이 가능하다. 따라서 SGL은 labeling 되어 있지 않은 그래프를 통해서도 학습을 수행할 수 있는 장점을 가진다.

7) SimGCL

Dropout 기반 그래프 증강을 수행하여 대조적인 뷰를 사용하는 SGL은 학습하는데 많은 자원과 시간이 필요하다. 이러한 제약을 없애고 작은 자원과 시간으로도 효율적인 랜덤 노이즈 기반의 그래프 데이터 증강을 활용하는 SimGCL[25]이 제안되었다. SimGCL은 그래프 콘볼루션 네트워크의 학습 과정에서 학습 데이터의 노이즈를 일정하게 증가시켜 운동량처럼 작용하여 수렴 속도를 높이며, contrastive 학습을 통해 보다 정확한 학습을 수행한다. SimGCL은 대조적인 손실 함수를 사용하여 유사한 samples를 grouping하고, grouping 된 samples를 구분하기 위해 contrastive 학습을 수행한다. 그 결과 SimGCL은 LightGCN을 백본 네트워크로 사용하고, 균일한 랜덤 노이즈가 있는 그래프 증강을 활용하여 contrastive 학습만을 적용하여 정확도와 다양성을 개선할 수 있었다.

8) XSimGCL

XSimGCL [26]은 SimGCL의 확장이며 더 간단한 아키텍처를 제안하였다. SimGCL은 입력 노드 임베딩을 업데이트하기 위해 세 번의 정방향/역방향 패스가 필요하며, 동일한 노드의 한 쌍의 뷰 사이의 상호 정보는 항상 매우 높을 수 있으며, 이에 따라 대조 효과가 떨어질 수 있다. 이 문제를 해결하기 위

해 XSimGCL은 몇 가지 공통 정보를 공유하지만, 집계된 이웃과 추가된 노이즈가 서로 다른 대조적인 서로 다른 Layer 임베딩을 사용한다. 또한 인코딩 프로세스를 병합하여 SimGCL의 아키텍처를 간소화하였으며, 미니 배치 계산에서 순방향/역방향 패스를 한 번만 수행하는 아키텍처를 만들었다. 이를 통해 XSimGCL은 noise를 통한 그래프 데이터 증강 효과와 효율성 측면에서 SimGCL을 능가하여 추천 정확도와 학습 효율성이 더 뛰어나다.

이러한 GCN 기반 추천 시스템 알고리즘은 다양한 콘텐츠 추천 서비스에서 좋은 성능을 보였다. 하지만 이러한 GCN 알고리즘들은 학습 과정에서 임베딩 값의 값이 상당히 작은 값으로 수렴되는 임베딩 값 소실 문제를 가지고 있다. 따라서 본 논문에서는 propagation rule에 진행되기 전 임베딩 값에 가중치를 곱한 뒤 다음 Layer로 forwarding 하여 임베딩 값 소실 문제를 완화하여 학습 속도 및 정확도를 개선할 수 있는 WF 기법과 기존보다 넓은 초기화 범위를 통해 기존 그래프 구조에서 포착되는 각 노드의 정보보다 더 다양한 정보를 캡처하여 각 노드의 정보가 더욱 포착되어 전파되는 Egress 초기화 기법 2가지의 기법을 제안한다.

제 3 장 그래프 콘볼루션 네트워크 학습 방식

그림 5는 추천 시스템에 적용된 일반적인 GCN 알고리즘의 학습 구조를 일반화하여 보여주고 있다. 그림에서 보이듯이 일반적인 GCN 알고리즘에서는 학습 초기에 사용자-아이템 행동 기록의 행렬을 통해 초깃값을 생성한다. 여기서 생성된 초기 Layer(shallow 임베딩)인 $e_u^{(0)}$와 $e_i^{(0)}$만을 학습에 사용하고 다음 Layer에는 propagation rule에 따라 전파된다. 수식 1은 NGCF알고리즘의 propagation rule이고, 수식 2는 LightGCN의 propagation rule이다. 여기서 hop은 그래프에서 노드 사이의 거리를 의미하며, GCN 알고리즘의 Layer는 주변 노드의 정보를 집계하는 역할을 한다. 따라서 propagation rule을 진행함에 있어, 첫 번째 Layer를 거치면, 대상이 되는 노드의 1-hop neighbor의 정보를 집계하며, 2 Layer의 경우 1-hop의 neighbor, 즉 2-hop까지의 정보를 집계하게 되며, N개의 Layer를 거친다면 N-hop까지의 정보를 집계한다. 이렇게 집계되는 정보를 NGCF의 저자가 처음으로 협업 신호라 명명하였으며, 이러한 고차 연결성의 정보를 내포하고 있는 협업 신호를 GCN에 전파하여 학습시킴으로써 추천 성능을 MF 방식에 비해 정확도를 향상하게 되었으며, 추천 시스템에 딥 러닝을 활용하게 되었다.

수식 1, 2의 공통으로 나타나는 N_u와 N_i는 각각 사용자 u와 아이템 i의 연결된 노드를 의미하며, symmetric normalization term(=1/$\sqrt{|N_u|}\sqrt{|N_i|}$)에 따라 convolution 연산을 진행함에 있어 임베딩 값이 커지지 않도록 정규화하는 역할을 하며, 이웃 노드(사용자, 아이템)의 개수로 나눠주는 것을 의미한다. 정규화를 진행하는 이유는 사용자 u를 기준으로 연결된 아이템이 많을수록 협업 신호가 점점 커지기 때문이다. $e_u^{(k)}$는 사용자 u의 k번째 Layer에서의 임베딩이고, $e_i^{(k)}$는 아이템 i의 k번째 Layer에서의 임베딩이다. 그래프에서 k번째 hop을 포함하는 $k+1$번째 Layer의 임베딩 값을 얻기 위해 사용자 u와 아이템 i의 neighbors의 이전 임베딩 값을 정규화된 가중치로 합산한다.

일반적인 GCN 알고리즘은 사용자와 아이템에 대한 예측 Layer(최종 임베딩 vector(수식 3, 4)를 얻기 위해 propagation rule에 따라 나온 Layer 별 임베딩들을 통해 계산한다. 수식 3은 NGCF의 예측 Layer이며, 수식 4는

LightGCN의 예측 Layer이다. NGCF 알고리즘의 선호도 예측 Layer는 수식 3과 같이 L 차까지의 임베딩 벡터를 combination 하여 최종 임베딩 벡터를 계산한 후, 사용자-아이템 벡터를 내적하여 최종 선호도 예측값(수식 5)를 계산한다. 하지만 LightGCN 알고리즘은 최종 임베딩 벡터는 k-층으로 된 Layer의 임베딩을 각각 α_k배 하여 가중합으로 최종 임베딩 벡터를 계산한 후 NGCF와 동일하게 사용자-아이템 벡터를 내적 하여 최종 선호도 예측 값(수식 5)를 계산한다. 여기서 k는 Layer의 index를 의미하고, NGCF Prediction Layer의 W_1, W_2는 weight matrix를 나타내며, 이전 임베딩 Layer 값과 행렬곱을 수행한다. LightGCN Prediction Layer의 a_k는 k번째 Layer 임베딩의 중요도다. LightGCN에서는 a_k를 $1/(K+1)$로 설정하고 있다. 이는 깊어질수록(멀어질수록) 가중치는 작아진다는 의미한다.

$$e_u^{(k+1)} = \sigma\left(W_1 e_u^{(k)} + \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} \left(W_1 e_i^{(k)} + W_2 \left(e_i^{(k)} \odot e_u^{(k)} \right) \right) \right),$$

$$e_i^{(k+1)} = \sigma\left(W_1 e_i^{(k)} + \sum_{i \in N_i} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} \left(W_1 e_u^{(k)} + W_2 \left(e_u^{(k)} \odot e_i^{(k)} \right) \right) \right) \quad \text{(수식 1)}$$

$$e'^{(k+1)}_u = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e'^{(k)}_i;$$

$$e'^{(k+1)}_i = \sum_{i \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e'^{(k)}_u \quad \text{(수식 2)}$$

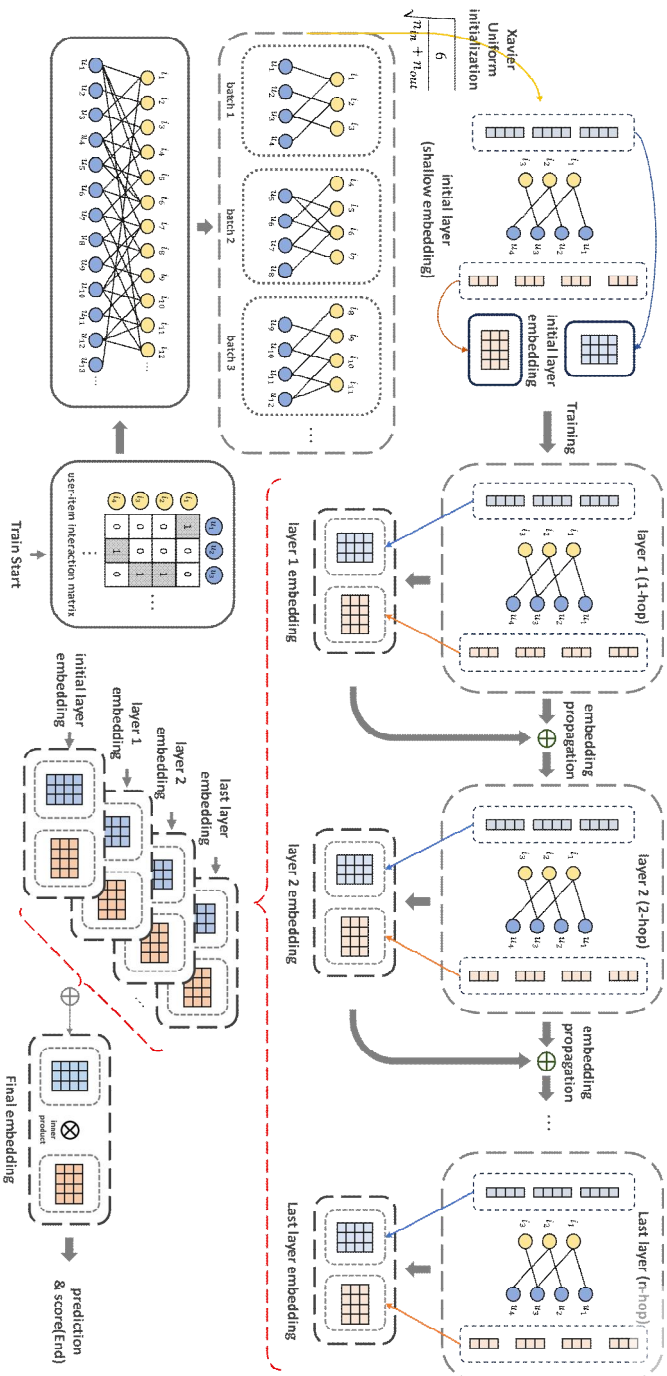
$$e_u^* = e_u^{(0)} \| \cdots \| e_u^{(L)}, \; e_i^* = e_i^{(0)} \| \cdots \| e_i^{(L)} \quad \text{(수식 3)}$$

$$e'_u = \sum_{k=0}^{n} \alpha_k e'^{(k)}_u; \; e'_i = \sum_{k=0}^{n} \alpha_k e'^{(k)}_i \quad \text{(수식 4)}$$

$$\hat{y}_{ui} = e_u^T e_i \quad \text{(수식 5)}$$

NGCF의 가장 핵심적인 부분만 사용하여 더 정확하고 가벼운 추천 알고리즘인 LightGCN 알고리즘에서는 초기 Layer로부터 propagation rule에 따라 각 Layer의 임베딩을 계산하고 각 Layer 임베딩 값을 가중합 함으로서 사용자와 아이템의 최종 임베딩 값인 e_u와 e_i를 계산해 낸다. 사용자 u와 아이템 i 간의 선호도를 나타내는 score $\hat{y}_{ui}$는 e_u와 e_i를 수식 5과 같이 내 적(inner product)함으로써 계산된다. LightGCN에서는 이전 홉 Layer에서 도출된 임베딩 값을 가중 합함으로써 그래프 convolution이 가진 self-connected 효과를 유지하여 comprehensive representation 한 추출을 유도한다.

그러나 NGCF, LightGCN은 다음 Layer의 임베딩을 구하기 위해 symmetric normalization term을 수행하며 해당 과정을 통해 다음 Layer에 값을 전달하는 과정을 반복함으로써 Layer를 지나갈 때마다 임베딩 값이 사라지는 embedding 값 소실 문제를 겪게 된다. 해당 문제로 인하여 GCN 알고리즘의 전체적인 학습 과정을 저해하며, 추천 성능을 떨어트리는 것이다. 이러한 문제를 극복하기 위해 symmetric normalization term을 수행하기 전 임베딩 값에 스칼라 곱을 수행하여 동일한 값을 배로 증가시켜 다음 Layer에 전달함으로써 GCN 알고리즘의 학습 과정을 개선하며, 추천 성능을 증가하기 위한 WF 기법을 제안한다. 또한 GCN의 학습을 반복하는 과정 속 초기화의 범위가 상당히 많은 영향을 미친다. 기존 자비에르 초기화를 사용하면 적은 범위의 초깃값으로 인하여 안정적인 학습이 가능하지만, embedding 값 소실 문제가 발생하는 것으로 보아 더욱 큰 범위를 가져도 되는 것으로 판단되어 기존 초깃값의 범위보다 약 5배의 더 큰 범위를 가지며 그래프 임베딩 크기에 맞춘 초기화 범위를 갖는 Egress 초기화 기법을 제안한다.



<그림 5> 그래프 콘볼루션 네트워크의 학습 과정 상세 구조도

제 4 장 제안 방법

본 연구에서는 이러한 사용자-아이템의 임베딩 값의 변화를 주기 위한 2가지 기법(Egress 초기화 기법, 가중치 전달(Weight Forwarding, WF)기법)을 제안하며, 제안 방법으로 인해 변화된 임베딩 값에 대한 다양한 분석을 수행하였다.

제 1 절 Egress 초기화 (Egress Initialization) 기법

딥 러닝의 학습 과정에서 초기화 값은 학습에 많은 영향을 끼친다는 것은 저명하게 알려져 있다. GCN의 학습 과정에서 임베딩 값의 평균값이 부록 A를 통해 알 수 있듯 사용자 임베딩: $7.91E-02$, 아이템 임베딩: $2.23E-02$으로 작은 값으로 수렴되는 것을 실험 결과적으로 확인하였다. 이러한 원인은 다양한 요인이 영향을 줄 수 있으나, epoch에 따른 지속적인 학습을 반복할수록 초기화 값과 범위에 따라 학습 과정에 영향을 많이 끼치게 된다. 딥 러닝의 학습 과정에서 전통적으로 우수한 Xavier Uniform 초기화(Xavier Uniform Initialization, 수식 6) 기법은 신경망의 각 Layer에서 출력의 분산이 입력의 분산과 동일하게 유지되도록 설계되었으며, 이에 따라 학습 중에 가중치가 너무 작아지거나 커지는 것을 방지한다. 수식 6의 $U(a,b)$는 $[a,b]$ 범위의 균일 분포, N_{in}는 입력 유닛의 수, N_{out}는 출력 유닛의 수(임베딩 크기)이다.

$$U\left(-\sqrt{\frac{6}{N_{in}+N_{out}}},\sqrt{\frac{6}{N_{in}+N_{out}}}\right) \quad \text{(수식 6)}$$

Xavier Uniform 초기화 기법은 활성화 함수가 선형이거나 tanh와 같은 대칭적인 활성화 함수를 사용할 때 특히 효과적이다. 이와 같은 Xavier Uniform 초기화 기법이 주요하게 사용되는 이유는 GCN의 비선형 활성화 함수를 제거하는 것이 성능적인 측면과 효율적인 측면에서 더욱 우수하다는 LightGCN을 통해 입증되었기 때문이며, 이와 같은 이유로 LightGCN 알고리즘과 LightGCN을 기본 구조로 활용하는 후속 알고리즘들에서는 Xavier Uniform 초기화 기법이 주로 사용되고 있다.

또한 Xavier Uniform 초기화 기법은 입출력을 모두 고려하기에 입력 유닛의 수가 많으면 많을수록 더 작은 초기화 범위를 갖기에 대규모 데이터 집합에 경우 상당히 작은 값의 초기화 범위를 갖는다. 하지만 학습 과정에서 너무 작은 초기화 범위를 사용하면 여러 문제를 초래할 수 있다. 우선 기울기 소실 문제를 발생시킬 수 있어 네트워크의 깊은 층에서 학습을 방해하며, 학습이 매우 느려지거나 정지할 수 있으며, 학습 속도가 느려지고 알고리즘이 더 많은 학습 시간이 필요하게 되며 이는 학습 시간의 증가를 초래한다. 작은 초깃값은 비용 함수의 높은 초깃값을 초래할 수 있어 알고리즘이 전역 최적해에 도달하는 데 더 많은 시간이 필요하게 된다. 마지막으로, 모든 뉴런이 동일한 출력을 생성하게 되어 네트워크가 복잡한 패턴을 학습하는 데 어려움을 겪게 되는 대칭성 문제도 발생할 수 있으며 이러한 문제를 극복하여 정확도 및 학습 시간의 감소를 위해 더욱 넓은 범위를 갖는 초기화 기법이 필요로 하였다. 따라서 제안하는 Egress 초기화(Egress Initialization, 수식 7) 기법은 Layer의 출력(임베딩 크기)만을 고려하여 더욱 넓은 범위의 초기화 범위 및 분포를 갖게 하여 실험을 진행하였다.

$$U\left(-\frac{1}{\sqrt{N_{out}}},\frac{1}{\sqrt{N_{out}}}\right) \qquad \text{(수식 7)}$$

예를 들어, 입력 유닛의 수가 10,000이고 출력 유닛의 수가 64라면, Xavier Uniform 초기화 기법을 사용할 경우 $(-0.024,0.024)$의 범위로 초기화되며, 제안하는 Egress 초기화 기법을 사용한다면, $(-0.125,0.125)$의 범위로 초기화가 진행된다. 제안하는 Egress 초기화 방법은 5.2배 가량 더 높은 범위를 가진다. 이러한 점은 가중치 발산/폭발과 같은 문제를 가져올 수도 있지만, GCN 알고리즘의 정확도 및 학습 속도의 향상을 위해서는 기존보다 넓은 초기화 범위가 필요하며 넓어진 범위를 통해 각 노드의 정보가 전파되기에 동일한 그래프 구조에서 얻게 되는 정보가 기존 초기화 기법보다 더욱 다양한 정보를 캡처할 수 있기에 더 넓은 범위의 갖는 것이 GCN의 정확도와 학습 속도를 더욱 개선했으며, 해당 실험을 진행하기 위해 기존 알고리즘에 Egress 초기화 기법만을 적용한 실험을 진행하였다.

제 2 절 가중치 전달 (Weighted Forwarding, WF) 기법

제 3 장에서 설명한 일반적인 GCN 알고리즘에서 각 알고리즘의 사용자 임베딩 값과 아이템 임베딩 값을 통계적으로 분석 확인한 결과, 사용자-아이템 임베딩 값의 평균값은 사용자 임베딩은 $7.91E-02$, 아이템 임베딩은 $2.23E-02$로 확인되었다.

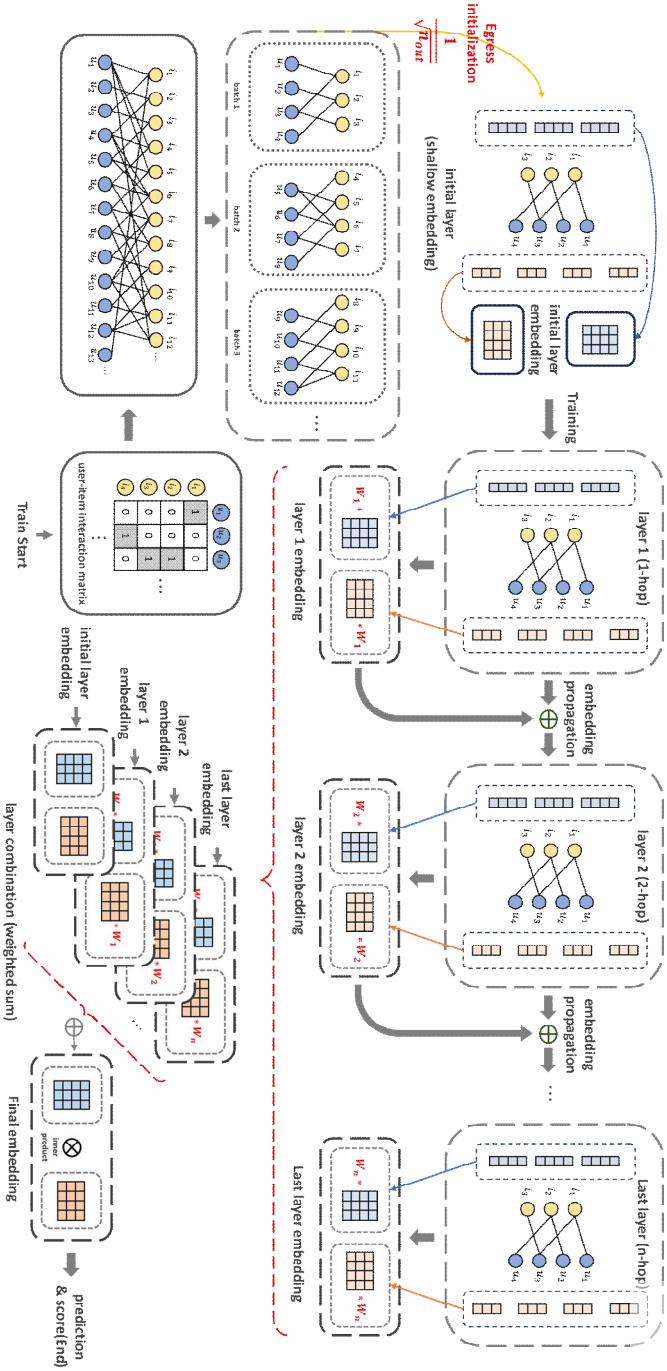
이를 통해 전체 임베딩 값이 상당히 작은 값들로 학습되었기에 임베딩 값 소실 문제가 발생하였음을 알 수 있었다. 이러한 원인으로 인해 기존의 일반적인 GCN 알고리즘은 사용자가 지속해서 보던 아이템 특징만을 강건하게 학습됨으로써 추천 시스템의 본질적인 사용 이유인 사용자가 다른 방법으로는 접하지 못했을 새롭고 관련성 높은 콘텐츠를 발견하기 위한 확률이 더욱 저하되었다.

그림 6은 임베딩 값 소실 문제를 극복하고자 본 논문에서 제안하는 학습 과정(WF 기법, Egress 초기화 기법)의 구조이다. 제안하는 구조는 수식 8과 같이 노드들의 임베딩 값에 특정 가중치를 곱한 값을 다음 Layer로 전달함으로써 보다 sharpened embedding 값($e''^{(k+1)}_u$, $e''^{(k+1)}_i$)을 얻는다.

$$e''^{(k+1)}_u = w_k \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e''^{(k)}_i;$$

$$e''^{(k+1)}_i = w_k \sum_{i \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e''^{(k)}_u \quad \text{(수식 8)}$$

제안하는 Egress 초기화 기법과 WF 기법을 시행하여 얻은 sharpened embedding 값의 평균값은 사용자 임베딩: $-3.03E+00$, 아이템 임베딩: $-9.98E-01$로 약 38배, 48배 증가하였으며, 이러한 임베딩 값의 증가 원인으로 인하여 학습 속도 개선 및 정확도 개선이 이루어졌다.



<그림 6> 제안하는 Egress 초기화 기법과 WF 기법을 적용한 상세 구조도

제 3 절 제안 방법의 수식 표현(Equation Expressions)

본 절에서는 기존의 GCN 알고리즘에서 사용된 수식적 표현들은 LightGCN을 바탕으로 제안하는 WF 기법에 관해 설명하며, NGCF 알고리즘의 Weight Matrix와의 차이점을 설명한다.

제안하는 WF 기법은 스칼라 곱(수식 9)으로 행렬의 모든 원소에 주어진 스칼라값을 곱한다. 예를 들어, 스칼라 w와 행렬 A가 있다면, 스칼라 곱 wA는 아래와 같이 수행된다.

$$wA = \begin{bmatrix} w^*a_{11} & w^*a_{12} & \cdots \\ w^*a_{21} & w^*a_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad \text{(수식 9)}$$

스칼라 곱은 행렬의 차원을 변경하지 않지만, 모든 원소를 부여한 스칼라값으로 스케일링한다.

NGCF에서 수행되었던 Weight Matrix는 이전 임베딩에 행렬 곱(수식 10)을 수행하여 다음 Layer에 전달하는 방식을 가지고 있으며, 행렬 A와 B를 받아 새로운 행렬 C를 생성하며, 행렬 A의 각 행과 행렬 B의 각 열 간의 내적을 계산하여 C의 해당 위치의 원소를 얻는다. 행렬 A가 $m \times n$ 차원이고 행렬 B가 $n \times p$ 차원이라면, 결과 행렬 C는 $m \times p$ 차원이 된다. 이러한 과정을 통해 얻게 되는 행렬 C를 임베딩으로 사용한다.

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \times B_{kj} \quad \text{(수식 10)}$$

행렬 곱은 일반적으로 행렬의 차원을 변경하며, 두 행렬의 정보를 종합적으로 결합한다. 따라서 제안하는 WF 기법은 기존의 NGCF에서 사용되던 행렬 곱 연산과는 다른 스칼라 곱 연산을 사용하였다.

이러한 임베딩 벡터의 스칼라 곱은 사용자와 아이템 간의 상호 작용의 포착 결과에 큰 영향을 미치기 때문에 연산의 영향을 분석하기 위해서는 임베딩의 통계치의 값들에 대한 분석이 수행되어야 한다. 임베딩의 평균과 중앙값 변화

는 임베딩 공간의 전반적인 이동과 분포를 나타내며, 특히 평균의 변화는 임베딩 공간의 중심 이동과 사용자의 선호도 변화를 반영하며, 중앙값은 데이터의 비대칭성과 이상치에 대한 정보 갖는다. 이러한 통계치의 변화는 추천의 정확도, 다양성 및 설명 가능성과 같은 다양한 평가 메트릭과 연관되어 있다.

이러한 스칼라 곱을 통한 추천 시스템의 성능 개선을 파악하기 위해서 임베딩의 통계적 값을 분석하여 알고리즘들의 성능 개선 원인과 새로운 임베딩이 상호 작용을 더 상세하게 파악할 수 있다. 따라서 스칼라 곱을 통한 임베딩의 평균과 중앙값의 변화는 추천 시스템의 성능에 중요한 역할을 하며, 다음은 LigthGCN 알고리즘에 WF 기법을 적용한 수식 표현이다. R은 사용자-아이템 interaction matrix(수식 11)를 의미하며 M과 N은 각각 사용자의 수, 아이템의 수를 의미한다.

$$R \in R^{M \times N} \quad \text{(수식 11)}$$

사용자-아이템 그래프의 adjacency matrix A는 수식 12와 같이 표현하였으며, diagonal matrix는 대각 성분을 제외한 나머지 index는 전부 0의 값을 가진 행렬을 의미한다.

$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix} \quad \text{(수식 12)}$$

여기서 0번째 Layer의 임베딩 매트릭스는 수식 13과 같이 표현되며, T는 임베딩 크기를 의미한다. propagation rule에 따라 계산되는 $k+1$번째 Layer의 임베딩 값은 수식 14와 같이 표현된다. 여기서 D는 $(M+N)\times(M+N)$차원을 가진 diagonal matrix를 의미한다.

$$E^{(0)} \in R^{(M \times N) \times T} \quad \text{(수식 13)}$$

$$E^{(k+1)} = \left(D^{-\frac{1}{2}} A D^{\frac{1}{2}}\right) E^{(k)} \quad \text{(수식 14)}$$

수식 15는 제안하는 WF 기법의 수식 표현이다. WF 기법은 k번째 Layer의 학습을 거치면서 나오게 되는 $E^{(k)}$에 가중치 w_k를 곱하여 $k+1$번째 임베딩 값을 결정한다. w_k을 통해 각 Layer마다 서로 다른 가중치를 부여할 수 있으며, 이를 통해 임베딩에 대한 영향도를 조절할 수 있다.

$$E'^{(k+1)} = w_k\left(D^{-\frac{1}{2}}AD^{\frac{1}{2}}\right)E^{(k)} \quad \text{(수식 15)}$$

수식 16은 제안하는 학습 과정을 마친 후 얻게 되는 임베딩을 표현한 것이다. 여기서 E'는 사용자와 아이템에 대한 최종 임베딩 값이 들어 있는 벡터를 의미한다. 해당 최종 임베딩을 통해 추천 성능을 파악할 수 있는 예측값($\hat{y}_{ui}$)을 얻는다.

$$E' = \sum_{k=0}^{n}\alpha_k E^{'(k)} \rightarrow \hat{y}_{ui} \quad \text{(수식 16)}$$

제 5 장 실 험

본 연구에서는 이러한 추천 시스템에 추천 성능을 저해하는 임베딩 값 소실 문제를 개선하여 추천 성능 및 학습 효율성을 개선하기 위한 방법으로 Egress 초기화 기법과 WF 기법을 제안하였으며, 제안하는 학습 과정의 검증을 위해 위의 LightGCN, BUIR, MixGCF, SGL, SimGCL, XSimGCL, MF, DirectAU, SelfCF을 활용하였으며, MF, DirectAU, SelfCF 알고리즘은 기본 구조가 달리 사용되었기에 제안하는 WF 기법의 적용이 불가했다. 나머지 알고리즘은 모두 LightGCN을 기본 구조로 사용하여 layer combination(weighted sum)의 구조가 같기에 6개의 알고리즘에 제안 방법이 동일하게 적용할 수 있다. 전체적인 실험에는 SELFRec [27]을 활용하였으며, SELFRec는 GCN 기반 추천 시스템 알고리즘을 위한 open source library이다. 해당 library는 PyTorch 기반의 다양한 추천 알고리즘이 구현되어 있으며, 추천 시스템 연구 및 개발을 용이하게 한다. 또한 관련 연구 중 가장 최신 연구인 XSimGCL의 연구에서 본 실험과 동일한 라이브러리를 활용하였다. 실험 과정에 SELFRec를 통해 구현된 알고리즘에 제안 방법론을 적용하였으며, 실험에 사용된 데이터 집합과 환경 및 매개변수는 5장 1절, 5장 2절에, 알고리즘의 추천 성능 평가를 위한 평가 함수는 5장 3절에 실험 결과에 대한 분석은 5장 4절에 나타냈다.

제 1 절 실험 데이터

제안하는 방법의 검증을 위한 실험에는 4가지 종류의 데이터 집합을 활용했다. MovieLens 데이터 집합 [28] 은 영화 추천 시스템에 많이 사용되는 데이터 집합으로 사용자들의 영화에 대한 평점에 대한 상호 작용 정보를 포함하고 있다. 상호작용 데이터는 사용자와 아이템 사이의 시청 기록을 의미하며 이러한 상호 작용 관계를 이분 그래프로 표현하고, 해당 상호 작용을 학습하여 사용자와 아이템의 매핑 예측을 수행한다. 해당 데이터 집합의 크기는 사용자의 시청 기록의 개수를 기준으로 나누어져 있다. 이와 마찬가지로 FilmTrust 데이터 집합은 FilmTrust 웹사이트에서 사용자의 영화 평가 정보에 대해서 크롤링하여 만들어진 데이터 집합이며 MovieLens 데이터 집합과 마찬가지로 영화 추천 시스템에 많이 사용된다. 실험에 있어 MovieLens 데이터 집합의 평점 정보는 4점에서 5점 사이의 데이터만 학습에 사용되었으나, Film-Trust 데이터 집합은 0.5점에서 4점까지 평점 정보를 학습에 활용하였다. Douban-book 데이터 집합은 중국의 도서 리뷰 웹사이트인 Douban에서 크롤링하여 만든 데이터 집합이며, 해당 데이터 집합은 도서 평가와 사용자 정보를 포함하고 있다. 실험에 사용된 해당 데이터 집합의 평점 범위는 4점에서 5점까지 활용되었다. Yelp2018 데이터 집합은 Yelp Challenge 2018에서 비롯되었으며, 해당 데이터 집합은 두 국가에 걸쳐 10개 대도시 지역의 레스토랑이나 바와 같은 지역 사업체에 대한 평점 기록으로 사용자와 아이템과 상호 작용이 있는 것들만 표시되어 있다. 4가지 데이터 집합의 상세한 개수는 표 1을 통해 작성하였다.

본 연구의 실험에서 사용된 데이터 집합들은 선행 연구에서 활용되었던 데이터 집합들과 동일하게 사용하였으며, 해당 데이터 집합들은 추천 시스템 알고리즘의 성능 평가를 위한 벤치마크 데이터 집합으로 널리 사용되고 있다.

<표 1> 실험 데이터 집합의 상세 표

	FilmTrust		Douban-book	
	Train	Test	Train	Test
User	1,499	812	12,638	10,882
Item	2,033	340	22,222	19,075
Interaction	33,750	1,747	478,730	119,690
Rathing range	0.5~4	0.5~4	4~5	4~5
Sparsity	98.89%	99.37%	99.83%	99.94%
	MovieLens-1M		Yelp2018	
	Train	Test	Train	Test
User	6,038	5,989	31,668	31,668
Item	3,492	3,190	38,048	36,073
Interaction	460,359	114,922	1,237,259	324,147
Rathing range	4~5	4~5	None	None
Sparsity	97.82%	99.40%	99.90%	99.97%

제 2 절 실험 환경 및 매개변수

실험 환경(Experimental environments)은 표 2과 같은 환경에서 이루어졌으며, 표 3에는 알고리즘의 사용된 매개변수를 나타냈다. 학습 데이터는 선행 연구에서 사용하던 학습, 평가 데이터를 그대로 사용하였다. 모든 알고리즘은 Layer 3개, embedding size 64, learning rate 0.001, reg lambda 0.0001로 동일하게 수행되었다.

<표 2> 실험 환경

Resource	Property
OS	Ubuntu 18.04.06 LTS
Nvidia Driver	525.78.01
CUDA	12.0
CuDNN	11.6
Python	3.7.11
Pytorch	1.10.0
Seed	42
Optimizer	Adam
Epoch	300
Batch size	2048

<표 3> 실험에 사용된 알고리즘 별 매개변수

BUIR		MixGCF		LightGCN	
tau	0.995	n_negs	64		
Drop rate	0.2				
SGL		**XSimGCL**		**SimGCL**	
temperature	0.2	l*	1	lambda	0.5
lambda	0.1	lambda	0.2	eps	0.1
droprate	0.1	eps	0.2		
		tau	0.15		
MF		**SelfCF**		**DirectAU**	
		tau	0.05	gamma	2

제 3 절 실험 평가

알고리즘의 성능 평가를 위한 평가 함수로는 총 2가지가 사용했다. 추천 시스템 알고리즘 성능 평가에 사용되는 Recall@K, Normalized Discounted Cumulative Gain@K(NDCG@K)은 다양한 측면에서 알고리즘의 품질을 평가하기 위해 사용했다. Recall@K은 추천의 정확성과 다양성 측면을 평가하고, NDCG@K은 사용자가 관심 있는 아이템들을 상위 순위로 정확하게 추천하는지를 평가한다.

Recall@K은 실제 사용자가 관심 있는 아이템 중 추천 시스템이 제안한 아이템의 비율을 나타낸다. 즉, 추천 시스템이 얼마나 많은 관련 아이템을 추천하였는지를 측정하며, 높은 재현율은 추천 시스템이 사용자에게 놓치지 않고 많은 관련 아이템을 제안하는 것을 의미한다. NDCG@K는 추천된 아이템들의 관련성과 그 순위를 고려하여 평가하는 함수로, 사용자가 선호하는 아이템들이 상위에 위치할수록 높은 값을 가진다. 따라서 높은 NDCG@K는 추천 시스템이 사용자의 관심 있는 아이템들을 높은 순위로 정확하게 제안하는 것을 나타낸다. 실험에는 K = 20을 기준으로 실험 평가를 진행하였다

5장 4절에는 결과에 대한 요약을 위해 선행 연구에서 주요한 성능 평가 함수로 활용하였던 Recall@20, NDCG@20의 결과를 나타냈으며, 부록 1에 평가 평가 함수의 수학적 표현을 작성하였으며, 부록 2, 3에 모든 실험 결과에 대한 사용자-아이템 임베딩 값들의 통계치와 평가 함수에 따른 모든 실험 성능 결과를 나타냈다.

제 4 절 실험 결과

표 4에는 기존 알고리즘의 사용자-아이템의 임베딩 값에 대한 통계치의 평균값과 제안하는 방법 중 가장 개선된 결과에 대한 통계치의 평균값을 나타냈으며, 전체적인 임베딩 값들의 통계치는 부록 2를 통해 나타냈다. 여기서 가중치를 부여하는 실험의 가중치는 2의 0승부터 2의 11까지 실험을 진행하였고, Egress 초기화의 적용 여부에 따른 실험도 수행하였으며, 부록 2에는 알고리즘별 기존 실험 대비하여 제안 방법 중 가장 개선된 결과만 나타내었다. 또한 본 연구에서는 선행 연구들과의 비교를 위해 모든 알고리즘을 동일하게 3개의 Layer를 기준으로 하여 실험을 수행했다. 수행된 실험 결과에 대한 분석을 진행하기 위해 사용자-아이템 임베딩 값에 대한 통계적 분석 및 주성분 분석(Principle Component Analysis, PCA)과 가우시안 커널 밀도 추정(Gaussian kernel density estimation, GKDE)을 수행하였으며, 그림과 표를 통해 시각화 및 정량적으로 표현하였다.

우선 실험에 사용한 데이터 집합별 사용자-아이템 임베딩 값 통계치의 평균값을 통해 임베딩 값의 변화에 대해 분석하였으며, 정량적인 수치는 표 4에 나타내었다. 통계치의 평균값의 분석 결과, 제안된 2개의 방법론은 일반적으로 사용자와 아이템 임베딩의 평균값을 감소시켰다. 특히, FilmTrust 데이터 집합에서는 사용자-아이템 임베딩의 표준 편차가 많이 증가하였으며, 이는 임베딩의 분포가 더 넓어졌음을 나타낸다. 이러한 넓어진 분포는 임베딩이 사용자와 아이템의 특성이나 상호 작용을 더 넓은 범위로 잡아내고 있을 수 있음을 의미한다. 사용자-아이템 임베딩의 평균값과 중앙값은 기존 방법보다 제안 방법에서 감소하였으며, 이러한 감소한 현상이 보이는 것은 임베딩 벡터의 전반적인 값들이 낮아진 것을 나타내며, 이는 임베딩 공간의 중심 위치가 변화했음을 의미한다. 또한 데이터 포인트들이 특정 영역에 더 밀집되어 있을 가능성을 시사한다. 이러한 변화는 제안된 방법론을 통해 알고리즘이 데이터의 다양한 특성이나 패턴을 새롭게 포착하거나, 불필요한 정보나 노이즈를 줄이면서 주요 특성을 더 강조하게 되었다는 것을 의미한다.

Yelp2018 데이터 집합에서의 분석 결과, 제안된 방법론들은 사용자와 아이템 임베딩의 평균 및 중앙값을 눈에 띄게 증가시켰다. 이는 임베딩 벡터들의

전반적인 값들이 상승하였음을 나타내며, 임베딩 공간의 중심 위치가 이동했음을 의미한다. 이런 변화는 데이터 포인트들이 특정 영역에서 더 분산되어 있을 수 있음을 나타낼 수 있다. 또한 표준 편차의 변화는 미미하여 분포가 상대적으로 일정하게 유지되고 있다. 이러한 결과는 제안된 방법론이 데이터의 특성이나 패턴을 다르게 해석하게 했다. 또한, 제안된 방법을 통해 불필요한 정보나 노이즈의 영향을 줄이고, 주요 특성이나 패턴을 더욱 강조하여 임베딩을 생성하였다.

MovieLens-1m과 douban-book 데이터 집합에서는 제안된 방법론들을 적용하였을 때, 사용자-아이템 임베딩의 표준 편차에 큰 변화는 관찰되지 않았다. 그러나 임베딩의 평균 및 중앙값에서는 눈에 띄는 감소가 있었다. 이는 임베딩 벡터들의 전반적인 값들이 낮아졌음을 나타내며, 임베딩 공간의 중심 위치가 변화했음을 의미한다. 이러한 변화는 데이터 포인트들이 임베딩 공간에서 특정 영역에 더 밀집되어 있다는 것을 의미한다. 이러한 밀집된 영역은 제안된 방법론을 통해 알고리즘이 데이터의 다양한 특성이나 패턴을 더욱 명확하게 포착하게 되었을 수 있음을 나타낸다. 또한, 이러한 변화는 Yelp2018 데이터 집합의 분석 결과와 유사하게 알고리즘이 불필요한 정보나 노이즈를 제거하면서 주요 특성들을 더욱 강조하게 되었다는 것을 나타낼 수 있습니다. 추가로 douban-book 데이터 집합에서는 표준 편차의 감소하였으며 이를 통해 임베딩의 분포도 더 밀집되었음을 확인할 수 있다.

이러한 데이터 집합 별로 관찰된 임베딩 값들의 통계 변동성은 제안된 방법의 영향이 데이터의 성격과 특성에 따라 다르게 나타날 수 있음을 의미한다. 이러한 임베딩 평균값의 증감과 함께 표준 편차와 같은 분포 지표의 변화는 제안된 방법론이 임베딩 공간을 재구성하여 더 다양한 사용자-아이템 관계를 포착할 수 있음을 나타낸다. 임베딩의 평균과 중앙값의 변화는 알고리즘이 사용자와 항목 간의 상호 작용을 어떻게 해석하는지에 대한 통찰을 제공하며, 임베딩 변화가 추천 성능에 미치는 영향을 이해할 수 있다.

<표 4> 데이터 집합 별 기존 방법과 제안 방법의
사용자-아이템 임베딩 값들에 대한 평균값

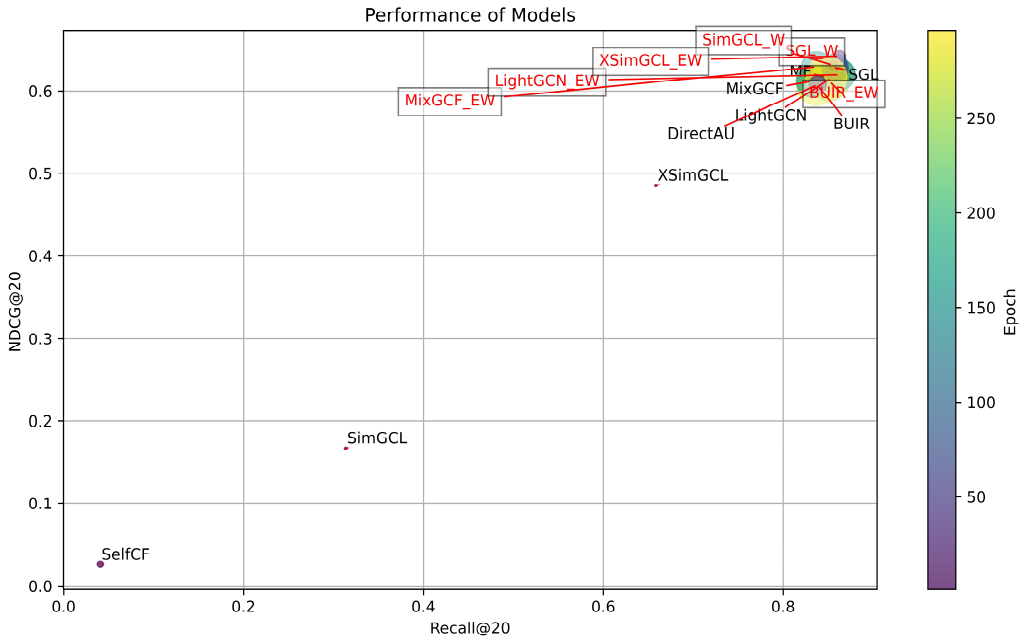
		FilmTrust		Yelp2018	
		기존 평균값	제안 방법 평균값	기존 평균값	제안 방법 평균값
사용자 임베딩	평균	4.97E-02	-3.02E+00	-6.88E-02	5.07E-02
	표준편차	8.06E-01	1.60E+01	3.21E+00	3.25E+00
	표준 오차	2.60E-03	5.16E-02	2.25E-03	2.28E-03
	분산	1.63E-01	2.42E+02	2.87E+00	3.25E+00
	중앙값	9.65E-02	-2.86E+00	-5.05E-02	5.12E-02
아이템 임베딩	평균	6.37E-03	-9.37E-01	-5.77E-02	3.69E-02
	표준 편차	7.25E-01	1.20E+01	2.85E+00	2.91E+00
	표준 오차	2.01E-03	3.31E-02	1.82E-03	1.86E-03
	분산	1.34E-01	1.33E+02	2.34E+00	2.69E+00
	중앙값	8.01E-04	-5.17E-01	-3.81E-02	4.23E-02
		MovieLens-1M		Douban-book	
		기존 평균값	제안 방법 평균값	기존 평균값	제안 방법 평균값
사용자 임베딩	평균	1.06E-01	-1.96E-02	-7.80E-03	-4.58E-02
	표준편차	1.63E+00	1.58E+00	2.28E+00	2.04E+00
	표준 오차	2.62E-03	2.55E-03	2.54E-03	2.27E-03
	분산	6.46E-01	6.46E-01	1.03E+00	7.82E-01
	중앙값	1.22E-01	-3.78E-02	-1.05E-02	-4.16E-02
아이템 임베딩	평균	9.01E-02	-5.17E-02	-1.65E-02	-4.63E-02
	표준 편차	2.09E+00	2.13E+00	1.69E+00	1.43E+00
	표준 오차	4.43E-03	4.51E-03	1.42E-03	1.20E-03
	분산	1.01E+00	1.07E+00	6.07E-01	3.89E-01
	중앙값	7.30E-02	-6.55E-02	-2.07E-02	-3.48E-02

그림 7, 8, 9, 10, 11, 12는 제안 방법을 적용한 알고리즘과 적용하지 않은 기존 알고리즘의 성능 비교를 위한 산점도이며, 전체의 그래프를 통해 파악하기 어려운 데이터 집합의 결과의 경우 밀집된 영역을 확대한 그래프를 나타내

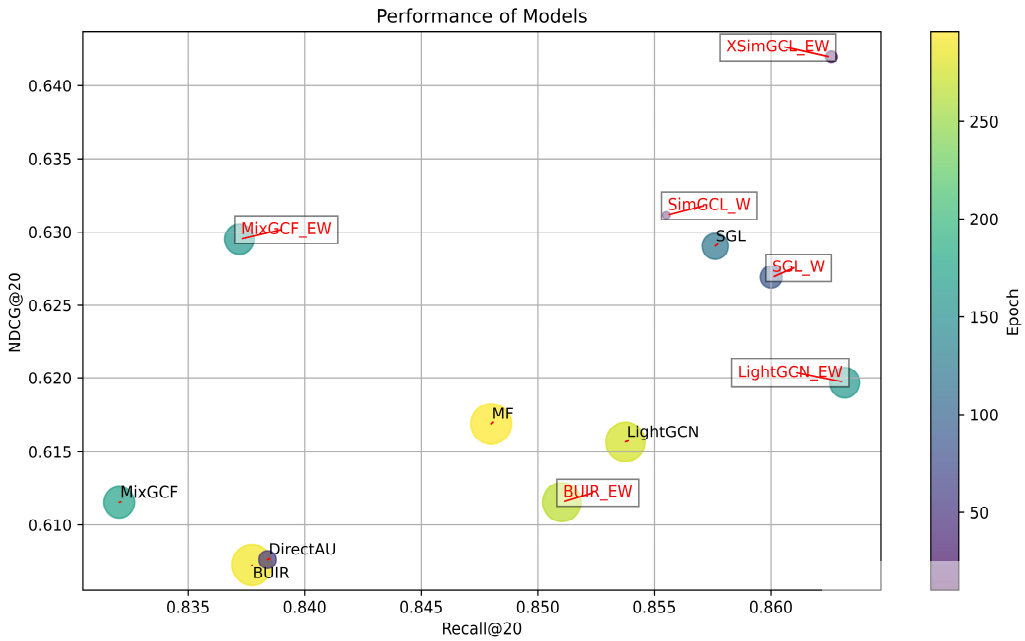
었으며, 기존 알고리즘 명만 표시된 것은 기존 알고리즘을 의미하고, 기존 알고리즘에서 Xavier Uniform 초기화 기법을 사용하였고, 알고리즘 명_E는 기존 알고리즘에 Xavier Uniform 초기화 기법을 대체하여 Egress 초기화 기법을 사용한 것을 의미하며, 알고리즘 명_W는 기존 알고리즘에 WF 기법을 적용하고, 기존에 사용되던 Xavier Uniform 초기화 기법을 사용한 것을 의미하며, 알고리즘 명_EW는 Egress 초기화 기법과 WF 기법을 모두 적용한 알고리즘을 의미한다.

해당 그래프를 통해 알 수 있는 점은, NDCG@20, Recall@20에 대한 성능 비교와 가장 좋은 성능을 내기까지 학습에 소모된 epoch를 알 수 있다. 하지만 전체 결과를 하나의 그래프에 시각화하기 위해서는 나타내는 수치의 범위에 제한이 있었기에 일정 범위내에 있는 알고리즘들만 나타내었다. 또한 표 4에 해당 결과에 대한 정량 평가에 대한 결과를 수치상으로 나타내었다.

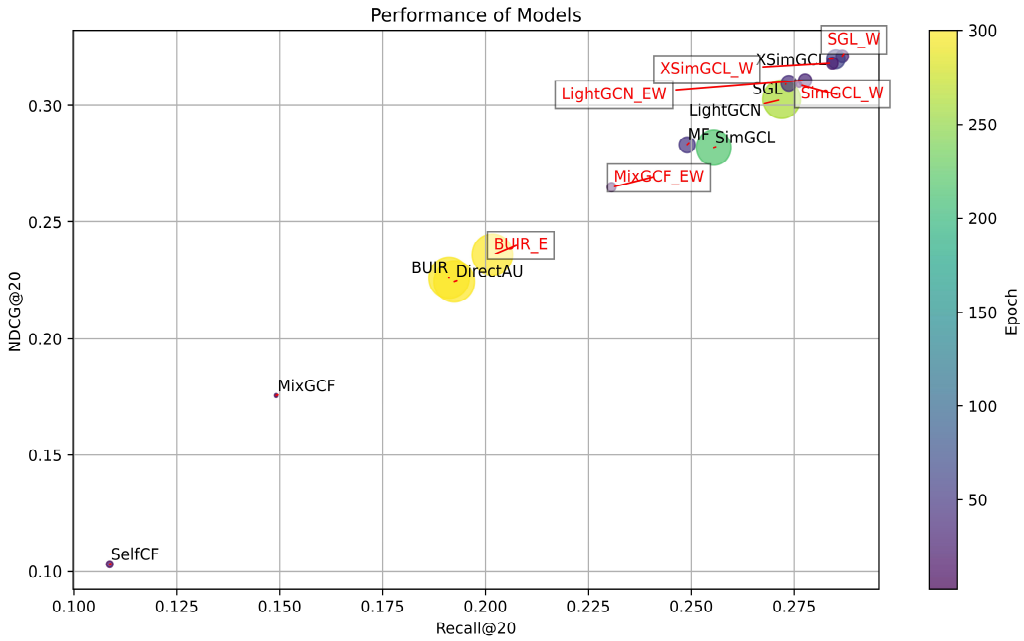
대부분의 알고리즘에서 제안하는 방법론 2가지를 통해 더 높은 Recall@20과 NDCG@20 점수를 더 적은 epoch안에 달성했으며, 학습 시간을 상당히 줄이는 결과를 보였다. 이는 스칼라 곱으로 이루어진 WF 기법과 더 큰 초기화 범위를 갖는 Egress 초기화가 벡터 공간에서 임베딩의 상대적 위치와 분포가 사용자-아이템 상호 작용을 효과적으로 포착하는 데 중요한 역할을 한다는 것을 확인하였다. 또한 FilmTrust 데이터 집합에서의 SimGCL, XSimGCL 알고리즘과 MovieLens-1M 데이터 집합에서의 MixGCF 알고리즘의 경우 실험 결과 epoch 1~2에서 최고 성능을 보이고 개선이 이루어지지 않는 것으로 보아 학습이 제대로 이루어지지 않았지만, 제안하는 방법들을 통해 이러한 문제를 극복하였다.



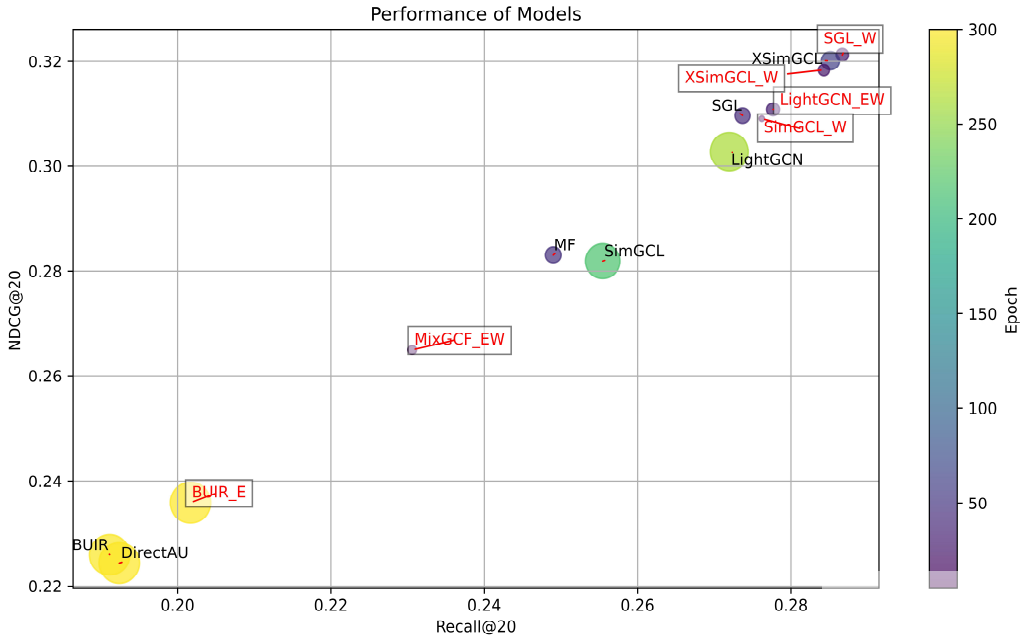
<그림 7> FilmTrust 데이터 집합의 알고리즘 별 성능 산점도 (전체)



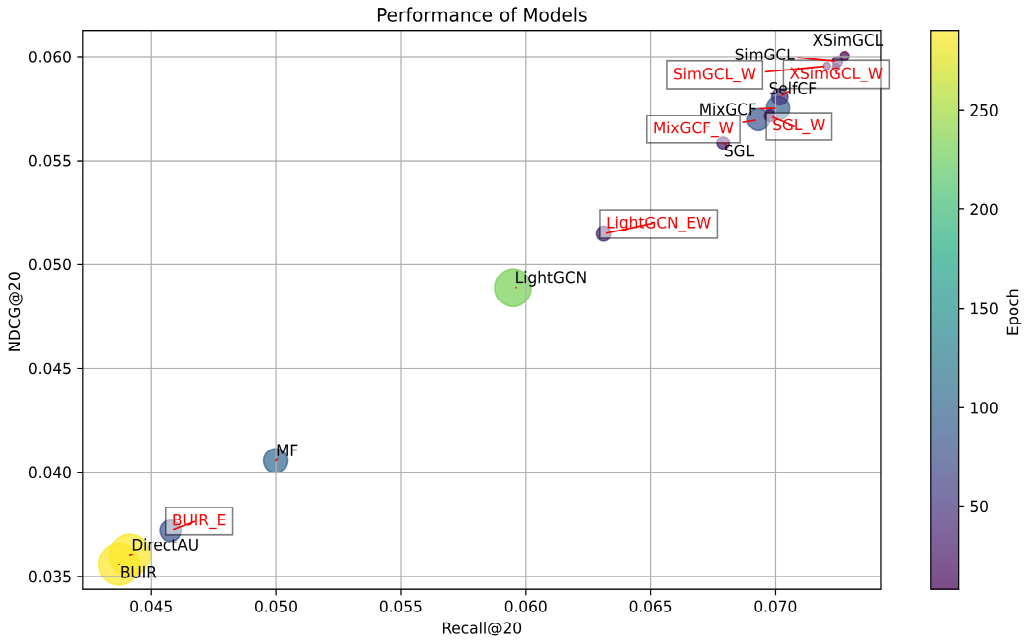
<그림 8> FilmTrust 데이터 집합의 알고리즘 별 성능 산점도 (확대)



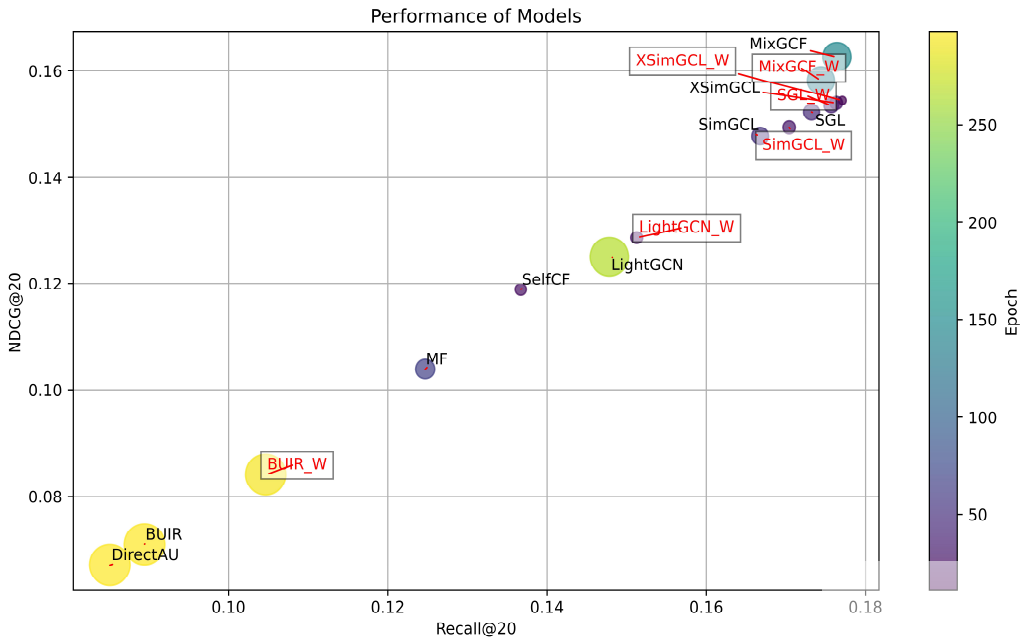
<그림 9> MovieLens-1M 데이터 집합의 알고리즘 별 성능 산점도 (전체)



<그림 10> MovieLens-1M 데이터 집합의 알고리즘 별 성능 산점도 (확대)



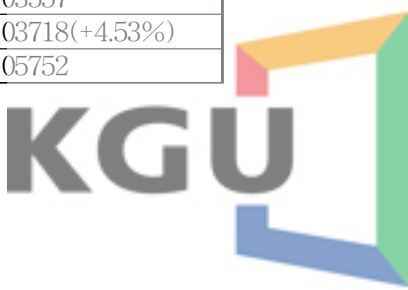
<그림 11> Yelp2018 데이터 집합의 알고리즘 별 성능 산점도 (전체)



<그림 12> Douban-book 데이터 집합의 알고리즘 별 성능 산점도 (전체)

<표 5> 데이터 집합 및 알고리즘 별 기존 방법과 제안 방법 적용에 따른 베스트 학습 속도 및 추천 성능 개선 수치

FilmTrust				
Algorithm	Weight	Epoch	Recall@20	NDCG@20
BUIR	-	291	0.83774	0.60724
BUIR_EW	512	266(-8.59%)	0.85103(+1.59%)	0.61154(+0.71%)
MixGCF	-	173	0.83204	0.61152
MixGCF_EW	2	159(-8.09%)	0.83720(+0.62%)	0.62950(+2.94%)
LightGCN	-	276	0.85376	0.61566
LightGCN_EW	4	161(-41.67%)	0.86316(+1.1%)	0.61970(+0.66%)
SGL	-	121	0.85761	0.62903
SGL_W	2	86(-28.93%)	0.86001(+0.28%)	0.62691(+0.34%)
SimGCL	-	1	0.31356	0.16678
SimGCL_W	32	10	0.85551(+172.84%)	0.63114(+278.43%)
XSimGCL	-	1	0.65846	0.48539
XSimGCL_EW	4	24	0.86259(+31%)	0.64195(+32.25%)
MF	-	296	0.84800	0.61690
SelfCF	-	7	0.04075	0.02646
DirectAU	-	54	0.83840	0.60760
MovieLens-1M				
	Weight	Epoch	Recall@20	NDCG@20
BUIR	-	298	0.19115	0.22593
BUIR_E	-	299(+0.34%)	0.20169(+5.51%)	0.23589(+4.41%)
MixGCF	-	2	0.14913	0.17538
MixGCF_EW	8	14(+600%)	0.23060(+54.63%)	0.26500(+51.1%)
LightGCN	-	261	0.27196	0.30274
LightGCN_EW	16	31(-88.12%)	0.27767(+2.1%)	0.31078(+2.66%)
SGL	-	42	0.27369	0.30960
SGL_W	8	28(-33.33%)	0.28670(+4.75%)	0.32120(+3.75%)
SimGCL	-	217	0.25545	0.28191
SimGCL_W	16	5(-97.7%)	0.27620(+8.12%)	0.30900(+9.61%)
XSimGCL	-	61	0.28515	0.32007
XSimGCL_W	2	23(-62.3%)	0.28430(-0.3%)	0.31830(-0.55%)
MF	-	46	0.24900	0.28310
SelfCF	-	8	0.10870	0.10300
DirectAU	-	300	0.1924	0.2244
Yelp2018				
	Weight	Epoch	Recall@20	NDCG@20
BUIR	-	290	0.04371	0.03557
BUIR_E	-	79(-72.76%)	0.04578(+4.74%)	0.03718(+4.53%)
MixGCF	-	96	0.07010	0.05752



MixGCF_W	2	85(-11.46%)	0.06931(-1.13%)	0.05697(-0.96%)
LightGCN	-	231	0.05949	0.04887
LightGCN_EW	8	36(-84.42%)	0.06312(+6.1%)	0.05148(+5.34%)
SGL	-	27	0.06791	0.05585
SGL_W	2	23(-14.81%)	0.06978(+2.75%)	0.05716(+2.35%)
SimGCL	-	18	0.07248	0.05976
SimGCL_W	2	8(-55.56%)	0.07205(-0.59%)	0.05955(-0.35%)
XSimGCL	-	14	0.07277	0.06002
XSimGCL_W	2	9(-35.71%)	0.07242(-0.48%)	0.05949(-0.88%)
MF	-	101	0.04998	0.04055
SelfCF	-	47	0.07017	0.05807
DirectAU	-	290	0.04414	0.03601
Douban-book				
	Weight	Epoch	Recall@20	NDCG@20
BUIR	-	298	0.08945	0.07097
BUIR_W	2048	294(-1.34%)	0.10467(+17.02%)	0.08412(+18.53%)
MixGCF	-	145	0.17641	0.16265
MixGCF_W	2	133(-8.28%)	0.17440(-1.14%)	0.15820(-2.74%)
LightGCN	-	266	0.14783	0.12508
LightGCN_W	16	26(-90.23%)	0.15127(+2.33%)	0.12867(+2.87%)
SGL	-	46	0.17322	0.15225
SGL_W	2	38(-17.39%)	0.17570(+1.43%)	0.15350(+0.82%)
SimGCL	-	51	0.16677	0.14781
SimGCL_W	2	28(-45.1%)	0.17041(+2.18%)	0.14940(+1.08%)
XSimGCL	-	29	0.17632	0.15397
XSimGCL_W	2	11(-62.07%)	0.17709(+0.44%)	0.15444(+0.31%)
MF	-	66	0.12470	0.10390
SelfCF	-	23	0.13670	0.11890
DirectAU	-	297	0.08509	0.06709

그림 11, 12, 13, 14는 임베딩 값과 데이터 집합을 통한 차원 감소 기법(Dimensionality Reduction Techniques, DRT)을 통한 분석 결과이다. 이러한 분석에는 데이터 집합 및 알고리즘 별로 주성분 분석(Principle Component Analysis, PCA)과 가우시안 커널 밀도 추정(Gaussian kernel density estimation, GKDE) 시각화를 수행하여 임베딩의 분포와 밀도를 분석하였다. 또한 시각화를 위해 사용자와 아이템 항목을 인기도로 순위를 매기는 데이터 샘플링 과정을 진행한 뒤, 상위 5%의 사용자 및 아이템 그룹에서 500개의 hot 사용자들(blue), hot 아이템들(green) 항목을 무작위로 샘플링하였고, 하위

80%의 사용자 및 아이템 그룹에서 500개의 cold 사용자들(red), cold 아이템들(orange) 항목을 무작위로 샘플링하였다. 샘플링 과정을 통해 학습된 표현을 t-SNE를 사용하여 2차원 공간으로 매핑하여, 2차원 공간에 매핑된 표현을 통해 PCA 시각화를 진행하였다. 또한 GKDE 시각화 결과는 아래의 나타내었으며 각 점 (x,y)에 대해 $\text{atan}(y/x)$의 GKDE를 사용하여 시각화하였다.

이러한 2개의 시각화 결과를 통해 다음과 같은 인사이트를 얻을 수 있다. 첫 번째 분리력(Separability)은 서로 다른 카테고리나 그룹의 데이터 포인트들이 잘 구분되어 있는지를 나타낸다. 예를 들어, 그림에서 hot 사용자들과 cold 사용자들의 데이터 포인트들이 서로 명확하게 구분된 영역에 자리 잡고 있다면, 해당 알고리즘이 각 그룹을 잘 구분하고 있음을 의미한다.

두 번째 밀도(Density)는 GKDE과 같은 밀도 추정 그래프를 통해 데이터 포인트들의 밀도를 확인하며, 높은 밀도의 영역은 데이터 포인트들이 집중적으로 위치한 지역을 나타내며, 이는 해당 영역에 중요한 패턴이나 특성이 있을 수 있음을 시사한다.

세 번째 분포의 균일성(Uniformity)은 데이터 포인트들이 전체 공간에 균일하게 분포되어 있는지 의미한다. 균일한 분포는 알고리즘이 데이터의 다양한 특성을 잘 반영하고 있음을 나타낸다.

네 번째 군집화(Clustering)는 유사한 특성을 가진 데이터 포인트들이 군집을 형성하고 있는지를 확인 가능하며, 잘 형성된 군집은 알고리즘이 데이터의 특정 패턴이나 구조를 잘 파악하고 있음을 의미한다.

다섯 번째 노이즈(Noise)와 이상치(Outlier)는 데이터 포인트들 중에서 일반적인 분포에서 벗어난 위치에 있는 이상치나 노이즈가 있는지 확인 가능하며, 이러한 점들은 알고리즘이 일부 데이터에 대해 잘못된 표현을 학습했을 가능성이 있음을 나타낸다.

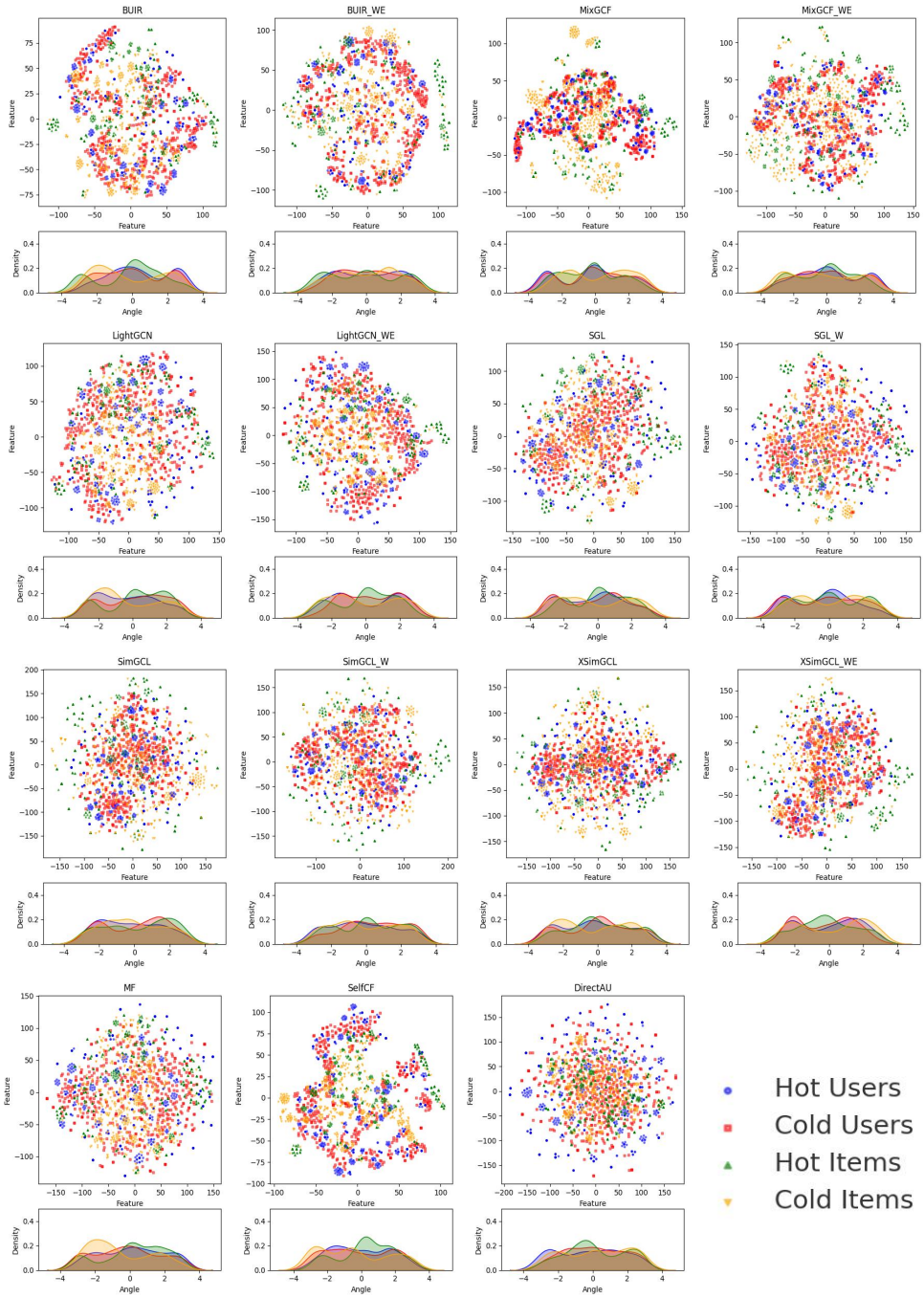
여섯 번째 알고리즘의 안정성(Stability)은 동일한 데이터 집합에 대해 여러 번 학습을 진행했을 때, 학습된 표현의 분포가 안정적인지 확인할 수 있으며 일관된 결과를 보이는 알고리즘은 더욱 신뢰할 수 있다.

이러한 PCA, GKDE 시각화를 통해서 데이터의 표현 분포의 시각적 분석이 가능하며, 알고리즘이 얼마나 잘 작동하는지 평가할 수 있으며 기존 방법과

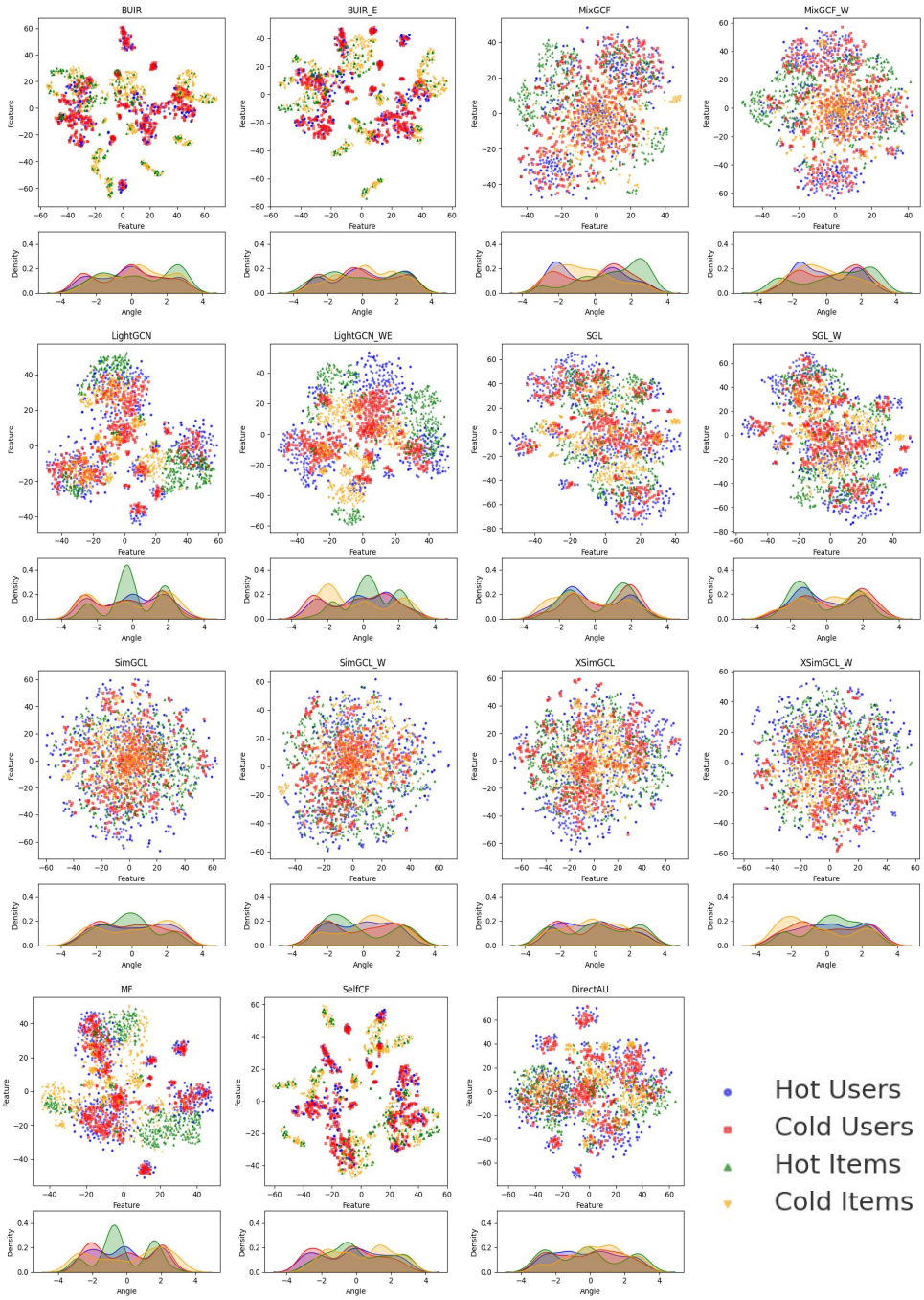
제안 방법의 차이점을 시각적으로 확인할 수 있다.

그림 11, 12, 13, 14를 통해 데이터 집합, 기본 알고리즘, 적용 방법의 가장 좋은 결과를 함께 나타내었으며, 시각화 결과 기존 알고리즘의 경우, PCA에서 여러 개의 군집으로 나타났으나, 각 군집 간의 경계는 상대적으로 뚜렷하지 않다. GKDE에서도 비슷한 양상을 보였으며, 주요 분포(distribution)를 나타내는 peak가 존재하며 이러한 현상은 한 곳에 특징값들이 밀집되어 있기 때문이다. 반면, 제안 방법을 적용한 알고리즘의 임베딩 결과에서는 PCA에서 비교적 뚜렷한 군집화 결과와 넓은 분산을 확인할 수 있다. 이는 데이터의 다양성과 특징을 비교적 포착하고 있음을 의미한다. GKDE에서도 부드러운 곡선이 관찰되었으며, 이상치도 거의 관찰되지 않는다.

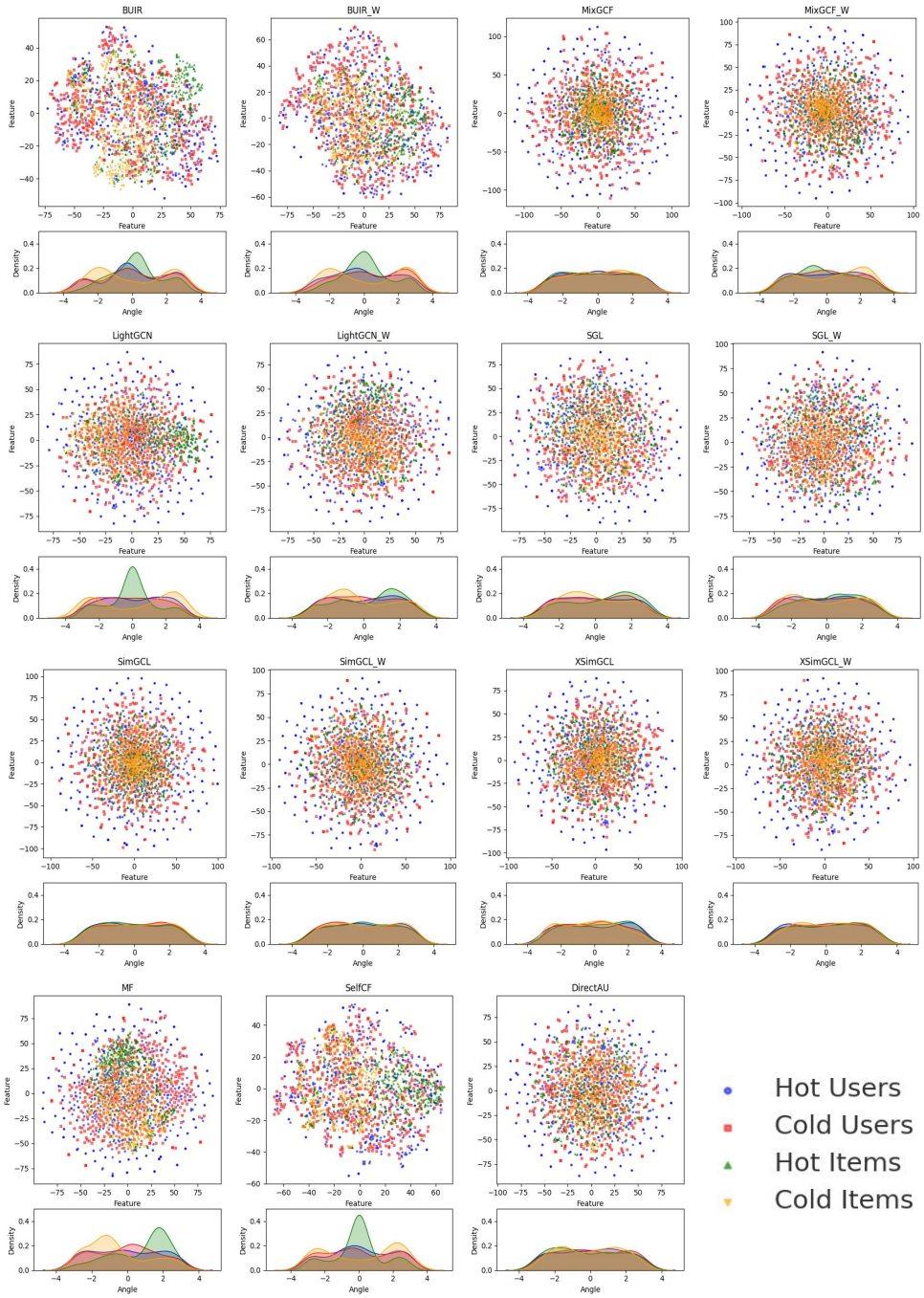
결론적으로, 임베딩에 스칼라 곱이 적용되면서 임베딩의 평균과 중앙값이 중심 값 주변으로 더 집중되게 변했음을 확인하였으며, 제안하는 WF 기법과 Egress 초기화 기법을 통한 알고리즘의 임베딩이 기존 알고리즘에 비해 특성(property)을 잘 포착한다는 것을 시사하며, 제안 방법들을 통한 성능 향상 원인을 차원 감소 기법을 통해 확인했다.



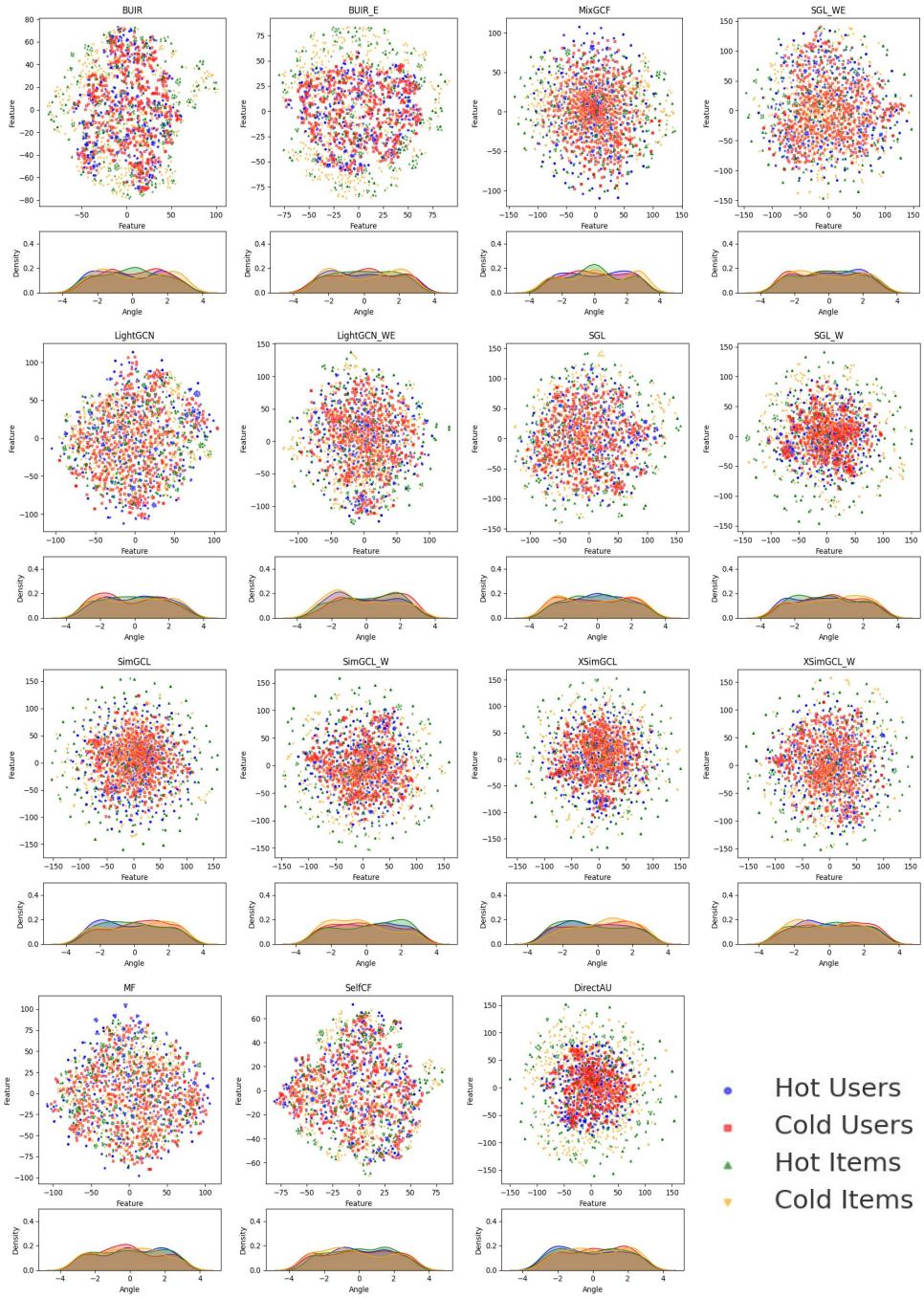
<그림 13> FilmTrust 데이터 집합의 제안 방법에 따른
기존 알고리즘과 개선 알고리즘의 PCA, GKDE 시각화 결과



<그림 14> Yelp2018 데이터 집합의 제안 방법에 따른
기존 알고리즘과 개선 알고리즘의 PCA, GKDE 시각화 결과



<그림 15> Douban-book 데이터 집합의 제안 방법에 따른
기존 알고리즘과 개선 알고리즘의 PCA, GKDE 시각화 결과



<그림 16> MovieLens-1M 데이터 집합의 제안 방법에 따른 기존 알고리즘과 개선 알고리즘의 PCA, GKDE 시각화 결과

제 6 장 결 론

본 연구에서는 추천 시스템에서 사용되는 GCN 알고리즘의 학습 과정 및 추천 성능을 개선하기 위한 Egress 초기화 기법과 WF 기법을 제안했다. Egress 초기화 기법은 기존의 저명하게 사용되던 Xavier Uniform 초기화 기법에 비해 더 넓은 범위의 초기화 범위를 가지며, 해당 범위를 활용하여 각 노드의 정보가 더 넓은 값 범위를 통해 전파되어 기존 그래프 구조에서보다 더 다양한 정보를 캡처가 가능하기에 알고리즘의 정확도의 개선이 가능하였으며, 더 넓은 범위를 통해 알고리즘의 비용 함수가 전역 최적해에 도달하는 데 걸리는 시간을 단축하여 학습 속도 저하 문제를 완화하였다. 또한 WF 기법은 GCN 알고리즘에서 각 Layer의 임베딩 값을 계산할 때 이전 임베딩 Layer의 임베딩 값에 가중치를 곱하여 다음 Layer로 forwarding한다. 그 결과 neighbor가 많은 노드의 임베딩 값에 더욱 집중하여 적은 수의 홉 Layer를 학습한 GCN 알고리즘에서도 더욱 깊은 홉 Layer를 학습한 효과를 가져올 수 있다. 즉 GCN 알고리즘에서 홉 Layer를 깊이 계산하기 위한 연산을 수행하지 않으면서도 사용자와 아이템 간의 깊이 있는 관계를 유추해 낼 수 있는 장점을 가지게 되었다. 따라서 WF 기법은 기존의 GCN 알고리즘이 가지는 임베딩 값 소실 문제를 완화할 수 있는 장점을 가지게 되어, 전체적인 학습 속도 및 추천 정확도의 개선을 보였다.

제안한 Egress 초기화 기법과 WF 기법의 성능을 확인하기 위해 본 논문에서는 LightGCN을 기본 구조로 사용하며, 추천 시스템 분야에서 우수한 성능을 보이는 GCN 알고리즘인 LightGCN, BUIR, MixGCF, SGL, SimGCL, XSimGCL에 Egress 초기화 기법과 WF 기법을 적용하고 MovieLens 데이터 집합, Yelp2018 데이터 집합, douban-book 데이터 집합, FilmTrust 데이터 집합을 통해 그 성능을 평가하였다. 성능 평가에는 적용 여부에 따른 4가지 경우의 수를 고려하여 수행되었다.

FilmTrust 데이터 집합에서 BUIR_EW은 Epoch를 291에서 266로 줄여 8.59% 감소시키고, Recall@20은 1.59% 증가, NDCG@20은 0.71% 증가시켰습니다. MixGCF_EW은 Epoch를 173에서 159로 줄여 8.09% 감소시키고,

Recall@20은 0.62% 증가, NDCG@20은 2.94% 증가시켰습니다. LightGCN_EW은 Epoch를 276에서 161로 줄여 41.67% 감소시키고, Recall@20은 1.1% 증가, NDCG@20은 0.66% 증가시켰습니다. SGL_W은 Epoch를 121에서 86로 줄여 28.93% 감소시키고, Recall@20은 0.28% 증가, NDCG@20은 0.34% 증가시켰습니다. SimGCL_W은 Epoch를 1에서 10로 증가시키고, Recall@20은 172.84% 증가, NDCG@20은 278.43% 증가시켰습니다. XSimGCL_EW은 Epoch를 1에서 24로 증가시키고, Recall@20은 31% 증가, NDCG@20은 32.25% 증가시켰습니다. 해당 데이터 집합에서의 SimGCL 알고리즘과 XSimGCL은 Epoch 1에서 학습이 제대로 이루어지지 않는 문제를 제안된 Egress 초기화, WF 기법을 통해 성능을 비약적으로 상승시켰다.

MovieLens-1M 데이터 집합에서 BUIR_E은 Epoch를 298에서 299로 약간 증가시키고, Recall@20은 5.51% 증가, NDCG@20은 4.41% 증가시켰습니다. MixGCF_EW은 Epoch를 2에서 14로 크게 증가시켜 600% 증가하였지만, Recall@20은 54.63% 증가, NDCG@20은 51.1% 증가시켰습니다. LightGCN_EW은 Epoch를 261에서 31로 크게 줄여 88.12% 감소시키고, Recall@20은 2.1% 증가, NDCG@20은 2.66% 증가시켰습니다. SGL_W은 Epoch를 42에서 28로 줄여 33.33% 감소시키고, Recall@20은 4.75% 증가, NDCG@20은 3.75% 증가시켰습니다. SimGCL_W은 Epoch를 217에서 5로 크게 줄여 97.7% 감소시키고, Recall@20은 8.12% 증가, NDCG@20은 9.61% 증가시켰습니다. XSimGCL_W은 Epoch를 61에서 23로 줄여 62.3% 감소시키고, Recall@20은 0.3% 감소, NDCG@20은 0.55% 감소시켰습니다.

Yelp2018 데이터 집합에서 BUIR_E은 Epoch를 290에서 79로 크게 줄여 72.76% 감소시키고, Recall@20은 4.74% 증가, NDCG@20은 4.53% 증가시켰습니다. MixGCF_W은 Epoch를 96에서 85로 줄여 11.46% 감소시키고, Recall@20은 1.13% 감소, NDCG@20은 0.96% 감소시켰습니다. LightGCN_EW은 Epoch를 231에서 36로 크게 줄여 84.42% 감소시키고, Recall@20은 6.1% 증가, NDCG@20은 5.34% 증가시켰습니다. SGL_W은 Epoch를 27에서 23로 줄여 14.81% 감소시키고, Recall@20은 2.75% 증가, NDCG@20은 2.35% 증가시켰습니다. SimGCL_W은 Epoch를 18에서 8로 줄여

55.56% 감소시키고, Recall@20은 0.59% 감소, NDCG@20은 0.35% 감소시켰습니다. XSimGCL_W은 Epoch를 14에서 9로 줄여 35.71% 감소시키고, Recall@20은 0.48% 감소, NDCG@20은 0.88% 감소시켰습니다.

Douban-book 데이터 집합에서 BUIR_W은 Epoch를 298에서 294로 줄여 1.34% 감소시키고, Recall@20은 17.02% 증가, NDCG@20은 18.53% 증가시켰습니다. MixGCF_W은 Epoch를 145에서 133로 줄여 8.28% 감소시키고, Recall@20은 1.14% 감소, NDCG@20은 2.74% 감소시켰습니다. LightGCN_W은 Epoch를 266에서 26로 크게 줄여 90.23% 감소시키고, Recall@20은 2.33% 증가, NDCG@20은 2.87% 증가시켰습니다. SGL_W은 Epoch를 46에서 38로 줄여 17.39% 감소시키고, Recall@20은 1.43% 증가, NDCG@20은 0.82% 증가시켰습니다. SimGCL_W은 Epoch를 51에서 28로 줄여 45.1% 감소시키고, Recall@20은 2.18% 증가, NDCG@20은 1.08% 증가시켰습니다. XSimGCL_W은 Epoch를 29에서 11로 줄여 62.07% 감소시키고, Recall@20은 0.44% 증가, NDCG@20은 0.31% 증가시켰습니다.

본 논문에서 선행적으로 연구되었던 알고리즘은 모두 학습 과정의 Layer combination(weighted sum)을 진행하여 나온 최종 임베딩을 사용하여 추천을 진행한다. 하지만 해당 실험 결과를 통해 모든 Layer 임베딩이 필요하지 않을 수도 있다는 가능성을 확인하였으며, 따라서 추후 연구로는 GCN 알고리즘의 layer combination(weighted sum)의 구조를 개선하기 위한 연구와 임베딩에 적용되지 않은 weight 에 따른 연관성에 대한 분석을 수행할 예정이다.

참고문헌

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using Collaborative Filtering to Weave an Information Tapestry,” *Communications of the ACM*, vol. 35, no.12, pp. 61–70, 1992.

[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: an open architecture for collaborative filtering of netnews,” in *Proc. CSCW*, Chapel Hill, NC, USA, pp. 175–186, 1994.

[3] N. Friedman, D. Geiger and M. Goldszmidt, “Bayesian Network Classifiers,” *Machine Learning*, vol. 29, pp. 131–163, 1997.

[4] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, “Item-based Collaborative Filtering Recommendation Algorithms,” in *Proc. WWW*, Hong Kong, pp. 285–295, 2001.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “An Algorithmic Framework for Performing Collaborative Filtering,” in *Proc. SIGIR*, Berkley, CA, USA, pp. 230–237, 1999.

[6] M. Goyani and N. Chaurasiya, “A review of movie Recommendation system: Limitations, Survey and Challenges,” *ELCVIA: electronic letters on computer vision and image analysis*, vol. 19, no.3, pp. 18–37, 2020.

[7] H. Wang, Z. Le and X. Gong, "Recommendation System based on Heterogeneous Feature: A Survey," *IEEE Access*, vol. 8, pp. 170779–170793, 2020.

[8] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton *et al.*, “Graph Convolutional Neural Networks for Web-Scale Recommender Systems,” in *Proc. KDD*, London, United Kingdom, pp. 974–983, 2018.

[9] S. Zhang, H. Tong, J. Xu and R. Maciejewski, “Graph convolutional networks: a comprehensive review.” *Computational Social Networks*, vol. 6, 2019.

[10] P. Chen, J. Zhao and X. Yu, “LighterKGCN: A Recommender System algorithm based on Bi-layer Graph Convolutional Networks,” *Journal of Internet Technology*, vol. 23, no.3, pp. 621–629, 2022.

[11] Y. Koren, R. Bell and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems,” in *Computer*, vol. 42, no. 8, pp. 30-37, 2009.

[12] C. Wang, Y. Yu, W. Ma, M. Zhang, C. Chen, *et al.*, “Towards Representation Alignment and Uniformity in Collaborative Filtering,” in *Proc. KDD*, New York, NY, USA, pp. 1816–1825, 2022.

[13] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, “The Graph Neural Network Algorithm,” *IEEE Transactions on Neural Networks*, vol. 20, no.1, pp. 61–80, Jan. 2009.

[14] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang *et al.*, “A Comprehensive Survey on Graph Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no.1, pp. 4–24, 2021.

[15] K. Xu, W. Hu, J. Leskovec and S. Jegelka, “How Powerful are Graph Neural Networks?” in *Proc. ICLR*, New Orleans, Louisiana, United States, 2019.

[16] J. You, J. Leskovec, K. He, S. Xie, “Graph Structure of Neural Networks,” in *Proc. ICML,* Vienna, Austria, pp. 10881–10891, 2020.

[17] X. Wang, X. He, M. Wang, F. Feng and T. S. Chua, “Neural Graph Collaborative Filtering,” in *Proc. SIGIR*, Paris, France, pp. 165–174, 2019.

[18] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang *et al.*, “LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation,” in *Proc. SIGIR*, China, pp. 639–648, 2020.

[19] D. Lee, S. Kang, H. Ju, C. Park and H. Yu, “Bootstrapping user and item Representations for One-Class Collaborative Filtering,” in *Proc. SIGIR*, Canada, 2021.

[20] X. Zhou, A. Sun, Y. Liu, J. Zhang, and C. Miao, "SelfCF: A Simple Framework for Self-supervised Collaborative Filtering," *ACM Transactions on Recommender Systems*, vol. 1, issue. 2, pp. 1-25, 2023.

[21] T. Huang, Y. Dong, M. Ding, Z. Yang, W. Feng *et al.*, "MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems," in *Proc. KDD*, Singapore, 2021.

[22] J. Wu, X. Wang, F. Feng, X. He, L. Chen *et al.,* “Self-supervised Graph Learning for Recommendation,” in *Proc. SIGIR*, Canada, pp. 726–735, 2021.

[23] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui *et al.*, “Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation,” in *Proc. SIGIR*, Madrid, Spain, pp. 1294–1303, 2022.

[24] J. Yu, X. Xia, T. Chen, L. Cui, N. Q. V. Hung and H. Yin, “XSimGCL: Towards Extremely Simple Graph Contrastive Learning for Recommendation,” in *IEEE Transactions on Knowledge and Data Engineering*, pp. 1-14, 2023.

[25] J. Yu, H. Yin, X. Xia, T. Chen, J. Li and Z. Huang, “Self-Supervised Learning for Recommender Systems: A Survey,” in *IEEE Transactions on Knowledge and Data Engineering*, pp. 1-20, 2023.

[26] F. M. Harper and J. A. Konstan, “The MovieLens Datasets: History and Context,” *ACM Transactions on Interactive Intelligent Systems*, vol 5, no.19. pp. 1–19, 2015.

부록 1. 평가 함수의 수학적 표현

$$Recall@K = \frac{1}{|U|}\sum_{u \in U}\frac{|\hat{R}(u) \cap R(u)|}{|R(u)|} \tag{1}$$

$$NDCG@K = \frac{1}{|U|}\sum_{u \in U}\left(\frac{1}{\sum_{i=1}^{\min(|R(u)|,K)}\frac{1}{\log_2(i+1)}}\sum_{i=1}^{K}\delta(i \in R(u))\frac{1}{\log_2(i+1)}\right) \tag{2}$$

부록 2. 기존 알고리즘과 개선된 제안 방법의 임베딩 통계치

<표 6> FilmTrust 데이터 집합에서의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 임베딩 값의 통계치

		BUIR	BUIR WE(512)	MixGCF	MixGCF WE(2)
사용자 임베딩	평균	3.32e-02	-3.02e+00	4.27e-03	4.48e-03
	표준 편차	1.74e-01	1.56e+01	1.79e-01	1.32e-01
	표준 오차	5.60e-04	5.02e-02	5.78e-04	4.25e-04
	분산	3.01e-02	2.42e+02	3.20e-02	1.73e-02
	중앙값	7.64e-02	-2.87e+00	1.70e-03	3.65e-03
아이템 임베딩	평균	1.18e-02	-9.37e-01	-1.17e-04	7.14e-04
	표준 편차	1.56e-01	1.16e+01	1.45e-01	1.31e-01
	표준 오차	4.34e-04	3.20e-02	4.03e-04	3.63e-04
	분산	2.45e-02	1.33e+02	2.11e-02	1.72e-02
	중앙값	1.12e-02	-5.16e-01	-3.37e-05	4.32e-05
		LightGCN	LightGCN WE(4)	SGL	SGL W(2)
사용자 임베딩	평균	1.23e-02	1.56e-04	-8.74e-05	1.48e-03
	표준 편차	2.54e-01	3.09e-01	1.90e-01	2.07e-01
	표준 오차	8.22e-04	9.97e-04	6.14e-04	6.69e-04
	분산	6.48e-02	9.53e-02	3.62e-02	4.30e-02
	중앙값	2.03e-02	1.67e-03	-1.89e-03	6.90e-03
아이템 임베딩	평균	-4.89e-03	-2.79e-04	-4.06e-04	9.69e-05
	표준 편차	2.41e-01	2.56e-01	1.75e-01	1.85e-01
	표준 오차	6.67e-04	7.11e-04	4.84e-04	5.14e-04
	분산	5.79e-02	6.58e-02	3.05e-02	3.44e-02
	중앙값	-9.68e-03	-9.20e-04	-7.51e-04	1.16e-04
		SimGCL	SimGCL W(32)	XSimGCL	XSimGCL WE(4)
사용자 임베딩	평균	-9.59e-06	1.52e-02	-1.60e-05	-1.53e-03
	표준 편차	4.33e-03	3.29e-01	4.85e-03	1.11e-01
	표준 오차	1.40e-05	1.06e-03	1.57e-05	3.58e-04
	분산	1.87e-05	1.08e-01	2.36e-05	1.23e-02
	중앙값	-9.03e-06	1.20e-02	2.90e-05	-2.36e-03
아이템 임베딩	평균	-2.32e-05	2.10e-03	8.66e-06	-1.46e-04
	표준 편차	3.75e-03	2.74e-01	4.24e-03	1.06e-01
	표준 오차	1.04e-05	7.59e-04	1.18e-05	2.93e-04
	분산	1.40e-05	7.49e-02	1.80e-05	1.12e-02

	중앙값	1.46e-05	-9.67e-04	5.15e-05	5.72e-04
		MF	SelfCF	DirectAU	
사용자 임베딩	평균	4.74e-02	2.00e-02	2.40e-05	
	표준 편차	3.65e-01	2.92e-01	1.32e-02	
	표준 오차	1.18e-03	9.44e-04	4.26e-05	
	분산	1.33e-01	8.55e-02	1.74e-04	
	중앙값	1.57e-01	1.30e-01	-4.20e-06	
아이템 임베딩	평균	-2.73e-02	9.04e-03	5.08e-05	
	표준 편차	3.20e-01	2.53e-01	1.15e-02	
	표준 오차	8.86e-04	7.02e-04	3.19e-05	
	분산	1.02e-01	6.42e-02	1.32e-04	
	중앙값	-2.71e-02	7.98e-03	-2.38e-06	

<표 7> Yelp2018 데이터 집합에서의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 임베딩 값의 통계치

		BUIR	BUIR E	MixGCF	MixGCF W(2)
사용자 임베딩	평균	-6.31e-02	4.54e-02	-1.11e-03	1.23e-03
	표준 편차	1.50e+00	1.64e+00	3.06e-01	2.26e-01
	표준 오차	1.05e-03	1.15e-03	2.15e-04	1.59e-04
	분산	2.25e+00	2.70e+00	9.33e-02	5.09e-02
	중앙값	-4.61e-02	4.37e-02	-5.14e-04	3.11e-03
아이템 임베딩	평균	-5.82e-02	3.48e-02	1.76e-03	8.49e-04
	표준 편차	1.38e+00	1.51e+00	2.56e-01	1.97e-01
	표준 오차	8.81e-04	9.65e-04	1.64e-04	1.26e-04
	분산	1.89e+00	2.27e+00	6.53e-02	3.90e-02
	중앙값	-4.40e-02	3.29e-02	4.08e-03	2.52e-03
		LightGCN	LightGCN EW(8)	SGL	SGL W(2)
사용자 임베딩	평균	-5.57e-03	5.99e-03	1.08e-03	-2.71e-03
	표준 편차	4.94e-01	4.61e-01	3.59e-01	3.68e-01
	표준 오차	3.47e-04	3.24e-04	2.52e-04	2.58e-04
	분산	2.44e-01	2.13e-01	1.29e-01	1.35e-01
	중앙값	-8.12e-03	1.03e-02	3.15e-03	-4.12e-03
아이템 임베딩	평균	-2.73e-04	-5.02e-03	-1.52e-04	3.70e-03
	표준 편차	4.17e-01	3.95e-01	3.21e-01	3.33e-01
	표준 오차	2.67e-04	2.53e-04	2.06e-04	2.13e-04
	분산	1.74e-01	1.56e-01	1.03e-01	1.11e-01
	중앙값	9.80e-04	2.00e-04	4.75e-04	4.52e-03
		SimGCL	SimGCL W(2)	XSimGCL	XSimGCL W(2)

사용자 임베딩	평균	-7.28e-04	7.58e-04	5.83e-04	4.93e-06
	표준 편차	2.78e-01	2.69e-01	2.74e-01	2.83e-01
	표준 오차	1.95e-04	1.89e-04	1.92e-04	1.99e-04
	분산	7.71e-02	7.21e-02	7.48e-02	8.01e-02
	중앙값	-8.41e-04	7.97e-05	1.92e-03	-1.90e-03
아이템 임베딩	평균	-6.32e-04	7.19e-04	-2.14e-04	1.81e-03
	표준 편차	2.34e-01	2.28e-01	2.40e-01	2.49e-01
	표준 오차	1.50e-04	1.46e-04	1.54e-04	1.59e-04
	분산	5.46e-02	5.22e-02	5.74e-02	6.18e-02
	중앙값	2.63e-04	-5.99e-04	8.27e-05	2.77e-03
		MF	SelfCF	DirectAU	
사용자 임베딩	평균	1.44e-02	-2.61e-02	1.63e-04	
	표준 편차	6.54e-01	1.77e+00	7.07e-02	
	표준 오차	4.60e-04	1.25e-03	4.97e-05	
	분산	4.28e-01	3.15e+00	5.00e-03	
	중앙값	1.72e-02	-6.57e-02	2.25e-04	
아이템 임베딩	평균	-5.31e-03	-4.93e-03	-1.63e-04	
	표준 편차	4.16e-01	1.62e+00	6.61e-02	
	표준 오차	2.66e-04	1.04e-03	4.24e-05	
	분산	1.73e-01	2.63e+00	4.37e-03	
	중앙값	-3.18e-03	-2.16e-02	-5.85e-04	

<표 8> Douban-book 데이터 집합에서의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 임베딩 값의 통계치

		BUIR	BUIR W(2048)	MixGCF	MixGCF W(2)
사용자 임베딩	평균	-1.67e-02	-5.19e-02	2.56e-03	3.22e-04
	표준 편차	6.78e-01	4.78e-01	2.39e-01	2.28e-01
	표준 오차	7.54e-04	5.32e-04	2.66e-04	2.53e-04
	분산	4.59e-01	2.29e-01	5.71e-02	5.18e-02
	중앙값	-1.90e-02	-2.62e-02	9.95e-04	9.33e-04
아이템 임베딩	평균	-1.35e-02	-3.83e-02	-6.95e-04	-5.14e-05
	표준 편차	5.50e-01	3.82e-01	1.66e-01	1.66e-01
	표준 오차	4.62e-04	3.20e-04	1.39e-04	1.40e-04
	분산	3.03e-01	1.46e-01	2.75e-02	2.77e-02
	중앙값	-1.29e-02	-2.52e-02	-8.79e-04	1.39e-05
		LightGCN	LightGCN W(16)	SGL	SGL W(2)
사용자 임베딩	평균	-2.79e-04	1.21e-02	6.55e-03	-6.70e-03
	표준 편차	5.17e-01	5.11e-01	3.23e-01	3.49e-01
	표준 오차	5.75e-04	5.68e-04	3.59e-04	3.88e-04

	분산	2.68e-01	2.61e-01	1.04e-01	1.22e-01
	중앙값	4.91e-04	-9.61e-03	6.83e-03	-6.99e-03
아이템 임베딩	평균	-1.20e-03	-4.94e-03	-7.72e-05	-1.96e-03
	표준 편차	4.05e-01	3.12e-01	2.40e-01	2.64e-01
	표준 오차	3.40e-04	2.62e-04	2.02e-04	2.21e-04
	분산	1.64e-01	9.74e-02	5.77e-02	6.96e-02
	중앙값	-3.19e-03	-6.23e-03	-2.03e-03	-1.96e-03
		SimGCL	SimGCL W(2)	XSimGCL	XSimGCL W(2)
사용자 임베딩	평균	1.25e-03	6.25e-04	-1.18e-03	-2.51e-04
	표준 편차	2.97e-01	2.86e-01	2.26e-01	1.92e-01
	표준 오차	3.30e-04	3.18e-04	2.52e-04	2.13e-04
	분산	8.80e-02	8.17e-02	5.12e-02	3.68e-02
	중앙값	2.08e-03	1.02e-03	-1.89e-03	-7.48e-04
아이템 임베딩	평균	-5.77e-04	-1.43e-03	-4.56e-04	4.31e-04
	표준 편차	1.82e-01	1.80e-01	1.47e-01	1.28e-01
	표준 오차	1.53e-04	1.51e-04	1.23e-04	1.07e-04
	분산	3.31e-02	3.24e-02	2.16e-02	1.63e-02
	중앙값	-1.13e-03	-2.25e-03	-5.93e-04	7.77e-04
		MF	SelfCF	DirectAU	
사용자 임베딩	평균	-1.82e-02	9.09e-02	-2.42e-04	
	표준 편차	4.83e-01	7.28e-01	2.33e-02	
	표준 오차	5.37e-04	8.09e-04	2.59e-05	
	분산	2.34e-01	5.30e-01	5.42e-04	
	중앙값	-2.35e-02	1.70e-01	-2.60e-04	
아이템 임베딩	평균	6.69e-03	7.06e-02	-1.51e-04	
	표준 편차	3.72e-01	5.86e-01	1.94e-02	
	표준 오차	3.12e-04	4.92e-04	1.63e-05	
	분산	1.39e-01	3.44e-01	3.78e-04	
	중앙값	4.52e-03	1.10e-01	-1.73e-04	

<표 9> MovieLens-1M 데이터 집합의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 임베딩 값의 통계치

		BUIR	BUIR E	MixGCF	MixGCF WE(8)
사용자 임베딩	평균	9.08e-02	-1.56e-02	4.74e-06	4.85e-06
	표준 편차	6.02e-01	6.38e-01	1.10e-03	1.18e-03
	표준 오차	9.69e-04	1.03e-03	1.77e-06	1.91e-06
	분산	3.63e-01	4.08e-01	1.21e-06	1.40e-06
	중앙값	1.02e-01	-2.94e-02	5.57e-06	7.13e-06
아이템	평균	8.32e-02	-2.38e-02	-9.72e-07	8.38e-07

임베딩	표준 편차	7.08e-01	7.53e-01	8.25e-04	1.17e-03
	표준 오차	1.50e-03	1.59e-03	1.74e-06	2.47e-06
	분산	5.02e-01	5.66e-01	6.80e-07	1.36e-06
	중앙값	6.79e-02	-4.20e-02	-9.59e-07	-8.43e-07
		LightGCN	LightGCN EW(16)	SGL	SGL W(8)
사용자 임베딩	평균	4.93e-03	-6.27e-03	7.00e-03	5.16e-03
	표준 편차	3.72e-01	3.06e-01	2.32e-01	2.72e-01
	표준 오차	5.98e-04	4.93e-04	3.73e-04	4.38e-04
	분산	1.38e-01	9.39e-02	5.38e-02	7.40e-02
	중앙값	6.57e-03	-1.03e-02	8.71e-03	4.21e-03
아이템 임베딩	평균	3.82e-03	-2.51e-02	5.25e-04	-2.38e-03
	표준 편차	4.78e-01	4.60e-01	2.72e-01	3.85e-01
	표준 오차	1.01e-03	9.74e-04	5.75e-04	8.13e-04
	분산	2.28e-01	2.12e-01	7.40e-02	1.48e-01
	중앙값	4.15e-03	-1.95e-02	-1.94e-03	-5.38e-03
		SimGCL	SimGCL W(16)	XSimGCL	XSimGCL (2)
사용자 임베딩	평균	9.64e-04	-1.28e-03	2.15e-03	-1.60e-03
	표준 편차	2.52e-01	2.20e-01	1.67e-01	1.45e-01
	표준 오차	4.06e-04	3.55e-04	2.69e-04	2.34e-04
	분산	6.35e-02	4.86e-02	2.80e-02	2.11e-02
	중앙값	1.01e-03	-4.33e-04	3.37e-03	-1.85e-03
아이템 임베딩	평균	1.74e-03	-3.05e-04	8.55e-04	-1.04e-04
	표준 편차	3.76e-01	3.17e-01	2.60e-01	2.16e-01
	표준 오차	7.94e-04	6.70e-04	5.50e-04	4.56e-04
	분산	1.41e-01	1.00e-01	6.75e-02	4.66e-02
	중앙값	2.14e-03	7.87e-04	7.21e-04	5.66e-04
		MF	SelfCF	DirectAU	
사용자 임베딩	평균	-2.61e-02	1.91e-02	-7.65e-06	
	표준 편차	3.58e-01	5.89e-01	1.14e-02	
	표준 오차	5.75e-04	9.48e-04	1.84e-05	
	분산	1.28e-01	3.47e-01	1.30e-04	
	중앙값	-3.45e-02	7.71e-03	5.36e-05	
아이템 임베딩	평균	1.65e-02	1.53e-02	4.47e-05	
	표준 편차	4.24e-01	7.03e-01	1.37e-02	
	표준 오차	8.97e-04	1.49e-03	2.90e-05	
	분산	1.80e-01	4.94e-01	1.88e-04	
	중앙값	1.51e-02	1.93e-02	-1.17e-05	

부록 3. 기존 알고리즘과 개선된 제안 방법의 전체 성능

<표 10> FilmTrust 데이터 집합에서의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 수치적 실험 결과

	FilmTrust			
	Weight	Best Epoch	recall	NDCG
BUIR	-	291	0.83774	0.60724
BUIR_W	[2048,2048,2048]	246	0.84692	0.60861
BUIR_E	-	276	0.83684	0.60783
BUIR_EW	[512,512,512]	266	0.85103	0.61154
MixGCF	-	173	0.83204	0.61152
MixGCF_W	[2,2,2]	189	0.83640	0.62930
MixGCF_E	-	296	0.83370	0.60688
MixGCF_EW	[2,2,2]	299	0.83768	0.62941
LightGCN	-	276	0.85376	0.61566
LightGCN_W	[8,8,8]	76	0.86131	0.61938
LightGCN_E	-	266	0.85287	0.61690
LightGCN_EW	[4,4,4]	161	0.86316	0.61970
SGL	-	121	0.85761	0.62903
SGL_W	[2,2,2]	86	0.86001	0.62691
SGL_E	-	63	0.85227	0.63580
SGL_EW	[4,4,4]	27	0.85718	0.63460
SimGCL	-	1	0.31356	0.16678
SimGCL_W	[32,32,32]	10	0.85551	0.63114
SimGCL_E	-	1	0.67154	0.41195
SimGCL_EW	[8,8,8]	13	0.85506	0.63040
XSimGCL	-	1	0.65846	0.48539
XSimGCL_W	[8,8,8]	6	0.85703	0.63120

XSimGCL_E	-	1	0.68843	0.48096
XSimGCL_EW	[4,4,4]	24	0.86259	0.64195
MF	-	296	0.84800	0.61690
DirectAU	-	7	0.04075	0.02646
SelfCF	-	54	0.83840	0.60760

<표 11> Yelp2018 데이터 집합에서의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 수치적 실험 결과

	Yelp2018			
	Weight	Best Epoch	recall	NDCG
BUIR	-	290	0.04371	0.03557
BUIR_W	[1024,1024,1024]	294	0.04521	0.03708
BUIR_E	-	79	0.04578	0.03718
BUIR_EW	[2,2,2]	298	0.04530	0.03707
MixGCF	-	96	0.07010	0.05752
MixGCF_W	[2,2,2]	85	0.06931	0.05697
MixGCF_E	-	111	0.07009	0.05736
MixGCF_EW	[2,2,2]	86	0.06875	0.05655
LightGCN	-	231	0.05949	0.04887
LightGCN_W	[8,8,8]	56	0.06191	0.05086
LightGCN_E	-	186	0.06059	0.04993
LightGCN_EW	[8,8,8]	36	0.06312	0.05148
SGL	-	27	0.06791	0.05585
SGL_W	[2,2,2]	23	0.06978	0.05716
SGL_E	-	20	0.06480	0.05304
SGL_EW	[4,4,4]	19	0.06750	0.05490
SimGCL	-	18	0.07248	0.05976
SimGCL_W	[2,2,2]	9	0.07207	0.05938
SimGCL_E	-	18	0.07217	0.05932

SimGCL_EW	[2,2,2]	10	0.07162	0.05927
XSimGCL	-	14	0.07277	0.06002
XSimGCL_W	[2,2,2]	8	0.07215	0.05956
XSimGCL_E	-	13	0.07241	0.05964
XSimGCL_EW	[2,2,2]	7	0.07170	0.05898
MF	-	101	0.04998	0.04055
DirectAU	-	47	0.07017	0.05807
SelfCF	-	290	0.04414	0.03601

<표 12> Douban-book 데이터 집합에서의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 수치적 실험 결과

	Douban-book			
	Weight	Best Epoch	recall	NDCG
BUIR	-	298	0.08945	0.07097
BUIR_W	[2048,2048,2048]	294	0.10467	0.08412
BUIR_E	-	55	0.09774	0.07554
BUIR_EW	[2048,2048,2048]	277	0.10414	0.08367
MixGCF	-	145	0.17641	0.16265
MixGCF_W	[2,2,2]	133	0.17440	0.15820
MixGCF_E	-	135	0.17390	0.15990
MixGCF_EW	[2,2,2]	177	0.17329	0.15608
LightGCN	-	266	0.14783	0.12508
LightGCN_W	[16,16,16]	26	0.15127	0.12867
LightGCN_E	-	221	0.14695	0.12327
LightGCN_EW	[8,8,8]	41	0.14975	0.12468
SGL	-	46	0.17322	0.15225
SGL_W	[2,2,2]	36	0.17441	0.15360
SGL_E	-	30	0.17016	0.14813
SGL_EW	[2,2,2]	23	0.17161	0.14966

SimGCL	-	51	0.16677	0.14781
SimGCL_W	[2,2,2]	28	0.17041	0.14940
SimGCL_E	-	52	0.16766	0.14802
SimGCL_EW	[2,2,2]	27	0.16655	0.14708
XSimGCL	-	29	0.17632	0.15397
XSimGCL_W	[2,2,2]	11	0.17709	0.15444
XSimGCL_E	-	28	0.17729	0.15364
XSimGCL_EW	[2,2,2]	11	0.17473	0.15313
MF	-	46	0.24900	0.28310
DirectAU	-	8	0.10870	0.10300
SelfCF	-	300	0.19240	0.22440

<표 13> MovieLens-1M 데이터 집합에서의 알고리즘 별 기존 방법과 제안 방법 적용에 따른 수치적 실험 결과

	MovieLens-1M			
	Weight	Best Epoch	recall	NDCG
BUIR	-	298	0.19115	0.22593
BUIR_W	[4,4,4]	288	0.19308	0.22740
BUIR_E	-	299	0.20169	0.23589
BUIR_EW	[8,8,8]	289	0.19418	0.22996
MixGCF	-	2	0.14913	0.17538
MixGCF_W	[16,16,16]	4	0.21307	0.24429
MixGCF_E	-	15	0.16410	0.18650
MixGCF_EW	[8,8,8]	14	0.23060	0.26500
LightGCN	-	261	0.27196	0.30274
LightGCN_W	[64,64,64]	11	0.27716	0.30852
LightGCN_E	-	196	0.27088	0.30331
LightGCN_EW	[16,16,16]	31	0.27767	0.31078
SGL	-	42	0.27369	0.30960

SGL_W	[8,8,8]	28	0.28670	0.32120
SGL_E	-	22	0.26960	0.30656
SGL_EW	[4,4,4]	18	0.28149	0.31729
SimGCL	-	217	0.25545	0.28191
SimGCL_W	[16,16,16]	5	0.27555	0.30903
SimGCL_E	-	217	0.25593	0.28349
SimGCL_EW	[16,16,16]	7	0.27277	0.30427
XSimGCL	-	61	0.28515	0.32007
XSimGCL_W	[2,2,2]	23	0.28430	0.31830
XSimGCL_E	-	62	0.28558	0.32034
XSimGCL_EW	[2,2,2]	23	0.28553	0.31886
MF	-	66	0.12470	0.10390
DirectAU	-	23	0.13670	0.11890
SelfCF	-	297	0.08509	0.06709

Abstract

MS.Thesis

Optimization methods for Graph Convolution Networks in Recommendation Systems

Lee Sangmin
Department of Computer Science
Graduate School
Kyonggi University

Recommendation Systems help users quickly find the content they want on the Internet, where there is a lot of information, and recommendation systems play an important role in enhancing user experience and increasing the profitability of platforms by accurately analyzing user preferences and behavior and providing personalized products or services. To implement these recommendation systems efficiently, Graph Convolution Network (GCN) algorithms are widely used. These algorithms play an important role in enhancing the user experience and increasing the profitability of the platform by analyzing user preferences and behaviors more accurately than traditional methodologies and delivering personalized products or services. GCN excel at modeling complex interactions between users and items by leveraging their graph structure, which can improve the accuracy and efficiency of recommendation systems. However, GCN-based recommendation systems face the problem of loss of embedding values and inability to build deep layers during the learning process, which negatively affects the learning speed and accuracy.

In Recommender Systems, learning speed and accuracy are very important and directly affect the quality of recommendations and user experience. Slow learning speeds can make it difficult for recommendation systems to quickly reflect the latest behavioral patterns and preferences of users. This can lead to inaccurate recommendations based on outdated information, which can reduce user satisfaction. In addition, if the algorithm's recommendation accuracy is low, the system may not be able to understand the user's true preferences, which increases the likelihood of recommending items that are less relevant to the user. Therefore, this paper proposes two techniques to improve the learning speed and accuracy of recommendation systems utilizing graph convolutional networks. To improve the learning speed and accuracy of the recommendation system through the proposed techniques, we propose the Egress initialization technique and the Weighted Forwarding (WF) technique. The egress initialization technique initializes the embedding values to a wider range to prevent the loss of embedding values and accelerate the learning process, and the information of each node propagates through a wider value range to capture more diverse information in the graph structure, thereby improving the recommendation accuracy. The WF technique enhances the embedding value by multiplying the embedding value with a weight and passing it to the next layer, which improves the learning speed and recommendation accuracy of GCN. To analyze the experimental results of the recommendation performance of the proposed techniques, we performed statistical analysis, PCA, and GKDE on the embedding values, and the experimental results showed that the two proposed techniques improved the overall learning speed and accuracy, and also solved the problem that the existing algorithms did not work properly in certain data sets. This means that the proposed methods can better utilize the results of modeling the deep relationship between users and items in the existing algorithm and improve accuracy by mitigating the problems that occurred during the learning process.