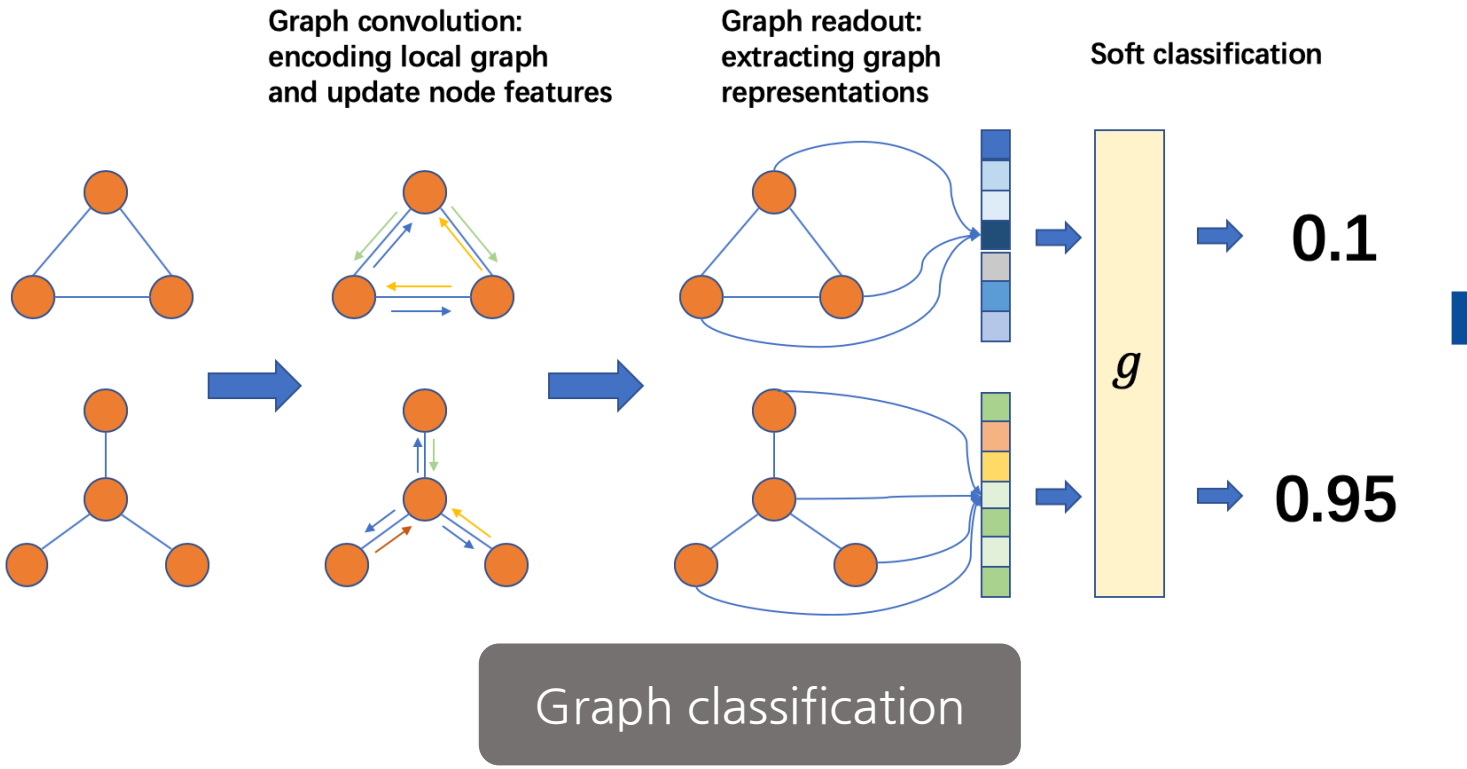
LightGCN

이은경

목차

1. Abstract

1) 정의

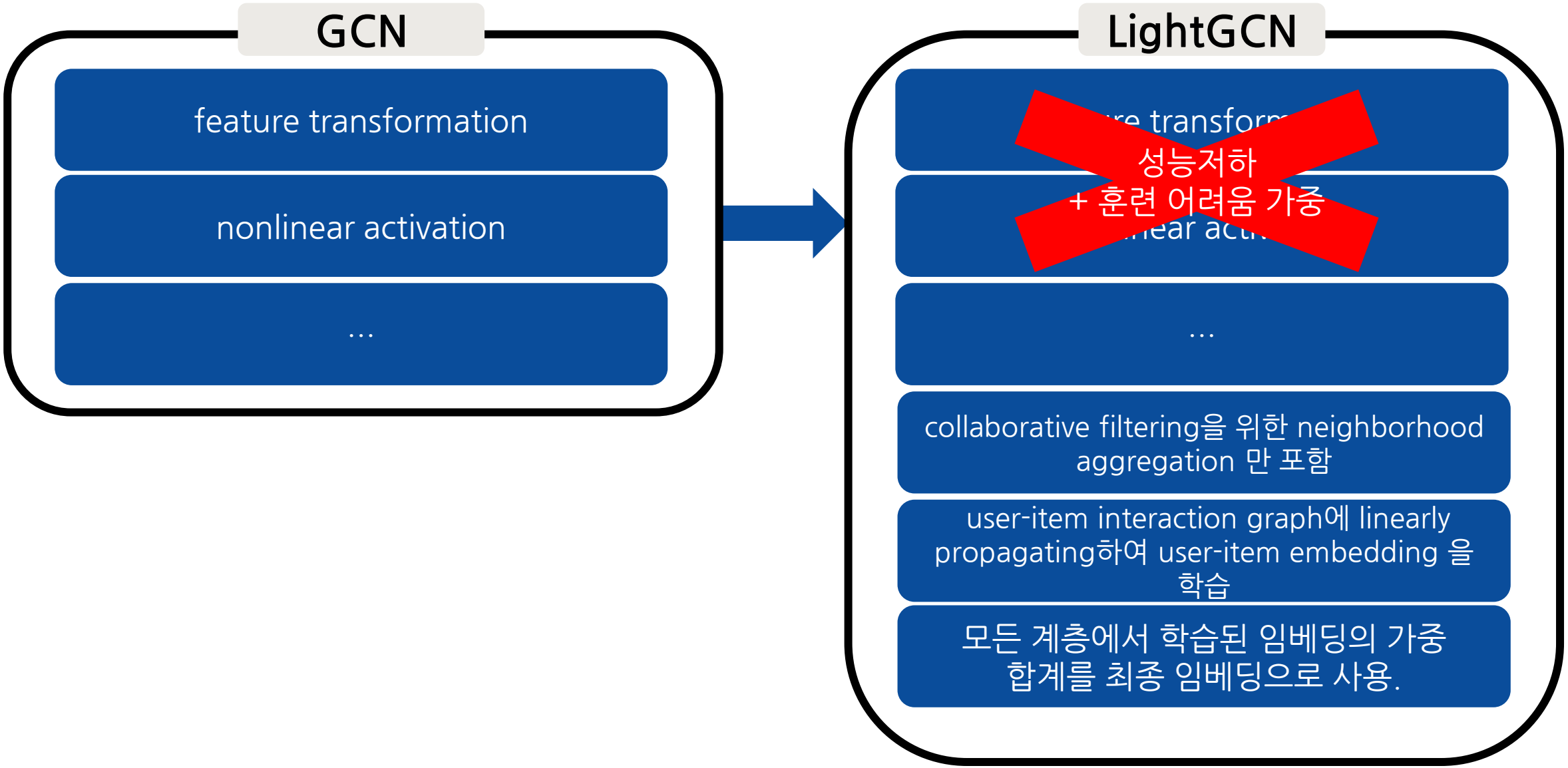


recommandation

neural network operation의
ablation analyses 부족

1) 개요

1. Abstract



2. Introduction

기여

2. Introduction

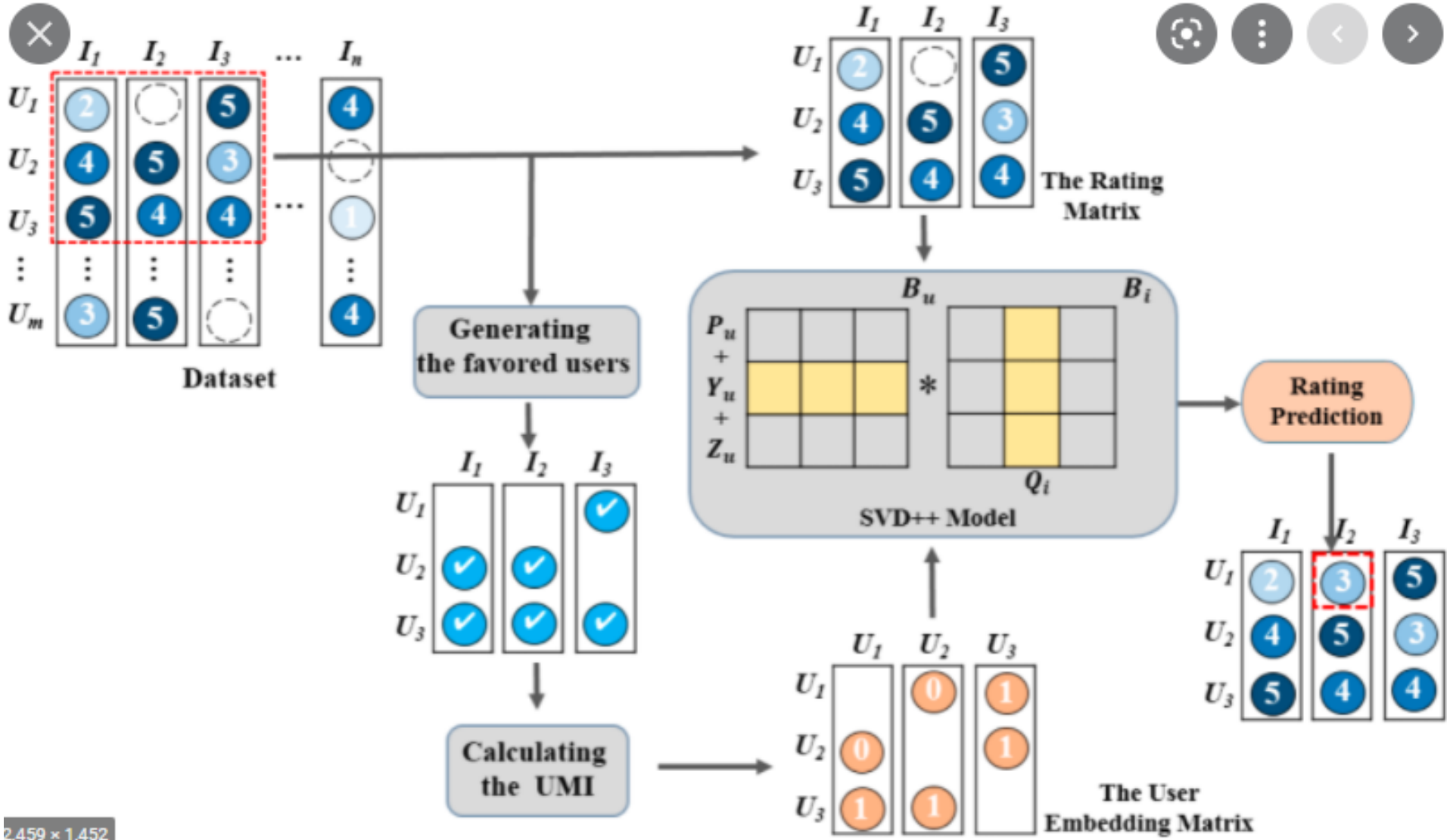
GCN의 feature transformation 과 nonlinear activation가 collaboration filtering의 효과에 대해 부정적 영향을 미침을 확인.

recommendation에 GCN의 가장 중요한 구성 요소만 포함함으로써 모델 단순화

동일한 설정을 따라 LightGCN과 NGCF를 경험적으로 비교/개선.
기술적/경험적 관점에서 LightGCN의 합리성 분석

3. Relate work

1) Collaborative Filtering



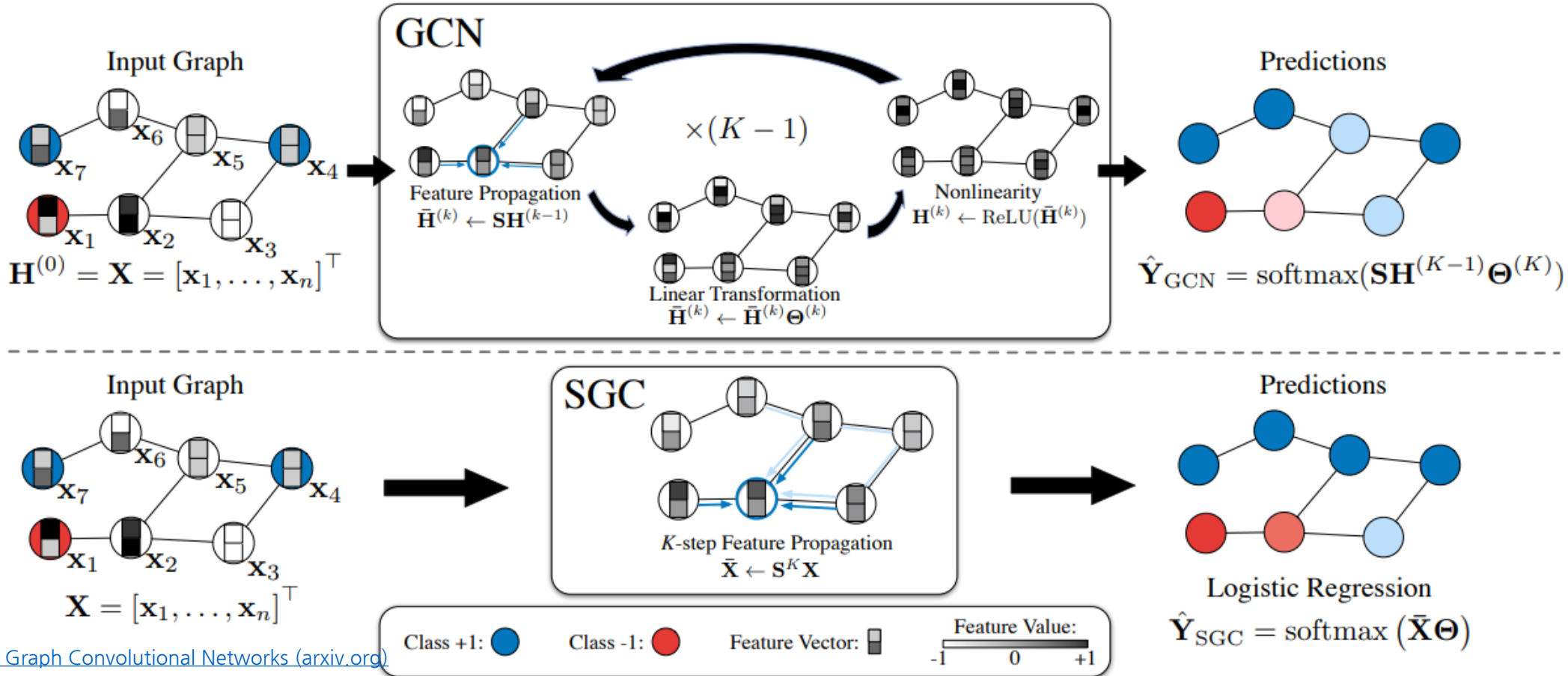
출처 : Symmetry | Free Full-Text | User Embedding for Rating Prediction in SVD++-Based Collaborative Filtering | HTML (mdpi.com)

2) Graph Methods for Recommendation

3. Relate work

1. ItemRank : label propagation mechanism을 사용하여 그래프를 통해 user preference scores를 직접 propagation > 연결된 노드가 유사한 레이블을 갖도록

2. Wu et al. [40] : Nonlinearity 제거 / 다중 가중치 Matrix 하나로 > simplified GCN(SGCN) 모델을 개발



출처 : Simplifying Graph Convolutional Networks (arxiv.org)

2) Graph Methods for Recommendation

- 차이점은 LightGCN과 SGCN은 서로 다른 작업에 대해 개발되기 때문에 모델 단순화의 합리성이 다름.
  - SGCN은 node classification. 모델 해석성과 효율성을 위한 단순화를 수행
  - LightGCN은 node가 ID feature 갖는 협업 필터링(CF)에 있음. 따라서 비선형성과 가중치 메트릭스는 CF에 유용하지 않으며 모델 training에도 해를 끼침.
  - node 정밀도의 경우 SGCN은 GCN과 동등(때로는 약함)하며, CF 정확도의 경우 LightGCN이 GCN을 큰 폭으로 능가(NGCF에 비해 15% 이상 개선).

• NGCF에서 비선형성이 불필요하다는 것을 발견하고 CF에 대한 선형 GCN 모델을 개발.

• LightGCN은 한 단계 더 발전. 중복 매개 변수를 제거하고 ID 임베딩만 유지하므로 모델이 MF처럼 간단함.

4. Preliminaries

1) NGCF Brief

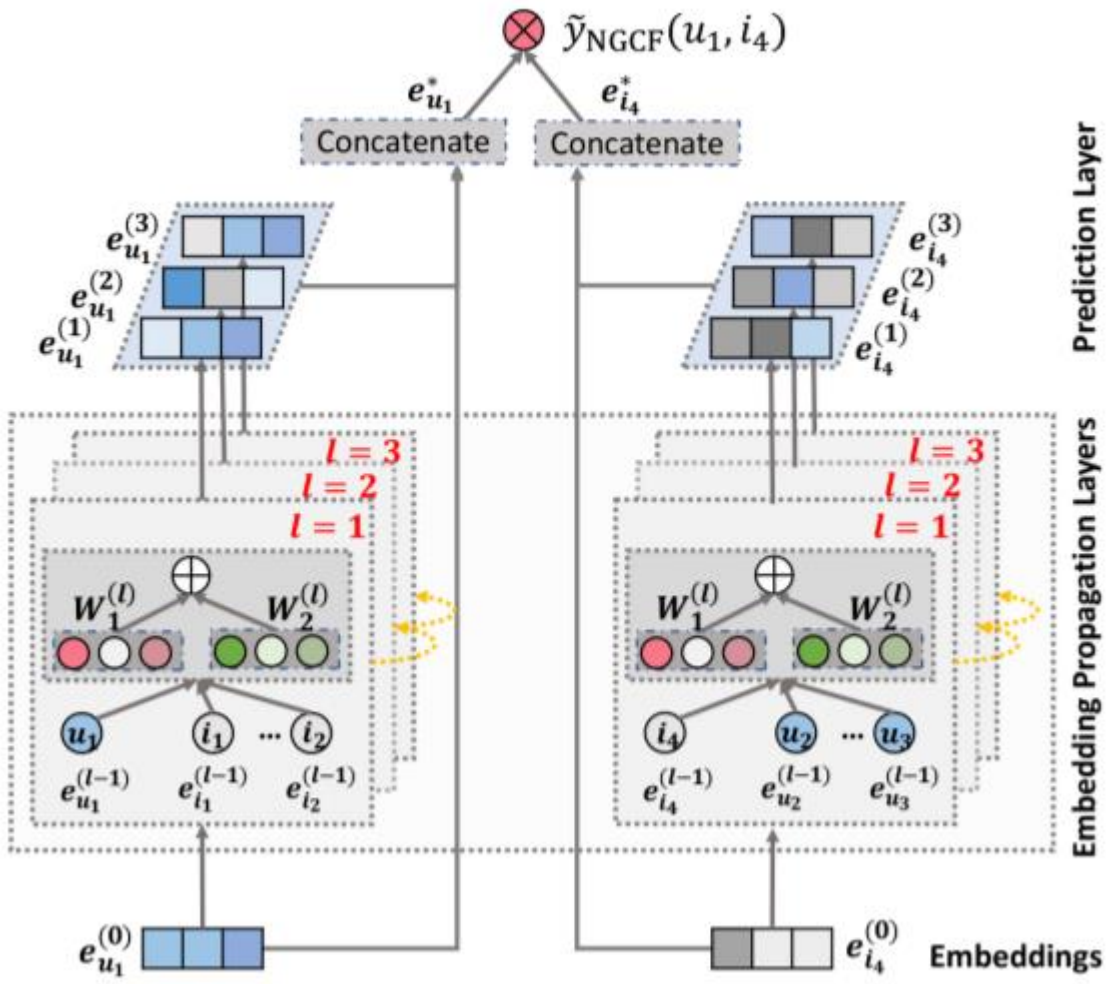
Table 1: Performance of NGCF and its three variants.

	Gowalla		Amazon-Book	
	recall	ndcg	recall	ndcg
NGCF	0.1547	0.1307	0.0330	0.0254
NGCF-f	0.1686	0.1439	0.0368	0.0283
NGCF-n	0.1536	0.1295	0.0336	0.0258
NGCF-fn	0.1742	0.1476	0.0399	0.0303

- NGCF-f : the feature transformation matrices W1 and W2 제거
- NGCF-n : non-linear activation function σ 제거
- NGCF-fn : the feature transformation matrices and non-linear activation function 둘 다 제거

➢ 비선형 활성화를 제거해도 정확도에 큰 영향을 미치지 않으나 feature transformation(즉, NGCF-fn) 제거를 기반으로 비선형 활성화를 제거하면 성능이 크게 향상

1) NGCF Brief



4. Preliminaries

$$\mathbf{E} = [\underbrace{\mathbf{e}_{u_1}, \cdots, \mathbf{e}_{u_N}}_{\text{users embeddings}}, \underbrace{\mathbf{e}_{i_1}, \cdots, \mathbf{e}_{i_M}}_{\text{item embeddings}}]. \tag{1}$$

Construction

$$\mathbf{m}_{u \leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i + \mathbf{W}_2(\mathbf{e}_i \odot \mathbf{e}_u)\right), \tag{3}$$

$\mathbf{e}_i$: $\mathbf{e}_i$, $\mathbf{e}_u$ 상호

Aggregation

$$\mathbf{e}_u^{(1)} = \text{LeakyReLU}\left(\mathbf{m}_{u \leftarrow u} + \sum_{i \in \mathcal{N}_u} \mathbf{m}_{u \leftarrow i}\right), \tag{4}$$

Propagation

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u \leftarrow u}^{(l)} + \sum_{i \in \mathcal{N}_u} \mathbf{m}_{u \leftarrow i}^{(l)}\right), \tag{5}$$

All

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}))\right),$$

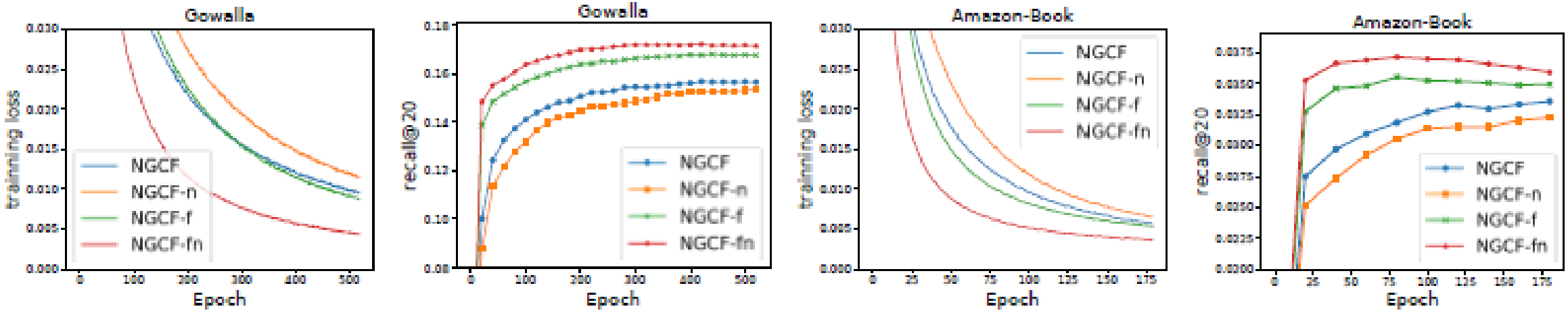
$$\mathbf{e}_i^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_u^{(k)} + \mathbf{W}_2(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)}))\right),$$

2) Empirical Explorations on NGCF

- nonlinear activation와 feature transformation의 효과를 탐구하기 위해 NGCF에 대한 ablation study 수행.
- GCN의 핵심 : propagation의 임베딩을 refine하는 것. 동일한 크기의 임베딩의 quality에 더 관심.
- 따라서, 최종 임베딩을 얻는 방법을 concat $e_u^* = e_u^{(0)} \,||\, \cdots \,||\, e_u^{(L)}$ 에서 합 $e_u^* = e_u^{(0)} + \cdots + e_u^{(L)}$ 로 변경.
- 이는 NGCF의 성능에 거의 영향을 미치지 않지만, GCN에 의해 정제된 임베딩 품질을 더 잘 나타내도록 함.

1. feature transformation을 추가하면 NGCF에 부정적 영향. NGCF 및 NGCF-n 에서 제거되면 성능이 크게 향상
2. Non-linear activation를 추가하면 feature transformation이 포함될 때는 약간의 영향을 미치지만 feature transformation이 비활성화되면 부정적인 영향.
3. feature transformation과 nonlinear activation를 동시에 제거함으로써 NGCF-fn이 NGCF에 비해 큰 개선(recall 9.57% 개선)을 나타내기 때문에 전체적으로 NGCF에 다소 부정적인 영향.

2) Empirical Explorations on NGCF



(a) Training loss on Gowalla (b) Testing recall on Gowalla (c) Training loss on Amazon-Book (d) Testing recall on Amazon-Book

Figure 1: Training curves (training loss and testing recall) of NGCF and its three simplified variants.

NGCF가 악화되는 이유를 이해하기 위해, training loss 및 test 기록된 모델 상태의 곡선

5. Method

0) 기존 GCN

- GCN의 기본 아이디어 : 그래프의 feature를 smoothing해 노드 representation 학습하기 위해 그래프 컨볼루션을 반복적으로 수행. > 대상 노드의 새로운 representation 으로 이웃의 특징을 aggregate.

$$\mathbf{e}_u^{(k+1)} = \mathrm{AGG}(\mathbf{e}_u^{(k)}, \{\mathbf{e}_i^{(k)} : i \in \mathcal{N}_u\}). \qquad (2)$$

- AGG는 대상 노드와 인접 노드에 대한 k번째 계층의 표현을 고려하는 aggregation function
- 예) GIN[42]의 weighted sum aggregator, GraphSAGE의 LSTM aggregator[14] 및 BGNN[48] 등의 bilinear interaction aggregator 등

- 그러나, 대부분 AGG 함수와 feature transformation 또는 nonlinear activation을 결합.
- semantic input features을 가진 node 또는 graph classification tasks에서 잘 수행되지만 collaborative filtering 에서는 부담이 될 수 있음

1. LightGCN
2. Model Analysis
3. Training

5. Method

1) Light Graph Convolution (LGC)

- 간단한 weighted sum aggregator를 채택
- feature transformation과 nonlinear activation 미사용.

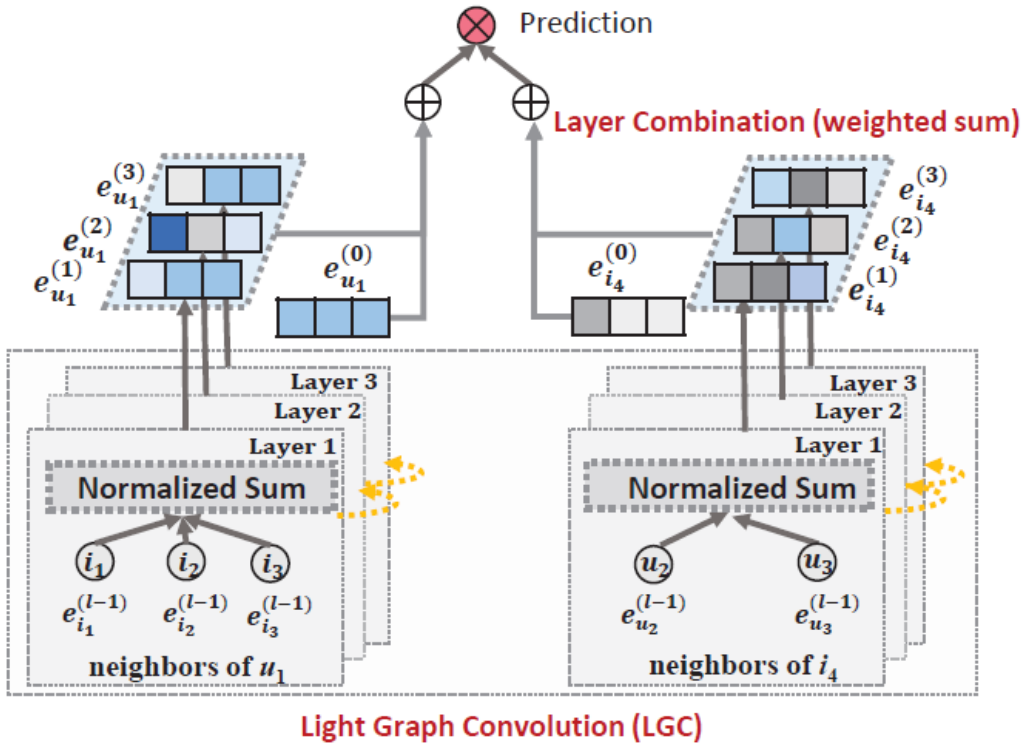


Figure 2: An illustration of LightGCN model architecture. In LGC, only the normalized sum of neighbor embeddings is performed towards next layer; other operations like self-connection, feature transformation, and nonlinear activation are all removed, which largely simplifies GCNs. In Layer Combination, we sum over the embeddings at each layer to obtain the final representations.

1) Light Graph Convolution (LGC)

- LightGCN의 그래프 컨볼루션 연산(일명 propagation rule[39])의 정의.

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}. \quad (3)$$

대칭 정규화항 그래프 컨볼루션 연산에 따른 임베딩의 scale 증가 방지

- L1 norm과 같은 다른 정규화도 사용가능하나, 경험적으로 대칭 정규화가 좋은 성능(섹션 4.4.2)
- 확장된 이웃을 집계하고 self-connection을 특별히 처리해야 하는 대부분의 기존 그래프 연산과 달리 LGC에서는 연결된 이웃만 집계하고 대상 노드 자체(즉, 자체 연결)를 통합하지 않음.
- Layer Combination은 self-connection과 동일한 효과라서 self-connection을 포함할 필요가 없음.

2) Layer Combination and Model Prediction.

- training 가능한 parameter는 0 layer(모든 user에 대한 $e_u^{(0)}$와 모든 item에 대한 $e_i^{(0)}$ 의 임베딩)
- K 레이어 LGC 후 각 레이어에서 얻은 Embedding을 추가로 결합하여 user(item)의 최종 표현을 생성.
-

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}. \quad (3)$$

대칭 정규화항

- $\alpha_k \leq 0$: 최종 임베딩을 구성하는 k번째 레이어 임베딩의 중요성. (hyperparameter)
- αk를 $\frac{1}{K+1}$로 균일하게 설정하는 것이 일반적으로 좋은 성능.(단순성 유지)

3) Layer Combination and Model Prediction.

1. **LightGCN**
2. Model Analysis
3. Training

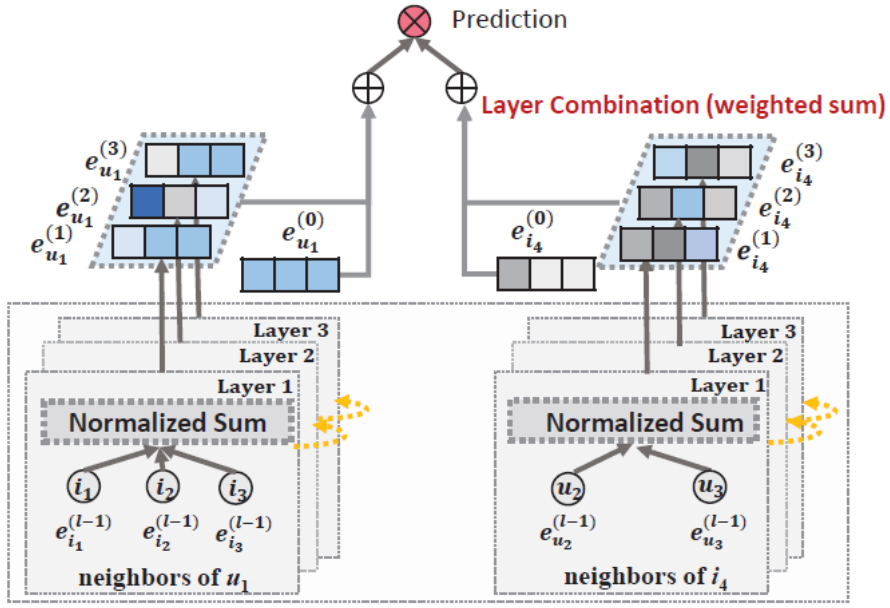
5. Method

- 최종 representation을 얻기 위해 **레이어 조합을** 수행하는 이유.

1) layer 의 수가 너무 많이 증가하면 임베딩이 over-smoothed [27]. 단순히 last layer를 사용하면 문제 발생.

2) 다른 layer의 임베딩은 다른 semantics을 포착. 예) 첫 번째 layer 은 상호작용이 있는 user와 item에 smoothness를 적용하고, 두 번째 계층은 상호작용된 item(user)에 겹치는 user(item)를 smooth하며, higher-order proximity[39]를 포착. > layer를 결합하면 표현이 더 comprehensive 됨.

3) weighted sum 와 서로 다른 layer 의 임베딩을 결합하면 그래프 컨볼루션의 효과를 self-connections 로 포착



4) Matrix Form

- 구현과 기존 모델과의 논의를 촉진하기 위해 LightGCN의 매트릭스 형태를 제공. user-item 상호 작용 매트릭스를$R \in \mathbb{R}^{M \times N}$
  - 여기서 M과 N는 각각 user 수와 item 수
- item i와 상호 작용한 경우 각 entry R_{ui}는 1이 됨. 그런 다음 user item 그래프의 인접 행렬을 구함.

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix}, \qquad (6)$$

- 0번째 레이어 embedding 행렬을 $E^{(0)} \in \mathbb{R}^{(M+N) \times T}$로 하자. 여기서 T는 embedding size.
- LGC의 행렬 등가형을 다음과 같이 구할 수 있음

$$\mathbf{E}^{(k+1)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(k)}, \qquad (7)$$

- D는 $(M + N) \times (M + N)$ 대각 행렬이며, 여기서 각 entry D_{ii}는 Adjacency matrix A(degree matrix)의 i번째 행 벡터에 있는 0이 아닌 item의 수.

4) Matrix Form

- 모델 예측에 사용되는 최종 embedding matrix

$$\begin{aligned}\mathbf{E} &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} + \ldots + \alpha_K\mathbf{E}^{(K)} \\ &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \alpha_2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + \alpha_K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)},\end{aligned} \tag{8}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized matrix.

2. Model Analysis

1) 그래프 컨볼루션에 self-connection을 통합하는 최신 선형 GCN 모델인 Simplified GCN(SGCN) [40]과의 연결.

- **레이어 조합**이 self-connection의 효과를 나타내므로 추가할 필요가 없다는 것을 보여줌.

2) 개인화된 PageRank [15]에서 영감을 받아 oversmoothing 을 해결하는 Approximate Personalized Propagation of Neural Predictions (APP) [24]와의 연결.

- LightGCN과 APPNP 사이의 기본적인 동등성을 보여주므로 LightGCN은 제어 가능한 oversmoothing으로 long-range propagating의 benefits 을 얻음.

3) second-layer LGC를 분석하여 second-order neighbors와 user를 smooths하게하는 법을 보여주어 통찰력 제공.

1) Relation with SGCN.

1. LightGCN
2. **Model Analysis**
3. Training

5. Method

- Simplified GCN(SGCN) [40]의 저자는 노드 분류에 대한 GCN의 불필요한 복잡성을 주장하고 SGCN을 제안.

- Non-linearity을 제거하고 가중치 행렬을 하나의 가중치 행렬로 결합함으로써 GCN을 단순화.

- SGCN의 Graph Convolution

$$\mathbf{E}^{(k+1)} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}\mathbf{E}^{(k)}, \qquad (9)$$

생략 생략

- 여기서 $I \in \mathbb{R}^{(M+N)\times(M+N)}$는 항등 행렬로, self-connections을 포함하도록 A에 추가.
- SGCN에서 마지막 계층에서 얻은 임베딩은 다운스트림 예측 작업에 사용되며 다음과 같이 표현.

$$\begin{aligned}\mathbf{E}^{(K)} &= (\mathbf{A}+\mathbf{I})\mathbf{E}^{(K-1)} = (\mathbf{A}+\mathbf{I})^{K}\mathbf{E}^{(0)} \\ &= \binom{K}{0}\mathbf{E}^{(0)} + \binom{K}{1}\mathbf{A}\mathbf{E}^{(0)} + \binom{K}{2}\mathbf{A}^{2}\mathbf{E}^{(0)} + \ldots + \binom{K}{K}\mathbf{A}^{K}\mathbf{E}^{(0)}. \end{aligned} \qquad (10)$$

- 위의 파생은 A에 self-connections을 삽입하고 그 안에 임베딩을 전파하는 것이 본질적으로 각 LGC 층에서 전파되는 임베딩의 가중치 합계와 같다는 것을 보여줌.

2) Relation with APPNP.

- Approximate Personalized Propagation of Neural Predictions (APP)[24]의 저자들은 GCN을 Personalized PageRank[15]와 연결하며, 이를 통해 oversmoothing 위험 없이 long range propagate가 가능한 APPNP라는 GCN 변형을 제안.
- APPNP는 complements each propagation layer with the starting features (즉, 0번째 layer embedding)으로 보완하여 locality 를 보존하고(즉, oversmoothing을 완화하기 위해 루트 노드에 가까이 머무름) large neighborhood 의 정보를 활용 가능. APPNP의 propagation 계층의 정의

$$\mathbf{E}^{(k+1)} = \beta\mathbf{E}^{(0)} + (1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(k)}, \qquad (11)$$

- 여기서 β는 propagation 에서 starting features 의 유지를 제어하는 teleport 확률이고, $\tilde{A}$는 정규화 adjacency 행렬.

2) Relation with APPNP.

- APPNP에서 마지막 계층은 final prediction을 위해 사용.

$$\begin{aligned}\mathbf{E}^{(K)} &= \beta\mathbf{E}^{(0)} + (1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(K-1)}, \\ &= \beta\mathbf{E}^{(0)} + \beta(1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + (1-\beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(K-2)} \\ &= \beta\mathbf{E}^{(0)} + \beta(1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \beta(1-\beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + (1-\beta)^K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}.\end{aligned} \tag{12}$$

- 방정식 (8)에 따라, LightGCN이 α_k를 설정함으로써 APPNP에서 사용되는 prediction embedding을 완전히 복구 가능함을 확인.
- LightGCN은 α를 적절하게 설정함으로써, 제어 가능한 장거리 모델링에 large K를 사용가능

3) Second-Order Embedding Smoothness.

- second-layer LightGCN을 분석하여 합리성 입증.
- 예) user 측면에서 두 번째 계층은 상호 작용한 item에서 겹치는 user를 부드럽게함.

$$\mathbf{e}_u^{(2)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{|\mathcal{N}_i|} \sum_{v \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}} \mathbf{e}_v^{(0)}. \tag{13}$$

- 다른 user v가 target user u와 상호 작용한 경우, u에 대한 v의 smoothness 강도는 coefficient 에 의해 측정

$$c_{v->u} = \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}} \sum_{i \in \mathcal{N}_u \cap \mathcal{N}_v} \frac{1}{|\mathcal{N}_i|}. \tag{14}$$

- Coefficient 해석 : 1) 공동 상호작용 item의 수가 많을수록 더 크고, 2) 공동 상호작용 item의 인기가 낮을수록 v의 활동이 더 크고, 3) 활동량이 적을수록, 더 크다. > user 유사성 측정에서 CF의 가정을 잘 수용하고 LightGCN의 타당성을 입증.

3. Model Training

- training 가능한 parameters는 0-th layer($\Theta = \{E^{(0)}\}$의 임베딩) 뿐
- BPR Loss[32]를 사용. 관측된 entry가 관측되지 않은 entry에 비해 높게 예측하도록 권장하는 pairwise loss.

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda ||\mathbf{E}^{(0)}||^2 \qquad (15)$$

- λ는 L2 regularization strength를 제어. Adam [22] optimizer 사용. 미니 배치 방식 사용.
- dropout mechanisms은 미사용 : 이유는 LightGCN에 feature transformation weight matrices가 없기 때문에 embedding 레이어에 L2 regularization를 적용하면 과적합 방지.
- 두 가지 드롭아웃 비율(노드 드롭아웃 및 메시지 드롭아웃)을 추가로 조정하고 각 레이어 투닛 길이의 임베딩을 정규화해야 하는 NGCF보다 training 및 튜닝이 쉽다.

3. Model Training

- Layer combination coefficients$\{\alpha_k\}_{k=0}^{K}$을 학습하거나 attention network를 사용하여 매개 변수를 지정하는 것도 기술적으로 실행 가능하나, training 데이터에 대한 α 학습이 개선없음을 발견.
- training 데이터가 미지의 데이터로 일반화할 수 있는 좋은 α를 학습하기에 충분한 신호를 포함하지 않다고 추측.
- 검증 데이터에 대한 하이퍼 파라미터를 학습하는 [5]에서 영감을 받아 검증 데이터에서 α를 학습하려고 노력. 실적이 약간 개선(1% 미만)

6. Experiments

6. Experiments

1) Experimental Settings

- 실험양을 줄이고 비교를 공정하게 하기 위해, NGCF 의 설정을 면밀히 준수[39].
- 통계가 표 2에 나와 있는 저자에게 실험 dataset(train/test split 포함)를 요청해 사용. Gowalla와 Amazon-Book은 사용된 NGCF 와 정확히 같으며, sowe는 NGCF의 결과를 직접 사용.
- 유일한 예외는 수정된 버전인 Yelp2018. 이전 버전은 dataset 에서 cold start item을 걸러내지 않았고, 수정된 버전만 공유. 따라서 Yelp2018 데이터에 대해 NGCF를 다시 실행.
- evaluation metrics 는 all-ranking protocol — user가 상호 작용하지 않는 모든 item이 후보로 계산된 recall@20 및 ndcg@20

Table 2: Statistics of the experimented data.

Dataset	User #	Item #	Interaction #	Density
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

2) Compared Methods.

- NGCF / GCN 기반 모델 GC-MC [35]와 PinSage [45]
- 신경망 기반 모델 NeuMF [19] 및 CMN [10] 및 인수분해 기반 모델을 포함한 여러 방법을 능가.
- Mult-VAE [28] : VAE(Variative Autoencoder)를 기반으로 한 item 기반 CF 방법. 데이터가 다항 분포에서 생성되고 모수 추정에 변동 추론을 사용하는 것으로 가정. [0, 0.2, 0.5]에서 드롭아웃 비율을 조정 [0.2, 0.4, 0.6, 0.8]에서 β를 조정.
- GRMF [30] : 그래프 라플라시안 regularizer를 추가하여 matrix factorization 을 smooths. rating prediction loss 을 BPRloss로 변경.

$$L = -\sum_{u=1}^{M}\sum_{i\in\mathcal{N}_u}\left(\sum_{j\notin\mathcal{N}_u}\ln\sigma(\mathbf{e}_u^T\mathbf{e}_i-\mathbf{e}_u^T\mathbf{e}_j)+\lambda_g||\mathbf{e}_u-\mathbf{e}_i||^2\right)+\lambda||\mathbf{E}||^2, \tag{16}$$

- λ는 [1e-5, 1e-4, ..., 1e-1] 에서 검색. 그래프 라플라시안 정규화 GRMF-norm 추가 $\lambda_g \left\|\frac{e_u}{\sqrt{|\mathcal{N}_u|}}-\frac{e_i}{\sqrt{|\mathcal{N}_i|}}\right\|^2$ 변형과 비교.

3) Hyper-parameter Settings.

- 임베딩 사이즈는 모든 모델에 대해 64로 고정
- 임베딩 매개 변수는 Xavier 방법으로 초기화[12].
- Adam[22]으로 LightGCN을 최적화하고 기본 학습 속도 0.001과 기본 미니 배치 크기 1024를 사용(Amazon-Book에서는 속도를 위해 미니 배치 크기를 2048).
- L2 정규화 계수 λ는 {1e-6, 1e-5, ..., 1e-2} 범위에서 검색되며 대부분의 경우 최적 값은 1e-4.
- layer combination coefficient α_k는 $\frac{1}{1+K}$로 균일하게 설정되며, 여기서 K는 layer 의 수. 1~4 범위에서 K를 테스트하고, K가 3일 때 만족스러운 성능.
- early stopping 와 validation strategies는 NGCF와 동일.
- 일반적으로 LightGCN은 1000 Epoch이면 충분히 수렴

2. Performance Comparison with NGCF

- 서로 다른 layer(1-4)에서 성능을 기록하면서 NGCF와 세부 비교를 수행

Table 3: Performance comparison between NGCF and LightGCN at different layers.

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

*The scores of NGCF on Gowalla and Amazon-Book are directly copied from Table 3 of the NGCF paper (https://arxiv.org/abs/1905.08108)

2. Performance Comparison with NGCF

- training 과정을 명확히 하기 위해 그림 3에 training loss 및 test recall의 training 곡선을 추가로 그림.

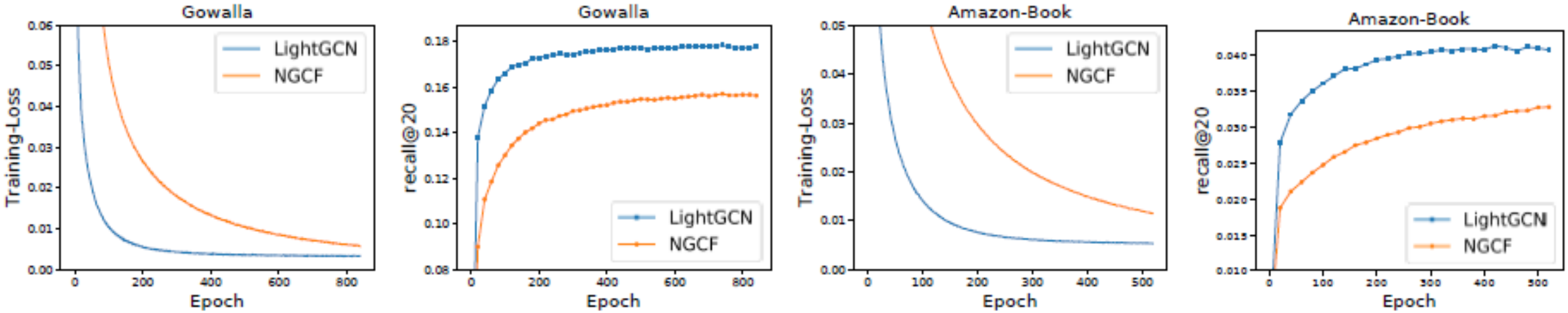


Figure 3: Training curves of LightGCN and NGCF, which are evaluated by training loss and testing recall per 20 epochs on Gowalla and Amazon-Book (results on Yelp2018 show exactly the same trend which are omitted for space).

2. Performance Comparison with NGCF

- 표 4를 섹션 2의 표 1과 정렬하면 LightGCN이 NGCF-fn보다 성능 우수.

Table 1: Performance of NGCF and its three variants.

	Gowalla		Amazon-Book	
	recall	ndcg	recall	ndcg
NGCF	0.1547	0.1307	0.0330	0.0254
NGCF-f	0.1686	0.1439	0.0368	0.0283
NGCF-n	0.1536	0.1295	0.0336	0.0258
NGCF-fn	0.1742	0.1476	0.0399	0.0303

Table 4: The comparison of overall performance among LightGCN and competing methods.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
GRMF	0.1477	0.1205	0.0571	0.0462	0.0354	0.0270
GRMF-norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
LightGCN	**0.1830**	**0.1554**	**0.0649**	**0.0530**	**0.0411**	**0.0315**

3. Performance Comparison with State-of-the-Arts

- α_k를 튜닝하면 더욱 개선될 수 있지만, 과도한 튜닝을 피하기 위해 K + 1의 균일한 설정만 사용.

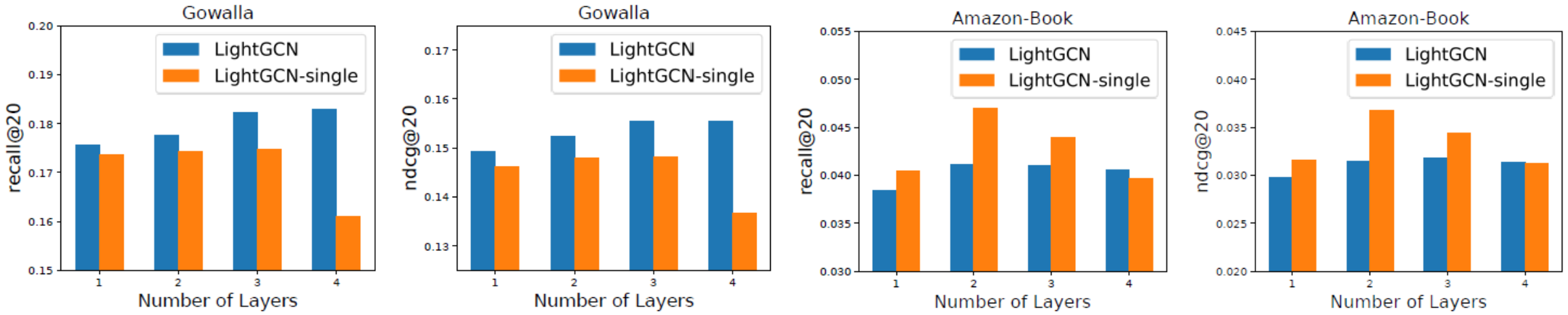


Figure 4: Results of LightGCN and the variant that does not use layer combination (i.e., LightGCN-single) at different layers on Gowalla and Amazon-Book (results on Yelp2018 shows the same trend with Amazon-Book which are omitted for space).

4. 2) Impact of Symmetric Sqrt Normalization.

- neighborhood aggregation(eq.3)를 수행할 때 각 neighbor embedding에 대칭 sqrt 정규화 사용
- L1 : L1 정규화

$$\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$$

L R

Table 5: Performance of the 3-layer LightGCN with different choices of normalization schemes in graph convolution.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
LightGCN-L_1-L	0.1724	0.1414	0.0630	0.0511	**0.0419**	**0.0320**
LightGCN-L_1-R	0.1578	0.1348	0.0587	0.0477	0.0334	0.0259
LightGCN-L_1	0.159	0.1319	0.0573	0.0465	0.0361	0.0275
LightGCN-L	0.1589	0.1317	0.0619	0.0509	0.0383	0.0299
LightGCN-R	0.1420	0.1156	0.0521	0.0401	0.0252	0.0196
LightGCN	**0.1830**	**0.1554**	**0.0649**	**0.0530**	0.0411	0.0315

Method notation: -L means only the left-side norm is used, -R means only the right-side norm is used, and -L_1 means the L_1 norm is used.

4. 3) Analysis of Embedding Smoothness

$$S_U = \sum_{u=1}^{M}\sum_{v=1}^{M} c_{v\to u}\left(\frac{\mathbf{e}_u}{||\mathbf{e}_u||^2} - \frac{\mathbf{e}_v}{||\mathbf{e}_v||^2}\right)^2, \tag{17}$$

Table 6: Smoothness loss of the embeddings learned by LightGCN and MF (the lower the smoother).

Dataset	Gowalla	Yelp2018	Amazon-book
	Smoothness of User Embeddings		
MF	15449.3	16258.2	38034.2
LightGCN-single	12872.7	10091.7	32191.1
	Smoothness of Item Embeddings		
MF	12106.7	16632.1	28307.9
LightGCN-single	5829.0	6459.8	16866.0

5. Hyper-parameter Studies

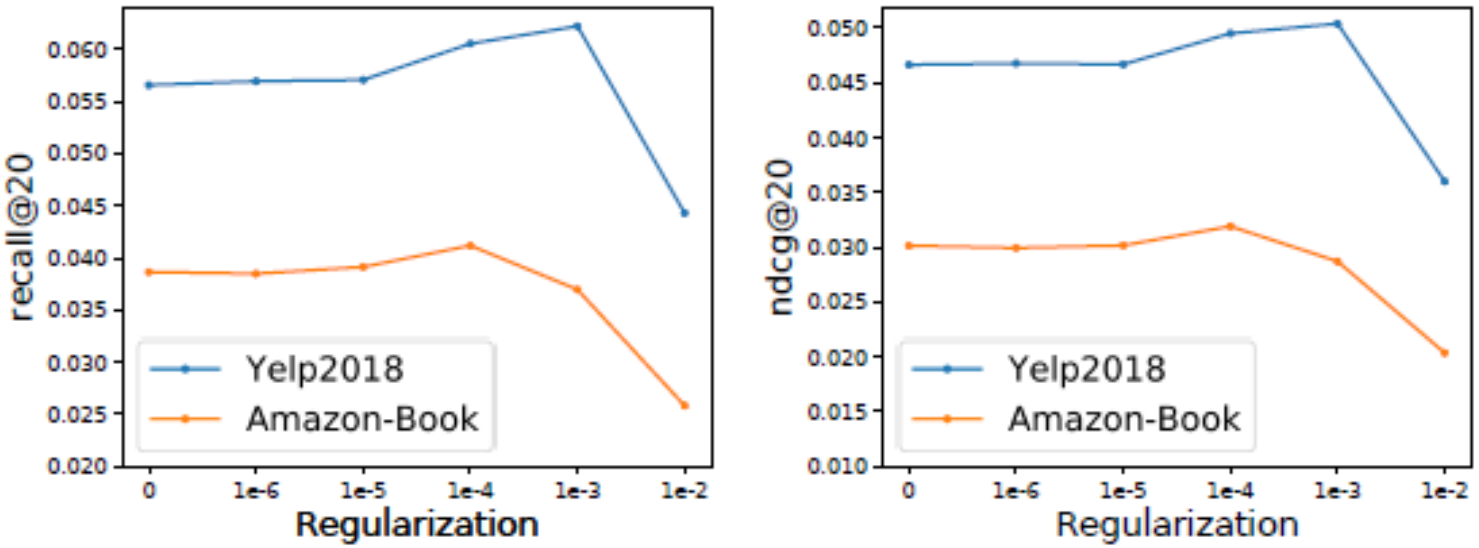


Figure 5: Performance of 2-layer LightGCN w.r.t. different regularization coefficient λ on Yelp and Amazon-Book.

7. Conclusion

7. Conclusion

- GCN 설계가 불필요하게 복잡하다 주장하고, 이 주장을 정당화하기 위해 경험적 연구 수행.
- Light Graph Convolution과 layer combination이라는 두 가지 필수 구성 요소로 구성된 LightGCN을 제안.
- Feature transformation 과 nonlinear activation, GCN의 두 가지 연산을 폐기하여 training 난이도를 낮춤.
- layer combination 에서, 노드의 final embedding 을 모든 레이어에 embedding 의 가중치로 구성해 self-connections 의 효과를 반영. Oversmoothing 제어.
- LightGCN 의 간단함을 입증하기 위한 실험을 수행. 즉, training이 더 쉽고 일반화 능력이 더 우수하며 더 효과적.
- 예측 모델에서 개체 간의 관계를 명시적으로 활용함으로써 관계를 암시적으로 모델링하는 기존의 supervised learning 체계에 유리[17, 33].
- 예) itemknowledge graph [38], social network [41] and multimedia content [44]과 같은 보조 정보를 recommendation 활용

Q & A
감사합니다