LightGCN: Simplifiying and Powering Graph Convolution Network for Recommendation

링크 : https://arxiv.org/pdf/2002.02126.pdf

ABSTRACT

GCN은 협업필터링에서 새로운 sota가 되었다.

그럼에도 추천에 대한 효과의 원인은 잘 이해되지 못했다.

GCN을 추천에 적용하는 작업들에서 어떤 것이 효과를 발휘했는지에 대한 조사가 충분하지 않았다.

(GCN은 원래 그래프 분류 태스크를 위해 만들어졌고, 다수의 신경망 연산이 포함된다.)

하지만 경험적으로 협업 필터링에는 큰 영향을 주지 않는 GCN의 2가지 설계를 찾았다.

1. 피쳐 트렌스포메이션
2. 비선형 활성화 함수

이 연구에서는 추천에 더 간결하고 적합하게 GCN의 설계를 단순화 하는데 초점을 맞춘다.

LightGCN을 제안하는데, 이는 협업 필터링을 위해 GCN의 필수적인 구성만 포함한다.(이웃 집계)

특히 LightGCN은 유저/아이템 임베딩을 유저-아이템 상호작용 그래프에서 그들을 선형 전파함으로써 학습하고, 임베딩의 모든 레이어를 가중합 하여 학습한 최종 임베딩을 사용한다.

이 간결한 모델은 실행하고 학습시키기 쉽고, 완전히 동일한 세팅에서 NGCF에 비해 상당한 개선을 보여준다.

INTRODUCTION

웹의 정보의 홍수를 해소하기 위해, 추천 시스템은 개인화된 정보 필터링 효과를 보이고 있다.

추천 시스템이 코어는 어떤 유저가 어떤 아이템에 상호작용할지 예측하는 것이다. (클릭, 평점부여, 구매 등 상호작용)

협업 필터링은 (과거 유저 아이템 상호작용을 취하여 예측을 달성하는데 집중한다) 효과적인 개인화 추천의 근간 작업이다.

CF를 위한 가장 일반적인 틀은 단일 유저/아이템을 설명하기 위해 잠재 피쳐를 학습하고, 임베딩 벡터 기반의 예측을 수행하는 것이다.

MF는 직접적으로 유저의 아이디를 임베딩에 투영하는 초기 모델이다.

이후에 몇 연구는 유저 아이디와 상호작용 기록을 입력으로 하면 임베딩의 품질을 높일 수 있다는 사실을 찾았다.

유저-아이템 상호작용 그래프 관점에서 이러한 개선은 임베딩 학습을 개선하기 위해 한 유저의 서브그래프 구조를 사용하는 것으로 부터 볼 수 있다.

여러 거리를 떨어져있는 서브 그래프 구조를 사용하는 것으로 최근에 NGCF는 CF에서 sota를 달성했다.

NGCF는 GCN으로부터 영감을 받고, 임베딩을 정의하기 위해 동일한 전파 규칙을 사용한다.

(피쳐 트렌스포메이션, 비선형 활성화함수 등)

NGCF가 뛰어난 성능을 보여줬지만, 그것의 디자인이 꽤 무겁고 부담스럽다고 생각하다. 많은 연산들은 큰 이유 없이 GCN으로 부터 직접적으로 차용한 것이다.

결과적으로 CF 태스크에서 꼭 유용한 것은 아니었다. 좀 더 자세히 하면 GCN은 원래 그래프의 속성인 각 노드가 입력으로 쓸 충분한 속성을 갖고있는 노드 분류를 위해 제안되었다.

반면에 CF에서 유저-아이템 상호작용은 구체적인 의미를 갖지 않고 식별자의 역할을 하는 원핫 벡터의 형태로만 표현될 수 있다.

이런 경우 id임베딩이 입력으로 주어지고, 여러 레이어에서 비선형 변환을 수행(최근 신경망의 성공의 키)하는 것은 이익은 없고, 모델학습에 더 안좋은 영향을 준다.

NGCF와 동일한 데이터와 메트릭으로 비교

GCN으로부터 [피쳐 변환과 비선형 활성함수]를 NGCF로 가져오는 것은 의미가 없었다.(없애는게 성능이 더 좋음)

목표하는 태스크에 불필요한 연산을 더하는 것은, 성능을 저하한다는 것을 보여준다.

LightGCN은 GCN으로부터 CF에 핵심적인 성분인 이웃 집계를 포함하는 모델이다.

각 유저를 id 임베딩으로 연관시킨 후 유저-아이템 상호작용 그래프로 전파시키면서 임베딩을 재 정의 하였다.

서로 다른 전파 층으로부터 학습된 임베딩을 가중합하여 추론을 위한 최종 임베딩을 얻었다.

모델은 전체적으로 단순하면서도 학습하기 쉽고 성능도 Mult-VAE만큼 달성하였다.

요약

- GCN에서 피쳐 변환과 비선형 활성함수는 CF에 효과가 없다는 것을 확인
- LightGCN이라는 GCN에서 가장 핵심적인 성분만을 포함하도록 설계된 단순화된 모델을 제안
- NGCF과 LightCGN을 동일한 세팅으로 비교함으로써 성능 향상을 설명.

PRELIMINARIES

NGCF Breif

NGCF 설명

피쳐 변환

GCN에서는 단어와 같은 의미적 임베딩이 있었지만,

CF에서는 아이디와 같이 의미가 있지는 않은 형태의 임베딩이라서 다층 비선형 활성함수와 피쳐변환을 활용하는 것에 이점이 없다.

Empirical Exploration on NGCF

GCN에서는 concat했던 임베딩을 sum으로 변경. 같은 사이즈의 임베딩에서의 성능에 관심을 갖고 있어서

NGCF-f : W1, W2(피쳐 변환)을 없앰

NGCF-n : 비선형 활성함수를 없앰

NGCF-fn : 위 둘다 없앰

위 조건 외 다른 하이퍼 파라미터는 NGCF와 동일하게 세팅

Table 1: Performance of NGCF and its three variants.

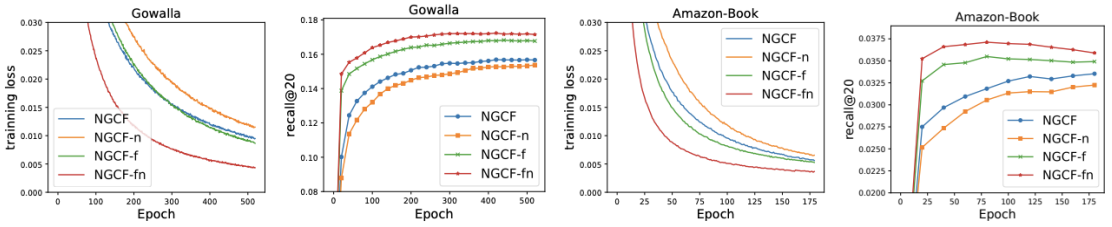
	Gowalla		Amazon-Book	
	recall	ndcg	recall	ndcg
NGCF	0.1547	0.1307	0.0330	0.0254
NGCF-f	0.1686	0.1439	0.0368	0.0283
NGCF-n	0.1536	0.1295	0.0336	0.0258
NGCF-fn	0.1742	0.1476	0.0399	0.0303

데이터 셋의 2층 세팅의 결과를 적어두었다.

피쳐 변환을 없앤 것이 일관되게 개선된 것을 볼 수 있다.

비선형 함수 제거는 그 단일로는 성능이 저하되지만, 피쳐 변환과 함께 제거되었을 때는 개선되는 것을 볼 수 있다.

1. 피쳐 변환을 더하는 것은 NGCF에 안좋은 효과를 냈다.
2. 비선형 함수 추가는 피쳐변환이 포함되어 있을 때만 약간 효과를 냈고, 피쳐변환 없이는 없는 것이 나았다.
3. 전체적으로 봤을 때, 피쳐변환과 비선형 활성화함수는 오히려 안좋은 영향을 주기 때문에, 동시에 제거하는 NGCF-fn이 큰 개선을 보인다.



(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.

추천을 위한 모델을 설계할 때는, 어떤 연산이 의미 있는지 확인해보는 것이 중요하다.

그렇지 않으면 모델이 불필요하게 복잡해지고, 학습 난이도가 올라가 모델의 효과를 떨어뜨릴 수 있다.

METHOD

LightGCN

Light Graph Convolution (LGC)

피쳐 변환과 비선형 활성함수를 제거

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

분모는 GCN 표준을 따른 정규화로, 합성곱 연산을 하면서 임베딩이 계속 커지는 것을 막을 수 있다.

셀프 커넥션을 제거하고, 이웃의 정보만을 집계하는 방식으로 변경

Layer Combination and Model Prediction

LightGCN에서 학습가능한 파라미터는 0층에 있는 임베딩이 유일하다.

그 임베딩이 주어지고 나면, 그 상위 층은 3번의 식을 활용하여 계산된다.

k 층의 LGC 후에, 임베딩을 연결하여 유저나 아이템의 최종 표현을 얻을 수 있다.

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}, \quad (4)$$

a_k는 양수로, k 층의 중요도를 의미한다.

수동으로 하이퍼 파라미터 튜닝을 할 수 있지만, 여기서는 간단히 하기위해 k+1로 나누어 사용했다.

최종 표현을 만드는데 층 결합을 사용한데는 3가지 이유가 있다.

1. 층의 수가 높아질 수록 임베딩이 오버스무딩 된다.
2. 다른 층의 임베딩은 의미적으로 다른 특징을 포착한다. 그러므로 결합하는 것은 더 종합적이다
3. 다른 층의 임베딩ㅇ르 가중합 하는 것은 GCN의 셀프 커넥션과 같은 역할 을 할 수 있다.

모델 예측은 유저,아이템 최종 표현을 내적하는 것으로 정의된다.(이는 추천을 위한 랭킹스코어 생성에도 사용된다.)

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i, \tag{5}$$

Matrix Form

실행과 기존 모델과의 논의를 쉽게하기 위해 행렬 형태의 LightGCN을 제공

유저-아이템 상호작용 행렬을 R (M(user) x N(item) 차원)이라고 할 때, 인접 행렬을 다음과 같이 얻을 수 있다.

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

0층의 임베딩 행렬

$$\mathbf{E}^{(0)} \in \mathbb{R}^{(M+N)\times T}$$

각 층의 임베딩

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

D는 인접행렬의 대각 성분의 값으로 최종 임베딩 행렬은 다음과 같다.

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

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

Model Analysis

Relation with SGCN

GCN의 복잡한 구조가 필요 없다고 말하면서 비선형성을 지우고 가중치 행렬을 하나로 줄였다.

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

위와 같이 정의되고, I는 단위 행렬로, 셀프 커넥션을 의미한다.

최종 임베딩은 아래와 같이 표현할 수 있다.

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

Relation with APPNP

GCN의 변형인 APPNP

오버 스무딩의 위험 없이 긴 거리를 전파할 수 있는

패이지 랭크의 텔레포트 아이디어에서 영감을 얻어, APPNP는 각 전파 층을 시작 피쳐(예. 0층 임베딩)를 가지고 간다. 이것은 지역성을 보전하는 것과 많은 이웃으로부터의 정보를 활용하는데 균형을 맞출 수 있다.

APPNP에서의 전파 층은 다음과 같이 정의

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

베타는 텔레포트 확률로 전파에서 시작 특징을 보전하는 것을 통제하기 위해 둔다.

A틸다는 정규화된 인접 행렬

최종 층은 다음과 같이 표현할 수 있다.

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

식 8에서 a_k를 쓰는 것이 APPNP의 임베딩을 완전히 복원할 수 있다.

적당한 알파를 설정함으로써 큰 긴 범위의 모델링에서 오버스무딩을 컨드롤 할 수 있는 오버스무딩에 대한 강점을 공유한다.

작은 차이는 셀프커넥션으로, LightGCN에서는 다른 층의 가중 합과 중복이다.

Second-Order Embedding Smoothness

LightGCN의 선형성과 단순성으로 임베딩을 원활하게 하는 방법에 대한 인사이트를 얻을 수 있다.

2층 LightGCN을 설명을 위해 사용

직관적으로 아이템 인터렉션을 갖는 유저가 스무딩 되는 2층을 예시로 취했다.

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

보듯이, 만약 다른 유저 v가 타겟 유저 u에 공통으로 상호작용했다면, v의 u에 대한 스무딩 세기는 계수로 측정될 수 있다.

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

이것은 2단계 이웃 v→u에 대한 영향을 설명한다.

1. 같이 상호작용한 아이템이 많을 수록 영향도가 커지고
2. 같이 상호작용한 아이템의 인기도가 낮을수록 (개인화된 선호를 잘 나타내서) 영향도가 커지고
3. v의 활동성이 작을수록 영향도가 커진다.

이러한 설명 가능성은 유저 유사도를 측정하데 CF의 가정에 적합하다는 것을 말한다.

동일한 식으로 아이템에 대한 비슷한 분석도 할 수 있다.

Model Training

0층만 학습 가능한 파라미터로 두기 때문에 MF와 동일한 복잡성을 간진다.

BPR로스 사용하여 관측된 항복에 대한 예측이 관측되지 않은 것 보다 높게 했다.

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

람다는 L2 정규화 세기

아담 옵티마이저 사용

네거티프 샘플링은 사용하지 않음

드랍아웃도 사용하지 않음 - 피쳐 변환을 사용하지 않기 때문에, l2 정규화가 오버피팅을 막는데 충분하다고 봄.

a도 어텐션으로 파라미터화 할 수 있었지만, 훈련데이터로 학습한 a는 개선을 이끌어내지 못하는 것을 확인했다.(훈련 데이터는 언노운 데이터에 대해 일반화하는 좋은 알파를 찾지 못하기 때문이다.)

검증 데이터로 해봤지만, 아주 조금 개선되었고, 이것도 제외시켰다.

EXPERIMENTS

Experimental Settings

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

Compared Methods

- Mult-VAE
  - 데이터는 멀티노미알 분포로부터 생성된다는 가정
- GRMF
  - 라플라시안 레귤라이저로 임베딩 스무딩

Hyper-parameter Settings

embedding_size : 64

초기화 : 사비에르

opt : adam

lr : 0.001

batch_size : 1024

L2 정규화 계수 람다 : 1e-4

층 결합 계수 a_k : 1/ (1+k) (k : 1~4)

epoch : 1000

Performance Comparison with NGCF

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)

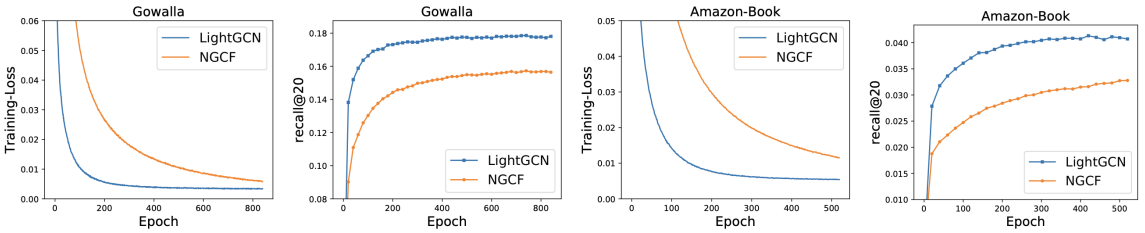


Figure 3: Training curves of LightGCN and NGCF, which are evaluated by training loss and testing recall per 20 epochs on Gowalla and Amazon-Book (results on Yelp2018 show exactly the same trend which are omitted for space).

- LightGCN이 더 성능이 좋다.
- NGCF-fn 보다 도 좋다. (셀프커넥션, 합성곱에서 유저-아이템 임베딩 상호작용, 드랍아웃 등 차이)
- 층을 늘릴수록 성능 개선. 이익은 줄어듬
- 훈련도 LightGCN이 더 잘된다.

Performance Comaprsion with State-of-the-Arts

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**

LightGCN이 높은 성능을 보임

a_k를 개선할 여지가 있음

Ablation and Effectiveness Analyses

Impact of Layer Combination

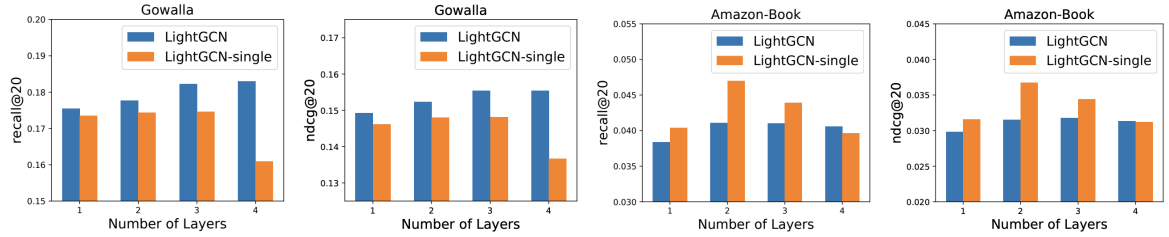


Figure 4: Results of LightGCN and the variant that does not use layer combination (i.e., LightGCN-single) at different layers on Gowalla and Amazon-Book (results on Yelp2018 shows the same trend with Amazon-Book which are omitted for space).

- LightGCN-single
  - 층이 증가함에 따라 처음에는 성능이 오르다가 떨어지는 것을 볼 수 있다.
  - 대부분 2층에서 피크 포인트를 가짐
  - 1차 2차 이웃에 의해 스무딩 된 임베딩이 CF에 유용하다는 것을 보여준다.
- LightGCN
  - 층이 증가함에 따라 성능이 나아지는 것을 보임. 낮아지는 경우는 없음
  - 이것이 층결합이 오버스무딩에 대해서 효과가 있음을 보여준다.
- 두 방법 비교
  - LightGCN-single은 특정 층의 가중치를 1로 두고 나머지를 0으로 둔 것과 같다.

- LightGCN에서는 a_k를 1/(k+1)로 균등하게 세팅하고 있다.
- 이러한 것들을 고려해봤을 때 a_k를 튜닝하여 개선할 여지가 있다.

Impact of Symmetric Sqrt Normalization

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.

- 아이템/ 유저 한쪽에 대해서만 정규화하거나, L1을 적용 해봄
- 정규화를 삭제하면 수치적으로 불안정하고 Nan 이슈가 발생하기도 해서 적용하지 않음
- 양쪽에 정규화 세팅을 하는게 가장 좋다.
  - 한쪽이라도 제거하면 성능 손실이 크다
- 두번째 좋은 세팅은, L1을 왼쪽에만 사용하는 것이다.
  - 이것은 인접행렬을 차수에 따라 확률적 행렬로 하는 것과 같다.
- 동일하게 양쪽을 정규화하는 것은 sqrt 정규화에서는 도움이 되지만, l1을 취하면 성능이 떨어진다

Analysis of Embedding Smoothness

임베딩 스무딩을 측정해보는 것은 LightGCN의 효과를 보는 키

유저 임베딩의 스무딩 정도를 다음과 같이 정의

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

임베딩에 l2놈을 취하면, 임베딩 스케일의 효과가 사라진다.

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

lgc가 추전에 더 맞는 것을 볼 수 있다.

Hyper-parameter Studies

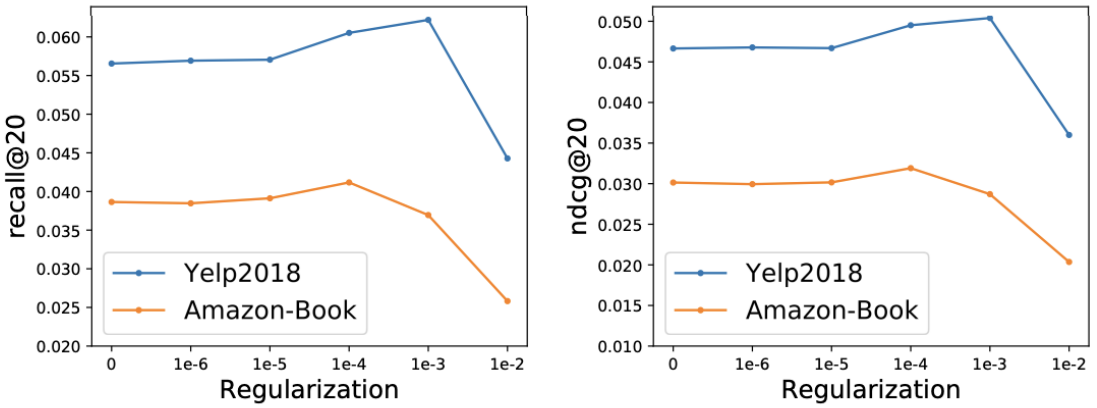


Figure 5: Performance of 2-layer LightGCN w.r.t. different regularization coefficient λ on Yelp and Amazon-Book.

LightGCN이 람다에 민감하지 않음을 보여주고, 람다가 0이어도 NGCF보다 나은 성능을 보인다.

LigthGCN이 덜 오버피팅되는 것을 볼 수 있는데, 이것은 학습 가능한 파라미터가 0층의 ID임베딩 뿐이기 때문에 학습과 정규화가 쉽ㄴ다.

RELATED WORK

Collaborative Filtering

Graph Methods for Recommendation

CONCLUSION AND FUTURE WORK

GCN에서 CF에 불필요한 복잡한 디자인에 대해서 다뤘다.

핵심적인 2가지 항목(light graph convolution, layer combination)만 포함하는 LightGCN을 제안

light graph convolution에서는 피쳐 변환과 비선형 활성함수를 제거했다.

층 결합에서는 셀프커넥션과 같은 효과를 내고 그것은 오버스무딩을 통제하는데 도움이된다.

학습하기 쉬울수록, 일반화 능력이 낫고, 더 효과적이었다.

실제 어플리캐이션에서 연결 그래프가 성행하면서, 추천에서도 중요해졌다.