Recommendation System

Neural Collaborative Filtering(2017, Xiangnan He)

https://arxiv.org/pdf/1708.05031.pdf

Abstract

Implicit Feedback에 기반하는 핵심적인 문제인 Collaborative Filtering을 Neural Network 방식으로 개발

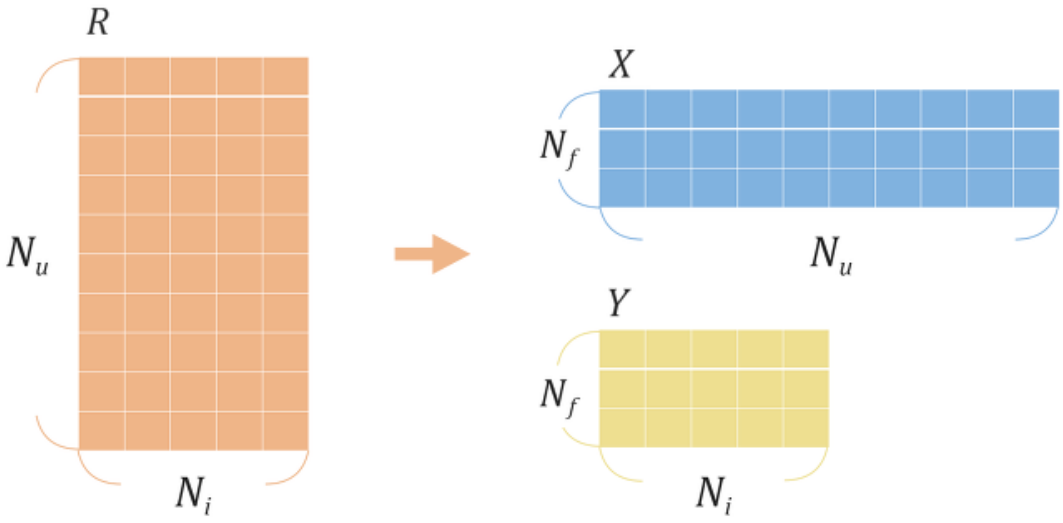
▼ Explict Feedback vs Implicit Feedback

Explicit Feedback은 어떠한 영화를 보고 1~5 평점을 매기거나, 좋아요 or 싫어요 등으로 어떠한 것에 대하여 피드백을 명시적으로 확실하게 주는 것을 말한다. 명시적으로 표현된 신뢰가능한 Feedback이기 때문에 상대적으로 모델의 성능향상에 큰 도움이 될 수도 있지만, 수집하기 어렵다는 단점이 존재한다.

Implicit Feedback은 어떠한 영화를 여러번 보는 행위처럼 확실하게 피드백을 명시적으로 보여준 것은 아니지만, 은연 중에 유저의 피드백을 나타낸다. 특정 영화를 여러 번 시청한 것이 좋다고 확신할 수는 없지만, 관심이 있다고 유추할 수는 있다. 명시적으로 표현한 확실히 신뢰할 수 있는 Feedback은 아니기에 모델의 성능을 향상시키는데에 비교적 효과가 떨어질 수도 있지만, 예를 들어 클릭정보는 자동으로 모아지기 때문에 수집하기 쉽다는 장점이 있다.

- 추천 시스템에 딥러닝을 활용한 연구들을 살펴보면, 대부분 아이템의 설명 텍스트, 노래의 음향정보와 같은 부가적인 정보를 활용하는 경우가 많음
- Collaborative Filtering의 Key Factor을 모델링하기 위해서는 유저와 아이템 간의 interaction이 요구되며, 여전이 Matrix Factoriation 기법에 의존하여 유저와 아이템의 Latent Features의 내적 과정을 거침

▼ Matrix Factorization



평점 행렬은 대부분의 정보가 존재하기 않는 Sparse Matrix 형태이므로, Matrix Factorization을 통하여 학습을 진행

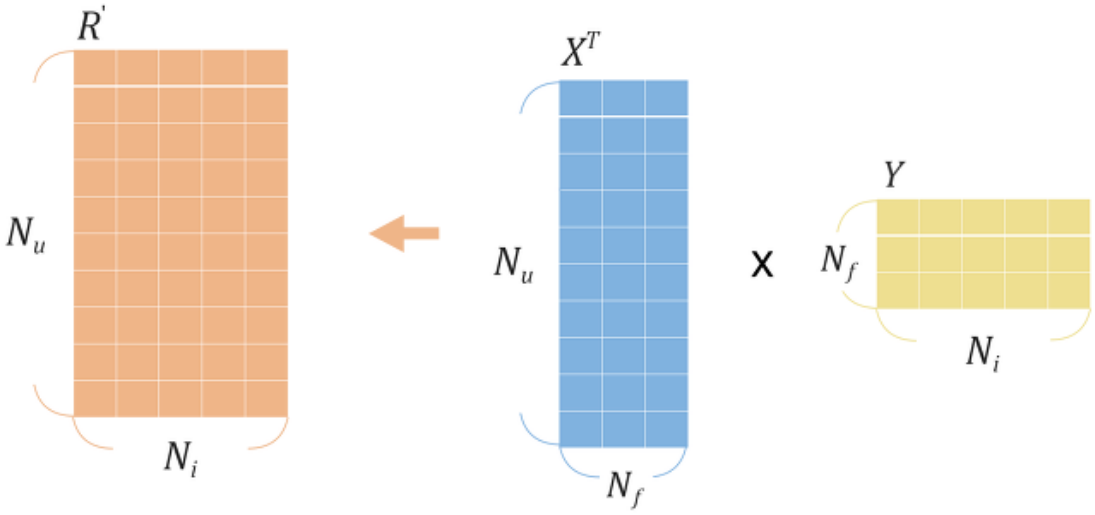
평점행렬을 Nf 라는 잠재차원으로 사용자와 아이템 Latent Factor 행렬로 분해

R은 사용자가 아이템에 남긴 평점을 행렬로 나타낸 것

- R의 크기는 사용자 수(Nu) x 아이템의 수(Ni)

Nf는 Matrix Factorization 학습 시에 정하는 임의의 차원 수

X와 Y는 사용자와 아이템의 Latent Factor 행렬을 나타내며, 우리가 학습시키고자 하는 대상



분해된 행렬을 다시 곱하여 예측 평점 행렬을 계산

내적 과정을 비선형적 관계를 학습할 수 있는 인공신경망 구조로 대체함으로써 NCF라는 일반적인 Architecture 제시

- NCF framework는 MF의 일반화된 모델
- User-Item Interaction Function을 MLP를 활용해 학습함으로써 비선형성을 추가

Introduction

다양한 Collaborative Filtering 방법 중에서, Matrix Factorization이 가장 대중적인 알고리즘

- 유저와 아이템의 Latent Features를 벡터화함으로써, 유저와 아이템을 한 공간으로 투영

Matrix Factorization 알고리즘을 개선하기 위해 다양한 노력

- 이웃 기반 모델과 통합
- 아이템 데이터의 토픽 모델링을 결합
- Feature를 포괄적으로 사용하기 위해 Fatorization Machine으로 확장

하지만 Matrix Factorization은 내적을 통하여 Interaction을 학습하므로, 성능 향상에 방해

- Latent Feature의 곱셈을 선형으로 단순하게 결합하는 내적의 경우 데이터의 복잡한 관계를 학습함에 있어 충분치 못한다고 해석할 수 있음

사용자의 선호도를 간접적으로 반영하는 Implicit Feedback에 초점을 맞춤

- Explicit Feedback에 비해 콘텐츠 제공자가 훨씬 쉽게 수집하는 것이 가능
- 사용자 만족도가 관측되지 않고, 부정적인 피드백이 부족하기 때문에 활용하기 어려움

논문의 주요 내용

1. 유저와 아이템의 Latent Feature를 모델링하는 인공신경망 구조를 제시하고 구체화하여, 인공신경망 기반 Collaborative Filtering 모델인 NCF를 설계
2. MF 알고리즘을 기본적으로 따르는 NCF는 MLP를 통해 고차원의 비선형적인 특징을 학습
3. 딥러닝 방식의 Collaborative Filtering의 성능 향상과 효과성

PRELIMINARIES

2-1. Learning from Implicit Data

$$y_{u,i} = \begin{cases} 1, & \text{if interaction(user } u\text{, item } i\text{) is observed} \\ 0, & \text{otherwise} \end{cases}$$

유저와 아이템의 수를 M과 N으로 하는 R행렬을 정의할 때, feedback의 발생 여부를 기준으로 0, 1로 행렬을 정의

- Implicit Feedback이란 유저와 아이템의 Interaction, 발생여부를 선호한다고 정의하는 것은 무리
- Interaction이 발생하지 않아도, 싫어하는 것이 아닌 Interaction이 발생하지 않은 것임

Implicit Feedback의 추천 과정에서 발생하는 문제

- 아이템을 랭킹화하기 위한 Interaction이 존재하지 않는 아이템 Set의 score를 예측하는 과정에서 발생

$$\hat{y}_{u,i} = f(u, i|\Theta),$$
$$where\ f\ \text{ is interaction function, } \Theta \text{ is model parameters}$$

*f*는 유저와 아이템이 입력으로 주어졌을 때, score를 return하는 interaction function

Θ 는 해당 function의 parameter

해당 function의 parameter를 최적화할 수 있는 목적함수는 두가지가 존재

- $L_1 = min\frac{1}{2}(\hat{y}_{u,i} - y_{u,i})^2$: pointwise loss
- $L_2 = max(0, f(y_{unobs}) - f(y_{obs}) + \alpha)\ \ s.t\ \ rank(y_{obs}) > rank(y_{unobs})$: pairwise loss

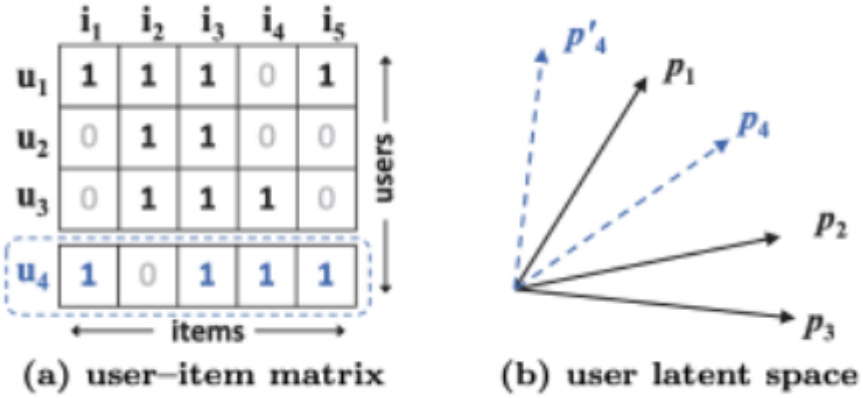
pointwise loss의 경우에는 실제값과 예측값의 차이를 최소화(minimize) 하는 목적 함수로 보통 회귀 문제에서 사용

- negative feedback 데이터의 부족을 처리하기 위해서, interaction이 발생하지 않은 데이터를 하여금 negative feedback으로 처리하거나 sampling 하여 처리하는 방법을 적용

pairwise loss의 경우에는 관측된 데이터(interaction이 발생한)가 그렇지 않은 데이터에 비해 링킹이 높아야 한다는 아이디어에 기반하여, 관측된 데이터와 관측되지 않은 데이터의 랭킹 차를 최대화(maximize)하는 목적 함수

2-2. Matrix Factorization

MF, Matrix Factorization은 유저 *u*와 아이템 *i*의 latent features인 *p_u, q_i* 의 반복적인 내적을 통해 interaction인 *y_(u,i)*를 추정



(a) user–item matrix (b) user latent space

유사도 순으로 정렬하게 되면 유저 (2,3), 유저 (1,2), 유저 (1,3) 순으로 높은 유사도

$$s_{23}(0.66) > s_{12}(0.5) > s_{13}(0.4)$$

새로운 유저 4가 등장, 다른 유저들과의 유사도가 계산되어 다음과 같은 유사도 관계가 형성되게 되면 MF 모델에서 공유하고 있던 벡터 공간에서 유저 4를 표현하는 것에 있어 한계를 가지게 되고, 결과적으로 ranking loss가 커지는 현상이 발생

$$s_{41}(0.6) > s_{43}(0.4) > s_{42}(0.2)$$

Matrix Factorization의 단순하고 고정된 내적 과정은 저차원 latent space에서 복잡한 유저와 아이템의 관계를 추정하는 것에 한계점

이러한 문제점을 해결하는 방법 중 하나는 매우 큰 K차원의 latent factor를 지정하는 것이나, 모델의 generalization 관점에 있어 부정적인 결과를 낳을 수 있음

NEURAL COLLABORATIVE FILTERING

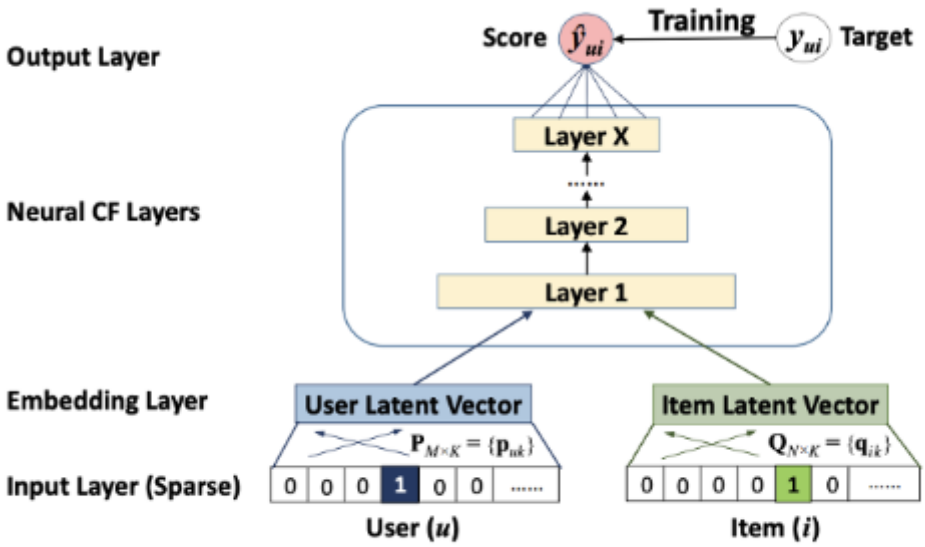


Figure 2: Neural collaborative filtering framework

3-1. General Framework

Input Layer

- 사용자와 아이템의 one-hot encoding이 입력으로 사용

Embedding Layer

- Input인 Sparse Vector를 Dense Vector로 Projection 해주는 Fully Connected Layer
- Dense Vector가 흔히 말하는 유저와 아이템의 Latent Vector
- **N*eural CF Layers*** 라고 할 수 있는 MLP 구조에 입력하게 되면서 score를 예측
- Fully-connected layer를 통해 Dense Vector를 얻는 과정은 Hidden Layer의 Weight가 업데이트되는 결과물

Neural CF Layers

- User Embedding과 Item Embedding은 prediction score로 mapping 시켜주는 Neural CF에 입력

- User-Item Interaction의 특정한 Latent Structure를 발견할 수 있도록 Custom 가능

Output Layer

- 예측점수를 예측

3.1.1 Learning NCF

MLP 구조는 최종적으로 특정 유저 u와 특정 아이템 i의 Score를 실제값과의 차이를 최소화하는 Pointwise Loss를 적용하여 추정

오차를 활용하는 Pointwise Loss에서는 Gaussian 분포에 근거하여 관측치를 설명하기 때문에 Interaction의 발생 여부 1 / 0 으로 이루어져 있는 Implicit data에 적용하기 적합하지 않음

Implicit data의 target value가 1일 경우, item i가 user u와 관련이 있고, 0은 아님

- NCF의 출력에 Logisitic이나 Probit Function을 Activation Function으로 사용하여 [0,1] 범위로 제한
- Prediction Score는 i가 u와 얼마나 관련이 있는지로 해석하는 것이 가능
- 데이터가 Bernoulli Distribution을 따른다고 가정할 수 있음

목적함수로 사용하기 위한 Negative Log Likelihood

$$
\begin{aligned}
L &= -\sum_{(u,i)\in\mathcal{Y}} y_{u,i}\, log\, \hat{y}_{u,i} - (\sum_{(u,j)\, in\mathcal{Y}^-} (1-y_{u,i})\, log\, (1-\hat{y}_{u,j})) \\
&= -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} (y_{u,i}\, log\, \hat{y}_{u,i} + (1-y_{u,i})\, log\, (1-\hat{y}_{u,i}))
\end{aligned}
$$

NCF를 최적화하기 위한 목적 함수이며, SGD를 통해 최적화될 수 있고, Binary Cross Entropy Loss와 같은 꼴을 가짐

해당 목적 함수에서 존재하는 y− 은 interaction이 발생하지 않은 데이터

unobserved interaction 데이터에서 negative sampling을 통해 구축

3-2. Generalized Matrix Factorization (GMF)

GMF는 본래의 Matrix Factorization과 크게 다른 점은 없지만 내적인 아닌 element-wise product가 이뤄지고, edge weights와 activation function을 사용

$$\hat{y}_{ui} = a_{out}(h^T(p_u \odot q_i))$$
$$where \;\; a_{out} : \text{sigmoid function}, \sigma(x), \;\; h^T : \text{edge weights}$$

기존 MF와 약간의 다른 구조를 보이는 GMF가 가지는 장점

- h 가중치에 non-uniform 특징으로 학습하면 Latent Dimension의 다양한 중요성을 허용 가능
- a_out에 non-linear function을 적용하게 되면, linear한 MF 모델보다 풍부한 표현력을 가짐

3-3 Multi-Layer Perceptron (MLP)

NCF 모델에서는 유저와 아이템이라는 2가지 벡터를 입력으로 받기 때문에 두 벡터를 연결(concat)하는 것으로 MLP에 대한 입력 벡터를 구성

유저와 아이템 간의 복잡한 interaction 관계를 학습하기 위해서 hidden layer를 여러 개 추가함으로써 flexibility하고 non-linearity한 딥러닝의 장점을 모델에 적용할 수 있음

concat된 입력 벡터가 layer를 통과하면, 선형 계산과 비선형 계산의 반복이 이뤄지게 되고 output layer에서 sigmoid function을 적용함으로써 특정 유저와 특정 아이템 간의 interaction 발생 여부에 대한 확률값을 추정할 수 있음

3-4 Fusion of GMF and MLP

latent feature interaction에 대한 선형 계산의 **GMF**와 비선형 계산의 **MLP**를 NCF 모델 구조에서 함께 사용하게 된다면 각 알고리즘의 장점으로 유저와 아이템 간의 복잡한 관계를 학습함에 있어 보다 시너지 효과를 낼 것

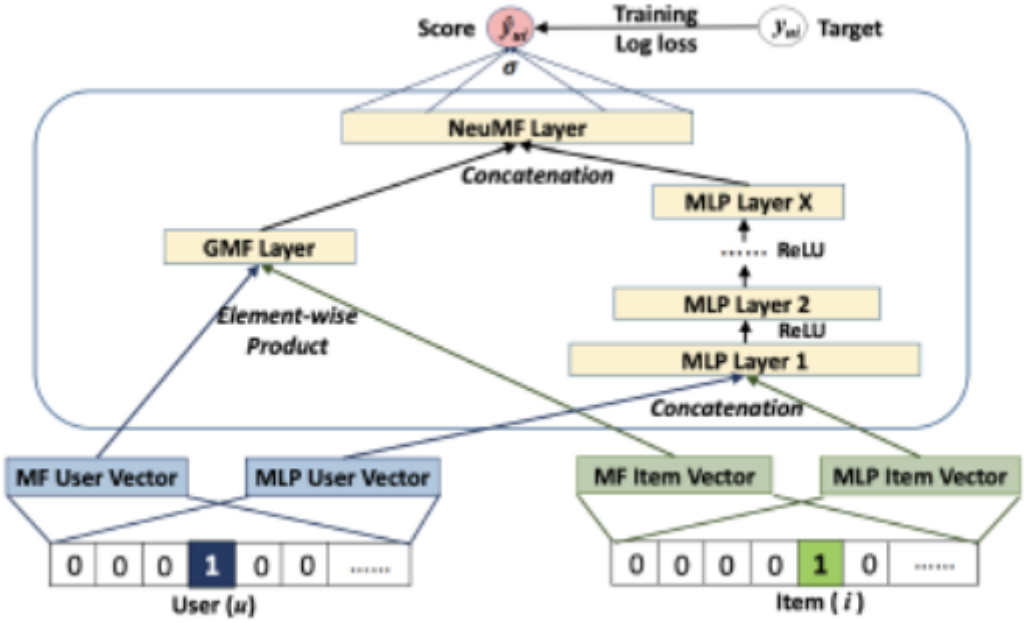


Figure 3: Neural matrix factorization model

가장 단순한 방법으로는 GMF와 MLP 모두 같은 embedding layer를 공유하는 것

각 모델의 결과값을 결합하여 확률값을 추정

- 같은 embedding layer를 공유한다는 것은 같은 embedding vector를 사용
- 하지만 이렇게 embedding layer를 공유하게 되면, 임베딩 차원이 같아야 함을 의미
  각 모델에 적합한 임베딩 차원을 택할 수 없기 때문에 최적의 앙상블 결과를 얻을 수 없을 수도 있음

하이브리드 모델에 유연성을 부여하기 위해서, GMF와 MLP에 다른 embedding layer를 사용함으로써 다른 embbedding vector를 사용하고 각 모델의 결과값을 연결(concat)하여 확률값을 추정할 수 있음

4. EXPERIMENTS

설계한 NeuMF의 모델 성능을 측정하기 위한 실험에 대해 다뤄보고 있으며, 다음 3가지 질문을 기반으로 실험을 진행

RQ1 제시한 NCF 기법이 implicit CF 기법에서 높은 성능을 보이는가?

RQ2 제안한 최적 구조가 추천 과제에 있어 어떻게 작동하는가?

RQ3 DNN 구조가 유저 아이템의 interation을 학습함에 있어 효과적인가?

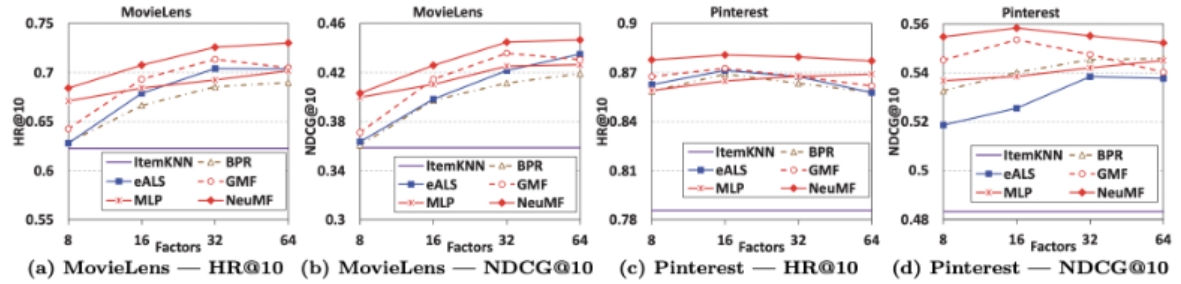
4-1 Experimental Settings

Table 1: Statistics of the evaluation datasets.

Dataset	Interaction#	Item#	User#	Sparsity
MovieLens	1,000,209	3,706	6,040	95.53%
Pinterest	1,500,809	9,916	55,187	99.73%

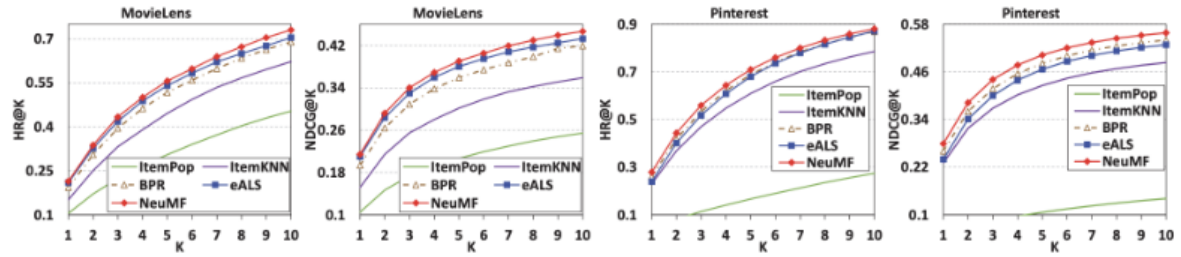
MovieLens와 Pinterest, 2개의 데이터에서 실험을 진행

4-2 Performance Comparison (RQ1)



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

Figure 4: Performance of HR@10 and NDCG@10 *w.r.t.* the number of predictive factors on the two datasets.



(a) MovieLens — HR@K (b) MovieLens — NDCG@K (c) Pinterest — HR@K (d) Pinterest — NDCG@K

Figure 5: Evaluation of Top-K item recommendation where K ranges from 1 to 10 on the two datasets.

성능 평가를 위한 HR@10과 NDCH@10에 대한 각 모델 별 성능 평가표

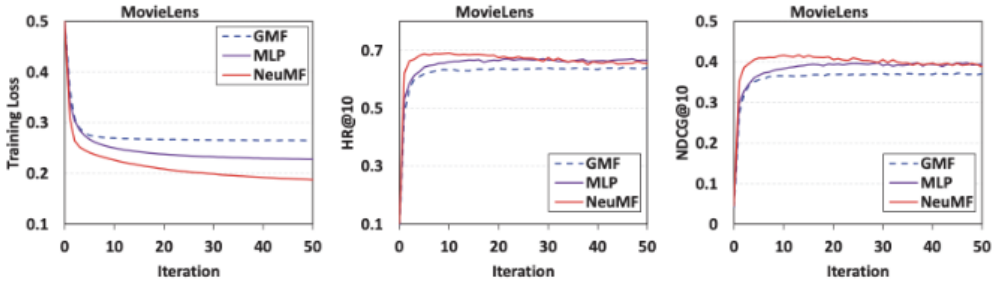
상단 성능 평가표의 x축인 Factors는 마지막 hidden layer의 차원을 의미

- model capability 그리고 predictive factors

하단 성능 평가표는 각 성능 지표의 Top-K를 의미

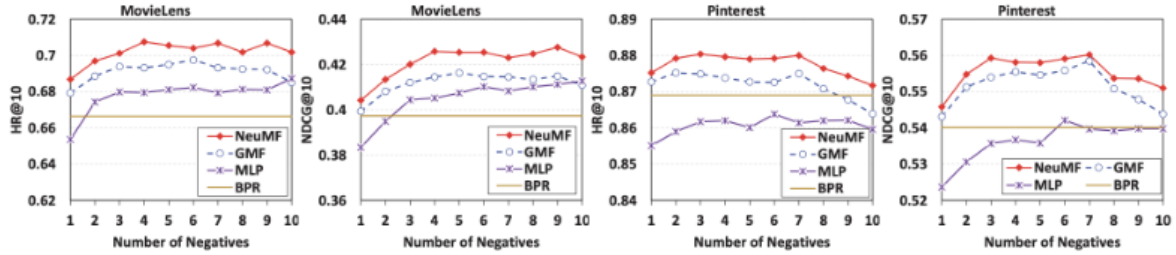
- NeuMF가 가장 좋은 성능

4-3 Log Loss with Negative Sampling (RQ2)



(a) Training Loss (b) HR@10 (c) NDCG@10

Figure 6: Training loss and recommendation performance of NCF methods *w.r.t.* the number of iterations on MovieLens (factors=8).



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

Figure 7: Performance of NCF methods *w.r.t.* the number of negative samples per positive instance (factors=16). The performance of BPR is also shown, which samples only one negative instance to pair with a positive instance for learning.

Iteration에 따른 각 history와 negative sampling ratio에 대한 성능 평가표

- 학습이 진행됨에 따라 loss가 줄어들고 성능이 향상됨으로써 제안한 NeuMF가 implicit data에 적합
- pairwise objective function은 positive와 negative가 1대1의 쌍이 이뤄져야하는 반면에 pointwise loss을 통해 샘플링 비율을 조절할 수 있었고, negative sampling의 비율을 다르게 함으로써 성능을 비교할 수 있었음

4-4 Is Deep Learning Helpful? (RQ3)

Table 3: HR@10 of MLP with different layers.

Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.452	0.628	0.655	0.671	**0.678**
16	0.454	0.663	0.674	0.684	**0.690**
32	0.453	0.682	0.687	0.692	**0.699**
64	0.453	0.687	0.696	0.702	**0.707**
Pinterest					
8	0.275	0.848	0.855	0.859	**0.862**
16	0.274	0.855	0.861	0.865	**0.867**
32	0.273	0.861	0.863	**0.868**	0.867
64	0.274	0.864	0.867	0.869	**0.873**

Table 4: NDCG@10 of MLP with different layers.

Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.253	0.359	0.383	0.399	**0.406**
16	0.252	0.391	0.402	0.410	**0.415**
32	0.252	0.406	0.410	**0.425**	0.423
64	0.251	0.409	0.417	0.426	**0.432**
Pinterest					
8	0.141	0.526	0.534	0.536	**0.539**
16	0.141	0.532	0.536	0.538	**0.544**
32	0.142	0.537	0.538	0.542	**0.546**
64	0.141	0.538	0.542	0.545	**0.550**

유저와 아이템의 관계를 학습함에 있어 DNN을 적용해본 간단한 연구였지만, 추천 과제에 있어 DNN 구조가 굉장히 적합

해당 성능 평가표는 마지막 hidden layer의 capacity인 factor와 layer-depth에 따른 각 성능 지표

- layer가 깊어질수록 높은 성능을 보이고 있음