Graph Convolutional Matrix Completion

Graph Neural Networks,
Matrix Completion
Collaborative Filtering

추천시스템 Boot Camp 4기 강석우

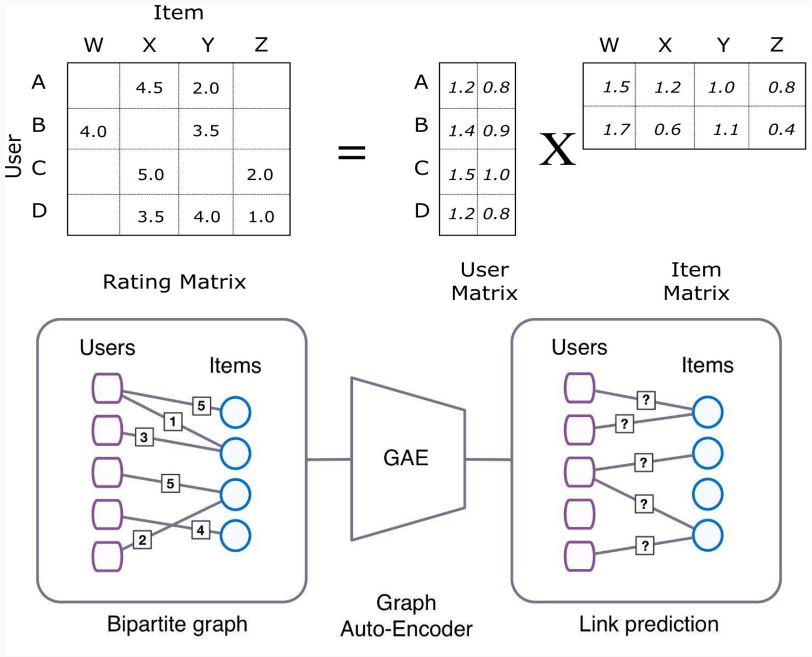
Abstract

추천 시스템을 위한 Matrix Completion을 graph의 link prediction 로 접근

관측된 Interaction data 를 Bipartite User-Item graph로 표현해서 사용

Graph Auto-Encoder Framework 제안 (Differentiable message passing)

표준 CF 모델 들과 본 논문의 모델 비교로 성능 비교



1. INTRODUCTION-1

추천 시스템이 중요한 역할을 하고 있고 Matrix Completion은 중요한 서브 과제!

Interaction Data를 이분(Bipartite) Graph 형태로 만듬. (User와 Item 노드)

User와 Item 노드 간 연결 강도를 Rating으로 사용 -> 노드 간 link를 예측

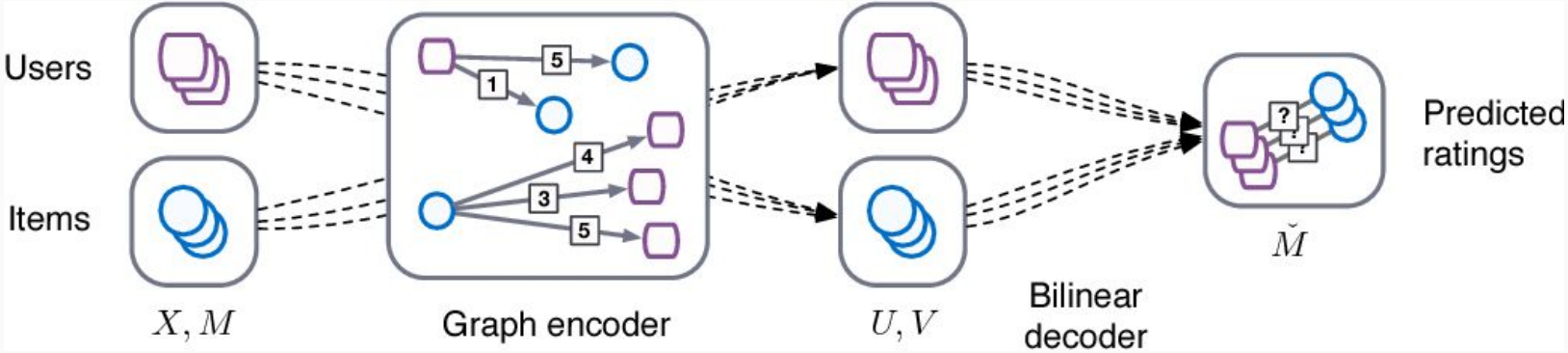
→ 외부에 구조화된 User, Item 의 고유 Feature를 함께 사용 가능 (Cold-Start 완화)

Graph Convolutional - Matrix Completion (GC-MC) 를 사용

1. INTRODUCTION-2

인코더가 bipartite interaction data로 User와 Item Latent Feature Vector 생성

두 Latent Feature Vector 를 Bilinear 디코더가 사용해서 Link를 생성한다.



Graph Neural Network

(http://www.secmem.org/blog/2019/08/17/gnn/)

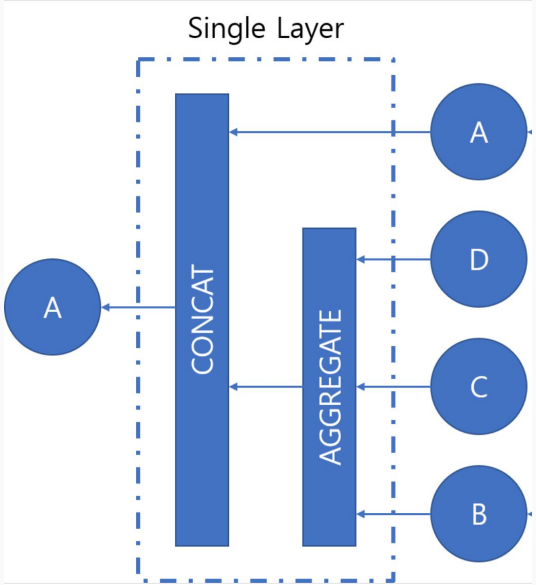
그래프 구조의 데이터 전용 인공 신경망

인접한 노드들의 정보와 (Node Feature)

노드 간 연결(Adjacency)을 함께 사용하는 구조

이웃 노드들의 정보를 모으고 (Aggregate)

모은 정보와 자기 자신을 값을 합침 (Concatenate)



Graph Convolutional Network

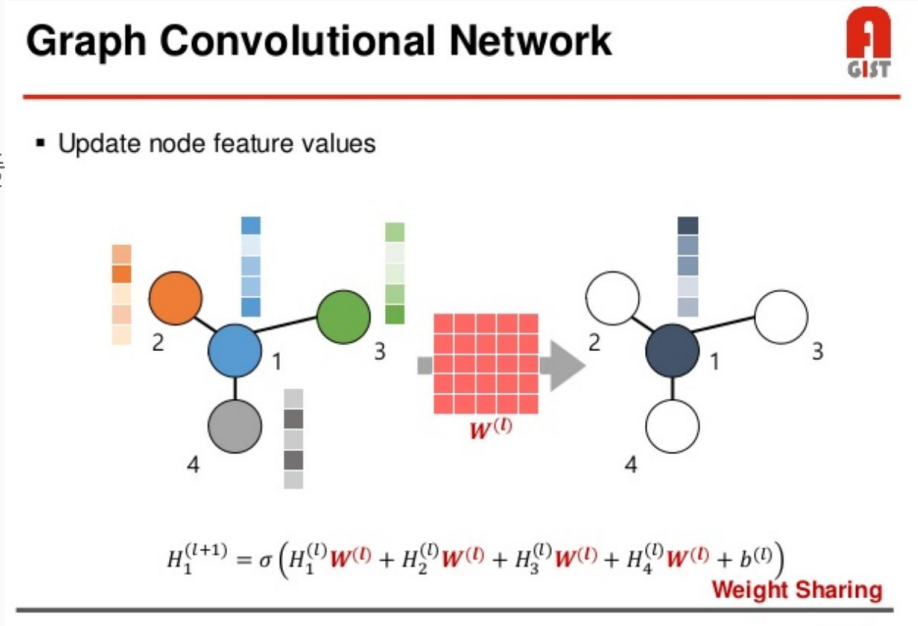
Convolutional : 이웃하는 픽셀들에 대해

동일한 파라미터를 사용해서 피처를 추출

Graph Convolutional Network 에서는

Neighborhood aggregation 에

동일한 파라미터(Weight Share)를 사용



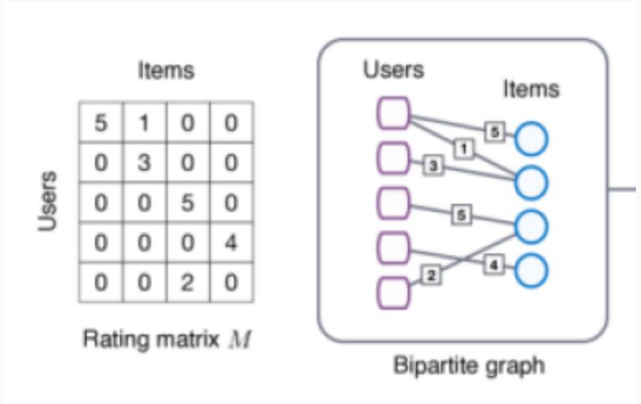
2. MATRIX COMPLETION AS LINK PREDICTION IN BIPARTITE GRAPHS

Rating Matrix는 전체 유저 수 X 전체 아이템 수로 구성$N_u \times N_v$

Rating Matrix를 undirected(방향이 없는) 그래프로 표현한다$G = (W, E, R)$

$W = W_u \cup W_v \; u_i \in W_u \, (user\, i) \, , \, v_j \in W_v (item\, j))$

$(u_i, r, v_j) \in E \, (r = rating\, level \,)$



2.1 Revisiting graph auto-encoders

그래프 오토인코더는 end-to-end 의 비지도 학습 모델로 처음 소개되고, 방향성이 없는 그래프의 link prediction에 사용 되었음.

인코더 모델, Z = f(X, A) $[Z_u, Z_v] = f(X_u, X_v, M_1, \ldots, M_R)$

A : 그래프의 인접 행렬 shape = (N, N)

X : 노드의 피처로 만들어진 행렬 (N, D)

Weight Matrix(D, H) 를 학습하며 조정

Z : 노드 임베딩 벡터 (N, H) (Node-Level Output)

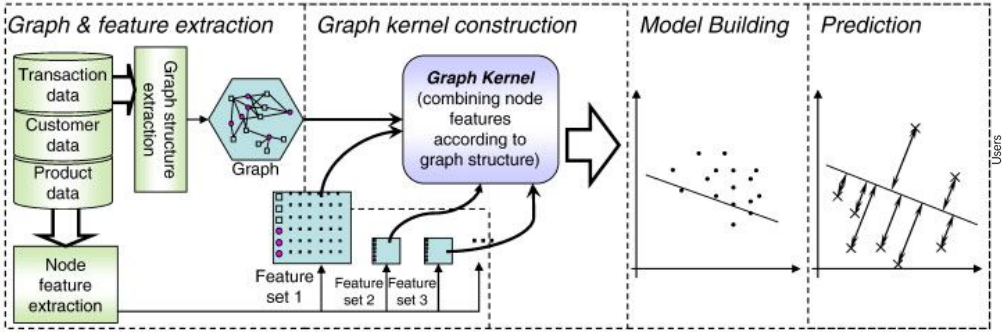
디코더 모델 $\check{A}$ = g(Z) $\check{M} = g(Z_u, Z_v)$

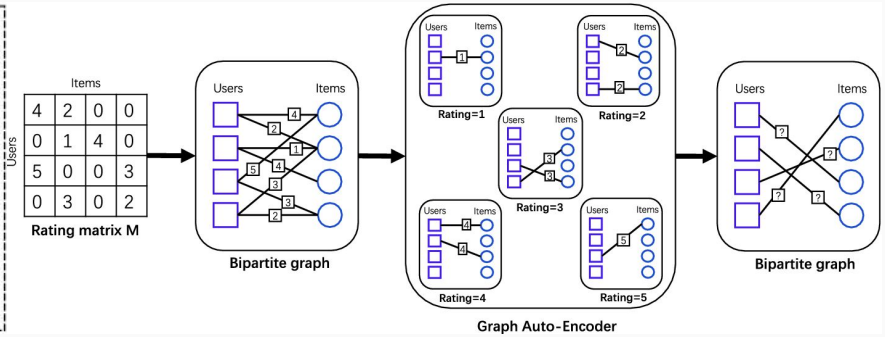
Z : 아이템과 유저 노드의 임베딩 벡터 쌍

$\check{A}$: 그래프의 인접 행렬을 예측한다.

학습 - Ground-Truth와 Predicted Rating 간 Error 최소화

Recommendation as link prediction in bipartite graphs





2.2 Graph convolutional encoder

그래프의 모든 위치에서 동일한 가중치 (Weight Sharing) W_r

Rating Type(Edge Type) 별로 분리된 연산을 다르게 한다. (Edge-Type Specific

$$h_i = \sigma\left[\text{accum}\left(\sum_{j\in\mathcal{N}_{i,1}} \mu_{j\to i,1}, \ldots, \sum_{j\in\mathcal{N}_{i,R}} \mu_{j\to i,R}\right)\right], \qquad \mu_{j\to i,r} = \frac{1}{c_{ij}} W_r x_j \,.$$

(graph convolution layer)

$$u_i = \sigma(W h_i)\,.$$ (dense layer)

c는 정규화 상수이고 (left, symmetric), accum()은 (stack, sum), Activation (Relu)

Bilinear decoder

각 rating level을 분리된 Class로 취급하고, Softmax 함수를 사용해서 각 rating level에 대해 확률 분포를 구한다.

$$p(\check{M}_{ij} = r) = \frac{e^{u_i^T Q_r v_j}}{\sum_{s \in R} e^{u_i^T V_s v_j}},$$

$$\check{M}_{ij} = g(u_i, v_j) = \mathbb{E}_{p(\check{M}_{ij}=r)}[r] = \sum_{r \in R} r\, p(\check{M}_{ij} = r)\,.$$

Model Training

$$\mathcal{L} = -\sum_{i,j;\Omega_{ij}=1} \sum_{r=1}^{R} I_{r,M_{ij}} \log p(\check{M}_{ij} = r)\,,$$

관측된 rating 만 사용해서 학습을 한다. 아래는 온전한 Loss 식 (0, 1에 대해서)

$$\begin{aligned} &maximize \;\; y \log \hat{y} + (1-y) \log(1-\hat{y}) \\ &= minimize \;\; -y \log \hat{y} - (1-y) \log(1-\hat{y}) \end{aligned}$$

Model Training

Mini-batching (Stochastic)

effective means of regularization, reduces the memory requirements

Node dropout

more efficient in regularization

weight sharing

rating level에 대해서도

Input feature representation and side information

$$h_i = \sigma\left[\text{accum}\left(\sum_{j \in \mathcal{N}_{i,1}} \mu_{j \to i,1}, \ldots, \sum_{j \in \mathcal{N}_{i,R}} \mu_{j \to i,R}\right)\right],$$

$$\mu_{j \to i,r} = \frac{1}{c_{ij}} W_r x_j \,.$$

(graph convolution layer)

$$u_i = \sigma(W h_i) \,.$$ (dense layer)

graph convolution layer (severe bottle neck) 이 아닌 dense layer 에 넣는다.

$$u_i = \sigma(W h_i + W_2^f f_i) \;\;, \;\; \text{with} \;\;\; f_i = \sigma(W_1^f x_i^f) \,,$$

EXPERIMENTS

Dataset	Users	Items	Features	Ratings	Density	Rating levels
Flixster	3,000	3,000	Users/Items	26,173	0.0029	0.5, 1, ..., 5
Douban	3,000	3,000	Users	136,891	0.0152	1, 2, ..., 5
YahooMusic	3,000	3,000	Items	5,335	0.0006	1, 2, ..., 100
MovieLens 100K (ML-100K)	943	1,682	Users/Items	100,000	0.0630	1, 2, ..., 5
MovieLens 1M (ML-1M)	6,040	3,706	—	1,000,209	0.0447	1, 2, ..., 5
MovieLens 10M (ML-10M)	69,878	10,677	—	10,000,054	0.0134	0.5, 1, ..., 5

Table 1: Number of users, items and ratings for each of the MovieLens datasets used in our experiments. We further indicate rating density and rating levels.

Accumulate Func (Stack vs Sum), ordinal weight sharing in the encoder Normalization (Left, Symmetric), Drop out rate, weight sharing in decoder, Layer size

Experiments

Model	ML-100K + Feat
MC [3]	0.973
IMC [11, 31]	1.653
GMC [12]	0.996
GRALS [25]	0.945
sRGCNN [22]	0.929
GC-MC (Ours)	0.910
GC-MC+Feat	**0.905**

Table 2: RMSE scores[6] for the MovieLens 100K task with side information on a canonical 80/20 training/test set split. Side information is either presented as a nearest-neighbor graph in user/item feature space or as raw feature vectors. Baseline numbers are taken from [22].

Model	Flixster	Douban	YahooMusic
GRALS	1.313/1.245	0.833	38.0
sRGCNN	1.179/0.926	0.801	22.4
GC-MC	**0.941/0.917**	**0.734**	**20.5**

Table 3: Average RMSE test set scores for 5 runs on Flixster, Douban, and YahooMusic, all of which include side information in the form of user and/or item graphs. We replicate the benchmark setting as in [22]. For Flixster, we show results for both user/item graphs (right number) and user graph only (left number). Baseline numbers are taken from [22].

Model	ML-1M	ML-10M
PMF [20]	0.883	–
I-RBM [26]	0.854	0.825
BiasMF [16]	0.845	0.803
NNMF [7]	0.843	–
LLORMA-Local [17]	0.833	0.782
I-AUTOREC [27]	0.831	0.782
CF-NADE [32]	**0.829**	**0.771**
GC-MC (Ours)	0.832	0.777

Table 4: Comparison of average test RMSE scores on five 90/10 training/test set splits (as in [32]) without the use of side information. Baseline scores are taken from [32]. For CF-NADE, we report the best-performing model variant.

EXPERIMENTS

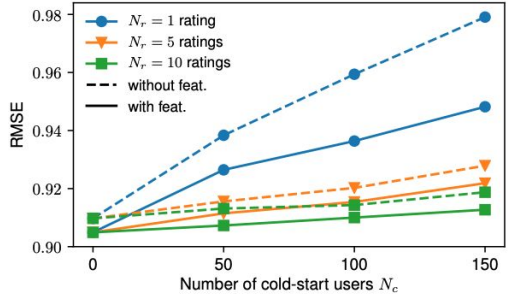


Figure 3: Cold-start analysis for ML-100K. Test set RMSE (average over 5 runs with random initialization) for various settings, where only a small number of ratings N_r is kept for a certain number of cold-start users N_c during training. Standard error is below 0.001 and therefore not shown. Dashed and solid lines denote experiments without and with side information, respectively.

Nc : fixed number of cold-start users

Nr : all ratings except for a minimum number are removed from the training set

Conclusion

추천 시스템의 Matrix Completion 에 graph auto-encoder 를 적용함

bipartite user-item interaction 그래프를 구성해 message passing 형태로 변형

graph convolution layer를 가지고 encoder를 구성해 embedding을 생성

bilinear 구조를 가지는 decoder를 구성해 그래프의 edge 형태의 rating을 예측

유저와 아이템 간 관계와 함께 유저 아이템의 피처를 함께 사용 가능

Reference

[기초개념]Graph Convolutional Network (GCN)

jihoon 블로그 Graph Neural Network

gc-mc pytorch implements repository (github)

https://github.com/riannevdberg/gc-mc