Recommendation System Study Report

Yutong Wang, Guangyao Liu
TEAM, ECE
University of Rochester
Rochester, NY 14627

May 7, 2020

Abstract

The purpose of this project is to study different techniques to solve recommendation system problems. From the traditional approaches to the most recent and trendy strategies, we mainly studied and discussed four methods in this report: Collaborative Filtering(CF), Matrix completion using Convolutional Neural Network, Inductive Graph-based Matrix Completion and Matrix completion using Separable Recurrent Graph Convolutional Neural Network. The mathematical theory and advantages of each technique will be discussed in this report. In order to compare the performances of each method, we will test them on several datasets such as Douban, Yahoo music and Netflix, but mainly on the MovieLens datasets. Then, we will analysis the root mean square estimation(RMSE) and summarize the results.

1 Introduction

Nowadays, recommendation systems are widely used in our life. Everytime we surf on the internet, we would be potentially affected by recommendation systems. Netflix recommends movies to us, Facebook introduces us to friends we might know, LinkedIn notifies job posts we are interested in and there are more examples involving recommender systems. These systems help us make decisions and improve our search experience, and meanwhile help companies make more profit as well. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users.

The main objective of this project is to study and implement recommender systems. There will be four different methods of the recommender system to be tested and evaluated: Collaborative Filtering(CF), Matrix completion using Convolutional Neural Network, Inductive Graph-based Matrix Completion and Matrix completion using Separable Recurrent Graph Convolutional Neural Network.

This project report will be organized into four main sections: 1. Data, 2. Techniques, 3. Results and 4. Summary. In the Data Section, since we will mainly be testing the algorithms on MovieLens dataset, so we describe this dataset in detail in this section. For the Technique Section, there will be four parts which each discuss one recommender system method and introduce the mathematical theory behind the algorithm. In the first part of this project, we used model-based Collaborative Filtering to compute the prediction matrix. In the second part, we adopted CNN to get two feature vectors, user vector and movie vector. With these vectors we can compute the prediction matrix as well as give recommendations based on a given use or movie. For part three and part four, we searched another two recent papers that propose new approaches to the recommender system problems: optimized matrix completion with graph neural networks. The method in part three is called inductive graph-based matrix completion, and this method was

proposed by Zhang and Chen in 2020[1]. Then the fourth part is Geometric Matrix Completion with Recurrent Multi-graph Neural Networks which was first introduced by Monti, Bronstein and Bresson in 2017[2]. Then for the Results Section and Summary Section, we will analyze the performance of each algorithm and summarize the fundings.

2 Data

In this project, we used several datasets that are widely used in recommendation systems. They are MovieLens 100K, douban, Yahoo Music and Netflix. Here we take MovieLens 100K as an example to show its format, it contains 3 files, user.dat, movies.dat and rating.dat, which are shown in Figure 1 and Figure 2.

In the user dataset, there are 5 attributes, which are UserID, Gender, Age, OccupationID and ZIP-code. The former 4 attributes of them are useful for us. In the movies dataset, there are 3 attributes that are useful. In the rating dataset, there are 4 attributes. The timestamps are not relevant to the prediction of rating, so we ignore them.

The basic information of the datasets are shown in Figure 3. (The Netflix dataset we used is not the standard one, it has been processed in order to train the Network and we cannot extract the basic information. So it is not listed here.)

	UserID	Gender	Age	OccupationID	Zip-code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

	MovieID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

Figure 1: movie and user datasets

	UserID	MovieID	Rating	timestamps
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

Figure 2: rating datasets

Dataset	Users	Items	Ratings	Density	Rating types
Douban	3,000	3,000	136,891	0.0152	1, 2, 3, 4, 5
YahooMusic	3,000	3,000	5,335	0.0006	1, 2, 3, ..., 100
ML-100K	943	1,682	100,000	0.0630	1, 2, 3, 4, 5

Figure 3: Basic information of datasets

For convenience, we have to do pre-processing to the data. In order to simplify the computation, we transfer all data into digits. For example, for the Gender attribute in user dataset, we transfer 'F' and 'M' to '0' and '1'.

3 Techniques

3.1 Collaborative Filtering

Collaborative Filtering is the oldest and most widespread technique for recommendation systems. There are two types of Collaborative Filtering models, memory-based and model-based. The memory-based models are relatively easy to understand and simple to implement. Within memory-based models, there are two different approaches: item-based and user-based. The differences of these two approaches are self explanatory by their names. Item-based collaborative filtering method finds similar items using side information and recommends to the users who give the related item a high rate, and user-based collaborative filtering method recommends to a user by analysing other similar users' preference. The matching process can be done using nearest neighbor algorithm or item to item collaborative filtering.

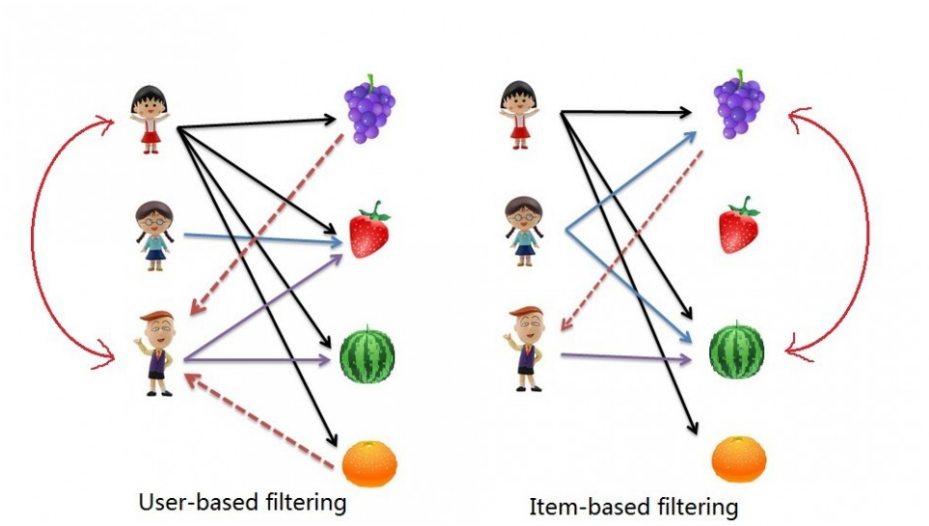


Figure 4: User-based VS Item-based
[3]

3.1.1 Problems with Collaborative Filtering

Collaborative filtering is a simple but powerful method, but it still has some significant weaknesses. First of all, collaborative filtering is not effective when dealing with sparsity. For example, if very few movies are rated by more than one user, it will be hard to create a useful similarity matrix. Then the algorithm can't make accurate predictions. Note that data can be spread even though there is a large amount of data. In contrast, model-based approaches using matrix factorization handle the sparsity much better. Second problem is if a new user is introduced to the system without rating any movies and no side information, the system will not be able to find other users who match this new user, and it is the same situation for a new movie without rating. Thirdly, for some users who have no 'neighbours' with similar preference, they will not receive accurate recommendations.

3.1.2 Model-Based: Collaborative Filtering with Spark

For this project, the alternating least square algorithm in spark.ml is used to learn latent factors in order to fill in the missing entries.

Algorithm 1 ALS for Matrix Completion

Initialize X, Y

repeat

for $u = 1 \dots n$ **do**

$$x_u = \Big(\sum_{r_{ui} \in r_{u*}} y_i y_i^\intercal + \lambda I_k \Big)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i \tag{2}$$

end for

for $i = 1 \dots m$ **do**

$$y_i = \Big(\sum_{r_{ui} \in r_{*i}} x_u x_u^\intercal + \lambda I_k \Big)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u \tag{3}$$

end for

until convergence

Figure 5: Alternating Least Square algorithm
[4]

We adjust the following parameters in order to implement ALS algorithm in spark.ml

- rank is the number of latent factors in the model (2, 4, 6, 8, 10, 12).
- maxIter is the maximum number of iterations to run (10).
- regParam specifies the regularization parameter in ALS (0.1).
- alpha is a parameter applicable to the implicit feedback variant of ALS that governs the baseline confidence in preference observations (0.02).

Package: pyspark.mllib.recommendation(ALS) - Python

3.2 Convolutional Neural Network

In this Method, the high level idea is to get user feature and movie feature vectors by training the CNN. With these two vectors we can get the prediction of the rating data, which means we can predict how much score will any user give to any movie. When we have the prediction, we can give recommendation to any user.

To get the user feature vector, we take some pre-processed data as input to train the CNN.The work flow is shown below.

For movie dataset, we can process them and get the movie feature. In recommendation, the Genres and titles actually matter. For example, if some one rated an action movie a very high score, he is likely to like another popular action movies. Also, we can obtain some information from its title, such as we see a title like 'Casualties of War ', we can gess this is a war movie, etc. So there is a little different when we get movie feature vector. We do the same process to movie ID and Genres as we did to user data. We use an another CNN that is intended to process natural language to process the movie titles.[5]

After having these two vectors, we can obtain the prediction matrix. Then we will do a regression by using the prediction matrix and the real rating data. We are going to use Mean Square Error (MSE) loss function. Besides, instead of using this method, we can also take the two vectors as input, use a fully connected layer to do a regression with the real rating data.

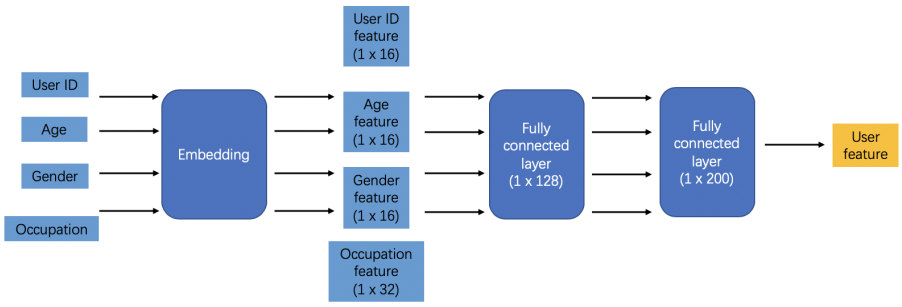


Figure 6: The process of getting user feature

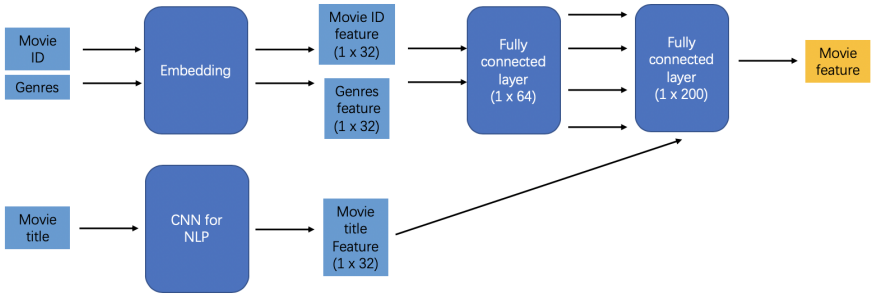


Figure 7: The process of getting movie feature

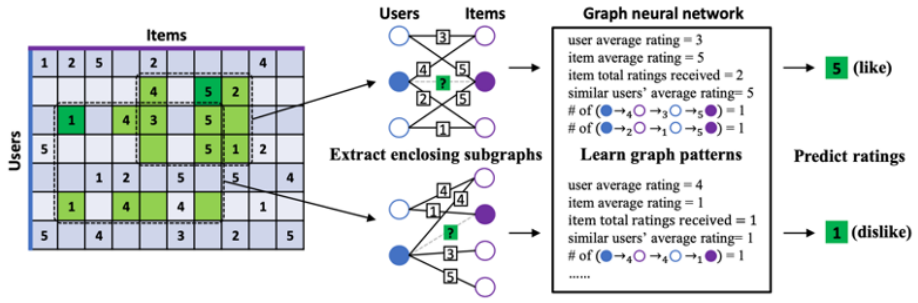
3.3 Inductive Matrix Completion Based On Graph Neural Networks

This section will present the Inductive Graph-based Matrix Completion (IGMC) framework without using side information.

There are some matrix completion algorithms use factorization techniques achieved great successes (Adomavicius Tuzhilin, 2005[6]; Schafer et al., 2007[7]; Koren et al., 2009[8]; Bobadilla et al.,2013[9]). However, matrix factorization is intrinsically transductive, which means when the rating matrix has changed values or has new rows/columns added, it often requires a complete retraining to get the new embeddings. That's why we proposed inductive matrix completion.

Also, most previous works about inductive matrix completion use side information such as user's age, movies genre to make predictions. However, high-quality content is not always available. What if the available information is just the rating matrix, can we get a good result? Then we introduced Inductive Graph-based Matrix Completion (IGMC) model to address this problem. IGMC trains a graph neural network (GNN) based purely on 1-hop subgraphs around (user, item) pairs generated from the rating matrix and maps these subgraphs to their corresponding ratings.

We used G to represent the undirected bipartite graph constructed from the given rating matrix $\mathbf{R}$. In G, a node is either a user or an item. Edges can exist between a user and an item but cannot exist between two users or two items. Each edge *(u,v)* has a value $r = R_{u,v}$, corresponding to the rating that u gives to v. We use R to denote the set of all possible ratings (e.g., R = 1, 2, 3, 4, 5 in MovieLens), and use N_r (u) to denote the set of u's neighbors that connect to u with edge type r.



[1]

Figure 8: The basic idea of Inductive Graph-based Matrix Completion

The work flow is shown in Figure 7. The input of this system is the rating data which is a matrix with row represents users and column represents items(movies). Every entry is the rating that given by a user to a movie (In this dataset, its 1,2,3,4,5). The basic idea of this method is shown in figure x. we extract a local enclosing subgraph around each rating (dark green block), and train a GNN to map subgraph to ratings.

3.3.1 Enclosing Subgraph Extraction

The first part of IGMC is enclosing subgraph extraction. For each observed rating $R_{u,v}$, we extract an h-hop enclosing subgraph around (u,v) from G. Algorithm 1 describe the BFS procedure for extracting h-hop enclosing subgraphs.

Algorithm 1 ENCLOSING SUBGRAPH EXTRACTION

input: h, target user-item pair (u, v), the bipartite graph G
output: the h-hop enclosing subgraph $G^h_{u,v}$ for (u, v)
$U = U_{fringe} = \{u\}, V = V_{fringe} = \{v\}$
for $i = 1, 2, \ldots, h$ **do**
  $U'_{fringe} = \{u_i : u_i \sim V_{fringe}\} \setminus U$
  $V'_{fringe} = \{v_i : v_i \sim U_{fringe}\} \setminus V$
  $U_{fringe} = U'_{fringe}$, $V_{fringe} = V'_{fringe}$
  $U = U \cup U_{fringe}$, $V = V \cup V_{fringe}$
  Let $G^h_{u,v}$ be the vertex-induced subgraph from G using vertices U, V
  Remove edge (u, v) from $G^h_{u,v}$.
end for
return $G^h_{u,v}$

Note: $\{u_i : u_i \sim V_{fringe}\}$ is the set of nodes that are adjacent to at least one node in V_{fringe} with any edge type.

[1]

Figure 9: Algorithm 1

3.3.2 Node Labeling

Before we feed an enclosing subgraph to the GNN, we first apply a node labeling to it. we propose a node labeling as follows: We first give label 0 and 1 to the target user and target item, respectively. Then, we determine other nodes' labels according to at which hop they are included in the subgraph in Algorithm 1. If a user-type node is included at the ith hop, we will give it a label $2i$. If an item-type node is included at the i-th hop, we will give it $2i + 1$.

Ideally, our node labeling should be able to :1) distinguish the target user and target item between which the target rating is located, and 2) differentiate user-type nodes from item-type nodes.

3.3.3 Graph Neural Network Architecture

The third part of IGMC is to train a graph neural network model predicting ratings from the enclosing subgraphs. This network applies a graph-level GNN to the enclosing subgraph around *(u,v)* and maps the subgraph to the rating. Thus, there are two components in this network: 1) message passing layers that extract a feature vector for each node in the subgraph, and 2) a pooling layer to summarize a subgraph representation from node features. We adopted the relational graph convolutional operator(R-GCN)[10] as our GNN's message passing layers, which has the following form:

$$\mathbf{x}_i^{l+1} = \mathbf{W}_0^l \mathbf{x}_i^l + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} \frac{1}{|\mathcal{N}_r(i)|} \mathbf{W}_r^l \mathbf{x}_j^l, \tag{1}$$

where x_l^i denotes node i's feature vector at layer l, W_l^0 and $\{W_l^r | r \epsilon R\}$ are learnable parameter matrices.

3.3.4 Model Training

Loss function

In the training process, we minimize the mean squared error (MSE) between the predictions and the ground truth rating:

$$\mathcal{L} = \frac{1}{|\{(u,v)|\mathbf{\Omega}_{u,v} = 1\}|} \sum_{(u,v):\mathbf{\Omega}_{u,v}=1} (R_{u,v} - \hat{R}_{u,v})^2, \tag{2}$$

Adjacent rating regularization

The R-GCN layer used in our GNN has different parameters Wr for different rating types. One drawback here is that it fals to take the magnitude of ratings into consideration. For instance, a rating of 4 and a rating of 5 in MovieLens both indicate like, while a rating of 1 indicates don't like. Ideally, we expect the model knows that the rating of 4 is closer to 1 than the 5. However, in R-GCN, they are treated as three independent edge types – the magnitude and order information of the ratings is completely lost. To address this problem, we proposed an adjacent rating regularization (ARR) technique, which encourages ratings adjacent to each other to have similar parameter matrices. Assume the rating R exhibit an ordering r1, r2, ... r R which indicates increasingly higher preference that users have for items. Then, the ARR regularizer is:

$$\mathcal{L}_{\text{ARR}} = \sum_{i=1,2,\ldots,|\mathcal{R}|-1} \|\mathbf{W}_{r_{i+1}} - \mathbf{W}_{r_i}\|_F^2, \tag{3}$$

where $\|\cdot\| F$ denotes the Frobenius norm of a matrix.

3.4 Geometric Matrix Completion with Recurrent Multi-graph Neural Networks

Geometric matrix completion with recurrent multi-graph neural networks was first introduced by Federico Monti, Michael Bronstein and Xavier Bresson in 2017[2]. This hybrid technique boosted the performance of matrix completion with deep learning approaches. In order to study this method, we break it down into three parts: traditional matrix completion, graph convolutional neural network and recurrent neural network.

3.4.1 Matrix Completion

The Matrix Completion method is a traditional and popular approach to recommendation system problems. Whenever a recommendation problem needs to be solved, engineers and scientists will consider and try matrix completion. It is because it fits the problem so well. The columns of the matrix represent the users and the rows represent the items, and all the numbers in the matrix are the ratings of the items from the users. The ideal situation of matrix completion is that the rank of the matrix r is as low as possible:

$$\min_{\mathbf{X}} \ \mathbf{rank}(\mathbf{X}) \quad \text{s.t.} \quad x_{ij} = y_{ij}, \ \forall ij \in \Omega \tag{4}$$

In order to fix the noisy scenario, s convex relaxation of the algorithm that introduced by Candles and Plan[11] was:

$$\min_{\mathbf{X}} \ \|\mathbf{X}\|_\star + \frac{\mu}{2}\|\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{Y})\|_{\mathrm{F}}^2 \tag{5}$$

However, this algorithm is computational complex for real datasets.

3.4.2 Graph Convolutional Neural Network

Earlier, we studied and experimented with the convolutional neural network method. CNN had major contributions to computer-vision related problems, and it was not used on the recommendation system problems until recent years because it can only process Euclidean-structure data which recommendation system problems had graph-structure data. In 2016, the development of geometric deep learning[12] broke the wall between them and gave recommendation system problems more possible solutions. For this method, we will be using a theorem of Laplacian matrix, a graph representation matrix: $L = D - A$. For which D is the degree matrix and A is the adjacency matrix. Using spectral decomposition, L is given by: $L = V * \Lambda * V^T$. where $\Lambda = diag(\lambda_1, \lambda_2, ...)$ is a diagonal matrix collecting the eigenvalues of L, and V is a matrix with the corresponding eigenvectors. The role of eigenvalues here is the same as frequencies and eigenvectors as Fourier atoms compare with a classical harmonic analysis. In this case for $x = (x_1, ..., x_n)^T$ on the vertices of the graph, its graph Fourier transform is given by $\hat{X} = V^T X$.

3.4.3 Multi-Graphs Convolution

This method use matrix factorization to divide data into two matrixs:$X = WH^T$, for which W is a $m \times r$ item matrix and H is a $n \times r$ user matrix. The purpose is to reduce the degree of freedom from $m \times n$ to $m + n$ in order to make the algorithm computational easy.

In the multi-graphs setting, we are using two-dimensional Fourier transform: $\hat{X} = V_r^T X V_c$ for which V_r and V_c are the $m \times m$ and $n \times n$ eigenvector matrices of Laplacian matrices L_r and L_c of W and H^T. Then, apply one-dimensional convolution to each graph with respective factors:

$$\tilde{w_l} = \sum_{l'=1}^{q'} V_r * \hat{Y_r}^{ll'} * V_r^T * w_{l'}, l = 1, ..., q \tag{6}$$

$$\tilde{h_l} = \sum_{l'=1}^{q'} V_c * \hat{Y_c}^{ll'} * V_c^T * h_{l'}, l = 1, ..., q \tag{7}$$

For which $\hat{Y}_r^{ll'} = diag(\hat{y}^r_{ll',1}, ..., \hat{y}^r_{ll',m})$ and $\hat{Y}_c^{ll'} = diag(\hat{y}^r_{ll',1}, ..., \hat{y}^r_{ll',n})$. Then algorithm process is explained in the following figures:

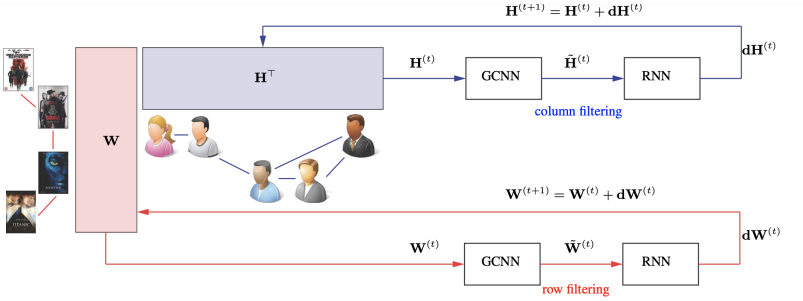


Figure 10: Separable Recurrent GCNN Algorithm [2]

Algorithm 2 Factorized matrix completion model using sRGCNN

input $m \times r$ factor $\mathbf{H}^{(0)}$ and $n \times r$ factor $\mathbf{W}^{(0)}$ representing the matrix $\mathbf{X}^{(0)}$

for $t = 0 : T$ **do**
  Apply the Graph CNN on $\mathbf{H}^{(t)}$ producing an $n \times q$ output $\tilde{\mathbf{H}}^{(t)}$.
  for $j = 1 : n$ **do**
    Apply RNN to feature vector $\tilde{\mathbf{h}}_j^{(t)} = (\tilde{h}_{j1}^{(t)}, \ldots, \tilde{h}_{jq}^{(t)})$ producing the predicted incremental value $dh_j^{(t)}$
  end for
  Update $\mathbf{H}^{(t+1)} = \mathbf{H}^{(t)} + \mathbf{dH}^{(t)}$
  Apply the Graph CNN on $\mathbf{W}^{(t)}$ producing an $m \times q$ output $\tilde{\mathbf{W}}^{(t)}$.
  for $i = 1 : m$ **do**
    Apply RNN to feature vector $\tilde{\mathbf{w}}_i^{(t)} = (\tilde{w}_{i1}^{(t)}, \ldots, \tilde{w}_{iq}^{(t)})$ producing the predicted incremental value $dw_i^{(t)}$
  end for
  Update $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \mathbf{dW}^{(t)}$
end for

Figure 11: Separable Recurrent GCNN Pseudocode [2]

4 Results And Analyses

Root Mean Square Estimation (RMSE)

RMSE is used in this project to estimate the average error between the predicted value and the actual value.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{8}$$

The Results of All Models

We have tested those models with the datasets.(The first two method just tested by MovieLens 100k). The results are shown in Figure 14. The RMSE we got are all a little higher than the results in papers, we think that the processing of training and predicting is a random process, we cannot get the exact same result.

	MovieLens	Douban	YahooMusic	Netflix	Flixster
CF	0.92234	/NA	/NA	1.66387	/NA
CNN	0.8023652	/NA	/NA	/NA	/NA
Inductive Graph-based MC	0.916	0.746	20.3	/NA	0.896
sRGCNN	1.131428	0.795920	24.4149	0.682848	1.1788

Figure 12: RMSE results of all models

Collaborative Filtering

Pro:

CF as a traditional approach of recommendation system problems is very easy to understand and simple to implement, and it is a powerful method that can give decent performances on some datasets.

Con:

This method cannot handle sparsity appropriately. Also, if a brand new user or item joins in, this algorithm can't make accurate predictions.

CNN

Pro:

1.This method is comparatively easy to build, the complexity of this model is not very high.

2.It uses many side information, sometimes high quality side information is not available(such as some video stream websites, we don't have so many user information)

Inductive Graph-based MC

Pro:

1.This model doesn't need side information, it could be implemented on some online video or music application.

2.It is inductive, so we can use it to make predictions by just training it for once. The most important is that it is based on graph pattern, which makes it more general. It can be used in many other datasets after training by one dataset, which is a very useful property.

3.When adding new rating information into the dataset, it doesn't influence the model,while some transductive models have to train again.

sRGCNN

Pro:

This method uses the approach of recurrent muli-graph CNN while fully exploiting the local stationarity structures of user and item graphs. Solving the problem of the number of parameters to learn is linear. Meanwhile, this method reduce the complexity of the RGCNN from (m*n) to (m+n) for which m represents m items and n represents n users.
Con:

The performances of this method compared to other graph-based algorithm are not competitive.

5 Summary

From the results, Inductive Graph-based Matrix Completion method has the best performance among almost all datasets. IGMC performs better than sRGCNN in all of those datasets, and also, the biggest advantage of this model is that it is inductive, it learns the local graph patterns related to ratings inductively. Besides, it does not need content information of users/items, which is a very desirable property of real scenario.

From the RMSE result table, the data of YahooMusic are outstanding and relatively larger than the other dataset. This is because the ratings in YahooMusic dataset are from 1 to 100, and for other datasets, the ratings range from 1 to 5.

Data science models have evolved considerably over the last 20 years. After studying algorithms of different methods of building recommendation systems, we are amazed by how fast the machine learning and deep learning research findings rapidly increased and developed. Recommendation systems have become an integral part of everyone's life, and it will continue to serve people search online who need some hints to make their decisions.

6 Acknowledgement

We would like to express our special thanks to our teacher Gonzalo Mateos for offering us a wonderful opportunity to do this project on the topic Recommender System and also helped a lot of this project.

References

[1] Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058*, 2019.

[2] Michael M. Bronstein Federico Monti and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *arXiv preprint arXiv:1704.06803*, 2017.

[3] Carlos Pinela. Recommender systems — user-based and item-based collaborative filtering.

[4] Databricks Reza Zadeh and Stanford. Distributed algorithms and optimization. page 2, 2015.

[5] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *arXiv e-prints*, page arXiv:1408.5882, August 2014.

[6] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[7] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[9] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[10] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[11] Emmanuel J. Candes and Benjamin Recht. Exact matrix completion via convex optimization. *arXiv preprint arXiv:0805.4471*, 2008.

[12] Yann LeCun Arthur Szlam Michael M. Bronstein, Joan Bruna and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *arXiv preprint arXiv:1611.08097*, 2017.

[13] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion, 2017.

[14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, Jan 2009.

[15] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2018.

[16] Federico Monti, Michael M. Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks, 2017.

[17] N.S. Altman. An introduction to kernel and nearest neighbor nonparametric regression, 1991.

[18] Jennifer A. Jacobi Gregory D.Linden and Eric A. Benson. Collaborative recommendations using item-to-item similarity mappings, 1999.