BOOK RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING

Ishaan Sood, Mayank Gupta

Mtech,CDS
IISc, Bangalore

ABSTRACT

Reccommender systems provide users with personalized suggestions for products or services, mostly using collaborating filtering (CF), where past transactions are analyzed in order to establish connections between users and products[1]. The most common approach to CF is based on neighborhood models, which originate from similarities between products or users. Some more accurate results are obtained from low-rank factorization methods, which has been in vogue since the Netflix Prize. Recently, Neural Networks were employed for the same and produced some remarkable results. In this work, authors intend to train the neural network using Goodbooks-10k dataset containing 10,000 reviewed books, extracted from www.goodreads.com and compare the same against classical techniques that have been widely used so far. We see that Neural methods are far superior on topK-evaluation metric [2], though there are certain drawbacks that need to be addressed in order to remove biases from these methods towards more popular items. The final code can be found at
https://github.com/ishaan16/ds294/tree/master/project

Index Terms— Machine Learning, Collaborative Filtering, Deep Learning, Matrix Factorization, Recommendation systems

1. INTRODUCTION

Users of the electronic retailers are flooded with the amount of choices available to them. Any e-retailer or social media websites would like to improve its user satisfaction, which depends significantly on quality of recommendations made. Since user choices and ratings can display very complex patterns the area of Recommender System has gained a lot of interest and there have been a lot of work in the previous decade. Recommender systems are generally based on Collaborative Filtering(CF) which takes user's reviews, ratings of items, past interactions in the form of purchase history, browsing history, search patterns or mouse clicks and even movements, etc. as input and gives a list of items which the user is most probable to like or interact with. CF have become the most favoured approach due to their ability to analyze and uncover the complex user behaviour without being domain specific and their capability of avoiding the need to make explicit profiles of users.

The methods used in CF have changed substantially over the years and in this project the authors demonstrate the performance of the most widely used CF techniques namely Nearest Neighbour, Matrix Factorization [], and Neural Network based CF [3] on the metric of topK-evaluation [2] on goodbooks-10k Dataset [4] and conclude that although the accuracy have increased over the years there is still room for a lot of improvement.

2. LITERATURE REVIEW

Collaborative filtering was named to signify that people collaborate to help one another perform filtering by recording their reactions to documents they read [1]. This reaction can be both explicit in form of ratings or implicit in form of interaction (such as read the document or not). Neighborhood-based filtering was the most common approach that was used. Over the years, it went through several changes such as changing from user-user collaboration to item-item collaboration [5], using factorized neighborhood models [2] etc.

Since the Netflix Prize, where a competition was held to improve the Netflix recommendations, latent factor models have gained immense popularity, thanks to their improved accuracy [6], [7], [8]. In [9], neighborhood-based and latent-factor models were merged to get improved accuracy while retaining the intuitiveness of the neighborhood-based models. This new model,called SVD++, offered improved accuracy at the cost of greater training time.

Evaluation metrics for recommendations have also evolved over time. While minimizing the Root Mean Square error was popular even in Netflix Prize, [2] introduced the topK-evaluation that has been used in the current work and has a desirable property of broader spectrum for comparison of algorithms.

Deep learning approaches to collaborative filtering have mainly focused on DNNs as tools to model auxiliary information required by recommendation systems [10],[11]. However recent work by [3] shows immense promise of using neural networks to learn interaction from the data. This is the main subject of current work where authors have used these models for Goodbooks-10k dataset [4] and compared them against standard SVD [12] and baseline-adjusted KNN approach (explained in sec.2.2 of [2] on basis of topK-evaluation.

3. CLASSICAL TECHNIQUES

An item-item neighborhood approach evaluates the preference of a user for an item based on ratings of similar items by the same user. In a sense, these methods transform users to the item space by viewing them as baskets of rated items. Hence they are intuitive and can be easily explained. Most of the internet leaders employ these models for recommendations. However, these methods having little to none training time are not scalable and less accurate.

Latent factor models, such as singular value decomposition (SVD), comprise an alternative approach by transforming both items and users to the same latent factor space, thus making them directly comparable. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback. For example, when the products are books, factors might measure obvious dimensions such as fiction vs non-fiction, poetic vs prosaic, or orientation to children as well as less well defined dimensions such as character, plot, writing-style or even completely uninterpretable dimensions. For users, the latent factors measure their affinity for a set of books, which is a more natural factor for recommendation compared to peers' tastes.

3.1. Baseline adjusted K-Nearest Neighbors

Baseline scores for each pair is defined as:

$$b_{ui} = \mu + b_u + b_i \tag{1}$$

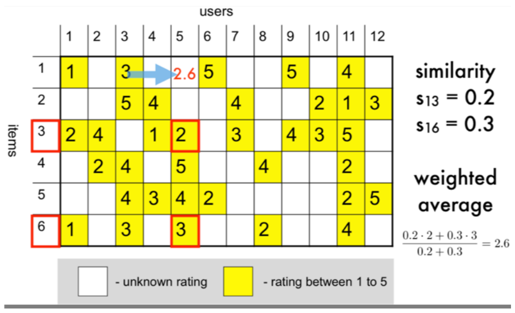
The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. For example, suppose that we want a baseline estimate for the rating of the movie Titanic by user Joe. Now, say that the average rating over all movies, μ, is 3.7 stars. Furthermore, Titanic is better than an average movie, so it tends to be rated 0.5 stars above the average. On the other hand, Joe is a critical user, who tends to rate 0.3 stars lower than the average. Thus, the baseline estimate for Titanics rating by Joe would be 3.9 stars by calculating 3.70.3+0.5. In order to estimate b_u and b_i one can solve the least squares problem:

$$\min_{b} \sum_{(u,i)\in\mathcal{Y}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda \left(\sum_u b_u^2 + \sum_i b_i^2 \right) \tag{2}$$

This is done using stochastic gradient descent method. Once the baseline scores are estimated and similarity measured using Pearso's coefficient, the final ratings are given by:

$$r_{ui} = b_{ui} + \frac{\sum_j \in \mathcal{N}_i^k \text{sim}(i,j).(r_{uj} - b_{uj})}{\sum_j \in \mathcal{N}_i^k \text{sim}(i,j)} \tag{3}$$

This is illustrated in fig.(1)



	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Fig. 1. Working of KNN algorithm illustration

3.2. SVD Low rank factorization

In latent factors model,the sparse matrix is approximated as product of 2 dense matrices in latent variable space calculated by minimizing the following loss function:

$$L = \sum_{r_{ui}\in\mathcal{Y}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i^2|| + ||p_u^2|| \right) \tag{4}$$

This minimization is performed by singular value decomposition of ratings matrix into low rank U and V matrices.

4. NEURAL COLLABORATIVE FILTERING

Let M and N denote the number of users and items, respectively. We define the useritem interaction matrix $Y \in \mathbb{R}^{M\times N}$ such that:

$$y_{ui} = \begin{cases} 1 \text{ if user u has interacted with item i} \\ 0 \text{ else} \end{cases} \tag{5}$$

Such system of ratings is designed for implicit data and is therefore noisy. The recommendation problem with implicit feedback is formulated as the problem of estimating the scores of unobserved entries in Y, which are used for ranking the items. The problem is to learn $\hat{y}_{ui} = f(u, i|\Theta)$ where $\hat{y}$ denotes the predicted score of interaction y_{ui},θ denotes model parameters, and f denotes the interaction function that maps model parameters to the predicted score. MF associates each user and item with a real-valued vector of latent features. Let p_u and q_i denote the latent vector for user u and item i, respectively; MF estimates an interaction y_{ui} as the inner product of p_u and q_i:

$$\hat{y}_{ui} = p_u^T q_i \tag{6}$$

Under the NCF framework this is modeled as:

$$\hat{y}_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f) \tag{7}$$

Here, $P \in \mathbb{R}^{M\times K}$ and $Q \in \mathbb{R}^{N\times K}$, denote the latent factor matrix for users and items, respectively; and θ_f denotes the model parameters of the interaction function f. For implicit data, the target value y_{ui} is a binarized 1 or 0. [3] uses a probabilistic approach for learning the pointwise NCF that pays special attention to the binary property of implicit data. Considering the one-class nature of implicit feedback, we can view the value of y_{ui} as a label 1 which means item i is relevant to u, and 0 otherwise. The likelihood function then becomes:

$$p(\mathcal{Y}, \mathcal{Y}^-) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{ui} \prod_{(u,j)\in\mathcal{Y}^-} \hat{y}_{uj} \qquad (8)$$

The negative log likelihood that we intend to minimize then becomes a cross-entropy loss function:

$$L = -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}) \qquad (9)$$

4.1. Generalized Matrix Factorization

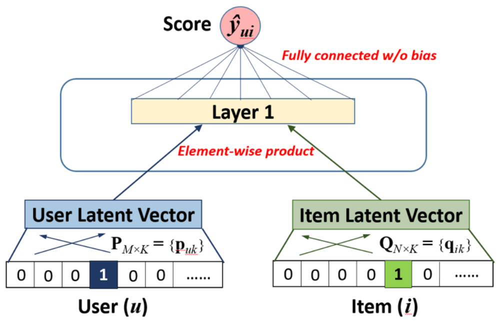


Fig. 2. Generalized Matrix Factorization

In the first layer of GMF network shown in fig.(2), element-wise product of the latent vectors of user and item is taken. This is followed by a fully connected layer connected to the output where a sigmoid activation function is used to convey the final value of $\hat{y}_{ui} \in [0, 1]$:

$$\hat{y}_{ui} = a_{\text{out}}(h^T(p_u \odot q_i)) \qquad (10)$$

By learning the weights h and embedding matrices P and Q, the factorization is in the most generalized form and can replicate SVD,SVD++ etc.

4.2. Multi-Layer Perceptron

This approach follows from the concatenating the user-item latent vectors instead of element wise multiplying as in the first layer of GMF. To get any meaningful result of this concatenated vector, there should be more number of hidden layer in the network. Thus the network shown in Figure 3 emerges as the obvious candidate. This network is named Multi-Layer Perceptron (MLP).

$$\phi_1(\boldsymbol{p}_u, \boldsymbol{q}_i) = \begin{bmatrix}\boldsymbol{p}_u \\ \boldsymbol{q}_i\end{bmatrix} \qquad (11)$$

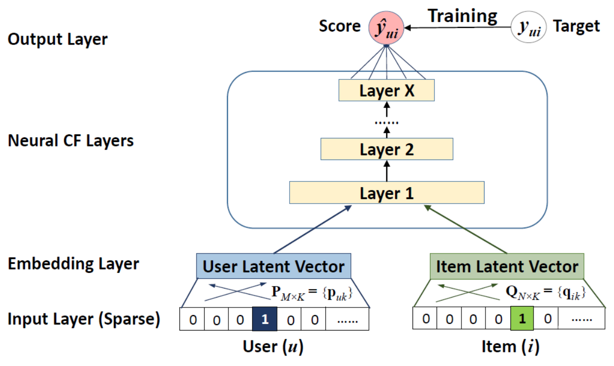


Fig. 3. Multi-Layer Perceptron

The ReLU is used as activation functions in each of the node (empirically the ReLU outperforms sigmoid and tanh). For the output sigmoid is used as the result needs to be in $(0, 1)$. The loss function used is the cross-entropy loss same as that in GMF.

4.3. Neural-MF

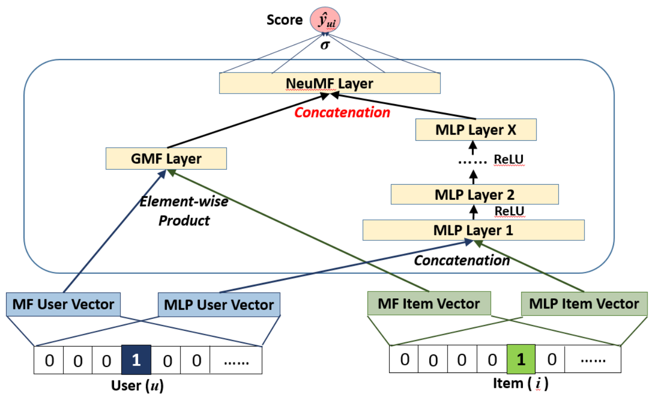


Fig. 4. NeuMF- Fusion of MLP and GMF

One of the straightforward ways to fuse GMF and MLP is as shown as Figure 4. Here the two networks do not share the embeddings so that it can be learnt according to the respective complexity. The last layer is a concatenated version of both the networks. The objective function of this network is non-convex and gradient based optimization function may stop at a locally optimum solution. To avoid this problem the two networks are trained separately first and the learnt parameters are used to to initialize neuMF network.

5. EXPERIMENTS

5.1. Dataset

The dataset used is goodboks-10k [4] which contains 10,000 books and there are approx. 6 Million reviews in ratings in total from 55000 users approximately. The statistics for the reviews per book distribution and reviews per user distribution is given in Table 1.

Statistic	Reviews per book	Reviews per user
Mean	598	112
Std Dev.	1267	26
25% percentile	155	96
50% percentile	248	111
75% percentile	503	128
Min & Max	8 & 22806	19 & 200

Table 1. Statistics of Goodbooks 10k dataset

So as can be seen from statistcs the data is highly sparse with all users having read at max 200 books out of 10000.

5.2. Top-K Evaluation Metric

The authors use the top-K Evaluation metric as was used in [2] to predict K best suitable items for a user. To get a validation-set needed for the metric the data is divided in 80:20 ratio in training and validation-set before the start of algorithm. A user, book pair is selected from the validation-set which has a 5 star rating and K-1 other books out of the 10,000 books are selected such that the first book is not present in them. These K books are then fed to the trained model to predict the probabilities of interaction between the user and the books. The rank of originally 5-star rated book is calculated among these K predictions. This is done with every 5-star rating present in the validation-set. The predicted rank should be higher up as the other books are randomly selected and thus have a low probability of being liked by user (due to sparse data).

This metric gives a very useful practically metric as opposed to RMSE because the range of values which RMSE can take is very limited, approximately between 1.1 and 0.8 while top-K Metric can show sufficient amount of changes with change of algorithm.

5.3. Results

Using the ranks predicted above, the fraction of times a calculated rank is within a rank percentile is plotted in Figure 6, with K =1000, for different methods. Only first 2 percentiles are shown as only a limited number of items can be suggested to a user. As can be seen that SVD performs worse compared to Neural CF based methods and among them the best performance is shown by NeuCF. K-NN is not very scalable and we were able to compute this only for K=100 as K=1000 couldn't complete even after running for a day.

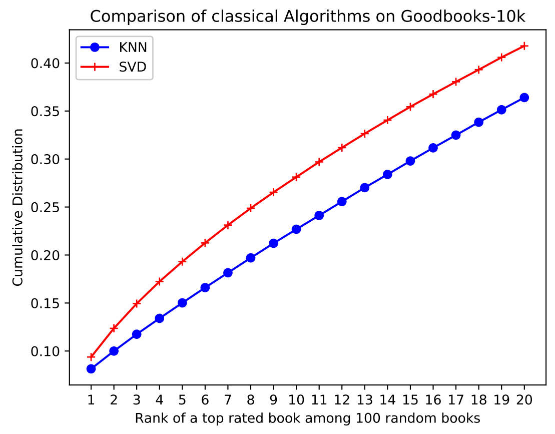


Fig. 5. Top-100 Recommends plot for SVD and KNN observing the distribution of rank of a top rated book among the 100 random books for a given user

6. DISCUSSION

The code is implemented using Surprise [13] package for classical CFs and Keras with Theano backend for Neural CF. The code can be found at xxx

7. REFERENCES

[1] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[2] Yehuda Koren and Yahoo ! Research, "Factor in the Neighbors: Scalable and Accurate Collaborative Filtering," *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 1, 2010.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua, "Neural Collaborative Filtering *," pp. 17–4, 2017.

[4] "Goodbooks-10K," `https://www.kaggle.com/zygmunt/goodbooks-10k`, 2017.

[5] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the tenth international conference on World Wide Web - WWW '01*, 2001, pp. 285–295.

[6] Robert M. Bell and Yehuda Koren, "Lessons from the Netflix prize challenge," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 75, 2007.

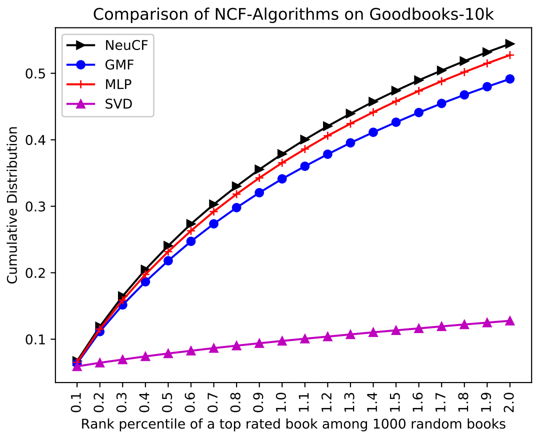


Fig. 6. Top-1000 Recommends plot for GMF,MLP,NeuCF and SVD observing the distribution of rank of a top rated book among the 1000 random books for a given user

[7] Thomas Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[8] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk, "Major components of the gravity recommendation system," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 80, 2007.

[9] Yehuda Koren, Park Ave, Florham Park, H Database Management, and Database Applications, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–434, 2008.

[10] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma, "Collaborative Knowledge Base Embedding for Recommender Systems," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 353–362, 2016.

[11] Hao Wang, Naiyan Wang, and Dit-Yan Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[12] Arkadiusz Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, 2007, vol. 2007, pp. 5–8.

[13] Nicolas Hug, "Surprise, a Python library for recommender systems," `http://surpriselib.com`, 2017.