Cooperative Neural Collaborative Filtering

Corsin Gutkowski, Adrian Meier, Lukas Schär, Elia Schudel

Department of Computer Science, ETH Zurich, Switzerland
group: MeAndTheBoys
{*gucorsin,meiera,schaerl,eschudel*}*@student.ethz.ch*

***Abstract*—The importance of recommender systems is growing ever more with the expansion of offerings found online. Recommender systems can be used directly for item recommendations or even for personalized advertisements. This is a highly competitive business and better recommendations result in higher revenue.**

This paper describes a new approach to the problem, combining multiple approaches from classical matrix factorization and leveraging the power of neural networks. Our experiments show that this outperforms each one of the individual methods, leading to a RMSE of 0.98104 on the experimental data.

I. Introduction

With the ever-growing usage of massive online services such as Netflix or Amazon, it becomes more difficult to provide the users with personalized recommendations. Since custom support is not an option due to the enormous number of users and supplied goods, Collaborative Filtering (CF) algorithms are often used to produce these recommendations.

The goal of CF is to generate recommendations for a user based on their past interactions and the interactions of all the other users with the available items. The basic assumption behind these approaches is that there are latent factors represented in shared lower dimensional space that apply to the users and the consumed goods, which allow us to justify these recommendations. A classical approach to this problem is Matrix Factorization (MF), for example with SVD-esque strategies [1]. More recently, there were many approaches trying to leverage the power of neural networks to CF [2][3].

The competition around the Netflix prize manifested that combinations of several methods prevail over single methods [4]. This led us to the idea of combining classical matrix factorization techniques with modern neural network (NN) approaches. The novelty of our approach lies therein that we apply MF for data completion, and train a NN on a low rank approximation of the completed data to predict the final ratings. This approach delivered promising results surpassing the best results of each method on its own, implying the usefulness of the direction of cooperative neural collaborative filtering (C-NCF).

Our contribution includes the following:

1) We present cooperative neural collaborative filtering, a combination of various matrix factorization models and a neural network.
2) We show that cooperation between various traditional matrix completion methods improves the prediction score beyond each individual score.
3) Based on experimental results we conclude the promising nature of this novel approach to CF.

II. Data

The dataset[1] we work with contains roughly 1.2 million ratings a set of 10000 users made for a set of 1000 items. Due to the context of the Netflix Prize we use the words 'item' and 'movie' interchangeably throughout the paper. All ratings are integer values between 1 and 5. The ratings can be represented in a matrix M, where $r_{u,i}$ holds the rating of user u set for item i. That way we have a sparse matrix where close to 12% of the cells contain ratings and the remaining cells are empty. The entries of M are such, that no entire row nor column is empty, i.e. every user provided some ratings and no item is without any rating. This data set is lacking any additional information about users and items such as age of the user, category of the item or time of the rating. For data sets containing additional information, there are advantageous ways of utilizing this information to generate recommendations to the users [5] [6] . In our scenario we have to, solely based on the provided ratings, extrapolate rating-predictions for missing entries of the matrix M. This we call imputation. The predicted ratings are evaluated by a hidden test set of additional ratings. Aiming to produce as small an error as feasible, our prediction is scored according to the root mean squared error (RMSE).

III. Models and Methods

A. Individual models

1) Matrix Factorization: For the purpose of matrix factorization we drew on several approaches. The most basic one of them being the k-nearest neighbor (kNN) method. This method imputes missing entries by similarities of users and their preferences of items, i.e. user-item similarity. Further we used an iterative imputation method which in contrast to kNN tries to model each user as a function of other users compared to the whole rating matrix, resulting in

[1] https://inclass.kaggle.com/c/cil-collab-filtering-2019/data

more complex but more accurate recommendations. Our last approach focuses on an imputation method based on soft iterative thresholding of SVDs [7].

Each model described has its own strengths and weaknesses. Thus we tried to improve results of the MF by combining the individual results, yielding the cooperative part of this approach. Trivially one could compute the uniform average of the imputed matrices. A more elaborate implementation is to search for the best possible linear combination, taking the uniform average into account as well. By doing so we achieved the best score from among all MF approaches we tested. We compare the different models according to their achieved RMSE in section IV.

2) Neural Network: The neural network employed in our work is based on the paper from [2] modified to only use explicit feedback. We used feed-forward networks, where the output of a layer serves as input to the next layer, and the final output $\hat{\mathbf{y}}$ is the prediction. The architecture of the network can be expressed as follows:

$$\begin{aligned}
\mathbf{x_1} &= \phi(\mathbf{i}, \mathbf{u}) = [\mathbf{e_i}, \mathbf{e_u}] \\
\mathbf{x_2} &= a_1(\mathbf{W_2 x_1}) \\
&\dots \\
\mathbf{x_k} &= \begin{bmatrix} x_{k-1} \\ M[i,u] \end{bmatrix} \\
&\dots \\
\mathbf{x_L} &= a_{L-1}(\mathbf{W_L x_{L-1}}) \\
\hat{\mathbf{y}}_{\mathbf{iu}} &= \sigma(\mathbf{x_L})
\end{aligned}$$

where $\phi(i, u)$ is the embedding function constructing embedding vectors for users $\mathbf{e_u}$ and items $\mathbf{e_i}$ based on their indexes $\mathbf{u}$ and $\mathbf{i}$. These embedding vectors are then concatenated and input to the MLP. With

$$\mathbf{W_i} = \begin{bmatrix} w_{i1}^T \\ w_{i2}^T \\ \dots \\ w_{in_i}^T \end{bmatrix}$$

being the weight matrix of layer i containing the row-wise weight vectors of each unit $j \in [1, n_i]$ of the corresponding layer with the bias term absorbed into these vectors. $a_i, \mathbf{x_i}$ denote the activation function and the output of the i-th layer, respectively. The layer $\mathbf{x_k}$ is where we combine our approaches, and feed the result from the matrix factorization into the last part of the network. To ensure predictions within range of the ratings we define $\sigma(\mathbf{x_L})$ as follows

$$\sigma(x) = \begin{cases} 1 & \text{for } x < 1 \\ 5 & \text{for } x > 5 \\ x & \text{otherwise} \end{cases}$$

The implementation of the NN is based on the code of [8], which builds a deep model of 10+ hidden layers. The original model that was implemented by [8] is further called NCF, as it implements the network described in [2]. It takes a user and a movie index as input and analyzes the correlation of the latent factors of the embedding vectors of the input. To find good parameters for the network (in reasonable time) we employed a randomized grid search. This way we trained NNs with 50 different settings for 10 epochs. Based on the smallest mean square error achieved on the evaluation set, we chose the parameters for our experiments. Following [9] we used the rectified linear unit (ReLu) as activation function on each hidden layer, as it surpasses most other activation functions for deep-neural networks. The models parameters were trained with the default Adam optimizer [10].

B. Combination

Motivated by the work of [2], where matrix factorization is performed by a neural network, we decided to take this approach and modify the part of the generalized matrix factorization (GMF). The architecture of the original NN can be seen in figure 1. Our approach uses the strengths

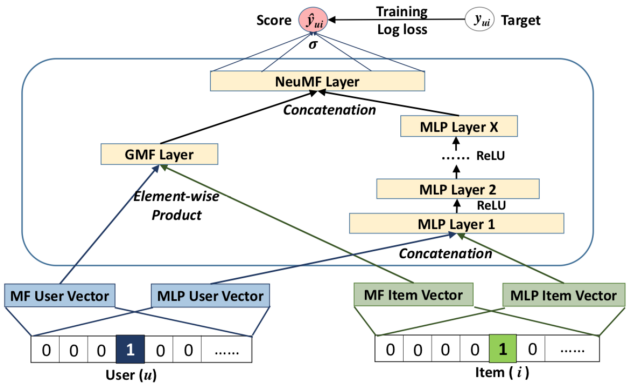

Figure 1. NCF using GMF, taken from [2]

of traditional matrix completion techniques described in section III-A1 and the strength of neural networks to find non-linear interactions between users and movies. Thus we pulled the GMF layer of figure 1 outside of the NN, and replaced it with our cooperative matrix imputation. The computed matrix was then fed into the NN to be concatenated with the output of the MLP structure of the NN. We call the resulting model C-NCF (Cooperative Neural Collaborative Filtering). The input to the C-NCF looks as follows. Let the movie index be i, let the user index be j and let M be the matrix produced by the matrix completion technique, then the input to our NN looks like this: $(i, j, M[i, j])$. Since M agrees on the observed values with the original data set, we cannot use M directly for training of the NN. Thus we use a low-rank approximation (with rank 50) of M to train the network. Again a randomized grid search helped us find good parameters comparatively quickly.

C. Experimental Setup

The following section is split into two parts. First we describe the setup of the experiments to determine the best scoring matrix factorization and the corresponding evaluation. Second we state the setup and the manner of evaluation of the experiments for the neural networks.

1) Matrix Factorization: To evaluate our models for MF we made a 90:10 split on the given ratings into a training and validation set. Using the training set, we impute the sparse matrix M for each method. Then we evaluate these matrices with the validation set with respect to the RMSE:

$$RMSE(M) = \sqrt{\|\vec{m} - \vec{v}\|^2}$$

where $\vec{m}$ is a vector representation of the matrix entries and $\vec{v}$ is a vector representation of the validation set: Each entry v_a of $\vec{v}$ represents an entry $r_{i,j}$ in the matrix of the validation set, and m_a is the corresponding predicted value $\vec{M}_{i,j}$.

For the sake of finding the optimal linear combination of all four approaches (the fourth approach is using the average of the other three) weights have been added. Thus, we solve the following problem:

$$\arg\min_{\vec{w}} \left((\sum_{i=1}^{4} w_i \vec{m_i}) - \vec{v} \right)$$

where w_i represent the weighting coefficients of each model i.

To pass the best possible matrix to our NN the different models have been trained on all the given data. The pre-computed weights w_i on the validation set are assumed to be fairly similar to the exact weights. Hence these pre-computed weights are used for assembling the final matrix.

2) Neural Networks: We compared the performance of a NN that predicts based only on user and movie embeddings, to a NN which additionally is fed the resulting matrix from our cooperative matrix factorization. Both the NCF and our C-NCF were trained with the same settings, the predicted scores were post-processed to ensure they lie in the range $[1, 5]$ and both prediction files were uploaded to kaggle [2] where the resulting score could be directly compared. The data used to train the C-NCF was only pre-processed by adding the predicted ratings (described in section III-A1) obtained from our MF predictions. We then trained our model on 90% of the data and used the remaining 10% of the provided ratings as validation set.

IV. Results

Table I shows the results of the different models for MF. As a baseline algorithm we simply filled missing matrix entries with the mean rating for the given item.

[2]https://inclass.kaggle.com/c/cil-collab-filtering-2019/overview

MF model	RMSE
Mean	1.02982
kNN	1.18763
IterativeImputer	1.06490
SoftImpute	0.99350
Uniform Average	1.02012
Linear Combination	0.98344

Table I: Validation Scores on different MF models

model	RMSE
NCF	1.01233
C-NCF SI	0.98210
C-NCF LC	0.98104

Table II: Comparison NCF to C-NCF

One can observe that the SoftImpute method performs significantly better compared to kNN or IterativeImputer. Using only the mean as imputing method delivered a surprisingly good result. With a linear combination the best MF could be achieved.

Table II summarizes the results of our experiments, where we compare the performance of our C-NCF model against the baseline NCF. C-NCF SI denotes a C-NCF built with a soft impute matrix factorization, whereas C-NCF LC stands for C-NCF using the linear combination of different matrix factorizations. We can see that C-NCF outperforms the baseline with either matrix factorization technique used. The best score is achieved by using a linear combination of all matrix factorization techniques, as described in section III-A1. C-NCF performed about 3% better than NCF, and it also outperformed every single matrix factorization technique as can be seen by comparing table I and table II.

V. Discussion

Our approach in contrast to NCF enables the matrix factorization to be done using the best mathematical models established in the past and yet use the non-linearity of embeddings to explore user-movie interactions by a NN. On the downside we find that our model is not extendable to unseen users or movies. To be able to predict ratings for users that joined after the model was trained, the model needs to be trained again with the new users added to the dataset. Further our experiments cannot be used to make a statement whether C-NCF in general outperforms NCF. We found that on equal settings and with our implementation C-NCF outperforms NCF, but we cannot make a statement about which method prevails were both models configured with their respective optimal parameters. Further work could be done by exploring the k-approximation that is used to train the NN of C-NCF. Varying this parameter could lead to better performance, which we did not explore in our experiments.

For the cooperative matrix factorization we could imagine that using a NN to deduce the best coefficients to linearly combine each entry would perform better, than simply finding a linear combination of the entire matrices - as done in this paper. On top of that, we expect a further improvement by including additional MF techniques in linear combination.

VI. Summary

We found that C-NCF is a promising way of performing the task of collaborative filtering. First we saw that combining several matrix factorization techniques serves the purpose of getting better predictions, and then we were able to observe that applying neural networks on top of the predicted matrices further enhances the predictive strength of our model.

References

[1] S. Funk, "Netflix update: Try this at home," Jun 2006. [Online]. Available: https://sifter.org/~simon/journal/20061211.html

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[3] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering," *arXiv preprint arXiv:1605.09477*, 2016.

[4] A. Töscher, M. Jahrer, and R. M. Bell, "The bigchaos solution to the netflix grand prize," *Netflix prize documentation*, pp. 1–52, 2009.

[5] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *In IEEE International Conference on Data Mining (ICDM 2008*, 2008, pp. 263–272.

[6] R. Bell, Y. Koren, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 08, pp. 30–37, aug 2009.

[7] Z. S. Jian-Feng Cai, Emmanuel J. Candes, "A singular value thresholding algorithm for matrix completion," 2008. [Online]. Available: https://arxiv.org/pdf/0810.3286.pdf

[8] "Neural networks for collaborative filtering." [Online]. Available: https://nipunbatra.github.io/blog/2017/neural-collaborative-filtering.html

[9] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.