An improved SVD approach for Collaborative Filtering

Florian Morath, Tobias Grob, Jonas Gude and Lucien Schaller
Group: yucca, Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—The goal of Collaborative Filtering (CF) is to predict a user's unknown rating for a certain item, based on its own available ratings for other items, and the preferences of other users. In this project we focused on two approaches to CF: latent factor models and neighborhood models. The methods we implemented include: Average-over-items, Average-over-users, SVD with dimension reduction, ALS, kNN and NMF. The best result we got with an improved version of SVD, referred to as SVD+. Compared to the basic SVD with dimension reduction, this gained a 7.66% improvement on the Kaggle public dataset.**

I. Introduction

With the recent advancement in machine learning techniques, recommender systems have become a popular way to predict the preferences of a user over a set of items. Such systems have become an integral part for almost all e-commerce and entertainment platforms, such as Amazon or Netflix [1], [2].
One approach to design such recommender systems that is widely popular nowadays is to use Collaborative Filtering. Collaborative Filtering methods are based on collecting and analyzing large datasets of information about the users' preferences and predicting what users will like. The key advantage of this approach is that it does not require and actual understanding of the items themselves for accurately recommending complex items such as music or movies.
In this project we explore two approaches, one is an improved SVD variant that is based on the famous idea of Simon Funk in 2006 during the Netflix Prize competition. His idea is to use SGD to minimize the SVD approximation error and to drop the orthogonality constraints. This allows him to only consider the available ratings and thus not having to deal with the initialization of missing ratings [3], [4]. The other approach applies deep learning methods to achieve a more accurate prediction model.

The paper is organized as follows. In Section 2, we first introduce the CF models used in our prediction system, with a focus on our two main approaches, an improved SVD variant and a Deep-Learning based approach. In Section 3, we describe the experimental setup and present the obtained results. Finally, we discuss our results in Section 4 and present a short summary in Section 5.

II. Models and Methods

In this section we present the different models used to predict the missing ratings. We use several algorithms as a baseline, among those are: AoI, AoU, SVD, ALS, k-NN and NMF. Furthermore we present our two advanced models SVD+ and DCF.

A. Average over Items (AoI)

In this simple baseline algorithm, for each user, we predict its missing ratings by the average of that user's available ratings.

B. Average over Users (AoU)

In this simple baseline algorithm, for each item, we predict its missing ratings by the average of the available ratings for that item.

C. SVD Basic

In this approach, we represent the ratings as entries in a matrix, with a row for each user and a column for each item. Now the assumption is that the data has an underlying structure where each user and each item has a representation in the same lower-dimensional concept space. The idea behind the approach is to apply SVD to the rating matrix to find the low-dimensional embeddings. The missing ratings can be initialized for example by using Average over Items. By discarding small singular values we can remove noise and approximate true rating matrix.

D. Alternating Least Squares (ALS)

A rating of user u for item i is estimated based on the following equation:

$$r_{ui} = \mu + b_u + b_i$$

where μ is the overall average rating, b_u and b_i are biases that capture tendencies of users to rate higher/lower and tendencies of items to be rated higher/lower.

The regularized square loss which we minimize looks as follows:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - (\mu + b_u + b_i))^2 + \lambda(b_u^2 + b_i^2)$$

The optimization method used to minimize the loss function above is ALS which alternatively updates b_u and b_i to convexify the loss function.

E. *Non-Negative Matrix Factorization (NMF)*

NMF is a technique where we factor a matrix V into two matrices W and H, which are non-negative. This approach is similar to the singular value decomposition SVD.
The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = q_i^T p_u$$

where user and item factors are kept positive.

F. *k-Nearest-Neighbor (k-NN)*

KNN based methods will predict ratings based on similarity to other users and items. For example in a user-based approach, a user u rates an item i based on ratings of other users that are similar to u. For the similarity metric we use the Pearson correlation coefficient.

The prediction is computed as follows [5]:

$$r_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v)(r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

where $N_i^k(u)$ are the the k users, that rated item i, which are most similar to user u. b_{ui} is the global average plus a user and item bias:

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

The prediction basically takes b_{ui} and adds the deviation of similar users that rated item i to it.

G. *Improved SVD (SVD+)*

Computing the eigenvectors of XX^T and X^TX, where X is the rating matrix, and the associated eigenvalues is not the only way of computing the SVD of a rating matrix! We can also compute it based on a minimization problem:

$$min_{p_u,q_i} \sum_{x_{ui}} (r_{ui} - p_u * q_i)^2 \text{ s.t. } p_u\text{'s and } q_i\text{'s are orthogonal}$$

Now the main idea of Simon Funk is to just ignore missing ratings and the orthogonality constraints. He uses SGD to optimize the objective and also adds regularizers which are important because we can easily overfit. In the end we thus have an approximation of SVD.

To go one step further one can also incorporate user and item biases. Those are based on the observation that some users tend to give higher/lower ratings then others. And some items tend to receive higher/lower ratings than others (relatively seen).

The estimate of a rating is then

$$r_{ui} = \mu + b_i + b_u + q_i^T p_u$$

where μ: average rating over all movies, b_i: deviation of item i from average and b_u: deviation of user u from average.

The final objective can the be written as follows:

$$\begin{aligned} min_{p_u,q_i} \sum_{x_{ui}} (r_{ui} - (\mu + b_u + b_i + p_u^T q_i))^2 \\ + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2) \end{aligned}$$

H. *Deep Collaborative Filtering (DCF)*

The first idea is to create a neural network which learns a lower dimensional embedding for users and movies and predicts a missing rating with the dot product of the embeddings.

Enhancing this idea, one can add additional fully connected layers to the network, to learn a mapping from the dot product of the embeddings to the predicted rating.

Even further, one can introduce dropout layers to avoid overfitting.

III. Experiment

A. *Data*

The data set consists of 1176952 movie ratings, generated from 10000 users and 1000 movies. The rating values are integers between 1 and 5.

When analyzing the principal singular values of the input data, we can observe the amplitude droping very quickly and forming a knee at around singular value number 15, as can be seen in Figure 1.

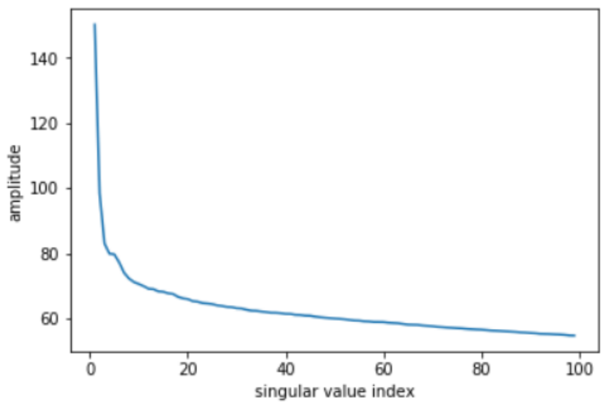

Figure 1. Principal Singular values of dataset

This already suggests that the underlying information of the dataset can be represented in a much lower-dimensional space with minimal loss.

B. *Metric*

The performance of evaluated algorithms is measured by the root-mean-squared error (RMSE).

$$RMSE = \sqrt{\frac{1}{[R]} \sum_{u,i \in R} (\hat{r}_{ui} - r_{ui})^2}$$

where R is the set of user/movie ratings to be tested. The resulting scores correspond to the achieved public scores on Kaggle.

To tune the hyperparameters of our models we used ten-fold cross-validation.

C. Experimental Setup

Computationally intensive parts like grid search with cross-validation or neural network training were performed on the ETH Leonhard cluster on around 20 cores with 20GB of RAM per core.

D. Results

In Table I we can see the best Kaggle public scores achieved by each of our six baseline algorithms and our two advanced approaches. We can observe that the two simplest models AoI and AoU performed relatively poor, as was expected.
Furthermore, even the simple SVD approach does not perform well. In addition we can observe that the more advanced baseline algorithms: ALS, k-NN and NMF all perform better than the simpler versions.

Our SVD+ algorithm achieves the best result and outperforms all other implementations with a score of 0.98144. On the other hand, our DCF implementation does not perform significantly better than the baseline implementations, in some cases it even performs worse with a score of 1.05401.

Algorithm	RMSE
SVD+	0.98144
KNN	0.99016
NMF	0.99203
ALS	0.99952
AoI	1.03198
DCF	1.05401
SVD	1.05666
AoU	1.09593

Table I
KAGGLE PUBLIC SCORES OF ALL EVALUATED MODELS.

Figure 2 shows the training error for the SVD+ implementation. We notice a very fast drop in loss over the first few epochs, followed by a steady but slower decrease.

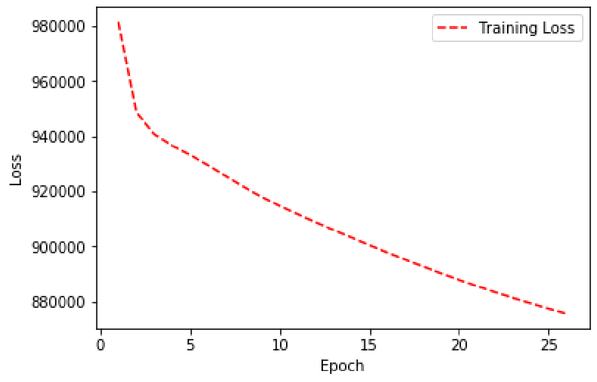

Figure 2. SVD+ training error over time.

The performance of the SDV+ implementation depends heavily on the chosen input parameters for regularization and the learning rate. We performed an extensive grid-search to tune the various hyper-parameters.

In Figure 3 we can observe both the training and the test error of the deep collaborative filtering implementation. We can observe that the training error decreases steadily with each epoch. Furthermore, the test error decreases rapidly in the beginning and then starts to slightly increase which is a sign of overfitting.

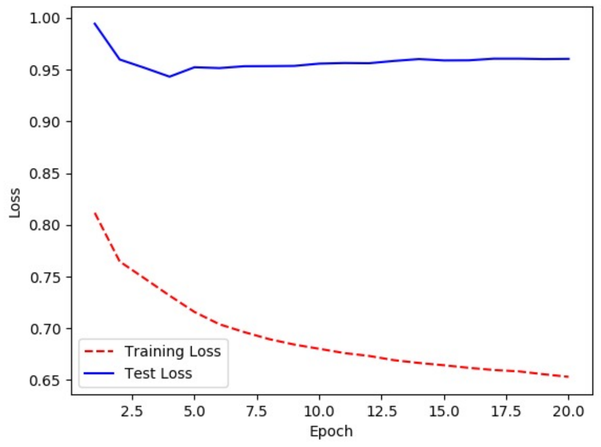

Figure 3. DCF training loss and test loss over time based on RMSE.

IV. Discussion

A. SDV+

Given the analysis of the singular values of the dataset, shown in Figure 1, we were confident that an SVD-based solution will yield good results. In the end, our approach outperformed all our other approaches. Furthermore, since this SVD approach incorporates stochastic gradient descent it runs really efficient even for large data matrices.

B. DCF

The deep collaborative filtering approach did not perform as well as hoped. However, we believe that the main reason for this was the limited resources (time and computational resources). While looking for suitable methods we tried a wide number of algorithms and approaches. A lot of effort was spent optimizing the parameters for our SVD+ implementation. In the end, we had only limited time to optimize our neural-net approach. Nonetheless, we are convinced that a neural-net based approach does yield better results with better parameter optimization.

V. Conclusion & Future Work

To conclude, in this project we developed two novel techniques for collaborative filtering. The first technique is an adapted version of SVD, and the second technique is based on a deep neural network. These two novel solutions were tested against 6 baseline algorithms. Our SVD based version outperformed all other implementations and achieved the best RMSE. However, our research showed that neural-networks are considered the state of the art approaches to collaborative filtering. Thus we are confident that by further developing this approach, still better results could be achieved.

References

[1] B. Smith and G. Linden, "Two decades of recommender systems at amazon.com," *IEEE Internet Computing*, vol. 21, no. 3, pp. 12–18, May-June 2017. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MIC.2017.72

[2] J. Bennett, S. Lanning, and N. Netflix, "The netflix prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[3] S. Funk, "Netflix update: Try this at home." [Online]. Available: http://sifter.org/simon/journal/20061211.html

[4] R. Bell, Y. Koren, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, 08 2009. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MC.2009.263

[5] Y. Koren, "Factor in the neighbors: scalable and accurate collaborative filtering, 2010."