CIL Project: Collaborative Filtering

Ulla Aeschbacher, David Eschbach, Xiyue Shi, Fanyi Zhou
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Recommender systems and collaborative filtering have become very popular lately. For example streaming platforms use them to recommend new movies to their users based on ratings that users have previously assigned to other movies. For this purpose recommender systems try to exploit similarities between the rated items and potentially also between users. To achieve this, collaborative filtering aims to learn a low dimensional representation of those items.**

In this paper we present our approach to the problem, which is based on a combination of Singular Value Decomposition (SVD) and k-nearest-neighbors (KNN) working with the Pearson correlation. Our final solution has a Root Mean Square Error (RMSE) of 0.98362.

I. Introduction

Collaborative filtering has been known for a while and there are many important contributions by research. For example in [1] there is an overview and an empirical analysis of multiple algorithms including memory-based algorithms predicting the rating of an item by the active user based on a weighted sum of the other users' rating of that item. Another example taken from [1] are model-based algorithms like the Bayesian Network Model that take a probabilistic approach to the problem by setting the rating to its expected value according to some probability distribution. In [2] the authors focus on SVD and in particular elaborate on the importance of regularization techniques. They further demonstrate that the combination of several distinct predictors can lead to even better results. Many of the more recent contributions make use of deep neural networks. For example in [3] the authors introduce neural collaborative filtering. This is a neural network technique based on a combination of generalized matrix factorization and multi-layer perceptron optimizing the logarithm of a likelihood. In [4] the authors rely on autoencoders to solve the problem of collaborative filtering. One of the main advantages of their system AutoRec compared to multiple other collaborative filtering solutions is that it directly optimizes the Root Mean Square Error (RMSE). Other contributions rely on clustering techniques like k-nearest neighbors. For example [5] presents a cluster-based collaborative filtering framework that puts much effort into smoothing clusters.

Similar to [2], our contribution is based on a combination of multiple predictors. This predictors include KNN based solutions and dimensionality reduction through SVD. We also experimented with neural networks in particular by building a bidirectional Long Short-Term Memory (LSTM) network that directly predicts ratings. Furthermore we tried to achieve the dimensionality reduction with an autoencoder built of LSTM cells. However, both of these aproaches did not yield an improvement on our results, so we decided to drop them.

Model	**RMSE**
User-User k-NN model based on MSD similarity	1.010925
Item-Item k-NN model based on MSD similarity	0.995846
User-User k-NN model based on Pearson correlation similarity	0.999997
Item-Item k-NN model based on Pearson correlation similarity	0.991212
SVD++ model factors=10	1.003788
SVD++ model factors=20	1.003131
SVD++ model factors=30	1.003923

Table I
RMSE of different models

II. Our Novel Solution

Our novel solution is based on the idea of fusion by ensemble averaging, a process of creating multiple models and averaging their outputs to produce a desired output[6]. Usually a model created by ensemble averaging performs better than any individual model, because it "averages out" errors of each individual model. To evaluate our models, we use a random 80:20 split over the original training dataset. All algorithms ran on the same split.

For this project, we first constructed k-NN models with different similarity metrics and different perspectives, and SVD++ models with different sizes of factors (the same values were used for other parameters). Table I shows the Root-Mean-Squared-Error of each model. Item based k-NN models perform better than user based k-NN models, and k=NN models based on Pearson correlation similarity perform better than k-NN models based on Mean Squared Difference (MSD) similarity. There is not much difference between SVD++ models with different sizes of factors. Among those individual models, item-item k-NN model based on Pearson correlation similarity performs best, and its RMSE on our validation set is 0.991212.

A simple fusion of these models can then be constructed by averaging outputs of those basic models. We tried to construct fusion models with different combinations, and Table II shows results of part of our experiments. Previous works suggest to apply fusion to k-NN models or matrix

Fusion Model	**RMSE**
k-NN-User-User-Pearson + k-NN-Item-Item-Pearson	0.990154
k-NN-Item-Item-MSD + k-NN-Item-Item-Pearson	0.988693
k-NN-User-User-MSD + k-NN-User-User-Pearson + k-NN-Item-Item-MSD + k-NN-Item-Item-Pearson	0.990045
SVD++ factors=10 + SVD++ factors=20 + SVD++ factors=30	0.999779
SVD++ factors=20 + k-NN-User-User-Pearson + k-NN-Item-Item-Pearson	0.987597
SVD++ factors=20 + k-NN-Item-Item-MSD + k-NN-Item-Item-Pearson	0.988114
Fusion of all models	0.989956

Table II
RMSE OF FUSION MODELS

factorization models only[6], but in our solution we apply fusion to both k-NN models and matrix factorization models. All fusion models perform significantly better than individual models. The model which averages outputs of SVD++ model with factor size equals to 20, User-User k-NN model based on Pearson correlation similarity and Item-Item k-NN model based on Pearson correlation similarity achieved the best result, and its RMSE on our validation set is 0.987597. Comparing the model with the best individual model in our previous experiments, the fusion model reduces the RMSE by 3.647%. With the model, we achieved a score of 0.98362 on kaggle, which is the best among all of our submissions.

III. Comparing To Baselines

We are comparing our novel solution to two others such that we can see how much improvement was achieved. The first of the two other solutions is the one from Assignment 3. There we were first completing the missing ratings in the matrix by setting them to the average over all observed ratings for a particular item. Then we tried to find an underlying structure in the data by performing SVD, where $X^{train} = UDV^T$. We varied the number k of eigenvalues used and truncated U and V accordingly, by taking the first k columns of each of the matrixes. Figure 1 shows the Root-Mean-Squared-Error for k between 0 and 95, in intervals of 5. In Figure 2 we can see the error more clearly for k between 5 and 95. Apparently, taking the biggest 10 eigenvalues is giving us the best approximation to the data structure. The best result we achieved on kaggle this way was an error of 1.16473.

The second solution we are presenting here for comparison is a neural network. We trained a bidirectional LSTM. It has a bias layer, so that we could take into account that different people are rating higher or lower in general. We tried different dropout values, numbers of nodes, batch-sizes and trained over various epochs. In Figure 3 shows the Mean-Squared-Error loss over ten epochs of the neural network. We set a dropout of 0.75, 40 nodes in each layer and a batch size of 50, as these values were performing the best in our experiment. The best result we achieved on kaggle this way was an error of 1.04627. There are people that achieved much better results with a neural network [7], but our feeling is that our data was too sparse for this.

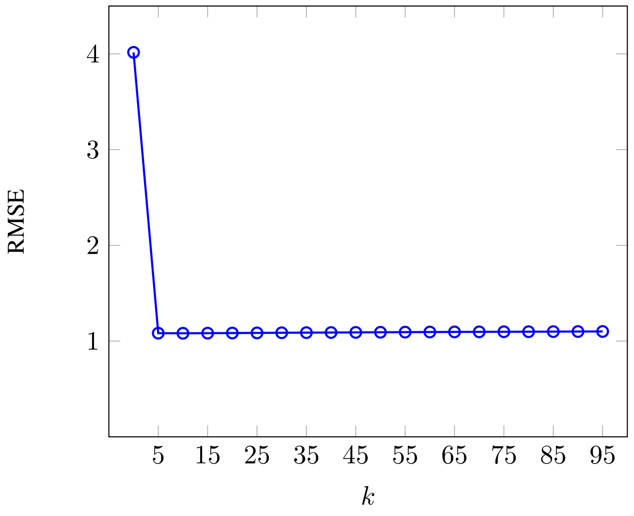

Figure 1. RMSE in SVD with different k

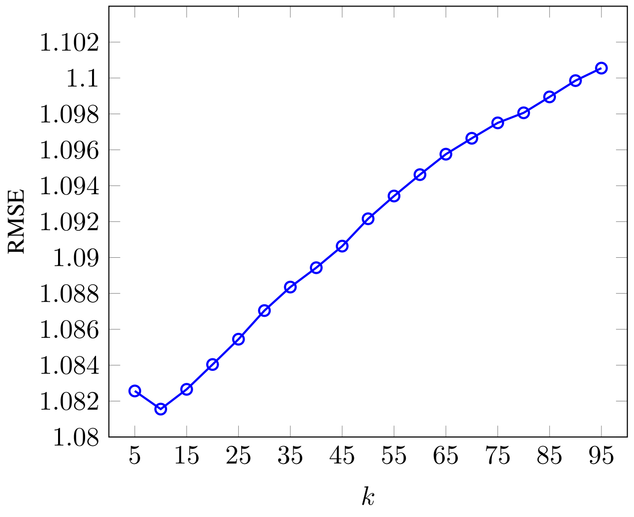

Figure 2. RMSE in SVD with different k, shown in more detail

Comparing the two solutions with our novel approach result of 0.98362 we can see that we managed to get much better results with.

IV. Summary

In summary, our novel solution adopts an ensemble learning approach and averages the results of different prediction algorithms, namely SVD, KNN-User-User-Pearson and KNN-Item-Item-Pearson. Traditional ensemble learning approaches usually train multiple models of a single

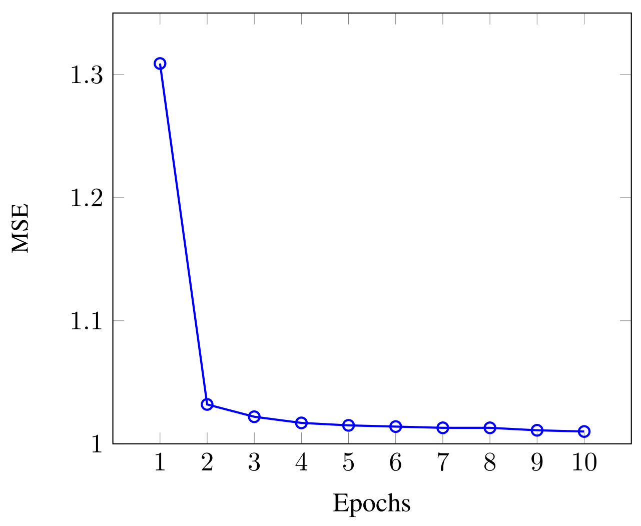

Figure 3. MSE loss in LSTM over 10 epochs.

learning algorithm. Here, our novel fusion approach managed to generate better results by first choosing the best parameters for the base collaborative filtering algorithms, then combining models with different algorithms as well as different parameters of the same algorithm as in a traditional approach. Experimental results suggest consistent better performance when several different algorithms are used. The final ensemble is then selected based on cross-validation results in a systematic comparison of different algorithm combinations. This approach is also flexible and time-efficient in the case where more base CF algorithms are added and cross-validation needs to be carried out for a new ensemble. Since our solution computes a simple average of selected models, investigation into the performance of a weighted sum and how to obtain the optimal weights in the final ensemble might be required to achieve further improvement to the prediction results.

References

[1] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

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

[3] 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.

[4] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 111–112.

[5] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 114–121.

[6] A. Bar, L. Rokach, G. Shani, B. Shapira, and A. Schclar, "Boosting simple collaborative filtering models using ensemble methods," 2012.

[7] G. Tseng. (2017) Clustering and collaborative filtering - implementing neural networks. [Online]. Available: https://medium.com/@gabrieltseng/clustering-and-collaborative-filtering-implementing-neural-networks-bccf2f9ff988.

[8] C. Olah. (2015) Understanding lstm networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[9] M. A. Nielsen. (2015) Neural networks and deep learning. [Online]. Available: http://neuralnetworksanddeeplearning.com/