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 a rating 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 systems aim 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-neighbor (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 from 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 from [1] are model-based algorithms like the Bayesian Network Model that take a probabilistic approach to the problem. In [2] the authors focus on SVD and in particular elaborate on the importance of regularization techniques. They further demonstrate how effective the combination of several distinct predictors can be. More recent contributions make use of deep neural networks. For example in [3] the authors introduce neural collaborative filtering which is a neural network technique based on a combination of generalized matrix factorization and multi-layer perceptron optimizing a log-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 in [5] a cluster-based collaborative filtering framework that puts much effort in smoothing clusters is built.

Our contribution is based on a combination of multiple 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 directly predicting ratings and we tried to achieve the dimensionality reduction with an autoencoder. However, both of these aproaches did not yield an improvement on our results, so we decided to drop them.

II. Our Novel Solution

III. Comparing To Baselines

We are comparing our novel solution to two others so that we can see how much improvement was done. The first of the two other solutions is the one from Assignment 3, where 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 vaired the number k of eigenvalues used and truncated U and V accordingly. 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 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 trid different dropout values, number of nodes, batch-sizes and trained over various epochs. In Figure 3 you can see the Mean-Squared-Error loss over ten epochs of the neural network. The best result we achieved on kaggle this way was 1.04627. There are people that achieved much better

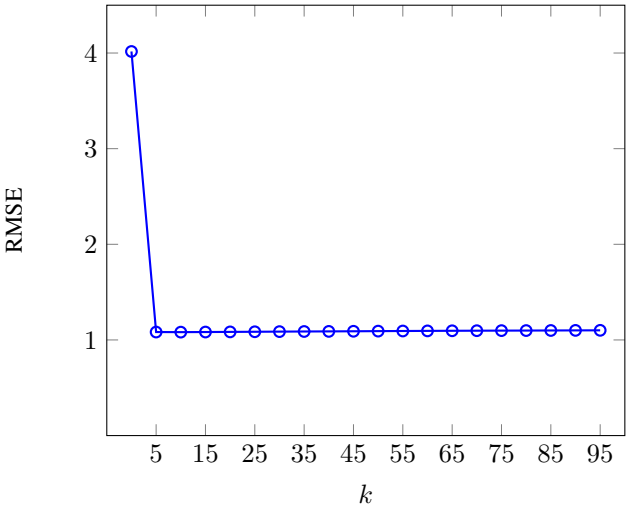

Figure 1. RMSE in SVD with different k

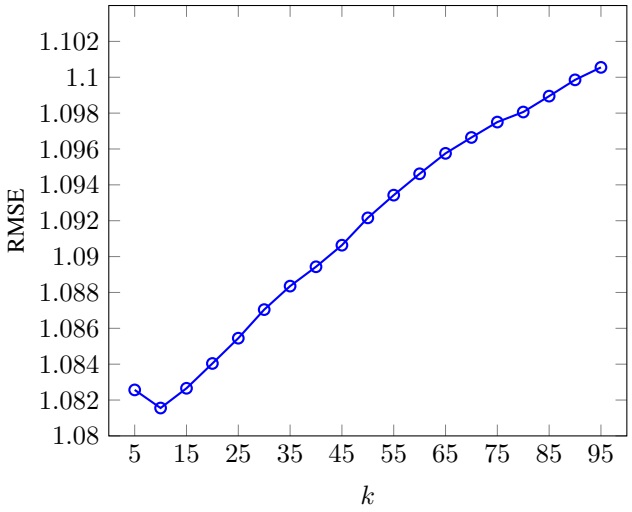

Figure 2. RMSE in SVD with different k, shown in more detail

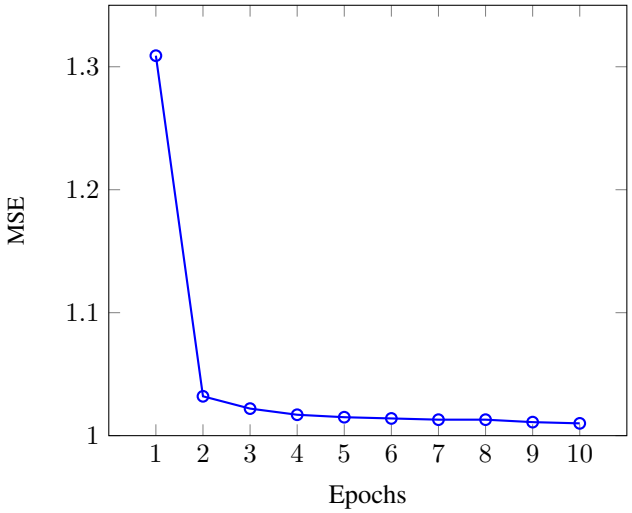

Figure 3. MSE loss in LSTM over 10 epochs.

results with a neural network [6], but our feeling is that our data was too sparse for this.

Comparing the two solutions with our novel approach result of 0.98362 we can see

IV. Summary

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] G. Tseng. (2017) Clustering and collaborative filtering - implementing neural networks. Https://medium.com/@gabrieltseng/clustering-and-collaborative-filtering-implementing-neural-networks-bccf2f9ff988.