Can You Factor in your Neighbours?
A Hybrid Collaborative Filtering Algorithm

Fizza Zafar, Supraja Sridhara
Department of Computer Science, ETH Zurich, Switzerland (Team Name: Duel)

***Abstract*—Collaborative filtering is a technique used by recommendation systems, wherein ratings of other users and items are used to predict the rating for a given (user, item) pair. To this end, popular collaborative filtering algorithms use matrix factorization and neighbourhood based methods. We propose a novel approach that combines ideas of both factorization and neighbourhood based methods. We obtain ratings by considering hidden factors for the overall ratings matrix as well as sub-matrices obtained by clustering users and items. We evaluate our approach on the provided dataset and compare it to existing approaches. On the given data, this approach gives better performance than other existing algorithms.**

I. Introduction

Web based activity, such as e-commerce and online content distribution has increased drastically in the last decade resulting in a vast amount of content becoming available over the internet. This has made filtering relevant information a cumbersome task for users. To this end, recommendation systems are used by a variety of applications to serve personalized content to users. Effective recommendation systems exploit information about users' preferences to provide personalized content. Two techniques, content-based and collaborative filtering, are used by recommendation systems. Content based filtering depends on the users' specification of interests whereas collaborative filtering systems predict a user's preferences based on data from other users and/or items.

Due to their vast applicability, a large number of collaborative filtering algorithms have been proposed in literature. These span correlation, clustering and matrix factorization based techniques, as well as their hybrids. The correlation and clustering based techniques use different metrics to identify similar users/ items and predict preferences based on them. Correlation based approaches are computationally expensive as they require pairwise correlation calculations whereas clustering based approaches are sensitive to initialization and do not take into account item synonymy. On the other hand, factorization based methods assume that there are unobserved factors that explain the rating given to different items by different users. They express both users and items in terms of these unknown factors. User ratings are predicted based on a low-rank approximation of the ratings matrix. We propose a novel method that predicts ratings based on global latent factors as well as those specific to a particular group of users/items. Our proposed algorithm gives a lower Root Mean Squared Error (RMSE), evaluated over the public test set on Kaggle.

In subsequent sections, we formulate the problem formally (section II), outline existing approaches (section III) and our proposed solution (section IV). We evaluate all these techniques (section V) and, finally, discuss the results we obtained from them (section VI).

II. Problem Formulation

Given a sparse matrix $A \in \mathbb{R}^{m \times n}$ such that $A_{ij} \in [1, 5]$ represents the rating of user i for item j, we need to find a dense matrix $B \in \mathbb{R}^{m \times n}$ with $B_{ij} \in [1, 5]$ such that it most closely approximates the true (unseen) dense ratings matrix. We consider A to be a partially observed version of the true dense matrix R. The goal is to find B such that it most closely approximates R

III. Baseline Models

We evaluated a variety of models, described below. For most of these methods, we used implementations provided in the Python Surprise [1] library.

A. Random Value Imputation

The simplest solution that we implemented randomly sampled from a normal distribution over the training data.

$$r_{ij} \sim N(\mu, \sigma^2)$$

where μ is the average observed rating and σ^2 is its variance.

B. Mean Value Imputation

Next, we predicted missing rating as the sum of mean of all observed ratings and the user/ item bias.

$$r_{ij} = \mu + b_i + b_j$$

The algorithm uses Alternating Least Squares (ALS) to find optimal values for the user and item biases such that the (regularized) mean squared error over observed ratings is minimized.

C. Matrix Factorization Methods

Matrix factorization is one of the most prominent and effective techniques used for collaborative filtering. These models map users and items to the same latent factor space and unknown ratings are predicted as dot product of the concerned user and item embedding. They find $U \in \mathbb{R}^{m \times k}$, and $V \in \mathbb{R}^{n \times k}$ such that

$$A \approx UV^T$$

Different matrix factorization techniques find the best approximation of A under different constraints and objective functions. In order to avoid over-fitting, we regularized the factors. We now describe each of them.

1) Singular Value Decomposition: Singular Value Decomposition [2], [3] of a matrix A consists of three matrices U,D and V such that

$$X = UDV^T$$

where $U \in \mathbb{R}^{m \times m}, D \in \mathbb{R}^{m \times n}$ and $V \in \mathbb{R}^{n \times n}$. Matrices U and V are the user and item embeddings, respectively and D is the diagonal matrix of singular values of A. It represents the strength of each concept in explaining A.

2) SVD with Implicit Feedback: SVDpp [3], [4], on top of the original SVD, this incorporates implicit feedback i.e. the fact that a user chose to rate a particular item, regardless of the rating value.

3) Non-Negative Matrix Factorization: Similar to other factorization techniques, NMF [5] factorizes the ratings matrix into two lower rank matrices, minimizing the difference between the original and the low-rank approximation. However, it constrains the factor matrices to be non-negative as they are interpreted as probabilities. We used a (regularized) Stochastic Gradient Descent procedure to get the factor matrices.

D. K-Nearest Neighbors

This approach predicts ratings for a particular user, item combination based on the ratings by other similar users (user-based), items (content-based) or a hybrid of both. The idea is that like-minded users will rate items on a similar scale and similar items are more likely to have similar ratings. Since there is no additional data about users/items, similarity is inferred from the available ratings. r_{ij} is predicted as the average rating of item j by 40 users most similar to user i. We used mean squared difference similarity metric for finding the nearest neighbour.

E. Neural Network Based Approaches

Learning techniques are used in collaborative filtering, owing to their ability to learn non-linearity in the data. Deep neural networks are used to extract semantic information reflecting user-user, item-item and user-item correlation. We used the fastAI [6], [7] framework to train neural networks

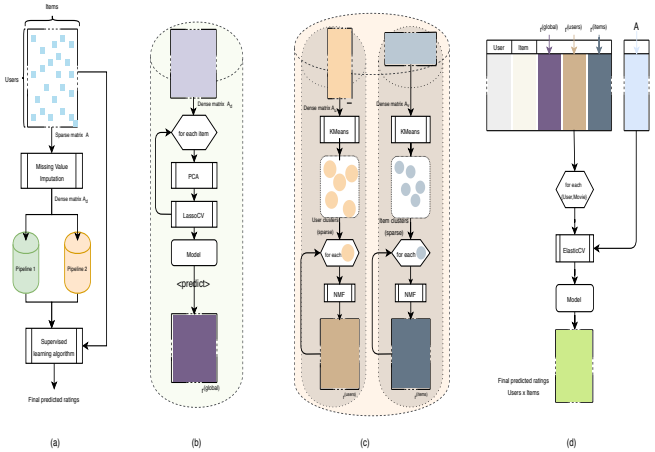

Figure 1: (a) The algorithm performs matrix factorization before branching into two independent pipelines. The pipelines then converge to solve a supervised learning problem. (b) Pipeline 1 performs PCA and LassoCV regression on the dense matrix to produce $r^{(global)}$ (c) Pipeline 2 has 2 sub-pipelines that perform clustering and matrix factorization to produce $r^{(user)}$ and $r^{(item)}$ (d) The final stage takes as input $\{r^{(global)}, r^{(user)}, r^{(item)}\}$ and uses supervised learning methods to generate final predictions

of different configurations, varying the number of layers, units per layer and activation functions. We also evaluated predictions by AutoRec, one of the state-of-the-art neural network frameworks for collaborative filtering. AutoRec [8] is a collaborative filtering model based on the autoencoder paradigm. The algorithm uses back-propagation to account for ratings being only partially observed and regularises the learned parameters to prevent overfitting. In contrast to matrix factorization approaches, the item-based AutoRec model embeds only items into latent spaces and can learn nonlinear latent representations through its activation function.

IV. Proposed Method

Matrix factorization algorithms, while widely popular, try to find latent factors that accurately describe all users and items. Neighbourhood models, on the other hand, base ratings on user/item similarity, ignoring the underlying structure of the ratings matrix. Based on the ideas presented in [9], we propose a hybrid method that predicts ratings based on 1) a user's ratings of all other items, 2) the strength of latent factors that describe all users/items and 3) the strength of latent factors that describe only a particular subgroup of users/items. The final rating blends all the above. The complete method is illustrated in Figure 1(a). We now describe the algorithm in detail.

Pipeline 1: This pipeline is illustrated in Figure1(b). It can be divided into two phases: Imputation and Regression. We use NMF to impute unobserved values and transform the sparse matrix A into a dense matrix A_d. A and A_d differ only in the unobserved values. In the regression phase, as

described in [3], we independently fit regularized regressions on all columns of A_d, using all the other columns as predictor variables. Since the dataset has 1000 items, the feature matrix has a high dimensionality. In order to make the problem more tractable, we perform principal component analysis on it to reduce dimensionality while preserving the information from all columns. We use Lasso regularization in the linear models and select the regularization parameter using cross-validation. We call the output ratings matrix $r^{(global)}$

Pipeline 2: Matrix factorization returns two factor matrices representative of user and item embedding independently. We use these as indicators of similarity between users/items. We cluster users and items based on the columns of the user and item embeddings, respectively, followed by factorization of the sparse sub-matrix obtained for each cluster. We define a sub-matrix for user cluster u_1 as all the entries in A corresponding to users in u_1. A parallel definition holds for the items sub-matrix. We call the output ratings matrix based on user and item clusters $r^{(user)}$ and $r^{(item)}$ respectively. The 2 phases of this pipeline are illustrated in Fig1(c).

Combining both pipelines: We then combine the three predictions for every user-item pair. The final rating is given as:

$$r_{ij} = \pi_0 r_{ij}^{(global)} + \pi_1 r_{ij}^{(user)} + \pi_2 r_{ij}^{(item)}$$

We find the optimal parameters π that minimize the sum of squared errors over the observed ratings. With the observed ratings in the sparse matrix A as the dependent variable and the predictions $\{r_{ij}^{(global)}, r_{ij}^{(user)}, r_{ij}^{(item)}\}$ as independent variables the algorithm performs a regularized regression to determine the mixing weights for each of the three sub-components (see Fig1(d)).

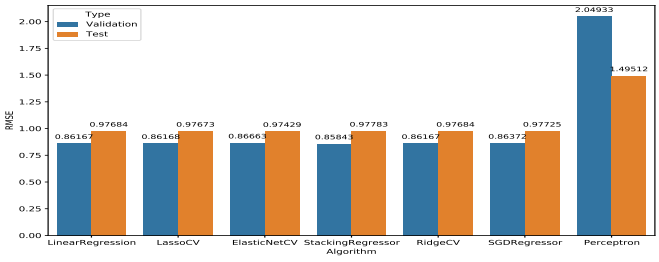

Figure 2: RMSE scores for different blending models

V. Results

A. Parameter Tuning

Due to constrained computational resources, we parameter tuned alternatively, first tuning the cluster-based parts of our algorithm (keeping the results of the first pipeline fixed) and then tuning NMF imputation on top of that (keeping the results of the previous pipeline fixed). Table I summarizes

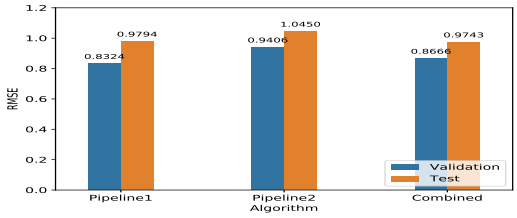

Figure 3: RMSE scores for evaluating pipelines separately

the values of the different parameters we used to train our final model.

Parameter Name	Value	Parameter Name	Value
Global NMF epochs	185	User NMF factors	30
Global NMF factors	900	Item clusters	2
User clusters	7	Item NMF epochs	10
User NMF epochs	8	Item NMF factors	30

Table I: Parameters to train final model

Clustering and Sub-Matrix Factorization: Figure 5 shows the results for a grid search over number of clusters, factors and epochs used in Pipeline 2 (see Figure 1(c)). The grid search allowed us to narrow down possible parameter combinations which we then evaluated over the public test set to get the final combination. RMSE of different combinations on the public test set can be seen in Figure 4. We chose the combination, given in Table I, that gave the minimum test RMSE, .

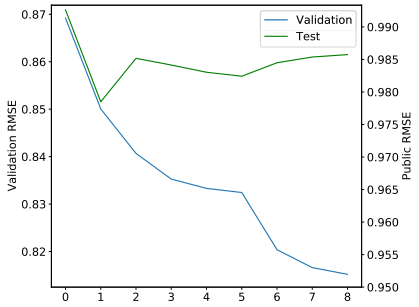

Figure 4: Validation and Test RMSE for different parameters of local factorization

Imputation and Regression: On top of the chosen parameters for the cluster-based part of the algorithm, we tried different parameters for NMF imputation. We performed a random search over the number of factors and the number of epochs for this NMF. In the first phase we searched the complete grid with factors $\in$ [100, 300] and epochs $\in$ [100, 300]. As expected, the performance of the algorithm improved with the number of factors. Moreover, for any given number of factors, epochs between 150 and 200 gave the best validation RMSE. In the second phase, we tried multiple combinations with factors $\in$ [700, 900] and epochs $\in$ [150,200]. The results of the 2 phases are summarized in Figure 6. Based on these findings, we evaluated the test

RMSE for factors = 900 and epochs = [150,175,185,200] and found 900 factors and 185 to give the best performance. Table III gives the obtained Test RMSE for different epochs.

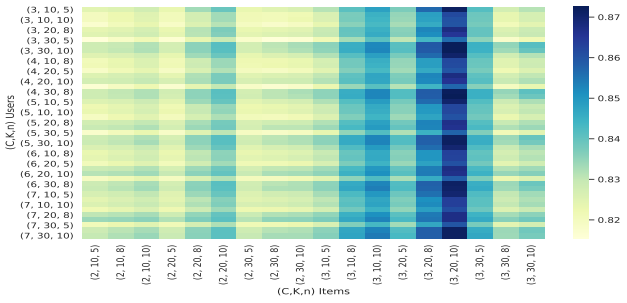

Figure 5: RMSE scores for Number of Clusters, Hidden Factors, Number of Epochs (C,K,n) for user and item clusters, produced by grid search with 2 fold cross-validation

Blending Models: We also evaluated the performance of different blending models that were available in Python's sklearn library [10]. The results are summarized in Figure 2. We observe that non-linear methods overfit very easily whereas linear models generalize better. The best performance was given by ElasticNetCV which uses a combination of lasso and ridge regression, with the optimal regularisation parameter being found using cross-validation.

B. Evaluating the Effect of Blending

To evaluate whether blending the local and global pipelines actually lead to an improvement in RMSE, we evaluated both separately on the test data set. In order to draw a fair comparison, we used the parameters that gave best results for each pipeline separately. Figure 3 gives an overview of the results. Combining both pipelines gives a better test RMSE than generating predictions from either pipeline independently. This implies that the model does indeed have something to gain from the latent factors discovered through clustering and sub-matrix factorization.

Algorithm	Validation	Test	Parameters
NMF	1.0714975	1.18537	epochs = 200, factors = 200
SVDpp	1.0168	1.23459	epochs = 200, factors = 200
Neural Network	1.007911	0.99635	layers=[200,128,64,16]
AutoRec	1.3462	1.39741	epochs=1, hidden neurons=100, batch size=100
Baseline	0.9996473	0.99768	method=ALS, epochs = 10
Random	1.48162	1.48097	method=ALS, epochs = 10
KNN	1.00912	1.22238	K = 40
Hybrid NMF	0.86663	0.97429	See Table I

Table II: Validation and Test rmse for different cf techniques

C. Comparison to Baselines

We evaluated our fine-tuned proposed approach against existing techniques discussed in Section III. Table II summarizes the results. The worst performance is given by the random sampling baseline that does not take into consideration the underlying structure of the ratings matrix at all. Neural network based models over-fit to the data very easily. We tuned the number of epochs to get the best performing value but were unable to further tune both approaches due to time and compute constraints. Individually, both neighbourhood and factorization based approaches do not perform very well. On the other hand, our hybrid model gives approximately 20% improvement in RMSE compared to performance of either model in isolation.

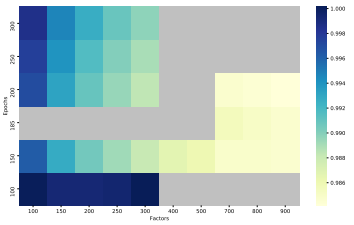

Figure 6: RMSE scores for epochs vs. factors of global NMF produced by grid search with 5 fold cross-validation

Epochs	Test RMSE
150	0.97639
175	0.97472
185	0.97429
200	0.97449

Table III: Test rmse for varying number of epochs of global nmf for 900 factors

VI. Discussion

Our proposed method achieves better accuracy than existing models and is efficiently parallelizable as the different pipelines are independent. As a result, we do not expect the training time to become a performance bottleneck. The approach considers latent factors at both global and local levels. Since this encompasses a broad set of hidden variables we expect it to generalize well to unseen data. We note that the usefulness of this method hinges on the assumption that the set of local factors has something to add to the set of global factors, in explaining the rating for a given (user, item) pair. In fact, we expect it to work best when these two sets are disjoint. In the event that one is a subset of the other, one of the pipelines may possibly become obsolete.

VII. Summary

In this work we have presented a novel ensemble based approach to collaborative filtering that uses non-negative matrix factorization, clustering and regression. We combine these methods to capture both global and local latent factors. By comparing our solution to several existing approaches we have shown that it performs significantly better on the given dataset. We tried different blending methods and found cross-validated, elastic net regularized regression to give the lowest test RMSE. Further work can include investigating other matrix factorization and clustering approaches e.g. soft clustering to encompass a wider range of factors.

REFERENCES

[1] N. Hug, "Surprise, a Python library for recommender systems," http://surpriselib.com, 2017.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[3] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[4] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, Y. Li, B. Liu, and S. Sarawagi, Eds. ACM, 2008, pp. 426–434. [Online]. Available: https://doi.org/10.1145/1401890.1401944

[5] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.

[6] J. Howard *et al.*, "fastai," https://github.com/fastai/fastai, 2018.

[7] J. Howard and S. Gugger, "Fastai: A layered api for deep learning," *Information*, vol. 11, p. 108, 02 2020.

[8] 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*, ser. WWW '15 Companion. New York, NY, USA: Association for Computing Machinery, 2015, pp. 111–112. [Online]. Available: https://doi.org/10.1145/2740908.2742726

[9] C. Chen, D. Li, Q. Lv, J. Yan, S. M. Chu, and L. Shang, "Mpma: Mixture probabilistic matrix approximation for collaborative filtering," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016, p. 1382–1388.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.