Augmented Matrix Factorization

Véronique Kaufmann, Liine Kasak, Martin Kučera, Aleksandar Milojevic
Kaggle-Group: Mystery AG
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In the midst of rising interest for targeted product recommendations, the demand for highly accurate rating predictors keeps growing. One of the more famous approaches to train such a predictor is collaborative filtering. After surveying several known collaborative filtering models on a given dataset, this paper presents a novel approach. *Augmented Matrix Factorization* combines singular value thresholding with generalized matrix factorization to exploit the advantages of simple, straightforward matrix factorization models. The proposed method is able to augment the results of optimized singular-value-decomposition-based matrix factorization models and hence manages to outperform all of the baseline approaches that were surveyed.**

I. Introduction

In today's world of fast-paced online consumerism, huge online retailers and streaming services are ubiquitous. To keep the customer engaged with their offerings, interaction-oriented websites often use so-called recommender systems to recommend enticing products to the customer. The goal usually is to predict how each user would rate unseen items. Subsequently, items with high rating predictions can be advertised to the respective users.

Depending on the underlying approach, these predictions can be based on different kinds of available information, such as user meta-data, per-user item interactions, or the totality of all available user-item interactions. Collaborative filtering is an approach that uses the last option. It encapsulates all user-item interactions as a ratings matrix $\mathbf{R}$, whose element in the i-th row and j-th column represents the rating made by the i-th user for the j-th item. For a large database, $\mathbf{R}$ will typically only have a small number of observed entries. Thus, the goal of collaborative filtering comes down to approximating a reconstruction of the fully observed matrix $\mathbf{R}$.

Empirically, the latent information which models the user-item interactions lives in a low-dimensional space. Therefore, it is sufficient to reconstruct a low-rank approximation of the matrix $\mathbf{R}$. A popular way to approach this low-rank reconstruction is to use matrix factorization methods.

Running various baseline approaches on our data resulted in the best performance for Singular Value Decomposition (SVD) oriented models, as seen in [1] and [2]. *Augmented Matrix Factorization*, leverages the results of an SVD-based matrix factorization model by both generating initialization values and learning weights for the latent feature embeddings. The simplicity of the proposed model makes it easy to understand and interpret. Section II explains various methods that aim at solving the collaborative filtering task, some readily available, some resulting from our own development process. Our final model is then introduced in Section III. Section IV and Section V present experiments and highlight the advantages of our final model over the surveyed approaches.

II. Models and Methods

In the following paragraphs, we examine various collaborative filtering methods that we consider as baseline approaches to estimate and report the performance of our final method in Section IV. Some of them are suggestions from recommended literature, and others are attempts at creating our own custom approaches.

A. SVD Based Matrix Factorization

Because of its ability to conveniently perform low-rank approximations, SVD plays a fundamental role in collaborative filtering [2]. This can be further related to matrix factorization and SVD via the Eckart-Young Theorem [3], according to which SVD can be used to generate an optimal rank-k approximation of a fully observed matrix. The matrix is decomposed into its SVD components $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$, and all singular values except for the first k are set to zero. The low-rank approximation matrix can then be obtained by combining the manipulated SVD components for matrix construction.

With $\mathbf{U}\mathbf{D}\mathbf{V}^\top$ being the singular value decomposition of the ratings matrix $\mathbf{R}$, we can view the first k columns of $\mathbf{U}\sqrt{\mathbf{D}}$ as latent feature vectors for the users, and the first k columns of $\mathbf{V}\sqrt{\mathbf{D}}$ for the movies respectively. Consequently, the prediction of a rating $\hat{r}_{ui}$ of a user u for an item i can be seen as the dot-product of two vectors containing the latent k-dimensional representation of the respective user and item. Thus, the rating can be predicted as $\hat{r}_{ui} = p_u \cdot q_i$, where p_u, q_i are the latent vector representations of the user and item.

Biases. Furthermore, biases b_i, b_u can be added to the model to account for a users preference (e.g. a user might always predict a higher rating than the average user) [4]. Doing this, the rating of user u for movie i is computed as

$$\hat{r}_{ui} = p_u \cdot q_i + b_u + b_i. \tag{1}$$

Optimization Methods. It can be shown that in the case of fully observed matrices, the SVD based approach described above leads to optimal results. In reality however, we often deal with partially observed matrices. Since the problem of finding an optimal approximation of the low-rank factorization is generally non-convex [4], we aim at solving it through optimization algorithms by minimizing a regularized loss function of the form:

$$\frac{1}{2}\sum_{ij\in R}(r_{ui}-(p_u\cdot q_i+b_u+b_i))^2+ \\ \lambda(\|p_u\|^2+\|q_i\|^2+b_u^2+b_i^2) \quad (2)$$

where r_{ui} is the actual observed rating and λ acts as regularization parameter. Two approaches used to minimize Equation 2 are *Alternating Least Squares (ALS)* and *Stochastic Gradient Descent (SGD)*. The first method optimizes fully over one parameter subspace while keeping the other parameters fixed, and the second one optimizes through small updates to all parameters at once.

B. K Nearest Neighbours (KNN)

In addition to working with matrix factorization, we explored other means of extracting information from the underlying data. Since we expect users with a similar latent representation to have similar opinions about the same items, we perform a nearest neighbour search on the user representations obtained through the optimized SVD-based approach and use it as input for a KNN model. For a new user, we then find the k closest users $N_k(u)$ to user u and predict the rating of user u for item i as

$$\hat{r}_{ui}=\frac{1}{k}\sum_{u'\in N_k(u)} r_{u'i} \quad (3)$$

C. Multinomial Logistic Regression

Another way of augmenting the result of the SVD-based approach is to look at the problem as a multi-class classification problem and use the optimized latent vectors as input for a multi-class classifier. Provided input pairs of the form $(user_u, movie_i)$, the task consists of predicting for each rating r the probability of user u giving movie i rating r. Since the SVD decomposition seems to encapsulate most of the useful information about users and movies, using them as input features seems appropriate. This is done using both the movie and user embeddings, as well as the biases. After performing feature extraction, we feed all the features into a Gradient Boosting model to perform the classification. The final rating is composed using the probabilities for each of the rating-values:

$$\hat{r}_{ui}=\sum_{r=1}^{5} p_i(u,i,r)\cdot r \quad (4)$$

where $p(u,i,r)$ represents the probability of user u giving item i the rating r (as computed by the classifier).

D. Autoencoder

An alternative approach for collaborative filtering is to use autoencoders. These reconstruct dense rating vectors from sparse input rating vectors through a hidden, low-dimensional representation. As opposed to the linearity of matrix factorization models, autoencoders can learn complex nonlinear relations. Multiple variants of autoencoders exist, which are commonly used for collaborative filtering. Denoising autoencoders (DAEs) corrupt ratings by random drop-out noise [5]. Deep autoencoders have many hidden layers, compared to the standard autoencoders with one encode and decode layer [6]. Variational autoencoders (VAEs) consist of a generative model as the encoder, which encodes ratings as a probability distribution, and a variational approximation as the decoder [7].

As autoencoders take the entire ratings matrix as the input data, a key problem is initializing unknown values. The performance of these models can be significantly improved when initializing them with results from matrix factorization, such as SVD.

E. Neural Collaborative Filtering

In most models based on matrix factorization, a prediction of a rating for a particular user-item pair usually corresponds to the dot product of the respective latent representations. Neural collaborative filtering (NCF) [8] suggests replacing the dot product with a neural network architecture which is, in theory, able to learn an arbitrary function [9]. Thus, the goal of NCF is to find a more suitable function to process the latent representations. The model consists of two parts: Multi-Layer Perceptron (MLP) which is a standard deep neural network, and Generalized Matrix Factorization (GMF), a weighted dot-product of the latent representations. If no better function exists, the model should just learn to apply the dot-product, which corresponds to a unit-weight GMF. Thus, the model is expected to always perform at least as well as the standard dot-product approach.

F. CF-NADE

Another neural model that has performed well on various datasets is CF-NADE [10]. It is heavily inspired by the Restricted Boltzmann Machine model [11], and the Neural Autoregressive Distribution Estimator (NADE), which is used for unsupervised distribution and density estimation [12].

G. Singular Value Thresholding (SVT)

Since the SVD-based algorithm seems sensitive to initialization, we investigate more elaborate initialization methods than simply initializing randomly or with the low-rank approximation. One way is to use the decomposed result of the SVT algorithm, as seen in [13].

The basic idea of the SVT algorithm is to relax the underlying non-convex problem of minimal rank approximation with the convex-envelope of this problem, namely nuclear norm minimization. The nuclear norm is defined as the sum of its singular values [14] and thus, minimizing the nuclear norm of a matrix will result in a sparse matrix with regards to its singular values. As we have seen in Section II-A, sparseness with respect to singular values results in low-rank approximations. The algorithm is a descent algorithm that at each iteration uses a *shrinkage operator* that *shrinks* the nuclear norm of the result matrix of the previous iteration and uses it to update the descent direction as follows:

$$\mathbf{A}^{t+1} = \mathbf{A}^t + \eta_t \Pi_\Omega(\mathbf{A} - \textit{shrink}_\tau(\mathbf{A}^t)) \tag{5}$$

where η_t is the step size and Π_Ω a projection to the space of observed entries. The shrinkage operator $\textit{shrink}_\tau$ decomposes the input matrix into its SVD components, goes over its singular values and either subtracts the *shrinkage value* $\tau < \infty$ from each singular value, or sets the singular value to zero, if the value is smaller than the shrinkage value τ:

$$\text{shrink}_\tau(\mathbf{A}) := \mathbf{U}\text{diag}(\sigma_i - \tau)_+\mathbf{V} \tag{6}$$

where the subscript + signifies the clamping of the values to zero. The converged result will be exactly reproducing the observed entries, which cannot be guaranteed for the non-relaxed problem.

III. Augmented Matrix Factorization

To boost the root-mean-squared error (RMSE) of SVD-based matrix factorization while maintaining a reasonable running time, we propose *Augmented Matrix Factorization* (AuMF). This model uses both SVT and GMF as a means to increase the performance of SVD-based matrix factorization. In particular, the decomposition of an SVT-generated matrix is used as initialization for the user and item vectors, that are then optimized with stochastic gradient descent using the best performing matrix factorization model. The output of the matrix factorization is then used as input for the GMF. AuMF allows keeping the underlying matrix factorization model simple while still providing a means to yield better results. The model consists of three phases.

Phase 1: Generating the Initialization Matrix. The first phase uses the SVT algorithm to generate a low-rank approximation of the data matrix. SVT reduces the rank implicitly by applying the shrinkage operator mentioned in Section II-G. For an appropriately high shrinkage value ($\sim 100'000$), the resulting approximated matrix will have a rank of around one-fifth of the original rank. This matrix will be used to initialize phase 2.

Phase 2: Matrix Factorization. The second phase is used to generate optimized matrix-factorization parameters according to Equation 2. In our case, the best working approach is to use distinct optimization methods for the two categories of parameters. First, the bias terms b_u and b_i are initialized with zero values, and alternating least squares is used to optimize towards a local optimum for those parameters. Second, the result matrix of phase 1 is decomposed and its decomposition is used to initialize both user and item embedding vectors p_u and q_i with a fixed number of features k. Then, by setting the optimized bias vectors as fixed values, the algorithm uses stochastic gradient descent to optimize only over the embedding vectors. The results are the learned embedding matrices $\mathbf{P}$ and $\mathbf{Q}$, as well as the bias vectors b_u and b_i, both of which are used as input for the next phase. We call this matrix factorization approach an SVD-based matrix factorization using hybrid optimization.

Phase 3: Weighted Dot Product. To predict the rating of user u for movie i we use the embeddings p_u and q_i obtained by matrix factorization. Instead of the regular dot-product we compute a weighted dot-product, and and the biases learned in the previous phase:

$$\hat{r}_{ui} = \sum_{j=1}^{k} w_j p_{uj} q_{ij} + b_u + b_i, \tag{7}$$

where $(w_1, \ldots, w_k)$ are the weights of individual latent features obtained from the matrix factorization. The weights are learned using stochastic gradient descent. This can also be viewed as linear regression on the element-wise products of the user-movie embedding pairs.

IV. Results

A. Experimental Setup

Scoring. As a measure, we use the root-mean-squared error (RMSE). The reported scores are obtained performing 5-fold cross-validation, reporting the mean of all five runs.

Dataset. The provided dataset contains 1000 movies and 10000 users. It contains tuples of the form (user, item, rating). All other information about the items and users is unknown. The corresponding data matrix (users x movies) has a sparsity of $\sim 12\%$. Further, we note that the dataset is highly imbalanced. Figure 1 shows the distribution of ratings $\{1, 2, 3, 4, 5\}$ contained in the training data.

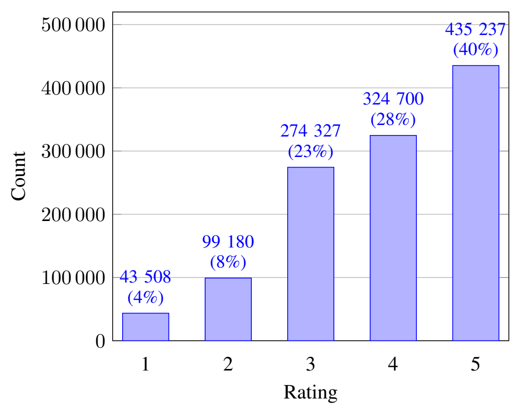

Figure 1. Number of entries per rating and percentage of all ratings.

Baselines. All algorithms used to compare our obtained score to are explained in Section II, and their implementation can be found in the project repository.

B. *Results*

Table I shows that Augmented Matrix Factorization outperforms all other approaches presented in this paper. This is particularly remarkable, considering the simplicity of the approach compared to much more elaborate methods such as CF-NADE. Further, it can be seen that our approach manages to achieve better performance than all its individual components separately.

Method	Average RMSE
Plain GMF	1.1688
Plain MLP	1.1562
NCF	1.1474
CF-NADE	1.0958
Variational Autoencoder	1.0919
Plain SVD	1.0712
Plain SVT	1.0389
KNN	1.0290
SVD-based MF using SGD	0.9842
Logistic Regression	0.9831
SVD-based MF using hybrid optimization	0.9831
SVT intialized SVD-based MF with hybrid opt.	0.9817
Augmented Matrix Factorization	**0.9805**

Table I
RESULTS OBTAINED APPLYING DIFFERENT COLLABORATIVE FILTERING METHODS TO THE DATASET PROVIDED.

V. DISCUSSION

In Section IV we can see that adding more complexity to our matrix-factorization model resulted in a decrease in performance. Thus, we were not able to overcome a certain local optimum by adding more complexity but had to keep the underlying model simple. Augmented Matrix Factorization improves the performance of a simple matrix factorization model through manipulating the input and output of the model rather than adding complexity to it. Thus, it allows the matrix-factorization model to stay as simple as needed while still being able to boost its performance. If phase 1 is not pre-trained, a weakness of the model in its current state is its running time. For the SVT algorithm to generate an effective result, a minimum of 2000 iterations is necessary. The proposed model is able to overcome the initial issues we had with the score not improving by adding complexity, as well as leverage the results of SVD-based matrix factorization.

Figure 2 shows the proportionate distribution of predictions of AuMF for each ground-truth label. Predictions were generated from 5-fold cross-validation. One can see that the model has quite a strong bias towards the mean-value. However, since for a user with little data available, predicting close to the mean-rating for a movie is usually sufficient. Comparing this to Figure 1, it can be seen that labels with a higher frequency in the trainset are predicted with greater confidence. This could be improved by looking more closely at the data distribution, which would need further investigation.

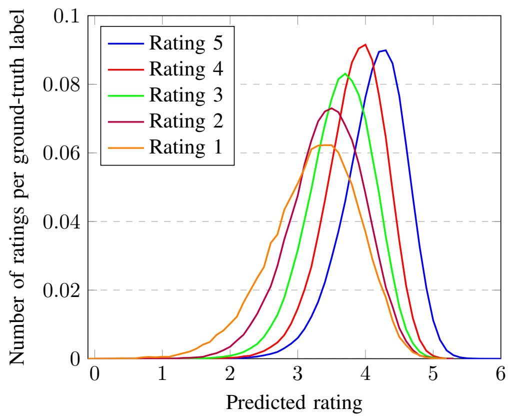

Figure 2. Distribution of AuMF predicted ratings for each ground-truth label

VI. SUMMARY AND FUTURE WORK

Applying collaborative filtering on the given data set, we have discovered that, for the given data set, it is more effective to augment the performance of a matrix factorization model by manipulating its input and output, rather than to increase its complexity. Augmented Matrix Factorization makes use of this insight by computing an intermediate low-rank approximation of the data matrix through SVT, which is used as an input to the matrix factorization model. The output of the matrix factorization model is then used to calculate weights for the embedding vectors. This way the core factorization model can be kept simple, while still boosting the performance of the model. The proposed approach, while straightforward to implement, has a lot of potential for further improvements. The SVT algorithm relies on the number of observed entries of the initial data matrix. If some of the entries of the initial matrix could be imputed correctly with high probability, the prediction score might improve. Additionally, the SVT algorithm's running time could be cut down significantly by optimizing the SVD subroutine. This way, the model would not rely as much on pre-training and could be run from scratch if needed. Furthermore, using more powerful neural network approaches to calculate the feature weights in the third phase would add another improvement of performance. All of these propositions would naturally need further investigation.

The proposed Augmented Matrix Factorization model keeps all of its promises of boosting the results of a simple matrix factorization approach and still has a lot of potential for further improvement.

References

[1] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *Neural Information Processing Systems(NIPS08)*, pp. 1257–1264, 01 2008.

[2] S. Funk, "Netflix update: Try this at home," 2006, https://sifter.org/ simon/journal/20061211.html, accessed: 06/2021.

[3] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," vol. 1, no. 3, pp. 211–218. [Online]. Available: https://doi.org/10.1007/BF02288367

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

[5] Y. Wu, C. DuBois, A. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," 02 2016, pp. 153–162.

[6] O. Kuchaiev and B. Ginsburg, "Training deep autoencoders for collaborative filtering," 08 2017.

[7] D. Kingma and M. Welling, "Auto-encoding variational bayes," 12 2014.

[8] X. He, L. Liao, and H. Zhang, "Neural collaborative filtering," *Proceedings of the 26th International Conference on World Wide Web*, 08 2017.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[10] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering," 2016.

[11] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," vol. 227, 01 2007, pp. 791–798.

[12] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator." *Journal of Machine Learning Research - Proceedings Track*, vol. 15, pp. 29–37, 01 2011.

[13] J.-F. Cai, E. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, 2010.

[14] B. Recht, M. Fazel, and P. Parillo, "Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization," *SIAM J. Optim.*, vol. 52, no. 3, pp. 471–501, 2010.