Collaborative Filtering

Yuqing Huang, Slobodan Jenko, Hei Yi Mak, Joao Mendonca (Group: Computing Intelligently)
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Collaborative filtering is a technique used in recommendation systems to analyze user-item interaction data and find similarities between users and items. It does not rely on item attributes or user characteristics, as content-based approaches do. Collaborative filtering is widely employed in various applications, such as suggesting movies, products, or music to users based on their past preferences and behaviour.**

This paper presents a comprehensive review of collaborative filtering methods for building recommendation systems. Bayesian Factorization Machines (BFM) outperform other methods significantly. Ensembling predictions of models with similar performances proves beneficial. Further, modifications to ALS show promising results. The incorporation of bias parameters in the ALS factorization model improves performance directly. Additionally, we show promising results for the use of deep sigmoid factorization if able to be paired with ALS optimization. Including the item count of the user in the feature vector enhances generalization for BFM.

I. Introduction

A recommender system is a class of machine learning that provides personalized suggestions for users, based on their past behaviours. There are two main approaches for building a recommender system: content-based and collaborative filtering. The content-based approach requires extensive information about items and suffers from a lack of diversity in predictions, while the latter approach makes predictions based on the user's preferences towards other items as well as other users' preference data. In real-life scenarios, users provide preference information only to a small subset of the available items. Therefore, sparse data is one of the important challenges faced by collaborative filtering approaches. In this paper, we study different classes of collaborative filtering methods, develop a novel solution that achieves competitive prediction accuracy, and compare the novel solution against baselines and other state-of-the-art approaches.

II. Models and Methods

A. Problem Statement

In this problem setting, we consider a large number of users and relevant items. They form a matrix where rows correspond to users, columns correspond to items, and each entry $x_{u,i}$ is the rating given by user u on item i. Given a set of sparsely observed entries, the goal is to develop accurate and efficient for predicting unobserved entries.

B. Singular Value Decomposition

Singular Value Decomposition (SVD) decomposes matrix A in $\mathbb{R}^{n\times m}$ into orthogonal matrices U in $\mathbb{R}^{n\times n}$ and V in $\mathbb{R}^{m\times m}$, along with a diagonal matrix Σ in $\mathbb{R}^{n\times m}$. The matrix Σ contains the singular values $\sigma_1, \sigma_2, \ldots, \sigma_{\min\{n,m\}}$, where $\sigma_i \geq \sigma_{i+1}$ for all i.

The Eckart-Young theorem highlights the importance of SVD in matrix approximation, showing that for each k between 1 and $\min\{n, m\}$, we can use SVD to obtain the best rank-k approximation of matrix A with respect to both the Frobenius norm and the spectral norm:

$$A_k = U_k \Sigma_k V_k^\top$$

- U_k is an orthogonal matrix in $\mathbb{R}^{n\times k}$.
- Σ_k is a diagonal matrix in $\mathbb{R}^{k\times k}$ containing the k largest singular values of A in decreasing order.
- V_k is an orthogonal matrix in $\mathbb{R}^{m\times k}$.

Retaining only the first k singular values and their corresponding singular vectors provides a compact representation of the original matrix while preserving its essential features. By reducing the dimensionality of the data, it uncovers latent factors that represent user preferences and item characteristics.

The method used here for SVD involves filling in missing matrix entries with the column average (average movie rating). However, this approach assumes complete knowledge and can adversely affect the matrix's singular values, making it unsuitable for direct computation of low-rank approximations for incomplete matrix representations.

Nevertheless, SVD remains a fundamental tool underlying various matrix approximation algorithms, three of which we have implemented: SVD with implicit feedback (SVD++), Singular Value Projection (SVP), and Nuclear Norm Relaxation (NNR). SVD++ [1] [2] incorporates implicit feedback based on user-item interactions, leveraging the fact that any user rating implies a level of interest regardless of the actual rating value. Additionally, it includes biases for users and movies, along with regularization. On the other hand, SVP combines SVD with gradient descent, using non-convex projection to achieve a k-rank approximation in the matrix. Finally, NNR applies convex relaxation to matrix completion, maintaining a sparse iterate sequence instead of a strictly k-rank matrix.

C. Alternating Least Square

The Alternating Least Squares (ALS) algorithm is an optimization algorithm relying on the existence of separable least squares problems that iteratively monotonically improves the objective and converges to a fixed point by fully optimizing half of the parameter space in each iteration, making it a compelling alternative to gradient descent methods.

Our base implementation of ALS utilizes a factored parameterization model via U in $\mathbb{R}^{n\times k}$ and V in $\mathbb{R}^{k\times m}$ such that $A \approx UV$ and an approximation error described by the objective:

$$l(U,V) = \frac{1}{2}\|\Pi_\Omega(A-UV)\|_F^2 + \lambda_u\|U\|_F^2 + \lambda_v\|V\|_F^2,$$

where $\lambda_u, \lambda_v > 0$, and Π_Ω zeroes out entries not observed in the original matrix A. If we treat U (V) as a fixed matrix, we can optimize optimally for each v_j (u_i) separately, as it is just a least squares problem:

$$\boldsymbol{v_j^*} = \left(\sum_i \omega_{ij}\boldsymbol{u_i}\boldsymbol{u_i}^\top + 2\lambda_v I)^{-1}\right)\left(\sum_i \omega_{ij}a_{ij}\boldsymbol{u_i}\right)$$

where ω_{ij} is equal to 1 iff entry $(A)_{ij}$ is observed in the original matrix, else 0.

1) Sigmoid Factorization: One of the downsides of the factored parameterization model is the mismatch between the domain of $(A)_{ij} \in [1,5]$ and the domain of the predictions $(\hat{A})_{ij} \in \mathbb{R}$. In the standard factored model, each prediction $(\hat{A})_{ij}$ is given by the scalar product of the learned user and item embedding vectors. Even though we know user ratings fall in the 1 to 5 range, the prediction given by $\sum_{i\le k} u_i v_i$ are potentially unbounded. This issue can be alleviated by defining the prediction of an element as

$$1 + 4 * \sigma(\langle \boldsymbol{u_i}, \boldsymbol{v_j}\rangle)$$

, where σ is the standard sigmoid function used in logistic regression. We dub this method ALSsig.

This model can't be optimized using the Alternating Least Squares algorithm because of the non-linearity introduced by the sigmoid function. We instead use the gradient-based Adam optimizer to learn U and V. In section III-C2 we compare this model to the original factored model.

2) Bias Incorporation: The original ALS algorithm factorizes the interaction matrix $A \approx UV$, but it lacks the ability to capture the individual characteristics of users and items. In our updated model (ALSb), we aim to overcome this limitation by expressing the rating of user i for movie j as $a_{ij} \approx \mu + b_i + b_j + \langle \boldsymbol{u_i}, \boldsymbol{v_j}\rangle$. Here, μ represents the mean rating of movies in matrix A, b_i is a bias parameter specific to user i, and b_j is a bias parameter specific to movie j. This enhancement enables the model to consider the inherent differences in how users rate movies and how movies are rated. For example, a user i known for providing very negative reviews will have a negative b_i value to account for this behavior. This way, the updated model can better represent user preferences and item characteristics, leading to more accurate recommendations.

Our updated model is implemented by introducing regularization factors for user biases (λ_{bu}) and movie biases (λ_{bv}). To accommodate these biases in the main weight matrices, we extend matrix U by adding 2 columns: one column to capture the user biases and another column filled with 1's. Similarly, we modify matrix V by adding 2 rows: one row to accommodate the movie biases and another row filled with 1's. This allows the biases to be seamlessly integrated while maintaining the mathematical structure of the matrix factorization.

3) Deep Sigmoid Factorization: The factored model imposes certain constraints on the predictions that might not accurately reflect reality. For example for $n = m = 2$, $k = 1$ we have $(\hat{A})_{11}(\hat{A})_{12} = (\hat{A})_{21}(\hat{A})_{22}$. A natural question, therefore, is whether we can modify the model in a way that allows it to better reflect the true relationships in the dataset. We have attempted to do this by applying a parametrized monotone function to the outputs of the factored model:

$$(\hat{A})_{ij} = 1 + 4f_L \circ \dots f_1(\sigma(\langle \boldsymbol{u_i}, \boldsymbol{v_j}\rangle))$$

, where $f_l(x) = \sigma(b_l + w_l x)$. This is a generic approach, inspired by the success of deep neural networks.

Our experiments show that the modified model struggles to achieve better validation results than the ALSsig, even though it typically reaches lower training loss. We conclude that other pieces are required to achieve results better than ALSsig with this method. Because of the negative results, we don't include this method in the following sections.

D. Bayesian Factorization Machine

Factorization machines are a generic supervised learning approach that accounts for feature interactions with factorized parameters. [3] In the matrix completion problem, a feature vector $\boldsymbol{x} = (I_u, I_i)$ is created for each user-item pair, where I_u represents an indicator variable for the active user and I_i is an indicator variable for the active item.

Given a feature vector $x \in \mathbb{R}$, the model for a factorization machine of degree $d = 2$ is formulated as follows.

$$\hat{y}(\boldsymbol{x}) \coloneqq w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} w_{ij}x_i x_j$$

The model parameters consist of a global bias w_0, i-th variable strength w_i, and interaction parameters w_{ij}. For Bayesian Factorization Machines, the Bayesian variable selection (BVS) method was proposed to capture the heredity between feature interactions. [4] The model is then optimized using alternating gradient descent. [3] [5]

E. Ensemble

To combine the power of the trained models, we use linear regression to learn a weight vector that aggregates the predictions of the models. This technique is known as linear blending [6]. Given a set of M models, training data $\mathcal{D}$, and the predictions $\hat{\boldsymbol{y}}_i^{train} \in \mathbb{R}^{|\mathcal{D}|}$ by each model i, the ensemble applies linear regression to find the optimal weights $\boldsymbol{w}^* = (w_1^*, w_2^*, \ldots, w_M^*) \in \mathbb{R}^M$ and bias $b^* \in \mathbb{R}$ that minimizes the least square loss,

$$\ell(\boldsymbol{w}, b) = \left\| \left(\sum_{i=1}^{M} w_i \hat{\boldsymbol{y}}_i^{train} + b \cdot \mathbf{1} \right) - \boldsymbol{y} \right\|_2^2$$

where $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{D}|}$ are the true ratings and $\mathbf{1}$ is a vector of ones. To perform inference on unseen data $\mathcal{D}'$, the ensemble output $\hat{\boldsymbol{y}} \in \mathbb{R}^{|\mathcal{D}'|}$ is an affine function of the models' predictions $\boldsymbol{y}_i$'s according to $\boldsymbol{w}^*$ and b^*,

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{M} w_i^* \boldsymbol{y}_i + b^* \cdot \mathbf{1}.$$

Since the models' predictions and the ensemble output should all be nonnegative, it may be helpful to consider the nonnegative linear regression problem where $w_i \geq 0$ for all $i \in [M]$. We can also disable the bias by setting $b = 0$.

III. Experiment Results

Data set: The data set used in the experiments contains 10000 users and 1000 items, with 11.77% of the matrix entries observed. Each entry is a rating of integer value between 1 and 5. The test set contains another 11.77% of the matrix that is unobserved.

Evaluation Metric: The collaborative filtering algorithms are evaluated on root-mean-squared error (RMSE).

A. Performance of Models

We performed 5-fold cross-validated grid-search to optimize the parameters of the collaborative filtering methods introduced in section II. We then train a model for each method with its best parameters using the whole training dataset. We also implemented the Kaggle baseline as introduced in the project tutorial.

The optimal parameters of the models are reported in table I. The validation and test RMSEs (public scores) of the models are summarized in table II. The best validation RMSE and test RMSE are achieved by BFM which are below 0.98. Most SVD-based methods have slightly higher errors than the baseline. ALS methods are able to score better than the baseline in the validation and test sets.

B. Comparison of Ensemble against Baselines

Our final solution is obtained by combining the predictions of the models in table I (excluding Baseline) using the ensembling method introduced in section II-E with nonnegative weights constraint and $b = 0$. The test RMSE is improved to 0.97375 which is significantly better than the baseline, SVD methods, and ALS methods, and is slightly better than BFM alone. We found that enabling bias does not help and allowing negative weights can severely hinder performance. It is also worth noting that the ensemble assigns large weight (≥ 0.95) on BFM and small weights (≤ 0.05) on other models, meaning that the ensemble's predictions are dominated by the predictions from BFM. If we exclude BFM, the ensemble achieves a test RMSE of 0.97898, which is a more significant improvement over the second-best model, ALSb. The weights are also distributed more evenly, allowing models to contribute more to the ensemble predictions. This suggests that our ensembling method is more suitable for combining weak models with similar performances.

Model	Parameters
SVD	k=3, iters=150
SVD++	k=3, iters=100, $\eta = .005$, $\lambda = .04$
SVP	k=3, iters=50, $\eta = 2.0$
ALS	k=5, iters=60, $\lambda_u = 13$, $\lambda_v = 17$
ALSsig	k=5, iters=1000, $\lambda_u = 0.3$, $\lambda_v = 0.3$
ALSb	k=5, iters=90, $\lambda_u = 23$, $\lambda_v = 37$, $\lambda_{bu} = 1$, $\lambda_{bv} = 100$
BFM	k=10, iters=300, sample=280
NNR	k=10, iters=120, $\tau = 600$, $\eta = 0.1$

Table I
Optimal parameters of the models.

Model	Validation RMSE ($\pm$ std.)	Test RMSE
Baseline	0.99239 ($\pm$0.00125)	0.98787
SVD	0.99379 ($\pm$0.00131)	0.98891
SVD++	0.99393 ($\pm$0.00100)	0.98931
SVP	0.99305 ($\pm$0.00145)	0.98830
ALS	0.99111 ($\pm$0.00160)	0.98542
ALSsig	0.99091 ($\pm$0.00141)	0.98627
ALSb	0.98604 ($\pm$0.00144)	0.98143
BFM	**0.97898** ($\pm$0.00081)	**0.97403**
NNR	0.99468 ($\pm$0.00119)	0.99096

Table II
Validation and test performance of the models.

C. Ablation Study

1) ALS + Bias (ALSb): In this experiment, we aim to fairly compare the performance of two matrix reconstruction methods, ALS and ALSb, by using their respective optimal parameters as listed in Table I. The comparison is done across different numbers of iterations to observe their evolution over time. The results in Table II confirm our expectations, showing that ALSb outperforms ALS. This improvement can be attributed to the incorporation of bias parameters, which enhances the expressiveness of the UV factorization model, despite its inherent simplicity. Notably, the introduction of a bias parameter for each movie and user leads to a significant 0.05 reduction in RMSE, making it our second most powerful model, following BFM. Furthermore, Table III demonstrates that ALSb consistently outperforms

the original ALS at all training stages and converges after a similar number of iterations. This indicates that the computational complexity of ALSb is not significantly higher, yet it achieves significantly better performance than the original ALS method.

Iterations	ALS (± std.)	ALSb (± std.)
15	0.9872 (± 0.0009)	0.9918 (± 0.0013)
30	0.9864 (± 0.0019)	0.9917 (± 0.0010)
45	0.9861 (± 0.0015)	0.9912 (± 0.0016)
60	0.9860 (± 0.0015)	0.9912 (± 0.0016)
75	0.9860 (± 0.0014)	0.9911 (± 0.0016)
90	0.9860 (± 0.0015)	0.9913 (± 0.0015)

Table III
COMPARISON OF THE ORIGINAL ALS AND ALSB TRAINED ON THEIR OPTIMAL PARAMETERS FOR A DIFFERENT NUMBER OF ITERATIONS. RMSE CALCULATED BY CROSS VALIDATION.

2) ALSsig: In this experiment, for a fair comparison, we use Adam to train both the standard and sigmoid factored models. We use the same fixed k and the number of iterations for both models but find separate λ_u and λ_v for each model using grid search. The experiment shows that the sigmoid model outperforms the standard one by a large margin when coupled with the Adam optimizer. It is also interesting to note that ALS finds a much better model than Adam for the standard model (see Table II), indicating the effectiveness of ALS compared to the more generic gradient-descent-based approaches.

k	iters	original RMSE (± std.)	sigmoid RMSE (± std.)
3	700	1.0062 ± 0.0012	**0.994** ($\pm$0.0014)
5	1000	1.0046 ± 0.002	**0.991** ($\pm$0.0014)
7	700	1.008 ± 0.0019	**0.992** ($\pm$0.0013)

Table IV
COMPARISON OF THE ORIGINAL FACTORED MODEL AND SIGMOID FACTORED MODEL. BOTH MODELS ARE TRAINED BY THE ADAM OPTIMIZER. RMSE CALCULATED BY CROSS VALIDATION.

3) BFM: For the Bayesian Factorization Machine method, we perform an ablation study on how different ranks and feature constructions impact the prediction accuracy. The RMSE errors are reported on both validation and test sets in table V. First, we observe that models with a too-low rank cannot capture well the complex patterns in data and models with a too-high rank have less generalizability. Then, we explored different feature creation methods. The standard feature consists of one-hot encodings of user-ID and item-ID. An extension is to add the observed item count per user to the standard feature. The new feature is expressed as $\boldsymbol{x} = (I_u, I_i, c_u)$, where c_u is the number of observed entries for user u. Table V shows that adding the observed count to the feature vector gives slight improvement to the prediction accuracy in all experiments.

Rank	Feature	Validation RMSE (± std.)	Test RMSE
k=5	one-hot	0.98236 ($\pm$0.00081)	0.97863
k=10	one-hot	0.98024 ($\pm$0.00104)	0.97513
k=15	one-hot	0.98149 ($\pm$0.00125)	0.97594
k=5	one-hot & count	0.98086 ($\pm$0.00089)	0.97752
k=10	one-hot & count	**0.97898** ($\pm$0.00081)	**0.97403**
k=15	one-hot & count	0.97958 ($\pm$0.00087)	0.97433

Table V
VALIDATION AND TEST PERFORMANCE FOR VARIOUS BFM MODELS AND FEATURE SELECTIONS.

IV. CONCLUSION

Collaborative filtering methods are used to build recommendation systems by finding similarities between users and items using user-item interaction data, without relying on item attributes or user characteristics as content-based approaches do.

In our report, we conduct a comprehensive review of various collaborative filtering methods, namely: SVD, SVD++, SVP, ALS, ALSsig, ALSb, BFM, and NNR. Our findings reveal that Bayesian Factorization Machines (BFM) outperform the other methods significantly. However, we also present promising results for simpler methods with modifications on ALS performed by our group.

Ensembling the predictions of different models proves to be beneficial, especially when the models have similar performances. The linear regression ensembling method is simple to implement, requires only the predictions of the models (no retraining of individual models), and generates predictions quickly. However, it may suffer from assigning too much weight to a single model if it significantly outperforms others. Thus, having models with similar performance is crucial for effective ensembling.

Furthermore, our ablation studies on ALS demonstrate the potential of enhancing relatively simple methods to achieve more competitive performances. ALSb increases the expressiveness of the model via bias incorporation while attaining a closed-form solution, allowing the use of ALS optimization and, thus, better results. Encouragingly, our ALSsig experiments suggest that a closed-form solution for ALSsig that enables ALS as the optimization algorithm (instead of ADAM) may further boost the model's performance, especially when combined with the bias incorporation of ALSb.

Lastly, our ablation study on BFM indicates that including the item count of the user in the feature vector improves generalization to unseen data.

REFERENCES

[1] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008.

[2] N. Hug, "Matrix factorization-based algorithms." [Online]. Available: https://surprise.readthedocs.io/en/stable/matrix_factorization.html

[3] Y. Chen, Y. Wang, P. Ren, M. Wang, and M. de Rijke, "Bayesian feature interaction selection for factorization machines," *Artificial Intelligence*, vol. 302, p. 103589, 2022.

[4] S. Rendle, "Factorization machines," *2010 IEEE International Conference on Data Mining*, pp. 995–1000, 2010.

[5] T. Ohtsuki, "myfm - bayesian factorization machines in python/c++." [Online]. Available: https://myfm.readthedocs.io/en/stable/

[6] A. Töscher and M. Jahrer, "The bigchaos solution to the netflix grand prize," *Netflix prize documentation (2009)*, pp. 1–52, 2009.