Collaborative Filtering for Music Recommendation

Zander Meitus **Yiming Zhang**
University of Chicago
{zmeitus,yimingz0}@uchicago.edu

1 Introduction

In this project, we study music recommendation using tools we learned in the class and collaborative filtering (CF) methods. Collaborative filtering performs matrix completion by leveraging user-level and album-level similarities: for a set of n users and p items, the user-item matrix X takes the form of $X = \mathbb{R}^{n \times p}$, with potentially missing values when no rating is available. Then, the recommendation problem boils down to identifying a set of "good" items $\mathcal{I}_i$ for a user $i \in [n]$.

To study the problem of music recommendation, we create a new dataset **AOTY** as a testbed for music recommendation. We chose to experiment with two recommendation algorithms, EASE$^{\text{R}}$ [7] and UserKNN [4], for their state-of-the-art performance and computational tractability demonstrated by a recent survey [1]. After implementation, we evaluate their performance on **AOTY**. Both EASE$^{\text{R}}$ and UserKNN show strong recommendation performance: In a retrieval setting, EASE$^{\text{R}}$ achieves an F1@20 score of 27.0, an impressive score considering there are over 8k potential albums. In a classification setting, UserKNN achieves an F1 of 82.2 on heldout user-item predicitions. All implementations are from scratch, and the code is publicly available at `https://github.com/Y0mingZhang/music-recommendation`.

2 Literature Review

Collaborative filtering recommendation systems have been a major focus of machine learning for several decades, with applications in commerce, entertainment, professional networking and beyond [6, 2]. In recommendation settings, observations for a given user are often sparse, and the model is tasked with predicting user preferences without much training data for the particular user. There are a variety of model families that are well-suited to this task, including neighborhood-based, linear, matrix factorization, and neural models, among others [1]. Beyond computational complexity and performance on a given task, the model families differ in the types of tasks they are most adept at handling. For example, some algorthmic families are better at handling implicit feedback datasets (where a value of 1 indicates a user interaction and a 0 indicates no user interaction), while others handle explicit feedback (where a user gives an item a rating on an ordinal or continuous scale) [7].

As Schedl [5] illustrate, recommendation for music has unique differences from other recommendation tasks. Individual songs last minutes, while other common recommendation items like movies or books are consumed over a much longer time period. Songs are consumed sequentially, where the particular order is meaningful. A given music catalog may contains tens of millions of entries. Schedl [5] argue that because of music's unique attributes, it is worthwhile to study recomendation systems as applied to music in isolation.

Our study is unique from existing music recommendation systems work both in its application and the dataset. Our unit of recommendation is an album, while most commercial music recommendation systems operate at the song level [5]. This distinction is important in an era where music critics and fans bemoan how the "endless mix-tape" of song recommendation algorithms detracts from the experience of listening to an album from start to finish [9, 3]. The dataset we've compiled and curated from `aoty.org` provides explicit feedback from users at the album level, which enables us to research performance differences in models intended for explicit and implicit feedback.

3 Method

Collaborative Filtering, the technique of inferring the preferences of one user using known information about other users, is the dominant class of algorithm for recommendation systems. In this project, we plan to implement, extend and evaluate two CF algorithms, EASE^R and UserKNN, demonstrated as among the state-of-the-art in a comprehensive survey by Anelli et al. [1].

EASE^R. EASE^R [7] is a linear model parameterized by an item-item matrix $B \in \mathbb{R}^{p \times p}$. The weights B are optimized with respect to the simple objective

$$\min_B \|X - XB\|_F^2 + \lambda \cdot \|B\|_F^2. \tag{1}$$

Intuitively, EASE^R reconstructs user preferences using information about items exclusively. In addition, EASE^R adds a matrix norm penalty, encouraging B to be low-rank. B admits a degenerate solution $B = I$, under which the recommendation system always recommends items that the user likes. To avoid this solution, the author added an additional constraint that $\operatorname{diag}(B) = \mathbf{0}$. Learning the weights of B is a (constrained) convex optimization problem, and the optimal solution is given by

$$\hat{B}_{i,j} = \begin{cases} 0 & \text{if } i = j \\ -\frac{\hat{P}_{i,j}}{\hat{P}_{j,j}} & \text{otherwise,} \end{cases} \tag{2}$$

where $\hat{P} = (X^\mathsf{T} X + \lambda I)^{-1}$.

Despite its simplicity, computing this closed-form solution requires inverting a $p \times p$ matrix, taking $\mathcal{O}(p^3)$ time in a typical commercial implementation. To save computational cost, we experiment with mini-batch stochastic gradient descent as an alternative technique for computing $\hat{B}$. To compute the loss $\|X - X\hat{B}\|_F^2 + \lambda \cdot \|\hat{B}\|_F^2$, we need to multiply a sparse matrice X with B, which can be computed in $\mathcal{O}(xp)$ time, where x is the number non-zero elements in X. This alternative training procedure takes $\mathcal{O}(kxp)$ time, where k is the number of iterations until convergence.

UserKNN. Neighborhood-based recommendation methods essentially treat the preference of a user u as a weighted sum of preferences of other users under some notion of similarity. For a pair of users u, v, UserKNN [4] measures the similarity between users with the correlation coefficient r_{uv} between items that u and v both rated. The predicted score for an item is the weighted sum of ratings of similar users, after adjusting for the "baseline rating" that the user gives to an average item.

$$P = \vec{\mu} + \frac{(A - \bar{J})R}{M|R|} \tag{3}$$

It is important to note that in this setting, the n albums are the matrix rows, while the p users are the columns, which is the reverse of the EASE^R setup. P is prediction matrix of n albums and p users. μ is the average rating of each reviewer. A is the provided $n \times p$ ratings matrix. $\bar{J}$ is an $n \times p$ matrix where the i, j entry is the average rating of the jth reviewer, exluding the ith rating. R is the reviwer correlation matrix with its diagonal set to 0. M is a mask of A, where cells with ratings in A are 1 in M, and all other values of M are 0. The approach is non-parametric in that there is no loss function to optimize for and no parameters to tune. The algorithm simply aggregates known user data to produce predictions. However, the original paper only provides the equation for a single rating estimate and we had to extrapolate to the full matrix application.

4 Experimental Setup

Data curation. For this project, we have created a dataset of album reviews, named **AOTY**, by scraping data from `https://aoty.org`. On this website, users create profiles and rank albums from 0 to 100. After filtering out albums with < 100 total reviews and users with < 10 positive reviews, **AOTY** has a total of $8,855$ albums and $22,167$ users. The average album in the dataset has 426.8 reviews. The user-album matrix is very sparse: for **AOTY**, the sparsity $s = 1.9 \times 10^{-2}$. The *data sparsity* problem [8] in the user-item matrix is a major challenge in most real-world recommendation problems, and we believe this property makes **AOTY** a suitable testbed for studying recommendation.

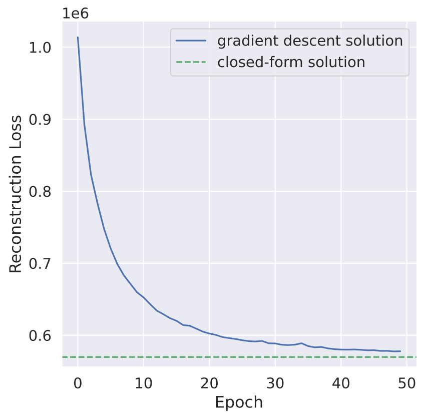
Evaluation setup and metrics. EASE^R assumes *implicit feedback*, where interactions are binary labels indicating whether a user likes an item or not. Therefore, we binarize the **AOTY** dataset by considering all reviews with rating ≥ 75 as the positive class, and all other reviews as the negative class. In a *strong generalization* setting (i.e., testing on heldout users in training), we measure the performance of EASE^R on three common retrieval metrics. Precision @ k measures the percentage of recommended items that a user likes in top-k recommendations, and recall @ k measures the coverage of all items liked by a user in top-k recommendations. F1 is defined as the harmonic mean of precision and recall. Beyond these common retrieval metrics, we further consider the following metrics following Anelli et al. [1]:

- *Item Coverage*: Item Coverage (IC) measures the percentage of items that is recommended to at least one user.
- *Gini*: Gini coefficient of the distribution of recommendation measures its inequality.

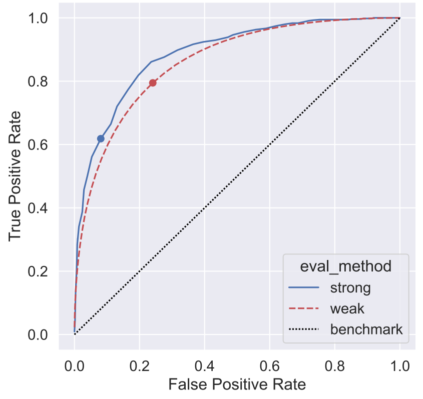
UserKNN takes *explicit feedback* (ratings) as input and generates explicit ratings as output, so we evaluate the model on mean absolute error (MAE) and root mean squared error (RMSE). We also convert the true and predicted ratings to implicit feedback following the same procedure as for EASE^R and evaluate on previously discussed retrieval metrics. Since evaluation with UserKNN is slow, we randomly selected 10 users in the test set for evaluation under the *strong generalization setting*. We further report performance in the *weak generalization* setting, where a random sample of 20% of ratings are omitted from the user-rating matrix X to create a user-rating matrix X'. The algorithm is then run on X' to generate predictions. The predictions for the omitted ratings are then compared to the true ratings to evaluate performance.

Both algorithms were created from scratch in Python3 using matrix functions in NumPy and pandas.

5 Results



(a) Convergence of stochastic gradient descent (blue) to the optimal, closed-form solution (green) in EASE^R.



(b) UserKNN performance on different implicit feedback thresholds. Dots represent evaluated threshold.

Figure 1

EASE^R. We evaluate the both the closed-form version of EASE^R and our proposed stochastic gradient descent (SGD) approach on a heldout out test set of users. We are interested in quantifying the recommendation quality of both algorithms, and understanding whether our proposed approach produces a solution that converges to the closed-form solution.

One way to understand the convergence behavior of the algorithm is by measuring the reconstruction loss $\|X - X\hat{B}\|_F^2$. In Figure 1a, we plot the reconstruction loss corresponding to $\hat{B}^{(t)}$, the learned weights, against the epoch number t. The green line corresponds to the reconstruction loss of the optimal closed-form solution. We find that the reconstruction loss of stochastic gradient descent indeed converges to that of the closed-form solution. This observation isn't surprising given that

Evaluation metrics @ 20	EASE^R (closed-form)	EASE^R (SGD)
Precision $\uparrow$	69.2	69.2
Recall $\uparrow$	21.0	21.0
F1 $\uparrow$	27.0	27.0
Item Coverage $\uparrow$	34.7%	34.7%
Gini $\downarrow$	0.925	0.925
Optimization time $\downarrow$	5 seconds	107 seconds

Table 1: Evaluation of top-20 recommendations of EASE^R on a heldout test set. $\uparrow$ means higher is better, and $\downarrow$ means lower is better.

EASE^R has a convex objective, and gradient descent provably converges for convex optimization problems. That said, SGD-based optimization takes 107 seconds in our experiment, 21x longer than that of computing the closed-form solution directly. We suspect that although our proposed solution has smaller big-$\mathcal{O}$ complexity on paper, it's not well-tuned compared NumPy's matrix inversion algorithm. In addition, unlike normal matrix multiplication, sparse matrix multiplication does not take advantage of memory locality, a hardware level optimization that makes sequential memory access fast.

EASE^R shows strong recommendation performance when evaluated on unseen users in training, highlighted by a Precision @ 20 score of 69.2, meaning users provided positive reviews for the majority of items we recommended. While the Recall @ 20 score is only 21.0, the cutoff of 20 items is less than the average number of reviews per user, making a perfect score unattainable. We also argue that precision is the more relevant metric in recommendation, as we care mostly about recommending some, rather than all, items that a user will like. We observe significant concentration in the recommendations of EASE^R, with roughly one-third of all albums recommended to at least one user. The Gini score of 0.925 highlights a strong inequality in the recommendation, with a small subset of albums being heavily favored over the rest.

Evaluation metrics	UserKNN- Weak Gen.	UserKNN- Strong Gen.
Number of Ratings	736,944	1,121
Precision $\uparrow$	85.1	85.0
Recall $\uparrow$	79.5	62.4
F1 $\uparrow$	82.2	72.0
AUC $\uparrow$	86.1	88.7
Mean Abs. Error $\downarrow$	8.55	8.17
RMSE $\downarrow$	11.8	10.7

Table 2: Evaluation of UserKNN on a heldout test set in weak generalization setting. $\uparrow$ means higher is better, and $\downarrow$ means lower is better.

UserKNN. We evaluate UserKNN in both a weak generalization and strong generalization setting, utilizing both the original explicit feedback ratings and engineered implicit feedback. The weak setting included 736,944 test predictions sampled randomly across all users and albums, while the strong setting involved just 10 randomly sampled users across all albums for a total of 1,121 test predictions.

The algorithm performs strongly in both the weak and strong generalization settings, with mean absolute error of 8.55 and 8.17 respectively. To evaluate the model's performance on implicit feedback data, all true ratings ≥ 75 were converted to 1, while all other ratings were converted to 0. Model predictions were also converted using the same threshold, though the ROC curve in figure 1b shows strong performance across different thresholds for model predictions with AUCs of 86.1 and 88.7 respectively. At the selected threshold, the model has comparable recall in the weak and strong settings at 85.1 and 85.0, but the model has better recall in the weak setting. This could be due to the small sample size in the strong setting or the threshold selection.

References

[1] V. W. Anelli, A. Bellogín, T. Di Noia, D. Jannach, and C. Pomo. Top-N Recommendation Algorithms: A Quest for the State-of-the-Art. In *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*, pages 121–131, July 2022. doi: 10.1145/3503252.3531292.

[2] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), dec 2016. ISSN 2158-656X. doi: 10.1145/2843948. URL `https://doi.org/10.1145/2843948`.

[3] R. Hilton. Do You Really Listen To Full Albums? `https://www.npr.org/sections/allsongs/2013/05/20/185534315/do-you-really-listen-to-full-albums`, 2013. [Online; accessed 2-December-2022].

[4] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, Oct. 1994. Association for Computing Machinery. ISBN 978-0-89791-689-9. doi: 10.1145/192844.192905.

[5] Z. H. C. C. Schedl, M. Current challenges and visions in music recommender systems research. *International Journal of Information Retrieval*, 7:95–116, 2018. ISSN 1687-7470. doi: 10.1007/s13735-018-0154-2.

[6] B. Smith and G. Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21(3):12–18, 2017. doi: 10.1109/MIC.2017.72.

[7] H. Steck. Embarrassingly Shallow Autoencoders for Sparse Data. In *The World Wide Web Conference*, WWW '19, pages 3251–3257, New York, NY, USA, May 2019. Association for Computing Machinery. ISBN 978-1-4503-6674-8. doi: 10.1145/3308558.3313710.

[8] X. Su and T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009:e421425, Oct. 2009. ISSN 1687-7470. doi: 10.1155/2009/421425.

[9] J. J. Toth. Too Much Music: A Failed Experiment In Dedicated Listening. `https://www.npr.org/sections/therecord/2018/01/16/578216674/too-much-music-a-failed-experiment-in-dedicated-listening`, 2018. [Online; accessed 2-December-2022].