Objective Functions for Collaborative Filtering

Krishan Prashanth **Colby Endres** **Caleb Lee** **Vidith Madhu**

Abstract

With the vast swathes of data available, an efficient recommendation system for entertainment platforms is necessary. Approaches based on collaborative filtering have been effective at using movie and user data to provide accurate predictions. One way of accomplishing this is via low-rank matrix approximations. We provide a classical implementation of matrix approximation and suggest several other objective functions for the prediction functions.

1 Introduction

In 2006, Netflix announced a $1 million competition for the best Movie Recommendation model for their users. This led to an influx in work and research for collaborative filtering techniques, which could estimate a user's preferences based on their prior ratings. Using these ratings, they can generate recommendations for new items the user would like. This method has been stretched to alternate domains, such as books, music, etc.
One particularly successful approach that was developed by Simon Funk is Matrix Factorization, which ended up winning the competition [8]. Matrix Factorization involves breaking down a sparse user-item rating matrix into two smaller matrices encoding latent factors regarding the respective users and movies using gradient descent. The benefits of doing so are twofold:

1. **Space Reduction:** The original user-rating matrices are usually extremely sparse, because most users only watch a fraction of all the available movies. Hence, if the sparse matrix can be represented as the product of two smaller, dense matrices then it reduces the computational difficulty of maintaining the user-item matrix as it can be recovered through matrix multiplication, if need be.
2. **Learned Embeddings:** Once these latent factors are learned for users and movies, it quantifies the user's interests and the movie's genre. This compressed representation of the user-ratings matrix provides an easy way to compute user's ratings for movies they are yet to watch.

In this paper, we will examine different Matrix Factorization methods.

2 Preliminaries

Suppose we have a matrix $R \in \mathbb{R}^{m \times n}$, where the entry r_{ij} represents the rating of movie j by viewer i. Naturally, this matrix will be missing large amounts of data, so we seek to predict these values. Our methodology relies on the assumption that there are k inherent attributes for both viewers and movies that determine a rating. We start off by initializing two low-dimensional matrices: movie matrix ($U \in \mathbb{R}^{m \times k}$) and user matrix ($V \in \mathbb{R}^{n \times k}$) with random values. We can then approximate $\tilde{R} = UV^T$, where our predicted rating $\tilde{r}_{ij} = u_i v_j^T$. This error is then used to calculate the gradient and update the embeddings accordingly to minimize the error. This process is repeated for all known ratings to train the model.

Preprint. Under review.

3 Fixed-Rank Methods

The simplest choice of objective function is:

$$f(U,V) = \frac{1}{2} \sum_{(i,j)\in\Omega} (r_{ij} - u_i v_j^T)^2 + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right)$$

where λ is a regularization parameter designed to prevent U and V from becoming too large. In the event that the entire ratings matrix R is known and regularization is ignored, an analytical solution can be easily obtained by rewriting the above as:

$$f(U,V) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} (r_{ij} - u_i v_j^T)^2 = \frac{1}{2} \left\| R - UV^T \right\|_F^2$$

It is well-known that the best rank k approximation to R is $\tilde{R} = U\Sigma_k V^T$, where Σ_k is the diagonal matrix of the first k singular values of R [1]. However, having a complete ratings matrix is unlikely in practice, so we turn to descent based approaches. To utilize stochastic gradient descent, we first hold the vector u_i as constant and denote Ω_{u_i} as the number of movies rated by user u_i. Then:

$$\begin{aligned}
f_u &= \frac{1}{2}\sum_{j\in\Omega}(r_{ij} - u_i v_j^T)^2 + \frac{\lambda}{2}\sum_{j=1}^{n} \|u_i\|_2^2 + \|v_j\|_2^2 \\
&= \frac{1}{2\cdot\Omega_{u_i}} \sum_{j\in\Omega} \Omega_{u_i}\left[(r_{ij} - u_i v_j^T)^2 + \frac{\lambda}{2}\left(\|u_i\|_2^2 + \|v_j\|_2^2\right)\right] \\
&= \frac{1}{2\cdot\Omega_{u_i}} \sum_{k=1}^{\Omega_{u_i}} f_{u_k}
\end{aligned}$$

Similarly, if v_j is fixed and Ω_{v_j} gives the number of users viewing movie v_j:

$$\begin{aligned}
f_v &= \frac{1}{2}\sum_{i\in\Omega}(r_{ij} - u_i v_j^T)^2 + \frac{\lambda}{2}\sum_{i=1}^{m} \|u_i\|_2^2 + \|v_j\|_2^2 \\
&= \frac{1}{2\cdot\Omega_{v_j}} \sum_{i\in\Omega} \Omega_{v_j}\left[(r_{ij} - u_i v_j^T)^2 + \frac{\lambda}{2}\left(\|u_i\|_2^2 + \|v_j\|_2^2\right)\right] \\
&= \frac{1}{2\cdot\Omega_{v_j}} \sum_{k=1}^{\Omega_{v_j}} f_{j_k}
\end{aligned}$$

A straightforward calculation reveals:

$$\frac{\partial f_{u_k}}{\partial u_i} = \Omega_{u_i}(-v_j(r_{ij} - u_i v_j^T) + \lambda u_i) \qquad \frac{\partial f_{v_k}}{\partial v_j} = \Omega_{v_j}(-u_i(r_{ij} - u_i v_j^T) + \lambda v_j)$$

This gives us our gradient update for both vectors:

$$\begin{aligned}
u_i^{(n+1)} &= u_i^{(n)} - \alpha \cdot \frac{\partial f_{u_k}}{\partial u_i} \\
v_j^{(n+1)} &= v_j^{(n)} - \alpha \cdot \frac{\partial f_{v_k}}{\partial v_j}
\end{aligned}$$

We also consider alternative choices of loss functions, other than the usual choice of mean-squared error. Namely:

$$f_{MAE}(U,V) = \sum_{(i,j)\in\Omega} |r_{ij} - u_i v_j^T| + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right)$$

which uses mean-absolute error. This incurs less of a penalty for large misses than mean-squared error. If we define $\frac{\partial}{\partial x}|x| = \text{sgn}(x)$, this objective function is differentiable and the above descent procedure can be used to calculate U and V.

4 Alternative Methods

The previous strategies limit the rank U and V to be no more than k. If we remove this restriction, we can encourage the matrix UV^T to be low-rank via the nuclear-norm objective:

$$f_*(U, V) = \sum_{(i,j)\in\Omega} (r_ij - u_i v_j^T)^2 + \frac{\lambda}{2} \left\| UV^T \right\|_*^2$$

This has the added benefit of being convex, guaranteeing a unique local minimizer. However, the nuclear norm is not differentiable and more sophisticated methods like semidefinite programming [6] or subgradient descent [5] must be used. We will make a crude approximation by replacing the nuclear norm with the sum of $\|u_i\|_1$ and $\|v_j\|_1$ in each gradient update. Penalizing the one-norm encourages the vector to be sparse, reducing the rank, and consequently the nuclear norm, of our approximation. Our final metric is cosine similarity, in which we normalize the dot product:

$$\cos(\theta) = \frac{uv^T}{\|u\|_2 \cdot \|v\|_2}$$

Note that this similarity score is bounded between $[-1, 1]$, unlike all previous metrics. This technique has been used frequently in scenarios where there is a notion of semantic similarity between vectors [7]. Since collaborative filtering assumes some notion of similarity between users and movies, this metric is appropriate.

5 Experimental Results

We begin by discussing our experimental setup, then present our results. We used Python to implement our filtering algorithms due to the availability of scientific computing packages such as NumPy and pandas. Our implementation is setup in a Jupyter notebook to enable rapid, shared prototyping and evaluation. Our code can be split up into two main components: a data extraction/construction phase and a training/evaluation phase. Our chosen dataset is the MovieLens dataset, which contains over 1 million user ratings of nearly 4000 movies [2]. We chose this dataset because it was developed by researchers at the University of Minnesota for the specific purpose of academic research in information and collaborative filtering.
We chose to split this dataset using a 70/30 split: 70% of the entries are used for training our model, and 30% are used for evaluation. We used several objective functions for training/evaluation: mean squared error, mean absolute error, nuclear norm, and cosine similarity. As detailed in the above section, our training implementation uses regularized gradient descent with respect to the chosen objective to learn a good low-rank factorization of the training set. Our evaluation uses a few different quality metrics depending on the objective; mean squared error and nuclear norm use root mean squared error (RMSE), while mean absolute error, and cosine similarity use mean error.

6 Related Work

6.1 Differential Privacy

One important concern with efforts to develop stronger collaborative filtering methods is user privacy. Since many of the applications relying on collaborative filtering techniques are user-facing platforms, careful handling of datasets is required to prevent leaking sensitive information about user identities. In fact, the Netflix Prize discussed in section 1 was suspended in 2010 due to an information leak. The dataset used for the competition was structured similarly to the MovieLens dataset used in this project, with anonymized user ratings. However, researchers were able to reconstruct user identities by comparing entries in the dataset with publicly available user ratings on the Internet Movie Database (IMDb), where user identities are known. This led to multiple lawsuits from Netflix customers and privacy concerns from the Federal Trade Commission, halting the competition.

Fortunately, significant research has been conducted on methods to prevent such information leakage. An emerging technique is differential privacy [4], where random noise is applied to a dataset to reduce the viability of reconstructing user identities while preserving the statistical results of the data. Hence, an interesting avenue of future work is at the intersection of differential privacy and collaborative filtering.

6.2 ML for Collaborative Filtering

Over recent years, machine learning has taken off due to the rise of algorithms well-suited to parallelization on GPU for training of models. Research has been done to enhance matrix factorization based collaborative filtering models through the usage of neural networks. Neural networks such as a multilayer perceptron may be used to capture nonlinear relationships between users and items to improve accuracy of recommendation. [3]

7 Conclusion and Discussion

Our results show that for a sparse dataset of user-item interactions such as the MovieLens dataset, mean squared error with nuclear norm performs best with the least error. However, the nuclear norm to mean squared error did not seem to have a significant difference in comparison to regular mean squared error for this particular dataset.

Table 1: Error Evaluation

Objective Function	Mean Squared Error
Mean Squared Error	0.9917985402899018
Nuclear Norm	0.9944815883517955
Objective Function	Mean Error
Mean Absolute Error	0.8533531583990308
Cosine Similarity	0.11242773451387633

Below are plots for mean errors over training iterations for each objective function.

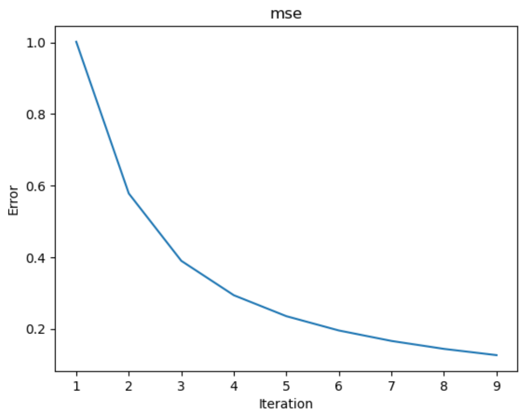

Figure 1: Mean Squared Error

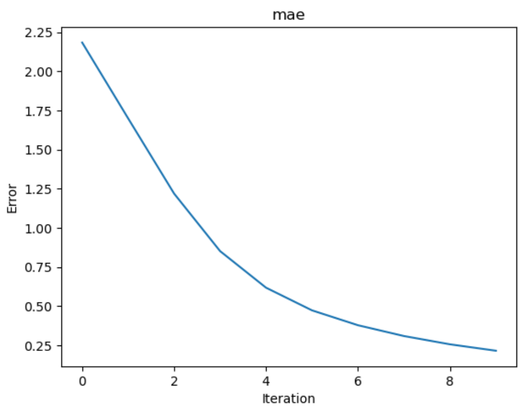

Figure 2: Mean Absolute Error

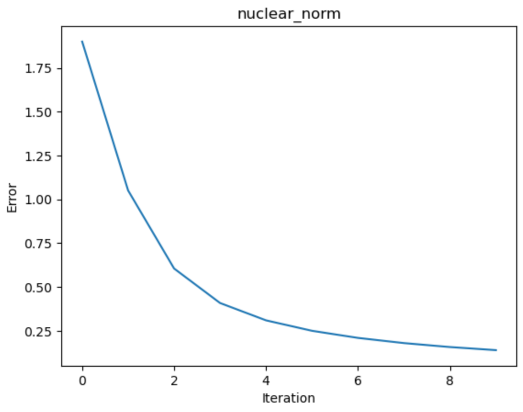

Figure 3: Nuclear Norm Error

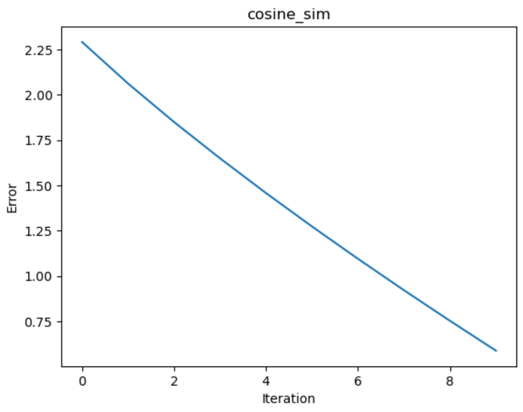

Figure 4: Cosine Similarity Error

We also tested differing numbers of latent features using the mean squared error objective function, and found it to perform best with 14 latent features.

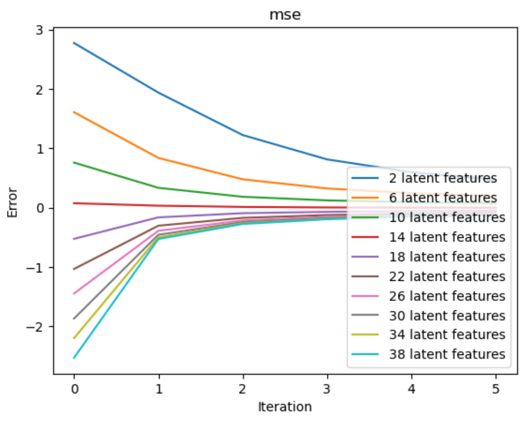

Figure 5: Mean Squared Error vs Number of Latent Features

The current approach targets user-item interactions, but future work would explore item-item or user-user interactions on more datasets.

References

[1] C Eckart and G Young. The approximation of one matrix by another of lower rank. 1936.

[2] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. 2017.

[4] Michael Hilton and Cal. Differential privacy : A historical survey. 2012.

[5] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 457–464. Association for Computing Machinery, 2009.

[6] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, January 2010.

[7] Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. Is cosine-similarity of embeddings really about similarity? *arXiv preprint arXiv:2403.05440*, 2024.

[8] B Webb. Netflix update: Try this at home. 2006.