MS VALDOM - Machine Learning and Optimization Project

Nirina Andriamino[1]
[1]INP Toulouse ENSEEIHT

Februrary 2025

Abstract

Recommender systems play a crucial role in various industries, particularly in entertainment and e-commerce. This project focuses on collaborative filtering to predict missing movie ratings by factorizing the user-movie rating matrix into two lower-rank matrices. We implement Stochastic Gradient Descent (SGD) optimization with both a fixed and an adaptive learning rate determined via a line search. Our results demonstrate that adaptive learning significantly improves convergence speed and predictive accuracy. By tuning model parameters, we achieve a lower RMSE than the baseline BIAS model. This study highlights the effectiveness of matrix factorization in recommendation systems and the impact of optimization techniques on performance.

Contents

1 Introduction

Recommender systems have become an essential tool in various industries, especially in entertainment, e-commerce, or online streaming platforms. One of the most widely used techniques in recommendation systems is collaborative filtering. Collaborative filtering is an information retrieval method that recommends items to users based on how other users with similar preferences and behavior have interacted with that item.

In this project, we focus on a collaborative filtering problem where the main goal is to predict missing movie ratings given a partially observed user-movie matrix. Generally, most users rate only a small fraction of available movies, leaving a significant portion of the rating matrix unknown. Accurately filling in missing values would allow for making reliable movie recommendations.

To address this problem, we will factorize the theoretical matrix into two matrices. The main idea is to approximate the theoretical rating matrix by the product of two lower-rank matrices. We will use gradient descent to solve the optimization problem and regularized mean squared error (RMSE) as a loss function. The dataset used in this project comes from MovieLens, a well-known benchmark dataset for evaluating recommendation algorithms.

This report provides a detailed overview of our methodology, including data preprocessing, the optimization algorithm, implementation details, and an analysis of the results.

2 Dataset and preprocessing

As mentioned earlier, we will use the MovieLens dataset, a widely recognized benchmark for evaluating recommendation algorithms. Each rating is a score ranging from 0 to 5. The dataset consists of 610 users and 9,724 movies, with a total of 100,836 recorded ratings. Since the complete user-movie rating matrix would theoretically contain 5,931,640 values, approximately 98.3% of the entries are missing, highlighting the sparsity of the dataset.
We split the data into two equally sized partitions, and each partition was further divided into a training and a test set. Each training set contains 99.4% of the data from its respective partition. Following this, each split dataset was converted into an RDD to facilitate subsequent computations.

3 Methodology

Let's focus on the methodology, we will first define two matrices: P and Q.

$$\begin{aligned} P \in \mathbb{R}^{n\times k}: &\quad \text{This is the user feature matrix where n is the number of users and k is the number of feature} \\ Q \in \mathbb{R}^{m\times k}: &\quad \text{This is the movie features matrix where m is the number of movies and k is the same number of feature} \end{aligned} \tag{1}$$

By multiplying these matrices together, we can obtain an approximation of the theoritial rating matrix. The obtimization problem can be written as:

$$\min_{P,Q\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}} \quad \frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}\left(r_{i,j}-p_i^T q_j\right)^2+\lambda\left(\sum_{i=1}^{n}\|p_i\|^2+\sum_{j=1}^{m}\|q_j\|^2\right) \tag{2}$$

To solve the optimization problem, we will use gradient descent, so we need the gradients of the loss function with respect to both matrices. The gradient descent updates can be derived by performing the following computations:

$$\frac{\partial J}{\partial p_i} = -2\sum_{j:(i,j)\in\Omega}\left(r_{i,j}-p_i^T q_j\right)q_j + 2\lambda p_i \tag{3}$$

$$\frac{\partial J}{\partial q_j} = -2\sum_{i:(i,j)\in\Omega}\left(r_{i,j}-p_i^T q_j\right)p_i + 2\lambda q_j \tag{4}$$

We will compare an SGD method with a constant learning rate and an SGD method with a learning rate chosen through a line search. To conduct the line search we will need the two Wolf conditions:

$$\begin{cases} L(x_k+\alpha_k d_k) \le L(x_k) + c_1\alpha_k\nabla L(x_k)^T d_k \\ \\ \nabla L(x_k+\alpha_k d_k)^T d_k \ge c_2\nabla L(x_k)^T d_k \end{cases} \tag{5}$$

With $d_k = -(\nabla L_Q^k(x^k) + \nabla L_P^k(x^k))$.

For the scoring, we will perform a 2-folds cross validation with the RMSE.

4 Implementation

Given the large amount of data, we need to be cautious about how we compute the loss function and gradients. The computation time could become significant because we will repeatedly call the function that computes both the gradients and the loss. Transforming the data into RDDs is useful because it allows us to process all the data efficiently in parallel.

5 Results

First, we trained a simple Stochastic Gradient Descent (SGD) model with a constant learning rate. The initialization of the two matrices played a crucial role in the training process. If the initial values were too far from the theoretical rating matrix, the algorithm struggled to converge and, in some cases, failed entirely.

To improve convergence, I initialized the P matrix with values drawn uniformly from the range $[0.45, 0.55]$, ensuring a narrow spread around a central value. Similarly, the Q matrix was initialized with values uniformly distributed in the range $[0.2, 0.8]$ to introduce more variance. For the learning rate, I chose 0.4. If the learning rate was too high, the algorithm was more likely to diverge, whereas if it was too small, the algorithm did not converge fast enough.

With this setup, the final overall loss value reached 2.747, indicating the performance of the model.

We then tried to run an SGD with a learning rate chosen using a line search. The initialization parameters for the SGD were the same as before. For the parameters of the line search, I chose $(\beta_1, \beta_2) = (0.1, 0.97)$. This means that I allow weaker changes in the loss function value and small reductions in the gradient norm at each step.

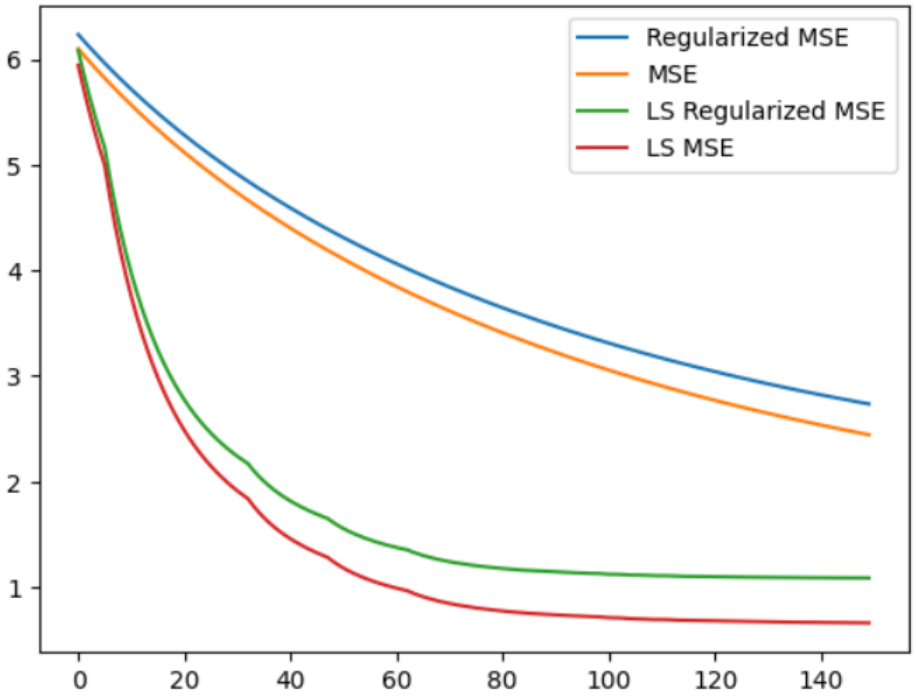

Figure 1: Loss evolution though iterations for the training set of the first fold

Figure 1 shows the evolution of the MSE and the regularized MSE for the regular SGD and the SGD with a line search. The first observation is that the regularized MSE is always above the regular MSE, which is expected due to the regularization term.
Figure 1 illustrates that, in this case, using a line search to adapt the learning rate significantly improves the performance of our model. Specifically, the convergence is much faster with the line search, allowing us to reach lower loss values more quickly compared to when no line search is used.

Evaluation	**Value**
Cross Validation Training	1.5671
Cross Validation Test	2.0284
Cross Validation Training (Adaptive Learning Rate)	0.8138
Cross Validation Test (Adaptive Learning Rate)	0.9536

Table 1: Cross-validation results for training and testing with and without adaptive learning rate

The table 1 presents the RMSE in 2 folds cross validation for the training and the testing set.As we said earlier, we obtain significantly better result with a line search on both the training and the testing set.

Now we will compare our performance with the BIAS model. The RMSE in cross-validation obtained with this model is equal to 0.9467. Let's see if we can obtain better performance by tuning the number of features of the two factorizing matrices and the penalty applied to the regularization of the MSE. I decided to let each matrix have 15 columns. This means that users and movies have more characteristics to represent them. Additionally, I relaxed the penalty on the regularization term to 0.0025. The cross-validation RMSE obtained was 0.93794, which is slightly better than with the BIAS model.

6 Conclusion

In this project, we explored a collaborative filtering approach for movie recommendation by factorizing the user-movie rating matrix into two lower-rank matrices. Using gradient descent optimization, we aimed to minimize the regularized mean squared error (RMSE) to achieve accurate rating predictions.

Our experiments compared two Stochastic Gradient Descent (SGD) methods: one with a fixed learning rate and another with an adaptive learning rate determined via a line search. The results demonstrated that incorporating a line search significantly improved convergence speed and final performance, allowing the model to reach lower loss values faster.

Furthermore, we evaluated our matrix factorization model against the baseline BIAS model. By tuning the number of latent features and adjusting the regularization penalty, we achieved an RMSE of 0.93794 in cross-validation, outperforming the BIAS model's RMSE of 0.9467. This indicates that our factorization approach, with optimized hyperparameters, can provide better recommendations.

Overall, this study highlights the potential of matrix factorization for recommendation systems and the impact of optimization choices on predictive performance.