Matrix factorization for movie recommendations

Ilmari Vahteristo

March 27, 2023

Abstract

This is a report for "Mathematics for machine learning" course. In this project, the assignment was to implement three different matrix factorization algorithms with Numpy, based on the Small movie lens dataset, and see how they perform on the task of predicting movie ratings. The algorithms implemented were:

- Singular Value Decomposition (SVD)
- Alternating Least Squares (ALS)
- Gradient Descent (GD)

1 Introduction

In this project, the goal was to create a recommendation system for movies, based on a dataset containing users and their rated movies. The dataset is available at: https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset.

The data consists of 100,836 ratings of 9,742 movies by 610 users. The data was read to a Numpy matrix, where the rows correspond to users and the columns to movies.

All the data was read to a matrix R, resulting in a matrix of size $(610, 9742)$, where users are on the rows and movies are on the columns. The index (i, j)

corresponds to the rating user i gave to movie j, or 0 if the user hasn't rated the movie.

The directed approach for this problem is to perform a low rank approximation of the ratings matrix R, such that $R = UM$, or the norm $||R - UM||$ is minimized.

The factorizations U and M can be thought of as the users and movies hidden features, respectively. The features capture the latent features of the users and movies, such as their preferences and genres, and can be used to predict the ratings of the users for the movies they haven't rated.

The low rank approximation can be performed using various optimization algorithms, such as Gradient Descent, Alternating Least Squares (ALS), and Singular Value Decomposition (SVD). These algorithms are used to minimize the difference between the predicted ratings and the actual ratings in the training set.

1.1 Singular value decomposition

Singular Value Decomposition (SVD) is a matrix factorization method that decomposes a matrix A (n, m) into three matrices:

$$A = U\Sigma V^T$$

where A is the matrix to be decomposed, U is an orthogonal matrix of size (n,n), Σ is a diagonal matrix of size $n \times m$ with non-negative real numbers on the main diagonal, and V^T is an orthogonal matrix of size $m \times m$.

The diagonal entries of Σ are known as the singular values of A, and the columns of U and V are known as the left and right singular vectors of A respectively.

SVD is used in many applications, including image compression, data analysis, and recommendation systems. In recommendation systems, SVD can be used to predict the ratings of items that a user has not yet rated, based on the

ratings of other users and items.

The rank k approximation of matrix A, is then calculated by multiplying k first columns from U (n,k), k first singular values from Σ (k,k) and k first rows from V^T (k,m).

1.2 Alternating least squares

ALS is an iterative algorithm that can be used to approximate a matrix R with two non-negative matrices U and M, such that $R \approx UM$.

The algorithm works by alternating between fixing U and solving for M, and fixing M and solving for U, until convergence.

ALS is commonly used in recommendation systems to predict the ratings of items that a user has not yet rated, based on the ratings of other users and items.

Pseudocode for the ALS algorithm:

Algorithm 1 Alternating Least Squares (ALS)

Initialize U (nusers,k) and M (k,nmovies) from $Unif(0, 1)$
while not converged **do**
    for each row i of U **do**

$$U_i = (MM^T)^{-1}(R_i M^T)^T$$

    end for
    for each column j of M **do**

$$U_j = (U^T U)^{-1} R_j U$$

    end for
end while

1.3 Gradient Descent

Gradient Descent is an optimization algorithm that is used to minimize a function by iteratively moving in the direction of steepest descent as defined by

the negative of the gradient. In the context of matrix factorization, Gradient Descent can be used to minimize the difference between the predicted ratings and the actual ratings in the training set.

The algorithm works by initializing the factor matrices U and M with small random values between 0 and 1, and then iteratively updating them using the following rules:

$$U_{t+1} = U_t + \alpha(R - U_t M_t)M_t^T$$

$$M_{t+1} = M_t + \alpha U_t^T(R - U_t M_t)$$

where α is a learning rate, which controls the size of the step.

1.4 Dealing with sparseness

Since the user-movie matrix R is usually very sparse (in this case, over 98% of the ratings were 0), some methods have a hard time giving good recommendations, especially if the rank k is high. To remedy this, each missing value in R was substituted by an estimate based on the global average rating of a movie, the users average rating, and the movies average rating.

$$E = G_m + (G_m - U_m) + (G_m - M_m) \tag{1}$$

where G_m is the global average rating of a movie, U_m is the users average rating of a movie, and M_m is the movies average rating.

2 Methodology

A Python script was written to perform and compare the different factorization algorithms. For each of the methods, 10% of non-zero values were randomly

removed from the matrix R and stored in a separate matrix R_t. The matrix R_t is then the matrix, where we can test how well a method has predicted a set of actual values.

After removing (marking as unwatched 0) the test values from the matrix R, all the missing values in R were filled with the estimate from equation 1.

A chosen matrix factorization was performed, and the reconstruction error $||R - UM||$ was measured with the frobenius norm.

However, the main metric of the 'goodness' of an approximation was Mean Absolute Error (MAE), due to its simplicity and interpretability; it tells you how far, on average, each prediction was from the actual value. The MAE was calculated by comparing each non-zero value in the test matrix R_t, with the corresponding index in the matrix $R' = UM$.

3 Results

All of the factorization methods did similarly well at their best parameters.

3.1 Singular value decomposition

With SVD, the lowest MAE was achieved with a rank 20 approximation, yielding an MAE of 0.67.

3.2 Alternating Least Squares

The best results achieved with ALS was an MSE 0.68, which is slightly higher than SVD. ALS was only tested for k values of 1 - 13, because there was no reason to believe higher values would've yielded better results. The best results came with a low rank approximation.

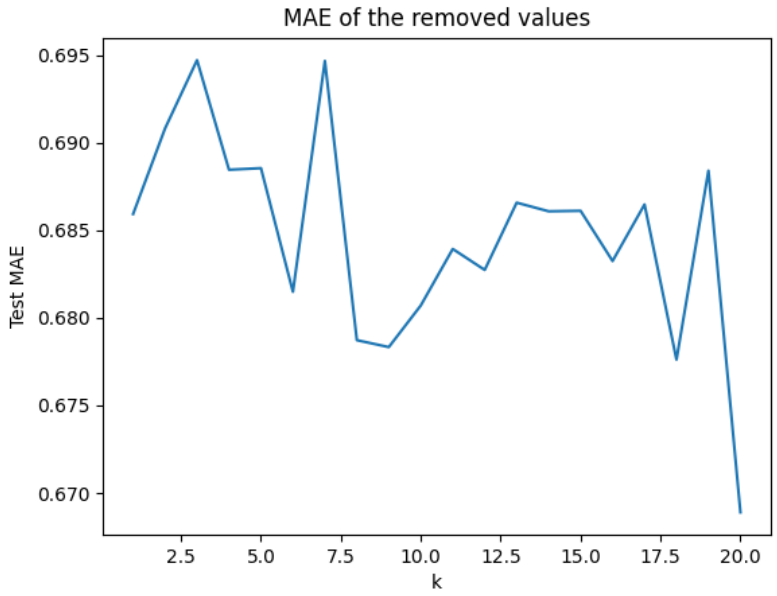

Figure 1: The MAE when using SVD for matrix factorization with rank ranging from 1 to 20.

3.3 Gradient Descent

Gradient descent was quite fast, since I wasn't using the stochastic version of it. However, the results were not any better than SVD. The lowest MAE of 0.68 was achieved with k = 8. Varying k from 1-19 produced roughly the same results each time, when using a learning rate α of $2 \times 10^{-}4$ and a maximum of 500 iterations.

3.4 Movies to features graph

As part of the assignment, I chose 30 movies from the dataset. I then performed one the factorization methods, in this case gradient descent, and did factorization for a rank 2 approximation. I then selected each movies index, and took its projection to the feature space of the movies receiving a 2 dimensional point.

I then plotted each point to a graph, where the first axis describes the value of the first hidden feature and the y axis the second hidden feature.

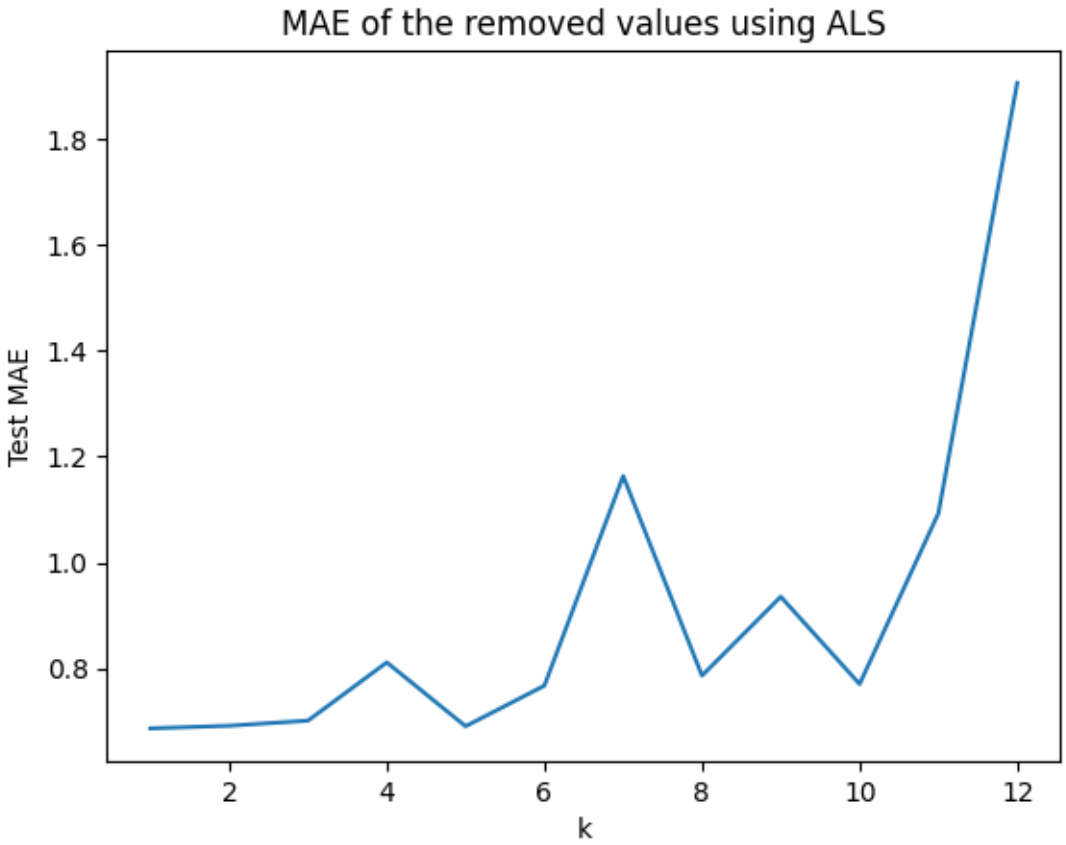

Figure 2: The MAE when using ALS for matrix factorization with rank ranging from 1 to 13.

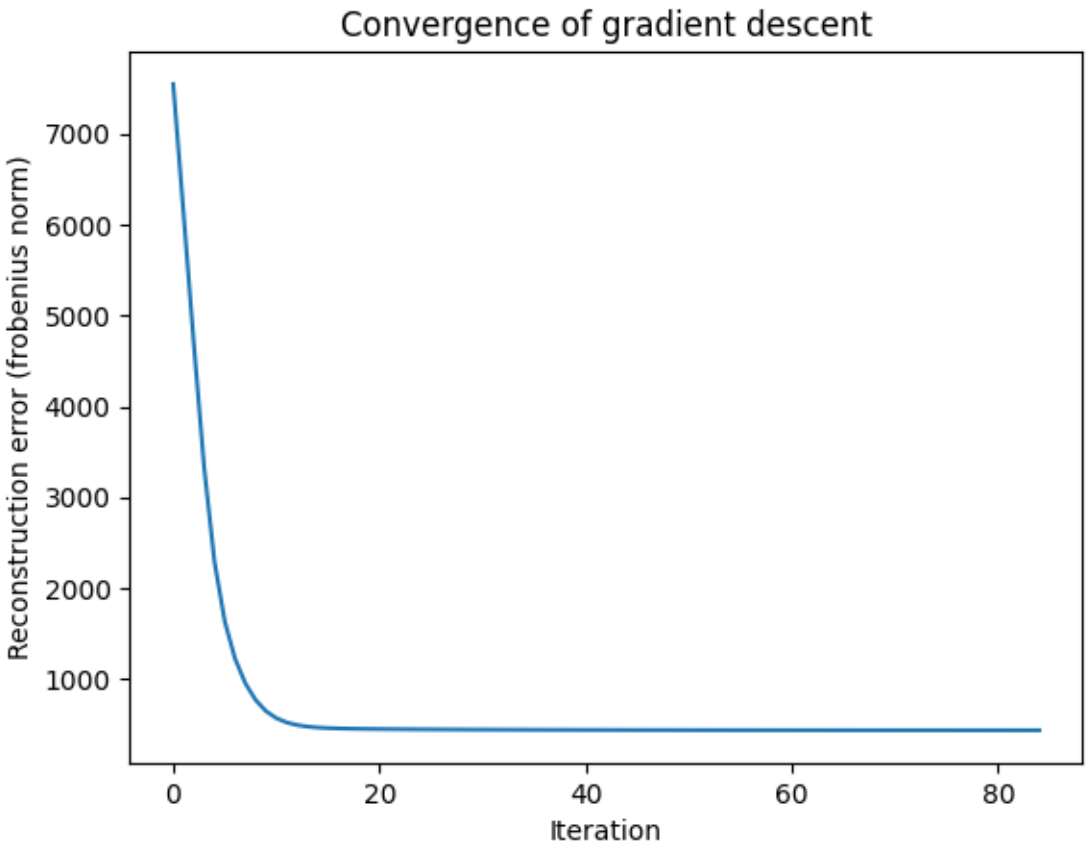

Figure 3: The convergence of gradient descent, with $k = 2$ and $\alpha = 2 * 10^{-}4$

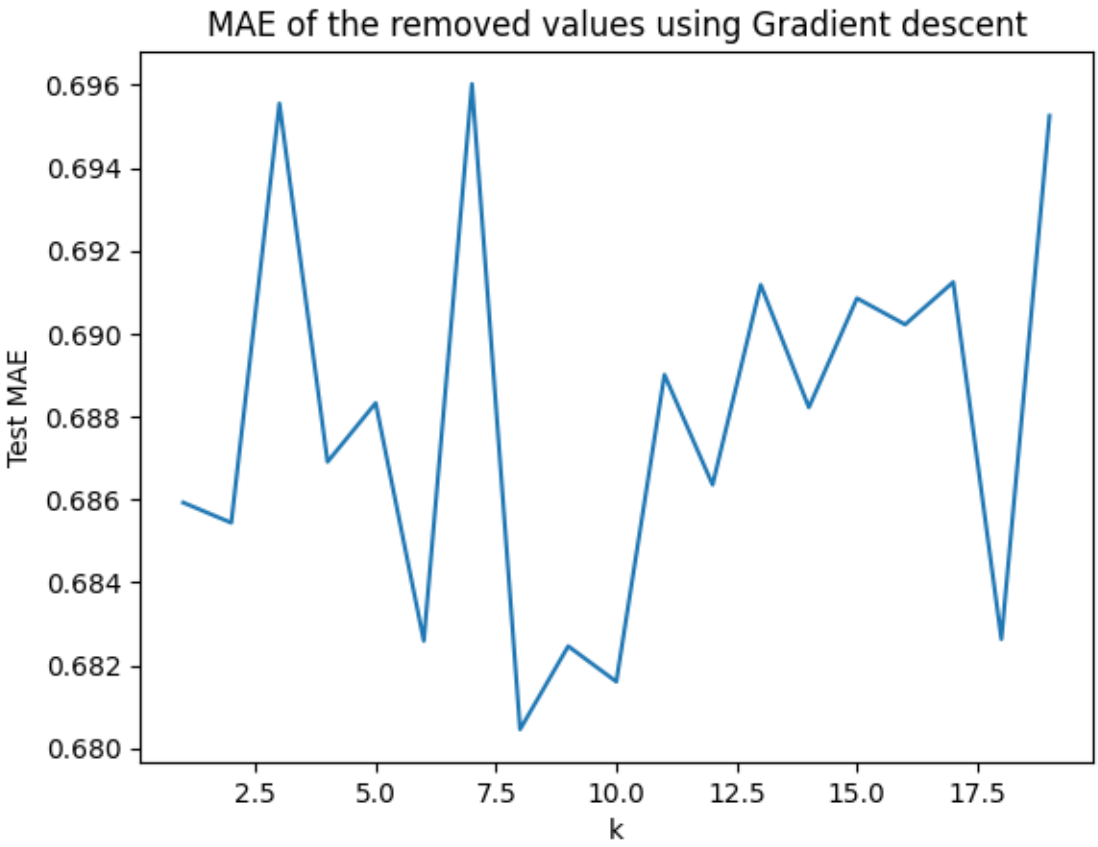

Figure 4: The MAE when using gradient descent for matrix factorization with rank ranging from 1 to 19.

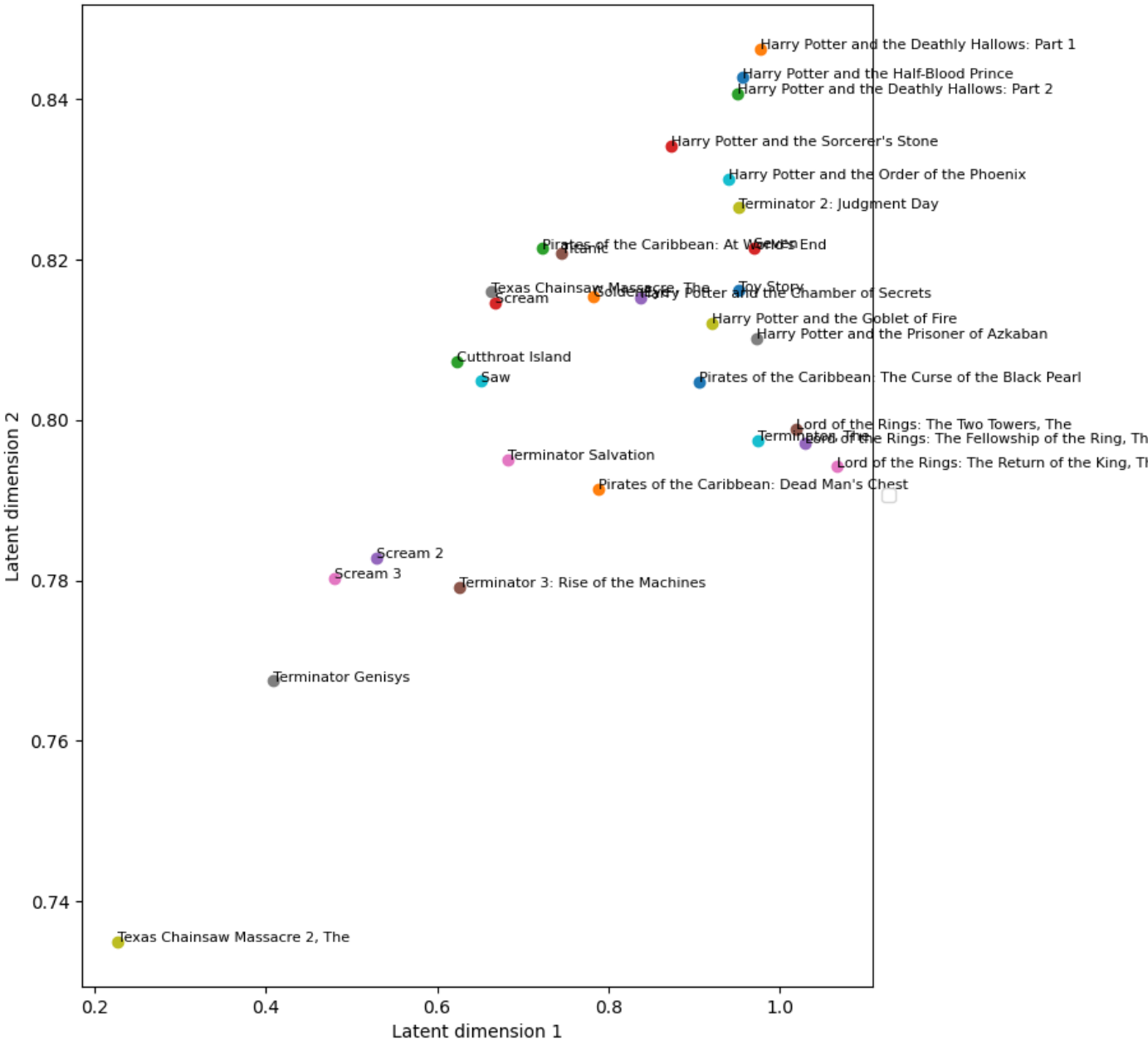

Figure 5: The hidden features for a rank 2 approximation for some known movies.

From figure 5 it isn't obvious what the latent features describe, but a reasonable quess could be, that the first feature corresponds a bit to popularity. For example the Lord Of The rings movies are on the right. Maybe it could also correspond to how good the movie is, given that Lord of the Rings is the best.

The second axis y, might correspond more to fantasy or something similar, given that the Harry Potter movies are high on it.

The mapping seems reasonable, since similar movies are mostly grouped, most notably the Harry Potters and Lord of the rings'.