Recommender System

Taj Gillin

April 24, 2025

1 Problem Description

The goal of this assignment is to implement a recommender system using matrix factorization. We are given a training set of user-movie ratings, comprising an incomplete matrix of user ratings for movies. Our goal is to predict the missing ratings in the matrix.

2 Approach

To solve this problem, I implemented matrix factorization with bias correction, optimizing with gradient descent. The core concept is that we can decompose the rating matrix R into two lower-rank matrices P and Q such that $R \approx P^T Q$. Fundementally, we are finding some latent representation space that relates users and movies. One possibel way to think of this is as "type" of user. That is, maybe some users like action movies, and these users will rate certain movies higher. Then, the goal is to represent each user and movie in this latent space, such that we can use this to predict ratings.

There are three main components to this:

- We are trying to learn the latent factor matrices. That is, for n users and m movies, we are trying to learn two matrices $P \in \mathbb{R}^{n \times k}$ and $Q \in \mathbb{R}^{m \times k}$ where k is the latent factor dimension.
- We additionally use a bias term globally, for each movie, and for each user. This helps account for some shift in tendency to rate movies higher or lower (e.g. some users are just more positive or negative, some movies are often rated higher or lower, etc).

- Our final prediction is then given by $\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$ for user u and movie i. We also clip predictions to be between 0.5 and 5.0.

3 Implementation

To optimize the parameters, we use SGD. This consists of a forward pass (prediction), error computation, and gradient descent update. I also use some regularization to try to prevent overfitting. In the end, the hyperparameters picked were:

- Rank $k = 200$
- Learning rate $\eta = 0.005$
- Regularization $\lambda = 0.02$
- Epochs $N = 60$

4 Results

Notably, we do not have the test set, so we are quite prone to overfitting. Regularization certainly helps, but before we get the final results, I can only write on the results of the training set and the 1% shown by the auto-grader. After training, I get a loss of 0.3167 on the training set and 0.7171 on the 1% of the test set.