Movie Recommendation System

Jacob Steimer

2023-10-27

Using machine learning techniques to recommend movies and predict ratings

Introduction

This report is meant to describe the development of a simple movie recommendation system using machine learning techniques. The process included data preparation, data exploration, and numerous rounds of improving the model.

The primary goal was to improve the accuracy of the recommendation system. This was measured by how accurately the models predicted ratings (on a 1-5 star scale) in a test set. Accuracy was measured using Root Mean Square Errors (RMSE). RMSE is a common metric for assessing the accuracy of predictive models. It measures the distance between the predicted values and the actual observed values. A lower RMSE indicates a better fit of the model to the data. In this case, an RMSE of 1 would tell us our predictions were, on average, one star away from the actual rating.

Data Overview

I built the recommendation system using 10 million rows of movie review data from MovieLens. Using a couple of files from MovieLens and code provided by edx, I created a dataframe with six columns and 10 million rows: userId, movieId, rating, timestamp, title and genres. UserId differentiated each user who rated movies. MovieId indicated which movie was being rated in each row. Rating was measured on a 1-5 star scale, with half stars possible. Timestamp measured when the movie was rated. Title matched with movieId and indicated the title of the movie. Genres included all genres associated with each movie.

The huge number of rows made creating models difficult, as it limited the types of functions that could be applied to the data set.

Data preparation

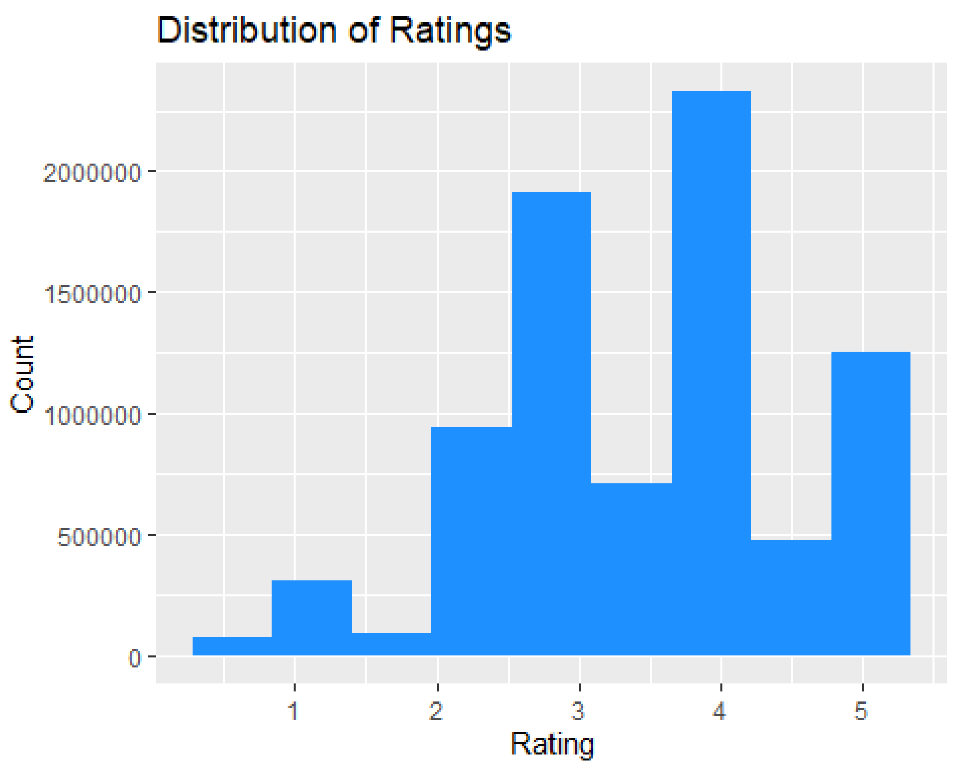
To prepare the data for analysis, it was split into a training set, a testing set and a final holdout set. I used the training set to develop my models and the testing set to test for accuracy. The final holdout set was not used until I completed a final model. It provided a second check of the model's accuracy.

Data exploration

Using the functions nrow, mean, median, sd, n_distinct, and summarize, I explored the dataset. This allowed me to understand its basic size, average values, distribution, and number of distinct users and movies. It showed me that the dataset has far more users than movies, that the ratings have a standard deviation of about 1, and that the average rating is either about 3.5 or 4, depending on if you use mean or median.

```
## Number of rows in the dataset: 8100067

## Mean of 'rating': 3.512509

## Median of 'rating': 4

## Standard Deviation of 'rating': 1.060242

## Number of unique users: 69878

## Number of unique movies: 10677

## # A tibble: 10 × 2
##    rating   count
##     <dbl>   <int>
##  1    0.5   76805
##  2    1    310999
##  3    1.5   95704
##  4    2    640140
##  5    2.5  299780
##  6    3   1909414
##  7    3.5  712861
##  8    4   2329196
##  9    4.5  474017
## 10    5   1251151
```

I then created a histogram to visualize the distribution of the ratings, which showed how skewed they are to the top half of the 5-point scale. While the mean and median helped portray this as well, the histogram drove the point home. Also, given just the high mean and median, I would have expected more ratings of 5, but the histogram showed the overwhelming use of 4 as a rating.



Analysis / model building

Baseline linear models

As I began to construct a model, I started with a simple linear formula.

I calculated the mean of all movie ratings (mu) and the average distance from mu for each movie (b_i). This allowed me to predict ratings based on which movie is being rated, which produced an RMSE of .944.

I then similarly calculated the user effects (b_u) by finding the average deviation of users' ratings from the overall mean, accounting for movie effects. This allowed me to build a linear model with both user and movie effects, which produced an RMSE of .866.

Penalized linear model

Since some of the movies have very few reviews and some users entered few reviews, I decided to use a penalization technique to prevent overtraining. This allowed me to constrain/regularize individual effect sizes for both movies and users.

When testing this penalized model, the RMSE ended up being roughly equal to the model without this penalization.

An attempt at incorporating genre effects

I next tried to incorporate effects from the genre column. Since the genre column combined all genres that applied to the given movie, I started by separating the genres out from each other using a matrix. Then, I calculated an average effect size for each individual genre and used these to try to better predict movie ratings.

Unfortunately, this approach was less accurate at predicting ratings than my prior models that didn't include genre effects. My original attempt produced an RMSE of .935, and a second, tempered, attempt produced an RMSE of .884.

Matrix Factorization using the Recosystem package

Because the dataset was too large to use many functions that help build predictive models, I decided to conduct matrix factorization using the Recosystem package, which is designed for the analysis of large databases. The package operates on sparse matrices. This technique is quite memory efficient for a database like this one, where there are a lot of "empty" data points (each user generally rates a small percentage of the movies). The package is also easy to use; for instance, it allowed me to quickly, easily and precisely regularize/penalize the dataset.

Matrix factorization has been shown to be helpful for recommendation systems, since it can find relationships between users and their ratings that go beyond a linear model.

I started with a large number of dimensions to help identify complex patterns in this huge dataset. However, I chose to reduce it to c(20,30) to speed up the process for replication.

When selecting regularization parameters, I tried to balance the need to prevent over-fitting with the restraints of my laptop. For both costp_12 and costq_12, I ended up using c(.01, .1).

With the lrate, I similarly wanted to balance accuracy with the time it would take to run the function. I thus selected c(.005., .05).

Using these conditions, I then used the Recosystem to predict movie ratings. Compared to the true values in the test set, it's RMSE was .78965, which was significantly lower than my linear models.

Results

Using my best model, developed using the Recosystem package, I predicted reviews in the final holdout dataset. The RMSE was 0.78963, which was almost exactly the same as the RMSE from the original test set and a significant improvement on my basic linear model.

Conclusion

This report has documented the development of a movie recommendation system, able to recommend relevant movie suggestions to users and predict how users would rate them. It has explored several models, including simple linear models, regularized linear models, and matrix factorization. The matrix factorization model emerged as the most promising approach.

My findings underscore the usefulness of matrix factorization techniques for building effective recommendation systems. While I made significant progress, there is always room for further enhancements and refinements in the pursuit of more precise movie recommendations.