Report - Movielens Project

Summary

The Movielens database contains millions of user reviews of movies. These reviews contain information about what genre a movie is classified as, when the review was submitted and what rating from 1-5 the user gave the movie as well as the year a movie was released. I've been asked to predict how users will rate movies they haven't seen based on how they have rated movies they have seen. To succeed in this project my prediction should have an RMSE (Root Mean Squared Error) of less than 0.86490.

Workflow

I explored the data provided to look for connections and relationships in the data. Next I evaluated which approaches I could try, mainly resulting in not trying "traditional" Machine Learning algorithms. For example Linear Regression can be bogged down by the sheer quantity of data in this dataset. Finally I tried three different approaches to predict how people would rate movies:

1) Naive approach:
    a) start with predicting the overall average
    b) then try the average for each movie
    c) adjust further with other factors to arrive at a final model.
2) Try to create a Matrix Factorization algorithm manually in R. I abandoned this as too slow, same as traditional Machine Learning, as for example one gradient descent step took over 1 hour on my computer and had not completed.
3) Use a pre-written R package to achieve sufficient parallelization to make Matrix Factorization viable.

In the end, the approach I chose to accomplish this task was Matrix Factorization, using the package `recosystem`. Using this approach, **I achieved an RMSE of 0.78005**, well below the required 0.86490.

Exploring the Movielens dataset

Size of the datasets and complications

I started off reviewing some key statistics about this dataset. The data has been split into two sets, one for training and model evaluation (there are over $9,000,000$ reviews in this dataset) and one for final validation (there are $999,999$ reviews in this dataset). There is a total of $69,878$ unique users and $10,677$ unique movies. These numbers made me worried about traditional Machine Learning approaches on my home computer as this might be outside of it's capacity.

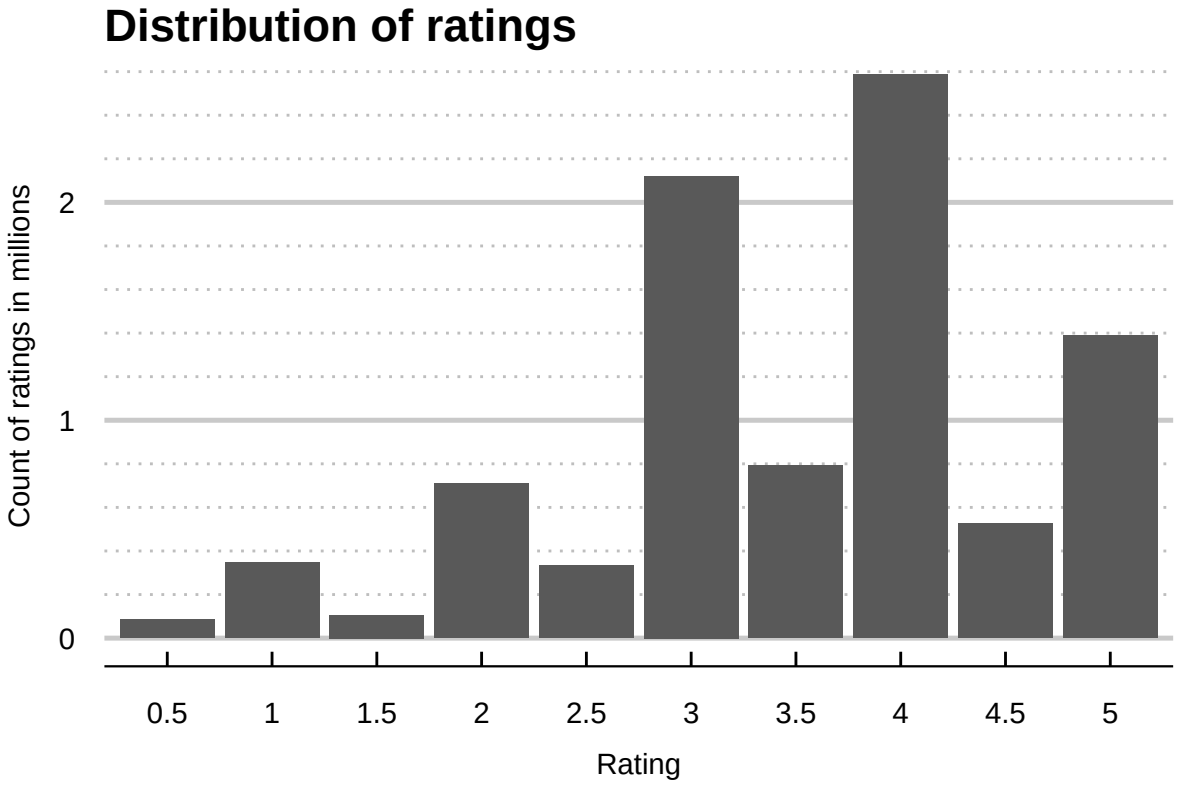
Sparsity of data

Even with this large number of reviews the total number of possible reviews is users $\times$ movies $= 746,087,406$ and thus there is a sparsity in a users $\times$ movies matrix of 1.21% (1.21% of possible reviews are in the dataset) which means it's also hard to generalize trends that are valid for the unknown values using traditional Machine Learning methods.

For the rest of this section on exploring the dataset the exploration is done only on the training set of data.

Distribution of ratings

Taking a look at how the different ratings are distributed with a histogram counting how many million times each rating has been given as a review:



Immediately a couple of things are visible, half-point reviews like $\{0.5, 1.5, 2.5, 3.5, 4.5\}$ are far less common than the whole-point reviews $\{1, 2, 3, 4, 5\}$, also high ratings are more common than low with a mean for the whole dataset of 3.51 which is 27.7% more than the median of the possible ratings, 2.75.

The effect of the numbers of reviews on averages

Highest and lowest rated movies

Looking at the highest rated movies I found several movies with 1 or 2 reviews getting an average rating of 5, if I filter out movies with less than 10 ratings the top 5 movies sorted by average rating are:

Title	Rating	Reviews
Shawshank Redemption, The (1994)	4.46	28,015
Godfather, The (1972)	4.42	17,747
Usual Suspects, The (1995)	4.37	21,648
Schindler's List (1993)	4.36	23,193
Casablanca (1942)	4.32	11,232

and the bottom 5 movies are:

Title	Rating	Reviews
Pokémon Heroes (2003)	1.03	137
From Justin to Kelly (2003)	0.90	199
Disaster Movie (2008)	0.86	32
Hip Hop Witch, Da (2000)	0.82	14
SuperBabies: Baby Geniuses 2 (2004)	0.79	56

As suspected from the distribution of ratings the highest rated movies have a lot more reviews than the lowest rated movies, this makes it interesting to see if the amount of reviews affect the average rating across the whole dataset.

Users with the highest and lowest average rating

Looking at the users with the highest average rating I found that there are several users who have only given 5s to movies,

User	Rating	Reviews
11,564	5	18
12,682	5	29
21,472	5	18
51,551	5	85
27,126	5	20

and users giving only 0.5s to movies have reviewed a similar amount of movies:

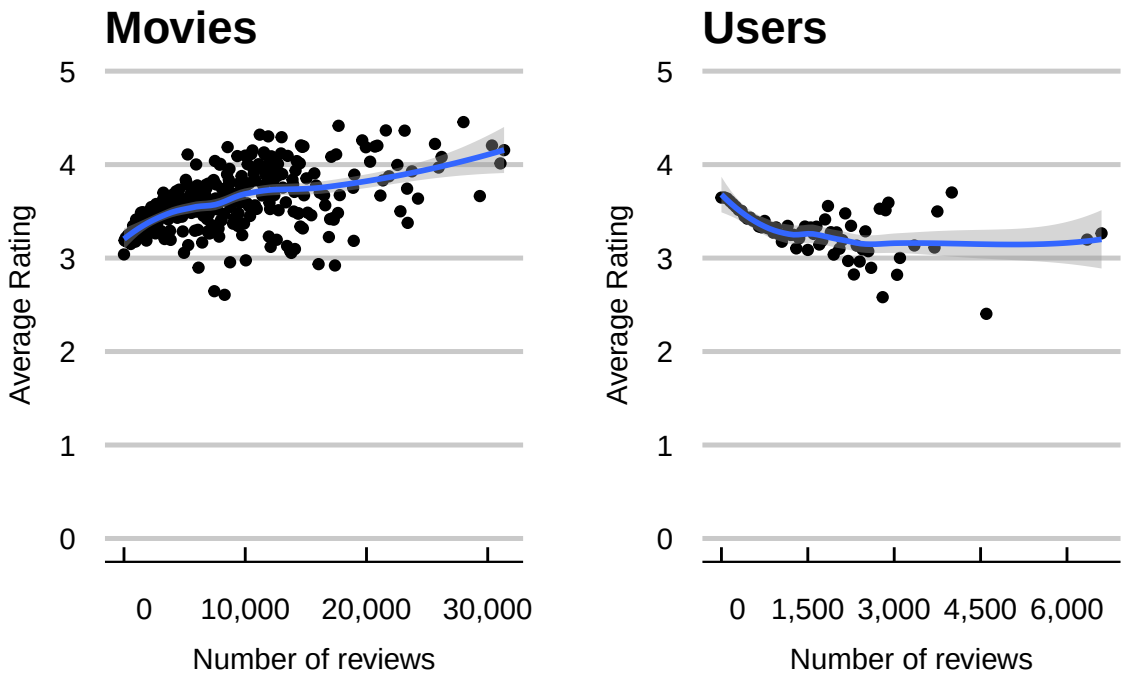
User	Rating	Reviews
47,033	0.5	25
61,436	0.5	20
48,722	0.5	17
61,992	0.5	18
13,140	0.5	17

Both of these amounts are small compared to the large number of reviews in the dataset so it seems user review habits affect the average rating less than the movie's average.

Ratings as a function of number of reviews

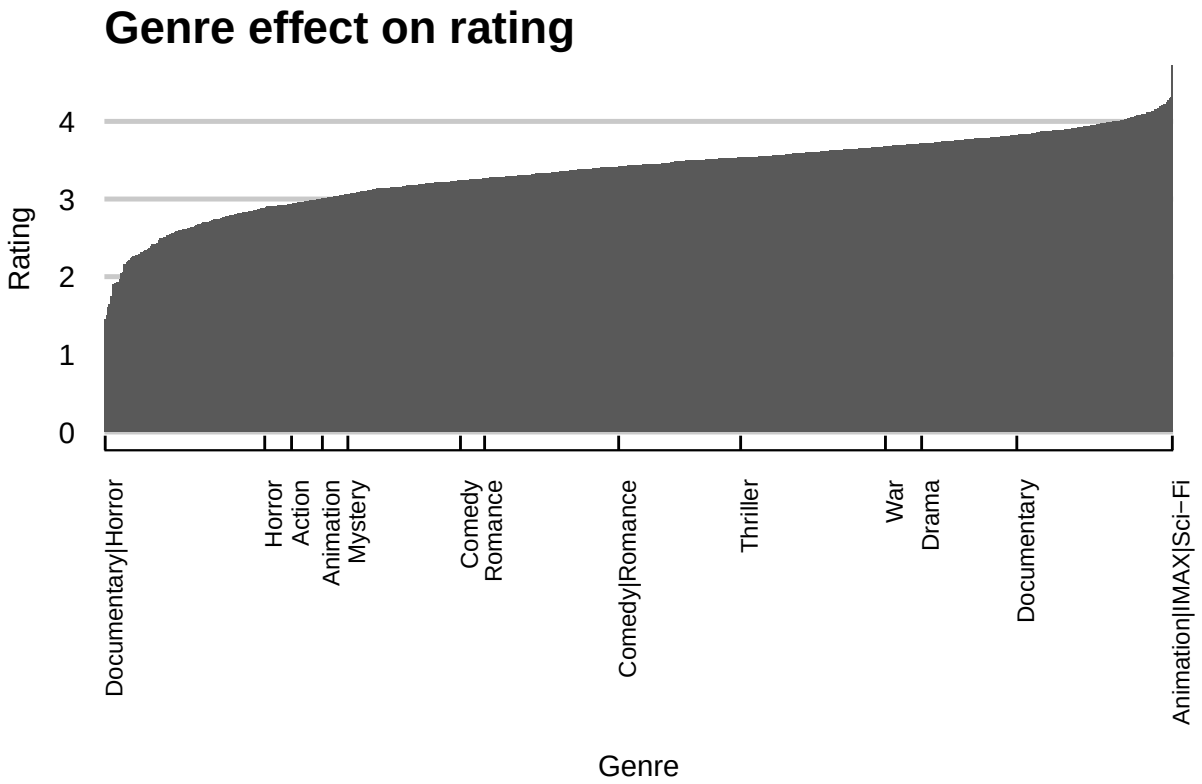
Looking at average rating as a simple function of number of reviews there does appear to be a trend, but the graphs with this many datapoints are quite hard to read and heavy computationally. To make it a bit easier to read the graphs and also to make the computations faster, I created groups of movies and users for every 50 reviews received or given, naturally the groups with fewer reviews will have more members than the groups with more reviews. Interestingly there are opposing trends in these two perspectives on reviews. Movies who receive more reviews receive higher ratings, while users who give more reviews give lower rating up to a certain point where this trend flattens out a bit above a rating of 3.

Comparison of users and movies by amount of reviews



Effects of genre on ratings

There are over 700 different "Genres" in this dataset, created from keywords like "Drama" and "Comedy" in different combinations. Some genres have as few as 2 reviews or 1 movie others have over $700,000$ reviews and over 1000 movies. Overall there is a wide spread of average ratings received depending on which specific genre a movie belongs to. Here's a barplot of the genres arranged by average rating from smallest to largest with 13 examples displayed to show the spread in average ratings.



As seen here the lowest rated genre received an average rating around 1.5 while the highest rated genre has an average over 4. Considering how spread out major genres appear to be using information about other movies in a genre should help give an indication of trends for a movie.

Effects of time on ratings

The Movielens dataset contains the release year of the movie as part of the title field as well as the date and time of the review as a timestamp. From these I've created some time-related fields to evaluate which if any might be interesting to look at.

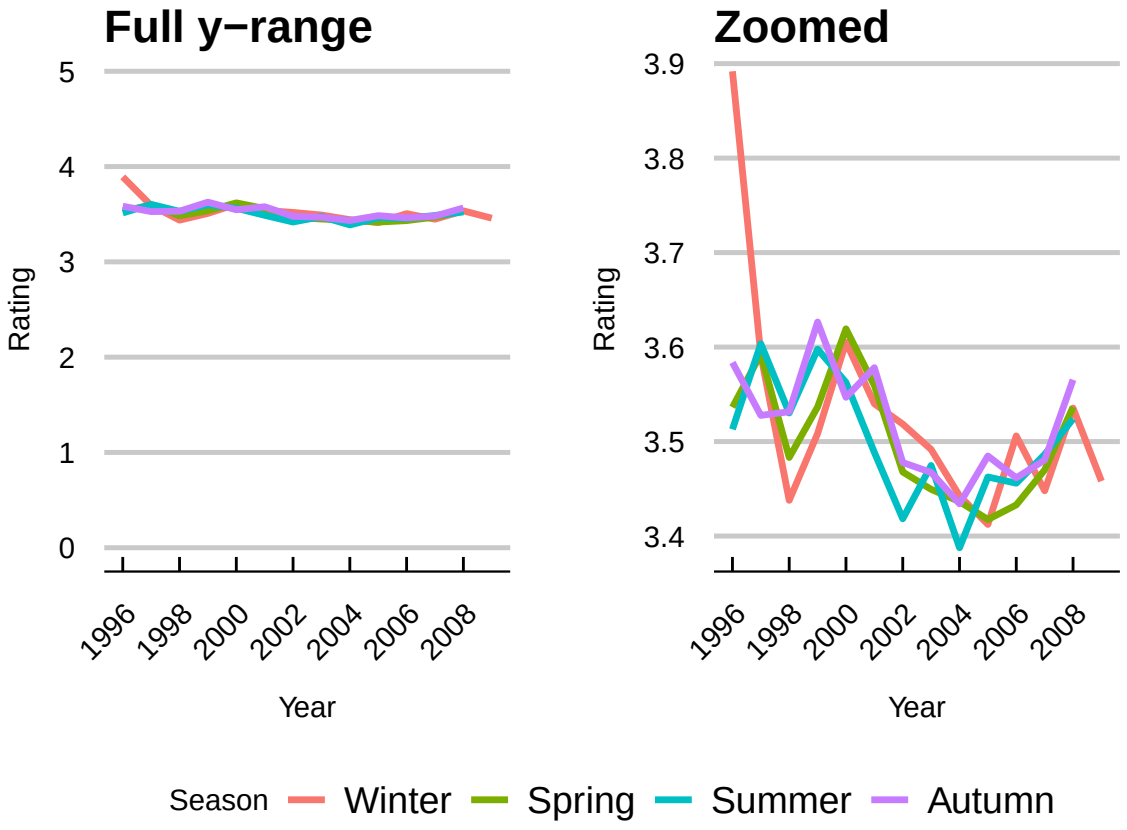
Seasonal reviews

Classifying months $1 \rightarrow 3$ as Winter, $4 \rightarrow 6$ as Spring, $7 \rightarrow 9$ as Summer and $10 \rightarrow 12$ as Autumn gives us a seasonal information about the reviews. There appears to be hardly any difference between the seasons, so unfortunately no easy wins like people give lower reviews in winter when they're more blue because of the season.

Winter	Spring	Summer	Autumn
3.50	3.50	3.49	3.54

Year and season of review

While there doesn't seem to be much seasonal variance, it does appear like there is a small drop in reviews over time indicating that maybe newer movies receive lower reviews than older, but this is a very slight effect that is only visible when one zooms in on the graph:

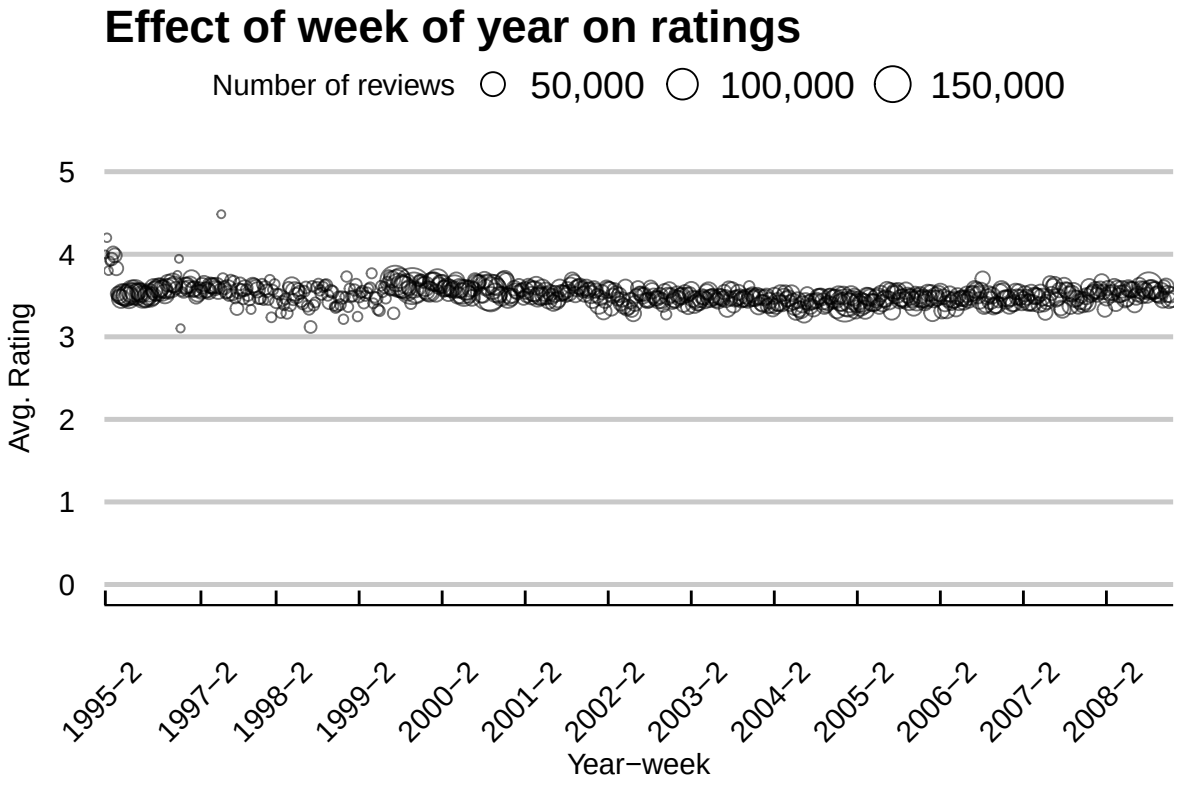


Winter 1996 The spike in ratings in Winter 1996 is due to very few reviews being given to the movies reviewed in that time period, probably this is when the reviews for the dataset first started being collected. The movies with the most reviews in this period and their average ratings are:

Title	Reviews	Rating
Pulp Fiction (1994)	105	4.32
Apollo 13 (1995)	98	4.57
Batman Forever (1995)	85	3.92
Batman (1989)	79	3.94
Dances with Wolves (1990)	78	4.37

Week of review

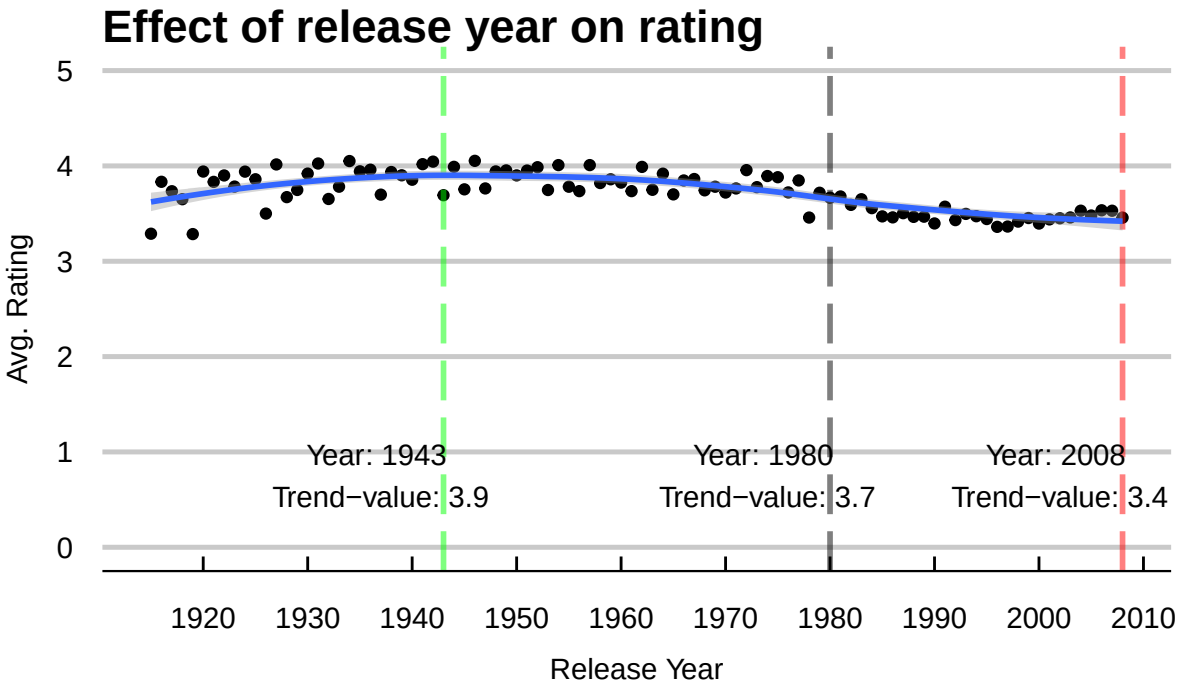
A more fine grained inspection of the seasonal trend by looking at the week of each year reveals that the time of year really does not appear to have any significant impact on the average review given, maybe this could be made more useful if it could be combined with at least which hemisphere a review comes from since I don't know if for example Australian user reviews will cancel out with Canadian reviews due to being seasonal opposites.



As seen here there's, apart from a few outliers, little variation away from 3.5 for any random week selected in the range of our dataset.

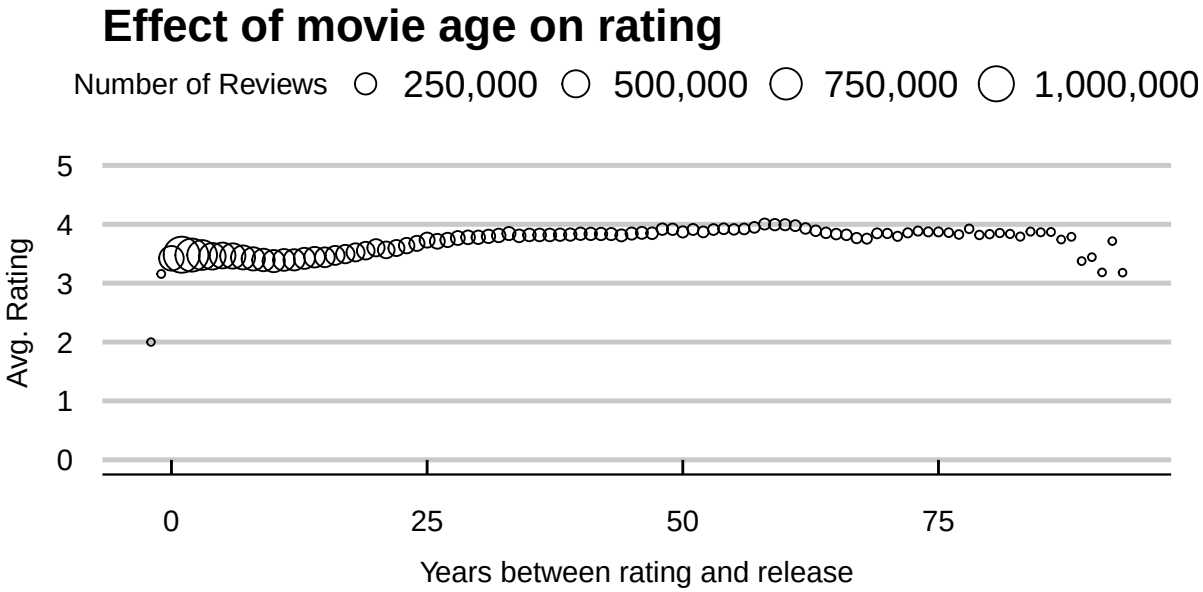
Release year

Since there appears to possibly be a trend that newer movies receive lower ratings than older I looked at the release year of movies next. There is a slight rising trend in the reviews based on movie release year up until 1943 when the average rating seems to taper off and movies from 1980 and later are getting a markedly lower average than before 1980.



Age of movie when reviewed

When looking at the time between reviews and the movie's release it seems that most reviews come in the first few years after release, but there's a slight upward trend in ratings given as the years go by.



While this increase is modest, and in part can be explained by movies being released before the debut of collecting reviews, it is quite a bit more significant than for example the season when a review was given.

Predicting reviews - results

From the instructions for this project I've received two datasets, one intended purely for final validation and one for working with my model. To be able to perform intermediate evaluation of a model's performance I've split the working dataset into a training set (80% of the working set) and a test set (20% of the working set).

Building a simple model based on training data

Guessing every review is the global average

I start predicting scores in a simple and "naive" way by predicting that all reviews are equal to the average review over the whole training dataset: 3.51 this gives me a pretty large RMSE of 1.06046 which is significantly above the goal of 0.86490.

Method	RMSE	Goal met:
Average only:	1.06046	FALSE

Guessing every review is each movie's average rating

My next attempt is to predict that each review is equal to the individual movie's average and this gets me closer:

Method	RMSE	Goal met:
Average only:	1.06046	FALSE
Movie effect:	0.94358	FALSE

Taking into account the individual user's average

I expand on this method by taking into account the effect of each user on my predictions thus having my first full matrix of users $\times$ movies populated with individual predictions and this gets pretty close to the target of 0.86490:

Method	RMSE	Goal met:
Average only:	1.06046	FALSE
Movie effect:	0.94358	FALSE
User effect:	0.86632	FALSE

Additional effects

From my exploratory data analysis I know there are a number of other factors that can have fairly large effect on the final average rating of a movie I'll populate our table with some of them now:

Number of ratings

Method	RMSE	Goal met:
Average only:	1.06046	FALSE
Movie effect:	0.94358	FALSE
User effect:	0.86632	FALSE
Rating group effect:	0.86585	FALSE

Movie genre

Method	RMSE	Goal met:
Average only:	1.06046	FALSE
Movie effect:	0.94358	FALSE
User effect:	0.86632	FALSE
Rating group effect:	0.86585	FALSE
Genre effect:	0.86566	FALSE

Week of review

Method	RMSE	Goal met:
Average only:	1.06046	FALSE
Movie effect:	0.94358	FALSE
User effect:	0.86632	FALSE
Rating group effect:	0.86585	FALSE
Genre effect:	0.86566	FALSE
Week effect:	0.86552	FALSE

Release year

Method	RMSE	Goal met:
Average only:	1.06046	FALSE
Movie effect:	0.94358	FALSE
User effect:	0.86632	FALSE
Rating group effect:	0.86585	FALSE
Genre effect:	0.86566	FALSE
Week effect:	0.86552	FALSE
Release year effect:	0.86537	FALSE

Age of movie at time of review

Method	RMSE	Goal met:
Average only:	1.06046	FALSE
Movie effect:	0.94358	FALSE
User effect:	0.86632	FALSE
Rating group effect:	0.86585	FALSE
Genre effect:	0.86566	FALSE
Week effect:	0.86552	FALSE
Release year effect:	0.86537	FALSE
Age of movie effect:	0.86516	FALSE

Review of results After all this work I'm seeing very modest improvements for each factor added, while close to the target RMSE of 0.8649 the results are not quite there. It might be time to re-evaluate the approach to this problem.

Matrix factorization

Why change approach

In my naive approach I tried to account for all the factors found in the data and adjust predictions based on their impact on reviews. What I've not accounted for are unseen factors like a movie's director, the actors etc. For example someone who is a big fan of Tarantino might give all Tarantino movies a review of 5 based on that alone. This type of information is not apparent or exploitable in the naive approach. Matrix factorization tries to discover unseen factors like these to account for variations in the data. Programming a matrix factorization algorithm in R is not very easy as it requires intense use of multiple cores to do effectively, so I've chosen to use the package `recosystem` that is written in C for this.

Brief simplified theory

Matrix factorization works by considering that the combination of users and movies form a matrix with one row for every user and one column for every movie (or vice versa). This matrix will be mostly empty because each user have seen only a small selection of the movies. In the context of this project the intention is to replace all the empty fields with a prediction for how each user will rate a movie they haven't seen. To do this the users × movies matrix is considered to be a matrix product of two unknown matrices that are users × hidden factors and hidden factors × movies. The number of hidden factors is somewhat arbitrary and becomes a parameter that can be tuned to obtain the best results.

Possible reviews by users in rows and movies in columns, empty fields are represented by zeroes:

$$\begin{pmatrix} 1 & 5 & 0 & \dots & 0 & 3 \\ 0 & 5 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 2 & 0 & 1 & \dots & 4 & 0 \end{pmatrix}$$

Users and two hidden factors could for example be represented like this:

$$\begin{pmatrix} 1 & 0.5 \\ 3 & 1.6 \\ \vdots & \vdots \\ 2 & 0.2 \end{pmatrix}$$

And two hidden factors and movies like this:

$$\begin{pmatrix} 3 & 2 & \dots & 1 \\ 0.78 & 1.64 & \dots & 0.43 \end{pmatrix}$$

The matrix product of these two matrices would be the prediction for the full matrix of users $\times$ movies to use to for example recommend movies with a high predicted rating to the user as something they might like.

The way this matrix is populated is done like this:

1. Generating random values for the hidden factors for both users and movies
2. Calculate the filled out matrix with predicted values.
3. Compare predicted values to the values that exist in the training set
4. Consider one of the matrices for example the user matrix "locked"
5. Perform a gradient descent step on the other matrix (movie in this example) to move closer towards the correct values
6. Calculate a new hidden factors $\times$ movies matrix.
7. Repeat from 3 but swap which of the two matrices is kept unmodified
8. Repeat $3 - 7$ until convergence.

Recosystem results

Even with default options `recosystem` produces an RMSE of 0.83662 which is quite well below the required 0.86490, if the algorithm is tuned that can be improved by a fairly large amount, but this tuning process takes time to run (several hours on my computer).

Tuning parameters

Default options Recosystem comes with two good default choices for each of the options that can be tuned, there are 6 of these options giving $2^6 = 64$ combinations, this takes a fair amount of time to run through. After running the 64 combinations the default tuned parameters of `recosystem` gives an RMSE of 0.79519.

Custom tuning Each of the 6 options can be given ranges to fine tune the performance, but as each change to one parameter can cause the ideal choice for another to change this can be very time consuming. I spent a few hours tuning this with moderate improvements at best over the default tuning, but after running the tuning process to have a final list of parameters I achieved an RMSE of 0.78915 on the test set. Since that is quite well below the target I re-train the model using the parameters I found using the full working dataset to train and then run a final validation on the validation dataset set aside for this.

Final run

Finally after all the hard work I've trained my model with $9,000,055$ datapoints, and achieved a final RMSE of 0.78005 on the $999,999$ datapoints in the validation set. I can now recommend new movies to users based on their history of reviews with a high degree of confidence.

Conclusion

There are many factors that can impact a prediction of a review from a given user on any movie, I've explored some of them in this report. The achieved RMSE of 0.78005 is quite respectable and is a good result.

Next steps

To further improve the on this I might try other Matrix Factorization packages, but I would also have liked to try a Neural Net on this task. Unfortunately my laptop isn't equipped with an nVidia GPU and that makes Neural Nets quite a bit slower to use for large matrix calculations, but in future that's a possibly strong contender for this type of prediction.