Like-Wise Tastes and Suggestions: Recommender Systems

An examination and technical data analysis of the MovieLens 10M dataset

Anthony Balducci

2023-11-01

A brief history and overview of recommender systems

While ubiquitous on both the internet and in society today, recommender systems are not quite as recent development as one may realize. The first such system, the 'Grundy' system was first developed all the way back in 1979 by Elaine Rich as a means of providing a potential user a book recommendation based on

their 'stereotype'[1] However, such applications and research in this area didn't really take off until the rise of Amazon, social networks, and especially Netflix with its Netflix Prize in '08-'09 that offered a million dollar prize to any group that could increase the accuracy of their recommendation system by 10%.[2].

Aside from customer satisfaction– or perhaps more importantly to businesses, the potential for increased sales, one of the large motivating developments in research in this area is an issue only large online businesses face.

While a 'brick and mortar' store can have prominent in-person displays, they are also restricted to a limited number of items they can actually stock. When Jeff Bezos at the start of Amazon once famously said he wanted them to carry 'a million books', a big question was how do you get a customer interested in a product they might like, yet have never even heard about?

However, even as research and development has significantly advanced in the implementation and deployment of such systems they remain subject to significant issues of potential bias.[3] One such particularly devious cited example pertains to the case of 'position bias' – That is a customer or user, in receiving a recommendation and being notified as much, may instead be being driven by the seller or service towards a product *they* wish the customer to buy. Which is to say, as the vast majority of these systems are opaque in design to the end-user, they can't quite be sure that what is being recommended to them necessarily has their individual tastes and preferences in mind. In such cases then this form of engagement serves as more a means of implicit advertising than a helpful suggestion.

Even when bias (intended or not) isn't present there are other potential concerns and challenges in ensuring an appropriate experience for the user. While not directly a concern in our case of examining the MovieLens Database (there are no users we encounter here that have *no* reviews– the fewest is userId 62516, which has at least ten), consider that has no reviews nor purchases? In industry parlance this is as what is known as a 'Cold Start'. With no prior history to guide recommendations one may have noticed that many new services request of the user when they first sign up to express what may be some of their likes and preferences, if nothing else as a way to prime the system.

Overview

Present day recommendation systems utilize a number of different algorithms. Currently collaborative filtering is one of the most commonly used recommendation algorithms. It recommends items based on preference information from many users. There are two main approaches: user-based and item-based. User-based CF finds people with similar interests, analyzes their behavior, and recommends the same items. Item-based CF looks at items similar to ones which the user bought earlier, and recommends products which are like them. On the other hand content based filtering recommends items by comparing the content of the items and a user profile. The content of each item is represented as a set of descriptors, such as the words in a document.

Whereas hybrid systems use a combination of CBF and CF. They can provide more accurate recommendations by benefiting from both content and collaborative data. In addition,more recent systems depend on neural nets as well.

Finally, there is matrix decomposition to predict unknown grades. It approximates a rating from a user to a file with the dot product of two vectors, in this case userId and movieId.

The MovieLens database we are focused on in this assignment is the 10M data point variant containing the reviews of 69878 unique users rating 10677 films over a period from 1995 to 2009. While the data used in the Netflix prize is not publicly available, this resource from the Grouplens organization out of the University of Minnesota is, and it is great to practice on.[4]

[1] RICH, Elaine. User modeling via stereotypes. Cognitive science, 1979, 3. Jg., Nr. 4, S. 329–354.

[2] Further, it is worth noting at that time it took the winning team 2 years to solve a task we are now asked to complete in a mere 6 months

[3] https://towardsdatascience.com/biases-in-recommender-systems-top-challenges-and-recent-breakthroughs-edcda59d30bf

[4] https://grouplens.org/

R, Data Science and the emerging challenges of dealing with 'Big Data':

While over the past 30 or so years computing power has become exponentially both more powerful and ubiquitous, occurring in lockstep with that has come the ease of increasing data collection. On the surface of it, towards the end of any decision or reasoning process, at first having 'more data' always seems better. As the central limit theorem states, the larger the sample size, the closer to the true mean of the population in question one can get.

However, simply having more data also makes it increasingly more challenging many times to process, store and analyze. For myself, this is certainly the case in the examination of the MovieLens Dataset. While we will soon take a look and a few basic visual explorations of the data, aside from drawing a few broad and overall conclusions we now we find we have less access to both visualizing it conceptually or, more importantly, in the case of providing individualized recommendations at the granular level.

Thus, in my mind is why we have encountered the rising of 'Data Science' as a discipline, while building and in conjunction with, that is separate for formal Statistics. Upfront, with that in mind I would like to share a few insights I've gain in the process of analysis:

1. When you can, go 'parallel': While even a low-end inexpensive laptop or computer is quite fast these days, processor manufacturers have also realized they have hit a physical limit due to heat dissipation, transistor density and other concerns to simply keep accelerating clock speeds. Thus, though even today many pieces of software still don't make use of them, almost all processors on the market are 'multi-core'– meaning they can process more than one piece of data at a time simultaneously.

Luckily for us, many packages in R do support this, but first one must enable it. The core package that provides this is library(parallel). While this can be used directly to parallelize even one's own custom R code outside of a library via the 'foreach'[5] syntax it does have a bit of a steep learning curve. For most users a 'helper' package more straight forward to get stated. There are any number of these available, such as doMC, SNOW, etc, in my case I found these unsuitable as they only work in a UNIX/Linux environment while I am on Windows.

```
library(parallel)
library(doParallel)
numCores<-detectCores() # Detect cores
pCluster <- makeCluster(numCores[1]-4) # Select the number of cores to assign
registerDoParallel(pCluster) # Register the cluster
```

It is pretty easy to get started ! However, do refer to the extensive notes on this in my code for much more detailed notes– Don't be surprised if you crash your system a few times until you get the tuning worked out right for your particular system.

And finally, even though multi-core is now enabled and many packages, including caret support it, it is not 'turned-on' by default, so always check your packages documentation. As an example from my code in the case of caret, this occurs in the particular function one defines for the 'trControl' (training control) command. The key here is to set the optional parameter 'allowParallel = TRUE'.

```
trainControl (verboseIter = TRUE, allowParallel = TRUE, number = 5, method = "cv")
```

2. While enabling parallel processing is any easy way to boost performance that nearly any user can accommodate this days, in the end, when working with large datasets in R it is 'all about the RAM'. RAM, RAM, and even more RAM. During the course of my studies on this project I ended up upgrading

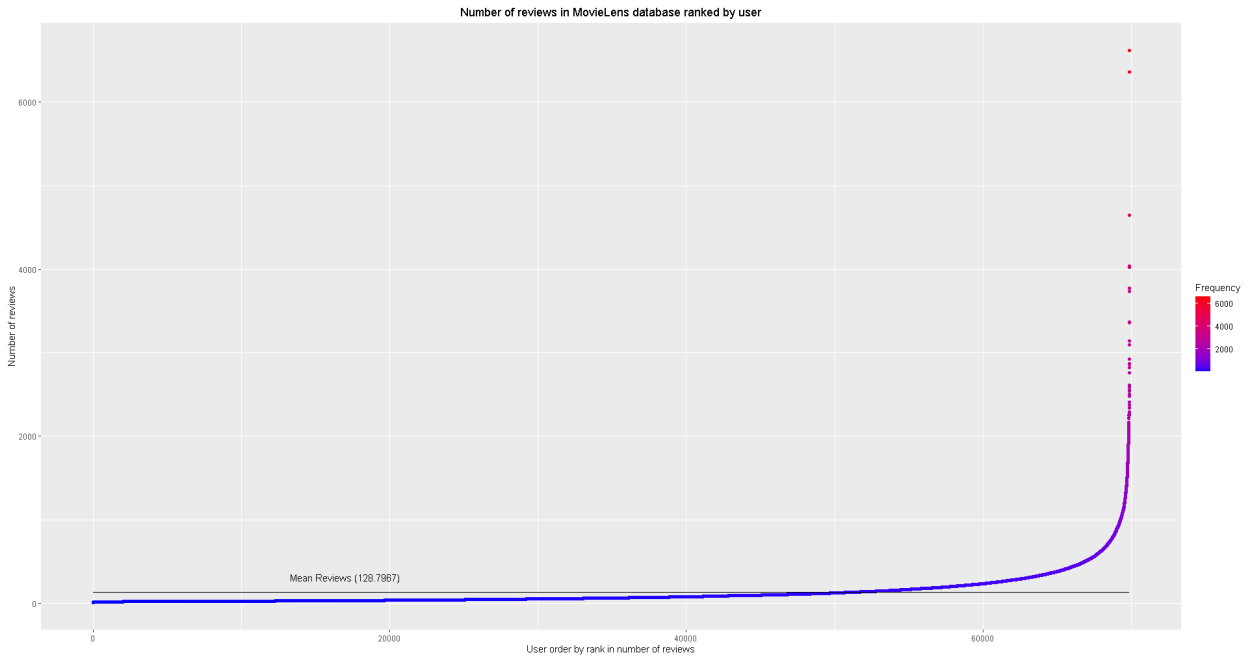
[5]https://nceas.github.io/oss-lessons/parallel-computing-in-r/parallel-computing-in-r.html
Therefore I eventually settled on the 'doParallel' library which *does* work in a Windows environment. Once installed, from my code, my setup for this package is as follows:

my system three times in obstinance not to be deterred from the challenge, from 16 GB, which is honestly fine for basically all the other work I do, three times. First to 48 GB, then to 96 GB, and finally to 128 GB.

It simply was just never enough to perform a full, all out traditional analysis on this data set. When working with Big Data in R, this will truly be your bottleneck. On certain attempted forms of regression it would even soak up all 128 GB and be looking for still more. Thus it is important that other, more approachable methods of being creative with various data analysis techniques such as Bootstrapping and Ensembles is crucial and should not be ignored.

3. While on the face of it more data may always seem better it is important to consider the quality of that data and consider whether or not it sufficiently contributes useful information. Claude Shannon, the so-called "father of information theory" in his seminal 1948 work, "A Mathematical Theory of Communication" outlined his concept of information theory as "as a measure of the information content in a message, which is a measure of uncertainty reduced by the message".[6] Within the context of Machine Learning, especially, and Statistics, additionally one important consideration regards the role of outliers in a data set. Typically these are treated as 'distortions' in the signal in the 'information' we which to seek to derive and predict outcomes.

As an example of this consider this plot from the EDX train set of the MovieLens database:



We can determine the mean, min, and max number of user reviews as follows:

```
numReviewsByUser <- table(edx$userId) %>% as.data.frame() %>% arrange(Freq)
meanNumReviewsByUser <- mean(numReviewsByUser$Freq)
maxNoReviews <- max(numReviewsByUser$Freq)
minNoReviews <- min(numReviewsByUser$Freq)
```

meanReviews	maxReviews	minReviews
128.7967	6616	10

[6] https://en.wikipedia.org/wiki/Claude_Shannon

One thing one notices right away is a rather small segment of the population apparently watches a *lot* of movies. In fact, 75% of the population data has reviewed 141 films or less.

Second is the fact that, at least as far as the number of reviews per user goes, this relationship is definitely *not* linear but more exponential. Were we considering this fact alone a standard linear model would not fit these outcomes well.

However, in the end the work at hand is to build a recommendation system. Thus ought we discard the tier end of the top 25% as 'outliers'? Or, like the guy that used to work at the video store and has seen *everything* make them better suited to recommend new films to others?

What about the bottom 25% that has reviewed only 10 to 32 films? Are they too *inexperienced* to provide suitable recommendations?

Provided the mean number of reviews is 128.7967, if we removed all users from the data set we'd automatically decrease its overall size by roughly 8%. The questions and decisions thus made by the data scientist are thus circumstance specific and part of the reason why there remains an 'art' to data science.

In this case it is important to keep in mind that Data Science itself provides us with certain tools to help inform us in this decision. A particularly useful one is the concept of regularization, which I have taken advantage of here, be it via Ridge Regression, Lasso, or Elastic Net methods.

Regularization works by adding a penalty to the loss function of the model. This penalty discourages the model from assigning too much importance to any single feature, thereby reducing the model's complexity. The effect is that the model's coefficient estimates are shrunk towards zero, which can result in a more robust model that generalizes better to unseen data.

In closing, to return to the notion expressed by Claude Shannon, there remains the question which of these subsets of the population is truly providing us with the most 'information'?

4. Finally, while of course at least having *access* to more information is preferable to less, one has to further take into account the costs, be they time of analysis or equipment expense, that goes into properly utilizing it.

Methods and Analysis

From the beginning, aside from exploration and dealing with large data sets, our primary goal was to reduce our Root Mean Squared Error as much as possible. The Root Mean Square Error (RMSE) is a measure used to assess the performance of a regression model. It calculates the average distance between the predicted values from the model and the actual values in the dataset. The formula is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

Not satisfied with the very rough estimation methods we went over in the course– nor to mention that these methods still would not be able to lead one to actual predictions, I was determined, despite the constraints of such a large data set on resources to find a workable solution. – And I admit, I spent around two months beating my head against the wall in the process. Either the models available that I first tried were too inefficient, ran for days without results, or even with a souped up desktop as I previously mentioned would run over night and then just crash.

Smaller sample taken from the test sets, either via sample_() or slice_sample() often produced lack luster results.

Before we delve into that lets consider the forms of equations that were considered. One of the first, and at times, most important questions arises when one must consider feature selection, or additionally feature engineering.

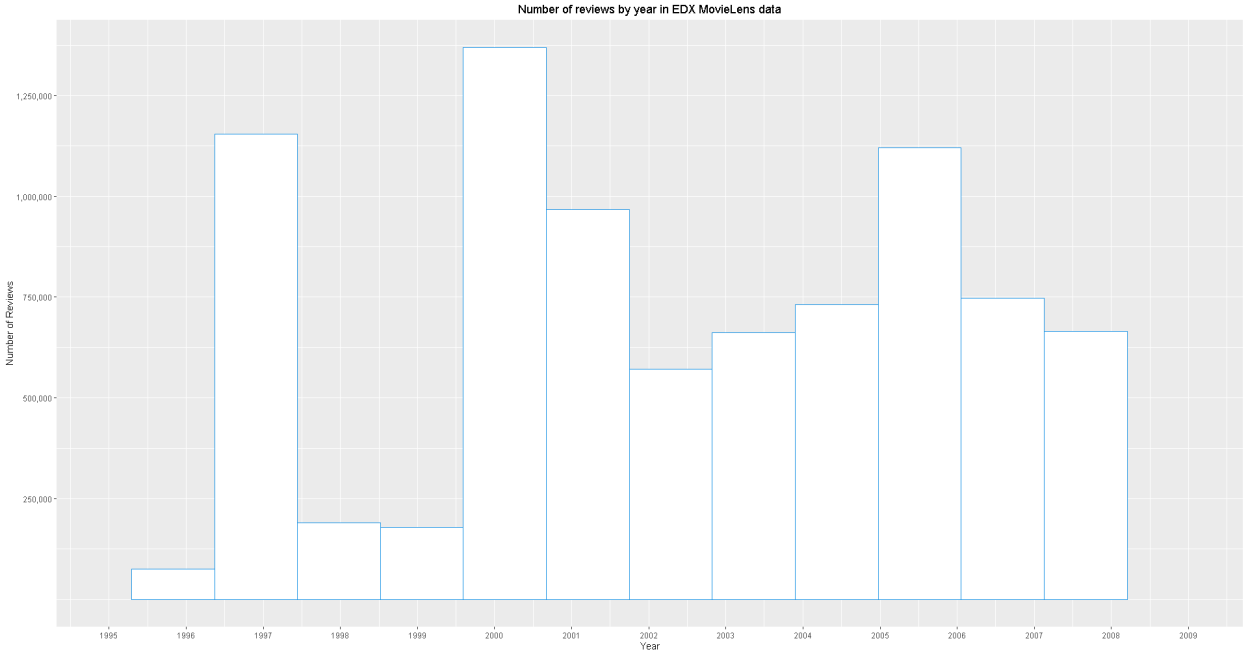
While at first I tried for some time working with standard lm() or logistic based regressions, I finally found the 'glmnet' package worked the best for meeting my needs.[7] Produced out of Stanford University it provides automated LASSO regularization coupled with built in cross validation in a very fast, easy to use package that integrates well with caret. I would highly recommend it to any one for other use cases.

As to selection of features, userId and movieId were of course obvious choices leading to a base equation of the form:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

Most of my regressions were run on this model form.

However, when I decoded the timestamp unix/posix data I also realized that 14 years of cumulative data had been collected during that time, likely from some of the same users reviewing further films at later dates. Might not that signal a potential sign of variance or change in preference over time that could contain useful information?



Thus I did a bit of feature engineering, first creating a 'date' column in the edx data.frame where time stamps were converted to standard date() format.

Then, from there using cut() I binned them into numbered bins according to a 40 day range called 'date_cut'. For one, I felt that the granularity of a single day was not that important– People generally don't change their view points from day to day, and further it places more weight on the model processing procedure. While there is only *one* unique film, and *one* unique day, there is nothing really distinctive about 'one day or another'.

At that point, still not getting the RMSE results I sought, I was about to try carretEnsemble to perform ensemble models to improve my results. While I did not get to use it in this project, I would surely recommend to anyone to look into. It is streamed lined as caret, well documented and easy to use.

In addition it provides multiple ways that ensembles can be performed, either in a list, or all together at once with even a final regression supplanted on top to tie the entire ensemble together.

My only hesitation actually came about, running out of time, I discovered the 'recosystem' package[8]. Unlike

[7] https://glmnet.stanford.edu/articles/glmnet.html
[8] https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html

other packages that provide collaborative filtering and other methods, this is pure Matrix Factorization. And it is fast. *really fast*

For all that time spent running regressions for days and hours, it ran on the *entire* edx set, barely using any RAM (comparatively) in about only 4-5 minutes with a resulting .8 < RSME on the final hold out set.

I must admit, I was quite blown away and that goes back to my earlier comment on making sure, as a Data Scientist, you are using the right tool for the job. One size does *not* fit all, though that comes with experience.

Results

Here I am posting my final RSME results. However, for the sake of consistency I am posting the results from the code I am including with the project with which I have selected an abysimally small 30,000 sample size.

This can easily be changed. On my machine I was running samples typically in the 250-300k range (which produces substantially better results), but that would take forever if anyone wanted to try.

User	User_Movie	User_Movie_Date	Recosystem
1.11348	1.04127	1.022325	0.7806992

Conclusion

There is and old adage which states: 'If all you have is a hammer, everything you see is a nail'. As we have seen this applies just as well to Data Science. While many such projects may in fact benefit from using simpler models such as various forms of regression rather than say, a complicated neural network, as we have seen in this case the use of the 'right tool'– in this case Matrix Factorization, can make a whole world of difference.

As a suggestion, which I have not implemented and thus is only briefly covered here, it would be interesting to stratify different subsets of the population and examine more closely things such as which films are they watching? Does their number of reviews correlate either to the films they've seen or their average rating? Further, how might such an analysis further help us in making successful predictions?

The hope is that one might grow to learn and know more and I appreciate the challenge and the experience.