Recommender System: Personality Theory Insights on the MovieLens Dataset

Harvard Data Science Professional Certificate Capstone Project

Klebert Toscano de S. Cintra

Introduction

Overview

Businesses in competitive markets are constantly seeking for tools that might be an advantage over competitors. That is often related to the ability to anticipate preferences of their customers and offering products that the client will like, and not regret after. Also typical of competitive markets is the abundance of options to be chosen, and while this might seem like a desirable aspect of customization for the client, research[1] has demonstrated that it makes the decision process more stressful, time consuming and, after the choice has been made, it is perceived as less satisfying. Providing good recommendations can attenuate the problems with cognitive overload, so there is high demand on the market to use data to guide the recommendations.

Here we try two different approaches to make recommendations using data of a very competitive market: the movie industry.

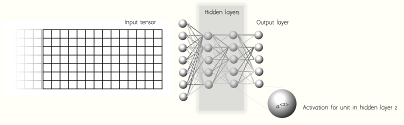
Executive Summary

Among the many methods for recommender systems a very popular one due to its simplicity, interpretability and having low computational demands is the search for a function that best describes the relationship between two or more variables and by doing so, make predictions. This is called a **Linear Model**. The weights of the variables to be combined are then estimated by minimizing the distance between the observed data and the line generated by the function. This is the Least Squares method and the estimates for the weights for the variables are the **Least Squares Estimates (LSE)**.

The other method applied here makes use of **Deep Neural Networks**, which are very popular due to their applicability to many types of data. Unlike **LSE**, it can be quite computationally demanding, and not easily interpretable. They are often represented using units (neurons) that are aggregating functions, organized in layers that are able to assign numeric value to different levels of abstraction on the input data. The number of layers constitutes the *depth* of the neural network.

The **Least Squares Estimates** method describes how a random variable we want to predict Y is defined by the linear combination of the independent variables x_i given the weights β and some random error ε:

$$Y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i x_i + \varepsilon_i,\ i = 1, \ldots, N.$$



THE GOAL of this project is to compare these two approaches in a large dataset to create a recommender system. The data used here is the *MovieLens Dataset* consisting of 10 million ratings. The recommender system implemented here is inspired by the famous Netflix Prize, which awarded the winners 1 Million Dollars for getting the **Root Mean Squared Error (RMSE)** of **0.8572**. We will compare different methods using this metric.

THE 10M MOVIELENS DATASET has ratings by 69878 unique users for 10677 movies. The ratings consist in assigning a number of stars compatible with the appreciation of the user for a particular movie. The first lines of the data frame can be seen below with a short description of variables to the right.

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

THE VARIABLES

userId: unique identifier for user.

movieId: unique identifier for movie

rating: how a user rated a particular movie.

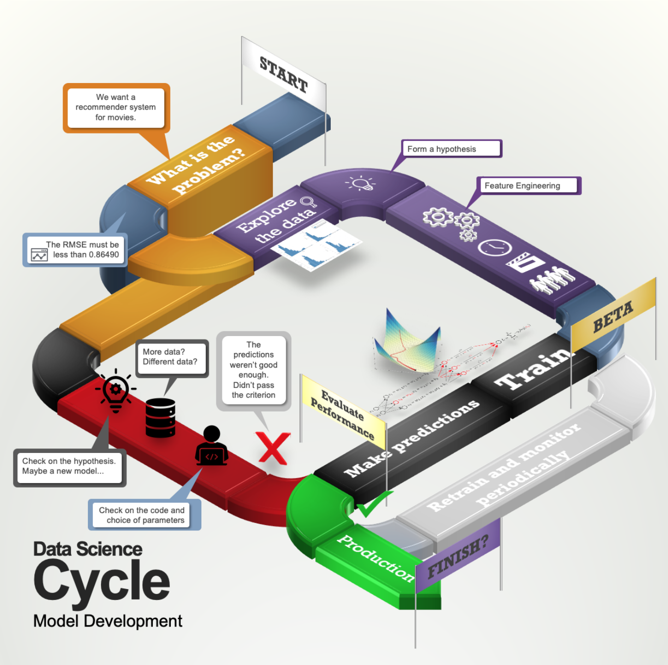
timestamp: the time when the rating happened in seconds since 01/01/1970.

title: the title of the movie.

genres: the genres that describe the movie.

The procedure proposed here includes the following parts:

1. Data acquisition and cleaning. Download and partition of the data with 10% of observations for validation and 90% for training of the algorithm.
2. Exploratory Data Analysis (EDA) and Visualization.
3. Data transformation for the Linear Models.
4. Modeling approach 1 – Linear Models.
5. Data transformation for the Deep Neural Network.
6. Modeling approach 2 – Deep Neural Network.
7. Evaluation of models and comparison of results.
8. Conclusion and final considerations.



Infographic of the framework adopted for this recommender system. Depicts the cyclical nature of the model development that ultimately might go to deployment/production.

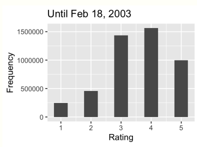
Exploratory Data Analysis (EDA) and Visualization

This report will focus on the relevant visualizations for the methods of choice, and is not an exhaustive assessment of the data.

We start with the value we want to predict, which is the *rating* that a user would give to a movie. The ratings range from 0.5 stars to 5 stars in steps of 0.5, resulting in 10 different possible ratings.

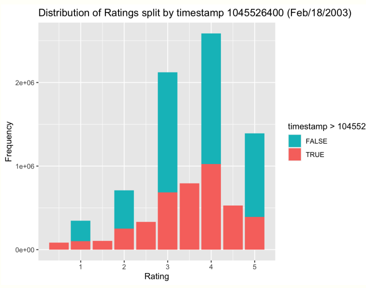
Distribution of the ratings for all users in the *edx* partition, the training set.

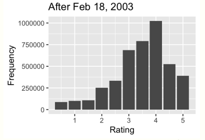
The *timestamp* is the count of seconds since the January 01 of 1970. Visual inspection of the dataset shows a change in the way the ratings are distributed. Until February 18 of 2003 (timestamp = 1045526400) the ratings are only integer numbers. After that moment, instead of 5 possible ratings (1 to 5) there are 10 different possibilities. If we repeat the plot of frequency for ratings with the bars colored based on this particular timestamp we have clear indication this is a relevant aspect to the value we want to predict, and for that reason, this should be taken into consideration.



---

[1] This is a well documented psychological phenomenon. To know more read about the jams experiment and this meta-analysis on choice overload. Barry Schwartz's book The Paradox of Choice also covers the subject nicely.

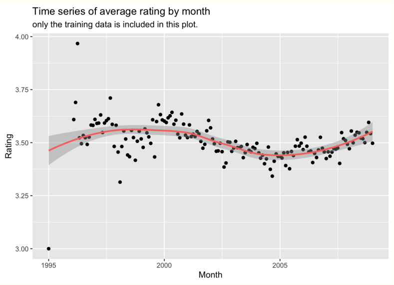




Upper plot shows the ratings distribution before timestamp **1045526400** (corresponding to the date 02/18/2003), with granularity of 1. The lower panel shows the distribution after this timestamp, with granularity of 0.5.

Left figure: Distribution of the ratings for all users in the *edx* partition, before and after February 18 of 2003.

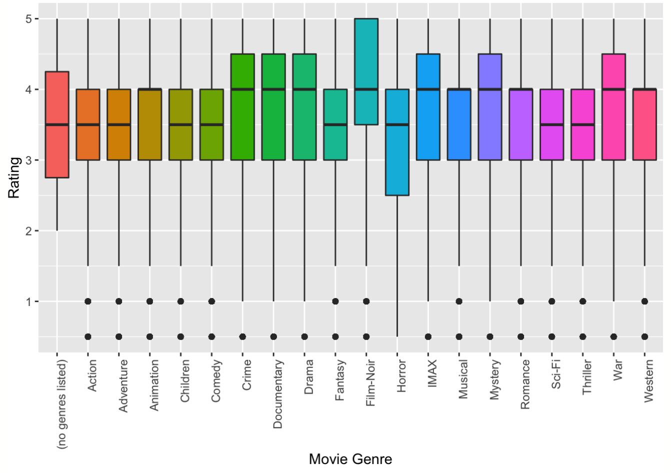
In addition to this labeling difference dependent on time, we can look for other time-dependent effects. See the picture below and observe the average ratings by month seem to vary slightly. Observe also the clear anomaly in ratings for the early years, showing much higher rating average by month than the remaining time window.



Time series: average of ratings for all users per month

As previously shown in *table 1*, numeric values are used as unique identifiers for users and movies, but movies also have a variable for the movie title, which include the year of release between parenthesis. This can be used in future analysis, but for the scope of this project we will not consider characteristics of the movie title or the year it was released.

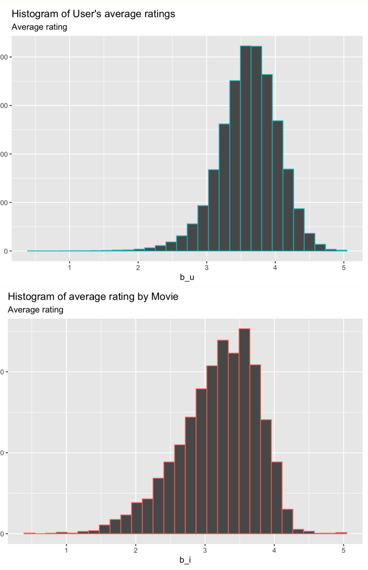
The key variable to be examined is the *genres* column of the dataset. It is a way of describing movies implying a relationship in a higher level of abstraction between movies. It consists of strings with all the genres in which a movie can be classified separated by the character "|". Here we can see the boxplots for the aggregated ratings on all the listed genres.



Boxplots of ratings grouped by genre.

There's no obvious difference between genres to make a prediction based on this dimension alone.

To finish this exploratory data analysis we check on the distribution of user rating averages and the average for movies.



Histograms of average ratings for user (top) and for movies (bottom).

Data Transformation for the Linear Model

The transformations applied to data are based in the following assumptions:

1. Movie genres provide valuable information about abstract dimensions of the movies. A genre usually have a typical pace of the story, the content of the scenes, saturation of colors, angles for the cameras, emotions provoked, the of casting of actors, and all of those influence the expectations of the audience and might interfere with the ratings even before a movie is watched. So, it could be a less computationally expensive way to numerically describe a movie[2].

2. The evaluation a user provides to a single movie has little generalization for other movies, but the consistent rating of a user for a genre (be it positive or negative) describes the subject[3].

3. The more genres a movie has, the more eclectic it might be. Hence, it might attract a larger audience, which result in more ratings, and as a consequence, greater predictive value.

4. The label to be predicted is not a continuous value ranging from 0 to 5. Instead, they are categories that depend on the timestamp variable, as demonstrated in the pictures above.

Genre Transformation for the Linear Model

The proposed procedure is to replicate each line for a "userId" + "movieId" pairing for each unique genre this movie has as attribute.

Let's use as example the first line of the original dataset. We can see there is one pairing of userId = 1 and movieId = 122:

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

After the transformation, this pair (userId = 1, movieId = 122) will be featured twice, for having 2 genres for this movie: "Comedy" and "Romance".

userId	movieId	rating	genre
1	122	5	Comedy
1	122	5	Romance
1	185	5	Action
1	185	5	Crime
1	185	5	Thriller
1	231	5	Comedy

The variables *timestamp* and *title* were removed as they are not used for this analysis.

Modeling approach 1 - Linear Models.

This class of models is based on the premise that the response variable $Y_{u,i}$ representing the rating user u assigned to movie i is equal to a theoretical film rating described by the linear combination of some random variables. The model definition is:

$Y_{u,i} = \mu + b_i + b_u + \sum_{k=1}^{K} x_{u,i}\beta_k + \varepsilon_{u,i}$, with $x_{u,i}^k = 1$ if $g_{u,i}$ is genre k.

where:

μ = The mean of all ratings.

b_u = The user effect.

b_i = The item effect, in this case, the movie.

$g_{u,i}$ = Genre for user u rating of movie i.

$\varepsilon_{u,i}$ = The error. Randomness on the data, noise in the system.

$\sum_{k=1}^{K} x_{u,i}\beta_k$ = The summation of all the genres' effects for that movie-user

[2] Matrix Factorization using **Principal Component Analysis** and other transformations on this dataset made necessary more than 2000 *principal components* to explain 90% of the variability on data. Hence, for each prediction on the pairing User-Movie a 2000x2000 vector should be multiplied, and the estimation for 1 million data points would be costly with common personal computers' CPUs.

[3] If an individual shows a consistent preference or response that is stable over time and different contexts is considered a **trait**. Trait psychology and the study of individual differences can be applied to the creation or refinement of recommender systems. The book Hierarchical Cognitive Models has many examples of applications for inferences on latent traits.

combination.

Our goal is to find the βs to this equation in order to minimize the error. The error is the difference between the predicted value and the actual rating collected. To quantify this error we use the square root of the difference (or distance), which penalizes more the farther away the prediction is from the real data. Added a penalization term the equation for the *Root Mean Square Error* is:

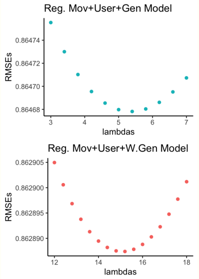
$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

where $y_{u,i}$ is the rating for movie i by user u and denote our prediction with $\hat{y}_{u,i}$. The penalty term λ limits the total variability of the effect sizes, lest the model incorporates random variability unrelated to the theoretical variables we want to describe and generalize with our model.

We now compare the application of this model with added genre weights and without it in the table below.

model	RMSE
Mean-Baseline Model	1.0606506
Mean-Baseline Model with Weighed Genre	1.0524433
Regularized Movie+User+Genre Based Model	0.8646782
Regularized Movie+User+Weighed Genre Based Model	0.8628874

The difference in **RMSE** points to the predictive advantage in applying weights relative to the genres.



Optimization of the Regularization parameter for the Linear models without added movie genre weights (top) and with it (bottom).

Data transformation for the Neural Network.

For the second part of this analysis, we will use a *Deep Neural Network*. Several approaches are described in the literature and they are proved successful, of which one of the most popular might be the winner of the Netflix challenge that used Restricted Boltzmann Machines as one of the algorithms.

The output of the network could be a real number representing the rating of the user. But it is not possible, for example, a user to rate a movie 3.17 stars. That means the rating process can be best described as a classification task with multiple classes, and for that the *rating* values must be transformed to factors or categories.

Genre Transformation

What is proposed here is that instead of feeding the neural network just the user and ratings, and expecting the appropriate relationships to be derived automatically, we first transform the data in order to give a multidimensional description of the user based on their affinity or preference for the genre. This is based on personality psychology models that define traits as factors related to human behavior, which are derived from responses to questionnaires[4]. If a user can be described by the consistent of behaviors, then we can create a vector that describes a *userId* using their average *rating* for movies of a *genre*.

Timestamp Transformation

Because there is a difference in the number of values for ratings (classes) dependent on the time that rating was given, the predictions will be more accurate if the timestamp is informed. Nonetheless, there's no need to know the exact moment in time that rating was given. For that this variable will be transformed to be 0 or 1 assigned to a variable named *timestamp_binary*, which should be enough to determine if the rating is from a moment before or after the change in classes.

The first columns of the final dataset are:

userId	movieId	rating	Action_u	Adventure_u	Animation_u
1	122	5	2.380952	1.190476	0.7142857
1	185	5	2.380952	1.190476	0.7142857
1	231	5	2.380952	1.190476	0.7142857
1	292	5	2.380952	1.190476	0.7142857
1	316	5	2.380952	1.190476	0.7142857
1	329	5	2.380952	1.190476	0.7142857

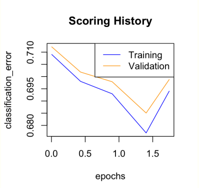
Modeling approach 2 - Deep Neural Network

The neural network used here is a fully-connected neural network implemented by the H2O package. The input data is a matrix of size M (total number of ratings = 9000061) by N (the columns: userId, movieId, 19 columns being one per genre, and the timestamp_binary that signals the change in output labels = 22).

To calculate the conditional probabilities for each rating the *Softmax* function will be used in the last layer of the network. The loss function used here is the automatically chosen based on the type of label on the validation data. Dropout (0.2) and early stopping were implemented to prevent overfitting. For more information, read the documentation.

Evaluation of Models and Comparison of Results

The table below shows the performance difference regarding errors in the predictions for all models presented here. Though the training of the neural network was more than 10 times longer than the linear models the performance was also much better. This particular implementation had even a better performance than the one mentioned as the motivation for this project which had **RMSE = 0.8572.**



Classification error for training set and validation set over epochs.

model	RMSE
Mean-Baseline Model	1.0606506
Mean-Baseline Model with Weighed Genre	1.0524433
Regularized Movie+User+Genre Based Model	0.8646782
Regularized Movie+User+Weighed Genre Based Model	0.8628874
Deep Neural Network	0.8312306

Conclusion and Final Considerations

Deep Neural Networks are capable of modeling highly complex relationships between variables if properly structured. The risk of overfitting can be managed via the proper tuning of parameters and in the case of the implementation described here, the performance is better than the linear models that used the same variables.

Knowledge of other areas, especially related to the problem at hand , is much valuable when creating hypothesis, defining models and transforming the data. If implementation expertise and final application of the solution can be combined the process of creation of models and interpretation of outputs will be more efficient.

---

[4] Popular personality theories based on this assumption include the Five Factor Model by Costa and McCrae, and the work of Goldberg, and DeYoung to name a few.