Movie Project

piRsquare

11/18/2019

Executive summary

The objective is to propose a prediction for movie ratings, using machine learning techniques. The original data is based on the 10M movielens file available at https://grouplens.org/datasets/movielens/10m/

The data provides 10M ratings for 10.6K movies, done by 70K users. For this project, the data is initially split into 2 sets, the **edx** with 90% of the records, to train the prediction model; the **validation** has 10% of the records and it is used to assess the model using RMSE (root mean square errors) between the model prediction and the actual rating found in the validation set.

To achieve a maximum of 25 points, the RMSE must be below 0.8649. This paper proposed a model trained using regularization and cross-validation techniques, and that has an RMSE of 0.8636, better than the 0.8649 target.

Methods and Analysis

Variable definitions and initial exploration

As mentioned in the introduction, the original dataset is used to generate a file edx, and validation file. For training and testing the model, we split the edx into a train set with 8.1 M ratings, and a test set with 0.9M. The validation file is reserved for the final check of the RMSE of the trained model.

All files contain the following variables:

Variable	Definition	Comments
userId	Integer that provides a unique identification for a rating user	70K users, top user rated 6.6K movies, lightest user rated just 10
movieId	Number that provide unique identificationfor a movie	10.6K movies, highest was rated by 31K users, lowest rated by 1
rating	Numbers in 0.5,1.1.5, . . . ,5.0	Average 3.5, most popular rating is 4 in 29% of cases
timestamp	Integer that provides time of the review in seconds since January 1, 1970	Not used in our model; ratings stamped between Jan 1995 to Jan 2009
title	Character	Movie title with year of release
genres	Character	One of more movie genres separated by a pipe. Not used in our model

Some unsuccesful explorations

Given the size of the dataset, a key concern was to produce a model that would not crash in our laptop. A first idea was to build individual prediction models for each user. An exploration was started by extracting the data for the highest rater and trying to fit different machine learning models. One user was fine, but repeating it for many was a challenge since the computer would crash. Additionally, it was difficult to find a simple way to integrate a large number of user-models into one to make predictions. We abandoned that

path, for this project.

Another approach that failed was to separate the genres, and try to explore a model that would include them as additional refinement. We managed to separate the genres for the top user, but R crashed when trying to do it for ALL the users. After these failures, we tried a different approach which is explained in the next section.

Mathematical background for Regularization

We try a reagularization approach similar to the one explained in the textbook but with one useful enhancement. At the end of section 33.9.3, the author proposes a code to estimate movie and user bias parameters in two-steps. The movie bias is estimated first using the formula for a model with only one type of bias. Then, the user bias is estimated second, using again the same type of formula and the movie bias found in step 1.

As we know for calculus, in a problem with two variables we might do better if we optimize on both variables simultaneously, than if we optimize on one varible first, fix it, and then optimize the second variable. We use calculus to derive the correct formulas, based on the model:

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

where for user u and movie i, the rating $Y_{u.i}$ is a random variable that depends on the overall average rating μ, a movie bias b_i, a user bias b_u, and the residual is an error $\epsilon_{u,i}$

To regularize, we propose the formula below, with a penalty factor λ. Note that there it has a slight difference with repect to the textbook, there is no $1/N$ in the first sum.

$z = \Sigma_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\Sigma_i b_i^2 + \Sigma_u b_u^2)$

For a movie I,

$\partial z/\partial b_I = 0 = -2\Sigma_{u,I}(y_{u,I} - \mu - b_I - b_u) + 2\lambda b_I$

which yields

$\hat{b}_I = \frac{1}{(\lambda + n_I)}\Sigma_{u,I}(Y_{u,I} - \hat{\mu} - \hat{b}_u)$, where n_I is the number of ratings for movie I, and the sum is done over users u who rated movie I

For a user U,

$\partial z/\partial b_U = 0 = -2\Sigma_{U,i}(y_{U,i} - \mu - b_i - b_U) + 2\lambda b_U$

which yields

$\hat{b}_U = \frac{1}{(\lambda + n_U)}\Sigma_{U,i}(Y_{U,i} - \hat{\mu} - \hat{b}_i)$, where n_U is the number of movies rated by user U, and the sum is done over movies i rated by user U

The two type of equations are partially coupled, the movie bias depends on the user biases, and the user bias depends on the movie biases. So we solve them using an iterative process.

1. We initialize the calculation of movie biases, assuming the b_u's are zero
2. We used the movie biases b_i's found and the second formula to calculate the user bias b_u
3. Then we use the b_u's found in the first formula to re-calculate the movie biases b_i's
4. We iterate 2 and 3, until we have a solution that has converged close to the actual solution.

Cross-validation and Iteration

Now we have two paramerers to find.

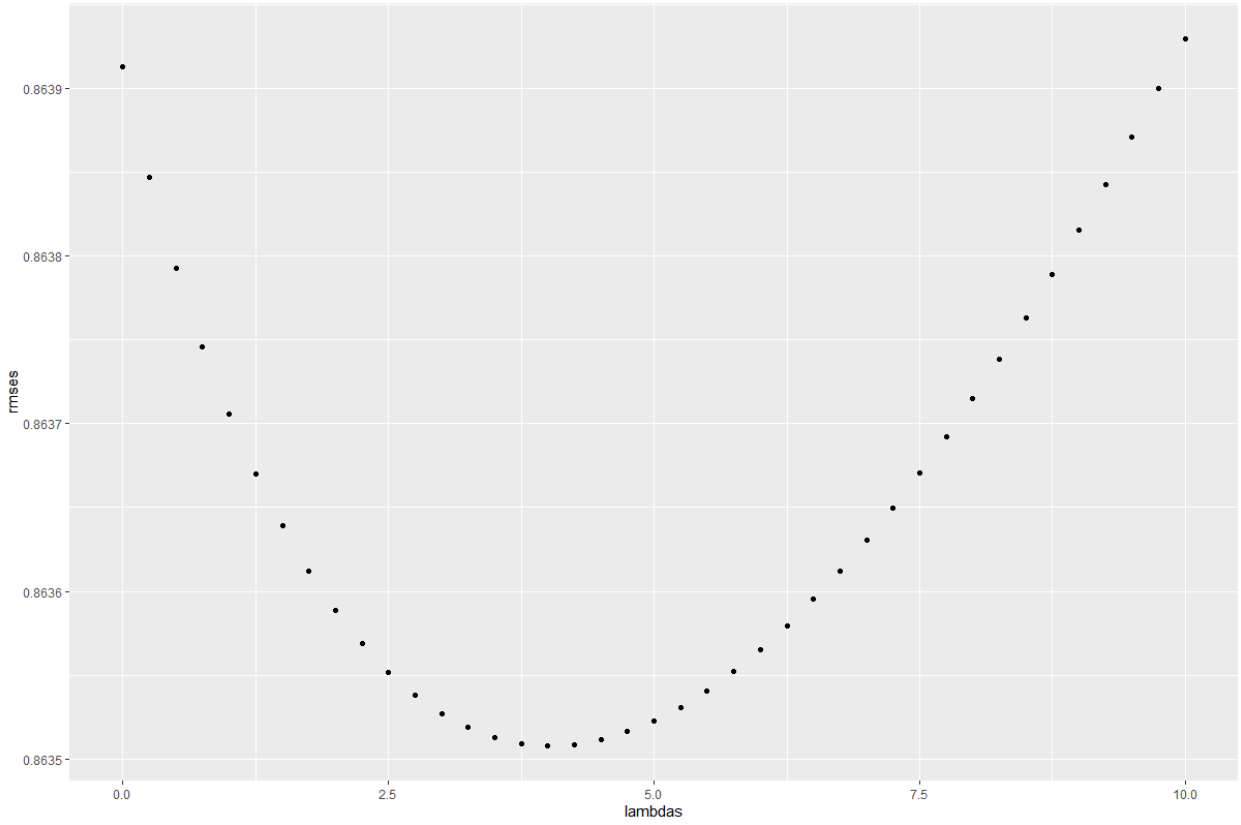
- We need to find the parameter λ that gives us the best penalty to minimize the RMSE. We will use cross-validation
- We also need to determine how many times we iterate to solve the linear equations used to calculate the movie and user bias. We will use manual exploration

We start by splitting the set edx randomly into a train set and a test set. The train set has 8.9 million movie ratings, and the test has 0.9 M. Special care was done to check that for all records in the test set, the movie is also in the train set (rated by at least one other user so we calculate movie bias), and the user too (rating at least one another movie so we can calculate user bias)

For λ we cross-validate in the range 0 to 10, in steps of 0.25. For the number of iterations, we run the same code multiple times by changing the number of iterations. The following table summarize the optimal values found for the RMSE when the model is applied to predict the results in the test set

Iterations	RMSE Test Set	Best λ
0	0.86495	4.75
1	0.86393	3.75
2	0.86356	4
3	0.86351	4
4	0.86349	4

We can see that we need at least 2 iterations to identify the correct $\lambda = 4$, and that the more we iterate, the closer our b_u, b_i's are to the optimal ones, at the expense of increasing computational time. Here is the cross-validation chart for 3 iterations, showing the optimum $\lambda = 4$.



After a few experimental manual runs, we set 3 iterations in the cross-validations used to find λ. Once λ was

found, we increased to 10 iterations to find biases closer to the optimum for our final prediction model. These apporach reduced the burden of calculations during the time-consuming cross-validation, and increased it when we needed more precision.

Results

The predictions were validated on the validation set, with a RMSE of 0.8636, 1.5% better than the 0.8649 target.

The script takes approximately 12.5 minutes to run from start to end. About 4 minutes are used to run the code to download the files and create the **edx** and **validation** sets; 8 minutes to complete the cross-validations, and 0.5 minutes to calculate the prediction model and apply it to the Validation set.

The run was done in a Lenovo Yoga laptop, with Intel Core i7, 2.8Ghz, 16 GB of RAM, y 450 GB of hard drive (80% free disk space).

Conclusions

The main learning from this project was that we have to find a balance with what the computer at hand could do for you with a data set of this size. The challenge was to come with an approach that would run in a reasonable time, and that would have an RMSE performance to meet the target for the grading criteria. My laptop provided practical physical constraints, and a few ideas that might have potential could not be implemented because R would crash.

The second learning is a confirmation of the power of regularization and cross-validation. Both of them contributed to achieve the target, and to give confidence the value chosen for lambda was the best available.

The third learning is the importance of having clarity about your objective and to remain focused to achieve it. This project was a reminder that *the perfect is the enemy of the good*. I started with the objective of looking for the best solution and dreaming about finding something better than the Netflix winning team. There are too many models and methods to try, and it takes and enormous amount of time to try, especially when a few make your computer crash. Hence, I changed my target to have a performance to get full credit, and had fun and learned a lot by doing it.

In terms of potential future enhancements, the model used can be expanded including factors for the movie genres. That approach represented some computational challenges to run without crashing R in the hardware available. Most likely, it might require to decompose the data into smaller subsets, and then integrate the models found in those subsets. That requires additional research.