MovieLens Project

Tianying Jiang

5/18/2019

R Markdown

This is an R Markdown document to ducment the project MovieLens. The instruction of the project listed below:

Your submission for this project is three files: * Your report in PDF format * Your report in Rmd format * A script in either R or Rmd format that generates your predicted movie ratings and RMSE score To upload and submit your files press the "Choose Files" button, select three files at once (using the control key on a Windows machine or command key on a Mac) and press "Choose," type a description for each, and then press the "Upload files" button.

We recommend also providing a link to a GitHub repository containing the three files above.

Note that when downloading files for peer assessments, R and Rmd files will be downloaded as txt files by default.

Overview of the training set

First, let me assume that you have ran the codes provided at the beginning of this project and did the quiz in the previous section. I only include the pre-povided codes in the script and .rmd file, but would not provide in the pdf file. Let's just further review the training data set *edx* as below.

Let's quickly have a look at the training set again using:

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

It is clear that we might want to consider the factors of movie, user, time, pereference of genres (based on specific user). So let's do that step by step as in pervious course.

Simplest model based on average of the rarting

Assuming that each individual rating can be estimated as a universal number true nuber with a random error (which is based on different user (u), movie (i), rating date (d) and genres (g)) as the following format:

$$Y_{u,i,d,g} = mu + e_{u,i,d,g}$$

To minimize the residual mean square error, we know that we should estimate mu as the average of the rating in the training set using the following code:

```
#Calculate the average of the rating and use the simplest based on average
mu<-mean(edx$rating) #calculate the average across the training set
rmse_0 <- RMSE(validation$rating, mu) #cacluate rmse
rmse_results <- data_frame(method = "Just the average", RMSE = rmse_0)
#create a dataframe to save/compare results
rmse_results
```

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

Including the movie effects

After we have consider the average of rating, we want to include the effect of individual movie as we know some movies are generally considered good despite being rated by different user, and some are considered bad. Therefore we include the term b_i in the model:

$$Y_{u,i,d,g} = mu + b_i + e_{u,i,d,g}$$

b_i is the term for the average rating effect based on movie, with i as the movie index. To minimize the residual mean square error, we know that we should estimate b_i for different movies as the average of error as for different movies:

```
#include the movie effect and compare with previous model
movie_avg<-edx%>%group_by(movieId)%>%summarize(b_i=mean(rating)-mu)
predicted_ratings_1 <- mu + validation %>%
  left_join(movie_avg, by='movieId') %>%
  pull(b_i)

model_1_rmse <- RMSE(predicted_ratings_1, validation$rating)
rmse_results <- bind_rows(rmse_results,
              data_frame(method="Movie Effect Model", RMSE = model_1_rmse))
rmse_results
```

```
## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Just the average   1.06
## 2 Movie Effect Model 0.944
```

Including the user effects

Now we want to include the effect of individual user as we know some users are more "strict" while some are not. Therefore we include the term b_u in the model:

$$Y_{u,i,d,g} = mu + b_i + b_u + e_{u,i,d,g}$$

b_u is the term for the average rating effect based on user after we deduct the effect of movie, with u as the user index. To minimize the residual mean square error, we know that we should estimate b_u for different movies as the average of error as for different users:

```
#include the user effect and compare with previous models
user_avg<-edx%>%left_join(movie_avg, by='movieId')%>%
  group_by(userId)%>%summarize(b_u=mean(rating-mu-b_i))
predicted_ratings_2<-predicted_ratings_1+validation%>%
  left_join(movie_avg,by='movieId')%>%left_join(user_avg,by="userId")%>%
  pull(b_u)
model_2_rmse<-RMSE(predicted_ratings_2,validation$rating)
rmse_results<-bind_rows(rmse_results,data_frame(method="Movie+User Effect Model",
                                                RMSE=model_2_rmse))
rmse_results
```

```
## # A tibble: 3 x 2
##   method                    RMSE
##   <chr>                    <dbl>
## 1 Just the average         1.06
## 2 Movie Effect Model       0.944
## 3 Movie+User Effect Model 0.865
```

Including the time effects

Now we want to have a look if there is any effect of time to decide whether we want to include that term into the model using the following code:

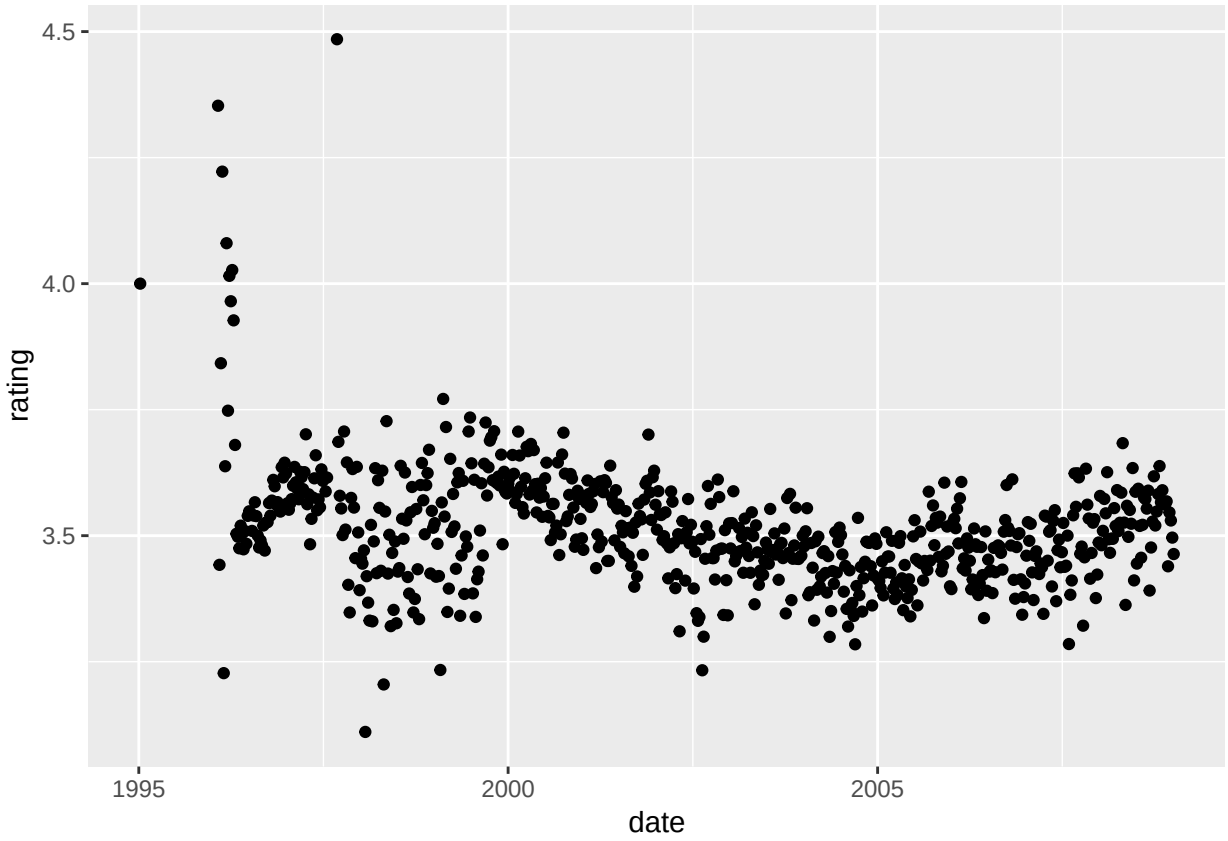
```
#check if there is any time effect
if(!require(lubridate)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
#add a new data frame edx_1 to include date info
edx_1<-edx%>%mutate(date=round_date(as_datetime(timestamp),unit="week"))
#check the trend of rating vs. date
edx_1%>%group_by(date)%>%
      summarize(rating=mean(rating))%>%
      ggplot(aes(date,rating))+
      geom_point()
```



We can see clearly there is some trend. Therefore, we want to include time effect f_d (f(date)) which is a function of the date.

$$Y_{u,i,d,g} = mu + b_i + b_u + f(d) + e_{u,i,d,g}$$

Unlike the term for movie effect and user effect, time is a continuous parameter, and we can simply use the local weighted regression to predict this term.

```
summary_edx_1<-edx_1%>%left_join(movie_avg,by='movieId')%>%
  left_join(user_avg,by='userId')%>%
  group_by(date)%>%
  summarize(date_eff=mean(rating-mu-b_i-b_u),timestamp=first(timestamp),n_rating=n())%>%
  data.frame()%>%
  mutate(date_eff=ifelse(n_rating<10,0,date_eff))
fit_date<-loess(date_eff~timestamp,degree=1,span=0.05,data=summary_edx_1)
date_avg<-data.frame(date=summary_edx_1$date,f_d=fit_date$fitted)
validation_1<-validation%>%mutate(date=round_date(as_datetime(timestamp),unit="week"))
predicted_ratings_3<-predicted_ratings_2+validation_1%>%
  left_join(date_avg,by='date')%>%
  pull(f_d)
model_3_rmse<-RMSE(predicted_ratings_3,validation$rating)
rmse_results<-bind_rows(rmse_results,data_frame(
  method="Movie+User+Time Effect Model", RMSE=model_3_rmse))
rmse_results
```

```
## # A tibble: 4 x 2
##   method                          RMSE
##   <chr>                          <dbl>
```

```
## 1 Just the average              1.06
## 2 Movie Effect Model            0.944
## 3 Movie+User Effect Model       0.865
## 4 Movie+User+Time Effect Model 0.865
```

Actually we do not see much improvement after including the time effect. This might be explained as correlation of the time effect with user or movie effect. Therefore we see minor improvment when adding that term.

Including the effect of genres interest of different users

As we kmow, different users might like different type of movies, therefore we want to add a new term $genres_{eff}$ which is a function of user and genres. As we seem from the training date set, most movies can have more than one genres. To esimate that effect, it might be easier to spread out the data and estimate the effect based on individual user and genres. Then we we predict the rating of the test set, we can simply estimate the effect based on the average of the effect when one movie has fallen into different category.

$$Y_{u,i,d,g} = mu + b_i + b_u + f(d) + mean(genres_{(eff)u,g}) + e_{u,i,d,g}$$

```
#study the effect of genres
#First deduct the effect of avg, movie, user, and date, then group by user,genres
genres_summary<-edx_1%>%left_join(movie_avg,by='movieId')%>%
  left_join(user_avg,by='userId')%>%left_join(date_avg,by="date")%>%
  mutate(genres_eff=rating-mu-b_u-b_i-f_d)%>%
  separate_rows(genres, sep = "\\|")%>%group_by(userId,genres)%>%
  summarize(genres_eff=mean(genres_eff))%>%data.frame()%>%
  spread(genres,genres_eff,fill = 0)%>%
  gather("genres","genres_eff",-userId)
genres_avg<-validation%>%separate_rows(genres, sep = "\\|")%>%
  left_join(genres_summary,by=c("userId","genres"))%>%
  group_by(userId,movieId)%>%summarize(genres_eff=mean(genres_eff))
predicted_ratings_4<-predicted_ratings_3+validation%>%
  left_join(genres_avg,by=c('movieId',"userId"))%>%pull(genres_eff)
model_4_rmse<-RMSE(predicted_ratings_4,validation$rating)
rmse_results<-bind_rows(rmse_results,data_frame(
  method="Movie+User+Time+Genres Effect Model", RMSE=model_4_rmse))
rmse_results
```

```
## # A tibble: 5 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Just the average                     1.06
## 2 Movie Effect Model                   0.944
## 3 Movie+User Effect Model              0.865
## 4 Movie+User+Time Effect Model         0.865
## 5 Movie+User+Time+Genres Effect Model 0.850
```

Obviously there is some improvement in rmse when we include that term.

Including the effect of regularization

As we have learnt in previous course, regularization helps us to penalize large estimate based on small sample size. So we want to include that effect. Here we want to explore the effect of different penalty value l, and

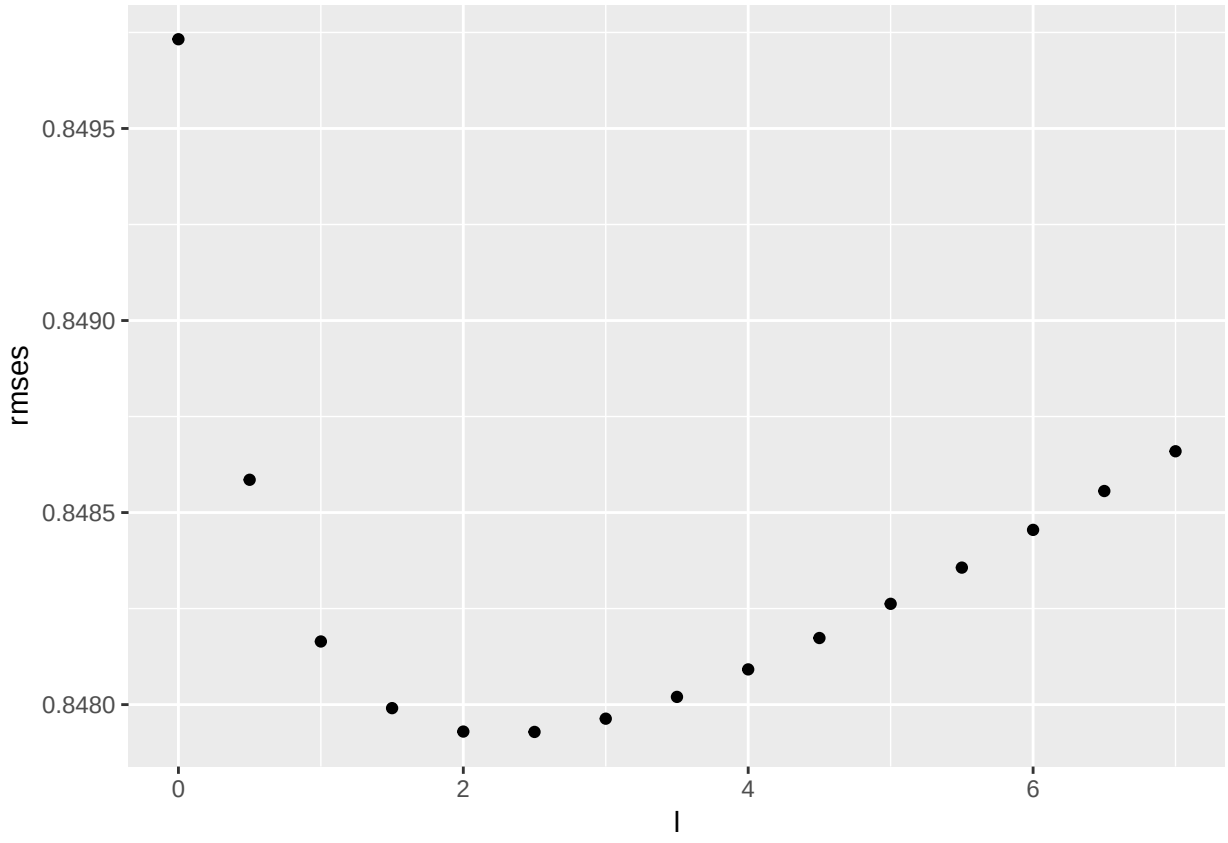
figure out which minimize RMSE. We can run the code as following. Please note that it might take some time to run this code, and sapply function appeared to be slower than using *for* loop. So I use *for* loop here and for every cycle the index i is printed so the process can be tracked.So we can simply modify the code for caculation mean value to .../(n()+l), where n() represent the sample size.

```
#regularized effect
l<- seq(0,7,0.5)
rmses<-rep(NA,15)
for (i in 1:15) {
  mu<-mean(edx$rating) #calculate the average across the training set
  #include the movie effect
  movie_avg<-edx%>%group_by(movieId)%>%summarize(b_i=sum(rating-mu)/(n()+l[i]))
  predicted_ratings_1 <- mu + validation %>%
    left_join(movie_avg, by='movieId') %>%
    pull(b_i)

  #include the user effect
  user_avg<-edx%>%left_join(movie_avg, by='movieId')%>%
    group_by(userId)%>%summarize(b_u=sum(rating-mu-b_i)/(n()+l[i]))
  predicted_ratings_2<-predicted_ratings_1+validation%>%
    left_join(movie_avg,by='movieId')%>%left_join(user_avg,by="userId")%>%
    pull(b_u)

  #include time of rating effect
  #check if there is any time effect
  if(!require(lubridate)) install.packages("tidyverse",
                                           repos = "http://cran.us.r-project.org")
  #add a new data frame edx_1 to include date info
  edx_1<-edx%>%mutate(date=round_date(as_datetime(timestamp),unit="week"))

  summary_edx_1<-edx_1%>%left_join(movie_avg,by='movieId')%>%
    left_join(user_avg,by='userId')%>%
    group_by(date)%>%
    summarize(date_eff=sum(rating-mu-b_i-b_u)/(n()+l[i]),timestamp=first(timestamp))%>%
    data.frame()

  fit_date<-loess(date_eff~timestamp,degree=1,span=0.05,data=summary_edx_1)
  date_avg<-data.frame(date=summary_edx_1$date,f_d=fit_date$fitted)
  validation_1<-validation%>%mutate(date=round_date(as_datetime(timestamp),unit="week"))
  predicted_ratings_3<-predicted_ratings_2+validation_1%>%
    left_join(date_avg,by='date')%>%
    pull(f_d)

  #study the effect of genres
  #First deduct the effect of total avg, movie avg, user avg, and date avg, then group by user & genres
  genres_summary<-edx_1%>%left_join(movie_avg,by='movieId')%>%
    left_join(user_avg,by='userId')%>%left_join(date_avg,by="date")%>%
    mutate(genres_eff=rating-mu-b_u-b_i-f_d)%>%
    separate_rows(genres, sep = "\\|")%>%group_by(userId,genres)%>%
    summarize(genres_eff=sum(genres_eff)/(n()+l[i]))%>%data.frame()%>%
    spread(genres,genres_eff,fill = 0)%>%
    gather("genres","genres_eff",-userId)
  genres_avg<-validation%>%separate_rows(genres, sep = "\\|")%>%
    left_join(genres_summary,by=c("userId","genres"))%>%
```

```
    group_by(userId,movieId)%>%summarize(genres_eff=mean(genres_eff))
  predicted_ratings_4<-predicted_ratings_3+validation%>%
    left_join(genres_avg,by=c('movieId',"userId"))%>%pull(genres_eff)

  rmses[i]<-RMSE(predicted_ratings_4, validation$rating)
  print (i) #use this to track the process
  }
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
```

```
qplot(l, rmses)
```



```
lambda <- l[which.min(rmses)]
print (lambda)
```

```
## [1] 2.5
```

```
rmse_results<-bind_rows(rmse_results,data_frame(
  method="Regularized Movie+User+Time+Genres Effect Model", RMSE=min(rmses)))
rmse_results
```

```
## # A tibble: 6 x 2
##   method                                          RMSE
##   <chr>                                          <dbl>
## 1 Just the average                               1.06
## 2 Movie Effect Model                             0.944
## 3 Movie+User Effect Model                        0.865
## 4 Movie+User+Time Effect Model                   0.865
## 5 Movie+User+Time+Genres Effect Model            0.850
## 6 Regularized Movie+User+Time+Genres Effect Model 0.848
```

So we can see that the best penalty term is 2.5, which give us rmse 0.848 (there is only a slight improvement compared to the preivous result).Nevertheless, that is the best result we have got so far after including regularized movie, user, time and genres effect.