Movie Recommendation System

yfu2015

2022-09-20

Executive Summary

1. This movie recommendation model augments the regularized modeling movie + user effects (Harvard Data Science/Machine Learning/Section 6 Model Fitting and Recommendation Systems) by further adding the **date** effects to model regularized movie + user + date effects. The **date** is defined as the week since January 1, 1970.
2. The model defines $d_{u,i}$ as the day for user's (u) rating of movie i, the recommendation model is: $Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \epsilon_{u,i}$
3. The model constructs predictors, calculates the RMSE, and determine the minimum RMSE as well as the optimal lambda

```
# install packages

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:dplyr':
##
```

```
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
if(!require(dslabs)) install.packages("dslabs", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dslabs
```

```
# Loading packages

library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(dslabs)
```

```
# MovieLens 10M dataset:

dl <- tempfile()
download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))    ##str(ratings)

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")


# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))    ##str(movies)

movielens <- left_join(ratings, movies, by = "movieId")
```

```
# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)


# Convert the 'timestamp' to date in which the rating was provided,
# and then to week in which the rating was provided

edx <- mutate(edx, date = round_date(as_datetime(timestamp), unit = "week"))
validation <- mutate(validation, date = round_date(as_datetime(timestamp), unit = "week"))


# Remove the column of timestamp
edx <- edx[, c("movieId", "userId", "rating", "title", "genres", "date") ]
validation <- validation[, c("movieId", "userId", "rating", "title", "genres", "date") ]
```

Comments

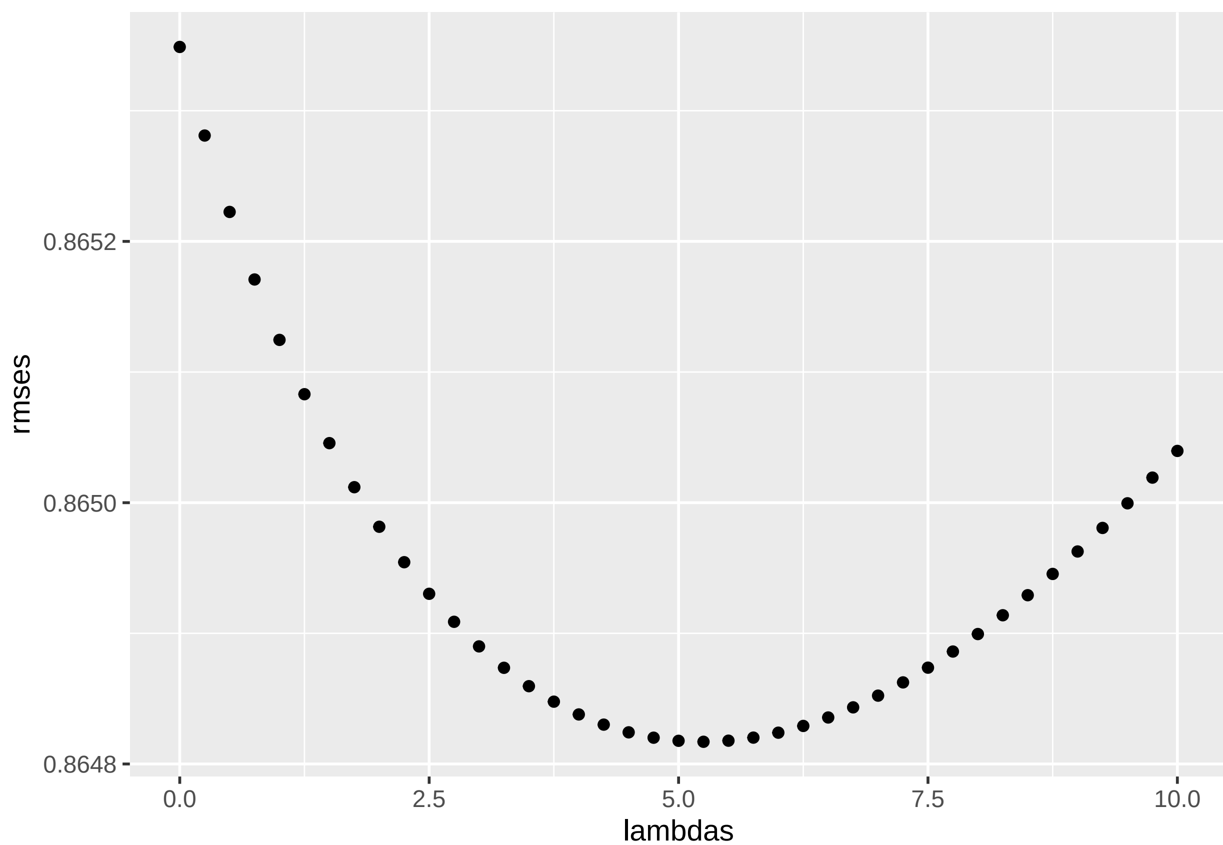
1. Before modeling movie + user + **date** effects, I will first check the RMSE of modeling movie + user effects on the data set downloaded above.
2. Regularization has applied on this "movie + user effects" model which penalized large estimates that were formed using small sample sizes.
3. Use cross validation to pick the penalized lambda and calculate the minimum RMSE. Because it is too slow to train the model as train(rating ~ as.factor(movieId) + as.factor(userId), method='lm', data=edx), I will compute an approximation by computing μ, b_i, and estimating b_u as the average of $y_{u,i} - \mu - b_i$

Reference

Modeling Movie + User Effect - Harvard Data Science Machine Learning Section 6

```
## Build regularized  movie + user Effect model

## This model will be compared to the final Modeling movie + User + date Effects

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}


lambdas <- seq(0, 10, 0.25)
```

```r
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

    return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```




```r
min(rmses)
```

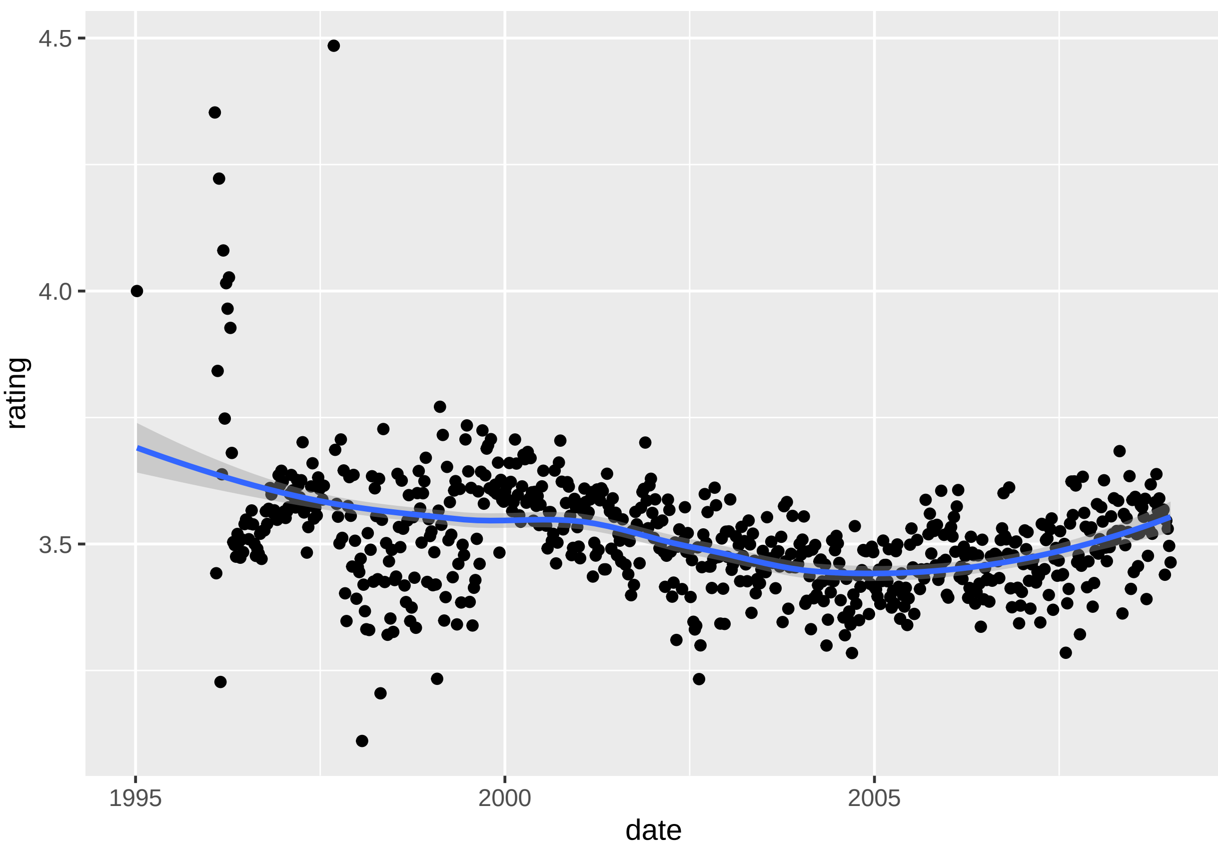
```
## [1] 0.864817
```

Observation

1. The minimum RMSE of the modeling movie + user effects is 0.864817
2. Next, plot/visualize the rating vs the date. If there is an evidence of the time effects on the rating, I will augment the model by adding the date effects and calculate RMSE.

```
# 1. Compute the average rating for each week ( the week since January 1, 1970)
# and plot this average against date

edx %>%
    group_by(date) %>%
    summarize(rating = mean(rating)) %>%
    ggplot(aes(date, rating)) +
    geom_point() +
    geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Observation

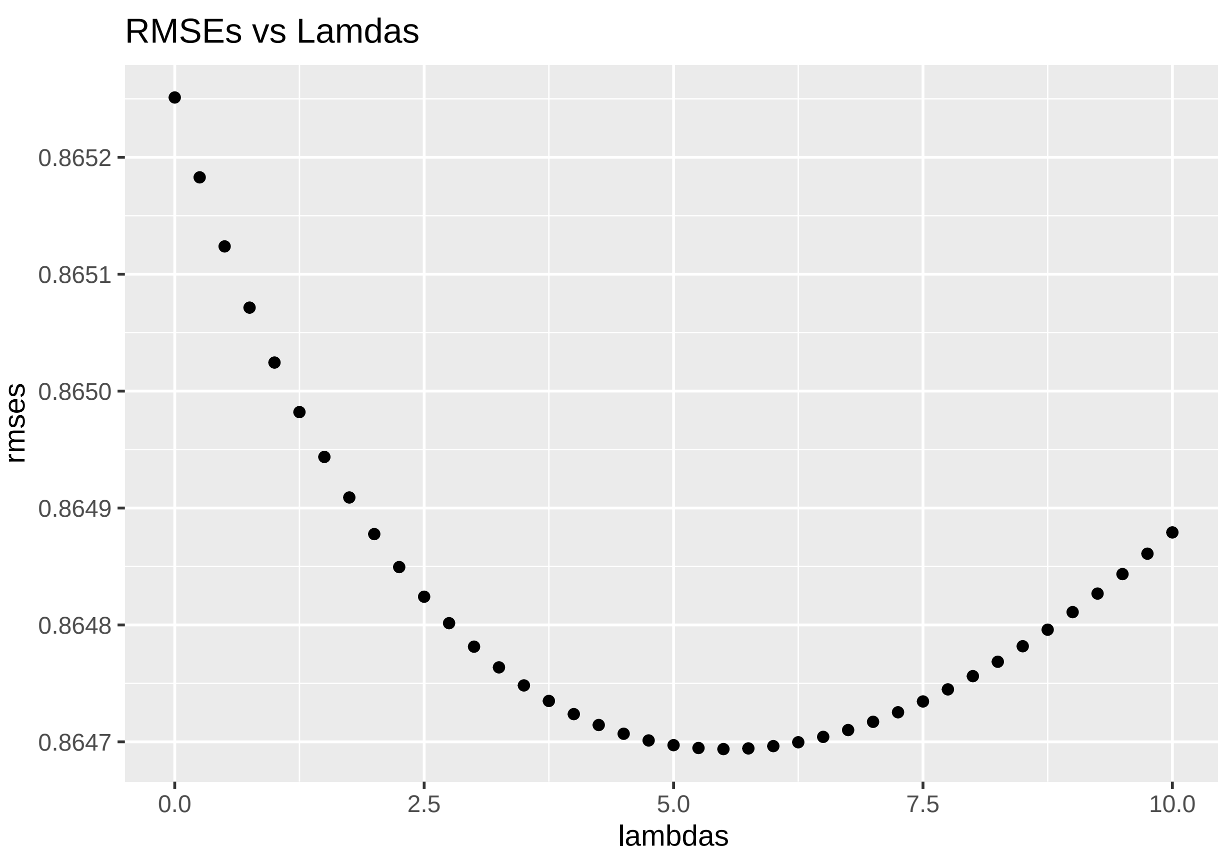
1. There is some evidence of a time effect on the rating as shown in the plot.
2. This implies that a further improvement to the model of regularized "modeling movie + user effects" is: $Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \epsilon_{u,i}$
3. To fit the model, we could use cross validation as: train(rating ~ as.factor(movieId) + as.factor(userId) + as.factor(date), method='lm', data=edx) Because it is very slow to train the model, I will compute an approximation by computing μ, b_i, b_μ, and estimating $f(d_{u,i})$, as the average of $y_{u,i} - \mu - b_i - b_u$

Reference

HarvardX PH125.8x Data Science: Machine Learning 6.2: Recommendation Systems

```r
#2. Define the function RMSE

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Define a range of lambdas

lambdas <- seq(0, 10, 0.25)

# Construct a function and calculate residual mean squared error

rmses <- sapply(lambdas, function(l){

  # Average rating on the training data set

  mu <- mean(edx$rating)

  # calculate b_i by applying penalized lambda l in the equation
  # -- b_i is regularized

  b_i <- edx  %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  # calculate b_u for a user of a movie by applying penalized lambda l in the equation
  # -- b_u is regularized

  b_u <- edx  %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  # calculate b_date for a user of a movie by applying penalized lambda l in the equation
  # -- b_date is regularized

  b_date_m <- edx  %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId')   %>%
    group_by(date) %>%
    summarize( b_date = sum(rating - b_i - b_u - mu)/(n() + l))

  # replace those NA values of b_date with median value of b_date_m$b_date

  b_date_m$b_date[is.na(b_date_m$b_date)] <- median(b_date_m$b_date, na.rm=T)


  # Calculate the predicted ratings

  predicted_ratings <-
    validation %>%
    #left join dataset b_i in order to retrieve regularized b_i
    left_join(b_i, by = "movieId") %>%
    #left join dataset b_u in order to retrieve regularized b_u
    left_join(b_u, by = "userId") %>%
    #left join b_date_m in order to retrieve regularized value b_date
    left_join(b_date_m, by=c('date' )) %>%
    # the predicted rating
```

```r
    mutate(pred = mu + b_i + b_u + b_date) %>%
    pull(pred)

  # Replace those NA values of predicted_ratings with median value of predicted_ratings

  predicted_ratings[is.na(predicted_ratings)] <- median(predicted_ratings, na.rm=T)


  # Calculate RMSEs

  return(RMSE(predicted_ratings, validation$rating))

})

qplot(lambdas, rmses, main="RMSEs vs Lamdas")
```




```r
rmse_results <- tibble(Name="min rmse", Value=min(rmses) )


rmse_results <- bind_rows(rmse_results,
                              tibble(Name="Optimal Lambda",
                                         Value=lambdas[which.min(rmses)]    ))
rmse_results %>% knitr::kable(align='c')
```

Name	Value
min rmse	0.8646938
Optimal Lambda	5.5000000

Observation

The RMSE of the model of movie + user + **date** effects improves to 0.8646938.

Reference

Introduction to Data Science – 34.7.5 Modeling movie effects