Harvard-PH125.9x - MovieLens Report

Camilo Lillo

29/5/2021

1. Introduction

The aims of this study is to create a movie recommendation system using the MovieLens dataset. The develop of the movieLens recommendation system consists of exploratory data analysis, formulation and, develop and machine learning models comparison on the rating of a user who watches a movie.

This document is organized as follows: First, repeating codes on the training and test set available in the course. Second, exploratory data analysis. Third, formulation and evaluationof predictive models and, finally, conclusion.

2. Playback of edx codes

Below are the exd codes:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(stringi)) install.packages("stringi", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("scales", repos = "http://cran.us.r-project.org")
if(!require(dismo)) install.packages("dismo", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(stringi)
library(scales)
library(dismo)
library(dplyr)
library(recosystem)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://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"))

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

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

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

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

edx0 = edx
```

Note that we add some packages in this step. In addition, we copy de edx data frame.

3. Exploratory data analysis

A continuación se presenta la variable de respuesta, denotada por Y, la cual corresponde al rating de la película proporcionada por el usuario. El soporte de la variable corresponde a los valores $i/2$, con $i = 1, 2, \ldots, 10$. La distribución es presentada a continuación:

```
fig1 <- edx %>% group_by(rating) %>% summarize(total = length(rating)) %>%
  ggplot(aes(x = rating, y = 100*total/(sum(total)))) +
  geom_bar(stat="identity",
           position=position_dodge(), width = 0.4,
           color = "black", fill = "orange") +
  geom_text(aes(label=percent(total/(sum(total)) %>% round(1))), vjust=1.6, color="black",
            position = position_dodge(0.9), size = 3) +

  labs(title="Rating Distribution",
       x="Rating", y = "Relative Frequency (%)")
fig1
```

Note que la distribución es una mezcla de dos distirbuciones; ntergers and non-integers. La distirbución de ratings es asimétrica a la izquierda. El número total de datos es de 9,000,055. Note que es más probable pertenecer a la distribución de enteros respecto a los no-enteros, es probable que sea difícil de separar estas dos poblaciones al momento de modelar. Entonces, por ahora sólo se utilizará un moelo global. Es decir, no se intentará realizar algún tipo de algoritmo supervisado para separar estas dos distribuciones.

Un análisis interesante corresponde a observar la distribución del rating average by user.

Si asumimos que la distribución del rating de cada usuario, es posible considerar intervalos de confianza para el rating de cada usuario (denotados por b_u), dados por:

$$CI(b_u)_{95\%} = \tilde{b}_u \pm 1.96\frac{\tilde{\sigma}_u}{\sqrt{n_u}},$$

donde $\tilde{b}_u$ corresponde al rating promedio del usuario u, mientras que $\tilde{sigma}$ y n_u corresponden a la desviación estándar del rating y número de películas que observó el usuario u.

El siguiente plot corresponde a la distribución del rating promedio respecto al usuario y su respectivo intervalo de confianza del 95% (userId variable). Para robustecer el análisis se utilizaron los usuarios que hayan visto al menos 100 películas.

```
tab_userId = edx %>% group_by(userId) %>%
  summarize(avg = mean(rating),
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

tab_userId = tab_userId[order(-tab_userId$avg),]
tab_userId = tab_userId %>% mutate(sds = ifelse(is.na(sds), 0, sds),
                                   cv  = ifelse(is.na(cv), 0, cv))
tab_userId$rank = c(1:NROW(tab_userId))
tab_userId = tab_userId %>% mutate(icsup = avg + 1.96*(sds/sqrt(len)))
tab_userId = tab_userId %>% mutate(icinf = avg - 1.96*(sds/sqrt(len)))

fig2 = tab_userId %>% filter(len > 100) %>% ggplot(aes(rank)) +
  geom_line(aes(y = avg, colour = "rating average")) +
  geom_line(aes(y = icsup, colour = "upper IC(95%)")) +
  geom_line(aes(y = icinf, colour = "lower IC(95%)")) +
  labs(title="", y = "ratings",
       x = "users ranking (users who have watched more than 100 movies)")
fig2
```

Note que existen usuarios que aparentemente son diferentes que otros a la hora de evaluar. ¿Ocurrirá lo mismo cuando agrupados por película A continuación se presenta un análisis similar al anterior, pero agrupando por película.

```
tab_movieId = edx %>% group_by(title) %>%
  summarize(avg = mean(rating),
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

tab_movieId = tab_movieId[order(-tab_movieId$avg),]
tab_movieId = tab_movieId %>% mutate(sds = ifelse(is.na(sds), 0, sds),
                                     cv  = ifelse(is.na(cv), 0, cv))
```

```
tab_movieId$rank = c(1:NROW(tab_movieId))
tab_movieId = tab_movieId %>% mutate(icsup = avg + 1.96*(sds/sqrt(len)))
tab_movieId = tab_movieId %>% mutate(icinf = avg - 1.96*(sds/sqrt(len)))

fig3 = tab_movieId %>% filter(len > 100) %>% ggplot(aes(rank)) +
  geom_line(aes(y = avg, colour = "rating average")) +
  geom_line(aes(y = icsup, colour = "upper IC(95%)")) +
  geom_line(aes(y = icinf, colour = "lower IC(95%)")) +
  labs(title="", x = "movie ranking (viewed more than 100 times)", y = "ratings")
fig3
```

Note que los resultados son bastante similares. Hay películas que poseen altos valores y son significativamente diferentes de otras películas con ratings más bajos.

A continuación se comparará la distribución del rating promedio del usuario respecto al rating promedio de la película, un plot de empirical cumulative distirbution function.

```
tab_userId  =  tab_userId[order(tab_userId$avg),]
tab_movieId = tab_movieId[order(tab_movieId$avg),]

tab_userId$k  = (c(1:NROW(tab_userId) ) - 0.5)/NROW(tab_userId )
tab_movieId$k = (c(1:NROW(tab_movieId)) - 0.5)/NROW(tab_movieId)

fig4 =
  ggplot() +
  geom_line(data=tab_userId,  aes(x = avg, y = k, colour = "rating (users) dist."))  +
  geom_line(data=tab_movieId, aes(x = avg, y = k, colour = "rating (movies) dist.")) +
  labs(title="", x = "ratings", y = "ECDF")
fig4
```

Note que las distribuciones son diferentes. Primero, la distribución del rating promedio por usuario posee mayores estadísticos de tendencia central. Este análisis podría ser evidencia de interacción entre película y usuario, ya que al aislar el usuario, o la película, se obtienen distribuciones diferentes en tendencia y forma (por ejemplo, la asimetría de la distribución). Este tema lo abordaremos más adelante.

Por otro lado, nosotros analizaremos dos aspectos asociados a la película. El año de estreno y el género de la película.

```
edx$year = as.numeric(stri_reverse(substr(stri_reverse(edx$title), 2, 5)))
edx = edx %>% mutate(year = ifelse(year < 1980, "< 1980", year))
tab_years2 = edx %>%
  group_by(year) %>%
  summarize(avg = mean(rating),
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

fig5.2 = tab_years2 %>% ggplot(aes(x=year, y=avg)) +
  geom_line(aes(y=avg, group = 1), col="black") +
  geom_point(aes(y=avg, group = 1), col="black") +
  geom_errorbar(aes(ymin=avg-(1.96*sds/sqrt(len)),
                    ymax=avg+(1.96*sds/sqrt(len))), width=.2,
                position=position_dodge(0.05)) +
  labs(title="", x = "year", y = "rating average") +
```

```
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
fig5.2
```

```
tab_genres = edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(avg = mean(rating),
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

b <- max(tab_genres$len)/(max(tab_genres$avg) - 3)
a <- b*(0 - 3)

fig6 = tab_genres %>% ggplot(aes(x=reorder(genres, -avg), y=len)) +
  geom_bar(col="black", stat="identity", fill = "orange") +
  geom_line(aes(y=avg * b + a, group = 1), col = "darkgreen", size = 1.2) +
  scale_y_continuous(name="n", sec.axis=sec_axis(~(. - a)/b, name = "Rating average")) +
  labs(title="", x = "") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
fig6
```

Se aprecia que tanto el año como el genero cambian el promedio de la distribución. Esto podría ser un indicio de que estas covariables podrían ser inputs de modelos predictivos para el rating.

No obsante de lo anterior, primero se provarán modelos numéricos clásicos relacionados a los sistemas de recomendaciones que sólo incluyen los efectos del usuario/película dentro de la formulación. A continuación se presentan estos modelos.

4. Modeling

En este trabajo nosotros abordamos los modelos mas usuales para sistemas de recomendación. Los cuales son presentados a continuación:

$$\mathrm{m}_1 : r_{iu} = \mu + e_{iu},$$

$$\mathrm{m}_2 : r_{iu} = \mu + b_i + e_{iu},$$

$$\mathrm{m}_3 : r_{iu} = \mu + b_i + b_u + e_{iu},$$

$$\mathrm{m}_4 : r_{iu} = \mu + b_i + b_u + q_i^\top p_u + e_{iu},$$

donde r_{iu} corresponde al rating de la i-th movie en el u-th user, μ es la media general, b_i is the movie bias, b_u is the user bias, el producto $q_i^\top p_u$ son vectores de parámetros que corresponden a la interacción usuarui/movie y e_{iu} es el error aleatorio.

El modelo m_1 corresponde a un modelo de media global, m_2 corresponde a un modelo con movie effect, m_3 corresponds to a model with movie and user effects. Finally, m_4 adds the movie/user interaction to the above model.

Para comparar la performance de estos modelos, nosotros utilizamos una validacion cruzada K-folds, con $K = 10$, utilizando el package dismo. Esto nos permite comparar exactamente los mismos subsets (or folds) para cada molesto, permitiendo una comparación más justa entre los 4 modelos propuestos.

Acontinuación se presenta el código para el modelo m_1, m_2 y m_3:

```
KF = 10
set.seed(2020-12-31, sample.kind = "Rounding")
edx$id_cv = kfold(1:NROW(edx), k = KF)
edx_val   = numeric()
for(u in 1:KF){
  m1_folds = edx %>% filter(id_cv != u) %>%
    summarise(avg = mean(rating))

  edx_test = edx %>% filter(id_cv == u)
  edx_test = edx_test[,c("userId", "movieId", "rating", "id_cv")]
  edx_test$pred_m1 = m1_folds

  edx_val = rbind(edx_val, edx_test, fill = TRUE)
}

RMSE_m1 = RMSE(pred = edx_val$pred_m1, obs = edx_val$rating)
```

```
KF = 10
set.seed(2020-12-31, sample.kind = "Rounding")

# ratings model for movie effects

edx$movieId = as.factor(edx$movieId)
edx_val_m2  = numeric()
for(u in 1:KF){
  xb <- mean(edx$rating)
  m2_folds = edx %>% filter(id_cv != u) %>%
    group_by(movieId) %>%
    summarise(b_i = mean(rating - xb))
  edx_test = edx %>% filter(id_cv == u)
  edx_test$pred_m2 <- xb + edx_test %>%
    left_join(m2_folds, by = "movieId") %>%
    pull(b_i)
  edx_val_m2 = rbind(edx_val_m2, edx_test, fill = TRUE)
}

edx_val_m2 = edx_val_m2[,c("userId", "movieId", "pred_m2")]
edx_val_m2 = na.omit(edx_val_m2)
edx_val_m2$movieId = as.numeric(as.character(edx_val_m2$movieId))

edx_val  = edx_val[,-1]
edx_val  = left_join(edx_val, edx_val_m2, by = c("userId", "movieId"))

RMSE_m2 = RMSE(pred = edx_val$pred_m2, obs = edx_val$rating, na.rm = TRUE)
```

```
KF = 10
set.seed(2020-12-31, sample.kind = "Rounding")

# add user effect

edx$movieId = as.factor(edx$movieId)
edx_val_m3  = numeric()
for(u in 1:KF){
```

```
  xb <- mean(edx$rating)
  m3_folds = edx %>% filter(id_cv != u) %>%
    group_by(movieId) %>%
    summarise(b_i = mean(rating - xb))

  m3_add_user_in_folds = edx %>% filter(id_cv != u) %>% left_join(m3_folds, "movieId") %>%
    group_by(userId) %>% summarise(b_u = mean(rating - xb - b_i))

  edx_test = edx %>% filter(id_cv == u)

  edx_test$pred_m3 <- edx_test %>%
    left_join(m3_folds, by = "movieId") %>%
    left_join(m3_add_user_in_folds, by = "userId") %>%
    mutate(rating_pred = xb + b_i + b_u) %>%
    pull(rating_pred)

  edx_val_m3 = rbind(edx_val_m3, edx_test, fill = TRUE)
}

edx_val_m3 = edx_val_m3[,c("userId", "movieId", "pred_m3")]
edx_val_m3 = na.omit(edx_val_m3)

edx_val$userId    = as.character(edx_val$userId)
edx_val_m3$userId = as.character(edx_val_m3$userId)

edx_val$movieId    = as.character(edx_val$movieId)
edx_val_m3$movieId = as.character(edx_val_m3$movieId)

edx_val   = left_join(edx_val, edx_val_m3, by = c("userId", "movieId"))
RMSE_m3 = RMSE(pred = edx_val$pred_m3, obs = edx_val$rating, na.rm = TRUE)
```

Para el modelo m_4, nosotros utilizamos el package recosystem, el cual realiza una validación cruzada para determinar ciertos inputs que utiliza el modelo. Entonces realizaremos nuestra validación cruzada de la siguiente manera: (1) determinaremos los parámetros del m_4 para todo el conjunto edx. (2) utilizaremos los 10-folds determinados anteriormente para evaluar el setting del step (1) y comparar el RMSE con el resto de los modelos. Entonces, el siguiente Rcode presenta el tuning de parámetros (please not run, this takes too long).

```
## not run!
set.seed(2021-05-28, sample.kind = "Rounding")

train_edx <-  with(edx0, data_memory(user_index = userId,
                                     item_index = movieId,
                                     rating = rating))
test_edx  <-  with(validation, data_memory(user_index = userId,
                                           item_index = movieId,
                                           rating = rating))
# r    =  recosystem::Reco()
# opts =  r$tune(train_edx, opts = list(dim = c(20, 30, 40),
#                                       lrate = seq(0.025, 0.1, 0.025),
#                                       costp_l2 = c(0.05, 0.075, 0.1),
#                                       costq_l2 = c(0.001, 0.005, 0.01, 0.015),
#                                       nthread  = 8, niter = 10))
```

El siguiente código proporciona la validación cruzada 10-folds.

```
optim_par = list()
optim_par[["dim"]]      = 30
optim_par[["costp_l1"]] = 0
optim_par[["costp_l2"]] = 0.075
optim_par[["costq_l1"]] = 0
optim_par[["costq_l2"]] = 0.015
optim_par[["lrate"]]    = 0.05
optim_par[["loss_fun"]] = 0.7987279

edx$movieId = as.factor(edx$movieId)
edx_val_m4  = numeric()
for(u in 1:KF){

  m4_folds = edx %>% filter(id_cv != u)
  edx_test = edx %>% filter(id_cv == u)

  mfold <-  with(m4_folds, data_memory(user_index = userId,
                                       item_index = movieId,
                                       rating = rating))
  tfold  <-  with(edx_test, data_memory(user_index = userId,
                                        item_index = movieId,
                                        rating = rating))
  r    =  recosystem::Reco()
  r$train(mfold, opts = c(optim_par, nthread = 8, niter = 20))

  pred_rating_folds <-  r$predict(tfold, out_memory())

  edx_test$pred_m4 = pred_rating_folds

  edx_val_m4 = rbind(edx_val_m4,
                     edx_test[,c("userId", "movieId", "pred_m4")],
                     fill = TRUE)

}

edx_val_m4 = edx_val_m4[,-1]

edx_val$userId    = as.character(edx_val$userId)
edx_val_m4$userId = as.character(edx_val_m4$userId)

edx_val$movieId    = as.character(edx_val$movieId)
edx_val_m4$movieId = as.character(edx_val_m4$movieId)

edx_val   = left_join(edx_val, edx_val_m4, by = c("userId", "movieId"))
RMSE_m4 = RMSE(pred = edx_val$pred_m4, obs = edx_val$rating, na.rm = TRUE)
```

Finalmente, tenemos la siguiente comparación de RMSE para seleccionar el mejor modelo

Table 1: Cross validation 10-folds - performance comparison

	Global average	Movie effect	Movie and user effects	Movie and user effects and the interaction movie/user
RMSE	1.060332	0.9436378	0.8656021	0.7882029

El modelo con el RMSE más bajo consiste en el movie, user effects and interaction movie/user model. Entonces, utilizando este último modelo se realizará la predicción en el set de validación del curso. Tenemos el siguiente código: