MovieLens Recommendation System

Kim Wager

2024-01-10

Create the datasets as instructed

```
##########################################################
# Create edx and final_holdout_test sets
##########################################################

# Note: this process could take a couple of minutes

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

```
Loading required package: tidyverse
```

```
Warning: package 'lubridate' was built under R version 4.3.3
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0
──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.2
```

```
── Conflicts ────────────────────────────────────────── tidyverse_conflicts()
──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

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

```
Loading required package: caret
```

```
Warning: package 'caret' was built under R version 4.3.3
```

```
Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift
```

```
library(tidyverse)
library(caret)

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

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip",
dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings  <-  as.data.frame(str_split(read_lines(ratings_file),  fixed("::"),
simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies   <-   as.data.frame(str_split(read_lines(movies_file),   fixed("::"),
```

```
simplify = TRUE),
                          stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
```

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

```
# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

```
Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

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

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

First, we implement the simple baseline model and calculate the RMSE. This baseline model makes the same prediction (the overall average rating) for every movie, regardless of the specific movie or user. While this is obviously not very sophisticated, it gives us:

- A starting point for measuring improvement
- A simple implementation to verify the RMSE calculation function
- A benchmark that any more complex model should beat

Step 1: Calculate the mean rating (μ)

This calculates μ (mu), which represents the average of all ratings in the training dataset (edx). We use this as our simplest possible prediction - assuming every movie would be rated at this average value.

```
# Calculate the overall mean rating from training data
mu <- mean(edx$rating)
print(paste("Overall mean rating:", round(mu, 4)))
```

```
[1] "Overall mean rating: 3.5125"
```

Step 2: Create predictions for validation set

This creates a vector of predictions by repeating the mean rating (mu) for each row in the test dataset. The rep() function repeats the value, and nrow(final_holdout_test) tells it how many times to repeat.

```
# Create predictions using just the mean (mu) from the previous step
predictions <- rep(mu, nrow(final_holdout_test))
```

Step 3: Define RMSE function and calculate error

The RMSE function calculates the Root Mean Square Error, which measures how far the predictions deviate from the actual ratings. It works by:

1. Taking the difference between true and predicted ratings
2. Squaring these differences
3. Calculating the mean of the squared differences
4. Taking the square root of that mean

```
# Define RMSE function
# Takes two parameters: true ratings (edx dataset) and predicted ratings
(final_holdout_test dataset)
# Returns a single number representing prediction accuracy (lower is better)
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Calculate RMSE
# Print the mean rating with 4 decimal places for clarity
# paste() combines text and numeric value into a single string
naive_rmse <- RMSE(final_holdout_test$rating, predictions)
print(paste("RMSE for baseline model:", round(naive_rmse, 4)))
```

```
[1] "RMSE for baseline model: 1.0612"
```

Step 4: Create results table

Here we create a data frame to store the results. This is used to help us compare different models as we develop more complex ones. The method column describes the approach used and the RMSE column stores the error value for that method.

```
# Create results table
rmse_results <- tibble(
  Method = "Baseline model",
  RMSE = naive_rmse
)

knitr::kable(rmse_results, digits = 4,
             caption = "Baseline Model Performance")
```

Method	RMSE
Baseline model	1.0612

Table 1: Baseline Model Performance