# Harvardx Capstone MovieLense Project

Lindsay Snyman

2024-10-09

`{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE)` ##Overview This report summarises the development and evaluation of a recommendation system using matrix factorization on the MovieLens dataset. Matrix factorization is very often referred to in the literature as one of the most effective collaborative filtering methods, which performs well on a large-scale recommendation problem.The original dataset, which is very large, includes user ratings of movies. Therefore, for the purpose of this exercise, it has used just a 30% sample of the data that was provided. The model was developed using the recosystem library that provides a very efficient implementation of different matrix factorization methods.

##Introduction Recommender systems became necessary for any platform that gives users a wide field of choice: from movies to music. These systems allow users to find an item relevant to their preferences and behavioural patterns. In this regard, collaborative filtering approaches became popular; among them, one of the recognized ones was matrix factorization since it was able to model complex interactions between users and items.(Koren et al., 2009) In this project, a MovieLens dataset provided by EdX was used for creating a recommendation system. It consists of millions of user ratings for thousands of movies (Harper & Konstan, 2016). The objective in this was to be able to train a model of matrix factorization on a subset of the data, tune the model's hyperparameters, and then assess its performance by the use of the Root Mean Squared Error (RMSE) metric, which is used to evaluate the performance of predictive models(Steck, 2013) "'{r}
############################################################
# Create edx set, validation set, and submission file ##########################################

## Note: this process could take a couple of minutes for loading required package: tidyverse and package caret

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

ratings <- read.table(text = gsub("::", "", 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")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId], title = as.character(title), genres = as.character(genres)) movielens <- left_join(ratings, movies, by = "movieId")

## The Validation subset will be 10% of the MovieLens data.

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 subset: 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, temp, movielens, removed)

```
##Method/Analysis
The MovieLens 10M dataset comprises ratings of movies rated by approximately 10 million users. From this
```{r}
sample_size <- 0.30
edx_sample <- edx %>% sample_frac(sample_size)
```

The data was the looked at to see what is included in the data. This was inspected. `{r} summary(edx_sample) glimpse(edx_sample)` A basic EDA was performed to see the distribution of the ratings. `{r , echo=FALSE} ggplot(edx_sample, aes(x = rating)) + geom_histogram(binwidth = 0.5, fill = "blue", color = "black", alpha = 0.7) + labs(title = "Distribution of Ratings in edx Sample Dataset (30%)", x = "Rating", y = "Count")` The unique users and movies were observed in the smaple. `{r} num_users <- edx_sample %>% distinct(userId) %>% nrow() num_movies <- edx_sample %>% distinct(movieId) %>% nrow() cat("Number of unique users in the sample:", num_users, "\n") cat("Number of unique movies in the sample:", num_movies, "\n")`

First of all, the data needed preprocessing into a form suitable for matrix factorization. A random sample for training and evaluation of 30% was used, out of which the training set was used to build a recommendation model. Using the recosystem library, the ratings were converted into a sparse matrix format. "`{r} # Convert data into the format required for`recosystem' train_data <- data_memory(user_index = train_set$userId, item_index = train_set$movieId, rating = train_set$rating, index1 = TRUE)

test_data <- data_memory(user_index = test_set$userId, item_index = test_set$movieId, rating = test_set$rating, index1 = TRUE)

# Initialize the recommender model from `recosystem`

recommender <- Reco()

```
The recosystem library has a very efficient implementation for matrix factorization by using Stochastic
The model was tuned for the following hyperparameters:
dim: The number of latent factors.
costp_l2: Regularization parameter for user-specific latent factors.
costq_l2: Regularization parameter for item-specific latent factors.
lrate: Learning rate for the gradient descent optimization.
These parameters take ranges of values, and we have done a grid search over these to find the optimal se
```{r}
# Tuning over a different range of parameters for more detailed exploration
tune_result <- recommender$tune(train_data, opts = list(dim = c(10, 20, 40),
                                                        costp_l2 = c(0.01, 0.1, 0.2),
```

```
costq_l2 = c(0.01, 0.1, 0.2),
lrate = c(0.05, 0.1, 0.2),
niter = 15,
nthread = 4))
```

Having identified the optimal hyperparameters, we went ahead and trained our model using the matrix factorization algorithm in the recosystem. It ran 30 iterations to make sure that it had converged. `{r} recommender$train(train_data, opts = c(tune_result$min, niter = 30))`

The model was finally tested on the test set via RMSE. The RMSE is one of the standard metrics used for evaluating recommender systems, and it is the magnitude of the prediction error, averaged (Aggarwal, 2016.) `{r} calculate_rmse <- function(actual, predicted) {   sqrt(mean((actual - predicted) ^ 2)) }`

##Results On running the model on test set and validation set respectively, following were results observed: Test set - RMSE 0.8457103 Validation Set - RMSE 0.8465137 The small value of the RMSE of the matrix factorization model shows that it is able to predict user preferences quite accurately. The scatter plot of actual vs. predicted ratings showed that for most user-movie pairs, its predictions were close to the actual ratings.

##Discussion The results show that matrix factorization is quite effective in predicting MovieLens data. Reduction of the RMSE value was important, and hyperparameter tuning helped in its reduction; the grid search approach was used for the selection of the best model. Regularization helps to avoid overfitting not to give too much emphasis on any particular user or item when the number of ratings is few. By decomposing the user-item interaction matrix into latent factors, the model is able to capture hidden patterns of user preferences and movie characteristics. Using the implementation of matrix factorization provided by the recosystem library allowed for efficient training and prediction on a large dataset.

##Conclusion The solution described in this work developed a recommendation system by carrying out matrix factorization on a 30% sample of the MovieLens dataset. The model obtained had a low RMSE on both the test(0.8465137) and validation sets (0.8457103), showing the capabilities of matrix factorization with respect to predicting user ratings. Further improvements could be done based on extra features such as movie genres or timestamps, which might enhance the model's accuracy even more.

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

`{r cars} summary(cars)`

## Including Plots

You can also embed plots, for example:

`{r pressure, echo=FALSE} plot(pressure)`

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.