Project 4 — Main

Ziqin Zhao, Xinlin Zhang, Jiadong Wu, Kaiqi Wang, Marko Konte

4/18/2020

Project 4 — Group 5

In our project, we explored matrix factorization methods for recommender system. The goal is to match consumers with most appropriate products. Matrix factorization generally has 3 parts:

Algorithm : Stochastic Gradient Descent (A1) and Gradient Descent With Probabilistic Assumption (A2)

Regularization : Penalty of Magnitudes (R1) Bias and Intercepts (R2)

Postprocessing : SVD with KNN (P2)

Our pairings:

1. A1 + R1 + R2 + P2
2. A2 + P2

Step 1 Load Data and Train-test Split

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(ggplot2)
library(coop)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  2.1.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.3

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
data <- read.csv("../data/ml-latest-small/ratings.csv")


set.seed(0)
test_idx <- sample(1:nrow(data), round(nrow(data)/5, 0))
train_idx <- setdiff(1:nrow(data), test_idx)
data_train <- data[train_idx,]
data_test <- data[test_idx,]
```

###Step 2 Matrix Factorization #### Step 2.1 Algorithm and Regularization

We only choose the first 5000 rows from the original dataset to train and test our model.

```r
U <- length(unique(data[1:5000,]$userId))
I <- length(unique(data[1:5000,]$movieId))
```

Step 2.2 Parameter Tuning

we tune parameters by cross validation (K=5) All of functions for cross validation is in the lib ()

Step 2.2.1 Tunning parameter for A1+R1+R2 (only 5000 rows)

Functions for cross validation of A1+R1+R2 is in the lib (‘cross_validation_A1+R1+R2.R’). Funcrions for Matrix Factorization of A1+R1+R2 is in the lib (‘Matrix_Factorization_A1+R1+R2.R’)

```r
source("../lib/cross_validation_A1+R1+R2.R")
source("../lib/Matrix_Factorization_A1+R1+R2.R")

f_list <- seq(10, 20, 10)
l_list <- seq(-2, -1, 1)
f_l <- expand.grid(f_list, l_list)
```

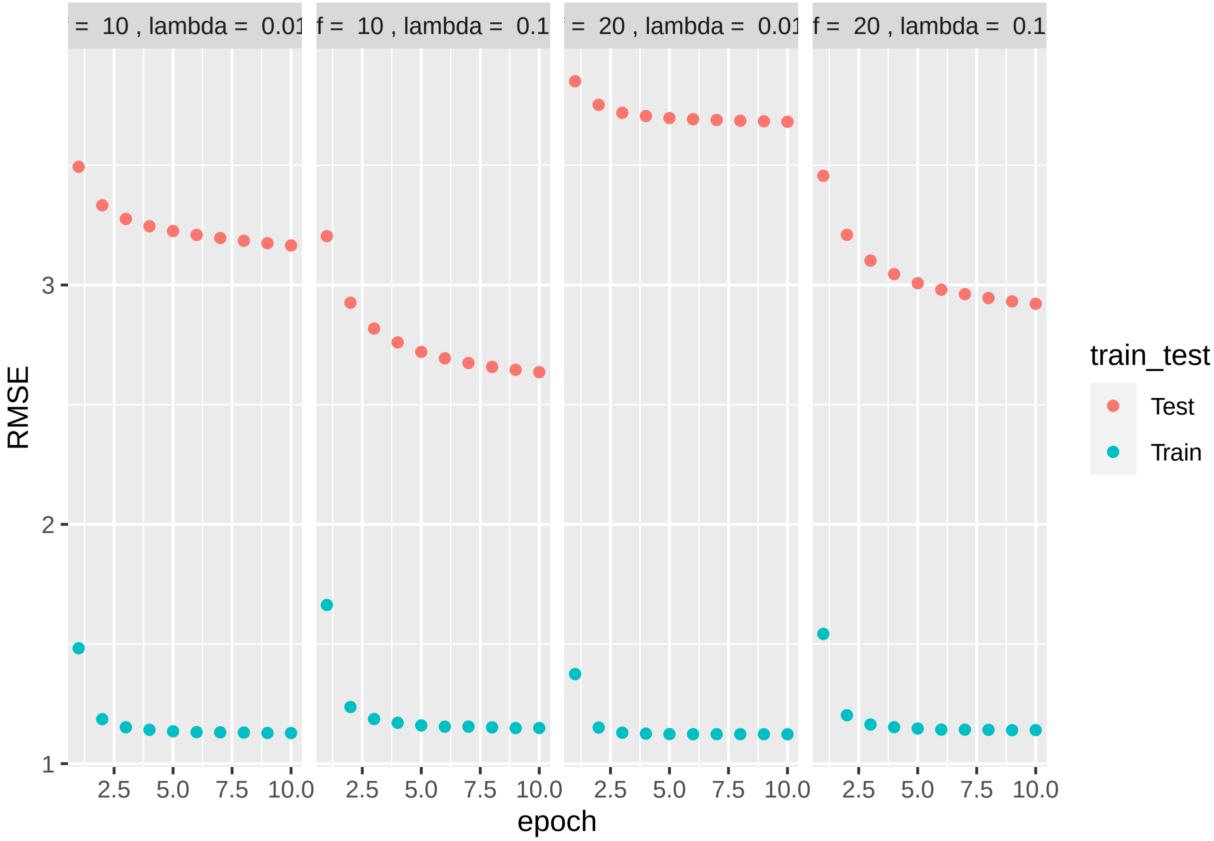
Note: you can just use the result of this part (run for too much time).

```r
# use cross validation = 5 to get the rmse
result_summary_r12 <- array(NA, dim = c(nrow(f_l), 10, 4))
run_time <- system.time(for(i in 1:nrow(f_l)){
    par <- paste("f = ", f_l[i,1], ", lambda = ", 10^f_l[i,2])
    cat(par, "\n")
    current_result <- cv.function.r12(data[1:5000,], K = 5, f = f_l[i,1], lambda = 10^f_l[i,2])
    result_summary_r12[,,i] <- matrix(unlist(current_result), ncol = 10, byrow = T)
    print(result_summary_r12)
})

save(result_summary_r12, file = "../output/rmseR12.Rdata")
```

Plot tuning parameters of A1+R1+R2

```r
load(file = "../output/rmseR12.Rdata")

rmse <- data.frame(rbind(t(result_summary_r12[1,,]), t(result_summary_r12[2,,])), train_test = rep(c("T
```

```r
rmse$epoch <- as.numeric(gsub("X", "", rmse$epoch))
rmse %>% ggplot(aes(x = epoch, y = RMSE, col = train_test)) + geom_point() + facet_grid(~par)
```



From the plot, we use parameters f = 10 and lambda=0.1 in this pair (A1+R1+R2)

Step 2.2.2 Tunning parameter for A2 (only 5000 rows)

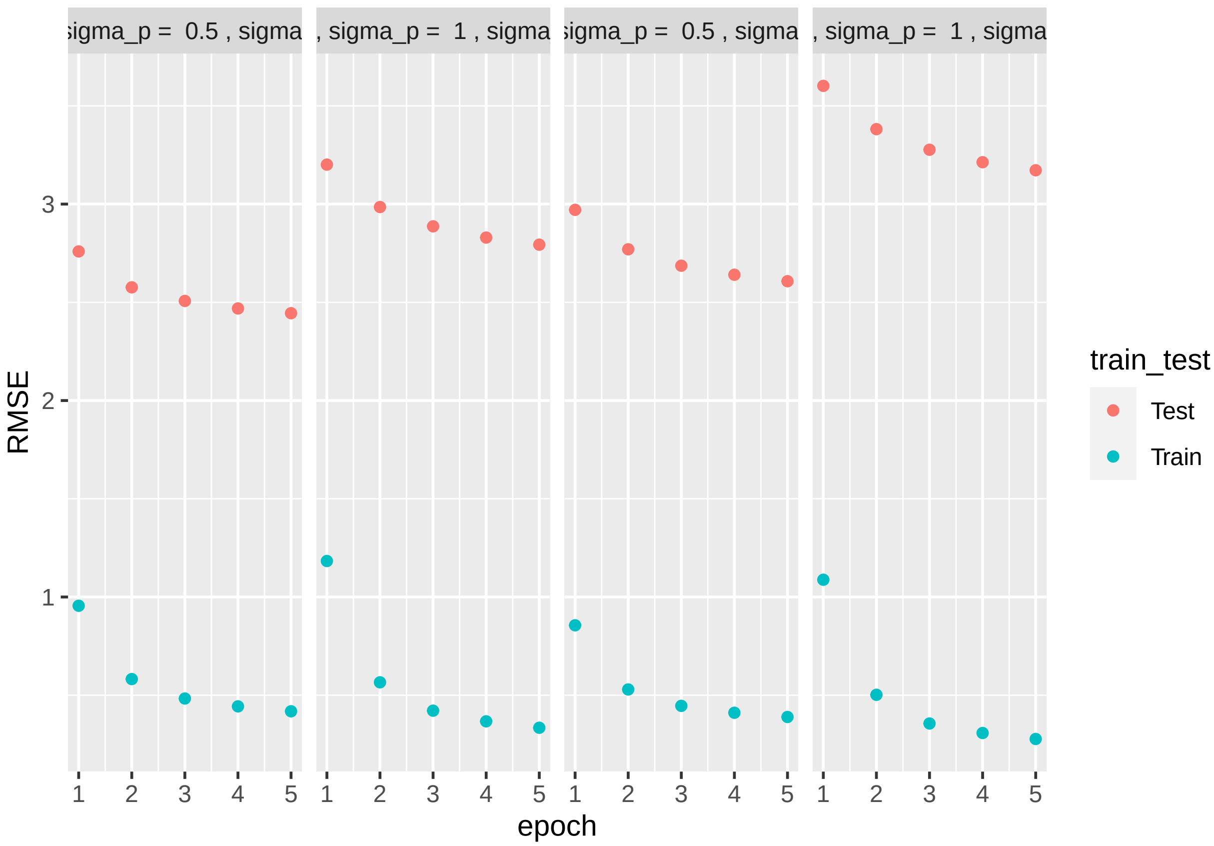
```r
source("../lib/cross_validation_pmf.R")
source("../lib/Matrix_Factorization_pmf.R")

f_list <- c(10,20,10,20)
lp_list <- c(1,1,0.5,0.5)
lq_list <- c(1,1,0.5,0.5)
f_l = cbind(f_list,lp_list,lq_list)
```

Note: you can just use the result of this part (run for too much time).

```r
#cross-validation with K = 5
result_summary <- array(NA, dim = c(nrow(f_l), 5, 4))
run_time <- system.time(for(i in 1:nrow(f_l)){
    par <- paste("f = ", f_l[i,1], ", sigma_p = ", f_l[i,2],", sigma_q = ", f_l[i,3])
    cat(par, "\n")
    current_result <- cv.function_pmf(data[1:5000,], K = 5, f = f_l[i,1],sigma_p=f_l[i,2],sigma_q= f_l[
    result_summary[,,i] <- matrix(unlist(current_result), ncol = 5, byrow = T)
    print(result_summary)
})
save(result_summary, file = "../output/rmse_pmf.Rdata")
```

Plot tuning parameters of A2

```r
load(file = "../output/rmse_pmf.Rdata")

rmse_pmf <- data.frame(rbind(t(result_summary[1,,]), t(result_summary[2,,])), train_test = rep(c("Train
rmse_pmf$epoch <- as.numeric(gsub("X", "", rmse_pmf$epoch))
rmse_pmf %>% ggplot(aes(x = epoch, y = RMSE, col = train_test)) + geom_point() + facet_grid(~par)
```



From the plot, we use parameters f = 10 and sigma_p=0.5 sigma_q=0.5 in the model (A2)

Step 3 Postprocessing: SVD with KNN

```r
set.seed(0)

#get the subset of train set and test set from the first 5000 data
test_idx2 <- sample(1:5000, 5000/5, 0)
train_idx2 <- setdiff(1:5000, test_idx2)
data_train2 <- data[train_idx2,]
data_test2 <- data[test_idx2,]
```

3.1 Postprocessing for A1+R1+R2

Get the result from best parameters in model 1(A1+R1+R2)

Note: you can just use the result of this part (run for too much time).

```r
resultr12 <- gradesc.r12(f = 10, lambda = 0.1,lrate = 0.01, max.iter = 100,
                         stopping.deriv = 0.01,data = data[1:5000,],
                         train = data_train2, test =data_test2)
save(resultr12, file = "../output/mat_facR12.RData")
```

We have already wrote the function of SVD with KNN in the lib ('KNN.R'). Therefore, we just used it directly from lib.

```
load(file = "../output/mat_facR12.RData")
source("../lib/KNN.R")
KNN_Rating <- KNN.post(data_train2, resultr12) # Rating after applying KNN
Rating <- t(resultr12$p)%*%(resultr12$q)  # Rating without knn
#Knn_Rating: (32*2427: the number in the matrix is the score of ratings)
```

```
RMSE <- function(rating, est_rating){
  sqr_err <- function(obs){
    sqr_error <- (obs[3] - est_rating[as.character(obs[1]), as.character(obs[2])])^2
    return(sqr_error)
  }
  return(sqrt(mean(apply(rating, 1, sqr_err))))
}

A1_chart <- tibble(rating=c("Without KNN","With KNN"),
                   train=c(RMSE(data_train2,Rating),RMSE(data_train2,KNN_Rating)),
                   test=c(RMSE(data_test2,Rating),RMSE(data_test2,KNN_Rating)))
A1_chart
```

```
## # A tibble: 2 x 3
##   rating      train  test
##   <chr>       <dbl> <dbl>
## 1 Without KNN  1.52  1.64
## 2 With KNN     1.52  1.50
```

```
# train mse with knn is the same as mse without knn.
```

3.1 Postprocessing for A2

Get the result from best parameters for model 2(A2)

Note: you can just use the result of this part (run for too much time).

```
result <- gradesc_pmf(f = 10, sigma_p = 0.5,sigma_q = 0.5, lrate = 0.01, max.iter = 100, stopping.deriv
save(result, file = "../output/mat_fac_pmf.RData")
```

Using the function from lib ('KNN.R')

```
load(file = "../output/mat_fac_pmf.RData")
source("../lib/KNN.R")
KNN_Rating <- KNN.post(data_train2, result) # Rating after applying KNN
Rating <- t(result$p)%*%(result$q)  # Rating without KNN
```

```
RMSE <- function(rating, est_rating){
  sqr_err <- function(obs){
    sqr_error <- (obs[3] - est_rating[as.character(obs[1]), as.character(obs[2])])^2
    return(sqr_error)
  }
  return(sqrt(mean(apply(rating, 1, sqr_err))))
}
A2_chart<-tibble(rating=c("Without KNN","With KNN"),
       train=c(RMSE(data_train2,Rating),RMSE(data_train2,KNN_Rating)),
       test=c(RMSE(data_test2,Rating),RMSE(data_test2,KNN_Rating)))
A2_chart
```

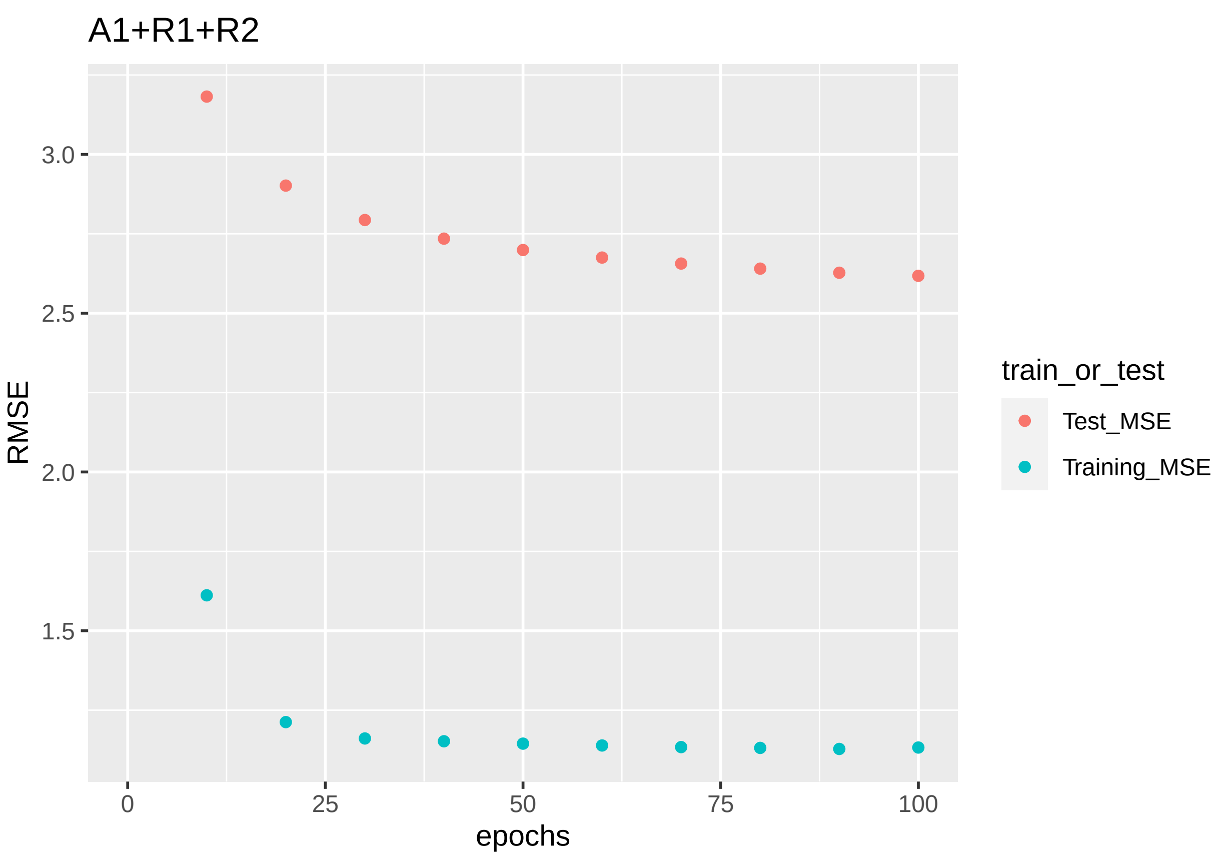
```
## # A tibble: 2 x 3
##   rating      train  test
##   <chr>       <dbl> <dbl>
## 1 Without KNN  1.11  1.18
## 2 With KNN     1.11  1.17
```

Step 4 Evaluation (Model Comparision)

4.1 A1+R1+R2

```
RMSE <- data.frame(epochs = seq(10, 100, 10),
                   Training_MSE = resultr12$train_RMSE,
                   Test_MSE = resultr12$test_RMSE) %>%
  gather(key = train_or_test, value = RMSE, -epochs)

RMSE %>% ggplot(aes(x = epochs, y = RMSE,col = train_or_test)) +
  geom_point() +
  scale_x_discrete(limits = seq(10, 100, 10)) +
  xlim(c(0, 100))+
  labs(x = "epochs",  title = "A1+R1+R2")
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

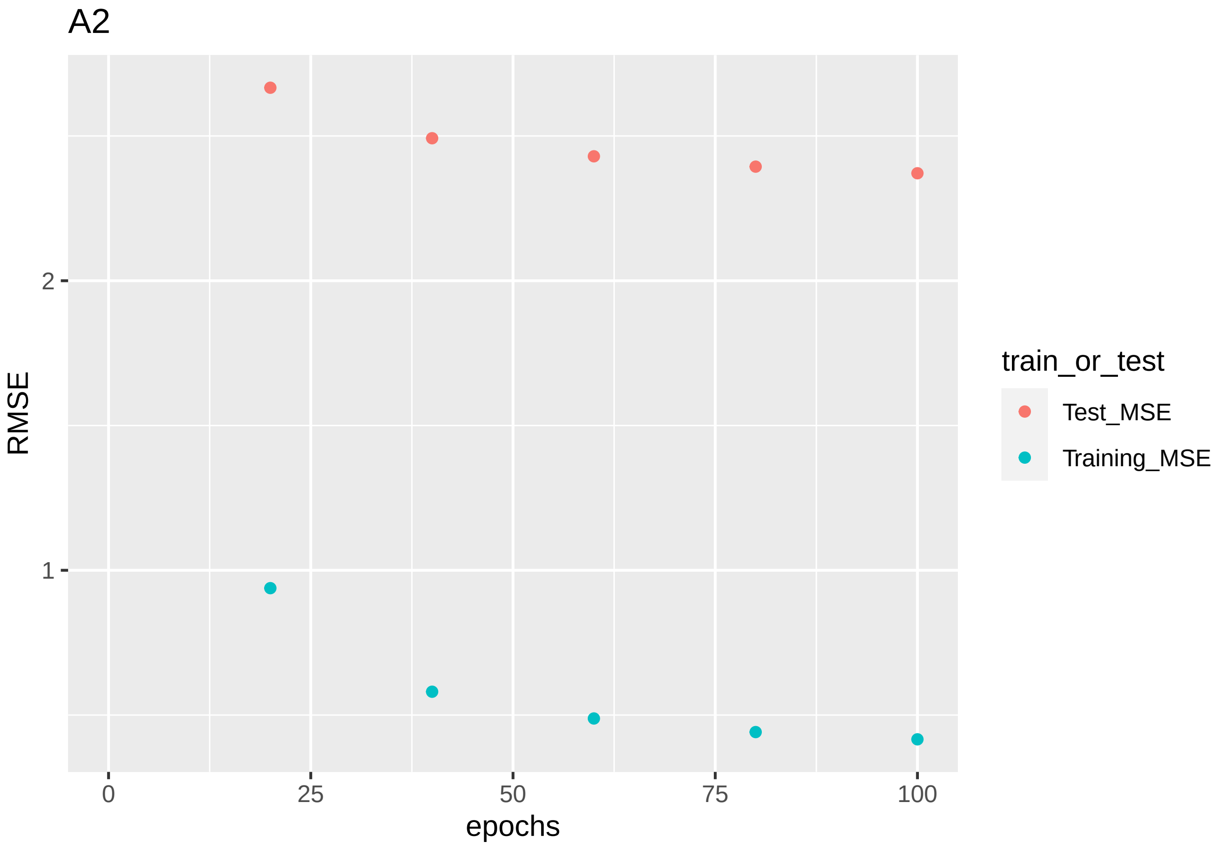


4.2 A2

```
RMSE <- data.frame(epochs = seq(20, 100, 20),
                   Training_MSE = result$train_RMSE,
```

```r
                    Test_MSE = result$test_RMSE) %>%
  gather(key = train_or_test, value = RMSE, -epochs)

RMSE %>% ggplot(aes(x = rep(seq(20, 100, 20),2), y = RMSE,col = train_or_test)) +
  geom_point() +
  scale_x_discrete(limits = seq(20, 100, 20)) +
  xlim(c(0, 100)) +
  labs(x = "epochs",  title = "A2")
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



#Step 5: Conclusions

Without postprocessing (SVD with KNN):

Test RMSE for A1 + R1 + R2 : 1.642475

Test RMSE for A2 : 1.178571

After postprocessing:

Test RMSE for A2 + R1 + R2 + P2 : 1.495440

Test RMSE for A2 + P2 : 1.174408

After comparing all the results, we find A2 + P2 has the best performance.