Project 4 ALS (A3)

HANG HU hh2718

11/20/2019

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)
data <- read.csv("C:/Users/huhan/OneDrive/Desktop/2019 Fall/GR 5243/fall2019-project4-sec2-grp2-master/fall2
019-project4-sec2-grp2-master/data/ml-latest-small/ratings.csv")
set.seed(0)
## shuffle the row of the entire dataset
data <- data[sample(nrow(data)),]
## get a small dataset that contains all users and all movies
unique.user<-duplicated(data[,1])
unique.movie<-duplicated(data[,2])
index<-unique.user & unique.movie
all.user.movie <- data[!index,]

## split training and test on the rest
rest <- data[index,]
test_idx <- sample(rownames(rest), round(nrow(data)/5, 0))
train_idx <- setdiff(rownames(rest), test_idx)

## combine the training with the previous dataset, which has all users and all movies
data_train <- rbind(all.user.movie, data[train_idx,])
data_test <- data[test_idx,]

## sort the training and testing data by userId then by movieId,
## so when we update p and q, it is less likely to make mistakes
data_train <- arrange(data_train, userId, movieId)
data_test <- arrange(data_test, userId, movieId)
```

###Step 2 Matrix Factorization #### Step 2.1 Algorithm and Regularization Here I perform stochastic gradien descent to do matrix factorization. Your algorithm should consider case that there are new users and movies adding to the dataset you used to train. In other words, the dimension your matrix R, q, p is dynamic.

- For algorithms, the referenced paper are:

A1. Stochastic Gradient Descent Section: Learning Algorithms-Stochastic Gradient Descent

A2. Gradient Descent with Probabilistic Assumptions Section 2

A3. Alternating Least Squares Section 3.1

- For regularizations, the referenced paper are:

R1. Penalty of Magnitudes Section: a Basic Matrix Factorization Model

R2. Bias and Intercepts Section: Adding Biases

R3. Temporal Dynamics Section 4

```r
U <- length(unique(data$userId))
I <- length(unique(data$movieId))
source("C:/Users/huhan/OneDrive/Desktop/2019 Fall/GR 5243/fall2019-project4-sec2-grp2-master/fall2019-projec
t4-sec2-grp2-master/lib/ALS1.R")
```

Step 2.2 Parameter Tuning

Here you should tune parameters, such as the dimension of factor and the penalty parameter \(\lambda\) by cross-validation.

```r
source("C:/Users/huhan/OneDrive/Desktop/2019 Fall/GR 5243/fall2019-project4-sec2-grp2-master/fall2019-projec
t4-sec2-grp2-master/lib/cross_validation_als.R")
f_list <- seq(10, 20, 10)
l_list <- seq(-1, 0, 1)
f_l <- expand.grid(f_list, l_list)
```

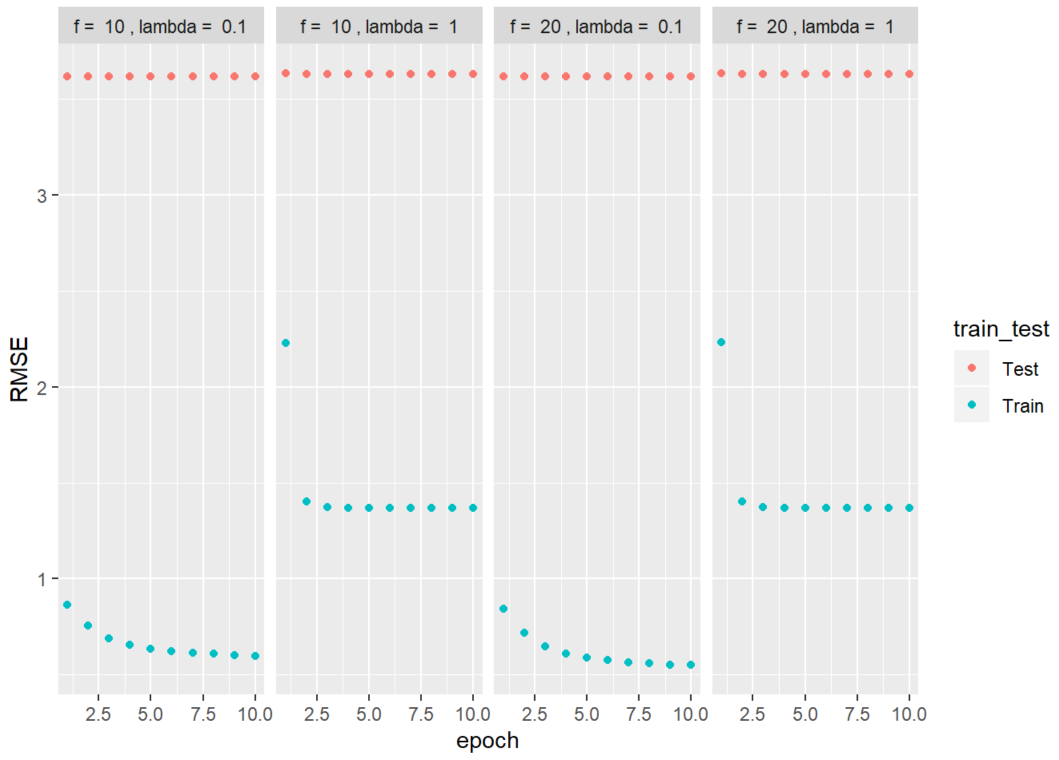
```r
result_summary <- array(NA, dim = c(nrow(f_l), 10, 4))
run_time <- system.time(for(i in 1:nrow(f_l)){
    par <- paste("f = ", f_l[i,1], ", lambda = ", 10^f_l[i,2])
    cat(par, "\n")
    current_result <- cv.function(data, K = 5, f = f_l[i,1], lambda = 10^f_l[i,2])
    result_summary[,,i] <- matrix(unlist(current_result), ncol = 10, byrow = T)
    print(result_summary)

})

save(result_summary, file = "C:/Users/huhan/OneDrive/Desktop/2019 Fall/GR 5243/fall2019-project4-sec2-grp2-m
aster/fall2019-project4-sec2-grp2-master/output/rmsea3new.Rdata")
```

It takes a while to run this chunck so we have saved the output and can be loaded directly.

```r
load("C:/Users/huhan/OneDrive/Desktop/2019 Fall/GR 5243/fall2019-project4-sec2-grp2-master/fall2019-project4
-sec2-grp2-master/output/rmsea3new.Rdata")
rmse <- data.frame(rbind(t(result_summary[1,,]), t(result_summary[2,,])), train_test = rep(c("Train", "Test"
), each = 4), par = rep(paste("f = ", f_l[,1], ", lambda = ", 10^f_l[,2]), times = 2)) %>% gather("epoch",
"RMSE", -train_test, -par)
rmse$epoch <- as.numeric(gsub("X", "", rmse$epoch))
rmse %>% ggplot(aes(x = epoch, y = RMSE, col = train_test)) + geom_point() + facet_grid(~par)
```



Step 3 Postprocessing

After matrix factorization, postporcessing will be performed to improve accuracy. The referenced papers are:

P1:Global bias correction Section 4.1

P2:Postprocessing SVD with KNN Section 3.5

P3:Postprocessing SVD with kernel ridge regression Section 3.6

P4:Linearly combination of predictors Section 4.1

```
result <- ALS1(f = 10, lambda =0.1,
                    max.iter = 5, data = data, train = data_train, test = data_test)

save(result, file = "C:/Users/huhan/OneDrive/Desktop/2019 Fall/GR 5243/fall2019-project4-sec2-grp2-master/fa
ll2019-project4-sec2-grp2-master/output/mat_fac_a3_new.RData")
```

Knn PostProcessing

```
load("C:/Users/huhan/OneDrive/Desktop/2019 Fall/GR 5243/fall2019-project4-sec2-grp2-master/fall2019-project4
-sec2-grp2-master/output/mat_fac_a3_new.RData")

library(lsa)
```

```
## Loading required package: SnowballC
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
ratingmean<-function(data){
  mean(data$rating)
}

rating<-ddply(data,.(movieId),ratingmean)
rating$index<-c(1:nrow(rating))


l_distance<-cosine(result$Movie)
diag(l_distance)<-0
cos_id<-rep(1:I,ncol=I)
knn_r<-rep(1:I,ncol=I)
for (i in 1:I){
    cos_id[i]<-which.max(rank(l_distance[,i]))
    knn_r[i]<-rating[rating$index==cos_id[i],]$V1
  }
```

Linear Regression for Training dataset

```
data_train$movieindex<-dense_rank(data_train$movieId)

r<-rep(0,nrow(data_train))
for (i in 1:nrow(data_train)){
  rowindex<-data_train$userId[i]
  columindex<-data_train$movieindex[i]
  qi <- as.matrix(result$User[,rowindex])
  qj <- as.matrix(result$Movie[,columindex])
  r[i]<-t(qi)%*%qj
  }
w<-knn_r[data_train$movieindex]
```

```
data_train_linear<-as.data.frame(cbind(data_train$rating,r,w))
fit<-lm(V1~r+w,data=data_train_linear)
exp_rating<-predict(fit,data_train_linear)
(rmse_adj <- sqrt(mean((data_train_linear$V1 - exp_rating)^2)))
```

```
## [1] 0.5893299
```

```
cat("The RMSE train of the adjusted model is", rmse_adj)
```

```
## The RMSE train of the adjusted model is 0.5893299
```

prediction

```
data_test$movieindex<-dense_rank(data_test$movieId)

r1<-rep(0,nrow(data_test))
for (i in 1:nrow(data_test)){
  rowindex<-data_test$userId[i]
  columindex<-data_test$movieindex[i]
  qi <- as.matrix(result$User[,rowindex])
  qj <- as.matrix(result$Movie[,columindex])
  r1[i]<-t(qi)%*%qj
}

w1<-knn_r[data_test$movieindex]
```

```
data_test_linear<-as.data.frame(cbind(data_test$rating,r1,w1))
exp_rating_test<-predict(fit,data_test_linear)
rmse_adj_test <- sqrt(mean((data_test_linear$V1 - exp_rating_test)^2))
cat("The RMSE test of the adjusted model is", rmse_adj_test)
```

```
## The RMSE test of the adjusted model is 1.344724
```

Step 4 Evaluation

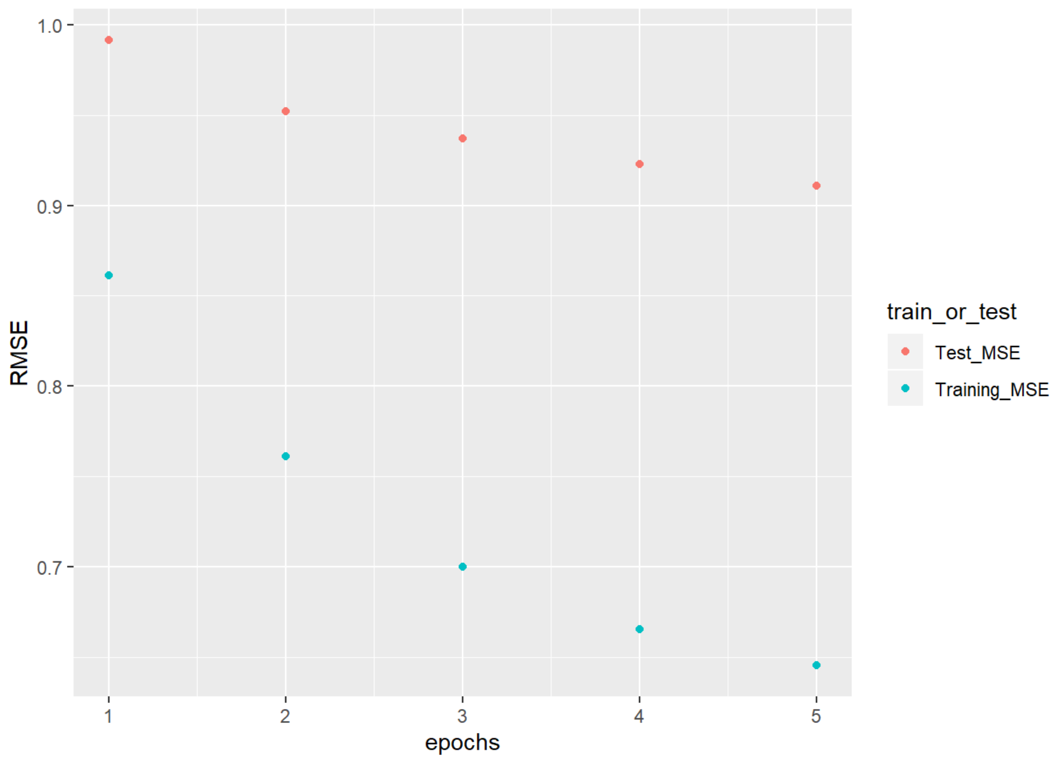
You should visualize training and testing RMSE by different dimension of factors and epochs (One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE).

```
library(ggplot2)

RMSE <- data.frame(epochs = seq(1, 5, 1), Training_MSE = result$train_RMSE, Test_MSE = result$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 = se
q(1, 5, 1)) + xlim(c(1, 5))
```

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


1.0
0.9
0.8
0.7
RMSE
1
2
3
4
5
epochs
train_or_test
Test_MSE
Training_MSE