# Project4

## Project 4, Group 1

For our project,

We do all the three parts.

Algorithm : Stochastic Gradient Descent (A1)

Regularization : Penalty of Magnitudes (R1) Bias and Intercepts (R2) Temporal Dynamics (R3)

Postprocessing : SVD with KNN (P2)

Our pairings:

1. A1 + P2
2. A1 + R1R2 + P2
3. A1 + R3 + P2

## Step 1 Load Data and Train-test Split

```
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
```

```
library(tidyr)
library(ggplot2)
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.

```
U <- length(unique(data[1:5000,]$userId))
I <- length(unique(data[1:5000,]$movieId))
source("../lib/Matrix_Factorization.R")
```

## Step 2.2 Parameter Tuning

Here we tune parameters, such as the dimension of factor and the penalty parameter $\lambda$ by cross-validation.

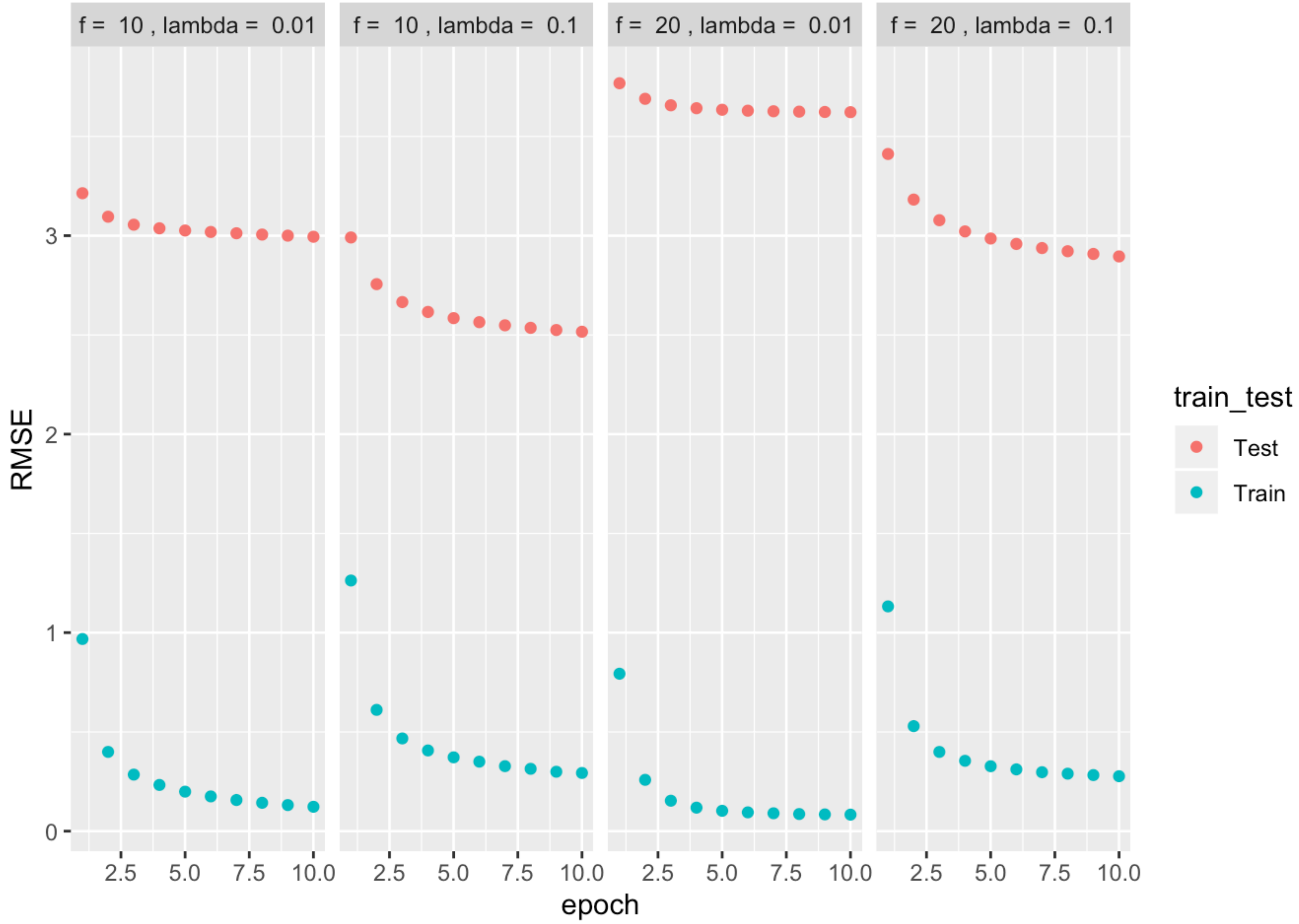
Because we want to maintain the consistancy, so we only use 5000 rows to tune parameter for SGD model.

### Step 2.2.1 Tunning parameter for A1 (only 5000 rows)

```
#result_summary_sgd_5000 <- 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[1:5000,], K = 5, f = f_l[i,1], lambda = 10^f_
l[i,2])
#    result_summary_sgd_5000[,,i] <- matrix(unlist(current_result), ncol = 10, byrow
= T)
#    print(result_summary_sgd_5000)

#})

#save(result_summary_sgd_5000, file = "../output/rmse_sgd_5000.Rdata")
```

Plot our tunning parameters for A1 (only 5000 rows)

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

load(file = "../output/rmse_sgd_5000.Rdata")
rmse <- data.frame(rbind(t(result_summary_sgd_5000[1,,]), t(result_summary_sgd_5000[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_gr
id(~par)
```

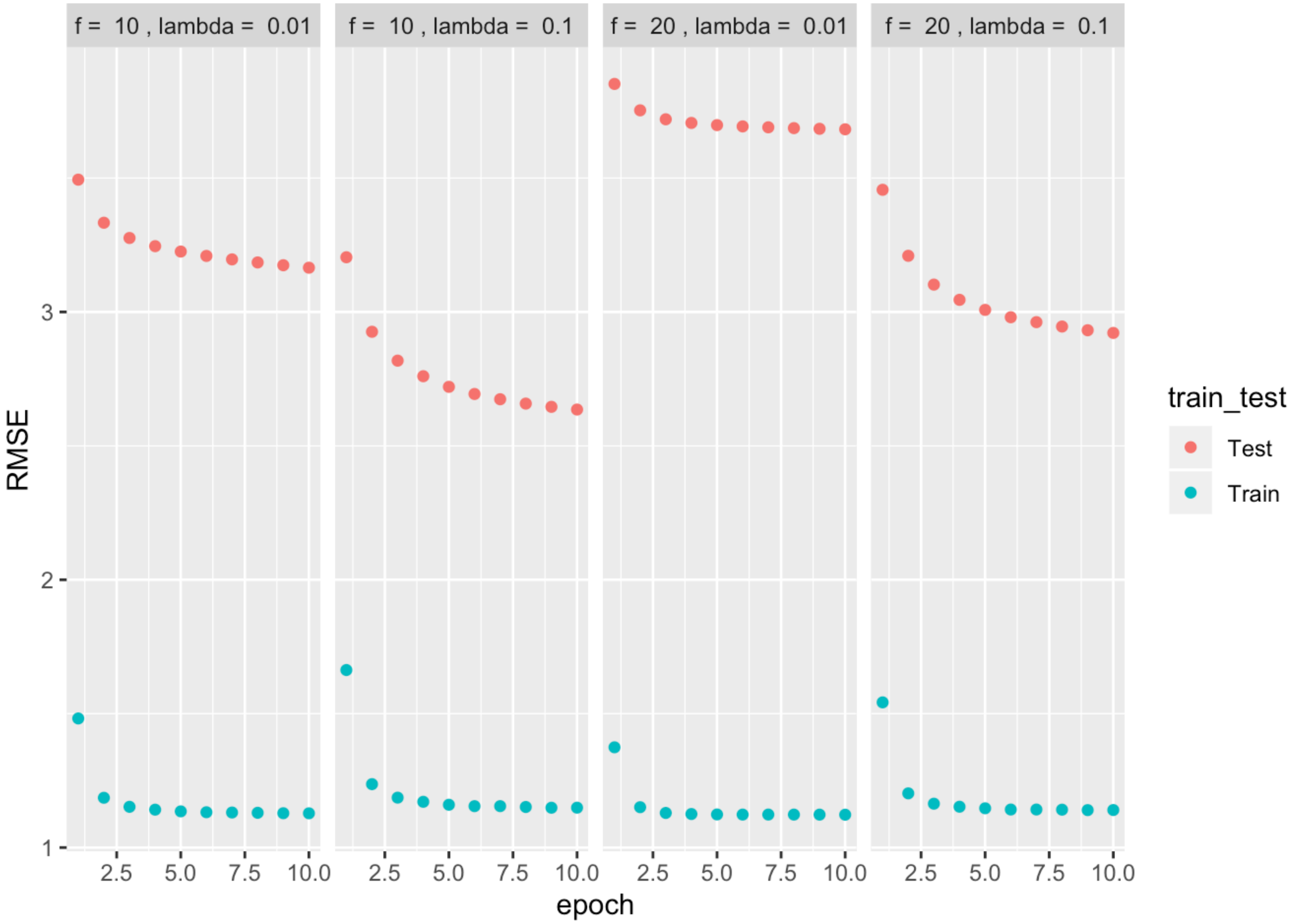



### Step 2.2.2 Tunning parameter for A1 + R1R2 (only 5000 rows)

```
source("../lib/cross_validation_r1+r2.R")
source("../lib/Matrix_Factorization_r1+r2.R")

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

# 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 our tunning parameters for A1 + R1R2 (only 5000 rows)

```
load(file = "../output/rmseR12.Rdata")
rmse <- data.frame(rbind(t(result_summary_r12[1,,]), t(result_summary_r12[2,,])), tra
in_test = rep(c("Train", "Test"), each = 4), par = rep(paste("f = ", f_l[,1], ", lamb
da = ", 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_gr
id(~par)
```

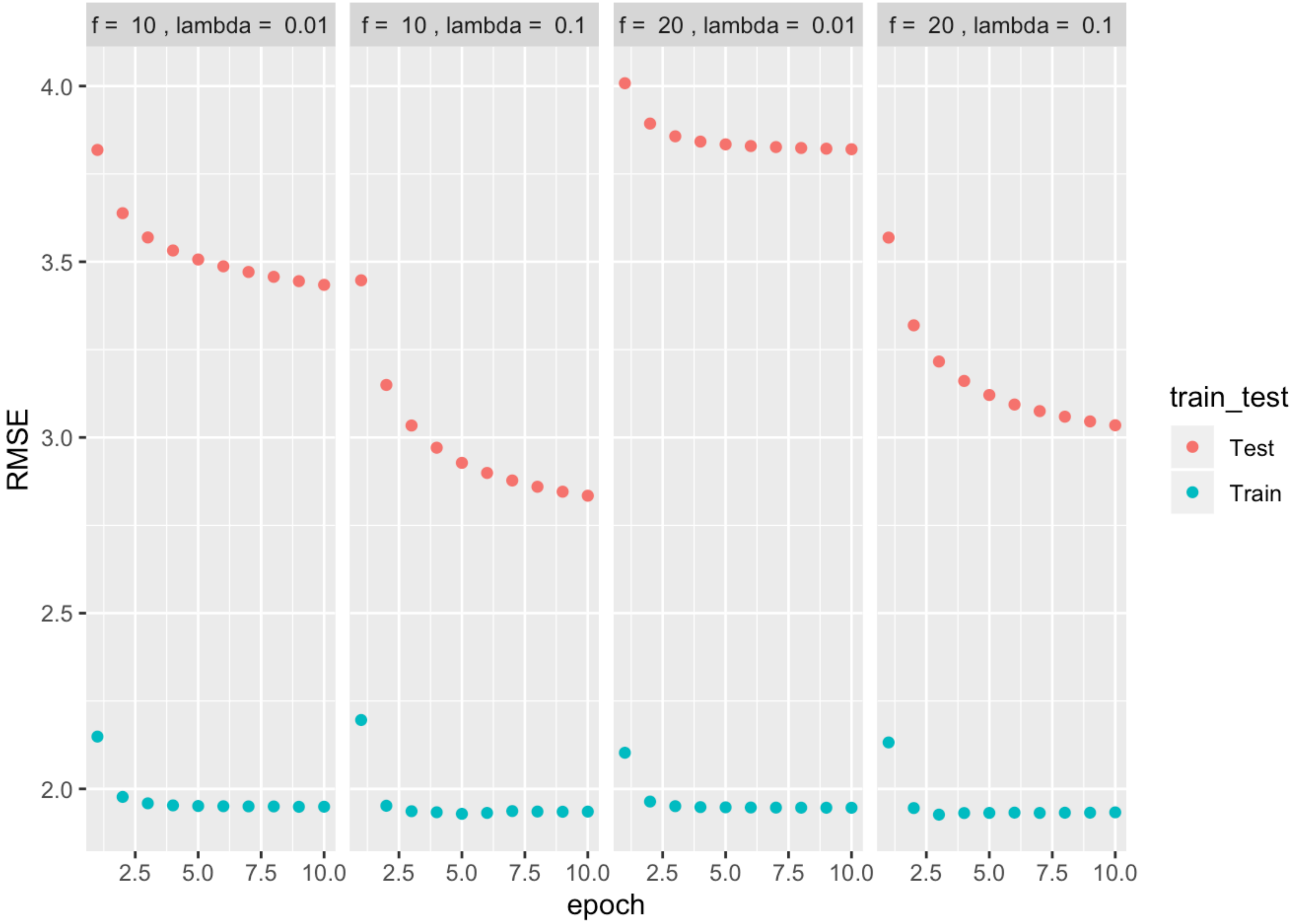



### Step 2.2.3 Tunning parameter for A1 + R3 (only 5000 rows)

```r
source("../lib/cross_validation_r3.R")
source("../lib/Matrix_Factorization_r3.R")

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

# result_summary_r3 <- 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.r3(data[1:5000,], K = 5, f = f_l[i,1], lambda = 1
0^f_l[i,2])
#     result_summary_r3[,,i] <- matrix(unlist(current_result), ncol = 10, byrow = T)
#     print(result_summary_r3)
#
# })
#
# save(result_summary_r3, file = "../output/rmseR3.Rdata")
```

Plot our tunning parameters for A1 + R3 (only 5000 rows)

```r
load(file = "../output/rmseR3.Rdata")
rmse <- data.frame(rbind(t(result_summary_r3[1,,]), t(result_summary_r3[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_gr
id(~par)
```




# Step 3 Postprocessing: SVD with KNN

After matrix factorization, postporcessing will be performed to improve accuracy.

RMSE Function

```
RMSE2 <- function(rating,est_rating){
  sqrt(mean((rating$rating-est_rating)^2))
}
```

KNN Function

```
vec <- function(x) {

  return(sqrt(sum(x^2)))

  }

pred_knn <- function(data_train, data_test, q)
{

  norm_q <- apply(q,2,vec)
  sim <- t(t((t(q) %*% q)/ norm_q) / norm_q)
  colnames(sim) <- colnames(q)
  rownames(sim) <- colnames(q)
  pred_test <- rep(0,nrow(data_test))

  for (i in 1:nrow(data_test)){
    user_id <- data_test$userId[i]
    movie_id <- data_test$movieId[i]
    train <- data_train[data_train$userId == user_id & data_train$movieId != movie_id
,]
    movie_train <- train$movieId
    sim_vec <- sim[rownames(sim) == movie_id, colnames(sim) %in% movie_train]
    movie <- names(sim_vec)[which.max(sim_vec)]
    pred_test[i] <- train[train$movieId == movie,][3]
  }

  pred_test <- as.matrix(unlist(pred_test))
  rmse_test <- sqrt(mean((data_test$rating-pred_test)^2))
  return(list(pred_test = pred_test, rmse_test = rmse_test))

}
```

## Step 3.1.1 RMSE for A1 (only 5000 rows)

Choose only 5000 rows for train data and test data.

```
sub_test_idx <- sample(1:5000, 5000/5, 0)
sub_train_idx <- setdiff(1:5000, sub_test_idx)
sub_data_train <- data[sub_train_idx,]
sub_data_test <- data[sub_test_idx,]
```

```
# result_sgd5000 <- gradesc(f = 10, lambda = 0.1,lrate = 0.01, max.iter = 100, stoppi
ng.deriv = 0.01,
#                     data = data[1:5000,], train = sub_data_train, test = sub_data_te
st)
#
# save(result_sgd5000, file = "../output/mat_fac_sgd5000.RData")

load(file = "../output/rmse_sgd_5000.Rdata")
load(file = "../output/mat_fac_sgd5000.Rdata")

pred_rating <- t(result_sgd5000$q) %*% result_sgd5000$p
rmse_sgd5000 <- RMSE2(sub_data_test,pred_rating)
cat("The RMSE of A1 model with only 5000 rows is", rmse_sgd5000)
```

```
## The RMSE of A1 model with only 5000 rows is 2.663459
```

## Step 3.1.2 RMSE for A1 + P2 (only 5000 rows)

```
load(file = "../output/mat_fac_sgd5000.Rdata")
q <- result_sgd5000$q
p2_result_test <- pred_knn(data_train = sub_data_train, data_test = sub_data_test, q)
test_rmse_p2 <- p2_result_test['rmse_test']
cat("The RMSE of A1 with P2 model with only 5000 rows is", as.numeric(test_rmse_p2))
```

```
## The RMSE of A1 with P2 model with only 5000 rows is 1.077033
```

## Step 3.2.1 RMSE for A1 + R1R2 (only 5000 rows)

```
# resultr12 <- gradesc.r12(f = 10, lambda = 0.1,lrate = 0.01, max.iter = 100, stoppin
g.deriv = 0.01,
#                     data = data[1:5000,], train = sub_data_train, test = sub_data_te
st)
#
# save(resultr12, file = "../output/mat_fac_r12.RData")
```

```
load(file = "../output/mat_fac_r12.RData")

pred_rating <- t(resultr12$q) %*% resultr12$p
rmse_r12 <- RMSE2(sub_data_test,pred_rating)
cat("The RMSE of A1 with R1 R2 model is", rmse_r12)
```

```
## The RMSE of A1 with R1 R2 model is 2.71485
```

## Step 3.2.2 RMSE for A1 + R1R2 + P2(only 5000 rows)

```
q <- resultr12$q
p2_result_test <- pred_knn(data_train = sub_data_train, data_test = sub_data_test, q)
test_rmse_p2 <- p2_result_test['rmse_test']
cat("The RMSE of A1 and R1 R2 with P2 model is", as.numeric(test_rmse_p2))
```

```
## The RMSE of A1 and R1 R2 with P2 model is 1.197289
```

### Step 3.3.1 RMSE for A1 + R3 (only 5000 rows)

```
# resultr3 <- gradesc.r3(f = 10, lambda = 0.1,lrate = 0.01, max.iter = 100, stopping.
deriv = 0.01,
#                     data = data[1:5000,], train = sub_data_train, test = sub_data_te
st)
#
# save(resultr3, file = "../output/mat_fac_r3.RData")
```

```
load(file = "../output/mat_fac_r3.RData")

pred_rating <- t(resultr3$q) %*% resultr3$p
rmse_r3 <- RMSE2(sub_data_test,pred_rating)
cat("The RMSE of A1 and R3 model is", rmse_r3)
```

```
## The RMSE of A1 and R3 model is 2.797368
```

### Step 3.3.2 RMSE for A1 + R3 + P2 (only 5000 rows)

```
q <- resultr3$q
p2_result_test <- pred_knn(data_train = sub_data_train, data_test = sub_data_test, q)
test_rmse_p2 <- p2_result_test['rmse_test']
cat("The RMSE of A1 and R3 with P2 model is", as.numeric(test_rmse_p2))
```

```
## The RMSE of A1 and R3 with P2 model is 1.283355
```

# 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 (https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9)).

```
library(ggplot2)

#SGD (5000 rows)
#RMSE <- data.frame(epochs = seq(10, 100, 10), Training_MSE = result_sgd5000$train_RM
SE, Test_MSE = result_sgd5000$test_RMSE) %>% gather(key = train_or_test, value = RMSE
, -epochs)

#RMSE %>% ggplot(aes(x = seq(10, 100, 10), y = RMSE,col = train_or_test)) + geom_poin
t() + scale_x_discrete(limits = seq(10, 100, 10)) + xlim(c(0, 100))

#SGD + R1R2
#RMSE <- data.frame(epochs = seq(10, 100, 10), Training_MSE = resultr12$train_RMSE, T
est_MSE = resultr12$test_RMSE) %>% gather(key = train_or_test, value = RMSE, -epochs)

#RMSE %>% ggplot(aes(x = epochs, y = RMSE,col = train_or_test)) + geom_point() + scal
e_x_discrete(limits = seq(10, 100, 10)) + xlim(c(0, 100))

#SGD + R3
#RMSE <- data.frame(epochs = seq(10, 100, 10), Training_MSE = resultr3$train_RMSE, Te
st_MSE = resultr3$test_RMSE) %>% gather(key = train_or_test, value = RMSE, -epochs)

#RMSE %>% ggplot(aes(x = epochs, y = RMSE,col = train_or_test)) + geom_point() + scal
e_x_discrete(limits = seq(10, 100, 10)) + xlim(c(0, 100))
```

# Step 5: Conclusions

Because we only use 5000 rows from original data to train our model, so our RMSE is a little bit higher than using the whole dataset.

Test RMSE for A1 : 2.663459

Test RMSE for A1 + P2 : 1.077033

Test RMSE for A1 + R1R2 : 2.71485

Test RMSE for A1 + R1R2 + P2 : 1.197289

Test RMSE for A1 + R3 : 2.797368

Test RMSE for A1 + R3 + P2 : 1.283355

And the best parameters for all models is F = 10, lambda = 0.1 .

After comparing all the results, we find A1 + P2 has the best performance.