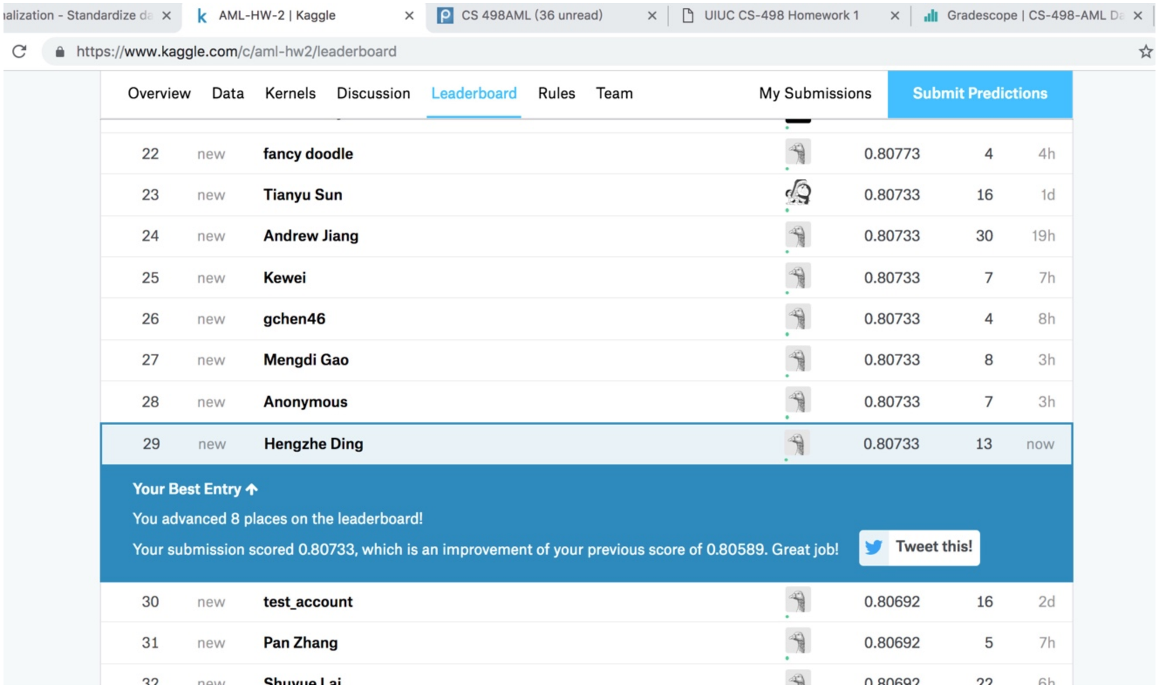
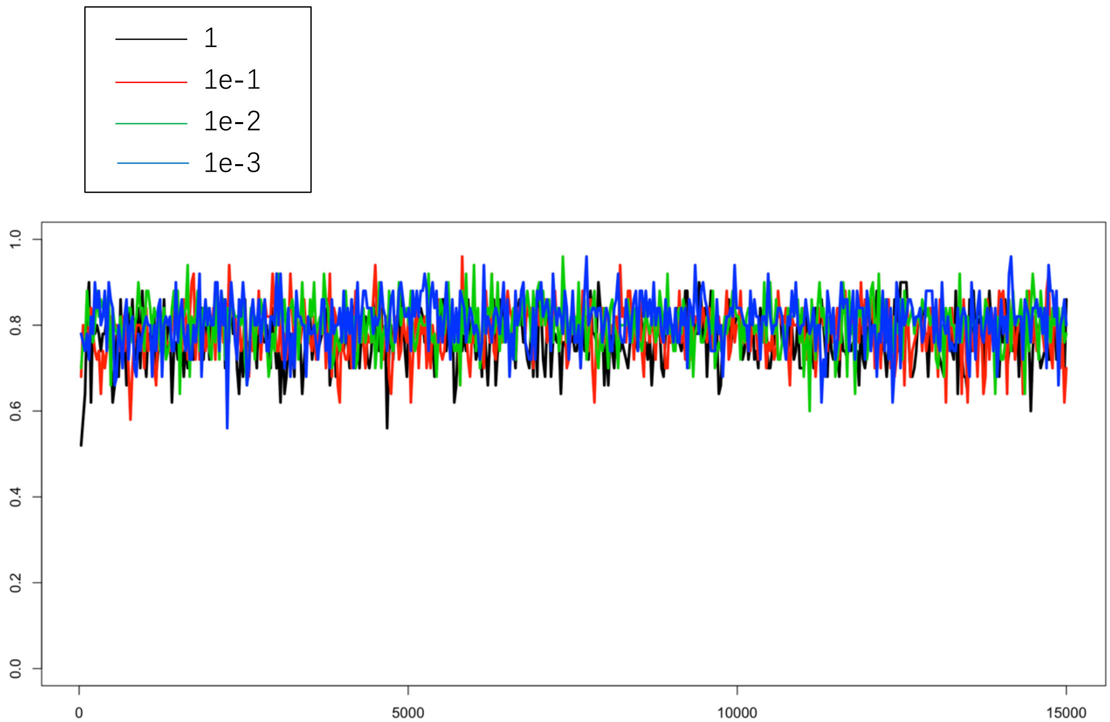
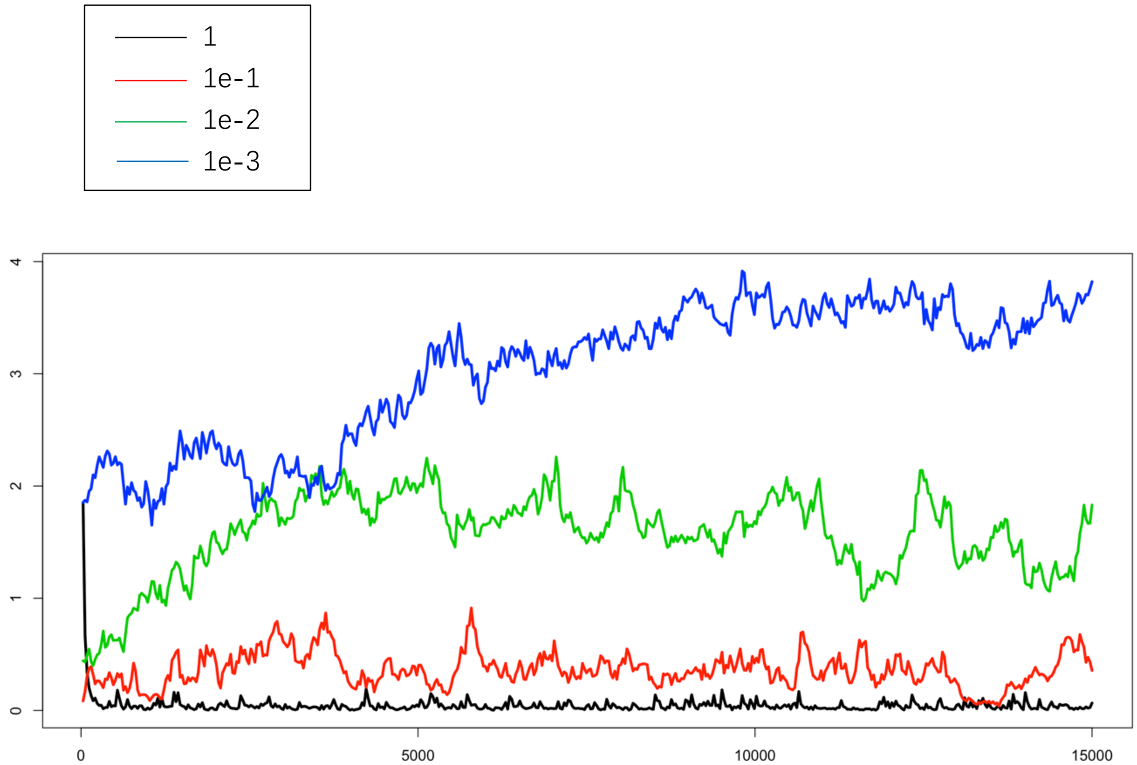


My best test dataset accuracy obtained on Kaggle is 80.733%.



The y axis is the accuracy and x axis is overall step count number.



The y axis is the magnitude of vector a and x axis is overall step count number.

The best value of the regularization constant is 1e-3. The reason why I think it is a good value is that the accuracy result is the best with this constant. In addition, little regularization constant means hyperplane with little margin between the two class, “-1” and “1”, which is good in this case, because the feature vector of two class is close to each other.

My choice of learning rate is 1/(0.01s+50).
Because in the equation of learning rate, m/(s+n), m should be small and n should be large.

Library used & normalizing data:

```r
library(caret)
# Take in data.
setwd("/Users/hengzhe/Library/Mobile Documents/com~apple~CloudDocs/UIUC/CS-498-Applie
train_data = read.table("train.csv", sep=",")
test_data = read.table("test.csv", sep=",")

# Function that normalize data.
translate_data <- function(input_data){
  for (i in c(1,3,5,11:13)){
    input_data[[sprintf("V%d", i)]] <- scale(input_data[[sprintf("V%d", i)]])
  }
  return (input_data)
}
```

SVM training:

```r
# Randomly split training data to 2 parts, 90% training part adn 10% validation part.
ind <- createDataPartition(train_data$V15, times=1, p=0.9, list=F)
val_train <- train_data[ind,]
val_test <- train_data[-ind,]

# start loop: this is a three layer loop, the first layer is to choose which regulatization constant has the highest
# accuracy rate, the second layer is season/epoch and the last layer is step.
for (m in 1:length(lambda_list)){
  lambda <- lambda_list[m]
  for (i in 1:Ns){
    # Setting the learning rate.
    g_s <- 1/(0.01*i+50)
    for (j in 1:step){
      index <- sample(1:nrow(val_train), 1)
      r <- val_train[index,]
      if (as.numeric(levels(r$V15)[r$V15])*(sum(a*r[c(1,3,5,11:13)])+b) >= 1){
        a <- a - g_s*lambda*a
        b <- b
      } else {
        a <- a - g_s*(lambda*a - as.numeric(levels(r$V15)[r$V15])*r[c(1,3,5,11:13)])
        b <- b + g_s * as.numeric(levels(r$V15)[r$V15])
      }
      if (j %% 30 == 0){
        val_sample <- val_test[sample(1:nrow(val_test), 50),]
        pred_sample <- c()
        for (q in 1:nrow(val_sample)){
          if(sum(a*val_sample[q,][c(1,3,5,11:13)])+b >= 0)
            pred_sample[q] <- 1
          else
            pred_sample[q] <- -1
        }
        pred_sample_table <- table(actual=val_sample$V15, predict=pred_sample)
        acc <- sum(diag(pred_sample_table))/sum(pred_sample_table)
        x[countxy] <- (i-1)*step + j
        y[countxy] <- acc
        mag_a[countxy] <- sum(a*a)
        countxy <- countxy + 1
      }
    }
  }

  # Calculate accuracy in different regularization constant and
  # choose the regularization constant with the highest accuracy.
  pred <- c()
  for (k in 1:nrow(val_test)){
    if (sum(a*val_test[k,][c(1,3,5,11:13)])+b >= 0)
      pred[k] <- 1
    else
      pred[k] <- -1
  }
  pred_table <- table(actual=val_test$V15, predict=pred)
  print(sum(diag(pred_table))/sum(pred_table))
  if (sum(diag(pred_table))/sum(pred_table) > accuracy){
    accuracy <- sum(diag(pred_table))/sum(pred_table)
    final_a <- a
    final_b <- b
    final_lambda <- lambda
  }
}
```