Adult Income prediction using SVM

CS 498 Applied ML

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

Column Names and data dictionary age: continous workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, epochsver-worked. fnlwgt: continuous. education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool. education-num: continuous. marital-status: Married-civ-spouse, Divorced, epochsver-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaepochsrs, Machiepochs-op-istepspct, Adm-clerical, Farming-fishing, Trastepsport-moving, Priv-house-serv, Protective-serv, Armed-Forces. relatiostepship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. sex: Female, Male. capital-gain: continuous. capital-loss: continuous. hours-per-week: continuous. native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippiepochss, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-epochstherlands.

```
data1 <- read.table("adult.data", sep = "," ,stringsAsFactors =FALSE)
data2 <- read.table("adult.test", sep = ",", stringsAsFactors= FALSE)
data <- rbind(data1,data2)
#Remove all categorical variables as assignment requires to only use continous variables
data <- data[,-c(1,2,4,6,7,8,9,10,14)]
#remove any leading and trailing whitespace. trimws is a builtin function in R that hand
les this problem
data <- data.frame(apply(data,2,function(x) trimws(x)), stringsAsFactors = FALSE)
#some values in Y also have trailing dot ".", so we epochsed to remove that as well
data[data[,6]=="<=50K" | data[,6]=="<=50K." ,6] <- -1
data[data[,6]==">50K" | data[,6]==">50K."  ,6] <- 1
#convert all strings to numeric
data <- data.frame(apply(data,2,function(x)as.numeric(x)), stringsAsFactors = FALSE)

tr <-createDataPartition(y = data[,6], p=0.8, list = FALSE )
train <- data[tr,]
other <- data [-tr,]
t <-createDataPartition(y = other[,6], p=0.5, list = FALSE )
validation <- other[t,]
test <-other[-t,]
```

```
dim(data)
```

```
## [1] 48842     6
```

```
head(data1)
```

```
##   V1                V2     V3         V4 V5                  V6
## 1 39         State-gov  77516  Bachelors 13       Never-married
## 2 50  Self-emp-not-inc  83311  Bachelors 13  Married-civ-spouse
## 3 38           Private 215646    HS-grad  9            Divorced
## 4 53           Private 234721       11th  7  Married-civ-spouse
## 5 28           Private 338409  Bachelors 13  Married-civ-spouse
## 6 37           Private 284582    Masters 14  Married-civ-spouse
##                  V7             V8     V9     V10  V11 V12 V13
## 1       Adm-clerical  Not-in-family  White    Male 2174   0  40
## 2    Exec-managerial        Husband  White    Male    0   0  13
## 3  Handlers-cleaners  Not-in-family  White    Male    0   0  40
## 4  Handlers-cleaners        Husband  Black    Male    0   0  40
## 5     Prof-specialty           Wife  Black  Female    0   0  40
## 6    Exec-managerial           Wife  White  Female    0   0  40
##             V14    V15
## 1  United-States  <=50K
## 2  United-States  <=50K
## 3  United-States  <=50K
## 4  United-States  <=50K
## 5           Cuba  <=50K
## 6  United-States  <=50K
```

```
dim(train)
```

```
## [1] 39074     6
```

```
dim(validation)
```

```
## [1] 4884    6
```

```
dim(test)
```

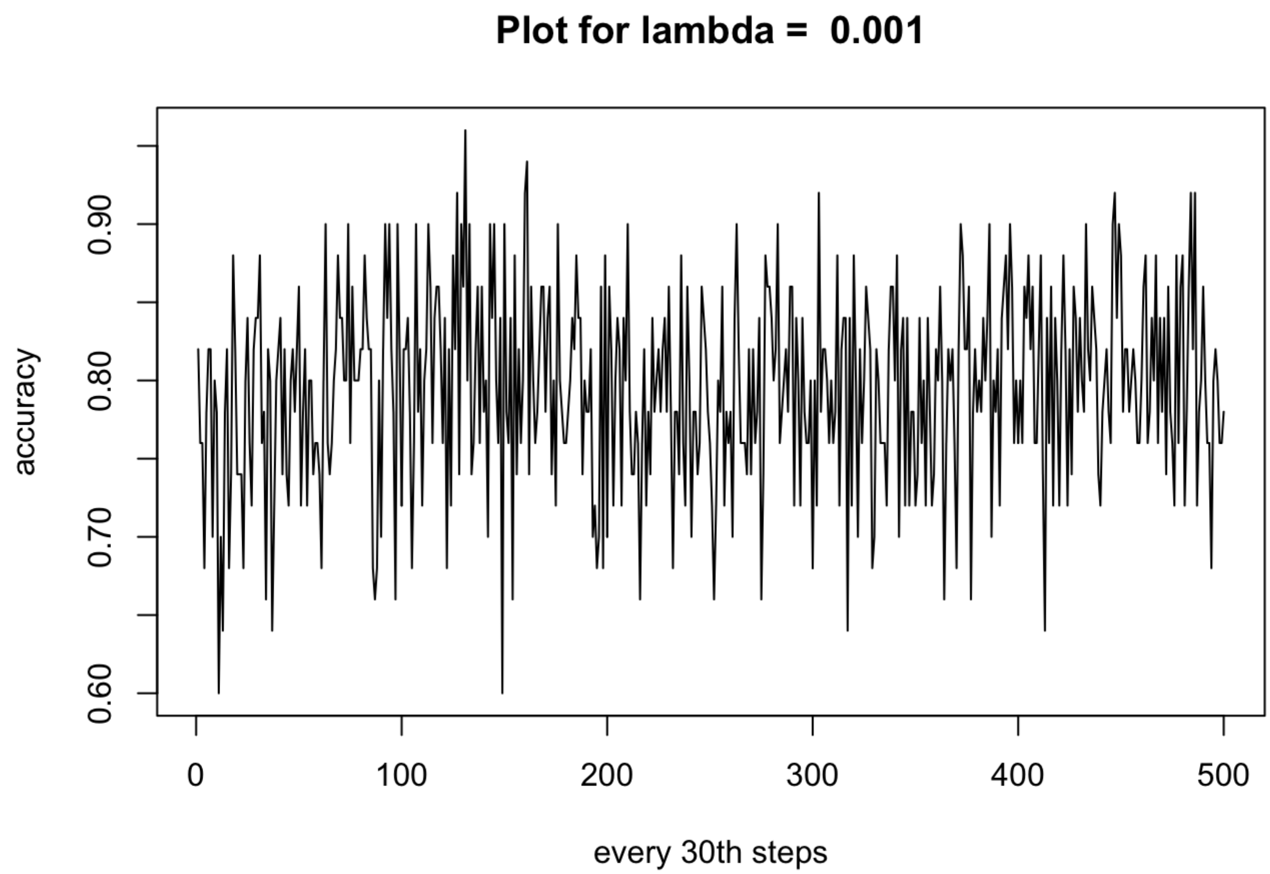
```
## [1] 4884    6
```

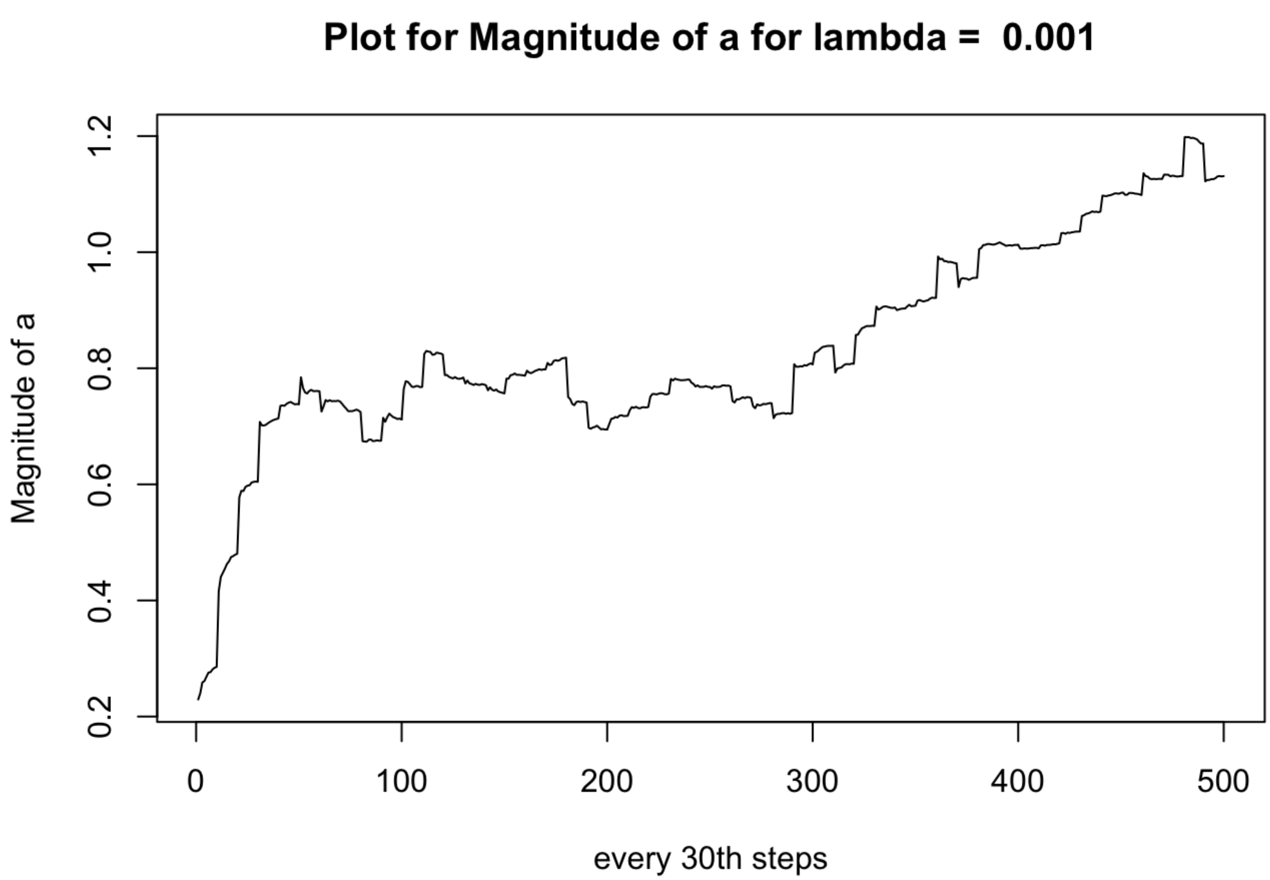
```r
trainX <- train [,-6]
trainY <-train [,6]
validationX <- validation [,-6]
validationY <-validation [,6]
testX <- test [,-6]
testY <-test [,6]


#Scale data using mean and sd of train
meanTrain <- sapply(trainX,mean)
sdTrain <- sapply(trainX,sd)
trainOffsets <- t(t(trainX) - meanTrain)
trainXScaled  <- t(t(trainOffsets) / sdTrain)
validationOffsets <- t(t(validationX) - meanTrain)
validationXScaled  <- t(t(validationOffsets) / sdTrain)
testOffsets <- t(t(testX) - meanTrain)
testXScaled  <- t(t(testOffsets) / sdTrain)
```

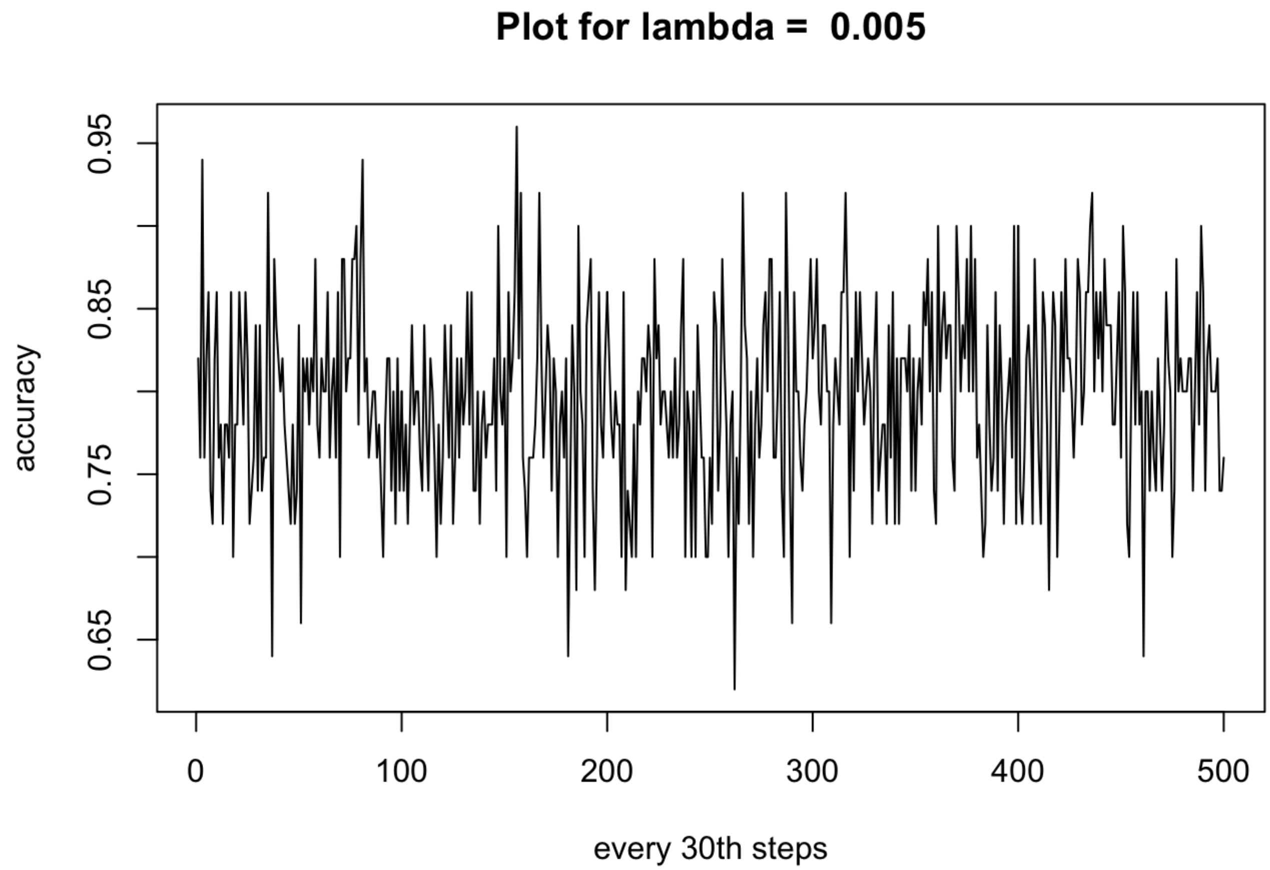
```r
epochs <- 50
steps <- 300
constant1<-0.1
constant2<-1
lambdas<-c(.001,.005, .01, .1, 1,3) # regularizer

accuracy <-function(Xtrain,Ytrain, a, b){
  sample50 <- sample(1:NROW(Xtrain),50)
  Xtrain <- Xtrain[sample50,]
  Ytrain <- Ytrain[sample50]
  ctr = 0
  for(i in 1:NROW(Xtrain)){
    #predict first
    gamma <-sum(Xtrain[i,]*a)+b
    if(gamma<0 & Ytrain[i]==-1){
      #correct
      ctr =ctr+1
    }else if(gamma>0 & Ytrain[i]==1){
      #correct
      ctr =ctr+1
    }else{
      #wrong
      ctr=ctr
    }

  }
  return(ctr/NROW(Xtrain))
}
```

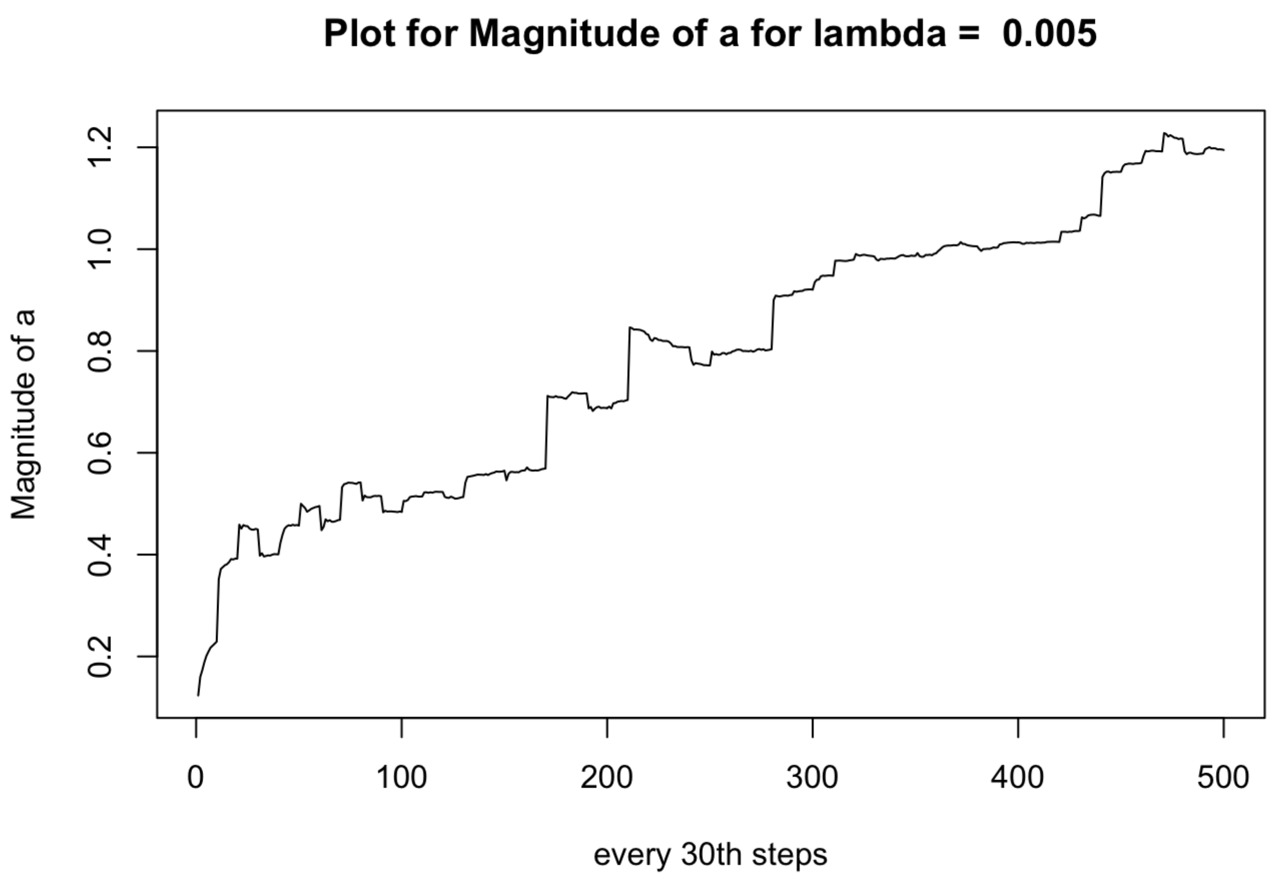
Train model on training set

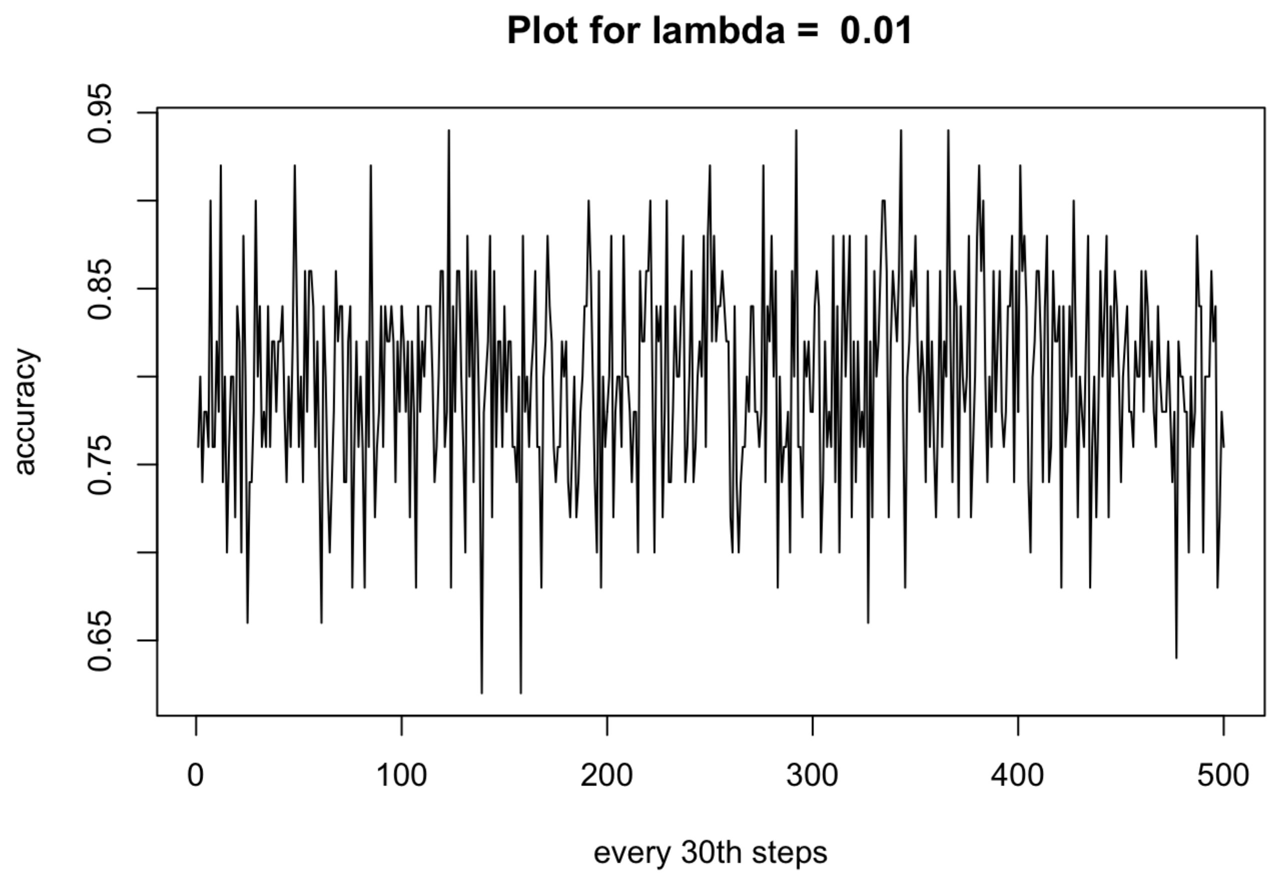
```
lambdaAccuracies<-c()

for(lambda in lambdas) {
  #goal is to predict set of a and b; start with both to be 0s
  #f(x)or gamma =ax+b
  a<-matrix(data=0, ncol=NCOL(trainXScaled))
  b<-0
  accuracies<-c()
  magnitude <- c()
  for (i in 1:epochs){

    for (j in 1:steps){
        n<-constant1/(j+constant2) #learning rate
        num<-sample(1:NROW(trainY),1)
        y<-trainY[num]
        x<-trainXScaled[num,]

        #f(x)/gamma = a*x+b
        gamma<-sum(a*x) + b

        #hingeloss = max(0,1 -y*f(x))
        if(y*gamma >= 1) {
         # cost will be 0; correctly classified
          regularization<-lambda*a
          a<-a-n*regularization
        }
        else {
          #page35
        regularization<-lambda*a
        delta<-n*(regularization-(y*x))
        #update x and y
        a<-a-delta
        b<-b-n*(-y)
        }
      #record accuracy at every 30th step
        if(j%%30==0) {
          accuracies<- append(accuracies,accuracy(trainXScaled, trainY, a, b))
          magnitude <- append(magnitude,norm(a, type="2"))
        }
    }
  }
  plot(accuracies,xlab="every 30th steps", ylab="accuracy",type='l')
  title(main = paste("Plot for lambda = ", lambda))

  plot(magnitude,xlab="every 30th steps", ylab="Magnitude of a",type='l')
  title(main = paste("Plot for Magnitude of a for lambda = ", lambda))

  lambdaAccuracies<- append(lambdaAccuracies,accuracy(validationXScaled, validationY, a,
 b))
}
```


Plot for lambda = 0.001
accuracy
0.60
0.70
0.80
0.90
0
100
200
300
400
500
every 30th steps

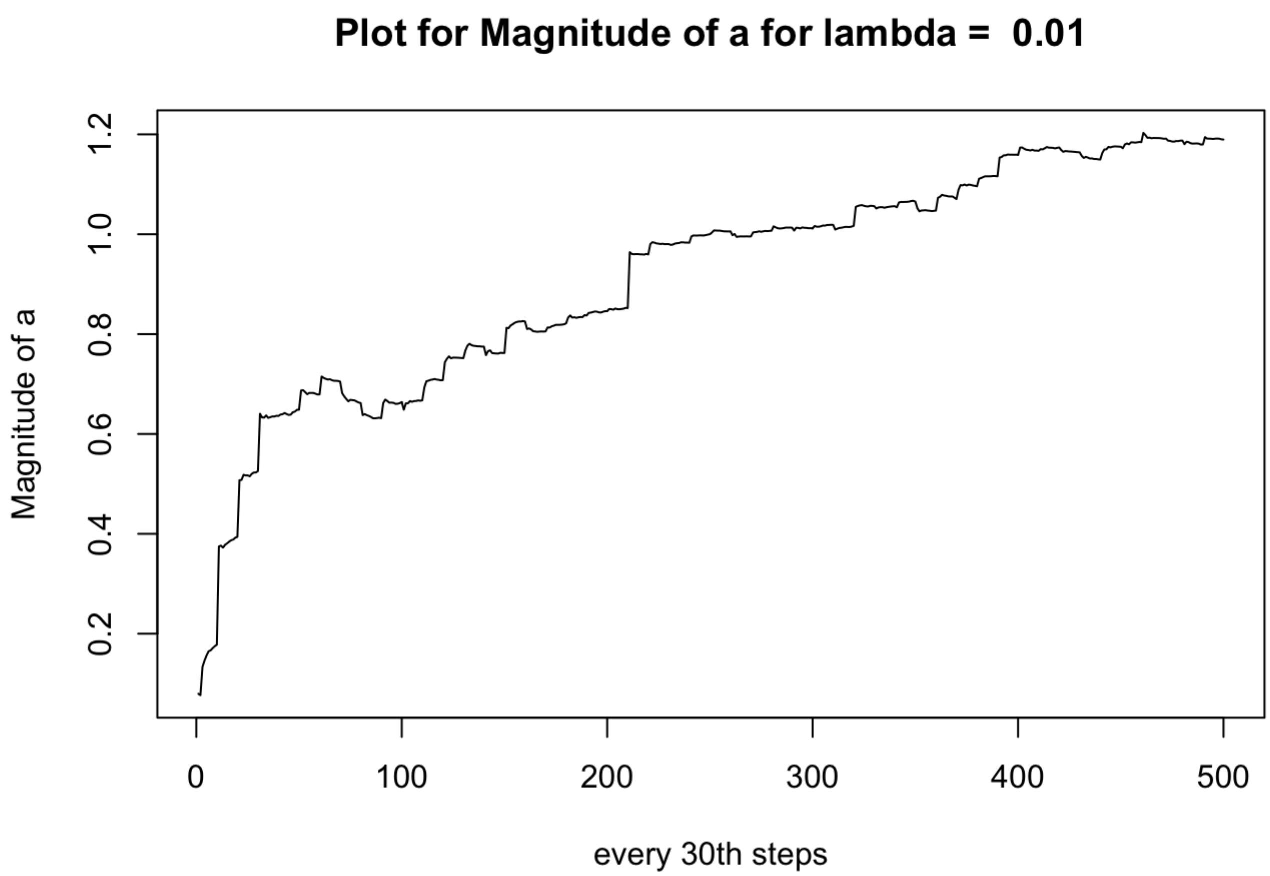

Plot for Magnitude of a for lambda = 0.001
Magnitude of a
0.2
0.4
0.6
0.8
1.0
1.2
0
100
200
300
400
500
every 30th steps


Plot for lambda = 0.005
accuracy
0.65
0.75
0.85
0.95
0
100
200
300
400
500
every 30th steps

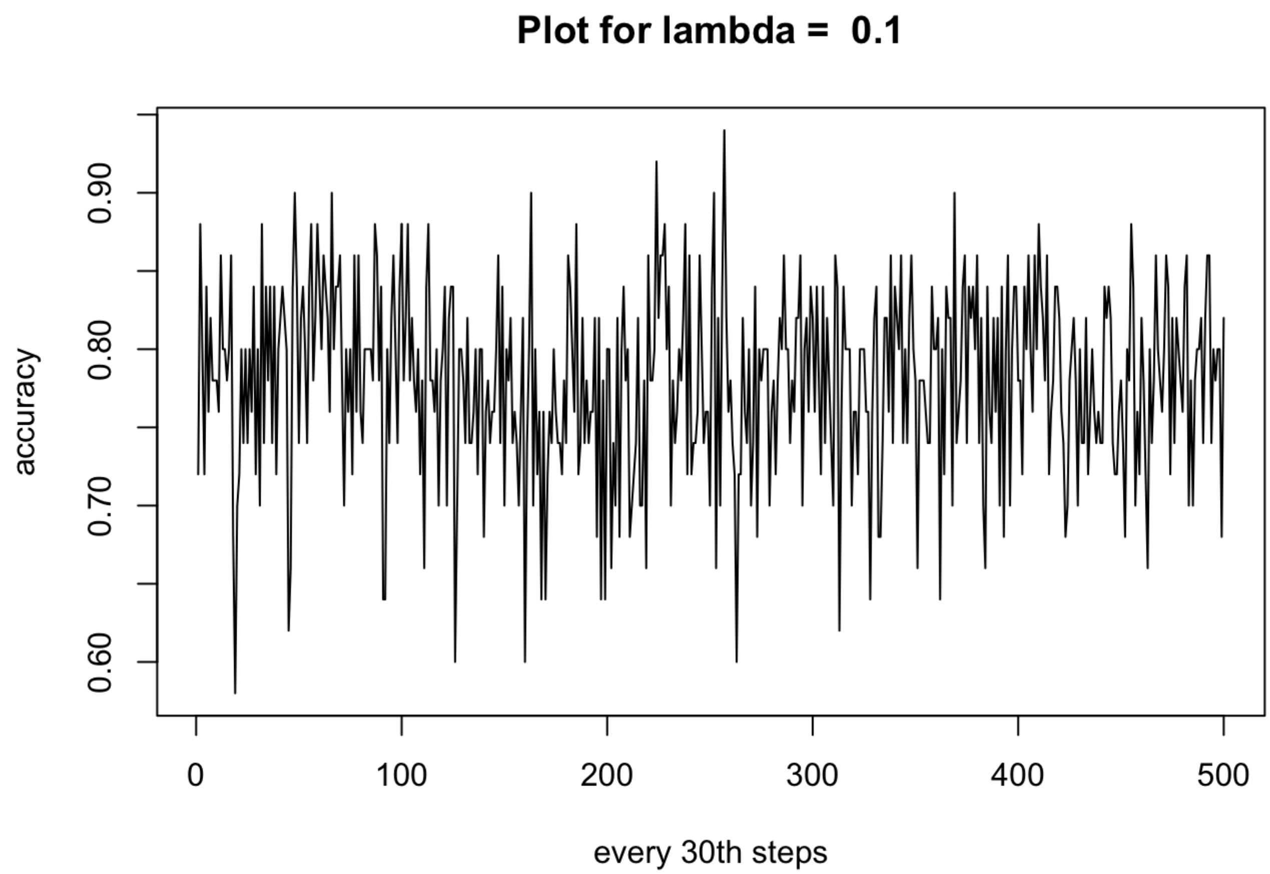

Plot for Magnitude of a for lambda = 0.005
Magnitude of a
0.2
0.4
0.6
0.8
1.0
1.2
0
100
200
300
400
500
every 30th steps

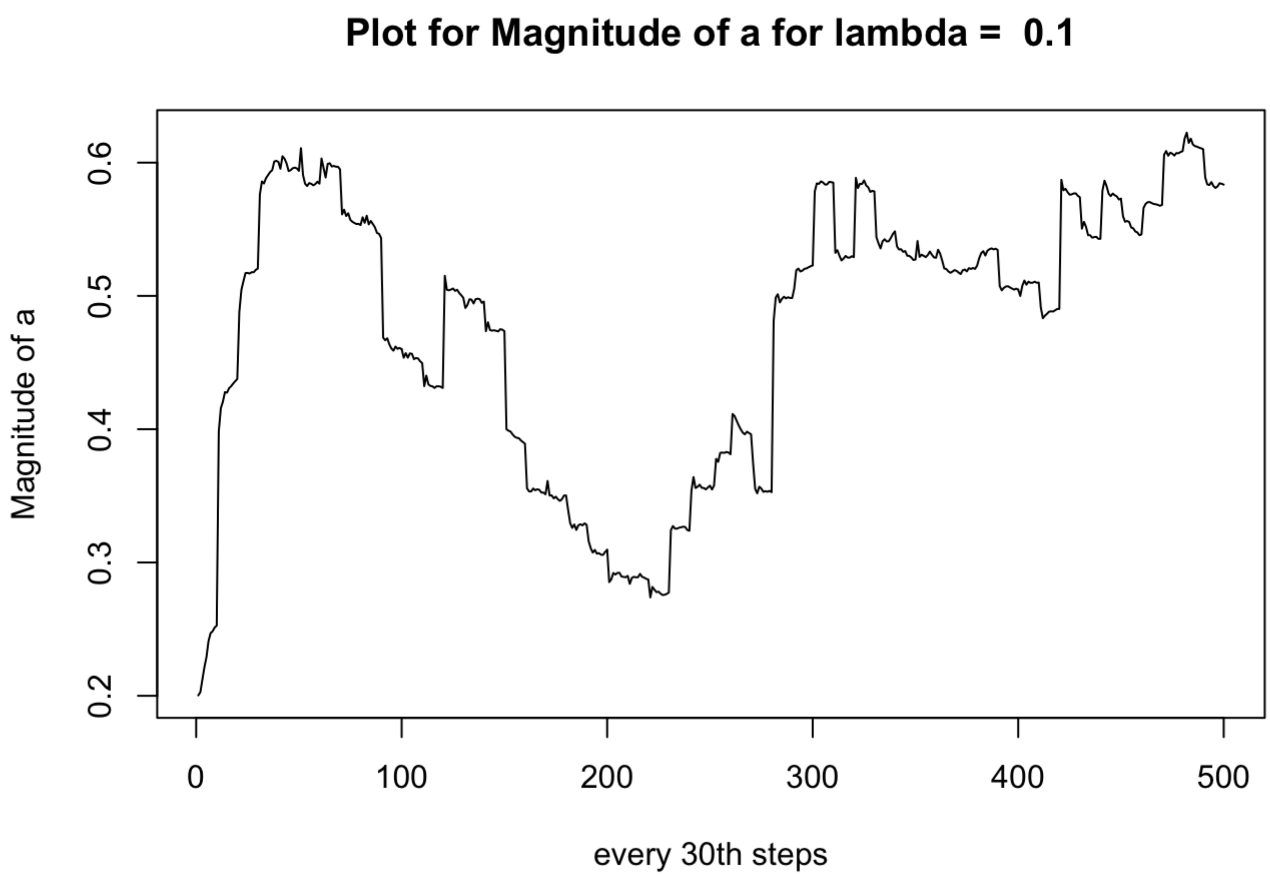


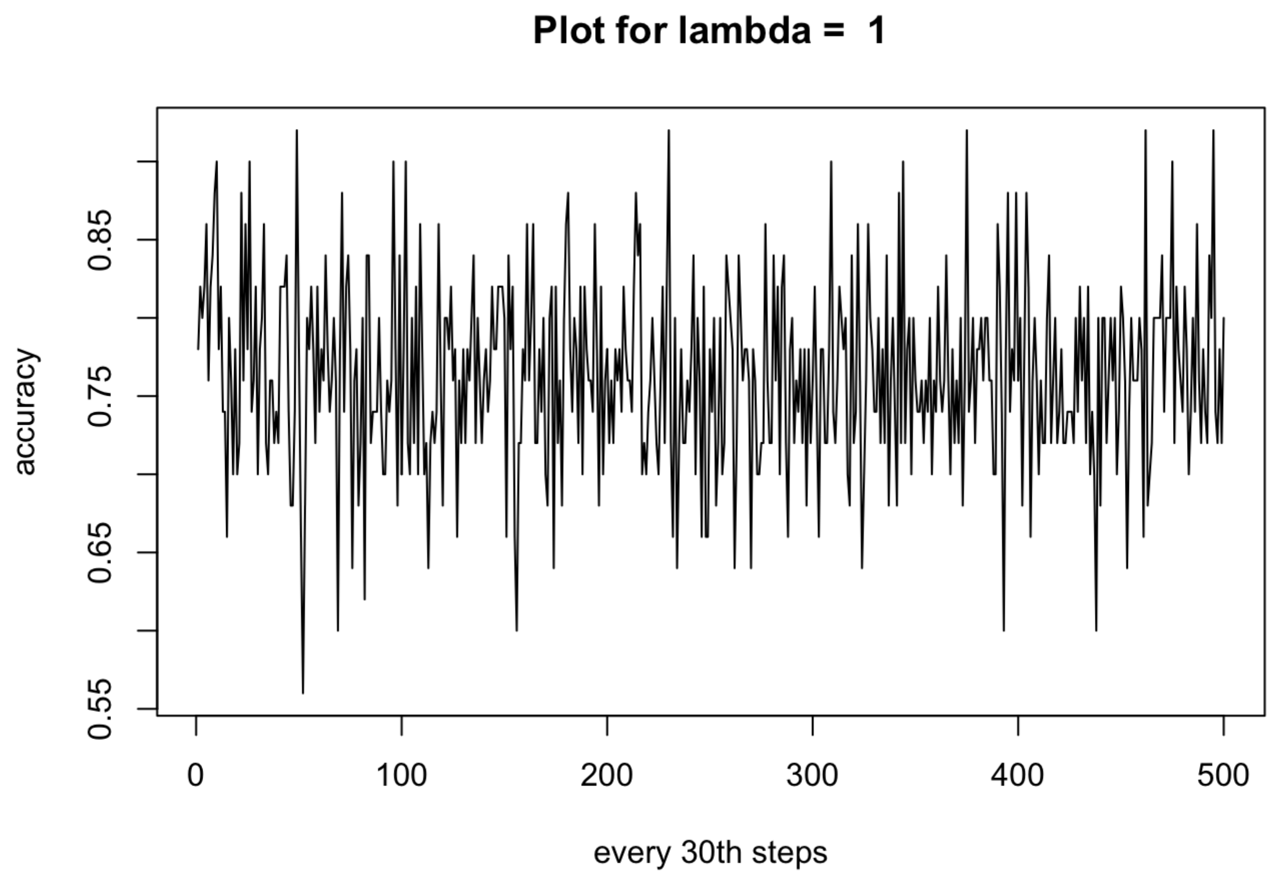
Plot for lambda = 0.01
accuracy
0.65
0.75
0.85
0.95
0
100
200
300
400
500
every 30th steps

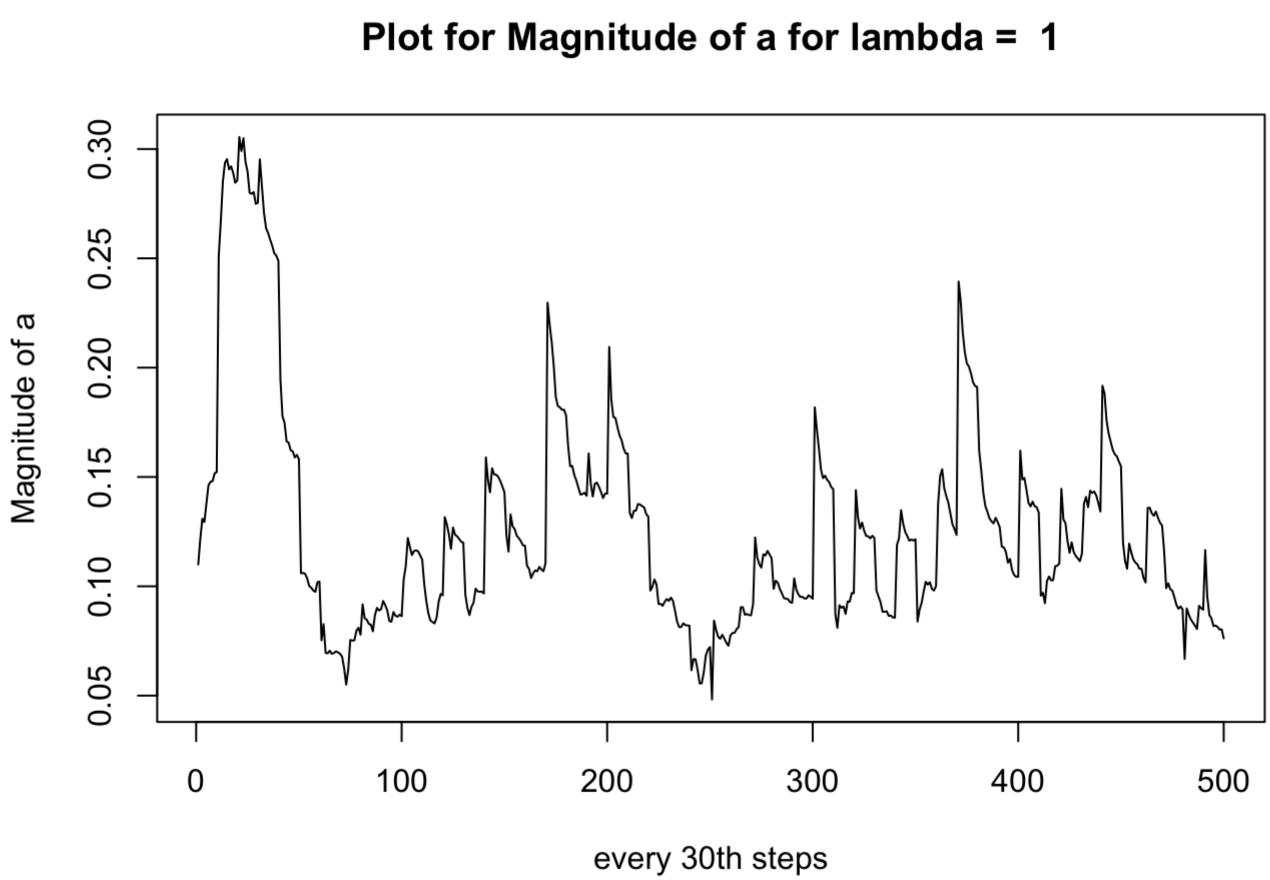


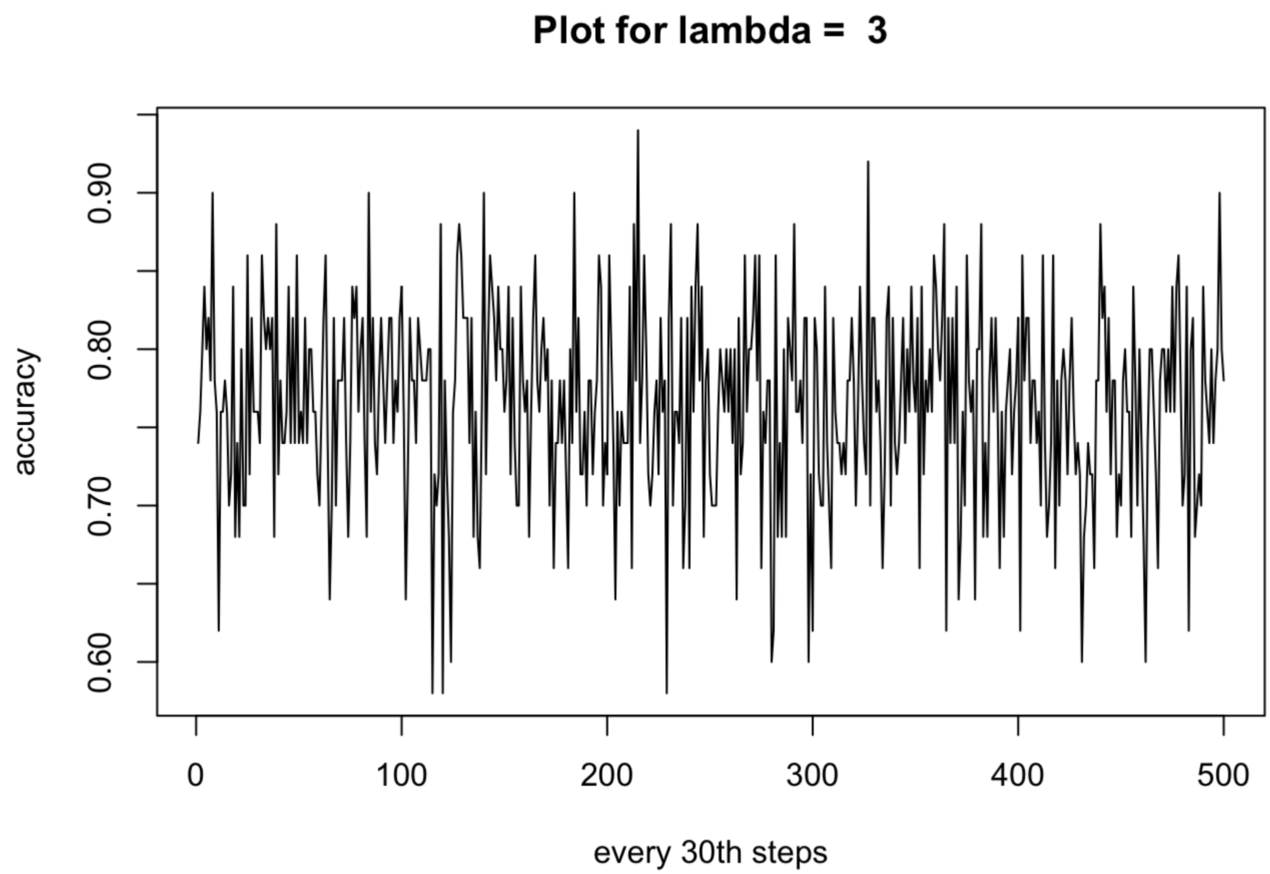
Plot for Magnitude of a for lambda = 0.01
Magnitude of a
0.2
0.4
0.6
0.8
1.0
1.2
0
100
200
300
400
500
every 30th steps

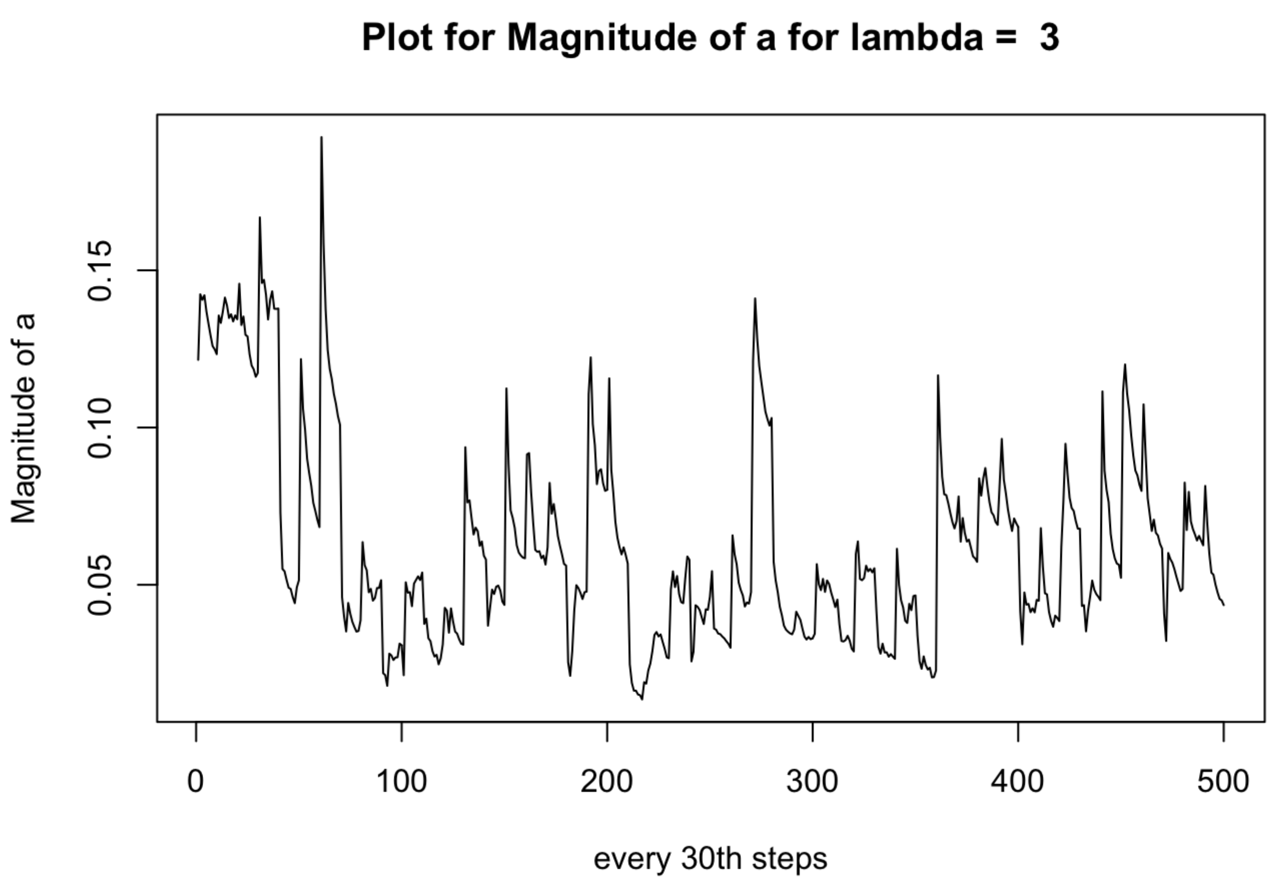

Plot for lambda = 0.1
accuracy
0.60
0.70
0.80
0.90
0
100
200
300
400
500
every 30th steps


Plot for Magnitude of a for lambda = 0.1
Magnitude of a
0.2
0.3
0.4
0.5
0.6
0
100
200
300
400
500
every 30th steps

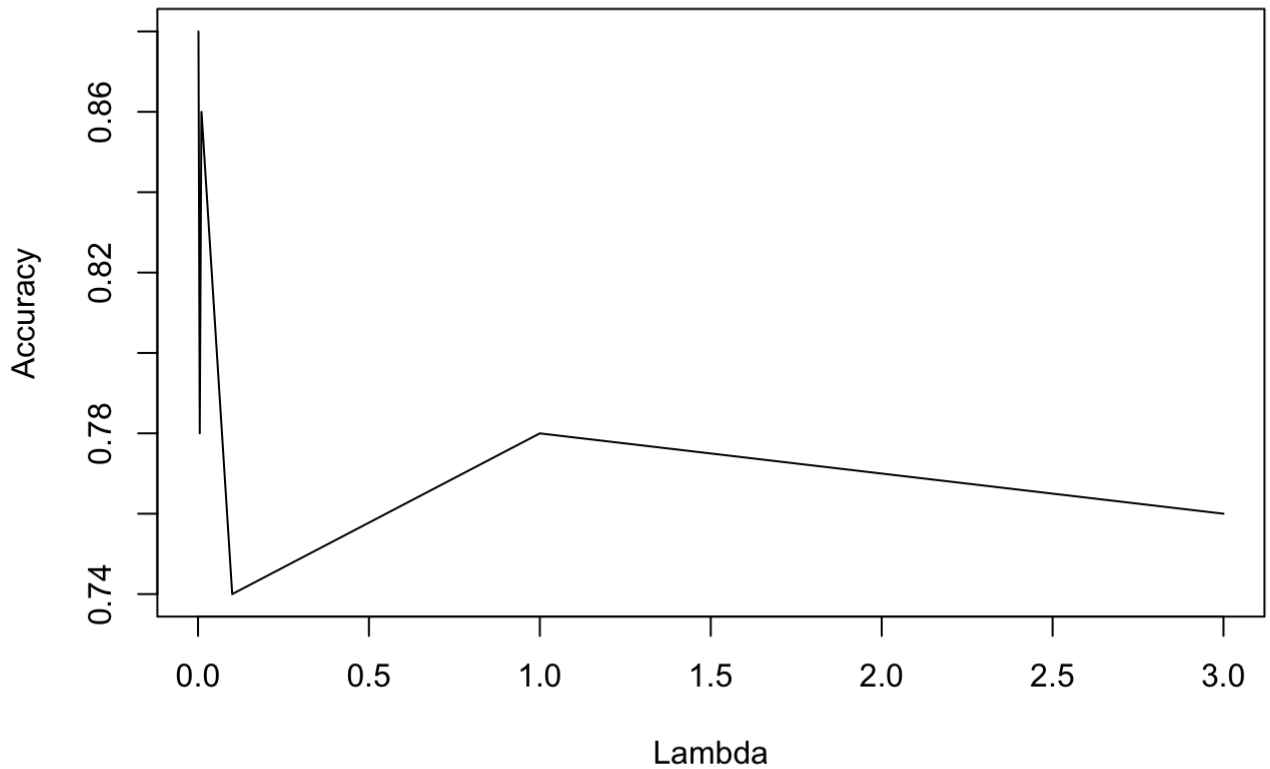

Plot for lambda = 1
accuracy
0.55
0.65
0.75
0.85
0
100
200
300
400
500
every 30th steps


Plot for Magnitude of a for lambda = 1
Magnitude of a
0.05
0.10
0.15
0.20
0.25
0.30
0
100
200
300
400
500
every 30th steps


Plot for lambda = 3
accuracy
0.60
0.70
0.80
0.90
0
100
200
300
400
500
every 30th steps


Plot for Magnitude of a for lambda = 3
Magnitude of a
0.05
0.10
0.15
0
100
200
300
400
500
every 30th steps

```
plot(x = lambdas, y = lambdaAccuracies, ylab = "Accuracy", xlab = "Lambda", type = "l")
```



Regularizer parameter serves as a degree of importance that is given to the missed classification. By having larger lambda, our classfier model will overfit its training data. By using smaller lambdam our classifer model will be more generalized. From the above comparison that we did on validation set using different regularizer parameter, we have found out that 0.01 regularizer gives the best accuracy in the validation. From this analysis, we will conlude to use regularizer of 0.01 to predict our test case

```r
lambda <- 0.01
a<-matrix(data=0, ncol=NCOL(trainXScaled))
b<-0
accuracies<-c()
magnitude <- c()
for (i in 1:epochs){
  for (j in 1:steps){
    n<-constant1/(j+constant2) #learning rate
    num<-sample(1:NROW(trainY),1)
    y<-trainY[num]
    x<-trainXScaled[num,]

    #f(x)/gamma = a*x+b
    gamma<-sum(a*x) + b

    #hingeloss = max(0,1 -y*f(x))
    if(y*gamma >= 1) {
     # cost will be 0; correctly classified
      regularization<-lambda*a
      a<-a-n*regularization
    }
    else {
      regularization<-lambda*a
      delta<-n*(regularization-(y*x))
      #update x and y
      a<-a-delta
      b<-b-n*(-y)
    }
  }
}
print(paste("Test set accuracy: ",accuracy(testXScaled, testY, a, b)))
```

```
## [1] "Test set accuracy:  0.88"
```