





After running the code 7 times, I obtained the following table of cross-validation accuracies

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Average accuracy |
|  | 0.809235 | 0.813607 | 0.815874 | 0.806334 | 0.79973 | 0.81139 | 0.805773 | 0.808849 |
|  | 0.812034 | 0.805219 | 0.814501 | 0.809752 | 0.79973 | 0.80309 | 0.795032 | 0.805623 |
|  | 0.800373 | 0.800559 | 0.814959 | 0.80041 | 0.795005 | 0.802859 | 0.797046 | 0.801602 |
|  | 0.764692 | 0.786114 | 0.791857 | 0.797904 | 0.768902 | 0.763662 | 0.786082 | 0.779887 |

It seems that the accuracies from the first three values of are relatively close. However, looking at the held-out accuracy plot, the value seemed to be getting the best accuracies. So I decided to go with that.

The learning rate was , where is the season number.. This was chosen, because as the number of seasons increases, the step size decreases, because we assume that the algorithm is close to converging.

batch\_size <- 1

#Split the data into 90% training, and 10% validation data

split <- test\_train\_split(nrow(training\_data))

held\_out\_accuracies\_lambda\_1 <- c()

coefficient\_magnitude\_1 <- c()

a\_1 <- c(0,0,0,0,0,0)

b\_1 <- 1

for (k in 1:100) {

#print(paste("Lengths of a and b:", length(a), length(b)))

step\_size <- 1/(0.01\*k + 50)

ret\_vec <- do\_season(split[[1]], 1e-4, a\_1, b\_1, 1, step\_size)

#print(paste("ret\_vec:",length(ret\_vec)))

a\_1 <- ret\_vec[11:16]

b\_1 <- ret\_vec[17]

#print(c(a,b))

held\_out\_accuracies\_lambda\_1 <- c(held\_out\_accuracies\_lambda\_1, ret\_vec[1:10])

coefficient\_magnitude\_1 <- c(coefficient\_magnitude\_1, ret\_vec[18:27])

print(paste("Done season", k))

}

test\_predictions\_1 <- lapply(split[[2]], function(x) {return(ifelse(sum(a\_1\*training\_data[x,])+b\_1> 0, 1, -1))})

accuracy\_1 <- length(which(test\_predictions\_1 == labels[split[[2]]]))/length(split[[2]])

…

held\_out\_accuracies\_lambda\_4 <- c()

coefficient\_magnitude\_4 <- c()

a\_4 <- c(0,0,0,0,0,0)

b\_4 <- 0

for (k in 1:100) {

#print(paste("Lengths of a and b:", length(a), length(b)))

step\_size <- 1/(0.01\*k + 50)

ret\_vec <- do\_season(split[[1]], 1e-1, a\_4, b\_4, 1, step\_size)

#print(paste("ret\_vec:",length(ret\_vec)))

a\_4 <- ret\_vec[11:16]

b\_4 <- ret\_vec[17]

#print(c(a,b))

held\_out\_accuracies\_lambda\_4 <- c(held\_out\_accuracies\_lambda\_4, ret\_vec[1:10])

coefficient\_magnitude\_4 <- c(coefficient\_magnitude\_4, ret\_vec[18:27])

print(paste("Done season", k))

}

test\_predictions\_4 <- lapply(split[[2]], function(x) {return(ifelse(sum(a\_4\*training\_data[x,])+b\_4 > 0, 1, -1))})

accuracy\_4 <- length(which(test\_predictions\_4 == labels[split[[2]]]))/length(split[[2]])

#Performs a step of the stochastic gradient descent process. Returns a vector containing

#a and b.

do\_step <- function(rows, lambda, a, b, batch\_size, step\_size) {

batch <- sample(rows, 1)

gamma <- labels[batch]\*(sum(a\*unlist(training\_data[batch,])) + b)

if (gamma >= 1) {

a <- a - step\_size\*lambda\*a

} else {

a <- a - step\_size\*(lambda\*a - labels[batch]\*unlist(training\_data[batch,]))

b <- b - step\_size\*(-labels[batch])

}

return(c(a,b))

}

#Does an iteration of a season. Involves taking 300 steps. Every 30 steps, it checks

#the current a and b values agains the hold-out set, and keeps track of the accuracy.

#It also keeps track of the magnitude of the vector a. The function returns

#the accuracies (every 30 steps), a, b, and the magnitude of a (every 30 steps)

do\_season <- function(rows, lambda, a, b, batch\_size, step\_size) {

#Get held out set for the season

hold\_out <- sample(rows, 50, replace = F)

#Vectors to store accuracies agains hold out set and magnitude of a, every 30 steps.

store\_accuracies <- c()

mag\_a <- c()

#Take 300 steps. Every 30 steps, calculate accuracy of held out set.

for (step in 1:300) {

if ((step > 0) && ((step %% 30) == 0)) {

#Test agains hold out set. Store the accuracy in a vector.

predictions <- lapply(hold\_out, function(x) {return(ifelse(sum(a\*training\_data[x,])+b > 0, 1, -1))})

store\_accuracies <- c(store\_accuracies, length(which(predictions == labels[hold\_out]))/50)

mag\_a <- c(mag\_a, sum(a\*a))

}

#Get updated values after stepping.

new\_u <- do\_step(rows, lambda, a, b, batch\_size, step\_size)

a <- new\_u[1:6]

b <- new\_u[7]

}

return(c(store\_accuracies, a, b, mag\_a))

}

**Full Code**

library(ggplot2)

library(dplyr)

library(tidyr)

################DATA SETUP#####################

training\_data <- read.csv("training\_data.csv")

training\_data <- training\_data[,c(1,3,5,11,12,13,15)]

testing\_data <- read.csv("testing\_data.csv")

testing\_data <- testing\_data[,c(1,3,5,11,12,13)]

#Scales the feature variables to have mean of 0 and variance of 1

training\_means <- apply(training\_data[,1:6], 2, mean)

training\_sds <- apply(training\_data[,1:6], 2, sd)

testing\_means <- apply(testing\_data, 2, mean)

testing\_sds <- apply(testing\_data, 2, sd)

#training\_data[,1:6] <- ((training\_data[,1:6] - training\_means)/training\_vars)

training\_data[,1:6] <- as.data.frame(Map("-", training\_data[,1:6], training\_means))

training\_data[,1:6] <- as.data.frame(Map("/", training\_data[,1:6], training\_sds))

#testing\_data <- apply(testing\_data, 2, function(x) {return((x - mean(x))/var(x))})

testing\_data <- as.data.frame(Map("-", testing\_data, testing\_means))

testing\_data <- as.data.frame(Map("/", testing\_data, testing\_sds))

names(training\_data) <- c("age", "fnlwgt", "edu-num", "gain", "loss", "hours", "income")

income <- training\_data[,7]

labels <- ifelse(income == income[1], 1, -1)

training\_data <- training\_data[,1:6]

names(testing\_data) <- c("age", "fnlwgt", "edu-num", "gain", "loss", "hours")

######################FUNCTIONS#########################

#Given a number, gives ~90% of them a lable of 1 and 10% of them a

#label of 0. The number of rows in a data frame will be passed here

#to split them into test-train groups.

test\_train\_split <- function(num\_rows) {

newCol <- rbinom(n = num\_rows, size = 1, prob = 0.9)

split <- list(which(newCol == 1), which(newCol == 0))

return(split)

}

#Performs a step of the stochastic gradient descent process. Returns a vector containing

#a and b.

do\_step <- function(rows, lambda, a, b, batch\_size, step\_size) {

batch <- sample(rows, 1)

gamma <- labels[batch]\*(sum(a\*unlist(training\_data[batch,])) + b)

if (gamma >= 1) {

a <- a - step\_size\*lambda\*a

} else {

a <- a - step\_size\*(lambda\*a - labels[batch]\*unlist(training\_data[batch,]))

b <- b - step\_size\*(-labels[batch])

}

return(c(a,b))

}

#Does an iteration of a season. Involves taking 300 steps. Every 30 steps, it checks

#the current a and b values agains the hold-out set, and keeps track of the accuracy.

#It also keeps track of the magnitude of the vector a. The function returns

#the accuracies (every 30 steps), a, b, and the magnitude of a (every 30 steps)

do\_season <- function(rows, lambda, a, b, batch\_size, step\_size) {

#Get held out set for the season

hold\_out <- sample(rows, 50, replace = F)

#Vectors to store accuracies agains hold out set and magnitude of a, every 30 steps.

store\_accuracies <- c()

mag\_a <- c()

#Take 300 steps. Every 30 steps, calculate accuracy of held out set.

for (step in 1:300) {

if ((step > 0) && ((step %% 30) == 0)) {

#Test agains hold out set. Store the accuracy in a vector.

predictions <- lapply(hold\_out, function(x) {return(ifelse(sum(a\*training\_data[x,])+b > 0, 1, -1))})

store\_accuracies <- c(store\_accuracies, length(which(predictions == labels[hold\_out]))/50)

mag\_a <- c(mag\_a, sum(a\*a))

}

#Get updated values after stepping.

new\_u <- do\_step(rows, lambda, a, b, batch\_size, step\_size)

a <- new\_u[1:6]

b <- new\_u[7]

}

return(c(store\_accuracies, a, b, mag\_a))

}

########LEARNING AND CROSS VALIDATION##############

batch\_size <- 1

#Split the data into 90% training, and 10% validation data

split <- test\_train\_split(nrow(training\_data))

held\_out\_accuracies\_lambda\_1 <- c()

coefficient\_magnitude\_1 <- c()

a\_1 <- c(0,0,0,0,0,0)

b\_1 <- 1

for (k in 1:100) {

#print(paste("Lengths of a and b:", length(a), length(b)))

step\_size <- 1/(0.01\*k + 50)

ret\_vec <- do\_season(split[[1]], 1e-4, a\_1, b\_1, 1, step\_size)

#print(paste("ret\_vec:",length(ret\_vec)))

a\_1 <- ret\_vec[11:16]

b\_1 <- ret\_vec[17]

#print(c(a,b))

held\_out\_accuracies\_lambda\_1 <- c(held\_out\_accuracies\_lambda\_1, ret\_vec[1:10])

coefficient\_magnitude\_1 <- c(coefficient\_magnitude\_1, ret\_vec[18:27])

print(paste("Done season", k))

}

test\_predictions\_1 <- lapply(split[[2]], function(x) {return(ifelse(sum(a\_1\*training\_data[x,])+b\_1> 0, 1, -1))})

accuracy\_1 <- length(which(test\_predictions\_1 == labels[split[[2]]]))/length(split[[2]])

held\_out\_accuracies\_lambda\_2 <- c()

coefficient\_magnitude\_2 <- c()

a\_2 <- c(0,0,0,0,0,0)

b\_2 <- 0

for (k in 1:100) {

#print(paste("Lengths of a and b:", length(a), length(b)))

step\_size <- 1/(0.01\*k + 50)

ret\_vec <- do\_season(split[[1]], 1e-3, a\_2, b\_2, 1, step\_size)

#print(paste("ret\_vec:",length(ret\_vec)))

a\_2 <- ret\_vec[11:16]

b\_2 <- ret\_vec[17]

#print(c(a,b))

held\_out\_accuracies\_lambda\_2 <- c(held\_out\_accuracies\_lambda\_2, ret\_vec[1:10])

coefficient\_magnitude\_2 <- c(coefficient\_magnitude\_2, ret\_vec[18:27])

print(paste("Done season", k))

}

test\_predictions\_2 <- lapply(split[[2]], function(x) {return(ifelse(sum(a\_2\*training\_data[x,])+b\_2 > 0, 1, -1))})

accuracy\_2 <- length(which(test\_predictions\_2 == labels[split[[2]]]))/length(split[[2]])

held\_out\_accuracies\_lambda\_3 <- c()

coefficient\_magnitude\_3 <- c()

a\_3 <- c(0,0,0,0,0,0)

b\_3 <- 0

for (k in 1:100) {

#print(paste("Lengths of a and b:", length(a), length(b)))

step\_size <- 1/(0.01\*k + 50)

ret\_vec <- do\_season(split[[1]], 1e-2, a\_3, b\_3, 1, step\_size)

#print(paste("ret\_vec:",length(ret\_vec)))

a\_3 <- ret\_vec[11:16]

b\_3 <- ret\_vec[17]

#print(c(a,b))

held\_out\_accuracies\_lambda\_3 <- c(held\_out\_accuracies\_lambda\_3, ret\_vec[1:10])

coefficient\_magnitude\_3 <- c(coefficient\_magnitude\_3, ret\_vec[18:27])

print(paste("Done season", k))

}

test\_predictions\_3 <- lapply(split[[2]], function(x) {return(ifelse(sum(a\_3\*training\_data[x,])+b\_3 > 0, 1, -1))})

accuracy\_3 <- length(which(test\_predictions\_3 == labels[split[[2]]]))/length(split[[2]])

held\_out\_accuracies\_lambda\_4 <- c()

coefficient\_magnitude\_4 <- c()

a\_4 <- c(0,0,0,0,0,0)

b\_4 <- 0

for (k in 1:100) {

#print(paste("Lengths of a and b:", length(a), length(b)))

step\_size <- 1/(0.01\*k + 50)

ret\_vec <- do\_season(split[[1]], 1e-1, a\_4, b\_4, 1, step\_size)

#print(paste("ret\_vec:",length(ret\_vec)))

a\_4 <- ret\_vec[11:16]

b\_4 <- ret\_vec[17]

#print(c(a,b))

held\_out\_accuracies\_lambda\_4 <- c(held\_out\_accuracies\_lambda\_4, ret\_vec[1:10])

coefficient\_magnitude\_4 <- c(coefficient\_magnitude\_4, ret\_vec[18:27])

print(paste("Done season", k))

}

test\_predictions\_4 <- lapply(split[[2]], function(x) {return(ifelse(sum(a\_4\*training\_data[x,])+b\_4 > 0, 1, -1))})

accuracy\_4 <- length(which(test\_predictions\_4 == labels[split[[2]]]))/length(split[[2]])

###########################PLOT GENERATION###########################33

df\_coefficient\_magnitudes <- data.frame(coefficient\_magnitude\_1, coefficient\_magnitude\_2,

coefficient\_magnitude\_3, coefficient\_magnitude\_4)

names(df\_coefficient\_magnitudes) <- c("1e-4", "1e-3", "1e-2", "1e-1")

df\_coefficient\_magnitudes %>%

gather(Var, Val) %>%

mutate(x = rep(1:1000, 4)) %>%

ggplot(aes(x, Val)) +

geom\_line(aes(color = Var)) +

xlab("Steps (incremented by 30)") +

ylab("Magnitude of Coefficient") +

ggtitle("Coefficient Magnitude Variation for \nMultiple Regularization Constants") +

scale\_color\_manual(values = c("black", "blue", "green", "yellow"))

df\_accuracies <- data.frame(held\_out\_accuracies\_lambda\_1, held\_out\_accuracies\_lambda\_2,

held\_out\_accuracies\_lambda\_3, held\_out\_accuracies\_lambda\_4)

names(df\_accuracies) <- c("1e-4", "1e-3", "1e-2", "1e-1")

df\_accuracies %>%

gather(Var, Val) %>%

mutate(x = rep(1:1000, 4)) %>%

ggplot(aes(x, Val)) +

geom\_line(aes(color = Var)) +

xlab("Steps (incremented by 30)") +

ylab("Held out accuracy") +

ggtitle("Held Out Accuracies with \nMultiple Regularization Constants") +

scale\_color\_manual(values = c("light green", "black", "light blue", "light yellow"))

#################FINAL TEST FOR SUBMISSION######################

#I trained on the entire data set, and did more seasons so that I could be more confident

#that we saw all the points, and of convergence.

a\_final <- c(0,0,0,0,0,0)

b\_final <- 0

for (k in 1:450) {

#print(paste("Lengths of a and b:", length(a), length(b)))

step\_size <- 1/(0.01\*k + 50)

ret\_vec <- do\_season(c(1:nrow(training\_data)), 1e-2, a\_final, b\_final, 1, step\_size)

#print(paste("ret\_vec:",length(ret\_vec)))

a\_final <- ret\_vec[11:16]

b\_final <- ret\_vec[17]

print(paste("Done season", k))

}

test\_predictions\_final <- lapply(c(1:nrow(testing\_data)), function(x) {return(ifelse(sum(a\_final\*testing\_data[x,])+b\_final > 0, 1, -1))})

test\_predictions\_final\_labels <- lapply(test\_predictions\_final, function(x) {return(ifelse(x == 1, ">50K", "<=50K"))})

sink("submission.txt")

rapply(test\_predictions\_final\_labels, print)

sink()

**REFERENCES**

Received help from the following Stack Overflow post

<https://stackoverflow.com/questions/54543101/r-ggplot2-legend-not-appearing-for-line-graph?noredirect=1#comment95888194_54543101>