HW13

Jackie Liang

December 10, 2018

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.4.4

##   
## 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(caret)

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.4

library(nnet)  
  
if (!exists("mtrain")) {  
 mtrain <- read.csv("mnist\_train.csv", header=F) %>% as.matrix  
 train\_classification <- mtrain[,1]  
 y <- factor(train\_classification, levels=c(0,1))  
 mtrain <- mtrain[,-1]/256 #this is the x matirix  
   
 colnames(mtrain) <- 1:(28^2)  
   
 x <- mtrain[1:1000,]  
   
}  
  
  
for (i in 1:length(train\_classification)){  
 cn <- train\_classification[i]  
 if (cn==3){  
 cn <- 1  
 } else {  
 cn <- 0  
 }  
 y[i] <- cn  
}  
  
y <- factor(y, levels = c(0,1))  
y <- y[1:1000]

# caret 0 decay

tuning\_df3 <- data.frame(size=5, decay=0)  
tuning\_df1 <- data.frame(size=1, decay=0)  
tuning\_df2 <- data.frame(size=3, decay=0)  
  
fitControl <- trainControl(## 2-fold CV  
 method = "repeatedcv",  
 number = 2,  
 repeats = 2)  
  
t\_out1 <- caret::train(x=x, y=y, method="nnet",  
 trControl = fitControl,  
 tuneGrid=tuning\_df1, maxit=1000, MaxNWts=10000)  
  
t\_out2 <- caret::train(x=x, y=y, method="nnet",  
 trControl = fitControl,  
 tuneGrid=tuning\_df2, maxit=1000, MaxNWts=10000)  
  
t\_out3 <- caret::train(x=x, y=y, method="nnet",  
 trControl = fitControl,  
 tuneGrid=tuning\_df3, maxit=1000, MaxNWts=10000)  
  
y\_fit3 <- predict(t\_out3, x)  
y\_fit1 <- predict(t\_out1, x)  
y\_fit2 <- predict(t\_out2, x)  
  
true\_y <- y  
  
#model 1  
n\_samples <- nrow(x)  
error1 <- sum(true\_y != y\_fit1)/n\_samples  
pred\_error1 <- error1  
  
pred\_error1  
  
error2 <- sum(true\_y != y\_fit2)/n\_samples  
pred\_error2 <- error2  
  
pred\_error2  
  
error3 <- sum(true\_y != y\_fit3)/n\_samples  
pred\_error3 <- error3  
  
pred\_error3  
  
#Models with decay  
  
tuning1 <- data.frame(size=3, decay=0)  
tuning2 <- data.frame(size=3, decay=.5)  
tuning3 <- data.frame(size=3, decay=1)  
  
  
model1 <- caret::train(x=x, y=y, method="nnet",  
 trControl = fitControl,  
 tuneGrid=tuning1, maxit=1000, MaxNWts=10000)  
model2 <- caret::train(x=x, y=y, method="nnet",  
 trControl = fitControl,  
 tuneGrid=tuning2, maxit=1000, MaxNWts=10000)  
model3 <- caret::train(x=x, y=y, method="nnet",  
 trControl = fitControl,  
 tuneGrid=tuning3, maxit=1000, MaxNWts=10000)

error4 <- sum(true\_y != model1)/n\_samples  
pred\_error4 <- error4  
pred\_error4

## [1] 1

error5 <- sum(true\_y != model2)/n\_samples  
pred\_error5 <- error5  
pred\_error5

## [1] 1

error6 <- sum(true\_y != model3)/n\_samples  
pred\_error6 <- error6  
pred\_error6

## [1] 1

# Checking against mnist\_test

if (!exists("mtrain2")) {  
 mtrain2 <- read.csv("mnist\_test.csv", header=F) %>% as.matrix  
 train\_classification2 <- mtrain2[,1]  
 mtrain2 <- mtrain2[,-1]/256 #x values  
   
   
 colnames(mtrain2) <- 1:(28^2)  
 rownames(mtrain2) <- NULL  
   
 x2 <- mtrain2[1:1000,]  
}  
  
y2 <- rep(NA, length(train\_classification))  
#Converting all threes to one and all other numbers to zero  
for (i in 1:length(train\_classification2)){  
 cn <- train\_classification2[i]  
 if (cn==3){  
 cn <- 1  
 } else {  
 cn <- 0  
 }  
 y2[i] <- cn  
}  
  
y2 <- factor(y, levels=c(0,1))  
y2 <- y[1:1000]

true\_y2 <- y2  
pred\_y2 <- predict(t\_out1, x2)  
n\_samples2 <- nrow(x2)  
  
error\_a <- sum(true\_y2 != pred\_y2)/n\_samples2  
pred\_error\_a <- error\_a  
  
pred\_error\_a

## [1] 0.19