Johnson_HW13

Daniel Johnson

December 9, 2018

MNist

Making mnist train and mnist test

```
###Making the mtrain matrix###
library(dplyr)
if (!exists("mtrain")) {
  mtrain <- read.csv("mnist_train.csv", header=F) %>% as.matrix
  train_classification <- mtrain[,1] #Go through this vector, set equal to zero if not 3 and 1 if 3. Gi
  mtrain <- mtrain[,-1]/256 #x values


  colnames(mtrain) <- 1:(28^2)
  rownames(mtrain) <- NULL

  x <- mtrain[1:1000,]
}
y <- rep(NA, length(train_classification))

#Converting all threes to one and all other numbers to zero
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]

#Essentially repeating the above for the test data set
###Making the mtrain matrix###
if (!exists("mtrain2")) {
  mtrain2 <- read.csv("mnist_test.csv", header=F) %>% as.matrix
  train_classification2 <- mtrain2[,1] #Go through this vector, set equal to zero if not 3 and 1 if 3.
  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]
```

> head(y)

[1] 0 0 0 0 0 0

Levels: 0 1

> head(y2)

[1] 0 0 0 0 0 0

Levels: 0 1

Caret

Running caret through train with decay=0

```
library(caret)
```

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

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

```
library(dplyr)

# fit the data to a neural net, nnet model in caret
# the nnet model has the following parameters:  size, decay
#tuning_df <- data.frame(size=8:12, decay=c(0, .1, .5, 1, 2)) #Size and decay have to be the same size
tuning_df <- data.frame(size=7:12, decay=0)

fitControl <- trainControl(method="none")

fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
  repeats = 2)

t_out <- caret::train(x=x, y=y, method="nnet",
                      trControl = fitControl,
```

```
                      tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)

true_y <- y
pred_y <- predict(t_out, x)

n_samples <- nrow(x)
error <- sum(true_y != pred_y)/n_samples

pred_error <- error
cat("train prediction error", pred_error, "\n")
```

Output not included because it's so long, but optimal number of nodes is 11

Running caret through train with variable decay

```
# fit the data to a neural net, nnet model in caret
# the nnet model has the following parameters:  size, decay
tuning_df <- data.frame(size=8:12, decay=c(0, .1, .5, 1, 2)) #Size and decay have to be the same size

fitControl <- trainControl(method="none")

fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
  repeats = 2)

t_out <- caret::train(x=x, y=y, method="nnet",
                      trControl = fitControl,
                      tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)

true_y <- y
pred_y <- predict(t_out, x)

n_samples <- nrow(x)
error <- sum(true_y != pred_y)/n_samples

pred_error <- error
cat("train1 prediction error", pred_error, "\n")
```

Output excluded for length. Optimal model has size of 9 and decay of .1

Now running caret with the optimal levels only and testing against the test data

```
# fit the data to a neural net, nnet model in caret
# the nnet model has the following parameters:  size, decay
tuning_df <- data.frame(size=9, decay=.1) #Size and decay have to be the same size
#tuning_df <- data.frame(size=8:12, decay=0)

fitControl <- trainControl(method="none")

fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
```

```
  repeats = 2)

t_out <- caret::train(x=x, y=y, method="nnet",
                      trControl = fitControl,
                      tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)

true_y <- y
pred_y <- predict(t_out, x)

n_samples <- nrow(x)
error <- sum(true_y != pred_y)/n_samples

pred_error <- error
cat("train1 prediction error", pred_error, "\n")

#Now recreating that to compare to the test data
true_y2 <- y2
pred_y2 <- predict(t_out, x2)

n_samples2 <- nrow(x2)
error2 <- sum(true_y2 != pred_y2)/n_samples2

pred_error2 <- error2
cat("test prediction error", pred_error2, "\n")
```

Output excluded for length. train1 prediction error 0

test prediction error 0.164