Homework 13

Ziyun Wang

12/10/2018

```
library(dplyr)
```

```
##
## 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: lattice
```

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

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2018g.
## 1.0/zoneinfo/America/New_York'
```

```
library(nnet)
```

Process the minst_train data

```
if (!exists("mtrain1")) {
  mtrain1 <- read.csv("mnist_train.csv", header=F) %>% as.matrix
  train_classification1 <- mtrain1[,1]
  # x matrix
  mtrain1 <- mtrain1[,-1]/256
  x1 <- mtrain1[1:1000,]

  colnames(mtrain1) <- NULL
  rownames(mtrain1) <- NULL
}

y1 <- rep(NA, length(train_classification1))
for (i in 1:length(train_classification1)) {
  number <- train_classification1[i]
  if (number==3) {
    number <- 1
  } else {
    number <- 0
  }
  y1[i] <- number
}
```

```
y1 <- as.factor(y1)[1:1000]
```

Process the minst_test data

```
if (!exists("mtrain2")) {
  mtrain2 <- read.csv("mnist_test.csv", header=F) %>% as.matrix
  train_classification2 <- mtrain2[,1]
  # x matrix
  mtrain2 <- mtrain2[,-1]/256
  x2 <- mtrain2[1:1000,]

  colnames(mtrain2) <- NULL
  rownames(mtrain2) <- NULL
}

y2 <- rep(NA, length(train_classification2))
for (i in 1:length(train_classification2)) {
  number <- train_classification2[i]
  if (number==3) {
    number <- 1
  } else {
    number <- 0
  }
  y2[i] <- number
}

y2 <- as.factor(y2)[1:1000]
```

Fit Neural Net

Decay = 0, Change only the number of the nodes

```
m <- rep(NA,7)
size  <- c(1,2,4,5,6,7,8)

for (i in 1:length(size)) {
  tuning_df <- data.frame(size=size[i], decay=0)

  fitControl <- trainControl(method="none")

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

  t_out <- caret::train(x=x1, y=y1, method="nnet",
                        trControl = fitControl,
                        tuneGrid=tuning_df, maxit=1000,
```

```
                        MaxNWts=10000)

  true_y1 <- y1
  predict_y1 <- predict(t_out,x1)

  n_samples1 <- nrow(x1)
  m[i] <- sum(true_y1 != predict_y1)/n_samples1
}
```

```
df_1 <- data.frame(num_nodes=size, train_prediction_error=m, stringsAsFactors = F)
df_1
```

```
##   num_nodes train_prediction_error
## 1         1                 0.017
## 2         2                 0.014
## 3         4                 0.002
## 4         5                 0.000
## 5         6                 0.000
## 6         7                 0.000
## 7         8                 0.000
```

Keeping the number of nodes as 7 and Chenge the decay

```
decay <- c(0,.1,.5,1)
n <- rep(NA,4)
for (i in 1: length(decay)) {
  tuning_df <- data.frame(size=7, decay=decay[i])

  fitControl <- trainControl(method="none")

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

  t_out <- caret::train(x=x1, y=y1, method="nnet",
                        trControl = fitControl,
                        tuneGrid=tuning_df, maxit=1000,
                        MaxNWts=10000)

  true_y1 <- y1
  predict_y1 <- predict(t_out,x1)

  n_samples1 <- nrow(x1)
  n[i] <- sum(true_y1 != predict_y1)/n_samples1
}
```

```
df_2 <- data.frame(num_nodes=7, decay=decay, train_prediction_error=n,
                   stringsAsFactors = F)
df_2
```

```
##   num_nodes decay train_prediction_error
## 1         7   0.0                 0.000
```

```
## 2         7   0.1                 0.000
## 3         7   0.5                 0.000
## 4         7   1.0                 0.002
```

Chosen Optimal model: size = 7, decay = 0.1

```
tuning_df <- data.frame(size=7, decay=0.1)

fitControl <- trainControl(method="none")

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

t_out <- caret::train(x=x1, y=y1, method="nnet",
                      trControl = fitControl,
                      tuneGrid=tuning_df, maxit=1000,
                      MaxNWts=100000)

true_y1 <- y1
predict_y1 <- predict(t_out,x1)

n_samples1 <- nrow(x1)
predict_error1 <- sum(true_y1 != predict_y1)/n_samples1
```

```
cat("train prediction error", predict_error1, "\n")
```

```
## train prediction error 0
```

Evaluate the accuracy of the neural net using minst_test data

```
true_y2 <- y2
predict_y2 <- predict(t_out,x2)

n_samples2 <- nrow(x2)
predict_error2 <- sum(true_y2 != predict_y2)/n_samples2
cat("train prediction error", predict_error2, "\n")
```

```
## train prediction error 0.047
```