HW_13

Jaquelin Martinez

December 10, 2018

```
library(dplyr, quietly = T)
```

```
##
## 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, quietly = T)
library(nnet, quietly = T)


# get the training datasets
if (!exists("mtrain")) {
  mtrain <- read.csv("mnist_train.csv", header=F) %>% as.matrix
  train_classification <- mtrain[,1] #y values
  output_vector <- rep(NA, length(train_classification))

  for (i in 1:length(train_classification)) {
    c_number <- train_classification[i]
    if (c_number==3) {
       output_vector[i] <- 1
    } else {
      output_vector[i] <- 0
    }
  }

  y <- factor(output_vector, levels=c(0,1))
  y <- y[1:1000]
  # for caret, y variable should be a factor
  # see line 54 in caret_intro_2d.R
  mtrain <- mtrain[,-1]/256 # x matrix

  colnames(mtrain) <- 1:(28^2)
  x <- mtrain[1:1000,]
```

```
}



tuning_df <- data.frame(size=7:12, decay=0)

fitControl <- trainControl(method="none")

fitControl <- trainControl(##10-fold CV
  method = "repeatedcv",
  number = 2,
  ## repeated ten times
  repeats = 2)

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

prediction_errors <- function(x, y)
{
  true_y <- y
  pred_y <- predict(t_out, x)

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

prediction_errors(x,y)

## [1] 0

print(t_out)

## Neural Network
##
## 1000 samples
##  784 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 2 times)
## Summary of sample sizes: 499, 501, 501, 499
## Resampling results across tuning parameters:
##
```

```
##   size  Accuracy   Kappa
##    7    0.9589918  0.7568800
##    8    0.9599858  0.7579217
##    9    0.9689999  0.8074180
##   10    0.9559998  0.7309645
##   11    0.9639999  0.7844205
##   12    0.9629979  0.7740867
##
## Tuning parameter 'decay' was held constant at a value of 0
## Accuracy was used to select the optimal model using the largest val
ue.
## The final values used for the model were size = 9 and decay = 0.
```

```
  # changing the size, and allowing variation for the decay

tuning_df <- data.frame(size=8:12, decay=c(0, 0.1, 0.5, 1, 2))

fitControl <- trainControl(method="none")

fitControl <- trainControl(##10-fold CV
  method = "repeatedcv",
  number = 2,
  ## repeated ten times
  repeats = 2)

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


prediction_errors(x,y)

## [1] 0.002



print(t_out)

## Neural Network
##
## 1000 samples
##  784 predictor
##    2 classes: '0', '1'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (2 fold, repeated 2 times)
## Summary of sample sizes: 500, 500, 499, 501
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy   Kappa
##    8    0.0    0.9680129  0.8054644
##    9    0.1    0.9710049  0.8186033
##   10    0.5    0.9715049  0.8195924
##   11    1.0    0.9730049  0.8256388
##   12    2.0    0.9725069  0.8166570
##
## Accuracy was used to select the optimal model using the largest val
ue.
## The final values used for the model were size = 11 and decay = 1.
```

```
  # now that I have an optimal number of nodes and decay level, I just
it using those - size =1 and decay =1

tuning_df <- data.frame(size=11, decay=1)

fitControl <- trainControl(method="none")

fitControl <- trainControl(##10-fold CV
  method = "repeatedcv",
  number = 2,
  ## repeated ten times
  repeats = 2)

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

prediction_errors(x,y)

## [1] 0.001
```

running the mnist_test dataset to test the accuracy of my neural net

```
if (!exists("mtrain2")) {
  mtrain2 <- read.csv("mnist_test.csv", header=F) %>% as.matrix
  train_classification2 <- mtrain2[,1] #y values
  output_vector2 <- rep(NA, length(train_classification2))
```

```r
  for (i in 1:length(train_classification2)) {
    c_number <- train_classification2[i]
    if (c_number==3) {
       output_vector2[i] <- 1
    } else {
      output_vector2[i] <- 0
    }
  }

  y2 <- factor(output_vector2, levels=c(0,1))
  y2 <- y2[1:1000]
  # for caret, y variable should be a factor
  # see line 54 in caret_intro_2d.R
  mtrain2 <- mtrain2[,-1]/256 # x matrix

  colnames(mtrain2) <- 1:(28^2)
  x2 <- mtrain2[1:1000,]

}


  # prediction of errors with neural net
prediction_errors(x2,y2)
```

```
## [1] 0.047
```