MNIST Prediction and Testing using CARET

Samuel Joslin

12/9/2018

The MNIST Database is a collection of handwritten numbers. The goal of this project is to train a neural network to accurately identify the handwritten number "3".

I will first read in a pre-paritioned MNIST dataset. For any given row, the first entry indicates the actual number and the next 784 entires represent pixel color intesnsity for a 28x28 image. The output below will be a matrix variable called "mtrain" which 1000x784 matrix entires represent pixel color intesnsity for a 28x28 image and "train_classification" which is a numerical vectors where the entries are either 1 or 0 depending on whether the given number is 3.

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

if (!exists("mtrain")) {
  mtrain <- read.csv("mnist_train.csv", header=F) %>% as.matrix
  for (i in 1:nrow(mtrain)){
    cn <- mtrain[i,1]
    if (cn == "3"){
      mtrain[i,1] <- 1
    } else {
      mtrain[i,1] <- 0
    }
  }
  train_classification <- mtrain[,1]
  mtrain <- mtrain[,-1]/256
  colnames(mtrain) <- 1:784
  rownames(mtrain) <- NULL
  mtrain <- mtrain[1:1000,]
}
print(dim(mtrain))
```

```
## [1] 1000  784
```

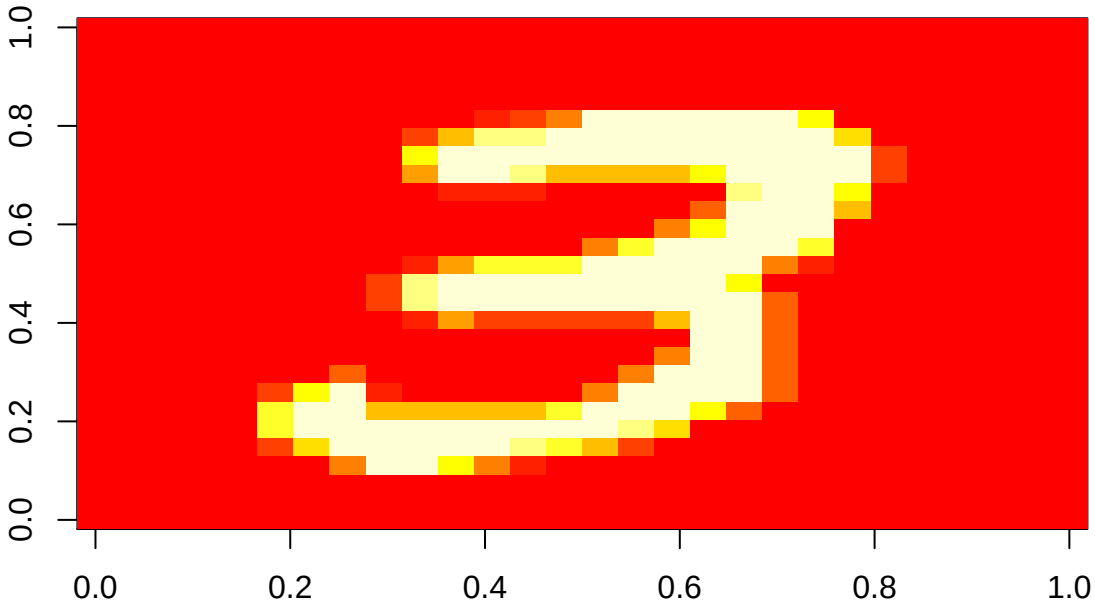
To properly understand the MNIST dataset, here is a function that plots the 28x28 image when a row is specified.

```
show_number <- function(m, i, oriented=T)
{
  im <- matrix(mtrain[i,], byrow=T, nrow=28)

  if (oriented) {
    im_orient <- matrix(0, nrow=28, ncol=28)
    for (i in 1:28)
      im_orient[i,] <- rev(im[,i])

    im <- im_orient
  }
  image(im)
}
```

```
show_number(mtrain, 8)
```



Training the Neural Network

I train a Neural Network that has decay equal to 0. I provide it with the training classification and the training data. I cross-validate by paritioning into 2 part, and repeating 3 times in order to optimze the amount of hidden nodes.

```
tuning_df <- data.frame(size=9, decay=0)

fitControl <- trainControl(
  method = "repeatedcv",
  number = 2,
  repeats = 3)

y <- factor(train_classification[1:1000])
x <- mtrain

t_out <- caret::train(x=x, y=y, method="nnet",
                      trControl = fitControl,
                      tuneGrid=tuning_df, maxit = 10000, MaxNWts = 10000, trace = F)
t_out %>% print
```

```
## Neural Network
##
## 1000 samples
##  784 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 3 times)
## Summary of sample sizes: 500, 500, 499, 501, 500, 500, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9623333  0.7635821
```

```
##
## Tuning parameter 'size' was held constant at a value of 9
## Tuning
##  parameter 'decay' was held constant at a value of 0
```

I create a Neural Network with a variable amount of decay. I cross-validate by paritioning into 2 part, and repeating 3 times in order to optimze the amount of hidden nodes and decay. I provide it with the training classification and the training data.

```
## Neural Network
##
## 1000 samples
##  784 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 3 times)
## Summary of sample sizes: 499, 501, 500, 500, 500, 500, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9716646  0.8150803
##
## Tuning parameter 'size' was held constant at a value of 5
## Tuning
##  parameter 'decay' was held constant at a value of 1
```

Testing on the test data set

By giving the neural network a section of the data it has not been exposed to I can test the neural networks accuracy. Below is a function that calcuates the percent error of the neural network based on unseen data and an associated classification.

```
prediction_errors <- function(classification, unseen_data, nnet)
{
  y <- factor(classification[1:1000])
  x <- unseen_data

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

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

Below creates a variable called "mtest" that I will use to find the percent error of the neural network's prediction.

```
if (!exists("mtest")) {
  mtest <- as.matrix(read.csv("mnist_test.csv", header=F))
  for (i in 1:6000){
    cn <- mtest[i,1]
    if (cn == "3"){
      mtest[i,1] <- 1
    } else {
      mtest[i,1] <- 0
```

```
    }
  }
  test_classification <- mtest[,1]
  mtest <- mtest[,-1]/256
  colnames(mtest) <- 1:784
  rownames(mtest) <- NULL
}
mtest <- mtest[1:1000,]

print(prediction_errors(test_classification,mtest,t_out))
```

```
## [1] 0.051
```

```
print(prediction_errors(test_classification,mtest,t_out2))
```

```
## [1] 0.044
```

Above shows the repective neural network prediction error when predicting on a new data set.