Digit Recognizer

James Whedbee

October 27, 2015

Problem Description

The goal in this Kaggle competition is to take an image of a handwritten single digit, and determine what that digit is.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

Analysis Approach

As an initial approach, I decided to compare test digit images to prototypical images of each digit which would be constructed from the training data. Whichever prototypical digit most closely matched the test digit image would be assigned as the "label" of the test digit image.

Initial Solution

The following functions were created to first build prototypical digits and then assign each test digit image a label.

A prototypical digit simply has the mean pixel value of all the training digit images with a given label. Test digit images were then compared to prototypical digits using sum of squared residuals.

```
meanDigits <- function(){

    digitTrain <- read.csv("~/Kaggle/Digits/train.csv")
    digitTest <- read.csv("~/Kaggle/Digits/test.csv")

    #Build prototypical digits from training data
    meanDigits <- matrix(nrow=0,ncol=785)
    for (digit in seq(0,9,1)){
        meanPixels <- sapply(digitTrain[digitTrain$label==digit,],mean)
        meanDigits <- rbind(meanDigits,meanPixels)
    }

    meanDigitFrame <- as.data.frame(meanDigits)
    row.names(meanDigitFrame) <- NULL

    #Label each test digit image by comparison with the prototypical digits
    ImageId <- seq(28000)
    Label <- apply(digitTest,1,digitDiff)
    prediction <- data.frame(ImageId,Label)
    prediction
```

```
}

digitDiff <- function(digitLine){
    # Finds the sum of the squared residuals of pixels for each prototypical digit
    #
    # Args:
    #   digitLine: Vector of pixel information for a given digit
    # Returns:
    #   "Average" digit with least sqaured residuals
    residuals <- apply(meanDigitFrame[,-1],1,function(x){sum((digitLine-x)^2)})
    which.min(residuals)-1
}
```

Initial Solution Analysis

This approach earned a score of .80614 on the test data. Certain digits are too similar and handwriting is too variable for this approach to beat more sophisticated methods.

Revised Solution and Analysis

As an alternative approach, I tried implementing a random forest. The baseline here was quite high, earning a score of .93514 on the test data.

Intuitively, the explanation for this is that there are certain areas on the image that if darkened immediately rule out the possibility of certain digits. We can see that a very small number of pixels provide most of the information necessary to distinguish digits, which is why the random forest is so effective.

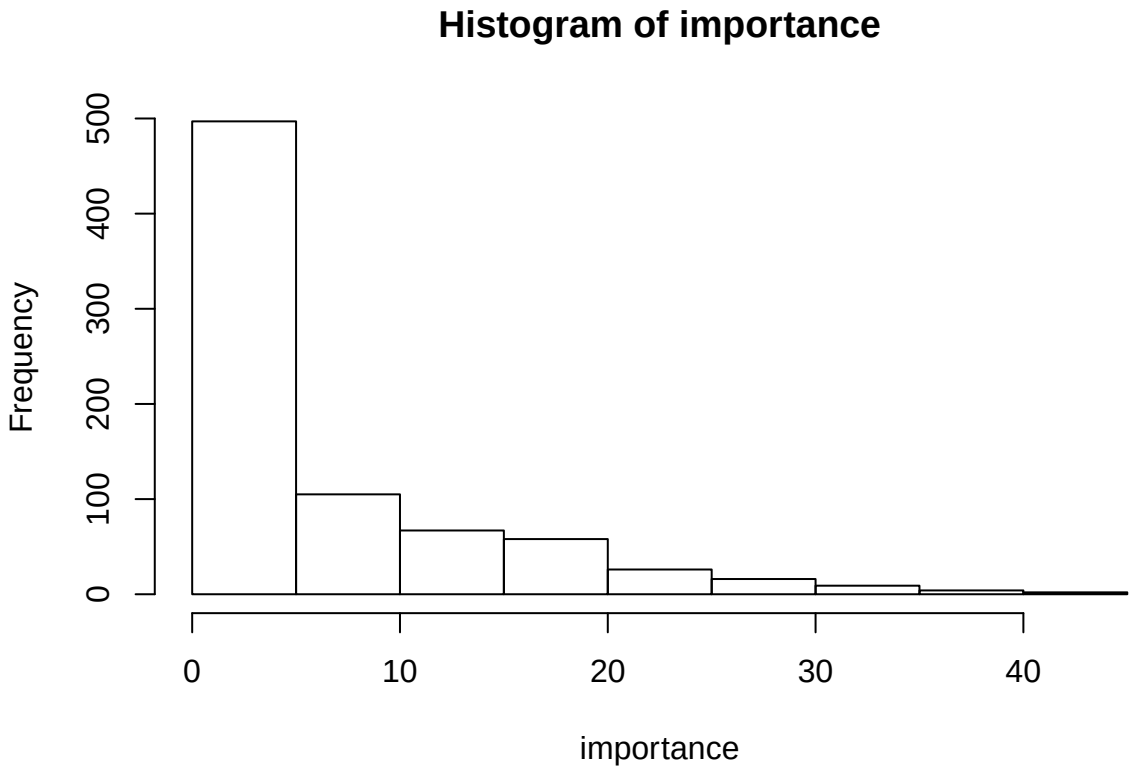
```
    digitTrain <- read.csv("~/Kaggle/Digits/train.csv")
    digitTest <- read.csv("~/Kaggle/Digits/test.csv")

    set.seed(0)
    numTrain <- 5000
    numTrees <- 50

    rows <- sample(1:nrow(digitTrain), numTrain)
    labels <- as.factor(digitTrain[rows,1])
    train <- digitTrain[rows,-1]

    digitForest <- randomForest(train, labels, xtest=digitTest, ntree=numTrees,keep.forest = TRUE)

    importance <- importance(digitForest)
    hist(importance)
```




```
    quantile(importance,probs = c(.25,.5,.75,.9,.95,1))
```

```
##       25%       50%       75%       90%       95%      100%
##  0.000000  1.195659  9.338471 17.389292 23.499293 40.879094
```