Homework 6

Kyle Walter

2/28/2021

# Working with Handwritten Numbers

##Background and Preprocessing The challenge at hand, can a predictive algorithm predict the value of a hand written number. Forty-two thousand images of hand digits between zero and nine that have been scanned into a computer. The value in the pixel ranges from 0 to 255 representing the gray scale transaction of image. This usually means that 0 represents white and 255 means the color of that pixel is black. There are 784 pixels per image. Can we utilize the information to predict which numbers are written in each image.

Due to capacity issues, we have selected to utilize a smaller sampling of the data using divide ten selection. In other words selecting every 10 records from the training and test set.

First let’s bring in the data

trainingData <- read.csv("R:/Graduate School/IST707 - Data Analysis/Week 7/Kaggle-digit-train-sample-small-1400.csv")  
testData <- read.csv("R:/Graduate School/IST707 - Data Analysis/Week 7/Kaggle-digit-test-sample1000.csv")

Next we’ll transform the labels from integer into factors for the class predction

trainingData$label <- as.factor(trainingData$label)

Lastly we’ll set training controls on this data in order to do our Cross Validation. In this case we’ll set the number of folds to three. This means when the model is training it will break the data into thirds. It will use 2/3rd of the data to train the model and 1/3 of the data test the accuracy of the model. It will do this once for each 3rd of the data and tune the results of the model on this outcome. This will help us achieve a more optimum model on a single run though the computational time could be significantly longer.

require(caret)

## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2

Controls <- trainControl(method="cv", number = 3)

##Decsion Tree Model

A decsion tree is a model which outputs a flow chart type structure making it easier to understand why a particular record was choosen. For this case we’ll use the weka implementation of the C4.5 algorithm called J48. In order to make sure that result in the same the outcome every time where random groups are being chose, we’ll set the input to randome number generation. Since tehnical systems use formulas to generate random numbers, but setting the seed we’re working with the same random list every time and thus can reproduce the results.

There are two parameters we can adjust to see how the model will function, 1 the confidence value, mean how aggressively do we want the algorithm to simplify the tree. 2 the M value or the number of instances per leaf.

For our training purpose we’ll utilize the Caret Package from R for the J48 algorithm. The advantage of this is it will not only allow us to test the folds that we announced earlier, but we’ll also see results for the confidence interval and the number of instances per leaf.

set.seed(223)  
Tree <- train(label~., data = trainingData, method="J48", trControl=Controls)  
Tree

## C4.5-like Trees   
##   
## 1400 samples  
## 784 predictor  
## 10 classes: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 932, 934, 934   
## Resampling results across tuning parameters:  
##   
## C M Accuracy Kappa   
## 0.010 1 0.7107039 0.6783272  
## 0.010 2 0.7242673 0.6933982  
## 0.010 3 0.7135621 0.6814582  
## 0.255 1 0.7021172 0.6687735  
## 0.255 2 0.7228336 0.6918796  
## 0.255 3 0.7056846 0.6727396  
## 0.500 1 0.7006866 0.6671764  
## 0.500 2 0.7235458 0.6926715  
## 0.500 3 0.7078213 0.6750908  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were C = 0.01 and M = 2.

So now as we can see from the results of the model, a confidence of .01 and M value (minimum number of instances per leaf), the optimum decision tress found with an accuracy of 72.4% and the model shows us a Kappa recall of 69% over the iterations. Mean this model is about average.

## Bayesian Logistic Regression Model

The other algorithm that we’ll test the data on is the Naive Bayes which predicts the probability that a classifier is A or B. We’ll set the seed again the fold process works for optimization and run the model

x = trainingData[,-1]  
y = trainingData[,1]  
set.seed(223)  
NbModel <- train(x, y, method="nb", trControl = Controls)

Now that the model has been created, let’s take a look at it’s outputand see how this model is preforming compared to the decision tree.

NbModel

## Naive Bayes   
##   
## 1400 samples  
## 784 predictor  
## 10 classes: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 932, 934, 934   
## Resampling results across tuning parameters:  
##   
## usekernel Accuracy Kappa   
## FALSE NaN NaN  
## TRUE 0.1792885 0.06979496  
##   
## Tuning parameter 'fL' was held constant at a value of 0  
## Tuning  
## parameter 'adjust' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were fL = 0, usekernel = TRUE and adjust  
## = 1.

Overall this model on the data is significantly less impactful than that of the decision tree. It shows an accuracy of 17.9% meaning it only correctly is able to predict the value of the hand written number 18% of time when tested and optimized over three runs. Additionally by holding out different 3rd of the date the kappa reflects only 6.9%. The likelihood it would recognize the similar value in test set is almost near zero.

## Model Choice

Given the outputs of the above two models, it makes more sense in this case to select the decision tree as it has both better classification accuracy of the 10 possibly options and and also provides an overall process recall that is respectible for kappa. This may be driven by the smaller date set, but we could apply to the process on a system with more computational power and retry with the larger data set. It could potentially show a better performance for the naive bays model. Though keep in mind this classifier is better suited for yes or no prediction than those of multiple classes. As such it’s performance in this case is greatly reduced.

As for the predictions here we go

ChosenModel <- predict(Tree, testData)  
ChosenModel

## [1] 2 0 9 0 3 3 0 3 0 5 5 7 3 6 4 5 5 1 9 0 7 1 1 1 7 4 2 9 9 7 7 5 4 2 6 2 9  
## [38] 2 8 6 7 7 4 9 8 7 8 3 0 7 6 8 1 5 5 2 1 2 4 6 7 1 7 3 0 0 2 9 0 1 6 5 3 8  
## [75] 2 2 2 9 2 6 5 9 1 0 9 2 4 5 9 7 2 0 6 6 1 4 3 8 7 4 0 3 2 0 5 3 6 5 6 3 6  
## [112] 0 6 2 1 7 7 1 3 1 3 7 2 1 6 3 8 7 3 4 4 5 6 0 0 0 3 1 3 0 5 5 4 5 5 8 9 7  
## [149] 2 1 4 3 5 6 5 8 7 5 7 1 3 0 4 3 1 2 7 6 3 0 3 7 4 6 1 3 7 4 1 2 4 1 8 2 4  
## [186] 9 2 1 8 0 6 1 4 9 6 0 9 7 0 9 1 8 0 5 9 0 8 4 6 2 0 9 3 6 7 2 1 6 1 4 2 5  
## [223] 3 0 2 0 9 6 1 0 0 6 9 1 7 8 3 1 8 6 8 6 2 2 5 5 9 3 3 3 9 5 1 9 8 4 5 1 4  
## [260] 5 2 3 3 5 7 0 6 8 5 1 6 0 6 3 3 9 1 9 8 4 0 6 2 6 3 3 7 8 9 9 2 9 7 9 9 6  
## [297] 3 2 3 3 6 9 8 2 8 2 7 1 4 5 8 0 8 0 0 3 2 4 1 5 4 1 1 4 8 6 3 1 4 3 8 3 1  
## [334] 8 0 8 0 0 6 0 6 7 1 8 8 0 4 0 2 2 4 7 7 2 9 3 8 9 0 9 3 6 2 8 3 2 3 8 1 2  
## [371] 5 8 9 9 7 9 9 5 2 7 6 5 5 9 2 6 2 1 5 4 1 7 9 9 6 7 1 1 7 3 9 7 6 1 1 1 9  
## [408] 1 8 6 5 0 4 1 2 9 1 2 3 5 9 6 6 5 1 1 4 7 4 7 4 8 7 2 0 1 8 9 5 0 8 4 7 4  
## [445] 8 4 7 5 3 9 9 7 6 9 3 9 2 3 9 0 1 7 5 4 5 1 7 4 6 1 1 4 9 5 6 0 1 5 1 0 4  
## [482] 8 1 5 9 9 4 8 3 7 0 4 2 4 2 7 2 8 1 0 6 6 9 7 1 8 3 5 0 0 7 3 5 7 8 3 8 7  
## [519] 0 2 6 7 2 7 7 1 6 1 1 7 0 0 3 8 3 5 6 5 5 3 0 1 8 1 0 3 0 9 2 1 1 3 0 8 0  
## [556] 0 7 5 6 2 8 3 1 1 6 4 4 1 1 9 6 8 8 6 0 9 1 3 4 3 5 1 7 6 7 1 7 6 8 8 8 3  
## [593] 3 5 2 6 4 6 0 2 1 1 7 1 8 2 8 2 6 8 7 6 2 2 3 4 7 0 0 3 8 5 4 4 0 3 8 3 6  
## [630] 7 6 1 8 4 1 4 7 7 8 3 0 5 9 8 5 7 5 4 1 8 4 9 0 0 8 2 4 3 3 7 2 3 4 0 4 5  
## [667] 1 3 9 6 3 9 4 3 8 7 7 0 2 0 6 7 8 2 1 9 9 6 4 9 2 6 0 1 8 4 3 9 8 8 4 0 3  
## [704] 0 6 0 6 4 6 4 5 8 1 5 8 6 2 3 7 3 9 5 1 6 1 6 6 4 7 0 7 1 3 2 7 7 1 3 1 5  
## [741] 9 4 3 4 3 9 0 7 2 6 9 9 4 4 7 4 7 1 1 0 2 0 4 0 4 0 0 3 1 9 2 5 0 1 6 5 4  
## [778] 6 3 6 1 6 5 8 6 5 5 9 2 6 3 5 4 0 1 3 1 6 9 7 1 1 3 3 1 4 9 6 4 1 4 6 2 8  
## [815] 4 5 0 9 7 4 3 0 5 0 1 9 0 4 0 2 8 2 5 9 3 9 6 1 3 6 1 9 0 8 8 6 7 2 0 5 8  
## [852] 9 0 7 1 1 1 1 4 6 0 9 1 4 2 3 6 9 2 3 9 5 4 2 2 2 8 1 2 1 4 5 1 3 2 7 1 5  
## [889] 4 9 1 7 6 0 4 6 9 4 7 2 8 2 9 7 9 7 5 5 3 7 3 0 4 7 1 6 4 6 1 0 7 3 0 9 4  
## [926] 7 9 2 6 2 3 3 1 4 8 3 7 7 9 5 6 1 6 1 0 9 3 2 9 8 6 5 5 2 3 2 8 3 2 6 7 9  
## [963] 4 0 7 5 1 8 8 5 3 1 9 6 7 0 8 0 7 3 5 8 7 5 7 7 0 3 6 2 1 9 0 6 8 3 6 2 1  
## Levels: 0 1 2 3 4 5 6 7 8 9