NB\_SVM\_KNN\_RF - Digit Recognizer Competition

Jose Reyes

8.26.2020

#### Introduction

The Digit Recognizer Competition challenges participants to correctly identify numbers from a dataset containing thousands of handwritten images. The data includes each of these image’s grey scale values of each pixel which can be used to train several learning techniques.

In this report, five models will attempt to overcome this challenge including Naïve Bayes, Decision Tree, K Nearest Neighbor (KNN) algorithm, Support Vector Machine (SVM), and Random Forrest. Each model has its own different approach will be discussed and compared to each other by performing k-fold cross validation.

When it comes to data mining, there is often not a silver bullet when choosing a model for analysis. Understanding a variety of learning techniques’ pros and cons will allow data scientists to conduct the appropriate analysis for the right data.

#### Analysis and Models

#### About the Data

The dataset contains 42,000 images and their gray scale values for 785 different pixels. Each image is a single handwritten digit between 0 and 9 and is labeled as such.

Both the number of images and pixels used for this analysis are reduced due to time constraints and to remove noise, redundancy, and decrease dimensionality.

To do this, Principal Component Analysis (PCA) was used to select features based on variance.

### Load Train Data

data <- read.csv("Kaggle-digit-train.csv", header = TRUE, stringsAsFactors = TRUE)  
data$label <- as.factor(data$label)

### Use Principal Component Analysis to select features based on variance

pca\_data <- PCA(t(select(data,-label)), ncp = 10, graph=FALSE)

### Reduce number of dimensions to 10

data = data.frame(data$label, pca\_data$var$coord)

### Reduce number of samples used to 25% of original number of rows

set.seed(275)  
Reduce\_Data\_25 <- sample(nrow(data), nrow(data)\*.25)  
Reduced\_Data <- data[Reduce\_Data\_25,]  
dim(Reduced\_Data)

### Build Decision Naïve Bayes Model k-fold cross validation sets

The first model is Naïve Bayes, which is a set of supervised learning algorithms based on applying Bayes theorem with the “naïve” assumption that each pair of features being classified is independent of each other. The model was built with a 10-fold cross validation to check for accuracy.

### 10-fold CV  
N = nrow(Reduced\_Data)  
holdout <- split(sample(1:N), 1:10)  
  
AllResults <- list()  
AllLabels <- list()  
for (k in 1:10){  
Test <- Reduced\_Data[holdout[[k]],]   
Train=Reduced\_Data[-holdout[[k]],]  
  
### Remove Labels  
Test\_noLabel <- Test[-c(1)]  
Test\_Label <- Test$data.label  
  
### Build Naive Bayes Model  
NB <- naiveBayes(data.label~., data = Reduced\_Data, na.action = na.pass)  
  
### Use Naive Bayes to predict labels in test data  
NB\_pred <- predict(NB, Test\_noLabel)  
  
### Test accuracy of Naive Bayes model with train data sub set  
confusionMatrix(NB\_pred, Test$data.label)  
  
### Store Results   
AllResults <- c(AllResults, NB\_pred)  
AllLabels <- c(AllLabels, Test\_Label) }

### Confusion Matrix Results

Naive Bayes Confusion Matrix  
## 1 2 3 4 5 6 7 8 9 10  
## 1 854 0 24 9 13 46 22 11 5 26  
## 2 0 1178 31 11 14 14 18 33 31 11  
## 3 2 16 734 24 13 23 34 14 21 5  
## 4 7 18 26 828 1 134 6 0 100 11  
## 5 4 2 41 10 742 29 13 44 8 233  
## 6 123 1 13 52 7 585 39 10 40 11  
## 7 22 1 46 7 27 14 930 1 9 3  
## 8 1 2 19 10 4 5 1 903 4 54  
## 9 5 4 44 147 3 31 0 14 744 28  
## 10 0 1 3 23 236 38 0 61 53 627

Overall accuracy: 0.773809523809524

Incorrect classification: 0.226190476190476

An overall accuracy of 77% is a good start but can be improved upon by exploring other modelling techniques.

### Build Decision Tree Model k-fold cross validation sets

The second model is a decision tree, which splits the source set into subsets based on a certain attribute value test. This splitting of data occurs on a number of nodes, and the process is repeated until splitting no longer adds value to the prediction. In this case, the decision tree split the set of images based on their gray scale values for each pixel. Like Naïve Bayes, this model was also built with a 10-fold cross validation to check for accuracy.

### 10-fold CV  
N = nrow(Reduced\_Data)  
holdout <- split(sample(1:N), 1:10)  
  
AllResults <- list()  
AllLabels <- list()  
for (k in 1:10){  
  
Test <- Reduced\_Data[holdout[[k]],]   
Train=Reduced\_Data[-holdout[[k]],]  
  
### Remove Labels  
Test\_noLabel <- Test[-c(1)]  
Test\_Label <- Test$data.label  
  
### Build Decision Tree Model   
Decision\_Tree <- ctree(data.label~., data = Reduced\_Data)  
  
### Use Decision Tree to predict labels in test data  
Decision\_Tree\_pred <- predict(Decision\_Tree, Test\_noLabel)  
  
### Test accuracy of Decision Tree model with train data sub set  
confusionMatrix(Decision\_Tree\_pred, Test\_Label)  
  
### Store Results for each cold  
AllResults <- c(AllResults, Decision\_Tree\_pred)  
AllLabels <- c(AllLabels, Test\_Label) }

### 

### Confusion Matrix

Decision Tree Confusion Matrix  
## 1 2 3 4 5 6 7 8 9 10  
## 1 924 0 16 7 6 43 13 2 6 12  
## 2 0 1186 6 4 3 3 4 1 13 0  
## 3 19 5 841 24 25 12 27 26 27 8  
## 4 8 11 20 916 2 40 7 5 72 14  
## 5 2 1 7 10 831 10 10 23 10 137  
## 6 37 3 12 48 9 743 21 7 41 16  
## 7 7 2 9 9 24 20 971 0 10 9  
## 8 0 8 12 8 17 10 1 955 13 55  
## 9 5 6 52 83 12 22 7 7 804 23  
## 10 16 1 6 12 131 16 2 65 19 735

Overall accuracy: 0.848190476190476

Incorrect classification: 0.151809523809524

The accuracy of 85% is an improvement when compared to Naïve Bayes.

### Build KNN Model k-fold cross validation sets

The K nearest neighbors is an algorithm that stores all available cases and classifies new cases based on a similarity measure. The value “k” can be customized to determine how many closest neighbors will be used to determine a category. Due to computing time constraints, 2-fold cross validations will be used for this model as well as SVM and Random Forest.

### 2-fold CV  
AllResults <- data.frame(orig=c(), pred=c())  
for (k in 1:2){  
   
Test <- data[holdout[[k]],]   
Train <- data[-holdout[[k]],]  
  
### Remove Labels  
Test\_noLabel <- Test[-c(1)]  
Test\_Label <- Test$label  
  
### Use Decision Tree to predict labels in test data  
KNN\_pred <- knn(train = Train, test = Test, cl = Train$label, k=3, prob=FALSE)  
  
### Store Results for each   
AllResults <- rbind(AllResults, data.frame(orig=Test$label, pred=KNN\_pred)) }

### Confusion Matrix

KNN Confusion Matrix  
## 0 1 2 3 4 5 6 7 8 9  
## 0 598 0 0 0 0 2 3 0 1 1  
## 1 0 751 1 1 1 0 0 3 0 0  
## 2 12 18 491 8 2 3 3 13 7 3  
## 3 2 10 8 616 0 20 3 6 8 4  
## 4 1 12 2 0 560 0 8 2 0 35  
## 5 7 6 0 18 5 489 16 0 6 10  
## 6 8 2 0 0 1 7 602 1 2 0  
## 7 1 16 3 0 4 1 0 634 0 16  
## 8 6 15 4 27 5 16 4 5 516 15  
## 9 6 4 1 7 21 1 1 28 1 543

Overall accuracy: 0.920634920634921

Incorrect classification: 0.0793650793650794

The model was run multiple times for different k values to produce the highest accuracy. K=3 provided the highest overall accuracy of 92%

### Build SVM Model k-fold cross validation sets

Support Vector Machine (SVM) can be used for both regression and classification by creating a hyperplane that distinctly classifies the given data points. Hyperplanes are decision boundaries that help classify the data points and the dimension can vary based on the number of features.

Different SVM algorithms use different types of kernel functions such as radial and sigmoid. The cost (C) of misclassification on the training data an also be customized. SVM can handle both binarized and continuous data and both types were used and compared for accuracy.

### Binarize Data

cols\_to\_remove = c()  
for (col in colnames(Binarized\_Data)) {  
 if (col != "label"){  
 if (length(unique(Binarized\_Data[,c(col)])) ==1){  
 cols\_to\_remove <- c(cols\_to\_remove, col)  
 }  
 }  
}  
  
Binarized\_Data\_trainset <- Binarized\_Data[-which(colnames(Binarized\_Data) %in% cols\_to\_remove)]  
  
### 2-fold CV  
AllResults <- data.frame(orig=c(), pred=c())  
for (k in 1:2){  
   
Test <- Binarized\_Data\_trainset[holdout[[k]],]   
Train <- Binarized\_Data\_trainset[-holdout[[k]],]  
  
### Remove Labels  
Test\_noLabel <- Test[-c(1)]  
Test\_Label <- Test$label  
  
SVM <- svm(label~., Train, kernel ="radial", cost=100, na.action=na.pass)  
SVM\_pred <- predict(SVM, Test\_noLabel, type=c("class"))  
  
### Store Results   
AllResults <- rbind(AllResults, data.frame(orig=Test$label, pred=SVM\_pred))}

### Confusion Matrix

SVM Confusion Matrix  
## 0 1 2 3 4 5 6 7 8 9  
## 0 578 0 3 0 2 7 9 0 5 1  
## 1 0 748 2 1 2 1 0 1 2 0  
## 2 6 1 485 12 13 2 17 11 12 1  
## 3 1 4 9 609 0 31 1 7 11 4  
## 4 3 0 2 0 576 1 3 2 0 33  
## 5 3 4 5 27 6 489 10 1 8 4  
## 6 7 2 6 1 8 12 584 0 3 0  
## 7 1 4 8 3 7 1 0 618 1 32  
## 8 4 11 13 31 3 17 5 4 515 10  
## 9 8 3 2 7 28 4 0 28 3 530

## Overall accuracy: 0.90984126984127

## Incorrect classification: 0.0901587301587301

The model was run multiple times to test for different kernels and cost values. A high cost value with a radial kernel provided the highest overall accuracy of 91%. The binarized data also produced drastically higher accuracy than continuous data.

### Build Random Forest Model k-fold cross validation sets

The final model is Random Forest, which is an ensemble learning method that creates a multitude of decision trees and produces the mode or mean prediction.

### 2-fold CV  
AllResults <- data.frame(orig=c(), pred=c())  
for (k in 1:2){  
   
Test <- data[holdout[[k]],]   
Train <- data[-holdout[[k]],]  
  
### Remove Labels  
Test\_noLabel <- Test[-c(1)]  
Test\_Label <- Test$label  
  
RF <- randomForest(label~., Train, na.action=na.pass)  
RF\_pred <- predict(RF,Test\_noLabel, type=c("class"))  
  
  
### Store Results   
AllResults <- rbind(AllResults, data.frame(orig=Test$label, pred=RF\_pred)) }

### Confusion Matrix

RF Confusion Matrix  
## 0 1 2 3 4 5 6 7 8 9  
## 0 591 0 1 0 0 1 6 0 6 0  
## 1 0 750 0 2 2 1 0 2 0 0  
## 2 4 4 514 6 7 0 6 7 11 1  
## 3 4 3 6 615 2 19 2 7 13 6  
## 4 3 0 2 0 584 0 7 1 4 19  
## 5 6 6 2 22 3 503 4 1 6 4  
## 6 10 3 3 0 3 10 592 0 2 0  
## 7 3 3 11 1 9 1 0 632 2 13  
## 8 4 7 3 16 2 8 7 1 550 15  
## 9 7 2 1 10 13 4 2 19 4 551

## Overall accuracy: 0.933650793650794

## Incorrect classification: 0.0663492063492064

#### Results

|  |  |  |
| --- | --- | --- |
| **Model** | **Overall Accuracy** | **Incorrect classification** |
| Naïve Bayes | 77.38% | 22.62% |
| Decision Tree | 84.82% | 15.18% |
| kNN | 92.06% | 7.94% |
| SVM | 90.98% | 9.02% |
| Random Forest | 93.37% | 6.63% |

Across all 5 models, the accuracy ranged from 77.38% to 93.37. It appears that as the learning models became more complex (such as a simple decision tree vs. an ensemble of decision tree), the resulting accuracies were higher.

There are clear advantages to testing several types of learning and modelling techniques. There is almost usually never one model in data mining that produces perfect predicting accuracy and exploring and understanding the strengths and weaknesses of several learning techniques will allow analyst to better achieve their goals. In this analysis, Random Forest provided the highest accuracy with KNN following closely behind.

#### Conclusions

To identify numbers from a dataset of handwritten images, several learning techniques including Naïve Bayes, Decision Tree, K Nearest Neighbor (KNN) algorithm, Support Vector Machine (SVM), and Random Forrest were used. Each model was trained with a dataset that includes thousands of images and their grey scale values of each pixel.

Each model was discussed and compared to each other by performing either 10 or 2-fold cross validation. Random Forest produced the highest accuracy of 93.37% while Naïve Bayes produced the lowest with 77.38%. Overall, the use of several learning techniques demonstrated the importance of testing and comparing different methods to produce the best results.