Neural Net Project

Guan-Yuan Wang

2020/9/4

Neural Net Project

Let's wrap up this course by taking a a quick look at the effectiveness of Neural Nets!

We'll use the Bank Authentication Data Set from the UCI repository.

The data consists of 5 columns:

- variance of Wavelet Transformed image (continuous)
- skewness of Wavelet Transformed image (continuous)
- curtosis of Wavelet Transformed image (continuous)
- entropy of image (continuous)
- class (integer) Where class indicates whether or not a Bank Note was authentic.

Get the Data

Use `read.csv` to read the bank_note_data.csv file.

```
data <- read.csv("bank_note_data.csv")
```

Check the head of the data frame and its structure.

```
head(data)
```

```
##   Image.Var Image.Skew Image.Curt  Entropy Class
## 1   3.62160     8.6661    -2.8073 -0.44699     0
## 2   4.54590     8.1674    -2.4586 -1.46210     0
## 3   3.86600    -2.6383     1.9242  0.10645     0
## 4   3.45660     9.5228    -4.0112 -3.59440     0
## 5   0.32924    -4.4552     4.5718 -0.98880     0
## 6   4.36840     9.6718    -3.9606 -3.16250     0
```

```
str(data)
```

```
## 'data.frame':    1372 obs. of  5 variables:
##  $ Image.Var : num  3.622 4.546 3.866 3.457 0.329 ...
##  $ Image.Skew: num  8.67 8.17 -2.64 9.52 -4.46 ...
##  $ Image.Curt: num  -2.81 -2.46 1.92 -4.01 4.57 ...
##  $ Entropy   : num  -0.447 -1.462 0.106 -3.594 -0.989 ...
##  $ Class     : int  0 0 0 0 0 0 0 0 0 0 ...
```

EDA

Create whatever visualizations you are interested in. We'll skip this step for the solutions notebook/video because the data isn't easily interpretable since its just statistical info on images.

Train Test Split

Use the `caTools` library to split the data into training and testing sets.

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.6.3
```

```
split <- sample.split(data$Class, .7)
train <- subset(data, split == T)
test <- subset(data, split == F)
```

Check the structure of the train data and note that Class is still an int data type. We won't convert it to a factor for now because the neural net requires all numeric information.

```
str(train)
```

```
## 'data.frame':    960 obs. of  5 variables:
##  $ Image.Var : num  4.546 3.866 0.329 3.591 3.203 ...
##  $ Image.Skew: num  8.17 -2.64 -4.46 3.01 5.76 ...
##  $ Image.Curt: num  -2.459 1.924 4.572 0.729 -0.753 ...
##  $ Entropy   : num  -1.462 0.106 -0.989 0.564 -0.613 ...
##  $ Class     : int  0 0 0 0 0 0 0 0 0 0 ...
```

Building the Neural Net

Call the `neuralnet` library

```
library(neuralnet)
```

Browse through the documentation of neuralnet

```
#help("neuralnet")
```

Use the neuralnet function to train a neural net, set `linear.output=FALSE` and choose 10 hidden neurons(`hidden=10`)

```
names <- names(train)
func <- as.formula(
  paste("Class ~", paste(names[!names %in% "Class"], collapse = " + "))
)

nn <- neuralnet(func, data = train, hidden = 10, linear.output = F)
plot(nn)
```

Predictions

Use `compute()` to grab predictions useing your nn model on the test set. Reference the lecture on how to do this.

```
pred.nn.values <- compute(nn, test[1:4])
true.pred <- pred.nn.values$net.result
```

Check the head of the predicted values. You should notice that they are still probabilities.

```
head(true.pred)
```

```
##            [,1]
## 1  0.0005408811
## 4  0.0005482068
## 6  0.0005104464
## 8  0.0016207155
## 10 0.0006486182
## 11 0.0007570789
```

Apply the round function to the predicted values so you only 0s and 1s as your predicted classes.

```
true.pred <- round(true.pred)
head(true.pred)
```

```
##    [,1]
## 1     0
## 4     0
## 6     0
## 8     0
## 10    0
## 11    0
```

Use `table()` to create a confusion matrix of your predictions versus the real values

```
table(true.pred, test$Class)
```

```
##
## true.pred   0   1
##         0 229   0
##         1   0 183
```

You should have noticed that you did very well! Almost suspiciously well! Let's check our results against a randomForest model!

Comparing Models

Call the `randomForest` library

```r
library(randomForest)
```

Run the Code below to set the Class column of the data as a factor (randomForest needs it to be a factor, not an int like neural nets did. Then re-do the train/test split

```r
train$Class <- as.factor(train$Class)
test$Class <- as.factor(test$Class)
```

Create a randomForest model with the new adjusted training data.

```r
randomForestModel <- randomForest(Class ~ ., train)
```

Use `predict()` to get the predicted values from your rf model.

```r
rf.pred <- predict(randomForestModel, test)
```

Use `table()` to create the confusion matrix.

```r
table(rf.pred, test$Class)
```

```
##
## rf.pred   0   1
##       0 229   1
##       1   0 182
```

How did the models compare?