Sean Toon

IT-460-18500-M01 Machine Learning 2024 C-5

10/27/24

Module Seven Lab – Artificial Neural Networks

The dataset that I am going to be working with for this lab consists of 20000 examples of 26 English alphabet capital letters using 20 different randomly distorted and reshaped black-and-white fonts. First, I explored and prepared the data:

I started by importing the data with the following command:

```
> letters <- read.csv("C:\\Users\\toons\\Downloads\\letterdata.csv")
```

And then viewed the data with the following command:

```
> str(letters)
'data.frame':   20000 obs. of  17 variables:
 $ letter: chr  "T" "I" "D" "N" ...
 $ xbox  : int  2 5 4 7 2 4 4 1 2 11 ...
 $ ybox  : int  8 12 11 11 1 11 2 1 2 15 ...
 $ width : int  3 3 6 6 3 5 5 3 4 13 ...
 $ height: int  5 7 8 6 1 8 4 2 4 9 ...
 $ onpix : int  1 2 6 3 1 3 4 1 2 7 ...
 $ xbar  : int  8 10 10 5 8 8 8 8 10 13 ...
 $ ybar  : int  13 5 6 9 6 8 7 2 6 2 ...
 $ x2bar : int  0 5 2 4 6 6 6 2 2 6 ...
 $ y2bar : int  6 4 6 6 6 9 6 2 6 2 ...
 $ xybar : int  6 13 10 4 6 5 7 8 12 12 ...
 $ x2ybar: int  10 3 3 4 5 6 6 2 4 1 ...
 $ xy2bar: int  8 9 7 10 9 6 6 8 8 9 ...
 $ xedge : int  0 2 3 6 1 0 2 1 1 8 ...
 $ xedgey: int  8 8 7 10 7 8 8 6 6 1 ...
 $ yedge : int  0 4 3 2 5 9 7 2 1 1 ...
 $ yedgex: int  8 10 9 8 10 7 10 7 7 8 ...
```

Because every feature is an integer, I did not need to prepare the data any further. I then created training and testing sets with the following commands. The training set uses 16000 of the 20000 entries:

```
> letters train <- letters[1:16000, ]
```

And the testing set consists of the remaining data:

```
> letters test  <- letters[16001:20000, ]
```

Next, I installed the kernlab package to use the ksvm() function to specify the linear kernel with the following code:

```
> letter_classifier <- ksvm(letter ~ ., data = letters_train, kernel = "vanilladot")
 Setting default kernel parameters
> letter_classifier
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 7037

Objective Function Value : -14.1746 -20.0072 -23.5628 -6.2009 -7.5524 -32.7694 -49.9786
Training error : 0.130062
> 
```

Next, I evaluated the model performance to get a better idea about how well the model would perform in the real world. I used the predict() function and was able to get the first six predicted letters U, N, V, X, N, and H. I then compared the predicted letter to the true letter using the data set while using the table() function to get a better visualization:

```
> letter_predictions <- predict(letter_classifier, letters_test)
> head(letter_predictions)
[1] U N V X N H
Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
> table(letter_predictions, letters_test$letter)

letter_predictions   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W
                 A 144   0   0   0   0   0   0   0   0   1   0   0   1   2   2   0   5   0   1   1   1   0   1
                 B   0 121   0   5   2   0   1   2   0   0   1   0   1   0   0   2   2   3   5   0   0   2   0
                 C   0   0 120   0   4   0  10   2   2   0   1   3   0   0   2   0   0   0   0   0   0   0   0
                 D   2   2   0 156   0   1   3  10   4   3   4   3   0   5   5   3   1   4   0   0   0   0   0
                 E   0   0   5   0 127   3   1   1   0   0   3   4   0   0   0   0   2   0  10   0   0   0   0
                 F   0   0   0   0   0 138   2   2   6   0   0   0   0   0   0  16   0   0   3   0   0   1   0
                 G   1   1   2   1   9   2 123   2   0   0   1   2   1   0   1   2   8   2   4   3   0   0   0
                 H   0   0   0   1   0   1   0 102   0   2   3   2   3   4  20   0   2   3   0   3   0   2   0
                 I   0   1   0   0   0   1   0   0 141   8   0   0   0   0   0   1   0   0   3   0   0   0   0
                 J   0   1   0   0   0   1   0   2   5 128   0   0   0   0   1   1   3   0   2   0   0   0   0
                 K   1   1   9   0   0   0   2   5   0   0 118   0   0   2   0   1   0   7   0   1   3   0   0
                 L   0   0   0   0   2   0   1   1   0   0   0 133   0   0   0   0   1   0   5   0   0   0   0
                 M   0   0   1   1   0   0   1   1   0   0   0   0 135   4   0   0   0   0   0   0   3   0   8
                 N   0   0   0   0   0   1   0   1   0   0   0   0   0 145   0   0   0   3   0   0   1   0   2
                 O   1   0   2   1   0   0   1   2   0   1   0   0   0   1  99   3   3   0   0   0   3   0   0
                 P   0   0   0   1   0   2   1   0   0   0   0   0   0   0   2 130   0   0   0   0   0   0   0
                 Q   0   0   0   0   0   0   8   2   0   0   0   3   0   0   3   1 124   0   5   0   0   0   0
                 R   0   7   0   0   1   0   3   8   0   0  13   0   0   1   1   1   0 138   0   1   0   1   0
                 S   1   1   0   0   1   0   3   0   1   1   0   1   0   0   0   0  14   0 101   3   0   0   0
                 T   0   0   0   0   3   2   0   0   0   0   1   0   0   0   0   0   0   0   3 133   1   0   0
                 U   1   0   3   1   0   0   0   2   0   0   0   0   0   0   1   0   0   0   0   0 152   0   0
                 V   0   0   0   0   0   1   3   4   0   0   0   0   1   2   1   0   3   1   0   0   0 126   1
                 W   0   0   0   0   0   0   1   0   0   0   0   0   2   0   0   0   0   0   0   0   4   4 127
                 X   0   1   0   0   2   0   0   1   3   0   1   6   0   0   1   0   0   0   1   0   0   0   0
                 Y   3   0   0   0   0   0   0   1   0   0   0   0   0   0   0   7   0   0   0   3   0   0   0
                 Z   2   0   0   0   1   0   0   0   3   4   0   0   0   0   0   0   0   0  18   3   0   0   0

letter_predictions   X   Y   Z
                 A   0   0   1
                 B   1   0   0
                 C   0   0   0
                 D   3   3   1
                 E   2   0   3
                 F   1   2   0
                 G   1   0   0
                 H   0   1   0
```

By returning vector values of true or false (whether the value matches the test data set) I can see that the classifier correctly identified 84 percent of the set:

```
> agreement <- letter_predictions == letters_test$letter
> table(agreement)
agreement
FALSE  TRUE
  643  3357
>  prop.table(table(agreement))
agreement
  FALSE    TRUE
0.16075 0.83925
>
```

To attempt to improve the model performance, I first changed the SVM kernel function with the following code. I began with the Gaussian RBF kernel using the ksvm() function:

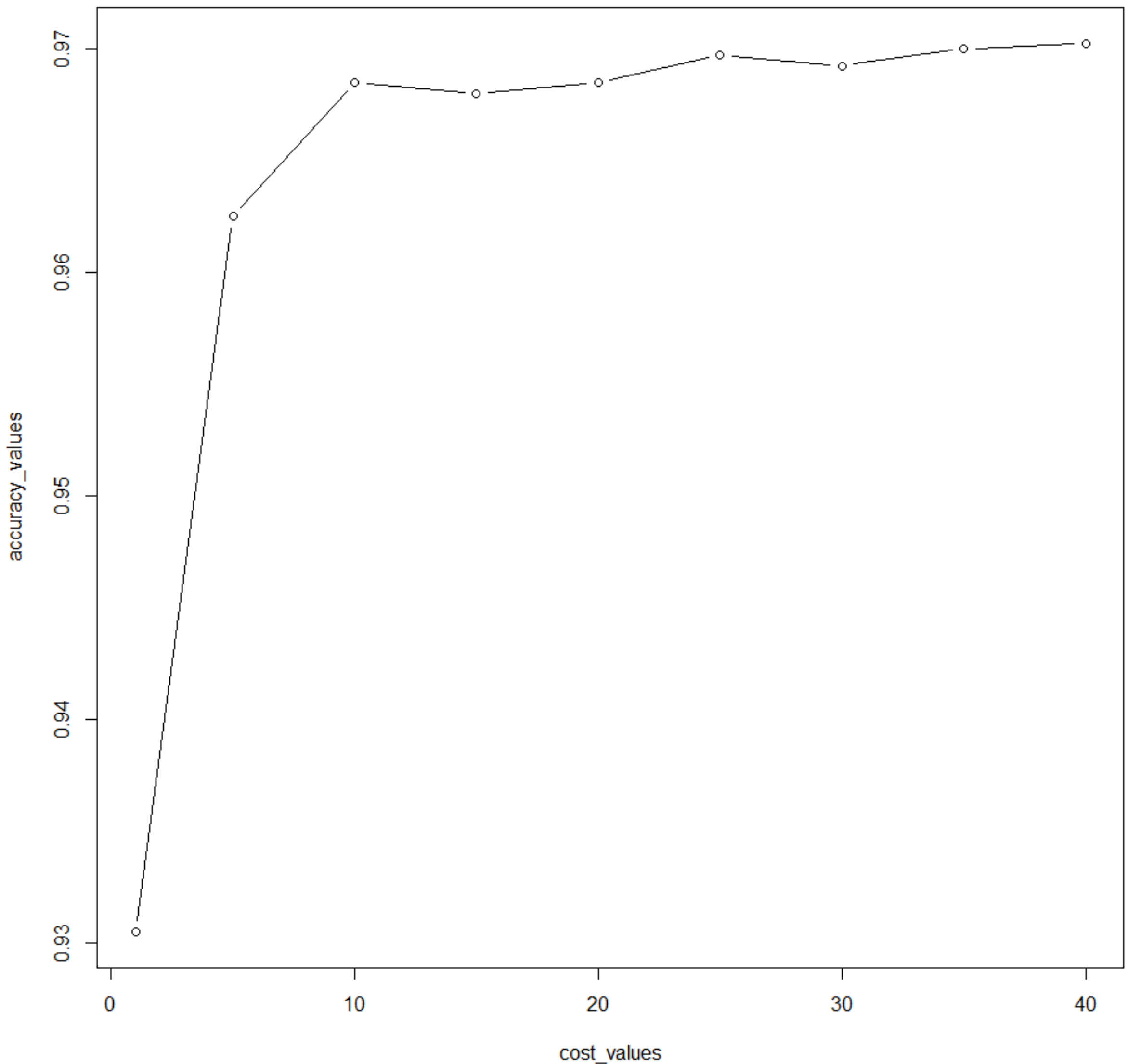
```
> letter_classifier_rbf <- ksvm(letter ~ ., data = letters_train,
+                                kernel = "rbfdot")
> letter_predictions_rbf <- predict(letter_classifier_rbf,
+                                    letters_test)
```

Then compared the accuracy with the linear SVM using the following code and I was able to see that by changing the kernel function I was able to increase the accuracy of the model from 84 percent to 93 percent:

```
> agreement_rbf <- letter_predictions_rbf == letters_test$letter
> table(agreement_rbf)
agreement_rbf
FALSE  TRUE
  281  3719
> prop.table(table(agreement_rbf))
agreement_rbf
  FALSE    TRUE
0.07025 0.92975
>
```

Next, to try and increase the model performance even further, I identified the best SVM cost parameter. I used the sapply() function to apply a custom function to a vector of potential cost values. The seq() function to generate the vector as a sequence counting from to forty, by five and the plot() function to help visualize the result:

```
> cost_values <- c(1, seq(from = 5, to = 40, by = 5))
> accuracy_values <- sapply(cost_values, function(x) {
+     set.seed(12345)
+     m <- ksvm(letter ~ ., data = letters_train,
+               kernel = "rbfdot", C = x)
+     pred <- predict(m, letters_test)
+     agree <- ifelse(pred == letters_test$letter, 1, 0)
+     accuracy <- sum(agree) / nrow(letters_test)
+     return (accuracy)
+   })
```



By identifying the best SVM cost parameter, I was able to increase the accuracy of the model to 97 percent!

References

Lantz, B. (2019). *Machine Learning with R* (3rd ed.). Packt Publishing. https://mbsdirect.vitalsource.com/books/9781788291552