R Notebook

Bheeni Garg

Optical Character Recognition using Support Vector Machines—-

Step 1: Collecting Data

This exercise uses a dataset donated to the UCI Machine Learning Data Repository link by W. Frey and D. J. Slate. The dataset contains 20,000 examples of 26 English alphabet capital letters as printed using 20 different randomly reshaped and distorted black and white fonts.

Step 2: Exploring and preparing the data —-

```
# read in data and examine structure
letters <- read.csv("letterdata.csv")
str(letters)
```

```
## 'data.frame':    20000 obs. of  17 variables:
##  $ letter: Factor w/ 26 levels "A","B","C","D",..: 20 9 4 14 7 19 2 1 10 13 ...
##  $ 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 ...
```

```
# divide into training and test data
letters_train <- letters[1:16000, ]
letters_test  <- letters[16001:20000, ]
```

Step 3: Training a model on the data —-

```
# begin by training a simple linear SVM
library(kernlab)
letter_classifier <- ksvm(letter ~ ., data = letters_train,
                          kernel = "vanilladot")
```

```
##  Setting default kernel parameters
```

```
# look at basic information about the model
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 -18.1824 -62
## Training error : 0.130062
```

Step 4: Evaluating model performance —-

```
# predictions on testing dataset
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
##                  A 144   0   0   0   0   0   0   0   0   1   0   0   1   2
##                  B   0 121   0   5   2   0   1   2   0   0   1   0   1   0
##                  C   0   0 120   0   4   0  10   2   2   0   1   3   0   0
##                  D   2   2   0 156   0   1   3  10   4   3   4   3   0   5
##                  E   0   0   5   0 127   3   1   1   0   0   3   4   0   0
##                  F   0   0   0   0   0 138   2   2   6   0   0   0   0   0
##                  G   1   1   2   1   9   2 123   2   0   0   1   2   1   0
##                  H   0   0   0   1   0   1   0 102   0   2   3   2   3   4
##                  I   0   1   0   0   0   1   0   0 141   8   0   0   0   0
##                  J   0   1   0   0   0   1   0   2   5 128   0   0   0   0
##                  K   1   1   9   0   0   0   2   5   0   0 118   0   0   2
##                  L   0   0   0   0   2   0   1   1   0   0   0 133   0   0
##                  M   0   0   1   1   0   0   1   1   0   0   0   0 135   4
##                  N   0   0   0   0   0   1   0   1   0   0   0   0   0 145
##                  O   1   0   2   1   0   0   1   2   0   1   0   0   0   1
##                  P   0   0   0   1   0   2   1   0   0   0   0   0   0   0
##                  Q   0   0   0   0   0   0   8   2   0   0   0   3   0   0
##                  R   0   7   0   0   1   0   3   8   0   0  13   0   0   1
##                  S   1   1   0   0   1   0   3   0   1   1   0   1   0   0
##                  T   0   0   0   0   3   2   0   0   0   0   1   0   0   0
##                  U   1   0   3   1   0   0   0   2   0   0   0   0   0   0
##                  V   0   0   0   0   0   1   3   4   0   0   0   0   1   2
##                  W   0   0   0   0   0   0   1   0   0   0   0   0   2   0
##                  X   0   1   0   0   2   0   0   1   3   0   1   6   0   0
```

```
##                   Y   3   0   0   0   0   0   0   1   0   0   0   0   0   0
##                   Z   2   0   0   0   1   0   0   0   3   4   0   0   0   0
##
## letter_predictions   O   P   Q   R   S   T   U   V   W   X   Y   Z
##                   A   2   0   5   0   1   1   1   0   1   0   0   1
##                   B   0   2   2   3   5   0   0   2   0   1   0   0
##                   C   2   0   0   0   0   0   0   0   0   0   0   0
##                   D   5   3   1   4   0   0   0   0   0   3   3   1
##                   E   0   0   2   0  10   0   0   0   0   2   0   3
##                   F   0  16   0   0   3   0   0   1   0   1   2   0
##                   G   1   2   8   2   4   3   0   0   0   1   0   0
##                   H  20   0   2   3   0   3   0   2   0   0   1   0
##                   I   0   1   0   0   3   0   0   0   0   5   1   1
##                   J   1   1   3   0   2   0   0   0   0   1   0   6
##                   K   0   1   0   7   0   1   3   0   0   5   0   0
##                   L   0   0   1   0   5   0   0   0   0   0   0   1
##                   M   0   0   0   0   0   0   3   0   8   0   0   0
##                   N   0   0   0   3   0   0   1   0   2   0   0   0
##                   O  99   3   3   0   0   0   3   0   0   0   0   0
##                   P   2 130   0   0   0   0   0   0   0   0   1   0
##                   Q   3   1 124   0   5   0   0   0   0   0   2   0
##                   R   1   1   0 138   0   1   0   1   0   0   0   0
##                   S   0   0  14   0 101   3   0   0   0   2   0  10
##                   T   0   0   0   0   3 133   1   0   0   0   2   2
##                   U   1   0   0   0   0   0 152   0   0   1   1   0
##                   V   1   0   3   1   0   0   0 126   1   0   4   0
##                   W   0   0   0   0   0   0   4   4 127   0   0   0
##                   X   1   0   0   0   1   0   0   0   0 137   1   1
##                   Y   0   7   0   0   0   3   0   0   0   0 127   0
##                   Z   0   0   0   0  18   3   0   0   0   0   0 132
```

```
# look only at agreement vs. non-agreement
# construct a vector of TRUE/FALSE indicating correct/incorrect predictions
agreement <- letter_predictions == letters_test$letter
table(agreement)
```

```
## agreement
## FALSE  TRUE
##   643  3357
```

```
prop.table(table(agreement))
```

```
## agreement
##   FALSE    TRUE
## 0.16075 0.83925
```

Step 5: Improving model performance —-

```
set.seed(12345)
letter_classifier_rbf <- ksvm(letter ~ ., data = letters_train, kernel = "rbfdot")
letter_predictions_rbf <- predict(letter_classifier_rbf, letters_test)

agreement_rbf <- letter_predictions_rbf == letters_test$letter
table(agreement_rbf)
```

```
## agreement_rbf
## FALSE  TRUE
##   275  3725
```

```
prop.table(table(agreement_rbf))
```

```
## agreement_rbf
##   FALSE    TRUE
## 0.06875 0.93125
```

Thus, in percentage terms, the accuracy is about 93 percent which is pretty good!