Classification Performance of Linear Regression and k-Nearest Neighbor on Zip Code Data

Lucy Ly

2023-12-24

This exercise was derived from "The Elements of Statistical Learning," chapter 2 exercise 2.8: "Compare the classification performance of linear regression and k-nearest neighbor classification on the zip code data. In particular, consider only the 2's and 3's, and k = 1, 3, 5, 6, and 15. Show both the training and test error for each choice."

The `zipcode` data are normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations. The images have been de-slanted and size normalized, resulting in 16x16 gray scale images. The data are in two gzipped files, and each line consists of the digit id (0-9) (column 1) followed by the 256 gray scale values (columns 2 to 257).

```
library(class)
library(caret)

train <- read.table("zip.train.gz")
test <- read.table("zip (1).test.gz")
```

```
# Subset rows for reading 2's and 3's only
filtered_train <- subset(train, V1 %in% c(2, 3))
filtered_test <- subset(test, V1 %in% c(2, 3))
rownames(filtered_train) <- NULL
rownames(filtered_test) <- NULL

# Setting up train and test data
y_train = data.frame(filtered_train$V1)
x_train = filtered_train[, c(2:257)]
y_test = data.frame(filtered_test$V1)
```

```
x_test = filtered_test[, c(2:257)]

# Reassigning values for classification
filtered_train$V1[filtered_train$V1 == 3] = 1
filtered_train$V1[filtered_train$V1 == 2] = 0
filtered_test$V1[filtered_test$V1== 3] = 1
filtered_test$V1[filtered_test$V1 == 2] = 0

# A utility function to assign prediction
reassign <- function(df, column_name) {
  column_values <- df[[column_name]]
```

```
  df[[column_name]] <- ifelse(column_values >= 05, 1, 0)
}
```

```
# Performing Linear Regression on training sets
lm_model <- lm(V1 ~ . , data = filtered_train)

# Predictions on the training set
pred_train <- predict(lm_model, newdata = x_train)
pred_train_df = data.frame(pred_train)
reassign(pred_train_df, 'pred_train')

# Calculate and print the training error rate
train_error_rate_df <- (filtered_train$V1 - pred_train_df) ** 2
print("Mean squared error of training set:")
```

```
## [1] "Mean squared error of training set:"
```

```
print(mean(train_error_rate_df$pred_train))
```

```
## [1] 0.02481172
```

```
# Predictions on the test set using model
pred_test <- predict(lm_model, newdata = x_test)
pred_test_df = data.frame(pred_test)
reassign(pred_test_df, 'pred_test')

# Calculate and print the test error rate
test_error_rate_df <- (filtered_test$V1 - pred_test_df) ** 2
print("Mean squared error of test set:")
```

```
## [1] "Mean squared error of test set:"
```

```
print(mean(test_error_rate_df$pred_test))
```

```
## [1] 0.1516653
```

```
# Setting up data for knn regression
knn_train_x = filtered_train[,-1]
knn_train_x = scale(knn_train_x)[,]
knn_train_y = filtered_train[,1]

knn_test_x = filtered_test[,-1]
knn_test_x = scale(knn_test_x)[,]
knn_test_y = filtered_test[,1]
```

```
k_and_mse = data.frame()

for (j in c(1, 3, 5, 7, 15)){
  knn_model_fit = knnreg(x = knn_train_x, y = knn_train_y, k=j)
  knn_pred_y = predict(knn_model_fit, knn_test_x)
```

```
  knn_pred_y2 = predict(knn_model_fit, knn_train_x)

  training_mse = mean((knn_train_y - knn_pred_y2)^2)
  test_mse = mean((knn_test_y - knn_pred_y)^2)
  k_and_mse <- rbind(k_and_mse, c(j, training_mse, test_mse))
}

names(k_and_mse) = c("k", "Training MSE", "Test MSE")
print(k_and_mse)
```

```
##    k Training MSE   Test MSE
## 1  1  0.000000000 0.02747253
## 2  3  0.005119590 0.02533578
## 3  5  0.006450684 0.02505495
## 4  7  0.008183835 0.02685580
## 5 15  0.012724607 0.03358383
```

For the linear regression, the percent mean squared error of training set was about 0.025, while the same of the test set was about 0.152. The percentage MSE for the training and test sets for the KNN regressions are shown above in the table.

The classification performance of the linear regression is evidently significantly worse than that for the KNN regression. And as the number of neighbors `k` increases, we see that both the KNN train and test error rates tend to increase. The test MSE for KNN briefly dips before increasing, which is characteristic of the prediction error for the test sample for K-nearest neighbors.