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

Lucy Ly

2023-12-24

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(readr)
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
library(class)

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))

# Setting up train and test data
y_train = filtered_train["V1"]
x_train = filtered_train[, c(2:257)]
y_test = 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
y_train$V1[y_train$V1 == 3] = 1
y_train$V1[y_train$V1 == 2] = 0
y_test$V1[y_test$V1 == 3] = 1
y_test$V1[y_test$V1 == 2] = 0

# A utility function to assign prediction
assign <- function(arr) {
  arr[arr >= 0.5] <- 1
  arr[arr < 0.5] <- 0
}

# A utility function to calculate error rate of predictions
```

```r
getErrorRate <- function(true_y, pred_y) {
  diff = true_y - pred_y
  diff_squared = diff * diff
  return(mean(diff_squared))
}
```

```r
# 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)

# Calculate and print the training error rate
train_error_rate_df <- (y_train - pred_train_df) ** 2
print(mean(train_error_rate_df$V1))
```

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

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

# Calculate and print the test error rate
test_error_rate_df <- (y_test - pred_test_df) ** 2
print(mean(test_error_rate_df$V1))
```

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

```r
# Running separate k-Nearest Neighbors classifiers:

normsq <- function(x) {x %*% x}

perm.nearest <- function(x, training_xs) {
  rows <- nrow(training_xs)
  repx <- matrix(rep(1, rows), nrow = rows) %*% x
  temp <- apply(training_xs - repx, 1, normsq)
  order(temp)
}

nn.value <- function(k, training_ys, perm) {
  perm <- perm[1:k]
  sum(training_ys[perm]) / k
}

y_test_mat = as.matrix(y_test)
x_test_mat = as.matrix(x_test)
y_train_mat = as.matrix(y_train)
x_train_mat = as.matrix(x_train)


rec.err <- function(test_xs, test_ys, training_xs, training_ys, kays) {
  lkays <- length(kays)
  rec <- matrix(rep(0, 3 * lkays), nrow = lkays, ncol = 3)

  for (j in 1:lkays) {
    k <- kays[[j]]
    rec[j, 1] <- k
  }

  for (i in 1:nrow(test_xs)) {
    x <- test_xs[i,]
    perm <- perm.nearest(x, training_xs)

    for (j in 1:lkays) {
      k <- kays[[j]]
      estimate <- nn.value(k, training_ys, perm)
      error <- estimate - test_ys[i]

      if (abs(error) > 1) {
        rec[j, 2] <- rec[j, 2] + 1
      }

      rec[j, 3] <- rec[j, 3] + (error^2)
    }

    if (i %% 100 == 0) cat("row ", i, ", ", sep = "")
    if (i %% 800 == 0) cat("\n")
  }

  print(rec)
}

rec.err(x_test_mat, y_test_mat, x_train_mat, y_train_mat, c(1, 3, 5, 7, 15))
```

```
## row 100, row 200, row 300,      [,1] [,2]     [,3]
## [1,]    1    0 9.000000
## [2,]    3    0 8.333333
## [3,]    5    0 8.320000
## [4,]    7    0 8.571429
## [5,]   15    0 9.760000
```