HW5

2024-11-30

```
library(data.table)
set.seed(123)
load_digits <- function(subset=NULL, normalize=TRUE) {
  df <- fread("digits.csv")
  df <- as.matrix(df)
  if (length(subset) > 0) {
    c <- dim(df)[2]
    l_col <- df[, c]
    index <- NULL
    for (i in 1:length(subset)){
      number <- subset[i]
      index <- c(index, which(l_col == number))
    }
    df <- df[sort(index), ]
  }
  digits <- df[, -1]
  labels <- df[, c]
  if (normalize) {
    digits <- digits - min(digits)
    digits <- digits / max(digits)
  }
  for (i in 1:length(subset)) {
    labels[labels == subset[i]] <- i - 1
  }

  return(list(digits, labels))
}

result <- load_digits(subset=c(1, 7), normalize=TRUE)
digits <- result[[1]]
labels <- result[[2]]
split_samples <- function(digits, labels) {
  num_samples <- dim(digits)[1]
  num_training <- round(num_samples * 0.7)
  indices <- sample(1:num_samples, size = num_samples)
  training_idx <- indices[1:num_training]
  testing_idx <- indices[-(1:num_training)]

  return(list(digits[training_idx, ], labels[training_idx], digits[testing_idx, ], labels[testing_idx]))
}

result <- split_samples(digits, labels)
training_digits <- result[[1]]
training_labels <- result[[2]]
```

```r
testing_digits <- result[[3]]
testing_labels <- result[[4]]

linear_svm <- function(training_digits, training_labels, testing_digits, testing_labels, C = 1) {
  n <- nrow(training_digits)
  d <- ncol(training_digits)
  w <- matrix(0, nrow = d, ncol = 1)
  b <- 0
  for (epoch in 1:100) {
    for (i in 1:n) {
      # Compute the margin
      margin <- training_labels[i] * (as.vector(t(w)) %*% training_digits[i, ] + b)

      if (margin < 1) {
        w <- w + C * (training_labels[i] * matrix(training_digits[i, ], nrow = d, ncol = 1))
        b <- b + C * training_labels[i]
      }
    }
  }
  predict <- function(x) {
    if (as.vector(t(w)) %*% x + b >= 0) {
      return(1)
    } else {
      return(0)
    }
  }
  train_predictions <- apply(training_digits, 1, predict)
  test_predictions <- apply(testing_digits, 1, predict)

  training_accuracy <- mean(train_predictions == training_labels)
  testing_accuracy <- mean(test_predictions == testing_labels)

  return(list(training_accuracy = training_accuracy, testing_accuracy = testing_accuracy))
}
model_results <- linear_svm(training_digits, training_labels, testing_digits, testing_labels, C = 1)
cat("Training Accuracy: ", model_results$training_accuracy, "\n")
```

```
## Training Accuracy:  0.4782609
```

```r
cat("Testing Accuracy: ", model_results$testing_accuracy, "\n")
```

```
## Testing Accuracy:  0.537037
```

```r
C_values <- c(0.01, 0.1, 1, 10, 100)
results_c <- data.frame(C = C_values, Training_Accuracy = NA, Testing_Accuracy = NA)

for (i in 1:length(C_values)) {
  model_results <- linear_svm(training_digits, training_labels, testing_digits, testing_labels, C = C_va
  results_c$Training_Accuracy[i] <- model_results$training_accuracy
  results_c$Testing_Accuracy[i] <- model_results$testing_accuracy
}

print(results_c)
```

```
##       C Training_Accuracy Testing_Accuracy
## 1 1e-02         0.4782609         0.537037
## 2 1e-01         0.4782609         0.537037
## 3 1e+00         0.4782609         0.537037
## 4 1e+01         0.4782609         0.537037
## 5 1e+02         0.4782609         0.537037
```

```r
poly_kernel <- function(x1, x2, degree = 2) {
  return((x1 %*% t(x2) + 1) ^ degree)
}

kernel_svm <- function(training_digits, training_labels, testing_digits, testing_labels, C = 1) {
  n <- nrow(training_digits)
  alpha <- rep(0, n)
  threshold <- 0.001
  max_iter <- 100
  degree <- 2

  for (iter in 1:max_iter) {
    for (i in 1:n) {
      output <- sum(alpha * training_labels * poly_kernel(training_digits, training_digits[i, , drop = F
      error <- training_labels[i] - output
      if (!is.na(error) && abs(error) > threshold) {
        alpha[i] <- alpha[i] + C * error
      }
    }
  }
  predict_kernel <- function(x) {
    output <- sum(alpha * training_labels * poly_kernel(training_digits, t(x), degree))
    return(ifelse(output >= 0, 1, 0))
  }

  train_predictions <- apply(training_digits, 1, predict_kernel)
  test_predictions <- apply(testing_digits, 1, predict_kernel)

  training_accuracy <- mean(train_predictions == training_labels)
  testing_accuracy <- mean(test_predictions == testing_labels)

  return(list(training_accuracy = training_accuracy, testing_accuracy = testing_accuracy))
}
model_results_kernel <- kernel_svm(training_digits, training_labels, testing_digits, testing_labels, C =
cat("Kernel SVM Training Accuracy: ", model_results_kernel$training_accuracy, "\n")
```

```
## Kernel SVM Training Accuracy:  NA
```

```r
cat("Kernel SVM Testing Accuracy: ", model_results_kernel$testing_accuracy, "\n")
```

```
## Kernel SVM Testing Accuracy:  NA
```