Лабораторная работа №7 «Бустинг»

Работу выполнила студентка группы 5140201/30301 Фазылова Алика

Задание 1.

Исследуйте зависимость тестовой ошибки от количества деревьев в ансамбле для алгоритма adaboost.M1 на наборе данных Vehicle из пакета mlbench (обучающая выборка должна состоять из 7/10 всех прецедентов, содержащихся в данном наборе данных). Постройте график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, . . . , 301, объясните полученные результаты.

Решение

Построим график зависимости тестовой ошибки от количества деревьев (рисунок 1)

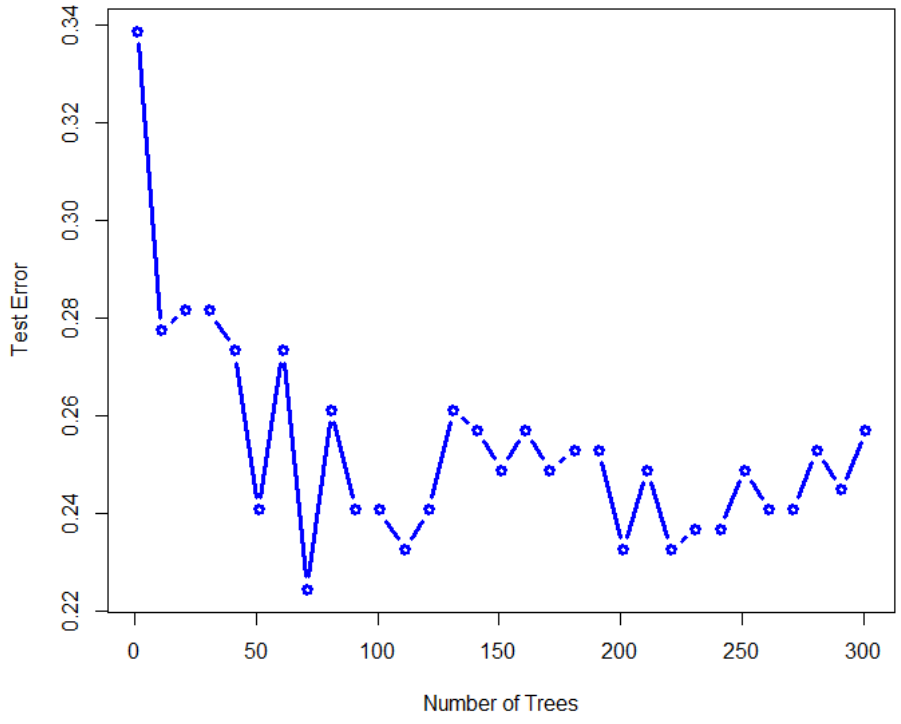

Рисунок 1 – Зависимость ошибки от количества деревьев

Минимальная ошибка на тестовой выборке достигается при 71 деревьях и равна 0.2244898, а максимальная 0.3387755 при 1 дереве.

На графике видно с увеличением количества деревьев ошибка в среднем уменьшается.

Листинг кода 1 задачи:

```
library(adabag)
library(mlbench)

data(Vehicle)
Vehicle
set.seed(14)
sample <- sample(c(TRUE, FALSE), nrow(Vehicle), replace = TRUE, prob = c(0.7, 0.3))
train_data <- Vehicle[sample, ]
test_data <- Vehicle[!sample, ]

maxdepth <- 5
test_errors <- numeric(length = 31)

for (i in seq(1, 301, by = 10)) {
  model <- boosting(Class ~ ., data = train_data, mfinal = i, maxdepth = maxdepth)
  predictions <- predict.boosting(model, newdata = test_data)
  test_errors[i %/% 10 + 1] <- predictions$error
}

plot(seq(1, 301, by = 10), test_errors, type = "b", xlab = "Number of Trees", ylab = "Test Error",col = "blue",lwd=3)
print(data.frame(Number_of_Trees = seq(1, 301, by = 10), Test_Error = test_errors))

min_error <- min(test_errors)
max_error <- max(test_errors)
min_error_index <- which.min(test_errors)
max_error_index <- which.max(test_errors)
min_error_trees <- seq(1, 301, by = 10)[min_error_index]
max_error_trees <- seq(1, 301, by = 10)[max_error_index]
cat("Минимальная тестовая ошибка:", min_error, "достигается при", min_error_trees, "деревьях")
cat("Максимальная тестовая ошибка:", max_error, "достигается при", max_error_trees, "деревьях")
```

Задание 2.

Исследовать зависимость тестовой ошибки от количества деревьев в ансамбле для алгоритма bagging на наборе данных Glass из пакета mlbench (обучающая выборка должна состоять из 7/10 всех прецедентов, содержащихся в данном наборе данных). Построить график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, ... , 201, объясните полученные результаты.

Решение

Построим график зависимости тестовой ошибки от количества деревьев (рисунок 2)

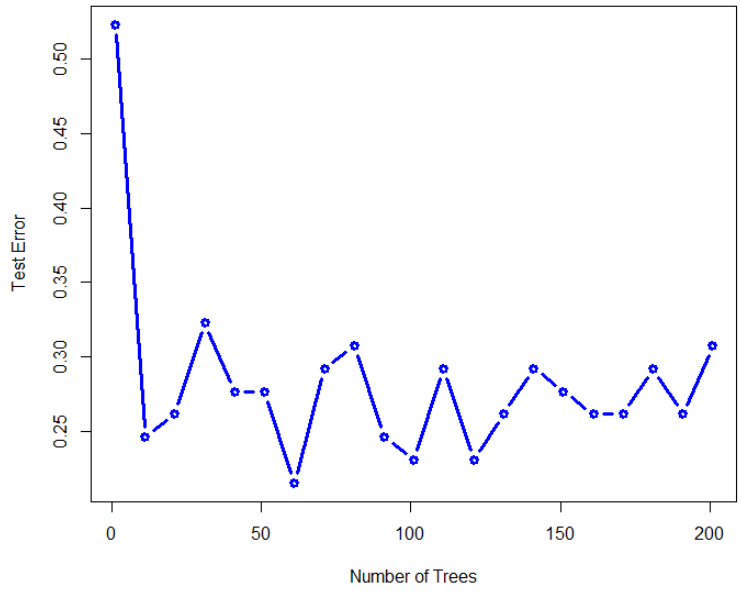

Рисунок 1 – Зависимость ошибки от количества деревьев

Минимальная ошибка на тестовой выборке достигается при 61 дереве и равна 0.2153846, а максимальная 0.5230769 при 1 дереве.

На графике видно с увеличением количества деревьев ошибка в среднем уменьшается.

Листинг кода 2 задачи:

```
library(adabag)
library(mlbench)
```

```
data(Glass)

set.seed(3)

sample <- sample(c(TRUE, FALSE), nrow(Glass), replace = TRUE, prob = c(0.7, 0.3))
train_data <- Glass[sample, ]
test_data <- Glass[!sample, ]

test_errors2 <- numeric(length = 21)

for (i in seq(1, 201, 10)) {
  model <- bagging(Type ~ ., train_data, mfinal = i)
  prediction <- predict.bagging(model, test_data)
  test_errors2[i %/% 10 + 1] <- prediction$error
}

plot(seq(1, 201, by = 10), test_errors2, xlab = "Number of Trees", ylab = "Test Error", type = "b",col = "blue",lwd=3)

min_error2 <- min(test_errors2)
max_error2 <- max(test_errors2)
min_error_index2 <- which.min(test_errors2)
max_error_index2 <- which.max(test_errors2)
min_error_trees2 <- seq(1, 201, by = 10)[min_error_index2]
max_error_trees2 <- seq(1, 201, by = 10)[max_error_index2]

cat("Минимальная тестовая ошибка:", min_error2, "достигается при", min_error_trees2, "деревьях")
cat("Максимальная тестовая ошибка:", max_error2, "достигается при", max_error_trees2, "деревьях")
```

Задание 3.

Реализуйте бустинг алгоритм с классификатором K ближайших соседей. Сравните тестовую ошибку, полученную с использованием данного классификатора на наборах данных Vehicle и Glass, с тестовой ошибкой, полученной с использованием единичного дерева классификации.

Решение

В результате работы был реализован бустинг алгоритм с классификатором K ближайших соседей и оценены ошибки на тестовой выборке.

Для набора данных Vehicle

Ошибка на тестовой выборке для единичного дерева: 0.3755102

Ошибка на тестовой выборке для Knn бустинг алгоритма: 0.4285714

Для набора данных Glass

Ошибка на тестовой выборке для единичного дерева: 0.296875

Ошибка на тестовой выборке для Knn бустинг алгоритма: 0.40625

Для обоих наборов данных ошибка для единичного дерева оказалась меньше.

Листинг кода 3 задачи:

```
library(rpart)
library(mlbench)
library(adabag)
library(dplyr)


knn_w <- function(target, train, k, w) {
  return(list(target = target, train = train, levels= levels(train[, target]), k = k, w = w))
}


knn_w_pred <- function(clfier, testdata) {
  n <- nrow(testdata)
  pred <- rep(NA_character_, n)
  trainlabels <- clfier$train[, clfier$target]
```

```
  train <- clfier$train[, !(names(clfier$train) %in%clfier$target)]
  test <- testdata[, !(names(testdata) %in%clfier$target)]

  for (i in 1:n) {
    n_number <- order(apply(train, 1, function(x)
      sum((test[i,] - x)^2)))[1:clfier$k]

    myfreq <- data.frame(names = clfier$levels, freq
                         = rep(0, length(clfier$levels)))
    for (t in n_number) {
      myfreq[myfreq$names == trainlabels[t], ][2] <-
        myfreq[myfreq$names == trainlabels[t], ][2] +
        clfier$w[t]
    }
    most_frequent <- clfier$levels[myfreq$freq ==
                                   max(myfreq$freq)]
    pred[i] <- sample(most_frequent, 1)
  }

  factor(pred, levels = levels(trainlabels))
}

knn_boosting <- function(target, data, k = 11, mfinal= 2, ...) {

  n <- nrow(data)
  w <- rep(1/n, each = n)

  classifiers <- list()
  alphas <- vector()

  for (t in 1:mfinal) {
    clfier <- knn_w(target, train = data, k = k, w)
    knn_predicted <- knn_w_pred(clfier, data)
    error <- vector()
    for (i in 1:n) {
```

```
    if (data[[target]][i] != knn_predicted[i])
      error <- append(error, w[i])
  }

  if (sum(error) >= 0.5) {
   break()
  }

  classifiers[[t]] <- clfier
  alphas[[t]] <- log((1 - sum(error)) /sum(error)) / 2

  for (i in 1:n) {
   if (knn_predicted[i] != data[[target]][i]) {
     w[i] <- w[i]*exp(alphas[[t]])
   } else {
     w[i] <- w[i]*exp(-alphas[[t]])
   }
  }
 }

 result <- list()
 result$classifiers <- classifiers
 result$alphas <- alphas
 result$levels <- levels(data[, target])
 return(result)
}

boosting_pred <- function(clfier, testdata) {
 n <- nrow(testdata)
 pred = rep(NA_character_, n)

 for (i in 1:n) {
  myfreq <- data.frame(names = clfier$levels, freq
                = rep(0, length(clfier$levels)))
```

```
    for (j in 1:length(clfier$classifiers)) {
      prediction <-
        knn_w_pred(clfier$classifiers[[j]], testdata[i, ])
      myfreq[myfreq$names == prediction, ][2] <-
        myfreq[myfreq$names == prediction, ][2] +
        clfier$alphas[j]
    }

    most_frequent = clfier$levels[myfreq$freq ==
                                  max(myfreq$freq)]
    pred[i] <- sample(most_frequent, 1)
  }
  factor(pred, levels = clfier$levels)
}

data(Vehicle)
set.seed(14)

sample <- sample(c(TRUE, FALSE), nrow(Vehicle), replace = TRUE, prob = c(0.7, 0.3))
Vehicle_train <- Vehicle[sample, ]
Vehicle_test <- Vehicle[!sample, ]

Vehicle_rpart <- rpart(Class ~ ., data = Vehicle_train, maxdepth = 5)
Vehicle_rpart_pred <- predict(Vehicle_rpart, newdata= Vehicle_test, type = 'class')
tbl_rpart <- table(Vehicle_rpart_pred,Vehicle_test$Class)
error.rpart <- 1 - (sum(diag(tbl_rpart)) / sum(tbl_rpart))
cat("Ошибка для единичного дерева (Vehicle):",error.rpart)

clfier <- knn_boosting('Class', Vehicle_train,mfinal = 1)
pred <- boosting_pred(clfier, Vehicle_test)
tbl_knn <- table(Vehicle_test$Class, pred)
error.kknn <- 1 - sum(diag(tbl_knn)) / sum(tbl_knn)
cat("Ошибка для KNN-бустинг алгоритма (Vehicle):",error.kknn)

data("Glass")
```

```
set.seed(14)
sample <- sample(c(TRUE, FALSE), nrow(Glass), replace = TRUE, prob = c(0.7, 0.3))
Glass_train <- Glass[sample, ]
Glass_test <- Glass[!sample, ]

Glass_rpart <- rpart(Type ~ ., data = Glass_train, maxdepth = 5)
Glass_rpart_pred <- predict(Glass_rpart, Glass_test,type = 'class')
tbl_rpart <- table(Glass_rpart_pred,Glass_test$Type)
error.rpart <- 1 - (sum(diag(tbl_rpart)) /sum(tbl_rpart))
cat("Ошибка для единичного дерева (Glass):",error.rpart)

clfier <- knn_boosting('Type', Glass_train, mfinal = 1)
pred <- boosting_pred(clfier, Glass_test)
tbl_knn <- table(Glass_test$Type, pred)
error.kknn <- 1 - sum(diag(tbl_knn)) / sum(tbl_knn)
cat("Ошибка для KNN-бустинг алгоритма (Glass):",error.kknn)
```