Министерство образования и науки РФ
Санкт-Петербургский Политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 7

«Boostig»

по дисциплине «Машинное обучение»

Выполнила:

студентка гр. 3540201/20301 _____ Климова О. А.

подпись, дата

Проверил:

д.т.н., проф. _____ Уткин Л. В.

подпись, дата

Санкт-Петербург

2022

Содержание

Постановка задачи

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.

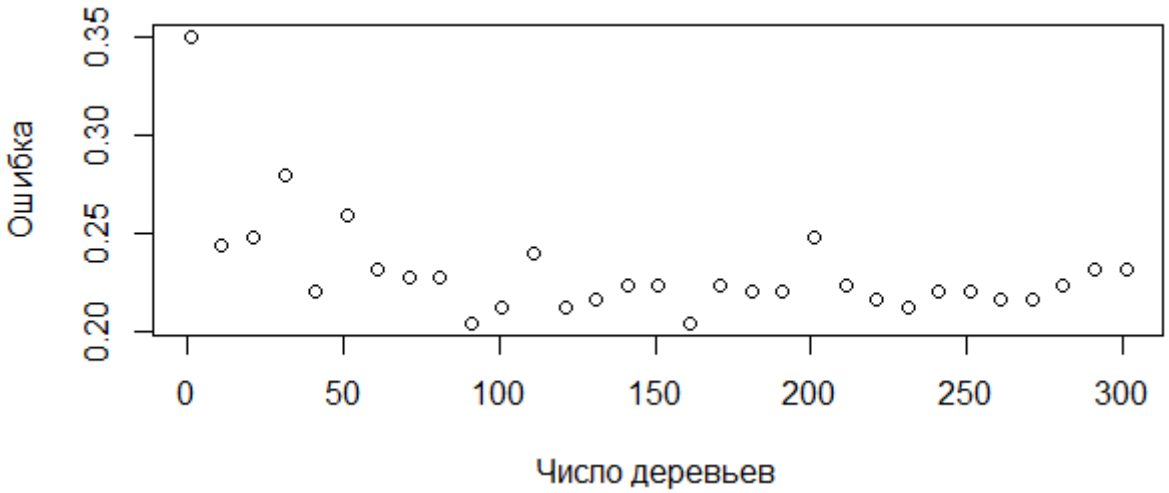
2) Исследуйте зависимость тестовой ошибки от количества деревьев в ансамбле для алгоритма bagging на наборе данных Glass из пакета mlbench (обучающая выборка должна состоять из 7/10 всех прецедентов, содержащихся в данном наборе данных). Постройте график зависимости тестовой ошибки при числе деревьев,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

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.

1 Задание 1

В данном задании была исследована 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).

Был построен график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, . . ., 301:



По полученному графику можно видеть, что максимальная ошибка (0.3503937) при одном дереве в ансамбле, а минимальная (0.2047244) при 91 и 161.

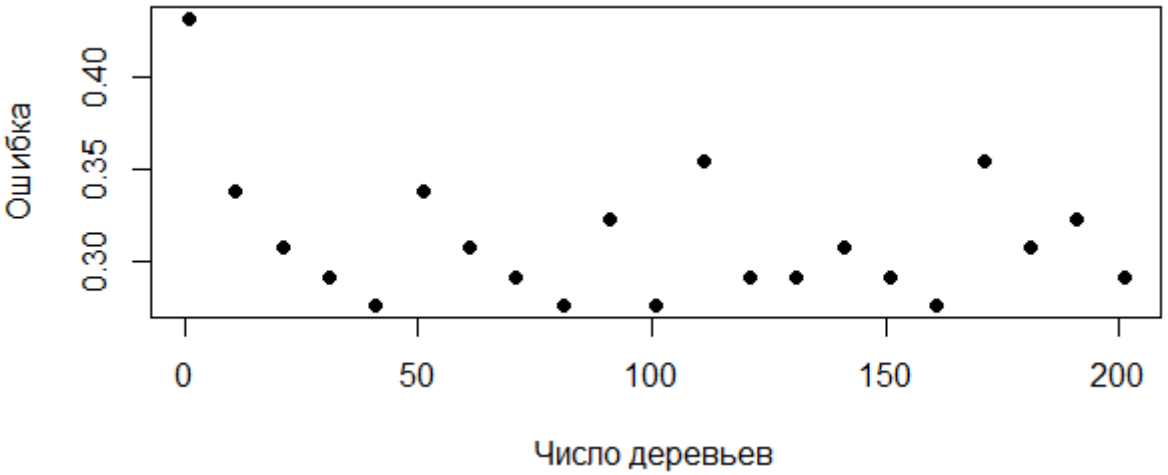
Максимальная глубина дерева была взята равной 5: maxdepth = 5.

Код представлен в Приложении 1.

2 Задание 2

В данном задании была исследована 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).

Был построен график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, . . ., 201:



По полученному графику можно видеть, что самая большая ошибка при одном дерев в ансамбле (0.4307692), а самая маленькая (0.2769231) при 41, 81, 101, 161 дереве в ансамбле.

Максимальная глубина дерева была взята равной 5: maxdepth = 5.

Код представлен в Приложении 2.

3 Задание 3

Функция в R, реализующая алгоритм adaboost использует в качестве слабых классификаторов только деревья решений (нельзя использовать слабые классификаторы k ближайших соседей), поэтому в данном задании были сравнены ошибки при использовании слабого классификатора на k ближайших соседей и единичного дерева классификации.

Тестовая ошибка, полученная с использованием классификатора на k ближайших соседей для данных Vehicle и Glass: 0.2755906 и 0.2598425.

Тестовая ошибка, полученная с использованием единичного дерева классификации для Vehicle и Glass: 0.2615385 и 0.3846154.

Можно сделать вывод о том, что точность классификации для датасета Vehicle выше при использовании единичного дерева, а для Glass при использовании слабого классификатора на основе k ближайших соседей.

Код представлен в приложении 3.

Бустинг на knn

Был реализован бустинг алгоритм, использующий k = 5 ближайших соседей. Тестовая ошибка, полученная с использованием данного алгоритма для Vehicle и Glass составила: 0.3133333 и 0.3692308.

Таким образом, можно сделать вывод о том, что для данных датасетов тестовая ошибка на единичном дереве оказалась больше для Glass, но меньше для Vehicle, чем на реализованном бустинге.

Код представлен в приложении 4.

Приложение 1. Код для задания 1

```
#ЗАДАНИЕ 1
library(rpart)
library(mlbench)
library(adabag)
data(Vehicle)
n <- dim(Vehicle)[1]
Vehicle_rand <- Vehicle[ order(runif(n)),]
#разделим выборку на 30% теста и 70% для обучения
nt <- as.integer(n*0.7)
Vehicle_train <- Vehicle_rand[1:nt, ]
Vehicle_test <- Vehicle_rand[(nt+1):n, ]

#Зададим максимальное число итераций в алг. adaboost.M1, равным от 1 до 301 с шагом
10:
mfinal = c()
mfinal <- append(mfinal, 1)
for(i in 1:30){
  mfinal <- append(mfinal, 1+10*i)
}
#Зададим максимальную глубину каждого дерева:
maxdepth <- 5
err = c()

for(i in 1:length(mfinal)){
#Построим ансамбль деревьев решений
Vehicle.adaboost <- boosting(Class ~.,data=Vehicle_train, mfinal=mfinal[i],
maxdepth=maxdepth)
#Используя построенную модель, предскажем ответы на тестовой выборке
Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost, Vehicle_test)
#Вычислим ошибку предсказания:
err <- append(err, Vehicle.adaboost.pred$error)
}
plot(mfinal, err, xlab="Число деревьев", ylab="Ошибка")
err
```

Приложение 2. Код для задания 2

```
#ЗАДАНИЕ 2
library(rpart)
library(mlbench)
library(adabag)
data(Glass)
n <- dim(Glass)[1]
Glass_rand <- Glass[ order(runif(n)),]
#разделим выборку на 30% теста и 70% для обучения
nt <- as.integer(n*0.7)
Glass_train <- Glass_rand[1:nt, ]
Glass_test <- Glass_rand[(nt+1):n, ]
#Зададим максимальную глубину каждого дерева:
maxdepth <- 5
#Зададим максимальное число итераций:
mfinal = c()
mfinal <- append(mfinal, 1)
for(i in 1:20){
  mfinal <- append(mfinal, 1+10*i)
}

err = c()
for(i in 1:length(mfinal)){
  Glass.bagging <- bagging(Type ~., data=Glass_train, mfinal=mfinal[i],
                           maxdepth=maxdepth)
  Glass.bagging.pred <- predict.bagging(Glass.bagging, Glass_test)
  err <- append(err, Glass.bagging.pred$error)
}
plot(mfinal, err, xlab="Число деревьев", ylab="Ошибка", pch = 19)
err
```

Приложение 3. Код для задания 3

```
#ЗАДАНИЕ 3
library(kknn)
library(rpart)
library(mlbench)
library(adabag)
#Vehicle
data(Vehicle)
n <- dim(Vehicle)[1]
Vehicle_rand <- Vehicle[ order(runif(n)),]
#разбиваем на обучающую и тестовую выборку
nt <- as.integer(n*0.7)
Vehicle_train <- Vehicle_rand[1:nt, ]
Vehicle_test <- Vehicle_rand[(nt+1):n, ]
#knn
Vehicle_classifier <- kknn(Class~., Vehicle_train, Vehicle_test, distance = 2, k = 5)
fit <- fitted(Vehicle_classifier)
tb <- table(fit, Vehicle_test$Class)
error.kknn <- 1-(sum(diag(tb))/sum(tb))
#использование единичного дерева
maxdepth <- 5
Vehicle.rpart <- rpart(Class ~., data=Vehicle_train, maxdepth=maxdepth)
Vehicle.rpart.pred <- predict(Vehicle.rpart, Vehicle_test, type="class")
tb <- table(Vehicle.rpart.pred, Vehicle_test$Class)
error.rpart <- 1-(sum(diag(tb))/sum(tb))

error.kknn
error.rpart

#GLASS
data(Glass)
n <- dim(Glass)[1]
Glass_rand <- Glass[ order(runif(n)),]
#делим на обучающую и тестовую
nt <- as.integer(n*0.7)
Glass_train <- Glass_rand[1:nt, ]
Glass_test <- Glass_rand[(nt+1):n, ]
#knn
Glass_classifier <- kknn(Type~., Glass_train, Glass_test, distance = 2, k = 7)
fit <- fitted(Glass_classifier)
tb <- table(fit, Glass_test$Type)
error.kknn <- 1-(sum(diag(tb))/sum(tb))
#использование единичного дерева
Glass.rpart <- rpart(Type ~., data=Glass_train, maxdepth=maxdepth)
Glass.rpart.pred <- predict(Glass.rpart, Glass_test, type="class")
tb <- table(Glass.rpart.pred, Glass_test$Type)
error.rpart <- 1-(sum(diag(tb))/sum(tb))

error.kknn
error.rpart
```

Приложение 4. Бустинг алгоритм с k ближайших соседей

```
#ЗАДАНИЕ 3 (Бустинг с knn)
library(dplyr)

#Анализ тестовых ошибок для Glass и Vehicle с использованием бустинга с knn для k = 5
Glass_boosting <- knn_boosting('Type', Glass_train, k = 5, mfinal = 4)
Glass_pred <- boosting_pred(Glass_boosting, Glass_test)
tab1 <- table(Glass_test$Type, Glass_pred)
#ошибка классификации для Glass
1 - sum(diag(tab1)) / sum(tab1)

Vehicle_boosting <- knn_boosting('Class', Vehicle_train, k = 5, mfinal = 4)
Vehicle_pred <- boosting_pred(Vehicle_boosting, Vehicle_test)
tab2 <- table(Vehicle_test$Class, Vehicle_pred)
#ошибка классификации для Vehicle
1 - sum(diag(tab2)) / sum(tab2)

knn_boosting <- function(target, data, k, mfinal) {
  #число строк в массиве данных
  n <- nrow(data)
  #инициализация одинаковых весов
  w <- rep(1/n, each = n)

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

  for (t in 1:mfinal) {
    #обучение слабого классификатора
    clfier <- knn_w(target, train = data, k = k, w)
    knn_predicted <- knn_w_predicted(clfier, data)
    #ошибки слабого классификатора
    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
    #вес слабого классификатора
    alphas[[t]] <- log((1 - sum(error)) / sum(error)) / 2

    #модификация весов для следующей итерации (большие для неправильно
определенных)
```

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

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

knn_w_predicted <- 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))
}

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$classifiers)) {
      prediction <- knn_w_predicted(clfier$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)
}
```