Лабораторная работа № 10: «Градиентный бустинг»

Для выполнения задания используйте набор данных boston из библиотеки sklearn

https://scikit-learn.org/stable/datasets/index.html#boston-dataset

Задание.

1. Загрузите данные с помощью библиотеки sklearn.
2. Разделите выборку на обучающую (75%) и контрольную (25%).
3.
4. Заведите массив для объектов DecisionTreeRegressor (они будут использоваться в качестве базовых алгоритмов) и для вещественных чисел (коэффициенты перед базовыми алгоритмами).
5. В цикле обучите последовательно 50 решающих деревьев с параметрами max_depth=5 и random_state=42 (остальные параметры - по умолчанию). Каждое дерево должно обучаться на одном и том же множестве объектов, но ответы, которые учится прогнозировать дерево, будут меняться в соответствие с отклонением истинных значений от предсказанных.
6. Попробуйте всегда брать коэффициент равным 0.9. Обычно оправдано выбирать коэффициент значительно меньшим - порядка 0.05 или 0.1, но на стандартном наборе данных будет всего 50 деревьев, возьмите для начала шаг побольше.
7. В процессе реализации обучения вам потребуется функция, которая будет вычислять прогноз построенной на данный момент композиции деревьев на выборке X. Реализуйте ее. Эта же функция поможет вам получить прогноз на контрольной выборке и оценить качество работы вашего алгоритма с помощью mean_squared_error в sklearn.metrics.
8. Попробуйте уменьшать вес перед каждым алгоритмом с каждой следующей итерацией по формуле 0.9 / (1.0 + i), где i - номер итерации (от 0 до 49). Какое получилось качество на контрольной выборке?
9. Исследуйте, переобучается ли градиентный бустинг с ростом числа итераций, а также с ростом глубины деревьев. Постройте графики. Какие выводы можно сделать?
10. Сравните качество, получаемое с помощью градиентного бустинга с качеством работы линейной регрессии. Для этого обучите LinearRegression из sklearn.linear_model (с параметрами по умолчанию) на обучающей выборке и оцените для прогнозов полученного алгоритма на тестовой выборке RMSE.
11. Ответы на вопросы представьте в виде отчета.

Реализация:

In[1]:

```
import copy
import os

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
from scipy import misc
from datetime import datetime
import pandas as pd
from mpl_toolkits.mplot3d import axes3d
import scipy.optimize
from scipy import stats
from sklearn.tree import *
from sklearn.ensemble import *


import math
from sklearn.model_selection import train_test_split
```

1.1 Загрузите данные с помощью библиотеки sklearn

In[2]:

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
X = boston_dataset["data"]
y = boston_dataset["target"]
```

1.2 Разделите выборку на обучающую (75%) и контрольную

In[3]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.75, test_s
ize=0.25, random_state=42)
```

1.4 В процессе реализации обучения вам потребуется функция, которая будет вычислять прогноз построенной на данный момент композиции деревьев на выборке X. Реализуйте ее. Эта же функция поможет вам получить прогноз на контрольной выборке и оценить качество работы вашего алгоритма с помощью mean_squared_error в sklearn.metrics

In[4]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

def grad(z, y):
    return y - z

def gbm_predict(X, base_algorithms_list, coefficients_list):
    return [sum([coeff * algo.predict([x])[0] for algo, coeff in zip(base_algo-
rithms_list, coefficients_list)]) for x in X]

def gbm_train(X_train, y_train, max_depth=5, iters=50, coef=0.9):
    algos, coefs = [], []

    target = y_train
    for i in range(iters):
        tree = DecisionTreeRegressor(max_depth=max_depth, random_state=42)
        tree.fit(X_train, target)
        algos.append(tree)

        if isinstance(coef, (int, float)):
            coefs.append(coef)
        else:
            coefs.append(coef(i))

        target = grad(gbm_predict(X_train, algos, coefs), y_train)

    return algos, coefs
```

1.5 В цикле обучите последовательно 50 решающих деревьев с параметрами max_depth=5 и random_state=42 (остальные параметры - по умолчанию). Каждое дерево должно обучаться на одном и том же множестве объектов, но ответы, которые учится прогнозировать дерево, будут меняться в соответствие с отклонением истинных значений от предсказанных

In[5]:

```
algos, coefs = gbm_train(X_train, y_train)
mean_squared_error(y_test, gbm_predict(X_test, algos, coefs))
```

```
13.577188945603083
```

1.6 Попробуйте всегда брать коэффициент равным 0.9. Обычно оправдано выбирать коэффициент значительно меньшим - порядка 0.05 или 0.1, но на стандартном наборе данных будет всего 50 деревьев, возьмите для начала шаг побольше

In[6]:

```
algos, coefs = gbm_train(X_train, y_train)
mean_squared_error(y_test, gbm_predict(X_test, algos, coefs))
```

Out[6]:

```
13.577188945603083
```