Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем

Кафедра Информатики

ЛАБОРАТОРНАЯ РАБОТА №10
«Градиентный бустинг»

БГУИР 1-40 81 04

Магистрант:
гр. 858642
Кукареко А.В.

Проверил:
Стержанов М. В.

Минск, 2019

ХОД РАБОТЫ

Задание.

Для выполнения задания используйте набор данных boston из библиотеки sklearn - https://scikit-learn.org/stable/datasets/index.html#boston-datase.

1. Загрузите данные с помощью библиотеки sklearn.
2. Разделите выборку на обучающую (75%) и контрольную (25%).
3. Заведите массив для объектов DecisionTreeRegressor (они будут использоваться в качестве базовых алгоритмов) и для вещественных чисел (коэффициенты перед базовыми алгоритмами).
4. 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.
5. 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.
6. 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.
7. 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?
8. Исследуйте, переобучается ли градиентный бустинг с ростом числа итераций, а также с ростом глубины деревьев. Постройте графики. Какие выводы можно сделать?
9. 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.

10. Ответы на вопросы представьте в виде отчета.

Результат выполнения:

1. Загрузите данные с помощью библиотеки sklearn.

```
data = datasets.load_boston()
X = data['data']
y = data['target']

X.shape, y.shape
```

```
((506, 13), (506,))
```

```
data.keys()
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

2. Разделите выборку на обучающую (75%) и контрольную (25%).

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=10)
```

```
X_train.shape, X_test.shape
```

```
((379, 13), (127, 13))
```

3. Заведите массив для объектов DecisionTreeRegressor (они будут использоваться в качестве базовых алгоритмов) и для вещественных чисел (коэффициенты перед базовыми алгоритмами).

```
trees_1 = []
coeffs_1 = []
```

4. 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.

```
iter = 50
last_y_pred_1 = y_train

for i in range(iter):
    model = tree.DecisionTreeRegressor(max_depth=5, random_state=42)
    model.fit(X_train, last_y_pred_1)
    coef = 0.9
    trees_1.append(model)
    coeffs_1.append(coef)
    last_y_pred_1 = y_train - gbm_predict(X_train, trees_1, coeffs_1)
```

5. 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.

Реализовано в пункте № 4.

6. 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.

Функция вычисления прогноза композиции:

```
def gb_predict(X, trees, coeffs):
    m,n = X.shape
    iterations = len(trees)
    result = np.zeros(m)

    for i in range(iterations):
        result += coeffs[i] * trees[i].predict(X)

    return result
```

Функция оценки качества работы:

```
def rmse(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))
```

7. 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?.

```
trees_2 = []
coeffs_2 = []

iter_2 = 50
last_y_pred_2 = y_train

for i in range(iter_2):
    model = tree.DecisionTreeRegressor(max_depth=5, random_state=42)
    model.fit(X_train, last_y_pred_2)
    coef = 0.9 / (1.0 + i)
    trees_2.append(model)
    coeffs_2.append(coef)
    last_y_pred_2 = y_train - gb_predict(X_train, trees_2, coeffs_2)

y_test_pred_2 = gbm_predict(X_test, trees_2, coeffs_2)
print(f'RMSE: {rmse(y_test, y_test_pred_2)}')
```

При коэффициенте «0.9», метрика RMSE показала значение: «4.0434». После применения уменьшающегося коэффициента, метрика RMSE упала до «3.7670» на тестовой выборке. Это означает, что качество чуть-чуть улучшилось.

8. Исследуйте, переобучается ли градиентный бустинг с ростом числа итераций, а также с ростом глубины деревьев. Постройте графики. Какие выводы можно сделать?

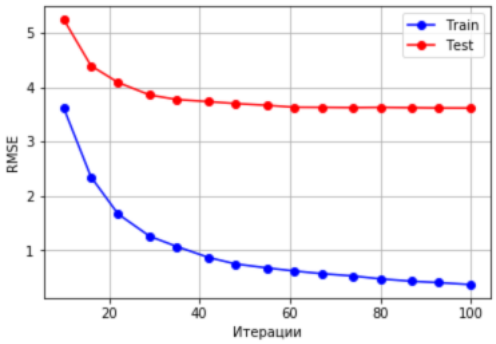

Рисунок 1 – график зависимости ошибки от числа итераций.

Если посмотреть на рисунок 1, то видно, что ошибка на Test и на Training довольна мала и она потихоньку уменьшается и выходит на плато, что свидетельствует об отсутствии переобучения.

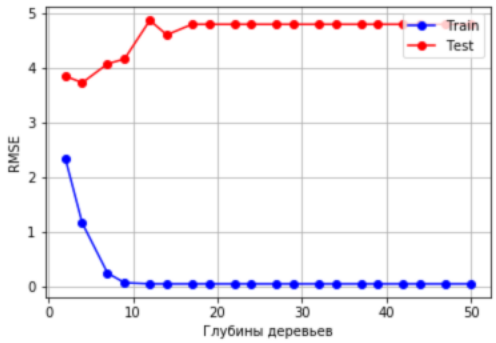

Рисунок 2 – график зависимости ошибки от глубины деревьев.

Если посмотреть на рисунок 2, то видно, что ошибка на Training падает, с увеличением числа деревьев, а вот ошибка на Test возрастает до определенного кол-ва глубины(примерно до ~15), а потом выходит на плато. Это свидетельствует о переобучении модели.

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.

```
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred_linear = linear_model.predict(X_test)
rmse(y_test, y_pred_linear)
```

```
5.6958350306172365
```

Ошибка RMSE линейной регрессии получилась «5.6958», что больше чем «3.7670» у модели с «градиентым бустингом». Из этого можно сделать

вывод, что без дополнительных настроек, «градиентный бустинг» способен выдавать результат более лучшего качества, чем линейная регрессия.

Вывод.

В ходе выполнения лабораторной работы я ознакомился алгоритмами «дерево принятия решений» и «градиентный бустинг». Так же я реализовал «градиентный бустинг», проверил его работоспособность на наборе данных «boston-dataset» и сравнил его эффективность с «линейной регрессией».