Лабораторная работа 3. Композиции алгоритмов

Цель работы: научиться работать со случайным лесом, подбирать его параметры и решать с его помощью задачи регрессии; научиться работать с градиентным бустингом и подбирать его гиперпараметры, а также сравнивать разные способы построения композиций, научиться использовать метрику log-loss.

1 Размер случайного леса

Случайный лес – это модель классификации, объединяющая некоторое количество решающих деревьев в одну композицию, за счет чего улучшается их качество работы и обобщающая способность. Деревья строятся независимо друг от друга. Чтобы они отличались друг от друга, обучение проводится не на всей обучающей выборке, а на ее случайном подмножестве. Также, для дальнейшего уменьшения схожести деревьев, оптимальный признак для разбиения выбирается не из всех возможных признаков, а лишь из их случайного подмножества. Прогнозы, выданные деревьями, объединяются в один ответ путем усреднения.

Особенность случайного леса заключается в том, что он не переобучается по мере увеличения количества деревьев в композиции. Это достигается за счет того, что деревья не зависят друг от друга, и поэтому добавление нового дерева в композицию не усложняет модель, а лишь понижает уровень шума в прогнозах.

В библиотеке scikit-learn случайные леса реализованы в классах sklearn.ensemble.RandomForestClassifier (для классификации) и sklearn.ensemble.RandomForestRegressor (для регрессии). Обучение модели производится с помощью функции fit, построение прогнозов – с помощью функции predict. Число деревьев задается с помощью поля класса n_estimators.

Пример использования:

```
import numpy as np
from sklearn.ensemble import RandomForestRegressor
X = np.array ([[1, 2], [3, 4], [5, 6]])
y = np.array ([-3, 1, 10])
clf = RandomForestRegressor (n_estimators=100)
clf.fit (X, y)
predictions = clf.predict (X)
```

Также в этом задании понадобится вычислять качество предсказаний на тестовой выборке. Для того будет использоваться метрика R2 – по сути, это среднеквадратичная ошибка (MSE), нормированная на отрезок [0, 1] и обращенная так, чтобы ее наилучшим значением была единица. Её можно вычислить с помощью функции sklearn.metrics.r2_score. Первым аргументом является список правильных ответов на выборке, вторым – список предсказанных ответов. Пример использования приведён ниже:

```
from sklearn.metrics import r2_score
print r2_score ([10, 11, 12], [9, 11, 12.1])
```

В настоящем задании необходимо проследить за изменением качества случайного леса в зависимости от количества деревьев в нем. Для выполнения задания необходимо:

1) Загрузить данные из файла abalone.csv. Это датасет, в котором требуется предсказать возраст ракушки (число колец) по физическим измерениям.
2) Преобразовать признак Sex в числовой: значение F должно перейти в -1, I – в 0, M – в 1. Если вы используете Pandas, то подойдет следующий код:

```
data['Sex'] = data['Sex'].map(lambda x: 1 if x == 'M' else (-1 if x
== 'F' else 0))
```

3) Разделить содержимое файлов на признаки и целевую переменную. В последнем столбце записана целевая переменная, в остальных – признаки.
4) Обучить случайный лес (sklearn.ensemble.RandomForestRegressor) с различным числом деревьев: от 1 до 50 (random_state=1). Для каждого из вариантов оценить качество работы полученного леса на кросс-валидации по 5 блокам. Использовать параметры "random_state=1"и "shuffle=True" при создании генератора кросс-валидации sklearn.cross_validation.KFold. В качестве меры качества воспользоваться коэффициентом детерминации (sklearn.metrics.r2_score).
5) Определить, при каком минимальном количестве деревьев случайный лес показывает качество на кросс-валидации выше 0.52. Это количество и будет ответом на задание.
6) Обратить внимание на изменение качества по мере роста числа деревьев. Ухудшается ли оно?

2 Градиентный бустинг над решающими деревьями

Построение композиции – важный подход в машинном обучении, который позволяет объединять большое количество слабых алгоритмов в один сильный. Данный подход широко используется на практике в самых разных задачах.

Градиентный бустинг последовательно строит композицию алгоритмов, причем каждый следующий алгоритм выбирается так, чтобы исправлять ошибки уже имеющейся композиции. Обычно в качестве базовых алгоритмов используют деревья небольшой глубины, поскольку их достаточно легко строить, и при этом они дают нелинейные разделяющие поверхности.

Другой метод построения композиций – случайный лес. В нем, в отличие от градиентного бустинга, отдельные деревья строятся независимо и без каких-либо ограничений на глубину – дерево наращивается до тех пор, пока не покажет наилучшее качество на обучающей выборке.

В настоящем задании мы будет использоваться задача классификации. В качестве функции потерь будем использовать log-loss:

$$L(y; z) = -y \log z - (1 - y) \log 1 - z.$$

Здесь через y обозначен истинный ответ, через z – прогноз алгоритма. Данная функция является дифференцируемой, и поэтому подходит для использования в градиентном бустинге. Также можно показать, что при её использовании итоговый алгоритм будет приближать истинные вероятности классов.

В пакете scikit-learn градиентный бустинг реализован в модуле ensemble в виде классов GradientBoostingClassifier и GradientBoostingRegressor. Основные параметры, которые будут нас интересовать: n_estimators, learning_rate. Иногда может быть полезен параметр verbose для отслеживания процесса обучения.

Чтобы была возможность оценить качество построенной композиции на каждой итерации, у класса есть метод staged_decision_function. Для заданной выборки он возвращает ответ на каждой итерации.

Помимо алгоритмов машинного обучения, в пакете scikit-learn представлено большое число различных инструментов. В настоящем задании будет предложено воспользоваться функцией train_test_split модуля cross_validation. С помощью неё можно разбивать выборки случайным образом. На вход можно передать несколько выборок (с условием, что они имеют одинаковое количество строк). Пусть, например, имеются данные X и y, где X – это признаковое описание объектов, y – целевое значение. Тогда следующий код будет удобен для разбиения этих данных на обучающее и тестовое множества:

```
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size =
0.33, random_state = 42)
```

Обратите внимание, что при фиксированном параметре random_state результат разбиения можно воспроизвести.

Метрика log-loss реализована в пакете metrics. Данная метрика предназначена для классификаторов, выдающих оценку принадлежности классу, а не бинарные ответы. И градиентный бустинг, и случайный лес умеют строить такие прогнозы – для этого нужно использовать метод predict_proba:

```
pred = clf.predict_proba (X_test)
```

Метод predict_proba возвращает матрицу, i-й столбец которой содержит оценки принадлежности i-му классу.

Для рисования кривых качества на обучении и контроле можно воспользоваться следующим кодом:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure ()
plt.plot (test_loss, 'r', linewidth=2)
plt.plot (train_loss, 'g', linewidth=2)
plt.legend (['test', 'train'])
```

В рамках данного задания рассмотривается датасет с конкурса Predicting a Biological Response.

Для выполнения задания необходимо:

1) Загрузить выборку из файла gbm-data.csv с помощью pandas и преобразовать её в массив numpy (параметр values у датафрейма). В первой колонке файла с данными записано, была или нет реакция. Все остальные колонки (d1 - d1776) содержат различные характеристики молекулы, такие как размер, форма и т.д. Разбейте выборку на обучающую и тестовую, используя функцию train_test_split с параметрами test_size = 0.8 и random_state = 241.
2) Обучить GradientBoostingClassifier с параметрами n_estimators=250, verbose=True, random_state=241 и для каждого значения learning_rate из списка [1, 0.5, 0.3, 0.2, 0.1] проделать следующее: использовать метод staged_decision_function для предсказания качества на обучающей и тестовой выборке на каждой итерации; преобразовать полученное предсказание по формуле $\frac{1}{1+e^{-y\_pred}}$, где y_pred – предсказанное значение; вычислить и построить график значений log-loss на обучающей и тестовой выборках, а также найти минимальное значение метрики и номер итерации, на которой оно достигается.

3) Как можно охарактеризовать график качества на тестовой выборке, начиная с некоторой итерации: переобучение (overfitting) или недообучение (underfitting)? В ответе указать одно из слов (overfitting или underfitting).

4) Привести минимальное значение log-loss на тестовой выборке и номер итерации, на которой оно достигается, при learning_rate = 0.2.

5) На этих же данных обучить RandomForestClassifier с количеством деревьев, равным количеству итераций, на котором достигается наилучшее качество у градиентного бустинга из предыдущего пункта, random_state=241 и остальными параметрами по умолчанию. Какое значение log-loss на тесте получается у этого случайного леса? (Не забывайте, что предсказания нужно получать с помощью функции predict_proba. В данном случае брать сигмоиду от оценки вероятности класса не нужно.)

Обратите внимание, что, хотя в градиентного бустинге гораздо более слабые базовые алгоритмы, он выигрывает у случайного леса благодаря более "направленной" настройке – каждый следующий алгоритм исправляет ошибки имеющейся композиции. Также он обучается быстрее случайного леса благодаря использованию неглубоких деревьев. В то же время, случайный лес может показать более высокое качество при неограниченных ресурсах – так, он выиграет у градиентного бустинга на наших данных, если увеличить число деревьев до нескольких сотен (проверьте самостоятельно).