

Градиентный бустинг над решающими деревьями

Данное задание основано на материалах лекций по композициям алгоритмов.

Вы научитесь:

- работать с градиентным бустингом и подбирать его гиперпараметры
- сравнивать разные способы построения композиций
- понимать, в каком случае лучше использовать случайный лес, а в каком — градиентный бустинг
- использовать метрику $\log\text{-loss}$

Введение

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.

На лекциях был рассмотрен метод градиентного бустинга, который 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 этом задании мы будем иметь дело с задачей 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 ее использовании итоговый алгоритм будет приближать истинные вероятности классов.

Реализация в sklearn

В пакете `scikit-learn` градиентный бустинг реализован в модуле `ensemble` в виде классов `GradientBoostingClassifier` и `GradientBoostingRegressor`. Основные параметры, которые будут интересовать нас: `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 этом задании будет предложено воспользоваться функцией `train_test_split` модуля `cross_validation`. С помощью нее 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`. Заметим, что 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'])
```

Материалы

Подробнее о градиентном бустинге и особенностях его применения к деревьям: http://www.machinelearning.ru/wiki/images/7/7e/Sem03_ensembles_2014.pdf

Данные

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

Инструкция по выполнению

1. Загрузите выборку из файла `gbm-data.csv` с помощью `pandas` и преобразуйте ее 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. Обучите `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]` сделайте следующее:
 - Используйте метод `staged_decision_function` для предсказания качества на обучающей и тестовой выборке на каждой итерации.
 - Преобразуйте полученное предсказание по формуле $\frac{1}{1+e^{-y_pred}}$, где `y_pred` — предсказанное значение.
 - Вычислите и постройте график значений `log-loss` на обучающей и тестовой выборках, а также найдите 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 укажите одно из слов `overfitting` либо `underfitting`.
4. Приведите минимальное значение `log-loss` на тестовой выборке и номер итерации, на котором оно достигается, при `learning_rate = 0.2`.
5. На этих же данных обучите `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)

Если ответом является нецелое число, то целую и дробную часть необходимо разграничивать точкой, например, 0.42. При необходимости округляйте дробную часть до двух знаков.

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 выигрывает у градиентного бустинга на наших данных, если увеличить число деревьев до нескольких сотен (проверьте сами!).

Ответ на каждое задание — текстовый файл, содержащий ответ в первой строчке. Обратите внимание, что отправляемые файлы не должны содержать перенос строки в конце.