Работа № 3.7. Бустинг (AdaBoost, LogitBoost, BrownBoost)

Теоретические сведения

Бустинг — это семейство ансамблевых алгоритмов, суть которых заключается в создании сильного классификатора на основе нескольких слабых. Для этого сначала создаётся одна модель, затем другая модель, которая пытается исправить ошибки в первой. Модели добавляются до тех пор, пока тренировочные данные не будут идеально предсказываться или пока не будет превышено максимальное количество моделей.

AdaBoost был первым действительно успешным алгоритмом бустинга, разработанным для бинарной классификации. Именно с него лучше всего начинать знакомство с бустингом. Современные методы вроде стохастического градиентного бустинга основываются на AdaBoost.

AdaBoost используют вместе с короткими деревьями решений. После создания первого дерева проверяется его эффективность на каждом тренировочном объекте, чтобы понять, сколько внимания должно уделить следующее дерево всем объектам. Тем данным, которые сложно предсказать, даётся больший вес, а тем, которые легко предсказать, — меньший. Модели создаются последовательно одна за другой, и каждая из них обновляет веса для следующего дерева. После построения всех деревьев делаются предсказания для новых данных, и эффективность каждого дерева определяется тем, насколько точным оно было на тренировочных данных.

Так как в этом алгоритме большое внимание уделяется исправлению ошибок моделей, важно, чтобы в данных отсутствовали аномалии.

1. Набор данных Bioresponse

https://www.kaggle.com/c/bioresponse

- 3751объектов, 1777признаков.
- Объект представляет из себя характеристики некоторой молекулы
- Класс объекта – вызвала ли молекула реакцию или нет
- Классификация. Способ классификации - Boosting

Предобработка данных

Пропуски в данных? Нет

Набор данных был разделен на обучающую и тестовую выборку.

Обучающая выборка содержит 3 000 объектов, а тестовая выборка – 751.

Проблема

Можно ли из полученных классификаторов построить агрегированный классификатор с предсказательной точностью выше, чем у найденных классификаторов?

Boosting

Бустинг — это семейство ансамблевых алгоритмов, суть которых заключается в создании сильного классификатора на основе нескольких слабых. Для этого сначала создаётся одна модель, затем другая модель, которая пытается исправить ошибки в первой. Модели добавляются до тех пор, пока тренировочные данные не будут идеально предсказываться или пока не будет превышено максимальное количество моделей.

AdaBoost

AdaBoost был первым действительно успешным алгоритмом бустинга, разработанным для бинарной классификации. Именно с него лучше всего начинать знакомство с бустингом.

Современные методы вроде стохастического градиентного бустинга основываются на AdaBoost.

AdaBoost используют вместе с короткими деревьями решений. После создания первого дерева проверяется его эффективность на каждом тренировочном объекте, чтобы понять, сколько внимания должно уделить следующее дерево всем объектам. Тем данным, которые сложно предсказать, даётся больший вес, а тем, которые легко предсказать, — меньший. Модели создаются последовательно одна за другой, и каждая из них обновляет веса для следующего дерева. После построения всех деревьев делаются предсказания для новых данных, и эффективность каждого дерева определяется тем, насколько точным оно было на тренировочных данных.

Так как в этом алгоритме большое внимание уделяется исправлению ошибок моделей, важно, чтобы в данных отсутствовали аномалии.

Алгоритм AdaBoost строит «сильный» классификатор вида:

$$F_T(x) = sign(f_T(x)) = sign(\sum_{t=1}^{T} \beta_t h(x, a_t))$$

где$w_t \in \mathbb{R}$, $h(x, a_t): X \times A \to \{-1,1\}$ - «слабый» классификатор, принадлежащий некоторому семейству классификаторов H, а A – пространство параметров этого семейства.

Пример H - одноуровневые деревья принятия решений.

Для нахождения классификатора $F_T(x)$ алгоритм AdaBoost последовательно ищет оптимальные параметры β_t и a_t $(1 \le t \le T)$ для построения классификатора $F_t(x)$, используя найденный ранее классификатор $F_{t-1}(x)$:

$F_t(x) = sign(f_{t-1}(x) + \beta_t h(x, a_t)), 1 \le t \le T$

при условии $f_0(x) = 0$.

Шаг 1: Инициализируем веса объектов:

$$w_i = \frac{1}{n}, i = 1, \dots, n$$

Шаг 2: Последовательное построение классификаторов $F_t(x), 1 \le t \le T$

Для *t = 1,...,T:*

a) Обучить слабый классификатор $h(x, a_t) \in H$, используя в качества функции потерь:

$$L(a) = \sum_{i=1}^{n} w_i \, I[\![h(x_i, a) \neq y_i]\!]$$

где $I[\![h(x_i, a) \neq y_i]\!]$ - индикатор ошибки.

b) Подсчет β_t:

$$\beta_t = \frac{1}{2} ln \frac{1 - L(a_t)}{L(a_t)}$$

c) Обновление весов объектов:

$$w_i = \frac{w_i e^{-\beta_t y_i h(x_i, a_t)}}{Z_t}, i = 1, \dots, n$$

где $Z_t -$ нормировочный множитель.

Шаг 3: Построение итогового классификатора:

$$F_T(x) = sign(\sum_{t=1}^{T} \beta_t h(x, a_t))$$

AdaBoost (Scikit-learn)

```
classifier =
ensemble.AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),n_estimato
rs=10)
```

```
#обучениеклассификатора
classifier.fit(train[:,0:n],train[:,n])
#предсказание
prediction = classifier.predict(test[:,0:n])
tab = pd.crosstab(index = prediction, columns= test[:,n])

print(tab)
```

LogitBoost

- Основное отличие LogitBoost от AdaBoost состоит в том, что AdaBoost использует экспоненциальную функцию потерь, а LogitBoost – логистическую.
- За счет этого, в некоторых случаях LogitBoost может превосходить по точности AdaBoost, а также быть более устойчивым к шумам в данных.

$$y_i \in \{0,1\}$$

Шаг 1: Инициализируем переменные и вероятность того, что объект относится к классу 1:

$$f_T(x) = 0$$

$$w_i = \frac{1}{n} \text{ и } p(x_i) = \frac{1}{2}, i = 1, \dots, n$$

Шаг 2: Последовательное построение классификаторов $F_t(x), 1 \le t \le T$

Для *t = 1,...,T:*

a) Расчет w_i и z_i:

$$w_i = p(x_i)(1 - p(x_i))$$

$$z_i = \frac{y_i - p(x_i)}{p(x_i)(1 - p(x_i))}$$

b) Обучить $h(x, a_t) \in H$, используя в качества функции потерь:

$$L(a) = \sum_{i=1}^{n} w_i \, (h(x_i, a) - z_i)^2$$

c) Обновление $f_T(x)$ и $p(x)$:

$$f_T(x) = f_T(x) + \frac{1}{2} h(x, a_t) p(x) = (e^{f_T(x)})/(e^{f_T(x)} + e^{-f_T(x)})$$

Шаг 3: Построение итогового классификатора:

$$F_T(x) = \begin{cases} 1, \text{ если } sign(f_T(x)) \ge 0 \\ 0, \text{если } sign(f_T(x)) < 0 \end{cases}$$

BrownBoost

- В алгоритме BrownBoost используется дополнительная переменная – «время» работы алгоритма.
- Алгоритм BrownBoost более устойчив к шумам в данных (McDonald R. A. et al. An empirical comparison of three boosting algorithms on real data sets with artificial class noise).

$$y_i \in \{-1,1\}$$

Шаг 1: Подсчитываем «время» работы алгоритма c:

$$c = erfinv^2(1 - \varepsilon)$$

где ε – заданная точность классификации для функции потерь $\frac{1}{n}\sum_{i=1}^{n} |F_T(x_i) - y_i|$,

$erfinv$ – обратная функция к функции $erf(z) = \frac{2}{\pi}\int_0^z e^{-x^2} dx$

Шаг 2: Инициализируем «оставшиеся время» работы алгоритма s_1 и предсказанное значение $r_1(i)$ для объекта i:

$s_1 = c$ и $r_1(i) = 0, i = 1, \dots, n$

Шаг 3: Последовательное построение классификаторов $F_k(x)$

Для *k = 1,2,...* пока $s_k > 0$

a) Задание весов объектов:

$$w_i = \frac{e^{-(r_k(i)+s_k)^2/c}}{Z_k}, i = 1, \dots, n$$

где $Z_k -$ нормировочный множитель.

b) Нахождение слабого классификатора $h(x, a_k) \in H$ такого, что $\sum_{i=1}^{n} w_i h(x_i, a_k) y_i > 0$.

c) Для нахождения $t_k = t^* > 0$, $\beta_k = \beta^*$ таких, что $\gamma^* \leq \nu$ (ν – малая заданная константа, используемая для исключения вырожденных случаев) или $t^* = s_k$ решить дифференциальное уравнение с вещественными переменными γ, β, t:

$$\frac{dt}{d\beta} = \gamma = \frac{\sum_{i=1}^{n} \exp(-\frac{1}{c}(r_k(i) + \beta h(x_i, a_k) y_i + s_k - t)^2)\, h(x_i, a_k) y_i}{\sum_{i=1}^{n} \exp(-\frac{1}{c}(r_k(i) + \beta h(x_i, a_k) y_i + s_k - t)^2)}$$

с краевыми условиями $t = 0, \beta = 0$.

d) Задание s_{k+1} и $r_{k+1}(i)$:

$$r_{k+1}(i) = r_k(i) \beta_k h(x_i, a_k) y_i,\ i = 1, \dots, n$$

$$s_{k+1} = s_k - t_k$$

Шаг 4: Построение итогового классификатора:

$$F_T(x) = sign(\sum_{t=1}^{k-1} \beta_t h(x, a_t))$$

Практическое задание

1. Постройте график зависимости точности классификации и времени работы AdaBoost, LogitBoost, BrownBoost от максимального количества итераций.
2. Постройте график зависимости точности классификации и времени работы BrownBoost от значений параметров ε и ν.
3. Выполните предобработку данных:
   а) анализ и удаление выбросов
   б) снижение размерности
4. Оцените точность классификаторов, найденных AdaBoost, LogitBoost, BrownBoost.