

Решающие деревья. Random Forest. Бустинг.

Санкт-Петербургский государственный университет
Кафедра статистического моделирования

Санкт-Петербург, 2025

Решающее дерево — это бинарное дерево, где

- во всех внутренних вершинах v задан некоторый предикат $\beta_v : \mathbf{X} \rightarrow \{0,1\}$,
- в каждой листовой вершине v задан прогноз $c_v \in \mathbf{Y}$ (для классификации возможно: $c_v \in \mathbb{R}_+^K$, $\sum_{k=1}^K c_{v_k} = 1$ — вектор вероятностей принадлежности к классу).

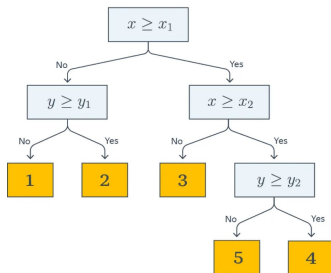


Рис.: Решающее бинарное дерево

Решающие деревья можно применять как для задач регрессии, так и для задач классификации.

Пусть \mathbf{X} — множество объектов, \mathbf{Y} — множество ответов, $y : \mathbf{X} \rightarrow \mathbf{Y}$ — некоторая зависимость.

Есть обучающая выборка $\mathbf{D} = \{(x_i, y_i)\}_{i=1}^n$, где x_i — входные данные, y_i — известные ответы.

- $y_i \in \{1, \dots, K\} \Rightarrow$ задача классификации.
- $y_i \in \mathbb{R} \Rightarrow$ задача регрессии.

Решающие деревья в задаче регрессии

Пусть $\mathbf{X} \in \mathbb{R}^{n \times p}$ — матрица данных (где p — признаки, n — наблюдения), $\mathbf{Y} \in \mathbb{R}^n$ — отклик.

Идея: разбиение совокупности всех возможных значений $\mathbf{X}_1, \dots, \mathbf{X}_p$ на J непересекающихся областей R_1, \dots, R_J .

Предсказание для объекта x :

$$f(x) = \sum_{j=1}^J c_j \mathbf{1}(x \in R_j).$$

Многомерные прямоугольники R_1, \dots, R_J выбираем так, чтобы минимизировать сумму квадратов остатков:

$$RSS = \sum_{j=1}^J \sum_{i \in \mathbf{R}_j} (y_i - f(x_i))^2 \rightarrow \min_{R_1, \dots, R_J}.$$

Тогда

$$\hat{c}_j = \frac{1}{|R_j|} \sum_{x_i \in R_j} y_i.$$

Решающие деревья в задаче регрессии

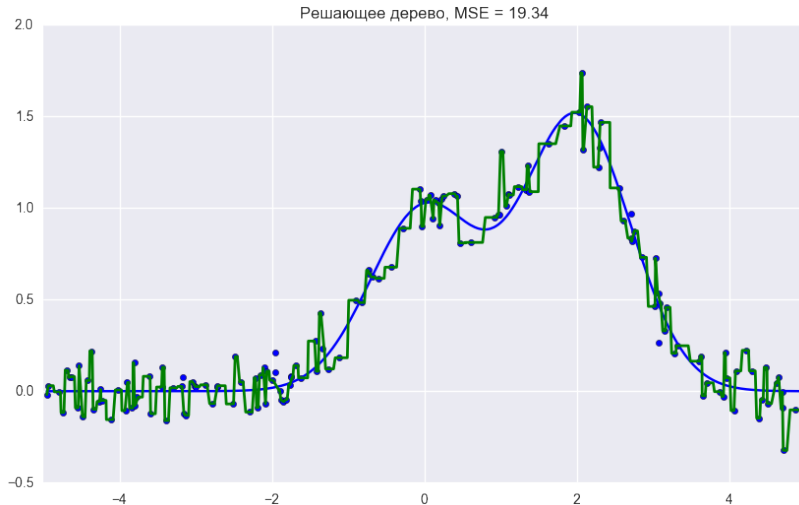


Рис.: Плохо обобщает закономерность и более склонна к переобучению

Решающие деревья в задаче классификации

В задаче классификации R_1, \dots, R_J минимизируется число ошибок классификации

$$M(j) = 1 - \max_k (\hat{p}_{jk}),$$

где \hat{p}_{jk} — доля объектов обучающей выборки класса k попавших в R_j .

На практике чаще используют две других метрики:

- $G(j) = \sum_{k=1}^K \hat{p}_{jk}(1 - \hat{p}_{jk})$ — индекс Джини,
- $CI(j) = - \sum_{k=1}^K \hat{p}_{jk} \log \hat{p}_{jk}$ — коэффициент перекрестной энтропии.

Предсказание для объекта x :

$$f(x) = \operatorname{argmax}_{k \in \mathbf{Y}} \hat{p}_{jk}.$$

Решающие деревья в задаче классификации

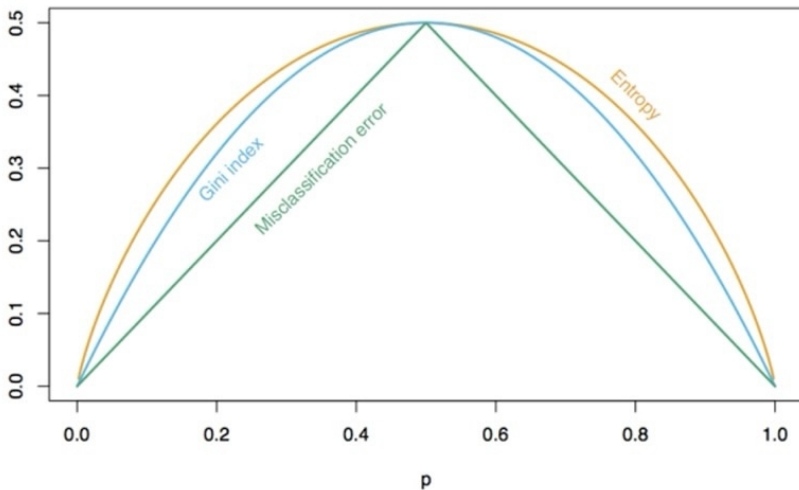


Рис.: Информационные индексы M , G , CI в случае двух классов

Решающие деревья в задаче классификации

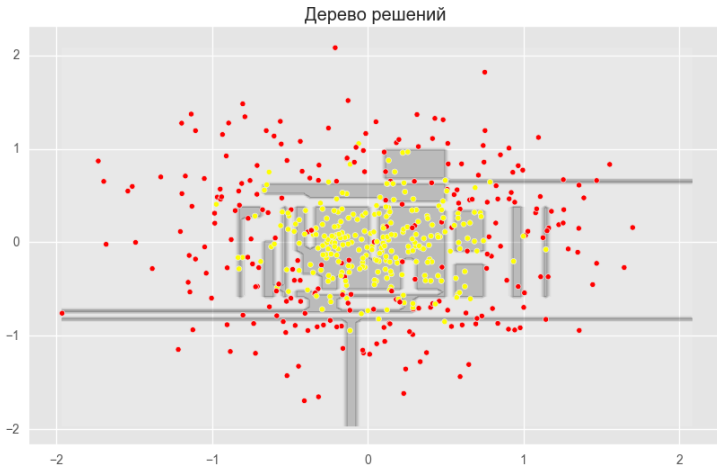


Рис.: Разделяющая граница дерева решений очень рваная и на ней много острых углов, что говорит о переобучении и слабой обобщающей способности

Жадный алгоритм построения решающего дерева

Нахождение оптимального дерева — это NP-полная задача.

Пусть X — исходное множество обучающей выборки, а X_m — множество объектов, попавших в текущий лист (в самом начале $X_m = X$). Тогда жадный алгоритм можно верхнеуровнево описать следующим образом:

- 1 Создаём вершину v ;
- 2 Если выполнен критерий остановки $\text{Stop}(X_m)$, то останавливаемся, объявляем эту вершину листом и ставим ей в соответствие ответ $\text{Ans}(X_m)$, после чего возвращаем её;
- 3 Иначе: находим предикат $B_{j,t}$ (сплит), который определит наилучшее разбиение текущего множества объектов X_m на две подвыборки X_l и X_r , максимизируя критерий ветвления $\text{Branch}(X_m, j, t)$;
- 4 Для X_l и X_r рекурсивно повторим процедуру.

На выходе получаем дерево, в каждом из листов которого содержится по крайней мере один объект исходной выборки.

$\text{Ans}(X_m)$ вычисляет ответ для листа по попавшим в него объектам из обучающей выборки. Может быть:

- ❶ в случае задачи классификации — меткой самого частого класса или оценкой дискретного распределения вероятностей классов для объектов, попавших в этот лист;
- ❷ в случае задачи регрессии — средним, медианой или другой статистикой;
- ❸ простой моделью. К примеру, листы в дереве, задающем регрессию, могут быть линейными функциями или синусоидами, обученными на данных, попавших в лист.

Критерий остановки $\text{Stop}(X_m)$ — функция, которая решает, нужно ли продолжать ветвление или пора остановиться.

- Ограничение максимальной глубины дерева.
- Ограничение минимального числа объектов в листе.
- Ограничение максимального количества листьев в дереве.
- Остановка в случае, если все объекты в листе относятся к одному классу.
- Остановка в случае, если изменение метрики меньше порога.

Замечание

Для очень глубоких деревьев имеем переобучение.

$\text{Branch}(X_m, \text{feature}, \text{value})$ — функция, измеряющая, насколько улучшится финальная метрика дерева при предлагаемом сплите.

Пусть $L(y_i, c)$ — функция потерь, c — константное предсказание. Информативность узла (хотим минимизировать):

$$H(X_m) = \frac{1}{|X_m|} \sum_{(x_i, y_i) \in X_m} L(y_i, c).$$

Если разделить на два узла:

$$\begin{aligned} & \frac{1}{|X_m|} \left(\sum_{(x_i, y_i) \in X_l} L(y_i, c_l) + \sum_{(x_i, y_i) \in X_r} L(y_i, c) \right) = \\ & = \frac{|X_l|}{|X_m|} H(X_l) + \frac{|X_r|}{|X_m|} H(X_r). \end{aligned}$$

Тогда критерий ветвления:

$$\text{Branch}(X_m, j, t) = |X_m| \cdot H(X_m) - |X_l| \cdot H(X_l) - |X_r| \cdot H(X_r).$$

Чем больше изменение метрики, тем лучше.

- MSE: $L(y_i, c) = (y_i - c)^2$, $H(X_m) = \sum_{(x_i, y_i) \in X_m} \frac{(y_i - \bar{y})^2}{|X_m|}$;
- MAE: $L(y_i, c) = |y_i - c|$,
 $H(X_m) = \sum_{(x_i, y_i) \in X_m} \frac{|y_i - \text{MEDIAN}(Y_m)|}{|X_m|}$;
- Для классификации можно рассматривать: misclassification error ($H(X_m) = 1 - \max p_i$, p_i — доля класса i), энтропию ($H(X_m) = -\sum_{k=1}^K p_k \log p_k$) или критерий Джини ($H(X_m) = \sum_{k=1}^K p_k (1 - p_k)$).

- **Проблема:** переобучение — небольшое смещение, но большая дисперсия.
- **Решение:** объединяя некоторые R_j можем уменьшить дисперсию за счет небольшого увеличения смещения.
- Выращиваем дерево только до тех пор, пока уменьшение RSS из-за разбиения превышает некоторый (высокий) порог.
- Однако можно пропустить хорошее разбиение, остановившись слишком рано. Поэтому можно выращивать большие деревья T_0 , а затем обрезать его для получения поддерева.

- Получим большое дерево T_0 и обрежем его в узле t , получив поддерево $T^t \subset T_0$.
- Рассмотрим последовательность деревьев проиндексированных положительным параметром α . Каждому α соответствует поддерево $T \subset T_0$, минимизирующее критерий

$$Q_\alpha(T) = Q(T) + \alpha |l(T)|,$$

где $Q(T)$ — разница между прогнозируемым и фактическим выходом модели на этапе её обучения, $\alpha \geq 0$, $|l(T)|$ — число листьев в поддереве T .

- Выберем α с помощью кросс-валидации и возьмем соответствующее поддерево.

Преимущества и недостатки решающих деревьев

Преимущества:

- Простота интерпретации.
- Простота визуализации.
- Пригодность и для задач регрессии, и для задач классификации.
- Возможность работы с пропусками в данных.
- Возможность работы с категориальными значениями.

Недостатки:

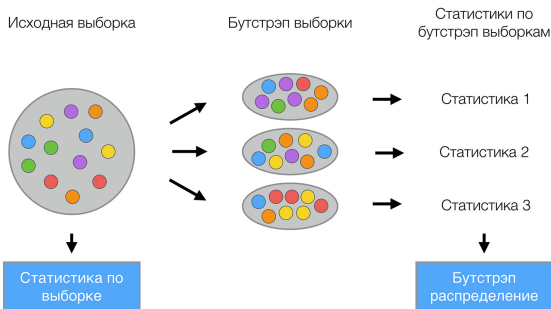
- Метод является неустойчивым и склонным к переобучению.

Замечание

*Агрегирования множества деревьев решений такими способами, как *bagging*, *random forest* и *boosting*, позволяет значительно улучшить качество предсказания.*

Метод бутстрэпа заключается в следующем:

- Дана выборка \mathbf{X} размера M .
- Равномерно возьмем из выборки M объектов с возвращением (из-за возвращения среди них окажутся повторы), обозначим новую выборку через \mathbf{X}_1 .
- Повторив N раз, сгенерируем N подвыборок $\mathbf{X}_1, \dots, \mathbf{X}_N$.



Пример построения композиции алгоритмов.

- Дана конечная выборка $\mathbf{X} = (x_i, y_i)$ с вещественными ответами.
- Равномерно возьмем из выборки l объектов с возвращением (из-за возвращения среди них окажутся повторы), обозначим новую выборку через \mathbf{X}_1 .
- Повторив N раз, сгенерируем N подвыборок $\mathbf{X}_1, \dots, \mathbf{X}_N$.
- Обучим по каждой из них линейную модель регрессии, получив базовые алгоритмы $b_1(x), \dots, b_N(x)$.
- Пусть \exists модель $y(x) = \sum \beta_i x_i + \varepsilon_i$ и $p(x)$ — распределение \mathbf{X} .
- **Ошибка каждой функции регрессии:**
 $\varepsilon_j(x) = b_j(x) - y(x), \quad j = 1, \dots, N.$
- **Матожидание среднеквадратичной ошибки:**
 $\mathbf{E}_x(b_j(x) - y(x))^2 = \mathbf{E}_x \varepsilon_j^2(x).$

Среднеквадратичная ошибка

Средняя ошибка построенных функций регрессии:

$$\mathbf{E}_1 = \frac{1}{N} \sum_{j=1}^N \mathbf{E}_x \varepsilon_j^2(x).$$

Пусть ошибки несмещены и некоррелированы:

$$\mathbf{E}_x \varepsilon_j(x) = 0, \quad \mathbf{E}_x \varepsilon_j(x) \varepsilon_i(x) = 0, \quad i \neq j.$$

Построим теперь новую функцию регрессии, которая будет усреднять ответы построенных нами функций:

$$\alpha(x) = \frac{1}{N} \sum_{j=1}^N b_j(x).$$

Тогда

$$E_N = \mathbf{E}_x \left(\frac{1}{N} \sum_{j=1}^N b_j(x) - y(x) \right)^2 = \mathbf{E}_x \left(\frac{1}{N} \sum_{j=1}^N \varepsilon_j(x) \right)^2 =$$

$$= \frac{1}{N^2} \mathbf{E}_x \left(\sum_{j=1}^N \varepsilon_j^2(x) + \sum_{i \neq j} \varepsilon_i(x) \varepsilon_j(x) \right) = \frac{1}{N} E_1.$$

Таким образом, усреднение ответов позволило уменьшить средний квадрат ошибки в N раз.

Замечание

Рассмотренный пример не очень применим на практике, так как мы предположили, что ошибки некоррелированы, это редко выполняется. Если это предположение неверно, то уменьшение ошибки оказывается не таким значительным.

Bagging — метод, который позволяет уменьшить разброс модели. Этот метод обучает некоторое число алгоритмов так, что каждый алгоритм обучается на отдельных выборках, которые получены из исходной с помощью bootstrap.

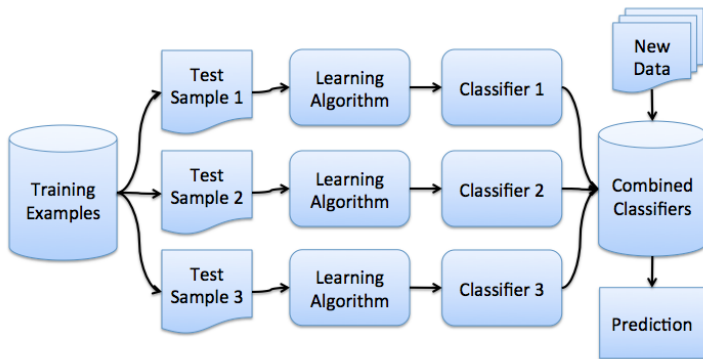


Рис.: Схема реализации bagging

Пусть имеется обучающая выборка \mathbf{X} .

- ❶ С помощью bootstrap сгенерируем из \mathbf{X} выборки $\mathbf{X}_1, \dots, \mathbf{X}_N$.
- ❷ На каждой выборке строим решающее дерево $b_n(x)$:
 - дерево строится, пока в каждом листе не окажется не более n_{\min} объектов.
 - при каждом разбиении сначала выбирается m случайных признаков из p , оптимальное разделение ищется только среди них.

❸ находим оценку:

- для задачи регрессии: $\alpha(x) = \frac{1}{N} \sum_{n=1}^N b_n(x)$.
- для задачи классификации с K классами: записываем класс предсказанный каждым деревом, итоговое предсказание часто встречающийся класс среди предсказаний.

Bagging в задаче регрессии

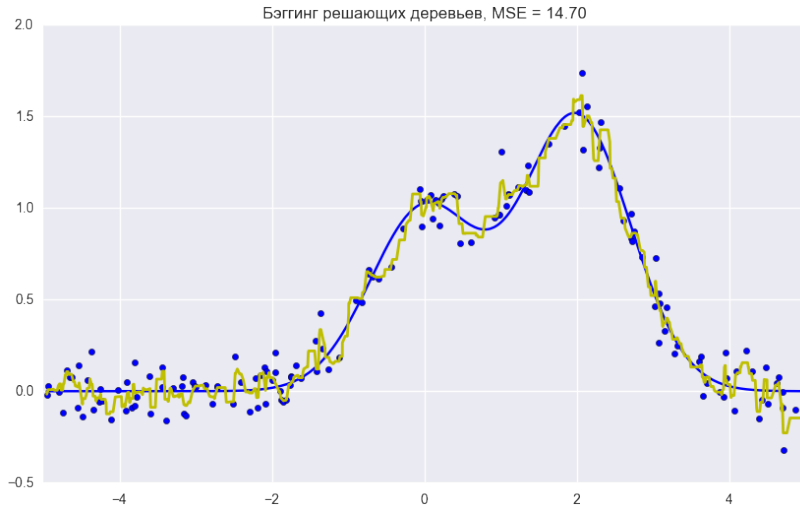


Рис.: Лучше обобщает закономерность и менее склонна к переобучению, чем любое отдельное дерево

Bagging в задаче классификации

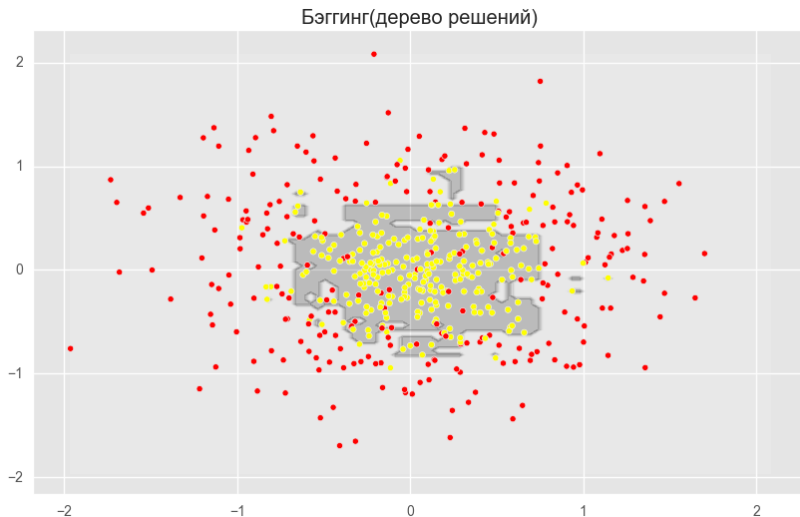


Рис.: Граница сглаженная и практически нет признаков переобучения

- Дерево b_n , обученное с помощью bootstrap, использует в среднем $2/3$ наблюдений.
- Оставшиеся $1/3$ наблюдений, которые не вошли в bootstrap-выборку \mathbf{X}_n , являются контрольными для данного дерева.
- Значит, можно для каждого объекта x_i найти деревья, которые были обучены без него, и вычислить по их ответам out-of-bag ошибку:

$$OOB = \sum_{i=1}^l L \left(y_i, \frac{1}{\sum_{n=1}^N [x_i \notin \mathbf{X}_n]} \sum_{n=1}^N [x_i \notin \mathbf{X}_n] b_n(x_i) \right),$$

где $L(y, z)$ — функция потерь.

Random Forest — улучшение bagging решающих деревьев. Деревья становятся менее коррелированными благодаря тому, что при построении дерева в каждой вершине признак выбирается из случайного подмножества заданного размера m . Как правило, $m = \sqrt{d}$, где d — число признаков.

Рекомендуется

- в задачах классификации строить каждое дерево до тех пор, пока в каждом листе не окажется по одному объекту,
- в задачах регрессии строить каждое дерево до тех пор, пока в каждом листе не окажется по пять объектов.

Random forest в задаче регрессии

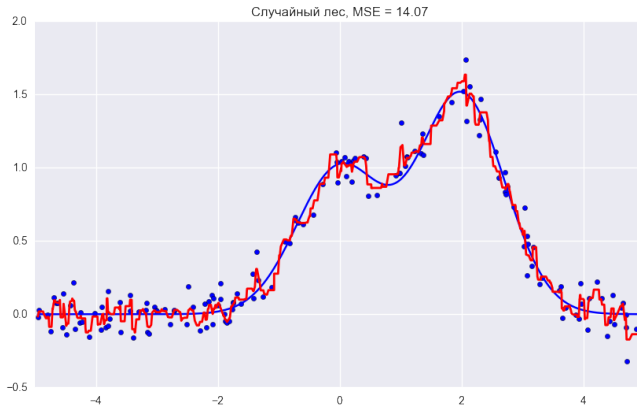


Рис.: Различие случайного леса и бэггинга на деревьях решений заключается в том, что в случайном лесе выбирается случайное подмножество признаков, и лучший признак для разделения узла определяется из подвыборки признаков, в отличие от бэггинга, где все функции рассматриваются для разделения в узле

Random forest в задаче классификации

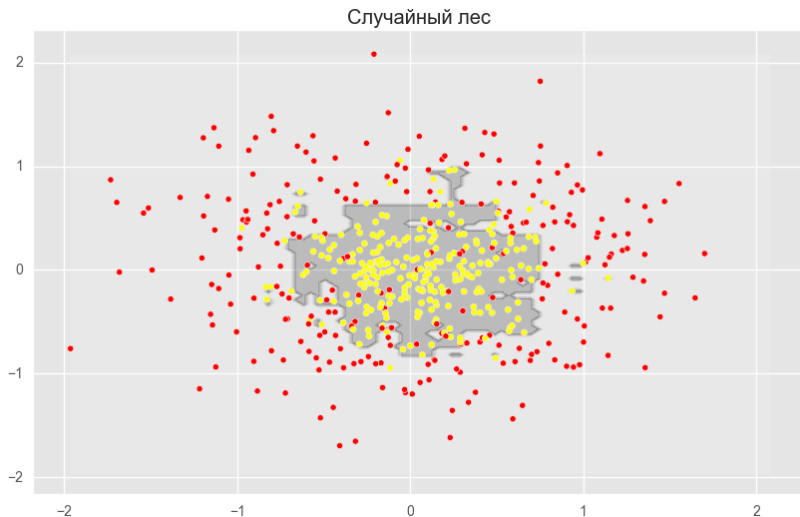


Рис.: Граница сглаженная и практически нет признаков переобучения

Boosting — техника последовательного обучения множества слабых базовых моделей для построения более качественной общей модели предсказания. В этом случае используется одно обучающее множество. Первая модель в последовательности обучается предсказывать, а последующие стремятся уменьшить ошибки предсказания предыдущих.

В итоге получаем аддитивную модель как взвешенную последовательность множества базовых.

Рассмотрим в общем виде **Gradient Boosting** для задачи регрессии и **AdaBoost** для задачи классификации.

Gradient Boosting для задачи регрессии

В качестве базовой модели берем дерево решений. Boosting использует комбинацию большого количества деревьев.

Деревья строятся последовательно, при этом каждое следующее дерево использует информацию предыдущих деревьев.

Алгоритм:

- ❶ Устанавливаем $b(x) = 0$ и $r_i = y_i$ для всех i на обучающем множестве (\mathbf{X}, r) .
- ❷ Для $n = 1, 2, \dots, N$ повторяем:
 - Обучаем дерево b_n с d сплитами ($d + 1$ терминальных узлов) на обучающем множестве (\mathbf{X}, r) .
 - Обновляем b : $b(x) \leftarrow b(x) + \lambda b_n(x)$
 - Обновляем остатки: $r_i \leftarrow r_i - \lambda b_n(x_i)$
- ❸ Возвращаем модель:

$$b(x) = \sum_{n=1}^N \lambda b_n(x).$$

Три настраиваемых параметра:

- ❶ Количество деревьев N . При слишком большом значении N может привести к переобучению. Можно использовать кросс-валидацию для выбора N .
- ❷ Параметр регуляризации λ — небольшое положительное число, которое контролирует скорость обучения. При очень малом значении λ необходимо использовать очень большое значение N , чтобы получить приемлимый результат.
- ❸ Количество сплитов d в каждом дереве. Параметр определяет сложность ансамбля.

Общая идея:

- Последовательно строим модель на базе слабых классификаторов при этом уменьшаем ошибку классификатора на предыдущем шаге.
- Если на обучающем множестве были случаи неправильной классификации, то для этих наблюдений увеличивается вес.
- Следующий классификатор при обучении использует эти веса, чтобы сделать акцент на случаях, где была ошибка.
- После этого обученный классификатор может сам давать ошибочные классификации, что ведет к изменению весов наблюдений для последующего классификатора.
- Каждый классификатор в последовательности имеет свой вес.
- Таким образом, получаем конечный классификатор как линейную комбинацию последовательно обученных классификаторов.

Алгоритм:

- ❶ Инициализация весов наблюдений $w_i = \frac{1}{n}$ где $i = 1, 2, \dots, n$.
- ❷ for $m = 1$ to M :
 - Обучение классификатора $T_m(x)$ на обучающих данных с весами w_i .
 - Вычисление ошибки: $err_m = \frac{\sum_{i=1}^n w_i \mathbf{1}(y_i \neq T_m(x_i))}{\sum_{j=1}^n w_j}$.
 - Вычисление веса классификатора $G_m(x)$:
$$\alpha_m = \log \left[\frac{1 - err_m}{err_m} \right] + \log(K - 1), \text{ где } K — \text{ количество классов.}$$
 - Обновление весов $w_i \leftarrow w_i \times \exp(\alpha_m \times \mathbf{1}(y_i \neq T_m(x_i)))$.
 - Нормализация весов w_i .
- ❸
$$C(x) = \operatorname{argmax}_k \left\{ \sum_{m=1}^M \alpha_m \times \mathbf{1}(T_m(x) = k) \right\}.$$

В Random forest:

- Используются глубокие деревья, так как от базовых алгоритмов требуется низкое смещение.
- Разброс устраняется за счет усреднения ответов различных деревьев.

В Boosting:

- Каждый следующий алгоритм понижает ошибку композиции.
- Переобучение при большом количестве базовых моделей.
- Можно понизить смещение моделей, но разброс или останется таким же, или увеличится.
- Используются неглубокие решающие деревья.