Градиентный бустинг

Московский физико-технический институт

Москва, 19 февраля 2025 г.

План лекции

- Воспоминания из предыдущей лекции
- Градиентный бустинг
- Learning rate
- Сравнение случайного леса и градиентного бустинга
- Реализации градиентного бустинга

Определение решающего дерева (воспоминание)

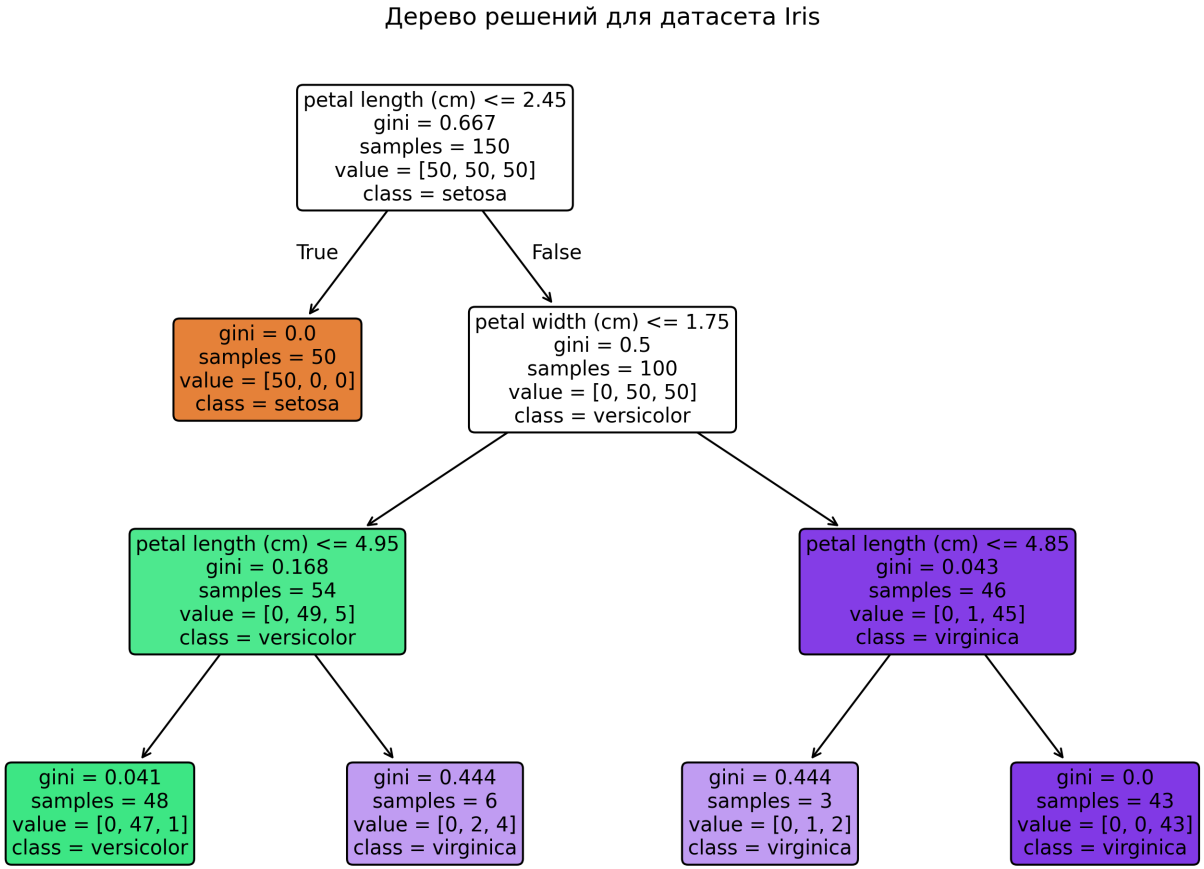
Решающие дерево - это **бинарное дерево**:

- каждой внутренней вершине v соответствует предикат $B_v : X \to \{0, 1\}$
- каждому листу v соответствует прогноз $c_v \in Y$
- $B(x, j, t) = [x_j \leq t]$

Предсказание решающего дерева

- Для объекта x начинаем движение из корня.
- В вершине v: движемся **вправо**, если $B(x, j, t) = 1$, иначе **влево**.
- Движемся до тех пор, пока не достигнем листа (осуществим прогноз).

Пример решающего дерева (воспоминание)



- **узел** - это предикат (правило)
- если предикат верен, переходим в **правого потомка**, если нет — **в левого**
- **листья** - это классы

Пусть дана обучающая выборка (X, y), $|X| = n$

Генерация подвыборки

Выберем из выборки (X, y) равновероятно n объектов. Получим новую выборку (X^i, y^i).

Генерация подвыборки

Обучим на выборке (X^i, y^i) дерево таким образом, что в процессе обучения каждого дерева в каждой вершине случайно выбираются $d_i < d$ признаков, где d — полное число признаков.

классификация

$$y(x) = \arg\max_k \sum_{i=1}^{B} \mathbf{1}(b_i(x) = k),$$

регрессия

$$y(x) = \frac{1}{B} \sum_{b=1}^{B} b_i(x).$$

Постановка задачи градиентного бустинга

Пусть у нас есть обучающая выборка $\{(x_i, y_i)\}_{i=1}^n$

- $x_i \in \mathbb{R}^d$ — признаковое описание объекта
- $y_i \in \mathbb{R}$ — целевая переменная (для регрессии)
- d — число признаков объекта
- n — количество объектов в выборке

Цель: найти функцию $F(x)$, которая минимизирует заданный функционал потерь $L(y, F(x))$:

$$F^*(x) = \arg\min_{F(x)} \sum_{i=1}^{n} L(y_i, F(x_i)).$$

Градиентный бустинг строит композицию моделей $F(x)$ как **сумму базовых моделей** $h_m(x)$:

$$F_M(x) = \sum_{m=0}^{M} \gamma_m h_m(x),$$

- M – количество итераций (количество базовых моделей)
- $h_m(x)$ – базовая модель (например, решающее дерево)
- γ_m – коэффициент, определяющий "вклад"каждой модели

На каждой итерации m мы добавляем **новую базовую модель** $h_m(x)$, которая минимизирует функционал потерь:

$$h_m(x) = \arg\min_h \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + h(x_i)\right).$$

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)},$$

где r_{im} называется **псевдооткликом или градиентом остатков**. Затем мы обучаем базовую модель $h_m(x)$ для аппроксимации этих псевдооткликов:

$$h_m(x) = \arg\min_h \sum_{i=1}^{n} (r_{im} - h(x_i))^2.$$

После этого определяется оптимальный коэффициент γ_m, который минимизирует функционал потерь:

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

Наконец, обновляется композиция:

$$F_m(x) = F_{m-1}(x) + \nu\gamma_m h_m(x),$$

где $\nu \in (0, 1]$ — **параметр скорости обучения (learning rate)**.

1. Инициализация модели:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. Для каждой итерации $m = 1, 2, \ldots, M$: - Вычисление градиентов остатков:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}.$$

$$h_m(x) = \arg\min_{h} \sum_{i=1}^{n} (r_{im} - h(x_i))^2.$$

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

$$F_m(x) = F_{m-1}(x) + \nu\gamma_m h_m(x).$$

3. Вывод финальной модели:

$$F_M(x) = \sum_{m=0}^{M} \nu\gamma_m h_m(x).$$

Важные моменты

- Какие функции потерь можно использовать?
- Какой выбрать learning rate?
- В чем разница между γ_m и ν?

Среднеквадратичная ошибка (MSE)

$$L(y, F(x)) = \frac{1}{2}(y - F(x))^2.$$

$$\frac{\partial L(y, F(x))}{\partial F(x)} = -(y - F(x)).$$

$$r_{im} = y_i - F_{m-1}(x_i).$$

Средняя абсолютная ошибка (MAE)

$$L(y, F(x)) = |y - F(x)|.$$

$$\frac{\partial L(y, F(x))}{\partial F(x)} = \begin{cases} -1, & \text{если } y > F(x), \\ 1, & \text{если } y < F(x), \\ 0, & \text{если } y = F(x). \end{cases}$$

$$r_{im} = \begin{cases} 1, & \text{если } y_i > F_{m-1}(x_i), \\ -1, & \text{если } y_i < F_{m-1}(x_i). \end{cases}$$

Log loss

Пусть k — количество классов, а $p_j(x) = \frac{e^{F_j(x)}}{\sum_{l=1}^{k} e^{F_l(x)}}$ — вероятность принадлежности объекта к классу j. Тогда функция потерь:

$$L(y, F(x)) = -\sum_{j=1}^{k} \mathbf{1}(y = j) \log(p_j(x)),$$

$$\frac{\partial L(y, F(x))}{\partial F_j(x)} = p_j(x) - \mathbf{1}(y = j).$$

Скорость обучения (learning rate)

небольшой learning rate

- Значения $\nu \in (0, 0.1]$ являются стандартными для большинства задач.
- Маленький learning rate помогает избежать переобучения, так как он позволяет модели постепенно адаптироваться к данным.
- Модель может найти более гладкое решение, которое лучше работает на новых данных.

большой learning rate

При большом ν можно использовать меньшее количество итераций, но риск переобучения возрастает.

γ_m

γ_m — это коэффициент масштабирования для конкретной базовой модели $h_m(x)$ на m-й итерации градиентного бустинга.
Он **подбирается индивидуально** для каждой базовой модели таким образом, чтобы минимизировать функционал потерь.

ν

Learning rate (ν) — это гиперпараметр алгоритма градиентного бустинга, который масштабирует все обновления модели на каждом шаге.
Он **применяется к каждому** γ_m, чтобы сделать шаги более консервативными.

У нас есть данные о том будет ли человек гулять (y) в зависимости от температуры (X):

Температура X	будет гулять y
10	No
15	No
20	Yes
25	Yes
30	Yes
35	Yes

Мы будем использовать решающие деревья как базовые модели и логарифмическую функцию потерь (log loss).

Мы используем логарифмическую функцию потерь (log loss):

$$L(y, F(x)) = \log(1 + e^{-yF(x)}),$$

- $y \in \{-1, +1\}$ — истинная метка,
- $F(x)$ — текущий прогноз модели.

Градиент этой функции потерь:

$$\frac{\partial L(y, F(x))}{\partial F(x)} = -\frac{y}{1 + e^{yF(x)}}.$$

Псевдоотклики (градиенты остатков):

$$r_{im} = -\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F(x_i)} = \frac{y_i}{1 + e^{y_i F_{m-1}(x_i)}}.$$

Начальная модель $F_0(x)$ определяется как константа, равная среднему значению целевой переменной, преобразованному через логарифмические шансы:

$$F_0(x) = \frac{1}{N} \sum_{i=1}^{N} y_i.$$

Подсчитаем:

$$F_0(x) = \frac{-1 - 1 + 1 + 1 + 1 + 1}{6} = \frac{2}{6} = 0.333.$$

Таким образом, начальная модель — это константа: $F_0(x) = 0.333$.

Для первой итерации вычислим псевдо-остатки, используя производную логистической функции потерь:

$$r_{im} = -\frac{\partial L}{\partial F(x_i)} = y_i \cdot \frac{e^{-y_i F(x_i)}}{1 + e^{-y_i F(x_i)}}.$$

Здесь $p_i = \frac{1}{1+e^{y_i F(x_i)}}$ – вероятность ошибки для объекта x_i.
Подставляем значения:

x_i	y_i	$F_0(x_i) = 0.333$	$p_i = \frac{1}{1+e^{y_i \cdot 0.333}}$	$r_{i1} = y_i \cdot p_i$
10	-1	0.333	$\frac{1}{1+e^{-(-1)\cdot 0.333}} = \frac{1}{1+e^{0.333}} \approx 0.426$	$-1 \cdot 0.426 = -0.426$
15	-1	0.333	$\frac{1}{1+e^{-(-1)\cdot 0.333}} = \frac{1}{1+e^{0.333}} \approx 0.426$	$-1 \cdot 0.426 = -0.426$
20	$+1$	0.333	$\frac{1}{1+e^{-1\cdot 0.333}} = \frac{1}{1+e^{-0.333}} \approx 0.574$	$+1 \cdot 0.574 = 0.574$
25	$+1$	0.333	$\frac{1}{1+e^{-1\cdot 0.333}} = \frac{1}{1+e^{-0.333}} \approx 0.574$	$+1 \cdot 0.574 = 0.574$
30	$+1$	0.333	$\frac{1}{1+e^{-1\cdot 0.333}} = \frac{1}{1+e^{-0.333}} \approx 0.574$	$+1 \cdot 0.574 = 0.574$
35	$+1$	0.333	$\frac{1}{1+e^{-1\cdot 0.333}} = \frac{1}{1+e^{-0.333}} \approx 0.574$	$+1 \cdot 0.574 = 0.574$

Обучение решающего дерева $h_1(x)$

На этом шаге мы обучаем решающее дерево $h_1(x)$, чтобы аппроксимировать псевдо-остатки r_{i1}. Для простоты предположим, что дерево имеет глубину 1 (одно разбиение).
Разделим данные по температуре $X = 18$:

x_i	r_{i1}	Класс после разбиения
10	-0.426	Среднее $(-0.426 - 0.426)/2 = -0.426$
15	-0.426	Среднее $(-0.426 - 0.426)/2 = -0.426$
20	0.574	Среднее $(0.574 + 0.574 + 0.574 + 0.574)/4 = 0.574$
25	0.574	Среднее $(0.574 + 0.574 + 0.574 + 0.574)/4 = 0.574$
30	0.574	Среднее $(0.574 + 0.574 + 0.574 + 0.574)/4 = 0.574$
35	0.574	Среднее $(0.574 + 0.574 + 0.574 + 0.574)/4 = 0.574$

Таким образом, решающее дерево $h_1(x)$ имеет вид:

$$h_1(x) = \begin{cases} -0.426, & \text{если } x < 18, \\ 0.574, & \text{если } x \geq 18. \end{cases}$$

Теперь найдем оптимальное значение γ_1, которое минимизирует функционал потерь:

$$\gamma_1 = \arg\min_{\gamma} \sum_{i=1}^{N} \log(1 + e^{-y_i(F_0(x_i)+\gamma h_1(x_i))}).$$

Это можно сделать численно или аналитически. Предположим, что $\gamma_1 = 0.9$. Обновим общую модель:

$$F_1(x) = F_0(x) + \gamma_1 h_1(x).$$

Подставим значения:

$$F_1(x) = \begin{cases} 0.333 + 0.9 \cdot (-0.426) = -0.050, & \text{если } x < 18, \\ 0.333 + 0.9 \cdot 0.574 = 0.849, & \text{если } x \geq 18. \end{cases}$$

1. Для нового объекта x вычислить выход модели $F_M(x)$:

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x).$$

2. Преобразовать $F_M(x)$ в вероятность:

$$P(y = +1 \mid x) = \frac{1}{1 + e^{-F_M(x)}}.$$

3. Определить метку класса:

$$\hat{y} = \begin{cases} +1, & \text{если } P(y = +1 \mid x) \geq 0.5, \\ -1, & \text{если } P(y = +1 \mid x) < 0.5. \end{cases}$$

или, эквивалентно:

$$\hat{y} = \begin{cases} +1, & \text{если } F_M(x) \geq 0, \\ -1, & \text{если } F_M(x) < 0. \end{cases}$$

Случайный лес

- **Тип ансамбля:** Случайный лес использует метод бэггинга (bagging), где каждое дерево обучается независимо на случайной подвыборке данных с возвращением (bootstrap).
- **Связь между деревьями:** Деревья в случайном лесе не зависят друг от друга.
- **Цель:** Уменьшить дисперсию модели за счет усреднения предсказаний нескольких деревьев.

Градиентный бустинг

- **Тип ансамбля:** Градиентный бустинг использует метод бустинга (boosting), где деревья строятся последовательно, каждое новое дерево компенсирует ошибки предыдущих.
- **Связь между деревьями:** Деревья зависят друг от друга: каждое новое дерево обучается на ошибках предыдущей модели.
- **Цель:** меньшить как смещение (bias), так и дисперсию модели через последовательное улучшение.

Случайный лес

- Каждое дерево обучается на своей **случайной подвыборке** данных.
- При каждом разбиении узла **используется только подмножество признаков** (например, $\sqrt{D}$ для задач классификации или $D/3$ для задач регрессии, где D — общее количество признаков).

Градиентный бустинг

- Каждое дерево **обучается на всех данных**, но с весами, которые зависят от ошибок предыдущей модели.
- При обучении нового дерева минимизируется функционал потерь, что делает процесс более сложным и вычислительно затратным.

Сравнение случайного леса и градиентного бустинга

Случайный лес

- Обучение каждого дерева происходит **параллельно**, что делает случайный лес более быстрым на больших наборах данных.
- Требует меньше вычислительных ресурсов, так как деревья независимы.

Градиентный бустинг

- Обучение деревьев происходит **последовательно**, что делает процесс более медленным.
- Требует больше вычислительных ресурсов из-за необходимости оптимизации функционала потерь на каждом шаге.

На данный момент существуют **три популярные реализации** градиентного бустинга, которые в целом используют общии идеи:

- XGBoost
- LightGBM
- CatBoost

Одно из основных отличий LightGBM, XGBoost и CatBoost — **форма решающих деревьев**.

XGBoost

XGBoost использует **жадный алгоритм для поиска оптимальных разбиений**. На каждом шаге он оценивает все возможные точки разбиения для каждого признака и выбирает ту, которая максимизирует прирост в функционале потерь.

LightGBM

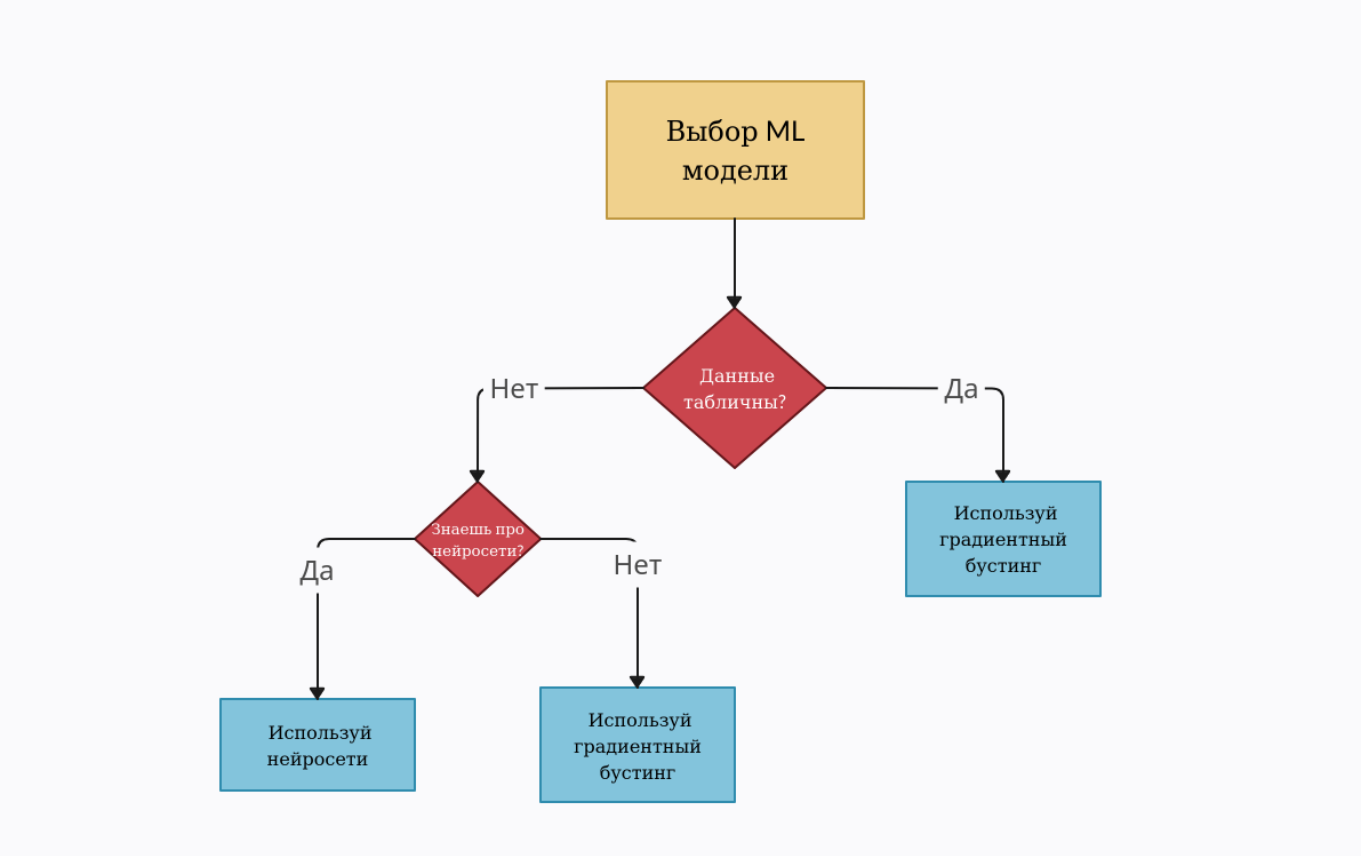
Вместо того чтобы строить деревья уровня за уровнем ("level-wise"), LightGBM использует стратегию "leaf-wise"— выбирает лучшее возможное разбиение среди всех листьев текущего дерева. Это позволяет добиться более глубоких и точных деревьев, но увеличивает риск переобучения..

CatBoost

CatBoost использует стратегию "symmetric trees"— **сбалансированные деревья** фиксированной глубины. Это позволяет упростить вычисления и уменьшить вероятность переобучения.

Аспект	**XGBoost**	**LightGBM**	**CatBoost**
Разбиение	Level-wise	Leaf-wise	Symmetric
Категории	One-hot	EFB	Встроенные
Регуляризация	L1/L2	L1/L2	L2
Скорость	Средняя	Быстрая	Медленная
Переобучение	Среднее	Высокое	Низкое

Краткое сравнение XGBoost, LightGBM и CatBoost



Стоит отметить, что со временем данный метод постепенно может уйти в прошлое. Это связано с тем, что прогресс в разработке архитектур глубокого обучения, а также совершенствование вычислительного оборудования уменьшает его приемущества по сравнению с нейросетевыми решениями.

Литература

- Полезные ссылки
- Полезные ссылки

Вопросы?