Градиентный бустинг

Паточенко Евгений

НИУ ВШЭ

План занятия

- Напоминание
- Бустинг
- XGBoost
- CatBoost
- LightGBM

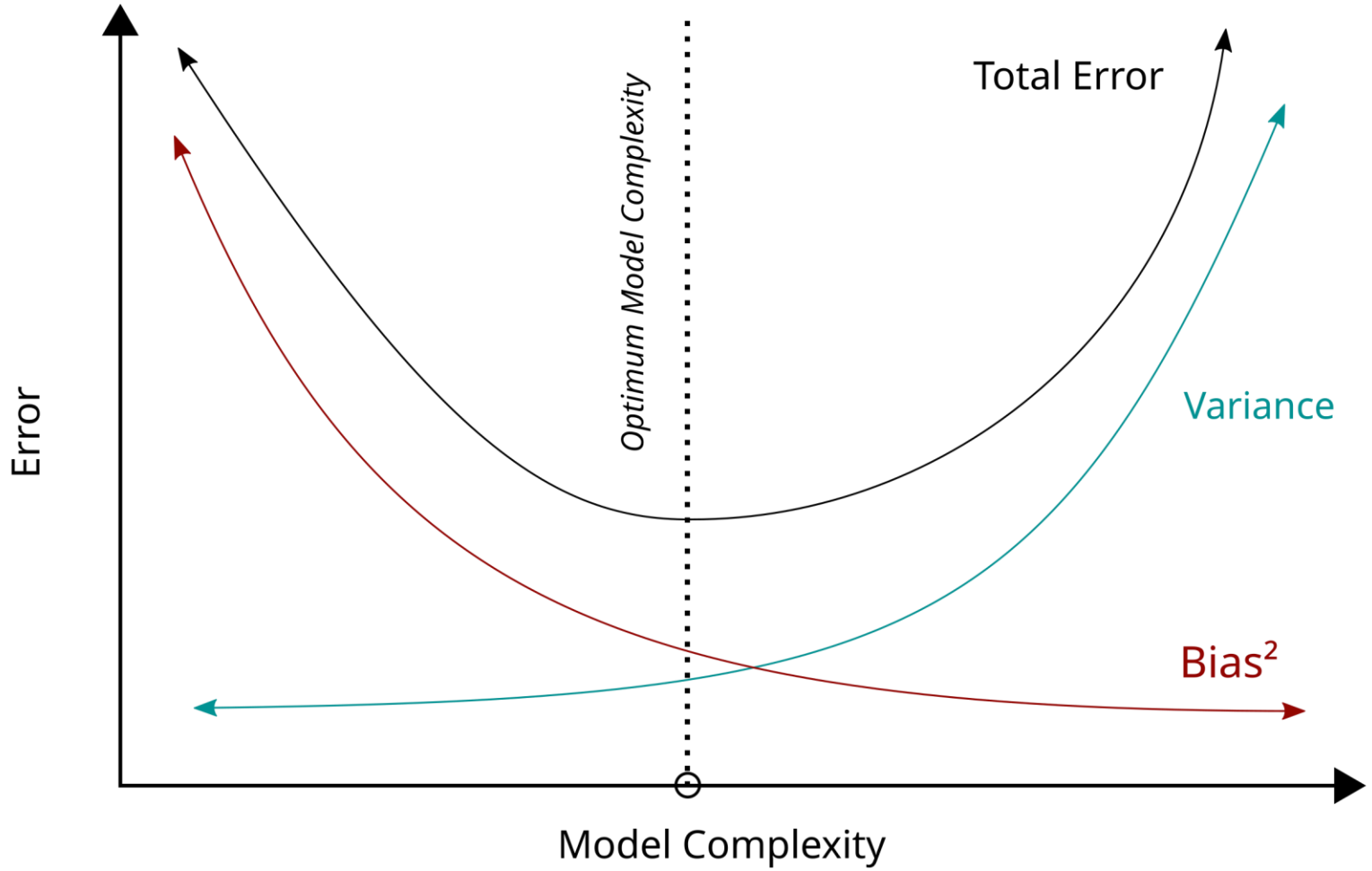
Напоминание: декомпозиция ошибки

Компоненты ошибки:

- Смещение (bias) — способность модели приблизить лучшую среди всех возможных моделей
- Разброс (variance) — устойчивость модели к изменениям в обучающей выборке
- Шум (noise) — характеристика сложности и противоречивости данных

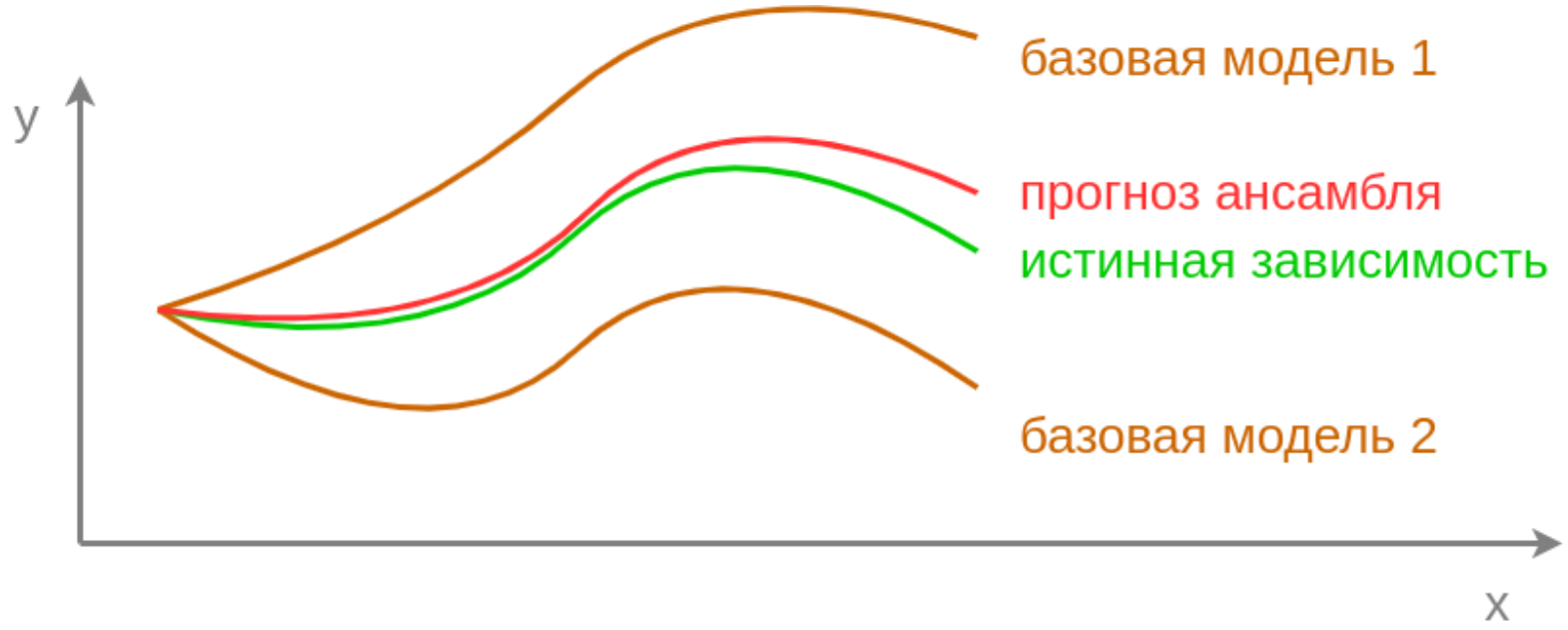
$$Q(a) = \mathbb{E}_x bias_X^2 a(x, X) + \mathbb{E}_x \mathbb{V}_x[a(x, X)] + \sigma^2$$

Напоминание: декомпозиция ошибки



Источник: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Напоминание: ансамбли и бэггинг



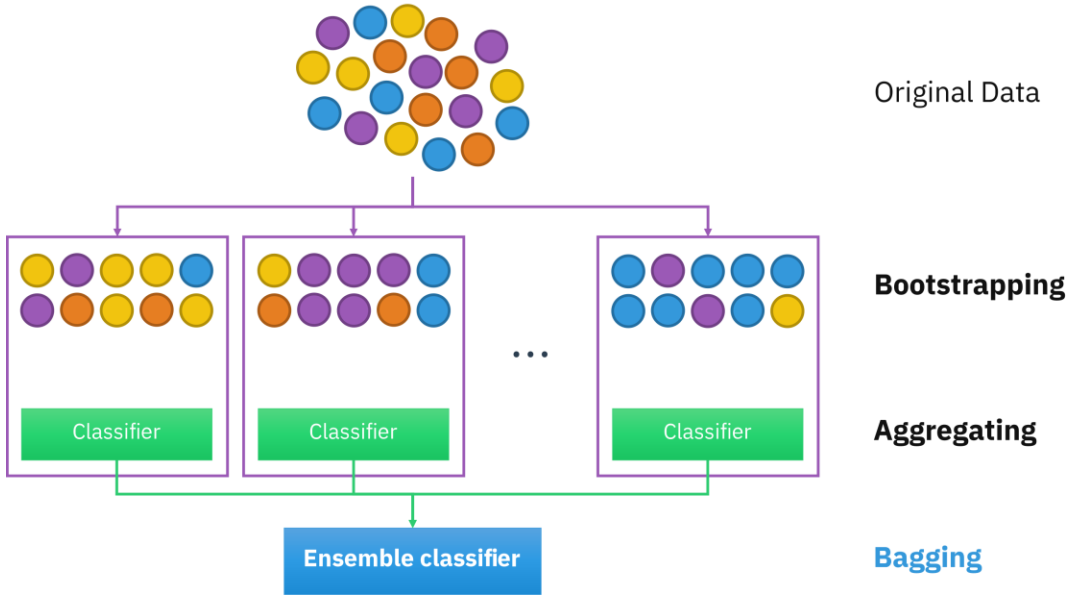
Источник: https://deepmachinelearning.ru/docs/Machine-learning/Model-ensembles/Model-ensembles

С помощью ансамблирования снижаем разброс и смещение

Напоминание: ансамбли и бэггинг

Бэггинг (bagging, bootstrap aggregating) — метод, при котором несколько моделей обучаются независимо на случайных подвыборках данных:

1. Создаются бутстреп-выборки
2. Обучаются одинаковые модели
3. Предсказания усредняются или голосуются

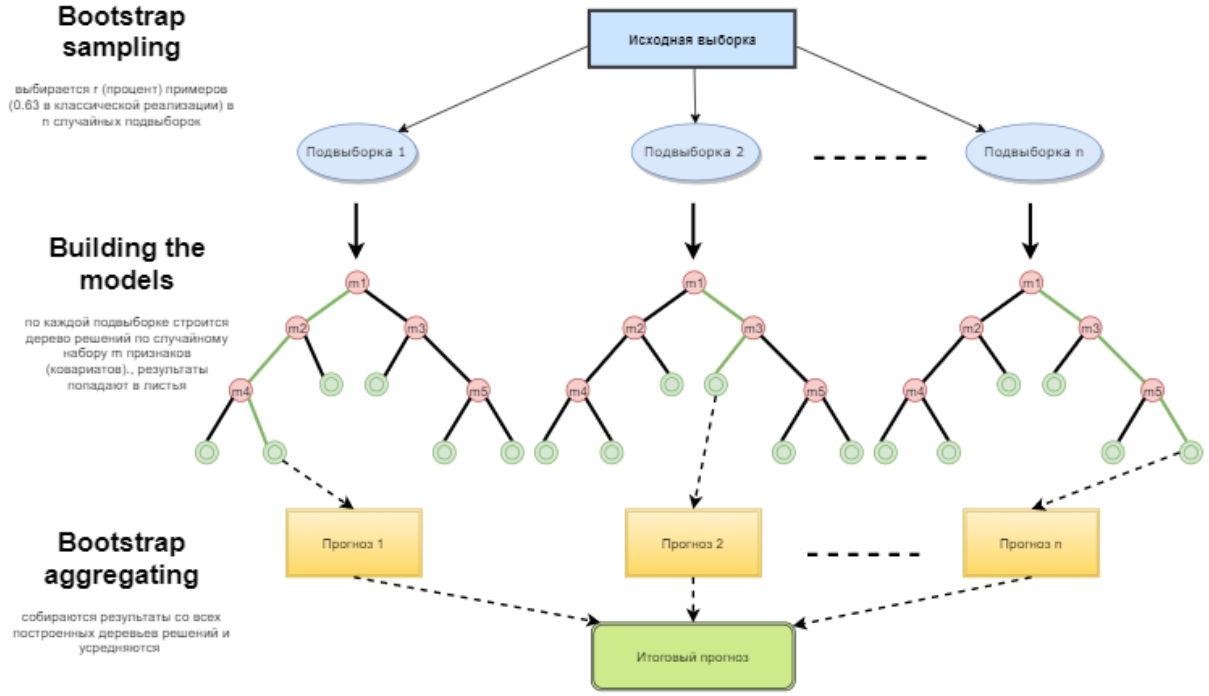


Источник: https://en.wikipedia.org/wiki/Bootstrap_aggregating

Напоминание: случайный лес

Базовые модели — деревья решений. На каждом шаге обучения используется дополнительное случайное подмножество признаков для разделения узлов:

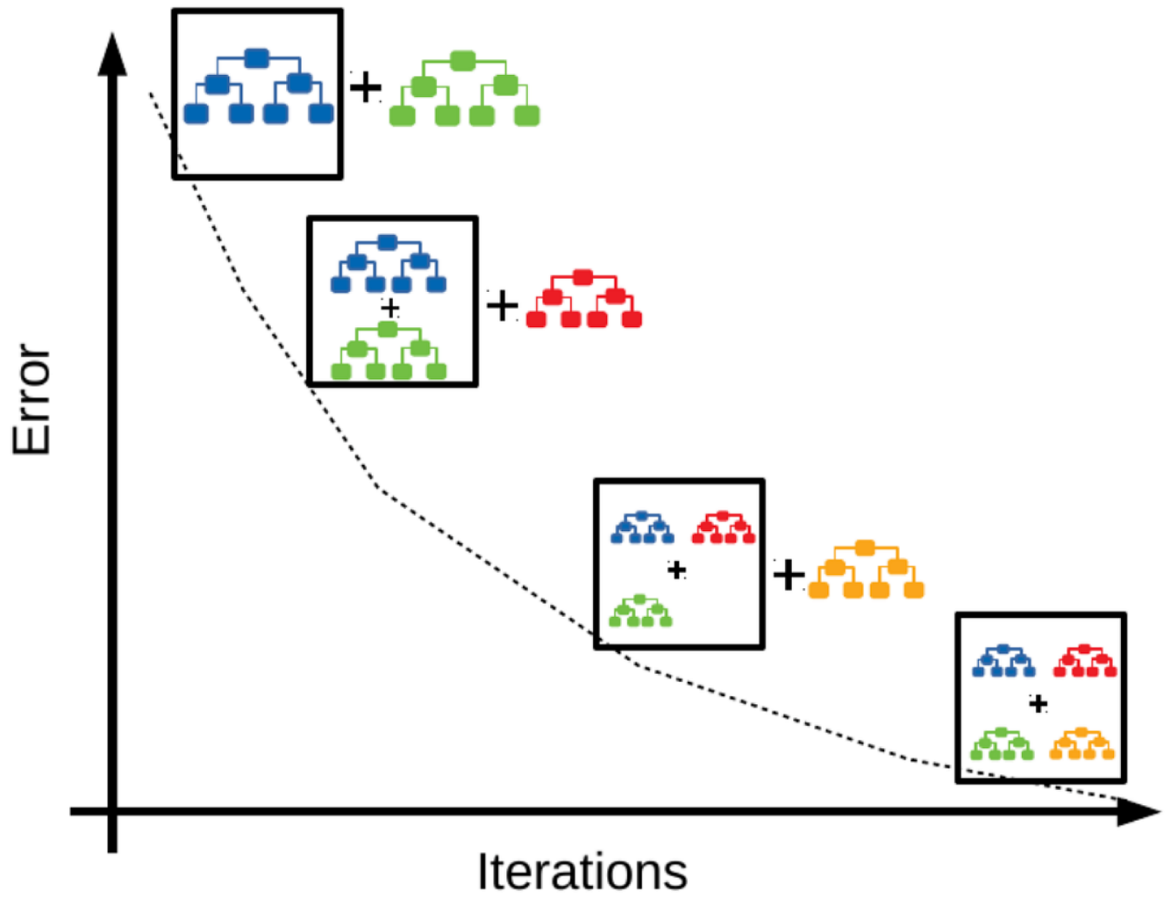
1. Создаются бутстреп-выборки
2. Обучаются несколько деревьев
3. При построении каждого узла дерева выбирается случайное подмножество признаков, из которых выбирается лучший сплит
4. Предсказания усредняются или голосуются



Источник: https://www.mql5.com/ru/articles/3856

Бустинг

Набор алгоритмов, каждый из которых исправляет ошибки предыдущего



Источник: https://medium.com/@hemashreekilari9/understanding-gradient-boosting-632939b98764

Бустинг в задаче регрессии

Решаем задачу регрессии с минимизацией квадратичной ошибки:

$$\frac{1}{2}\sum_{i=1}^{l}(a(x_i)-y_i)^2 \to \min_a$$

Ищем алгоритм $a(x)$ в виде суммы N базовых алгоритмов

$$a(x)=\sum_{n=1}^{N} b_n(x)\,,$$

где базовые алгоритмы $b_n(x)$ принадлежать некоторому семейству A

Бустинг в задаче регрессии

1. Ищем алгоритм $b_1(x),$ минимизирующий ошибку:

$$b_1(x) = \underset{b \in A}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - y_i)^2$$

Ошибка на объекте x:

$$s = \mathrm{y} - b_1(x),$$

Бустинг в задаче регрессии

1. Ищем алгоритм $b_1(x),$ минимизирующий ошибку:

$$b_1(x) = \underset{b \in A}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - y_i)^2$$

Ошибка на объекте x:

$$s = \mathrm{y} - b_1(x),$$

Целевая переменная для следующего алгоритма — вектор ошибок s (а не исходный вектор y)

Бустинг в задаче регрессии

1. Ищем алгоритм $b_1(x)$, минимизирующий ошибку:

$$b_1(x) = \underset{b \in A}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - y_i)^2$$

2. Ищем алгоритм $b_2(x)$, настраивающийся на ошибки s первого алгоритма:

$$b_2(x) = \underset{b \in A}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - \boldsymbol{s}_i)^2$$

Бустинг в задаче регрессии

Каждый следующий алгоритм настраиваем на ошибку предыдущих

Ошибка:

$$s_i^{(N)} = \mathrm{y}_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i)$$

Алгоритм $b_N(x)$:

$$b_N(x) = \underset{b \in A}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - \boldsymbol{s_i^{(N)}})^2$$

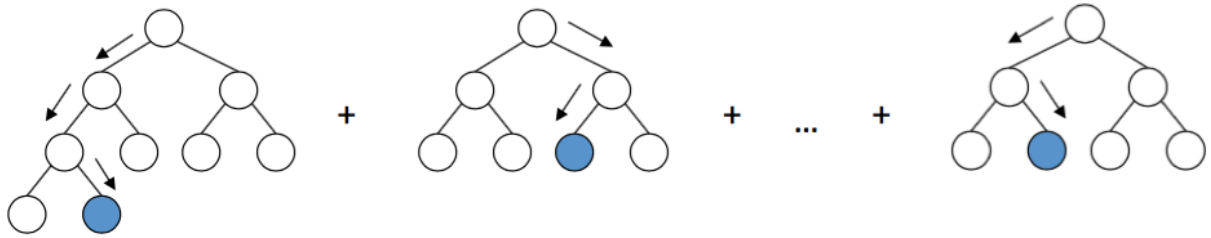
Градиентный бустинг

Ошибка на N-м шаге — это **антиградиент** функции потерь по ответу модели, вычисленный в точке ответа уже построенной композиции:

$$s_i^{(N)} = \mathrm{y}_i - a_{N-1}(x_i) = -\frac{\partial}{\partial z}\frac{1}{2}(z - y_i)^2, \quad z = a_{N-1}(x_i)$$

Градиентный бустинг

Как и в других ансамблях, в бустинге в качестве базовой модели чаще всего используются **решающие деревья (градиентный бустинг над решающими деревьями, Gradient Boosting on Decision Trees, GBDT)**



Можно использовать и линейные модели, но тогда алгоритм будет представлять линейную комбинацию линейных моделей (то есть будет оставаться линейной моделью) и не сможет эффективно работать с нелинейными зависимостями

Градиентный бустинг

Оценка градиента деревьями решений

При построении следующего решающего дерева для оценки качества приближения вектора антиградиента g на i-ом объекте используют следующие функции:

$$L_2(g,p) = \sum_{i=1}^{N} (p_i - g_i)^2 ,$$

$$Cosine(g,p) = -\frac{\sum_{i=1}^{N}(p_i \cdot g_i)}{\sqrt{\sum_{i=1}^{N} {p_i}^2} \cdot \sqrt{\sum_{i=1}^{N} {g_i}^2}}$$

где p_i — предсказание антиградиента на i-ом объекте, а g_i — его истинное значение

Градиентный бустинг

Слишком простые базовые алгоритмы плохо приближают антиградиент функции потерь, то есть градиентный бустинг может свестись к случайному блужданию

Слишком сложные базовые алгоритмы приводят к переобучению, то есть бустинг подгонится под обучающую выборку за несколько шагов.

Градиентный бустинг

Слишком простые базовые алгоритмы плохо приближают антиградиент функции потерь, то есть градиентный бустинг может свестись к случайному блужданию

Слишком сложные базовые алгоритмы приводят к переобучению, то есть бустинг подгонится под обучающую выборку за несколько шагов.

Для решающих деревьев оптимальная глубина — от 3 до 6 в зависимости от конкретной задачи.

Градиентный бустинг

Слишком простые базовые алгоритмы плохо приближают антиградиент функции потерь, то есть градиентный бустинг может свестись к случайному блужданию

Слишком сложные базовые алгоритмы приводят к переобучению, то есть бустинг подгонится под обучающую выборку за несколько шагов.

Можно также сократить шаг обучения. Чем меньше темп, тем меньше степень доверия к каждому базовому алгоритму и тем лучше качество итоговой композиции.

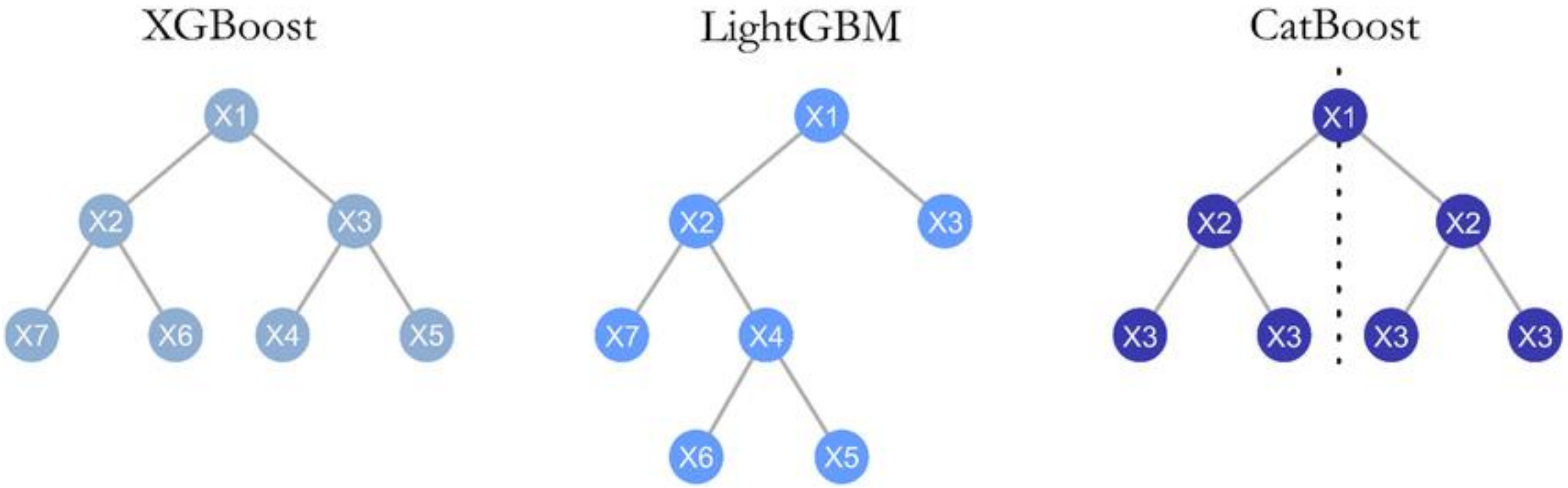
Стохастический градиентный бустинг

Аналогично логике градиентного спуска: обучаем базовый алгоритм b_N не на всей выборке X, а на случайной подвыборке $X^k \subset X$

- ✓ Снижается уровень шума в данных
- ✓ Вычисления становятся быстрее

Обычно берут $\left|X^k\right| = \frac{1}{2}|X|$

Реализации градиентного бустинга



Источник: https://www.researchgate.net/figure/Different-tree-structures-and-split-indexes-shown-inside-each-node-generated-by_fig1_373472181

XGBoost (eXtreme Gradient Boosting)

- Базовый алгоритм приближает направление, посчитанное с учетом второй производной функции потерь
- Функционал регуляризуется — добавляются штрафы за количество листьев и за норму коэффициентов
- При построении дерева используется критерий информативности, зависящий от оптимального вектора сдвига
- Критерий останова также зависит от оптимального сдвига

$$\sum_{i=1}^{l}(-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i)) + \gamma J + \frac{\lambda}{2}\sum_{j=1}^{J} b_j^2 \to \min_b$$

XGBoost (eXtreme Gradient Boosting)

- **Базовый алгоритм приближает направление, посчитанное с учетом второй производной функции потерь**

Первая производная говорит в каком направлении уменьшать ошибку

Вторая производная говорит как быстро ошибка меняется в этом направлении

То есть благодаря второй производной алгоритм знает не только в каком направлении делать шаг, но и оптимальный размер шага

XGBoost (eXtreme Gradient Boosting)

- **Базовый алгоритм приближает направление, посчитанное с учетом второй производной функции потерь**

Это позволяет:

— улучшить точность выбора разбиений дерева,

— делать обновления весов более оптимальными,

— ускорять обучение,

— уменьшать переобучение,

— стабилизировать обучение на сложных функциях потерь.

CatBoost (Categorical Boosting)

Алгоритм, разработанный в Яндексе. Оптимизация XGBoost, которая:

— умеет обрабатывать категориальные признаки

— использует симметричные деревья решений с одним и тем же решающим правилом на одном и том же уровне

— при кодировании категориальных признаков используется набор методов: OHE, счетчики, комбинации признаков и т.д.

— работает с пропусками в данных

CatBoost (Categorical Boosting)

Алгоритм, разработанный в Яндексе. Оптимизация XGBoost, которая:

— обучается быстрее, чем XGBoost

— показывает хороший результат даже без тонкой настройки гиперпараметров

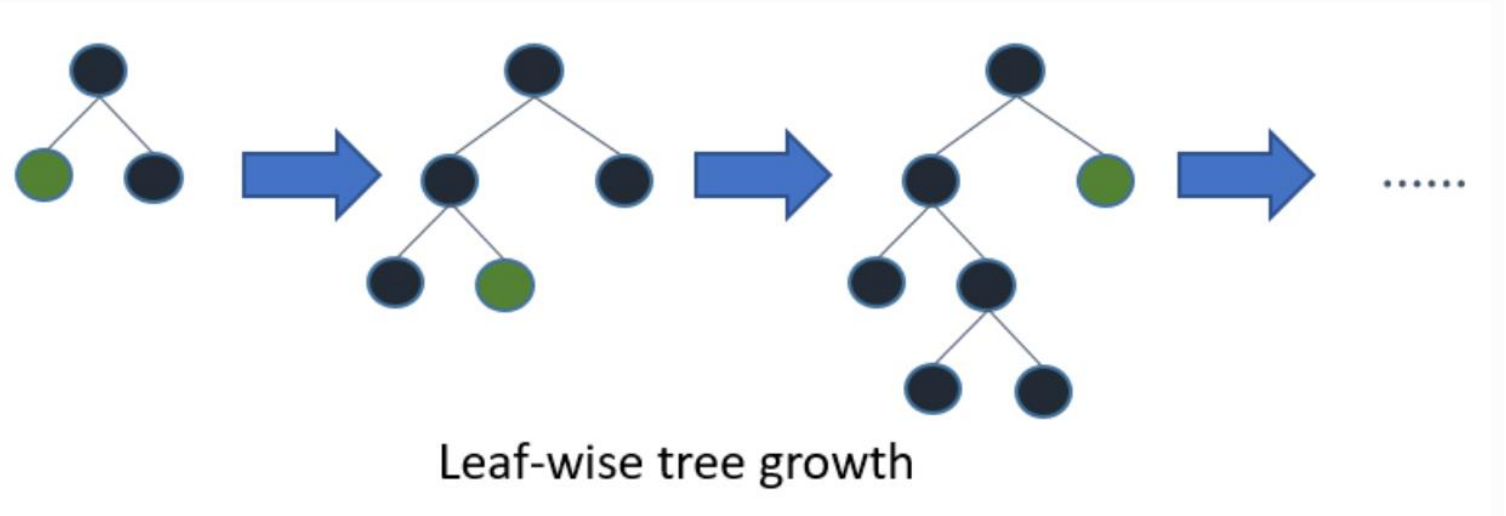
— имеет встроенную кросс-валидацию

— имеет детекцию переобучения

— имеет встроенное вычисление значений метрик

LightGBM (Light Gradient-Boosting Machine)

Строит деревья, добавляя на каждом шаге один лист:

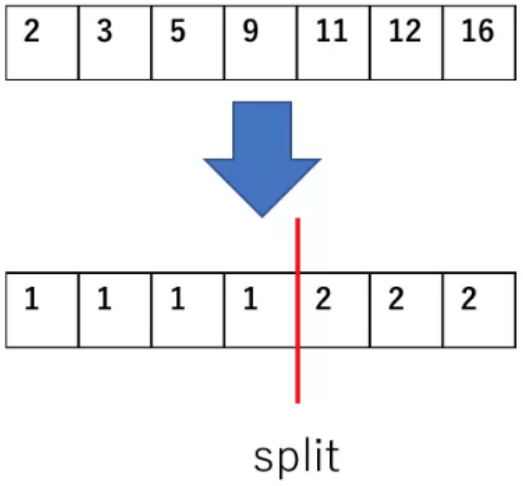


Источник: https://lightgbm.readthedocs.io/en/latest/Features.html

Такой подход позволяет добиться более высокой точности решения задач оптимизации

LightGBM (Light Gradient-Boosting Machine)

Ускоряет построение деревьев за счет биннаризации признаков



Источник: https://towardsdatascience.com/

LightGBM (Light Gradient-Boosting Machine)

Ключевые особенности:

— как правило деревья решений имеют несимметричную форму

— хорошо оптимизирован и не требует большого количества вычислительных ресурсов

— может работать с категориальными признаками и делает их разбиение на два подмножества достаточно эффективно за $O(k * \log k)$ операций

Реализации градиентного бустинга

	XGBoost	LightGBM	CatBoost
Общие характеристики	1. Алгоритм градиентного бустинга на деревьях решений 2. Жадные алгоритмы		
Преимущества	1. Параллельные вычисления 2. Вторая производная функции потерь 3. Поддержка линейных моделей 4. Регуляризация 5. Фактор усадки (shrinkage) 6. Выбор столбцов 7. Обработка пропусков	1. Алгоритм на основе гистограмм 2. Градиентный односторонний отбор (GOSS) 3. Эксклюзивное объединение признаков (EFB) 4. Алгоритм построения листьев 5. Параллельное ускорение обработки данных 6. Оптимизация кеша	1. Сокращает необходимость подбора гиперпараметров 2. Интерфейс для интеграции с scikit, а также с R и командной строкой 3. Поддержка категориальных признаков 4. Самообучающиеся функции потерь 5. Масштабируемая версия для GPU 6. Эффективное развёртывание
Недостатки	1. Нужно проходить весь датасет при разбиении узлов 2. Высокая сложность предварительной сортировки	1. Может создавать более глубокие деревья при построении по листьям, что вызывает переобучение 2. Чувствительность к шуму	1. Обработка категориальных признаков требует много памяти и времени 2. Настройки генератора случайных чисел влияют на конечный результат
Гиперпараметры	learning_rate, max_depth, min_child_weight, colsample_bytree, subsample, n_estimators	learning_rate, max_depth, min_data_in_leaf, categorical_feature, feature_fraction, bagging_fraction, num_iterations	learning_rate, depth, l2_reg_leaf, cat_features, one_hot_max_size, rsm, iterations

Источник: https://www.researchgate.net/figure/The-main-characteristics-and-important-parameters-of-XGBoost-LightGBM-and-CatBoost_tbl1_365435743

Реализации градиентного бустинга

Характеристика	XGBoost	LightGBM	CatBoost
Основное преимущество	Регуляризация, сильные значения по умолчанию	Очень быстрый на больших датасетах, экономит память	Нативная работа с категориальными признаками, меньше настройки
Построение дерева	По уровням (сбалансированный)	По листьям (быстрее, риск переобучения)	Симметричные разбиения через ordered boosting
Работа с категориальными признаками	Требуется кодирование	Нужно вручную указывать категориальные столбцы	Автоматически обрабатывает категориальные признаки через target encoding
Скорость и использование памяти	Быстрый, использует больше памяти	Чрезвычайно быстрый и экономичный по памяти	Нормальная скорость, отлично работает с данными с большим числом категорий
Лучший сценарий применения	Малые и средние числовые датасеты	Критична высокая скорость, большие табличные данные	Смешанные типы данных, много категорий, автономная стабильная работа

Источник: https://medium.com/codex/gradient-boosting-xgboost-lightgbm-and-catboost-explained-3b90a753bf98

Документация

- XGBoost
- CatBoost
- LightGBM