Градиентный бустинг: имплементации.

Повторение

Идея бустинга

- Бустинг (англ. boosting — усиление)
- Возьмём простые базовые модели
- Будем строить композицию последовательно и жадно
- Каждая следующая модель будет строиться так, чтобы максимально корректировать ошибки построенных моделей

Идея бустинга

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

- Обучение N-й модели (b_N) :

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i, a_{N-1}(x_i) + b_N(x_i)\big) \to \min_{b_N(x)}$$

- Пытаемся подобрать модель b_N, минимизирующую ошибку итоговой композиции $a_N = a_{N-1} + b_N$

Идея бустинга

- Обучение N-й модели (b_N) :

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\left(y_i, a_{N-1}(x_i) + b_N(x_i)\right) \to \min_{b_N(x)}$$

- Нет такого алгоритма машинного обучения, который учил бы «добавку»

- Попробуем понять, как можно сформулировать задачу с точки зрения ML

Бустинг для MSE

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

- Обучение N-й модели:

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\Big(b_N(x_i) - \underbrace{\big(y_i - a_{N-1}(x_i)\big)}_{s_i^{(N)}}\Big)^2 \to \min_{b_N(x)}$$

$s_i^{(N)} = y_i - a_{N-1}(x_i)$ — остатки

Бустинг для MSE

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\left(b_N(x_i) - s_i^{(N)}\right)^2 \to \min_{b_N(x)}$$

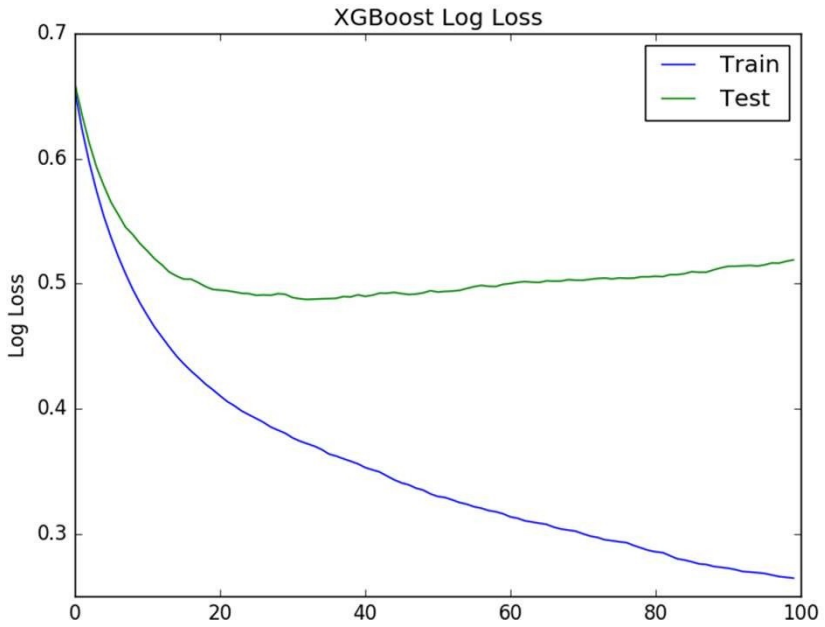
- $s_i^{(N)} = y_i - a_{N-1}(x_i)$ — остатки
- Если b_N научится выдавать остатки $s_i^{(N)}$, то задача будет решена идеально

$$y_i = a_{N-1}(x_i) + s_i^{(N)} = a_{N-1}(x_i) + b_N(x_i)$$

Третья итерация

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\Big(b_3(x_i)-\big(y_i-b_1(x_i)-b_2(x_i)\big)\Big)^2\to\min_{b_3(x)}$$

Ошибка бустинга на обучении и тесте



Задача обучения базовой модели

- Может, просто обучаться на остатки, как в MSE?

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i - a_{N-1}(x_i), b_N(x_i)\big) \to \min_{b_N(x)}$$

- Хотим, чтобы модель b_N выдавала $y_i - a_{N-1}(x_i)$

Логистическая функция потерь

$$a_N(x) = \text{sign} \sum_{n=1}^{N} b_n(x)$$

$$L(y,z) = \log(1 + \exp(-yz))$$

- Может, просто обучаться на остатки, как в MSE?

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \log\left(1 + \exp\left(-(y_i - a_{N-1}(x_i))b_N(x_i)\right)\right) \to \min_{b_N(x)}$$

- Если $y_i = a_{N-1}(x_i)$, то объект не участвует в обучении
- Тогда модель $b_N(x_i)$ может выдавать что угодно и испортить композицию

MSLE

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

$$L(y, z) = (\log(z + 1) - \log(y + 1))^2$$

- Может, просто обучаться на остатки, как в MSE?

$$\frac{1}{\ell}\sum_{i=1}^{\ell}(\log(b_N(x_i) + 1) - \log(y_i - a_{N-1}(x_i) + 1))^2 \to \min_{b_N(x)}$$

- Аргумент второго логарифма может оказаться отрицательным

Задача обучения базовой модели

- Обучение N-й модели:

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i, a_{N-1}(x_i) + b_N(x_i)\big) \to \min_{b_N(x)}$$

- Как посчитать, куда и как сильно сдвигать $a_{N-1}(x_i)$, чтобы уменьшить ошибку?

- Посчитать производную

Задача обучения базовой модели

- Обучение N-й модели:

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i, a_{N-1}(x_i) + b_N(x_i)\big) \to \min_{b_N(x)}$$

- Посчитаем антипроизводную:

$$s_i^{(N)} = -\frac{\partial}{\partial z} L(y_i, z)\Big|_{z=a_{N-1}(x_i)}$$

правильный ответ

производная по z

прогноз модели

в качестве z подставляем в результат предсказание композиции

Задача обучения базовой модели

- Посчитаем антипроизводную:

$$s_i^{(N)} = -\frac{\partial}{\partial z} L(y_i, z)\Big|_{z=a_{N-1}(x_i)}$$

- Знак показывает, в какую сторону сдвигать прогноз на x_i, чтобы уменьшить ошибку композиции на нём
- Величина показывает, как сильно можно уменьшить ошибку, если сдвинуть прогноз
- Если ошибка почти не сдвинется, то нет смысла что-то менять

Градиентный бустинг

- Обучение N-й модели:

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\left(b_N(x_i) - s_i^{(N)}\right)^2 \to \min_{b_N(x)}$$

$s_i^{(N)} = -\frac{\partial}{\partial z}L(y_i, z)\Big|_{z=a_{N-1}(x_i)}$ — сдвиги

- Таким образом, мы обучаем базовую модель b_N так, чтобы на x_i она выдавала $s_i^{(N)}$

Градиентный бустинг

- Обучение N-й модели:

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\left(b_N(x_i) - s_i^{(N)}\right)^2 \to \min_{b_N(x)}$$

$s_i^{(N)} = -\frac{6}{6z} L(y_i, z)\Big|_{z=a_{N-1}(x_i)}$ — сдвиги

- Как бы градиентный спуск в пространстве алгоритмов
- Базовая модель будет делать корректировки на объектах так, чтобы как можно сильнее уменьшить ошибку композиции
- Сдвиги учитывают особенности функции потерь

Глубина деревьев

- Градиентный бустинг уменьшает смещение базовых моделей
- Разброс может увеличиться
- Поэтому в качестве базовых моделей стоит брать **неглубокие** деревья

Проблемы бустинга

- Сдвиги показывают направление, в котором надо сдвинуть композицию на всех объектах обучающей выборки
- Базовые модели, как правило, очень простые
- Могут не справиться с приближением этого направления
- Выход: добавлять деревья в композицию с небольшим весом

Длина шага

$$a_N(x) = a_{N-1}(x_i) + \eta b_N(x_i)$$

- $\eta \in (0, 1]$ — длина шага (learning rate)
- Можно сказать, что это регуляризация композиции
- Снижает вклад каждой модели в композицию
- Чем меньше η, тем больше надо деревьев

Рандомизация

- Можно обучать деревья на случайных подмножествах признаков
- Бустинг уменьшает смещение, поэтому итоговая композиция всё равно получится качественной
- Может снизить переобучение
- Можно обучать деревья на подмножествах объектов — способ борьбы с шумом в данных

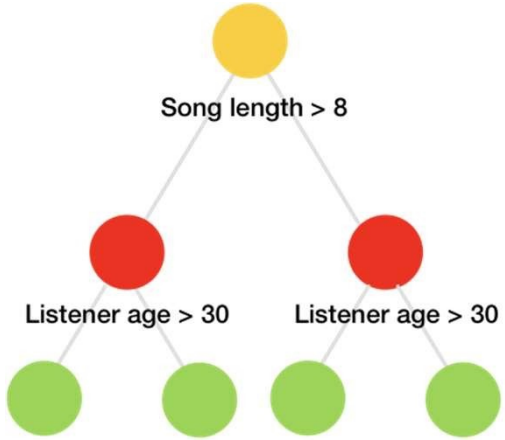
Гиперпараметры

- Глубина базовых деревьев
- Число деревьев
- Длина шага
- Размер подвыборки для обучения
- и т.д.

Вариации бустинга

ODT

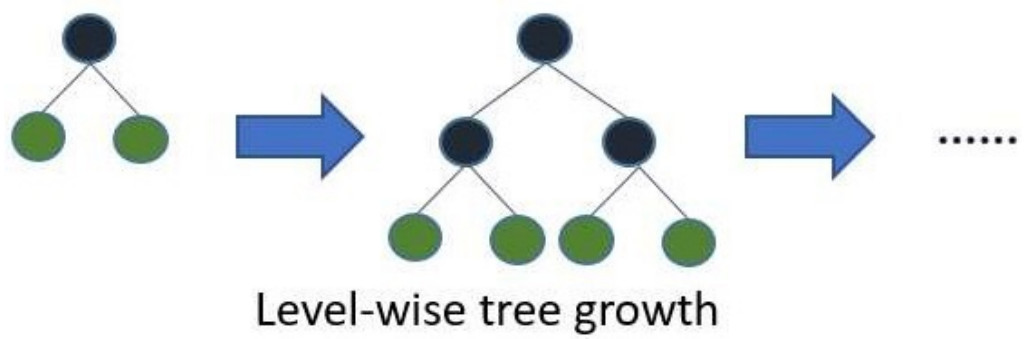
- Oblivious decision trees
- Ограничение: на одном уровне дерева используется один и тот же предикат



https://catboost.ai/

Способ построения дерева

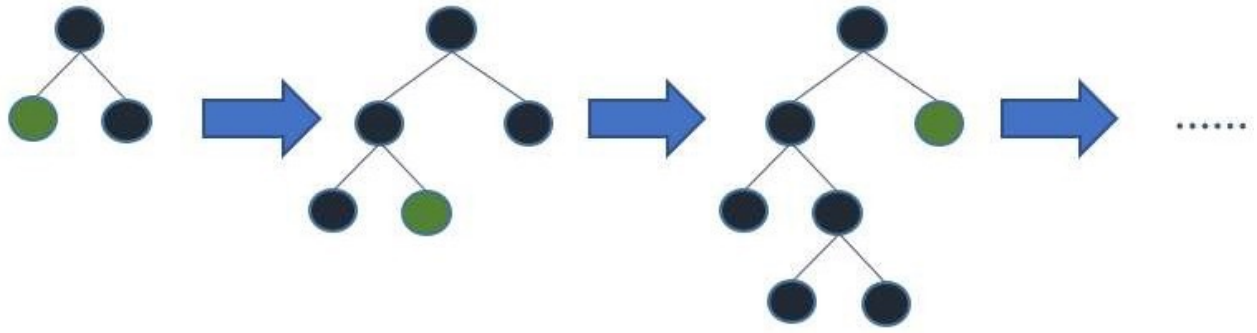
- Level-wise: дерево строится рекурсивно до тех пор, пока не достигнута максимальная глубина



https://lightgbm.readthedocs.io/

Способ построения дерева

- Level-wise: дерево строится рекурсивно до тех пор, пока не достигнута максимальная глубина
- Leaf-wise: среди текущих листьев выбирается тот, чьё разбиение сильнее всего уменьшает ошибку



Leaf-wise tree growth

https://lightgbm.readthedocs.io/

Выбор лучшего порога для предиката

- $[x_j < t]$ — как выбрать t?
- Вариант 1: перебрать все известные значения признака
- Вариант 2: построить гистограмму для признака и искать пороги среди границ на гистограмме
- Вариант 3: случайно выбрать объекты с близкими к нулю значениями производной функции потерь

Регуляризация деревьев

- Базовая регуляризация: введение длины шага и количества выбираемых признаков
- Штрафы за число листьев в дереве
- Штрафы за величину прогнозов в листьях дерева

Улучшенное обучение

- Мы обучаем деревья на сдвиги, ошибка измеряется с помощью MSE
- Когда дерево построено, можно подобрать оптимальные значения в листьях с точки зрения исходной функции потерь

Имплементации

- XGBoost
- LightGBM: leaf-wise growth, поиск порогов на основе производных
- CatBoost: ODT

РАБОТА С ПРИЗНАКАМИ

КОДИРОВАНИЕ КАТЕГОРИАЛЬНЫХ ПРИЗНАКОВ: ONE-HOT ENCODING

- Предположим, категориальный признак $f_j(x)$ принимает m различных значений: $C_1, C_2, \ldots, C_m$.

Пример: еда может быть *горькой, сладкой, солёной или кислой* (4 возможных значения признака).

КОДИРОВАНИЕ КАТЕГОРИАЛЬНЫХ ПРИЗНАКОВ: ONE-HOT ENCODING

- Предположим, категориальный признак $f_j(x)$ принимает m различных значений: $C_1, C_2, \ldots, C_m$.

Пример: еда может быть *горькой, сладкой, солёной или кислой* (4 возможных значения признака).

- Заменим категориальный признак на m бинарных признаков: $b_i(x) = [f_j(x) = C_i]$ (индикатор события).

Тогда One-Hot кодировка для нашего примера будет следующей:

горький = *(1,0,0,0)*, *сладкий* = *(0,1,0,0)*,

солёный = *(0,0,1,0)*, *кислый* = *(0,0,0,1)*.

ХЭШИРОВАНИЕ ПРИЗНАКОВ

- Возьмем некоторую функцию (hash-функция), которая переводит значения категориального признака в числа от *1* до *B*: $h: U \rightarrow \{1,2,\dots,B\}$.
- То есть для каждого объекта:

$$g_j(x) = \left[h\left(f(x)\right) = j\right], j = 1,\dots,B$$

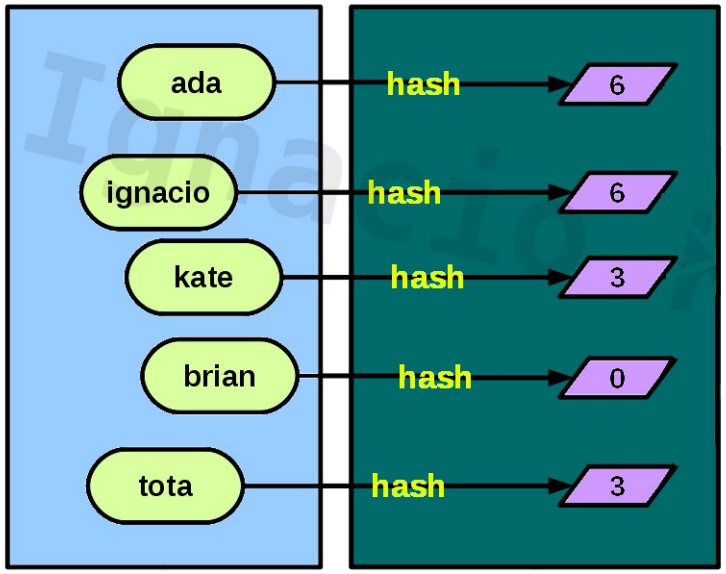
ХЭШИРОВАНИЕ ПРИЗНАКОВ

- Возьмем некоторую функцию (hash-функция), которая переводит значения категориального признака в числа от *1* до *B*: $h: U \rightarrow \{1,2,\dots,B\}$.
- То есть для каждого объекта:

$$g_j(x) = [h(f(x)) = j], j = 1,\dots,B$$

Идея: *хэширование группирует значения признака. Так как часто встречающихся значений немного, они редко попадают в одну группу при группировке.*

ХЭШИРОВАНИЕ ПРИЗНАКОВ



0	1	2	3	4	5	6
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	1	0	0	0
1	0	0	0	0	0	0
0	0	0	1	0	0	0

ХЭШИРОВАНИЕ ПРИЗНАКОВ

- Возьмем некоторую функцию (hash-функция), которая переводит значения категориального признака в числа от *1* до *B*: $h: U \rightarrow \{1,2, \dots, B\}$.
- То есть для каждого объекта:

$$g_j(x) = \left[h\left(f(x)\right) = j\right], j = 1, \dots, B$$

+ позволяет закодировать любое значение категориального признака (в том числе, то, которого не было в тренировочной выборке)

ХЭШИРОВАНИЕ

- Хороший способ работать с категориальными данными, принимающими множество различных значений
- Хорошие результаты на практике
- Позволяет понизить размерность пространства признаков с незначительным снижением качества

Статья про хэширование:

https://arxiv.org/abs/1509.05472

СЧЕТЧИКИ

- Пусть целевая переменная y принимает значения от 1 до K.
- Закодируем категориальную переменную $f(x)$ следующим способом:

$$counts(u, X) = \sum_{(x,y) \in X} [f(x) = u]$$

$$successes_k(u, X) = \sum_{(x,y) \in X} [f(x) = u][y = k], k = 1, \ldots, K$$

СЧЁТЧИКИ: ПРИМЕР

city	target	0	1	2
Moscow	1	1/4	1/2	1/4
London	0	1/2	0	1/2
London	2	1/2	0	1/2
Kiev	1	1/2	1/2	0
Moscow	1	1/4	1/2	1/4
Moscow	0	1/4	1/2	1/4
Kiev	0	1/2	1/2	0
Moscow	2	1/4	1/2	1/4

СЧЕТЧИКИ

- Пусть целевая переменная y принимает значения от 1 до K.
- Закодируем категориальную переменную $f(x)$ следующим способом:

$$counts(u, X) = \sum_{(x,y)\in X} [f(x) = u]$$

$$successes_k(u, X) = \sum_{(x,y)\in X} [f(x) = u][y = k], k = 1, \dots, K$$

Тогда кодировка:

$$\boldsymbol{g_k(x, X) = \frac{successes_k(f(x), X) + c_k}{counts(f(x), X) + \sigma_{i=1}^{K} c_i} \approx p(y = k|f(x))},$$

c_i - чтобы не было деления на 0.

СЧЁТЧИКИ: ОПАСНОСТЬ ПЕРЕОБУЧЕНИЯ

Вычисляя счётчики, мы закладываем в признаки информацию о целевой переменной и, тем самым, переобучаемся!

СЧЁТЧИКИ: КАК ВЫЧИСЛЯТЬ

- Можно вычислять счётчики так:

city	target
Moscow	1
London	0
London	2
Kiev	1
Moscow	1
Moscow	0
Kiev	0
Moscow	2

Вычисляем счетчики по этой части

Кодируем признак вычисленными счётчиками и обучаемся по этой части

СЧЁТЧИКИ: КАК ВЫЧИСЛЯТЬ

Более продвинутый способ (по кросс-валидации):

1) Разбиваем выборку

на m частей $X_1, \dots, X_m$

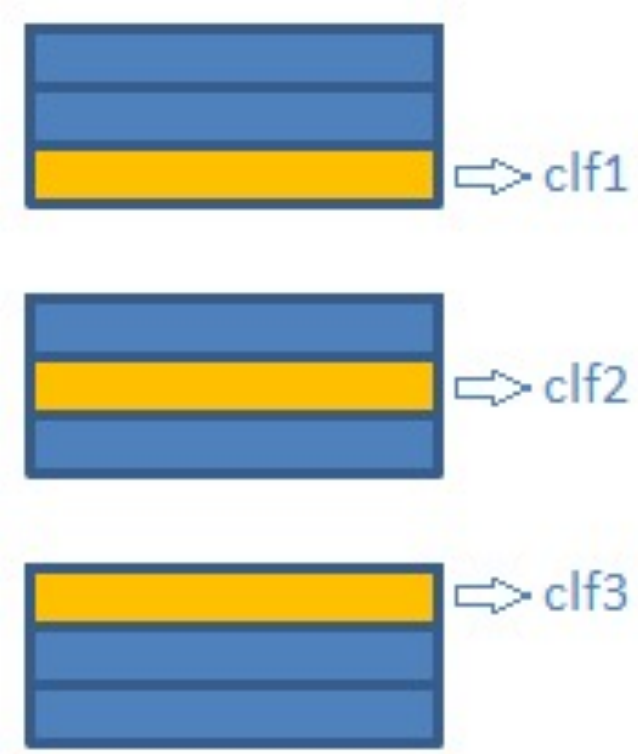
2)На каждой части X_i

значения признаков

вычисляются по

оставшимся частям:

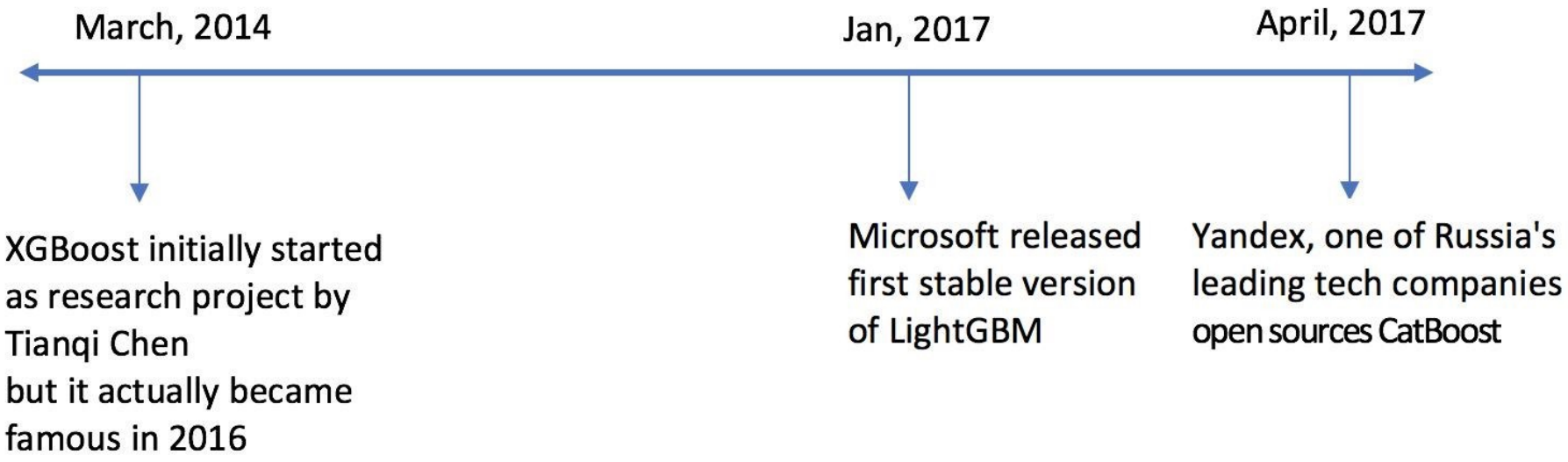
$x \in X_i \Rightarrow g_k(x) = g_k(x, X \backslash X_i)$



РЕАЛИЗАЦИИ ГРАДИЕНТНОГО БУСТИНГА:

- Xgboost
- CatBoost
- LightGBM

XGBOOST, LIGHTGBM, CATBOOST



- https://github.com/dmlc/xgboost
- https://github.com/Microsoft/LightGBM
- https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

XGBOOST (EXTREME GRADIENT BOOSTING)

- На каждом шаге градиентного бустинга решается задача

$$\sum_{i=1}^{l} (b(x_i) - s_i)^2 \to \min_b$$

$$\sum_{i=1}^{l} \left(-s_i b(x_i) + \frac{1}{2} b^2(x_i) \right)^2 \to \min_b$$

XGBOOST (EXTREME GRADIENT BOOSTING)

- На каждом шаге xgboost решается задача

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

$$h_i = \frac{\partial^2 L}{\partial z^2}\Phi_{a_{N-1}(x_i)}$$

XGBOOST

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

Основные особенности xgboost:

- 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 при обучении дерева также зависит от оптимального сдвига

CATBOOST

CatBoost – алгоритм, разработанный в Яндексе. Он является оптимизацией Xgboost и в отличие от Xgboost умеет обрабатывать категориальные признаки.

https://github.com/catboost/catboost

CATBOOST

Особенности catboost:

- используются симметричные деревья решений

CATBOOST

Особенности catboost:

- Для кодирования категориальных признаков используется набор методов (one-hot encoding, счётчики, комбинации признаков и др.)

CATBOOST

Особенности catboost:

- динамический бустинг

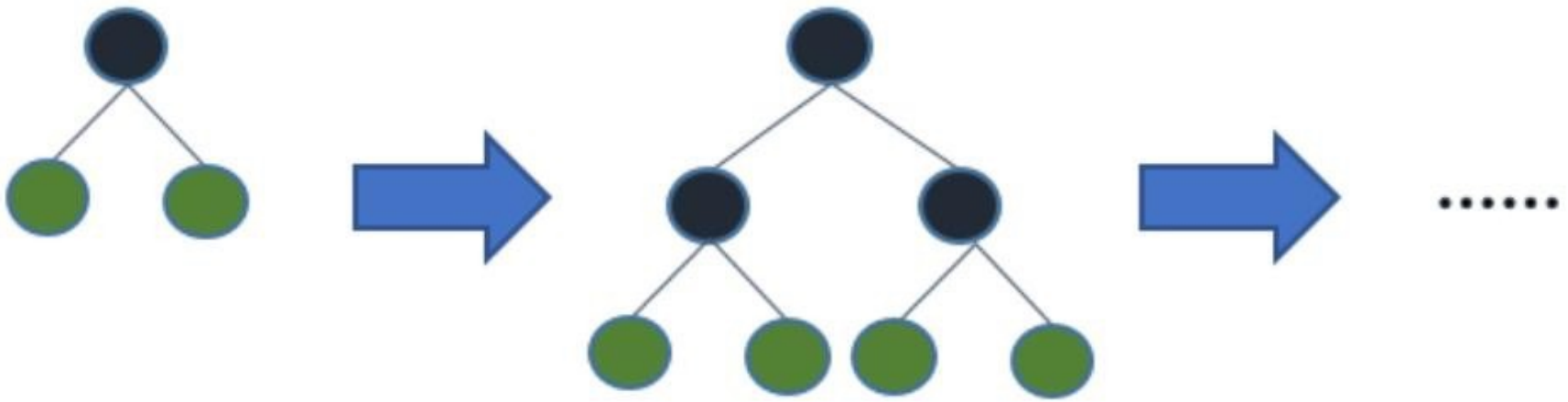
CATBOOST

Бонусы реализации:

- Поддержка пропусков в данных
- Обучается быстрее, чем xgboost
- Показывает хороший результат даже без подбора параметров
- Удобные методы: проверка на переобученность, вычисление значений метрик, удобная кросс-валидация и др.

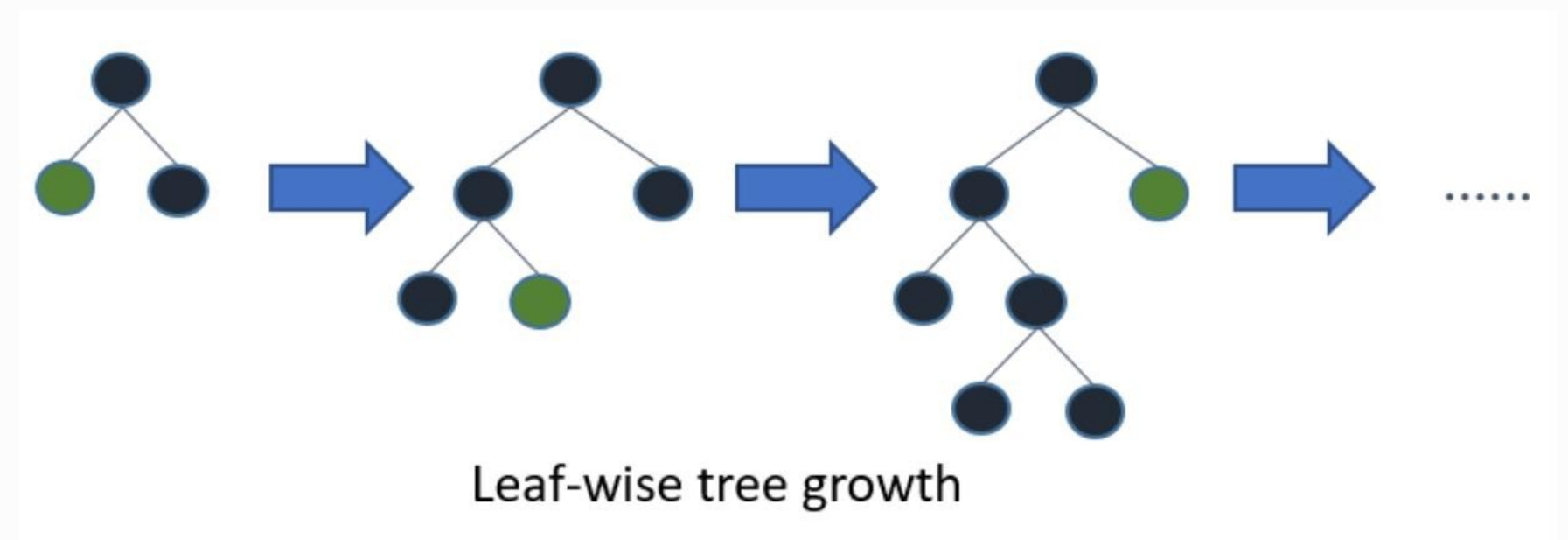
LIGHTGBM

В других реализациях градиентного бустинга деревья строятся по уровням:



LIGHTGBM

LightGBM строит деревья, добавляя на каждом шаге один лист:



Такой подход позволяет добиться более высокой точности решения задачи оптимизации.

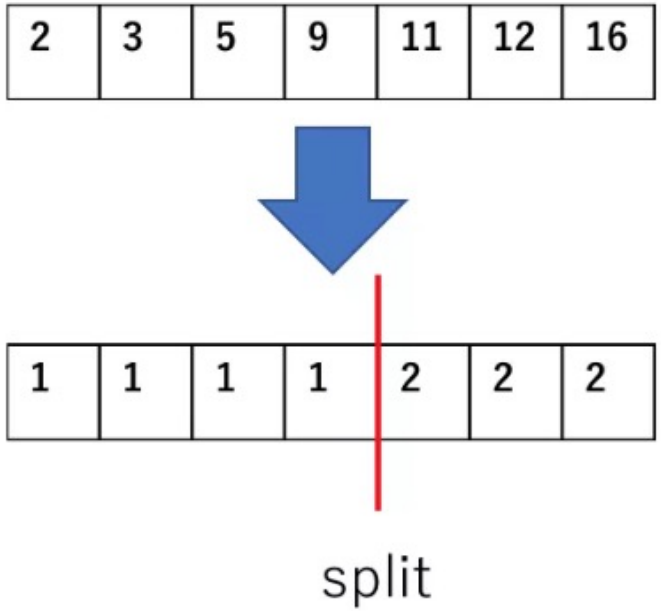
LIGHTGBM

Кодирование категориальных признаков.

- LightGBM разбивает значения категориального признака на два подмножества в каждой вершине дерева, находя при этом наилучшее разбиение
- Если категориальный признак имеет k различных значений, то возможных разбиений $2^{k-1} - 1$. В LightGBM реализован способ поиска оптимального разбиения за $O(klogk)$ операций.

LIGHTGBM

Ускорение построения деревьев за счёт бинаризации признаков:



An example of how binning can reduce the number of splits to explore. The features must be sorted in advance for this method to be effective.

Спасибо за внимание!

Ildar Safilo

@Ildar_Saf
irsafilo@gmail.com
https://www.linkedin.com/in/isafilo/