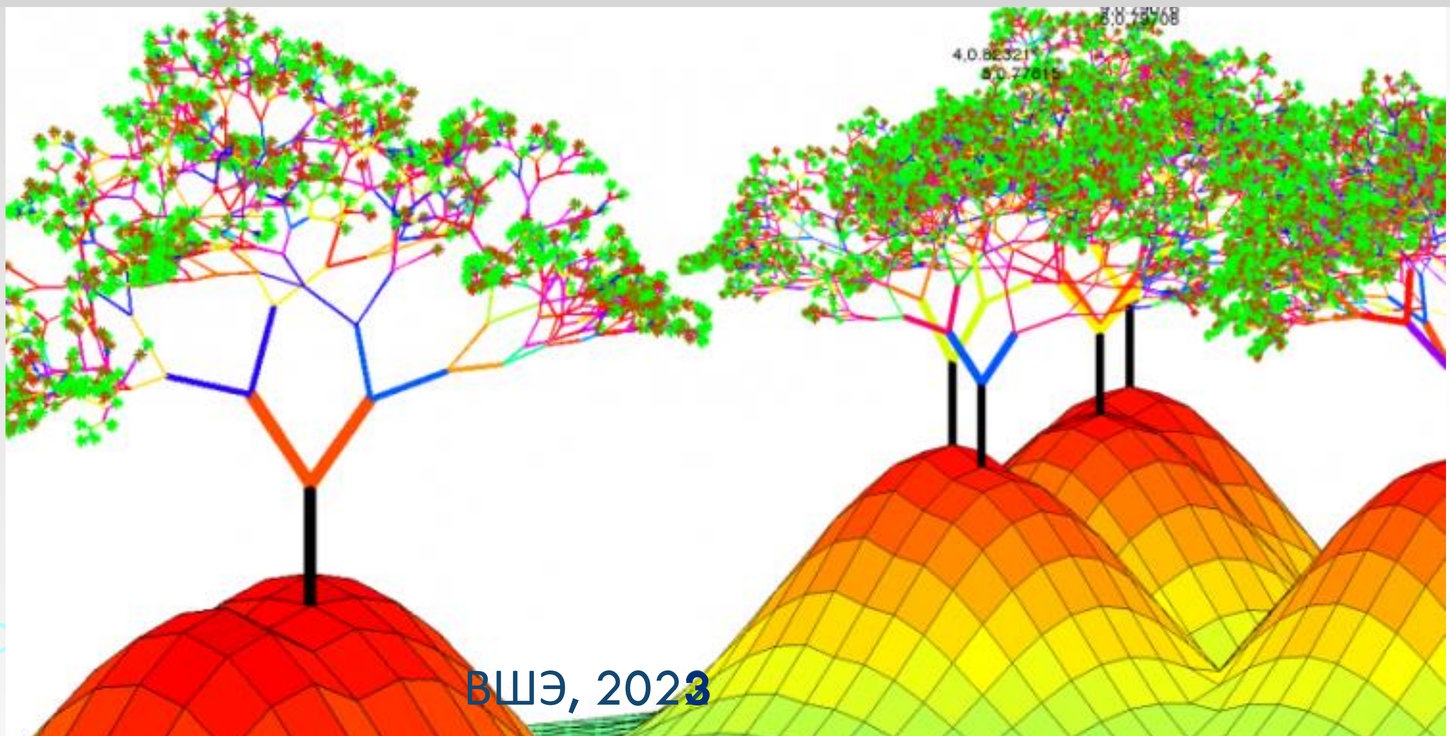
Бустинг.

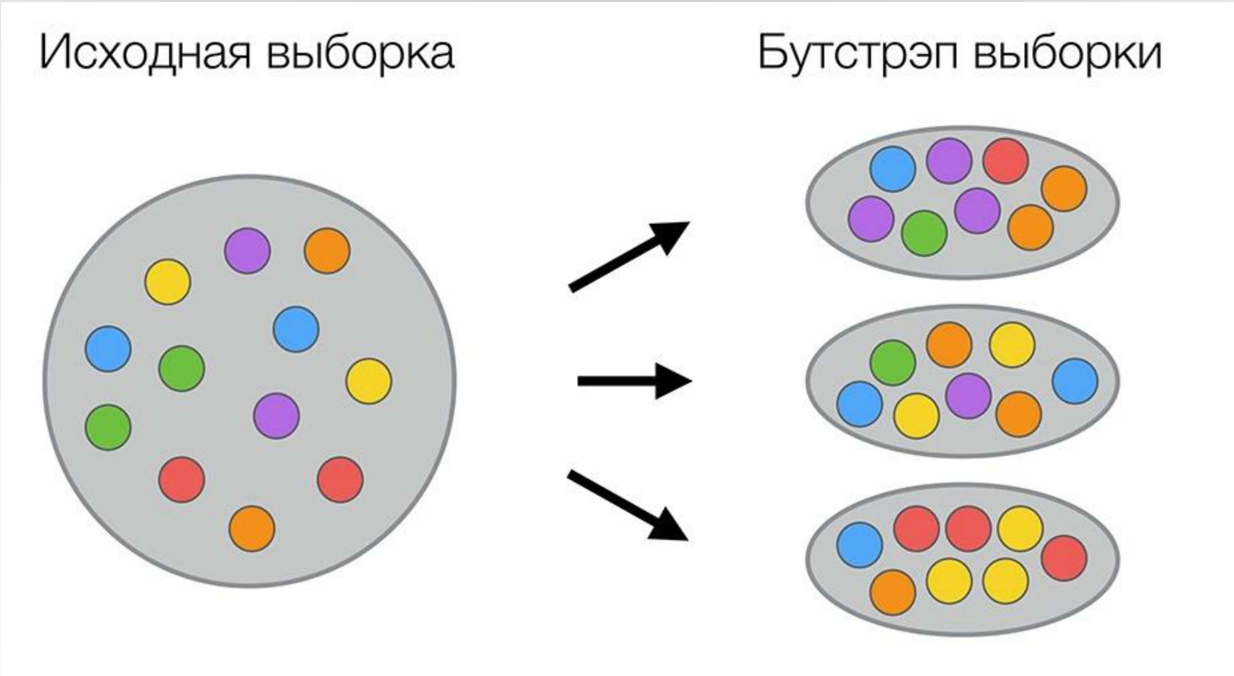


БУТСТРЭП

Дана выборка X.

Бутстрэп: равномерно возьмем из выборки X l объектов с возвращением (т.е. в новой выборке будут повторяющиеся объекты). Получим выборку X_1.

- Повторяем процедуру N раз, получаем выборки $X_1, \dots, X_N$.



БЭГГИНГ (BOOTSTRAP AGGREGATION)

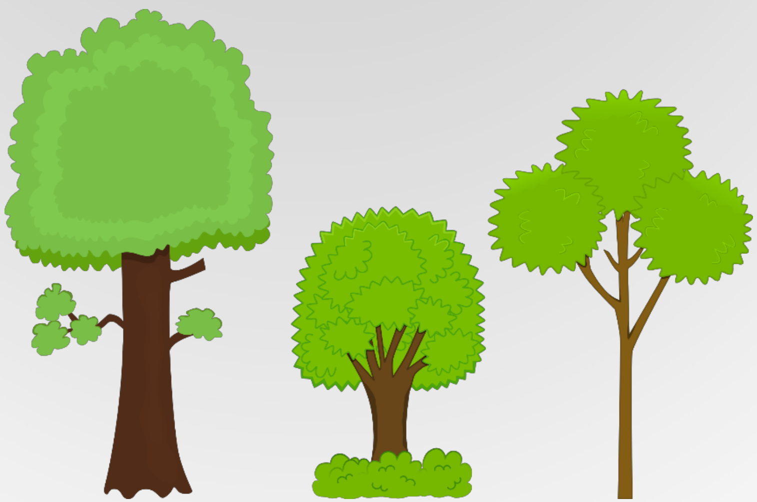
С помощью бутстрэпа мы получили выборки $X_1, \dots, X_N$.

- Обучим по каждой из них модель – получим базовые алгоритмы $b_1(x), \dots, b_N(x)$.
- Построим новую функцию регрессии:

$$\boldsymbol{a(x) = \frac{1}{N}\sum_{j=1}^{N} b_j(x)}$$

СЛУЧАЙНЫЙ ЛЕС (RANDOM FOREST)

- Возьмем в качестве базовых алгоритмов для бэггинга ***решающие деревья***, т.е. каждое случайное дерево $b_i(x)$ построено по своей бутстрепной подвыборке X_i.
- В каждой вершине дерева будем искать ***разбиение не по всем признакам, а по подмножеству признаков***.
- Итоговая композиция имеет вид $\boldsymbol{a(x) = \frac{1}{N}\sum_{j=1}^{N} b_j(x)}$.



Разложение ошибки (bias-variance decomposition)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

Утверждение: *ошибку модели $a(x)$ можно представить в виде*

$$\boxed{\mathbf{Err(x) = Bias^2\big(a(x)\big) + Var\big(a(x)\big) + \sigma^2}.}$$

- $\mathbf{Bias\big(a(x)\big)}$ - средняя ошибка по всем возможным наборам данных – **смещение**.

Смещение показывает, насколько в среднем модель хорошо предсказывает целевую переменную:

✓ ***маленькое смещение - хорошее предсказание***

✓ ***большое смещение – плохое предсказание***

РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

Утверждение: *ошибку модели $a(x)$ можно представить в виде*

$$\mathbf{Err(x) = Bias^2\big(a(x)\big) + Var\big(a(x)\big) + \sigma^2.}$$

- $\mathbf{Var\big(a(x)\big)}$ - дисперсия ошибки, т.е. как сильно различается ошибка при обучении на различных наборах данных – **разброс**.

Большой разброс означает, что ошибка очень чувствительна к изменению обучающей выборки, т.е.:

✓ ***большой разброс – сильно переобученная модель***

Разложение ошибки (bias-variance decomposition)

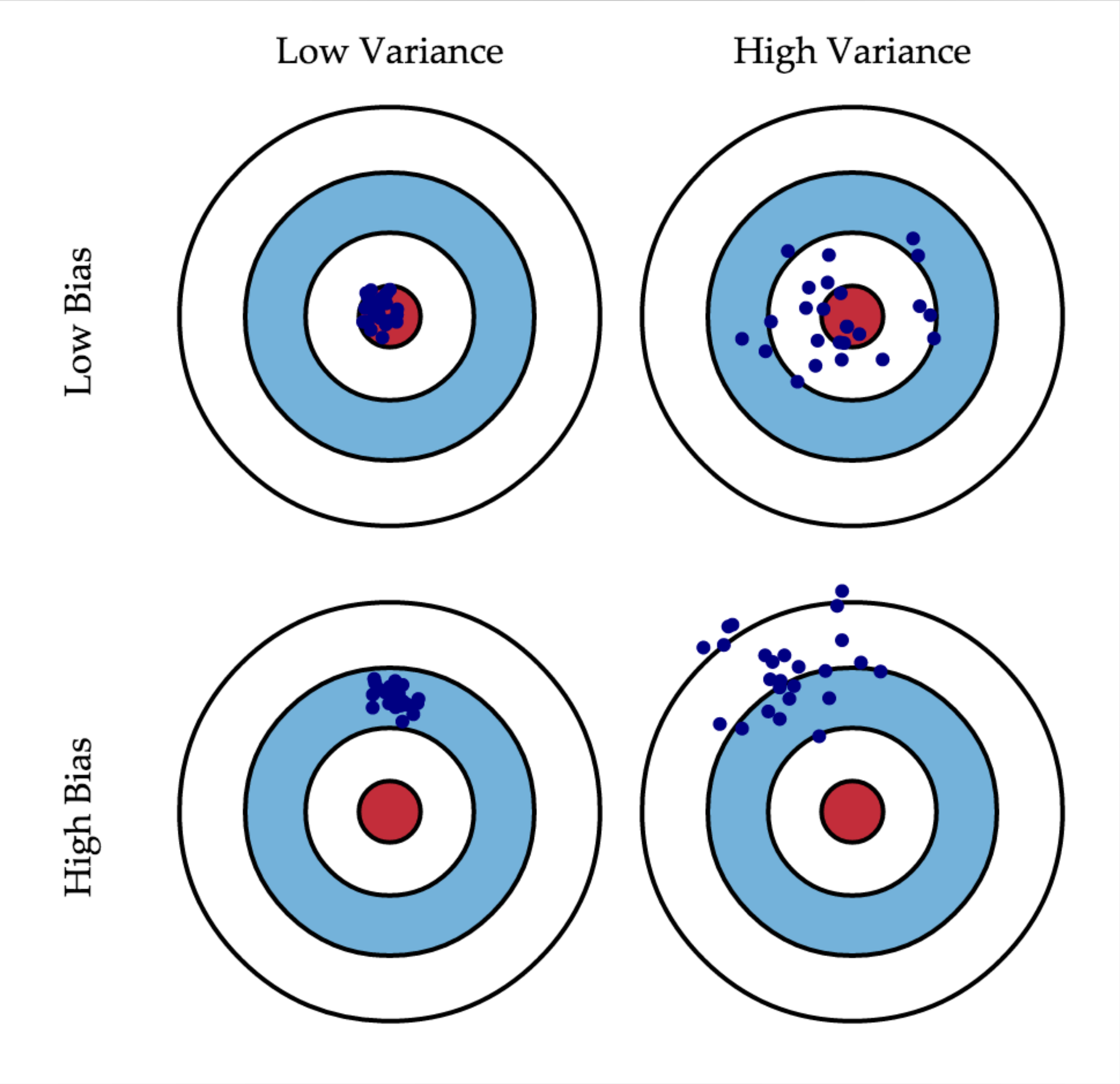
Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

Утверждение: *ошибку модели $a(x)$ можно представить в виде*

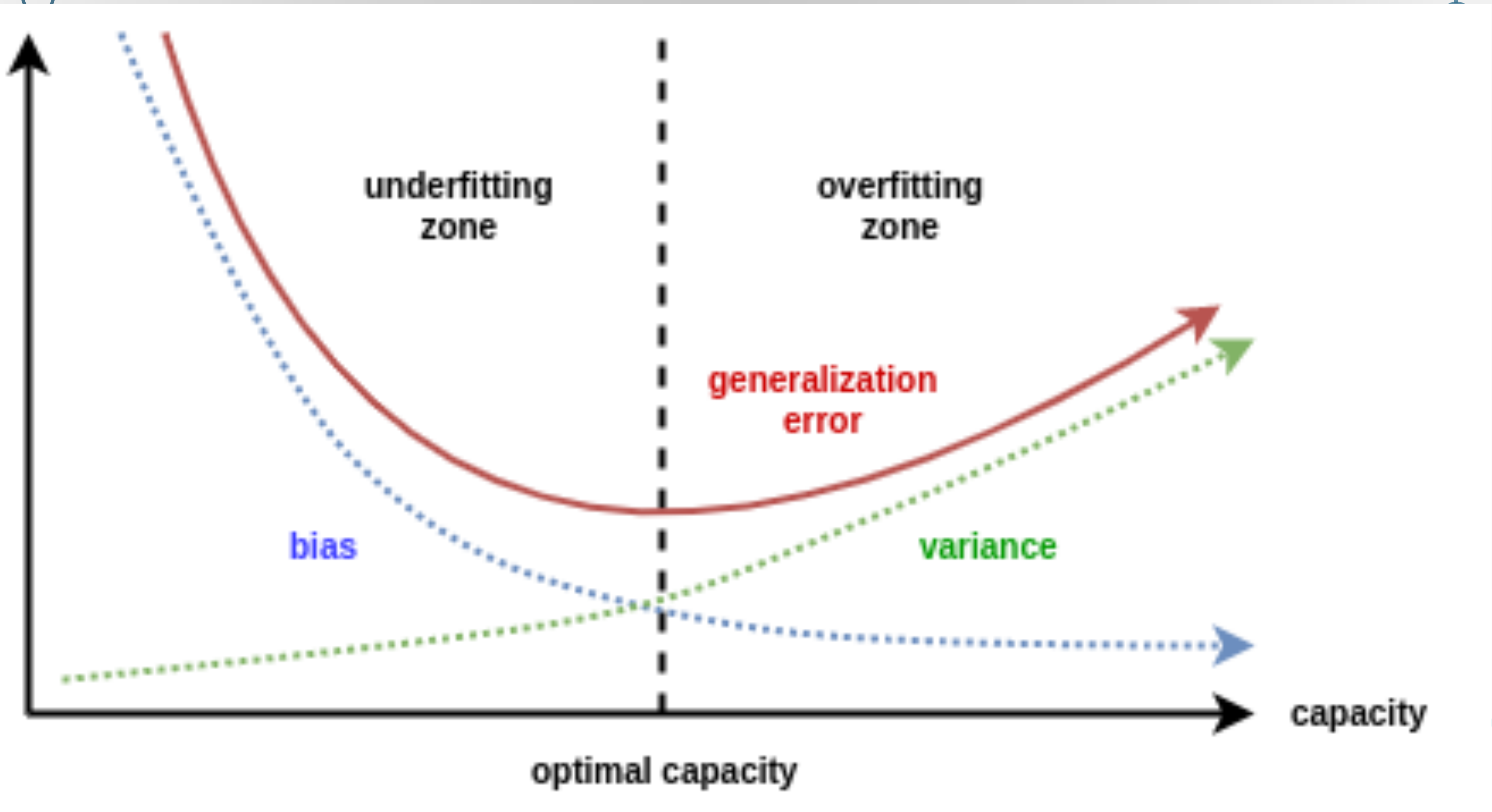
$$\mathbf{Err(x) = Bias^2\big(a(x)\big) + Var\big(a(x)\big) + \sigma^2}.$$

- $\mathbf{Bias\big(a(x)\big)}$ - средняя ошибка по всем возможным наборам данных – **смещение**.
- $\mathbf{Var\big(a(x)\big)}$ - дисперсия ошибки, т.е. как сильно различается ошибка при обучении на различных наборах данных – **разброс**.
- $\boldsymbol{\sigma}^{\mathbf{2}}$ - неустранимая ошибка – **шум**.

СМЕЩЕНИЕ И РАЗБРОС



BIAS-VARIANCE TRADEOFF



СМЕЩЕНИЕ И РАЗБРОС У БЭГГИНГА

Утверждение.

1) ***Бэггинг не ухудшает смещенность модели***, *т.е. смещение $a_N(x)$ равно смещению одного базового алгоритма.*

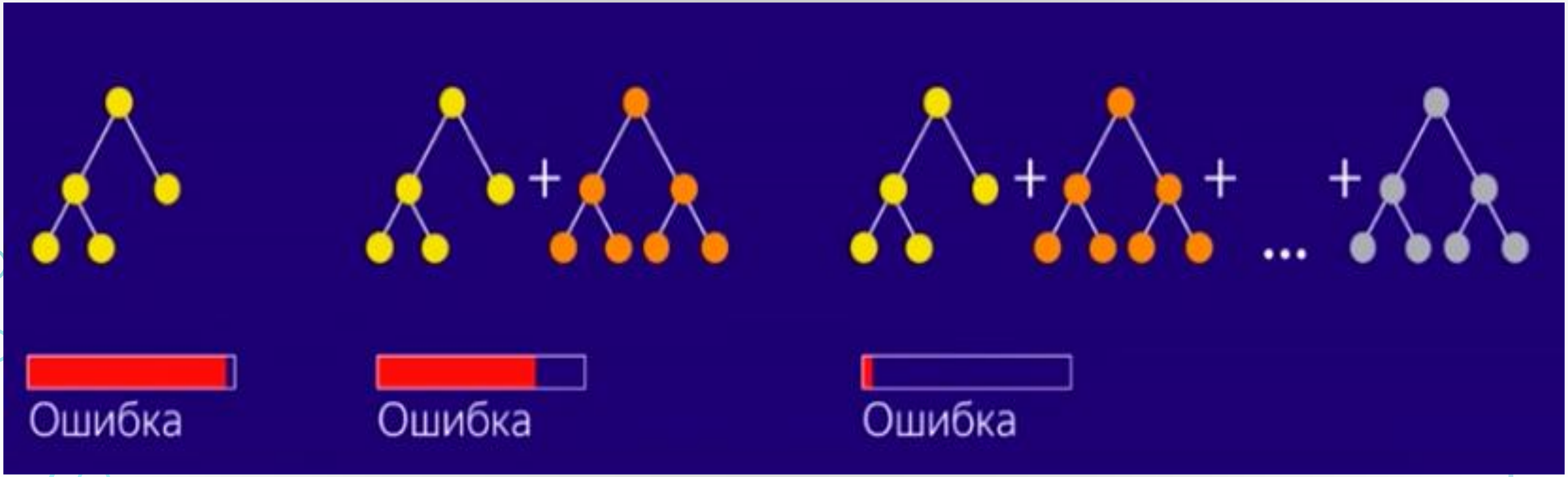
2) Если базовые алгоритмы некоррелированы, то ***дисперсия бэггинга $a_N(x)$ в N раз меньше дисперсии отдельных базовых алгоритмов.***

БУСТИНГ

<u>Идея</u>: строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих.

БУСТИНГ

Идея: строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих.



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 $b_n(x)$ принадлежат некоторому семейству A.

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

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

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

- Ошибка на объекте x:

$$s = y - b_1(x)$$

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 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:

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

Следующий алгоритм должен настраиваться на эту ошибку, т.е. ***целевая переменная для следующего алгоритма – это вектор ошибок $\boldsymbol{s}$*** (а не исходный вектор y)

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

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

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

Шаг 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

<u>Шаг 1:</u> Ищем алгоритм $b_1(x)$, минимизирующий ошибку:

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

<u>Шаг 2:</u> Ищем алгоритм $b_2(x)$, настраивающийся на ошибки s первого алгоритма:

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

Следующий алгоритм $\mathrm{b}_3(\mathrm{x})$ будем выбирать так, чтобы он минимизировал ошибку предыдущей композиции (т.е. $\mathrm{b}_1(\mathrm{x}) + \mathrm{b}_2(\mathrm{x})$) и т.д.

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Каждый следующий алгоритм настраиваем на ошибку предыдущих.

Шаг N: Ошибка: $\boldsymbol{s_i^{(N)}} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i)$

Ищем алгоритм $b_N(x)$:

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

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Каждый следующий алгоритм настраиваем на ошибку предыдущих.

Шаг N: Ошибка: $s_i^{(N)} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i)$

Ищем алгоритм $b_N(x)$:

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

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

$$s_i^{(N)} = y_i - a_{N-1}(x_i) = -\frac{\partial}{\partial z} \frac{1}{2} (z - y_i)^2 \Big|_{z = a_{N-1}(x_i)}$$

ГРАДИЕНТНЫЙ БУСТИНГ

Пусть $L(y, z)$ – произвольная дифференцируемая функция потерь. Строим алгоритм $a_N(x)$ вида

$$\boldsymbol{a_L(x) = \sum_{n=1}^{L} \gamma_n b_n(x)}$$

ГРАДИЕНТНЫЙ БУСТИНГ

Пусть $L(y,z)$ – произвольная дифференцируемая функция потерь. Строим алгоритм $a_N(x)$ вида

$$\boldsymbol{a_L(x) = \sum_{n=1}^{L} \gamma_n b_n(x)},$$

где на N-м шаге

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

$$\boldsymbol{s_i^{(N)} = y_i - a_{N-1}(x_i)}?$$

ГРАДИЕНТНЫЙ БУСТИНГ

Пусть $L(y, z)$ – произвольная дифференцируемая функция потерь. Строим алгоритм $a_N(x)$ вида

$$\boldsymbol{a_L(x) = \sum_{n=1}^{L} \gamma_n b_n(x)},$$

где на N-м шаге

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

$$\cancel{\boldsymbol{s_i^{(N)} = y_i - a_{N-1}(x_i)}} \qquad \boldsymbol{s_i^{(N)} = -\frac{\partial L}{\partial z}}$$

ГРАДИЕНТНЫЙ БУСТИНГ

Пусть $L(y, z)$ – произвольная дифференцируемая функция потерь.

Строим алгоритм $a_N(x)$ вида

$$\boldsymbol{a_L(x) = \sum_{n=1}^{L} \gamma_n b_n(x)},$$

где на N-м шаге

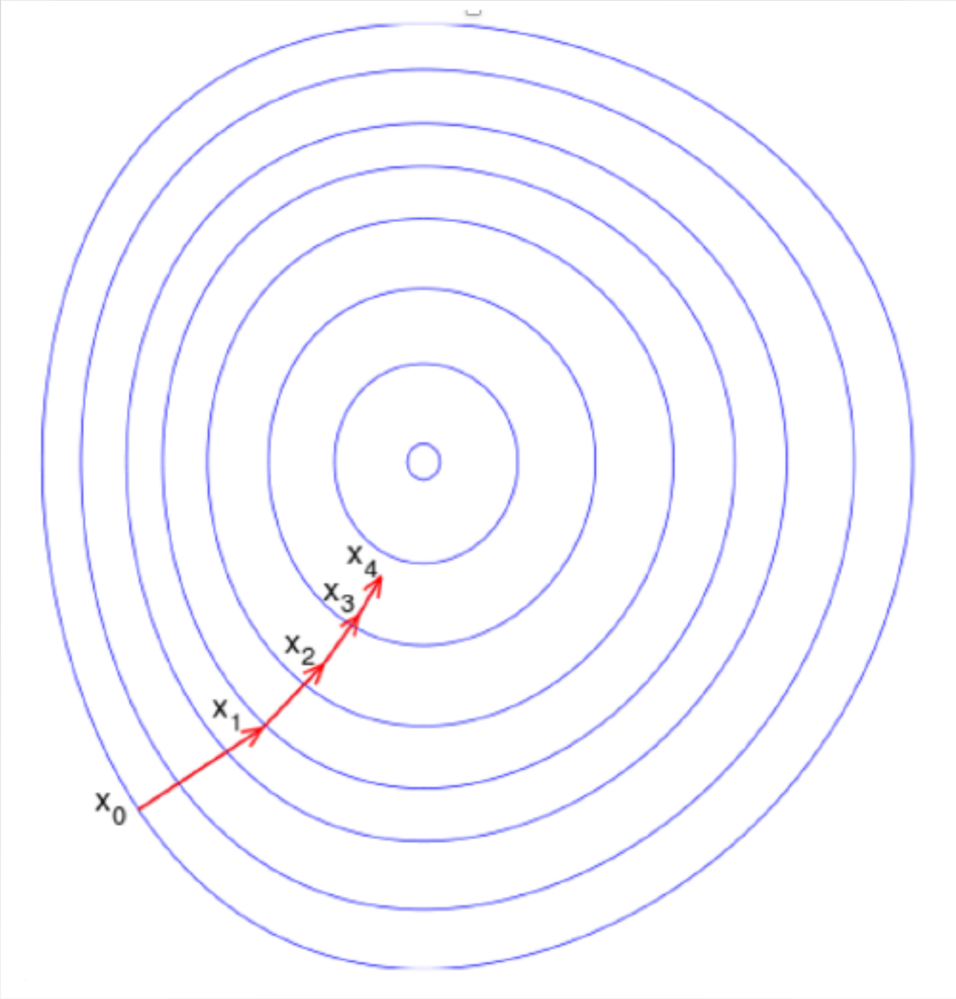
$$\boldsymbol{b_N(x) = \underset{b \in A}{\operatorname{argmin}} \sum_{i=1}^{l} \left(b(x_i) - s_i^{(N)}\right)^2},$$

$$\boldsymbol{s_i^{(N)} = -\frac{\partial L}{\partial z}}$$

Коэффициент γ_N должен минимизировать ошибку:

$$\boldsymbol{\gamma_N = \min_{\gamma \in \mathbb{R}} \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + \gamma_N b_N(x_i))}$$

ГРАДИЕНТНЫЙ СПУСК В ПРОСТРАНСТВЕ ФУНКЦИЙ



ВЫБОР БАЗОВЫХ АЛГОРИТМОВ

- *Что произойдет с предсказанием бустинга, если базовые алгоритмы слишком простые?*
- *Что будет, если базовые алгоритмы слишком сложные?*

Бустинг: выбор базовых алгоритмов

- Если базовые алгоритмы очень простые, то они плохо приближают антиградиент функции потерь, т.е. градиентный бустинг может свестись к случайному блужданию.
- Если базовые алгоритмы сложные, то за несколько шагов бустинг подгонится под обучающую выборку, и получим переобученный алгоритм.

БУСТИНГ: ВЫБОР БАЗОВЫХ АЛГОРИТМОВ

- Если базовые алгоритмы очень простые, то они плохо приближают антиградиент функции потерь, т.е. градиентный бустинг может свестись к случайному блужданию.
- Если базовые алгоритмы сложные, то за несколько шагов бустинг подгонится под обучающую выборку, и получим переобученный алгоритм.

Чаще всего в качестве базовых алгоритмов используют ***решающие деревья***.

БУСТИНГ: ВЫБОР БАЗОВЫХ АЛГОРИТМОВ

- Если базовые алгоритмы очень простые, то они плохо приближают антиградиент функции потерь, т.е. градиентный бустинг может свестись к случайному блужданию.
- Если базовые алгоритмы сложные, то за несколько шагов бустинг подгонится под обучающую выборку, и получим переобученный алгоритм.

Чаще всего в качестве базовых алгоритмов используют ***решающие деревья***.

В таком случае *решающие деревья не должны быть очень маленькими, а также очень глубокими.*

Оптимальная глубина – от 3 до 6 (зависит от задачи).

СОКРАЩЕНИЕ ШАГА (РЕГУЛЯРИЗАЦИЯ)

- Если базовые алгоритмы очень простые, то они плохо приближают антиградиент функции потерь, т.е. градиентный бустинг может свестись к случайному блужданию.
- Если базовые алгоритмы сложные, то за несколько шагов бустинг подгонится под обучающую выборку, и получим переобученный алгоритм.

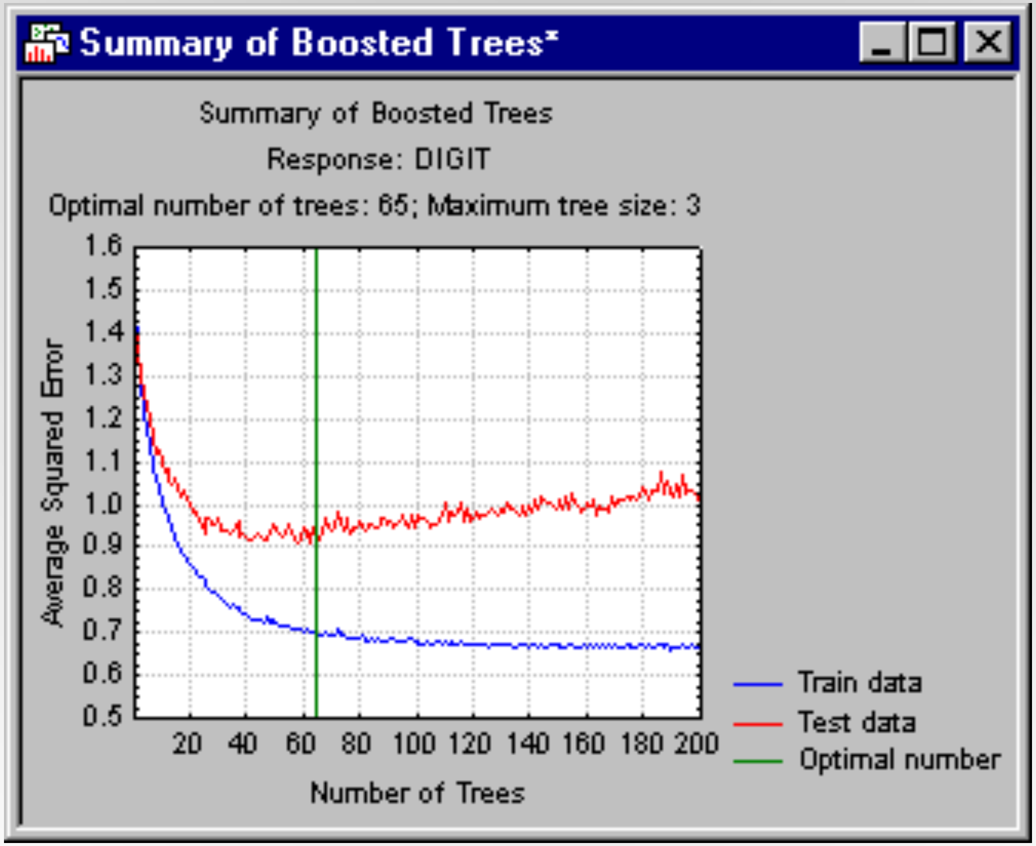
Возможное решение – сокращение шага:

$$a_N(x) = a_{N-1}(x) + \boldsymbol{\eta}\gamma_N b_N(x), \eta \in (0; 1]$$

Чем меньше темп обучения η, тем меньше степень доверия к каждому базовому алгоритму, и тем лучше качество итоговой композиции.

КОЛИЧЕСТВО ИТЕРАЦИЙ БУСТИНГА

Так как на каждом шаге бустинга целенаправленно уменьшается ошибка на тренировочной выборке, то если процесс не остановить, то мы достигнем нулевой ошибки, а значит, переобучимся!



СТОХАСТИЧЕСКИЙ ГРАДИЕНТНЫЙ БУСТИНГ

- Будем обучать базовый алгоритм b_N не по всей выборке X, а по случайной подвыборке $X^k \subset X$.

+: снижается уровень шума в данных

+: вычисления становятся быстрее

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

СМЕЩЕНИЕ И РАЗБРОС БУСТИНГА

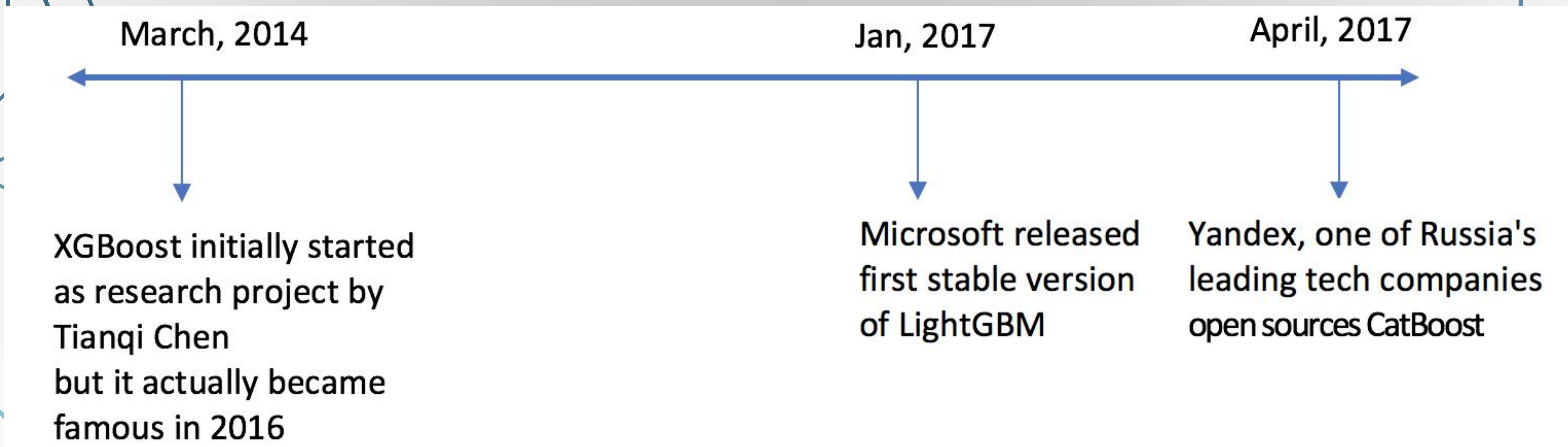
- Бустинг целенаправленно уменьшает ошибку, т.е. смещение у него маленькое.
- Алгоритм получается сложным, поэтому разброс может быть большим.

Значит, чтобы не переобучиться, в качестве базовых алгоритмов надо брать неглубокие деревья (глубины 3-6).

ИМПЛЕМЕНТАЦИИ 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

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

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

- На каждом шаге xgboost решается задача

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

$$\boldsymbol{h_i} = \frac{\partial^2 L}{\partial z^2}\bigg|_{a_{N-1}(x_i)}$$

XGBOOST

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

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

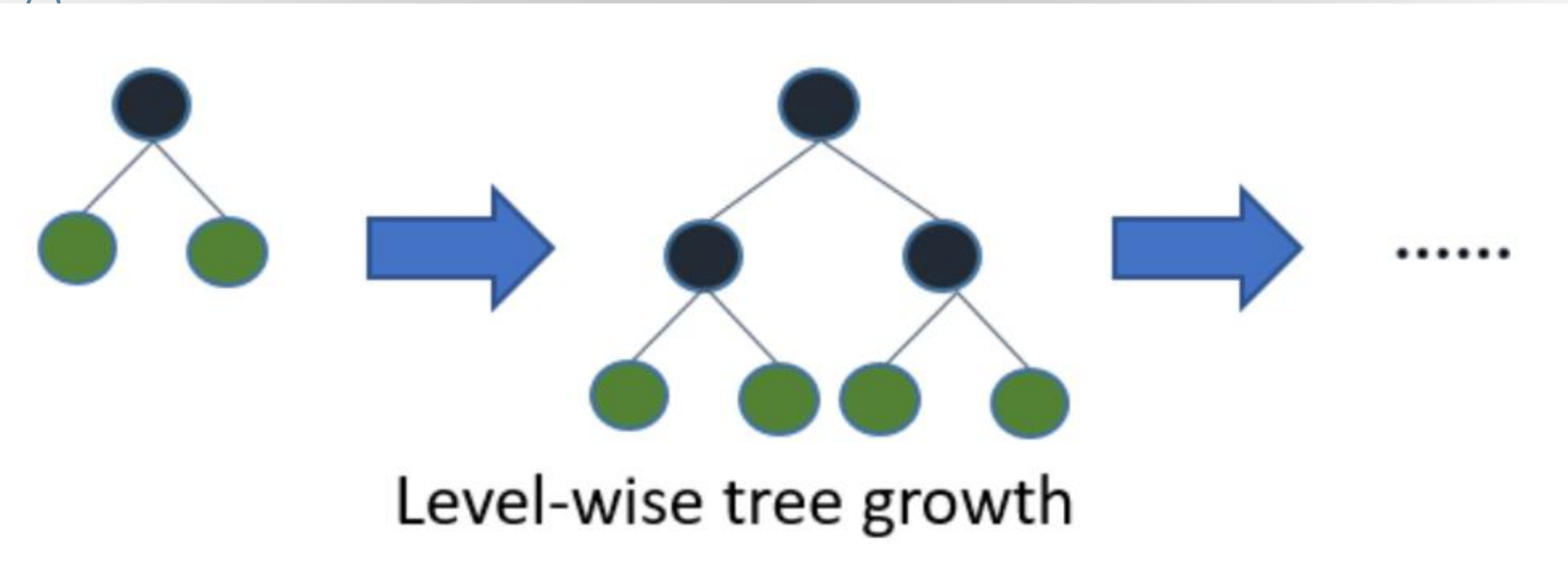
CATBOOST

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.

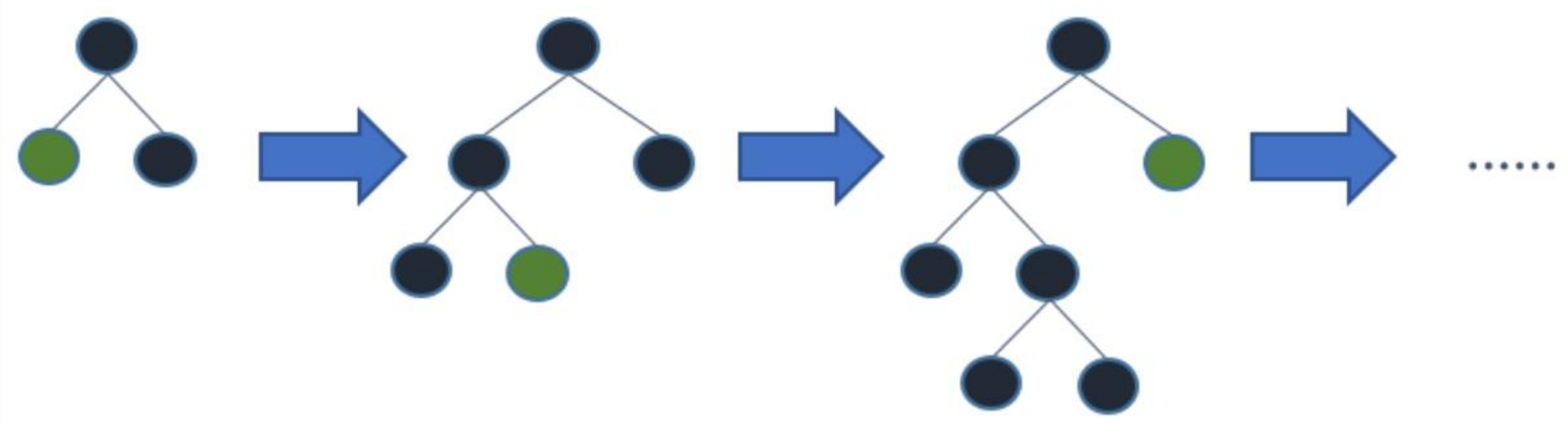
LIGHTGBM

В других реализациях градиентного бустинга деревья строятся по уровням:



LIGHTGBM

LightGBM строит деревья, добавляя на каждом шаге один лист:



Leaf-wise tree growth

Такой подход позволяет добиться более высокой точности решения задачи оптимизации.

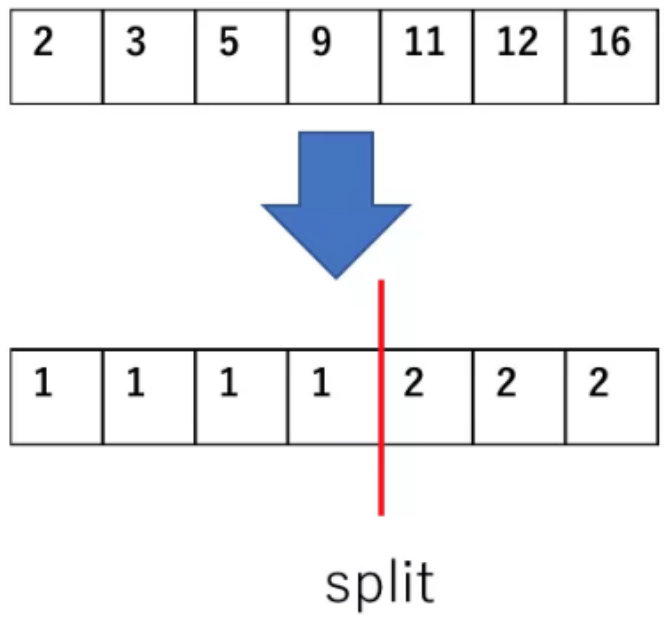
LIGHTGBM

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 $O(k \log k)$ 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.