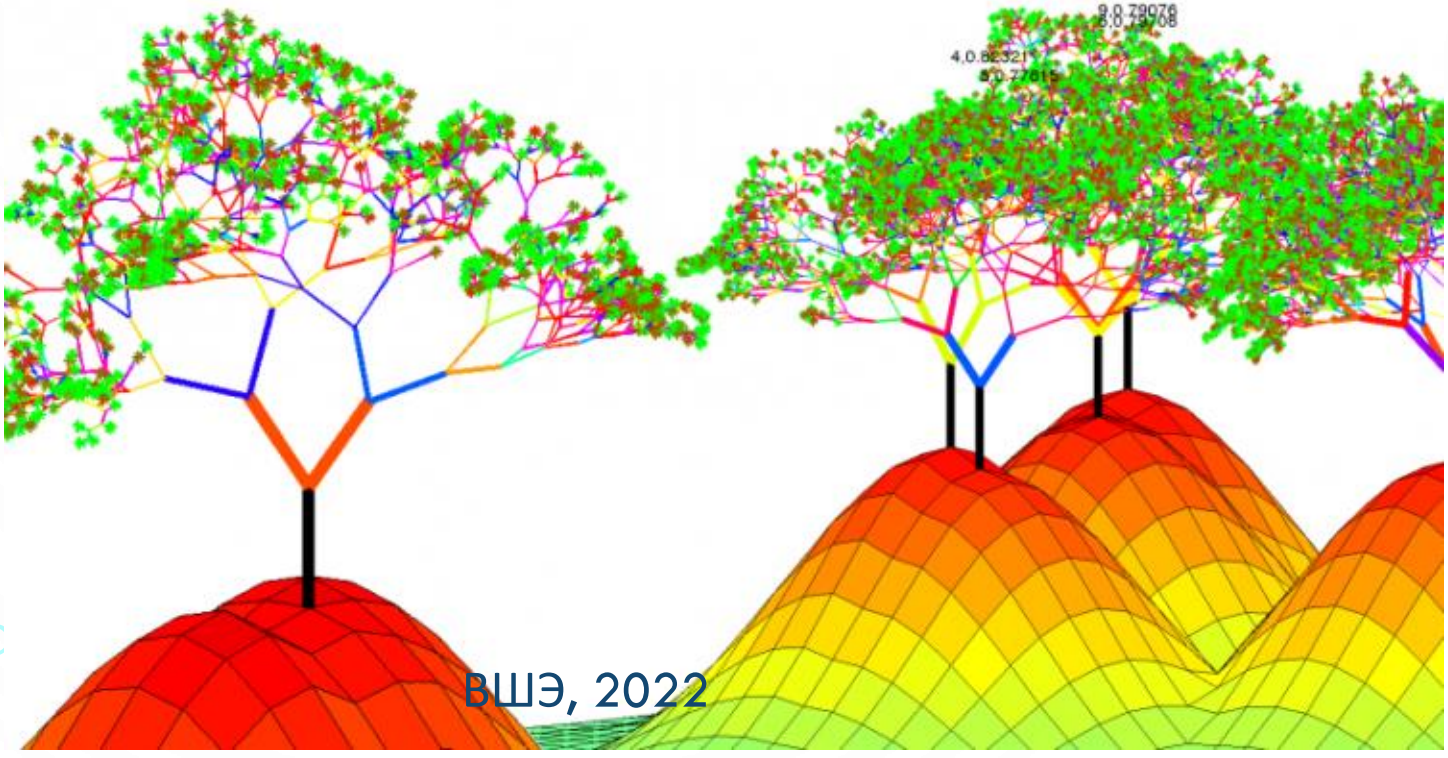
# Занятие 8
# Бустинг.

Елена Кантонистова

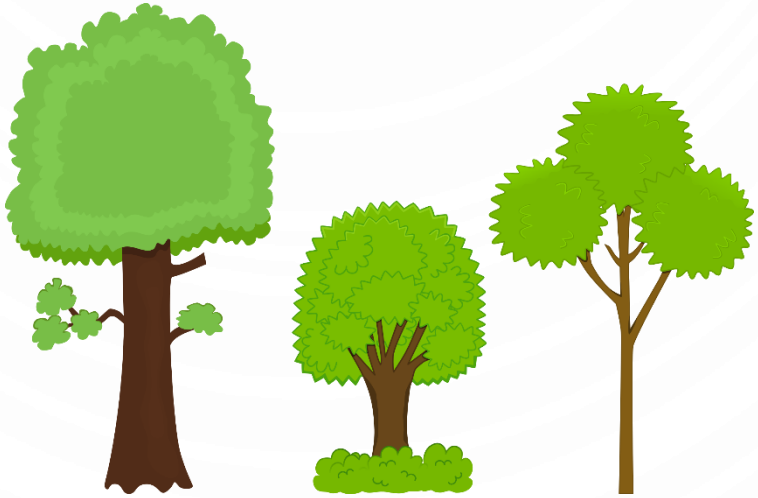
elena.kantonistova@yandex.ru



ВШЭ, 2022

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



# СМЕЩЕНИЕ И РАЗБРОС У БЭГГИНГА

**Утверждение.**

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

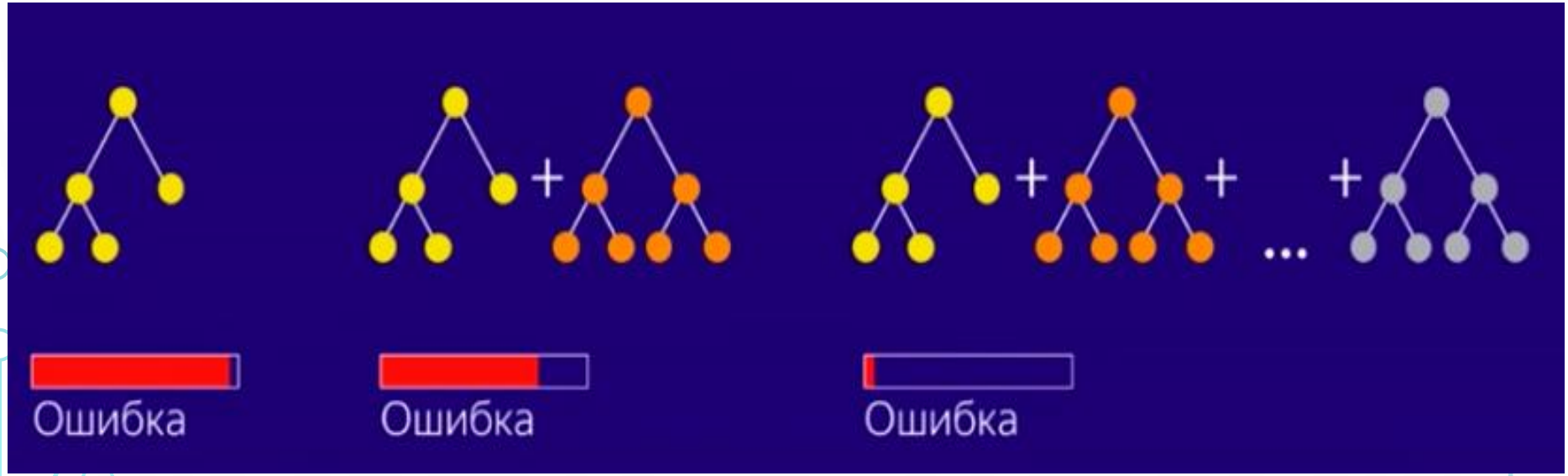
*2) Если базовые алгоритмы некоррелированы, то* ***дисперсия бэггинга*** $a_N(x)$ ***в N раз меньше дисперсии отдельных базовых алгоритмов.***

# БУСТИНГ

*Идея: 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

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:

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

Следующий алгоритм $b_3(x)$ будем выбирать так, чтобы он минимизировал ошибку предыдущей композиции (т.е. $b_1(x) + b_2(x)$) 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

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

где на $N$-м шаге

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

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

# ГРАДИЕНТНЫЙ БУСТИНГ

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

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

где на $N$-м шаге

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

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

# ГРАДИЕНТНЫЙ БУСТИНГ

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

Строим алгоритм $a_N(x)$ вида

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

где на $N$-м шаге

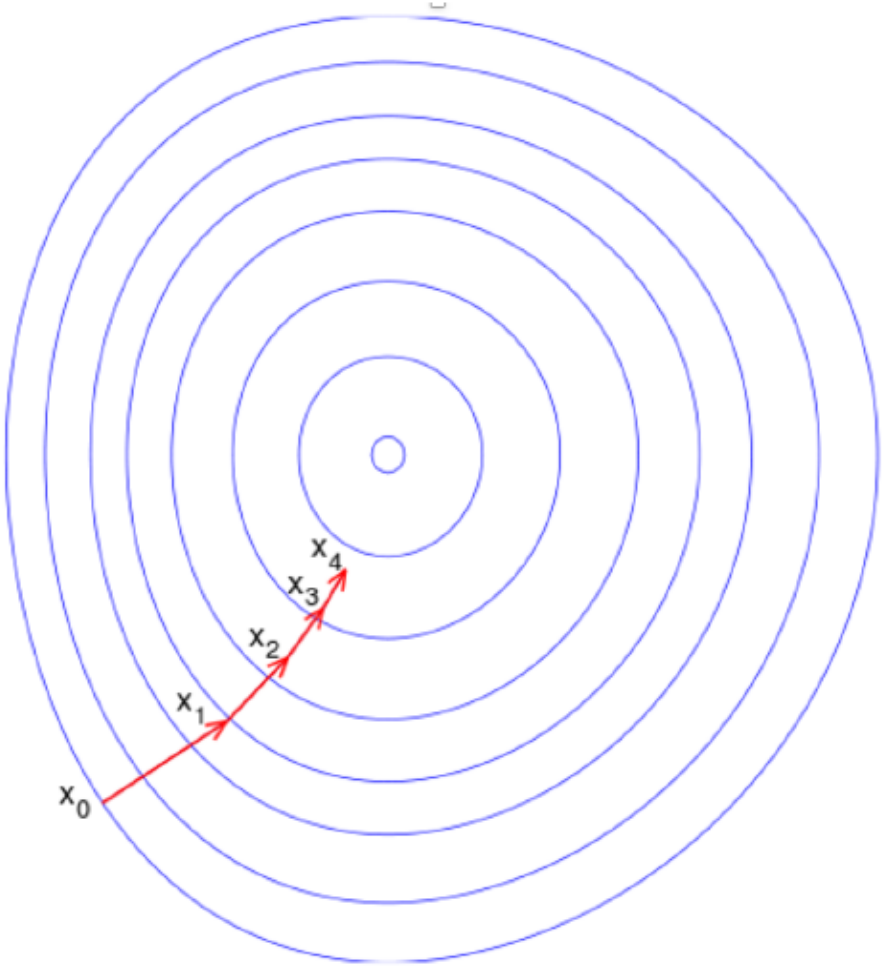
$$b_N(x) = \underset{b \in A}{\operatorname{argmin}} \sum_{i=1}^{l} \left( b(x_i) - s_i^{(N)} \right)^2,$$

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

Коэффициент $\gamma_N$ должен минимизировать ошибку:

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

# 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.

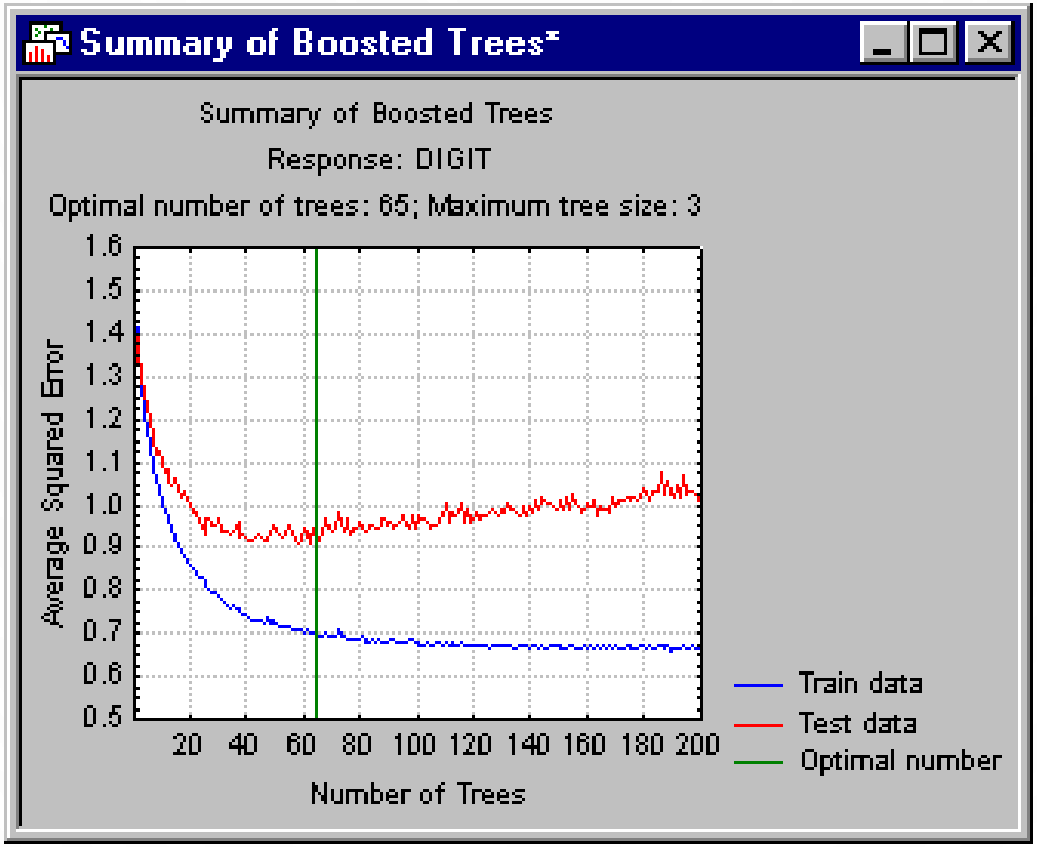
Возможное решение – сокращение шага:

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

Чем меньше темп обучения $\eta$, 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

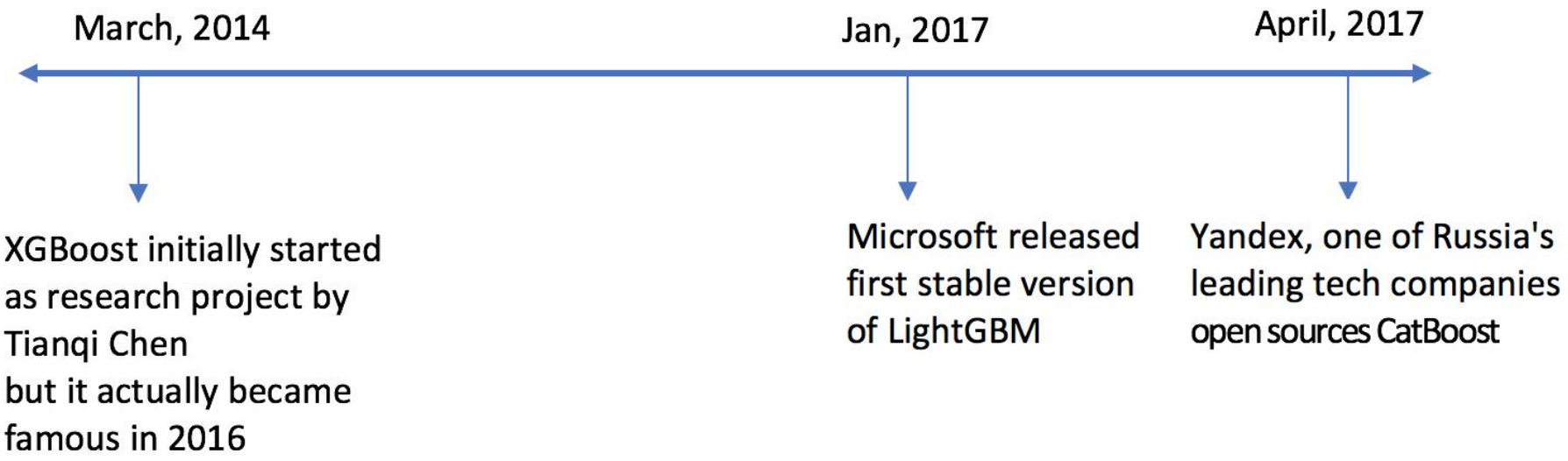
- Бустинг целенаправленно уменьшает ошибку, т.е. смещение у него маленькое.
- Алгоритм получается сложным, поэтому разброс может быть большим.

*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

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

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

# 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:

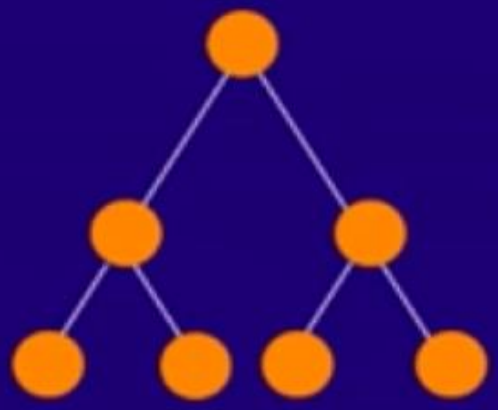
- Для кодирования категориальных признаков используется набор методов (one-hot encoding, счётчики, комбинации признаков и др.)

# CATBOOST

Особенности catboost:

- динамический бустинг

## Динамический бустинг



$$\text{leafValue(doc)} = \sum_{i=1}^{doc} \frac{g(\text{approx(i)}, \text{target(i)})}{docs\ in\ the\ past}$$

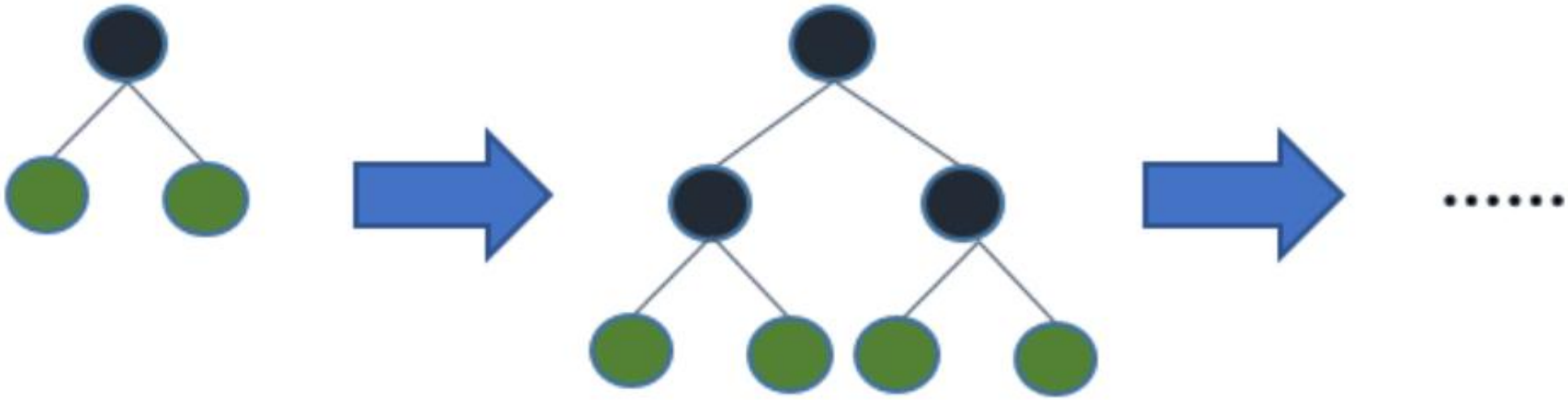
# CATBOOST

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

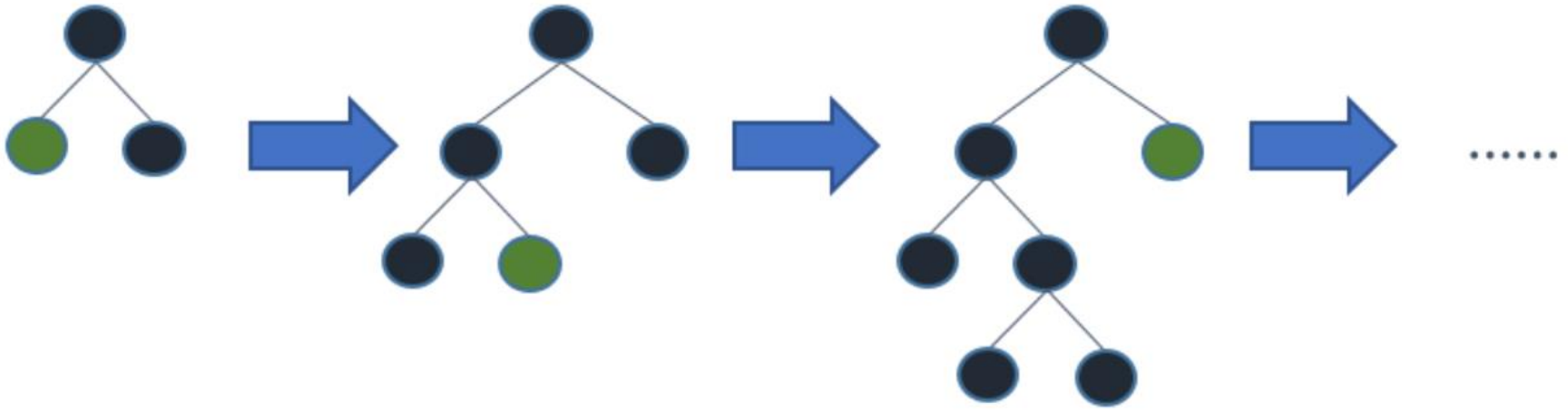
В других реализациях градиентного бустинга деревья строятся по уровням:




Level-wise tree growth

# LIGHTGBM

LightGBM строит деревья, добавляя на каждом шаге один лист:



Leaf-wise tree growth

Такой подход позволяет добиться более высокой точности решения задачи оптимизации.

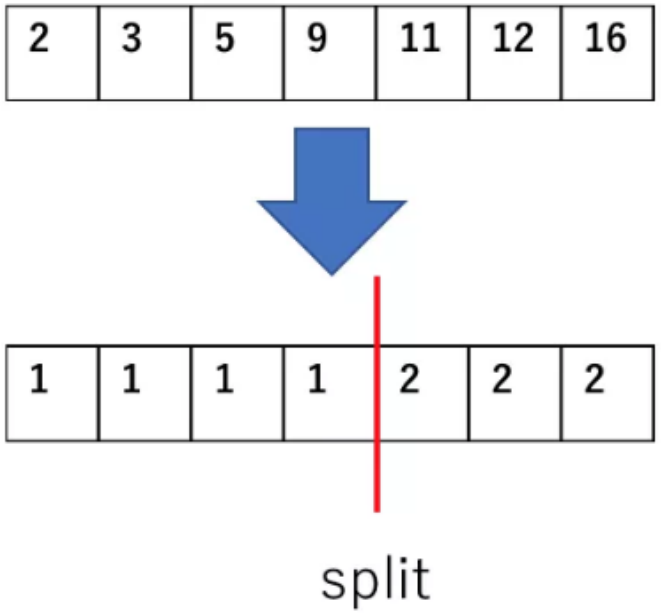
# LIGHTGBM

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 $O(klogk)$ 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.