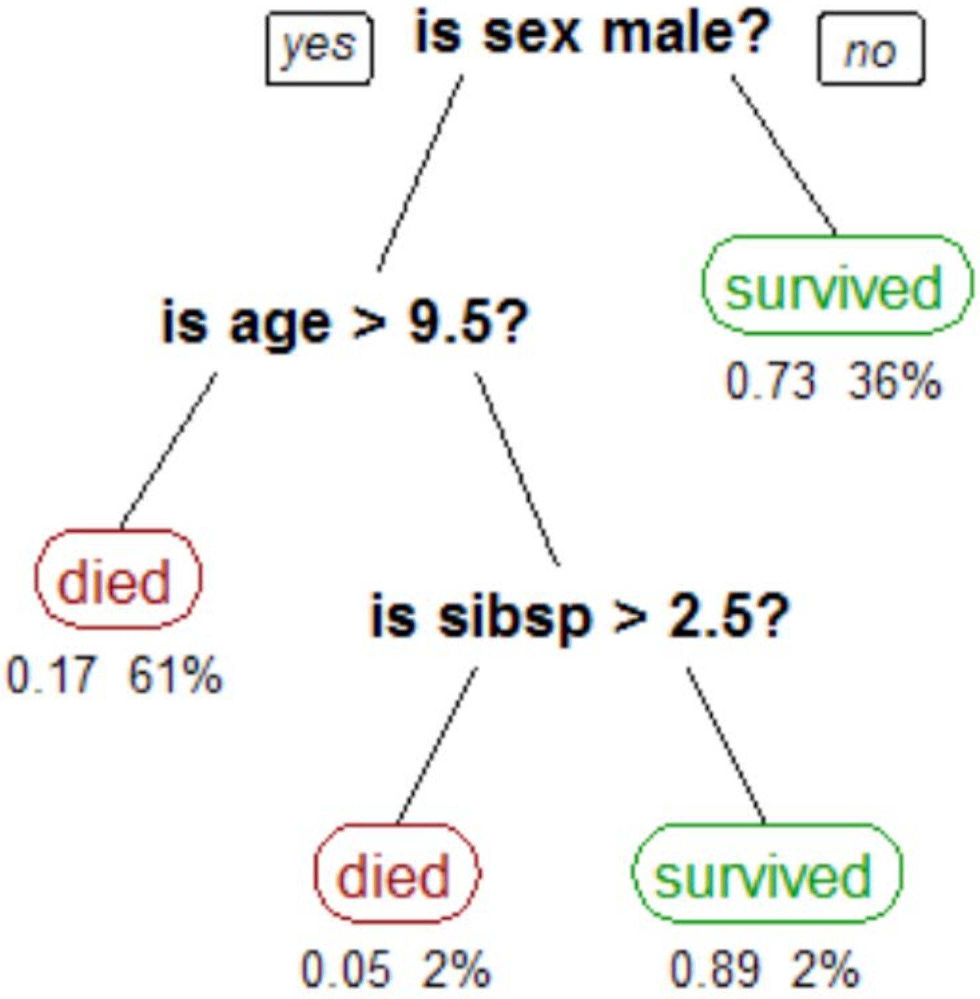
MML minor #2

Градиентный бустинг

Решающее дерево



Критерий информативности

$\{x_i, y_i\}$ — выборка
$A = \{i_1, \ldots, i_n\}$ — индексы подвыборки

Классификация (k классов)

Энтропийный критерий

$$H(A) = \sum_{i=1}^{k} p_i \log p_i, \quad p_k = \frac{1}{|A|} \sum_{i \in A} [y_i = k]$$

Регрессия

Критерий дисперсии

$$H(A) = \frac{1}{|A|} \sum_{i \in A} (y_i - \overline{y})^2, \quad \overline{y} = \frac{1}{|A|} \sum_{i \in A} y_i$$

Обучение дерева

Выбор разбиения для подвыборки A

- Рассматриваем критерии вида $[x^j < t]$ для $j = 1, \dots, d$ (признаки) и $t \in \mathbb{R}$.
- Качество разбиения A на A_l и A_r:

$$Q(A, j, t) = H(A) - \frac{|A_l|}{|A|} H(A_l) - \frac{|A_r|}{|A|} H(A_r)$$

- Наилучшее разбиение:

$$Q(A, j, t) \to \max_{j,t}$$

Критерии остановки разбиения

- Глубина
- Размер подвыборки

Решающий лес

$a_1(x), \ldots, a_n(x)$ — набор решающих деревьев

$a(x) = \text{композиция}(a_1(x), \ldots, a_n(x))$ — решающий лес

Композиция для классификации

$$a(x) = \arg\max_y \sum_{i=1}^{n} [a_i(x) = y]$$

Компопзиция для регрессии

$$a(x) = \frac{1}{n} \sum_{i=1}^{n} a_i(x)$$

Обучение леса

Как строить набор деревьев $a_1(x), \dots, a_n(x)$?

- **Бэггинг**
  Каждое дерево обучается на случайной подвыборке
- **Метод случайных подпространств**
  Каждое дерево обучается на случайном подмножестве признаков
- **Случайный лес**
  Каждое разбиение каждого дерева рассматривает случайное подмножество признаков

Бустинг для регрессии

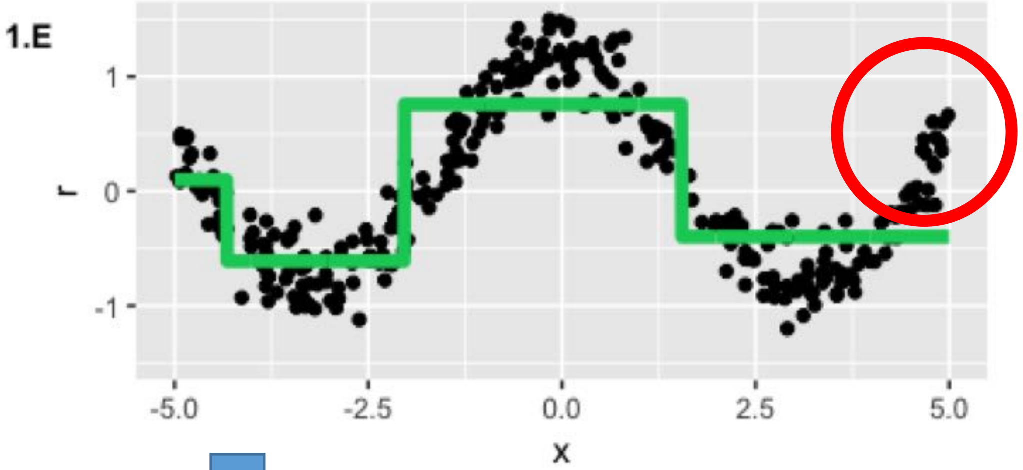
- **Композиция с помощью суммы**

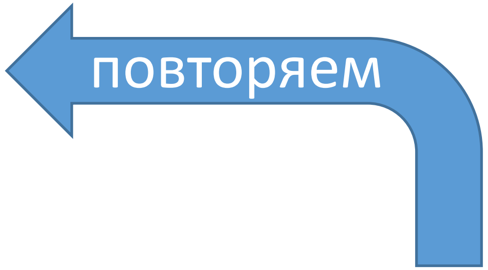
$$a(x) = a_1(x) + a_2(x) + \ldots + a_K(x)$$

- **Обучение на ошибках**
  $a_k(x)$ обучается на ошибках $a_1(x) + \ldots + a_{k-1}(x)$.

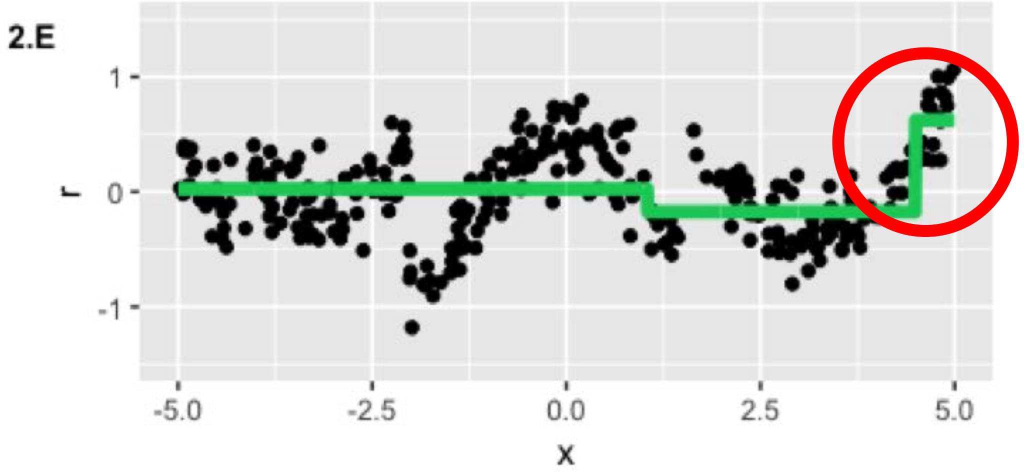
Исправляем ошибки предыдущей комбинации

Первое дерево

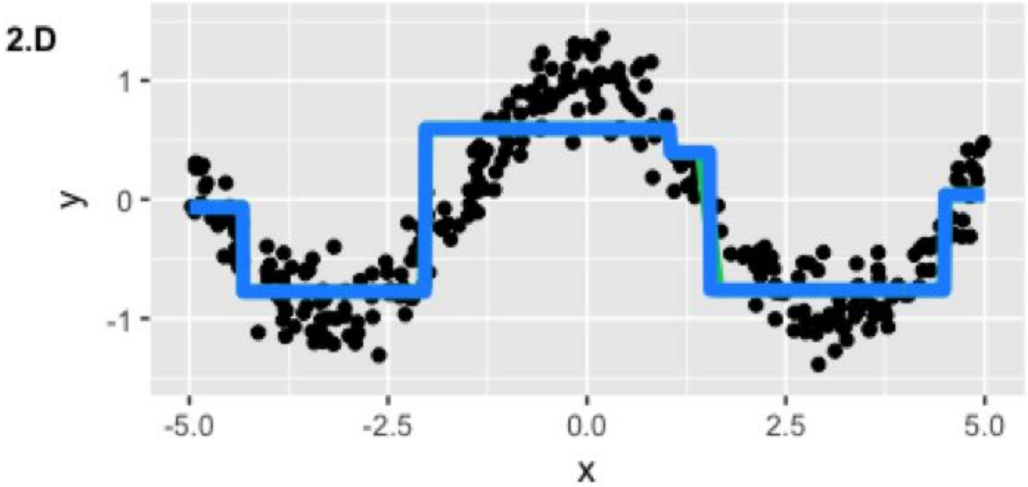




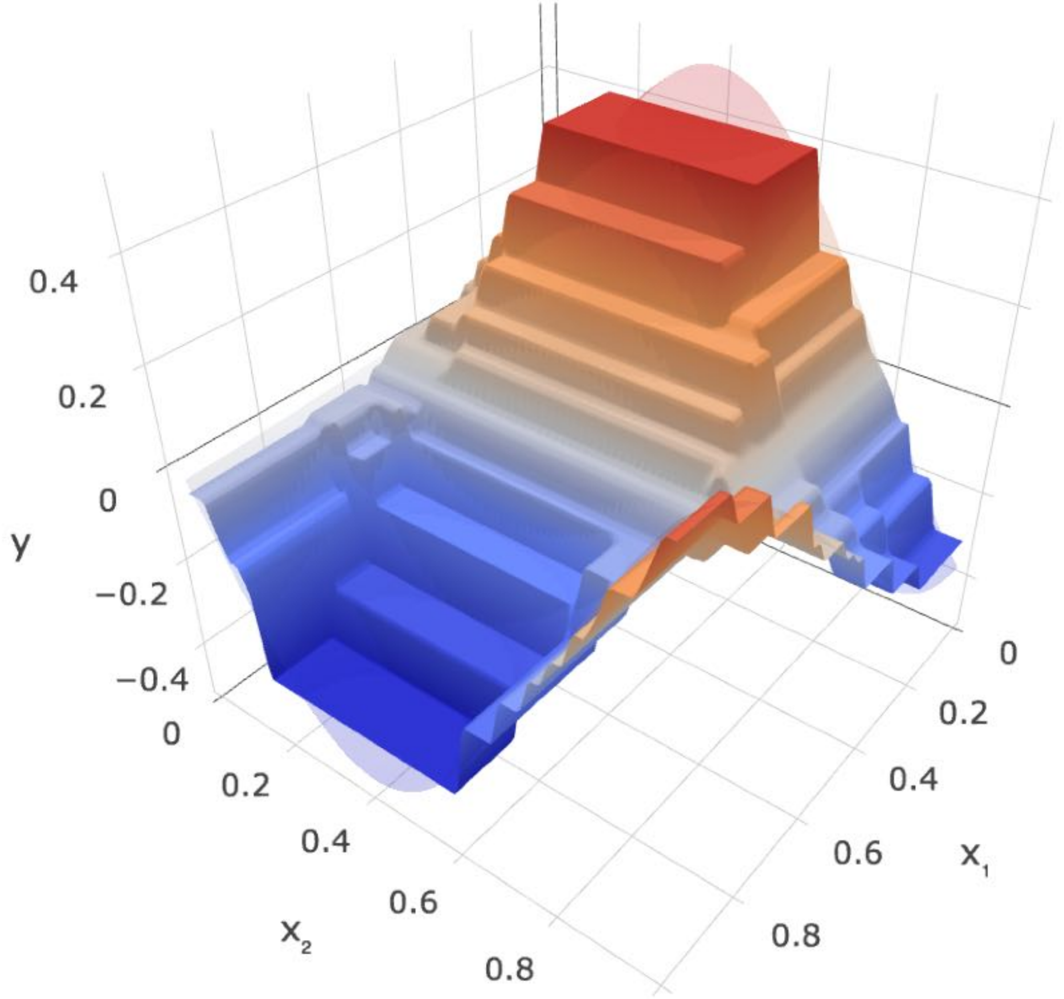
Остаток (разность)



Сумма двух деревьев

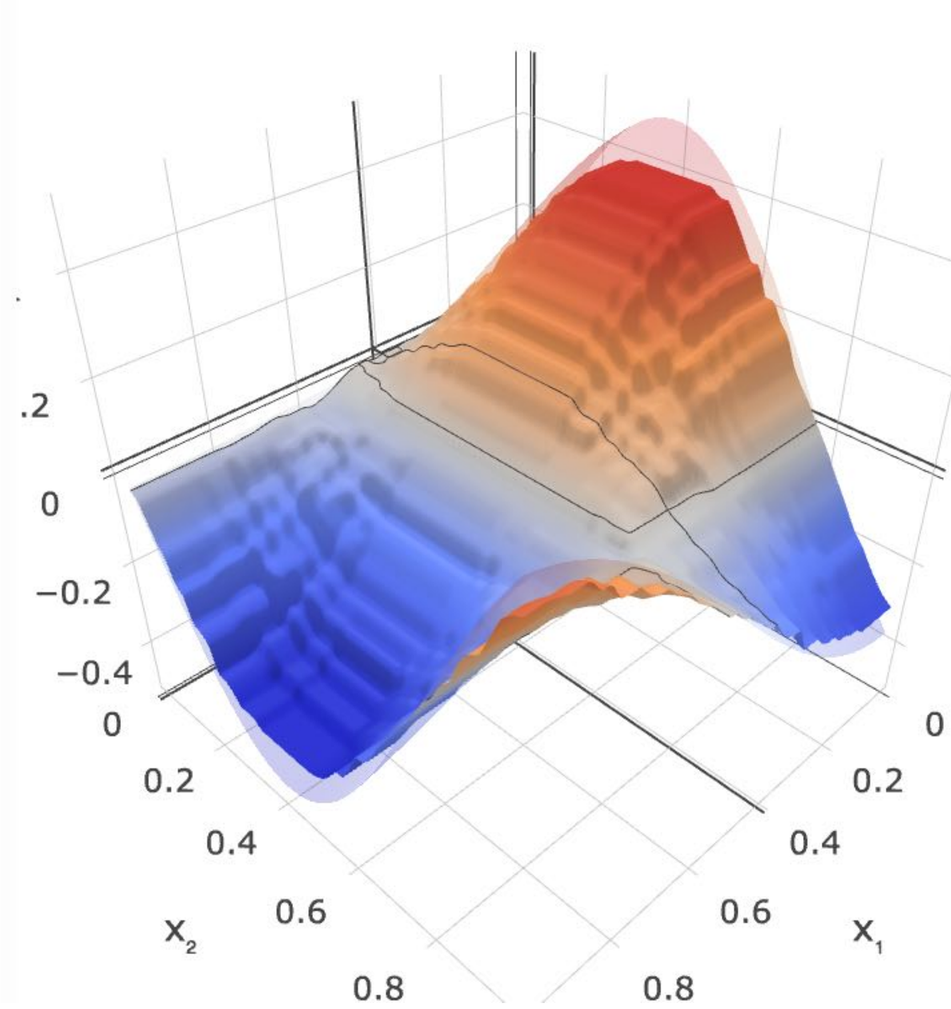


Одно дерево глубины 6

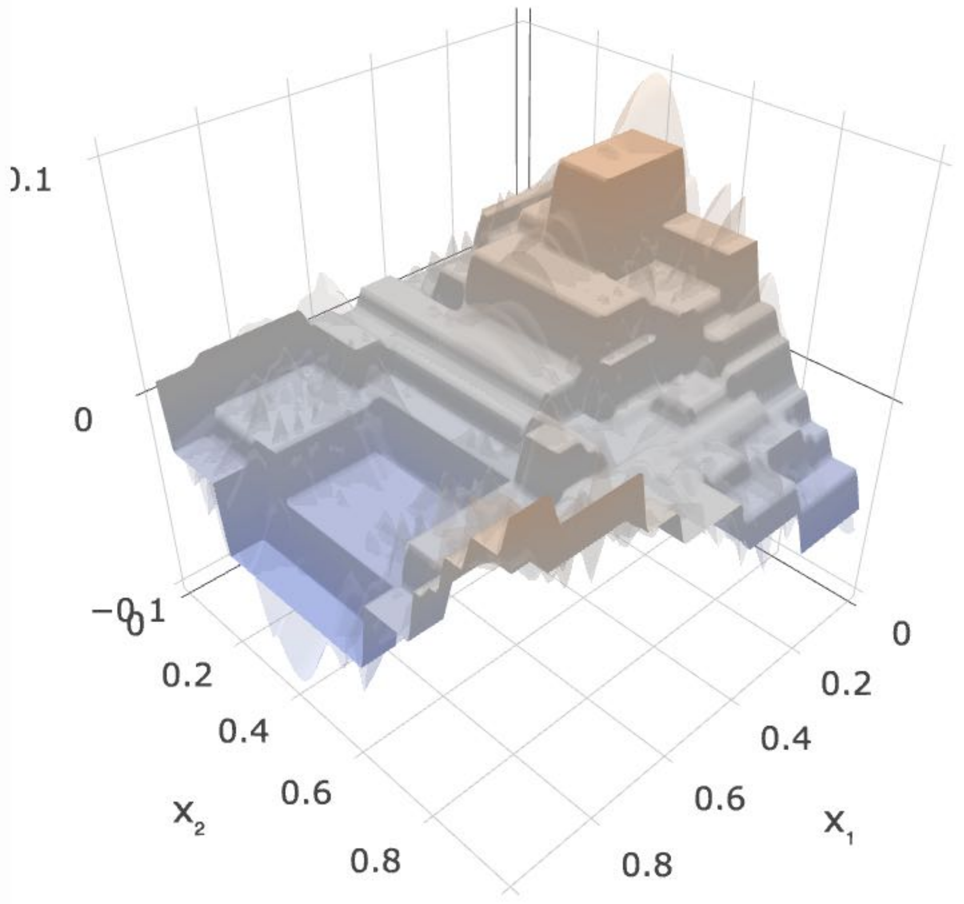


http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

Строим следующее дерево на остатки



Уже построили 6 деревьев



Строим седьмое на остатки

http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

Бустинг для регрессии

$L = \{x_i, y_i\}_{i=1}^N$ — обучающая выборка

K — количество деревьев, λ — скорость обучения

- Инициализация

$$a(x) := 0, \quad r_i = y_i \text{ для } i = 1, \dots, N$$

- Для $k = 1, \dots, K$
  - Новая обучающая выборка

$$L' = \{x_i, r_i\}_{i=1}^N$$

  - Обучение решающего дерева a_k на L'
  - Обновление алгоритма и ошибки

$$a(x) := a(x) + \lambda a_k(x)$$

$$\boxed{r_i := y_i - a(x_i)} \text{ для } i = 1, \dots N$$

Остаток и градиент потерь

- Заметим, что остатки r_i могут быть найдены как антиградиент функции потерь по ответу модели a_i, посчитанный в точке ответа уже построенной композиции $a_{N-1}(x_i)$:

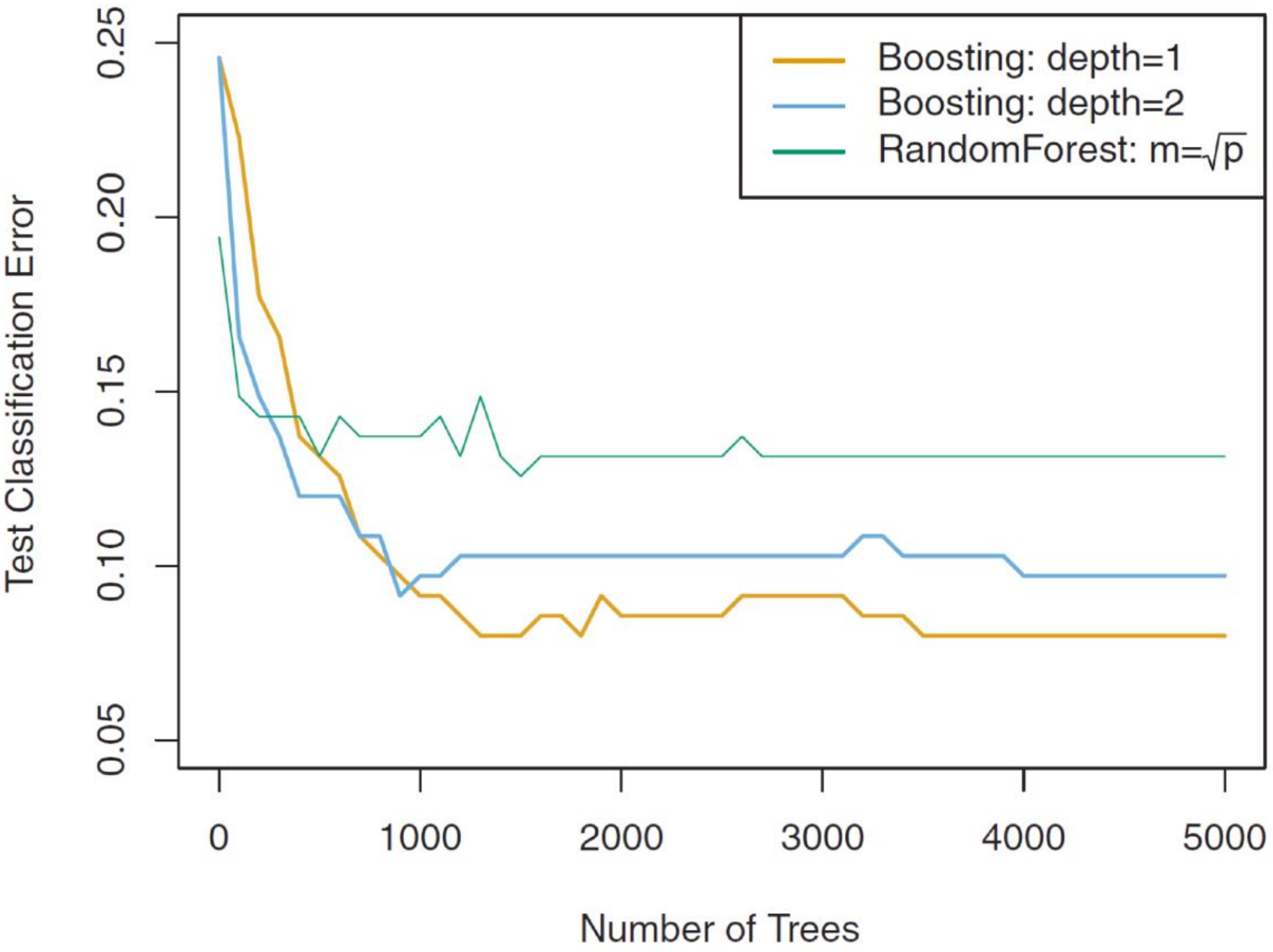
$$r_i = \boxed{y_i - a_{N-1}(x_i)} = -\frac{\partial}{\partial a_i} \frac{1}{2} \sum_{k=1}^{l} (a_k - y_k)^2 \Bigg|_{a_i = a_{N-1}(x_i)}$$

- То есть мы настраиваем следующее дерево на антиградиент потерь, тем самым мы делаем градиентный спуск в пространстве алгоритмов!

Особенности бустинга

- **Переобучение**
  Нужна сильная регуляризация деревьев.
  Например: глубина 1 или 2.
- **Медленное обучение**
  Сильная регуляризация $\to$ медленная сходимость
- **Высокая эффективность**
  (сильная регуляризация $+$ много итераций)

Пример кривой потерь



Бустинг для классификации

$\{x_i, y_i\}_{i=1}^N$ — обучающая выборка

$y_i \in \{-1, +1\}$

$$a_i(x) \in \{-1, +1\} \text{ для } i = 1, \ldots, K$$

$$a(x) = a_1(x) + \ldots + a_K(x)$$

На итерации k: $\quad a_k(x) \sim \{x_i, r_i\}$

$$r_i = y_i - a(x_i) = y_i - (a_1(x_i) + \ldots + a_{k-1}(x_i))$$

Это вообще законно?

Бустинг для классификации

Добавляем компонент:

$$Q(a, a_{k+1}) = \sum_{i=1}^{N} [y_i \neq \text{sign}(a(x_i) + a_{k+1}(x_i))]$$

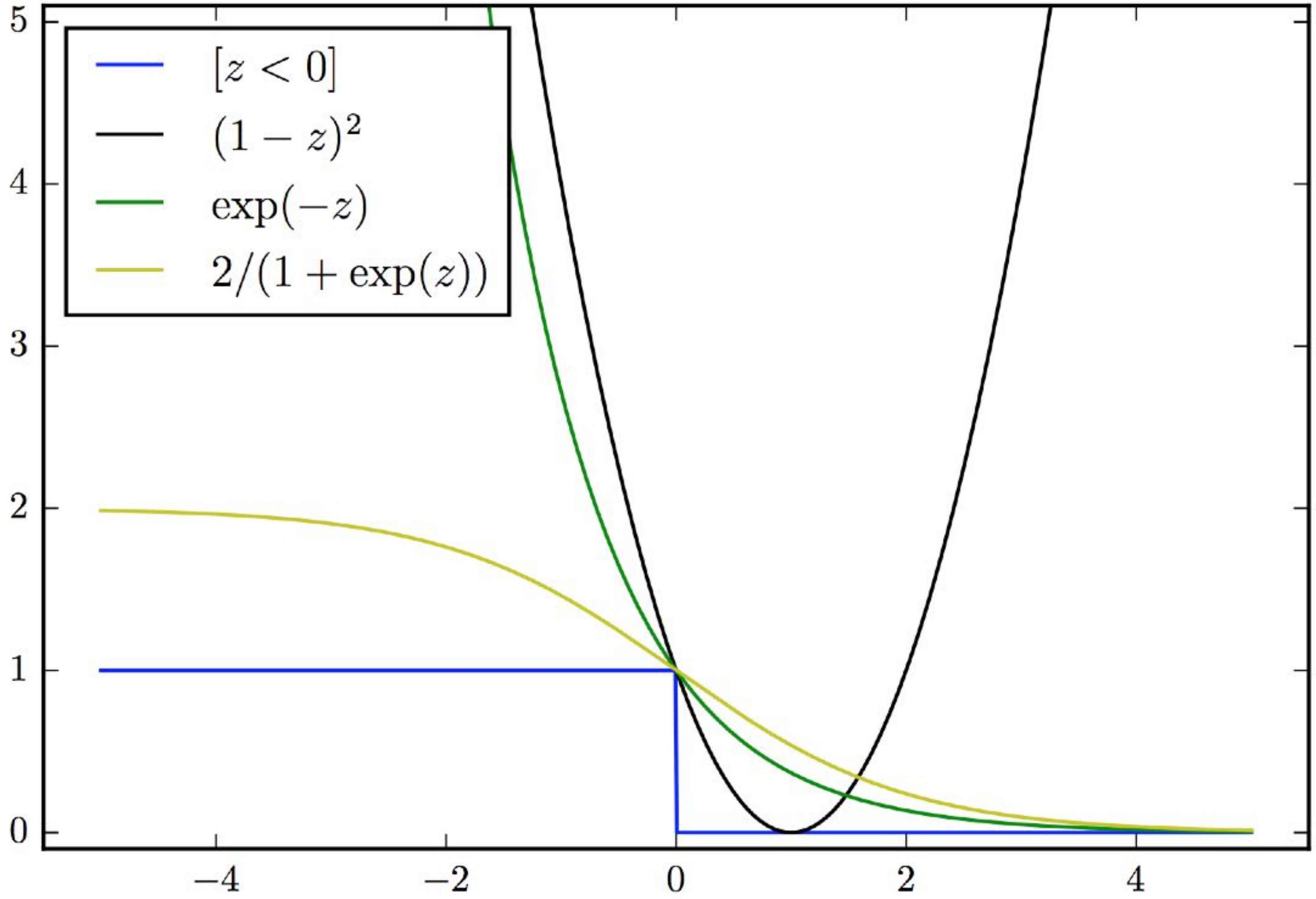
$$Q(a, a_{k+1}) \to \min_{a_{k+1}}$$

$$\sum_{i=1}^{N} [y_i \neq sign(f(x_i))] = \sum_{i=1}^{N} [\underbrace{y_i f(x_i)}_{\text{Отступ на объекте}} < 0]$$

Оптимизировать сложно!

Верхние оценки

- $T(z) = \sum_{i=1}^{N} [z = y_i f(x_i) < 0]$



AdaBoost

$$a(x) = \text{sign}(\alpha_1 t_1(x) + \ldots + \alpha_k t_k(x))$$

$$\alpha_j \in \mathbb{R}, \quad t_j(x) \in \{-1, +1\} \text{ для } j = 1, \ldots, k$$

$$Q(a) = \sum_{i=1}^{N} \exp\left(-y_i \sum_{j=1}^{k} \alpha_j t_j(x_i)\right)$$

Добавляем компонент:

$$Q(a, \alpha_{k+1}, t_{k+1}) = \sum_{i=1}^{N} \exp\left(-y_i \left(\sum_{j=1}^{k} \alpha_j t_j(x_i) + \alpha_{k+1} t_{k+1}(x_i)\right)\right)$$

$$Q(a, \alpha_{k+1}, t_{k+1}) \to \min_{\alpha_{k+1}, t_{k+1}}$$

AdaBoost

$$Q(a, \alpha_{k+1}, t_{k+1}) =$$

$$= \sum_{i=1}^{N} \exp\left(-y_i \left(\sum_{j=1}^{k} \alpha_j t_j(x_i) + \alpha_{k+1} t_{k+1}(x_i)\right)\right) =$$

$$= \sum_{i=1}^{N} \exp\left(-y_i \left(\sum_{j=1}^{k} \alpha_j t_j(x_i)\right)\right) \exp(-y_i \alpha_{k+1} t_{k+1}(x_i)) =$$

$$= \sum_{i=1}^{N} w_i \exp(-y_i \alpha_{k+1} t_{k+1}(x_i)).$$

$$w_i = \exp\left(-y_i \left(\sum_{j=1}^{k} \alpha_j t_j(x_i)\right)\right)$$

AdaBoost

$$Q(\alpha_{k+1}, t_{k+1}) = \sum_{i=1}^{N} w_i \exp\left(-y_i \alpha_{k+1} t_{k+1}(x_i)\right)$$

Обучение t_{k+1}

- Взвешенные объекты: w_i
- Исходные метки: y_i
- Ошибка классификации: $w_i[y_i \neq t_{k+1}(x_i)]$

Выбор α_{k+1} (без доказательства)

$$\varepsilon = \frac{\sum_{i=1}^{N} w_i[y_i \neq t_{k+1}(x_i)]}{\sum_{i=1}^{N} w_i}$$

$$\alpha = \ln\left((1-\varepsilon)/\varepsilon\right)$$

Взвешенные решающие деревья

Взвешенные доли классов:

$$p_k = \frac{1}{|A|} \sum_{i \in A} w_i [y_i = k]$$

Энтропия:

$$H(A) = \sum_{i=1}^{k} p_i \log p_i,$$

Качество разбиения A на A_l и A_r:

$$Q(A, j, t) = H(A) - \frac{|A_l|}{|A|} H(A_l) - \frac{|A_r|}{|A|} H(A_r)$$

Алгоритм AdaBoost

$L = \{x_i, y_i\}_{i=1}^N$ — обучающая выборка

- Инициализация решающей функции

$$f(x) := 0, \quad w_i = 1/N \text{ для } i = 1, \dots, N$$

- Для $k = 1, \dots, K$
  - Взвешенная обучающая выборка

$$L' = \{x_i, y_i, w_i\}_{i=1}^N$$

  - Обучение решающего дерева t_k на L'
  - Вычисление α_k по величине ошибки t_k на L'
  - Обновление алгоритма и весов

$$f(x) := f(x) + \alpha_k t_k(x)$$

$$w_i = \exp(-y_i f(x_i))$$

- Итоговый классификатор: $a(x) = \operatorname{sign}(f(x))$

AdaBoost

Достоинства

- Высокая обобщающая способность

Недостатки

- Неустойчивость к шуму (экспоненциальная функция потерь)
- Плохая интерпретируемость

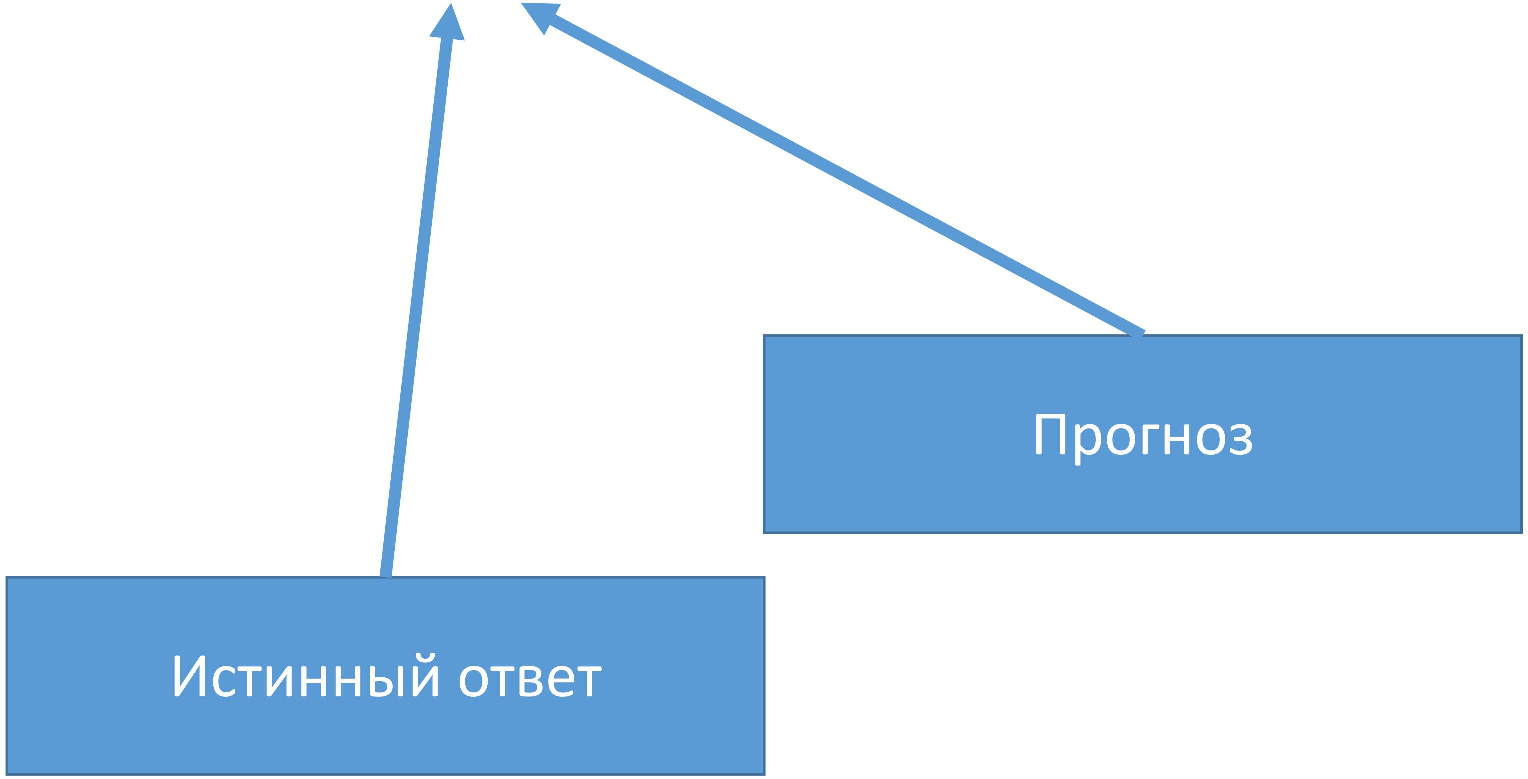
Градиентный бустинг в общем виде

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

Базовый алгоритм

Функция потерь

- Ошибка на одном объекте: $L(y, z)$



Функция потерь

- Ошибка на одном объекте: $L(y, z)$
- MSE: $L(y, z) = (y - z)^2$
- Логистическая функция потерь: $L(y, z) = \log(1 + \exp(-yz))$
- …

Инициализация

- $b_0(x)$ — первый алгоритм в композиции

- Примеры:
- $b_0(x) = 0$
- $b_0(x) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$
- $b_0(x) = \arg\max_{y \in \mathbb{Y}} \sum_{i=1}^{\ell} [y_i = y]$

Обучение базового алгоритма

- Уже построили:

$$a_{N-1}(x) = \sum_{n=0}^{N-1} b_n(x)$$

Обучение базового алгоритма

- Задача:

$$\sum_{i=1}^{\ell} L\big(y_i, a_{N-1}(x_i) + b(x_i)\big) \to \min_b$$

Оптимальный сдвиг

- Какие прогнозы оптимальны для обучающей выборки?

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i) \to \min_{s_1, \dots, s_\ell}$$

Оптимальный сдвиг

- Вектор сдвигов: $s = (s_1, \dots, s_\ell)$

$$F(s) = \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i) \to \min_s$$

Оптимальный сдвиг

- Сдвинемся в сторону наискорейшего убывания:

$$s = -\nabla F = \left(\underbrace{-L'_z(y_1, a_{N-1}(x_1)), \dots}_{\text{Сдвиг по первому объекту}}, -L'_z(y_\ell, a_{N-1}(x_\ell))\right)$$

$$F(s) = \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i) \to \min_s$$

Ошибка на одном объекте: $L(y, z)$

Оптимальный сдвиг

- Сдвинемся в сторону наискорейшего убывания:

$$s = -\nabla F = \left(-L'_z\big(y_1, a_{N-1}(x_1)\big), \dots, \underbrace{-L'_z\big(y_\ell, a_{N-1}(x_\ell)\big)}_{\text{Сдвиг по } \ell \text{ -му объекту}}\right)$$

$$F(s) = \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i) \to \min_s$$

Ошибка на одном объекте: $L(y, z)$

Обучение базового алгоритма

- Знаем $b(x)$ для обучающей выборки
- Нужно найти функцию для всего пространства объектов
- Задача машинного обучения

Обучение базового алгоритма

$$b_N(x) = \arg\min_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2$$

- Вся информация о функции потерь L содержится в сдвигах s_i
- Используем MSE независимо от исходной задачи

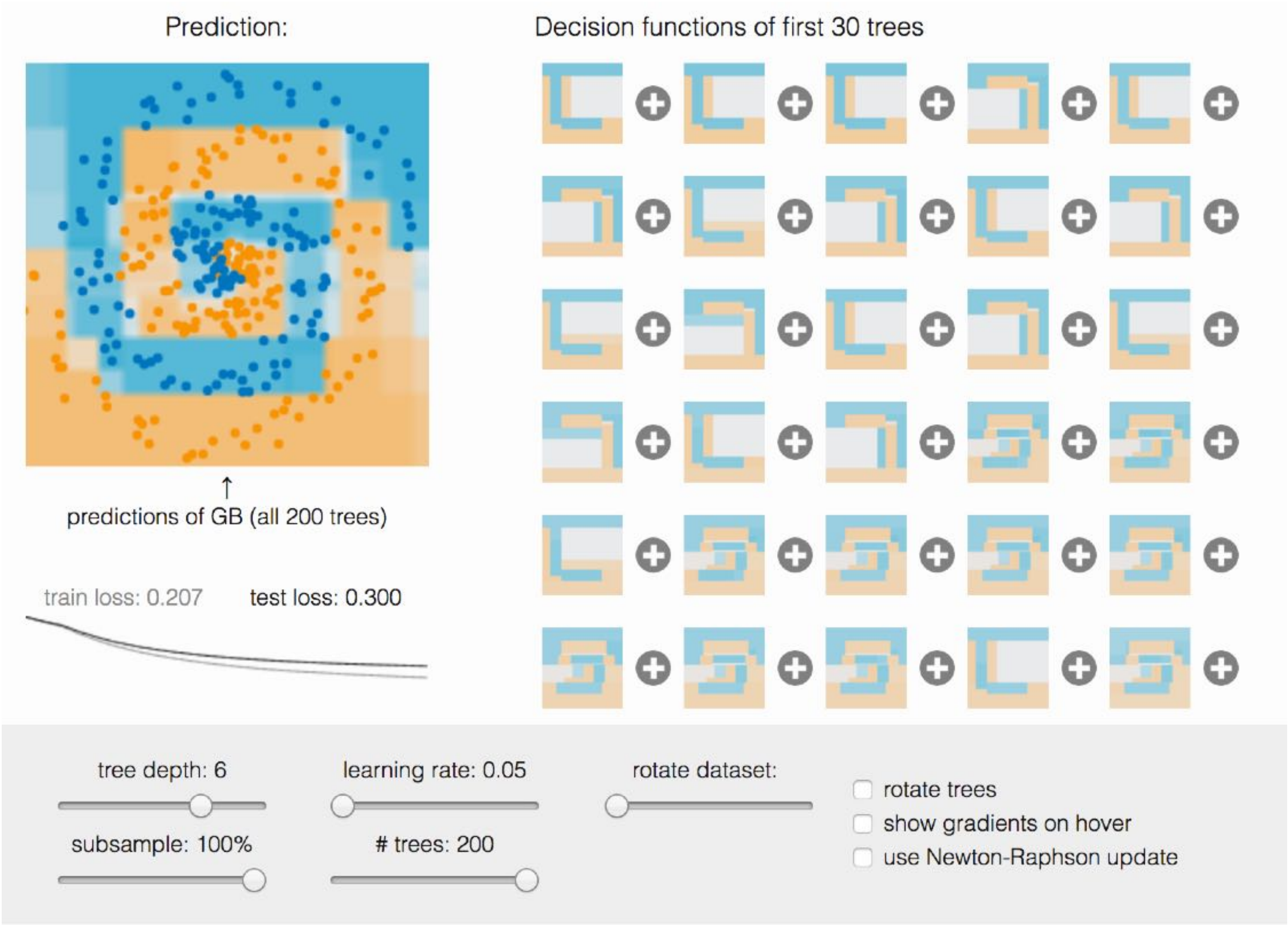
Градиентный бустинг

1. Построить начальный алгоритм $b_0(x)$
2. Для $n = 1, \dots, N$:
3. Вычислить сдвиги:
   - $s = \left(-L'_z\big(y_1, a_{n-1}(x_1)\big), \dots, -L'_z\big(y_\ell, a_{n-1}(x_\ell)\big)\right)$
4. Обучить новый базовый алгоритм:
   - $b_N(x) = \arg\min\limits_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2$
5. Добавить алгоритм в композицию: $a_n(x) = \sum_{m=1}^{n} b_m(x)$

Резюме

- Градиентный бустинг последовательно строит композицию
- Базовый алгоритм приближает антиградиент функции ошибки
- Результат — градиентный спуск в пространстве алгоритмов

Как приближает

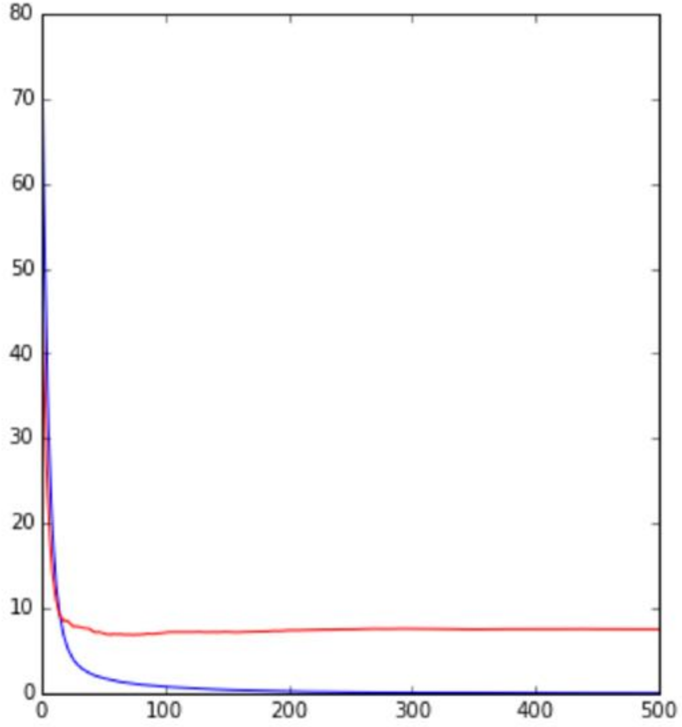


http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

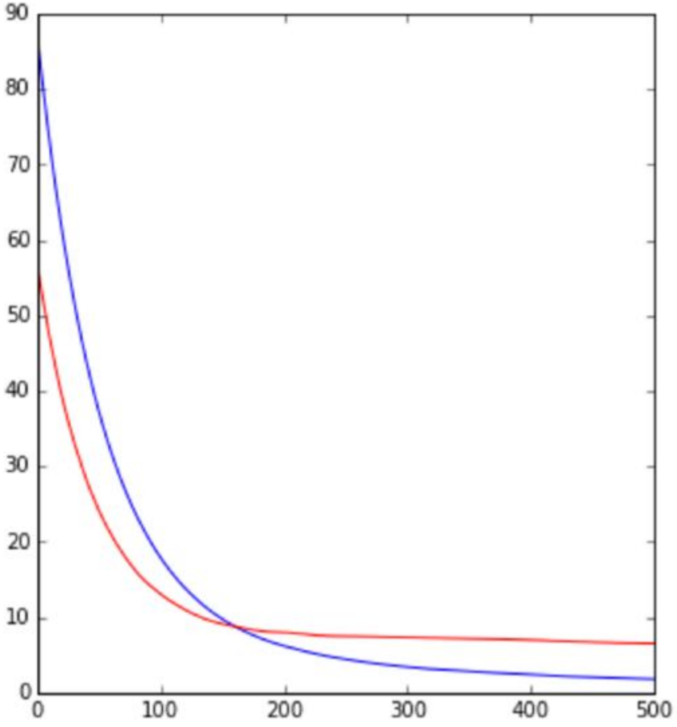
Сокращение шага

Как и в градиентном спуске шаг надо подбирать и уменьшать с итерациями

$\eta = 1$



$\eta = 0.1$



$\eta = 0.01$

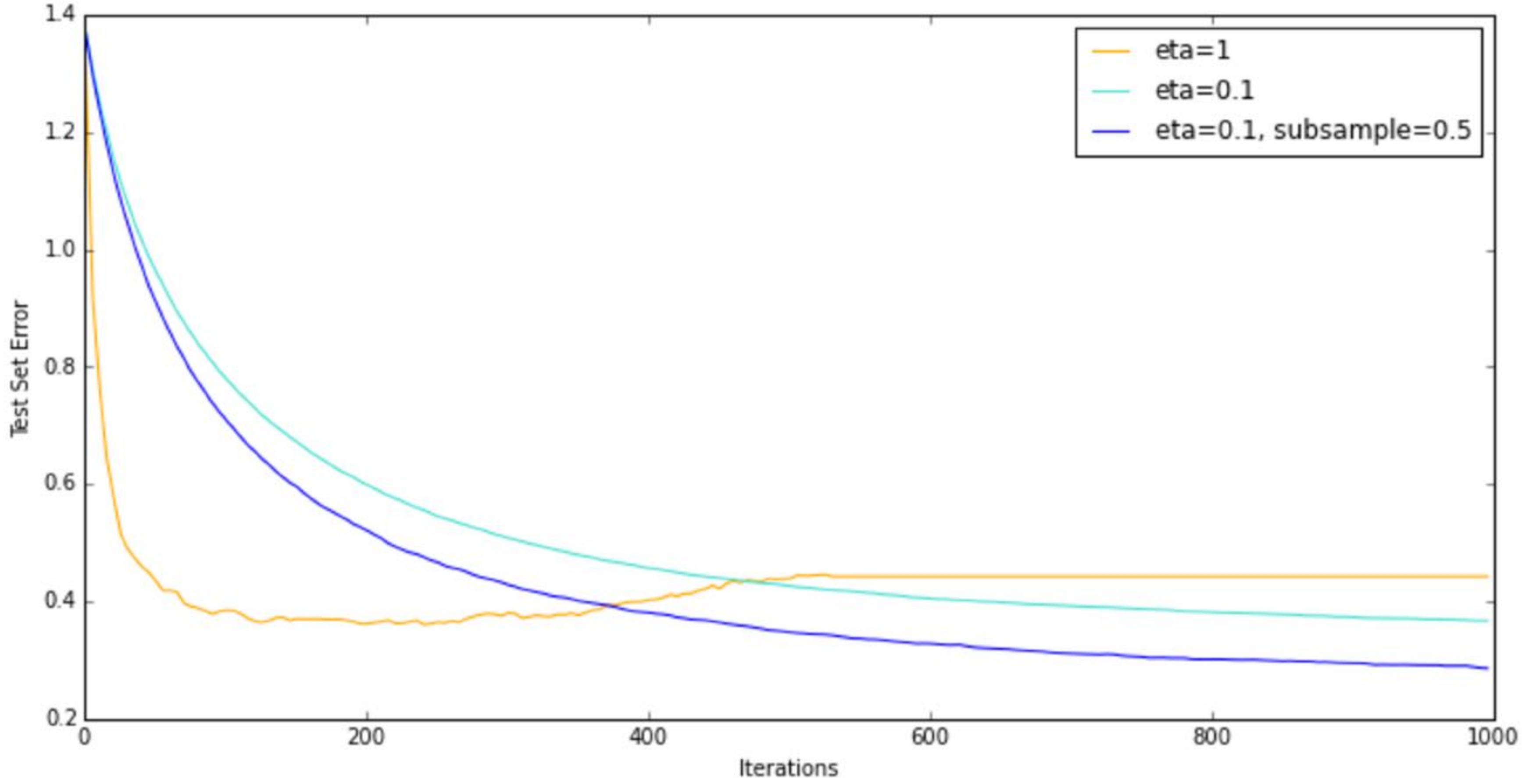
Сокращение шага

- Чем меньше шаг, тем больше нужно базовых алгоритмов
- Сокращение шага — гиперпараметр
- Две стратегии перебора:
  - Зафиксировать η, подбирать N
  - Зафиксировать N, подбирать η

Стохастический градиентный бустинг

- Обучаем каждый алгоритм по случайной подвыборке

Стохастический градиентный бустинг



Резюме

- Градиентный бустинг рано или поздно переобучается
- Решение: сокращение шага
- Решение: стохастический градиентный бустинг

Ссылки

- https://alexanderdyakonov.wordpress.com/2017/06/09/градиентный-бустинг/
- http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html
- http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html