

КОМПОЗИЦИИ ПРОСТЫХ АЛГОРИТМОВ

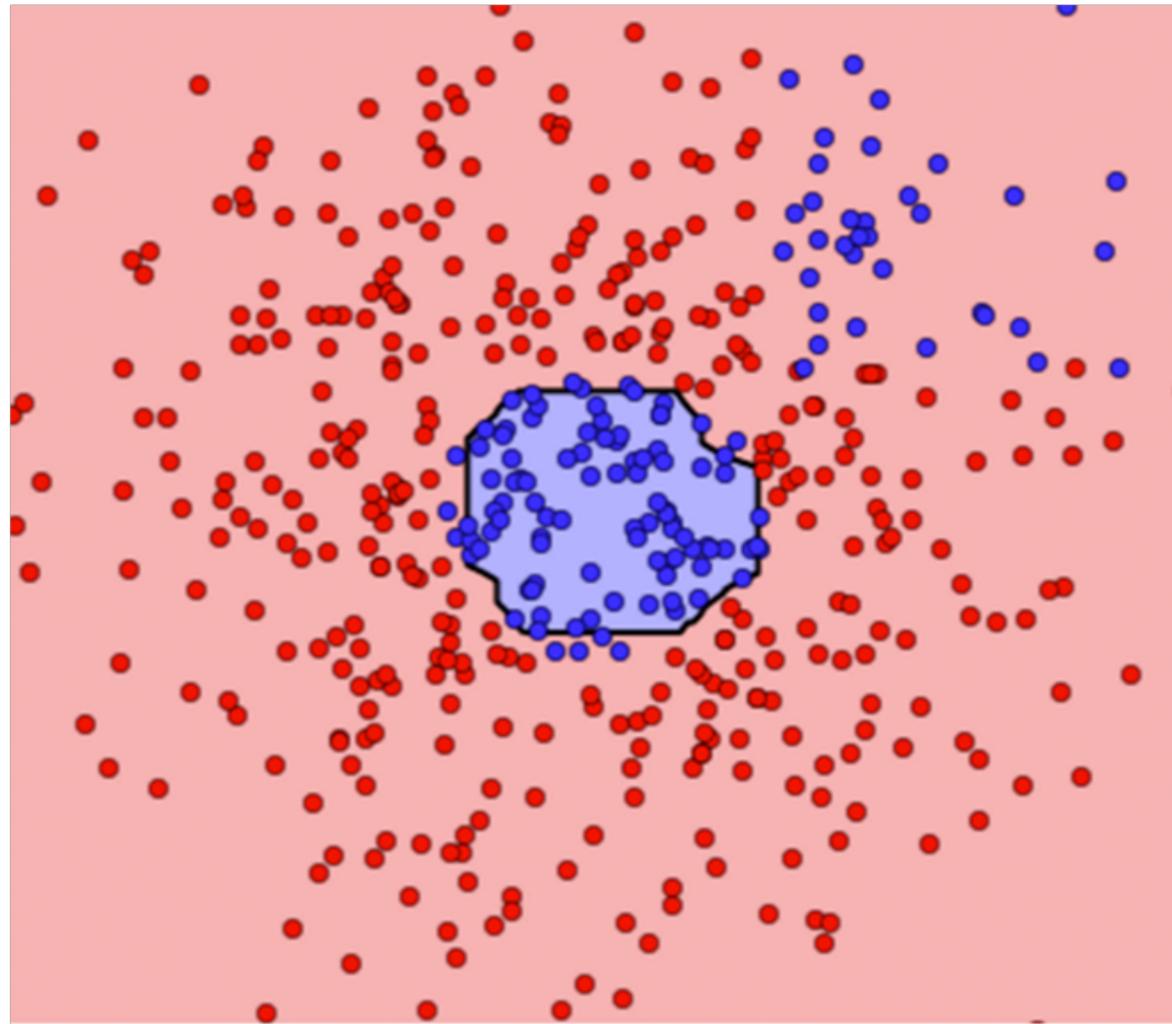
Случайный лес

- › Усреднение глубоких деревьев
- › Базовые алгоритмы независимы

Случайный лес

- › Усреднение глубоких деревьев
- › Базовые алгоритмы независимы
- › Обучение глубоких деревьев — трудоемкая процедура
- › Что, если ограничить глубину?

СЛУЧАЙНЫЙ ЛЕС



Случайный лес

- › Построение деревьев ненаправленное
- › Нужно много деревьев для сложных задач

БУСТИНГ

- › Последовательное обучение базовых алгоритмов
- › Каждый следующий исправляет ошибки предыдущих
- › За счёт этого достаточно простых базовых алгоритмов

ПРОСТОЙ ПРИМЕР

- » Регрессия
- » Среднеквадратичная ошибка:

$$\text{MSE}(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2$$

ПРОСТОЙ ПРИМЕР

- » Обучим простой алгоритм (неглубокое дерево):

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

ПРОСТОЙ ПРИМЕР

- » Обучим простой алгоритм (неглубокое дерево):

$$b_1(x) = \operatorname{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - y_i)^2$$

Ответы	y_1	y_2	...	y_ℓ
Прогнозы	$b_1(x_1)$	$b_1(x_2)$...	$b_1(x_\ell)$

ПРОСТОЙ ПРИМЕР

» Как исправить ошибки $b_1(x)$?

$$b_1(x_i) + b_2(x_i) = y_i$$

ПРОСТОЙ ПРИМЕР

» Как исправить ошибки $b_1(x)$?

$$b_1(x_i) + b_2(x_i) = y_i$$

Ответы	y_1	y_2	...	y_ℓ
Прогнозы	$b_1(x_1)$	$b_1(x_2)$...	$b_1(x_\ell)$
Поправка	$y_1 - b_1(x_1)$	$y_2 - b_1(x_2)$...	$y_\ell - b_1(x_\ell)$

ПРОСТОЙ ПРИМЕР

» Как исправить ошибки $b_1(x)$?

$$b_2(x) = \operatorname{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} \left(b(x_i) - (y_i - b_1(x_i)) \right)^2$$

ПРОСТОЙ ПРИМЕР

» Как исправить ошибки $b_1(x)$?

$$b_N(x) = \underset{b}{\operatorname{argmin}} \frac{1}{\ell} \sum_{i=1}^{\ell} \left(b(x_i) - \left(y_i - \sum_{n=1}^{N-1} b_n(x_i) \right) \right)^2$$

РЕЗЮМЕ

- › Случайный лес может оказаться слишком медленным
- › Бустинг — направленное построение композиции
- › Простой пример: бустинг для MSE

ГРАДИЕНТНЫЙ БУСТИНГ

КОМПОЗИЦИЯ

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



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:
 - ▶ $b_0(x) = 0$
 - ▶ $b_0(x) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$
 - ▶ $b_0(x) = \operatorname{argmax}_{y \in \mathbb{Y}} \sum_{i=1}^{\ell} [y_i = y]$

ОБУЧЕНИЕ БАЗОВОГО АЛГОРИТМА

› Уже построили:

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

ОБУЧЕНИЕ БАЗОВОГО АЛГОРИТМА

› Задача:

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

ОПТИМАЛЬНЫЙ СДВИГ

- › Какие прогнозы оптимальны для обучающей выборки?

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

ОПТИМАЛЬНЫЙ СДВИГ

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

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

ОПТИМАЛЬНЫЙ СДВИГ

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

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

ОПТИМАЛЬНЫЙ СДВИГ

- › Сдвинемся в сторону наискорейшего убывания:

$$s = -\nabla F = \left(\boxed{-L'_z(y_1, a_{N-1}(x_1)), \dots}, -L'_z(y_\ell, a_{N-1}(x_\ell)) \right)$$

Сдвиг по первому объекту

ОПТИМАЛЬНЫЙ СДВИГ

- › Сдвинемся в сторону наискорейшего убывания:

$$s = -\nabla F = \left(-L'_z(y_1, a_{N-1}(x_1)), \dots, \boxed{-L'_z(y_\ell, a_{N-1}(x_\ell))} \right)$$

Сдвиг по ℓ -му объекту

ОБУЧЕНИЕ БАЗОВОГО АЛГОРИТМА

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

ОБУЧЕНИЕ БАЗОВОГО АЛГОРИТМА

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

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

ГРАДИЕНТНЫЙ БУСТИНГ

› Построить начальный алгоритм $b_0(x)$

› Для $n = 1, \dots, N$:

► Вычислить сдвиги:

$$s = \left(-L'_z(y_1, a_{n-1}(x_1)), \dots, -L'_z(y_\ell, a_{n-1}(x_\ell)) \right)$$

► Обучить новый базовый алгоритм:

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

► Добавить алгоритм в композицию:

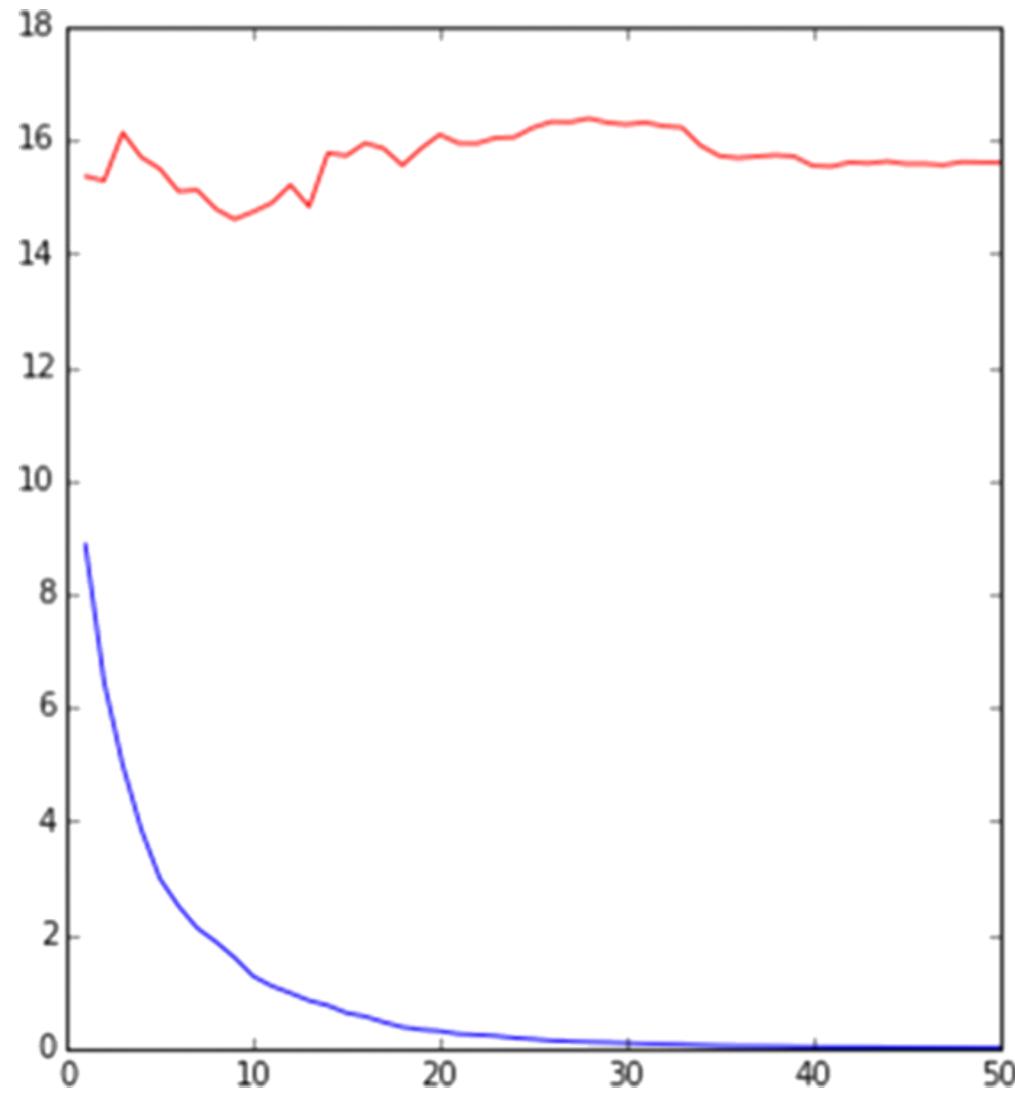
$$a_n(x) = \sum_{m=1}^n b_m(x)$$

РЕЗЮМЕ

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

БОРЬБА С ПЕРЕОБУЧЕНИЕМ В ГРАДИЕНТНОМ БУСТИНГЕ

ГРАДИЕНТНЫЙ БУСТИНГ



ПРИЧИНЫ ПЕРЕОБУЧЕНИЯ

- › Базовый алгоритм должен приближать вектор антиградиента
- › Базовые алгоритмы очень слабые
- › Приближение плохое
- › Вместо градиентного спуска получаем случайное блуждание

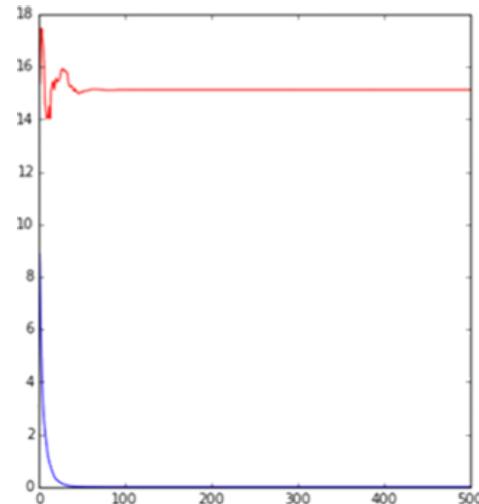
СОКРАЩЕНИЕ ШАГА

› Не будем доверять базовому алгоритму:

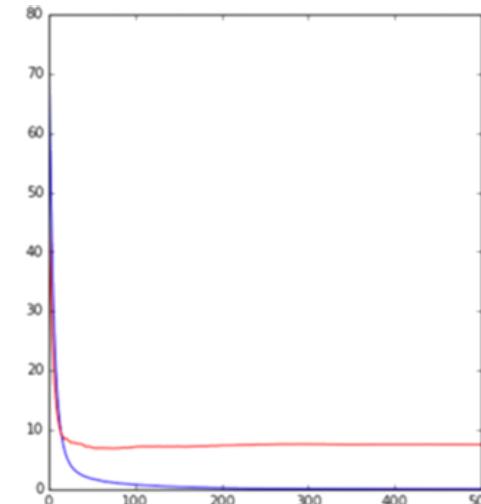
$$a_N(x) = a_{N-1}(x) + \eta b_N(x)$$

› $\eta \in (0, 1]$ — длина шага

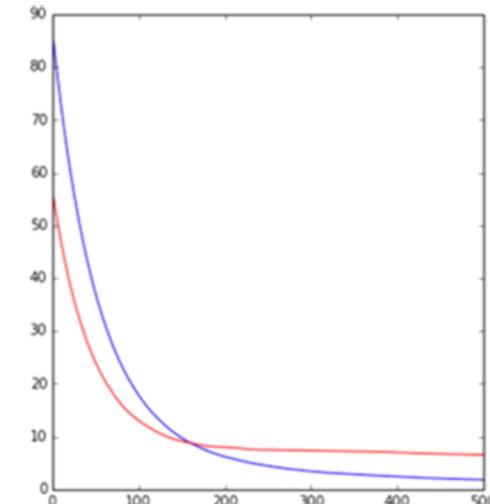
СОКРАЩЕНИЕ ШАГА



$$\eta = 1$$



$$\eta = 0.1$$



$$\eta = 0.01$$

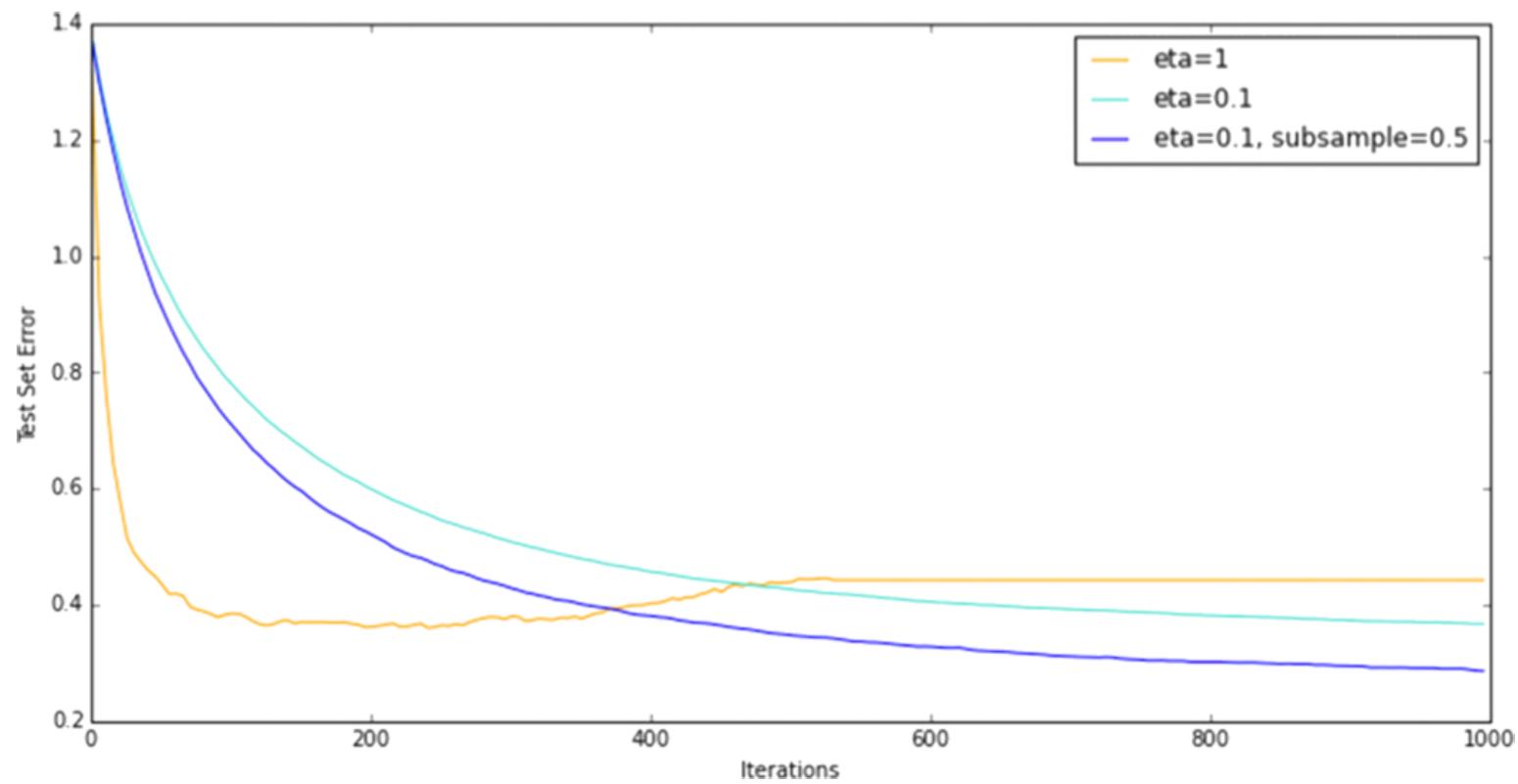
ПРИЧИНЫ ПЕРЕОБУЧЕНИЯ

- › 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 переобучается из-за слабых базовых алгоритмов
- › Решение: сокращение шага
- › Стохастический градиентный бустинг

ГРАДИЕНТНЫЙ БУСТИНГ ДЛЯ РЕГРЕССИИ И КЛАССИФИКАЦИИ

ГРАДИЕНТНЫЙ БУСТИНГ

› Построить начальный алгоритм $b_0(x)$

› Для $n = 1, \dots, N$:

► Вычислить сдвиги:

$$s = \left(-L'_z(y_1, a_{n-1}(x_1)), \dots, -L'_z(y_\ell, a_{n-1}(x_\ell)) \right)$$

► Обучить новый базовый алгоритм:

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

► Добавить алгоритм в композицию:

$$a_n(x) = \sum_{m=1}^n b_m(x)$$

ГРАДИЕНТНЫЙ БУСТИНГ

- › Базовые алгоритмы — решающие деревья
- › Ограничение на глубину (2-8)
- › Сокращение шага
- › Обучение на подвыборках

РЕГРЕССИЯ

- » Среднеквадратичная ошибка:

$$\text{MSE}(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2$$

- » $L(y, z) = (z - y)^2$

- » $L'_z(y, z) = 2(z - y)$

РЕГРЕССИЯ

› Вектор сдвигов:

$$s = (-2(a_{N-1}(x_1) - y_1), \dots, -2(a_{N-1}(x_\ell) - y_\ell))$$

КЛАССИФИКАЦИЯ

- » $Y = \{-1, +1\}$
- » Логистическая ошибка:
$$\sum_{i=1}^n \log (1 + \exp (-y_i a(x_i)))$$
- » $a(x)$ — оценка принадлежности
положительному классу

КЛАССИФИКАЦИЯ

› Логистическая ошибка:

$$\sum_{i=1}^n \log (1 + \exp (-y_i a(x_i)))$$

› $a(x)$ — оценка принадлежности
положительному классу

› $L(y, z) = \log (1 + \exp (-yz))$

› $L'_z(y, z) = -\frac{y}{1+\exp(-yz)}$

КЛАССИФИКАЦИЯ

› Вектор сдвигов:

$$s = \left(\frac{y_1}{1 + \exp(y_1 a_{N-1}(x_1))}, \dots, \dots, \frac{y_\ell}{1 + \exp(y_\ell a_{N-1}(x_\ell))} \right)$$

КЛАССИФИКАЦИЯ

› Вероятности классов:

$$P(y = 1|x) = \frac{1}{1 + \exp(-a_N(x))}$$

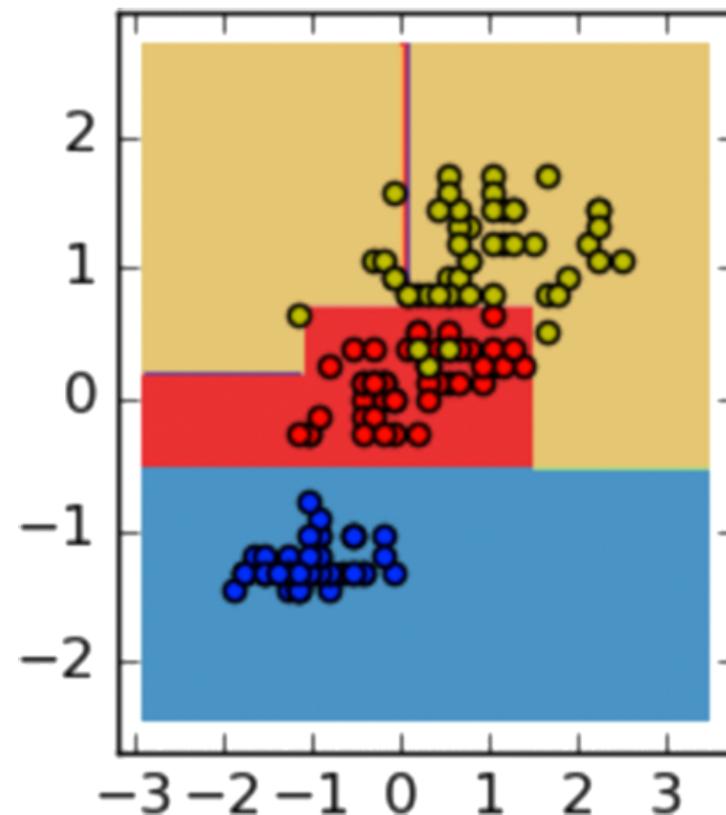
$$P(y = -1|x) = \frac{1}{1 + \exp(a_N(x))}$$

РЕЗЮМЕ

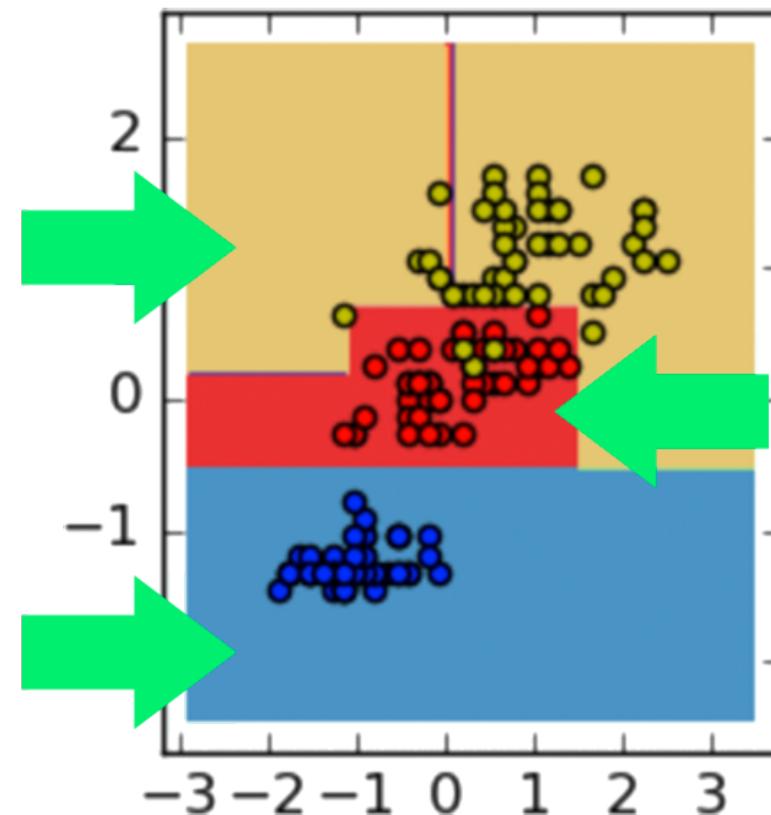
- › Градиентный бустинг над деревьями
- › Вид для MSE и логистических потерь

ГРАДИЕНТНЫЙ БУСТИНГ НАД РЕШАЮЩИМИ ДЕРЕВЬЯМИ

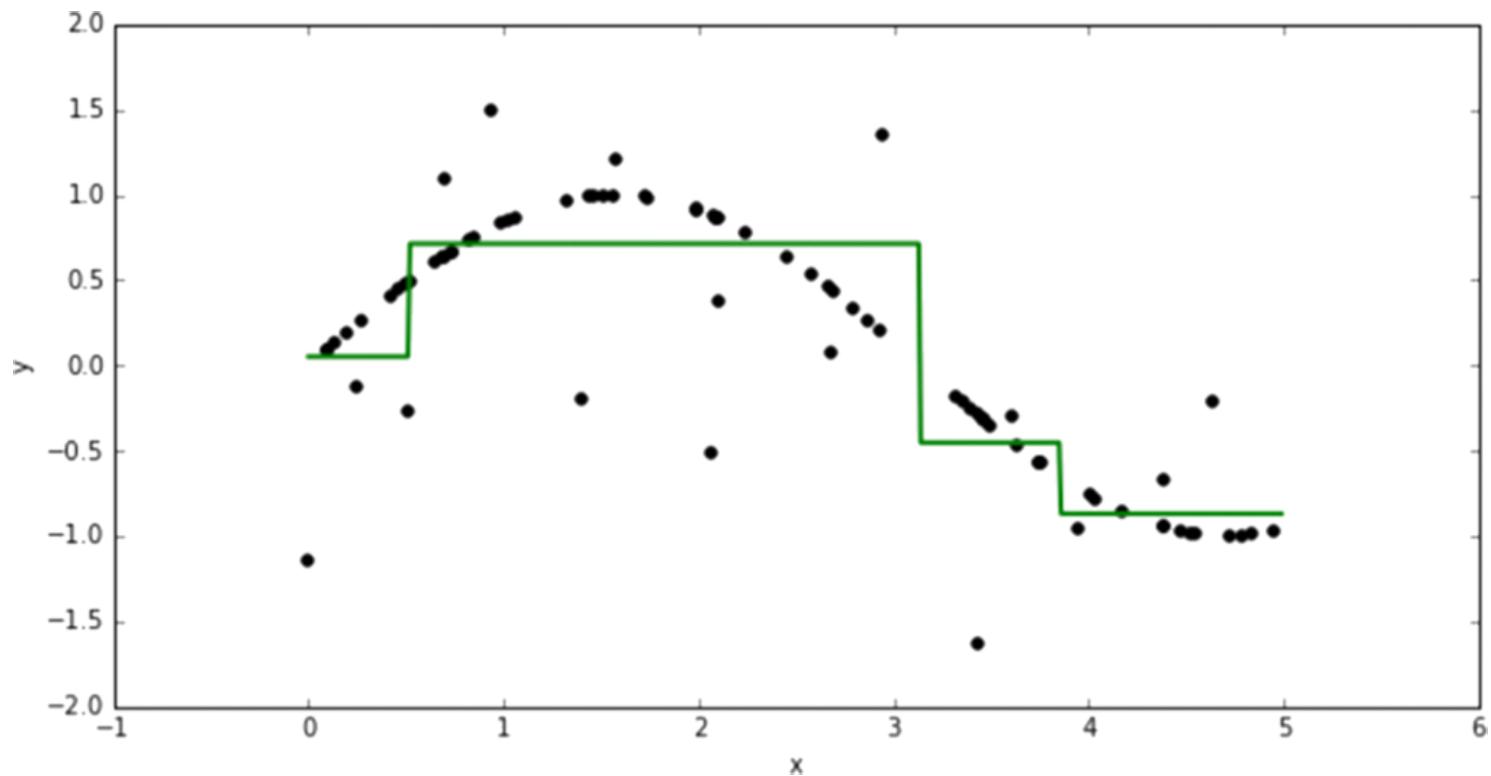
РЕШАЮЩЕЕ ДЕРЕВО



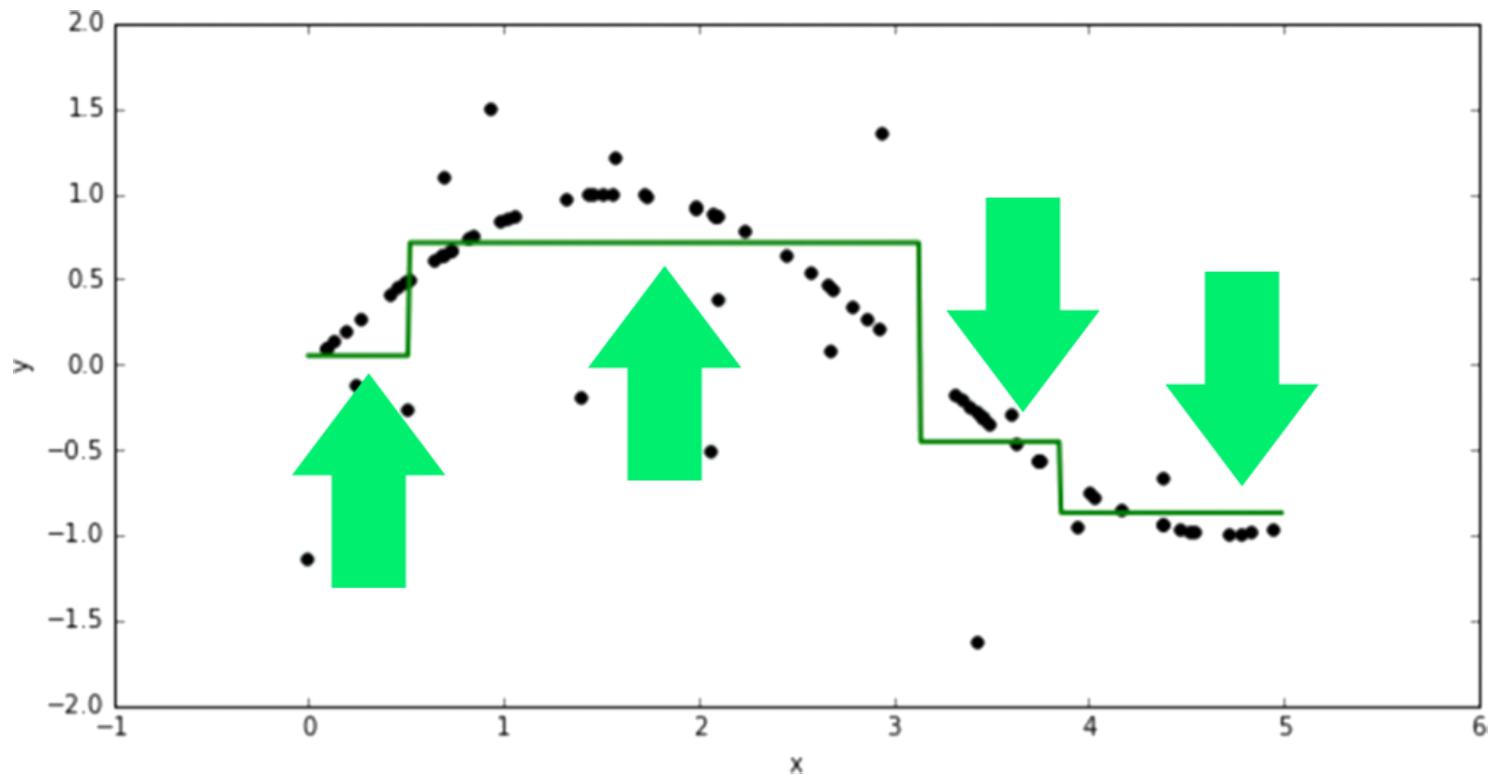
РЕШАЮЩЕЕ ДЕРЕВО



РЕШАЮЩЕЕ ДЕРЕВО



РЕШАЮЩЕЕ ДЕРЕВО



РЕШАЮЩЕЕ ДЕРЕВО

- › Разбивает пространство объектов на области R_1, \dots, R_J
- › В каждой области — константный ответ b_1, \dots, b_J

$$b(x) = \sum_{j=1}^J [x \in R_j] b_j$$

ГРАДИЕНТНЫЙ БУСТИНГ

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

$$b_N(x) = \sum_{j=1}^J [x \in R_{Nj}] b_{Nj}$$

ГРАДИЕНТНЫЙ БУСТИНГ

$$a_N(x) = a_{N-1}(x) + \sum_{j=1}^J [x \in R_{Nj}] b_{Nj}$$



Простейшие алгоритмы

ГРАДИЕНТНЫЙ БУСТИНГ

$$a_N(x) = a_{N-1}(x) + \sum_{j=1}^J [x \in R_{Nj}] b_{Nj}$$

- › Перенастроим b_{Nj} так, чтобы оптимизировать исходную функцию потерь L

ПЕРЕНАСТРОЙКА В ЛИСТЬЯХ

$$\sum_{i=1}^{\ell} L\left(y_i, a_{N-1}(x) + \sum_{j=1}^J [x \in R_{Nj}] b_j\right) \rightarrow \min_{b_1, \dots, b_J}$$

ПЕРЕНАСТРОЙКА В ЛИСТЬЯХ

$$b_{Nj} = \operatorname{argmin}_{\gamma \in \mathbb{R}} \sum_{x_i \in \mathbb{R}_j} L(y_i, a_{N-1}(x) + \gamma)$$

ПЕРЕНАСТРОЙКА В ЛИСТЬЯХ

- › Структура дерева настраивается по MSE
- › Прогнозы в листьях подбираются под исходную функцию потерь
- › Ускоряет сходимость — нужно меньше деревьев

ПРИМЕР

- › Классификация
- › Логистическая функция потерь

$$b_{Nj} = - \frac{\sum_{x_i \in \mathbb{R}_j} s_i}{\sum_{x_i \in \mathbb{R}_j} |s_i|(1 - |s_i|)}$$

РЕЗЮМЕ

- › Решающее дерево — кусочно-постоянный алгоритм
- › Настраивается на MSE
- › Можно перенастроить прогнозы в листьях
- › Повышает скорость сходимости