# Gradient Boosting – идея метода
# Вер 2 09.2019

## 1. Вспомним метод скорейшего спуска.

Задача: минимизация функции $f(t_{1,}t_{2,}\dots t_k)$

Задача: найти $argmin(f(t_{1,}t_{2,}\dots t_k))$

Временно рассмотрим функции одного аргумента

$f(t)$

$t_0$ - начальное приближение

$\lambda$ - скорость обучения (learning rate)

$$t_{i+1}=t_i-\lambda\cdot f'(t_i)$$

Получаем последовательность точек

$t_0$

$t_1=t_0-\lambda_0\cdot f'(t_0)$

$t_2=t_1-\lambda_1\cdot f'(t_1)=t_0-\lambda_0\cdot f'(t_0)-\lambda_1\cdot f'(t_1)$

$t_3=t_2-\lambda_2\cdot f'(t_2)=t_0-\lambda_0\cdot f'(t_0)-\lambda_1\cdot f'(t_1)-\lambda_2\cdot f'(t_2)$

$t_4=t_3-\lambda_3\cdot f'(t_3)=t_0-\lambda_0\cdot f'(t_0)-\lambda_1\cdot f'(t_1)-\lambda_2\cdot f'(t_2)-\lambda_3\cdot f'(t_3)$

...

$$t_{k+1}=t_k-\lambda_k\cdot f'(t_k)=t_0-\lambda_0\cdot f'(t_0)-\lambda_1\cdot f'(t_1)-\lambda_2\cdot f'(t_2)-\dots-\lambda_k\cdot f'(t_k)$$

## 2. Boosting – последовательное улучшение моделей

Пример, когда следующая модель строится для остатков
Регрессия для решения задач классификации

## 3. Вспомним наш подход к построению моделей.

Имеем семейство функций
В этом семейсве выбираем функцию ***h*** так, чтобы по иксам «хорошо» распознавать игреки, по предикторам распознавать отклик:

$$h(x_i)=\hat{y}_i\approx y_i$$

Искомая функция ***h*** будет минимизировать критерий качества (с оговорками, какими?)

Критерий качества $Q=\frac{1}{n}\cdot\sum_{i=1}^{n}L(y_i,h(x_i))=\frac{1}{n}\cdot\sum_{i=1}^{n}L(y_i,\hat{y}_i)$

Примеры критериев качества

1. MSE $L_1=(y_i-h(x_i))^2$
2. MAD $L_2=|y_i-h(x_i)|$
3. LogLoss $L_3=-[y_i\cdot\log(h(x_i))+(1-y_i)\cdot\log(1-h(x_i))]$

Дополнительно предположим, что у L имеется первая производная.

Что мы меняем, чтобы минимизировать ***Q***?

НЕ $x_i$ - это наблюдения, они фиксированные.

НЕ $y_i$ - это наблюдения, они фиксированные.

НЕ функцию ***L***, она выбирается в самом начале анализа.

Подбираем функцию ***h.***

Далее — среди всех сумм деревьев регрессии

Но не среди всех деревьев классификации.

**4. Объединим эти две идеи.**

Так же, как и в методе скорейшего спуска, где

$t_{k+1}=t_k-\lambda_k\cdot f'(t_k)=t_0-\lambda_0\cdot f'(t_0)-\lambda_1\cdot f'(t_1)-\lambda_2\cdot f'(t_2)-...-\lambda_k\cdot f'(t_k)$

ищем модель в виде суммы подмоделей

$H^k=h_0+\lambda_0\cdot h_1+\lambda_1\cdot h_1+\lambda_2\cdot h_1+...+\lambda_k\cdot h_k$

При этом $H^k$ модель, полученная комбинированием *k* простых моделей, $\lambda_i$ - числа, веса простых моделей, $h_i$ - функции, комбинируемые модели, вроде тех, которые мы изучали ранее, почти всегда на практике это деревья регрессии.

Шаг 1. Выбор начального значения

Вместо $t_0$ будет некоторая функция $h_0$ . Она постоянная, ее значение всегда равно одному и тому же числу.

Функцию $h_0$ вроде надо выбирать произвольной, но можно поступать лучше.

В задачах регрессии часто полагают $h_0=\frac{1}{n}\cdot\sum_{i=1}^{n}y_i$

В задачах распознавания двух классов «0» и «1», когда ***h*** интерпретируем как вероятность того, что объект принадлежит классу «1», часто полагают $h_0 = \frac{1}{2}$ .

Совсем правильно найти число $h_0$ минимизируя $Q = \frac{1}{n} \cdot \sum_{i=1}^{n} L(y_i, h_0)$

Объяснение как бы по индукции.

Предположим, что построена функция $H^k()$ .
Как найти следующую модели $H^{k+1}()$ ?

$$H^{k+1} = H^k + \lambda_k \cdot h_{k+1} = (h_0 + \lambda_0 \cdot h_1 + \lambda_1 \cdot h_1 + \lambda_2 \cdot h_1 + ... + \lambda_k \cdot h_k) + \lambda_k \cdot h_{k+1}$$

Что теперь будет аналогом $f'(t_k) = \frac{df}{dt}(t_k)$ из формулы $t_{k+1} = t_k - \lambda_0 \cdot f'(t_k)$
Friedman предложил использовать

$$r_{ik} = -\frac{\partial L(y_i, h_k(x_i))}{\partial h_k(x_i)}$$

Например для MISE $Q = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - h(x_i))^2$

$$r_{ik} = -2 \cdot (y_i - h_k(x_i))$$

Теперь организуем матрицу, у которой первые столбцы как у исходной, а вместо столбца со значениями отклика — столбец из $r_{ik}$ . Строки, как и ранее соответствуют наблюдениям.

Как определять очередное значение $\lambda$ ?
Для каждого слабого классификатора индивидуально, так, чтобы минимизировать *Q*.