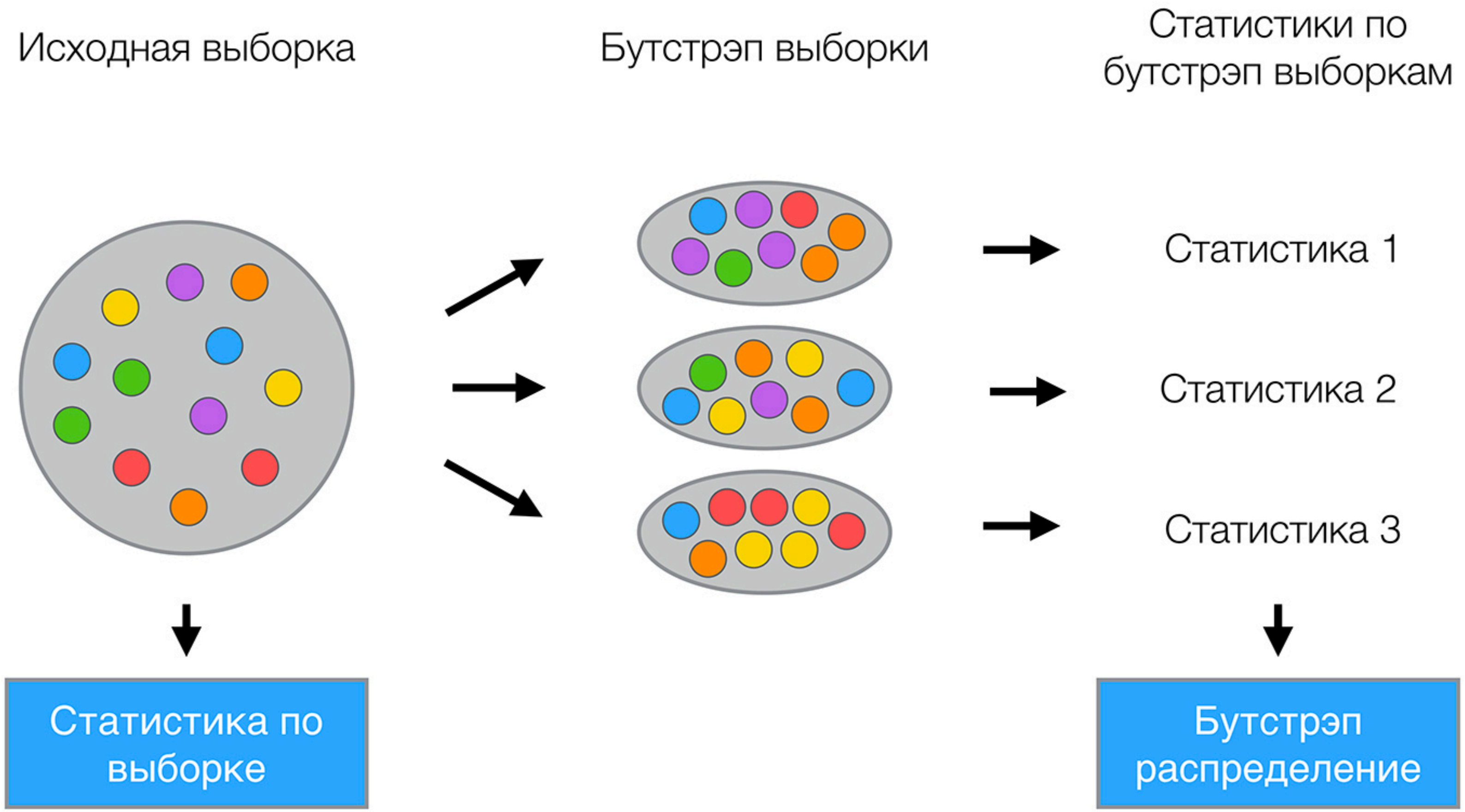
Ансамбли моделей. Градиентный бустинг

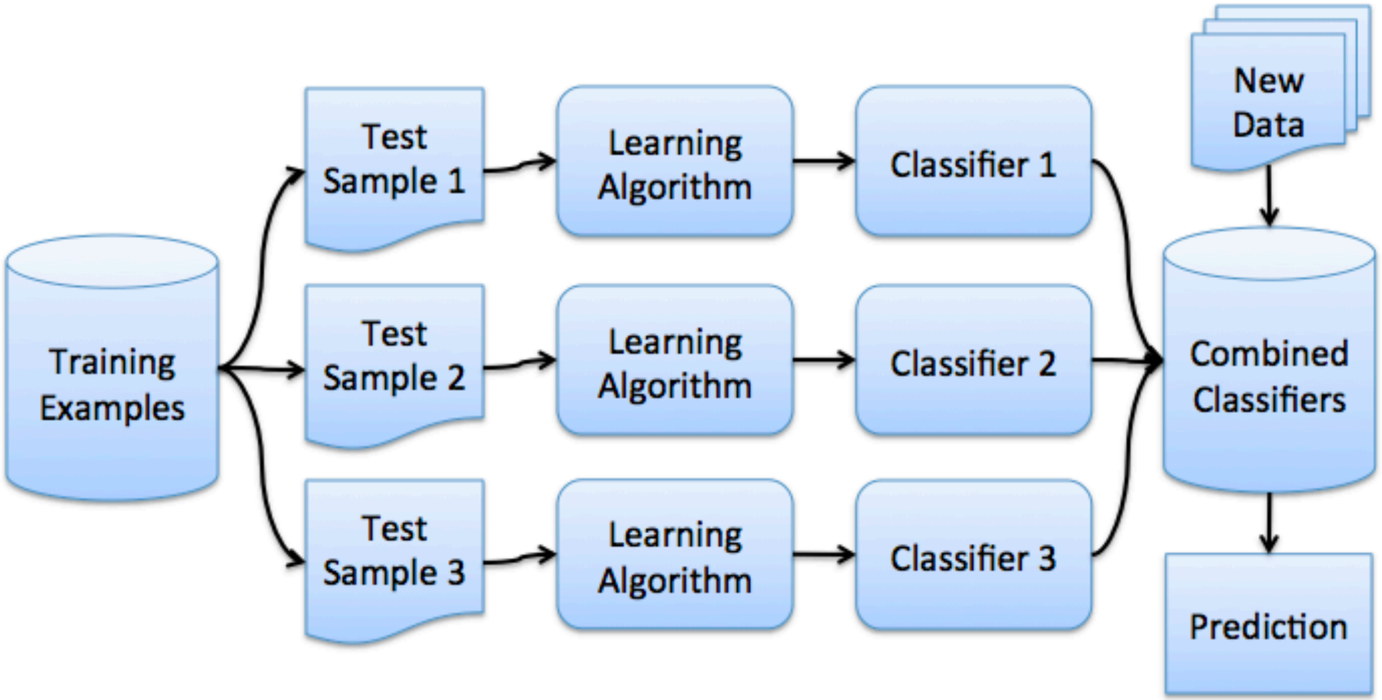
МАКСИМОВСКАЯ АНАСТАСИЯ

Бутстрэп



Источник: [URL]

Бэггинг



Источник: [URL]()

Бэггинг

- Если алгоритмы $b_1(x), \dots, b_n(x)$ некоррелированы, то среднеквадратичная ошибка алгоритма a(x), полученного при помощи бэггинга, в N раз меньше среднеквадратичной ошибки исходных алгоритмов $b_j(x)$.

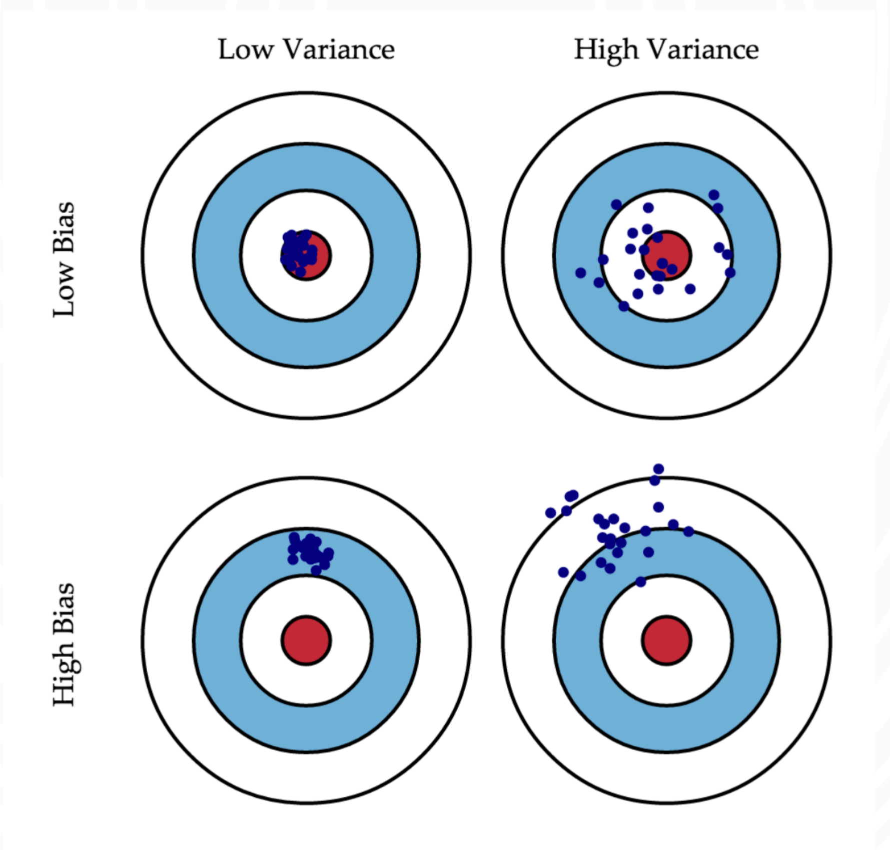
$$a(x) = \frac{1}{N}\sum_{j=1}^{N} b_j(x)$$

Разложение ошибки на смещение и разброс (bias-variance decomposition)

Ошибку модели ($a(x)$ можно представить как:

$$Error(x) = Bias^2(a(x)) + Var\left(a(x)\right) + \sigma^2$$

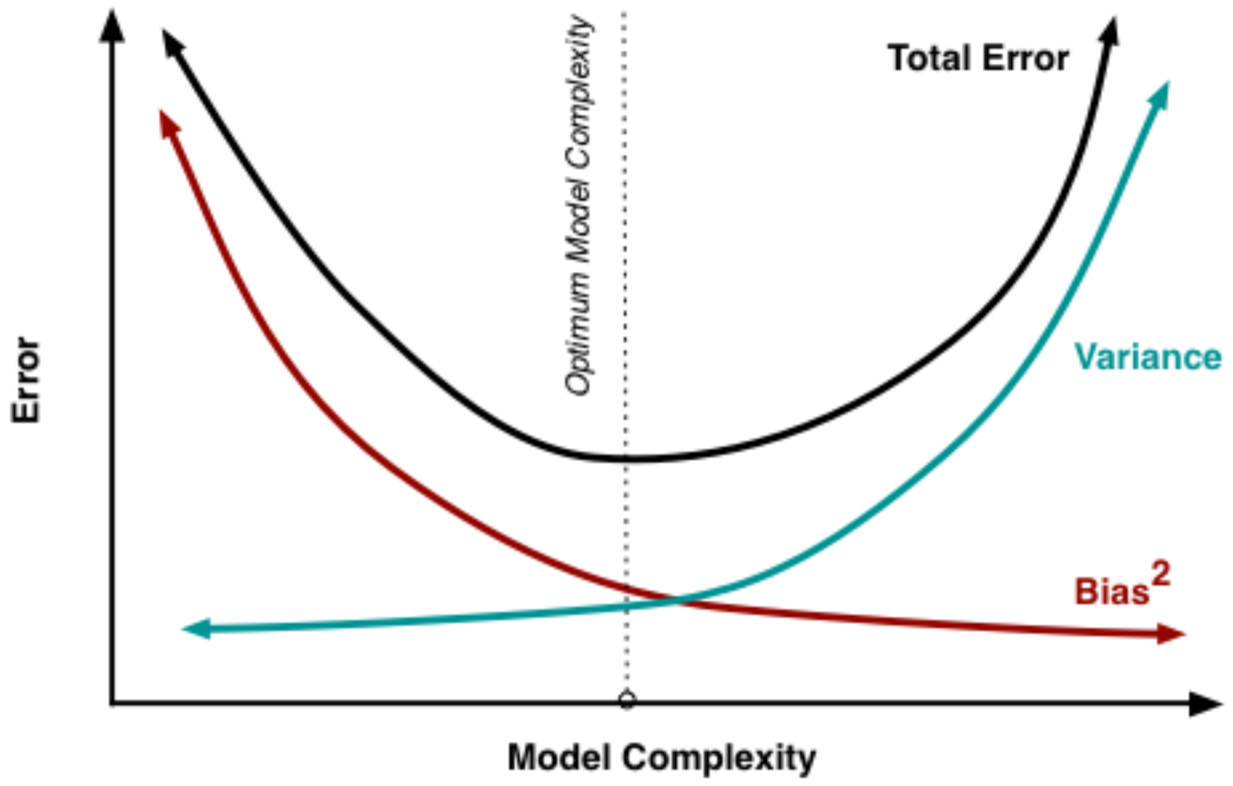
Смещение и разброс



Bias-Variance Trade-Off

- У простой модели (например, линейная регрессия) обычно большое смещение и маленький разброс
- Чем сложнее модель (чем больше у неё настраиваемых параметров), тем меньше у неё смещение и тем больше разброс

Bias-Variance Trade-Off



Bias и Variance в бэггинге

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

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

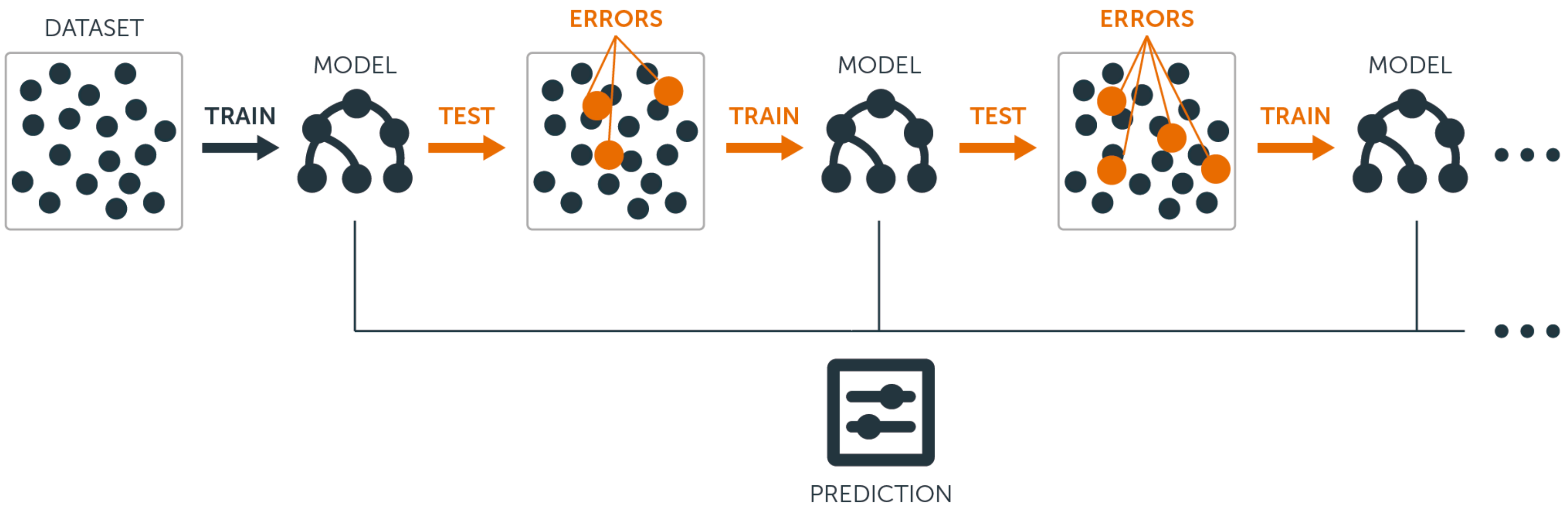
Алгоритм случайного леса

- Алгоритм построения случайного леса, состоящего из N деревьев, выглядит следующим образом:

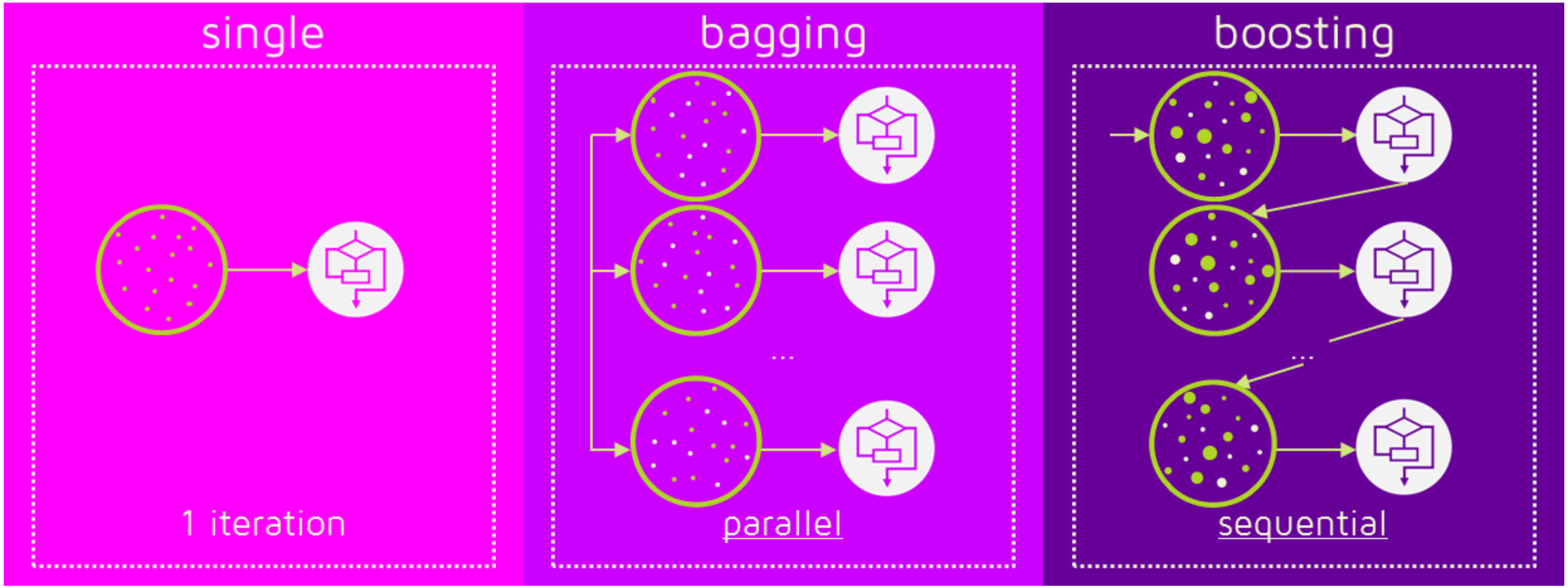
- Для каждого $n = 1, \dots, N$:
  - Сгенерировать выборку X_n с помощью бутстрэпа;
  - Построить решающее дерево b_n по выборке X_n :
    — по заданному критерию мы выбираем лучший признак, делаем разбиение в дереве по нему и так до исчерпания выборки
    — дерево строится, пока в каждом листе не более n_{min} объектов или пока не достигнем определенной высоты дерева
    — при каждом разбиении сначала выбирается m случайных признаков из n исходных, и оптимальное разделение выборки ищется только среди них.

Бустинг

- Строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих



Бустинг



Бустинг для регрессии

- Решаем задачу регрессии с минимизацией квадратичной ошибки:

$$\frac{1}{2}\sum_{i=1}^{l}(a(x_i) - y_i)^2 \to \min_{a}$$

Бустинг для регрессии

- 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 для регрессии

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:

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

Бустинг для регрессии

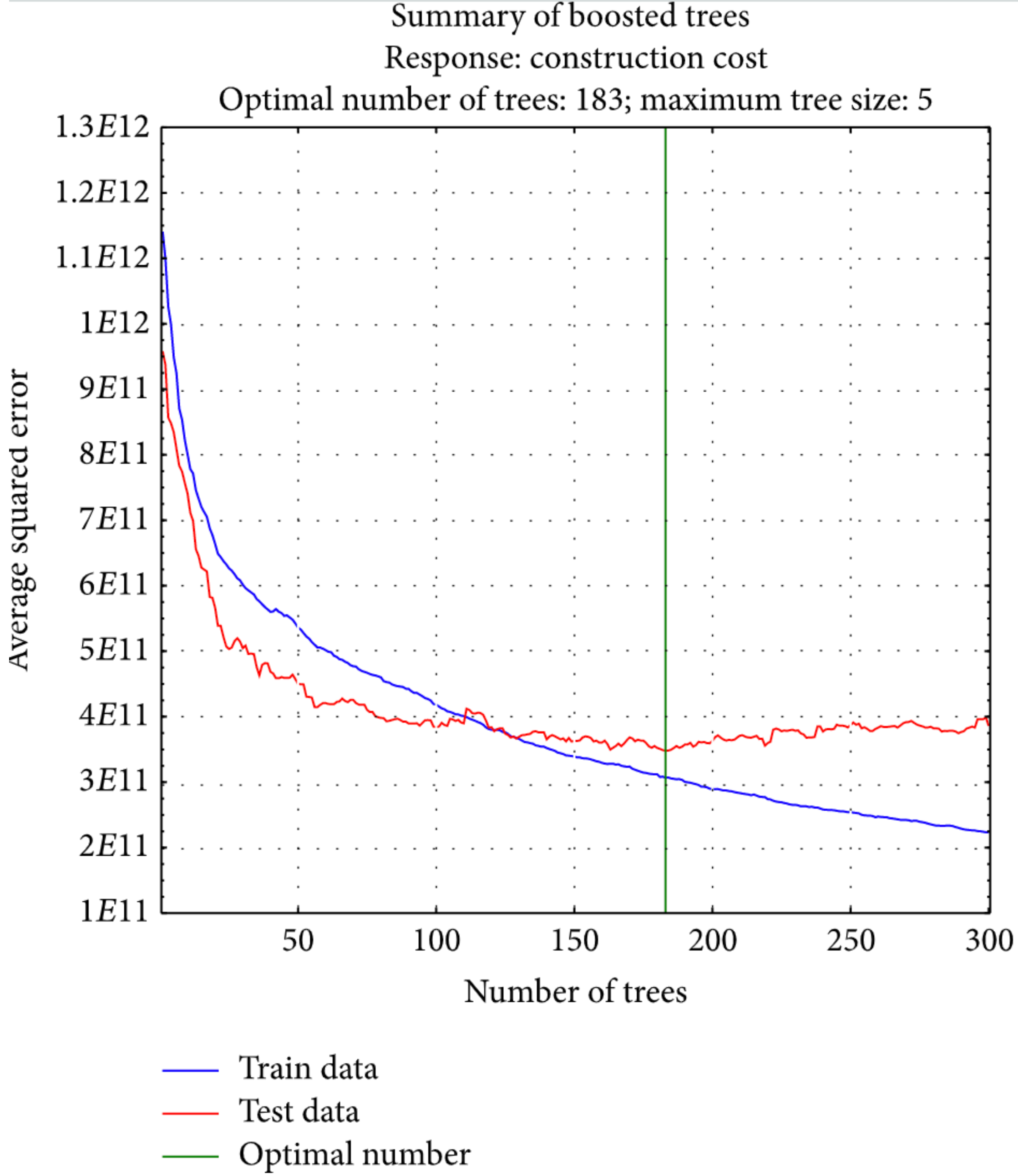
2. Ищем алгоритм $b_2(x)$, настраивающийся на ошибки s первого алгоритма:

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

3. Алгоритм $b_3(x)$ будем настраивать на ошибку предыдущей композиции $b_1(x) + b_2(x)$

Базовые алгоритмы

- 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



Summary of boosted trees
Response: construction cost
Optimal number of trees: 183; maximum tree size: 5
1.3E12
1.2E12
1.1E12
1E12
9E11
8E11
7E11
6E11
5E11
4E11
3E11
2E11
1E11
Average squared error
50
100
150
200
250
300
Number of trees
Train data
Test data
Optimal number

Градиентный бустинг

- Пусть дана некоторая дифференцируемая функция потерь $L(y, z)$. Будем строить взвешенную сумму базовых алгоритмов:

$$a_N(x) = \sum_{n=0}^{N} \gamma_n b_n(x)$$

Начальный алгоритм

Коэффициент при начальном алгоритме обычно берут 1, а сам начальный алгоритм можно задать как:

- 0

$$b_0(x) = 0$$

- Самый популярный класс (в задаче классификации)

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

- Средний ответ (в задаче регрессии)

$$b_0(x) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$$

Градиентный бустинг

- Строя композицию каждый следующий алгоритм выбираем так, чтобы как можно сильнее уменьшать ошибку:

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + \gamma_N b_N(x_i)) \to \min_{b_N, \gamma_N}$$

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

Градиентный бустинг

- Требуем, чтобы сдвиг был противоположен производной функции потерь в точке $z = a_{N-1}(x_i)$

$$s_i = -\left.\frac{\partial L}{\partial z}\right|_{z=a_{N-1}(x_i)}$$

- Двигаемся в сторону убывания функции потерь

Градиентный бустинг

- Такой вектор сдвигов совпадает с антиградиентом:

$$\left(-\frac{\partial L}{\partial z}\Big|_{z=a_{N-1}(x_i)} \right)_{i=1}^{\ell} = -\nabla_z \sum_{i=1}^{\ell} L(y_i, z_i)\Big|_{z_i=a_{N-1}(x_i)}$$

Градиентный бустинг

- Возьмем среднеквадратичную ошибку для поиска базового алгоритма, приближающего градиент функции потерь на обучающей выборке:

$$b_N(x) = \underset{b \in \mathcal{A}}{\arg\min} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2$$

Градиентный бустинг

- После того, как новый базовый алгоритм найден, можно подобрать коэффициент при нем по аналогии с наискорейшим градиентным спуском:

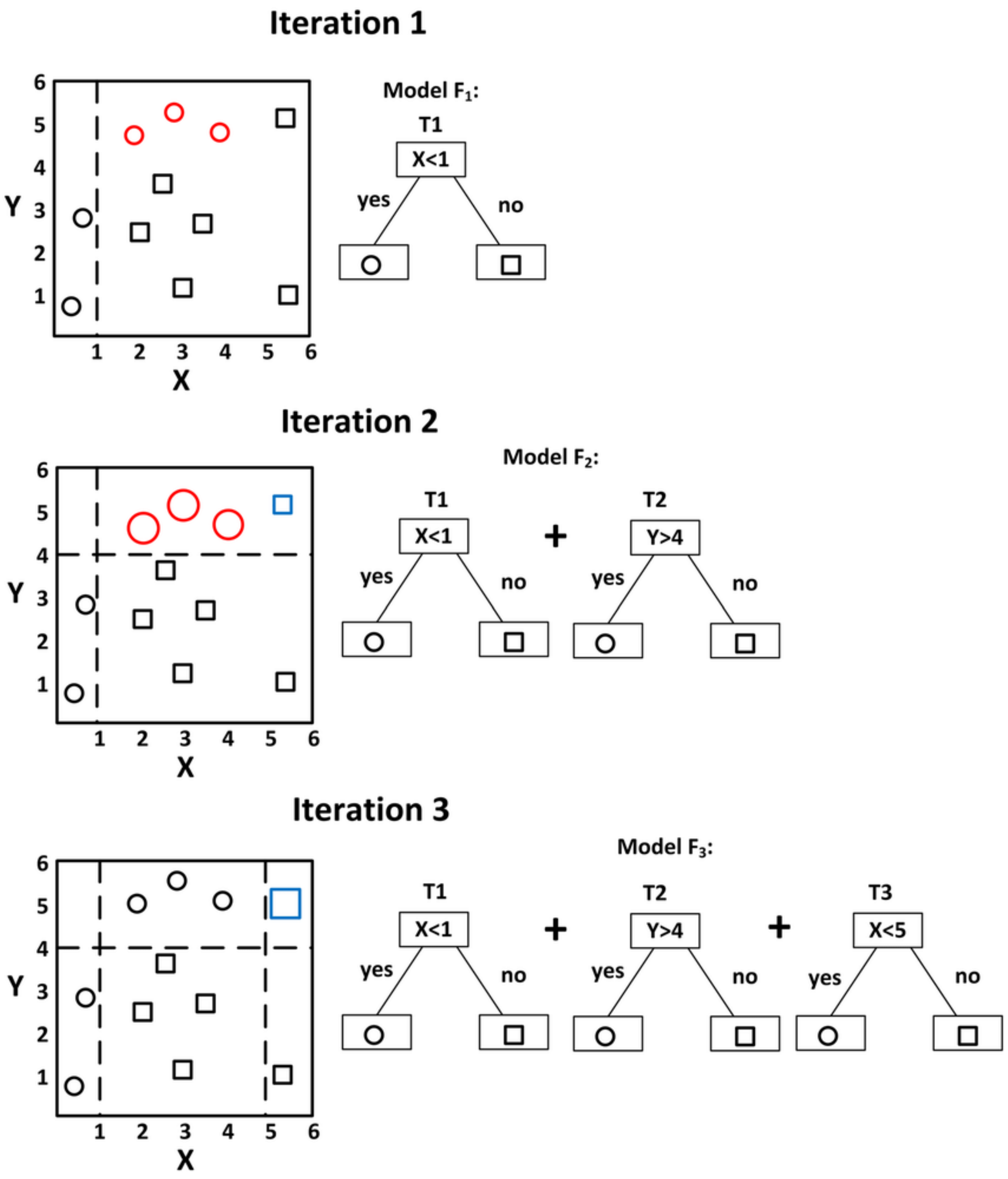
$$\gamma_N = \arg\min_{\gamma \in \mathbb{R}} \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + \gamma b_N(x_i))$$

Algorithm 1: Gradient Boost

- Initialize $F_0(x) = \arg\min_\rho \sum_{i-1}^{N} L(y_i, \rho)$
- For $m = 1$ to M do:
  - Step 1. Compute the negative gradient
  $$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F_{x_i}}\right]$$
  - Step 2. Fit a model
  $$\alpha_m = \arg\min_{\alpha,\beta} \sum_{i=1}^{N} [\tilde{y} - \beta h(x_i; \alpha_m)]^2$$
  - Step 3. Choose a gradient descent step size as
  $$\rho_m = \arg\min_{\rho} \sum_{i-1}^{N} L(y_i, Fm - 1(x_i) + \rho h(x_i; \alpha))$$
  - Step 4. Update the estimation of $F(x)$
  $$F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m)$$
- end for
- Output the final regression function $F_m(x)$

Fig. 1. Gradient boosting algorithm

Градиентный бустинг над деревьями



Iteration 1
Model F_1:
T1
X<1
yes
no
Y
X
1 2 3 4 5 6
Iteration 2
Model F_2:
T1
X<1
+
T2
Y>4
yes
no
Iteration 3
Model F_3:
T1
X<1
+
T2
Y>4
+
T3
X<5
yes
no

Градиентный бустинг

age	square footage	location	price
5	1500	5	480
11	2030	12	1090
14	1442	6	350
8	2501	4	1310
12	1300	9	400
10	1789	11	500

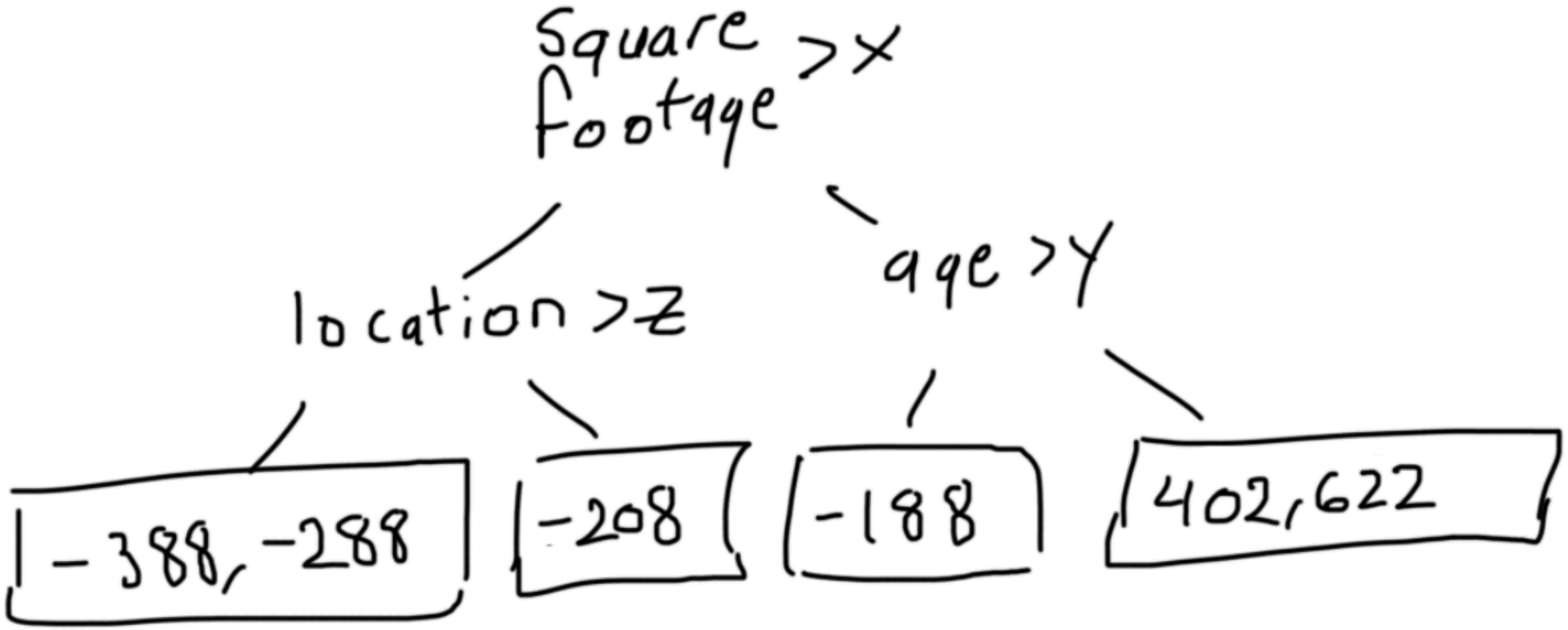
Градиентный бустинг

$$\frac{490 + 1090 + 350 + 1310 + 400 + 500}{6} = 688$$

Градиентный бустинг

age	square footage	location	price	residuals
5	1500	5	480	-208
11	2030	12	1090	402
14	1442	6	350	-338
8	2501	4	1310	622
12	1300	9	400	-288
10	1789	11	500	-188

Градиентный бустинг



Градиентный бустинг

$$a_N(x) = \sum_{n=0}^{N} \gamma_n b_n(x)$$

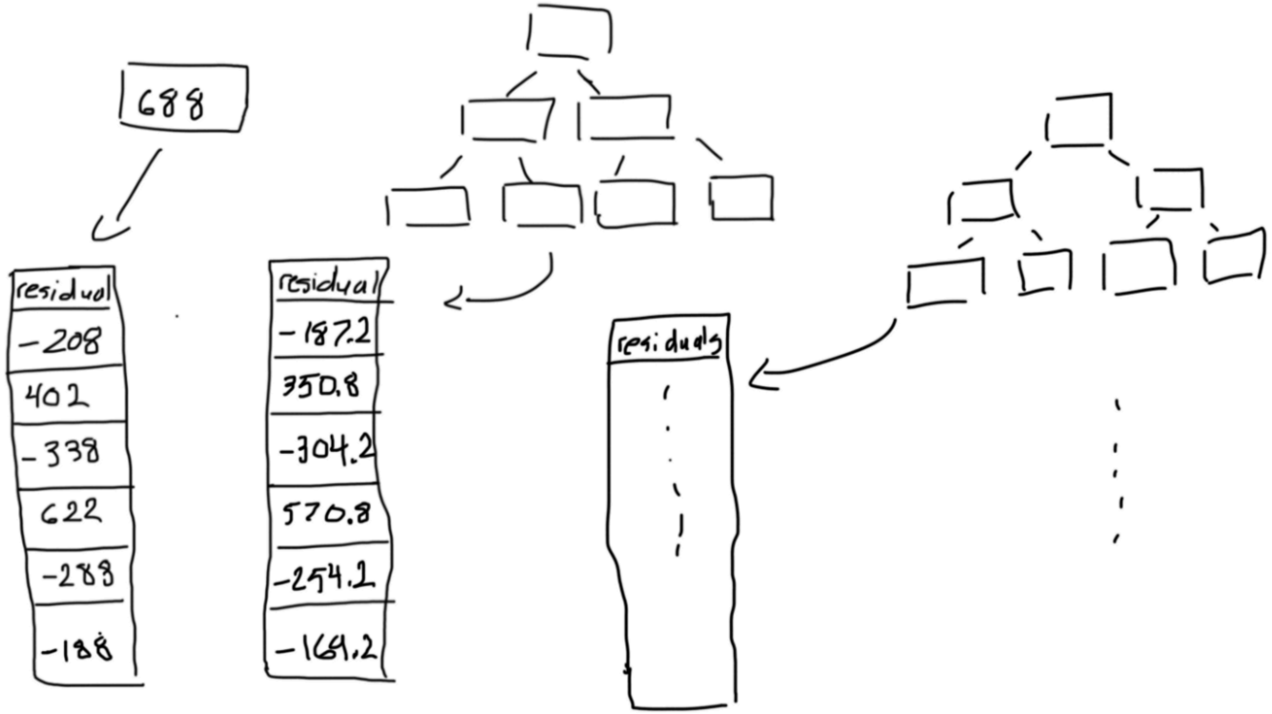
$$\overset{\text{Average price}}{\boxed{688}} + \underset{=0.1}{\text{Learning Rate}} \times \overset{\text{Residual predicted by decision tree}}{\boxed{-338}}$$

$$\text{predicted price} = 688 + 0.1 \times -338 = 654.2$$

Градиентный бустинг

$$\text{residual} = \text{actual price} - \text{predicted price} = 350 - 654.2 = -304.2$$

Градиентный бустинг



Градиентный бустинг

$$a_N(x) = \sum_{n=0}^{N} \gamma_n b_n(x)$$

$$\overset{\text{Average price}}{\boxed{688}} + \underset{=0.1}{\text{Learning Rate}} \times \overset{\text{Residual predicted by decision tree}}{\boxed{-188}} + \underset{=0.1}{\text{learning Rate}} \times \overset{\text{Residual predicted by decision tree}}{\boxed{-169.2}} + \ldots$$

Регуляризация

- Сокращение шага

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

- Число итераций

Регуляризация

- Стохастический градиентный бустинг: обучаем алгоритм $b_n(x)$ не по всей выборке, а по ее подмножеству
- Понижается уровень шума
- Повышается эффективность вычислений
- Обычно берут подвыборки в 2 раза меньше исходной

Функции потерь: регрессия

- Квадратичная функция потерь (рассмотрена выше)
- Модуль отклонения $L(y, z) = |y - z|$. Антиградиент считается по формуле:

$$s_i^{(N)} = -\operatorname{sign}(a_{N-1}(x_i) - y_i)$$

Функции потерь: классификация

- Логистическая: $L(y, z) = \log(1 + \exp(-yz))$
- Тогда задача поиска базового алгоритма:

$$b_N = \underset{b \in \mathcal{A}}{\arg\min} \sum_{i=1}^{\ell} \left(b(x_i) - \frac{y_i}{1 + \exp(y_i a_{N-1}(x_i))} \right)^2$$