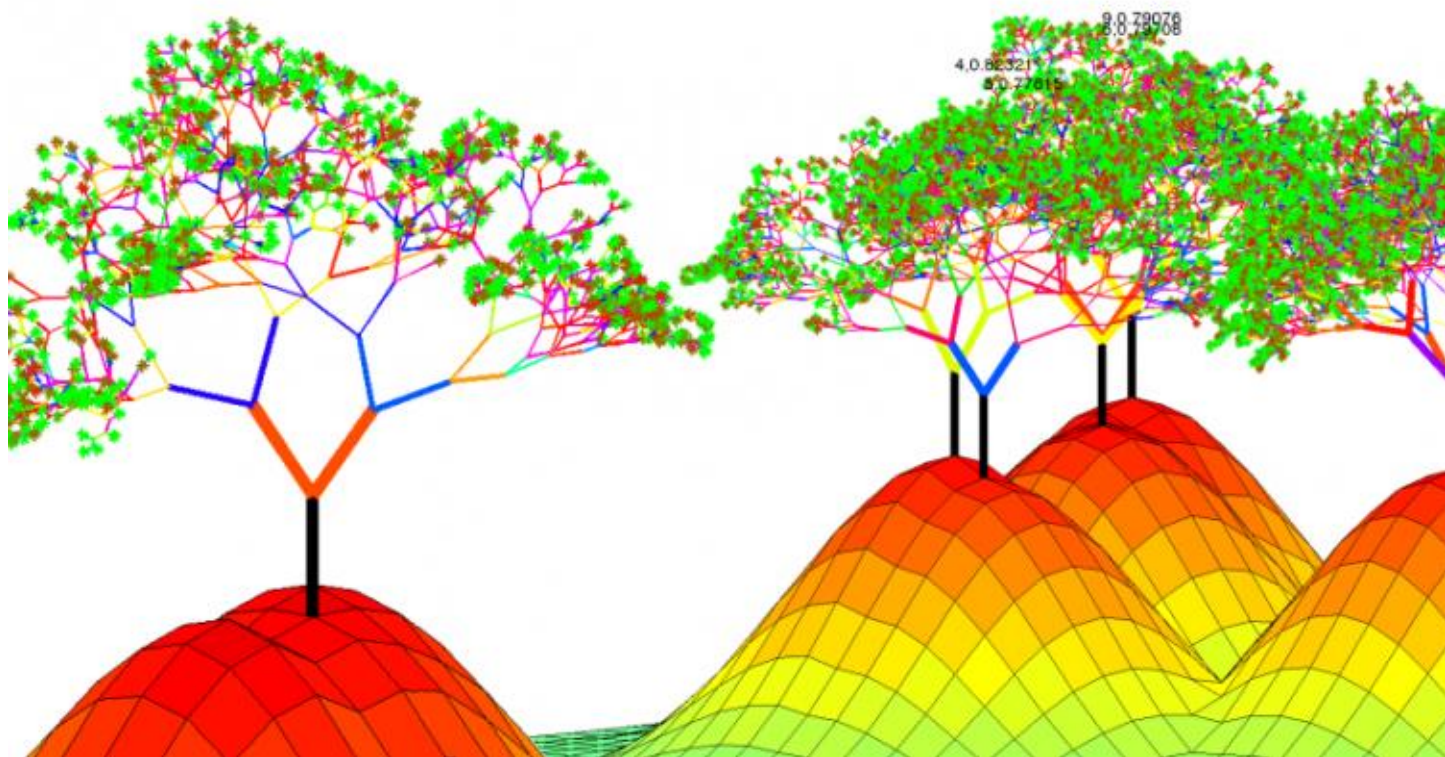


# Композиции алгоритмов

Елена Кантонистова



# РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

- *Модель переобучена?*
- *Модель плохо предсказывает целевую переменную?*
- *В самих данных много неточностей (шумов)*

# РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

**Утверждение:** *ошибку модели  $a(x)$  можно представить в виде*

$$\text{Err}(x) = \text{Bias}^2(a(x)) + \text{Var}(a(x)) + \sigma^2.$$

# РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

**Утверждение:** ошибку модели  $a(x)$  можно представить в виде

$$\text{Err}(x) = \text{Bias}^2(a(x)) + \text{Var}(a(x)) + \sigma^2.$$

- $\text{Bias}(a(x))$  - средняя ошибка по всем возможным наборам данных – **смещение**.

# РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

**Утверждение:** ошибку модели  $a(x)$  можно представить в виде

$$\text{Err}(x) = \text{Bias}^2(a(x)) + \text{Var}(a(x)) + \sigma^2.$$

- $\text{Bias}(a(x))$  - средняя ошибка по всем возможным наборам данных – **смещение**.

*Смещение показывает, насколько в среднем модель хорошо предсказывает целевую переменную:*

- ✓ *маленькое смещение - хорошее предсказание*
- ✓ *большое смещение – плохое предсказание*

# РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

**Утверждение:** ошибку модели  $a(x)$  можно представить в виде

$$\text{Err}(x) = \text{Bias}^2(a(x)) + \text{Var}(a(x)) + \sigma^2.$$

- $\text{Var}(a(x))$  - дисперсия ошибки, т.е. как сильно различается ошибка при обучении на различных наборах данных – **разброс**.

# РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

**Утверждение:** ошибку модели  $a(x)$  можно представить в виде

$$\text{Err}(x) = \text{Bias}^2(a(x)) + \text{Var}(a(x)) + \sigma^2.$$

- $\text{Var}(a(x))$  - дисперсия ошибки, т.е. как сильно различается ошибка при обучении на различных наборах данных – **разброс**.

*Большой разброс означает, что ошибка очень чувствительна к изменению обучающей выборки, т.е.:*

✓ *большой разброс – сильно переобученная модель*

# РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

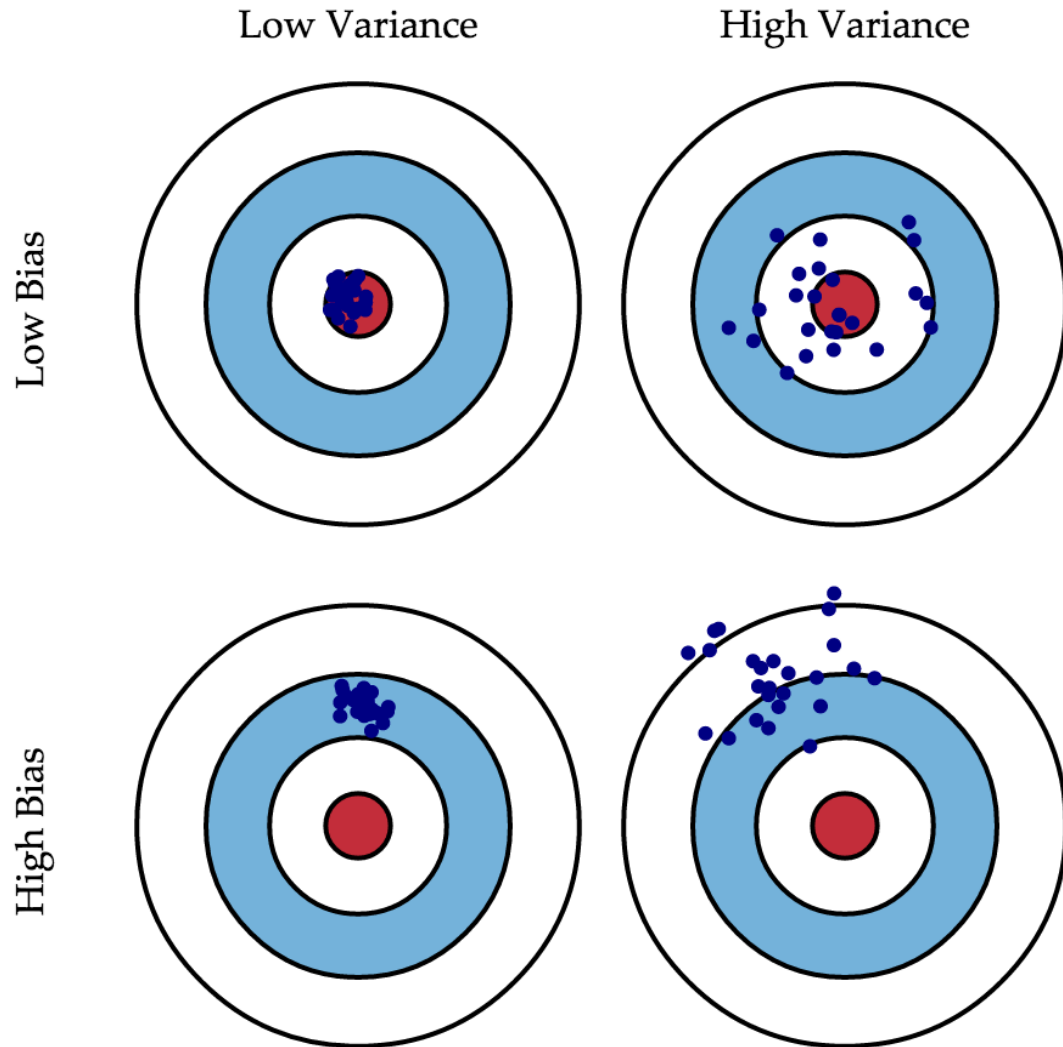
**Утверждение:** ошибку модели  $a(x)$  можно представить в виде

$$\text{Err}(x) = \text{Bias}^2(a(x)) + \text{Var}(a(x)) + \sigma^2.$$

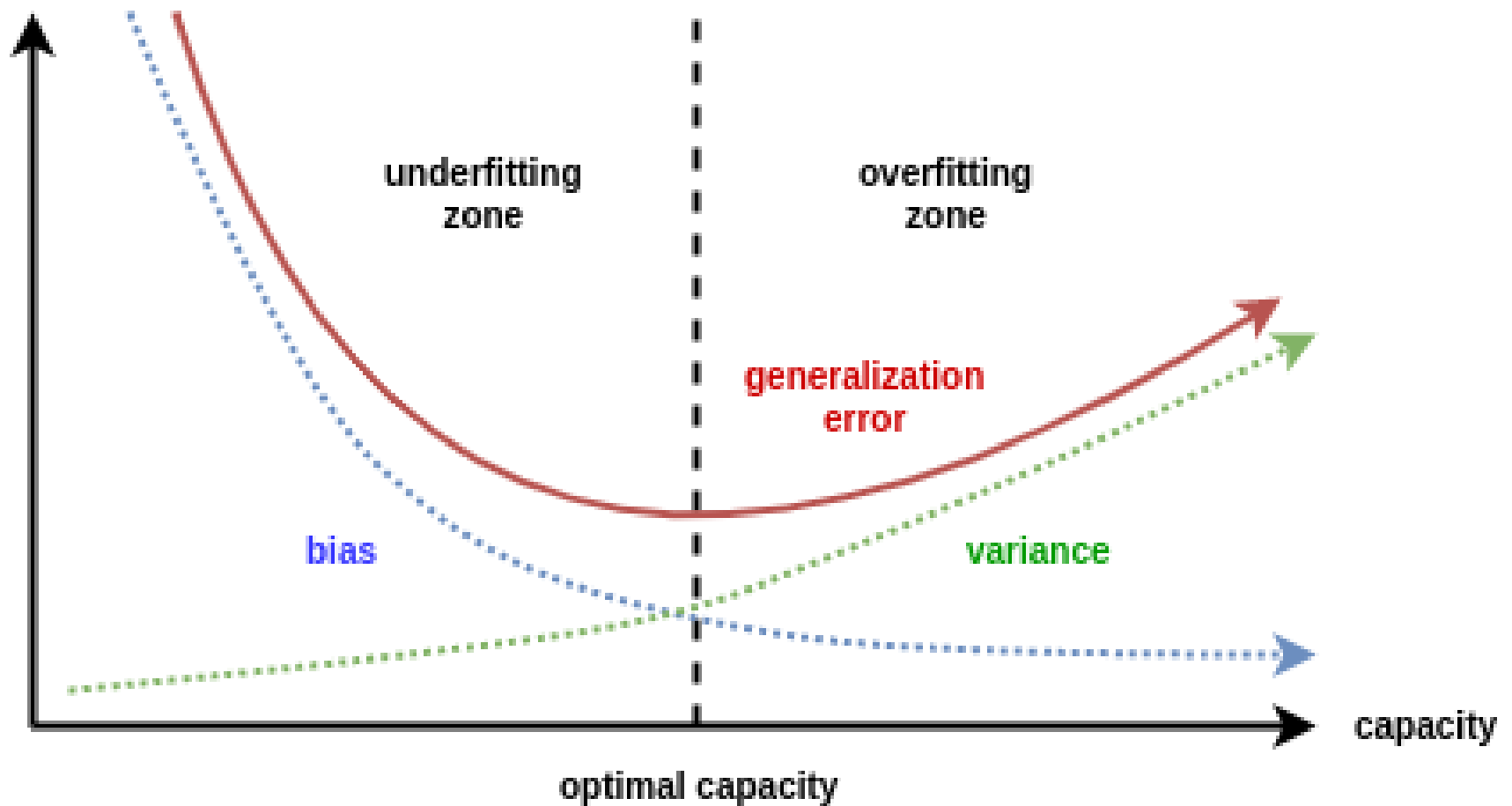
- $\text{Bias}(a(x))$  - средняя ошибка по всем возможным наборам данных – **смещение**.
- $\text{Var}(a(x))$  - дисперсия ошибки, т.е. как сильно различается ошибка при обучении на различных наборах данных – **разброс**.
- $\sigma^2$  - неустраняемая ошибка – **шум**.



# СМЕЩЕНИЕ И РАЗБРОС



# BIAS-VARIANCE TRADEOFF



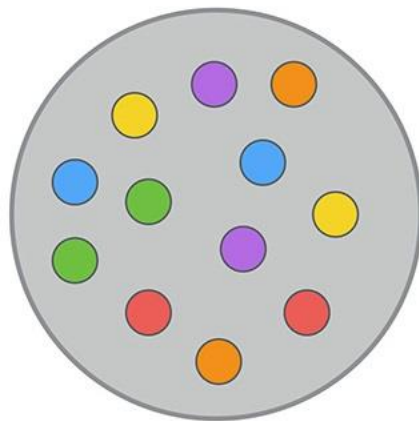
# БУТСТРЭП

Дана выборка  $X$ .

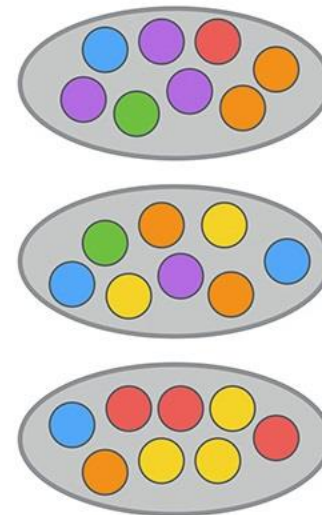
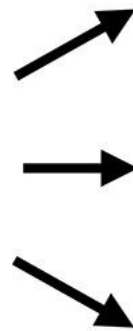
**Бутстрэп**: равномерно возьмем из выборки  $X$   $l$  объектов с возвращением (т.е. в новой выборке будут повторяющиеся объекты). Получим выборку  $X_1$ .

- Повторяем процедуру  $N$  раз, получаем выборки  $X_1, \dots, X_N$ .

Исходная выборка



Бутстрэп выборки



# БЭГГИНГ (BOOTSTRAP AGGREGATION)

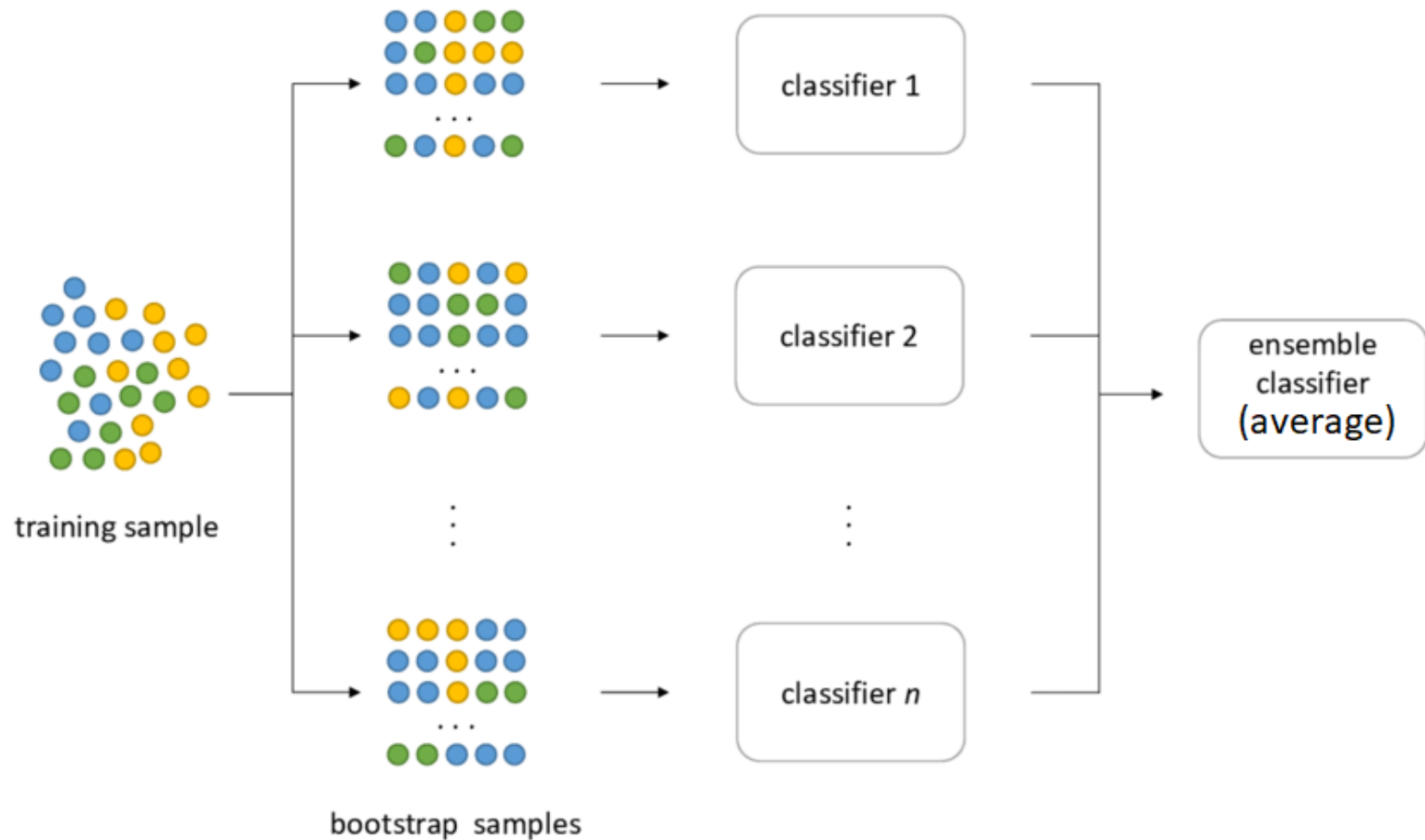
С помощью бутстрэпа мы получили выборки  $X_1, \dots, X_N$ .

- Обучим по каждой из них модель – получим базовые алгоритмы  $b_1(x), \dots, b_N(x)$ .
- Построим новую функцию регрессии:

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

# БЭГГИНГ (BOOTSTRAP AGGREGATION)

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



# СМЕЩЕНИЕ И РАЗБРОС У БЭГГИНГА

$$\text{Бэггинг: } a_N(x) = \frac{1}{N} \sum_{n=1}^N b_n(x) = \frac{1}{N} \sum_{n=1}^N \tilde{\mu}(X)(x)$$

(здесь  $\tilde{\mu}(X) = \mu(\tilde{X})$  – алгоритм, обученный на подвыборке  $\tilde{X}$ )

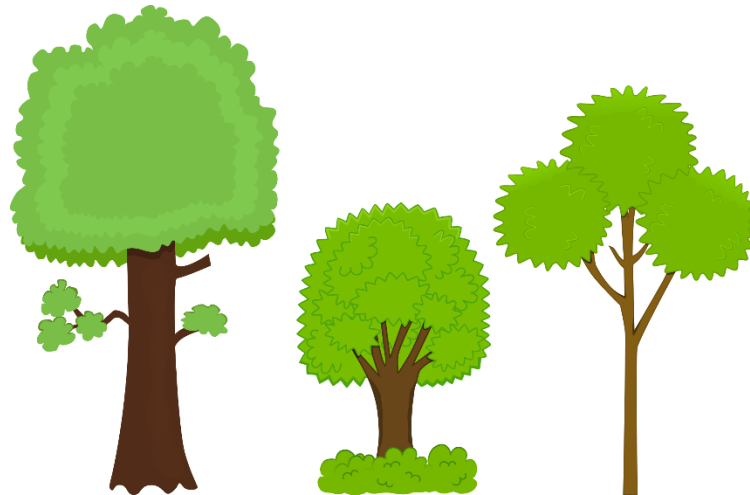
**Утверждение (с док-вом):**

*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.*

# СЛУЧАЙНЫЙ ЛЕС (RANDOM FOREST)

- Возьмем в качестве базовых алгоритмов для бэггинга **решающие деревья**, т.е. каждое случайное дерево  $b_i(x)$  построено по своей подвыборке  $X_i$ .
- В каждой вершине дерева будем искать **разбиение не по всем признакам, а по подмножеству признаков**.
- Дерево строится до тех пор, пока в листе не окажется  $n_{min}$  объектов.



# RANDOM FOREST

---

## Алгоритм 3.1. Random Forest

---

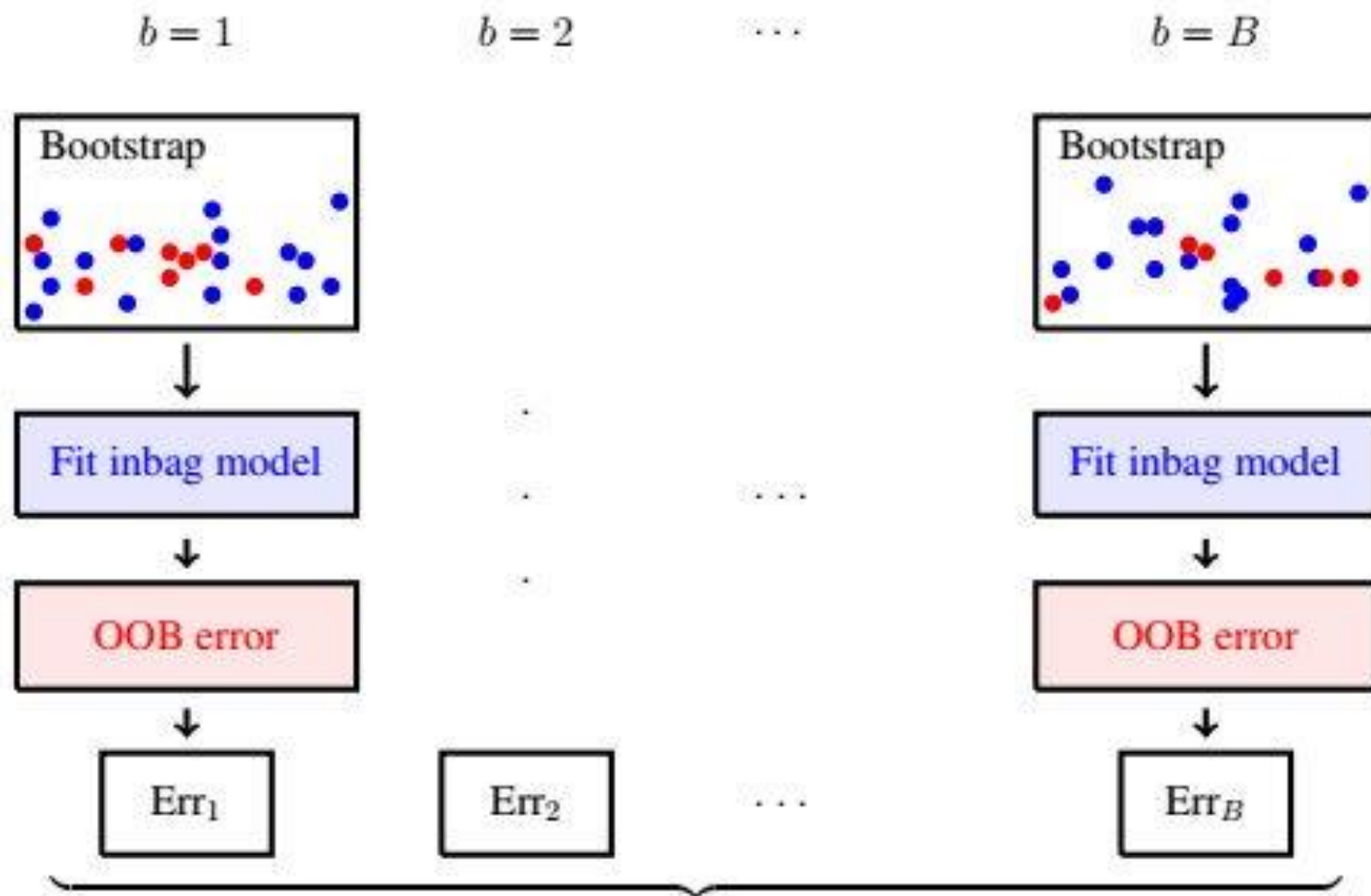
- 1: для  $n = 1, \dots, N$
  - 2: Сгенерировать выборку  $\tilde{X}_n$  с помощью бутстрэпа
  - 3: Построить решающее дерево  $b_n(x)$  по выборке  $\tilde{X}_n$ :
    - дерево строится, пока в каждом листе не окажется не более  $n_{\min}$  объектов
    - при каждом разбиении сначала выбирается  $m$  случайных признаков из  $p$ , и оптимальное разделение ищется только среди них
  - 4: Вернуть композицию  $a_N(x) = \frac{1}{N} \sum_{n=1}^N b_n(x)$
-



# RANDOM FOREST – ПРАКТИЧЕСКИЕ РЕКОМЕНДАЦИИ

- Если  $p$  – количество признаков, то при классификации обычно берут  $m = \lfloor \sqrt{p} \rfloor$ , а при регрессии -  $m = \lfloor \frac{p}{3} \rfloor$  признаков
- При классификации обычно дерево строится, пока в листе не окажется  $n_{min} = 1$  объект, а при регрессии  $n_{min} = 5$

# OUT-OF-BAG ОШИБКА



$$\text{Err}_{\text{oob}} = \frac{\text{Err}_1 + \dots + \text{Err}_B}{B} = \frac{1}{B} \sum_{b=1}^B \text{Err}_b$$

# OUT-OF-BAG ОШИБКА

- Каждое дерево в случайном лесе обучается по некоторому подмножеству объектов
- Значит, для каждого объекта есть деревья, которые на этом объекте не обучались.

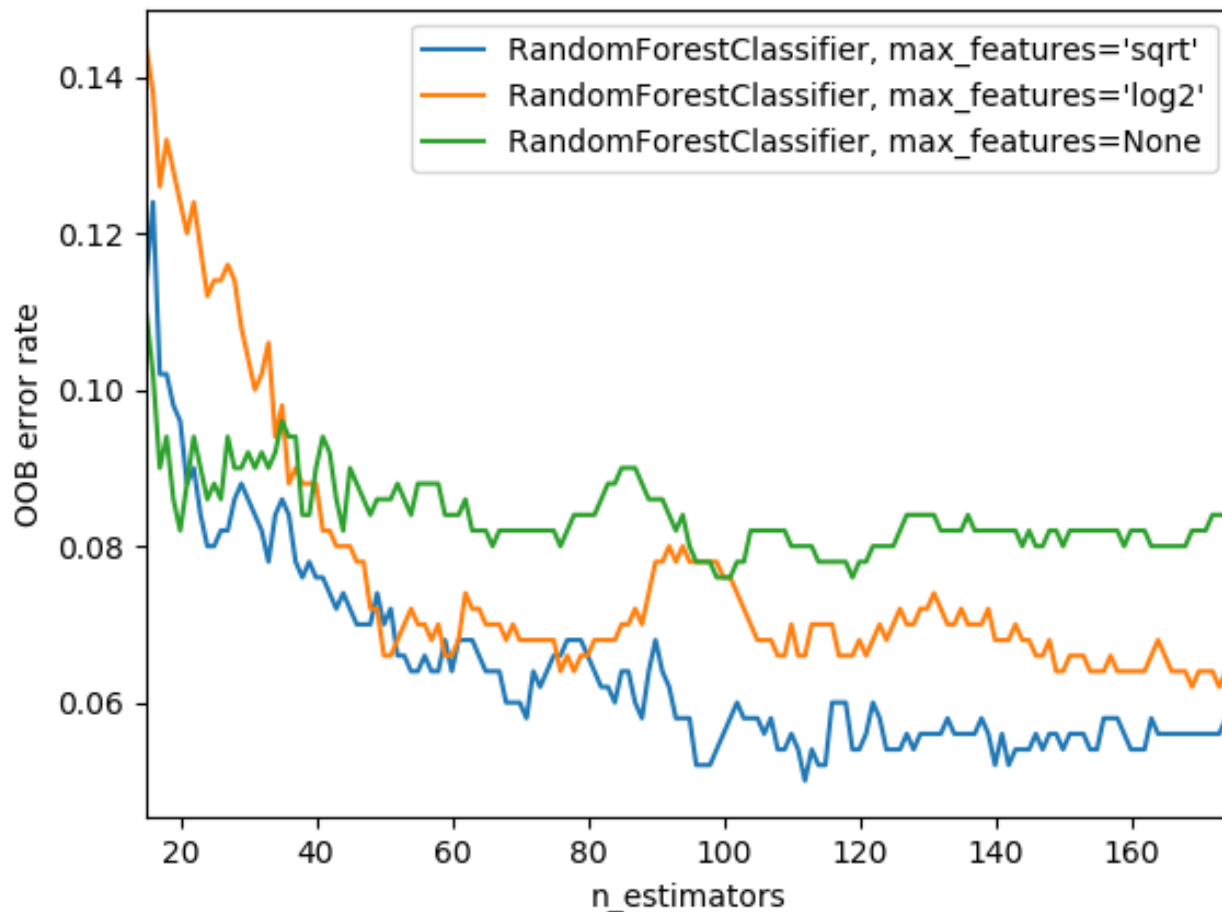
Out-of-bag ошибка:

$$OOB = \sum_{i=1}^l L(y_i, \frac{\sum_{n=1}^N [x_i \notin X_n] b_n(x_i)}{\sum_{n=1}^N [x_i \notin X_n]})$$

**Утверждение.** При  $N \rightarrow \infty$   $OOB$  оценка стремится к *leave-one-out* оценке.

# OOB-SCORE

По графику out-of-bag ошибки можно, например, подбирать количество деревьев в случайном лесе

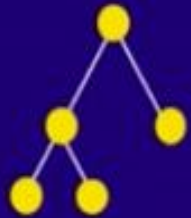


# БУСТИНГ

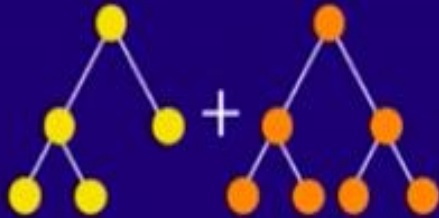
Идея: строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих.

# БУСТИНГ

*Идея: строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих.*



Ошибка



Ошибка



Ошибка

# БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Решаем задачу регрессии с минимизацией квадратичной ошибки:

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

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:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - \mathbf{y}_i)^2$$

- Ошибка на объекте  $x$ :

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



# БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 1: Ищем алгоритм  $b_1(x)$ , минимизирующий ошибку:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - y_i)^2$$

- Ошибка на объекте  $x$ :

$$\mathbf{s} = y - b_1(x)$$

Следующий алгоритм должен настраиваться на эту ошибку, т.е. *целевая переменная для следующего алгоритма – это вектор ошибок  $\mathbf{s}$*  (а не исходный вектор  $y$ )

# БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 1: Ищем алгоритм  $b_1(x)$ , минимизирующий ошибку:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - y_i)^2$$

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

$$b_2(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - \mathbf{s}_i)^2$$

# БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 1: Ищем алгоритм  $b_1(x)$ , минимизирующий ошибку:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - y_i)^2$$

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

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

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:  $\mathbf{s}_i^{(N)} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i)$

Ищем алгоритм  $b_N(x)$ :

$$b_N(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l \left( b(x_i) - \mathbf{s}_i^{(N)} \right)^2$$

# 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).*

# ГРАДИЕНТНЫЙ БУСТИНГ

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

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

# 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

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

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

# 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

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

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



# ГРАДИЕНТНЫЙ БУСТИНГ

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

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

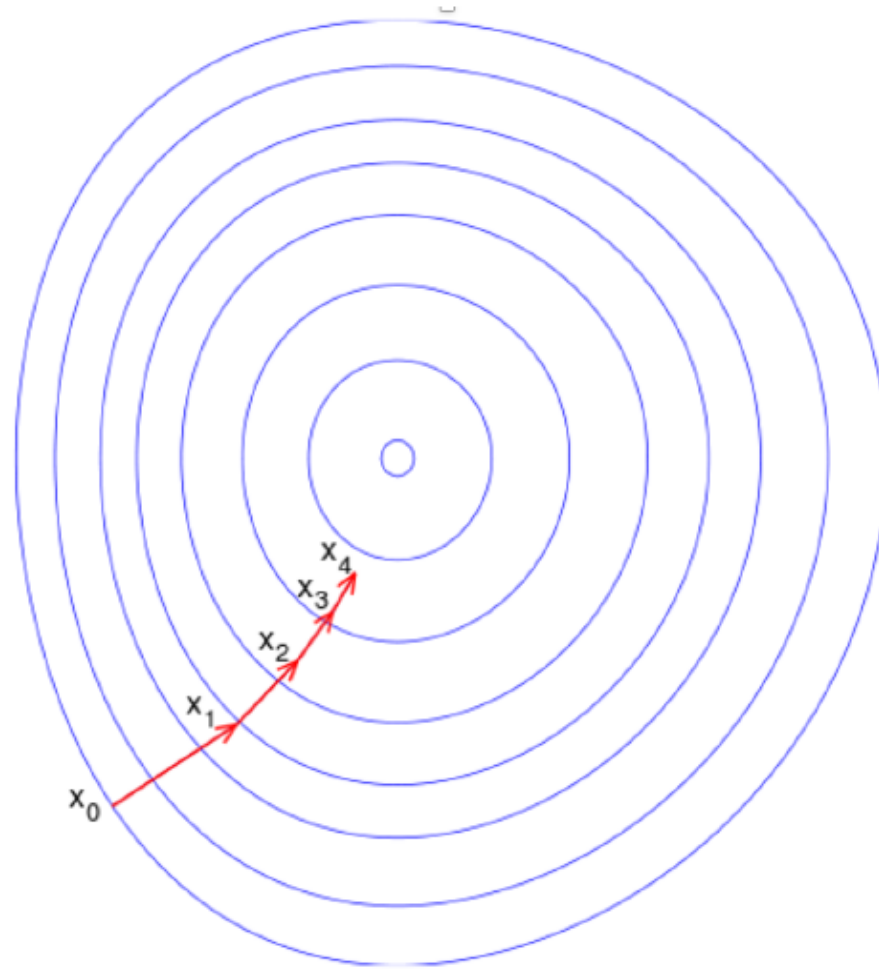
где на  $N$ -м шаге

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

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



# 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  $b_N(x)$ :

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

**Утверждение.** Ошибка на  $N$ -м шаге – это антиградиент функции потерь по ответу модели, вычисленный в точке ответа уже построенной композиции:

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

# 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:

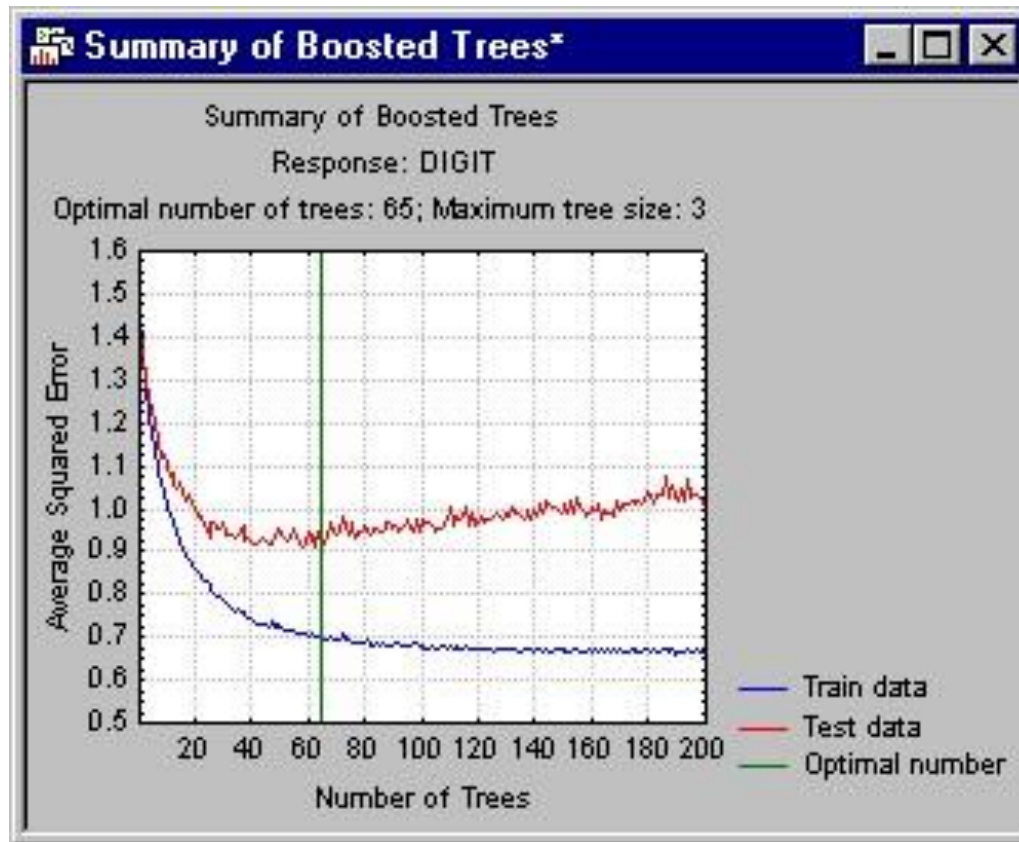
$$a_N(x) = a_{N-1}(x) + \eta \gamma_N b_N(x), \eta \in (0; 1]$$

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  $|X^k| = \frac{1}{2} |X|$ .

# 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).*