Лекция 11 Композиции алгоритмов. Часть 2.

Кантонистова Е.О.

ВШЭ, 2020

ЧАСТЬ 1. БУСТИНГ.

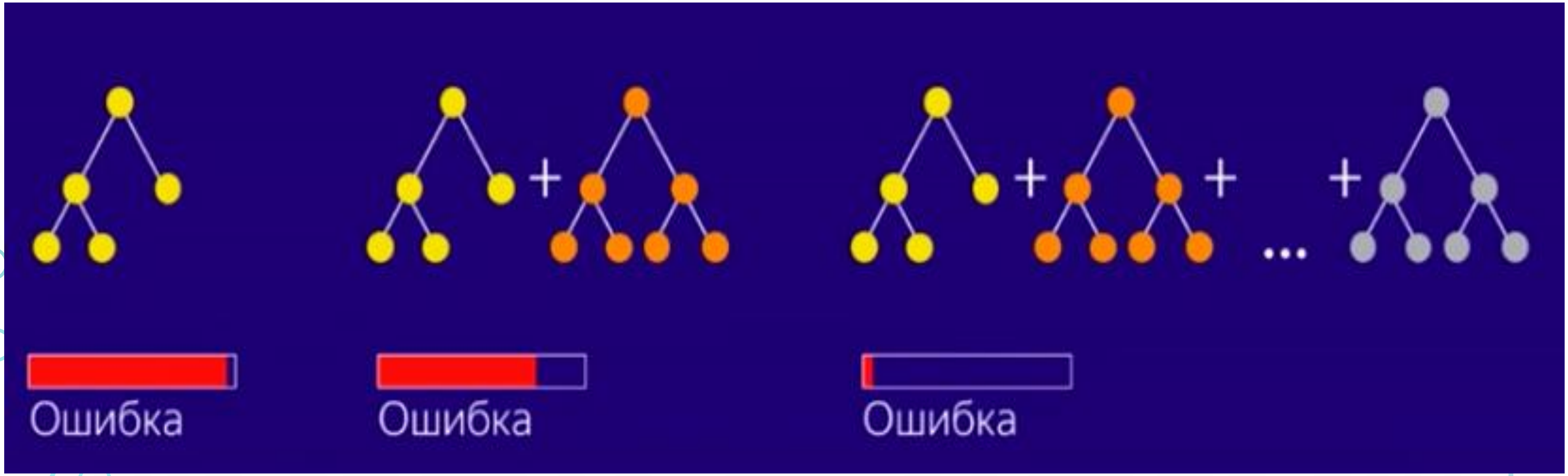
- Бустинг для регрессии с MSE
- Градиентный бустинг

БУСТИНГ

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

БУСТИНГ

<u>Идея</u>: строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих.



Бустинг в задаче регрессии

Решаем задачу регрессии с минимизацией квадратичной ошибки:

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

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:

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

- Ошибка на объекте x:

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

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 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:

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

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

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

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

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

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

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

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

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

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

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

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

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.

<u>Шаг N:</u> Ошибка: $s_i^{(N)} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i)$

Ищем алгоритм $b_N(x)$:

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

Бустинг в задаче регрессии

Каждый следующий алгоритм настраиваем на ошибку предыдущих.

<u>Шаг N:</u> Ошибка: $s_i^{(N)} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i)$

Ищем алгоритм $b_N(x)$:

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

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

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

ГРАДИЕНТНЫЙ БУСТИНГ

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

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

ГРАДИЕНТНЫЙ БУСТИНГ

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

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

где на N-м шаге

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

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

ГРАДИЕНТНЫЙ БУСТИНГ

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

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

где на N-м шаге

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

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

ГРАДИЕНТНЫЙ БУСТИНГ

Пусть $L(y, z)$ – произвольная дифференцируемая функция потерь.

Строим алгоритм $a_N(x)$ вида

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

где на N-м шаге

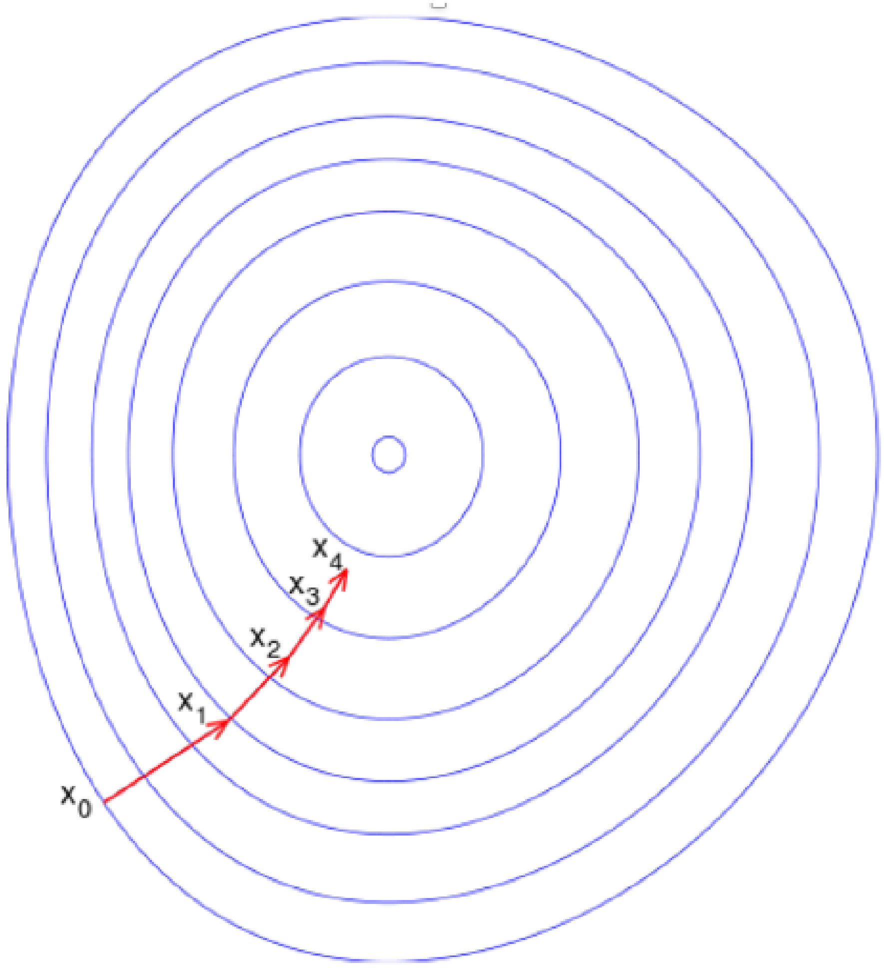
$$\boldsymbol{b_N(x) = \underset{b \in A}{\operatorname{argmin}} \sum_{i=1}^{l} \left(b(x_i) - s_i^{(N)} \right)^2},$$

$$\boldsymbol{s_i^{(N)} = -\frac{\partial L}{\partial z}}$$

Коэффициент γ_N должен минимизировать ошибку:

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

ГРАДИЕНТНЫЙ СПУСК В ПРОСТРАНСТВЕ ФУНКЦИЙ



ВЫБОР БАЗОВЫХ АЛГОРИТМОВ

- *Что произойдет с предсказанием бустинга, если базовые алгоритмы слишком простые?*
- *Что будет, если базовые алгоритмы слишком сложные?*

Сокращение шага (регуляризация)

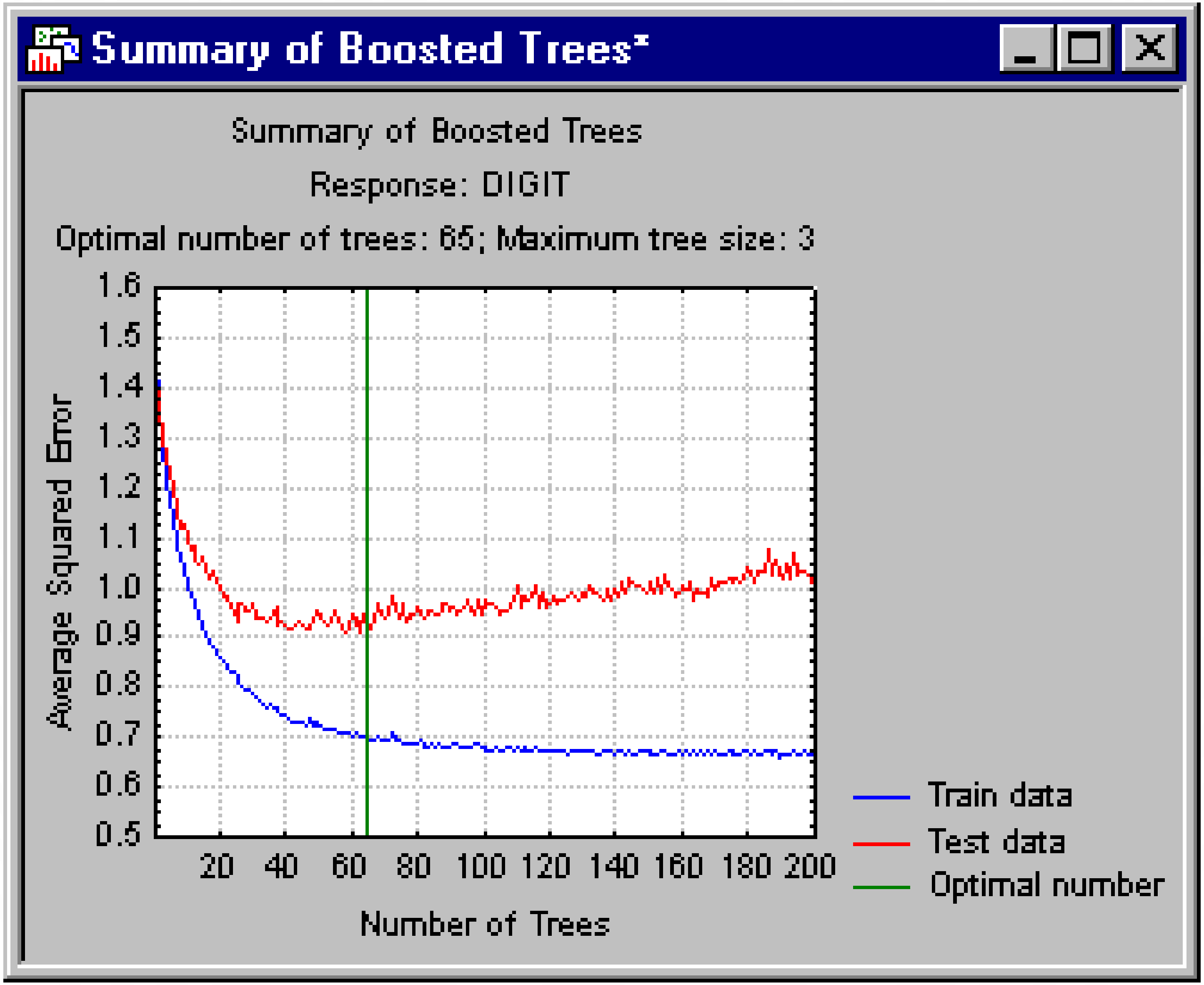
- Если базовые алгоритмы очень простые, то они плохо приближают антиградиент функции потерь, т.е. градиентный бустинг может свестись к случайному блужданию.
- Если базовые алгоритмы сложные, то за несколько шагов бустинг подгонится под обучающую выборку, и получим переобученный алгоритм.

Возможное решение – сокращение шага:

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

Чем меньше темп обучения η, тем меньше степень доверия к каждому базовому алгоритму, и тем лучше качество итоговой композиции.

КОЛИЧЕСТВО ИТЕРАЦИЙ БУСТИНГА



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

Смещение и разброс

Какими будут смещение и разброс у бустинга?

СМЕЩЕНИЕ И РАЗБРОС

- Бустинг целенаправленно уменьшает ошибку, т.е. **смещение у него маленькое**.
- Алгоритм получается сложным, поэтому **разброс большой**.

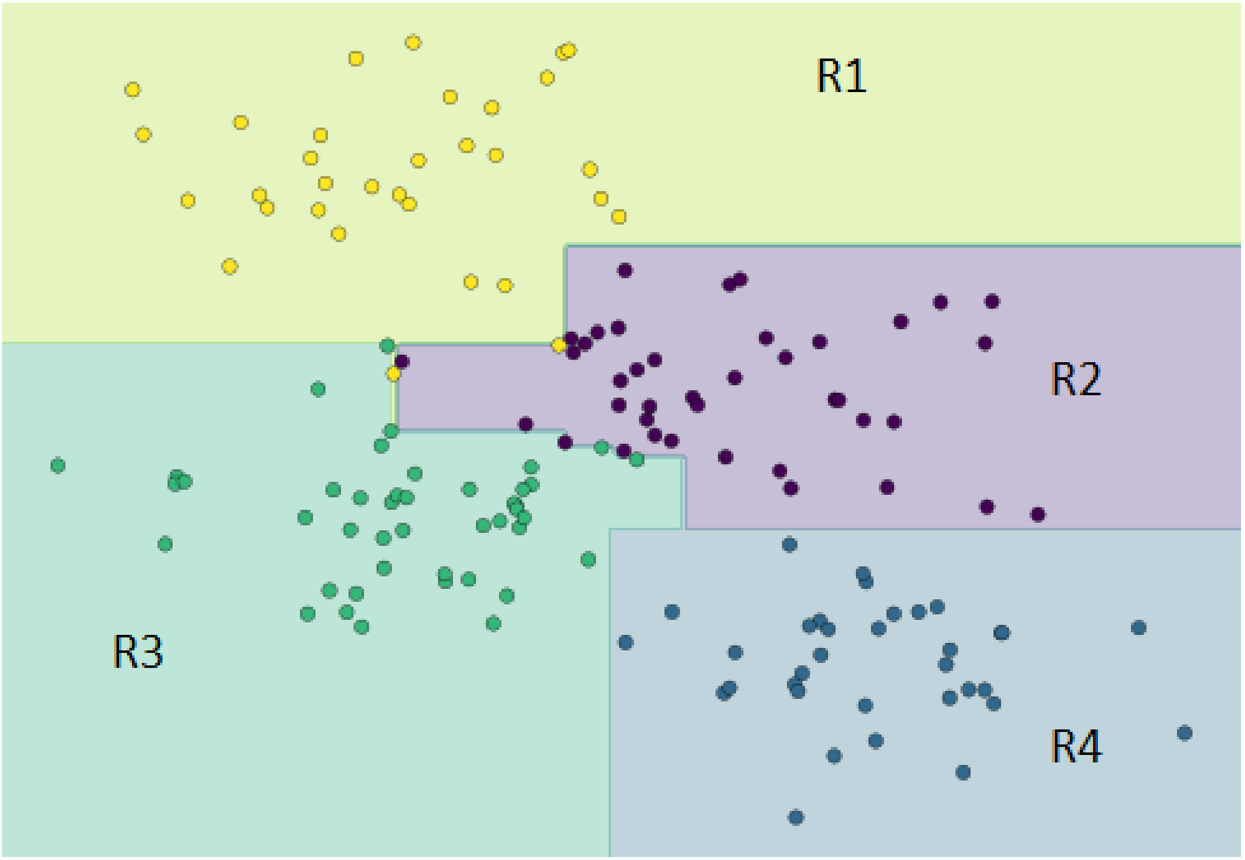
Значит, чтобы не переобучиться, в качестве базовых алгоритмов надо брать неглубокие деревья (глубины 3-6).

ЧАСТЬ 2.
ЧАСТНЫЕ СЛУЧАИ БУСТИНГА

- Бустинг над решающими деревьями
- Бустинг с логистической функцией потерь
- Бустинг с экспоненциальной функцией потерь

ГРАДИЕНТНЫЙ БУСТИНГ НАД ДЕРЕВЬЯМИ

- Решающее дерево разбивает пространство объектов на области, в каждой из который предсказывает некоторый ответ: $$b_n(x) = \sum_{j=1}^{J} b_{nj}[x \in R_j]$$



ГРАДИЕНТНЫЙ БУСТИНГ НАД ДЕРЕВЬЯМИ

- На N-й итерации бустинга:

$$a_N(x) = a_{N-1}(x) + \boldsymbol{\gamma_N} \sum_{\boldsymbol{j=1}}^{\boldsymbol{J_N}} \boldsymbol{b_{Nj}}[\boldsymbol{x} \in \boldsymbol{R_j}],$$

Добавление одного дерева равносильно добавлению J_N предикатов.

ГРАДИЕНТНЫЙ БУСТИНГ НАД ДЕРЕВЬЯМИ

- На N-й итерации бустинга:

$$a_N(x) = a_{N-1}(x) + \gamma_N \sum_{j=1}^{J_N} b_{Nj}[x \in R_j],$$

Добавление одного дерева равносильно добавлению J_N предикатов.

- Улучшим предсказание, подобрав при каждом предикате свой коэффициент:

$$\sum_{i=1}^{l} L\left(y_i, a_{N-1}(x_i) + \sum_{j=1}^{J_N} \boldsymbol{\gamma_{Nj}}[x \in R_j]\right) \to \min_{\{\gamma_{Nj}\}}$$

ГРАДИЕНТНЫЙ БУСТИНГ НАД ДЕРЕВЬЯМИ

$$\sum_{i=1}^{l} L\left(y_i, a_{N-1}(x_i) + \sum_{j=1}^{J_N} \gamma_{Nj} [x \in R_j]\right) \to \min_{\{\gamma_{Nj}\}}$$

- Области R_j не пересекаются, значит, задача разбивается на несколько независимых подзадач:

$$\boldsymbol{\gamma_{Nj} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_j} L(y_i, a_{N-1}(x_i) + \gamma)}$$

ADABOOST

- Рассмотрим экспоненциальную функцию потерь:

$$L(y, z) = \exp(-yz)$$

- Функционал ошибки после $N - 1$ шага:

$$L(a, X) = \sum_{i=1}^{l} \exp(-y_i a_{N-1}(x_i)) = \sum_{i=1}^{l} \exp\left(-y_i \sum_{n=1}^{N-1} \gamma_n b_n(x_i)\right)$$

ADABOOST

- Рассмотрим экспоненциальную функцию потерь:

$$L(y, z) = \exp(-yz)$$

- Функционал ошибки после $N - 1$ шага:

$$L(a, X) = \sum_{i=1}^{l} \exp(-y_i a_{N-1}(x_i))$$

- “Ошибка” после $N - 1$ итерации:

$$s_i = -\frac{\partial L(y_i, z)}{\partial z}\bigg|_{z=a_{N-1}(x_i)} = \boldsymbol{y_i \cdot \exp(-y_i a_{N-1}(x_i))},$$

$$exp(-y_i a_{N-1}(x_i)) \ - \text{"вес" объекта } x_i.$$

ADABOOST

- Рассмотрим экспоненциальную функцию потерь:
$$L(y, z) = \exp(-yz)$$
- $L(a, X) = \sum_{i=1}^{l} \exp(-y_i a_{N-1}(x_i))$
- $s_i = -\frac{\partial L(y_i,z)}{\partial z}\Big|_{z=a_{N-1}(x_i)} = \boldsymbol{y_i \cdot \exp(-y_i a_{N-1}(x_i))}$

На N-м шаге базовый алгоритм ищется по правилу

$$b_N(x) = \underset{b \in A}{argmin} \sum_{i=1}^{l} (b(x_i) - \boldsymbol{s_i})^2$$

ADABOOST: ШТРАФЫ НА ОБЪЕКТАХ

- $s_i = -\frac{\partial L(y_i, z)}{\partial z}\Big|_{z=a_{N-1}(x_i)} = \boldsymbol{y_i \cdot \exp(-y_i a_{N-1}(x_i))}$

$$b_N(x) = \underset{b \in A}{argmin} \sum_{i=1}^{l} (b(x_i) - \boldsymbol{s_i})^2$$

- если все объекты имеют вес 1, то $s_i = y_i$ - алгоритм настраивается на исходные ответы. Тогда штраф на объектах в худшем случае будет $(+1-(-1))^2 = 4$.

ADABOOST: ШТРАФЫ НА ОБЪЕКТАХ

- $s_i = -\frac{\partial L(y_i,z)}{\partial z}|_{z=a_{N-1}(x_i)} = \boldsymbol{y_i \cdot \exp(-y_i a_{N-1}(x_i))}$

$$b_N(x) = \underset{b \in A}{argmin} \sum_{i=1}^{l} (b(x_i) - \boldsymbol{s_i})^2$$

- если все объекты имеют вес 1, то $s_i = y_i$ - алгоритм настраивается на исходные ответы. Тогда штраф на объектах в худшем случае будет $(+1 - (-1))^2 = 4$.
- если отступ $\boldsymbol{y_i a_{N-1}(x_i)}$ на объекте большой положительный, то $s_i \approx 0$, значит, штраф за предсказание $(\pm 1 - 0)^2 = 1$.

ADABOOST: ШТРАФЫ НА ОБЪЕКТАХ

- $s_i = -\frac{\partial L(y_i, z)}{\partial z}\Big|_{z=a_{N-1}(x_i)} = \boldsymbol{y_i \cdot \exp(-y_i a_{N-1}(x_i))}$

$$b_N(x) = \underset{b \in A}{argmin} \sum_{i=1}^{l} (b(x_i) - \boldsymbol{s_i})^2$$

- если все объекты имеют вес 1, то $s_i = y_i$ - алгоритм настраивается на исходные ответы. Тогда штраф на объектах в худшем случае будет $(+1 - (-1))^2 = 4$.
- если отступ $\boldsymbol{y_i a_{N-1}(x_i)}$ на объекте большой положительный, то $s_i \approx 0$, значит, штраф за предсказание $(\pm 1 - 0)^2 = 1$.
- если объект имеет большой отрицательный вес, то следующий базовый алгоритм очень сильно настраивается на этот объект.

AdaBoost: штрафы на объектах

- $s_i = -\frac{\partial L(y_i, z)}{\partial z}\Big|_{z=a_{N-1}(x_i)} = \boldsymbol{y_i \cdot \exp(-y_i a_{N-1}(x_i))}$

$$b_N(x) = \underset{b \in A}{argmin} \sum_{i=1}^{l} (b(x_i) - \boldsymbol{s_i})^2$$

- если все объекты имеют вес 1, то $s_i = y_i$ - алгоритм настраивается на исходные ответы. Тогда штраф на объектах в худшем случае будет $(+1-(-1))^2 = 4$.
- если отступ $\boldsymbol{y_i a_{N-1}(x_i)}$ на объекте большой положительный, то $s_i \approx 0$, значит, штраф за предсказание $(\pm 1 - 0)^2 = 1$.
- если объект имеет большой отрицательный вес, то следующий базовый алгоритм очень сильно настраивается на этот объект.

То есть AdaBoost настраивается на шумовые объекты.

БУСТИНГ С ЛОГИСТИЧЕСКОЙ ФУНКЦИЕЙ ПОТЕРЬ

Логистическая функция потерь:

$$L(y, z) = \log(1 + \exp(-yz))$$

$$-\frac{\partial L}{\partial z}(x_i) = -\frac{\partial \log(1 + \exp(-yz))}{\partial z} = \frac{y_i}{1 + \exp\big(y_i a_{N-1}(x_i)\big)} \Rightarrow$$

$$\boldsymbol{b_N = \underset{b \in A}{\mathbf{argmin}} \sum_{i=1}^{l} \left(b(x_i) - \frac{y_i}{1 + \exp\big(y_i a_{N-1}(x_i)\big)} \right)^2}$$

БУСТИНГ С ЛОГИСТИЧЕСКОЙ ФУНКЦИЕЙ ПОТЕРЬ

Логистическая функция потерь:

$$L(y, z) = \log(1 + \exp(-yz))$$

- Ошибка на N-й итерации:

$$Q(a_N) = \sum_{i=1}^{l} \log\left[1 + \exp\left(-y_i a_N(x_i)\right)\right] =$$

$$= \sum_{i=1}^{l} \log\left[1 + \exp\left(-\boldsymbol{y_i a_{N-1}(x_i)}\right) \cdot \exp(-y_i \gamma_N b_N(x_i))\right]$$

Бустинг с логистической функцией потерь

Логистическая функция потерь:

$$L(y, z) = \log(1 + \exp(-yz))$$

- Ошибка на N-й итерации:

$$Q(a_N) = \sum_{i=1}^{l} \log(1 + \exp(-y_i a_N(x_i)) =$$

$$= \sum_{i=1}^{l} \log[1 + \exp(-\boldsymbol{y_i a_{N-1}(x_i)}) \cdot \exp(-y_i \gamma_N b_N(x_i))]$$

- Если отступ $\boldsymbol{y_i a_{N-1}(x_i)}$ на объекте x_i большой положительный, то $exp(-y_i a_{N-1}(x_i)) \approx 0$, т.е. объект не вносит вклад в ошибку, и можно его исключить на данной итерации.

БУСТИНГ С ЛОГИСТИЧЕСКОЙ ФУНКЦИЕЙ ПОТЕРЬ (ВЛИЯНИЕ ШУМА)

$$\mathbf{s_i} = \frac{\boldsymbol{y_i}}{\mathbf{1} + \boldsymbol{exp}\left(\boldsymbol{y_i a_{N-1}}(\boldsymbol{x_i})\right)} = \boldsymbol{y_i} \cdot \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{exp}\left(\boldsymbol{y_i a_{N-1}}(\boldsymbol{x_i})\right)}$$

Бустинг с логистической функцией потерь: штрафы

$$\mathbf{s_i} = \frac{\boldsymbol{y_i}}{\mathbf{1} + \boldsymbol{exp}\big(\boldsymbol{y_i a_{N-1}}(\boldsymbol{x_i})\big)} = \boldsymbol{y_i} \cdot \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{exp}\big(\boldsymbol{y_i a_{N-1}}(\boldsymbol{x_i})\big)}$$

- все веса не больше 1
- если отступ большой отрицательный (шумовой объект), то вес ≈ 1.
- если отступ примерно 0, то вес $\approx \frac{1}{2}$.

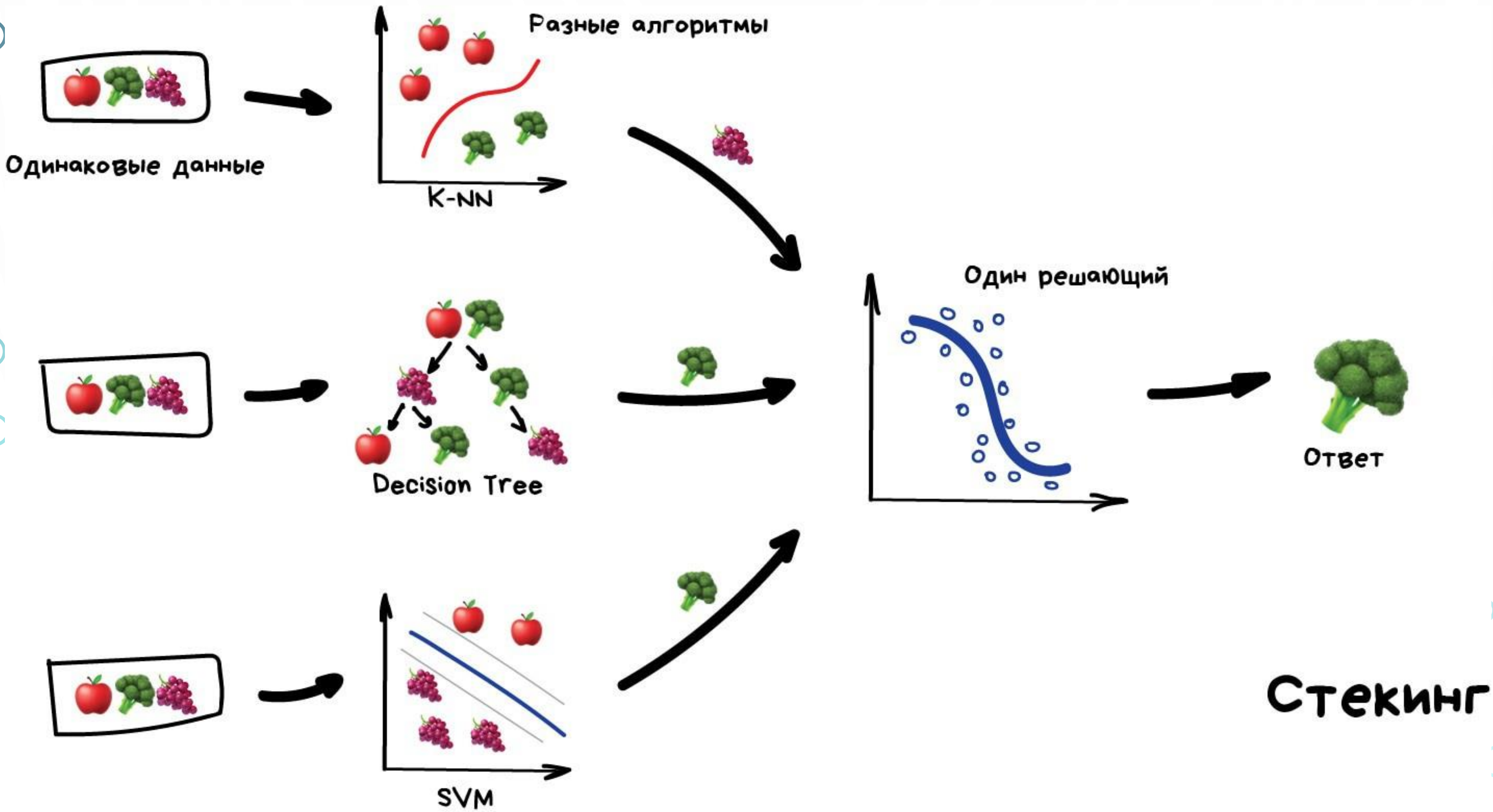
Алгоритм гораздо более устойчив к шумам, чем AdaBoost.

ЧАСТЬ 3. ДРУГИЕ СПОСОБЫ ПОСТРОЕНИЯ КОМПОЗИЦИЙ

- Стекинг (Stacking)
- Блендинг (Blending)

СТЕКИНГ (STACKING)

<u>Идея</u>: обучаем несколько разных алгоритмов и передаём их результаты на вход последнему, который принимает итоговое решение.



Стекинг (stacking)

- Пусть мы обучили N базовых алгоритмов $b_1(x), b_2(x), \dots, b_N(x)$ на выборке X.
- Обучим теперь мета-алгоритм $a(x)$ на прогнозах этих алгоритмов (т.е. прогнозы алгоритмов – это по сути новые признаки):

$$\sum_{i=1}^{l} L(y_i, \boldsymbol{a}(b_1(x_i), b_2(x_i), \dots, b_N(x_i))) \to \min_a$$

Стекинг (Stacking)

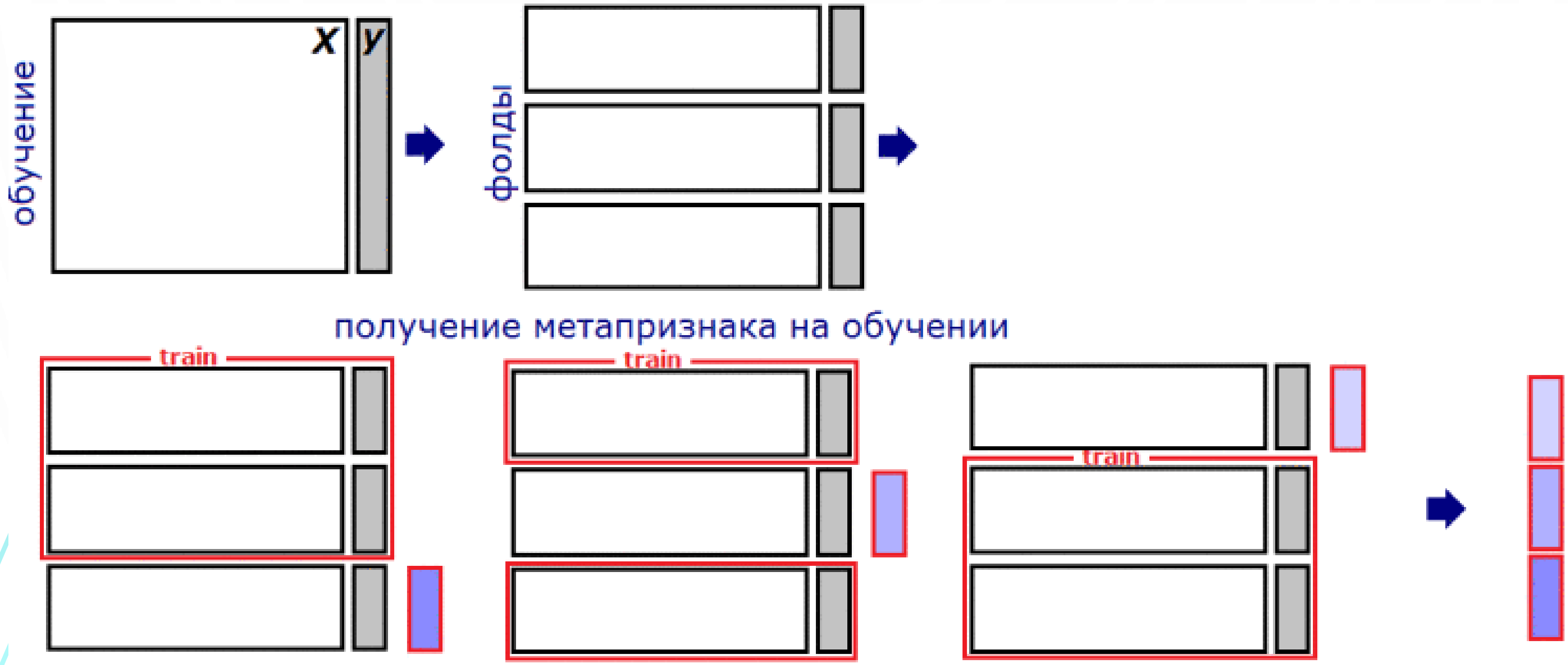
- Пусть мы обучили N базовых алгоритмов $b_1(x), b_2(x), \dots, b_N(x)$ на выборке X.
- Обучим теперь мета-алгоритм $a(x)$ на прогнозах этих алгоритмов (т.е. прогнозы алгоритмов – это по сути новые признаки):

$$\sum_{i=1}^{l} L(y_i, \boldsymbol{a}(b_1(x_i), b_2(x_i), \dots, b_N(x_i))) \to \min_a$$

Алгоритм $a(x)$ будет больше опираться на предсказание тех алгоритмов, которые сильнее подстроились под обучающую выборку $\Rightarrow$ будет переобучен.

СТЕКИНГ (STACKING)

<u>Решение: </u>будем обучать базовые алгоритмы и мета-алгоритм на разных выборках.



СТЕКИНГ (STACKING)

Решение: будем обучать базовые алгоритмы и мета-алгоритм на разных выборках.

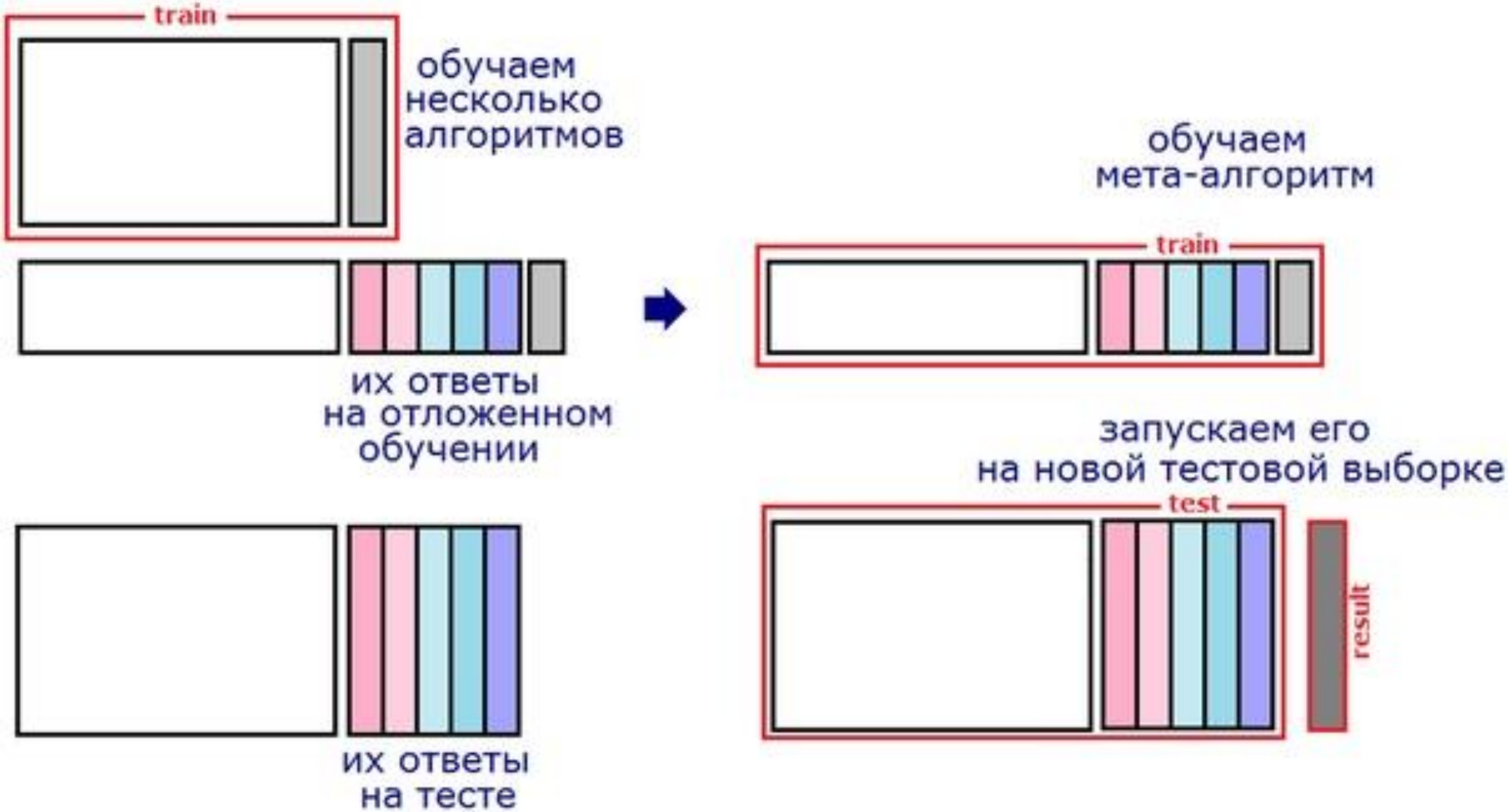
- Разобъем выборку на K частей: $X_1, X_2, \ldots, X_K$.
- Пусть $b_j^{-k}(x)$ - j-й алгоритм, обученный на всех блоках, кроме k-го.

Для обучения мета-алгоритма будем минимизировать функционал:

$$\sum_{k=1}^{K} \sum_{(x_i,y_i)\in X_k} L\left(y_i, a\left(b_1^{-k}(x_i), b_2^{-k}(x_i), \ldots, b_N^{-k}(x_i)\right)\right) \to \min_a$$

Теперь алгоритм а обучается на объектах, на которых не обучались базовые алгоритмы $\Rightarrow$ нет переобучения.

ИСПОЛЬЗОВАНИЕ МЕТАПРИЗНАКОВ ВМЕСТЕ С ПРИЗНАКАМИ



[Статья про stacking и blendind из блога А.Дьяконова]()

БЛЕНДИНГ (BLENDING)

Блендинг – это частный случай стекинга, в котором мета-алгоритм линеен:

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