



УНИВЕРСИТЕТ ИТМО

# Лекция 5. Ансамблирование моделей

Олег А. Евстафьев<sup>1</sup> Михаил А. Каканов<sup>1</sup>

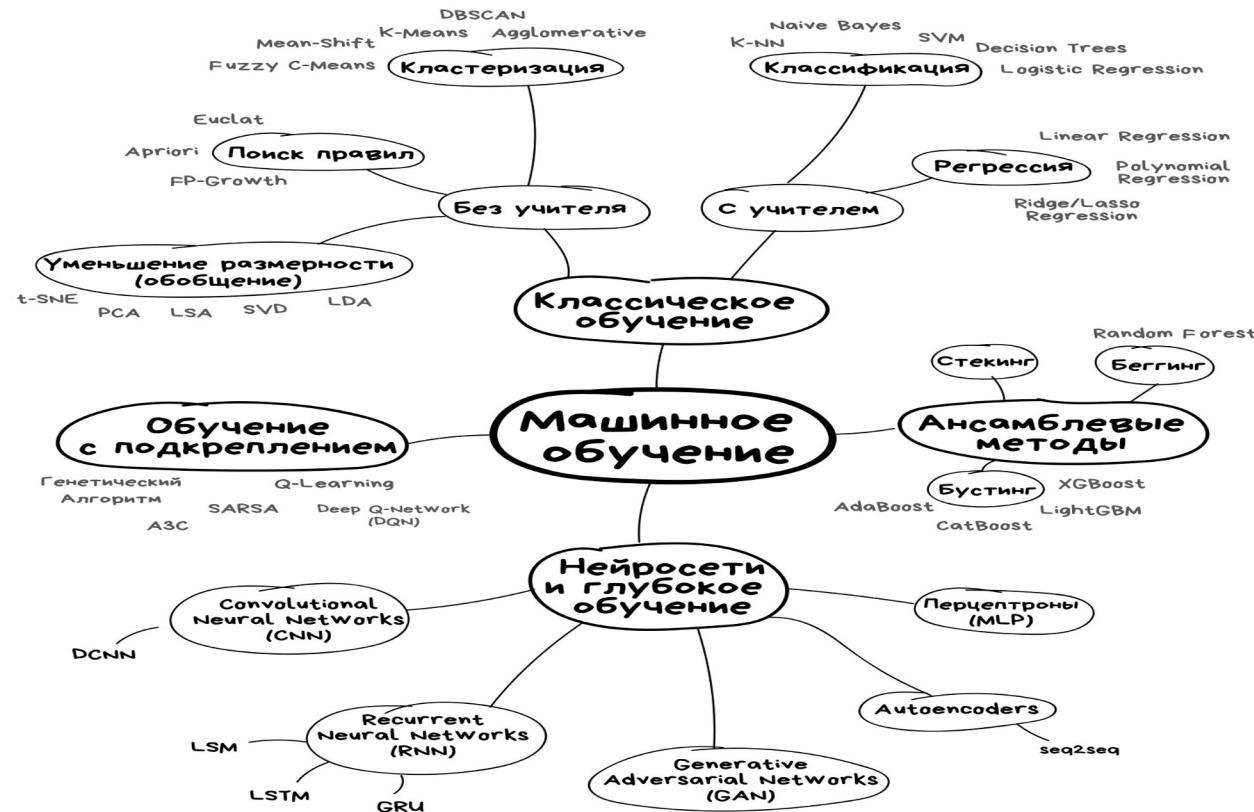
<sup>1</sup> Факультет систем управления и робототехники, Университет ИТМО  
[{oaevstafev, makakanov}@itmo.ru](mailto:{oaevstafev, makakanov}@itmo.ru)

# Сегодня вы узнаете



- Ансамблирование моделей
- Основные понятия. Простое голосование (комитет большинства). Классификация видов нейронных сетей
- Взвешенное голосование
- Линейные ансамбли
- Бутстрэп, Бэггинг, бустинг, градиентный бустинг
- Алгоритм Random Forest
- Алгоритм AdaBoost

## МАШИННОЕ ОБУЧЕНИЕ



## Основные виды машинного обучения



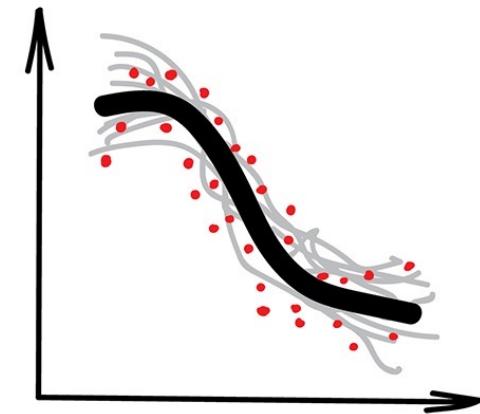
# Ансамбли

**Ансамблевые методы** — это парадигма машинного обучения, где несколько моделей обучаются для решения одной и той же проблемы и объединяются для получения лучших результатов. Основная гипотеза состоит в том, что при правильном сочетании слабых моделей мы можем получить более точные и/или надежные модели.

**Ансамбль методов** в статистике и обучении машин использует несколько обучающих алгоритмов с целью получения лучшей эффективности прогнозирования, чем могли бы получить от каждого обучающего алгоритма по отдельности. В отличие от статистического ансамбля в статистической механике, который обычно бесконечен, ансамбль методов в обучении машин состоит из конкретного конечного множества альтернативных моделей, но, обычно, позволяет существовать существенно более гибким структурам.

## Используются:

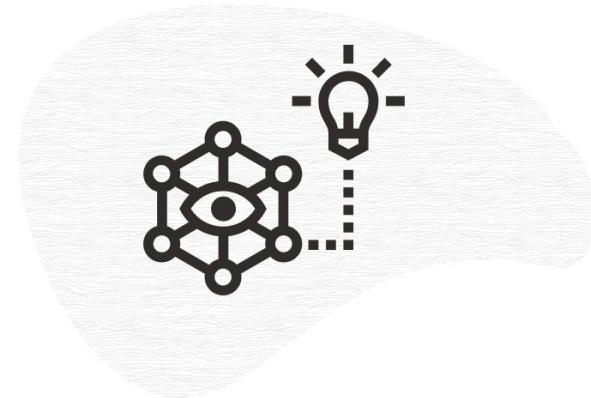
- Используются как и классические алгоритмы (но работают точнее)
- Поисковые системы
- Компьютерное зрение
- Распознавание объектов



Ensemble Methods

# Применение ансамблей

- Дистанционное зондирование Земли
  - Отражение растительного покрова
  - Обнаружение изменений
- Защита компьютера
  - DoS-атака
  - Обнаружение вредоносных программ
  - Обнаружение вторжения
- Распознавание лиц
- Распознавание эмоций
- Выявление мошенничества
- Принятие финансовых решений
- Медицина



# Ансамблирование моделей

## Идея алгоритма

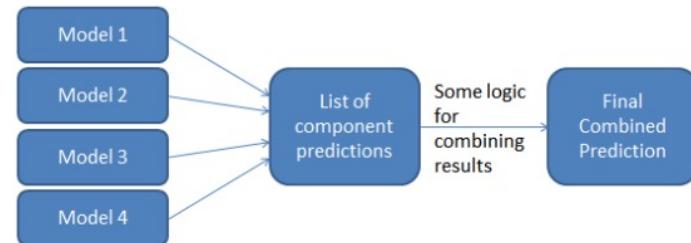
Обучаем несколько базовых моделей, а затем агрегируем их результаты по какому-либо правилу и выдаем окончательный результат.

Зачем это нужно:

- В совокупности получаем более сложную модель, чем каждая в отдельности
- Уменьшение разбора
- Избежание переобучения/недообучения
- Возможность работы с признаками разной природы (использовать разные алгоритмы)

Для простоты рассмотрим задачу бинарной классификации. Пусть всего  $N$  базовых моделей и каждая предсказывает класс  $c_1$  или  $c_2$ . Тогда агрегированный алгоритм может выдавать класс  $c_1$  по следующим правилам:

- AND-правило: если все базовые модели выдали  $c_1$
- OR-правило: если хотя бы одна базовая модель выдала  $c_1$
- $k$ -out-of- $N$ : если хотя бы  $k$  базовых моделей из  $N$  выдали  $c_1$
- majority vote: если большинство базовых моделей выдало  $c_1$



## Обобщение с весами

Также, если используются правила  $k$ -out-of- $N$  или majority vote, можно каждой базовой модели присвоить вес, основываясь на качестве предсказания на валидационной выборке.

# Ансамблирование моделей

## Предсказание класса по уровням ранжирования

Пусть теперь рассматривается задачи многоклассовой классификации с  $C$  классами. Пусть каждая  $k$ -ая базовая модель выдает некую отранжированную информацию о классе объекта:

$$c_{k_1} \succ c_{k_2} \succ \cdots \succ c_{k_C}$$

Это означает, что класс  $c_{k_1}$  наиболее вероятен для рассматриваемого объекта, а класс  $c_{k_C}$  --- наименее вероятен.

Пусть  $B_k(i)$  --- сколько классов было отранжировано ниже  $i$ -го класса  $k$ -ой базовой моделью. Чем  $B_k(i)$  выше, тем более вероятен  $i$ -ый класс. Поэтому, в качестве совокупного рейтинга построим следующую величину:

$$g_i(x) = \sum_k B_k(i, x)$$

Тогда результирующее предсказание на объекте  $x$ :

$$\hat{y}(x) = \operatorname{argmax}_{i \in [1, \dots, C]} g_i(x)$$

# Ансамблирование моделей

## Предсказание класса по вероятностям

Опять рассмотрим задачу многоклассовой классификации с  $C$  классами. Пусть каждая  $k$ -ая базовая модель выдает вектор вероятностей из принадлежностей к каждому классу:

$$[p^k(c_1), p^k(c_2), \dots, p^k(c_C)]$$

Тогда  $\hat{y}(x) = c_i$ , где  $i = \operatorname{argmax}_{i \in [1, \dots, C]} F(p^1(c_i), p^2(c_i), \dots, p^N(c_i))$

$F$  --- среднее арифметическое или медиана.

# Определение ансамбля

$X^\ell = (x_i, y_i)_{i=1}^\ell \subset X \times Y$  — обучающая выборка,  $y_i = y^*(x_i)$

$a_t: X \rightarrow Y$ ,  $t = 1, \dots, T$  — обучаемые базовые алгоритмы

**Идея ансамбля:** возможно ли из множества плохих алгоритмов  $a_t$  построить один хороший?

**Декомпозиция** базовых алгоритмов  $a_t(x) = C(b_t(x))$

$a_t: X \xrightarrow{b_t} R \xrightarrow{C} Y$ , где  $R$  — более удобное пространство оценок,  $C$  — решающее правило, как правило, весьма простого вида

**Ансамбль** базовых алгоритмов  $b_1, \dots, b_T$ :

$$a(x) = C(F(b_1(x), \dots, b_T(x), x)),$$

$F: R^T \times X \rightarrow R$  — агрегирующая функция или мета-алгоритм

# Определение ансамбля

- **Пример 1:** классификация,  $Y$  — конечное множество,  $R = Y$ ,  $C(b) \equiv b$  — решающее правило не используется.
- **Пример 2:** классификация на 2 класса,  $Y = \{-1, +1\}$ ,

$$a(x) = \text{sign}(b(x)),$$

где  $R = \mathbb{R}$ ,  $b: X \rightarrow \mathbb{R}$ ,  $C(b) \equiv \text{sign}(b)$ .

- **Пример 3:** классификация на  $M$  классов  $Y = \{1, \dots, M\}$ ,

$$a(x) = \arg \max_{y \in Y} b_y(x),$$

где  $R = \mathbb{R}^M$ ,  $b: X \rightarrow \mathbb{R}^M$ ,  $C(b_1, \dots, b_M) \equiv \arg \max_{y \in Y} b_y$ .

- **Пример 4:** регрессия,  $Y = R = \mathbb{R}$ ,  
 $C(b) \equiv b$  — решающее правило не нужно.

# Корректирующие функции

Общие требования к агрегирующей функции:

- $F(b_1, \dots, b_T, x) \in [\min_t b_t, \max_t b_t]$  — среднее по Коши  $\forall x$
- $F(b_1, \dots, b_T, x)$  монотонно не убывает по всем  $b_t$

Примеры агрегирующих функций:

- простое голосование (simple voting):

$$F(b_1, \dots, b_T) = \frac{1}{T} \sum_{t=1}^T b_t$$

- взвешенное голосование (weighted voting):

$$F(b_1, \dots, b_T) = \sum_{t=1}^T \alpha_t b_t, \quad \sum_{t=1}^T \alpha_t = 1, \quad \alpha_t \geq 0$$

- смесь алгоритмов (mixture of experts)

с функциями компетентности (gating function)  $g_t: X \rightarrow \mathbb{R}$

$$F(b_1, \dots, b_T, x) = \sum_{t=1}^T g_t(x) b_t(x)$$

# Ансамбли моделей



**Стекинг.** Могут рассматриваться разнородные отдельно взятые модели. Существует мета-модель, которой на вход подаются базовые модели, а выходом является итоговый прогноз.

**Бэггинг.** Рассматриваются однородные модели, которые обучаются независимо и параллельно, а затем их результаты просто усредняются. Ярким представителем данного метода является случайный лес.

**Бустинг.** Рассматриваются однородные модели, которые обучаются последовательно, причем последующая модель должна исправлять ошибки предыдущей. Конечно, в качестве примера здесь сразу приходит на ум градиентный бустинг.

Стекинг (Stacked Generalization или Stacking) — один из самых популярных способов ансамблирования алгоритмов, т.е. использования нескольких алгоритмов для решения одной задачи машинного обучения. Это вполне естественно, его идея лежит на поверхности. Известно, что если обучить несколько разных алгоритмов, то в задаче регрессии их среднее, а в задаче классификации — голосование по большинству, часто превосходят по качеству все эти алгоритмы.

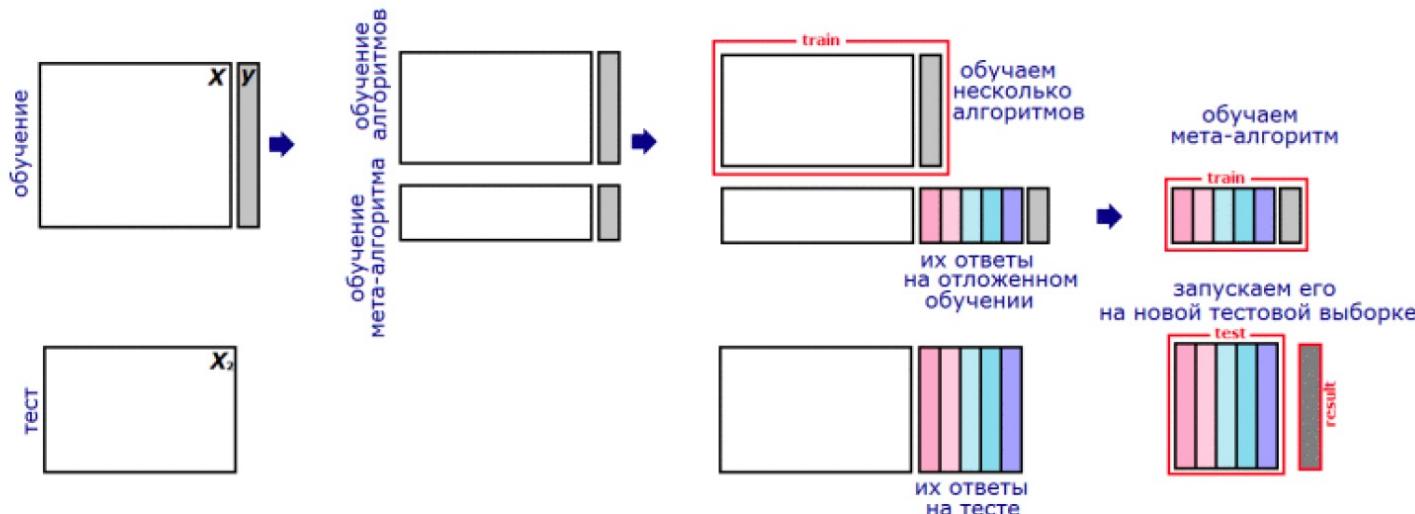
## Алгоритм обучения:

- Делим выборку на  $k$  фолдов (тот же смысл, что и в кросс-валидации).
- Для объекта из выборки, который находится в  $k$ -ом фолде, делается предсказание слабыми алгоритмами, которые были обучены на  $k-1$  фолдах. Этот процесс итеративен и происходит для каждого фолда.
- Создается набор прогнозов слабых алгоритмов для каждого объекта выборки.
- На сформированных низкоуровневыми алгоритмами прогнозах в итоге обучается метамодель.

# Блендинг (Blending) – смещивание базовых алгоритмов

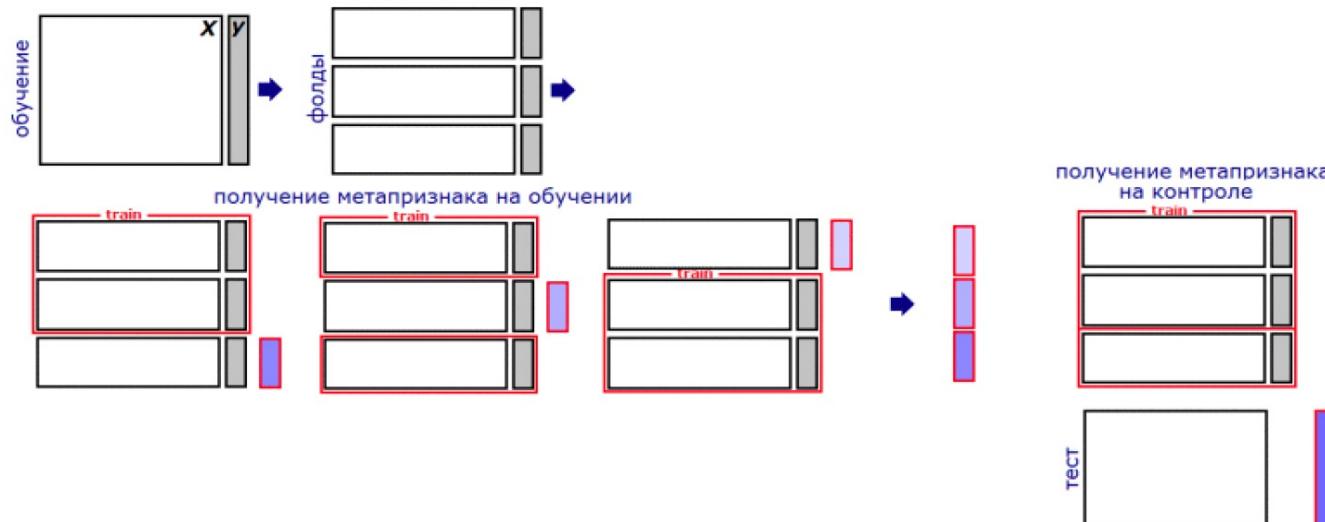
**Идея:** базовые алгоритмы  $b_t(x)$  как (мета)признаки подаём на вход любому ML алгоритму, не обязательно линейному.

**Проблема:** этот (мета)алгоритм нельзя обучать на тех же данных, что и базовые  $b_t(x)$ , будет переобучение!



**Новая проблема:** для обучения используется не вся выборка.

Решение проблемы: разбиение выборки на  $k$  блоков ( $k$ -fold)



Новая проблема: вместо одного метапризнака  $b_t(x)$  имеем  $k$  похожих, но разных  $b_{tj}(x)$ ,  $j = 1, \dots, k$ .

Вариант решения: усреднение метапризнаков  $b_t(x) = \frac{1}{k} \sum_{j=1}^k b_{tj}(x)$

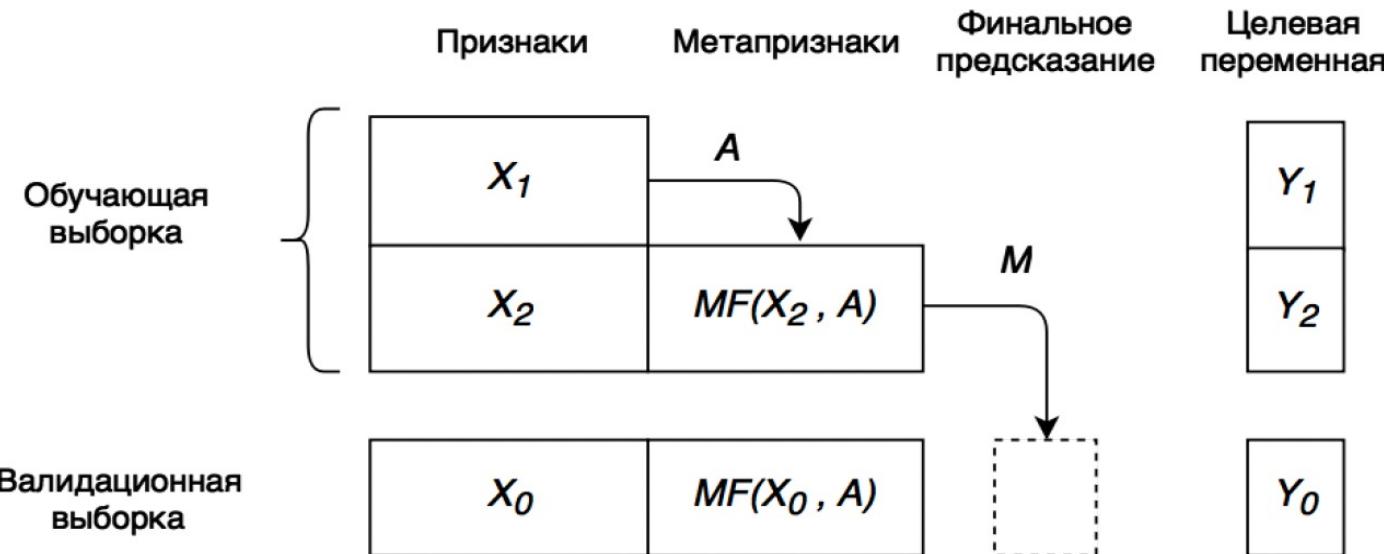
# Стекинг по схеме K-fold



**Преимущества:** использование всей обучающей выборки

**Недостатки:** переобучение, "неоднородность" метапризнаков

# Стекинг по схеме hold-out



**Преимущества:** отсутствие переобучения

**Недостатки:** неэффективное использование обучающей выборки

# Стекинг моделей

Рассмотрим задачу регрессии. Пусть всего  $K$  базовых моделей каждая модель --- это  $f_k(x)$  алгоритмов регрессии. Результирующую модель строим следующим образом:

$$f(x) = \sum_{k=1}^K w_k f_k(x)$$

Можно находить веса следующим образом:

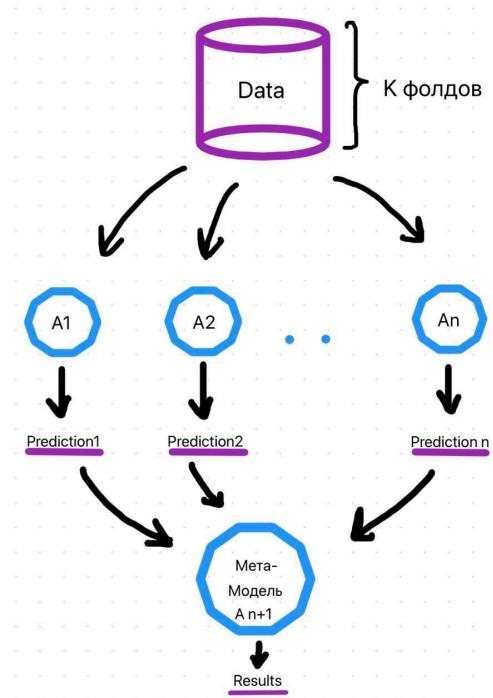
$$\hat{w} = \operatorname{argmin}_w \sum_{i=1}^N \mathcal{L}(y_i, \sum_{k=1}^K w_k f_k(x_i))$$

Но такой способ приведет к переобучению. Поэтому будем находить веса при помощи кросс-валидации, а именно: разобьем выборку на  $M$  частей. Пусть  $fold(i)$  --- та часть, которая содержит  $i$ -ый объект, а  $f_k^{fold(i)}$  --- алгоритм, обученный на всех фолдах, кроме  $fold(i)$ . Тогда:

$$\hat{w} = \operatorname{argmin}_w \sum_{i=1}^N \mathcal{L}(y_i, \sum_{k=1}^K w_k f_k^{fold(i)}(x_i))$$

Для уменьшения переобучения можно добавить условия на неотрицательность весов или добавить к

$$\text{функционалу регуляризатор } \lambda \sum_{k=1}^K (w_k - \frac{1}{K})^2$$



# Обобщенный Стекинг

Предполагаем, что

$$f(x) = A_\theta(f_1(x), \dots, f_K(x))$$

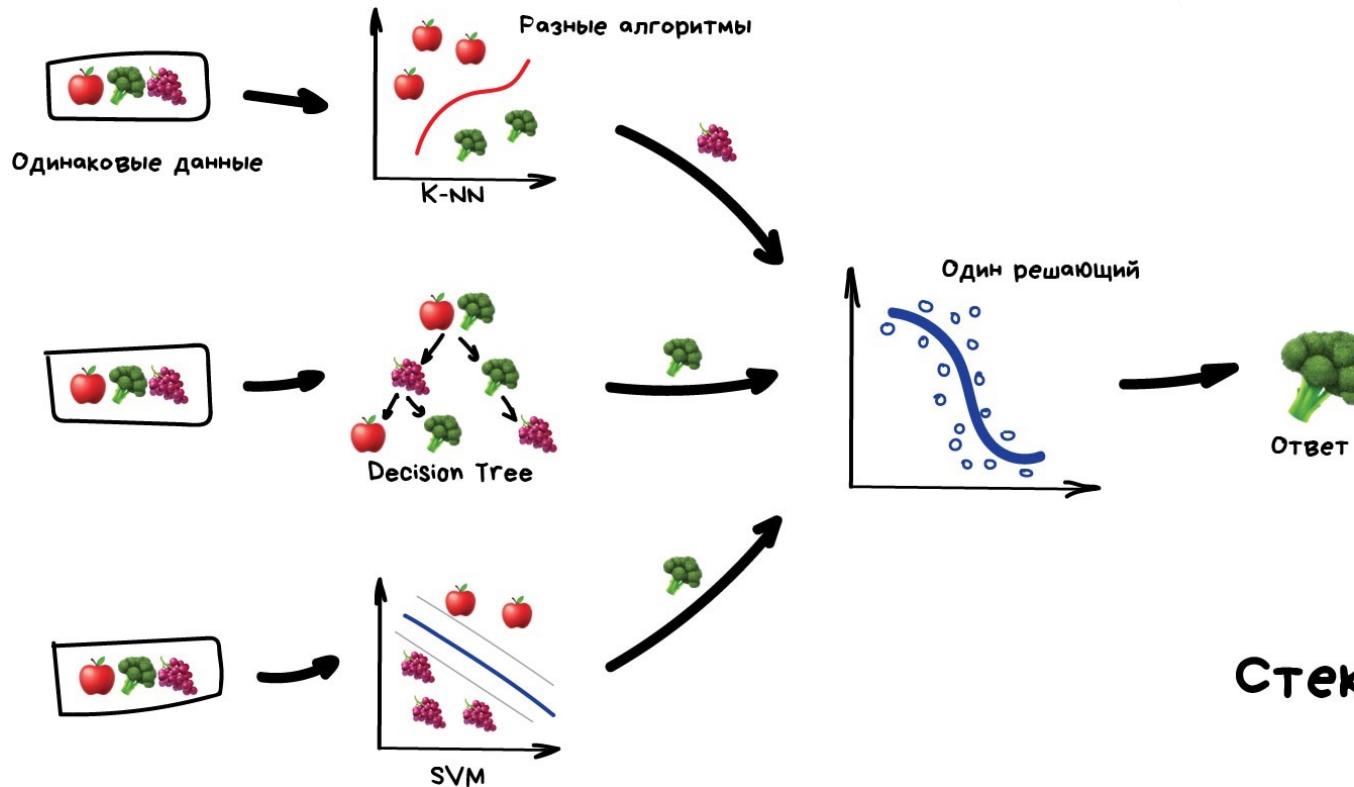
, где  $\theta$  --- вектор параметров:

$$\hat{\theta} = \operatorname{argmin}_\theta \sum_{i=1}^N \mathcal{L}(y_i, A_\theta(f_1^{-fold(i)}(x), \dots, f_K^{-fold(i)}(x)))$$

$f_i(x)$ :

- Номер класса
- Вектор вероятностей классов
- Любой изначальный или сгенерированный признак

# Стекинг

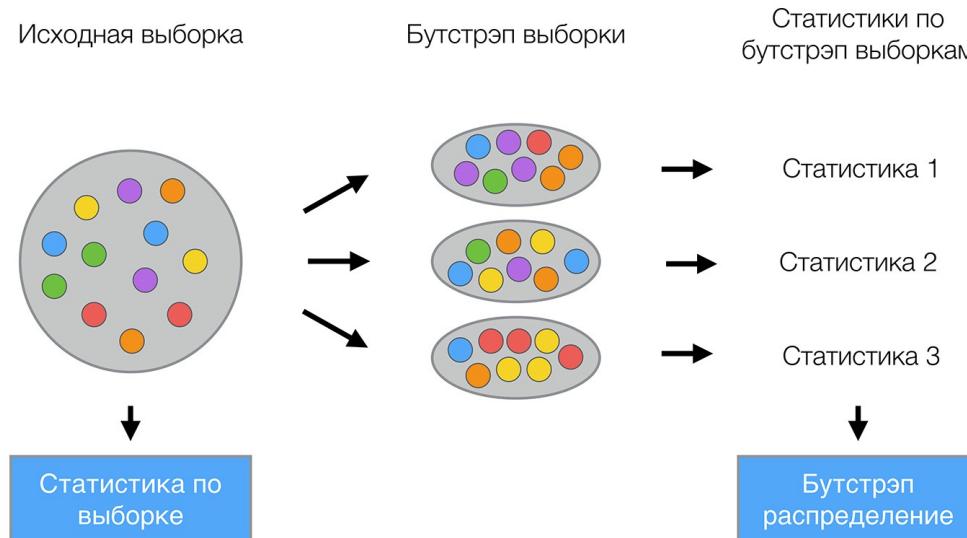


Стекинг

- Бэггинг (от англ. "bagging") не имеет ничего общего с мешками - "bags". Слово является сокращением от *bootstrap aggregation*. Это технология классификации, использующая ансамбли моделей, каждая из которых обучается независимо. Результат классификации определяется путем голосования.
- Бэггинг позволяет снизить процент ошибки классификации в случае, когда высока дисперсия ошибки базового метода. Эффективность бэггинга достигается благодаря тому, что базовые алгоритмы, обученные по различным подвыборкам, получаются достаточно различными, и их ошибки взаимно компенсируются при голосовании, а также за счёт того, что объекты-выбросы могут не попадать в некоторые обучающие подвыборки.
- Пример бэггинга – случайный лес

# Бутстррап

- Позволяет просто и быстро оценивать самые разные статистики (доверительные интервалы, дисперсию, корреляцию и так далее) для сложных моделей.
- Бутстрэп-выборки оказываются очень эффективны в оценке распределений на маленьких датасетах.



Генерируем  $K$  выборок фиксированного размера  $M$ , выбирая с возвращением из  $N$  имеющихся объектов. Доказывается, что каждый объект попадает в выборку с вероятностью  $1 - e^{-1}$ , если  $M = N$ .

Настраиваем  $K$  базовых моделей на этих выборках и агрегируем результат.

## Плюсы:

- Уменьшает переобучение, если базовые модели были переобучены (например, решающие деревья)

## Минусы:

- Время обучения увеличивается в  $K$  раз

# Бэггинг



## Метод

## Описание

### Бэгинг

Подвыборка обучающей выборки берётся с помощью бутстрепа

### Пэстинг (Pasting)

Случайная обучающая подвыборка

### Случайные подпространства (Random Subspaces)

Случайное подмножество признаков

### Случайные патчи (Random Patches)

Одновременно берём случайное подмножество объектов и признаков

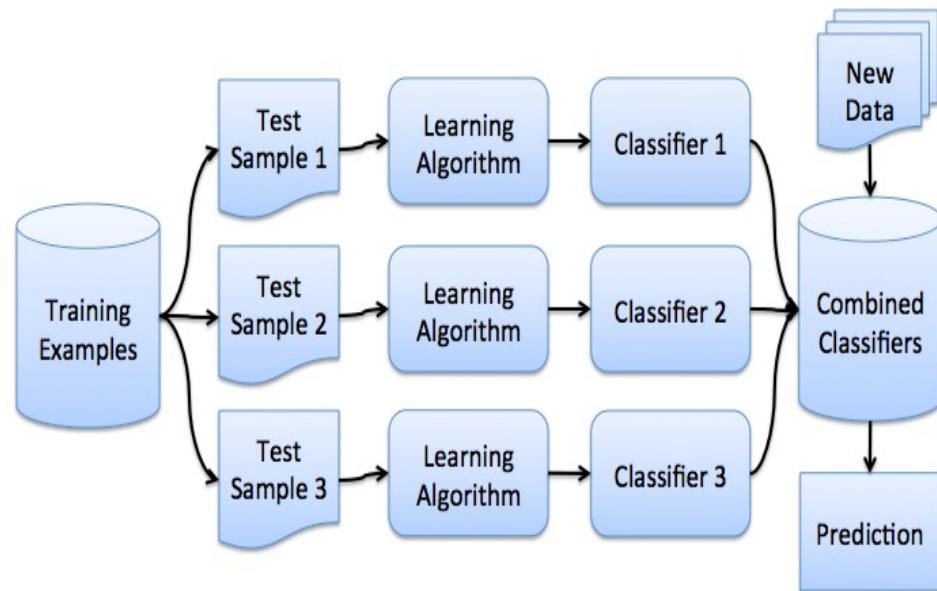
### cross-validated committees

k обучений на (k-1)-м фолде

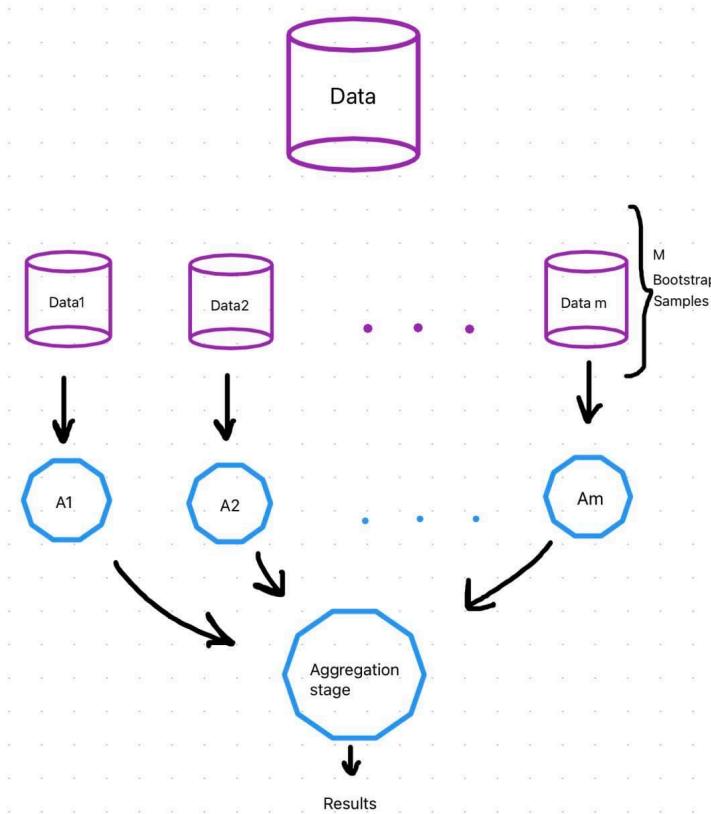
$$rez_{avg} = \frac{1}{m} \sum_{i=1}^m a_i$$

Задача регрессии

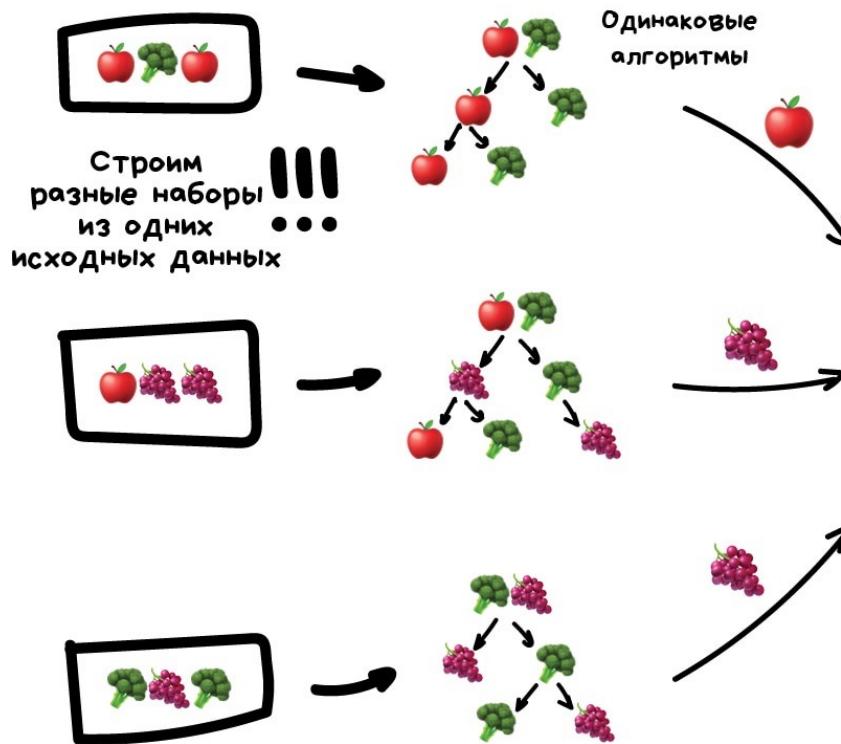
$$rez_{max} = argmax_i(cardinality(i|a_i = k))$$
 Задача классификации на  $k$  классов



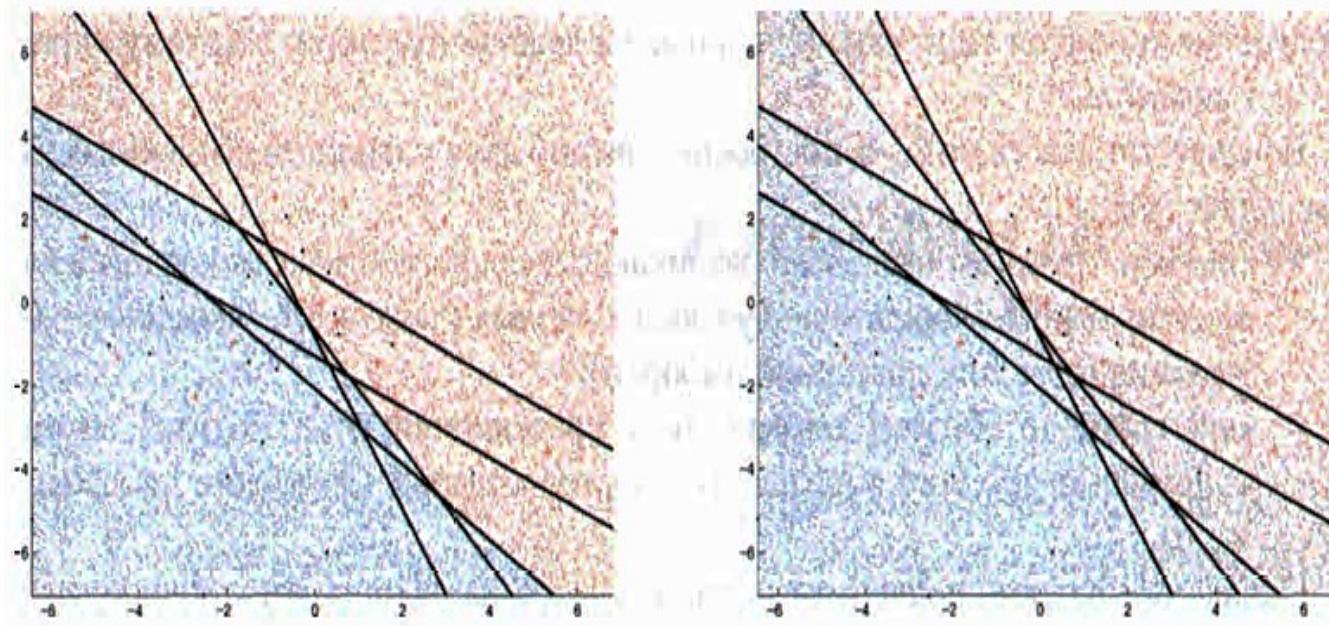
# Бэггинг



# Бэггинг

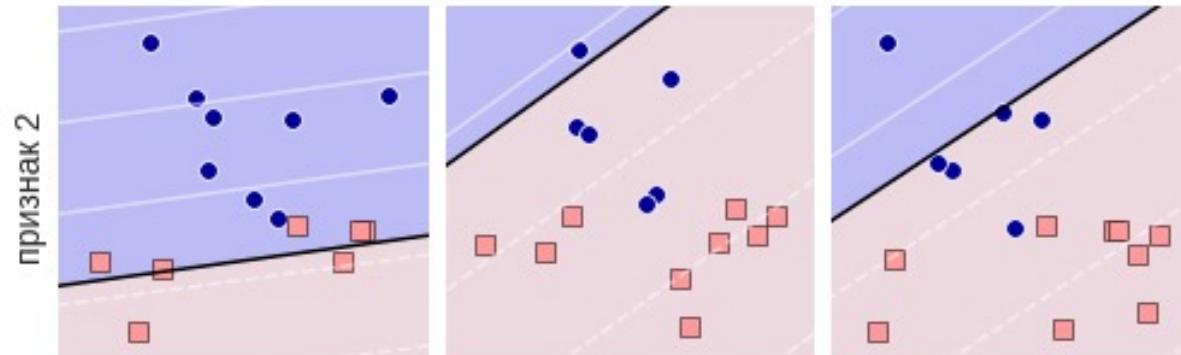


# Бэггинг линейных классификаторов



**(Слева)** Ансамбль из пяти базовых линейных классификаторов, построенный из усиливающих выборок с помощью баггинга. Решающим правилом является большинство голосов, оно порождает кусочно-линейную решающую границу. **(Справа)** Если преобразовать голоса в вероятности, то ансамбль превращается в группирующую модель: каждый сегмент пространства объектов получает немного отличающуюся вероятность

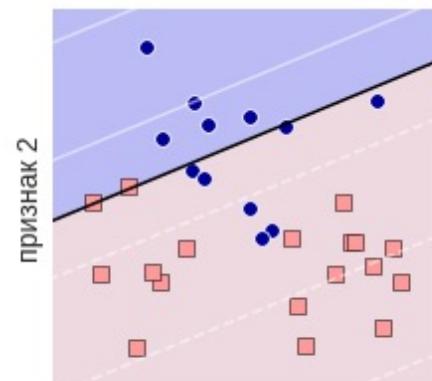
# Бэггинг пример



признак 1

признак 1

признак 1



признак 1

- класс 1
- класс 0

Способы повышения разнообразия с помощью рандомизации:

- bagging (bootstrap aggregating) — подвыборки обучающей выборки «с возвращением», в каждую выборку попадает  $(1 - \frac{1}{e}) \approx 63.2\%$  объектов
- pasting — случайные обучающие подвыборки
- random subspaces — случайные подмножества признаков
- random patches — случ. подмн-ва и объектов, и признаков
- cross-validated committees — выборка разбивается на  $k$  блоков ( $k$ -fold) и делается  $k$  обучений без одного блока

Пусть  $\mu: (G, U) \mapsto b$  — метод обучения по подвыборке  $U \subseteq X^\ell$ , использующий только признаки из  $G \subseteq F^n = \{f_1, \dots, f_n\}$

# Методы стохастического ансамблирования

**Вход:** обучающая выборка  $X^\ell$ ; параметры:  $T$ ,  
 $\ell'$  — объём обучающих подвыборок,  
 $n'$  — размерность признаковых подпространств,  
 $\varepsilon_1$  — порог качества базовых алгоритмов на обучении,  
 $\varepsilon_2$  — порог качества базовых алгоритмов на контроле;

**Выход:** базовые алгоритмы  $b_t$ ,  $t = 1, \dots, T$ ;

- 1: **для всех**  $t = 1, \dots, T$
- 2:  $U_t :=$  случайная подвыборка объёма  $\ell'$  из  $X^\ell$ ;
- 3:  $G_t :=$  случайное подмножество мощности  $n'$  из  $F^n$ ;
- 4:  $b_t := \mu(G_t, U_t)$ ;
- 5: **если**  $Q(b_t, U_t) > \varepsilon_1$  **то** не включать  $b_t$  в ансамбль;
- 6: **если**  $Q(b_t, X^\ell \setminus U_t) > \varepsilon_2$  **то** не включать  $b_t$  в ансамбль;

Ансамбль — простое голосование:  $b(x) = \frac{1}{T} \sum_{t=1}^T b_t(x)$

# Оценка ошибок

*Out-of-bag* — несмешённая оценка ансамбля на объекте:

$$\text{OOB}(x_i) = \frac{1}{|T_i|} \sum_{t \in T_i} b_t(x_i), \quad T_i = \{t : x_i \notin U_t\}$$

Несмешённая оценка ошибки ансамбля на обучающей выборке:

$$\text{OOB}(X^\ell) = \sum_{i=1}^{\ell} \mathcal{L}(\text{OOB}(x_i), y_i),$$

где  $\mathcal{L}(b(x_i), y_i)$  — значение функции потерь на объекте  $x_i$ .

**Оценивание важности признаков**  $f_j, j = 1, \dots, n$ :

$$\text{importance}_j = \frac{\text{OOB}^j(X^\ell) - \text{OOB}(X^\ell)}{\text{OOB}(X^\ell)} \cdot 100\%,$$

где при вычислении  $b_t(x_i)$  для  $\text{OOB}^j$  значения признака  $f_j$  случайным образом перемешиваются на всех объектах  $x_i \neq U_t$ .

# Случайный лес (Random Forest)

Базовые алгоритмы --- решающие деревья. Пусть всего  $B$  базовых алгоритмов и размер подвыборки признаков ---  $m$ . Тогда, алгоритм построения случайного леса следующий:

- Генерируем при помощи бэггинга  $B$  выборок
- Обучаем каждое решающее дерево на своей выборке, причем в **каждом узле признаки рассматриваются из случайно выбранного подмножества размера  $m$  из всех признаков.**

Агрегирование результата в случае классификации производится при помощи голосования большинства, а в случае регрессии --- среднее арифметическое.

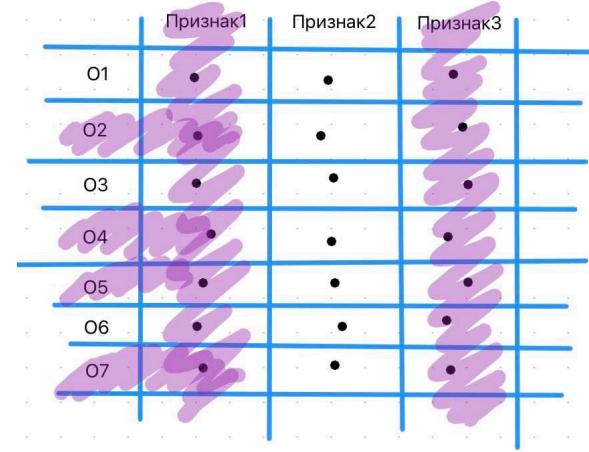
## Плюсы:

- Можно осуществить параллельную реализацию
- Не переобучается с ростом  $B$

## Минусы:

- Менее интерпретируемый, чем решающее дерево
- Деревья не исправляют ошибки друг друга

Для обучения должны использоваться глубокие деревья, иначе бэггинг над простыми моделями даст простую модель.



# Случайный лес (Random Forest)

## Обучение случайного леса:

- бэггинг над решающими деревьями, без pruning
- признак в каждой вершине дерева выбирается из случайного подмножества  $k$  из  $n$  признаков. По умолчанию  
 $k = \lfloor n/3 \rfloor$  для регрессии,  $k = \lfloor \sqrt{n} \rfloor$  для классификации

## Параметры, которые можно настраивать (в частности, по ОOB):

- число  $T$  деревьев
- число  $k$  случайно выбираемых признаков
- максимальная глубина деревьев
- минимальное число объектов в расщепляемой подвыборке
- минимальное число объектов в листьях
- критерий расщепления: MSE для регрессии, энтропийный или Джини для классификации

# Extra Random Trees

В каждом узле дерева генерируется случайно  $m$  пар (признак, порог).

## Плюсы:

- Упрощение модели Random Forest
- Более быстрые, чем Random Forest
- Не переобучается с ростом  $B$

## Минусы:

- Bias выше, чем у Random Forest, а variance --- меньше.

Для обучения должны использоваться глубокие деревья

# Случайный лес (Random Forest)

## Построение случайного леса

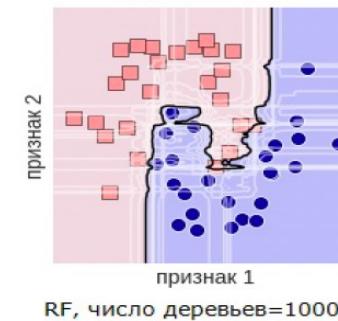
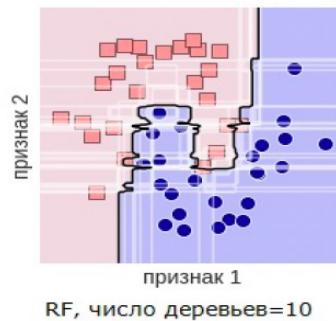
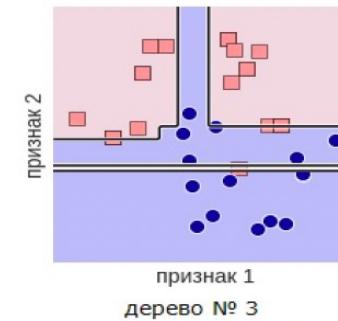
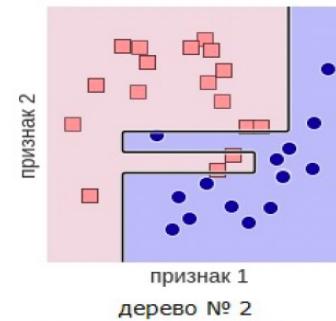
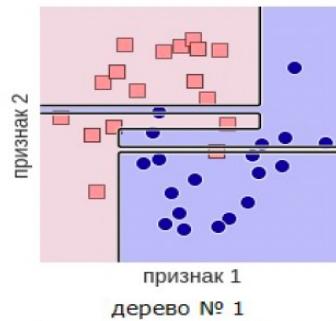
1. Выбирается бутстреп-подвыборка – на ней строится дерево
2. Строим дерево
  - 2.1. Для построения каждого расщепления просматриваем **max\_features** случайных признаков
  - 2.2. Как правило, дерево строится до исчерпания выборки (без прунинга)

Ответ леса:

по большинству (в задачах классификации),  
среднее арифметическое (в задачах регрессии)

# Случайный лес (Random Forest)

Пример разделения выборки с помощью отдельных деревьев  
(показаны соответствующие бутстреп-подвыборки)  
и случайного леса с числом деревьев 10, 100, 1000:



# Разновидности случайных лесов

- Случайный лес (Random Forest)
- Использование большого числа простых решающих деревьев в качестве признаков, в любом классификаторе.
- Oblique Random Forest, Rotation Forest  
 $f_v(x)$  — линейные комбинации признаков, выбираемые по энтропийному критерию информативности.
- Решающий список из решающих деревьев:
  - при образовании статистически ненадёжного листа этот лист заменяется переходом к следующему дереву;
  - следующее дерево строится по объединению подвыборок, прошедших через ненадёжные листы предыдущего дерева.

# Бустинг

Бустинг (англ. boosting — улучшение, усиление) — это процедура последовательного направленного построения ансамбля моделей машинного обучения, когда каждый следующий алгоритм стремится компенсировать ошибки предыдущих алгоритмов.

Изначально понятие бустинга возникло в связи с вопросом: возможно ли, имея множество относительно слабых (незначительно отличающихся от случайных) и простых моделей, построить сильную модель?

Бустинг направлен скорее на уменьшение смещения в данных, чем на снижение разброса в них. Поэтому в качестве базовых алгоритмов могут браться модели с достаточно высоким смещением, например, неглубокие случайные деревья.

$$rez_m = \sum_{i=1}^m c_i * a_i$$

$c_i$  — весовые коэффициенты

$a_i$  — Результаты базовых моделей

# Бустинг для классификации с двумя классами

Возьмём  $Y = \{\pm 1\}$ ,  $b_t: X \rightarrow \{-1, 0, +1\}$ ,  $C(b) = \text{sign}(b)$ .  
 $b_t(x) = 0$  — отказ (лучше промолчать, чем соврать).

Взвешенное голосование:

$$a(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t b_t(x)\right), \quad x \in X.$$

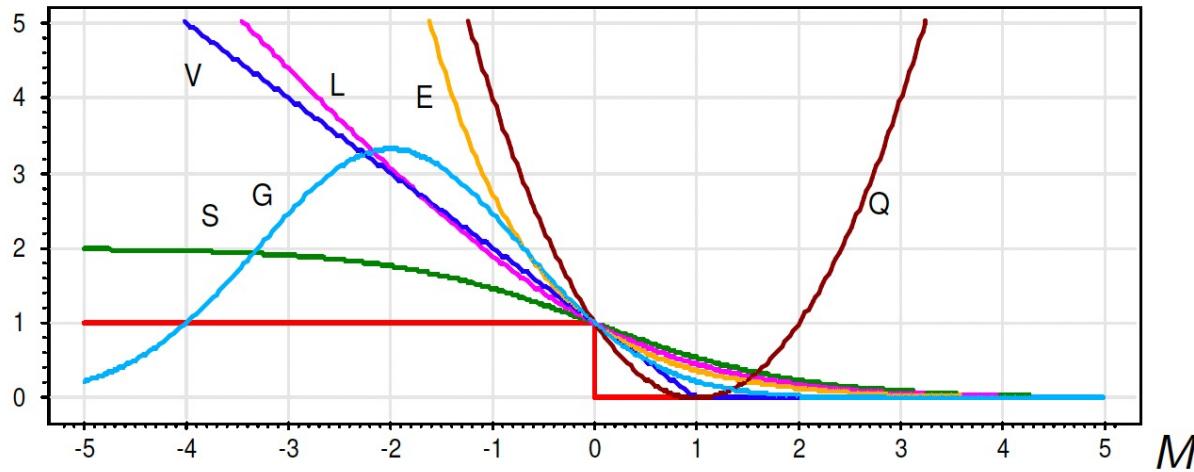
Функционал качества композиции — число ошибок на  $X^\ell$ :

$$Q_T = \sum_{i=1}^{\ell} \left[ y_i \sum_{t=1}^T \alpha_t b_t(x_i) < 0 \right].$$

Две основные эвристики бустинга:

- фиксация  $\alpha_1 b_1(x), \dots, \alpha_{t-1} b_{t-1}(x)$  при добавлении  $\alpha_t b_t(x)$ ;
- гладкая аппроксимация пороговой функции потерь [ $M \leq 0$ ].

# Гладкие аппроксимации пороговой функции потерь



$E(M) = e^{-M}$  — экспоненциальная (AdaBoost);

$L(M) = \log_2(1 + e^{-M})$  — логарифмическая (LogitBoost);

$Q(M) = (1 - M)^2$  — квадратичная (GentleBoost);

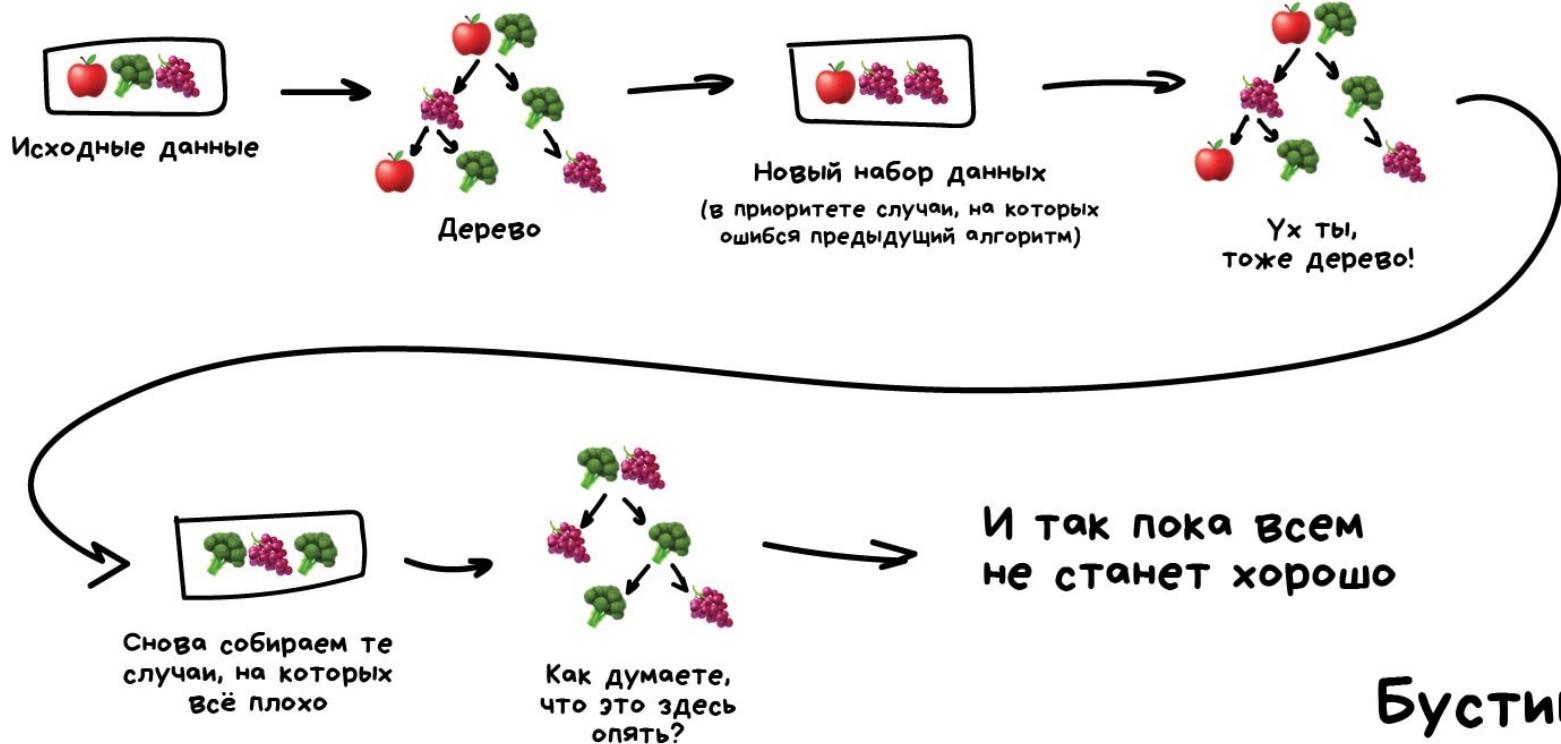
$G(M) = \exp(-cM(M + s))$  — гауссовская (BrownBoost);

$S(M) = 2(1 + e^M)^{-1}$  — сигмоидная;

$V(M) = (1 - M)_+$  — кусочно-линейная (из SVM);

# Бустинг

УНИВЕРСИТЕТ ИТМО



Бустинг

# Алгоритм Adaboost

**Вход:** обучающая выборка  $X^\ell$ ; **параметр  $T$** ;

**Выход:** базовые алгоритмы и их веса  $\alpha_t b_t$ ,  $t = 1, \dots, T$ ;

1: инициализировать веса объектов:

$$w_i := 1/\ell, \quad i = 1, \dots, \ell;$$

2: **для всех**  $t = 1, \dots, T$

3: обучить базовый алгоритм:

$$b_t := \arg \min_b N(b; W^\ell);$$

$$4: \quad \alpha_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^\ell)}{N(b_t; W^\ell)};$$

5: обновить веса объектов:

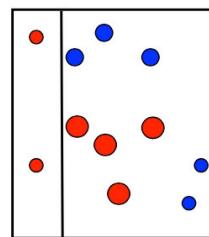
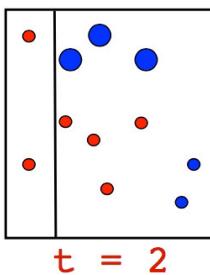
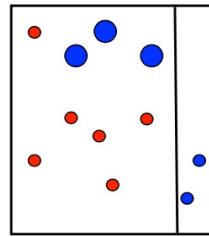
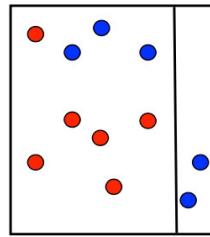
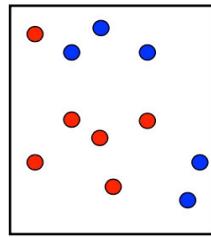
$$w_i := w_i \exp(-\alpha_t y_i b_t(x_i)), \quad i = 1, \dots, \ell;$$

6: нормировать веса объектов:

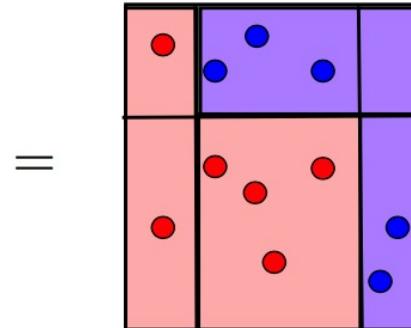
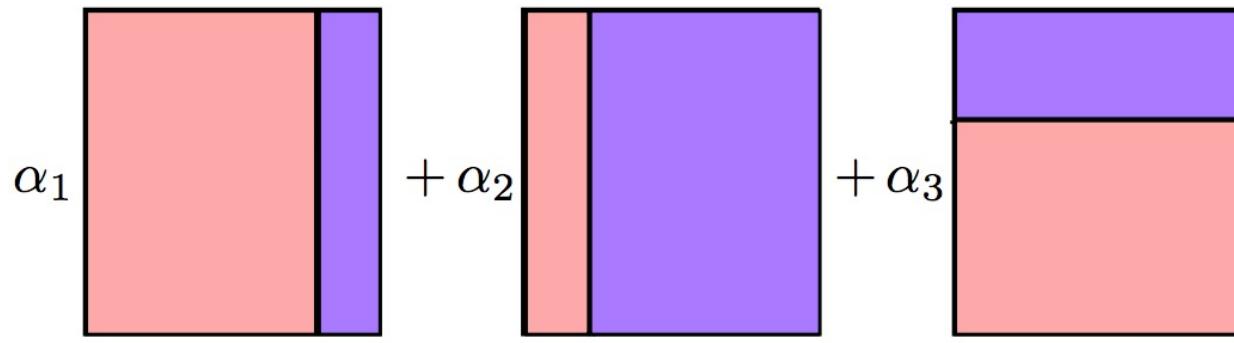
$$w_0 := \sum_{j=1}^{\ell} w_j;$$

$$w_i := w_i / w_0, \quad i = 1, \dots, \ell;$$

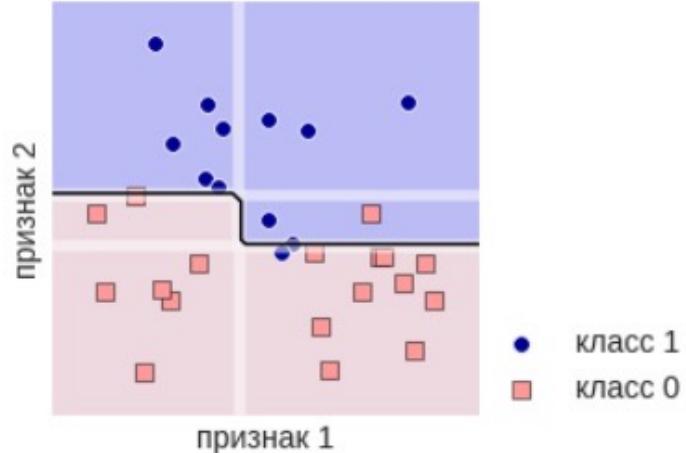
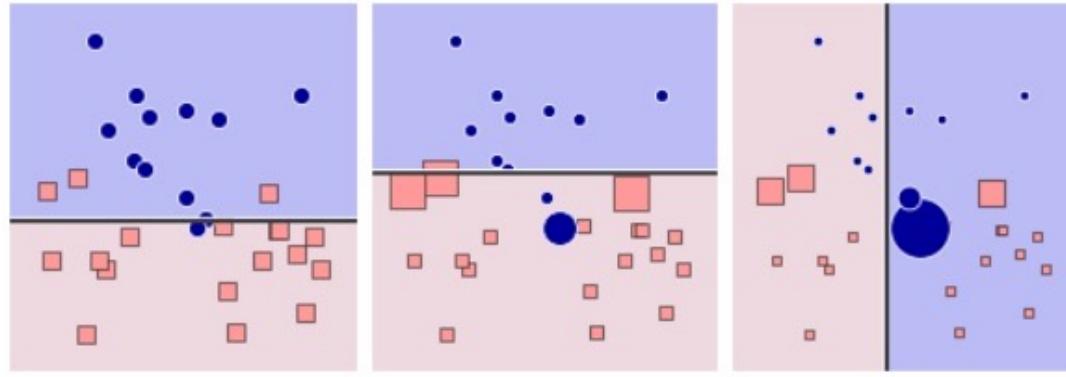
## Пример работы Adaboost для ансамбля из трех простых деревьев (пней)



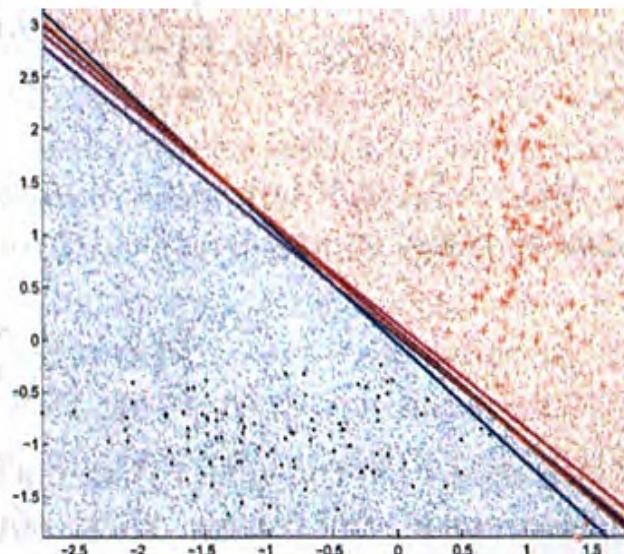
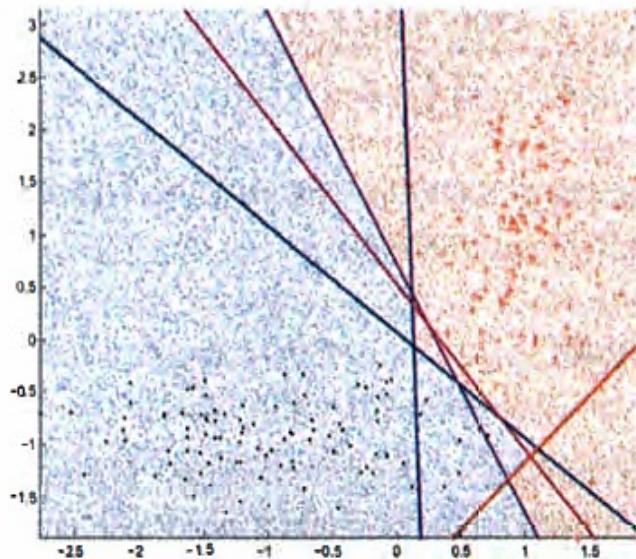
# Пример работы Adaboost (итоговый классификатор)



# Пример работы Adaboost

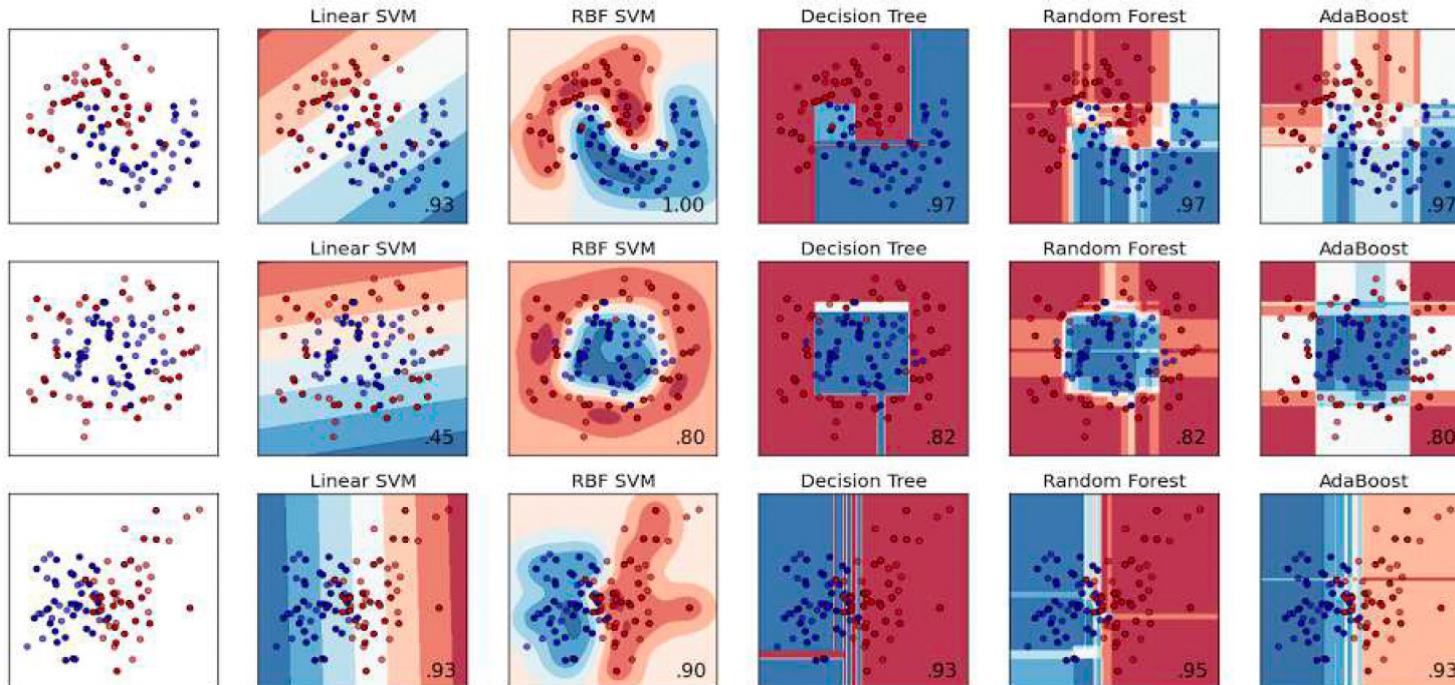


# Сравнение результатов бустинга для слабых и сильных моделей



# Случайный лес и бустинг в сравнении с другими методами

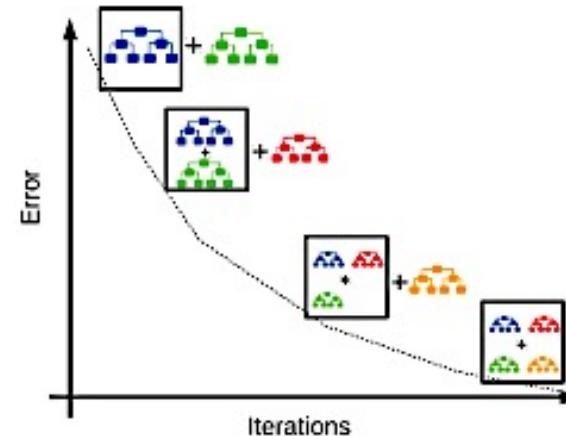
Эксперименты на трёх двумерных модельных выборках:



Решения могут выглядеть странно... тем не менее, RF и бустинг — одни из самых сильных универсальных методов в ML

# Градиентный бустинг

Градиентный бустинг – класс алгоритмов, представляющих бустинг как процесс градиентного спуска. В основе алгоритма лежит последовательное уточнение функции, представляющей собой линейную комбинацию базовых классификаторов, с тем чтобы минимизировать дифференцируемую функцию потерь. Градиентный бустинг – это один из самых универсальных и сильных методов машинного обучения, известных на сегодняшний день. В частности, на градиентном бустинге над деревьями решений основан алгоритм ранжирования выдачи компании Яндекс.



# Градиентный бустинг в задаче регрессии

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

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

$$b_1(x) := \arg \min_{b \in \mathcal{A}} \frac{1}{2} \sum_{i=1}^{\ell} (b(x_i) - y_i)^2$$

$$s_i^{(1)} = y_i - b_1(x_i)$$

$$b_2(x) := \arg \min_{b \in \mathcal{A}} \frac{1}{2} \sum_{i=1}^{\ell} (b(x_i) - s_i^{(1)})^2$$

# Градиентный бустинг в задаче регрессии

Каждый следующий алгоритм тоже будем настраивать на остатки предыдущих:

$$s_i^{(N)} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i), \quad i = 1, \dots, \ell;$$

$$b_N(x) := \arg \min_{b \in \mathcal{A}} \frac{1}{2} \sum_{i=1}^{\ell} (b(x_i) - s_i^{(N)})^2$$

Заметим, что остатки могут быть найдены как антиградиент функции потерь

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

Таким образом, выбирается такой базовый алгоритм, который как можно сильнее уменьшит ошибку композиции это свойство вытекает из его близости к антиградиенту функционала на обучающей выборке.

# Градиентный бустинг (общий случай)

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

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

- Способы выбора  $b_0(x)$ :
- самый популярный класс (в задачах классификации):

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

средний ответ (в задачах регрессии):

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

# Градиентный бустинг (продолжение)

Допустим, мы построили композицию  $a_{N-1}(x)$  из  $N - 1$  алгоритма, и хотим выбрать следующий базовый алгоритм  $b_N(x)$  так, чтобы как можно сильнее уменьшить ошибку:

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

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

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

вектор сдвигов  $s = (s_1, \dots, s_\ell)$  совпадает с антиградиентом:

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

# Градиентный бустинг (продолжение)

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

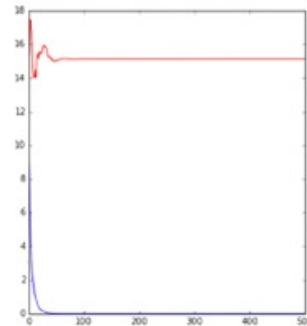
$$\gamma_N = \arg \min_{\gamma \in \mathbb{R}} \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + \gamma b_N(x_i))$$

- Если базовые алгоритмы очень простые, то они плохо приближают вектор антиградиента. По сути, добавление такого базового алгоритма будет соответствовать шагу вдоль направления, сильно отличающегося от направления наискорейшего убывания.
- Если базовые алгоритмы сложные, то они способны за несколько шагов бустинга идеально подогнаться под обучающую выборку что будет являться переобучением, связанным с излишней сложностью семейства алгоритмов.

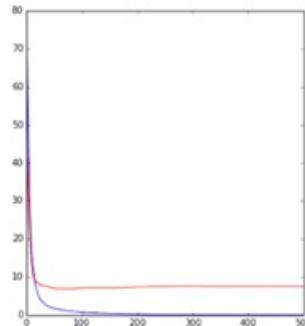
# Сокращение шага

Решение проблемы - сокращение шага: вместо перехода в оптимальную точку в направлении антиградиента делается укороченный шаг

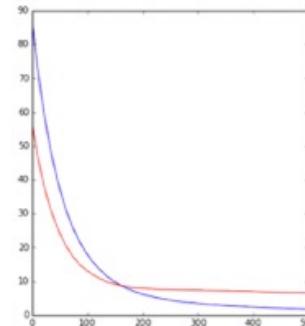
$$a_N(x) = a_{N-1}(x) + \eta \gamma_N b_N(x), \quad \eta \in (0, 1]$$



$$\eta = 1$$



$$\eta = 0.1$$



$$\eta = 0.01$$

# ФУНКЦИИ ПОТЕРЬ

- Регрессия: квадратичная  $L(y, z) = \frac{1}{2}(y - z)^2$
- Классификация: логистическая  $L(y, z) = \log(1 + \exp(-yz)).$

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

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

# Ансамбли моделей

- Ансамбли позволяют решать сложные задачи, которые плохо решаются отдельными базовыми алгоритмами
- Обычно ансамбль строится *алгоритмом-обёрткой* (*envelop*): базовые алгоритмы обучаются готовыми методами
- Базовые алгоритмы: компромисс качества/различность
- Две основные эвристики бустинга (и не только AdaBoost):
  - обучать базовые алгоритмы по одному
  - использовать гладкую замену пороговой функции потерь
- Важное открытие середины 90-х: обобщающая способность бустинга не ухудшается с ростом сложности  $T$
- Практическое сравнение бустинга и бэггинга:
  - бустинг лучше для классов с границами сложной формы
  - бэггинг лучше для коротких обучающих выборок
  - бэггинг легче распараллеливается
- Обучать ансамбль целиком слишком сложно.  
Поэтому обучаем базовые алгоритмы по одному.
- Градиентный бустинг — наиболее общий из всех бустингов:
  - произвольная функция потерь
  - произвольное пространство оценок  $R$
  - подходит для регрессии, классификации, ранжирования
- Чаще всего GB применяется к решающим деревьям