Машинное обучение (Machine Learning)

Композиционные методы машинного обучения

Уткин Л.В.

ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

Содержание

Презентация является компиляцией и заимствованием материалов из замечательных курсов и презентаций по машинному обучению:

К.В. Воронцова, А.Г. Дьяконова, Н.Ю. Золотых, С.И. Николенко, Andrew Moore, Lior Rokach, Matthias Schmid, Rong Jin, Cheng Li, Luis F. Teixeira, Alexander Statnikov и других.

Бэггинг, случайные пространства и стэкинг

Композиция классификаторов

"If you torture the data long enough, it will confess to anything"
- Ronald Coase

Теорема Кондорсе о присяжных (Condorcet Jury Theorem, 1784)

Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри, и стремиться к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.

Теорема Кондорсе о присяжных

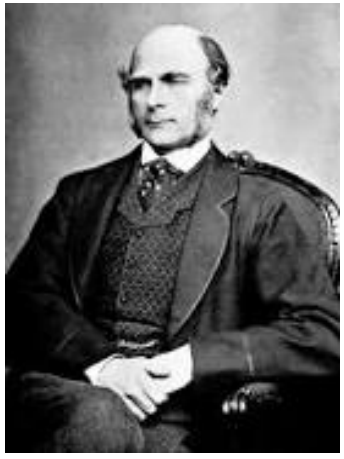
- N - число членов жюри
- p - вероятность правильного решения одного члена жюри
- μ - вероятность правильного решения жюри
- $$\mu = \sum_{i=m}^{N} C_N^i p^i (1-p)^{N-i}$$
- Если $p > 0.5$, то $\mu > p$.
- Если $N \to \infty$, то $\mu \to 1$.

Мудрость толпы (Wisdom of crowds)

- Почему вместе мы умнее, чем поодиночке?
- Совокупность знаний независимой группы людей превышает знания любого отдельного человека.

Мудрость толпы (Wisdom of crowds)

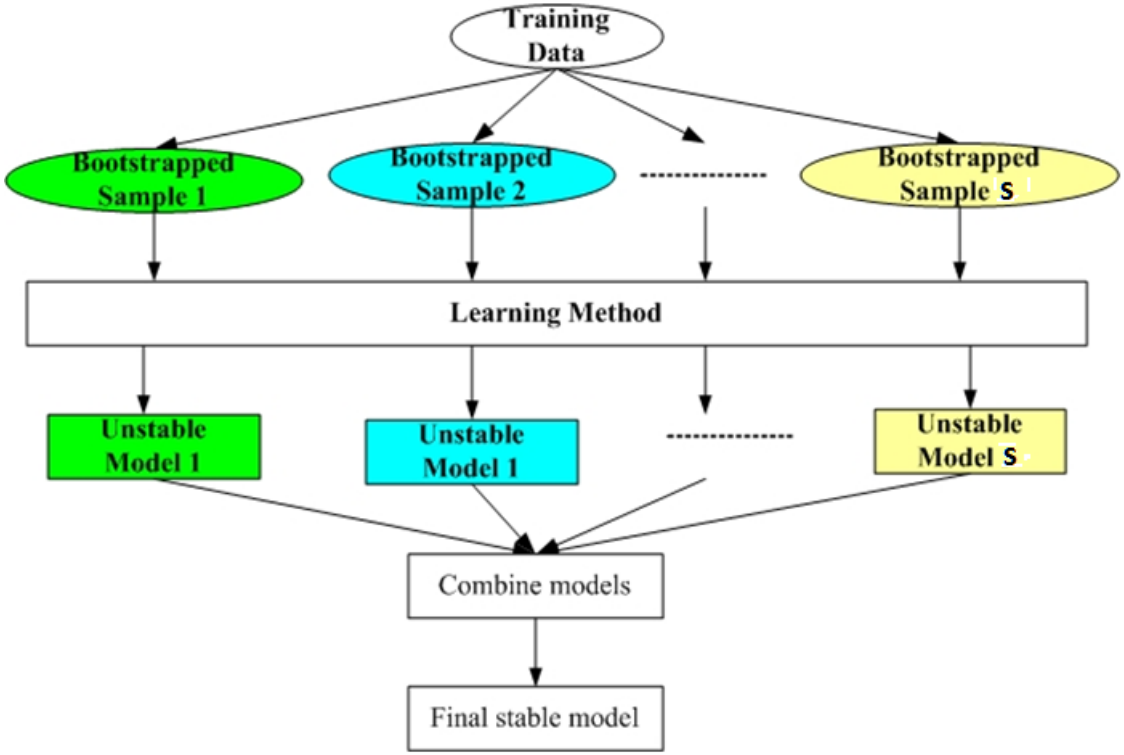
Francis Galton



- В 1906 посетил рынок скота.
- Крестьяне были приглашены, чтобы отгадать вес одного быка.
- Около 800 человек дали свои оценки и ни один не дал более-менее точное число: 1198 фунтов.
- Однако, на удивление, среднее этих 800 отгадываний было очень близко к точному значению. Оно было 1197 функтов.

Бэггинг

Bagging *(bootstrap aggregation)*



Слабый и сильный классификаторы

- **Слабый классификатор** - алгоритм обучения, позволяющий с вероятностью ошибки меньшей, чем простое угадывание (0.5 для бинарной классификации).
- **Сильный классификатор** - алгоритм обучения, позволяющий добиться произвольно малой ошибки обучения.

Бэггинг (алгоритм)

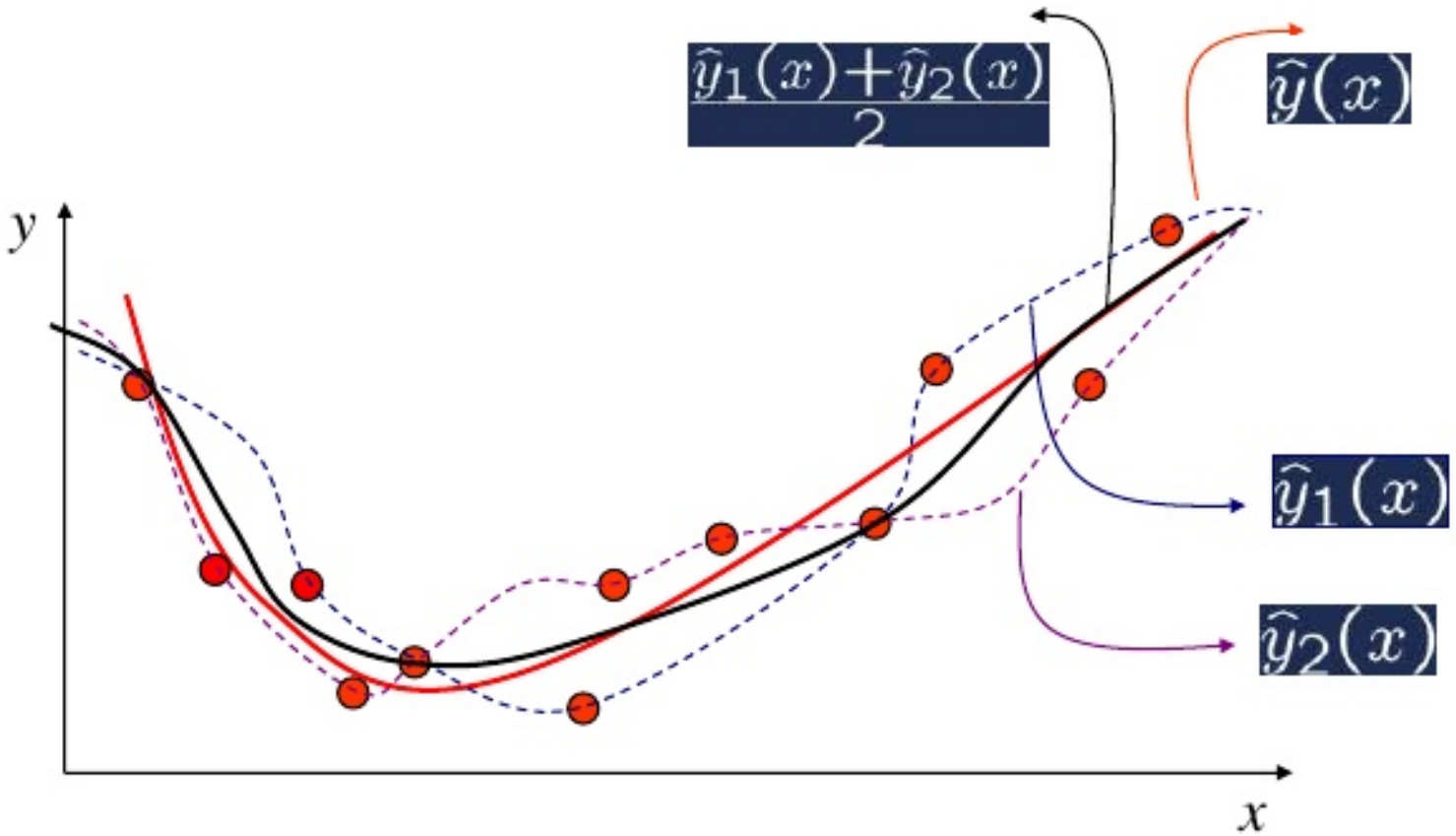
1. Дано: обучающая выборка $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$
2. Случайно выбираем t элементов из D с возвращением s раз: $D_1, ..., D_s$
3. Обучаемся на каждом $D_1, ..., D_s$ (**комитет моделей**) и получаем последовательность s выходов: $f_1(\mathbf{x}), ..., f_s(\mathbf{x})$ (**базовые алгоритмы или слабые классификаторы**)
4. Итоговый (**сильный**) классификатор: $f(\mathbf{x}) = \sum_{i=1}^{s} \operatorname{sign}(f_i(\mathbf{x}))$ (**простое голосование - классификация**) или $f(\mathbf{x}) = \frac{1}{s} \sum_{i=1}^{s} f_i(\mathbf{x})$ (**среднее - регрессия**)

Почему бэггинг?

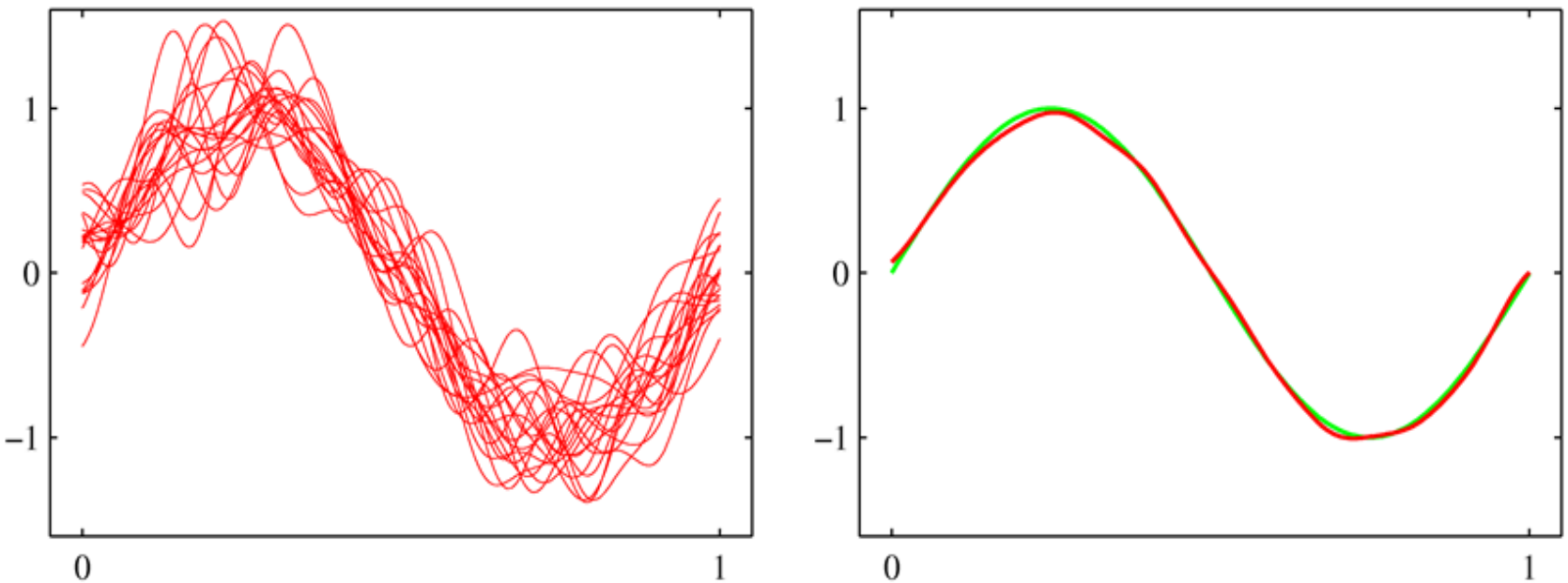
Эффективность бэггинга объясняется следующими обстоятельствами:

- Благодаря различности базовых алгоритмов, их ошибки взаимно компенсируются при голосовании.
- Объекты-выбросы могут не попадать в некоторые обучающие подвыборки.
- Если каждый классификатор имеет высокую дисперсию (нестабильность), то комбинированный классификатор имеет меньшую дисперсию по сравнению с отдельными классификаторами.

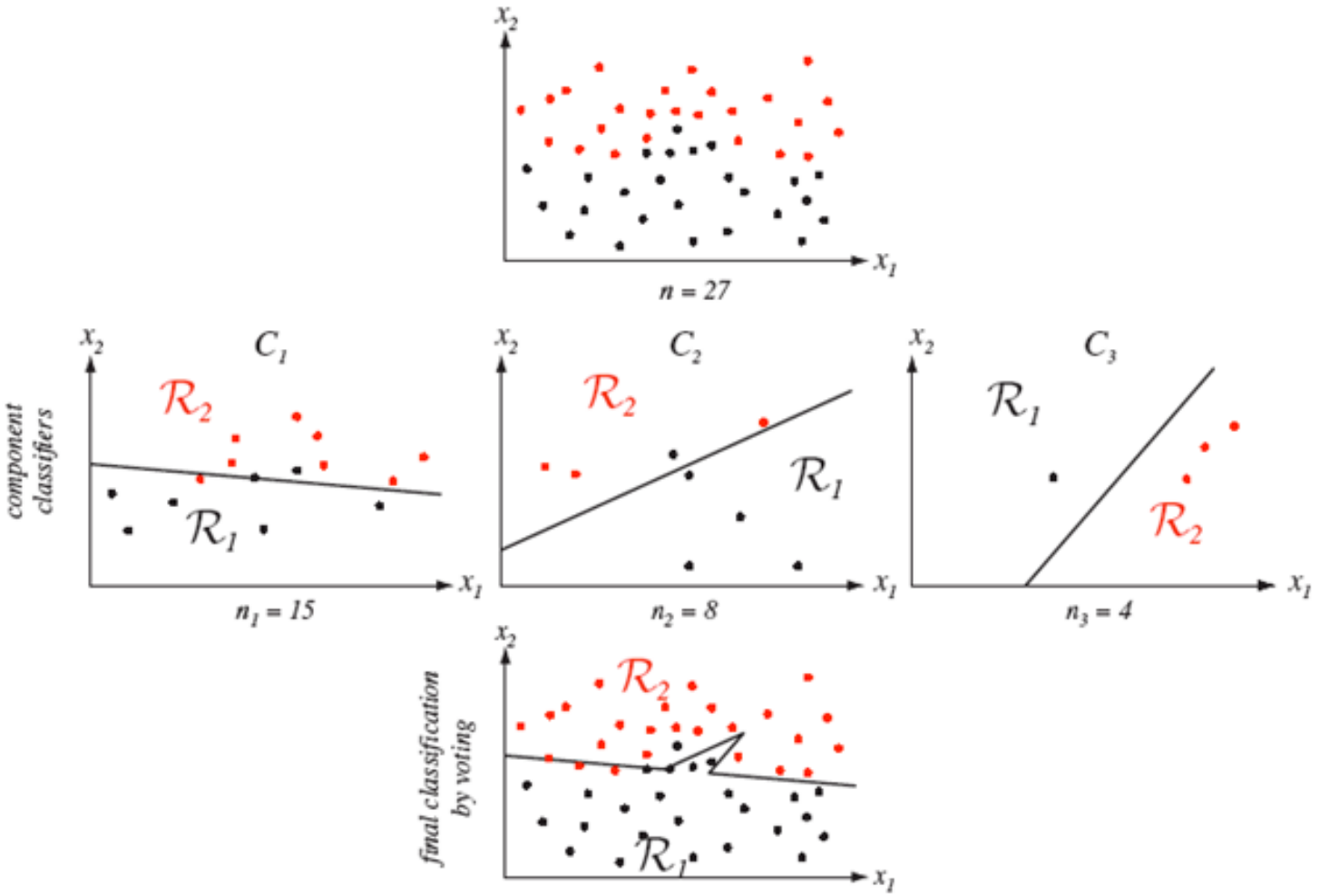
Почему бэггинг?



Почему бэггинг?



Бэггинг (пример нелинейности)



Почему бэггинг?

- Пусть настоящая функция, которую пытаемся предсказать – $h(\mathbf{x})$, т.е. модели выглядят как

$$f_i(\mathbf{x}) = h(\mathbf{x}) + \epsilon_i(\mathbf{x})$$

- Средняя ошибка модели

$$\mathbb{E}_{\mathbf{x}}\left[(f_i(\mathbf{x}) - h(\mathbf{x}))^2\right] = \mathbb{E}_{\mathbf{x}}\left[\epsilon_i^2(\mathbf{x})\right]$$

- Средняя ошибка всех моделей

$$E_{\text{ср}} = \frac{1}{s}\sum_{i=1}^{s}\mathbb{E}_{\mathbf{x}}\left[\epsilon_i^2(\mathbf{x})\right]$$

Почему бэггинг?

- Ошибка комитета (бэггинга)

$$E_{\text{ком}} = \mathbb{E}_{\mathbf{x}}\left[\left(\frac{1}{s}\sum_{i=1}^{s}\left(y_i(\mathbf{x}) - h(\mathbf{x})\right)\right)^2\right] = \mathbb{E}_{\mathbf{x}}\left[\left(\frac{1}{s}\sum_{i=1}^{s}\epsilon_i(\mathbf{x})\right)^2\right]$$

- Сравним

$$E_{\text{ср}} = \frac{1}{s}\sum_{i=1}^{s}\mathbb{E}_{\mathbf{x}}\left[\epsilon_i^2(\mathbf{x})\right], \quad E_{\text{ком}} = \mathbb{E}_{\mathbf{x}}\left[\left(\frac{1}{s}\sum_{i=1}^{s}\epsilon_i(\mathbf{x})\right)^2\right]$$

- Если предположить, что $\mathbb{E}_{\mathbf{x}}\left[\epsilon_i(\mathbf{x})\right] = 0$, и ошибки некоррелированы: $\mathbb{E}_{\mathbf{x}}\left[\epsilon_i(\mathbf{x})\epsilon_j(\mathbf{x})\right] = 0$, получим $E_{\text{ком}} = E_{\text{ср}}/s$.

Почему бэггинг?

- Не все так просто: ошибки на самом деле сильно коррелированы и $\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})\right] \neq 0$.
- Но всегда $E_{\text{ком}} \leq E_{\text{ср}}$!

Метод случайных подпространств (RSM)

- Random subspace method (RSM)
- Базовые алгоритмы обучаются на различных подмножествах признакового описания, которые также выделяются случайным образом.
- Метод может быть **эффективен**, когда:
  - большое число признаков
  - относительно небольшое число объектов
  - наличие избыточных неинформативных признаков.

Стэкинг (stacking) - "мета-классификация"

Стэкинг использует концепцию метаобучения, т.е. пытается обучить каждый классификатор, используя алгоритм, который позволяет обнаружить лучшую комбинацию выходов базовых моделей (Wolpert, 1992).

- Пусть D - обучающая выборка, и выбрано множество алгоритмов $A_1, ..., A_s$ (базовые классификаторы).
- Входные данные мета-классификатора - решения базовых классификаторов, т.е. множество меток классов, к которым базовые класс-ры отнесли описание входного объекта. Множество меток на входе мета-классификатора интерпретируется как множество признаков нового признакового пространства.

Стэкинг (stacking) - “мета-классификация”

- Обучающие и тестирующие данные для мета-классификатора (мета-данные) формируются базовыми классификаторами на основании тех данных, которыми располагают базовые классификаторы, т.е. метаданные - множества кортежей меток классов, полученных в качестве решений алгоритмами $A_1, ..., A_s$ при их тестировании на множестве входных описаний экземпляров объектов.

Стэкинг (stacking) - “мета-классификация”

- Каждому кортежу решений ставится метка (кортеж меток) класса, к которому этот объект относится на самом деле. Т.о., обучающие данные метаклассификатора - векторы вида $(x_{1i}, ..., x_{si}, y_i)$, где i - индекс экземпляра из тестовой выборки, $1, ..., s$ - индексы базовых классификаторов, x_{ki} - метки классов, полученных базовыми классификаторами A_k для тестируемого примера с номером i, y_i - истинное значение метки тестируемого примера.

Стэкинг


Мета класси-
фикатор
Алгоритм обучения мета
классификатора
Обучающие данные мета
классификатора
(мета данные)
Тестирую-
щие данные
Базовый класси-
фикатор 1
Базовый класси-
фикатор 2
...
Базовый класси-
фикатор S
Алгоритмы обучения базовых классификаторов
Обучающие и
тестирующие
данные 1
Обучающие и
тестирующие
данные 2
...
Обучающие и
тестирующие
данные k

Бустинг

Бустинг (Schapire, 89)

Бустинг *- итерационный алгоритм, реализующий "**сильный**" классификатор, который позволяет добиться произвольно малой ошибки обучения (на обучающей выборке) на основе композиции "**слабых**" классификаторов, каждый из которых лучше, чем просто угадывание, т.е. вероятность правильной классификации больше 0.5.*

Ключевая идея*: использование весовой версии одних и тех же обучающих данных вместо случайного выбора их подмножества.*

Отличие бустинга от бэггинга

- Бэггинг - примеры выбираются так, что каждый пример имеет одинаковые шансы попасть в обучающую подвыборку.
- Бустинг - обучающая выборка на каждой итерации определяется, исходя из ошибок классификации на предыдущих итерациях.

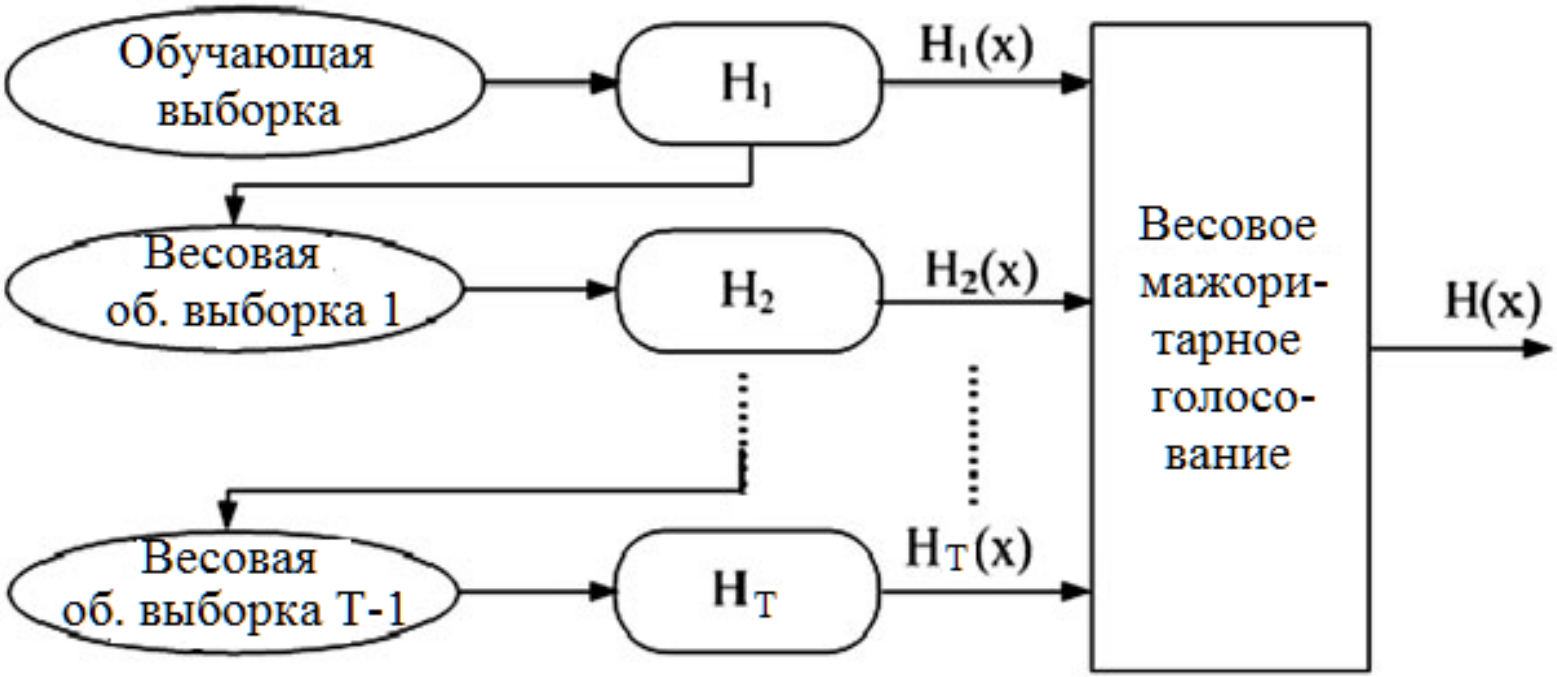
Особенности бустинга

- **Ключевая идея**: использование весовой версии одних и тех же обучающих данных вместо случайного выбора их подмножества.
- Слабые классификаторы образуются последовательно, различаясь только весами обучающих данных, которые зависят от точности предыдущих классификаторов.
- Большие веса назначаются “плохим” примерам, что позволяет на каждой итерации сосредоточиться на примерах, неправильно классифицированных.

Особенности бустинга

- Базовые классификаторы должны быть слабыми, из сильных хорошую композицию не построить (“бритва Оккама”)
- Причины этого:
  - сильный классификатор, давая нулевую ошибку на обучающих данных, не адаптируется и композиция будет состоять из одного классификатора
  - один, даже сильный, классификатор может дать “плохое” предсказание на данных тестирования, давая “хорошие” результаты на на обучающих данных.

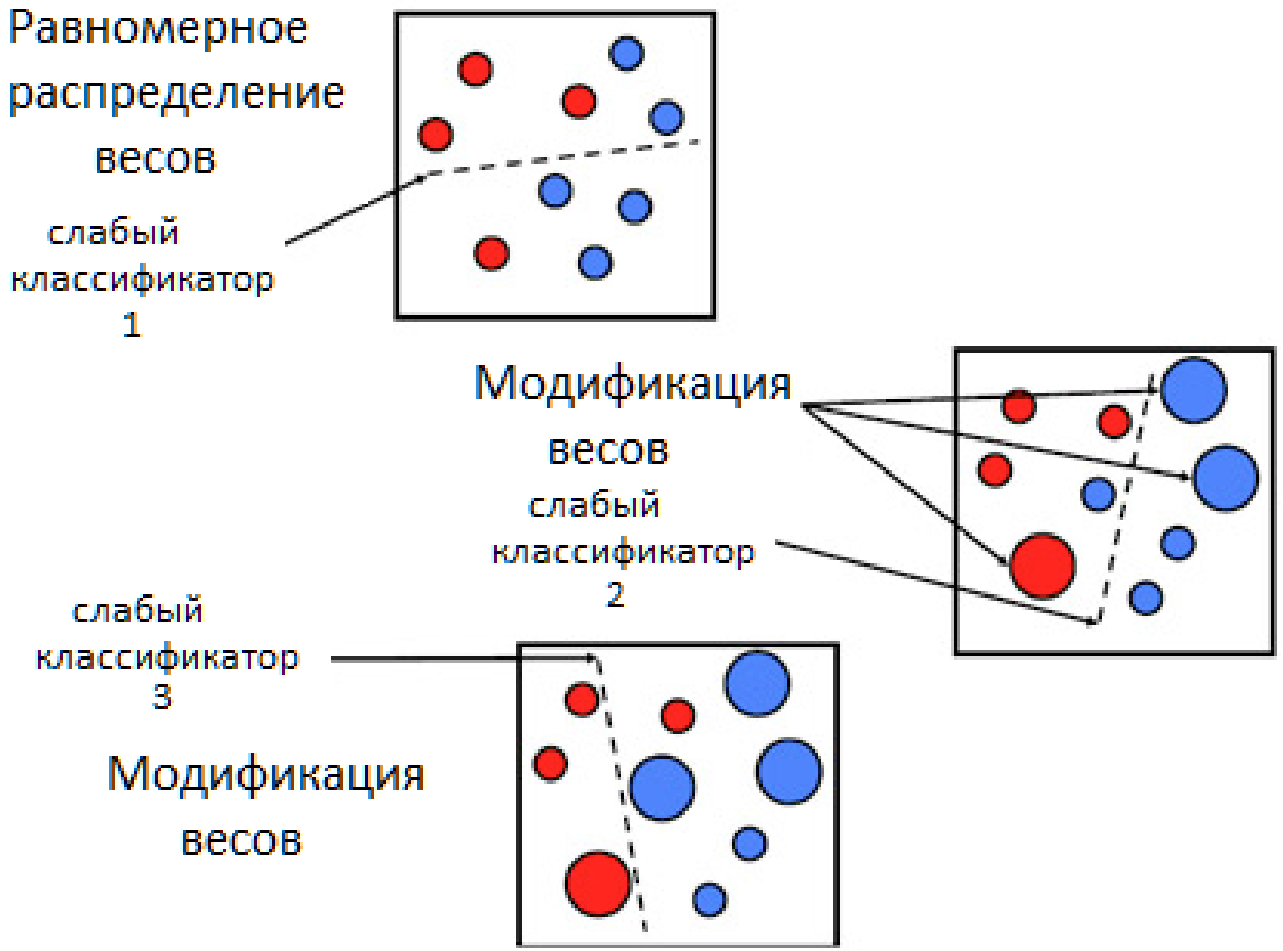
Бустинг (общая схема)



AdaBoost (Freund & Schapire, 1996)

1. **Исходные данные**:
   - n примеров $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$
   - слабый классификатор $h = h(\mathbf{x}, \theta, \mathbf{w})$
2. **Инициализировать** одинаковые веса примеров $w_i = 1/n,\ i = 1, ..., n$
3. **Цикл по** $t = 1, ..., T$:
   1. обучение слабого классификатора в соответствии с весами $\mathbf{w}$ и вычисление $h_t = h(\mathbf{x}, \theta_t, \mathbf{w})$
   2. вычисление средней ошибки классификации ε_t
   3. вычисление веса α_t слабого классификатора h_t (его надежность)
   4. модификация весов для следующей $t + 1$-ой итерации
4. **Конец цикла по** t
5. **Выход**: линейная комбинация $h_1, ..., h_t$

AdaBoost

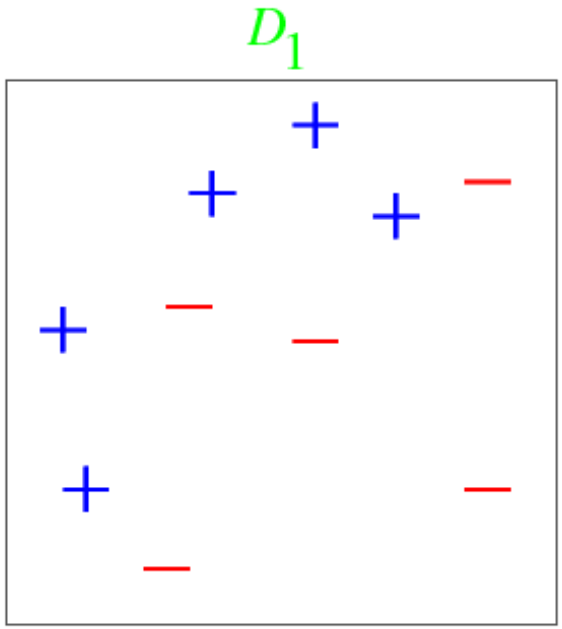


$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

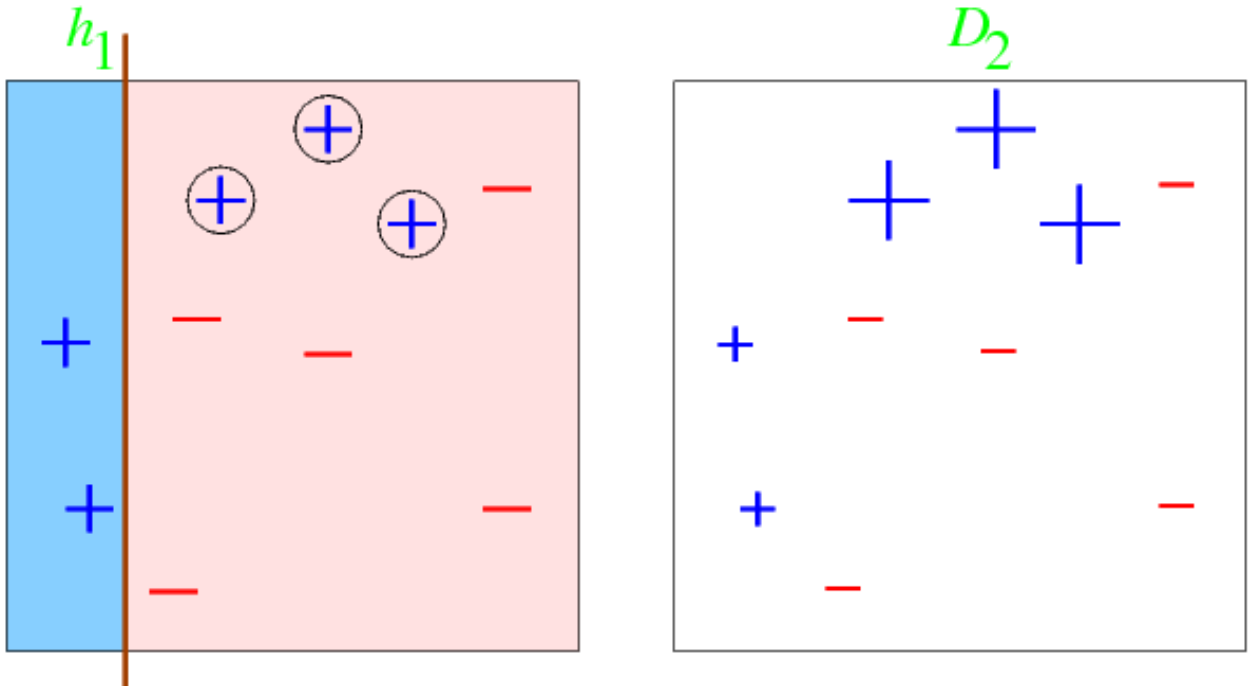
AdaBoost

1. **Исходные данные**: $S = \{(\mathbf{x}_1, y_1), .., (\mathbf{x}_n, y_n)\}$, $h = h(\mathbf{x}, \theta, \mathbf{w})$
2. **Инициализировать** веса $w_i = 1/n$, $i = 1, ..., n$
3. **Цикл по** $t = 1, ..., T$:
   1. обучение слабого классификатора $h_t(\mathbf{x}) = h(\mathbf{x}, \theta_t, \mathbf{w})$
   2. ошибка слабого классификатора $\varepsilon_t \leftarrow \sum_{i:h_t(\mathbf{x}_i) \neq y_i} w_i(t)$
   3. вес слабого классификатора $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
   4. $w_i(t+1) \leftarrow w_i(t) \cdot \exp\left(-\alpha_t \cdot y_i \cdot h_t(\mathbf{x}_i)\right)$
4. **Конец цикла по** t
5. **Выход**: $h(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

AdaBoost (пример)



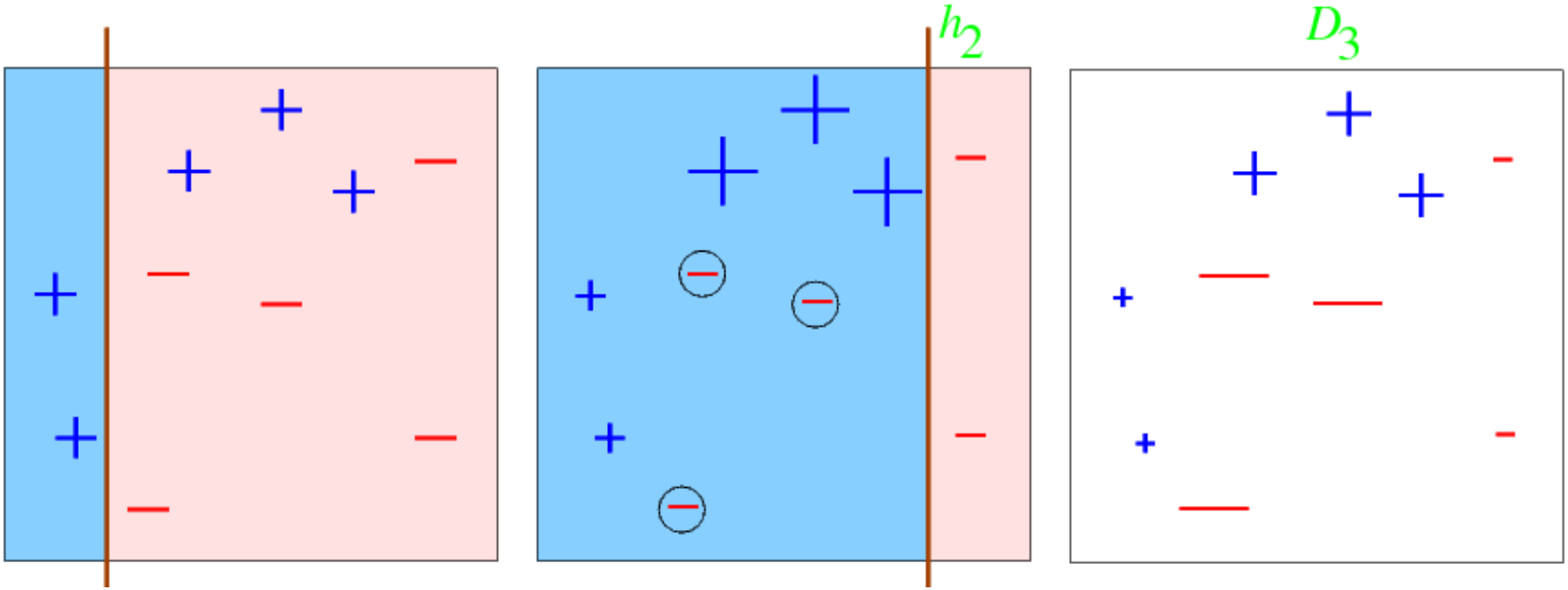
AdaBoost (пример)



$\epsilon_1 = 0.1 + 0.1 + 0.1 = 0.3,$

вес$=\alpha_1 = \frac{1}{2} \ln \frac{1-\epsilon_1}{\epsilon_1} = 0.42$

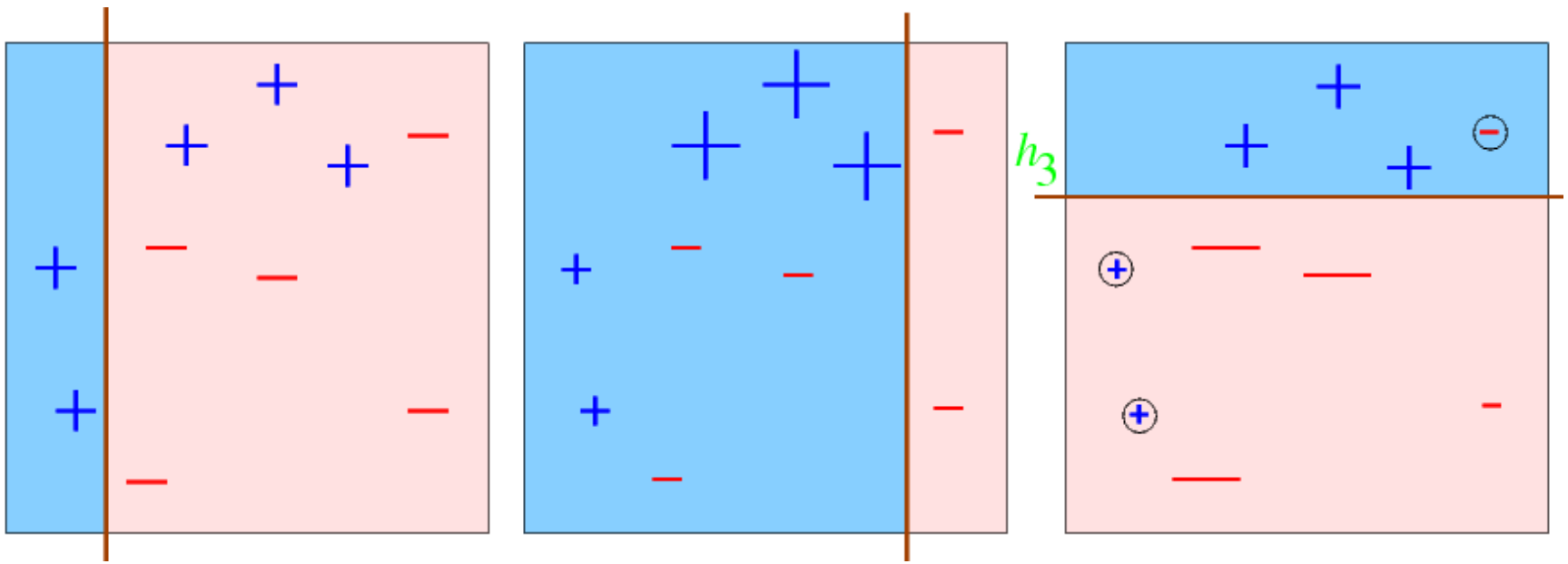
AdaBoost (пример)



$\epsilon_2 = 0.07 + 0.07 + 0.07 = 0.21,$

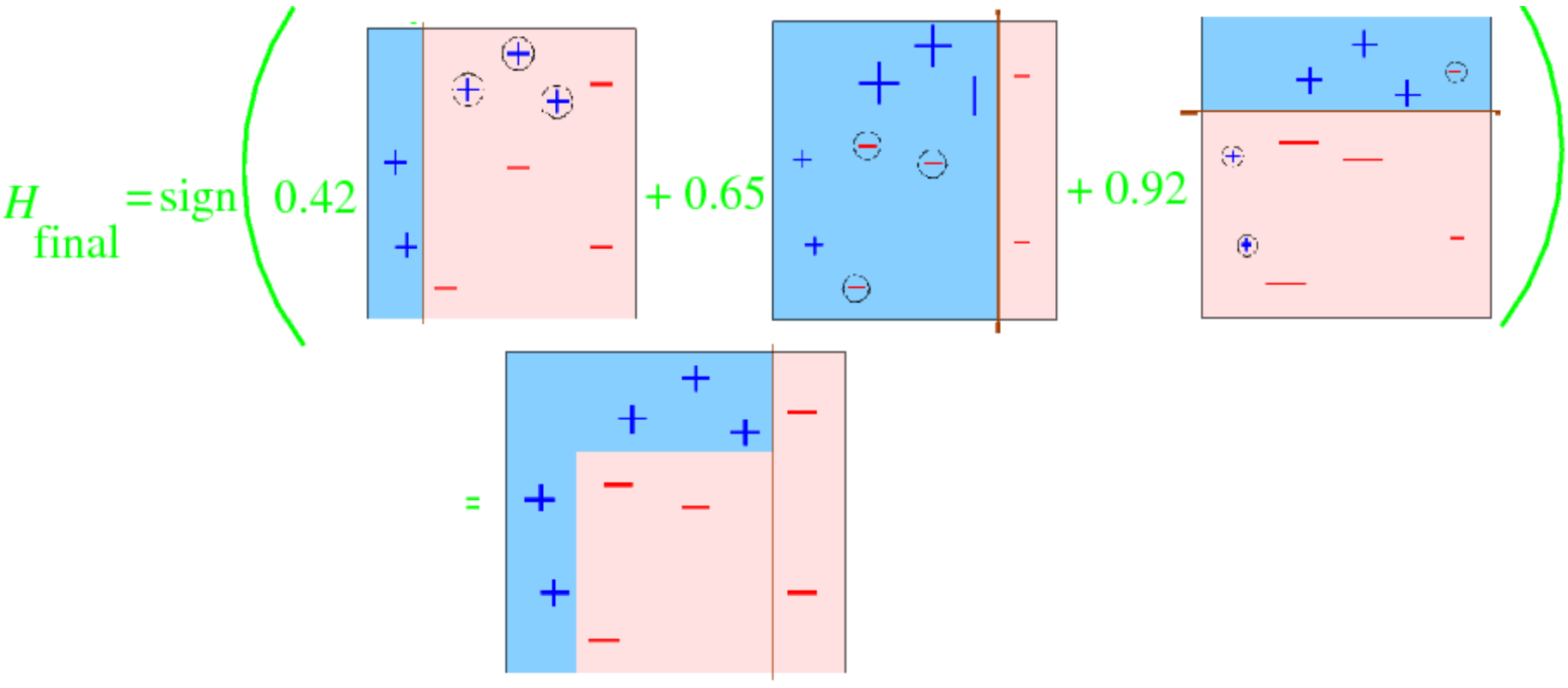
вес=$\alpha_2 = \frac{1}{2} \ln \frac{1-\epsilon_2}{\epsilon_2} = 0.65$

AdaBoost (пример)



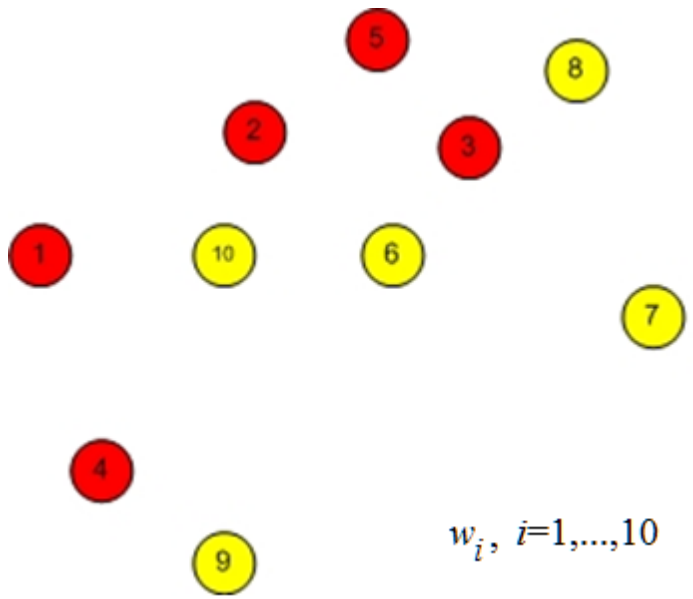
$\epsilon_3 = 0.14$, вес=$\alpha_3 = \frac{1}{2} \ln \frac{1-\epsilon_3}{\epsilon_3} = 0.92$

AdaBoost (пример)

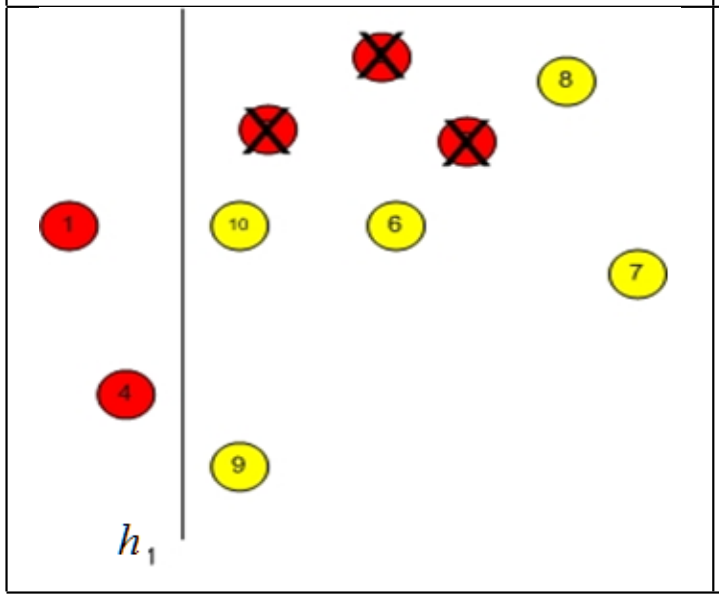
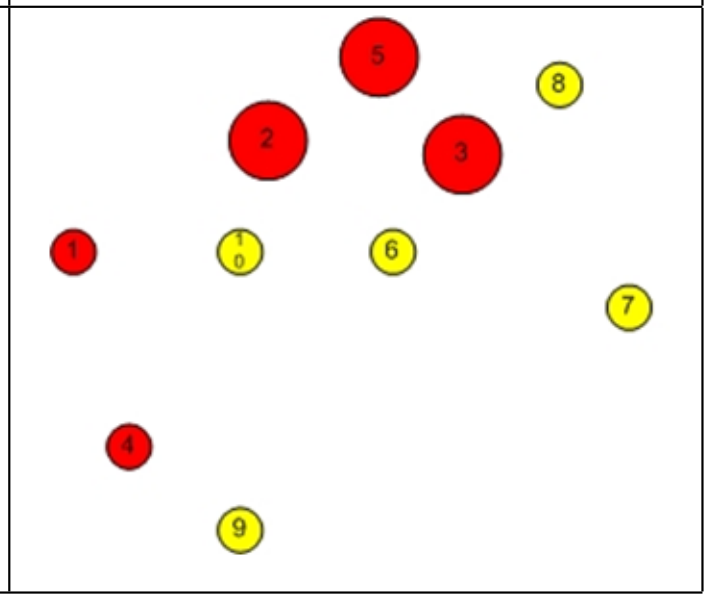


AdaBoost (пример)

Данные	1	2	3	4	5	6	7	8	9	10
W_i	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

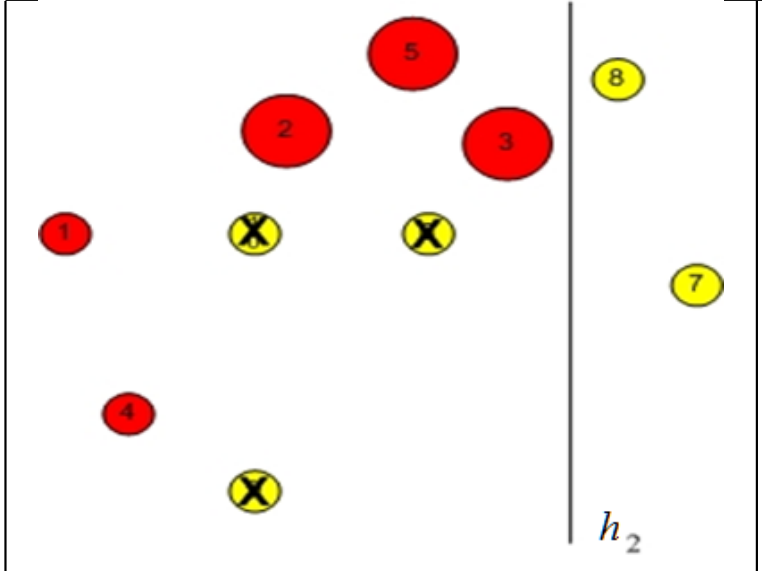
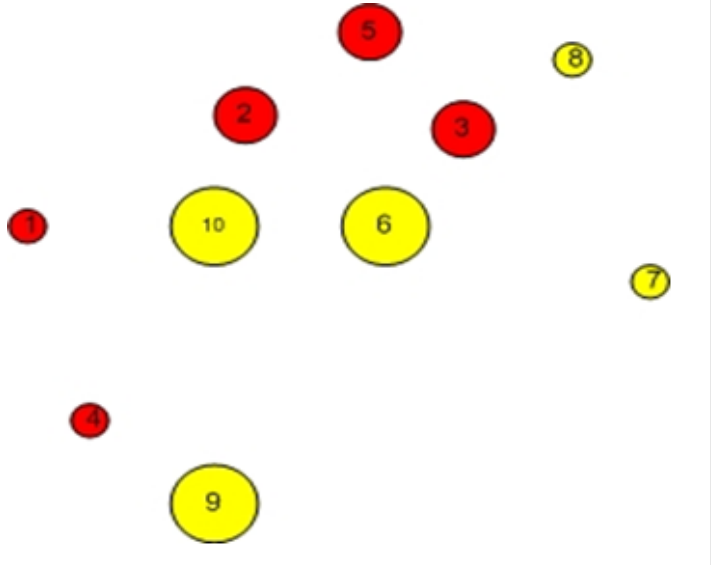


AdaBoost (пример)

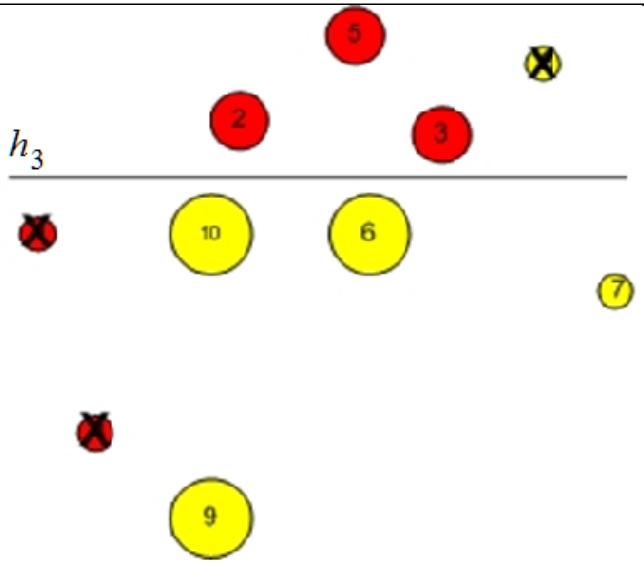
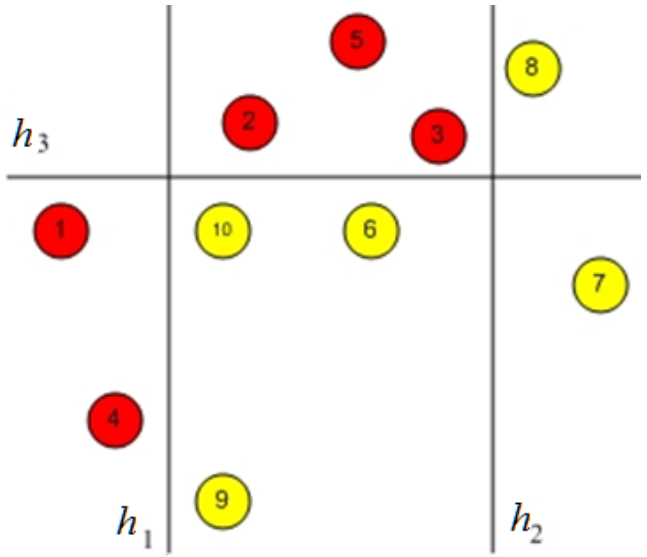
Первый (слабый) к-р h_1	Изменение весов
	

Данные	1	2	3	4	5	6	7	8	9	10
ош.	0	1	1	0	1	0	0	0	0	0
w_i	0.07	0.17	0.17	0.07	0.17	0.07	0.07	0.07	0.07	0.07

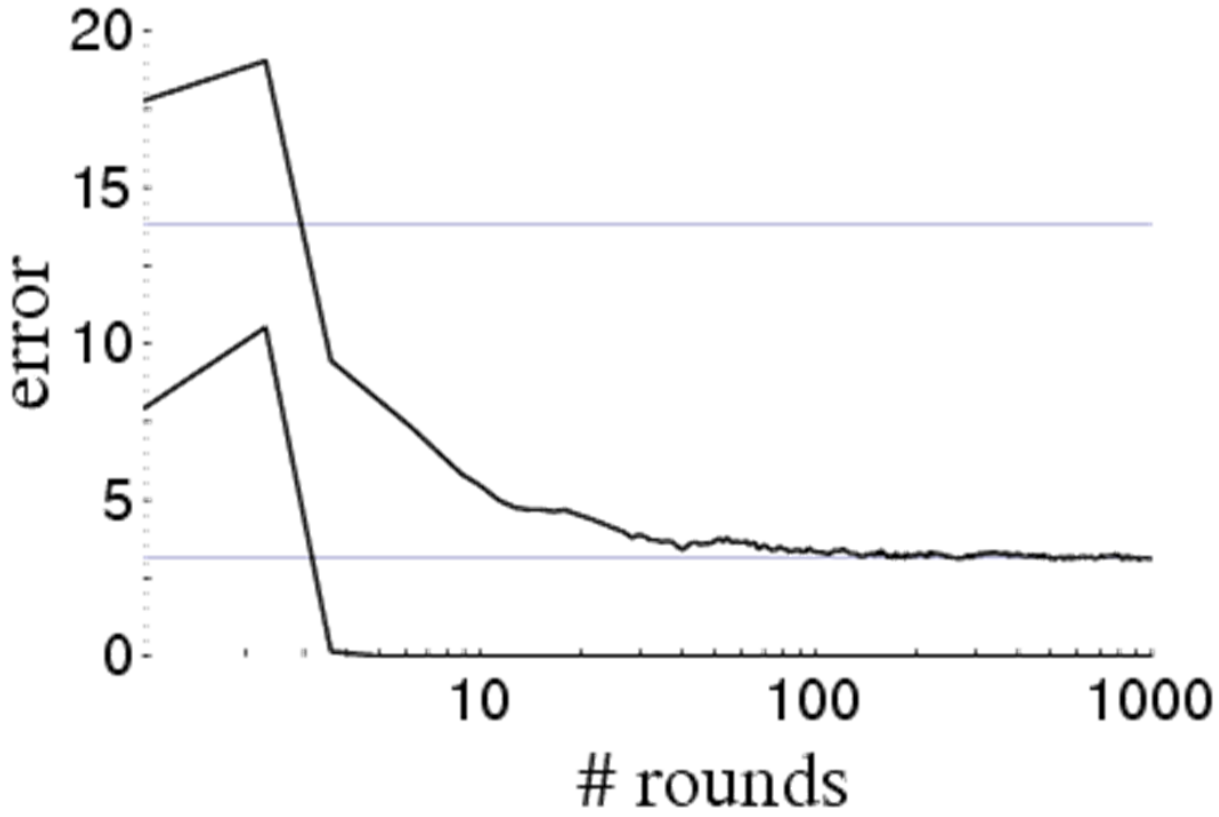
AdaBoost (пример)

Второй (слабый) к-р h_2	Изменение весов
	

AdaBoost (пример)

Третий (слабый) к-р h_3	Комбинирование к-ров
	

Экспериментальные характеристики бустинга (Schapire, 1989)



Как классифицировать с весами?

Общий подход: **Функционал качества** алгоритма a на выборке X есть

$$Q(a, X) = \sum_{i=1}^{n} w_i \cdot L(a, x_i)$$

В частности: **Метод ближайших соседей**:

$$a(x^*) = \arg\max_{y \in Y} \sum_{i=1}^{n} [y_i = y]\, w_i$$

Метод опорных векторов (Лагранжиан):

$$\max_{\alpha} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right)$$

при ограничениях

$$0 \leq \alpha_i \leq C w_i, \;\; i = 1, \ldots, n, \;\; \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Ошибка обучения AdaBoost

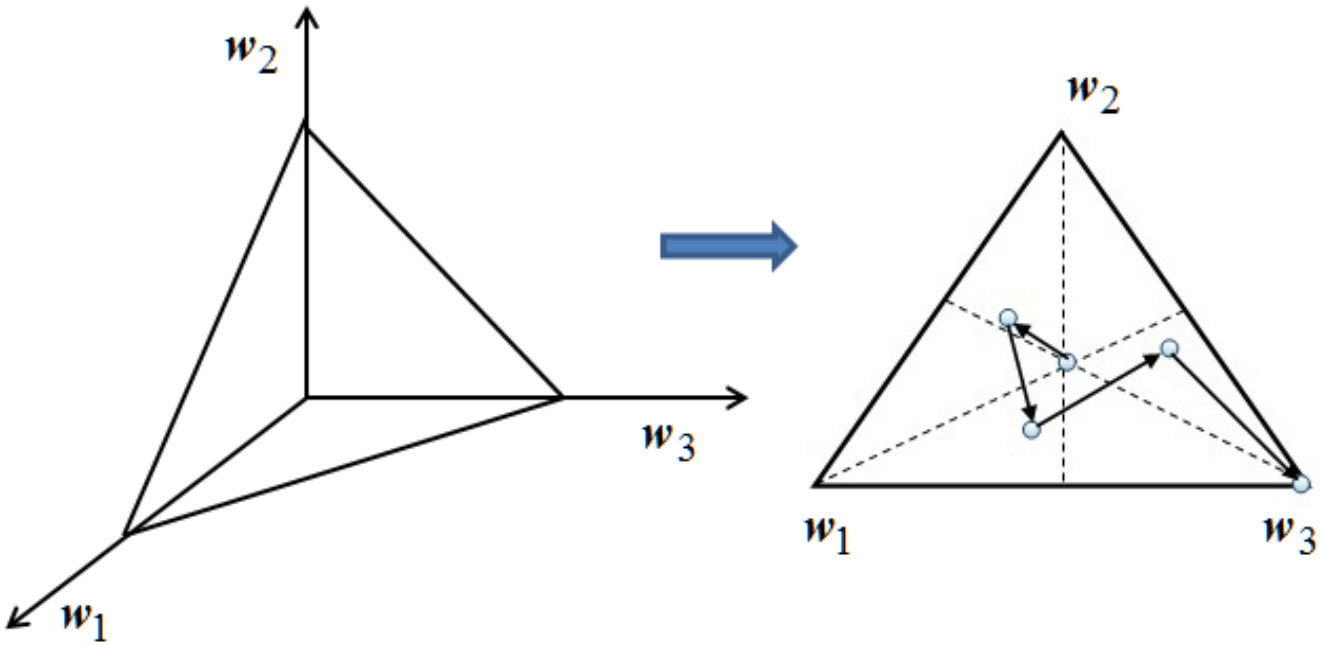
$$\text{ошибка обучения } (h(\mathbf{x})) \leq e^{-2\gamma T},$$

- $\gamma = \min_{t=1,\ldots,T} \gamma_t$, где $\varepsilon_t = 0.5 - \gamma_t$
- T - число итераций

Так ли все хорошо?

Симплекс весов в AdaBoost и переобучение

3 наблюдения: $w_1 + w_2 + w_3 = 1$



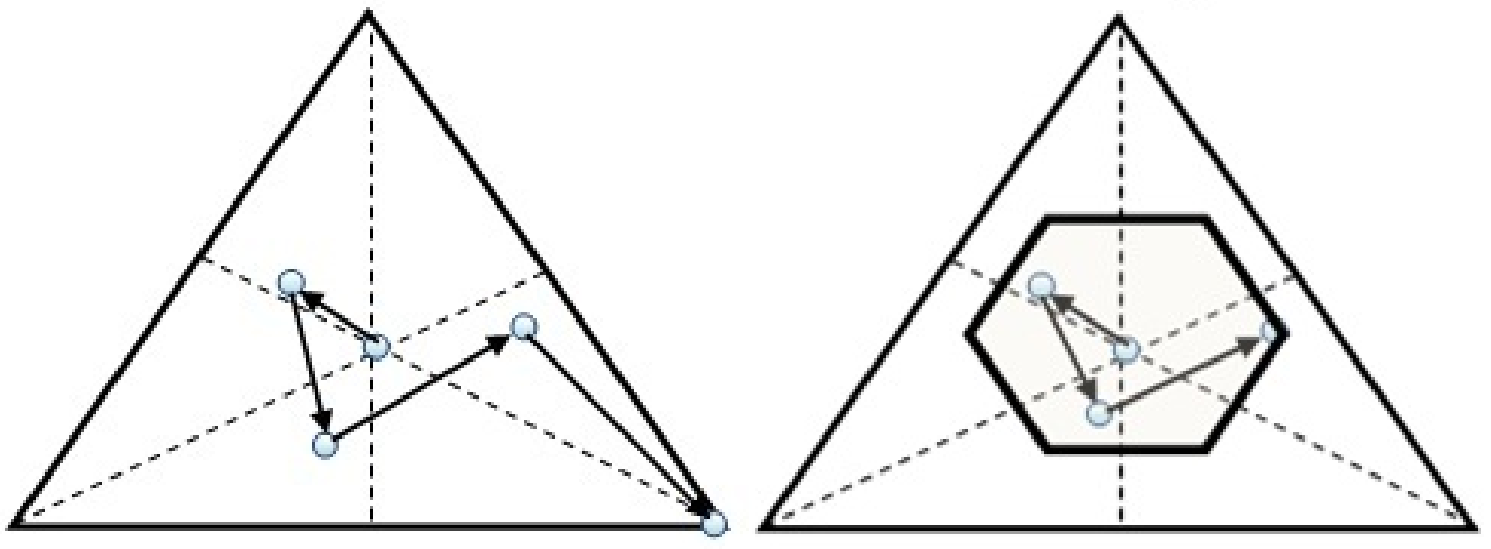
Переобучение: слишком большой вес назначается “плохим” примерам, например, выбросам, а “хорошие” примеры практически не участвуют в обучении.

Переобучение и как с ним бороться

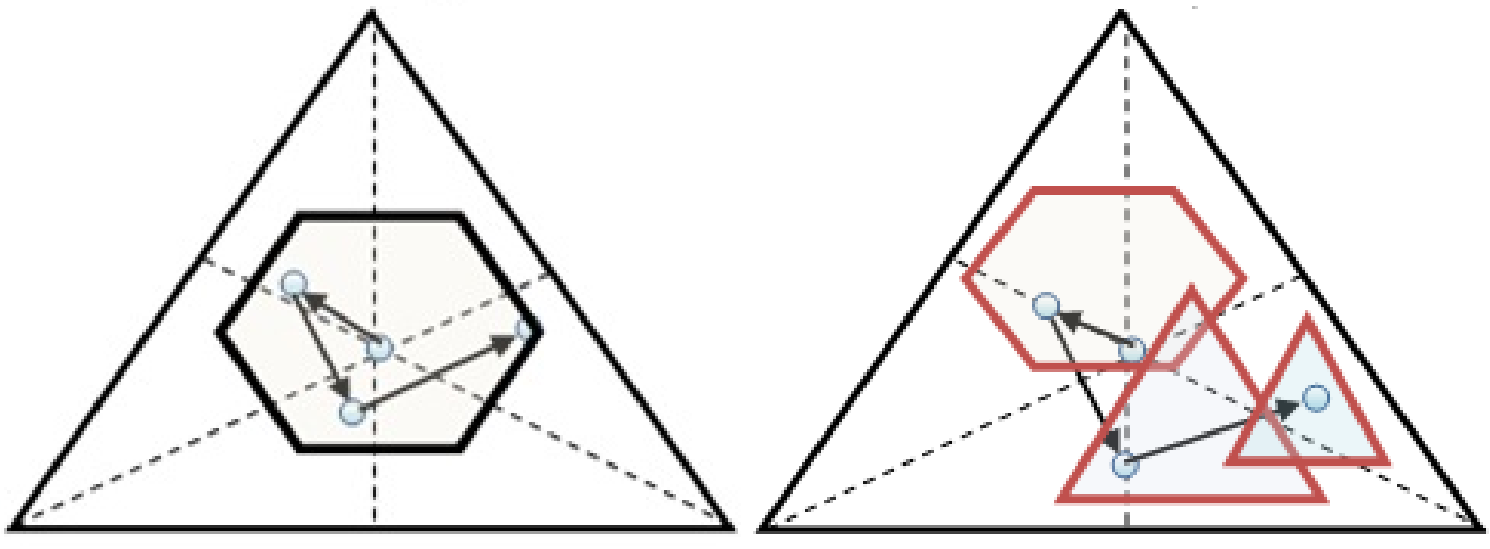
Два основных подхода:

1. Ограничение числа итераций
2. Ограничение множества возможных весов примеров в обучающей выборке

Ограничение множества весов



Ограничение множества весов



Ограничение множества весов и их двойная адаптация

1. Адаптация подмножеств весов (малых треугольников или многоугольников)
2. Адаптация внутри подмножеств весов (EPIBoost (Extreme Points Imprecise Boost) и IDMBoost (Imprecise Dirichlet Model Boost))
   - Utkin L.V. An imprecise boosting-like approach to classification // International Journal of Pattern Recognition and Artificial Intelligence, 2013, 27, Pp. 1-24.
   - Utkin L.V. The imprecise Dirichlet model as a basis for a new boosting classification algorithm // Neurocomputing, 2015, 151, Pp. 1374-1383.

Достоинства AdaBoost

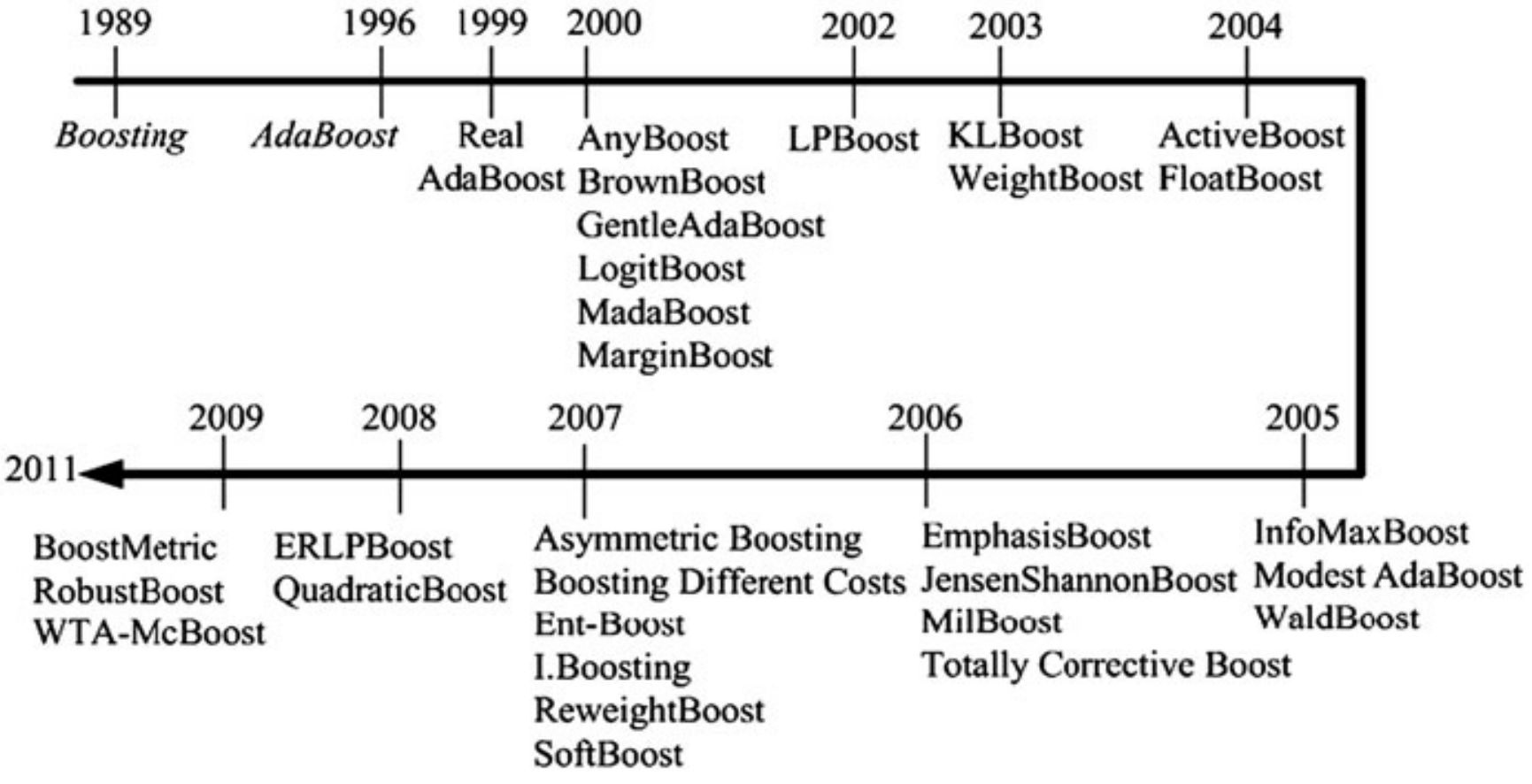
- Эффективный с вычислительной точки зрения
- Позволяет решать сложные задачи, которые плохо решаются отдельными алгоритмами
- Простой с точки зрения программирования
- Только один параметр настройки - число итераций
- Не требует априорной информации о слабом классификаторе
- Обеспечивает во многих случаях высокую точность прогнозирования
- Прост для модификаций

Недостатки AdaBoost

- Слишком эффективные или сложные слабые классификаторы могут привести к **переобучению**
- Чрезмерная чувствительность к выбросам (опять **переобучение**)
- Громоздкие композиции из сотен алгоритмов не интерпретируемы
- Требуются достаточно большие обучающие выборки (иначе **переобучение**)
- Не удается строить короткие композиции из «сильных» алгоритмов типа SVM (только длинные из слабых)
- Слишком “слабые” слабые классификаторы могут привести к слишком малым изменениям весов

Модификации AdaBoost

A.J.Ferreira and M.T. Figueiredo. Boosting Algorithms:A Review of Methods, Theory, and Applications, 2012



Алгоритмы бустинга для регрессии

Дано: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$
Найти: функцию $f(\mathbf{x})$, аппроксимирующую данные или минимизирующую функцию потерь
Функция потерь: разность квадратов
$L = \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2$

Алгоритм AdaBoost.RT (Solomatine and Shrestha, 2004)

Самый простой алгоритм:

- Ввести порог τ для относительной ошибки регрессии

$$e_i = |f(\mathbf{x}_i) - y_i| / y_i, \quad i = 1, ..., n,$$

  на каждой итерации
- Свести задачу к классификации с двумя классами: если $e_i > \tau$, то ошибка в точке i
- Далее AdaBoost

Алгоритм AdaBoost.RT

1. **Инициализировать** веса $w_i(1) \leftarrow 1/n$, $i = 1, ..., n$
2. **Цикл по** $t = 1, ..., T$:
   1. Построим регрессионную модель f_t, используя веса $w(t)$
   2. Относительная ошибка $e_i(t)$ для каждого примера: $e_i(t) \leftarrow |y_i - f_t(\mathbf{x}_i, \mathbf{w})|/y_i$
   3. Общая ошибка f_t: $\varepsilon_k \leftarrow \sum_{i:e_i(t)>\tau} w_i$
   4. $\alpha_t \leftarrow \ln\left((1-\varepsilon_t^l)/\varepsilon_t^l\right)$, где l может быть $1, 2, ...$
   5. Если $\varepsilon_k \leq \tau$, то $w_i(t+1) \leftarrow w_i(t) \cdot \exp\left(-\alpha_t(1-e_i(t))\right)$, иначе $w_i(t+1) \leftarrow w_i(t)$
3. **Конец цикла по** t
4. Результат $f(\mathbf{x}) \leftarrow \sum_{i=1}^{T} \alpha_t f_t(\mathbf{x})$

Алгоритм AdaBoost.R2 (Drucker, 1997)

1. **Инициализировать** веса $w_i(1) \leftarrow 1/n$, $i = 1, \ldots, n$
2. **Цикл по** $t = 1, \ldots, T$:
   1. Построим регрессионную модель f_t, используя веса $w(t)$
   2. $D_t \leftarrow \max_{j=1,\ldots,n} |y_j - f_t(\mathbf{x}_j, \mathbf{w})|$.
   3. Относительная ошибка $e_i(t)$ для каждого примера: $e_i(t) \leftarrow |y_i - f_t(\mathbf{x}_i, \mathbf{w})|/D_t$
   4. Вес функции f_t: $\epsilon_t \leftarrow \sum_{i=1}^{n} e_i(t) w_i(t)$
   5. **Если** $\epsilon_t > 0.5$, **то** Выход, **иначе**
      1. $\alpha_t \leftarrow \ln\left((1 - \epsilon_t)/\epsilon_t\right)$
      2. $w_i(t+1) \leftarrow w_i(t) \cdot \exp\left(-\alpha_t(1 - e_i(t))\right)$
3. **Конец цикла по** t
4. Результат $f(\mathbf{x}) \leftarrow \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x})$

Градиентный бустинг

Градиентный бустинг для регрессии

- Предположим, что мы угадали (но не точно): $f(x_1) = 0.8$, когда $y_1 = 0.9$, затем $f(x_2) = 1.4$, когда $y_2 = 1.3$, ...
- Как улучшить модель, если нельзя изменить параметры $f(x)$?

Градиентный бустинг для регрессии

- Предположим, что мы угадали (но не точно): $f(x_1) = 0.8$, когда $y_1 = 0.9$, затем $f(x_2) = 1.4$, когда $y_2 = 1.3$, ...
- Как улучшить модель, если нельзя изменить параметры $f(x)$?
- **Идея**: Можно добавить дополнительную модель h к f так, что новая функция: $f(x) + h(x)$

Градиентный бустинг для регрессии

$$\begin{aligned} f(x_1) + h(x_1) &= y_1 \\ f(x_2) + h(x_2) &= y_2 \\ &\dots \\ f(x_n) + h(x_n) &= y_n \end{aligned} \quad \text{или} \quad \begin{aligned} h(x_1) &= y_1 - f(x_1) \\ h(x_2) &= y_2 - f(x_2) \\ &\dots \\ h(x_n) &= y_n - f(x_n) \end{aligned}$$

Построим регрессионную модель для $h(x)$, т.е., модель для новой обучающей выборки

$$\{(x_1, y_1 - f(x_1)), ..., (x_n, y_n - f(x_n))\}$$

Градиентный бустинг для регрессии

- Построим регрессионную модель для $h(x)$, т.е., модель для новой обучающей выборки

$$\{(x_1, y_1 - f(x_1)), ..., (x_n, y_n - f(x_n))\}$$

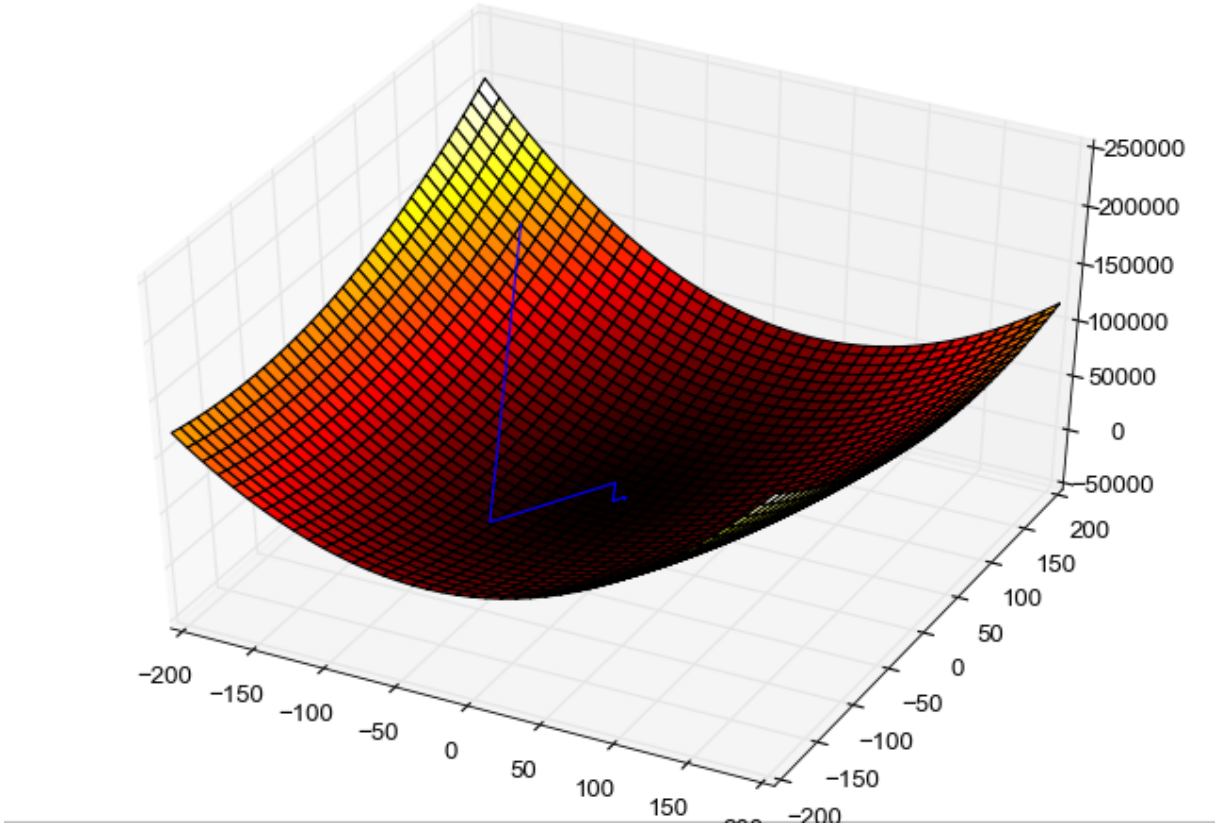
- Роль $h(x)$ - компенсировать недостаток существующей модели $f(x)$.
- Что делать, если $f(x) + h(x)$ снова нас не удовлетворяет?
- **Можем добавить другую модель $h_2(x)$!**

Градиентный бустинг для регрессии

Как это все соотносится с понятием градиентного спуска?
Градиентный спуск: минимизация функции, двигаясь в направлении, противоположном градиенту

$$\theta_i \leftarrow \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

Градиентный спуск



Градиентный бустинг для регрессии

- Если функция потерь $L = (y - f(x))^2/2$, то для поиска $f(x_1), ..., f(x_n)$ нужно минимизировать функционал риска

$$J = \sum_{j=1}^{n} L(y_j, f(x_j))$$

- Рассмотрим число $f(x_i)$ как параметр и возьмем производную

$$\frac{\partial J}{\partial f(x_i)} = \frac{\partial \sum_{j=1}^{n} L(y_j, f(x_j))}{\partial f(x_i)} = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = f(x_i) - y_i$$

- Отклонение $y_i - f(x_i) = -\partial J/\partial f(x_i)$ - отрицательный градиент

Градиентный бустинг для регрессии

$$f(x_i) \leftarrow f(x_i) + h(x_i)$$
$$f(x_i) \leftarrow f(x_i) + y_i - f(x_i)$$
$$f(x_i) \leftarrow f(x_i) - 1\frac{\partial J}{\partial f(x_i)}$$
$$\theta_i \leftarrow \theta_i - \rho\frac{\partial J}{\partial \theta_i}$$

- Модификация f на основе **отклонений** $\Leftrightarrow$ модификация f на основе отрицательного **градиента**
- Модифицируем модель, используя градиентный спуск
- Оказывается **градиенты** - более общее и полезное понятие, чем **отклонения**

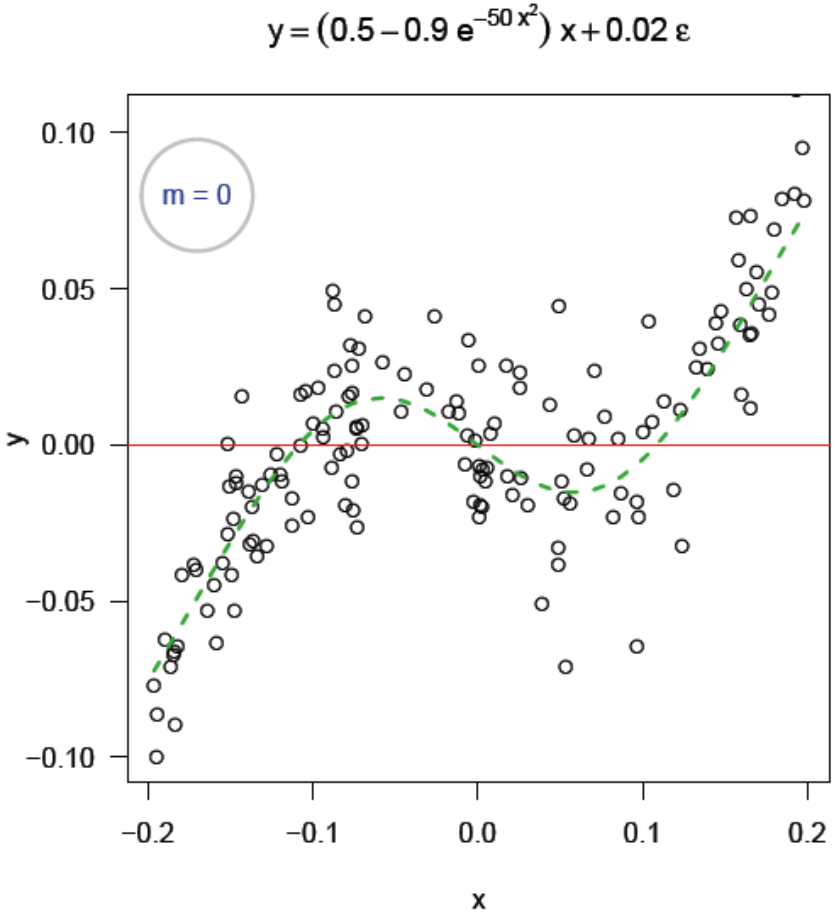
Алгоритм градиентного бустинга для регрессии с квадратичными потерями

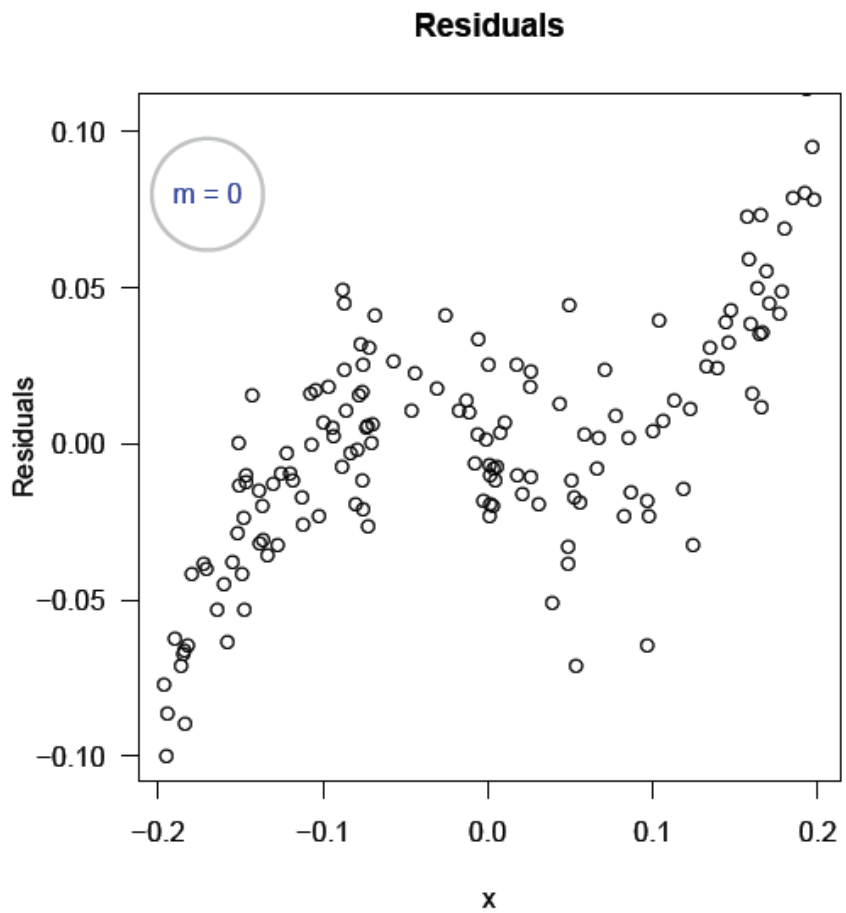
$$-g(x_i) = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = y_i - f(x_i)$$

1. Начинаем с исходной модели, например, $f(x) = \sum_{j=1}^{n} y_j / n$.
2. **Цикл по** $t = 1, ..., T$:
   - вычислить $-g_t(x_i)$
   - построить регрессию h_t по $-g_t(x_i)$
   - $f(x_i) \leftarrow f(x_i) + \rho_t \cdot h_t(x_i)$
3. **Конец цикла по** t

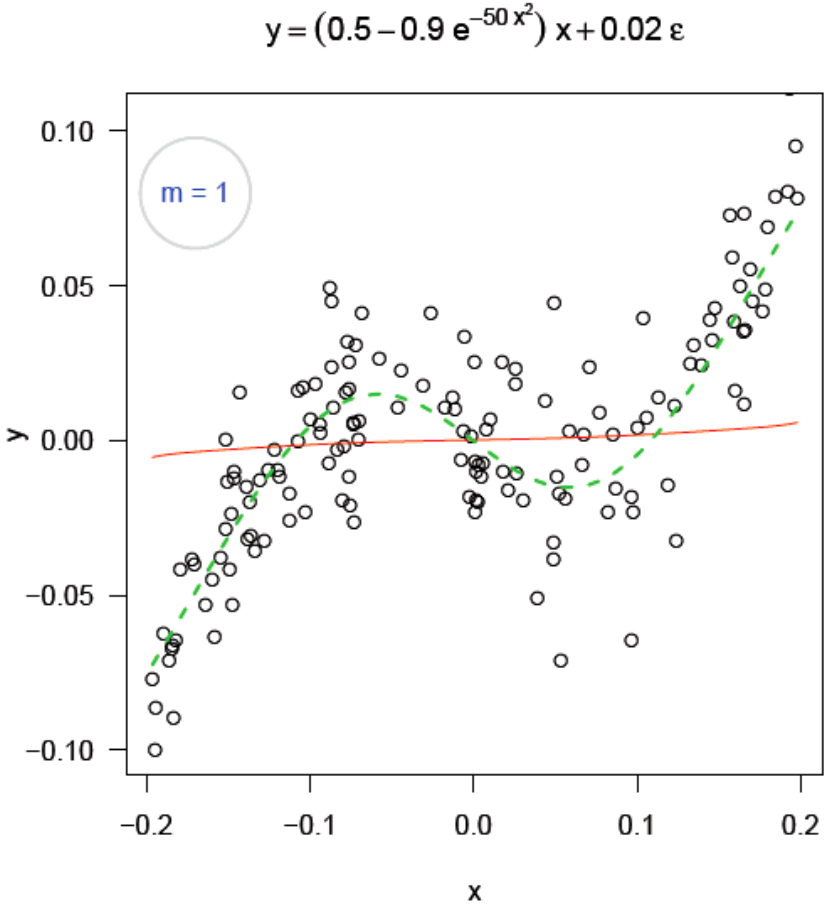
Преимущество формулировки алгоритма, используя градиент, в том, что можно рассматривать другие функции потерь.

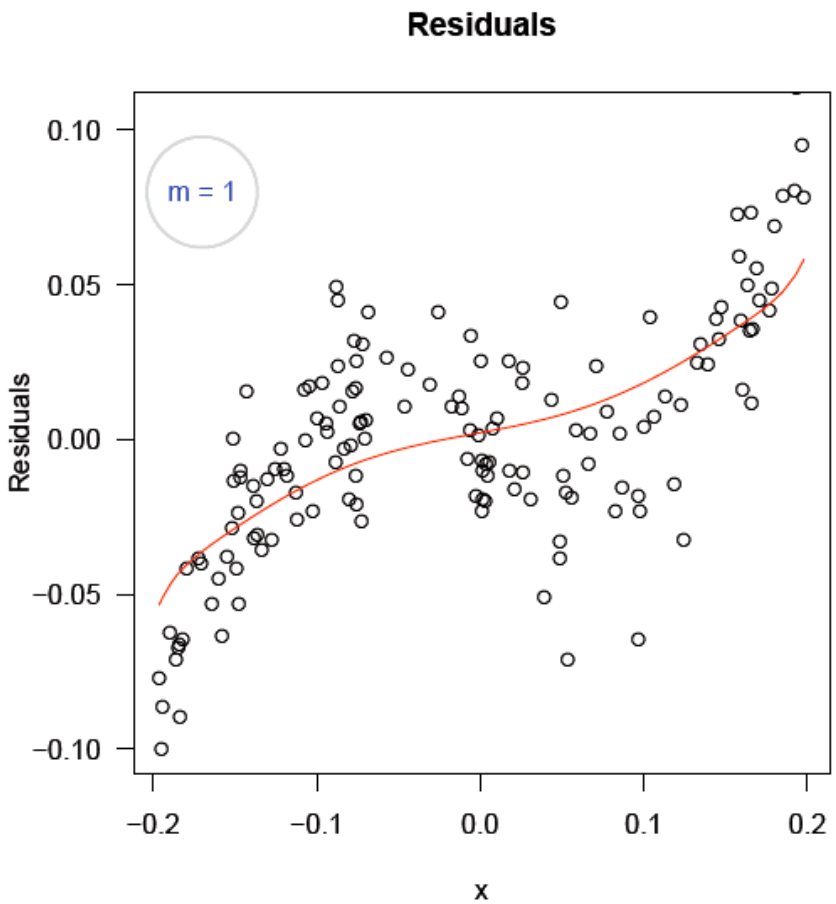
Пример градиентного бустинга



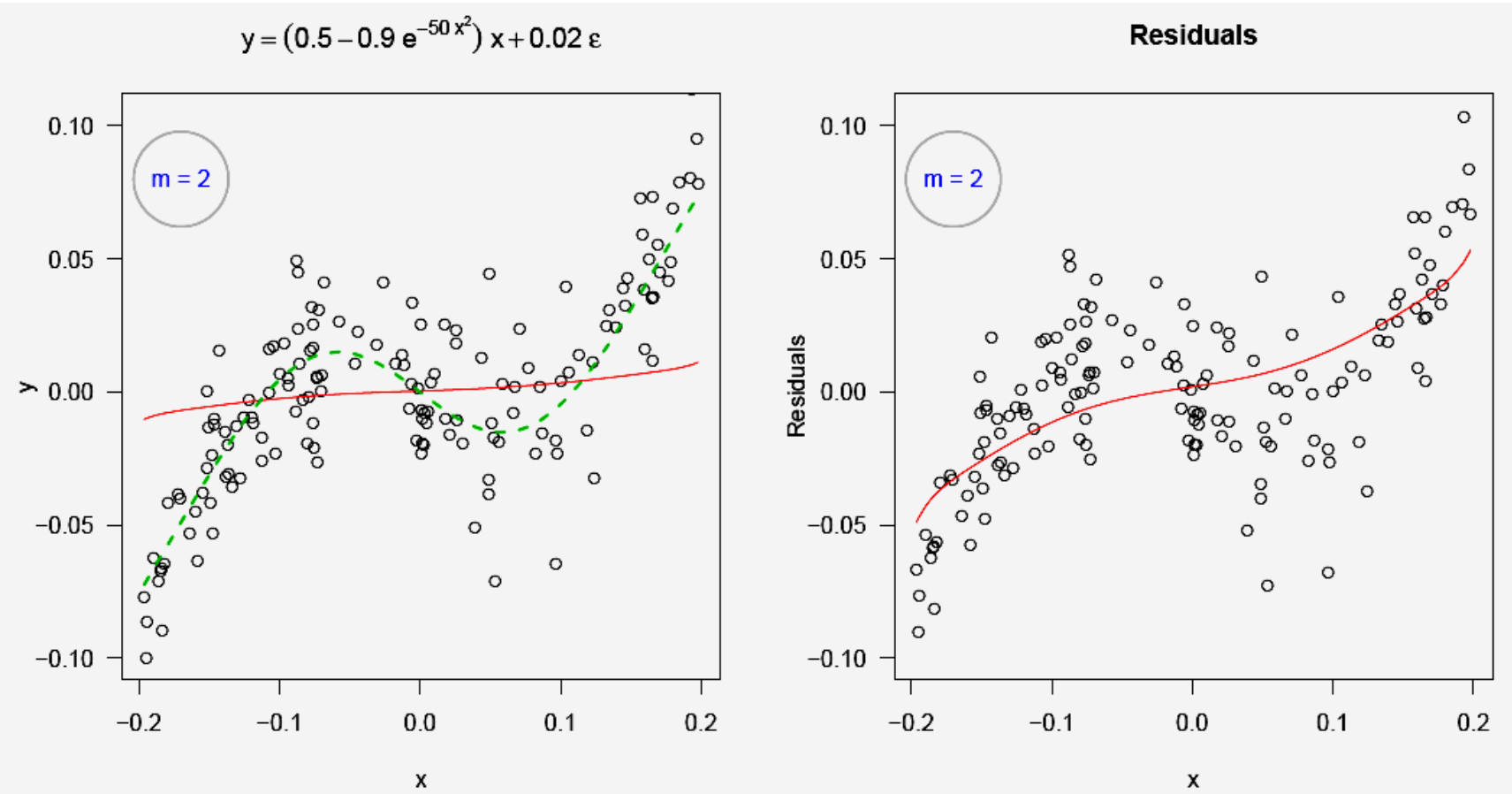


Пример градиентного бустинга

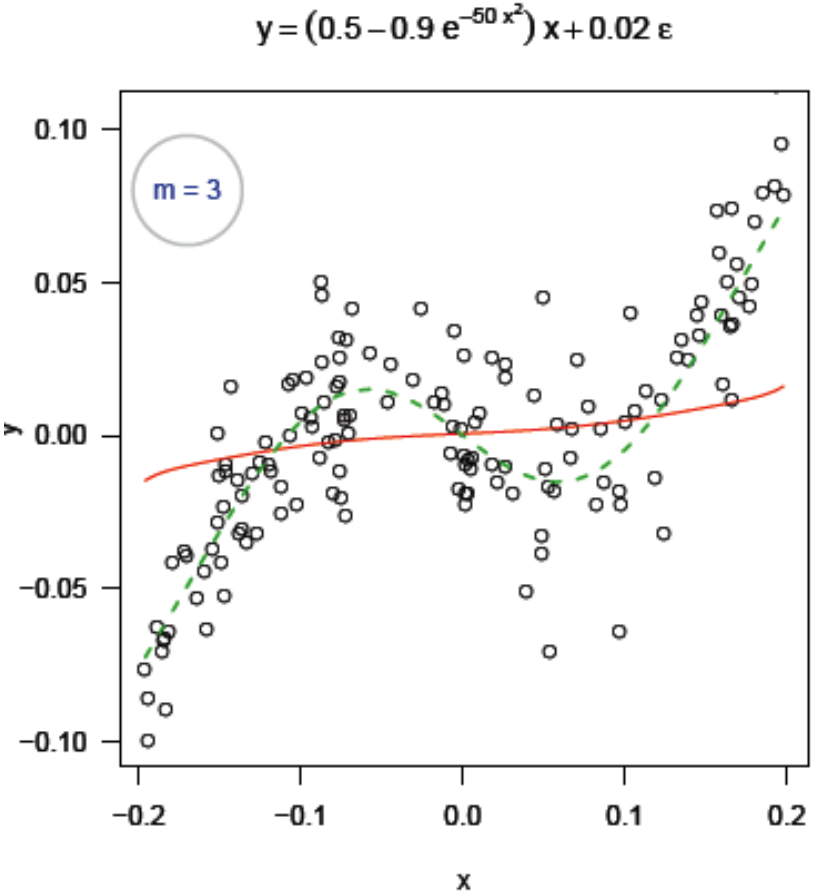


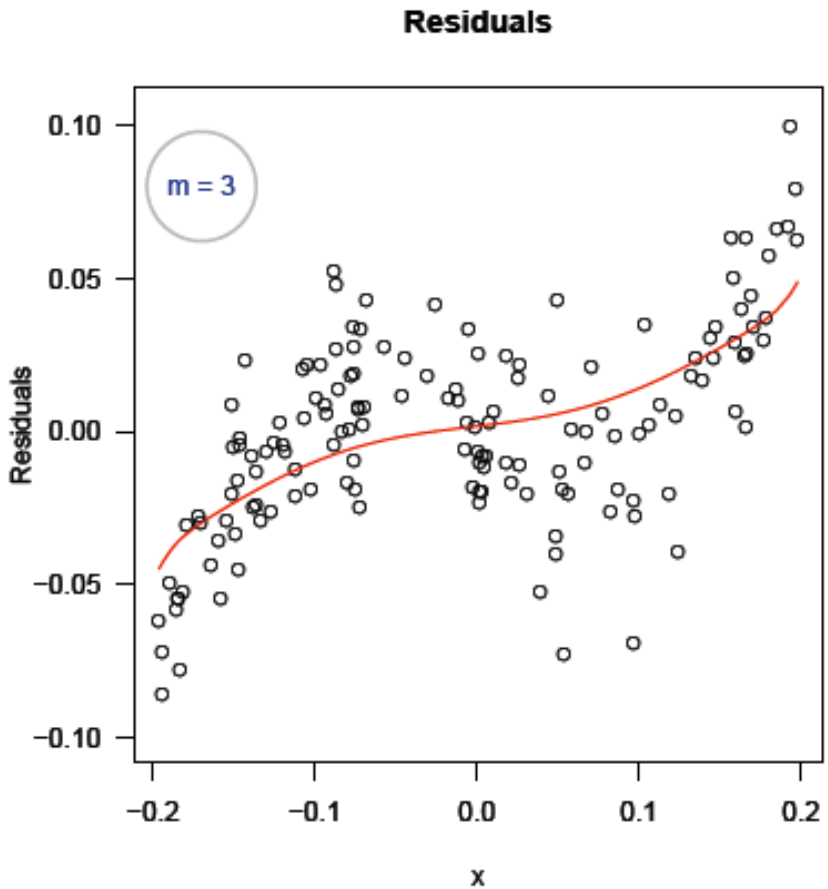


Пример градиентного бустинга

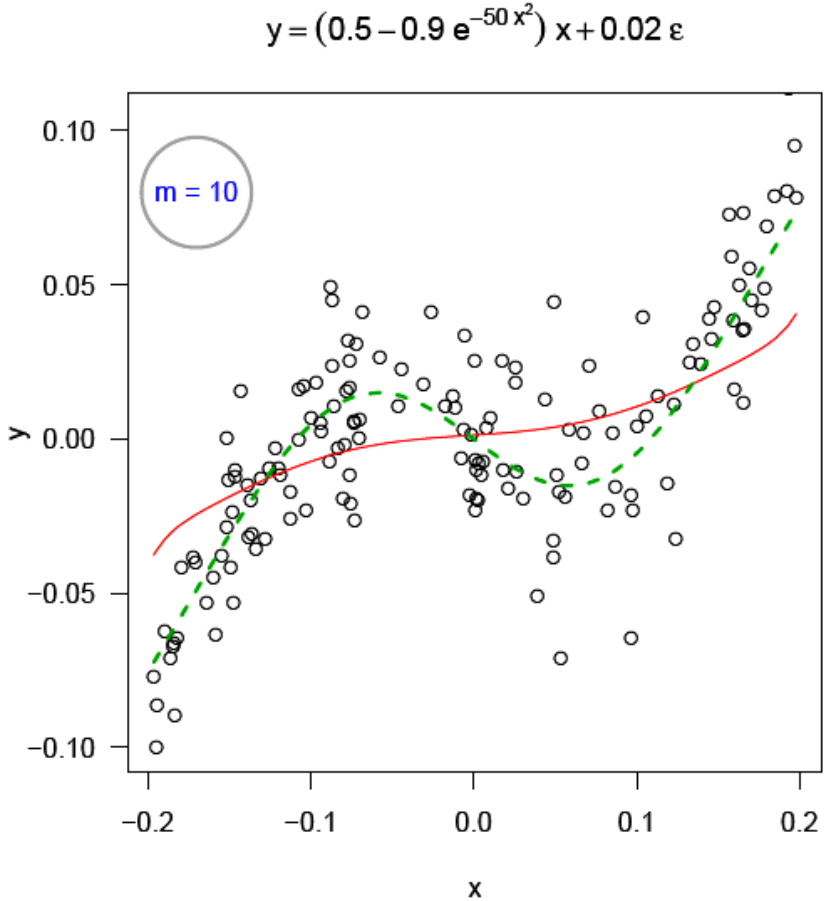


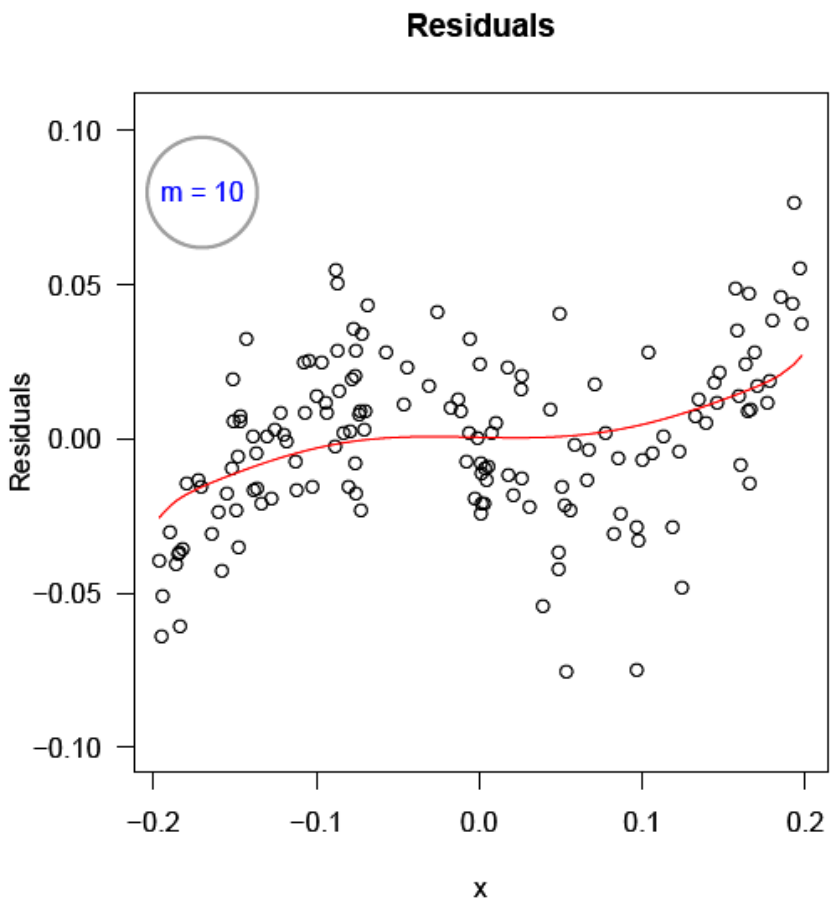
Пример градиентного бустинга



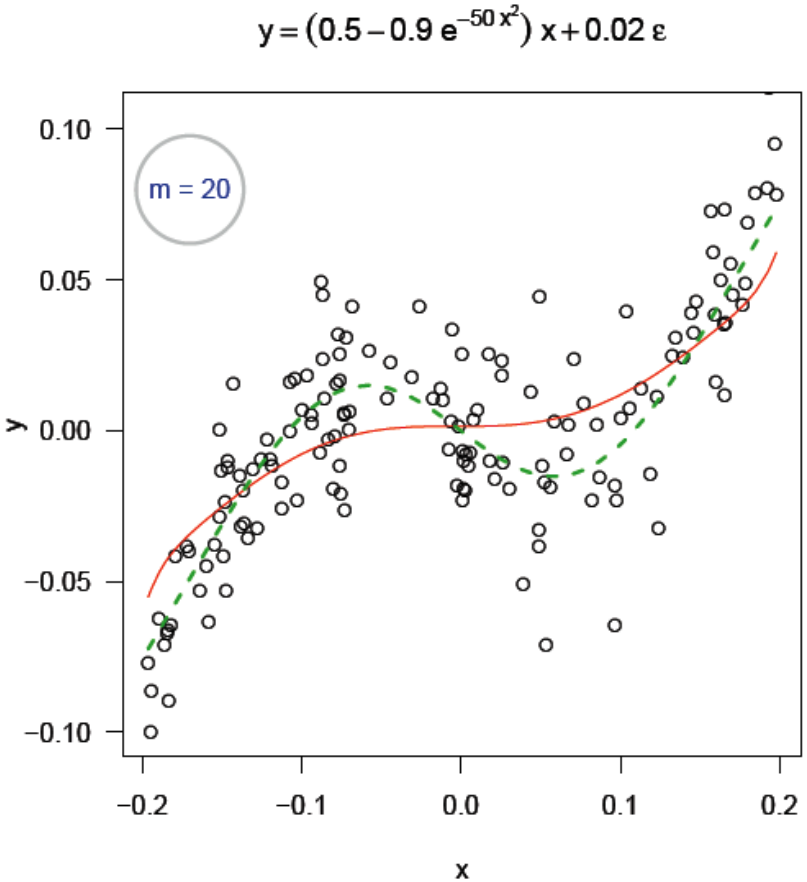


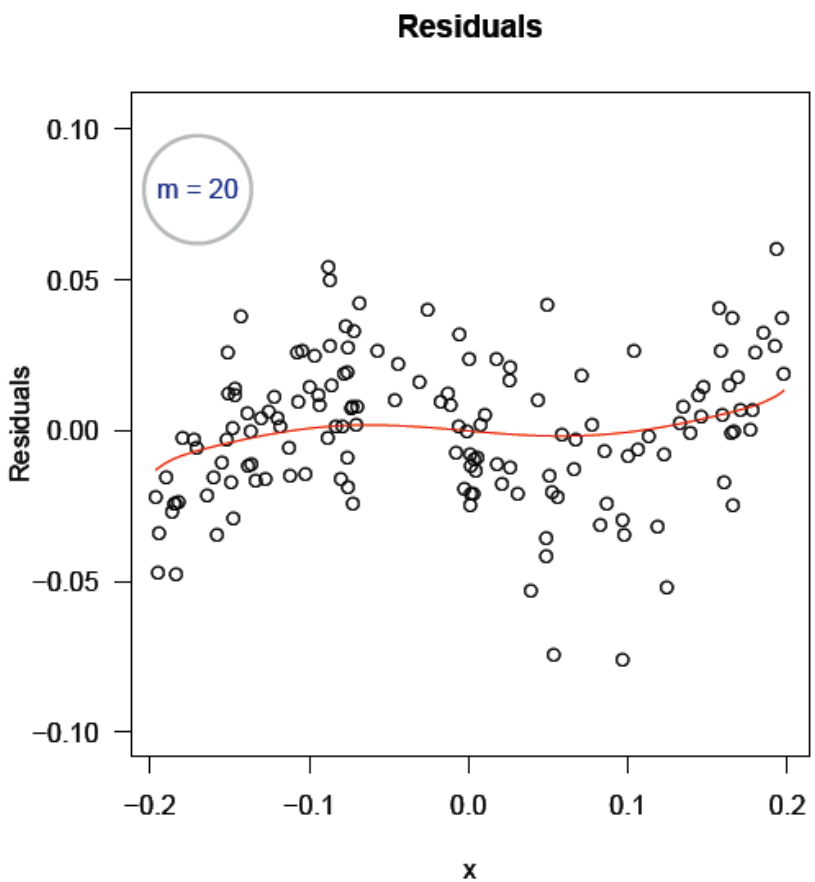
Пример градиентного бустинга



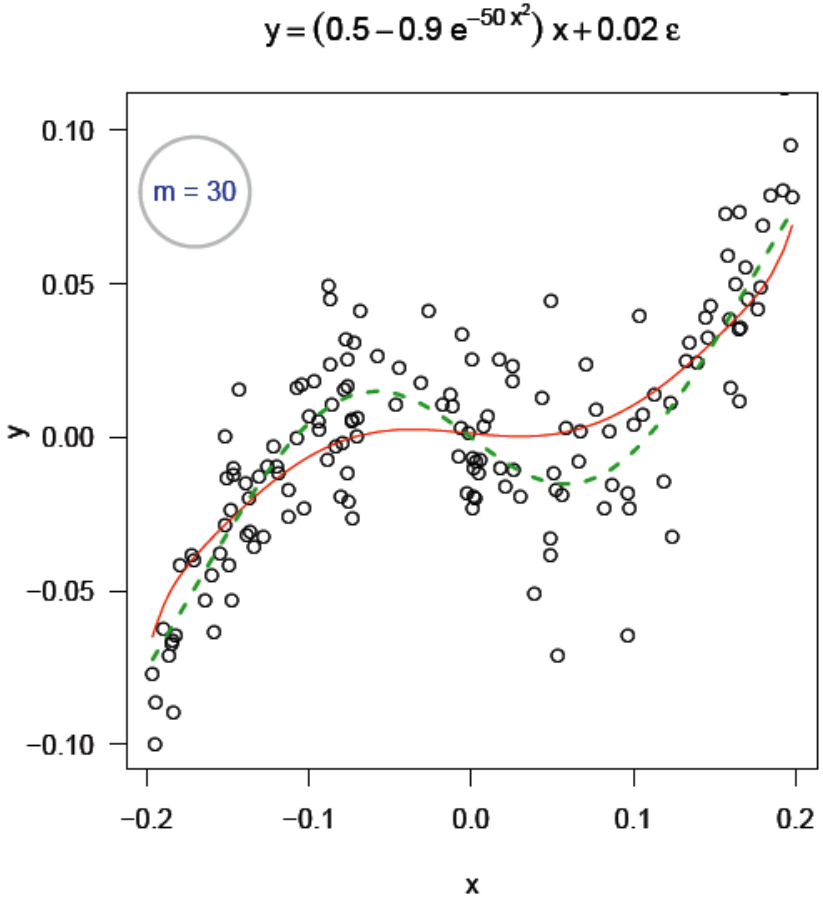


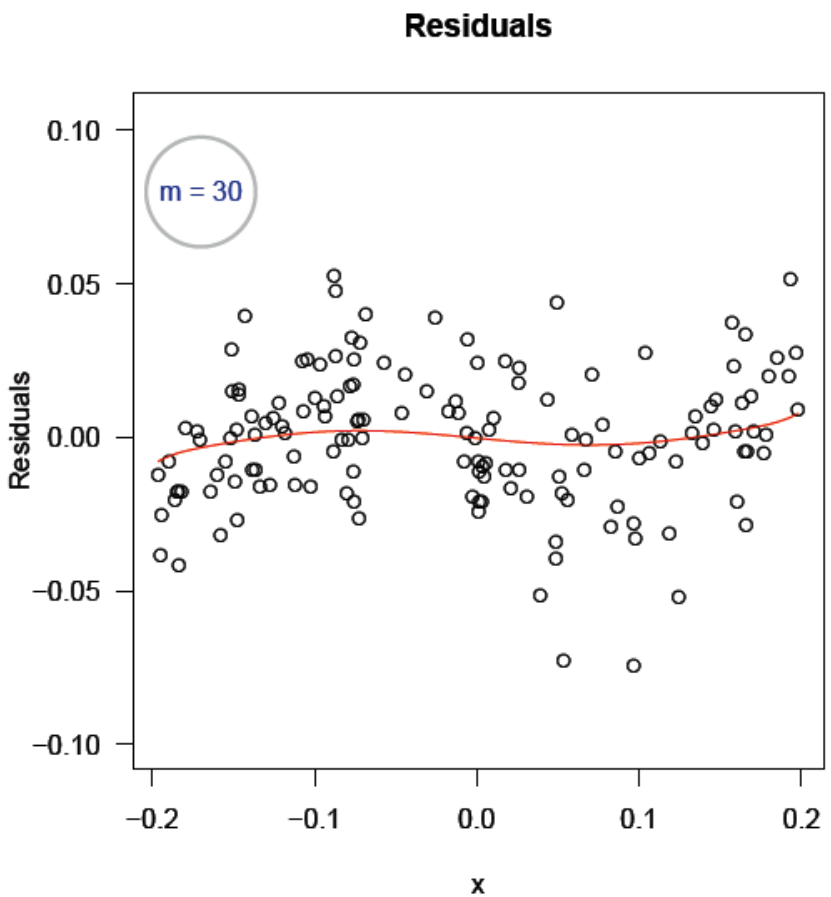
Пример градиентного бустинга



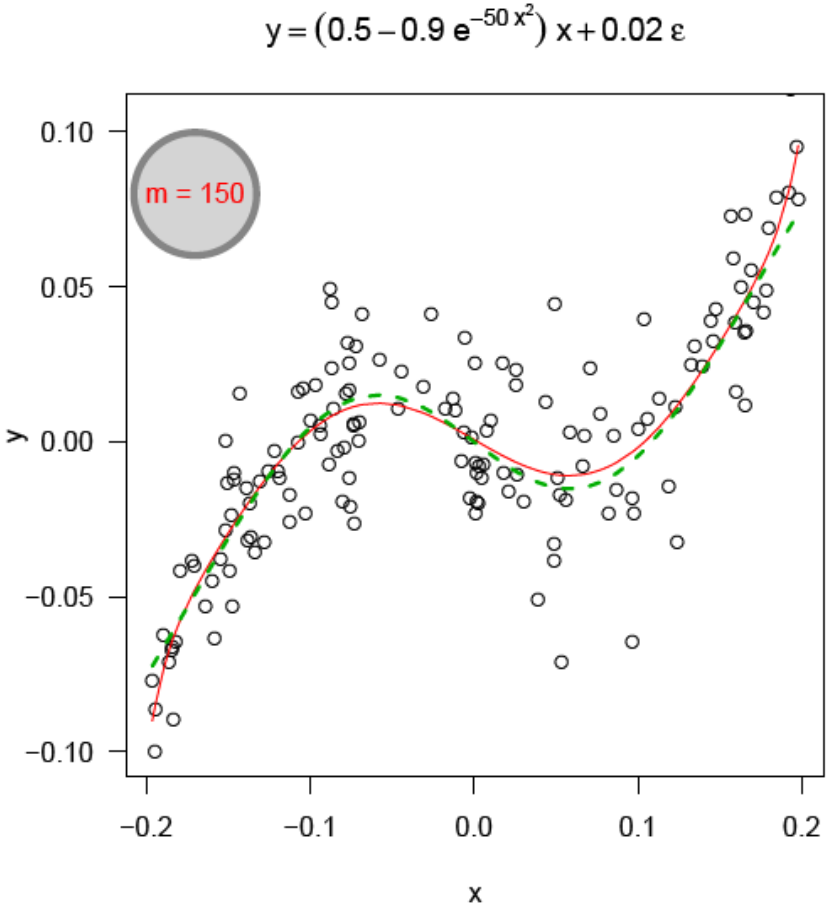


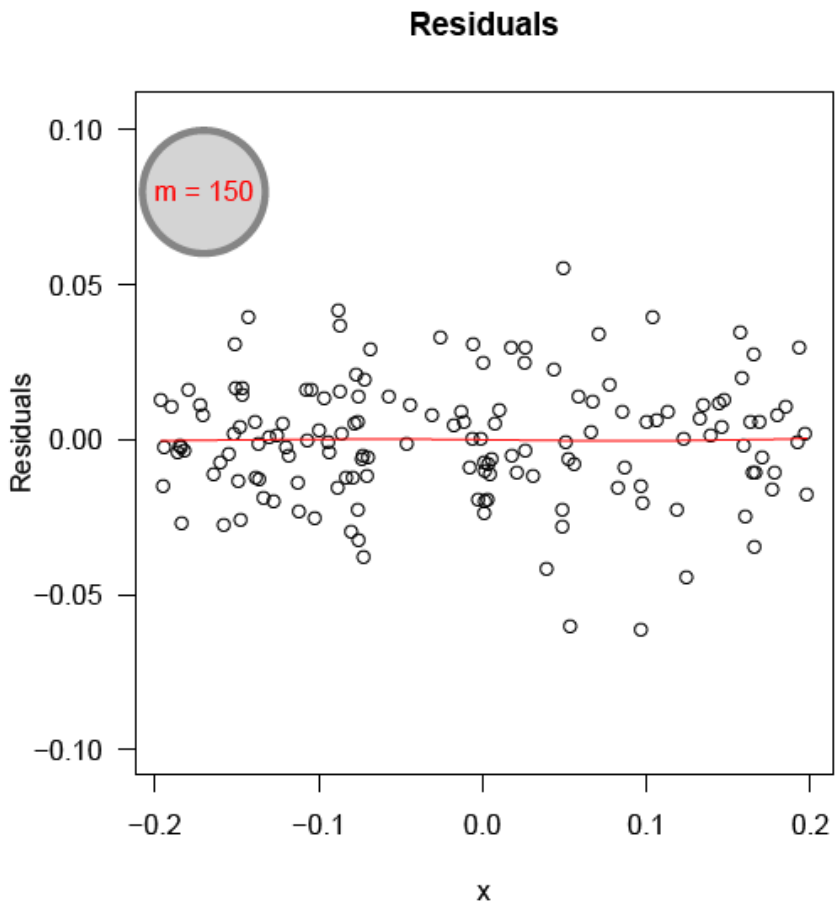
Пример градиентного бустинга





Пример градиентного бустинга





Особенности квадратичной функции потерь в градиентном бустинге

- Достаточно простая функция с математической точки зрения
- Не является робастной к выбросам, которые сильно влияют, так как величина ошибки - в квадрате

y_i	0.5	1.2	2	**5**
$f(x_i)$	0.6	1.4	1.5	**1.7**
$L = (y - f)^2/2$	0.005	0.02	0.125	**5.445**

Другие функции потерь (более робастные к выбросам)

- Абсолютные потери $L(y, f) = |y - f|$
- Функция потерь Хьюбера

$$L(y, f) = \begin{cases} (y - f)^2/2, & \text{если } |y - f| \leq \delta, \\ \delta\,(|y - f|) - \delta/2, & \text{если } |y - f| > \delta. \end{cases}$$

y_i	0.5	1.2	2	**5**
$f(x_i)$	0.6	1.4	1.5	**1.7**
Квадратич. потери	0.005	0.02	0.125	**5.445**
Абсолютные потери	0.1	0.2	0.5	**3.3**
Хьюбера потери ($\delta = 0.5$)	0.005	0.02	0.125	**1.525**

Алгоритм градиентного бустинга для регрессии с абсолютными потерями

Отрицательный градиент:

$$-g(x_i) = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = \text{sign}\,(y_i - f(x_i))$$

1. Начинаем с исходной модели, например, $f(x) = \sum_{j=1}^{n} y_j / n$.
2. **Цикл по** $t = 1, ..., T$:
   - вычислить $-g_t(x_i)$
   - построить регрессию h_t по $-g_t(x_i)$
   - $f(x_i) \leftarrow f(x_i) + \rho_t \cdot h_t(x_i)$
3. **Конец цикла по** t

Алгоритм градиентного бустинга для регрессии с потерями Хьюбера

Отрицательный градиент:

$$-g(x_i) = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} =$$

$$= \begin{cases} y_i - f(x_i), & \text{если } |y_i - f(x_i)| \leq \delta, \\ \delta \cdot \operatorname{sign}(y_i - f(x_i)), & \text{если } |y_i - f(x_i)| > \delta. \end{cases}$$

Градиентный бустинг для регрессии в общем виде

Используем любую дифференцируемую функцию потерь L

1. Начинаем с исходной модели, например, $f(x) = \sum_{j=1}^{n} y_j / n$.
2. **Цикл по** $t = 1, ..., T$:
   - вычислить $-g_t(x_i) = -\frac{\partial L(y_i, f_t(x_i))}{\partial f_t(x_i)}$
   - построить регрессию h_t по $-g_t(x_i)$
   - $f(x_i) \leftarrow f(x_i) + \rho_t \cdot h_t(x_i)$
3. **Конец цикла по** t
4. Результат $f(x) = \sum_{i=1}^{T} \rho_t h_t(x)$

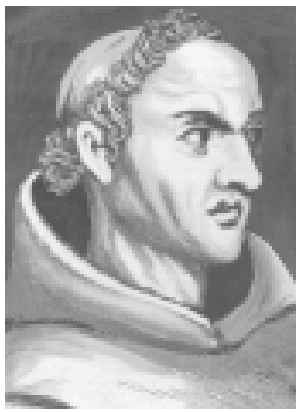
Градиентный бустинг для регрессии

Как выбрать подходящую скорость обучения ρ для алгоритма градиентного бустинга?
См. Friedman J.H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics, pages 1189–1232.

Достоинства градиентного бустинга

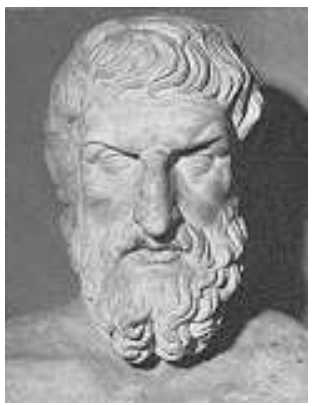
- Наиболее общий из всех бустингов
- Может использоваться произвольная функция потерь
- Подходит для регрессии, классификации и ранжирования

“Бритва Оккама” (Occam’s Razor)



- В общем философском понимании: “Не следует множить сущее без необходимости”.
- В машинном обучении: “Между двумя моделями, которые дают эквивалентные предсказания, следует выбрать более простую.”

“Принцип Эпикура” (Epicurus’ principle)



- Если более одной теорий согласуются с наблюдениями, то следует оставить все теории. (If more than one theory is consistent with the observations, keep all theories.)

Противоречие с бритвой Оккама

- Методы композиции противоречат бритве Оккама.
- Больше итераций $\Rightarrow$ больше базовых алгоритмов для голосования $\Rightarrow$ больше сложность
- При отсутствии ошибок обучения более сложный классификатор может быть хуже

Две бритвы (Domingos, 1999)

1. Первая бритва: Если даны две модели с одинаковыми ошибками тестирования, то простейшая из них предпочтительнее, так как простота лучше сама по себе.
2. Вторая бритва: Если даны две модели с одинаковыми ошибками обучения, то простейшая из них предпочтительнее, так как она имеет большие шансы иметь меньше ошибок тестирования.

Программная реализация в R

- https://cran.r-project.org/web/views/MachineLearning.html
- Package **gbm**, реализует стандартный AdaBoost и алгоритм градиентного бустинга Фридмана
- Package **mboost**, реализует алгоритмы градиентного бустинга
- Package **xgboost**, реализует алгоритмы градиентного бустинга
- Package **adabag**, реализует AdaBoost.M1
- Package **ada**, реализует стохастический градиентный бустинг (SGB)
- Package **randomForest**, реализует алгоритмы случайных лесов

Вопросы

?