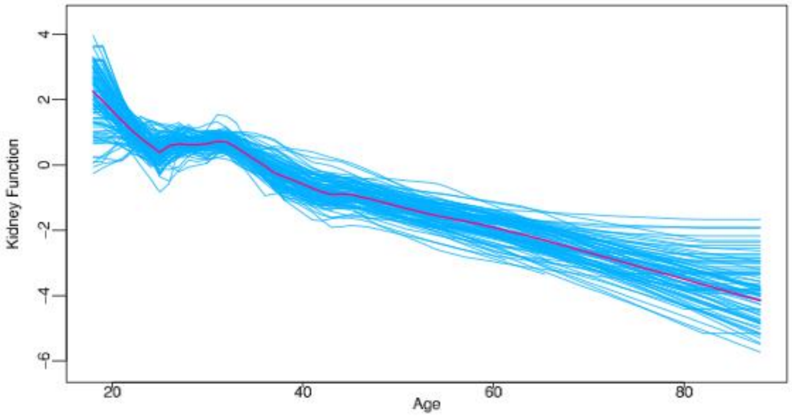
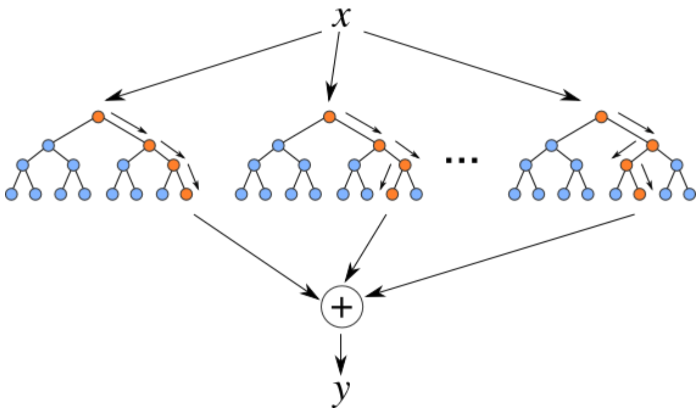




Анализ данных на практике

Композиции алгоритмов

Виктор Кантор

Бутстреп

Выборка из некоторого распределения:

№	значение
1	
2	
3	
N	

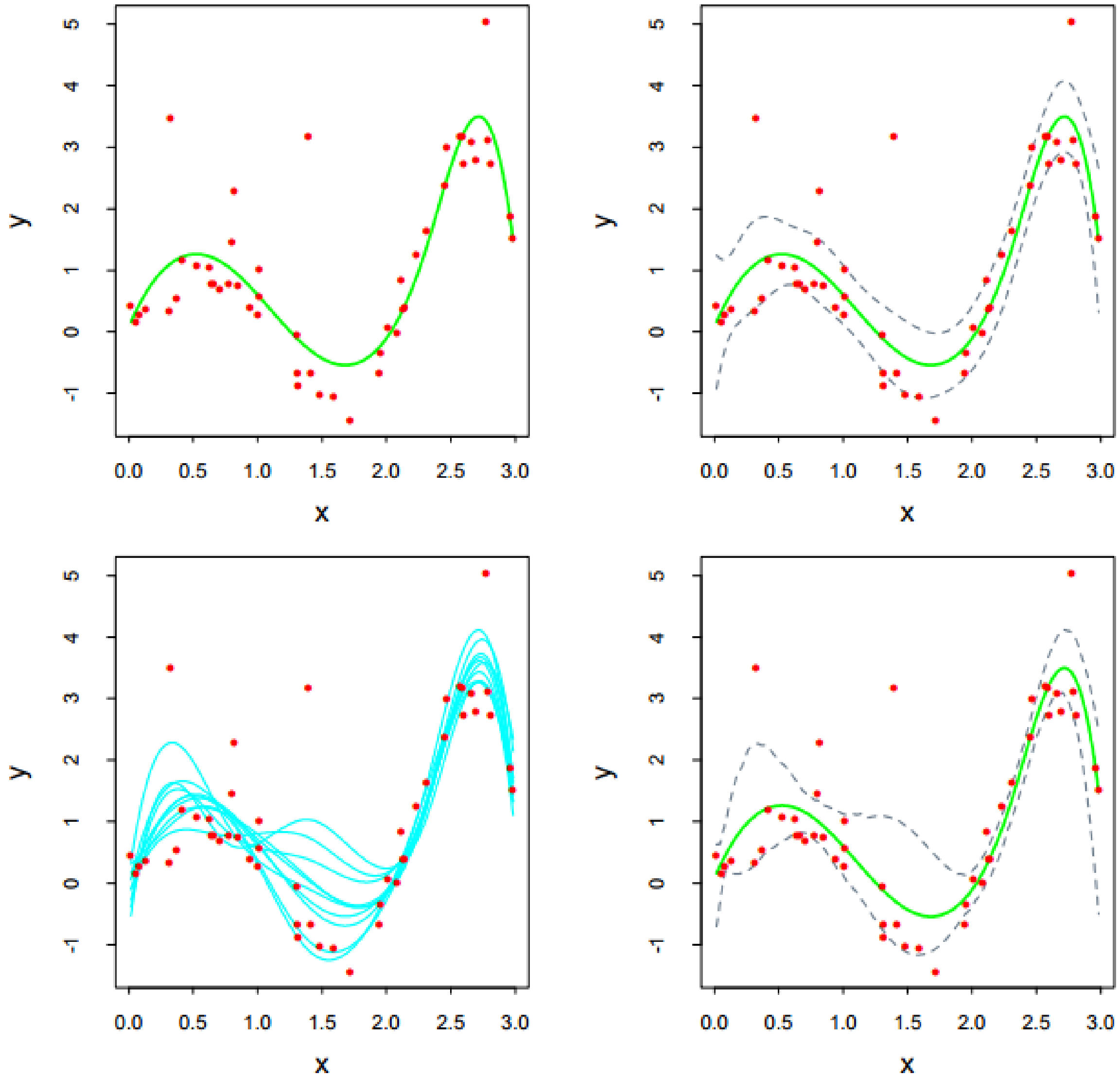
Хотим вычислить какую-то величину X по данным наблюдениями.

Было бы здорово вычислить X на многих выборках из распределения, а потом усреднить, но их у нас нет

Решение:

1. Выбираем наугад одно наблюдение из имеющихся.
2. Повторяем пункт 1 столько раз, сколько у нас есть наблюдений. При этом некоторые из них мы выберем несколько раз
3. Считаем интересующие нас величины по новой выборке. Запоминаем результат.
4. Повторяем пункты 1-3 много раз и усредняем

Бутстреп


y
5 4 3 2 1 0 -1
0.0 0.5 1.0 1.5 2.0 2.5 3.0
x
y
5 4 3 2 1 0 -1
0.0 0.5 1.0 1.5 2.0 2.5 3.0
x
y
5 4 3 2 1 0 -1
0.0 0.5 1.0 1.5 2.0 2.5 3.0
x
y
5 4 3 2 1 0 -1
0.0 0.5 1.0 1.5 2.0 2.5 3.0
x

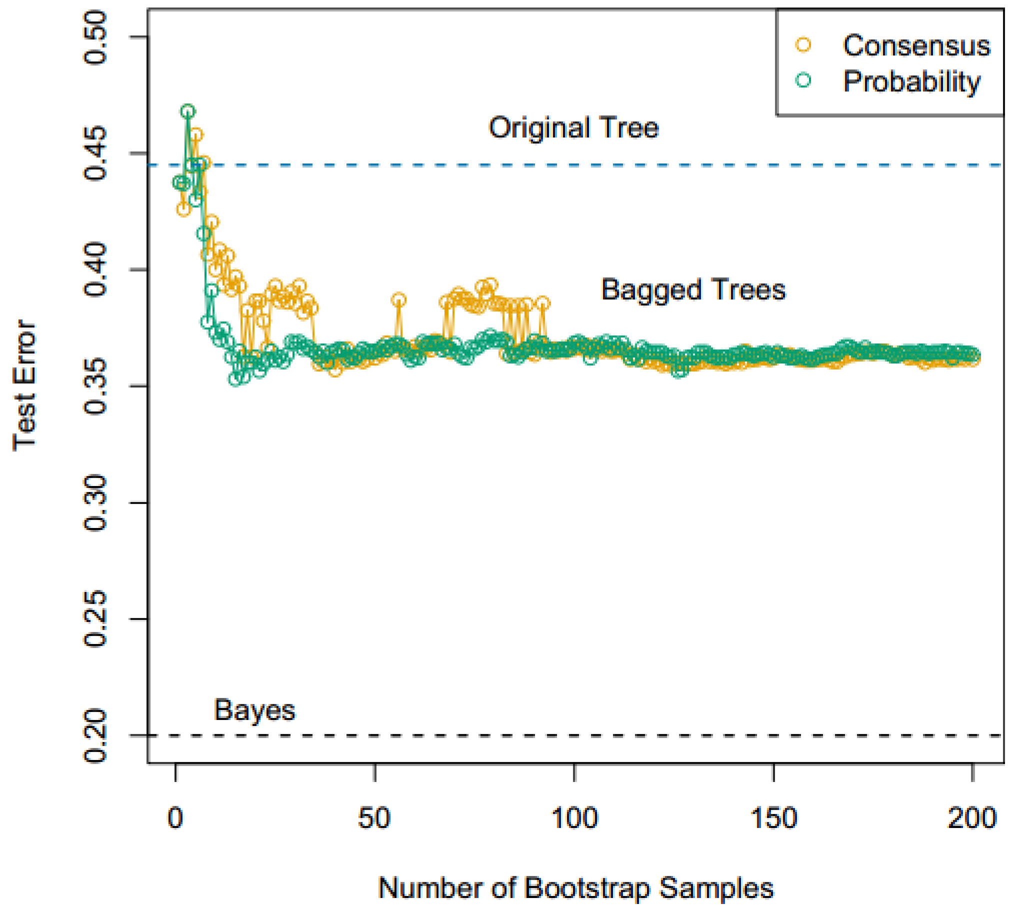
Обзор методов построения композиций

Bagging

Bagging = Bootstrap aggregation

По схеме выбора с возвращением, генерируем M обучающих выборок такого же размера, обучаем на них классификаторы и усредняем

Бэггинг в классификации



Вариации: Pasting, RSM

- RSM – Random Subspace Method, выбираем не объекты, а признаки
- Pasting – выбираем объекты без возвращения

Stacking

Обучающая выборка:

F1	F2	F3							FN

A → Обучающая выборка для M базовых алгоритмов

B → Обучающая выборка для линейной регрессии

Обучаем M базовых алгоритмов на выборке A

Считаем их прогнозы на выборке B

B1	B2			BM

Обучаем линейную регрессию с $w_0 = 0$

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

- Очень прост идейно, хорошо работает, логичен
- Иногда надо перебирать веса или использовать дискретную оптимизацию
- Не всегда композиция в виде взвешенной суммы – то, что надо. Иногда нужна более сложная архитектура классификации

Boosting

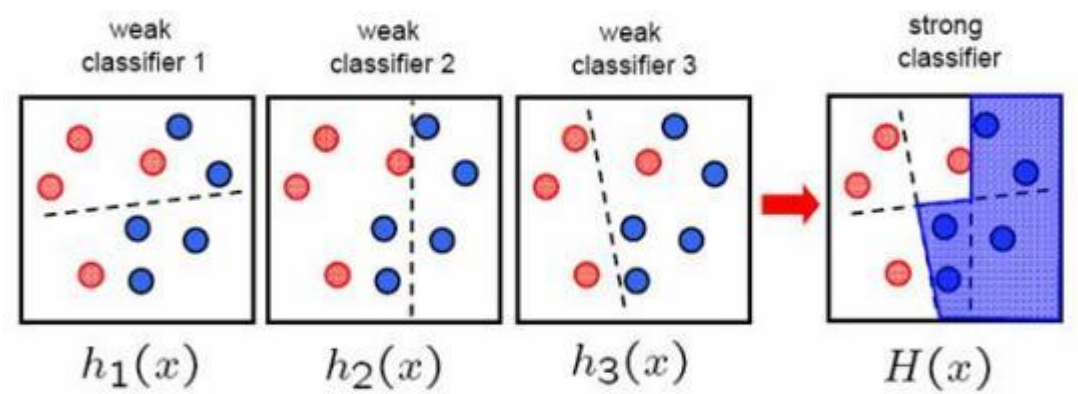
Бустинг – жадное построение взвешенной суммы слабых алгоритмов $b_t(x)$ – как правило, решающих деревьев небольшой глубины или линейных классификаторов.

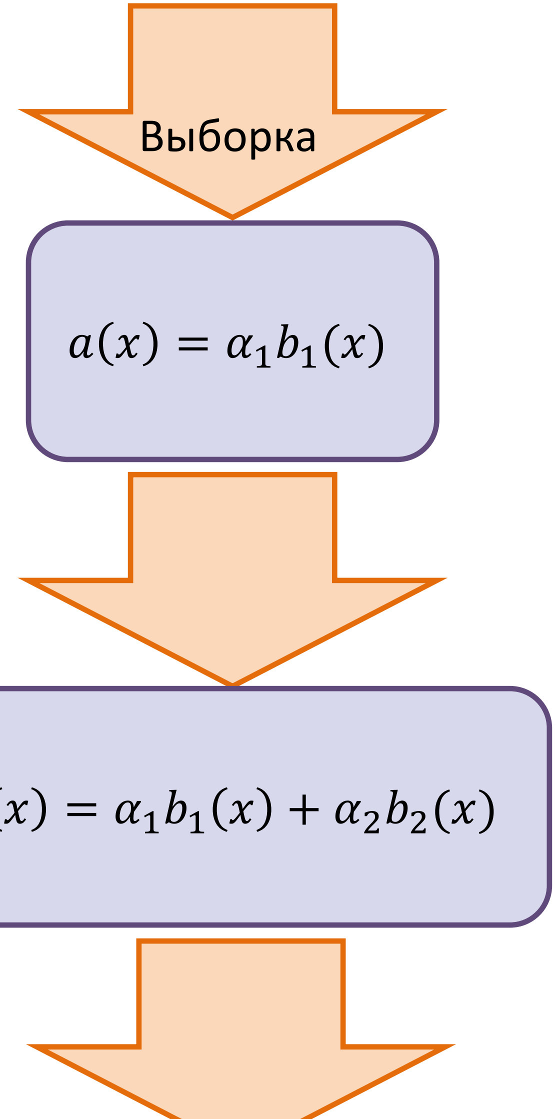
Регрессия:

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

Бинарная классификация:

$$a(x) = sign \sum_{t=1}^{T} \alpha_t b_t(x)$$

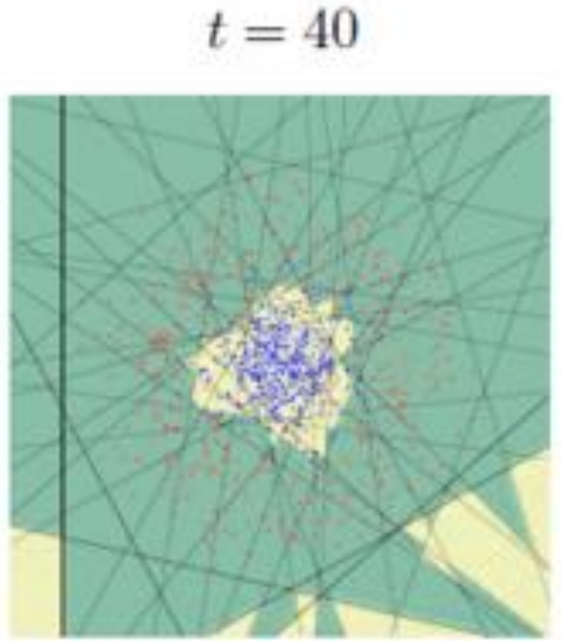


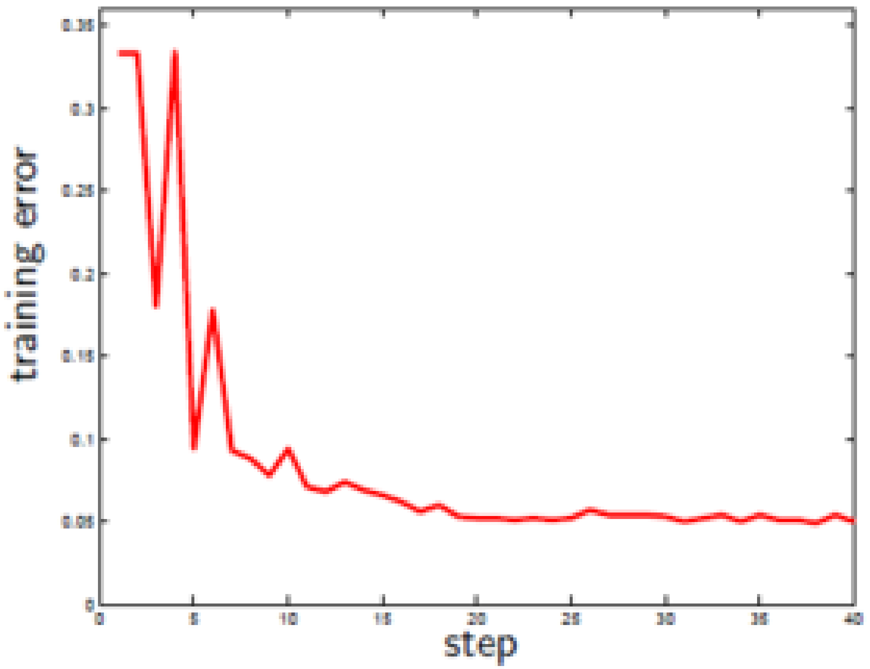


Алгоритмы бустинга

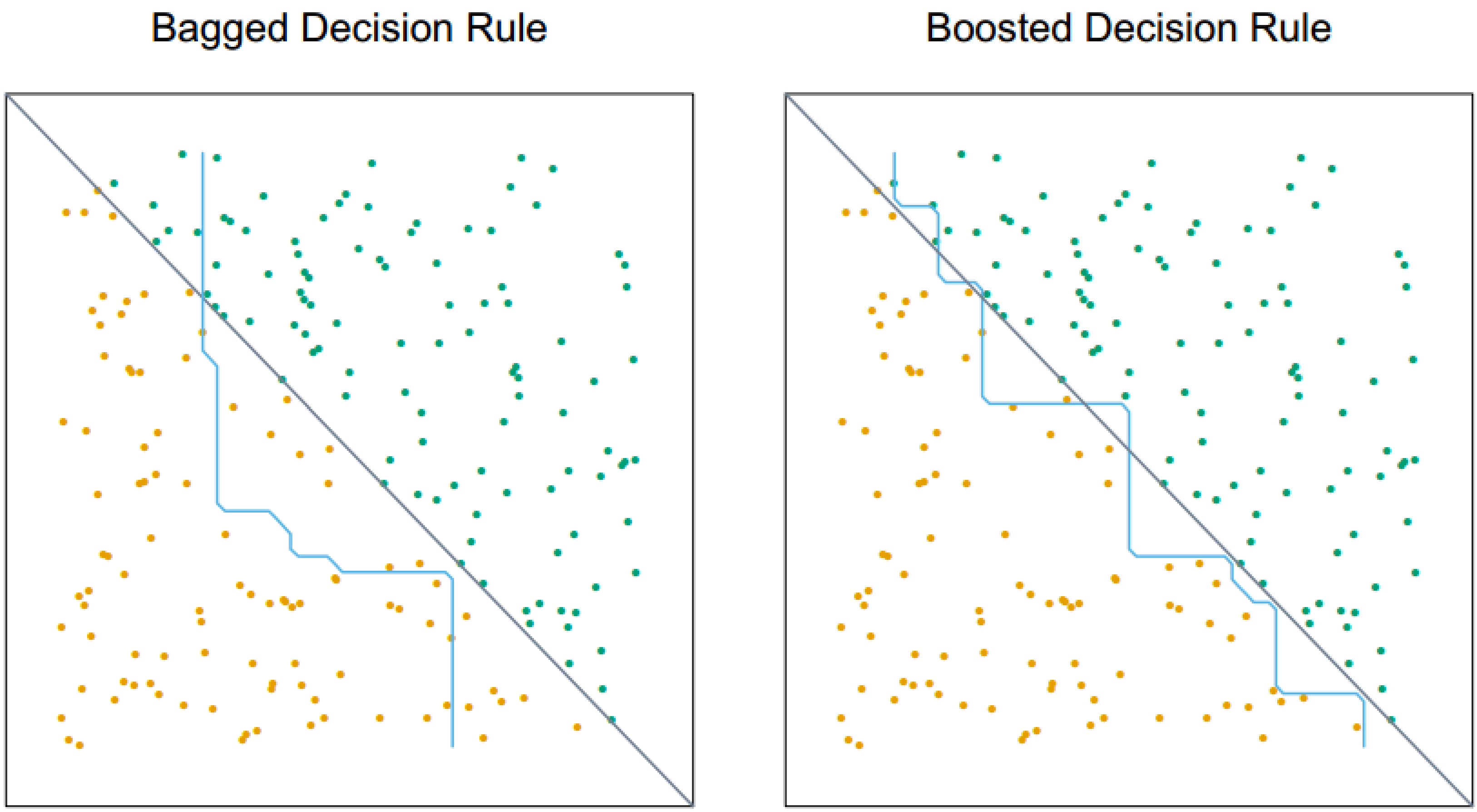
- Бустинг != AdaBoost
- Основные алгоритмы:
  - Градиентный бустинг
  - Адаптивный бустинг (AdaBoost)
- Вариации:
  - AnyBoost (произвольная функция потерь)
  - BrownBoost
  - GentleBoost
  - LogitBoost
  - ….

Бустинг над линейными классификаторами

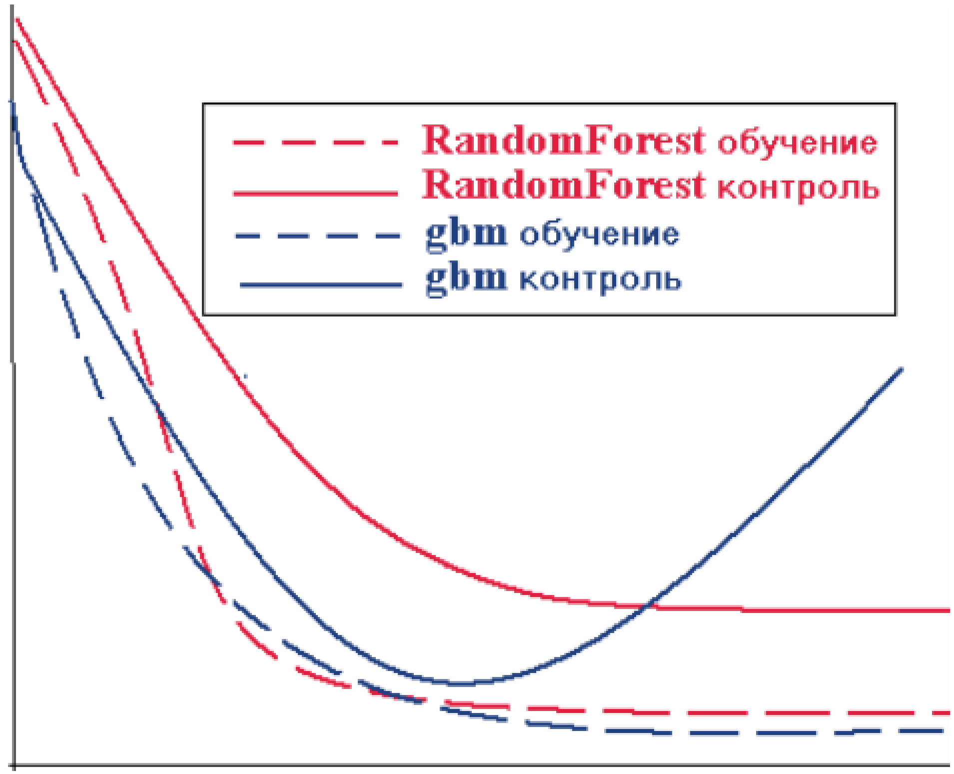




Бэггинг и бустинг



Бэггинг и бустинг: переобучение



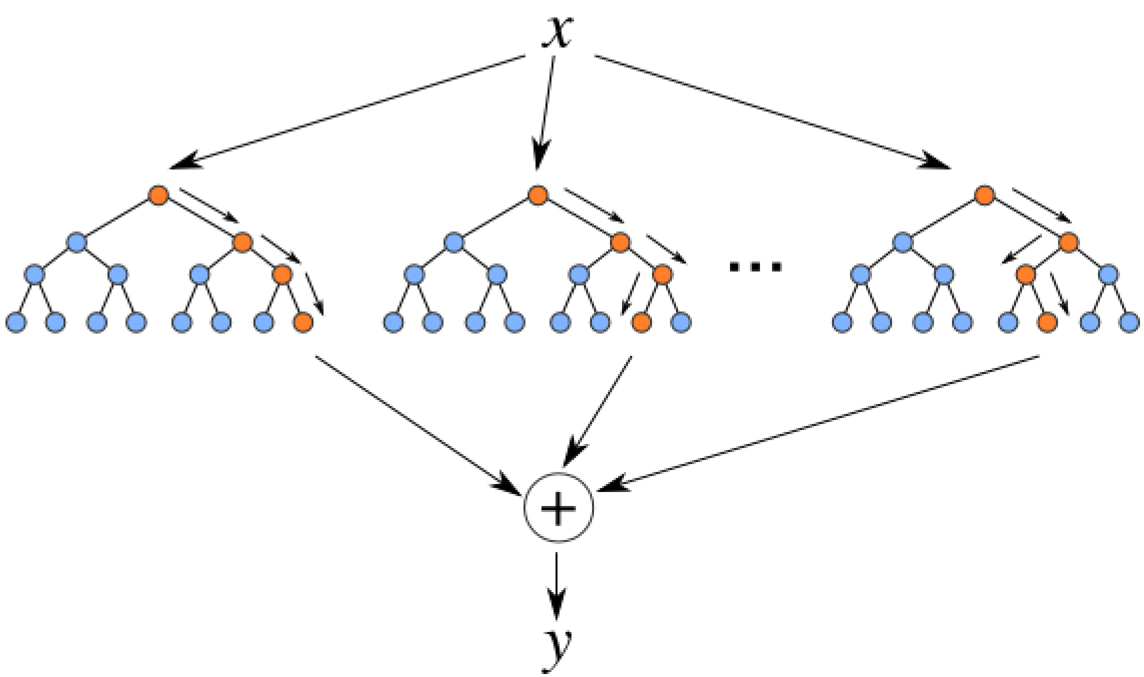
Преимущества и недостатки бустинга

- Позволяет очень точно приблизить восстанавливаемую функцию или разделяющую поверхность классов
- Плохо интерпретируем
- Композиции могут содержать десятки тысяч слабых классификаторов и долго обучаться
- Переобучение на выбросах при избыточном количестве классификаторов

Часто используемые алгоритмы

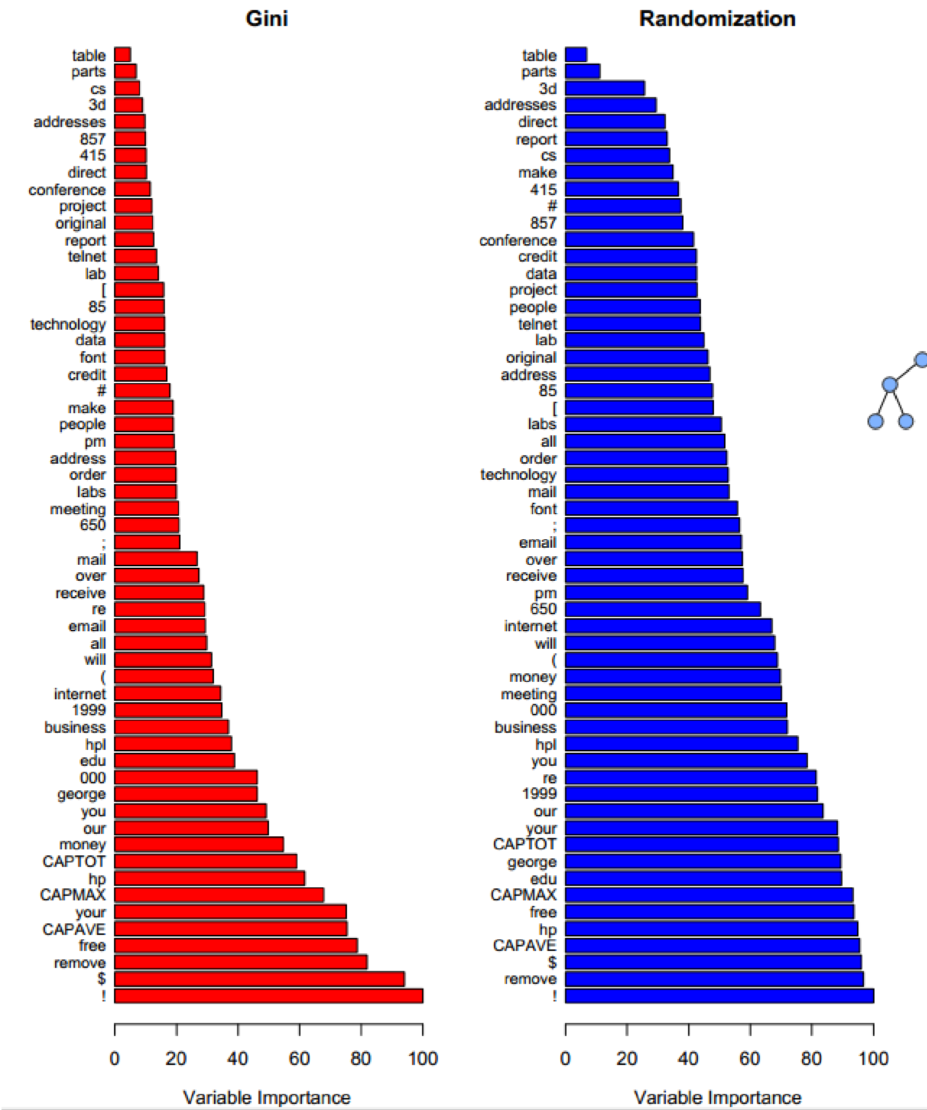
Леса решающих деревьев

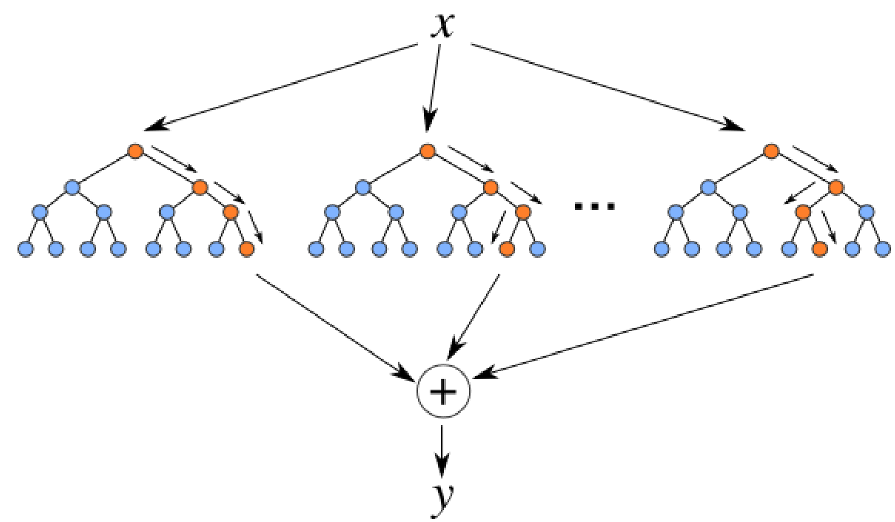
Random Forest



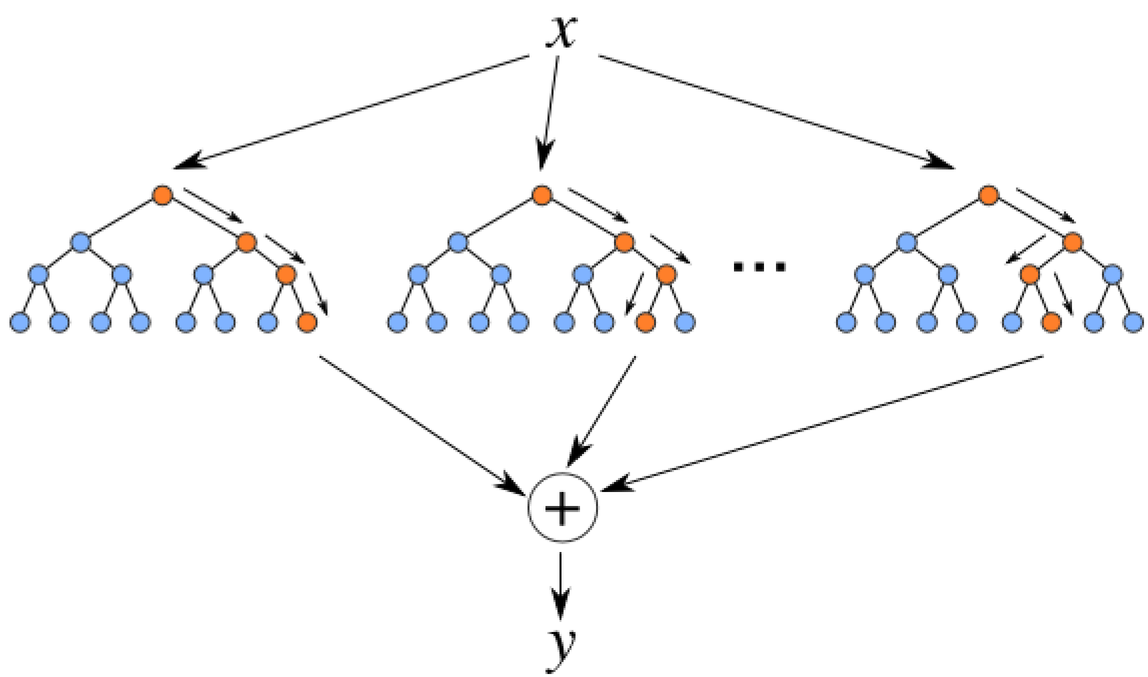
1. Бэггинг над деревьями
2. Рандомизированные разбиения в деревьях: выбираем k случайных признаков и ищем наиболее информативное разбиение по ним

Отбор признаков с помощью леса





Extreemly Randomized Trees

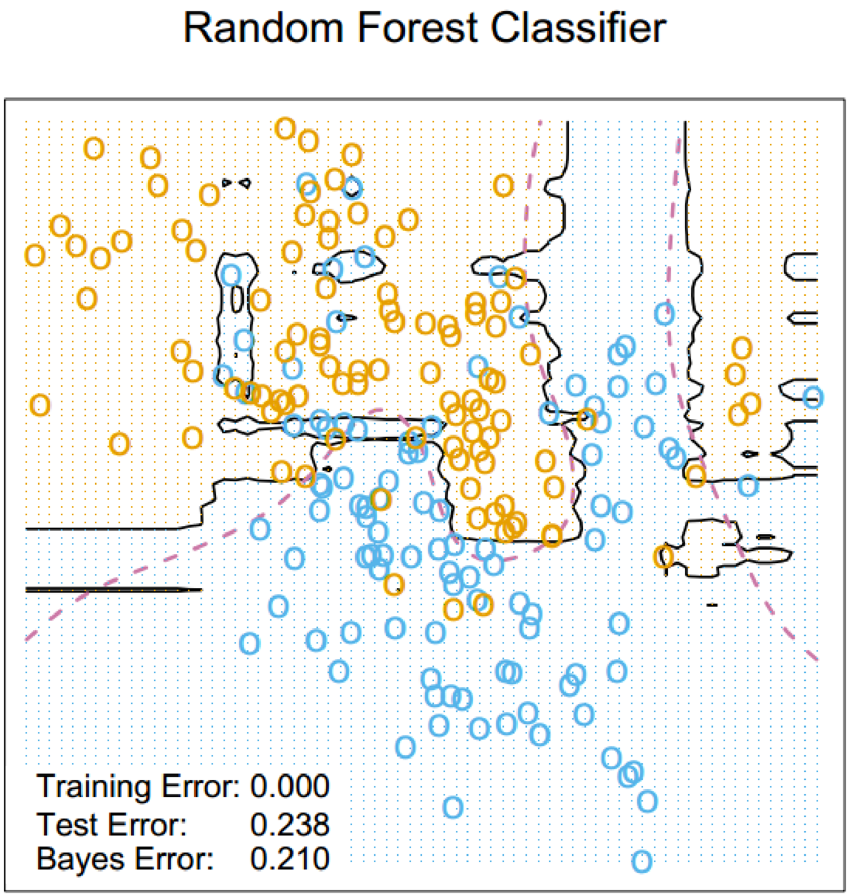


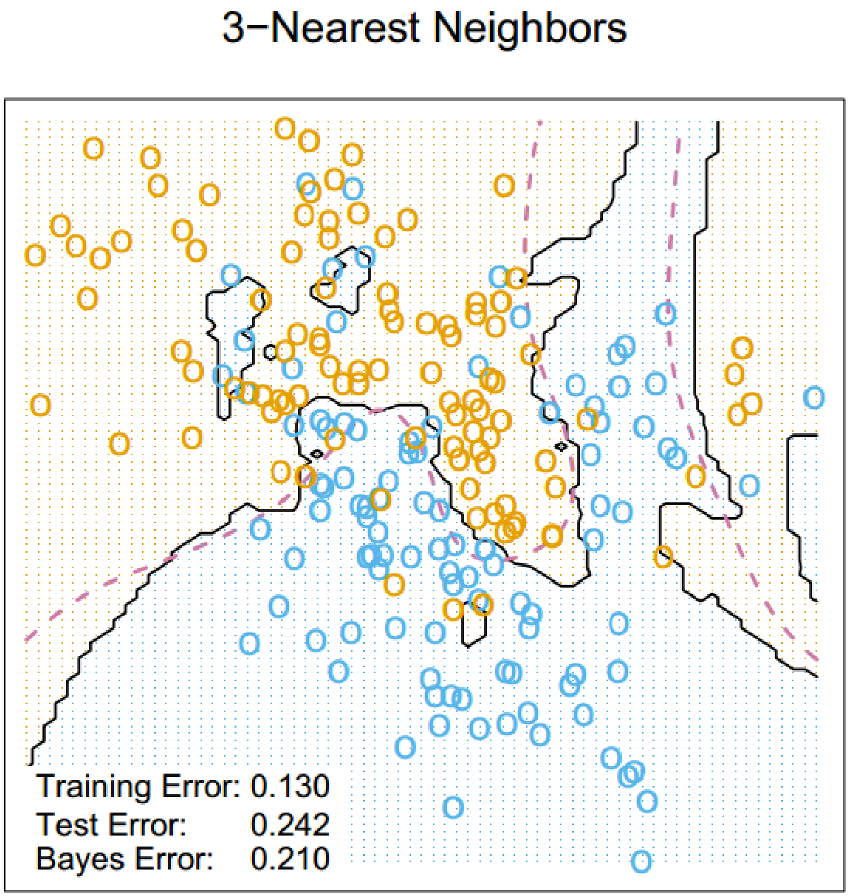
1. Бэггинг над «сильно рандомизированными» деревьями
2. При разбиении в дереве выбираем k случайных признаков и случайные пороги по ним, затем ищем наиболее информативное из этих разбиений

Нестандартные применения деревьев

- Метрика и поиск похожих объектов
- Преобразование признаков

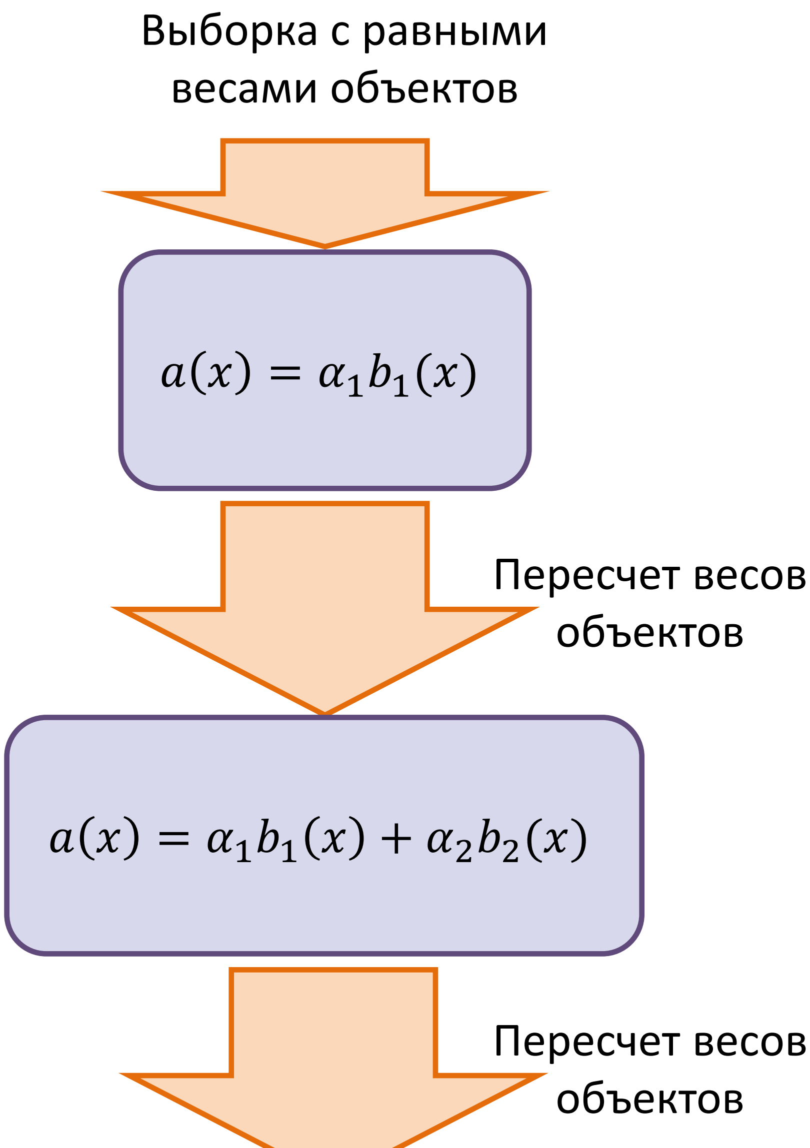
Может ли *** работать лучше RF





AdaBoost

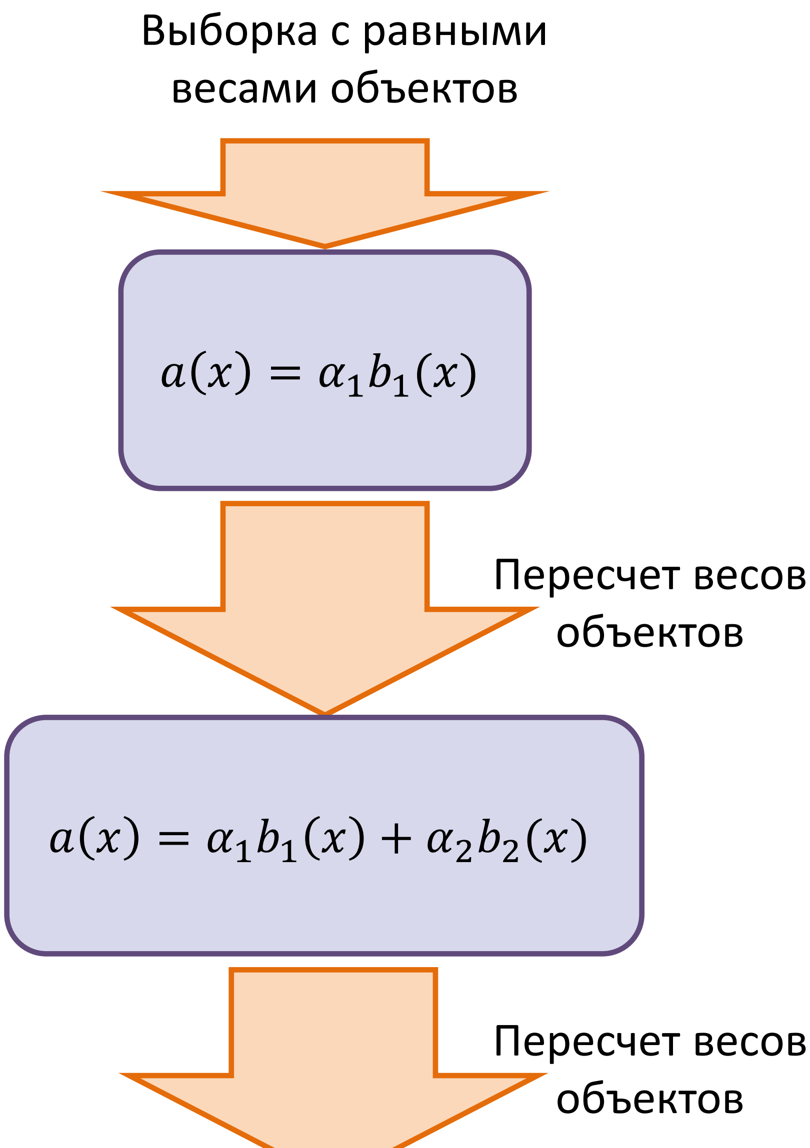
Идея AdaBoost



Ошибка нового алгоритма $\mathrm{b_t}(x)$:

$$\sum_{i=1}^{l} w_i L(x_i, b_t(x_i)) \to min$$

Идея AdaBoost



Ошибка нового алгоритма $\mathrm{b_t}(x)$:

$$\sum_{i=1}^{l} w_i L(x_i, b_t(x_i)) \to min$$

<u>Пересчет весов:</u>

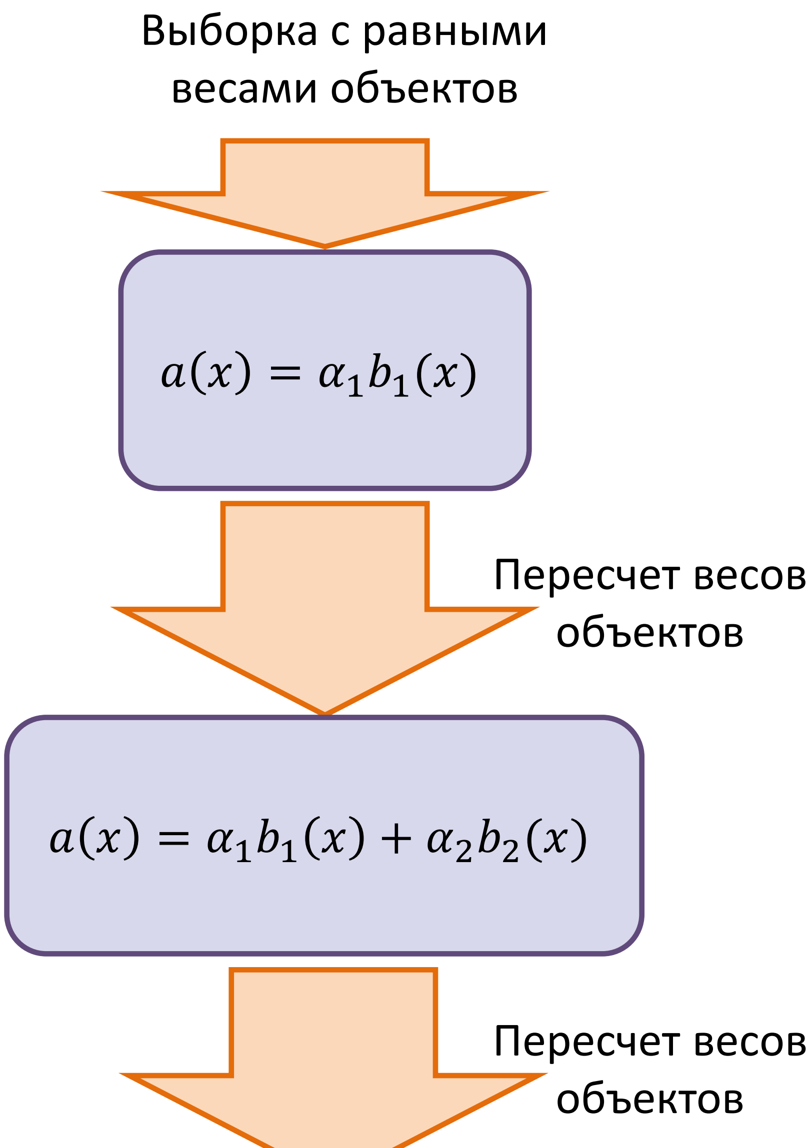
Если $\mathrm{b_t}(x_i) \neq y_i$:

увеличить вес w_i объекта x_i

Иначе:

уменьшить вес w_i объекта x_i

Идея AdaBoost



Ошибка нового алгоритма $\mathrm{b_t}(x)$:

$$\sum_{i=1}^{l} w_i L(x_i, b_t(x_i)) \to min$$

<u>Пересчет весов:</u>

Если $\mathrm{b_t}(x_i) \neq y_i$:

увеличить вес w_i объекта x_i

Иначе:

уменьшить вес w_i объекта x_i

Мы рассмотрим случай классов +1 и -1.
Отказ от классификации будем обозначать нулем.

Алгоритм AdaBoost

$$P(b_T;W^l) = \sum_{i=1}^{l} w_i[b_T(x_i) = y_i] \qquad N(b_T;W^l) = \sum_{i=1}^{l} w_i[b_T(x_i) = -y_i]$$

Алгоритм 1.1. AdaBoost — построение линейной комбинации классификаторов

Вход:

X^ℓ, Y^ℓ — обучающая выборка; T — максимальное число базовых алгоритмов;

Выход:

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

1: инициализация весов объектов: $w_i := 1 \qquad i = 1, \dots, \ell$;

2: **для всех** $t = 1, \dots, T$, пока не выполнен критерий останова

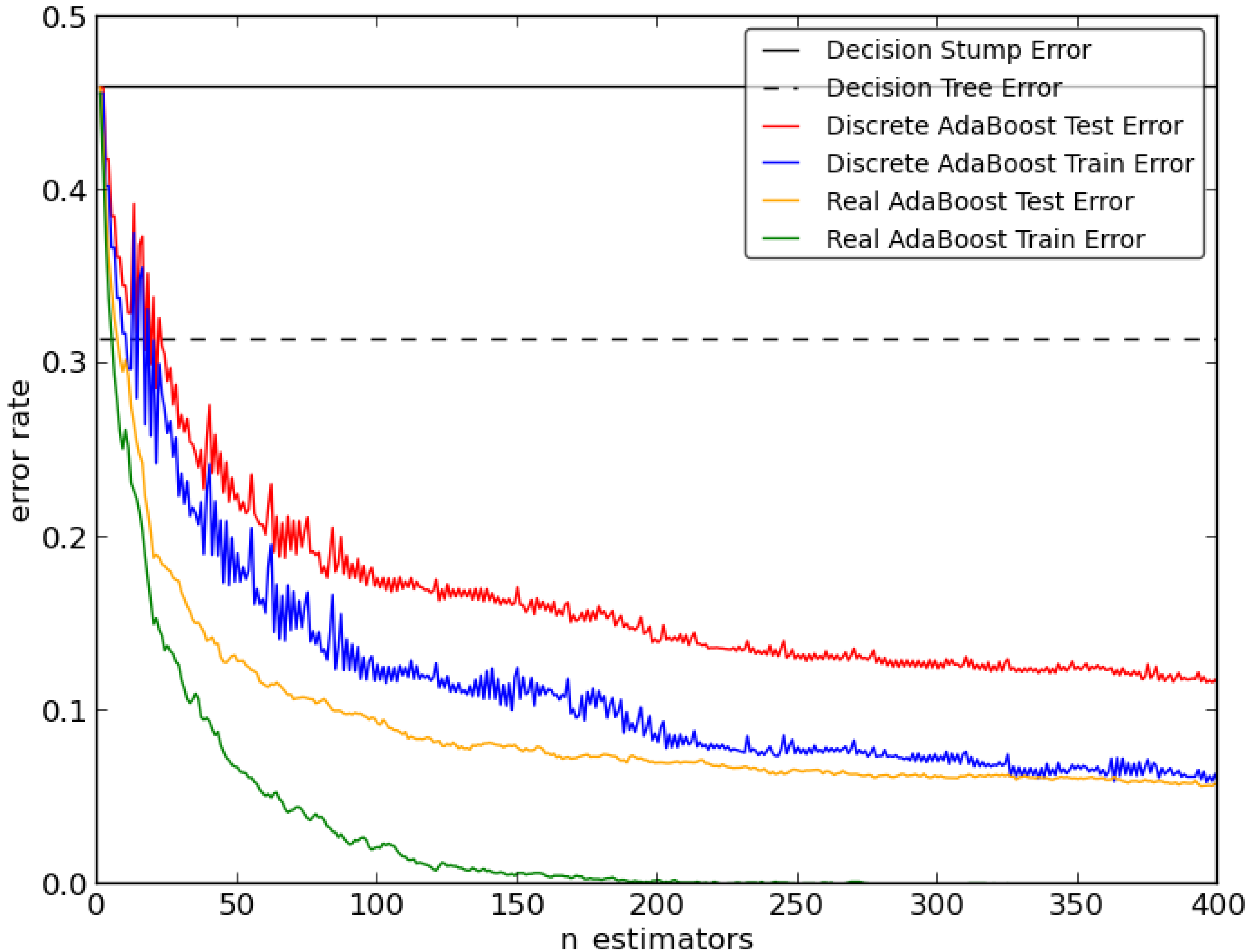
3: $\quad b_t := \arg\min_b N(b; W^\ell)$;

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

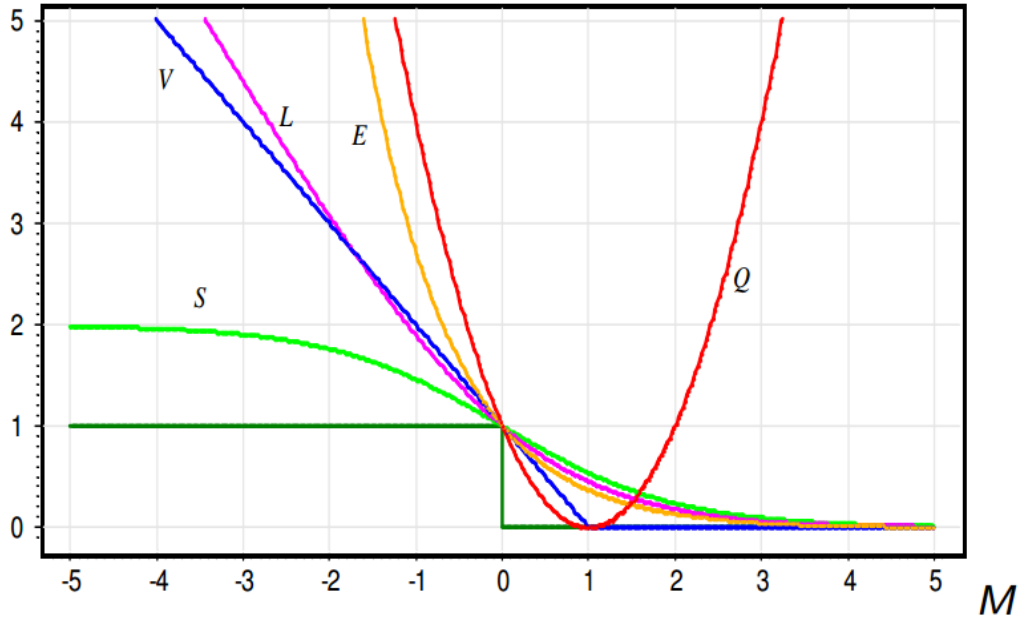
5: $\quad$ пересчет весов объектов: $w_i := w_i \exp\bigl(-\alpha_t y_i b_t(x_i)\bigr)$, $i = 1, \dots, \ell$;

[Discrete AdaBoost, источник: К.В. Воронцов, лекции по композициям алгоритмов]

AdaBoost над решающими деревьями

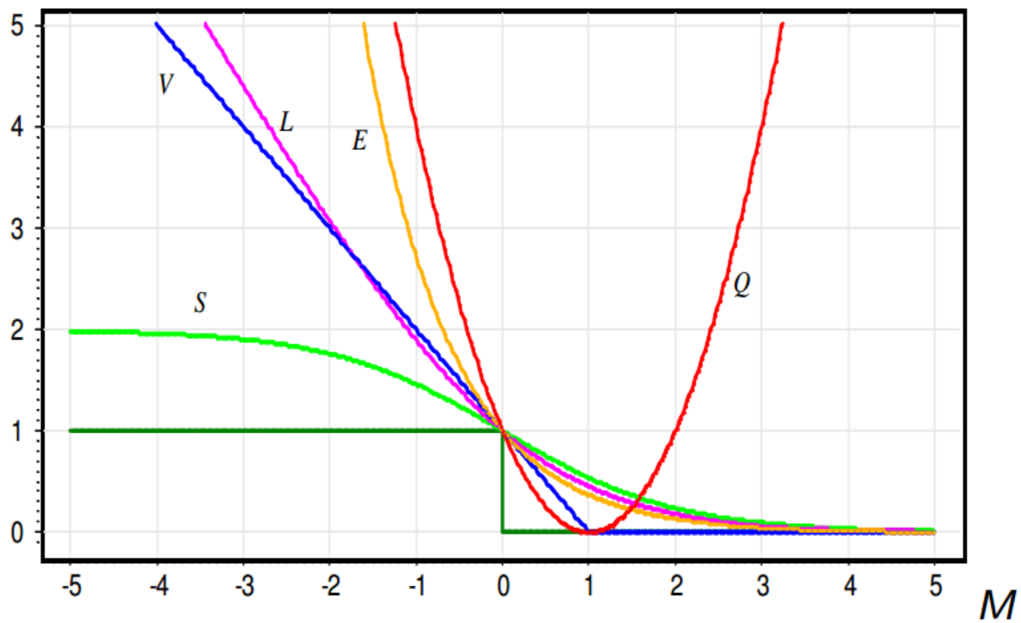


Функция потерь в AdaBoost



$$Q(M) = (1 - M)^2$$
$$V(M) = (1 - M)_+$$
$$S(M) = 2(1 + e^M)^{-1}$$
$$L(M) = \log_2(1 + e^{-M})$$
$$E(M) = e^{-M}$$

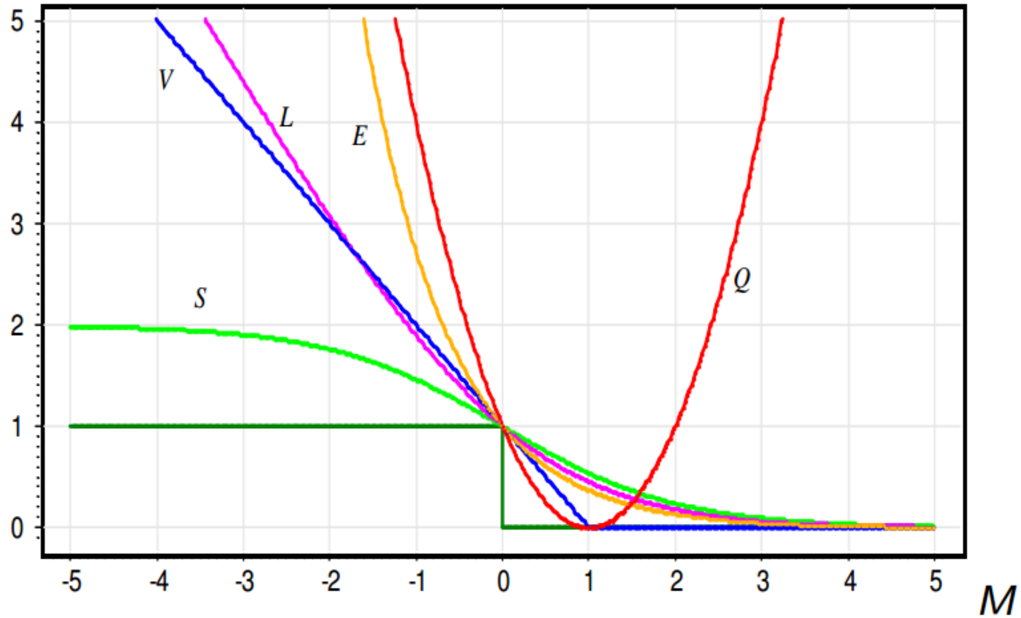
Функция потерь в AdaBoost



$Q(M) = (1 - M)^2$
$V(M) = (1 - M)_+$
$S(M) = 2(1 + e^M)^{-1}$
$L(M) = \log_2(1 + e^{-M})$
$E(M) = e^{-M}$

$$\sum_{i=1}^{l} [a(x_i) \neq y_i] = \sum_{i=1}^{l} [M_i \leq 0] \leq \sum_{i=1}^{l} L(M_i) =$$

Функция потерь в AdaBoost



$Q(M) = (1 - M)^2$
$V(M) = (1 - M)_+$
$S(M) = 2(1 + e^M)^{-1}$
$L(M) = \log_2(1 + e^{-M})$
$E(M) = e^{-M}$

$$\sum_{i=1}^{l} [a(x_i) \neq y_i] = \sum_{i=1}^{l} [M_i \leq 0] \leq \sum_{i=1}^{l} L(M_i) =$$

$$= \sum_{i=1}^{l} e^{-M_i} = \sum_{i=1}^{l} e^{-y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)}$$

Вывод формул для AdaBoost

Верхняя оценка для ошибки композиции из T+1 базового алгоритма:

$$\tilde{Q}_{T+1}(a, X^l) = \sum_{i=1}^{l} e^{-y_i \sum_{t=1}^{T+1} \alpha_t b_t(x_i)} = \sum_{i=1}^{l} e^{-y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) - y_i \alpha_{T+1} b_{T+1}(x_i)} =$$

Вывод формул для AdaBoost

Верхняя оценка для ошибки композиции из T+1 базового алгоритма:

$$\tilde{Q}_{T+1}(a, X^l) = \sum_{i=1}^{l} e^{-y_i \sum_{t=1}^{T+1} \alpha_t b_t(x_i)} = \sum_{i=1}^{l} e^{-y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) - y_i \alpha_{T+1} b_{T+1}(x_i)} =$$

$$= \sum_{i=1}^{l} e^{-y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)} e^{-y_i \alpha_{T+1} b_{T+1}(x_i)} = \sum_{i=1}^{l} w_i e^{-y_i \alpha_{T+1} b_{T+1}(x_i)}$$

(вот зачем ещё оказалась нужна экспоненциальная функция потерь)

Вывод формул для AdaBoost

Верхняя оценка для ошибки композиции из T+1 базового алгоритма:

$$\tilde{Q}_{T+1}(a, X^l) = \sum_{i=1}^{l} w_i e^{-y_i \alpha_{T+1} b_{T+1}(x_i)}$$

Для краткости обозначим далее $\alpha_{T+1} = \alpha, \; b_{T+1} = b$

$$e^{-y_i \alpha b(x_i)} = \begin{cases} e^{-\alpha}, & b(x_i) = y_i \\ e^{\alpha}, & b(x_i) = -y_i \\ 1, & b(x_i) = 0 \end{cases}$$

Вывод формул для AdaBoost

Верхняя оценка для ошибки композиции из T+1 базового алгоритма:

$$\tilde{Q}_{T+1}(a, X^l) = \sum_{i=1}^{l} w_i e^{-y_i \alpha_{T+1} b_{T+1}(x_i)}$$

Для краткости обозначим далее $\alpha_{T+1} = \alpha, \;\; b_{T+1} = b$

$$e^{-y_i \alpha b(x_i)} = \begin{cases} e^{-\alpha}, & b(x_i) = y_i \\ e^{\alpha}, & b(x_i) = -y_i \\ 1, & b(x_i) = 0 \end{cases}$$

$$\tilde{Q}_{T+1}(a, X^l) = \sum_{i: b(x_i) = y_i}^{l} w_i e^{-\alpha} + \sum_{i: b(x_i) = -y_i}^{l} w_i e^{\alpha} + \sum_{i: b(x_i) = 0}^{l} w_i$$

Вывод формул для AdaBoost

Верхняя оценка для ошибки композиции из T+1 базового алгоритма:

$$\tilde{Q}_{T+1}(a, X^l) = \sum_{i=1}^{l} w_i e^{-y_i \alpha_{T+1} b_{T+1}(x_i)}$$

Для краткости обозначим далее $\alpha_{T+1} = \alpha, \;\; b_{T+1} = b$

$$e^{-y_i \alpha b(x_i)} = \begin{cases} e^{-\alpha}, & b(x_i) = y_i \\ e^{\alpha}, & b(x_i) = -y_i \\ 1, & b(x_i) = 0 \end{cases}$$

$$\tilde{Q}_{T+1}(a, X^l) = \sum_{i:b(x_i)=y_i}^{l} w_i e^{-\alpha} + \sum_{i:b(x_i)=-y_i}^{l} w_i e^{\alpha} + \sum_{i:b(x_i)=0}^{l} w_i$$

$$P(b_T; W^l) = \sum_{i=1}^{l} w_i [b_T(x_i) = y_i] \qquad N(b_T; W^l) = \sum_{i=1}^{l} w_i [b_T(x_i) = -y_i]$$

Вывод формул для AdaBoost

$$\tilde{Q}_{T+1}(a, X^l) = e^{-\alpha} P(b_T; W^l) + e^{\alpha} N(b_T; W^l) + \sum_{i=1}^{l} w_i [b(x_i) = 0] =$$

$$= e^{-\alpha} P(b_T; W^l) + e^{\alpha} N(b_T; W^l) +$$

$$+ \tilde{Q}_T - P(b_T; W^l) - N(b_T; W^l)$$

$$P(b_T; W^l) = \sum_{i=1}^{l} w_i [b_T(x_i) = y_i] \qquad N(b_T; W^l) = \sum_{i=1}^{l} w_i [b_T(x_i) = -y_i]$$

Вывод формул для AdaBoost

$$\tilde{Q}_{T+1}(a, X^l) = e^{-\alpha} P(b_T; W^l) + e^{\alpha} N(b_T; W^l) + \sum_{i=1}^{l} w_i [b(x_i) = 0] =$$

$$= e^{-\alpha} P(b_T; W^l) + e^{\alpha} N(b_T; W^l) +$$
$$+\tilde{Q}_T - P(b_T; W^l) - N(b_T; W^l)$$

$$\frac{\partial}{\partial \alpha} \tilde{Q}_{T+1} = -\mathrm{e}^{-\alpha}\mathrm{P} + \mathrm{e}^{\alpha}\mathrm{N} = 0$$

$$\alpha = -\frac{1}{2} ln \frac{P}{N}$$

$$P(b_T; W^l) = \sum_{i=1}^{l} w_i [b_T(x_i) = y_i] \qquad N(b_T; W^l) = \sum_{i=1}^{l} w_i [b_T(x_i) = -y_i]$$

Вывод формул для AdaBoost

$$\alpha = \frac{1}{2} ln \frac{P}{N}$$

$$\tilde{Q}_{T+1} = \mathrm{e}^{-\alpha}\mathrm{P} + \mathrm{e}^{\alpha}\mathrm{N} + \tilde{Q}_T - P - N = \tilde{Q}_T - \left(\sqrt{P} - \sqrt{N}\right)^2 \to min$$

$$P\left(b_T; W^l\right) = \sum_{i=1}^{l} w_i [b_T(x_i) = y_i] \qquad N\left(b_T; W^l\right) = \sum_{i=1}^{l} w_i [b_T(x_i) = -y_i]$$

Некоторые подробности

- Откуда N в алгоритме из Воронцова
- Симметричное семейство классификаторов
- AnyBoost и неудачная замена функции потерь
- Какой функцией потерь правильно бороться с переобучением на выбросах

Преимущества и недостатки AdaBoost

- Переобучение на выбросах (обычная проблема бустинга, которую усиливает экспоненциальная функция потерь)
- Построение сильного алгоритма из слабых, но быстрых => работает быстро когда композиция уже построена
- Построение композиции из любых алгоритмов одного семейства

* AnyBoost

Рассмотрим построение композиции с произвольной функцией потерь:

$$Q_T \leqslant \widetilde{Q}_T = \sum_{i=1}^{\ell} \mathscr{L}\Biggl(\underbrace{y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)}_{M_T(x_i)}\Biggr) = \sum_{i=1}^{\ell} \mathscr{L}\bigl(M_{T-1}(x_i) + y_i \alpha_T b_T(x_i)\bigr)$$

$$\lambda(\alpha_T) = \mathscr{L}\bigl(M_{T-1}(x_i) + y_i \alpha_T b_T(x_i)\bigr)$$

$$\widetilde{Q}_T \approx \sum_{i=1}^{\ell} \mathscr{L}\bigl(M_{T-1}(x_i)\bigr) - \alpha_T \sum_{i=1}^{\ell} \underbrace{-\mathscr{L}'\bigl(M_{T-1}(x_i)\bigr)}_{w_i} y_i b_T(x_i)$$

$$\sum_{i=1}^{\ell} w_i y_i b(x_i) \to \max_b$$

* AnyBoost

Алгоритм 1.2. AnyBoost — обобщённый бустинг с произвольной функцией потерь

Вход:

X^ℓ, Y^ℓ — обучающая выборка; T — максимальное число базовых алгоритмов;

Выход:

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

1: инициализация отступов: $M_i := 0$, $i = 1, \dots, \ell$;

2: **для всех** $t = 1, \dots, T$, пока не выполнен критерий останова

3: $\quad b_t := \arg\max\limits_{b} \sum\limits_{i=1}^{\ell} w_i y_i b(x_i)$, где $w_i = -\mathscr{L}'(M_i)$, $i = 1, \dots, \ell$;

4: $\quad \alpha_t := \arg\min\limits_{\alpha > 0} \sum\limits_{i=1}^{\ell} \mathscr{L}\big(M_i + \alpha b_t(x_i) y_i\big)$;

5: $\quad$ пересчёт отступов: $M_i := M_i + \alpha_t b_t(x_i) y_i$; $i = 1, \dots, \ell$;

[Источник: К.В. Воронцов, лекции по композициям алгоритмов]