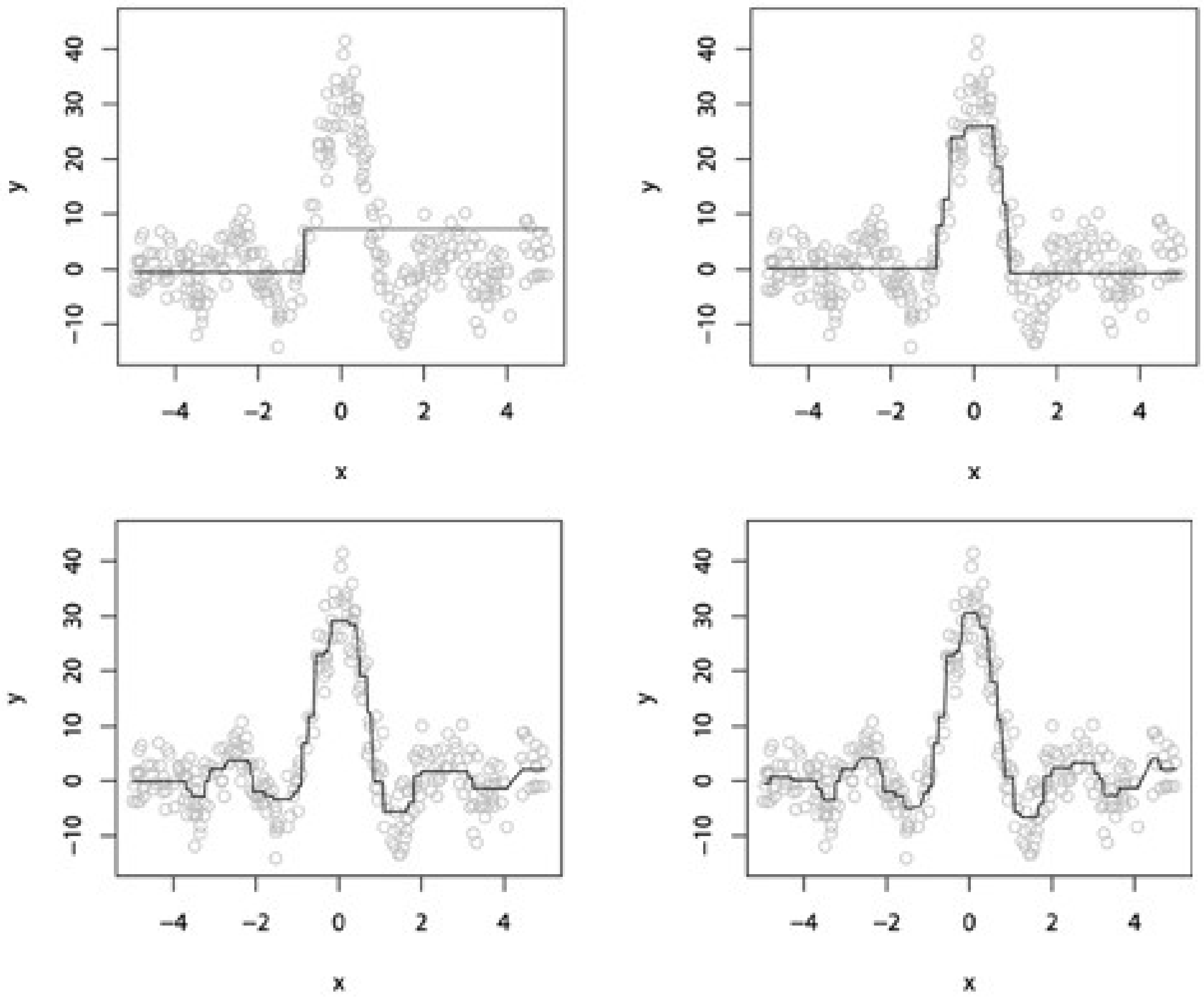
# Машинное обучение
# Композиции алгоритмов




https://yandexdataschool.ru/edu-process/courses/machine-learning

# Содержание лекции

- Определение
- AdaBoost
- AnyBoost
- Градиентный бустинг
- Бэггинг и метод случайных подпространств

# Определение

- Алгоритмы классификации часто представимы в виде: $a(x) = C(b(x))$, где функция $b : X \to R$ называется **алгоритмическим оператором**, $C : R \to Y$ — **решающим правилом**.
- **Композицией** $T$ алгоритмов $a_t(x) = C(b_t(x))$, $t = 1, \dots, T$ называется суперпозиция алгоритмических операторов $b_t : X \to R$, корректирующей операции $F : R^T \to R$ и решающего правила $C : R \to Y$ :

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

# Примеры

- Простое голосование

$$F\left(b_1(x), \ldots, b_T(x)\right) = \frac{1}{T} \sum_{t=1}^{T} b_t(x),$$

- Взвешенное голосование

$$F\left(b_1(x), \ldots, b_T(x)\right) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

# Общий алгоритм построения композиции

$$a(x) = C\big(F(b_1(x), \dots, b_T(x))\big) = \operatorname{sign}\Big(\sum_{t=1}^{T} \alpha_t b_t(x)\Big)$$

$$Q_T = \sum_{i=1}^{\ell} \Big[ y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) < 0 \Big] \to \textbf{min}$$

$Y = \{\pm 1\}, \;\; b_t \colon X \to \{-1, 0, +1\}, \;\; C(b) = \operatorname{sign}(b)$

$b_t(x) = 0$ — отказ (лучше промолчать, чем соврать)

- При добавлении в композицию слагаемого $\alpha_t b_t(x)$ оптимизируется только базовый алгоритм $b_t$ и коэффициент при нём $\alpha_t$ , а все предыдущие слагаемые $\alpha_1 b_1(x), \dots, \alpha_{t-1} b_{t-1}(x)$ полагаются фиксированными
- Пороговая функция потерь в функционале $Q_T$ аппроксимируется (заменяется) непрерывно дифференцируемой оценкой сверху.

# Общий алгоритм построения композиции

- Итерационный процесс:

$$b_1 = \arg\min_b Q\big(b, X^\ell\big);$$

$$b_2 = \arg\min_{b,F} Q\big(F(b_1, b), X^\ell\big);$$

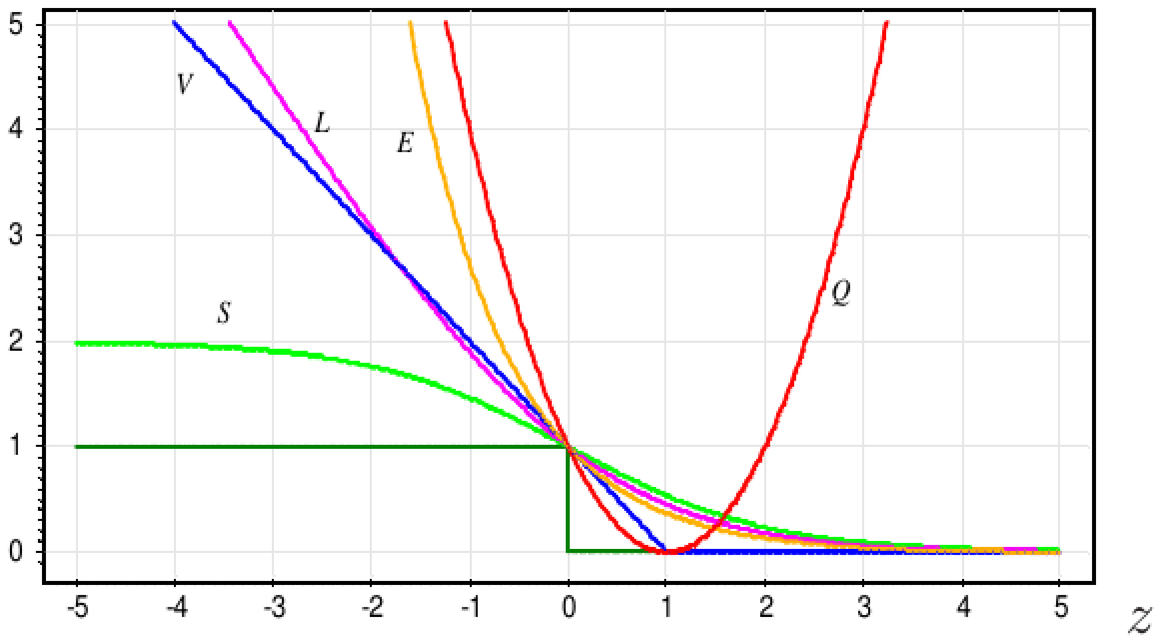
$$\dots$$

$$b_t = \arg\min_{b,F} Q\big(F(b_1, \dots, b_{t-1}, b), X^\ell\big)$$

- Алгоритм сильно упрощается, если задача для $b_t$ сводится к исходной с весами объектов и с, возможно, другой функцией потерь:

$$b_t = \arg\min_b \sum_{i=1}^{\ell} w_i \widetilde{\mathscr{L}}\big(b(x_i), y_i\big)$$

# Аппроксимации функции потерь




$$S(z) = 2(1+e^{z})^{-1} \text{ — сигмоидная;}$$
$$L(z) = \log_2(1+e^{-z}) \text{ — логарифмическая;}$$
$$V(z) = (1-z)_+ \text{ — кусочно-линейная;}$$
$$E(z) = e^{-z} \text{ — экспоненциальная;}$$
$$Q(z) = (1-z)^2 \text{ — квадратичная.}$$

# AdaBoost

- Исторически первый
- Использует экспоненциальную аппроксимацию $[y_i b(x_i) < 0] \leqslant e^{-y_i b(x_i)}$
- Оценим функционал качества сверху

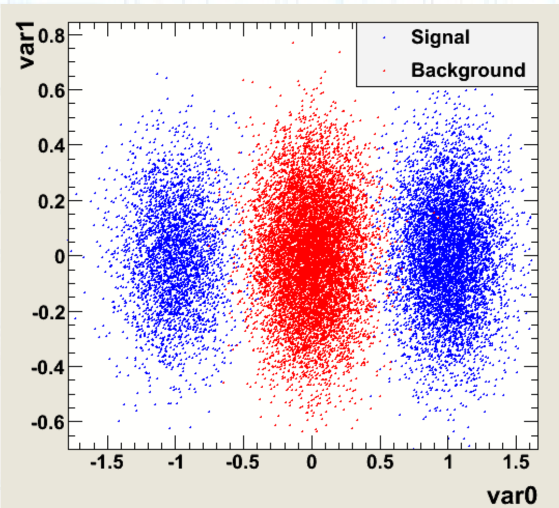
$$Q_T \leqslant \widetilde{Q}_T = \sum_{i=1}^{\ell} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)\right) =$$

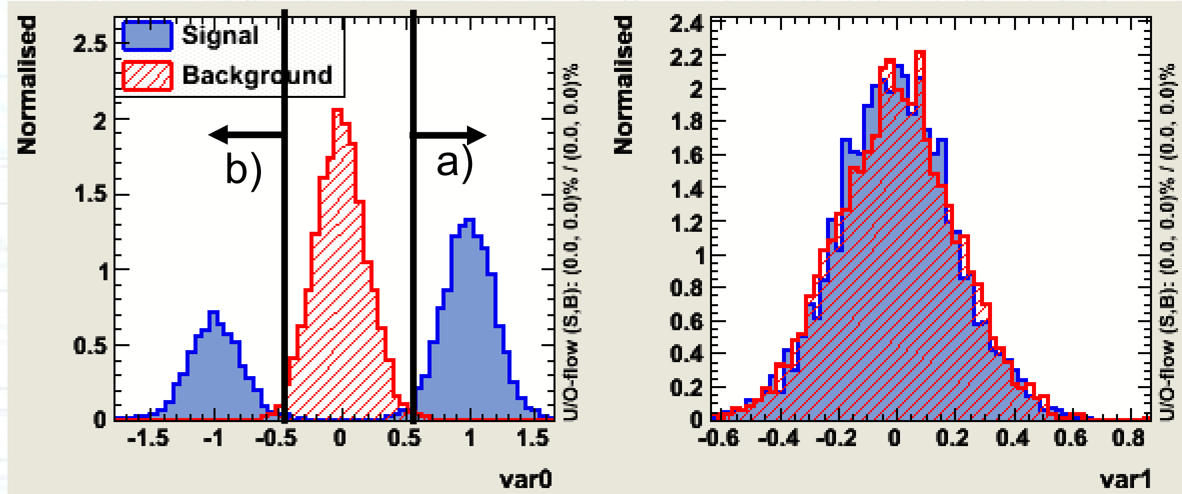
$$= \sum_{i=1}^{\ell} \underbrace{\exp\left(-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)\right)}_{w_i} e^{-y_i \alpha_T b_T(x_i)} \to \mathbf{min}$$

- Таким образом, задача сведена к исходной, но с весами и другой функцией потерь

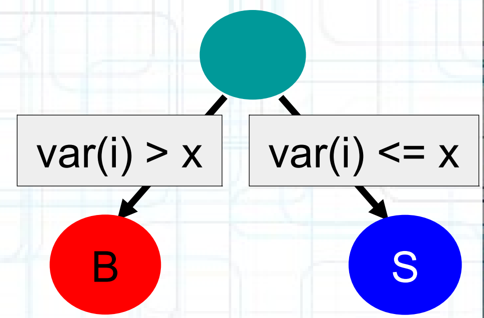
# AdaBoost: демонстрация

Обучающая выборка:




Базовый классификатор:




Всего два нормальных пороговых предиката:

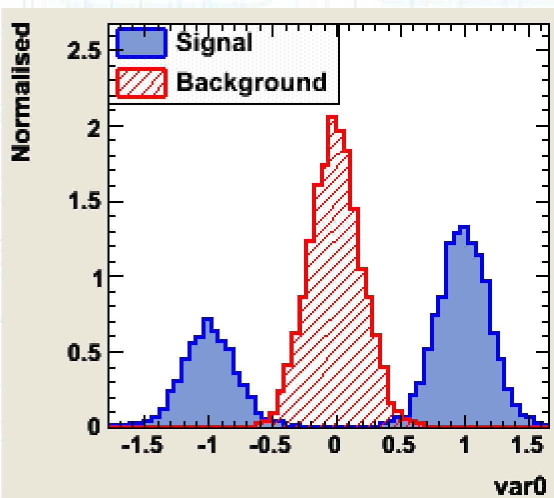
a) Var0 > 0.5 → $\varepsilon_{signal}$=66% $\varepsilon_{bkg} \approx 0\%$ - общая ошибка 16.5%

b) Var0 < -0.5 → $\varepsilon_{signal}$=33% $\varepsilon_{bkg} \approx 0\%$ - общая ошибка 33%

На первой итерации бустинга будет выбрано дерево с разделителем a)

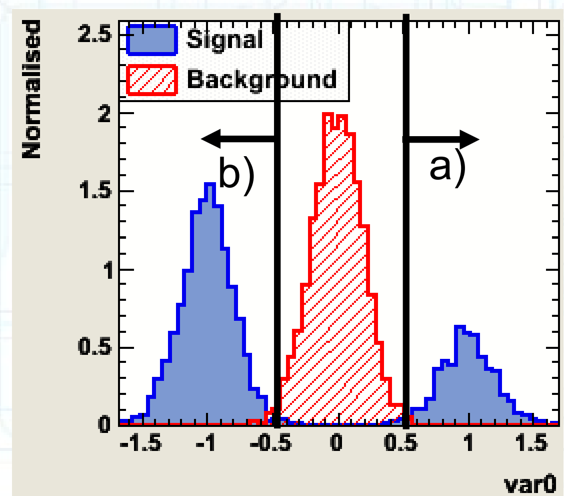
# AdaBoost: демонстрация

- Первый классификатор, выбрав разрез а) ошибается в 16.5 % случаев
- Перед построением следующего дерева присвоим объектам веса exp(-M)




Новый

вес




Теперь разрез b) приводит к меньшей ошибке, чем а).
Второе дерево разделит выборку по предикату Var0 < -0.5

# AdaBoost

- Нормируем веса, чтобы их сумма = 1
- Определим две метрики качества:

$$N(b; U^\ell) = \sum_{i=1}^{\ell} u_i [b(x_i) = -y_i]; \qquad P(b; U^\ell) = \sum_{i=1}^{\ell} u_i [b(x_i) = y_i].$$

- Теорема (Freund, Schapire, 1996)

Пусть для любого нормированного вектора весов $U^\ell$ существует алгоритм $b \in B$, классифицирующий выборку хотя бы немного лучше, чем наугад: $P(b; U^\ell) > N(b; U^\ell)$.

Тогда минимум функционала $\widetilde{Q}_T$ достигается при

$$b_T = \arg\max_{b \in B} \sqrt{P(b; \widetilde{W}^\ell)} - \sqrt{N(b; \widetilde{W}^\ell)}.$$

$$\alpha_T = \frac{1}{2} \ln \frac{P(b_T; \widetilde{W}^\ell)}{N(b_T; \widetilde{W}^\ell)}.$$

# Доказательство

Воспользуемся тождеством $\forall\alpha \in \mathbb{R},\ \forall b \in \{-1, 0, +1\}$:
$e^{-\alpha b} = e^{-\alpha}[b{=}1] + e^{\alpha}[b{=}{-1}] + [b{=}0]$.

Положим для краткости $\alpha = \alpha_T$ и $b_i = b_T(x_i)$. Тогда

$$\widetilde{Q}_T = \Biggl(e^{-\alpha}\underbrace{\sum_{i=1}^{\ell}\tilde{w}_i[b_i{=}y_i]}_{P} + e^{\alpha}\underbrace{\sum_{i=1}^{\ell}\tilde{w}_i[b_i{=}{-y_i}]}_{N} + \underbrace{\sum_{i=1}^{\ell}\tilde{w}_i[b_i{=}0]}_{1-P-N}\Biggr)\underbrace{\sum_{i=1}^{\ell} w_i}_{\widetilde{Q}_{T-1}}$$

$$= \bigl(e^{-\alpha}P + e^{\alpha}N + (1 - P - N)\bigr)\widetilde{Q}_{T-1} \to \min_{\alpha,b}.$$

$$\frac{\partial}{\partial\alpha}\widetilde{Q}_T = \bigl(-e^{-\alpha}P{+}e^{\alpha}N\bigr)\widetilde{Q}_{T-1} = 0 \ \Rightarrow\ e^{-\alpha}P = e^{\alpha}N \ \Rightarrow\ e^{2\alpha} = \frac{P}{N}.$$

Получили требуемое: $\boxed{\alpha_T = \frac{1}{2}\ln\frac{P}{N}}$.

# Доказательство

Подставим оптимальное значение $\alpha = \frac{1}{2}\ln\frac{P}{N}$ обратно в $\widetilde{Q}_T$:

$$\widetilde{Q}_T = \bigl(e^{-\alpha}P + e^{\alpha}N + (1-P-N)\bigr)\widetilde{Q}_{T-1} =$$
$$= \Bigl(1 + \sqrt{\tfrac{N}{P}}P + \sqrt{\tfrac{P}{N}}N - P - N\Bigr)\widetilde{Q}_{T-1} =$$
$$= \left(1 - \bigl(\sqrt{P} - \sqrt{N}\bigr)^2\right)\widetilde{Q}_{T-1} \to \min_b .$$

Поскольку $\widetilde{Q}_{T-1}$ не зависит от $\alpha_T$ и $b_T$, минимизация $\widetilde{Q}_T$ эквивалентна либо максимизации $\sqrt{P} - \sqrt{N}$ при $P > N$, либо максимизации $\sqrt{N} - \sqrt{P}$ при $P < N$, однако второй случай исключён условием теоремы.

Получили $\boxed{b_T = \arg\max_b \sqrt{P} - \sqrt{N}}$. Теорема доказана.

# Классический AdaBoost

- Пусть отказов нет, $b_t : X \to \{\pm 1\}$. Тогда $P = 1 - N$
- Тогда минимум функционала $Q_T$ достигается при

$$b_T = \arg\min_{b \in B} N(b; \widetilde{W}^\ell).$$

$$\alpha_T = \frac{1}{2} \ln \frac{1 - N(b_T; \widetilde{W}^\ell)}{N(b_T; \widetilde{W}^\ell)}.$$

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

2: **для всех** $t = 1, \dots, T$

3: обучить базовый алгоритм:
$b_t := \arg\min\limits_b N(b; W^\ell)$;

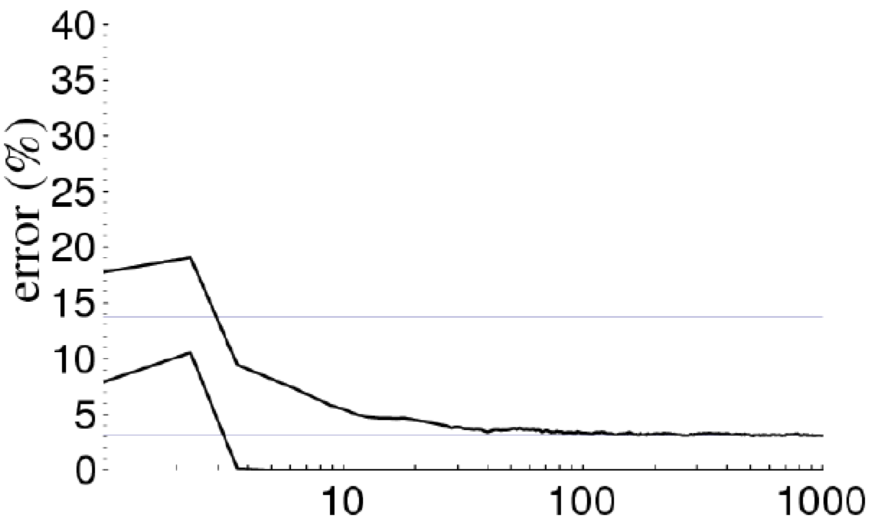
4: $$\alpha_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^\ell)}{N(b_t; W^\ell)};$$

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

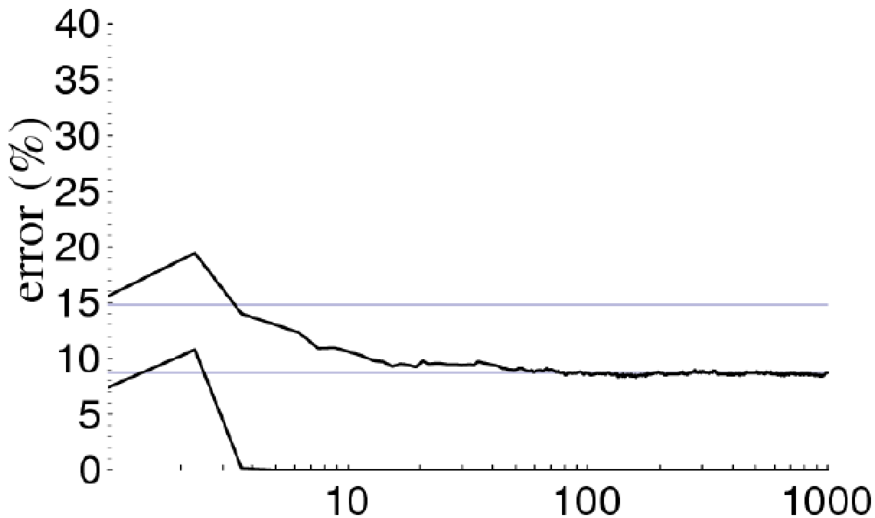
6: нормировать веса объектов:
$w_0 := \sum_{j=1}^{\ell} w_j$;
$w_i := w_i / w_0$, $i = 1, \dots, \ell$;

# Обобщающая способность не ухудшается с ростом сложности T

Задача UCI:letter




Задача UCI:satimage




Schapire, Freund, Lee, Bartlett (1998) Boosting the margin: a new explanation for the effectiveness of voting methods // Annals of Statistics Vol.26, No.5, Pp. 1651–1686.

# Замечания

- Базовые классификаторы (weak classifiers):
  - решающие деревья — используются чаще всего
  - пороговые правила (data stumps)

$$B = \Big\{ b(x) = \big[ f_j(x) \leqslant \theta \big] \;\Big|\; j = 1, \ldots, n,\ \theta \in \mathbb{R} \Big\};$$

- Базовые классификаторы должны быть слабыми, из сильных хорошую композицию не построить
- Недостатки AdaBoost:
  - чрезмерная чувствительность к выбросам из-за $e^{-M}$

# Обобщение: $\mathcal{L} = \mathcal{L}(M)$. AnyBoost

Возьмём $Y = \{\pm 1\}$, $b_t \colon X \to \mathbb{R}$, $C(b) = \mathrm{sign}(b)$;
$\mathscr{L}(M)$ — функция потерь, гладкая функция отступа $M$;

$M_T(x_i) = y_i \sum\limits_{t=1}^{T} \alpha_t b_t(x_i)$ — отступ композиции на объекте $x_i$;

Оценка сверху для числа ошибок композиции:

$$Q_T \leqslant \widetilde{Q}_T = \sum_{i=1}^{\ell} \mathscr{L}\bigl(M_{T-1}(x_i) + \alpha y_i b(x_i)\bigr) \to \min_{\alpha, b}.$$

Линеаризация функции потерь по $\alpha$ в окрестности $\alpha = 0$:

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

где $w_i$ — веса объектов.

# Обобщение: $\mathcal{L} = \mathcal{L}(M)$. AnyBoost

Минимизация линеаризованного $\widetilde{Q}_T$ при фиксированном $\alpha$

$$\widetilde{Q}_T \approx \sum_{i=1}^{\ell} \mathscr{L}\bigl(M_{T-1}(x_i)\bigr) - \alpha \sum_{i=1}^{\ell} w_i y_i b(x_i) \to \min_b .$$

приводит к принципу *явной максимизации отступов* (direct optimization of margin, DOOM):

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

Затем $\alpha$ определяется путём одномерной минимизации $\widetilde{Q}_T$.

Итерации этих двух шагов приводят к алгоритму AnyBoost.

**Замечание.** AnyBoost переходит в AdaBoost в частном случае, при $b_t\colon X \to \{-1, 0, +1\}$ и $\mathscr{L}(M) = e^{-M}$.

# Алгоритм AnyBoost

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

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

1: инициализировать отступы: $M_i := 0$, $i = 1, \dots, \ell$;

2: **для всех** $t = 1, \dots, T$

3: вычислить веса объектов:

$w_i = -\mathscr{L}'(M_i)$, $i = 1, \dots, \ell$;

4: обучить базовый алгоритм согласно принципу DOOM:

$$b_t := \arg\max_b \sum_{i=1}^{\ell} w_i y_i b(x_i);$$

5: решить задачу одномерной минимизации:

$$\alpha_t := \arg\min_{\alpha > 0} \sum_{i=1}^{\ell} \mathscr{L}\bigl(M_i + \alpha b_t(x_i) y_i\bigr);$$

6: обновить значения отступов:

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

# Обобщение: $\mathcal{L}$ - ∀ . Градиентный бустинг

Линейная (выпуклая) комбинация базовых алгоритмов:

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}_+.$$

Функционал качества с произвольной функцией потерь $\mathscr{L}(a, y)$:

$$Q(\alpha, b; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\Bigl(\underbrace{\underbrace{\sum_{t=1}^{T-1} \alpha_t b_t(x_i)}_{f_{T-1,i}} + \alpha b(x_i)}_{f_{T,i}}, y_i\Bigr) \to \min_{\alpha, b}.$$

$f_{T-1} = (f_{T-1,i})_{i=1}^{\ell}$ — текущее приближение
$f_T = (f_{T,i})_{i=1}^{\ell}$ — искомый вектор, решение задачи $Q(f) \to \min$

Friedman G. Greedy Function Approximation: A Gradient Boosting Machine. 1999.

# Обобщение: $\mathcal{L}$ - $\forall$. Градиентный бустинг

Градиентный метод минимизации $Q(f) \to \min$, $f \in \mathbb{R}^\ell$:

$$f_0 := \text{начальное приближение};$$

$$f_{T,i} := f_{T-1,i} - \alpha g_i, \quad i = 1, \dots, \ell;$$

$g_i = \mathscr{L}'\big(f_{T-1,i}, y_i\big)$ — компоненты вектора градиента,
$\alpha$ — градиентный шаг.

**Наблюдение:** это очень похоже на одну итерацию бустинга!

$$f_{T,i} := f_{T-1,i} + \alpha b(x_i), \quad i = 1, \dots, \ell$$

**Идея:** будем искать такой базовый алгоритм $b_T$, чтобы вектор $\big(b_T(x_i)\big)_{i=1}^\ell$ приближал вектор градиента $\big(-g_i\big)_{i=1}^\ell$:

$$b_T := \arg\min_b \sum_{i=1}^{\ell} \big(b(x_i) + g_i\big)^2$$

# Алгоритм градиентного бустинга

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

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

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

2: **для всех** $t = 1, \dots, T$

3: найти базовый алгоритм, приближающий градиент:

$$b_t := \arg\min_b \sum_{i=1}^{\ell} \bigl(b(x_i) + \mathscr{L}'(f_i, y_i)\bigr)^2;$$

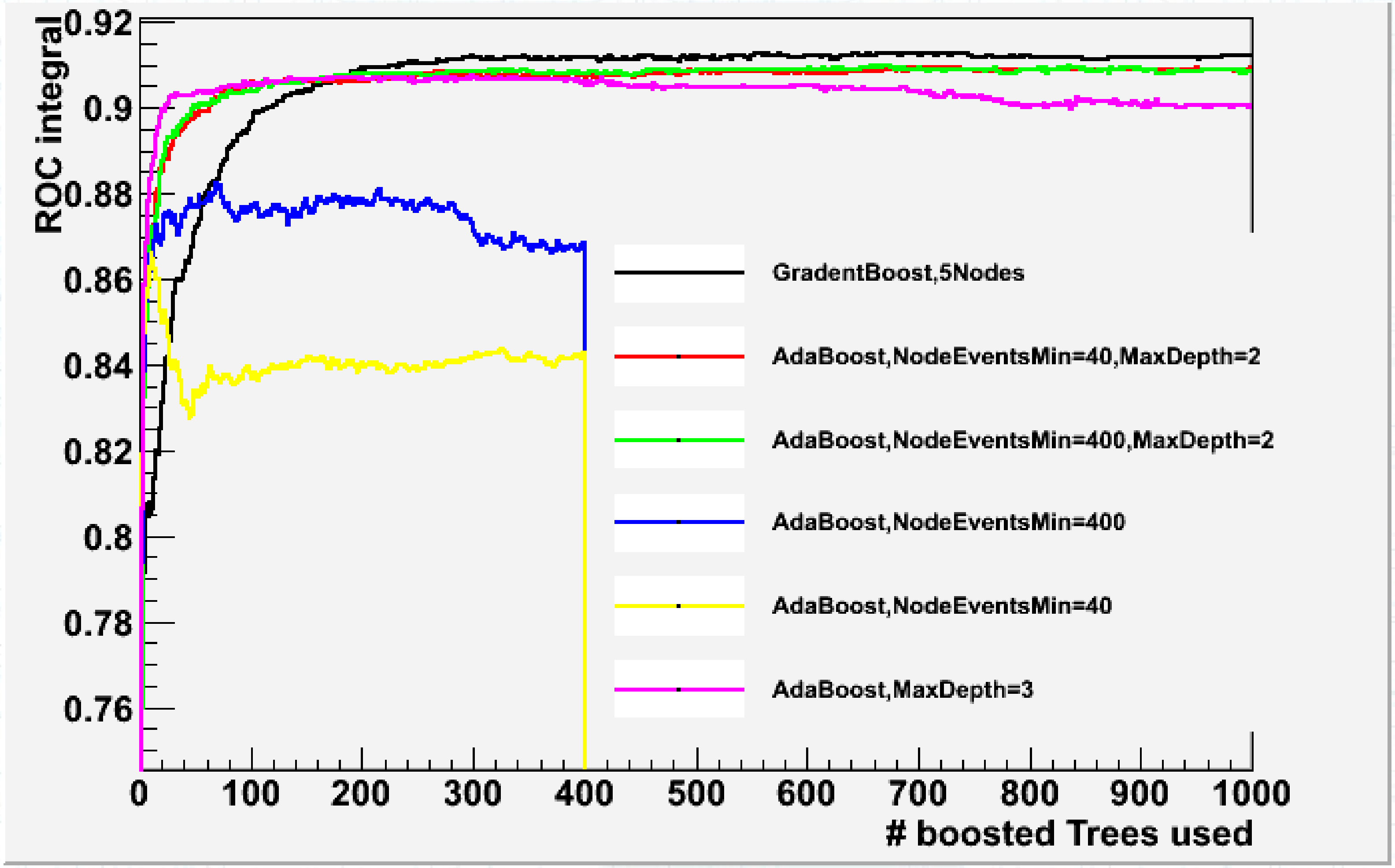
4: решить задачу одномерной минимизации:

$$\alpha_t := \arg\min_{\alpha > 0} \sum_{i=1}^{\ell} \mathscr{L}\bigl(f_i + \alpha b_t(x_i), y_i\bigr);$$

5: обновить значения композиции на объектах выборки:

$f_i := f_i + \alpha_t b_t(x_i)$; $i = 1, \dots, \ell$;

# Эксперименты




Бустинг работает лучше всего для “слабых” базовых классификаторов

Необходимо подбирать параметры деревьев

# Выводы

- Градиентный бустинг — наиболее общий из всех бустингов:
  - произвольная функция потерь
  - произвольное пространство оценок R
  - подходит для регрессии, классификации, ранжирования
- Интересная интерпретация бустинга: добавление базового алгоритма — это одна итерация градиентного спуска
- Обычно GB применяется к решающим деревьям
- Градиентный бустинг над ODT = Yandex.MatrixNet

# Стохастические методы построения композиций

- Bagging – обучает базовые алгоритмы по случайным подвыборкам (подмножество строк матрицы О-П)
- Метод случайных подпространств (RSM) – обучает базовые алгоритмы по случайным подмножествам признаков (подмножество строк матрицы О-П)

# Алгоритм

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

$\ell'$ — длина обучающих подвыборок;

$n'$ — длина признакового подописания;

$\varepsilon_1$ — 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: $U :=$ случайное подмножество $X^\ell$ длины $\ell'$;
3: $\mathscr{G} :=$ случайное подмножество $\mathscr{F}$ длины $n'$;
4: $b_t := \mu(\mathscr{G}, U)$;
5: **если** $Q(b_t, U) > \varepsilon_1$ или $Q(b_t, X^\ell \setminus U) > \varepsilon_2$ **то**
6: не включать $b_t$ в композицию;

**Композиция** — простое голосование: $a(x) = C\left(\sum_{t=1}^{T} b_t(x)\right)$.

# Сравнение

- Бустинг лучше для больших обучающих выборок и для классов с границами сложной формы.
- Бэггинг и RSM лучше для коротких обучающих выборок.
- RSM лучше в тех случаях, когда признаков больше, чем объектов, или когда много неинформативных признаков.
- Бэггинг и RSM эффективно распараллеливаются, бустинг выполняется строго последовательно.

# Случайный лес (Random forest)

- каждое дерево $b_t(x)$ обучается по случайной подвыборке с повторениями (Bagging)
- в каждой вершине каждого дерева рассматривается случайное подмножество из $\sqrt{n}$ признаков (напоминает RSM)
- для ветвления выбирается признак с наилучшим пороговым предикатом по энтропийному критерию
- усечений (pruning) деревьев нет