Композиции классификаторов

К. В. Воронцов
vokov@forecsys.ru

Этот курс доступен на странице вики-ресурса
http://www.MachineLearning.ru/wiki
«Машинное обучение (курс лекций, К.В.Воронцов)»

16 апреля 2018

Содержание

Определение композиции

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

$a(x) = C(b(x))$ — алгоритм, где

$b\colon X \to R$ — *базовый алгоритм* (алгоритмический оператор),

$C\colon R \to Y$ — *решающее правило*,

R — *пространство оценок*;

Определение

Композиция базовых алгоритмов $b_1, \ldots, b_T$

$$a(x) = C\bigl(F(b_1(x), \ldots, b_T(x))\bigr),$$

где $F\colon R^T \to R$ — *корректирующая операция*.

Зачем вводится R?

В задачах классификации множество отображений $\{F\colon R^T \to R\}$ существенно шире, чем $\{F\colon Y^T \to Y\}$.

Примеры пространств оценок и решающих правил

- **Пример 1:** классификация на 2 класса, $Y = \{-1, +1\}$:
$$a(x) = \operatorname{sign}\bigl(b(x)\bigr),$$
где $R = \mathbb{R}$, $b\colon X \to \mathbb{R}$, $C(b) \equiv \operatorname{sign}(b)$.
- **Пример 2:** классификация на M классов $Y = \{1, \dots, M\}$:
$$a(x) = \arg\max_{y \in Y} b_y(x),$$
где $R = \mathbb{R}^M$, $b\colon X \to \mathbb{R}^M$, $C(b_1, \dots, b_M) \equiv \arg\max\limits_{y \in Y} b_y$.
- **Пример 3:** регрессия, $Y = R = \mathbb{R}$:
$C(b) \equiv b$ — решающее правило не нужно.

Примеры корректирующих операций

- **Пример 1:** Простое голосование (Simple Voting):

$$F\big(b_1(x),\dots,b_T(x)\big) = \frac{1}{T}\sum_{t=1}^{T} b_t(x), \quad x \in X.$$

- **Пример 2:** Взвешенное голосование (Weighted Voting):

$$F\big(b_1(x),\dots,b_T(x)\big) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}.$$

- **Пример 3:** Смесь алгоритмов (Mixture of Experts)

$$F\big(b_1(x),\dots,b_T(x)\big) = \sum_{t=1}^{T} g_t(x) b_t(x), \quad x \in X, \quad g_t\colon X \to \mathbb{R}.$$

Бустинг для задачи классификации с двумя классами

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

Взвешенное голосование:

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

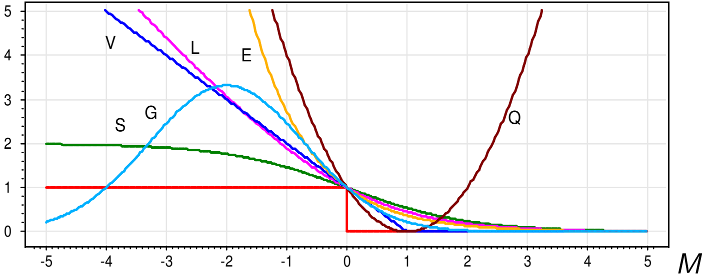
Функционал качества композиции — число ошибок на X^ℓ:

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

Две основные эвристики бустинга:

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

Гладкие аппроксимации пороговой функции потерь $[M < 0]$



$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);
$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LogitBoost);
$Q(M) = (1 - M)^2$ — квадратичная (GentleBoost);
$G(M) = \exp\bigl(-cM(M + s)\bigr)$ — гауссовская (BrownBoost);
$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная;
$V(M) = (1 - M)_+$ — кусочно-линейная (из SVM);

Экспоненциальная аппроксимация пороговой функции потерь

Оценка функционала качества Q_T сверху:

$$Q_T \leqslant \widetilde{Q}_T = \sum_{i=1}^{\ell} \underbrace{\exp\left(-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)\right)}_{w_i} \exp\bigl(-y_i \alpha_T b_T(x_i)\bigr)$$

Нормированные веса: $\widetilde{W}^\ell = \bigl(\tilde{w}_1, \dots, \tilde{w}_\ell\bigr)$, $\tilde{w}_i = w_i \,/\, \sum_{j=1}^{\ell} w_j$.

Взвешенное число ошибочных (negative) и правильных (positive) классификаций при векторе весов $U^\ell = (u_1, \dots, u_\ell)$:

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

$1 - N - P$ — взвешенное число отказов от классификации.

Основная теорема бустинга (для AdaBoost)

Пусть B — *достаточно богатое* семейство базовых алгоритмов.

Теорема (Freund, Schapire, 1996)

Пусть для любого нормированного вектора весов U^ℓ 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 (шаг 1 из 2)

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 (шаг 2 из 2)

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

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

Поскольку $\widetilde{Q}_{T-1}$ не зависит от α_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.

Следствие 1. Классический вариант AdaBoost

Пусть отказов нет, $b_t\colon X \to \{\pm 1\}$. Тогда $P = 1 - N$.

Теорема (Freund, Schapire, 1995)

Пусть для любого нормированного вектора весов U^ℓ существует алгоритм $b \in B$, классифицирующий выборку хотя бы немного лучше, чем наугад: $N(b; U^\ell) < \frac{1}{2}$.

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

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

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

Следствие 2. Сходимость

Теорема

Если на каждом шаге семейство B и метод обучения обеспечивают построение базового алгоритма b_t такого, что

$$\sqrt{P(b_t;\widetilde{W}^\ell)} - \sqrt{N(b_t;\widetilde{W}^\ell)} = \gamma_t > \gamma$$

при некотором $\gamma > 0$, то за конечное число шагов будет построен корректный алгоритм $a(x)$.

Доказательство. Q_T сходится к нулю со скоростью геометрической прогрессии:

$$Q_{T+1} \leqslant \widetilde{Q}_{T+1} = \widetilde{Q}_T(1-\gamma^2) \leqslant \cdots \leqslant \widetilde{Q}_1(1-\gamma^2)^T.$$

Наступит момент, когда $\widetilde{Q}_T < 1$.
Но тогда $Q_T = 0$, поскольку $Q_T \in \{0, 1, \ldots, \ell\}$.

Алгоритм AdaBoost

Вход: обучающая выборка X^ℓ; 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:
$b_t := \arg\min\limits_b N(b; W^\ell)$;

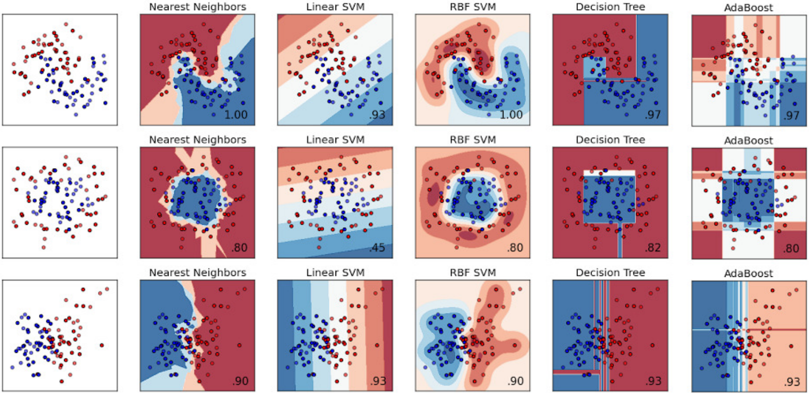
4: $\alpha_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^\ell)}{N(b_t; W^\ell)}$;

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

Бустинг и другие методы классификации

Эксперименты на трёх двумерных модельных выборках:

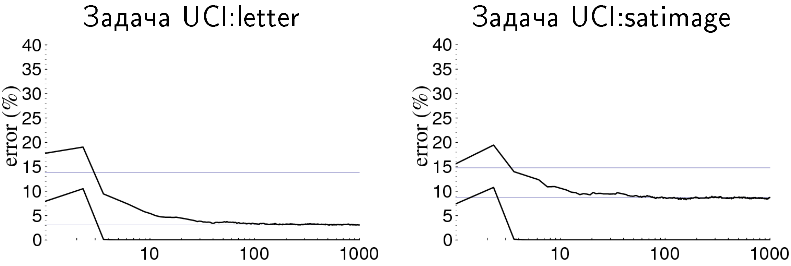


Эвристики и рекомендации

- **Базовые классификаторы (weak classifiers):**
  — решающие деревья — используются чаще всего;
  — пороговые правила (data stumps)
  $$B = \Bigl\{ b(x) = \bigl[f_j(x) \lessgtr \theta \bigr] \Bigm| j = 1, \dots, n,\ \theta \in \mathbb{R} \Bigr\};$$
  — для SVM бустинг не эффективен.
- **Отсев шума:** отбросить объекты с наибольшими w_i.
- **Модификация формулы для α_t на случай $N = 0$:**
  $$\alpha_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^\ell) + \frac{1}{\ell}}{N(b_t; W^\ell) + \frac{1}{\ell}};$$
- **Дополнительный критерий остановки:**
  увеличение частоты ошибок на контрольной выборке.

Эксперименты с бустингом

Удивительное отсутствие переобучения вплоть до $T = 1000$ (нижняя кривая — обучение, верхняя – контроль):



Schapire, Freund, Lee, Bartlett. Boosting the margin: a new explanation for the effectiveness of voting methods // Annals of Statistics, 1998.

Обоснование бустинга

Усиление понятия *частоты ошибок* алгоритма $a(x) = \operatorname{sign} b(x)$:

$$\nu_\theta(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \bigl[b(x_i) y_i \leqslant \theta\bigr], \quad \theta > 0.$$

Обычная частота ошибок $\nu_0(a, X^\ell) \leqslant \nu_\theta(a, X^\ell)$ при $\theta > 0$.

Теорема (Freund, Schapire, Bartlett, 1998)

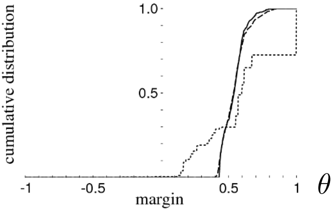
Если $|B| < \infty$, то $\forall \theta > 0$, $\forall \eta \in (0,1)$ с вероятностью $1 - \eta$

$$\mathsf{P}[y a(x) < 0] \leqslant \nu_\theta(a, X^\ell) + C \sqrt{\frac{\ln |B| \ln \ell}{\ell \theta^2} + \frac{1}{\ell} \ln \frac{1}{\eta}}$$

Основной вывод: оценка зависит от $|B|$, но не от T.
Голосование не увеличивает сложность эффективно используемого множества алгоритмов.

Обоснование бустинга: что же всё-таки происходит?

Распределение отступов: доля объектов, имеющих отступ меньше заданного θ после 5, 100, 1000 итераций (Задача UCI:vehicle)



- С ростом T распределение отступов сдвигается вправо, то есть бустинг «раздвигает» классы в пространстве векторов растущей размерности $\bigl(b_1(x), \ldots, b_T(x)\bigr)$
- Значит, в оценке можно уменьшить второй член, увеличив θ и не изменив $\nu_\theta(a, X^\ell)$.
- Можно уменьшить второй член, если уменьшить $|B|$, то есть взять простое семейство базовых алгоритмов.

Schapire R., Freund Y., Lee W. S., Bartlett P. Boosting the margin: a new explanation for the effectiveness of voting methods. 1998.

Недостатки AdaBoost

- Чрезмерная чувствительность к выбросам из-за e^{-M}
- AdaBoost строит «чёрные ящики» — громоздкие неинтерпретируемые композиции из сотен алгоритмов
- Не удаётся строить короткие композиции из «сильных» алгоритмов типа SVM (только длинные из слабых)
- Требуются достаточно большие обучающие выборки (бэггинг обходится более короткими — см. далее)

Способы устранения:

- Другие аппроксимации пороговой функции потерь
- Непрерывные вещественные базовые алгоритмы $b_t\colon X \to \mathbb{R}$
- Явная оптимизация отступов, без аппроксимации
- Менее жадные стратегии наращивания композиции

Градиентный бустинг для произвольной функции потерь

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 $\mathscr{L}(a, y)$:

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

$f_{T-1} = \bigl(f_{T-1,i}\bigr)_{i=1}^{\ell}$ — текущее приближение
$f_T = \bigl(f_{T,i}\bigr)_{i=1}^{\ell}$ — следующее приближение

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

Параметрическая аппроксимация градиентного шага

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

$$f_0 := \text{начальное приближение};$$
$$f_{T,i} := f_{T-1,i} - \alpha g_i, \quad i = 1, \dots, \ell;$$

$g_i = \mathscr{L}'\bigl(f_{T-1,i},\, y_i\bigr)$ — компоненты вектора градиента,
α — 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 $\bigl(b_T(x_i)\bigr)_{i=1}^{\ell}$ приближал вектор антиградиента $\bigl(-g_i\bigr)_{i=1}^{\ell}$:

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

Алгоритм градиентного бустинга (Gradient Boosting)

Вход: обучающая выборка X^ℓ; 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: базовый алгоритм, приближающий антиградиент:

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

4: задача 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: обновление вектора значений на объектах выборки:

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

Стохастический градиентный бустинг (SGB)

Идея: на шагах 3–5 использовать не всю выборку X^ℓ, а случайную подвыборку без возвращений

Преимущества:

- улучшается качество
- улучшается сходимость
- уменьшается время обучения

Friedman G. Stochastic Gradient Boosting. 1999.

Регрессия и AdaBoost

Регрессия: $\mathscr{L}(a,y) = (a-y)^2$

- $b_T(x)$ обучается на разностях $y_i - \sum\limits_{t=1}^{T-1} \alpha_t b_t(x_i)$
- если регрессия линейная, то α_t можно не обучать.

Классификация: $\mathscr{L}(a,y) = e^{-ay}$, $b_t \in \{-1, 0, +1\}$

- GB в точности совпадает с AdaBoost.

Классификация: $\mathscr{L}(a,y) = \mathscr{L}(-ay)$, $b_t \in \mathbb{R}$

- GB совпадает с AnyBoost — см. далее.

AnyBoost: классификация с произвольной функцией потерь

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 α 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.

Принцип явной максимизации отступов

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

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

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 α 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^ℓ; 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:

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

XGBoost — популярная и быстрая реализация GB над деревьями

Деревья регрессии и классификации (CART):

$$b(x) = \sum_{j=1}^{J} w_j \bigl[x \in R_j\bigr]$$

где R_j — область пространства, покрываемая листом j,
w_j — веса листьев, J — число листьев в дереве.

Функционал качества с суммой L_0, L_1, L_2 регуляризаторов:

$$Q\bigl(b, \{w_j\}_{j=1}^{J}; X^\ell\bigr) = \sum_{i=1}^{\ell} \mathscr{L}\biggl(\sum_{t=1}^{T-1} \alpha_t b_t(x_i) + \alpha b(x_i), y_i\biggr) +$$

$$+ \gamma \sum_{j=1}^{J} [w_j \neq 0] + \mu \sum_{j=1}^{J} |w_j| + \frac{\lambda}{2} \sum_{j=1}^{J} w_j^2 \to \min_{b,\{w_j\}}.$$

По w_j задача имеет аналитическое решение.

Стохастические методы построения композиций

Чтобы алгоритмы в композиции были различными

- их обучают по (случайным) подвыборкам,
- либо по (случайным) подмножествам признаков.

Первую идею реализует bagging (bootstrap aggregation) [Breiman, 1996]: подвыборки длины ℓ с повторениями, доля объектов, попадающих в выборку: $(1 - \frac{1}{e}) \approx 0.632$

Вторую идею реализует RSM (random subspace method) [Ho, 1998].

Совместим обе идеи в одном алгоритме.

$\mathscr{F} = \{f_1, \ldots, f_n\}$ — признаки,
$\mu(\mathscr{G}, U)$ — метод обучения алгоритма по подвыборке $U \subseteq X^\ell$, использующий только признаки из $\mathscr{G} \subseteq \mathscr{F}$.

Бэггинг и метод случайных подпространств

Вход: обучающая выборка X^ℓ; параметры: T;
ℓ' — длина обучающих подвыборок;
n' — длина признакового подописания;
ε_1 — порог качества базовых алгоритмов на обучении;
ε_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^ℓ длины ℓ';
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: $a(x) = C\Bigl(\sum\limits_{t=1}^{T} b_t(x)\Bigr)$.

Сравнение: boosting — bagging — RSM

- 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

И ещё несколько эмпирических наблюдений:

- Веса алгоритмов не столь важны для выравнивания отступов
- Веса объектов не столь важны для обеспечения различности
- Короткие композиции из «сильных» алгоритмов типа SVM строить труднее, чем длинные из слабых

Оптимизация распределения отступов на каждом шаге

Возьмём $Y = \{\pm 1\}$, $F(b_1, \dots, b_T) = \frac{1}{T} \sum\limits_{t=1}^{T} b_t$, $C(b) = \mathrm{sign}(b)$.

Функционал качества композиции — число ошибок на обучении:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell} \bigl[y_i a(x_i) < 0\bigr] = \sum_{i=1}^{\ell} \bigl[\underbrace{y_i b_1(x_i) + \dots + y_i b_T(x_i)}_{M_{iT}} < 0\bigr],$$

$M_{it} = y_i b_1(x_i) + \dots + y_i b_t(x_i)$ — *отступ* (margin) объекта x_i.

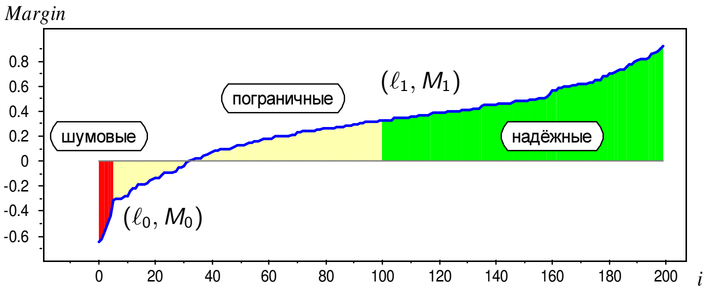
Эвристика: чтобы b_{t+1} компенсировал ошибки композиции,

$$Q(b, U) = \sum_{x_i \in U} \bigl[y_i b(x_i) < 0\bigr] \to \min_b,$$

где $U = \bigl\{x_i \colon M_0 < M_{it} \leqslant M_1\bigr\}$,
M_0, M_1 — параметры метода обучения.

Подбор параметров M_0 и M_1

Упорядочим объекты по возрастанию отступов M_{it}:



Принцип максимизации и выравнивания отступов.

Два случая, когда b_{t+1} на объекте x_i обучать не надо:

$M_{it} < M_0$, $i < \ell_0$ — объект x_i шумовой;

$M_{it} > M_1$, $i > \ell_1$ — объект x_i уже надёжно классифицируется.

Алгоритм ComBoost (Committee Boosting)

Вход: обучающая выборка X^ℓ; параметры $T, \ell_0, \ell_1, \ell_2, \Delta\ell$;

Выход: $b_1, \dots, b_T$

1: $b_1 := \arg\min\limits_b Q(b, X^\ell)$;
упорядочить X^ℓ по возрастанию $M_i = y_i b_t(x_i)$, $i = 1, \dots, \ell$;

2: **для всех** $t = 1, \dots, T$

3: **для всех** $k = \ell_1, \dots, \ell_2$ с шагом $\Delta\ell$

4: $U = \{x_i \in X^\ell \colon \ell_0 \leqslant i \leqslant k\}$;

5: $b_{tk} := \arg\min\limits_b Q(b, U)$;

6: выбрать наилучший $b_t \in \{b_{tk}\}$ по критерию Q;

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

8: упорядочить выборку X^ℓ по возрастанию отступов M_i;

9: опция: скорректировать значения параметров $\ell_0, \ell_1, \Delta\ell$;

10: **пока** Q существенно улучшается.

Результаты эксперимента на 4 задачах из репозитория UCI

Средняя частота ошибок на контроле по 50 случайным разбиениям в отношении «обучение : контроль» = 4 : 1.

	ionoshere	pima	bupa	votes
SVM	12,9	24,2	42	4,6
$\mathtt{ComBoost}_0$[SVM]	12,6	23,1	34,2	4
ComBoost[SVM]	**12,3**	**22,5**	30,9	**3,8**
AdaBoost[SVM]	15	22,7	**30,6**	4
Parzen	6,3	25,1	41,6	6,9
$\mathtt{ComBoost}_0$[Parzen]	6,1	25	38,1	6,8
ComBoost[Parzen]	**5,8**	**24,7**	30,6	**6,2**
AdaBoost[Parzen]	6	24,8	**30,5**	6,5

$\mathtt{ComBoost}_0$ — с подбором ℓ_0 и $\ell_1 = \ell_2$ по скользящему контролю;
ComBoost — с подбором длины подвыборки U;
Parzen — окно Парзена с евклидовой метрикой и подбором ширины окна скользящим контролем LOO.

Результаты эксперимента на 4 задачах из репозитория UCI

Мощность композиций:

Число базовых алгоритмов	ionoshere	pima	bupa	votes
$\texttt{ComBoost}_0$ над SVM	**4**	**2**	**5**	**2**
ComBoost над SVM	5	**2**	5	3
AdaBoost над SVM	65	18	15	8

Критерий останова: отсутствие существенного улучшения качества классификации обучающей выборки.

Маценов А. А. Комитетный бустинг: минимизация числа базовых алгоритмов при простом голосовании // ММРО-13, 2007.

Обобщение для задач с произвольным числом классов

Пусть теперь $Y = \{1, \dots, M\}$.

Композиция — простое голосование, причём
каждый базовый алгоритм b_{yt} голосует только за свой класс y:

$$a(x) = \arg\max_{y \in Y} \Gamma_y(x); \qquad \Gamma_y(x) = \frac{1}{|T_y|} \sum_{t \in T_y} b_{yt}(x).$$

В алгоритме только два изменения:
— изменится определение отступа M_i:

$$M_i = \Gamma_{y_i}(x_i) - \max_{y \in Y \setminus \{y_i\}} \Gamma_y(x_i).$$

— в алгоритме ComBoost на шаге 3 придётся решать,
за какой класс строить очередной базовый алгоритм,
кроме того, немного изменится шаг 7 (пересчёт отступов).

Преобразование простого голосования во взвешенное

Линейный классификатор над признаками $b_t(x)$:

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

1. **Метод обучения:** SVM, логистическая регрессия, и т.п.:

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

2. **Регуляризация:** $\alpha_t \geqslant 0$ либо LASSO: $\sum\limits_{t=1}^{T} |\alpha_t| \leqslant \varkappa$.

3. **Наивный байесовский классификатор**
приводит к простому аналитическому решению:

$$\alpha_t = \ln \frac{1 - p_t}{p_t}, \quad t = 1, \dots, T,$$

где p_t — оценка вероятности ошибки базового алгоритма b_t.

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
- регрессия: $k = \lfloor n/3 \rfloor$; классификация: $k = \lfloor \sqrt{n} \rfloor$

Подбор числа деревьев T по критерию *out-of-bag*: число ошибок на объектах x_i, если не учитывать голоса деревьев, для которых x_i был обучающим:

$$\text{out-of-bag}(a) = \sum_{i=1}^{\ell} \Bigl[\text{sign}\Bigl(\sum_{t=1}^{T} [x_i \notin U_t]\, b_t(x_i) \Bigr) \neq y_i \Bigr] \to \min$$

Это несмещённая оценка обобщающей способности.

Breiman L. Random Forests // Machine Learning. 2001.

Квазилинейная композиция (смесь алгоритмов)

Смесь алгоритмов (Mixture of Experts)

$$a(x) = C\left(\sum_{t=1}^{T} g_t(x) b_t(x)\right),$$

$b_t\colon X \to \mathbb{R}$ — базовый алгоритм,
$g_t\colon X \to \mathbb{R}$ — функция компетентности, шлюз (gate).
Чем больше $g_t(x)$, тем выше доверие к ответу $b_t(x)$.

Условие нормировки: $\sum\limits_{t=1}^{T} g_t(x) = 1$ для любого $x \in X$.

Нормировка «мягкого максимума» SoftMax: $\mathbb{R}^T \to \mathbb{R}^T$:

$$\tilde{g}_t(x) = \mathrm{SoftMax}_t\bigl(g_1(x), \dots, g_T(x); \gamma\bigr) = \frac{e^{\gamma g_t(x)}}{e^{\gamma g_1(x)} + \dots + e^{\gamma g_T(x)}}.$$

При $\gamma \to \infty$ SoftMax выделяет максимальную из T величин.

Вид функций компетентности

Функции компетентности выбираются из содержательных соображений и могут определяться:

- признаком $f(x)$:
$$g(x;\alpha,\beta) = \sigma(\alpha f(x) + \beta), \quad \alpha,\beta \in \mathbb{R};$$
- неизвестным направлением $\alpha \in \mathbb{R}^n$:
$$g(x;\alpha,\beta) = \sigma(x^{\mathsf{T}}\alpha + \beta), \quad \alpha \in \mathbb{R}^n,\ \beta \in \mathbb{R};$$
- расстоянием до неизвестной точки $\alpha \in \mathbb{R}^n$:
$$g(x;\alpha,\beta) = \exp(-\beta\|x-\alpha\|^2), \quad \alpha \in \mathbb{R}^n,\ \beta \in \mathbb{R};$$

где $\alpha,\beta \in \mathbb{R}$ — параметры, *частично* обучаемые по выборке, $\sigma(z) = \frac{1}{1+e^{-z}}$ — сигмоидная функция.

Выпуклые функции потерь

Функция потерь $\mathscr{L}(b, y)$ называется *выпуклой* по b, если $\forall\, y \in Y,\ \forall\, b_1, b_2 \in R,\ \forall\, g_1, g_2 \geqslant 0\colon\ g_1 + g_2 = 1$, выполняется

$$\mathscr{L}(g_1 b_1 + g_2 b_2, y) \leqslant g_1 \mathscr{L}(b_1, y) + g_2 \mathscr{L}(b_2, y).$$

Интерпретация: потери растут не медленнее, чем величина отклонения от правильного ответа y.

Примеры выпуклых функций потерь:

$$\mathscr{L}(b, y) = \begin{cases} (b - y)^2 & \text{— квадратичная (МНК-регрессия);} \\ e^{-by} & \text{— экспоненциальная (AdaBoost);} \\ \log_2(1 + e^{-by}) & \text{— логарифмическая (LR);} \\ (1 - by)_+ & \text{— кусочно-линейная (SVM).} \end{cases}$$

Пример невыпуклой функции потерь: $\mathscr{L}(b, y) = [by < 0]$.

Основная идея применения выпуклых функций потерь

Пусть $\forall x \;\; \sum\limits_{t=1}^{T} g_t(x) = 1$ и функция потерь $\mathscr{L}$ выпукла.
Тогда $Q(a)$ распадается на T независимых функционалов Q_t:

$$Q(a) = \sum_{i=1}^{\ell} \mathscr{L}\biggl(\sum_{t=1}^{T} g_t(x_i) b_t(x_i), y_i\biggr) \leqslant \sum_{t=1}^{T} \underbrace{\sum_{i=1}^{\ell} g_t(x_i) \mathscr{L}\bigl(b_t(x_i), y_i\bigr)}_{Q_t(g_t, b_t)}.$$

Итерационный процесс, аналогичный EM-алгоритму:

1: начальное приближение функций компетентности g_t;
2: **повторять**
3: **M-шаг:** при фиксированных g_t обучить все b_t;
4: **E-шаг:** при фиксированных b_t оценить все g_t;
5: **пока** значения компетентностей $g_t(x_i)$ не стабилизируются.

Алгоритм ME: обучение смеси алгоритмов

Итерационный процесс, аналогичный EM-алгоритму:

Вход: выборка X^ℓ, нормированные $(g_t)_{t=1}^T$, параметры T, δ, γ;

Выход: $g_t(x), b_t(x),\ \ t = 1, \dots, T$;

1: **повторять**

2: $g_t^0 := g_t$ для всех $t = 1, \dots, T$;

3: **M-шаг:** при фиксированных g_t обучить все b_t:

$$b_t := \arg\min_b \sum_{i=1}^{\ell} g_t(x_i) \mathscr{L}\bigl(b(x_i), y_i\bigr), \qquad t = 1, \dots, T;$$

4: **E-шаг:** при фиксированных b_t оценить все g_t:

$$g_t := \arg\min_{g_t} \sum_{i=1}^{\ell} \mathscr{L}\left(\frac{\sum_{s=1}^T e^{\gamma g_s(x_i)} b_s(x_i)}{\sum_{s=1}^T e^{\gamma g_s(x_i)}}, y_i\right), \qquad t = 1, \dots, T;$$

5: нормировать компетентности:

$$\bigl(g_1(x_i), \dots, g_T(x_i)\bigr) := \mathrm{SoftMax}\bigl(g_1(x_i), \dots, g_T(x_i); \gamma\bigr);$$

6: **пока** $\max\limits_{t,i} \bigl| g_t(x_i) - g_t^0(x_i) \bigr| > \delta$.

Обучение смеси с автоматическим определением числа T

Вход: выборка X^ℓ, параметры ℓ_0, $\mathscr{L}_0$, δ, γ;
Выход: T, $g_t(x), b_t(x)$, $t = 1, \dots, T$;

1: начальное приближение:

$$b_1 := \arg\min_b \sum_{i=1}^{\ell} \mathscr{L}\bigl(b(x_i), y_i\bigr), \quad g_1(x_i) := 1, \ i = 1, \dots, \ell;$$

2: **для всех** $t = 2, \dots, T$

3: множество трудных объектов:
$X_t := \bigl\{x_i \colon \mathscr{L}(a_{t-1}(x_i), y_i) > \mathscr{L}_0\bigr\}$;

4: **если** $|X_t| \leqslant \ell_0$ **то выход**;

5: $b_t := \arg\min_b \sum_{x_i \in X_t} \mathscr{L}\bigl(b(x_i), y_i\bigr)$;

6: $g_t := \arg\min_{g_t} \sum_{i=1}^{\ell} \mathscr{L}\left(\sum_{s=1}^{t} g_s(x_i) b_s(x_i), y_i\right)$;

7: $(g_s, b_s)_{s=1}^t := \mathrm{ME}\bigl(X^\ell, (g_s)_{s=1}^t, t, \delta, \gamma\bigr)$;

Резюме

- Композиции позволяют решать сложные задачи, которые плохо решаются отдельными алгоритмами
- Бустинг — обучает базовые алгоритмы по очереди
- Бэггинг — обучает базовые алгоритмы независимо
- Важное открытие середины 90-х: обобщающая способность бустинга не ухудшается с ростом сложности T
- ComBoost — короткие композиции из сильных базовых
- Градиентный бустинг — наиболее общий из всех бустингов
- XGBoost — наиболее популярный вариант GB
- Смеси алгоритмов — модели областей компетентности