

Композиция методов. Бустинг.

Григорьева Ирина Владимировна, гр. 622

Санкт-Петербургский государственный университет
Прикладная математика и информатика
Статистическое моделирование

Санкт-Петербург, 2017г.

Постановка задачи

$\langle X, Y, f, X^n \rangle$, где

- X — пространство объектов, Y — множество ответов,
- $f : X \rightarrow Y$ — неизвестная целевая зависимость,
- $X^n = (x_1, \dots, x_n)$ — обучающая выборка,
- $Y^n = (y_1, \dots, y_n)$ — вектор ответов на обучающих объектах, где $y_i = f(x_i)$.

Требуется построить алгоритм $a(x) = C(b(x))$, аппроксимирующий целевую зависимость f на всем X , где

- $b : X \rightarrow R$ — базовый алгоритм,
- $C : R \rightarrow Y$ — решающее правило,
- R — пространство оценок.

В случае решения задачи классификации $b(x)$ может являться вероятность принадлежности объекта x классу, которое решающее правило C переводит в номер класса.

В случае же регрессии $C(b) = b$.

Изменение постановки задачи

► Вместо одного базового алгоритма b рассматривается b_1, \dots, b_T .

$\mathcal{B}(\Theta) = \{b(\cdot; \theta) | \theta \in \Theta\}$ — параметризованное множество базовых алгоритмов,

Выбор базового алгоритма: выбор $\theta \in \Theta$ и $b(x) = b(x; \theta) \in \mathcal{B}(\Theta)$:

- решающие деревья — используются чаще всего;
- пороговые правила (data stumps): $b(x, \theta = \{i, s\}) = [f_i(x) \leq s]$.

Определение

Композиция базовых алгоритмов $b_1, \dots, b_T \in \mathcal{B}$ имеет вид

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

где $F : R^T \rightarrow R$ — корректирующая операция, параметризованная с помощью $\omega \in \Omega$.

Например, F — линейная, то есть $\omega = (\omega_1, \dots, \omega_T) \in \mathbb{R}^T$

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

Задача: Подбор параметров ω и алгоритмов $\{b_t(x)\}_{t=1}^T$.

Примеры пространства оценок и решающих правил:

- Классификация на 2 класса, $Y = \{-1, 1\}$:

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

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

- Классификация на M классов, $Y = \{1, \dots, M\}$:

$$a(x) = \underset{y \in Y}{\operatorname{argmax}} b_y(x),$$

где $R = \mathbb{R}^M$, $b : X \rightarrow \mathbb{R}^M$, $C(b) = \underset{y \in Y}{\operatorname{argmax}} b_y(x)$.

Примеры корректирующих операций:

- Простое голосование:

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

- Взвешенное голосование:

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

Bagging:

Построение деревьев для bootstrap sample и создание единой предсказательной модели.

- Наблюдения имеют одинаковые шансы попасть в обучающую выборку.
- Каждое дерево строится независимо от других деревьев (параллельное обучение базовых алгоритмов).

Boosting:

Работает аналогичным образом, за исключением того, что

- Обучающая выборка на каждой итерации определяется, исходя из ошибок классификации на предыдущих итерациях.
- Каждое дерево строится с использованием информации из ранее выращенных деревьев (последовательное обучение базовых алгоритмов).

► Жадность алгоритма:

Пусть $T_0 < T$, выбрали $\{b_t(x)\}_{t=1}^{T_0}$ и параметры $\{\omega_t\}_{t=1}^{T_0}$.

На $(T_0 + 1)$ шаге выбираем ω_{T_0+1} и базовый алгоритм b_{T_0+1} так, чтобы исправить ошибки композиции, основанной на на первых T_0 алгоритмах

$$a(x) = C \left(\sum_{t=1}^{T_0} \omega_t b_t(x) \right).$$

При этом базовые алгоритмы $\{b_t\}_{t=1}^{T_0}$ и параметры корректирующей функции $\{\omega_t\}_{t=1}^{T_0}$ остаются без изменений.

Пример работы бустинга для регрессии

Задача регрессии: $C(b) = b$, $Y = \mathbb{R}$, $R = \mathbb{R}$.

Рассмотрим $\mathcal{B}(\Theta)$ — семейство неглубоких решающих деревьев, где θ — число терминальных узлов дерева.

Пусть

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

Обучим алгоритм

$$b_1(x) = \operatorname{argmin}_{b \in \mathcal{B}} \frac{1}{n} \sum_{i=1}^n (b(x_i) - y_i)^2.$$

Добавим b_2 , исправляющий ошибки b_1 : $b_1(x_i) + b_2(x_i) = y_i$.

Поправка $y_i - b_1(x_i)$, $i = 1, \dots, n$.

$$b_2(x) = \operatorname{argmin}_{b \in \mathcal{B}} \frac{1}{n} \sum_{i=1}^n (b(x_i) - (b_1(x_i) - y_i))^2,$$

...

$$b_T(x) = \operatorname{argmin}_{b \in \mathcal{B}} \frac{1}{n} \sum_{i=1}^n (b(x_i) - (\sum_{t=1}^{T-1} b_t(x_i) - y_i))^2.$$

Бустинг для задачи классификации на два класса

Пусть $Y = \{-1, 1\}$, $C(b) = \text{sign}(b)$,

$b_t : X \rightarrow \{-1, 0, 1\}$, $b_t \in \mathcal{B}(\Theta)$, $b_t(x) = 0$ — отказ базового алгоритма от классификации x .

Положим

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

Тогда

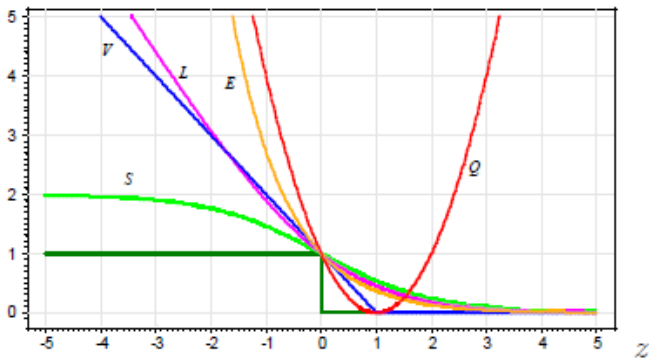
$$a(x) = \text{sign}\left(\sum_{t=1}^T \omega_t b_t(x)\right), \quad x \in X.$$

Функционал качества композиции — число ошибок на X^n :

$$Q_T = \sum_{i=1}^n [y_i \sum_{t=1}^T \omega_t b_t(x_i) < 0]$$

Q_T мажорируется некоторой непрерывно дифференцируемой функцией, поддающейся эффективной оптимизации. Выбор функции зависит от характера задачи.

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



$S(z) = 2(1 + \exp(z))^{-1}$ — сигмоидная,

$L(z) = \log_2(1 + \exp(-z))$ — логарифмическая,

$V(z) = (1 - z)_+$ — кусочно-линейная,

$E(z) = \exp(-z)$ — экспоненциальная,

$Q(z) = (1 - z)^2$ — квадратичная.

Оценка функционала Q_T сверху имеет вид:

$$Q_T \leq \tilde{Q}_T = \sum_{i=1}^n \underbrace{\exp\{-y_i \sum_{t=1}^{T-1} \omega_t b_t(x_i)\}}_{v_i} \exp\{-y_i \omega_T b_T(x_i)\}.$$

Нормированные веса: $\tilde{V}_n = (\tilde{v}_1, \dots, \tilde{v}_n)$, $\tilde{v}_i = v_i / \sum_{j=1}^n v_j$

Введем функционалы качества алгоритма b на выборке X^n , Y^n с нормированным вектором весов $U_n = (u_1, \dots, u_n)$ — суммарный вес ошибочных (negative) и правильных (positive) классификаций:

$$N(b, U_n) = \sum_{i=1}^n u_i [b(x_i) = -y_i], \quad P(b, U_n) = \sum_{i=1}^n u_i [b(x_i) = y_i].$$

При отсутствии отказов b от классификации: $N + P = 1$.

Основная теорема бустинга (для AdaBoost)

Теорема (Freund, Shapire, 1996)

Пусть для любого нормированного вектора весов U_n существует алгоритм $b \in \mathcal{B}(\Theta)$, классифицирующий выборку немного лучше, чем наугад: $P(b, U_n) > N(b, U_n)$.

Тогда минимум функционала \tilde{Q}_T достигается при

$$b_T = \operatorname{argmax}_{b \in \mathcal{B}} (\sqrt{P(b, \tilde{V}_n)} - \sqrt{N(b, \tilde{V}_n)}),$$

$$\omega_T = \frac{1}{2} \ln \frac{P(b_T, \tilde{V}_n)}{N(b_T, \tilde{V}_n)}.$$

Пусть $b_t : X \rightarrow \{-1; 1\}$. Тогда $P = 1 - N$.

Теорема (Freund, Shapire, 1995)

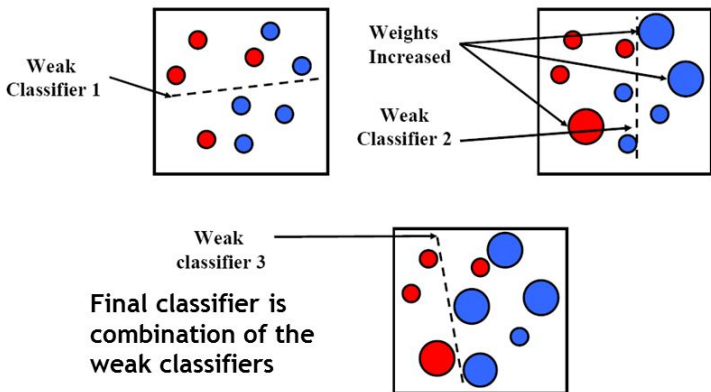
Пусть для любого нормированного вектора весов U_n существует алгоритм $b \in \mathcal{B}(\Theta)$, классифицирующий выборку немного лучше, чем наугад: $N(b, U_n) < \frac{1}{2}$.

Тогда минимум функционала \tilde{Q}_T достигается при

$$b_T = \operatorname{argmin}_{b \in \mathcal{B}} N(b, \tilde{V}_n),$$

$$\omega_T = \frac{1}{2} \ln \frac{1 - N(b_T, \tilde{V}_n)}{N(b_T, \tilde{V}_n)}.$$

Алгоритм AdaBoost



$$a(x) = \text{sign}(\omega_1 b_1(x) + \dots + \omega_T b_T(x)), \quad x \in X.$$

Вход: X^n , T — максимальное число базовых алгоритмов.

Выход: базовые алгоритмы b_t и их веса ω_t , $t = 1, \dots, T$.

1: $v_i := \frac{1}{n}$, $i = 1, \dots, n$;

2: **для всех** $t = 1, \dots, T$

3: обучить базовый алгоритм:

$$b_t := \operatorname{argmin}_{b \in \mathcal{B}} N(b, \tilde{V}_n);$$

$$4: \omega_t := \frac{1}{2} \ln \frac{1 - N(b_t, \tilde{V}_n)}{N(b_t, \tilde{V}_n)};$$

5: обновить веса объектов:

$$v_i := v_i \exp\{-\omega_t y_i b_t(x_i)\}, \quad i = 1, \dots, n;$$

6: нормировать веса объектов:

$$v_0 := \sum_{j=1}^n v_j;$$

$$v_i := \frac{v_i}{v_0}, \quad i = 1, \dots, n.$$

- Модификация формулы для ω_t на случай $N = 0$:

$$\omega_t := \frac{1}{2} \ln \frac{1 - N(b_t, \tilde{V}_n) + \frac{1}{n}}{N(b_t, \tilde{V}_n) + \frac{1}{n}}.$$

- Объекты с большими весами v_i — выбросы. Исключаем их и строим композицию заново.
- Требуемая длина обучающей выборки оценивается величиной порядка $10^4 \dots 10^6$.
- Базовые алгоритмы должны быть слабыми, из сильных хорошую композицию не построить:
 - Сильный алгоритм, давая нулевую ошибку на обучающих данных, не адаптируется и композиция будет состоять из одного базового алгоритма.
 - Один, даже сильный, алгоритм может дать «плохое» предсказание на данных тестирования, давая «хорошие» результаты на обучающих данных.

Обобщение бустинга. Gradient Boosting

Алгоритм бустинга обобщается на тот случай, когда пороговая функция потерь Q_T оценивается сверху произвольной невозрастающей функцией $\mathcal{L}(a, y)$.

b_t возвращают произвольные вещественные значения, не обязательно ± 1 , то есть $R = \mathbb{R}$.

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

Функционал качества:

$$Q_T \leq \tilde{Q}_T = \sum_{i=1}^n \mathcal{L} \left(y_i \underbrace{\sum_{t=1}^{T-1} \omega_t b_t(x_i)}_{f_{T-1,i}} + y_i \omega_T b_T(x_i) \right),$$

$\underbrace{\hspace{10em}}_{f_{T,i}}$

где $\mathcal{L}(a, y)$ — произвольная функция потерь,

$f_{T-1} = (f_{T-1,i})_{i=1}^n$ — текущее приближение,

$f_T = (f_{T,i})_{i=1}^n$ — следующее приближение.

Рассмотрим функцию потерь \mathcal{L} как функцию от параметра ω_T :

$$\lambda(\omega_T) := \mathcal{L}(f_{T-1,i} + y_i \omega_T b_T(x_i)).$$

Линеаризуем $\lambda(\omega_T)$ в окрестности $\omega_T = 0$, разложив в ряд Тейлора и отбросив старшие члены:

$$\lambda(\omega_T) \approx \lambda(0) + \omega_T \lambda'(0),$$

что приведет в линеаризации \tilde{Q}_T по параметру ω_T :

$$\tilde{Q}_T \approx \sum_{i=1}^n \mathcal{L}(f_{T-1,i}) - \omega_T \sum_{i=1}^n \underbrace{-\mathcal{L}'(f_{T-1,i}) y_i b_T(x_i)}_{v_i},$$

где v_i — веса объектов.

Для минимизации функционала качества \tilde{Q}_T ищут такой базовый алгоритм b_T , что $\{b_T(x_i)\}_{i=1}^n$ приближает вектор антиградиента $\{-\mathcal{L}(f_{T-1,i})\}_{i=1}^n$:

$$b_T := \operatorname{argmin}_{b \in \mathcal{B}} \sum_{i=1}^n \left(b(x_i) + \mathcal{L}'(f_{T-1,i}) \right)^2.$$

После построения b_T , параметр ω_T определяется путем одномерной минимизации функционала \tilde{Q}_T .

Итерации этих двух шагов приводят к обобщенному алгоритму бустинга AnyBoost.

► AnyBoost переходит в AdaBoost при $b_t : X \rightarrow \{-1, 1\}$ и $\mathcal{L}(f) = e^{-f}$

Вход: X^n, Y^n — обучающая выборка,

T — максимальное число базовых алгоритмов.

Выход: базовые алгоритмы b_t и их веса ω_t , $t = 1, \dots, T$.

1: $f_i := 0$, $i = 1, \dots, n$;

2: **для всех** $t = 1, \dots, T$

3: найти базовый алгоритм, приближающий градиент:

$$b_t := \operatorname{argmin}_{b \in \mathcal{B}} \sum_{i=1}^n (b(x_i) + \mathcal{L}'(f_i))^2;$$

4: $\omega_t := \operatorname{argmin}_{\omega > 0} \sum_{i=1}^n \mathcal{L}(f_i + y_i \omega b_t(x_i))$;

5: обновить значения f_i на объектах выборки:

$$f_i := f_i + \omega_t b_t(x_i) y_i, \quad i = 1, \dots, n.$$

Стохастический градиентный бустинг (SGB): на шагах 3-5 использовать не всю выборку X^n , а случайную подвыборку без возвращений.

- Градиентный бустинг — наиболее общий из всех бустингов:
 - произвольная функция потерь \mathcal{L} ,
 - произвольное пространство оценок R ,
 - подходит для регрессии, классификации, ранжирования.
- Хорошая обобщающая способность.
- Временная сложность построения композиции определяется временем обучения базовых алгоритмов.
- Простота реализации.
- Возможность идентифицировать выбросы.

- Жадная стратегия приводит к построению неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам и обучать их заново.
- Построение громоздких композиций, исключающих возможность содержательной интерпретации.
- Требуются большие ресурсы памяти для хранения базовых алгоритмов и существенные затраты времени на вычисление классификаций.