Композиция методов. Бустинг.

Романова Елизавета, Горбачук Анна, Сидоренко Денис

Санкт-Петербург
2019г.

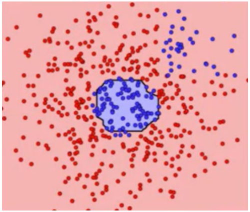
Случайный лес.

- композиция большого количества глубоких деревьев;
- базовые алгоритмы (базовые решающие деревья) независимы.

Проблемы:

- обучение глубоких деревьев — трудоемкая процедура (построение деревьев ненаправленное, нужно много деревьев для сложных задач);
- если ограничить глубину, деревья улавливают не все группы.

Пример: синий класс состоит из двух групп: одна в центре, одна с краю. Из-за того, что деревья очень небольшой глубины (в данном случае — 2), они могут уловить только одну из этих групп — ту, которая в центре, а на второй они полностью ошибаются.



Бустинг. Постановка задачи.

Бустинг:

- последовательное обучение базовых алгоритмов;
- каждый следующий исправляет ошибки предыдущих;
- за счет этого достаточно простых базовых алгоритмов.

Постановка задачи:
Пусть X — множество объектов, Y — множество ответов
$y : X \to Y$ — неизвестная зависимость.

Дано: обучающая выборка — $X^n = (x_i, y_i)_{i=1}^n$,
$y_i = y(x_i), i = 1, \ldots, n$ — известные ответы.

Требуется построить алгоритм $a(x) = C(b(x))$, аппроксимирующий целевую зависимость y на всём множестве X.

Изменение постановки задачи.

Замечание: вместо одного базового алгоритма b рассматривается несколько алгоритмов $b_1(x), \ldots, b_T(x)$.

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

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

В качестве базовых алгоритмов обычно выступают:

- решающие деревья (неглубокие 2-8) — используются чаще всего;
- пороговые правила (data stumps).

Задача: Подбор оптимальных (в смысле рассматриваемой функции потерь) базовых алгоритмов $\{b_t(x)\}_{t=1}^{T}$.

Простой пример для задачи регресии.

Хотим минимизировать среднеквадратичную ошибку

$$MSE(a, X) = \frac{1}{n}\sum_{i=1}^{n}(a(x_i) - y_i)^2.$$

- обучим простой алгоритм (неглубокое дерево):

  $b_1(x) = \arg\min\limits_{b \in \mathcal{B}} \frac{1}{n} \sum\limits_{i=1}^{n} (b(x_i) - y_i)^2;$
- хотим добавить еще один алгоритм b_2. Возникает вопрос: какие ответы b_2 должен давать на объектах обучающей выборки, чтобы ошибка нашей композиции была как можно меньше?

$$b_1(x_i) + b_2(x_i) = y_i \Rightarrow$$

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

$\ldots$

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

Понятие композиции алгоритмов.

Введем вспомогательное множество R, называемое пространством оценок. Будем рассматривать алгоритмы, имеющие вид суперпозиции $a(x) = C(b(x))$, где функция $b : X \to R$ называется **алгоритмическим оператором**, функция $C : R \to Y$ — **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 $C : R \to Y$:

$$a(x) = C(F(b_1(x), \dots, b_T(x))), \quad x \in X. \tag{1}$$

Суперпозиции вида $F(b_1, \dots, b_T)$ являются отображениями из X в R, то есть алгоритмическими операторами.

Пространство оценок. Примеры.

Пространство оценок R вводится для того, чтобы расширить множество допустимых корректирующих операций.

Пример 1. В задачах классификации на два класса, $Y = \{-1, +1\}$, в качестве пространства оценок обычно используется множество действительных чисел $R = \mathbb{R}$.
В этом случае алгоритмические операторы называют также вещественнозначными классификаторами: $C(b(x)) = sign b(x)$.

Пример 2. В задачах классификации на M классов, $Y = \{1, \ldots, M\}$, в качестве пространства оценок обычно используется $R = \mathbb{R}^M$. Алгоритмический оператор $b(x)$ выдаёт вектор оценок принадлежности объекта x каждому из классов, $b(x) = b^1(x), \ldots, b^M(x)$.
Решающее правило C относит объект к тому классу, для которого оценка максимальна: $C(b(x)) \equiv C(b^1(x), \ldots, b^M(x)) = \arg\max_{y \in Y} b^y(x)$.

Пример 3. В задачах регрессии множество Y уже достаточно богато, обычно $Y = \mathbb{R}$, поэтому использовать решающее правило нет особого смысла. В этом случае обычно полагают $R = \mathbb{R}$, $C(b) \equiv b$.

Примеры корректирующих операций.

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

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

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

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

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

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

Взвешенное голосование.

Корректирующая операция F может иметь параметры, настраиваемые по обучающей выборке, наряду с параметрами базовых алгоритмов. Например, в линейной комбинации настраиваются веса α_t базовых алгоритмов:

$$b(x) = F(b_1(x), \ldots, b_T(x)) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in R. \tag{2}$$

Если веса α_t неотрицательны и нормированы, $\sum_{t=1}^{T} \alpha_t = 1$, то композицию (2) называют **выпуклой комбинацией** базовых алгоритмов.

В задачах классификации корректирующая операция (2) называется **взвешенным голосованием** (weighted voting).

Бустинг в задачах классификации.

Рассмотрим задачу классификации на два класса, $Y = \{1, +1\}$. Допустим, что решающее правило фиксировано, $C(b) = sign(b)$, базовые алгоритмы возвращают ответы $-1, 0, +1$.

Ответ $b_t(x) = 0$ означает, что базовый алгоритм b_t отказывается от классификации объекта x, и ответ $b_t(x)$ не учитывается в композиции.

Искомая алгоритмическая композиция имеет вид:

$$a(x) = C(F(b_1(x), \dots, b_T(x)) = sign\left(\sum_{t=1}^{T} \alpha_t b_t(x)\right), \quad x \in X. \quad (3)$$

Функционал качества.

Определим функционал качества композиции как число ошибок, допускаемых ею на обучающей выборке:

$$Q_T = \sum_{i=1}^{n} \left[y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) < 0 \right]. \tag{4}$$

Для упрощения задачи минимизации функционала Q_T введём две эвристики (не полностью математически обоснованные, но при этом практически полезные алгоритмы).

Эвристика 1. При добавлении в композицию слагаемого $\alpha_t b_t(x)$ оптимизируется только базовый алгоритм b_t и коэффициент при нём α_t, а все предыдущие слагаемые $\alpha_1 b_1(x), \ldots, \alpha_{t-1} b_{t-1}(x)$ полагаются фиксированными.

Эвристика 2. Пороговая функция потерь в функционале Q_t аппроксимируется (заменяется) непрерывно дифференцируемой оценкой сверху.

Вторая эвристика широко используется в теории классификации.

При использовании экспоненциальной аппроксимации $[y_i b(x_i) < 0] \leq e^{y_i b(x_i)}$ эти две эвристики приводят к алгоритму AdaBoost.

Оценим функционал Q_T сверху:

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

$$= -\sum_{i=1}^{n} \underbrace{\exp\left(-y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)\right)}_{\omega_i} e^{y_i \alpha_T b_T(x_i)}.$$

Заметим, что введённые здесь веса объектов ω_i не зависят от $\alpha_T b_T$ и могут быть вычислены перед построением базового алгоритма b_T.

AdaBoost.

Введём вектор нормированных весов $\widetilde{W}^n = \widetilde{\omega}_1, \dots, \widetilde{\omega}_n$, где $\widetilde{\omega}_i = \omega_i / \sum_{j=1}^n \omega_j$.

Определим два функционала качества алгоритма классификации b на обучающей выборке $X^n = (x_i, y_i)_{i=1}^n$ с нормированным вектором весов объектов $U^n = (u_1, \dots, u_n)$: суммарный вес ошибочных (negative) классификаций $N(b; U^n)$ и суммарный вес правильных (positive) классификаций $P(b; U^n)$:

$$N(b; U^n) = \sum_{i=1}^n u_i [b(x_i) = -y_i],$$

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

Заметим, что $1 - N - P$ есть суммарный вес отказов от классификации. Если отказов нет, то $N + P = 1$.

Основная теорема бустинга (для AdaBoost).

Пусть $\mathcal{B}$ — достаточно богатое семейство базовых алгоритмов.

Теорема (Freund, Schapire, 1996)

Пусть для любого нормированного вектора весов U^n существует алгоритм $b \in \mathcal{B}$, классифицирующий выборку хотя бы немного лучше, чем наугад: $P(b; U^n) > N(b; U^n)$.
Тогда минимум функционала $\widetilde{Q}_T$ достигается при

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

$$a_t = \frac{1}{2} \ln \frac{P(b_t; \widetilde{W}^n)}{N(b_t; \widetilde{W}^n)}.$$

Алгоритм AdaBoost.

Вход: $X^n = (x_i, y_i)_{i=1}^n$ - 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:
   $\omega_i := 1/n$, $i = 1, \dots, n$;
2. для всех $t = 1, \dots, T$, пока не выполнен критерий остановки:
3. обучить базовый алгоритм:
   $b_t := \arg\min_{b \in \mathcal{B}} N(b; W^n)$;
4. $a_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^n)}{N(b_t; W^n)}$;
5. пересчет весов объектов:
   $\omega_i := \omega_i e^{\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, n$;
6. нормировка весов объектов:
   $\omega_0 := \sum_{j=1}^n \omega_j$; $\omega_i := \omega_i / \omega_0$, $i = 1, \dots, n$.

Достоинства AdaBoost.

- Хорошая обобщающая способность. В реальных задачах (не всегда, но часто) удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться (в некоторых задачах) по мере увеличения числа базовых алгоритмов.
- Простота реализации.
- Накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов.
- Возможность идентифицировать выбросы. Это "наиболее трудные"объекты x_i, для которых в процессе наращивания композиции веса ω_i принимают наибольшие значения.

Недостатки AdaBoost.

- AdaBoost склонен к переобучению при наличии значительного уровня шума в данных.
- AdaBoost требует достаточно длинных обучающих выборок.
- Бустинг может приводить к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти и существенных затрат времени.
- Жадная стратегия последовательного добавления приводит к построению неоптимального набора базовых алгоритмов.

Обобщение бустинга (AnyBoost).

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

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

Оценка сверху для числа ошибок композиции:

$$Q_T \leq \widetilde{Q}_T = \sum_{i=1}^{n} \mathscr{L}(M_{T-1}(x_i) + y_i \alpha_T b_T(x_i)) \to \min_{\alpha, b \in \mathcal{B}} .$$

Рассмотрим функцию потерь $\mathscr{L}$ как функцию параметра α_T,

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

и линеаризуем её в окрестности значения $\alpha_T = 0$, разложив в ряд Тейлора и отбросив старшие члены: $\lambda(\alpha_T) \approx \lambda(0) + \alpha_T \lambda'(0)$.
Это приведет к линеаризация функционала $\widetilde{Q}_T$ по α_T:

$$\widetilde{Q}_T \approx \sum_{i=1}^{n} \mathscr{L}(M_{T-1}(x_i)) - \alpha \sum_{i=1}^{n} \underbrace{-\mathscr{L}'(M_{T-1}(x_i))}_{\omega_i} y_i b(x_i) \to \min_{b \in \mathcal{B}},$$

где w_i — веса объектов.

Принцип явной максимизации отступов.

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

$$\widetilde{Q}_T \approx \sum_{i=1}^{n} \mathscr{L}(M_{T-1}(x_i)) - \alpha \sum_{i=1}^{n} \omega_i y_i b(x_i) \to \min_{b \in \mathcal{B}}$$

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

$$\sum_{i=1}^{n} \omega_i y_i b(x_i) \to \max_{b \in \mathcal{B}}.$$

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 $b_t : X \to \{-1, 0, +1\}$ и $\mathscr{L}(M) = e^{-M}$.

Алгоритм AnyBoost.

Вход: $X^n = (x_i, y_i)_{i=1}^n$ - обучающая выборка, T - максимальное число базовых алгоритмов.

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

1. инициализация отступов: $M_i := 0$, $i = 1, \dots, n$;
2. для всех $t = 1, \dots, T$, пока не выполнен критерий остановки:
3. вычислить веса объектов: $\omega_i = -\mathscr{L}'(M_i)$, $i = 1, \dots, n$;
4. обучить базовый алгоритм согласно принципу DOOM: $b_t := \arg\max\limits_{b \in \mathcal{B}} \sum_{i=1}^n \omega_i y_i b(x_i)$;
5. решить задачу одномерной минимизации: $a_t := \arg\max\limits_{\alpha} \sum_{i=1}^n \mathscr{L}(M_i + \alpha b_t(x_i) y_i)$;
6. пересчет отступов: $M_i := M_i + \alpha b_t(x_i) y_i$; $i = 1, \dots, n$.