Санкт-Петербургский государственный университет
Прикладная математика и информатика
Статистическое моделирование

Григорьева Ирина Владимировна, гр. 622

Композиция методов. Бустинг.

Конспект

Санкт-Петербург
2017

1. Введение

При решении сложных задач классификации, регрессии часто оказывается, что ни один из алгоритмов не обеспечивает желаемого качества восстановления зависимости. В таких случаях имеет смысл строить композиции алгоритмов, в которых ошибки алгоритмов взаимно компенсируются.

2. Постановка задачи

Рассмотрим задачу обучения: $< X, Y, f, X^n >$, где

- X — пространство объектов, Y — множество ответов,
- $f : X \to Y$ — 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

- $b : X \to R$ — базовый алгоритм (алгоритмический оператор),
- $C : R \to Y$ — решающее правило,
- R — пространство оценок.

Наряду с X и Y вводится вспомогательное множество R, чтобы расширить множество допустимых корректирующих операций.

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 решающее правило не нужно $C(b) = b$, так как регрессия дает богатое множество допустимых элементов на выходе Y.

3. Изменение постановки задачи

Вместо одного базового алгоритма b рассматривается несколько алгоритмов $b_1, \dots, b_T$.
$\mathcal{B}(\Theta) = \{b(\cdot;\theta) | \theta \in \Theta\}$ — параметризованное множество базовых алгоритмов,
Выбор базового алгоритма: выбор $\theta \in \Theta$ и $b(x) = b(x;\theta) \in \mathcal{B}(\Theta)$.
В качестве базовых алгоритмов обычно выступают:
— решающие деревья (неглубокие 2-8) — используются чаще всего;
— пороговые правила (data stumps): $b(x, \theta = \{i, s\}) = [f_i(x) \lesseqgtr s]$.

Определение 1. *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 $F : R^{\mathrm{T}} \to R$ *— корректирующая операция, параметризованная с помощью* $\omega \in \Omega$ *(агрегирует значения базовых алгоритмов).*

Например, F может быть линейной функцией, то есть $\omega = (\omega_1, \dots, \omega_T) \in \mathbb{R}^{\mathrm{T}}$ и

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

F может иметь параметры, настраиваемые по обучающей выборке наряду с параметрами базовых алгоритмов: ω — веса, которые указывают, в какой степени мы доверяем алгоритмам (часто ошибаются – меньше вес).
Задача: Подбор оптимальных (в смысле рассматриваемой функции потерь) параметров ω и базовых алгоритмов $\{b_t(x)\}_{t=1}^{T}$.

Примеры пространства оценок и решающих правил:

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

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

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

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

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

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

Примеры корректирующих операций:

- Простое голосование:

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

- Взвешенное голосование:

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

4. Сравнение композиционных методов

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).

Используется весовая версия одних и тех же обучающих данных вместо случайного выбора подмножества. Большие веса назначаются объектам, которые были плохо классифицированы предыдущими алгоритмами, что позволяет на каждой итерации сосредоточится на этих наблюдениях.

5. Boosting

Основная идея:

Заметим, что оптимизация функции потерь происходит по многомерному множеству параметров, поэтому точная многомерная оптимизация не выглядит перспективной. Концептуальная идея Boosting заключается в использовании жадной стратегии оптимизации. Неформально идею можно изложить следующим образом:

Пусть для некоторого фиксированного $T_0 < T$ уже выбрали алгоритмы $\{b_t(x)\}_{t=1}^{T_0}$ и параметры корректирующей операции $\{\omega_t\}_{t=1}^{T_0}$. На (T_0+1) шаге выбираем параметр ω_{T_0+1} и базовый алгоритм b_{T_0+1} так, чтобы исправить ошибки композиции, основанной лишь на первых T_0 алгоритмах

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

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

Обратимся к следующему примеру, чтобы рассмотреть работу жадного алгоритма.

Пример работы бустинга для 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

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

Обучим простой 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 $y_i - b_1(x_i),\ i = 1, \ldots, n.$

Обучаем b_2 так, чтобы его прогноз был близок к поправке (скорее всего точно решить задачу не получится и b_2 будет чуть улучшать качество b_1)

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

$$\ldots$$

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

6. AdaBoost

Рассмотрим задачу классификации на два класса $Y = \{-1, 1\}$ и определим решающее правило $C(b) = \mathrm{sign}(b)$, тогда $b_t : X \to \{-1, 0, 1\}$, $b_t \in \mathcal{B}(\Theta)$, где $b_t(x) = 0$ — отказ базового алгоритма от классификации объекта x.

Положим

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

Тогда

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

Функционал качества композиции — число ошибок на обучающей выборке X^n:

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

Прямая оптимизация такого функционала является затруднительной, поэтому Q_T мажорируется некоторой непрерывно дифференцируемой функцией, поддающейся эффективной оптимизации. Выбор функции зависит от характера задачи. На Рис. 1 представлены различные варианты гладких мажорант.

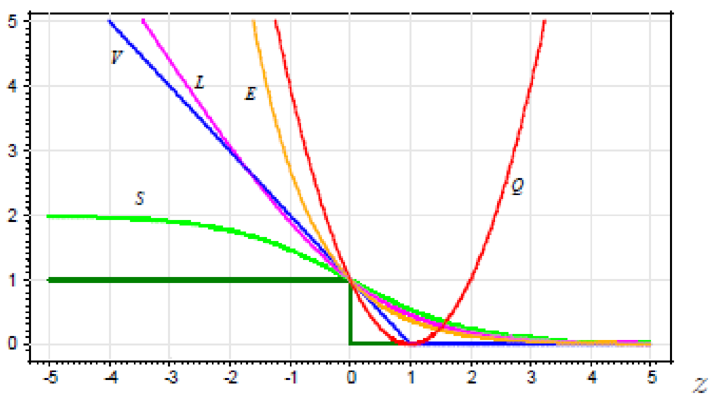

Рис. 1. Гладкие аппроксимации пороговой функции потерь [z<0]:

$S(z) = 2(1 + \exp(z))^{-1}$ — 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.

Например, логарифмическая функция связана с принципом максимума правдоподобия и применяется в логистической регрессии, кусочно линейная-аппроксимация связана с принципом максимизации зазора между классами и применяется в методе опорных векторов. Все эти функции можно использовать, получая различные варианты бустинга.

Рассмотри алгоритм AdaBoost, который использует экспоненциальную аппроксимацию. Оценка функционала Q_T сверху имеет вид:

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

Заметим, что введенные веса v_i не зависят от ω_T и b_T и могут быть вычислены перед построением b_T.

Введем вектор нормированных весов объектов $\tilde{V}_n = (\tilde{v}_1, \ldots, \tilde{v}_n),\ \tilde{v}_i = v_i / \sum_{j=1}^{n} v_j$.

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

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

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

Теорема 1 (**Основная теорема бустинга, Freund, Shapire, 1996**). *Пусть для любого нормированного вектора весов U_n существует алгоритм $b \in \mathcal{B}(\Theta)$ (заранее заданное богатое семейство базовых алгоритмов), классифицирующий выборку немного лучше, чем наугад: $P(b, U_n) > N(b, U_n)$.*

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

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

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

Рассмотрим важный частный случай, когда базовые алгоритмы не отказываются от классификации.

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

Теорема 2 (Классический вариант AdaBoost, 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

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

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

На Рис. 2 наглядно изображена работа алгоритма AdaBoost. Первый слабый базовый алгоритм как-то разбивает выборку на два класса, увеличивается вес неправильно классифицируемых объектов. Второй алгоритм, классифицируя, пытается исправить ошибки предыдущего, снова увеличивается вес неправильно классифицируемых объектов и. т. д. Финальный классификатор — композиция слабых классификаторов

$$a(x) = \operatorname{sign}(\omega_1 b_1(x) + \ldots + \omega_T b_T(x)),\ x \in X.$$

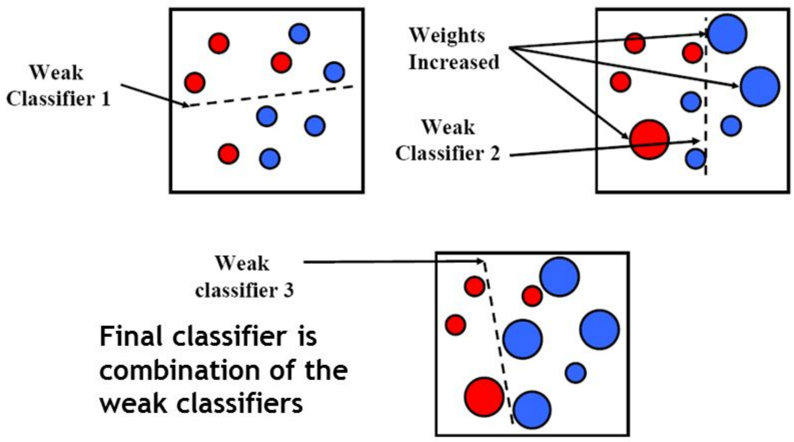

Рис. 2. Алгоритм AdaBoost.

Алгоритм AdaBoost

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: $b_t := \mathrm{argmin}_{b \in \mathcal{B}} N(b, \tilde{V_n});$

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

Вес увеличится в e^{ω_t} раз, когда b_t ошибается, и уменьшается во столько же раз, когда b_t классифицирует правильно.

6: нормировать веса объектов: $v_0 := \sum_{j=1}^{n} v_i$; $v_i := \frac{v_i}{v_0},\ i = 1, \dots, n.$

Рекомендации

- Модификация формулы для ω_t на случай $N = 0$ (нет ошибок на обучающей выборке), чтобы вес не уходил на бесконечность:

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

- Экспоненциальная функция потерь сильно увеличивает веса v_i трудно распознаваемых объектов, а именно такие объекты чаще всего оказываются выбросами. После построения некоторого количества b_t, нужно посмотреть на распределение весов, исключить объекты с большими весами из рассмотрения и построить 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.

7. Обобщение бустинга. Gradient Boosting

Исторически сложилось, что первым появился алгоритм AdaBoost, который использует экспоненциальную аппроксимацию. Теперь рассмотрим общий случай, когда пороговая функция потерь Q_T оценивается сверху произвольный невозрастающей функцией $\mathcal{L}(a, y)$. Базовые алгоритмы b_t возвращают произвольные вещественные значения, не обязательно ± 1, то есть $R = \mathbb{R}$.

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

Функционал качества имеет вид:

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

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:

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

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 $\{-\mathcal{L}(f_{T-1,i})\}_{i=1}^n$:

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

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.

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

Алгоритм AnyBoost

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:

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

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

5: обновить значения f_i на объектах выборки:

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

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

8. Заключение

Достоинства

- 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 (бустинг можно использовать как универсальный метод фильтрации выбросов перед применением любого другого метода классификации)

Недостатки

- Жадная стратегия приводит к построению неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам и обучать их заново.

- Бустинг может приводить к построению композиций, состоящих из сотен алгоритмов. Такие композиции решают поставленную задачу, но исключают содержательную интерпретацию.

- Требуются большие ресурсы памяти для хранения базовых алгоритмов и существенные затраты времени на вычисление классификаций.