Композиции классификаторов и случайный лес

К. В. Воронцов, А.В. Зухба
vokov@forecsys.ru
a__l@mail.ru

март 2016

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 $b_1, \dots, b_T$

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

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):

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

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

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

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

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

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], причём подвыборки берутся длины ℓ с возвращениями, как в методе bootstrap.

Вторую идею реализует RSM (random subspace method) [Duin, 2002].

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, $t = 1, \ldots, T$;

1: **для всех** $t = 1, \ldots, T$
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: $a(x) = C\left(\sum_{t=1}^{T} b_t(x)\right)$.

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.
- Не удаётся строить короткие композиции из «сильных» алгоритмов типа SVM (только длинные из слабых).

Возможно ли строить композиции проще и аккуратнее?

Простое голосование в задаче классификации

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

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.

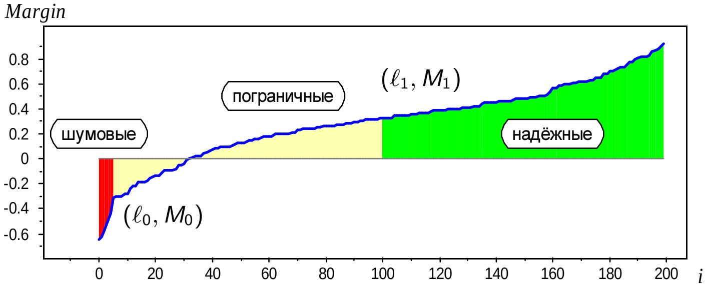
Эвристика: чтобы b_{t+1} компенсировал ошибки композиции,

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

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.

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: $\sum_{t=1}^{T} |\alpha_t| \leqslant \varkappa$.

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.

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: $\mathbb{R}^T \to \mathbb{R}^T$:

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

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 $\forall\, y \in Y,\ \forall\, b_1, b_2 \in R,\ \forall\, g_1, g_2 \geqslant 0$: $g_1 + g_2 = 1$, 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:

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

Итерационный процесс, аналогичный ЕМ-алгоритму:

1: начальное приближение функций компетентности g_t;
2: **повторять**
3: **М-шаг:** при фиксированных g_t обучить все b_t;
4: **Е-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:

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

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

5: $b_t := \arg\min\limits_b \sum\limits_{x_i \in X_t} \mathscr{L}\bigl(b(x_i), y_i\bigr)$;

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

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

Резюме

- Обучение смесей алгоритмов основано на принципе «разделяй и властвуй».
- Смеси алгоритмов имеет смысл строить в тех задачах, где есть априорные соображения о виде областей компетентности.
- Кроме последовательного метода построения смесей алгоритмов, известен ещё иерархический.