

# Лекция 7. Композиции алгоритмов

## Основы интеллектуального анализа данных

Полузёров Т. Д.

БГУ ФПМИ

## 1 Общие принципы

- Композиция
- Агрегирующая функция

## 2 Беггинг

- Смещение и разброс
- Некоррелированность алгоритмов
- Случайный лес

## 3 Бустинг

- Градиентный бустинг

# Общие идеи

Имеется размеченная выборка  $X^\ell = (x_i, y_i)_{i=1}^\ell \subset \mathbb{X} \times \mathbb{Y}$ .

Задача — приблизить истинную зависимость  $y^* : \mathbb{X} \rightarrow \mathbb{Y}$

Ранее мы строили один сильный алгоритм  $a(x) := C(b(x))$ , такой что  $\mathbb{X} \xrightarrow{b} \mathbb{R} \xrightarrow{C} \mathbb{Y}$ .

Быть может лучше использовать несколько базовых алгоритмов  $(b_i(x))_{i=1}^k$ , а затем учесть ответы их всех

$a(x) = C(F(b_1(x), \dots, b_k(x)))$ . То есть  $\mathbb{X} \xrightarrow{b} \mathbb{R}^k \xrightarrow{F} \mathbb{R} \xrightarrow{C} \mathbb{Y}$

Здесь  $C$  — решающее правило,  $F$  — агрегирующая операция.

Для наших целей неважно как именно реализованы базовые алгоритмы, они лишь должны поддерживать интерфейс

- $fit(X, Y)$  — обучиться по выборке  $X = (x, y)$
- $predict(X)$  — выдать прогнозы  $\hat{y} = b(X)$

Мы же сфокусируемся на построении композиции и улучшении итогового качества. А именно: выбрать агрегирующую функцию и предложить метод обучения композиции.

Общие требования к  $F$ :

- $\min b_i \leq F(b_1, \dots, b_k) \leq \max b_i$
- $F(b_1(x), \dots, b_k(x))$  монотонно не убывает по всем  $b_i$

# Примеры агрегирующих функций

- простое голосование

$$F(b_1, \dots, b_k) = \frac{1}{k} \sum_{i=1}^k b_i(x)$$

- взвешенное голосование,  $\sum_{i=1}^k \alpha_i = 1, \alpha_i \geq 0$

$$F(b_1, \dots, b_k) = \sum_{i=1}^k \alpha_i b_i(x)$$

- смесь алгоритмов с функциями компетентности  $g : \mathbb{X} \rightarrow \mathbb{R}$

$$F(b_1, \dots, b_k) = \sum_{i=1}^k g_i(x) b_i(x)$$

# Функционал риска

Рассмотрим задачу регрессии с квадратичной функцией потерь. Качество алгоритма  $a$ :

$$Q(a) = \mathbb{E}_x \mathbb{E}_{X,\varepsilon} [y(x, \varepsilon) - a(x, X)]^2$$

- $X$  — обучающая выборка
- $y = f(x) + \varepsilon$  — целевая зависимость, наблюдаемая с точностью до шума  $\varepsilon$
- $a(x, X)$  — ответ алгоритма в точке  $x$ , обученного по выборке  $X$
- $\mathbb{E}_x$  — среднее по всем тестовым точкам,
- $\mathbb{E}_{X,\varepsilon}$  — среднее по всем обучающим выборкам и случайному шуму

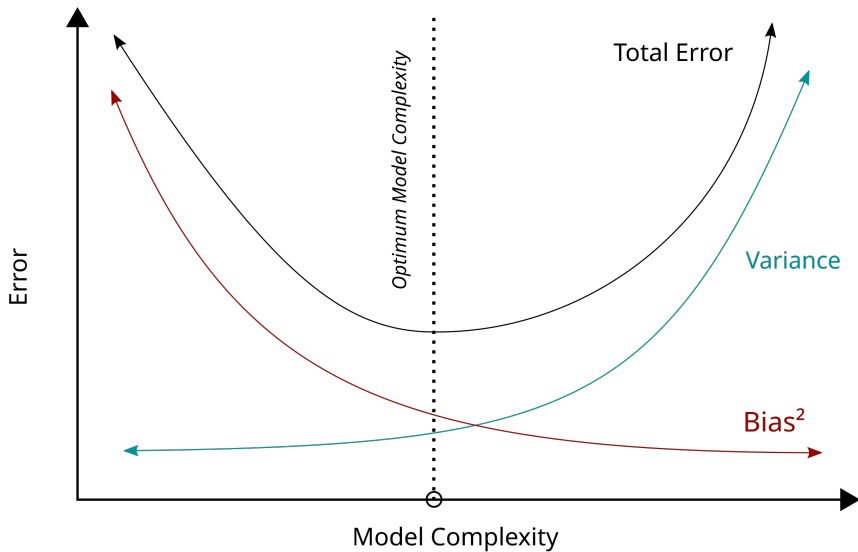
# Смещение и разброс

Ошибку можно представить в виде трех слагаемых:

$$Q(a) = \mathbb{E}_x \text{bias}_X^2(a) + \mathbb{E}_x \text{variance}_X(a) + \sigma^2$$

где

- $\text{bias}_X(a) = f(x) - \mathbb{E}_X[a(x, X)]$  — смещение
- $\text{variance}(a) = \mathbb{E}_X[a(x, X) - \mathbb{E}_x[a(x, X)]]^2$  — разброс
- $\sigma^2 = \mathbb{E}_x \mathbb{E}_\varepsilon[y(x, \varepsilon) - f(x)]^2$  — неустранимый шум





# Ошибка простого голосования

Рассмотрим композицию — простое голосование

$$a(x) = \frac{1}{k} (b_1(x) + \cdots + b_k(x))$$

Как зависит смещение и разброс композиции от базового алгоритма  $b$ ?

# Смещение композиции

$$\begin{aligned} \text{bias}_X(a) &= f(x) - \mathbb{E}_X[a(x, X)] = \\ &= f(x) - \mathbb{E}_X \left[ \sum_{i=1}^k \frac{1}{k} b(x, X^i) \right] = \\ &= f(x) - \mathbb{E}_X b(x, X) = \\ &= \text{bias}_X(b) \end{aligned}$$

Смещение ансамбля определяется смещением базового алгоритма.

# Разброс композиции

$$\begin{aligned} \text{variance}_X(a) &= \mathbb{E}_X[a(x, X) - \mathbb{E}_x[a(x, X)]]^2 = \\ &= \mathbb{E}_X \left[ \frac{1}{k} \sum_{i=1}^k b_i - \mathbb{E}_X \left[ \frac{1}{k} \sum_{i=1}^k b_i \right] \right]^2 = \\ &= \frac{1}{k^2} \mathbb{E}_X \left[ \sum_{i=1}^k (b_i - \mathbb{E}_X b_i) \right]^2 = \\ &= \frac{1}{k^2} \sum_{i=1}^k \text{variance}_X(b_i) + \frac{1}{k^2} \sum_{i \neq j} \text{cov}(b_i, b_j) \end{aligned}$$

# Некоррелированные базовые алгоритмы

В случае если базовые алгоритмы некоррелированы, то

$$\begin{aligned} \text{variance}_X(a) &= \frac{1}{k^2} \sum_{i=1}^k \text{variance}_X(b_i) = \\ &= \frac{1}{k^2} \sum_{i=1}^k \text{variance}_X(b) = \\ &= \frac{1}{k} \text{variance}_X(b) \end{aligned}$$

Используя некоррелированную композицию алгоритмов, можно добиться уменьшения разброса в  $k$  раз!

## Как добиться некоррелированности?

На практике строгое выполнение требования некоррелированности — необязательно, достаточно, чтобы базовые алгоритмы были непохожи друг на друга.

Два способа увеличить непохожесть, строить  $b_i(x)$ :

- по случайной подвыборке объектов
- по псевдовыборке исходных объектов — **бутстрап**
- по случайному подмножеству признаков — **метод случайных подпространств**

# Бутстреп

Пусть имеется выборка  $X$  объема  $\ell$ .

С помощью случайного **выбора с возвращением** сформируем новую «выборку»  $X^1$  тоже объема  $\ell$ . Полученную подвыборку называют **псевдовыборкой**.

Прделаем эту операцию  $k$  раз и получим  $\{X^1, \dots, X^k\}$  псевдовыборок.

Такой метод получения псевдовыборок называется **бутстрепом** (bootstrap).

Применяется в статистике для проверки гипотез, построения доверительных интервалов ...

# Бэггинг

Беггинг (bootstrap aggregation) — простое голосование, где при построении базовых алгоритмов используется бутстрап и метод случайных подпространств.

Можно внести модификации:

- ввести порог  $\alpha$  и не добавлять  $b_i$  если не достигает нужного качества
- обучение  $b_i$  идет не по всей выборке  $U_i \subset X^\ell$  — можно оценить качество на невиданных данных  $X^\ell \setminus U_i$

# Общий алгоритм обучения бэггинга

**Data:** обучающая выборка  $X^\ell$ ,

$k$  — длина композиции,

$\alpha$  — порог качества

**Result:** базовые алгоритмы  $b_1, \dots, b_k$

**for**  $i = 1, \dots, k$  **do**

$U_i :=$  случайная подвыборка размера  $\ell'$

$G_i :=$  случайное подмножество признаков размера  $n'$

    обучить  $b_i$  по  $G_i$  и  $U_i$

**if**  $Q(b_i, U_i) < \alpha$  **then**

        | добавить  $b_i$  в композицию

**end**

    оценить качество  $Q(b_i, X^\ell \setminus U_i)$  на отложенных данных

**end**



# Случайный лес

Случайный лес (Random Forest) — бэггинг над решающими деревьями.

Остаются вопросы

- Какой глубины  $h$  строить деревья?
- Сколько признаков  $n'$  использовать для обучения?
- Сколько деревьев  $k$  использовать в композиции?

## Глубина деревьев

Ошибка состоит из смещения и разброса. Разброс снижается за счет композиции, а смещение определяется смещением базового дерева. Поэтому необходимо использовать **деревья с низким смещением**.

- Неглубокие деревья имеют малое число параметров и поэтому строят только верхнеуровневые зависимости. При разных обучающих подвыборках алгоритмы не будут значительно отличаться (низкий разброс, высокое смещение).
- Глубокие деревья наоборот чересчур сильно подстраиваются под данные, поэтому имеют сильный разброс, но низкое смещение.

Используем **глубокие деревья**.

# Число признаков

Большое число признаков обеспечивает слабое различие деревьев, поэтому эффект от бэггинга — слабый.

С другой стороны, используя малое число признаков, базовые деревья будут слабыми.

Практическая рекомендация:

- Для задач регрессии —  $n' = \lfloor n/3 \rfloor$
- Для задач классификации —  $n' = \lfloor \sqrt{n} \rfloor$

## Размер ансамбля

С ростом числа деревьев снижается разброс, но число признаков и вариантов подвыборки — ограничены. Поэтому бесконечно уменьшать разброс не получится.

Имеет смысл построить график зависимости ошибки от числа деревьев и остановиться в тот момент, когда ошибка перестанет значительно уменьшаться

Так же ограничением может выступать время работы ансамбля. Большее число деревьев — большее время принятия решения. Однако, алгоритмы стоятся независимо друг от друга и это позволяет распаралеливать построение отдельных деревьев.

## Переход к взвешенному голосованию

Пусть уже построен набор базовых алгоритмов  $b_1, \dots, b_k$ .

Взвешенная композиция

$$a(x) = \sum_{i=1}^k \alpha_i b_i(x)$$

есть ни что иное как линейная модель с новыми признаками — ответами базовых моделей.

Настройка параметров

$$Q(\alpha) = \sum_{i=1}^{\ell} \mathcal{L}(a(x_i, \alpha), y_i) \rightarrow \min_{\alpha}$$

# Идея бустинга

За основу возьмем взвешенное голосование

$$a(x) = \sum_{i=1}^k \alpha_i b_i(x)$$

Задан функционал качества

$$Q(a) = \sum_{i=1}^{\ell} \mathcal{L}(a(x_i), y_i)$$

Основная идея — строить последовательность  $b_1, \dots, b_k$  жадным способом. На итерации  $t$  считаем вычисленными и зафиксированными  $\alpha_1 b_1, \dots, \alpha_{t-1} b_{t-1}$ .

# Градиентный бустинг

Рассмотрим наиболее общий алгоритм бустинга — градиентный бустинг.

Градиентный спуск для скалярной функции  $Q(a) \rightarrow \min$  в пространстве ответов на обучающей выборке.

Строится последовательность точек

$$a_t = a_{t-1} - \alpha_t \frac{\partial Q}{\partial a} \Big|_{a=a_{t-1}}$$

Схоже с добавлением нового базового алгоритма на каждой итерации.

Будем строить композицию, где  $b_t$  приближает  $\frac{\partial Q}{\partial a}$  в точке  $a_{t-1}$

$$b_t(x) \approx - \frac{\partial Q}{\partial a} \Big|_{a=a_{t-1}}$$

# Приближение антиградиента

Приближение антиградиента можно сформулировать так

$$g_{t-1,i} = \left( -\frac{\partial Q}{\partial a} \Big|_{a=a_{t-1}} \right)_i$$

—  $i$ -я компонента градиента, вычисленного в точке  $a_{t-1}$  на всех объектах выборки

$$b_t(x) = \arg \max_{b \in B} \sum_{i=1}^{\ell} (b(x_i) - g_{t-1,i})^2$$

То есть  $b_t$  решает задачу регрессии, приближая антиградиент функции потерь.



# Алгоритм градиентного бустинга

**Data:** обучающая выборка  $X^\ell$ ,

**Result:** базовые алгоритмы и веса  $\alpha_1 b_1, \dots, \alpha_k b_k$

Построить первый алгоритм  $b_1(x)$

**for**  $t = 2, \dots, k$  **do**

    Вычислить

$$g := -\frac{\partial Q}{\partial a} \Big|_{a=a_{t-1}}$$

    Обучить алгоритм  $b: x \rightarrow g$

$$b_t := \arg \min_b \sum_{j=1}^{\ell} (b(x_j) - g)^2$$

    Подобрать  $\alpha_t$

$$\alpha_t := \arg \min_{\alpha > 0} \sum_{j=1}^{\ell} L(a_{t-1} + \alpha b_t(x_j), y_j)$$

**end**

# Заключение

- Композиции позволяют решать сложные задачи
- Почти всегда базовые алгоритмы — решающие деревья
- Строить композиции целиком — сложно. Поэтому базовые алгоритмы строятся по одному
- Построение беггинга можно распаралелить. Бустинга — нет
- Обобщающая способность бустинга не ухудшается с ростом композиции