Машинное обучение

Лекция 8. Бэггинг, bias-variance разложение и случайные леса

Автор: Рустам Азимов

Санкт-Петербургский государственный университет

Санкт-Петербург

Решающие деревья

- Решающие деревья — семейство моделей, которые позволяют восстанавливать нелинейные зависимости произвольной сложности
- Но неустойчивы к малейшим изменениям в данных и сами по себе деревья не очень хороши
- Зато показывают очень хорошие результаты при объединении в композицию

Ансамбли

- Бэггинг — независимо и параллельно обучаем несколько простых моделей (weak learners) и усредняем их ответы
- Бустинг — обучаем простые модели последовательно, и каждая следующая модель исправляет ошибки предыдущей
- Stacking — параллельно обучаем разнообразные простые модели и объединяем их, обучая новую мета-модель, которая получает на вход предсказания простых моделей

Бутстрап

- Рассмотрим простой пример построения композиции алгоритмов
- Пусть дана конечная выборка $X = \{(x_i, y_i)\}$, где $y_i \in \mathbb{R}$
- Будем решать задачу линейной регрессии и сгенерируем подвыборку с помощью **бутстрапа**
- Равномерно возьмем из выборки l объектов с возвращением
- Получим выборку X_1. Сделаем так N раз и получим N подвыборок $X_1, \dots, X_N$
- Обучим по каждой из них линейную модель регрессии, получив базовые алгоритмы $b_1(x), \dots, b_N(x)$

Бутстрап

- Предположим, что существует истинная функция ответа для всех объектов $y(x)$, а также задано распределение на объектах $p(x)$
- Тогда ошибка каждой функции регрессии имеет вид

$$\varepsilon_j(x) = b_j(x) - y(x)$$

- А матожидание среднеквадратичной ошибки

$$\mathbb{E}_x(b_j(x) - y(x))^2 = \mathbb{E}_x \varepsilon_j^2(x)$$

- Тогда средняя ошибка построенных функций регрессии имеет вид

$$E_1 = \frac{1}{N} \sum_{j=1}^{N} \mathbb{E}_x \varepsilon_j^2(x)$$

Бутстрап

- Предположим, что ошибки несмещены и некоррелированы: $\mathbb{E}_x \varepsilon_j(x) = 0$ и $\mathbb{E}_x \varepsilon_i(x) \varepsilon_j(x) = 0$ при $i \neq j$
- Построим теперь новую функцию регрессии, которая будет усреднять ответы построенных нами функций

$$a(x) = \frac{1}{N} \sum_{j=1}^{N} b_j(x)$$

- Её среднеквадратичная ошибка тогда такая

$$E_N = \mathbb{E}_x \Big(\frac{1}{N} \sum_{j=1}^{N} \varepsilon_j(x)\Big)^2 = \frac{1}{N} E_1$$

Бутстрап

- Таким образом, усреднение ответов позволило уменьшить средний квадрат ошибки в N раз
- Но рассмотренный нами пример не очень применим на практике, поскольку мы сделали предположение о некоррелированности ошибок, что редко выполняется
- В таком случае уменьшение ошибки оказывается не таким значительным

Bias-Variance decomposition

- Ошибка любой модели складывается из трех факторов:
  - сложности самой выборки
  - сходства модели с истинной зависимостью ответов от объектов в выборке
  - богатства семейства, из которого выбирается конкретная модель
- Между этими факторами существует некоторый баланс, и уменьшение одного из них приводит к увеличению другого
- Такое разложение ошибки носит название разложения на смещение и разброс

Bias-Variance decomposition

- Будем считать, что на пространстве всех объектов и целевых признаков $\mathbb{X} \times \mathbb{Y}$ существует распределение $\rho(x, y)$, из которого и сгенерирована выборка X и целевые признаки для нее
- Для того, чтобы построить идеальную функцию регрессии, необходимо знать это распределение $\rho(x, y)$, что, как правило, невозможно
- На практике вместо этого выбирается некоторый метод обучения $\mu : (\mathbb{X} \times \mathbb{Y})^l \to \mathcal{A}$, который произвольной обучающей выборке ставит в соответствие некоторый алгоритм из семейства $\mathcal{A}$
- В качестве меры качества метода обучения можно взять усредненный по всем выборкам среднеквадратичный риск алгоритма, выбранного методом μ по выборке:

$$L(\mu) = \mathbb{E}_X[\mathbb{E}_{x,y}[(y - \mu(X)(x))^2]]$$

Bias-Variance decomposition

- После преобразований можем получить следующее:

$$L(\mu) = \underbrace{\mathbb{E}_{x,y}\Big[\big(y - \mathbb{E}[y \mid x]\big)^2\Big]}_{\text{шум}} +$$

$$+ \underbrace{\mathbb{E}_{x}\Big[\big(\mathbb{E}_X\left[\mu(X)\right] - \mathbb{E}[y \mid x]\big)^2\Big]}_{\text{смещение}} + \underbrace{\mathbb{E}_{x}\Big[\mathbb{E}_X\Big[\big(\mu(X) - \mathbb{E}_X\left[\mu(X)\right]\big)^2\Big]\Big]}_{\text{разброс}}$$

Bias-Variance decomposition

- Первая компонента характеризует **шум** в данных и равна ошибке идеального алгоритма (невозможно построить алгоритм, имеющий меньшую среднеквадратичную ошибку)
- Вторая компонента характеризует **смещение (bias)** метода обучения, то есть отклонение среднего ответа обученного алгоритма от ответа идеального алгоритма
- Третья компонента характеризует **дисперсию (variance)**, то есть разброс ответов обученных алгоритмов относительно среднего ответа

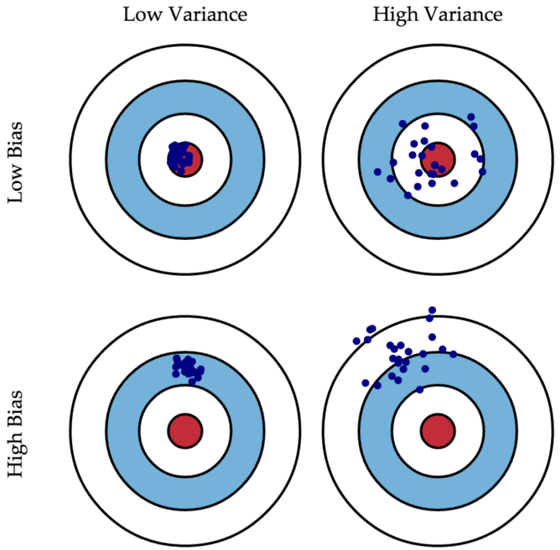
Смещение

- Смещение показывает, насколько хорошо с помощью данных метода обучения и семейства алгоритмов можно приблизить оптимальный алгоритм
- Как правило, смещение маленькое у сложных семейств (например, у деревьев)
- И большое у простых семейств (например, линейных классификаторов)

Дисперсия

- Дисперсия показывает, насколько сильно может изменяться ответ обученного алгоритма в зависимости от выборки
- Она характеризует чувствительность метода обучения к изменениям в выборке
- Как правило, простые семейства имеют маленькую дисперсию, а сложные семейства — большую дисперсию

Иллюстрация смещения и разброса для различных моделей



Смещение и разброс для различных моделей

- Большой сдвиг соответствует тому, что в среднем точки не попадают в центр, то есть в среднем они не соответствуют лучшей модели
- Большой разброс означает, что модель может попасть по качеству куда угодно — как в центр, так и в область с большой ошибкой
- Мы рассмотрели декомпозицию на шум, смещение и разброс только для квадратичной функции потерь
- Для большинства распространённых функций потерь такие рассуждения также верны и ошибка метода обучения складывается из аналогичных трех компонент

Бэггинг

- Пусть имеется некоторый метод обучения $\mu(X)$
- Построим на его основе метод $\tilde{\mu}(X)$, который генерирует случайную подвыборку $\tilde{X}$ с помощью бутстрапа и подает ее на вход метода $\mu: \tilde{\mu}(X) = \mu(\tilde{X})$
- Сэмплирование с возвращениями поэтому помещение нескольких копий одного объекта в бутстрапированную выборку соответствует выставлению веса при данном объекте
- Соответствующее ему слагаемое несколько раз войдет в функционал, и поэтому штраф за ошибку на нем будет больше

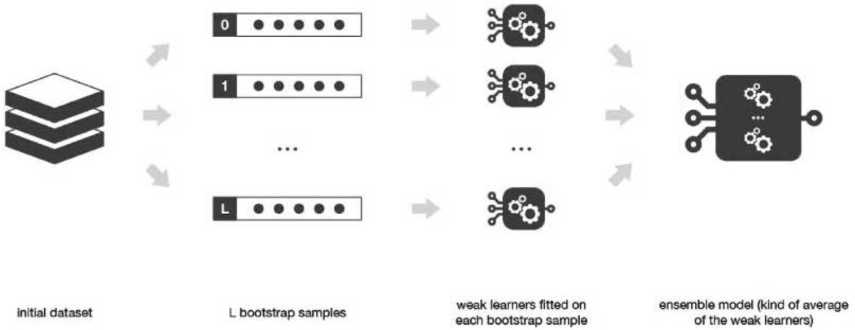
Бэггинг

- В **бэггинге (bagging, bootstrap aggregation)** предлагается обучить некоторое число алгоритмов $b_n(x)$ с помощью метода $\tilde{\mu}$ и построить итоговую композицию как среднее данных базовых алгоритмов:

$$a_N(x) = \frac{1}{N} \sum_{n=1}^{N} b_n(x) = \frac{1}{N} \sum_{n=1}^{N} \tilde{\mu}(X)(x)$$

- Если посчитать смещение композиции, полученной с помощью бэггинга, то получится что она совпадает со смещением одного базового алгоритма
- Таким образом, бэггинг не ухудшает смещенность модели

Бэггинг


0
1
...
L
...
initial dataset
L bootstrap samples
weak learners fitted on each bootstrap sample
ensemble model (kind of average of the weak learners)

Бэггинг

- Если посчитать разброс композиции, то получится что это сумма дисперсии одного базового алгоритма, деленная на длину композиции N и ковариации между двумя базовыми алгоритмами
- Поэтому если базовые алгоритмы некоррелированы, то дисперсия композиции в N раз меньше дисперсии отдельных алгоритмов
- Если же корреляция имеет место, то уменьшение дисперсии может быть гораздо менее существенным

Случайные леса

- Как мы выяснили, бэггинг позволяет объединить несмещенные, но чувствительные к обучающей выборке алгоритмы в несмещенную композицию с низкой дисперсией
- Хорошим семейством базовых алгоритмов здесь являются решающие деревья — они достаточно сложны и могут достигать нулевой ошибки на любой выборке (следовательно, имеют низкое смещение)
- **Метод случайных лесов** основан на бэггинге над решающими деревьями
- В случайных лесах корреляция между деревьями понижается путем рандомизации по двум направлениям:
  - по объектам (каждое дерево обучается по бутстрапированной подвыборке)
  - по признакам (в каждой вершине разбиение ищется по подмножеству признаков)

Метод случайных лесов

Алгоритм 3.1. Random Forest

1: **для** $n = 1, \dots, N$
2: Сгенерировать выборку $\tilde{X}_n$ с помощью бутстрэпа
3: Построить решающее дерево $b_n(x)$ по выборке $\tilde{X}_n$:

- дерево строится, пока в каждом листе не окажется не более $n_{\min}$ объектов
- при каждом разбиении сначала выбирается m случайных признаков из p, и оптимальное разделение ищется только среди них

4: Вернуть композицию $a_N(x) = \frac{1}{N} \sum_{n=1}^{N} b_n(x)$

Случайные леса

- В случайных лесах признак, по которому производится разбиение, выбирается не из всех возможных признаков, а лишь из их случайного подмножества размера m
- Рекомендуется в задачах классификации брать $m = [\sqrt{d}]$, а в задачах регрессии — $m = [d/3]$, где d — число признаков
- Также рекомендуется в задачах классификации строить каждое дерево до тех пор, пока в каждом листе не окажется по одному объекту
- А в задачах регрессии — пока в каждом листе не окажется по пять объектов

Случайные леса

- Случайные леса — один из самых сильных методов построения композиций
- На практике он может работать немного хуже градиентного бустинга, но при этом он гораздо более прост в реализации

Out-of-Bag

- Каждое дерево в случайном лесе обучается по подмножеству объектов
- Это значит, что те объекты, которые не вошли в бутстрапированную выборку X_n дерева b_n, по сути являются контрольными для данного дерева
- Значит, мы можем для каждого объекта x_i найти деревья, которые были обучены без него, и вычислить по их ответам **out-of-bag-ошибку** с функцией потерь L:

$$\text{OOB} = \sum_{i=1}^{\ell} L\left(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N}[x_i \notin X_n] b_n(x_i)\right)$$

Out-of-Bag

- Можно показать, что по мере увеличения числа деревьев N данная оценка стремится к **leave-one-out-оценке**
- Но при этом out-of-bag-ошибку существенно проще вычислять

Источники

- https://scikit-learn.org/stable/modules/ensemble.html
- http://www.machinelearning.ru/