Содержание

1 Решающие деревья

Пусть $\mathbf{X} \in \mathbb{R}^{n \times k}$ — матрица данных, n — число индивидов, k — число признаков, $Y \in \mathbb{R}^n$ — вектор отклика, X_i — вектор признака, $i \in 1:k$.

Определение 1. Дерево решений — бинарное дерево (V, E), в котором каждой $v \in V$, не являющейся листом соответствует функция, являющаяся предикатом $\beta_v : X \to \{0, 1\}$, а каждому листу $l \in V$ соответствует прогноз $f_l : X \to Y$.

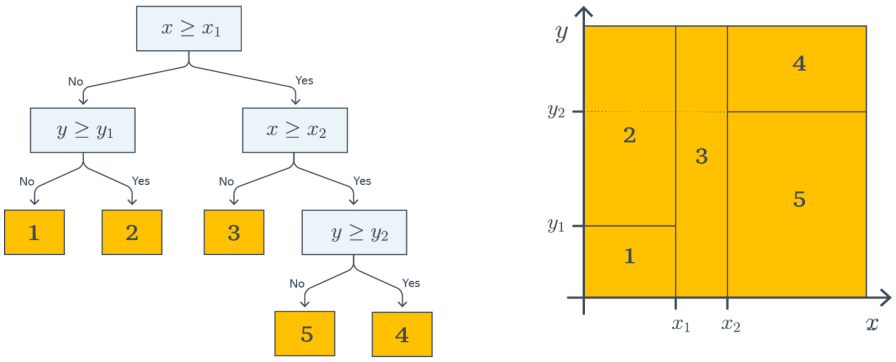

Рис. 1: Пример дерева решений

Деревья решений можно применять для классификации и регрессии.

1.1 Задача регрессии

Разобьём всевозможные значение X_i, $i \in 1:k$ на J непересекающихся множеств $\{R_j\}_{j=1}^J$, $R_j \in \mathbb{R}^k$. Предсказание для $x \in X$ равно

$$f(x) = \sum_{j=1}^{J} c_j \mathbb{1}(x \in R_j).$$

Выбирать набор R_j будем решая оптимизационную задачу минимизации суммы квадратов остатков

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - f(x_i))^2 \to \min_{R_1, \ldots, R_J}.$$

Тогда оценка c_j равна

$$\widehat{c}_j = \frac{1}{|R_j|} \sum_{x_i \in R_j} y_i.$$

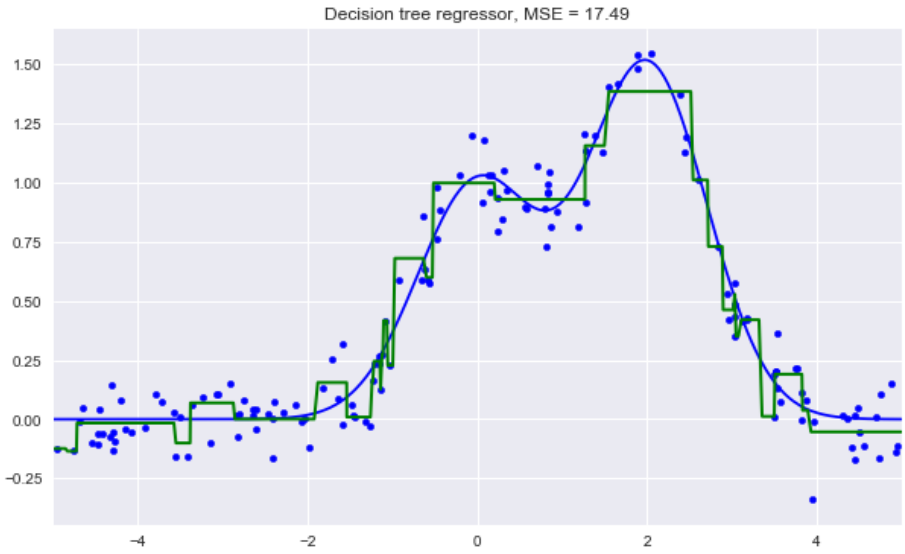

Рис. 2: Решающее дерево в задаче регрессии

1.2 Задача классификации

Обозначим за p_{jk} долю объектов класса $c \in Y$ в многомерном прямоугольнике $R_j \in \mathbb{R}^p$

$$\widehat{p}_{jc} = \frac{1}{N_j} \sum_{x_i \in R_j} \mathbb{1}(y_i = c)$$

.

$\{R_j\}_{j=1}^{J}$ — решение оптимизационной задачи

$$\text{ME} = 1 - \max_{j,c} \widehat{p}_{jc} \to \min_{R_1,\ldots,R_J}.$$

Однако ME не является дифференцируемой функцией(?вроде дифференцируемость и не нужна при построении), поэтому используются другие функции:

- Индекс Джини $G(j) = \sum\limits_{k=1}^{K} \widehat{p}_{jc}(1 - \widehat{p}_{jc})$
- Кросс-энтропия $CI(j) = -\sum\limits_{k=1}^{K} \widehat{p}_{jc} \log \widehat{p}_{jc}$

Предсказание для объекта $x \in X$ равно

$$f(x) = \operatorname*{argmax}_{c \in Y} \widehat{p}_{ck},$$

где j — индекс многомерного многоугольника, в который попадает x.

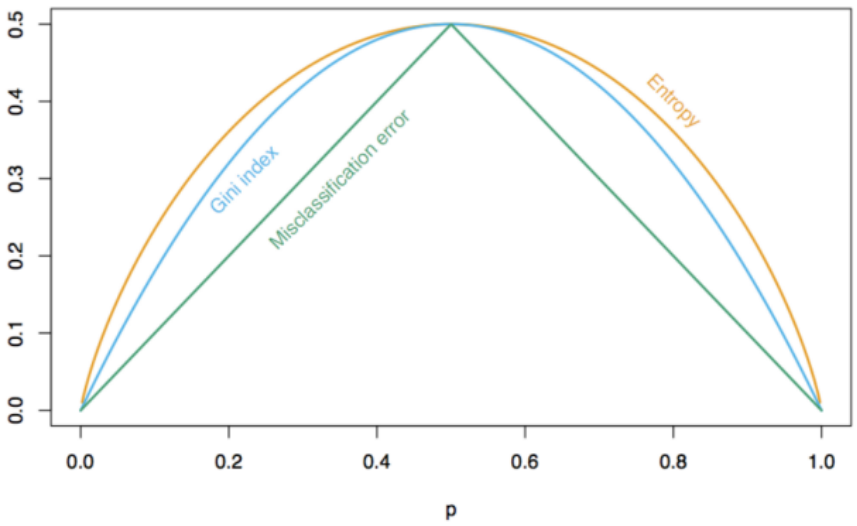

Рис. 3: Информационные индексы ME, G, CI в случае двух классов

1.3 Алгоритм построения дерева

1. Выбираем признак X_j и порог s так, чтобы разбиение X^n на $R_1(j,s) = \{x \in X^n | X_j < s\}$ и $R_2(j,s) = \{x \in X^n | X_j \geqslant s\}$ решало оптимизационную задачу, соответствующую задаче классификации или регрессии:

$$\sum_{i:x_i \in R_1(j,s)} \text{error}(y_i, \hat{y}_{R_1}) + \sum_{i:x_i \in R_2(j,s)} \text{error}(y_i, \hat{y}_{R_2}) \to \min_{j,s},$$

   где error — функция потерь, в случае регрессии сумма остатков, в случае классификации индекс Джини или кросс-энтропия.

2. Разбиваем выборку на области R_1 и R_2, образуя две дочерние вершины.

3. Повторяем процедуру в пределах каждой получаемой области, пока не выполнится критерий остановки.

Очевидный критерий остановки — остановка в случае если все индивиды в листе принадлежат одному классу. Дополнительно можно задать следующие критерии остановки:

- Ограничение максимальной глубины дерева
- Ограничение минимального числа объектов в поддереве
- Ограничение максимального числа индивидов в листе

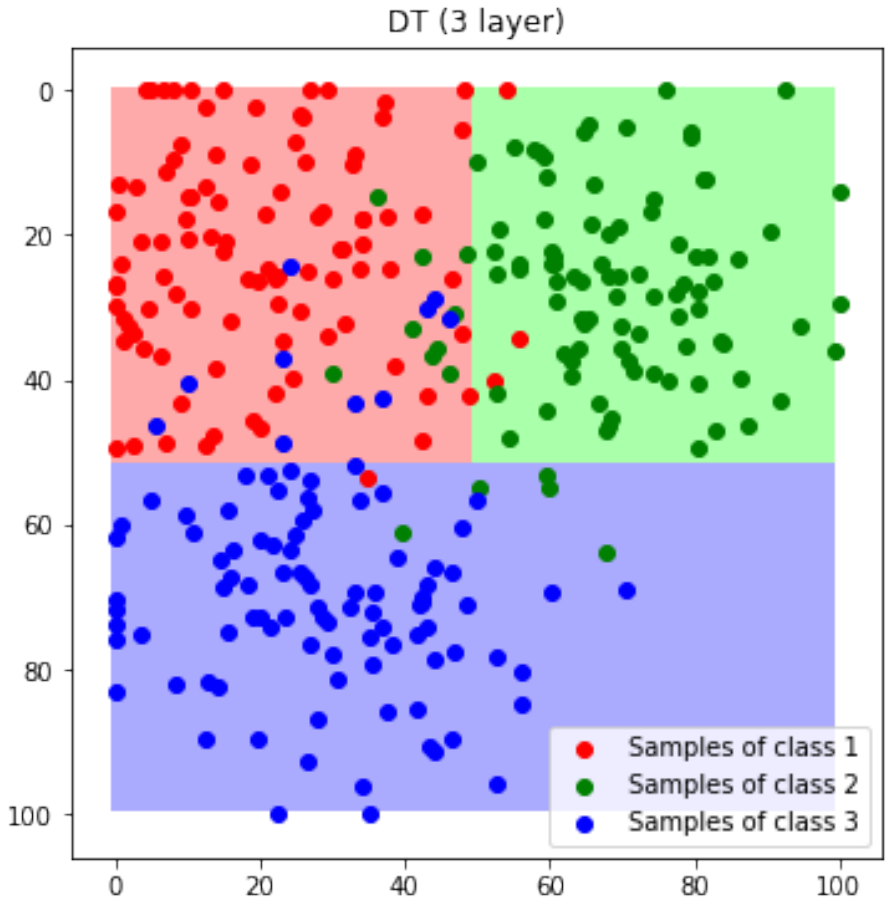

Рис. 4: Решающее дерево в задаче классификации

1.4 Стрижка деревьев

Очевидно, что можно построить дерево решений с нулевой ошибкой на тренировочной выборке. Достаточно разбить $\mathbb{R}^p$ на n областей R_j, $j \in 1:n$, получится глубокое дерево. Однако в таком случае ошибка на тестовой выборке может быть большой.

Для устранения такой проблемы деревья "стригут"(prune).

Пусть T_0 — большое построенное дерево. Можно в качестве T_0 взять дерево с нулевой ошибкой на тренировочной выборке. Рассмотрим поддерево $T \subset T_0$, в качестве оптимизируемой функции возьмём

$$Q_\alpha(T) = \text{error}(T) + \alpha|l(T)|,$$

где $\text{error}(T)$ — ошибка на тренировочной выборке, $|l(T)|$ — количество ли-

стьев в дереве, α — положительный гиперпараметр. α подбирается с помощью кросс-валидации. После подбора α строится поддерево T, на котором достигается минимум $Q_\alpha(T)$.

1.5 Обработка категориальных данных

Можно разбить вершину по категориям, однако такой подход может давать слишком глубокие деревья. Есть другой способ обработки категориальных значений, пусть категориальный признак имеет M значений $\mathbb{S} = \{s_i\}_{i=1}^{M}$, будем рассматривать всевозможные разбиения $\mathbb{S}$ на 2 дизъюнктных подмножества $\mathbb{S} = V \cup W$, $V \cap W = \emptyset$. В таком случае предикат в вершине есть $\mathbb{1}(x \in V)$. Проблема такого подхода заключается в том, что нужно перебрать $2^{M-1} - 1$ разбиений, но есть алгоритм, позволяющий избавиться от перебора всех множеств и дающий результат, совпадающий с результатом полного перебора.

1.6 Обработка пропусков

Есть несколько способов обработки пропусков:

- Убрать индивидов с пропусками (pairwise). Однако после выкидывания всех пропусков, размер выборки может сильно уменьшиться.
- На стадии выбора разбиения пропуски не учитываются и значения с пропусками спускаются в одно поддерево
- Не учитывать на шаге поиска разбиения пропуски, но спустить индивидов в оба поддерева с весами

1.7 Преимущества и недостатки

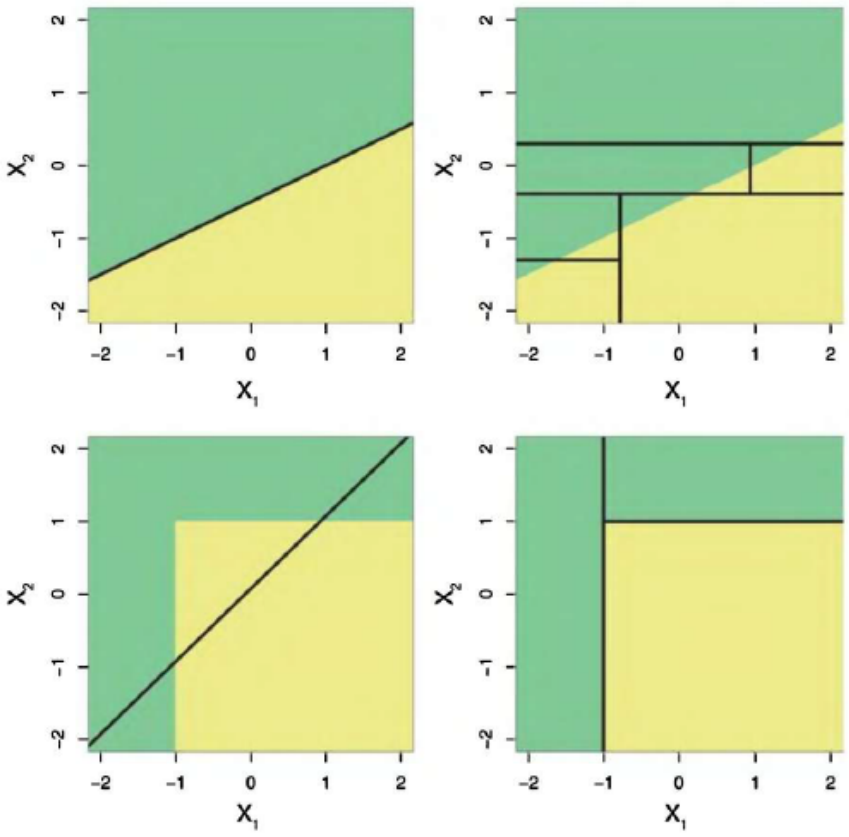

Рис. 5: Примеры решений задач классификации с линейной (верхний ряд) и нелинейной (нижний ряд) зависимостью. В левой части решение с помощью линейной модели, в правой — с помощью решающего дерева.

Преимущества:

- Простота интерпретации
- Пригодность и для задач регрессии, и для задач классификации
- Возможность работы с пропусками в данных

Недостатки:

- Основан на «жадном» алгоритме (решение является лишь локально оптимальным в каждом узле и может быть неоптимальным для всего дерева)
- Метод явялется неустойчивым и склонным к переобучению

2 Вероятностная постановка

2.1 Генеральная постановка

Предполагаем, что η и $\boldsymbol{\xi}$ функционально зависимы:

$$\eta = \varphi(\boldsymbol{\xi}) + \varepsilon, \tag{1}$$

φ — неизвестная функция.
$\eta \in \mathbb{R}$ — случайная величина, зависимая переменная.
$\boldsymbol{\xi} \in \mathbb{R}^p$ — случайный вектор, признаки.
$\varepsilon \in \mathbb{R}$ — случайная величина, ошибка.

2.2 Выборочная постановка

$$y_i = \varphi(\mathbf{x}_i) + \varepsilon_i, \tag{2}$$

φ — неизвестная функция.
y_i — реализация случайной величины η, зависимая переменная.
$\mathbf{x}_i$ — реализация случайного вектора $\boldsymbol{\xi}$, признаки.
$\varepsilon_i \in \mathbb{R}$ — реализация случайной величины ε, ошибка.

3 Алгоритмы

3.1 CART (использует Индекс Джини)

Выбираем признак X_j и порог s так, чтобы разбиение $\mathbf{X}$ на

$$R_1(j,s) = \{\mathbf{x}_i \in \mathbf{X} | X_j < s\}$$

и

$$R_2(j,s) = \{\mathbf{x}_i \in \mathbf{X} | X_j \geqslant s\}$$

решало задачу

$$\sum_{i:\mathbf{x}_i \in R_1(j,s)} (y_i - \hat{c}_1) + \sum_{i:\mathbf{x}_i \in R_2(j,s)} (y_i - \hat{c}_2) \to \min_{j,s} \tag{3}$$

где оценка коэффициента $\hat{c}_j = \dfrac{1}{|R_j|} \sum\limits_{\mathbf{x}_i \in R_j(j,s)} y_i, \quad j = 1, 2.$

1. Перебираем все возможные s_j и выбираем то значение, при котором Индекс Джини минимален.
2. Разбиваем выборку на области R_1 и R_2, образуя две дочерние вершины L_v и R_v.
3. Повторяем процедуру, разбивая каждый из получившихся регионов, пока не будет достигнута максимальная глубина.
4. Алгоритм CART — жадный, он выбирает наилучшее расщепление на текущем уровне, что не обязательно приводит к наименьшей загрязненности на уровнях ниже. Алгоритм хорош, но не всегда оптимален.

3.2 ID3 (использует Кросс-энтропию)

Идея алгоритма заключается в последовательном дроблении выборки на две части до тех пор, пока в каждой части не окажутся объекты только одного класса. Нам необходимо выбирать такой предикат, чтобы ветвление дерева было максимально информативно

1. $\mathbf{X}$ — обучающая выборка, $\mathbf{y} \in \{1, \cdots, k\}$.
2. Если все $\mathbf{x}_i$ имеют класс k, ставим метку 1 в корень и выходим из цикла.
3. Если ни один $\mathbf{x}_i$ не имеет класс k, ставим метку 0 в корень и выходим из цикла.
4. Предикат $R(\mathbf{x}_i) := \{\mathbf{x}_i | X_j \leqslant s_j\}$ для которого информационная выгода наибольшая.

5. Разбиваем $\mathbf{X}$ на $\mathbf{X}_0$ и $\mathbf{X}_1$ по предикату R

$$\mathbf{X}_0 := \{\mathbf{x}_i \in \mathbf{X} : R(\mathbf{x}_i) = 0\},$$

$$\mathbf{X}_1 := \{\mathbf{x}_i \in \mathbf{X} : R(\mathbf{x}_i) = 1\}.$$

6. Если $\mathbf{X}_0 = \varnothing$ или $\mathbf{X}_1 = \varnothing$, создаем новый лист v, k_v — класс, в котором находится большинство элементов $\mathbf{x}_i$.

7. Иначе создаем внутреннюю вершину v:

   (a) $R_v = R$;

   (b) L_v;

   (c) R_v.

3.3 Стрижка деревьев

Описанный выше процесс может дать хорошие прогнозы на обучающем наборе, но, вероятно, *переобучится*, что приведет к плохим результатам на тестовых наборах. Почему?

Меньшее дерево с меньшим количеством разбиений (то есть с меньшим количеством областей $R_1, \cdots, R_J$) может привести к меньшей дисперсии и лучшей интерпретации за счет небольшого смещения. Мы можем пожертвовать смещением, но получить меньшую дисперсию.

Одна из возможных альтернатив описанному выше процессу — выращивать дерево только до тех пор, пока уменьшение RSS из-за каждого разбиения превышает некоторый (высокий) порог.

Эта стратегия приведет к уменьшению размеров деревьев, но она слишком недальновидна: за кажущимся бесполезным разбиением в начале дерева может последовать очень хорошее разбиение — то есть разделение, которое в дальнейшем приводит к значительному сокращению RSS.

Лучшая стратегия — вырастить очень большое дерево T_0, а затем обрезать его, чтобы получить поддерево.

Cost complexity prunning — также называется сокращением наиболее слабых звеньев — используется для этого.

Мы рассматриваем последовательность деревьев, с настраиваемыми параметрами α. Каждому значению α соответствует поддерево $T \subset T_0$ (является подмножеством) такое, что

$$\sum_{j=1}^{|T|} \sum_{\mathbf{x}_i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|. \tag{4}$$

настолько мало насколько это возможно. Здесь $|T|$ указывает количество конечных узлов дерева T, R_j — это прямоугольник (**то есть подмножество пространства предикторов**), соответствующий j-му конечному узлу, а $\hat{y}_{R_j}$ — это среднее значение обучающих наблюдений в R_j.

Критерий остановки:

1. Ограничение макс. глубины дерева;
2. Ограничение мин. числа объектов в листе n_{min};
3. Ограничение макс. количества листьев в дереве;
4. Остановка в случае, если все объекты в листе относятся к одному классу.

4 Бутстрап

Пусть дана конечная выборка $X = (x_i, y_i)$ с вещевственными ответами. Сгенерируем подвыборку с помощью бутстрапа. Равномерно возьмем из выборки l объектов с возвращением. Отметим, что из-за возвращения среди них окажутся повторы. Обозначим новую выборку через X_1. Повторим процедуру N раз и получим подвыборки $X_1, \ldots, X_N$. Обучим по каждой выборке модель линейной регрессии и получим базовые алгоритмы $b_1(x), \ldots, b_N(x)$.

Предположим, что существует истинная функция ответов для всех объектов $y(x)$ и задано распределение на объектах $p(x)$. Тогда мы можем записать ошибку каждой функции регрессии, как

$$\epsilon_j(x) = b_j(x) - y(x), j = 1, ..., N, \tag{5}$$

и записать матожидание среднеквадратичной ошибки:

$$\mathbb{E}_x \left(b_j(x) - y(x)\right)^2 = \mathbb{E}_x \epsilon_j^2(x) \tag{6}$$

Средняя ошибка построенных функций регрессии имеет вид
$E_1 = \frac{1}{N} \sum_{j=1}^{N} \mathbb{E}_x \epsilon_j^2(x)$
Предположим, что ошибки несмещены и некоррелированы:
$\mathbb{E}_x \epsilon_j(x) = 0$
$\mathbb{E}_x \epsilon_i(x)\epsilon_j(x) = 0,\ i \neq j$
Построим новую функцию регрессии, которая будет усреднять ответы построенных нами функций:
$a(x) = \frac{1}{N} \sum_{j=1}^{N} b_j(x)$
Ее среднеквадратичная ошибка:
$E_N = \mathbb{E}_x \left(\frac{1}{N} \sum_{j=1}^{N} b_j(x) - y(x)\right)^2 = \mathbb{E}_x \left(\frac{1}{N} \sum_{j=1}^{N} \epsilon_j(x)\right)^2 =$
$= \frac{1}{N^2} \mathbb{E}_x \left(\sum_{j=1}^{N} \epsilon_j^2(x) + \sum_{i \neq j} \epsilon_i(x)\epsilon_j(x)\right) = \frac{1}{N} E_1$
Усреднение ответов позволило уменьшит средний квадрат ошибки в N раз.

Это является идеальным случаем, так как на практике некоррелируемость ошибок редко случается.

5 Bias-Variance decomposition

Ошибка любой модели складывается из трех факторов: сложности самой выборки, сходства модели с истинной зависимостью ответов от объектов в выборке, и 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.

Пусть задана выборка $X = (x_i, y_i)_{i=1}^{l}$ с вещественными ответами $y_i \in \mathbb{R}$. Будем считать что на пространстве $\mathbb{X} \times \mathbb{Y}$ существует распределение $p(x, y)$, из которого сгенерирована выборка X и все ответы на ней.

Рассмотрим квадратичную функцию потерь $L(y,a) = (y - a(x))^2$ и ее среднеквадратичный риск

$R(a) = \mathbb{E}_{x,y}\left[(y - a(x))^2\right] \int_{\mathbb{X}} \int_{\mathbb{Y}} p(x,y)(y - a(x))^2 dxdy$

Такой функционал усредняет ошибку модели в каждой точке пространства x и для каждого возможного ответа y, причём вклад пары (x,y), по сути, пропорционален вероятности получить её в выборке $p(x,y)$. Разумеется, на практике мы не можем вычислить данный функционал, поскольку распределение $p(x,y)$ неизвестно. Тем не менее, в теории он позволяет измерить качество модели на всех возможных объектах, а не только на обучающей выборке.

5.1 Минимум среднеквадратичного риска

???

5.2 Ошибка метода обучения

Для того, чтобы построить идеальную функцию регрессии, необходимо знать распределение на объектах и ответах $p(x,y)$, что, как правило, невозможно. На практике вместо этого выбирается некоторый метод обучения $\mu : (\mathbb{X} \times \mathbb{Y})^l \to \mathbf{A}$, который произвольной обучающей выборке ставит в соответствие некоторый алгоритм из семейства $\mathbf{A}$. В качестве меры качества метода обучения можно взять усредненный по всем выборкам среднеквадратичный риск алгоритма, выбранного методом μ по ввыборке:

$$L(\mu) = \mathbb{E}_X\left[\mathbb{E}_{x,y}\left[(y - \mu(X)(x)))^2\right]\right] =$$
$$= \int_{(\mathbb{X}\times\mathbb{Y})^l} \int_{\mathbb{X}\times\mathbb{Y}} (y - \mu(X)(x))^2 p(x,y) \prod_{i=1}^{l} p(x_i,y_i) dxdydx_1dy_1,\dots dx_ldy_l \quad (7)$$

Здесь матожидание $\mathbb{E}_X[\cdot]$ берется по всем возможным выборкам $(x_1,y_1),...,(x_l,y_l)$ из распределения $\prod_{i=1}^{l} p(x_i,y_i)$.

Среднеквадратичный риск на фиксированной выборке X можно расисать как

$\mathbb{E}_{x,y} = \left[(y - \mu(X))^2\right] = \mathbb{E}_{x,y}\left[(y - \mathbb{E}[y|x])^2\right] + \mathbb{E}_{x,y}\left[(\mathbb{E}[y|x] - \mu(X))^2\right]$

Подставим эту формулу в (7).

$$L(\mu) = \mathbb{E}_X\left[\underbrace{\mathbb{E}_{x,y}\left[(y - \mathbb{E}[y|x])^2\right]}_{\text{не завсисит от X}} + \mathbb{E}_{x,y}\left[(\mathbb{E}[y|x] - \mu(X))^2\right]\right] = \quad (8)$$
$$= \mathbb{E}_{x,y}\left[(y - \mathbb{E}[y|x])^2\right] + \mathbb{E}_{x,y}\left[\mathbb{E}_X\left[(\mathbb{E}[y|x] - \mu(X))^2\right]\right]$$

Преобразовываем второе слагаемое:

$$\mathbb{E}_{x,y}\left[\mathbb{E}_X\left[(\mathbb{E}[y|x]-\mu(X))^2\right]\right] =$$
$$= \mathbb{E}_{x,y}\left[\mathbb{E}_X\left[(\mathbb{E}[y|x]-\mathbb{E}_X[\mu(X)]+\mathbb{E}_X[\mu(X)]-\mu(X))^2\right]\right] =$$
$$= \mathbb{E}_{x,y}\left[\mathbb{E}_X\left[\underbrace{(\mathbb{E}[y|x]-\mathbb{E}_X\mu(X))^2}_{\text{не завсисит от X}}\right]\right] + \mathbb{E}_{x,y}\left[\mathbb{E}_X\left[(\mathbb{E}_X\mu(X)-\mu(X))^2\right]\right] +$$
$$+ 2\mathbb{E}_{x,y}\left[\mathbb{E}_X\left[(\mathbb{E}[y|x]-\mathbb{E}[\mu(X)])(\mathbb{E}_X[\mu(X)]-\mu(X))\right]\right] \quad (9)$$

Последнее слагаемое обращается в ноль.
Подставим (9) в 8.

$$L(\mu) = \underbrace{\mathbb{E}_{x,y}\left[(y-\mathbb{E}[y|x])^2\right]}_{\text{шум}} +$$
$$+ \underbrace{\mathbb{E}_x\left[(\mathbb{E}_X[\mu(X)]-\mathbb{E}[y|x])^2\right]}_{\text{смещение}} + \underbrace{\mathbb{E}_x\left[\mathbb{E}_X\left[(\mu(X)-\mathbb{E}_X[\mu(X)])^2\right]\right]}_{\text{разброс}} \quad (10)$$

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

На рис. 6 изображены модели с различными сдвигом и разбросом. Модели изображены синими точками, одна точка соответствует модели, обученной по одной из возможных обучающих выборок. Каждый круг характеризует качество модели — чем ближе точка к центру, тем меньше ошибок на контрольной выборке достигает данный алгоритм. Видно, что 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.

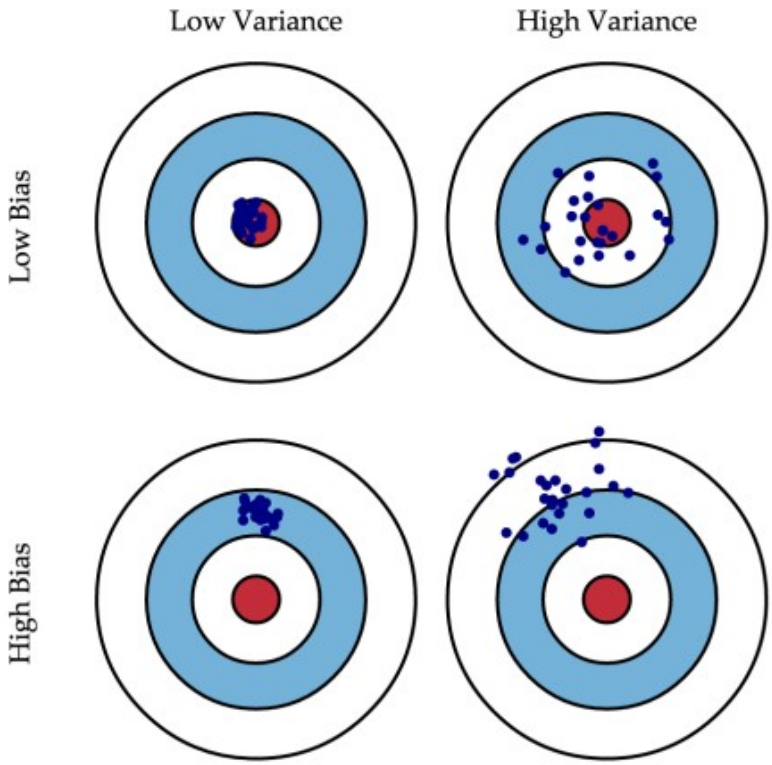

Рис. 6: Сдвиг и разброс разных моделей

6 Bagging

Деревья решений, страдают от высокой дисперсии. Это означает, что если разделить обучающие данные на две части случайным образом и применить дерево решений к обеим половинам, результаты, которые мы получим, могут быть совершенно разными. Напротив, процедура с низкой дисперсией даст схожие результаты при многократном применении к разным наборам данных; линейная регрессия имеет тенденцию к низкой дисперсии, если отношение n к p умеренно велико. Бэггинг, — это процедура для уменьшения дисперсии статистического метода обучения; она особенно полезна и часто используется в контексте деревьев решений.

Пусть у нас имеется набор из n независивых наблюдений $X_1, \dots X_n$, каждый из которых имеет дисперсию σ^2, дисперсия $\bar{X}$ будет равна σ^2/n. Другими словами, усреднение набора наблюдений снижает дисперсию.

Следовательно, естественный способ уменьшить дисперсию и, следовательно, увеличить точность предсказания статистического метода обучения

заключается в том, чтобы взять множество обучающих наборов из выборки, построить отдельную модель предсказания, используя каждый набор обучения и усреднить полученные прогнозы.

$$a_N(x) = \frac{1}{N}\sum_{n=1}^{N}\hat{\mu}(x) \tag{11}$$

Однако на практике это не реализовать, потому что у нас нет нескольких сетов обучающих данных. Вместо этого, мы можем примеить бутстрап беря несколько сэмплов из обучающего набора данных. При таком подходе, мы генерируем B разных подвыборок с помощью бутсрапа. Затем мы обучаем модель на b-ом сете обучающих данных и усредняем прогнозы.

Это называется бэггингом.

Заметим, что в методе обучения для бэггинга появляется ещё один источник случайности — взятие подвыборки. Чтобы функционал качества $L(\mu)$ был детерминированным, мы будем далее считать, что матожиданием$\mathbb{E}_X[\cdot]$ берётся не только по всем обучающим выборкам X, но ещё и по всем возможным подвыборкам $\hat{X}$, получаемым с помощью бутстрапа. Это вполне логичное обобщение, поскольку данное матожидание вводится в функционал именно для учёта случайностей, связанных с процедурой обучения модели.

Найдём смещение из разложения (10) для бэггинга:

$$\mathbb{E}_{x,y}\left[\left(\mathbb{E}_X\left[\frac{1}{N}\sum_{b=1}^{N}\hat{\mu}(X)(x)\right]-\mathbb{E}[y|x]\right)^2\right]=\mathbb{E}_{x,y}\left[\left(\frac{1}{N}\sum_{b=1}^{N}\mathbb{E}_X[\hat{\mu}(X)(x)]-\mathbb{E}[y|x]\right)^2\right]=$$
$$=\mathbb{E}_{x,y}\left[\left(\mathbb{E}_X\left[\hat{\mu}(X)(x)\right]-\mathbb{E}[y|x]\right)^2\right] \tag{12}$$

Мы получили, что смещение композиции, полученной с помощью бэггинга, совпадает со смещением одного базового алгоритма. Таким образом, бэггинг не ухудшает смещенность модели.

Теперь рассмотрим разброс. Из (10) он равен:

$$\frac{1}{N}\mathbb{E}_{x,y}\left[\mathbb{E}_X\left[\left(\hat{\mu}(X)(x)-\mathbb{E}_X[\hat{\mu}(X)(x)]\right)^2\right]\right]+$$
$$+\frac{N(N-1)}{N^2}\mathbb{E}_{x,y}\left[\mathbb{E}_X\left[\left(\hat{\mu}(X)(x)-\mathbb{E}_X[\hat{\mu}(X)(x)]\right)\times\left(\hat{\mu}(X)(x)-\mathbb{E}_X[\hat{\mu}(X)(x)]\right)\right]\right] \tag{13}$$

Первое слагаемое — это дисперсия одного базового алгоритма, деленная на длину композиции N. Второе — ковариация между двумя базовыми алгоритмами. Мы видим, что 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.

До сих пор мы описывали бэггинг в контексте регрессии, для предсказания количественного результата y. Как можно распространить бэггинг на задачу классификации, где Y является качественным? В этой ситуации существует существует несколько возможных подходов, но самый простой заключается в следующем. Для тестового сета, мы можем записать класс, предсказанный каждым из N деревьев, и сделать "*majority vote*— это наиболее часто встречающийся класс среди N предсказаний.

6.1 Out-of-Bag Error Estimation

Ключевой момент bagging модели заключается в том, что деревья многократно подгоняются к бутстреп-подмножествам наблюдений. В среднем каждое дерево сетки использует около двух третей наблюдений. Оставшаяся треть наблюдений, не использованная для подгонки данного дерева сетки, называется *out-of-bag* наблюдениями (OOB).

Мы можем предсказать ответ для i-го наблюдения, используя каждое из деревьев, в которых это наблюдение было OOB. Это даст около N/3 предсказаний для i-го наблюдения. Для того чтобы получить единое предсказание для i-го наблюдения, мы можем усреднить эти предсказанные ответы (при регрессионной модели) или взять majority vote (при классификации). Это приводит к единственному OOB-предсказанию для каждого наблюдения. Предсказание OOB может быть получено таким образом для каждого из n наблюдений, из чего следует, что общий MSE OOB (для задачи регрессии) или ошибка классификации (для задачи классификации) может быть вычислена. Полученная ошибка OOB является достоверной оценкой тестовой ошибки для bagging модели, поскольку прогноз для каждого наблюдения предсказывается с использованием только тех деревьев, которые не подошли для данного наблюдения.

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