Решающие деревья. Random Forest

Рукавишникова Анна
Страшко Владислав
Сандул Михаил

Санкт-Петербург
2019 г.

Кусочно-постоянные функции

Кусочно-постоянной функцией $f : \mathbb{R}^l \to \mathbb{R}$, заданной на конечном разбиении $\mathbb{R}^l = A_1 \vee \ldots \vee A_m$ назовем

$$f(x) = \sum_{i=1}^{m} c_i \mathbb{1}_{A_i}(x),$$

где c_i — различные вещественные числа, $\mathbb{1}_{A_i}(x)$ — индикаторная функция множества A_i.

Мы будем рассматривать только многомерные прямоугольники в качестве элементов разбиения: такие множества легко задать с помощью системы неравенств.

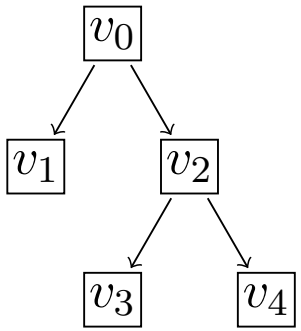
Кусочно-постоянные функции. Мотивация и свойства

- Относительно простой математический объект.
- Удобный инструмент для аппроксимации гладких функций.
- Пространство конечно-постоянных функций линейно.

Представление кусочно-постоянных функций. Определения

Дерево — конечный связный ациклический граф с множеством вершин V и выделенной вершиной $v_0 \in V$, в которую не входит ни одно ребро. Вершина v_0 называется **корнем дерева**.

Вершины, из которых не выходит ни одного ребра, называются **терминальными** (или **листами**), остальные вершины называются **внутренними**.



Бинарное дерево — дерево, из любой внутренней вершины которого выходит ровно два ребра. Они связывают внутреннюю вершину с левой дочерней вершиной L_v и с правой дочерней вершиной R_v.

Представление кусочно-постоянных функций

Лемма

Любую кусочно-постоянную функцию $f(x) = \sum_{i=1}^{m} c_i \mathbb{1}_{A_i}(x)$, *заданную на разбиении* $\mathbb{R}^l = A_1 \vee \ldots \vee A_m$, *состоящем из многомерных прямоугольников, можно представить в виде дерева с конечным числом вершин, во внутренних вершинах которого находятся условия на значения переменных, а в листах — значения функции* c_i.

Определение

Дерево, которое является представлением кусочно-постоянной функции будем называть **решающим**.

Лемма

Любое решающее дерево можно представить в виде бинарного решающего дерева.

Бинарное решающее дерево

1: $v := v_0$;
2: **пока** вершина v внутренняя
3: **если** $\beta_v(x) = 1$ **то**
4: $v := R_v$; (переход вправо)
5: **иначе**
6: $v := L_v$; (переход влево)
7: **вернуть** c_v.

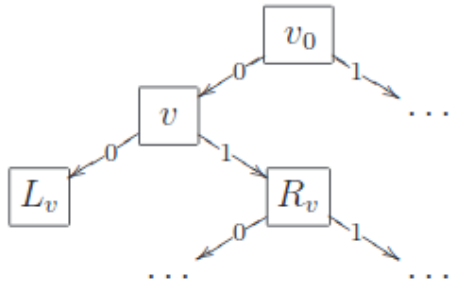


Рис.: Бинарное решающее дерево

- Каждой внутренней вершине v приписана функция (или предикат) $\beta_v : X \to \{0, 1\}$.
- Каждой листовой вершине v приписан прогноз $c_v \in Y$ (в случае с классификацией листу также может быть приписан вектор вероятностей).

Постановка задачи

Решающие деревья можно применять как для задач регрессии, так и для задач классификации.

Пусть X — множество объектов, Y — множество ответов
$y : X \to Y$ — неизвестная зависимость.

Дано: обучающая выборка — $X^n = (x_i, y_i)_{i=1}^n$,
$y_i = y(x_i), i = 1, \ldots, n$ — известные ответы.

- $y_i \in \{1, \ldots, K\} \Rightarrow$ задача классификации.
- $y_i \in \mathbb{R} \Rightarrow$ задача регрессии.

Решающие деревья в задаче регрессии

Пусть $X \in \mathbb{R}^{n \times p}$ — матрица данных с p признаками для n наблюдений, $Y \in \mathbb{R}^n$ — вектор ответов.

Идея: разбить пространство признаков, т.е. совокупность всех возможных значений $X_1, \ldots, X_p$ на J непересекающихся областей $R_1, \ldots, R_J$ (многомерных прямоугольников).

Предсказание для объекта x:

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

Области $R_1, \ldots, R_J$ выбираются, исходя из условия:

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

Тогда

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

Решающие деревья в задаче регрессии

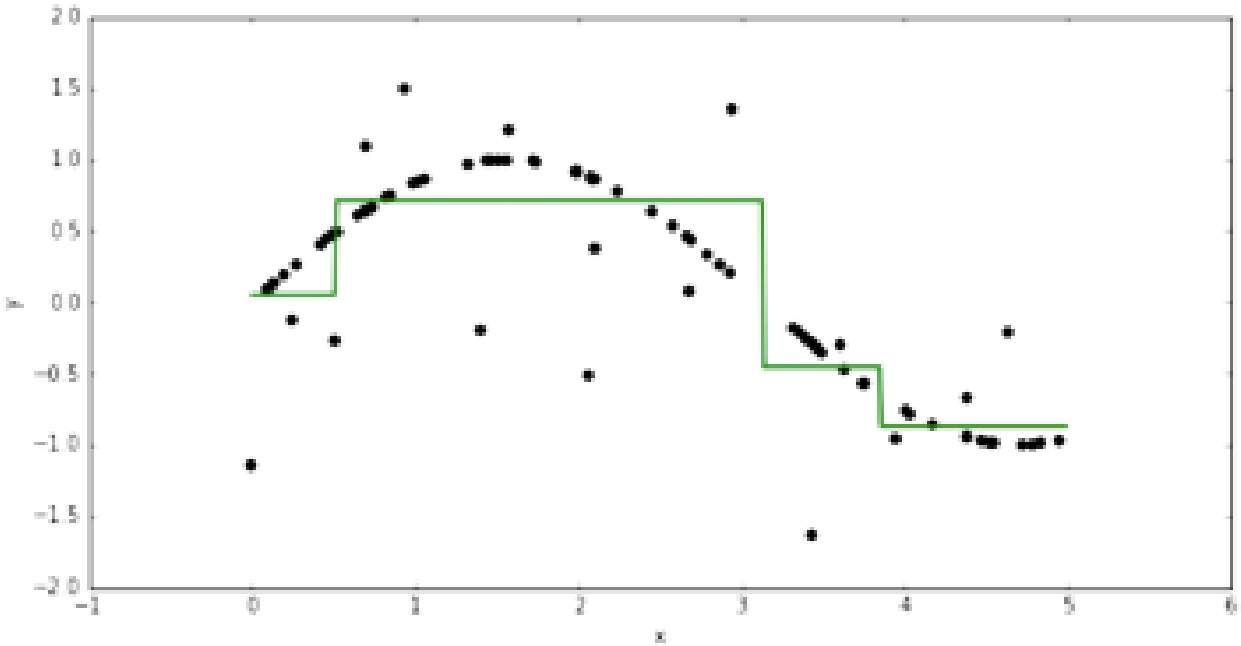


Рис.: Использование решающих деревьев в задачах регрессии

Решающие деревья в задаче классификации

Делаем аналогичное разбиение на области $R_1, \ldots, R_J$, которые выбираются из условия:

$$E = 1 - \max_k(\hat{p}_{jk}) \to \min_{R_1,\ldots,R_j},$$

где $\hat{p}_{jk} = \frac{1}{|R_j|} \sum\limits_{x_i \in R_j} \mathbb{1}_{(y_i=k)}$, $j = 1, \ldots, J$.

На практике чаще всего используют два других (информационных) критерия для фиксированного j

- $G = \sum\limits_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk})$ — индекс Джини,
- $D = -\sum\limits_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk}$ — коэффициент перекрёстной энтропии.

Предсказание для объекта x:

$$f(x) = \operatorname*{argmax}_{k \in Y} \hat{p}_{jk}.$$

Решающие деревья в задаче классификации

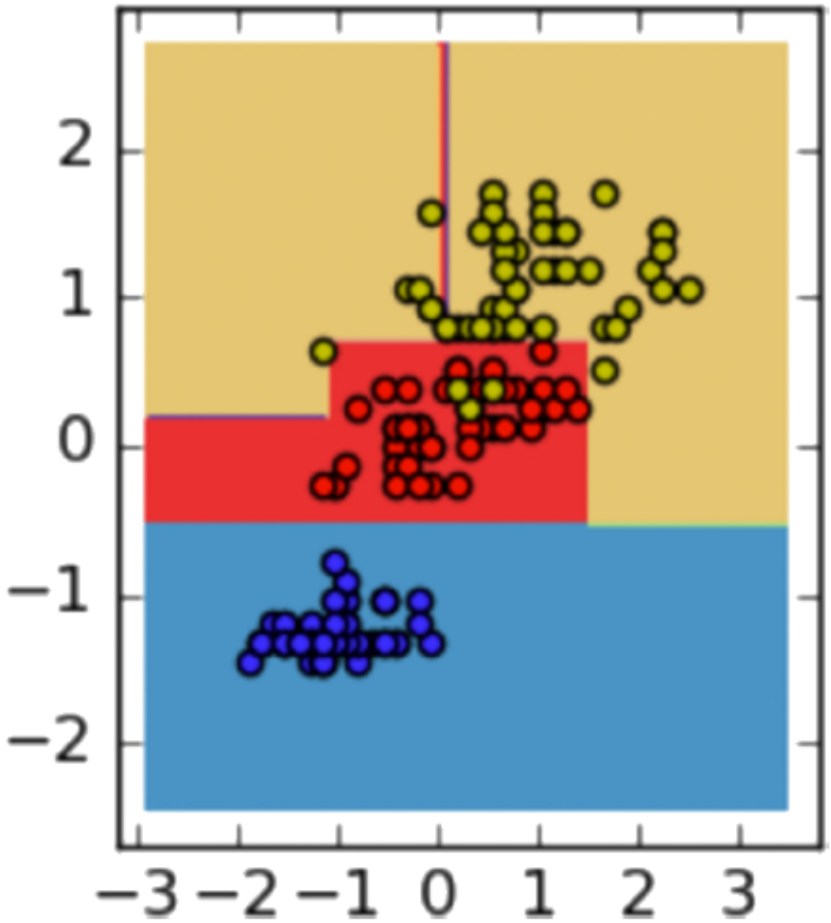


Рис.: Использование решающих деревьев в задачах классификации

Жадный алгоритм построения решающего дерева

Жадный нисходящий алгоритм построения дерева (для задачи регрессии):

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 $R_2(j,s) = \{x \in X^n | X_j \geq s\}$ решало задачу:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \to \min_{j,s},$$

где $\hat{y}_{R_l} = \frac{1}{|R_l|} \sum_{i:x_i \in R_l(j,s)} y_i, \quad l = 1, 2.$

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.

На выходе получаем дерево, в каждом из листов которого содержится по крайней мере 1 объект исходной выборки X^n.

Критерии остановки

- Ограничение максимальной глубины дерева.
- Ограничение минимального числа объектов в листе n_{min}.
- Ограничение максимального количества листьев в дереве.
- Остановка в случае, если все объекты в листе относятся к одному классу.

Проблема: для очень глубоких деревьев имеем переобучение.

Стрижка деревьев (pruning tree)

Цель: борьба с переобучением за счёт уменьшения дисперсии и увеличения смещения.

Cost-complexity pruning

Пусть T_0 — дерево, полученное в результате работы жадного алгоритма, $T^t \subset T_0$ — обрезанное в узле t поддерево.
Функция cost-complexity:

$$Q_\alpha(T) = Q(T) + \alpha|l(T)|,$$

где $Q(T)$ — training error, $\alpha \geq 0$ — параметр (компромисс между размером дерева и его соответствию данным), $|l(T)|$ — число листьев в поддереве T.

Идея: для каждого α найти такое поддерево $T^t \subset T_0$, чтобы минимизировать $Q_\alpha(T)$.

Стрижка деревьев (продолжение)

Можно показать, что существует последовательность вложенных деревьев с одинаковыми корнями:

$$T_K \subset T_{K-1} \subset \ldots \subset T_0,$$

где каждое дерево T_i минимизирует функцию $Q_\alpha(T)$ для $\alpha \in [\alpha_i, \alpha_{i+1})$, причём $0 = \alpha_0 < \alpha_1 < \ldots < \alpha_K < \infty$.

Выберем α с помощью кросс-валидации и поддерево, соответствующее этому α.

Сравнение деревьев с линейными моделями

Модель линейной регрессии:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j. \tag{1}$$

Модель регрессионного дерева:

$$f(X) = \sum_{j=1}^{J} c_j \mathbb{1}_{(X \in R_j)}. \tag{2}$$

Если зависимость между $X_1, \ldots, X_p$ и Y приближённо можно считать линейной, то лучше использовать (1), а в случае сложной нелинейной зависимости используем (2).

Сравнение деревьев с линейными моделями

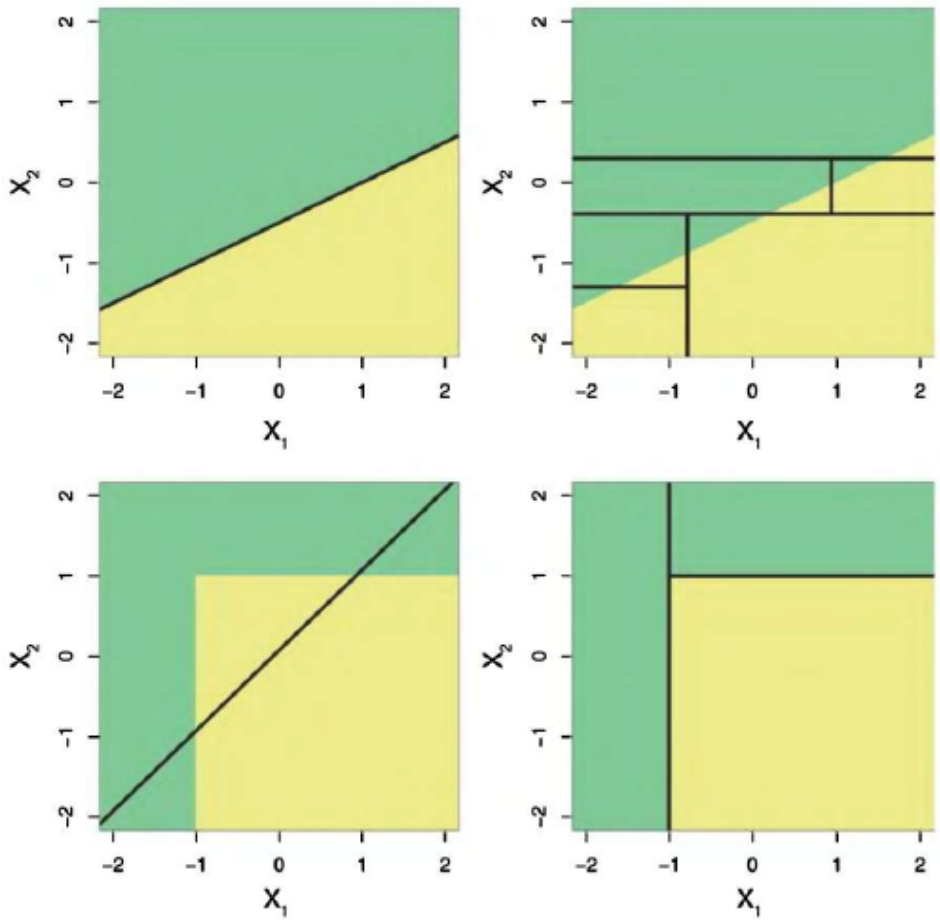


Рис.: Примеры решений задач классификации с линейной (верхний ряд) и нелинейной (нижний ряд) зависимостью. В левой части решение с помощью линейной модели, в правой — с помощью решающего дерева.

Преимущества и недостатки решающих деревьев

Преимущества:

- простота интерпретации результатов,
- пригодность и для задач регрессии, и для задач классификации,
- возможность работать с пропущенными данными.

Недостатки:

- склонность к переобучению,
- низкая точность прогнозов по сравнению с другими методами машинного обучения.

Bagging

Цель: уменьшение дисперсии модели с сохранением низкого смещения.

Идея: пусть $\xi_1, \ldots, \xi_n$ — н. о. р. с. в., $\mathrm{D}\,\xi_i = \sigma^2$, тогда $\mathrm{D}\,\bar{\xi} = \frac{\sigma^2}{n}$.

Реализация:
$X^n = (x_i, y_i)_{i=1}^n$ — обучающая выборка.

- используем bootstrap для генерации B обучающих выборок (отбор с возвращением) $X_b^{*n}, \quad b = 1, \ldots, B$,
- на основе полученных выборок строим решающие деревья $\{T_b\}_{b=1}^B$,
- находим оценку:
  - в задаче регрессии $\hat{f}_{bag}(x) = \frac{1}{B} \sum\limits_{b=1}^{B} T_b(x)$,
  - в задаче классификации с K классами: $\hat{f}_{bag}(x) = [p_1(x), \ldots, p_K(x)]$ — K-мерный вектор, где $p_k(x)$ — доля деревьев, предсказавших класс k для x (в качестве предсказания берём $majority\ vote\{\hat{f}_b(x)\}_{b=1}^B$, где $\hat{f}_b(x)$ — предсказание класса b-м решающим деревом.)

Random forest

Идея: уменьшение разброса композиции за счёт уменьшения корреляции базовых алгоритмов.

Алгоритм построения случайного леса

1. строим B bootstrap-выборок $X_b^{*n}, \quad b = 1, \ldots, B$,
2. на основе X_b^{*n} рекурсивно строим решающее дерево T_b, пока не достигнем критерия остановки $(n_{min} = c)$ по следующим правилам для каждого листа:
   - среди p признаков случайным образом выбираются m,
   - повторяем 1 и 2 шаги жадного алгоритма, выбирая признак X_j из имеющихся m и порог s.
3. построенные деревья $\{T_b\}_{b=1}^B$ объединяются в композицию:
   - в задаче регрессии: $\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$,
   - в задаче классификации: $\hat{f}_{rf}^B(x) = majority\ vote\{\hat{f}_b(x)\}_{b=1}^B$, где $\hat{f}_b(x)$ — предсказание класса b-м решающим деревом.

Оценка ошибки out-of-bag

Дерево, обученное по bootstrap-выборке, использует приблизительно $2/3$ полной выборки $\Rightarrow$ оставшуюся $1/3$ выборки используем для оценки обобщающей способности.

Оценка качества случайного леса из B деревьев в рамках подхода out-of-bag:

$$OOB = \sum_{i=1}^{N} L\left(y_i, \frac{1}{\sum\limits_{b=1}^{B}[x_i \notin X_b^{*n}]} \sum_{b=1}^{B}[x_i \notin X_b^{*n}] T_b(x_i)\right).$$

Почему работают bagging и random forest?

Пусть $L(y) = (f(x) - y)^2$ — квадратичная функция потерь, $X^n = (x_i, y_i)_{i=1}^n \sim p(x, y)$, μ — метод обучения.
Среднеквадратический риск:

$$\mathrm{E}_{x,y}(f(x) - y)^2 = \int_X \int_Y L(y)p(x, y)dxdy.$$

Минимум среднеквадратического риска:

$$f^* = \mathrm{E}(y|x) = \int_Y yp(y|x)dx.$$

Мера качества обучения μ:

$$Q(\mu) = \mathrm{E}_{X^n}\,\mathrm{E}_{x,y}(\mu(X^n)(x) - y)^2,$$

где $\mu(X^n)(x)$ — результат применения алгоритма, построенного по выборке X^n, к объекту x.

Bias–variance decomposition

Теорема

В случае квадратичной функции потерь для любого μ

$$Q(\mu) = \underbrace{\mathrm{E}_{x,y}(f^*(x) - y))^2}_{\text{шум (noise)}} + \underbrace{\mathrm{E}_{x,y}(\bar{f}(x) - f^*(x))^2}_{\text{смещение (bias)}} +$$

$$+ \underbrace{\mathrm{E}_{x,y}\,\mathrm{E}_{X^n}(\mu(X^n)(x) - \bar{f}(x))^2}_{\text{разброс (variance)}},$$

где $\bar{f}(x) = \mathrm{E}_{X^n}(\mu(X^n)(x))$.

Смещение и разброс композиции алгоритмов

Пусть b_t, $t = 1, \ldots, T$ — базовые алгоритмы, обучающиеся по случайным подвыборкам, $f_T(x) = \frac{1}{T} \sum_{t=1}^{T} b_t(x)$ — композиция алгоритмов.

Смещение композиции совпадает со смещением базового алгоритма:

$$bias = \mathrm{E}_{x,y}(\mathrm{E}_{X^n} b_t(x) - f^*(x))^2.$$

Разброс состоит из дисперсии и ковариации:

$$variance = \frac{1}{T} \mathrm{E}_{x,y} \mathrm{E}_{X^n}(b_t(x) - \mathrm{E}_{X^n} b_t(x))^2 +$$
$$+ \frac{T-1}{T} \mathrm{E}_{x,y} \mathrm{E}_{X^n}(b_t(x) - \mathrm{E}_{X^n} b_t(x))(b_s(x) - \mathrm{E}_{X^n} b_s(x)).$$

Таким образом, чем меньше коррелируют базовые алгоритмы, тем более эффективна их композиция.