Exercise 4. Task 7: Linear Model vs Regression Tree

Daniil Koveh

2025-11-04

Содержание

1 Теория **1**

2 Жизненный пример **1**

3 Академическое решение **1**
3.1 План решения 1
3.2 Многократная симуляция 3
3.3 Интерпретация 3
3.4 Что запомнить 4

1. Теория

При генерации линейных данных модель линейной регрессии является корректно специфицированной и должна превосходить деревья. Регрессионное дерево, даже с pruning, вынуждено аппроксимировать линию ступенчатой функцией, что увеличивает тестовую ошибку. Задача показывает ограничение деревьев в «гладких» сценариях.

2. Жизненный пример

Пусть x — количество часов, затраченных на проект, а y — результат. Связь линейна с небольшим шумом. Линейная модель точно описывает зависимость, дерево же делит часы на интервалы и присваивает им усреднённые результаты. Оно менее гибкое в линейной среде и проигрывает в точности.

3. Академическое решение

3.1. План решения

- Сгенерировать линейно зависимую выборку, обучить линейную регрессию и дерево, визуализировать их поведение на одном наборе.
- Повторить эксперимент много раз, чтобы сравнить распределения тестовых ошибок и размер дерева после pruning.
- Зафиксировать статистики (среднее, СКО) и интерпретировать, почему линейная модель стабильно выигрывает.

```
library(rpart) # дерево
library(ggplot2) # визуализация
```

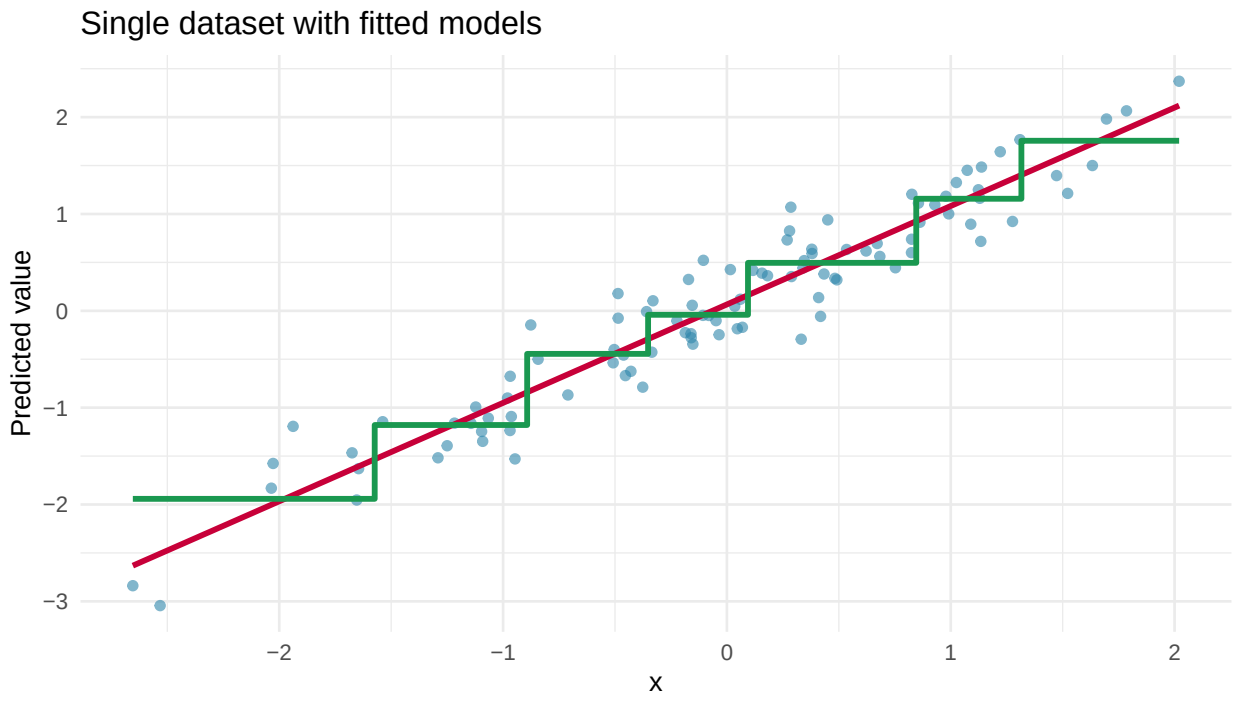
```
set.seed(20250410) # фиксируем генератор
n <- 100 # размер выборки
```

```r
sigma_eps <- sqrt(0.1) # стандартное отклонение шума
x <- rnorm(n) # генерируем x
y <- x + rnorm(n, sd = sigma_eps) # генерируем y
df <- data.frame(x = x, y = y) # собираем данные

lm_fit <- lm(y ~ x, data = df) # линейная модель
tree_fit <- rpart(y ~ x, data = df,
                  method = "anova",
                  control = rpart.control(cp = 0.001)) # дерево без ограничений
best_cp <- tree_fit$cptable[which.min(tree_fit$cptable[, "xerror"]), "CP"] # выбираем cp по минимуму xer
tree_pruned <- prune(tree_fit, cp = best_cp) # подрезаем дерево

new_x <- data.frame(x = seq(min(x), max(x), length.out = 200)) # сетка для прогнозов
pred_lm <- predict(lm_fit, newdata = new_x) # прогноз линейного
pred_tree <- predict(tree_pruned, newdata = new_x) # прогноз дерева

ggplot(df, aes(x = x, y = y)) +
  geom_point(alpha = 0.6, color = "#2E86AB") +
  geom_line(data = data.frame(x = new_x$x, y = pred_lm),
            aes(x = x, y = y), color = "#C70039", size = 1.1) +
  geom_step(data = data.frame(x = new_x$x, y = pred_tree),
            aes(x = x, y = y), color = "#1A9850", size = 1.1) +
  labs(title = "Single dataset with fitted models",
       y = "Predicted value") +
  theme_minimal()
```



3.2. Многократная симуляция

```
set.seed(20250410) # фиксируем генератор
simulate_once <- function(n = 100, sigma_eps = sqrt(0.1)) { # функция одного эксперимента
  x <- rnorm(n) # генерируем x
  y <- x + rnorm(n, sd = sigma_eps) # генерируем y
  train <- data.frame(x = x, y = y) # обучающая выборка
  test <- data.frame(x = rnorm(1000)) # создаём тестовую сетку
  test$y <- test$x + rnorm(1000, sd = sigma_eps) # добавляем шум к тестовым значениям

  lm_fit <- lm(y ~ x, data = train) # линейная модель
  lm_pred <- predict(lm_fit, newdata = test) # прогнозы
  lm_mse <- mean((lm_pred - test$y)^2) # MSE линейной модели

  tree_fit <- rpart(y ~ x, data = train,
                    method = "anova",
                    control = rpart.control(cp = 0.001)) # дерево
  best_cp <- tree_fit$cptable[which.min(tree_fit$cptable[, "xerror"]), "CP"] # лучший cp
  tree_pruned <- prune(tree_fit, cp = best_cp) # подрезка
  tree_pred <- predict(tree_pruned, newdata = test) # прогнозы дерева
  tree_mse <- mean((tree_pred - test$y)^2) # MSE дерева
  tree_size <- sum(tree_pruned$frame$var == "<leaf>") # размер дерева (число листьев)

  c(lm_mse = lm_mse, tree_mse = tree_mse, tree_size = tree_size) # возвращаем метрики
}

res_mat <- replicate(100, simulate_once()) # выполняем 100 повторов
summary_stats <- t(apply(res_mat, 1, function(x) c(mean = mean(x), sd = sd(x))))
tree_sizes <- res_mat["tree_size", ]
summary_stats
```

```
##                mean          sd
## lm_mse    0.1021376 0.005502468
## tree_mse  0.1748488 0.017813882
## tree_size 7.4100000 0.853927350
```

```
table(tree_sizes)
```

```
## tree_sizes
##  5  6  7  8  9 10
##  1 12 40 40  6  1
```

3.3. Интерпретация

- Средняя тестовая MSE линейной модели ≈ 0.102 при стандартном отклонении 0.006. Это совпадает с истинной дисперсией шума (0.1), поэтому модель практически воспроизводит генератор данных.
- Дерево (после оптимального pruning) даёт среднюю MSE ≈ 0.175 со стандартным отклонением 0.018, то есть стабильно хуже: кусочно-постоянная аппроксимация не умеет гладко следовать линейной тенденции.
- Размер дерева после pruning чаще всего 7 листа (см. таблицу выше); встречаются варианты с 2–4 листьями. Это означает, что дерево разбивает ось x на крупные блоки и усредняет в каждом блоке — отсюда повышенная ошибка.
- Вывод: в линейных сценариях дерева проигрывают независимо от настройки глубины. Pruning полезен лишь для контроля переобучения, но не исправляет принципиальный разрыв между «ступеньками» дерева и плавной линейной зависимостью.

3.4. Что запомнить

- Если данные действительно линейны, линейная модель остаётся эталоном: дерево лишь аппроксимирует её ступеньками.
- Pruning не исправляет фундаментальное несоответствие модели данным — он только контролирует переобучение.
- Анализ нескольких симуляций позволяет оценить стабильность выводов и убедиться, что результат не случайный.