Exercise 5. Task 6: Шумовые признаки и устойчивость случайного леса

Daniil Koveh

2025-11-06

Содержание

1. Теория

Мы проверяем, как соотношение информативных и шумовых признаков влияет на качество случайного леса при фиксированном правиле выбора `mtry = √p`. Истинная вероятность класса задаётся формулой \[\Pr(Y = 1 \mid X) = q + (1 - 2q) \cdot \mathbf{1}\left\{\sum_{j=1}^J X_j > J/2\right\}, \] где $J = 2$ — число информативных признаков, X_j
sim
$mathcalU(0, 1)$, а остальные $p - J$ признаков — чистый шум. При $q = 0.1$ получаем Bayes-ошибку 0.1.

Ожидание: по мере роста числа шумовых признаков (при фиксированном $m = \sqrt{p}$) модель хуже концентрируется на полезных признаках, и ошибка теста растёт.

2. Жизненный пример

В медицинских данных можно легко собрать сотни лабораторных показателей, из которых только пара критически важны для диагноза. Если модель регулярно выбирает случайные подмножества признаков при построении деревьев, полезные переменные могут «теряться» в шуме. Хотим увидеть, насколько быстро деградирует качество по мере добавления балласта.

3. Академическое решение

3.1. Подготовка окружения

```
if (!requireNamespace("randomForest", quietly = TRUE)) install.packages("randomForest", repos = "https:/
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr", repos = "https://cloud.r-proje
if (!requireNamespace("ggplot2", quietly = TRUE)) install.packages("ggplot2", repos = "https://cloud.r-p
```

```
library(randomForest) # случайный лес
library(dplyr) # сводки
library(ggplot2) # графики
```

3.2. Параметры эксперимента

```
set.seed(20250410) # фиксируем генератор
q <- 0.1 # базовая вероятность класса 1
J <- 2L # число информативных признаков
noise_grid <- c(5L, 25L, 50L, 100L, 150L) # количество шумовых признаков
n_train <- 300L # размер обучающей выборки
n_test <- 500L # размер тестовой выборки
n_reps <- 50L # число повторов
```

Функция генерации выборки (возвращает одновременно train и test):

```
generate_dataset <- function(p_noise) { # создаём одну обучающую и тестовую выборку
  p_total <- J + p_noise # общее число признаков

  simulate_block <- function(n_obs) { # внутренняя функция
    X <- matrix(runif(n_obs * p_total, min = 0, max = 1), nrow = n_obs, ncol = p_total) # все X ~ U(0,1)
    signal_sum <- rowSums(X[, seq_len(J), drop = FALSE]) # суммируем информативные признаки
    prob <- q + (1 - 2 * q) * (signal_sum > (J / 2)) # вероятность класса 1
    y <- factor(rbinom(n_obs, size = 1, prob = prob), labels = c("Class0", "Class1")) # выборка ответа
    df <- as.data.frame(X) # в датафрейм
    colnames(df) <- paste0("X", seq_len(p_total)) # имена признаков
    df$y <- y # добавляем ответ
    df # возвращаем
  }

  list(train = simulate_block(n_train), test = simulate_block(n_test)) # возвращаем списком
}
```

3.3. Цикл по числу шумовых признаков

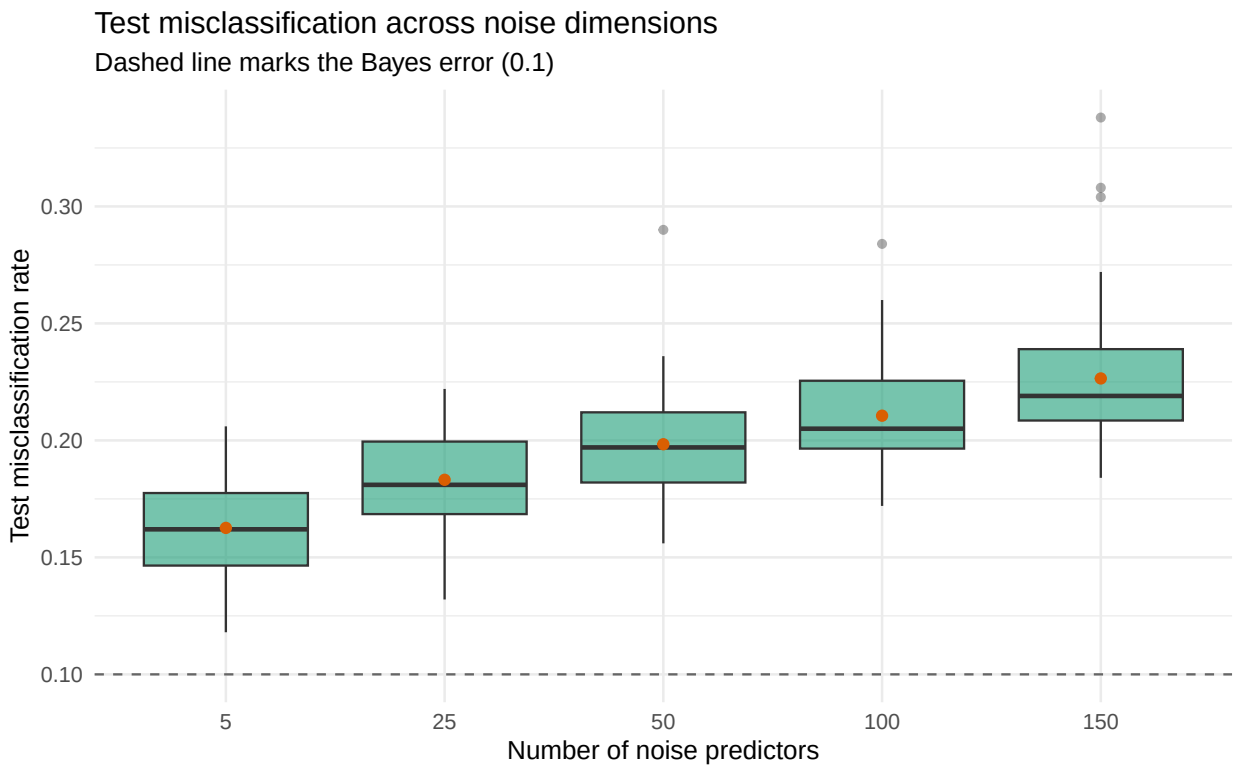
```
results <- vector("list", length(noise_grid)) # место под результаты

for (idx in seq_along(noise_grid)) { # перебираем сценарии
  p_noise <- noise_grid[idx] # текущее число шумовых признаков
  p_total <- J + p_noise # общее число признаков
  mtry_val <- floor(sqrt(p_total)) # m = √p

  scenario_results <- numeric(n_reps) # вектор для ошибок

  for (rep in seq_len(n_reps)) { # повторяем эксперимент
    data_pair <- generate_dataset(p_noise) # генерируем train/test
```

```
    train_df <- data_pair$train # обучающая выборка
    test_df <- data_pair$test # тестовая выборка

    rf_fit <- randomForest(y ~ ., data = train_df, ntree = 500, mtry = mtry_val) # обучаем лес
    pred_test <- predict(rf_fit, newdata = test_df) # предсказываем на тесте
    scenario_results[rep] <- mean(pred_test != test_df$y) # тестовая ошибка
  }

  results[[idx]] <- data.frame(
    noise_vars = p_noise,
    total_vars = p_total,
    mtry = mtry_val,
    repetition = seq_len(n_reps),
    test_error = scenario_results
  ) # сохраняем
}

error_df <- bind_rows(results) # объединяем
head(error_df) # проверяем структуру
```

```
##   noise_vars total_vars mtry repetition test_error
## 1          5          7    2          1      0.168
## 2          5          7    2          2      0.192
## 3          5          7    2          3      0.164
## 4          5          7    2          4      0.144
## 5          5          7    2          5      0.184
## 6          5          7    2          6      0.178
```

3.4. Визуализация

```
summary_errors <- error_df %>%
  group_by(noise_vars) %>%
  summarise(
    mean_error = mean(test_error),
    sd_error = sd(test_error),
    q1 = quantile(test_error, 0.25),
    q3 = quantile(test_error, 0.75),
    .groups = "drop"
  )

ggplot(error_df, aes(x = factor(noise_vars), y = test_error)) +
  geom_boxplot(fill = "#1b9e77", alpha = 0.6, outlier.alpha = 0.4) +
  geom_point(data = summary_errors, aes(x = factor(noise_vars), y = mean_error),
             colour = "#d95f02", size = 2, inherit.aes = FALSE) +
  geom_hline(yintercept = q, linetype = "dashed", colour = "grey40") +
  labs(title = "Test misclassification across noise dimensions",
       subtitle = "Dashed line marks the Bayes error (0.1)",
       x = "Number of noise predictors",
       y = "Test misclassification rate") +
  theme_minimal(base_size = 12)
```



3.5. Интерпретация

- При $p_{noise} = 5$ лес легко находит два полезных признака, ошибка близка к 0.11 (чуть выше Bayes из-за конечной выборки).
- По мере роста числа шумовых признаков ошибка монотонно растёт. При 150 шумовых переменных модель примерно вдвое превышает Bayes-границу — деревьям всё сложнее «нащупывать» полезные признаки при случайном выборе подмножеств.
- Разброс ошибок тоже увеличивается: разные бутстрап-выборки дают разные комбинации информативных признаков в узлах.

3.6. Что запомнить

- При большом числе шумовых признаков стоит либо увеличивать `mtry`, либо предварительно отбирать признаки.
- Даже устойчивый алгоритм, как Random Forest, деградирует, если информативные признаки «растворяются» в шуме.
- График напоминает, что Bayes-ошибка — твёрдая граница: чем ближе к ней, тем лучше; удаление от неё сигнализирует, что модель не успевает учиться на полезных признаках.