Конькова Нина Александровна, https://vk.com/id285308879
27 задание ЕГЭ

Средняя задача

Условие:
Вы являетесь разработчиком более сложной системы машинного обучения, которая классифицирует объекты. У вас есть набор данных, сохранённый в файле **data_medium.txt**. В этом файле каждая строка содержит два числа: первое число **x** — это целочисленное значение признака (например, вес объекта), второе число **y** — это метка класса (0 или 1). Найдите оптимальный порог **t**, который позволяет разделить объекты двух классов, минимизируя количество ошибок классификации, и посчитайте, сколько объектов было ошибочно классифицировано при использовании оптимального порога **t**. Выведите в ответ два числа: найденный порог t (если таких несколько, выведите порог с минимальным индексом) и количество ошибок классификации.

Объект считается ошибочно классифицированным, если:

- $x<t$, а $y=1$, или
- $x\geq t$, а $y=0$.

Пример содержимого файла data_medium.txt:
1 0
2 0
3 0
4 1
5 1
6 1

Пример вывода:
4
2

Решение:

1. Считываем данные полностью из файла и сразу сортируем их по значению признака x. Это упрощает расчёты и позволяет перебирать пороги без лишних операций.
2. Подсчитываем общее количество точек для каждого класса (count_0 для y=0, count+1 для y=1).
3. Используем нарастающие суммы для оптимизации подсчёта ошибок:
   - Left_0 и left_1 — количество точек классов y=0 и y=1 слева от текущего порога.
   - Right_0 и right_1 — остаток точек этих классов справа от порога, вычисляем через разность с общими счетчиками.
4. Для каждого значения x из данных считаем количество ошибок на текущем пороге: errors_left+errors_right. Если ошибок меньше, чем текущий минимум, обновляем минимальное число ошибок и запоминаем порог.
5. После обработки всех данных выводим оптимальный порог t и минимальное число

ошибок.

```python
def find_best_threshold_fast(file_path):
    # Шаг 1: Считываем данные
    with open(file_path, 'r') as file:
        data = [tuple(map(int, line.split())) for line in file]

    # Шаг 2: Сортируем данные по x
    data.sort()

    # Шаг 3: Подсчитываем общее количество классов
    count_0 = sum(1 for _, y in data if y == 0)
    count_1 = sum(1 for _, y in data if y == 1)

    # Шаг 4: Инициализация переменных
    min_errors = float('inf')
    best_t = None
    left_0, left_1 = 0, 0

    # Шаг 5: Проходим по данным и считаем ошибки на каждом пороге
    for i in range(len(data)):
        x, y = data[i]

        # Обновляем нарастающие суммы
        if y == 0:
            left_0 += 1
        else:
            left_1 += 1

        if i < len(data) - 1:
            next_x = data[i + 1][0]
            t = (x + next_x) // 2 + (x + next_x // 2) % 2
        else:
            t = x  # Для последней точки берём её значение

        # Подсчёт ошибок
        errors_left = left_1  # Ошибки в левой части (1 остались слева от t)
        errors_right = count_0 - left_0  # Ошибки в правой части (0 остались справа от t)
        total_errors = errors_left + errors_right

        # Сохраняем порог, если нашли меньше ошибок, или выбираем минимальный индекс при равных ошибках
        if total_errors < min_errors or (total_errors == min_errors and (best_t is None or t < best_t)):
            min_errors = total_errors
            best_t = t

    return best_t, min_errors

file_path = 'data_easy.txt'
best_t, min_errors = find_best_threshold_fast(file_path)

print(best_t, min_errors)
```

Листинг кода:

```
# Средний уровень
def find_best_threshold_fast(file_path):
    # Шаг 1: Считываем данные
    with open(file_path, 'r') as file:
        data = [tuple(map(int, line.split())) for line in file]

    # Шаг 2: Сортируем данные по x
```

```
    data.sort()

    # Шаг 3: Подсчитываем общее количество классов
    count_0 = sum(1 for _, y in data if y == 0)
    count_1 = sum(1 for _, y in data if y == 1)

    # Шаг 4: Инициализация переменных
    min_errors = float('inf')
    best_t = None
    left_0, left_1 = 0, 0

    # Шаг 5: Проходим по данным и считаем ошибки на каждом пороге
    for i in range(len(data)):
        x, y = data[i]

        # Обновляем нарастающие суммы
        if y == 0:
            left_0 += 1
        else:
            left_1 += 1

        if i < len(data) - 1:
            next_x = data[i + 1][0]
            t = (x + next_x) // 2 + (x + next_x // 2) % 2
        else:
            t = x  # Для последней точки берём её значение

        # Подсчёт ошибок
        errors_left = left_1  # Ошибки в левой части (1 остались слева от t)
        errors_right = count_0 - left_0  # Ошибки в правой части (0 остались справа от t)
        total_errors = errors_left + errors_right

        # Сохраняем порог, если нашли меньше ошибок, или выбираем минимальный индекс
при равных ошибках
        if total_errors < min_errors or (total_errors == min_errors and (best_t is None or t <
best_t)):
            min_errors = total_errors
            best_t = t

    return best_t, min_errors

# Пример вызова
file_path = 'data_medium.txt'
best_t, min_errors = find_best_threshold_fast(file_path)

print(best_t, min_errors)
```

Ответ для файла: 2184 9865