ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 4

« Задача отбора признаков для модели МО »

Выполнил работу

Карташов Игорь

Академическая группа №J3111

Принято

Вершинин Владислав

Санкт-Петербург

2024

Введение

Цель работы — выбрать минимальное количество признаков из исходного набора, при которых качество модели МО не ухудшится более чем на 10% по сравнению с вариантом, где в модели использованы все признаки.

Задачи:

1. Выбрать подходящий набор данных для задачи регрессии.
2. Разработать алгоритм перебора признаков, позволяющий определить их минимально необходимый набор.
3. Провести эксперименты и сравнить результаты по времени и точности с теоретическими оценками.

Теоретическая подготовка

Для решения задачи отбора признаков использовались подходы полного перебора комбинаций признаков. Поскольку метод полного перебора имеет экспоненциальную сложность O(2^n), на больших наборах данных он становится неэффективным, что нужно учесть при выполнении лабораторной работы.

Заданная модель машинного обучения принимает на вход данные и возвращает оценку эффективности, которая лежит в пределах от 0 до бесконечности. Качество работы модели оценивается с помощью метрики RMSE (Root Mean Squared Error), где низкое значение RMSE указывает на высокую точность модели.

Реализация

1. Загрузка и предобработка данных
   - Данные загружаются из CSV-файла с использованием библиотеки mlpack. Формат данных — матрица arma::mat. Она позволяет быстро и эффективно работать с данными.
   - Выбранные данные Concrete_Data_Yeh.csv содержат информацию о бетоне и о его прочности.

2. Основной цикл перебора
   - Используется цикл для перебора всех возможных комбинаций признаков, где количество комбинаций задано *2^n*, где *n*— количество признаков, за исключением целевой переменной.
   - Для каждой комбинации создается бинарное представление с помощью std::bitset, которое указывает, какие признаки включены в текущий набор.
   - Комбинации с количеством признаков 0 и 11 пропускаются, так как они не подходят для данной задачи.

3. Оценка качества комбинаций
   - Каждый поднабор признаков используется для создания подматрицы данных, которая затем подается на вход функции evaluate_dataset
   - Если результат по RMSE для поднабора признаков меньше порогового значения и имеет меньшее количество признаков по сравнению с предыдущими комбинациями, то он сохраняется как текущий лучший результат.

4. Вывод результатов

После перебора всех комбинаций выводится лучшая метрика RMSE на полном наборе признаков и на минимальном подмножестве признаков, а также индексы отобранных признаков.

Экспериментальная часть

Результаты эксперимента с набором данных Concrete_Data_Yeh показывают, что минимальное подмножество признаков, соответствующее допустимому уровню ошибки до 50%, включает пять признаков. RMSE при

использовании всех признаков и при отобранном подмножестве были получены следующие:

Число признаков	RMSE для полного набора	RMSE для сокращенного набора
8	107.197	155.287

Результаты эксперимента с набором данных WineQT показывают, что минимальное подмножество признаков, соответствующее допустимому уровню ошибки до 10%, включает пять признаков. RMSE при использовании всех признаков и при отобранном подмножестве были получены следующие:

Число признаков	RMSE для полного набора	RMSE для сокращенного набора
11	0.405982	0.431823

Подсчёт по памяти

Основные структуры данных:

- dataset хранит набор данных в виде матрицы arma::mat. В зависимости от числа строк m и столбцов n, общая память для dataset составляет m * n * 4
- IndsVector вектор, используемый для хранения индексов признаков в текущей комбинации. Его максимальный размер равен числу признаков, т.е. n*4
- binary объект std::bitset, который содержит до 16 бит, потребляет 2 байта.
- Indices вектор, в котором хранятся индексы строк для текущего поднабора признаков. Максимальный размер равен числу признаков: n * 4

- slice3 подматрица arma::mat на основе подмножества признаков. Размер матрицы зависит от числа выбранных признаков k и числа строк m, занимая m*k*4

Подсчёт памяти для циклов и структур:

- Внешний цикл по комбинациям O(2^n)
- Создание битовой строки и вектора индексов O(n) на каждой итерации
- Формирование подматрицы slice3 O(m*n) на каждой итерации
- Оценка модели O(m)

Итоговая сложность:

O(2^n * (n + m*n + m)) = O(2^n * m * n)

Заключение

В ходе выполнения работы мною был реализован алгоритм отбора признаков для модели МО с использованием метода полного перебора. Цель работы была достигнута путем нахождения минимального подмножества признаков, при котором качество модели ухудшилось не более чем на 50% от начального уровня. Полученные результаты совпадают с теоретическими оценками сложности алгоритма, подтверждая экспоненциальный рост времени выполнения при увеличении числа признаков

В качестве дальнейших исследований можно предложить оптимизацию алгоритма за счет использования жадных методов. Также можно попробовать применить корреляционный анализ для отбора признаков.

Приложения

Листинг кода файла feature_selection.cpp

```
/*
Как установить mlpack читать тут:
https://www.mlpack.org/doc/quickstart/cpp.html

Компиляция программы на mac m1:
g++-14 -O3 -std=c++17 -I/opt/homebrew/include -L/opt/homebrew/lib
feature_selection.cpp -o ./main -larmadillo -fopenmp
*/

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
```

```cpp
#include <vector>
#include <unordered_map>
#include <cmath>
#include <bitset>

// библиотеки которые необходи установить. Читать тут: https://www.mlpack.org/download.html
#define MLPACK_PRINT_INFO
#define MLPACK_PRINT_WARN
#include <mlpack.hpp>
#include <mlpack/methods/linear_regression/linear_regression.hpp>

// Файл в этой же папке, который я написал для задачи. Пока он обучает модель (черный ящик)
// и возвращает метрику качества (RMSE по умолчанию)
#include "modelling.hpp"

int feature_selection()
{
    // path to *.csv file
    // Будьте осторожны с путем. Сейчас вы в папке, но файл находится на уровень выше
    const char* path = "D:/Projects/mlpack_project/polygon/data/Concrete_Data_Yeh.csv";//"D:/Projects/mlpack_project/polygon/data/WineQT.csv"; //"../data/WineQT.csv";

    // Код, которым можно считать набор данных, который дальше требуется анализировать
    arma::mat dataset;
    if (!mlpack::data::Load(path, dataset)) {
        // not fatal and not transpose (если false, false, mlpack::data::FileType::AutoDetect)
        throw std::runtime_error("Could not read *.csv!");
    }
    // Инициализация переменных для оценки качества
    float best_score, score, optimal_score=100000;
    best_score = evaluate_dataset(dataset, 8);

    float porog_score = best_score * 1.5; // Порог для отбора признаков, тут я поставил 50 %, т.к. ниже нет таких комбинаций для моей даты

    std::vector<int> result_inds; // Вектор для хранения индексов лучших признаков
    int n=dataset.n_rows - 2; // Количество признаков (без целевой переменной)
    int num_combinations = std::pow(2,n); // Общее количество комбинаций
    int max_features_num = dataset.n_rows - 2;
```

```
    // Перебираем все возможные комбинации признаков
    for (int i=0; i < num_combinations; i++) {
        std::bitset<16> binary(i); // Бинарное представление комбинации
        std::vector<int> IndsVector; // Вектор индексов выбранных признаков
        std::string binary_digit = binary.to_string().substr(16 - n);
        int num_nozero_inds=0;

        // Переводим битовую строку в индексы признаков
        for (int j=0; j < binary_digit.size(); j++) {
            if (binary_digit[j]=='1') { // Если бит равен 1, добавляем признак
                IndsVector.push_back(j);
                num_nozero_inds += 1;
            }
        }

        // Пропускаем комбинации с нулем или максимальным количеством признаков
        if (IndsVector.size()==0 | IndsVector.size()==8 ) {
            continue;
        }

        IndsVector.push_back(8); // тут мы добавляем индекс таргета

        arma::uvec indices = arma::conv_to<arma::uvec>::from(IndsVector);
        arma::mat slice3 = dataset.rows(indices); // Отбираем нужные строки из
набора данных

        score = evaluate_dataset(slice3, num_nozero_inds);

        // Проверяем, лучше ли текущее качество и меньше ли количество признаков
        if (score <= porog_score) {
            if  (num_nozero_inds  <  max_features_num  ||  (num_nozero_inds  ==
max_features_num && score < optimal_score)) {
                max_features_num = num_nozero_inds;
                optimal_score = score;
                result_inds = IndsVector;
            }
        }
    }

    // Это просто пример вывода для отладки
    std::cout << "RMSE on all features: " << best_score << std::endl;
    std::cout << "RMSE on best min features: " << optimal_score << std::endl;
    for (int i=0; i < result_inds.size(); i++) {
        std::cout << i << " ";
    }

    return best_score;
```

```
}

void run_tests() {
    // Проверка, что код работает корректно
    feature_selection();
}

int main() {
    run_tests();

    return 0;
}
```

Листинг кода файла modelling.hpp

```
#ifndef MODELLING_HPP
#define MODELLING_HPP

#include <mlpack.hpp>
#include <vector>

// Объявления функций
arma::mat drop_columns(arma::mat &dataset, std::vector<int> indices);
float evaluate_dataset(arma::mat dataset, short int target_column_index, short
int id_column_index = -1);

#endif // MODELLING_HPP
```