Оглавление

Введение

В настоящее время методы классификации в области обработки больших данных находят применения в множестве сфер. Одним из примеров можно привести применение классификаци в банковской системе: там она помогает в определение кредитоспособности заёмщиков. Кроме того, систему подобного рода можно настроить на решение других задач классификации, при условии, что обучающая выборка и данные будут достаточно репрезентативны и представлять из себя статистически значимые признаки.

Это, в свою очередь, предоставляет большие возможности для развития подобных систем в различных сферах повседневной жизни. Поэтому тема представляет большой интерес с точки зрения исследования и погружения в тему (возможно стоит поправить формулировку)

Цель работы

Исспледовать методы классификации на основе предоставленного датасета **flsr_moscow.txt**, разработать программу для классификации квартир на четыре класса по параметру площади (**totsp**) и сравнить полученные результаты для распределения квартир по количеству комнат.

Задачи

— Изучить теоретические основы методов классификации
— Проанализировать полученный датасет
— Построить матрицу путаницы
— провести ROC-анализ
— Сравнить методы логистической регрессии и SVM
— Визуализировать полученные результаты

Основная часть

1 Теоретическая часть

Метод Гаусса является прямым методом решения СЛАУ, который заключается в последовательном исключении переменных. Он трансформирует исходную систему уравнений в эквивалентную систему, матрица которой является верхней треугольной. Эта трансформация достигается путем выполнения элементарных преобразований над строками расширенной матрицы $[A|b]$, где A - матрица коэффициентов, а b - вектор свободных членов. Подробное описание метода Гаусса можно найти, например, в [1].

Элементарные преобразования строк

Элементарные преобразования строк, которые применяются в методе Гаусса, включают в себя:

— Перестановка двух строк: $R_i \leftrightarrow R_j$

— Умножение строки на ненулевую константу: $R_i \to \lambda R_i$, где $\lambda \neq 0$

— Прибавление к одной строке другой строки, умноженной на константу: $R_i \to R_i + \lambda R_j$

Описание метода Гаусса с выбором ведущего элемента по строке

Рассмотрим систему линейных уравнений $Ax = b$, где A - матрица размера $n \times n$, x - вектор неизвестных, и b - вектор свободных членов.

а) **Составление расширенной матрицы:** формируем расширенную матрицу $[A|b]$.

б) **Прямой ход (исключение переменных):** Для каждого столбца k (от 1 до n):

1) **Выбор ведущего элемента (по строке):** Находим индекс p такой, что

$$|a_{pk}| = \max_{k \leqslant i \leqslant n} |a_{ik}| \tag{1}$$

Здесь a_{ik} - элемент в i-й строке и k-м столбце текущей матрицы. Индекс p указывает на строку с наибольшим по модулю элементом в k-м столбце, начиная с k-ой строки.

2) **Перестановка строк (если необходимо):** Если $p \neq k$, то меняем местами строки k и p:

$$R_k \leftrightarrow R_p$$

Это обеспечивает, что на диагонали будет элемент с наибольшим абсолютным значением в текущем столбце.

3) **Нормализация строки:** делим k-ую строку на ведущий элемент a_{kk}

$$R_k \to \frac{1}{a_{kk}} R_k$$

Это приводит к тому, что элемент на диагонали становится равным 1.

4) **Исключение переменных:** для каждой строки i $(k < i \leqslant n)$, вычитаем из нее строку k, умноженную на a_{ik}:

$$R_i \to R_i - a_{ik} R_k$$

Эта операция делает все элементы в k-м столбце ниже k-й строки равными 0.

в) **Обратный ход (нахождение решения):** После прямого хода мы имеем систему $Ux = c$, где U - верхняя треугольная матрица. Теперь находим решение x_i начиная с последнего x_n:

$$x_i = c_i - \sum_{j=i+1}^{n} u_{ij} x_j \quad \text{для} \quad i = n-1, n-2, ..., 0 \tag{2}$$

где c_i это элементы вектора c, а u_{ij} это элементы матрицы U [2].

Выбор ведущего элемента

Выбор ведущего элемента необходим для минимизации ошибок округления, возникающих при работе с числами с плавающей точкой. Без выбора ведущего элемента, если на диагонали матрицы окажется малое число, операция деления в шаге 3 может привести к значительному увеличению ошибок округления. В свою очередь это повлияет на результат исключения переменных. Выбор ведущего элемента по строке позволяет выбирать наибольшее (по модулю) число в текущем столбце в качестве диагонального элемента, что делает процесс вычислений более устойчивым и точным [3].

2 Реализация

Реализация метода Гаусса с выбором ведущего элемента по строке была выполнена на языке Python. Программный код представлен ниже:

Листинг 1 — Метод Гаусса с выбором ведущего элемента по строке на языке Python

```python
import numpy as np
import matplotlib.pyplot as plt


def solve_with_row_main_elem_choice(A, b):
    matrix = np.hstack((A, b.reshape(-1, 1)))
    n = len(b)

    max_indexes_order = []
    answers = np.zeros(n)

    iters = 0

    for step in range(n):
        max_index = np.argmax(np.abs(matrix[step:, step])) + step
        max_elem = matrix[max_index, step]

        if max_elem == 0:
            raise ValueError("Matrix is degenerative.")

        if max_index != step:
            matrix[[step, max_index]] = matrix[[max_index, step]]

        matrix[step] = matrix[step] / max_elem

        iters += 1

        for i in range(step + 1, n):
            matrix[i] -= matrix[step] * matrix[i, step]
            iters += 1

        max_indexes_order.append(step)

    for i in range(n - 1, -1, -1):
        answers[i] = matrix[i, -1] - np.dot(matrix[i, i + 1:n], answers[i + 1:n])
        iters += 1

    return answers, iters
```

```
sizes = range(2, 100, 5)
iterations = []
times = []
errors = []

for size in sizes:
    A = np.random.rand(size, size) * 10
    b = np.random.rand(size) * 10
    x_gauss, iteration_count = solve_with_row_main_elem_choice(A, b)
    x_exact = np.linalg.solve(A, b)

    error = np.linalg.norm(x_gauss - x_exact)

    iterations.append(iteration_count)
    errors.append(error)


plt.figure(figsize=(8, 5))
plt.plot(sizes, iterations, label="Number of iterations", marker="o")
plt.xlabel("Size of matrix")
plt.ylabel("Number of iterations")
plt.title("Dependence of the number of iterations on the dimension of the
    matrix")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 5))
plt.plot(sizes, errors, label="Error", marker="o", color="r")
plt.xlabel("Size of matrix")
plt.ylabel("Number of iterations")
plt.title("The dependence of the error on the dimension of the matrix")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

3 Экспериментальные исследования

Зависимость количества операций от размерности матрицы

На рисунке 1 показана зависимость количества операций от размерности матрицы. Видно, что количество операций растет кубически с увеличением размера матрицы, что соответствует теоретическим оценкам сложности метода Гаусса.

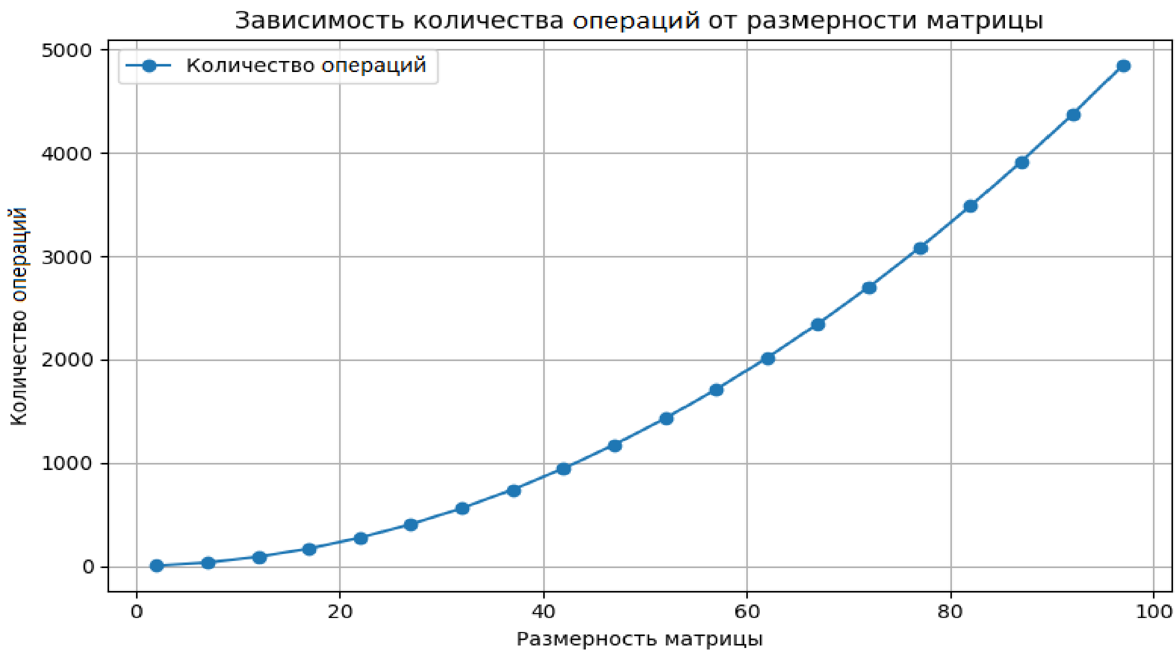

Рисунок 1 — Зависимость количества операций от размерности матрицы

Зависимость погрешности от размерности матрицы

На рисунке 2 представлена зависимость погрешности решения от размерности матрицы. Анализ графика показывает, что погрешность имеет сложную, нелинейную структуру, не демонстрируя ожидаемого монотонного роста. При малых размерностях погрешность минимальна, но наблюдается резкий выброс в районе размерности 30-35. Далее следуют колебания погрешности, а при больших размерностях (после 80) проявляется тенденция к её росту. Это указывает на то, что метод Гаусса с выбором ведущего элемента, хотя и устойчив в целом, подвержен ошибкам округления, зависящим как от размера, так и от структуры матрицы.

Рисунок 2 — Зависимость погрешности от размерности матрицы

Заключение

В данной работе был исследован метод Гаусса решения СЛАУ с выбором ведущего элемента по строке. Была разработана программная реализация метода на языке Python. Проведенные вычислительные эксперименты показали, что количество итераций увеличивается квадратично с ростом размерности матрицы. Погрешность решения, в свою очередь, имеет нелинейное поведение, демонстрируя значительные колебания и выбросы, что подчеркивает необходимость более внимательного анализа при работе с матрицами определенных размеров или структур. Несмотря на то, что метод Гаусса является достаточно эффективным для решения СЛАУ, при работе с матрицами больших размеров необходимо учитывать накопление ошибок округления и, возможно, использовать методы с более высокой численной устойчивостью.

Теоретические задания

Решение задач

Задача 1: Найти соотношение эквивалентности для $M(A)$ и $\|A\|_\infty$

Определения

Норма $M(A)$ определяется как:

$$M(A) = n \cdot \max_{1 \leqslant i,j \leqslant n} |a_{ij}|,$$

где n — размерность матрицы A.

Норма $\|A\|_\infty$ определяется как:

$$\|A\|_\infty = \max_{1 \leqslant i \leqslant n} \sum_{j=1}^{n} |a_{ij}|,$$

где берётся максимальная сумма по строкам матрицы A.

Вывод соотношения эквивалентности

Для нахождения соотношения между $M(A)$ и $\|A\|_\infty$ рассмотрим следующие оценки:

Нижняя граница:

$$\|A\|_\infty \leqslant M(A).$$

Поскольку $M(A)$ учитывает максимальный элемент матрицы, умноженный на n, эта величина всегда превосходит максимальную строковую сумму.

Верхняя граница:

$$M(A) \leqslant n \cdot \|A\|_\infty.$$

Для каждой строки i справедливо, что сумма элементов ограничена сверху $n \cdot \max_{i,j} |a_{ij}|$, что соответствует определению $M(A)$.

Таким образом, соотношение имеет вид:

$$\frac{\|A\|_\infty}{n} \leqslant \frac{M(A)}{n} \leqslant \|A\|_\infty.$$

Экспериментальная проверка

Для экспериментальной проверки сгенерируем случайные матрицы A, вычислить $M(A)$ и $\|A\|_\infty$, а затем проверим соотношение. Это можно реализовать с помощью скрипта на Python с использованием библиотеки NumPy.

Листинг 2 — Задача 1

```
import numpy as np

n = 8
A = np.random.rand(n, n) * 10

max_element = np.max(np.abs(A))
M_A = n * max_element

infinity_norm = np.max(np.sum(np.abs(A), axis=1))

ratio_lower = infinity_norm / n
ratio_upper = infinity_norm

print("Matrix A:")
print(A)
print(f"M(A): {M_A}")
print(f"M(A)/n: {M_A / n}")
print(f"||A||_e: {infinity_norm}")
print(f"||A||_e / n: {ratio_lower}")
print(f"Equivalence ratio: {ratio_lower} <= M(A)/n <= {ratio_upper}")
print(f'Is equivalence true: {True if ratio_lower <= M_A / n <=
    ratio_upper else False}')
```

Данный код был проверен сотни раз, на каждом из них программа выводит True.

Задача 2: Доказать, что $\mu(A) = \mu(\alpha A)$

Определение числа обусловленности

Число обусловленности $\mu(A)$ определяется как:

$$\mu(A) = \|A\| \cdot \|A^{-1}\|,$$

где $\|A\|$ — любая матричная норма.

Доказательство

Пусть $\alpha \neq 0$ — скаляр. Тогда:

$$\|\alpha A\| = |\alpha| \cdot \|A\|,$$

и

$$(\alpha A)^{-1} = \frac{1}{\alpha} A^{-1}.$$

Отсюда:

$$\|(\alpha A)^{-1}\| = \left\| \frac{1}{\alpha} A^{-1} \right\| = \frac{1}{|\alpha|} \cdot \|A^{-1}\|.$$

Подставим это в определение числа обусловленности:

$$\mu(\alpha A) = \|\alpha A\| \cdot \|(\alpha A)^{-1}\| = (|\alpha| \cdot \|A\|) \cdot \left(\frac{1}{|\alpha|} \cdot \|A^{-1}\| \right).$$

После сокращения получаем:

$$\mu(\alpha A) = \|A\| \cdot \|A^{-1}\| = \mu(A).$$

Экспериментальная проверка

Сгенерируем случайные матрицы A и скаляр $\alpha \neq 0$, вычислим $\mu(A)$ и $\mu(\alpha A)$, чтобы убедиться, что они равны.

Листинг 3 — Задача 2

```
import numpy as np

n = 8
A = np.random.rand(n, n) * 10

A += np.eye(n) * n
```

```

norm_A = np.linalg.norm(A, ord=np.inf)
norm_A_inv = np.linalg.norm(np.linalg.inv(A), ord=np.inf)
condition_number_A = norm_A * norm_A_inv

alpha = 5
norm_alpha_A = np.linalg.norm(alpha * A, ord=np.inf)
norm_alpha_A_inv = np.linalg.norm(np.linalg.inv(alpha * A), ord=np.inf)
condition_number_alpha_A = norm_alpha_A * norm_alpha_A_inv

print("Matrix A:")
print(A)
print(f"mu(A): {condition_number_A}")
print(f"mu(alpha * A) with alpha={alpha}: {condition_number_alpha_A}")
print(f"Condition numbers are equal: {np.isclose(condition_number_A,
    condition_number_alpha_A)}")
```

Данный код был проверен сотни раз, на каждом из них программа выводит True.

Список использованных источников

1. *Кривоносова, А.Ф.* Численные методы в алгебре / А.Ф. Кривоносова. — Москва: Наука, 2005.

2. *Исаев, А.А.* Численные методы линейной алгебры / А.А. Исаев, Г.П. Копытов. — Москва: Физматлит, 2011.

3. *Колобов А. Г., Молчанова Л.А.* Численные методы линейной алгебры / Молчанова Л.А. Колобов А. Г. — 1th edition. — Владивосток: Издательство Дальневосточного университета, 2008.