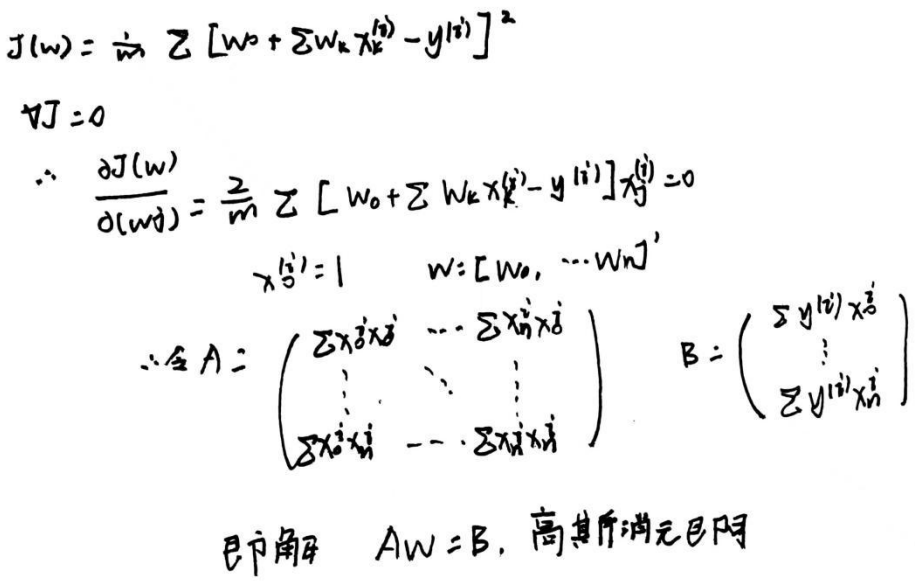
21374389 牛鹏军 210711

理论分析

针对数据集 LR_Data.txt，利用线性方程组求解算法，构造线性回归模型： y=w0+w1*x1+w2*x2+w3*x3


按照上式，读入 x 与 y 后，求出 A 与 B，利用高斯消元法解ω。

算法设计

读取文件中 X 与 y 后，利用 for 循环构造 A 与 B 矩阵，代入高斯消元法求解ω，其中高斯消元法采用第 5 周作业中直接封装好的函数。

编程实现

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void gaussianElimination(double** matrix, double* vector, int n, double *w) {
    for (int k = 0; k < n - 1; k++) {
        for (int i = k + 1; i < n; i++) {
            double factor = matrix[i][k] / matrix[k][k];

            for (int j = k; j < n; j++) {
                matrix[i][j] -= factor * matrix[k][j];
```

```
                }

                vector[i] -= factor * vector[k];
            }
        }

        double* solution = (double*)malloc(n * sizeof(double));

        for (int i = n - 1; i >= 0; i--) {
            double sum = 0.0;

            for (int j = i + 1; j < n; j++) {
                sum += matrix[i][j] * solution[j];
            }

            solution[i] = (vector[i] - sum) / matrix[i][i];
        }

        printf("参数向量  w:\n");

        for (int i = 0; i < n; i++) {
            printf("w%d = %lf\n", i, solution[i]);
            w[i] = solution[i];
        }

        free(solution);
}

int main() {
        FILE* file = fopen("LR_Data.txt", "r");
        if (file == NULL) {
            printf("无法打开文件  LR_Data.txt\n");
            return 1;
        }

        int numSamples, numFeatures;
        fscanf(file, "%d", &numSamples);
        fscanf(file, "%d", &numFeatures);
        numFeatures++;

        double** X = (double**)malloc(numSamples * sizeof(double*));
        for (int i = 0; i < numSamples; i++) {
            X[i] = (double*)malloc(numFeatures * sizeof(double));
            X[i][0] = 1;
```

```
    }

    double* y = (double*)malloc(numSamples * sizeof(double));

    for (int i = 0; i < numSamples; i++) {
        fscanf(file, "%lf", &y[i]);
        for (int j = 1; j < numFeatures; j++) {
            fscanf(file, "%lf", &X[i][j]);
        }
    }

    fclose(file);

    double** A = (double**)malloc(numFeatures * sizeof(double*));
    for (int i = 0; i < numFeatures; i++) {
        A[i] = (double*)malloc(numFeatures * sizeof(double));
        for (int j = 0; j < numFeatures; j++) {
            double temp = 0.0;
            for (int p = 0; p < numSamples; p++) {
                temp += X[p][j] * X[p][i];
            }
            A[i][j] = temp;
        }
    }

    double* B = (double*)malloc(numFeatures * sizeof(double));
    for (int i = 0; i < numFeatures; i++) {
        double temp = 0.0;
        for (int j = 0; j < numSamples; j++) {
            temp += y[j] * X[j][i];
        }
        B[i] = temp;
    }

    double *w;
    w = (double*)malloc(numFeatures * sizeof(double));
    gaussianElimination(A, B, numFeatures, w);

    for (int i = 0; i < numSamples; i++) {
        double ans = w[0] + w[1] * X[i][1] + w[2] * X[i][2] + w[3] * X[i][3];
        printf("The y%-2d is %7.4lf.    ", i + 1, y[i]);
        printf("The y%-2d predicted is %7.4lf.    ", i + 1, ans);
        printf("Their difference is %.4lf.\n", fabs(ans - y[i]));
```

```
    }

    for (int i = 0; i < numFeatures; i++) {
        free(A[i]);
    }
    free(A);
    free(B);
    for (int i = 0; i < numSamples; i++) {
        free(X[i]);
    }
    free(X);
    free(y);

    return 0;
}
```

测试分析

```
参数向量 w:
w0 = 14.115227
w1 = 0.000160
w2 = 0.028847
w3 = -0.002731
The y1  is 14.3900.  The y1  predicted is 12.7160.  Their difference is 1.6740.
The y2  is 12.9800.  The y2  predicted is 12.5429.  Their difference is 0.4371.
The y3  is 11.6000.  The y3  predicted is 12.3074.  Their difference is 0.7074.
The y4  is 11.4500.  The y4  predicted is 12.0050.  Their difference is 0.5550.
The y5  is 11.2100.  The y5  predicted is 11.4795.  Their difference is 0.2695.
The y6  is 10.5500.  The y6  predicted is 10.4175.  Their difference is 0.1325.
The y7  is 10.4200.  The y7  predicted is  9.6352.  Their difference is 0.7848.
The y8  is 10.0600.  The y8  predicted is  9.1785.  Their difference is 0.8815.
The y9  is  9.1400.  The y9  predicted is  8.8435.  Their difference is 0.2965.
The y10 is  8.1800.  The y10 predicted is  8.7095.  Their difference is 0.5295.
The y11 is  7.5800.  The y11 predicted is  8.3789.  Their difference is 0.7989.
The y12 is  6.9500.  The y12 predicted is  7.9152.  Their difference is 0.9652.
The y13 is  6.4500.  The y13 predicted is  7.5208.  Their difference is 1.0708.
The y14 is  6.0100.  The y14 predicted is  7.0319.  Their difference is 1.0219.
The y15 is  5.8700.  The y15 predicted is  6.1245.  Their difference is 0.2545.
The y16 is  5.8900.  The y16 predicted is  5.3386.  Their difference is 0.5514.
The y17 is  5.3800.  The y17 predicted is  4.5680.  Their difference is 0.8120.
The y18 is  5.2400.  The y18 predicted is  4.0120.  Their difference is 1.2280.
The y19 is  5.4500.  The y19 predicted is  6.0751.  Their difference is 0.6251.

--------------------------------
Process exited after 0.1542 seconds with return value 0
请按任意键继续. . .
```

将 LR_Data.txt 文件放在源代码同一目录下，最终得到ω向量各值为：14.115227，0.000160，0.028847，-0.002731。最终用得到的回归方程预测值与实际值对比，相差最大为 1.6740，最小为 0.1325，拟合的还可以。

结论

线性回归是一种数理统计中常用的机器学习算法，这种算法整体相对简单，在线性较好的数据上比较适用。