**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 |
| 0b13a601adf0cd662f84e07bd9d9909 |

按照上式，读入x与y后，求出A与B，利用高斯消元法解。

**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，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**

|  |
| --- |
|  |

将LR\_Data.txt文件放在源代码同一目录下，最终得到向量各值为：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。