

原

高斯消元(and牛顿迭代法)详解及模板

2018年07月26日 15:37:18 bestsort 阅读数 1113

版权声明：本文为博主原创文章，转载注明出自CSDN bestsort。 <https://blog.csdn.net/bestsort/article/details/81221258>

高斯消元的最主要作用为求解线性方程组，说白了，就是解方程.....
解方程还有一种为牛顿迭代法，我们每次枚举一个值 X_0 ，代入方程看是否为根，不是的话则将 X_0 的值变为：
 $X_0 = X_0 - F(X_0)/F'(X_0)$ (其中 $F'(x)$ 为 $F(x)$ 的导数)
其证明过程借用了高数中的基本知识泰勒级数，此处不再赘述。
有趣的是，在 雷神之锤III 中,卡马克用牛顿迭代法,令 $X_0 = 0x5f3759df$ 后求出数字开根后的倒数的速度居然比库函数快4倍!而这个数字也成为了神奇

消元的过程就是模拟人手算解方程的过程

其算法过程就2个

- 1. 代入消元
- 2. 将算出的值代回，最终求得所有未知数的解

$$\begin{cases} x + 2y + z = 7 & A \\ 2x - y + 3z = 7 & B \\ 3x + y + 2z = 18 & C \end{cases}$$

通俗点，比如说对于这个方程组
第一个过程是将其中的一个变量消元化为 $x=num \ || \ y=num \ || \ z=num$ 的形式，其中 num 为常数
第二个过程就是将这个已经求出来的 $x \ || \ y \ || \ z$ 带回到原方程组中求出剩余两个未知数点值:

这里我们用矩阵模拟这个消元回代的过程
还是用上面的那个方程做例子

$$\begin{cases} x + 2y + z = 7 & A \\ 2x - y + 3z = 7 & B \\ 3x + y + 2z = 18 & C \end{cases}$$

转化为矩阵后为:

$$\begin{vmatrix} 1 & 2 & 1 & 7 \\ 2 & -1 & 3 & 7 \\ 3 & 1 & 2 & 18 \end{vmatrix}$$

其中最后一列为等号右边的值，设从上到下每行依次为 R_1,R_2,R_3 ,从左到右每一列依次为 x,y,z 的系数;

需要注意的是这里我们要把 x 最大的放在第一行，这里是为了在进行除法点时候减小误差
现在开始化简:

$$\begin{vmatrix} 3 & 1 & 2 & 18 \\ 0 & -5 & 5 & -15 \\ 1 & 2 & 1 & 7 \end{vmatrix}$$

3*R2-2*R1 ->

$$\begin{vmatrix} 3 & 1 & 2 & 18 \\ 0 & -5 & 5 & -15 \\ 0 & 5 & 1 & 3 \end{vmatrix}$$

3*R3-R1->

$$\begin{vmatrix} 3 & 1 & 2 & 18 \\ 0 & -5 & 5 & -15 \\ 0 & 0 & -6 & 12 \end{vmatrix}$$

R3*(-1)-R2->

到这里已经可消元完毕了，对应式子如下:

$$\begin{cases} 3x + y + 2z = 18 \\ -5y + 5z = -15 \\ -6z = 12 \end{cases}$$

然后就是回代求解了，这里就不细说了，解方程大家都会

以下模板来自kuangbin

```

1  #include<stdio.h>
2  #include<algorithm>
3  #include<iostream>
4  #include<string.h>
5  #include<math.h>
6  using namespace std;
7  const int MAXN=50;
8  int a[MAXN][MAXN]; //增广矩阵
9  int x[MAXN]; //解集
10 bool free_x[MAXN]; //标记是否是不确定的变元
11 inline int gcd(int a,int b)
12 {
13     int t;
14     while(b!=0)
15     {
16         t=b;
17         b=a%b;
18         a=t;
19     }
20     return a;
21 }
22 inline int lcm(int a,int b)
23 {
24     return a/gcd(a,b)*b; //先除后乘防溢出
25 }
26
27 // 高斯消元法解方程组(Gauss-Jordan elimination).(-2表示有浮点数解，但无整数解，
28 // -1表示无解，0表示唯一解，大于0表示无穷解，并返回自由变元的个数)
29 // 有equ个方程，var个变元。增广矩阵行数为equ,分别为0到equ-1,列数为var+1,分别为0到var.
30 int Gauss(int equ,int var)
31 {
32     int i,j,k;
33     int max_r; // 当前这列绝对值最大的行.
34     int col; // 当前处理的列
35     int ta,tb;
36     int LCM;
37     int temp;
38     int free_x_num;
39     int free_index;
40
41     for(int i=0; i<=var; i++)
42     {
43         x[i]=0;
44         free_x[i]=true;
45     }
46
47     // 转换为阶梯阵.
48     col=0; // 当前处理的列
49     for(k=0; k<equ && col<var; k++,col++)
50     {
51         // 枚举当前处理的行.
52         // 找到该col列元素绝对值最大的那行与第k行交换.(为了在除法时减小误差)
53         max_r=k;
54         for(i=k+1; i<equ; i++)
55         {
56             if(abs(a[i][col])>abs(a[max_r][col])) max_r=i;
57         }
58         if(max_r!=k)
59         {
60             // 交换行
61             for(j=0; j<var+1; j++)

```



3



```

63     for(j=k; j<var+1; j++) swap(a[k][j],a[max_r][j]);
64 }
65 if(a[k][col]==0)
66 {
67     // 说明该col列第k行以下全是0了, 则处理当前行的下一列.
68     k--;
69     continue;
70 }
71 for(i=k+1; i<equ; i++)
72 {
73     // 枚举要删去的行.
74     if(a[i][col]!=0)
75     {
76         LCM = lcm(abs(a[i][col]),abs(a[k][col]));
77         ta = LCM/abs(a[i][col]);
78         tb = LCM/abs(a[k][col]);
79         if(a[i][col]*a[k][col]<0)tb=-tb;    //异号的情况是相加
80         for(j=col; j<var+1; j++)
81         {
82             a[i][j] = a[i][j]*ta-a[k][j]*tb;
83         }
84     }
85 }
86 }
87
88 // 1. 无解的情况: 化简的增广阵中存在(0, 0, ..., a)这样的行(a != 0).
89 for (i = k; i < equ; i++)
90 {
91     // 对于无穷解来说, 如果要判断哪些是自由变元, 那么初等行变换中的交换就会影响, 则要记录交换.
92     if (a[i][col] != 0) return -1;
93 }
94 // 2. 无穷解的情况: 在var * (var + 1)的增广阵中出现(0, 0, ..., 0)这样的行, 即说明没有形成严格的上三角阵.
95 // 且出现的行数即为自由变元的个数.
96 if (k < var)
97 {
98     // 首先, 自由变元有var - k个, 即不确定的变元至少有var - k个.
99     for (i = k - 1; i >= 0; i--)
100     {
101         // 第i行一定不会是(0, 0, ..., 0)的情况, 因为这样的行是在第k行到第equ行.
102         // 同样, 第i行一定不会是(0, 0, ..., a), a != 0的情况, 这样的无解的.
103         free_x_num = 0; // 用于判断该行中的不确定的变元的个数, 如果超过1个, 则无法求解, 它们仍然为不确定的变元.
104         for (j = 0; j < var; j++)
105         {
106             if (a[i][j] != 0 && free_x[j]) free_x_num++, free_index = j;
107         }
108         if (free_x_num > 1) continue; // 无法求解出确定的变元.
109         // 说明就只有一个不确定的变元free_index, 那么可以求解出该变元, 且该变元是确定的.
110         temp = a[i][var];
111         for (j = 0; j < var; j++)
112         {
113             if (a[i][j] != 0 && j != free_index) temp -= a[i][j] * x[j];
114         }
115         x[free_index] = temp / a[i][free_index]; // 求出该变元.
116         free_x[free_index] = 0; // 该变元是确定的.
117     }
118     return var - k; // 自由变元有var - k个.
119 }
120 // 3. 唯一解的情况: 在var * (var + 1)的增广阵中形成严格的上三角阵.
121 // 计算出Xn-1, Xn-2 ... X0.
122 for (i = var - 1; i >= 0; i--)
123 {
124     temp = a[i][var];
125     for (j = i + 1; j < var; j++)
126     {
127         if (a[i][j] != 0) temp -= a[i][j] * x[j];    //--因为x[j]存的是temp/a[i][j]的值, 即是a[i][j]=1时x[j]对应的值
128     }
129     if (temp % a[i][i] != 0) return -2; // 说明有浮点数解, 但无整数解.
130     x[i] = temp / a[i][i];
131 }
132

```



```
134     return 0;
135 }
136 int main(void)
137 {
138     freopen("in.txt", "r", stdin);
139     freopen("out.txt", "w", stdout);
140     int i, j;
141     int equ,var;
142     while (scanf("%d %d", &equ, &var) != EOF)
143     {
144         memset(a, 0, sizeof(a));
145         for (i = 0; i < equ; i++)
146         {
147             for (j = 0; j < var + 1; j++)
148             {
149                 scanf("%d", &a[i][j]);
150             }
151         }
152         int free_num = Gauss(equ,var);
153         if (free_num == -1) printf("无解!\n");
154         else if (free_num == -2) printf("有浮点数解, 无整数解!\n");
155         else if (free_num > 0)
156         {
157             printf("无穷多解! 自由变元个数为%d\n", free_num);
158             for (i = 0; i < var; i++)
159             {
160                 if (free_x[i]) printf("x%d 是不确定的\n", i + 1);
161                 else printf("x%d: %d\n", i + 1, x[i]);
162             }
163         }
164         else
165         {
166             for (i = 0; i < var; i++)
167             {
168                 printf("x%d: %d\n", i + 1, x[i]);
169             }
170             printf("\n");
171         }
172     }
173     return 0;
174 }
```

👍

3

💬

🔖

📄

<

>

转载注明出处，谢谢合作

白开水+它，喝完排出“巨便”，大肚子没了

舜飞

想对作者说点什么

高斯消元算法

阅读数 613

一、基本描述 高斯消元主要用来求解线性方程组，也可以求解矩阵的秩、矩阵的逆。它的时间复杂度是 n^3 ，主要... 博文 来自: mathor的博客

高斯消元

阅读数 739

题目背景Gauss消元题目描述给定一个线性方程组，对其求解输入输出格式输入格式： 第一行，一个正整数 n第二至 ... 博文 来自: sslz_fsy的博客

高斯消元快速入门

阅读数 2万+

高斯消元快速入门一、基本描述学习一个算法/技能，首先要知道它是干什么的，那么高斯消元是干啥的呢？高斯消... 博文 来自: Pengwill's Blog

高斯消元法

阅读数 1937

高斯消去法是一种常用的求解线性方程组的方法，通过逐次消元后，在回代求解，实际计算中常用的一种方法。顺序... 博文 来自: tyxr

华为云年中大促 消费送P30 [广告][关闭]

https://blog.csdn.net/bestsort/article/details/81221258

4/11