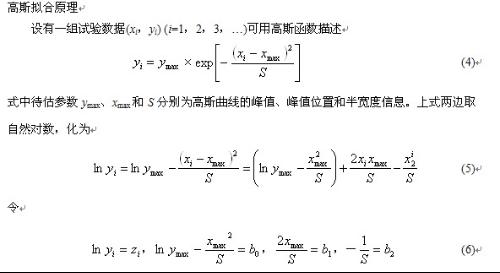
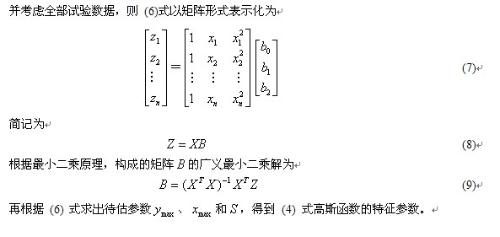
高斯[拟合](http://www.baidu.com/s?wd=%E6%8B%9F%E5%90%88&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)(Gaussian Fitting)即使用形如：  
   
 Gi(x)=Ai\*exp((x-Bi)^2/Ci^2)  
  
 的[高斯函数](http://www.baidu.com/s?wd=%E9%AB%98%E6%96%AF%E5%87%BD%E6%95%B0&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)对数据点集进行函数逼近的[拟合](http://www.baidu.com/s?wd=%E6%8B%9F%E5%90%88&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)方法。  
  
 其实可以跟[多项式](http://www.baidu.com/s?wd=%E5%A4%9A%E9%A1%B9%E5%BC%8F&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)[拟合](http://www.baidu.com/s?wd=%E6%8B%9F%E5%90%88&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)类比起来，不同的是[多项式](http://www.baidu.com/s?wd=%E5%A4%9A%E9%A1%B9%E5%BC%8F&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)拟合是用[幂函数](http://www.baidu.com/s?wd=%E5%B9%82%E5%87%BD%E6%95%B0&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)系，  
 而高斯拟合是用[高斯函数](http://www.baidu.com/s?wd=%E9%AB%98%E6%96%AF%E5%87%BD%E6%95%B0&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)系。  
  
 使用[高斯函数](http://www.baidu.com/s?wd=%E9%AB%98%E6%96%AF%E5%87%BD%E6%95%B0&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)来进行拟合，优点在于计算积分十分简单快捷。这一点  
 在很多领域都有应用，特别是[计算化学](http://www.baidu.com/s?wd=%E8%AE%A1%E7%AE%97%E5%8C%96%E5%AD%A6&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)。著名的[化学软件](http://www.baidu.com/s?wd=%E5%8C%96%E5%AD%A6%E8%BD%AF%E4%BB%B6&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)Gaussian98  
 就是建立在高斯[基函数](http://www.baidu.com/s?wd=%E5%9F%BA%E5%87%BD%E6%95%B0&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)拟合的[数学基础](http://www.baidu.com/s?wd=%E6%95%B0%E5%AD%A6%E5%9F%BA%E7%A1%80&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)上的。





**高斯拟合算法解决思路**

www.MyException.Cn   发布于：2014-02-25 23:11:14   浏览：0次

http://bdimg.share.baidu.com/static/images/type-button-1.jpg?cdnversion=201208310

高斯拟合算法  
#include<iostream.h>  
#include<math.h>  
#include<stdlib.h>  
#include <windows.h>  
double f(int n,double x){       //f(n,x)用来返回x的n次方  
 double y=1.0;  
 if(n==0)return 1.0;  
 else{  
 for(int i=0;i<n;i++)y\*=x;  
 return y;  
 }  
}  
int xianxingfangchengzu(double \*\*a,int n,double \*b,double \*p,

double dt)//用高斯列主元法来求解法方程组  
 {  
   int i,j,k,l;  
   double c,t;  
   for(k=1;k<=n;k++)  
   {  
   c=0.0;  
   for(i=k;i<=n;i++)  
    if(fabs(a[i-1][k-1])>fabs(c))  
    {  
    c=a[i-1][k-1];  
    l=i;  
    }if(fabs(c)<=dt)  
    return(0);  
    if(l!=k)  
    {  
    for(j=k;j<=n;j++)  
    {  
    t=a[k-1][j-1];  
    a[k-1][j-1]=a[l-1][j-1];  
    a[l-1][j-1]=t;  
    }  
    t=b[k-1];  
    b[k-1]=b[l-1];  
    b[l-1]=t;  
    }  
    c=1/c;  
    for(j=k+1;j<=n;j++)  
    {  
    a[k-1][j-1]=a[k-1][j-1]\*c;  
    for(i=k+1;i<=n;i++)  
    a[i-1][j-1]-=a[i-1][k-1]\*a[k-1][j-1];  
    }  
    b[k-1]\*=c;  
     for(i=k+1;i<=n;i++)  
 b[i-1]-=b[k-1]\*a[i-1][k-1];  
    }  
    for(i=n;i>=1;i--)  
    for(j=i+1;j<=n;j++)  
    b[i-1]-=b[j-1]\*a[i-1][j-1];  
      
   cout.precision(12);  
   for(i=0;i<n;i++)p[i]=b[i];  
}  
double\*\* create(int a,int b)//动态生成数组  
{  
 double \*\*P=new double \*[a];  
 for(int i=0;i<b;i++)  
 P[i]=new double[b];  
 return P;  
}  
void zuixiaoerchengnihe(double x[],double y[],int n,double a[],int m)  
{  
 int i,j,k,l;  
 double \*\*A,\*B;  
 A=create(m,m);  
 B=new double[m];  
 for(i=0;i<m;i++)  
 for(j=0;j<m;j++)A[i][j]=0.0;  
 for(k=0;k<m;k++)  
 for(l=0;l<m;l++)  
 for(j=0;j<n;j++)A[k][l]+=f(k,x[j])\*f(l,x[j]);//计算法方程组系数矩阵A[k][l]  
 cout<<"法方程组的系数矩阵为:"<<endl;  
 for(i=0;i<m;i++)  
 for(j=0,k=1;j<m;j++,k++){  
 cout<<A[i][j]<<'\t';  
 if(k&&k%m==0)cout<<endl;  
 }  
    for(i=0;i<m;i++)B[i]=0.0;  
 for(i=0;i<m;i++)  
 for(j=0;j<n;j++)B[i]+=y[j]\*f(i,x[j]);  
 for(i=0;i<m;i++)cout<<"B["<<i<<"]="<<B[i]<<endl;//记录B[n]  
 xianxingfangchengzu(A,m,B,a,1e-6);  
 delete[]A;  
 delete B;  
}  
double pingfangwucha(double x[],double y[],int n,double a[],int m)//计算最小二乘解的平方误差  
{  
 double deta,q=0.0,r=0.0;  
 int i,j;  
 double \*B;  
 B=new double[m];  
 for(i=0;i<m;i++)B[i]=0.0;  
 for(i=0;i<m;i++)  
 for(j=0;j<n;j++)B[i]+=y[j]\*f(i,x[j]);  
 for(i=0;i<n;i++)q+=y[i]\*y[i];  
 for(j=0;j<m;j++)r+=a[j]\*B[j];  
 deta=fabs(q-r);  
 return deta;  
 delete B;  
}   
void main(void){  
 int i,n,m;  
 double \*x,\*y,\*a;  
 char ch='y';  
 do{  
 system("cls");  
 cout<<"请输入所给拟合数据点的个数n=";  
 cin>>n;  
 cout<<"请输入所要拟合多项式的项数m=";  
 cin>>m;  
 while(n<=m){  
    cout<<"你所输入的数据点无法确定拟合项数，请重新输入"<<endl;  
    Sleep(1000);  
    system("cls");  
    cout<<"请输入所给拟合数据点的个数n=";  
    cin>>n;  
    cout<<"请输入所要拟合多项式的项数m=";  
    cin>>m;  
    }  
 x=new double[n];                        //存放数据点x  
 y=new double[n];                             //存放数据点y  
 a=new double[m];                             //存放拟合多项式的系数  
 cout<<"请输入所给定的"<<n<<"个数据x"<<endl;

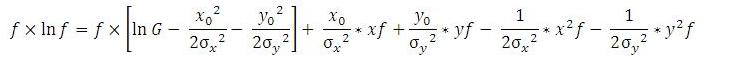
for(i=0;i<n;i++)  
 {  
 cout<<"x["<<i+1<<"]=";  
 cin>>x[i];  
 }  
 cout<<"请输入所给定的"<<n<<"个数据y"<<endl;  
 for(i=0;i<n;i++)  
 {  
 cout<<"y["<<i+1<<"]=";  
 cin>>y[i];  
 }  
 zuixiaoerchengnihe(x,y,n,a,m+1);  
    cout<<endl;  
 cout<<"拟合多项式的系数为:"<<endl;  
 for(i=0;i<=m;i++)cout<<"a["<<i<<"]="<<a[i]<<'\t';  
 cout<<endl;  
 cout<<"平方误差为："<<pingfangwucha(x,y,n,a,m+1)<<endl;  
 delete x;   delete y;  
 cout<<"按y继续，按其他字符退出"<<endl;  
 cin>>ch;  
 }while(ch=='y'

**（1）二维高斯去曲面拟合推导**

一个二维高斯方程可以写成如下形式：

http://img.my.csdn.net/uploads/201301/10/1357808005_7372.JPG

其中，G为高斯分布的幅值，,为x,y方向上的标准差，对式（1）两边取对数，并展开平方项，整理后为：



假如参与拟合的数据点有N个，则将这个N个数据点写成矩阵的形式为：**A = B C**，

其中：

A为N\*1的向量，其元素为：

http://img.my.csdn.net/uploads/201301/10/1357808203_3588.JPG

B为N\*5的矩阵：

http://img.my.csdn.net/uploads/201301/10/1357808300_2360.JPG

C为一个由高斯参数组成的向量：

http://img.my.csdn.net/uploads/201301/10/1357808354_3941.JPG

**（2）求解二维高斯曲线拟合**

N个数据点误差的列向量为：**E=A-BC**，用最小二乘法拟合，使其N个数据点的均方差最小，即：

http://img.my.csdn.net/uploads/201301/10/1357808625_4392.JPG

在图像数据处理时，数据量比较大，为减小计算量，将矩阵B进行QR分解，即：B=QR，分解后Q为一个N\*N的正交矩阵，R为一个N\*5的上三角矩阵，对**E=A-BC**进行如下推导：

http://img.my.csdn.net/uploads/201301/10/1357808851_5291.JPG

由于Q为正交矩阵，可以得到：

http://img.my.csdn.net/uploads/201301/10/1357808994_7004.JPG

令：

http://img.my.csdn.net/uploads/201301/10/1357809063_5551.JPG

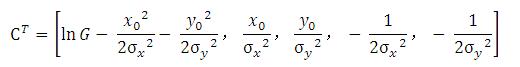
其中：  
S为一个5维列向量；T为一个N-5维列向量；R1为一个5\*5的上三角方阵，则

http://img.my.csdn.net/uploads/201301/10/1357809136_6999.JPG

上式中，当S = R1C时取得最小值，因此只需解出：

http://img.my.csdn.net/uploads/201301/10/1357809363_1661.JPG

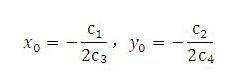
即可求出：



中的

http://img.my.csdn.net/uploads/201301/10/1357809498_9521.JPG

这些参数，这里先给出：



这里：

http://img.my.csdn.net/uploads/201301/12/1357999306_4474.JPG

**（3）C++代码实现，算法的实现过程中由于涉及大量的矩阵运算，所以采用了第三方的开源矩阵算法Eigen，这里真正用于高斯拟合的函数是**

**bool GetCentrePoint(float& x0,float& y0)**

**关于Eigen的用法请参考：http://blog.csdn.net/hjx\_1000/article/details/8490941**

**[cpp]** [view plaincopy](http://blog.csdn.net/houjixin/article/details/8490653)

1. #include "stdafx.h"
2. #include "GaussAlgorithm.h"
4. VectorXf m\_Vector\_A;
5. MatrixXf m\_matrix\_B;
6. **int** m\_iN =-1;
8. **bool** InitData(**int** pSrc[100][100], **int** iWidth, **int** iHeight)
9. {
10. **if** (NULL == pSrc || iWidth <=0 || iHeight <= 0)
11. **return** **false**;
12. m\_iN = iWidth\*iHeight;
13. VectorXf tmp\_A(m\_iN);
14. MatrixXf tmp\_B(m\_iN, 5);
15. **int** i =0, j=0, iPos =0;
16. **while**(i<iWidth)
17. {
18. j=0;
19. **while**(j<iHeight)
20. {
21. tmp\_A(iPos) = pSrc[i][j] \* log((**float**)pSrc[i][j]);
23. tmp\_B(iPos,0) = pSrc[i][j] ;
24. tmp\_B(iPos,1) = pSrc[i][j] \* i;
25. tmp\_B(iPos,2) = pSrc[i][j] \* j;
26. tmp\_B(iPos,3) = pSrc[i][j] \* i \* i;
27. tmp\_B(iPos,4) = pSrc[i][j] \* j \* j;
28. ++iPos;
29. ++j;
30. }
31. ++i;
32. }
33. m\_Vector\_A = tmp\_A;
34. m\_matrix\_B = tmp\_B;
36. }
37. **bool** GetCentrePoint(**float**& x0,**float**& y0)
38. {
39. **if** (m\_iN<=0)
40. **return** **false**;
41. //QR分解
42. HouseholderQR<MatrixXf> qr;
43. qr.compute(m\_matrix\_B);
44. MatrixXf R = qr.matrixQR().triangularView<Upper>();
45. MatrixXf Q =  qr.householderQ();
47. //块操作，获取向量或矩阵的局部
48. VectorXf S;
49. S = (Q.transpose()\* m\_Vector\_A).head(5);
50. MatrixXf R1;
51. R1 = R.block(0,0,5,5);
53. VectorXf C;
54. C = R1.inverse() \* S;
56. x0 = -0.5 \* C[1] / C[3];
57. y0 = -0.5 \* C[2] / C[4];
59. **return** **true**;
60. }

**其中：**

函数 bool InitData(int pSrc[100][100], int iWidth, int iHeight)主要用于将数组转换成相应的矩阵，因为我的所有数据都在一个整形的二维数组中存着，所以需要做这种转换。

函数bool GetCentrePoint(float& x0,float& y0)主要用于对数据点进行二维高斯曲面拟合，并返回拟合的光点中心。