【机器学习】实验一 实验报告

原创

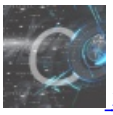
炒扁豆 于 2018-11-12 19:47:35 发布 6307 收藏 11

分类专栏： 机器学习

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/C2681595858/article/details/83997463

版权



机器学习 专栏收录该内容

6 篇文章 1 订阅

订阅专栏

文章目录

一、 参数说明

1、在开始贝叶斯判定前，先要设置上面几个参数。

2、样例文件格式说明

二、实验结果

三、设计说明

1、文件关系

2、各个文件详细设计及测试

①一些说明

②各个函数的测试

a)、matrix.h的测试

1) 构造函数及print函数测试：

2）等号重载及行列式函数测试

3）乘号重载验证

4）逆矩阵测试

5）将矩阵作为向量使用时正确性验证及矩阵转置测试

6）矩阵加法、数乘及setElemet函数验证

b)storage.h测试

c) bayes.h测试

1）马氏距离函数验证

3、正确性说明

四、遇到的困难

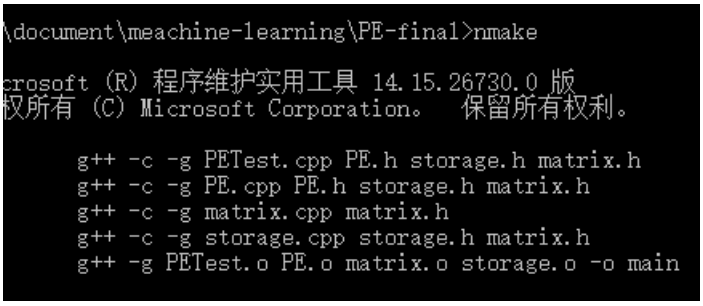
[报错：缺少合适的复制构造函数](https://blog.csdn.net/C2681595858/article/details/83956459)

[程序莫名奔溃，原来是strcpy_s出问题了](https://blog.csdn.net/C2681595858/article/details/83990068)

一、 参数说明

github最新代码下载链接

-这个上面是最新的代码，需要使用命令行g++6.3.0以上的编译器编译，若使用linux，需要将Makefile文件中的rm改为del，将main改为main.exe




```
Bayes bayes;
bayes.setSampleFile("sample.txt");
bayes.setClassSize(3);
bayes.setSampleSize(10);
bayes.setVectorLen(3);
```

1、在开始贝叶斯判定前，先要设置上面几个参数。

第一个是样本数据所在文件名。

第二个是设置有多少个类别

第三个是每个类别有多少个样本数据

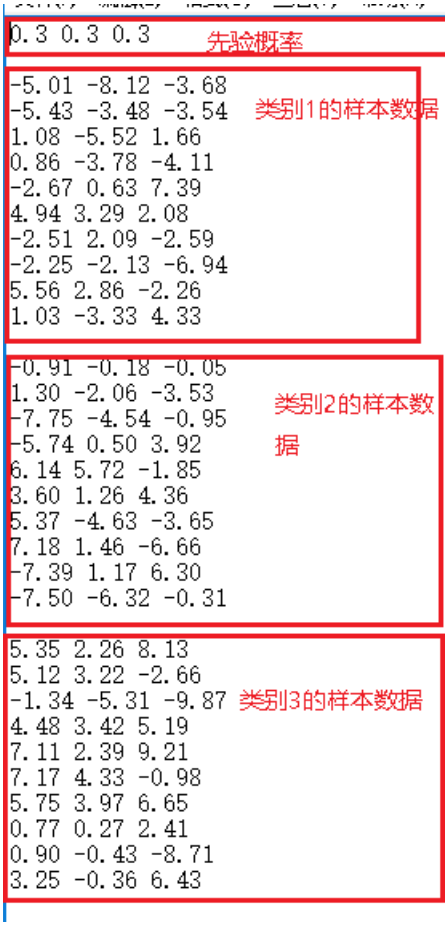
第三个是每个样本数据的维度

设置完这些，还有一项没设置就是先验概率。

先验概率在“sample.txt”文件中进行设置，具体sample.txt文件格式要求如下：

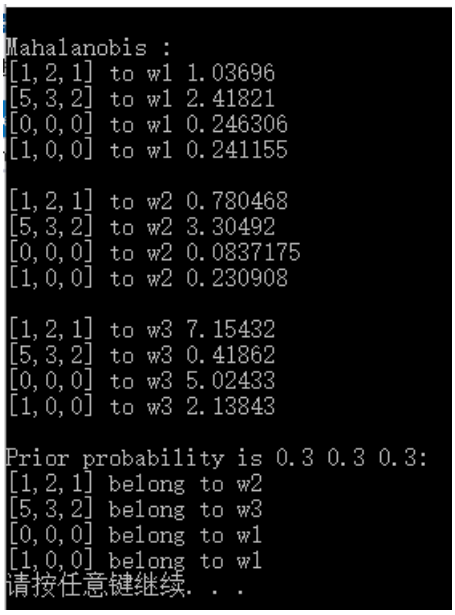
2、样例文件格式说明

文件开头是各个类的先验概率，有多少个类就应该有多少个先验概率，就是说各个类的先验概率相同也都要写进去，而不能省略。
然后后续是各个类的样本数据，数据需要按类别的顺序依次排放。具体如下：



二、实验结果

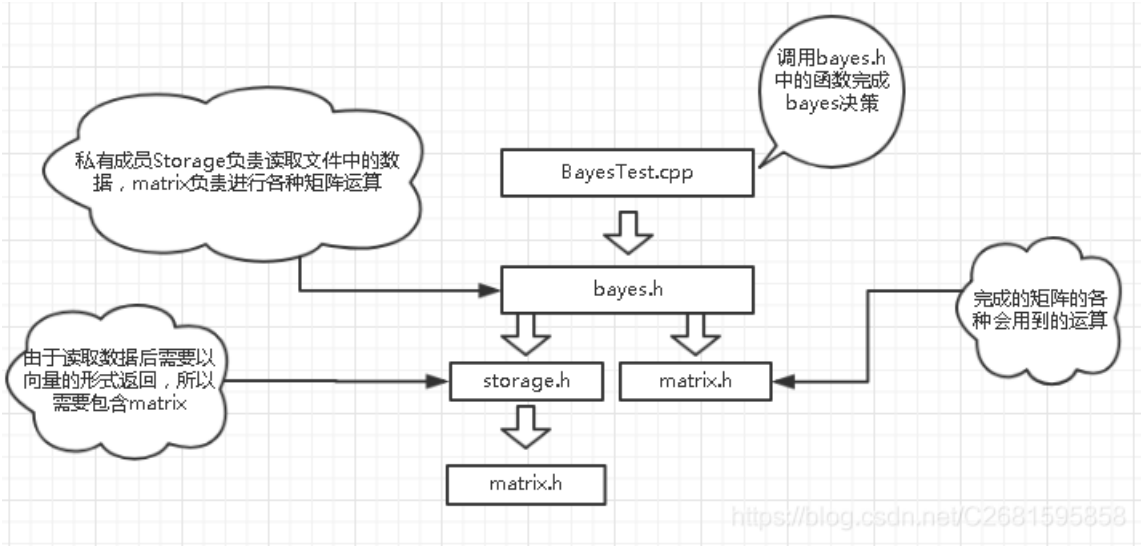
- 这是在win10 vs2017 c++环境下的实验结果



上面显示的是一个向量到某一个类别的马氏距离。下面显示各个向量应该是哪个类。

三、设计说明

1、文件关系



2、各个文件详细设计及测试

①一些说明

- 文件详细设计在各个.h文件中有详细注释，目的导向作业，所以matrix只实现了作业中会用到的算法。
  - matrix.h中求行列式的算法det（）没有使用了递归, 为了避免在类之外使用其递归部分，将其放在了private中。
  - inverse()求逆矩阵，没有使用高斯方法，直接使用了伴随矩阵，效率较低，但是实现简单。

②各个函数的测试

a)、matrix.h的测试

测试文件matrixTest.cpp

1) 构造函数及print函数测试：

代码：

```
float array[] = { 1.02, 2.03, 3.04,  4.05,  5.06, 6.07, 7.08, 8.09,  9.10, 5.06, 6.07, 8.09,  3.04, 6.07, 9.10, 8.09 };
 Matrix matrix(4, 4, array);
 matrix.printMatrix();
```

结果：

```
1.02  2.03  3.04  4.05
5.06  6.07  7.08  8.09
9.1  5.06  6.07  8.09
3.04  6.07  9.1  8.09
```

2）等号重载及行列式函数测试

代码：

```
float array[] = { 1.02, 2.03, 3.04,  4.05,  5.06, 6.07, 7.08, 8.09,  9.10, 5.06, 6.07, 8.09,  3.04, 6.07, 9.10, 8.09 };
 Matrix matrix(4, 4, array);

 Matrix matrix1;
 matrix1 = matrix;
 cout << matrix1.det() << endl;
```

结果：

```
83.2484
请按任意键继续. . .
```

结果正确性验证（使用了matlab)

```
b =

    1.0200    2.0300    3.0400    4.0500
    5.0600    6.0700    7.0800    8.0900
    9.1000    5.0600    6.0700    8.0900
    3.0400    6.0700    9.1000    8.0900

>> det(b)

ans =

   83.2483
```

https://blog.csdn.net/C2681595858

3）乘号重载验证

代码：

```
float array[] = { 1.02, 2.03, 3.04,  4.05,  5.06, 6.07, 7.08, 8.09,  9.10, 5.06, 6.07, 8.09,  3.04, 6.07, 9.10, 8.09 };
 Matrix matrix(4, 4, array);

 Matrix matrix1;
 matrix1 = matrix;

 Matrix matrix2(2, 6);
 matrix2 = (matrix * matrix1);

 matrix2.printMatrix();
```

结果：

```
51.2882  54.3586  72.781  77.9118
124.897  132.048  174.953  192.325
114.716  129.008  173.953  192.345
141.219  138.168  181.073  200.485

请按任意键继续. . .
```

验证结果：

```
>> b*b

ans =

   51.2882   54.3586   72.7810   77.9118
  124.8970  132.0478  174.9526  192.3246
  114.7162  129.0077  173.9527  192.3448
  141.2186  138.1684  181.0732  200.4854

>>
```

https://blog.csdn.net/C2681595858

4）逆矩阵测试

代码：

```
float array[] = { 1.02, 2.03, 3.04,  4.05,  5.06, 6.07, 7.08, 8.09,  9.10, 5.06, 6.07, 8.09,  3.04, 6.07, 9.10, 8.09 };
 Matrix matrix(4, 4, array);
 (matrix.inverse()).printMatrix();
```

结果：

```
-0.198265  -0.14827  0.19802  0.049505
-0.594795  1.04034  -0.39604  -0.346535
-0.198265  -0.643319  0.19802  0.544554
0.7438  -0.00122547  2.39478e-08  -0.247525

请按任意键继续. . .
```

验证结果：

```
>> inv(b)

ans =

   -0.1983   -0.1483    0.1980    0.0495
   -0.5948    1.0403   -0.3960   -0.3465
   -0.1983   -0.6433    0.1980    0.5446
    0.7438   -0.0012   -0.0000   -0.2475

>>
```

5）将矩阵作为向量使用时正确性验证及矩阵转置测试

代码：

```
float vect[2] = { 2,2.34 };
 Matrix matrix5(2, 1, vect);
 (matrix5*(matrix5.trans())).printMatrix();
```

结果：

```
4  4.68
4.68  5.4756

请按任意键继续. . .
```

结果验证：

```
c =

    2.0000
    2.3400

>> c*c'

ans =

    4.0000    4.6800
    4.6800    5.4756
```

6）矩阵加法、数乘及setElemet函数验证

代码：

```
Matrix matrix2(2, 6);
 matrix2.setElement(1, 2, 6);
 matrix2.setElement(2, 3, 6);
 matrix2.setElement(1, 1, 6);
 matrix2.setElement(2, 6, 6);
 matrix2.printMatrix();

 (matrix2 + matrix2).printMatrix();
 (3.99 * matrix2).printMatrix();
```

结果：

```
6  6  0  0  0  0
0  0  6  0  0  6


12  12  0  0  0  0
0  0  12  0  0  12


23.94  23.94  0  0  0  0
0  0  23.94  0  0  23.94

请按任意键继续. . .
```

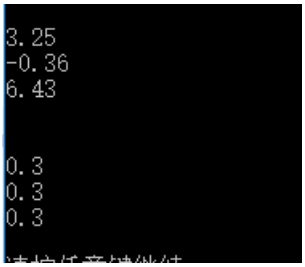
b)storage.h测试

```
Storage storage;
 storage.setfile("sample.txt");

 storage.setVectorLen(3);
 //先读开头的先验概率
 storage.readPostPro();
 storage.readPostPro();
 storage.readPostPro();
 //读30次到文件末尾
 for(int counter = 0; counter < 30; counter++)
  (storage.readData()).printMatrix();
 //返回开头
 storage.reset();

 //这里应该是先验概率
 (storage.readData()).printMatrix();
```

结果：最后都到了最后一个向量然后返回头部，又都到了先验概率，说明所有函数正确



c) bayes.h测试

这里主要用到了均值向量求解函数和协方差阵求解函数，只要需验证这两个正确，而马氏距离的计算用到了上述两个结果，所以只要马氏距离正确，则函数正确：

特殊说明：由于在写下面这部分的时候我的样本文件是有问题的，所以mahal距离可能和大家的不同，但是我在matlab输入的样本文件和我程序中用的样本文件是一样的（都是有错误的），所以最终还是能够验证我的算法是正确的

1）马氏距离函数验证

代码：

```
Bayes bayes;
 bayes.setSampleFile("sample.txt");
 bayes.setClassSize(3);
 bayes.setSampleSize(10);
 bayes.setVectorLen(3);

 cout << "\nMahalanobis :\n";
 cout << "[1,2,1] to w1 " << bayes.mahal("[1,2,1]", 1) << " " << endl;
```

结果：

```
Mahalanobis :
[1,2,1] to w1 1.03017
请按任意键继续. . .
```

结果验证：

```
a =

   -5.0100   -8.1200   -3.6800
   -5.4300   -3.4800   -3.5400
    1.0800   -5.5200    1.6600
    0.8600   -3.7800   -4.1100
   -2.6700    0.6300    7.3900
    4.9400    3.2900    2.0800
   -2.5100    2.0900   -2.5900
   -2.2500   -2.1300   -6.9400
    5.5600    2.8600   -2.2600
    1.0300   -3.3300    4.3300

>> d = [1,2,1]

d =

     1     2     1

>> mahal(d,a)

ans =

    1.0302

>>
```

https://blog.csdn.net/C2681595858

3、正确性说明

总上，从底层到顶层各个函数都和matlab中相关函数运算结果一致，所以最终结果可信度较高。

四、遇到的困难

- 本来早早写完了，然后写完发现判定结果和大家的判定结果不一致，但是就像上面这样，每一步都是通过matlab验证了的，难道我对概念理解有误，查了很多资料。发现实现方法上确实有些不同，但是本质是没有变的。然后就照搬别人思路，实现代码，这样一改，不仅没有改好，反而越改越乱了。然后我确信程序没有问题以后，就放弃了修改，就这样吧。但是不甘心啊，每一步都对，最后一步出错，真的难受，第二天中午，一觉醒来看了下QQ群，大家再说自己样本文件出了差错，导致出错，我也看看是不是我也是这样，看了一遍，没看出啥东西来。但是我还是有点不太确定它有没有错误，毕竟其他都可以验证是正确的，唯独它没有验证。所以我重新从课本上抄了一遍文件。我天。。正确了，真是太激动了哈哈哈哈。这个故事告诉我们，在可控因素确信无疑时，一定要仔细的研究不可控因素。说不定猫腻就在那里呢。
- 我另写了两篇博客说明了遇到的其他困难和解决办法：

报错：缺少合适的复制构造函数

程序莫名奔溃，原来是strcpy_s出问题了