信号统计建模实验报告

Signal statistical Modeling Experiment Report

张博栋 PB10210189

Bodong Zhang PB10210189

Build A 2-Class Classifier With Gaussian Models

You are asked to solve a 2-class (class A & B) classification problem based on multivariate Gaussian models. Assume two classes have equal prior probabilities. Each observation feature is a 3-dimension vector. Assume you can collect a set of training data for each class. Based on the training data (provided in the course Web), you consider the following different ways to build such a classifier. Then the estimated models are used to classify some new test data (also provided in the course Web).

判别时如果采用后验概率（认为 A，B 概率不等）精确度可能更高，但是要求中认为先验概率相同，所以实验中采用似然估计。

实验编程语言 C++，实验代码见后，共 2228 行。

Project 1

1. First of all, let’s consider to build a very simple classifier based on single multivariate Gaussian model. Each class is modelled by a single 3-D multivariate Gaussian distribution. For simplicity, we assume each multivariate Gaussian has a diagonal covariance matrix. Show how to estimate Gaussian mean vector and covariance matrix for each class based on the Maximum likelihood (ML) estimation. Report the classification accuracy of the ML-trained models as measured in the test data set.

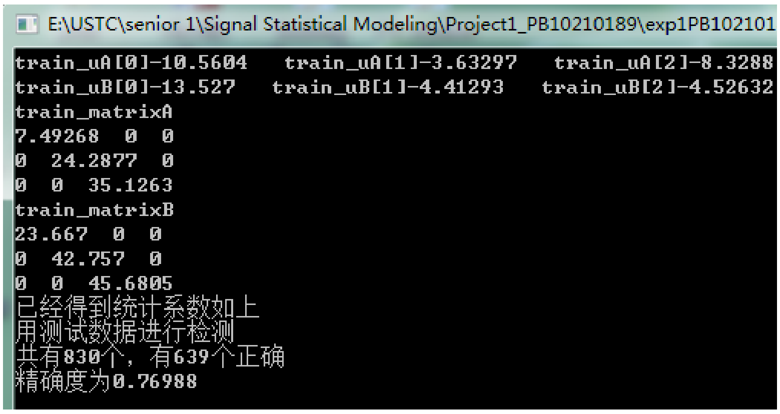
由最大似然估计 $\mu_{\mathrm{ML}} = \frac{\sum_{k=1}^{n} x_k}{n}$ ， $\Sigma_{\mathrm{ML}} = \frac{1}{n}\sum_{k=1}^{n}(X_k - \mu)(X_k - \mu)^{\mathrm{T}}$

编程实现，计算训练数据，得到均值和协方差矩阵，再把由训练数据得到的参数代入公式，

$$p(X \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left[-\frac{(X-\mu)^{\mathrm{T}} \Sigma^{-1} (X-\mu)}{2} \right]$$

由最大似然估计公式，将测试数据带入公式，比较概率大小，若 PA 概率大则判为 PA，若 PB 概率大则判为 PB。

得到结果如下（忽略协方差矩阵非对角线上的值）



E:\USTC\senior 1\Signal Statistical Modeling\Project1_PB10210189\exp1PB102101

```
train_uA[0]-10.5604   train_uA[1]-3.63297   train_uA[2]-8.3288
train_uB[0]-13.527   train_uB[1]-4.41293   train_uB[2]-4.52632
train_matrixA
7.49268  0  0
0  24.2877  0
0  0  35.1263
train_matrixB
23.667  0  0
0  42.757  0
0  0  45.6805
已经得到统计系数如上
用测试数据进行检测
共有830个，有639个正确
精确度为0.76988
```

根据训练数据，样本 A 均值（-10.5604，-3.63297，-8.3288）

协方差矩阵 7.49268　　0　　0
　　　　　　0　　24.2877　0
　　　　　　0　　　0　　35.1263

样本 B 均值（-13.527，-4.41293，-4.52632）

协方差矩阵 23.667　　0　　0
　　　　　　0　　42.757　　0
　　　　　　0　　　　0　45.6805

830 个测试数据进行分类，再根据实际类别比较，发现正确 639 个，精度为 0.76988

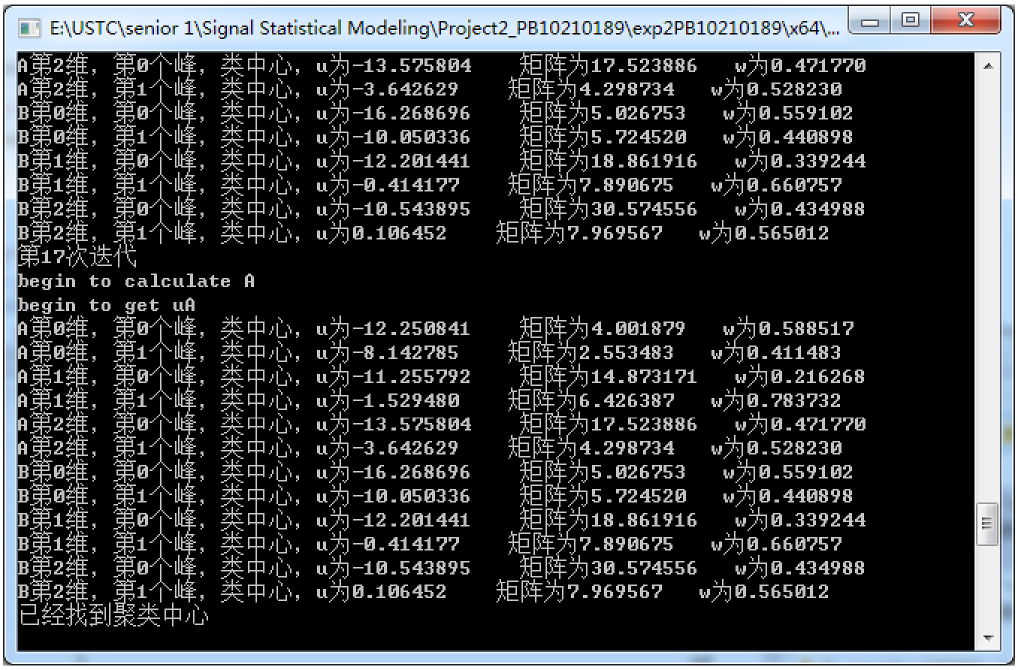
第二道题

Project2

Consider to improve the above simple Gaussian classifier by using a more complicated models, namely Gaussian mixture models (GMM), to model each class. Again, you can assume each Gaussian has a diagonal covariance matrix. Use the k-means method to initialize the GMM's. Then improve the GMM models iteratively based on the EM algorithm. Investigate and report results for GMM's which have 2, 4, 8 mixture components respectively.

一、先讨论 GMM 为 2 个峰的情况

首先用 K-means 算法得到初始值，进行 K-means 算法时不停地迭代，直到数据收敛为止，每次不断根据数据的分类寻找新的参数（w，u，协方差矩阵），直到迭代前后参数完全不变为止。


E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
A第2维, 第0个峰, 类中心, u为-13.575804    矩阵为17.523886    w为0.471770
A第2维, 第1个峰, 类中心, u为-3.642629    矩阵为4.298734    w为0.528230
B第0维, 第0个峰, 类中心, u为-16.268696    矩阵为5.026753    w为0.559102
B第0维, 第1个峰, 类中心, u为-10.050336    矩阵为5.724520    w为0.440898
B第1维, 第0个峰, 类中心, u为-12.201441    矩阵为18.861916    w为0.339244
B第1维, 第1个峰, 类中心, u为-0.414177    矩阵为7.890675    w为0.660757
B第2维, 第0个峰, 类中心, u为-10.543895    矩阵为30.574556    w为0.434988
B第2维, 第1个峰, 类中心, u为0.106452    矩阵为7.969567    w为0.565012
第17次迭代
begin to calculate A
begin to get uA
A第0维, 第0个峰, 类中心, u为-12.250841    矩阵为4.001879    w为0.588517
A第0维, 第1个峰, 类中心, u为-8.142785    矩阵为2.553483    w为0.411483
A第1维, 第0个峰, 类中心, u为-11.255792    矩阵为14.873171    w为0.216268
A第1维, 第1个峰, 类中心, u为-1.529480    矩阵为6.426387    w为0.783732
A第2维, 第0个峰, 类中心, u为-13.575804    矩阵为17.523886    w为0.471770
A第2维, 第1个峰, 类中心, u为-3.642629    矩阵为4.298734    w为0.528230
B第0维, 第0个峰, 类中心, u为-16.268696    矩阵为5.026753    w为0.559102
B第0维, 第1个峰, 类中心, u为-10.050336    矩阵为5.724520    w为0.440898
B第1维, 第0个峰, 类中心, u为-12.201441    矩阵为18.861916    w为0.339244
B第1维, 第1个峰, 类中心, u为-0.414177    矩阵为7.890675    w为0.660757
B第2维, 第0个峰, 类中心, u为-10.543895    矩阵为30.574556    w为0.434988
B第2维, 第1个峰, 类中心, u为0.106452    矩阵为7.969567    w为0.565012
已经找到聚类中心
```

经过第 17 次迭代后，找到聚类中心。

之后再进行 EM 算法，利用 K-means 算法得到的值作为 EM 算法初始值，不断进行迭代，迭代公式

$$\frac{\partial Q}{\partial \mu_k}=0 \Rightarrow \mu_k^{(i+1)}=\frac{\sum_{t=1}^{T} X_t \cdot p(l_t=k \mid X_t,\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)})}{\sum_{t=1}^{T} p(l_t=k \mid X_t,\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)})}$$

$$\frac{\partial Q}{\partial \Sigma_k}=0 \Rightarrow \Sigma_k^{(i+1)}=\frac{\sum_{t=1}^{T}(X_t-\mu_k^{(i)})^{\mathrm{T}}\cdot(X_t-\mu_k^{(i)})\cdot p(l_t=k \mid X_t,\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)})}{\sum_{t=1}^{T} p(l_t=k \mid X_t,\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)})}$$

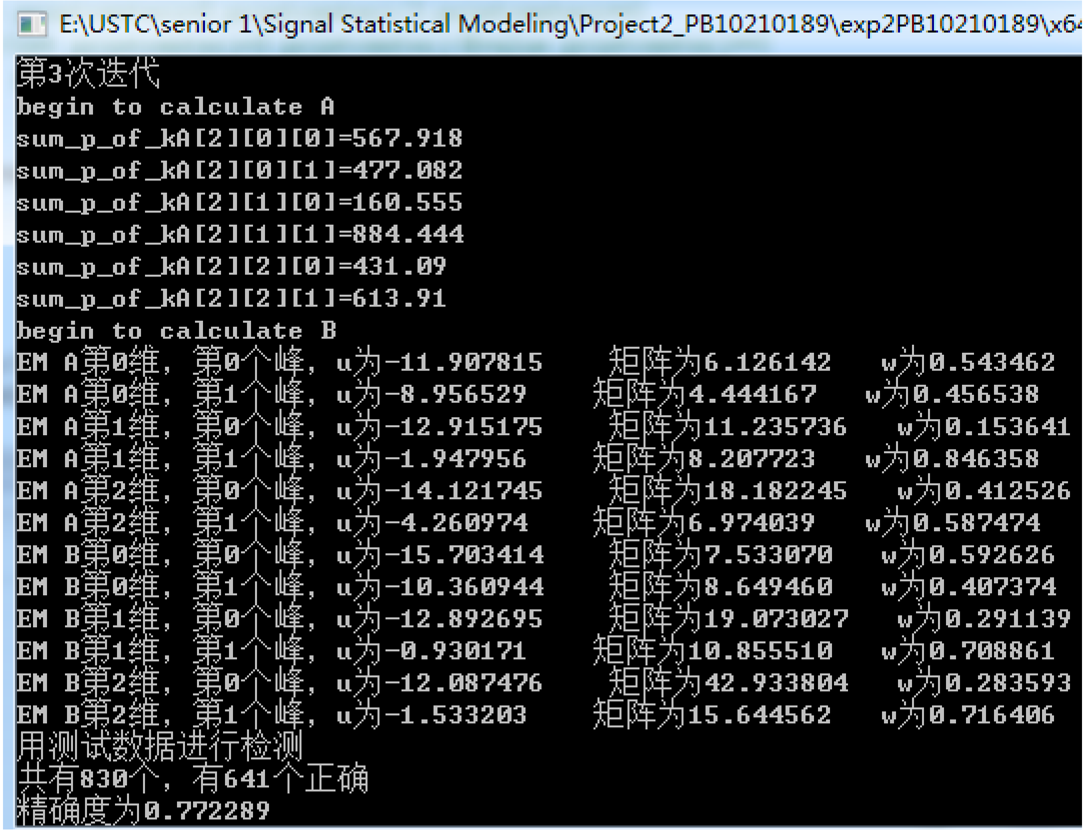
$$\frac{\partial}{\partial \omega_k}[Q-\lambda(\sum_{k=1}^{K}\omega_k-1)]=0 \Rightarrow \omega_k=\frac{\sum_{t=1}^{T} p(l_t=k \mid X_t,\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)})}{\sum_{k=1}^{K}\sum_{t=1}^{T} p(l_t=k \mid X_t,\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)})}=\frac{\sum_{t=1}^{T} p(l_t=k \mid X_t,\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)})}{T}$$

$$p(l_t=k \mid X_t,\{\omega_k^{(i)},\mu_k^{(i)},\Sigma_k^{(i)}\})=\frac{\omega_k^{(i)}\cdot N(X_t \mid \mu_k^{(i)},\Sigma_k^{(i)})}{\sum_{k=1}^{K}\omega_k^{(i)}\cdot N(X_t \mid \mu_k^{(i)},\Sigma_k^{(i)})}$$

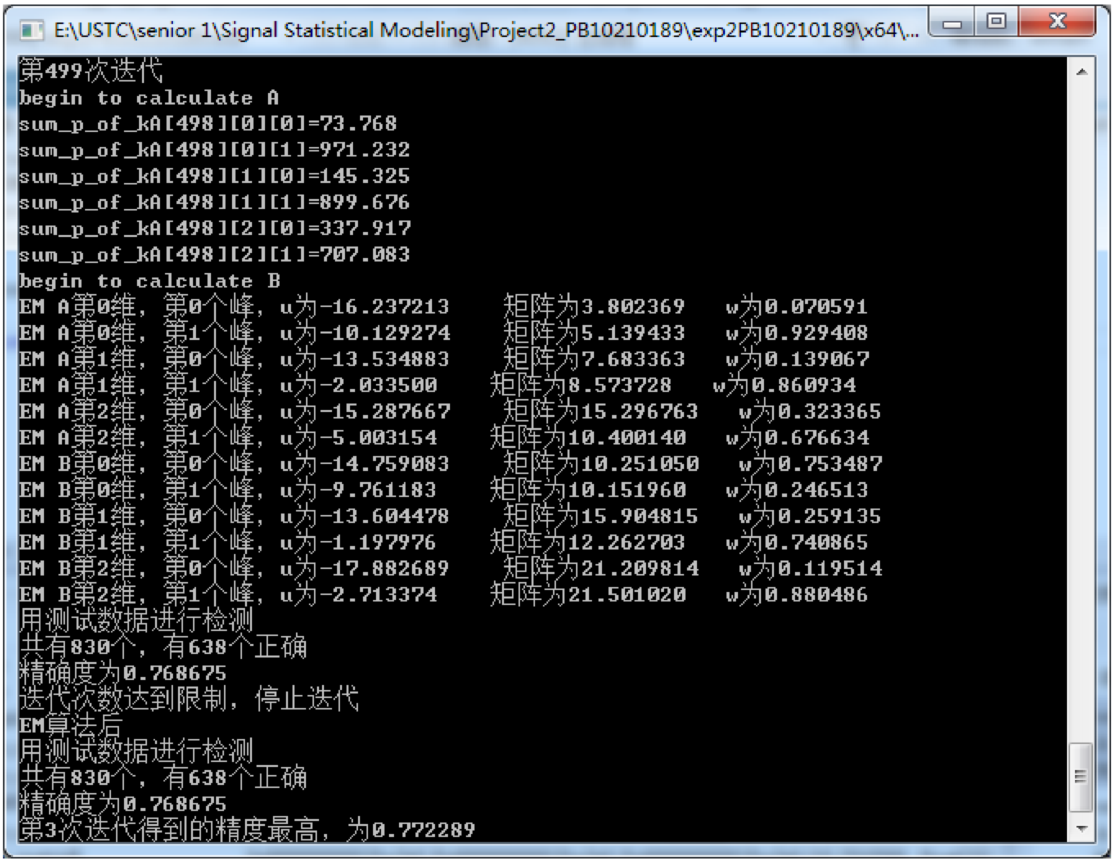
其中

由于上式在求解 w，u，方差时都要用到，所以在程序中先算出，在实验中每次迭代后都用测试数据进行测试，在前 500 次迭代中第 3 次迭代精度最高，为 0.772289,第 499 次迭代后精度为 0.768675，迭代精度略有下降。

迭代结果如下


E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64

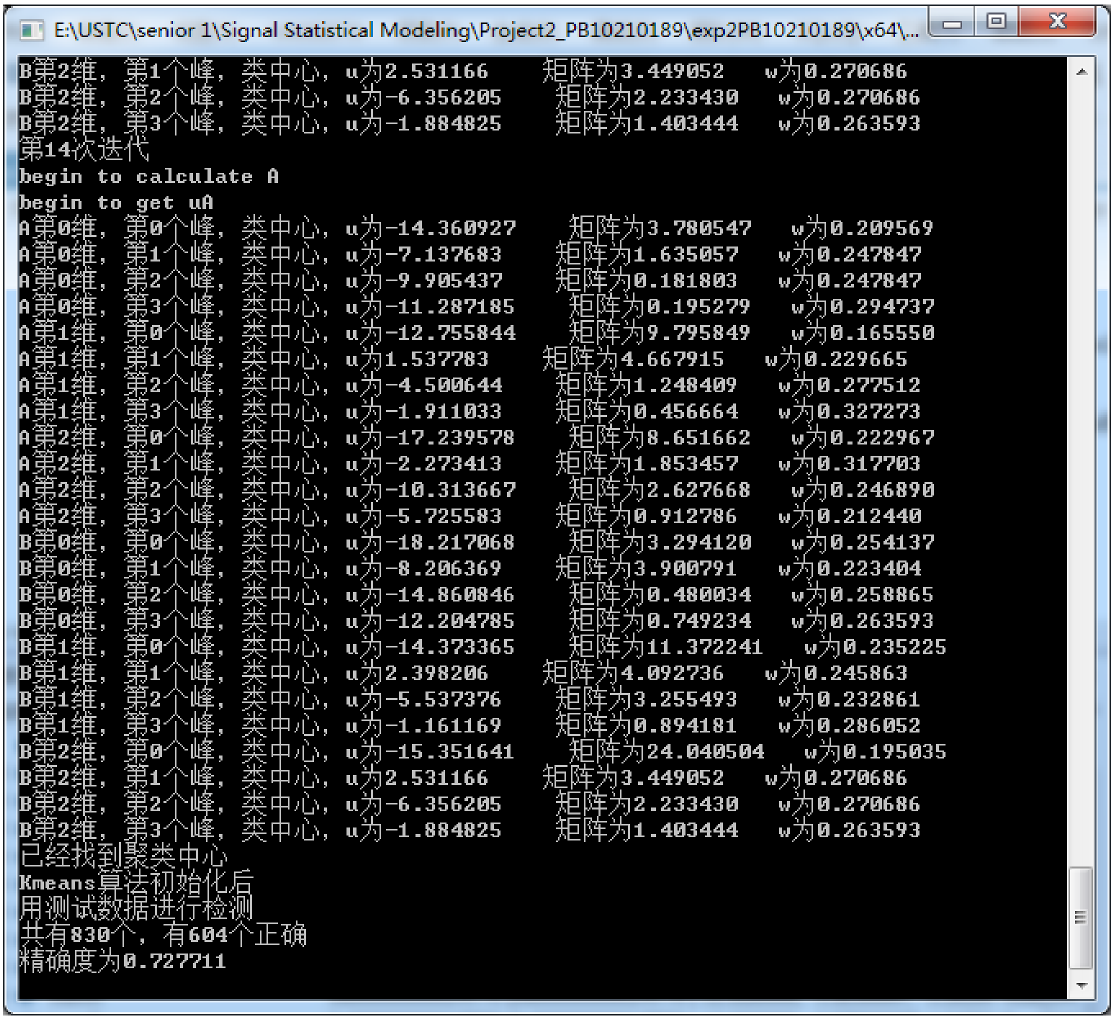
```
第3次迭代
begin to calculate A
sum_p_of_kA[2][0][0]=567.918
sum_p_of_kA[2][0][1]=477.082
sum_p_of_kA[2][1][0]=160.555
sum_p_of_kA[2][1][1]=884.444
sum_p_of_kA[2][2][0]=431.09
sum_p_of_kA[2][2][1]=613.91
begin to calculate B
EM A第0维, 第0个峰, u为-11.907815     矩阵为6.126142    w为0.543462
EM A第0维, 第1个峰, u为-8.956529     矩阵为4.444167    w为0.456538
EM A第1维, 第0个峰, u为-12.915175     矩阵为11.235736    w为0.153641
EM A第1维, 第1个峰, u为-1.947956     矩阵为8.207723    w为0.846358
EM A第2维, 第0个峰, u为-14.121745     矩阵为18.182245    w为0.412526
EM A第2维, 第1个峰, u为-4.260974     矩阵为6.974039    w为0.587474
EM B第0维, 第0个峰, u为-15.703414     矩阵为7.533070    w为0.592626
EM B第0维, 第1个峰, u为-10.360944     矩阵为8.649460    w为0.407374
EM B第1维, 第0个峰, u为-12.892695     矩阵为19.073027    w为0.291139
EM B第1维, 第1个峰, u为-0.930171     矩阵为10.855510    w为0.708861
EM B第2维, 第0个峰, u为-12.087476     矩阵为42.933804    w为0.283593
EM B第2维, 第1个峰, u为-1.533203     矩阵为15.644562    w为0.716406
用测试数据进行检测
共有830个, 有641个正确
精确度为0.772289
```


E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
第499次迭代
begin to calculate A
sum_p_of_kA[498][0][0]=73.768
sum_p_of_kA[498][0][1]=971.232
sum_p_of_kA[498][1][0]=145.325
sum_p_of_kA[498][1][1]=899.676
sum_p_of_kA[498][2][0]=337.917
sum_p_of_kA[498][2][1]=707.083
begin to calculate B
EM A第0维, 第0个峰, u为-16.237213     矩阵为3.802369    w为0.070591
EM A第0维, 第1个峰, u为-10.129274     矩阵为5.139433    w为0.929408
EM A第1维, 第0个峰, u为-13.534883     矩阵为7.683363    w为0.139067
EM A第1维, 第1个峰, u为-2.033500    矩阵为8.573728    w为0.860934
EM A第2维, 第0个峰, u为-15.287667     矩阵为15.296763    w为0.323365
EM A第2维, 第1个峰, u为-5.003154    矩阵为10.400140    w为0.676634
EM B第0维, 第0个峰, u为-14.759083     矩阵为10.251050    w为0.753487
EM B第0维, 第1个峰, u为-9.761183    矩阵为10.151960    w为0.246513
EM B第1维, 第0个峰, u为-13.604478     矩阵为15.904815    w为0.259135
EM B第1维, 第1个峰, u为-1.197976    矩阵为12.262703    w为0.740865
EM B第2维, 第0个峰, u为-17.882689     矩阵为21.209814    w为0.119514
EM B第2维, 第1个峰, u为-2.713374    矩阵为21.501020    w为0.880486
用测试数据进行检测
共有830个, 有638个正确
精确度为0.768675
迭代次数达到限制, 停止迭代
EM算法后
用测试数据进行检测
共有830个, 有638个正确
精确度为0.768675
第3次迭代得到的精度最高, 为0.772289
```

4 个峰时的情况

K-means 算法初始化



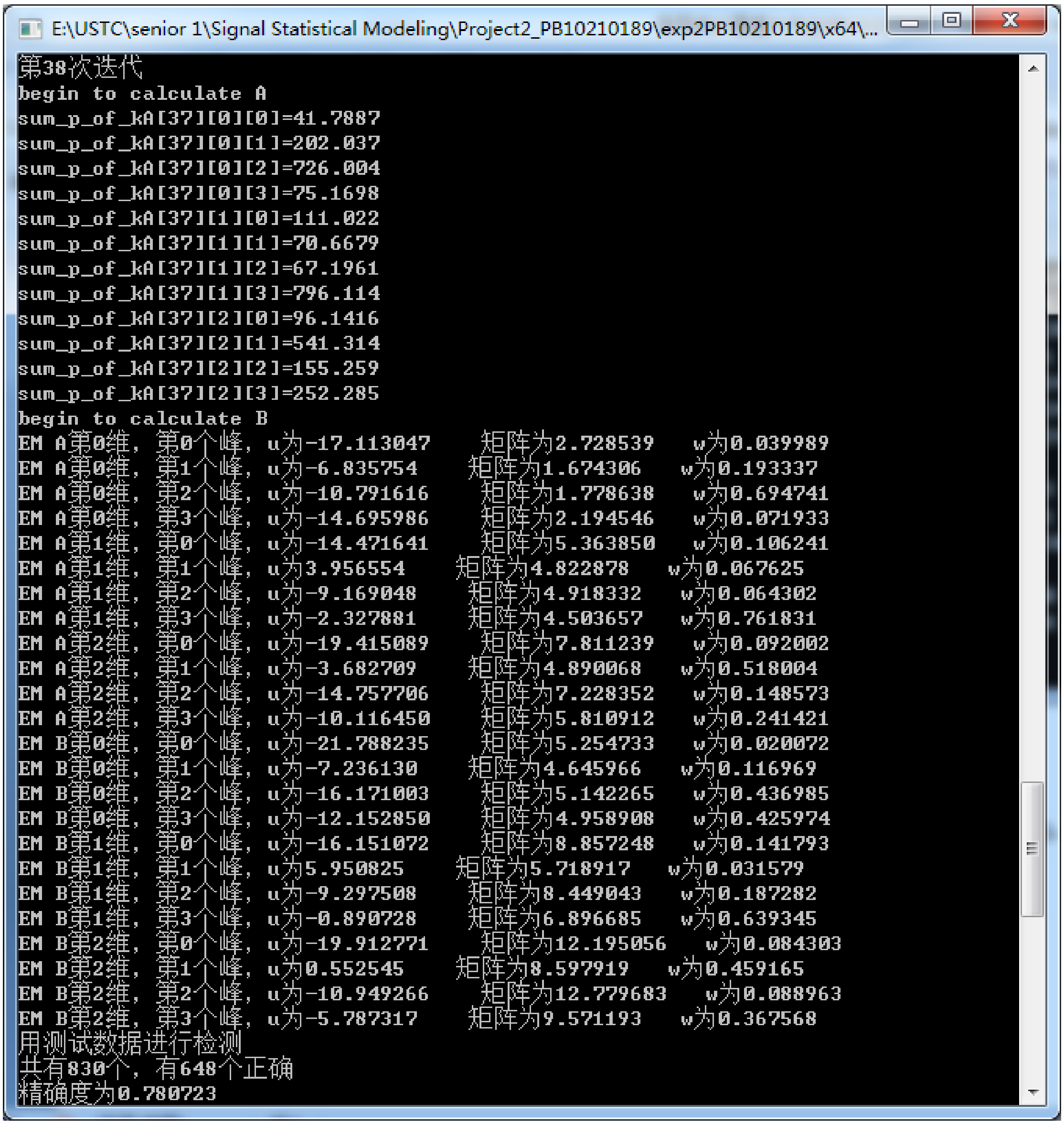
E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
B第2维, 第1个峰, 类中心, u为2.531166    矩阵为3.449052   w为0.270686
B第2维, 第2个峰, 类中心, u为-6.356205    矩阵为2.233430   w为0.270686
B第2维, 第3个峰, 类中心, u为-1.884825    矩阵为1.403444   w为0.263593
第14次迭代
begin to calculate A
begin to get uA
A第0维, 第0个峰, 类中心, u为-14.360927    矩阵为3.780547   w为0.209569
A第0维, 第1个峰, 类中心, u为-7.137683    矩阵为1.635057   w为0.247847
A第0维, 第2个峰, 类中心, u为-9.905437    矩阵为0.181803   w为0.247847
A第0维, 第3个峰, 类中心, u为-11.287185    矩阵为0.195279   w为0.294737
A第1维, 第0个峰, 类中心, u为-12.755844    矩阵为9.795849   w为0.165550
A第1维, 第1个峰, 类中心, u为1.537783    矩阵为4.667915   w为0.229665
A第1维, 第2个峰, 类中心, u为-4.500644    矩阵为1.248409   w为0.277512
A第1维, 第3个峰, 类中心, u为-1.911033    矩阵为0.456664   w为0.327273
A第2维, 第0个峰, 类中心, u为-17.239578    矩阵为8.651662   w为0.222967
A第2维, 第1个峰, 类中心, u为-2.273413    矩阵为1.853457   w为0.317703
A第2维, 第2个峰, 类中心, u为-10.313667    矩阵为2.627668   w为0.246890
A第2维, 第3个峰, 类中心, u为-5.725583    矩阵为0.912786   w为0.212440
B第0维, 第0个峰, 类中心, u为-18.217068    矩阵为3.294120   w为0.254137
B第0维, 第1个峰, 类中心, u为-8.206369    矩阵为3.900791   w为0.223404
B第0维, 第2个峰, 类中心, u为-14.860846    矩阵为0.480034   w为0.258865
B第0维, 第3个峰, 类中心, u为-12.204785    矩阵为0.749234   w为0.263593
B第1维, 第0个峰, 类中心, u为-14.373365    矩阵为11.372241   w为0.235225
B第1维, 第1个峰, 类中心, u为2.398206    矩阵为4.092736   w为0.245863
B第1维, 第2个峰, 类中心, u为-5.537376    矩阵为3.255493   w为0.232861
B第1维, 第3个峰, 类中心, u为-1.161169    矩阵为0.894181   w为0.286052
B第2维, 第0个峰, 类中心, u为-15.351641    矩阵为24.040504   w为0.195035
B第2维, 第1个峰, 类中心, u为2.531166    矩阵为3.449052   w为0.270686
B第2维, 第2个峰, 类中心, u为-6.356205    矩阵为2.233430   w为0.270686
B第2维, 第3个峰, 类中心, u为-1.884825    矩阵为1.403444   w为0.263593
已经找到聚类中心
Kmeans算法初始化后
用测试数据进行检测
共有830个, 有604个正确
精确度为0.727711
```

K-means 算法找到聚类中心，参数如上，得到的精度为 0.727711

再用上面的参数进行 EM 算法。

用测试数据进行检测，第 38 次迭代后精度最高，为 0.780723，如下

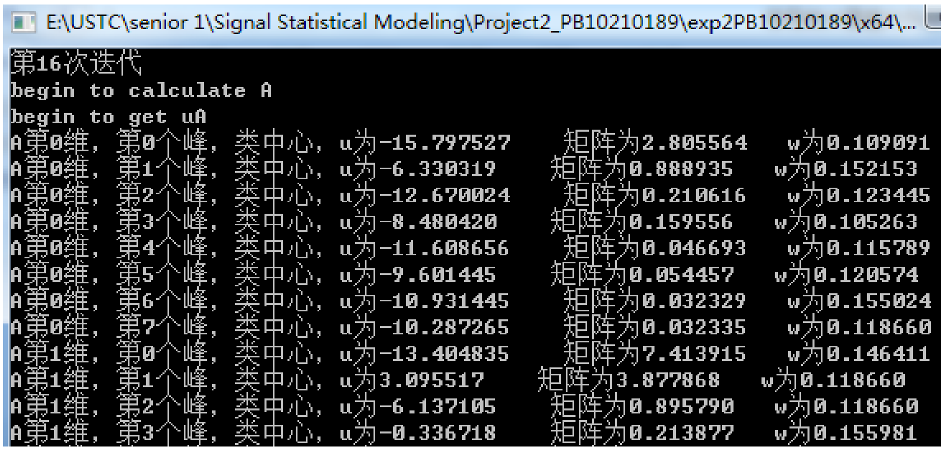


E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
第38次迭代
begin to calculate A
sum_p_of_kA[37][0][0]=41.7887
sum_p_of_kA[37][0][1]=202.037
sum_p_of_kA[37][0][2]=726.004
sum_p_of_kA[37][0][3]=75.1698
sum_p_of_kA[37][1][0]=111.022
sum_p_of_kA[37][1][1]=70.6679
sum_p_of_kA[37][1][2]=67.1961
sum_p_of_kA[37][1][3]=796.114
sum_p_of_kA[37][2][0]=96.1416
sum_p_of_kA[37][2][1]=541.314
sum_p_of_kA[37][2][2]=155.259
sum_p_of_kA[37][2][3]=252.285
begin to calculate B
EM A第0维, 第0个峰, u为-17.113047    矩阵为2.728539    w为0.039989
EM A第0维, 第1个峰, u为-6.835754    矩阵为1.674306    w为0.193337
EM A第0维, 第2个峰, u为-10.791616    矩阵为1.778638    w为0.694741
EM A第0维, 第3个峰, u为-14.695986    矩阵为2.194546    w为0.071933
EM A第1维, 第0个峰, u为-14.471641    矩阵为5.363850    w为0.106241
EM A第1维, 第1个峰, u为3.956554    矩阵为4.822878    w为0.067625
EM A第1维, 第2个峰, u为-9.169048    矩阵为4.918332    w为0.064302
EM A第1维, 第3个峰, u为-2.327881    矩阵为4.503657    w为0.761831
EM A第2维, 第0个峰, u为-19.415089    矩阵为7.811239    w为0.092002
EM A第2维, 第1个峰, u为-3.682709    矩阵为4.890068    w为0.518004
EM A第2维, 第2个峰, u为-14.757706    矩阵为7.228352    w为0.148573
EM A第2维, 第3个峰, u为-10.116450    矩阵为5.810912    w为0.241421
EM B第0维, 第0个峰, u为-21.788235    矩阵为5.254733    w为0.020072
EM B第0维, 第1个峰, u为-7.236130    矩阵为4.645966    w为0.116969
EM B第0维, 第2个峰, u为-16.171003    矩阵为5.142265    w为0.436985
EM B第0维, 第3个峰, u为-12.152850    矩阵为4.958908    w为0.425974
EM B第1维, 第0个峰, u为-16.151072    矩阵为8.857248    w为0.141793
EM B第1维, 第1个峰, u为5.950825    矩阵为5.718917    w为0.031579
EM B第1维, 第2个峰, u为-9.297508    矩阵为8.449043    w为0.187282
EM B第1维, 第3个峰, u为-0.890728    矩阵为6.896685    w为0.639345
EM B第2维, 第0个峰, u为-19.912771    矩阵为12.195056    w为0.084303
EM B第2维, 第1个峰, u为0.552545    矩阵为8.597919    w为0.459165
EM B第2维, 第2个峰, u为-10.949266    矩阵为12.779683    w为0.088963
EM B第2维, 第3个峰, u为-5.787317    矩阵为9.571193    w为0.367568
用测试数据进行检测
共有830个, 有648个正确
精确度为0.780723
```

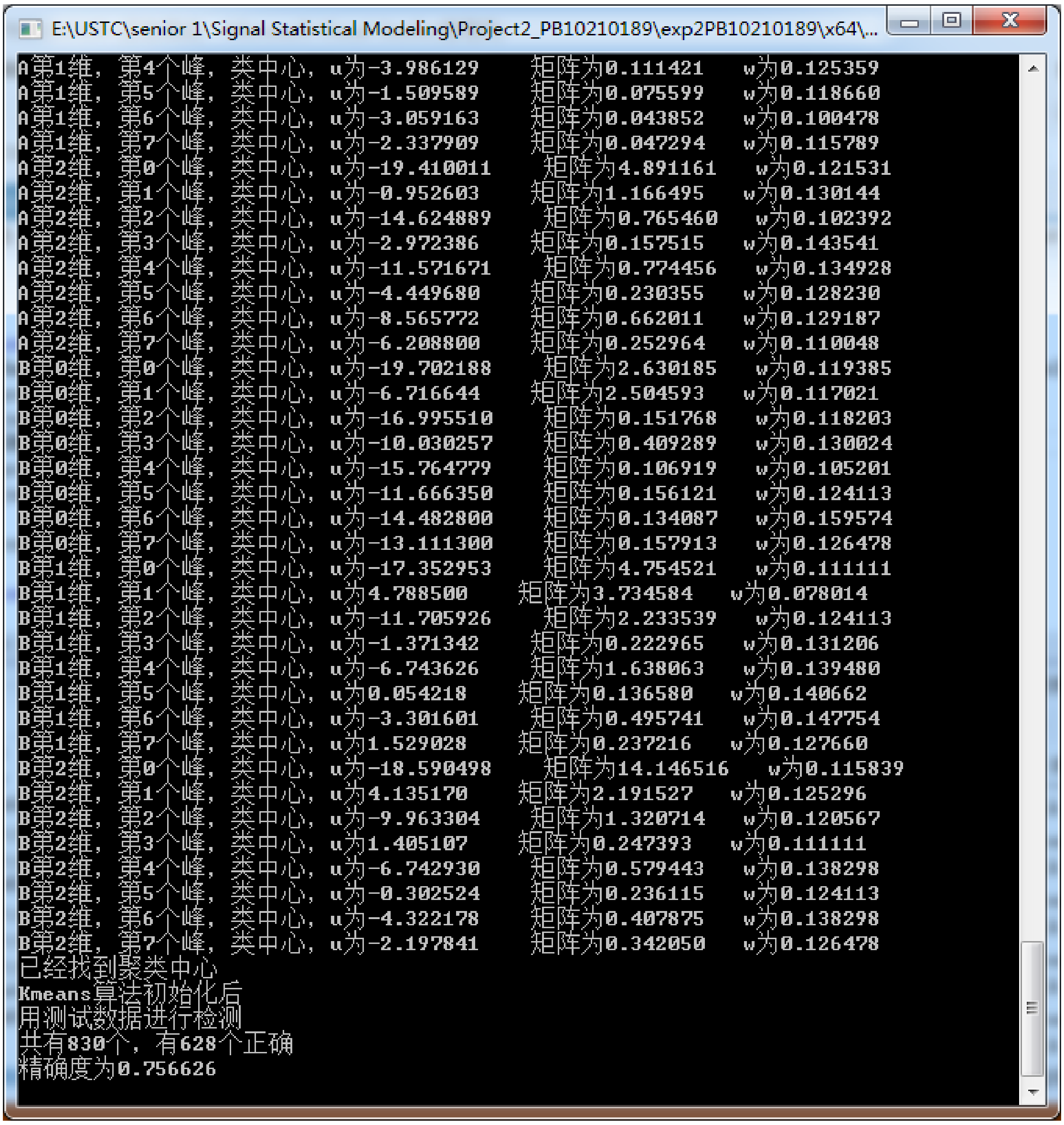
8 个峰时的情况

K-means 算法初始化



E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
第16次迭代
begin to calculate A
begin to get uA
A第0维, 第0个峰, 类中心, u为-15.797527    矩阵为2.805564    w为0.109091
A第0维, 第1个峰, 类中心, u为-6.330319    矩阵为0.888935    w为0.152153
A第0维, 第2个峰, 类中心, u为-12.670024    矩阵为0.210616    w为0.123445
A第0维, 第3个峰, 类中心, u为-8.480420    矩阵为0.159556    w为0.105263
A第0维, 第4个峰, 类中心, u为-11.608656    矩阵为0.046693    w为0.115789
A第0维, 第5个峰, 类中心, u为-9.601445    矩阵为0.054457    w为0.120574
A第0维, 第6个峰, 类中心, u为-10.931445    矩阵为0.032329    w为0.155024
A第0维, 第7个峰, 类中心, u为-10.287265    矩阵为0.032335    w为0.118660
A第1维, 第0个峰, 类中心, u为-13.404835    矩阵为7.413915    w为0.146411
A第1维, 第1个峰, 类中心, u为3.095517    矩阵为3.877868    w为0.118660
A第1维, 第2个峰, 类中心, u为-6.137105    矩阵为0.895790    w为0.118660
A第1维, 第3个峰, 类中心, u为-0.336718    矩阵为0.213877    w为0.155981
```



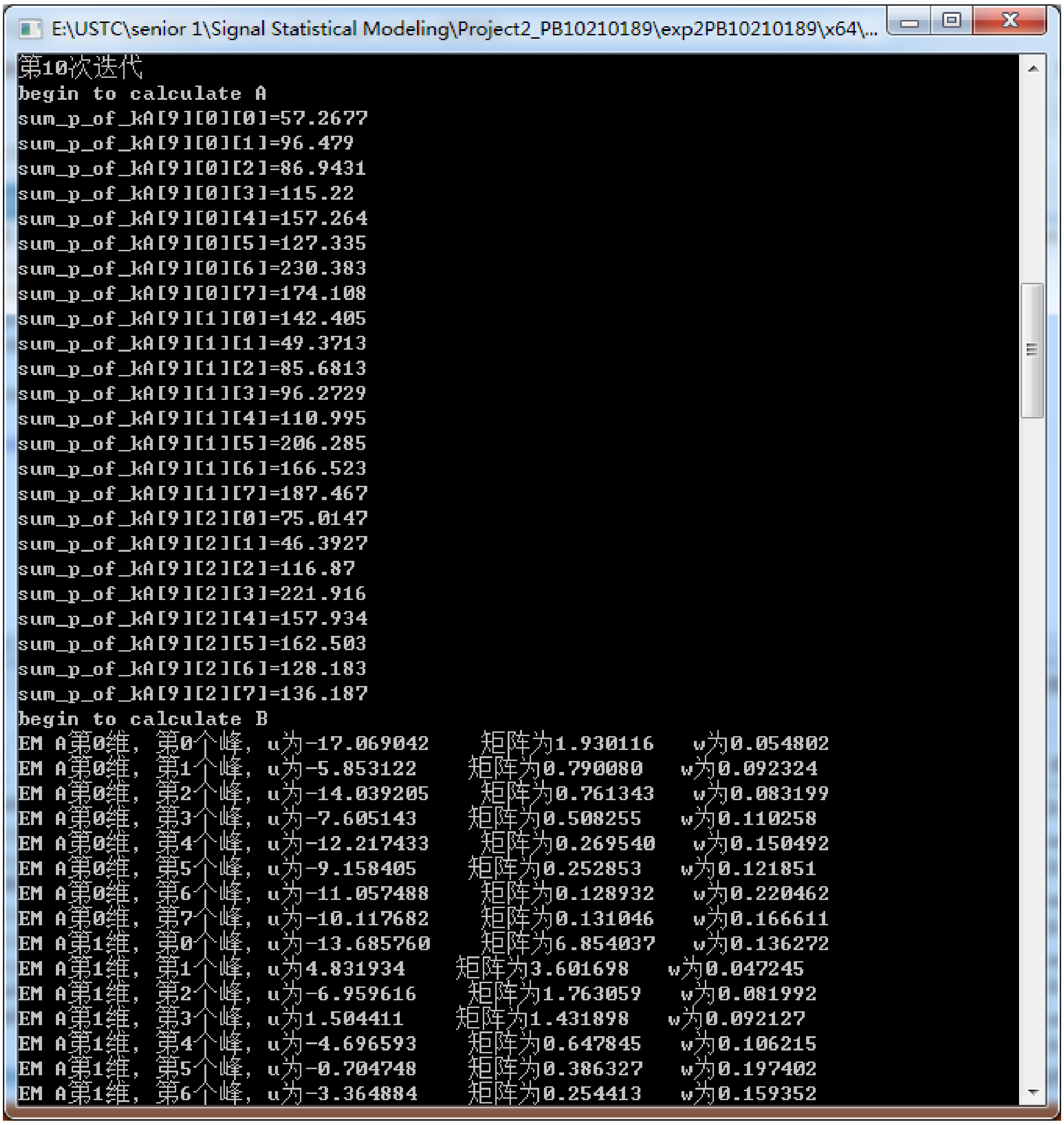
E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
A第1维, 第4个峰, 类中心, u为-3.986129    矩阵为0.111421   w为0.125359
A第1维, 第5个峰, 类中心, u为-1.509589    矩阵为0.075599   w为0.118660
A第1维, 第6个峰, 类中心, u为-3.059163    矩阵为0.043852   w为0.100478
A第1维, 第7个峰, 类中心, u为-2.337909    矩阵为0.047294   w为0.115789
A第2维, 第0个峰, 类中心, u为-19.410011    矩阵为4.891161   w为0.121531
A第2维, 第1个峰, 类中心, u为-0.952603    矩阵为1.166495   w为0.130144
A第2维, 第2个峰, 类中心, u为-14.624889    矩阵为0.765460   w为0.102392
A第2维, 第3个峰, 类中心, u为-2.972386    矩阵为0.157515   w为0.143541
A第2维, 第4个峰, 类中心, u为-11.571671    矩阵为0.774456   w为0.134928
A第2维, 第5个峰, 类中心, u为-4.449680    矩阵为0.230355   w为0.128230
A第2维, 第6个峰, 类中心, u为-8.565772    矩阵为0.662011   w为0.129187
A第2维, 第7个峰, 类中心, u为-6.208800    矩阵为0.252964   w为0.110048
B第0维, 第0个峰, 类中心, u为-19.702188    矩阵为2.630185   w为0.119385
B第0维, 第1个峰, 类中心, u为-6.716644    矩阵为2.504593   w为0.117021
B第0维, 第2个峰, 类中心, u为-16.995510    矩阵为0.151768   w为0.118203
B第0维, 第3个峰, 类中心, u为-10.030257    矩阵为0.409289   w为0.130024
B第0维, 第4个峰, 类中心, u为-15.764779    矩阵为0.106919   w为0.105201
B第0维, 第5个峰, 类中心, u为-11.666350    矩阵为0.156121   w为0.124113
B第0维, 第6个峰, 类中心, u为-14.482800    矩阵为0.134087   w为0.159574
B第0维, 第7个峰, 类中心, u为-13.111300    矩阵为0.157913   w为0.126478
B第1维, 第0个峰, 类中心, u为-17.352953    矩阵为4.754521   w为0.111111
B第1维, 第1个峰, 类中心, u为4.788500    矩阵为3.734584   w为0.078014
B第1维, 第2个峰, 类中心, u为-11.705926    矩阵为2.233539   w为0.124113
B第1维, 第3个峰, 类中心, u为-1.371342    矩阵为0.222965   w为0.131206
B第1维, 第4个峰, 类中心, u为-6.743626    矩阵为1.638063   w为0.139480
B第1维, 第5个峰, 类中心, u为0.054218    矩阵为0.136580   w为0.140662
B第1维, 第6个峰, 类中心, u为-3.301601    矩阵为0.495741   w为0.147754
B第1维, 第7个峰, 类中心, u为1.529028    矩阵为0.237216   w为0.127660
B第2维, 第0个峰, 类中心, u为-18.590498    矩阵为14.146516   w为0.115839
B第2维, 第1个峰, 类中心, u为4.135170    矩阵为2.191527   w为0.125296
B第2维, 第2个峰, 类中心, u为-9.963304    矩阵为1.320714   w为0.120567
B第2维, 第3个峰, 类中心, u为1.405107    矩阵为0.247393   w为0.111111
B第2维, 第4个峰, 类中心, u为-6.742930    矩阵为0.579443   w为0.138298
B第2维, 第5个峰, 类中心, u为-0.302524    矩阵为0.236115   w为0.124113
B第2维, 第6个峰, 类中心, u为-4.322178    矩阵为0.407875   w为0.138298
B第2维, 第7个峰, 类中心, u为-2.197841    矩阵为0.342050   w为0.126478
已经找到聚类中心
Kmeans算法初始化后
用测试数据进行检测
共有830个, 有628个正确
精确度为0.756626
```

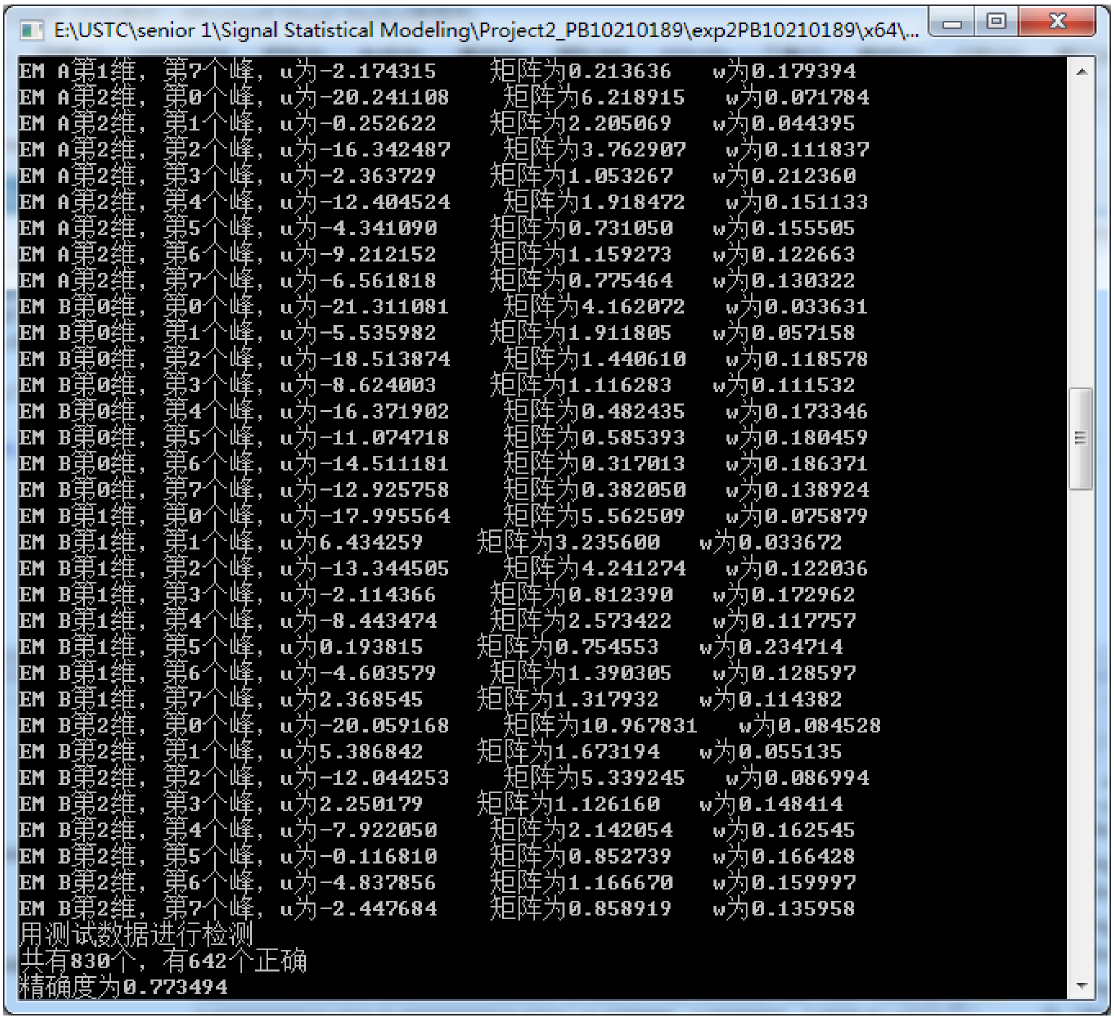
K-means 算法找到聚类中心，参数如上，得到的精度为 0.756626

再用上面的参数进行 EM 算法。

用测试数据进行检测，第 38 次迭代后精度最高，为 0.773494，如下


E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
第10次迭代
begin to calculate A
sum_p_of_kA[9][0][0]=57.2677
sum_p_of_kA[9][0][1]=96.479
sum_p_of_kA[9][0][2]=86.9431
sum_p_of_kA[9][0][3]=115.22
sum_p_of_kA[9][0][4]=157.264
sum_p_of_kA[9][0][5]=127.335
sum_p_of_kA[9][0][6]=230.383
sum_p_of_kA[9][0][7]=174.108
sum_p_of_kA[9][1][0]=142.405
sum_p_of_kA[9][1][1]=49.3713
sum_p_of_kA[9][1][2]=85.6813
sum_p_of_kA[9][1][3]=96.2729
sum_p_of_kA[9][1][4]=110.995
sum_p_of_kA[9][1][5]=206.285
sum_p_of_kA[9][1][6]=166.523
sum_p_of_kA[9][1][7]=187.467
sum_p_of_kA[9][2][0]=75.0147
sum_p_of_kA[9][2][1]=46.3927
sum_p_of_kA[9][2][2]=116.87
sum_p_of_kA[9][2][3]=221.916
sum_p_of_kA[9][2][4]=157.934
sum_p_of_kA[9][2][5]=162.503
sum_p_of_kA[9][2][6]=128.183
sum_p_of_kA[9][2][7]=136.187
begin to calculate B
EM A第0维, 第0个峰, u为-17.069042     矩阵为1.930116    w为0.054802
EM A第0维, 第1个峰, u为-5.853122     矩阵为0.790080    w为0.092324
EM A第0维, 第2个峰, u为-14.039205     矩阵为0.761343    w为0.083199
EM A第0维, 第3个峰, u为-7.605143     矩阵为0.508255    w为0.110258
EM A第0维, 第4个峰, u为-12.217433     矩阵为0.269540    w为0.150492
EM A第0维, 第5个峰, u为-9.158405     矩阵为0.252853    w为0.121851
EM A第0维, 第6个峰, u为-11.057488     矩阵为0.128932    w为0.220462
EM A第0维, 第7个峰, u为-10.117682     矩阵为0.131046    w为0.166611
EM A第1维, 第0个峰, u为-13.685760     矩阵为6.854037    w为0.136272
EM A第1维, 第1个峰, u为4.831934     矩阵为3.601698    w为0.047245
EM A第1维, 第2个峰, u为-6.959616     矩阵为1.763059    w为0.081992
EM A第1维, 第3个峰, u为1.504411     矩阵为1.431898    w为0.092127
EM A第1维, 第4个峰, u为-4.696593     矩阵为0.647845    w为0.106215
EM A第1维, 第5个峰, u为-0.704748     矩阵为0.386327    w为0.197402
EM A第1维, 第6个峰, u为-3.364884     矩阵为0.254413    w为0.159352
```



E:\USTC\senior 1\Signal Statistical Modeling\Project2_PB10210189\exp2PB10210189\x64\...

```
EM A第1维, 第7个峰, u为-2.174315     矩阵为0.213636    w为0.179394
EM A第2维, 第0个峰, u为-20.241108     矩阵为6.218915    w为0.071784
EM A第2维, 第1个峰, u为-0.252622     矩阵为2.205069    w为0.044395
EM A第2维, 第2个峰, u为-16.342487     矩阵为3.762907    w为0.111837
EM A第2维, 第3个峰, u为-2.363729     矩阵为1.053267    w为0.212360
EM A第2维, 第4个峰, u为-12.404524     矩阵为1.918472    w为0.151133
EM A第2维, 第5个峰, u为-4.341090     矩阵为0.731050    w为0.155505
EM A第2维, 第6个峰, u为-9.212152     矩阵为1.159273    w为0.122663
EM A第2维, 第7个峰, u为-6.561818     矩阵为0.775464    w为0.130322
EM B第0维, 第0个峰, u为-21.311081     矩阵为4.162072    w为0.033631
EM B第0维, 第1个峰, u为-5.535982     矩阵为1.911805    w为0.057158
EM B第0维, 第2个峰, u为-18.513874     矩阵为1.440610    w为0.118578
EM B第0维, 第3个峰, u为-8.624003     矩阵为1.116283    w为0.111532
EM B第0维, 第4个峰, u为-16.371902     矩阵为0.482435    w为0.173346
EM B第0维, 第5个峰, u为-11.074718     矩阵为0.585393    w为0.180459
EM B第0维, 第6个峰, u为-14.511181     矩阵为0.317013    w为0.186371
EM B第0维, 第7个峰, u为-12.925758     矩阵为0.382050    w为0.138924
EM B第1维, 第0个峰, u为-17.995564     矩阵为5.562509    w为0.075879
EM B第1维, 第1个峰, u为6.434259    矩阵为3.235600   w为0.033672
EM B第1维, 第2个峰, u为-13.344505     矩阵为4.241274    w为0.122036
EM B第1维, 第3个峰, u为-2.114366     矩阵为0.812390    w为0.172962
EM B第1维, 第4个峰, u为-8.443474     矩阵为2.573422    w为0.117757
EM B第1维, 第5个峰, u为0.193815    矩阵为0.754553   w为0.234714
EM B第1维, 第6个峰, u为-4.603579     矩阵为1.390305    w为0.128597
EM B第1维, 第7个峰, u为2.368545    矩阵为1.317932   w为0.114382
EM B第2维, 第0个峰, u为-20.059168     矩阵为10.967831    w为0.084528
EM B第2维, 第1个峰, u为5.386842    矩阵为1.673194   w为0.055135
EM B第2维, 第2个峰, u为-12.044253     矩阵为5.339245    w为0.086994
EM B第2维, 第3个峰, u为2.250179    矩阵为1.126160   w为0.148414
EM B第2维, 第4个峰, u为-7.922050     矩阵为2.142054    w为0.162545
EM B第2维, 第5个峰, u为-0.116810     矩阵为0.852739    w为0.166428
EM B第2维, 第6个峰, u为-4.837856     矩阵为1.166670    w为0.159997
EM B第2维, 第7个峰, u为-2.447684     矩阵为0.858919    w为0.135958
用测试数据进行检测
共有830个, 有642个正确
精确度为0.773494
```

8 个峰时精度略有下降可能是过拟合的原因。

在实际操作中，由于在 K-means 之前去初始值不能太偏以至于有某一类没有被分到数据，所以另写了一个函数尽可能使各类均匀分到函数。如：有四个峰时，使得有 12.5%的数据小于均值 1，37.5%的数据小于均值 2，62.5%的数据小于均值 3,87.5%的数据小于均值 4。详细函数见后。

三、Consider to improve the ML-trained single Gaussian classifier in step 1 by using some more advanced training techniques. In this part, you are asked to improve the single Gaussian classifier you got in the first step based on one of the discriminative training methods, namely minimum classification error (MCE) estimation with gradient probabilistic descent (GPD) algorithm. Study how the following quantities change as you proceed with more and more iterations in GPD training: 1) the actual error rate in training data; 2) the objective function in MCE; 3) the actual error rate in test set. Please note that in GPD training, the key issue is how to choose a proper step size experimentally.

根据 MCE 公式，

For each training data, (X_t, C_t), define misclassification measure:

$$d(X_t, C_t) = -p(C_t)p(X_t \mid \lambda_{C_t}) + \max_{C \neq C_t} p(C)p(X_t \mid \lambda_C)$$

If $d(X_t, C_t) > 0$, incorrect classification for X_t ➔ 1 error

If $d(X_t, C_t) <= 0$, correct classification for X_t ➔ 0 error

$$Q(\Lambda) \approx Q'(\Lambda) = \sum_{t=1}^{T} l(d(X_t, C_t))$$

where $l(d) = \dfrac{1}{1+e^{-a\cdot d}}$

a>0 is a parameter to control its shape.

根据实验数据观察，a 取 1000000 较为合理。

$$\lambda_i^{(n+1)} = \lambda_i^{(n)} - \varepsilon \cdot \frac{\partial}{\partial \lambda_i} Q'(\lambda_1 \cdots \lambda_N)\Big|_{\lambda_i = \lambda_i^{(n)}}$$

迭代公式

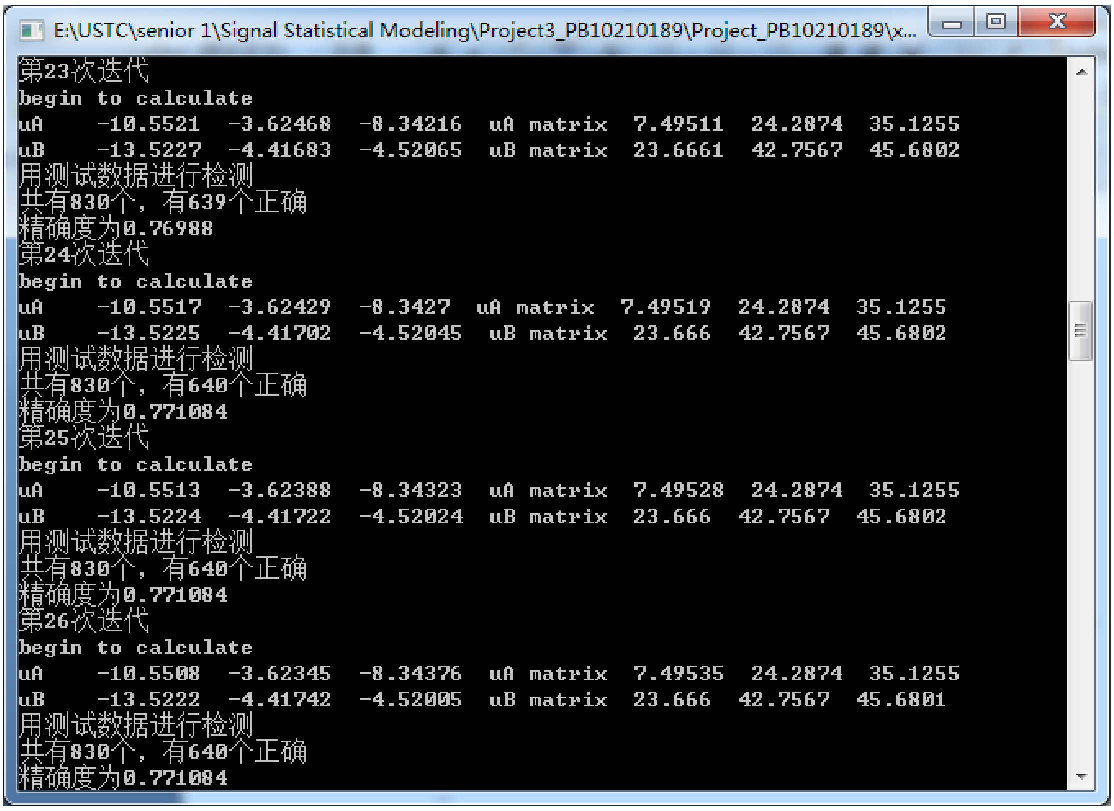
$$\frac{\partial}{\partial \lambda_i} Q'(\lambda_1 \cdots \lambda_N) = \sum_{t=1}^{T} \frac{\partial}{\partial \lambda_i} l[d(X_t, C_t)]$$

$$= \sum_{t=1}^{T} \frac{\partial l(d)}{\partial d} \cdot \frac{\partial d(X_t, C_t)}{\partial \lambda_i}$$

$$= \sum_{t=1}^{T} a \cdot l(d) \cdot [1 - l(d)] \cdot \frac{\partial d(X_t, C_t)}{\partial \lambda_i}$$

根据实验数据，ε 取 0.00001。

第 24 次迭代后测试数据的精度最高，为 0.771084.


E:\USTC\senior 1\Signal Statistical Modeling\Project3_PB10210189\Project_PB10210189\x...

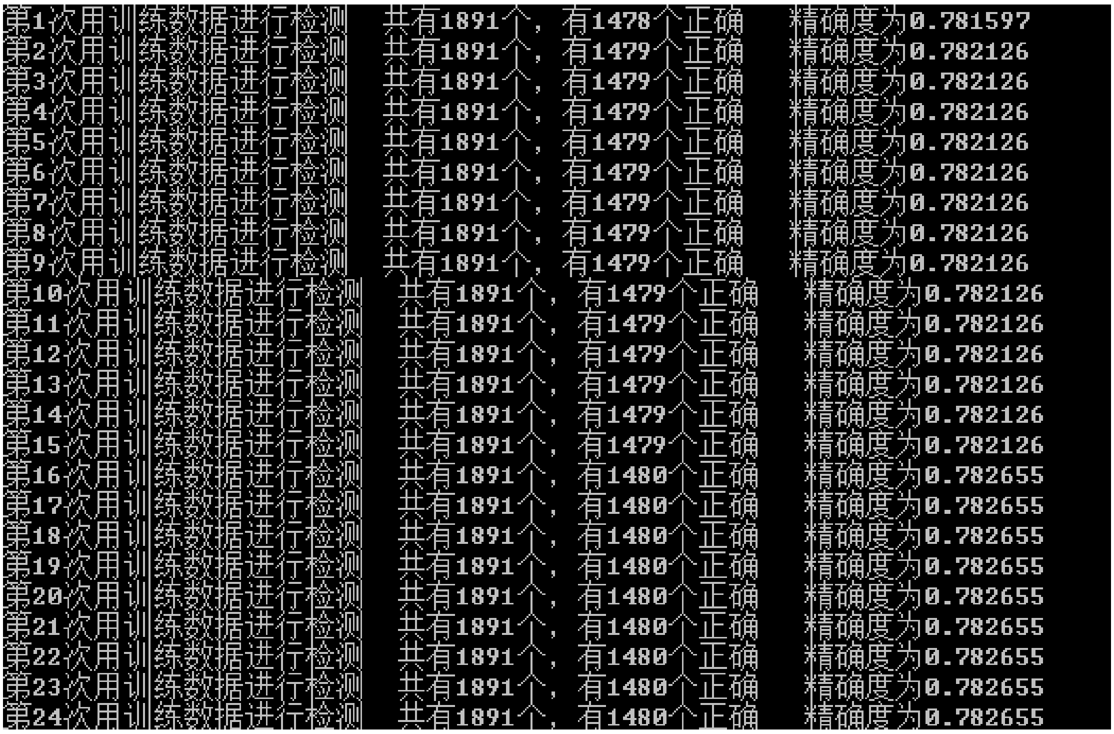
```
第23次迭代
begin to calculate
uA     -10.5521  -3.62468  -8.34216  uA matrix  7.49511  24.2874  35.1255
uB     -13.5227  -4.41683  -4.52065  uB matrix  23.6661  42.7567  45.6802
用测试数据进行检测
共有830个，有639个正确
精确度为0.76988
第24次迭代
begin to calculate
uA     -10.5517  -3.62429  -8.3427  uA matrix  7.49519  24.2874  35.1255
uB     -13.5225  -4.41702  -4.52045  uB matrix  23.666  42.7567  45.6802
用测试数据进行检测
共有830个，有640个正确
精确度为0.771084
第25次迭代
begin to calculate
uA     -10.5513  -3.62388  -8.34323  uA matrix  7.49528  24.2874  35.1255
uB     -13.5224  -4.41722  -4.52024  uB matrix  23.666  42.7567  45.6802
用测试数据进行检测
共有830个，有640个正确
精确度为0.771084
第26次迭代
begin to calculate
uA     -10.5508  -3.62345  -8.34376  uA matrix  7.49535  24.2874  35.1255
uB     -13.5222  -4.41742  -4.52005  uB matrix  23.666  42.7567  45.6801
用测试数据进行检测
共有830个，有640个正确
精确度为0.771084
```

下面分别求 1) the actual error rate in training data; 2) the objective function in MCE; 3) the actual error rate in test set.

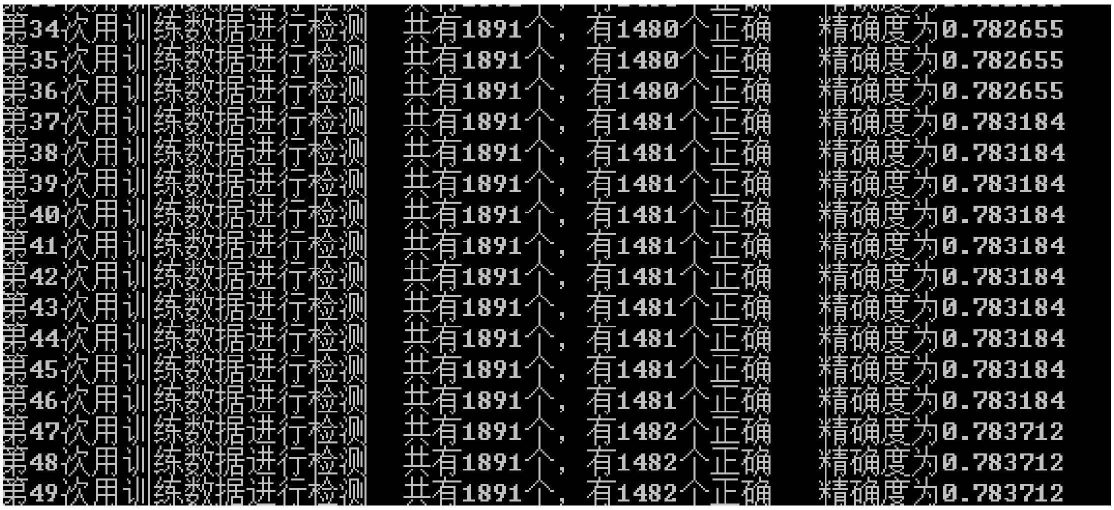
1) the actual error rate in training data

由实验可以发现随着迭代次数不断增加，训练数据的精确度不断上升，且上升不断变慢，第一次迭代 1891 组数据有 1478 个正确，在第 499 次迭代后，1891 组数据有 1485 个正确，精确度 0.785299。

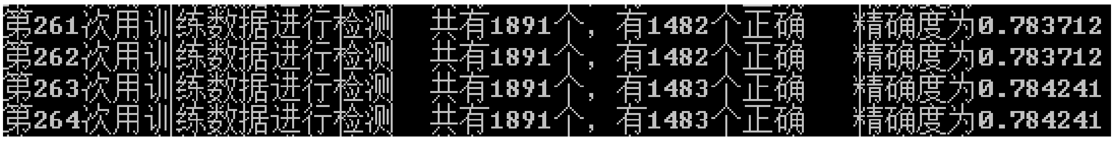
下面列出精确度变化的地方



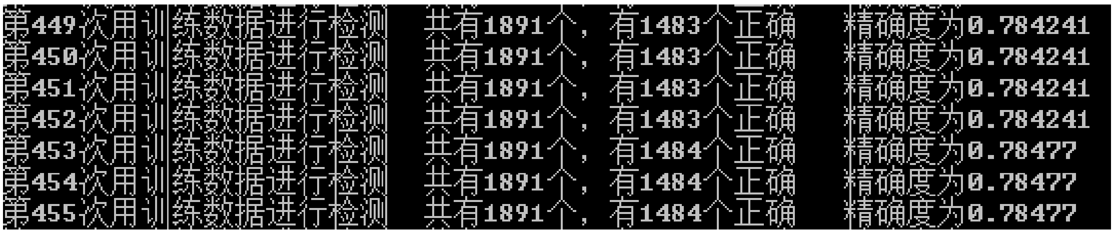
```
第1次用训练数据进行检测  共有1891个，有1478个正确   精确度为0.781597
第2次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第3次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第4次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第5次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第6次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第7次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第8次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第9次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第10次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第11次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第12次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第13次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第14次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第15次用训练数据进行检测  共有1891个，有1479个正确   精确度为0.782126
第16次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第17次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第18次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第19次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第20次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第21次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第22次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第23次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第24次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
```



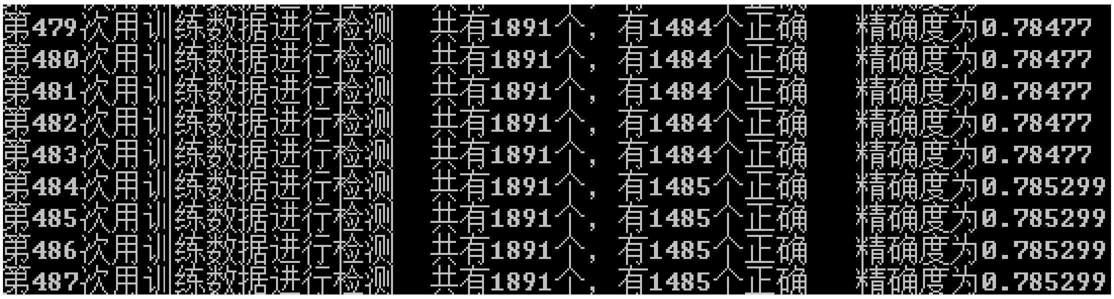
```
第34次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第35次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第36次用训练数据进行检测  共有1891个，有1480个正确   精确度为0.782655
第37次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第38次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第39次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第40次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第41次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第42次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第43次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第44次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第45次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第46次用训练数据进行检测  共有1891个，有1481个正确   精确度为0.783184
第47次用训练数据进行检测  共有1891个，有1482个正确   精确度为0.783712
第48次用训练数据进行检测  共有1891个，有1482个正确   精确度为0.783712
第49次用训练数据进行检测  共有1891个，有1482个正确   精确度为0.783712
```



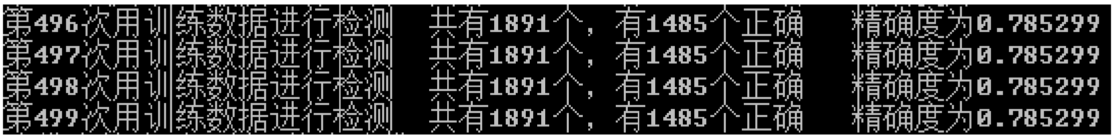
```
第261次用训练数据进行检测  共有1891个，有1482个正确   精确度为0.783712
第262次用训练数据进行检测  共有1891个，有1482个正确   精确度为0.783712
第263次用训练数据进行检测  共有1891个，有1483个正确   精确度为0.784241
第264次用训练数据进行检测  共有1891个，有1483个正确   精确度为0.784241
```



```
第449次用训练数据进行检测   共有1891个, 有1483个正确    精确度为0.784241
第450次用训练数据进行检测   共有1891个, 有1483个正确    精确度为0.784241
第451次用训练数据进行检测   共有1891个, 有1483个正确    精确度为0.784241
第452次用训练数据进行检测   共有1891个, 有1483个正确    精确度为0.784241
第453次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
第454次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
第455次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
```



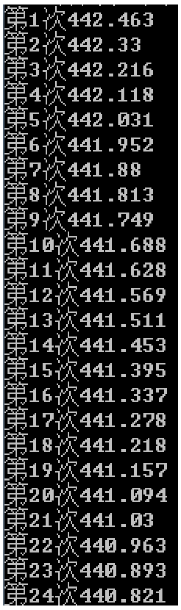
```
第479次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
第480次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
第481次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
第482次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
第483次用训练数据进行检测   共有1891个, 有1484个正确    精确度为0.78477
第484次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
第485次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
第486次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
第487次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
```



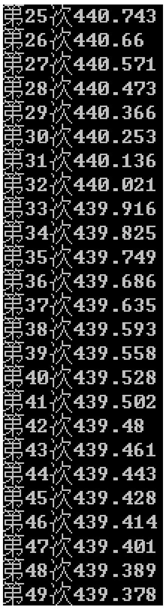
```
第496次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
第497次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
第498次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
第499次用训练数据进行检测   共有1891个, 有1485个正确    精确度为0.785299
```

2) the objective function in MCE

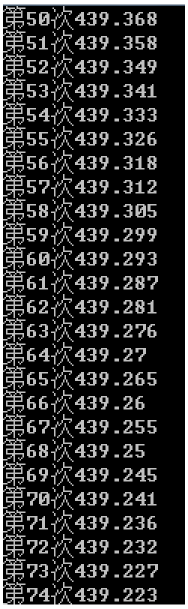
经过实验，目标函数 Q 值不断下降



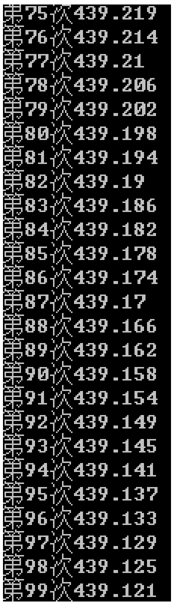
```
第1次442.463
第2次442.33
第3次442.216
第4次442.118
第5次442.031
第6次441.952
第7次441.88
第8次441.813
第9次441.749
第10次441.688
第11次441.628
第12次441.569
第13次441.511
第14次441.453
第15次441.395
第16次441.337
第17次441.278
第18次441.218
第19次441.157
第20次441.094
第21次441.03
第22次440.963
第23次440.893
第24次440.821
```



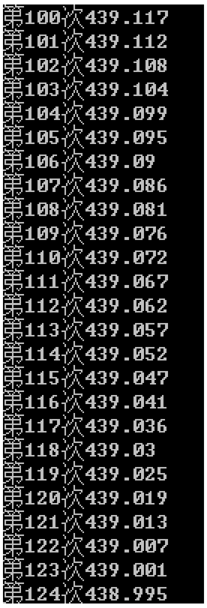
```
第25次440.743
第26次440.66
第27次440.571
第28次440.473
第29次440.366
第30次440.253
第31次440.136
第32次440.021
第33次439.916
第34次439.825
第35次439.749
第36次439.686
第37次439.635
第38次439.593
第39次439.558
第40次439.528
第41次439.502
第42次439.48
第43次439.461
第44次439.443
第45次439.428
第46次439.414
第47次439.401
第48次439.389
第49次439.378
```



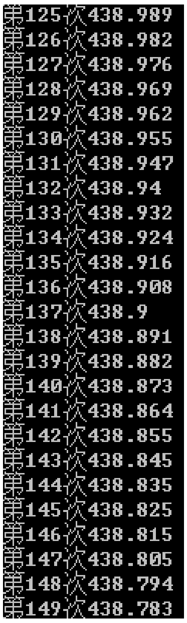
```
第50次439.368
第51次439.358
第52次439.349
第53次439.341
第54次439.333
第55次439.326
第56次439.318
第57次439.312
第58次439.305
第59次439.299
第60次439.293
第61次439.287
第62次439.281
第63次439.276
第64次439.27
第65次439.265
第66次439.26
第67次439.255
第68次439.25
第69次439.245
第70次439.241
第71次439.236
第72次439.232
第73次439.227
第74次439.223
```



```
第75次439.219
第76次439.214
第77次439.21
第78次439.206
第79次439.202
第80次439.198
第81次439.194
第82次439.19
第83次439.186
第84次439.182
第85次439.178
第86次439.174
第87次439.17
第88次439.166
第89次439.162
第90次439.158
第91次439.154
第92次439.149
第93次439.145
第94次439.141
第95次439.137
第96次439.133
第97次439.129
第98次439.125
第99次439.121
```



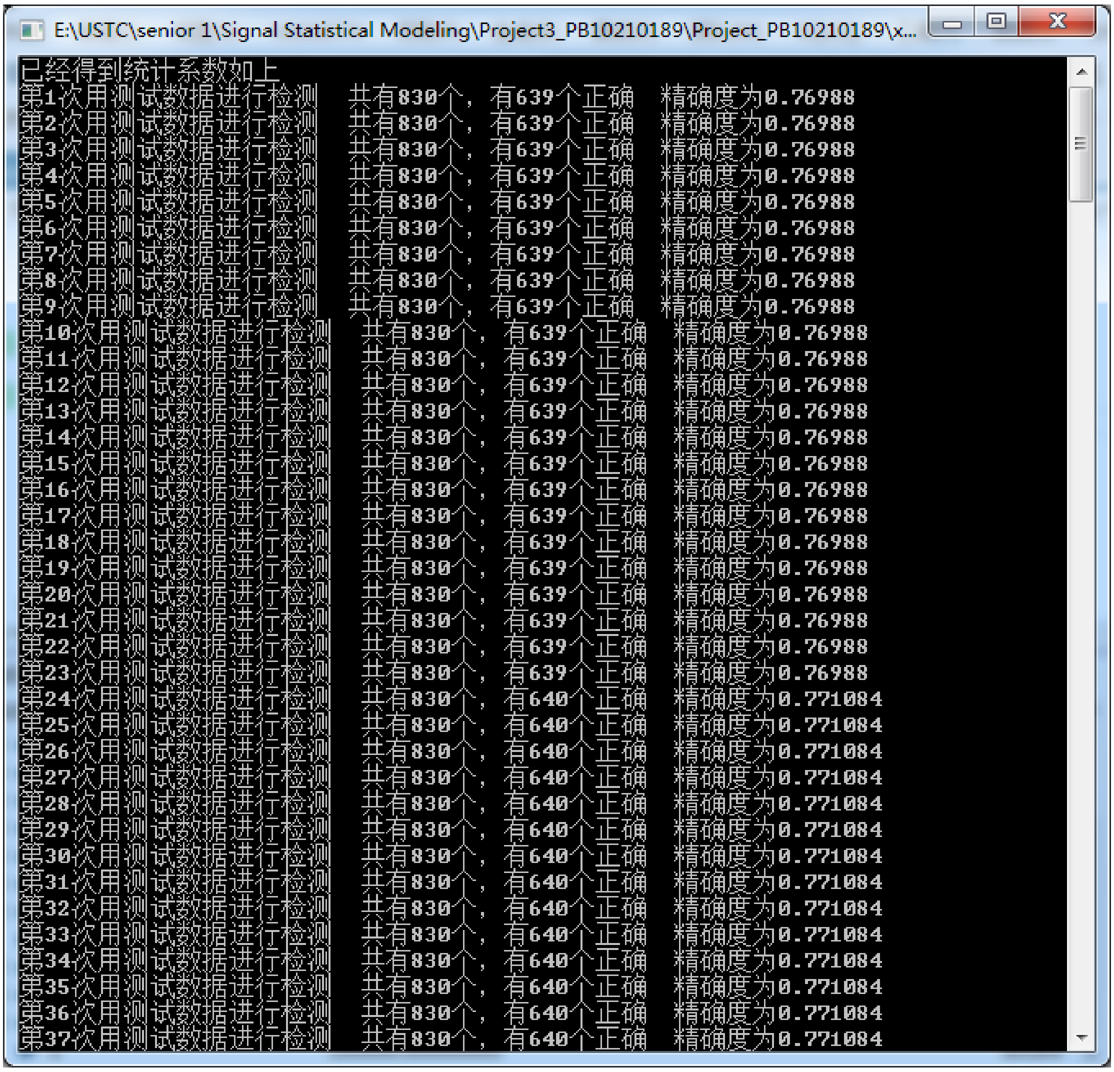
```
第100次439.117
第101次439.112
第102次439.108
第103次439.104
第104次439.099
第105次439.095
第106次439.09
第107次439.086
第108次439.081
第109次439.076
第110次439.072
第111次439.067
第112次439.062
第113次439.057
第114次439.052
第115次439.047
第116次439.041
第117次439.036
第118次439.03
第119次439.025
第120次439.019
第121次439.013
第122次439.007
第123次439.001
第124次438.995
```



```
第125次438.989
第126次438.982
第127次438.976
第128次438.969
第129次438.962
第130次438.955
第131次438.947
第132次438.94
第133次438.932
第134次438.924
第135次438.916
第136次438.908
第137次438.9
第138次438.891
第139次438.882
第140次438.873
第141次438.864
第142次438.855
第143次438.845
第144次438.835
第145次438.825
第146次438.815
第147次438.805
第148次438.794
第149次438.783
```

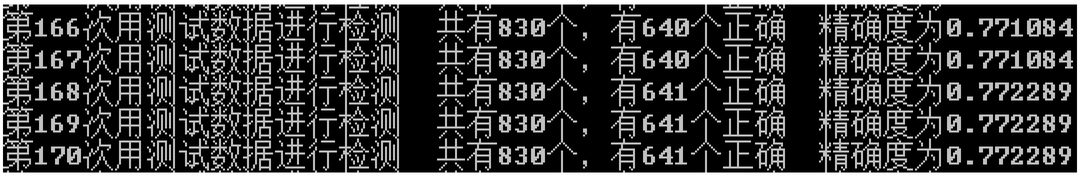
3) the actual error rate in test set

发现测试集中实际错误率降低，之后略有上升。

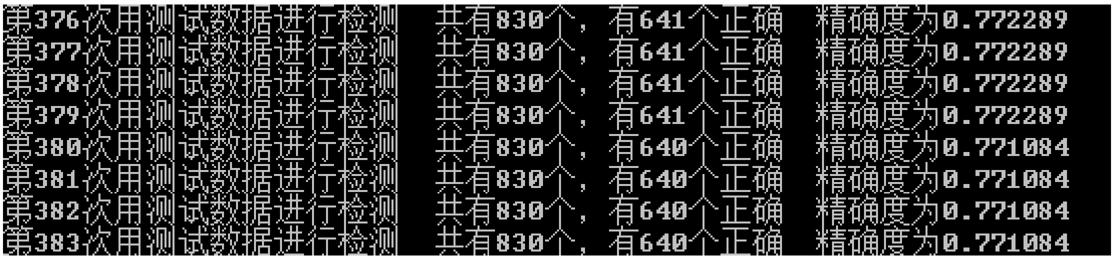


E:\USTC\senior 1\Signal Statistical Modeling\Project3_PB10210189\Project_PB10210189\x...

```
已经得到统计系数如上
第1次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第2次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第3次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第4次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第5次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第6次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第7次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第8次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第9次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第10次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第11次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第12次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第13次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第14次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第15次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第16次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第17次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第18次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第19次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第20次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第21次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第22次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第23次用测试数据进行检测  共有830个，有639个正确  精确度为0.76988
第24次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第25次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第26次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第27次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第28次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第29次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第30次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第31次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第32次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第33次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第34次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第35次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第36次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第37次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
```



```
第166次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第167次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第168次用测试数据进行检测  共有830个，有641个正确  精确度为0.772289
第169次用测试数据进行检测  共有830个，有641个正确  精确度为0.772289
第170次用测试数据进行检测  共有830个，有641个正确  精确度为0.772289
```



```
第376次用测试数据进行检测  共有830个，有641个正确  精确度为0.772289
第377次用测试数据进行检测  共有830个，有641个正确  精确度为0.772289
第378次用测试数据进行检测  共有830个，有641个正确  精确度为0.772289
第379次用测试数据进行检测  共有830个，有641个正确  精确度为0.772289
第380次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第381次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第382次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
第383次用测试数据进行检测  共有830个，有640个正确  精确度为0.771084
```

代码(共 2228 行)

实验一（291行）

```
// exp1PB10210189.cpp : 定义控制台应用程序的入口点。
//writer: Bodong Zhang
#include "stdafx.h"
#include"math.h"
#include <opencv.hpp>
#include <stdlib.h>
#include"highgui.h"
#include <fstream>
#include <iostream> // not required by most systems
using namespace std;//!!!!!!!!!!!!!!!!!!!can not use ofstream without this line
#define PI 3.1415926
void load_train_data(float trainA[1045][3],float trainB[846][3])
{
    char pathname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\train.txt";
    ifstream o_file;
    o_file.open(pathname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    for(Aroll=0;Aroll<1045;Aroll++)
    {
        o_file>>trainA[Aroll][0];//从文件输入一个数值。
        //cout<<trainA[Aroll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][1];//从文件输入一个数值。
        //cout<<trainA[Aroll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][2];//从文件输入一个数值。
        //cout<<trainA[Aroll][2]<<"\n"<<endl;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    for(Broll=0;Broll<845;Broll++)
    {
        o_file>>trainB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    o_file>>trainB[Broll][0];//从文件输入一个数值。
    //cout<<trainB[Broll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>trainB[Broll][1];//从文件输入一个数值。
    //cout<<trainB[Broll][1]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>trainB[Broll][2];//从文件输入一个数值。
    //cout<<trainB[Broll][2]<<"\n"<<endl;
    o_file.close();
}




void get_u_matrix(float trainA[1045][3],float trainB[846][3],float train_uA[3],float
train_matrixA[3][3],float train_uB[3],float train_matrixB[3][3])
{
    train_uA[0]=0.0;train_uA[1]=0.0;train_uA[2]=0.0;
    train_uB[0]=0.0;train_uB[1]=0.0;train_uB[2]=0.0;
    train_matrixA[0][0]=0.0;train_matrixA[0][1]=0.0;train_matrixA[0][2]=0.0;
    train_matrixA[1][0]=0.0;train_matrixA[1][1]=0.0;train_matrixA[1][2]=0.0;
    train_matrixA[2][0]=0.0;train_matrixA[2][1]=0.0;train_matrixA[2][2]=0.0;
    train_matrixB[0][0]=0.0;train_matrixB[0][1]=0.0;train_matrixB[0][2]=0.0;
    train_matrixB[1][0]=0.0;train_matrixB[1][1]=0.0;train_matrixB[1][2]=0.0;
    train_matrixB[2][0]=0.0;train_matrixB[2][1]=0.0;train_matrixB[2][2]=0.0;
    int i;
    for(i=0;i<1045;i++)
    {
        train_uA[0]+=trainA[i][0];
        train_uA[1]+=trainA[i][1];
        train_uA[2]+=trainA[i][2];
    }
    train_uA[0]=train_uA[0]/1045;
    train_uA[1]=train_uA[1]/1045;
    train_uA[2]=train_uA[2]/1045;
    cout<<"train_uA[0]"<<train_uA[0]<<"   ";
```

```
cout<<"train_uA[1]"<<train_uA[1]<<"   ";
cout<<"train_uA[2]"<<train_uA[2]<<"\n";
for(i=0;i<846;i++)
{
    train_uB[0]+=trainB[i][0];
    train_uB[1]+=trainB[i][1];
    train_uB[2]+=trainB[i][2];
}
train_uB[0]=train_uB[0]/846;
train_uB[1]=train_uB[1]/846;
train_uB[2]=train_uB[2]/846;
cout<<"train_uB[0]"<<train_uB[0]<<"   ";
cout<<"train_uB[1]"<<train_uB[1]<<"   ";
cout<<"train_uB[2]"<<train_uB[2]<<"\n";
for(i=0;i<1045;i++)
{
    train_matrixA[0][0]+=(trainA[i][0]-train_uA[0])*(trainA[i][0]-train_uA[0]);
    train_matrixA[1][1]+=(trainA[i][1]-train_uA[1])*(trainA[i][1]-train_uA[1]);
    train_matrixA[2][2]+=(trainA[i][2]-train_uA[2])*(trainA[i][2]-train_uA[2]);
}
train_matrixA[0][0]=train_matrixA[0][0]/1045.0;
train_matrixA[1][1]=train_matrixA[1][1]/1045.0;
train_matrixA[2][2]=train_matrixA[2][2]/1045.0;
for(i=0;i<846;i++)
{
    train_matrixB[0][0]+=(trainB[i][0]-train_uA[0])*(trainB[i][0]-train_uA[0]);
    train_matrixB[1][1]+=(trainB[i][1]-train_uB[1])*(trainB[i][1]-train_uB[1]);
    train_matrixB[2][2]+=(trainB[i][2]-train_uB[2])*(trainB[i][2]-train_uB[2]);
}
train_matrixB[0][0]=train_matrixB[0][0]/846.0;
train_matrixB[1][1]=train_matrixB[1][1]/846.0;
train_matrixB[2][2]=train_matrixB[2][2]/846.0;

  /*
for(i=0;i<1045;i++)//实际上协方差系数较大
{
    train_matrixA[0][1]+=(trainA[i][0]-train_uA[0])*(trainA[i][1]-train_uA[1]);
    train_matrixA[0][2]+=(trainA[i][0]-train_uA[0])*(trainA[i][2]-train_uA[2]);
    train_matrixA[1][0]+=(trainA[i][1]-train_uA[1])*(trainA[i][0]-train_uA[0]);
    train_matrixA[1][2]+=(trainA[i][1]-train_uA[1])*(trainA[i][2]-train_uA[2]);
    train_matrixA[2][0]+=(trainA[i][2]-train_uA[2])*(trainA[i][0]-train_uA[0]);
    train_matrixA[2][1]+=(trainA[i][2]-train_uA[2])*(trainA[i][1]-train_uA[1]);
}
```

```
    for(i=0;i<846;i++)
    {
        train_matrixB[0][1]+=(trainB[i][0]-train_uB[0])*(trainB[i][1]-train_uB[1]);
        train_matrixB[0][2]+=(trainB[i][0]-train_uB[0])*(trainB[i][2]-train_uB[2]);
        train_matrixB[1][0]+=(trainB[i][1]-train_uB[1])*(trainB[i][0]-train_uB[0]);
        train_matrixB[1][2]+=(trainB[i][1]-train_uB[1])*(trainB[i][2]-train_uB[2]);
        train_matrixB[2][0]+=(trainB[i][2]-train_uB[2])*(trainB[i][0]-train_uB[0]);
        train_matrixB[2][1]+=(trainB[i][2]-train_uB[2])*(trainB[i][1]-train_uB[1]);
    }
      */
    train_matrixA[0][1]=train_matrixA[0][1]/1045.0;
    train_matrixA[0][2]=train_matrixA[0][2]/1045.0;
    train_matrixA[1][0]=train_matrixA[1][0]/1045.0;
    train_matrixA[1][2]=train_matrixA[1][2]/1045.0;
    train_matrixA[2][0]=train_matrixA[2][0]/1045.0;
    train_matrixA[2][1]=train_matrixA[2][1]/1045.0;

    train_matrixB[0][1]=train_matrixB[0][1]/846.0;
    train_matrixB[0][2]=train_matrixB[0][2]/846.0;
    train_matrixB[1][0]=train_matrixB[1][0]/846.0;
    train_matrixB[1][2]=train_matrixB[1][2]/846.0;
    train_matrixB[2][0]=train_matrixB[2][0]/846.0;
    train_matrixB[2][1]=train_matrixB[2][1]/846.0;

    cout<<"train_matrixA\n"<<train_matrixA[0][0]<<"  "<<train_matrixA[0][1]<<"
"<<train_matrixA[0][2]<<"\n";
    cout<<train_matrixA[1][0]<<"  "<<train_matrixA[1][1]<<"
"<<train_matrixA[1][2]<<"\n";
    cout<<train_matrixA[2][0]<<"  "<<train_matrixA[2][1]<<"
"<<train_matrixA[2][2]<<"\n";

    cout<<"train_matrixB\n"<<train_matrixB[0][0]<<"  "<<train_matrixB[0][1]<<"
"<<train_matrixB[0][2]<<"\n";
    cout<<train_matrixB[1][0]<<"  "<<train_matrixB[1][1]<<"
"<<train_matrixB[1][2]<<"\n";
    cout<<train_matrixB[2][0]<<"  "<<train_matrixB[2][1]<<"
"<<train_matrixB[2][2]<<"\n";

}


void judge(/*float test[3],*/float train_uA[3],float train_matrixA[3][3],float
train_uB[3],float train_matrixB[3][3])//判断某个数据属于哪一类
{
    float testA[470][3],testB[360][3];
    float right=0.0;float sum=0.0;
    char pathtestname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\test.txt";
    ifstream o_file;
    o_file.open(pathtestname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    float Areverse_matrix[3],Breverse_matrix[3],valueA,valueB;
    Areverse_matrix[0]=1.0/train_matrixA[0][0];
    Areverse_matrix[1]=1.0/train_matrixA[1][1];
    Areverse_matrix[2]=1.0/train_matrixA[2][2];
    Breverse_matrix[0]=1.0/train_matrixB[0][0];
    Breverse_matrix[1]=1.0/train_matrixB[1][1];
    Breverse_matrix[2]=1.0/train_matrixB[2][2];
    for(Aroll=0;Aroll<470;Aroll++)
    {
        o_file>>testA[Aroll][0];//从文件输入一个数值。
        //cout<<trainA[Aroll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testA[Aroll][1];//从文件输入一个数值。
        //cout<<trainA[Aroll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testA[Aroll][2];//从文件输入一个数值。
        //cout<<trainA[Aroll][2]<<"\n"<<endl;

    valueA=exp(-0.5*((testA[Aroll][0]-train_uA[0])*(testA[Aroll][0]-train_uA[0])*Areverse_matrix[0]+(testA[Aroll][1]-train_uA[1])*(testA[Aroll][1]-train_uA[1])*Areverse_matrix[1]+(testA[Aroll][2]-train_uA[2])*(testA[Aroll][2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));

    valueB=exp(-0.5*((testA[Aroll][0]-train_uB[0])*(testA[Aroll][0]-train_uB[0])*Breverse_matrix[0]+(testA[Aroll][1]-train_uB[1])*(testA[Aroll][1]-train_uB[1])*Breverse_matrix[1]+(testA[Aroll][2]-train_uB[2])*(testA[Aroll][2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
        sum=sum+1.0;
        if(valueA>valueB)
            right=right+1.0;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    for(Broll=0;Broll<359;Broll++)
    {
        o_file>>testB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;

    valueA=exp(-0.5*((testB[Broll][0]-train_uA[0])*(testB[Broll][0]-train_uA[0])*Areverse_matrix[0]+(testB[Broll][1]-train_uA[1])*(testB[Broll][1]-train_uA[1])*Areverse_matrix[1]+(testB[Broll][2]-train_uA[2])*(testB[Broll][2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));

    valueB=exp(-0.5*((testB[Broll][0]-train_uB[0])*(testB[Broll][0]-train_uB[0])*Breverse_matrix[0]+(testB[Broll][1]-train_uB[1])*(testB[Broll][1]-train_uB[1])*Breverse_matrix[1]+(testB[Broll][2]-train_uB[2])*(testB[Broll][2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
        sum=sum+1.0;
        if(valueA<valueB)
            right=right+1.0;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    o_file>>testB[Broll][0];//从文件输入一个数值。
    //cout<<trainB[Broll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testB[Broll][1];//从文件输入一个数值。
    //cout<<trainB[Broll][1]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testB[Broll][2];//从文件输入一个数值。
    //cout<<trainB[Broll][2]<<"\n"<<endl;
    valueA=exp(-0.5*((testB[Broll][0]-train_uA[0])*(testB[Broll][0]-train_uA[0])*Areverse_matrix[0]+(testB[Broll][1]-train_uA[1])*(testB[Broll][1]-train_uA[1])*Areverse_matrix[1]+(testB[Broll][2]-train_uA[2])*(testB[Broll][2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));
    valueB=exp(-0.5*((testB[Broll][0]-train_uB[0])*(testB[Broll][0]-train_uB[0])*Breverse_matrix[0]+(testB[Broll][1]-train_uB[1])*(testB[Broll][1]-train_uB[1])*Breverse_matrix[1]+(testB[Broll][2]-train_uB[2])*(testB[Broll][2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
    sum=sum+1.0;
    if(valueA<valueB)
        right=right+1.0;
    float accuracy=right/sum;
    cout<<"用测试数据进行检测\n共有"<<sum<<"个，有"<<right<<"个正确\n精确度为"<<accuracy;
    o_file.close();
}

int _tmain(int argc, _TCHAR* argv[])
{
    float trainA[1045][3],trainB[846][3];//A有1045行数据，B有846行数据
    load_train_data(trainA,trainB);//把数据读到数组里
    float train_uA[3],train_matrixA[3][3],train_uB[3],train_matrixB[3][3];
    get_u_matrix(trainA,trainB,train_uA,train_matrixA,train_uB,train_matrixB);
    cout<<"已经得到统计系数如上\n";
    judge(train_uA,train_matrixA,train_uB,train_matrixB);//判断数据属于哪一类


    /*
    char pathname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\train.txt";
    ifstream o_file;
    o_file.open(pathname);
    float i,j,k,l;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    o_file>>i;//从文件输入一个数值。
    cout<<i<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>j;//从文件输入一个数值。
    cout<<j<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>k;//从文件输入一个数值。
    cout<<k<<"\n"<<endl;
    */

    //o_file.seekg(1,ios::cur);

    //ifstream fin;
    //fin.get();
    //get成员函数会读取一个字符包括空格 和换行

    //o_file.seekg(4,ios::cur);//读指针位置向后4格
    //o_file>>l;//从文件输入一个数值。
    //cout<<l<<"\n"<<endl;
    //printf("sdfa%fdasfa",i);
    /*
    char q,w,e,r;
    o_file>>q;
    cout<<q<<"\n";
    o_file>>w;
    cout<<w<<"\n";
    o_file>>e;
    cout<<e<<"\n";
    o_file>>r;
    cout<<r<<"\n";
    o_file>>l;//从文件输入一个数值。
    cout<<l<<"\n"<<endl;
    o_file.close();
    */
    return 0;
};
```

实验二

K-means 初始化前得到较为合理的参数的代码（414 行）

```
// get_stat.cpp : 定义控制台应用程序的入口点。
//PB10210189 writer: Bodong Zhang

#include "stdafx.h"
#include"math.h"
#include <opencv.hpp>
#include <stdlib.h>
#include"highgui.h"
#include <fstream>
#include <iostream> // not required by most systems
using namespace std;//!!!!!!!!!!!!!!!!!!!!can not use ofstream without this line
#define PI 3.1415926

void load_train_data(float trainA[1045][3],float trainB[846][3])
{
    char pathname[356]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\train.txt";
    ifstream o_file;
    o_file.open(pathname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    for(Aroll=0;Aroll<1045;Aroll++)
    {
        o_file>>trainA[Aroll][0];//从文件输入一个数值。
        //cout<<trainA[Aroll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][1];//从文件输入一个数值。
        //cout<<trainA[Aroll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][2];//从文件输入一个数值。
        //cout<<trainA[Aroll][2]<<"\n"<<endl;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
```

```
        for(Broll=0;Broll<845;Broll++)
        {
            o_file>>trainB[Broll][0];//从文件输入一个数值。
            //cout<<trainB[Broll][0]<<"\n"<<endl;
            o_file.seekg(1,ios::cur);
            o_file>>trainB[Broll][1];//从文件输入一个数值。
            //cout<<trainB[Broll][1]<<"\n"<<endl;
            o_file.seekg(1,ios::cur);
            o_file>>trainB[Broll][2];//从文件输入一个数值。
            //cout<<trainB[Broll][2]<<"\n"<<endl;
            o_file>>a1;
            o_file>>a1;
            o_file>>a1;
            o_file>>a1;//转到下一行
        }
        o_file>>trainB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;
        o_file.close();
    }




    int _tmain(int argc, _TCHAR* argv[])
    {
        float trainA[1045][3],trainB[846][3];//A有1045行数据，B有846行数据
        load_train_data(trainA,trainB);//把数据读到数组里
        float train_uA[3],train_matrixA[3][3],train_uB[3],train_matrixB[3][3];

        int stat1,stat2,stat3;
        int sum[6]={0,0,0,0,0,0};
        int flag[6][8];
        for(int ii=0;ii<6;ii++)
            for(int jj=0;jj<8;jj++)
                flag[ii][jj]=1;
        //use when 4 peaks
        /*
        for(float countnum=-35.0;countnum<10.0;countnum=countnum+0.5)
        {
            stat1=0;
            for(int i=0;i<1045;i++)
            {
                if(trainA[i][0]>countnum&&trainA[i][0]<countnum+0.5)
                    stat1++;
            }
            sum[0]+=stat1;


            for(int ii=0;ii<4;ii++)
            {
                if(flag[0][ii]==1&&sum[0]>=131+1045*ii/4)//1045*0.135=131
                {
                    printf("A0 4个峰%f\n",countnum);
                    flag[0][ii]=0;
                }
            }
            //printf("group A  dimmension 0   %f  to  %f  有%d个
    \n",countnum,countnum+0.5,stat1);
        }





        printf("\n");
        for(float countnum=-35.0;countnum<10.0;countnum=countnum+0.5)
        {
            stat2=0;
            for(int i=0;i<1045;i++)
            {
                if(trainA[i][1]>countnum&&trainA[i][1]<countnum+0.5)
                    stat2++;
            }
            sum[1]+=stat2;
            for(int ii=0;ii<4;ii++)
            {
                if(flag[1][ii]==1&&sum[1]>=131+1045*ii/4)//1045*0.135=131
                {
                    printf("A1 4个峰%f\n",countnum);
                    flag[1][ii]=0;
                }
            }



            //printf("group A  dimmension 1   %f  to  %f  有%d个
    \n",countnum,countnum+0.5,stat2);
        }

        printf("\n");
        for(float countnum=-35.0;countnum<6.0;countnum=countnum+0.5)
        {
            stat3=0;
            for(int i=0;i<1045;i++)
            {
                if(trainA[i][2]>countnum&&trainA[i][2]<countnum+0.5)
                    stat3++;
            }

            sum[2]+=stat3;
            for(int ii=0;ii<4;ii++)
            {
                if(flag[2][ii]==1&&sum[2]>=131+1045*ii/4)//1045*0.135=131
                {
                    printf("A2 4个峰%f\n",countnum);
                    flag[2][ii]=0;
                }
            }

            //printf("group A  dimmension 2   %f  to  %f  有%d个
    \n",countnum,countnum+0.5,stat3);
        }

    printf("\n");
        for(float countnum=-35.0;countnum<0.0;countnum=countnum+0.5)
        {
            stat1=0;
            for(int i=0;i<846;i++)
            {
                if(trainB[i][0]>countnum&&trainB[i][0]<countnum+0.5)
                    stat1++;
            }
            sum[3]+=stat1;
            for(int ii=0;ii<4;ii++)
            {
                if(flag[3][ii]==1&&sum[3]>=106+846*ii/4)//846*0.135=106
                {
                    printf("B0 4个峰%f\n",countnum);
                    flag[3][ii]=0;
                }
            }



            //printf("group B  dimmension 0   %f  to  %f  有%d个
    \n",countnum,countnum+0.5,stat1);
        }
        printf("\n");
        for(float countnum=-35.0;countnum<10.0;countnum=countnum+0.5)
        {
            stat2=0;
            for(int i=0;i<846;i++)
            {
                if(trainB[i][1]>countnum&&trainB[i][1]<countnum+0.5)
                    stat2++;
            }
            sum[4]+=stat2;
            for(int ii=0;ii<4;ii++)
            {
                if(flag[4][ii]==1&&sum[4]>=106+846*ii/4)//846*0.135=106
                {
                    printf("B1 4个峰%f\n",countnum);
                    flag[4][ii]=0;
                }
            }

            //printf("group B  dimmension 1   %f  to  %f  有%d个
    \n",countnum,countnum+0.5,stat2);
        }

        printf("\n");
        for(float countnum=-35.0;countnum<8.0;countnum=countnum+0.5)
        {
            stat3=0;
            for(int i=0;i<846;i++)
            {
                if(trainB[i][2]>countnum&&trainB[i][2]<countnum+0.5)
```

```
                stat3++;
        }

        sum[5]+=stat3;
        for(int ii=0;ii<4;ii++)
        {
            if(flag[5][ii]==1&&sum[5]>=106+846*ii/4)//846*0.135=106
            {
                printf("B2 4个峰%f\n",countnum);
                flag[5][ii]=0;
            }
        }


        //printf("group B  dimmension 2   %f  to  %f  有%d个
\n",countnum,countnum+0.5,stat3);
    }

    */




    for(float countnum=-35.0;countnum<10.0;countnum=countnum+0.1)
    {
        stat1=0;
        for(int i=0;i<1045;i++)
        {
            if(trainA[i][0]>countnum&&trainA[i][0]<countnum+0.1)
                stat1++;
        }
        sum[0]+=stat1;


        for(int ii=0;ii<8;ii++)
        {
            if(flag[0][ii]==1&&sum[0]>=65+1045*ii/8)//1045/16=65
            {
                printf("A0 8个峰%f\n",countnum);
                flag[0][ii]=0;
            }
        }
        //printf("group A  dimmension 0   %f  to  %f  有%d个
\n",countnum,countnum+0.1,stat1);
    }
    printf("\n");
    for(float countnum=-35.0;countnum<10.0;countnum=countnum+0.1)
    {
        stat2=0;
        for(int i=0;i<1045;i++)
        {
            if(trainA[i][1]>countnum&&trainA[i][1]<countnum+0.1)
                stat2++;
        }
        sum[1]+=stat2;
        for(int ii=0;ii<8;ii++)
        {
            if(flag[1][ii]==1&&sum[1]>=65+1045*ii/8)
            {
                printf("A1 8个峰%f\n",countnum);
                flag[1][ii]=0;
            }
        }



        //printf("group A  dimmension 1   %f  to  %f  有%d个
\n",countnum,countnum+0.1,stat2);
    }

    printf("\n");
    for(float countnum=-35.0;countnum<6.0;countnum=countnum+0.1)
    {
        stat3=0;
        for(int i=0;i<1045;i++)
        {
            if(trainA[i][2]>countnum&&trainA[i][2]<countnum+0.1)
                stat3++;
        }

        sum[2]+=stat3;
        for(int ii=0;ii<8;ii++)
        {
            if(flag[2][ii]==1&&sum[2]>=65+1045*ii/8)
            {
                printf("A2 8个峰%f\n",countnum);
                flag[2][ii]=0;
            }
        }

        //printf("group A  dimmension 2   %f  to  %f  有%d个
\n",countnum,countnum+0.1,stat3);
    }

printf("\n");
    for(float countnum=-35.0;countnum<0.0;countnum=countnum+0.1)
    {
        stat1=0;
        for(int i=0;i<846;i++)
        {
            if(trainB[i][0]>countnum&&trainB[i][0]<countnum+0.1)
                stat1++;
        }
        sum[3]+=stat1;
        for(int ii=0;ii<8;ii++)
        {
            if(flag[3][ii]==1&&sum[3]>=53+846*ii/8)//53
            {
                printf("B0 8个峰%f\n",countnum);
                flag[3][ii]=0;
            }
        }



        //printf("group B  dimmension 0   %f  to  %f  有%d个
\n",countnum,countnum+0.1,stat1);
    }
    printf("\n");
    for(float countnum=-35.0;countnum<10.0;countnum=countnum+0.1)
    {
        stat2=0;
        for(int i=0;i<846;i++)
        {
            if(trainB[i][1]>countnum&&trainB[i][1]<countnum+0.1)
                stat2++;
        }
        sum[4]+=stat2;
        for(int ii=0;ii<8;ii++)
        {
            if(flag[4][ii]==1&&sum[4]>=53+846*ii/8)
            {
                printf("B1 8个峰%f\n",countnum);
                flag[4][ii]=0;
            }
        }

        //printf("group B  dimmension 1   %f  to  %f  有%d个
\n",countnum,countnum+0.1,stat2);
    }

    printf("\n");
    for(float countnum=-35.0;countnum<8.0;countnum=countnum+0.1)
    {
        stat3=0;
        for(int i=0;i<846;i++)
        {
            if(trainB[i][2]>countnum&&trainB[i][2]<countnum+0.1)
                stat3++;
        }

        sum[5]+=stat3;
        for(int ii=0;ii<8;ii++)
        {
            if(flag[5][ii]==1&&sum[5]>=53+846*ii/8)//846*0.135=106
            {
                printf("B2 8个峰%f\n",countnum);
                flag[5][ii]=0;
            }
        }


        //printf("group B  dimmension 2   %f  to  %f  有%d个
\n",countnum,countnum+0.5,stat3);
    }




    /*
    char pathname[356]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\train.txt";
    ifstream o_file;
    o_file.open(pathname);
    float i,j,k,l;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    o_file>>i;//从文件输入一个数值。
    cout<<i<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>j;//从文件输入一个数值。
    cout<<j<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>k;//从文件输入一个数值。
    cout<<k<<"\n"<<endl;
    */

    //o_file.seekg(1,ios::cur);

    //ifstream fin;
    //fin.get();
    //get成员函数会读取一个字符包括空格 和换行

    //o_file.seekg(4,ios::cur);//读指针位置向后4格
    //o_file>>l;//从文件输入一个数值。
    //cout<<l<<"\n"<<endl;
    //printf("sdfa%fdasfa",i);
    /*
    char q,w,e,r;
    o_file>>q;
    cout<<q<<"\n";
    o_file>>w;
    cout<<w<<"\n";
    o_file>>e;
    cout<<e<<"\n";
    o_file>>r;
    cout<<r<<"\n";
    o_file>>l;//从文件输入一个数值。
    cout<<l<<"\n"<<endl;
    o_file.close();
    */
    return 0;
}
```

实验二算法代码（934 行）

```
// exp2PB10210189.cpp : 定义控制台应用程序的入口点。
//信号统计建模  张博栋 PB10210189
```

```
#include "stdafx.h"
#include"math.h"
#include <opencv.hpp>
#include <stdlib.h>
#include"highgui.h"
#include <fstream>
#include <iostream> // not required by most systems
using namespace std;//!!!!!!!!!!!!!!!!!!!!can not use ofstream without this line
#define PI 3.1415926
int Kvalue=8;//K的值2,4,8
int best_times=0,get_accuracy_flag=0;
float best_accuracy=0;
void load_train_data(float trainA[1045][3],float trainB[846][3])
{
    char pathname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\train.txt";
    ifstream o_file;
    o_file.open(pathname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    for(Aroll=0;Aroll<1045;Aroll++)
    {
        o_file>>trainA[Aroll][0];//从文件输入一个数值。
        //cout<<trainA[Aroll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][1];//从文件输入一个数值。
        //cout<<trainA[Aroll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][2];//从文件输入一个数值。
        //cout<<trainA[Aroll][2]<<"\n"<<endl;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    for(Broll=0;Broll<845;Broll++)
    {
        o_file>>trainB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
```

```
            o_file>>trainB[Broll][2];//从文件输入一个数值。
            //cout<<trainB[Broll][2]<<"\n"<<endl;
            o_file>>a1;
            o_file>>a1;
            o_file>>a1;
            o_file>>a1;//转到下一行
        }
        o_file>>trainB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;
        o_file.close();
    }


    //判断某一组数据各维分别属于GMM的哪个峰,用最大似然估计,compare the values of w1*N(),w2*N(),w3*N(),w4*N()...and get maximum
    void get_class(float train_u[3][8],float train_matrix[3][3][8],float train_w[3][8],float train[3],int labelclass[3])
    {
        float reverse_matrix[3][8],value[3][8];//value用来得到各个值，根据最大的值来判断属于谁
        for(int i1=0;i1<Kvalue;i1++)
        {
            reverse_matrix[0][i1]=1.0/train_matrix[0][0][i1];
            reverse_matrix[1][i1]=1.0/train_matrix[1][1][i1];
            reverse_matrix[2][i1]=1.0/train_matrix[2][2][i1];
        }
        for(int dim3=0;dim3<3;dim3++)
            for(int i1=0;i1<Kvalue;i1++)
            {//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@应该仔细考虑是否加上w，我认为理论上要加，但是却使得易趋于一个峰
                //经过实验，不能加w

        //value[dim3][i1]=train_w[dim3][i1]*exp(-0.5*((train[dim3]-train_u[dim3][i1])*(train[dim3]-train_u[dim3][i1])*reverse_matrix[dim3][i1]))/(pow(2*PI,0.5)*pow(train_matrix[dim3][dim3][i1],0.5));

        value[dim3][i1]=exp(-0.5*((train[dim3]-train_u[dim3][i1])*(train[dim3]-train_u[dim3][i1])*reverse_matrix[dim3][i1]))/(pow(2*PI,0.5)*pow(train_matrix[dim3][dim3][i1],0.5))
```

```
;
        }
    //别忘了乘以w!！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
    //int q1=0,q2=0,q3=0;
    float max[3];
    for(int dim3=0;dim3<3;dim3++)
    {
        labelclass[dim3]=0;
        max[dim3]=value[dim3][0];
        for(int i1=1;i1<Kvalue;i1++)
        {
            //if(value[dim3][i1]>max[labelclass[dim3]])之前写的，写错了！！！！！！！！！！一直有错，引以为鉴！！！
            if(value[dim3][i1]>max[dim3])
            {
                labelclass[dim3]=i1;
                max[dim3]=value[dim3][i1];
            }

        }



    }
}




//函数承接初始的参数u，w，matrix，以及各数据，最后得到新的参数，以及每个数据分别属于GMM中的哪个峰
void Kmeans(float train_uA[3][8],float train_matrixA[3][3][8],float train_wA[3][8],float train_uB[3][8],float train_matrixB[3][3][8],float train_wB[3][8],float trainA[1045][3],float trainB[846][3],int labelA[1045][3],int labelB[846][3])
{
    float roll_train_uA[500][3][8],roll_train_matrixA[500][3][3][8],roll_train_uB[500][3][8],roll_train_matrixB[500][3][3][8];
    float roll_train_wA[500][3][8],roll_train_wB[500][3][8];
    int matrixclass;
    for(int ai=0;ai<3;ai++)
        for(int aj=0;aj<8;aj++)
        {
            roll_train_uA[0][ai][aj]=train_uA[ai][aj];
            roll_train_uB[0][ai][aj]=train_uB[ai][aj];//K均值算法的初始值
            roll_train_wA[0][ai][aj]=train_wA[ai][aj];
            roll_train_wB[0][ai][aj]=train_wB[ai][aj];
            for(int ak=0;ak<3;ak++)
            {
                roll_train_matrixA[0][ai][ak][aj]=train_matrixA[ai][ak][aj];
                roll_train_matrixB[0][ai][ak][aj]=train_matrixB[ai][ak][aj];
            }
        }
    for(int ai=0;ai<3;ai++)
        for(int aj=0;aj<8;aj++)
        {
            roll_train_matrixA[0][ai][ai][aj]=train_matrixA[ai][ai][aj];//对角线上的值
            roll_train_matrixB[0][ai][ai][aj]=train_matrixB[ai][ai][aj];//对角线上的值
        }//赋初值




    int i=0,j,dataclass[3];//i represent times

    int labelx,labely,labelz;
    int countnum_uA[3][8],countnum_uB[3][8];//表示每一类有几个数据
    int holddata;
    int get_terminal_i;
    while(1)
    {
        printf("第%d次迭代\n",i+1);
        //roll1[i+1][0]=0;roll1[i+1][1]=0;roll1[i+1][2]=-1;
        //roll2[i+1][0]=0;roll2[i+1][1]=0;roll2[i+1][2]=-1;
        for(int counti=0;counti<3;counti++)
            for(int countj=0;countj<8;countj++)
            {
                countnum_uA[counti][countj]=0;
                countnum_uB[counti][countj]=0;
            }
        //初始化为0

    roll_train_uA[i+1][0][0]=0.0;roll_train_uA[i+1][0][1]=0.0;roll_train_uA[i+1][0][2]=0.0;roll_train_uA[i+1][0][3]=0.0;roll_train_uA[i+1][0][4]=0.0;roll_train_uA[i+1][0][5]=0.0;roll_train_uA[i+1][0][6]=0.0;roll_train_uA[i+1][0][7]=0.0;

    roll_train_uA[i+1][1][0]=0.0;roll_train_uA[i+1][1][1]=0.0;roll_train_uA[i+1][1][2]=0.0;roll_train_uA[i+1][1][3]=0.0;roll_train_uA[i+1][1][4]=0.0;roll_train_uA[i+1][1][5]=0.0;roll_train_uA[i+1][1][6]=0.0;roll_train_uA[i+1][1][7]=0.0;
    roll_train_uA[i+1][2][0]=0.0;roll_train_uA[i+1][2][1]=0.0;roll_train_uA[i+1][2][2]=0.0;roll_train_uA[i+1][2][3]=0.0;roll_train_uA[i+1][2][4]=0.0;roll_train_uA[i+1][2][5]=0.0;roll_train_uA[i+1][2][6]=0.0;roll_train_uA[i+1][2][7]=0.0;



    roll_train_uB[i+1][0][0]=0.0;roll_train_uB[i+1][0][1]=0.0;roll_train_uB[i+1][0][2]=0.0;roll_train_uB[i+1][0][3]=0.0;roll_train_uB[i+1][0][4]=0.0;roll_train_uB[i+1][0][5]=0.0;roll_train_uB[i+1][0][6]=0.0;roll_train_uB[i+1][0][7]=0.0;

    roll_train_uB[i+1][1][0]=0.0;roll_train_uB[i+1][1][1]=0.0;roll_train_uB[i+1][1][2]=0.0;roll_train_uB[i+1][1][3]=0.0;roll_train_uB[i+1][1][4]=0.0;roll_train_uB[i+1][1][5]=0.0;roll_train_uB[i+1][1][6]=0.0;roll_train_uB[i+1][1][7]=0.0;

    roll_train_uB[i+1][2][0]=0.0;roll_train_uB[i+1][2][1]=0.0;roll_train_uB[i+1][2][2]=0.0;roll_train_uB[i+1][2][3]=0.0;roll_train_uB[i+1][2][4]=0.0;roll_train_uB[i+1][2][5]=0.0;roll_train_uB[i+1][2][6]=0.0;roll_train_uB[i+1][2][7]=0.0;


        //roll_train_wA[i+1][3][8]可以不赋初值
        //roll_train_wA[i+1][3][8]


        for(int dividei=0;dividei<3;dividei++)//matrix初始化为0
            for(int wt=0;wt<Kvalue;wt++)
            {
                roll_train_matrixA[i+1][dividei][dividei][wt]=0.0;
                roll_train_matrixB[i+1][dividei][dividei][wt]=0.0;
            }




        ////////////////////////////////////////////////////////////////

        cout<<"begin to calculate A\n";
        for(j=0;j<1045;j++)//处理不同数据
        {
            //if(get_distance(sample[j],roll1[i])<=get_distance(sample[j],roll2[i]))

    get_class(roll_train_uA[i],roll_train_matrixA[i],roll_train_wA[i],trainA[j],labelA[j]);//判断三个维度各应该属于哪一类,赋到labelA[j][3]中

            labelx=labelA[j][0];//第一维的属于哪一类峰
            labely=labelA[j][1];
            labelz=labelA[j][2];
            countnum_uA[0][labelx]++;
            countnum_uA[1][labely]++;
            countnum_uA[2][labelz]++;
            //roll1[i+1][0]+=sample[j][0];
            //roll1[i+1][1]+=sample[j][1];
            roll_train_uA[i+1][0][labelx]+=trainA[j][0];
            roll_train_uA[i+1][1][labely]+=trainA[j][1];
            roll_train_uA[i+1][2][labelz]+=trainA[j][2];


        }
        //计算新的聚合中心
        //num_of1=get_num_of1(sample);
        //num_of2=20-num_of1;
        //roll1[i+1][0]=roll1[i+1][0]/num_of1;
        //roll2[i+1][0]=roll2[i+1][0]/num_of2;
        //roll1[i+1][1]=roll1[i+1][1]/num_of1;
        //roll2[i+1][1]=roll2[i+1][1]/num_of2;//新的类中心

        cout<<"begin to get uA\n";

        for(int dividei=0;dividei<3;dividei++)//得到新的迭代的u
            for(int dividej=0;dividej<Kvalue;dividej++)
            {
                if(countnum_uA[dividei][dividej]!=0)

    roll_train_uA[i+1][dividei][dividej]=roll_train_uA[i+1][dividei][dividej]/countnum_uA[dividei][dividej];
                else {

    roll_train_uA[i+1][dividei][dividej]=roll_train_uA[i][dividei][dividej];
                    printf("没被分到类!\n");}
            }
        for(int hhhi=0;hhhi<1045;hhhi++)//得到矩阵matrix
            for(int dividei=0;dividei<3;dividei++)//得到新的迭代的matrix
                {

    matrixclass=labelA[hhhi][dividei];//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

    roll_train_matrixA[i+1][dividei][dividei][matrixclass]+=(trainA[hhhi][dividei]-roll_train_uA[i+1][dividei][matrixclass])*(trainA[hhhi][dividei]-roll_train_uA[i+1][dividei
][matrixclass]);
                }

        for(int dim3=0;dim3<3;dim3++)//得到矩阵matrix
            for(int divsummat=0;divsummat<Kvalue;divsummat++)
            {
                holddata=countnum_uA[dim3][divsummat];
                if(holddata>0)

    roll_train_matrixA[i+1][dim3][dim3][divsummat]=roll_train_matrixA[i+1][dim3][dim3][
divsummat]/holddata;
            }



        for(int dim3=0;dim3<3;dim3++)//得到w
            for(int divsummat=0;divsummat<Kvalue;divsummat++)
            {
                holddata=countnum_uA[dim3][divsummat];
                if(holddata>0)
                    roll_train_wA[i+1][dim3][divsummat]=holddata/1045.0;
                else roll_train_wA[i+1][dim3][divsummat]=0.0;

            }


        ////////////////////////////////////////中





        for(j=0;j<846;j++)//处理不同数据
        {
            //if(get_distance(sample[j],roll1[i])<=get_distance(sample[j],roll2[i]))

    get_class(roll_train_uB[i],roll_train_matrixB[i],roll_train_wB[i],trainB[j],labelB[
j]);//判断三个维度各应该属于哪一类,赋到labelB[j][3]中

            labelx=labelB[j][0];//第一维的属于哪一类峰
            labely=labelB[j][1];
            labelz=labelB[j][2];
            countnum_uB[0][labelx]++;
            countnum_uB[1][labely]++;
            countnum_uB[2][labelz]++;
            //roll1[i+1][0]+=sample[j][0];
            //roll1[i+1][1]+=sample[j][1];
            roll_train_uB[i+1][0][labelx]+=trainB[j][0];
            roll_train_uB[i+1][1][labely]+=trainB[j][1];
            roll_train_uB[i+1][2][labelz]+=trainB[j][2];


        }
        //计算新的聚合中心
        //num_of1=get_num_of1(sample);
        //num_of2=20-num_of1;
        //roll1[i+1][0]=roll1[i+1][0]/num_of1;
        //roll2[i+1][0]=roll2[i+1][0]/num_of2;
        //roll1[i+1][1]=roll1[i+1][1]/num_of1;
        //roll2[i+1][1]=roll2[i+1][1]/num_of2;//新的类中心

        for(int dividei=0;dividei<3;dividei++)//得到新的迭代的u
            for(int dividej=0;dividej<Kvalue;dividej++)
            {
                if(countnum_uB[dividei][dividej]!=0)

    roll_train_uB[i+1][dividei][dividej]=roll_train_uB[i+1][dividei][dividej]/countnum_uB[dividei][dividej];
                else {
                    printf("wrong!\n");
                    printf("wrong!\n");}
            }
        for(int hhhi=0;hhhi<846;hhhi++)//得到矩阵matrix
            for(int dividei=0;dividei<3;dividei++)//得到新的迭代的matrix
                {
                    matrixclass=labelB[hhhi][dividei];

    roll_train_matrixB[i+1][dividei][dividei][matrixclass]+=(trainB[hhhi][dividei]-roll_train_uB[i+1][dividei][matrixclass])*(trainB[hhhi][dividei]-roll_train_uB[i+1][dividei][matrixclass]);
                }
            //w,   class B
        for(int dim3=0;dim3<3;dim3++)//得到矩阵matrix
            for(int divsummat=0;divsummat<Kvalue;divsummat++)
            {
                holddata=countnum_uB[dim3][divsummat];
                if(holddata>0)
    roll_train_matrixB[i+1][dim3][dim3][divsummat]=roll_train_matrixB[i+1][dim3][dim3][
divsummat]/holddata;
            }



        for(int dim3=0;dim3<3;dim3++)//得到w
            for(int divsummat=0;divsummat<Kvalue;divsummat++)
            {
                holddata=countnum_uB[dim3][divsummat];
                roll_train_wB[i+1][dim3][divsummat]=holddata/846.0;
            }



/////////////////////////////////////////////////////////////////////////////////
















        //printf("类中心，分别为\n(%f,%f),
(%f,%f)\n",roll1[i+1][0],roll1[i+1][1],roll2[i+1][0],roll2[i+1][1]);
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
            {
                printf("A第%d维，第%d个峰，类中心，u为%f
",printcenteri,printcenterj,roll_train_uA[i+1][printcenteri][printcenterj]);
                printf("矩阵为%f
",roll_train_matrixA[i+1][printcenteri][printcenteri][printcenterj]);
                printf("w为%f\n",roll_train_wA[i+1][printcenteri][printcenterj]);
            }
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
                {
                    printf("B第%d维，第%d个峰，类中心，u为%f ",printcenteri,printcenterj,roll_train_uB[i+1][printcenteri][printcenterj]);
                    printf("矩阵为%f ",roll_train_matrixB[i+1][printcenteri][printcenteri][printcenterj]);
                    printf("w为%f\n",roll_train_wB[i+1][printcenteri][printcenterj]);

                }



        int whetherbreak=1;
        //float roll_train_uA[500][3][8],roll_train_matrixA[500][3][3][8],roll_train_uB[500][3][8],roll_train_matrixB[500][3][3][8];
        //float roll_train_wA[500][3][8],roll_train_wB[500][3][8];

        for(int printcenteri=0;printcenteri<3;printcenteri++)
            for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
            {

    if(roll_train_uA[i][printcenteri][printcenterj]!=roll_train_uA[i+1][printcenteri][printcenterj])
                {
                    whetherbreak=0;
                }

    if(roll_train_matrixA[i][printcenteri][printcenteri][printcenterj]!=roll_train_matrixA[i+1][printcenteri][printcenteri][printcenterj])
                {
                    whetherbreak=0;
                }

    if(roll_train_uB[i][printcenteri][printcenterj]!=roll_train_uB[i+1][printcenteri][printcenterj])
                {
                    whetherbreak=0;
                }

    if(roll_train_matrixB[i][printcenteri][printcenteri][printcenterj]!=roll_train_matrixB[i+1][printcenteri][printcenteri][printcenterj])
                {
                    whetherbreak=0;
                }

    if(roll_train_wA[i][printcenteri][printcenterj]!=roll_train_wA[i+1][printcenteri][p
rintcenterj])
                {
                    whetherbreak=0;
                }

    if(roll_train_wB[i][printcenteri][printcenterj]!=roll_train_wB[i+1][printcenteri][p
rintcenterj])
                {
                    whetherbreak=0;
                }
            }

        if(whetherbreak==1)
        {
            printf("已经找到聚类中心\n");
            get_terminal_i=i+1;
            break;
        }
        if(i>=40)
        {
            printf("迭代次数达到500次，停止迭代\n");
            for(int printcenteri=0;printcenteri<3;printcenteri++)
                for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
                    printf("A第%d维，第%d个峰，类中心，u
为%f\n",printcenteri,printcenterj,roll_train_uA[i+1][printcenteri][printcenterj]);

            for(int printcenteri=0;printcenteri<3;printcenteri++)
                for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
                    printf("B第%d维，第%d个峰，类中心，u
为%f\n",printcenteri,printcenterj,roll_train_uB[i+1][printcenteri][printcenterj]);
            get_terminal_i=i+1;
            break;
        }
        i++;
    }

    for(int ai=0;ai<3;ai++)
        for(int aj=0;aj<8;aj++)
        {
            train_uA[ai][aj]=roll_train_uA[get_terminal_i][ai][aj];
            train_uB[ai][aj]=roll_train_uB[get_terminal_i][ai][aj];//K均值算法的初始值
            train_wA[ai][aj]=roll_train_wA[get_terminal_i][ai][aj];
            train_wB[ai][aj]=roll_train_wB[get_terminal_i][ai][aj];
            for(int ak=0;ak<3;ak++)
            {

    train_matrixA[ai][ak][aj]=roll_train_matrixA[get_terminal_i][ai][ak][aj];

    train_matrixB[ai][ak][aj]=roll_train_matrixB[get_terminal_i][ai][ak][aj];
            }
        }




}

void judge(float train_uA[3][8],float train_matrixA[3][3][8],float train_wA[3][8],float
train_uB[3][8],float train_matrixB[3][3][8],float train_wB[3][8]);

float p_in_EM(int k,float X,float train_u[8],float matrix[8],float train_w[8])
{
    float sum=0.0,numerator=0.0;
    for(int ii=0;ii<Kvalue;ii++)

    sum+=train_w[ii]*exp(-0.5*(X-train_u[ii])*(X-train_u[ii])/matrix[ii])/(pow(2*PI,0.5
)*matrix[ii]);
    numerator=train_w[k]*exp(-0.5*(X-train_u[k])*(X-train_u[k])/matrix[k])/(pow(2*PI,0.
5)*matrix[k]);
    numerator=numerator/sum;
    //cout<<"p  "<<k<<" 的值为"<<numerator<<"\n";
    return numerator;
}


void EM_algorithm(float train_uA[3][8],float train_matrixA[3][3][8],float
train_wA[3][8],float train_uB[3][8],float train_matrixB[3][3][8],float
train_wB[3][8],float trainA[1045][3],float trainB[846][3])
{
    int get_terminal_i;
    float
roll_train_uA[500][3][8],roll_train_matrixA[500][3][3][8],roll_train_uB[500][3][8],roll
_train_matrixB[500][3][3][8];
    float roll_train_wA[500][3][8],roll_train_wB[500][3][8];
```

```
int matrixclass;
for(int ai=0;ai<3;ai++)
    for(int aj=0;aj<8;aj++)
    {
        roll_train_uA[0][ai][aj]=train_uA[ai][aj];
        roll_train_uB[0][ai][aj]=train_uB[ai][aj];//K均值算法的初始值
        roll_train_wA[0][ai][aj]=train_wA[ai][aj];
        roll_train_wB[0][ai][aj]=train_wB[ai][aj];
        for(int ak=0;ak<3;ak++)
        {
            roll_train_matrixA[0][ai][ak][aj]=train_matrixA[ai][ak][aj];
            roll_train_matrixB[0][ai][ak][aj]=train_matrixB[ai][ak][aj];
        }
    }
//cout<<roll_train_wB[0][1][0]<<"\n";
//cout<<roll_train_uB[0][1][0]<<"\n";
//cout<<roll_train_matrixA[0][2][2][0]<<"\n";
for(int ai=0;ai<3;ai++)
    for(int aj=0;aj<8;aj++)
    {
        roll_train_matrixA[0][ai][ai][aj]=train_matrixA[ai][ai][aj];//对角线上的值
        roll_train_matrixB[0][ai][ai][aj]=train_matrixB[ai][ai][aj];//对角线上的值
    }//赋初值




int i=0,j,dataclass[3];//i represent times

int labelx,labely,labelz;
int holddata;

while(1)
{
    printf("第%d次迭代\n",i+1);
    //roll1[i+1][0]=0;roll1[i+1][1]=0;roll1[i+1][2]=-1;
    //roll2[i+1][0]=0;roll2[i+1][1]=0;roll2[i+1][2]=-1;

    //初始化为0


    //以下

roll_train_uA[i+1][0][0]=0.0;roll_train_uA[i+1][0][1]=0.0;roll_train_uA[i+1][0][2]=
0.0;roll_train_uA[i+1][0][3]=0.0;roll_train_uA[i+1][0][4]=0.0;roll_train_uA[i+1][0][5]=
0.0;roll_train_uA[i+1][0][6]=0.0;roll_train_uA[i+1][0][7]=0.0;

    roll_train_uA[i+1][1][0]=0.0;roll_train_uA[i+1][1][1]=0.0;roll_train_uA[i+1][1][2]=
0.0;roll_train_uA[i+1][1][3]=0.0;roll_train_uA[i+1][1][4]=0.0;roll_train_uA[i+1][1][5]=
0.0;roll_train_uA[i+1][1][6]=0.0;roll_train_uA[i+1][1][7]=0.0;

    roll_train_uA[i+1][2][0]=0.0;roll_train_uA[i+1][2][1]=0.0;roll_train_uA[i+1][2][2]=
0.0;roll_train_uA[i+1][2][3]=0.0;roll_train_uA[i+1][2][4]=0.0;roll_train_uA[i+1][2][5]=
0.0;roll_train_uA[i+1][2][6]=0.0;roll_train_uA[i+1][2][7]=0.0;



    roll_train_uB[i+1][0][0]=0.0;roll_train_uB[i+1][0][1]=0.0;roll_train_uB[i+1][0][2]=
0.0;roll_train_uB[i+1][0][3]=0.0;roll_train_uB[i+1][0][4]=0.0;roll_train_uB[i+1][0][5]=
0.0;roll_train_uB[i+1][0][6]=0.0;roll_train_uB[i+1][0][7]=0.0;

    roll_train_uB[i+1][1][0]=0.0;roll_train_uB[i+1][1][1]=0.0;roll_train_uB[i+1][1][2]=
0.0;roll_train_uB[i+1][1][3]=0.0;roll_train_uB[i+1][1][4]=0.0;roll_train_uB[i+1][1][5]=
0.0;roll_train_uB[i+1][1][6]=0.0;roll_train_uB[i+1][1][7]=0.0;

    roll_train_uB[i+1][2][0]=0.0;roll_train_uB[i+1][2][1]=0.0;roll_train_uB[i+1][2][2]=
0.0;roll_train_uB[i+1][2][3]=0.0;roll_train_uB[i+1][2][4]=0.0;roll_train_uB[i+1][2][5]=
0.0;roll_train_uB[i+1][2][6]=0.0;roll_train_uB[i+1][2][7]=0.0;



        for(int dividei=0;dividei<3;dividei++)//matrix初始化为0
            for(int wt=0;wt<Kvalue;wt++)
            {
                roll_train_matrixA[i+1][dividei][dividei][wt]=0.0;
                roll_train_matrixB[i+1][dividei][dividei][wt]=0.0;
            }

        //以上为初始化


        ///////////////////////////////////////////////////////////

        cout<<"begin to calculate A\n";


        float sum_p_of_kA[500][3][8],sum_p_of_kB[500][3][8];//算u，matrix，w都需要用到
```

```
的这个数据,先算好
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                sum_p_of_kA[i][first3][firstk]=0.0;
                sum_p_of_kB[i][first3][firstk]=0.0;//全代码都要注意是i还是i+1(要算的是i，已知的是i+1)@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

            }

        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                for(int cali=0;cali<1045;cali++)

    sum_p_of_kA[i][first3][firstk]+=p_in_EM(firstk,trainA[cali][first3],roll_train_uA[i][first3],roll_train_matrixA[i][first3][first3],roll_train_wA[i][first3]);

    cout<<"sum_p_of_kA["<<i<<"]["<<first3<<"]["<<firstk<<"]="<<sum_p_of_kA[i][first3][firstk]<<"\n";
            }
        //下面开始求A的下一次迭代的u
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                for(int cali=0;cali<1045;cali++)
                {

    roll_train_uA[i+1][first3][firstk]+=trainA[cali][first3]*p_in_EM(firstk,trainA[cali][first3],roll_train_uA[i][first3],roll_train_matrixA[i][first3][first3],roll_train_wA[i][first3]);

                }

    roll_train_uA[i+1][first3][firstk]=roll_train_uA[i+1][first3][firstk]/sum_p_of_kA[i][first3][firstk];
            }



        //下面开始求matrix
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                for(int cali=0;cali<1045;cali++)
```

```
                    {
                        roll_train_matrixA[i+1][first3][first3][firstk]+=(trainA[cali][first3]-roll_train_uA[i][first3][firstk])*(trainA[cali][first3]-roll_train_uA[i][first3][firstk])*p_in_EM(firstk,trainA[cali][first3],roll_train_uA[i][first3],roll_train_matrixA[i][first3][first3],roll_train_wA[i][first3]);
                        //按定义
                    }
                roll_train_matrixA[i+1][first3][first3][firstk]=roll_train_matrixA[i+1][first3][first3][firstk]/sum_p_of_kA[i][first3][firstk];
                }

        //下面开始求w
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                roll_train_wA[i+1][first3][firstk]=sum_p_of_kA[i][first3][firstk]/1045.0;

            }



    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////





            cout<<"begin to calculate B\n";

        /*
        float sum_p_of_kA[500][3][8],sum_p_of_kB[500][3][8];//算u，matrix，w都需要用到的这个数据,先算好
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                sum_p_of_kA[i][3][8]=0;
                sum_p_of_kB[i][3][8]=0;//全代码都要注意是i还是i+1(要算的是i，已知的是i+1)@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            }
        */
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
                for(int cali=0;cali<846;cali++)

    sum_p_of_kB[i][first3][firstk]+=p_in_EM(firstk,trainB[cali][first3],roll_train_uB[i][first3],roll_train_matrixB[i][first3][first3],roll_train_wB[i][first3]);

        //下面开始求B的下一次迭代的u
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                for(int cali=0;cali<846;cali++)
                {

    roll_train_uB[i+1][first3][firstk]+=trainB[cali][first3]*p_in_EM(firstk,trainB[cali][first3],roll_train_uB[i][first3],roll_train_matrixB[i][first3][first3],roll_train_wB[i][first3]);

                }

    roll_train_uB[i+1][first3][firstk]=roll_train_uB[i+1][first3][firstk]/sum_p_of_kB[i][first3][firstk];
            }


        //下面开始求B matrix
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {
                for(int cali=0;cali<846;cali++)
                {

    roll_train_matrixB[i+1][first3][first3][firstk]+=(trainB[cali][first3]-roll_train_uB[i][first3][firstk])*(trainB[cali][first3]-roll_train_uB[i][first3][firstk])*p_in_EM(firstk,trainB[cali][first3],roll_train_uB[i][first3],roll_train_matrixB[i][first3][first3],roll_train_wB[i][first3]);
                        //按定义
                }

    roll_train_matrixB[i+1][first3][first3][firstk]=roll_train_matrixB[i+1][first3][first3][firstk]/sum_p_of_kB[i][first3][firstk];
            }

        //下面开始求B w
        for(int first3=0;first3<3;first3++)
            for(int firstk=0;firstk<Kvalue;firstk++)
            {

    roll_train_wB[i+1][first3][firstk]=sum_p_of_kB[i][first3][firstk]/846.0;

            }




    ///////****************************************************************************
******************************

        //printf("类中心，分别为\n(%f,%f),
(%f,%f)\n",roll1[i+1][0],roll1[i+1][1],roll2[i+1][0],roll2[i+1][1]);
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
            {
                printf("EM A第%d维，第%d个峰，u为%f
",printcenteri,printcenterj,roll_train_uA[i+1][printcenteri][printcenterj]);
                printf("矩阵为%f
",roll_train_matrixA[i+1][printcenteri][printcenteri][printcenterj]);
                printf("w为%f\n",roll_train_wA[i+1][printcenteri][printcenterj]);

            }
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
                {
                    printf("EM B第%d维，第%d个峰，u为%f
",printcenteri,printcenterj,roll_train_uB[i+1][printcenteri][printcenterj]);
                    printf("矩阵为%f
",roll_train_matrixB[i+1][printcenteri][printcenteri][printcenterj]);
                    printf("w为%f\n",roll_train_wB[i+1][printcenteri][printcenterj]);

                }

        get_accuracy_flag=0;
        //test accuracy

    //get_accuracy=judge(roll_train_uA[i+1],roll_train_matrixA[i+1],roll_train_wA[i+1],
roll_train_uB[i+1],roll_train_matrixB[i+1],roll_train_wB[i+1]);

    judge(roll_train_uA[i+1],roll_train_matrixA[i+1],roll_train_wA[i+1],roll_train_uB[i
+1],roll_train_matrixB[i+1],roll_train_wB[i+1]);
      if(get_accuracy_flag==1)
      {
         best_times=i+1;
      }
      get_accuracy_flag=0;






      int whetherbreak=1;
      //float
roll_train_uA[500][3][8],roll_train_matrixA[500][3][3][8],roll_train_uB[500][3][8],roll
_train_matrixB[500][3][3][8];
      //float roll_train_wA[500][3][8],roll_train_wB[500][3][8];

      for(int printcenteri=0;printcenteri<3;printcenteri++)
         for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
         {

   if(roll_train_uA[i][printcenteri][printcenterj]!=roll_train_uA[i+1][printcenteri][p
rintcenterj])
            {
               whetherbreak=0;
            }

   if(roll_train_matrixA[i][printcenteri][printcenteri][printcenterj]!=roll_train_matr
ixA[i+1][printcenteri][printcenteri][printcenterj])
            {
               whetherbreak=0;
            }

   if(roll_train_uB[i][printcenteri][printcenterj]!=roll_train_uB[i+1][printcenteri][p
rintcenterj])
            {
               whetherbreak=0;
            }

   if(roll_train_matrixB[i][printcenteri][printcenteri][printcenterj]!=roll_train_matr
ixB[i+1][printcenteri][printcenteri][printcenterj])
            {
               whetherbreak=0;
            }
```

```
    if(roll_train_wA[i][printcenteri][printcenterj]!=roll_train_wA[i+1][printcenteri][p
rintcenterj])
                {
                    whetherbreak=0;
                }

    if(roll_train_wB[i][printcenteri][printcenterj]!=roll_train_wB[i+1][printcenteri][p
rintcenterj])
                {
                    whetherbreak=0;
                }
            }

        if(whetherbreak==1)
        {
            printf("已经找到类中心\n");
            get_terminal_i=i+1;
            break;
        }
        if(i>=11)
        {
            printf("迭代次数达到限制，停止迭代\n");
            /*
            for(int printcenteri=0;printcenteri<3;printcenteri++)
                for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
                    printf("EM A第%d维，第%d个峰，类中心，u
为%f\n",printcenteri,printcenterj,roll_train_uA[i+1][printcenteri][printcenterj]);

            for(int printcenteri=0;printcenteri<3;printcenteri++)
                for(int printcenterj=0;printcenterj<Kvalue;printcenterj++)
                    printf("EM B第%d维，第%d个峰，类中心，u
为%f\n",printcenteri,printcenterj,roll_train_uB[i+1][printcenteri][printcenterj]);

            */
            get_terminal_i=i+1;
            break;
        }
        i++;
    }

    for(int ai=0;ai<3;ai++)
        for(int aj=0;aj<8;aj++)
        {
            train_uA[ai][aj]=roll_train_uA[get_terminal_i][ai][aj];
            train_uB[ai][aj]=roll_train_uB[get_terminal_i][ai][aj];//K均值算法的初始值
            train_wA[ai][aj]=roll_train_wA[get_terminal_i][ai][aj];
            train_wB[ai][aj]=roll_train_wB[get_terminal_i][ai][aj];
            for(int ak=0;ak<3;ak++)
            {

    train_matrixA[ai][ak][aj]=roll_train_matrixA[get_terminal_i][ai][ak][aj];

    train_matrixB[ai][ak][aj]=roll_train_matrixB[get_terminal_i][ai][ak][aj];
            }
        }




}

//float train_uA[3][8],float train_matrixA[3][3][8],float train_wA[3][8],float
train_uB[3][8],float train_matrixB[3][3][8],float train_wB[3][8],float
trainA[1045][3],float trainB[846][3]

//void judge(float train_uA[3],float train_matrixA[3][3],float train_uB[3],float
train_matrixB[3][3])//判断某个数据属于哪一类
void judge(float train_uA[3][8],float train_matrixA[3][3][8],float train_wA[3][8],float
train_uB[3][8],float train_matrixB[3][3][8],float train_wB[3][8])
{
    float testA[470][3],testB[360][3];
    float right=0.0;float sum=0.0;
    char pathtestname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\test.txt";
    ifstream o_file;
    o_file.open(pathtestname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    float
Areverse_matrix[3][8],Breverse_matrix[3][8],valueA[3],valueB[3],total_valueA,total_valu
eB;
    for(int accuracyt=0;accuracyt<Kvalue;accuracyt++)
    {
    Areverse_matrix[0][accuracyt]=1.0/train_matrixA[0][0][accuracyt];
    Areverse_matrix[1][accuracyt]=1.0/train_matrixA[1][1][accuracyt];
    Areverse_matrix[2][accuracyt]=1.0/train_matrixA[2][2][accuracyt];
    Breverse_matrix[0][accuracyt]=1.0/train_matrixB[0][0][accuracyt];
```

```
Breverse_matrix[1][accuracyt]=1.0/train_matrixB[1][1][accuracyt];
Breverse_matrix[2][accuracyt]=1.0/train_matrixB[2][2][accuracyt];
}
for(Aroll=0;Aroll<470;Aroll++)
{
    o_file>>testA[Aroll][0];//从文件输入一个数值。
    //cout<<trainA[Aroll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testA[Aroll][1];//从文件输入一个数值。
    //cout<<trainA[Aroll][1]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testA[Aroll][2];//从文件输入一个数值。
    //cout<<trainA[Aroll][2]<<"\n"<<endl;
    valueA[0]=0.0;valueA[1]=0.0;valueA[2]=0.0;
    valueB[0]=0.0;valueB[1]=0.0;valueB[2]=0.0;
    for(int accdim3=0;accdim3<3;accdim3++)
        for(int accuracyk=0;accuracyk<Kvalue;accuracyk++)
        {

valueA[accdim3]+=train_wA[accdim3][accuracyk]*exp(-0.5*((testA[Aroll][accdim3]-train_uA[accdim3][accuracyk])*(testA[Aroll][accdim3]-train_uA[accdim3][accuracyk])*Areverse_matrix[accdim3][accuracyk]))/(pow(2*PI*train_matrixA[accdim3][accdim3][accuracyk],0.5));

valueB[accdim3]+=train_wB[accdim3][accuracyk]*exp(-0.5*((testA[Aroll][accdim3]-train_uB[accdim3][accuracyk])*(testA[Aroll][accdim3]-train_uB[accdim3][accuracyk])*Breverse_matrix[accdim3][accuracyk]))/(pow(2*PI*train_matrixB[accdim3][accdim3][accuracyk],0.5));
        }
    total_valueA=valueA[0]*valueA[1]*valueA[2];
    total_valueB=valueB[0]*valueB[1]*valueB[2];
    sum=sum+1.0;
    if(total_valueA>total_valueB)
        right=right+1.0;
    o_file>>a1;
    o_file>>a1;
    o_file>>a1;
    o_file>>a1;//转到下一行
}
for(Broll=0;Broll<359;Broll++)
{
    o_file>>testB[Broll][0];//从文件输入一个数值。
    //cout<<trainB[Broll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
```

```
        o_file>>testB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;



        valueA[0]=0.0;valueA[1]=0.0;valueA[2]=0.0;
        valueB[0]=0.0;valueB[1]=0.0;valueB[2]=0.0;
        for(int accdim3=0;accdim3<3;accdim3++)
            for(int accuracyk=0;accuracyk<Kvalue;accuracyk++)
            {
    valueA[accdim3]+=train_wA[accdim3][accuracyk]*exp(-0.5*((testB[Broll][accdim3]-train_uA[accdim3][accuracyk])*(testB[Broll][accdim3]-train_uA[accdim3][accuracyk])*Areverse_matrix[accdim3][accuracyk]))/(pow(2*PI*train_matrixA[accdim3][accdim3][accuracyk],0.5));

    valueB[accdim3]+=train_wB[accdim3][accuracyk]*exp(-0.5*((testB[Broll][accdim3]-train_uB[accdim3][accuracyk])*(testB[Broll][accdim3]-train_uB[accdim3][accuracyk])*Breverse_matrix[accdim3][accuracyk]))/(pow(2*PI*train_matrixB[accdim3][accdim3][accuracyk],0.5));
            }
        total_valueA=valueA[0]*valueA[1]*valueA[2];
        total_valueB=valueB[0]*valueB[1]*valueB[2];
        sum=sum+1.0;
        if(total_valueA<=total_valueB)
            right=right+1.0;

        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    o_file>>testB[Broll][0];//从文件输入一个数值。
    //cout<<trainB[Broll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testB[Broll][1];//从文件输入一个数值。
    //cout<<trainB[Broll][1]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testB[Broll][2];//从文件输入一个数值。
    //cout<<trainB[Broll][2]<<"\n"<<endl;
    valueA[0]=0.0;valueA[1]=0.0;valueA[2]=0.0;
    valueB[0]=0.0;valueB[1]=0.0;valueB[2]=0.0;
    for(int accdim3=0;accdim3<3;accdim3++)
        for(int accuracyk=0;accuracyk<Kvalue;accuracyk++)
        {

    valueA[accdim3]+=train_wA[accdim3][accuracyk]*exp(-0.5*((testB[Broll][accdim3]-train_uA[accdim3][accuracyk])*(testB[Broll][accdim3]-train_uA[accdim3][accuracyk])*Areverse_matrix[accdim3][accuracyk]))/(pow(2*PI*train_matrixA[accdim3][accdim3][accuracyk],0.5));

    valueB[accdim3]+=train_wB[accdim3][accuracyk]*exp(-0.5*((testB[Broll][accdim3]-train_uB[accdim3][accuracyk])*(testB[Broll][accdim3]-train_uB[accdim3][accuracyk])*Breverse_matrix[accdim3][accuracyk]))/(pow(2*PI*train_matrixB[accdim3][accdim3][accuracyk],0.5));
        }
    total_valueA=valueA[0]*valueA[1]*valueA[2];
    total_valueB=valueB[0]*valueB[1]*valueB[2];
    sum=sum+1.0;
    if(total_valueA<=total_valueB)
        right=right+1.0;


    float accuracy=right/sum;
    if(accuracy>best_accuracy)
    {
        get_accuracy_flag=1;
        best_accuracy=accuracy;
    }


    cout<<"用测试数据进行检测\n共有"<<sum<<"个，有"<<right<<"个正确\n精确度为"<<accuracy<<"\n";
    o_file.close();
}


int _tmain(int argc, _TCHAR* argv[])
{
    //先K均值算法，得到初始值
    float trainA[1045][3],trainB[846][3];//A有1045行数据，B有846行数据
    load_train_data(trainA,trainB);//把数据读到数组里
    float train_uA[3][8],train_matrixA[3][3][8],train_uB[3][8],train_matrixB[3][3][8],train_wA[3][8],train_wB[3][8];
    for(int i=0;i<3;i++)
        for(int j=0;j<8;j++)
        {
            train_uA[i][j]=-10.0+i;
            train_uB[i][j]=-13.0+i;//K均值算法的初始值
            train_wA[i][j]=1.0/Kvalue;
            train_wB[i][j]=1.0/Kvalue;
            for(int k=0;k<3;k++)
            {
                train_matrixA[i][k][j]=0.0;
                train_matrixB[i][k][j]=0.0;
            }
        }

    /*
    // 2个 峰时用
    train_uA[0][0]=-20.0;train_uA[0][1]=0.0;train_uA[0][2]=-18.0;train_uA[0][3]=-2.0;train_uA[0][4]=-16.0;train_uA[0][5]=-4.0;train_uA[0][6]=-14.0;train_uA[0][7]=-6.0;
    train_uA[1][0]=-4.0;train_uA[1][1]=-3.0;train_uA[1][2]=-5.0;train_uA[1][3]=-2.0;train_uA[1][4]=-6.0;train_uA[1][5]=-1.0;train_uA[1][6]=-7.0;train_uA[1][7]=0.0;
    train_uA[2][0]=-9.0;train_uA[2][1]=-8.0;train_uA[2][2]=-7.0;train_uA[2][3]=-10.0;train_uA[2][4]=-11.0;train_uA[2][5]=-6.0;train_uA[2][6]=-12.0;train_uA[2][7]=-5.0;


    train_uB[0][0]=-23.0;train_uB[0][1]=-3.0;train_uB[0][2]=-21.0;train_uB[0][3]=-5.0;train_uB[0][4]=-19.0;train_uB[0][5]=-7.0;train_uB[0][6]=-17.0;train_uB[0][7]=-9.0;
    train_uB[1][0]=-4.0;train_uB[1][1]=-3.0;train_uB[1][2]=-5.0;train_uB[1][3]=-2.0;train_uB[1][4]=-6.0;train_uB[1][5]=-1.0;train_uB[1][6]=-7.0;train_uB[1][7]=0.0;
    train_uB[2][0]=-5.0;train_uB[2][1]=-4.0;train_uB[2][2]=-3.0;train_uB[2][3]=-6.0;train_uB[2][4]=-7.0;train_uB[2][5]=-2.0;train_uB[2][6]=-8.0;train_uB[2][7]=-1.0;
    */
    /*

    //4,2个峰时用

    train_uA[0][0]=-13.5;train_uA[0][1]=-7.5;train_uA[0][2]=-10.0;train_uA[0][3]=-11.5;train_uA[0][4]=-16.0;train_uA[0][5]=-4.0;train_uA[0][6]=-14.0;train_uA[0][7]=-6.0;
    train_uA[1][0]=-10.5;train_uA[1][1]=0.5;train_uA[1][2]=-4.0;train_uA[1][3]=-2.0;train_uA[1][4]=-6.0;train_uA[1][5]=-1.0;train_uA[1][6]=-7.0;train_uA[1][7]=0.0;
    train_uA[2][0]=-16.5;train_uA[2][1]=-2.5;train_uA[2][2]=-10.0;train_uA[2][3]=-5.0;train_uA[2][4]=-11.0;train_uA[2][5]=-6.0;train_uA[2][6]=-12.0;train_uA[2][7]=-5.0;

    train_uB[0][0]=-18.0;train_uB[0][1]=-9.0;train_uB[0][2]=-15.0;train_uB[0][3]=-12.5;train_uB[0][4]=-19.0;train_uB[0][5]=-7.0;train_uB[0][6]=-17.0;train_uB[0][7]=-9.0;
    train_uB[1][0]=-14.0;train_uB[1][1]=1.5;train_uB[1][2]=-5.0;train_uB[1][3]=-1.0;train_uB[1][4]=-6.0;train_uB[1][5]=-1.0;train_uB[1][6]=-7.0;train_uB[1][7]=0.0;
    train_uB[2][0]=-12.0;train_uB[2][1]=2.0;train_uB[2][2]=-6.0;train_uB[2][3]=-1.5;train_uB[2][4]=-7.0;train_uB[2][5]=-2.0;train_uB[2][6]=-8.0;train_uB[2][7]=-1.0;



    */


    ///*

    //8个峰时用

    train_uA[0][0]=-15.1;train_uA[0][1]=-6.3;train_uA[0][2]=-12.5;train_uA[0][3]=-8.4;train_uA[0][4]=-11.5;train_uA[0][5]=-9.7;train_uA[0][6]=-10.9;train_uA[0][7]=-10.3;
    train_uA[1][0]=-14.3;train_uA[1][1]=2.4;train_uA[1][2]=-6.4;train_uA[1][3]=-0.4;train_uA[1][4]=-4.1;train_uA[1][5]=-1.4;train_uA[1][6]=-3.2;train_uA[1][7]=-2.3;
    train_uA[2][0]=-18.9;train_uA[2][1]=-1.2;train_uA[2][2]=-14.2;train_uA[2][3]=-2.9;train_uA[2][4]=-11.2;train_uA[2][5]=-4.1;train_uA[2][6]=-8.3;train_uA[2][7]=-6.0;

    train_uB[0][0]=-19.4;train_uB[0][1]=-7.2;train_uB[0][2]=-17.0;train_uB[0][3]=-10.3;train_uB[0][4]=-15.6;train_uB[0][5]=-11.7;train_uB[0][6]=-14.4;train_uB[0][7]=-13.2;
    train_uB[1][0]=-16.6;train_uB[1][1]=3.0;train_uB[1][2]=-11.2;train_uB[1][3]=-0.9;train_uB[1][4]=-6.5;train_uB[1][5]=-0.3;train_uB[1][6]=-3.4;train_uB[1][7]=1.7;
    train_uB[2][0]=-17.4;train_uB[2][1]=3.8;train_uB[2][2]=-9.5;train_uB[2][3]=1.2;train_uB[2][4]=-6.7;train_uB[2][5]=-0.6;train_uB[2][6]=-4.4;train_uB[2][7]=-2.5;



    //*/



    for(int i=0;i<3;i++)
        for(int j=0;j<8;j++)
        {
            train_matrixA[i][i][j]=10.0;//对角线上的值,原先写的是60
            train_matrixB[i][i][j]=10.0;//对角线上的值
        }
    int labelA[1045][3], labelB[846][3];
    Kmeans(train_uA,train_matrixA,train_wA,train_uB,train_matrixB,train_wB,trainA,trainB,labelA,labelB);
    printf("Kmeans算法初始化后\n");
    judge(train_uA,train_matrixA,train_wA,train_uB,train_matrixB,train_wB);
    EM_algorithm(train_uA,train_matrixA,train_wA,train_uB,train_matrixB,train_wB,trainA
,trainB);
    printf("EM算法后\n");
    judge(train_uA,train_matrixA,train_wA,train_uB,train_matrixB,train_wB);

    cout<<"第"<<best_times<<"次迭代得到的精度最高，为"<<best_accuracy;

    return 0;
}
```

实验三代码（589 行）

```
// Project_PB10210189.cpp : 定义控制台应用程序的入口点。
//writer: Bodong Zhang

#include "stdafx.h"
#include"math.h"
#include <opencv.hpp>
#include <stdlib.h>
#include"highgui.h"
#include <fstream>
#include <iostream> // not required by most systems
using namespace std;//!!!!!!!!!!!!!!!!!!!!can not use ofstream without this line
#define PI 3.1415926
#define yipu_u 0.00001//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#define yipu_mat 0.00001//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
int best_times=0,get_accuracy_flag=0;
long double best_accuracy=0;
void load_train_data(long double trainA[1045][3],long double trainB[846][3])
{
    char pathname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\train.txt";
    ifstream o_file;
    o_file.open(pathname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    for(Aroll=0;Aroll<1045;Aroll++)
    {
        o_file>>trainA[Aroll][0];//从文件输入一个数值。
        //cout<<trainA[Aroll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][1];//从文件输入一个数值。
        //cout<<trainA[Aroll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainA[Aroll][2];//从文件输入一个数值。
        //cout<<trainA[Aroll][2]<<"\n"<<endl;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    for(Broll=0;Broll<845;Broll++)
    {
        o_file>>trainB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>trainB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    o_file>>trainB[Broll][0];//从文件输入一个数值。
    //cout<<trainB[Broll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>trainB[Broll][1];//从文件输入一个数值。
    //cout<<trainB[Broll][1]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>trainB[Broll][2];//从文件输入一个数值。
    //cout<<trainB[Broll][2]<<"\n"<<endl;
    o_file.close();
}

void get_u_matrix(long double trainA[1045][3],long double trainB[846][3],long double
train_uA[3],long double train_matrixA[3][3],long double train_uB[3],long double
train_matrixB[3][3])
{
    train_uA[0]=0.0;train_uA[1]=0.0;train_uA[2]=0.0;
    train_uB[0]=0.0;train_uB[1]=0.0;train_uB[2]=0.0;
    train_matrixA[0][0]=0.0;train_matrixA[0][1]=0.0;train_matrixA[0][2]=0.0;
```

```
train_matrixA[1][0]=0.0;train_matrixA[1][1]=0.0;train_matrixA[1][2]=0.0;
train_matrixA[2][0]=0.0;train_matrixA[2][1]=0.0;train_matrixA[2][2]=0.0;
train_matrixB[0][0]=0.0;train_matrixB[0][1]=0.0;train_matrixB[0][2]=0.0;
train_matrixB[1][0]=0.0;train_matrixB[1][1]=0.0;train_matrixB[1][2]=0.0;
train_matrixB[2][0]=0.0;train_matrixB[2][1]=0.0;train_matrixB[2][2]=0.0;
int i;
for(i=0;i<1045;i++)
{
    train_uA[0]+=trainA[i][0];
    train_uA[1]+=trainA[i][1];
    train_uA[2]+=trainA[i][2];
}
train_uA[0]=train_uA[0]/1045.0;
train_uA[1]=train_uA[1]/1045.0;
train_uA[2]=train_uA[2]/1045.0;
///////////////cout<<"train_uA[0]"<<train_uA[0]<<"   ";
///////////////cout<<"train_uA[1]"<<train_uA[1]<<"   ";
////////////////cout<<"train_uA[2]"<<train_uA[2]<<"\n";
for(i=0;i<846;i++)
{
    train_uB[0]+=trainB[i][0];
    train_uB[1]+=trainB[i][1];
    train_uB[2]+=trainB[i][2];
}
train_uB[0]=train_uB[0]/846.0;
train_uB[1]=train_uB[1]/846.0;
train_uB[2]=train_uB[2]/846.0;
/////////////////cout<<"train_uB[0]"<<train_uB[0]<<"   ";
/////////////////cout<<"train_uB[1]"<<train_uB[1]<<"   ";
/////////////////cout<<"train_uB[2]"<<train_uB[2]<<"\n";
for(i=0;i<1045;i++)
{
    train_matrixA[0][0]+=(trainA[i][0]-train_uA[0])*(trainA[i][0]-train_uA[0]);
    train_matrixA[1][1]+=(trainA[i][1]-train_uA[1])*(trainA[i][1]-train_uA[1]);
    train_matrixA[2][2]+=(trainA[i][2]-train_uA[2])*(trainA[i][2]-train_uA[2]);
}
train_matrixA[0][0]=train_matrixA[0][0]/1045.0;
train_matrixA[1][1]=train_matrixA[1][1]/1045.0;
train_matrixA[2][2]=train_matrixA[2][2]/1045.0;
for(i=0;i<846;i++)
{
    train_matrixB[0][0]+=(trainB[i][0]-train_uA[0])*(trainB[i][0]-train_uA[0]);
    train_matrixB[1][1]+=(trainB[i][1]-train_uB[1])*(trainB[i][1]-train_uB[1]);
    train_matrixB[2][2]+=(trainB[i][2]-train_uB[2])*(trainB[i][2]-train_uB[2]);
    }
    train_matrixB[0][0]=train_matrixB[0][0]/846.0;
    train_matrixB[1][1]=train_matrixB[1][1]/846.0;
    train_matrixB[2][2]=train_matrixB[2][2]/846.0;

      /*
    for(i=0;i<1045;i++)//实际上协方差系数较大
    {
        train_matrixA[0][1]+=(trainA[i][0]-train_uA[0])*(trainA[i][1]-train_uA[1]);
        train_matrixA[0][2]+=(trainA[i][0]-train_uA[0])*(trainA[i][2]-train_uA[2]);
        train_matrixA[1][0]+=(trainA[i][1]-train_uA[1])*(trainA[i][0]-train_uA[0]);
        train_matrixA[1][2]+=(trainA[i][1]-train_uA[1])*(trainA[i][2]-train_uA[2]);
        train_matrixA[2][0]+=(trainA[i][2]-train_uA[2])*(trainA[i][0]-train_uA[0]);
        train_matrixA[2][1]+=(trainA[i][2]-train_uA[2])*(trainA[i][1]-train_uA[1]);
    }

    for(i=0;i<846;i++)
    {
        train_matrixB[0][1]+=(trainB[i][0]-train_uB[0])*(trainB[i][1]-train_uB[1]);
        train_matrixB[0][2]+=(trainB[i][0]-train_uB[0])*(trainB[i][2]-train_uB[2]);
        train_matrixB[1][0]+=(trainB[i][1]-train_uB[1])*(trainB[i][0]-train_uB[0]);
        train_matrixB[1][2]+=(trainB[i][1]-train_uB[1])*(trainB[i][2]-train_uB[2]);
        train_matrixB[2][0]+=(trainB[i][2]-train_uB[2])*(trainB[i][0]-train_uB[0]);
        train_matrixB[2][1]+=(trainB[i][2]-train_uB[2])*(trainB[i][1]-train_uB[1]);
    }
      */
    train_matrixA[0][1]=train_matrixA[0][1]/1045.0;
    train_matrixA[0][2]=train_matrixA[0][2]/1045.0;
    train_matrixA[1][0]=train_matrixA[1][0]/1045.0;
    train_matrixA[1][2]=train_matrixA[1][2]/1045.0;
    train_matrixA[2][0]=train_matrixA[2][0]/1045.0;
    train_matrixA[2][1]=train_matrixA[2][1]/1045.0;

    train_matrixB[0][1]=train_matrixB[0][1]/846.0;
    train_matrixB[0][2]=train_matrixB[0][2]/846.0;
    train_matrixB[1][0]=train_matrixB[1][0]/846.0;
    train_matrixB[1][2]=train_matrixB[1][2]/846.0;
    train_matrixB[2][0]=train_matrixB[2][0]/846.0;
    train_matrixB[2][1]=train_matrixB[2][1]/846.0;

    ////////////////cout<<"train_matrixA\n"<<train_matrixA[0][0]<<"
"<<train_matrixA[0][1]<<"  "<<train_matrixA[0][2]<<"\n";
    ////////////////cout<<train_matrixA[1][0]<<"  "<<train_matrixA[1][1]<<"
"<<train_matrixA[1][2]<<"\n";
```

```
    ////////////////cout<<train_matrixA[2][0]<<"  "<<train_matrixA[2][1]<<"
"<<train_matrixA[2][2]<<"\n";

    ////////////////cout<<"train_matrixB\n"<<train_matrixB[0][0]<<"
"<<train_matrixB[0][1]<<"  "<<train_matrixB[0][2]<<"\n";
    ////////////////cout<<train_matrixB[1][0]<<"  "<<train_matrixB[1][1]<<"
"<<train_matrixB[1][2]<<"\n";
    ////////////////cout<<train_matrixB[2][0]<<"  "<<train_matrixB[2][1]<<"
"<<train_matrixB[2][2]<<"\n";

}

void judge(/*long double test[3],*/long double train_uA[3],long double
train_matrixA[3][3],long double train_uB[3],long double train_matrixB[3][3])//判断某个数
据属于哪一类
{
    long double testA[470][3],testB[360][3];
    long double right=0.0;long double sum=0.0;
    char pathtestname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\test.txt";
    ifstream o_file;
    o_file.open(pathtestname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    long double Areverse_matrix[3],Breverse_matrix[3],valueA,valueB;
    Areverse_matrix[0]=1.0/train_matrixA[0][0];
    Areverse_matrix[1]=1.0/train_matrixA[1][1];
    Areverse_matrix[2]=1.0/train_matrixA[2][2];
    Breverse_matrix[0]=1.0/train_matrixB[0][0];
    Breverse_matrix[1]=1.0/train_matrixB[1][1];
    Breverse_matrix[2]=1.0/train_matrixB[2][2];
    for(Aroll=0;Aroll<470;Aroll++)
    {
        o_file>>testA[Aroll][0];//从文件输入一个数值。
        //cout<<trainA[Aroll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testA[Aroll][1];//从文件输入一个数值。
        //cout<<trainA[Aroll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testA[Aroll][2];//从文件输入一个数值。
        //cout<<trainA[Aroll][2]<<"\n"<<endl;

    valueA=exp(-0.5*((testA[Aroll][0]-train_uA[0])*(testA[Aroll][0]-train_uA[0])*Arever
se_matrix[0]+(testA[Aroll][1]-train_uA[1])*(testA[Aroll][1]-train_uA[1])*Areverse_matri
x[1]+(testA[Aroll][2]-train_uA[2])*(testA[Aroll][2]-train_uA[2])*Areverse_matrix[2]))/(
pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));

    valueB=exp(-0.5*((testA[Aroll][0]-train_uB[0])*(testA[Aroll][0]-train_uB[0])*Brever
se_matrix[0]+(testA[Aroll][1]-train_uB[1])*(testA[Aroll][1]-train_uB[1])*Breverse_matri
x[1]+(testA[Aroll][2]-train_uB[2])*(testA[Aroll][2]-train_uB[2])*Breverse_matrix[2]))/(
pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
        sum=sum+1.0;
        if(valueA>valueB)
            right=right+1.0;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    for(Broll=0;Broll<359;Broll++)
    {
        o_file>>testB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>testB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;

    valueA=exp(-0.5*((testB[Broll][0]-train_uA[0])*(testB[Broll][0]-train_uA[0])*Arever
se_matrix[0]+(testB[Broll][1]-train_uA[1])*(testB[Broll][1]-train_uA[1])*Areverse_matri
x[1]+(testB[Broll][2]-train_uA[2])*(testB[Broll][2]-train_uA[2])*Areverse_matrix[2]))/(
pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));

    valueB=exp(-0.5*((testB[Broll][0]-train_uB[0])*(testB[Broll][0]-train_uB[0])*Brever
se_matrix[0]+(testB[Broll][1]-train_uB[1])*(testB[Broll][1]-train_uB[1])*Breverse_matri
x[1]+(testB[Broll][2]-train_uB[2])*(testB[Broll][2]-train_uB[2])*Breverse_matrix[2]))/(
pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
        sum=sum+1.0;
        if(valueA<valueB)
            right=right+1.0;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    o_file>>testB[Broll][0];//从文件输入一个数值。
    //cout<<trainB[Broll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testB[Broll][1];//从文件输入一个数值。
    //cout<<trainB[Broll][1]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>testB[Broll][2];//从文件输入一个数值。
    //cout<<trainB[Broll][2]<<"\n"<<endl;
    valueA=exp(-0.5*((testB[Broll][0]-train_uA[0])*(testB[Broll][0]-train_uA[0])*Areverse_matrix[0]+(testB[Broll][1]-train_uA[1])*(testB[Broll][1]-train_uA[1])*Areverse_matrix[1]+(testB[Broll][2]-train_uA[2])*(testB[Broll][2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));
    valueB=exp(-0.5*((testB[Broll][0]-train_uB[0])*(testB[Broll][0]-train_uB[0])*Breverse_matrix[0]+(testB[Broll][1]-train_uB[1])*(testB[Broll][1]-train_uB[1])*Breverse_matrix[1]+(testB[Broll][2]-train_uB[2])*(testB[Broll][2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
    sum=sum+1.0;

    if(valueA<valueB)
        right=right+1.0;
    long double accuracy=right/sum;

    if(accuracy>best_accuracy)
    {
        get_accuracy_flag=1;
        best_accuracy=accuracy;
    }
    cout<<"用测试数据进行检测  共有"<<sum<<"个，有"<<right<<"个正确  精确度为"<<accuracy<<"\n";
    o_file.close();
}


void judge_train(/*long double test[3],*/long double train_uA[3],long double train_matrixA[3][3],long double train_uB[3],long double train_matrixB[3][3])//判断某个数据属于哪一类
{
    long double train_testA[1045][3],train_testB[846][3];
    long double right=0.0;long double sum=0.0;
    char pathtestname[256]="E:\\USTC\\senior 1\\Signal Statistical Modeling\\train.txt";
    ifstream o_file;
    o_file.open(pathtestname);
    int Aroll,Broll;
    char a1;
    o_file.seekg(4,ios::cur);//读指针位置向后4格
    long double Areverse_matrix[3],Breverse_matrix[3],valueA,valueB;
    Areverse_matrix[0]=1.0/train_matrixA[0][0];
    Areverse_matrix[1]=1.0/train_matrixA[1][1];
    Areverse_matrix[2]=1.0/train_matrixA[2][2];
    Breverse_matrix[0]=1.0/train_matrixB[0][0];
    Breverse_matrix[1]=1.0/train_matrixB[1][1];
    Breverse_matrix[2]=1.0/train_matrixB[2][2];
    for(Aroll=0;Aroll<1045;Aroll++)
    {
        o_file>>train_testA[Aroll][0];//从文件输入一个数值。
        //cout<<trainA[Aroll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>train_testA[Aroll][1];//从文件输入一个数值。
        //cout<<trainA[Aroll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>train_testA[Aroll][2];//从文件输入一个数值。
        //cout<<trainA[Aroll][2]<<"\n"<<endl;

    valueA=exp(-0.5*((train_testA[Aroll][0]-train_uA[0])*(train_testA[Aroll][0]-train_uA[0])*Areverse_matrix[0]+(train_testA[Aroll][1]-train_uA[1])*(train_testA[Aroll][1]-train_uA[1])*Areverse_matrix[1]+(train_testA[Aroll][2]-train_uA[2])*(train_testA[Aroll][2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));

    valueB=exp(-0.5*((train_testA[Aroll][0]-train_uB[0])*(train_testA[Aroll][0]-train_uB[0])*Breverse_matrix[0]+(train_testA[Aroll][1]-train_uB[1])*(train_testA[Aroll][1]-train_uB[1])*Breverse_matrix[1]+(train_testA[Aroll][2]-train_uB[2])*(train_testA[Aroll][2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
        sum=sum+1.0;
        if(valueA>valueB)
            right=right+1.0;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    for(Broll=0;Broll<845;Broll++)
    {
        o_file>>train_testB[Broll][0];//从文件输入一个数值。
        //cout<<trainB[Broll][0]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>train_testB[Broll][1];//从文件输入一个数值。
        //cout<<trainB[Broll][1]<<"\n"<<endl;
        o_file.seekg(1,ios::cur);
        o_file>>train_testB[Broll][2];//从文件输入一个数值。
        //cout<<trainB[Broll][2]<<"\n"<<endl;

    valueA=exp(-0.5*((train_testB[Broll][0]-train_uA[0])*(train_testB[Broll][0]-train_uA[0])*Areverse_matrix[0]+(train_testB[Broll][1]-train_uA[1])*(train_testB[Broll][1]-train_uA[1])*Areverse_matrix[1]+(train_testB[Broll][2]-train_uA[2])*(train_testB[Broll][2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));

    valueB=exp(-0.5*((train_testB[Broll][0]-train_uB[0])*(train_testB[Broll][0]-train_uB[0])*Breverse_matrix[0]+(train_testB[Broll][1]-train_uB[1])*(train_testB[Broll][1]-train_uB[1])*Breverse_matrix[1]+(train_testB[Broll][2]-train_uB[2])*(train_testB[Broll][2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
        sum=sum+1.0;
        if(valueA<valueB)
            right=right+1.0;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;
        o_file>>a1;//转到下一行
    }
    o_file>>train_testB[Broll][0];//从文件输入一个数值。
    //cout<<trainB[Broll][0]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>train_testB[Broll][1];//从文件输入一个数值。
    //cout<<trainB[Broll][1]<<"\n"<<endl;
    o_file.seekg(1,ios::cur);
    o_file>>train_testB[Broll][2];//从文件输入一个数值。
    //cout<<trainB[Broll][2]<<"\n"<<endl;
    valueA=exp(-0.5*((train_testB[Broll][0]-train_uA[0])*(train_testB[Broll][0]-train_uA[0])*Areverse_matrix[0]+(train_testB[Broll][1]-train_uA[1])*(train_testB[Broll][1]-train_uA[1])*Areverse_matrix[1]+(train_testB[Broll][2]-train_uA[2])*(train_testB[Broll][2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));
    valueB=exp(-0.5*((train_testB[Broll][0]-train_uB[0])*(train_testB[Broll][0]-train_uB[0])*Breverse_matrix[0]+(train_testB[Broll][1]-train_uB[1])*(train_testB[Broll][1]-train_uB[1])*Breverse_matrix[1]+(train_testB[Broll][2]-train_uB[2])*(train_testB[Broll][2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,1.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
    sum=sum+1.0;

    if(valueA<valueB)
```

```
        right=right+1.0;
    long double accuracy=right/sum;

    if(accuracy>best_accuracy)
    {
        get_accuracy_flag=1;
        best_accuracy=accuracy;
    }
    //cout<<"用训练数据进行检测  共有"<<sum<<"个，有"<<right<<"个正确   精确度为"<<accuracy<<"\n";
    o_file.close();
}



long double MCE_d(long double X[3],int Ct,long double train_uA[3],long double train_matrixA[3][3],long double train_uB[3],long double train_matrixB[3][3])
{
    long double Areverse_matrix[3],Breverse_matrix[3],valueA,valueB,lastvalue;
    Areverse_matrix[0]=1.0/train_matrixA[0][0];
    Areverse_matrix[1]=1.0/train_matrixA[1][1];
    Areverse_matrix[2]=1.0/train_matrixA[2][2];
    Breverse_matrix[0]=1.0/train_matrixB[0][0];
    Breverse_matrix[1]=1.0/train_matrixB[1][1];
    Breverse_matrix[2]=1.0/train_matrixB[2][2];
    valueA=exp(-0.5*((X[0]-train_uA[0])*(X[0]-train_uA[0])*Areverse_matrix[0]+(X[1]-train_uA[1])*(X[1]-train_uA[1])*Areverse_matrix[1]+(X[2]-train_uA[2])*(X[2]-train_uA[2])*Areverse_matrix[2]))/(pow(2*PI,0.5)*pow(train_matrixA[0][0]*train_matrixA[1][1]*train_matrixA[2][2],0.5));
    valueB=exp(-0.5*((X[0]-train_uB[0])*(X[0]-train_uB[0])*Breverse_matrix[0]+(X[1]-train_uB[1])*(X[1]-train_uB[1])*Breverse_matrix[1]+(X[2]-train_uB[2])*(X[2]-train_uB[2])*Breverse_matrix[2]))/(pow(2*PI,0.5)*pow(train_matrixB[0][0]*train_matrixB[1][1]*train_matrixB[2][2],0.5));
    long double pA=1045.0/1891.0;
    long double pB=846.0/1891.0;
    if(Ct==0)//A
        lastvalue=pB*valueB-pA*valueA;
    else
        lastvalue=pA*valueA-pB*valueB;

    return lastvalue;


}
```

```
long double MCE_l_d(long double X[3],int Ct,long double a,long double train_uA[3],long
double train_matrixA[3][3],long double train_uB[3],long double train_matrixB[3][3])
{
    long double
lastvaluel=1.0/(1.0+exp((0.0-a)*MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrix
B)));
    return lastvaluel;
}

long double MCE_Q(long double trainA[1045][3],long double trainB[846][3],long double a,long
double train_uA[3],long double train_matrixA[3][3],long double train_uB[3],long double
train_matrixB[3][3])
{
    int t;
    long double lastvalueQ=0.0;
    for(t=0;t<1045;t++)

    lastvalueQ+=MCE_l_d(trainA[t],0,a,train_uA,train_matrixA,train_uB,train_matrixB);
    for(t=0;t<846;t++)

    lastvalueQ+=MCE_l_d(trainB[t],1,a,train_uA,train_matrixA,train_uB,train_matrixB);
    cout<<lastvalueQ<<"\n";
    return lastvalueQ;
}




long double calculus_d_by_x(int which_xcanshu,long double X[3],int Ct,long double
train_uA[3],long double train_matrixA[3][3],long double train_uB[3],long double
train_matrixB[3][3])
{
    //canshu 0------11
    long double cal_d_value;
    if(which_xcanshu==0||which_xcanshu==1||which_xcanshu==2)//修改A的均值
    {
        long double duA=0.00001;
        long double nominator1,nominator2;
        nominator1=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);
        train_uA[which_xcanshu]=train_uA[which_xcanshu]-duA;
        nominator2=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);
        cal_d_value=(100000*nominator1-100000*nominator2);
        train_uA[which_xcanshu]=train_uA[which_xcanshu]+duA;

    }
    else if(which_xcanshu==3||which_xcanshu==4||which_xcanshu==5)//修改A的方差
    {
        long double duA=0.00001;
        long double nominator1,nominator2;
        nominator1=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);

    train_matrixA[which_xcanshu-3][which_xcanshu-3]=train_matrixA[which_xcanshu-3][which_xcanshu-3]-duA;
        nominator2=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);
        cal_d_value=(100000*nominator1-100000*nominator2);

    train_matrixA[which_xcanshu-3][which_xcanshu-3]=train_matrixA[which_xcanshu-3][which_xcanshu-3]+duA;

    }
    else if(which_xcanshu==6||which_xcanshu==7||which_xcanshu==8)//修改B的均值
    {
        long double duB=0.00001;
        long double nominator1,nominator2;
        nominator1=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);
        train_uB[which_xcanshu-6]=train_uB[which_xcanshu-6]-duB;
        nominator2=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);
        cal_d_value=(100000*nominator1-100000*nominator2);
        train_uB[which_xcanshu-6]=train_uB[which_xcanshu-6]+duB;

    }
    else if(which_xcanshu==9||which_xcanshu==10||which_xcanshu==11)//修改B的方差
    {
        long double duB=0.00001;
        long double nominator1,nominator2;
        nominator1=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);

    train_matrixB[which_xcanshu-9][which_xcanshu-9]=train_matrixB[which_xcanshu-9][which_xcanshu-9]-duB;
        nominator2=MCE_d(X,Ct,train_uA,train_matrixA,train_uB,train_matrixB);
        cal_d_value=(100000*nominator1-100000*nominator2);

    train_matrixB[which_xcanshu-9][which_xcanshu-9]=train_matrixB[which_xcanshu-9][which_xcanshu-9]+duB;

    }
    return cal_d_value;
}

long double calculus_Q_by_x(long double a,int which_xcanshu,long double
trainA[1045][3],long double trainB[846][3],long double train_uA[3],long double
train_matrixA[3][3],long double train_uB[3],long double train_matrixB[3][3])
{
    /*
    int t;
    long double lastvalueQ=0.0;
    for(t=0;t<1045;t++)

    lastvalueQ+=MCE_l_d(trainA[t],0,a,train_uA,train_matrixA,train_uB,train_matrixB);
    for(t=0;t<846;t++)

    lastvalueQ+=MCE_l_d(trainB[t],1,a,train_uA,train_matrixA,train_uB,train_matrixB);
    return lastvalueQ;
    */
    int t;
    long double lastvalueQ=0.0;
    for(t=0;t<1045;t++)
    {
        long double
ld=MCE_l_d(trainA[t],0,a,train_uA,train_matrixA,train_uB,train_matrixB);

    lastvalueQ+=a*ld*(1-ld)*calculus_d_by_x(which_xcanshu,trainA[t],0,train_uA,train_ma
trixA,train_uB,train_matrixB);
    }
    for(t=0;t<846;t++)
    {
        long double
ld=MCE_l_d(trainB[t],1,a,train_uA,train_matrixA,train_uB,train_matrixB);

    lastvalueQ+=a*ld*(1-ld)*calculus_d_by_x(which_xcanshu,trainB[t],1,train_uA,train_ma
trixA,train_uB,train_matrixB);
    }
    return lastvalueQ;//第一个数据的值是50
}

void iteration(long double trainA[1045][3],long double trainB[846][3],long double
train_uA[3],long double train_matrixA[3][3],long double train_uB[3],long double
train_matrixB[3][3])
{
    long double
roll_train_uA[500][3],roll_train_matrixA[500][3][3],roll_train_uB[500][3],roll_train_ma
trixB[500][3][3];
    for(int ai=0;ai<3;ai++)
    {
            roll_train_uA[0][ai]=train_uA[ai];
            roll_train_uB[0][ai]=train_uB[ai];//初始值
            for(int ak=0;ak<3;ak++)
            {
                roll_train_matrixA[0][ai][ak]=train_matrixA[ai][ak];
                roll_train_matrixB[0][ai][ak]=train_matrixB[ai][ak];
            }
    }
    int time_i=0;
    //long double a=1000000.0;//@@@@@@@@@@@@@@10000比较合理  @@还要监视yipu的效果
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    long double a=1000000.0;//@@@@@@@@@@@@@@10000,1000000比较合理  @@还要监视yipu的效果
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

    while(1)
    {
        printf("第%d次",time_i+1);
        for(int dividei=0;dividei<3;dividei++)//u and matrix初始化为0
        {
            roll_train_uA[time_i+1][dividei]=roll_train_uA[time_i][dividei];
            roll_train_uB[time_i+1][dividei]=roll_train_uB[time_i][dividei];
            for(int dividej=0;dividej<3;dividej++)
            {

    roll_train_matrixA[time_i+1][dividei][dividej]=roll_train_matrixA[time_i][dividei][
dividej];

    roll_train_matrixB[time_i+1][dividei][dividej]=roll_train_matrixB[time_i][dividei][
dividej];
            }
        }

        ////////////////cout<<"begin to calculate\n";
        for(int iter12=0;iter12<12;iter12++)
        {
            if(iter12==0||iter12==1||iter12==2)//修改A的均值

    roll_train_uA[time_i+1][iter12]-=yipu_u*calculus_Q_by_x(a,iter12,trainA,trainB,roll
_train_uA[time_i],roll_train_matrixA[time_i],roll_train_uB[time_i],roll_train_matrixB[t
ime_i]);
            if(iter12==3||iter12==4||iter12==5)

    roll_train_matrixA[time_i+1][iter12-3][iter12-3]-=yipu_mat*calculus_Q_by_x(a,iter12,trainA,trainB,roll_train_uA[time_i],roll_train_matrixA[time_i],roll_train_uB[time_i],roll_train_matrixB[time_i]);
            if(iter12==6||iter12==7||iter12==8)

    roll_train_uB[time_i+1][iter12-6]-=yipu_u*calculus_Q_by_x(a,iter12,trainA,trainB,roll_train_uA[time_i],roll_train_matrixA[time_i],roll_train_uB[time_i],roll_train_matrixB[time_i]);
            if(iter12==9||iter12==10||iter12==11)

    roll_train_matrixB[time_i+1][iter12-9][iter12-9]-=yipu_mat*calculus_Q_by_x(a,iter12,trainA,trainB,roll_train_uA[time_i],roll_train_matrixA[time_i],roll_train_uB[time_i],roll_train_matrixB[time_i]);
        }


        //只用于输出Q的值

    //MCE_Q(trainA,trainB,a,roll_train_uA[time_i+1],roll_train_matrixA[time_i+1],roll_train_uB[time_i+1],roll_train_matrixB[time_i+1]);

        //输出系数
        ////////////////cout<<"uA  ";
        ////////////////
        /*
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            cout<<"  "<<roll_train_uA[time_i+1][printcenteri];
        cout<<"  uA matrix";
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            cout<<"  "<<roll_train_matrixA[time_i+1][printcenteri][printcenteri];
        cout<<"\n";
        cout<<"uB  ";
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            cout<<"  "<<roll_train_uB[time_i+1][printcenteri];
        cout<<"  uB matrix";
        for(int printcenteri=0;printcenteri<3;printcenteri++)
            cout<<"  "<<roll_train_matrixB[time_i+1][printcenteri][printcenteri];
        cout<<"\n";
        */
        get_accuracy_flag=0;
        //test accuracy
    //get_accuracy=judge(roll_train_uA[i+1],roll_train_matrixA[i+1],roll_train_wA[i+1],roll_train_uB[i+1],roll_train_matrixB[i+1],roll_train_wB[i+1]);

    judge(roll_train_uA[time_i+1],roll_train_matrixA[time_i+1],roll_train_uB[time_i+1],roll_train_matrixB[time_i+1]);

    //judge_train(roll_train_uA[time_i+1],roll_train_matrixA[time_i+1],roll_train_uB[time_i+1],roll_train_matrixB[time_i+1]);
        if(get_accuracy_flag==1)
        {
            best_times=time_i+1;
        }
        get_accuracy_flag=0;





        int whetherbreak=1;

        for(int printcenteri=0;printcenteri<3;printcenteri++)
        {

    if(roll_train_uA[time_i][printcenteri]!=roll_train_uA[time_i+1][printcenteri])
                whetherbreak=0;

    if(roll_train_matrixA[time_i+1][printcenteri][printcenteri]!=roll_train_matrixA[time_i][printcenteri][printcenteri])
                whetherbreak=0;

    if(roll_train_uB[time_i+1][printcenteri]!=roll_train_uB[time_i][printcenteri])
                whetherbreak=0;

    if(roll_train_matrixB[time_i+1][printcenteri][printcenteri]!=roll_train_matrixB[time_i][printcenteri][printcenteri])
                whetherbreak=0;

        }




        if(whetherbreak==1)
        {
            printf("已经找到类中心\n");
            break;
        }
        if(time_i>=498)
        {
            printf("迭代次数达到限制，停止迭代\n");
            break;
        }

        time_i++;
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //测试长双精度型是否精确
    /*
    long double try1=7.123456789,try2=7.123456788;
    try1=try1-try2;
    try1=try1*100000000.0;
    cout<<try1<<"\n";
    */

    //12个参数train_uA[3],train_matrixA[3][3],train_uB[3],train_matrixB[3][3]
    long double trainA[1045][3],trainB[846][3];//A有1045行数据，B有846行数据
    load_train_data(trainA,trainB);//把数据读到数组里
    long double train_uA[3],train_matrixA[3][3],train_uB[3],train_matrixB[3][3];
    get_u_matrix(trainA,trainB,train_uA,train_matrixA,train_uB,train_matrixB);
    cout<<"已经得到统计系数如上\n";
    iteration(trainA,trainB,train_uA,train_matrixA,train_uB,train_matrixB);
    cout<<"第"<<best_times<<"次迭代得到的精度最高，为"<<best_accuracy;


    return 0;
}
```