机器学习第四次作业——关于无监督学习

统计81 于越 *

April 26, 2021

1 无监督学习与聚类简介

无监督学习和有监督学习不同，我们事先不知道训练样本的标记信息，因此我们需要让学习机对这些无标记训练样本进行学习，从而得到数据所隐含的信息，进而进行后续的分类或是回归操作。基于训练样本的标记信息未知的特点，由于我们事先未知训练样本的标签，其泛化应用的意义也就更加明显，因为我们可以说事先不知道分布，进而我们让学习机采用近似的方式自身构造出了数据的分布，因而感官上让人感觉更加只能。而无监督学习中的代表就是聚类算法。

聚类算法的目的在于将数据集中的样本划分为若干个通常是不相交的子集，进而达到分类的目的。聚类可以揭示数据集的结构，这些结构可以应用于测试集上，从而形成新数据集的分类。聚类的算法有很多种，常用的有k-means聚类方法、GMM混合高斯聚类方法以及层次聚类方法等等。本文中，我将主要介绍这几种聚类方法，并且对于k-means与GMM混合高斯这两种聚类方法，我还自行模拟了均匀分布与高斯分布的测试集进行分类的预测从而进行对比。最后我给出了对于结果以及聚类这一算法的一些思考，这些思考有些是读完周志华老师的《机器学习》后的理解，若有考虑不周全的地方，还希望老师能够指出，谢谢。

2 原型聚类——k-means聚类方法

2.1 原理介绍

原型聚类可以理解为“基于原型的聚类”，主要是基于“训练集原型”进行聚类。而k-means是原型聚类中最基础、最直观的一种。在给定样本集$D = \{x_1, x_2, \cdots, x_m\}$的情况下，我们需要针对聚类的簇划分$C = \{C_1, C_2, \cdots, C_k\}$最小化平方误差

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||_2^2 \tag{1}$$

*学号：2183210516

其中

$$\mu_i = \frac{1}{|C_i|}\sum_{x\in C_i} x \tag{2}$$

是每一个聚类簇C_i的均值向量。

我们可以很显然地看到，k-means中的“距离”就是我们平常接触最多、理解最深刻的“欧几里得距离”。而对于簇的中心，我们也很自然地用每个簇中的元素求均值的方法来计算，因此这一方法很直观也很易于理解。

具体给出k-means的聚类过程如下：在样本集D中随机选择k个样本作为初始均值向量$\{mu_1, mu_2, \cdots, \mu_k\}$.我们开始令各个聚类簇$C_i$都为空集，接着对于每一个样本$x_j, j = 1, 2, \cdots, m$，计算样本与各均值向量$\mu_i, 1 \le i \le k$之间的距离

$$d_{ji} = ||x_j - \mu_i||_2 \tag{3}$$

接着根据距离最近的均值向量确定样本点x_j的簇标记为

$$\lambda_j = argmin_{i\in\{1,2,\cdots,k\}} d_{ji} \tag{4}$$

然后就将x_j给划入对应的聚类簇。这样我们就得到了一些新的簇。

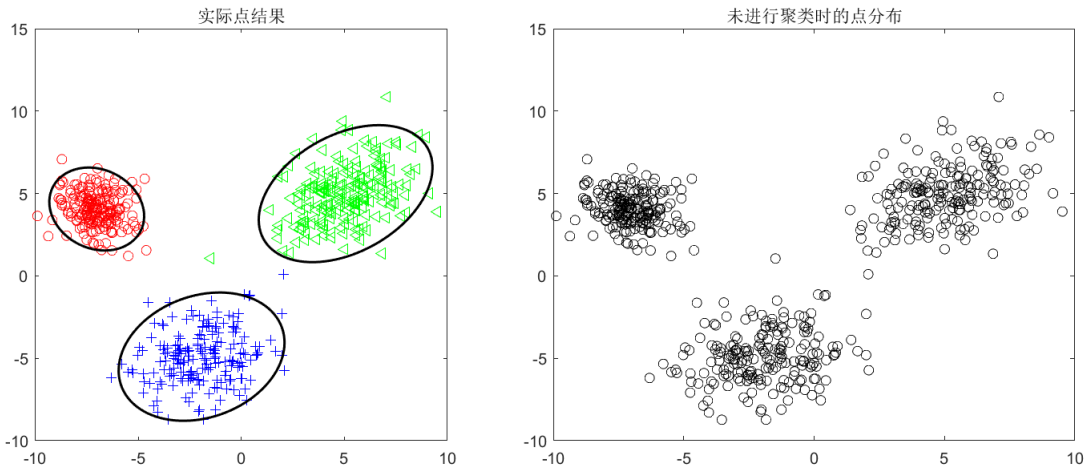
接着对于一次迭代后新生成的聚类簇，用之前计算均值向量的方法计算新的均值向量，并将前后两个的均值向量进行比较，当二者的距离小于某一事先预设好的误差值时就可以保持当前均值不变，否则就用新的均值向量迭代旧的均值向量，完成一次k-means迭代。接着持续进行迭代，直到每一个聚类簇的均值几乎都不发生变化为止，即可得到最终的k个聚类簇，即为聚类结果。

2.2 详细实验——训练集分类

我们在横纵坐标的[-10,10]区间上通过二维高斯分布的方法生成600个点，分为3个高斯分布，每个高斯分布200个点。这600个点就是训练集。在这里，主要对k=3,4,5三种情况进行k-means聚类分类，得到的对应结果如下所示：

2.2.1 k=3情况

我们先给出实际的分类以及未经分类情况下的原数据点，接着再给出实际的分类结果。



(a) k=3的实际情况 (b) k=3的未聚类数据集

Figure 1: k=3情况下的原数据集与实际分类

经过k-means聚类方法得到的结果如下所示：

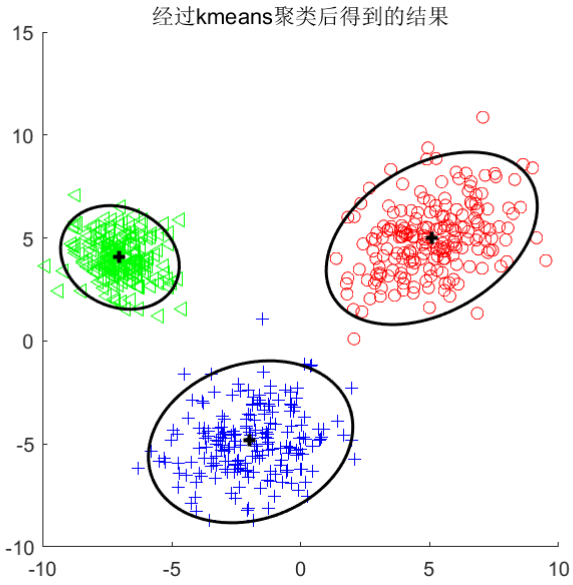


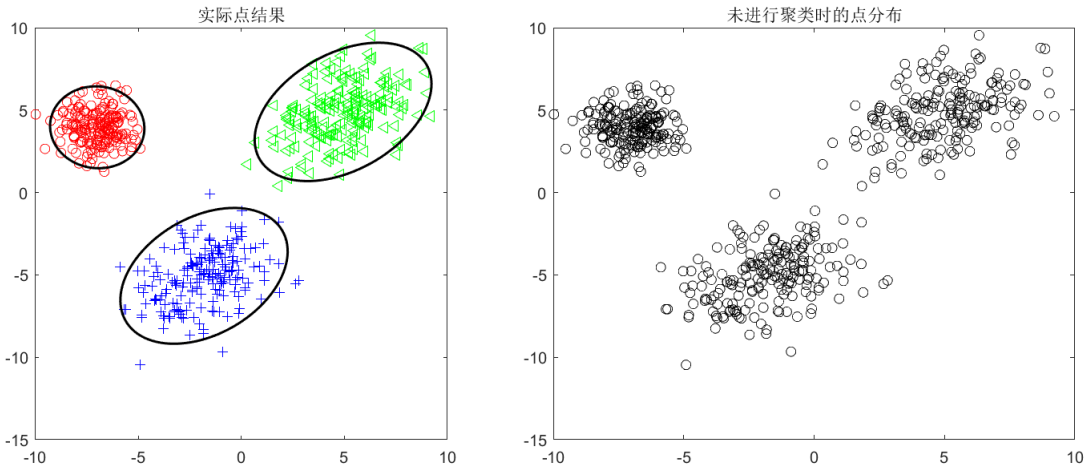
Figure 2: k=3情况下的kmeans结果

图中的分类主要由置信椭圆划分开，聚类簇的中心用黑色实心的十字记号表示。可以看到分类的情况基本上和原先的实际情况是基本类似的，大致只有位于三个簇中心的两个点没有分对，别的基本都是正确的。

值得一提的是，在图像上原来红色的点和绿色的点位置发生了相反的情况，这侧面反映了聚类方法的无监督学习的特性，数据集都是不带标签的，因此聚类的核心任务在于把数据集分类，而具体的属于哪一类，该类有着怎么样的特性还需要后续的操作。

2.2.2 k=4情况

k-means的聚类结果显然是由聚类簇的个数k所决定的。对于不同的k，因为数据是不带标签的，所以可以对于之前的情况再继续细分。k=4的情况下结果如下所示：



(a) k=4的实际情况 (b) k=4的未聚类数据集

Figure 3: k=4情况下的原数据集与实际分类

当k=4时的k-means聚类方法得到的结果如下图：

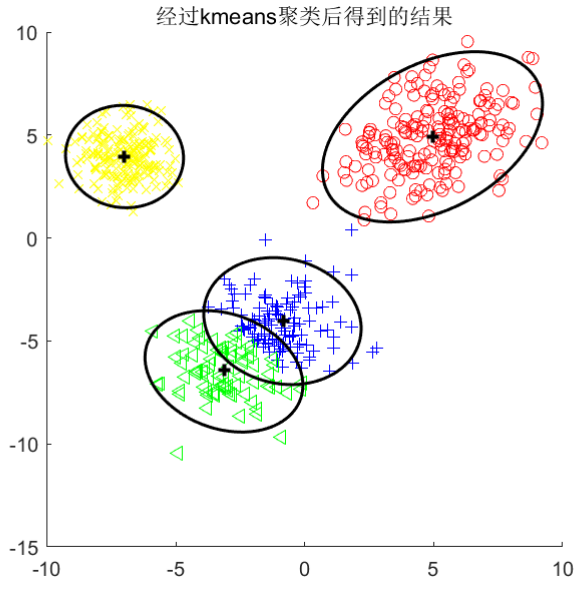


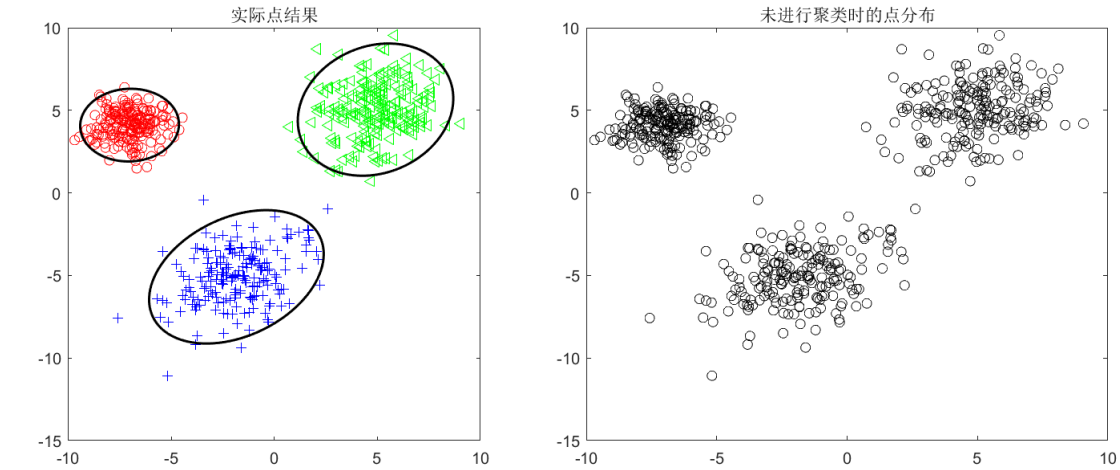
Figure 4: k=4情况下的kmeans结果

我们可以清楚地看到，在图中下面的数据集被分为了两类，经过迭代产生了两个聚类中心。我认为这一情况可以更为清晰地说明k-means聚类方法是更为直观的方法。而且可以看到，此时出现了“多分一类”的情况，但是这一情况仅仅出现在原来3个高斯

分布中的一个高斯分布上。从这一点上可以体现出k-means聚类方法是一种“硬聚类”或者说是“硬分类”的方法，而且值得一提的是，蓝色的点和绿色的点是十分分明的，二者之间不会出现所谓的“交集”，这也体现了“硬分类”的特性。

2.2.3 k=5情况

我们对于k=5的情况也进行同样的处理，先生成未经分类的数据集如下所示：



(a) k=5的实际情况 (b) k=5的未聚类数据集

Figure 5: k=5情况下的原数据集与实际分类

当k=5时的k-means聚类方法得到的结果如下图：

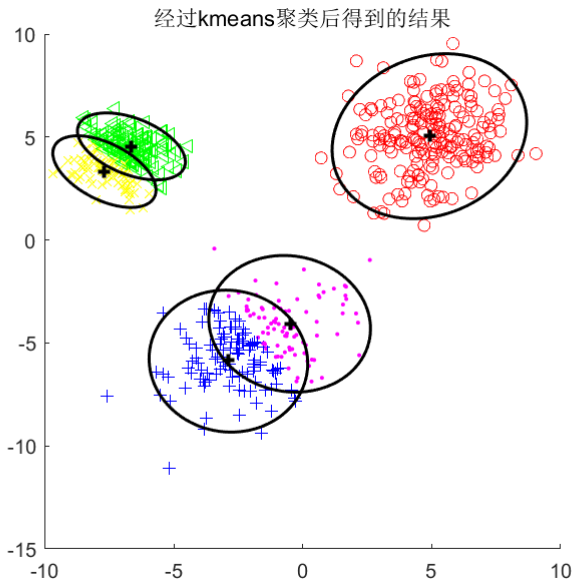


Figure 6: k=5情况下的kmeans结果

可以预料到与k=4时的类似的结果。原来的3个高斯分布中点集分布较为密集的两个高斯分布每个都被分成了新的两类，而且显然边界是十分“分明”的，绝对不会有交集的现象出现。而且对比k=4时的情况，发现位于图像下方的高斯分布的分类几乎没有发生变化，这说明学习机已经认为迭代收敛的聚类不会再进行新的调整，这和“硬聚类”的思想也是契合的。

2.3 关于k-means用于预测

从之前的实验中，我们可以体会到k-means的聚类思想并不复杂，主要是基于欧氏距离出发进行均值的迭代分类。经过训练集的训练后，我们自行给出100个数据点作为测试集进行分类预测。在这里基于k-means的预测思想也是简单的，我自行生成的100个数据点主要是用均匀分布生成了在[-10,10]区间上的100个横纵坐标进行组合，从而生成了测试集。分类预测主要是基于已经聚类完成的聚类均值中心进行的，分别计算100个点与k个聚类中心的2范数距离，与哪一个聚类中心的距离最近就默认属于哪一个聚类。同样地对k=3,4,5进行实验，得到的结果如下：

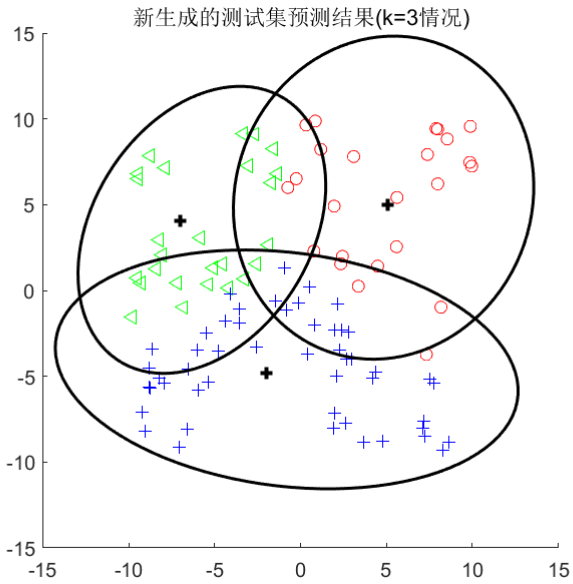


Figure 7: k=3情况下的kmeans训练集分类预测

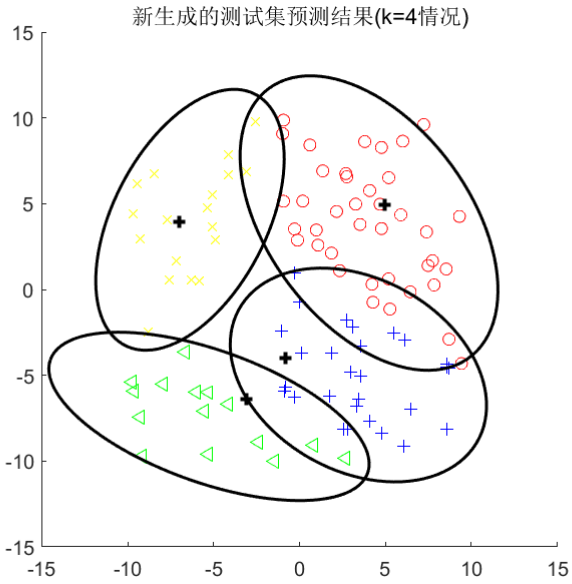


Figure 8: k=4情况下的kmeans训练集分类预测

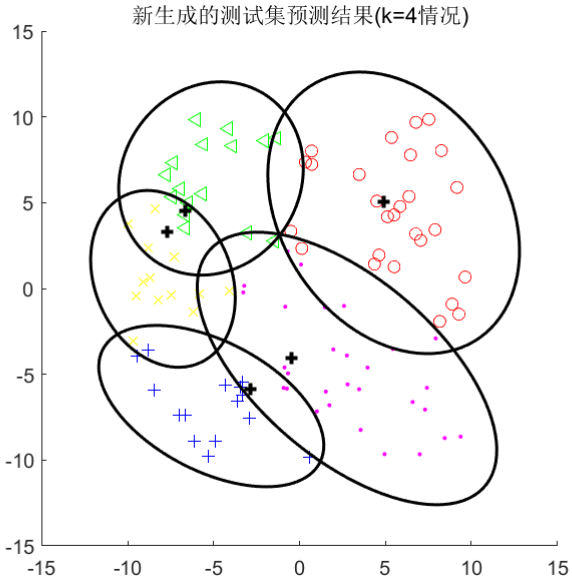


Figure 9: k=5情况下的kmeans训练集分类预测

可以看到此时的分类可以认为是成功的，因为确实完成了分类，而且分类看上去也都大体在置信椭圆内。但是不足之处也是十分明显的。首先，由于是基于直观的欧氏距离所进行的分类，因此很明显地出现了分类边界如何确定的情况。这样的完全基于距离的分类方式，会或多或少地忽略掉数据点的别的特点，从而导致分类的边界不够明晰。其次，这里在k值较小时虽然分类看上去较为清晰，但是在k=5时，对于位于中心的点很容易受到数据集扰动的影响，进而无法得知具体的这一数据点属于哪一类，换言之，此时对于位于中心点的数据的分类是模糊的，好像分哪个类都有着一定道理。最后，显

然当k较大，譬如k=5时，分类的点的聚类中心显然不是原来的黑色实心十字记号的位置，换言之，这里的聚类更新是不到位的，很有可能导致分类的错误。我认为这些和“硬分类”的缺陷也是吻合的。

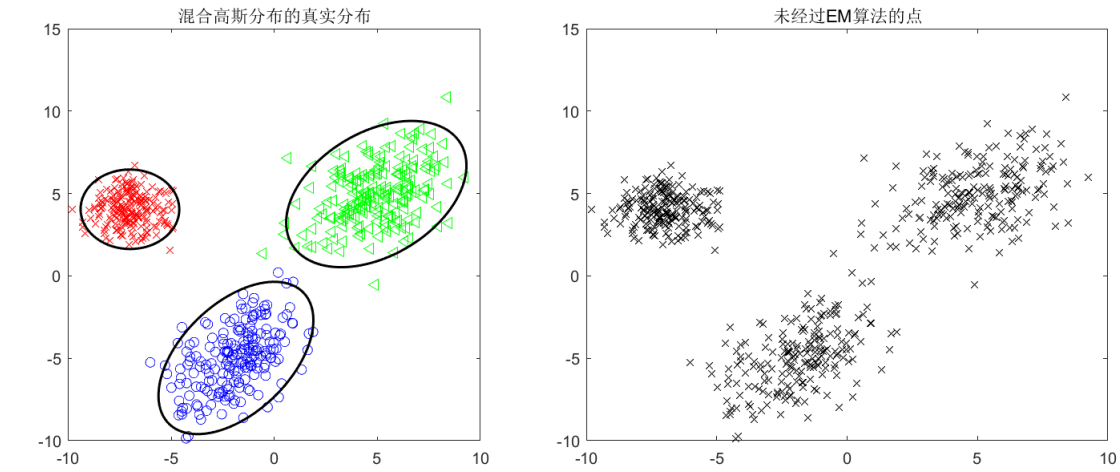
总之,k-means是简单而直观的聚类方法，但终究属于“硬聚类”，在实际应用很多情况下的操作效果并没有十分令人满意。

3 原型聚类——GMM高斯聚类方法

GMM高斯聚类方法我在第三次机器学习关于机器学习中的概率统计意义中已经做过实验和公式推导并且写进了报告里，具体推导可以参照我的第三次机器学习作业。和k-means聚类方法不同，GMM高斯聚类的方法，是有着本身基于的分布的，因此不像k-means方法纯粹进行距离判别的聚类，没有这么的直观，但是通过具体训练集数据点属于哪一个高斯分布的方法进行分类，是一种“软聚类”的方法，可以更好地处理数据的分类问题。

3.1 关于GMM方法的实现与预测

我们也像做k-means聚类那样，先给出600个点形成3个高斯分布的数据集，进行实际的绘图并且生成未经聚类的数据集，如下所示：



(a) GMM实际情况 (b) GMM的未聚类数据集

Figure 10: GMM混合高斯聚类原数据集与实际分类

此时经过GMM混合高斯聚类得到的结果如下所示：

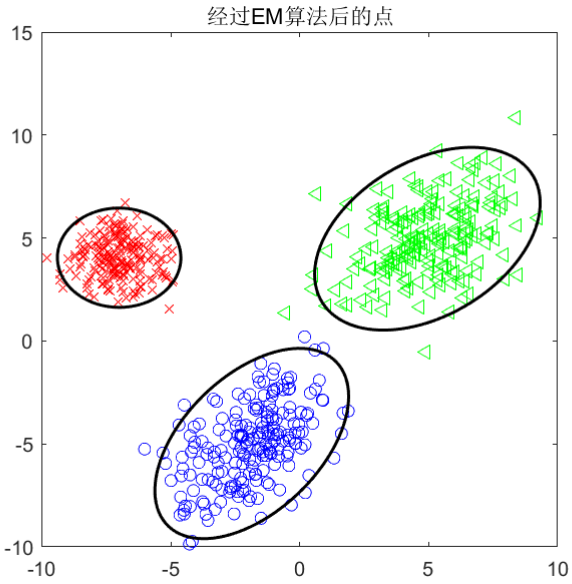


Figure 11: GMM混合高斯聚类结果

可以看到，此时的分类结果与figure10中的实际情况几乎完全一样。值得一提的是对比k-means聚类方法中k=3的情况，位于三个置信椭圆外的点也分类完全正确。从这点来说GMM方法是比k-means方法更好的方法。GMM的思想在于，用混合高斯分布表示所有的点，进而在混合高斯分布中完成参数的迭代，最后将数据点集分配给对应的高斯分布完成聚类。此时的分类准则不会完全基于距离这一单调的分类准则，而是引入了一个具体的分布，因此此时纳入考虑范围的事项更多，分类结果显然也比之前更好。

也同样地新生成100个数据点形成测试集进行分类预测。具体结果如下：

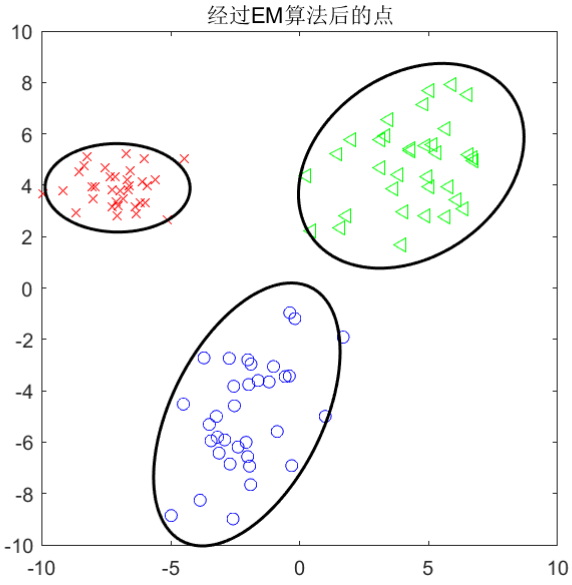


Figure 12: GMM混合高斯聚类的测试集预测

由于此时的数据点也是基于高斯分布生成的，因此看上去的分类结果比之前k-means的也要好上不少。

4 层次聚类方法AGNES

4.1 AGNES介绍

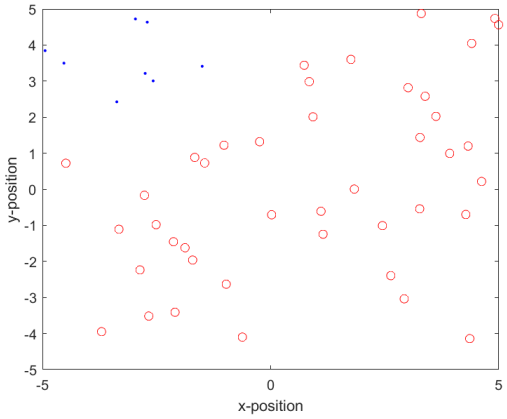
经过前面的两种原型聚类方法——k-means方法与GMM混合高斯方法的实验，我们发现聚类时对于具体分成几个聚类簇是一个较为难以定夺的问题。由此可以引入层次聚类的方法，试图在不同层次上对数据集进行划分，这里层次的划分主要由聚类簇数k来决定。我们可以直观地将数据集的划分理解为一种“自底向上”的聚合策略或者是“自顶向下”的拆分策略。这样对于聚类的模式是更为分明的。

在这里介绍AGNES的层次聚类方法。这是一种自底向上的聚合策略的层次聚类法。我们首先将数据集中的每个样本看作一个初始聚类簇，然后随着算法进行，找出距离最近的两个聚类簇进行合并，不断重复和迭代该过程，直到达到预设的聚类簇的个数为止。此处涉及不同聚类之间的距离，且此时的聚类显然是比之前更为繁多的，因此我们需要引入距离矩阵M.

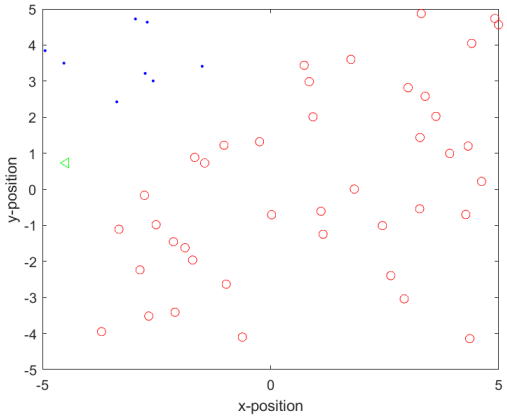
首先对于训练集$D = \{x_1, x_2, \cdots, x_m\}$，设定最终聚类数k.首先我们将每一个点都看成一个单独的聚类簇，接着对于这些聚类簇，计算出它们之间的距离矩阵(对称矩阵)，然后设置下一步的聚类个数q，只要q¿k，我们就合并所有聚类中距离最近的两个聚类簇，在合并完成后进行重新编号，删去已被合并的聚类，更新新的聚类簇。接着再次重复上述的操作，每次迭代都使得聚类个数q减去1，直到q=k为止，完成迭代，即可得到最终的聚类结果。

4.2 具体实现

在这里我采用的数据集是在区间[-5,5]上随着均匀分布生成横纵坐标而随机生成的50个点。我们希望对这50个点进行分类。我们首次设定的聚类簇数q=30，接着进行聚类AGNES方法自底向上的迭代，最后给出k=2,3,4,5,6,7情况下的聚类结果，如下所示：

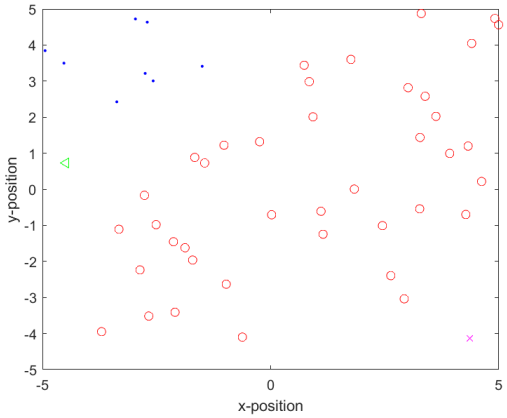


(a) AGNES设定k=2结果

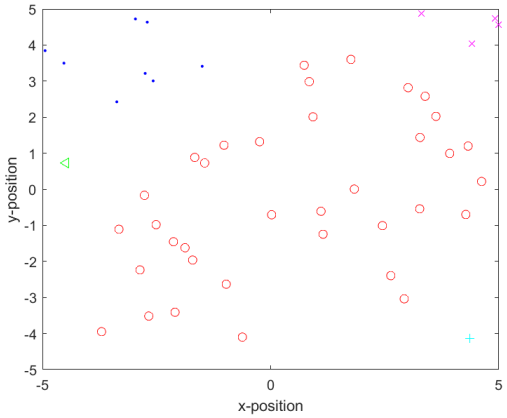


(b) AGNES设定k=3结果

Figure 13: AGNES结果

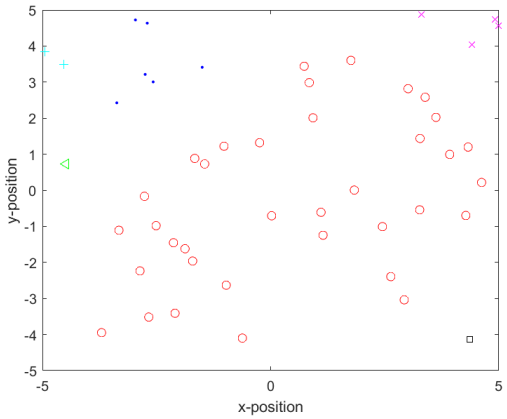


(a) AGNES设定k=4结果

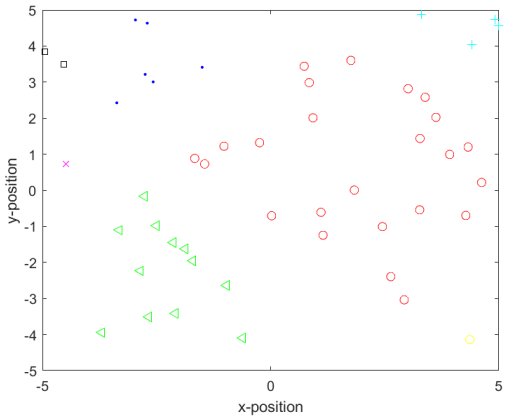


(b) AGNES设定k=5结果

Figure 14: AGNES结果



(a) AGNES设定k=6结果



(b) AGNES设定k=7结果

Figure 15: AGNES结果

注：k=7时7种颜色的点分别为红、蓝、绿、黑(左上角)、紫(左上角)、天蓝(右上角)、黄(右下角一个点，可能不清楚)

可以明显地看出，AGNES方法的优势在于，我们可以采用循环迭代的方式画出不同的k情况下的所有的分类情况。而且从这些图像中，我们也可以筛选出最适合这一数据集的分类簇数k.但这一方法的缺陷在于，当分类簇数过少时，会出现明显的一类簇中元素个数大于另外的簇中元素的情况。因此用AGNES方法，筛选这一工作是必要的。

层次聚类的另一处优势在于我们可以画出层次聚类的谱系图，这使得结果更为清晰明了。对于这一实验的谱系层次聚类图如下所示：

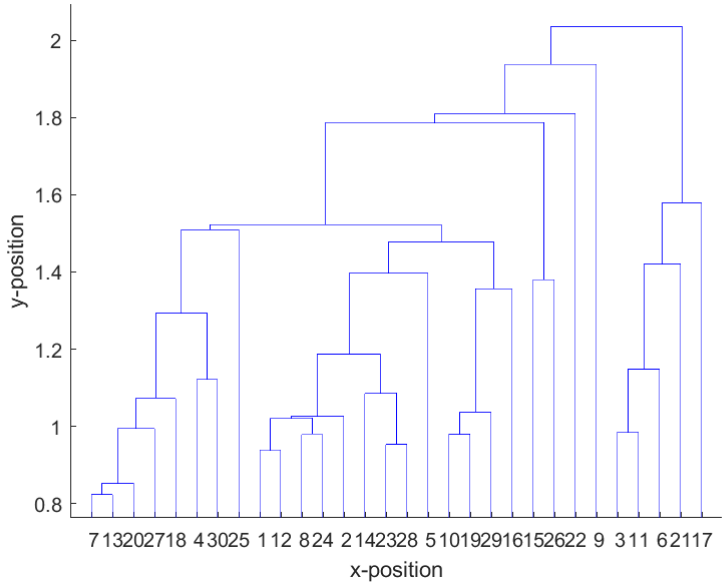


Figure 16: 层次聚类方法的谱系层次聚类树图

这时具体的分类就十分清晰，我们也可以基于这一谱系图进行筛选，得到最优的聚类k值。这一方法在小样本聚类时效果十分突出，分类细节都被完全体现出来了。

5 一些思考与总结

5.1 关于无监督学习

首先，机器学习中的“不带标签”这一特点引导我去思考无监督学习。什么是具体的“无监督学习”，我看到这样一个形象的比喻。有监督学习和无监督学习类似于考试和做题，有监督学习可以说是考试时的客观题，因为所有的数据集是带标签的，故我们可以认为这些题目具有所谓的“标准答案”，此时需要让学习机依据“标准答案”进行学习，学习到“答案”背后的信息和精髓，而作用于测试集时，我们可以理解为教会学生“客观题”后让他举一反三，变一变题目的数据让他考试，再将实际的测试集结果作为考试答案检验他学习“客观题答案”的成果。而对于无监督学习，可以理解为一些开

放性的考试或者竞赛，作为学习机的训练者，我们只需要出题目就行，至于这里的题目因为是开放的，所以答案更多偏向主观性，而学习机作为考试的人，需要做到是根据我们所出的“题目”，找到题目所蕴含的信息以及结构，给出他们所理解的解答。这也可以理解为在没有老师的情况下，学生自己的自学和归纳总结，试题没有准确唯一的答案，学生要靠自己的理解进行解答。

因此在有监督学习中，我们需要告诉学习机，这些数据具体是什么，具体的有何作用，而在他们学习完成后，再给出类似的东西让他识别。而对于无监督学习，我们直接给出数据集激发他们的学习兴趣，我们只给特征，但是没有标签，一切都没有标准答案，言之有理即可。

因此，为什么说无监督学习可以进一步引出元学习，元学习需要让学习机自发地掌握学习的模式，在面临不同问题时它能自己意识到应该采用何种方法。这也就类比一个学习过程中的演变问题，有监督学习是老师的指导，我们作为“老师”要引导学习机的学习，接着无监督学习我们要学会让学习机“发散思维”，不要仅仅拘泥于我们所教会它的东西，最后我们要学会放手，让学习机在没有引导的情况下，学会独立地处理问题。我觉得从这一角度出发，问题就易于理解了很多。

5.2 关于两种不同聚类方法以及软、硬聚类

在上文中我已经提到了，k-means聚类方法是一种“硬聚类”的方法，可以看出在聚类时，分类的边界是十分分明的，不同的聚类簇之间显然不会出现交集。这是由于直观的距离作为判别准则从而引起的结果，或者说，距离的判别准则将一切规则给定死了，准则的东西得以用距离的定义完成了所谓的量化，因而不同的聚类簇之间不会有交集，划分是清楚而明确的。但是这也导致了一系列的问题，在预测分类的过程中，可以看到分类效果并不是特别好，新生成的分类之间仍旧有些点是不明确的，这在进一步的预测中会出现问题。如果去比喻一下，硬聚类方法下的学习机就像一个“不知变通、按部就班去工作的人”，依照上级或是先人所给出的“距离”这个准则去判别一切是非好坏，但是一旦遇到较为灵活而又模糊的问题，就只能含糊其辞蒙混过关了，因而效果也就不好。

因此，我们需要将我们的目光投向软聚类。软聚类和硬聚类其实可以和之前回归问题中的正则进行联系。显然，在软聚类中，预测问题可以得到更好的解决，这是易于理解的，因为软聚类算法，譬如GMM混合高斯聚类方法，不再仅仅是用点的均值和距离作为判别依据，其中加入了分布的概念，因此数据集背后的更多信息就被挖掘了出来，此时的学习机掌握了更多信息，运用的判别准则也就更加灵活，对于一些模糊的或者说是“刁难式”的问题就可以给出更好的解决方案。换言之，此时的分类是更加细化的，一些细节的东西被加入其中综合考虑，因而尽管此时的分类边界可能没有那么地分明，不同类之间也许会产生交集，但是正是由于这些类似于“噪声”的交集数据点的存在，软聚类才能更优地运用到后续的分类预测中。这也因此告诉我们，没有掌握完全的信息去判断一个人、一件事往往是片面的、危险的。

这也就启发我们，在真正的AI中，我们真正需要改进的核心是什么，也许不单单仅是一个方法这么简单，或者说更深层次的，是思想亦或是总体方向这一类的东西。我想这也是为什么信息时代，掌握越多信息相当于拥有更多财富，因为越多的信息会使得做出正确决策的概率加大不少。我认为这也是后续元学习要实现的困难之一，要让一个学习机拥有人的思维，学会“变通”，学会“因地制宜”、“一个萝卜一个坑”是困难的。但我想这终究会有实现的手段，而一切也都归因于机器学习那简而美的学习之道，尽管道路曲折，但前途终究光明。

A 相关代码

基础算法：

kmeans

```
function [center res]=kmeans(data,N)%表示聚类类数N
[m n]=size(data);%为数据个数m,为数据维数n
variety=zeros(m,n+1);%新多出来一列表示数据属于哪一类
center=zeros(N,n);
variety(:,1:n)=data(:,:);
for k=1:N
center(k,:)=data(randi(m,1),:);
end
while 1
distance=zeros(1,N);
num=zeros(1,N);
new_center=zeros(N,n);

for i=1:m
for j=1:N
distance(j)=norm(data(i,:)-center(j,:));
end
[~,temp]=min(distance);%求最小距离
variety(i,n+1)=temp;
end

kn=0;%表示已经达到聚类要求的类数
for k=1:N
for i=1:m
if (variety(i,n+1)==k)
new_center(k,:)=new_center(k,:)+variety(i,1:n);
num(k)=num(k)+1;
end
end
new_center(k,:)=new_center(k,:)/num(k);
if (norm(new_center(k,:)-center(k,:))<0.1)
kn=kn+1;
end
```

```
end
if kn==N
break;
else
center=new_center;
end
end

res=variety;
```

k时=3主函数，训练集分类加测试集预测：kmeans

```
Num=600;
pi_real=[1/3,1/3,1/3];%设定真实值pi
mu_real=[-7,4;-2,-5;5,5];%设定真实期望
cov_real(:,:,1)=[1,0;0,1];
cov_real(:,:,2)=[3,1;1,3];
cov_real(:,:,3)=[3,1;1,3];%设定真实方差
X_1=mvnrnd(mu_real(1,:),cov_real(:,:,1),Num*
    pi_real(1));
X_2=mvnrnd(mu_real(2,:),cov_real(:,:,2),Num*
    pi_real(2));
X_3=mvnrnd(mu_real(3,:),cov_real(:,:,3),Num*
    pi_real(3));
%生成真实数据点集
X=[X_1;X_2;X_3];
X=X(randperm(size(X,1)),:);
maxiter=100;
x=-10:0.5:10;
y=-10:0.5:10;
figure(1);%绘出真实的数据点与置信椭圆
plot(X_1(:,1),X_1(:,2),'ro');
hold on;
plot(X_2(:,1),X_2(:,2),'b+');
hold on;
plot(X_3(:,1),X_3(:,2),'g<');
```

```
hold on;
confiEllipse(X_1,0.95);
hold on;
confiEllipse(X_2,0.95);
hold on;
confiEllipse(X_3,0.95);
hold on;
title('实际点结果');
figure(2);%绘出未分类的点集
plot(X(:,1),X(:,2),'ko');
title('未进行聚类时的点分布');
[center res]=kmeans(X,3);
[m,n]=size(res);
one=find(res(:,3)==1);%将点分类后给每一个点贴上标签
two=find(res(:,3)==2);
three=find(res(:,3)==3);
figure(3);
hold on;
for i=1:m%绘出分类完成的点集
if res(i,3)==1
plot(res(i,1),res(i,2),'ro');
elseif res(i,3)==2
plot(res(i,1),res(i,2),'b+');
else
plot(res(i,1),res(i,2),'g<');
end
end
for k=1:3
plot(center(k,1),center(k,2),'k+','linewidth',2)
    ;
end
confiEllipse(X(one,:),0.95);%绘制置信椭圆
hold on;
confiEllipse(X(two,:),0.95);
hold on;
confiEllipse(X(three,:),0.95);
hold on;
```

```
title('经过聚类后得到的结果kmeans');
z1=unifrnd(-10,10,[100,1]);%预测点集由均匀分布横纵坐
    标生成
z2=unifrnd(-10,10,[100,1]);
Z=zeros(size(z1,1),2);
Z(:,1)=z1;
Z(:,2)=z2;
dist=zeros(size(Z,1),size(center,1));%计算每个点与
    聚类中心的距离
for i=1:size(Z,1)
for k=1:3
dist(i,k)=norm(Z(i,:)-center(k,:));%计算相关的距离
    矩阵
end
end
pre=zeros(size(Z,1),1);%初始化预测结果
for i=1:size(Z,1)
minvalue=min(dist(i,:));
pre(i,1)=find(dist(i,:)==minvalue);
end
preone=find(pre(:,1)==1);%为预测的点集贴上标签并分类
pretwo=find(pre(:,1)==2);
prethree=find(pre(:,1)==3);
figure(4);
hold on;
for i=1:size(Z,1)%绘制预测点集的分布
if pre(i,1)==1
plot(Z(i,1),Z(i,2),'ro');
elseif pre(i,1)==2
plot(Z(i,1),Z(i,2),'b+');
else
plot(Z(i,1),Z(i,2),'g<');
end
end
hold on;
for k=1:3
```

```
plot(center(k,1),center(k,2),'k+','linewidth',2)
    ;
end
confiEllipse(Z(preone,:),0.95);
hold on;
confiEllipse(Z(pretwo,:),0.95);
hold on;
confiEllipse(Z(prethree,:),0.95);
hold on;
title('新生成的测试集预测结果(k情况=3)');
```

k=4,k情况类似，代码在=5k基础上改进即可=3绘制置信椭圆：

```
function confiEllipse(datamatrix,p)
%print 2-demension confidence ellipse
%In:×n2 matrix, confidence probability p

data = datamatrix;
covariance = cov(data);
[eigenvec,eigenval] = eig(covariance);

[sortEigenval,index] = sort(diag(eigenval),'
    descend');
sortEigenvec = eigenvec(:,index);

largestEigenval = sortEigenval(1);
smallestEigenval = sortEigenval(end); %求最小特征
    值
largestEigenvec = sortEigenvec(:,1); %求最大特征向
    量（椭圆的长轴）

angle = atan2(largestEigenvec(2),
    largestEigenvec(1));
%计算轴和最大特征向量之间的角度x, [-pi,pi]

if(angle < 0)
```

```
angle = angle + 2*pi;
end

avg = mean(data); %计算两列数据的均值

%计算置信椭圆的参数
chisquareVal = sqrt(chi2inv(p,2)); %卡方值
thetaGrid = linspace(0,2*pi);
phi = angle; %旋转角度
X0=avg(1);
Y0=avg(2);
a=chisquareVal*sqrt(largestEigenval); %轮距 长度
b=chisquareVal*sqrt(smallestEigenval);

ellipseXR = a*cos( thetaGrid ); %作用于直角坐标系
ellipseYR = b*sin( thetaGrid );

R = [ cos(phi) sin(phi); -sin(phi) cos(phi) ]; %
    旋转矩
    阵

rEllipse = [ellipseXR;ellipseYR]' * R; %旋转
plot(rEllipse(:,1) + X0,rEllipse(:,2) + Y0,'k-',
    'linewidth',1.5);
axis square
end高斯混合聚类算法:



N=600;
pi_real=[1/3,1/3,1/3];%设定真实值pi
mu_real=[-7,4;-2,-5;5,5];%设定真实期望
cov_real(:,:,1)=[1,0;0,1];
cov_real(:,:,2)=[3,1;1,3];
cov_real(:,:,3)=[3,1;1,3];%设定真实方差
```

```
X_1=mvnrnd(mu_real(1,:),cov_real(:,:,1),N*
    pi_real(1));
X_2=mvnrnd(mu_real(2,:),cov_real(:,:,2),N*
    pi_real(2));
X_3=mvnrnd(mu_real(3,:),cov_real(:,:,3),N*
    pi_real(3));
%生成真实数据点集
X=[X_1;X_2;X_3];
X=X(randperm(size(X,1)),:);
maxiter=100;
res_real=zeros(1,N);
res=zeros(1,N);
res_after=zeros(1,N);
for j=1:floor(N*pi_real(1))
res_real(1,j)=1;
end
for j=floor(N*pi_real(1))+1:floor(N*(pi_real(1)+
    pi_real(2)))
res_real(1,j)=2;
end
for j=floor(N*(pi_real(1)+pi_real(2)))+1:N
res_real(1,j)=3;
end
x=-10:0.5:10;
y=-10:0.5:10;
[x y]=meshgrid(x,y);
mesh=[x(:),y(:)];
nor_real=pi_real(1)*mvnpdf(mesh,mu_real(1,:),
    cov_real(:,:,1))+
pi_real(2)*mvnpdf(mesh,mu_real(2,:),cov_real
    (:,:,2))+
pi_real(3)*mvnpdf(mesh,mu_real(3,:),cov_real
    (:,:,3));
figure(1);
plot(X_1(:,1),X_1(:,2),'rx',X_2(:,1),X_2(:,2),'
    bo',
X_3(:,1),X_3(:,2),'g<');
```

```
title('混合高斯分布的真实分布');
hold on;
confiEllipse(X_1,0.95);
hold on;
confiEllipse(X_2,0.95);
hold on;
confiEllipse(X_3,0.95);
hold on;
figure(2);
contour(x,y,reshape(nor_real,size(x,2),size(y,2)
    ));
figure(3);
surf(x,y,reshape(nor_real,size(x,2),size(y,2)));
figure(4);
plot(X(:,1),X(:,2),'kx');
title('未经过算法的点EM');

pi=[1/3,1/3,1/3];%初始化的取值pi
cov(:,:,1)=[1,0;0,1];
cov(:,:,2)=[1,0;0,1];
cov(:,:,3)=[1,0;0,1];
mu_y_init=(max(X(:,1))+min(X(:,1)))/2;
mu_x1_init=max(X(:,2))/4+3*min(X(:,2))/4;
mu_x2_init=2*max(X(:,2))/4+2*min(X(:,2))/4;
mu_x3_init=3*max(X(:,2))/4+min(X(:,2))/4;
gamma=zeros(size(X,1),length(pi));
%gamma(i,j),是样本数量i,是聚类数量j,gamma(i,j)表示样
    本属于i聚类的概率，最大值表示为的聚类
ji
mu=[mu_x1_init,mu_y_init;mu_x2_init,mu_y_init;
    mu_x3_init,mu_y_init];

NMI=zeros(maxiter,1);

%算法：EM
iter=50;
for i=1:iter
```

```
for j=1:length(pi)
gamma(:,j)=pi(j)*mvnpdf(X,mu(j,:),cov(:,:,j));
end
gamma=gamma./repmat(sum(gamma,2),1,size(gamma,2)
    );
%建立和一样规模的矩阵，矩阵每一行
    为gammagamma(:,1)+gamma(:,2)值
pi=sum(gamma,1)./size(gamma,1);
mu=gamma'*X;
mu=mu./repmat((sum(gamma,1))',1,size(mu,2));
for j=1:length(pi)
vari=repmat(gamma(:,j),1,size(X,2)).*(X-repmat(
    mu(j,:),size(X,1),1));
cov(:,:,j)=(vari'*vari)/sum(gamma(:,j),1);
end
[c estimate]=max(gamma,[],2);

nor_iter=pi(1)*mvnpdf(mesh,mu(1,:),cov(:,:,1))+
pi(2)*mvnpdf(mesh,mu(2,:),cov(:,:,2))+
pi(3)*mvnpdf(mesh,mu(3,:),cov(:,:,3));
one_iter=find(estimate==1);
two_iter=find(estimate==2);
three_iter=find(estimate==3);
for j=1:size(one_iter,1)
res(1,j)=1;
end
for j=size(one_iter,1)+1:size(one_iter,1)+size(
    two_iter,1)
res(1,j)=2;
end
for j=size(one_iter,1)+size(two_iter,1)+1:N
res(1,j)=3;
end
NMI(i+1,1)=nmi_1(res,res_real);
end

%估计
```

```
[c estimate]=max(gamma,[],2);

nor=pi(1)*mvnpdf(mesh,mu(1,:),cov(:,:,1))+
pi(2)*mvnpdf(mesh,mu(2,:),cov(:,:,2))+
pi(3)*mvnpdf(mesh,mu(3,:),cov(:,:,3));
figure(5);
contour(x,y,reshape(nor,size(x,2),size(y,2)));

one=find(estimate==1);
two=find(estimate==2);
three=find(estimate==3);

for j=1:size(one,1)
res_after(1,j)=1;
end
for j=size(one,1)+1:size(one,1)+size(two,1)
res_after(1,j)=2;
end
for j=size(one,1)+size(two,1)+1:N
res_after(1,j)=3;
end

figure(6);
plot(X(one,1),X(one,2),'rx',X(two,1),X(two,2),'
    bo',
X(three,1),X(three,2),'g<');
title('经过算法后的点EM');
hold on;
confiEllipse(X_1,0.95);
hold on;
confiEllipse(X_2,0.95);
hold on;
confiEllipse(X_3,0.95);
hold on;预测的代码与上面类似
()层次聚类主函数:
```

```
z1=unifrnd(-5,5,[50,1]);
z2=unifrnd(-5,5,[50,1]);
Z=zeros(size(z1,1),2);
Z(:,1)=z1;
Z(:,2)=z2;
k=4;
%step1
[distance_matrix,cluster_set]= init_hierarchy2(Z
    );
%step2
cluster_result = hierarchy_clustering(
    distance_matrix,cluster_set,k);
figure(1);
show_cluster_tree(Z);
figure(2);
show(cluster_result);
k=5;
%初始化聚类集与距离矩阵step1
[distance_matrix,cluster_set]= init_hierarchy2(Z
    );
%得出相应的层次聚类结果step2
cluster_result = hierarchy_clustering(
    distance_matrix,cluster_set,k);
figure(3);
show2(cluster_result);
k=6;
%step1
[distance_matrix,cluster_set]= init_hierarchy2(Z
    );
%step2
```

```
cluster_result = hierarchy_clustering(
    distance_matrix,cluster_set,k);
figure(4);
show3(cluster_result);
k=7;
%step1
[distance_matrix,cluster_set]= init_hierarchy2(Z
    );
%step2
cluster_result = hierarchy_clustering(
    distance_matrix,cluster_set,k);
figure(5);
show4(cluster_result);
k=3;
%step1
[distance_matrix,cluster_set]= init_hierarchy2(Z
    );
%step2
cluster_result = hierarchy_clustering(
    distance_matrix,cluster_set,k);
figure(6);
show5(cluster_result);
k=2;
%step1
[distance_matrix,cluster_set]= init_hierarchy2(Z
    );
%step2
cluster_result = hierarchy_clustering(
    distance_matrix,cluster_set,k);
figure(7);
show6(cluster_result);初始化层次聚类:



function [dist_matrix,cluster_set]=
    init_hierarchy(Z
```

```
)%初始化层次聚类函数
cluster_set=[];
[m,~] =size(Z);%生成新的数据结构：索引数据点坐标+
dist_matrix = zeros(m,m);
for i=1:m%聚类的初始化
cluster.ind = i;
cluster.data = Z(i,:);
cluster_set = [cluster_set;cluster];
end
for i = 1:m%计算点集之间的距离矩阵
for j = i+1:m
dist_matrix(i,j) =get_dist(cluster_set(i),
    cluster_set(j));
dist_matrix(j,i) =dist_matrix(i,j);
end
end
end计算距离矩阵：
```

```
function dist =get_dist(cluster1,cluster2)
[m1,~]=size(cluster1.data);%2x2
dist = 10000;
for i =1:m1
tm_dist = pdist2(cluster2.data,cluster1.data(i
    ,:));    %计算两个聚类之间的距离，中每个点到的距
    离cluster1cluster2,并且取出其中的最小值

[min_value,~]=min(tm_dist);
if min_value < dist
dist = min_value;
end
end
end寻找最为接近的两个聚类：
```

```
function [Ci,Cj]= find_closest_cluster(
    dist_matrix)
[m,n] = size(dist_matrix);
min_dist = 10000;
start=2;Ci=0;Cj=0;
for i = 1:m-1%合并生成新的聚类并且更新聚类的索引值,
[tem_dist,min_ind] = min(dist_matrix(i,start:n))
    ;
if tem_dist<min_dist
min_dist=tem_dist;
Ci =i;
Cj = min_ind+start-1;
end
start = start+1;
end
if Ci>Cj
temp = Cj;
Cj =Ci;
Ci =temp;
end
end更新聚类集函数:
```

```
function cluster = update_cluster_set(Ci,Cj,
    cluster_set)
for i =Cj+1:length(cluster_set)
cluster_set(i).ind = cluster_set(i).ind -1;%合并
    并且更新新的聚类集
end
% 合并聚类簇序号小的, 合并序号大的  Ci < Cj
cluster_set(Ci).data = [cluster_set(Ci).data;
    cluster_set(Cj).data];
cluster_set(Cj)=[]; %删除冗余的已经被合并的聚类集
```

```
cluster = cluster_set;
```
end更新合并聚类集后的距离矩阵：

```
function dist_m = update_dist_matrix(Ci,Cj,
cluster_set,dist_matrix)%更新距离矩阵的函数
%step2.3
dist_matrix(:,Cj)=[];
dist_matrix(Cj,:)=[];
%updat 行,: Ci1cluster_set(end).ind 列
for j = 1:cluster_set(end).ind%在合并聚类集后重新计
    算距离矩阵的函数
dist_matrix(Ci,j) = get_dist(cluster_set(Ci),
    cluster_set(j));
dist_matrix(j,Ci) = dist_matrix(Ci,j);
end
dist_m = dist_matrix;
```
end绘出聚类结果图：（以分类为例，其他类数的以此类推类比即可）

4

```
function show(cluster_ret)%画出聚类图
plot(cluster_ret(1).data(:,1),cluster_ret(1).
    data(:,2),'ro');
hold on;
plot(cluster_ret(2).data(:,1),cluster_ret(2).
    data(:,2),'b.');
hold on;
plot(cluster_ret(3).data(:,1),cluster_ret(3).
    data(:,2),'g<');
hold on;
plot(cluster_ret(4).data(:,1),cluster_ret(4).
    data(:,2),'mx');
```

```
hold on;
xlabel('x-position');
ylabel('y-position');
end绘出层次聚类谱系图:
```

```
function show_cluster_tree(Z)%画出层次聚类的谱系图
Y=pdist(Z);%计算数据点之间的距离
H = linkage(Y,'single');%由最短距离的算法生成聚类树
dendrogram(H)%生成聚类谱系图
xlabel('x-position');
ylabel('y-position');
end
```