0、 运行源代码方法

运行 python source.py 即可, 功能包括构建数据集, 聚类以及处理前后的数据作图。

代码会作图 2 张。如果用控制台运行, 需要关闭第 1 张才能继续运行并显示第 2 张。

一、 数据集构建

使用标签 0 来代表类 A，标签 1 来代表类 B，标签 2 来代表类 C

分布设置：A、B 和 C 是三个 2 维高斯分布。三者的 x1 和 x2 都是相互独立的，其协方差矩阵为([1,0],[0,1])。A 的均值为[1,3]，B 的均值为[0,0]，C 的则是[4,1]。

采样设置：利用 numpy 自带的函数 multivariate_normal，对三个高斯分布分别采样 1000 个，再使用 numpy 自带的函数 vstack 进行合并，成为一个 3000 个数据的数据集，然后再用 numpy 自带的函数 shuffle 打乱顺序，成为一个混合数据集。

二、 聚类

首先构造出混合高斯模型类 myGMMmodel。

初始化参数包括高斯分布的个数 k，高斯分布的维度 D，以及数据集大小 n。

在此处，k=3，D=2，n=3000。

需要优化的参数有：

$\boldsymbol{\pi}$：长度为 k（k 为分布的个数）的向量，$\boldsymbol{\pi_k}$表示样本属于第 k 个分布的概率

$\boldsymbol{\mu}$：大小为 k×(1×D)的矩阵，$\boldsymbol{\mu_k}$表示第 k 个分布的均值

$\boldsymbol{\sigma^2}$：大小为 k×(D×D)的矩阵，$\boldsymbol{\sigma^2}_{\boldsymbol{k}}$表示第 k 个分布的协方差矩阵

为了优化这 3 个参数，还需要一个参数γ。

$\boldsymbol{\gamma}$：大小为 n×k 的矩阵。$\boldsymbol{\gamma_{nk}}$表示第 n 个样本属于第 k 个分布的后验概率。

优化参数的步骤可以分为 2 步：E 步和 M 步。

E步　先固定参数μ,σ，计算后验分布$p(z^{(n)}|x^{(n)})$，即

$$\begin{aligned}\gamma_{nk} &\triangleq p(z^{(n)}=k|x^{(n)}) \\ &= \frac{p(z^{(n)})p(x^{(n)}|z^{(n)})}{p(x^{(n)})} \\ &= \frac{\pi_k \mathcal{N}(x^{(n)};\mu_k,\sigma_k)}{\sum_{k=1}^{K}\pi_k \mathcal{N}(x^{(n)};\mu_k,\sigma_k)},\end{aligned}$$

其中γ_{nk}定义了样本$x^{(n)}$属于第k个高斯分布的后验概率.

因此，将 E 步定义为函数 E-Step：

```
def E_Step(self,x,pi,sigma,miu):
    k = self.k
    D = self.D
    n = x.shape[0]
    gamma = np.zeros((n, k))
    N = np.zeros(k)
    for p in range(n):
        for q in range(k):
            N[q] = pi[q] * st.multivariate_normal.pdf(x[p], miu[q], sigma[q])
        sumN = np.sum(N)
        for q in range(k):
            gamma[p, q] = N[q] / sumN
    return gamma
```

对于多维高斯分布的概率密度函数，可以利用 scipy 库中的 scipy.stat，其中的函数 multivariate_normal.pdf 能直接获得密度函数。

M 步则用于优化$\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\sigma^2}$三个参数。

$$\begin{aligned}\pi_k &= \frac{N_k}{N}, \\ \mu_k &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma_{nk}x^{(n)}, \\ \sigma_k^2 &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma_{nk}(x^{(n)}-\mu_k)^2,\end{aligned}$$

其中

$$N_k = \sum_{n=1}^{N}\gamma_{nk}.$$

不过，这是一维的表示形式，如果到二维：

对于$\boldsymbol{\mu}$的优化有：

```
for i in range(k):
    miu[i] = 0
    for j in range(n):
        miu[i] += (gamma[j , i] * x[j])
    miu[i] = miu[i] / Nk[i]
```

对于$\boldsymbol{\sigma^2}$则可以使用下面的表示形式

$$\sigma_k^2 = \frac{\sum_{i=1}^{n} \gamma_{ik}(x_i - \mu_k) \cdot (x_i - \mu_k)^T}{N_k}$$

```
for j in range(k):
    sigma[j] = np.zeros((D,D))
    for i in range(n):
        sigma[j] += gamma[i, j] * np.dot((x[i]-miu[j]).reshape(D,1), ((x[i]-miu[j]).reshape(D,1)).T)
    sigma[j] = sigma[j] / Nk[j]
```

接下来则是要定义训练函数。

训练的主体部分是轮流进行 E 步和 M 步，反复更新参数，以优化变量。

训练之前，需要进行对变量的初始化:

$\boldsymbol{\pi}$：长度为 k 的向量，由于各项之和为 1，不妨将每一项设为 1/k。

$\boldsymbol{\mu}$：大小为 k×(1×D)的矩阵，代表 3 个均值。从所有数据里随机取 3 个点即可。这样可以避免设置随机数范围，也防止直接随机出来的数过大或过小，影响效果。

$\boldsymbol{\sigma^2}$：大小为 k×(D×D)的矩阵，代表 3 个协方差矩阵。初始化为 3 个单位阵$\boldsymbol{I}$。

初始化之后，反复执行 E 步和 M 步来优化变量。目前 epoch 数设置为 50。

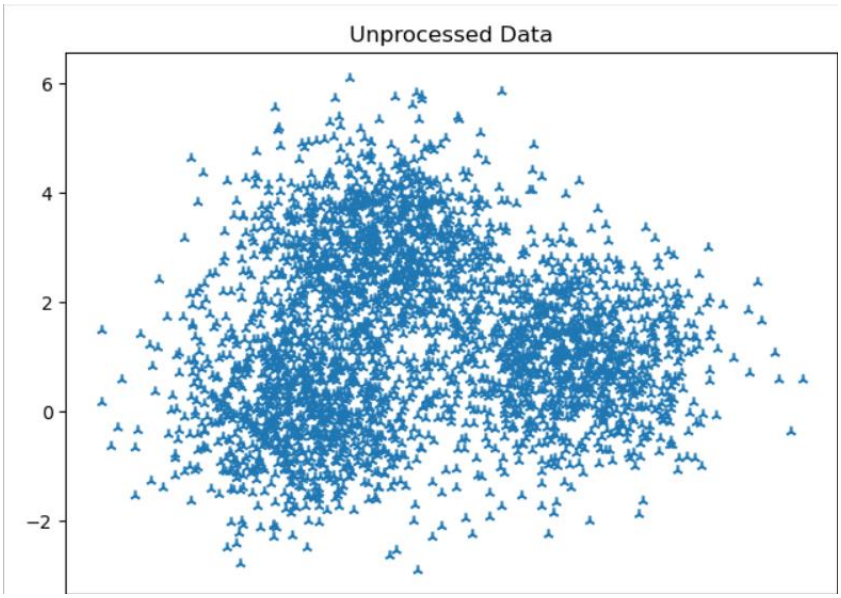
三、 结果

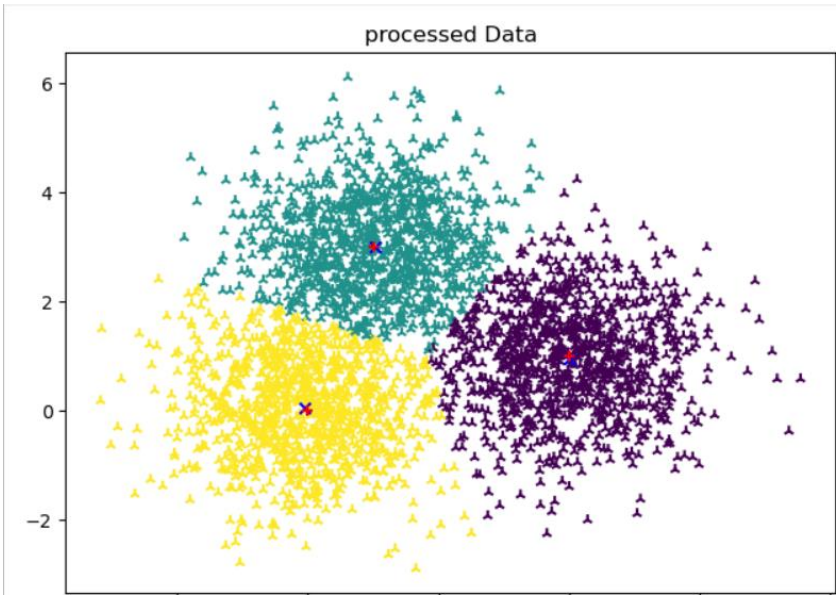
在 50 个 epoch 跑完之后，此时的$\boldsymbol{\gamma}$也就是后验概率，代表着第 n 个样本属于第 k 个分布的后验概率。因此，对于每一个样本，都有一个$\boldsymbol{\gamma}$向量，分别代表该样本属于三个分布的概率。以这个概率为指标，利用 argmax 函数将样本归到概率最大那一类。

然后就可以按照类别，重新做一张图。关于分类别给点上色的写法，比较新奇，这个是从其他同学处讨教而来。

另外，也可以标出当前的均值$\boldsymbol{\mu}$和真实值在图中的位置，（图中的×标记）用来

观察模型的效果。两图对比大致如下:





总体感觉还行。

四、 进一步的讨论

在完成了基本的聚类后，可以进行一些探索。

1、关于初始化的讨论

目前使用的初始化方案是这样的:

$\boldsymbol{\pi}$：长度为 k 的向量，由于各项之和为 1，不妨将每一项设为 1/k。↵

$\boldsymbol{\mu}$：大小为 k×(1×D)的矩阵，代表 3 个均值。从所有数据里随机取 3 个点即可。这样可以避免设置随机数范围，也防止直接随机出来的数过大或过小，影响效果。↵

$\boldsymbol{\sigma^2}$：大小为 k×(D×D)的矩阵，代表 3 个协方差矩阵。初始化为 3 个单位阵$\boldsymbol{I}$。↵

对于$\boldsymbol{\sigma^2}$，最开始的时候，我使用了 3 个随机矩阵来进行初始化，发生了报错。后来发现，协方差矩阵具有非负定的性质，随机数是不行的。

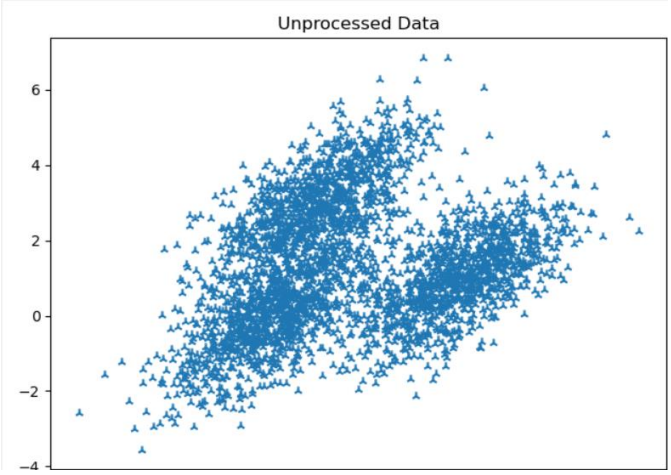
然后，我使用了全 1 矩阵来进行初始化，发生了报错，报错的结果是“singular matrix”，即奇异矩阵。意为协方差矩阵不满秩。尽管协方差矩阵并不一定要满秩，但为了防止这个错误，最后将对于$\boldsymbol{\sigma^2}$的初始化修改为 3 个单位阵，确保满足上面两个要求。

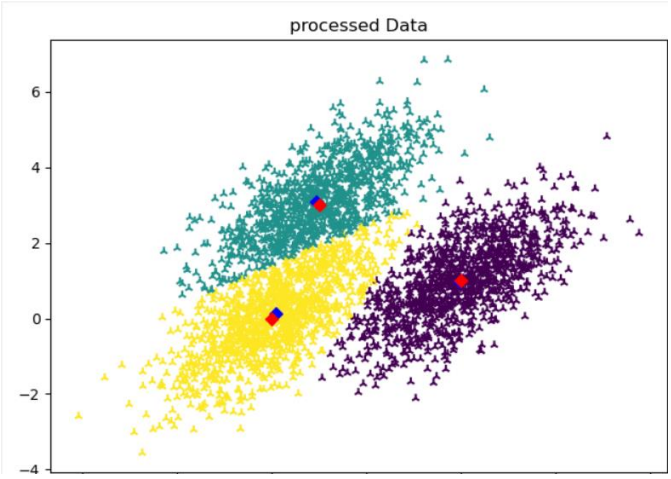
对于$\boldsymbol{\mu}$，我还尝试过将三个均值都初始化为整个数据集的平均值的初始化方式。后来发现，这样操作会使得训练失败，所有的点被聚为一类，因此最后舍弃了这种初始化方式。

2、不同协方差的数据

之前的采样中，协方差矩阵为单位阵。考虑到初始化的时候也使用了单位阵，这可能存在“取巧”的意味。因此，可以通过设置不同的协方差来观察模型的效果。

下图中，协方差矩阵为$\begin{bmatrix} 1.2 & 0.8 \\ 0.8 & 1.2 \end{bmatrix}$。

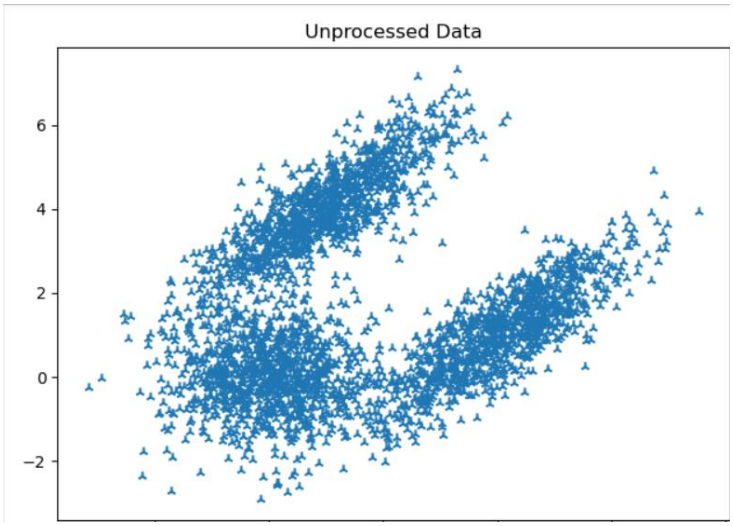


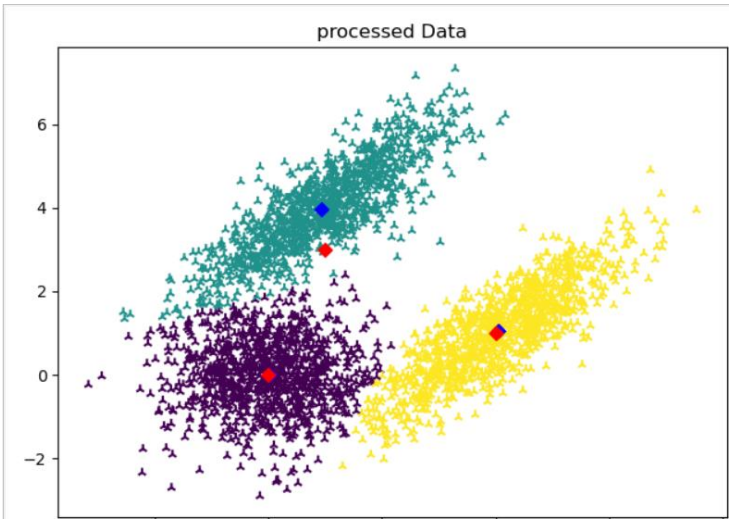


模型的效果稍微下降了，不过对于均值的估计还是较为准确的。

不过，这里仍然是三个同分布的模型。可以尝试令三个高斯分布的分布不同。

下图中，三个分布的协方差矩阵不同。





可以看到，对于均值的估计也出现了一定偏差。不过由于交叉部分不大，所以分类还是分得比较开，但是对于均值的估计并不那么准确。

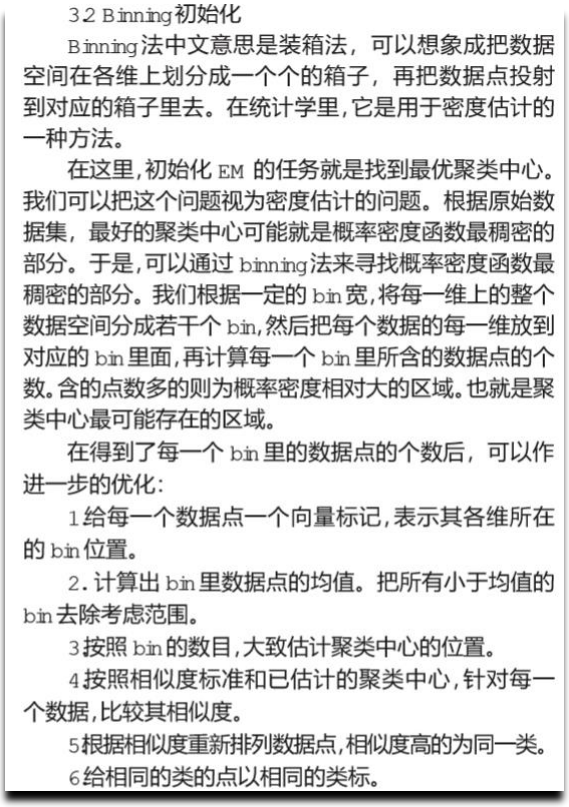
总结上面的分析可以知道，对于 GMM 而言，初始化非常的重要。在当前的初始化方法下，在某些情况下，也会偶尔收敛到非常奇怪的点上，不过总体性能比较好。如果使用了并不那么恰当的初始化，可能会影响模型的正确性。因此，审慎地选择初始化方法是很有必要的。

3、其他的初始化办法

因此，就初始化方法和舍友进行了讨论。在讨论过程中，我们发现一篇论文是专门讲这个的：《高斯混合模型聚类中 EM 算法及初始化的研究》。

在文章中提到了，可以用 kmeans 来做初始化。Kmeans 本身也是一种聚类的方法。用 kmeans 先做一个简单的聚类，然后以其结果为初始化，也是一种不错的办法。

在文章中还额外提到了一种初始化办法：Binning。



3.2 Binning初始化

Binning法中文意思是装箱法，可以想象成把数据空间在各维上划分成一个个的箱子，再把数据点投射到对应的箱子里去。在统计学里，它是用于密度估计的一种方法。

在这里，初始化 EM 的任务就是找到最优聚类中心。我们可以把这个问题视为密度估计的问题。根据原始数据集，最好的聚类中心可能就是概率密度函数最稠密的部分。于是，可以通过 binning法来寻找概率密度函数最稠密的部分。我们根据一定的 bin 宽，将每一维上的整个数据空间分成若干个 bin，然后把每个数据的每一维放到对应的 bin 里面，再计算每一个 bin 里所含的数据点的个数。含的点数多的则为概率密度相对大的区域。也就是聚类中心最可能存在的区域。

在得到了每一个 bin 里的数据点的个数后，可以作进一步的优化：

1 给每一个数据点一个向量标记，表示其各维所在的 bin 位置。

2. 计算出 bin 里数据点的均值。把所有小于均值的 bin 去除考虑范围。

3 按照 bin 的数目，大致估计聚类中心的位置。

4 按照相似度标准和已估计的聚类中心，针对每一个数据，比较其相似度。

5 根据相似度重新排列数据点，相似度高的为同一类。

6 给相同的类的点以相同的类标。

这些办法都给初始化提供了一个不错的思路。不过时间有限，就并没有进行代码的测试。如果有时间，就可以考虑测试一下。