Assignment 3

Part I

第一部分构造 N 个无标签的聚类数据集，可选择构造 K 个二维高斯模型做成的高斯混合模型，默认随机生成每个高斯函数的被选取概率，也支持手动输入。

实际生成数据的过程就模拟选择的过程：每次根据已有概率选择 K 个模型之一，生成服从该模型的数据点，重复上述操作，得到待聚类的无标签二维高斯混合模型数据集。其中不支持命令行指定各模型的参数（均值向量和协方差矩阵）

具体实现在 GMM.py generate_data(N, K, p, para)中。

Part II

模型描述：

定义 GMM 类，其中包括三个部分：

类初始化，必要的输入为待分类的数据集 Data，类别数量 K，其余是可选择的参数：先验的权重向量，即对应每个模型被选择的概率；初始状态下 K 个模型对应的参数，分别用对应的矩阵存储输入。初始化时，存储数据集以及类别数，根据 Data 获取 dim，同时对于空缺值或者不合法的输入进行如下处理：默认条件下，假设每个模型被选择的概率相同，即 1/K；模型的参数直接使用随机值，生成并且记录 K * dim 的均值矩阵，K * dim * dim 的协方差矩阵。

计算多维高斯分布的概率，输入为一个数据点以及一个模型的参数，输出为该数据点在该模型上的概率，内部直接代用多维度的公式，定义在函数 def Gaussian(self, x, mean, cov)。

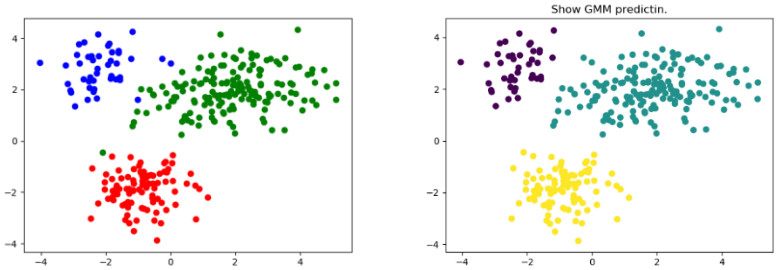
模型的主体部分，就是 E-M 算法实现的部分，也如算法所示存在两个部分，通过迭代求取参数的极大似然估计：

E 步：计算每一个变量 x 属于每一个高斯变量的后验概率，即期望。

M 步：根据 E 步的概率，计算更新参数 weight，mean，cov

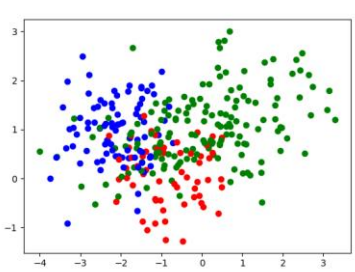
理论上每一次迭代都会增大近似极大化的最大似然，设置阈值作为输入，当增长的数值低于阈值则判断近似达到训练效果，然后根据此时得到的概率，选择最大概率的高斯分布对数据集完成聚类，存储在 pred 之中。

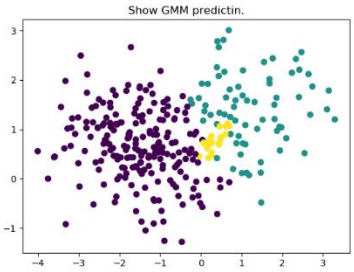
模型表现：



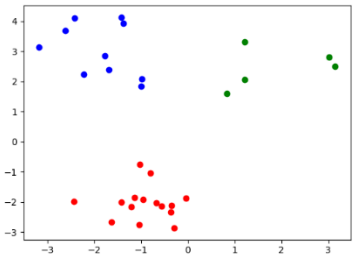
和有监督的的聚类、分类问题一样，对于有区别度较大的高斯分布生成而且数据充足的数据集聚类效果比较优秀，上图则是对于 300 个数据点，给定区分度较大的三个高斯分布以差距不大的概率选取生成的数据进行聚类的结果，虽然不是完全正确，但是就聚类而言表现比原图更加大的类间区分度和类内相似度。而且仅需要少量的迭代次数就能达到最终收敛。

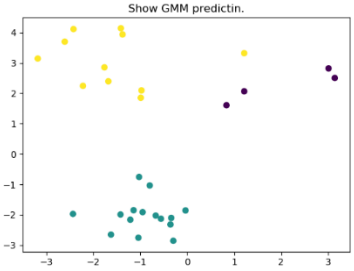
将本次数据集作为对照组，进行不同条件下的尝试比较。

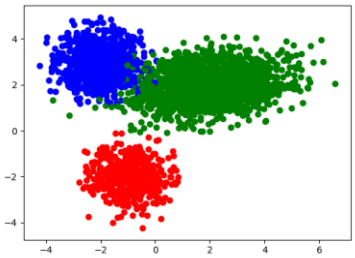


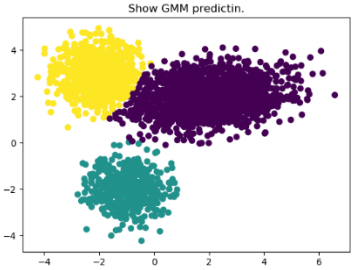


与之相比，使用区分度低的数据集，其他条件和上一数据集保持一致，但是得到的聚类结果和原分类相差比较大，但是可以接受，生成的概率以及各高斯分布的参数显然与原分布有较大区别，而且在相同阈值下进行了 600 多次迭代，是上一数据集的将近 20 倍。

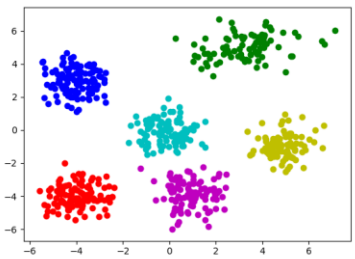


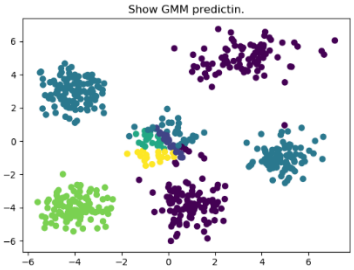


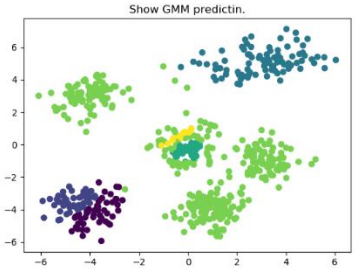




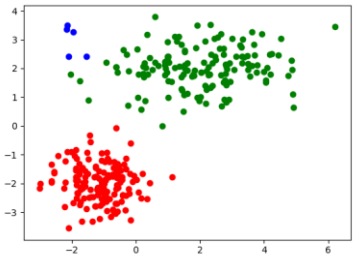
测试小规模和大规模的数据集，分别在相同情况下，取测试集 1/10 和 10 倍大小的数据集， 分别如上图所示，虽然改变了数据的规模，但是依然能够较为优秀的进行聚类，虽然数据集的增加会带来更加准确的聚类，但是提升不大，而且 300 和 3000 的数据聚类基本效果差不多，而且均在大约 60 次迭代后达到收敛（30 的数据集只花了 20 次迭代）。

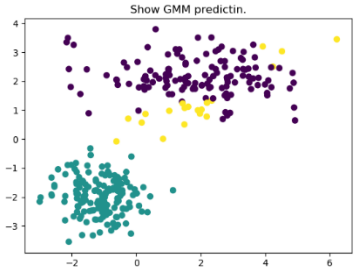


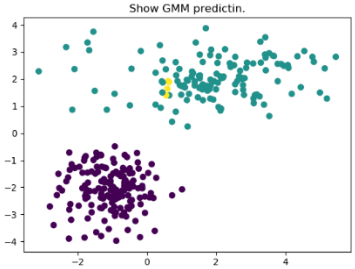




测试多类别聚类，大致保持每一类的数量和对照集相一致，最终使用 600 组数据进行 6 类聚类，最终进行了 600 次以上的迭代，但是虽然部分类别被较为正确的聚类，但是也有部分表现不理想，而且即使修改了阈值进行更高次数的迭代，也不能改进该结果。尝试使用对应的参数去进行初始化我的模型，也得不到合适的分类





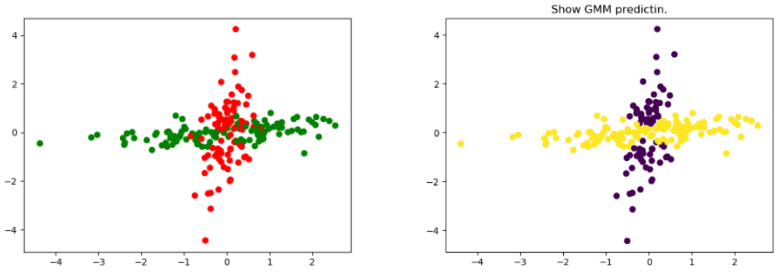


测试有偏的数据集，对于对照集采用 0.49，0.49，0.02 的概率生成数据集，生成概率较高的两个分布能够较好地进行聚类，但是数据量小的分布就不能进行有效的聚类，而且需要的迭代次数达到将近 1000。后选择对应的概率作为初始的模型参数，能够在 60 次迭代之内达到收敛，和对照集相类，而且分类效果要好于之前。

实验结果分析：

由于对于 GMM 的 EM 算法，主要考察的是每一个点属于不同高斯模型的概率，然后最终生成合适的高斯模型使得参数总的似然估计最大。所以对于具有明前区分度的模型即使使用随机参数，也能快速找到极大最优的参数，每个分布使用少量数据就能在不高的迭代下达到收敛，所以基本不受数据规模的影响，都能达到理想的聚类效果。

对于相重叠的类，重叠部分相当于属于各个分类的概率相差不大，本来就难以进行区分，所以虽然能够处理存在重叠的聚类结果，但是要求生成分布的协方差有一定的区分度，否则也是不能进行区分，如下图所示：



但是对于 EM 算法，其推导公式为：log p(x; θ) = ELBO(q, x; θ) + KL(q(z)||p(z|x; θ))，可以做如下推理：

$L(\theta) = \log p(x;\theta) = \log \sum_z p(x,z;\theta) = \log \sum_z p(x|z;\theta)p(z;\theta)$

对于已经迭代了 i 次的模型，在 M 步还要得出一个更大的似然估计，即表示为求 $L(\theta^{(i+1)}) \geq L(\theta^{(i)})$，又有如下结果：

$L(\theta^{(i+1)}) \geq ELBO(q,x;\theta^{(i)})$可以得到结论就是上述两个函数看作关于θ的函数的话，$L(\theta)$总在$ELBO(q,x;\theta)$上方，每次 E 步的时候总时更新θ的值使得$ELBO(q,x;\theta)$的新值与原来$L(\theta)$相等。于是，如果$L(\theta)$存在一点与$ELBO(q,x;\theta)$相等，但是不是最优值，于是很可能在在该点得到局部最优解，而不是全局最优，导致最终的聚类出现问题。推测在上述使用六类进行聚类的过程中就遇到了这种情况，由于该方法考察的是概率，而不关心每一类中点的关系与位置，所以可能发生这样的情况：原本的一个类被划分为多个分类函数生成，或者多个类被认为是相同的分类函数生成。

对于有偏的数据，主要是小概率数据可能应为出现太少，导致模型检测时直接忽略，作为一个其他分布的一部分，如果数据量不足容易导致无法近似得到对应的参数，就会导致聚类上的错误。

而使用先验知识确定概率以及参数，一定程度上可以优化聚类结果，但是在更多情况下是可以优化迭代次数，因为获得的参数已经近似在最优解附近，所以可以避免初始状态的多次尝试，直接进入最后收敛的阶段。而且先验知识的使用一定程度上可以避免局部最优解，而最初的概率分布的先验则可以有效帮助模型确定各个聚类的数据量。

运行指令：

运行默认的聚类：python source.py

可选参数：

- -k 整数，用来确定聚类的数量
- -n 整数，用来确定数据集的总数量
- -p bool，如果为真，则后续提示为为一个模型输入一个概率（后续会自动进行归一化处理）