Assignment 3 报告

1. 模型描述

1.1 问题提出　我们的数据集是由若干个高斯分布生成的，但是数据集中没有每个数据点对应的高斯分布的标签，因此我们只知道数据的边际似然函数：

$$p(\boldsymbol{x}|\theta) = \sum_{z} p(\boldsymbol{x}, z|\theta)$$

其中 z 表示数据点 $\boldsymbol{x}$ 的原始标签。我们的目标是要学习到 z 的信息。

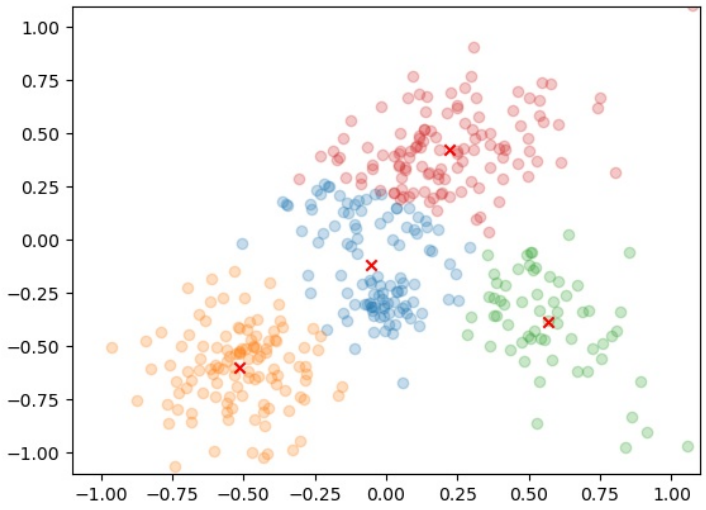
在后面的测试中，我们将使用四个随机生成的高斯分布作为测试数据。高斯分布的参数可以查看 source.py 中 PRIORS、MEANS、COVS 三个变量的配置。

1.2 K-means 算法　给定超参数 K 表示数据中估计的聚类数量。初始时选定数据集中随机的 K 个点作为 K 个聚类的中心，之后反复迭代以下过程来获得一个大致的划分：

1. 将数据集中每个点划分到离它最近的聚类中心的聚类中。
2. 对每个聚类，将该聚类内所有点的重心作为新的中心。

下图是一个 K-means 算法的示例，可以使用以下命令行来运行：

```
python source.py run fake.data -k 100 -e 0 -K 4 -p
```



可以看出 K-means 算法大致将数据集分成了合理的四部份，但是因为其边界是线性的，对于多个高斯分布的情况，不难发现 K-means 算法给出的结果不能很好地处理高斯边界的划分。

1.3 EM 算法　引入变分函数 $q(\boldsymbol{z})$，定义证据下界

$$\mathcal{L}(q, \boldsymbol{x}, \theta) = \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z}|\theta)}{q(\boldsymbol{z})}$$

它是 $\ln p(\boldsymbol{x}|\theta)$ 由 Jensen 不等式给出的下界。从信息论的角度可以证明，$\ln p(\boldsymbol{x}|\theta) \geqslant \mathcal{L}(q, \boldsymbol{x}, \theta)$，并且只在 $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x}, \theta)$ 时取等。

EM 算法的过程分为两步 [1]：

1. 用 θ 计算出 $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x}, \theta)$。使得 $\ln p(\boldsymbol{x}|\theta) = \mathcal{L}(q, \boldsymbol{x}, \theta)$。
2. 计算 $\theta' = \arg\max\limits_{\theta} \mathcal{L}(q, \boldsymbol{x}, \theta)$。

1.4 GMM 模型　接下我们建立数据的多重高斯分布模型。首先高斯分布的概率密度函数的形式如下：

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\intercal}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right)$$

现假设有 n 个样本点，m 个分类（聚类）。引入隐变量 z，假设其服从多项分布，并且多项分布的参数为 $\boldsymbol{\pi}$。于是：

$$p(\boldsymbol{x}_i|z = j, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

由此可以计算出对应的后验概率：

$$p(z = j|\boldsymbol{x}_i) = \frac{\pi_j p(\boldsymbol{x}_i|z = j)}{\sum_{k=1}^{m} \pi_k p(\boldsymbol{x}_i|z = k)} =: \gamma_{ij}$$

利用之前提到的 EM 算法，我们计算证据下界：

$$\mathcal{L} = \sum_{i=1}^{n}\sum_{j=1}^{m} \gamma_{ij} \ln \frac{\pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\gamma_{ij}}$$

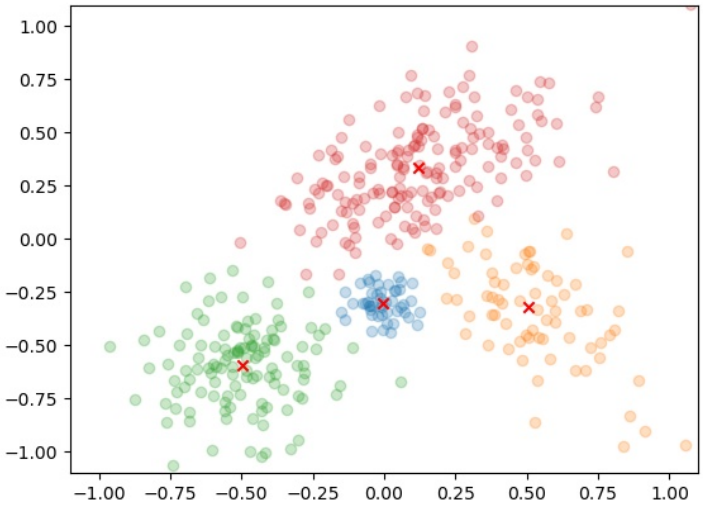
不难发现，不同聚类的参数可以分开优化。对上式求梯度后可以求得极值点。令 $n_j = \sum_{i=1}^{n} \gamma_{ij}$，则有 [2]：

$$\begin{aligned}
\boldsymbol{\mu}_j' &= \frac{1}{n_j}\sum_{i=1}^{n} \gamma_{ij}\boldsymbol{x}_i \\
\boldsymbol{\Sigma}_j' &= \frac{1}{n_j}\sum_{i=1}^{n} \gamma_{ij}(\boldsymbol{x}_i - \boldsymbol{\mu}_j')(\boldsymbol{x}_i - \boldsymbol{\mu}_j')^{\intercal} \\
\pi_j' &= \frac{n_j}{n}
\end{aligned}$$

使用 EM 算法反复迭代直至收敛即可。

2. 测试结果

下图是使用 K-means 算法做初始化，EM 算法做后续迭代得到的结果。其中 K-means 算法进行了 10 次迭代，EM 算法进行了 100 次迭代。



使用以下命令行运行：

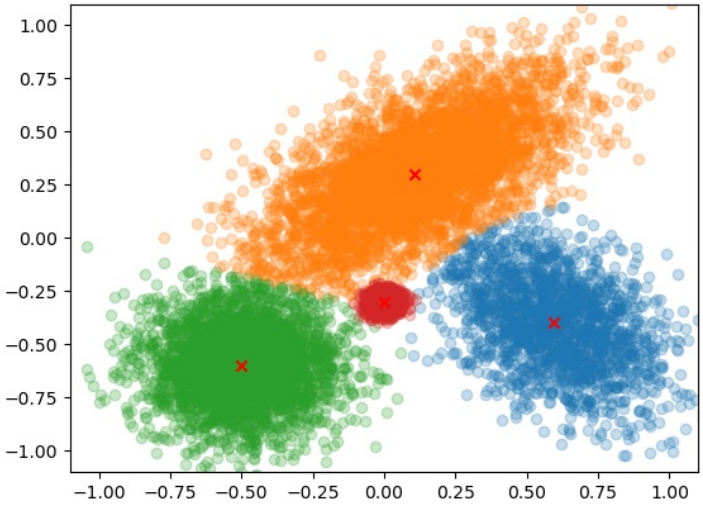
```
python source.py run fake.data -k 10 -e 100 -K 4 -p
```

算法最终得到的聚类中心为：

```
(-0.00547330, -0.29970292)
( 0.50585871, -0.31761383)
(-0.49717926, -0.59399257)
( 0.11859225,  0.33466699)
```

可以发现算法给出的结果还是比较接近原始的高斯分布中心的。

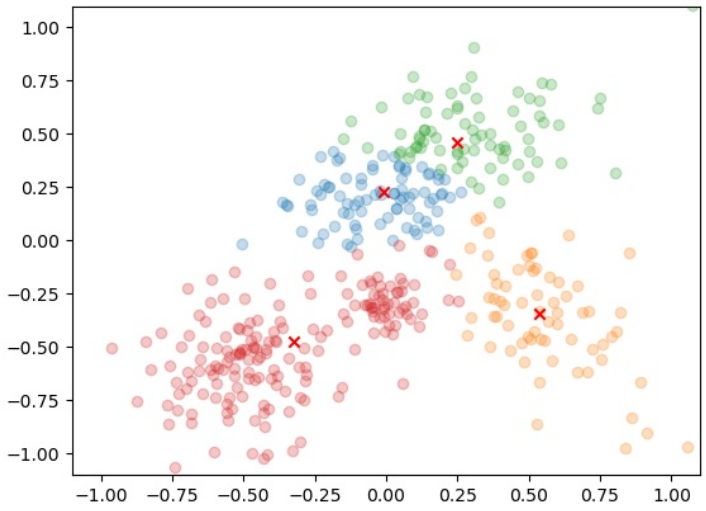
将小数据换成大数据 large.data [3] 的结果如下：



3. 模型分析

从上一节小数据时两个算法结果的对比可以看出，基于 GMM 模型的 EM 算法能够非常准确地划分出每个高斯分布的区域，而 K-means 算法会存在较大误差。这种误差主要体现在聚类的边界上，K-means 算法只能给出线性的决策边界，而 GMM 模型能够给出更合理的边界。得益于生成式模型的优势，EM 算法得到的结果的边界实际上不是硬性的，而是可以得到每个数据点属于每个聚类的确信程度。

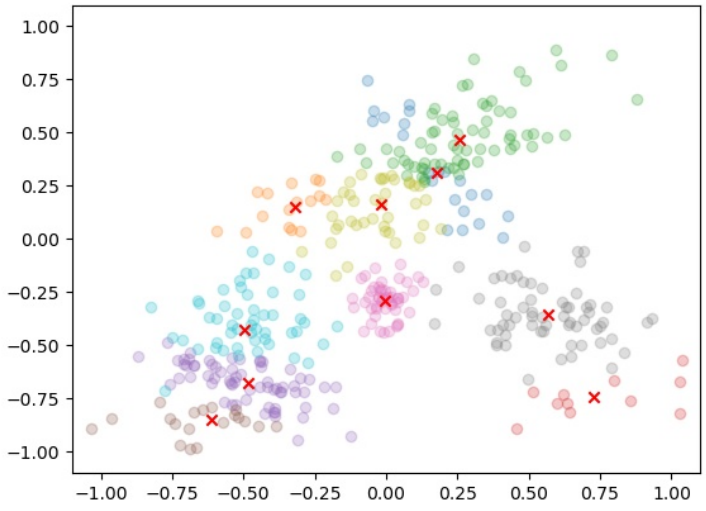
3.1 算法稳定性 但 EM 算法也有一定的缺陷。当数据集较小时，如果不使用 K-means 算法做初始话，EM 算法很容易收敛到一个不优的结果，如下图所示：



使用 K-means 算法做初始化可以提高算法的健壮性。在保持算法总迭代次数（K-means 算法初始化的迭代次数加上 EM 算法的迭代次数）不变的情况下，最终结果的平均正确率如下表所示（取 20 次运行结果的平均值）：

数据集	fake.data	large.data
不使用 K-means	81.61%	91.55%
使用 K-means	96.75%	97.55%

3.2 超参数 K 的影响 如果数据中聚类数量 K 未知，那么我们需要通过尝试不同的参数 K 来寻找最优的参数。下图展示了当设定 K 比原始聚类数量多的时候的情况：



命令行：

```
python source.py run fake.data -k 1 -e 99 -K 10 -p
```

此时算法将无法给出原始聚类中心的预测，但是可以看出预测的中心会倾向于分布在原始高斯分布中心的周围，因此可以考虑利用这一点大致判断出原始数据中有几个分布。

4. 数据生成

使用以下命令行可以生成一份包含 1000 个数据点的数据，保存到文件 test.data 中。同时会生成另外一个文件 test.data.label，保存的是生成数据时每个数据点的真实标签。

```
python generate -n 1000 -o test.data
```

5. 参考资料

1. "含隐变量的参数估计"，《神经网络与深度学习》，邱锡鹏，https://nndl.github.io/。
2. "Pattern Recognition and Machine Learning," Christopher M. Bishop.
3. 文件 large.data，https://riteme.site/large.data.