机器学习实验报告(二) 聚类

姓名： 袁子麒 班级：2017211301 学号：2017211594

（1）实验目标

基础要求使用 GMM 及 EM 算法，尝试使用不同个数的混合成分。对不同的高斯分布，尝试使用关联的协方差 矩阵和独立的协方差矩阵，分析聚类结果。

提高要求 按照 8:2 的比例，随机将数据分为训练集和测试集; 对于不同个数的混合成分， 绘制随着迭代的 进行，模型在训练集和测试集上的似然，并对结果进行讨论。

（2）算法原理

首先，我们定义高斯混合分布

$$p_M(x) = \sum_{i=1}^{k} \alpha_i \cdot p(x|\mu_i, \Sigma_i) \tag{1}$$

该分布由 k 个混合成分构成，每个混合成分对应一个高斯分布，其中 μ_i, Σ_i 是第 i 个高斯混合成分的参数，而 α_i 为相应的 "混合系数" $\sum_{i=1}^{k} \alpha_i = 1$。GMM 模型假设样本的生成过程由高斯混合分布生成：首先根据 $\alpha_1, \cdots, \alpha_k$ 定义的先验分布选择高斯混合成分，然后根据被选择的混合成分的概率密度函数进行采样，从而生成相应的样本。

从原型聚类的角度来看，高斯混合聚类是采用概率模型（高斯分布）对原型进行刻画，簇划分则由原型对应的后验概率确定。模型的本质就是求一组最合适的高斯混合分布参数 $\{(\alpha_i, \mu_i, \Sigma_i)|1 \le i \le k\}$ 。容易想到，可以使用极大似然估计，即最大化（对数）似然函数：

$$LL(D) = \ln(\Pi_{j=1}^{m} P_M(x_j)) = \sum_{j=1}^{m} \ln(\sum_{i=1}^{k} \alpha_i \cdot p(x_j|\, u_i, \Sigma_i)) \tag{2}$$

常采用 EM 算法进行迭代优化求解，若参数 $\{(\alpha_i, \mu_i, \Sigma_i)|1 \le i \le k\}$ 能使 (2) 最大化，则 $\frac{\partial LL(D)}{\partial \mu_i} = 0$, $\frac{\partial LL(D)}{\partial \Sigma_i} = 0$, 由此可以推得 （参考推导）

$$\mu_i = \frac{\sum_{j=1}^{m} \gamma_{ji} x_j}{\sum_{j=1}^{m} \gamma_{ji}} \tag{3}$$

$$\Sigma_i = \frac{\sum_{j=1}^{m} \gamma_{ji} (x_j - \mu_j)(x_j - \mu_j)^T}{\sum_{j=1}^{m} \gamma_{ji}} \tag{4}$$

对于混合系数 α_i 除了最大化 $LL(D)$ 还学要哦满足 $\alpha_i \ge 0, \sum_{i=1}^{k} \alpha_i = 1$,由此推得其更新公式为

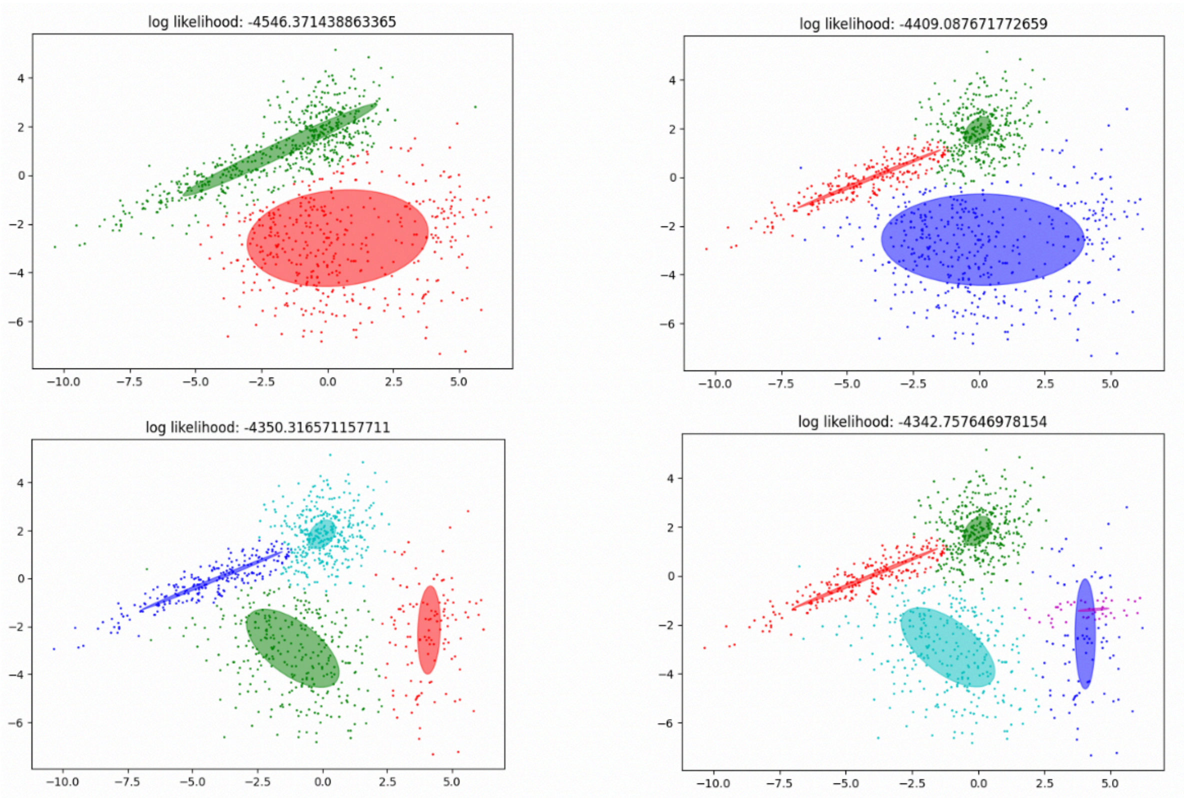
$$\alpha_i = \frac{1}{m} \sum_{j=1}^{m} \gamma_{ji} \tag{5}$$

基于以上公式，已经 EM 算法，我们可以得到我们需要的算法流程图。(算法流程图见西瓜书P210,为了留篇幅写结果分析，这里省略了)

（3）实验结果分析

（3.1）基础实验

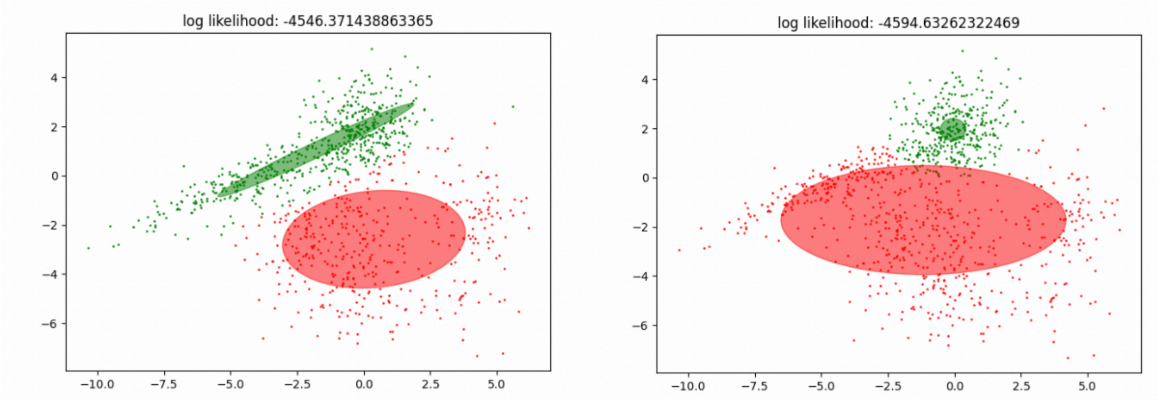
- 尝试使用不同个数的混合成分。
    - 我们这里分别在 type = “full”的情况下进行了 k = 2， 3， 4，5 的实验，实验结果如下：
    - 我们可以看到随着components数量的增加，模型的log likelihood 逐渐提升。原因是，components的增加导致了模型参数的增加，赋予了模型更好的表述能力，故此模型效果应该有所提升。



- 尝试使用关联的协方差矩阵和独立的协方差矩阵，分析聚类结果。

  本次实验中我们对比了 $k=2$ 时，独立的协方差矩阵（假设协方差矩阵为对角阵）和关联的协方差矩阵（协方差矩阵没有假设）之间的结果以及效果上的差别。

  实验效果图展示如下：(左侧为独立情况，右侧为关联情况)



- 两者之间比较显著的差别首先在于其椭圆的形状（是否发生倾斜）。 对于 **独立的协方差矩阵**（假设协方差矩阵为对角阵）来说，由于 “独立假设”，导致两个随机变量之间不存在相关性，表现在其椭圆是没有倾斜的。对于**关联的协方差矩阵** （协方差矩阵没有假设），二维随机变量之间存在了线形相关，这种相互的影响关系表现在椭圆的倾斜上面。
- 同时，我们可以看到，由于独立依赖的存在，模型相较于无独立依赖的情况少了 2 (n_components) * (2*2-2) $(n\_features * n\_feautres - n\_features)$（ $4/14 \approx 28.5$ ） 个参数 。模型的性能上有小幅度下降,似然函数的对数值从 -4546.3 降低到 -4594.6。尽管如此，可以看出模型少部分到参数影响了绝大部分的效果。
- 补充：在实现模型的过程中，本人参考了sklearn库的官方实现。不得不说，sklearn 库源码阅读就是一个大型的矩阵论实践课。在效率方面，sklearn为了提升 test 的性能，在训练时保存了精度矩阵（协方差矩阵的逆），以及精度矩阵的 cholesky 分解，以此降低了多元高斯分布的概率密度函数的计算开销。（尽管本人在实验过程中虽然没有采用cholesky 分解优化的方法，主要原因是时间紧迫）

(3.2）扩展实验

- 按照 8:2 的比例，随机将数据分为训练集和测试集 使用 sklearn 中的 train_test_spit 即可。

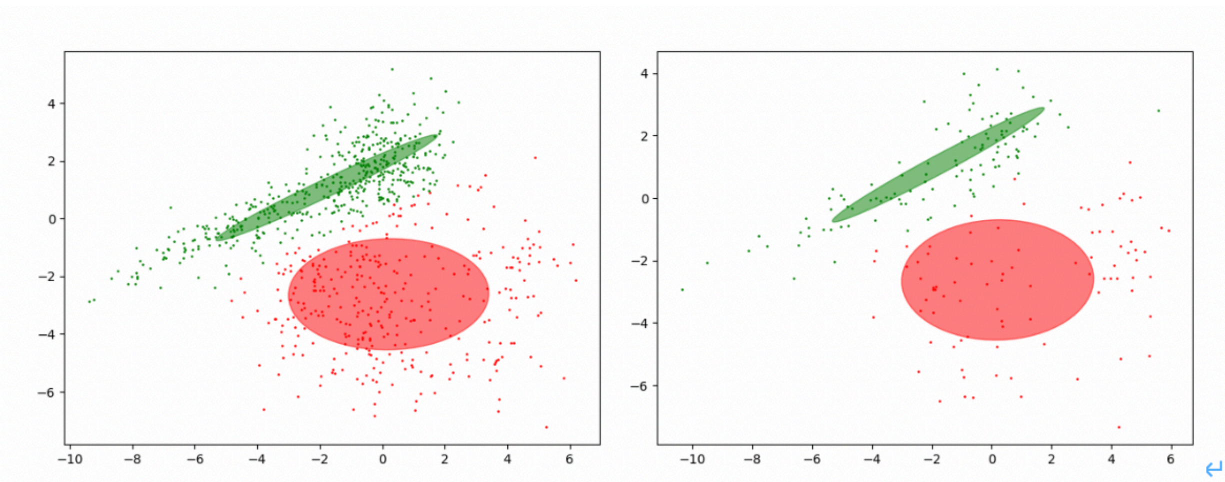
```
from sklearn.model_selection import train_test_split

X_train, X_test = train_test_split(PointSet, test_size=0.2,
random_state=16)
```

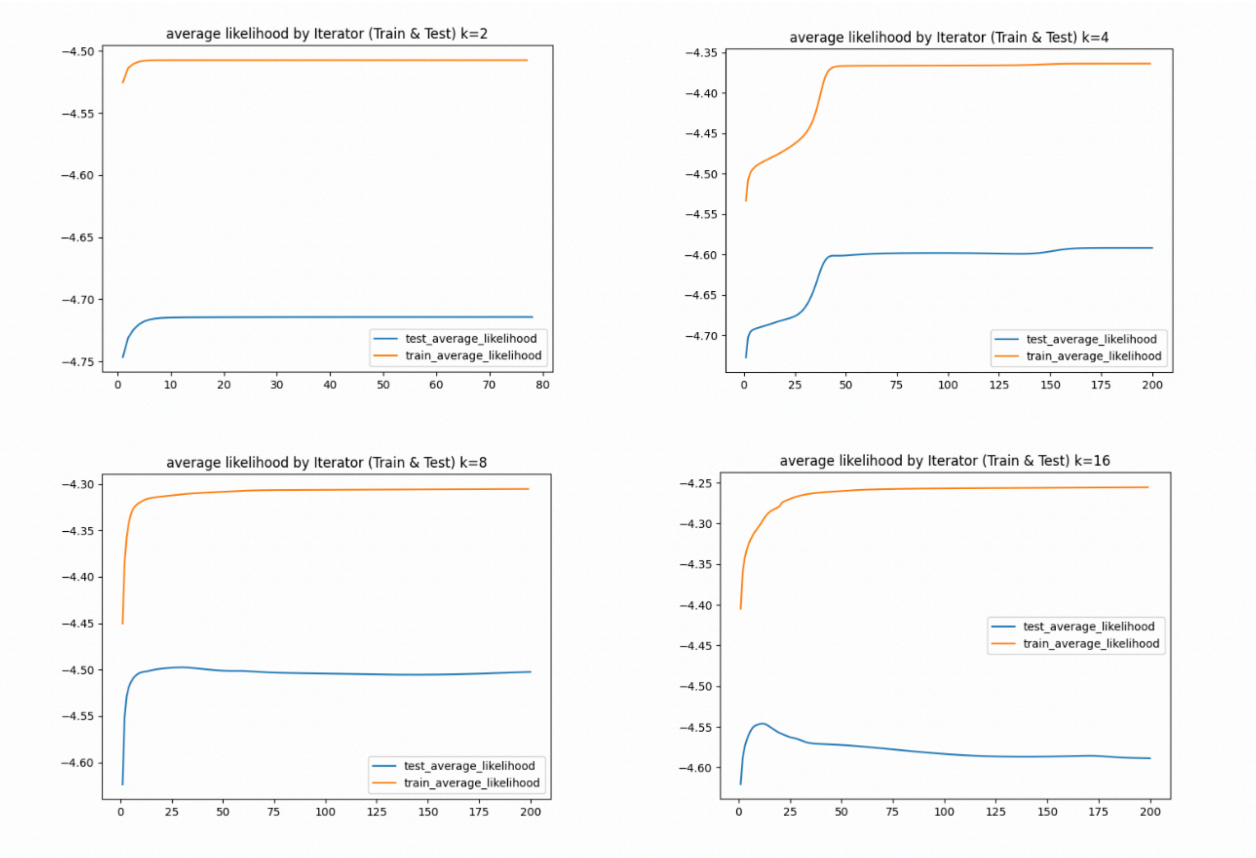
- 对于不同个数的混合成分， 绘制随着迭代的 进行，模型在训练集和测试集上的似然，并对结果进行讨论。

  为了方便显示，我们不直接使用模型在train/test上的likelihood进行分析，而是使用平均likelihood 进行分析。原因是：likelihood函数受到样本数量的直接影响，样本越多生成样本的概率（似然）自然越低。

  我们首先画出了k=2时模型在测试集/训练集上的效果，如下：可以看出，在一下训练/测试集的划分下，模型的结果和基础部分中效果类似。



- 然后，我们进一步画出了对于不同个数的混合成分(k=2,4,8,16)， 随着迭代的进行，模型在训练集和测试集上的似然曲线。



- 通过实验，我们可以看出，k=2的时候，模型在测试/训练集上的averave likelihood 在6 epoch 就达到了瓶颈,

  k=4 的时候模型训练过程总体比较平稳，模型在测试/训练集上的likelihood均一直在提升。k=8的时候，模型开始出现在训练集上仍有提升，在test set上效果下滑的现象。k=16的时候，在大约8 epochs的时候，模型在测试集上效果最优，之后出现测试集上效果下滑问题。

- 可以分析的出，k=2的时候，模型参数过少（模型过于简单），导致模型出现严重的欠拟合现象。k=4时候模型是我们最希望看到的模型，它即有更好的likelihood由能使模型在未知数据上表现更好。随着模型逐渐复杂，k=8/16的时候，模型开始出现过拟合的情况。模型在训练集上效果很好，但是其在未见的数据上效果反而不如k=4的情况。