PRML ass3 高斯混合模型 实验报告

Part 01 代码运行命令

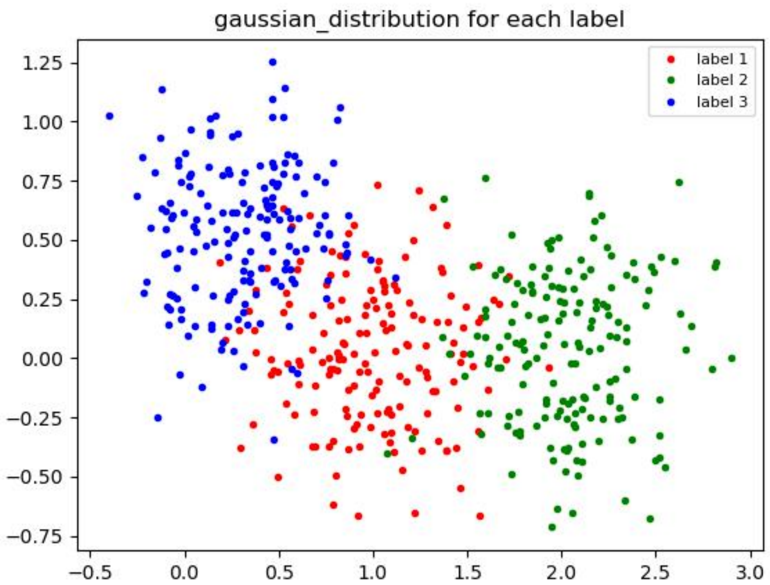
>>> python source.py

Part02 无标签聚类数据构建

程序采用高斯分布构造数据。由于提交的数据集 dataset.data 不允许超过 20KB，故提交数据中三组高斯分布的均值、协方差和数据大小为：

```
mean = np.array([[1, 0], [2, 0], [0.3, 0.5]])
cov = np.array([[[0.1, 0], [0, 0.1]]]*3)
size = np.array([170, 170, 170])
```

所构造出的数据集图像如下：



其中三种颜色的点分别表示三个不同的高斯分布。数据集 dataset.data 中三组数据打乱存放，未标注标签。

Part 03 高斯混合模型的描述与分析

X_j 表示第 j 个观测数据，j=1,2,...,N；
K 是混合模型中子高斯模型的数量，k=1,2,...,K；

α_k 是观测数据属于第 k 个子模型的概率，$\alpha_k \geq 0, \sum_{k=1}^{K} \alpha_k = 1$；

$\Phi\left(x \mid \theta_k\left(\mu_k, \sigma_k^2\right)\right)$ 是第 k 个子模型的高斯分布密度函数，

$\theta_k\left(x, \mu_k, \sigma_k^2\right) = \frac{1}{(2\pi)^{0.5\dim s}|\Sigma|^{0.5}} \exp\left(\left(x-\mu\right)^T \Sigma^{-1}\left(x-\mu\right)\right)$，对于这个模型而言，参数 μ_k，σ_k，

表示每个子模型的期望和协方差；

```
P[l] = self.p[l] * (np.exp(-0.5 * diff.T.dot(np.linalg.inv(self.cov[l])).dot(diff))) / (
            pow(2 * np.pi, self.dims / 2) * pow(np.linalg.det(self.cov[l]), 0.5))
```

γ_{jk} 表示第 j 个观测数据属于第 k 个子模型的概率；

高斯混合模型的概率分布为：$P(x\mid\theta)=\sum_{k=1}^{K}\alpha_k\Phi(x\mid\theta_k)$

高斯混合模型的 log-似然函数为：$\log L(\theta)=\sum_{j=1}^{N}\log\left(\sum_{k=1}^{K}\alpha_k\Phi(x\mid\theta_k)\right)$。

EM 算法迭代包含 E 步骤：对每个 j=1,2,...,N 求期望 $E(\gamma_{jk}\mid X,\theta)$，M 步骤：求极大，计算新一轮的模型参数。

- 首先对参数进行初始化：

```
def __init__(self, dataset, k, dims, length):
    self.k = k # 子高斯分布数
    self.mean = np.array([dataset[random.randint(0, len(dataset))] for i in range(k)]) #任取三点作为均值点初始值
    self.cov = np.full((k, dims, dims), np.diag(np.full(dims, 1.0))) #协方差
    self.p = np.ones(k)/k #先验概率
    self.dims= dims #高斯分布数据维数
    self.length = length #总数据数
```

- E-step：依据当前参数，计算每个数据 j 来自子模型 k 的可能性：$\gamma_{jk}=\frac{\alpha_k\Phi(x_j\mid\theta_k)}{\sum_{k=1}^{K}\alpha_k\Phi(x_j\mid\theta_k)}$

```
for l in range(self.k):
    diff = dataset[j] - self.mean[l]
    P[l] = self.p[l] * (np.exp(-0.5 * diff.T.dot(np.linalg.inv(self.cov[l])).dot(diff))) / (
                pow(2 * np.pi, self.dims / 2) * pow(np.linalg.det(self.cov[l]), 0.5))
posterior[j] = P / np.sum(P)
```

- M-step：计算新一轮迭代的模型参数：

$$\mu_k=\frac{\sum_{j}^{N}(\gamma_{jk}x_j)}{\sum_{j}^{N}\gamma_{jk}},k=1,2,\ldots,K$$

```
self.mean = np.array([1 / nextp[j] * np.sum(dataset * posterior[:, j].reshape((self.length, 1)),
axis=0) for j in range(self.k)])
```

$$\Sigma_k=\frac{\sum_{j}^{N}\gamma_{jk}(x_j-\mu_k)(x_j-\mu_k)^T}{\sum_{j}^{N}\gamma_{jk}},k=1,2,\ldots,K$$

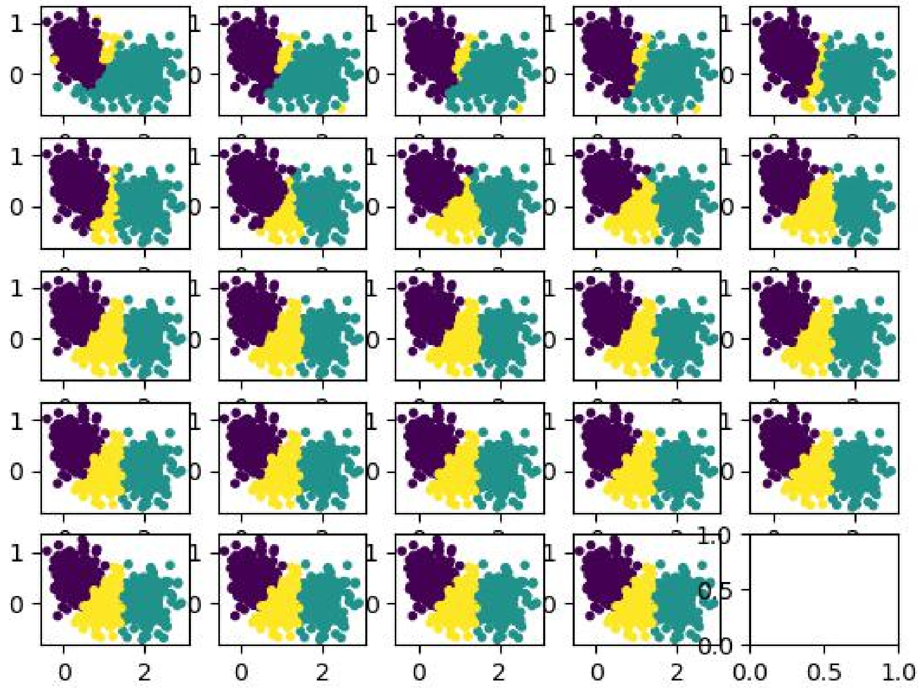
```
for l in range(self.length):
    self.cov[j] += np.dot((dataset[l].reshape(1, self.dims) - self.mean[j]).T, (dataset[l] - self.mean[j]).reshape(1, self.dims)) * posterior[l, j]
self.cov[j] = self.cov[j] / nextp[j]
```

$$\alpha_k = \frac{\sum_{j=1}^{N} \gamma_{jk}}{N}$$

```
self.p = nextp / self.length
```

重复多次计算 E-step 和 M-step 即可得到最终收敛时的高斯混合模型参数。

对于 dataset.data 中的数据，每 4 次迭代所得到的 label 分类图如下：



可以看出约 40~50 次迭代后可以求出最优参数。

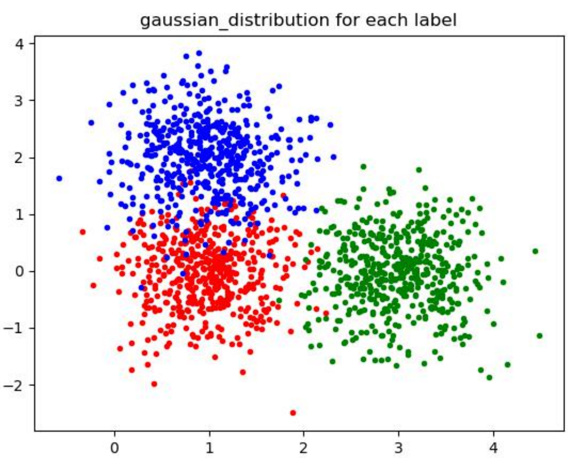
Part 04 特殊情况讨论

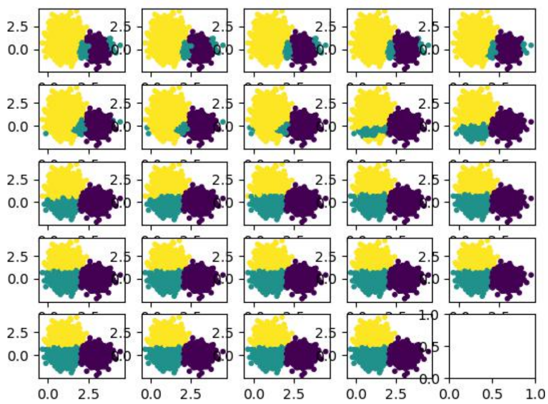
01 某几个数据集较相近

三组高斯分布数据采用以下均值，协方差和数据大小：

```
mean = np.array([[1, 0], [3, 0], [1, 2]])
cov = np.array([[[.2, 0], [0, .4]]]*3)
size = np.array([500, 500, 500])
```

数据分布点图和每 12 次迭代所得到的 label 分布图如下：





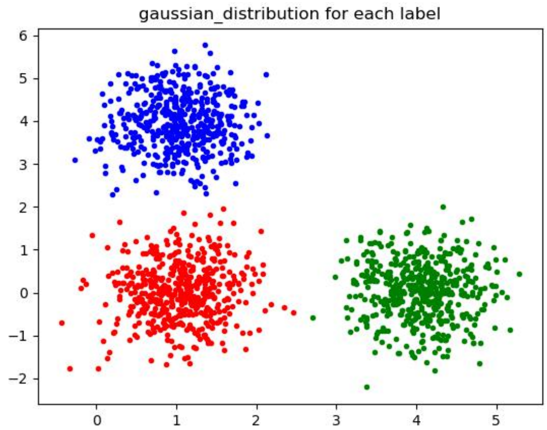
可以看出针对这组数据，大约在 180 次迭代后可以得到较准确分布图。在执行多组数据并分析后得出，根据 GMM 初始均值点取值不同，EM 所需迭代次数不同，总体上所需 EM 迭代次数高于三组高斯分布数据集分散的情况。

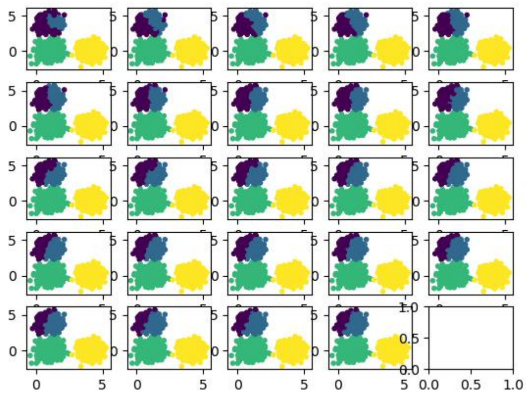
02 GMM 迭代的 k 与数据集数不同

三组高斯分布数据采用以下均值，协方差和数据大小：

```
mean = np.array([[1, 0], [4, 0], [1, 4]])
cov = np.array([[[.2, 0], [0, .4]]]*3)
size = np.array([500, 500, 500])
```

数据分布点图和每 12 次迭代所得到的 label 分布图如下：





其中 GMM 类执行的子高斯分布数 k=4，可以看出如果 k 与数据集中高斯分布集数不同，则会将某几个数据集分裂为多个数据集。

Part 05 参考网页

https://zhuanlan.zhihu.com/p/30483076