GMM

1. 数据集介绍

1. Land Mines数据集:该数据集是地雷探测的数据集，有4个离散的feature，target是常见的5种地雷类型，共有338条数据
2. Iris数据集:该数据集是最常见的数据集，共有120条数据，每行数据有4个维度的特征，且都为连续变量

目标类别有三种，分别为Iris Setosa, Iris Versicolour和Iris Virginica

2. 程序模块介绍

2.1 GMM流程介绍

2.1.1 参数初始化

这里主要初始化4个参数:下面三个和γ

Initialize parameters $\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i)\}$

代码如下:

```
self.alpha=np.random.rand(k)
self.alpha=self.alpha/np.sum(self.alpha) #保证和为1
self.miu=np.random.rand(self.k,self.dim)
#生成维度为[k,dim,dim]的对称阵,[dim,dim]对角线元素为1
self.covariance=np.array([np.identity(self.dim) for _ in range(self.k)])
self.y=np.random.rand(self.data_n,self.k)  #叫y是因为和γ很像....
```

2.1.2 e-step算法实现

repeat

  for $j = 1,2,..,n$ **do**

    compute $\gamma_{ji} = p_{\mathcal{M}}(z_j = i|\boldsymbol{x}_j)(1 \leq i \leq k)$

从PPT可以看出,e-step核心就是对每个xi,计算pM函数，也就是高斯混合函数，这里参考PPT

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

$$p_{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i \cdot p(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i), \quad \sum_{i=1}^{k} \alpha_i = 1$$

转换为python代码实现如下:

```
"""
参数解释:
x:当前输入值
miu:当前μ值
convariance:当前协方差矩阵
"""
def calculate_gaussian(self,x,miu,convariance):
        k1=np.power(2*np.pi,self.dim/2)
        k2=np.linalg.det(convariance)
        k2=np.power(k2,0.5)
        k3=np.exp(-0.5*np.dot(np.dot((x-miu).T,np.linalg.inv(convariance)),x-
miu))
        return k3/(k1*k2)
prob_list.append(self.alpha[i]*self.calculate_gaussian(xj,self.miu[i],self.cova
riance[i]))
```

2.1.3 m-step算法实现

m-step则主要实现对3个参数的更新,因为推导过程已经在上课时讲解，所以我们只需要实现下图3个更新的方程即可:

for $i = 1,2,...,k$ **do**

compute $\boldsymbol{\mu}_i' = \frac{\sum_{j=1}^{n}\gamma_{ji}\boldsymbol{x}_j}{\sum_{j=1}^{n}\gamma_{ji}}$, $\Sigma_i' = \frac{\sum_{j=1}^{n}\gamma_{ji}(\boldsymbol{x}_j-\boldsymbol{\mu}_i)(\boldsymbol{x}_j-\boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n}\gamma_{ji}}$, $\alpha_i' = \frac{1}{n}\sum_{j=1}^{n}\gamma_{ji}$

update $\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i)\}$ with $\{(\alpha_i', \boldsymbol{\mu}_i', \Sigma_i')\}$

更新参数的核心代码如下:

```
self.miu[i]=np.sum(self.y_x_list,axis=0)/np.sum(self.y_list)
self.covariance[i]=np.sum(self.y_xmiu_list,axis=0)/np.sum(self.y_list)
self.alpha[i]=np.sum(self.y_list)/self.data_n
```

2.1.4 迭代终止条件

until stop criteria is satisfied

在这里我们需要设置循环终止条件，我们每次epoch时计算该值，记录最大值出现时各参数的值和γ，在测试时发现log后面项为0可能会导致nan，所以对log项内结果加1e-5来避免nan

在这里我们记录n个epoch内最大值作为最终的输出值

$$\max_{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\Sigma}} \sum_{j=1}^{n}\sum_{i=1}^{k} \gamma_{ji} \log\left(\alpha_i \cdot p(\boldsymbol{x}_j;\boldsymbol{\mu}_i,\Sigma_i)\right)$$

核心代码如下:

```
def max_fun(self,x):
    res=0.0
    for j in range(0,self.data_n):
        for i in range(0,self.k):
            res+=self.y[j]
[i]*np.log(self.alpha[i]*self.calculate_gaussian(x[j],self.miu[i],self.covarian
ce[i])/10000)
    return res
```

2.1.5 指标展示

在这里指标展示，我们选用聚类纯度purity来分析GMM的效果，因为数据集的feature都是超过二维的，直接用散点图展示聚类效果，需要使用PCA对特征进行降维，展示的效果不一定能符合实际数据分布，在查询后我们选择了purity作为指标，因为我们的数据集实际是有label的，所以我们可以对聚类效果进行定量的分析，其公式如下:

$$\mathrm{Purity}(\Omega, C) = \frac{1}{N}\sum_{k}\max_{j}\left|w_k \cap c_j\right|$$

其对应核心代码如下:

```
for cluster in true_labels:
    if cluster==[]:
        most_common_rates.append(0)
```

```
        continue
    label_counts = Counter(cluster)
    most_common_label, most_common_count = label_counts.most_common(1)[0]
    most_common_rate = most_common_count / len(cluster)
    most_common_counts.append(most_common_count)
    most_common_rates.append(most_common_rate)
```

2.2 代码整体流程

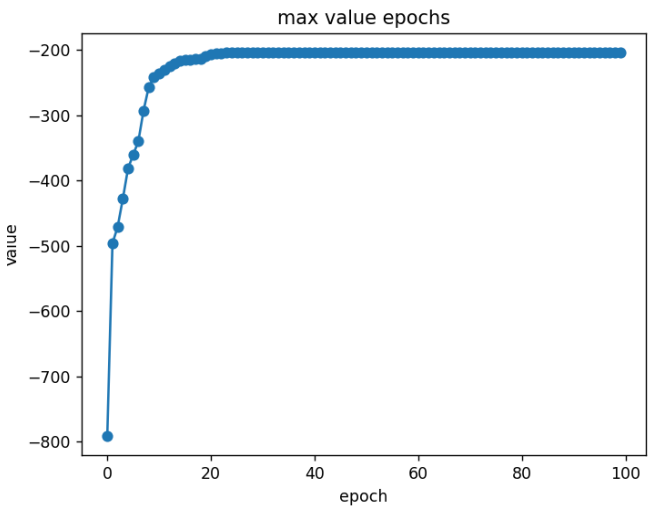
在完成Gmm核心部分后，我们就可以实现整个流程了

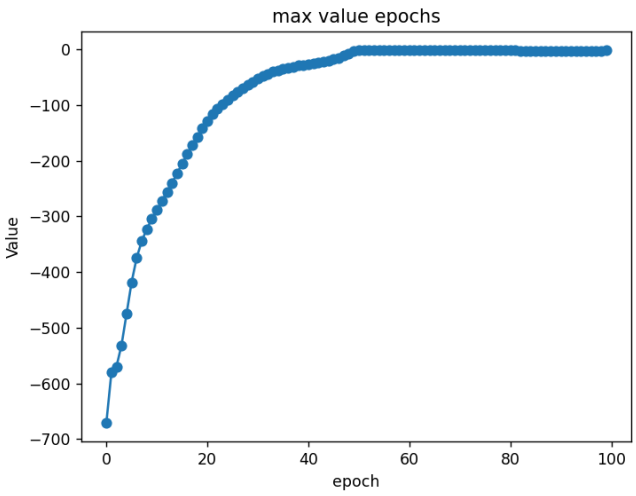
1. 实现数据读取，在这里实现dataReader类，来完成对不同数据集的统一读取，输入数据集id，返回(feature,target,num_classes)即可
2. 模型训练过程，调用前面实现的Gmm模型，训练n个epoch后输出目标函数最大值对应的聚类，并计算类内纯度

3. 结果展示

3.1 最大值变化曲线

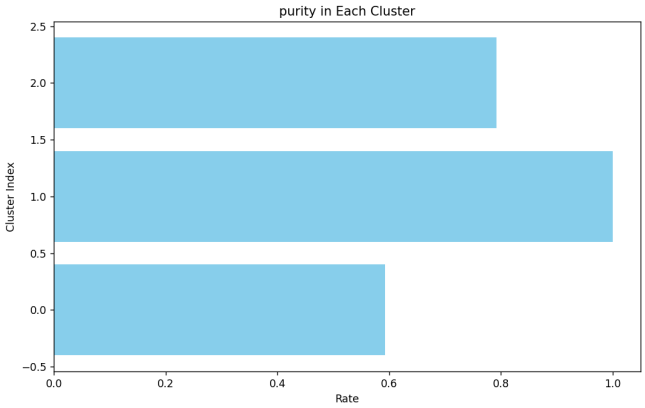
该曲线展示两个数据集随着GMM训练过程中，max-step目标函数的变化过程(log内加了1e-5防止nan,和PPT内有些不一样),两个数据集的变化曲线如下:可以看出对于iris数据集，其在20个epoch左右就收敛了，而对于较复杂的land mines，大概在60个epoch左右收敛

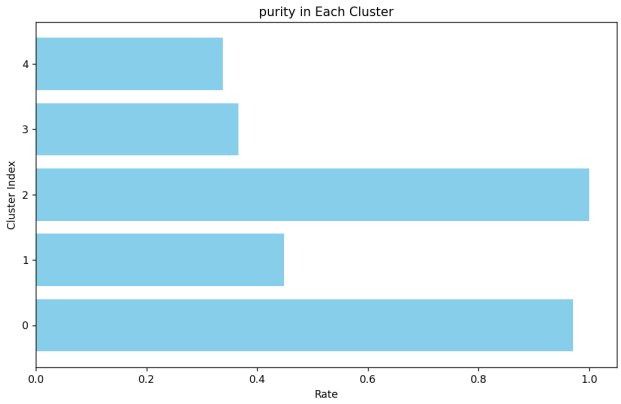




3.2 聚类纯度purity

在2.1.5中我们讲解了purity这一指标，在这里我们给出两个数据集的purity结果展示:对于iris数据集，其3个聚类的类内purity基本超过60%，而对于land mines数据集，其1,3,4聚类的purity较差，说明该模型的分类效果一般





4. 改进和展望

1. 因为选取的模型都是有标签数据集，所以从聚类结果可以判断出各聚类内部的聚合程度，从结果看，手动实现的GMM模型分类效率一般，可能还有一个没有注意到tips可以改进
2. 模型参数更新和目标值计算都是利用numpy将PPT上的公式转换为python代码实现，在计算速度上可能有所欠缺，在模型训练速度上还可以继续改进