# 模式识别实验报告

## 实验二 GMM分类器

学院：计算学部

姓名： 张亚博

学号：1170300521

1. **实验内容**
2. 使用Python或Matlab编程实现GMM算法：要求独立完成算法编程，禁止调用已有函数库或工具箱中的函数；
3. 使用仿真数据测试算法的正确性：两类2维各1000个训练样本Train1和Train2分别采样自如下两个GMM，使用训练样本分别估计包含2个分量高斯的GMM参数。

GMM1： ，，

，，

GMM2： ，，

，，

构造区分两类的GMM分类器，测试采样自同样GMM的测试样本Test1和Test2。

1. MNIST数据集测试：使用TrainSamples中的30000个17维特征手写数字样本训练GMM分类器区分10个类别，TrainLabels中包含训练样本的标签；测试设置不同高斯数量GMM分类器对TestSamples中10000个样本的识别正确率。
2. **程序代码**

（GMM参数估计部分和GMM分类器部分代码）

def Gaussian(self, x, mean, cov):  
 *"""  
 高斯分布概率密度函数  
 :param x: 输入数据  
 :param mean: 均值数组  
 :param cov: 协方差矩阵  
 :return: x的概率  
 """* dim = np.shape(cov)[0]  
 if np.isnan(np.sum(cov)):  
 print("cov is nan!!!!!!!!!!!!")  
 cov = np.eye(dim) \* 0.01  
 covdet = np.linalg.det(cov)  
 covinv = np.linalg.inv(cov)  
 xdiff = (x - mean).reshape((1, dim))  
 # 概率密度  
 a = np.power(2 \* np.pi, dim)  
 a = np.power(a \* np.abs(covdet), 0.5)  
 b = xdiff.dot(covinv).dot(xdiff.T)  
 b = self.clamp(b) # 解决溢出问题  
 b = np.exp(-0.5 \* b)[0][0]  
 prob = 1.0 / (a) \* b  
 return prob  
  
  
def EM(self):  
 *"""  
 训练阶段 , EM算法  
 :return:  
 # self.posibility 表示第j个观测数据属于第k个gmm的概率  
 # self.prediction 表示第i个数据的类别(取argmax得到的)  
 """* log\_likelyhood = 0  
 old\_log\_likelyhood = 1  
 len, dim = np.shape(self.Data)  
 # gamma表示第j个观测数据属于第k个gmm的概率  
 gammas = [np.zeros(self.K) for i in range(len)]  
 while np.abs(log\_likelyhood - old\_log\_likelyhood) > 0.01:  
 old\_log\_likelyhood = log\_likelyhood  
 # E-step  
 for n in range(len):  
 # 计算后验概率  
 respons = [self.weights[k] \* self.Gaussian(self.Data[n], self.means[k], self.covars[k]) for k in  
 range(self.K)]  
 respons = np.array(respons) + 1e-6  
 sum\_respons = np.sum(respons)  
 sum\_respons = 1e12 if sum\_respons == np.inf else sum\_respons  
 gammas[n] = respons / sum\_respons  
  
 # M-step  
 for k in range(self.K):  
 # N\_k表示N个样本中有多少属于第k个高斯  
 N\_k = np.sum([gammas[n][k] for n in range(len)])  
 # 更新每个高斯分布的概率  
 self.weights[k] = 1.0 \* N\_k / len  
 # 更新高斯分布的均值  
 self.means[k] = (1.0 / N\_k) \* np.sum([gammas[n][k] \* self.Data[n] for n in range(len)], axis=0)  
 xdiffs = self.Data - self.means[k]  
 # 更新高斯分布的协方差矩阵  
 self.covars[k] = (1.0 / N\_k) \* np.sum(  
 [gammas[n][k] \* xdiffs[n].reshape((dim, 1)).dot(xdiffs[n].reshape((1, dim))) for n in range(len)],  
 axis=0)  
 log\_likelyhood = []  
 for n in range(len):  
 tmp = [np.sum(self.weights[k] \* self.Gaussian(self.Data[n], self.means[k], self.covars[k])) for k in  
 range(self.K)]  
 tmp = np.log(np.array(tmp) + 1e-6) # 防止溢出，加快收敛  
 log\_likelyhood.append(list(tmp))  
 log\_likelyhood = np.sum(log\_likelyhood)  
 for i in range(len):  
 gammas[i] = gammas[i] / np.sum(gammas[i])  
  
 self.posibility = gammas  
 self.prediction = [np.argmax(gammas[i]) for i in range(len)]

1. **实验结果**
2. 仿真数据实验结果：给出估计出的两个GMM 模型参数，以及测试样本的识别结果。

GMM估计模型参数

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| GMM1-Gauss1 | 0.3412 | [9.9641, 9.9442] | [[2.0671 , 2.4271],  [2.4271, 5.4250]] |
| GMM1-Gauss2 | 0.6679 | [-0.0484,  -0.0329] | [[2.8612, 0.9851],  [0.9851, 0.9802 ]] |
| GMM2-Gauss1 | 0.3320 | [14.9664, 19.9883] | [[5.3474, 2.2363],  [2.2363, 1.1780]] |
| GMM2-Gauss2 | 0.6679 | [ 2.0228, 10.1679] | [[0.9766, 0.9171],  [0.9172, 2.7513]] |

**GMM分类器识别结果**

|  |  |  |
| --- | --- | --- |
|  | 正确识别数 | 正确识别率 |
| Test1 | 1000 | 1000 |
| Test2 | 1000 | 1000 |

1. MNIST数据集实验结果：

**GMM分类器识别正确率**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 高斯数 | 1 | 2 | 3 | 4 | 5 |
| 正确识别数 | 9329 | 9480 | 9415 | 9461 | 9449 |
| 正确识别率 | 0.9329 | 0.9480 | 0.9415 | 0.9461 | 0.9449 |