# 模式识别实验报告

## 实验二 GMM分类器

学院：机电工程

姓名： 王一径

学号：16S008203

1. **实验内容**
2. 使用C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10000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(python)**

（GMM参数估计部分和GMM分类器部分代码）

class GMM(object):

#==============

#@param

#data: 训练数据

#k: 高斯数

#dim: 维度

#==============

def \_\_init\_\_(self, data, k, dim):

self.data = np.array(data).astype(float)

self.k = k

self.dim = dim

self.a = np.random.rand(k)

self.a /= np.sum(self.a)

if len(self.data) > 0:

tempMax = np.amax(self.data)

tempMin = np.amin(self.data)

self.mu = np.random.uniform(tempMin-1, tempMax+1, k\*dim).reshape(k,dim)

self.sigma = np.zeros((k,dim,dim))

self.sigmaI = np.zeros((k,dim,dim))

i=0

while i<k:

self.sigma[i] = np.diag(np.random.rand(dim)+1e+10)

i += 1

i=0

while i<k:

self.sigmaI[i] = np.array(np.mat(self.sigma[i]).I)

i += 1

self.P = np.random.rand(self.data.shape[0], k)

tempSum = np.sum(self.P, axis=1)

self.P = np.divide(self.P.T, tempSum).T

#计算P(w|x)

def calP(self, x):

tempP = np.zeros(self.k)

tempSum = 0.0

i=0

while i<self.k:

tempP[i] = 1.0/((2\*np.pi)\*\*(self.dim/2)\*(np.linalg.det(self.sigma[i])\*\*0.5))\*\

np.exp((-0.5)\*np.dot(np.dot((x-self.mu[i]),self.sigmaI[i]),(x-self.mu[i])))\*\

self.a[i]

tempSum += tempP[i]

i += 1

tempP /= tempSum

return tempP

#实现EM算法进行参数估计部分

def train(self):

e = 1

while e > 1e-10:

#E step

i = 0

while i < self.data.shape[0]:

self.P[i] = self.calP(self.data[i])

i += 1

#M step

tempA = copy.deepcopy(self.a)

self.a = np.mean(self.P, axis=0)

tempMu = copy.deepcopy(self.mu)

i = 0

while i<self.k:

self.mu[i] = (np.sum(self.P[:, i].reshape(self.P.shape[0],1)\*self.data, axis=0) / np.sum(self.P[:, i]))[0]

i += 1

tempSigma = copy.deepcopy(self.sigma)

i = 0

while i<self.k:

tempDif = self.data - self.mu[i]

self.sigma[i] = np.dot(tempDif.T, tempDif\*self.P[:,i].reshape(self.P.shape[0], 1)) / np.sum(self.P[:,i])

i += 1

i = 0

while i<self.k:

self.sigmaI[i] = np.array(np.mat(self.sigma[i]).I)

i += 1

#终止条件判断

e = np.sum((self.a-tempA)\*\*2) + np.sum((self.mu-tempMu)\*\*2)+np.sum((self.sigma-tempSigma)\*\*2)

#使用GMM进行预测部分

#==========================

#@param

#testData:训练数据

#modelSet: 10个加载了相应训练好的参数的模型列表

#w: 10个模型的先验概率，即都是1/10

#

#@return

#输出预测结果

#===============================================

def predict(testData, modelSet, w):

testData = np.array(testData).astype(float)

predictPro = np.zeros((testData.shape[0], len(w)))

i = 0

while i < testData.shape[0]:

j = 0

while j < len(w):

predictPro[i, j] = modelSet[j].calPro(testData[i])\*w[j]

j += 1

i += 1

return np.argmax(predictPro, axis=1)

#比较预测标签和真实标签，返回正确识别率

def compare(predictMnistLabel, MnistLabel):

MnistLabel = np.array(MnistLabel).T[0].astype(float)

temp = np.ones(MnistLabel.shape[0])

return np.sum(temp[predictMnistLabel == MnistLabel]) / MnistLabel.shape[0]

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 | 0.65870479 | [-0.05011465,-0.05011465] | [[2.84635624, 0.96638648],[0.96638648, 0.96522422]] |
| GMM1-Gauss2 | 0.34129521 | [9.96695004, 9.96695004] | [[2.03422746, 2.37535034],[2.37535034, 5.34103512]] |
| GMM2-Gauss1 | 0.66791357 | [2.02163155, 2.02163155] | [[0.96597167, 0.90962258],[0.90962258, 69.09025683]] |
| GMM2-Gauss2 | 0.33207643 | [14.96847383, 14.96847383] | [[5.31081784, 2.19810306],[2.19810306, 26.35778457]] |

**GMM分类器识别结果**

|  |  |  |
| --- | --- | --- |
|  | 正确识别数 | 正确识别率 |
| Test1 | 955 | 95.5% |
| Test2 | 1000 | 100% |

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 | 9178 | 9331 | 9410 | 9422 | 9427 |
| 正确识别率 | 91.78% | 93.31% | 94.10% | 94.22% | 94.27% |