混合高斯模型与 EM 算法

宋浩瑜 ZY2203207

目录

1. 混合高斯模型(GMM)

1.1 高斯分布

高斯模型是一种常用的变量分布模型，在数理统计领域有着广泛的应用。一维高斯分布的概率密度函数如下：

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{1.1}$$

其中μ和σ^2分别表示高斯分布的均值和方差。

更一般的情况，考虑到输入数据集为 d 维数据，可以使用多变量高斯模型来进行拟合，其中高斯分布的概率密度为：

$$N(x|\mu,\Sigma)=\frac{1}{\sqrt{(2\pi)^d\det(\Sigma)}}\exp\left[-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T\right] \tag{1.2}$$

x为一个 N×d 的向量，表示为 N 组 d 维的数据；

μ是一个 1×d 的向量，表示为每个维度的数学期望；

Σ是一个 d×d 的矩阵，表示为模型的协方差矩阵；

Det是求解该矩阵的行列式。

1.2 高斯混合模型

在现实生活中，由于数据之间可能会有比较紧密的联系，往往一个数据的表现可以通过多个维度的数据进行描述和表现。由于数据的复杂，通常实际数据的分布无法只使用一个高斯分布进行拟合，这时候我们可以将其看作多个随机过程的混合，所以我们可以定义高斯混合模型是K个高斯分布的组合，来进行复杂数据的分布拟合。

对于数据集：$x=x_1,x_2,\dots,x_N$，可以定义高斯混合模型为：

$$\begin{aligned}p(x)&=\sum_{k=1}^{K}\alpha_k N(x|\mu_k,\Sigma_k)\\&=\sum_{k=1}^{K}\alpha_k\frac{1}{\sqrt{(2\pi)^d\det(\Sigma_k)}}\exp\left[-\frac{1}{2}(x-\mu_k){\Sigma_k}^{-1}(x-\mu_k)^T\right]\end{aligned} \tag{1.3}$$

其中α_k被称为混合系数，是第 k 个高斯分布的先验概率，并且有：

$$\sum_{k=1}^{K}\alpha_k=1 \tag{1.4}$$

1.3 极大似然估计

对于相互独立的一组数据，使用极大似然估计是最方便的，用来估计分布参数的估计方法。

通常，当我们进行最大似然估计时，为了避免出现对于乘法求导的不便，我们将原似然函数取对数之后求解其的最大值，对于混合高斯分布，可以表示为：

$$\log L(\mu,\Sigma,\alpha)=\sum_{i=1}^{N}\log\left[\sum_{k=1}^{K}\alpha_k N(x_i|\mu_k,\Sigma_k)\right] \tag{1.5}$$

$$\widehat{\Theta}_{MLE}=argmax_{\Theta}\left\{\sum_{i=1}^{N}\log\left[\sum_{k=1}^{K}\alpha_k N(x_i|\mu_k,\Sigma_k)\right]\right\} \tag{1.6}$$

对于高斯混合模型进行最大似然估计，求得的似然函数比较复杂，由于多变量 GMM 似然函数涉及多个矩阵的求逆和乘积等运算。

所以要准确估计K组高斯模型的参数较为困难。

此时，我们就可以引入 EM 算法来解决。EM 算法解决的就是具有隐变量的混合模型的参数估计问题。

2. EM(Expectation Maximization)算法

EM 算法是一种迭代的算法。算法求解的过程分为两步：第一步，假设知道各个高斯分布的参数（可以初始化一个，或者基于上一步迭代结果），去估计每个高斯模型的隐变量；第二步，基于估计的隐变量，回过头再去确定高斯分布的参数。重复这两个步骤，直到收敛。

2.1 Jensen 不等式

用以化简目标函数

对于一个凸函数f和随机变量x，假设x在闭区间$[a,b]$均匀分布，$f(x)$的期望要大于$f[E(x)]$。

$$f[E(x)]\leq E[f(x)] \tag{2.1}$$

对于一个凹函数f和随机变量x，假设x在闭区间$[a,b]$均均匀分布，$f(x)$的期望要小于$f[E(x)]$。

$$f[E(x)]\geq E[f(x)] \tag{2.2}$$

2.2 隐变量

为了帮助迭代算法的过程，EM 算法提出了隐参数z，每次迭代，首先使用

上一次的参数计算隐参数z的分布，然后使用z更新似然函数，对目标参数进行估计。

在 GMM 估计问题中，将隐参数引入概率估计中：

$$p(x)=\sum_{k=1}^{K}\alpha_k N(x|\mu_k,\Sigma_k)=\sum_{k=1}^{K}p(z=k)p(x|z=k,\mu_k,\Sigma_k) \tag{2.3}$$

可以看出$p(z=k)=\alpha_k$为z的先验分布。并且将隐参数z引入似然函数中，可以得到：

$$\begin{aligned}\log L&=\sum_{i=1}^{N}\log\left[\sum_{k=1}^{K}\alpha_k N(x_i|\mu_k,\Sigma_k)\right]\\&=\sum_{i=1}^{N}\log\left[\sum_{k=1}^{K}p(z=k)p(x_i|z=k,\mu_k,\Sigma_k)\right]\\&=\sum_{i=1}^{N}\log\left[\sum_{k=1}^{K}p(z=k)p(z=k|x_i,\mu_k,\Sigma_k)\frac{p(x_i|z=k,\mu_k,\Sigma_k)}{p(z=k|x_i,\mu_k,\Sigma_k)}\right]\end{aligned} \tag{2.4}$$

2.3 似然函数简化

令

$$u=\frac{p(x_i|z=k,\mu_k,\Sigma_k)p(z=k)}{p(z=k|x_i,\mu_k,\Sigma_k)} \tag{2.5}$$

则

$$E(u)=\sum_{k=1}^{K}p(z|x_i,\mu_k,\Sigma_k)u \tag{2.6}$$

得到：

$$\log L\geq\sum_{i=1}^{N}\sum_{k=1}^{K}p(z=k|x_i,\mu_k,\Sigma_k)\log\frac{p(x_i|z=k,\mu_k,\Sigma_k)p(z=k)}{p(z=k|x_i,\mu_k,\Sigma_k)} \tag{2.7}$$

于是将似然函数化简成对数函数的两重求和，等式右侧给似然函数提供下界。则我们可以根据贝叶斯公式推导其中的后验概率：

$$p(z=k|x_i,\mu_k,\Sigma_k)=\frac{\alpha_k N(x_i|\mu_k,\Sigma_k)}{\sum_{k=1}^{K}\alpha_k N(x_i|\mu_k,\Sigma_k)} \tag{2.8}$$

定义令$\gamma(i,k)=\ p(z=k|x_i,\mu_k,\Sigma_k)$，则

$$\gamma(i,k)=\frac{\alpha_k N(x_i|\mu_k,\Sigma_k)}{\sum_{k=1}^{K}\alpha_k N(x_i|\mu_k,\Sigma_k)} \tag{2.9}$$

则

$$\log L \geq \sum_{i=1}^{N}\sum_{k=1}^{K}\gamma(i,k)\log\frac{\alpha_k N(x_i|\mu_k,\Sigma_k)}{\gamma(i,k)} \tag{2.10}$$

不等式的右侧给似然函数提供了一个下界。EM 算法提出迭代逼近的方法，不断提高下界，从而逼近似然函数。每次迭代都以下面这个目标函数作为优化目标：

$$Q(\Theta,\Theta^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{K}\gamma(i,k)^{(t)}\log\frac{\alpha_k N(x_i|\mu_k,\Sigma_k)}{\gamma(i,k)^{(t)}} \tag{2.11}$$

这个式子表示，在第 t 次迭代后，获得参数 $\Theta^{(t)}$，然后就可以计算隐参数概率 $\gamma(i,k)^{(t)}$。将隐参数代回 $Q(\Theta,\Theta^{(t)})$, 进行最大似然优化，利用$\gamma(i,k)^{(t)}$当前值, 最大化目标函数，即可求出更优的参数 $\Theta^{(t+1)}$ 。

3. EM 算法求解 GMM

当使用 EM 算法求解 GMM 时，得到

$Q(\Theta,\Theta^{t}) =$

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\gamma(i,k)^{(t)}\log\frac{\alpha_k}{\gamma(i,k)^{(t)}}\frac{1}{\sqrt{(2\pi)^d\det(\Sigma_k)}}\exp\left[-\frac{1}{2}(x_i-\mu_k){\Sigma_k}^{-1}(x_i-\mu_k)^T\right] \tag{3.1}$$

x_i是 1×d 的向量,

μ_k 是 1×d 的向量,

Σ_k是 d×d 的矩阵,

$\gamma(i,k)^{(t)}$是 N×K 的矩阵

使用 EM 算法求解 GMM 可以被拆分成两个部分，可以分别记作 E 步和 M 步。

3.1 E 步

E 步就是求当前给定的参数的期望，目标函数更新如下：

$Q(\Theta,\Theta^{(t)}) =$

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\gamma(i,k)^{(t)}\left[\log\alpha_k-\log\gamma(i,k)^{(t)}-\frac{d}{2}\log 2\pi-\frac{1}{2}\log\det(\Sigma_k)-\frac{1}{2}(x_i-\mu_k){\Sigma_k}^{-1}(x_i-\mu_k)^T\right] \tag{3.2}$$

3.2 M 步

M 步就是求使期望最大的参数值。

3.2.1 更新α_k

α_k的估计是一个受限优化问题，可以考虑使用拉格朗日乘子法计算，构造拉格朗日乘子

$$\mathcal{L}(\alpha_k, \lambda) = \sum_i \sum_k \gamma(i,k)^{(t)} \log \alpha_k + \lambda \left[\sum_k \alpha_k - 1 \right] \tag{3.3}$$

对拉格朗日方程求极值，也就是对α_k求导数为 0，即是我们要更新的α_k值。

$$\frac{\partial \mathcal{L}(\alpha_k, \lambda)}{\partial \alpha_k} = \sum_i \gamma(i,k)^{(t)} \frac{1}{\alpha_k} + \lambda = 0 \tag{3.4}$$

$$\Rightarrow \alpha_k = -\frac{\sum_i \gamma(i,k)^{(t)}}{\lambda} \tag{3.5}$$

将所有 k 项累加，就可以求得$\lambda = -N$

因此我们可以得到α_k在 t+1 次迭代的更新方程

$$\alpha_k = \frac{\sum_i \gamma(i,k)^{(t)}}{N} \tag{3.6}$$

3.2.2 更新μ_k

μ_k并没有类似α_k的限制条件，可以直接令目标函数对μ_k求导等于 0 得到极值。

$$\frac{\partial Q\left(\Theta, \Theta^{(t)}\right)}{\partial \mu_k} = \sum_i \gamma(i,k)^{(t)} \frac{\partial \left[-\frac{1}{2} (x_i - \mu_k) {\Sigma_k}^{-1} (x_i - \mu_k)^T \right]}{\partial \mu_k} = 0 \tag{3.7}$$

最终μ_k的更新方程同样是x的加权平均

$$\mu_k{}^{(t+1)} = \frac{\sum_i \gamma(i,k)^{(t)} x_i}{\sum_i \gamma(i,k)^{(t)}} \tag{3.8}$$

3.2.3 更新Σ_k

令目标函数对${\Sigma_k}^{-1}$求导等于 0 得到极值

$$\begin{aligned} \frac{\partial Q\left(\Theta, \Theta^{(t)}\right)}{\partial {\Sigma_k}^{-1}} &= \sum_i \gamma(i,k)^{(t)} \frac{\partial \left[-\frac{1}{2} \log \det(\Sigma_k) - \frac{1}{2} (x_i - \mu_k) {\Sigma_k}^{-1} (x_i - \mu_k)^T \right]}{\partial {\Sigma_k}^{-1}} \\ &= -\frac{1}{2} \sum_i \gamma(i,k)^{(t)} \left[\frac{\partial \log \det(\Sigma_k)}{\partial {\Sigma_k}^{-1}} + \frac{\partial (x_i - \mu_k) {\Sigma_k}^{-1} (x_i - \mu_k)^T}{\partial {\Sigma_k}^{-1}} \right] \\ &= 0 \end{aligned} \tag{3.9}$$

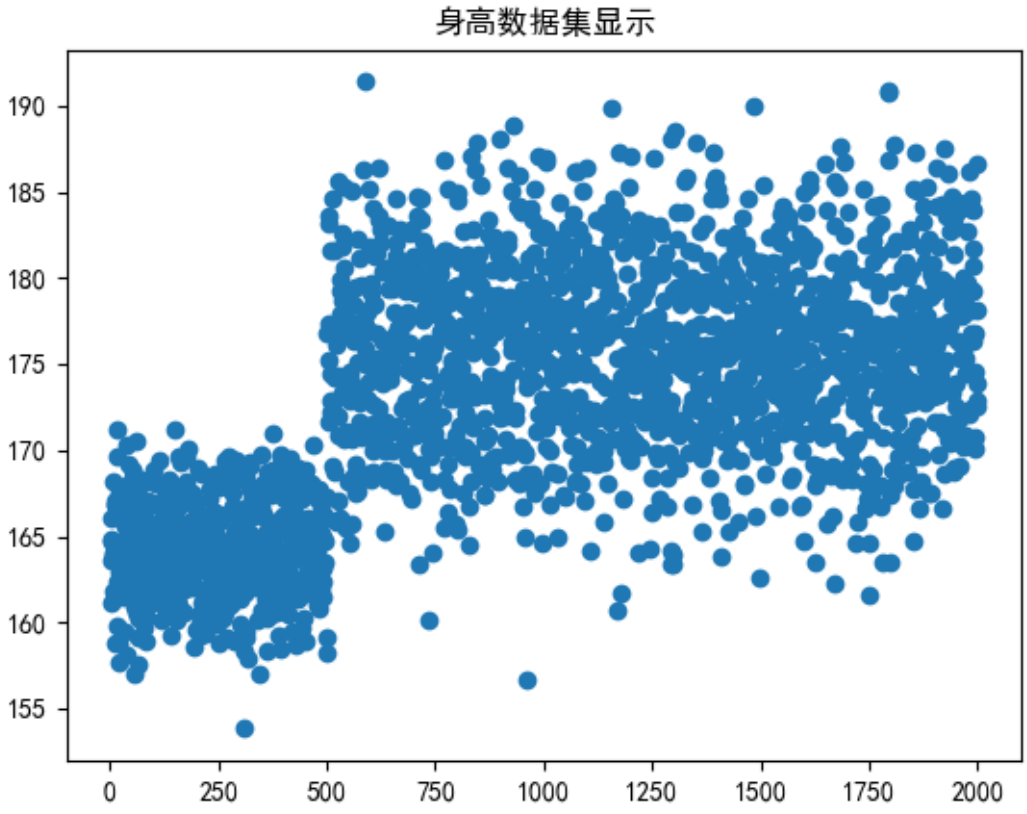
利用矩阵的求导公式，最终求解极大值，由于Σ_k的迭代更新依赖μ_k。所以我们需要先计算μ_k的迭代更新，然后更新Σ_k

$$\Sigma_k{}^{(t+1)} = \frac{\sum_i \gamma(i,k)^{(t)} \left(x_i - \mu_k{}^{(t+1)}\right)^T \left(x_i - \mu_k{}^{(t+1)}\right)}{\sum_i \gamma(i,k)^{(t)}} \tag{3.10}$$

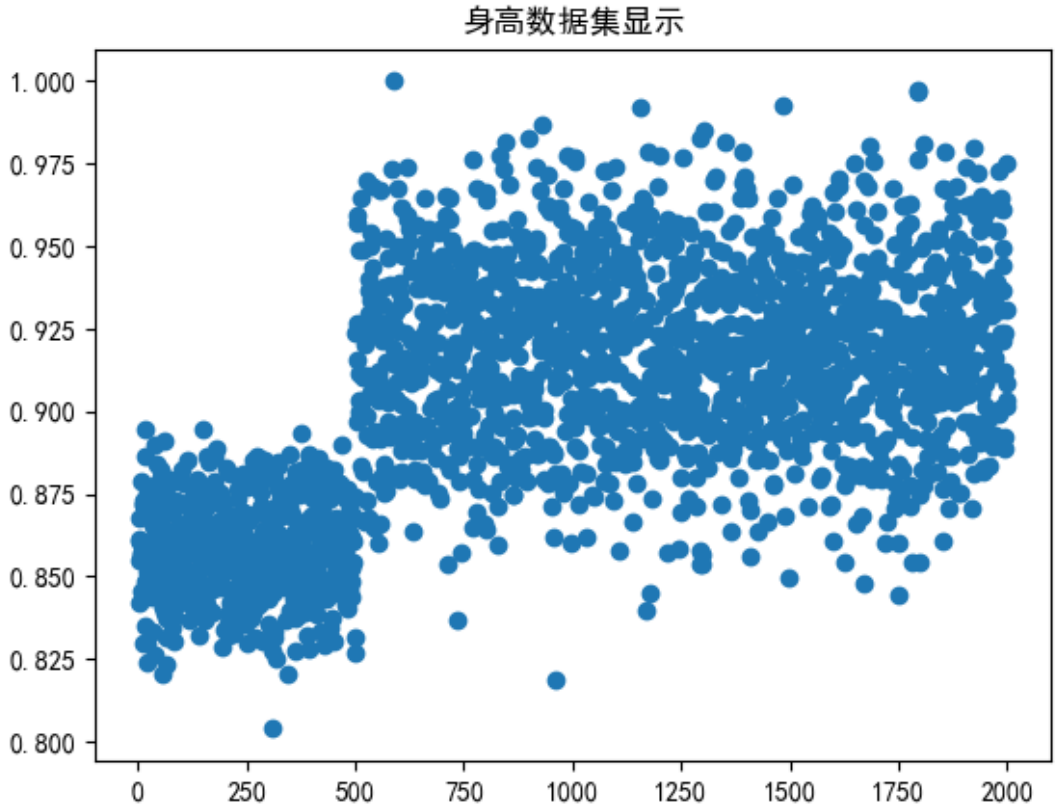
4. 代码实现

4.1 数据集

首先运行 student_height.py 文件，获取随机的学生身高数据。将获取的数据在二维平面上显示，显示效果如下图所示：



为了方便后续训练，我们还需要对对数据进行归一化处理，得到的结果如下图所示：



可以看出此时的数据被归一化到一定的范围内了。

4.2 初始化

需要将待求解的参数初始化。

4.3 高斯概率密度

建立高斯概率分布的函数，同时为了防止协方差矩阵的行列式为零，这样会导致后续求逆以及一系列计算出现错误，这里我们将矩阵加上一个较小的偏置。

4.4 迭代求解

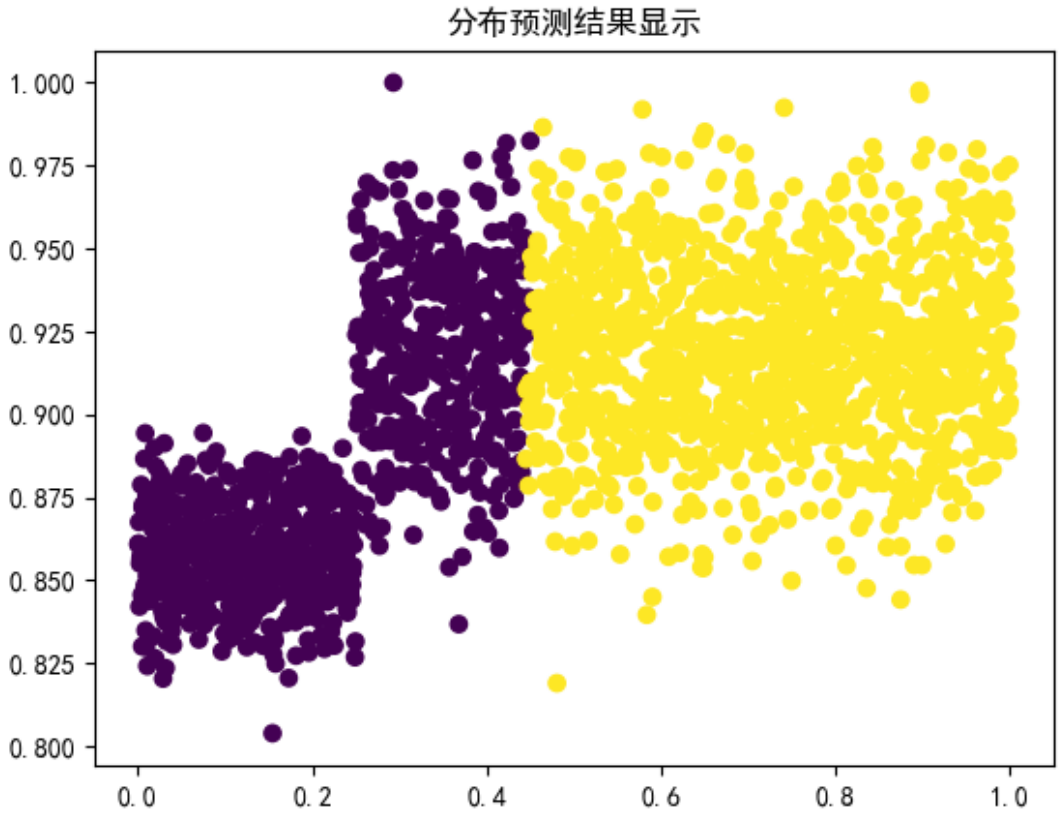
进行迭代的计算求解。

首先根据原理部分的要求，建立“gama”值用来表征第 n 个样本属于第 k 个混合高斯的概率。然后限制限差，即迭代最小的残差。之后分别进行 E 步和 M 步的计算。

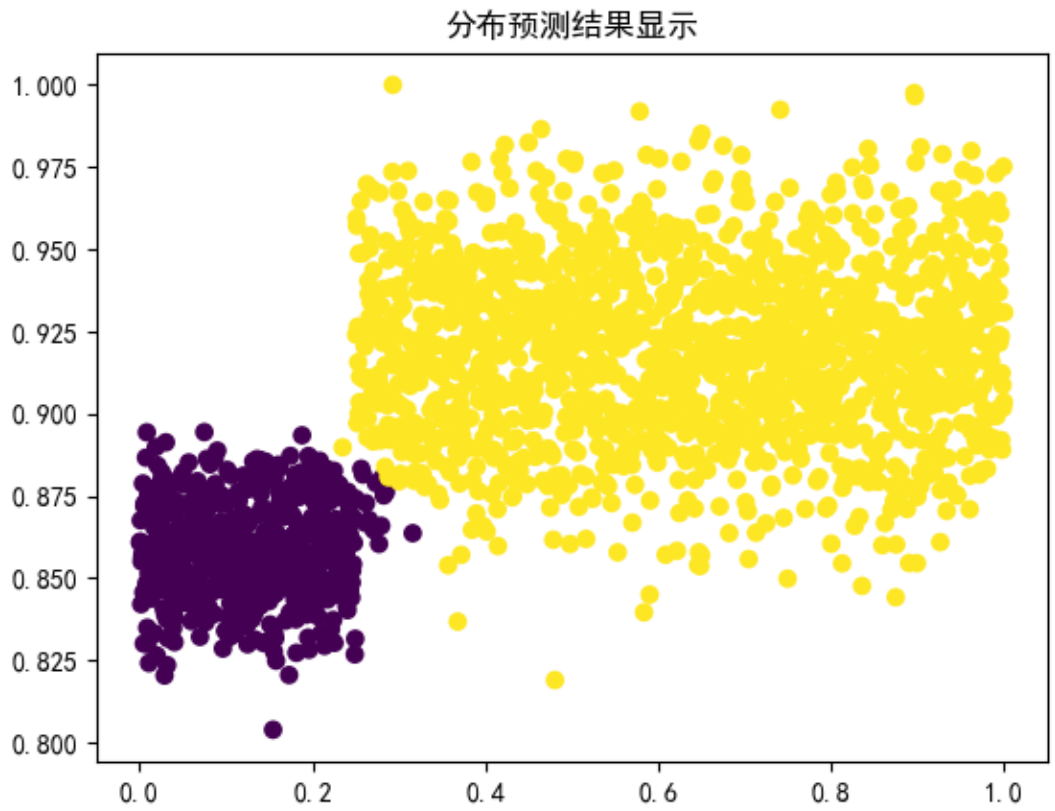
我们需要注意的是，在其中的取对数运算时，为了防止给定的输入过小使得取对数运算的结果趋近无穷，我们再取对数运算的过程中同样加上一个较小的偏置。

5. 分析

通过代码运行，我们可以对比分析真实分布与 EM 算法得到的分布结果，如下图所示。



可以看到预测分布的结果非常的相近，但是在二者交接的地方仍然有一些没有准确预测的结果。也有可能是设置偏置的问题导致最终误差的增大。



消除偏置之后可以看出，此时的检测结果非常已经非常接近我们想要得到的理想效果，左右分布被很好的分开，并且运行时间也获得了较大的提升。

6. EM 算法总结

EM 算法是一种迭代算法，对于 GMM 不能像单高斯模型那样利用 MLE 来直接给出结果，而是只能通过不断的求期望和最大化，一步一步的逼近最终的结果。

期望最大算法是一种从不完全数据或有数据丢失的数据集（存在隐含变量）中求解概率模型参数的最大似然估计方法。

EM 算法的流程：

1.初始化分布参数 θ；

2.E 步骤：根据参数初始值或上一次迭代的模型参数来计算出隐性变量的后验概率，其实就是隐性变量的期望。作为隐藏变量的现估计值：

$$Q_i\left(z^{(i)}\right) = p\left(z^{(i)}\middle|z^{(i)};\theta\right) \tag{6.1}$$

M 步骤：将似然函数最大化以获得新的参数值：

$$\theta = argmax \sum_{i} \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(z^{(i)}|z^{(i)};\theta)}{Q_i(z^{(i)})} \tag{6.2}$$

这个不断的迭代，就可以得到使似然函数 L(θ)最大化的参数 θ 了