信号建模与算法实践

多高斯混合模型

Xu Weiye

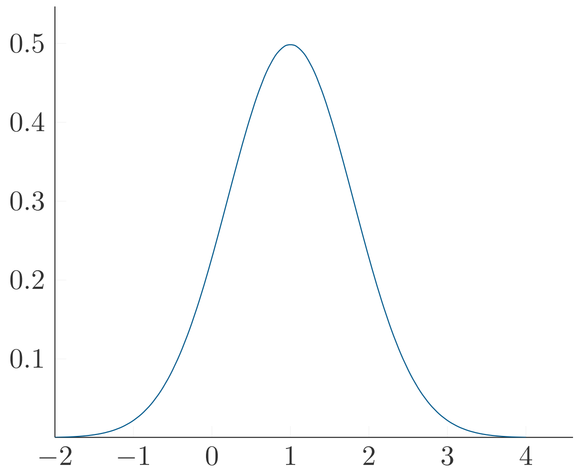
2022-10-31

Content

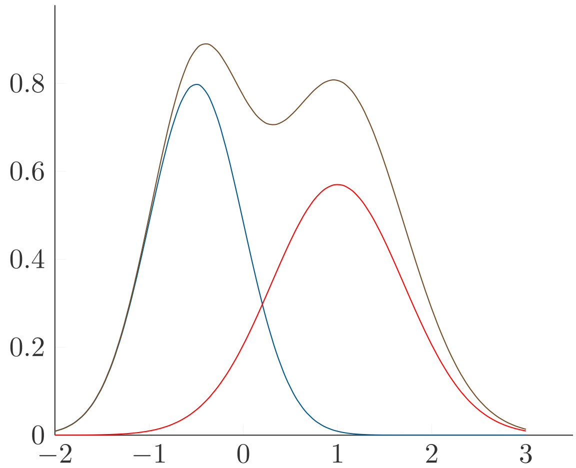
1 Introduction

一般的高斯模型如下：

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



但是大部分数据都不是按照如下分布的，但是我们还是可以依照傅里叶展开的思路，将一个分布用高斯分布展开



那么可以将表达式写成多元高斯加权和的形式

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

$$p(X) = \sum_{k=1}^{\mathbf{K}} \alpha_k \frac{1}{\sqrt{(2\pi)^d det\,|\Sigma_k|}} \exp\left[-\frac{1}{2}(X-\mu_k)\Sigma_k^{-1}(X-\mu_k)^T\right]$$

其中，Σ 是协方差矩阵，α_k 代表属于不同单个高斯分布的概率。
这样就能将单个多元高斯混合在一起表达更加复杂的分布

2 proving

根据最大似然估计

$$L(X \mid \Theta)=\prod_{i=0}^{N} p(x_i \mid \Theta)$$

其中 Θ 是所有分布参数 μ, σ, α 的集合,N 是样本个数。
同时我们取 log 不影响最大最小值，那么展开后可以得到，

$$L(X \mid \Theta)=\sum_{i=0}^{N} \ln \left[\sum_{k=0}^{\mathcal{K}} \alpha_k \mathcal{N}(X, \mu_k, \sigma_k)\right]$$

对于 mu_k 的推导

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \ln p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})=\sum_{n=1}^{N} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \frac{\partial \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\partial \boldsymbol{\mu}_k}$$

$$=-\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \boldsymbol{\Sigma}_k^{-1}\left(\mathbf{x}_n-\boldsymbol{\mu}_k\right)$$

$$=-\sum_{n=1}^{N} \gamma_{nk} \boldsymbol{\Sigma}_k^{-1}\left(\mathbf{x}_n-\boldsymbol{\mu}_k\right)=0$$

$$\sum_{n=1}^{N} \gamma_{nk}\left(\mathbf{x}_n-\boldsymbol{\mu}_k\right)=\sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n-N_k \boldsymbol{\mu}_k=0$$

$$\boldsymbol{\mu}_k=\frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n$$

$$\text{where: } N_k=\sum_{n=1}^{N} \gamma_{nk}$$

对于 Σ_k 的推导

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \ln p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) &=\sum_{n=1}^{N} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \frac{\partial \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\partial \boldsymbol{\Sigma}_k} \\
&=\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \frac{\partial \ln \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\partial \boldsymbol{\Sigma}_k} \\
&=\sum_{n=1}^{N} \gamma_{nk}\left(\frac{1}{2} \boldsymbol{\Sigma}_k-\frac{1}{2}\left(\mathbf{x}_n-\boldsymbol{\mu}_k\right)\left(\mathbf{x}_n-\boldsymbol{\mu}_k\right)^{\mathrm{T}}\right) \\
&=0
\end{aligned}$$

$$\boldsymbol{\Sigma}_k=\frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk}\left(\mathbf{x}_n-\boldsymbol{\mu}_k\right)\left(\mathbf{x}_n-\boldsymbol{\mu}_k\right)^{\mathrm{T}}$$

对于 π_k 的推导拉格朗日函数:

$$\ln p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \pi_k = 1$$

求偏导:

$$\sum_{n=1}^{N} \frac{\mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} + \lambda = 0$$

两边同乘 π_k 再对 k 求和, 得

$$\lambda = -N$$

代入拉格朗日函数:

$$\sum_{n=1}^{N} \frac{\gamma_{nk}}{\pi_k} - N = \frac{N_k}{\pi_k} - N = 0$$

$$\hat{\pi}_k = \frac{N_k}{N}$$

3 code

其中 E-STEP 如下:

```
#E-Step
p = np.zeros((N, K))
for k in range(K):
p[:, k] = alpha[k] * gaussian(X, mu[k], cov[k])
sumP = np.sum(p, axis=1)
omega = p / sumP[:, None]
```

M-STEP 如下:

```
#M-Step
omega_sum = np.sum(omega, axis=0)
alpha = omega_sum / N
for k in range(K):
    omegaX = X * omega[:, [k]]
    mu[k] = np.sum(omegaX, axis=0) / omega_sum[k]

    X_mu_k = X- mu[k]
    omega_X_mu_k =omega[:, [k]] * X_mu_k
    cov[k] = np.dot(np.transpose(omega_X_mu_k), X_mu_k) / omega_sum[k]
```

如果使用 K-means 初始化如下：

```
kmeans = KMeans(
    n_clusters=K,
    init='k-means++',
    n_init=10,
    max_iter=300)
    kmeans.fit(Y)
```

4 result

Table 1: 测试结果分析

分类超参 K	迭代 2 次	迭代 10 次	迭代 50 次
2	76.39%	76.87%	76.99%
4	76.51%	78.19%	79.52%
8	78.19%	79.28%	80.36%
16	79.88%	79.52%	80.36%