EM 算法

1 EM 算法

1.1 算法的导出

概率模型可以通过观测变量建立极大似然函数，并通过最大化极大似然函数得出参数的估计。不过有时候，模型不仅有观测变量，还含有**隐变量**或**潜在变量**。隐变量无法观测但会影响数据的分布形式。

用 Y 表示观测随机变量的数据，Z 表示隐随机变量的数据。Y, Z 连在一起时称为完全数据，观测数据 Y 称为不完全数据。对于含有隐变量的概率模型，目标是**极大化不完全数据 Y 关于参数 θ 的对数似然函数**，即极大化：

$$
\begin{aligned}
L(\theta) := \log P(Y|\theta) &= \log \sum_Z P(Y, Z|\theta) \\
&= \log(\sum_Z P(Y|Z,\theta)P(Z|\theta))
\end{aligned}
$$

我们希望极大化 $L(\theta)$。假设在第 i 次迭代后 θ 的估计值是 θ^i，我们希望找到新的估计值 θ 使得 $L(\theta) > L(\theta^i)$，并逐步达到最大值。为此，考虑：

$$
L(\theta) - L(\theta^i) = \log(\sum_Z P(Y|Z,\theta)P(Z|\theta)) - \log P(Y|\theta^i)
$$

利用 Jansen 不等式，有：

$$
\begin{aligned}
L(\theta) - L(\theta^i) &= \log(\sum_Z P(Z|Y,\theta^i)\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta^i)}) - \log P(Y|\theta^i) \\
&\geq \sum_Z P(Z|Y,\theta^i)\log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta^i)} - \sum_Z P(Z|Y,\theta^i)\log P(Y|\theta^i) \\
&= \sum_Z P(Z|Y,\theta^i)\log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta^i)P(Y|\theta^i)}
\end{aligned}
$$

令

$$B(\theta,\theta^i) := L(\theta^i) + \sum_Z P(Z|Y,\theta^i)\log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta^i)P(Y|\theta^i)}$$

则有 $L(\theta) \geq B(\theta,\theta^i)$。因此 $B(\theta,\theta^i)$ 是 $L(\theta)$ 的一个下界。且易知 $L(\theta^i) = B(\theta^i,\theta^i)$。因此任何使 $B(\theta,\theta^i)$ 增大的 θ，也可以使 $L(\theta)$ 变大。为了使 $L(\theta)$ 尽可能地大，选择 θ^{i+1} 使得 $B(\theta,\theta^i)$ 达到极大，即：

$$\theta^{i+1} = arg\max_\theta B(\theta,\theta^i)$$

略去 $B(\theta,\theta^i)$ 中与 θ 无关的常数项，则：

$$\begin{aligned}
\theta^{i+1} &= arg\max_\theta [L(\theta^i) + \sum_Z P(Z|Y,\theta^i)\log\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta^i)P(Y|\theta^i)}] \\
&= arg\max_\theta [\sum_Z P(Z|Y,\theta^i)\log(P(Y|Z,\theta)P(Z|\theta))] \\
&= arg\max_\theta [\sum_Z P(Z|Y,\theta^i)\log P(Y,Z|\theta)] \\
&:= arg\max_\theta Q(\theta,\theta^i)
\end{aligned}$$

因此，对于不完全数据的参数估计问题，可以转化为如下重复的两步：

1. **E 步**：根据当前的 θ^i 计算 $Q(\theta,\theta^i)$：

$$\begin{aligned}
Q(\theta,\theta^i) &= E_Z[\log P(Y,Z|\theta)|Y,\theta^i] \\
&= \sum_Z \log P(Y,Z|\theta)P(Z|Y,\theta^i)
\end{aligned}$$

2. **M 步**：求 $\theta^i = arg\max_\theta Q(\theta,\theta^i)$

因此，该算法被称为 EM 算法。EM 算法步骤如下：

输入：观测变量数据 Y，隐变量 Z，联合分布 $P(Y,Z|\theta)$ 和条件分布 $P(Z|Y,\theta)$。

1. 选择参数的初始值 θ^0，开始迭代
2. E 步：计算 $Q(\theta,\theta^i)$
3. M 步：得到 $\theta^i = arg\max_\theta Q(\theta,\theta^i)$
4. 重复 2,3 直至收敛

输出：模型参数 θ。

注意：EM 算法对初值很敏感，可能会收敛到局部最优值，可尝试选取多个初值进行迭代。

1.2 收敛性分析

由于 $P(Y|\theta)=\frac{P(Y,Z|\theta)}{P(Z|Y,\theta)}$，因此

$$\log P(Y|\theta)=\log P(Y,Z|\theta)-\log P(Z|Y,\theta)$$

令

$$Q(\theta,\theta^i)=\sum_Z \log P(Y,Z|\theta)P(Z|Y,\theta^i)$$
$$H(\theta,\theta^i)=\sum_Z \log P(Z|Y,\theta)P(Z|Y,\theta^i)$$

因此，$\log P(Y|\theta)=Q(\theta,\theta^i)-H(\theta,\theta^i)$，所以：

$$\begin{aligned} L(\theta^{i+1})-L(\theta^i) &= \log P(Y|\theta^{i+1})-\log P(Y|\theta^i) \\ &= [Q(\theta^{i+1},\theta^i)-Q(\theta^i,\theta^i)]-[H(\theta^{i+1},\theta^i)-H(\theta^i,\theta^i)] \end{aligned}$$

由于 θ^{i+1} 使得 $Q(\theta,\theta^i)$ 达到极大，因此 $Q(\theta^{i+1},\theta^i)-Q(\theta^i,\theta^i)\geq 0$。此外：

$$\begin{aligned} H(\theta^{i+1},\theta^i)-H(\theta^i,\theta^i) &= \sum_Z (\log \frac{P(Z|Y,\theta^{i+1})}{P(Z|Y,\theta^i)})P(Z|Y,\theta^i) \\ &\leq \log(\sum_Z \frac{P(Z|Y,\theta^{i+1})}{P(Z|Y,\theta^i)}P(Z|Y,\theta^i)) \\ &\leq \log\sum_Z P(Z|Y,\theta^{i+1})=0 \end{aligned}$$

因此 $L(\theta^{i+1})\geq L(\theta^i)$。当 $Q(\theta,\theta^{'})$ 和 $L(\theta)$ 满足一定条件时（一般情况下很容易达到），EM 算法可以收敛到 $L(\theta)$ 的稳定点（因此不能保证收敛到极大值点）。

1.3 高斯混合模型

EM 算法一个重要应用场景是高斯混合模型。

定义：高斯混合模型是具有如下形式的概率分布模型：

$$P(y|\theta)=\sum_{k=1}^{K}\alpha_k\phi(y|\theta_k)$$

其中，$\alpha_k\geq 0$ 是系数，$\sum_{k=1}^{K}\alpha_k=1$，$\phi(y|\theta_k)=\phi(y|(\mu_k,\sigma_k^2))$ 是高斯分布密度

$$\phi(y|\theta_k) = \frac{1}{\sqrt{2\pi}\sigma_k}\exp(-\frac{(y-\mu_k)^2}{2\sigma_k^2})$$

假设观测数据 $y_1, y_2, ..., y_N$ 有高斯混合模型 $P(y|\theta) = \sum_{k=1}^{K}\alpha_k\phi(y|\theta_k)$ 生成，则可使用 EM 算法估计参数 $\theta = (\alpha_1, \alpha_2, ..., \alpha_K; \theta_1, \theta_2, ..., \theta_K)$ 的值。

该模型的隐变量 γ_{jk} 可定义为：

$$\gamma_{jk} = \begin{cases} 1 & , y_j \in model_k \\ 0 & , else \end{cases}$$

因此，对于完全数据 $(y_j, \gamma_{j1}, \gamma_{j2}, ..., \gamma_{jK}), j = 1, 2, ..., N$，完全数据的似然函数：

$$\begin{aligned} P(y,\gamma|\theta) &= \prod_{j=1}^{N} P(y_j, \gamma_{j1}, \gamma_{j2}, ..., \gamma_{jK}|\theta) \\ &= \prod_{k=1}^{K}\prod_{j=1}^{N}[\alpha_k\phi(y_j|\theta_k)]^{\gamma_{jk}} \\ &= \prod_{k=1}^{K}\alpha_k^{n_k}\prod_{j=1}^{N}[\phi(y_j|\theta_k)]^{\gamma_{jk}} \\ &= \prod_{k=1}^{K}\alpha_k^{n_k}\prod_{j=1}^{N}[\frac{1}{\sqrt{2\pi}\sigma_k}\exp(-\frac{(y_j-\mu_k)^2}{2\sigma_k^2})]^{\gamma_{jk}} \end{aligned}$$

其中，$n_k = \sum_{j=1}^{N}\gamma_{jk}, \sum_{k=1}^{K} = N$。因此，完全数据的对数似然函数是：

$$\log P(y,\gamma|\theta) = \sum_{k=1}^{K}\{n_k\log\alpha_k + \sum_{j=1}^{N}\gamma_{jk}[\log(\frac{1}{\sqrt{2\pi}}) - \log\sigma_k - \frac{1}{2\sigma_k^2}(y_j-\mu_k)^2]\}$$

所以，高斯混合模型的 **E 步**为确定 Q 函数：

$$\begin{aligned} Q(\theta,\theta^i) &= E_Z[\log P(Y,Z|\theta)|Y,\theta^i] \\ &= E_{\gamma_{jk}}[\sum_{k=1}^{K}\{n_k\log\alpha_k + \sum_{j=1}^{N}\gamma_{jk}[\log(\frac{1}{\sqrt{2\pi}}) - \log\sigma_k - \frac{1}{2\sigma_k^2}(y_j-\mu_k)^2]\}] \\ &= \sum_{k=1}^{K}\{\sum_{j=1}^{N}(E\gamma_{jk})\log\alpha_k + \sum_{j=1}^{N}(E\gamma_{jk})[\log(\frac{1}{\sqrt{2\pi}}) - \log\sigma_k - \frac{1}{2\sigma_k^2}(y_j-\mu_k)^2]\} \end{aligned}$$

其中，

$$
\begin{aligned}
\hat{\gamma}_{jk} &= E(\gamma_{jk}|y,\theta) = P(\gamma_{jk}=1|y,\theta) \\
&= \frac{P(\gamma_{jk}=1, y_j|\theta)}{P(y_j|\theta)} \\
&= \frac{P(\gamma_{jk}=1, y_j|\theta)}{\sum_{k=1}^{K} P(\gamma_{jk}=1, y_j|\theta)} \\
&= \frac{P(y_j|\gamma_{jk}=1,\theta)P(\gamma_{jk}=1|\theta)}{\sum_{k=1}^{K} P(y_j|\gamma_{jk}=1,\theta)P(\gamma_{jk}=1|\theta)} \\
&= \frac{\alpha_k \phi(y_j|\theta)}{\sum_{k=1}^{K} \alpha_k \phi(y_j|\theta)}, \quad j=1,2,...,N; k=1,2,...,K
\end{aligned}
$$

代入即有：

$$
Q(\theta,\theta^i) = \sum_{k=1}^{K}\{\sum_{j=1}^{N}\hat{\gamma}_{jk}\log\alpha_k + \sum_{j=1}^{N}\hat{\gamma}_{jk}[\log(\frac{1}{\sqrt{2\pi}}) - \log\sigma_k - \frac{1}{2\sigma_k^2}(y_j-\mu_k)^2]\}
$$

高斯混合模型的 **M 步**即为最大化函数 $Q(\theta,\theta^i)$。添加上约束条件 $\sum_{k=1}^{K}\alpha_k = 1$，使用拉格朗日乘数法可以得到以下估计：

$$
\begin{aligned}
\hat{\mu}_k &= \frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}y_j}{\sum_{j=1}^{N}\hat{\gamma}_{jk}} &&, k=1,2,...,K \\
\hat{\sigma}_k^2 &= \frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}(y_j-\mu_k)^2}{\sum_{j=1}^{N}\hat{\gamma}_{jk}} &&, k=1,2,...,K \\
\hat{\alpha}_k &= \frac{n_k}{N} = \frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}}{N} &&, k=1,2,...,K
\end{aligned}
$$

重复 E 步和 M 步，直到对数似然函数值不再有明显的变化为止。高斯混合模型的 EM 算法步骤如下：

输入：观测数据 $y_1, y_2, ..., y_N$ 和分模型个数 K。

1. 取参数的初始值开始迭代
2. E 步：依据当前模型参数，计算分模型 k 对观测数据 y_j 的响应度

$$
\hat{\gamma}_{jk} = \frac{\alpha_k \phi(y_j|\theta_k)}{\sum_{k=1}^{K}\alpha_k\phi(y_j|\theta_k)}, \quad j=1,2,...,N; k=1,2,...,K
$$

3. M 步：计算新一轮迭代的模型参数

$$
\begin{aligned}
\hat{\mu}_k &= \frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}y_j}{\sum_{j=1}^{N}\hat{\gamma}_{jk}} &&, k=1,2,...,K \\
\hat{\sigma}_k^2 &= \frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}(y_j-\mu_k)^2}{\sum_{j=1}^{N}\hat{\gamma}_{jk}} &&, k=1,2,...,K \\
\hat{\alpha}_k &= \frac{n_k}{N} = \frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}}{N} &&, k=1,2,...,K
\end{aligned}
$$

4. 重复步骤 2 和 3，直到收敛

输出：高斯混合模型的参数。

2 代码实现

本次使用的数据为-67,-48,6,8,14,16,23,24,28,29,41,49,56,60,75。分模型个数为 2。

2.1 sklearn 实现

准备数据：

```
import numpy as np
X = np.array([-67,-48,6,8,14,16,23,24,28,29,41,49,56,60,75]).reshape(-1,1)
```

高斯混合模型的接口位于 sklearn.mixture.GaussianMixture, 其文档可见**此处**。其中较为重要的参数有：

- n_components：分模型个数
- max_iter：最大迭代次数
- weights_init,means_init：自主设置初始化参数

```
from sklearn.mixture import GaussianMixture
clf = GaussianMixture(n_components=2)
clf.fit(X)
```

```
print('权重：',clf.weights_,'\n平均值：\n',clf.means_,'\n方差：\n',clf.covariances_[:,:,0])
```

```
## 权重： [0.13317238 0.86682762]
## 平均值：
##  [[-57.51107027]
##  [ 32.98489643]]
## 方差：
```

```
## [[ 90.24987882]
## [429.45764867]]
```

2.2 高斯混合模型

此处只实现一维情况下的高斯混合模型。同时模型的初值是随机从原始数据中选取两个值得到的。

```
import math
import random

class GaussianMixture:

    def __init__(self,n_components,esp=1e-3):
        self.n_components = n_components
        self.esp = esp

    def gaussian(self,x,mu,sigma):
        return 1/math.sqrt(2*math.pi*sigma)*math.exp(-(x-mu)**2/(2*sigma))


    def fit(self,X):
        self.alpha = np.array([1/self.n_components for _ in range(self.n_components)])
        self.mean = np.array(random.sample(list(X),2)).reshape(-1,1)
        self.sigma = np.array([np.var(X).reshape(1,1)/self.n_components for _ in range(self.n_comp
        err = self.esp+1

        while err>self.esp:
            mean = self.mean
            sigma = self.sigma
            alpha = self.alpha

            ## compute gamma
            gamma = []
            for k in range(self.n_components):
                for j in range(len(X)):
                    gamma.append(self.gaussian(X[j,:],self.mean[k],self.sigma[k]))
            gamma = np.array(gamma).reshape(self.n_components,-1)
            gamma = gamma/np.sum(gamma,axis=0)
```

```
        ## compute mu,sigma,alpha
        self.mean = np.array([np.sum(gamma[k,:].reshape(-1,1)*X)/np.sum(gamma[k,:]) for k in r
        self.sigma = np.array([(np.sum(gamma[k,:].reshape(-1,1)*(X-self.mean[k])**2)\
                               /np.sum(gamma[k,:]))\
                               for k in range(self.n_components)])
        self.alpha = (np.sum(gamma,axis=1)/X.shape[0]).reshape(-1)

        err = np.mean(np.abs(mean-self.mean))+np.mean(np.abs(sigma-self.sigma))+\
                np.mean(np.abs(alpha-self.alpha))
```

在不同的种子下，EM 算法输出的结果不同：

```
random.seed(15)
clf = GaussianMixture(2)
clf.fit(X)
print('权重：',clf.alpha,'\n平均值：\n',clf.mean,'\n方差：\n',clf.sigma)
```

```
## 权重： [0.86669097 0.13330903]
## 平均值：
##  [ 32.99771897 -57.50167052]
## 方差：
##  [428.48138525  90.24999966]
```

```
random.seed(25)
clf = GaussianMixture(2)
clf.fit(X)
print('权重：',clf.alpha,'\n平均值：\n',clf.mean,'\n方差：\n',clf.sigma)
```

```
## 权重： [0.4532828 0.5467172]
## 平均值：
##  [ 6.61029534 32.80855304]
## 方差：
##  [2127.19989767  357.31428709]
```