M步解法 $\langle \pi, A, \phi \rangle$

解最大似然方程，首先定义拉格朗日式：

$$L = Q(\theta,\theta') + \lambda_1\left(\sum_{k=1}^{K}\pi_k - 1\right) + \sum_{j=1}^{K}\lambda_2^j\left(\sum_{l=1}^{K}\Lambda_{jl} - 1\right)$$

求解初始状态概率为：

$$\frac{\partial L}{\partial \pi_k} = \frac{\gamma(z_{1k})}{\pi_k} + \lambda_1 = 0$$

$$\frac{\partial L}{\partial \lambda_1} = \sum_{k=1}^{K}\pi_k - 1 = 0$$

$$\pi_k = \frac{\gamma(z_{1,k})}{\sum_{j=1}^{K}\gamma(z_{1,j})} = \frac{\alpha(z_{1k})\beta(z_{1k})}{\sum_{j=1}^{K}\alpha(z_{1k})\beta(z_{1k})}$$

同理，求解状态转换概率为：

$$\frac{\partial L}{\partial \Lambda_{jk}} = \frac{\sum_{n=2}^{N}\vartheta(z_{n-1,j}, z_{n,k})}{\Lambda_{jk}} + \lambda_2^j = 0$$

$$\frac{\partial L}{\partial \lambda_2^j} = \sum_{l=1}^{K}\Lambda_{jl} - 1 = 0$$

← E步[illegible]

$$\Lambda_{jk} = \frac{\sum_{n=2}^{N}\vartheta(z_{n-1,j}, z_{n,k})}{\sum_{n=2}^{N}\sum_{l=1}^{K}\vartheta(z_{n-1,j}, z_{n,l})}$$

这个过程用Python代码表示：

```
# M步骤，估计参数
self.start_prob = post_state[0] / np.sum(post_state[0])
for k in range(self.n_state):
    self.transmat_prob[k] = post_adj_state[k] / np.sum(post_adj_state[k])
```

不同类型的发射概率计算

下面我们解决不同类型的发射概率计算。

发射概率$P(x_n|\emptyset_k)$为高斯分布时$N(x_n|\mu_k,\Sigma_k)$

$$N(x_n|\mu_k,\Sigma_k)=\frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma_k|^{1/2}}exp\left(-\frac{1}{2}(x_n-\mu_k)^T{\Sigma_k}^{-1}(x_n-\mu_k)\right)$$

均值求解：

$$\frac{\partial L}{\partial \mu_k}=\sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_k)=0$$

$$\mu_k=\frac{\sum_{n=1}^{N}\gamma(z_{nk})x_n}{\sum_{n=1}^{N}\gamma(z_{nk})}$$

同理协方差求解：

$$\Sigma_k=\frac{\sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_k)(x_n-\mu_k)^T}{\sum_{n=1}^{N}\gamma(z_{nk})}$$

相关Python代码：


```python
    def emit_prob_updated(self, X, post_state): # 更新发射概率
        for k in range(self.n_state):
            for j in range(self.x_size):
                self.emit_means[k][j] = np.sum(post_state[:,k] *X[:,j]) / np.sum(post_state[:,k])

            X_cov = np.dot((X-self.emit_means[k]).T, (post_state[:,k]*(X-self.emit_means[k]).T).T)
            self.emit_covars[k] = X_cov / np.sum(post_state[:,k])
            if det(self.emit_covars[k]) == 0: # 对奇异矩阵的处理
                self.emit_covars[k] = self.emit_covars[k] + 0.01*np.eye(len(X[0]))
```

关于离散概率分布函数的更新，离散概率分布类似于一个表格，观测值x只能包含有限的特定值，而离散概率分布表示为由某状态得到某观测值的概率。由此我们重新定义拉格朗日式，这里增加的一项指某状态生成所有观测值的概率之和应该为1。

$$L=Q(\theta,\theta')+\lambda_1\left(\sum_{k=1}^{K}\pi_k-1\right)+\lambda_2\left(\sum_{j=1}^{K}\sum_{k=1}^{K}\Lambda_{jk}-1\right)+\sum_{k=1}^{K}\lambda_3^k\left(\sum_{j=1}^{J}P(x_j|z_k)-1\right)$$

关于离散概率分布函数的更新，离散概率分布类似于一个表格，观测值x只能包含有限的特定值，而离散概率分布表示为由某状态得到某观测值的概率。由此我们重新定义拉格朗日式，这里增加的一项指某状态生成所有观测值的概率之和应该为1。

$$L = Q(\theta,\theta') + \lambda_1\left(\sum_{k=1}^{K}\pi_k - 1\right) + \lambda_2\left(\sum_{j=1}^{K}\sum_{k=1}^{K}\Lambda_{jk} - 1\right) + \sum_{k=1}^{K}\lambda_3^k\left(\sum_{j=1}^{J}P\left(x_j|z_k\right) - 1\right)$$

然后我们求解离散概率分布函数:

$$\frac{\partial L}{\partial P\left(x_j|z_k\right)} = \sum_{x_n=x_j}\frac{\gamma(z_{nk})}{P\left(x_j|z_k\right)} + \lambda_3^k = 0$$

$$\frac{\partial L}{\partial \lambda_3^k} = \sum_{j=1}^{J}P\left(x_j|z_k\right) - 1 = 0$$

$$P\left(x_j|z_k\right) = -\frac{\sum_{x_n=x_j}\gamma(z_{nk})}{\lambda_3^k}$$

$$\sum_{j=1}^{J}-\frac{\sum_{x_n=x_j}\gamma(z_{nk})}{\lambda_3^k} - 1 = 0$$

$$\lambda_3^k = -\sum_{j=1}^{J}\sum_{x_n=x_j}\gamma(z_{nk})$$

$$P\left(x_j|z_k\right) = -\frac{\sum_{x_n=x_j}\gamma(z_{nk})}{\lambda_3^k} = \frac{\sum_{x_n=x_j}\gamma(z_{nk})}{\sum_{j=1}^{J}\sum_{x_n=x_j}\gamma(z_{nk})}$$

相关Python代码为:

```
def emit_prob_updated(self, X, post_state): # 更新发射概率
    self.emission_prob = np.zeros((self.n_state, self.x_num))
    X_length = len(X)
    for n in range(X_length):
        self.emission_prob[:,int(X[n])] += post_state[n]

    self.emission_prob+= 0.1/self.x_num
    for k in range(self.n_state):
        if np.sum(post_state[:,k])==0: continue
        self.emission_prob[k] = self.emission_prob[k]/np.sum(post_state[:,k])
```