# 基于高斯混合模型的独立向量分析

FanmingL

在经典的IVA(Independent Vector Analysis)[1][2]中，我们根据经验，假设声源信号满足超高斯分布，然后用一个非常简单的多元拉普拉斯分布作为声源的声学模型，即

$$q(\mathbf{y_i}) = \alpha\left(-\sqrt{\sum_{\mathbf{k=1}}^{\mathbf{K}} |\mathbf{y_i^{(k)}}|^2}\right)$$

但是，一般情况下，不同声源的概率模型未必都是这样一个多元拉普拉斯分布，都用相同的多元拉普拉斯分布去作为声源的概率模型很明显是不准确的[3]。于是我们想到用GMM(Gaussian Mixture Model)去拟合声源概率模型，理论上只要有足够多的数量和合适的参数，GMM能够近似一切连续的声源概率模型。考虑两个声源两个麦克风的情况，假设声源相互独立，声源概率模型可以这样表示，

$$p\left(\mathbf{X}_1,\ldots,\mathbf{X}_K\right) = \prod_{j=1}^{2} p(X_{j1},\ldots,p_{jK})$$

其中

$$p\left(X_{j1},\ldots,X_{jK}\right) = \sum_{s_j} p(s_j)\prod_k N(X_{jk};0,v_{ks_j})$$

可得总的表达式为

$$p\left(\mathbf{X}_1,\ldots,\mathbf{X}_K\right) = \left(\sum_{s_1} p(s_1)\prod_k N(X_{1k};0,v_{ks_1})\right)\left(\sum_{s_2} p(s_2)\prod_k N(X_{2k};0,v_{ks_2})\right)$$

$$p\left(\mathbf{X}_1,\ldots,\mathbf{X}_K\right) = \sum_{s_1}\sum_{s_2} p(s_1)p(s_2)\prod_k N(X_{1k};0,v_{ks_1})\prod_k N(X_{2k};0,v_{ks_2})$$

$$p\left(\mathbf{X}_1,\ldots,\mathbf{X}_K\right) = \sum_{s_1}\sum_{s_2} p(s_1)p(s_2)\prod_k N(X_{1k};0,v_{ks_1})N(X_{2k};0,v_{ks_2})$$

$$p\left(\mathbf{X}_1,\ldots,\mathbf{X}_K\right) = \sum_{s_1}\sum_{s_2} p(s_1)p(s_2)\prod_k N(\mathbf{X}_k;0,\mathbf{\Phi}_{k\mathbf{s}})$$

其中$\mathbf{s} = (s_1, s_2)$,$\mathbf{\Phi}_{k\mathbf{s}} = \begin{pmatrix} v_{ks_1} & 0 \\ 0 & v_{ks_2} \end{pmatrix}$。并需注意，这里$v_{ks_1}, v_{ks_2}$并不是指协方差，而是指协方差的倒数。故$N(\mathbf{X}_k;0,\Phi_{\mathbf{s}_t})$的具体表达式为$N(\mathbf{X}_k;0,\mathbf{\Phi}_{k\mathbf{s}}) = \dfrac{|det\mathbf{\Phi}|^{\frac{1}{2}}}{2\pi}exp(-\dfrac{1}{2}\mathbf{X}_k^{\dagger}\mathbf{\Phi}_{k\mathbf{s}}\mathbf{X}_k)$。

频域混合模型如下

$$\mathbf{Y}_{kt} = \mathbf{A}_k\mathbf{X}_{kt}$$

## 白化(Whitening)

为了更方便的导出之后的算法，我们需要对观测到的信号进行白化，即去除两个观测信号的二阶相关性，使得对应频率点的向量的协方差矩阵为单位矩阵。论文[3]中提出了如下方法，如下

$$\mathbf{Q}_k = \sum_{t=0}^{T}\mathbf{Y}_{kt}\mathbf{Y}_{kt}^{\dagger}$$

$$\mathbf{Y}_{kt} := \mathbf{Q}_k^{-\frac{1}{2}}\mathbf{Y}_{kt}$$

其中†为Hermite算子(共轭转置)。

可以看出[3]的作者的想法是这样的，首先求出每个频率点的协方差矩阵$\mathbf{Q}_k$，然后对每个频率点的向量左乘$\mathbf{Q}_k^{-\frac{1}{2}}$，但是我觉得这样操作并不能使协方差矩阵单位化，因为此时的协方差矩阵

$$\mathbf{Q}_k^* = \mathbf{Q}_k^{-\frac{1}{2}}\left(\sum_{t=0}^{T}\mathbf{Y}_{kt}\mathbf{Y}_{kt}^\dagger\right)\mathbf{Q}_k^{-\frac{1}{2}\dagger} = \mathbf{Q}_k^{-\frac{1}{2}}\mathbf{Q}_k\mathbf{Q}_k^{-\frac{1}{2}\dagger}$$

其中，$\mathbf{Q}_k^*$为经过上述算法白化之后的协方差矩阵。

这个式子的计算结果并不是一个单位矩阵，我不是很能理解这个白化。

实际上批处理的白化相当简单，首先求出变量的协方差矩阵$\mathbf{C}_{\mathbf{Y}_{kt}} = E[\mathbf{Y}_{kt}\mathbf{Y}_{kt}^\dagger]$，求出$\mathbf{C}_{\mathbf{Y}_{kt}}$的特征向量矩阵$\mathbf{E}$，以及以$\mathbf{C}_{\mathbf{Y}_{kt}}$的特征值为对角元素的对角矩阵$\mathbf{D}$，线性白化变换即为[4]，

$$\mathbf{V}_k = \mathbf{D}^{-\frac{1}{2}}\mathbf{E}^\dagger$$

且参考[4]可以用如下方式实现在线白化

$$\mathbf{Y}_{kt} := \mathbf{V}_k\mathbf{Y}_{kt}$$

$$\mathbf{V}_k := \mathbf{V}_k + \gamma(\mathbf{I} - \mathbf{Y}_{kt}\mathbf{Y}_{kt}^\dagger)\mathbf{V}_k$$

其中$\mathbf{V}_k$为白化变换矩阵，可以看出在平衡点处，当$\mathbf{V}_k$的值的变化平均为零时，下式成立

$$(\mathbf{I} - E[\mathbf{Y}_{kt}\mathbf{Y}_{kt}^\dagger])\mathbf{V}_k = 0$$

此时，$E[\mathbf{Y}_{kt}\mathbf{Y}_{kt}^\dagger] = \mathbf{I}$，所以$\mathbf{V}_k$的确为$\mathbf{Y}_{kt}$的一个白化矩阵。

由于白化之后$\mathbf{I} = E[\mathbf{Y}_{kt}\mathbf{Y}_{kt}^\dagger] = E[\mathbf{A}_k\mathbf{X}_{kt}\mathbf{X}_{kt}^\dagger\mathbf{A}_k^\dagger] = \mathbf{A}_k E[\mathbf{X}_{kt}\mathbf{X}_{kt}^\dagger]\mathbf{A}_k^\dagger$，这里我们先假设$E[\mathbf{X}_{kt}\mathbf{X}_{kt}^\dagger] = \mathbf{I}$，即单位矩阵，在最后解混之后我们再对$\mathbf{X}_{kt}$的大小进行调整。所以，$\mathbf{I} = \mathbf{A}_k E[\mathbf{X}_{kt}\mathbf{X}_{kt}^\dagger]\mathbf{A}_k^\dagger = \mathbf{A}_k\mathbf{A}_k^\dagger$，即$\mathbf{A}_k$是酉矩阵(unitary matrix)，所以解混矩阵$\mathbf{W}_k = \mathbf{A}_k^{-1}$也为酉矩阵。酉矩阵性质很好，它可以如下表示。

$$\mathbf{W}_k = \begin{pmatrix} a_k & b_k \\ -b_k^* & a_k^* \end{pmatrix} \qquad s.t.\ a_k a_k^* + b_k b_k^* = 1$$

正是因为酉矩阵的这个性质，只要确定两个变量就能求出解混矩阵，这对后面的推导大有裨益。

下面由式()和式()推导$p_y\left(\mathbf{Y}_1,\ldots,\mathbf{Y}_K\right)$，因为有线性变换$\mathbf{Y}_{kt} = \mathbf{A}_k\mathbf{X}_{kt}$，所以

$$p_y\left(\mathbf{Y}_1,\ldots,\mathbf{Y}_K\right) = \left(\prod_k \frac{1}{|det\mathbf{A}_k|}\right)p_x(\mathbf{A}_1^{-1}\mathbf{Y}_1,\ldots,\mathbf{A}_K^{-1}\mathbf{Y}_K)$$

$$p_y\left(\mathbf{Y}_1,\ldots,\mathbf{Y}_K\right) = p_x(\mathbf{W}_1\mathbf{Y}_1,\ldots,\mathbf{W}_K\mathbf{Y}_K)$$

$$p_y\left(\mathbf{Y}_1,\ldots,\mathbf{Y}_K\right) = \sum_{s_1}\sum_{s_2} p(s_1)p(s_2)\prod_k N(\mathbf{W}_k\mathbf{Y}_k; 0, \mathbf{\Phi}_{k\mathbf{s}})$$

$$p_y\left(\mathbf{Y}_1,\ldots,\mathbf{Y}_K\right) = \sum_{s_1}\sum_{s_2} p(s_1)p(s_2)\prod_k \frac{1}{2\pi|\mathbf{\Phi}_{k\mathbf{s}}|^{-\frac{1}{2}}}exp(-\frac{1}{2}(\mathbf{W}_k\mathbf{Y}_k)^\dagger\mathbf{\Phi}_{ks}(\mathbf{W}_k\mathbf{Y}_k))$$

$$p_y\left(\mathbf{Y}_1,\ldots,\mathbf{Y}_K\right) = \sum_{s_1}\sum_{s_2} p(s_1)p(s_2)\prod_k N(\mathbf{Y}_k; 0, \mathbf{\Sigma}_{k\mathbf{s}})$$

其中$\mathbf{s} = (s_1, s_2)$,$\mathbf{\Phi}_{k\mathbf{s}} = \begin{pmatrix} v_{ks_1} & 0 \\ 0 & v_{ks_2} \end{pmatrix}$,$\mathbf{\Sigma}_{\mathbf{ks}} = \mathbf{W}_k^\dagger\mathbf{\Phi}_{k\mathbf{s}}\mathbf{W}_k$.

由此可以得到极大似然函数

$$L(\theta) = \sum_t log\left(p_y(\mathbf{Y}_{1t},\ldots,\mathbf{Y}_{Kt})\right)$$

$$L(\theta)=\sum_t log\left(\frac{\sum_{s_{1t}}\sum_{s_{2t}}p(s_{1t})p(s_{2t})\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})q(\mathbf{s}_t)}{q(\mathbf{s}_t)}\right)$$

$$L(\theta)=\sum_t log\left(E\left[\frac{p(s_{1t})p(s_{2t})\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})}{q(\mathbf{s}_t)}\right]\right)$$

$$L(\theta)\geq\sum_t E\left[log\left(\frac{p(s_{1t})p(s_{2t})\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})}{q(\mathbf{s}_t)}\right)\right]=\sum_t\sum_{\mathbf{s}_t}q(\mathbf{s}_t)log\left(\frac{p(s_{1t})p(s_{2t})\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})}{q(\mathbf{s}_t)}\right)=F(q,\theta)$$

令$p(\mathbf{s}_t)=p(\mathbf{s}_{1t})p(\mathbf{s}_{2t})$，上式简化为

$$F(q,\theta)=\sum_t\sum_{\mathbf{s}_t}q(\mathbf{s}_t)log\left(\frac{p(\mathbf{s}_t)\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})}{q(\mathbf{s}_t)}\right)$$

## EM算法(Expectation-Maximization Algorithm)

下面开始用EM算法来求模型参数$\theta=(\mathbf{W}_k,v_{ks_j},p(s_j))$。

**E-step**,寻找$q(\mathbf{s}_t)$使得$F(q,\theta)$成为$L(\theta)$的一个紧的下界，为了使得上式中$log$括号中的式子恒为常数，我们令$q(\mathbf{s}_t)\propto p(s_{1t})p(s_{2t})\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})$，考虑到$q(\mathbf{s}_t)$也是一个概率分布，所以需要满足$\sum_{\mathbf{s}_t}q(\mathbf{s}_t)=1$，所以我们令

$$q(\mathbf{s}_t)=\frac{p(\mathbf{s}_t)\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})}{\sum_{\mathbf{s}'_t}p(\mathbf{s}'_t)\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}'_t})}$$

至此，我们推导完了E-step，事实上，在E-Step上，批算法和在线算法的唯一区别就是，批算法需要对每一帧都计算$q(\mathbf{s}_t)$，而在线算法只需要计算当前帧的，因为在M-Step，批处理算法需要用到所有帧的$q(\mathbf{s}_t)$去估计模型参数$\theta=(\mathbf{W}_k,v_{ks_j},p(s_j))$。

**M-Step**，此步的目的是寻找

$$\underset{\theta}{argmax}\quad F(q,\theta)$$

## 批算法(batch algorithm)

首先找到

$$\underset{\mathbf{W}_k}{argmax}\quad F(q,\theta)\qquad s.t.\quad det\mathbf{W}_k=1$$

$$F(q,\theta)=\sum_t\sum_{\mathbf{s}_t}q(\mathbf{s}_t)log\left(\frac{p(\mathbf{s}_t)\prod_k N(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})}{q(\mathbf{s}_t)}\right)$$

$$F(q,\theta)=\sum_t\sum_{\mathbf{s}_t}q(\mathbf{s}_t)\left[log(p(\mathbf{s}_t))+\sum_k logN(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})+log(q(\mathbf{s}_t))\right]$$

考虑到中括号中1、3项均与$\mathbf{W}_k$无关，故可以看成常量，所以

$$F(q,\theta)=\sum_t\sum_{\mathbf{s}_t}q(\mathbf{s}_t)\left[\sum_k logN(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})\right]$$

$$F(q,\theta)=const.+\sum_t\sum_{\mathbf{s}_t}\sum_k q(\mathbf{s}_t)logN(\mathbf{Y}_{kt};0,\mathbf{\Sigma}_{k\mathbf{s}_t})$$

$$F(q,\theta)=const.+\sum_t\sum_{\mathbf{s}_t}\sum_k q(\mathbf{s}_t)(-\frac{1}{2}\mathbf{Y}_{kt}^\dagger\mathbf{\Sigma}_{\mathbf{s}_t}\mathbf{Y}_{kt})$$

$$F(q,\theta)=const.+\sum_t\sum_{\mathbf{s}_t}\sum_k q(\mathbf{s}_t)(-\frac{1}{2}\mathbf{Y}_{kt}^\dagger\mathbf{W}_k^\dagger\mathbf{\Phi}_{k\mathbf{s}_t}\mathbf{W}_k\mathbf{Y}_{kt})$$

这里作如下变换

$$\mathbf{\Phi}_{k\mathbf{s}_t}=\begin{pmatrix}v_{ks_1} & 0\\ 0 & v_{ks_2}\end{pmatrix}=\begin{pmatrix}v_{ks_1}-v_{ks_2} & 0\\ 0 & 0\end{pmatrix}+\begin{pmatrix}v_{ks_2} & 0\\ 0 & v_{ks_2}\end{pmatrix}=\mathbf{\Phi}_{k\mathbf{s}_t}^{(1)}+\mathbf{\Phi}_{k\mathbf{s}_t}^{(2)}$$

式()可化为

$$F(q,\theta)=const.+\sum_t\sum_{\mathbf{s}_t}\sum_k q(\mathbf{s}_t)(-\frac{1}{2}\mathbf{Y}_{kt}^\dagger\mathbf{W}_k^\dagger(\mathbf{\Phi}_{k\mathbf{s}_t}^{(1)}+\mathbf{\Phi}_{k\mathbf{s}_t}^{(2)})\mathbf{W}_k\mathbf{Y}_{kt})$$

而

$$\mathbf{Y}_{kt}^\dagger\mathbf{W}_k^\dagger\mathbf{\Phi}_{k\mathbf{s}_t}^{(2)}\mathbf{W}_k\mathbf{Y}_{kt}=v_{ks_2}\mathbf{Y}_{kt}^\dagger\mathbf{W}_k^\dagger\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\mathbf{W}_k\mathbf{Y}_{kt}=v_{ks_2}\mathbf{Y}_{kt}^\dagger\mathbf{Y}_{kt}$$

这样一来包含$\mathbf{\Phi}_{k\mathbf{s}_t}^{(2)}$的项也与$\mathbf{W}_k$无关了，剩下

$$F(q,\theta)=const.+\sum_t\sum_{\mathbf{s}_t}\sum_k q(\mathbf{s}_t)\left(-\frac{1}{2}\mathbf{Y}_{kt}^\dagger\mathbf{W}_k^\dagger\begin{pmatrix}v_{ks_1}-v_{ks_2} & 0\\ 0 & 0\end{pmatrix}\mathbf{W}_k\mathbf{Y}_{kt}\right)$$

将$\mathbf{W}_k=\begin{pmatrix}a_k & b_k\\ -b_k^* & a_k^*\end{pmatrix}$,$\mathbf{Y}_{kt}=\begin{pmatrix}Y_{1kt}\\ Y_{2kt}\end{pmatrix}$,上式展开为

$$F(q,\theta)=const.+\sum_t\sum_{\mathbf{s}_t}\sum_k q(\mathbf{s}_t)\left(-\frac{1}{2}(v_{ks_1}-v_{ks_2})\left[Y_{1kt}Y_{1kt}^*a_ka_k^*+Y_{1kt}Y_{2kt}^*a_kb_k^*+Y_{1kt}^*Y_{2kt}a_k^*b_k+Y_{2kt}Y_{2kt}^*b_kb_k^*\right]\right)$$

为了求$\underset{a_k,b_k}{argmax}\quad F(q,\theta)\qquad s.t.\quad a_ka_k^*+b_kb_k^*=1$，由于存在约束$a_ka_k^*+b_kb_k^*=1$，故引入拉格朗日算子$\beta_k$，问题转化为

$$\underset{a_k,b_k}{argmax}\sum_t\sum_{\mathbf{s}_t}\sum_k q(\mathbf{s}_t)\left(-(v_{ks_1}-v_{ks_2})\left[Y_{1kt}Y_{1kt}^*a_ka_k^*+Y_{1kt}Y_{2kt}^*a_kb_k^*+Y_{1kt}^*Y_{2kt}a_k^*b_k+Y_{2kt}Y_{2kt}^*b_kb_k^*\right]\right)+\beta_k(a_ka_k^*+b_kb_k^*-1)$$

为方便，用$L(q,\theta)$代替上式右侧表达式，

$$\underset{a_k,b_k}{argmax}\quad L(q,\theta)$$

用分别令$L(q,\theta)$对$a_k,b_k$的偏导数为零得到

$$\frac{\partial L(q,\theta)}{\partial a_k}=-\sum_t\sum_{\mathbf{s}_t}(v_{ks_1}-v_{ks_2})q(\mathbf{s}_t)\left[Y_{1k}Y_{1k}^*a_k^*+Y_{1k}Y_{2k}^*b_k^*\right]+\beta_kb_k^*=0$$

$$\frac{\partial L(q,\theta)}{\partial b_k}=-\sum_t\sum_{\mathbf{s}_t}(v_{ks_1}-v_{ks_2})q(\mathbf{s}_t)\left[Y_{1k}^*Y_{2k}a_k^*+Y_{2k}Y_{2k}^*b_k^*\right]+\beta_kb_k^*=0$$

所以

$$\nabla_{\begin{pmatrix}a_k\\ b_k\end{pmatrix}}L(q,\theta)=-\sum_t\sum_{\mathbf{s}_t}(v_{ks_1}-v_{ks_2})q(\mathbf{s}_t)\begin{pmatrix}Y_{1k}Y_{1k}^*a_k^*+Y_{1k}Y_{2k}^*b_k^*\\ Y_{1k}^*Y_{2k}a_k^*+Y_{2k}Y_{2k}^*b_k^*\end{pmatrix}+\beta_k\begin{pmatrix}a_k^*\\ b_k^*\end{pmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}$$

整理可得

$$\nabla_{\begin{pmatrix}a_k\\ b_k\end{pmatrix}}L(q,\theta)=-\sum_t\sum_{\mathbf{s}_t}(v_{ks_1}-v_{ks_2})q(\mathbf{s}_t)\begin{pmatrix}Y_{1k}\\ Y_{2k}\end{pmatrix}\begin{pmatrix}Y_{1k}^* & Y_{2k}^*\end{pmatrix}\begin{pmatrix}a_k^*\\ b_k^*\end{pmatrix}+\beta_k\begin{pmatrix}a_k^*\\ b_k^*\end{pmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}$$

$$\sum_t\sum_{\mathbf{s}_t}(v_{ks_1}-v_{ks_2})q(\mathbf{s}_t)\begin{pmatrix}Y_{1k}\\ Y_{2k}\end{pmatrix}\begin{pmatrix}Y_{1k}^* & Y_{2k}^*\end{pmatrix}\begin{pmatrix}a_k^*\\ b_k^*\end{pmatrix}=\beta_k\begin{pmatrix}a_k^*\\ b_k^*\end{pmatrix}$$

$$\sum_t \sum_{\mathbf{s}_t} (v_{ks_1} - v_{ks_2}) q(\mathbf{s}_t) \mathbf{Y}_{kt} \mathbf{Y}_{kt}^\dagger \begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix} = \beta_k \begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix}$$

令

$$\mathbf{M}_{kT} = \sum_t \sum_{\mathbf{s}_t} (v_{ks_1} - v_{ks_2}) q(\mathbf{s}_t) \mathbf{Y}_{kt} \mathbf{Y}_{kt}^\dagger$$

代入上式，

$$\mathbf{M}_{kT} \begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix} = \beta_k \begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix}$$

这是一个特征值求解问题，二维矩阵$\mathbf{M}_{kT}$有两个特征值，这里我们取较小的特征值作为$\beta_k$，将$\mathbf{M}_{kT}$如下表示

$$\mathbf{M}_{kT} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

我们可以计算得到，较小的特征值$\beta_k$为

$$\beta_k = \frac{M_{11} + M_{22}}{2} - \sqrt{\frac{(M_{11} - M_{22})^2}{4} + |M_{12}|^2}$$

$$\begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix} = \frac{1}{\sqrt{1 + (\frac{\beta_k - M_{11}}{M_{12}})^2}} \begin{pmatrix} 1 \\ \frac{\beta_k - M_{11}}{M_{12}} \end{pmatrix}$$

至此，我们推出了$\mathbf{W}_k$的更新公式。

为了方便的更新$(v_{ks_j}, p(s_j))$，我们首先需要先求出联合概率分布$q(\mathbf{s}_t)$的边缘分布$q(s_{1t}), q(s_{2t})$。即

$$q(s_{1t}) = \sum_{s_{2t}} q(\mathbf{s}_t)$$

$$q(s_{2t}) = \sum_{s_{1t}} q(\mathbf{s}_t)$$

$(v_{ks_j}, p(s_j))$两个量可以用经典的GMM的EM算法求得[3][5]。

$$\frac{1}{v_{ks_j=r}} = \frac{\left[\sum_t q(s_{jt} = r) \mathbf{W}_k \mathbf{Y}_{kt} \mathbf{Y}_{kt}^\dagger \mathbf{W}_k^\dagger\right]_{jj}}{\sum_t q(s_{jt} = r)}$$

$$p(s_j = r) = \frac{\sum_{t=1}^T q(s_{jt} = r)}{T}$$

所以，批EM算法可以简单描述如下：

E-Step：更新每一个时间点$t$的每一个高斯混合序号组合$\mathbf{s}_t = (s_1, s_2)$，

$$q(\mathbf{s}_t) := \frac{p(\mathbf{s}_t) \prod_k N(\mathbf{Y}_{kt}; 0, \mathbf{\Sigma}_{k\mathbf{s}_t})}{\sum_{\mathbf{s}_t'} p(\mathbf{s}_t') \prod_k N(\mathbf{Y}_{kt}; 0, \mathbf{\Sigma}_{k\mathbf{s}_t'})}$$

M-Step：对每个频率点，

1、$\mathbf{M}_{kT} := \sum_t \sum_{\mathbf{s}_t} (v_{ks_1} - v_{ks_2}) q(\mathbf{s}_t) \mathbf{Y}_{kt} \mathbf{Y}_{kt}^\dagger$

2、$\beta_k := \frac{M_{11} + M_{22}}{2} - \sqrt{\frac{(M_{11} - M_{22})^2}{4} + |M_{12}|^2}$

3、$\begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix} := \frac{1}{\sqrt{1 + (\frac{\beta_k - M_{11}}{M_{12}})^2}} \begin{pmatrix} 1 \\ \frac{\beta_k - M_{11}}{M_{12}} \end{pmatrix}$

4、$\mathbf{W}_k := \begin{pmatrix} a_k & b_k \\ -b_k^* & a_k^* \end{pmatrix}$

5、$q(s_{1t}) = \sum_{s_{2t}} q(\mathbf{s}_t)$；$q(s_{2t}) = \sum_{s_{1t}} q(\mathbf{s}_t)$

6、估计分离信号$\hat{\mathbf{X}}_{kt} := \mathbf{W}_k \mathbf{Y}_{kt}$

7、对每个信号源$s_j$的第$r$个高斯分量更新$\frac{1}{v_{ks_j=r}} := \frac{\sum_t q(s_{jt}=r)\hat{X}_{jkt}\hat{X}_{jkt}^\dagger}{\sum_t q(s_{jt}=r)}$

8、对每个信号源$s_j$的第$r$个高斯分量更新$p(s_j = r) := \frac{\sum_{t=1}^T q(s_{jt}=r)}{T}$

## 在线算法(online algorithm)

可以看出批算法的处理量非常大，特别是当高斯分量的数量增长的时候，$\mathbf{s}_t$的组合会二次增长，而本身E-Step又要计算每种$\mathbf{s}_t$的组合的所有帧的$q(\mathbf{s}_t)$，而每次计算$q(\mathbf{s}_t)$都要算一个二元高斯分布，运算量很大。为了能放在dsp等芯片级的处理器上运行，需要对batch算法进行优化，使其能够在dsp上实时运行。我们需要将批算法中的需要用到所有帧的公式，优化为只需要当前帧。

实际上，要用到所有帧的需要用到的公式为M-Step的1、7、8步，

$\mathbf{M}_{kT} := \sum_t \sum_{\mathbf{s}_t} (v_{ks_1} - v_{ks_2}) q(\mathbf{s}_t) \mathbf{Y}_{kt} \mathbf{Y}_{kt}^\dagger$的优化较为简单，即

$$\mathbf{M}_{kT} := \lambda \mathbf{M}_{kT} + \sum_{\mathbf{s}_t} (v_{ks_1} - v_{ks_2}) q(\mathbf{s}_t) \mathbf{Y}_{kt} \mathbf{Y}_{kt}^\dagger$$

其中$\lambda$的值在0到1之间，代表之前的数据对当前的重要程度。

为了后续推导方便，首先得到对$\sum_{t=1}^T q(s_{jt}=r)$的实时估计

$$m_{jr} := \lambda m_{jr} + q(s_{jt}=r)$$

可以用这次EM算法上式运算前的$m_{jr}$，即$m_{jr-last}$，推出

$$\sum_t q(s_{jt}=r)\hat{X}_{jkt}\hat{X}_{jkt}^\dagger = \frac{1}{v_{ks_j=r}} * m_{jr-last} + q(s_{jt}=r)\hat{X}_{jkt}\hat{X}_{jkt}^\dagger$$

所以

$$\frac{1}{v_{ks_j=r}} := \frac{\sum_t q(s_{jt}=r)\hat{X}_{jkt}\hat{X}_{jkt}^\dagger}{\sum_t q(s_{jt}=r)} = \frac{\left(\frac{1}{v_{ks_j=r}} * m_{jr-last} + q(s_{jt}=r)\hat{X}_{jkt}\hat{X}_{jkt}^\dagger\right)}{m_{jr}}$$

$$\frac{1}{v_{ks_j=r}} := \frac{1}{v_{ks_j=r}} * \frac{m_{jr-last}}{m_{jr}} + \frac{q(s_{jt}=r)\hat{X}_{jkt}\hat{X}_{jkt}^\dagger}{m_{jr}}$$

考虑到$p(s_j)$是一个概率密度分布所以$\sum_r p(s_j = r) = 1$，且$p(s_j = r) \propto m_{jr}$，所以可以得到一个简单的表示

$$p(s_j = r) := \frac{m_{jr}}{\sum_r m_{jr}}$$

所以，在线EM算法可以简单描述如下：

E-Step：每一个高斯混合序号组合$\mathbf{s}_t = (s_1, s_2)$，$q(\mathbf{s}_t) := \frac{p(\mathbf{s}_t)\prod_k N(\mathbf{Y}_{kt}; 0, \mathbf{\Sigma}_{k\mathbf{s}_t})}{\sum_{\mathbf{s}_t'} p(\mathbf{s}_t')\prod_k N(\mathbf{Y}_{kt}; 0, \mathbf{\Sigma}_{k\mathbf{s}_t'})}$

M-Step：

1、$q(s_{1t}) = \sum_{s_{2t}} q(\mathbf{s}_t)$；$q(s_{2t}) = \sum_{s_{1t}} q(\mathbf{s}_t)$

2、$m_{jr-old} := m_{jr}$
3、$m_{jr} := \lambda m_{jr} + q(s_{jt} = r)$

4、对每个信号源$s_j$的第$r$个高斯分量更新$p(s_j = r) := \dfrac{m_{jr}}{\sum_r m_{jr}}$

对每个频率点，

5、$\mathbf{M}_{kT} := \lambda \mathbf{M}_{kT} + \sum_{\mathbf{s}_t} (v_{ks_1} - v_{ks_2}) q(\mathbf{s}_t) \mathbf{Y}_{kt} \mathbf{Y}_{kt}^{\dagger}$

6、$\beta_k := \dfrac{M_{11} + M_{22}}{2} - \sqrt{\dfrac{(M_{11} - M_{22})^2}{4} + |M_{12}|^2}$

7、$\begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix} := \dfrac{1}{\sqrt{1 + (\frac{\beta_k - M_{11}}{M_{12}})^2}} \begin{pmatrix} 1 \\ \frac{\beta_k - M_{11}}{M_{12}} \end{pmatrix}$

8、$\mathbf{W}_k := \begin{pmatrix} a_k & b_k \\ -b_k^* & a_k^* \end{pmatrix}$

9、估计分离信号$\hat{\mathbf{X}}_{kt} := \mathbf{W}_k \mathbf{Y}_{kt}$

10、对每个信号源$s_j$的第$r$个高斯分量更新$\dfrac{1}{v_{ks_j=r}} := \dfrac{1}{v_{ks_j=r}} * \dfrac{m_{jr-last}}{m_{jr}} + \dfrac{q(s_{jt} = r)\hat{X}_{jkt}\hat{X}_{jkt}^{\dagger}}{m_{jr}}$

### 信号重建(Signal Reconstruction)

至此，我们推导出了估计模型参数$\theta = (\mathbf{W}_k, v_{ks_j}, p(s_j))$的在线EM算法和批EM算法，并且源信号$\hat{\mathbf{X}}_{kt}$也已经在算法中将信号分离出来了，但是此时的信号不能直接用于信号重构，我们需要对其大小进行调整，因为

$$\hat{\mathbf{X}} = \mathbf{W}_k \mathbf{V}_k \mathbf{Y}_{kt}$$

且之前有过假设$E[\hat{\mathbf{X}}_{kt}\hat{\mathbf{X}}_{kt}^{\dagger}] = \mathbf{I}$，它的频谱是平的，这是不能用于信号重构的，所以用于重构的信号$\widetilde{\mathbf{X}}_{kt}$应该这样得到

$$\widetilde{\mathbf{X}}_{kt} = diag((\mathbf{W}_k \mathbf{V}_k)^{-1})\hat{\mathbf{X}}_{kt}$$

最后，连续的信号将由overlap-add的方式转至时域。

最后总结整个在线信号分离流程

1、根据经验初始化白化、EM算法中的系数，用经典的EM算法[5]以及一段时长的声源信号训练GMM，直接用训练得到的$v_{ks_j}, p(s_j)$来作为$v_{ks_j}, p(s_j)$的初始值。

2、对输入信号分帧进行快速傅立叶变换(FFT)

3、白化

a. $\mathbf{Y}_{kt} := \mathbf{V}_k \mathbf{Y}_{kt}$
b. $\mathbf{V}_k := \mathbf{V}_k + \gamma(\mathbf{I} - \mathbf{Y}_{kt}\mathbf{Y}_{kt}^{\dagger})\mathbf{V}_k$

4、在线EM算法更新$(\mathbf{W}_k, v_{ks_j}, p(s_j))$，$\hat{\mathbf{X}}_{kt}$

5、信号重建

a. $\widetilde{\mathbf{X}}_{kt} = diag((\mathbf{W}_k \mathbf{V}_k)^{-1})\hat{\mathbf{X}}_{kt}$
b. overlap-add的方式变换至时域

# 仿真结果

## 参考文献

[1] Kim T, Eltoft T, Lee T W. Independent vector analysis: An extension of ICA to multivariate components[C]//International Conference on Independent Component Analysis and Signal Separation. Springer, Berlin, Heidelberg, 2006: 165-172.
[2] Kim T. Real-time independent vector analysis for convolutive blind source separation[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2010, 57(7): 1431-1438.
[3] Hao J, Lee I, Lee T W, et al. Independent vector analysis for source separation using a mixture of gaussians prior[J]. Neural computation, 2010, 22(6): 1646-1673.
[4]A. Hyvärinen and E. Oja, Independent Component Analysis. Hoboken, NJ: Wiley, 2002.
[5] http://cs229.stanford.edu/notes/cs229-notes7b.pdf