T-61.5140 Machine Learning: Advanced Probabilistic Methods Project 2016

Jussi Ojala, 544605, Kunal Ghosh 546247
jussi.k.ojala@aalto.fi, kunal.ghosh@aalto.fi

April 29, 2016

1 Mathematical Derivations

In this project we consider the mixture of two linear model. The model is given by

$$y_t \propto wNormal(x_t^T\phi_1, \sigma_1^2) + (1-w)Normal(x_t^T\phi_2, \sigma_2^2) \tag{1}$$

where the $\phi_1, \phi_2 \in \Re^{K\times 1}$ and their conjugate prior distribution is

$$\begin{aligned} \phi_1|\sigma_1 &\propto Normal(\mu_\phi, \sigma_1^2\Sigma_\phi) &(2)\\ \phi_2|\sigma_2 &\propto Normal(\mu_\phi, \sigma_2^2\Sigma_\phi) &(3)\\ & &(4)\end{aligned}$$

and the conjugate prior distribution for the $\sigma_1, \sigma_2 \in \Re$ is

$$\sigma^2 \propto Inversegamma(\alpha_{\sigma^2}, \beta_{\sigma^2}) \tag{5}$$

where α, β are hyper-parameters and if the ϕ_k are a-priori uncorrelated (as we will use later) then Σ_ϕ is diagonal. In the first sub-section we derive the full posterior distribution and the full log posterior distribution (sometimes also called likelihood). Each data observation $y_t \in \Re, Y \in \Re^{T\times 1}, x_t \in \Re^{K\times 1}, X \in \Re^{T\times K}$ and $t = 1, ..., T$. The prior of the w is given to be

$$w \propto Beta(\alpha_w, \beta_w) \tag{6}$$

1.1 Problem 1: Posterior distributions

In this problem we want to first find the complete posterior distribution of mixture model with two linear components. The model formulation as well as the conjugate priors were given in the project description. For the start the likelihood $p(Data|Parameters) = p(X, Y|\phi_1, \phi_2, \sigma_1, \sigma_2, w)$ for mixture model with linear components, this is given by

$$p(X, Y|\phi_1, \phi_2, \sigma_1, \sigma_2, w) \propto \prod_{t=1}^{T} w * N(y_t|x_t^T\phi_1, \sigma_1) + (1-w) * N(y_t|x_t^T\phi_2, \sigma_2) \tag{7}$$

where the constant term has been left out of the equation. To be able to convert the sum part of the equation to product, latent variable Z is introduced, where

$$z_t = \begin{cases} 1 & \text{if and only if data is coming from first linear model.} \\ 0 & \text{if and only if data is coming from second linear model.} \end{cases}$$

is Bernoulli distributed with parameter w. Utilizing latent variable in the equation, product form of the complete likelihood can be written as follow

$$\begin{aligned} p(X, Y, Z|\phi_1, \phi_2, \sigma_1, \sigma_2, w) &= p(X, Y|Z, \phi_1, \phi_2, \sigma_1, \sigma_2, w)p(Z|\phi_1, \phi_2, \sigma_1, \sigma_2, w) \\ &= p(X, Y|Z, \phi_1, \phi_2, \sigma_1, \sigma_2)p(Z|w) \\ &= \prod_{t=1}^{T} p(x_t, y_t|z_t, \phi_1, \phi_2, \sigma_1, \sigma_2)p(z_t|w) \\ &\propto \prod_{t=1}^{T} [N(y_t|x_t^T\phi_1, \sigma_1)w]^{z_t}[N(y_t|x_t^T\phi_2, \sigma_2)(1-w)]^{1-z_t} \qquad (8)\end{aligned}$$

The full posterior distribution (in this project called posterior likelihood) is then achieved utilising this form (and independences of the variables)

$$\begin{aligned}
p(X,Y,Z,\phi_1,\phi_2,\sigma_1,\sigma_2,w) &= p(X,Y,Z|\phi_1,\phi_2,\sigma_1,\sigma_2,w) * p(\phi_1,\phi_2,\sigma_1,\sigma_2,w)\\
&= p(X,Y,Z|\phi_1,\phi_2,\sigma_1,\sigma_2,w)\\
&* \; p(\phi_1|\sigma_1) * p(\sigma_1) * p(\phi_2|\sigma_2) * p(\sigma_2) * p(w)\\
&\propto \prod_{t=1}^{T}[N(y_t|x_t^T\phi_1,\sigma_1)w]^{z_t}[N(y_t|x_t^T\phi_2,\sigma_2)(1-w)]^{1-z_t}\\
&* \; N(\phi_1|\mu_\phi,\sigma_1^2\Sigma_\phi) * InvGamma(\alpha_{\sigma^2},\beta_{\sigma^2})\\
&* \; N(\phi_2|\mu_\phi,\sigma_2^2\Sigma_\phi) * InvGamma(\alpha_{\sigma^2},\beta_{\sigma^2}) * Beta(\alpha_w,\beta_w) \qquad (9)
\end{aligned}$$

If variables $\phi = (\phi_1, \phi_2)$ are a-priori uncorrelated Σ_ϕ is diagonal and derivation becomes simpler. That formula will be used later, the mathematical formulas are written for completeness below.

$$\begin{aligned}
p(X,Y,Z,\phi_1,\phi_2,\sigma_1,\sigma_2,w) &\propto \prod_{t=1}^{T}[\frac{1}{\sqrt{2\pi\sigma_1^2}}exp(-\frac{(x_t^T\phi_1-y_t)^2}{2\sigma_1^2}) * w]^{z_t}[\frac{1}{\sqrt{2\pi\sigma_2^2}}exp(-\frac{(x_t^T\phi_2-y_t)^2}{2\sigma_2^2})(1-w)]^{1-z_t}\\
&* \; (2\pi\sigma_1^2)^{-0.5K}|\Sigma_\phi|^{-0.5}exp(-\frac{1}{2\sigma_1^2}(\phi_1-\mu_\phi)^T\Sigma_\phi^{-1}(\phi_1-\mu_\phi))\\
&* \; \frac{\beta_{\sigma^2}^{\alpha_{\sigma^2}}}{\Gamma(\alpha_{\sigma^2})}(\sigma_1^2)^{-\alpha_{\sigma^2}-1}exp(-\frac{\beta_{\sigma^2}}{\sigma_1^2})\\
&* \; (2\pi\sigma_2^2)^{-0.5K}|\Sigma_\phi|^{-0.5}exp(-\frac{1}{2\sigma_2^2}(\phi_2-\mu_\phi)^T\Sigma_\phi^{-1}(\phi_2-\mu_\phi))\\
&* \; \frac{\beta_{\sigma^2}^{\alpha_{\sigma^2}}}{\Gamma(\alpha_{\sigma^2})}(\sigma_2^2)^{-\alpha_{\sigma^2}-1}exp(-\frac{\beta_{\sigma^2}}{\sigma_2^2})\\
&* \; \frac{\Gamma(\alpha_w+\beta_w)}{\Gamma(\alpha_w)\Gamma(\beta_w)}w^{\alpha_w-1}(1-w)^{\beta_w-1} \qquad (10)
\end{aligned}$$

We do not try to simplify this, since we will use the distributions and only use this opened up version when necessary, and only from those parts that are necessary to open up. The log posterior distribution is then

$$\begin{aligned}
log(p(X,Y,Z,\phi_1,\phi_2,\sigma_1,\sigma_2,w)) &\propto \sum_{t=1}^{T} z_t * log[N(y_t|x_t^T\phi_1,\sigma_1)w] + (1-z_t) * log[N(y_t|x_t^T\phi_2,\sigma_2)(1-w)]\\
&+ \; log(N(\phi_1|\mu_\phi,\sigma_1^2\Sigma_\phi)) + log(InvGamma(\alpha_{\sigma^2},\beta_{\sigma^2}))\\
&+ \; log(N(\phi_2|\mu_\phi,\sigma_2^2\Sigma_\phi)) + log(InvGamma(\alpha_{\sigma^2},\beta_{\sigma^2}))\\
&= \sum_{t=1}^{T} z_t[-0.5 * log(2\pi\sigma_1^2) - \frac{(x_t^T\phi_1-y_t)^2}{2\sigma_1^2} + log(w)]\\
&+ \; (1-z_t)[-0.5 * log(2\pi\sigma_2^2) - \frac{(x_t^T\phi_2-y_t)^2}{2\sigma_2^2} + log(1-w)]\\
&- \; 0.5K * log(2\pi\sigma_1^2) - 0.5log(|\Sigma_\phi|) - \frac{1}{2\sigma_1^2}(\phi_1-\mu_\phi)^T\Sigma_\phi^{-1}(\phi_1-\mu_\phi)\\
&+ \; (\alpha_{\sigma^2}) * log(\beta_{\sigma^2}) - log(\Gamma(\alpha_{\sigma^2})) + (-\alpha_{\sigma^2}-1) * log(\sigma_1^2) - \frac{\beta_{\sigma^2}}{\sigma_1^2}\\
&- \; 0.5K * log(2\pi\sigma_2^2) - 0.5log(|\Sigma_\phi|) - \frac{1}{2\sigma_2^2}(\phi_2-\mu_\phi)^T\Sigma_\phi^{-1}(\phi_2-\mu_\phi)\\
&+ \; (\alpha_{\sigma^2}) * log(\beta_{\sigma^2}) - log(\Gamma(\alpha_{\sigma^2})) + (-\alpha_{\sigma^2}-1) * log(\sigma_2^2) - \frac{\beta_{\sigma^2}}{\sigma_2^2}\\
&+ \; log(\Gamma(\alpha_w+\beta_w)) - log(\Gamma(\alpha_w)) - log(\Gamma(\beta_w))\\
&+ \; (\alpha_w-1) * log(w) + (\beta_w-1) * log(1-w) \qquad (11)
\end{aligned}$$

1.2 Update equations for (Soft) EM algorithm

In this section we will write the update equations of the Soft-EM algorithm. In the first step we calculate the posterior distribution of the latent variables given the current estimates of the parameters. First the parameters are put to the initial value, i.e. iteration step $t = 0$ (NOTE: unfortunately there is two variables called t but the iteration variable is in the brackets)

$$p(z_t = 1|x_t, y_t, \phi_1^{(t)}, \phi_2^{(t)}, \sigma_1^{(t)}, \sigma_2^{(t)}, w^{(t)}) \propto w^{(t)} N(y_t|x_t^T \phi_1^{(t)}, \sigma_1^{(t)}) \tag{12}$$

$$p(z_t = 0|x_t, y_t, \phi_1^{(t)}, \phi_2^{(t)}, \sigma_1^{(t)}, \sigma_2^{(t)}, w^{(t)}) \propto (1 - w^{(t)}) N(y_t|x_t^T \phi_2^{(t)}, \sigma_2^{(t)}) \tag{13}$$

we normalise this equation and get

$$\gamma(z_t = 1) = \gamma(z_t) \propto \frac{w^{(t)} N(y_t|x_t^T \phi_1^{(t)}, \sigma_1^{(t)})}{w^{(t)} N(y_t|x_t^T \phi_1^{(t)}, \sigma_1^{(t)}) + (1 - w^{(t)}) N(y_t|x_t^T \phi_2^{(t)}, \sigma_2^{(t)})} \tag{14}$$

Then to finalise the Soft-EM expectation step we evaluate the expectation of complete data log likelihood over the posterior of latent variables z, for simplicity we mark that function $Q(\theta, \theta_t)$, where θ includes all the parameters in our model and θ_t is the current estimate of those values.

$$\begin{aligned}
Q(\theta, \theta_t) &= E_{z|x,y,\theta_t}[log(p(X, Y, Z, \phi_1, \phi_2, \sigma_1, \sigma_2, w))] \\
&= E_{z|x,y,\theta_t}[log(p(X, Y, Z|\phi_1, \phi_2, \sigma_1, \sigma_2, w))] + log(p(\phi_1, \phi_2, \sigma_1, \sigma_2, w)) \\
&= \sum_z \gamma(Z) * log(p(X, Y, Z|\phi_1, \phi_2, \sigma_1, \sigma_2, w))] + log(p(\phi_1, \phi_2, \sigma_1, \sigma_2, w)) \qquad (15)
\end{aligned}$$

Our aim in maximazation step is to find θ that solves the maximization problem $\text{argmax}_\theta\, Q(\theta, \theta_0)$ thus we do not have to care about the constant values (that do not depend on θ) and we can write (for sake of simplicity we leave the iteration step away for following equations).

$$\begin{aligned}
Q(\theta, \theta_t) &= \sum_{t=1}^{T} \gamma(z_t)[-0.5 * log(2\pi\sigma_1^2) - \frac{(x_t^T \phi_1 - y_t)^2}{2\sigma_1^2} + log(w)] \\
&+ (1 - \gamma(z_t))[-0.5 * log(2\pi\sigma_2^2) - \frac{(x_t^T \phi_2 - y_t)^2}{2\sigma_2^2} + log(1 - w)] \\
&- 0.5K * log(2\pi\sigma_1^2) - 0.5log(|\Sigma_\phi|) - \frac{1}{2\sigma_1^2}(\phi_1 - \mu_{\phi_1})^T \Sigma_\phi^{-1}(\phi_1 - \mu_\phi) \\
&+ (\alpha_{\sigma^2}) * log(\beta_{\sigma^2}) - log(\Gamma(\alpha_{\sigma^2})) + (-\alpha_{\sigma^2} - 1) * log(\sigma_1^2) - \frac{\beta_{\sigma^2}}{\sigma_1^2} \\
&- 0.5K * log(2\pi\sigma_2^2) - 0.5log(|\Sigma_\phi|) - \frac{1}{2\sigma_2^2}(\phi_2 - \mu_\phi)^T \Sigma_\phi^{-1}(\phi_2 - \mu_\phi) \\
&+ (\alpha_{\sigma^2}) * log(\beta_{\sigma^2}) - log(\Gamma(\alpha_{\sigma^2})) + (-\alpha_{\sigma^2} - 1) * log(\sigma_2^2) - \frac{\beta_{\sigma^2}}{\sigma_2^2} \\
&+ log(\Gamma(\alpha_w + \beta_w)) - log(\Gamma(\alpha_w)) - log(\Gamma(\beta_w)) \\
&+ (\alpha_w - 1) * log(w) + (\beta_w - 1) * log(1 - w) \qquad (16)
\end{aligned}$$

Then the update equations can be derived by finding the zero point of partial derivatives of Q. The update formulas are given as follows. Left side of the equations is the updated value at iteration $t + 1$

and in the right side latest values of the other parameters needed for update are used. The natural order for updates is to first update w then ϕ(s) and last σ(s). First w is derived

$$\begin{aligned}
\frac{\partial Q(\theta,\theta_t)}{\partial w} &= \sum_{t=1}^{T}\frac{\gamma(z_t)}{w} - \sum_{t=1}^{T}\frac{1-\gamma(z_t)}{1-w} + \frac{\alpha_w - 1}{w} - \frac{\beta_w - 1}{1-w} = 0 \iff \\
\frac{\alpha_w - 1 + \sum_{t=1}^{T}\gamma(z_t)}{w} &= \frac{\beta_w - 1 + T - \sum_{t=1}^{T}\gamma(z_t)}{1-w} \iff \\
w &= \frac{\alpha_w - 1 + \sum_{t=1}^{T}\gamma(z_t)}{\beta_w + \alpha_w + T - 2} \qquad (17)
\end{aligned}$$

Then we use the fact that for scalar value $x_t^T\phi = \phi^T x_t$ to derive update for ϕ_1.

$$\begin{aligned}
\frac{\partial Q(\theta,\theta_t)}{\partial \phi_1} &= -\frac{\sum_{t=1}^{T}\gamma(z_t) * 2 * (\phi_1^T x_t - y_t)x_t^T}{2\sigma_1^2} - \frac{2 * (\phi_1 - \mu_\phi)^T * \Sigma_\phi^{-1}}{2\sigma_1^2} = 0 \iff \\
\phi_1^T \sum_{t=1}^{T}\gamma(z_t)x_t x_t^T + \phi_1^T\Sigma_\phi^{-1} &= \mu_\phi^T\Sigma_\phi^{-1} + \sum_{t=1}^{T}\gamma(z_t)y_t x_t^T \iff \\
\phi_1^T &= \Big(\mu_\phi^T\Sigma_\phi^{-1} + \sum_{t=1}^{T}\gamma(z_t)y_t x_t^T\Big)\Big(\sum_{t=1}^{T}\gamma(z_t)x_t x_t^T + \Sigma_\phi^{-1}\Big)^{-1} \qquad (18)
\end{aligned}$$

following update formula of σ_1^2.

$$\begin{aligned}
\frac{\partial Q(\theta,\theta_t)}{\partial \sigma_1^2} &= \frac{-\sum_{t=1}^{T}\gamma(z_t)}{\sigma_1^2} + \frac{\sum_{t=1}^{T}\gamma(z_t)(x_t^T\phi_1 - y_t)^2)}{2 * \sigma_1^4} \\
&- \frac{0.5K}{\sigma_1^2} + \frac{(\phi_1 - \mu_\phi)^T\Sigma_\phi^{-1}(\phi_1 - \mu_\phi)}{2 * \sigma_1^4} - \frac{\alpha_{\sigma^2} + 1}{\sigma_1^2} + \frac{\beta_{\sigma^2}}{\sigma_1^4} = 0 \iff \\
\sigma_1^2 &= \frac{2 * \beta_{\sigma^2} + (\phi_1 - \mu_\phi)^T\Sigma_\phi^{-1}(\phi_1 - \mu_\phi) + \sum_{t=1}^{T}\gamma(z_t)(x_t^T\phi_1 - y_t)^2}{K + 2 * (\alpha_{\sigma^2} + 1) + \sum_{t=1}^{T}\gamma(z_t)} \qquad (19)
\end{aligned}$$

. In the similar manner we can derive update formulas for the second distribution below

$$\begin{aligned}
\frac{\partial Q(\theta,\theta_t)}{\partial \phi_2} &= -\frac{\sum_{t=1}^{T}(1-\gamma(z_t)) * 2 * (\phi_2^T x_t - y_t)x_t^T}{2\sigma_2^2} - \frac{2 * (\phi_2 - \mu_\phi)^T\Sigma_\phi^{-1}}{2\sigma_2^2} = 0 \iff \\
\phi_2^T \sum_{t=1}^{T}(1-\gamma(z_t))x_t x_t^T + \phi_2^T\Sigma_\phi^{-1} &= \mu_\phi^T\Sigma_\phi^{-1} + \sum_{t=1}^{T}(1-\gamma(z_t))y_t x_t^T \iff \\
\phi_2^T &= \Big(\mu_\phi^T\Sigma_\phi^{-1} + \sum_{t=1}^{T}(1-\gamma(z_t))y_t x_t^T\Big)\Big(\sum_{t=1}^{T}(1-\gamma(z_t))x_t x_t^T + \Sigma_\phi^{-1}\Big)^{-1} \qquad (20)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial Q(\theta,\theta_t)}{\partial \sigma_2^2} &= \frac{-\sum_{t=1}^{T}(1-\gamma(z_t))}{\sigma_2^2} + \frac{\sum_{t=1}^{T}(1-\gamma(z_t))(x_t^T\phi_2 - y_t)^2)}{2 * \sigma_2^4} \\
&- \frac{0.5K}{\sigma_2^2} + \frac{(\phi_2 - \mu_{\phi_2})^T\Sigma_\phi^{-1}(\phi_2 - \mu_{\phi_2})}{2 * \sigma_2^4} - \frac{\alpha_{\sigma^2} + 1}{\sigma_2^2} + \frac{\beta_{\sigma^2}}{\sigma_2^4} = 0 \iff \\
\sigma_2^2 &= \frac{2 * \beta_{\sigma^2} + (\phi_2 - \mu_{\phi_1})^T\Sigma_{\phi_2}^{-1}(\phi_2 - \mu_{\phi_1}) + \sum_{t=1}^{T}(1-\gamma(z_t))(x_t^T\phi_2 - y_t)^2}{K + 2 * (\alpha_{\sigma^2} + 1) + \sum_{t=1}^{T}(1-\gamma(z_t))} \qquad (21)
\end{aligned}$$

2 Model Implementation

The model is implemented with Python language and use 5 example files provided with the project description (em_algo.py, em_algo_lm.py, em_algo_mm.py, generator.py, main.py). The initial values of hyper-parameters are given

$$\alpha_w = 3, \quad \beta_w = 3, \quad \lambda_\phi = 1, \quad \mu_\phi = 0, \quad \alpha_{\sigma^2} = 5, \quad \beta_{\sigma^2} = 1 \tag{22}$$

In the following subsections we briefly describe important part of the implementation i.e. model initialisation, posterior function and update equation implementations.

2.1 Model Initialisation

The model initialisation was done mainly in the example code which gave the hyper parameters *hyperp=get_ hyperp()* and generated random normally distributed data *X=generate_ X(ndata, pdata)* and with selected model *Y, Z = generate_ YZ(X, true_ model).* Thus for initialisation only needed addition was to create data from the distributions given by the hyperparameters:

```
# initial parameter estimates drawn from prior
self.p            = dict()
self.p["sigma2_1"] = 1.0 / gamma(self.alpha_s20 , (1.0 / self.beta_s20) )  # inverse gamma
self.p["phi_1"]    = mvnormal(self.mu_phi0 , self.p["sigma2_1"] * self.sigma_phi0)
self.p["sigma2_2"] = 1.0 / gamma(self.alpha_s20 , (1.0 / self.beta_s20))  # inverse gamma
self.p["phi_2"]    = mvnormal(self.mu_phi0, self.p["sigma2_2"] * self.sigma_phi0)
self.p["w"]        = beta(self.alpha_w, self.beta_w)  # beta distribution
```

2.2 Log posterior likelihood function

For log posterior likelihood function we thought that since the soft EM algorithm was used instead of using the log likelihood defined with z we used the formula without latent variable z, which have the sum of logarithms.

```
ll = zeros(14)
phie_1 = self.p["phi_1"] - self.mu_phi0
phie_2 = self.p["phi_2"] - self.mu_phi0
w = self.p["w"]
normpdf = norm.pdf

ll[0] = sum([logaddexp(log(w) - 0.5*log(2*pi*self.p["sigma2_1"]) - 0.5 * (1/self.p["sigma2_1"]) * (xt_phi1
in zip(self.X.dot(self.p["phi_1"]),self.X.dot(self.p["phi_1"]),self.Y)])
# p(phi_1)
ll[1] = - 0.5 * log(2 * pi * self.lbd_phi0 * self.p["sigma2_1"]) * self.pdata
ll[2] = - 0.5 * phie_1.T.dot(phie_1) / (self.lbd_phi0 * self.p["sigma2_1"])
# p(phi_2)
ll[3] = - 0.5 * log(2 * pi * self.lbd_phi0 * self.p["sigma2_2"]) * self.pdata
ll[4] = - 0.5 * phie_2.T.dot(phie_2) / (self.lbd_phi0 * self.p["sigma2_2"])
# Common code for p(sigma2_1) and p(sigma2_2)
const1 = self.alpha_s20 * log(self.beta_s20)
const2 = - gammaln(self.alpha_s20)
# p(sigma2_1)
ll[5] = const1
ll[6] = const2
ll[7] = - (self.alpha_s20 + 1.0) * log(self.p["sigma2_1"])
ll[8] = - self.beta_s20 / self.p["sigma2_1"]
# p(sigma2_2)
ll[9] = const1
ll[10] = const2
ll[11] = - (self.alpha_s20 + 1.0) * log(self.p["sigma2_2"])
ll[12] = - self.beta_s20 / self.p["sigma2_2"]
return sum(ll), ll
```

2.3 Update equations

The update equations are highlighted below

```
#Calculate expectation of latent variable gamma
        mean_Z1=self.X.dot(self.p["phi_1"])
        mean_Z2=self.X.dot(self.p["phi_2"])
        pdf_t_z1 = norm(mean_Z1,sqrt(self.p["sigma2_1"] * ones(self.Y.shape))).pdf(self.Y)
        pdf_t_z0 = norm(mean_Z2,sqrt(self.p["sigma2_2"] * ones(self.Y.shape))).pdf(self.Y)
        gamma_Z=(self.p["w"]*pdf_t_z1)/(self.p["w"]*pdf_t_z1+(1-self.p["w"])*pdf_t_z0)

#update of phi-1
        myTSigma    = self.mu_phi0.T.dot(self.sigma_phi0_inv)
        SumgammayxT_S = sum(self.X.T*self.Y * gamma_Z ,axis=1)+ myTSigma
        SumgammaxxT_S     = (gamma_Z*self.X.T).dot(self.X) +self.sigma_phi0_inv
        self.p["phi_1"] = SumgammayxT_S.dot(inv(SumgammaxxT_S ))
        self.assert_logl_increased("phi_1 update")
#update of phi-2
        SumgammayxT_S2 = sum(self.X.T*self.Y * (1-gamma_Z) ,axis=1)+ myTSigma
        SumgammaxxT_S2     = ((1-gamma_Z)*self.X.T).dot(self.X) +self.sigma_phi0_inv
        self.p["phi_2"] = SumgammayxT_S2.dot(inv(SumgammaxxT_S2 ))
        self.assert_logl_increased("phi_2 update")

#update of sigma2_1
        phiSphi=((self.mu_phi0-self.p["phi_1"]).T.dot(self.sigma_phi0_inv)).dot((self.mu_phi0-self.p["phi_1"]))
        err22= (self.X.dot(self.p["phi_1"])-self.Y)**2
        sum_err = sum(err22*gamma_Z)
        num=phiSphi+2*self.beta_s20+sum_err
        den=self.pdata+2*(self.alpha_s20+1)+sum(gamma_Z)
        self.p["sigma2_1"]=num/den
        if self.p["sigma2_1"] < 0.0 :
        raise ValueError("sigma2_1 < 0.0")
        self.assert_logl_increased("sigma2_1 update")
#update of sigma2_2
        phiSphi2=((self.mu_phi0-self.p["phi_2"]).T.dot(self.sigma_phi0_inv)).dot((self.mu_phi0-self.p["phi_2"]))
        err222= (self.X.dot(self.p["phi_2"])-self.Y)**2
        sum_err2 = sum(err222*(1-gamma_Z))
        num2=phiSphi2+2*self.beta_s20+sum_err2
        den2=self.pdata+2*(self.alpha_s20+1)+sum((1-gamma_Z))
        self.p["sigma2_2"]=num2/den2
        if self.p["sigma2_2"] < 0.0 :
        raise ValueError("sigma2_2 < 0.0")
        self.assert_logl_increased("sigma2_2 update")
        self.p["w"]=(self.alpha_w-1+sum(gamma_Z))/(self.alpha_w+self.beta_w-2+self.ndata)
        self.assert_logl_increased("w update")
```

3 Model Testing

In this section the model is tested in different validation scenarios. The dimensionality and the number of training data is varied while the number of validation data is kept fixed (50). Mean Squared Error (MSE) is calculated for one randomly selected initialisation point of the parameters and compared to the model having best MSE from 100 different random initialisation point. Following test cases were selected Case 1 to 9: (#dimensions, #of data), (2,10), (2,50), (2,100), (5,10), (5,50), (5,100), (10,10) ,(10,50), (10,100). As can be seen in the table below the results improve when starting from the multiple different initial locations and selecting the best one. This is due to the fact that EM algorithm find the locally optimised parameters and starting from different initial point find mind different local parameters. This effect increase when the dimensionality increase.

Table 1: Mean Square Error of selected cases. O=One run, B=Best from 100 run

# of data points → Dimensionality ↓	O:10	B:10	O:50	B:50	O:100	B:100
2	0.203	0.182	0.191	0.177	0.175	0.100
5	1.323	0.658	0.184	0.159	0.172	0.168
10	5.267	1.839	1.649	0.267	0.698	0.292

For the completeness we did another run and compared the MSE results from the 100 run best model to the MSE from the true model. From that we can see more clearly that when the number of training samples increase the trained model has similar MSE as the true model (on validation set). This effect of increased data points were not as clear in the above table.

Table 2: Comparing Mean Square Error of True Model (variance included to the data) to the MSE of Trained Model

# of data points → Dimensionality ↓	Trained 10	True 10	Trained 50	True 50	Trained 100	True 100
2	0.323	0.237	0.529	0.479	0.232	0.234
5	0.687	0.317	0.311	0.198	0.349	0.354
10	1.346	0.320	0.336	0.156	0.470	0.315

4 Model comparison

In this section the **Mixture model with two linear components** is compared to the **simple Linear model**. Comparison is analysed in Cases 1-9 (as described in the previous section) and with similarity study cases 1,3,7,9. The validation set size is 50 as before. Furthermore, data is generated for both models from 10 different random initialisation points and Model with the best MSE is chosen. The data is derived from simple linear model or from different mixture models, as stated in following subsections.

4.1 Data from simple Linear model

When data is generated from simple Linear model both simple Linear Model and Mixture Model seem to have similar prediction possibilities with this range of X. It also seems that even with 10 samples the models are reasonably good and adding data points do not make significant improvements to the result.

Table 3: Trained model cross-validation MSE and corresponding Log likelihood, of LM=Linear Model and MM=Mixture Model, when data generated from another Linear Model.

# of data points → Dimensionality ↓	LM 10	MM 10	LM 50	MM 50	LM 100	MM 100
2	0.256 -39.712	0.273 -43.996	0.255 -36.490	0.278 -39.593	0.200 -31.218	0.181 -28.546
5	0.151 -27.278	0.172 -28.538	0.115 -18.939	0.101 -14.108	0.139 -23.357	0.137 -25.603
10	0.211 -43.379	0.209 -39.788	0.374 -57.912	0.383 -58.502	0.224 -45.707	0.231 -46.717

4.2 Data from one Mixture model

When data is generated from two linear component mixture model the possibilities of the linear model to explain the data varies depending on the true model parameters. Increase of the training data shows the superiority of Mixture Model to explain the data. Small amount of the training data is not sufficient especially for high dimensional data.

Table 4: Trained model cross Validation MSE of LM=Linear Model and MM=Mixture Model when data generated from another Mixture Model.

# of data points → Dimensionality ↓	LM 10	MM 10	LM 50	MM 50	LM 100	MM 100
2	0.348	0.249	1.366	0.424	0.369	0.165
5	0.559	0.570	0.585	0.245	0.776	0.273
10	1.100	1.017	4.840	0.343	0.705	0.248

4.3 Data from different Mixture models

We draw data from 100 different mixture model and took the cosine similarity between linear components for every mixture model. We calculated the MSE for both trained LM and MM. The difference of these MSE was normalised with MM MSE. The results were plotted below with least square linear trend. We expected that the difference would be much higher when the similarity is smaller. The w would also play an important role, when w is very small or very big the difference between LM and MM MSE would be small, since most of the data could be explained with one linear model.

Figure 1 was bit different than we expected, but due to the time limit we did not try to find all the reasons for that. It seems that in the high dimension with a lot of training data even bigger cosine similarity the MM is clearly better, with smaller data the trend in this case is as we expected. We could have studied different X values as well to get the difference more clearer to the trend.

5 Summary

With given initial parameters the randomness of the results vary quite a lot between the runs. Averaging the initial starting point makes the mixture model clearly better. This improves the EM algorithm that could find only local optimum, and mixture model with more parameters have more local optimums. Even when the data is drawn from linear model, mixture model can explain the data well (prediction clearly outside the range of current could be different issue). This means that mixture model is better starting point when trying to explain the data.

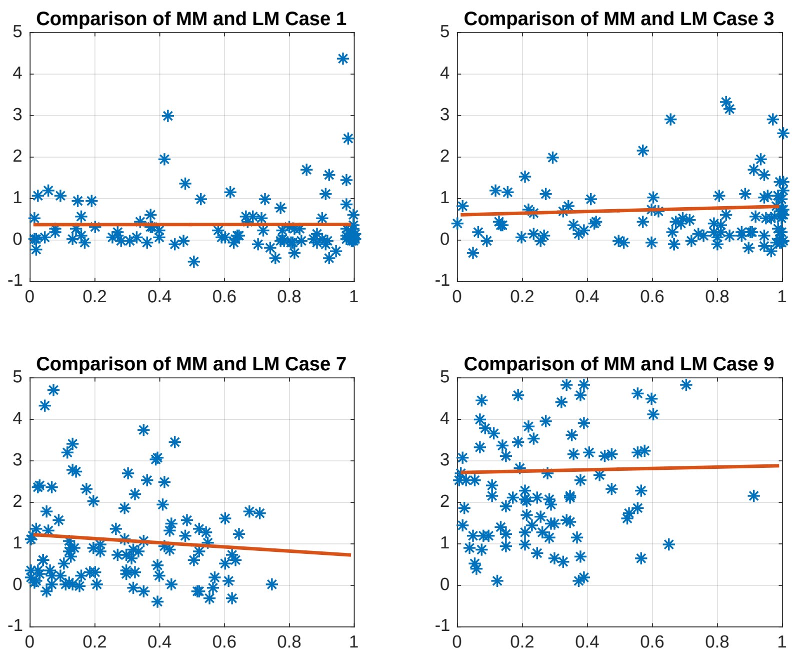

Figure 1: The normalised difference of MSE (on y axis) between trained MM and LM as a function of cosine similarity (on x axis) of the data generating mixture model