6.434J/16.391J
Statistics for Engineers and Scientists May 4
MIT, Spring 2006 Handout #17

Solution 7

Problem 1: Generating Random Variables

Each part of this problem requires implementation in MATLAB. For the results, you should submit your code, explanation of the parameters selected and correctly labeled results where needed.

(a) Given a probability mass function (pmf) of a discrete random variable, write an algorithm to generate N samples from the given pmf. Test your algorithm for some arbitrary pmf and observe the histogram of samples drawn by your algorithm. You may need a large N for smoother histogram.

(b) Implement the Box-Muller algorithm that was discussed in class to generate Gaussian random variable. Are there any numerical stability issues? ($\log 0$?) Can you modify the algorithm to avoid them? Watch your algorithm for its computation time — you may compare this with MATLAB's *randn*. On a computer, operations like *sin*, *cos* and *log* are expensive to implement. We can remove *sin* and *cos* very easily. Implement the version without them.

(c) Another popular method of generating random variables is via *ratio of uniforms*. In simple terms, the idea is this: if U_1 and U_2 are i.i.d., uniformly distributed between 0 and 1, and $R = U_2/U_1$. Then conditioned on the event $A = \{U_1 \leq \sqrt{f(U_2/U_1)}\}$, the random variable R has the density function f.

Implement this method to obtain samples from exponential distribution $\lambda = 2$.

Solution

(a)

```
N = 10000;
% generate a random pmf, with 10 values:
OMEGA = 0:10:90;
pmf = rand(1,10);
pmf = pmf/sum(pmf);
X = [];
for i=1:N,
    Xi = discreteRV(pmf); % this function is defined below
    X = [X Xi];
end
% X contains the indices to random variable values, corresponding to pmf
% random variable can be obtained from the indices:
Xval = OMEGA(X);

% See how good the historgram looks:
for index = 1:length(pmf),
    pmf_e(index) = sum(X==index)/N;
end
stem(pmf);
hold on;
stem(pmf_e,'r');
%-----------------------------

function x = discreteRV(pmf)
%  function x = discreteRV(pmf)
% x is the index for the discrete value in pmf.
% This function generates a sample of random variable, distributed
% according to the given pmf. The support of the discrete pmf is
% assumed to be [0,M-1] where M is the length of the pmf vector.
%

u = rand;
x = 1;
while (u > sum(pmf(1:x)))
    x = x+1;
end
```

(b) The Box-Muller algorithm is a nice way to generate Gaussians. However, in its basic form, it can get trapped into the problems of $\log 0$, and have increased computations because of sin and cos. The way to avoid these problems is to start the so-called 'hit and miss' approach that is common in Monte Carlo methods. The idea is to generate samples of two uniforms U_1 and U_2 in $[-1,+1]$, instead of $[0,1]$, and then chooses only those which belong to the unit circle or its inside. This gives us direct way of computing sin and cos by taking ratio, and also avoids the $\log 0$. Code is given below.

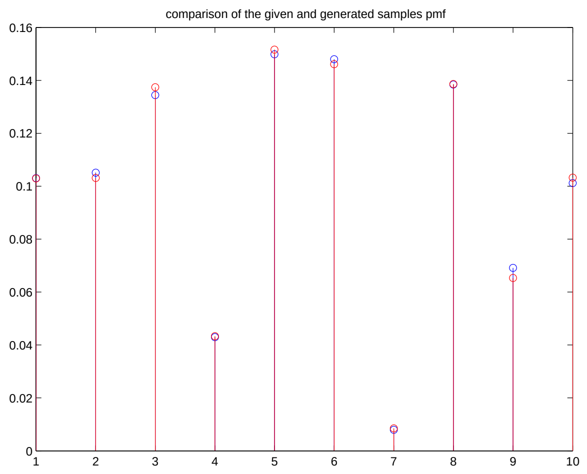

Figure 1: Plot of the desired (blue) and the obtained (red) pmf.

Computation time in MATLAB is much higher than that of *randn*. First the basic Box-Muller:

```
function X = BoxMuller1(N);
%  Box-Muller algorithm for generating Gaussian

U1 = rand(1,N); U2 = rand(1,N);
Phi = 2*pi*U1;
R = sqrt(-2*log(U2));
X = R.*cos(Phi);
```

Modified Box-Muller:

```
function X = BoxMuller2(N);
%  Box-Muller algorithm for generating Gaussian, modified for computation
X = [];
count = 0;
while (count < N)
    U1 = 2*rand(1,2*N) - 1;
    U2 = 2*rand(1,2*N) - 1;
    R = U1.^2 + U2.^2;
    indices = find(R <= 1);
    R = R(indices);
    U1 = U1(indices);
    X = [X sqrt(-2*log(R)) .* U1./sqrt(R)];
    count = count + length(indices);
end
% throw away the extra samples
X = X(1:N);
```

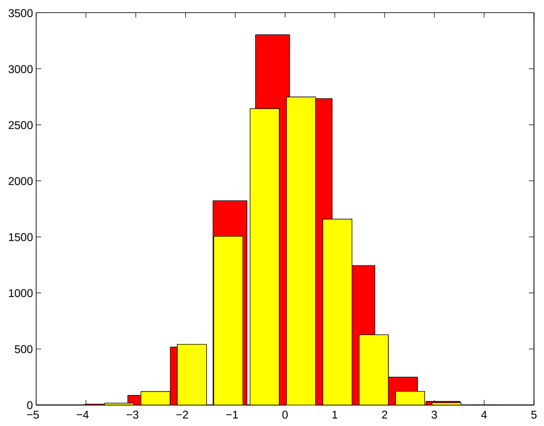

Figure 2: Plot of the histograms for Gaussian variable, N(0,1): Box-Muller Algorithm 1 (red) and 2 (yellow).

(c)

```
function X = ratioOfUniforms(N)
%
% Ratio of Uniforms method to generate desired Random Variable
%
X = [];
count = 0;
while (count < N)
    U1 = rand(1,2*N);
    U2 = rand(1,2*N);
    R = U2./U1;
    indices = find(U1 < sqrt(fx_exp(R)));
    X = [X R(indices)];
    count = count + length(indices);
```

```
end
% throw away the extra samples
X = X(1:N);

function y=fx_exp(x); lambda = 2; y = lambda*exp(-lambda*x);
```

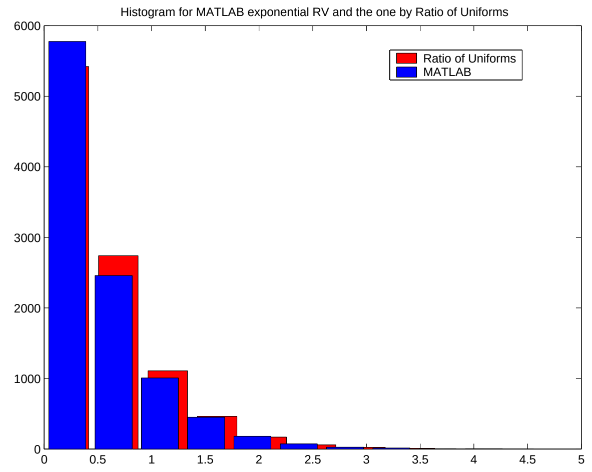

Figure 3: Sample histograms: MATLAB's exponential random variable (blue) and the one via Ratio of Uniforms (red).

Problem 2: Gibbs Sampler

Background: In Monte Carlo based solutions, a very common requirement is to sample from a desired distribution. There are various schemes that are generally available. One of them is *Gibbs Sampling*, which is reasonably simple to implement as well as efficient when the samples to be drawn are multi-dimensional. The basic idea of Gibbs sampler is to draw one component at a time, of a K-dimensional sample, from the conditional distribution of the specific component; conditioned upon the rest of the components. For example, consider a K-component vector $\mathbf{x}$, and we want to draw N_s samples of this. Gibbs sampler runs as follows:

1. Start $n = -N_b$. Initialize $\mathbf{x}^{(n)} = [x_1^{(n)}, x_2^{(n)}, ...x_K^{(n)}]$ from random samples or any reasonable values if known.

2. For $n = -N_b : N_s$,
   - draw sample $x_1^{(n)}$ from $P(x_1|x_2^{(n-1)}, ...x_K^{(n-1)})$

- draw sample $x_2^{(n)}$ from $P(x_2|x_1^{(n)}, x_3^{(n-1)}....x_K^{(n-1)})$

....

- draw sample $x_{K-1}^{(n)}$ from $P(x_K|x_1^{(n)}, x_2^{(n)}, ...x_{K-1}^{(n)})$

Samples from $n = 1$ to N_s are the desired samples. They can then be used in Monte Carlo based calculations afterwards – for example, in our case, we'll compute sample means of the last 500 samples to estimate the desired parameters. N_b is the so-called burn-in period of the Gibbs sampler, that is the time required for the sampler to settle down.

Note that, to draw the nth sample vector, $\mathbf{x}_n = [x_1^{(n)}, x_2^{(n)}, ...x_K^{(n)}]$, Gibbs sampler draws $x_1^{(n)}$ from $p(x_1|x_2^{(n-1)}, x_3^{(n-1)}, ...x_K^{(n-1)})$. Then for the next component $x_2^{(n)}$, the distribution is conditioned on the current sample component 1, $x_1^{(n)}$ and the other components from the previous sample, $x_2^{(n-1)}$, $x_3^{(n-1)}$...,$x_K^{(n-1)}$.

Problem Description: In this problem, we want to implement a Gibbs Sampler for the estimation of mean and variance of a Gaussian distribution. Assume we have L i.i.d.samples Y_1,Y_2,...,Y_L from $\mathcal{N}(\mu, v)$, with unknown mean μ and variance v. That is,

$$p(Y|\mu, v) = \frac{1}{\sqrt{2\pi v}} \exp(-\frac{1}{2v}(y-\mu)^2) \tag{1}$$

We want to use the given L samples of Y (in *hw7p2.mat*) and estimate the unknowns; (note that in some practical systems, it is very expensive to obtain a sample — because of destructive sampling, for example — so the sample size L may be very small). Implement a Gibbs Sampler to draw samples related to the μ and v respectively. The idea is that after the sampler has "settled" well (we are not going to explore the convergence of the sampler here, instead we choose to run it for 10000 iterations ($N_b + N_s = 10000$), the drawn samples will be close to the actual distributions. Take the sample averages of last 500 samples as estimates of the required parameters.

In Gibbs sampler, we need to derive the conditional densities $p(\mu|Y, v)$ and $p(v|Y, \mu)$ to draw samples from them. We can use Bayes rule for this derivation from (1), but it also needs some prior distribution of the unknown parameter. From the knowledge about the Gaussian distribution, a good selection of prior for μ is $\mathcal{N}(0, 1)$, and prior for the v is Inverse Gamma distribution. Or it is instructive to estimate $1/v$, with its prior as the Gamma distribution. That is, we estimate $v_{\text{inv}} = 1/v$ and take $v_{\text{inv}} \sim \mathcal{G}(2, 1)$. You can initialize the first samples to any reasonable guess. A simple choice may be $\mu = 0$ and $v_{\text{inv}} = 1$.

(1) can be re-written as:

$$p(Y|\mu, v_{\text{inv}}) = \sqrt{\frac{v_{\text{inv}}}{2\pi}} \exp(-\frac{v_{\text{inv}}}{2}(y-\mu)^2) \tag{2}$$

In MATLAB, $\mathcal{N}(0,1)$ distribution can be generated by *randn*, and $\mathcal{G}(2,1)$ is obtained by *gamrnd(a,1/b,...)*, where a and b are the parameters of the Gamma distribution as:

$$p_{\mathcal{G}}(x) \propto x^{a-1} \exp(-bx)$$

(See, density has been mentioned only up to an unknown constant; that constant is not important in solving this problem. The purpose in mentioning the density is to point out the difference in MATLAB's pdf and the one commonly seen (and given above): MATLAB's definition has $\exp(-x/b)$, so we call its function with $1/b$.)

Solution The joint distribution is given using the priors:

$$\begin{aligned} p(y, \mu, v_{\text{inv}}) &= \left\{ \prod_{i=1}^{L} p(y_i|\mu, v_{\text{inv}}) \right\} \pi(\mu)\pi(v_{\text{inv}}) \\ &= (2\pi)^{-(L+1)/2} v_{\text{inv}}^{L/2} \exp\left\{ -\frac{v_{\text{inv}}}{2} \sum_{i=1}^{L} (y_i - \mu)^2 \right\} \exp\left\{-\mu^2/2\right\} v_{\text{inv}} \exp\left\{-v_{\text{inv}}\right\}. \end{aligned}$$

where π is used to denote the distribution, as the Gibbs Sampler iterates a Markov chain. To find full conditional for μ we select the terms from $p(y, \mu, v_{\text{inv}})$ that contains μ and normalize.

$$\begin{aligned} \pi(\mu|v_{\text{inv}}, y) &= \frac{\pi(\mu, v_{\text{inv}}|y)}{\pi(v_{\text{inv}}|y)} \\ &= \frac{\pi(\mu, v_{\text{inv}}, y)}{\pi(v_{\text{inv}}, y)} \\ &\propto \pi(\mu, v_{\text{inv}}, y) \end{aligned}$$

So, we write the conditional distribution, dropping the constants or terms that do not involve μ:

$$\begin{aligned} \pi(\mu|v_{\text{inv}}, y) &\propto \exp\left\{ -\frac{v_{\text{inv}}}{2} \sum_{i=1}^{L} (y_i - \mu)^2 \right\} \exp\left\{-\mu^2/2\right\} \\ &\propto \exp\left\{ -\frac{1}{2}(1 + Lv_{\text{inv}}) \left(\mu - \frac{v_{\text{inv}} \sum y_i}{1 + Lv_{\text{inv}}} \right)^2 \right\} \end{aligned}$$

where we have again not cared for any factors unrelated to μ and have focused on getting some close form for μ. This is clearly a Gaussian distribution,

$\mathcal{N}\left(\frac{v_{\text{inv}}\sum y_i}{1+Lv_{\text{inv}}}, \frac{1}{1+Lv_{\text{inv}}}\right)$ and its samples can be drawn easily. Now concentrate on getting conditional distribution of v_{inv}.

$$\begin{aligned}\pi(v_{\text{inv}}|\mu, y) &\propto v_{\text{inv}}^{L/2} \exp\left\{-\frac{v_{\text{inv}}}{2}\sum_{i=1}^{L}(y_i-\mu)^2\right\} v_{\text{inv}} \exp\{-v_{\text{inv}}\} \\ &= v_{\text{inv}}^{L/2+1} \exp\left\{-v_{\text{inv}}\left[1+\frac{1}{2}\sum_{i=1}^{L}(y_i-\mu)^2\right]\right\}\end{aligned}$$

which we identify as the Gamma distribution (unnormalized form), that is: $\mathcal{G}(2+L/2, 1+\frac{1}{2}\sum_{i=1}^{L}(y_i-\mu)^2)$.

Note that the given samples of Y are used in these pdf's. The Gibbs sampler will recursively draw samples from these distributions. MATLAB code for this is shown below:

```
% The mu = mean and vinv = 1/variance are the unknown parameters,
% that Gibbs wants to estimate from only 50 samples of observations
% of Gaussian y (which in this experiment is 2 mean, 9 variance).
% We run NN iterations of Gibbs.
% The proposed density for mu is N(0,1) and tau Gamma(2,1),
% that is vinv.exp(-vinv).
% From this information, we find p(y,mu,vinv) by multiplying individual
% densities.
% Farther, marginalize to get pi(mu | vinv,y) and pi(vinv | mu,y).
% Which are obtained to be, respectively:
%    exp(-1/2 * (1+n*vinv) (mu - vinv*sum_y / (1+n*vinv) ))
%    vinv^(n/2 + 1) . exp ( -vinv [1+1/2 sum_{1_to_n} (yi-mu)^2])
%-----------------------------------%
NN = 10000;
mus = []; vinvs = [];
load hw7p2;
suma = sum(y);
mu = 0; % set the parameters as prior means
vinv = 1; %

for i = 1 : NN
    new_mu = sqrt(1/(1+n*vinv)) * randn + (vinv*suma)/(1+n*vinv);

    par = 1 + 1/2 * sum( (y - mu).^2 );
    new_vinv = gamrnd(2 + n/2, 1/par,1,1);
    mus = [mus new_mu];
    vinvs = [vinvs new_vinv];
    mu = new_mu;
    vinv = new_vinv;
end

mu_hat = mean(mus(end-499:end))
vinv_hat = mean(vinvs(end-499:end))

hist(mus)
figure;
hist(vinvs)
%-------------------------------------------------------%
```

The result of sample average from the last 500 samples of the Gibbs are: $\hat{\mu} = 1.9075, \hat{v_{\text{inv}}} = 0.0976$. which are quite close to their actual values of 2 and 1/9 respectively. Histograms below show the overall samples pattern.

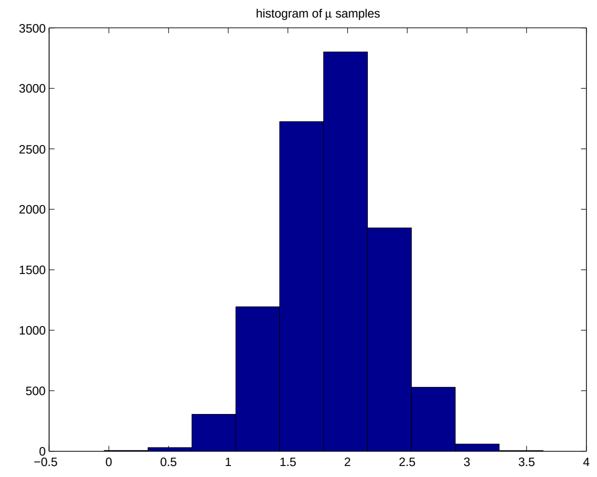

Figure 4: Histogram of samples generated for mean via Gibbs sampling.

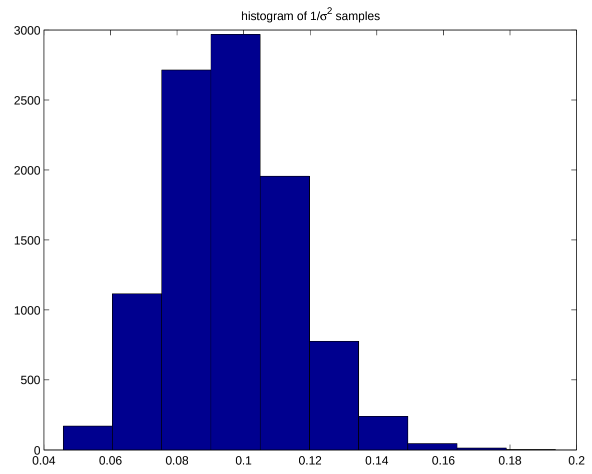

Figure 5: Histogram of samples generated for variance via Gibbs sampling.

Problem 3: [Gaussian Mixture] Let $X_1, X_2, \ldots, X_n$ be an i.i.d. sample from the mixture density,

$$p(x \mid \boldsymbol{\theta}) = \sum_{i=1}^{m} \alpha_i f(x \mid \mu_i, \sigma_i^2),$$

where $\boldsymbol{\theta} \triangleq (\alpha_1, \mu_1, \sigma_1^2,\ \alpha_2, \mu_2, \sigma_2^2,\ \ldots,\ \alpha_m, \mu_m, \sigma_m^2)$ are unknown parameters:

$$\begin{aligned} -\infty < \mu_i < \infty & \qquad \text{for } i = 1, 2, \ldots, m; \\ \sigma_i > 0 & \qquad \text{for } i = 1, 2, \ldots, m; \\ \alpha_i \geq 0 & \qquad \text{for } i = 1, 2, \ldots, m; \text{ and} \\ \sum_{i=1}^{m} \alpha_i = 1. & \end{aligned}$$

Here, function $f(\cdot \mid \mu_i, \sigma_i^2)$ is the density of a normal $N(\mu_i, \sigma_i^2)$ random variable:

$$f(x \mid \mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}}\, \mathrm{e}^{-\frac{1}{2\sigma_i^2}(x-\mu_i)^2}.$$

(a) Estimate $\boldsymbol{\theta}$ by using the EM algorithm.

(b) Interpret the M-steps.

(c) How does the EM algorithm change when we know a priori that all μ_i's and σ_i^2's are equal? Interpret your results.

(d) Assume that $m = 2$ (two-mixture model) and that unknown parameter α_1 takes a value from the set $\{0, 1\}$. How does the EM algorithm behave?

Solution

(a) Following the steps and notations of lecture on Mixture of Density, the M-step is given by:

$$\widehat{\boldsymbol{\theta}}^{(n+1)}$$
$$= \arg\max_{\boldsymbol{\theta}\in\Theta} \underbrace{\sum_{k=1}^{n}\sum_{j=1}^{m} \mathbb{P}\left\{Y_k = j \,|\, X_k = x_k, \widehat{\boldsymbol{\theta}}^{(n)}\right\} \Big[\ln f(x_k \,|\, \mu_j, \sigma^2 j] + \ln \alpha_j\Big]}_{\triangleq h(\boldsymbol{\theta})},$$

where the feasible set of parameters is

$$\begin{aligned}\Theta \triangleq \Big\{(\alpha_1, \mu_1, \sigma^2 1,\ \alpha_2, \mu_2, \sigma^2 2,\ \ldots \alpha_m, \mu_m, \sigma^2 m)\,\Big|\, &\sum_{i=1}^{m} \alpha_i = 1;\\ &\alpha_i \geq 0,\ \text{for } i = 1, 2, \ldots, m;\\ &-\infty < \mu_i < \infty,\ \text{for } i = 1, 2, \ldots, m;\\ &\sigma^2 i > 0,\ \text{for } i = 1, 2, \ldots, m\Big\}.\end{aligned}$$

Here, we have defined the variance $\sigma^2 i \triangleq \sigma_i^2$.

From the lecture, for each $l = 1, 2, \ldots, m$, the next estimate of the weight α_l is given by

$$\widehat{\alpha}_l^{(n+1)} = \frac{\sum_{k=1}^{n} \mathbb{P}\left\{Y_k = l \,|\, X_k = x_k, \widehat{\boldsymbol{\theta}}^{(n)}\right\}}{n}, \tag{3}$$

where the probability in the numerator is

$$\mathbb{P}\left\{Y_k = l \,|\, X_k = x_k, \widehat{\boldsymbol{\theta}}^{(n)}\right\} = \frac{f(x_k \,|\, \widehat{\mu}_l^{(n)}, \widehat{\nu}_l^{(n)}) \cdot \widehat{\alpha}_l^{(n)}}{\sum_{i=1}^{m} f(x_k \,|\, \widehat{\mu}_i^{(n)}, \widehat{\nu}_i^{(n)}) \cdot \widehat{\alpha}_i^{(n)}}.$$

Next, we obtain $\widehat{\mu}_l^{(n+1)}$ and $\widehat{\nu}_l^{(n+1)}$ simultaneously for each $l = 1, 2, \ldots, m$. Setting the partial derivative of $h(\boldsymbol{\theta})$ with respect to μ_l to zero and setting the partial derivative of $h(\boldsymbol{\theta})$ with respect to $\sigma^2 l$ to zero yield two equations,

$$\begin{aligned}
0 &= \frac{\partial}{\partial \mu_l} h(\boldsymbol{\theta}) \\
&= \sum_{k=1}^{n} \mathbb{P}\left\{ Y_k = l \,|\, X_k = x_k, \widehat{\boldsymbol{\theta}}^{(n)} \right\} (\mu_l - x_k), \\
0 &= \frac{\partial}{\partial \sigma^2 l} h(\boldsymbol{\theta}) \\
&= \sum_{k=1}^{n} \mathbb{P}\left\{ Y_k = l \,|\, X_k = x_k, \widehat{\boldsymbol{\theta}}^{(n)} \right\} \left(1 - \frac{(\mu_l - x_k)^2}{\sigma^2 l}\right)
\end{aligned}$$

with two unknowns, μ_l and $\sigma^2 l$. Solving the above equations for μ_l and $\sigma^2 l$ yields the maximizers,

$$\begin{aligned}
\widehat{\mu}_l^{(n+1)} &= \frac{\sum_{k=1}^{n} \mathbb{P}\left\{ Y_k = l \,|\, X_k = x_k, \widehat{\boldsymbol{\theta}}^{(n)} \right\} \cdot x_k}{n \widehat{\alpha}_l^{(n+1)}} \\
\widehat{\nu}_l^{(n+1)} &= \frac{\sum_{k=1}^{n} \mathbb{P}\left\{ Y_k = l \,|\, X_k = x_k, \widehat{\boldsymbol{\theta}}^{(n)} \right\} \cdot (x_k - \widehat{\mu}_l^{(n+1)})^2}{n \widehat{\alpha}_l^{(n+1)}}.
\end{aligned}$$

(b) Expressions for $\widehat{\alpha}_l^{(n+1)}$ and $\widehat{\mu}_l^{(n+1)}$ are identical to those in the lecture. Hence, the interpretation for them are as given in the lecture note.

An interpretation for $\widehat{\nu}_l^{(n+1)}$ is as follows. The numerator is the weighted sum of the squared errors. The denominator is the expected number of samples from the l^{th} machine. Hence, the ratio is the approximated variance of the data that are generated by the l^{th} machine.

(c) When $\mu_i = \mu$ and $\sigma_i^2 = \sigma$, for all $i = 1, 2, \ldots, m$, we have one density as opposed to a mixture of m distinct densities:

$$\begin{aligned}
p(x \,|\, \boldsymbol{\theta}) &\triangleq \sum_{i=1}^{m} \alpha_i f(x \,|\, \mu, \sigma^2) \\
&= f(x \,|\, \mu, \sigma^2).
\end{aligned}$$

Thus, the problem reduces to finding the maximum likelihood estimator of unknown parameters, μ and σ, from samples of normal $N(\mu, \sigma^2)$ density. We can solve this problem by using maximum likelihood estimation.

(d) If the initial estimates is $\widehat{\alpha}_1^{(0)} = 1$ and $\widehat{\alpha}_2^{(0)} = 0$, the expression for $\widehat{\alpha}_l^{(n+1)}$

in equation (3) implies that

$$\begin{aligned} 1 &= \widehat{\alpha}_1^{(1)} = \widehat{\alpha}_1^{(2)} = \widehat{\alpha}_1^{(3)} = \dots \\ 0 &= \widehat{\alpha}_2^{(1)} = \widehat{\alpha}_2^{(2)} = \widehat{\alpha}_2^{(3)} = \dots . \end{aligned}$$

Similarly, if $\widehat{\alpha}_1^{(0)} = 0$ and $\widehat{\alpha}_2^{(0)} = 1$, we will have

$$\begin{aligned} 0 &= \widehat{\alpha}_1^{(1)} = \widehat{\alpha}_1^{(2)} = \widehat{\alpha}_1^{(3)} = \dots \\ 1 &= \widehat{\alpha}_2^{(1)} = \widehat{\alpha}_2^{(2)} = \widehat{\alpha}_2^{(3)} = \dots . \end{aligned}$$

In conclusion, the sequence of estimates, $\widehat{\alpha}_l^{(n)}$, for $n = 1, 2, 3, \dots,$ and $l \in \{1, 2\}$, will be equal to the initial estimate. If the initial estimate of (α_1, α_2) is not equal to the ML estimate, the EM algorithm will never converge to the ML estimate.