MAST30027: Modern Applied Statistics

Assignment 4, Solution, 2021

Problem 1: Posterior inference using Gibbs sampling

(a) **Solution** Let $n = 100$ and $m = 150$, we have

$$
\begin{aligned}
& p(\mu_1|\mu_2, x_1, \cdots, x_n, y_1, \cdots, y_m) \\
\propto\, & p(x_1, \cdots, x_n|\mu_1)p(y_1, \cdots, y_m|\mu_2)p(\mu_1, \mu_2) \\
\propto\, & \exp\left[-\frac{1}{2}\left(\sum_{i=1}^{n}(x_i - \mu_1)^2 + 2\sum_{i=1}^{m}(y_i - \mu_2)^2 + (3\mu_1^2 + 4\mu_1\mu_2 + 3\mu_2^2)\right)\right] \\
\propto\, & \exp\left[-\frac{1}{2}\left((n+3)\mu_1^2 - 2\left(\sum_{i=1}^{n} x_i - 2\mu_2\right)\mu_1\right)\right] \\
\propto\, & \exp\left[-\frac{n+3}{2}\left(\mu_1^2 - 2\frac{\sum_{i=1}^{n} x_i - 2\mu_2}{n+3}\mu_1\right)\right],
\end{aligned}
$$

Thus, we have $\mu_1|\mu_2, x_1, \cdots, x_n, y_1, \cdots, y_m \sim \text{Normal}(\frac{\sum_{i=1}^{n} x_i - 2\mu_2}{n+3}, \frac{1}{n+3})$.

Similarly, we have $\mu_2|\mu_1, x_1, \cdots, x_n, y_1, \cdots, y_m \sim \text{Normal}(\frac{2\sum_{i=1}^{m} y_i - 2\mu_1}{2m+3}, \frac{1}{2m+3})$.

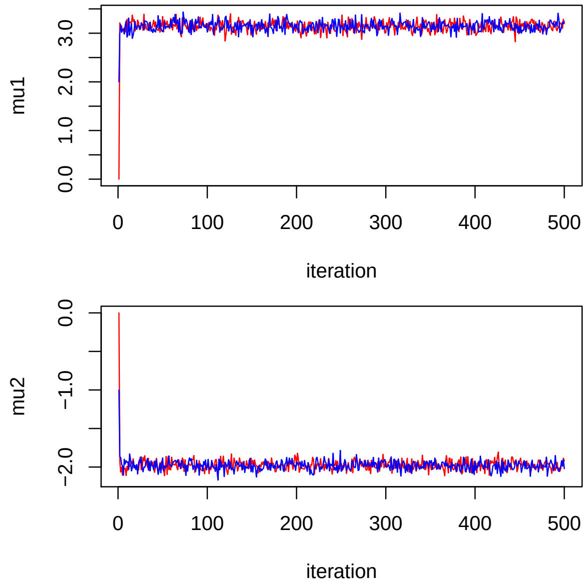
(b) **Solution**

```
> x = scan(file="assignment4_x_2021.txt", what=double())
> y = scan(file="assignment4_y_2021.txt", what=double())
> set.seed(30027)
> # Implement Gibbs Sampler
> GibbsS <- function(mu1.0, mu2.0, nreps, x, y){
+
+   Gsamples <- matrix(nrow=nreps, ncol=2)
+   Gsamples[1,] <- c(mu1.0, mu2.0)
+
+   # main loop
+   n = length(x)
+   m = length(y)
+   for (i in 2:nreps) {
+     mu1 = Gsamples[i-1,1]
+     mu2 = Gsamples[i-1,2]
+     mu1 = rnorm(1, (sum(x)-2*mu2)/(n+3), sqrt(1/(n+3)))
+     mu2 = rnorm(1, (2*sum(y)-2*mu1)/(2*m+3), sqrt(1/(2*m+3)))
+     Gsamples[i,] <- c(mu1, mu2)
+   }
+   return(Gsamples=Gsamples)
+ }
> # number of iterations
> nreps <- 500
> # run two Gibbs sampling chains
> GibbsS1 = GibbsS(mu1.0=0, mu2.0=0, nreps, x, y)
> GibbsS2 = GibbsS(mu1.0=2, mu2.0=-1, nreps, x, y)
```

Make a trace plot for each of parameters.

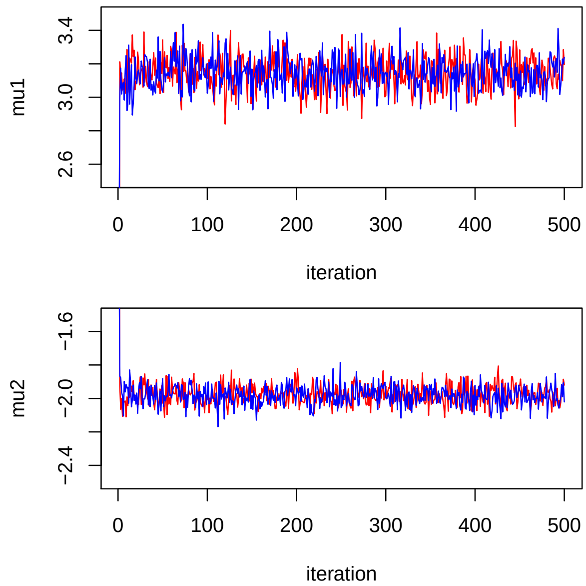
```
> par(mfrow=c(2,1), mar=c(4,4,1,1))
> plot(1:nreps, GibbsS1[,1], type="l", col="red",
+      ylim = c(min(GibbsS1[,1],GibbsS2[,1]), max(GibbsS1[,1],GibbsS2[,1])),
```

```
+       xlab = "iteration", ylab ="mu1")
> points(1:nreps, GibbsS2[,1], type="l", col="blue")
> plot(1:nreps, GibbsS1[,2], type="l", col="red",
+       ylim = c(min(GibbsS1[,2],GibbsS2[,2]), max(GibbsS1[,2],GibbsS2[,2])),
+       xlab = "iteration", ylab ="mu2")
> points(1:nreps, GibbsS2[,2], type="l", col="blue")
```



Let's zoom in on the trace plot.

```
> par(mfrow=c(2,1), mar=c(4,4,1,1))
> plot(1:nreps, GibbsS1[,1], type="l", col="red", ylim = c(2.5,3.5),
+       xlab = "iteration", ylab ="mu1")
> points(1:nreps, GibbsS2[,1], type="l", col="blue")
> plot(1:nreps, GibbsS1[,2], type="l", col="red", ylim = c(-2.5, -1.5),
+       xlab = "iteration",ylab ="mu2")
> points(1:nreps, GibbsS2[,2], type="l", col="blue")
```

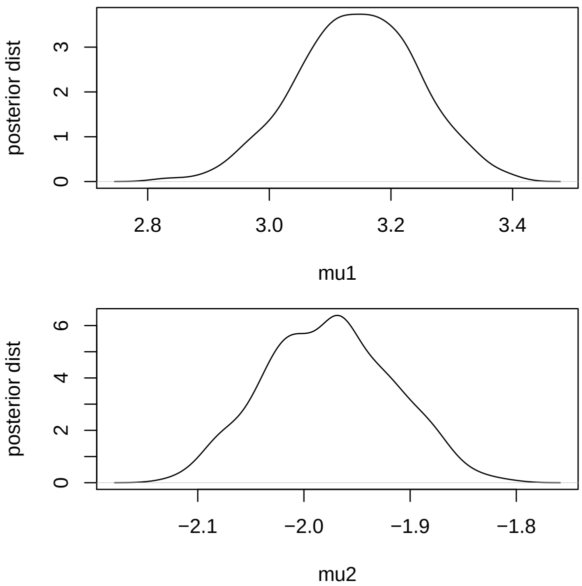


The trace plots show that samples from different chains are mixed well and behave similarly.

(c) **Solution** We will remove the first 50 samples as burn-in period.

1) make a plot that shows empirical (estimated) marginal posterior distribution.

```
> par(mfrow=c(2,1), mar=c(4,4,1,1))
> plot(density(GibbsS1[-(1:50),1]), ylab="posterior dist", xlab="mu1", main="")
> plot(density(GibbsS1[-(1:50),2]), ylab="posterior dist", xlab="mu2", main="")
```



2) estimate marginal posterior mean.

```
> # for mu1
> mean(GibbsS1[-(1:50),1])

[1] 3.144036

> # for mu2
> mean(GibbsS1[-(1:50),2])

[1] -1.976291
```

3) report a 90% credible interval for the marginal posterior distribution.

```
> quantile(GibbsS1[-(1:50),1], probs=c(0.05, 0.95))

      5%      95%
2.974518 3.306848

> # for mu2
> quantile(GibbsS1[-(1:50),2], probs=c(0.05, 0.95))

       5%       95%
-2.076106 -1.876740
```

Problem 2: Posterior inference using the Metropolis-Hastings (MH) algorithm

(a) **Solution** The posterior distribution is proportional to the product of the likelihood and prior distribution. Thus, we will use the product of the likelihood and prior distribution as a target distribution $\pi(\mu_1, \mu_2)$ in the Metropolis-Hastings algorithm.

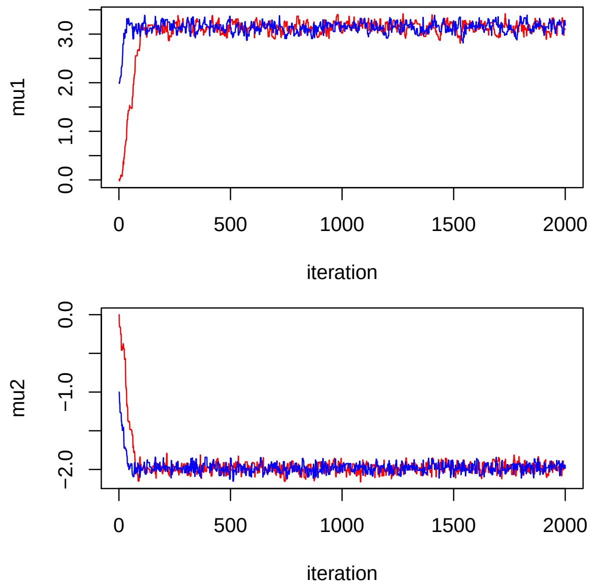
```
> # log likelihood
> likelihood <- function(param, x, y){
+     mu1 = param[1]
+     mu2 = param[2]
```

```
+
+     singlelikelihoodsX = dnorm(x, mean = mu1, sd = 1, log = T)
+     singlelikelihoodsY = dnorm(y, mean = mu2, sd = sqrt(1/2), log = T)
+     sumll = sum(singlelikelihoodsX) + sum(singlelikelihoodsY)
+     return(sumll)
+ }
> # log prior
> prior <- function(param){
+     mu1 = param[1]
+     mu2 = param[2]
+     return(-0.5*(3*mu1^2 + 4*mu1*mu2 + 3*mu2^2))
+ }
> # log posterior distribution up to a constant
> posterior <- function(param, x, y){
+    return (likelihood(param, x, y) + prior(param))
+ }
> # proposal function
> proposalfunction <- function(param, delta){
+   mu1 = param[1]
+   mu2 = param[2]
+   new.mu1 = rnorm(1, mu1, delta)
+   new.mu2 = rnorm(1, mu2, delta)
+
+   return(as.vector(c(new.mu1, new.mu2)))
+ }
> # log prob of proposal function
> proposal.prob <- function(old.param, new.param, delta){
+   mu1 = old.param[1]
+   mu2 = old.param[2]
+   new.mu1 = new.param[1]
+   new.mu2 = new.param[2]
+
+   return(dnorm(new.mu1, mean = mu1, sd = delta, log = T) +
+            dnorm(new.mu2, mean = mu2, sd= delta, log=T))
+ }
> # Metropolis algorithm
> run_metropolis_MCMC <- function(x, y, startvalue, iterations, delta){
+     chain = array(dim = c(iterations,2))
+     chain[1,] = startvalue
+     for (i in 1:(iterations-1)){
+         proposal = proposalfunction(chain[i,], delta)
+         probab = exp(posterior(proposal,x, y) +
+                        proposal.prob(proposal, chain[i,], delta) -
+                        posterior(chain[i,], x, y) -
+                        proposal.prob(chain[i,], proposal, delta))
+         if (runif(1) < probab){
+             chain[i+1,] = proposal
+         }else{
+             chain[i+1,] = chain[i,]
+         }
+     }
+     return(chain)
+ }
> # number of iterations
> nreps <- 2000
> # run two MH chains
> MHS1 = run_metropolis_MCMC(x, y, startvalue = c(0,0), nreps, delta = 1/10)
```

```
> MHS2 = run_metropolis_MCMC(x, y, startvalue = c(2,-1), nreps, delta = 1/10)
```

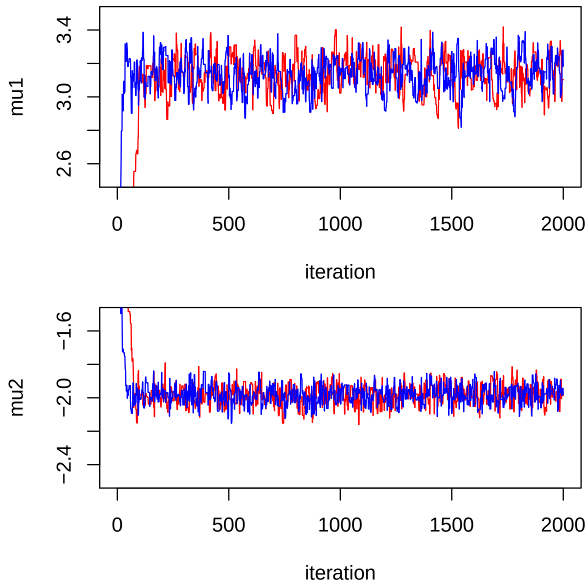
Make a trace plot for each of parameters.

```
> par(mfrow=c(2,1), mar=c(4,4,1,1))
> plot(1:nreps, MHS1[,1], type="l", col="red",
+      ylim = c(min(MHS1[,1],MHS2[,1]), max(MHS1[,1],MHS2[,1])),
+      xlab = "iteration", ylab ="mu1")
> points(1:nreps, MHS2[,1], type="l", col="blue")
> plot(1:nreps, MHS1[,2], type="l", col="red",
+      ylim = c(min(MHS1[,2],MHS2[,2]), max(MHS1[,2],MHS2[,2])),
+      xlab = "iteration", ylab ="mu2")
> points(1:nreps, MHS2[,2], type="l", col="blue")
```



Let's zoom in on the trace plot.

```
> par(mfrow=c(2,1), mar=c(4,4,1,1))
> plot(1:nreps, MHS1[,1], type="l", col="red", ylim = c(2.5, 3.5),
+      xlab = "iteration", ylab ="mu1")
> points(1:nreps, MHS2[,1], type="l", col="blue")
> plot(1:nreps, MHS1[,2], type="l", col="red", ylim = c(-2.5, -1.5),
+      xlab = "iteration", ylab ="mu2")
> points(1:nreps, MHS2[,2], type="l", col="blue")
```

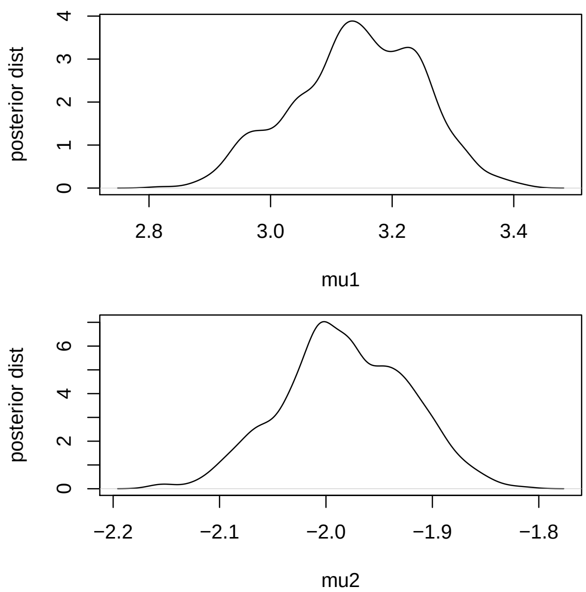


The trace plots show that samples from different chains are mixed well and behave similarly after 2000 iterations.

(b) **Solution** We will remove the first 300 samples as burn-in period.

1) make a plot that shows empirical (estimated) marginal posterior distribution.

```
> par(mfrow=c(2,1), mar=c(4,4,1,1))
> plot(density(MHS1[-(1:300),1]), ylab="posterior dist", xlab="mu1", main="")
> plot(density(MHS1[-(1:300),2]), ylab="posterior dist", xlab="mu2", main="")
```



2) estimate marginal posterior mean.

```
> # for mu1
> mean(MHS1[-(1:300),1])

[1] 3.142711

> # for mu2
> mean(MHS1[-(1:300),2])
```

```
[1] -1.982071
```

3) report a 90% credible interval for the marginal posterior distribution.

```
> # for mu1
> quantile(MHS1[-(1:300),1], probs=c(0.05, 0.95))

      5%      95%
2.955025 3.305431

> # for mu2
> quantile(MHS1[-(1:300),2], probs=c(0.05, 0.95))

       5%       95%
-2.081424 -1.889318
```

Problem 3: Posterior inference using Variational Inference (VI)

(a) **Solution** From problem 1, we have

$$\begin{aligned}
&\log p(\mu_1, \mu_2, x_1, \cdots, x_n, y_1, \cdots, y_m) \\
=& -\frac{1}{2}\left(\sum_{i=1}^{n}(x_i-\mu_1)^2 + 2\sum_{i=1}^{m}(y_i-\mu_2)^2 + (3\mu_1^2 + 4\mu_1\mu_2 + 3\mu_2^2)\right) + \text{const}
\end{aligned}$$

Then

$$\begin{aligned}
\log q_{\mu_1}^*(\mu_1) &= -\frac{1}{2}E_{\mu_2}\left[\sum_{i=1}^{n}(x_i-\mu_1)^2 + 2\sum_{i=1}^{m}(y_i-\mu_2)^2 + (3\mu_1^2 + 4\mu_1\mu_2 + 3\mu_2^2)\right] + \text{const} \\
&= -\frac{1}{2}\left(\sum_{i=1}^{n}(x_i-\mu_1)^2 + E_{\mu_2}\left[2\sum_{i=1}^{m}(y_i-\mu_2)^2\right] + 3\mu_1^2 + 4\mu_1 E_{\mu_2}(\mu_2)\right) + \text{const} \\
&= -\frac{1}{2}\left((n+3)\mu_1^2 - 2\left(\sum_{i=1}^{n} x_i - 2E_{\mu_2}(\mu_2)\right)\mu_1\right) + \text{const}
\end{aligned}$$

Hence, $q_{\mu_1}^*(\mu_1)$ is the pdf of $N(\mu_1^*, \sigma_1^{2*})$, where $\mu_1^* = \frac{\sum_{i=1}^{n} x_i - 2E_{\mu_2}(\mu_2)}{n+3}$ and $\sigma_1^{2*} = \frac{1}{n+3}$.
Similarly,

$$\begin{aligned}
\log q_{\mu_2}^*(\mu_2) &= -\frac{1}{2}E_{\mu_1}\left[\sum_{i=1}^{n}(x_i-\mu_1)^2 + 2\sum_{i=1}^{m}(y_i-\mu_2)^2 + (3\mu_1^2 + 4\mu_1\mu_2 + 3\mu_2^2)\right] + \text{const} \\
&= -\frac{1}{2}\left(E_{\mu_1}\left[\sum_{i=1}^{n}(x_i-\mu_1)^2\right] + 2\sum_{i=1}^{m}(y_i-\mu_2)^2 + 4E_{\mu_1}(\mu_1)\mu_2 + 3\mu_2^2\right) + \text{const} \\
&= -\frac{1}{2}\left((2m+3)\mu_2^2 - 2\left(2\sum_{i=1}^{m} y_i - 2E_{\mu_1}(\mu_1)\right)\mu_2\right) + \text{const}
\end{aligned}$$

Hence, $q_{\mu_2}^*(\mu_2)$ is the pdf of $N(\mu_2^*, \sigma_2^{2*})$, where $\mu_2^* = \frac{2\sum_{i=1}^{m} y_i - 2E_{\mu_1}(\mu_1)}{2m+3}$ and $\sigma_2^{2*} = \frac{1}{2m+3}$.

(b) **Solution**

$$\begin{aligned}
&ELBO(q_{\mu_1}^*(\mu_1), q_{\mu_2}^*(\mu_2)) = ELBO(\mu_1^*, \mu_2^*) \\
=& E_{\mu_1,\mu_2}\left[\log p(\mu_1, \mu_2, x_1, \cdots, x_n, y_1, \cdots, y_m)\right] - E_{\mu_1,\mu_2}\left[\log(q_{\mu_1}^*(\mu_1))\right] - E_{\mu_1,\mu_2}\left[\log(q_{\mu_2}^*(\mu_2))\right]
\end{aligned}$$

Each term is computed as follow,

$$
\begin{aligned}
&E_{\mu_1,\mu_2}\left[\log p(\mu_1,\mu_2,x_1,\cdots,x_n,y_1,\cdots,y_m)\right] \\
=&-\frac{1}{2}E_{\mu_1,\mu_2}\left[\sum_{i=1}^{n}(x_i-\mu_1)^2+2\sum_{i=1}^{m}(y_i-\mu_2)^2+(3\mu_1^2+4\mu_1\mu_2+3\mu_2^2)\right]+\text{const} \\
=&-\frac{1}{2}\left[\sum_{i=1}^{n}E_{\mu_1}(x_i-\mu_1)^2+2\sum_{i=1}^{m}E_{\mu_2}(y_i-\mu_2)^2+3(\sigma_1^{2*}+\mu_1^{*2})+4\mu_1^*\mu_2^*+3(\sigma_2^{2*}+\mu_2^{*2})\right]+\text{const} \\
=&-\frac{1}{2}\left[\sum_{i=1}^{n}\left(\sigma_1^{2*}+\mu_1^{*2}-2x_i\mu_1^*\right)+2\sum_{i=1}^{m}\left(\sigma_2^{2*}+\mu_2^{*2}-2y_i\mu_2^*\right)+3(\sigma_1^{2*}+\mu_1^{*2})+4\mu_1^*\mu_2^*+3(\sigma_2^{2*}+\mu_2^{*2})\right]+\text{const}
\end{aligned}
$$

$$
\begin{aligned}
&E_{\mu_1,\mu_2}\left[\log(q_{\mu_1}^*(\mu_1))\right] \\
=&-\frac{1}{2}E_{\mu_1,\mu_2}\left[\frac{(\mu_1-\mu_1^*)^2}{\sigma_1^{2*}}\right]+\text{const} \\
=&-\frac{1}{2}+\text{const}
\end{aligned}
$$

$$
\begin{aligned}
&E_{\mu_1,\mu_2}\left[\log(q_{\mu_2}^*(\mu_2))\right] \\
=&-\frac{1}{2}E_{\mu_1,\mu_2}\left[\frac{(\mu_2-\mu_2^*)^2}{\sigma_2^{2*}}\right]+\text{const} \\
=&-\frac{1}{2}+\text{const}
\end{aligned}
$$

(c) **Solution**

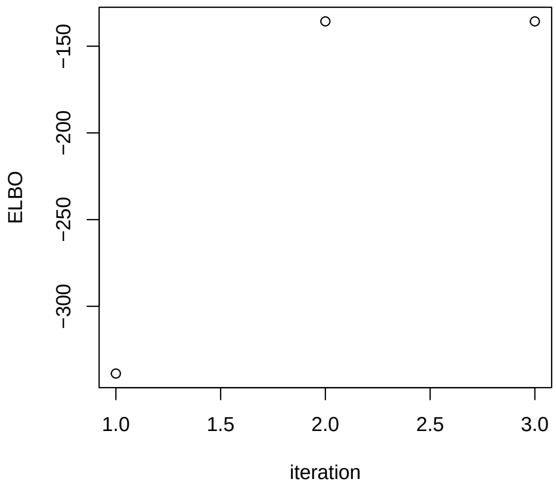
Implementation CAVI algorithm

```
> # x y : data
> # initial values for mu1*, sigma1.2*, mu2*, sigma2.2*:
> #           mu1.vi.init, sigma1.2.vi.init, mu2.vi.init, sigma2.2.vi.init
> # epsilon : If the ELBO has changed by less than epsilon,
> #           the CAVI algorithm will stop
> # max.iter: maximum number of iteration
> cavi.normal <- function(x, y, mu1.vi.init, sigma1.2.vi.init,
+                         mu2.vi.init, sigma2.2.vi.init, epsilon=1e-5, max.iter=100){
+
+   n = length(x)
+   m = length(y)
+
+   mu1.vi = mu1.vi.init
+   sigma1.2.vi = sigma1.2.vi.init
+   mu2.vi = mu2.vi.init
+   sigma2.2.vi = sigma2.2.vi.init
+
+   # store the ELBO for each iteration
+   elbo = c()
+
+   # I will store mu1*, sigma1.2*, mu2*, sigma2.2* for each iteration
+   mu1.vi.list = sigma1.2.vi.list = mu2.vi.list = sigma2.2.vi.list = c()
+
+   # compute the ELBO using initial values of mu1*, sigma1.2*, mu2*, sigma2.2*
+   Elogq.mu1 = Elogq.mu2 = -1/2
+   # Elogp.x.y.mu1.mu2
```

```
+   A = sigma1.2.vi + mu1.vi^2 - 2*x*mu1.vi + x*x
+   B = sigma2.2.vi + mu2.vi^2 - 2*y*mu2.vi + y*y
+   Elogp.x.y.mu1.mu2 = -0.5*sum(A) - sum(B) -
+     0.5*(3*(sigma1.2.vi + mu1.vi^2) + 4*mu1.vi*mu2.vi + 3*(sigma2.2.vi + mu2.vi^2))
+
+   elbo = c(elbo, Elogp.x.y.mu1.mu2 -Elogq.mu1 - Elogq.mu2)
+
+   mu1.vi.list = c(mu1.vi.list, mu1.vi)
+   sigma1.2.vi.list = c(sigma1.2.vi.list, sigma1.2.vi)
+   mu2.vi.list = c(mu2.vi.list, mu2.vi)
+   sigma2.2.vi.list = c(sigma2.2.vi.list, sigma2.2.vi)
+
+   # set the change in the ELBO with 1
+   delta.elbo = 1
+
+   # number of iteration
+   n.iter = 1
+
+   # If the elbo has changed by less than epsilon, the CAVI will stop.
+   while((delta.elbo > epsilon) & (n.iter <= max.iter)){
+
+     # Update mu1.vi and sigma1.2.vi
+     mu1.vi = (sum(x) - 2*mu2.vi)/(n + 3)
+     sigma1.2.vi = 1/(n+3)
+
+     # Update mu2.vi and sigma2.2.vi
+     mu2.vi = (2*sum(y) - 2*mu1.vi)/(2*m + 3)
+     sigma2.2.vi = 1/(2*m + 3)
+
+     # compute the ELBO using the current values of mu1*, sigma1.2*, mu2*, sigma2.2*
+     Elogq.mu1 = Elogq.mu2 = -1/2
+     # Elogp.x.y.mu1.mu2
+     A = sigma1.2.vi + mu1.vi^2 - 2*x*mu1.vi + x*x
+     B = sigma2.2.vi + mu2.vi^2 - 2*y*mu2.vi + y*y
+     Elogp.x.y.mu1.mu2 = -0.5*sum(A) - sum(B) -
+       0.5*(3*(sigma1.2.vi + mu1.vi^2) + 4*mu1.vi*mu2.vi + 3*(sigma2.2.vi + mu2.vi^2))
+
+     elbo = c(elbo, Elogp.x.y.mu1.mu2 -Elogq.mu1 - Elogq.mu2)
+
+     mu1.vi.list = c(mu1.vi.list, mu1.vi)
+     sigma1.2.vi.list = c(sigma1.2.vi.list, sigma1.2.vi)
+     mu2.vi.list = c(mu2.vi.list, mu2.vi)
+     sigma2.2.vi.list = c(sigma2.2.vi.list, sigma2.2.vi)
+
+     # compute the change in the elbo
+     delta.elbo = elbo[length(elbo)]  - elbo[length(elbo)-1]
+
+     # increase the number of iteration
+     n.iter = n.iter + 1
+   }
+
+   return(list(elbo = elbo,
+               mu1.vi.list = mu1.vi.list, sigma1.2.vi.list=sigma1.2.vi.list,
+               mu2.vi.list = mu2.vi.list, sigma2.2.vi.list=sigma2.2.vi.list))
+ }
```

Applying the implemented algorithm. Run the CAVI algorithm with different initial values and check that the ELBO increases at each step by plotting them.

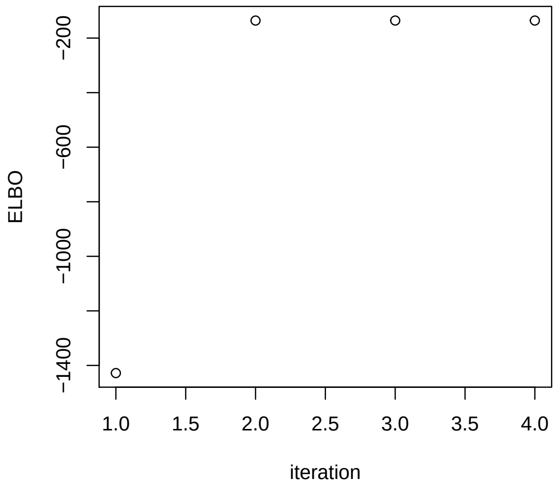
```
> cavi1 = cavi.normal(x, y,
+                     mu1.vi.init = 3, sigma1.2.vi.init = 1,
+                     mu2.vi.init = -2, sigma2.2.vi.init = 1,
+                     epsilon=1e-5, max.iter=100)
> cavi.res = cavi1
> cavi.res$elbo

[1] -338.7661 -135.6699 -135.6699

> plot(cavi.res$elbo, ylab='ELBO', xlab='iteration')
> print(paste("mu1* and sigma1.2* = (",
+             round(cavi.res$mu1.vi.list[length(cavi.res$mu1.vi.list)],2), ",",
+             round(cavi.res$sigma1.2.vi.list[length(cavi.res$sigma1.2.vi.list)],4),
+             ")", sep=""))

[1] "mu1* and sigma1.2* = (3.14,0.0097)"

> print(paste("mu2* and sigma2.2* = (",
+             round(cavi.res$mu2.vi.list[length(cavi.res$mu2.vi.list)],2), ",",
+             round(cavi.res$sigma2.2.vi.list[length(cavi.res$sigma2.2.vi.list)],4),
+             ")", sep=""))

[1] "mu2* and sigma2.2* = (-1.98,0.0033)"
```




```
> cavi2 = cavi.normal(x, y,
+                     mu1.vi.init = 0, sigma1.2.vi.init = 1,
+                     mu2.vi.init = 0, sigma2.2.vi.init = 1,
+                     epsilon=1e-5, max.iter=100)
> cavi.res = cavi2
> cavi.res$elbo

[1] -1428.0667  -135.7461  -135.6699  -135.6699

> plot(cavi.res$elbo, ylab='ELBO', xlab='iteration')
> print(paste("mu1* and sigma1.2* = (",
+             round(cavi.res$mu1.vi.list[length(cavi.res$mu1.vi.list)],2), ",",
+             round(cavi.res$sigma1.2.vi.list[length(cavi.res$sigma1.2.vi.list)],4),
+             ")", sep=""))

[1] "mu1* and sigma1.2* = (3.14,0.0097)"
```

```
> print(paste("mu2* and sigma2.2* = (",
+             round(cavi.res$mu2.vi.list[length(cavi.res$mu2.vi.list)],2), ",",
+             round(cavi.res$sigma2.2.vi.list[length(cavi.res$sigma2.2.vi.list)],4),
+             ")", sep=""))
```

```
[1] "mu2* and sigma2.2* = (-1.98,0.0033)"
```



The two CAVI runs have equally highest ELBO. You can see that approximated posterior distributions from the runs are the same. I will use the output from the first run: $q^*_{\mu_1}(\mu_1)$ is a pdf of $\mathrm{N}(3.14, 0.0097)$ and $q^*_{\mu_2}(\mu_2)$ is a pdf of $\mathrm{N}(-1.98, 0.0033)$.