Using Gibbs sampler and EM algorithm to find posterior mean for a normal model with conjugate priors

Chao Yang

Mar 26 2020

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

1. An overview of the two methods from an example

In this report we will use two methods, the sampling method (Gibbs sampler) and approximating method (expectation-maximization, EM for short) to find the posterior mean for a normal model with conjugate priors.

Suppose the observed data is $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T \overset{iid}{\sim} N(\mu, \sigma^2)$, where both μ and σ^2 are unknown. But we are more interested in μ (and thus σ^2 is a nuisance parameter). We assume independent (conjugate) priors for μ, σ^2. Specifically, we suppose $\mu \sim N(\mu_0, \sigma_0^2)$ and non-informative prior for σ^2, that is, $p(\sigma^2) \propto \frac{1}{\sigma^2}$ (or equivalently, $p(log\sigma) \propto 1$)

1. The Gibbs sampler approach

The unknown parameters are $\boldsymbol{\theta} = (\mu, \sigma^2)$.

The joint density is $p(\mu, \sigma^2|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mu, \sigma^2)p(\mu)p(\sigma^2)$

$$\propto \Pi_{i=1}^n (2\pi\sigma^2)^{-\frac{1}{2}} exp(-\frac{1}{2\sigma^2}(y_i-\mu)^2) \times (2\pi\sigma_0^2)^{-\frac{1}{2}} exp(-\frac{1}{2\sigma_0^2}(\mu-\mu_0)^2) \times \frac{1}{\sigma^2}$$

$$\propto (\sigma^2)^{-\frac{n}{2}-1} exp[-\frac{1}{\sigma^2}\frac{1}{2}\Sigma_{i=1}^N (y_i-\mu)^2] exp(-\frac{1}{2\sigma_0^2}(\mu-\mu_0)^2) - (1)$$

We will then use precision insteand of variance in the following derivation.

Let $\tau = \frac{1}{\sigma^2} = g(\sigma^2)$ and $\tau_0 = \frac{1}{\sigma_0^2}$, and we have $\sigma^2 = g^{-1}(\tau)$ and $|\frac{d}{d\tau}\sigma^2| = |\frac{d}{d\tau}g^{-1}(\tau)| = |\frac{d}{d\tau}(\frac{1}{\tau})| => \frac{d\sigma^2}{d\tau} = \tau^{-2}$ and $f_{\sigma^2}(\sigma^2) = f_\tau(g^{-1}(\tau))|\frac{d}{d\tau}g^{-1}(\tau)| = f_\tau(g^{-1}(\tau))\tau^{-2}$

Thus, the above joint density becomes

$$p(\mu, \tau|\boldsymbol{y}) \propto (\tau)^{\frac{n}{2}+1-2} exp[-\tau\frac{1}{2}\Sigma_{i=1}^N (y_i-\mu)^2] exp(-\frac{\tau_0}{2}(\mu-\mu_0)^2)$$

Therefore, it can be recognized that

$$p(\tau|\mu, \boldsymbol{y}) \propto \Gamma(\frac{n}{2}, \frac{1}{2}\Sigma_{i=1}^N (y_i-\mu)^2)$$

and that

$p(\mu|\tau, \mathbf{y}) \sim N(Q_\mu^{-1} l_\mu, Q_\mu^{-1})$, where $Q_\mu = n\tau + \tau_0, l_\mu = n\tau\bar{y} + \tau_0\mu_0$ (that is, $\tilde{\mu} = \frac{n\tau}{n\tau+\tau_0}\bar{y} + \frac{\tau_0}{n\tau+\tau_0}\mu_0$)

We will run Markov chain Monte Carlo for 20000 interations with the first 10000 as burn-in that will be discarded for posterior inference.

2. EM algorithm

Again we will use the joint density (the complete data likelihood) from (1):

$p(\mu, \tau|\mathbf{y}) \propto (\tau)^{\frac{n}{2}+1-2} exp[-\tau\frac{1}{2}\Sigma_{i=1}^N (y_i - \mu)^2] exp(-\frac{\tau_0}{2}(\mu - \mu_0)^2)$

Taking logrithm, we have log complete data likelihood as follows:

$l(\mu, \tau) = log(p(\mu, \tau|\mathbf{y})) \stackrel{c}{=} log[(\tau)^{\frac{n}{2}+1-2} exp[-\tau\frac{1}{2}\Sigma_{i=1}^N (y_i - \mu)^2] exp(-\frac{\tau_0}{2}(\mu - \mu_0)^2)]$

$\stackrel{c}{=} (\frac{n}{2} - 1)log(\tau) - \tau\Sigma_{i=1}^N (y_i - \mu)^2 - \frac{\tau_0}{2}(\mu - \mu_0)^2$

1. E-step

We wish to integrate out nuisance parameter τ. Taking expectation with respect to $\tau|\mathbf{y}, \mu^{(t-1)}$, and using the fact that $\tau \sim \Gamma(\frac{n}{2}, \frac{1}{2}\Sigma_{i=1}^N (y_i - \mu^{(t-1)})^2)$, we have:

$Q(\mu, \mu^{(t-1)}) = E_{\tau|\mathbf{y},\mu^{(t-1)}}(l(\mu, \tau))$,

$\stackrel{c}{=} -E_{\tau|\mathbf{y},\mu^{(t-1)}}(\tau)\Sigma_{i=1}^N (y_i - \mu)^2 - \frac{\tau_0}{2}(\mu - \mu_0)^2$

$\stackrel{c}{=} k\Sigma_{i=1}^N (y_i - \mu)^2 - \frac{\tau_0}{2}(\mu - \mu_0)^2$

where $k = \frac{n/2}{\frac{1}{2}\Sigma_{i=1}^N (y_i - \mu^{(t-1)})^2)} = \frac{n}{\Sigma_{i=1}^N (y_i - \mu^{(t-1)})^2)}$ is a constant not involving parameter of interest μ

Note that we did not explicitly calculate $E_{\tau|\mu^{(t-1)},\mathbf{y}}((\frac{n}{2} - 1)log(\tau))$ since this term does not involve μ and thus the term will be dropped in the M-step ($\frac{d}{d\mu}C = 0$, and therefore it is absorbed in "up to a constant" notation).

2. M-step

$\frac{d}{d\mu}Q(\mu, \mu^{(t-1)}) \stackrel{set}{=} 0$, where $k = \frac{n}{\Sigma_{i=1}^n (y_i - \mu^{(t-1)})^2}$

$=> \mu^{(t)} = \frac{\tau_0\mu_0 + k \times \Sigma_{i=1}^n y_i}{nk + \tau_0}$

Given initial values for μ, we can then iteratively find the estimate for $[\mu|\mathbf{y}]$ when some criteria are met.

We consider using two criteria, absolute relative difference and relative difference. The first is that $|\mu^{(t)} - \mu^{(t-1)}| < \epsilon$, and the other is that $|\frac{|\mu^{(t)} - \mu^{(t-1)}}{\mu^{(t-1)}}| < \epsilon$, for a small $\epsilon > 0$.

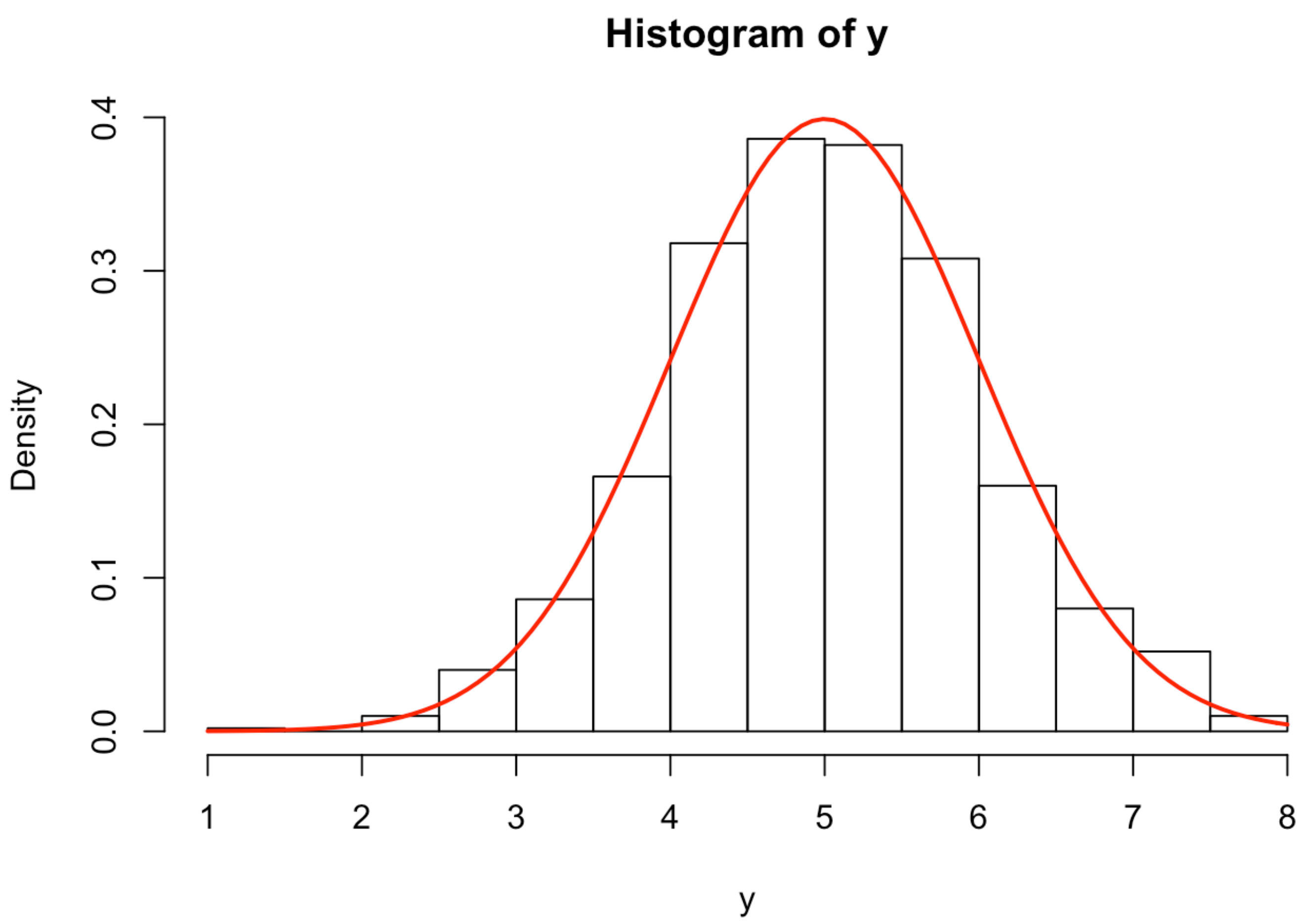
2. Implementation

We assume the true underlying distribution for $\mathbf{y}$ was $N(5, 1^2)$.

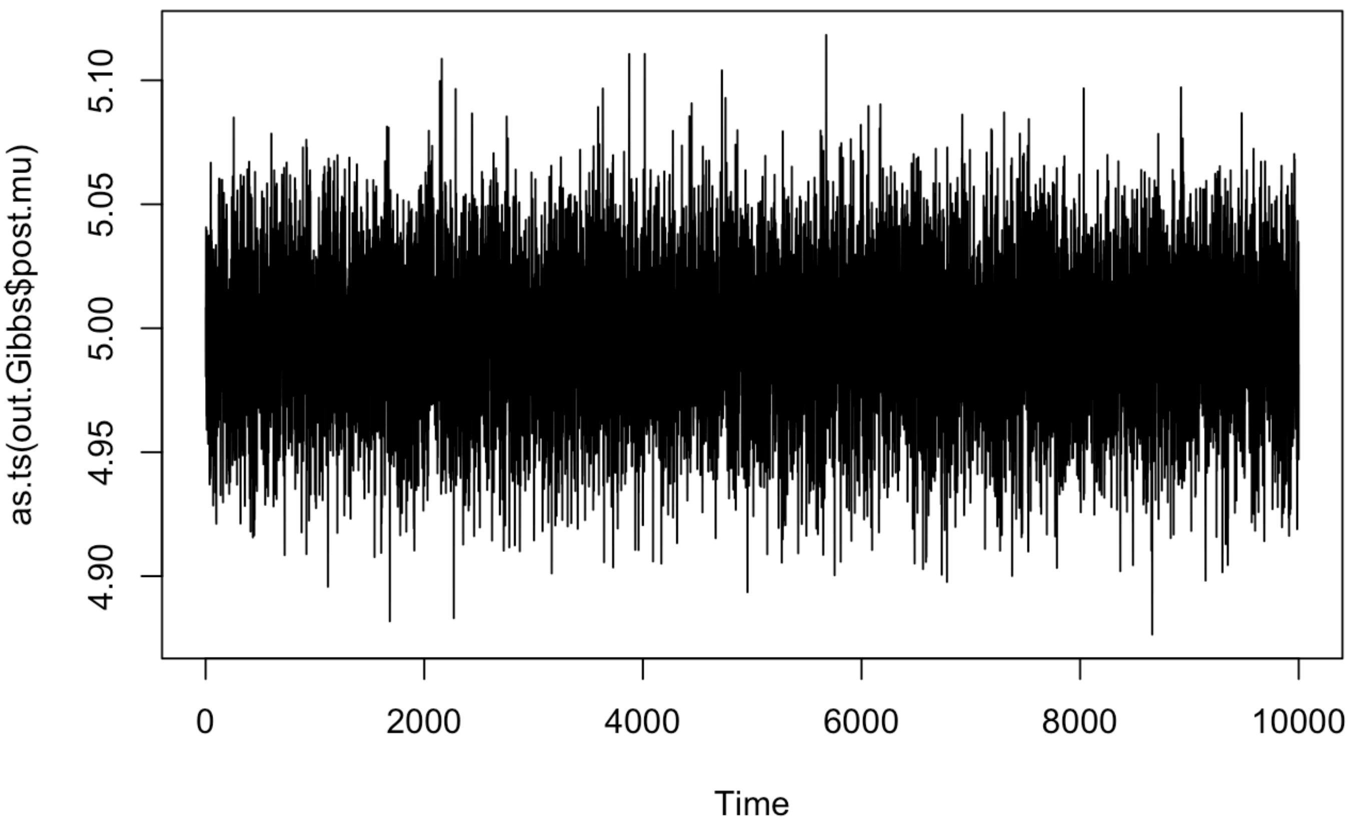
Priors: $\mu \sim N(2, (\sqrt{2})^2), p(\tau) \propto 1/\tau$

The initial values for (μ, τ) were $(100, 10)$

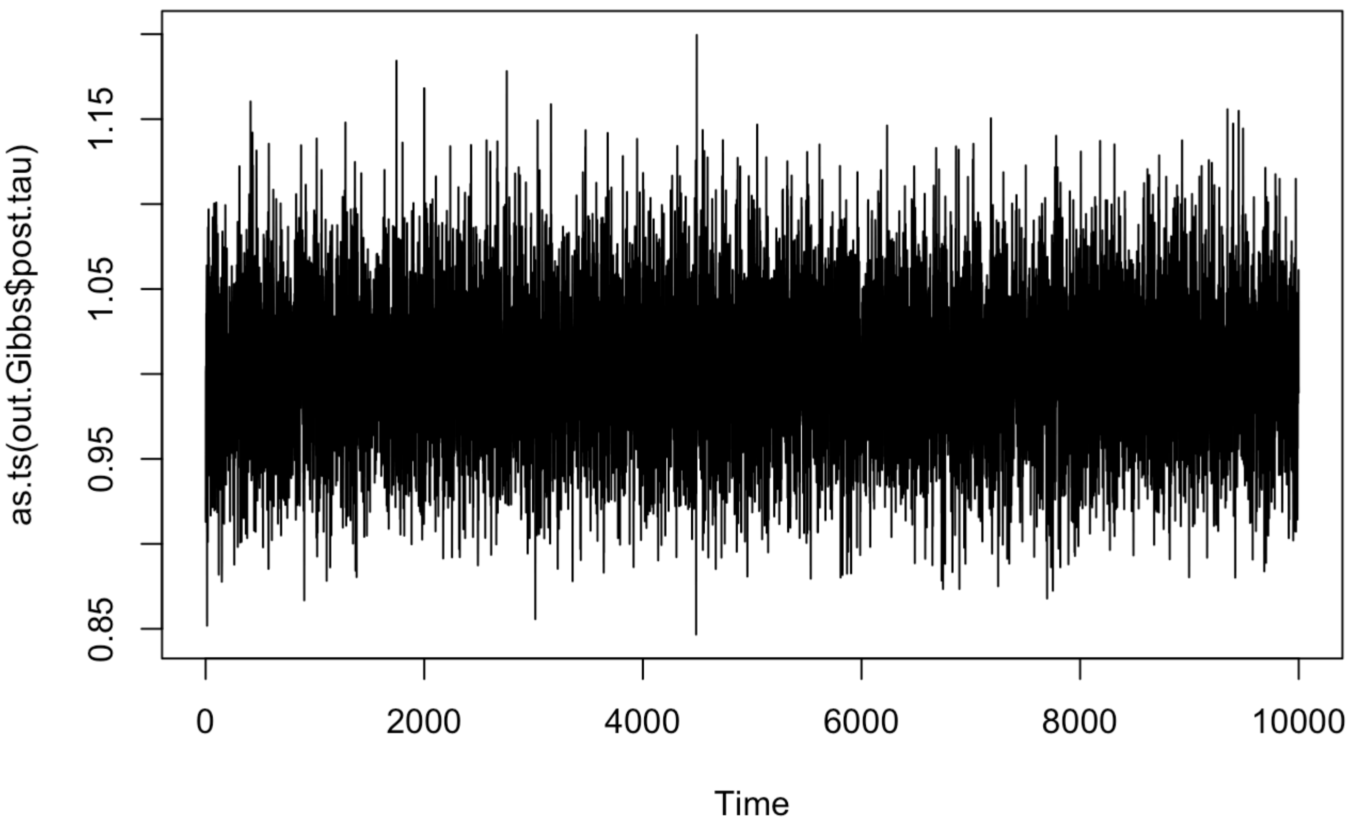
```
# using Gibbs sampler and EM algo. to find posterior mean (variance as a nuisance par
ameter)

calc4Mu= function(y, mu0, tau, tau0) {
    n= length(y)
    Q= n* tau+ tau0
    l= tau* sum(y)+ tau0* mu0
    # return mean and precision
    return(c(Q^-1*l, Q))
}

set.seed(21287)
n= 1000
y= rnorm(n= n, mean= 5, sd= 1)
hist(y, freq= F)
curve(dnorm(x, mean= 5, sd= 1), lwd= 2, col= 2, add= T)
```



```r
mu0= 2
tau0= 1/2

# method 1, Gibbs sampler
S= 10000
post.mu= numeric(S)
post.tau= numeric(S)

init1= list(mu= 100, tau= 10)

runGibbs= function(S, init, y, mu0, tau0, nBurnin) {
    n= length(y)
    mu= init$mu
    tau= init$tau

    for(s in 1: S) {
        muPara= calc4Mu(y= y, mu0= mu0, tau= tau, tau0= tau0)
        mu= rnorm(n= 1, mean= muPara[1], sd= sqrt(1/muPara[2]))
        tau= 1/ rgamma(n= 1, shape= n/2, rate= 1/2* sum((y-mu)^2) )
        post.mu[s]= mu
        post.tau[s]= tau
    }
    return(list(post.mu= post.mu[-c(1:nBurnin)],
                post.tau= post.tau[-c(1:nBurnin)]))

}

set.seed(21201)
out.Gibbs= runGibbs(S= 2*10^4, init= init1, y= y, mu0= mu0, tau0= tau0,
                    nBurnin= 10^4)
plot(as.ts(out.Gibbs$post.mu))
```




```
plot(as.ts(out.Gibbs$post.tau))
```




```
mean(out.Gibbs$post.mu) #  4.995631
```

```
## [1] 4.995631
```

```
mean(1/out.Gibbs$post.tau) # 1.000062
```

```
## [1] 1.000062
```

```r
# method 2, EM algo.
EMUpdate= function(y, mu0, tau0, mu) {
    k= n/sum((y-mu)^2)
    mu.new= (tau0* mu0+ k* sum(y))/(n*k+ tau0)
    #a= tau0* mu0+ n*sum(y)/sum((y-mu)^2)
    #b= n* n/sum((y-mu)^2)+tau0
    #mu.new= a/b
    return(mu.new)
}

# (1) use abs(rel.diff)<epsilon as stopping rule
runEM.1= function(y, init, mu0, tau0, epsilon) {
    mu.vec= c()
    n= length(y)
    mu= init$mu
    tau= init$tau
    count= 0
    #diff= 1
    rel.diff= 1

    while(abs(rel.diff)> epsilon) {
        mu= EMUpdate(y= y, mu0= mu0, tau0= tau0, mu= mu)
        mu.vec= c(mu.vec, mu)
        if(count> 1) {
            #diff= mu.vec[count]- mu.vec[count-1]
            rel.diff= (mu.vec[count]- mu.vec[count-1])/mu.vec[count-1]
        }
        count= count+ 1
    }
    return(list(mu= mu.vec))
}

out.EM.1= runEM.1(y= y, init= init1, mu0= mu0, tau0= tau0, epsilon=1e-10)
out.EM.1$mu
```

```
## [1] 2.543638 4.986740 4.995725 4.995725 4.995725 4.995725
```

```r
out.EM.1$mu[length(out.EM.1$mu)]
```

```
## [1] 4.995725
```

```
# (2) use abs(diff)<epsilon as stopping rule
runEM.2= function(y, init, mu0, tau0, epsilon) {
    mu.vec= c()
    n= length(y)
    mu= init$mu
    tau= init$tau
    count= 0
    diff= 1

    while(abs(diff)> epsilon) {
        mu= EMUpdate(y= y, mu0= mu0, tau0= tau0, mu= mu)
        mu.vec= c(mu.vec, mu)
        if(count> 1) {
            diff= mu.vec[count]- mu.vec[count-1]
            #rel.diff= (mu.vec[count]- mu.vec[count-1])/mu.vec[count-1]
        }
        count= count+ 1
    }
    return(list(mu= mu.vec))
}

out.EM.2= runEM.2(y= y, init= init1, mu0= mu0, tau0= tau0, epsilon=1e-10)
out.EM.2$mu
```

```
## [1] 2.543638 4.986740 4.995725 4.995725 4.995725 4.995725
```

```
out.EM.2$mu[length(out.EM.2$mu)]
```

```
## [1] 4.995725
```

3. Results

Though we used an unlikely initial guess of μ as start point, the chains for μ and τ mixed well and seemed to converge. The posterior mean was 4.995631 for μ and 1.000062 for τ.

Using EM, either using absolute relative difference/ difference criteria, the estimate for posterior μ was 4.995725.

Esitmates for posterior μ from both methods were quite close to the true value of 5.

Appendix- A note on Kullback Leiber (KL) divergence

Suppose P and Q are distributions of a continuous r.v., with densities p and q, respectively, KL divergence is defined as:

$$D_{KL}(P||Q) = \int_{-\infty}^{+\infty} ln(\frac{p(x)}{q(x)})p(x)dx = -\int_{-\infty}^{+\infty} ln(\frac{q(x)}{p(x)})p(x)dx$$

Since $f(x) = -ln(x)$ is convex on $(0, +\infty)$, by Jensen inequality, we have $f(E(X)) \leq E(f(X))$

$\because \int_{-\infty}^{+\infty}[-ln(\frac{q(x)}{p(x)})p(x)]dx = E_X(-ln(\frac{q(x)}{p(x)}), X \sim p_X(x)$

$\therefore -ln(E_X(\frac{q(x)}{p(x)})) = -ln\int_{-\infty}^{+\infty}\frac{q(x)}{p(x)}p(x)dx = -ln1 = 0 \leq E_X(-ln(\frac{q(x)}{p(x)}) = D_{KL}(P||Q)$

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.