MTH 511a - 2020: Lecture 27

Instructor: Dootika Vats

1 Bayesian models

In the last video, we introduced the philosophy of Bayesian modeling and worked through two popular examples. In both examples, the posterior distributions were available in closed-form after some algebra tricks.

However, in many situations, the posterior distribution is not easily obtained. Let's first see an example.

Example 1 (t-distribution likelihood). Suppose $Y_1, \dots, Y_n | \mu \sim t_\nu(\mu)$, which is the t distribution with ν degrees of freedom and mean μ. Let's assume that ν is known.

Consider the prior $\mu \sim N(0,1)$ on μ. The posterior distribution of μ is,

$$
\begin{aligned}
\pi(\mu \mid y_1, \dots, y_n) &\propto \pi(\mu) f(y|\mu) \\
&= e^{-\frac{\mu^2}{2}} \prod_{i=1}^{n} \left(\frac{\Gamma(\nu + 1/2)}{\sqrt{\nu\pi}\Gamma(\nu/2)} \left(1 + \frac{(y_i - \mu)^2}{\nu}\right)^{-\frac{\nu+1}{2}} \right) \\
&\propto e^{-\frac{\mu^2}{2}} \prod_{i=1}^{n} \left(1 + \frac{(y_i - \mu)^2}{\nu}\right)^{-\frac{\nu+1}{2}} .
\end{aligned}
$$

This does not look like the functional form of any known density. We may be able to find the MAP estimate using the many optimization techniques that we've learned, however, if we want to find the quantiles and mean of this distribution, it will be further challenging.

The solution to this problem is sampling and Monte Carlo!

Instead of finding the closed form expression of the posterior distribution, we will obtain samples from this posterior distribution and *estimate* posterior quantities using Monte Carlo based estimators.

General sampling framework

We have a target density $\pi(\theta|y)$ whose expectation and quantiles are of interest. Notice that in the above example, we have a target density whose proportionality constant we do not know. We are now in the situation where

$$\pi(\theta|y) = c\,\tilde{\pi}(\theta|y)\,,$$

where c is *unknown*. We have seen this before in importance sampling. Now we see when such instances occur!

We will use two methods to estimate quantities of interest.

- Accept-reject sampling
- Markov chain Monte Carlo (MCMC)

In both methods, we will sample values $X_1, \dots, X_n$ from $\pi(\theta|y)$ and using sample mean and sample quantiles, to estimate the posterior mean and credible intervals.

You are already familiar with accept-reject sampling, however, you only known how to implement it when c is known. Now we will learn how to implement accept-reject when this constant is unknown.

1.1 Accept-Reject Sampling

Recall that, to implement accept-reject on a given target density, we need to find a proposal distribution G with density g such that

$$\sup_{x \in \mathcal{X}} \frac{\pi(x)}{g(x)} \le M < \infty\,.$$

Since the upper bound, M, is subsequently used in the algorithm, M needs to be known constant. However, when

$$\pi(x) = c\tilde{\pi}(x)\,,$$

M cannot be known, since c is unknown. Turns out, accept-reject can be used to sample from a unnormalized distribution as well, however, we lose out on some benefits.

Suppose $g(x)$ is a proposal density such that $g(x) = r\tilde{g}(x)$, where r is a known or unknown constant. Suppose there exists M such that

$$\sup_{x \in \mathcal{X}} \frac{\tilde{\pi}(x)}{\tilde{g}(x)} \le M\,.$$

Notice that,

$$\begin{aligned}
&\sup_{x\in\mathcal{X}} \frac{\tilde{\pi}(x)}{\tilde{g}(x)} \le M \\
\Rightarrow &\sup_{x\in\mathcal{X}} \frac{m\tilde{\pi}(x)}{r\tilde{g}(x)} \le \frac{m}{r}M \\
\Rightarrow &\sup_{x\in\mathcal{X}} \frac{\pi(x)}{g(x)} \le \frac{m}{r}M := M'
\end{aligned}$$

Thus, upper bounding $\tilde{\pi}/\tilde{g}$ by M is similar to upper bounding π/g by the *unknown* M'.

Algorithm 1 Accept-Reject for unknown constants

```
1: Generate $Y \sim G$ and generate $U \sim U[0,1]$
2: If $U \le \frac{\tilde{\pi}(y)}{M\tilde{g}(y)}$, then set $X = Y$
3: Else, go to step 1.
```

Then, the usual AR would be accepted if

$$U \le \frac{\pi(x)}{M'g(x)} \Rightarrow U \le \frac{\tilde{\pi}(x)}{M\tilde{g}(x)}\,.$$

So they are equivalent. Essentially, we don't need to know the normalizing constants for both π and g. However, a significant drawback here is that, the probability of acceptance of any proposal is still

$$\Pr(\text{Acceptance}) = \frac{1}{M'}$$

and since M' is unknown, this probability is unknown. Thus, there isn't any way to assess how efficient the algorithm will be.

Example 2 (Bayesian $t-$likelihood). Recall the posterior distribution

$$\pi(\mu|y) \propto e^{-\mu^2/2} \prod_{t=1}^{n} \left(1 + \frac{(y_i-\mu)^2}{\nu}\right)^{-(\nu+1)/2}$$

Consider the proposal distribution $N(0,1)$.

$$\begin{aligned}
\frac{\tilde{\pi}(\mu)}{\tilde{g}(\mu)} &= \frac{e^{-\mu^2/2} \prod_{t=1}^{n} \left(1 + \frac{(y_i-\mu)^2}{\nu}\right)^{-(\nu+1)/2}}{e^{-\mu^2/2}} \\
&= \prod_{t=1}^{n} \left(1 + \frac{(y_i-\mu)^2}{\nu}\right)^{-(\nu+1)/2}
\end{aligned}$$

$$\leq 1 := M_1\,.$$

This is not a tight bound, but an easily available bound. Nonetheless, we can implement an A-R algorithm, but it is not very efficient.

An alternative is to use $\hat{\mu}_{\text{MLE}}$ as an upper bound. This we known how to do using optimization techniques a similar Cauchy example.

$$\begin{aligned}\frac{\tilde{\pi}(x)}{\tilde{g}(x)} &= \frac{e^{-\mu^2/2}\prod_{t=1}^{n}\left(1+\frac{(y_i-\mu)^2}{\nu}\right)^{-(\nu+1)/2}}{e^{-\mu^2/2}} \\ &= \prod_{t=1}^{n}\left(1+\frac{(y_i-\mu)^2}{\nu}\right)^{-(\nu+1)/2} \\ &\leq \prod_{t=1}^{n}\left(1+\frac{(y_i-\hat{\mu}_{\text{MLE}})^2}{\nu}\right)^{-(\nu+1)/2} := M_2\,.\end{aligned}$$

This will be more efficient, but $\hat{\mu}_{\text{MLE}}$ needs to be found numerically. Notice that the bound gets worse when

- the number of data points, n increases
- if the true value of μ is far from 0.

```
#############################################
## Accept-reject to sample from posterior distrbituon
# from t likelihood and normal priors
# Code takes some time to run
#############################################
library(MASS)
set.seed(10)

#Log posterior
log.post <- function(y, mu, nu)
{
  -mu^2/2 - (nu + 1)/2 *sum( log( 1 + (y-mu)^2/nu ) )
}

# Accept-reject for a given bound M
AR_tmodel <- function(N = 1e2, M = 1)
{
  count <- 0
  attempts <- 0
  samp <- numeric(length = N)
  while(count < N)
  {
    attempts <- attempts + 1
    prop <- rnorm(1)
    ratio <- log.post(y = y, mu = prop, nu = nu) + prop^2/2 - log(M)
    if(exp(ratio) > 1) print(exp(ratio)) # Making sure M is correct
    if(runif(1) < exp(ratio))
    {
      count <- count + 1
      samp[count] <- prop
      if(count%% (N) == 0) print(paste("Accepted = ",count, ", Accept Prob.
         = ", count/attempts))
    }
  }
  return(samp)
}
```

We will now generate data from a t_3 distribution with the true $\mu = 0$ and generate 10000 iid samples from the posterior using the AR with M_1 and M_2.

```
# Generate the data set.
# Small n and mu close to zero (the prior)
nu <- 3
n <- 10
mu <- 0
y <- mu + rt(n, df = nu) # Generate the data

mle <- fitdistr(y, "t", df = 3)$estimate[1] # Find the MLE to construct the
    tighter upper bound for pi(x)/g(x)
M2 <- prod( (1 + (y - mle)^2/nu ) )^(-(nu+1)/2) + 1e-5 # adding a little to
    remove numerical approximation errors

# Run the A-R sampler using the two different upper bounds
system.time(out1 <- AR_tmodel(N = 1e4, M = 1) ) # About 50 seconds
#[1] "Accepted = 10000 , Accept Prob. = 0.00146156500580753"
system.time(out2 <- AR_tmodel(N = 1e4, M = M2) ) # About .11 seconds
#[1] "Accepted = 10000 , Accept Prob. = 0.37267543696195"
```

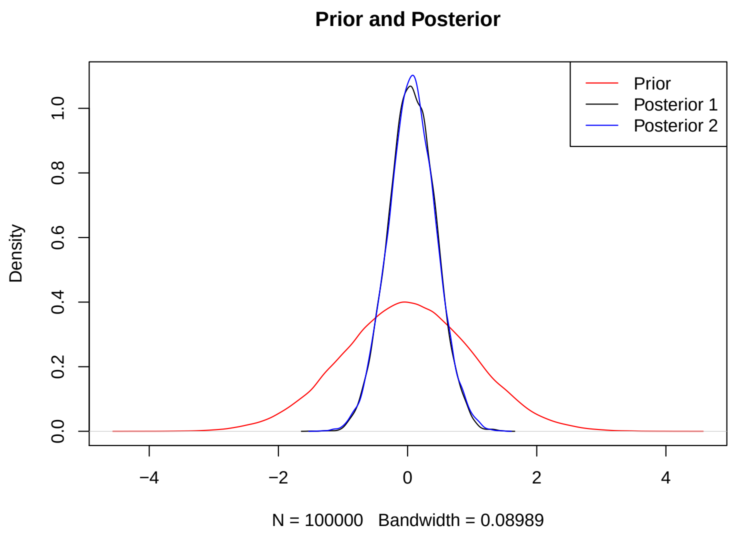
Although the AR algorithms are different, both methods with M_1 and M_2 will yield iid samples from the same posterior. We can check the posterior means and quantiles (for credible intervals) and draw the posterior density estimate plot:

```
# Posterior mean estimates
c(mean(out1), mean(out2))
#[1] 0.06775302 0.06930398

# 95% credible interval
quantile(out1, c(.025, .975))
#     2.5%     97.5%
#-0.6487182 0.7868883
```

```
quantile(out2, c(.025, .975))
#      2.5%      97.5%
#-0.6534764 0.8166141


# Compare the performance
plot(density(rnorm(1e5)), col = "red", type = 'l', ylim = c(0,1.1), main =
    "Prior and Posterior")#prior
lines(density(out1)) # samples posterior 2
lines(density(out2), col = "blue") # sampled posterior 2
legend("topright", legend = c("Prior", "Posterior 1", "Posterior 2"), col =
    c("red", "black", "blue"), lty = 1)
```



We repeat the same but now I increase my data size from $n = 10$ to $n = 20$. This small change will dramatically decrease the acceptance probability for both M_1 and M_2.

```
# Increase n
n <- 20
y <- mu + rt(n, df = nu) # Generate the data

mle <- fitdistr(y, "t", df = 3)$estimate[1] # Find the MLE to construct the
    tighter upper bound for pi(x)/g(x)
M2 <- prod( (1 + (y - mle)^2/nu ) )^(-(nu+1)/2) + 1e-6


# Run the A-R sampler using the two different upper bounds
system.time(out1 <- AR_tmodel(N = 1e1, M = 1) ) # too slow for even 10
    samples
```

```
# [1] "Accepted = 10 , Accept Prob. = 2.39168192161119e-07"
system.time(out2 <- AR_tmodel(N = 1e4, M = M2) ) # About .11 seconds
#[1] "Accepted = 10000 , Accept Prob. = 0.128182121157741"
```

2 Questions to think about

- Repeat the simulation above with the true data generated from $\mu = 5$. How does the AR change? How does the posterior change?
- How do you think AR will fare for higher dimensional problems?