Coursera Notes for Bayesian Statistics: Techniques and Models

Jordan Katz

Week 1

Bayesian Modeling

data y, parameter θ
likelihood: $p(y|\theta)$
prior: $p(\theta)$
posterior: $p(\theta|y)$

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta)p(\theta)d\theta} \propto p(y|\theta)p(\theta)$$

If we do not use conjugate priors, or if the models are more complicated, then the posterior distribution may not have a "standard" or well-known form.

Monte Carlo Estimation

Using simulation to determine some properties of a distribution, e.g. mean, variance, probability of an event, quantiles (which all use integration)

Example: Suppose we have $\theta \sim \text{Ga}(a, b)$ and want to know $E[\theta]$

$$E[\theta] = \int_{-\infty}^{\infty} \theta p(\theta) d\theta = \int_0^{\infty} \theta \frac{b^a}{\Gamma(a)} \theta^{a-1} e^{-b\theta} d\theta = \frac{a}{b}$$

To verify with Monte Carlo, take samples θ_i^* for $i = 1, \dots, m$ from the Gamma distribution. Estimate sample mean as

$$\overline{\theta^*} = \frac{1}{m} \sum_{i=1}^{m} \theta_i^*$$

Suppose we have some function $h(\theta)$ and we want $E[h(\theta)]$. Can estimate

$$E[h(\theta)] = \int h(\theta) p(\theta) d\theta \approx \frac{1}{m} \sum_{i=1}^{m} h(\theta_i^*)$$

In particular, if $h(\theta)$ is $I_A(\theta)$, i.e. the indicator function for some event A, then we can approximate probabilities as well: $Pr[\theta \in A]$.

Question: How good is this estimate from sampling? By the Central Limit Theorem we know

$$\overline{\theta^*} \mathrel{\dot\sim} N\Big(E(\theta), \frac{Var(\theta)}{m}\Big)$$

The variance of the estimate is given by

$$\widehat{Var(\theta)} = \frac{1}{m}\sum_{i=1}^{m}(\theta_i^* - \overline{\theta^*})^2$$

The standard error (SE) is given by

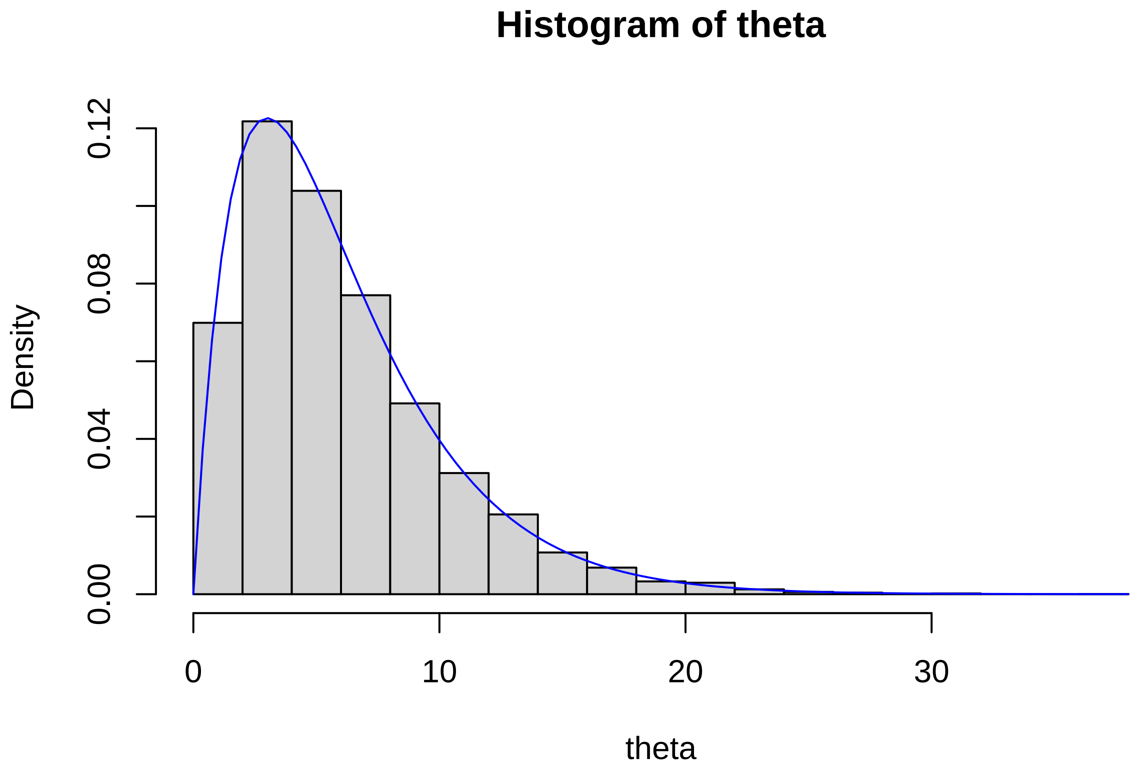
$$\sqrt{\frac{\widehat{Var(\theta)}}{m}}$$

```
set.seed(32)

m=10000
a=2
b=1/3

theta = rgamma(n=m, shape=a, rate=b)

hist(theta, freq=FALSE)
curve(dgamma(x, shape=a, rate=b), col="blue", add=TRUE)
```



```
mean(theta) # Estimated mean
```

```
## [1] 6.022368
```

```
a/b # True mean
```

```
## [1] 6
```

```r
var(theta) # Estimated variance
```

```
## [1] 18.01033
```

```r
a/b^2 # True variance
```

```
## [1] 18
```

```r
ind = theta < 5
mean(ind) # Estimated Prob[theta < 5]
```

```
## [1] 0.4974
```

```r
pgamma(q=5, shape=a, rate=b) # True Prob[theta < 5]
```

```
## [1] 0.4963317
```

```r
quantile(theta, probs=0.9) # Estimated quantile
```

```
##      90%
## 11.74426
```

```r
qgamma(p=0.9, shape=a, rate=b) # True quantile
```

```
## [1] 11.66916
```

```r
se = sd(theta) / sqrt(m) # Standard error of mean
mean(theta) - 2*se # Lower bound CI
```

```
## [1] 5.937491
```

```r
mean(theta) + 2*se # Upper bound CI
```

```
## [1] 6.107245
```

As we can see, Monte Carlo does a pretty good job.

Example: Suppose we have

$$y|\phi \sim Bin(10, \phi)$$

$$\phi \sim Beta(2, 2)$$

and we want to simulate from marginal distribution of y (which can be difficult to do in general). Can do the following procedure:
1. Draw $\phi_i^* \sim Beta(2, 2)$
2. Given ϕ_i^*, draw $y_i^* \sim Bin(10, \phi_i^*)$

Results in a list of independent pairs (y_i^*, ϕ_i^*) drawn from the joint distribution. Discarding the ϕ_i^*s effectively results in a sample from the marginal distribution of y.

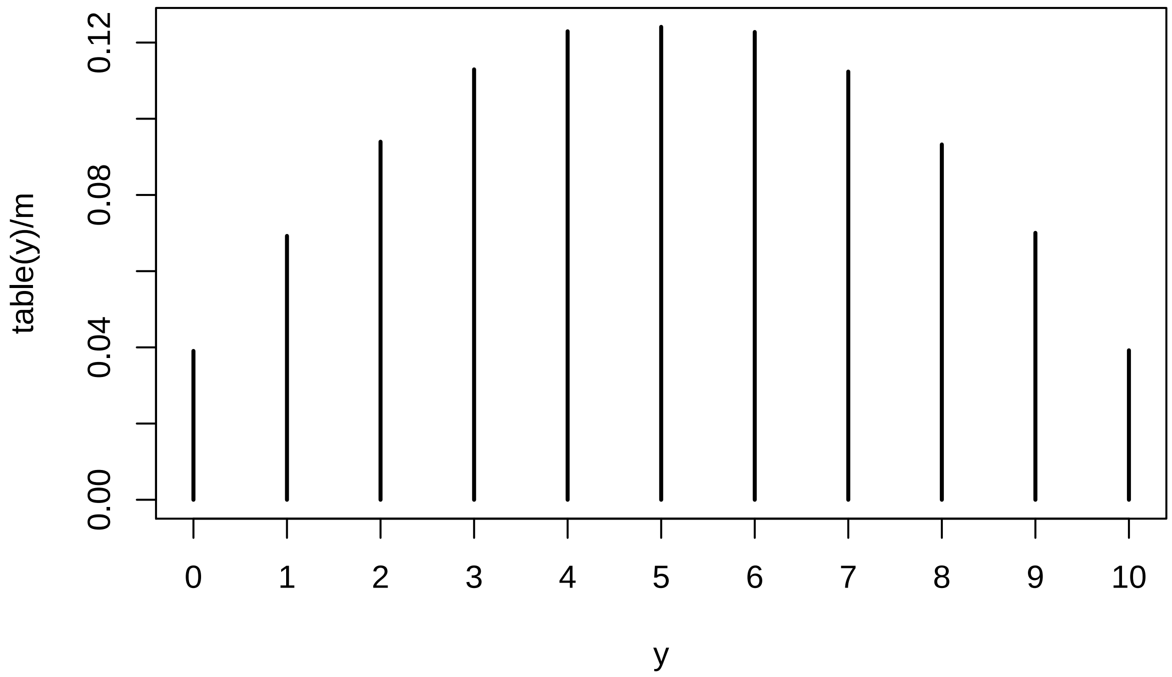
```r
m = 1e5

phi = rbeta(m, shape1=2, shape2=2)
y = rbinom(m, size=10, prob=phi)

table(y) / m
```

```
## y
##       0       1       2       3       4       5       6       7       8       9
## 0.03906 0.06925 0.09398 0.11296 0.12296 0.12412 0.12277 0.11238 0.09325 0.07005
##      10
## 0.03922
```

```r
plot(table(y) / m) # Estimated marginal distribution of y
```



```r
mean(y) # Estimate mean of y
```

```
## [1] 5.00046
```

Week 2

Metropolis-Hastings

Allows us to sample from generic distribution (whose normalizing constant may not be known). To accomplish this, we effectively construct a Markov Chain whose stationary distribution is the target distribution.

Say we want to know $p(\theta)$ but we only know $g(\theta)$ where $p(\theta) \propto q(\theta)$.

Algorithm:

```
1. Select initial value $\theta_0$
2. for $i = 1, \dots, m$ repeat:
    a. Draw candidate $\theta^* \sim q(\theta^*|\theta_{i-1})$
    b. Define $\alpha = \frac{g(\theta^*)/q(\theta^*|\theta_{i-1})}{g(\theta_{i-1})/q(\theta_{i-1}|\theta^*)} = \frac{g(\theta^*)}{g(\theta_{i-1})} \frac{q(\theta_{i-1}|\theta^*)}{q(\theta^*|\theta_{i-1}))}$
        i. if $\alpha \geq 1$:
           accept $\theta^*$ and set $\theta_i \leftarrow \theta^*$
        ii. $0 < \alpha < 1$:
           with prob $\alpha$: accept $\theta^*$ and set $\theta_i \leftarrow \theta^*$
           with prob $1 - \alpha$: reject $\theta^*$ and set $\theta_i \leftarrow \theta_{i-1}$
```

Where q here is the candidate generating distribution which may or may not depend on θ_{i-1}.

One choice is to make q the same distribution regardless of the value θ_{i-1}. If we take this option, we want $q(\theta)$ to be similar to $p(\theta)$ to best approximate it. A high acceptance rate is a good sign here but still may want q to have a larger variance then p to assure we are exploring the space well.

Another choice – one which *does* depend on θ_{i-1} – is to choose a distribution q that is centered on θ_{i-1}.

A common choice for such a distribution is $N(\theta_{i-1}, 1)$, or in order words, a Gaussian random walk: $\theta^* = \theta_{i-1} + N(0, 1)$ In this particular case, we have

$$q(\theta^*|\theta_{i-1}) = \frac{1}{\sqrt{2\pi}} \exp\left[-0.5(\theta^* - \theta_{i-1})^2\right] = q(\theta_{i-1}|\theta^*)$$

The "size" of the random walk step can affect acceptance (and thus convergence) rate. A high acceptance rate is not a good sign here. If random walk is taking too small of steps, it will accept candidate more often but will take a long time to fully explore the space. If it is taking too large of steps, many proposals with have low probabilities which leads to a low acceptance rate. This amounts to "wasted" samples. Ideally, a random walk sampler should have an acceptance rate between 23% and 50%.

In any symmetric case, we have the property $q(a|b) = q(b|a)$, so step 2 in the algorithm above reduces to

$$\alpha = \frac{g(\theta^*)}{g(\theta_{i-1})}$$