Slice sampling

Dr. Jarad Niemi

STAT 615 - Iowa State University

November 14, 2017

Slice sampling

Suppose the target distribution is $p(\theta|y)$ with scalar θ. Then,

$$p(\theta|y) = \int_0^{p(\theta|y)} 1\, du$$

Thus, $p(\theta|y)$ can be thought of as the marginal distribution of

$$(\theta, U) \sim \text{Unif}\{(\theta, u) : 0 < u < p(\theta|y)\}$$

where u is an auxiliary variable.

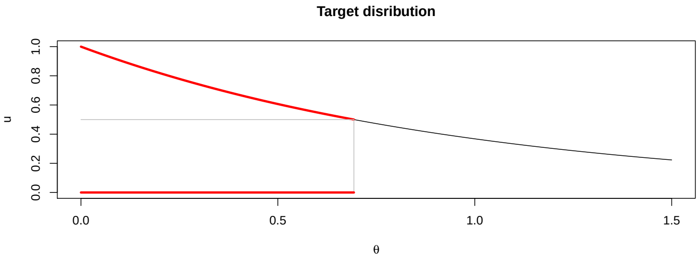
Slice sampling performs the following Gibbs sampler:

1. $u^t|\theta^{t-1}, y \sim \text{Unif}\{u : 0 < u < p(\theta^{t-1}|y)\}$ and
2. $\theta^t|u^t, y \sim \text{Unif}\{\theta : u^t < p(\theta|y)\}$.

Slice sampler for exponential distribution

Consider the target $\theta|y \sim Exp(1)$,then

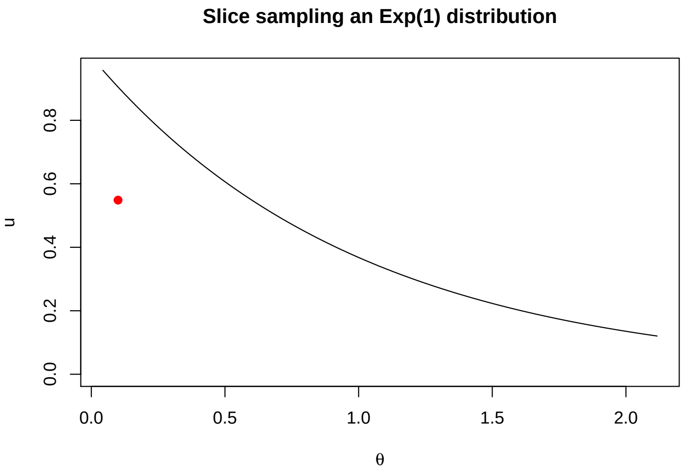
$$\{\theta : u < p(\theta|y)\} = (0, -\log(u)).$$



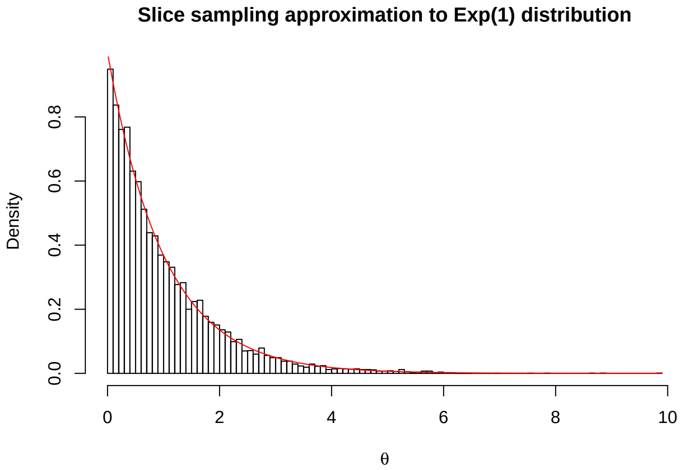
Slice sampling in R

```r
slice = function(n,init_theta,target,A) {
  u = theta = rep(NA,n)
  theta[1] = init_theta
  u[1] = runif(1,0,target(theta[1])) # This never actually gets used

  for (i in 2:n) {
    u[i] = runif(1,0,target(theta[i-1]))
    endpoints = A(u[i],theta[i-1]) # The second argument is used in the second example
    theta[i] = runif(1, endpoints[1],endpoints[2])
  }
  return(list(theta=theta,u=u))
}
```

```r
set.seed(6)
A = function(u,theta=NA) c(0,-log(u))
res = slice(10, 0.1, dexp, A)
```


Slice sampling an Exp(1) distribution
u
0.0
0.2
0.4
0.6
0.8
0.0
0.5
1.0
1.5
2.0
θ

Histogram of draws



Normal model with unknown mean

Let

$$Y_i \stackrel{ind}{\sim} N(\theta,1) \quad \text{and} \quad \theta \sim La(0,1)$$

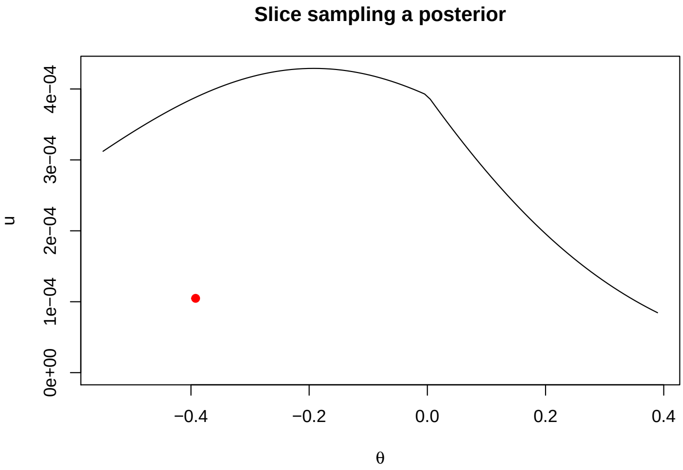
then

$$p(\theta|y) \propto \left[\prod_{i=1}^n N(y_i;\theta,1)\right] La(\theta;0,1)$$

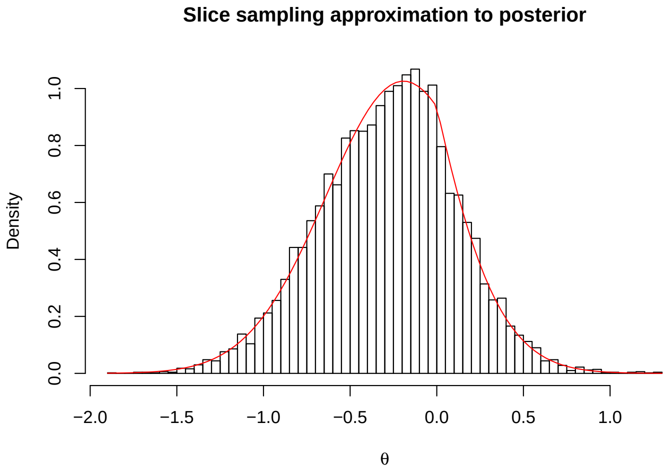
```
n = 5
y = rnorm(n,.2)
f = Vectorize(function(theta, y.=y) exp(sum(dnorm(y., theta, log=TRUE)) + dexp(abs(theta), log=TRUE)))

# Function to numerically find endpoints
A = function(u, xx, f.=f) {
  left_endpoint  = uniroot(function(x) f.(x) - u, c(-10^10, xx))
  right_endpoint = uniroot(function(x) f.(x) - u, c( 10^10, xx))
  c(left_endpoint$root, right_endpoint$root)
}
```

```
res = slice(20, mean(y), f, A)
```

Slice sampling using numerically calculated endpoints



Histogram of draws



An alternative augmentation

Suppose

$$Y_i \stackrel{ind}{\sim} N(\theta, 1) \quad \text{and} \quad \theta \sim La(0, 1)$$

but now, we will use the augmentation

$$p(u, \theta) \propto p(\theta)\mathrm{I}(0 < u < p(y|\theta))$$

The full conditional distributions are now

1. $u|\theta, y \sim Unif(0, p(y|\theta))$ and
2. $\theta|u, y \sim p(\theta)\mathrm{I}(u < p(y|\theta))$.

Sampling $\theta|u, y$

Now we need to sample from

$$p(\theta)\mathrm{I}(u < p(y|\theta)).$$

If $p(\theta)$ is unimodal, then this is equivalent to

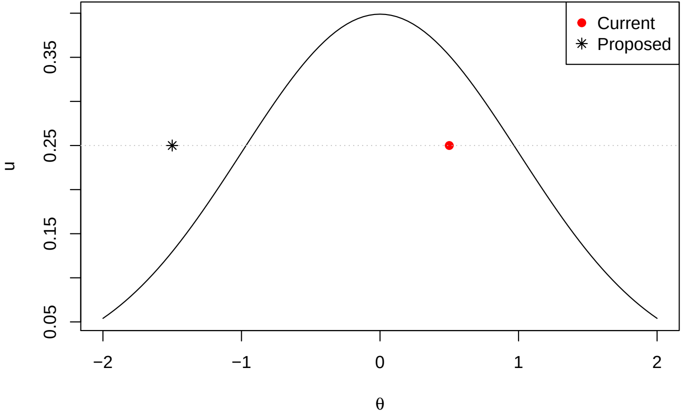
$$p(\theta)\mathrm{I}(\theta_L(u) < \theta < \theta_U(u))$$

for some bounds $\theta_L(u)$ and $\theta_U(u)$ which depend on u.

One way to learn these is to sample from $p(\theta)$ and update the bounds, e.g. if $\theta^{(i-1)}$ is our current value in the chain, we know $u < p(y|\theta^{(i-1)})$ or, equivalently, $\theta_L(u) < \theta^{(i-1)} < \theta_U(u)$. Letting $u^{(i)}$ be the current value for the auxiliary variable and setting $\theta_L(u^{(i)})$ [$\theta_U(u^{(i)})$] to the lower [upper] bound of the support for θ, we can

1. Sample $\theta^* \sim p(\theta)\mathrm{I}(\theta_L(u^{(i)}) < \theta < \theta_U(u^{(i)}))$.
2. Set $\theta^{(i)} = \theta^*$ if $u^{(i)} < p(y|\theta^*)$, otherwise
    a. set $\theta_L(u^{(i)}) = \theta^*$ if $\theta^* < \theta^{(i-1)}$ or
    b. set $\theta_U(u^{(i)}) = \theta^*$ if $\theta^* > \theta^{(i-1)}$ and
    c. return to Step 1

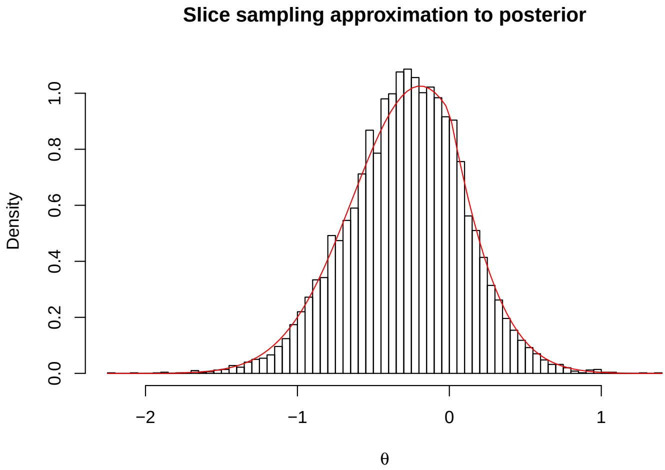
Learning the endpoints



R code

```
slice2 = function(n, init_theta, like, qprior) {
    u = theta = rep(NA, n)
    theta[1] = init_theta
    u[1] = runif(1, 0, like(theta[1]))

    for (i in 2:n) {
        u[i] = runif(1, 0, like(theta[i - 1]))
        success = FALSE
        endpoints = 0:1
        while (!success) {

            # Inverse CDF
            up = runif(1, endpoints[1], endpoints[2])
            theta[i] = qprior(up)

            if (u[i] < like(theta[i])) {
                success = TRUE
            } else {
                # Updated endpoints when proposed value is rejected
                if (theta[i] > theta[i - 1])
                  endpoints[2] = up
                if (theta[i] < theta[i - 1])
                  endpoints[1] = up
            }
        }

    }
    return(list(theta = theta, u = u))
}
```

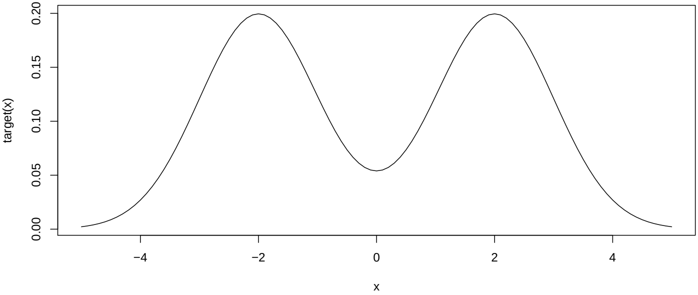
Histogram


Slice sampling approximation to posterior
Density
0.0
0.2
0.4
0.6
0.8
1.0
−2
−1
0
1
θ

Bimodal target distributions

Consider the posterior

$$p(\theta|y) = \frac{1}{2}N(\theta; -2, 1) + \frac{1}{2}N(\theta; 2, 1)$$



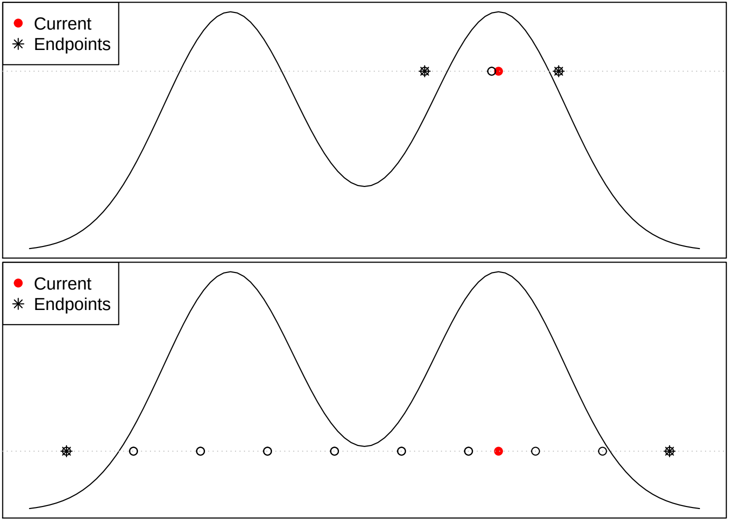
Stepping-out slice sampler

To sample from $\theta|u, y$, let

- $\theta^{(i-1)}$ be the current draw for θ
- $u^{(1)}$ be the current draw for the auxiliary variable u
- w be a tuning parameter that you choose

Perform the following

1. Randomly place an interval $(\theta_L(u^{(i)}), \theta_U(u^{(i)}))$ of length w around the current value $\theta^{(i-1)}$.
2. Step the endpoints of this interval out in increments of w until $u^{(i)} > p(\theta_L(u^{(i)})|y)$ and $u^{(i)} > p(\theta_B(u^{(i)})|y)$.
3. Sample $\theta^* \sim Unif(\theta_L(u^{(i)}), \theta_L(u^{(i)}))$.
4. If $u^{(i)} < p(\theta^*|y)$, then set $\theta^{(i)} = \theta^*$, otherwise
   a. set $\theta_L(u^{(i)}) = \theta^*$ if $\theta^* < \theta^{(i-1)}$ or
   b. set $\theta_U(u^{(i)}) = \theta^*$ if $\theta^* > \theta^{(i-1)}$ and
   c. return to Step 3.

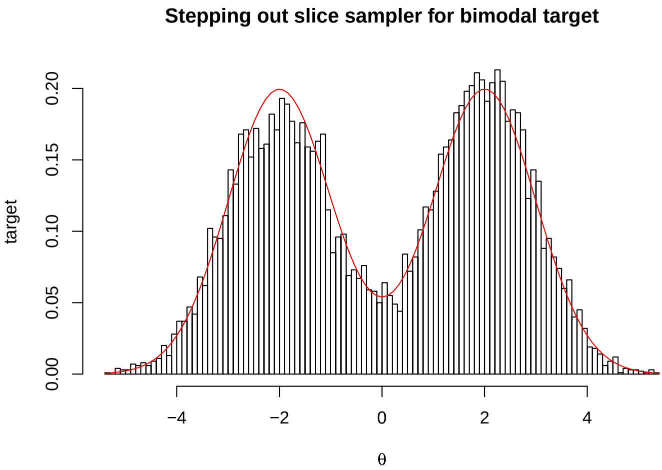

Current
Endpoints
Current
Endpoints

```
create_interval = function(theta, u, target, w, max_steps) {
  L = theta - runif(1,0,w)
  R = L + w

  # Step out
  J = floor(max_steps * runif(1))
  K = (max_steps - 1) - J
  while ((u < target(L)) & J > 0) {
    L = L - w
    J = J - 1
  }
  while ((u < target(R)) & K > 0) {
    R = R + w
    K = K - 1
  }

  return(list(L=L,R=R))
}

shrink_and_sample = function(theta, u, target, int) {
  L = int$L; R = int$R

  repeat {
    theta_prop = runif(1, L, R)
    if (u < target(theta_prop))
      return(theta_prop)

    # shrink
    if (theta_prop > theta)
      R = theta_prop
    if (theta_prop < theta)
      L = theta_prop
  }
}
```

```
slice = function(n, init_theta, target, w, max_steps) {
  u = theta = rep(NA, n)
  theta[1] = init_theta

  for (i in 2:n) {
    u[i] = runif(1, 0, target(theta[i - 1]))
    theta[i] = shrink_and_sample(theta = theta[i-1],
                                 u = u[i],
                                 target = target,
                                 int = create_interval(theta = theta[i-1],
                                                       u = u[i],
                                                       target = target,
                                                       w = w,
                                                       max_steps = max_steps))
  }
  return(data.frame(theta = theta, u = u))
}
```

Sampling from mixture of normals

```
res = slice(n = 1e4, init_theta=0, target=target, w=1, max_steps=10)
```



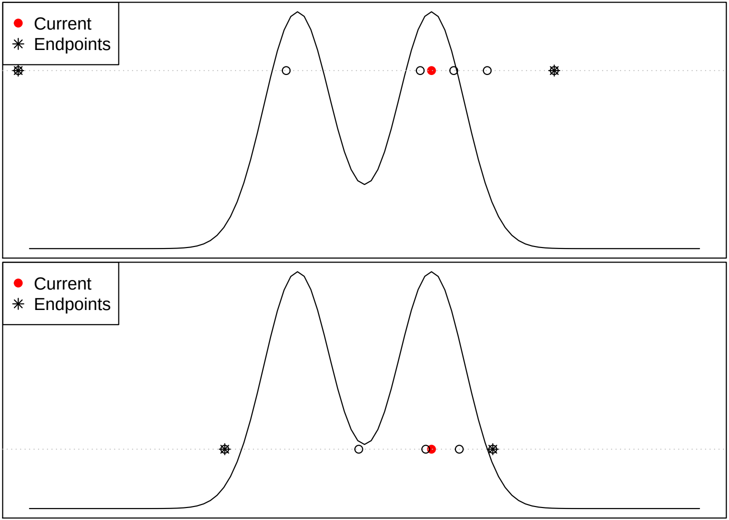
Doubling slice sampler

To sample from $\theta|u, y$, let

- $\theta^{(i-1)}$ be the current draw for θ
- $u^{(1)}$ be the current draw for the auxiliary variable u
- w be a tuning parameter that you choose

Perform the following

1. Randomly place an interval $(\theta_L(u^{(i)}), \theta_U(u^{(i)}))$ of length w around the current value $\theta^{(i-1)}$.
2. Randomly double the size of the interval to either the left or right until $u^{(i)} > p(\theta_L(u^{(i)})|y)$ and $u^{(i)} > p(\theta_B(u^{(i)})|y)$.
3. Sample $\theta^* \sim Unif(\theta_L(u^{(i)}), \theta_L(u^{(i)}))$.
4. If $u^{(i)} < p(\theta^*|y)$ and a reversibility criterion is satisfied, then set $\theta^{(i)} = \theta^*$, otherwise
   a. set $\theta_L(u^{(i)}) = \theta^*$ if $\theta^* < \theta^{(i-1)}$ or
   b. set $\theta_U(u^{(i)}) = \theta^*$ if $\theta^* > \theta^{(i-1)}$ and
   c. return to Step 3.


Current
Endpoints
Current
Endpoints

Reversibility criterion

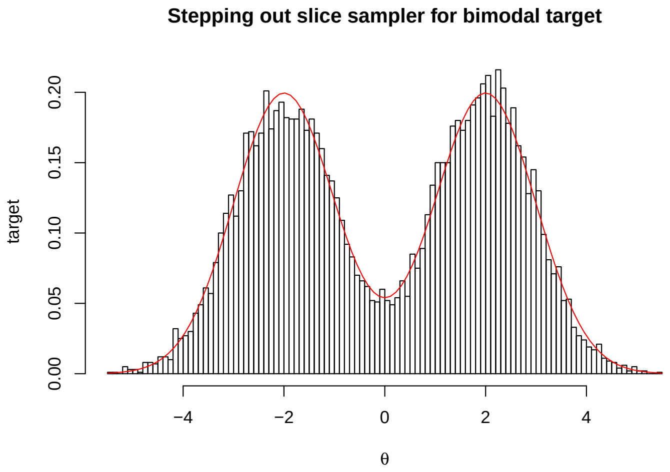
This procedure works backward through the intervals that the doubling procedure would pass through to arrive at [the doubled interval] when starting from the new point, checking that none of [the intermediate intervals] has both ends outside the slice, which would lead to earlier termination of the doubling procedure.

```
accept = function(theta0, theta1, L, R, u, w) {
    D = FALSE
    while (R - L > 1.1 * w) {
        M = (L + R)/2
        if ((theta0 < M & theta1 >= M) | (theta0 >= M & theta1 < M))
            D = TRUE
        if (theta1 < M) {
            R = M
        } else {
            L = M
        }
        if (D & u >= target(L) & u >= target(R)) {
            return(FALSE)
        }
    }
    return(TRUE)
}
```

```r
slice = function(n, init_theta, target, w, max_doubling) {
    u = theta = rep(NA, n)
    theta[1] = init_theta

    for (i in 2:n) {
        u[i] = runif(1, 0, target(theta[i - 1]))
        L = theta[i - 1] - runif(1, 0, w)
        R = L + w

        # Step out
        K = max_doubling
        while ((u[i] < target(L) | u[i] < target(R)) & K > 0) {
            if (runif(1) < 0.5) {
                L = L - (R - L)
            } else {
                R = R + (R - L)
            }
            K = K - 1
        }

        # Sample and shrink
        repeat {
            theta[i] = runif(1, L, R)
            if (u[i] < target(theta[i]) & accept(theta[i - 1], theta[i], L, R, u[i], w))
                break

            # shrink
            if (theta[i] > theta[i - 1])
                R = theta[i]
            if (theta[i] < theta[i - 1])
                L = theta[i]
        }
    }
    return(list(theta = theta, u = u))
}
```

Doubling slice sampler for bimodal target

```
res = slice(n=1e4, init_theta=0, target=target, w=1, max_doubling=10)
```



Multivariate slice sampling

Suppose, we are interested in sampling from

$$p(\theta_1, \theta_2|y) = \int_0^{p(\theta_1,\theta_2|y)} 1\, du$$

- Treat each variable independently, i.e.
  1. $u|\theta_1, \theta_2, y \sim Unif(0, p(\theta_1, \theta_2|y))$
  2. $\theta_1|u, \theta_2, y \sim Unif(u < p(\theta_1, \theta_2|y))$
  3. $\theta_2|u, \theta_1, y \sim Unif(u < p(\theta_1, \theta_2|y))$
  - Use overrelaxation to avoid random walk behavior
- Hyperrectangle slice sampling
  - Replace interval constructed from w with a hyperrectangle W placed randomly over the slice
  - Shrink as points are rejected
- Reflective slice sampling
  - Candidate samples are kept within the bounds by reflecting the direction of sampling when the boundary is hit