MH General

Tianwei Yue

2018

Multivariate Gaussian

Define Functions

```
Mu        = c(3, 6)
Sigma     = matrix(c(2, .5, .5, 1), nrow=2, ncol=2)
Sigma.inv = solve(Sigma)
Params    = list('mu'=Mu, 'sigma'=Sigma, 'sigma.inv'=Sigma.inv)

# Calculates the log-likelihood
loglike <- function(x, params=Params){
  mu        = params$mu
  sigma.inv = params$sigma.inv

  return(as.numeric(- t((x - mu)) %*% sigma.inv %*% (x - mu)/2) )
}

# Calculate the gradient of the log-likelihood
loglike.grad <- function(x, params=Params){
  mu        = params$mu
  sigma.inv = params$sigma.inv

  return(as.numeric(sigma.inv %*% (x - mu)))
}
```

Parameters Settings

```
eval  = eigen(Sigma)$values
evec  = eigen(Sigma)$vectors
L = 1/min(eval)
m = 1/max(eval)
```

MRW

```
#MRW sampler. Note that this proposal is symmetric.
mrw <- function(N = 1e5, sigma)
{
  chain.mrw <- matrix(0, nrow = N, ncol = 2)
  accept <- 0
  chain.mrw[1,] <- c(0,0)
  for(i in 2:N)
  {
    prop <- rnorm(2, mean = chain.mrw[i-1, ], sd = sigma)
    log.ratio <- loglike(prop) - loglike(chain.mrw[i-1, ])
```

```r
    if(log(runif(1)) < log.ratio)
    {
      chain.mrw[i, ] <- prop
      accept <- accept+1
    }else{
      chain.mrw[i, ] <- chain.mrw[i-1, ]
    }
  }
  return(list("chain" = chain.mrw, "accept" = accept/N))
}
```

MALA

```r
# Calculates the log gradient of the density
proplike <- function(x, y, sigma)
{
  grad <- loglike.grad(y) #Sigma.inv %*% (y - mu)
  mu.m <- y + sigma^2 * grad/2
  return(as.numeric(- t((x - mu.m)) %*% (x - mu.m)/(2*sigma^2) ) )
}
# Mala sampler
mala <- function(N = 1e5, sigma)
{
  chain.mala <- matrix(0, nrow = N, ncol = 2)
  accept <- 0
  # Starting value is the origin
  chain.mala[1,] <- c(0,0)
  for(i in 2:N)
  {
    grad <- loglike.grad(chain.mala[i-1, ]) #Sigma.inv %*% (chain.mala[i-1, ] - mu)
    mu.m <- chain.mala[i-1, ] + sigma^2 * grad/2
    prop <- rnorm(2, mean = mu.m, sd = sigma)
    log.ratio <- loglike(prop) + proplike(chain.mala[i-1, ], prop, sigma) -
                 loglike(chain.mala[i-1, ]) - proplike(prop, chain.mala[i-1, ], sigma)
    log.ratio <- min(0, log.ratio)
    if(log(runif(1)) < log.ratio)
    {
     chain.mala[i, ] <- prop
     accept <- accept+1
    }else{
      chain.mala[i, ] <- chain.mala[i-1, ]
    }
  }
  return(list("chain" = chain.mala, "accept" = accept/N))
}
```

Compare Results

```r
out.mrw <- mrw(N = 1e4*3, sigma = 1)
out.mrw$accept
```

```
## [1] 0.5905333
```

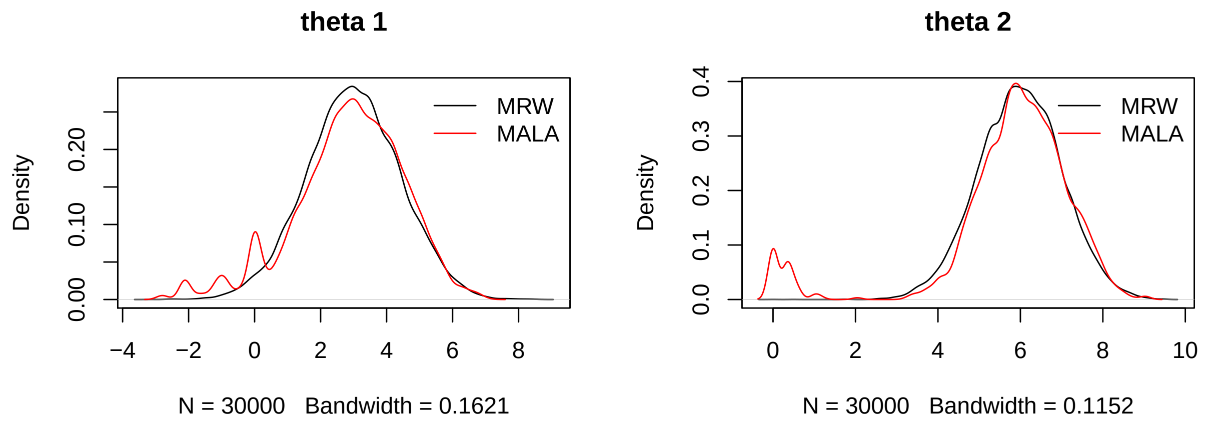
```r
out.mala <- mala(N = 1e4*3, sigma = 1)
out.mala$accept
```

```
## [1] 0.2813667
```

```r
par(mfrow = c(1,2))

plot(density(out.mrw$chain[,1]), ylab = "Density", main = "theta 1")
lines(density(out.mala$chain[,1]), col = "red")
legend("topright", col = c("black", "red"), lty = c(1,1), bty = "n",
       legend = c("MRW", "MALA"))

plot(density(out.mrw$chain[,2]), ylab = "Density", main = "theta 2")
lines(density(out.mala$chain[,2]), col = "red")
legend("topright", col = c("black", "red"), lty = c(1,1), bty = "n",
       legend = c("MRW", "MALA"))
```



```r
# Calibrated sigma to get close to optimal rate
out.mrw <- mrw(N = 1e4*3, sigma = 2.5)
out.mrw$accept
```

```
## [1] 0.2588333
```

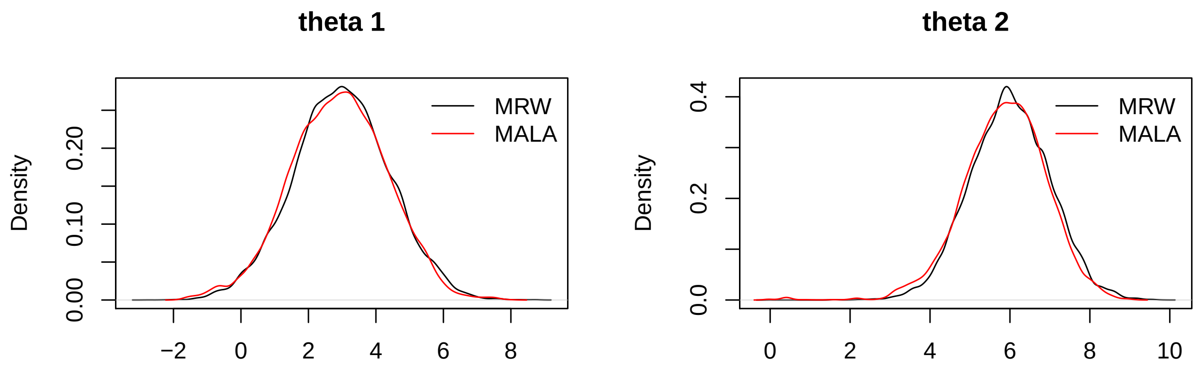
```r
out.mala <- mala(N = 1e4*3, sigma = .5)
out.mala$accept
```

```
## [1] 0.5810333
```

```r
par(mfrow = c(1,2))

plot(density(out.mrw$chain[,1]), xlab = "", ylab = "Density", main = "theta 1")
lines(density(out.mala$chain[,1]), col = "red")
legend("topright", col = c("black", "red"), lty = c(1,1), bty = "n",
       legend = c("MRW", "MALA"))

plot(density(out.mrw$chain[,2]), xlab = "", ylab = "Density", main = "theta 2")
lines(density(out.mala$chain[,2]), col = "red")
legend("topright", col = c("black", "red"), lty = c(1,1), bty = "n",
       legend = c("MRW", "MALA"))
```

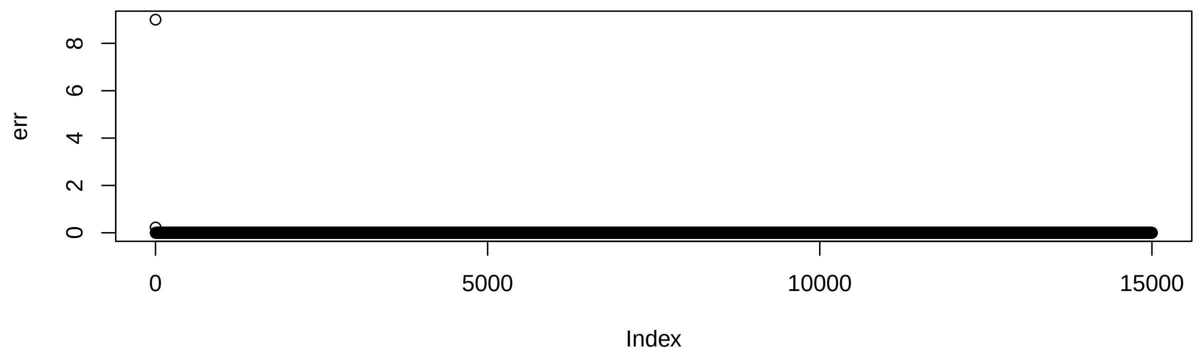


```r
#Calculate L1 err of mean estimation
library(base) #colMeans()
chain = out.mala$chain

err  = numeric(dim(chain)[1]/2)
ii   = 0

# for (i in c(seq(1,1000), seq(1000, dim(chain)[1], length.out = 100)))
for (i in seq(1, dim(chain)[1], length.out=2)){
  ii      = ii+1
  err[ii] = norm(data.matrix(
                 colMeans(matrix(chain[1:i,], ncol=dim(chain)[-1]))-Mu), type="1")
}

plot(err)
```


err
Index