MAST30027: Modern Applied Statistics

Week 10 Lab Sheet

Metropolis-Hastings Algorithm

Recall that $\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$ and $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$, iff $\mathbf{X}$ has joint density

$$f_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{x}) = \frac{1}{2\pi|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

where $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.

1. Write an R function that evaluates the density of a bivariate normal distribution. The function should take as input the point $\mathbf{x}$, the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$.

   You will find the functions `solve` and `det` useful.

   Solution:

```
> dbinorm <- function(x, mu, Si) {
+   # x and mu are vectors length 2 and Si a 2x2 matric
+   # returns the density at x of a bivariate normal mean mu var Si
+   exp(-t(x - mu)%*%solve(Si, x - mu)/2)/2/pi/sqrt(det(Si))
+ }
```

   You can check that it is working by noting that `dbinorm(c(1,1), c(0,0), matrix(c(1,0,0,1),2,2))` gives the same answer as `dnorm(1)^2`.

2. Write a program in R that uses the Metropolis-Hastings algorithm to generate a sample of size $n = 1000$ from the $N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}\right)$ distribution. Use the symmetric random walk proposal distribution $N\left(\mathbf{x}, \sigma^2 I\right)$ with $\sigma = 2.5$.

   Use $\mathbf{X}(0) = \begin{pmatrix} 6 \\ -6 \end{pmatrix}$ as your initial state. Report the proportion of accepted values.

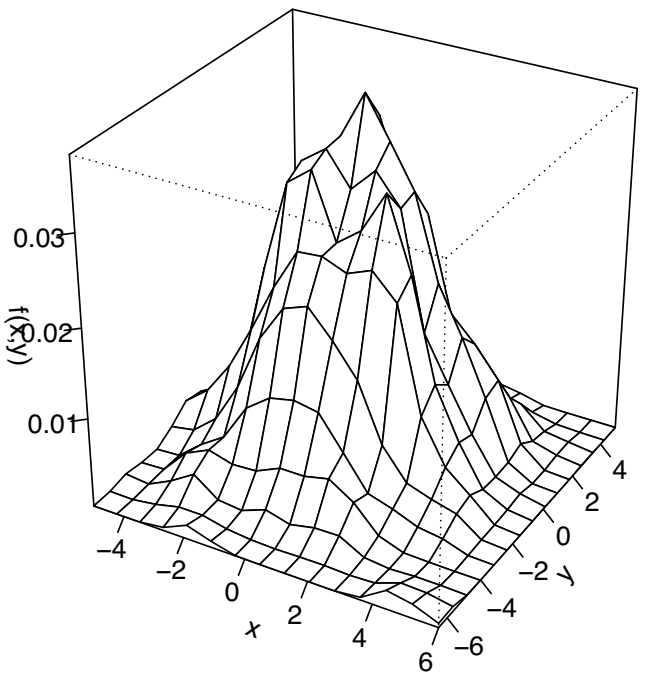
   Solution:

```
> # Metropolis-Hastings simulation of a bivariate normal
> # inputs
> mu <- c(0, 0)                       # mean
> Si <- matrix(c(4, 1, 1, 4), 2, 2) # variance
> iterations <- 1000      # sample size
> startvalue <- c(6, -6)  # initial value
> sd <- 2.5               # std-dev for proposal chain
> # main loop
> chain <- matrix(nrow = iterations+1, ncol = 2)
> chain[1,] <- startvalue
> accepted <- 0 # counts num accepted proposals
> for (i in 1:iterations){
+   proposal <- rnorm(2, chain[i,], sd)
+   prob <- dbinorm(proposal, mu, Si)/dbinorm(chain[i,], mu, Si)
+   if (is.nan(prob)) prob <- 0
+   if (runif(1) < prob) {
+     chain[i+1,] <- proposal
+     accepted <- accepted + 1
+   } else {
+     chain[i+1,] <- chain[i,]
+   }
```

```
+ }
> # acceptance rate
> accepted/iterations

[1] 0.457

> # 2D density plot
> library(MASS)
> chain_density <- kde2d(chain[,1], chain[,2], n = 15)
> persp(chain_density, phi = 30, theta = 30, d = 5,
+       xlab = "x", ylab = "y", zlab = "f(x,y)",
+       ticktype = "detailed")
```



The acceptance rate was 46% (this will vary a little every time you run the program).

To check that the output looks ok, I have plotted (a kernel density estimate of) the joint density. Easier than plotting a joint density would be to plot histograms/densities of the marginal samples, using `hist` or `density`.

3. Let $\mathbf{X}(n)$ be the n-th sample point. Plot $X_i(n)$ and the cumulative averages $\overline{X}_i(n) = n^{-1}\sum_{j=1}^{n} X_i(j)$, for $i = 1, 2$ (over iteration numbers). The cumulative averages should give a rough idea of how quickly the $\mathbf{X}(n)$ converge in distribution.

   Solution:

```
> # cumulative averages
> par(mfrow=c(2,1), mar=c(4,4,1,1))
> plot(chain[,1], xlab = "iteration", ylab = "x1")
> cumavg1 <- cumsum(chain[,1])/1:(iterations + 1)
```

```
> lines(1:(iterations + 1), cumavg1, col = "red")
> plot(chain[,2], xlab = "iteration", ylab = "x2")
> cumavg2 <- cumsum(chain[,2])/1:(iterations + 1)
> lines(1:(iterations + 1), cumavg2, col = "red")
>
>
```