

## Computer Practical: Metropolis-Hastings Model Answers

### Model code in Python

```

1 from __future__ import division
2 from scipy import *
3 from scipy import linalg
4 import pylab
5 import sys
6 sys.path.append("/home/ludger/lib/python2.6/site-packages/")
7 import statistics
8
9 #Task 1: Evaluate the density of a bivariate distribution at a single point
10 def density(x,mu,sigma):
11     inv_sigma=linalg.inv(sigma)
12     x_minus_mu=x-mu
13     return exp(-0.5*dot(dot(transpose(x_minus_mu),inv_sigma),x_minus_mu))/(2*pi*sqrt(
14 linalg.det(sigma) ) )
15
16 mu=array([0,0])
17 sigma=array([[4,1],[1,4]])
18
19 #Task 2: Metropolis Hastings
20
21 #Set the standard deviation of the proposal
22 sigma_prop=2.5
23 #Set the desired sample size
24 n=1000
25 #Set the starting values
26 x=array([0,0])
27 accepted_n=0
28 f=density(x,mu,sigma)
29 X1=[x[0]]
30 X2=[x[1]]
31
32 for i in xrange(1,n):
33     x_0=x[0]+random.normal(0,sigma_prop)
34     x_1=x[1]+random.normal(0,sigma_prop)
35     new_x=array([x_0,x_1])
36     new_f=density(new_x,mu,sigma)
37     if (random.random()<(new_f/f)):
38         accepted_n+=1
39         x=new_x
40         f=new_f
41     X1.append(x[0])
42     X2.append(x[1])
43
44 #Proportion of accepted values
45 print "The proportion of accepted values is", accepted_n/n
46
47 #Task 3: Diagnostics
48
49 #Sample plots for both values
50 pylab.figure(0)
51 pylab.plot(X1,'b')
52 pylab.title("Sample path of X_1")
53 pylab.figure(1)
54 pylab.plot(X2,'r')
55 pylab.title("Sample path of X_2")
56
57 #Cumulative averages

```

```

58 X1_cummean = cumsum( X1 ) / ( 1 + arange( len( X1 )))
59 X2_cummean = cumsum( X2 ) / ( 1 + arange( len( X1 )))
60 pylab.figure( 2 )
61 pylab.plot( X1_cummean,"b" )
62 pylab.title("Empirical_mean_of_X_1")
63
64 pylab.figure( 3 )
65 pylab.plot( X2_cummean,"r" )
66 pylab.title("Empirical_mean_of_X_2")
67 pylab.show()
68
69 #Autocorrelation
70 X1_sd = sqrt( var( X1 ) )
71 X2_sd = sqrt( var( X2 ) )
72 X1_autocorr = statistics.correlation( X1[1: ], X1[:-1])
73 X2_autocorr = statistics.correlation( X2[1: ], X2[:-1])
74 print "The autocorrelation of X_1 is", X1_autocorr
75 print "The autocorrelation of X_2 is", X2_autocorr
76
77 #Effective sample size
78 X1_ess = n * ( 1 - X1_autocorr ) / ( 1 + X1_autocorr )
79 X2_ess = n * ( 1 - X2_autocorr ) / ( 1 + X2_autocorr )
80 print "The effective sample size of X_1 is", X1_ess
81 print "The effective sample size of X_2 is", X2_ess
82
83 #Task 4: Repeat with sigma_prop = 0.1, sigma_prop = 10
84
85 #Task 5: Repeat with sigma = array([[4,2.8],[2.8,4]])

```

## Model code in R

```
1 #Task 1
2 #Evaluates the density of a bivariate normal distribution at a single point
3 mvdnorm <- function(x, mu, sigma){
4     x.minus.mu <- x - mu
5         #subtract mu from x
6     exp.arg <- -0.5 * sum(x.minus.mu * solve(sigma, x.minus.mu))
7         #evaluates the expression inside exp(...)
8     return( 1 / (2 * pi * sqrt(det(sigma))) * exp(exp.arg) )
9 }
10
11 #Task 2
12 #Metropolis Hastings
13
14 #set the mean parameter
15 mu <- c(0,0)
16 #set the covarince parameter
17 sigma <- matrix(c(4,1,1,4), nrow=2)
18
19 #set the standard deviation of the proposal
20 sd.proposal <- 2.5
21 #set the desired sample size
22 n <- 1000
23 x <- matrix(nrow=n, ncol=2)
24
25 #set the starting value
26 cur.x <- c(0,0)
27 #evaluate the density at the starting value
28 cur.f <- mvdnorm(cur.x, mu, sigma)
29
30 n.accepted <- 0
31 for(i in 1:n){
32     new.x <- cur.x + sd.proposal * rnorm(2)
33     new.f <- mvdnorm(new.x, mu, sigma)
34     if(runif(1) < new.f/cur.f){
35         n.accepted <- n.accepted + 1
36         cur.x <- new.x
37         cur.f <- new.f
38     }
39     x[i,] <- cur.x
40 }
41
42 #proportion of accepted values
43 n.accepted/n
44
45 #Task 3
46 #look at sample plots of both variables
47 par(mfrow=c(2,1))
48 plot(x[,1], type="l", xlab="t", ylab="X_1", main="Sample_path_of_X_1")
49 plot(x[,2], type="l", xlab="t", ylab="X_2", main="Sample_path_of_X_2")
50
51 #compute cumulative averages
52 x1_cumulative <- rep(0, times=n)
53 x2_cumulative <- rep(0, times=n)
54 for(i in 1:n){
55     x1_cumulative[i] <- mean(x[1:i,1])
56     x2_cumulative[i] <- mean(x[1:i,2])
57 }
58
59 par(mfrow=c(2,1))
60 plot(x1_cumulative, type="l", xlab="t", ylab="mean_of_X_1", main="Empirical_mean_of_X_1")
61 plot(x2_cumulative, type="l", xlab="t", ylab="mean_of_X_2", main="Empirical_mean_of_X_2")
62
63 #correlations
64 cor.x1 <- cor(x[-1,1], x[-nrow(x),1])
65 cor.x2 <- cor(x[-1,2], x[-nrow(x),2])
66 c(cor.x1, cor.x2)
```

```
67 |
68 | #effective sample size
69 | ess.x1 <- n * (1-cor.x1) / (1+cor.x1)
70 | ess.x2 <- n * (1-cor.x2) / (1+cor.x2)
71 | c(ess.x1, ess.x2)
72 |
73 | #Task 4
74 | #Change the standard deviation of the proposal, and repeat.
75 | sd.proposal <- 0.1
76 | sd.proposal <- 10
77 |
78 | #Task 5
79 | #Now set the covarince parameter as follows, and repeat.
80 | sigma <- matrix(c(4,2.8,2.8,2), nrow=2)
```