Computer Practical: Gibbs Sampling

In this computer practical, you can use either Python or R. There is a list of useful Python and R functions at the end of this handout.

Theoretical background: bivariate Normal distribution

This computer practical focuses on sampling from the bivariate Normal distribution using the Gibbs sampling algorithm. The density of $\mathbf{X} = (X_1, X_2)' \sim \mathsf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution with $\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$ and $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$ is

$$f_{(\boldsymbol{\mu},\boldsymbol{\Sigma})}(x_1, x_2) = \frac{1}{2\pi|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right),$$

where $\mathbf{x} = (x_1, x_2)$.

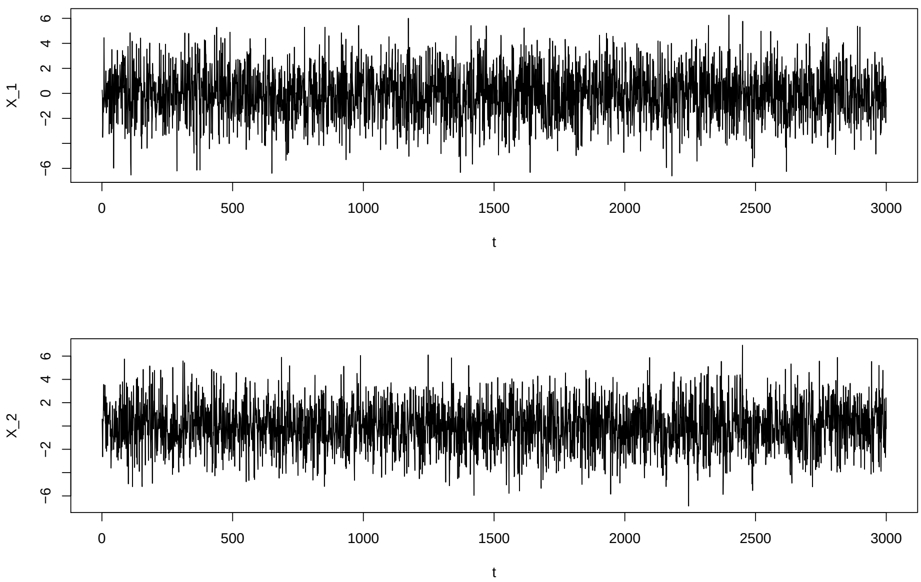
Sampling from the bivariate Normal distribution using the Gibbs sampler

We have seen in the lectures that, in order to sample from the joint distribution of (X_1, X_2) using the Gibbs sampler, we need to be able to sample from the full conditional distributions $X_1|X_2 = x_2$ and $X_2|X_1 = x_1$. In the case of the bivariate Normal distribution, the conditional distribution of X_1 given X_2 is

$$X_1|X_2 = x_2 \sim \mathsf{N}\left(\mu_1 + \frac{\sigma_{12}}{\sigma_2^2}(x_2 - \mu_2), \sigma_1^2 - \frac{\sigma_{12}^2}{\sigma_2^2}\right).$$

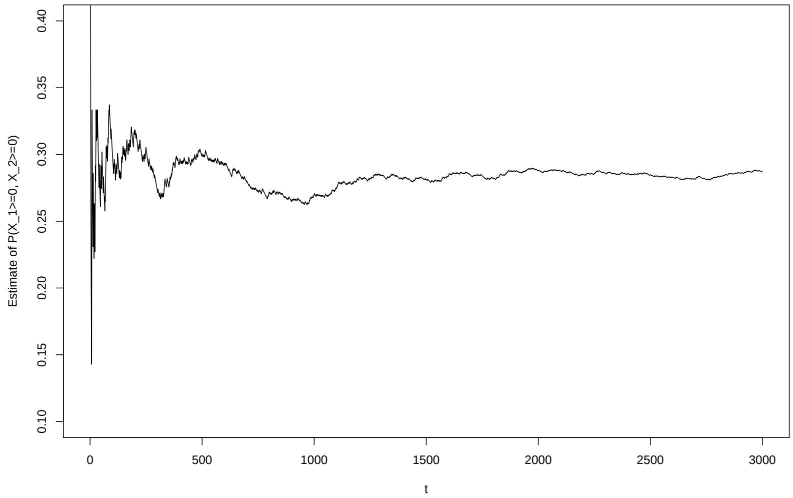
Task 1. Implement the Gibbs sampling algorithm to generate an approximate sample of size $n = 3000$ from the bivariate Normal distribution with mean $\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and covariance $\boldsymbol{\Sigma} = \begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}$.

Task 2. To assess how well the Gibbs sampler is mixing, plot the last 100 values, with subsequent values linked by a line, as well as trace plots of both variables. Your trace plots should look similar to the one below.



Task 3. We want to compute a Monte Carlo estimate of $\mathbb{P}(X_1 \geq 0, X_2 \geq 0)$. Modify your code from Task 1 to keep track of the estimate at each iteration; at iteration t, the estimate is $t^{-1}\sum_{i=1}^{t} \mathbb{I}(x_1^{(i)} \geq 0, x_2^{(i)} \geq 0)$, where $(x_1^{(i)}, x_2^{(i)})$ is the Gibbs

sampler draw at iteration i. Plot this estimate against the iteration number t. Your figure should be similar to the one below.



Task 4. Run the Gibbs sampler again to generate 3000 samples from the bivariate Normal distribution with $\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and covariance $\boldsymbol{\Sigma} = \begin{pmatrix} 4 & 2.8 \\ 2.8 & 2 \end{pmatrix}$. Compare the mixing properties of this Gibbs sampling algorithm to the previous one.

Useful Python functions

The following Python functions might be of interest:

Sampling from the univariate normal distribution The function `random.normal(`*mean*`,`*sd*`)` draws a sample of size 1 from the univariate $\mathsf{N}(mean, sd^2)$ distribution.

Sampling from the multivariate normal distribution The function `random.multivariate_normal(`*mu*`,`*Sigma*`,`*n*`)` draws a sample of size n from the multivariate $\mathsf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution.

```
from scipy import*
from pylab import plot, show, figure, axis, step

#Generate a sample of size 1000 using PYTHON
n = 1000
# we shall sample directly from the bivariate Normal distribution;

mu = [0,0]
# setting the mean parameter
cov = [[4,1],[1,4]]
# setting the covariance matrix
x,y = random.multivariate_normal(mu,cov,n).T

# plot the last 100 values as in Figure 4.1 on the notes.
step(x[900:],y[900:])
show()

# compute the estimate of P(X1 >= 0, X2 >= 0)
prob = zeros(n)
prob[0]= (x[0]>=0 and y[0]>=0)

for i in xrange(1,n):
   prob[i]= prob[i-1]+(x[i]>=0 and y[i]>=0)
y = prob/xrange(1,n+1)

# plotting the resulting probabilities.
plot(y)
axis([-200, n+200, 0.1, 0.4])
show()
```

Useful R functions

Note that it is possible to sample from a bivariate Normal distribution using having to resort to MCMC methods. If $\mathbf{Z} \sim \mathsf{N}(\mathbf{0}, \mathbf{I})$, then $\boldsymbol{\Sigma}^{1/2}\mathbf{Z} + \boldsymbol{\mu} \sim \mathsf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. A function for sampling from the bivariate Normal distribution is available in the package **MASS** in **R**.

The following R functions might be of interest:

Sampling from the univariate normal distribution The function `rnorm(`*n*`,`*mean*`,`*sd*`)` draws a sample of size n from the univariate $\mathsf{N}(mean, sd^2)$ distribution.

Sampling from the multivariate normal distribution The function `mvrnorm(`*n*`,`*mu*`,`*Sigma*`)` draws a sample of size n from the multivariate $\mathsf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution.

```
library(MASS)
#set the mean parameter
mu <- c(0,0)
#set the mean parameter
sigma <- matrix(c(4,1,1,4), nrow=2)

#generate a bivariate normal random variable by the above method
sigma.chol <- chol(sigma)
z <- rnorm(2)
y <- mu + t(sigma.chol) %*% z

#generate a sample of size 1000 using the R function
n <- 1000
```

```
#we sample directly from the bivariate Normal distribution;
#in the practical, sampling is done approximately by the Gibbs sampling algorithm.
x <- mvrnorm(n, mu, sigma)

#plot the last 100 values, with subsequent values linked by a line
plot(x[901:1000,], type='b', xlab="X_1", ylab="X_2")

#look at sample plots of both variables
par(mfrow=c(2,1))
plot(x[,1], type="l", xlab="t", ylab="X_1")
plot(x[,2], type="l", xlab="t", ylab="X_2")

#compute the estimate of P(X_1 >=0, X_2 >=0)
prop <- rep(0, times=n)
prop[1] <- as.numeric((x[1,1] >= 0) && (x[1,2] >= 0))
for(i in 2:n){
  prop[i] <- prop[i-1] + as.numeric((x[i,1] >= 0) && (x[i,2] >= 0))
}

prop <- prop/1:n
plot(prop, ylim=c(0.1,0.4), type='l', xlab="t", ylab="Estimate of P(X_1>=0, X_2>=0)")
```