Introduction to Monte-Carlo Methods

Rohan L. Fernando

May 2015

Mean and Variance of Truncated Normal

Suppose $Y \sim N(\mu_Y, V_Y)$.

The mean and variance of Y given truncation selection are:

$$E(Y|Y > t) = \mu_Y + V_Y^{1/2} i$$

where

$$i = \frac{f(s)}{p}$$

$f(s)$ is the standard normal density function

$$s = \frac{t - \mu_Y}{V_Y^{1/2}}$$

$$p = \Pr(Y > t)$$

$$Var(Y|Y > t) = V_Y[1 - i(i - s)]$$

Proof:

Start with mean and variance for a standard normal variable given truncation selection.

Let $Z \sim N(0, 1)$.

The density function of Z is:

$$f(z) = \sqrt{\frac{1}{2\pi}} e^{-\frac{1}{2}z^2}$$

The density function for Z given truncation selection is

$$f(z|z > s) = f(z)/p$$

From the definition of the mean:

$$
\begin{aligned}
E(Z|Z > s) &= \frac{1}{p}\int_s^\infty zf(z)dz \\
&= \frac{1}{p}[-f(z)]_s^\infty \\
&= \frac{f(s)}{p} \\
&= i
\end{aligned}
$$

because the first derivative of $f(z)$ with respect to z is:

$$
\begin{aligned}
\frac{d}{dz}f(z) &= \sqrt{\frac{1}{2\pi}}e^{-\frac{1}{2}z^2}(-z) \\
&= -zf(z)
\end{aligned}
$$

Now, to compute the variance of Z given selection, consider the following identity:

$$
\begin{aligned}
\frac{d}{dz}zf(z) &= f(z) + z\frac{d}{dz}f(z) \\
&= f(z) - z^2f(z)
\end{aligned}
$$

Integrating both sides from s to ∞ gives

$$
zf(z)]_s^\infty = \int_s^\infty f(z)dz - \int_s^\infty z^2f(z)dz
$$

Upon rearranging this gives:

$$
\begin{aligned}
\int_s^\infty z^2f(z)dz &= \int_s^\infty f(z)dz - zf(z)]_s^\infty \\
\frac{1}{p}\int_s^\infty z^2f(z)dz &= \frac{1}{p}\int_s^\infty f(z)dz + \frac{f(s)}{p}s \\
&= 1 + is
\end{aligned}
$$

So,

$$
\begin{aligned}
Var(Z|Z > s) &= E(Z^2|Z > s) - [E(Z|Z > s)]^2 \\
&= 1 + is - i^2 \\
&= 1 - i(i - s)
\end{aligned}
$$

Results for Y

Results for Y follow from the fact that

$$
\mu_Y + V_Y^{1/2}Z \sim N(\mu_Y, V_Y)
$$

So, let

$$
Y = \mu_Y + V_Y^{1/2}Z,
$$

Then, the condition

$$
Y > t
$$

is equivalent to

$$\begin{aligned}
\mu_Y + V_Y^{1/2} Z &> t \\
V_Y^{1/2} Z &> t - \mu_Y \\
Z &> \frac{t - \mu_Y}{V_Y^{1/2}} \\
Z &> s
\end{aligned}$$

Then,

$$\begin{aligned}
E(Y|Y > t) &= E(\mu_Y + V_Y^{1/2} Z|Z > s) \\
&= \mu_Y + V_Y^{1/2} i,
\end{aligned}$$

and

$$\begin{aligned}
Var(Y|Y > t) &= Var(\mu_Y + V_Y^{1/2} Z|Z > s) \\
&= V_Y[1 - i(i - s)]
\end{aligned}$$

Numerical Example

In [39]:
```
μ = 10
σ = 10
t = 15
s = (t-μ)/σ
d = Normal(0.0,1.0)
i = pdf(d,s)/(1-cdf(d,s))
meanTruncatedNormal = μ + σ*i
variTruncatedNormal = σ*σ*(1 - i*(i-s))
@printf "mean     = %8.2f  \n" meanTruncatedNormal
@printf "variance = %8.2f  \n" variTruncatedNormal
```

```
mean     =    21.41
variance =    26.85
```

Monte-Carlo Approach:

In [43]:
```
using Distributions
μ = 10
σ = 10
z = rand(Normal(μ,σ),10000);
```

In [56]:
```
mcmcMean = mean(z[z.>t])
mcmcVar = var(z[z.>t])
@printf "MC mean     = %8.2f  \n" mcmcMean
@printf "MC variance = %8.2f  \n" mcmcVar
```

```
MC mean     =    21.34
MC variance =    25.78
```

Bivariate Normal Example

Let $(Y) \sim N(\mu, \mathbf{V})$

$$\mu = \begin{bmatrix} 10 \\ 20 \end{bmatrix}, \mathbf{V} = \begin{bmatrix} 100 & 50 \\ 50 & 200 \end{bmatrix}$$

In [54]:
```
μ = [10.0;20.0]
V = [100.0 50.0
    50.0  200.0]
d = MvNormal(μ,V)
XY = rand(d,10000)'
```

Out[54]:
```
10000x2 Array{Float64,2}:
 10.3117    41.2371
  8.49604   30.121
  1.49591    5.04669
  2.0137    21.2858
  8.12043    9.99512
 17.9018    16.9568
  1.01726   20.0321
 -8.29162   40.2454
 14.6496    45.1535
 13.9381    12.9118
 -0.612875  24.1609
 20.5875    15.1366
 16.2409    25.9275
  ⋮
  3.98896    3.67185
 13.8927    24.0219
  3.93784   11.8521
  3.83364    4.41762
 20.7947    37.1139
  9.11036   15.7678
  4.45919   32.2166
 19.5114    21.9018
 12.777     29.3537
 18.1348    11.6092
  0.640994  14.6436
  3.39195   27.4398
```

In [111]:
```
sel = XY[:,1].>10
xxy= [XY sel]
```

Out[111]:
```
10000x3 Array{Float64,2}:
 10.3117    41.2371   1.0
  8.49604   30.121    0.0
  1.49591    5.04669  0.0
  2.0137    21.2858   0.0
  8.12043    9.99512  0.0
 17.9018    16.9568   1.0
  1.01726   20.0321   0.0
 -8.29162   40.2454   0.0
 14.6496    45.1535   1.0
 13.9381    12.9118   1.0
 -0.612875  24.1609   0.0
 20.5875    15.1366   1.0
 16.2409    25.9275   1.0
  ⋮
  3.98896    3.67185  0.0
 13.8927    24.0219   1.0
  3.93784   11.8521   0.0
  3.83364    4.41762  0.0
 20.7947    37.1139   1.0
  9.11036   15.7678   0.0
  4.45919   32.2166   0.0
 19.5114    21.9018   1.0
 12.777     29.3537   1.0
 18.1348    11.6092   1.0
  0.640994  14.6436   0.0
  3.39195   27.4398   0.0
```

In [115]:
```
(xxy[:,1][xxy[:,3].==1])
```

Out[115]:
```
18.03854352069298
```

In [59]:
```
selY = XY[sel,2]
```

Out[59]:
```
5026-element Array{Float64,1}:
 41.2371
 16.9568
 45.1535
 12.9118
 15.1366
 25.9275
 17.4284
 20.6601
 44.2587
  7.21451
 26.9525
 29.502
 41.1791
  ⋮
 41.4734
 20.1128
 33.6962
 17.7152
 16.6372
 48.6728
 27.0785
 24.0219
 37.1139
 21.9018
 29.3537
 11.6092
```

In [60]:
```
mean(selY[selY.>30])
```

Out[60]:
```
38.95540792778809
```

In [61]:
```
var(selY[selY.>30])
```

Out[61]:
```
52.61527300087836
```

Markov Chain Monte-Carlo Methods

- Often no closed form for $f(\boldsymbol{\theta}|\boldsymbol{y})$
- Further, even if computing $f(\boldsymbol{\theta}|\boldsymbol{y})$ is feasible, obtaining $f(\theta_i|\boldsymbol{y})$ would require integrating over many dimensions
- Thus, in many situations, inferences are made using the empirical posterior constructed by drawing samples from $f(\boldsymbol{\theta}|\boldsymbol{y})$
- Gibbs sampler is widely used for drawing samples from posteriors

Gibbs Sampler

- Want to draw samples from $f(x_1, x_2, \ldots, x_n)$
- Even though it may be possible to compute $f(x_1, x_2, \ldots, x_n)$, it is difficult to draw samples directly from $f(x_1, x_2, \ldots, x_n)$
- Gibbs:
  - Get valid a starting point $\mathbf{x}^0$
  - Draw sample $\mathbf{x}^t$ as:

$$
\begin{array}{ccc}
x_1^t & \text{from} & f(x_1 | x_2^{t-1}, x_3^{t-1}, \ldots, x_n^{t-1}) \\
x_2^t & \text{from} & f(x_2 | x_1^{t}, x_3^{t-1}, \ldots, x_n^{t-1}) \\
x_3^t & \text{from} & f(x_3 | x_1^{t}, x_2^{t}, \ldots, x_n^{t-1}) \\
\vdots & & \vdots \\
x_n^t & \text{from} & f(x_n | x_1^{t}, x_2^{t}, \ldots, x_{n-1}^{t})
\end{array}
$$

- The sequence $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n$ is a Markov chain with stationary distribution $f(x_1, x_2, \ldots, x_n)$

Making Inferences from Markov Chain

Can show that samples obtained from a Markov chain can be used to draw inferences from $f(x_1, x_2, \ldots, x_n)$ provided the chain is:

- Irreducible: can move from any state i to any other state j
- Positive recurrent: return time to any state has finite expectation
- *Markov Chains*, J. R. Norris (1997)

Bivariate Normal Example

Let $f(\mathbf{x})$ be a bivariate normal density with means

$$\mu' = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

and covariance matrix

$$\mathbf{V} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2.0 \end{bmatrix}$$

Suppose we do not know how to draw samples from $f(\mathbf{x})$, but know how to draw samples from $f(x_i | x_j)$, which is univariate normal with mean:

$$\mu_{i.j} = \mu_i + \frac{v_{ij}}{v_{jj}}(x_j - \mu_j)$$

and variance

$$v_{i.j} = v_{ii} - \frac{v_{ij}^2}{v_{jj}}$$

In [125]:
```
m = fill(0,2)
nSamples = 2000
m = [1.0, 2.0]
v = [1.0 0.5; 0.5 2.0]
y   = fill(0.0,2)
sum = fill(0.0,2)
s12 = sqrt( v[1,1] - v[1,2]*v[1,2]/v[2,2])
s21 = sqrt(v[2,2] -  v[1,2]*v[1,2]/v[1,1])
m1 = 0
m2 = 0;
for (iter in 1:nSamples)
    m12 = m[1] + v[1,2]/v[2,2]*(y[2] - m[2])
    m21 = m[2] + v[1,2]/v[1,1]*(y[1] - m[1])
    y[1] = rand(Normal(m12,s12),1)[1]
    y[2] = rand(Normal(m21,s21),1)[1]
    sum += y
    mean = sum/iter
    if iter%100 == 0
        @printf "%10d %8.2f %8.2f \n" iter mean[1]  mean[2]
    end
end
```

```
       100     1.09     2.21
       200     1.06     2.16
       300     1.06     2.16
       400     1.05     2.12
       500     1.03     2.11
       600     1.01     2.10
       700     1.00     2.09
       800     1.01     2.09
       900     1.00     2.08
      1000     1.02     2.10
      1100     1.00     2.09
      1200     1.01     2.08
      1300     1.01     2.08
      1400     1.02     2.08
      1500     1.03     2.10
      1600     1.02     2.08
      1700     1.02     2.08
      1800     1.02     2.08
      1900     1.03     2.07
      2000     1.02     2.06
```

Metropolis-Hastings Algorithm

- Sometimes may not be able to draw samples directly from $f(x_i|\mathbf{x}_{i\_})$
- Convergence of the Gibbs sampler may be too slow
- Metropolis-Hastings (MH) for sampling from $f(\mathbf{x})$:

- a candidate sample, y, is drawn from a proposal distribution $q(y|x^{t-1})$

$$x^t = \begin{cases} y & \text{with probability } \alpha \\ x^{t-1} & \text{with probability } 1-\alpha \end{cases}$$

$$\alpha = \min(1, \frac{f(y)q(x^{t-1}|y)}{f(x^{t-1})q(y|x^{t-1})})$$

- The samples from MH is a Markov chain with stationary distribution $f(x)$

Bivariate Normal Example

In [127]:
```
nSamples = 10000
m = [1.0, 2.0]
v = [1.0 0.5; 0.5 2.0]
vi = inv(v)
y   = fill(0.0,2)
sum = fill(0.0,2)

m1 = 0
m2 = 0
xx = 0
y1 = 0
delta = 1.0
min1 = -delta*sqrt(v[1,1])
max1 = +delta*sqrt(v[1,1])
min2 = -delta*sqrt(v[2,2])
max2 = +delta*sqrt(v[2,2])
z = y-m
denOld = exp(-0.5*z'*vi*z)
d1 = Uniform(min1,max1)
d2 = Uniform(min2,max2)
ynew = fill(0.0,2);
for (iter in 1:nSamples)
    ynew[1] = y[1] + rand(d1,1)[1]
    ynew[2] = y[2] + rand(d2,1)[1]
   denNew = exp(-0.5*(ynew-m)'*vi*(ynew-m));
   alpha = denNew/denOld;
    u = rand()
    if (u < alpha[1])
        y = copy(ynew)
        denOld = exp(-0.5*(y-m)'*vi*(y-m))
    end
    sum += y
    mean = sum/iter
    if iter%1000 == 0
        @printf "%10d %8.2f %8.2f \n" iter mean[1]  mean[2]
    end
end
```

```
      1000     1.04     1.93
      2000     1.10     1.91
      3000     1.13     1.91
      4000     1.13     1.98
      5000     1.05     1.96
      6000     1.03     1.94
      7000     1.03     1.96
      8000     1.03     1.96
      9000     1.04     1.96
     10000     1.06     1.97
```