STA 250 Lecture 18 Notes

Chuan Qin

December 2, 2013

1. **Notes on Homework 4**:
   - Only need to write one kernel which obtains samples from a truncated normal distribution for both Q1 and Q2.
   - Your code for the kernel in Q1 has to be robust/error free or else it might cause problem when used for Q2.
   - Probit MCMC:

$$Y_i|Z_i = 1_{\{Z_i>0\}}$$
$$Z_i|\beta \sim N(x_i^T\beta, 1)$$
$$\beta_i \sim N(\beta_0, \Sigma_0)$$
$$\Rightarrow P(\beta|Z,Y) \sim \text{Normal}$$
$$\Rightarrow P(Z_i|\beta_i, Y_i) \sim \text{Truncated Normal}$$

   MCMC:

```
for (iter in (niter + burnin)){
    if (use GPU){
    z = rtruncnormGPU(...)   # CUDA and kernel in (...)
    } else{
    z = rtruncnormCPU(...)  # regular R/Python in (...)
    }
    beta = rmvnorm(...)
}
```

2. **C/C++**
   - C is a very fast complied language.
   - Data types need to be explicitly defined.

- Vectors/matrices are typically implemented using "pointers".
- Pointers point to memory locations, from which you can look up values at those locations.
- About homework:
  - "__global__ void" tells the compiler that this function is a kernel and it does not return any value.
  - The samples are written into the memory locations pointed by the input arguments, defined as pointers.

3. **Truncated Normal Sampling**

If

$$x \sim N(\mu, \sigma^2) 1_{\{x \in (a,b)\}},$$

then

$$x \sim \text{Truncated} - \text{Normal}(\mu, \sigma^2; (a, b)).$$

The simplest sampling method to implement is rejection sampling, which repeatedly sample from $N(\mu, \sigma^2)$ until the value falls in the interval (a, b):

```
accepted = False
numtries = 0
# Specify maxtimes as the maximum number of attempts
while (! accepted and numtries < maxtimes ){
    numtimes = numtimes + 1
    x = rnorm(mu, sigma)
    if (x>=a and x<=b){
    accepted = True
    }
}
```

However, this method is quite inefficient when a is several standard deviations larger than μ or b is several standard deviations smaller than μ. Instead, it is more advisable to use the following *rejection sampling* algorithm for sampling from tail truncated normal distribution.

Rejection Sampling

To sample from a distribution with p.d.f. $f(x)$, if we can find another distribution with p.d.f. $g(x)$ such that

$$f(x) \leq M g(x), \qquad \forall x,$$

then we can use g to sample from f as follows:

(i) Sample a value x^* from $g(x)$.

(ii) Sample $U \sim U[0,1]$.

(iii) If

$$U \leq \frac{f(x^*)}{Mg(x^*)},$$

then accept x^*. Otherwise return to (i).

Remark:

It is clear that the bigger the gap between $f(x)$ and $Mg(x)$, the lower the accepting probability. Therefore, ideally, we should choose $g(x)$ so that $f(x)$ and $Mg(x)$ are "close" for all x.

Robert (2009) proposed the following algorithm for sampling from a one-sided truncated normal distribution.

<u>Rejection Sampling One-sided Truncated Normal</u>

To sample from $X \sim N(0, 1; (\mu^-, \infty))$,

(i) Generate

$$z = \mu^- + \text{Expo}(\alpha).$$

(ii) Compute

$$\Psi(z) = \begin{cases} \exp\left(-\frac{(\alpha - z)^2}{2}\right), & \text{if } \mu^- < \alpha \\ \exp\left(-\frac{(\alpha - z)^2}{2} - \frac{(\mu^- - \alpha)^2}{2}\right), & \text{if } \mu^- \geq \alpha \end{cases}$$

(iii) If $U[0,1] < \Psi(z)$, accept; else try again.

Optimal choice of α:

$$\alpha^* = \frac{\mu^- + \sqrt{(\mu^-)^2 + 4}}{2}.$$

Remark:

In homework we need to sample from $N(\mu, \sigma^2; (a, \infty))$. To do this we can sample a value x from

$$N\left(0, 1; \left(\frac{a - \mu}{\sigma}, \infty\right)\right)$$

and then use $\mu + \sigma x$ as our sample.