# STA 250: Lecture 18 Notes

*Ozan Sonmez*

**12/2/2013**

---

**Final Projects**

-Extra Office Hours (maybe Wed or Thu)
-HW4 is due Friday (Dec 6th) at 11:59pm. HW4 and final projects will be handed in via email.

**HW4-comments**

-PyCUDA is more mature than RCUDA so it will be interesting to compare the performance of PyCUDA and RCUDA
-PyCUDA currently provides more informative error explanation than RCUDA. When using RCUDA, you will need to look up the error codes in `/usr/local/cuda/include/cuda.h`. this will likely be improved in future RCUDA releases.
-For problem2, do Gibbs Sampler: successivley sampling $\beta$ and the $z$'s

**Probit MCMC**

$$\begin{aligned}
&Y_i|Z_i = I_{\{z_i>0\}} \\
&Z_i|\beta \sim N(X_i'\beta, 1) \\
&\beta \sim N(\beta_0, \Sigma_0) \\
&\Rightarrow \quad P(\beta|Z, Y) \sim \text{Normal} \\
&\Rightarrow \quad P(Z_i|\beta, Y_i) \sim \text{Truncated Normal}
\end{aligned}$$

**MCMC**

```
for (iter in 1:(niter+burnin))
      if (use GPU){
          z=rtruncnormGPU(...)  #(...) is cudo/kernel
       }else{
          z=rtruncnormCPU(...)  #(...) is Regular R/Python
       }
    beta=rmvnorm(...)
```

Note: In practice you may want to avoid having `rtruncnorm` be a separate function and instead directly call `.cuda` (if using `RCUDA`) to avoid unnecessary memory copies).
Question1: Write code to obtain samples from a truncated normal
Question2: Problem1 needs to be coded robustly to do Problem 2

**C/C++**

- C is a fast, compiled language
- You need to explicitly tell C about all data types
- Data types need to be explicitly defined
- Vectors/matrices do not have a natural data type in `C`
- Vector/matrices are typically implemented using 'pointers'
- Pointers point to memory locations, from which you can look up values or those memory locations.

$$x \quad \rightarrow \quad [x_{[0]}|x_{[1]}|x_{[2]}|x_{[3]}|...|x_{[n-1]}]$$

HW kernel

-'`__global__`': Tell the compiler this function is a kernel (i.e., visible to both host and device)

-'`void`' : The kernel does not return anything (in R `dnorm(...)  return(exp(-x*x/2))` will return to the value ). In C it does not return to anything (there is no return value). Instead, the value is written into the memory locations pointed to by the input arguments.

**More code**

-Need to figure out what thread you are in, i.e.,

```
threadIdx.x, threadIdx.y, threadIdx.z, blockIdx.x,...
```

-Map these into simple index: idx

-Check whether your index (idx) is $< n$ (since we launch more than than we need, make sure to check that for the HW )

-Initialize Random Number Generator (RNG) (for HW initialize within each thread)

-then add integers

```
// To sample TN(mu,sigma^2,a,b):
int rng_a // RNG seed constant
int rng_b // RNG seed constant
int rng_c // RNG seed constant
curandState rng;
curan_init(rng_a+idx*rng_b,rng_c,0,&rng);
//Then sample the truncated normal
//mu for this index is mu[idx]
//sigma for this index is sigma[idx]
//a for this index is a[idx]
//b for this index is b[idx]
//X_~ Truncated Normal (mu.i,sigma.i, [a.i,b.i])
// Rejection sampling: while(...){...}
//Sample N(mu,sigma^2)
x[idx]=mu[idx]+sigma[idx]*curand_normal(&rng);
//to obtain ~ U[0,1] use: curand_uniform(&rng)
```

$$Z_i \sim TN(\mu_i, 1, [a_i, b_i])$$

-Add `maxtries` argument for safety

-Handle the corner cases via rejection sampling described in Robert's paper.

**Truncated Normal Sampling**

if $X \sim N(\mu, \sigma^2)I_{\{x\in(a,b)\}}$, then $X \sim TruncatedNormal(\mu, \sigma^2, a, b)$

```
accepted=False
while (!accepted and numtries<maxtries){
             numtries=numtries+1
             X=rnorm(mu,sigma)
    If (X>a and X <b){
          accepted=True
    }
}
return(X)
```

if it still doesn't work , need to use different rejection/acceptance sampling (refer to Robert's paper)

**Rejection Sampling**

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

$$f(x) \leq Mg(x)$$

then we can use g to sample from fas follows:

- (1) Sample a value $x_*$ from $g(x)$
- (2) Sample $U \sim U(0,1)$
- (3) If $U \leq \frac{f(x_*)}{Mg(x_*)}$ then accept $x_*$ else return (1).

-We need to scale $f$ such that $Mg(x) > f(x)$ for all $x$.
- Ideally $f(x)$ and $Mg(x)$ should be "close" to have high acceptance rate.

**From Robert 2009**

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

- (1) Generate $Z = \mu^- + Expo(\alpha)$
- (2) Compute

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

- (3) if $U[0,1] < \Psi(z)$ accept, else try again

**Optimal $\alpha$ is**

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

We need $X \sim N(\mu, \sigma^2, a, \infty)$. Let $Z \sim N(0, 1, \mu^-, \infty)$. What is the distribution of $Y = cZ + k$?

$$cZ \sim N(0, c^2, c\mu^-, \infty)$$
$$cZ + k \sim N(k, c^2, c\mu^- + k, \infty)$$

then figure out what to choose for $k, c, \mu^-$. Suppose we have $N(\mu, \sigma^2, a, \infty)$ then select $k, c, \mu^-$ such that

$$k = \mu, \quad c^2 = \sigma^2, \quad a = c\mu^- + k, \quad \mu^- = \frac{a - \mu}{\sigma}$$

For right truncation, we can use left truncation because it is symmetric. (use the same algorithm and take the negative at the end).

HW4 part-g: Question will be changed for $a$ to be $-\infty$.(Only right truncation)
-Be careful with PyCUDA: make sure to define data types as `numpy` data types e.g., (`np.int32` or `np.float32`)

-Write the Kernel inside the `SourceModule` function (In `RCUDA` the kernel is written in a different file and compiled)
- See `example0` in the `Lecture_Code > GPU > PyCUDA` directory of the course GitHub repo
- `a b` live on the CPU
-It will be copied to GPU via `drv.In(a)` and `drv.In(b)`
-The result will be copied back to CPU via `drv.Out`. Input and output arguments can use `drv.InOut`.