STA 250/MTH 342 Intro to Mathematical Statistics
Lab Session 5 / February 09, 2015 / Handout

In this session we make three cases. First, we approximate some pdf by a discrete step function, and then construct a discrete distribution that approximates the original one from which we can draw samples easily. While there are many other efficient sampling algorithms (e.g., rejection sampling), the discretization method is easy to understand with still acceptable precision.

The second case is a simple application of the law of large numbers to evaluate complicated definite integrals by Monte Carlo integration.

In the third case, we study how to use **R** to find local and global maxima of a function. This is useful when deriving the maximizer of a likelihood function is not straightforward.

See: `https://stat.duke.edu/courses/Spring15/sta250/labs/` for links to source code and data. Submit lab solutions via email to: `sta250@stat.duke.edu`. Any plots should be included in postscript form as attachments. The email subject must be "STA250 ..." with "..." replaced by your name.

1: A Bayesian analysis.

Consider the model $X \sim \mathsf{Bi}(n, p)$, $p \in [0, 1]$. Suppose p modeled with the pdf $\xi(p) = e^p/(e-1)$, $p \in [0, 1]$. The posterior based on an observations x is $\xi(p \mid x) = \text{Const} \times p^x(1-p)^{n-x}e^p$. The constant, which is one over the integral $\int_0^1 p^x(1-p)^{n-x}e^p \, \mathrm{d}p$, is quite hard (yet still doable) to derive analytically.

We now try to generate a sequence of independent random variables from the distribution $\xi(p \mid x)$. Since p takes value only on the interval $(0, 1)$, we use the trick of dividing the set into small intervals $(0, \frac{1}{N}]$, $(\frac{1}{N}, \frac{2}{N}]$, $\cdots$, $(\frac{N-1}{N}, 1)$. Here the integer N is very large. Then it is easy to see that the probability that p falls into an interval $(\frac{i}{N}, \frac{i+1}{N}]$, is approximately proportional to $\xi(\frac{i+0.5}{N} \mid x)$.

For numerical simulation, we let $n = 7$, $x = 3$, and $N = 300$. First, we generate a vector $\mathbf{u}$ which is proportional to $\left\{\xi(\frac{i+0.5}{N} \mid x = 3)\right\}_{i=1}^{N}$.

```
n  <- 7; x = 3; N = 300;
xi <- function(t) t ^ x * (1 - t) ^ (n - x) * exp(t);
t  <- ((0 : (N - 1)) + 0.5) / N;
```

Then we generate a sequence of $m = 10,000$ random variables from a discrete distribution on the set $\left\{\frac{0.5}{N}, \frac{1.5}{N}, \cdots \frac{N-0.5}{N}\right\}$ with the probability vector proportional to $\mathbf{u}$.

```
s <- sample(t, size = 10000, replace = T, prob = xi(t));
```

Note that in **R**, the probability vector will be internally normalized to sum 1. Now we plot the histogram of the samples $\mathbf{s}$, and add the curve of the density function. See Figure 1. They match quite well. The code is in file `case1.R`.

```
x.curv <- seq(0, 1, len = 500);
y.curv <- xi(x.curv) / integrate(xi, 0, 1)$value;
hist(s, freq = F, xlim = c(0, 1), ylim = c(0, 2.5), breaks = 50);
lines(x.curv, y.curv, col = "red", lwd = 2);
```

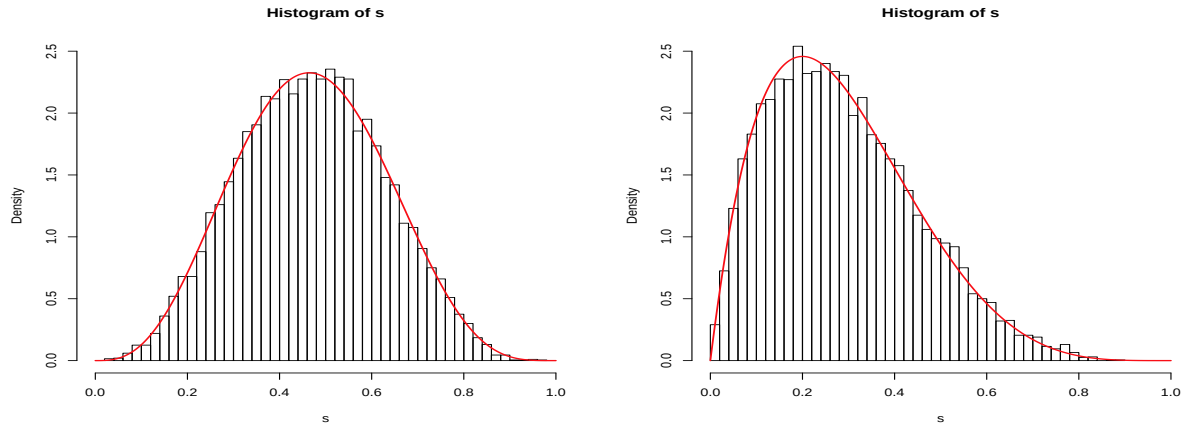

Figure 1: Left: the histogram of the 10,000 random samples generated from $\xi(p \mid x = 3)$. Right: the solution to task 1.

TASK 1 Please replace the distribution $\xi(p \mid x = 3)$ of p with $\mathsf{Be}(2, 5)$ and repeat the simulation above. Use and modify the code file `case1.R`. You should obtain a plot like the right one on Figure 1. Recall that $\mathsf{Be}(\alpha, \beta)$ has density function

$$f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad \text{for } 0 < x < 1.$$

2: Monte Carlo integration. Let's consider again the definite integral

$$\int_0^1 p^3(1-p)^4 e^p \, \mathrm{d}p.$$

Evaluating it will be extremely difficult. In fact, the integral equals $8742 - 3216e \approx 0.0056396757117909$, a horrible number. However, we may approximate it by the law of large numbers. Denote $f(p) = p^3(1-p)^4 e^p$.

Suppose $X \sim \text{Uniform}([0,1])$. We draw N independent random copies $\{X_i\}_{i=1}^N$ of X. We have

$$\frac{1}{N}\sum_{i=1}^{N} f(X_i) \approx \mathsf{E}[f(X)] = \int_0^1 p^3(1-p)^4 e^p \, \mathrm{d}p.$$

This method of approximating a definite integral is called Monte Carlo integration. Let's code it!

```
> f <- function(p) p ^ 3 * (1 - p) ^ 4 * exp(p);
> 0.0056396757117909 - mean(f(runif(1000)));
[1] -0.0001142902
> 0.0056396757117909 - mean(f(runif(10000)));
[1] 2.904246e-05
> 0.0056396757117909 - mean(f(runif(100000)));
[1] 1.496385e-05
```

Note that the factor $p^3(1-p)^4$ takes zero at the boundary of the integration interval. This somehow wastes the samples taken near the boundary. To make the simulation more efficient, we try to reduce the sampling density at the places the function has small values. Now suppose $X \sim \text{Beta}(4,5)$. Define $g(x) := \frac{\Gamma(4)\Gamma(5)}{\Gamma(9)} e^x = \frac{e^x}{280}$. Now draw N independent random copies $\{X_i\}_{i=1}^N$ of X to give

$$\frac{1}{N}\sum_{i=1}^{N} g(X_i) \approx \mathsf{E}[g(X)] = \int_0^1 \frac{\Gamma(4)\Gamma(5)}{\Gamma(9)} e^p \frac{\Gamma(9)}{\Gamma(4)\Gamma(5)} p^3(1-p)^4 \,\mathrm{d}p = \int_0^1 p^3(1-p)^4 e^p \,\mathrm{d}p.$$

Again we obtain a statistic which is close to the integral. We code it up.

```
> g <- function(x) exp(x) / 280;
> 0.0056396757117909 - mean(g(rbeta(shape1 = 4, shape2 = 5, n = 1000)));
[1] -3.103029e-05
> 0.0056396757117909 - mean(g(rbeta(shape1 = 4, shape2 = 5, n = 10000)));
[1] -6.104516e-06
> 0.0056396757117909 - mean(g(rbeta(shape1 = 4, shape2 = 5, n = 100000)));
[1] -7.749469e-08
```

Comparing with the previous simulation, we find that the new method is much more accurate!

TASK 2 Use the method with uniform random variables to evaluate

$$\int_0^1 \exp(\sin(x)) \,\mathrm{d}x.$$

Note that the integral equals $1.631869608418051348137161723744681\cdots$.

3: Optimization. **R** has some functions and packages for optimization. We can use them to find the maximum of, e.g., a likelihood function. Note that in some cases **R** is only able to achieve a local maximum. One needs to plot or use other tools to judge whether it is a global maximum.

Now suppose $\{X_1, \cdots, X_n\}$ are independent observations from a gamma distribution (with $\beta = 1$),

$$f(x \mid \alpha) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}.$$

Then the log likelihood function is

$$\log L(\alpha) = -N \log \Gamma(\alpha) + (\alpha - 1)\sum_{i=1}^{N} \log X_i - \sum_{i=1}^{N} X_i.$$

This attains its maximum at the same value of α that maximizes:

$$g(\alpha) := \frac{1}{N}\left(\log L(\alpha) + \sum_{i=1}^{N} \log X_i + \sum_{i=1}^{N} X_i\right) = \alpha\left(\frac{1}{N}\sum_{i=1}^{N} \log X_i\right) - \log \Gamma(\alpha).$$

Let's do some simulation. We first generate a sample, then use it to infer the parameter α.

```
> X <- rgamma(10000, shape = pi);
> g <- function(a) { a * mean(log(X)) - log(gamma(a)) };
> optimize(g, interval = c(0, 10), maximum = T);
$maximum
[1] 3.132265

$objective
[1] 2.231332
```

Here in the code, the function `optimize()` searches an interval to maximize/minimize a function. So it cannot optimize multivariate functions. To optimize multivariate functions, try the functions `optim()` or `nlm()`, or the package `nloptr`. The function `optimize()` asks us to give an interval which is reasonable in the numerical world. We see that in the above example the estimate is quite close to the truth, $\alpha = \pi = 3.14159$. On the other hand, we have learned this model in Question 8 of Homework 3, where we found (for $\beta = 1$) that

$$\psi(\alpha) = \frac{\Gamma'(\widehat{\alpha})}{\Gamma(\widehat{\alpha})} = \frac{1}{N}\sum_{i=1}^{N} \log X_i.$$

Here the function ψ is referred to as digamma function, and **R** has its implementation. So we can check if the above optimizer really returns a precise solution.

```
> digamma(optimize(g, interval = c(0, 10), maximum = T)$maximum);
[1] 0.9737164
> mean(log(X));
[1] 0.9737133
```

We see that the result is precise. However, one may improve the precision by setting the desired accuracy.

```
> digamma(optimize(g, interval = c(0,10), maximum=T)$maximum) - mean(log(X));
[1] 3.136381e-06
> digamma(optimize(g, interval = c(0,10), maximum=T, tol = 1e-10)$maximum) - mean(log(X));
[1] -8.285729e-09
```

TASK 3 Recall that for the Poisson distribution with mean λ, the MLE for λ is $\widehat{\lambda} = \bar{X}$, and the log likelihood function is

$$\log L(\lambda) = -N\lambda + \log\lambda \sum_{i=1}^{N} X_i - \sum_{i=1}^{N} \log(X_i!).$$

Let $N = 10,000$ and $\lambda = 3.5$, use the **R** command `rpois(`N`,`λ`)` to generate data, write the likelihood function (you may scale and shift it by multiplying and adding some constant) and use the above `optimize()` function to find $\widehat{\lambda}$ (set the searching interval yourself!). Compare it with $\bar{X}$.

$\sim\sim$END$\sim\sim$