STATS 406 Fall 2016: Lab 10

1 Monte Carlo Estimate, Error and C.I.

First we will do a brief review of **basic Monte Carlo integration**, and the quantities **Monte Carlo Estimate, Monte Carlo Error and Confidence Interval**. The idea of **Monte Carlo integration** is to transform an integration problem into evaluating an expectation by sample mean.

Question: Let $h : \mathbb{R}^d \to \mathbb{R}$ be a function, and $\pi(x)$ be a density on $\mathbb{R}^d$. We want to evaluate the integral $\mathbb{E}\left(h(X)\right) = \int h(x)\pi(x)dx$.

Monte Carlo Estimate: $\pi_n\left(h\right) \triangleq \frac{1}{n}\sum_{k=1}^{n} h(X_k)$ is called *Monte Carlo estimate* of $\int h(x)\pi(x)dx$, where $\vec{X} = \{X_1, X_2, \ldots, X_n\}$ is a random sample drawn from the density $\pi(x)$ in **Basic Monte Carlo**. $\pi_n\left(h\right)$ in R is simply $mean(h(\vec{X}))$

Monte Carlo Error: $\frac{s_n(h)}{\sqrt{n}} = \sqrt{s_n\left(h\right)^2/n}$ is called the *Monte Carlo error* of the estimate $\pi_n\left(h\right)$, where $s_n\left(h\right)^2/n = \frac{1}{n-1}\sum_{k=1}^{n}\left(h(X_k) - \pi_n(h)\right)^2/n$, which in R is simply $var(h(\vec{X}))/n$.
Confidence Interval: $\pi_n(h) \pm z_{\alpha/2}\frac{s_n(h)}{\sqrt{n}}$ gives the $(1-\alpha)$ level approximate confidence interval for the mean $\mathbb{E}\left(h(X)\right)$ that we wanted to estimate.

Example: Consider $h(x) = sin(xcos(x))$. Compute the integral

$$\int_{-\infty}^{\infty} h(x)dx.$$

Idea: consider $\int_{-\infty}^{\infty} h(x)dx = \int_{-\infty}^{\infty} \frac{h(x)}{\pi(x)} \cdot \pi(x)dx = \mathbb{E}\left(\frac{h(X)}{\pi(X)}\right)$, where $\pi(x)$ is the standard normal density.

Algorithm:
1. draw n samples $\{x_1, x_2, \ldots, x_n\}$ from standard normal distribution $\pi(x)$ by $rnorm(n)$;

2. compute the sample mean ($mean()$) π_n of the sample $\left\{\frac{h(x_1)}{\pi(x_1)}, \frac{h(x_n)}{\pi(x_n)}, \ldots, \frac{h(x_n)}{\pi(x_n)}\right\}$, which is the Monte Carlo estimate of the integral.

3. compute the Monte Carlo error $\frac{s_n}{\sqrt{n}} = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}\left(h(x_k)/\pi(x_k) - \pi_n\right)^2/n}$, where one can evaluate $\frac{1}{n-1}\sum_{k=1}^{n}\left(h(x_k)/\pi(x_k) - \pi_n\right)^2$ by the sample variance ($var()$) of the sample $\left\{\frac{h(x_1)}{\pi(x_1)}, \frac{h(x_n)}{\pi(x_n)}, \ldots, \frac{h(x_n)}{\pi(x_n)}\right\}$.

4. compute the 95% confidence interval $\pi_n \pm q_{0.975}\frac{s_n}{\sqrt{n}}$, where $q_{0.975} = z_{0.025}$ denotes the 0.975 quantile of standard normal.

Implementation:

```
integral2 <- function(n)
{
  x <- rnorm(n)
  integral <- mean(sin(x*cos(x)) / dnorm(x))
  mc_error <- sqrt(var(sin(x*cos(x)) / dnorm(x)) / n)
  z = qnorm(0.975)
  CIl = integral - z * mc_error;
  CIr = integral + z * mc_error;
  output <- list(integral, mc_error, CIl, CIr)
  return(output)
}

print(integral2(100000))
```

2 Importance Sampling

Importance Sampling is another Monte Carlo integration technique. Built on the Basic Monte Carlo which transforms an integral to an expectation estimated by appropriate sample mean, **Importance Sampling** is trying to transfer from one expectation representation to another for the same integral, since the original representation involves difficulty or impossibility of effectively generating a random sample.

Question(same): Let $h : \mathbb{R}^d \to \mathbb{R}$ be a function, and $\pi(x)$ be a density on $\mathbb{R}^d$. We want to evaluate the mean $\mathbb{E}\left(h(X)\right) = \int h(x)\pi(x)dx$.

Basic Monte Carlo: draw a random sample $\{x_1, x_2, \ldots, x_n\}$ from $\pi(x)$ and esimate $\int h(x)\pi(x)dx$ by the sample mean $\pi_n(h) = \frac{1}{n}\sum\limits_{k=1}^{n} h(x_k)$.

<u>Problem with Basic Monte Carlo</u>: it can be difficult or inefficient, or even impossible to draw a random sample from $\pi(x)$ that falls into the integration region, we will further explain this by two examples later, for now, **how to solve this problem? We turn to importance sampling**:

Importance Sampling with fully specified target density $\pi(x)$: consider

$$\int h(x)\pi(x)dx = \int h(x)\frac{\pi(x)}{g(x)}g(x)dx = \int h(x)\omega(x)g(x)dx$$

, where $\omega(x) = \frac{\pi(x)}{g(x)}$, now draw a random sample $\{x_1, x_2, \ldots, x_n\}$ from $g(x)$ and esimate $\int h(x)\pi(x)dx$ by the sample mean $\pi_{n,IS}(h) = \frac{1}{n}\sum\limits_{k=1}^{n} h(x_k)\omega(x_k)$.

Importance Sampling Error and C.I.: Similiar to Basic Monte Carlo, the Importance sampling error is given by $\frac{s_{n,IS}(h)}{\sqrt{n}} = \sqrt{s_{n,IS}\left(h\right)^2/n}$, where $s_{n,IS}\left(h\right)^2/n = \frac{1}{n-1}\sum\limits_{k=1}^{n}\left(h(x_k)\omega(x_k) - \pi_{n,IS}(h)\right)^2/n$ and the $1-\alpha$ level confidence interval is given by $\pi_{n,IS}(h) \pm z_{\alpha/2}\frac{s_{n,IS}(h)}{n}$.

Importance Sampling with unknown constant in $\pi(x) = \tilde{\pi}(x)/C$: notice that since $\pi(x) = \tilde{\pi}(x)/C$, and $\int \pi(x)dx = 1$ for any density function $\pi(x)$, we have

$$C = \int \tilde{\pi}(x)dx = \int \frac{\tilde{\pi}(x)}{g(x)}g(x)dx = \int \tilde{\omega}(x)g(x)dx$$

, where $\tilde{\omega}(x) = \frac{\tilde{\pi}(x)}{g(x)}$, and consider

$$\int h(x)\pi(x)dx = \frac{1}{C}\int h(x)\frac{\tilde{\pi}(x)}{g(x)}g(x)dx = \frac{\int h(x)\tilde{\omega}(x)g(x)dx}{C} = \frac{\int h(x)\tilde{\omega}(x)g(x)dx}{\int \tilde{\omega}(x)g(x)dx} = \frac{\mathbb{E}(h(X)\tilde{\omega}(X))}{\mathbb{E}(\tilde{\omega}(X))}$$

, where X follows the distribution $g(x)$. Now draw a random sample $\{x_1, x_2, \ldots, x_n\}$ from $g(x)$ and esimate $\int h(x)\pi(x)dx$ by the ratio of two sample means $\pi_{n,IS}(h) = \frac{\frac{1}{n}\sum_{k=1}^{n} h(x_k)\tilde{\omega}(x_k)}{\frac{1}{n}\sum_{k=1}^{n} \tilde{\omega}(x_k)} = \frac{\sum_{k=1}^{n} h(x_k)\tilde{\omega}(x_k)}{\sum_{k=1}^{n} \tilde{\omega}(x_k)}$. This sampling method is called "weighted average Importance Sampling".

Practice rule of thumb This method is only considered reliable when the weights are not too variable. As a rule of thumb, when

$$CV = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}\left(\frac{\tilde{w}(x_k)}{\bar{w}} - 1\right)^2} < 5, \quad \text{where } \bar{\omega} = \frac{1}{n}\sum_{k=1}^{n}\tilde{\omega}(x_k)$$

the method is reasonable. CV in R is simply $sqrt(var(\tilde{\omega}(\vec{x})/\bar{\omega}))$, where $\bar{\omega} = mean(\tilde{\omega}(\vec{x}))$, and why is this true?

Example 1: IS with fully specified target density $\pi(x)$:

We will look at a case where importance sampling provides a reduction in the variance of an integral approximation. Consider the function $h(x) = 10\exp(-2|x-5|)$. Suppose that we want to calculate $E(h(X))$, where $X \sim Uniform(0, 10)$. That is, we want to calculate the integral

$$\int_0^{10} \exp(-2|x-5|) \cdot dx = \int_0^{10} 10\exp(-2|x-5|) \cdot \frac{1}{10} dx$$

The true value of this integral is about 1. The simple way to do this is to use the basic monte carlo approach and generate X_i from the Uniform(0,10) density and look at the sample mean of $10h(X_i)$ (notice this is equivalent to importance sampling with importance function $w(x) = 1$):

```
X <- runif(100000,0,10)
Y <- 10*exp(-2*abs(X-5))
c( mean(Y), var(Y) )
[1] 0.9919611 3.9529963
```

The function h in this case is peaked at 5, and decays quickly elsewhere, therefore, under the uniform distribution, many of the points are contributing very little to this expectation. Something more like a gaussian function (ce^{-x^2}) with a peak at 5 and small variance, say, 1, would provide greater precision. We can re-write the integral as

$$\int_0^{10} 10 * \exp(-2|x-5|) \frac{1/10}{\frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2}} \frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2} dx$$

$$= \int_0^{10} 10\exp(-2|x-5|)\frac{1}{10}\sqrt{2\pi}e^{(x-5)^2/2}\frac{1}{\sqrt{2\pi}}e^{-(x-5)^2/2}dx$$

That is, $E(h(X)w(X))$, where $X \sim N(5,1)$, and $w(x) = \frac{\sqrt{2\pi}}{10}e^{(x-5)^2/2}$ is the importance function in this case.

```
X=rnorm(1e5,mean=5,sd=1)
hX = 10*exp(-2*abs(X-5))
Y=hX*dunif(X,0,10)/dnorm(X,mean=5,sd=1)
c( mean(Y), var(Y) )
[1] 0.9999271 0.3577506
```

Notice that the integral calculation is still correct, but with a variance this is approximately 1/10 of the simple monte carlo integral approximation. This is one case where importance sampling provided a substantial improvement in precision.

Comprehensive Example: IS with unknown constant in $\pi(x) = \tilde{\pi}(x)/C$:

Compute the mean of the distribution with density $\pi(x) \propto \tilde{\pi}(x) = \frac{e^{-x}}{1+x}$ for $x > 0$. Use exponential density for different choices of rate parameter λ as your trial density. Find the one which minimizes the CV (rule of thumb).

Solution: we want to compute

$$\int_0^{\infty} x\pi(x)dx \tag{2.0.1}$$

, but we only know the density $\pi(x)$ up to a normalizing constant since $\pi(x) \propto \tilde{\pi}(x) = \frac{e^{-x}}{1+x}$, put in math words: $\pi(x)/\tilde{\pi}(x) = 1/C$ where the constant C is unknown. Since $\pi(x)$ is a density, we know that $\int_0^{\infty}\pi(x)dx = 1$. So $\frac{1}{C}\int_0^{\infty}\tilde{\pi}(x)dx = 1$, that is we have an expression for C that $C = \int_0^{\infty}\tilde{\pi}(x)dx$. Note that the above is a standard analysis for normalizing constant in general.

Using the idea of "weighted average Importance Sampling", choosing $g(x) = \lambda e^{-\lambda x}$, we esimate (2.0.1) by the following algorithm:

Algorithm:

1. draw a sample $\vec{x} = \{x_k\}_{k=1:n\_mc}$ of size n_mc from $g(x) = \lambda e^{-\lambda x}$ by $rexp(n\_mc)$;

2. compute the sample mean $(S1)$ of $\{h(x_k)\tilde{\omega}(x_k)\}_{k=1:n\_mc}$, where $h(x) = x$, $\tilde{\omega}(x) = \frac{\tilde{\omega}(x)}{g(x)} = \frac{e^{(\lambda-1)x}}{\lambda(1+x)}$;

3. compute the sample mean $(S2)$ of $\{\tilde{\omega}(x_k)\}_{k=1:n\_mc}$,

4. Take the ratio of $\frac{S1}{S2}$, this is the importance sampling estimate of (2.0.1),

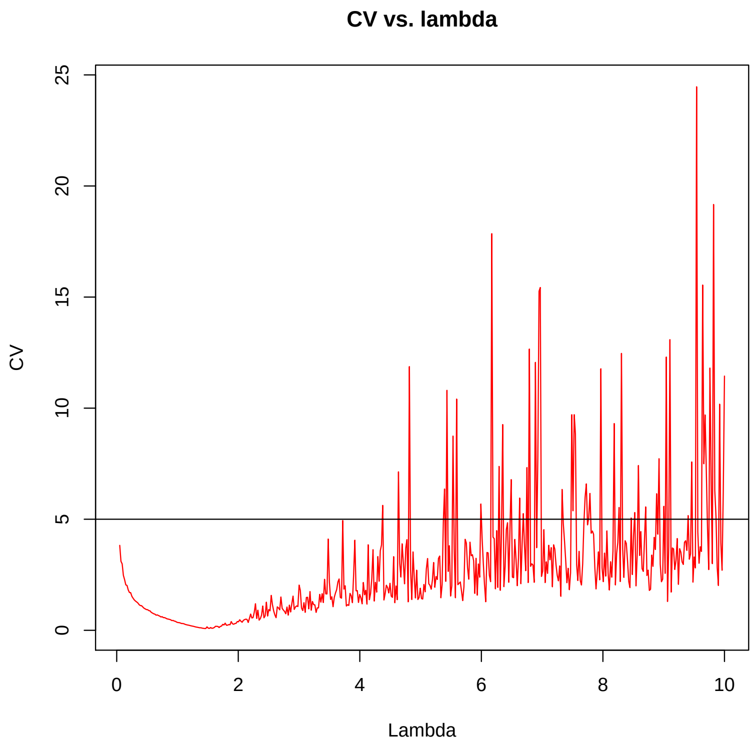
5. To choose the best λ that minimizes CV in $g(x)$, use a grid of λ values for $g(x)$ and compute the corresponding CV by $sqrt(var((\tilde{w}(\vec{x})/mean(\tilde{w}(\vec{x})))))$.

6. compute the estimation error as square root of $1/n$ times sample variance of $\left\{\frac{h(x_k)\tilde{\omega}(x_k)}{\bar{\omega}}\right\}_{k=1:n\_mc}$ where $\bar{\omega} = mean(\tilde{\omega}(\vec{x}))$ in R.

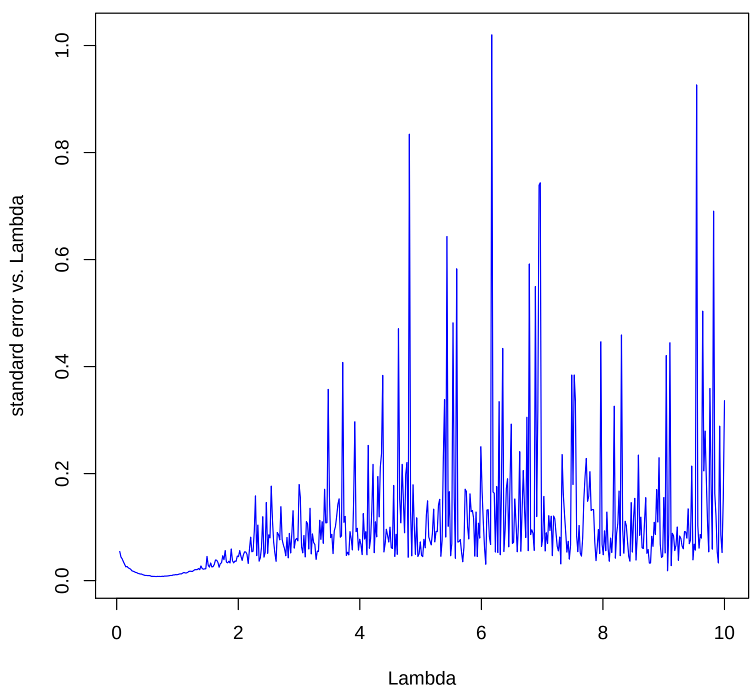
Implementation:

```
# n_mc is the number of monte carlo samples,
# lambda is the rate parameter for the g (exponential) distribution

IS = function(input){

lambda = input[1]
n_mc = input[2]

# Target density with unknow constant C
f = function(t) (exp(-t) / (t+1))

# trial density, g
g = function(t) dexp(t, lambda)

# importance function, actually the tilde w in algorithm step 2
w = function(t) f(t) / g(t)

# draw sample from g
X = rexp(n_mc, lambda)

# calculate the list of importance values
LW = w(X)

# importance sampling estimate
I = mean( X * LW ) / mean(LW)

# calculate sample coefficient of variation CV
CV = sqrt( var( LW / mean(LW) ) )

# calculate importance sampling error
sig.sq = var( LW*X / mean(LW) )
se = sqrt( sig.sq / n_mc )

output = c(I, se, CV)
return(output)

}

## calculate CV for a grid of values of lambda
lambda.val <- seq(.05, 10, length=500)
n_mc.val <- rep(1000, 500)
# inpt.mat is a 500 times 2 matrix containing all the inputs, each row
```

```
of inpt.mat is an input
inpt.mat <- matrix(c(lambda.val, n_mc.val), nrow = 500, ncol = 2)

## apply the weighted average Importance Sampling function

IS() to every row in inpt.mat
A <- apply(inpt.mat,1, IS) # each output of IS() function as c(I, se, CV) is put
as a column of A
A <- t(A) # transpose A to get each

row of A as one output.


# see where CV is low enough
plot(lambda.val, A[,3], ylab="CV", xlab="Lambda", main="CV vs. lambda", col=2,
type="l")
abline(h=5) # only those with a CV value below 5 are considered stable


# importance sampling error estimates (standard errors of IS estimates)
plot(lambda.val, A[,2], xlab="Lambda", ylab="standard error vs. Lambda", col=4,
type="l")


# final answer : the one exp(lambda) denstiy resulting in the
# smallest CV and
#its corresponding IS() outputs c(I, se, CV). You can see
#its estimation error se is also small among other choices of lambda.
indx = which.min(A[,3])
fin.ans <- c(lambda.val[indx], A[indx,])
```



CV vs. lambda
CV
25
20
15
10
5
0
0
2
4
6
8
10
Lambda



standard error vs. Lambda
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Lambda