STAT 406 Lab 7, 10/29/2015

1 Importance Sampling: more intuiation and some comparison

Recall: Last lab, we reviewed *basic Monte Carlo integration*, introduced two versions of importance sampling, the first one only works for fully specified target density $\pi(x)$; Another, called "*weighted average Importance sampling*" works even when $\pi(x)$ is only specified up to an multiplicative constant.

Today, we are interestd in the following question of *evaluating an expectation* with Importance Sampling:

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 mean $\mu = \mathbb{E}\left(h(X)\right) = \int h(x)\pi(x)dx$.

Importance Sampling Algorithm:

(1) Draw $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ from an appropriate trial distribution $g(\cdot)$;

(2) Calculate the *importance weight*

$$w_j = \frac{\pi(\mathbf{x_j})}{g(\mathbf{x_j})}, \quad \text{for } j = 1, 2, \ldots, n$$

(3) Approximate μ by **"weighted average"**

$$\hat{\mu} = \frac{w_1 h(\mathbf{x_1}) + \cdots + w_n h(\mathbf{x_n})}{w_1 + \cdots + w_n} \tag{1.0.1}$$

or by **"average of weighted sum"**

$$\tilde{\mu} = \frac{1}{n}\left\{w_1 h(\mathbf{x_1}) + \cdots + w_n h(\mathbf{x_n})\right\} \tag{1.0.2}$$

Remarks:

In the above importance sampling algorithm, a major advantage of using (1.0.1) instead of using the unbiased (1.0.2) is that in using the former, one needs *only* to know the ratio $\pi(\mathbf{x})/g(\mathbf{x})$ up to a multiplicative constant, versus in (1.0.2) the ratio needs to be known exactly.

Also, in order to make the estimation error small, one wants to choose $g(\mathbf{x})$ as close in shape to $h(x)\pi(x)$ as possible.

Let us work on an example to have a hands on experience of the above.

Example 1: Choice of trial distribution $g(x)$:

Suppose $\mathbf{x}$ follows an *exponential distribution* with rate $\lambda = 1$. Compute the expectation $\mathbb{E}\left[\mathbf{x}^2(cos(\mathbf{x}/3))^2\right]$.

Solution:

firstly, for exponential distribution with rate 1, $\pi(\mathbf{x}) = e^{-\mathbf{x}}$. We evaluate $\mu = \mathbb{E}(\mathbf{x}^2(cos(\mathbf{x}/3))^2) = \int_0^\infty x^2(cos(x/3))^2 e^{-x}dx$ by importance sampling.

How to choose the trial distribution $g(x)$? We will try the following two choices and compare them:

a). choose $g(x) = \lambda e^{-\lambda x}$, we will try $\lambda = 0.5$ and 1. Notice that when $\lambda = 1$, $g(x) = \pi(x)$ and the importance weights are all 1, which is just basic Monte Carlo.

Algorithm:

(1) Draw $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ by $\{-\log(\mathbf{u}_j)/\lambda\}$, $\mathbf{u}_j$ is uniform(0,1) variable , $j = 1, \ldots, n$;

(2) Calculate the *importance weight*

$$w_j = \frac{\pi(\mathbf{x_j})}{g(\mathbf{x_j})} \propto e^{-(1-\lambda)\mathbf{x}_j}, \quad \text{for } j = 1, 2, \ldots, n$$

(3) Approximate μ by

$$\hat{\mu} = \frac{w_1 h(\mathbf{x_1}) + \cdots + w_n h(\mathbf{x_n})}{w_1 + \cdots + w_n}$$

Implement:

b). choose $g(x)$ to be a $Gamma(\alpha = 3, \beta = 1)$ distribution density $g(x) = \frac{1}{\Gamma(3)} x^2 e^{-x}$, where $\Gamma(3) = 2$. In R, this Gamma random variables can be generated by $rgamma(n, shape = 3, rate = 1)$ or by the sum of exponential(1) random variables as introduced in Lab3.

Algorithm:

(1) Draw $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ by $rgamma(n, shape = 3, rate = 1)$;

(2) Calculate the *importance weight*

$$w_j = \frac{\pi(\mathbf{x_j})}{g(\mathbf{x_j})} \propto \mathbf{x}_j^{-2}, \quad \text{for } j = 1, 2, \ldots, n$$

(3) Approximate μ by

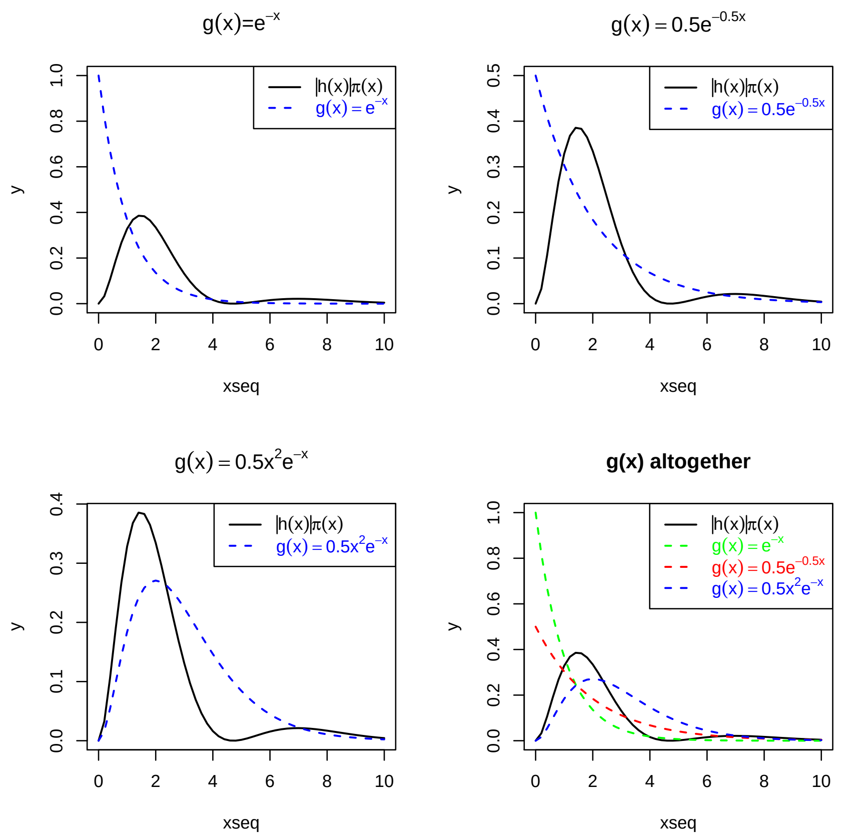
$$\hat{\mu} = \frac{w_1 h(\mathbf{x_1}) + \cdots + w_n h(\mathbf{x_n})}{w_1 + \cdots + w_n}$$

Implement:

Comparison of a) and b):

We will **compare the choices of $g(x)$ in a) and b)** according to suggestions specified in lecture notes of Professor Atchade:

- Coverage/Support matching: all of our trial densities $g(x)$ have the same coverage $(0, \infty)$ as $\pi(x)$, we are good;
- Resemblance: choose $g(x)$ such that it's close in shape to $|h(x)|\pi(x)$;
  The following plots compares the shapes of $|h(x)|\pi(x)$ versus various $g(x)$ we have used:



- Stability: the importance function $w(x) = \pi(x)/g(x) < M$, for some finite M constant. We are good here but will not go to details in this lab. If you are interested, a hint: we chose $\lambda = 0.5, 1 \leq 1$ in exponential trial and the rate parameter $\beta = 1 \leq 1$ in Gamma trial to make sure we are good here!

And the consequence of using these different $g(x)$ for the estimation?

For one run, when sample size $n = 1000$, I get the following estimate, error and 95% confidence intervals:

Trial density	Estimate	Estimation error	confidence interval
$g(x) = e^{-x}$	1.052	0.094	[0.868, 1.234]
$g(x) = 0.5e^{-0.5x}$	0.903	0.021	[0.862, 0.943]
$g(x) = 0.5x^2e^{-x}$	0.896	0.020	[0.857, 0.935]

And the true value $\mu \approx 0.89$, for which the Gamma trial is most accurate and precise in this run. A better comparison is to estimate μ along a sample path of increasing sample sizes, but it's safe to draw a conclusion here that $g(x) = \pi(x)$, i.e. basic Monte Carlo is not the best choice, and the closer $g(x)$ mimicks $|h(x)|\,\pi(x)$ the smaller the estimation error will be.

Example 2: IS with unknown constant C in target density $\pi(x)$:

Compute the first moment (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

$$\int_0^\infty x\pi(x)dx \tag{1.0.3}$$

, 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: $\pi(x) = \tilde{\pi}(x)/C$ where the constant C is unknow.

Here, only "weighted average" (1.0.1) of the two versions of importance sampling can be applied, since $w(x) = \pi(x)/g(x) = \tilde{\pi}(x)/Cg(x)$ can only be known as $w(x) \propto \tilde{\pi}(x)/g(x)$. Choosing $g(x) = \lambda e^{-\lambda x}$, we esimate (1.0.3) by the Importance Sampling algorithm:

Algorithm:

(1) Draw $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ by $rexp(n, rate = \lambda)$;

(2) Calculate the *importance weight*

$$w_j = \frac{\pi(\mathbf{x_j})}{g(\mathbf{x_j})} \propto \frac{e^{-(1-\lambda)\mathbf{x}_j}}{1+\mathbf{x}_j}, \quad \text{for } j = 1, 2, \ldots, n$$

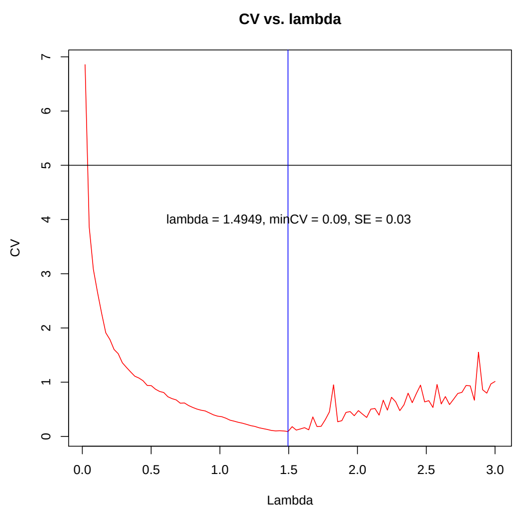
(3) Approximate μ by

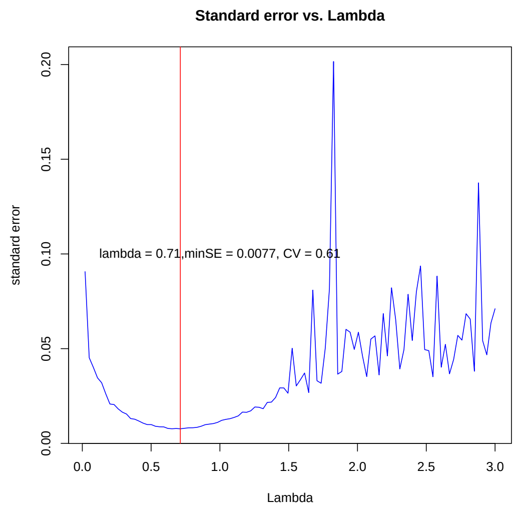
$$\hat{\mu} = \frac{w_1 h(\mathbf{x_1}) + \cdots + w_n h(\mathbf{x_n})}{w_1 + \cdots + w_n}$$

Implementation: notice to check this implementation with respect to the above algorithm and make sure you understand how do they match and how does it work for weighted average importance sampling method.

```
IS = function(input){

lambda = input[1]
n = 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
w = function(t) f(t) / g(t)

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

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

# importance sampling estimate
I = sum( X * W ) / sum(W)

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

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

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

}

## calculate CV for a grid of values of lambda
```

```
lambda.val <- seq(.02, 3, length=100)
n.val <- rep(1000, 100)
# inpt.mat is a 100 times 2 matrix containing all the inputs, each row

of inpt.mat is an input
inpt.mat <- matrix(c(lambda.val, n.val), nrow = 100, ncol = 2)

## estimate mu for each input row in inpt.mat
A <- apply(inpt.mat,1, IS)
# each output of IS() function as c(I, se, CV) is contained a column of A
A <- t(A)

## 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


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


indx = which.min(A[,3])
fin.ans <- c(lambda.val[indx], A[indx,])
# > fin.ans
# [1] 1.49494949 0.67352477 0.02648119 0.08984800
# CV-optimal lambda by minimize CV : 1.49494949 (>1)
# estimate of mu under CV-optimal lambda : 0.67352477
# standard error: 0.02648119
# minimum CV value : 0.08984800

## if we try to minimize the estimation standard error
indx_se <- which.min(A[,2])
ans.mse <- c(lambda.val[indx_se], A[indx_se,])
# > ans.mse
# [1] 0.712323232 0.650455484 0.007672835 0.612279636
# SE-optimal lambda by minimize standard error : 0.712323232 (<1)
# estimate of mu under SE-optimal lambda : 0.650455484
# standard error: 0.007672835
# corresponding CV value : 0.612279636
```





Which criterion do you think one should use ? minimize CV ? minimize SE ?