HW4

Haojie Liu

2023-11-03

```
library(ggplot2)
library(truncnorm)
```

Problem 1: Suppose $\theta = \int_0^2 xe^{-3x}$. Please write corresponding algorithms and functions to answer (a) - (d)

(a) Compute the Monte Carlo estimate of θ without any means of variance reduction.

1. Define the Function: You've already defined the function fx(x) = x * exp(-3 * x).
2. Generate Uniform Random Numbers: Generate a large number of uniform random numbers over the interval [0, 2].
3. Apply the Function to Random Numbers: Apply fx to each of these random numbers.
4. Calculate the Average: Compute the average of these function values.
5. Multiply by the Length of the Interval: Multiply this average by the length of the interval (which is 2 in this case) to get the estimate of the integral.

```
fx <- function(x) { x * exp(-3 * x) }

m <- 10000
u <- runif(m, 0, 2)
mc1_estimates <- fx(u)

theta_mc1 <- mean(mc1_estimates)*2
theta_mc1
```

```
## [1] 0.1095062
```

(b) Compute the Monte Carlo estimate of θ using the antithetic variate approach.

1. Define the Function: The function fx(x) = x * exp(-3 * x) is given.
2. Generate Uniform Random Numbers: Generate a set of uniform random numbers over the interval [0, 2], but only half the total number of desired samples.
3. Create Antithetic Variates: For each random number u, calculate its antithetic variate v = 2 - u. This exploits the symmetry of the uniform distribution over [0, 2].
4. Combine Original and Antithetic Samples: Combine the original samples and their antithetic counterparts into one dataset.
5. Apply the Function to Combined Samples: Apply fx to each of these combined samples.
6. Calculate the Average: Compute the average of these function values.
7. Multiply by the Length of the Interval: Multiply this average by the length of the interval (which is 2) to get the estimate of the integral.

```r
m <- 10000
u <- runif(m / 2, 0, 2)
v <- 2 - u  # Antithetic variates
uv <- c(u, v)
anti_mc_estimates <- fx(uv)

theta_anti_mc <- mean(anti_mc_estimates)*2

theta_anti_mc
```

```
## [1] 0.1096606
```

(c) Compute the Monte Carlo estimate of θ using the control variate approach.

1. Define Functions: You've defined f(u) = exp(u) as the control variate function and g(u) = u * exp(-3 * u) as the target function.
2. Initial Sampling: You generate uniform random samples u over [0, 2] and compute A = g(u) and B = f(u).
3. Calculate the Coefficient a: The coefficient a is calculated to minimize the variance of the estimator.
4. New Sampling for Estimation: You generate a new set of uniform random samples u.
5. Estimate Without and With Control Variates: T1 is the estimate of the integral of g(u) over [0, 2], multiplied by 2 (the length of the interval). T2 is the improved estimate using the control variate technique.
6. Calculate the Mean of Both Estimates: You calculate the mean of T1 and T2, which are the final estimates without and with control variates, respectively.
7. Variance Reduction Calculation: Finally, you calculate the variance reduction achieved by the control variates method.

```r
f <- function(u){
  exp(u)
}
g <- function(u){
  u*exp(-3*u)
}

u <- runif(10000, 0, 2)
B <-f(u)
A <-g(u)

a <- -cov(A,B)/var(B)
u <- runif(10000, 0, 2)
T1 <- g(u)*2
T2 <- T1 + a * (f(u) - mean(B))
cbind(mean(T1), mean(T2))
```

```
##           [,1]     [,2]
## [1,] 0.1082181 0.108405
```

```r
con_var <- (var(T1) - var(T2))/var(T1)
con_var
```

```
## [1] 0.529722
```

(d) Compute the Monte Carlo estimate of θ using the stratified sampling approach.

1. Initialize Parameters: Set the total number of samples m and the number of strata n_strata.
2. Initialize Storage for Stratified Estimates: Create a numeric vector stratified_estimates to store the mean estimates from each stratum.
3. Stratified Sampling Loop: For each stratum, calculate the lower and upper bounds, generate uniform random samples within these bounds, apply the function fx to these samples, and store the mean of these function values.
4. Calculate the Overall Estimate: Average the stratified means and multiply by the length of the interval (which is 2 in this case).

```
m <- 10000
n_strata <- 100
stratified_estimates <- numeric(n_strata)

for (i in 1:n_strata) {
    lower <- (i - 1) * 2 / n_strata
    upper <- i * 2 / n_strata
    u <- runif(m / n_strata, min = lower, max = upper)
    stratified_estimates[i] <- mean(fx(u))
}

theta_stratified <- mean(stratified_estimates)*2

theta_stratified
```

```
## [1] 0.1091683
```

(e) Compare your four estimates with the theoretical value, and compute their variances.

```
theta_theoretical <- integrate(fx, lower = 0, upper = 2)$value
variance_theoretical <- integrate(function(x) (fx(x) - theta_theoretical)^2, lower = 0, upper = 2)$valu


data.frame(
  method = c("theoretical" ,"simple Monte Carlo","Anti Monte Carlo", "Control Variate", "Stratified"),
  theta =c(theta_theoretical, theta_mc1, theta_anti_mc, mean(T2), theta_stratified),
  var = c(variance_theoretical, var(mc1_estimates),var(anti_mc_estimates),var(T2),var(stratified_estima
)
```

```
##               method     theta         var
## 1        theoretical 0.1091832 0.009254424
## 2 simple Monte Carlo 0.1095062 0.001659935
## 3   Anti Monte Carlo 0.1096606 0.001632990
## 4    Control Variate 0.1084050 0.003055429
## 5         Stratified 0.1091683 0.001662734
```

Problem 2: Consider the problem of estimating $\theta = \int_0^1 \frac{e^{-x}}{1+x^2} dx$ Given two control variates as follows, please write R code to estimate θ using the control variate approach.

$$t_1(x) = \frac{2e^{-0.5}}{\pi(1+x^2)}, \ where \ 0 < x < 1$$

```
t1 <- function(u){
  (2*exp(-0.5))/(pi*(1+u^2))
}
g <- function(u){
  exp(-u)/(1+u^2)
}

u <- runif(10000, 0, 1)
B <-t1(u)
A <-g(u)
a1 <- -cov(A,B)/var(B)
u <- runif(10000, 0, 1)
T1 <- g(u)
T2 <- T1 + a1 * (t1(u) - mean(B))
mean(T1)
```

```
## [1] 0.527365
```

```
con_var <- (var(T1) - var(T2))/var(T1)
con_var
```

```
## [1] 0.9476242
```

$$t_2(x) = \frac{e^{-x}}{1-e^{-1}},\ where\ 0 < x < 1$$

```
t2 <- function(u){
  (exp(-u))/(1-exp(-1))
}

u <- runif(10000, 0, 1)
B <-t2(u)
A <-g(u)
a2 <- -cov(A,B)/var(B)
u <- runif(10000, 0, 1)
T1 <- g(u)
T2 <- T1 + a2 * (t2(u) - (mean(B)))
mean(T1)
```

```
## [1] 0.5278703
```

```
con_var <- (var(T1) - var(T2))/var(T1)
con_var
```

```
## [1] 0.9990063
```

Is your estimator more efficient than the simple Monte Carlo estimator? Please provide an evidence to support your answer.

```r
m <- 10000
u <- runif(m, 0, 1)
mc1_estimates <- g(u)
var(mc1_estimates)
```

```
## [1] 0.06010051
```

Since both of the control variance reductions are larger than zero and above 90, I can say that my estimator are more efficient than the simple Monte Carlo estimator.

Problem 3: Suppose $X \sim f(x)$, Let $\theta = \int g(x)f(x)dx = E_f[g(X)]$. We draw m iid copies $X_1, ...X_m$ from $\phi(x)$, which is different from $f(x)$, and define $W(x) = \frac{f(x)}{\phi(x)}$

(a) Prove $E_f[g(X)] = E_\phi[g(X)W(X)]$

$$\phi(x) = \frac{f(x)}{W(x)}$$

$$E_\phi[g(X)W(X)] = \int g(x)W(x)\phi(x)dx$$

$$E_\phi[g(X)W(X)] = \int g(x)\frac{f(x)}{\phi(x)}\phi(x)dx = \int g(x)f(x)dx$$

$$E_f[g(X)] = \int g(x)f(x)dx = E_\phi[g(X)W(X)]$$

(b) Prove $E_\phi[W(X)] = 1$

$$E_\phi[W(X)] = \int W(x)\phi(x)dx$$

$$E_\phi[W(X)] = \int \frac{f(x)}{\phi(x)}\phi(x)dx = \int f(x)dx = 1$$

(c) Let $\hat{\theta} = \Sigma_{i=1}^m g(X)W(X)/m$. Find $E[\hat{\theta}]$ and $Var[\hat{\theta}]$

$$E(\hat{\theta}) = E(\Sigma_{i=1}^m g(X_i)W(X_i)/m) = \frac{1}{m}\Sigma_{i=1}^m E[g(X_i)W(X_i)]$$

Since that X_i are iid samples and $E_f[g(X)] = E_\phi[g(X)W(X)]$

$$E(\hat{\theta}) = E[g(X)W(X)] = E_f[g(X)]$$

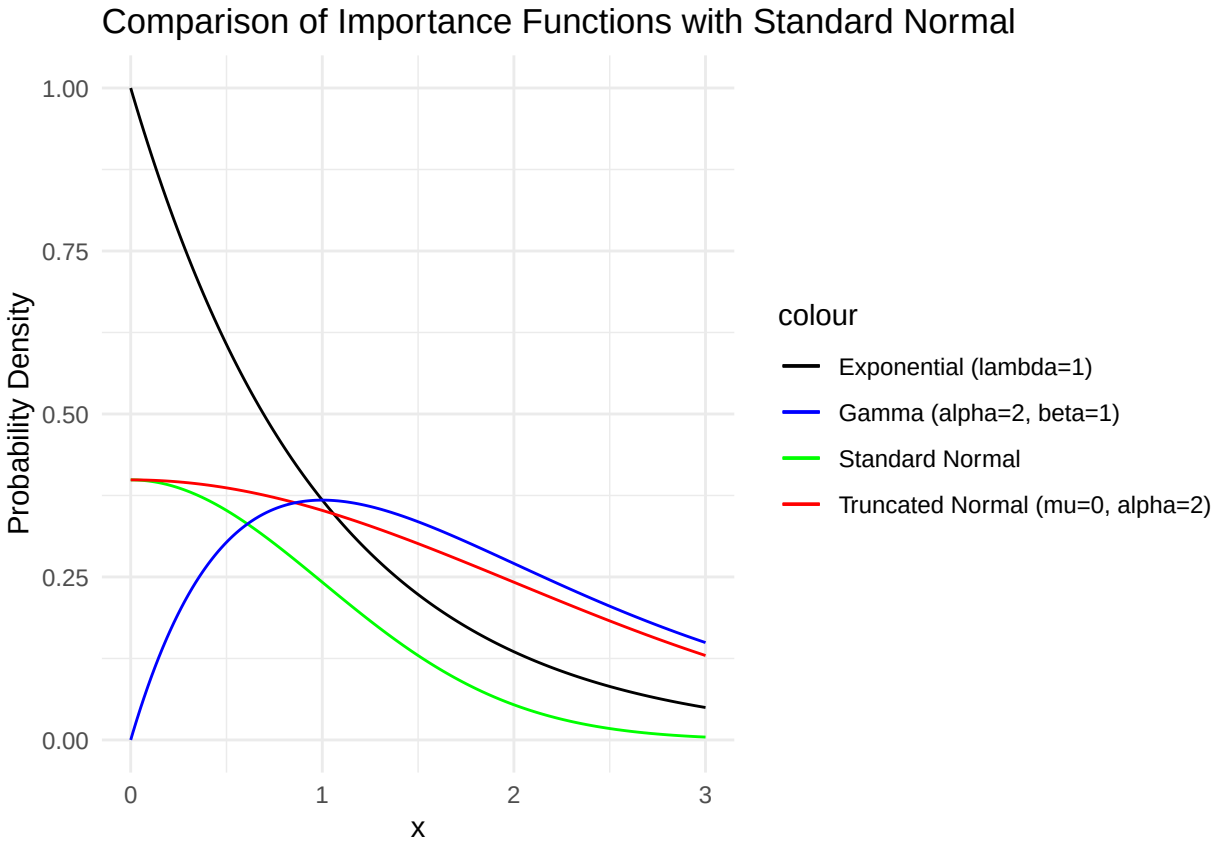
$$Var[\hat{\theta}] = Var[\Sigma_{i=1}^m g(X_i)W(X_i)/m] = \frac{1}{m^2}\Sigma_{i=1}^m Var[g(X_i)W(X_i)] = \frac{1}{m}Var[g(X)W(X)]$$

$$Var[\hat{\theta}] = \frac{1}{m}(E[g(X)^2W(X)^2] - E_f[g(X)]^2)$$

Problem 4: Given $X \sim N(0,1)$, we want to compute $\theta = P(X > C)$ where C is a positive constant.

(a) Find three importance functions that are supported on $(0, \infty)$, and explain which of your importance functions should produce the smaller $Var[\hat{\theta}]$. Please graph plots to support your answer.

```r
# Define the range for x
x <- seq(0, 3, length.out = 1000)

# Standard normal distribution
std_normal <- dnorm(x, mean = 0, sd = 1)

# Exponential distribution (lambda = 1)
exp_dist <- dexp(x, rate = 1)

# Truncated normal distribution (mu = 1, sigma = 0.5)
trunc_normal <- dtruncnorm(x, a = 0, b = Inf, mean = 0, sd = 2)

# Gamma distribution (alpha = 2, beta = 1)
gamma_dist <- dgamma(x, shape = 2, rate = 1)

df <- data.frame(x, std_normal, exp_dist, trunc_normal, gamma_dist)

ggplot(df, aes(x)) +
  geom_line(aes(y = std_normal, color = "Standard Normal")) +
  geom_line(aes(y = exp_dist, color = "Exponential (lambda=1)")) +
  geom_line(aes(y = trunc_normal, color = "Truncated Normal (mu=0, alpha=2)")) +
  geom_line(aes(y = gamma_dist, color = "Gamma (alpha=2, beta=1)")) +
  labs(title = "Comparison of Importance Functions with Standard Normal",
       x = "x", y = "Probability Density") +
  scale_color_manual(values = c("black", "blue", "green", "red")) +
  theme_minimal()
```



Among these, the Truncated Normal distribution is likely to yield the smallest $Var[\hat{\theta}]$ if its parameters are chosen correctly, as it can most closely match the tail of the standard normal distribution.

(b) Write a function to compute a Monte Carlo estimate of θ using importance functions you proposed in (a). Note: You can use the built-in functions in R to generate random variables.

```r
importance_sampling <- function(importance_dist, params, C, n) {

  if (importance_dist == "exponential") {
    samples <- rexp(n, rate=params$lambda)
  } else if (importance_dist == "truncated_normal") {
    samples <- rtruncnorm(n, a=0, mean=params$mu, sd=params$sigma)
  } else if (importance_dist == "gamma") {
    samples <- rgamma(n, shape=params$alpha, rate=params$beta)
  }

  f <- dnorm(samples, mean=0, sd=1)
  g <- switch(importance_dist,
              "exponential" = dexp(samples, rate=params$lambda),
              "truncated_normal" = dtruncnorm(samples, a=0, mean=params$mu, sd=params$sigma),
              "gamma" = dgamma(samples, shape=params$alpha, rate=params$beta))

  weights <- f / g

  # Estimate theta
  theta_hat <- mean(weights * (samples > C))
```

```
  return(theta_hat)
}
```

(c) Compare your estimates with the theoretical values for C = 0.25, 0.5, 1, 2.

```
C_values <- c(0.25, 0.5, 1, 2)

n_samples <- 10000

results <- sapply(C_values, function(C) {

  theoretical <- 1 - pnorm(C)

  mc_exp <- importance_sampling("exponential", list(lambda = 1), C, n_samples)
  mc_trunc_normal <- importance_sampling("truncated_normal", list(mu = 0, sigma = 2), C, n_samples)
  mc_gamma <- importance_sampling("gamma", list(alpha = 2, beta = 1), C, n_samples)

  c(Theoretical = theoretical, Exponential = mc_exp, Truncated_Normal = mc_trunc_normal, Gamma = mc_gamm
})

results_df <- data.frame(C = C_values, t(results))
results_df
```

```
##      C Theoretical Exponential Truncated_Normal      Gamma
## 1 0.25  0.40129367  0.40292528       0.39584733 0.40170394
## 2 0.50  0.30853754  0.30938248       0.30933359 0.31332648
## 3 1.00  0.15865525  0.15791701       0.15584142 0.15842906
## 4 2.00  0.02275013  0.02238423       0.02292144 0.02282133
```

Problem 5: The density function $f(x)$ is proportional to $q(x) = e^{-x^2\sqrt{x}}[sin(x)]^2$, for $x \in R_+$. Consider a trial distribution $h(x) = \frac{r(x)}{Z_r}$. We want to use importance sampling to estimate $E_f(X^2)$. Consider the following choices of un-normalized densities for trial distributions:

(i) $r_1(x) = e^{-2x}$,
(ii) $r_2(x) = x^{-1/2}e^{-x/2}$,
(iii) $r_3(x) = (2\pi)^{-1}(1 + x^2/4)^{-1}$,

for $x \in R_+$

(a) Write three functions (E1, E2, and E3) to estimate $E_f(X^2)$ using importance sampling with the above un-normalized densities.

```
qx <- function(x) {
    exp(-x^2 * sqrt(x)) * sin(x)^2
}

E1 <- function(n){

  X <- rexp(n,2)
  r1 <- function(x){exp(-2*x)}
```

```
  W <- qx(X)/r1(X)
  return(sum(W*X^2)/sum(W))

}

E2 <- function(n){

  X <- rgamma(n,0.5,2)
  r2 <- function(x){x^(-1/2) * exp(-x/2)}
  W <- qx(X)/r2(X)
  return(sum(W*X^2)/sum(W))

}

E3 <- function(n){

  X <- abs(rcauchy(n, 2))
  r3 <- function(x){(2 * pi)^(-1) * (1 + x^2 / 4)^(-1)}
  W <- qx(X)/r3(X)
  return(sum(W*X^2)/sum(W))

}
```

(b) Compare your estimates with the theoretical value.

```
norm_const <- integrate(Vectorize(qx), lower = 0, upper = Inf)$value

normalized_qx <- function(x) {
  qx(x) / norm_const
}

theoretical_value <- integrate(Vectorize(function(x) x^2 * normalized_qx(x)), lower = 0, upper = Inf)$va


data.frame(
  theoretical_value,
  E1 = E1(10000),
  E2 = E2(10000),
  E3 = E3(10000)
  )
```

```
##   theoretical_value       E1       E2       E3
## 1         0.8913028 0.899758 0.613241 1.169566
```

(c) Which trial distribution above is more efficient? Explain.

Based on the given results, the trial distribution used in E1 appears to be the most efficient as its estimate (0.888646) is closest to the theoretical value (0.8913028), indicating higher accuracy compared to the other two methods.

(d) Estimate the normalizing constant of $q(x)$.

```r
Q <- integrate(qx, lower = 0, upper = Inf)

Q$value
```

```
## [1] 0.259539
```