ps4\_rmd

Jared Yu

February 19, 2020

library(latex2exp) # Load libary  
### Problem 1  
set.seed(664)  
x <- runif(1000000)  
g <- function(x) {  
 4 \* sqrt(1 - x^2)  
}  
mean(g(x)) # 3.141308  
# analytically solves to pi  
  
### Problem 2  
# part (a)  
triangular\_pdf <- function(x) { # target distribution pdf  
 if((0 <= x) & (x <= 0.5)) {  
 return(4 \* x)  
 } else if((0.5 < x) & (x <= 1)) {  
 return(4 - 4 \* x)  
 } else {  
 return(0)  
 }  
}  
  
triangular\_pdf\_vec <- Vectorize(triangular\_pdf)  
xs <- seq(0, 1, length.out = 500)  
ys <- triangular\_pdf\_vec(x = xs)  
par(mfrow = c(1,2))  
plot(xs, ys, type = 'l', main = 'Plot of Triangular p.d.f.',  
 xlab = TeX('$x$'), ylab = TeX('$f(x)$'))  
abline(h = 0)  
abline(v = 0)  
  
triangular\_cdf <- function(x) { # target distribution CDF  
 if((0 <= x) & (x <= 0.5)) {  
 return(2 \* x^2)  
 } else if((0.5 < x) & (x <= 1)) {  
 return(-2 \* x^2 + 4 \* x - 1)  
 } else {  
 return(0)  
 }  
}  
  
triangular\_cdf\_vec <- Vectorize(triangular\_cdf)  
xs <- seq(0, 1, length.out = 500)  
ys <- triangular\_cdf\_vec(x = xs)  
plot(xs, ys, type = 'l', main = 'Plot of Triangular CDF',  
 xlab = TeX('$x$'), ylab = TeX('$F(x)$'))  
abline(h = 0)  
abline(v = 0)  
dev.off()  
  
# part (b)  
inverse\_cdf <- function(y) { # Inverse CDF  
 if ((0 <= y) & (y <= 0.5)) {  
 return(sqrt(y / 2))  
 } else if ((0.5 < y) & (y <= 1)) {  
 return((4 - sqrt(8 - 8 \* y)) / 4)  
 } else {  
 return(0)  
 }  
}  
  
G <- function(u) inverse\_cdf(u) # Function 'G(u)'  
G\_vec <- Vectorize(G)  
  
# part (c)  
set.seed(664)  
us <- runif(n = 1e3)  
inverse\_cdf\_samples <- G\_vec(us)  
hist(inverse\_cdf\_samples, xlab = 'Sample Values',  
 main = 'Histogram of Samples Generated by Inverse CDF Method')  
  
# part (d)  
xs <- seq(0, 1, length.out = 500) # graph of target distribution  
ys <- triangular\_pdf\_vec(x = xs)  
plot(xs, ys, type = 'l', main = 'Graph of Target and Envelope',  
 xlab = TeX('$x$'), ylab = TeX('$f(x)$'), ylim = c(0,2.5), xlim = c(0, 1.05))  
abline(h = 0)  
abline(v = 0)  
  
bounds <- seq(0, 1, length.out = 1000) # graph of instrumental distribution  
gs <- dnorm(bounds, mean = 0.5, sd = 0.25)  
alpha <- 0.75  
es <- gs / alpha  
lines(bounds, es, col = 'red')  
abline(v = 1, col = 'red')  
legend("topleft", c("Target Distribution", "Envelope"),  
 col = c("black", "red"), lty = c(1,1))  
  
rejection\_rule <- function(u, y, alpha = 0.75) { # Used in rejection\_sampling()  
 f\_y <- triangular\_pdf(y)  
 # Reference: https://www.r-bloggers.com/normal-distribution-functions/  
 envelope <- dnorm(y, mean = 0.5, sd = 0.25) / alpha  
 if ((u > (f\_y / envelope)) | (y < 0) | (y > 1)) {  
 return(FALSE)  
 } else {  
 return(TRUE)  
 }  
}  
  
rejection\_sampling <- function(num\_samples = 1000) { # Performs rejection sampling  
 sampling\_matrix <- matrix(NA, nrow = num\_samples)  
 count <- 0; rejects <- 0 # Initialize variables  
   
 while (count != num\_samples) { # Sample values  
 y <- rnorm(n = 1, mean = 0.5, sd = 0.25)  
 u <- runif(n = 1, min = 0, max = 1)  
   
 if (rejection\_rule(u, y)) { # Check for rejection  
 sampling\_matrix[count + 1] <- y  
 count <- count + 1 # Accept values  
 } else {  
 rejects <- rejects + 1  
 }  
 }  
 cat('Rejects:', rejects)  
 return(sampling\_matrix)  
}  
  
set.seed(664)  
rejection\_samples <- rejection\_sampling()  
1 - 334 / 1000 # acceptance rate = 0.666, somewhat close to alpha = 0.75  
hist(as.vector(rejection\_samples), xlab = 'Sample Value', # histogram  
 main = 'Histogram of Samples Using Rejection Sampling')  
  
# part (e)  
mean(inverse\_cdf\_samples^2) # 0.2916822  
7/24 # 0.2916667  
  
### Problem 3  
# part (a)  
std\_normal\_pdf <- function(x) dnorm(x = x, mean = 0, sd = 1)  
xs <- seq(-5, 5, length.out = 500) # graph of target distribution  
ys <- std\_normal\_pdf(x = xs)  
plot(xs, ys, type = 'l', main = 'Graph of Target and Envelope',  
 xlab = TeX('$x$'), ylab = TeX('$f(x)$'), ylim = c(0, 0.8))  
abline(h = 0)  
abline(v = 0)  
  
bounds <- seq(-5, 5, length.out = 1000) # graph of instrumental distribution  
normal\_1\_2 <- function(x) dnorm(x = x, mean = 1, sd = sqrt(2))  
gs <- normal\_1\_2(bounds)  
alpha <- 0.4  
es <- gs / alpha # envelope  
lines(bounds, es, col = 'red')  
legend("topleft", c("Target Distribution", "Envelope"),  
 col = c("black", "red"), lty = c(1,1))  
  
rejection\_rule2 <- function(u, y, alpha = 0.4) { # Used in rejection\_sampling()  
 f\_y <- std\_normal\_pdf(x = y)  
 # Reference: https://www.r-bloggers.com/normal-distribution-functions/  
 envelope <- normal\_1\_2(x = y) / alpha  
 if (u > (f\_y / envelope)) {  
 return(FALSE)  
 } else {  
 return(TRUE)  
 }  
}  
  
rejection\_sampling2 <- function(num\_samples = 10000) { # Performs rejection sampling  
 sampling\_matrix <- matrix(NA, nrow = num\_samples)  
 count <- 0; rejects <- 0 # Initialize variables  
   
 while (count != num\_samples) { # Sample values  
 y <- rnorm(n = 1, mean = 1, sd = sqrt(2))  
 u <- runif(n = 1, min = 0, max = 1)  
   
 if (rejection\_rule2(u, y)) { # Check for rejection  
 sampling\_matrix[count + 1] <- y  
 count <- count + 1 # Accept values  
 } else {  
 rejects <- rejects + 1  
 }  
 }  
 cat('Rejects:', rejects)  
 return(sampling\_matrix)  
}  
  
set.seed(664)  
rejection\_samples2 <- rejection\_sampling2()  
1 - 15024 / 10000 # acceptance rate = -0.5024, far from alpha = 0.4  
hist(as.vector(rejection\_samples2), xlab = 'Sample Value', # histogram  
 main = 'Histogram of Samples Using Rejection Sampling')  
mean(rejection\_samples2); var(rejection\_samples2)  
  
# part (b)  
# no code required  
  
# part (c)  
xs <- seq(-3, 3, length.out = 5000) # graph of target distribution  
ys <- std\_normal\_pdf(x = xs)  
plot(xs, ys, type = 'l', main = 'Graph of Target and Squeezing Function',  
 xlab = TeX('$x$'), ylab = TeX('$f(x)$'), ylim = c(0, 0.45))  
abline(h = 0)  
abline(v = 0)  
legend("topleft", c("Target Distribution", "Squeezing Function"),  
 col = c("black", "red"), lty = c(1,1))  
  
squeeze\_function <- function(x) { # squeezing function  
 if ((-0.5 <= x) & (x <= 0.5)) {  
 return(1/3)  
 } else if (((-1 <= x) & (x < -0.5)) | ((0.5 < x) & (x <= 1))) {  
 return(0.2)  
 } else if (((-1.8 <= x) & (x < -1)) | ((1 < x) & (x <= 1.8))) {  
 return(0.07)  
 } else {  
 return(0)  
 }  
}  
squeeze\_function\_vec <- Vectorize(squeeze\_function)  
bounds <- seq(-3, 3, length.out = 5000) # graph the squeeze function  
lines(bounds, squeeze\_function\_vec(bounds), col = 'red')