Question lab4

Helena Yi Yang

2025-02-16

Question 2

```
## Warning: package 'knitr' was built under R version 4.4.2
```

The boundaries of X with density $f(x_1, x_2) \propto \mathbf{1}\{x_1^2 + wx_1x_2 + x_2^2 < 1\}$ when $W = 1.999$ are shown as follows:

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

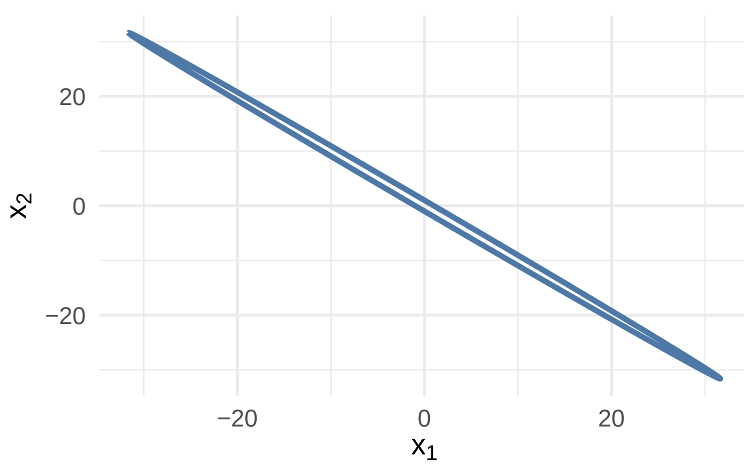

Figure 1: Boundaries of X

The conditional distribution of `X1` given `X2` is a uniform distribution on the interval:

$$\frac{-1.999X_2 - \sqrt{1.999^2X_2^2 + 4(1 - X_2^2)}}{2} < X_1 < \frac{-1.999X_2 + \sqrt{1.999^2X_2^2 + 4(1 - X_2^2)}}{2}$$

Since X has a uniform distribution, the density of

$$f(x_1|x_2) = \frac{1}{\text{interval length}}$$

where the interval length is given by:

$$\text{Interval Length} = \sqrt{1.999^2 x_2^2 + 4(1 - x_2^2)}$$

Thus:

$$f(x_1 \mid x_2) = \frac{1}{\sqrt{1.999^2 x_2^2 + 4(1 - x_2^2)}}$$

The conditional distribution of `X1` given `X2` has a similiar distribution :

$$f(x_2 \mid x_1) = \frac{1}{\sqrt{1.999^2 x_1^2 + 4(1 - x_1^2)}}$$

We are going to generate 1000 random vectors with `Gibbs sampling` method, the resulting plot is shown as follows:

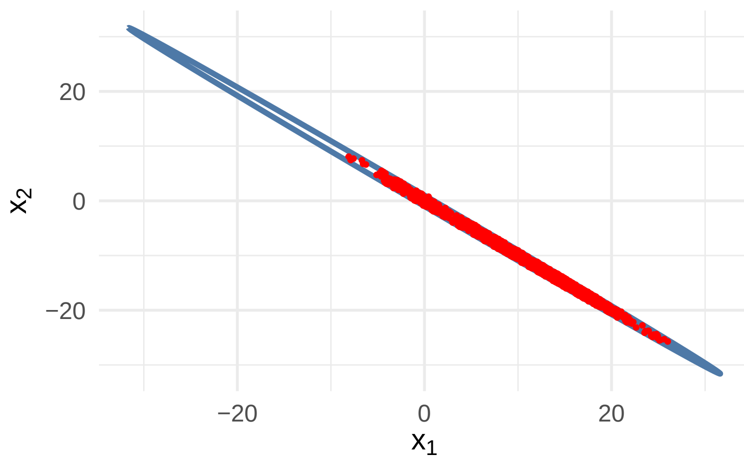

Figure 2: Gibbs Sampling Distribution

Then we repeat the algorithm 10 times and use the samples to calculate $P(X_1 > 0)$, the results are shown in the following table.

Table 1: Probability of X1>0

x
0.864
0.417
0.220
0.151
0.653
0.398
0.433
0.493
0.411
0.769

From the table we can observe that the probability of $X_1 > 0$ varies across different generations.However,the true probability should be **0.5**, as the boundaries of the density distribution are symmetric about the y-axis.

The reason why `Gibbs sampling` is less successful for $W = 1.999$ compared to $W = 1.8$ is that the distribution's boundaries become much narrower when $w = 1.999$.This leads to slower convergence in `Gibbs sampling`.Since `Gibbs sampling` updates one variable at a time while keeping the other variable fixed in this case.The narrow shape restricts the movement of $X1$ and $X2$.The sample values change less in each iteration,causing samples to get stuck in a limited region,which makes it difficult to cover the entire distribution efficiently.

Then, we transform the variable X and generate $U = (U_1, U_2) = (X_1 - X_2, X_1 + X_2)$ instead.By calculating $U_1 = X_1 - X_2$, $U_2 = X_1 + X_2$,we could determine the boundaries of the tranformed region. The plot is shown as follows:

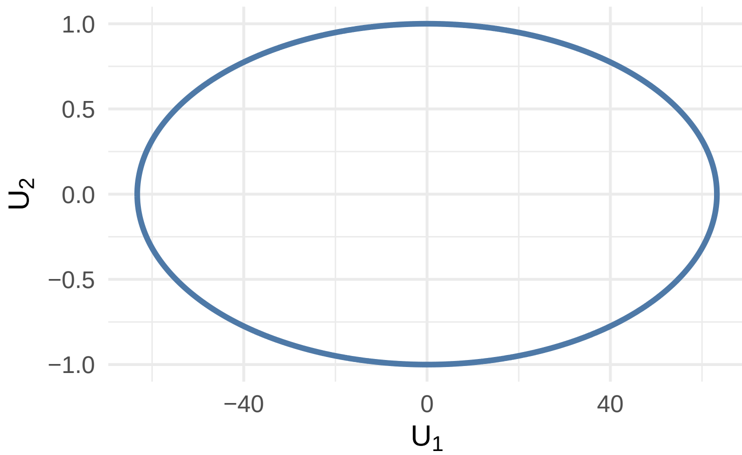

Figure 3: Boundaries of U

Then we use `Gibbs sampling` method to generate 1000 random variables and plot them.

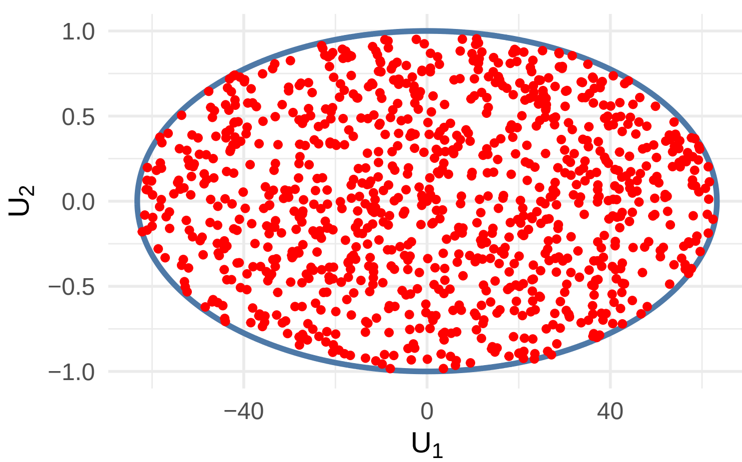

Figure 4: Gibbs Sampling Distribution of U

From the plot, we observe that the samples are approximately uniformly distributed within the boundaries. To verify this, we compute $P(X_1 > 0) = P((U_2 + U_1)/2 > 0)$ using the generated samples.

```
## The probability of X1 > 0: 0.519
```

Then we repeat the algorithm 10 times and the results are shown in the following table.

Table 2: Probability of X1>0

x
0.519
0.491
0.490
0.491
0.516
0.494
0.493
0.501
0.494
0.507

Compared to the result from part c, we observe that the probabilities approach 0.5 after the transformation, indicating improved generation ability.

Appendix

question 2

```r
library(ggplot2)
#a draw the boundary
w  <- 1.999
x_max <- sqrt(4 / (4 - w^2))  # maximum of x1
xv <- seq(-x_max, x_max, by=0.01)  # x1


x2_pos <- -(w/2)*xv + sqrt(1 - (1 - w^2/4) * xv^2)
x2_neg <- -(w/2)*xv - sqrt(1 - (1 - w^2/4) * xv^2)


ellipse_df <- data.frame(
  x1=c(xv,rev(xv)),
  x2=c(x2_pos,rev(x2_neg))
)

ggplot(ellipse_df,aes(x=x1,y=x2))+
  geom_path(color="#4E79A7",size=1)+
  labs(x=expression(x[1]),y=expression(x[2]))+
  theme_minimal()


#b

#c Gibbs sampling

gibbsSampling <- function(n,W){
```

```r
  #initialize x1 and x2
  X1 <- numeric(n)
  X2 <- numeric(n)

  #initialize the first iteration
  X1[1] <- runif(1,-1,1)
  X2[1] <- runif(1,-1,1)

  for(i in 2:n){
   #generating x1 given x2
   X1_range <- c(-0.5*W*X2[i-1]-sqrt(1-(1-1/4*W^2)*X2[i-1]^2),-0.5*W*X2[i-1]+sqrt(1-(1-1/4*W^2)*X2[i-1]
   X1[i] <- runif(1,min = X1_range[1],max = X1_range[2])

   #generating x2 given x1
   X2_range <- c(-0.5*W*X1[i]-sqrt(1-(1-1/4*W^2)*X1[i]^2),-0.5*W*X1[i]+sqrt(1-(1-1/4*W^2)*X1[i]^2))
   X2[i] <- runif(1,min = X2_range[1],max = X2_range[2])

  }
   result <- data.frame(X1,X2)
   return(result)
}
set.seed(12345)
samples <- gibbsSampling(1000,1.999)
ggplot(ellipse_df,aes(x=x1,y=x2))+
  geom_path(color="#4E79A7",size=1)+
  labs(x=expression(x[1]),y=expression(x[2]))+
  geom_point(data=samples,aes(x = X1, y = X2), color = "red", size = 0.5) +
  theme_minimal()


#repeat the sampling
prob_x1 <- function(sample,n){
  prob <- sum(sample$X1>0)/n
  return(prob)

}

probs <- numeric(10)
sample_result <- list()
for (i in 1:10) {
  sample_result[[i]] <- gibbsSampling(1000,1.999)
  probs[i] <- prob_x1(sample_result[[i]],1000)
}
cat("The probabilities of X1 >0 in 10 times :",probs,"\n")


#d

#e generate U


#calculate U1 = X1 - X2, U2 = X1 + X2
U1<- c(xv - x2_pos, rev(xv - x2_neg))
```

```r
U2 <- c(xv + x2_pos, rev(xv + x2_neg))

#plot the boundaries of U
ellipse_U_df <- data.frame(U1 = U1, U2 = U2)
ggplot(ellipse_U_df, aes(x = U1, y = U2)) +
  geom_path(color = "#4E79A7", size = 1) +
  labs(x = expression(U[1]), y = expression(U[2])) +
  theme_minimal()

#generate 1000 random variables using Gibbs Sampling
gibbsSampling_U <- function(n, W) {
  # initialize U1 and U2
  U1 <- numeric(n)
  U2 <- numeric(n)

  # initialize the first iteration
  U1[1] <- 0
  U2[1] <- 0

  for (i in 2:n) {
    # given U2[i-1] generate U1[i]
    numerator_U1 <- 4 - (2 + W) * U2[i-1]^2
    U1_max <- sqrt(numerator_U1 / (2 - W))
    U1_range <- c(-U1_max, U1_max)

    U1[i] <- runif(1, min = U1_range[1], max = U1_range[2])

    # given U1[i] generate U2[i]
    numerator_U2 <- 4 - (2 - W) * U1[i]^2
    U2_max <- sqrt(numerator_U2 / (2 + W))
    U2_range <- c(-U2_max, U2_max)
    U2[i] <- runif(1, min = U2_range[1], max = U2_range[2])
  }
  data.frame(U1, U2)
}
set.seed(12345)
sample_U <- gibbsSampling_U(1000,1.999)

#plot the samples of U
ggplot(ellipse_U_df, aes(x = U1, y = U2)) +
  geom_path(color = "#4E79A7", size = 1) +
  labs(x = expression(U[1]), y = expression(U[2])) +
  geom_point(data=sample_U,aes(x = U1, y = U2), color = "red", size = 1) +
  theme_minimal()

#calculate P(x1>0)
samples_U_transformed_df <- data.frame(
  X1=(sample_U$U1+sample_U$U2)/2,
  X2=(sample_U$U2-sample_U$U1)/2
)

prob_X1_gibbs <- sum(samples_U_transformed_df$X1 > 0) /1000
```

```r
cat("The probability of X1 > 0:",prob_X1_gibbs,"\n")


#repeat the sampling
prob_U <- function(sample,n){
  prob <- sum((sample$U2 + sample$U1) / 2 > 0) / 1000
  return(prob)

}
set.seed(12345)
probs_u <- numeric(10)
sample_U_result <- list()
for (i in 1:10) {
  sample_U_result[[i]] <- gibbsSampling_U(1000,1.999)
  probs_u[i] <- prob_U (sample_U_result[[i]],1000)
}
kable(probs_u,caption = "Probability of X1>0")
```