Centro de Investigación en Matemáticas (CIMAT)
Unidad Monterrey

Computo Estadístico

Tarea 3

Algoritmo MCMC y el método Bootstrap

Diego Paniagua Molina
diego.paniagua@cimat.mx

9 de Octubre del 2025

Problema 1

(a) Muestra que existen $\binom{2n-1}{n}$ distintas muestras de bootstrap de tamaño n.

(b) ¿Cuál es la probabilidad de que una muestra de bootstrap sea idéntica a la original?

(c) ¿Cuál es la muestra de bootstrap mas probable de ser seleccionada?

(d) ¿Cuál es la cantidad promedio de veces que X_i es seleccionada en una muestra de bootstrap?

SOLUCIÓN

Sea una muestra observada $\mathbf{X} = (x_1, \ldots, x_n)$ y sabemos que una réplica de bootstrap $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$ se forma muestreando n veces con reemplazo de $\mathbf{X}$.

a) Si consideramos las muestras de bootstrap como multiconjuntos (es decir, el orden de los elementos no importa), el problema se reduce a contar cuántas veces aparece cada observación original x_i en la réplica de bootstrap. Sea el vector de frecuencias $\mathbf{N} = (N_1, N_2, \ldots, N_n)$, donde N_i es el número de veces que la observación x_i es seleccionada en la muestra bootstrap, dado que el tamaño total de la muestra bootstrap es n, la suma de estas frecuencias debe ser igual a n:

$$N_1 + N_2 + \cdots + N_n = n, \quad N_i \in \{0, 1, 2, \ldots\} \tag{1}$$

Asi el problema se transformo en encontrar el número de soluciones enteras no negativas para esta ecuación, para resolverlo, utilizamos el método grafico de estrellas y barras, el cual nos dice lo siguiente:

1. **Representación con "Estrellas"**: Supongamos que tenemos n unidades que debemos distribuir entre los n posibles contenedores $(N_1, \ldots, N_n)$. Podemos visualizar estas n unidades como n estrellas $(\star)$.
2. **Representación con "Barras"**: Para separar estas estrellas en n grupos distintos, necesitamos $n-1$ separadores o "barras" $(|)$. Por ejemplo, los elementos a la izquierda de la primera barra corresponden a N_1, los que están entre la primera y la segunda barra a N_2, y así sucesivamente, hasta los elementos a la derecha de la última barra, que corresponden a N_n.

Entonces, el problema ahora es equivalente a encontrar el número de maneras distintas en que podemos arreglar en una secuencia las n estrellas y las $n-1$ barras, siendo que tenemos un total de $n+(n-1) = 2n-1$ posiciones en la secuencia. Esto es un problema de combinaciones, donde el número total de arreglos es el número de maneras de elegir cuáles de las $2n-1$ posiciones serán ocupadas por las n estrellas (las posiciones restantes serán automáticamente ocupadas por las barras). El número de formas de elegir k objetos de un conjunto de m es $\binom{m}{k}$ y en nuestro caso, $m = 2n-1$ y $k = n$.

Por lo tanto, el número de soluciones enteras no negativas es:

$$\binom{\text{total de posiciones}}{\text{posiciones para las estrellas}} = \binom{n+n-1}{n} = \binom{2n-1}{n}$$

Esta es la cantidad de multiconjuntos distintos de tamaño n que se pueden formar a partir de los n elementos originales, lo que corresponde al número de muestras de bootstrap distintas.

b) En el contexto de bootstrap, "idéntica" se refiere a que la muestra bootstrap, como multiconjunto, contiene cada una de las observaciones originales exactamente una vez. Esto no significa que el orden de aparición sea el mismo, sino que el vector de frecuencias de las observaciones es:

$$\mathbf{N} = (1, 1, \ldots, 1) \tag{2}$$

Para calcular esta probabilidad, necesitamos contar cuántas secuencias ordenadas de bootstrap cumplen esta condición y calcular la probabilidad de obtener una de esas secuencias específicas.

Una secuencia de bootstrap que cumple nuestra condición es simplemente una permutación de los elementos de la muestra original, el número de maneras distintas de ordenar n elementos únicos se calcula como:

- Para la primera posición, tenemos n opciones.
- Para la segunda, nos quedan $n - 1$ opciones.
- ... así hasta la última posición, para la que queda 1 opción.

El número total de secuencias favorables es el producto:

$$\text{Número de permutaciones} = n \times (n-1) \times \cdots \times 1 = n! \tag{3}$$

Ahora, para calcular la probabilidad sabemos que el muestreo bootstrap consiste en n extracciones independientes y con reemplazo de la muestra original. La probabilidad de seleccionar cualquier observación x_i en una única extracción es siempre $\frac{1}{n}$. Debido a la independencia, la probabilidad de obtener una secuencia ordenada específica de longitud n es el producto de las probabilidades de cada extracción:

$$P(\text{una secuencia específica}) = \underbrace{\frac{1}{n} \times \frac{1}{n} \times \cdots \times \frac{1}{n}}_{n \text{ veces}} = \left(\frac{1}{n}\right)^n = \frac{1}{n^n} \tag{4}$$

Es importante tener en mente que cualquier permutación de los datos originales, como $(x_1, x_2, \ldots, x_n)$ o $(x_2, x_1, \ldots, x_n)$, tiene exactamente la misma probabilidad de ocurrir. Entonces, la probabilidad total de un evento es la suma de las probabilidades de todos los resultados que lo componen. Dado que tenemos $n!$ secuencias favorables y cada una tiene una probabilidad de $1/n^n$:

$P(\text{réplica idéntica}) = (\text{Número de secuencias favorables}) \times (\text{Probabilidad de una secuencia específica})$

Por lo tanto, sustituyendo los resultados encontrados:

$$\boxed{P(\mathbf{N} = (1, \ldots, 1)) = n! \times \frac{1}{n^n} = \frac{n!}{n^n}}$$

c) La probabilidad de obtener un vector de cuentas específico $\mathbf{N} = (N_1, \ldots, N_n)$ se rige por la distribución multinomial:

$$P(\mathbf{N}) = \frac{n!}{\prod_{i=1}^{n} N_i!} \left(\frac{1}{n}\right)^n \tag{5}$$

Para encontrar la muestra más probable, debemos encontrar el vector $\mathbf{N}$ que maximiza esta expresión, sujeto a la restricción $\sum_{i=1}^{n} N_i = n$. Como los términos $n!$ y $(1/n)^n$ son constantes, maximizar $P(\mathbf{N})$ es equivalente a minimizar el producto de los factoriales en el denominador, es decir:

$$\min \left(\prod_{i=1}^{n} N_i! \right) \quad \text{sujeto a} \quad \sum_{i=1}^{n} N_i = n, \quad N_i \geq 0 \tag{6}$$

La función factorial $k!$ crece muy rápidamente, para mantener el producto $\prod N_i!$ lo más pequeño posible, debemos evitar que cualquiera de los N_i sea grande. La forma más eficiente de lograrlo es distribuir las n unidades de la manera más equitativa posible entre los n conteos. Por ejemplo, para $n = 4$, el producto $N_1!N_2!N_3!N_4!$ es mínimo para $\mathbf{N} = (1, 1, 1, 1)$, donde el producto es 1.

Supongamos que un vector de cuentas $\mathbf{N}$ no está perfectamente balanceado, es decir, existen al menos dos cuentas N_i y N_j tales que $N_i \geq N_j + 2$, creamos un nuevo vector $\mathbf{N}'$ moviendo una unidad de la cuenta más grande a la más pequeña: $N_i' = N_i - 1$ y $N_j' = N_j + 1$. Comparamos el producto de los factoriales de los términos que cambiaron:

$$\frac{N_i'!N_j'!}{N_i!N_j!} = \frac{(N_i - 1)!(N_j + 1)!}{N_i!N_j!} = \frac{N_j + 1}{N_i} \tag{7}$$

Dado que $N_i \geq N_j + 2 \implies N_i > N_j + 1$, el cociente es menor que 1, esto significa que el producto de factoriales para $\mathbf{N}'$ es más pequeño, y por ende, $\mathbf{N}'$ es más probable. Este proceso de "balanceo" puede repetirse hasta que no existan dos cuentas que difieran en más de 1, por lo tanto, para una suma de n sobre n elementos, la única configuración con esta propiedad es aquella en la que todos los conteos son 1:

$$\boxed{\mathbf{N}_{\text{óptimo}} = (1, 1, \ldots, 1)}$$

La muestra de bootstrap más probable es, por lo tanto, aquella que es una permutación de la muestra original, y su probabilidad es:

$$\boxed{P_{\max} = \frac{n!}{n^n}}$$

d) Buscamos el valor esperado (el promedio) del número de veces que una observación específica, digamos que x_i, aparece en una muestra bootstrap. Sea N_i la variable aleatoria que cuenta esta frecuencia, queremos calcular $E[N_i]$.

Consideremos cada una de las n extracciones como un ensayo de Bernoulli, un "éxito" en un ensayo ocurre si seleccionamos la observación x_i y tenemos que:

- El número de ensayos es n.
- Los ensayos son independientes.
- La probabilidad de éxito en cada ensayo es constante e igual a $p = 1/n$.

La variable aleatoria N_i, que cuenta el número total de éxitos en n ensayos, sigue por definición una distribución Binomial:

$$N_i \sim \text{Bin}\left(n, p = \frac{1}{n}\right) \tag{8}$$

El valor esperado de una variable aleatoria $\text{Bin}(n, p)$ es np, aplicando esto a nuestro caso:

$$E[N_i] = np = n\frac{1}{n} = 1 \tag{9}$$

Por lo tanto, en promedio, cada observación original aparece una vez en una réplica de bootstrap:

$$\boxed{E[N_i] = 1}$$

Problema 2

Sea $x_1, x_2, \ldots, x_n$ una muestra aleatoria de una normal $N(\theta, 1)$ y suponga que $\bar{x}$ es un estimador de θ. Sean $X_1^*, X_2^*, \ldots, X_n^*$ una muestra bootstrap de $N(\bar{x}, 1)$. Muestre que $\bar{X} - \theta$ y $\bar{X}^* - \bar{x}$ tienen la misma distribución $N(0, 1/n)$.

SOLUCIÓN

En esencia, buscamos demostrar que la distribución del error de muestreo en el "plano real" (la distancia entre el estimador y el parámetro verdadero) es idéntica a la distribución del error de muestreo en el "plano bootstrap" (la distancia entre el estimador bootstrap y el estimador original). Para ello, primero, necesitamos encontrar la distribución de la cantidad original, $\bar{X} - \theta$, y luego, encontrar la distribución de su análogo bootstrap, $\bar{X}^* - \bar{x}$.

Entonces, comenzando con la distribución de la cantidad original $\bar{X} - \theta$ supongamos que tenemos una muestra aleatoria $X_1, X_2, \ldots, X_n$ donde cada X_i es independiente y sigue una distribución Normal, es decir, $X_i \sim N(\theta, 1)$. Utilizando la propiedad fundamental de la distribución Normal que nos dice que la suma (y el promedio) de variables aleatorias normales independientes también es una variable aleatoria Normal podemos calcular la esperanza de $\bar{X}$:

$$E[\bar{X}] = E\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n}\sum_{i=1}^{n} E[X_i] \tag{10}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \theta = \frac{n\theta}{n} = \theta \tag{11}$$

Siendo la varianza de $\bar{X}$:

$$\mathrm{Var}(\bar{X}) = \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} \mathrm{Var}(X_i) \tag{12}$$

$$= \frac{1}{n^2}\sum_{i=1}^{n} 1 = \frac{n}{n^2} = \frac{1}{n} \tag{13}$$

Por lo tanto, la distribución del promedio muestral es $\bar{X} \sim N(\theta, 1/n)$, ahora podemos calcular la distribución de la diferencia $\bar{X} - \theta$, primero calculamos su esperanza:

$$E[\bar{X} - \theta] = E[\bar{X}] - \theta = \theta - \theta = 0 \tag{14}$$

Ahora la varianza (recordando que restar una constante no afecta la varianza):

$$\mathrm{Var}(\bar{X} - \theta) = \mathrm{Var}(\bar{X}) = \frac{1}{n} \tag{15}$$

Por lo tanto, la distribución del error de muestreo original es:

$$\boxed{\bar{X} - \theta \sim N(0, 1/n)}$$

Ahora replicamos el mismo proceso, pero en el "plano bootstrap", donde el parámetro "verdadero" es el estimador $\bar{x}$ que calculamos de la muestra original. Consideremos una muestra bootstrap $X_1^*, X_2^*, \ldots, X_n^*$ donde cada X_i^* es independiente y sigue una distribución Normal, es decir, $X_i^* \sim N(\bar{x}, 1)$. Aquí, $\bar{x}$ se trata como un valor fijo y conocido.

De manera análoga, primero calculamos la esperanza de $\bar{X}^*$:

$$E[\bar{X}^*] = E\left[\frac{1}{n}\sum_{i=1}^{n} X_i^*\right] = \frac{1}{n}\sum_{i=1}^{n} E[X_i^*] \tag{16}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \bar{x} = \frac{n\bar{x}}{n} = \bar{x} \tag{17}$$

Siendo la varianza:

$$\mathrm{Var}(\bar{X}^*) = \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i^*\right) = \frac{1}{n^2}\sum_{i=1}^{n} \mathrm{Var}(X_i^*)$$
$$= \frac{1}{n^2}\sum_{i=1}^{n} 1 = \frac{n}{n^2} = \frac{1}{n}$$

Por lo tanto, la distribución del promedio bootstrap es $\bar{X}^* \sim N(\bar{x}, 1/n)$. Finalmente, calculamos la distribución de la diferencia entre el estimador bootstrap y el estimador original, primero la esperanza:

$$E[\bar{X}^* - \bar{x}] = E[\bar{X}^*] - \bar{x} = \bar{x} - \bar{x} = 0 \tag{18}$$

Ahora la varianza:

$$\mathrm{Var}(\bar{X}^* - \bar{x}) = \mathrm{Var}(\bar{X}^*) = \frac{1}{n} \tag{19}$$

Por lo tanto, la distribución del error de muestreo bootstrap es:

$$\boxed{\bar{X}^* - \bar{x} \sim N(0, 1/n)}$$

Queda demostrado que tanto $\bar{X} - \theta$ como $\bar{X}^* - \bar{x}$ siguen exactamente la misma distribución, $N(0, 1/n)$. Esto nos indica que la relación entre la población y la muestra (cuantificada por la distribución de $\bar{X} - \theta$) es análoga a la relación entre la muestra y la muestra bootstrap (cuantificada por la distribución de $\bar{X}^* - \bar{x}$). Como no conocemos θ y no podemos observar directamente $\bar{X} - \theta$, usamos la distribución de $\bar{X}^* - \bar{x}$ que sí podemos simular y analizar, ya que conocemos $\bar{x}$ como una aproximación para hacer inferencias sobre θ.

Problema 3

Considere el conjunto de datos

$$2, 5, 3, 9.$$

Sean $X_1^*, X_2^*, X_3^*, X_4^*$ una muestra bootstrap de este conjunto de datos.

(a) Encuentre la probabilidad de que el promedio de la muestra bootstrap sea igual a 2.

(b) Encuentre la probabilidad de que el promedio de la muestra bootstrap sea igual a 9.

(c) Encuentre la probabilidad de que el promedio de la muestra bootstrap sea igual a 4.

SOLUCIÓN

Tenemos un conjunto de datos original con $n = 4$ elementos, sea:

$$D = \{2, 5, 3, 9\} \tag{20}$$

Una muestra bootstrap, $X_1^*, X_2^*, X_3^*, X_4^*$, se obtiene seleccionando 4 elementos de D con reemplazo. El espacio muestral consiste en todas las posibles muestras ordenadas, como hay 4 opciones para cada una de las 4 posiciones, el número total de muestras bootstrap posibles es:

$$\text{Total de Muestras Posibles} = 4^4 = 256 \tag{21}$$

Cada una de estas 256 muestras ordenadas es igualmente probable, con una probabilidad de 1/256. Buscamos encontrar la probabilidad de que el promedio bootstrap:

$$\bar{X}^* = \frac{1}{4}\sum_{i=1}^{4} X_i^* \tag{22}$$

tome ciertos valores, esto es equivalente a encontrar la probabilidad de que la suma de la muestra:

$$S^* = \sum_{i=1}^{4} X_i^* \tag{23}$$

tome valores específicos.

a) Primero buscamos la probabilidad de que el promedio sea 2, tenemos que:

$$\bar{X}^* = 2 \iff \frac{S^*}{4} = 2 \iff S^* = 8 \tag{24}$$

Entonces debemos encontrar todas las muestras bootstrap de 4 elementos de $D = \{2, 3, 5, 9\}$ cuya suma sea 8. Dado que el valor más pequeño en D es 2, la única manera de obtener una suma de 8 es seleccionando el 2 en las cuatro extracciones. Solo hay una muestra ordenada que corresponde a este multiconjunto: $(2, 2, 2, 2)$. Por lo tanto, solo hay 1 caso favorable. Calculamos la probabilidad:

$$P(\bar{X}^* = 2) = \frac{\text{Casos Favorables}}{\text{Total de Casos}} = \frac{1}{256} \tag{25}$$

Por lo tanto:

$$\boxed{P(\bar{X}^* = 2) = \frac{1}{256} \approx 0.0039}$$

b) Ahora buscamos la probabilidad de que el promedio sea 9, tenemos que:

$$\bar{X}^* = 9 \iff \frac{S^*}{4} = 9 \iff S^* = 36 \tag{26}$$

Ahora debemos encontrar las muestras cuya suma sea 36. Dado que el valor más grande en D es 9, la única manera de alcanzar una suma de 36 es seleccionando el 9 en las cuatro extracciones. Solo hay una muestra ordenada correspondiente: $(9, 9, 9, 9)$. Por lo tanto, solo hay 1 caso favorable. Calculamos la probabilidad:

$$\boxed{P(\bar{X}^* = 9) = \frac{1}{256} \approx 0.0039}$$

c) Por ultimo buscamos la probabilidad de que el promedio sea 4, tenemos que:

$$\bar{X}^* = 4 \iff \frac{S^*}{4} = 4 \iff S^* = 16 \tag{27}$$

Necesitamos encontrar todos los conjuntos de 4 números de D que sumen 16, siendo los multiconjuntos posibles:

- **Multiconjunto 1:** Si tomamos un 9, necesitamos que los otros tres números sumen $16 - 9 = 7$. La única combinación de tres números de D que suma 7 es $\{2, 2, 3\}$. Por lo tanto, obtenemos el multiconjunto $\{2, 2, 3, 9\}$.
- **Multiconjunto 2:** Ahora sin usar el 9, si tomamos dos 5, necesitamos que los otros dos números sumen $16 - 10 = 6$. La única combinación de dos números de D que suma 6 es $\{3, 3\}$. Por lo tanto, obtenemos el multiconjunto $\{3, 3, 5, 5\}$.

No existen otras combinaciones posibles. Para cada multiconjunto, debemos contar cuántas permutaciones distintas (muestras ordenadas) se pueden formar, donde la fórmula del coeficiente multinomial tine la forma:

$$\frac{n!}{n_1! n_2! \dots n_k!} \tag{28}$$

- Para $\{2, 2, 3, 9\}$: Tenemos cuatro elementos con un '2' repetido dos veces.

$$\text{Permutaciones} = \frac{4!}{2!\ 1!\ 1!} = \frac{24}{2} = 12$$

- Para $\{3, 3, 5, 5\}$: Tenemos cuatro elementos con un '3' repetido dos veces y un '5' repetido dos veces.

$$\text{Permutaciones} = \frac{4!}{2!\ 2!} = \frac{24}{4} = 6$$

El número total de casos favorables es la suma de las permutaciones de ambos multiconjuntos es de $12 + 6 = 18$. Por lo tanto la probabilidad de que el promedio sea 4 es:

$$\boxed{P(\bar{X}^* = 4) = \frac{9}{128} \approx 0.0703}$$

Problema 4

Maximice las siguientes 2 funciones utilizando el algoritmo de recocido simulado.

a) Función 1

$$f(x, y, \alpha, \beta) = \frac{\sin^2[(x+\alpha)^2 + (y+\beta)^2] - 0.5}{[1.0 + 0.001 \cdot ((x+\alpha)^2 + (y+\beta)^2)]^2}$$

con las restricciones:

$$-100 \leq x \leq 100$$
$$-100 \leq y \leq 100$$
$$-\infty \leq \alpha \leq \infty$$
$$-\infty \leq \beta \leq \infty$$

b) Función 2

$$f(x, y) = 21.5 + x\sin(4\pi x) + y\sin(20\pi y)$$

con las restricciones:

$$-3.0 \leq x \leq 12.1$$
$$4.1 \leq y \leq 5.8$$

SOLUCIÓN

Se utilizo el lenguaje de programación `R` *para resolver este problema, el código utilizado se encuentran al final del documento en el apéndice* **A.1**

Sabemos que el Recocido Simulado (SA) es un método de optimización estocástica diseñado para localizar el óptimo global de una función en un espacio de búsqueda extenso. Para un problema de maximización de una función objetivo $f(\boldsymbol{z})$, el algoritmo explora el espacio de soluciones partiendo de un estado inicial $\boldsymbol{z}_0$. En cada iteración, se genera un nuevo estado candidato $\boldsymbol{z}'$ a partir del estado actual $\boldsymbol{z}$ mediante un mecanismo de propuesta $q(\boldsymbol{z}'|\boldsymbol{z})$. La aceptación del nuevo estado se rige por la probabilidad de Metropolis, la cual depende de la mejora en la función objetivo, $\Delta f = f(\boldsymbol{z}') - f(\boldsymbol{z})$, y de un parámetro de control T, análogo a la temperatura:

$$\alpha(\boldsymbol{z}, \boldsymbol{z}'; T) = \min\left\{1, \exp\left(\frac{\Delta f}{T}\right)\right\} \tag{29}$$

Si $f(\boldsymbol{z}') > f(\boldsymbol{z})$, el nuevo estado se acepta incondicionalmente. Si $f(\boldsymbol{z}') \leq f(\boldsymbol{z})$, el estado puede ser aceptado con una probabilidad $e^{\Delta f/T}$, permitiendo al algoritmo escapar de óptimos locales. La temperatura T se reduce gradualmente mediante un esquema de enfriamiento, típicamente geométrico:

$$T_k = T_0\gamma^k \tag{30}$$

donde T_k es la temperatura en la iteración k, T_0 es la temperatura inicial, y $\gamma \in (0, 1)$ es el factor de enfriamiento.

a) Comenzando con la función 1, sea esta f_1, observamos que su estructura depende únicamente de la suma de cuadrados $(x+\alpha)^2 + (y+\beta)^2$. Mediante las sustituciones $u = x + \alpha$ y $v = y + \beta$, y definiendo $s = u^2 + v^2 \geq 0$, el problema se reduce a maximizar una función de una sola variable s:

$$\phi(s) = \frac{\sin^2(s) - \frac{1}{2}}{\left(1 + 0.001s\right)^2} \tag{31}$$

La condición de optimalidad de primer orden, $\phi'(s) = 0$, para un punto crítico interior $s^* > 0$ se expresa como:

$$\frac{\sin(2s^*)}{\sin^2(s^*) - 1/2} = \frac{0.002}{1 + 0.001s^*} \tag{32}$$

El numerador de $\phi(s)$ se maximiza en $s = \pi/2$, donde $\sin^2(s) = 1$. Sin embargo, en este punto, el denominador introduce una penalización. El análisis de la derivada $\phi'(s)$ revela que $\phi'(\pi/2) < 0$, indicando que el máximo global se encuentra en un punto s^* ligeramente a la izquierda de $\pi/2$.

Se aplicó SA con 8 arranques, $\gamma = 0.985$, $T_{\min} = 10^{-8}$ y 30,000 iteraciones por arranque. El algoritmo convergió a la solución:

$$\boxed{\boldsymbol{z}^* = (x^*, y^*, \alpha^*, \beta^*) \approx (1.8128, -0.8385, -0.6611, 0.3446)}$$

El valor óptimo y el radio correspondiente fueron:

$$\boxed{f_1^* \approx \mathbf{0.4984331455}, \quad \text{con } s^* = (x^* + \alpha^*)^2 + (y^* + \beta^*)^2 \approx \mathbf{1.5702971}}$$

Este resultado es consistente con la predicción analítica, ya que s^* es ligeramente menor que $\pi/2 \approx 1.5707963$.

b) Ahora para la funcion 2, sea esta f_2, observamos que es separable, es decir:

$$f_2(x, y) = 21.5 + g(x) + h(y) \tag{33}$$

Por lo tanto, su maximización puede descomponerse en la maximización independiente de:

$$g(x) = x\sin(4\pi x) \quad \text{y} h(y) = y\sin(20\pi y) \tag{34}$$

Las condiciones de primer orden $g'(x) = 0$ y $h'(y) = 0$ conducen a las siguientes ecuaciones trascendentales:

$$\tan(4\pi x) = -4\pi x \quad \text{y} \quad \tan(20\pi y) = -20\pi y \tag{35}$$

Asi el máximo global de f_2 corresponde a la combinación de las soluciones de estas ecuaciones dentro del dominio especificado que maximicen los valores de $g(x)$ y $h(y)$.

Se aplicó SA con 16 arranques desde puntos iniciales uniformemente distribuidos en el dominio, con $\gamma = 0.988$, $T_{\min} = 10^{-8}$ y 40,000 iteraciones por arranque. La solución encontrada fue:

$$\boxed{(x^*, y^*) \approx (11.62554, 5.72504)}$$

con un valor óptimo de:

$$\boxed{f_2^* \approx \mathbf{38.8502944794}}$$

La solución numérica se encuentra en el interior del dominio y es consistente con las soluciones de las ecuaciones de primer orden que corresponden a los picos más altos de las funciones $g(x)$ y $h(y)$ en sus respectivos intervalos.

A Código en R

A continuación, se presentan los scripts de R utilizados para cada uno de los problemas resueltos en este reporte.

A.1 Problema 2

```r
# ============================================================================
# Problema 4
# Script 1: Funciones a maximizar
# Autor: Diego Paniagua Molina
# Fecha: 2025-10-07
# ============================================================================


# Problema (a): f1(x, y, alpha, beta)-----------------------------------------
f1 <- function(z) {
  x <- z[1]; y <- z[2]; a <- z[3]; b <- z[4]
  s <- (x + a)^2 + (y + b)^2
  (sin(s)^2 - 0.5) / (1.0 + 0.001 * s)^2
}

# Restringe sólo x,y a [-100,100]; alpha,beta libres
clip_f1 <- function(z) {
  z <- as.numeric(z)
  z[1] <- min(100, max(-100, z[1]))
  z[2] <- min(100, max(-100, z[2]))
  z
}

solve_f1_multistart <- function(
    n_starts = 8, seed = 123,
    max_iters = 30000, gamma = 0.985, T_min = 1e-8
){
  best <- NULL
  for (i in 0:(n_starts - 1L)) {
    out <- simulated_annealing(
      f = f1, x0 = c(0,0,0,0),
      propose = propose_gaussian, clip = clip_f1,
      max_iters = max_iters, seed = seed + i,
      gamma = gamma, T0 = NULL, T_min = T_min
    )
    if (is.null(best) || out$best_f > best$best_f) best <- out
  }
  best
}

# Referencia analítica útil en s = pi/2
f1_reference <- function() {
  (sin(pi/2)^2 - 0.5) / (1 + 0.001 * (pi/2))^2
}

# Problema (b): f2(x, y) ------------------------------------------------------
f2 <- function(z) {
  x <- z[1]; y <- z[2]
  21.5 + x * sin(4 * pi * x) + y * sin(20 * pi * y)
}

# Proyección a la caja del dominio
clip_f2 <- function(z) {
  z <- as.numeric(z)
  z[1] <- min(12.1, max(-3.0, z[1]))
  z[2] <- min(5.8,  max(4.1, z[2]))
  z
}
```

```r

solve_f2_multistart <- function(
    n_starts = 16, seed = 456,
    max_iters = 40000, gamma = 0.988, T_min = 1e-8
){
  set.seed(seed)
  best <- NULL
  for (i in 1:n_starts) {
    x0 <- c(runif(1, -3.0, 12.1), runif(1, 4.1, 5.8))
    out <- simulated_annealing(
      f = f2, x0 = x0,
      propose = propose_gaussian, clip = clip_f2,
      max_iters = max_iters, seed = sample.int(1e6, 1),
      gamma = gamma, T0 = NULL, T_min = T_min
    )
    if (is.null(best) || out$best_f > best$best_f) best <- out
  }
  best
}
```

Listing 1: Script: Definicion de funciones a maximizar

```r
# ============================================================================
# Problema 4
# Script 2: Algoritmo de Recocido Simulado
# ============================================================================

simulated_annealing <- function(
    f, x0, propose, clip = NULL,
    max_iters = 20000, seed = 42,
    target_accept = c(0.25, 0.55),
    gamma = 0.98, T0 = NULL, T_min = 1e-6,
    window = 200
){
  stopifnot(is.numeric(x0))
  set.seed(seed)

  x  <- as.numeric(x0)
  if (!is.null(clip)) x <- clip(x)
  fx <- f(x)
  best_x <- x; best_f <- fx

  sigma <- rep(0.5, length(x))  # escala inicial por componente

  # Estima T0 con variaciones típicas de f cerca de x0
  if (is.null(T0)) {
    deltas <- numeric(64)
    for (i in seq_along(deltas)) {
      x_try <- propose(x, 1.0, sigma)
      if (!is.null(clip)) x_try <- clip(x_try)
      deltas[i] <- abs(f(x_try) - fx)
    }
    T0 <- max(stats::median(deltas), 1e-3)
  }
  T <- T0

  accept <- 0L; tried <- 0L

  # Historial del mejor valor (adelgazado cada 1000 pasos)
  hist_n  <- floor(max_iters/1000) + 1L
  history <- matrix(NA_real_, nrow = hist_n, ncol = 3L)
  hidx    <- 1L

  for (k in 0:(max_iters - 1L)) {
    # Propuesta y evaluación
    x_new <- propose(x, T, sigma)
    if (!is.null(clip)) x_new <- clip(x_new)
    f_new <- f(x_new); d <- f_new - fx

    # Regla de aceptación (Metrópolis para maximización)
    if (d >= 0 || runif(1) < exp(d / max(T, 1e-12))) {
      x <- x_new; fx <- f_new; accept <- accept + 1L
      if (f_new > best_f) { best_f <- f_new; best_x <- x_new }
    }
    tried <- tried + 1L

    # Enfriamiento
    T <- max(T * gamma, T_min)

    # Guardar historial
    if (k %% 1000L == 0L) {
      history[hidx, ] <- c(k, fx, best_f); hidx <- hidx + 1L
    }

    # Adaptar escala de propuesta
    if ((k + 1L) %% window == 0L) {
      acc <- accept / max(tried, 1L)
      if (acc < target_accept[1]) sigma <- sigma * 0.8
      else if (acc > target_accept[2]) sigma <- sigma * 1.2
```

```
      accept <- 0L; tried <- 0L
    }
  }

  history <- history[!is.na(history[,1L]), , drop = FALSE]
  colnames(history) <- c("iter", "f_actual", "f_mejor")

  list(best_x = best_x, best_f = best_f,
       final_sigma = sigma, history = history, T0 = T0)
}

# Propuesta gaussiana simétrica por componente
propose_gaussian <- function(x, T, sigma) {
  x + rnorm(length(x)) * sigma
}
```

Listing 2: Script: Algoritmo de Recocido Simulado

```r
# ============================================================================
# Problema 4
# Script 3: Ejecuta SA para f1
# ============================================================================

# Detecta la carpeta scripts/ para resolver rutas absolutas
.this_file <- function() {
  if (!is.null(sys.frames()[[1]]$ofile)) return(normalizePath(sys.frames()[[1]]$ofile))
  normalizePath("scripts/run_f1.R")
}
dir_scripts <- dirname(.this_file())
root        <- normalizePath(file.path(dir_scripts, ".."))

# Cargar dependencias locales
source(file.path(dir_scripts, "sa_core.R"))
source(file.path(dir_scripts, "problems.R"))

# Referencia analítica (s = pi/2)
ref_f1 <- f1_reference()

# Ejecutar SA (multi-arranque)
best1 <- solve_f1_multistart(
  n_starts = 8, seed = 123, max_iters = 30000, gamma = 0.985, T_min = 1e-8
)

# Reporte en consola
cat(sprintf("Referencia f1*  %.10f\n", ref_f1))
cat(sprintf("SA f1*  %.10f\n", best1$best_f))
bx <- best1$best_x
cat(sprintf("Argmax aprox (x,y,alpha,beta) = (%.10f, %.10f, %.10f, %.10f)\n",
            bx[1], bx[2], bx[3], bx[4]))

# Guardado (CSV + RDS)
out_dir <- file.path(root, "results", "f1")
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)
utils::write.csv(best1$history, file.path(out_dir, "f1_history.csv"), row.names = FALSE)
saveRDS(best1, file.path(out_dir, "f1_best.rds"))
```

Listing 3: Script: Ejecuta SA para f1

```r
# ============================================================================
# Problema 4
# Script 4: Ejecuta SA para f2
# ============================================================================

# Detecta la carpeta scripts/ para resolver rutas absolutas
.this_file <- function() {
  if (!is.null(sys.frames()[[1]]$ofile)) return(normalizePath(sys.frames()[[1]]$ofile))
  normalizePath("scripts/run_f2.R")
}
dir_scripts <- dirname(.this_file())
root        <- normalizePath(file.path(dir_scripts, ".."))

# Cargar dependencias locales
source(file.path(dir_scripts, "sa_core.R"))
source(file.path(dir_scripts, "problems.R"))

# Ejecutar SA (multi-arranque)
best2 <- solve_f2_multistart(
  n_starts = 16, seed = 456, max_iters = 40000, gamma = 0.988, T_min = 1e-8
)

# Reporte en consola
cat(sprintf("SA f2*  %.10f\n", best2$best_f))
bx <- best2$best_x
cat(sprintf("Argmax aprox (x,y) = (%.10f, %.10f)\n", bx[1], bx[2]))

# Guardado (CSV + RDS)
out_dir <- file.path(root, "results", "f2")
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)
utils::write.csv(best2$history, file.path(out_dir, "f2_history.csv"), row.names = FALSE)
saveRDS(best2, file.path(out_dir, "f2_best.rds"))
```

Listing 4: Script: Ejecuta SA para f2