Hoja de Prácticas: Codificación

Álgebra aplicada

Enunciados

Índice de Contenidos

Instrucciones

Implementa las soluciones en R utilizando las funciones de apoyo proporcionadas en el material de clase o diseñando tus propias funciones. Asegúrate de justificar matemáticamente cada resultado.

Ejercicios Teóricos

Ejercicio 1: Homomorfismo de Codificación

Demuestra formalmente que la función de codificación $\mathcal{C}_{\mathcal{G}}(\mathbf{x}) = \mathbf{x}\mathcal{G}$ definida por una matriz G de dimensiones $m \times n$ es un homomorfismo de grupos de $(\mathbb{Z}_2^m, +)$ en $(\mathbb{Z}_2^n, +)$.

Ejercicio 2: Distancia Mínima

Calcula la distancia mínima del código $C = \{0000, 0101, 1011, 1110\}$. Determina cuántos errores puede detectar y corregir este código.

Ejercicio 3: Matriz de Verificación de Paridad

Para la matriz generadora $G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$:

a) Calcula la matriz de verificación de paridad H.

b) Verifica que $GH = 0$ (módulo 2).

c) Comprueba que cada palabra código del código asociado satisface $\mathbf{c}H = \mathbf{0}$.

Ejercicio 4: Decodificación por Clases Laterales

Para el código con matriz generadora $G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$:

a) Construye la tabla completa de clases laterales.

b) Para cada clase lateral, identifica el representante de peso mínimo (líder de clase).

c) Utiliza la tabla de clases laterales para decodificar las siguientes palabras recibidas: $0111, 1001, 1100, 0010$.

d) Para cada palabra decodificada, extrae el mensaje original.

Ejercicios de Implementación en R

Para los siguientes ejercicios, puedes usar estas funciones base como punto de partida:

```r
# Funciones auxiliares (disponibles en la teoría)
generar_H <- function(G) {
  k <- nrow(G); n <- ncol(G)
  P <- G[, (k + 1):n]
  rbind(P, diag(1, n - k))
}

codificar <- function(mensaje, G) {
  (mensaje %*% G) %% 2
}

calcular_sindrome <- function(r, H) {
  (r %*% H) %% 2
}
```

Ejercicio 5: Matriz Generadora Sistemática

Diseña una función en R llamada `convertir_a_sistematica(G)` que, utilizando operaciones elementales por filas módulo 2, devuelva la forma sistemática $[I_k \mid P]$ de una matriz generadora dada.

Ejercicio 6: Capacidad del Código

Dada la matriz generadora:

$$\mathcal{G} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

a) Genera todas las palabras código del código asociado.

b) Determina la distancia mínima del código.

c) Calcula cuántos errores puede detectar y cuántos puede corregir.

Ejercicio 7: Síndromes y Decodificación

Utilizando la matriz $\mathcal{G}$ del ejercicio anterior:

a) Construye la tabla de síndromes y sus representantes de clase lateral (líderes de clase) de peso menor o igual a 1.

b) Si recibimos la palabra $\mathbf{r} = (1, 1, 1, 1, 0)$, calcula su síndrome y recupera el mensaje original.

c) Si recibimos la palabra $\mathbf{r} = (0, 1, 1, 0, 1)$, ¿qué ocurre? Interpreta el resultado.

Ejercicio 8: Análisis Automático de Códigos

Implementa una función en R que, dada una matriz generadora G en forma sistemática, devuelva el número de errores que detecta y que es capaz de corregir el código lineal asociado. Es posible que necesites funciones auxiliares, que tendrás que definir.

Ejercicio 9: Caso de Error Múltiple

Simula (mediante código) el envío de un mensaje de 4 bits utilizando un código $(7, 4)$. Introduce un error de 2 bits en la transmisión.

a) ¿Detecta el código que ha ocurrido un error?

b) ¿Es capaz de corregirlo correctamente? Justifica tu respuesta basándote en la distancia mínima.

Ejercicio Final: Proyecto Completo

Ejercicio 10: Diseño de un Sistema de Comunicación

1. Implementa en R las funciones `convertir_a_sistematica` y `extraer_mensaje` (si no lo has hecho ya).
2. Inventa una matriz generadora G en forma sistemática que proporcione un código $\mathcal{C}: \mathbb{Z}_2^4 \to \mathbb{Z}_2^7$ con capacidad para detectar dos errores y corregir uno.
3. Codifica los mensajes $\mathbf{m}_1 = (1,0,1,1)$ y $\mathbf{m}_2 = (0,1,0,0)$ utilizando la matriz generadora G obtenida.
4. Simula la transmisión de las palabras código obtenidas, introduciendo un error en una posición aleatoria de cada palabra.
5. Decodifica las palabras recibidas utilizando la decodificación por síndrome implementada. Verifica si se corrigen los errores.