Tema 1. Teoría de Códigos

Justo Peralta López

Juan Antonio López Ramos
Dpto. Álgebra y Análisis Matemático

Resumen: *En este tema, introducimos los parámetros más importantes de un código, cómo son la longitud de las palabras del código, el número de palabras del código y la distancia mínima. En cuanto esta última, es fundamental que el alumno comprenda la relación de la distancia mínima con la capacidad de detección y corrección del código, y como ésta dependerán también del tipo de canal en el cual nos encontremos. Finalmente daremos una clasificación de los códigos que vamos a estudiar en los sucesivos temas y veremos cómo se pueden obtener códigos a partir de otros.*

2.1. Código Bloque y Distancia Mínima

Definición 2.1.1. *Si A es un alfabeto, una palabra de longitud n es una secuencia de símbolos de dicho alfabeto. Al conjunto de todas las palabras de longitud n sobre ese alfabeto lo denotamos por A^n.*

Si A está formado por q símbolos, entonces podemos escoger entre q posibilidades en cada posición de la palabra a formar. Por lo tanto, el número total de palabras de longitud n será q^n.

Definición 2.1.2. *Un (n, M)-código bloque C sobre un alfabeto A, es un subconjunto C de A^n, con $|C| = M$.*

Normalmente, hablaremos de n como la longitud de C o longitud de palabra de C y de M como el tamaño o el número de palabras código de C.

Definición 2.1.3. *El peso Hamming de una palabra del código, $x = (x_1, x_2, \ldots, x_n)$, al cual denotaremos por $w(x)$,es el número de componentes distintos de cero.*

Definición 2.1.4. *La distancia Hamming entre dos palabras del código, x e y, a la cual denotaremos por $d(x, y)$, es el número de posiciones en que dichas palabras difieren. Es decir*

$$d(x, y) = w(x - y) = w(y - x)$$

Ejemplo 2.1.1. 1. *Sea* $x = (10011)$ *e* $y = (01010)$ *sobre* $GF(2)$. *Entonces* $w(x) = 3$, $w(y) = 2$ *y* $d(x, y) = 3$.

2. *Sea* $x = (20120)$ *e* $y = (10221)$ *sobre* $GF(3)$. *Entonces* $w(x) = 3$, $w(y) = 4$ *y* $d(x, y) = 3 = w(x - y) = w(10202)$

Lema 2.1.1. *La distancia Hamming es una métrica, es decir, verifica la siguientes propiedades: Para cualquier* x, y, z

1. $d(x, y)$ *es un número real no negativo.*
2. *Si* $d(x, y) = 0$, *si y solo si* $x = y$.
3. *d(x,y)=d(y,x).*
4. $d(x, y) \leq d(x, z) + d(z, y)$.

A esta última propiedad le llamamos la desigualdad triangular. Obsérvese que llamar a $d(x, y)$, distancia entre x e y, es muy apropiado ya que la medida en que se diferencia dos palabras se puede interpretar como la distancia entre x e y, es decir, cuanta más diferencias haya entre x e y, más alejadas se encontrará una palabra de otra.Intuitivamente, lo que nos dice la desigualdad triangular es que la distancia entre dos palabras o puntos, es siempre menor que la distancia si pasamos de x a y por un punto intermedio. Veamos la demostración del lema anterior de forma más rigurosa.

Demostración. Las propiedades de 1) a 3) se verifican por la definición de la distancia entre dos palabra. Demostremos pues la desigualdad triangular. Sean $x = (x_1, x_2, \ldots, x_n)$, $y = (y_1, y_2, \ldots, y_n)$ y $z = (z_1, z_2, \ldots, z_n)$. Entonces $d(x, z)$ nos mide el número de posiciones en los cuales difieren x y z.Notaremos ese conjunto por U, es decir,

$$d(x.z) = |U| = |\{i|x_i \neq z_i\}|$$

Sea $S = \{i|x_i \neq z_i \text{ y } x_i = y_i\}$ y $T = \{i|x_i \neq z_i \text{ y } x_i \neq y_i\}$. Entonces U será la unión disjunta de S y T. Luego

$$d(x, y) = |S| + |T|$$

Por definición de $d(x, y)$

$$|T| \leq d(x, y)$$

Por otro lado, si $i \in S$, entonces $y_i = x_i \neq z_i$. Por lo tanto

$$|S| \leq d(y, z)$$

□

Definición 2.1.5. *La distancia mínima,* d_{min}, *de un código* C *es la mínima de las distancias entre todos los pares de las palabras del código.*

A partir de ahora, un (n, M, d)-código representa un (n, M)-código bloque con mínima distancia d.

A continuación, veremos que relación hay entre la distancia mínima de un código y la capacidad de detección y corrección de dicho código. Para ello, debemos recordar que la probabilidad de

error p, se considera mucho menor que $1/2$. Esto quiere decir que la probabilidad de que se produzca muchos errores en una palabra, es mucho menor que la probabilidad de que se produzca pocos errores. La decodificación que se realiza, sea cual sea el algoritmo, siempre decodifica, dada una palabra recibida, por la palabra que más se le parece. Esto se realiza siempre de la misma forma, ya que la palabra que más se le parece es la que más probablemente ha sido enviada, ya que estamos considerando que el error cometido es el más pequeño. Por lo tanto, si recibimos una palabra del código, siempre consideraremos que la palabra llega correctamente, ya que la probabilidad de que no se haya producido ningún error, es mayor que la probabilidad de que se haya cometido alguno.

Teorema 2.1.1. *Es necesario y suficiente que la distancia mínima de un código sea mayor o igual que d, para poder detectar $d-1$ errores o menos.*

Demostración. Ya que la distancia mínima es d, cualquier dos palabras tienen como mínimo d posiciones diferentes. Si recibo una palabra, y esta pertenece al código lo aceptamos como buena, en caso contrario, como sólo se envían palabras del código, consideramos que se ha producido algún error. Luego, no seremos capaces de detectar los errores cometidos, cuando se produzcan tantos que la palabra de código enviada es modificada en otra palabra del código. Por lo tanto, siempre que se produzcan $d-1$ o menos errores, una palabra del código no se podrá transformar en otra palabra del código. □

Teorema 2.1.2. *Un código C puede corregir t errores o menos si y solo si $d_{min} \geq 2t+1$.*

Demostración. Supongamos que recibimos una palabra y con t' errores y $t' \leq t$. Como ya hemos comentado, decodificamos por la palabra del código que más se le parezca, es decir, por aquella palabra del código, x, con distancia con y menor.

Veamos que si x es la palabra enviada e y la recibida, y ésta última con menos de t errores, entonces no existe otra palabra del código, x', con $d(x',y) \leq t$. Si esto es cierto, la palabra que más se le parece a y será x, ya que no hay otra palabra del código con distancia con y menor que t. Supongamos que si lo hubiera, y llamémosla x'. Entonces $d(x,y) \leq t$ y $d(x',y) \leq t$. Pero por la desigualdad triangular

$$d(x,x') \leq d(x,y) + d(x',y) \leq 2t$$

lo cual contradice la hipótesis ya que la distancia mínima del código era mayor o igual a $2t+1$.

□

Hasta ahora, los teoremas anteriormente descritos, nos da una medida de la capacidad de detección cuando nuestro código sólo es utilizado para la detección de errores, o la capacidad de corrección cuando sólo se utiliza para la corrección de errores. El siguiente teorema, nos muestra la capacidad de corrección y detección cuando se utilizan estrategias mixtas.

Teorema 2.1.3. *Un código C puede corregir cualquier combinación de t errores y detectar d con $d \geq t$, si y solo si $d_{min} \geq t+d+1$*

Demostración. Para la demostración de este teorema, utilizaremos las esferas de radio t de la figura 2.1.

- Si se produce menos de t errores, la palabra y recibida, caerá dentro de una esfera y decodificaremos por su centro.
- Si se produce más de t errores, pero menos de d, la palabra recibida caerá entre dos esferas y podremos concluir que se ha producido algún error, pero no seremos capaces de corregirlo.

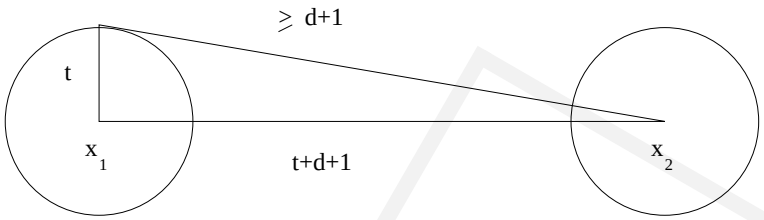

Figura 2.1: Esferas de radio t

- Si se produce más de d errores, entonces la palabra recibida y caerá en una esfera que no es la suya y decodificaremos de forma incorrecta.

Debemos observar que dos esferas siembre son disjuntas, ya que en caso contrario, dos palabras tendrían distancia $2t$ o menor, y la distancia mínima verifica por hipótesis que es mayor que $2t + 1$. □

Ejemplo 2.1.2. *1. Sea $C = \{00, 11\}$. La distancia mínima es 2. Como se puede observar, si recibimos la palabra 01 o 10, podemos detectar el error, pero no corregirlo. (Ver figura 2.2).*

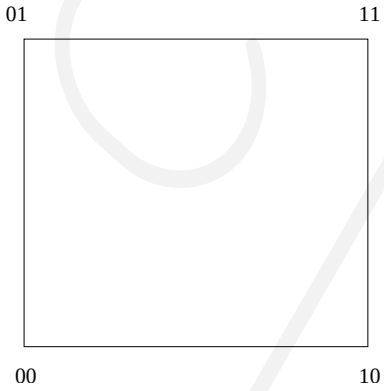

Figura 2.2: Ejemplo $C = \{00, 11\}$

2. $C = \{000, 111\}$ *Si un error ocurre, podemos corregir por aquella palabra del código más cercana. ($000 \to 010 \to 000$). Sin embargo, si dos errores ocurren ($000 \to 110 \to 111$), decodificamos por la más cercana de forma incorrecta. Es decir, en este caso, somos capaces de detectar y corregir un sólo error.*

   La otra posibilidad es usar el código sólo para detectar, en cuyo caso una palabra no contiene ningún error si y solo si pertenece al código, luego sólo podemos detectar dos errores.(Ver figura 2.3).

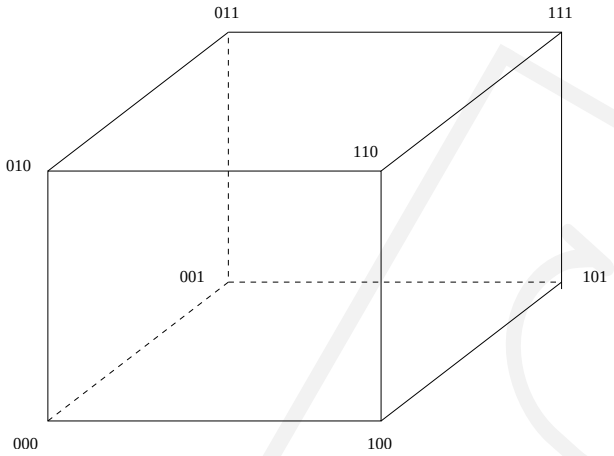

Figura 2.3: Ejemplo $C = \{000, 111\}$

3. $C = \{000000, 111111\}$ *En este caso la distancia mínima es 6 y tenemos las siguientes posibilidades.*

Casos	d	t
1	3	2
2	4	1
3	5	0

Obsérvese, que la capacidad de corrección también depende del tipo de canal en que nos encontramos. Hasta ahora sólo hemos considerado canales del tipo simétrico. Veamos que ocurre en el canal BEC.

En este caso, las posiciones donde se produce un error son conocidos. Esto es debido a que cuando se produce un error, un 1 no se cambia por un 0 o un 1 por un 0, sino que nos llega algo diferente a cualquier símbolo del alfabeto.

Ejemplo 2.1.3. $C = \{000, 111\}$ *La decodificación viene dada por la siguiente tabla*

000	111
$x00$	$x11$
$0x0$	$1x1$
$00x$	$11x$
$xx0$	$xx1$
$x0x$	$1x1$
$0xx$	$1xx$

Como se puede observar, si la palabra recibida contiene un 0 decodificamos por 000, y si contiene un 1, decodificamos por 111. Luego, es posible corregir 2 errores. Si solo utilizamos el código C para detectar, entonces podemos detectar todos los errores. En general, podemos afirmar el siguiente teorema.

Teorema 2.1.4. *Si C es un código con distancia mínima d en un BEC. Entonces podemos corregir $d - 1$ errores (si sólo se utiliza para corregir), y detectar d errores (si sólo se utiliza para detectar).*

2.2. Relación entre la mínima distancia y la probabilidad de error

Si utilizamos un código bloque de longitud n sobre un canal simétrico binario (BSC), donde la probabilidad de que se produzca un error es p. Entonces, la probabilidad de que se produzcan un error de peso k (o en k posiciones), es $p^k(1-p)^{n-k}$. Mientras que la probabilidad de que se produzca algún error de peso k, será $\binom{n}{k}p^k(1-p)^{n-k}$

2.3. Algunas clasificaciones de códigos

2.3.1. Códigos óptimos

Teorema 2.3.1. *Un q-código (no necesariamente lineal) con parámetros $(n, M, 2t+1)$ satisface la siguiente ecuación:*

$$M[\sum_{i=0}^{t}\binom{n}{i}(q-1)^i] \leq q^n$$

Si se verifica la igualdad entonces decimos que el código es perfecto

2.3.2. Códigos sistemáticos

Definición 2.3.1. *Un (n, q^k)-código es llamado sistemático, si existen k posiciones de las palabras del código, $i_1, i_2, \ldots, i_n$, tal que observando esas posiciones en todas las palabras del código, obtenemos todas las q^k posibles palabras, de longitud k sobre el alfabeto de q elementos.*

Ejemplo 2.3.1.

1. *Para $C = \{0000, 0110, 1001, 1010\}$, si seleccionamos la primera y la tercera posición en todas las palabras del código, obtenemos el conjunto $\{00, 01, 10, 11\}$, es decir, todas las posibles palabras de longitud 2 sobre el alfabeto binario.*

2. *Para el caso $C = \{000, 100, 010, 001\}$, el código no es sistemático.*

2.3.3. Códigos equivalentes

Definición 2.3.2. *Decimos que dos (n, M)-códigos, C_1 y C_2, son equivalentes, si existe una permutación σ de las coordenadas o posiciones de las palabras del código, y permutaciones $\pi_1, \pi_2, \ldots, \pi_n$ de los símbolos del alfabeto tal que $c_1c_2\ldots c_n \in C_1$ si y sólo si $\pi_1(c_{\sigma(1)})\pi_2(c_{\sigma(2)})\ldots\pi_n(c_{\sigma(n)}) \in C_2$*

Es decir, dos códigos son equivalentes si puedo transformar uno en otro aplicando permutaciones sobre sus posiciones y cambiando los símbolos del alfabeto.

Definición 2.3.3. *Dos (n, M)-códigos, C_1 y C_2, sobre $GF(q)$, se dicen que son equivalentes múltiplo por un escalar, si podemos obtener uno a partir del otro multiplicando los símbolos del alfabeto por un escalar.*

2.3.4. Códigos de repetición

Definición 2.3.4. *Un r-código C se dice que es de repetición si todas las posiciones de todas las palabras códigos tienen el mismo símbolo.*

Ejemplo 2.3.2. *Sea C es siguiente q-código.*

$$C = \{00\ldots0, 11\ldots1, \ldots, (q-1)(q-1)\ldots(q-1)\}$$

Obsérvese, que la razón del código es $R = 1/n$, donde n es la longitud de palabra del código.

2.3.5. Códigos lineales

Definición 2.3.5. *Un r-código C se dice que es lineal, si para toda $x, y \in C$, $x + y \in C$.*

Si C es un código lineal, lo notaremos por un (n, k, d)-código lineal donde n es la longitud de palabra del código, k la dimensión del código y d su distancia mínima.

Definición 2.3.6. *Si $x = x_1x_2\ldots X_n$ e $y = y_1y_2\ldots y_n$ son palabras de un código binario, entonces definimos la intersección de x con y por*

$$x \wedge y = (x_1y_1, x_2y_2, \ldots, x_ny_n)$$

Lema 2.3.1. *Para todo $x, y \in V(n, 2)$,*

$$d(x, y) = w(x) + w(y) - 2w(x \wedge y)$$

2.3.6. Códigos cíclicos

Definición 2.3.7. *Un r-código C es cíclico si es lineal y si para toda palabra $c = (c_1, c_1, \ldots, c_n) \in C$, entonces $c_n, c_1, \ldots, c_{n-1}) \in C$.*

Durante los sucesivos temas, veremos los códigos cíclicos más interesantes.

2.4. Obtención de nuevos códigos a partir de otros ya existentes

2.4.1. Por extensión

Si C es un (n, M, d)-código sobre el alfabeto $GF(q)$, el código extendido se obtiene como sigue

$$\hat{C} = \{c_1c_2\ldots c_nc_{n+1} | c_1c_2\ldots c_n \in C \text{y} \sum_{k=1}^{n+1} c_k = 0\ mod\ q\}$$

Teorema 2.4.1. *Si C es un (n, M, d)-código, entonces $\hat{C}$ es un $(n + 1, M, d$ o $d + 1)$-código.*

Ejemplo 2.4.1. *Sea $C = \{00, 01, 10, 11\}$, entonces $\hat{C} = \{000, 011, 101, 110\}$. Nótese que C posee distancia mínima 1 y $\hat{C}$ distancia mínima 2.*

2.4.2. Por punción

Es el proceso contrario al de extensión. En este caso, una o más posiciones son borradas en todas las palabras del código.

Teorema 2.4.2. *Si C es un (n, M, d)-código, entonces C^* es un $(n - 1, M, d$ o $d - 1)$*

Teorema 2.4.3. *Un código $(n, M, 2t+1)$-código binario existe si y sólo si existe un $(n+1, M, 2t+2)$-código binario.*

2.4.3. Por borrado

Consiste en borrar algunas de las palabras del código.

2.4.4. Por aumentación

Es el proceso inverso al anterior. Una forma es añadir las palabras complementarias del código para el caso binario. Por ejemplo, si $c = 010111$, su complementario $c^c = 101000$.

Lema 2.4.1. *Si x, y pertenecen a $V(n, 2)$ (el espacio vectorial formado por todas las palabras binarias de longitud n), entonces $d(x, y^c) = n - d(x, y)$*

Teorema 2.4.4. *Si C es un (n, M, d)-código binario, entonces*

$$d(C \cup C^c) = min\{d, n - d_{max}\}$$

donde d_{max} es la distancia máxima de las palabras de C.

2.4.5. Por recorte

Consiste en mantener sólo las palabras que poseen un determinado símbolo en una determinada posición. Al símbolo y la posición en cuestión se le llama sección de corte ($x_i = s$).

Teorema 2.4.5. *Si C es un (n, M, d)-código lineal binario, el código recortado con sección de corte $x_1 = 0$, es un $(n - 1, 1/2M, d)$-código.*

2.4.6. Por suma directa

Si C_1 es un (n_1, M_1, d_1)-código y C_2 un (n_2, M_2, d_2)-código, entonces la suma directa de C_1 y C_2 viene dado por

$$C_3 = \{cd | c \in C_1, d \in C_2\}$$

donde cd es la concatenación de esas dos palabras. Y C_3 será un $(n_1 + n_2, M_1 M_2, min\{d_1, d_2\})$-código.

2.4.7. La $(u, u + v)$ construcción

Es una variación de la anterior. Si C_1 es un (n, M_1, d_1)-código y C_2 un (n, M_2, d_2)-código, ambos con la misma longitud, entonces

$$C_1 \oplus C_2 = \{c(c + d) | c \in C_1, d \in C_2\}$$

con parámetros $(2n, M_1 M_2, min\{2d_1, d_2\})$.

2.5. Ejercicios

1. Calcular las distancias de hamming de los siguientes ejemplos:
   a) d(10111,11001)
   b) d(abccd,acbcd)
   c) d(12345,54321)
2. Calcular la probabilidad de decodificicación erronea para un código binario de repetición de longitud 4. ¿Cuantos errores puede detectar y corregir ?. ¿Cuantos errores puede detectar si solo se utiliza para esto ?. Razonar cada una de las respuestas.
3. Repetir el ejercicio número 2 para longitud 5.
4. Sea $C = \{11100, 01001, 10010, 00111\}$.
   a) ¿Es un código?.
   b) Calcular la mínima distancia de C.
   c) Decodificar las siguientes palabras $\{10000, 01100, 00100\}$.
   d) Calcular la razón del código.
5. Construir, si es posible, un código binario con los parámetros $(8, 4, 5)$.
6. ¿Existe un $(7, 3, 5)$-código?.
7. Construir, si es posible, códigos binarios con los siguientes parámetros (n, M, d):
   a) (6,2,6)
   b) (3,8,1)
   c) (4,8,2)
   d) (5,3,4)
   e) (8,30,3)
8. Mostrar que si existe un código binario con parámetros (n, M, d), entonces existe un $(n - 1, M', d)$ con $M' \geq M/2$.
9. Mostrar que si existe un (n, M, d)-código con d par , entonces existe otro código con los mismos parámetros y todas las palabras de peso par.
10. Suponemos una serie de códigos con longitud de palabra $n = 12$, dar ejemplos de errores E_1, E_2, E_3 y E_4 tal que:
    a) $w(E_1) = 3$ y E_1 un *burst error* de longitud 5.
    b) $w(E_2) = 2$ y E_2 un *burst error* de longitud 6.
    c) E_3, E_4 son *single bytes errors* de longitud 4.
11. Mostrar que la distancia de hamming es una métrica.

12. Si la distancia mínima es 7, dar todas las opciones de decodificación posible si
    a) Si el modelo asumido para los errores es del tipo BSC.
    b) Si el modelo es del tipo BEC.
13. Sea $C = \{00000, 01110, 10101, 11011\}$, y $p = 10^{-3}$. Calcular la probabilidad de error
    a) Si se usa para detectar.
    b) Para corregir errores simples.
    c) Para corregir todos los posibles.