Teoría de la Información y Teoría de Códigos

José Angel de Bustos Pérez
Curso $1.999 - 2.000$

Clases de D. José María Muñoz Porras

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

El tratamiento de la información supone la codificación de un mensaje original y su transmisión, por un transmisor, a traves de un medio hasta un receptor. El transmisor codifica el mensaje y lo transmite hasta el receptor, una vez recibido el mensaje el receptor ha de decodificar el mensaje para así obtener el mensaje original.

Pero durante la transmisión, a traves del medio o canal , se pueden producir errores, ocasionando estos una alteración en el mensaje codificado. Debido a esto el mensaje recibido no será el mensaje original.

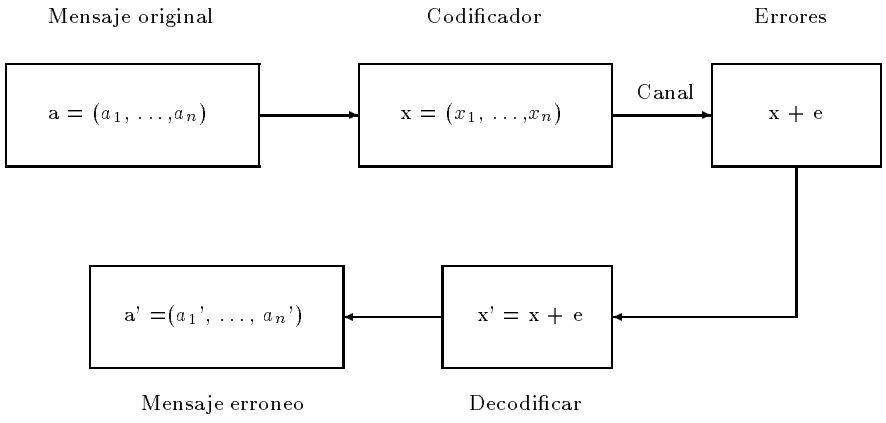

Figura 1.1: Transmisión de la información.

Una vez decodificado el mensaje erroneo a' se tratará de averiguar cual es, con mayor probabilidad, el mensaje original a.

1.1 Lenguajes y simbolos

Para emitir un mensaje hay que hacerlo utilizando un determinado lenguaje, común al transmisor y al receptor. Todo lenguaje estará formado por un conjunto de simbolos, los cuales formarán las palabras del lenguaje.

Los simbolos también recibiran el nombre de "bits".

Ejemplo 1.1

Cuando queremos comunicarnos con una persona, transmitir un mensaje, hablamos con esa persona en un idioma conocido por ambas partes, lenguaje, y los simbolos que utilizamos son las letras del abecedario, números, ... y el medio o canal utilizado puede ser el habla o la escritura, por ejemplo.

Ejemplo 1.2 (I.S.B.N.)

Sistema internacional de libros.

Este sistema se utiliza para catalogar y numerar los libros. Está formado por palabras de 10 *cifras y los simbolos que utiliza este lenguaje son:*

$$Smb_{I.S.B.N.} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Una palabra de este lenguaje será:

$$a = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}) \quad a_i \in Smb_{I.S.B.N.} \quad \forall \quad i = 1, \ldots, 10$$

1.2 Objetivos de la teoría de códigos

Los objetivos de la teoría de códigos son los siguientes:

- Construir códigos para la transmisión de información.
- Dichos códigos han de detectar cuando ha ocurrido un error en la transmisión de la información.
- Dichos códigos deben corregir el mayor número posible de errores.

Con el fin de detectar y corregir los posibles errores ocurridos durante la transmisión del mensaje original introduciremos información redundante acerca del mensaje original, con el fin de cotejar esta información con el mensaje original y poder comprobar de esta forma si se produjo algún error en la transmisión del mensaje.

No podemos abusar de la información redundante que introducimos. Al introducir información redundante en el mensaje a transmitir obtenemos un mensaje más largo de transmitir, por lo tanto será más costoso y lento el poder transmitirlo. Por lo tanto si introducimos demasiada información redundante tenemos la ventaja de poder detectar y corregir bastantes errores, pero también tenemos el inconveniente de un mayor costo al transmitir la información.

Otro objetivo que buscaremos será que cuando se origine un error en la transmisión de una palabra se produzca una palabra NO perteneciente al código. Por ejemplo si transmitimos una palabra "A", ocurre un error y, fruto de ese error la palabra que llega al receptor es "B", la situación ideal sería que "B" NO perteneciera a nuestro código. La explicación de esto es que si nos llega una palabra que NO es del código que estamos utilizando entonces se ha producido un error, mientras que sí la palabra es del código la daremos como buena y no nos percataremos de que se ha cometido un error en la transmisión.

1.2.1 Ejemplos de la utilización de códigos

Los códigos se usan de una manera continua en muchos campos, como por ejemplo:

- En las comunicaciones:
  - Radio.
  - Televisión.
  - Satélites.
  - Ordenadores.
- Sistemas de grabación de datos:
  - Voz.
  - Video.
  - CD-Rom.
- Comunicación escrita.
- Comunicación oral(sonora).

1.3 Notaciones

Trabajaremos siempre, salvo que no se diga lo contrario, sobre el conjunto $\mathbb{F}_q$. Este conjunto es un conjunto finito de q elementos y supondremos siempre que $q = p^n$, donde p es un número primo y $n \in \mathbb{N}$.

Nuestros códigos estarán formados por palabras de n "bits" o simbolos, donde cada palabra pertenece a $\mathbb{F}_q$.

El conjunto de todas las palabras de n simbolos donde cada simbolo es un elemento de $\mathbb{F}_q$ lo denotaremos como $\mathbb{F}_q^n$.

Luego tendremos que:

$$\mathbb{F}_q^n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in \mathbb{F}_q \quad \forall\ i = 1, \ldots, n\}$$

Como $\mathbb{F}_q$ tiene un número finito de elementos, q, dado un número entero $n < \infty$ podemos calcular la cantidad de elementos que tiene $\mathbb{F}_q^n$ al cual denotaremos como $|\mathbb{F}_q^n|$.

El número de elementos de $\mathbb{F}_q^n$, al que llamaremos orden de $\mathbb{F}_q^n$, será el número de combinaciones, distintas, que podemos hacer de n elementos de $\mathbb{F}_q^n$, es decir:

$$|\mathbb{F}_q^n| = q^n \ donde \ n \in \mathbb{N}$$

Los elementos de $\mathbb{F}_q^n$ los denotaremos de dos formas:

- $(a_1, \ldots, a_n)$ fundamentalmente cuando consideremos dicho elemento como elemento de una estructura algebraica.
- $a_1 \ldots a_n$ fundamentalmente cuando consideremos dicho elemento como elemento de un código.

pero ambas notaciones sirven para hacer referencia al mismo elemento.

1.4 Resumen

Para transmitir información lo que haremos será:

- Dividir la información en bloques o palabras.
- Codificar cada palabra del mensaje utilizando un código. Este paso supone añadir a cada palabra información redundante para poder controlar cuando se ha producido un error en la transmisión y, en algunos casos, corregir dicho error.
- Enviar el mensaje.

Una vez recibido el mensaje para poder conocer el mensaje original:

- Comprobar cada palabra del mensaje recibido, utilizando la información redundante[1], para comprobar que no hubo ningún error en la transmisión del mensaje. En caso de haberlo se tienen dos opciones:
  - Intentar corregir el error, en el caso que el código lo permita. Corregir el error debe enterderse como calcular la palabra del código que, con mayor probabilidad, fue transmitida originalmente.
  - Solicitar, de nuevo, la información erronea.
- Eliminar la información redundante introducida[2].

[1] "Bits" de control.

[2] Decodificar el mensaje.

1.4.1 Alfabeto

Definición 1.1 (Alfabeto)
*Un **"alfabeto"** será el conjunto de siímbolos utilizaremos para codificar el mensaje.*

En todo alfabeto existe un símbolo distinguido, que es el "*espacio en blanco*". Este símbolo nos permite distiguir cuando termina una palabra y comienza la siguiente.

Por ejemplo si utilizamos como alfabeto $\mathbb{F}_2$ los símbolos serian $\{0, 1\}$.

1.4.2 Palabras

Definición 1.2 (Palabra)
*Entenderemos por **"palabra"** a una sucesión ordenada de simbolos de un alfabeto.*

En los casos que vamos a considerar las palabras serán elementos de $\mathbb{F}_q^n$, puesto que vamos a utilizar palabras de longitud fija.

El número de elementos que posee $\mathbb{F}_q^n$ es $|\mathbb{F}_q|^n$.

1.4.3 Códigos

Definición 1.3 (Códigos)
*Llamaremos **"código"** al conjunto de palabras que utilizaremos para codificar información.*

En los casos que vamos a considerar los códigos serán subconjuntos de $\mathbb{F}_q^n$, es decir $\mathcal{C} \subset \mathbb{F}_q^n$.

Capítulo 2

Códigos de bloques

Hemos visto en el capítulo anterior que para enviar enviar información lo haremos dividiendola en bloques o palabras y mandando los bloques por separado. Pero no hemos dicho nada sobre la longitud de las palabras que vamos a utilizar para mandar los mensajes.

Cuando utilizemos un código para mandar información lo haremos siempre, a no ser que se diga lo contrario, con una longitud fija de palabra.

2.1 Código de bloques (n, m)

Definición 2.1 (Código de bloques de tipo (n, m))
Diremos que un código de bloques es de ***"tipo*** (n, m)*"* *cuando la longitud de palabras del código sea de n bits y de esos n bits utilizemos m para transmitir información. Es decir tendremos $n - m$ bits para añadir información redundante. A un código de este tipo lo denotaremos como $\mathcal{C}[n, m]$.*

Obviamente se tiene que $n, m \in \mathbb{N}$.

2.1.1 Longitud de un código

Definición 2.2 (Longitud de un código)
Dado un código de tipo (n, m) llamaremos ***"longitud del código"*** *al número n, es decir, a la longitud de las palabras empleadas por el código.*

2.1.2 Rango de un código

Definición 2.3 (Rango de un código)
Dado un código de tipo (n, m) llamaremos ***"rango del código"*** *al número m, es decir, al número de bits utilizados para transmitir información.*

2.1.3 Razón de un código

Definición 2.4 (Razón de un código)
Dado un código de tipo (n, m) llamaremos ***"razón del código"*** *al número $\frac{m}{n} \leq 1$.*

Nos interesan códigos cuya razón este cerca de 1. En estos códigos introduciremos pocos bits de control, por lo tanto son códigos en los que se emplea poco tiempo en su transmisión[1].

2.1.4 Tasa de información de un código

Definición 2.5
Dado un código de tipo (n, m) llamaremos ***"tasa de información del código"*** *$\mathcal{C}$ a:*

$$R = n^{-1} \cdot \log_q |\mathcal{C}|$$

el código está definido sobre un cuerpo $\mathbb{F}_q$.

Sí $\mathcal{C}$ es un código lineal del tipo (n, m) entonces:

$$R = n^{-1} \cdot \log_q |\mathcal{C}| = \frac{m}{n}$$

2.1.5 Radio de recubrimiento de un código

Definición 2.6 (Radio de recubrimiento)
Dado un código de tipo (n, m) llamaremos ***"radio de recubrimiento del código"*** *a:*

$$\rho(\mathcal{C}) = \max_{x \in \mathbb{F}_q^n} \{ \min_{c \in \mathcal{C}} \{ d(x, c) \} \}$$

La interpretación de este número es la siguiente:

> $\rho(\mathcal{C})$ es el menor número positivo, ρ, tal que las bolas $B_\rho(c)$ de radio ρ y con centro en los puntos $c \in \mathcal{C}$ recubren $\mathbb{F}_q^n$.

[1] En comparación con el mismo código pero con más bits de control.

2.2 Codificadores

Matemáticamente hablando un "*codificador*" es una aplicación que traduce una palabra a su equivalente respecto de un código.

Definición 2.7 (Codificador)
Sea $\mathcal{C}$ un código del tipo (n, m). Llamaremos ***"codificador"*** *a una aplicación biyectiva C, tal que:*

$$\begin{array}{rcl} C : \mathbb{F}_2^m & \xrightarrow{\sim} & \mathcal{C} \subset \mathbb{F}_q^n \\ x & \longrightarrow & u \end{array}$$

La labor del "*codificador*" consiste en asignar una, y solo una, palabra del código a cada palabra del lenguaje. Es decir, se limita a añadir los bits de control a cada palabra del lenguaje.

Supondremos que los "*codificadores*" son **estándar**, es decir:

$$C(x) = u = (x_1, \cdots, x_m, c_1, \cdots, c_{n-m}) \quad \textit{donde } x_i, c_j \in \mathbb{F}_q$$

donde los c_j para $j = 1, \cdots, n-m$ son los bits de control.

2.3 Decodificadores

Definición 2.8 (Decodificador)
Sea $\mathcal{C}$ un código del tipo (n, m), Llamaremos ***"decodificador"*** *a una aplicación biyectiva D, tal que:*

$$\begin{array}{rcl} D : \mathbb{F}_2^n \supset \mathcal{C} & \xrightarrow{\sim} & \mathbb{F}_2^m \\ u & \longrightarrow & x \end{array}$$

La labor del "*decodificador*" consiste en asignar una, y solo una, palabra del lenguaje a cada palabra del código. Es decir, se limita a quitar los bits de control a cada palabra del código.

2.4 Procesadores de error

Cuando hemos recibido una transmisión, ¿Como comprobar si ha habido algún error en la transmisión?. Esta labor la realizan los "*procesadores de error*".

Definición 2.9 (Procesadores de error)
Un ***"procesador de error"****, desde el punto de vista matemático, para un código de tipo* (n, m) *es una aplicación:*

$$\begin{array}{rccc} P: & \mathbb{F}_2^m & \longrightarrow & \mathbb{F}_2 \times \mathcal{C}[n,m] \\ & v & \longrightarrow & (0/1, u') \end{array}$$

tal que:

- *Si* $v \in \mathcal{C}[n,m]$ *entonces* $P(v) = (0, v)$.
- *Si* $v \notin \mathcal{C}[n,m]$ *entoces* $P(v) = (1, u')$. *Donde* $u' \in \mathcal{C}[n,m]$ *es la palabra que, con más probabilidad, se transmitió originalmente. Es decir* u' *será la palabra del código "más parecida" a* v.

Cuando se recibe una palabra y esta pertenece al código que se esta utilizando, no quiere decir que no haya habido algún error en la transmisión. Puede haber ocurrido algún error y la palabra resultante del error ser una palabra del código. Para evitar que ocurra esto se puede intentar "*separar*" las palabras del código lo más posible, pero para ello es necesario saber "*como medir distancias*" dentro del código.

Definición 2.10 (Transmisión correcta)
Entenderemos por ***"transmisión correcta"*** *aquella que verifique que* $P(v) = (0/1, u)$, *donde* P *es el procesador de error,* v *es la palabra recibida y* u *es la palabra transmitida originalmente.*

Intentaremos utilizar códigos en los que las palabras esten lo más "*separadas*" posible, ya que cuanto más "*separadas*" esten entre sí, más dificil será que un error en una palabra del código nos de otra palabra del código. Con lo cual el error sería indetectable[2].

Una vez detectado un error en la transmisión supondremos que la palabra que, con más probabildad, fue transmitida originalmente será aquella palabra del código que esté más "*cercana*" a la palabra recibida.

[2] A menos que se conozca, a priori, la palabra transmitida.

2.5 Noción de distancia

Dar una distancia en un código equivale a decir cuando dos palabras en el código son diferentes y en "*cuanto*" difieren.

Dadas dos palabras $a = (a_1, \cdots, a_n)$ y $b = (b_1, \cdots, b_n)$, donde $a, b \in \mathcal{C}[n, m] \subset \mathbb{F}_q^n$, se tienen las siguientes definiciones:

Definición 2.11 (Igualdad de palabras)
Se dice que dos palabras a y b, del mismo código, son ***"iguales"*** *cuando se verifica que* $\forall\ i$ *se tiene que* $a_i = b_i$*. En caso contrario ambas palabras son* ***"distintas"****.*

Definición 2.12 (Error en el lugar i-ésimo)
Dadas dos palabras a y b, del mismo código, diremos que hay un ***"error en el lugar*** i***-ésimo"*** *cuando* $a_i \neq b_i$*.*

Medir la diferencia de dos palabras dadas, de un mismo código, equivale a dar una noción de distancia, o lo que es lo mismo una "*métrica*", en el código. Esta diferencia la mediremos utilizando la "**distancia de Hamming**".

Definición 2.13 (Distancia de Hamming)
Dadas dos palabras a y b, del mismo código, definiremos su distancia como el numero de i tales que $a_i \neq b_i$*, o lo que es lo mismo, como el número de i tales que* $a_i - b_i \neq 0$*. La distancia entre dos palabras la denotaremos como* $d(a, b)$*.*

Matemáticamente hablando tendremos, que la "distancia de Hamming" vendrá dada por una aplicación:

$$\begin{array}{rccc} d: & \mathbb{F}_q^n \times \mathbb{F}_q^n & \longrightarrow & \mathbb{Z}^+ \\ & (a, b) & \longrightarrow & d(a, b) \end{array}$$

Como observación diremos que esta "*distancia*" también se verifica para $\mathbb{F}_q^n$, con lo cual la distancia no sólo está definida en el código, sino que también está definida en el total.

Definición 2.14 (Error de peso k)
Dadas dos palabras a y b, de un mismo código, diremos que existe un ***"error de peso*** k***"*** *si* $d(a, b) = k$*.*

Definición 2.15 (Peso de una palabra)
Llamaremos ***"peso de una palabra"*** *a su distancia con el* 0*. El peso de una palabra de n bits será:*

$$d((a_1, \ldots, a_n), (0, \overset{n)}{\ldots}, 0))$$

y lo denotaremos como $w(u)$*, con* $u = (a_1, \ldots, a_n)$*.*

2.5.1 Algunos ejemplos

Consideremos un código $\mathcal{C}[4,1] \subset \mathbb{F}_2^4$. Cada palabra de $\mathbb{F}_2^4$ será de la forma $a = (a_1, a_2, a_3, a_4)$.

- $a = (0,0,1,0)$ y $b = (1,1,1,1)$ entonces $d(a,b) = 3$ ya que $a_1 \neq b_1$, $a_2 \neq b_2$ y $a_4 \neq b_4$. Es decir hay 3 bits que no coinciden. Tenemos un error de peso 3.
- $a = (0,1,0,1)$ y $b = (0,1,0,1)$ entonces $d(a,b) = 0$ ya que $a_i = b_i \ \forall \ i$. Tenemos un error de peso 0, ambas palabras son iguales.
- $a = (1,0,0,1)$ y $b = (0,1,1,0)$ entonces $d(a,b) = 4$ ya que $a_i \neq b_i \ \forall \ i$. Es decir ambas palabras no coinciden en ningún bit. Tenemos un error de peso 4.

2.5.2 Propiedades de las distancias

Hemos definido antes una aplicación a la que hemos llamado "*distancia de Hamming*". Una "*distancia*" es una aplicación que cumple una serie de condiciones.

Dado un conjunto cualquiera A, diremos que una aplicación, d, es una distancia si es de la forma:

$$\begin{aligned} d : A \times A &\longrightarrow \mathbb{R}^+ \\ (a,b) &\longrightarrow d(a,b) \end{aligned}$$

y verifica las siguientes condiciones:

- $d(a,b) \geq 0 \ \forall \ a,b \in A$ y $d(a,b) = 0 \iff a = b$.
- $d(a,b) = d(b,a) \ \forall \ a,b \in A$.
- "*Desigualdad triangular*"

$$d(a,c) \leq d(a,b) + d(b,c) \quad \forall \ a,b,c \in A$$

En nuestros casos $A = \mathbb{F}_q^n$ y en lugar de considerar $\mathbb{R}^+$ consideraremos $\mathbb{Z}^+$.

2.5.3 Detección de errores mediante las distancias

Para detectar errores en una transmisión utilizaremos una propiedad de las distancias:

> La distancia entre dos palabras es nula $\Longleftrightarrow$ sí ambas palabras son la misma. $d(a,b) = 0 \Longleftrightarrow a = b$.

Cuando hayamos recibido una palabra en una transmisión lo que haremos para detectar si ocurrio algún error en la transmisión será:

- Calcular la distancia de la palabra recibida, u, a todas las del código.
- Si alguna de las distancias es nula entonces se tiene que la palabra recibida u pertenece al código. En caso contrario la palabra recibida u no pertenece al código, con lo cual habrá ocurrido algún error en la transmisión.

2.6 Elección de buenos códigos

Ya hemos comentado anteriormente que al transmitir una palabra se produzca un error, y este error genere otra palabra del código, con lo cual el error cometido sería indetectable. Por ejemplo en el "*código de triple repetición*", el cual está formado unicamente por dos palabras $\mathcal{C}[3,1] = \{000, 111\}$ supongamos que transmitimos la palabra 000 y se cometen errores:

- **de peso** 1, eso significaría que la palabra recibida sería una de las siguientes:
  - 100.
  - 010.
  - 001.

  donde el error es detectable, ya que ninguna de ellas pertenece al código.

- **de peso** 2, eso significaría que la palabra recibida sería una de las siguientes:
  - 110.
  - 101.
  - 011.

  donde el error sigue siendo detectable, ya que ninguna de ellas pertenece al código.

- **de peso** 3, eso significaría que la palabra recibida será la siguiente:
  - 111.

  donde el error es indetectable, ya que la palabra recibida pertenece al código.

Luego con este código podemos detectar errores de peso 2 ya que sus palabras tienen una distancia entre sí de 3.

De esto se deduce que para poder detectar errores de peso $s+1$ hemos de utilizar códigos cuyas palabras disten entre sí s.

2.6.1 Distancia mínima

Supongamos que tenemos un código $\mathcal{C}[n,m] \subset \mathbb{F}_q^n$, la distancia mínima de dicho código es un dato importante. Esto es debido a que nos indica que tipos de errores se detectan en el código.

Definición 2.16 (Distancia Mínima)
Llamaremos ***"distancia mínima"*** *de un código a la mínima distancia entre todos los posibles pares de palabras que podamos formar con las palabras del código. Es decir:*

$$d_{min}(\mathcal{C}[n,m]) = \min_{\substack{a,b \in \mathcal{C}[n,m] \\ a \neq b}} \{\ d(a,b)\ \}$$

Observar que todos los códigos que utilizaremos van a ser subconjuntos de cuerpos finitos y entonces tiene sentido hablar de "*distancia mínima*".

A esta distancia para abreviar nos referiremos como d_{min}.

Muchas veces un código $\mathcal{C}[n, m]$ se suele denotar como $\mathcal{C}[n, m, d_{min}]$.

Dado un código $\mathcal{C}[n, m, d_{min}]$ tendremos que la distancia mínima, entre dos palabras diferentes, de dicho código es de d_{min}. Luego difieren en, al menos, d_{min} bits. En un código de este tipo son detectables errores de peso menor, estrictamente, que d_{min}, mientras que los errores de peso mayor o igual que d_{min} son indetectables[3].

Teorema 2.1 (Teorema de detección de errores)
Un código detecta errores de peso menor o igual que $s \iff d_{min}$ es mayor, estrictamente, que s.

Demostración:

$\Rightarrow$ | Supongamos que tenemos un código $\mathcal{C}$ que detecta errores de peso menor o igual que s.

Según lo visto en el apartado (2.5.3), en la página 18, detectar un error de peso menor o igual que s significa que, si dada una palabra del código cualquiera alteramos, a lo sumo, s de sus bits, entonces la palabra resultante NO pertenece al código. Mientras que si alteramos más de s de sus bits la palabra resultante podría pertenecer al código. Esto nos dice que dos palabras del código difieren en $s + 1$ bits, por lo menos. O lo que es lo mismo que la distancia mínima, d_{min}, del código es mayor, estrictamente, que s.

$\Leftarrow$ | Supongamos que tenemos un código $\mathcal{C}$ con distancia mínima d_{min} verificando que $d_{min} > s$.

Entonces si, dada una palabra cualquiera del código alteramos, a lo sumo, s de sus bits entonces la palabra obtenida no pertenecerá al código. Pero según lo visto en el apartado (2.5.3), en la página 18, esto es equivalente a detectar errores de, a lo sumo, peso k.

■

De todo esto se deduce que, cuanto mayor sea la distancia mínima mayor número de errores detectará el código. Los códigos cuya distancia mínima sea "*grande*" serán buenos códigos, ya que podremos detectar un mayor número de errores. Así mismo cuanto mayor distancia mínima tenga un código más díficil será que un error en una palabra transmitida de como resultado otra palabra del código, con lo cual el error sería indetectable.

[3]Generalmente, ya que puede darse el caso en el que un error de peso mayor que la distancia mínima sea detectable.

2.6.2 Calculo de todas las distancias mínimas

Ya hemos visto que para calcular la distancia mímina de un código tenemos que calcular la distancia de cada elemento a todos y cada uno de los otros.

Como los códigos son finitos, tienen un número finito de elementos, podemos calcular el número de distancias que tenemos que calcular.

Dado un código $\mathcal{C}$ el cual tiene $|\mathcal{C}|$ palabras. Para calcular la distancia mínima del código tendremos:

- Dada una palabra del código tendremos que calcular la distancia de dicha palabra con el resto de palabras del código. Como el código tiene $|\mathcal{C}|$ palabras entonces tendremos que calcular $|\mathcal{C}| - 1$ distancias.
- Como tenemos que repetir esta operación con todas las palabras del código entonces tendremos que calcular $|\mathcal{C}| \cdot (|\mathcal{C}| - 1)$ distancias.

Luego el número total de distancias a calcular es de $|\mathcal{C}| \cdot (|\mathcal{C}| - 1)$ distancias.

2.7 Corrección de errores

Ya hemos visto como detectar cuando ha ocurrido un error en una transmisión, pero eso no es suficiente. También hay que saber cuando y como se puede corregir un error. Los procesadores de error[4] serán los encargados de detectar y corregir los errores.

Hemos visto que calculando la distancia mínima de un código podemos saber hasta que peso se pueden detectar errores.

Pero una vez detectado que se ha cometido un error, ¿como elegir la palabra adecuada que se transmitió originalmente?. Esto también lo haremos mediante la distancia mínima.

[4] Apartado (2.4) en la página 15.

Supongamos que recibimos una palabra w y deseamos ser capaces de corregir todos los errores de peso 1. Una vez recibida la palabra w y sabiendo que se ha cometido un error de peso 1 debería haber una única palabra del código u tal que $d(u,w) = 1$. Pero se puede dar el caso en el que existan $u, v \in \mathcal{C}$ tales que $d(u,w) = d(v,w) = 1$, entoces en esta situación, ¿que palabra elegiriamos como correcta?.

El mismo razonamiento se puede seguir para errores de peso k.

Definición 2.17 (Corrección de errores de peso k)
Dada una palabra w en la que sabemos que existe un error de peso k, a lo sumo, diremos que un código $\mathcal{C}$ corrige errores de peso menor o igual que k $\iff$ existe una y sólo una palabra $u \in \mathcal{C}$ tal que $d(u,w) \leq k$ para cualquier w en el que se cometa un error de peso k, a lo sumo.

Para solucionar este problema elegimos códigos en los que las palabras esten bastante separadas.

Teorema 2.2 (Teorema de corrección de errores)
Dado un código $\mathcal{C}$ se pueden corregir, sin problemas, todos los errores de peso menor o igual que t $\iff$ su distancia mínima, d_{min}, es mayor o igual que $2 \cdot t + 1$.

Demostración:
$\Rightarrow$ | Supongamos que tenemos un código que es capaz de corregir errores de peso menor o igual que t. Entonces si recibimos una palabra w en la que sabemos que existe un error de peso t, a lo sumo, entonces existirá una y sólo una palabra $u \in \mathcal{C}$ tal que $d(w,u) \leq t$.

Lo demostraremos por reducción al absurdo. Supondremos que existen dos palabras en el código, $u_1, u_2 \in \mathcal{C}$, tales que $d(u_1,u_2) \leq 2 \cdot t$.

Sea w una palabra tal que:

- Coincide con u_1 en los bits en los que coinciden u_1 y u_2.
- Coincide con u_1 en los t primeros bits en los que u_1 y u_2 no coinciden.
- Coincide con u_2 en los bits siguientes, a los t primeros, en los que u_1 y u_2 no coinciden.

De esta construcción es claro que $d(w,u_2) = t$.

Supongamos que u_1 y u_2 son palabras de n bits que coinciden en m bits. Entonces tendremos que $d(w, u_1) = n - m - ([n - m] - t) = t$.

Suponemos que recibimos la palabra w y sabemos que se cometió un error de peso t, a lo sumo, en la transmisión. Entonces la palabra que se transmitió originalmente es la única palabra del código, u, tal que $d(w, u) \leq t$ ya que por hipótesis el código corrige errores de peso menor o igual que t.

Ahora bien, tenemos en el código dos palabras, u_1, u_2, tales que verifican que $d(w, u_i) \leq t$ para $i = 1, 2$. Pero esto no puede ocurrir por hipótesis ya que unicamente puede haber una palabra en el código que cumpla la condición. Hemos llegado a una contradicción al suponer que $d(u_1, u_2) \leq 2 \cdot t$ para $u_1, u_2 \in \mathcal{C}$. Entonces se tendrá que $d(u_1, u_2) > 2 \cdot t$ para $u_1, u_2 \in \mathcal{C}$, es decir, $d(u_1, u_2) \geq 2 \cdot t + 1$ para $u_1, u_2 \in \mathcal{C} \Longrightarrow d_{min} \geq 2 \cdot t + 1$.

$\Leftarrow$ | Supongamos que tenemos un código cuya distancia mínima, d_{min}, es mayor o igual que $2 \cdot t + 1$.

Supongamos que se ha realizado una transmisión y se ha recibido la palabra w, conociendo que se ha cometido un error de, a lo sumo, peso t. Obviamente $w \notin \mathcal{C}$, ya que en caso contrario el error no sería detectable.

Supongamos que existen dos palabras en el código $u, v \in \mathcal{C}$ tal que $d(u, w) \leq t$ y $d(v, w) \leq t$. Entonces utilizando la desigualdad triangular se tiene que:

$$d(u, v) \leq d(u, w) + d(w, v) = d(u, w) + d(v, w) = t + t = 2 \cdot t$$

Es decir $d(u, v) \leq 2 \cdot t$, lo cual es falso ya que como $u, v \in \mathcal{C}$ se tiene, por hipótesis, que $d(u, v) \geq 2 \cdot t + 1$.

De todo esto se deduce que sólo hay una y sóla una palabra en el código, u, tal que $d(u, w) \leq t$, con lo cual u es la palabra que se transmitió originalmente. Es decir podemos corregir errores de peso menor o igual que t.

En todo esto hemos supuesto que siempre existe una palabra en el código, u, tal que $d(u, w) \leq t$. Si esta palabra no existiera no tendría sentido este teorema, ya que es en dicha palabra en la que se produce el error, luego siempre existe una palabra, por lo menos, en el código verificando la condición $d(u, w) \leq t$.

■

Podemos interpretar el resultado de este teorema como:

> Corregir adecuadamente cuesta el doble que detectar.

Definición 2.18 (Código t-detector de errores)
Diremos que un código es un ***"código t-detector de errores"*** *si verifica que* $d_{min} = 2 \cdot t + 1$.

Podemos mezclar los teoremas 2.1 y 2.2 en un sólo teorema:

Teorema 2.3 (Teorema de detección y corrección de errores)
Un código corrige errores de peso menor o igual que t *y detecta errores de peso menor o igual que* $s' = s + t \iff d_{min} \geq 2 \cdot t + s + 1 = t + s' + 1$.

Capítulo 3

Códigos lineales

Hemos visto que los códigos de bloques, con sus codificadores, decodificadores, procesadores de error se definen sobre un alfabeto $\mathbb{F}_q$, el cual es un conjunto cualquiera. Pero si sobre este conjunto podemos establecer alguna estructura algebraica, la cual nos permita realizar operaciones sobre los elementos del conjunto, podremos entonces simplificar el código. Esta simplificación nos permitirá trabajar de una forma más comoda.

3.1 Requisitos para códigos lineales

Cuando trabajemos con códigos lineales tendremos que $(\mathbb{F}_q, +, \cdot)$ es cuerpo, por ejemplo $\mathbb{F}_2 = \mathbb{Z}/2 = \mathbb{Z}_2 = \{0, 1\}$.

Dado un cuerpo cualquiera A se tiene que $A^n = A\times \overset{n)}{\cdots} \times A$ es un A-espacio vectorial con las siguientes operaciones:

- $(x_1, \cdots, x_n) + (x'_1, \cdots, x'_n) = (x_1 + x'_1, \cdots, x_n + x'_n)$, donde tenemos que $(x_1, \cdots, x_n)$ y $(x'_1, \cdots, x'_n)$ son dos palabras cualesquiera de A^n.
- $x \odot (x_1, \cdots, x_n) = (x \cdot x_1, \cdots, x \cdot x_n)$, donde tenemos que $x \in A$, $(x_1, \cdots, x_n) \in A^n$ y $\odot$ es el producto por escalares definido en A^n.

Nosotros trabajaremos con cuerpos de la forma $\mathbb{F}_q$ y con $\mathbb{F}_q^n$ que, por lo visto antes, serán $\mathbb{F}_q$-espacios vectoriales.

3.2 ¿Cuando un código es lineal?

Podemos distinguir unos códigos de bloques especiales que son los códigos lineales.

Definición 3.1 (Códigos Lineales)
Dado un cuerpo cualquiera, $\mathbb{K}$, y un $\mathbb{K}$-espacio vectorial de la forma $\mathbb{K}^n$ diremos que un subconjunto $\mathcal{C} \subset \mathbb{K}^n$ es un ***"código lineal"*** *si $\mathcal{C}$ es un subespacio vectorial de $\mathbb{K}^n$.*

Nosotros consideraremos $\mathbb{K} = \mathbb{F}_q$ y $\mathbb{K}^n = \mathbb{F}_q^n$.

3.2.1 Propiedades de los códigos lineales

Dado que los códigos lineales son subespacios vectoriales poseen unas propiedades especiales.

Sea $\mathcal{C} \subset \mathbb{F}_2^n$ un código lineal, entonces se verifican las siguientes propiedades:

- La suma de dos palabras del código es otra palabra del código.

$$(x_1, \cdots, x_n) + (x'_1, \cdots, x'_n) = (x_1 + x'_1, \cdots, x_n + x'_n) \in \mathcal{C}$$

- Multiplicar una palabra del código por un elemento del cuerpo es otra palabra del código.

$$x \odot (x_1, \cdots, x_n) = (x \cdot x_1, \cdots, x \cdot x_n) \in \mathcal{C}$$

- El 0 siempre es una palabra del código.

$$(0, \overset{n)}{\cdots}, 0) \in \mathcal{C}$$

3.3 Calculo de la distancia mínima

Debido a la estructura de espacios vectoriales que poseen estos códigos se puede simplificar el calculo de la distancia mínima, d_{min}. Según la definición de distancia mínima el número de distancias que tendremos que calcular para encontrala será de $\frac{|\mathcal{C}|^2}{2}$, es decir, tendríamos que calcular tantas distancias como parejas de dos elementos distintos del código podamos tener dividido por dos, ya que la distancia es simétrica.

Proposición 3.1 (Distancia mínima para códigos lineales)
Sea $\mathcal{C} \subset \mathbb{F}_q^n$ un código lineal, entonces:

$$d_{min} = \min_{\substack{u \in \mathcal{C} \\ u \neq 0}} \{ \ d(u,0) \ \}$$

Demostración:
Supongamos que la distancia mínima es:

$$d_{min} = d(u,v) \quad u, v \in \mathcal{C}$$

entonces por la definición de la distancia de Hamming se tiene que:

$$d(u,v) = d(u-v,0)$$

y como el código $\mathcal{C}$ es un código lineal se tiene entonces que $u - v \in \mathcal{C}$. Es decir:

$$d_{min} = \min_{\substack{u \in \mathcal{C} \\ u \neq 0}} \{ \ d(u,0) \ \}$$

■

Observacion: 3.3.1
De esta proposición se deduce que para calcular la distancia mínima de un código lineal basta con calcular la distancia de todas las palabras, salvo la palabra cero, al cero y tomar el mínimo de esas distancias. Con lo cual unicamente tendremos que calcular $|\mathcal{C}| - 1$ distancias en lugar de las $\frac{|\mathcal{C}|^2}{2}$ distancias que deberiamos calcular para un código no lineal.

3.4 Codificadores

Según el apartado 2.2, en la página 14, un codificador es una aplicación biyectiva.

Definición 3.2 (Codificador para códigos lineales)
*Sea $\mathcal{C} \subset \mathbb{F}_q^n$ un código lineal. Un **"codificador"** para un código lineal es un isomorfismo[1] del siguiente tipo:*

$$\begin{array}{rccc} C : \mathbb{F}_q^m & \xrightarrow{\sim} & \mathcal{C} \\ x & \longrightarrow & u \end{array}$$

tal que codifica palabras de longitud m con palabras de longitud n.

[1] Aplicación lineal biyectiva.

Observacion: 3.4.1

- *Al ser el codificador una aplicación biyectiva cada palabra tiene una, y sólo una, forma de codificarse.*
- *Todas las posibles palabras que puedan formar un mensaje se pueden codificar con una palabra del código. Esto es gracias a que el codificador es epiyectivo, y, por lo dicho en el punto anterior dicha palabra es única.*
- *Como el codificador es una aplicación lineal podemos representar el codificador por la matriz que representa a la aplicación lineal una vez fijadas las correspondientes bases. A está matriz se la conoce como "**matriz generadora**".*
- *Como la matriz depende de las bases escogidas, entonces la matriz del codificador no es única.*
- *La matriz de un codificador de un código lineal estará formada por tantas columnas como dimensión tenga el código. Y la columna i-ésima del codificador será el vector i-esimo de la base, del código, codificado según el código.*
- *Como el codificador es un isomorfismo, aplicación lineal biyectiva, la imagen de una base es otra base, entonces las columnas de la matriz generadora son base del código $\mathcal{C}$.*

3.4.1 Matriz generadora

Definición 3.3 (Matriz generadora)
*Dado un código lineal y su codificador llamaremos **"matriz generadora"** de código a la matriz de la aplicación lineal que representa al codificador respecto de unas bases fijadas.*

Para codificar una palabra basta con multiplicarla por la matriz generadora del código y obtendremos la palabra codificada. Por ejemplo, sea C la matriz generadora de un código y queremos codificar la palabra $(1, 1, 0)$:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Luego la palabra $(1,1,0)$ codificada es $(1,1,0,0,1,1)$.

La siguiente proposición nos dice cuando una matriz es una matriz generadora de un código, y en caso afirmativo, de que tipo de código se trata.

Proposición 3.2
Sea $\mathcal{C}[n,m]$ un código lineal y sea C una matriz de orden $n \times m$. C es una matriz generadora para el código $\mathcal{C}[n,m]$ sí y sólo sí tiene rango m y sus columnas son palabras del código.

Demostración:
$\Rightarrow$ | Sea C una matriz generadora para el código $\mathcal{C}[n,m]$.

Por definición de matriz generadora sus columnas serán palabras del código, y como el código lineal es de dimensión m, dimensión del subespacio imagen de la aplicación lineal que determina C, entonces el rango de C es m.

$\Leftarrow$ | Sean C_1 , ..., C_m las columnas de la matriz C, las cuales son, por hipótesis, palabras del código.

Sea $a = (a_1, \ldots, a_m)$ una palabra entonces $C \cdot a^t = a_1 \cdot C_1^t + \ldots + a_m \cdot C_m^t$ que es una combinación lineal de $\{C_i\}_{i=1}^m$ y como el código $\mathcal{C}[n,m]$ es lineal se tiene que las combinaciones lineales de palabras del código son palabras del código. Luego la matriz C transforma cualquier palabra, de longitud m, en una palabra del código.

La dimensión del subespacio imagen es el rango de la matriz C, luego la dimensión del subespacio imagen es m.

Como la matriz C codifica palabras de longitud m en palabras de longitud n en un subespacio de dimensión m entonces C es la matriz generadora de un código $\mathcal{C}[n,m]$.

■

3.4.2 Forma estándar de la matriz generadora

La expresión de la "**matriz generadora**" no es unica, sino que depende de las bases escogidas. Luego eligiremos un convenio para elegir las bases en las que expresaremos dicha matriz.

Definición 3.4 (Forma estándar de la matriz generadora)
Sea $\mathcal{C} \subset \mathbb{F}_q^n$ un código lineal que utilizaremos para codificar palabras de longitud m, $\mathbb{F}_q^m$. Sea C la matriz del codificador, una vez fijadas las bases correspondientes. Diremos que C está en ***"forma estándar"*** *cuando los m primeros elementos de la palabra codificada $C(x) \in \mathcal{C} \subset \mathbb{F}_q^n$ coincidan con $x \in \mathbb{F}_q^m$.*

Proposición 3.3
Sea $\mathcal{C}[n, m]$ un código lineal. La matriz generadora está en forma estándar sí y sólo sí en las m primeras columnas aparece la matriz identidad de orden m.

Demostración:
Tanto el directo como el recíproco son una mera comprobación.

■

3.5 Ecuaciones de los códigos lineales

Como hemos visto un código lineal $\mathcal{C}$ es un subespacio vectorial. Utilizando la teoría de espacios vectoriales podemos calcular las ecuaciones de un subespacio vectorial cualquiera. La utilidad de estas ecuaciones radica en que nos indican las condiciones que ha de cumplir un vector para pertenecer al código.

Sea $\mathcal{C} \subset \mathbb{F}_q^n$ un código lineal, donde $\mathbb{F}_q^n$ es un $\mathbb{F}_q$-espacio vectorial tal que $\dim_{\mathbb{F}_q} \mathbb{F}_q^n = n$. Supongamos que $\dim_{\mathbb{F}_q} \mathcal{C} = m$, con $m < n$ y que el espacio de palabras a codificar es $\mathbb{F}_q^m$.

3.5.1 Ecuaciones paramétricas

Una vez conocida la matriz generadora del código se pueden calcular las ecuaciones paramétricas del código. Dichas ecuaciones nos permitirán calcular todas las palabras del código.

Sea C la matriz generadora del código, para calcular las ecuaciones paramétricas del código bastará con aplicar la matriz generadora a un vector genérico del espacio de palabras a codificar.

$$C \cdot \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}$$

luego las ecuaciones paramétricas serán:

$$\begin{cases} x_1 & = & \theta_1 \\ x_2 & = & \theta_2 \\ \cdots & \cdots & \cdots \\ x_n & = & \theta_n \end{cases}$$

Tendremos que $\theta_i = f_i(\lambda_1, \ldots, \lambda_m)$ para $i = 1, \ldots, n$ y dando valores a $\{\lambda_j\}_{j=1}^{m}$ en el cuerpo $\mathbb{F}_q$ tendremos todos los elementos del código.

3.5.2 Ecuaciones implícitas

Para calcular las ecuaciones implícitas de $\mathcal{C}$ calcularemos el subespacio incidente a $\mathcal{C}$, $\mathcal{C}^\circ \subset (\mathbb{F}_q^n)^*$. El subespacio incidente está formado por las formas lineales que se anulan sobre $\mathcal{C}$.

El número de ecuaciones implícitas que tendrá nuestro código $\mathcal{C}$ será:

$$dim_{\mathbb{F}_q}\mathbb{F}_q^n - dim_{\mathbb{F}_q}\mathcal{C} = n - m$$

Como $\dim_{\mathbb{F}_q}\mathcal{C} = m$ tendremos que $\mathcal{C} =< c_1, \ldots, c_m >$. Para calcular las ecuaciones tendremos que resolver un sistema de ecuaciones que tendrá más de una solución.

Sean $\{x_i^j\}_{j=1}^m$ con $i = 1, \ldots, n$ la i-ésima coordenada del vector j-ésimo de la base de $\mathcal{C}$. El sistema que nos da las ecuaciones será:

$$\begin{pmatrix} x_1^1 & \ldots & \ldots & x_n^1 \\ x_1^2 & \ddots & & x_n^2 \\ \vdots & & \ddots & \vdots \\ x_1^m & \ldots & \ldots & x_n^m \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}$$

Cualquier $x = (x_1, \ldots, x_n)$ que verifique este sistema de ecuaciones será un elemento de $\mathcal{C}$, o lo que es lo mismo cualquier punto que no verifique el sistema de ecuaciones anterior NO pertenecerá a $\mathcal{C}$.

3.6 Detección de errores

Supondremos que tenemos un código lineal $\mathcal{C} \subset \mathbb{F}_q^n$ tal que $\dim_{\mathbb{F}_q} \mathcal{C} = m$, con $m < n$. De esto se deduce que utilizaremos el código para codificar palabras de longitud m luego es un código $\mathcal{C}[n, m]$.

Se detecta un error cuando una de las palabras recibidas no pertenece al código. En el caso de los códigos lineales es facil comprobar cuando una palabra pertenece o no al código.

Dado que los códigos lineales son subespacios vectoriales tienen asociadas unas ecuaciones:

- Ecuaciones paramétricas[2].
- Ecuaciones implícitas[3].

Para comprobar sí una palabra pertenece o no al código basta con ver sí verifica las ecuaciones que definen al código, paramétricas o implícitas.

[2] Apartado (3.5.1), en la página 31.
[3] Apartado (3.5.2), en la página 31.

3.6.1 Matriz de control

Definición 3.5 (Matriz de control)
Para un código lineal $\mathcal{C}$ diremos que una matriz M de orden $k \times n$ es una ***"matriz de control"*** *del código si verifica que $M \cdot u^t = 0$ para todo $u \in \mathbb{F}_q^n$ sí y sólo sí $u \in \mathcal{C}$.*

k puede ser cualquiera, pero normalmente $k = n - m$ ya que esa es la dimensión del subespacio incidente a $\mathcal{C}$.

La matriz de control la podemos obtener de:

- Las ecuaciones implícitas de $\mathcal{C}$.
- La base del subespacio incidente a $\mathcal{C}$.

La matriz de control es una matriz cuyas filas son una base de "vectores" incidentes a $\mathcal{C}$.

3.6.2 Forma estándar de la matriz de control

Al igual que en el caso de la matriz generadora la expresión de la **"matriz de control"** depende de la base escogida en el subespacio incidente a $\mathcal{C}$.

Definición 3.6 (Forma estándar de la matriz de control)
Sea M una matriz de control de orden $(n-m) \times n$, diremos que está en ***"forma estándar"*** *si $M = (M_1, M_2)$, donde M_2 es la matriz identidad de orden $(n-m) \times (n-m)$.*

3.7 Códigos duales

Definición 3.7 (Producto escalar en $\mathbb{F}_q^n$)
En $\mathbb{F}_q^n$ podemos definir el siguiente producto escalar:

$$\begin{array}{rccl} < , >: & \mathbb{F}_q^n \times \mathbb{F}_q^n & \longrightarrow & \mathbb{Z}^+ \\ & (x, y) & \longrightarrow & x_1 \cdot y_1 + \ldots + x_n \cdot y_n \end{array}$$

Se comprueba de forma inmediata que $< , >$ verifica las condiciones para ser producto escalar:

- $< x, y > \geq 0 \ \forall \ x, y \in \mathbb{F}_q^n$.
- $< x, x > > 0 \ \forall \ x \in \mathbb{F}_q^n$ y $u \neq 0$.

Una vez definido un producto escalar podemos definir una relación de ortoganilidad.

Definición 3.8 (Ortogonalidad entre vectores)
x e y son ortogonales sí y sólo sí $< x, y >= 0$.

Mediante esta definición de ortogonalidad podemos calcular el ortogonal a un subespacio dado.

Sea $\mathcal{C}[n, m]$ un código lineal. Entonces es un subespacio vectorial de $\mathbb{F}_q^n$ tal que $\dim_{\mathbb{F}_q} \mathcal{C}[n, m] = m$.

Definición 3.9 (Código dual)
*Se define el **"código dual"** de $\mathcal{C}[n, m]$ como el subespacio vectorial ortogonal a $\mathcal{C}[n, m]$ y lo denotaremos como $\mathcal{C}[n, m]^\perp$.*

$$\mathcal{C}[n, m]^\perp = \{ \ x \in \mathbb{F}_q^n \ \mid \ < x, \mathcal{C}[n, m] >= 0 \ \}$$

Observacion: 3.7.1

- $\mathcal{C}[n, m]^\perp = \mathcal{C}'[n, n - m]$ *ya que:*

$$\dim_{\mathbb{F}_q} \mathcal{C}[n, m]^\perp = \dim_{\mathbb{F}_q} \mathbb{F}_q^n - \dim_{\mathbb{F}_q} \mathcal{C}[n, m] = n - m$$

- *Sea G la matriz generadora del código $\mathcal{C}[n, m]$ y H la matriz generadora del código $\mathcal{C}[n, m]^\perp$ entonces se tiene que H^t es una matriz de control para el código $\mathcal{C}[n, m]$. O lo que es lo mismo:*

$$\left(\mathcal{C}[n, m]^\perp\right)^\perp = \mathcal{C}[n, m]$$

De las observaciones se deduce que el código dual determina al código:

$$\mathcal{C}[n, m] = \{\ x \in \mathbb{F}_q^n \mid H^t \cdot x^t = 0\ \}$$

donde H^t es la matriz traspuesta de la matriz generadora del código $\mathcal{C}[n, m]^{\perp}$.

Capítulo 4

Códigos cíclicos

Definición 4.1 (Códigos cíclicos)
Dado un código lineal $\mathcal{C}[n,m]$ *diremos que es un* ***"código cíclico"*** *si verifica que para toda palabra* $u \in \mathcal{C}[n,m]$*:*

$$u = (u_1, u_2 \ldots, u_{n-1}, u_n) \Longrightarrow (u_n, u_1, u_2, \ldots, u_{n-2}, u_{n-1}) \in \mathcal{C}[n,m]$$

Definición 4.2 (Permutación cíclica de orden uno)
Una ***"permutación cíclica de orden uno"*** *es una aplicación que desplaza un lugar hacía la derecha todos los elementos de una sucesión finita poniendo como primer elemento de la sucesión el último. Sí* σ *es una permutación cíclica de orden uno entonces:*

$$\sigma(\ (a_1, a_2, a_3, a_4)\ ) = (a_4, a_1, a_2, a_3)$$

Definición 4.3 (Permutación cíclica de orden k)
Una ***"permutación cíclica de orden*** k***"*** *consiste en aplicar* k *veces una permutación cíclica de orden uno.*

$$\sigma^k(\ (a_1, a_2, a_3, a_4)\ ) = (\sigma \circ \sigma \circ \overset{k}{\ldots} \circ \sigma)(\ (a_1, a_2, a_3, a_4)\ )$$

Proposición 4.1
Los códigos cíclicos son invariantes por permutaciones cíclicas.

Demostración:
Para ver que son invariantes por permutaciones cíclicas tenemos que ver aplicar una permutación cíclica de orden k, para cualquier k, a una palabra cualquiera del código produce otra palabra del código.

Aplicar una permutación cíclica de orden k equivale a aplicar k veces la permutación cíclica de orden uno, por definición.

Al aplicar la permutación cíclica de orden uno a un elemento de $\mathcal{C}[n,m]$ obtenemos otro elemento del código, ya que el código es cíclico. Sí a este elemento le volvemos a aplicar la permutación cíclica volvemos a obtener otra palabra del código y asi sucesivamente. Luego un código cíclico es invariante por permutaciones cíclicas.

■

4.1 El anillo $\mathbb{F}_q[x]/(x^n-1)$

$\mathbb{F}_q[x]/(x^n-1)$ es un anillo conmutativo con unidad y además es un $\mathbb{F}_q$-espacio vectorial[1].

$$\mathbb{F}_q[x]/(x^n-1) =< \overline{1}, \overline{x}, \ldots, \overline{x}^{n-1} >$$

como $\mathbb{F}_q$-espacio vectorial.

Sean $P(x), Q(x) \in \mathbb{F}_q[x]$ entonces son equivalentes módulo x^n-1 cuando su diferencia es un múltiplo de x^n-1.

$$P(x) \equiv Q(x)\ mod\ x^n-1 \Longleftrightarrow P(x)-Q(x) = (x^n-1)$$

Los códigos que estamos utilizando son siempre subespacios vectoriales del $\mathbb{F}_q$-espacio vectorial $\mathbb{F}_q^n$. Podemos establecer un isomorfismo entre $\mathbb{F}_q^n$ y $\mathbb{F}_q[x]/(x^n-1)$:

$$\begin{array}{rcl} \mathbb{F}_q[x]/(P(x)) & \stackrel{\sim}{\longrightarrow} & \mathbb{F}_q^n \\ a_0 + a_1 \cdot \overline{x} + \ldots + a_{n-1} \cdot \overline{x}^{n-1} & \longrightarrow & (a_0, a_1, \ldots, a_{n-1}) \end{array}$$

[1] Ejercicio $A.5$ en la página 48.

Proposición 4.2
Multiplicar por $\overline{x}$ *en* $\mathbb{F}_q[x]/(x^n-1)$ *equivale a aplicar una permutación cíclica de orden uno en* $\mathbb{F}_q^n$.

Demostración:
Sea $\overline{p(x)} = a_0 + a_1 \cdot \overline{x} + \ldots + a_{n-1} \cdot \overline{x}^{n-1}$ entonces por el isomorfismo anterior se corresponderá, de forma única, con $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_q^n$. Multipliquemos $\overline{p(x)}$ por $\overline{x}$:

$$\overline{x} \cdot \overline{p(x)} = a_0 \cdot \overline{x} + a_1 \cdot \overline{x}^2 + \ldots + a_{n-2} \cdot \overline{x}^{n-1} + a_{n-1} \cdot \overline{x}^n$$

pero como $x^n - 1 = 0$ en $\mathbb{F}_q[x]/(x^n-1)$ tenemos que $x^n = 1$ en $\mathbb{F}_q[x]/(x^n-1)$, luego:

$$\overline{x} \cdot \overline{p(x)} = a_{n-1} + a_0 \cdot \overline{x} + a_1 \cdot \overline{x}^2 + \ldots + a_{n-2} \cdot \overline{x}^{n-1}$$

el cual se corresponde, por el isomorfismo anterior, con:

$$(a_{n-1}, a_0, a_1, \ldots, a_{n-2}) \in \mathbb{F}_q^n$$

elemento que se obtiene al aplicar una permutación cíclica a $(a_0, a_1, \ldots, a_{n-1})$, elemento que representa a $\overline{p(x)}$ en $\mathbb{F}_q^n$.

■

Corolario 4.1
Multiplicar por $\overline{x}^i$ *con* $i = 0, \ldots, n-1$ *en* $\mathbb{F}_q[x]/(x^n-1)$ *equivale a aplicar una permutación cíclica de orden* i *en* $\mathbb{F}_q^n$.

Demostración:
Inmediata a partir de la proposición 4.2.

■

Teorema 4.1
Sea $\mathcal{C}[n,m] \subset \mathbb{F}_q[x]/(x^n-1)$ *un código.* $\mathcal{C}[n,m]$ *es cíclico sí y sólo sí para todo* $\overline{p(x)} \in \mathcal{C}[n,m]$ *se verifica que* $\overline{x} \cdot \overline{p(x)} \in \mathcal{C}[n,m]$.

Demostración:
Inmediata a partir de las definiciones y la proposición 4.2.

■

Corolario 4.2
Sí $\overline{p(x)} \in \mathbb{F}_q[x]/(x^n-1)$ *entonces para* $i \in \mathbb{Z}$ *se tiene que:*

$$\overline{x}^i \cdot \overline{p(x)} \in \mathbb{F}_q[x]/(x^n-1)$$

Demostración:
Es inmediato a partir del teorema 4.1 y $\overline{x}^i = \overline{x}^{(i-1)} \cdot \overline{x}$.

■

4.2 Caracterización de los códigos cíclicos

Los códigos cíclicos son ideales del anillo $\mathbb{F}_q[x]/(x^n-1)$.

Proposición 4.3
Sea $\mathcal{C}[n,m]$ un código cíclico. Para todo $\overline{p(x)} \in \mathcal{C}[n,m]$ y para cualquier $\overline{Q(x)} \in \mathbb{F}_q[x]/(x^n-1)$ se tiene que $\overline{p(x)} \cdot \overline{Q(x)} \in \mathcal{C}[n,m]$.

Demostración:
Sean:

$$\begin{aligned} \overline{p(x)} &= p_0 + p_1 \cdot \overline{x} + \ldots + p_{n-1} \cdot \overline{x}^{n-1} \quad \in \quad \mathcal{C}[n,m] \\ \overline{Q(x)} &= q_0 + q_1 \cdot \overline{x} + \ldots + q_{n-1} \cdot \overline{x}^{n-1} \quad \in \quad \mathbb{F}_q[x]/(x^n-1) \end{aligned}$$

entonces se tiene que:

$$\overline{p(x)} \cdot \overline{Q(x)} = q_0 \cdot \overline{p(x)} + q_1 \cdot \overline{x} \cdot \overline{p(x)} + \ldots + q_{n-1} \cdot \overline{x}^{n-1} \cdot \overline{p(x)}$$

como el código es cíclico se tiene que:

$$\overline{x}^i \cdot \overline{p(x)} \in \mathcal{C}[n,m] \quad i = 0, 1, \ldots$$

y como el código es lineal se tiene que:

$$\lambda \cdot \overline{h(x)} \in \mathcal{C}[n,m]$$

para $\lambda \in \mathbb{F}_q$ y $\overline{h(x)} \in \mathcal{C}[n,m]$. Luego $\overline{p(x)} \cdot \overline{Q(x)}$ es la suma de polinomios que pertenecen al código y por la linealidad de este la suma de elementos del código pertenece al código.

■

Teorema 4.2 (Caracterización de los códigos cíclicos)
Un código $\mathcal{C}[n,m]$ es cíclico sí y sólo sí como subconjunto es un ideal de $\mathbb{F}_q[x]/(x^n-1)$.

Demostración:
Es inmediato a partir de la definición de ideal y de la proposición 4.3.

■

Según este teorema todos los códigos cíclicos son ideales del anillo, de ideales principales, $\mathbb{F}_q[x]/(x^n-1)$. Luego todos sus ideales estarán generados por un elemento, es decir serán de la forma $(\overline{h(x)})$.

En el ejercicio *A*.5, en la página 48, se demuestra que todos los ideales del anillo $\mathbb{F}_q[x]/(x^n-1)$ están generados por los productos de los factores irreducibles en los que descompone x^n-1 en $\mathbb{F}_q[x]$.

Definición 4.4 (Polinomio generador del código cíclico)
Se llama ***"polinomio generador del código cíclico"*** *al polinomio que genera el ideal al que corresponde el código.*

4.3 Construcción de códigos cíclicos

Sea $\mathcal{C}[n,m]$ un código lineal cíclico. Tendremos entonces que:

$$\mathcal{C}[n,m] \subset \mathbb{F}_q^n \quad y \quad \mathcal{C}[n,m] \simeq \mathbb{F}_q^m$$

Además como el código es cíclico se corresponde con un ideal del anillo $\mathbb{F}_q[x]/(x^n-1)$:

$$\mathcal{C}[n,m] = (g(x)) \quad con\ g(x) = a_0 + a_1 \cdot \overline{x} + \ldots + a_{m-1} \cdot \overline{x}^{m-1}$$

4.3.1 Matriz generadora de un código cíclico

La matriz generadora del código cíclico $\mathcal{C}[n,m]$ será la matriz de la aplicación lineal:

$$\mathbb{F}_q^m \xrightarrow{\sim} \mathcal{C}[n,m]$$

Observar que $\mathbb{F}_q[x]/(x^n-1)$ es un $\mathbb{F}_q$-espacio vectorial, además de anillo, por lo cual cualquier ideal suyo es también subespacio vectorial.

Por el teorema 4.2 sabemos que los códigos cíclicos son los ideales del anillo $\mathbb{F}_q[x]/(x^n-1)$, o lo que es lo mismo su polinonio generador divide a x^n-1, sobre $\mathbb{F}_q$.

Sea $x^n-1 = p_1(x) \cdot p_r(x)$ la descomposición en irreducibles sobre $\mathbb{F}_q[x]$, y sea $g(x) = p_1(x) \cdot p_i(x)$ con $i < r$, entonces tendremos que $x^n-1 = g(x) \cdot h(x)$ con:

$$\begin{aligned} g(x) &= g_0 + g_1 \cdot x + \ldots + g_{n-k} \cdot x^{n-m} \\ h(x) &= h_0 + h_1 \cdot x + \ldots + h_k \cdot x^m \end{aligned}$$

Sea $\mathcal{C} = (g(x))$ y llamemos $A = \mathbb{F}_q[x]/(x^n-1)$, entonces:

$$A/\mathcal{C} = \mathbb{F}_q[x]/(g(x))$$

$\mathbb{F}_q[x]/(g(x))$ es de dimensión $n-m$, luego el código será de dimensión m. Para encontrar una base nos bastaría con encontrar m vectores linealmente

independientes entonces formarían base al estar en un subespacio de dimensión m.

$$\mathcal{C} =< g(x), x \cdot g(x), \ldots . x^{m-1} \cdot g(x) >$$

Que son base es inmediato, ya que al ser $\mathcal{C}$ un código cíclico y $g(x) \in \mathcal{C}$ entonces $x^i \cdot g(x) \in \mathcal{C}$. Dichos polinomios expresados en la base canónica de A serán:

$$\begin{aligned} g(x) &= (g_0, \ldots, g_{n-m}, 0, \overset{m-1)}{\ldots}, 0) \\ x \cdot g(x) &= (0, g_0, \ldots, g_{n-m}, 0, \overset{m-2)}{\ldots}, 0) \\ \ldots &\quad \ldots\ldots\ldots \\ x^{m-1} \cdot g(x) &= (0, \overset{m-1)}{\ldots}, 0, g_0, \ldots, g_{n-m}) \end{aligned}$$

que son m vectores linealmente independientes, luego generan. La matriz generadora del código será:

$$\begin{pmatrix} g_0 & 0 & \ldots & 0 \\ g_1 & g_0 & \ldots & 0 \\ g_2 & g_1 & \ldots & \vdots \\ \vdots & \vdots & \vdots & g_0 \\ g_{n-m} & \vdots & \vdots & \vdots \\ 0 & g_{n-m} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \vdots & g_{n-m} \end{pmatrix}$$

Luego tenemos un código de dimensión m y con palabras de longitud n, luego nuestro código será del tipo $\mathcal{C}[n, m]$.

4.3.2 Código dual de un código cíclico

Sea $x^n - 1 = g(x) \cdot h(x)$ y $\mathcal{C}[n, m] = (g(x))$ entonces definimos el código dual de $\mathcal{C}[n, m]$, y lo denotamos como $\mathcal{C}^{\perp}[n, m]$, al código cíclico generado por el polinomio $x^k \cdot h(x^{-1})$.

La matriz generadora de $\mathcal{C}^{\perp}[n, m]$ es la matriz de control de $\mathcal{C}[n, m]$.

4.4 Códigos de Reed-Solomon

Sea β una raiz primitiva n-ésima de la unidad[2] en $\mathbb{F}_q$ y $f(x)$ su polinomio mínimo, entonces:

$$x^n - 1 = (x-1) \cdot f_1(x) \cdot f_k(x)$$

definamos el polinomio $g(x)$ como el mínimo comúm multiplo de los polinomios mínimos $f_1, \ldots, f_k$. Entonces el código generado por dicho polinomio es un código *BCH*.

Si tomamos $n = q - 1$, $q = 2^k$, el código resultante se llama código de *Reed-Solomon*.

[2]Cualquier otra raiz primitiva n-ésima de la unidad es una potencia suya.

Apéndice A

Ejercicios propuestos en clase

Ejercicio A.1
Dados $n, m \in \mathbb{Z}$, $n > 0$ y $m \neq 0$. Probar que la condición necesaria y suficiente para que exista un número entero m' tal que $m \cdot m' \equiv 1$ mod n es que m y n sean primos entre sí.

Solución:

1. Supongamos que existe $m' \in \mathbb{Z}$ tal que $m \cdot m' \equiv 1 \bmod n$ y veamos que m y n son primos entre sí.

   Por hipótesis tendremos que:

   $$m \cdot m' = \overline{m} \cdot n + 1 \Longrightarrow m \cdot m' + (-\overline{m}) \cdot n = 1 \quad \overline{m} \in \mathbb{Z} \qquad (A.1)$$

   Por la *Identidad de Bezout* y $(A.1)$ tenemos que m y n son primos entre sí.

2. Supongamos que m y n son primos entre sí y veamos que existe un número entero, m', tal que $m \cdot m' \equiv 1 \bmod n$.

   Como m y n son primos por hipótesis aplicando la *Identidad de Bezout* tendremos que existen $a, b \in \mathbb{Z}$ tales que:

   $$a \cdot m + b \cdot n = 1$$

   es decir:

   $$a \cdot m = (-b) \cdot n + 1 \Longrightarrow a \cdot m \equiv 1 \ mod \ n$$

   Tomando a como m' hemos terminado.

■

Ejercicio A.2 *Comprobar que la distancia de Hamming es una distancia.*

Solución:
Sea $\mathcal{C}$ un código, entonces la distancia de Hamming será una aplicación de la forma:

$$d : \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{Z}^+$$

Recordemos que la distancia de Hamming, entre dos palabras, es el número de bits en el que no coinciden.

- $d(x,y) \geq 0$ y $d(x,y) = 0 \Longleftrightarrow x = y$
  Como la distancia entre dos palabras es el número de bits en el que no coinciden esta distancia siempre será mayor o igual que cero.

  La distancia entre dos palabras será cero sí y sólo sí coinciden en todos sus bits, luego ambas palabras serán las mismas.

- $d(x,y) = d(y,x)$
  El número de bits en el que no coincide x con y es el mismo que el de y con x.

- Desigualdad triangular: $d(x,z) \leq d(x,y) + d(y,z)$
  Sean $x, y, z \in \mathcal{C}$ tres palabras cualesquiera del código.

  Definamos los siguientes conjuntos:

  $$U = \{i \mid x_i \neq z_i\}$$

  U es el conjunto de indices en los que x y z no coinciden.

  $$S = \{i \mid x_i \neq z_i \ y \ x_i = y_i\}$$

  S es el conjunto de indices en los que x y z no coinciden y en los que x e y coinciden.

  $$T = \{i \mid x_i \neq z_y \ y \ x_i \neq y_i\}$$

  T es el conjunto de indices en los que x y z no coinciden y en los que x e y tampoco coinciden.

  Entenderemos por $\#U$, $\#S$ y $\#T$ al número de elementos de los conjuntos U, S y T respectivamente.

De las definiciones anteriores se deduce que:

$$\#U = d(x,z) = \#S + \#T \tag{A.2}$$

Como $d(x,y)$ es el número de indices en los que no coinciden x e y tenemos entonces:

$$d(x,y) \geq \#T \tag{A.3}$$

De la definición de $d(y,z)$ y de S se tiene que:

$$d(y,z) \geq \#S \tag{A.4}$$

Si sumamos miembro a miembro $(A.3)$ y $(A.4)$ y tenemos en cuenta $(A.2)$ tenemos:

$$d(x,z) \leq d(x,y) + d(y,z)$$

que es lo que queriamos demostrar.

■

Ejercicio A.3 *Dado un código $\mathcal{C} \subset \mathbb{F}_q^n$ tal que $d_{min} = 2 \cdot t + 1$ comprobar que si dado $z \in \mathbb{F}_q^n$ tal que existe $c \in \mathcal{C}$ verificando que $d(z,c) \leq t$ entonces la distancia de z a cualquier otra palabra del código es estrictamente mayor que t.*

Solución:
Dado $z \in \mathbb{F}_q^n$ supongamos que existe $c \in \mathcal{C}$ tal que $d(z,c) \leq t$. Ahora bien como $d_{min} = 2 \cdot t + 1$ tendremos que $d(c,c') \geq 2 \cdot t + 1 \ \forall \ c' \in \mathcal{C}$ con $c' \neq c$. Por la desigualdad triangular tendremos:

$$2 \cdot t + 1 \leq d(c,c') \leq d(c,z) + d(z,c')$$

para todo $z \in \mathbb{F}_q^n$, en particular para nuestro z. Como por hipótesis se tiene que $d(z,c) \leq t$:

$$2 \cdot t + 1 \leq t + d(z,c')$$

o lo que es lo mismo:

$$t + 1 \leq d(z,c') \Longrightarrow d(z,c') > t \quad \forall \ c' \neq c \in \mathcal{C}$$

■

Este ejercicio nos indica que si conocemos la distancia mínima, d_{min}, de un código $\mathcal{C}$ y calculamos t de tal manera que verifique $d_{min} = 2 \cdot t + 1$ entonces en el caso de recibir una transmisión incorrecta, z, tal que $d(z, c) \leq t$ se tiene que cualquier otra palabra del código, c', verifica que $d(z, c') > t$. Esto quiere decir que podemos corregir el error cometido ya que sólo existe una palabra del código a distancia t de w, entonces esa palabra del código es la que se transmitió originalmente, luego c es la palabra transmitida originalmente.

Ejercicio A.4
Dada la siguiente matriz de control de un código lineal binario:

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

calcular el código al que corresponde. Dicho código es el código de Hamming $Ham(3)$*, el cual es* 1*-perfecto.*

Solución:
Dado que H es la matriz de control de un código lineal por definición se tiene que cualquier palabra del código aplicada a la matriz será cero:

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

De donde obtenemos las ecuaciones implícitas del código:

$$\begin{aligned} x_5 &= x_1 + x_3 + x_4 \\ x_6 &= x_1 + x_2 + x_3 \\ x_7 &= x_2 + x_3 + x_4 \end{aligned}$$

Recordar que estamos trabajando sobre $\mathbb{F}_2$, donde tenemos que $-1 \equiv 1$.

Luego tenemos un código lineal con longitud de palabra 7 en el que los tres últimos bits vienen determinados por los cuatro primeros, son función lineal[1], luego son combinación lineal de los cuatro primeros bits. Es decir el código lineal es de dimensión cuatro. Luego el código buscado es un código $\mathcal{C}[7,4]$.

Para calcular todas las palabras del código basta con encontrar cuatro vectores linealmente independientes, ya que al ser el código de dimensión cuatro vectores linealmente independientes, que pertenezcan al código, generarán el código. Con el fin de dar la matriz generadora en forma estándar cogeremos los vectores $\{e_i\}_{i=1}^4$ donde de los cuatro primeros bits unicamente sea no nulo el i-esimo y los bits del quinto al septimo estarán determinados por las ecuaciones implícitas del código.

$$\begin{aligned} e_1 &= (1,0,0,0,1,1,0) \\ e_2 &= (0,1,0,0,0,1,1) \\ e_3 &= (0,0,1,0,1,1,1) \\ e_4 &= (0,0,0,1,1,0,1) \end{aligned}$$

Luego la matriz generadora, en forma estándar, del código será:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Como el código es de dimensión cuatro y estamos trabajando sobre un cuerpo finito de dos elementos, $\mathbb{F}_2$, tendremos que el código tendrá $2^4 = 16$ palabras. La forma de calcularlas será coger todas las posibles combinaciones, 16, de cuatro elementos de $\mathbb{F}_2$ y añadirle otros tres bits, los determinados por las ecuaciones implícitas del código.

■

[1] La suma de elementos es una función lineal.

Ejercicio A.5
Sea $P(x)$ un polinomio en x de grado n y con coeficientes en $\mathbb{F}_q$.

1. *Probar que $\mathbb{F}_q[x]/(P(x))$ es un espacio vectorial y que:*

$$\mathbb{F}_q[x]/(P(x)) =< \overline{1}, \overline{x}, \ldots, \overline{x}^{n-1} >$$

2. *Calcular los ideales del anillo $\mathbb{F}_q[x]/(x^n - 1)$.*

Los ideales del anillo $\mathbb{F}_q[x]/(x^n - 1)$ clasifican los códigos cíclicos.

Solución:

1. Sea $P(x)$ un polinomio en x de grado n y con coeficientes en $\mathbb{F}_q$.

   $\mathbb{F}_q[x]/(P(x))$ son los restos al dividir polinomios en x con coeficientes en $\mathbb{F}_q$ por $P(x)$. Luego serán polinomios en x con coeficientes en $\mathbb{F}_q$ y de grado menor o igual que $n-1$, al tener $P(x)$ grado n.

   Denotaremos a los elementos de $\mathbb{F}_q[x]/(P(x))$ como $\overline{p_i(x)}$. Entonces por la definicón de $\mathbb{F}_q[x]/(P(x))$ y por el *algoritmo de Euclides* tendremos que:

$$h(x) = h_i(x) \cdot P(x) + p_i(x)$$

   Luego $\overline{p_i(x)}$ representará, en $\mathbb{F}_q[x]/(P(x))$, a todos aquellos polinomios $h(x) \in \mathbb{F}_q[x]$ que tengan resto $p_i(x)$ al dividir por $P(x)$.

   Las operaciones de suma de polinomios, en $\mathbb{F}_q[x]/(P(x))$, así como multiplicar polinomios, de $\mathbb{F}_q[x]/(P(x))$, por escalares están bien definidas. Es de inmediata comprobación.

(a) $\mathbb{F}_q[x]/(P(x))$ es un $\mathbb{F}_q$-espacio vectorial.

i. $(\mathbb{F}_q[x]/(P(x)), +)$ es un grupo conmutativo.

Es inmediato, se deduce de la estructura de grupo conmutativo de $(\mathbb{F}_q, +)$ y de la definición de $\mathbb{F}_q[x]/(P(x))$.

ii. $\lambda \cdot (\overline{p_1(x)} + \overline{p_2(x)}) = \lambda \cdot \overline{p_1(x)} + \lambda \cdot \overline{p_2(x)}$ para cualesquiera que sean $\lambda \in \mathbb{F}_q$, $\overline{p_1(x)}, \overline{p_2(x)} \in \mathbb{F}_q[x]/(P(x))$.

Inmediato a partir de las definiciones.

iii. $(\lambda + \mu) \cdot \overline{p(x)} = \lambda \cdot \overline{p(x)} + \mu \cdot \overline{p(x)}$ para cualesquiera que sean $\lambda, \mu \in \mathbb{F}_q$ y $\overline{p(x)} \in \mathbb{F}_q[x]/(P(x))$.

Inmediato a partir de las definiciones.

iv. $(\lambda \cdot \mu) \cdot \overline{p(x)} = \lambda \cdot (\mu \cdot \overline{p(x)})$ para cualesquiera $\lambda, \mu \in \mathbb{F}_q$ y $\overline{p(x)} \in \mathbb{F}_q[x]/(P(x))$.

Inmediato a partir de las definiciones.

v. $1 \cdot \overline{p(x)} = \overline{p(x)}$ para todo $\overline{p(x)} \in \mathbb{F}_q[x]/(P(x))$.

Inmediato a partir de las definiciones.

(b) $\mathbb{F}_q[x]/(P(x)) = < \overline{1}, \overline{x}, \ldots, \overline{x}^{n-1} >$.

i. $< \overline{1}, \overline{x}, \ldots, \overline{x}^{n-1} >$ son un sistema de generadores.

Sea $\overline{p(x)} \in \mathbb{F}_q[x]/(P(x))$, luego su grado es menor que el grado de $P(x)$, que es n, luego es de grado menor o igual que $n-1$. Es obvio que $\overline{p(x)}$ es combinación lineal de $\{\overline{1}, \overline{x}, \ldots, \overline{x}^{n-1}\}$, ya que sí $\overline{p(x)} = a_0 + a_1 \cdot \overline{x} + \ldots + a_{n-1} \cdot \overline{x}^{n-1}$ tendremos que:

$$\overline{p(x)} = a_0 \cdot \overline{1} + a_1 \cdot \overline{x} + \ldots + a_{n-1} \cdot \overline{x}^{n-1}$$

con $a_0, a_1, \ldots, a_{n-1} \in \mathbb{F}_q$.

Luego $< \overline{1}, \overline{x}, \ldots, \overline{x}^{n-1} >$ son un sistema de generadores.

ii. $< \overline{1}, \overline{x}, \ldots, \overline{x}^{n-1} >$ son linealmente independientes.

Sean $\lambda_0, \ldots, \lambda_{n-1} \in \mathbb{F}_q$ tales que:

$$\lambda_0 \cdot \overline{1} + \lambda_1 \cdot \overline{x} + \ldots + \lambda_{n-1} \cdot \overline{x}^{n-1} = 0$$

Como la combinación lineal es cero entonces debe ser un múltiplo de $P(x)$, pero como la combinación lineal es un polinomio de grado $n-1$, tiene menor grado que $P(x)$, entonces la combinación lineal es el polinomio nulo, o lo que es lo mismo:

$$\lambda_0 = \lambda_1 = \ldots = \lambda_{n-1} = 0$$

luego $< \overline{1}, \overline{x}, \ldots, \overline{x}^{n-1} >$ son linealmente independientes.

2. Calcular los ideales del anillo $\mathbb{F}_q[x]/(x^n - 1)$.

Para calcular los ideales del anillo $\mathbb{F}_q[x]/(x^n-1)$ utilizaremos el siguiente teorema del albebra:

Teorema A.1
Sea $\phi : A \longrightarrow B$ un morfismo epiyectivo de anillos. Entonces existe la siguiente correspondencia biunívoca:

$$\begin{array}{ccc} \left[\begin{array}{c} Ideales \\ de\ B \end{array}\right] & \stackrel{\sim}{\longrightarrow} & \left[\begin{array}{c} Ideales\ de\ A\ que \\ continen\ a\ \ker\phi \end{array}\right] \\ I & \longmapsto & \phi^{-1}(I) \end{array}$$

Consideremos ahora la aplicación de paso al cociente:

$$\pi : \mathbb{F}_q[x] \longrightarrow \mathbb{F}_q[x]/(x^n - 1)$$

Es inmediato demostrar que π es un morfismo epiyectivo de anillos y que $\ker \pi = (x^n - 1)$. Luego por el teorema $A.1$ tendremos que los ideales de $\mathbb{F}_q[x]/(x^n - 1)$ se identificarán con los ideales de $\mathbb{F}_q[x]$ que contienen a $(x^n - 1)$.

Sea $x^n - 1 = p_1(x) \cdot \ldots \cdot p_r(x)$ la descomposición en irreducibles de $x^n - 1$ en $\mathbb{F}_q[x]$. Luego los ideales que contienen a $x^n - 1$ serán los ideales generados por los productos de los factores irreducibles en los que descompone $x^n - 1$.

Los ideales de $\mathbb{F}_q[x]/(x^n - 1)$ son los ideales generados en dicho anillo por los polinomios divisores de $x^n - 1$.

■

Ejercicio A.6
Sea $\mathbb{F}_2[x]/(x^7-1)$*, donde la descomposición de* x^7-1 *en irreducibles en* $\mathbb{F}_2[x]$ *es la siguiente:*

$$x^7 - 1 = (x+1)\cdot(x^3+x+1)\cdot(x^3+x^2+1)$$

Calcular la matriz generadora para el código cíclico que tiene por polinomio generador:

$$g(x) = (x+1)\cdot(x^3+x+1) = x^4+x^3+x^2+1$$

Solución:
La dimensión del código será el grado de x^3+x^2+1, luego será de dimensión tres.

Una base del código será:

$$\begin{aligned} g(x) &= x^4+x^3+x^2+1 \\ x\cdot g(x) &= x^5+x^4+x^3+x \\ x^2\cdot g(x) &= x^6+x^5+x^4+x^2 \end{aligned}$$

Luego la matriz generadora del código será:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

■

Ejercicio A.7
Sea $\mathbb{F}_2[x]/(x^7-1)$, donde la descomposición de x^7-1 en irreducibles en $\mathbb{F}_2[x]$ es la siguiente:

$$x^7 - 1 = (x+1)\cdot(x^3+x+1)\cdot(x^3+x^2+1)$$

Calcular las matrices generadoras y de control para todos los códigos que podemos definir.

Solución:
Los códigos que podemos definir son los definidos por los divisores de x^7-1 luego serán:

- $g(x) = x+1$, la dimensión del código será 6.

  Una base del código será:

$$\begin{aligned} g(x) &= x+1 \\ x\cdot g(x) &= x^2+x \\ x^2\cdot g(x) &= x^3+x^2 \\ x^3\cdot g(x) &= x^4+x^3 \\ x^4\cdot g(x) &= x^5+x^4 \\ x^5\cdot g(x) &= x^6+x^5 \end{aligned}$$

  Luego la matriz del código será:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

  La matriz de control será la traspuesta de la matriz generadora de su código dual. Su código dual estará generado por:

$$p(x) = x^6\cdot h(\frac{1}{x}) = 1+x+x^2+x^3+x^4+x^5+x^6$$

  con $h(x) = (x^3+x+1)\cdot(x^3+x^2+1)$. Luego una base del código dual será:

$$p(x) = 1+x+x^2+x^3+x^4+x^5+x^6$$

Luego la matriz de control de este código será:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- $g(x) = x^3 + x + 1$, la dimensión del código será 4.

  Una base del código será:

$$\begin{aligned} g(x) &= x^3 + x + 1 \\ x \cdot g(x) &= x^4 + x^2 + x \\ x^2 \cdot g(x) &= x^5 + x^3 + x^2 \\ x^3 \cdot g(x) &= x^6 + x^4 + x^4 \end{aligned}$$

  Luego la matriz del código será:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  La matriz de control será la matriz traspuesta de la matriz de control de su código dual. Su código dual estará generado por:

$$p(x) = x^4 \cdot h(\frac{1}{x}) = 1 + x^2 + x^3 + x^4$$

  con $h(x) = (x + 1) \cdot (x^3 + x^2 + 1)$. Una base del código dual será:

$$\begin{aligned} p(x) &= x^4 + x^3 + x^2 + 1 \\ x \cdot p(x) &= x^5 + x^4 + x^3 + x \\ x^2 \cdot p(x) &= x^6 + x^5 + x^5 + x^2 \end{aligned}$$

  Luego la matriz de control será:

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

- $g(x) = x^3 + x^2 + 1$, la dimensión del código será 4.

  Una base del código será:

$$\begin{aligned} g(x) &= x^3 + x^2 + 1 \\ x \cdot g(x) &= x^4 + x^3 + x \\ x^2 \cdot g(x) &= x^5 + x^4 + x^2 \\ x^3 \cdot g(x) &= x^6 + x^5 + x^3 \end{aligned}$$

  Luego la matriz del código será:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  La matriz de control del código será la traspuesta de la matriz generadora de su código dual. Su código dual estará generado por:

$$p(x) = x^4 \cdot h(\frac{1}{x}) = 1 + x + x^2 + x^4$$

  con $h(x) = (x+1) \cdot (x^3 + x + 1)$. Una base del código dual será:

$$\begin{aligned} p(x) &= x^4 + x^2 + x + 1 \\ x \cdot p(x) &= x^5 + x^3 + x^2 + x \\ x^2 \cdot p(x) &= x^6 + x^4 + x^3 + x^2 \end{aligned}$$

  Luego la matriz de control será:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

- $g(x) = (x+1) \cdot (x^3 + x + 1)$, la dimensión del código será 3.

$$g(x) = (x+1) \cdot (x^3 + x + 1) = x^4 + x^3 + x^2 + 1$$

Una base del código será:

$$\begin{aligned} g(x) &= x^4 + x^3 + x^2 + 1 \\ x \cdot g(x) &= x^5 + x^4 + x^3 + x \\ x^2 \cdot g(x) &= x^6 + x^5 + x^4 + x^2 \end{aligned}$$

Luego la matriz del código será:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

La matriz de control será la traspuesta de la matriz generadora de su código dual. Su código dual estará generado por:

$$p(x) = x^3 \cdot h(\frac{1}{x}) = 1 + x + x^3$$

con $h(x) = x^3 + x^2 + 1$. Una base del código dual será:

$$\begin{aligned} p(x) &= x^3 + x + 1 \\ x \cdot p(x) &= x^4 + x^2 + x \\ x^2 \cdot p(x) &= x^5 + x^3 + x^2 \\ x^3 \cdot p(x) &= x^6 + x^4 + x^3 \end{aligned}$$

Luego la matriz de control será:

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

- $g(x) = (x+1) \cdot (x^3 + x^2 + 1)$, la dimensión del código será 3.

$$g(x) = (x+1) \cdot (x^3 + x^2 + 1) = 1 + x + x^2 + x^4$$

Una base del código será:

$$\begin{aligned} g(x) &= x^4 + x^2 + x + 1 \\ x \cdot g(x) &= x^5 + x^3 + x^2 + x \\ x^2 \cdot g(x) &= x^6 + x^4 + x^3 + x^2 \end{aligned}$$

Luego la matriz del código será:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

La matriz de control será la traspuesta de la matriz generadora de su código dual. Su código dual estará generado por:

$$p(x) = x^3 \cdot h(\frac{1}{x}) = 1 + x^2 + x^3$$

con $h(x) = x^3 + x + 1$. Una base del código dual será:

$$\begin{aligned} p(x) &= x^3 + x^2 + 1 \\ x \cdot p(x) &= x^4 + x^3 + x \\ x^2 \cdot p(x) &= x^5 + x^4 + x^2 \\ x^3 \cdot p(x) &= x^6 + x^5 + x^3 \end{aligned}$$

Luego la matriz de control del código será:

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

- $g(x) = (x^3 + x + 1) \cdot (x^3 + x^2 + 1)$, la dimensión del código será 1.

  $g(x) = (x^3 + x + 1) \cdot (x^3 + x^2 + 1) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$

  Una base del código será:

  $$g(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

  Luego la matriz del código será:

  $$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

La matriz de control será la traspuesta de la matriz generadora de su código dual. Su código dual estará generado por:

$$p(x) = x \cdot h(\frac{1}{x}) = 1 + x$$

con $h(x) = x + 1$. Una base del código dual será:

$$\begin{aligned} p(x) &= x + 1 \\ x \cdot p(x) &= x^2 + x \\ x^2 \cdot p(x) &= x^3 + x^2 \\ x^3 \cdot p(x) &= x^4 + x^3 \\ x^4 \cdot p(x) &= x^5 + x^4 \\ x^5 \cdot p(x) &= x^6 + x^5 \end{aligned}$$

Luego la matriz de control del código será:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

■