Teoría de la Información y Teoría de Códigos

José Angel de Bustos Pérez
Curso $1.999 - 2.000$

Clases de D. José Angel Domínguez Pérez

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.

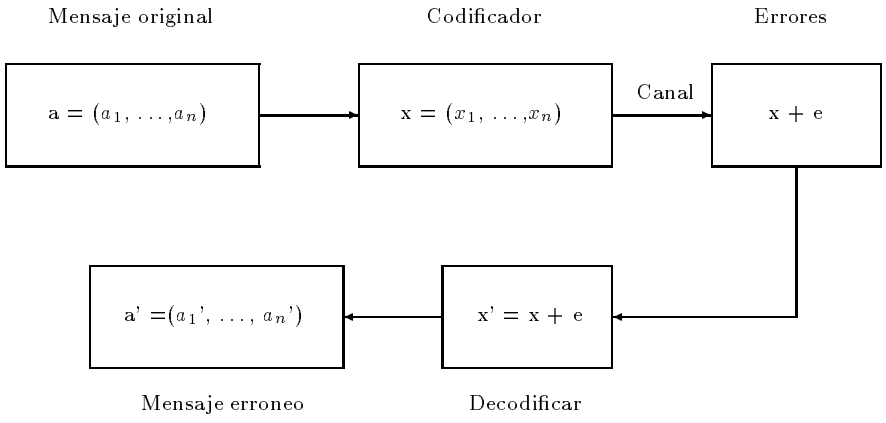

Figura 1.1: Transmisión de la información.

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 Los ejemplos fundamentales

Todo lo expuesto en estos apuntes será aplicado a los siguientes ejemplos:

- Bit de control de paridad.
- Triple repetición.
- Triple control.

De igual modo los códigos que utilizaremos estarán formados por palabras compuestas de 0 y 1, es decir los simbolos serán $\{0, 1\}$.

En la práctica la información redundante que introduciremos en los códigos se introduce al principio de la palabra, pero nosotros la introduciremos al final[1].

1.4.1 Código binario del "bit de control de paridad"

Este código estará formado por palabras de 8 simbolos.

$$\mathcal{C} = \{(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8) \mid a_i \in \mathbb{F}_2 \quad \forall \ i = 1, \ldots, 8\}$$

Obviamente se tiene que $\mathcal{C} \subset \mathbb{F}_2^8$, ya que sí se diera la igualdad unicamente estaríamos codificando la información sin introducir información redundante[2].

[1] Por comodidad

[2] Lo cual no sería efectivo pues no podríamos conocer cuando se ha cometido un error en la transmisión.

Ahora bien, como hemos comentado antes introduciremos en el código información redundante. Esto quiere decir que en los ocho digitos que componen cada palabra del código que vamos a utilizar tendremos unos digitos que no aportan nada al mensaje que queremos transmitir, su única utilidad será la de darnos información sobre la palabra que hemos transmitido y poder comprobar si hubo algún error en la transmisión de dicha palabra.

En este código se utiliza un único "bit" para transmitir esa información redundate. Con lo cual únicamente utilizaremos siete "bits" para codificar información y otro "bit" adicional para dar información sobre la palabra transmitida. Según esto el número de palabras de nuestro código será de $2^7 = 128$, ya que son siete los "bits" que se emplean para codificar información:

$$\mathcal{C} = \{(a_1, a_2, a_3, a_4, a_5, a_6, a_7, c_1) \mid a_i, c_1 \in \mathbb{F}_2 \quad \forall\, i = 1, \ldots, 7\}$$

El "bit" c_1 de las palabras de este código recibe el nombre de "bit de control" y aporta información sobre los siete "bits" anteriores.

Una vez conocidos $a_1a_2a_3a_4a_5a_6a_7$ la forma de determinar c_1 es la siguiente:

> c_1 será tal que en la palabra haya un número par de 1, es decir, si en los "bits" anteriores hay un número impar de 1 $c_1 = 1$, en caso contrario $c_1 = 0$.

Palabra	Bits de información	Bit de control	¿Palabra correcta?
01101111	0110111	1	Sí
11101000	1110100	0	Sí
00011000	0001100	0	Sí
11100011	1110001	1	No
00010000	0001000	0	No
01010100	0101010	0	No

Tabla 1.1: Ejemplos del código del bit de control de paridad.

Podemos ver que la información redundante, "bit" de control, lo hemos situado al final de la palabra, es decir, es el "bit" menos significativo. En la práctica esto no ocurre ya que los "bits" de control se situan al principio de la palabra, son los "bits" más significativos.

Ventajas del código

Este código tiene las siguientes ventajas:

- Añade poca información redundate, unicamente $\frac{1}{8}$ de la información transmitida es redundante.
- Detecta un error. Si en la transmisión se transmite un "bit" de forma erronea[3] entonces el "bit" de control no concuerda con los anteriores, se produjo un error.

En realidad si se produce un número impar de errores el código lo detecta, pero no hay forma de saber cuantos errores se han cometido, por lo cual se supone que se ha cometido un error.

Inconvenientes del código

Este código tiene los siguientes inconvenientes:

- Aunque se detecte un error no hay forma de saber en cual de los siete "bits" de información se ha producido, luego no corrige errores.
- Si se produce un número par de errores la palabra resultante es otra palabra del código, con lo cual no se detecta el error ya que se da por buena.

1.4.2 Código binario de "triple repetición"

Este código consiste en repetir cada "bit" de información tres veces.

Bit de información	Palabra del código
0	000
1	111

Tabla 1.2: Palabras del código de triple repetición.

$$\mathcal{C} = \{000, 111\} \subset \mathbb{F}_2^3$$

[3]Un 1 pasa a ser un 0 y viceversa.

Ventajas del código

Este código tiene las siguientes ventajas:

- Detecta hasta dos errores.
- Corrige un error.

Estamos suponiendo que en cada "bit" unicamente se produce un error, observar que si en un "bit" se produce un número par de errores el "bit" queda inalterado.

Para poder corregir, con este código, necesitamos una hipótesis adicional sobre la probabilidad de errores en el canal. Es decir necesitamos conocer que error es más probable que se cometa en la transmisión en el canal, dedicandonos a corregir el error que, con más probabilidad, se cometa.

Supongamos que se realiza una transmisión utilizando este código y se recibe 010. Sabiendo que en el canal es más probable que se cometan dos errores que uno entonces tendremos que la palabra que, con más probabilidad, se transmitió es 111. Esto no significa que la palabra transmitida fuera 111, ya que existe una probabilidad $1 - P$, donde P es la probabilidad de que se cometieran dos errores en la transmisión, de que se produzca un error en la transmisión. La palabra transmitida pudiera haber sido 000 y haberse cometido "un solo" error. Por eso en este código lo que se hace es corregir el error que tiene mayor probabilidad de cometerse, existiendo siempre un margen de error en la corrección. Dicho margen viene dado por la probabilidad de cometerse el error que no corregimos.

Si se produjeran tres errores, uno en cada "bit", obtendríamos la otra palabra del código, es decir no detectariamos el error.

Inconvenientes del código

Este código tiene los siguientes inconvenientes:

- Mucha información redundante. Este código es lento transmitiendo, razón por la cual no es efectivo ni práctico.

1.4.3 Código binario de "triple control"

De este código se podría decir que es una mezcla de los dos códigos anteriores. Consiste en dividir la información a transmitir en bloques de tres "bits" y añadirle otros tres "bits" de control. Según lo dicho nuestro código será $\mathcal{C} \subset \mathbb{F}_2^6$. Como la información se transmite en bloques de tres "bits"[4] el código tendrá $2^3 = 8$ palabras.

El código será:

$$\mathcal{C} = \{(a_1, a_2, a_3, c_1, c_2, c_3) \; donde \; a_i, c_i \in \mathbb{F}_2\}$$

Los "bits" de control c_1 ,c_2 y c_3 están determinados de la siguiente manera:

- c_1 es tal que el número de 1 en $a_1 a_2 c_1$ sea par.
- c_2 es tal que el número de 1 en $a_1 a_3 c_2$ sea par.
- c_3 es tal que el número de 1 en $a_2 a_3 c_3$ sea par.

Palabra	Bits de control
000000	000
001011	011
010101	101
100110	110
011110	110
101101	101
110011	011
111000	000

Tabla 1.3: Palabras del código de triple control.

Ventajas del código

Este código tiene las siguientes ventajas:

- Detecta un error y lo corrige.

[4]Los otros tres "bits" son información redundante.

Inconvenientes del código

Este código tiene los siguientes inconvenientes:

- La mitad de la información que se transmite es redundante.

1.5 ¿Como modelizar la situación?

Para poder elaborar una teoría que nos permita construir "buenos códigos"[5] con los que podamos transmitir información necesitaremos introducir una serie de hipótesis sobre el modelo.

1.5.1 Primera hipótesis: "Canal discreto"

Supondremos que en el canal por el que se transmite la información se transmite "a golpes" discretos. Es decir con "paquetes" a los que llamaremos *BLOQUES* o *PALABRAS*.

Resumiendo la información se divide en *PALABRAS* que se transmiten de forma separada, es decir no se transmiten todas juntas. Las *PALABRAS* de nuestro mensaje estarán formadas por *SIMBOLOS*, *BITS* o *LETRAS*.

1.5.2 Segunda hipótesis: "Errores aleatorios"

El tipo de canal por el que se efectua la transmisión introduce "*errores aleatorios*".

En la realidad los errores no son asi, son "*a rafagas*", es decir se deja de enterder desde que empieza el "*ruido*" hasta que termina.

Nuestro modelo supondrá que, dadas dos palabras A y B **existe una probabilidad (fija)** de que se transmita A y se reciba B. Esta probabilidad la denotaremos como $P_{A,B}$.

Definición 1.1 (Canal Simétrico)
Diremos que un canal es ***"simétrico"*** *cuando* $P_{A,B} = P_{B,A}$*, es decir, la probabilidad de que* A *pase a ser* B *es la misma probabilidad de que* B *pase a ser* A*.*

[5] Que detecten y corrijan errores, es decir que sean efectivos

Siempre supondremos que nuestro canal será:

- De errores aleatorios.
- Simétrico.
- $P_{A,B}$ no depende ni de A ni de B, por lo tanto utilizaremos la siguiente notación $P_{A,B} = P$.

Caso en el que $P < \frac{1}{2}$

Siempre supondremos esta hipótesis: $P < \frac{1}{2}$.

Caso en el que $P > \frac{1}{2}$

En este caso cogemos la palabra que nos ha llegado, la damos la vuelta y ya tenemos que $P < \frac{1}{2}$.

Caso en el que $P = \frac{1}{2}$

En este caso tenemos que si transmitimos un 0 puede llegar, indistintamente, un 0 o un 1, con lo cual el canal considerado no es nada fiable y no es util desde un punto de vista práctico. Luego no consideraremos nunca este caso.

1.6 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[6], 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[7].

1.6.1 Alfabeto

Definición 1.2 (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 $\{0, 1\}$.

1.6.2 Palabras

Definición 1.3 (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 $|\mathbb{F}_q|^n$.

[6] "Bits" de control.
[7] Decodificar el mensaje.

1.6.3 Códigos

Definición 1.4 (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 $\mathcal{C} \subset \mathbb{F}_q^n$.

1.6.4 Canales

El canal es el medio por el cual se transmite la información y sobre este canal supondremos siempre que:

- Cumple la hipótesis de "*canal discreto*"[8].
- Cumple la hipótesis de "*errores aleatorios*"[9].
- El canal es "*simétrico*".

1.7 Ejercicios

Ejercicio 1.1
Dada una palabra cualquiera del código de triple control, comprobar que es lo que falla sí se produce un fallo en cualquiera de los seis bits.

Solución:
Sea 110011 una palabra perteneciente al código de triple control.

Todas las palabras del código son de la forma:

$$\mathcal{C}[6,3] = \{\ (u_1, u_2, u_3, u_1+u_2, u_1+u_3, u_2+u_3)\ u_i \in \mathbb{F}_2\ \}$$

Llamaremos a los bit de control:

$$c_1 = u_1 + u_2 \quad c_2 = u_1 + u_3 \quad c_3 = u_2 + u_3$$

Introduciremos un error en cada bit y veremos que bits de control fallan, suponiendo siempre que los bits de información son correctos.

[8] Apartado 1.5.1 en la página 17.
[9] Apartado 1.5.2 en la página 17.

Bit de error	Palabra erronea	Bits de control	Bits correctos
Primer bit	010011	011	101
Segundo bit	100011	011	110
Tercer bit	111011	011	000
Cuarto bit	110111	111	011
Quinto bit	110001	001	011
Sexto bit	110010	010	011

■

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 Radio de recubrimiento de un código

Definición 2.5 (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 $\mathbb{F}_q^n$.

2.2 Codificadores

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

Definición 2.6 (Codificador)
Sea $\mathcal{C}$ un código del tipo (n, m). Llamaremos ***"codificador"*** *a una aplicación biyectiva C, tal que:*

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

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

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 **estandar**, 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 2.7 (Decodificador)
Sea $\mathcal{C}$ un código del tipo (n, m), Llamaremos ***"decodificador"*** *a una aplicación biyectiva D, tal que:*

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

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 2.8 (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 2.9 (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.

2.5 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 2.10 (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 2.11 (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**".

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

Definición 2.12 (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 2.13 (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 2.14 (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:

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

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 $\iff$ sí ambas palabras son la misma. $d(a,b) = 0 \iff a = b$.

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 2.15 (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].

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

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 27, 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 27, 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.

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 24.

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 2.16 (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 2.17 (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* $s' = s + t \iff d_{min} \geq 2 \cdot t + s + 1 = t + s' + 1$.

2.8 ¿Detectar o corregir?

Si observamos los ejercicios de este capítulo, en la página 44, podemos ver que en los códigos de "*triple repetición*" y de "*triple control*" tenemos una de las dos siguientes opciones:

- Detectar y corregir un error.
- Detectar dos errores y no corregir.

2.8.1 Probabilidad de error en el canal

Para saber cual de las dos opciones utilizar necesitaremos conocer "*algo*" más sobre el canal de transmisión. Ese "*algo*" será la "*probabilidad de errores en el canal*".

Teorema 2.4 (Resultados sobre la probabilidad de error)
Supongamos que tenemos un canal simétrico, aleatorio, binario y con una probabilidad de error P. En dicho canal transmitiremos palabras de longitud n mediante un código de bloques.

1. *La probabilidad de que ocurra un error concreto de peso k será:*

$$P^k \cdot (1-P)^{n-k}$$

2. *La probabilidad de que ocurra un error cualquiera de peso k será:*

$$\binom{n}{k} \cdot P^k \cdot (1-P)^{n-k}$$

Demostración:

1. Si se ha cometido un error especifico de peso k entonces tendremos que k bits son incorrectos y los $n-k$ restantes son correctos.

   La probabildad de tener un bit en particular incorrecto es P, por lo tanto la probabilidad de que un bit en particular sea correcto es $1-P$. Como el canal es aleatorio todas estas probabilidades son independientes.

   De todo esto y de que tenemos k bits erroneos y $n-k$ correctos se tiene que la probabilidad buscada es:

$$P^k \cdot (1-P)^{n-k}$$

2. El número total de palabras con un error de peso k es el mismo que la cantidad de formas de elegir k lugares de error, que es $\binom{n}{k}$. Como los errores son excluyentes, que sí ocurre uno no puede ocurrir otro, tenemos entonces que la probabilidad de que ocurra uno de ellos es la suma de las probabilidades de cada error, es decir:

$$\binom{n}{k} \cdot P^k \cdot (1-P)^{n-k}$$

■

Teorema 2.5 (Probabilidad de transmisiones correctas)
Supondremos que utilizamos un código de bloques para transmitir un mensaje sobre un canal binario y simétrico. Tenemos los siguientes resultados:

1. *La probabilidad de que un procesador de error produzca la palabra correcta es la suma de las probabilidades de error de los errores que el procesador de error puede corregir.*

2. *Si los errores que el procesador de error puede corregir son independientes de la palabra transmitida entonces la probabilidad de que el mensaje se haya recibido correctamente es la suma de las probabilidades de que se den los errores que el procesador corrige.*

Demostración:

1. Los errores que el procesador corrige son excluyentes, ya que si se produce un error no se puede producir otro. Entonces la probabilidad de que uno ocurra es la suma de sus probabilidades individuales.

2. Como los errores que el procesador corrige son independientes de la palabra entonces la probabilidad de transmisión correcta es la suma de las probabilidades individuales de los errores que el procesador de error corrige.

■

2.8.2 Ejemplos sobre probabilidad de error

Supongamos que vamos a transmitir un mensaje de 10000 bits por un canal con una probabilidad de error de $\frac{1}{1000}$. Esto significa que hay una probabildad de error en cada bit de $\frac{1}{1000}$, o lo que es lo mismo, que hay una probabilidad de transmisión correcta en un bit de $1 - \frac{1}{1000} = 0.999$.

Sin utilizar ningún tipo de codificación la probabildad de transmisión correcta es $(0.999)^{10000} \simeq 0.000045$.

Código del bit de control de paridad

Este código detecta hasta un error en cada palabra transmitida de longitud ocho, pero no lo corrige.

- Probabilidad de que no haya errores en una palabra.

  Como una palabra tiene 8 bits la probabilidad de que no haya error en una palabra será:

  $$(0.999)^8 \simeq 0.992028$$

- Probabilidad de que haya un error en una palabra.

  La probabilidad de que ocurra un error en una palabra es la probabilidad de que en una palabra haya siete bits correctos y uno incorrecto:

  $$(0.999)^7 \cdot \frac{8}{1000} \simeq 0.007944$$

- Probabilidad de transmisión correcta.

  Como queremos transmitir 1000 bits con este código los agruparemos en bloques de siete más el de control. Estas serán las palabras, luego la probabilidad de transmisión correcta será la probabilidad de que no haya ningún error en la transmisión, ya que el código no corrige errores:

  $$(0.992028)^{\frac{10000}{7}} \simeq 0.000011$$

- Probabilidad de que no ocurra un error indetectable.

  La probabilidad de que no ocurra un error indetectable es la misma de que ocurra un error detectable, es decir, de que no ocurra ningún error o de que ocurra un error:

  $$((0.992028) + (0.007944))^{\frac{10000}{7}} \simeq 0.960789$$

Código de triple repetición

Podemos ver en el ejercicio 2.4 que en este código se presentan dos situaciones:

1. Supondremos que es más probable que se cometan dos errores que uno en cada palabra transmitida. En este caso no se podrá corregir el error pero se detectarán errores simples y dobles.

   - Probabilidad de que no haya errores en una palabra.

     Como una palabra tiene 3 bits la probabilidad de que no haya error en una palabra:

     $$(0.999)^3 \simeq 0.997003$$

   - Probabilidad de transmisión correcta.

     Como hemos de transmitir 10000 bits y con este código por cada bit de información hemos de transmitir 3 bits, tendremos que vamos a transmitir un total de 30000 bits. Luego la probabilidad de transmisión correcta será:

     $$(0.999)^{30000} \simeq 9.218214^{-14}$$

   - Probabilidad de que no ocurra un error indetectable.

     En el caso que estamos considerando el código detecta que ha ocurrido un error y supondrá que es un error doble, luego el único caso en el que no detectará un error será cuando ocurra un error triple, es decir, que se inviertan todos los bits de una palabra transmitida. Como la probabilidad de error del canal es $\frac{1}{1000}$ entonces tenemos que la probabilidad de que se haya un error triple será:

     $$\left(\frac{1}{1000}\right)^3 = 10^{-9}$$

     Como vamos a transmitir 10000 bloques de tres bits entonces la probabilidad de que no ocurra un error triple, indetectable, será:

     $$(1 - 10^{-9})^{10000} \simeq 0.99999$$

Podemos ver que este código tiene un probabilidad practicamente nula de que ocurran errores indetectables, pero la probabilidad de que ocurran errores, detectables, en cada palabra es altísima. Y teniendo en cuenta que el código no corrige errores necesitaremos una gran cantidad de retransmisiones, que, junto con el hecho de que la razón del código es baja, $\frac{1}{3}$, hace que transmitir de esta manera sea lento y poco recomendable.

2. Supondremos que es más probable que se cometa un error que dos en cada palabra transmitida. En este caso podremos corregir el error.

   - Probabilidad de que no haya errores en una palabra.

     Como una palabra tiene 3 bits la probabilidad de que no haya error en una palabra:

     $$(0.999)^3 \simeq 0.997003$$

   - Probabilidad de que haya un error en una palabra.

     La probabilidad de que ocurra un error en una palabra es la probabilidad de que en una palabra haya dos bits correctos y uno incorrecto:

     $$(0.999)^2 \cdot \frac{3}{1000} \simeq 0.002994$$

   - Probabilidad de transmisión correcta.

     La probabilidad de transmisión correcta es la probabilidad de que no ocurra ningún error o de que ocurra uno[5] en todas y cada una de las palabras transmitidas, que son 10000, entonces:

     $$((0.997003) + (0.002994))^{10000} \simeq 0.970445$$

   - Probabilidad de que no ocurra un error indetectable.

     La probabilidad de que no ocurra un error indetectable es la misma de que ocurra un error detectable, es decir, de que no ocurra ningún error o de que ocurra un error:

     $$((0.997003) + (0.002994))^{10000} \simeq 0.970445$$

[5] El código corrige un error.

Código del triple control

Supondremos que es más probable que se cometa un error que dos en cada palabra transmitida. En este caso podemos corregir el error.

- Probabilidad de que no haya errores en una palabra.

  Como una palabra tiene 6 bits la probabilidad de que no haya errores en una palabra:

  $$(0.999)^6 \simeq 0.994015$$

- Probabilidad de que haya un error en una palabra.

  La probabilidad de que ocurra un error en una palabra es la probabilidad de que haya cinco bits correctos y uno incorrecto:

  $$(0.999)^5 \cdot \frac{6}{1000} \simeq 0.005970$$

- Probabilidad de transmisión correcta.

  La probabilidad de transmisión correcta es la probabilidad de que no ocurra ningún error o de que ocurra uno[6] en todas y cada una de las palabras transmitidas, que son $\frac{10000}{3}$:

  $$((0.994015) + (0.005970))^{\frac{10000}{3}} \simeq 0.951229$$

- Probabilidad de que no ocurra un error indetectable.

  La probabilidad de que no ocurra un error indetectable es la misma de que ocurra un error detectable, es decir, de que no ocurra ningún error o de que ocurra un error:

  $$((0.994015) + (0.005970))^{\frac{10000}{3}} \simeq 0.951229$$

[6] El código corrige un error.

2.9 El teorema de Shannon

Claude Shannon demostró en 1.948 que existe una constante llamada "*capacidad del canal*", $C(P)$, para cualquier canal simétrico, binario y con probabilidad P tal que siempre existen códigos de bloques donde la probabilidad de transmisión correcta está arbitrariamente próxima a $C(P)$.

Teorema 2.6 (de Shannon, 1.948)
Dado un canal simétrico, binario y con probabilidad P siempre existen códigos de bloques donde la probabilidad de transmisión correcta está arbitrariamente próxima a:

$$C(P) = 1 + P \cdot \log_2 P + (1 - P) \cdot \log_2(1 - P)$$

La constante $C(P)$ recibe el nombre de "capacidad del canal"*.*

Supongamos que la probabilidad de error del canal es $P = \frac{1}{2}$ entonces se tiene que $C(\frac{1}{2}) = 0$, es decir, la probabilidad de transmisión correcta en un canal de este tipo es practicamente nula. Con lo cual no es aconsejable trabajar con canales de este tipo.

2.10 Ejemplos

2.10.1 Código del bit de control de paridad

Este código será un código del tipo $(8, 7)$, es decir:

- El código utiliza palabras de 8 bits de longitud.
- El código utiliza 7 bits para transmitir información.

$$\mathcal{C}[8, 7] = \{(a_1, \cdots, a_7, a_8) \quad donde\ a_i \in \mathbb{F}_2\} \subset \mathbb{F}_2^8$$

donde a_8 es tal que el número de 1 en la palabra es par, es decir:

$$a_8 = \sum_{i=1}^{7} a_i$$

donde la suma es la suma en $\mathbb{F}_2$, sí la suma es par entonces cero y en caso contrario uno.

Codificador

El codificador para este código será el siguiente:

$$\begin{array}{rcl} C : \mathbb{F}_2^7 & \xrightarrow{\sim} & \mathcal{C} \\ (a_1, \cdots, a_7) & \longrightarrow & (a_1, \cdots, a_7, c_1 = \sum_{i=1}^{7} a_i) \end{array}$$

Decodificador

El decodificador para este código será el siguiente:

$$\begin{array}{rcl} D : \mathcal{C} & \xrightarrow{\sim} & \mathbb{F}_2^7 \\ (a_1, \cdots, a_7, c_1) & \longrightarrow & (a_1, \cdots, a_7) \end{array}$$

Distancia mínima del código

La distancia mínima para este código es 2.

Para calcular la distancia mínima de este código tendremos que calcular $2^7 \cdot (2^7 - 1) = 16.256$ distancias y quedarnos con la menor.

Razón del código

Este código utiliza 8 bits para transmitir, de los cuales 7 son de información su razón es $\frac{7}{8}$.

2.10.2 Código de triple repetición

Este código será un código del tipo $(3, 1)$, es decir:

- El código utiliza palabras de 3 bits de longitud.
- El código utiliza 1 bit para transmitir información.

$$\mathcal{C}[3, 1] = \{000, 111\} \subset \mathbb{F}_2^3$$

Codificador

El codificador para este código será el siguiente:

$$\begin{array}{rcl} C : \mathbb{F}_2 & \xrightarrow{\sim} & \mathcal{C} \\ 0 & \longrightarrow & 000 \\ 1 & \longrightarrow & 111 \end{array}$$

Decodificador

El decodificador para este código será el siguiente:

$$\begin{aligned} D : \mathcal{C} &\xrightarrow{\sim} \mathbb{F}_2 \\ 000 &\longrightarrow 0 \\ 111 &\longrightarrow 1 \end{aligned}$$

Distancia mínima

La distancia mínima para este código es 3.

Para calcular la distancia mínima de este código tendremos que calcular $2^1 \cdot (2^1 - 1) = 1$ distancia y quedarnos con la menor.

Razón del código

Este código utiliza 3 bits para información, de los cuales 1 es de información su razón es $\frac{1}{3}$.

2.10.3 Código del triple control

Este código será un código del tipo $(6, 3)$, es decir:

- El código utiliza palabras de 6 bits de longitud.
- El código utiliza 3 bits para transmitir información.

$$\mathcal{C}[6,3] = \{(a_1, a_2, a_3, c_1, c_2, c_3) \quad \textit{donde } a_i, c_i \in \mathbb{F}_2\} \subset \mathbb{F}_2^6$$

donde c_1, c_2 y c_3 estan determinados por a_1, a_2 y a_3.

Codificador

El codificador para este código será el siguiente:

$$\begin{aligned} C : \mathbb{F}_2^3 &\xrightarrow{\sim} \mathcal{C} \\ (a_1, a_2, a_3) &\longrightarrow (a_1, a_2, a_3, c_1, c_2, c_3) \end{aligned}$$

Decodificador

El decodificador para este código será el siguiente:

$$\begin{aligned} D : \mathcal{C} &\xrightarrow{\sim} \mathbb{F}_2^3 \\ (a_1, a_2, a_3, c_1, c_2, c_3) &\longrightarrow (a_1, a_2, a_3) \end{aligned}$$

Distancia mínima

La distancia mínima para este código es de 3.

Para calcular la distancia mínima de este código tendremos que calcular $2^3 \cdot (2^3 - 1) = 56$ y quedarnos con la menor.

Razón del código

Este código utiliza 6 bits para transmitir, de los cuales 3 son de información su razón es $\frac{3}{6} = \frac{1}{2}$.

2.11 Ejercicios

Ejercicio 2.1 *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 $d(x, y) = 0 \iff x = y$.

  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:

  $$\#U = d(x,z) = \#S + \#T \tag{2.1}$$

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

  $$d(x,y) \geq \#T \tag{2.2}$$

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

  $$d(y,z) \geq \#S \tag{2.3}$$

  Si sumamos miembro a miembro (2.2) y (2.3) y tenemos en cuenta (2.1) tenemos:

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

  que es lo que queriamos demostrar.

  ■

Ejercicio 2.2 *Comprobar que en el código del* "bit de control de paridad"*:*

- $d_{min} = 2$.
- *Detecta errores simples,* $s' = 1$.
- *No corrige ningún error,* $t = 0$.

Solución:

- **Distancia mínima.**
  - $d_{min} = 0$ no puede darse nunca, por definición de d_{min}.
  - $d_{min} = 1$ este caso tampoco puede darse. Supongamos que tenemos una palabra cualquiera del código, $a = (a_1, \cdots, a_7, c_1)$, y que existiera otra palabra del código, b, tal que $d(a,b) = 1$. Esto implicaría que todos los bits de ambas palabras son iguales excepto uno. Obviamente el último bit no podría ser ya que si a una palabra del código le cambiamos el último bit la palabra resultante no pertenece al código[7]. Luego para que $d(a,b) = 1$ tendríamos que alterar uno de los siete primeros bits, y en ese caso la palabra resultante tampoco pertenecería al código[8]. Luego deberiamos cambiar dos bits para que la palabra resultante perteneciera al código entonces $d(a,b) = 2$. Luego en el código no existen $a, b \in \mathcal{C}$ tales que $d(a,b) = 1$.
  - $d_{min} = 2$ De lo dicho en el apartado anterior se deduce que en el codigo existen palabras $a, b \in \mathcal{C}$ tal que $d(a,b) = 2$. Para ello basta con coger una palabra cualquiera del código, por ejemplo $a = (0,0,1,1,0,0,1,1)$, alterar uno cualquiera de sus siete primeros bits y poner como bit de control el correspondiente para cumplir la condición de paridad. Por ejemplo si cambiamos $a_2 = 0$ por 1 el bit de control pasaría de ser 1 a ser 0, con lo cual $b = (0,1,1,1,0,0,1,0) \Longrightarrow d(a,b) = 2$, y como $a, b \in \mathcal{C}$ se tiene que $d_{min} = 2$.
- **Errores que detecta.**
  Para ello utilizamos el teorema 2.1 de detección de errores. Por este teorema para detectar errores de peso s' se tiene que cumplir que $d_{min} \geq s' + 1$. Pero como tenemos que $d_{min} = 2$ entonces $s' = 1$. Luego este código detecta errores simples.

[7] No se cumple la condición de tener una cantidad par de unos en la palabra.

[8] Ya que tendriamos que cambiar el último bit para cumplir la condición de paridad

- **Errores que corrige.**
  Para ello utilizamos el teorema 2.3 de detección y corrección de errores. Por este teorema para corregir errores de peso t se tiene que cumplir que $d_{min} \geq t + s' + 1$. Pero como tenemos que $d_{min} = 2$ y $s' = 1$ entonces $t = 0$. Luego este código no corrige errores.

■

Ejercicio 2.3 *Comprobar que en el código de* "triple control"*:*

- $d_{min} = 3$.
- *Detecta errores dobles o simples,* $s' = 2$ *o* $s' = 1$.
- *No corrige ningún error o corrige uno,* $t = 0$ *o* $t = 1$.

Solución:
Recordemos que:

$$\mathcal{C} = \{(x, y, z, a, b, c) \in \mathbb{F}_2^6 \quad donde\ a = x + y,\ b = x + z,\ c = y + z\}$$

o lo que es lo mismo:

$$\mathcal{C} = \{\ (x, y, z, x + y, x + z, y + z)\ donde\ x, y, z \in \mathbb{F}_2\ \}$$

- **Distancia mínima.**
  - $d_{min} = 0$ no puede darse nunca, por definición de d_{min}.
  - $d_{min} = 1$ tampoco puede darse debido a que si tenemos dos palabras $a, b \in \mathcal{C}$ tales que $d(a, b) = 1$ entonces ambas palabras coincidirian en todos sus bits menos en uno. El bit donde no coinciden no puede ser ninguno de los tres últimos[9], ya que si coinciden en los tres primeros han de coincidir en los tres últimos para que ambas palabras pertenezcan al código. Luego si se diferencian en un, bit forzosamente ha de ser uno de los tres primeros. Si se diferecian en uno de los tres primeros bits,también se han de diferenciar en dos de los tres últimos[10] luego su distancia no seríauno, sino tres. Por lo tanto en el código no existen palabras cuya distancia entre sí sea uno.

[9] Bits de control.
[10] Para satisfacer las condiciones de paridad de unos.

- $d_{min} = 2$ por el mismo motivo que en el caso anterior si dos palabras del código se diferencian en dos bits, estos bits han de ser dos de los tres primeros. Si dos palabras del código se diferencian en dos de sus tres primeros bits, forzosamente se tienen que diferenciar en dos de los tres últimos y, problablemente, también en el otro bit del final. Con lo cual la distancia entre ambas palabras sería, como poco, de cuatro. Luego en el código no puede haber palabras cuya distancia entre sí sea dos.
- $d_{min} = 3$ cojamos una palabra cualquiera del código, por ejemplo $a = (0, 0, 1, 0, 1, 1)$. Cambiemos uno de sus tres primeros bits y pongamos los tres últimos de tal forma que la palabra obtenida pertenezca al código, por ejemplo $b = (1, 0, 1, 1, 0, 1)$. Entonces tenemos que $d(a, b) = 3$. Luego la $d_{min} = 3$.

- **Errores que detecta.**
  Para ello utilizamos el teorema 2.1 de detección de errores. Por este teorema para detectar errores de peso s' se tiene que cumplir que $d_{min} \geq s' + 1$. Pero como tenemos que $d_{min} = 3$ entonces $s' = 2$ o $s' = 1$. Luego este código detecta errores dobles o simples.

- **Errores que corrige.**
  Para ello utilizamos el teorema 2.3 de detección y corrección de errores. Por este teorema para detectar errores de peso s' y corregir errores de peso t se tiene que cumplir que $d_{min} \geq t + s' + 1$. Pero tenemos dos casos:

  - $s' = 1$, el código detecta errores simples. En este caso tendremos lo siguiente:

    $$3 = d_{min} \geq t + s' + 1 = t + 1 + 1 = t + 2$$

    con lo que la solución es $t = 1$. Luego si el código detecta un error lo corrige.
  - $s' = 2$, el código detecta errores dobles. En este caso tendremos lo siguiente:

    $$3 = d_{min} \geq t + s' + 1 = t + 2 + 1 = t + 3$$

    con lo que la única solución es $t = 0$. Luego si el código detecta dos errores no los corrige.

■

Según lo que hemos visto cuando utilizemos el código de triple repetición tenemos dos posibles opciones:

- Detectar cuando ocurren errores simples o dobles.

  En este caso detectaremos cuando ha ocurrido un error, sin saber si el error es de tipo simple o doble, pero no podremos corregirlo.

- Detectar y corregir errores simples.

  En este caso supondremos que cualquier error que se cometa es simple, corrigiendolo. En el caso de suponer que los errores que se comenten son simples, y corregirlos, cuando se cometa un error doble estaremos suponiendo que el error es simple y al corregirlo estaremos corrigiendo mal.

La forma de utilizar el código depende de las probabilidades de error del canal, si la probabilidad de errores dobles o simples son similares es conveniente utilizar el código para detectar errores. Sin embargo, si predomina una probabilidad sobre la otra utilizaremos el código para corregir, suponiendo que todos los errores que se cometan serán del tipo de error que más probabilidad tenga. Es claro que, cuando se cometa el error que menos probabilidades tiene estaremos corrigiendo mal.

Ejercicio 2.4 *Comprobar que en el código de* "triple repetición"*:*

- $d_{min} = 3$.
- *Detecta errores dobles o simples,* $s' = 2$ *o* $s' = 1$.
- *No corrige ningún error o corrige uno,* $t = 0$ $t = 1$.

Solución:

- **Distancia mínima.**
  Como el código sólo tiene dos palabras la distancia mínima se calcula de forma muy sencilla $d_{min} = d(000, 111) = 3$.

- **Errores que detecta.**
  Para ello utilizamos el teorema 2.1 de detección de errores. Por este teorema para detectar errores de peso s' se tiene que cumplir que $d_{min} \geq s'+1$. Pero como tenemos que $d_{min} = 3$ entonces $s' = 2$ o $s' = 1$. Luego este código detecta errores dobles o simples.

- **Errores que corrige.**
  Para ello utilizamos el teorema 2.3 de detección y corrección de errores. Por este teorema para detectar errores de peso s' y corregir errores de peso t se tiene que cumplir que $d_{min} \geq t + s' + 1$. Pero tenemos dos casos:

  - $s' = 1$, el código detecta errores simples. En este caso tendremos lo siguiente:

    $$3 = d_{min} \geq t + s' + 1 = t + 1 + 1 = t + 2$$

    con lo que la solución es $t = 1$. Luego si el código detecta un error lo corrige.
  - $s' = 2$, el código detecta errores dobles. En este caso tendremos lo siguiente:

    $$3 = d_{min} \geq t + s' + 1 = t + 2 + 1 = t + 3$$

    con lo que la única solución es $t = 0$. Luego si el código detecta dos errores no los corrige.

■

Según lo que hemos visto cuando utilizemos el código de triple control tenemos dos posibles opciones:

- Detectar cuando ocurren errores simples o dobles.

  Detectamos cuando ocurren errores simples o dobles, pero no podemos corregir.

- Detectar y corregir errores simples.

  Detectamos errores simples y los corregimos.

La forma de utilizar el código depende de las probabilidades de error que se den en el canal que estemos utilizando.

Ejercicio 2.5 *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.

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 23, 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 58.
[3] Apartado (3.5.2), en la página 58.

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 $(n-m) \times (n-m)$.*

3.6.3 Rango de la matriz de control

Proposición 3.4

1. *El rango de la matriz de control H de un código lineal $\mathcal{C}[n,m]$ es $n-m$.*
2. *En particular H tiene al menos $n-m$ filas.*
3. *Sí H es una matriz de control para el código lineal $\mathcal{C}[n,m]$ y K es una matriz que se obtiene añadiendo filas que son combinaciones lineales de las filas de H, entonces K también es una matriz de control para el código lineal $\mathcal{C}[n,m]$.*

Demostración:

1. Tenemos que $\dim_{\mathbb{F}_q} \mathcal{C}[n,m] = m$, $\mathcal{C}[n,m] \subset \mathbb{F}_q^n$ y que $\dim_{\mathbb{F}_q} \mathbb{F}_q^n = n$. Por algebra lineal sabemos que $\dim_{\mathbb{F}_q} \mathcal{C}[n,m]^\circ = n-m$. Por construcción las filas de la matriz H son una base del incidente al código, $\mathcal{C}[n,m]^\circ$, luego el rango de H es $n-m$.
2. Es inmediato a partir de la construcción de la matriz H.
3. Sea $u \in \mathcal{C}[n,m]$ entonces $H \cdot u^t = 0$. Construyamos una matriz K tal que sea la matriz H pero añadiendo filas que son combinación lineal de filas de la matriz H.

   Las primeras filas de H y K son iguales. Como H tiene, al menos, $n-m$ filas supongamos que H tiene $n-m$ filas.

   Por construcción de K tendremos que las $n-m$ primeras filas de $K \cdot u^t$ son cero, y el resto de filas también son cero ya que serán combinaciones lineales de las primeras $n-m$ filas, que son cero. Esto es debido a que las filas situadas después de las $n-m$ primeras en la matriz K son combinación lineal de las filas de H y que H es la matriz de control del código.

■

3.7 Ejemplos

3.7.1 Código del bit de control de paridad

Es gracias a la estructura algebraica que tiene $(\mathbb{F}_2, +, \cdot)$, cuerpo, podemos expresar este código como:

$$\mathcal{C} = \{ \ (x_1, \cdots, x_7, \sum_{i=1}^{7} x_i) \quad \{x_i\}_{i=1}^{7} \in \mathbb{F}_2 \ \}$$

El código del bit de control de paridad es lineal

Para ser un código lineal $\mathcal{C}$ ha de ser un subespacio vectorial de $\mathbb{F}_2^8$.

Para ver que es un subespacio vectorial veamos que:

- La suma de dos palabras del código está en el código.

  Consideremos dos palabras cualesquiera del código:

  $$(x_1, \cdots, x_7, \sum_{i=1}^{7} x_i)$$

  con $\{x_i\}_{i=1}^{7} \in \mathbb{F}_2$, y

  $$(x'_1, \cdots, x'_7, \sum_{i=1}^{7} x'_i)$$

  con $\{x'_i\}_{i=1}^{7} \in \mathbb{F}_2$:

  $$(x_1, \cdots, x_7, \sum_{i=1}^{7} x_i) + (x'_1, \cdots, x'_7, \sum_{i=1}^{7} x'_i) =$$

  $$= (x_1 + x'_1, \cdots, x_7 + x'_7, \sum_{i=1}^{7} (x_i + x'_i) \ ) \in \mathcal{C}$$

- El producto de un escalar por una palabra del código es otra palabra del código.

  Consideremos una palabra cualquiera del código:

$$(x_1, \cdots, x_7, \sum_{i=1}^{7} x_i)$$

  con $\{x_i\}_{i=1}^{7} \in \mathbb{F}_2$, y un elemento cualquiera del cuerpo, $x \in \mathbb{F}_2$:

$$x \odot (x_1, \cdots, x_7, \sum_{i=1}^{7} x_i) = (x \cdot x_1, \cdots, x \cdot x_7, \sum_{i=1}^{7} (x \cdot x_i)\ ) \in \mathcal{C}$$

Luego $\mathcal{C}$ es un subespacio vectorial de $\mathbb{F}_2^8$. Es un código lineal.

Una base de este subespacio vectorial es:

$$\begin{aligned}
e_1 &= (1,0,0,0,0,0,0,1) \\
e_2 &= (0,1,0,0,0,0,0,1) \\
e_3 &= (0,0,1,0,0,0,0,1) \\
e_4 &= (0,0,0,1,0,0,0,1) \\
e_5 &= (0,0,0,0,1,0,0,1) \\
e_6 &= (0,0,0,0,0,1,0,1) \\
e_7 &= (0,0,0,0,0,0,1,1)
\end{aligned}$$

Luego $\mathcal{C}$ es un subespacio vectorial tal que $\dim_{\mathbb{F}_2} \mathcal{C} = 7$.

Codificador

El codificador de este código será una aplicación lineal:

$$\begin{aligned}
C : \mathbb{F}_2^7 &\xrightarrow{\sim} \mathcal{C} \subset \mathbb{F}_2^8 \\
x &\longrightarrow (x_1, \ldots, x_7, \sum_{i=1}^{7} x_i)
\end{aligned}$$

Las columnas de la matriz que define esta aplicación lineal, codificador, serán $C(e_i)$, donde $i = 1, \ldots, 7$ y los e_i son vectores de una base del subespacio $\mathcal{C}$.

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

Figura 3.1: Matriz generadora, en forma estándar, del código del bit de control de paridad.

Para codificar una palabra:

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

La matriz generadora de este código se puede expresar en otras bases como:

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

la cual no está en forma estándar.

Ecuaciones paramétricas

Para calcular las ecuaciones paramétricas utilizaremos la matriz generadora en forma estándar.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \end{pmatrix} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_1 + \ldots + \lambda_7 \end{pmatrix}$$

Las ecuaciones paramétricas son:

$$\begin{cases} x_1 &= \lambda_1 \\ x_2 &= \lambda_2 \\ x_3 &= \lambda_3 \\ x_4 &= \lambda_4 \\ x_5 &= \lambda_5 \\ x_6 &= \lambda_6 \\ x_7 &= \lambda_7 \\ x_8 &= \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 \end{cases}$$

Figura 3.2: Ecuaciones paramétricas del código del bit de control de paridad.

Ecuaciones implícitas

Como $\dim_{\mathbb{F}_2} \mathbb{F}_2^8 = 8$ y $\dim_{\mathbb{F}_2} \mathcal{C} = 7$ tendremos $8 - 7 = 1$ ecuación implícita.

Planteemos el sistema de ecuaciones:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

De donde nos sale que $\mathcal{C}^\circ =< (1,1,1,1,1,1,1,1) >$. Luego la ecuación implícita del código será:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 0$$

Figura 3.3: Ecuación implícita del código del bit de control de paridad.

Matriz de control

La matriz de control utilizando las ecuaciones implícitas de este código es:

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

Figura 3.4: Matriz de control, en forma estándar, para el código del bit de control de paridad.

3.7.2 Código de triple repetición

Es gracias a la estructura algebraica que tiene $(\mathbb{F}_2, +, \cdot)$, cuerpo, podemos expresar este código como:

$$\mathcal{C} = \{ \ (x,x,x) \quad x \in \mathbb{F}_2 \ \}$$

El código de triple repetición es lineal

Para ser un código lineal $\mathcal{C}$ ha de ser un subespacio vectorial de $\mathbb{F}_2^3$.

Para ver que es un subespacio vectorial veamos que:

- La suma de dos palabras del código está en el código.

  Como el código sólo tiene dos palabras veamos que su suma es otra palabra del código.

  $$(0,0,0) + (1,1,1) = (1,1,1) \in \mathcal{C}$$

- El producto de un escalar por una palabra del código es otra palabra del código.

  Como el código unicamente tiene dos palabras veamoslo para cada una de ellas:

  Para $(0,0,0)$:

$$x \odot (0,0,0) = \begin{cases} (0,0,0) \in \mathcal{C} & para\ x = 0 \\ \\ (0,0,0) \in \mathcal{C} & para\ x = 1 \end{cases}$$

  Para $(1,1,1)$:

$$x \odot (1,1,1) = \begin{cases} (0,0,0) \in \mathcal{C} & para\ x = 0 \\ \\ (1,1,1) \in \mathcal{C} & para\ x = 1 \end{cases}$$

Luego $\mathcal{C}$ es un subespacio vectorial de $\mathbb{F}_2^3$. Es un código lineal.

Una base de este subespacio vectorial es:

$$e_1 \quad = \quad (1,1,1)$$

Luego $\mathcal{C}$ es un subespacio vectorial tal que $\dim_{\mathbb{F}_2} \mathcal{C} = 1$.

Codificador

El codificador de este código será una aplicación lineal:

$$\begin{array}{rcl} C : \mathbb{F}_2 & \stackrel{\sim}{\longrightarrow} & \mathcal{C} \subset \mathbb{F}_2^3 \\ x & \longrightarrow & (x,x,x) \end{array}$$

La columna de la matriz que define esta aplicación lineal, codificador, será $C(e_1)$, donde e_1 es un vector de la base del subespacio $\mathcal{C}$.

Para codificar cualquiera de las dos palabras:

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad o \quad \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

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

Figura 3.5: Matriz generadora, en forma estándar, del código de triple repetición.

Ecuaciones paramétricas

Para calcular las ecuaciones paramétricas utilizaremos la matriz generadora en forma estándar.

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \end{pmatrix} = \begin{pmatrix} \lambda_1 \\ \lambda_1 \\ \lambda_1 \end{pmatrix}$$

Las ecuaciones paramétricas son:

$$\begin{cases} x_1 = \lambda_1 \\ x_2 = \lambda_1 \\ x_3 = \lambda_1 \end{cases}$$

Figura 3.6: Ecuaciones paramétricas del código de triple repetición.

Ecuaciones implícitas

Como $\dim_{\mathbb{F}_2} \mathbb{F}_2^3 = 3$ y $\dim_{\mathbb{F}_2} \mathcal{C} = 1$ tendremos $3 - 1 = 2$ ecuaciones implícitas.

Planteemos el sistema de ecuaciones:

$$\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}$$

De donde nos sale que $\mathcal{C}^\circ =< (1,1,0), (1,0,1) >$. Luego las ecuaciones implícitas del código serán:

$$\begin{aligned} x + y &= 0 \\ x + z &= 0 \end{aligned}$$

Figura 3.7: Ecuaciones implícitas del código de triple repetición.

Matriz de control

La matriz de control utilizando las ecuaciones implícitas de este código es:

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

Figura 3.8: Matriz de control, en forma estándar, del código de triple repetición.

3.7.3 Código de triple control

Es gracias a la estructura algebraica que tiene $(\mathbb{F}_2, +, \cdot)$, cuerpo, podemos expresar este código como:

$$\mathcal{C} = \{\ (x, y, z, x + y, x + z, y + z) \quad x, y, z \in \mathbb{F}_2\ \}$$

El código de triple control es lineal

Para ser un código lineal $\mathcal{C}$ ha de ser un subespacio vectorial de $\mathbb{F}_2^6$.

Para ver que es un subespacio vectorial veamos que:

- La suma de dos palabras del código está en el código.

  Consideremos dos palabras cualesquiera del código:

  $$(x, y, z, x + y, x + z, y + z)$$

  con $x, y, z \in \mathbb{F}_2$, y

  $$(x', y', z', x' + y', x' + z', y' + z')$$

con $x', y', z' \in \mathbb{F}_2$:

$$(x, y, z, x+y, x+z, y+z) + (x', y', z', x'+y', x'+z', y'+z') =$$
$$= (x+x', y+y', z+z', x+x'+y+y', x+x'+z+z', y+y'+z+z') \in \mathcal{C}$$

- El producto de un escalar por una palabra del código es otra palabra del código.

  Consideremos una palabra cualquiera del código:

  $$(x, y, z, x+y, x+z, y+z)$$

  con $x, y, z \in \mathbb{F}_2$, y un elemento cualquiera del cuerpo, $w \in \mathbb{F}_2$:

  $$w \odot (x, y, z, x+y, x+z, y+z) = (w \cdot x, w \cdot y, w \cdot z, w \cdot x + w \cdot y, w \cdot x + w \cdot z, w \cdot y + w \cdot z) \in \mathcal{C}$$

Luego $\mathcal{C}$ es un subespacio vectorial de $\mathbb{F}_2^6$. Es un código lineal.

Una base de este espacio vectorial es:

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

Luego $\mathcal{C}$ es un subespacio vectorial tal que $\dim_{\mathbb{F}_2} \mathcal{C} = 3$.

Codificador

El codificador de este código será una aplicación lineal:

$$\begin{aligned} C : \mathbb{F}_2^3 &\xrightarrow{\sim} \mathcal{C} \subset \mathbb{F}_2^6 \\ (x, y, z) &\longrightarrow (x, y, z, x+y, x+z, y+z) \end{aligned}$$

Las columnas de la matriz que define esta aplicación lineal, codificador, serán $C(e_i)$, donde $i = 1, 2, 3$ y los e_i son vectores de una base del subespacio $\mathcal{C}$.

Para codificar una palabra:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_1 + x_2 \\ x_1 + x_3 \\ x_2 + x_3 \end{pmatrix}$$

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

Figura 3.9: Matriz generadora, en forma estándar, del código de triple control.

La matriz generadora de este código se puede expresar en otras bases como:

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

la cual no está en forma estándar.

Ecuaciones paramétricas

Para calcular las ecuaciones paramétricas utilizaremos la matriz generadora en forma estándar.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_1 + \lambda_2 \\ \lambda_1 + \lambda_3 \\ \lambda_2 + \lambda_3 \end{pmatrix}$$

$$\begin{cases} x_1 &= \lambda_1 \\ x_2 &= \lambda_2 \\ x_3 &= \lambda_3 \\ x_4 &= \lambda_1 + \lambda_2 \\ x_5 &= \lambda_1 + \lambda_3 \\ x_6 &= \lambda_2 + \lambda_3 \end{cases}$$

Figura 3.10: Ecuaciones paramétricas del código de triple control.

Ecuaciones implícitas

Como $\dim_{\mathbb{F}_2} \mathbb{F}_2^6 = 6$ y $\dim_{\mathbb{F}_2} \mathcal{C} = 3$ tendremos $6-3 = 3$ ecuaciones implícitas.

Planteemos el sistema de ecuaciones:

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

De donde nos sale que $\mathcal{C}^\circ = < (1,1,0,1,0,0), (1,0,1,0,1,0), (0,1,1,0,0,1) >$.

$$\begin{aligned} x_1 + x_2 + x_4 &= 0 \\ x_1 + x_3 + x_5 &= 0 \\ x_2 + x_3 + x_6 &= 0 \end{aligned}$$

Figura 3.11: Ecuaciones implícitas del código de triple control.

Matriz de control

La matriz de control utilizando las ecuaciones implícitas del código es:

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

Figura 3.12: Matriz de control, en forma estándar, del código de triple control.

3.8 Ejercicios

Ejercicio 3.1
Demostrar que pasar de la matriz generadora a la matriz de control equivale a pasar de las ecuaciones paramétricas a las implícitas.

Solución:
Las ecuaciones paramétricas se obtienen a partir de la matriz generadora, mientras que las ecuaciones implícitas se obtienen a partir de la matriz de control.

Ambas ecuaciones nos dan la estructura del código y se pueden utilizar tanto para construir las palabras del código como para verificar que una palabra dada pertenece al código.

Debido a la definición de dichas ecuaciones el paso de la matriz generadora a la de control equivale al paso de ecuaciones paramétricas a implícitas, ya que las ecuaciones paramétricas se obtienen de la matriz generadora y las implícitas de la matriz de control.

■

Capítulo 4

Procesamiento de error en códigos lineales

En este capítulo supondremos que estamos utilizando un código lineal $\mathcal{C}[n,m]$, es decir, $\mathcal{C}[n,m] \subset \mathbb{F}_q^n$.

Ya vimos en el apartado (2.4) lo que era un procesador de error para códigos de bloques.

En los códigos lineales nos es posible sumar palabras y multiplicarlas por constantes, de modo que utilizaremos estas propiedades para simplificar el procesamiento de errores.

Utilizando la linealidad de estos códigos construiremos una tabla con todas las palabras de $\mathbb{F}_q^n$ situadas por proximidad con las palabras del código $\mathcal{C}[n,m]$. La proximidad vendrá dada por la distancia de Hamming.

4.1 Construcción de la tabla de un código lineal

En la tabla estarán los q^n elementos de $\mathbb{F}_q^n$. La tabla la organizaremos por filas y columnas.

Como $\mathcal{C}$ tiene q^m elementos, es de dimensión m, organizaremos la tabla con q^{n-m} filas y q^m columnas. De esta forma en la tabla tendremos:

$$q^{n-m} \cdot q^m = q^{n-m+m} = q^n$$

elementos.

La justificación de elegir esta estructura para la tabla es que de esta manera pondremos como primera fila a todos los elementos del código.

Para referirnos individualmente a cada elemento de la tabla lo haremos utilizando la siguiente notación $Tb(\mathcal{C}[n,m])_{i,j}$ donde $i = 0, \ldots, q^{n-m} - 1$ nos indica la fila y $j = 0, \ldots, q^m - 1$ nos indica la columna.

Definición 4.1 (Fila 0 de la tabla)
La fila 0 de la tabla estará formada por todos los elementos del código $\mathcal{C}[n,m]$ en cualquier orden, pero con la condición de que el primer elemento sea el 0.

Definición 4.2 (Fila i de la tabla)
Supondremos que ya hemos construido todas las columnas hasta la $i-1$ inclusive. Entonces la construcción de la fila i la haremos en dos pasos:

1. *Eligiremos una palabra de $\mathbb{F}_q^n$ que no este en ninguna de las filas anteriores y la situaremos en $Tb(\mathcal{C}[n,m])_{i,0}$.*

2. *Para calcular el resto de los elementos de la fila lo haremos de la siguiente forma:*

$$Tb(\mathcal{C}[n,m])_{i,j} = Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,j} \qquad j = 1, \ldots, q^m - 1$$

Definición 4.3 (Tabla estándar de un código)
*Llamaremos **"tabla estándar"** de un código a una tabla construida de la manera anterior.*

4.2 Estructura de una tabla estándar

Ya hemos visto como se construye la tabla de un código, pero la construcción que hemos hecho no nos asegura que en la tabla esten todos los elementos de $\mathbb{F}_q^n$.

Teorema 4.1
En una tabla estándar para un código $\mathcal{C}[n,m]$ aparecen todas las palabras de $\mathbb{F}_q^n$ una sola vez.

Demostración:
Dados dos elementos cualesquiera de la tabla:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} &= Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,k} \\ Tb(\mathcal{C}[n,m])_{j,l} &= Tb(\mathcal{C}[n,m])_{j,0} + Tb(\mathcal{C}[n.m])_{0,l} \end{aligned}$$

Sí ambos elementos son iguales entonces su diferencia es 0, y como $\mathcal{C}[n,m]$ es lineal se tiene que $0 \in \mathcal{C}[n,m]$ entonces, y debido también a la linealidad del código, ambos elementos deben pertenecer al código, con lo cual $i = j$. Debido a la construcción de la tabla tendremos que $i = j = 0$ ya que los elementos del código están en la primera fila. Teniendo en cuenta esto tendremos que:

$$Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{j,l} = Tb(\mathcal{C}[n,m])_{0,k} - Tb(\mathcal{C}[n,m])_{0,l}$$

Tenemos dos posibilidades:

- $k = l$ entonces como $i = j$ tendremos que ambos elementos son el mismo. Es decir ambos elementos ocupan el mismo lugar en la tabla, son el mismo. Luego no aparece repetido en la tabla.
- $k \neq l$ no se puede dar ya que estamos en un código lineal y si la diferencia de dos palabras del código es cero entonces ambas palabras han de ser la misma, con lo cual $k = l$.

El número de elementos de $\mathbb{F}_q^n$ es q^n.

En la tabla tenemos q^{n-m} filas y q^m columnas entonces el número de elementos de la tabla es $q^{n-m} \cdot q^m = q^{n-m+m} = q^n$. Es decir en la tabla hay tantos elementos como en $\mathbb{F}_q^n$ y como en la tabla no se repiten elementos entonces tenemos que están todos los elementos.

■

Lema 4.1 (Diferencias horizontales)
Dado un código lineal $\mathcal{C}[n,m]$, con tabla $Tb(\mathcal{C}[n,m])$, dos elementos de la misma fila difieren en un elemento del código.

Demostración:
Dos elementos cualesquiera de la fila i-ésima serán de la forma:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} &= Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,k} \\ Tb(\mathcal{C}[n,m])_{i,l} &= Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,l} \end{aligned}$$

La diferencia entre ambos elementos es:

$$Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{i,l} = Tb(\mathcal{C}[n,m])_{0,k} - Tb(\mathcal{C}[n,m])_{0,l}$$

Por construcción los elementos de la fila cero son elementos del código, y, como este es lineal la diferencia de dos elementos del código es otro elemento del código.

■

Lema 4.2 (Diferencias verticales)
Dado un código lineal $\mathcal{C}[n,m]$, con tabla $Tb(\mathcal{C}[n,m])$, dos elementos distintos de la misma columna difieren en una palabra que no pertenece al código.

Demostración:
Dos elementos cualesquiera, $i \neq j$, de la columna k-ésima serán de la forma:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} &= Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,k} \\ Tb(\mathcal{C}[n,m])_{j,k} &= Tb(\mathcal{C}[n,m])_{j,0} + Tb(\mathcal{C}[n,m])_{0,k} \end{aligned}$$

La diferencia entre ambos elementos es:

$$Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{j,k} = Tb(\mathcal{C}[n,m])_{i,0} - Tb(\mathcal{C}[n,m])_{j,0}$$

Los elementos del código están en la fila cero y $Tb(\mathcal{C}[n,m])_{i,0}$, $Tb(\mathcal{C}[n,m])_{j,0}$ no pueden, ambos, estar en la fila cero. Luego al menos uno de ellos no pertenecerá al código, y al ser un código lineal cualquier operación que realizemos entre dos palabras, con una que no pertenezca al código, será otra palabra que no estará en el código.

■

Proposición 4.1 (Propiedades de la tabla estándar)
Sea $\mathcal{C}[n,m]$ un código lineal, entonces se tiene que:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{i,l} &= Tb(\mathcal{C}[n,m])_{j,k} - Tb(\mathcal{C}[n,m])_{j,l} \quad \forall\, i,j,k,l \\ Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{j,k} &= Tb(\mathcal{C}[n,m])_{i,l} - Tb(\mathcal{C}[n,m])_{j,l} \quad \forall\, i,j,k,l \end{aligned}$$

donde $Tb(\mathcal{C}[n,m])$ es la tabla estándar del código.

Demostración:
Por construcción de la tabla tenemos:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} &= Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,k} \\ Tb(\mathcal{C}[n,m])_{i,l} &= Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,l} \\ Tb(\mathcal{C}[n,m])_{j,k} &= Tb(\mathcal{C}[n,m])_{j,0} + Tb(\mathcal{C}[n,m])_{0,k} \\ Tb(\mathcal{C}[n,m])_{j,l} &= Tb(\mathcal{C}[n,m])_{j,0} + Tb(\mathcal{C}[n,m])_{0,l} \end{aligned}$$

Basta con calcular las diferencias:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{i,l} &= Tb(\mathcal{C}[n,m])_{0,k} - Tb(\mathcal{C}[n,m])_{0,l} \\ Tb(\mathcal{C}[n,m])_{j,k} - Tb(\mathcal{C}[n,m])_{j,l} &= Tb(\mathcal{C}[n,m])_{0,k} - Tb(\mathcal{C}[n,m])_{0,l} \end{aligned}$$

Ambas diferencias son iguales. Y:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{j,k} &= Tb(\mathcal{C}[n,m])_{i,0} - Tb(\mathcal{C}[n,m])_{j,0} \\ Tb(\mathcal{C}[n,m])_{i,l} - Tb(\mathcal{C}[n,m])_{j,l} &= Tb(\mathcal{C}[n,m])_{i,0} - Tb(\mathcal{C}[n,m])_{j,0} \end{aligned}$$

Ambas diferencias son iguales.

■

Ya hemos visto como se construye la tabla estándar de un código lineal $\mathcal{C}[n,m]$, y que en dicha tabla estan todos los elementos de $\mathbb{F}_q^n$.

Además de esto podemos ver la relación que guardan entre sí los elementos de cada fila.

Teorema 4.2
Sea $\mathcal{C}[n,m]$ un código lineal y $Tb(\mathcal{C}[n,m])$ su tabla en forma estándar. Sean $Tb(\mathcal{C}[n,m])_{i,k}$ y $Tb(\mathcal{C}[n,m])_{j,l}$ dos elementos de la tabla, entonces están en la misma fila sí y sólo sí su diferencia es una palabra del código.

Demostración:
$\Rightarrow$ | Es el lema 4.1 en la página 77.

$\Leftarrow$ | Sean dos elementos cualesquiera de $Tb(\mathcal{C}[n,m])$:

$$\begin{aligned} Tb(\mathcal{C}[n,m])_{i,k} &= Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,k} \\ Tb(\mathcal{C}[n,m])_{j,l} &= Tb(\mathcal{C}[n,m])_{j,0} + Tb(\mathcal{C}[n,m])_{0,l} \end{aligned}$$

tal que su diferencia sea una palabra del código.

Su diferencia será:

$$Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{j,l} = x + y \tag{4.1}$$

donde:

$$x = Tb(\mathcal{C}[n,m])_{0,k} - Tb(\mathcal{C}[n,m])_{0,l} \quad e \quad y = Tb(\mathcal{C}[n,m])_{i,0} - Tb(\mathcal{C}[n,m])_{j,0}$$

x es una palabra del código ya que es la diferencia de dos palabras del código. Es la diferencia de dos palabras de la fila cero.

Despejando y en (4.1) tenemos:

$$y = (Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{j,l}) - x$$

luego como $\mathcal{C}[n,m]$ es un código lineal e y es diferencia de dos palabras del código se tiene que $y \in \mathcal{C}[n,m]$.

Supongamos ahora que $i \neq j$ entonces como $Tb(\mathcal{C}[n,m])_{i,0}$ y $Tb(\mathcal{C}[n,m])_{j,0}$ pertenecen a la misma columna y distinta fila su diferencia no pertenecerá al código según el lema 4.2. Luego $y \notin \mathcal{C}[n,m]$ por ser la diferencia de una palabra del código y otra que no lo es, llegamos a contradicción ya que y sí que pertenece al código. Entoces $i = j$ luego ambas palabras están en la misma fila.

■

La interpretación matemática de este teorema es:

Corolario 4.1
Sea un código lineal $\mathcal{C}[n,m] \subset \mathbb{F}_q^n$. El $\mathbb{F}_q$-espacio vectorial $\mathbb{F}_q^n/\mathcal{C}[n,m]$ tiene tantas clases de equivalencia como filas tiene su tabla estándar, $Tb(\mathcal{C}[n,m])$. Cada clase de equivalencia estará formada por los elementos de una fila.

Demostración:
Es inmediata a partir del teorema 4.2 y del hecho de que dos elementos de la misma clase de equivalencia módulo $\mathcal{C}[n,m]$ difieren en un elemento de $\mathcal{C}[n,m]$.

■

Observacion 4.2.1

- *Dos filas tienen un elemento en común sí y sólo sí son la misma fila. Se deduce del hecho de que las clases de equivalencia son disjuntas entre sí y que cada fila forma una clase de equivalencia.*

A partir de ahora en la tabla de un código estándar eligiremos como primer elemento de cada fila el elemento que menor peso tenga de la fila.

4.3 Procesamiento de errores

Un error es la diferencia entre la palabra recibida y la transmitida. En los códigos lineales, gracias a su estructrura algebraica, ese error lo podemos expresar como la diferencia entre dos palabras. Si se transmitió la palabra u y se recibio la palabra v el error cometido en la transmisión del mensaje lo podemos expresar como $e = v - u$.

Definición 4.4 (Error cometido para códigos lineales)
Sea $\mathcal{C}[n,m]$ un código lineal. Sí se transmitió la palabra u y se recibió la palabra v diremos que el error que ocurrió en la transmisión es $e = v - u$.

Según esta definición tendremos:

- La palabra recibida será: $v = u + e$.
- La palabra transmitida será: $u = v - e$.

Proposición 4.2
Sea $\mathcal{C}[n,m]$ un código lineal y sea P un procesador de error para dicho código. Sí se recibe una palabra w entonces P corregirá w como $u = w - e$, donde e es el primer elemento de la fila en la que se encuentra w.

Desmostración:
$w = Tb(\mathcal{C}[n,m])_{i,k}$ entonces $e = Tb(\mathcal{C}[n,m])_{i,0}$.

Por la proposición 4.1, en la página 78, tendremos que:

$$Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{0,k} = Tb(\mathcal{C}[n,m])_{i,0} - Tb(\mathcal{C}[n,m])_{0,0}$$

donde $Tb(\mathcal{C}[n,m])_{0,k} = u$ y por contrucción de la tabla tenemos que:

$$Tb(\mathcal{C}[n,m])_{0,0} = (0, \overset{n)}{\ldots}, 0)$$

De donde se deduce que $w - u = e - 0$ entoces $u = w - e$.

■

Según esta proposición un procesador de error corrige errores en función del primer elemento de cada fila, luego los errores que corrige un procesador de errores son aquellos que están en primer lugar en cada fila.

Ejemplo 4.1
Sea $\mathcal{C}[m,m]$ un código lineal binario con tabla $Tb(\mathcal{C}[n,m])$:

- $Tb(\mathcal{C}[n,m])_{1,0} = 100000$ *nos indica que con esta tabla se pueden corregir errores de peso uno en los que el error esté en el primer bit.*
- $Tb(\mathcal{C}[n,m])_{2,0} = 010000$ *nos indica que con esta tabla se pueden corregir errores de peso uno en los que el error esté en el segundo bit.*
- $Tb(\mathcal{C}[n,m])_{3,0} = 001000$ *nos indica que con esta tabla se pueden corregir errores de peso uno en los que el error esté en el tercer bit.*
- $Tb(\mathcal{C}[n,m])_{4,0} = 000100$ *nos indica que con esta tabla se pueden corregir errores de peso uno en los que el error esté en el cuarto bit.*
- $Tb(\mathcal{C}[n,m])_{5,0} = 000010$ *nos indica que con esta tabla se pueden corregir errores de peso uno en los que el error esté en el quinto bit.*
- $Tb(\mathcal{C}[n,m])_{6,0} = 000001$ *nos indica que con esta tabla se pueden corregir errores de peso uno en los que el error esté en el último bit.*
- $Tb(\mathcal{C}[n,m])_{7,0} = 100001$ *nos indica que con esta tabla se pueden corregir errores de peso dos en los que el error esté en el primer y último bit.*

■

Proposición 4.3
Sea $\mathcal{C}[n,m] \subset \mathbb{F}_q^n$ un código lineal con una tabla estándar $Tb(\mathcal{C}[n,m])$ en la que el primer elemento de cada fila es el de menor peso de dicha fila. Sea $Tb(\mathcal{C}[n,m])_{i,k}$ una palabra cualquiera de la tabla entonces se tiene que:

$$d(Tb(\mathcal{C}[n,m])_{i,k}, Tb(\mathcal{C}[n,m])_{0,k}) \leq d(Tb(\mathcal{C}[n,m])_{i,k}, Tb(\mathcal{C}[n,m])_{0,j})$$

donde $j = 0, \ldots, q^m - 1$.

Demostración:
Tenemos por definición de distancia que:

$$d(Tb(\mathcal{C}[n,m])_{i,k}, Tb(\mathcal{C}[n,m])_{0,j}) = w(Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m]_{0,j}))$$

Según varia j tenemos que $T(\mathcal{C}[n,m])_{0,j}$ varia en $\mathcal{C}[n,m]$, luego por las propiedades de la tabla tendremos que $Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{0,j}$ varia en el conjunto formado por los elementos de la fila i. Luego el problema de encontrar el mínimo de:

$$w(Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{0,j})$$

equivale a encontrar la palabra de mínimo peso de la fila i, pero por construcción dicha palabra es $Tb(\mathcal{C}[n,m])_{i,0}$.

$$\min_{j=0,\ldots,q^m-1}\{\ w(Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{0,j})\ \} = Tb(\mathcal{C}[n,m]_{i,0})$$

Luego:

$$d(Tb(\mathcal{C}[n,m])_{i,k}, Tb(\mathcal{C}[n,m])_{0,j}) \geq w(Tb(\mathcal{C}[n,m])_{i,0}) \quad j = 0,\ldots,q^m-1$$

Pero como $Tb(\mathcal{C}[n,m])_{i,k} = Tb(\mathcal{C}[n,m])_{i,0} + Tb(\mathcal{C}[n,m])_{0,k}$, o lo que es lo mismo $Tb(\mathcal{C}[n,m])_{i,0} = Tb(\mathcal{C}[n,m])_{i,k} - Tb(\mathcal{C}[n,m])_{0,k}$ tenemos entonces:

$$d(Tb(\mathcal{C}[n,m])_{i,k}, Tb(\mathcal{C}[n,m])_{0,j}) \geq d(Tb(\mathcal{C}[n,m])_{i,k}, Tb(\mathcal{C}[n,m]_{0,k}))$$

para $j = 0,\ldots,q^m-1$.

■

4.3.1 Corrección de errores

Según la proposición 4.2 para corregir un error tendremos que realizar los siguientes pasos:

- Localizar la palabra recibida en una tabla estándar.
- Suponiendo que la palabra recibida es $Tb(\mathcal{C}[6,3])_{i,k}$ entonces tomaremos como palabra transmitida la palabra que esté en la misma columna pero en la fila cero, $Tb(\mathcal{C}[6,3])_{0,k}$. Siendo el error cometido el primer elemento de la fila en la que se encuentre la palabra recibida, $Tb(\mathcal{C}[6,3])_{i,0}$.
- El número de errores que corrige viene dado por el número de filas que posee la tabla del código, y los patrones de error que corrige son los primeros elementos de cada fila.
- La proposición 4.3 nos indica que corregiremos siempre por la palabra más cercana del código, según la distancia de Hamming.

4.4 Errores que se corrigen adecuadamente

Sí tenemos un código lineal $\mathcal{C}[n,m] \subset \mathbb{F}_q^n$ con una tabla estándar $Tb(\mathcal{C}[n,m])$, en la cual hay q^{n-m} filas y q^m columnas, ya hemos visto como corregir errores. Los errores los corregiremos sumando el primer elemento, de la fila en la que está la palabra recibida, a dicha palabra. Luego los patrones de error que se corrigen vienen dados por los primeros elementos de cada fila.

Debido al hecho de que siempre corregiremos por la palabra del código más cercana a la palabra recibida se deduce que "sólo debe haber una palabra del código" que tenga distancia mínima a la palabra recibida. En caso contrario no sabriamos cual de las palabras elegir como correcta, se dice entonces que no distinguiriamos el error.

La proposición 4.3 nos indica que utilizando tablas estándar tenemos solucionado este problema, pero no podemos corregir adecuadamente todos los errores mediante una tabla estándar.

El siguiente teorema nos indica cuando se distinguen dos errores.

Teorema 4.3
Sea $\mathcal{C}[n,m]$ un código lineal y sea P un procesador de error para dicho código. Sea $S = \{e_1, \ldots, e_s\}$ un conjunto de patrones de error. El procesador de error P puede distinguir estos patrones de error sí y sólo sí cada patrón está en una fila distinta de la de los otros patrones (en una tabla del código).

Demostración:
$\Rightarrow$ | Sea e un patrón de error y $v \neq e$ y que esté en la misma fila que e. Entonces tendremos que $u = v - e$, con $u \in \mathcal{C}[n,m]$ y tal que $u \neq 0$. Sí el procesador de error corrige el error e entonces corregirá v a u.

Supongamos que el procesador P corrige el error v, entonces corregirá v a 0, lo cual no es posible. Luego no existirá un procesador de error que distinga dos errores situados en la misma fila.

$\Leftarrow$ | Sean e_i y e_j con $i \neq j$, entonces podemos construir una tabla estándar con e_i y e_j como primeros elementos de dos filas.

Con dicha tabla se tiene que para toda palabra u del código el procesador corregirá las palabras $u + e_i$ y $u + e_j$ a u.

Luego el procesador de error P puede distinguir, corregir, dos patrones de error cuando esten en distinta fila, $i \neq j$.

■

Según todo lo que hemos visto podremos corregir tantos errores como clases de equivalencia tenga el $\mathbb{F}_q$-espacio vectorial $\mathbb{F}_q^n / \mathcal{C}[n, m]$, ya que un procesador de error, P, distingue dos errores sí y sólo sí estan en distinta fila, es decir, sí y sólo sí pertenecen a distinta clase de equivalencia.

4.4.1 Método práctico

Con lo que hemos visto un método "práctico" para detectar y corregir errores será el siguiente:

Dado un código lineal $\mathcal{C}[n, m]$:

- Construir una tabla estándar, utilizando como primer elemento de cada fila el tipo de error que queremos detectar y corregir.
- Una vez recibida una palabra comparar con las q^n palabras de la tabla.
- Seleccionar como palabra correcta la palabra que se encuentre en la fila cero y en la columna de la palabra recibida.

Este método aunque práctico es poco util, ya que hay que realizar q^n comparaciones y, a medida que aumenta n q^n crece aun más rapidamente.

4.5 Detección y corrección de errores mediante la matriz de control

Según el método que hemos visto antes necesitariamos almacenar una tabla de q^n elementos y realizar, a lo sumo, q^n comparaciones.

Esto no es práctico ya que si utilizamos códigos con palabras de longitud algo grande la cantidad de memoria necesaria para almacenar la tabla será bastante grande, así como el número de comparaciones necesarias. Por ejemplo para $q = 2$ y $n = 25$ tendremos que almacenar $2^{25} = 33.554.432$ palabras, lo cual requiere una gran inversión en memoria y gran cantidad de tiempo para realizar una comparación.

Para evitar las q^n comparaciones y el almacenamiento de toda la tabla estándar podemos utilizar la matriz de control del código lineal.

Proposición 4.4
Sea $\mathcal{C}[n,m]$ un código lineal y H su matriz de control. $u, v \in \mathbb{F}_q^n$ están en la misma fila sí y sólo sí $H \cdot u^t = H \cdot v^t$.

Demostración:
Como ya hemos visto antes u y v están en la misma fila sí y sólo sí se verifica que $u - v \in \mathcal{C}[n,m]$ entonces se tiene que:

$$H \cdot (u - v)^t = 0 \Longleftrightarrow H \cdot u^t = H \cdot v^t$$

■

Según esta proposición podemos asignar a cada fila de una tabla estándar un vector constante, el cual unicamente será nulo para la primera fila de dicha tabla.

Definición 4.5 (Sindrome de una palabra)
Dado un código lineal $\mathcal{C}[n,m]$ con matriz de control H llamaremos ***"sindrome de una palabra"****, $u \in \mathbb{F}_q^n$, al siguiente vector $(H \cdot u^t)^t$.*

Como todas las palabras de una fila tienen el mismo sindrome y el error que corregiremos será el primer elemento de cada fila entonces unicamente almacenaremos el primer elemento de cada fila junto con su sindrome. De esta forma necesitaremos menos memoria para almacenar la tabla.

El algoritmo para corregir un error es el siguiente:

- Recibimos la palabra v.
- Calculamos su sindrome $(H \cdot v^t)^t$.
- Buscamos en la nueva tabla el sindrome anterior y vemos cual de los elementos tiene dicho sidrome, supongamos que e_i es tal que:

$$(H \cdot e_i^t)^t = (H \cdot v^t)^t$$

- La palabra correcta será $u = v - e_i$.

Ademas de necesitar menos memoria para almacenar la tabla de los sindromes también necesitamos menos tiempo para corregir un error ya que unicamente tendremos que hacer q^{n-m} comparaciones, en lugar de las q^n requeridas por la utilización de la tabla estándar.

4.6 Condición para la corrección de errores de peso uno

Los errores de peso uno son una base del $\mathbb{F}_q$-espacio vectorial $\mathbb{F}_q^n$. Por ejemplo para $n = 3$ y $q = 2$:

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

Sea H la matriz de control de un código lineal:

$$\mathcal{C}[n,m] \subset \mathbb{F}_2^n =< e_1, \ldots, e_n >$$

donde los $\{e_i\}_{i=1}^n$ son los errores de peso uno. Según el teorema 4.3 un procesador de error P distingue dos errores e_i y $e_j \iff$ están en diferente fila $\iff H \cdot e_i^t \neq H \cdot e_j^t$. Ahora bien $H \cdot e_i^t = i$-ésima columna de la matriz de control H, luego de esto se deduce:

Corolario 4.2
Un código lineal binario puede distinguir todos los errores de peso uno sí y sólo sí la matriz de control tiene todas las columnas distintas y no nulas.

■

Este corolario lo podemos generalizar a canales no binarios:

Teorema 4.4 (Condición necesaria y suficiente)
Sea $\mathcal{C}[n,m]$ un código lineal con matriz de control H. El código puede corregir todos los errores de peso uno sí y sólo sí todas las columnas de la matriz de control son no nulas y ninguna columna es un multiplo de las restantes columnas[1].

Demostración:
Por el teorema 4.3 los errores de peso uno se distinguen y corrigen $\iff$ estan en diferente fila en una tabla estándar del código $\iff$ tienen distintos sindromes.

[1] Son linealmente independientes dos a dos.

Sea e un error de peso uno, es decir una palabra con todas sus componentes nulas excepto una. Entonces podemos distiguir dos casos:

- Canal binario, sobre $\mathbb{F}_2$.

  Tendremos que $e = e_i$ para algún i, luego $H \cdot e_i^t = i$-ésima columna de la matriz de control H.

- Canal no binario, sobre $\mathbb{F}_q$, con $q \neq 2$.

  Sea $a_i \in \mathbb{F}_q$ la componente no nula de e, entonces tendremos que $H \cdot e^t = a_i \cdot c_i$, donde c_i es la i-ésima columna de la matriz de control del código H.

Luego los sindromes de los errores de peso uno son multiplos de las columnas de la matriz de control, H.

Como el código distingue y corrige todos los errores de peso uno $\Longleftrightarrow$ tienen distinto sindrome, y los sindromes[2] son multiplos de las columnas de la matriz de control se tiene que:

> El código distingue y corrige todos los errores de peso uno $\Longleftrightarrow$
> no hay ninguna columna que sea multiplo de las demás.

Recordar que la palabra cero es multiplo de todas, basta con multiplicar cualquier palabra por el escalar cero.

■

Según este teorema podemos ver si un código distingue y corrige todos los errores de peso uno sólo con mirar las columnas de la matriz de control y comprobar sí son linealmente independientes dos a dos.

4.7 Relación entre la distancia mínima y la matriz de control

En el teorema 2.2, en la página 32, vimos que un código puede corregir todos los errores de peso t sí y sólo sí $d_{min} \geq 2 \cdot t + 1$.

[2] De los errores de peso uno.

Teorema 4.5
Sea $\mathcal{C}[n,m]$ un código lineal con matriz de control H. El código tiene distancia mínima, $d_{min} > d$ sí y sólo sí no existen d columnas de la matriz H que sean linealmente dependientes. Es decir cualquier conjunto de d columnas es linealmente independiente.

Demostración:
En la demostración utilizaremos los siguientes resultados:

1. Como el código es lineal su distancia mínima es el mínimo de los pesos de las palabras, no nulas, del código.

2. Una palabra u pertenece al código sí y sólo sí $H \cdot u^t = 0$.

3. Sea $u = (u_1, \ldots, u_n)$ verificando $H \cdot u^t = 0$ entonces las columnas de H, H_i, donde i es tal que $u_i \neq 0$ son linealmente dependientes.

$\Rightarrow$ | Supongamos que existe una palabra no nula del código, $u = (u_1, \ldots, u_n)$, tal que sea de peso menor o igual que d. Dado que $u \in \mathcal{C}[n,m]$ tendremos que $H \cdot u^t = 0$, es decir:

$$H \cdot u^t = u_1 \cdot H_1 + \cdots + u_n \cdot H_n = 0 \tag{4.2}$$

Como u es de peso, a lo sumo, d entonces tendrá, como mucho, d componentes no nulas, por comodidad supondremos que son las d primeras. De esta forma la ecuación (4.2) nos queda:

$$H \cdot u^t = u_1 \cdot H_1 + \cdots + u_d \cdot H_d = 0$$

Como $\{u_i\}_{i=1}^{d}$ son no nulos entonces tenemos d columnas linealmente dependientes en H.

Hemos llegado a esta conclusión suponiendo la existencia de un elemento de peso menor o igual que d en el código, pero por hipótesis esto no se puede dar ya que $d_{min} > d$, entonces en la matriz de control H no existen d columnas linealmente dependientes.

$\Leftarrow$ | Supongamos que H tiene d columnas linealmente dependientes, y supongamos que son las d primeras, por comodidad. Luego existen $\lambda_1, \ldots, \lambda_d \in \mathbb{F}_q$, no todos nulos, tales que:

$$\lambda_1 \cdot H_1 + \ldots + \lambda_d \cdot H_d = 0$$

Tomemos $u_i = \lambda_i$ para $i = 1, \ldots, d$ y $u_i = 0$ para $i = d+1, \ldots, n$ y construyamos $u = (u_1, \ldots, u_n)$. Por construcción tendremos que $H \cdot u^t = 0$, luego $u \in \mathcal{C}[n, m]$ y u es no nula ya que en sus d primeras componentes hay, por lo menos, una que no es nula. Es decir u es una palabra de peso, a lo sumo, d.

Luego si H tiene d columnas linealmente dependientes entonces el código tiene palabras de, a lo sumo, peso d. Como por hipótesis H no tiene d columnas linealmente dependientes entonces el código tiene palabras de peso mayor que d, es decir su distancia mínima es mayor que d, $d_{min} > d$.

■

Este teorema nos indica lo bueno que es un código lineal en virtud de las columnas de su matriz de control. El sistema para elegir buenos códigos consiste en elegir bien las columnas de su matriz de control y tomar como código el determinado por su matriz de control.

4.8 Ejemplos

4.8.1 Código del bit de control de paridad

Este código es un código de $2^7 = 128$ palabra palabras. Es un subconjunto de $\mathbb{F}_2^8$ el cual tiene $2^8 = 256$ palabras, con lo cual el número de patrones de error que tendremos será $2^8 - 2^7 = 128$.

Tabla estándar del código

Como el código tiene $2^7 = 128$ palabras la tabla tendrá $2^7 = 128$ columnas y el número de filas será $2^{8-7} = 2$.

Debido a la gran cantidad de palabras de este código no construiremos su tabla estándar.

Sindromes

Sea H la matriz de control, en forma estándar, del código:

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

Errores	*Sindromes*
00000000	0
10000000	1

Tabla 4.1: Tabla de sindromes del código del bit de control de paridad.

4.8.2 Código de triple repetición

Este código es un código de $2^1 = 2$ palabras. Es un subconjunto de $\mathbb{F}_2^3$ el cual tiene $2^3 = 8$ palabras, con lo cual el número de patrones de error que tendremos será $2^3 - 2^1 = 6$.

Tabla estándar del código

Como el código tiene $2^1 = 2$ palabras la tabla tendrá $2^1 = 2$ columnas y el número de filas será $2^{3-1} = 4$.

La primera fila estará formada por las palabras del código con la condición de que el 0 sea el primer elemento. Luego la primera fila será:

$$000 \quad 111$$

Para la segunda fila cogeremos un elemento de $\mathbb{F}_2^3$ que no este en la fila cero, y el primer elemento de cada fila ha de ser el de mínimo peso de dicha fial. El elemento 100 no esta en la fila cero, luego elegimos ese elemento como primer elemento de la fila uno ya que es de peso uno y no es posible encontrar otro de peso menor. El resto de elementos de la fila serán:

$$Tb(\mathcal{C}[3,1])_{1,i} = Tb(\mathcal{C}[3,1])_{1,0} + Tb(\mathcal{C}[3,1])_{0,i} \quad i = 0,1$$

Luego la segunda fila será:

$$100 \quad 011$$

Siguiendo el mismo razonamiento elegiremos como primer elemento de la fila tres un elemento que no haya aparecido en las filas anteriores y de peso mínimo, por ejemplo 010 y siguiendo el razonamiento anterior completaremos la tabla.

$$\begin{aligned} Tb(\mathcal{C}[3,1])_{2,0} &= 010 \\ Tb(\mathcal{C}[3,1])_{3,0} &= 001 \end{aligned}$$

000	111
100	011
010	101
001	110

Tabla 4.2: Tabla estándar del código de triple repetición.

Corrección de errores

Supongamos que recibimos la palabra 011, que no pertenece al código. Para corregir el código haremos:

- Localizamos el lugar de dicha palabra en la tabla.

$$Tb(\mathcal{C}[3,1])_{1,1} = 011$$

- Elegimos como palabra correcta la palabra que esté en la misma columna y en la fila cero.

$$Tb(\mathcal{C}[3,1])_{0,1} = 111$$

El error cometido vendrá dado por el primer elemento de la fila en la que se encuentre la palabra recibida:

$$Tb(\mathcal{C}[3,1])_{1,0} = 100$$

Luego en la transmisión se ha cometido un error de peso $w(100) = 1$ en el primer bit.

Sindromes

Supondremos que queremos corregir los mismos errores que se corrigen con la tabla 4.2.

Sea H la matriz de control, en forma estándar, del código:

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

Para corregir errores supongamos que recibimos la palabra 101, calculamos su sindrome que será 10. Entonces utilizando la tabla 4.3 buscamos el error que tiene sindrome 10, dicho error es 010. Luego la palabra transmitida será $101 - 010 = 111$. Recordar que $-1 = 1$ en $\mathbb{F}_2$.

Errores	*Sindromes*
000	00
100	11
010	10
001	01

Tabla 4.3: Tabla de sindromes del código de triple repetición.

4.8.3 Código de triple control

Este código es un código de $2^3 = 8$ palabras. Es un subconjunto de $\mathbb{F}_2^6$ el cual tiene $2^6 = 64$ palabras, con lo cual el número de patrones de error que tendremos será $2^6 - 2^3 = 56$.

Tabla estándar del código

Como el código tiene $2^3 = 8$ palabras la tabla tendrá $2^3 = 8$ columnas y el número de filas será $2^{6-3} = 8$.

La primera fila estará formada por las palabras del código con la condición de que el 0 sea el primer elemento. Luego la primera fila será:

000000 100110 010101 001011 111000 011110 101101 110011

Para la segunda fila cogeremos un elemento de $\mathbb{F}_2^6$ que no este en la fila cero, y el primer elemento de cada fila ha de ser el de mínimo peso de dicha fila. El elemento 100000 no está en la fila cero, luego elegimos ese elemento como primer elemento de la fila uno ya que es de peso uno y no es posible encontrar otro elemento de peso menor. El resto de elementos de la fila serán:

$$Tb(\mathcal{C}[6,3])_{1,i} = Tb(\mathcal{C}[6,3])_{1,0} + Tb(\mathcal{C}[6,3])_{0,i} \quad i = 1, \ldots, 7$$

Luego la segunda fila será:

100000 000110 110101 101011 011000 111110 001101 010011

Siguiendo el mismo razonamiento escogeremos como primer elemento de la fila dos un elemento de $\mathbb{F}_2^6$ que no aparezca en las filas cero y uno. Como hay elementos de peso uno que no aparecen en dichas filas elegiremos como primer elemento de la fila dos 010000. Y siguiendo el mismo razonamiento construiremos la fila y eligiremos los siguientes elementos:

$$\begin{aligned} Tb(\mathcal{C}[6,3])_{3,0} &= 001000 \\ Tb(\mathcal{C}[6,3])_{4,0} &= 000100 \\ Tb(\mathcal{C}[6,3])_{5,0} &= 000010 \\ Tb(\mathcal{C}[6,3])_{6,0} &= 000001 \end{aligned}$$

Para elegir el primer elemento de la fila siete observaremos que todas las palabras de $\mathbb{F}_2^6$ de peso uno estan en alguna de las filas anteriores, con lo cual el elemento de la fila siete de menor peso tendrá un peso mayor o igual que dos. El elemento 100001 no esta en ninguna de las filas anteriores con lo cual lo elegiremos como primer elemento de la fila siete.

En la tabla 4.4 podemos ver una tabla estándar para el código de triple

000000	100110	010101	001011	111000	011110	101101	110011
100000	000110	110101	101011	011000	111110	001101	010011
010000	110110	000101	011011	101000	001110	111101	100011
001000	101110	011101	000011	110000	010110	100101	111011
000100	100010	010001	001111	111100	011010	101001	110111
000010	100100	010111	001001	111010	011100	101111	110001
000001	100111	010100	001010	111001	011111	101100	110010
100001	000111	110100	101010	011001	111111	001100	010010

Tabla 4.4: Tabla estándar del código de triple control.

repetición. No es la única tabla estándar. Para obtener otras tablas estándar bastará con elegir distintos elementos como primer elemento de cada fila obteniendo las mismas filas, pero en ordenadas de distinta forma.

Corrección de errores

Supongamos que recibimos la palabra 111100, la cual no pertenece al código. Para corregir el error haremos:

- Localizamos el lugar de dicha palabra en la tabla.

$$Tb(\mathcal{C}[6,3])_{4,4} = 111100$$

- Elegimos como palabra correcta la palabra que esté en la misma columna y en la fila cero.

$$Tb(\mathcal{C}[6,3])_{0,4} = 111000$$

El error cometido vendrá dado por el primer elemento de la fila en la que se encuentre la palabra recibida:

$$Tb(\mathcal{C}[6,3])_{4,0} = 000100$$

Luego en la transmisión se ha cometido un error de peso $w(000100) = 1$ en el cuarto bit.

Sindromes

Supondremos que queremos corregir los mismos errores que se corrigen con la tabla 4.4.

Sea H la matriz de control, en forma estándar, del código:

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

Obviamente requiere menor costo almacenar la tabla 4.5 que almacenar la tabla 4.4.

Para corregir errores supongamos que recibimos la palabra 100011, calculamos su sindrome que será 101. Entonces utilizando la tabla 4.5 buscamos que error tiene sindrome 101, dicho error es 010000. Luego la palabra transmitida será $100011 - 010000 = 110011$. Recordar que $-1 = 1$ en $\mathbb{F}_2$.

Error	*Sindrome*
000000	000
100000	110
010000	101
001000	011
000100	100
000010	010
000001	001
100001	111

Tabla 4.5: Tabla de sindromes del código de triple control.

4.9 Ejercicios

Ejercicio 4.1
Programar la corrección de errores de un código lineal utilizando sindromes.

Solución:
Se ha programado el funcionamiento de un procesador de error para el código de triple control utilizando la tabla de sindromes.

El programa está escrito en lenguaje C y tanto el código como los binarios para *MS-DOS* estan el el disco adjunto.

El funcionamiento del programa es el siguiente:

- El programa pide, por teclado, una palabra de seis digitos binários. Para prevenir el funcionamiento anómalo del programa se ha recurrido a admitir una cadena, de hasta 80 caracteres, como entrada. El programa seleccionará los seis primeros caracteres numéricos y los convertirá a $\mathbb{F}_2$, considerando a estos como la palabra recibida.

- El programa conoce, a priori, las palabras del código, su matriz de control y los patrones de error.

- Seguidamente calculará el sindrome de la palabra recibida, así como los sindromes de todas palabras de peso uno, errores, y el de una palabra de peso dos, para detectar cuando ocurre un error de peso dos.

- Sí el error es peso menor o igual que uno el programa da la palabra transmitida originalmente.

- Sí el error es de peso dos el programa indica que ocurrio un error de peso dos, y no nos da la palabra que se transmitió originalmente. Para poder calcular esta palabra sería necesario tener alguna hipótesis adicional sobre que error de peso dos es el que más se da en el canal de transmisión.

■

Ejercicio 4.2
Programar la la corrección de errores de un código lineal utilizando una tabla estándar.

Solución:
Se ha programado el funcionamiento de un procesador de error para el código de triple control utilizando una tabla estándar.

El programa está escrito en lenguaje C y tanto el código como los binarios para *MS-DOS* estan el el disco adjunto.

El funcionamiento del programa es el siguiente:

- El programa pide, por teclado, una palabra de seis digitos binários. Para prevenir el funcionamiento anómalo del programa se ha recurrido a admitir una cadena, de hasta 80 caracteres, como entrada. El programa seleccionará los seis primeros caracteres numéricos y los convertirá a $\mathbb{F}_2$, considerando a estos como la palabra recibida.

- El programa conoce, a priori, las palabras del código y los patrones de error.

- Seguidamente calculará la tabla estándar a partir de las palabras del código y los patrones de error.

- Sí el error es peso menor o igual que uno el programa da la palabra transmitida originalmente.

- Sí el error es de peso dos el programa indica que ocurrio un error de peso dos, y no nos da la palabra que se transmitió originalmente. Para poder calcular esta palabra sería necesario tener alguna hipótesis adicional sobre que error de peso dos es el que más se da en el canal de transmisión.

■

Capítulo 5

Códigos r-perfectos

Definición 5.1 (Códigos r-perfectos)
*Sea $\mathcal{C}[n,m]$ un código. Se dice que es "**r-perfecto**" si para cada palabra $v \in \mathbb{F}_q^n$ existe una única palabra $u \in \mathcal{C}[n,m]$ tal que $d(u,v) \leq r$.*

Observacion 5.0.1

- *Como detecta y corrige errores de peso menor o igual que r entonces su distancia mínima verifica que $d_{min} \geq 2 \cdot r + 1$.*

Definición 5.2 (Bola o disco, abiertos, de centro u y radio r)
Llamaremos bola o disco, abiertos, de centro u y radio r y los denotaremos por $D(u,r)$ al siguiente conjunto:

$$D(u,r) = \{\ v \in \mathbb{F}_q^n \quad verificando\ d(u,v) < r\ \}$$

Definición 5.3 (Bola o disco, cerrados, de centro u y radio r)
Llamaremos bola o disco, cerrados, de centro u y radio r y los denotaremos por $\overline{D}(u,r)$ al siguiente conjunto:

$$\overline{D}(u,r) = \{\ v \in \mathbb{F}_q^n \quad verificando\ d(u,v) \leq r\ \}$$

En las dos definiciones anteriores $u \in \mathbb{F}_q^n$.

5.1 Códigos binarios r-perfectos

Los siguientes resultados pueden ser generalizados para códigos no binarios.

5.1.1 Número de palabras en un disco cerrado

Lema 5.1 (Número de palabras en un disco cerrado)
Dado un disco cerrado $\overline{D}(u,r)$ donde $u \in \mathbb{F}_2^n$ y $r \in \mathbb{Z}$ se tiene que el número de palabras que hay en dicho disco, $|\overline{D}(u,r)\ |$, es:

$$|\overline{D}(u,r)\ | = \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{r}$$

Desmostración:
El número de palabras en $\overline{D}(u,r)$ es el número de palabras que podemos construir variando hasta r bits.

Dada u hay $\binom{n}{i}$ palabras v tales que $d(u,v) = i$, luego por la definición de disco cerrado y distancia tendremos que:

$$|\overline{D}(u,r)\ | = \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{r}$$

■

Observacion 5.1.1

- *De este lema se deduce que el número de palabras que hay en un disco cerrado*[1] *únicamente depende de su radio r y nunca de su centro. Luego utilaremos la siguiente notación:*

$$|\overline{D}(u,r)\ | = |\overline{D}_r\ |$$

[1] Con la distancia de Hamming.

5.1.2 ¿Cuantos códigos r-perfectos existen?

Lema 5.2
Un código r-perfecto, $\mathcal{C}[n,m]$ es unión disjunta de discos cerrados centrados en palabras del código y radio r. Para códigos binários.

Demostración:
Como $\mathcal{C}[n,m]$ es un código r-perfecto se tiene que:

$$\mathbb{F}_2^n = \bigcup_{u \in \mathcal{C}[n,m]} \overline{D}(u,r) \tag{5.1}$$

donde las $\overline{D}(u,r)$ para $u \in \mathcal{C}[n,m]$ son disjuntas entre sí, es decir:

$$\overline{D}(u_i,r) \bigcap \overline{D}(u_j,r) = \emptyset \quad para\ i \neq j$$

La ecuación (5.1) se deduce de la definición de código r-perfecto:

- Que $\mathbb{F}_2^n$ es la unión de discos cerrados se deduce del hecho de que al ser $\mathcal{C}[n,m]$ un código r-perfecto dada $v \in \mathbb{F}_2^n$, una palabra cualquiera, siempre existe una palabra $u \in \mathcal{C}[n,m]$ tal que $d(u,v) \leq r$.
- Y el hecho de que los discos sean disjuntos se deduce del hecho de que las palabras $u \in \mathcal{C}[n,m]$ anteriores son unicas[2].

■

En $\mathbb{F}_2^n$ tenemos 2^n palabras y en $\mathcal{C}[n,m]$ tenemos 2^m palabras. De esto y de los lemas 5.1 y 5.2 se deduce que un código r-perfecto verifica:

$$2^n = 2^m \cdot \left(\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{r} \right)$$

Luego tendremos tantos códigos r-perfectos como n y m verifiquen la condición anterior.

[2]Por definición de código r-perfecto.

Proposición 5.1
Si existe un código $\mathcal{C}[n,m]$ r-perfecto entonces no existe ningún código $\mathcal{C}[n,m']$ con $m' > m$ y que tenga distancia mínima mayor que $2 \cdot r$.

Demostración:
Sea $\mathcal{C}[n,m]$ un código r-perfecto. Entonces por los lemas 5.1 y 5.2 tendremos que:

$$2^n = 2^m \cdot |\overline{D}_r| \tag{5.2}$$

Sea $\mathcal{C}'[n,m']$ con $m' > m$ tal que $d_{min} > 2 \cdot r$ es decir, $d_{min} \geq 2 \cdot r + 1$ entonces el código corrige errores de peso menor o igual que r, luego $\mathcal{C}'[n,m]$ es r-perfecto. Entonces:

$$2^n = 2^{m'} \cdot |\overline{D}_r|$$

Teniendo en cuenta esta última ecuación y (5.2) llegamos a la conclusión de que $m' = m$.

■

Luego para una distancia mínima fijada, $2 \cdot r + 1$, los códigos r-perfectos tienen los bits de información, m, máximos. Para códigos con igual longitud de palabra.

5.1.3 Casos posibles de códigos binarios perfectos

Los códigos de repetición

Estos códigos tienen longitud n y repiten cada bit de información $n-1$ veces. Estos códigos son de la forma $\mathcal{C}[n,1]$, donde $n = 2 \cdot r + 1$, con r arbitrario.

$$\begin{aligned} 2^n &= 2^{2 \cdot r+1} = (1+1)^{2 \cdot r+1} = \sum_{i=0}^{2 \cdot r+1} \binom{2 \cdot r+1}{i} 1^{2 \cdot r+1-i} \cdot 1^i = \\ &= \sum_{i=0}^{2 \cdot r+1} \binom{2 \cdot r+1}{i} = \binom{2 \cdot r+1}{0} + \binom{2 \cdot r+1}{1} + \cdots + \binom{2 \cdot r+1}{2 \cdot r+1} = \\ &= 2^1 \cdot \sum_{i=0}^{r} \binom{2 \cdot r+1}{i} \end{aligned}$$

Hemos utilizado el binomio de *Newton* y la siguiente propiedad de los números combinatorios:

$$\binom{n}{r} = \frac{n!}{r! \cdot (n-r)!} = \binom{n}{n-r}$$

con la cual se tiene que:

$$\binom{2 \cdot r + 1}{i} = \binom{2 \cdot r + 1}{2 \cdot r + 1 - i}$$

Estos códigos son r-perfectos.

Los códigos de Hamming

Los códigos de *Hamming* verifican:

$$2^{2^k - 1} = 2^{2^k - k - 1} \cdot \sum_{i=0}^{1} \binom{2^k - 1}{i}$$

Estos códigos son 1-perfectos.

El código de Golay G_{23}

Este código verifica:

$$2^{23} = 2^{12} \cdot \sum_{i=0}^{3} \binom{23}{i}$$

Este código es 3-perfecto.

Capítulo 6

Códigos lineales de Hamming

Hemos visto en el capítulo 4 que un código que corrija todos los errores de peso uno ha de tener una matriz de control con columnas linealmente independientes dos a dos, y en el caso de ser un código binario, sobre $\mathbb{F}_2$, basta con que las columnas sean no nulas y distintas entre sí.

6.1 Códigos binarios de Hamming

Estos códigos son binarios, sobre $\mathbb{F}_2$.

Definición 6.1 (Código binario de Hamming)
*El **"código binario de Hamming"** de orden k es aquel cuya matriz de control H_k es aquella matriz que tiene por columnas todas las palabras binarias no nulas de longitud k. A este código lo denotaremos como $Ham(k)$.*

$$H_3 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Figura 6.1: Matriz de control, en forma estándar, de $Ham(3)$.

Por contrucción estos códigos son binarios y con todas las columnas de su matriz de control distintas y no nulas, entonces por lo visto en el capítulo anterior tendremos que estos códigos corrijen todos los errores de peso uno.

Observando las matrices de control de los códigos de Hamming $Ham(3)$ y $Ham(4)$, en las figuras 6.1 y 6.2 respectivamente, podemos observar lo si-

$$H_4 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura 6.2: Matriz de control, en forma estándar, de $Ham(4)$.

guiente:

Código	Bits totales	Bits de información	Tipo de código
$Ham(3)$	7	4	$\mathcal{C}[7,4]$
$Ham(4)$	15	11	$\mathcal{C}[15,11]$

6.1.1 Los códigos de Hamming son lineales

Por definición un código de Hamming es aquel que tiene como matriz de control a H_k. Luego suponiendo que H_k es la matriz incidente de un subespacio vectorial nos basta con calcular que vectores se anulan al multiplicarlos por H_k y estos formarán un subespacio vectorial.

6.1.2 Parámetros de los códigos de Hamming

Proposición 6.1 (Parámetros de los códigos de Hamming)
El código de Hamming $Ham(k)$ tiene los siguientes parámetros:

1. *La longitud de las palabras de $Ham(k)$ es $2^k - 1$.*
2. *El rango del código $Ham(k)$ es $2^k - k - 1$.*
3. *La distancia mínima, d_{min}, de $Ham(k)$ es 3.*

Demostración:

1. La longitud de las palabras de $Ham(k)$ es $2^k - 1$.

   Como estos códigos son binarios las palabras del código estarán formadas por dos elementos, en particular las columnas de su matriz de control.

La columnas de su matriz de control son todas las palabras de longitud k no nulas. Los elementos de cada columna pueden ser dos, los elementos de $\mathbb{F}_2$, y cada columna tiene k elementos entonces el número de posibles columnas de k elementos de $\mathbb{F}_2$ es 2^k, como no estamos considerando el elemento nulo entonces tendremos que el número de columnas no nulas de longitud k es de $2^k - 1$.

Ahora bien, como el código es lineal tendremos que:

$$H_k \cdot u^t = 0$$

donde H_k es la matriz de control de $Ham(k)$ y $u^t \in Ham(k)$. Entonces para poder multiplicar H_k por el vector columna u^t se tiene que u^t ha de tener el mismo número de columnas que la matriz H_k y como esta tiene $2^k - 1$ columnas entonces el vector u^t tiene $2^k - 1$ filas, luego tiene una longitud de $2^k - 1$.

2. El rango del código $Ham(k)$ es $2^k - k - 1$.

   El rango de un código lineal coincide con el rango de su matriz de control. Para el código $Ham(k)$ se tiene que es del tipo $\mathcal{C}[n = 2^k - 1, m]$ y la matriz de control un código lineal del tipo $\mathcal{C}[n, m]$ es de orden $(n - m) \times n$. En el caso de los códigos lineales $Ham(k)$ se tiene que $n - m = k$ y $n = 2^k - 1$ entonces tendremos que $m = 2^k - k - 1$. De donde se deduce que $Ham(k)$ es un código $\mathcal{C}[2^k - 1, 2^k - k - 1]$, luego su rango es $2^k - k - 1$.

3. La distancia mínima, d_{min}, de $Ham(k)$ es 3.

   Por el teorema 4.5, en la página 89, se tiene que $Ham(k)$ corrige errores de peso 1 luego $d_{min} = 3$.

■

6.1.3 Palabras del código Ham(k)

Ya hemos visto como se construyen los códigos de Hamming, vienen determinados por su matriz de control. Como estos códigos son lineales podemos utilizar su matriz de control para calcular las palabras que forman el código.

El código de Hamming $Ham(3)$ utiliza siete bits para transmitir información, de los cuales tres son de control. Utilizando la matriz de control en forma estándar tendremos que el código será:

$$Ham(3) \subset \mathbb{F}_2^7 = \{ \ (x_1, x_2, x_3, x_4, c_1, c_2, c_3) \mid x_i, c_j \in \mathbb{F}_2 \ \}$$

Para determinar las palabras del código utilizaremos las ecuaciones implícitas del mismo:

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Como sabemos que el código es binario y utiliza cuatro bits para transmitir

$$\begin{aligned} c_1 &= x_1 + x_3 + x_4 \\ c_2 &= x_1 + x_2 + x_3 \\ c_3 &= x_2 + x_3 + x_4 \end{aligned}$$

Figura 6.3: Ecuaciones implícitas del código $Ham(3)$.

información tendremos que el código $Ham(3)$ tiene $2^4 = 16$ palabras.

Todas las palabras del código se pueden calcular a partir de las ecuaciones implícitas del código, figura 6.3.

Para calcular las palabras de cualquier código $Ham(k)$ basta con proceder de la misma manera que hemos hecho con el código $Ham(3)$.

6.1.4 La matriz generadora de Ham(k)

El código lineal $Ham(k)$ es un código del tipo $\mathcal{C}[2^k-1, 2^k-k-1]$, como ya habiamos visto. Luego su matriz generadora será una matriz de orden $(2^k-1)\times(2^k-k-1)$.

La matriz generadora la podemos dividir en dos partes. La primera parte será la matriz identidad de orden $(2^k-k-1)\times(2^k-k-1)$. Mientras que la segunda parte será una matriz de orden $k\times(2^k-k-1)$.

Cada columna de la matriz generadora, siempre en forma estándar, es una base de $Ham(k)$ de tal forma que la primera matriz en la que hemos subdividido la matriz generadora sea una matriz identidad. Una vez calculadas la ecuaciones implícitas de $Ham(k)$ podemos calcular una base para tener la matriz generadora en forma estándar, con lo cual obtenemos que la matriz generadora, en forma estándar, para $Ham(3)$ será:

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

6.2 Ham(k) es el mejor código con parámetros n, m y d

El código $Ham(k)$ es 1-perfecto, luego no existe ningún código con m mayor que este código. Luego su razón, $\frac{m}{n}$, es la mayor que se puede dar para códigos con longitud n y distancia mínima 3.

6.2.1 Ham(3) vs triple control

Nos referiremos al código de triple control como CTC. Los sindromes de ambos códigos son de tres bits.

Sindromes	Errores CTC	Errores $Ham(3)$
000	000000	0000000
100	000100	0000100
010	000010	0000010
001	000001	0000001
110	100000	1000000
101	010000	0001000
011	001000	0100000
111	100001	0010000

Tabla 6.1: Sindromes de $Ham(3)$ y CTC.

Con el código CTC podemos corregir todos los errores de peso uno y además un sólo error de peso dos. ¿Como elegir el error de peso dos que vamos a corregir?. La única forma en la que la elección de un error de peso dos nos daría una corrección fiable sería cuando en el canal de transmisión unicamente se cometíera un error de peso dos de todos los posibles errores de peso dos, algo poco probable.

Por el contrario en el $Ham(3)$ se corrigen todos los errores de peso menor o igual que uno y ningún error más, es decir, cualquier palabra en la que se haya cometido un error se puede transformar en una palabra mediante el cambio de un único bit. El cambio de este bit estaría determinado por el sindrome de la palabra recibida.

Proposición 6.2
Para cada palabra $v \in \mathbb{F}_2^n$ *con* $n = 2^k - 1$ *existe una única palabra* $u \in Ham(k)$ *tal que* $d(u, v) \leq 1$.

Demostración:

- Veamos la existencia.

  Sea $v \in \mathbb{F}_2^n$. $H_k \cdot v^t = 0$ sí y sólo sí $v \in Ham(k)$. En caso contrario el sindrome sería una palabra no nula de longitud k, y como todas las palabras de longitud k aparecen en la tabla de sindromes[1] su sindrome se corresponderá con un e_i, donde e_i denota a aquella palabra en la que todos sus bits son nulos excepto el i-ésimo que es uno. Luego se tendrá que $u = v - e_i$ y $u \in Ham(k)$.

- Veamos la unicidad.

  La unicidad se deduce de que la distancia mínima de los códigos $Ham(k)$ es 3, es decir, detecta y corrige todos los errores de peso uno. Que es el error que estamos corrigiendo.

■

Para la transmisión de un mensaje por un canal binario y simétrico con probabilidad de error $p = \frac{1}{1000}$ tenemos que si queremos transmitir un mensaje de 10000 bits la probabilidad de transmisión correcta será:

$$((0.999)^7 + 7 \cdot (0.001) \cdot (0.999)^6)^{\frac{10000}{4}} \simeq 0.949$$

Para el código de triple control está probabilidad era 0.951. Luego la probabilidad de transmisión correcta en ambos códigos es muy similar.

La razón de $Ham(3)$ es $\frac{4}{7} \simeq 0.5714$ mientras que la razón del código de triple control es $\frac{3}{6} = 0.5$. Esto nos indica que es más eficiente $Ham(3)$ ya que para mandar el mismo mensaje utilizará menos bits que el código de triple control.

Para una transmisión de 10000 bits el código de triple control mandará 20000 bits, mientras que el $Ham(3)$ mandará 17500.

[1] Se puede ver en la tabla 6.1 donde $k = 3$.

6.3 Simetrias en Ham(3)

Definición 6.2 (Palabras complementarias)
Para una palabra binaria u se define su ***"palabra complementaria"*** *v como aquella que se obtiene al cambiar ceros por unos y unos por ceros en la palabra u.*

0000000	1111111
1000110	0111001
0100011	1011100
0010111	1101000
0001101	1110010
1100101	0011010
1010001	0101110
1001011	0110100

Figura 6.4: Palabras del código $Ham(3)$.

En la tabla 6.4 se puede ver que en el código $Ham(3)$ esta una palabra y su complementaria.

Definición 6.3 (Rotaciones cíclicas)
Dada una palabra u ***"rotar cíclicamente"*** *u significa desplazar hacia la derecha todos sus bits y poner como primer bit su último bit. Sí tenemos que $u = a_1a_2a_3a_4a_5a_6a_7a_8$ entonces al rotar cíclicamente u obtendremos la siguiente palabra:*

$$a_8a_1a_2a_3a_4a_5a_6a_7$$

Utilizando la tabla 6.4 podemos ver que dada una palabra cualquiera de $Ham(3)$ su rotación cíclica también pertenece a $Ham(3)$. Entonces es obvio que rotar cíclicamente k veces una palabra en $Ham(3)$ produce otra palabra de $Ham(3)$.

Definición 6.4 (Códigos cíclicos)
Reciben el nombre de códigos cíclicos aquellos códigos en los que la rotación cíclica de cualquier palabra del código produce otra palabra del código.

De todo esto se deduce la siguiente proposición:

Proposición 6.3
El código $Ham(3)$ es un código cíclico en el cual las palabras complementarias de una del código pertenecen al código.

Capítulo 7

Códigos de Golay

Vamos a dar la construcción de *Tukyn* a partir de los códigos de *Hamming*.

Vamos a utilizar las propiedades de los códigos de *Hamming* siguientes:

- Son códigos cíclicos.
- Contienen palabras complementarias.

7.1 Preliminares

Consideremos el código $Ham(3)$, si rotamos cíclicamente sus palabras:

$$a_1a_2a_3a_4a_5a_6a_7 \rightarrow a_7a_1a_2a_3a_4a_5a_6$$

obtenemos otro código lineal, K con las mismas propiedades que $Ham(3)$:

- $n = 7$
- $m = 4$
- $d_{min} = 3$

y verificando que las únicas palabras que tienen en común ambos códigos son el 0 y el 1.

Además podemos añadir en los dos códigos un octavo bit de "control de paridad" y hacer que mantengan las propiedades. Luego tendremos unos nuevos códigos $Ham(3)'$ y K' de modo que:

- $n = 8$
- $m = 4$
- $d_{min} = 3$

Estos nuevos códigos lineales que hemos construidos nos van a servir para construir otros códigos lineales entendiendolos como subespacios. Esto lo podemos hacer aprovechando la estructura de espacios vectoriales que tienen los códigos $Ham(3)'$ y K', la cual nos permite realizar operaciones con dichos espacios:

- Sumas.
- Productos directos.
- Intersecciones.
- Uniones.

Las propiedades que tienen estos códigos se obtienen de las propiedades de las operaciones con subespacios:

$$\begin{aligned} \dim(Ham(3)' \times K') &= \dim Ham(3)' + \dim K' \\ \dim(Ham(3)' + K') &= \dim Ham(3)' + \dim K' - \dim(Ham(3)' \bigcap K') \end{aligned}$$

7.2 Construcción del código Golay G_{24}

Este código es un código lineal 3-perfecto, es del tipo $\mathcal{C}[24, 12]$ y lo denotaremos como G_{24}.

Definición 7.1 (Código de Golay G_{24})
El código de Golay G_{24} consiste en todas las palabras de longitud 24 que son de la forma:

$$a + x, b + x, a + b + x$$

donde $a, b \in Ham(3)'$ y $x \in K'$.

Por ejemplo, si $a = 11010001$, $b = 10010110$ y $x = 01011001$ la palabra del código G_{24} que definen es:

$$\underbrace{10001000}_{a+x}\underbrace{11001111}_{b+x}\underbrace{00011110}_{a+b+x}$$

Proposición 7.1
Sea $u \in G_{24}$ entonces está determinada de forma única por $a, b \in Ham(3)'$ y $x \in K'$.

Demostración:
Supongamos que existen $c, d \in Ham(3)'$ e $y \in K'$ tales que:

$$a + x, b + x, a + b + x = c + y, d + y, c + d + y$$

Como $a+x = c+y$ y $a, c \in Ham(3)'$ y $x, y \in K'$ tendremos que $a+c = x+y$, recordar que son palabras binarias y que $-1 \equiv 1$ en $\mathbb{F}_2$. Luego $a+c$ pertenece a $Ham(3)'$ y a K', entonces tenemos dos posibilidades:

- $a + c = 00000000$

  En este caso tendremos que $a = c$ de donde se deduce que $x = y$. Como tenemos, por hipótesis, que $a + b + x = c + d + y$ entonces $a + b + x = a + d + x$ o lo que es lo mismo $a + a + b = x + x + d$. La suma de una palabra binaria consigo misma es nula entonces $b = d$.

  Con lo cual hemos demostrado que, en este caso, la descomposición de una palabra de G_{24} es única.

- $a + c = 11111111$

  En este caso tendremos que a es la palabra complementaria de c y x lo será de y. Como tenemos, por hipótesis, que $a + b + x = c + d + y$ entonces tendremos que:

  $$\underbrace{a + c}_{11111111} + \underbrace{x + y}_{11111111} + b = d$$

  de donde se tiene que $b = d$.

  Con lo cual hemos demostrado que, en este caso, la descomposición de una palabra de G_{24} es única.

■

7.2.1 Propiedades de G_{24}

Proposición 7.2
La longitud de las palabras de G_{24} es 24 y su rango es 12.

Demostración:
La longitud de las palabras de G_{24} se deduce de su construcción y es 24.

G_{24} es un código lineal, donde cada palabra se construye a partir de tres palabras, cada una de las cuales pertenece a un código de rango 4. De esto se deduce que G_{24} es un código con $2^4 \cdot 2^4 \cdot 2^4 = 2^{12}$ palabras, luego su rango es 12.

■

Proposición 7.3
La distancia mínima de G_{24} es 8.

7.3 Construcción del código Golay G_{23}

A este código lo denotaremos como G_{23} y su construcción es igual que la del código G_{24} peso sin añadir el bit de control de paridad.

Las características de este código son:

- $n = 23$
- $m = 12$
- $d_{min} = 7$

Capítulo 8

Introducción a los códigos BCH

Estos códigos fueron descubiertos por *Hocquenghem* en 1.959 y también por *Bose* y *Ray-Chaudhuri* en 1.960. De las iniciales de sus nombres es de donde viene el nombre que se les da a estos códigos.

Hemos visto anteriormente que los códigos de Hamming corrigen los errores de peso menor o igual que uno. Podemos modificar los códigos de Hamming para corregir dos errores, pero para eso tenemos que aumentar la distancia mínima de tres a cinco. Esto lo podemos hacer de dos formas:

1. Aumentando la longitud de las palabras.

   En este caso corregir un error en una palabra de longitud cinco o corregir dos errores en una palabra de longitud diez es más o menos equivalente y no introduce mejoras sustanciales en el código.

2. Separando más las palabras del código.

   Este método será el que utilizaremos, y para conseguirlo eliminaremos palabras del código, con lo que obtendremos un subcódigo de un código de Hamming con una distancia mínima mayor, lo que nos permitirá corregir hasta errores de peso dos.

8.1 Matrices de Vandermonde

Para poder desarrollar la teoría de estos códigos vamos a utilizar algunas propiedades de un cierto tipo de matrices, "las matrices de Vandermonde".

Definición 8.1 (Matriz de Vandermonde)
Sea $\mathbb{K}$ un cuerpo cualquiera y sean $\lambda_1, \ldots, \lambda_n$ elementos distintos y no nulos de $\mathbb{K}$. Denotaremos la ***"matriz de Vandermonde"*** *asociada a estos elementos como $V(\lambda_1, \ldots, \lambda_n)$ y dicha matriz será:*

$$V(\lambda_1, \ldots, \lambda_n) = \begin{pmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \cdots & \lambda_n^2 \\ \vdots & \vdots & & \vdots \\ \lambda_1^n & \lambda_2^n & \cdots & \lambda_n^n \end{pmatrix}$$

Teorema 8.1
Sean $\lambda_1, \ldots, \lambda_n$ elementos distintos y no nulos del cuerpo $\mathbb{K}$. Entonces las columnas de la matriz de Vandermonde $V(\lambda_1, \ldots, \lambda_n)$ son linealmente independientes sobre el cuerpo $\mathbb{K}$.

8.2 Preliminares

Como hemos dicho antes estos códigos serán subcódigos de los códigos de Hamming. Para ello eliminaremos palabras de los códigos de Hamming con el objetivo de obtener un nuevo código con una distancia mínima mayor.

La forma de eliminar palabras será añadiendo nuevas condiciones al código, o lo que es lo mimo añadir nuevas filas a la matriz de control. Estas nuevas filas que añadiremos no serán combinaciones lineales de las filas de la matriz de control, ya que no estariamos haciendo nada. Matemáticamente estaremos añadiendo nuevas ecuaciones implícitas al código, con lo cual reduciremos su dimensión, y como consecuencia el número de palabras del código.

Las nuevas filas que añadiremos serán una función **no lineal** de la filas de la matriz de control. Al utilizar funciones **no lineales** tenemos un cierto control sobre la distancia mínima, ya que esta variará según la función que estemos utilizando.

8.2.1 Funciones no lineales

Ejemplos de funciones *no lineales* son aquellas en las que aparecen potencias de las variables:

$$\begin{aligned} f(x) &= x^2 \\ f(x) &= x^3 \\ &\dots \\ f(x) &= x^n \quad n = 2, 3, \dots \end{aligned}$$

Sobre $\mathbb{F}_2$ es dificil encontrar funciones *no lineales*, las del tipo anterior son lineales en este cuerpo:

$$\begin{aligned} \mathbb{F}_2 &\longrightarrow \mathbb{F}_2 \\ x &\longrightarrow x^n = x \end{aligned}$$

La función x^n es *no lineal* para valores de n distintos de 0 y 1, pero en $\mathbb{F}_2$ la función *no lineal* x^n, $n \neq 0$, es equivalente a la función x, la cual es lineal.

Es por este motivo que vamos a considerar funciones *no lineales* sobre potencias de $\mathbb{F}_2$, es decir sobre $\mathbb{F}_2^n$ con $n = 2, 3, \dots..$

8.2.2 Funciones no lineales sobre $\mathbb{F}_2^n$

Dado $\mathbb{F}_2^n$ podemos dotarle de estructura de grupo definiendo la operación suma componente a componente.

El número de elementos que tiene $\mathbb{F}_2^n$ es 2^n. Luego como tiene estructura de grupo podemos establecer un isomorfismo con un grupo que tenga orden 2^n, este grupo será $\mathbb{Z}/2^n$. El isomorfismo será el siguiente:

$$\begin{aligned} \mathbb{F}_2^n &\xrightarrow{\sim} \mathbb{Z}/2^n \simeq \mathbb{F}_{2^n} \\ (a_0, \dots, a_{n-1}) &\longrightarrow a_0 \cdot 2^0 + a_1 \cdot 2^i + \dots + a_{n-1} \cdot 2^{n-1} \end{aligned}$$

Es decir consideramos los elementos de $\mathbb{F}_2^n$ como la representación binaria de los 2^n primeros números enteros, incluido el cero. Luego podremos representar los números enteros comprendidos entre 0 y $2^n - 1$, ambos inclusive.

Ahora podemos definir la siguiente familia de funciones *no lineales*:

$$\begin{aligned} f_i : \mathbb{F}_{2^n} &\longrightarrow \mathbb{F}_{2^n} \\ x &\longrightarrow x^i \end{aligned}$$

para $i = 2, 3, \ldots$.

Por ejemplo para $i = 2$ y $n = 4$ tendremos:

$$\begin{aligned} f_2(0) &= 0^2 = 0 \\ f_2(1) &= 1^2 = 1 \\ f_2(2) &= 2^2 = 4 \\ f_2(3) &= 3^2 = 5 \end{aligned}$$

8.3 Construcción de un código BCH

Veamos la construcción de un código BCH en concreto.

Consideremos la matriz de control del código de Hamming $Ham(4)$, H_4. Las columnas de esta matriz son todas las palabras no nulas de $\mathbb{F}_2^4$, por el isomorfismo visto antes podemos asignar a cada columna de H_4 un elemento de $\mathbb{Z}/2^4$.

Podemos ordenar las columnas de H_4 de la forma que más nos guste, lo único que hay que tener en cuenta es que habrá que permutar los bits de las palabras del código de la misma manera en que permutemos las columnas de la matriz.

Antes de reordenar las columnas eligiremos un elemento primitivo de $\mathbb{Z}/2^n$, α. En nuestro caso elegiremos $\alpha = 2$.

Recordar que H_4 tendrá $2^4 - 1$ columnas y la columna i-ésima se corresponderá con un elemento $a_i \in \mathbb{Z}/2^4$. Pero al ser α un elemento primitivo de $\mathbb{Z}/2^4$ tendremos que $a_i = \alpha^j$ con $j = 0, 1, \ldots, 14$. Luego reordenaremos las columnas en orden descendente de potencias del elemento primitivo.

La reordenación de columnas será la siguiente:

$$\begin{pmatrix} 2^{14} & 2^{13} & 2^{12} & 2^{11} & 2^{10} & 2^9 & 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{pmatrix}$$

o lo que es lo mismo[1]:

$$\begin{pmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \end{pmatrix}$$

que expresado en binario será:

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

esta es la matriz de control de $Ham(4)$, reordenadas las columnas. Ahora para crear el código *BCH* hemos de eliminar palabras del código añadiendo nuevas condiciones a la matriz de control, mediante funciones no lineales.

Las funciones *no lineales* que utilizaremos serán de la forma $f_i(x) = x^i$ con $i = 2, 3, \ldots$. Por ejemplo si añadimos tres nuevas condiciones al código tendremos que utilizar las funciones *no lineales*:

$$\begin{aligned} f_2(x) &= x^2 \\ f_3(x) &= x^3 \\ f_4(x) &= x^4 \end{aligned}$$

Por lo tanto la matriz de control de nuestro código *BCH* será:

$$\begin{pmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \\ 12^2 & 6^2 & 3^2 & 13^2 & 10^2 & 5^2 & 14^2 & 7^2 & 15^2 & 11^2 & 9^2 & 8^2 & 4^2 & 2^2 & 1^2 \\ 12^3 & 6^3 & 3^3 & 13^3 & 10^3 & 5^3 & 14^3 & 7^3 & 15^3 & 11^3 & 9^3 & 8^3 & 4^3 & 2^3 & 1^3 \\ 12^4 & 6^4 & 3^4 & 13^4 & 10^4 & 5^4 & 14^4 & 7^4 & 15^4 & 11^4 & 9^4 & 8^4 & 4^4 & 2^4 & 1^4 \end{pmatrix}$$

o lo que es lo mismo:

$$\begin{pmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \\ 6 & 13 & 5 & 7 & 11 & 8 & 2 & 12 & 3 & 10 & 14 & 15 & 9 & 4 & 1 \\ 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 \\ 13 & 7 & 8 & 12 & 10 & 15 & 4 & 6 & 5 & 11 & 2 & 3 & 14 & 9 & 1 \end{pmatrix}$$

en la figura 8.1, página 120, podemos ver la matriz de control en binario.

[1]Este código es un código polinomial, cuyo generador es el polinomio $h(x) = x^4 + x^3 + 1$, el cual es un divisor de $x^6 - 1$ sobre $\mathbb{F}_2[x]$. Como 2 es un elemento primitivo de $\mathbb{F}_{2^4}$ entonces sabemos que $2^4 + 2^3 + 1 = 0$ en $\mathbb{F}_{2^4}$, lo cual nos servirá para calcular todos los elementos del cuerpo finito de 16 elementos como potencias de 2.

$$\left(\begin{array}{cccccccccccccc}
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
 & & & & & & & & & & & & & & \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
 & & & & & & & & & & & & & & \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & & & & & & & & & & & & & & \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
 & & & & & & & & & & & & & & \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1
\end{array}\right)$$

Figura 8.1: Matriz de control del código BCH(4,2).

8.4 Códigos BCH(k,t)

Definición 8.2 (Códigos BCH(k,t))
El código $BCH(k,t)$ sobre un cuerpo de orden 2^k, basado en el elemento primitivo α, tiene las siguientes columnas en su matriz de control:

$$\left(\begin{array}{c} \alpha^i \\ \alpha^{2\cdot i} \\ \vdots \\ \alpha^{2\cdot t\cdot i} \end{array}\right)$$

en la j-ésima columna $i = (2^k - 1) - j$. A la matriz de control del código $BCH(k,t)$ la denotaremos como $V_{k,t}$.

De esta definición se deduce que la longitud de palabra de un código $BCH(k,t)$ es $2^k - 1$, la misma longitud que para el código $Ham(k)$. Algo que era de esperar ya que es un subcodigo de $Ham(k)$.

8.4.1 Distancia mínima para BCH(k,t)

Los códigos $BCH(k,t)$ son t-perfectos, es decir corrigen error de peso menor o igual que t.

Teorema 8.2 (Los códigos BCH(k,t) son t-perfectos)
Los códigos $BCH(k,t)$ corrigen errores de peso t.

Demostración:
Por el teorema 4.5, en la página 89, para que los códigos $BCH(k,t)$ corrijan errores de peso t su distancia mínima ha de verificar $d_{min} > 2 \cdot t$ o lo que es lo mismo que en su matriz de control no existan $2 \cdot t$ columnas linealmente dependientes.

Sean $i_1, \ldots, i_{2 \cdot t}$ un conjunto de $2 \cdot t$ índices, distintos, cualesquiera del siguiente conjunto:

$$2^k - 2, \ldots, 0$$

Dicho conjunto son los exponentes de las potencias del elemento primitivo que forman la primera fila de la matriz de control. Podemos formar la siguiente matriz con las columnas determinadas por dichos elementos en la matriz de control:

$$\begin{pmatrix} \alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_{2 \cdot t}} \\ \alpha^{2 \cdot i_1} & \alpha^{2 \cdot i_2} & \cdots & \alpha^{2 \cdot i_{2 \cdot t}} \\ \alpha^{3 \cdot i_1} & \alpha^{3 \cdot i_2} & \cdots & \alpha^{3 \cdot i_{2 \cdot t}} \\ \vdots & & & \vdots \\ \alpha^{(2 \cdot t) \cdot i_1} & \alpha^{(2 \cdot t) \cdot i_2} & \cdots & \alpha^{(2 \cdot t) \cdot i_{2 \cdot t}} \end{pmatrix}$$

Es inmediato que la matriz que acabamos de construir es una matriz de Vandermonde $V(\alpha^{i_1}, \ldots, \alpha^{i_{2 \cdot t}})$. Entonces por el teorema 8.1, en la página 116, sus columnas son linealmente independientes. Sus columnas son $2 \cdot t$ columnas cualesquiera, distintas, de la matriz de control. Entonces se tiene que los códigos $BCH(k,t)$ tienen $d_{min} > 2 \cdot t$ o lo que es lo mismo que corrigen errores de peso t. Son t-perfectos.

■

Observacion 8.4.1

- *Gracias a las funciones* no lineales $f_i(x) = x^i$ *para* $i = 2, 3, \ldots$ *podemos controlar la distancia mínima, ya que gracias a la estructura de la matrices de* Vandermonde *podemos calcular cuantas restricciones añadir a un código de* Hamming *para que el código resultante nos corrija errores de peso* t.

- *Como los códigos* $BCH(k,t)$ *son* t*-perfectos, es decir corrigen todos los errores de peso menor o igual que* t*, tendremos que su distancia mínima será mayor o igual que* $2 \cdot t + 1$.

$$d_{min} \geq 2 \cdot t + 1$$

8.4.2 Matriz de control reducida

Las matrices de control para códigos $BCH(k,t)$ vistas hasta ahora poseen filas innecesarias, que son combinación lineal del resto. Para formar estas matrices hemos ampliado la matriz, de control, de un código de Hamming con filas que no fueran linealmente dependientes con las filas de la matriz, de control, del código de Hamming. Pero no hemos impuesto ninguna condición de no linealidad entre las nuevas filas que hemos añadido. Debido a esto las matrices de control que hemos visto no son prácticas.

Por algebra lineal sabemos que las filas de una matriz que son combinación lineal del resto no son necesarias para encontrar la solución de un sistema de ecuaciones, que es lo que hacemos con la matriz de control de un código lineal. Por lo tanto podemos eliminarlar y de esta forma obtenemos una mátriz más pequeña y manejable que hace la misma función que la anterior, servir de matriz de control. A esta matriz la llamaremos matriz de control reducida y la denotaremos por $H_{k,t}$

Por ejemplo la matriz de control, sin reducir, del código $BCH(4,3)$ será:

$$V_{4,3} = \begin{pmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \\ 6 & 13 & 5 & 7 & 11 & 8 & 2 & 12 & 3 & 10 & 14 & 15 & 9 & 4 & 1 \\ 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 \\ 13 & 7 & 8 & 12 & 10 & 15 & 4 & 6 & 5 & 11 & 2 & 3 & 14 & 9 & 1 \\ 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 \\ 5 & 8 & 3 & 15 & 1 & 5 & 8 & 3 & 15 & 1 & 5 & 8 & 3 & 15 & 1 \end{pmatrix}$$

mientras que la matriz de control reducida para el mismo código será:

$$H_{4,3} = \begin{pmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \\ 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 \\ 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 \end{pmatrix}$$

Si ponemos en notación binaria la matriz de control reducida observaremos que podemos seguir eliminando filas que son combinación lineal de otras filas presentes en la matriz, pero esto traería como consecuencia que no podríamos seguir utilizando la notación que hemos utilizado hasta ahora con $\mathbb{Z}/2^4$.

La matriz de control reducida, $H_{k,t}$, es más sencilla y manejable que la matriz de control original, $V_{k,t}$. Pero la matriz $V_{k,t}$ tiene como ventaja que nos permite calcular la distancia mínima del código, mediante las matrices de *Vandermonde*, este argumento no es valido para la matriz $H_{k,t}$.

8.5 Errores que corrige BCH(k,t)

Hemos visto anteriormente que el código $BCH(k,t)$ detecta y corrige todos los errores de peso menor o igual que t.

La siguiente proposición nos demuestra que todos los errores que detecta y corrige el código $BCH(k,t)$ están determinados de forma única.

Proposición 8.1
Sea $u \in BCH(k,t)$ y $v, e \in \mathbb{F}_q^n$ con $n = 2^k - 1$ tales que $v = u + e$. Siendo e un patrón de error de peso t, a lo sumo. Entonces e esta determinado de forma única por el sindrome $V_{k,t} \cdot v^t$.

Demostración:
Supongamos que existe un patrón, e', de error de peso t, a lo sumo, tal que produce una palabra, v', con el mismo sindrome que v.

$$v' = u' + e' \quad u' \in BCH(k,t)$$

Como v y v' tienen el mismo sindrome tendremos que $V_{k,t} \cdot v^t = V_{k,t} \cdot v'^t$, y como $u' \in BCH(k,t)$ entonces $V_{k,t} \cdot u'^t = 0$. De donde se deduce:

$$V_{k,t} \cdot v^t = V_{k,t} \cdot (u' + e')^t = V_{k,t} \cdot e'^t \Longrightarrow V_{k,t} \cdot (v - e')^t = 0$$

Luego $v - e' \in BCH(k,t)$. Como $v = u + e$ entonces tendremos que:

$$v - e' = u + e - e'$$

u y $v - e'$ son palabras de $BCH(k,t)$. Calculemos su distancia:

$$d(u, v - e') = d(u, u + e - e') = w(e - e')$$

Como e y e' son dos patrones de error de peso t, a lo sumo tendremos:

$$w(e) \leq t \ y \ w(e') \leq t \qquad \Longrightarrow \qquad w(e - e') \leq 2 \cdot t$$

Como $BCH(k,t)$ es t-perfecto, corrige todos los errores de peso menor o igual que t, su distancia mínima será:

$$d_{min} \geq 2 \cdot t + 1$$

es decir, las palabras del código estarán, como mínimo, a una distancia de $2 \cdot t + 1$ unas de otras. Luego no puede ser que $d(u, v - e') \leq 2 \cdot t$ con u y $v - e'$ en $BCH(k,t)$. Luego no puede existir un patrón de error con peso t, a lo sumo, tal que produzca una palabra con el mismo sindrome que v.

■

Este teorema nos indica que los errores que detecta y corrige el código $BCH(k,t)$ tienen sindromes distintos. Podemos utilizar el algoritmo de los sindromes que vimos en los códigos lineales para corregir errores. Estos códigos son lineales, ya que son subcódigos de los códigos de Hamming, que son lineales, y además los hemos definido como aquellos códigos que tienen una determinada matriz de control, matriz del subespacio incidente a un subespacio vectorial que será el código $BCH(k,t)$.