Códigos correctores de errores

Beatriz García Bosque

Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Directora del trabajo: Paz Jiménez Seral
27 de junio de 2022

Summary

In this project, we are going to study about coding theory, this is the study of the properties of codes and their respective fitness for specific applications. In particular, we are going to focus on the errors that are generated while coding a message.

To approach the error-correcting-codes theory, we will first introduce each of the elements that take part in a transmission of information from one place to another. We will see what each element consist of and what is their role within the communication process. The information will be sent by r-tuples of 0s and 1s where we will call each of the 0s or 1s that form the r-tuple, its symbols. The information becomes then elements of $\{0,1\}^r$. When the information is transmitted, some symbols can change, a one becomes a zero and vice versa, these changes are called errors. To fix these errors, redundancy symbols are added which allow us to detect the errors and correct them. Coding theory studies the procedures able to transform the information in an n-tuple (word) of zeros and ones with n bigger than r, so errors can be corrected and so the final probability of receiving different information from the one sent is decreased.

In addition of the elements of a transmission, we will define some basic concepts for coding theory. Such as the distance between two words $\big(d(\mathbf{x},\mathbf{y})\big)$ which describes the number of symbols that are different between both words, and the weight $\big(w(\mathbf{x})\big)$, that describes the minimum distance between a word and the null vector.

Besides, we will see that with different codes we can reduce the probability of receiving a message different from the original that is sent. But, to reduce this probability error, we will also need more 'effort'. This 'effort' can be measured with the time that we spend to transmit every element of the new message, or the energy needed to send it. We will measure this 'effort' with the information rate of the code (R). The smaller the rate, the more effort we are going to need.

We will also name some of the devices that take advantage of coding theory and we see daily up to the present day such as computers or telephones. And also some devices that we may not use, but have been some of the most beneficiaries from coding theory which is the case of some satellites as Mariners or Voyagers.

Along with this introduction, we will see some of the most known codes such as Reed-Solomon codes, which consist of a group of error-correcting codes that correct the errors made while coding in blocks for the sake of correcting the message errors that are held together. These of codes are really useful in moments when the errors that have been made in the code can be found next to each other. Which happens a lot of the times, since when there is an 'attack' to a code, (referring to attack to a disturbance made in our code which causes the change of some parts of it) it is easier that if there is more than one change in the code, they are placed together in the code word.

Another very important code that we will come across are Hamming codes. In particular, we will focus on $Hamming(7,4)$ which consist on linear and perfect codes made by 7-tuples. This code verifies that if there is one (or less) errors in the word transmitted, this can be decoded correctly. This is because all code words from Hamming (7,4) have a distance of three or more. And every 7-tuple is at distance one (or less) from one word of our code. This allow us to decode every 7-tuple that is received and guarantee us that if there has been less than two errors, we are going to decode it as the original word that was sent.

We will also mention some simpler codes such as the parity code, to create this codes we do the following. Given a tuple of length n, we add an extra symbol (so we have an $(n+1)$-tuple). If the original n-tuple has an odd number of 1s, the symbol that we add is a 1, and if it has an even number of 0s, the symbol we add is a 0. This result in an even number of 1s in both cases. When reading the

$(n+1)$-tuple, if there is an odd number of 1s, the decoder will know that an error has been made.

Further on, we will see what is the probability of error in the transmission, this is the probability that a word that has been sent is read incorrectly. With this, we will see some examples of how to calculate this probability and how coding decreases this probability. As well as how redundancy in codes can be used in a transmission. Where we refer to redundancy to the method of adding symbols that do not add any new information to the original word. Although redundancy in coding usually diminishes the probability of receiving the original message incorrectly it also requires more effort. Since, by lengthen our code words, our information rate will diminish (and therefore, as we have said before, our effort will be bigger).

Lastly, from this point we will be able to introduce certain theorem that C. E. Shannon published on his paper "Mathematical theory of communication" in 1948 (for more information, see [5]). Shannon said and proved that if the information is transmitted at a rate (R) less than the Channel capacity, which is the rate at which information can be reliably transmitted over a communication channel, then it is possible to transmit information nearly without error at any rate below a limiting rate, CC, where CC is the Channel capacity we have just described. This means that using the right code, it is possible to reduce the chance of error to arbitrarily low amounts, while retaining a decent rate of information transfer, specifically this rate is dependent only on the channel, and not on the error-bound we wish to achieve.

This theorem marks the beginning of coding theory, since it basically says that good codes exist and it is possible to send information nearly error-free up to a maximum rate.

Since these codes had to be used with the aid of often very small electronic apparatus one was specially interested in codes with a lot of structure which would allow relatively simple decoding algorithms. However it is very difficult to obtain highly regular codes without losing the property promised by Shannon's Theorem.

The opposite is also important. If the rate at which information is transmitted is greater than the Channel capacity, then an arbitrarily small probability of error is not possible to achieve. All codes will have a probability of error greater than a certain positive minimal level, and this level increases as the rate increases. So, information cannot be guaranteed to be transmitted reliably across a channel at rates beyond the channel capacity.

Another important thing to point out, is that Shannon's theorem states that good codes exist, but it doesn't show how to construct them. However, some coding techniques such as the Reed–Solomon codes we have named before and will see in more detail later, are close to reaching the theoretical Shannon limit (the maximum rate of error-free data that can theoretically be transferred over the channel if the message can be altered by random data transmission errors), but at a cost of high computational complexity.

Simple schemes such as sending the message n times and use the most repeated coded message if the copies differ are inefficient error-correction methods, unable to asymptotically guarantee that a block of data can be communicated free of error.

Going with Shannon's theorem we will also see how to prove it with the content learned during the mathematics degree being probability the one area of mathematics that is more used. Although some algebra, specially some facts from number theory is also necessary to construct the basic knowledge necessary to understand Shannon's theory and the coding theory that precede. This theorem is not usually seen in mathematics degrees, so it is odd to find the theorem proof in mathematics study books and it makes it difficult to find a detailed proof of the theorem. The proof is not a constructive proof, this is that we do not provide a code that fulfills the theorem statement. Instead, to prove it we will see that for every epsilon, there exists an n big enough such that there exists a code, at which the rate of error is nearly zero. To see this, we will consider all the possibles codes with an M number of words of length n, and we will find a bound for the mean of the probability of error rate of these codes. If we get to see that this bound is less than epsilon, we will have proved the theorem.

All the example codes and calculations that appear in the project are made with the aim to make easier the understanding of the theorem statement and its proof.

Índice general

Capítulo 1

Códigos correctores de errores

1.1. Introducción a los códigos correctores de errores

A la hora de enviar información a través de un medio de comunicación, es común que se produzcan errores en el proceso. La teoría de códigos correctores busca que a pesar de que se produzcan estos errores, la información sea recibida de manera correcta por el receptor del mensaje.

Estos errores ocurren cuando la información es transmitida a través de un medio de comunicación 'ruidoso'. Donde por 'ruido' nos referimos a toda señal no deseada que se mezcla con la señal útil que sí se quiere transmitir. Es decir, que un medio de comunicación ruidoso es aquel donde la información que se desea enviar puede ser alterada debido a causas externas. Por ejemplo, al hablar por teléfono a través del móvil estando en un tren eléctrico en funcionamiento, normalmente no podemos oír correctamente la voz de la otra persona. Ésto ocurre debido a la interferencia de la red del móvil con el campo electromagnético causado por el cable eléctrico del tren. Así que en este caso, el campo eléctrico es el que produce el ruido en la conversación por teléfono.

Los ejemplos más conocidos donde se transmite información a través de medios de comunicación ruidosos son: las conversaciones de teléfono como el caso previamente mencionado, los aparatos de almacenamiento de datos como los DVD o el disco duro que envían la información almacenada al ordenador o el telégrafo.

Uno de los ejemplos históricos más importantes de codificación es el de los satélites como los Mariners o los Voyagers. Para transmitir las fotografías de Marte, Saturno y más planetas tomadas por estos satélites a La tierra se coloca una cuadrícula en la fotografía y por cada cuadrado de la cuadrícula, la escala de grises es medida en un rango de 0 a 63. Estos números están expresados en sistema binario, es decir, cada cuadrado produce una secuencia de seis ceros y unos (de manera que tenemos $2^6 = 64$ tonos de grises como habíamos dicho). Los ceros y unos son transmitidos como dos señales diferentes a la estación receptora en tierra (el Laboratorio de Propulsión a Reacción del Instituto Tecnológico de California en Pasadena en ambos casos) (ver en [4]). Al llegar, debido al efecto del ruido térmico, a veces ocurre que una señal que ha sido transmitida como 0 es interpretada como un 1 y viceversa. Si la 6-tupla de 0s y 1s que hemos mencionado es transmitida de esta forma, los errores recibidos por el receptor tendrán un gran efecto en las fotografías. Para evitar ésto, se añade la llamada *redundancia* en la señal, ésto es que la secuencia transmitida consiste en más información de la necesaria.

Un ejemplo más simple de codificación para canales ruidosos es el sistema usado por una interfaz en serie entre una terminal y un ordenador o entre un ordenador y un teclado. Para poder representar $2^7 = 128$ símbolos distintos, se utilizan series de siete 0s y 1s (es decir, los números enteros de 0 a 127 en binario). En la práctica se añade un símbolo extra de manera que haya una 8-tupla con un número par de 1s. Un fallo en estas interfaces ocurre muy raramente, pero es posible que algún bit incorrecto ocurra. Si hay un fallo, entonces se tendrá un número impar de 1s en la 8-tupla. En este caso, la 8-tupla no es aceptada y la máquina se detiene porque ha detectado un error. Ésto es lo que se suele llamar *código detector de un solo error*.

A la hora de estudiar la teoria de códigos es importante saber los elementos que forman parte de

la transmisión de un mensaje. Hemos mencionado anteriormente que las 6-tuplas de 0s y 1s en una transmisión de fotografías (por ejemplo, en la antes mencionada de Mariner 1969) son sustituídas por series más largas (que llamaremos *palabras*). De hecho, en el caso de Mariner 1969 la palabra consistía en 32 símbolos. Es decir, que en este caso se había diseñado un dispositivo capaz de cambiar las 64 posibles cadenas de información (6-tuplas de 0s y 1s) en 64 posibles *palabras código* (formadas por 32-tuplas de 0s y 1s). Este aparato capaz de transformar las cadenas de información en palabras código es llamado el *codificador*.

Estas palabras código podrán ser afectadas por un ruido aleatorio, que como hemos mencionado antes produce de *errores* que alteran el mensaje original.

En el lado receptor, un aparato llamado el *decodificador* transforma la 32-tupla recibida, si no es una de las 64 palabras código posible, en la palabra código más probable y luego determina la correspondiente 6-tupla (en el caso de Mariner 1969 esta 6-tupla corresponde al tono de gris de un cuadrado en la cuadrícula).

En resumen, en un sistema de comunicación la información siempre va salir de una fuente emisora que quiere enviar un mensaje, la información es convertida por un codificador en un formato que sea posible de transmitir mediante un canal (el medio por el cual el mensaje es enviado). A veces, en el canal pueden ocurrir ciertas perturbaciones llamadas ruido que introducen errores a la información que está siendo transmitida. Finalmente, esta información es procesada por un decodificador para recuperar el formato original en el que se encontraba el mensaje y enviarlo al destinatario final, el receptor.

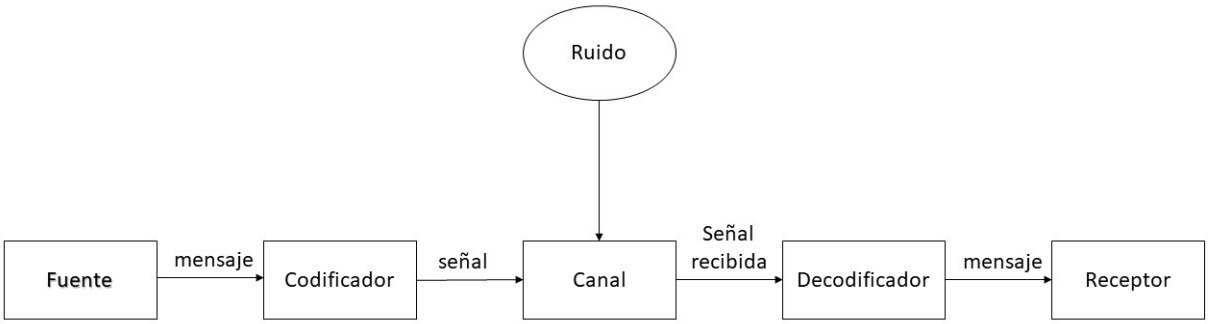

Figura 1.1: Una transmisión básica

El código de Mariner 1969 tiene la propiedad de que si no hay más de 7 de 32 símbolos incorrectos, el aparato está realizando la decisión correcta. Claramente, hay que pagar una 'cuota' para que se pueda corregir el error si hace falta. El precio de esta 'cuota' es el tiempo que tardamos en transmitir la información.

En este caso, si tenemos un tiempo fijo para enviar una palabra y en vez de en seis símbolos la mandamos en 32, el tiempo disponible para la transmisión de cada bit es de solamente $\frac{6}{32}$ del tiempo que tomaría sin codificación, lo que podría incrementar la probabilidad de error.

En la práctica, además de límite de tiempo, hay límites en la cantidad de energía disponible y por tanto límites a la cantidad de bits que podemos añadir al codificar.

Antes, hemos mencionado el uso de redundancia para disminuir la probabilidad de recibir una palabra distinta de la que se había enviado originalmente. La figura siguiente muestra un ejemplo de redundancia mediante repetición de los símbolos.

A simple vista podemos ver que cuanta más redundancia usemos mayor 'cuota' tendremos que 'pagar'. A pesar de que la redundancia nos puede servir para disminuir la probabilidad de que la palabra recibida sea incorrecta, también requiere más esfuerzo.

Aunque en general los símbolos a transmitir puedan ser símbolos de un alfabeto arbitrario, en este trabajo consideraremos siempre que se transmiten series de ceros y unos. Y que la información que queremos transmitir es un elemento de $\{0,1\}^r$

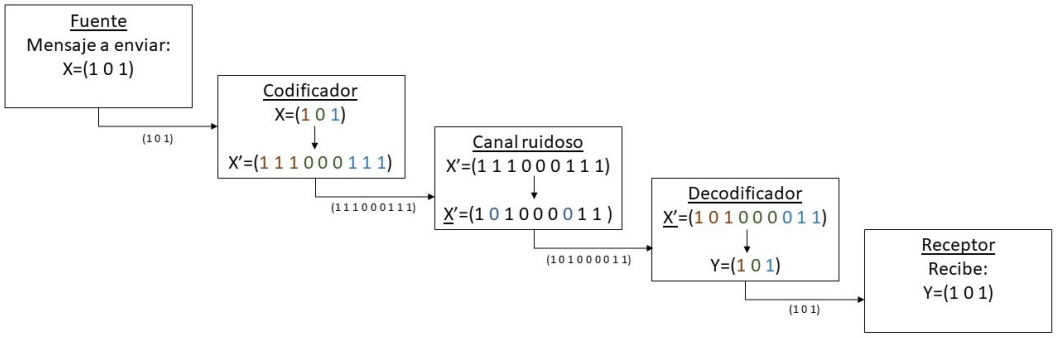

Figura 1.2: Ejemplo de redundancia.

Definición 1.1. Un código (bloque) C de longitud n es un subconjunto de $\{0,1\}^n$.

Si tenemos un código C compuesto por $|C|$ palabras de longitud n, entonces el valor

$$R := n^{-1}\log_2|C|$$

se dice que es la *tasa de información* (o solamente tasa) del código.

El concepto de tasa representa el esfuerzo que nos va a costar enviar el mensaje, a menor R mayor esfuerzo.

Ejemplo 1.1. Supongamos que realizamos dos lanzamientos de una moneda, y queremos transmitir el resultado de los dos lanzamientos mediante palabras de longitud 2 de la siguiente forma:

$$(cara, cara) \to (0,0)$$

$$(cara, cruz) \to (0,1)$$

$$(cruz, cara) \to (1,0)$$

$$(cruz, cruz) \to (1,1)$$

En este caso, en nuestro código tendremos $2^2 = 4$ palabras, por tanto:

$$R = \frac{1}{2}\log_2 4 = \frac{1}{2}2 = 1$$

Ejemplo 1.2. Ahora, supongamos que queremos transmitir el mismo mensaje, pero en vez de con palabras de longitud 2, con palabras de longitud 4, por ejemplo utilizando redundancia:

$$(cara, cara) \to (0,0,0,0)$$

$$(cara, cruz) \to (0,1,0,1)$$

$$(cruz, cara) \to (1,0,1,0)$$

$$(cruz, cruz) \to (1,1,1,1)$$

En este caso, nuestro código tendrá el mismo número de palabras que el código anterior, sin embargo n será distinta. Por tanto:

$$R = \frac{1}{4}\log_2 4 = \frac{1}{4}2 = \frac{1}{2}$$

Es decir, que para mandar un mismo mensaje nos cuesta el doble de esfuerzo mandarlo mediante palabras 4-tuplas que mediante 2-tuplas.

En el código usado en el Mariner, $n = 32$, $M = 64$, luego

$$R = \frac{1}{32}\log 2^6 = \frac{6}{32}\log 2 = \frac{6}{32} = \frac{3}{16}$$

En la teoría de codificación existen muchos tipos de códigos distintos. Por eso veremos ejemplos de distintos tipos de códigos teniendo en cuenta que conforme más símbolos extra usemos más esfuerzo necesitaremos para enviar el mensaje.

Definición 1.2. Un *código lineal* de longitud n es un subespacio de $\mathbb{F}_q^n$, donde $\mathbb{F}_q$ es un cuerpo finito de q elementos.

Notar que si C es un código lineal y su dimensión como espacio $\mathbb{F}_q$-vectorial es k, entonces $|C| = q^k$.

En nuestro caso, como todos los códigos que veamos serán códigos binarios tendremos que $\mathbb{F}_q^n = \{0,1\}^n$. Si C es un código lineal de dimensión k, su tasa es $\frac{k}{n}$.

Hay muchos tipos de códigos y diferentes algoritmos para decodificarlos. Estos algoritmos los podemos dividir en dos tipos. Un algoritmo decodificador *completo* (o *perfecto*) decodifica cada posible palabra recibida en una palabra código. Mientras que un algoritmo decodificador *incompleto* corregirá los mensajes recibidos que contienen unos pocos errores y para los otros posibles mensajes recibidos habrá un fracaso decodificador. En este último caso, el receptor ignora el mensaje, o si es posible, pide una retransmisión. Este último algoritmo puede ser preferible concretamente cuando un error decodificado es muy indeseado.

Ejemplo 1.3. Supongamos que tenemos el código C formado por las palabras $\{(0,0,0,0),(1,1,1,1)\}$. Y supongamos que llega la palabra (0,0,0,1). Esta palabra no se encuentra en nuestro código, por tanto una manera intuitiva de decodificar la palabra es decodificarla como (0,0,0,0) ya que es la palabra de nuestro código con la que tiene menos diferencias. Sin embargo, si nos llegara la palabra (0,0,1,1) ésta tiene la misma cantidad de símbolos iguales a (0,0,0,0) que a (1,1,1,1), por tanto habría que elegir al azar con cual de las dos palabras la decodificamos, añadir algún criterio para poder decodificarla o dejarla sin decodificar. Ésto sería una decodificación incompleta ya que no podríamos decidir como decodificar todas las palabras.

Definición 1.3. Si $\mathbf{x}$ e $\mathbf{y}$ son dos n-tuplas $\mathbf{x_1} = (x_1, x_2, ..., x_n)$, $\mathbf{y} = (y_1, y_2, ..., y_n) \in \{0,1\}^n$, entonces decimos que su distancia de Hamming (o solo distancia) es

$$d(\mathbf{x},\mathbf{y}) := |\{i | 1 \leq i \leq n,\ x_i \neq y_i\}|.$$

Llamamos peso de una palabra a

$$w(\mathbf{x}) = d(\mathbf{x},\mathbf{0})$$

donde por $\mathbf{0}$ nos referimos al vector $(0,0,...,0) \in \{0,1\}^n$

En el ejemplo 1.3,

$$d\big((0,0,0,0),(0,0,0,1)\big) = 1 \leq 3 = d\big((1,1,1,1),(0,0,0,1)\big)$$

$$d\big((0,0,0,0),(0,0,1,1)\big) = 2 = d\big((1,1,1,1),(0,0,1,1)\big)$$

Por tanto, la palabra (0,0,0,1) se encuentra a menor distancia de la palabra (0,0,0,0) que de la palabra (1,1,1,1). Mientras que la palabra (0,0,1,1) se encuentra a la misma distancia de ambas palabras.

La distancia de Hamming tiene las siguientes propiedades:

- $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$
- $d(\mathbf{x},\mathbf{y}) = 0$ si y solo si $\mathbf{x} = \mathbf{y}$
- $d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{z}) + d(\mathbf{z},\mathbf{y})$

- $d(\mathbf{x},\mathbf{y}) = w(\mathbf{x}-\mathbf{y})$

Notar, que si se comete un error en la palabra enviada $\mathbf{x}$, recibimos $\mathbf{v}$ tal que $d(\mathbf{v},\mathbf{x}) = 1$. Y si hubiera otra palabra $\mathbf{y}$ en nuestro código tal que $d(\mathbf{v},\mathbf{y}) = 1$, entonces la tercera propiedad obliga a que $d(\mathbf{x},\mathbf{y}) \leq 2$. Si tenemos un código donde todas las palabras se encuentran a distancia tres y recibimos una palabra a distancia uno de una de las palabras, la palabra recibida no se va a encontrar a distancia uno o menos de ninguna otra de las palabras del codigo.

Definición 1.4. Llamamos $d(C)$ a la mínima distancia que puede haber entre dos palabras cualesquiera de nuestro código.

$$d(C) = \text{mín}\{d(\mathbf{x_i},\mathbf{x_j}) | \mathbf{x_i},\mathbf{x_j} \in C \text{ y } \mathbf{x_i} \neq \mathbf{x_j}\}$$

Además, en códigos lineales se cumple que

$$d(C) = W(C) = \text{mín}\{w(\mathbf{x_i}) | \mathbf{0} \neq \mathbf{x_i} \in C\}$$

En el código del ejemplo 1.3, $d(C) = 4$ ya que solo teníamos dos palabras y la distancia entre ellas era 4. En el caso del Mariner 1969, que hemos mencionado previamente, se utilizaba un código Reed-Muller (32,5), compuesto por $2^6 = 64$ palabras de longitud 32 (ver [7]). En este código, dos palabras código distintas cualesquiera difieren en al menos 16 posiciones, es decir, $d(C) = 16$. Por tanto, una palabra recibida con menos de ocho errores se parece más a la palabra código deseada que a cualquier otra palabra código. Esto hace que el receptor sea capaz de corregir hasta siete errores en una palabra recibida. Y garantiza que si se realizan menos de ocho errores la palabra va a ser decodificada correctamente.

Otro ejemplo es el de los códigos Hamming que consisten en códigos lineales y perfectos (pueden corregir cualquier palabra) cuyos elementos están formados a partir de una base. Por ejemplo, el código $Hamming(7,4) \subset \{0,1\}^7$.

Ejemplo 1.4. $Hamming(4,7)$

Sus palabras son combinaciones lineales de la base

$$\big\{(1,0,1,0,1,0,0),(0,1,1,0,0,1,0),(0,0,1,1,0,0,1),(0,0,0,1,1,1,0)\big\}.$$

El 7 del nombre $Hamming(7,4)$ indica la longitud de las palabras y el 4 indica que el código contiene $2^4 = 16$ palabras en forma de 7-tuplas.

Como en todo código lineal, existe una matriz H, llamada la matriz de control, en este caso

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

tal que las palabras del código son todas las soluciones del sistema

$$H\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Este código es de Hamming porque las columnas de H son precisamente todas las ternas que existen distintas de la nula.

Como podemos ver con la matriz de control, para que $W(C) = 1$, tiene que existir $\mathbf{x_i}$ tal que $w(\mathbf{x_i}) = 1$ y esto solo ocurre cuando una columna es cero (ya que en caso contrario no sería solución del sistema). Además para que $W(C) = 2$, tiene que existir $\mathbf{x_i}$ tal que $w(\mathbf{x_i}) = 2$ y como estamos en módulo dos, ésto solo ocurre cuando dos columnas son iguales. Por tanto, a partir de nuestra matriz de control podemos ver que todas las palabras (menos la nula) tienen por lo menos peso 3. Además, como podemos ver en las palabras de nuestra base, en el código hay palabras de peso 3. Por tanto, el peso del código es 3 y $d(C) = 3$.

Además, solo va a haber $2^3 = 8$ 7-tuplas a distancia uno o menos de una palabra del código (siete a uno de distancia y ella misma). Como tenemos $16 = 2^4$ palabras en nuestro código, tenemos que hay $2^3 2^4 = 2^{16}$ 7-tuplas a distancia uno o menos de alguna palabra de nuestro código. Como una 7-tupla no puede estar a distancia uno o menos de dos palabras distintas de nuestro código (pues $d(C) = 3$), y el total de 7-tuplas posibles son 2^{16}, tenemos que toda 7-tupla se puede decodificar como una palabra del código, o lo que es lo mismo, decimos que el código $Hamming(7,4)$ es un código perfecto.

Igual que hemos formado la matriz H teniendo como columnas las ternas de ceros y unos no nulos. Dado un k entero, podemos formar una matriz H con todas las k-tuplas distintas de 0, H será una matriz (k,n) con $n = 2^k - 1$. Y podemos formar un código cuyas palabras (de longitud n) son las soluciones de $H\mathbf{x} = \mathbf{0}$. Por los mismos argumentos que antes $d(C) = 3$, y el código es perfecto. Ésto es el código $Hamming(n, n-k)$

En la práctica, también se distingue entre *decisiones estrictas* y *decisiones flexibles* según la interpretación de los símbolos recibidos. La mayoría de ellos serán fáciles de distinguir si representan un 0 o un 1 y por tanto el receptor no dudará. Para otros casos sin embargo, esto no será cierto y será cuando preferiramos poner un "?" en vez de decidir si el símbolo es un 0 o un 1. Ésto es comúnmente referido como un *borrado*. Cuanto más complicado sea el sistema más difícil será reconocer los símbolos.

Volvamos al caso donde queremos transmitir dos lanzamientos de un moneda y supongamos que debido a nuestra limitación de tiempo solo podemos transmitir dos símbolos por cada lanzamiento de la moneda. No tiene ningún sentido mandar cada símbolo dos veces en lugar de una vez.

Sin embargo, hay más maneras de codificación además de la repetición. Veamos el siguiente código donde no nos basamos solamente en la mínima distancia para decodificar.

Ejemplo 1.5. Transmitimos el resultado de dos lanzamientos de una moneda como sigue:

$$cara, cara \to (0,0,0,0),$$

$$cara, cruz \to (0,1,1,1),$$

$$cruz, cara \to (1,0,0,1),$$

$$cruz, cruz \to (1,1,1,0).$$

En este caso podemos observar que los dos primeros símbolos transmitidos conducen la verdadera información y los dos símbolos finales son redundantes. El decodificador utiliza el siguiente algoritmo completo de decodificación. Si una 4-tupla recibida no es ninguna de las anteriores, asume que el cuarto símbolo es correcto y que uno de los tres primeros símbolos es incorrecto. Es decir, si por ejemplo el decodificador recibe la secuencia $(1,1,1,1)$ ya que ésta no coincide con ninguna de las secuencias definidas, daría por hecho que el último 1 es correcto y por tanto esta secuencia podría solamente ser $(cara, cruz)$ o $(cruz, cara)$.

Extendamos ahora esta idea para transmitir el resultado de 3 lanzamientos de moneda en un intervalo de tiempo.

Ejemplo 1.6. La información que queremos transmitir es una 3-tupla de 0s y 1s, digamos (a_1, a_2, a_3). En lugar de esta 3-tupla, transmitiremos la 6-tupla $\mathbf{a} = (a_1, ..., a_6)$, donde $a_4 := a_2 + a_3$, $a_5 := a_1 + a_3$, $a_6 := a_1 + a_2$ (considerando la suma una suma módulo 2).

De manera que si sacamos por ejemplo (cara, cara, cruz), considerando entonces $a_1 = 0$, $a_2 = 0$, $a_3 = 1$ y $a_4 := a_2 + a_3 = 0 + 1 = 1$, $a_5 := a_1 + a_3 = 0 + 1 = 1$, $a_6 := a_1 + a_2 = 0 + 0 = 0$ la 6-tupla correspondiente sería: $\mathbf{a} = (0,0,1,1,1,0)$.

De esta forma, tenemos que nuestras $2^3 = 8$ posibles palabras serían:

$(cara, cara, cara) \to (0,0,0,0,0,0)$
$(cara, cara, cruz) \to (0,0,1,1,1,0)$
$(cara, cruz, cara) \to (0,1,0,1,0,1)$
$(cara, cruz, cruz) \to (0,1,1,0,1,1)$
$(cruz, cara, cara) \to (1,0,0,0,1,1)$
$(cruz, cara, cruz) \to (1,0,1,1,0,1)$
$(cruz, cruz, cara) \to (1,1,0,1,1,0)$
$(cruz, cruz, cruz) \to (1,1,1,0,0,0)$

De esta manera hemos creado un código que consiste en ocho palabras (puesto que solo hay $2^3 = 8$ resultados posibles) de longitud 6 cada una. Como hemos mencionado anteriormente, consideramos el ruido como algo añadido al mensaje, es decir la palabra que se reciba **b** será **a+e** donde $\mathbf{e} = (e_1, e_2, ..., e_6)$ es lo que llamaremos la *palabra error*. Notar que debido a que estamos trabajando en módulo dos:

$$b_2 + b_3 + b_4 = (a_2 + e_2) + (a_3 + e_3 + (a_4 + e_4) = (a_2 + a_3 + a_4) + (e_2 + e_3 + e_4) = (a_2 + a_3 + a_2 + a_3) + (e_2 + e_3 + e_4) = (2a_2 + 2a_3) + (e_2 + e_3 + e_4) = (0+0) + (e_2 + e_3 + e_4) = e_2 + e_3 + e_4$$

$$b_1 + b_3 + b_5 = (a_1 + e_1) + (a_3 + e_3 + (a_5 + e_{5)} = (a_1 + a_3 + a_5) + (e_1 + e_3 + e_5) = (a_1 + a_3 + a_1 + a_3) + (e_1 + e_3 + e_5) = (2a_1 + 2a_3) + (e_1 + e_3 + e_5) = (0+0) + (e_1 + e_3 + e_5) = e_1 + e_3 + e_5$$

$$b_1 + b_2 + b_6 = (a_1 + e_1) + (a_2 + e_2 + (a_6 + e_6) = (a_1 + a_2 + a_6) + (e_1 + e_2 + e_6) = (a_1 + a_2 + a_1 + a_2) + (e_1 + e_2 + e_6) = (2a_1 + 2a_2) + (e_1 + e_2 + e_6) = (0+0) + (e_1 + e_2 + e_6) = e_1 + e_2 + e_6$$

Por tanto, podemos definir s_1, s_2, s_3 de la siguiente forma

$$e_2 + e_3 + e_4 = b_2 + b_3 + b_4 := s_1,$$

$$e_1 + e_3 + e_5 = b_1 + b_3 + b_5 := s_2,$$

$$e_1 + e_2 + e_6 = b_1 + b_2 + b_6 := s_3.$$

Ya que el receptor conoce **b**, también conoce s_1, s_2, s_3. Dados s_1, s_2, s_3 el decodificador debe elegir la palabra **e** que es más probable que satisfaga las tres ecuaciones. La más probable será aquella con el menor número de símbolos 1. Si $(s_1, s_2, s_3) \neq (1,1,1)$, tomaremos como **e** a la única 6-tupla con solo un uno que resuelva el sistema.

Y si $(s_1, s_2, s_3) = (1,1,1)$ el decodificador deberá elegir una de las tres posibilidades (1, 0, 0, 1, 0, 0), (0, 1, 0, 0, 1, 0), (0, 0, 1, 0, 0, 1) para **e** (ya que estas tres 6-tuplas satisfacen las ecuaciones y además tienen el mismo número de unos). Es decir que de esta forma una palabra error con un solo error es decodificada correctamente y de entre las otras palabras error hay una con dos errores que es decodificado correctamente.

Las reglas de decodificación está basada en dos suposiciones. La primera es que asumimos que en la comunicación todas las palabras código son igualmente probables. La segunda, es que si $n_1 > n_2$, entonces la probabilidad de n_1 errores es menor que la probabilidad de n_2 errores.

Esto significa que si **y** es recibida, intentamos encontrar una palabra código **x** tal que $d(\mathbf{x}, \mathbf{y})$ sea mínima. Este principio es llamado *decodificación de máxima verosimilitud* y es el que utilizaremos habitualmente.

Si tomamos el código del ejemplo 1.5 y suponemos que recibimos la 4-tupla (1,1,1,1). Aplicando el método de máxima verosimilitud tenemos que las 4-tuplas a menor distancia de ésta son: (0,1,1,1) y (1,1,1,0) correspondientes a los resultados $(cara, cruz)$ y $(cruz, cruz)$ respectivamente, que se encuentran a distancia uno, es decir que la 4-tupla que se encuentra a menor distancia de (1,1,1,1) no es única. Sin embargo, si añadimos otro criterio como el mencionado antes donde asumimos que el último símbolo es correcto, podemos ver que entre nuestras dos posibles palabras la única que coincide en el último dígito con la palabra (1,1,1,1) es la palabra (0,1,1,1). Por tanto, aplicando el principio de máxima verosimilitud con el criterio extra de asumir que el último símbolo es siempre correcto, decodificaríamos (1,1,1,1) como $(cara, cruz)$.

1.1.1. Códigos Reed-Solomon

En la práctica, los errores muchas veces se encuentran juntos, por ejemplo en el caso de mandar una foto mediante una cuadrícula que hemos mencionado antes; si hay un error en un cuadrado, es más probable que se haya producido otro error en un cuadrado adyacente que en otro que se encuentre más lejos de éste. Por eso son interesantes los códigos Reed–Solomon, que consisten en códigos lineales

multicorrectores que se construyen a base de polinomios y pueden corregir errores a ráfagas, es decir corregir errores que aparecen muy juntos.

Para construir un código Reed-Solomon, usamos los siguientes pasos y observaciones:

1. Tomamos como $p(x)$ a un polinomio primitivo de grado m en $\mathbb{Z}_2[x]$, es decir un polinomio irreducible tal que una raíz suya $a = x + p(x)$ genere las unidades de $\mathbb{F} = \mathbb{Z}_2[x]/\langle p(x)\rangle$, o equivalentemente, que genere todos los elementos de $\mathbb{F}$ menos el cero.

2. $\mathbb{F}$ es un $\mathbb{Z}_2$-espacio vectorial con base $1, a, ..., a^{m-1}$.

3. $|\mathbb{F}| = 2^m$, y si $b \in \mathbb{F}$ con $b \neq 0$, $b^n = 1$ siendo $n = 2^m - 1$. Luego, b es raíz de $x^n - 1$. Así, $\mathbb{F} = \{\text{raíces de } x^{2^m} - x = x(x^n - 1)\}$.

4. Como las raíces de $x^n - 1$ son un grupo cíclico (es decir, un grupo que puede ser generado por un solo elemento), $\mathbb{F} = \{0, 1, a, .., a^{n-1}\}$

5. Para cada n-tupla de la forma $(a^{z_1}, a^{z_2}..., a^{z_n}) \in \mathbb{F}^n$, podemos formar tuplas de longitud $m \cdot n$ tomando sus coordenados respecto a la base de $\mathbb{F}$.

6. Existe una aplicación biyectiva de $\mathbb{F}^n$ a $\mathbb{F}[x]/\langle x^n - 1\rangle$.

7. Para $0 \leq 2t < n$, polinomio $g(x) = (x-a)(x-a^2)\cdots(x-a^{2t})$ es divisor de $x^n - 1$ $\big($ a^i es raíz de $x^n - 1$ $\big)$, tenemos que $\langle g(x)\rangle / \langle x^n - 1\rangle$ es $\mathbb{F}$-subespacio vectorial de $\mathbb{F}[x]/\langle x^n - 1\rangle$ de dimensión $n - 2t$.

8. El subespacio correspondiente en $\mathbb{F}^n$ será el código "RS(n,t)" capaz de corregir t-errores de $\mathbb{F}$

Notar que t-errores en $\mathbb{F}$ en realidad pueden ser hasta $m \cdot t$ errores de $\{0,1\}^n$.
Veámoslo con un ejemplo,

Ejemplo 1.7. Tomemos el caso $n = 3$ y vamos a construir un código RS(3,1). En este caso:

$$2^m - 1 = 3 \Leftrightarrow 2^m = 4 \Leftrightarrow m = \log_2 4 = 2$$

Por tanto, tenemos un cuerpo $\mathbb{F}$ de $2^2 = 4$ elementos. Es decir que tenemos que construir un cuerpo de 4 elementos con un polinomio irreducible de grado dos en $\mathbb{Z}_2[x]$. Como el único polinomio irreducible de grado dos en $\mathbb{Z}_2[x]$ es $x^2 + x + 1$, tenemos que $p = 1 + x + x^2$, y por tanto, $F = \{0, 1, a, a^2\}$ donde a es una raíz de p.

Por definición, el polinomio generador del código es $g(x) = (x-a)(x-a^2) = x^2 + (a+a^2)x + a^3 = x^2 + (a+a^2)x + 1 = x^2 + x + 1 = p$, de grado dos. Luego la dimensión del código sobre $\mathbb{F}$ es $n - 2t = 3 - 2 = 1$ y como $\mathbb{F}$ tiene 4 elementos, el código tiene 4 palabras:

$$\big\{(0,0,0), (1,1,1), (a,a,a), (a^2,a^2,a^2)\big\}$$

que corresponden con los polinomios:

$$(0,0,0) \to v_0 = 0$$
$$(1,1,1) \to v_1 = 1 + x + x^2$$
$$(a,a,a) \to v_2 = a + ax + ax^2$$
$$(a^2,a^2,a^2) \to v_3 = a^2 + a^2x + a^2x^2$$

Supongamos ahora que nos llega, por ejemplo, la palabra $(0,1,1,0,0,1)$. Notar que la palabra $(0,1,1,0,0,1)$, la podemos escribir de la siguiente forma:

$$(0,1,1,0,0,1) = (0,1|1,0|0,1) = (a|1|a)$$

debido a que estamos en la base $\{1, a\}$ y $a = (0,1)$ y $1 = (1,0)$ respecto a esta base.

La palabra $(a,1,a)$ no coincide con ninguna de nuestro código. Sin embargo, la podemos decodificar como (a,a,a) ya que es la palabra con la que tiene menos errores. De esta manera hemos corregido la palabra $(a,1,a)$ con un solo error, a la vez que hemos corregido una ráfaga de errores, ya que hemos corregido la palabra $(0,1,1,0,0,1)$ en $(0,1,0,1,0,1)$ (corrigiendo dos errores). Es decir, hemos corregido una ráfaga de errores tratándola como si fuera un solo error.

Capítulo 2

Probabilidad de error

Al usar un código para transmitir información, si se producen más errores de los controlables por el código, nos puede ocurrir que la palabra recibida no la podamos decodificar (en las decodificaciones incompletas) o que la decodifiquemos en una palabra código distinta de la que nos han enviado.

En este capítulo veremos cómo la probabilidad de error sin usar códigos se puede reducir usándolos.

Supongamos que podemos mandar dos símbolos diferentes, 0 y 1, a través de un canal ruidoso con probabilidad $p < \frac{1}{2}$ de que un 0 transmitido sea interpretado por el receptor como un 1 y viceversa. Al final de la transmisión el receptor tendrá una fracción p (la probabilidad de que un símbolo haya sido interpretado incorrectamente) de la información recibida que será incorrecta.

Ahora, si no tenemos el límite de tiempo T especificado anteriormente ni la energía de la que hablábamos en el capítulo anterior, podríamos alcanzar una pequeña probabilidad de error arbitraria mediante un *código de repetición* que consiste en lo siguiente.

1. Sea N impar. (Para que el código sea perfecto)

2. En vez de un 0 transmitimos N ceros. Y lo mismo con el 1.

3. Tenemos un código con dos palabras de longitud N llamadas $\mathbf{0} = (0,0,...,0)$ y $\mathbf{1} = (1,1,...,1)$.

4. El receptor considera la N-tupla recibida y la decodifica como el símbolo que aparece más veces repetidos.

Supondremos siempre que $q = 1 - p$. Como $p < \frac{1}{2}$ la probabilidad de que no haya error en la transmisión de un único bit cumple que $q > p$

Notar que en este caso para que el decodificador cometa un error a la hora de decodificar, tiene que haber más símbolos erróneos que correctos. Es decir, que si hay una palabra código con N símbolos, tiene que haber por lo menos $\frac{N}{2}$ errores para que la palabra sea decodificada incorrectamente. Por tanto, tenemos que en este caso la probabilidad de que el decodificador cometa un error es entonces de:

$$\sum_{0 \leq k \leq N/2} \binom{N}{k} q^k p^{N-k},$$

Además, $q^k p^{N-k} = p^N \left(\frac{q}{p}\right)^k$, como $\left(\frac{q}{p}\right) > 1$ y $k \leq \frac{N}{2}$ $\forall k$,

$$\left(\frac{q}{p}\right)^K \leq \left(\frac{q}{p}\right)^{\frac{N}{2}}$$

Luego,

$$q^k p^{N-k} \leq (pq)^{\frac{N}{2}}$$

También, recordar que como $\binom{N}{k} = \binom{N}{N-k}$,

$$\sum_{0\le k\le N/2} \binom{N}{k} = \frac{1}{2} \sum_{0\le k\le N} \binom{N}{k} = \frac{1}{2} 2^N = 2^{N-1}$$

Luego tenemos que:

$$\sum_{0\le k\le N/2} \binom{N}{k} q^k p^{N-k} \le (pq)^{\frac{N}{2}} \sum_{0\le k<N/2} \binom{N}{k} = 2^{N-1}(pq)^{\frac{N}{2}} = \frac{1}{2}\left(2(pq)^{\frac{1}{2}}\right)^N = \frac{1}{2}C^N$$

Por otra parte, el máximo valor de pq se da cuando $p = q = \frac{1}{2}$. Luego, $pq < \frac{1}{4} \Rightarrow 2(pq)^{\frac{1}{2}} < 1$.

Es decir, que la probabilidad de error en la transmisión de una palabra está acotada por $\frac{1}{2}C^N$ con $C = 2(pq)^{\frac{1}{2}} < 1$, lo cual tiende a 0 cuando $N \to \infty$.

Por ejemplo, si $p = 0,001$ $\big($y $q = 1 - 0,001 = 0,999\big)$,

$$\sum_{0\le k\le N/2} \binom{N}{k} q^k p^{N-k} \le \frac{1}{2}\left(2\big((0,001)(0,999)\big)^{\frac{1}{2}}\right)^N = \frac{1}{2}(0,063)^N < (0,07)^N$$

que tiende a 0 cuando $N \to \infty$.

Si enviamos las dos palabras sin codificar (0 y 1) sin codificar, la tasa es $R = 1$, mientras que si enviamos la información codificada, la tasa es $R = \frac{1}{N}$

Notar que este resultado implica que, teoricamente, si la probabilidad de error de un símbolo es $< \frac{1}{2}$, podemos enviar mensajes arbitrariamente largos con una probabilidad de error $< \varepsilon$, con ε arbitrariamente pequeño. Si el mensaje tiene r símbolos, basta con enviar cada bit repetido N veces con N suficientemente grande.

Siguiendo con $p = 0,001$ y $q = 0,999$, si consideramos el caso de dos lanzamientos de una moneda. Sin codificar (como en el código del ejemplo 1.1), la probabilidad de que dos resultados sean correctamente recibidos es de $q^2 = 0,998$. En el código del ejemplo 1.5 de longitud $n = 4$ donde a la hora de decodificar consideramos que el último símbolo es siempre correcto, esta probabilidad es de $q^4 + 3q^3 p = 0,999$ donde el segundo término de la izquierda $(3q^3 p)$ es la probabilidad de que la palabra recibida contenga un error, pero que éste no se encuentre en la cuarta posición. Así, hemos conseguido una sutil mejora de manera sencilla.

En el código del ejemplo 1.6, con $n = 6$ y donde una palabra error con un solo error es decodificada correctamente y de entre los otras palabras error hay una con dos errores que es decodificada correctamente, la probabilidad de que los tres primeros símbolos (a_1, a_2, a_3 que hemos llamado previamente) sean interpretados correctamente después del procedimiento es $q^6 + 6q^5 p + q^4 p^2 = 0,999986$. Lo cual es una gran mejora respecto a la probabilidad $q^3 = (0,997)$ que obtendríamos sin codificar.

Notar que para construir el código de repetición hemos tomado N impar, ésto lo hacemos para que el código sea perfecto y no tenga dudas de como decodificar una palabra, ya que si tomamos N par, no tiene por qué disminuir la probabilidad de error.

Ejemplo 2.1. Si tomamos el caso $N = 2$, tenemos:

$$\sum_{0\le k\le 1} \binom{2}{k} q^k p^{2-k} = \binom{2}{0} p^2 + \binom{2}{1} qp = p^2 + 2pq$$

Que es igual a $1 - q^2$ $\big($Notar que $p^2 + 2pq = (p^2 + 2pq + q^2) - q^2 = (p+q)^2 - q^2 = 1 - q^2\big)$ lo mismo que hubiéramos obtenido sin decodificación, es decir mandando el resultado de la moneda una vez. En otras palabras, mandando el mismo símbolo dos veces en vez de una no nos disminuye la probabilidad de error (aunque sigue requiriendo más esfuerzo que mandarlo solo una vez).

2.1. Probabilidad de error en la palabra recibida

Sean $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_M}$ las palabras código, utilizando la decodificación de máxima verosimilitud, definimos lo siguiente:

Definición 2.1. Llamamos P_i a la probabilidad de tomar una decisión incorrecta dado que $\mathbf{x_i}$ es transmitido.

En el caso anterior del código de repetición donde teníamos dos palabras de longitud N, **0** y **1**

$$P_{\mathbf{0}} < \frac{1}{2}(0,063)^N \quad y \quad P_{\mathbf{1}} < \frac{1}{2}(0,063)^N$$

Además, si conocemos P_i y el número de palabras del código. Suponiendo que todas la palabras código pueden ser elegidas con igual probabilidad, podemos definir P_C.

Definición 2.2. La probabilidad de una decodificación incorrecta de la palabra recibida viene dada por

$$P_C := M^{-1} \sum_{i=1}^{M} P_i.$$

Notar que si tomamos el caso de códigos de 4-tuplas para tansmitir dos lanzamientos como es el caso del ejemplo 1.2 o el ejemplo 1.5, habrá $\binom{16}{4}$ códigos posibles, cada uno con su respectivo P_C, ya que hay $2^4 = 16$ posibles 4-tuplas y $2^2 = 4$ posibles resultados. En el caso de códigos de 6-tuplas para transmitir tres lanzamientos, como el del ejemplo 1.6, habrá $\binom{64}{8}$ códigos posibles, y por tanto $\binom{64}{8}$ P_C posibles.

Veamos como hallar este P_i y este P_C que acabamos de definir en distintos casos.

2.1.1. Dos lanzamiento de una moneda

Tomemos ahora el ejemplo 1.5, y llamemos $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$ a:

$$cara, cara \to (0,0,0,0) = \mathbf{x_1}$$

$$cara, cruz \to (0,1,1,1) = \mathbf{x_2}$$

$$cruz, cara \to (1,0,0,1) = \mathbf{x_3}$$

$$cruz, cruz \to (1,1,1,0) = \mathbf{x_4}$$

Veamos cuánto es P_i aquí, es decir cual es la probabilidad de tomar una decisión incorrecta dado que $\mathbf{x_i}$ es transmitido.

Recordemos que estamos utilizando una decodificación de máxima verosimilitud, es decir que si recibimos **y**, lo decodificaremos como la palabra código con la que haya menos distancia, es decir que pensaremos siempre que se ha producido el menor número de errores posibles. Además, recordar que en este caso asumimos que el cuarto símbolo es correcto y que por tanto si hay algún error en la 4-tupla éste se encontrará entre los tres primeros símbolos.

Supongamos entonces que se transmite $\mathbf{x_1} = (0,0,0,0)$. A nosotros nos llegará una de las $2^4 = 16$ posibles 4-tuplas (y_1, y_2, y_3, y_4) donde $y_i = \{0,1\}$ para $i = 1,2,3,4$

Como hemos mencionado antes, esta 4-tupla la interpretaremos correctamente cuando no se haya producido ningún error o cuando solo se haya producido un error y éste no se encuentre en la cuarta posición. Y se interpretará erróneamente en todos los demás casos. Por tanto, sean $p = 0,001$ y $q = 1 - p = 0,999$, tenemos que:

$$P_1 = 1 - \left(q^4 + 3q^3p\right) = 1 - \left((0,999)^4 + 3(0,999)^3(0,001)\right) = 0,001003$$

Es decir que la probabilidad de interpretar $\mathbf{x_1}$ como una palabra distinta dado que $\mathbf{x_1}$ ha sido transmitida es de aproximadamente 0,001003.

Repitiendo el mismo proceso para las otras 3 posibles palabras $\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}$, tenemos que:

$$P_2 = 0,001003, \;\; P_3 = 0,001003, \;\; P_4 = 0,001003.$$

Y por tanto tenemos que la probabilidad de decodificación incorrectamente de la palabra recibida es:

$$P_C := M^{-1} \sum_{i=1}^{M} P_i = 4^{-1}\big(P_1 + P_2 + P_3 + P_4\big) = \frac{1}{4} \cdot 4 \cdot 0,001003 = 0,001003$$

En el código anterior, P_i era el mismo para todos los casos, es decir $P_1 = P_2 = P_3 = P_4$. Veamos un código de $n = 4$ que transmita dos lanzamientos de una moneda donde P_i no sea el mismo en todos los casos.

Ejemplo 2.2.

Definamos los resultados de la moneda como sigue:

$$(cara, cara) \to (0,0,0,0) = x_1$$

$$(cara, cruz) \to (0,1,1,1) = x_2$$

$$(cruz, cara) \to (1,1,1,0) = x_3$$

$$(cruz, cruz) \to (1,1,1,1) = x_4$$

Veamos con que palabra tienen menos distancia cada una de las posibles 4-tuplas:
$(0,0,0,0) \to (0,0,0,0) \Rightarrow \mathbf{x_1}$
$(0,0,0,1) \to (0,0,0,0) \Rightarrow \mathbf{x_1}$
$(0,0,1,0) \to (0,0,0,0) \Rightarrow \mathbf{x_1}$
$(0,0,1,1) \to (0,1,1,1) \Rightarrow \mathbf{x_2}$
$(0,1,0,0) \to (0,0,0,0) \Rightarrow \mathbf{x_1}$
$(0,1,0,1) \to (0,1,1,1) \Rightarrow \mathbf{x_2}$
$(0,1,1,0) \to (0,1,1,1)$ ó $(1,1,1,0) \Rightarrow \mathbf{x_2}$ ó $\mathbf{x_3}$
$(0,1,1,1) \to (0,1,1,1) \Rightarrow \mathbf{x_2}$
$(1,0,0,0) \to (0,0,0,0) \Rightarrow \mathbf{x_1}$
$(1,0,0,1) \to (0,0,0,0)$ ó $(1,1,1,1), \Rightarrow \mathbf{x_1}$ ó $\mathbf{x_4}$
$(1,0,1,0) \to (1,1,1,0) \Rightarrow \mathbf{x_3}$
$(1,0,1,1) \to (1,1,1,1) \Rightarrow \mathbf{x_4}$
$(1,1,0,0) \to (1,1,1,0) \Rightarrow \mathbf{x_3}$
$(1,1,0,1) \to (1,1,1,1) \Rightarrow \mathbf{x_4}$
$(1,1,1,0) \to (1,1,1,0) \Rightarrow \mathbf{x_3}$
$(1,1,1,1) \to (1,1,1,1) \Rightarrow \mathbf{x_4}$

Para decodificarlo, utilizaremos el método de decodificación de máxima verosimilitud, es decir que decodificaremos siempre el código recibido como la palabra con menos símbolos distintos. En este caso, en caso de empate elegiremos una de las palabras a menor distancia aleatoriamente con igual probabilidad para decodificar. Es decir que en el caso de que nos llegue $(0,1,1,0)$ habrá $\frac{1}{2}$ de probabilidad de decodificarla como $\mathbf{x_2}$ y $\frac{1}{2}$ de probabilidad de decodificarla como $\mathbf{x_3}$. Y lo mismo con $(1,0,0,1)$, habrá $\frac{1}{2}$ de probabilidad de decodificarla como $\mathbf{x_1}$ y $\frac{1}{2}$ de probabilidad de decodificarla como $\mathbf{x_4}$

Tomando $p = 0,001$ y $q = 0,999$, tenemos que sus respectivos P_i son:

$$P_1' = P\big((0,0,1,1)|\mathbf{x_1}\big) + P\big((0,1,0,1)|\mathbf{x_1}\big) + P\big((0,1,1,0)|\mathbf{x_1}\big) + P\big((0,1,1,1)|\mathbf{x_1}\big) + \frac{1}{2}P\big((1,0,0,1)|\mathbf{x_1}\big) +$$

$$P\big((1,0,1,0)|\mathbf{x_1}\big) + P\big((1,0,1,1)|\mathbf{x_1}\big) + P\big((1,1,0,0)|\mathbf{x_1}\big) + P\big((1,1,0,1)|\mathbf{x_1}\big) + P\big((1,1,1,0)|\mathbf{x_1}\big) + P\big((1,1,1,1)|\mathbf{x_1}\big) =$$

$$\frac{11}{2}p^2q^2 + 4p^3q + p^4 =$$

$$\frac{11}{2}(0,999)^2(0,001)^2+4(0,001)^3(0,999)+(0,001)^4=0,0000055$$

$$P_2'=P\big((0,0,0,0)|\mathbf{x_2}\big)+P\big((0,0,0,1)|\mathbf{x_2}\big)+P\big((0,0,1,0)|\mathbf{x_2}\big)+P\big((0,1,0,0)|\mathbf{x_2}\big)+\frac{1}{2}P\big((0,1,1,0)|\mathbf{x_2}\big)+$$

$$P\big((1,0,0,0)|\mathbf{x_2}\big)+P\big((1,0,0,1)|\mathbf{x_2}\big)+P\big((1,0,1,0)|\mathbf{x_2}\big)+P\big((1,0,1,1)|\mathbf{x_2}\big)+P\big((1,1,0,0)|\mathbf{x_2}\big)+$$

$$P\big((1,1,0,1)|\mathbf{x_2}\big)+P\big((1,1,1,0)|\mathbf{x_2}\big)+P\big((1,1,1,1)|\mathbf{x_2}\big)=$$

$$p^4+4p^3q+6p^2q^2+\frac{3}{2}pq^3=$$

$$(0,001)^4+4(0,001)^3(0,999)+6(0,001)^2(0,999)^2+\frac{3}{2}(0,001)(0,999)^3=0,0015$$

$$P_3'=P\big((0,0,0,0)|\mathbf{x_3}\big)+P\big((0,0,0,1)|\mathbf{x_3}\big)+P\big((0,0,1,0)|\mathbf{x_3}\big)+P\big((0,0,1,1)|\mathbf{x_3}\big)+P\big((0,1,0,0)|\mathbf{x_3}\big)+$$

$$P\big((0,1,0,1)|\mathbf{x_3}\big)+\frac{1}{2}P\big((0,1,1,0)|\mathbf{x_3}\big)+P\big((0,1,1,1)|\mathbf{x_3}\big)+P\big((1,0,0,0)|\mathbf{x_3}\big)+P\big((1,0,0,1)|\mathbf{x_3}\big)+$$

$$P\big((1,0,1,1)|\mathbf{x_3}\big)+P\big((1,1,0,1)|\mathbf{x_3}\big)+P\big((1,1,1,1)|\mathbf{x_3}\big)=$$

$$p^4+4p^3q+6p^2q^2+\frac{3}{2}pq^3=$$

$$(0,001)^4+4(0,001)^3(0,999)+6(0,001)^2(0,999)^2+\frac{3}{2}(0,001)(0,999)^3=0,0015$$

$$P_4'=P\big((0,0,0,0)|\mathbf{x_4}\big)+P\big((0,0,0,1)|\mathbf{x_4}\big)+P\big((0,0,1,0)|\mathbf{x_4}\big)+P\big((0,0,1,1)|\mathbf{x_4}\big)+P\big((0,1,0,0)|\mathbf{x_4}\big)+$$

$$P\big((0,1,0,1)|\mathbf{x_4}\big)+P\big((0,1,1,0)|\mathbf{x_4}\big)+P\big((0,1,1,1)|\mathbf{x_4}\big)+P\big((1,0,0,0)|\mathbf{x_4}\big)+\frac{1}{2}P\big((1,0,0,1)|\mathbf{x_4}\big)+$$

$$P\big((1,0,1,0)|\mathbf{x_4}\big)+P\big((1,1,0,0)|\mathbf{x_4}\big)+P\big((1,1,1,0)|\mathbf{x_4}\big)=$$

$$=p^4+4p^3q+\frac{11}{2})p^2q^2+2pq^3=$$

$$(0,001)^4+4(0,001)^3(0,999)+\frac{11}{2}(0,001)^2(0,999)^2+2(0,001)(0,999)^3=0,002$$

Y por lo tanto,

$$P_C'=4^{-1}\left[P_1'+P_2'+P_3'+P_4'\right]=\frac{1}{4}[0,0000055+0,0015+0,0015+0,002]=0,00125$$

Que es mayor que el P_C que habíamos calculado para el código del ejemplo 1.5, es decir:

$$P_C<P_C'$$

Notar, que a pesar de que el P_C' obtenido para este código es mayor que el P_C del código anterior, sigue siendo menor que la probabilidad de error que hubiéramos obtenido sin codificación (donde habríamos obtenido $P_C''=1-q^2=0,001999>0,00125$).

2.1.2. Hamming(7,4)

En el caso del código $Hamming(7,4)$ que hemos descrito en el ejemplo 1.4 habíamos dicho que todas las palabras se encuentran a distancia mínimo tres, por tanto dado que se envía una palabra $\mathbf{x_i}$, la probabilidad de que esta se reciba erróneamente es la probabilidad de que se hayan producido dos o más errores en la transmisión.

$$P_1 = P(\textit{por lo menos dos errores}|\mathbf{x_1}) = 1 - P(1 \textit{ o menos errores}|\mathbf{x_1}) =$$
$$1 - \big(P(0 \textit{ errores}|\mathbf{x_1}) + P(1 \textit{ error}|\mathbf{x_1})\big) = 1 - (q^7 + 7pq^6)$$

Como esto ocurre con todas las palabras del código, tenemos que $P_i = 1 - (q^7 + 7pq^6)$ para i=1,...,16. Por tanto tenemos que

$$P_C = \frac{1}{16}\sum_{i=1}^{16} P_i = 1 - (q^7 + 7pq^6) = 0,00002093$$

2.1.3. Hamming(3,1)

En el caso general de los códigos de Hamming hemos mencionado que son códigos perfectos cuyas palabras se encuentran como mínimo a distancia tres. Por tanto, este es también el caso del código $Hamming(3,1)$ compuesto por las palabras $\mathbf{x_1} = (0,0,0)$ y $\mathbf{x_2} = (1,1,1)$. Como $d(C) = 3$, se producirá un error de decodificación cuando se produzcan dos o más errores, es decir:

$$P_i = 3p^2q + p^3 = 3 \cdot (0,001)^2(0,999) + (0,001)^3 = 0,000002998 \qquad \text{para } i = 1,2$$

$$\Rightarrow P_C = M^{-1}\sum_{i=1}^{M} P_i = 0,000002998$$

2.1.4. Reed-Muller(32,5)

Al principio, hemos mencionado que el satélite Mariner 1969, utilizaba un código Reed-Muller(32,5) que consistía en 64 palabras de longitud de 32, donde todas las palabras se distanciaban por lo menos en 16 símbolos. Es decir que para que se produzca una decodificación errónea de una palabra, tiene que haber por lo menos 8 errores. Por tanto,

$$P_i = P(\textit{por lo menos } 16 \textit{ errores}|\mathbf{x_i}) = 1 - \big(P(0 \textit{ errores}|\mathbf{x_i}) + (P(1 \textit{ error}|\mathbf{x_i}) + ... + (P(16 \textit{ errores}|\mathbf{x_i})\big) =$$

$$1 - \left[\binom{32}{0}q^{32} + \binom{32}{1}q^{31}p + ... + \binom{32}{7}q^7p^7\right] = 1 - \sum_{k=0}^{7}\binom{32}{k}q^{32-k}p^k \qquad \text{para } i = 1,..,64$$

$$\Rightarrow P_C = 64^{-1}\sum_{i=1}^{64} P_i = P_i = 1 - \sum_{k=0}^{7}\binom{32}{k}q^{32-k}p^k$$

2.2. Teorema de Shannon

Tomando P_C como en la definición 2.2 y R como en la definición 1.1. Si ahora consideramos todos los posibles códigos C con M palabras de longitud n, con p probabilidad de error y definimos P^* como

$$P^*(M,n,p) := \text{mínimo valor de } P_C$$

Teorema 2.1. *(Shannon, 1948).*

$$\textit{Si } \ 0 < R < 1 + p\log p + q\log q \ \textit{ y } \ M_n := 2^{\lfloor R\cdot n\rfloor} \ \textit{ entonces } \ P^*(M_n,n,p) \to 0 \ \textit{ cuando } \ n \to \infty.$$

O equivalentemente, $\forall \varepsilon > 0$ existe n suficientemenete grande tal que $P_C^* < \varepsilon$. Es decir, existe un código C con $|C| = M_n$ tal que $P_C < \varepsilon$

El teorema de Shannon nos dice que existe un código para cualquier (M,n,p) con la probabilidad de error arbitrariamente pequeña, es decir que es posible transmitir cualquier información con una probabilidad de error casi nula. Sin embargo, a pesar de decirnos que existen tales códigos, el teorema no nos dice cual son, ni como encontrarlos, solamente que existen.

Shannon también demostró que si $R > 1 + p\log p + q\log q$, no se puede alcanzar una probabilidad arbitrariamente pequeña de error, siempre hay una probabilidad mínima de error, y ese error mínimo crece cuando R crece.

Remarcamos que en el ejemplo anterior de los lanzamientos de la moneda donde tomamos $p = 0{,}001$, es decir que tenemos $1 + p\log p + q\log q = 0{,}98859$ que es casi 1, donde 0.98859 representa la proporción entre tuplas necesarias para transmitir el mensaje y n-tuplas extra para redundancia. Si n es por ejemplo 100.000, queremos que $\frac{r}{100{,}000} = 0{,}98859$ con r tal que $M_n = 2^r$. El requisito en el experimento era que la tasa debía de ser al menos $\frac{1}{2}$. Es decir que para $\varepsilon > 0$ y n suficientemente grande hay un código C de longitud n, con tasa casi 1 y tal que $P_C < \varepsilon$.

Capítulo 3

Demostración del teorema de Shannon

La demostración del teorema no es constructiva, es decir no vamos a dar la forma de como construir un código que cumpla el teorema. En su lugar, para demostrar el teorema veremos que $\forall\varepsilon$, existe n suficientemente grande tal que existe un C con $|C| = M_n$ y $P_C < \varepsilon$. Para ello, consideraremos todos los códigos posibles C de M palabras de longitud n,

Además, encontraremos un ρ adecuado para el cual se cumpla que

$$P(w > \rho) < \frac{\varepsilon}{2}$$

donde w es el número de errores. Y para el cual

$$(M-1)2^{-n}|B_\rho| < \frac{\varepsilon}{2}$$

donde B_ρ es bola de radio ρ dada por:

$$B_\rho(\mathbf{x_i}) = \{\mathbf{y}|d(\mathbf{y},\mathbf{x_i}) \leq \rho\}$$

De esta manera, considerando los P_C como una variable aleatoria, halláremos su esperanza y veremos que ésta es menor que epsilón ($\mathbb{E}(P_C) < \varepsilon$). Así, utilizando que P^* se trata del mínimo de los P_C, y por tanto es menor que la esperanza de los P_C podremos ver que para todo epsilón, existe un n suficientemente grande tal que $P^* < \varepsilon$

Antes de ver la demostración del teorema, recordemos algunos conceptos básicos de estadística que necesitaremos para la demostración de nuestro teorema.

3.1. Conceptos básicos de estadística

Definición 3.1. La esperanza de una variable aleatoria X viene dada por

$$\mu = \mathbb{E}[X] = \sum_i P(X = x_i)x_i$$

Proposición 3.1. Sea x_i^* el mínimo de los x_i, entonces

$$x_i^* \leq \mu$$

Demostración.

$$\mu = \sum_i P(X = x_i)x_i \geq \sum_i P(X = x_i)x_i^* = x_i^* \sum_i P(X = x_i) = x_i^*$$

□

Definición 3.2. La varianza de una variable aleatoria X viene dada por

$$\sigma^2 = Var(X) = \sum_i P(X = x_i)x_i^2 - \mu^2$$

Si μ es la esperanza de X, también podemos definir su varianza como

$$\sigma^2 = \mathbb{E}\left[(X-\mu)^2\right]$$

Proposición 3.2. El número de errores en una palabra recibida es una variable aleatoria con esperanza np y varianza $np(1-p)$.

Demostración.

$$\mu = \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} k = \sum_{k=1}^{n} \binom{n}{k} p^k q^{n-k} k = \sum_{k=1}^{n} \frac{n\cdot(n-1)\cdots(n-k+1)}{k!} k p^k q^{n-k} =$$

$$\sum_{k=1}^{n} n\frac{(n-1)\cdots(n-k+1)}{(k-1)!} p^k q^{n-k} = \sum_{k=1}^{n} n\binom{n-1}{k-1} p^k q^{n-k} = n\cdot p \sum_{k=1}^{n} \binom{n-1}{k-1} p^{k-1} q^{(n-1)-(k-1)} =$$

Haciendo $j = k-1$

$$\mu = n\cdot p \sum_{j=1}^{n-1} \binom{n-1}{j} p^j q^{(n-1)-j} = n\cdot p\cdot (p+q)^{n-1} = n\cdot p\cdot (1)^{n-1} = np$$

$$\sigma^2 = \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} k^2 - \mu^2 = \sum_{k=1}^{n} kn\binom{n-1}{k-1} p^k q^{n-k} - (np)^2 =$$

$$np \sum_{k=1}^{n} k\binom{n-1}{k-1} p^{k-1} q^{(n-1)-(k-1)} - (np)^2 =$$

Tomando $j = k-1$ y $m = n-1$

$$\sigma^2 = np \sum_{j=0}^{m} (j+1)\binom{m}{j} p^j q^{m-j} - (np)^2 = np\left(\sum_{j=0}^{m} j\binom{m}{j} p^j q^{m-j} + \sum_{j=0}^{m} \binom{m}{j} p^j q^{m-j}\right) - (np)^2 =$$

$$np\left(\sum_{j=0}^{m} m\binom{m-1}{j-1} p^j q^{m-j} + \sum_{j=0}^{m} \binom{m}{j} p^j q^{m-j}\right) - (np)^2 =$$

$$np\left((n-1)p\sum_{j=1}^{m} \binom{m-1}{j-1} p^{j-1} q^{(m-1)-(j-1)} + \sum_{j=0}^{m} \binom{m}{j} p^j q^{m-j}\right) - (np)^2 =$$

$$np\left((n-1)p(p+q)^{m-1} + (p+q)^m\right) - n^2p^2 =$$

$$np\left((n-1)p+1\right) - n^2p^2 = n^2p^2 + np(1-p) - n^2p^2 = np(1-p)$$

□

Teorema 3.1. *(Desigualdad de Chebychev).*
Sea X una variable aleatoria de media μ y varianza finita σ^2, entonces, para todo número real $a > 0$,

$$P(|X-\mu| > a\sigma) \leq \frac{1}{a^2}$$

Demostración. Sea Y una variable aleatoria tal que

$$Y = \begin{cases} 0 \;\; si \;\; |X-\mu| \leq a\sigma \\ 1 \;\; si \;\; |X-\mu| > a\sigma \end{cases}$$

entonces tenemos

$$a\sigma Y \leq |X-\mu| \Rightarrow a^2\sigma^2 Y^2 \leq (X-\mu)^2$$

Tomando esperanzas a ambos lados

$$a^2\sigma^2\mathbb{E}[Y^2] \leq \mathbb{E}[(X-\mu)]^2 = \sigma^2 \Rightarrow \mathbb{E}[Y^2] \leq \frac{1}{a^2}$$

A su vez, como Y solo puede ser 0 ó 1,

$$\frac{1}{a^2} \geq \mathbb{E}[Y^2] = P(Y=1) = P(|X-\mu| > a\sigma) \Rightarrow P(|X-\mu| > a\sigma) \leq \frac{1}{a^2}$$

□

En nuestro caso, tomando como variable aleatoria al número de errores, y llamandolo W tenemos que nuestra esperanza es $\mathbb{E}[w] = \mu = np$ y nuestra varianza es $Var[w] = \sigma^2 = np(1-p)$, por tanto

$$P\left(|w-np| > a\big(np(1-p)\big)^{\frac{1}{2}}\right) \leq \frac{1}{a^2}$$

Tomemos a partir de aquí $a = \left(\frac{1}{\frac{\varepsilon}{2}}\right)^{\frac{1}{2}}$ para que $\frac{1}{a^2} = \frac{\varepsilon}{2}$

Proposición 3.3. Para cada ε y n suficientemente grande, existe $\rho < \frac{1}{2}n$ tal que

$$P(w > \rho) < \frac{\varepsilon}{2}$$

Demostración. Sea $b := \left(\frac{np(1-p)}{\frac{\varepsilon}{2}}\right)^{\frac{1}{2}}$, de manera que

$$a\sigma = (np(1-p))^{\frac{1}{2}}\left(\frac{1}{\frac{\varepsilon}{2}}\right)^{\frac{1}{2}} = \left(\frac{np(1-p)}{\frac{\varepsilon}{2}}\right)^{\frac{1}{2}} = b.$$

y definamos ρ como el número $\rho := \lfloor np+b \rfloor$

Aplicando la desigualdad de Chebychev, tenemos:

$$P(w > \rho) = P(w > np+b) = P(w > np+a\sigma) = P(w-np > a\sigma) \leq P(|w-np| > a\sigma) \leq \frac{1}{a^2} = \frac{1}{2}\varepsilon$$

Veamos como de grande tiene que ser n para que exista este ρ

Notar que $p < \frac{1}{2}$ es una desigualdad estricta y queremos que $p + \frac{b}{n} \leq \frac{1}{2}$ por tanto

$$\frac{b}{n} \leq \left(\frac{1}{2}-p\right) \iff \left(\frac{np(1-p)}{\frac{\varepsilon}{2}}\right)^{\frac{1}{2}} \leq \left(\frac{1}{2}-p\right)n \iff \frac{p(1-p)}{\frac{\varepsilon}{2}} \leq n\left(\frac{1}{2}-p\right)^2 \iff n \geq \frac{p(1-p)}{\frac{\varepsilon}{2}\left(\frac{1}{2}-p\right)^2}.$$

Es decir que si n es mayor o igual que $\frac{p(1-p)}{\frac{\varepsilon}{2}\left(\frac{1}{2}-p\right)^2}$ entonces $np+b \leq \frac{1}{2}n$ y por tanto $\lfloor np+b \rfloor < \frac{1}{2}n$ □

Por ejemplo tomemos el caso de una n-tupla con $p = 0{,}001$ busquemos cuanto tiene que ser n para que

$$P(w > np+b) \leq \frac{1}{2} \cdot 0{,}0000001$$

Es decir para $\varepsilon = 0{,}0000001$,

$$n \geq \frac{p(1-p)}{\frac{\varepsilon}{2}\left(\frac{1}{2}-p\right)^2} = \frac{0{,}001 \cdot 0{,}999}{\frac{0{,}0000001}{2}\left(\frac{1}{2}-0{,}001\right)^2} = 80240{,}64$$

Es decir que para que ρ sea menor o igual que $\frac{1}{2}n$, n tiene que ser mayor o igual que 80241. Supongamos que $n = 80241$, veamos que $\rho > \frac{1}{2}n$

$$\rho = \left\lfloor np + \left(\frac{np(1-p)}{\frac{\varepsilon}{2}}\right)^{\frac{1}{2}} \right\rfloor = \left\lfloor 80241 \cdot 0{,}001 + \left(\frac{80241 \cdot 0{,}001(0{,}999)}{\frac{0{,}0000001}{2}}\right)^{\frac{1}{2}} \right\rfloor = 40120 \leq \frac{1}{2}80241$$

Por último, antes de entrar en la demostración del Teorema de Shannon veamos algunos detalles que nos serán útiles para demostrar el teorema

Lema 3.2. *Sea $B_\rho(\mathbf{x})$ el conjunto de palabras $\mathbf{y}$ con $d(\mathbf{x},\mathbf{y}) \leq \rho$. Entonces,*

$$|B_\rho(\mathbf{x})| < \frac{1}{2}n \cdot \frac{n^n}{\rho^\rho (n-\rho)^{n-\rho}}$$

Demostración.

$$|B_\rho(\mathbf{x})| = \sum_{i \leq \rho} \binom{n}{i} < \frac{1}{2}n\binom{n}{\rho} \leq \frac{1}{2}n \cdot \frac{n^n}{\rho^\rho (n-\rho)^{n-\rho}}$$

Donde la primera desigualdad es debido a que, como $\rho < \frac{n}{2}$,

$$\sum_{i \leq \rho} \binom{n}{i} \leq \rho\binom{n}{\rho} < \frac{1}{2}n\binom{n}{\rho}$$

Y la última desigualdad viene de

$$n^n = \big[m + (n-m)\big]^n = \sum_{k=0}^{n} \binom{n}{k} m^k (n-m)^{n-k} \geq \binom{n}{m} m^m (n-m)^{n-m} \Leftrightarrow \binom{n}{m} \leq \frac{n^n}{m^m (n-m)^{n-m}}$$

□

Por último, si $\rho = \lfloor np+b \rfloor$. Como

$$b = \left(\frac{np(1-p)}{\frac{\varepsilon}{2}}\right) = C \cdot \left(n^{\frac{1}{2}}\right),$$

$$b \text{ es } O\left(n^{\frac{1}{2}}\right)$$

y como,

$$n + b - 1 \leq \lfloor np+b \rfloor \leq np+b,$$

tenemos que

$$p + O\left(n^{-\frac{1}{2}}\right) = \frac{np+b-1}{n} \leq \frac{\lfloor np+b \rfloor}{n} \leq \frac{np+b}{n} = p + O\left(n^{-\frac{1}{2}}\right)$$

Y por lo tanto podemos realizar las siguientes aproximaciones:

$$\frac{\rho}{n}\log\frac{\rho}{n} = \frac{1}{n}\lfloor np+b \rfloor \log\left(\frac{\lfloor np+b \rfloor}{n}\right) = p\log p + O\left(n^{-\frac{1}{2}}\right),$$

$$\left(1-\frac{\rho}{n}\right)\log\left(1-\frac{\rho}{n}\right)=\left(1-\frac{\lfloor np+b\rfloor}{n}\right)\log\left(1-\frac{\lfloor np+b\rfloor}{n}\right)=(1-p)\log(1-p)=q\log q+O\left(n^{-\frac{1}{2}}\right)$$

cuando $n\to\infty$

Donde $O\left(n^{-\frac{1}{2}}\right)$ denota al conjunto de funciones $f(n)$ tales que $f(n)\leq cn^{-\frac{1}{2}}$ para $n>n_0$ para cierto n_0, con $c>0$ una constante. Y que por tanto tienden a cero cuando $n\to\infty$

Notar que $O\left(n^{-\frac{1}{2}}\right)$ proviene de que

$$\frac{b}{n}=\left(\frac{\frac{np(1-p)}{\frac{\varepsilon}{2}}}{n^2}\right)^{\frac{1}{2}}=\left(\frac{\frac{p(1-p)}{\frac{\varepsilon}{2}}}{n}\right)^{\frac{1}{2}}$$

tiene orden de $n^{-\frac{1}{2}}$.

Lema 3.3. *Para cada ε y n suficientemente grande, tomando M como en el apartado 2.1 $M=2^{R\cdot n}$ se tiene que*

$$(M-1)2^{-n}|B_\rho|<\frac{\varepsilon}{2}$$

Demostración. Sea $A=(M-1)2^{-n}|B_\rho|$.

$$A=(M-1)2^{-n}|B_\rho|<M\cdot 2^{-n}|B_\rho|$$

Por el lema 3.2,

$$M\cdot 2^{-n}|B_\rho|<M\cdot 2^{-n}\frac{1}{2}n\cdot\frac{n^n}{\rho^\rho(n-\rho)^{n-\rho}}=M\cdot 2^{-n}\frac{1}{2}n\frac{n^\rho n^{n-\rho}}{\rho^\rho(n-\rho)^{n-\rho}}$$

Tomando logaritmos y diviendo por n, tenemos

$$n^{-1}\log A<n^{-1}\log\left(M\cdot 2^{-n}\frac{1}{2}n\frac{n^\rho n^{n-\rho}}{\rho^\rho(n-\rho)^{n-\rho}}\right)=$$

$$n^{-1}\log M+n^{-1}\log 2^{-n}+n^{-1}\log\frac{1}{2}n+n^{-1}\log\left(\frac{n}{\rho}\right)^\rho+n^{-1}\log\left(\frac{n}{n-\rho}\right)^{n-\rho}=$$

$$n^{-1}\log M-1+n^{-1}\log\frac{1}{2}n+\frac{\rho}{n}\log\left(\frac{n}{\rho}\right)+\left(1-\frac{\rho}{n}\right)\log\left(1-\frac{\rho}{n}\right)=$$

donde por las aproximaciones de antes y debido a que $n^{-1}\log\left(\frac{1}{2}n\right)\in O\left(n^{-\frac{1}{2}}\right)$, tenemos que es igual a

$$n^{-1}\log M-1+p\log p+q\log q+O\left(n^{-\frac{1}{2}}\right)$$

Luego,

$$n^{-1}\log A<n^{-1}\log M-\left(1-p\log p-q\log q\right)+O\left(n^{-\frac{1}{2}}\right)$$

Sustituyendo $M=M_n$ y utilizando la restricción sobre R que utilizamos en el enunciado del Teorema 2.1 $\left(0<R<1+p\log p+q\log q\right.$, y por tanto $\left.-(1+p\log p+q\log q)<-R\right)$ tenemos que para n suficientemente grande, existe un $\beta>0$ tal que

$$n^{-1}A<n^{-1}\log M_n-(1+p\log p+q\log q)+O(n^{-1/2})<n^{-1}\lfloor R\cdot n\rfloor-(1+p\log p+q\log q)+O\left(n^{-1/2}\right)=-\beta<0,$$

Por tanto,

$$n^{-1}\log(P^*(M_n,n,p))<-\beta\iff$$
$$\log(P^*(M_n,n,p))<-\beta\cdot n\iff$$
$$P^*(M_n,n,p)<2^{-\beta\cdot n}$$

Que como tiende a 0, cuando $\to\infty$, podemos decir que es menor que $\frac{\varepsilon}{2}$ para n suficientemente grande. $\square$

Finalmente introducimos dos funciones que usaremos en la demostración. Sean $\mathbf{u} \in \{0,1\}^n, \mathbf{v} \in \{0,1\}^n$. Entonces, definimos

$$f(\mathbf{u},\mathbf{v}) := \begin{cases} 0, \text{ si } d(\mathbf{u},\mathbf{v}) > \rho, \\ 1, \text{ si } d(\mathbf{u},\mathbf{v}) \leq \rho. \end{cases}$$

Si $\mathbf{x_i} \in C$ e $\mathbf{y} \in \{0,1\}^n$, entonces definimos

$$g_i(\mathbf{y}) := 1 - f(\mathbf{y},\mathbf{x_i}) + \sum_{j \neq i} f(\mathbf{y},\mathbf{x_i}).$$

Notar que si $\mathbf{x_i}$ es la única palabra código tal que $d(\mathbf{x_i},\mathbf{y}) \leq \rho$, entonces $g_i(\mathbf{y}) = 1 - 1 + (M-1)\cdot 0 = 0$ y si no, $g_i(\mathbf{y}) \geq 1$.

Tomemos el ejemplo 1.5 y tomemos $\varepsilon = 0,001$, en este caso

$\rho = \lfloor np + b \rfloor = \left\lfloor np + \left(\frac{np(1-p)}{\frac{\varepsilon}{2}}\right)^{\frac{1}{2}} \right\rfloor = \left\lfloor 4 \cdot 0,001 + \left(\frac{4 \cdot 0,001(1-0,001)}{\frac{0,001}{2}}\right)^{\frac{1}{2}} \right\rfloor = \left\lfloor 0,004 + \left(\frac{0,004(0,999)}{\frac{0,01}{2}}\right)^{\frac{1}{2}} \right\rfloor =$
$\lfloor 0,004 + 2,827 \rfloor = \lfloor 2,831 \rfloor = 2$

Supongamos que hemos recibido $\mathbf{y} = (1,0,0,0)$, de manera que
$d(\mathbf{y},\mathbf{x_1}) = d\big((1,0,0,0),(0,0,0,0)\big) = 1 \leq \rho \Rightarrow f(\mathbf{y},\mathbf{x_1}) = 1$
$d(\mathbf{y},\mathbf{x_2}) = d\big((1,0,0,0),(0,1,1,1)\big) = 4 > \rho \Rightarrow f(\mathbf{y},\mathbf{x_2}) = 0$
$d(\mathbf{y},\mathbf{x_3}) = d\big((1,0,0,0),(1,0,0,1)\big) = 1 \leq \rho \Rightarrow f(\mathbf{y},\mathbf{x_3}) = 1$
$d(\mathbf{y},\mathbf{x_4}) = d\big((1,0,0,0),(1,1,1,0)\big) = 2 \leq \rho \Rightarrow f(\mathbf{y},\mathbf{x_4}) = 1$
Y por tanto
$g_1(\mathbf{y}) = 1 - f(\mathbf{y},\mathbf{x_1}) + \sum_{j \neq i} f(\mathbf{y},\mathbf{x_1}) = 1 - 1 + (0+1+1) = 2$
$g_2(\mathbf{y}) = 1 - f(\mathbf{y},\mathbf{x_2}) + \sum_{j \neq i} f(\mathbf{y},\mathbf{x_2}) = 1 - 0 + (1+1+1) = 4$
$g_3(\mathbf{y}) = 1 - f(\mathbf{y},\mathbf{x_3}) + \sum_{j \neq i} f(\mathbf{y},\mathbf{x_3}) = 1 - 1 + (1+0+1) = 2$
$g_4(\mathbf{y}) = 1 - f(\mathbf{y},\mathbf{x_4}) + \sum_{j \neq i} f(\mathbf{y},\mathbf{x_4}) = 1 - 1 + (1+0+1) = 2$

3.2. Demostración del teorema de Shannon

Como hemos mencionado al principio de este cpítulo, la demostración del teorema de Shannon no es una demostración constructiva, es decir que el teorema no se basa a partir de un ejemplo dando el código que satisface el teorema de Shannon, sino que solamente prueba que tal código existe. Para demostrarlo realizamos lo siguiente.

Sea $M = M_n = 2^{\lfloor Rn \rfloor}$ y n suficientemente grande para que se cumplan la proposición 3.3 y el lema 3.3 y sea ε cualquiera. Elegimos las palabras código $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_M}$ aleatoriamemte y las decodificamos de la siguiente manera. Si $\mathbf{y}$ es recibida y solamete hay una palabra código $\mathbf{x_i}$ tal que $d(\mathbf{x_i},\mathbf{y}) \leq \rho$ entonces decodificamos $\mathbf{y}$ como $\mathbf{x_i}$. En otro caso declaramos un error (o si debemos decodificar sí o sí, la decodificamos siempre como $\mathbf{x_1}$).

Sea P_i definida como en la definición 2.1. Es decir como la probabilidad de decodificar erróneamente dado que se ha transmitido $\mathbf{x_i}$.

Y recordemos que $g_i(\mathbf{y}) = 0 \iff \mathbf{y}$ es la única palabra tal que $d(\mathbf{x_i},\mathbf{y}) \leq \rho$, es decir que según lo visto anteriormente si $g_i(\mathbf{y}) = 0$, $\mathbf{y}$ será decodificado como $\mathbf{x_i}$. Por tanto:

$$P_i \leq \sum_{\mathbf{y} \in \{0,1\}^n} P(\mathbf{y}|\mathbf{x_i}) g_i(\mathbf{y}) = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x_i}) \big\{1 - f(\mathbf{y},\mathbf{x_i})\big\} + \sum_{\mathbf{y}} \sum_{j \neq i} P(\mathbf{y}|\mathbf{x_i}) f(\mathbf{y},\mathbf{x_j})$$

Como $f(\mathbf{y},\mathbf{x_i}) := \begin{cases} 0, \text{ si } d(\mathbf{y},\mathbf{x_i}) > \rho, \\ 1, \text{ si } d(\mathbf{y},\mathbf{x_i}) \leq \rho. \end{cases}$,

$$\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x_i}) \big\{1 - f(\mathbf{y},\mathbf{x_i})\big\} = \sum_{\mathbf{y} \notin B_\rho} P(\mathbf{y}|\mathbf{x_i}) = P\big(\mathbf{y} \notin B_\rho(\mathbf{x_i})\big) = P(w > np + b)$$

Donde w es el número de errores. Como hemos visto en la proposición 3.3 $\left(P(w > np+b) \leq \frac{1}{2}\varepsilon\right)$ esta probabilidad es como mucho $\frac{1}{2}\varepsilon$. Por tanto, tenemos

$$P_C = M^{-1}\sum_{i=1}^{M} P_i \leq M^{-1}\sum_{i=1}^{M}\left[\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x_i})\{1-f(\mathbf{y},\mathbf{x_i})\} + \sum_{\mathbf{y}}\sum_{j\neq i} P(\mathbf{y}|\mathbf{x_i})f(\mathbf{y},\mathbf{x_j})\right] \leq$$

$$M^{-1}\sum_{i=1}^{M}\frac{1}{2}\varepsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i} P(\mathbf{y}|\mathbf{x_i})f(\mathbf{y},\mathbf{x_j}) \leq \frac{1}{2}\varepsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i} P(\mathbf{y}|\mathbf{x_i})f(\mathbf{y},\mathbf{x_j})$$

Para continuar la demostración hay que tener en cuenta que $P^*(M,n,p)$ por la proposición 3.2 es menor que el valor esperado de P_C sobre todos los posibles códigos C elegidos aleatoriamente.

$$P^*(M,n,p) \leq \mathbb{E}[P_C]$$

Por tanto, considerando todos los posibles códigos de M palabras de longitud n tenemos

$$P^*(M,n,p) \leq \mathbb{E}\left[\frac{1}{2}\varepsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i} P(\mathbf{y}|\mathbf{x_i})f(\mathbf{y},\mathbf{x_j})\right] = \mathbb{E}\left[\frac{1}{2}\varepsilon\right] + \mathbb{E}\left[M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i} P(\mathbf{y}|\mathbf{x_i})f(\mathbf{y},\mathbf{x_j})\right] \overset{P(\mathbf{y}|\mathbf{x_i}) \text{ y } f(\mathbf{y},\mathbf{x_j}) \text{ independientes}}{=}$$

$$\frac{1}{2}\varepsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i}\mathbb{E}\left[(P(\mathbf{y}|\mathbf{x_i})\right]\mathbb{E}\left[f(\mathbf{y},\mathbf{x_j})\right] \overset{(1)}{=} \frac{1}{2}\varepsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}(M-1)\mathbb{E}\left[P(\mathbf{y}|\mathbf{x_i})\right]\frac{|B_\rho|}{2^n} \overset{(2)}{=}$$

$$\frac{1}{2}\varepsilon + M^{-1}(M-1)2^{-n}|B_\rho|\sum_{i=1}^{M} 1 = \frac{1}{2}\varepsilon + M^{-1}(M-1)2^{-n}|B_\rho|M = \frac{1}{2}\varepsilon + (M-1)2^{-n}|B_\rho|$$

Debido a que

$$(1) \quad \mathbb{E}\left[f(\mathbf{y},\mathbf{x_j})\right] = 0\cdot P\big(d(\mathbf{y},\mathbf{x_j}) > \rho\big) + 1\cdot P\big(d(\mathbf{y},\mathbf{x_j}) \leq \rho\big) = P\big(\mathbf{x_j} \in B_\rho(\mathbf{y})\big) = \frac{|B_\rho|}{2^n}$$

$$(2) \quad \sum_{\mathbf{y}}\mathbb{E}\left[(P(\mathbf{y}|\mathbf{x_i})\right] = \mathbb{E}\left[\sum_{\mathbf{y}}(P(\mathbf{y}|\mathbf{x_i})\right] = 1$$

Utilizando el lema 3.3, tenemos que

$$P^*(M_n,n,p) < \frac{1}{2}\varepsilon + (M-1)2^{-n}|B_\rho| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

Luego, cuando $n \to \infty$, tenemos que P^* es arbitrariamente pequeño, o dicho de otra forma $P^* \to 0$ cuando $n \to \infty$ como queríamos demostrar. □

Bibliografía

[1] VAN LINT, J. H., *Introduction to Coding Theory*, Springer-Verlag, Eindhoven University of Technology, Eindhoven, 1982.

[2] XAMBÓ-DESCAMPS, SEBASTIÀ, *Block Error-Correcting Codes*. Universitat Politecnica de Catalunya, Barcelona, 2003.

[3] ALLENBY, R. B. J. T., *Rings, Fields and Groups*. Springer-Verlag, Great Britain, 1991.

[4] POSNER, E. C., *Combinatorial structures in planetary reconnaissance. In: Error Correcting Codes*. H. B. Mann, New York-London-Sydney-Toronto, 1968.

[5] SHANNON, C. E., *A mathematical theory of communication*. Bell System Technical Journal, **27** (1948).

[6] HAMMING, RICHARD WESLEY, *Error detecting and error correcting codes*. Bell System Technical Journal, **29** (1950).

[7] VAN TILBORG, H. *Weights in the Third-Order Reed-Muller Codes*. Communications Systems Research Section, **9** (2009).

[8] SANTOS, CAMILLE, *Shannon's Coding Theorems*. Saint Mary's College of California, 2016.

[9] M. COVER, THOMAS Y A. THOMAS, JOY, *Elements of Information Theory*. John Wiley Sons, Hoboken, New Jersey, 2006.

[10] ROHATGI, VIJAY K. Y EHSANES SALEH, A. K. MD., *An introduction to probability and statistics*. John Wiley Sons, Hoboken, New Jersey, 2015.

[11] KLIMA, RICHARD E., SIGMON, NEIL Y STITZINGER, ERNEST, *Applications of Abstract Algebra with MAPLE*. CRC, Florida, United States of America, 2000.