Introducción práctica a la Teoría de códigos

Roberto Maestre Martínez

4 de septiembre de 2008

1. Introducción

Teoría de códigos [1] es una especialidad matemática que trata de las leyes de la codificación de la información. A grandes rasgos, codificar es transformar una información en una señal convenida para su comunicación. Decodificar sería el proceso inverso y complementario del anterior por el cual la señal comunicada es transformada en la información original. El auge de las comunicaciones a partir de la segunda mitad del siglo XX motivó un fuerte desarrollo de la teoría de códigos.

Puesto que los códigos se usan para comunicar información, uno de los problemas a los que todo código se enfrenta es el error sistemático y, también, el fortuito. La redundancia es el unico medio de prevenir el error. Los lenguajes humanos tienen una gran redundancia que les da flexibilidad a costa, eso sí, de eficacia. Los códigos matemáticos utilizan una redundancia más racional.

Hay códigos llamados de detección de errores que permiten detectar alteraciones en un mensaje codificado. Se utilizan sobre todo en entornos donde el mensaje puede ser reenviado tantas veces como se necesite. Los protocolos de Internet, por ejemplo, están formados por un anidamiento de codificaciones desde el nivel de transporte hasta el nivel físico, teniendo cada nivel su propio sistema de detección de errores.

Este tipo de códigos resulta inadecuado en entornos donde la comunicación no se puede repetir y se necesita asegurar hasta cierto punto que se va a recibir la información correcta. Un ejemplo típico y vistoso es cuando se envía una nave espacial a los confines del sistema solar y desde allí debe enviar una serie de fotografías antes de que se le acaben, digamos, las pilas. Se trata de una situación delicada, porque si las ondas electromagnéticas que portan la información llegan distorsionadas toda la misión fracasa. Un código que sólo detectase que la información es incorrecta no serviría para nada. Es necesario algo más, un código no sólo detector sino corrector de errores.

2. Definiciones

2.1. Matriz para un código lineal sistemático

$$\begin{pmatrix} I_n & | & m \end{pmatrix}$$

Partiremos de un codigo (n=8,k=3) sistemático, para ello utilizaremos la matriz **C** :

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

n ->dimension de los vectores de la matriz C ; *k ->rango de la matriz C*
Nuestro codigo lineal binario podrá generar 2^k palabras, que son: *000 , 001 , 010 , 011 , 100 , 101 , 110 , 111*
Como ejemplo, una matriz (4,2) tendría asociado un código lineal sistemático

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

En este caso, el codigo podría generar 2^k palabras, que son: *00 , 01 , 10 , 11*

2.2. Matriz de control

CS = La matriz C sin la matriz identidad que contiene
Se define de la siguiente manera:

$$H = CS'|I$$

En nuestro caso:

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

2.3. Alfabeto fuente y, distancia del código y líderes

2.4. Alfabeto fuente

Lo definimos como alffuente(k) (la 'k' de nuestro dódigo).

$$alffuente(k = 1) = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

$$alffuente(k = 2) = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$$alffuente(k=3) = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$alffuente(k=4) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Para obtener el alfabeto fuente de k hacemos 2^k, de aquí obtenemos la longitud del vector, a partir del cual iremos entremezclando por las sucesivas k columnas
Es decir; si alffuente(3), sabemos que $2^3 = 8$ entonces [0,0,0,0,1,1,1,1] , y a partir de este vector obtenemos el resto

2.5. Palabras del código

Para obtener las palabras del código, basta con realizar la siguiente operación: $C' * alfuente(k)$,la matriz resultante contendrá todas las palabras de nuestro código.

En nuestro caso:

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

Donde cada columna es cada una de las palabras del código.
En nuestro caso [00000000,00110110,01010011,01100101,10011001,10101111,11001010,11111100]

2.6. Distancia del código y sus líderes

Peso de una palabra es el numero de 1 que tiene.
Los **líderes** de nuestro código son las palabras, distintas de 0, del código que tienen menor peso.
En nuestro caso, son:

peso([00110110])=4
peso([01010011])=4
peso([01100101])=4
peso([10011001])=4
peso([11001010])=4

Como el lider de las palabras de nuestro código tiene peso 4, **la distancia del código será 4** .
A partir de este dato obtenemos la capacidad de detección y correccion de errores:

DETECCIÓN $=> 4-2=2$ errores
CORRECIÓN $=> parteentera((4-2)/2) = 1$ error

3. Codificación, decodificación y reparación

3.1. Codificación. Función codificadora

La definición de la funcion codificadora es $f : \mathbb{Z}_2^3 \rightarrow \mathbb{Z}_3^8$ de la cual obtenemos la palabra codificada y lista para transmitir a partir de la palabra primitiva *'a'* es:

$$f(a) = C' * a$$

Así, f(1,0,1)=[1,0,1,0,1,1,1,1]

Tenemos que observar dos partes en el vector codificado y listo para transmitir; la palabra y la parte redundante que el código introduce para detectar y corregir errores en la transmisión:

[1,0,1,**0,1,1,1,1**]

3.2. Decodificación y reparación. Función síndrome

Con la función síndrome comprobaremos si ha habido errores en la transmisión, y si es así, intentaremos obtener la palabra primitiva.
La definición de la funcion codificadora es $f : \mathbb{Z}_2^8 \rightarrow \mathbb{Z}_3^5$ de la cual obtenemos el síndrome a través de la palabra codificada *'v'* es:

$$f(v) = H' * v$$

Así, f(1,0,1,0,1,1,1,1)=[0,0,0,0,0], y como el vector sindrome es [00000], entonces sabemos que la palabra se ha recibido correctamente siempre que no se hayan alterado mas de 2 (Recordar que este código solo puede detectar 2 errores y corregir 1 error).

Para decodifcar la palabra recibida tenemos dos opciones que vemos a contuniación

3.2.1. Decodificación por distancia mínima

La distancia mínima entre dos palabras es el numero de elementos que tienen diferentes según la posicion.

Por ejemplo la distancia entre [1,0,1] y [1,0,0] es = 1, ya que, según su posición, solo se diferencian en el ultimo componente
Asi que este método consiste en comparar la palabra recibida con todas las del código y elegir respecto a cual tiene distancia mínima, en este caso ya que no ha habido errores la palabra está entre las palabras del código tal cual.

Introducciendo un error
Introduciremos un error el el código [1,**1**,1,0,1,1,1,1], mirando enter las palabras del código obtenemos [1,0,1,0,1,1,1,1] que esta a distancia 1 de la palabra recibida, asi que obtendremos la palabra original, aún cuando se ha producido un error en la transmision.

Introducciendo dos errores
Si ahora introducimos 2 errores en el código [1,**1**,1,0,1,1,1,**0**] vamos a poder observar como a través de la función sñindrome podremos detectar un error pero sera imposible reparlo, debido a que nuestro código detectaba dos fallos, pero solo corregía uno.
Ahora nos encontramos que tenemos mas de una palabra a distancia mínima: [00000000 , 01010011 , 10011001]; en este caso no podremos descodificar correctamente la apalabra, pero si detectar que se ha producido un error.

Teniendo en cuenta las consideraciones mencionadas anteriormente, el número de operaciones que hay que realizar para descodificar la palabra recibida es $n|C| = np^k$; por ejemplo, en un (20,10) codigo lineal se realizarían 20.480 operaciones en cada decodificación

3.2.2. Decodificación por síndrome

Suponemos que tenemos la tabla de síndromes construida; es decir, cada posible palabra del código con su síndrome asociado, en nuestro caso: Para construir la tabla de síndromes, calcularemos el par [síndrome][lider de la orbita de la palabra]
(Para calcular la órbita de una palabra codificada, tenemos que sumr dicha palabara a todas las palabras del código (matriz 'palabrasdelcod') y elegir el lider de su órbita)

Algunos ejemplos para la tabla de síndromes de nuestro

[00000][00000000]
[10011][00110111]
[10010][00110110]

Para decodificar una palabra por el método de síndrome, tenemos que obtener su síndrome, y mirar en la tabla de síndromes cual es el lider que le corresponde, despues debemos sumar la palabra al lider de la tabla y truncar la redundancia para obtener la palabra decodificada.

Entonces si obtenemos la palabra codificada y=11101111 y obtenemos su síndrome a través de la matriz de control s(y)=[10011]. Miramos cual es el lider de la orbita que corresponde a este síndrome en la tabla de síndromes y obtenemos l(u)=[00110111], ahora solamente debemos calcular y+l(u) = 11101111 + 00110111 = [11011000], si truncamos los tres primeros bits, obtenemos la palabra del alfabeto fuente [110] decodificada correctamente.

Podemos observar la eficiencia de este método, ya que en un (20,10) código lineal, realizaremos 420 operaciones por cada decodificación, sin embargo, tenemos que realizar una inversion inicial para calcular la tabla de sídrome, teniendo asi que realizar 419.430.400 operaciones paracalcular la tabla de síndromes.

Referencias

[1] Wikipedia, Teoría de códigos.

[2] Teoría de códigos. Escuela Universitaria de Informática.

[3] BURGOS,.: "Álgebra lineal"Mc Graw Hill 3º Edición 2005, 1993.