Códigos lineales de bloque

Guido Sanchez

May 2021

1. Introducción

1.1. La matemática de los códigos binarios

Las matemáticas de la codificación pueden resultar bastante complicadas si se estudian todas las clases de códigos. Al restringirnos, por el momento, a códigos simples, podemos emplear matemáticas simples para familiarizarnos con el tema antes de intentar los códigos más difíciles. Como resultado, este capítulo no requerirá nada más difícil que la comprensión de la representación matricial de ecuaciones y la aplicación de funciones lógicas simples.

La razón principal por la que las matemáticas de la codificación pueden parecer complicadas es que necesitamos poder realizar operaciones aritméticas en lo que se llama un ***campo finito***. Cualquier código consta de varios símbolos que pueden tomar solo ciertos valores, siendo el ejemplo más simple de un símbolo un bit que solo puede tomar dos valores, aunque se pueden idear otros símbolos con más niveles. Es necesario definir nuestras operaciones aritméticas de manera que solo se puedan producir valores de símbolo válidos. Un ***campo finito*** es un conjunto definido de valores a los que se agregan dos operaciones definidas y sus inversas. Estas operaciones sólo pueden producir valores dentro del conjunto.

Las operaciones que deben realizarse para producir códigos lineales definidos sobre un conjunto o alfabeto finito de valores se denominan suma y multiplicación, y sus inversas pueden considerarse como resta y división. Sin embargo, **las operaciones en sí mismas no corresponderán a nuestra comprensión normal de esos términos**. Para todos los códigos, la definición de una aritmética apropiada es necesaria antes de poder explicar la codificación y decodificación. Para los códigos no binarios, estas definiciones no son sencillas.

Afortunadamente, solo hay una familia importante de códigos no binarios, a saber, los códigos de Reed Solomon, aunque para una comprensión adecuada de algunos otros códigos es valioso un enfoque de campos finitos. Sin embargo, podemos recorrer un largo camino tratando solo con campos binarios, para los cuales la aritmética apropiada es simplemente módulo-2:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 1 = 0$$

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 1 = 1$$

La inversa de la suma (resta) es equivalente a la suma, la división por cero no está permitida y la división por 1 es equivalente a la multiplicación por 1. Nuestra aritmética de campo finito será, por lo tanto, bastante fácil, y la única cuestión a destacar es que la suma módulo-2 se corresponde con la función OR exclusiva de la lógica booleana.

Información	Código
000	0001
001	0010
010	0100
011	0111
100	1000
101	1011
110	1101
111	1110

Tabla 1: Código de paridad impar

Información	Código
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

Tabla 2: Código de paridad par

1.2. Chequeos de paridad

Para obtener una idea de cómo se puede producir un código lineal, tomemos un ejemplo simple en el que se produce una palabra de código a partir de la información. Los bits de información se insertan directamente en la palabra de código y luego se agrega un solo bit calculado a partir de todos los bits de información. Consideraremos dos métodos para calcular este bit final:

1. El bit final se establece de manera que la suma módulo-2 de todos los bits de la palabra de código sea 1.

2. El bit final se establece de manera que la suma módulo 2 de todos los bits de la palabra de código sea 0.

En el primer caso, se dice que la palabra de código tiene una paridad impar, es decir, hay un número impar de unos en la palabra de código. En el segundo caso, hay un número par de unos en la palabra palabra de código que, por tanto, tiene paridad par. El bit adicional se denomina *bit de verificación de paridad* y puede denominarse verificación de paridad impar o par, según corresponda. Los códigos de paridad pares e impares se muestran en las Tablas 1 y 2, respectivamente, para el caso en el que hay tres bits de información.

Se puede observar que el código de la Tabla 1 no contiene la secuencia de todos ceros, la cual **siempre** debe ser parte de un código lineal. Así, la verificación de paridad impar produce un código no lineal. Por otro lado, el código de la Tabla 2 es lineal[1]; los sistemas que generan bits de paridad par a partir de algunos o todos los bits de información dan como resultado un código lineal. Tenga en cuenta que en este caso el bit de verificación de paridad es la suma módulo-2 de los bits a partir de los cuales se calcula. Por lo tanto, la paridad para la secuencia de información 101 es la suma módulo-2 de 1, 0 y 1, es decir, 0.

[1] ¿Por qué? ¡Verificar!

2. Distancia de Hamming

La distancia de Hamming se denomina así gracias a su inventor Richard Hamming, profesor de la Universidad de Nebraska, que fue el que introdujo el término para establecer una métrica capaz de establecer un código para la detección y auto-corrección de códigos. Se emplea en la transmisión de información digitalizada para contar el número de bits distintos en cadenas de igual longitud y estimar el error, por esto se denomina a veces como distancia de una señal. Si u, v y w son palabras de código, la distancia de Hamming tiene las siguientes propiedades.

$$\begin{aligned} d(u,v) &= d(v,u) \\ d(u,v) &= 0 \text{ si y solo si } u = v \\ d(u,v) + d(v,w) &\geq d(u,w) \end{aligned}$$

Donde la distancia d se calcula como el número de bits p en que son diferentes dos palabras de código. A la vez, hay algunas relaciones importantes con respecto al valor de la distancia d, tales como:

1. Si $d \geq p + 1$ entonces se puede detectar un error de p bits.
2. Si $d \geq 2p + 1$ entonces se pueden corregir p bits.

Por ejemplo, si queremos crear un código que pueda detectar 3 errores, entonces la distancia mínima de Hamming debe ser de $3 + 1 = 4$. Si queremos corregir 3 errores entonces la distancia mínima de Hamming debe ser de $2 \times 3 + 1 = 7$.

3. Códigos de bloque lineal

Los códigos de bloque lineal son códigos que codifican sus datos en bloques y además tienen la propiedad de la linealidad, es decir, la suma de dos palabras de código cualesquiera es también una palabra de código, y dado que las operaciones se realizan sobre los bits de origen en bloques, de ahí el nombre códigos de bloque lineal. Hay códigos de bloque que no son lineales, pero es difícil demostrar que un código es bueno sin esta propiedad.

3.1. Notación popular

Los códigos de bloques lineales se pueden caracterizar por sus alfabetos de símbolos (por ejemplo, binarios o ternarios) y los parámetros (n, m, d_{min}) donde

- n es el largo de la palabra de código, en símbolos,
- m es el número de símbolos de origen que se utilizarán para codificar a la vez,
- d_{min} es la distancia mínima de Hamming para el código.

3.2. Distancia mínima de Hamming de un código de bloque lineal

Una consecuencia de la linealidad es que la estructura de distancia del código es la misma independientemente de la palabra de código desde la que se calcule (ver Capítulo 1 de "Error Control Coding: From Theory to Practice"). Si u, v y w son palabras de código y $d(u, v)$ significa la operación distancia entre u y v, entonces

$$d(u,v) = d(u + w, v + w)$$

Las secuencias $u + w$ y $v + w$ son palabras de código, por lo que la relación entre u y v se repite en otros puntos del código. En particular, podemos suponer que $w = v$ para obtener

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

Información	Código
00	00000
01	01011
10	10101
11	11110

Tabla 3: Código (5,2).

Por lo tanto, podemos decir que la distancia entre cualquier par de palabras de código es la misma que la distancia entre alguna palabra de código y la secuencia de todos ceros. Por tanto, podemos llegar a la siguiente conclusión:

> *La **distancia mínima** de un código de bloque lineal es igual al número mínimo de símbolos distintos de cero que aparecen en cualquier palabra de código (excluyendo la palabra de código todo cero).*

El número de símbolos distintos de cero en una secuencia se denomina peso de la secuencia, por lo que la distancia mínima de un código de bloque lineal es igual al peso de la palabra de código de peso mínimo.

4. Ejemplos

4.1. Código (5,2)

La Tabla 3 nos muestra un código lineal formado a partir de dos bits de información y una codificación que se ha calculado de alguna manera a partir de los bits de información. El código código (5,2) tiene las siguientes propiedades:

1. Es un código lineal, ya que la suma de dos palabras cualesquiera del código pertenece al código.
2. El largo de la palabra de código $n = 5$,
3. El número de símbolos de origen utilizados $m = 2$
4. La distancia mínima de Hamming $d_{min} = 3$. Dado que es un código lineal, se obtiene contando la distancia mínima de cualquier palabra de código con la de la palabra 00000. Además dado que $d_{min} \geq 2 \times 1 + 1$, entonces este código puede detectar y corregir errores en 1 bit.

4.2. Código Hamming (7,4)

La Tabla 4 nos muestra un código Hamming (7,4), donde se agregan 3 bits de redundancia a los 4 bits de información. De esta forma, el código tiene las siguientes propiedades:

1. Es un código lineal, ya que la suma de dos palabras cualesquiera del código pertenece al código.
2. El largo de la palabra de código $n = 7$,
3. El número de símbolos de origen utilizados $m = 4$
4. La distancia mínima de Hamming $d_{min} = 3$. Dado que es un código lineal, se obtiene contando la distancia mínima de cualquier palabra de código con la de la palabra 0000000. Además dado que $d_{min} \geq 3$, entonces este código puede detectar y corregir errores en 1 bit.

Información	Código
0000	0000000
0001	0001011
0010	0010111
0011	0011100
0100	0100110
0101	0101101
0110	0110001
0111	0111010
1000	1000101
1001	1001110
1010	1010010
1011	1011001
1100	1100011
1101	1101000
1110	1110100
1111	1111111

Tabla 4: Código Hamming (7,4).

4.3. Código binario de Golay

El código binario de Golay[2] es un tipo de código corrector de errores usado en las comunicaciones digitales. Se ha utilizado en misiones de la NASA en el espacio, por ejemplo, cuando los satélites Voyager 1 y 2 necesitaban transmitir cientos de fotografías en color (que ocupaban 3 veces la cantidad de datos que las fotos blanco y negro) de Júpiter y Saturno entre 1979 y 1980 con un ancho de banda de comunicaciones muy bajo. Para ello se utilizó el código de Golay(24,12,8), que si bien solo corregía 3 errores, podía transmitir datos a mucha más velocidad que el código Reed-Muller, el cual corregía 7 errores y se utilizaba para transmitir fotos blanco y negro.

5. Referencias

1. Peter Sweeney, "Error Control Coding: From Theory to Practice", ISBN: 978-0-470-84356-7.
2. Coding theory: `https://en.wikipedia.org/wiki/Coding_theory`
3. Distancia de Hamming: `https://es.wikipedia.org/wiki/Distancia_de_Hamming`
4. Block codes – the (7,4) Hamming code: `http://www.inference.org.uk/mackay/itprnn/1997/l1/node7.html`
5. An ANSI C implementation of Hamming codes: `https://github.com/michaeldipperstein/hamming`

[2] `https://en.wikipedia.org/wiki/Binary_Golay_code`