SUMA 51
Febrero 2006, pp. 9-13

Las matrices, la base 2 y la tecnología digital

En este artículo presentamos, mediante dos sencillos juegos de adivinación de números en unas tablas, cómo la base 2 y las matrices son fundamentales en la tecnología digital que nos rodea en el mundo civilizado que habitamos. Además aportamos la implementación en Mathematica del algoritmo que permite realizar esta acción.

In this paper we give, by using two simple guess games of numbers in two tables, how basis 2 and matrices are fundamental in digital technology that rounds us in the civilized world that we habit. We also give an implementation using Mathematica of the algorithm that allows to develop this action.

El mundo civilizado actual es conocido como el mundo de la tecnología digital. Todo se basa en aparatos digitales, aparatos sin los cuales no sabríamos vivir o al menos, nuestra vida no sería tan cómoda. Pero, ¿conoce la gente que el funcionamiento de todos ellos se basa en matemáticas muy sencillas? En concreto en dos aspectos: la expresión en base 2 y el producto de matrices. Del primero deriva precisamente el término digital; el segundo, junto con el primero, proporciona métodos que nos permiten confiar en la tecnología digital. Nuestro objetivo en este artículo es proponer una actividad muy sencilla que puede llevarse a cabo en el aula con los alumnos, proponiendo un enfoque distinto de esta parte del álgebra lineal, como Baena (1994) al usar la criptografía para introducir o hablar de los métodos matriciales, al tiempo que haga comprender la importancia de las matrices en la Teoría de Códigos y las grandes aplicaciones que esto tiene en nuestro entorno.

La teoría de la codificación trata del estudio de la transmisión de datos a través de diversos medios y de la detección y corrección de los errores producidos durante dicha transmisión.

Primeramente, proponemos un pequeño juego que permitirá la comprensión de la importancia de la expresión en base 2 en nuestro mundo. En concreto, el juego consiste en la adivinación de un número escogido de una tabla por otra persona a partir de algunas pistas que se piden a ésta. Seguidamente, modificaremos el juego para ver la importancia del producto de matrices dentro de esta tecnología. Más precisamente, cambiaremos la tabla y haremos la misma operación, es decir, pediremos a una persona que nos dé pistas sobre un número elegido de esta segunda tabla, con la salvedad de que en esta ocasión están permitidas las mentiras.

La teoría de la codificación trata del estudio de la transmisión de datos a través de diversos medios y de la detección y corrección de los errores producidos durante dicha transmisión.

Ilustraremos estos juegos con algunos ejemplos. Finalmente, dado que el uso del ordenador es un recurso muy utilizado ya en la enseñanza de las matemáticas y que algunos, como nos-

Juan Antonio López Ramos
Universidad de Almería. Almería.
Justo Peralta López
Universidad de Almería. Almería.

otros, creemos imprescindible a partir de un cierto nivel, presentamos una implementación en Mathematica de una máquina que actúe como adivinadora de números, haciendo comprender de este modo la posibilidad de trasladar las matemáticas al mundo real, en concreto, a aparatos o programas que funcionan en los mismos, al tiempo que estos métodos constituyen la base del mundo actual. El uso de Mathematica se ha propuesto como recurso para la experimentación con álgebra lineal (González, 1998), además de que ya existe numerosa bibliografía al respecto, como por ejemplo *Alemany y otros, 2002*.

La base 2

Consideremos la siguiente tabla de números del 1 al 31:

1	2	4	8	16
3	3	5	9	17
5	6	6	10	18
7	7	7	11	19
9	10	12	12	20
11	11	13	13	21
13	14	14	14	22
15	15	15	15	23
17	18	20	24	24
19	19	21	25	25
21	22	22	26	26
23	23	23	27	27
25	26	28	28	28
27	27	29	29	29
29	30	30	30	30
31	31	31	31	31

Figura 1

Pedimos ahora a alguien que elija un número de los que aparecen en la tabla y, sin decirnos de qué número se trata, nos indique todas las columnas en las que se encuentra. Si ahora sumamos cada uno de los números que aparecen en la parte superior de las columnas indicadas, obtendremos el número elegido por el otro. ¿Magia? No, matemáticas, o más precisamente un simple cambio a base 2. La otra persona nos ha indicado el número mediante sus dígitos en base dos: uno para las columnas en las que se encontraba el número y cero para las que no. Así, para recuperar el número pensado basta multiplicar por cero o uno los números de la parte superior de las columnas y sumar los resultados. Si nos fijamos, la tabla no es ni más ni menos que una codificación de los números del 1 al 31, es decir, cada número aparece en aquellas columnas para las cuales, en su expresión en binario, el dígito correspondiente es un uno. Por ejemplo, el 3 expresado en base 2 es 11 y así, aparece en la primera y en la segunda; el 6 expresado en base 2 es 110, con lo que aparece en la segunda y en la tercera; las potencias de dos aparecerán obviamente en una sola columna, y así, con el resto de números.

Esta es la forma en la que todos los aparatos de la era digital, ordenadores, teléfonos móviles, televisión digital, etc. representan la información. Sonidos o imágenes, números o palabras son representados, transmitidos y recibidos como largas cadenas de ceros y unos. El término digital deriva precisamente del uso de los dígitos cero y uno. Incluso podríamos detallar aún más: 1 si por un circuito concreto pasa electricidad y 0 en caso contrario.

Por ejemplo, en el caso de los ordenadores, cada carácter (ya sea imprimible o no, como los saltos de línea) se representa por un número comprendido entre el 0 y el 255, conocido como código ASCII. Así, la oración *Las matemáticas son muy útiles*, tiene un equivalente en dicho código ASCII a la sucesión de números

76, 97, 115, 32, 109, 97, 116, 101, 109, 225, 116, 105, 99, 97, 115, 32, 115, 111, 110, 32, 109, 117, 121, 32, 250, 116, 105, 108, 101, 115

y que expresada en base 2 es el conjunto de números:

{01001100, 01100001, 01110011, 00100000, 011001101, 01100001, 01110100, 01100101, 01101101, 11100001, 01110100, 01101001, 01100011, 01100001, 01110011, 00100000, 01110011, 01101111, 01101110, 00100000, 01101101, 01110101, 01111001, 00100000, 11111010, 01110100, 01101001, 01101100, 01100101, 01110011}.

Si ahora unimos todos los elementos del conjunto obtenemos la cadena

0100110001100001011100110010000001100110101100001011101000110010101101101111000010111010001101001011000011011000010111001100100000011100110110111011011100001000000110110101110101011110010010000011111010011101000110100101101100011001010111001 1

que es lo que en realidad *entiende* el aparato digital en cuestión. Pero, ¿qué pasa si cambiamos un cero por un uno o viceversa?

Matrices y tecnología digital

Este intercambio de unos por ceros o viceversa puede deberse a una interferencia en la comunicación, a una caída de tensión, etc. La gran ventaja de establecer la comunicación en forma digital es que se han desarrollado métodos matemáticos que permiten detectar y corregir cuándo, al transmitir la información, se cambian ceros por unos o viceversa, es decir, se detectan y corrigen los errores.

Hagamos ahora una modificación en nuestro juego y veamos que en realidad esto es así.

El proceso de codificación de un número natural entre 0 y 15 de forma que podamos detectar errores dobles o mentiras dobles, y corregir errores simples o mentiras simples consiste en convertir dicho número en un vector binario de longitud 8. Para el alumno consiste en colocar un 1 en la posición i-ésima del vector si el número escogido se encuentra en la columna número *i* o un cero en caso contrario. La tabla utilizada es la siguiente:

8	4	2	1	1	1	1	2
9	5	3	3	2	2	3	3
10	6	6	5	4	5	4	4
10	7	7	7	7	6	6	5
12	12	10	9	9	8	8	8
13	13	11	11	10	11	10	9
14	14	14	13	12	12	13	14
15	15	15	15	15	15	15	15

Figura 2

Por ejemplo, la codificación del número 13 será el vector 11010010 ya que aparece en las columnas 1, 2, 4 y 7, mientras que la codificación del número 15 será 11111111 ya que aparece en todas las columnas. Obsérvese que el resultado de codificar cualquier entero entre 0 y 15 es una palabra código. Nuestro objetivo ahora es determinar cuándo uno o dos bits de dicha palabra cambian de 0 a 1 o de 1 a 0. Esto se corresponde en el juego a cuando el alumno ha mentido al indicar en qué columnas aparece el número en el cual previamente ha pensado.

Como podemos observar, esta tabla contiene sólo los números del 1 al 15, pero ello se debe únicamente al intento de evitar usar una tabla demasiado grande, y aún así, observamos que la posición que ocupan los números es muy distinta y que existen más columnas, pero si lo que queremos ahora es detectar y corregir los errores, hemos de proporcionar al receptor algo más de información denominada información de redundancia, es decir, la codificación de la información se ha llevado a cabo de forma distinta, aunque se sigue usando la base 2 (ceros y unos) tanto para la información base, como para la redundancia. El juego ahora consiste en lo mismo: pedimos a una segunda persona que elija un número de la tabla y le pedimos la misma información relativa a las columnas en las que aparece el número. Sin embargo ahora permitimos el hecho de que nos mienta (si quiere) como mucho dos veces, es decir, puede decirnos que el número se encuentra en una columna cuando esto no es así o que no se encuentra en una columna cuando en realidad el número sí que está en esa columna. Para llevar a cabo nuestra experiencia hemos considerado un código basado en un código Hamming (Hamming, 1950) capaz de detectar dos errores y corregir uno cometido durante la transmisión en un canal con ruido. El mecanismo de corrección de errores consiste en multiplicar un vector binario por una matriz *H*. En nuestro juego, utilizaremos la matriz siguiente

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	1	1	1	1	1	1

Figura 3

Aunque no es objeto de nuestra experiencia con los alumnos construir la matriz *H*, explicaremos brevemente su obtención y su relación con la nueva tabla de números considerada: un código Hamming binario, capaz de corregir un solo error, con longitud de palabra $2r$-1, tiene asociada una matriz *D*, conocida como matriz de paridad o matriz de decodificación, formada por todas las columnas binarias no nulas posibles. Ahora bien, podemos modificar esta matriz *D*, con r filas y $2r$-1 columnas, obteniendo una nueva matriz de paridad, de forma que nuestro código sea capaz de corregir un error y detectar dos, tan sólo añadiendo una fila con $2r$-1 unos y una columna de ceros, excepto el último elemento que es un uno. En nuestro caso hemos considerado la matriz *D* con r=3 y la hemos modificado del modo anterior. La segunda parte es saber cómo codificar la información, es decir, construir la tabla de la figura 2. Para ello hemos calculado lo que se conoce como matriz generadora del código, es decir, una matriz *G* tal que el producto de *G* por la matriz transpuesta de nuestra matriz H sea 0. Entonces codificamos los bits de información dados por el vector $u=(ur,..., u1)$, que es la representación binaria de los números, en nuestro caso, del 1 al 15, haciendo *uG*, obteniendo así un vector binario de longitud $2r$, que es lo que se representa en la tabla de la figura 2. Por ejemplo, el número 14 aparece en las tres primeras columnas de la tabla, ya que su vector de información en binario es u=(1, 1, 1, 0) y aparece en la última columna puesto que al hacer *uG*, el resultado es el vector (1, 1, 1, 0, 0, 0, 0, 1) (observemos que las cuatro primeras posiciones de este vector se corresponden con *u*, es decir, la información base y las otras cuatro constituyen la información de redundancia necesaria para nuestros fines).

Para un mayor detalle en el estudio de los códigos de Hamming podemos consultar, por ejemplo, el capítulo 8 del libro de Hill (1997).

La matriz de paridad o decodificación *H* la podemos definir con Mathematica mediante la siguiente asignación:

```
In[1]:=H={{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,
1,1,1,1,1,1,1}};
```

El proceso de multiplicar un vector binario o palabra del código por la transpuesta de la matriz de paridad *H* es lo que se llama en Teoría de Códigos cálculo del síndrome, el cual será, en nuestro caso, un vector fila de longitud 4. Para calcular dicho síndrome definimos la siguiente función con Mathematica:

```
In[2]:=Sindrome[a_,H_]:=Mod[a.Transpose[H],2];
```

Una vez calculado el síndrome, la decodificación se realiza como sigue:

1. Si el síndrome es 0, es decir, la columna cero, entonces no se ha producido ningún error. Lo cual nos indica en el juego que el alumno no ha mentido.

2. Si el síndrome es distinto de 0, quiere decir que se ha producido algún error o que el alumno ha mentido. Entonces podemos distinguir varios casos.

   2.1. Si la última posición del síndrome es 0, entonces se ha producido más de un error, en concreto, un número par de errores (probablemente 2), y el alumno ha mentido más de una vez.

   2.2. Si la última posición del síndrome es 1, entonces se ha producido sólo un error y el alumno ha mentido sólo una vez. Si los tres primeros bits del vector síndrome son ceros, entonces el error se ha producido en la última posición de la palabra código y el alumno ha mentido en la última columna. Y si los tres primeros bits son distintos de cero, entonces estos nos indica en binario o base dos en qué posición de la palabra se ha producido el error o en qué columna ha mentido el alumno.

Una vez que sabemos con exactitud en qué posición se ha cometido el error o en qué columna ha mentido el alumno podemos adivinar el número en que había pensado.

Se llama cálculo del síndrome al proceso de multiplicar un vector binario o palabra del código por la transpuesta de la matriz de paridad H.

Empieza el Juego

Supongamos que el alumno ha pensado en el número 15 y no ha mentido al indicar en qué columnas aparece. En este caso la codificación será 11111111 y se traducirá con la siguiente asignación en Mathematica:

```
In[3]:= palabra={1,1,1,1,1,1,1,1};
```

Ahora calculamos el síndrome:

```
In[4]:= Sindrome[palabra,H]

Out[4]:= {0,0,0,0}
```

De donde tenemos, según lo anterior, que el alumno no ha mentido y por tanto, la decodificación es 15, que es la información que proporcionan los cuatro primeros bits de la palabra.

Si el alumno miente en la última columna la codificación de 15 sería 11111110. Veamos cómo se realiza la codificación.

```
In[5]:=palabra={1,1,1,1,1,1,1,0};

In[6]:=Sindrome[palabra,H]

Out[6]:= {0,0,0,1}
```

Como la última posición del síndrome es 1, esto quiere decir que se ha producido un error y como además, las tres primeras son ceros, tenemos que dicho error se ha producido en la última columna, es decir, la decodificación sería 11111111, o lo que es lo mismo, de nuevo 15.

Si el alumno miente en la tercera columna, 15 quedaría codificado por 11011111. Veamos qué nos indica la decodificación:

```
In[7]:=palabra={1,1,0,1,1,1,1,1};

In[8]:=Sindrome[palabra,H]

Out[8]:= {0,1,1,1}
```

Como la última posición del síndrome es 1, esto quiere decir que se ha producido un error. Además, las tres primeras entradas del vector nos indican la posición en la que se ha producido: la 011, en binario, 3. Por tanto, la decodificación sería 11111111, con lo que, considerando una vez más los cuatro primeros bits, los de información, tenemos de nuevo 15.

Si el alumno miente en la tercera y última columna entonces 15 será codificado por 11011110. Veamos a continuación su decodificación.

```
In[9]:=palabra={1,1,0,1,1,1,1,0};

In[10]:=Sindrome[palabra,H]

Out[10]:= {0,1,1,0}
```

Como la última posición del síndrome es 0, esto quiere decir que el alumno ha mentido más de una vez, aunque ahora no sabemos dónde.

Veamos qué sucede si el alumno miente 3 veces y codificamos 15 por 10101011.

In[11]:=palabra={1,0,1,0,1,0,1,1};

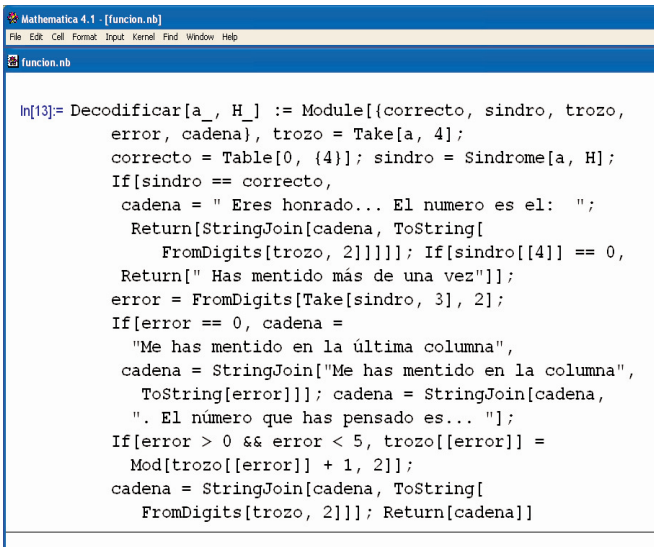
In[12]:=Sindrome[palabra,H]

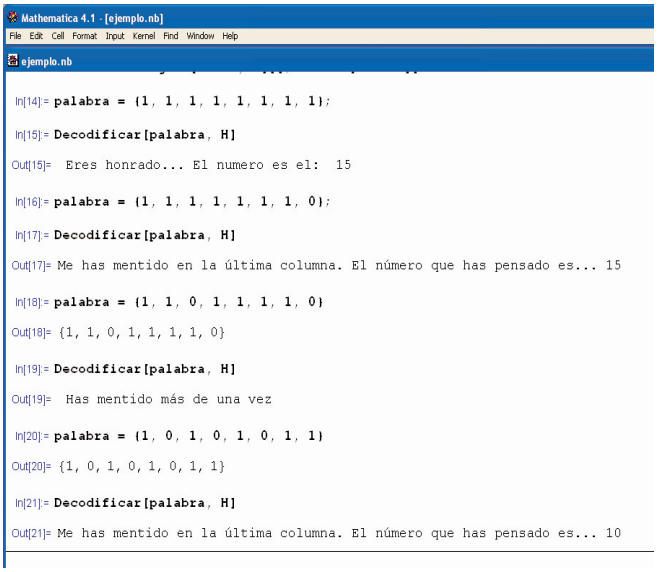
Out[12]:= {0,0,0,1}

Lo que nos indica que se ha producido un error en la última columna y así la decodificación sería 10101010, o lo que es lo mismo, 10 (dado por 1010). En este caso hemos fallado, pero ello se debe a que, como hemos indicado anteriormente, el código es capaz de detectar, a lo sumo, dos errores.

La función en Mathematica que nos permite realizar el proceso de decodificación o adivinación completo viene dado por la función Decodificar:

Ahora, si usamos esta función para probar de nuevo con los ejemplos anteriores obtenemos:





Con esta experiencia podemos tratar de concienciar a los alumnos de que cuando vemos la televisión digital, escuchamos un CD, visionamos un DVD, utilizamos un ordenador, un móvil o un cajero automático hacemos uso, sin darnos cuenta, de las matemáticas, concretamente de las matrices y la base 2.

Debido a este tipo de métodos, apreciamos que la reproducción de un CD o un DVD es perfecta, o percibimos que existe una gran diferencia de calidad entre la televisión analógica y la televisión digital y todo ello se debe a la posibilidad de arreglar los datos erróneos, ya sean debidos éstos a interferencias o a defectos del soporte que estemos utilizando (un CD o un DVD), por ejemplo, un arañazo.

De este modo, una vez más, gracias a las matemáticas obtenemos una mejor calidad de vida a través de la tecnología digital. ■

REFERENCIAS BIBLIOGRÁFICAS

BAENA RUÍZ, J. (1994): "Códigos secretos: otra forma de aplicar las matrices en Bachillerato (16-18)", *Suma*, n.º 14-15, 40-42.

GONZÁLEZ HENRÍQUEZ, J.J. (1998): "Experimentos en álgebra lineal con Mathematica", *Suma*, n.º 27, 97-102.

HAMMING, R.W. (1950): *Error detecting and error correcting codes*, Bell. System Tech. J. 29, 147-160.

HILL, R. (1997): *A first course in Coding Theory*, Oxford University Press, ISBN 0-19-853803-0.

MACWILLIAMS, F.J. y SLOANE, N.J.A. (1993): *The Theory of Error-Correcting Codes*, North-Holland Mathematical Library 16, North-Holland, ISBN 0-444-85193-3.

ALEMANY MARTÍNEZ E., BALAGUER BESER, A., MARÍN MOLINA, J. (2002): *Prácticas de Álgebra con Mathematica*, Universidad Politécnica de Valencia, ISBN 84-9705-313-3.