La Columna de Matemática Computacional

Sección a cargo de

Tomás Recio

El objetivo de esta columna es presentar de manera sucinta, en cada uno de los números de La Gaceta*, alguna cuestión matemática en la que los cálculos, en un sentido muy amplio, tengan un papel destacado. Para cumplir este objetivo el editor de la columna (sin otros méritos que su interés y sin otros recursos que su mejor voluntad) quisiera contar con la colaboración de los lectores, a los que anima a remitirle (a la dirección que se indica al pie de página*[1]*) los trabajos y sugerencias que consideren oportunos.*

En este número . . .

En este número de La Gaceta presentamos un artículo del profesor Carlos Munuera que presenta, en un estilo divulgativo, una sustanciosa (a pesar de la impuesta brevedad) introducción a la teoría de los códigos correctores de errores.

En las siguientes páginas el lector encontrará numerosos ejemplos (del entorno tecnológico en el que hoy todos nos movemos) que justifican el estudio de estos códigos; una amena presentación de la formalización de las ideas implicadas, a través de diversos juegos; y una descripción de los problemas que surgen en la búsqueda de buenos códigos y de su relación con áreas de la matemática tales como el álgebra lineal o la geometría algebraica.

El profesor Munuera, tras realizar su tesis con el profesor Tena (que también ha aportado su contribución a esta sección de La Gaceta[2]), trabaja –desde hace años– en codificación de la información y criptografía, en el Departamento de Matemática Aplicada de la Universidad de Valladolid. Ha publicado (junto con el prof. Tena) un libro y numerosos artículos sobre estos temas y es miembro de la Asociación Española de Criptología y Seguridad de la Información, de la Asociación Española de Biometría y del capítulo español del IEEE (de estos dos ultimos, miembro co-fundador).

[1] Tomás Recio. Departamento de Matemáticas. Facultad de Ciencias. Universidad de Cantabria. 39071 Santander. recio@matesco.unican.es

[2] J. Tena Ayuso: "Tests de Primalidad". La Gaceta, 3.3.

Códigos Correctores de Errores (o cuántas preguntas son necesarias para conocer un número)

por

Carlos Munuera

Introducción

Se dice a menudo que vivimos en plena era numérica. Efectivamente, buena parte de las informaciones y datos que cada día se manejan en nuestro mundo están representadas en formato *digital*, es decir numérico. Esta 'revolución digital' afecta a todos los aspectos de nuestra vida, desde la compra en el supermercado –donde la factura, que pagamos con una tarjeta de crédito, se ha elaborado a partir de los códigos de barras con que vienen marcados los productos– hasta las comunicaciones por teléfono e internet, o la música que en forma de disco compacto escuchamos en nuestra casa.

En todos estos casos, la información, que en su origen probablemente era analógica, se traduce a secuencias de números. Habitualmente estos números se escriben luego (se *codifican*) en forma binaria, como secuencias de 1 y 0 (*bits*). Por ejemplo, el bien conocido código ASCII (*'American Standard Code for Information Interchange'*) traduce cada letra en una secuencia de 8 bits. El comienzo de este texto está codificado en mi ordenador como 10000111110111101101101111011110...

Otro ejemplo paradigmático de objeto digital es el disco compacto[3]. Para la fabricación de un disco compacto de audio, se discretiza la función 'sonido' a partir de tomas discretas de datos: 2×44100 tomas por segundo (el sistema es estereofónico y permite reproducir frecuencias de hasta 20000 Hz.) cada una de las cuales se ajusta a una escala de 2^{16} niveles. Esto se traduce en $44100 \times 16 \times 2 = 1411200$ bits de información por segundo para almacenar en el disco.[4]

Con estos enormes volumenes de datos, no es extraño que uno de los problemas más importantes que genera la manipulación y transmisión de la información digital sea el de los errores. Basta una pequeña alteración del soporte que contiene o transmite la información (un rayón sobre un disco o una onda parásita) para que una parte del mensaje se corrompa: algunos de los 0 serán leídos como 1 y viceversa. En el caso del disco compacto, un pequeño rayón de 1 mm. puede alterar entre 2000 y 4000 bits. Por tanto es preciso desarrollar algún mecanismo que permita detectar cuando se han producido errores y, si es posible, corregirlos recuperando la información original. Con

[3]'Compact Disc'. Desarrollado por Philips y Sony

[4]Además, por motivos técnicos, cada 8 bits de datos (*audiobytes*) se graban en el disco mediante 17 bits. Con esto, cada segundo de música requiere almacenar 2998800 bits.

este propósito nacieron en los años 50 (coincidiendo con el desarrollo de las primeras computadoras) los *Códigos Correctores de Errores*.

En los inicios de la telefonía, la calidad del sonido transmitido era muy pobre. A menudo, para trasmitir una palabra se utilizaba un código basado en iniciales. Por ejemplo, la palabra *error* se codificaba como 'Eco-Romeo-Romeo-Omega-Romeo'. Este sistema fué luego adoptado por los ejércitos británico y norteamericano en sus transmisiones y aún lo escuchamos en películas antiguas. Desde la invención de estos primeros –e ingenuos– códigos, hasta los sumamente potentes que usamos hoy en día, se ha experimentado una tremenda evolución. Sin embargo la filosofía subyacente sigue siendo la misma: el mensaje original se trocea en partes, cada una de las cuales se codifica mediante la inclusión sistemática de información redundante. En base a esa redundancia es posible detectar los errores producidos y, si no sobrepasan ciertas cotas, corregirlos. Como contrapartida, el precio que pagamos es el aumento del tamaño de los mensajes transmitidos (o, visto de otra forma, la reducción en la cantidad de información 'neta' que contiene cada mensaje recibido).

La *Teoría de los Códigos Correctores de Errores* forma hoy un extenso y fructífero campo de interacción entre las Matemáticas y las tecnologías de la Información, en el que conceptos y resultados matemáticos abstractos permiten dar elegantes soluciones al problema de transmitir información de forma eficiente y segura. Algunos de estos tópicos matemáticos tienen aplicaciones técnicas bien establecidas desde antaño –tal es el caso del álgebra lineal o los polinomios–. Otros, por contra, han estado tradicionalmente confinados en el área de las matemáticas teóricas –como la teoría de números, los cuerpos finitos o la geometría algebraica– y esta interación ha venido a revitalizarlos y convertirlos de nuevo en objeto de estudio activo.

El fín de estas notas es proporcionar una introducción informal a este tema.

La definición

Un *código corrector de errores* es un subconjunto $\mathcal{C} \subseteq \mathcal{A}^n$, siendo $\mathcal{A}$ un conjunto finito –o *alfabeto*– y n un entero positivo. Los elementos de $\mathcal{C}$ son llamados *palabras* y n es su *longitud*.

Probablemente esta definición no resultará muy ilustrativa para el lector (por lo menos para los no familiarizados con el tema). Lo que tampoco se ve por nigún lado es la relación con el tratamiento de la información o la corrección de errores. Vayamos poco a poco.

Un juego

El subtítulo de este trabajo hace referencia a un acertijo matemático –bastante trivial– de esos que recurrentemente vemos aparecer en las secciones

de juegos matemáticos o grupos de matemáticas en las 'news' de internet. Digamos que Alicia elige un número m entre 0 y 15. Bernardo debe deducir (¡no adivinar!) este número haciéndole preguntas que ella contestará con *si* o *no*. La cuestión es *¿cuál es el mínimo número de preguntas que debe hacer Bernardo para conocer m?*

La solución es obvia. Basta escribir m en base dos para deducir, a partir de las respuestas de Alicia, cada una de las cifras ('bits') que forman esa escritura. Como la escritura binaria de 15 requiere $\lfloor \log_2(15) \rfloor + 1 = 4$ bits, son suficientes (y necesarias) cuatro preguntas. No es difícil diseñar un algoritmo para llevar a cabo esta tarea.[5]

Compliquemos un poco el juego para que no sea tan simple. En su versión original hemos asumido implícitamente que Alicia responderá siempre con la verdad. Si eliminamos esa condición –de manera que sus respuestas puedan ser verdaderas o falsas– ¿cuántas preguntas serán necesarias?

Desde luego esta versión modificada exige algunas reglas suplementarias. A saber, es preciso acotar la cantidad de mentiras que Alicia podrá emplear. De otro modo, podría responder simplemente al azar y no sería posible extraer ninguna información de sus respuestas (ni, en consecuencia, deducir el número secreto jamás). Pongamos que puede mentir una vez como máximo, *¿cuál es entonces el mínimo número de preguntas que debe hacer Bernardo para conocer m?* y *¿cuáles deben ser estas preguntas?*

Quizá en este momento no es demasiado fácil convencer al lector de que ese número es exáctamente siete. De hecho no conozco ningún argumento simple y directo que pudiera usar para demostrar tal afirmación. No obstante, si tiene la paciencia de seguir leyendo estas notas, podrá convencerse de que con siete preguntas (y no con menos) es posible saber si Alicia mintió (o no) y cual fué la respuesta falsa en su caso.

¿Qué tiene esto que ver con los codigos correctores?

La información digital se caracteriza por presentarse en un formato discreto, esto es, como una secuencia finita $m = x_1 x_2 \cdots \in \mathcal{A}^*$ de símbolos de un alfabeto finito $\mathcal{A}$. Un texto escrito (esta revista) es un buen ejemplo de información digital. Por razones de conveniencia, el alfabeto utilizado suele identificarse con algún conjunto numérico y muy a menudo –como ya antes señalamos– con $\{0, 1\}$, conjunto que interpretaremos como el cuerpo finito $\mathbb{F}_2$. Un ejemplo de codificación binaria lo da el primero de los juegos propuestos. De manera análoga, si el alfabeto $\mathcal{A}$ contiene q elementos y q es la potencia de un número primo, entonces $\mathcal{A}$ se identifica con el cuerpo finito con q elementos

[5] En el leguaje de la Teoría de la Información, cada respuesta proporciona un bit de información. Como el número secreto posee 4 bits, son necesarias cuatro preguntas. El algoritmo para deducir m consiste simplemente en solicitar a Alicia cada una de las cifras que forman su escritura binaria.

$\mathbb{F}_q$ [6]. Esta identificación permite aplicar a los problemas de codificación toda la potente artillería del álgebra y la geometría sobre cuerpos finitos.

Una vez se tiene la información en el formato digital adecuado (sea este binario o no) es apta para su manipulación o transmisión, siguiendo un esquema del tipo

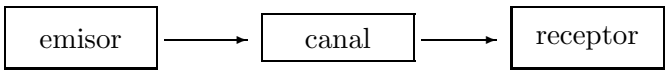

Figura 1: Esquema de una transmisión de información

En un sentido amplio, el canal puede ser espacial o temporal: envío por línea telefónica, óptica, almacenamiento en un disco, etc. Al recibir el mensaje, el receptor se encuentra precisamente en la situación descrita por el juego anterior (en su versión modificada): no puede estar seguro de cuando los datos que recibe coinciden con los enviados y cuando han resultado alterados durante la transmisión. Naturalmente es imposible imponer reglas al canal para que 'mienta' sólo una cantidad prefijada de veces. Pero sí podemos hacer algo semejante: estimar con que frecuencia (o con que probabilidad) se producen los errores.

Ahora, en lugar de enviar la información directamente, la transformamos (*codificamos*) añadiéndole cierta redundancia con arreglo a unas reglas sistemáticas. Es esta información codificada la que realmente se transmite por el canal (Figura 2). En base a la redundancia añadida el receptor puede corregir los (eventuales) errores producidos y devolverla a su formato original. Este proceso recibe el nombre de *descodificación.*[7]

[6] A pesar de que en el texto insistimos varias veces en la preponderancia de los códigos binarios en la práctica –y de que esta afirmación se repite con regularidad en la literatura sobre codificación– este dato es bastante inexacto en realidad. Muy a menudo sucede que la información se presenta como una sucesión de 0 y 1 por conveniencia de las máquinas que la gestionan, pero las operaciones matemáticas que involucra su codificación se realizan de hecho en algún cuerpo extensión de $\mathbb{F}_2$. Por ejemplo, tanto el código Reed-Solomon, utilizado habitualmente en corrección de errores (discos compactos, redes ADSL, telefonía móvil, etc.), como el AES (estándar de cifrado criptográfico), se basan en cálculos realizados en $\mathbb{F}_{2^8}$ con la aritmética de ese cuerpo. Claro está que cualquier elemento de $\mathbb{F}_{2^8}$ puede luego presentarse –y así se hace– como una sucesión de 8 bits en virtud del isomorfismo de espacios vectoriales $\mathbb{F}_{2^8} \cong \mathbb{F}_2^8$. En otros casos el cardinal del alfabeto $\mathcal{A}$ ni siquiera es una potencia de dos. Tal sucede, por ejemplo, con el código PDF417 (Symbol Technologies Inc.) –que el lector probablemente conoce como el código de barras con que graban sus datos fiscales cuando se 'declara' a Hacienda– que utiliza como alfabeto el cuerpo finito con 929 elementos.

[7] De las dos etapas involucradas en este proceso, la de corrección de errores es siempre la más compleja y la dominante desde el punto de vista computacional. Por esta razón, a

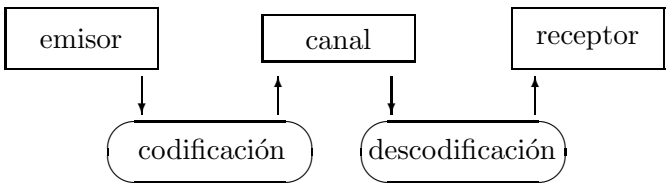

Figura 2: Esquema de una transmisión de información codificada

Lo que te diga tres veces es verdad[8]

Ilustremos las ideas que acabamos de exponer mediante unos ejemplos sencillos.

1. Códigos de repetición. Si el alfabeto usado es $\mathbb{F}_2$, una estrategia elemental es codificar 0 como 000 y 1 como 111. Una vez recibido el mensaje, se trocea en bloques de tres bits, cada uno de los cuales se descodifica por un simple criterio de mayoría. Es obvio que este código, llamado *de repetición*, permite corregir un error en cada bloque, al precio de multiplicar por tres el volumen de datos enviados. De forma más general, puede codificarse cada símbolo del alfabeto $\mathcal{A}$, repitiéndolo n veces (n impar). Esta estrategia permite corregir $(n-1)/2$ errores en cada palabra (de nuevo, mediante un criterio de mayoría). El precio que pagamos por este aumento en la seguridad –multiplicar por n el volumen de los datos enviados– es muy elevado, con lo que estos códigos son muy poco eficientes.

2. El código ASCII. En su versión habitual (no extendida), ASCII permite codificar $128 = 2^7$ símbolos (letras, números, signos y controles no imprimibles) de uso general para computadoras. A cada uno de ellos se le asigna un número de orden y se le codifica mediante la escritura binaria (con 7 bits) de ese número. Para aumentar la fiabilidad de esta codificación, a cada 7-upla $(x_1, \ldots, x_7) \in \mathbb{F}_2^7$ se le añade un bit control x_8, calculado de manera que $x_1 + \cdots + x_7 + x_8 \equiv 0 \pmod 2$. Este sistema permite detectar, aunque no corregir, cualquier número impar de errores.

3. El código 2/5. Es usado en varios tipos de códigos de barras (también llamados 2/5). Como se sabe, un *código de barras* es un símbolo gráfico que permite almacenar información en base a la alternancia de barras y espacios en blanco de distintos espesores. Los códigos de barras de la familia 2/5 son numéricos –permiten almacenar un número entero escrito en formato decimal– y binarios –utilizan barras y espacios de dos espesores distintos–.

---

menudo se usa en la literatura la palabra 'descodificación' como sinónimo de 'corrección de errores'.

[8] Lewis Carrol, *A la caza del snark* (1876).

Para almacenar un número m en uno de estos códigos de barras, previamente se realiza una transcripción binaria de m. Esta transcripción se realiza concatenando la codificación de cada una de las cifras decimales de m siguiendo la regla dada por la tabla

0	00110	5	10100
1	10001	6	01100
2	01001	7	00011
3	11000	8	10010
4	00101	9	01010

Nótese que cada cifra se codifica con cinco bits: dos 1 y tres 0 (de donde viene su nombre 2/5). Nótese también que existen precisamente diez de tales combinaciones. En el esquema de la definición que vimos al comienzo, el alfabeto usado es $\mathbb{F}_2 = \{0, 1\}$, el código es $\mathcal{C} = \{00110, \ldots, 01010\} \subseteq \mathbb{F}_2^5$ y su longitud es 5. Claramente se observa que $\mathcal{C}$ permite detectar (aunque no corregir) cualquier configuración con un número impar de errores en cualquiera de los 5 bits que componen una palabra.

Para aumentar la fiabilidad del código, se añade al mensaje m un dígito de control C. Este se coloca a la derecha del mensaje y se calcula del siguiente modo: se numeran las cifras de m de derecha a izquierda (comenzando por la de control C). Se calculan la suma de las que están en posición par, P, y de las que están en posición impar, I. El dígito de control queda determinado por la condición $3P + I \equiv 0 \pmod{10}$. Por ejemplo, el mensaje 1360140 se codifica como $1360140C$ con $3(1+6+1+0)+(3+0+4+C) = 24+7+C = 31+C \equiv 0 \pmod{10}$, luego $C = 9$ y la codificación es 13601409.

Finalmente se realiza la implementación gráfica, traduciendo en barras y espacios la escritura numérica binaria del mensaje (incluyendo el dígito de control). Para ello 1 significa elemento ancho y 0 elemento estrecho. La forma particular en que se lleva a cabo esta traducción gráfica no es única, y determina la existencia de distintas versiones de códigos 2/5 (existen al menos 7 de tales versiones). Por ejemplo, en la versión '2/5 con 5 barras' solamente las barras son significativas, con lo que el mensaje $m = 34$ queda

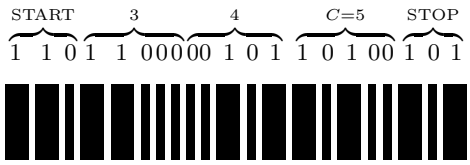

Figura 3: Codificación 2/5 del mensaje $m = 34$.

(La inclusión de los signos de comienzo y final –*start* y *stop*– tiene por finalidad delimitar el código, indicar el sentido correcto de lectura e identificar el tipo de codificación que se está usando). Se deja a cargo del lector analizar cuántos errores pueden corregirse y cómo llevar a cabo este proceso.

Formalizando

Como hemos señalado, la información se presenta originalmente como una secuencia $m = x_1 x_2 \cdots \in \mathcal{A}^*$. Fijamos dos enteros $k < n$ y troceamos m en bloques de longitud k : $m = (x_1 \cdots x_k)(x_{k+1} \cdots x_{2k}) \cdots$. Cada uno de estos bloques se codifica (y posteriormente se enviará y descodificará) independientemente de los demás, como su imagen mediante una aplicación inyectiva $c : \mathcal{A}^k \longrightarrow \mathcal{A}^n$.[9] La codificación del mensaje completo se obtiene concatenando la codificación de los bloques que lo integran:

$$c(m) = c(x_1, \ldots, x_k)c(x_{k+1}, \ldots, x_{2k}) \cdots$$

El conjunto $\mathcal{C} = \mathrm{Im}(c)$ es, por definición, el *código* utilizado. Cada palabra de $\mathcal{C}$ contiene k símbolos de información y $n-k$ símbolos redundantes: el número k/n se llama *tasa de transmisión* de $\mathcal{C}$.

Supongamos que se ha enviado una palabra $\mathbf{c} \in \mathcal{C}$ y recibido un vector $\mathbf{x} \in \mathcal{A}^n$. Si $\mathbf{x} \notin \mathcal{C}$ podemos estar seguros de que ha habido errores. De hecho, si p es la probabilidad de que un símbolo resulte alterado en la transmisión, podemos esperar una media de np símbolos erróneos en $\mathbf{x}$. La capacidad de corrección de errores de $\mathcal{C}$ debe superar al menos esa cota. Aún en el caso de que $\mathbf{x} \in \mathcal{C}$, nunca podremos estar realmente seguros de que no han existido errores. Ahora bien, si el código se ha diseñado correctamente, sus palabras serán muy 'diferentes' unas de otras, de manera que resulte 'suficientemente improbable' que una resulte transformada en otra por los errores aleatorios sufridos en el canal.

La forma adecuada de medir la diferencia entre dos palabras (o dos vectores de $\mathcal{A}^n$) es la distancia de Hamming. Dados $\mathbf{x}, \mathbf{y} \in \mathcal{A}^n$, llamamos *distancia de Hamming* entre $\mathbf{x}$ e $\mathbf{y}$ al número de coordenadas distintas que poseen:

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

Obsérvese que la función d es realmente una distancia en $\mathcal{A}^n$. El hecho de que d *no* sea invariante por cambios de base, hace que la Teoría de Códigos no sea

[9] En puridad deberíamos escribir $c : \mathcal{A}^k \longrightarrow \mathcal{B}^n$, puesto que los alfabetos en que esta escrita originalmente la información y en que se codifica no tienen por que coincidir. Sin embargo, si sólo estamos interesados en la codificación contra errores, podemos siempre asumir que $\mathcal{A} = \mathcal{B}$.

una parte trivial del álgebra lineal. La capacidad de corrección de errores de $\mathcal{C}$ viene determinada por su *distancia mínima*, definida como

$$d = d(\mathcal{C}) = \min\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in \mathcal{C},\, \mathbf{x} \neq \mathbf{y}\}.$$

En efecto, recibido el vector $\mathbf{x} \in \mathcal{A}^n$, se descodifica $\mathbf{x}$ por la palabra $\mathbf{c} \in \mathcal{C}$ más parecida a $\mathbf{x}$, es decir, que minimiza $d(\mathbf{x}, \mathbf{c})$ (si existe). Como las bolas (para la métrica de Hamming) de radio $(d-1)/2$ centradas en las palabras del código son disjuntas, si el número de errores en $\mathbf{x}$ no supera $\lfloor (d-1)/2 \rfloor$, entonces la palabra corregida coincide con la realmente enviada. En definitiva, nuestra estrategia permite detectar $d-1$ errores y corregir $\lfloor (d-1)/2 \rfloor$ errores.

El objetivo principal (o mejor, uno de los objetivos principales) de la Teoría de Códigos Correctores de Errores es encontrar *buenos* códigos, es decir, códigos que maximicen solidariamente los parámetros k/n y d/n. Sin embargo estas demandas son mutuamente contradictorias: al aumentar uno de los parámetros, el otro tiende siempre a disminuir. En la práctica habremos de conformarnos con un cierto equilibrio entre ellos.

Otro requerimiento importante para un buen código es que posea algún método de descodificación computacionalmente efectivo. El sistema al que nos hemos referido anteriormente –evaluar la distancia de $\mathbf{x}$ a todas las palabras de $\mathcal{C}$ y quedarnos con la más cercana– es inviable en la práctica (excepto para códigos de pequeño tamaño). Relativamente pocos códigos permiten estos métodos efectivos.[10] Retomaremos el tema de los buenos códigos un poco más adelante.

Otro ejemplo mas: el código de Hamming

Volvamos al juego con el que comenzamos (en su versión modificada). En términos de códigos correctores podemos interpretar el número secreto m (o su escritura binaria) como el mensaje original, las respuestas correctas a las preguntas hechas para conocerlo (escritas en términos de 0 y 1 si se quiere) como la codificación $\mathbf{c}$ de m y la respuestas efectivamente dadas a esas preguntas (incluyendo las mentiras deliberadamente introducidas) como el vector recibido $\mathbf{x}$. Con este esquema, la cuestión que planteábamos puede reescribirse como *determinar la menor longitud posible, n, para un código binario que codificando 4 bits (k=4) tenga distancia mínima* $d \geq 3$.

Veremos a continuación que efectivamente existe un código con las condiciones requeridas y longitud $n = 7$. Un poco más adelante comprobaremos que $n = 7$ es mínimo con esta propiedad.

Deseamos pues codificar una 4-upla $(x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4$. Vamos a hacerlo añadiendo a estos cuatro bits otros tres redundantes,

[10] En el lenguaje de la Teoría de la Complejidad Computacional, el problema de descodificar un código es NP-Completo.

$c(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. Para ello consideremos tres circunferencias cortándose en posición general, tal y como se ve en el dibujo. Estas tres circunferencias determinan 7 regiones (más la exterior no acotada). Numerémoslas.

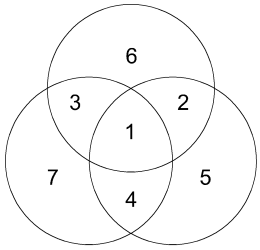

Figura 4: Tres circunferencias

Coloquemos cada bit x_i en la región i de la figura, $(i = 1, \ldots, 7)$. Los bits x_5, x_6, x_7, se calculan de manera que cada círculo contenga un número par de 1. Por ejemplo, 1010 se codifica como 1010100. El conjunto $\mathcal{C} = \{c(x_1, x_2, x_3, x_4) \mid x_1, x_2, x_3, x_4 \in \mathbb{F}_2^4\}$ se llama *código de Hamming binario de redundancia 3* y habitualmente se denota $\mathcal{H}_2(3)$ [11].

Es un ejercicio instructivo (y fácil) comprobar que dos palabras cualesquiera de $\mathcal{H}_2(3)$ se diferencian en al menos tres coordenadas (es decir, que $\mathcal{H}_2(3)$ tiene distancia mínima $d = 3$) y diseñar un algoritmo de corrección de errores para él. Se observará que en este código (a diferencia de lo que ocurre en general) la descodificación de cualquier vector recibido es siempre posible (por esta razón se dice, con cierto optimismo, que $\mathcal{H}_2(3)$ es un código *perfecto*). La descodificación obtenida será correcta cuando el vector recibido contenga un error como máximo e incorrecta en otro caso.

[11] Para cada potencia q de un número primo existe una familia infinita de códigos de Hamming sobre $\mathbb{F}_q$. Estos códigos fueron desarrollados por Richard W. Hamming (1915-1998) y Marcel J.E. Golay (1902-1989), quienes son considerados como los fundadores de la Teoría de la Codificación. Otros importantes códigos, introducidos por Golay y estudiados por ambos, son los llamados *códigos de Golay*, estrechamente relacionados con los de Hamming y a los que Mac Williams y Sloane ([3] de 1977) se refieren como *'probably the most important of all codes, for both theoretical and practical reasons'* (aunque esta afirmación probablemente ya no sigue vigente hoy en día). Para no alargar excesivamente la exposición, no trataremos aquí estos códigos, aunque recomendamos su estudio a todos los lectores interesados en estos temas.

Hágalo lineal

Hemos citado ya que entre los requisitos de un buen código $\mathcal{C}$ está el de poseer algoritmos eficaces de codificación y descodificación. Por lo general, esta condición pasa por que $\mathcal{C}$ posea alguna estructura algebraica. En todo lo que sigue supondremos que el alfabeto usado $\mathcal{A}$ tiene por cardinal, q, la potencia de un número primo e identificaremos $\mathcal{A}$ con $\mathbb{F}_q$ el cuerpo finito con q elementos.

Por ejemplo, el código de Hamming ¿posee alguna estructura algebraica? No hay más que transcribir la condición de que cada uno de los tres círculos contenga un número par de 1 en términos de ecuaciones,

$$\begin{cases} x_1 & +x_2 & +x_3 & & & +x_6 & & \equiv 0 \pmod 2 \\ x_1 & +x_2 & & +x_4 & +x_5 & & & \equiv 0 \pmod 2 \\ x_1 & & +x_3 & +x_4 & & & +x_7 & \equiv 0 \pmod 2. \end{cases}$$

¡ $\mathcal{H}_2(3)$ es un subespacio vectorial de $\mathbb{F}_2^7$! También la aplicación de codificación es lineal:

$$c(x_1,x_2,x_3,x_4) = (x_1,x_2,x_3,x_4) \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

En general, si la aplicación de codificación $c : \mathbb{F}_q^k \to \mathbb{F}_q^n$ es lineal, decimos asimismo que el código $\mathrm{Im}(c)$ es lineal. Por tanto un *código lineal q-ario de longitud n* no es más que un subespacio vectorial $\mathcal{C} \subseteq \mathbb{F}_q^n$. El entero k es la dimensión de $\mathcal{C}$ como espacio vectorial y nos referiremos a él simplemente como *dimensión de $\mathcal{C}$*.

Los códigos utilizados en la práctica (excepto algunos de pequeño tamaño, como 2/5) son siempre lineales. A continuación veremos como los procesos de codificación y descodificación, y el cálculo de la distancia mínima, son mucho más simples para los códigos lineales que para aquellos que no lo son.

Sea pues $\mathcal{C}$ un código lineal q-ario de longitud n y dimensión k (abreviadamente, un código $[n,k]_q$). Una matriz G cuyas filas constituyan una base de $\mathcal{C}$ se llama *matriz generadora* de $\mathcal{C}$. Además de determinar $\mathcal{C}$, G permite la codificación de cualquier mensaje $\mathbf{a} \in \mathbb{F}_q^k$ simplemente mediante la regla $c(\mathbf{a}) = \mathbf{a} \cdot G$.

Otra forma de determinar $\mathcal{C}$ es mediante un sistema de ecuaciones implícitas. La matriz de unas ecuaciones implícitas de $\mathcal{C}$ (es decir, una matriz H con n columnas, $n-k$ filas y rango $n-k$, tal que $\mathbf{x} \in \mathcal{C}$ si y sólo si $H\mathbf{x}^t = \mathbf{0}^t$) se llama *matriz de control de $\mathcal{C}$*. Obviamente las matrices generadora y de control están relacionadas por la condición $GH^t = 0$.

La distancia mínima de $\mathcal{C}$ se obtiene muy fácilmente a partir de una matriz de control, H, ya que d coincide con el mínimo número de columnas linealmente

dependientes de H. Por ejemplo, en el caso del código de Hamming *una* matriz de control es

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

luego su distancia mínima es 3 (no hay columnas repetidas y la suma de las dos primeras coincide con la última).[12]

Los códigos lineales poseen además un algoritmo genérico de descodificación mucho más eficiente que el de fuerza bruta. En lugar de describir este algoritmo en general, vamos a particularizarlo para el caso, que venimos tratando, de $\mathcal{H}_2(3)$.

Recordemos que este código tiene distancia 3, luego corrige un error. Supongamos emitida una palabra $\mathbf{c} \in \mathcal{H}_2(3)$ y recibido un vector $\mathbf{x} \in \mathbb{F}_2^7$, $\mathbf{x} = \mathbf{c}+\mathbf{e}$ siendo $\mathbf{e}$ el vector error. Llamamos *síndrome* de $\mathbf{x}$ al vector

$$s(\mathbf{x}) = H\mathbf{x}^t = H(\mathbf{c}+\mathbf{e})^t = H\mathbf{e}^t \in \mathbb{F}_2^3$$

(puesto que $H\mathbf{c}^t = \mathbf{0}^t$). Así pues $s(\mathbf{x}) = s(\mathbf{e})$ nos informa de la clase de equivalencia de $\mathbb{F}_2^7/\mathcal{H}_2(3)$ a la que pertenence el error. Si $s(\mathbf{x}) = \mathbf{0}^t$ entonces $\mathbf{x} \in \mathcal{H}_2(3)$ y damos esta palabra por buena. En caso contrario $s(\mathbf{x})$ coincidirá con una de las columnas de H. Si esta columna es la i-ésima, para descodificar $\mathbf{x}$ basta corregir su i-ésima coordenada. Si $\mathbf{x}$ contiene solamente un error, entonces $\mathbf{e}$ posee exáctamente una coordenada no nula y la descodificación es correcta.

Para otros códigos lineales distintos de $\mathcal{H}_2(3)$ –generalmente con mayor capacidad de corrección de errores– el método que hemos descrito se complica y habitualmente pasa por el uso de tablas. Por lo tanto pierde eficacia desde el punto de vista computacional. Sin embargo, códigos lineales particulares (construidos haciendo cada vez más fuerte la estructura algebraica subyacente) poseen algoritmos de descodificación muy eficaces. Son precisamente estos códigos y algoritmos los usados en la práctica.

¿Qué hemos ganado?

Ilustremos con un simple ejemplo como aumenta la fiabilidad de la transmisión mediante el empleo de los códigos correctores. Refiriéndonos de nuevo al código de Hamming $\mathcal{H}_2(3)$, recordemos que codifica tandas de 4 bits de

[12] Obsérvese que las columnas de H son las escrituras binarias de los enteros entre 1 y 7 (o, para los lectores más inclinados a la geometría, las coordenadas de los puntos del plano proyectivo sobre $\mathbb{F}_2$). Habitualmente el código de Hamming no se presenta exáctamente como lo estamos haciendo nosostros, sino que se permutan sus coordenadas para que estas columnas sean las escrituras binarias de los enteros 1 a 7 en su orden natural.

información representando cada una mediante 7 bits. El sistema permite detectar dos errores y corregir uno. Si, durante la transmisión, la probabilidad de error por bit es de 0.1 (suposición únicamente académica y –afortunadamente– muy poco realista), entonces un sencillo cálculo muestra que

- la probabilidad de transmisión sin error en 4 bits de información es 0.6561 sin codificar y 0.8503 codificando;
- la probabilidad de transmisión incorrecta no detectada en 4 bits de información es 0.3439 sin codificar y 0.0257 codificando (13.4 veces más pequeña).

Claro está que esta ganancia se consigue al precio de enviar un volumen de datos 7/4 veces mayor.

El dominio de los códigos

Retomemos el tema de los parámetros de un buen código. En todo lo que sigue nos referimos códigos lineales $\mathcal{C}$ definidos sobre $\mathbb{F}_q$, de tipo $[n, k]$ y distancia mínima d.

Los parámetros de $\mathcal{C}$ son tanto mejores cuanto mayores sean los cocientes $\delta = d/n$ y $R = k/n$. Como $0 \leq \delta, R \leq 1$, podemos representar $\mathcal{C}$ por el punto de coordenadas (δ, R) en el cuadrado unidad $[0, 1] \times [0, 1]$. Naturalmente, nuestro objetivo es que este punto se encuentre bastante cerca de la esquina superior derecha. Lamentablemente, por lo general sucede justo lo contrario y el punto obtenido se aproxima decepcionantemente a la inferior izquierda. De hecho, buena parte del cuadrado (incluyendo la apreciada esquina superior-derecha) está prohibido, en función de ciertas restricciones (o cotas) sobre los parámetros de $\mathcal{C}$.

Por ejemplo, si recordamos el juego con el que iniciamos este trabajo, teníamos pendiente comprobar que 7 era precisamente el número mínimo de preguntas que debemos hacer para conocer el número secreto o, equivalentemente, que $n = 7$ es la mínima longitud posible para un código binario con 16 palabras y distancia 3. Ahora bien, si existiera un tal código $\mathcal{C}$ de longitud $n = 6$, como las bolas de radio 1 y centro en las palabras de $\mathcal{C}$ son disjuntas, y como cada una contiene 7 vectores, se verificaría que $16 \cdot 7 \leq \#\mathbb{F}_2^6 = 2^6$, lo que es falso. Por tanto, son necesarias al menos 7 preguntas y por fín damos el juego por terminado.

De manera general, si un código $\mathcal{C} \subseteq \mathbb{F}_q^n$ es capaz de corregir t errores, entonces las bolas de centro las palabras de $\mathcal{C}$ y radio t son disjuntas, luego $\#\mathcal{C} \cdot b_q(n, t) \leq q^n$, siendo $b_q(n, r)$ el cardinal de una bola de radio r en $\mathbb{F}_q^n$

$$b_q(n, r) = 1 + \left(\begin{array}{c} n \\ 1 \end{array} \right) (q-1) + \cdots + \left(\begin{array}{c} n \\ r \end{array} \right) (q-1)^r.$$

Esta desigualdad recibe el nombre de *cota de Hamming* y es una de las principales cotas conocidas. Otra cota importante, y posiblemente la más simple de todas, es la siguiente:

Sea H una matriz de control de $\mathcal{C}$. Como sabemos, su distancia mínima coincide con el menor número de columnas linealmente dependientes en H. Ahora bien, como H tiene rango $n-k$, este número es $n-k+1$ como máximo. Es decir $d \leq n-k+1$, luego $d+k \leq n+1$ o equivalentemente

$$\delta + R \leq 1 + \frac{1}{n}$$

desigualdad que se conoce como *cota de Singleton.*[13]

En base a estas (y otras) restricciones, sólo una parte del cuadrado $[0,1] \times [0,1]$ es accesible a los parámetros (δ, R) de un código. El conjunto

$$U_q = \{(d/n, k/n) \mid \text{existe un código lineal } q\text{-ario de parámetros } [n,k,d] \ \}$$

es llamado *dominio de los códigos*. Más interés que el propio U_q lo tiene el conjunto de sus puntos de acumulación. U_q' está delimitado por los lados del cuadrado unidad y por la gráfica de la función $\alpha_q : [0,1] \to [0,1]$, $\alpha_q(\delta) = \sup\{R \mid (\delta, R) \in U_q'\}$.

Las desigualdades de Hamming y Singleton proporcionan cotas superiores para α_q. Una cota inferior la da el siguiente teorema, que enunciamos sin demostración.

Teorema (de Gilbert-Varshamov). *Si* $0 \leq \delta \leq (q-1)/q$, *entonces* $\alpha_q(\delta) \geq 1 - H_q(\delta)$, *siendo* H_q *la función entropía,* $H_q(\delta) = \delta \log_q(q-1) - \delta \log_q(\delta) - (1-\delta)\log_q(1-\delta)$. *Si* $(q-1)/q \leq \delta \leq 1$, *entonces* $\alpha_q(\delta) = 0$.

La figura siguiente muestra la interpretación gráfica de las versiones asintóticas de las cotas de Hamming y Gilbert-Varshamov para $q = 2$. La gráfica de la función α_q discurre –de algún modo desconocido hasta ahora– por la región que delimitan estas dos curvas, comenzando en el punto $(0,1)$ y terminando en $(1/2, 0)$ desde donde, siguiendo el eje, llega hasta $(1,0)$.

Durante mucho tiempo, uno de los problemas cruciales de la Teoría de Códigos fué encontrar una sucesión de códigos $\mathcal{C}_i$ de parámetros $[n_i, k_i, d_i]$ tales que $n_i \to \infty$, $d_i/n_i \to \delta$ y $k_i/n_i \to R$, estando el punto (δ, R) por encima de la cota de Gilbert-Varshamov. De hecho, hasta 1970 no fué posible encontrar una sucesión de códigos con límite (δ, R) tal que $\delta R \neq 0$ (estas sucesiones se llaman *asintóticamente buenas*). En la actualidad, la cuestion de encontrar sucesiones con límite por encima de Gilbert-Varshamov ha sido parcialmente resuelta con la ayuda de los códigos de Goppa que veremos a continuación.

[13] Existen códigos lineales para los que se alcanza la igualdad $d+k = n+1$. Estos códigos son llamados *'Maximum Distance Separable'* (o MDS) y juegan un papel preponderante en la Teoría de la Codificación.

Figura 5: Cotas de Hamming y Gilbert-Varshamov para $q = 2$.

Códigos y geometría

Citamos al comienzo de este trabajo que el desarrollo de la Teoría de Códigos Correctores ha impulsado a su vez el de otras ramas de las matemáticas, en ocasiones prácticamente abandonadas desde hacía mucho tiempo. Podríamos dar bastantes ejemplos de este fenómeno. Quizá el más representativo, y desde luego el más citado, es el de los códigos cíclicos y BCH, basados en la Teoría de Cuerpos finitos y en los que juegan un papel esencial tópicos como ideales, elementos primitivos, raíces de la unidad, etc. Sin embargo, no es este el ejemplo que deseamos exponer aquí (en todo caso el lector interesado puede encontrarlo en prácticamente cualquier libro de Teoría de Códigos, como [2, 3, 4, 5]). El ejemplo que nos interesa en este momento es el de los códigos obtenidos a partir de curvas algebraicas.

La posibilidad de aplicar las técnicas de la Geometría Algebraica sobre cuerpos finitos a la construcción de códigos fué descubierta por el matemático ruso V.D. Goppa en los años 80. Ya en 1973 Goppa había mostrado como usar funciones racionales sobre una recta proyectiva para construir las matrices de control de ciertos códigos (ahora conocidos como *de Goppa clásicos*) que hicieron posible, por primera vez, alcanzar la cota de Gilbert-Varshamov. El siguiente paso fué sustituir la recta proyectiva por una curva arbitraria y habilitar toda la potente maquinaria de la geometría algebraica para el estudio de los códigos obtenidos.

Sea $\mathcal{P} = \{P_1, \cdots, P_n\}$ un conjunto de puntos distintos pertenecientes a cierto 'objeto geométrico' $\mathcal{X}$ (el significado preciso de este término se irá aclarando posteriormente). Si V es un espacio vectorial de funciones $f : \mathcal{X} \longrightarrow \mathbb{F}_q$, podemos considerar la aplicación de *evaluación*

$$ev_{\mathcal{P}} : V \longrightarrow \mathbb{F}_q^n \ , \ ev_{\mathcal{P}}(f) = (f(P_1), \ldots, f(P_n)).$$

Si $ev_{\mathcal{P}}$ es lineal, su imagen es un subespacio vectorial de $\mathbb{F}_q^n$, es decir, un código lineal sobre $\mathbb{F}_q$ de longitud n. Sus palabras son los $ev_{\mathcal{P}}(f)$, $f \in V$. Decimos que este código es obtenido *por evaluación de las funciones de* V *en* $\mathcal{P}$.

Podemos aclarar el significado y alcance de esta definición mediante unos ejemplos.

Sean $\mathcal{X}$ la recta proyectiva sobre $\mathbb{F}_q$, $\mathcal{P} = \mathbb{F}_q = \{a_1, \ldots, a_q\}$ el conjunto de puntos afines de $\mathcal{X}$ y $V = \{\text{polinomios de } \mathbb{F}_q[X] \text{ con grado } < r\}$ para cierto $r \leq q$. El código de evaluación asociado a estos datos tiene por matriz generadora a

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_q \\ \vdots & \vdots & & \vdots \\ a_1^{r-1} & a_2^{r-1} & \cdots & a_q^{r-1} \end{pmatrix}$$

y se llama *código Reed-Solomon* (o RS) de dimensión r.

Calculemos sus parámetros. Su longitud es obviamente $n = q$. Su dimensión coincide con la de V (puesto que $ev_{\mathcal{P}}$ es inyectiva), es decir r. Para evaluar su distancia mínima observemos que un polinomio f de V tiene a lo más $r-1$ raíces, luego $ev_{\mathcal{P}}(f)$ tiene al menos $q-(r-1)$ coordenadas no nulas y $d \geq q - r + 1$. Si comparamos estos datos con la cota de Singleton (enunciada en el parágrafo anterior) observaremos que se alcanza la igualdad (luego $d = q - r + 1$): los códigos RS poseen los mejores parámetros posibles y son, de lejos, los más utilizados en la práctica (como ya hemos señalado, son usados en los discos compactos, televisión y telefonía digital, redes ADSL, etc.). Sin embargo, estos códigos poseen un grave inconveniente: su longitud es demasiado pequeña. Por ejemplo, la longitud de un código RS binario es 2. Para superar este problema podemos recurrir a dos tipos de estrategias. La primera consiste en utilizar códigos RS sobre extensiones $\mathbb{F}_{q^s}$ del alfabeto utilizado $\mathbb{F}_q$. La segunda se basa en encontrar alguna variación favorable en su método de construcción.

Una variación obvia es la siguiente: tomemos $\mathcal{X} = \mathcal{P} = \mathbb{F}_q^m$ y $V = \{\text{polinomios de } \mathbb{F}_q[X_1, \ldots, X_m] \text{ con grado } \leq r\}$. Los códigos obtenidos, llamados de *Reed-Muller*, $RM_q(r, m)$, tienen longitud $n = q^m$ arbitrariamente grande y fueron utilizados en los años 70 por las sondas 'Mariner' para la transmisión de datos desde el espacio exterior.

Otra generalización, quizá más interesante, es la siguiente: tomemos ahora como $\mathcal{X}$ una curva algebraica proyectiva, lisa y absolutamente irreducible definida sobre $\mathbb{F}_q$. Para cada divisor racional D de $\mathcal{X}$, el conjunto

$$V = \mathcal{L}(D) = \{\text{funciones racionales } f \in \mathbb{F}_q(\mathcal{X})^* \mid \operatorname{div}(f) + D \geq 0\} \cup \{0\}$$

(siendo $\operatorname{div}(f)$ el divisor de f), es un espacio vectorial sobre $\mathbb{F}_q$, cuya dimensión siempre es finita y denotamos por $\ell(D)$.

Sea $\mathcal{P} = \{P_1, \ldots, P_n\}$ un conjunto de n puntos de $\mathcal{X}$ racionales y distintos, con los que construimos el divisor $D = P_1 + \cdots + P_n$. Sea, finalmente, G otro

divisor sobre $\mathcal{X}$, racional y con soporte disjunto del de D, $\mathrm{sop}(G)\cap\mathrm{sop}(D)=\emptyset$. La aplicación de evaluación habitual

$$ev_{\mathcal{P}} : \mathcal{L}(G) \longrightarrow \mathbb{F}_q^n \ ; \ ev_{\mathcal{P}}(f) = (f(P_1), \ldots, f(P_n))$$

está bien definida y es lineal. En efecto, como $P_i \notin \mathrm{sop}(G)$, P_i no puede ser un polo de $f \in \mathcal{L}(G)$. Por otro lado, tanto P_i como G son racionales sobre $\mathbb{F}_q$, luego $f(P_i) \in \mathbb{F}_q$. Llamamos *código algebraico-geométrico* (o *código geométrico de Goppa*) a $\mathcal{C}(D,G) = ev_{\mathcal{P}}(\mathcal{L}(G))$.

$\mathcal{C}(D,G)$ es un código lineal de dimensión $k = \ell(G)-\ell(G-D)$ (que podemos estimar mediante el teorema de Riemann-Roch) y distancia $d \geq n - \deg(G)$ (ya que $f \in \mathcal{L}(G)$, posee a lo más $\deg(G)$ ceros en $\mathcal{P}$). Si, para simplificar, suponemos que $2g-2 < \deg(G) < n$, siendo g el género de $\mathcal{X}$, entonces el teorema de Riemann-Roch permite calcular exáctamente la fórmula para la dimensión y $k = \deg(G) + 1 - g$.

Para evaluar la calidad de los parámetros de $\mathcal{C}(D,G)$ (limitándonos por comodidad al caso $2g-2 < \deg(G) < n$), utilizamos las estimaciones anteriores y la cota de Singleton, obteniendo

$$n + 1 - g \leq k + d \leq n + 1.$$

Los códigos construidos a partir de curvas racionales (de género 0) son siempre MDS (de hecho, RS extendidos). La acotación va 'empeorando' a medida que crece el género de la curva. Obsérvese también que si hacemos el cociente $\deg(G)/n$ constante, entonces la calidad de los códigos obtenidos varía según el valor de n/g, lo que en definitiva depende de la proporción $\#\mathcal{X}(\mathbb{F}_q)/g$. Esta simple observación ha dado lugar en los últimos años a una intensa investigación para determinar cual es el máximo número de puntos de una curva sobre un cuerpo finito (con respecto a su género) y construir curvas con muchos puntos.

Por ejemplo, un tipo de curvas con muchos puntos son las modulares. Utilizando estas curvas como base pueden obtenerse sucesiones de códigos algebraico-geométricos que superan la cota de Gilbert-Varshamov (ver [6]):

Teorema (Tsfasman, Vladut, Zink, 1982) *Para $p > 7$, existe una sucesión de códigos algebraico-geométricos sobre $\mathbb{F}_{p^2}$ cuyo punto límite supera la cota de Gilbert-Varshamov.*

El lector interesado en obtener una información más detallada sobre los códigos AG puede consultar [1, 6].

A modo de epílogo: Los códigos correctores ayer y hoy

Ya hemos señalado que el origen de los códigos correctores aparece ligado al desarrollo de las primeras computadoras. Otro importante factor de desarrollo

lo encontramos en las transmisiones de datos desde el espacio (satélites de comunicaciones, sondas espaciales). Las señales transmitidas desde el exterior de la tierra resultan muy afectadas por interferencias y ruidos cósmicos. Así, desde el inicio de los programas espaciales, la NASA se convirtió en uno de los mayores centros de investigación, desarrollo y uso de las tecnologías de codificación.

Un pequeño paseo por la historia de los viajes espaciales nos lleva a 1965, año en que por primera vez fué posible fotografiar un planeta desde una nave espacial. La sonda Mariner 4 obtuvo 22 fotografías en blanco y negro de Marte. Cada fotografía se descomponía en 200×200 regiones (*pixeles*), cada una de los cuales se ajustaba a una escala de $64(= 2^6)$ niveles de gris, desde el blanco (000000) hasta el negro (111111). Como los datos eran transmitidos a una velocidad de $8\frac{1}{3}$ bits por segundo, la transmisión de cada fotografía requería unas 8 horas. Además de la lentitud del proceso, la calidad de las fotografías recibidas fué decepcionante. A partir de esta fecha, la NASA comenzó a utilizar códigos correctores en sus transmisiones.

Entre 1969 y 1977 los programas Mariner (6,7 y 9) y Viking permitieron obtener fotografías de Marte de mucha mayor calidad (todavía en blanco y negro en el caso de los Voyager, y en color con el programa Viking a partir de 1977). Las razones de esta mejora de calidad hay que buscarlas en la reducción del tamaño de los pixeles (que pasaron de 200×200 a 700×832, con el correspondiente aumento de nitidez) y al uso del código corrector Reed-Muller: se mantuvo la escala de 6 niveles de gris en cada pixel, pero se añadieron 26 bits adicionales para la corrección de errores[14].

Finalmente, entre 1979 y 1989 el programa Voyager permitió obtener fotografías de los planetas exteriores del sistema solar (principalmente Júpiter y Saturno). En estos casos, los códigos usados fueron los de Golay y Reed-Solomon.

En la actualidad los códigos correctores de errores se usan en prácticamente todos los dispositivos de tratamiento de información. El propósito que se persigue con su uso es –obviamente– corregir los errores que se producen en el manejo de la información. Pero no sólo eso. En el estado actual de la tecnología, en muchos casos podemos manipular información sin usar códigos correctores y sin cometer apenas errores. Pero no de forma eficiente. Usar códigos correctores permite *poder* cometer errores y, en consecuencia, aumentar la eficacia de los sistemas que gestionan la información. Aclaremos este punto con un ejemplo. Tomemos de nuevo el caso de un disco compacto. La información que contiene está grabada sobre su superficie a lo largo de una pista espiral de unos 5 km. de longitud y 6 μm. de anchura. Esta pista contiene tramos a dos alturas, llamados *picos* y *valles* (incidentalmente, 0 significa 'permanencia a la misma altura' mientras que 1 significa 'cambio de altura').

[14]En otras palabras, se utilizó el código Reed-Muller binario de parámetros [32,6]. La distancia mínima de este código es 16, luego permite corregir 7 errores en cada tanda de 32.

La altitud de los picos es de 0.12 μm (todos estos datos son nominales. En la práctica existe una cierta tolerancia sobre el estándar). Con estas medidas ciertamente se comenten errores en la lectura. Si deseamos suprimir esos errores, hagamos todas ellas (excepto naturalmente la longitud de la pista, que está acotada por el tamaño del disco) 100 veces mayores. No cometeremos ya (prácticamente) errores. Pero en ningún caso podremos ni siquiera aproximarnos a los 740 Mb. de capacidad de un disco compacto en su formato actual.

En definitiva, en el estado de nuestra tecnología, manejar y almacenar los enormes flujos de datos que demanda la socidad actual requiere usar códigos correctores de errores.

Referencias

[1] V.D. Goppa, *Geometry and Codes*, Kluwer Academic Publishers, Dordrecht-Boston-Londres, 1988.

[2] J.H. van Lint, *Introduction to Coding Theory*, Springer-Verlag, Berlín-Heidelberg-Nueva York, 1982.

[3] F.J. Mac Williams, N. Sloane, *The Theory of Error-Correcting codes*, North-Holland, Amsterdam, 1977.

[4] C. Munuera, J. Tena, *Codificación de la Información*, Pub. Univ. Valladolid, Valladolid, 1997.

[5] J. Rifá, Ll. Huguet, *Comunicación Digital*, Masson, Barcelona, 1991.

[6] M. Tsfasman, S. Vladut, *Algebraic-Geometric Codes*, Kluwer Academic Publishers, Dordrecht-Boston-Londres, 1991.

Carlos Munuera
Departamento de Matemática Aplicada
Universidad de Valladolid
Avda. de Salamanca SN
47014 Valladolid
Correo-electrónico: cmunuera@modulor.arq.uva.es