CÓDIGOS CORRECTORES DE ERRORES, QUINIELAS Y EMPAQUETAMIENTOS

Pablo Fernández Gallardo
Profesor Asociado de Matemáticas
Universidad Politécnica de Madrid

El año 2000 ha resultado ser muy especial, sobre todo para la comunidad matemática: con la tranquilidad que da el comprobar que el mundo no se acababa (pese a los augurios que, en forma de cataclismos —informáticos incluso—, se anunciaban), y con el aliento de la declaración del mismo como "Año Mundial de las Matemáticas", se ha desarrollado un intenso esfuerzo para mostrar el papel de las matemáticas en el entendimiento de la realidad que nos rodea.

En este capítulo intentaremos mostrar cómo el lenguaje de las matemáticas se puede aplicar a la descripción y análisis de ciertas cuestiones "de la vida cotidiana" que, aparentemente, no guardan conexión alguna; pero que, una vez puestas en términos convenientes, no resultan ser tan dispares. Las describimos ahora de la manera más informal posible (ya formalizaremos más adelante):

- Queremos transmitir información a través de un canal de comunicación que probablemente provoca errores. El desafío es intentar detectar estos hipotéticos errores y, en su caso, corregirlos.
- Deseamos apostar a las quinielas (digamos, futbolísticas): sea por intereses lúdicos o por motivos lucrativos, nos gustaría diseñar una estrategia razonable de apuestas.
- Por último, nos interesamos por el problema de empaquetar naranjas en una caja: ¿cuál es la mejor (más eficiente) manera de hacerlo?

Variadas y variopintas, estas cuestiones son el objeto de las siguientes secciones. Confiamos en que estas páginas deparen un rato de lectura amena, proporcionen algunos conocimientos que motiven una reflexión posterior y, que, si se desea, sirvan como material aplicable a la práctica docente.

Las estrellas invitadas: las listas de ceros y unos

Los objetos que nos van a permitir describir las realidades que antes enumerábamos son las listas de ceros y unos. Ahí va un ejemplo de una lista de longitud siete, con las dos notaciones que usaremos para describirlas:

$$\boxed{0\ 1\ 0\ 0\ 1\ 1\ 1} \qquad (0\ 1\ 0\ 0\ 1\ 1\ 1)$$

Necesitamos primero saber cuántas hay con una cierta longitud, n: con estas características, podremos construir hasta 2^n listas distintas; un número enorme, en cuanto n sea un poco grande. Por ejemplo, si las listas tienen 200 posiciones, el número 2^{200} tienen más de 60 cifras decimales. Así que parece garantizado que tendremos un número suficiente para describir las cuestiones que nos interesan[1].

Sorprende comprobar el conspicuo papel que estas listas desempeñan en muchos de los fenómenos de nuestra vida: por ejemplo, la tecnología digital, de la que disfrutamos día a día al escuchar música en un disco compacto. Y no olvidemos que las listas de ceros y unos son el lenguaje que utilizan los ordenadores: ¿hay mejor publicidad que ésa? Vayamos pues, con entusiasmo, a estudiarlas.

Su geometría

Al conjunto de las 2^n listas de longitud n formadas por ceros y unos lo llamaremos a partir de ahora $\{0,1\}^n$; y lo que queremos es establecer una geometría en él (lo que haremos midiendo "distancias" en este conjunto). Y lo haremos con la idea más natural posible (que resulta ser la más adecuada).

[1] Y no conviene olvidar que un número tan grande de objetos que manejar nos debe hacer reflexionar sobre cómo se almacena y se trabaja con esa información. Volveremos sobre este aspecto más adelante.

Comencemos con un sencillo ejemplo: supongamos que tenemos tres listas de ceros y unos de longitud 5,

1	0	0	0	1		1	1	0	0	1		0	0	1	1	1

Cualquier persona diría que las dos primeras se parecen entre sí, que están más "cerca" entre ellas que con respecto a la tercera. Estamos aplicando un criterio natural para compararlas: el número de posiciones en que las listas difieren (o coinciden). Así, las dos primeras sólo difieren en la segunda posición, mientras que, en relación a la tercera, difieren en más posiciones. Vamos a formalizar esta idea, y el resultado será una distancia (en el sentido matemático del término), así que la tentación primera de usar las palabras "cerca" o "lejos" resulta ser bastante acertada.

Definición 1 *Consideremos el conjunto de todas las n-listas formadas con ceros y unos. Definimos la* ***distancia de Hamming***[2] *entre dos listas $\mathbf{x} = (x_1, \ldots, x_n)$ e $\mathbf{y} = (y_1, \ldots, y_n)$ como el número de posiciones en que ambas difieren: en fórmula,*

$$d(\mathbf{x}, \mathbf{y}) = \#\{j \,:\, 1 \leq j \leq n\,,\, x_j \neq y_j\}\,.$$

Se puede comprobar que, efectivamente, esta distancia de Hamming es una **distancia**, en el sentido matemático del término: esto es, una función que a cada par de n-listas de ceros y unos le asocia un número real (natural, en este caso), cumpliendo las siguientes propiedades:

- es no negativa: en este caso, toma valores (enteros) entre 0 y n.
- Sólo toma el valor 0 cuando calculamos la distancia entre una lista y ella misma; esto es, $d(\mathbf{x}, \mathbf{y}) = 0$ si y sólo si $\mathbf{x} = \mathbf{y}$.
- Es simétrica, esto es, $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ para cualesquiera listas $\mathbf{x}$ e $\mathbf{y}$.
- Cumple la **desigualdad triangular**: para cualesquiera listas $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ de longitud n,
$$d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})\,.$$

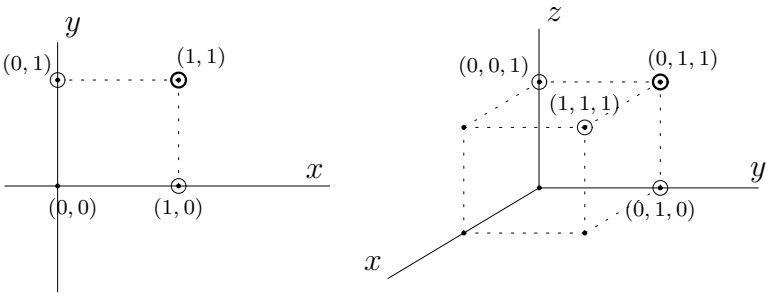
[2]Richard Hamming (1915-1998) es uno de los héroes de esta historia. Matemático de formación, trabajó en el laboratorio de Los Alamos en el proyecto de la bomba atómica, pero casi toda su carrera profesional se desarrolló en los Laboratorios Bell. Aparentemente, empezó a reflexionar sobre la posibilidad de diseñar códigos correctores de errores por motivos prácticos: los ordenadores de entonces trabajaban con cintas de papel perforadas, con los que se introducían datos y algoritmos. Si se detectaba algún error, la máquina paraba; si este contratiempo se daba entre semana, se podía corregir con la ayuda de operarios. Pero, durante el fin de semana, el trabajo simplemente se detenía y se pasaba a procesar el siguiente; el lunes siguiente se descubría, con cierta desolación, que nada se había avanzado. Motivo más que suficiente como para pararse a pensar en mejoras, ¿no?

Las tres primeras propiedades son inmediatas de comprobar, y sólo la cuarta requiere un momento de reflexión, que dejamos como ejercicio para el lector.

Por supuesto, la distancia habitual en el plano o en el espacio (la euclídea, a la que estamos tan acostumbrados) es una distancia en el sentido que acabamos de introducir[3]. Así que no es descabellado utilizar la intuición geométrica que tenemos en el plano o en el espacio. Por ejemplo, un objeto que será relevante más adelante es la bola de radio r en torno a la n-lista $\mathbf{a}$,

$$B_r(\mathbf{a}) = \{n\text{-listas } \mathbf{x} \,:\, d(\mathbf{x}, \mathbf{a}) \leq r\}\,,$$

esto es, el conjunto de las listas que difieren de $\mathbf{a}$ en no más de r posiciones. Aunque en nuestros argumentos utilizaremos habitualmente la imagen de las bolas euclídeas (en el plano) para describir estos objetos, conviene al menos dibujarlas en dos y tres dimensiones: las listas de dos y tres posiciones son las coordenadas de los vértices del cuadrado y del cubo unidad, respectivamente. En el dibujo aparecen un par de bolas de radio 1 (las listas incluidas en cada bola están rodeadas por un círculo, en negrita para el centro):



(a) $B_1(1,1)$ (b) $B_1(0,1,1)$

Nos va a interesar calcular el tamaño (número de elementos) de una de estas bolas: ¿cuántas n-listas viven en la bola de radio r (que supondremos un entero entre 1 y n) en torno a una cierta lista $\mathbf{a}$? La cuenta es sencilla: si $r = 0$, la propia lista $\mathbf{a}$; si $r = 1$, hay que contar, además, cuántas listas difieren en exactamente una posición de la original. Como sólo tendremos que cambiar una posición de la lista, bastará saber de cuántas maneras se puede elegir esa posición: por supuesto, de $\binom{n}{1}$ maneras. Para la bola de

[3]En realidad, la noción de distancia aparece en muchos contextos diferentes, y es esto lo que justifica el interés de introducir una definición general, que destile la esencia que subyace en esos ejemplos y a partir de la cual se puedan concluir propiedades comunes a todos ellos.

radio $r = 2$, habrá que contar las que difieren en exactamente cero, una y dos posiciones, y el resultado es

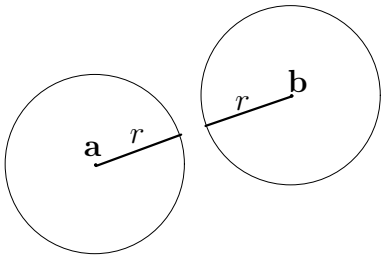
$$|B_2(\mathbf{a})| = \sum_{j=0}^{2} \binom{n}{j} = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} = 1 + n + \frac{n\,(n-1)}{2}\,.$$

Y, en general, para la bola de radio r,

$$|B_r(\mathbf{a})| = \sum_{j=0}^{r} \binom{n}{j}\,.$$

Obsérvese que la respuesta es independiente del centro $\mathbf{a}$ de la bola. Y que en cuanto $r \geq n$, en una bola de ese radio están todas las 2^n listas posibles.

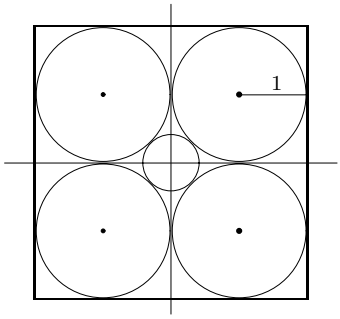
Estas bolas podrán, por supuesto, tener intersección, y aunque estimar el tamaño de estas intersecciones pueda resultar complicado en general, podemos establecer las condiciones en las que **no** tendrán intersección. Démonos dos n-listas $\mathbf{a}$ y $\mathbf{b}$ que distan d (esto es, difieren en d posiciones). Y consideremos dos bolas centradas en esas listas de radio r, donde $d \geq 2r + 1$. Entonces, las bolas son disjuntas. Si nos permitimos un dibujo "euclídeo", este resultado es obvio:



porque no podremos "acercar" las bolas lo suficiente como para que tengan intersección. ¿Y para las listas y la distancia de Hamming? Exactamente lo mismo, porque tras este argumento gráfico no hay otra cosa que la desigualdad triangular, que, como distancia que es, cumple la distancia de Hamming; así que podremos construir un argumento análogo (un ejercicio para el lector).

En lo sucesivo, recurriremos a veces a argumentos gráficos como éste, apelando a nuestra intuición geométrica euclídea. Eso sí, no nos confiemos: esa intuición nos puede jugar malas pasadas en dimensiones grandes. Vaya como ejemplo, aunque se aparte del contexto en el que vamos a trabajar,

la siguiente sorprendente construcción[4] geométrica: estamos en el plano y situamos cuatro círculos de radio 1 en los puntos de coordenadas $(\pm 1, \pm 1)$, como en el dibujo:



El círculo interior, tangente a los otros cuatro, está claramente dentro del cuadrado que los engloba. Lo mismo ocurriría con la construcción en tres dimensiones (la esfera interior quedaría dentro del cubo). Pero en dimensión n general hay que hacer un cálculo: la distancia de un punto $(\pm 1, \pm 1, \ldots, \pm 1)$ al origen es $\sqrt{n}$. Así que el radio de la esfera interior es $\sqrt{n} - 1$. Pero la distancia del origen al lado del (hiper)cubo es 2, independientemente de la dimensión. Así que (¡cielos!), a partir de $n = 9$, la bola "interior" se sale del hipercubo.

Y su Álgebra

Tenemos un conjunto (sencillo) de números, $\{0, 1\}$, y sumamos y multiplicamos estos números con una aritmética especial, la aritmética del 2, que consiste en leer el resultado que obtengamos de la siguiente manera: será 0 si el resultado es un número par, y 1 si es un número impar (volveremos sobre aritméticas como ésta más adelante). La única sorpresa que nos depara esta forma especial de operar es que, en contra de lo que mucha gente cree,

$$1 + 1 = 0\,.$$

Ahora nos situamos en nuestro conjunto favorito, el de las listas de ceros y unos de longitud n y definimos dos operaciones:

- **sumar listas**, y lo hacemos posición a posición (acordándonos de leer los resultados en la aritmética del 2):

$$(1\,0\,0\,0\,1\,1\,1) + (0\,1\,0\,1\,1\,1\,1) = (1\,1\,0\,1\,0\,0\,0)\,.$$

[4] Agradezco a Adolfo Quirós que me la haya sugerido. La construcción se puede encontrar en *What's happening in the Mathematical Science*, volumen 1 (1993), publicado por la American Mathematical Society.

- **Multiplicar listas por números**: en este caso, si multiplicamos por un 1, la lista queda tal como está, y si multiplicamos por un 0 obtenemos la lista formada únicamente por ceros. Por ejemplo,

$$1 \times (1\,0\,0\,0\,1\,1\,1) = (1\,0\,0\,0\,1\,1\,1) \qquad 0 \times (1\,0\,0\,0\,1\,1\,1) = (0\,0\,0\,0\,0\,0\,0).$$

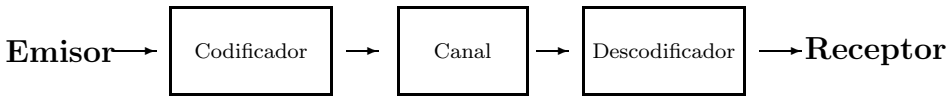
Pues bien, con estas dos sencillas operaciones lo que estamos haciendo es darle **estructura** a nuestro conjunto de las n-listas; y una estructura muy especial: obtenemos un **espacio vectorial**. Y claro, es una estructura que conocemos bien, y que por supuesto aprovecharemos.

Códigos detectores y correctores de errores

Vamos con nuestro problema: tenemos que enviar mensajes por un cierto canal en el que se pueden producir errores, y querríamos diseñar los mensajes de manera que estos errores pudieran ser detectados y, en la medida de lo posible, corregidos.

Empecemos, para fijar ideas, estableciendo un esquema general de transmisión de la información, que constaría de los siguientes elementos:

- Un **emisor**, que desea mandar un cierto mensaje.
- Un **codificador**, que transformaría el mensaje original de manera que pudiera ser transmitido (generalmente, traduciéndolo a un alfabeto binario; es decir, pasándolo a listas de ceros y unos).
- Un **canal** físico por el que se transmita el mensaje codificado (por ejemplo, ondas electromagnéticas).
- Un **descodificador**, que reciba el mensaje y sea capaz de deshacer la codificación[5] y realizar el proceso de detección y corrección de los posibles errores que se hayan originado en la transmisión del mensaje.
- Por último, el **receptor** final del mensaje.



[5]En la transmisión de información hay otros requerimientos, aparte de la fidelidad que aquí estudiaremos, que son relevantes: por ejemplo, el cifrado de mensajes, para asegurar la privacidad, o la compresión de información. No trataremos estas cuestiones aquí.

Supondremos que en todos los pasos del proceso la información se transmite mediante un alfabeto binario; esto es, los objetos que manejaremos serán, como ya hemos advertido, listas de ceros y unos[6].

Para entender algunas de las ideas que nos van a permitir detectar y corregir errores, empecemos con un par de ejemplos ilustrativos.

Primer ejemplo: DNI *versus* NIF

El Documento Nacional de Identidad, DNI, como bien sabemos, consta de ocho dígitos. Imaginemos que transmitimos uno de estos DNI, por ejemplo el 12345678. El proceso de transmisión podría consistir, simplemente, en teclear tal número para introducirlo en una base de datos de un ordenador. Un canal peculiar, "la mano que teclea", y que está expuesto a errores. Por ejemplo, podríamos equivocarnos y teclear el número 12344678. Si alguien consultara posteriormente la base de datos, no habría ninguna forma de que detectara el error, porque 123444678 es, en principio, un DNI válido.

Pero diseñemos un procedimiento mejor: el Número de Identificación Fiscal, NIF, se forma añadiendo al DNI una letra. Para obtenerla seguimos el siguiente procedimiento:

- Calculamos el valor de 12345678 en la aritmética[7] del 23 o módulo 23 (la elección del 23 no es casual, como veremos). Esto es, calculamos el resto de dividir el número 12345678 entre 23; así obtenemos el número 14. En fórmula,

$$12345678 \equiv 14 \pmod{23}.$$

[6]Un par de comentarios al respecto: la mayoría de las veces, como es obvio, el mensaje que queremos transmitir no será una lista de ceros y unos, así que debemos hacer una traducción previa (piénsese en la manera que tenemos de convertir símbolos en listas mediante el código ASCII, o en el código MORSE). El canal, del que detallaremos algunas características más adelante, se encargará de transmitir estas listas. Muchas veces el canal utilizará un medio analógico, como las ondas electromagnéticas, así que habrá procesos intermedios de modulación y desmodulación en los que no entraremos.

[7]Formalmente, hallamos la clase de congruencia módulo 23 a la que pertenece el número 12345678.

- Buscamos este número en la siguiente tabla[8] (¡que sí es arbitraria[9]!):

0	1	2	3	4	5	6	7	8	9	10	11
T	R	W	A	G	M	Y	F	P	D	X	B

12	13	14	15	16	17	18	19	20	21	22
N	J	Z	S	Q	V	H	L	C	K	E

  En nuestro caso, el 14 se corresponde con la letra Z.

Así que el NIF es $12345678\,Z$. Ahora transmitamos este NIF y supongamos que seguimos igual de torpes y cometemos un error:

$$12345678\,Z \quad \overset{\text{transmitimos}}{\longrightarrow} \quad 12344678\,Z$$

Si miramos lo que hemos recibido, podemos repetir el procedimiento y calcular el valor de 12344678 en la aritmética del 23; resulta ser 3, así que debería aparecer la letra A: ¡hemos sido capaces de detectar el error!

Uno se preguntaría si esto es casualidad, o si este sistema es siempre capaz de detectar **un** error en la transmisión: la respuesta es que sí, pero requiere un pequeño argumento[10], que explica[11] la elección del 23.

[8] Obsérvese que en ella no aparecen las letras O (se podría confundir con el 0), ni tampoco las letras I (por confundirse con el 1) y U (¿parecida a la V?). La ausencia de la Ñ tiene otra explicación, más obvia: la constante conspiración de la quinta columna antipatriótica que vive para minar las esencias patrias. Y bueno, ejem, quizás el que la Ñ no sea un símbolo internacional también ayuda.

[9] El motivo por el que se escogió esta tabla, en lugar de una más natural, como podría ser la que respeta el orden del abecedario, se me escapa (quizás alguien pueda ilustrarme). ¿Se pretendía dificultar, con esta asignación aleatoria, el "descifrado" del sistema? En todo caso, como sistema de cifrado no es gran cosa: una vez conocida la regla del 23, recuperar la tabla es tarea trivial (bastaría mirar unos cuantos DNI distintos).

[10] Ahí va: tenemos un número n de ocho dígitos (en base decimal), $n_7\,n_6\,\ldots n_1\,n_0$, esto es,

$$n = n_7 \times 10^7 + n_6 \times 10^6 + \cdots + n_1 \times 10 + n_0\,.$$

Si transmitimos $n_7 \ldots n_j \ldots n_0$ (con resto r módulo 23) y recibimos $n_7 \ldots N_j \ldots n_0$ (con resto R módulo 23), no detectaríamos el error si $r = R$. Pero esto supondría que

$$10^j\, n_j \equiv 10^j\, N_j \pmod{23}.$$

Y uno estaría tentado de cancelar el factor 10^j a ambos lados, para concluir que en realidad r no puede ser igual a R, a menos que se tratara del mismo número. ¿Se puede hacer esto?: sí, pero porque 23 es un número primo y, poniéndonos por un momento formales, todo entero que no sea múltiplo de 23 tiene inverso multiplicativo en $\mathbb{Z}_{23}$.

[11] Otra razón es que 23 está muy cercano al número de caracteres de que consta el alfabeto castellano.

Es un buen ejercicio de aritmética modular comprobar que el sistema no detecta, en general, dos errores, pero sí en un caso particular (bastante frecuente, por cierto): cuando el error consiste en trasponer dos dígitos.

Este sistema nos permite detectar errores (al menos si sólo se ha producido uno), pero, ¿y corregirlos? No puede hacerlo; el sistema no está diseñado para hacerlo. Veremos ejemplos más adelante en los que sí seremos capaces de corregir errores. Pero antes, resumamos las enseñanzas de este ejemplo:

- Podemos detectar errores porque transmitimos información **redundante**: el valor de la letra del NIF está implícito en el DNI, no es información nueva que introduzcamos.
- Con este sistema pasamos de un contexto (el mundo de los DNI) en el que cualquier combinación de dígitos están permitidas a otro (el de los NIF) en el que no todas son válidas. Por ejemplo, 123445678 Z no es, como hemos visto, un NIF verdadero.
- De hecho, "cerca" de 12345678 Z hay muchas "palabras" prohibidas: 12345678 T, 123445678 Z, etc.

Segundo ejemplo: el diccionario del castellano

Vamos con un ejemplo también revelador: tenemos un **abecedario** $a, b \dots z$ y un **diccionario** de palabras (formadas con los símbolos del alfabeto). La primera observación es que no todas las combinaciones de símbolos del alfabeto son palabras del diccionario, y esto nos va a permitir, al igual que en el ejemplo de los NIF, detectar errores.

Pero podemos ir más allá: supongamos que transmitimos la palabra "Zaragoza" y recibimos, por ejemplo, "Zatagoza". Por supuesto, detectamos que se ha producido algún error; pero aún más, cualquiera se sentiría en disposición de corregir el error: se ha producido en el tercer símbolo, y era una r en lugar de una t. La razón es clara: no hay palabras en castellano "cerca" (en el sentido de "parecidas") de Zaragoza.

Pero si transmitimos "casa" y recibimos "cusa", pese a que detectamos el error, ya no está tan claro cómo corregirlo: podríamos haber emitido lusa, musa, cuna, etc. Peor aún, podríamos haber recibido "tasa" en lugar de "casa" y ni siquiera podríamos detectar el error. La razón, la misma de antes, pero al revés: ahora hay muchas palabras semejantes a "casa".

Enumeremos las enseñanzas de este ejemplo:

- La estructura: un conjunto de símbolos (el abecedario) y unas palabras formadas con ellos (el diccionario).

- Las palabras del diccionario deben estar separadas (para detectar errores) ... y si están muy separadas, hasta nos atreveremos a corregir.

Pero también hay que señalar las "trampas" de este ejemplo:

- El diccionario del castellano es el que es: más bien habría que crear otro a partir de él (como hacíamos con los DNI y los NIF).

- El contexto de una frase puede ayudarnos a corregir errores: por ejemplo, si emitimos "en un lugar de la Tancha", cualquiera deduciría que se emitió "Mancha" (y no, por ejemplo, "Cancha", que en principio era un buen candidato). Cuando trabajemos con listas de ceros y unos, tendremos que crear un contexto que nos ayude a corregir.

El canal

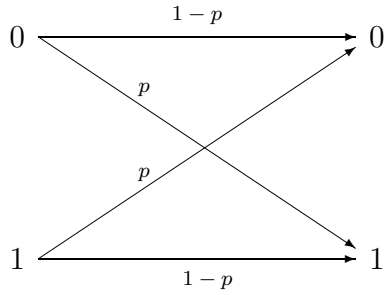
Situémonos en el mundo en el que vamos a trabajar: el de las listas de ceros y unos. Lo pertinente es definir primero cuáles son las características de nuestro canal:

- Es un canal binario: transmite ceros y unos.

- No tiene pérdida de información; es decir, si transmitimos una lista de ceros y unos con n posiciones, recibiremos con seguridad una lista de igual longitud (con los símbolos quizás alterados).

- Tiene **ruido** (¡claro!), es decir, hay una cierta probabilidad de cometer errores (cambiar un cero por un uno o viceversa).

- Pero el canal es **simétrico**, esto es, la probabilidad de cambiar un 0 por un 1 es la misma que la de cambiar un 1 por un 0. Esto supone que un único número, p, la probabilidad de digamos cambiar un 0 por un 1 (la probabilidad de "equivocarse"), describe todo el sistema:



- Y, muy importante, el que el canal cambie un símbolo de la lista es **independiente** de que cambie, digamos, el siguiente (o cualquier otro).

Un par de observaciones relevantes:

- Un único parámetro, p, un número entre 0 y 1 (que podemos suponer[12] que es $< 1/2$) define el canal. Generalmente, será un número pequeño; si no, el verdadero problema es el de mejorar el canal. De hecho, si $p = 1/2$, el canal es completamente aleatorio y nos podremos olvidar de cualquier intento de corregir errores.

- Y los errores se distribuyen aleatoriamente en el mensaje, según la probabilidad p. Podríamos imaginarnos el canal como un dispositivo que va leyendo la lista de ceros y unos y, en cada posición, lanza una moneda (con probabilidad p de que salga cara): si sale cara, cambia el símbolo, si sale cruz lo deja como está.

Por supuesto, éste es un modelo sencillo, y que muchas veces no será adecuado: por ejemplo, si tuviéramos errores de borrado (en los que, simplemente, no se transmite un símbolo), o los llamados **errores en ráfaga**, en los que una porción del mensaje está especialmente expuesta a errores (imagínese, por ejemplo, una transmisión de telefonía móvil mientras pasamos por un túnel, o mientras atravesamos una tormenta).

Primeros intentos

Nuestras palabras son (todas, en principio) las listas de ceros y unos de una cierta longitud, digamos k. El primer intento, enviarlas tal cual son, se salda con fracaso: cualquier error pasa inadvertido, porque lo recibido (coincida o no con lo emitido) es una palabra válida.

[12] Si p fuera mayor que 0.5, bastaría con cambiar los papeles de los ceros y los unos.

Un segundo intento, *à la NIF*, consistiría en añadir un **carácter de control** al final de la lista; por ejemplo, un control de paridad: añadimos un 1 si el número de unos de la lista es impar y un 0 en caso contrario. Tendríamos así el siguiente esquema de codificación:

$$\begin{array}{rcl} f: \quad \mathcal{P} = \{0,1\}^k & \longrightarrow & \mathcal{C} \subset \{0,1\}^{k+1} \\ \mathbf{p} = (x_1, \ldots, x_k) & \longrightarrow & \mathbf{c} = (x_1, \ldots, x_k, x_{k+1}) \,, \end{array}$$

de manera que $x_1 + x_2 + \cdots + x_{k+1} = 0$ en la aritmética del 2. Vamos ya reservando símbolos que utilizaremos más adelante: $\mathcal{P}$ será el diccionario original (listas de longitud k), y $\mathcal{C}$ será el **código** (en este caso, listas de longitud $k+1$). Las palabras del diccionario las nombraremos con $\mathbf{p}$ y las del código, con $\mathbf{c}$ (y éstas son las que enviaremos por el canal). La función f, la **función de codificación**, es la receta que hace corresponder a cada palabra del diccionario una del código (en este caso, añadir un último símbolo dependiendo de la paridad de la lista original).

Es un sistema idéntico al del NIF: nos va a permitir detectar un error porque añadimos información redundante (el dígito de paridad viene dado por la lista original); con ello separamos las palabras del código (la mitad de las listas de longitud $k+1$ no pertenecen a nuestro código).

Un intento mejor, en nuestro empeño de conseguir corregir errores, consistiría en enviar cada palabra de nuestro mensaje un cierto número de veces. Por ejemplo, supongamos que estamos manejando listas de longitud $k = 4$ y queremos emitir la palabra (0 0 1 1).

- Si la enviamos (repetida) dos veces, lo que estamos haciendo es codificar nuestra lista original (palabra del diccionario) con la lista (palabra del código) (0 0 1 1 0 0 1 1). Ahora la enviamos por el canal, se produce un error y recibimos

  $$\left\| 0 \;\; 0 \;\; 1 \;\; 1 \,\middle|\, 1 \;\; 0 \;\; 1 \;\; 1 \right\|$$

  Es claro que podemos detectar ese error (basta comparar las dos "mitades" de la lista recibida), pero no podremos corregirlo. Obsérvese que esta codificación duplica el tamaño de nuestras palabras, y a cambio consigue que no todas las listas de 8 posiciones sean palabras del código (pregunta: de las 2^8 listas de ocho posiciones, ¿cuántas pertenecen al código?). Esto es, "separa" las palabras código.

- Así que intentemos algo mejor: repetimos cada palabra tres veces. Esto es, codificamos la lista (0 0 1 1) como la lista (0 0 1 1 0 0 1 1 0 0 1 1). Si

ahora la enviamos y recibimos

$$\| 0\ 0\ 1\ 1 \mid 1\ 0\ 1\ 1 \mid 0\ 0\ 1\ 1 \|$$

detectamos nuevamente errores (por comparación), pero ya lanzados, nos atrevemos a corregir el error: hay dos "tercios" iguales y uno dispar, así que decidimos que el error está en la quinta posición. Hemos aplicado, simplemente, un criterio de "mayoría"; éste será, convenientemente formalizado, el procedimiento que seguiremos a la hora de corregir.

Vamos con un último ejemplo que nos será útil para entender los cálculos que haremos más adelante. Nuestro código consta únicamente[13] de dos palabras de longitud 5, (1 1 1 1 1) y (0 0 0 0 0). La clave de este diccionario estriba en que para **confundir** dos palabras del código hace falta cometer cinco errores. Ahora, supongamos que recibimos las siguientes listas:

$$(1\,0\,0\,0\,0) \quad (1\,1\,0\,0\,0) \quad (1\,0\,0\,1\,1)\,.$$

En todas ellas se han cometido errores, pues no son palabras del código. Es claro que este código nos permite detectar hasta cuatro errores (uno menos del número de posiciones que hay que cambiar para confundir palabras código). ¿Y corregir? Bien, uno apostaría a que, en el caso de la primera lista, se había emitido (0 0 0 0 0) y que se ha cometido un error en la primera posición (parece más probable que haya sucedido eso, en lugar de que se hubiera emitido la lista de unos y se hubieran producido cuatro errores). De la misma manera, diríamos que la segunda lista proviene también de la lista de ceros. Para la tercera, sin embargo, el mejor candidato sería la de unos; supongamos, por un momento, que en realidad hubiéramos transmitido la de ceros: se habrían producido tres errores, que corregiríamos equivocadamente (si empleamos este criterio, aún impreciso, de "más probable"). Así que con este código podemos corregir hasta dos errores (por cierto, la "mitad" de cinco).

Detección y corrección de errores

Vamos a formalizar las ideas expuestas en los ejemplos anteriores: tenemos el conjunto de palabras originales, (todas las) listas de longitud k con ceros y unos; y las codificamos construyendo, a partir de ellas, listas de longitud

[13] Visto de otra manera, nuestras palabras originales son, simplemente, listas de una posición, y transmitimos cada símbolo 5 veces. Esto es, cada símbolo va codificado por una de las listas de longitud 5 del código.

n, con una cierta receta que llamaremos f, la **codificación**:

$$\begin{array}{rccl} f: & \mathcal{P} = \{0,1\}^k & \longrightarrow & \mathcal{C} \subseteq \{0,1\}^n \\ & \mathbf{p} = (p_1, \ldots, p_k) & \to & f(\mathbf{p}) = \mathbf{c} = (c_1, \ldots, c_n) \end{array}$$

Hay un parámetro importante que deberemos tener en cuenta: se trata de la **tasa de transmisión** del código $\mathcal{C}$, que es la razón entre la longitud de las palabras antes y después de ser codificadas:

$$R_{\mathcal{C}} = \frac{k}{n}\,.$$

O, si queremos, $1 - R_{\mathcal{C}}$, lo que se llama la **redundancia** del código. Por supuesto, interesa que $R_{\mathcal{C}}$ sea cercano a 1, para que los mensajes no se alarguen excesivamente. Pero claro, añadir poca redundancia a la hora de codificar va a ir en contra de nuestro objetivo de detectar y corregir muchos errores: habrá que buscar un equilibrio.

Empecemos analizando la **capacidad de detección** que tiene un código como el descrito antes. Como nos sugería el último ejemplo de la sección anterior, esta capacidad vendrá dada por el número de posiciones que hay que cambiar para confundir palabras del código. Pero justo la distancia de Hamming viene dada por el número de posiciones en que difieren dos listas, así que será la herramienta que utilizaremos. En concreto, necesitamos definir la **distancia mínima** en el código,

$$d_{\mathcal{C}} = \min\{(\mathbf{x}, \mathbf{y})\}\,,$$

donde el mínimo se calcula en todos los pares de palabras $\mathbf{x}$ e $\mathbf{y}$ (distintas) del código. Este número (que está entre 1 y n, claro) nos informa de la menor distancia de Hamming que pueden guardar dos palabras del código. Sin más aditamentos, tenemos el resultado que buscábamos:

Teorema 1 *Un código $\mathcal{C}$ con distancia mínima $d_{\mathcal{C}}$ puede detectar hasta $d_{\mathcal{C}} - 1$ errores.*

Porque si transmitimos una palabra del código y no se cometen más de $d_{\mathcal{C}} - 1$ errores, nunca podremos recibir otra palabra del código.

Vayamos con el proceso de corrección; ahora tendremos que definir con cuidado los ingredientes: partimos de palabras $\mathbf{p} = (p_1, \ldots, p_k)$ que codificamos, mediante cierta regla f, como palabras código $\mathbf{c} = (c_1, \ldots, c_n)$.

Enviamos una palabra del código por el canal y recibimos una cierta lista de n posiciones, $\mathbf{x} = (x_1, \ldots, x_n)$:

$$\mathbf{p} = (p_1, \ldots, p_k) \xrightarrow{f} \mathbf{c} = (c_1, \ldots, c_n) \longrightarrow \boxed{\textbf{canal}} \longrightarrow \mathbf{x} = (x_1, \ldots, x_n)$$

Necesitamos

- un criterio que nos permita decidir qué palabra código habíamos transmitido realmente.

- Una vez tomada esta decisión, aplicamos la regla de codificación en sentido contrario, f^{-1}, para recuperar la lista de longitud k original.

Y el criterio que vamos a emplear para el primer paso es bastante natural, el del **vecino más próximo**: si recibimos $\mathbf{x}$, le asignamos la[14] palabra del código $\mathbf{c}$ que cumpla que $d(\mathbf{x}, \mathbf{c}) < d(\mathbf{x}, \mathbf{c}')$ para cualquier otra palabra $\mathbf{c}'$ del código.

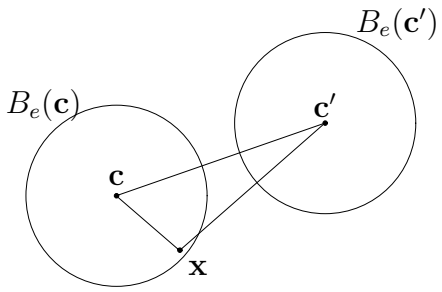
Ahora podemos cuantificar la **capacidad de corrección** de nuestro código.

Teorema 2 *Un código $\mathcal{C}$ con distancia mínima $d_{\mathcal{C}}$ puede corregir, aplicando el criterio de vecino más próximo, hasta* $\left\lfloor \dfrac{d_{\mathcal{C}} - 1}{2} \right\rfloor$ *errores*[15].

Para demostrar este resultado apelaremos al argumento geométrico que dábamos al principio de estas notas. Supongamos, por comodidad, que $d_{\mathcal{C}} = 2e+1$. Transmitimos $\mathbf{c}$ y recibimos $\mathbf{x}$ y no se han producido más de e errores. El criterio de vecino más próximo nos permite recuperar $\mathbf{c}$, si es que no hay otras palabras del código a distancia $\leq e$ de $\mathbf{x}$. Pero ya habíamos visto que esto no podía suceder; limitémonos a reproducir la imagen euclídea (tras la que se oculta el correspondiente argumento basado en la desigualdad triangular) para convencernos:

[14] Quizás el uso de "la" palabra sea demasiado optimista, porque tal palabra podría bien no ser única. En estos casos, se suele obviar esa porción del mensaje, o pedir que se reemita.

[15] El símbolo $\lfloor\ \rfloor$ que aparece en el teorema es el **suelo**: el suelo de un número es el mayor entero que es menor igual que el número dado, así que coincide con la conocida "parte entera", $[\ ]$. Hoy en día se prefiere utilizar el suelo porque tiene su (obvia) contrapartida: el **techo**, $\lceil\ \rceil$, definido como el menor entero que es mayor o igual que el número.

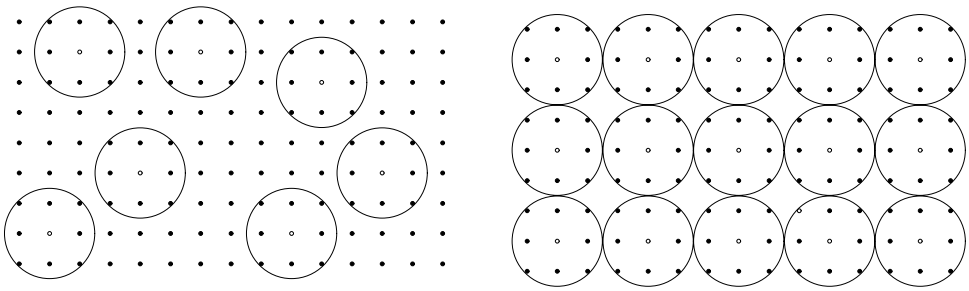


Una observación casi inmediata es que, si separamos mucho las palabras de nuestro código, no cabrán muchas. La pregunta es: si un código $\mathcal{C}$ (de listas de longitud n) tiene distancia mínima $d_{\mathcal{C}} = 2e + 1$, ¿cuántas palabras, como máximo, puede haber en $\mathcal{C}$? La respuesta es la llamada **cota de Hamming**:

$$|\mathcal{C}| \sum_{j=0}^{e} \binom{n}{j} \leq 2^n \,,$$

que limita el valor que puede tener $|\mathcal{C}|$, el número de palabras que forman el código $\mathcal{C}$. A la derecha estamos contando el número total de listas de longitud n; y a la izquierda, el número de listas que hay en las bolas de radio e en torno a las palabras del código (que, como la distancia mínima es $2e+1$, ¡son disjuntas!).

Observemos ahora los dos siguientes dibujos, que pretenden describir la geometría de dos códigos distintos de distancia mínima $2e + 1$:



Los puntos dibujados son todas las listas de longitud n, y los centros de las bolas de radio e que aparecen, las palabras del código. Ambos cumplen que las bolas de radio e son disjuntas dos a dos, así que corregirán hasta e errores. Pero en el de la derecha, además, cada lista de $\{0,1\}^n$ está en una única bola de radio e centrada en alguna palabra del código. Esto es muy útil, porque, recibamos la lista que recibamos, siempre que no se hayan cometido más de e errores, podremos recuperar la palabra código transmitida.

Estos códigos se llaman **códigos perfectos**, y se caracterizan porque se cumple la igualdad en la cota de Hamming. Claro que, hasta ahora, sólo hemos exhibido un dibujo pretendidamente descriptivo, pero, ¿existen realmente estos códigos perfectos? La respuesta es que sí, aunque no son muy abundantes. El ejemplo más sencillo lo forman los **códigos de repetición** (impar), que ya hemos visto en alguna ocasión: en el conjunto de las listas de longitud n, con n un número impar, digamos $n = 2e + 1$, consideramos el código formado (únicamente) por las listas

$$(0\,0\,0\,\ldots\,0) \quad \text{y} \quad (1\,1\,1\,\ldots\,1)\,.$$

Este código tiene distancia mínima $2e + 1$, así que detecta hasta $2e$ errores y corrige hasta e errores. Y un sencillo cálculo nos lleva a convencernos de que es un código perfecto[16].

Uno podría iniciar la búsqueda y captura de candidatos a ser códigos perfectos utilizando la cota de Hamming: para que tengamos la igualdad, n y e han de cumplir que

$$\sum_{j=0}^{e} \binom{n}{j}$$

(es decir, la suma de los primeros $e+1$ números de la fila n en el triángulo de Pascal) es una potencia de 2. Y uno puede imaginarse que esto sólo ocurrirá de casualidad. Y tan de casualidad, ¡no ocurre casi nunca! (aparte de los casos obvios, $e = 0$ y $e = n$). Por ejemplo, si $e = 1$ (códigos que corrigen un error), entonces n ha de ser de la forma $n = 2^t - 1$, para cierto entero t. Hay toda una familia de códigos perfectos, los t-**códigos de Hamming binarios**, que cumplen esta condición.

Si tomamos $e = 2$, el primer valor en que ocurre es $n = 5$ (el código de repetición que veíamos antes). Y el siguiente (y último) es $n = 90$. Pues bien, a pesar de que para estos parámetros podríamos tener un código perfecto, la realidad es que tal código no existe. Recordemos las condiciones de esta búsqueda: para que exista el código perfecto necesitamos igualdad en la cota de Hamming; pero luego hay que encontrar el código, ¡y a veces no lo hay!

[16]Recordemos que podíamos entender este código como la repetición, $2e + 1$ veces, de cada símbolo (cero o uno). Ajá, por eso era mejor repetir tres veces que dos. Es un excelente código a la hora de detectar y corregir, pero tiene un inconveniente importante: las palabras se hacen muy largas, la longitud del mensaje se hace $2e + 1$ veces más grande.

El caso $e = 3$ es muy interesante: con $n = 7$ tenemos el código de repetición correspondiente, y el siguiente caso es $n = 23$. El hipotético código constaría de 2^{12} palabras y corregiría hasta tres errores. Pues de hipotético, nada; suenen los clarines y demos la bienvenida a uno de los personajes famosos de esta historia: el **código de Golay**[17] **(binario)**. Sobre su construcción y su historia pueden consultarse las fuentes bibliográficas. Lo sorprendente del caso es que, aparte de los mencionados, ya no hay más códigos perfectos (lo que refuerza el carácter casi mágico del de Golay, que volverá a aparecer más adelante)

Si reflexionamos un poco sobre todas las consideraciones hechas hasta ahora, observamos que hemos hecho un análisis bastante completo de las propiedades que tiene un código, basándonos en argumentos geométricos con la distancia de Hamming. Pero hemos pasado por alto algunas dificultades:

- En principio, un código $\mathcal{C}$ es una colección grande de listas de longitud n. Y manejar toda esa información puede ser complicado.
- Y antes incluso: ¿cómo se construye un código de éstos?, ¿cómo definir la función de codificación? ¿Acaso simplemente listando las palabras originales y sus correspondencias en el código?
- Otro problema: calcular la distancia mínima en un código puede no ser tarea sencilla: hay que evaluar $\binom{|\mathcal{C}|}{2}$ distancias, ¿no?
- La aplicación del criterio de vecino más próximo también requiere evaluar un montón de distancias.

Y es que hasta aquí sólo hemos utilizado la geometría (la distancia) de las listas de ceros y unos. ¿Qué tal si le damos algo de estructura a nuestros códigos?

Añadimos estructura: los códigos lineales

El Álgebra Lineal suele ser un quebradero de cabeza para los alumnos, tanto si son alumnos de secundaria como de primeros cursos de Universidad: utiliza un lenguaje al que no están muy acostumbrados y que requiere un cierto nivel de abstracción. Pero, además, con las especificaciones que generalmente se utilizan (sobre el cuerpo de los reales), uno tiene a veces dificultades para

[17] Golay, ingeniero eléctrico suizo, publicó en 1949 un artículo de media página en el que exhibía, sin apenas justificación, sus famosos códigos. Trabajó también en la generalización de los códigos que Hamming había inventado.

captar la atención con aplicaciones a la ya famosa "vida cotidiana" (sobre esta cuestión, ¡no olviden consultar la referencia [CO]!).

El contexto de los códigos permite presentar una preciosa aplicación de esta materia (aunque sea sobre un cuerpo finito), que quizás ayude en esa tarea. Al fin y al cabo, el que estas técnicas sirvan para poder escuchar *Compact Discs* sin interferencias, o recibir transmisiones desde sondas espaciales, apelan a territorios cuando menos atractivos.

Vamos con ello: vimos que el conjunto $\{0,1\}^n$ de las listas de longitud n de ceros y unos formaban, con las operaciones de suma y multiplicación por escalares que allí describíamos, un espacio vectorial, que se suele nombrar como $\mathbb{F}_2^n$. Y en cuanto tenemos un espacio vectorial aparecen los conceptos de subespacio vectorial, base, dimensión, etc.

Un **código lineal** $\mathcal{C}$ es, simplemente, un subespacio vectorial de cierta dimensión, k, en $\mathbb{F}_2^n$. Esto es, de entre las 2^n listas de $\mathbb{F}_2^n$, elegimos 2^k que formen un espacio vectorial; serán las palabras seleccionadas para el código. ¿Y cómo hacemos esto? Basta con tomar una **base**, esto es, k listas linealmente independientes (obsérvese, sólo k, no 2^k), $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x}^k$. Con ellas formamos la llamada **matriz generatriz** del código, listando las coordenadas de cada elemento de la base en las filas[18] de la matriz:

$$G = \begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^k & x_2^k & \cdots & x_n^k \end{pmatrix}$$

Y se llama generatriz, por supuesto, porque genera todas las palabras del código (todas las combinaciones lineales de los elementos de la base),

$$(p_1, \ldots, p_k) \cdot G = (c_1 \ \ldots, c_n)\,.$$

¡Pero ésta es la codificación que buscábamos!, la regla para transformar todas las listas de longitud k en las 2^k palabras del código: la regla f es, simplemente, multiplicar por G, y la receta inversa, f^{-1}, vendrá dada por la matriz inversa de G.

La otra manera de describir un subespacio vectorial, además de con la base, es mediante ecuaciones: las condiciones que han de cumplir las

[18] Éste es el convenio habitual en el mundo de los códigos, aunque en otros contextos se listan en las columnas.

coordenadas para que el vector pertenezca al subespacio. En este contexto, esta descripción se recoge en la llamada **matriz de control**, H:

$$\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C} \quad \Longleftrightarrow \quad H \cdot \mathbf{x}^t = \mathbf{0}^t \,.$$

Pues bien, la matriz de control facilita enormemente la tarea de calcular la distancia mínima en un código lineal, así como permite diseñar un procedimiento (descodificación por **síndromes**) para la aplicación del criterio de vecino más próximo (los detalles pueden consultarse en [MT]).

Así que la estructura lineal[19] nos permite solventar satisfactoriamente la mayoría de las dificultades que enumerábamos anteriormente. Por supuesto, hay muchos otros tipos de códigos, más allá de los lineales, algunos basados en herramientas matemáticas sofisticadas. Pero no entraremos en ellos; el lector interesado en otros códigos puede consultar la bibliografía.

Antes de pasar a otro asunto, hagamos un par de breves repasos a unos asuntos interesantes: un análisis probabilístico de la eficacia de los códigos correctores y una somera explicación del funcionamiento de los *Compact Discs*.

Un análisis probabilístico

El análisis hecho hasta aquí ha podido probablemente convencer al lector de la utilidad de los códigos correctores, pero quizás no haya sido suficiente para exhibir su verdadera potencia. Creo que los siguientes cálculos lograrán ese objetivo.

Los ingredientes: tenemos, como siempre, palabras de longitud k, que codificamos con listas de longitud n. El código $\mathcal{C}$ consta de un cierto número de palabras, $|\mathcal{C}|$, y tiene distancia mínima $2e + 1$, así que corrige hasta e errores. El canal por el que transmitimos se "equivoca" con probabilidad p. Como advertimos en su momento, los errores se distribuyen aleatoriamente (según p) en el mensaje (recordemos la imagen de la moneda lanzada para cada símbolo transmitido).

[19] Obsérvese que conseguimos esta estructura de espacio vectorial porque el conjunto $\{0, 1\}$ con la aritmética módulo 2 tiene unas propiedades especiales: es un cuerpo (como lo son $\mathbb{R}$ ó $\mathbb{C}$). Pero en general, si tenemos más de dos símbolos no vamos a tener esta estructura de cuerpo (y, por tanto, adiós a los espacios vectoriales); en concreto, sólo lo podremos hacer cuando el número de símbolos sea una potencia de un primo. En particular, no vamos a tener códigos lineales con un alfabeto decimal, con diez símbolos.

Queremos transmitir un cierto número, h, de palabras:

- Si lo hacemos con las palabras originales, transmitiremos $k\,h$ símbolos; y la probabilidad de que el mensaje llegue correctamente será

$$prob(\text{transmisión correcta de } h \text{ palabras}) = (1-p)^{k\,h}\,,$$

porque necesitaremos que todos los símbolos se transmitan sin errores.

- Sin embargo, si nos ayudamos de la codificación, tendremos que enviar $n\,h$ símbolos (las palabras, tras la codificación, tienen n símbolos cada una). Ahora

$$prob(\text{trans. correcta de } \mathbf{una} \text{ palabra}) \geq prob(\text{cometer} \leq e \text{ errores})\,.$$

Pero la probabilidad de cometer no más de e errores en una palabra de longitud n viene dada por

$$\sum_{j=0}^{e}(1-p)^{n-j}\,p^j\,\binom{n}{j}\,,$$

puesto que hay que decidir, para evaluar esta probabilidad, en qué j posiciones (donde $0 \leq j \leq e$) de la lista se producen los errores, y luego medir la probabilidad de este suceso. Así que

$$prob(\text{transmisión correcta } h \text{ palabras}) = \left(\sum_{j=0}^{e}(1-p)^{n-j}\,p^j\,\binom{n}{j}\right)^h\,.$$

Quizás unos números nos ayuden a descubrir la diferencia de los dos sistemas: digamos que en la codificación transformamos listas de longitud 4 en listas de longitud 7, y que el código corrige un error (a cambio, casi doblamos la longitud de las palabras). Y que enviamos un mensaje de 400 ceros y unos (100 palabras) por un canal con $p = 0.01$:

- Sin codificar, el mensaje se transmite correctamente con probabilidad $(0.99)^{400} = 0.02$; esto es, un 2% de las veces.

- Pero si utilizamos nuestro sistema de codificación, la probabilidad de transmisión correcta del mensaje es

$$\left(0.99^7 + 7 \times 0.99^6 \times 0.001\right)^{100} = 0.81\,.$$

Así que se transmite correctamente en un 81% de las ocasiones.

Una mejora impresionante, ¿no?, y sólo duplicando el tamaño de nuestro mensaje.

Este tipo de argumentos probabilísticos permiten analizar algunos sistemas de juegos sencillos; por ejemplo, expliquemos brevemente el sistema con que se juegan los partidos de tenis. Por simplificar, consideremos un híbrido de tenis y ping-pong: somos A y jugamos contra B, a 21 puntos. Gana el primero que consiga 11 puntos (sin diferencia). Ganamos cada punto con probabilidad p.

Si hacemos el análisis en términos del número j (entre 11 y 21) de puntos que sumamos en las partidas ganadoras, obtenemos que

$$prob(A \text{ gana partido}) = \sum_{j=11}^{21} \binom{21}{j} p^j \, (1-p)^{21-j} \, .$$

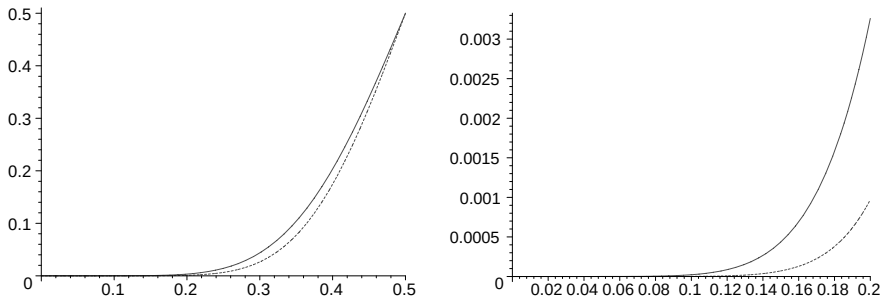
Pero ahora imaginemos que el partido se disputa con otro sistema: hay tres mangas de siete puntos cada una; gana una manga quien consiga cuatro puntos, y gana el partido aquél que gane dos mangas. Entonces,

$$prob(A \text{ gana manga}) = \sum_{j=4}^{7} \binom{7}{j} p^j \, (1-p)^{7-j} = S \, .$$

Y, por tanto,

$$prob(A \text{ gana partido}) = \sum_{j=2}^{3} \binom{3}{j} S^j \, (1-S)^{3-j} \, .$$

¿A quién favorece el sistema de mangas, al mejor o al peor jugador? La respuesta es que favorece al peor jugador, como nos muestran las siguientes gráficas, donde aparece la probabilidad de ganar en función de p (la línea punteada representa al sistema directo, la continua al de mangas):

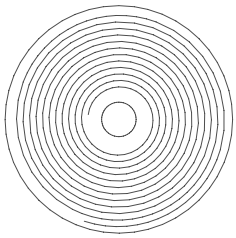


En la de la izquierda aparece el rango de 0 a 0.5 (más allá de $p = 0.5$, la gráfica es simétrica, cambiando los papeles del mejor y peor jugador). En la de la derecha aparece una ampliación del rango de 0 a 0.2, para observar con más detalle lo favorable que es el sistema con mangas para el peor jugador.

Los códigos de los discos compactos

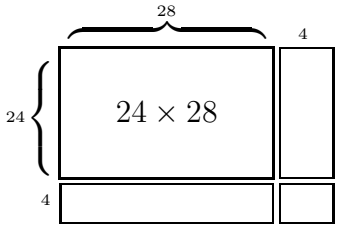
Los discos compactos, *CD*, han revolucionado en pocos años los sistemas de almacenamiento de música y datos, desterrando a los entrañables discos de vinilo. Quizás, alguno de los lectores se ha preguntado cómo es posible que, a pesar de que la superficie del disco esté seriamente dañada (o manchada, con huellas dactilares, por ejemplo), la calidad del sonido siga siendo alta. La clave es que una gran parte de la información almacenada en el disco es redundante, pues se usan potentes códigos correctores en el almacenamiento. Aquí intentaremos dar una idea aproximada (muy simplificada) de las características de estos códigos.

Un CD no es sino una larguísima espiral (de unos cinco kilómetros de longitud) sobre la que se graban una sucesión de "valles" y "llanuras", que representan ceros y unos para el lector láser. Cada segundo de música se corresponde con varios millones de *bits* (ceros o unos) sobre esta pista. Aunque la probabilidad de que el lector láser se equivoque sea baja, digamos $p = 0.001$, se podrían producir muchos (demasiados) errores por segundo.



Hay varias técnicas que permiten solventar estas dificultades; para hacernos una idea de ellas, veamos un ejemplo sencillo: disponemos de un código lineal $\mathcal{C}_1$ de parámetros $[28, 24]$ (longitud de las listas 28, dimensión del subespacio 24) con distancia mínima cinco (así que corrige hasta dos errores). Y otro código lineal, $\mathcal{C}_2$ de parámetros $[32, 28]$, con la misma distancia mínima.

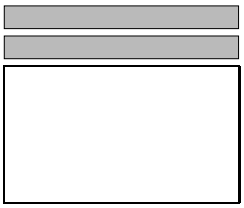
Queremos codificar una larguísima lista de ceros y unos con estos códigos; es claro que cada uno de ellos por separado no va a ser suficiente para nuestros propósitos (sobre todo si hay errores en ráfaga, algo muy habitual en un CD; piénsese en una arañazo sobre la pista). La clave va a ser combinarlos (en lo que se llama un **código producto**). Tomamos una sucesión de 24×28 símbolos y los disponemos en forma de matriz de esas dimensiones. A partir de ella formamos una nueva matriz



de manera que las filas sean palabras de $\mathcal{C}_2$ y las columnas, palabras de $\mathcal{C}_1$; para ello basta con utilizar las respectivas reglas de codificación (y comprobar que las codificaciones son compatibles en la esquina inferior derecha).

Estas matrices son las palabras de un nuevo código (que resulta ser lineal con la operación obvia de suma de matrices) que tiene distancia mínima 25 (queda esto como ejercicio). Por tanto, corrige hasta 12 errores. Por cierto, el nuevo código transforma cada 24×28 símbolos en 28×32.

¿Cómo corrige este tipo de código un error en ráfaga? Veamos un ejemplo: transmitimos los 28×32 símbolos por filas y las dos primeras se transmiten mal:



El código $\mathcal{C}_2$, el que se aplicaría a las filas, no basta para resolver esta situación. Pero, leídas por columnas, cada palabra sólo contiene dos errores; y el código $\mathcal{C}_1$ puede corregirlos sin problemas.

En un CD se utilizan códigos correctores que siguen estas ideas, aunque son más sofisticados. Con ellos se consigue corregir un número grande (del orden de millares) de errores consecutivos sobre la pista. Los discos compactos son, sin duda, una brillante simbiosis de tecnología y Matemáticas.

De cómo alguien que sepa Matemáticas ha de apostar a las quinielas

Las quinielas futbolísticas y ciertas estrategias que se pueden seguir a la hora de apostar en ellas son otro ejemplo de cómo algunas ideas expuestas hasta

aquí se pueden aplicar a realidades de nuestro entorno.[20] Antes de comenzar este análisis, un *caveat*: que nadie espere encontrar aquí la receta mágica para acertar los 14 cada semana; si tal receta obrara en mi poder, señores, probablemente no estaría escribiendo estas líneas[21].

Una apuesta a la quiniela futbolística es una lista de 14 posiciones, en cada una de las cuales podemos situar tres símbolos, $\{1, X, 2\}$. Hay 3^{14} posibles listas de éstas, en torno a los cinco millones. Y como, en principio, cualquier lista puede ser la ganadora, la única estrategia que nos puede garantizar los 14 aciertos (no entraremos aquí en los posibles premios de pleno al 15) consiste en rellenar tantos boletos como posibilidades hay; es claro que ésta no es una estrategia muy razonable.

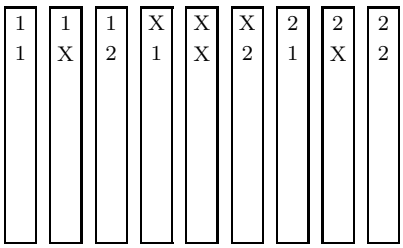
La pregunta es si hay estrategias alternativas que resulten adecuadas. Existen por ahí las llamadas "peñas quinielísticas", que publican llamativos anuncios en la prensa, intentando captar inversores; una búsqueda en la red me ha permitido encontrar los divertidos nombres de dos de ellas:

"La Timba" y "El Pelotazo" ... los nombres sugieren que se trata de un juego, futbolístico, pero además parece caber la posibilidad de hacer dinero con ellas. Esta última posibilidad, más allá de motivos lúdicos, es, por supuesto, la razón de ser de estas peñas. En algunas de las páginas web de estas peñas se exhiben tablas con los resultados obtenidos en años anteriores; en una de ellas, por ejemplo, se asegura que durante la temporada 1999/2000 consiguieron un beneficio del 32% sobre lo invertido, un margen nada desdeñable si consideramos otras opciones de inversión. Los números concretos revelan algunas cuestiones que luego aparecerán en nuestro análisis: tales márgenes de beneficios se consiguen con inversiones grandes, en torno a los 30 millones por temporada en el caso citado. Y, lo que es más importante, se consiguieron sin que en toda la temporada se obtuviera premio alguno de 14.

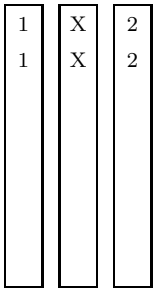
[20] Un título alternativo para esta sección podría ser: *Cuántas matemáticas saben y hacen quienes se toman en serio pensar en cómo apostar a las quinielas.*

[21] O quizás sí, que uno es un caballero.

Así que parece que hay estrategias lo suficientemente eficaces como para que merezca la pena intentar entender alguno de sus ingredientes. Empecemos con un ejemplo sencillo; de los 14 partidos, tenemos dudas en el resultado de dos de ellos, digamos los dos primeros. Para asegurarnos el acierto (suponiendo que en los restantes 12 partidos se dan los resultados que intuimos), parece que la única posibilidad es la de rellenar nueve boletos:



Pero observemos que, con sólo tres apuestas, por ejemplo las que mostramos aquí, garantizamos al menos **un** acierto en los dos partidos en los que teníamos dudas; es decir, 13 aciertos si es que nuestra intuición era correcta para el resto (y, con un poco de suerte, los 14 aciertos). Hemos conseguido reducir el número de apuestas, a cambio de conformarnos con garantizar un segundo premio. Por supuesto, la clave es que cualquier par de resultados difiere, como mucho, en una posición de alguna de estas tres.



Ya tenemos los ingredientes: la estrategia ha de ir dirigida a garantizar un segundo premio (o quizás un tercero, cuarto, etc.); y el ejemplo de la peña que antes comentábamos nos indica que ése es el camino (recordemos que su beneficio se había conseguido a base de acertar treces, doces, ...). El lector perspicaz ya se habrá dado cuenta (listas que difieren en no más de un cierto número de posiciones) de que las herramientas que nos van a permitir diseñar tales estrategias tienen mucho que ver con las ideas geométricas que hemos descrito en secciones anteriores.

Vamos a formalizar todas estas ideas: los objetos con los que vamos a trabajar son las listas de n posiciones (el número de partidos en los que tengamos dudas) formadas con tres símbolos, digamos $\{0, 1, 2\}$, por ponernos en plan matemático. Ahora nos hemos salido de nuestro mundo de las listas de ceros y unos, así que debemos recapacitar sobre las nociones que habíamos introducido; por ahora sólo las geométricas[22], que estaban basadas

[22] Aunque en las listas con tres símbolos también podríamos construir un Álgebra Lineal, que utilizaremos más adelante, porque 3 es un número primo, y $\mathbb{Z}_3$ es un cuerpo.

en la noción de distancia de Hamming. ¿Existe tal concepto en este nuevo contexto?; por supuesto, y con la misma definición: la distancia entre dos listas de longitud n formadas con $\{0,1,2\}$ es, de nuevo, el número de posiciones en que difieren.

Si ahora consideramos el conjunto de todas las n-listas con estas características (hay 3^n de ellas), lo que llamaremos $\{0,1,2\}^n$, nuestro objetivo es elegir unas cuantas de ellas de manera que cualquier otra esté en al menos una bola de radio 1 (recordemos, como mucho han de diferir en una posición, si queremos garantizar un segundo premio) en torno a alguna de las seleccionadas. ¡Queremos cubrir $\{0,1,2\}^n$ con bolas de radio 1! Es éste un objetivo un poco distinto del que teníamos a la hora de construir códigos, porque aquí no nos preocupa, en principio, que esas bolas tengan intersección.

Supongamos que S es una elección de listas (centros de las bolas) para la que tal cubrimiento se produce. ¿Cuál es el tamaño mínimo de S? Hay que contar cuántas listas viven en la bola de radio 1 centrada en una lista cualquiera, $\mathbf{x}$; y el resultado es

$$|B_1(\mathbf{x})| = \sum_{j=0}^{1} \binom{n}{j} 2^j \,.$$

Porque ahora no sólo hay que decidir qué posiciones cambiamos, sino también determinar a qué símbolo cambiamos (hay dos posibilidades, claro); de ahí el factor extra 2^j. Y como hemos supuesto que las bolas de radio 1 centradas en las listas de S cubren todo $\{0,1,2\}^n$, se tendrá que

$$|S| \sum_{j=0}^{1} \binom{n}{j} 2^j \geq 3^n \,.$$

Esta cota, que es análoga a la de Hamming, pero desde el punto de vista contrario (se llama **cota de cubrimiento**), nos dice el número de listas que ha de tener, como mínimo, S:

$$|S| \geq \frac{3^n}{1+2n} \,.$$

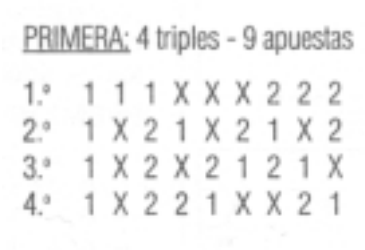
Para cada n, tenemos aquí una cota inferior para el número de listas que se requieren para cubrir todo el espacio. Por ejemplo, para $n = 2$, la estimación es que $|S| \geq 9/5$, pero se comprueba que tres apuestas son necesarias (como veíamos en el ejemplo del principio).

Para $n = 3$, tenemos $|S| \geq 27/7 \approx 3.86$. Así que podría ocurrir que cuatro listas bastaran para cubrir; pero no es el caso, son necesarias al menos 5. Por ejemplo, las 27 posibles listas están a no más de distancia 1 de al menos una de las siguientes:

$$(0\,0\,0) \quad (1\,0\,1) \quad (1\,1\,0) \quad (0\,1\,1) \quad (2\,2\,2)$$

El caso $n = 4$ parece interesante, porque ahora $|S| = 81/9 = 9$, así que cabe la posibilidad de tener lo que llamaríamos un cubrimiento perfecto (en el que no hay intersecciones entre las bolas). Volvamos un momento al mundo de los códigos: ¿qué códigos ternarios perfectos hay? Pues no muchos: toda una familia de códigos de Hamming (ternarios), que corrigen un error y, como no podía ser menos[23], un código de Golay ternario[24], que resulta ser lineal (recordemos que hay un Álgebra Lineal en este contexto, pues 3 es un número primo) de parámetros $[11, 6]$ (longitud de las listas 11, dimensión del espacio vectorial 6) que corrige dos errores.

Pero es que si nos vamos a las tablas de apuestas autorizadas de las quinielas (ahora hablaremos sobre ellas), encontramos: primera reducción autorizada, cuatro triples, nueve apuestas. Traducido a nuestro lenguaje, ¡es justo el código de Hamming correspondiente! Así que tampoco convenía olvidarnos de las construcciones lineales en este mundo con tres símbolos. Ya que las hemos mencionado, expliquemos en qué consiste eso de las "apuestas autorizadas". Si uno quisiera implementar una estrategia de apuestas siguiendo las ideas que estamos exponiendo, probablemente se vería en la necesidad de rellenar un número grande boletos. El Organismo Nacional de Loterías y Apuestas del Estado, ONLAE, facilita esa labor proporcionando una serie de apuestas establecidas. Así, por ejemplo, si queremos cubrir todos los posibles resultados (en la jerga de las quinielas, apostar triples) de cuatro partidos, basta con señalar la casilla correspondiente a la primera reducción autorizada (la de la figura). Apuestas autorizadas semejantes existen para triples en más de cuatro partidos, y para


PRIMERA: 4 triples - 9 apuestas

1.ª	1	1	1	X	X	X	2	2	2
2.ª	1	X	2	1	X	2	1	X	2
3.ª	1	X	2	X	2	1	2	1	X
4.ª	1	X	2	2	1	X	X	2	1

[23]Ni más, con más de tres símbolos ya no encontraremos códigos tan sorprendentes como los de Golay binario y ternario.

[24]No deja de ser curioso que este código apareciera en una revista finlandesa de quinielas en 1947, un par de años antes de que Golay publicara sus trabajos. No hace falta decir que Golay no era consciente de este hecho, y que este código, en esta su primera aparición, sólo lo hacía en calidad de buen cubrimiento, no como código corrector de errores.

combinaciones de triples y dobles (de los que hablaremos luego). Uno podría preguntarse cómo de eficaces son las estrategias que facilita el ONLAE. Vamos a ver que no siempre son la mejor elección.

Por supuesto, una vez definido el problema matemático, el de cubrir $\{0,1,2\}^n$ con bolas de radio 1, existen numerosos resultados sobre qué es lo mejor que podemos esperar. Con técnicas matemáticas variadas, se han conseguido cotas inferiores (más allá de la cota trivial de cubrimiento) y cotas superiores (construyendo cubrimientos explícitos, bien generados por ordenador, bien con argumentos combinatorios) sobre cubrimientos para cada valor de n. La siguiente tabla muestra los mejores valores conocidos hasta el momento para los primeros valores de n:

4	5	6	7	8	9	10	11	12	13
9	27	63-73	150-186	393-486	1048-1356	2818-3645	7767-9477	21395-27702	59049

Por ejemplo, para $n = 8$, se sabe que no se puede cubrir con menos de 393 bolas, y que con 486 sí se puede hacer. En algunos casos se tiene certeza sobre el mejor cubrimiento posible, como en $n = 13$, porque se tiene un código perfecto. Y ahora comparemos esta tabla con las apuestas que nos ofrece el ONLAE

	TRIPLES							
	4	5	6	7	8	9	10	11
0	9	27	81	243	729	2.187	6.561	19.683

para comprobar que, a partir de $n = 6$, la estrategia que se nos ofrece no es, ni mucho menos, la mejor posible. Antes de lanzarnos a llamar a una asociación de consumidores para protestar por el aparente engaño, observemos los números que aparecen en la lista de la ONLAE: son, simplemente, la primera reducida (la de 9 apuestas), multiplicada por tres las veces que corresponda. Y es que no se nos ofrece una estrategia específica para cada n, sino que, simplemente, se construye el cubrimiento para cuatro partidos y luego se apuesta a todos los posibles resultados de los restantes encuentros.

¿Y si uno quisiera seguir una estrategia más adecuada? Primero deberíamos conocerla; esto es, deberíamos buscar la fuente (artículos matemáticos, o quizás revistas de quinielas) en la que ese mejor cubrimiento esté descrito. Y segundo (y no menos importante), contar con la infraestructura técnica necesaria para rellenar un número tan grande de boletos.

Variaciones sobre el tema

En la jerga quinielística se maneja también el concepto de "doble", como ya hemos comentado. Y es que a veces no estamos seguros del resultado de unos cuantos partidos, pero sí intuimos que uno de los posibles (por ejemplo, el 2) no va a salir. Así que sólo querríamos cubrir dos de los tres posibles resultados.

Nuestros objetos de interés son ahora listas de $n+m$ posiciones; en las n primeras podemos situar cualquiera de los tres símbolos (triples), y en las m restantes, únicamente dos (dobles). Hay $3^n\,2^m$ listas de éstas.

La maquinaria es la misma que antes, sólo que ahora, por ejemplo, la cota de cubrimiento (con bolas de radio 1) es más complicada:

$$|S| \sum_{j=0}^{1} \sum_{k=0}^{1} \binom{n}{j} \binom{m}{k} 2^j \geq 3^n\,2^m \quad \Longrightarrow \quad |S| \geq \frac{3^n\,2^m}{1+2n+m}\,.$$

Por ejemplo, para 7 dobles ($n=0, m=7$), la cota exige que haya al menos 16 apuestas; justo las que se requieren en la apuesta reducida correspondiente (ver figura). Si apostamos con tres triples y tres dobles, necesitaríamos al menos 21.6 listas; pero se sabe que 24 son necesarias, y ése es también el número de apuestas recogido en la reducida correspondiente:

Pero más allá de estos ejemplos, la estrategia de la ONLAE no es la mejor posible.

Una observación que debemos hacer es que, en ambos esquemas (sólo triples o permitiendo también dobles), nuestra estrategia se basa en cubrir todo el "espacio" de las 3^n o $3^n\,2^m$, según el caso, posibles listas. Pero quizás haya algunas que haya que descartar *a priori*. Algunas estrategias razonables son las siguientes:

- Cubrir (con bolas de radio 1) una bola $B_r(\mathbf{x})$, de radio r y centrada en una lista $\mathbf{x}$. Expliquémoslo: estamos convencidos de que saldrá una cierta combinación de resultados, por ejemplo, todo victorias caseras,

y admitimos la posibilidad de que no sea así en un cierto número de partidos: por ejemplo, que salgan no más de 6 variantes (nombre en la jerga para la X y el 2). En algún caso, hasta podríamos centrar la bola en una combinación de resultados muy poco probable[25].

- O el caso contrario: cubrir todo el espacio menos una cierta bola (por ejemplo, centrada en la combinación en la que todo son victorias visitantes).

¿Cómo trabajan de verdad las peñas quinielísticas? Por supuesto, tienen medios técnicos suficientes como para implementar las estrategias que decidan. Y capital suficiente como para afrontar un gran número de apuestas.

Pero, ¿qué estrategias siguen? No tengo esa información, que supongo será secreto de sumario. Pero se intuye que añaden un ingrediente que no hemos tenido en cuenta hasta aquí (sólo parcialmente en estas últimas consideraciones): a saber, que no todos los resultados son igualmente probables. Quizás lo más razonable fuera asignar a cada posible lista una cierta probabilidad (basándose, por ejemplo, en datos históricos, o en análisis del estado de forma, clasificación, rendimiento fuera y en casa, de cada equipo en ese momento), y construir una estrategia de cubrimiento teniendo en cuenta esa asignación de probabilidades. Por supuesto, animo al lector a que reflexione sobre estas cuestiones, que quizás le permitan descubrir maneras eficientes de apostar.

En todo caso, y a modo de conclusión: si quiere apostar inteligentemente a las quinielas ... use las matemáticas.

[25] Al menos una peña cuya página *web* visité, basaba su estrategia en la búsqueda de uno de esos dos o tres premios multimillonarios que se dan todas las temporadas.

Empaquetamientos

Analicemos, finalmente, el problema del empaquetamiento de naranjas que presentábamos al comienzo de estas notas. O, mejor que naranjas, balas de cañón, que en esos términos planteaba el problema Sir Walter Raleigh a principios del siglo XVII: ¿cuál es la manera más eficiente de apilar balas de cañón en, digamos, la cubierta de un navío? En 1611, Kepler sugirió que la manera más eficaz de hacerlo era la que cualquiera utilizaría al apilar naranjas en una caja (la que aparece en el dibujo). Este aserto dio lugar a la famosa **conjetura de Kepler**[26], que explicaremos en un momento.

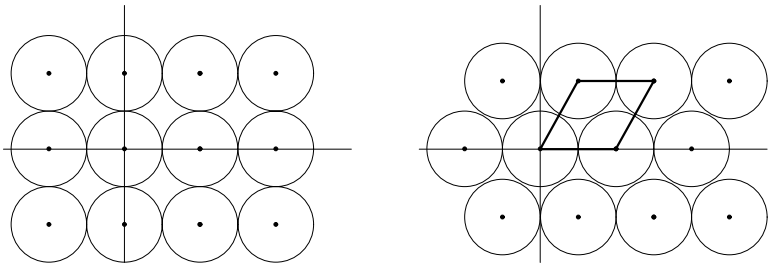
Estamos en el espacio euclídeo de n dimensiones, $\mathbb{R}^n$, y nos preguntamos por la manera en que deberemos situar (hiper)esferas de un cierto radio para ocupar el mayor espacio posible. En principio, pese a que estamos hablando también de "cubrir", este problema es distinto de los vistos hasta aquí: se trata de cubrir un espacio infinito, y la geometría es la inducida por distancia es la euclídea habitual. Aún así, veremos que hay una sorprendente conexión entre esta cuestión y el mundo de los códigos.

Tenemos una esfera S de radio r (y volumen $V_n \cdot r^n$, donde V_n es el volumen de la esfera n-dimensional de radio 1) y tomamos una sucesión infinita de puntos, $\{a_n\}$, los centros de las esferas. Tendremos un empaquetamiento si las esferas centradas en esos puntos son disjuntas dos a dos. La densidad de un empaquetamiento será la fracción de espacio ocupada por las esferas (por supuesto, hay una definición más formal de esta noción[27]).

[26] Esta conjetura fue incluida en la famosa lista de los 23 problemas que Hilbert propuso en el segundo Congreso Internacional de Matemáticos celebrado en París en 1900. En concreto, era una parte del decimoctavo problema, que incluía otras cuestiones sobre empaquetamientos. Muy recientemente, Hales ha anunciado una prueba del resultado que, muy en la línea de otras recientes demostraciones (como la del teorema de los cuatro colores), se apoya en el uso del ordenador para la verificación de un número grande de configuraciones.

[27] Y no es sencilla; nótese que se trata de una razón entre dos cantidades infinitas, el espacio total y el ocupado por las esferas. En este sentido, conviene reflexionar un momento sobre las características del problema: en principio, hay infinitos empaquetamientos posibles, y los argumentos que han llevado a la demostración final de Hales se basan en reducir el problema a la verificación de un número finito (aunque quizás muy grande) de casos.

Unos empaquetamientos especiales son aquéllos en los que, si sumo dos vectores correspondientes a dos centros, obtengo un punto que también es centro de una esfera (los centros forman un **retículo**). Por ejemplo, en dos dimensiones,

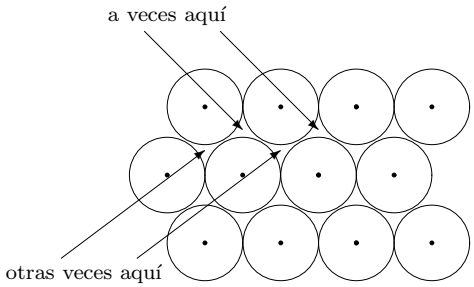


El empaquetamiento de la izquierda de círculos de área π (radio 1) situados en los puntos del retículo generado por los vectores $(0, 2)$ y $(2, 0)$ tiene densidad $\pi/4 \approx 0.7853$ (es un argumento geométrico elemental: dibújese el cuadrado cuyos vértices son cuatro centros contiguos del retículo y mídase qué área tienen las porciones de círculo que incluye). El de la derecha corresponde al retículo generado por $(2, 0)$ y $(1, \sqrt{3})$; y es fácil comprobar que tiene densidad $\pi/(2\sqrt{3}) \approx 0.9068$ (obsérvese que el paralelogramo dibujado contiene el equivalente de un círculo completo). Se sabe que éste es el empaquetamiento más denso posible.

Y a partir de él (superponiendo capas), se puede construir un empaquetamiento en tres dimensiones, cuya densidad es 0.7404. Pues bien, la conjetura de Kepler[28] afirma que este empaquetamiento (cúbico centrado en las caras, se denomina) es el óptimo en tres dimensiones.

Gauss ya sabía que estos dos empaquetamientos (en dos y tres dimensiones) eran los más densos de entre los reticulares. Pero quizás no debamos olvidar a los no reticulares (pese a que los primeros son innegablemente atractivos, por aquello del orden, la simetría). Por ejemplo, en tres dimensiones puede haber empaquetamientos no reticulares tan densos como el propuesto por Kepler: basta con ir colocando las capas de naranjas alternando (aleatoriamente) la fila de "huecos" sobre los que se sitúan. En el dibujo aparece la primera capa (vista "desde arriba") y las filas de huecos en las que colocar las siguientes:

[28] De la que muchas veces se afirma que "es algo que algunos matemáticos creen, y que todos los físicos saben."



De hecho, se sospecha que, en dimensiones altas, los empaquetamientos más densos no son los reticulares.

Vayamos con la conexión con los códigos correctores: para cada dimensión n, existen cotas teóricas para la densidad del mejor empaquetamiento posible. En la siguiente tabla mostramos estas cotas teóricas, junto con las densidades de los mejores empaquetamientos conseguidos (y el porcentaje de la cota teórica que suponen). Se observa que, conforme crece la dimensión, es cada vez más difícil rellenar eficazmente el espacio con esferas y que, por supuesto, para dimensiones altas se suele estar lejos de las cotas teóricas correspondientes:

n	1	2	3	4	$\cdots$	21	22	23	24	$\cdots$
teórica	1	0.9068	0.7404	0.6477	$\cdots$	0.006177	0.004549	0.003344	0.002455	$\cdots$
conseguida	1	0.9068	0.7404	0.6168	$\cdots$	0.002465	0.002127	0.001905	0.001929	$\cdots$
%	100	100	100	95.2	$\cdots$	39.9	46.7	56.9	78.5	$\cdots$

Pero, ¡oh, sorpresa!, nos encontramos que para dimensión 23 (y, sobre todo, para dimensión 24), el empaquetamiento está mucho más cerca de la cota teórica de lo que cabría suponer. ¿Y qué objeto que vivía en dimensión 23 ha aparecido en estas notas?, por supuesto, el código binario de Golay. En realidad, a partir de él se puede construir un código (extendido) en dimensión 24. Y este código (atención, son listas de ceros y unos) es el que se utiliza en la construcción del sorprendente empaquetamiento en $\mathbb{R}^{24}$ (es un trabajo de Leech, de los años 60).

El código binario de Golay tiene otras conexiones fascinantes, a diseños combinatorios, a teoría de grupos ... Pero, como dijo Andrew Wiles cuando escribió la última línea en la pizarra culminando, al fin, la prueba del Último Teorema de Fermat: "creo que lo dejaré aquí".

LECTURAS RECOMENDADAS

Un buen texto en castellano sobre codificación de la información (incluyendo códigos correctores, códigos compresores y sistemas de cifrado) es

[MT] Munuera, Carlos y Tena, Juan: *Codificación de la información.* Universidad de Valladolid. 1997.

Los capítulos 9 y 10 del (excelente) libro

[CO] *COMAP. las Matemáticas de la vida cotidiana.* Addison-Wesley/UAM. 1999.

tratan distintos tipos de códigos. Por cierto, los restantes 20 capítulos, aunque traten sobre otras cuestiones, son también muy recomendables. Algunos ejemplos de códigos correctores (junto con reflexiones para el aula) aparecen en

[EC] Espinel, M. C., Caballero, P.: La Matemática que protege de errores a los números de identificación. *Suma*, volumen 20 (1995) páginas 77-84.

También es recomendable el siguiente artículo divulgativo:

[LV] Lachaud, G., Vladut, S.: Los códigos correctores de errores. *Mundo Científico*, volumen 161 (octubre 1995), páginas 864-868.

En la dirección `http://pass.maths.org.uk/issue3/codes` puede encontrarse el artículo de Richard Finch "Coding Theory: the first 50 years", publicado en la revista *Plus Magazine* y que explica, incluyendo fotografías, cómo se usan los códigos correctores en los viajes espaciales.

En `www.spatula.net/proc/barcode/index.src` aparece la información sobre distintos tipos de códigos de barras (EAN, Postnet, etc.), junto con la posibilidad de generar las correspondientes imágenes.

Algunos textos en inglés son los siguientes:

[Ba] Bailys, John. *Error-correcting Codes. A Mathematical Introduction.* Chapman & Hall Matehamtics. 1998.

[Li] Lint, Jacobus H. van. *Introduction to coding theory.* Springer-Verlag. 1992.

[Hi] Hill, R. *A First Course in Coding Theory.* Oxford University Press. 1986.

[Th] Thompson, Thomas M. *From error-correcting codes through sphere packings to simple groups*. The Mathematical Association or America, The Carus Mathematical Monographs no. 21. 1983.

Este último título contiene numerosos detalles sobre el nacimiento de estos códigos, así como sobre la controversia sobre la paternidad de algunos de ellos (entre Golay y Hamming). Por último, citamos a continuación los artículos originales de estos dos autores:

[Ha] Hamming, R. W. Error Detecting and Error Correcting Codes, *Bell System Tech. J.*, 29 (1950).

[Go] Golay, M. J. E. Notes on Digital Coding, *Trans. IRE (IEEE)*, 37 (1949).

En cuanto a la aplicación de estas ideas a las quinielas, el artículo

[HHLO] Hämälainen, Honkala, Litsyn, Östegaard. *Football Pools —A Game for Mathematicians*, American Mathematical Monthly, Aug-Sep 1995,

junto con las referencias que contiene, puede servir como base para que cada uno construya sus propias estrategias de apuestas. *Vale.*