Información y Entropía de Shannon

Pablo Lessa

10 de octubre de 2014

1. Información y Entropía de Shannon

Claude Shannon (1916 a 2001) era un tipo muy interesante. Por ejemplo, hizo construir una maquina del tamaño de una caja de zapatos con una sola palanca que decía 'On-Off' (originalmente en 'Off') y con el título "The Ultimate Machine". Cuando el usuario movía la palanca para prender la maquina, la maquina empezaba a hacer ruido por unos segundos, hasta que se abría una tapa de la cual salía una mano mecánica que volvía a mover la palanca a Off. Muchos lo han imitado desde entonces dando lugar a un montón de videos en youtube con variantes de esta "Ultimate machine".

También hizo cosas menos importantes como fundar la teoría de la información. Claro que atribuirle la creación de una teoría es una simplificación grosera (hubo otra gente que hizo aportes significativos antes y después de Shannon) pero lo cierto es que, como mínimo, escribió un hermoso artículo en 1948 que ha sido ampliamente citado y, más importante aún, está lleno de ideas y frases interesantes como "El hecho de que la redundancia del idioma inglés sea del orden de 50 % hace posible la existencia de los crucigramas, si fuera del orden de 33 % uno podría crear crucigramas tridimensionales, etc" (esta frase aún no la entiendo... pero me gusta).

Vamos a discutir un caso de esta teoría en lo que sigue.

1.1. Un problema de comunicación

Imaginemos un sistema de comunicación absurdo que consiste en lo siguiente: Hay una casilla con dos ventanillas, una de "entrada" y otra de "salida". Dentro de la casilla hay un tipo encargado de transmitir un mensaje que le llega por la ventana de entrada a la de salida. Las reglas son las siguientes: Por la ventanilla de entrada le llega un texto en castellano a ritmo de una letra o caracter (incluyendo espacios, comas, etc) por segundo, y el tipo tiene derecho a mandar dos dígitos binarios (= bits = 0 o 1) por segundo por la segunda ventanilla. Puede usar el esquema que quiera para codificar el mensaje en ceros y unos pero lo tiene que declarar de antemano y la gente que recibe la tira de ceros y unos a la salida tiene que poder decodificarlo y obtener el mismo mensaje exacto que entró.

La pregunta es la siguiente: ¿Será posible que el tipo se mantenga a la par con el mensaje que va llegando o se le formará una 'fila de espera' de letras que aún no ha podido enviar?

Una reflexión inicial hace parecer que la tarea es imposible. Porque a la entrada hay al menos 27 símbolos que pueden llegar cada segundo, pero cada segundo sólo puede enviar uno de las 4 posibilidades $00, 01, 10$ o 11 por la ventanilla de salida.

Sin embargo muchas combinaciones de símbolos de la entrada o bien no son válidas según las reglas del castellano o aún siendo válidas se espera que lleguen con muy poca frecuencia (ejemplo, es razonable asumir que ' ffasaafadsjlñdf ' no va llegar muy seguido). Por lo tanto se podría inventar un esquema por el cual las letras o palabras más frecuentes del idioma castellano reciban códigos más cortos que los infrecuentes. ¿Puede funcionar un esquema de este tipo? ¿Cómo se puede modelar matemáticamente este problema?

1.2. Capacidad de un canal de comunicación

En el problema planteado hay dos 'canales de comunicación' que son la ventana de entrada y la de salida. Por la de entrada llega un símbolo de una lista finita A con más de 27 elementos (letras del idioma castellano, puntuaciones, etc) por segundo. Por el de salida salen un uno o un cero cada medio segundo.

La capacidad de un canal de comunicación se define como

$$C = \lim_{t \to +\infty} \frac{\log_2(M(t))}{t}$$

donde $M(t)$ es el número de mensajes diferentes que se pueden enviar por el canal en t segundos. La capacidad se mide en bits (dígitos binarios) por segundo. Esto es consistente con la forma en que se mide la velocidad de las conexiones a internet hoy en día (por ejemplo la de mi casa puede enviar 20 Mega bits por segundo, lo cual equivale a más de $2^{20\text{millones}}$ mensajes diferentes que pueden ser enviados en un segundo).

En nuestro problema de ejemplo la capacidad del canal de salida es 2 bits por segundo, y el de entrada es mayor a $\log_2(27) \approx 4{,}75\ldots$ bits por segundo. Esta fue la observación que hacía pensar que el problema de comunicación no tiene una solución satisfactoria en este caso. Sin embargo la capacidad del canal no es lo único que importa.

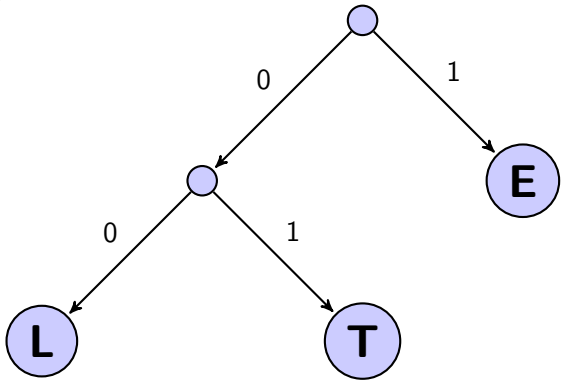
1.3. Códigos sin prefijos

Para codificar el mensaje vamos a admitir cualquier estrategia que asigne a cada bloque de m letras de A (donde m es un número finito) una tira finita de ceros y unos y que cumple la siguiente propiedad: **ninguna tira asignada es prefijo de otra**. Por ejemplo si a la letra 'e' (la más frecuente en los textos en castellano) se le asigna el código 0, entonces a todas las demás letras se les debe asignar un código que empiece con 1.

Un ejemplo clásico de códigos sin prefijos son los códigos telefónicos de los países. Si quiero llamar a Uruguay empiezo digitando 598 en el teléfono y ningún otro país empieza así, 595 es Paraguay, Argentina es 54 (y ningún otro país empieza con 54), Brasil es 55 (y ningún otro país empieza con 55), Colombia es 57, etc. Es decir, no a todos los países se les asigna códigos de la misma longitud, pero ningún código asignado es prefijo de otro (esto permite decodificar sin ambigüedad).

Estaremos interesados solamente en códigos binarios, es decir funciones $f : A^m \to \{0,1\}^*$ (donde $\{0,1\}^*$ es el conjunto de sucesiones finitas de ceros y unos) restrictas a la condición de ser libres de prefijos.

Los códigos binarios sin prefijos pueden representarse como árboles binarios con raíz. Las hojas del árbol son los códigos que se asignan. Por ejemplo en la figura que sigue está representado un código que asigna a la letra E el código 1, a la T el código 01 y a la L el código 00. La palabra 'tele' se codificaría como 011001. Para decodificar una tira de ceros y unos empezamos en la raíz del árbol yendo a la izquierda con cada cero y la derecha con cada 1, al llegar a una hoja anotamos la letra y volvemos a la raíz.



Ejercicio 1 (Desigualdad de Kraft)**.** *Mostrar que si* $f : A^m \to \{0,1\}^*$ *es un código binario sin prefijos entonces*

$$\sum_{a_1,\ldots,a_m \in A} 2^{-|f(a_1,\ldots,a_m)|} \leq 1$$

donde usamos $|\cdot|$ *para la longitud de una cadena de ceros y unos. Sugerencia: A cada cadena de ceros y unos puede asignarsele de forma natural un subintervalo de* $[0,1]$ *con extremos diádicos (e.g.* $1 \mapsto [0,1/2]$ *y* $0 \mapsto [1/2,1]$*). Alcanza con mostrar que los intervalos asociados a las hojas de un árbol binario siempre representan intervalos con interiores disjuntos (pueden compartir un punto pero no más).*

1.4. La fuente

Encaramos nuestro problema de comunicación desde el punto de vista del tipo en la casilla. Desde este punto de vista el mensaje de entrada es desconocido

a priori. Sabemos que va ser una tira de letras de un alfabeto finito A. Por lo tanto podemos modelar el espacio de mensajes de entrada posibles como $\Omega = A^{\mathbb{N}}$ (tiras infinitas de letras de A).

El hecho de que no sabemos qué mensaje va venir hace razonable que modelemos "la fuente del mensaje" (i.e. la gente que decide que mensaje tenemos que enviar) como una probabilidad μ en Ω.

En su artículo Shannon discute varias posibilidades (cada vez más realistas) para μ.

La primera es la medida μ_0 que cumple $\mu([a_1, \ldots, a_n]) = 1/|A|^n$ para todo cilindro $[a_1, \ldots, a_n]$ en Ω. Este modelo corresponde a pensar que las letras vienen elegidas al azar, independientes unas de otras, y además son todas igual de frecuentes.

Un modelo un poco más realista es la medida μ_1 que cumple $\mu([a_1, \ldots, a_n]) = p(a_1) \cdots p(a_n)$ donde p es una probabilidad en A que intenta modelar las frecuencias relativas de cada letra o símbolo en un texto castellano "medio". Es decir en este modelo las letras vienen al azar pero al menos vienen en las frecuencias "correctas" que uno esperaría de un texto en castellano.

Un modelo aún más realista es una medida Markoviana $\mu_{p,q}$ donde p es una probabilidad en A que modela las frecuencias relativas de las letras y símbolos en castellano, y $(q(a,b))_{a,b\in A}$ es una matriz de probabilidades que modela la frecuencia con la cual una letra b sigue a la letra a en un texto castellano típico. Una cadena típica para esta medida tendrá las frecuencias correctas para las letras y para los pares consecutivos de letras.

Esta idea puede generalizarse. Una medida de Markov de orden finito m es una probabilidad en $A^{\mathbb{N}}$ que cumple

$$\mu([a_1, \ldots, a_n]) = p(a_1, a_2, \ldots, a_m) q(a_1, \ldots, a_{m+1}) \cdots q(a_{n-m}, \ldots, a_n)$$

donde p es una probabilidad en A^m y $q : A^{m+1} \to [0,1]$ es una "matriz" de probabilidades que modela la frecuencia con la cual un símbolo a_{m+1} sigue a la secuencia $a_1, \ldots, a_m$ en un texto castellano "típico".

Para englobar estas opciones y otras **vamos a modelar la fuente del mensaje como una probabilidad μ en $A^{\mathbb{N}}$ que es shift invariante y ergódica** (notemos que las medidas Markovianas si bien pueden no ser invariantes a la larga se comportan como tales en condiciones muy generales como vimos en la sección anterior).

1.5. Entropía de una fuente

Supongamos un caso simplificado en el cual el alfabeto de entrada A es igual al de salida $\{0,1\}$ pero la medida μ en $A^{\mathbb{N}}$ es una medida μ_p donde $p(0) = 0{,}99$ y $p(1) = 0{,}01$. Esto modela que las 'letras' de la entrada se eligen tirando repetidas veces una moneda que tiene 9 veces más chances de caer sobre un lado que sobre el otro.

El código más ingenuo que podríamos producir es simplemente retransmitir los 'bits' que nos van llegando. Con este código por cada n bits de entrada producimos exactamente n de salida.

Sin embargo, aprovechando que la mayor parte de los dígitos de la entrada van a ser 1 podemos hacer algo mejor. Supongamos que codificamos cada tripleta de bits de la entrada según la siguiente tabla:

$$\begin{array}{llll} 111 \mapsto 1, & 011 \mapsto 011, & 101 \mapsto 010, & 110 \mapsto 001, \\ 100 \mapsto 00011, & 010 \mapsto 00010, & 001 \mapsto 00001, & 000 \mapsto 00000. \end{array}$$

Ejercicio 2. *Dibujar el árbol binario asociado al código anterior. Probar que, usando este código, para μ_p casi toda secuencia $(a_1, a_2, \ldots) \in \{0,1\}^{\mathbb{N}}$ se producen en promedio aproximadamente* 0,93 *bits de salida por cada bit de salida. Es decir*

$$\lim_{n\to+\infty} \frac{|f(a_1,a_2,a_3)| + \cdots + f(a_{3n-2}, a_{3n-1}, a_{3n}}{3n} \approx 0{,}93$$

donde $f : A^3 \to \{0,1\}^*$ es el código y usamos $|\cdot|$ para denotar la longitud de una cadena finita de ceros y unos. ¿Podés encontrar un código más eficiente que este?

La pregunta básica para responder es ¿Cuál es el límite de la eficiencia con la cual se puede codificar una fuente dada μ? Responderemos en breve.

1.6. Información

El primer paso para responder a la pregunta recién planteada es dar una medida de cantidad de información contenida en un evento de probabilidad p. Por ejemplo, en el caso de una moneda con 90 % de chances de salir cara si alguien tira la moneda y nos informan que salió cara nos están dando menos información que si nos informan que salió cruz.

Queremos definir una función $I(p)$ que mida la información contenida en un evento de probabilidad p. Intuitivamente esperamos que $I(0) = +\infty$ (un evento imposible nos da infinita información), $I(1) = 0$ (un evento casi seguro no nos da información), y $I(p)$ es decreciente. Si además agregamos la hipótesis $I(pq) = I(p) + I(q)$ (esto corresponde a que si nos informan de dos eventos independientes la cantidad total de información recibida es la suma) las únicas funciones posibles son $I(p) = -\log_b(p)$.

Para ser coherentes con cómo estamos midiendo la capacidad de canales, elegimos la base $b = 2$, y consideramos que los valores de I están dados en bits.

1.7. Entropía de una probabilidad en un conjunto finito

Sea μ una probabilidad en un conjunto finito A (por ejemplo μ modela tirar una moneda una única vez). Para cada $a \in A$ hay una cantidad de información $I(\mu(a))$ asociada. El valor esperado de esta cantidad de información es la entropía de μ, es decir

$$H(\mu) = \sum_{a\in A} \mu(a) I(\mu(a)).$$

Ejercicio 3. *Para $A = \{0,1\}$ y $\mu(1) = p, \mu(0) = 1-p$ graficar $H(\mu)$ en función de p.*

Ejercicio 4. *Demostrar que* $0 \leq H(\mu) \leq \log_2(|A|)$ *y que la entropía es nula si y sólo si* $\mu(a) = 1$ *para algún* $a \in A$ *y que es máxima si y solamente si* $\mu(a) = 1/|A|$ *para todo* $a \in A$.

Ejercicio 5. *Mostrar que si* μ *es una probabilidad en un conjunto finito* X *entonces*

$$H(\mu) = \sum_{x \in X} \mu(x) I(\mu(x)) \leq \sum_{x \in X}^{n} \mu(x) I(q_x)$$

para toda $q : X \to [0, 1]$ *con* $\sum q_x \leq 1$.

1.8. Entropía de Shannon

Ahora supongamos que μ es una medida shift invariante en $A^{\mathbb{N}}$ donde A es un conjunto finito. Definimos $H_n(\mu)$ como la entropía de la proyección de μ sobre las primeras n coordenadas, es decir

$$H_n(\mu) = \sum_{a_1,\ldots,a_n \in A} \mu([a_1, \ldots, a_n]) I(\mu([a_1, \ldots, a_n])).$$

La entropía asintótica o entropía de Shannon de la fuente μ está dada por

$$h(\mu) = \lim_{n \to +\infty} \frac{1}{n} H_n(\mu).$$

Ejercicio 6. *Calcular la entropía de Shannon de una sucesión infinita de tiradas de una moneda que sale cara con probabilidad* p.

Ejercicio 7. *Mostrar que para cualquier medida shift invariante* μ *se tiene* $0 \leq H_{m+n} \leq H_m + H_n$ *para todo* $m, n \in \mathbb{N}$. *Demostrar que esto implica que existe el límite* $h(\mu)$ *y es igual a* $\inf\{\frac{1}{m} H_m\}$.

1.9. Teorema de codificación de Shannon

El siguiente teorema muestra la importancia del teorema de Shannon para la teoría de comunicación.

Teorema 1 (Teorema de Codificación de Shannon)**.** *Sea* μ *una medida shift invariante y ergódica en* $A^{\mathbb{N}}$ *donde* A *es un conjunto finito. Para cualquier* $\epsilon > 0$ *existe un código binario sin prefijos de forma que* μ *casi toda frecuencia se puede transmitir usando este código usando en promedio* $H(\mu) + \epsilon$ *bits por letra. Además para todo código binario sin prefijos existe un conjunto de* μ *medida positiva de secuencias que se codifican usando más de* $H - \epsilon$ *bits por caracter.*

En nuestro problema de comunicación original que era transmitir un texto en castellano a un ritmo de 2 bits por segundo, si asumimos que la fuente es bien modelada por una probabilidad shift invariante, el problema se reduce a saber si la entropía de Shannon del idioma castellano (o de una cadena de Markov de orden grande que aproxime un texto típico) es mayor o menor a

2 bits por caracter. Las estimativas empíricas (basadas en sacar estadísticas de textos reales) dan que un buen modelo para el castellano tiene entropía de entre 1,2 y 1,5 bits por letra. O sea que en principio el problema original que planteamos tiene una solución en forma de código binario sin prefijos.

La demostración del teorema de codificación de Shannon se basa en el siguiente teorema ergódico que demostraremos en la siguiente sección.

Teorema 2 (Shannon, McMillan, y Breiman)**.** *Sea μ una probabilidad shift invariante y ergódica en $A^{\mathbb{N}}$ donde A es un conjunto finito. Entonces para μ casi toda sucesión $(a_1, \ldots, a_n, \ldots)$ se cumple*

$$\lim_{n\to+\infty} \frac{1}{n} I\left(\mu([a_1, \ldots, a_n])\right) = h(\mu).$$

Notemos que como corolario se obtiene lo siguiente

Corolario 1 (Equipartición asintótica)**.** *Sea μ una probabilidad shift invariante y ergódica en $A^{\mathbb{N}}$ donde A es un conjunto finito. Para todo $\epsilon > 0$ se cumple*

$$\lim_{n\to+\infty} \mu\left(\left\{(a_1, a_2, \ldots) : 2^{-(h+\epsilon)n} \leq \mu([a_1, \ldots, a_n]) \leq 2^{-(h-\epsilon)n}\right\}\right) = 1$$

donde $H = h(\mu)$.

Es decir para n grande la unión de todos los cilindros de largo n cuya probabilidad **no es** del orden de $2^{-h(\mu)n}$ tiene probabilidad muy chica. Esto es la propiedad básica que permite mostrar el teorema de codificación.

Demostración del teorema de codificación. Mostremos primero que se pueden crear códigos binarios sin prefijos que logran codificar μ-casi toda tira usando un número de bits por letra tan cercano a H como se quiera.

Para este propósito dado $\epsilon > 0$ y m digamos que un cilindro $[a_1, \ldots, a_m]$ es 'bueno' si cumple

$$2^{-(H+\epsilon)m} \leq \mu([a_1, \ldots, a_m]) \leq 2^{-(H-\epsilon)m}.$$

Por equipartición asintótica la masa de los cilindros buenos es $1 - \delta_m$ donde $\delta_m \to 0$ cuando $m \to +\infty$.

Para cada m hay menos de $2^{(H+\epsilon)m}(1-\delta_m)$ cilindros buenos y hay menos de $|A|^m$ cilindros en total. Por lo tanto, podemos crear un código binario sin prefijos $f_m : A^m \to \{0,1\}^*$ que asigna códigos de largo menor o igual a $2 + (H+\epsilon)m$ a los cilindros buenos y menor a $2+\log_2(|A|)m$ a los malos (e.g. los códigos malos empiezan con cero y tienen todos el mismo largo, los buenos empiezan con uno y tienen todos el mismo largo).

Por el teorema de Birkhoff la cantidad de bits por letra que se usa codificando μ-casi toda tira usando f_m es igual a

$$\sum_{a_1,\ldots,a_m} |f_m(a_1, \ldots, a_m)|\mu([a_1, \ldots, a_m]) \leq (1-\delta_m)(2/m+H+\epsilon)+\delta_m(2/m+\log_2(|A|)).$$

Esto se puede lograr que sea tan cercano a H como se quiera tomando $\epsilon > 0$ suficientemente chico y m suficientemente grande.

Nos queda demostrar que ningún código binario sin prefijos logra codificar μ-casi toda secuencia usando estrictamente menos que $H - \epsilon$ bits por letra para algún $\epsilon > 0$.

Sea $f : A^m \to \{0,1\}^*$ un código binario sin prefijos cualquiera. Recordemos que por la desigualdad de Kraft (ejercicio 1) se cumple

$$\sum_{a_1,\ldots,a_m \in A} 2^{-|f(a_1,\ldots,a_m)|} \leq 1.$$

Por otro lado por los ejercicios 7 y 5 tenemos

$$\begin{aligned} H \leq \frac{1}{m} H_m &= \sum_{a_1,\ldots,a_m} \mu([a_1,\ldots,a_m]) I(\mu([a_1,\ldots,a_m]) \\ &\leq \sum_{a_1,\ldots,a_m} \mu([a_1,\ldots,a_m]) I(\mu(q_{a_1,\ldots,a_m})) \end{aligned}$$

para toda elección de $q_{a_1,\ldots,a_m} \in [0,1]$ con suma menor o igual a 1.

Reemplazando $q_{a_1,\ldots,a_m}$ por $2^{-|f(a_1,\ldots,a_m)|}$ se obtiene el resultado. $\square$