Compresión sin pérdida de imágenes utilizando códigos óptimos para distribuciones geométricas bidimensionales

Pablo Rotondo

Tutores: Álvaro Martín, Fabián Crocce, Gadiel Seroussi.

Proyecto de Grado de Ingeniería en Computación y
Monografía de Licenciatura en Matemática

Universidad de la República,
Uruguay,
20 de mayo de 2014

***Resumen*— Recientemente se han caracterizado códigos óptimos para pares de variables geométricas independientes e idénticamente distribuidas, lo cual abre la interrogante de si estos pueden ser utilizados para obtener mayores tasas de compresión para la compresión sin pérdida de imágenes. Presentamos estos códigos óptimos, así como sus antecesores unidimensionales, en detalle. Incorporamos los códigos bidimensionales al algoritmo LOCO-I/JPEG-LS con el objetivo de codificar imágenes a color. Para ello, estudiamos una familia de baja complejidad de los códigos bidimensionales, similar a la utilizada en LOCO-I/JPEG-LS con los unidimensionales, mostrando reglas de decisión aproximadas para seleccionar un código de dicha familia. Finalmente, discutimos los resultados empíricos y los comparamos con los resultados teóricos.**

***Palabras claves*— distribución geométrica, códigos de prefijo, códigos de Golomb, compresión de imágenes sin pérdida.**

Índice general

CAPÍTULO 1

Introducción

Las imágenes digitales consisten en un conjunto de matrices bidimensionales de enteros, cada una denominada 'componente', donde cada entero es denominado 'muestra'. A las muestras correspondientes a una posición geométrica fija de las matrices, así como la propia posición, los denominamos 'píxel'. Dichos enteros generalmente pertenecen a un rango $\{0, 1, \ldots, 2^\beta - 1\}$, donde β denota el número de bits (precisión) en la representación de cada muestra. Por ejemplo, en el caso de imágenes de tono de gris, existe un único componente que representa, por cada píxel, la intensidad de luz en dicha posición, siendo entonces $2^\beta - 1$ blanco y 0 negro. Otro ejemplo es el espacio RGB, para el cual hay tres componentes (a veces llamados también canales), uno para la intensidad de cada color primario Rojo R, Verde G y Azul B.

La compresión consiste en la codificación de datos como secuencias de símbolos (usualmente bits), de manera que el largo de esta secuencia, llamado largo de código, sea típicamente corta, de modo de ahorrar espacio de almacenamiento o ancho de banda de comunicación. En el caso de imágenes digitales, es usual tener esquemas denominados 'con pérdida'. Estos se basan en que la imagen reconstruída pueda no ser idéntica a la original, de modo de obtener una representación más corta que, por ejemplo, tenga un error por píxel acotado, o que visualmente no sea distinguible de la original. Esto no siempre resulta aceptable, especialmente si la imagen es de importancia crítica o va a estar sujeta a más procesamiento a posteriori. En estos casos se utilizan esquemas denominados 'sin pérdida', para los cuales la imagen reconstruída es exactamente la imagen original.

En los esquemas de compresión sin pérdida, incluyendo los de imágenes, usualmente pueden identificarse dos componentes: modelado y codificación. El modelado consiste en ajustar un cierto modelo probabilístico de la entrada, según los datos dados. Por otro lado, el codificador utiliza el modelo probabilístico para codificar los datos de forma de minimizar la longitud de código esperada con respecto al modelo.

¿Por qué utilizar un modelo probabilístico? En general, estamos interesados en comprimir una cierta clase grande de objetos, por ejemplo imágenes naturales o texto en algún idioma, y no un único objeto especifico. Como en general no es posible reducir la longitud de codificación de todos los objetos de la familia, ya que si unos se codifican con pocos bits, otros necesariamente van a requerir códigos más largos, esperamos reducir el largo de las representaciones de los objetos que aparecen más frecuente-

mente. Por ejemplo, para comprimir texto, es natural asumir que un carácter de la secuencia está muy fuertemente correlacionado con los caracteres inmediatamente anteriores. Es así que una posiblidad es considerar un modelo de Markov, donde la distribución de probabilidad de un carácter condicionado en el valor de todos los caracteres que lo preceden depende únicamente de los k caracteres anteriores, para un número k fijo. Los parámetros del modelo en este ejemplo serían las probabilidades condicionales de cada letra, en cada uno de los estados (de k caracteres) de la cadena de Markov. Los valores apropiados de dichos parámetros pueden de depender de características más particulares de cada objeto concreto, como el idioma del texto o el estilo de escritura.

En general, los parámetros del modelo no son conocidos a priori, y deben ser aprendidos a partir de los datos del objeto concreto. Hay en principio dos formas de hacer esto: secuencialmente, o en dos 'pasadas'. En el primer caso, asumimos que el objeto a codificar está dado por una secuencia de símbolos (por ejemplo píxeles de una imagen o letras de un texto) que se codifican progresivamente y, el codificador, si bien tiene acceso a la secuencia completa a ser comprimida, utiliza en cada instante únicamente, para el modelo estadístico, los datos que ya han sido codificados hasta el momento, de manera que el decodificador tenga también la misma información a la hora de decodificar la siguiente muestra. Por el contrario, en un esquema en dos 'pasadas', los datos del modelo son aprendidos y codificados en una primera etapa y el objeto en sí mismo es codificado en una segunda etapa en base al modelo ajustado.

Los modelos probabilísticos con demasiados parámetros a estimar pueden ser perjudiciales en cuanto a la longitud de código resultante [19], ya que estos deben ser aprendidos o guardados para ser accedidos por el decodificador. Sin embargo, tener más parámetros también permite en general un mejor ajuste del modelo, lo cual, en principio, podría redundar en una menor longitud de código. Así pues, es necesario un compromiso en el número de parámetros.

El algoritmo LOCO-I [15] para la compresión sin pérdida de imágenes de tono de gris, luego estandarizado en el ISO JPEG-LS [16], aborda este problema utilizando la observación, ampliamente aceptada, de que los residuos de predicción de la intensidad de un píxel, en base a los vecinos ya codificados, se distribuyen de una manera bastante similar a una geométrica a dos lados (también conocida como distribución de Laplace) [22]. En las imágenes naturales, es natural asumir también que esta distribución varía poco al modificar levemente los píxeles vecinos, lo cual resulta en 'contextos condicionantes' más amplios, que engloban situaciones de los píxeles vecinos en las que esperamos que la distribución del residuo de predicción sea similar. Esta técnica permite reducir la cantidad de parámetros a estimar, manteniendo un buen rendimiento.

Se conocen codificaciones óptimas, es decir, que minimizan la longitud de código esperada, para variables con distribuciones geométricas [11, 12], que son aprovechadas en algoritmos tales como FELICS [14] y LOCO-I [16] para codificar residuos de predicción. La codificación óptima de una variable X con distribución geométrica de parámetro q, es decir $\Pr(X = n) = (1 - q)\, q^n$, está dada por un 'código de Golomb de orden m', donde el número natural m es un parámetro del código que depende de q. Estos códigos, resultan muy sencillos de implementar, con una complejidad baja comparada con la de otros esquemas de codificación, como la codificación aritmética [4, 20] y la codificación de Huffman [6, 21].

Para las distribuciones geométricas a dos lados se conocen también códigos óptimos [18], pero la caracterización, que utiliza los códigos de Golomb, resulta más compleja. En [18] se muestra también cómo utilizar códigos de Golomb donde m es una potencia de dos permite obtener una regla de selección sencilla del código en función de estadísticas de los datos a codificar, que es utilizada por LOCO-I.

Más recientemente se han caracterizado códigos óptimos para **pares** de variables aleatorias independientes con distribución geométrica [17], bajo un mismo parámetro del tipo $q = 2^{-1/k}$ con $k \in \mathbb{Z}_{>0}$. El

interés en codificar pares radica en que de esta manera es posible, en principio, obtener codificaciones más cortas que codificando cada símbolo individualmente. Esto abre la interrogante de si estas pueden ser utilizadas, manteniendo una complejidad de cómputo similar, para mejorar las tasas de compresión que se obtienen utilizando las distribuciones geométricas unidimensionales.

Utilizamos dichos códigos bidimensionales para la compresión de imágenes digitales a color en el espacio RGB, ya que consideramos que en este contexto los pares de variables geométricas a dos lados indpendientes e idénticamente distribuídas resultan un modelo natural para los residuos de predicción $(\epsilon_{R-G}, \epsilon_{B-G})$ de los componentes que resultan de restar G a R y B.

Resultados. Para que la codificación mantenga una complejidad similar a la de LOCO-I, es necesario definir una regla sencilla de selección de un código para pares de variables con distribución geométrica en función de estadísticas de los datos, para lo cual estudiamos la subfamilia de códigos bidimensionales de baja complejidad dada por $k = 2^r$. En este sentido, desarrollamos resultados asintóticos para la selección de códigos de dicha familia (en otras palabras, para seleccionar r) en base a estadísticas de los datos de entrada, que son similares a [18, Lemma 4] para los códigos de *Golomb-Potencia-De-Dos*, también conocidos como códigos de Rice. Estos resultados llevan a una regla de selección bastante natural, incluso justificando la utilización de prácticamente la misma regla que LOCO-I. Dichos resultados se encuentran descritos en la sección 4.3.2.

Implementamos entonces LOCO-I con las modificaciones para aplicar los códigos bidimensionales al codificar los componentes R y B, y comparamos los resultados obtenidos para un conjunto dado de imágenes con respecto a codificarlas mediante LOCO-I. Los resultados experimentales indican que no se obtiene una ventaja significativa al utilizar los códigos bidimensionales. Los posibles motivos para esto son estudiadas en el capítulo 5, estas incluyen los siguientes puntos.

- Al aplicar los códigos unidimensionales, las estadísticas son actualizadas luego de codificar el primer componente del par, y pueden ser utilizadas para codificar el segundo componente con estadísticas más recientes que cuando se codifica de a pares.
- Dependiendo de los píxels vecinos, algunos pares de residuos de predicción $(\epsilon_{R-G}, \epsilon_{B-G})$ pueden modelarse apropiadamente con componentes independientes e identicamente distribuidos y otros no. Los códigos bidimensionales son usados solo en el primer caso y por lo tanto la potenciales mejoras están condicionadas por la frecuencia con que los códigos bidimensionales son utilizados.
- El valor de los parámetros que son utilizados por lo general, ya que para $k = 0$ y $k = 1$ los códigos bidimensionales no presentan mejoras respecto a los códigos unidimensionales.

Organización del documento. El documento se encuentra dividido en dos partes.

- La primera parte expone los fundamentos teóricos básicos de la compresión y de los códigos óptimos para fuentes con distribución geométrica. Esta parte se compone de dos capítulos. El capítulo 2 presenta los fundamentos básicos de la compresión de datos, mientras que el capítulo 3 trata sobre los códigos óptimos para fuentes con distribución geométrica, mostrando la optimalidad de los códigos de Golomb [12], así como la de los bidimensionales [17]. Para ejemplificar cómo la misma idea se extiende a mayores dimensiones, mostramos también el caso tridimensional.
- La segunda parte trata los aspectos prácticos de los resultados teóricos expuestos, comenzando por discutir el algoritmo LOCO-I [15, 16], utilizado para compresión sin pérdidas de imágenes

de tonos de gris, y mostrando cómo lo modificamos, para utilizar los códigos bidimensionales para la compresión de imágenes a color. Luego de explicar los principios de funcionamiento del algoritmo en el capítulo 4, en el capítulo 5 se procede a mostrar y analizar los resultados experimentales obtenidos, contrastándolos con los resultados teóricos.

Parte I

Fundamentos Teóricos

CAPÍTULO 2

Fundamentos de Compresión de Datos

La codificación consiste en la sustitución de símbolos o palabras de un alfabeto, por palabras en otro alfabeto conveniente, como puede ser el alfabeto binario. Esta sustución no puede ser arbitraria, ya que, si la información va a ser almacenada, o transmitida, es importante que pueda luego ser decodificada, obteniendo el mensaje original. Esto no es todo, ya también es importante que la codificación resulte lo más corta posible en general, para poder así ahorrar espacio de disco, o ancho de banda.

Si lo que decodificamos es exactamente el mensaje original, la compresión se dice *sin pérdida*, para distinguirlo de otros esquemas que permiten una cierta pérdida de información en pos de obtener una mayor compresión.

En este capítulo se introducen conceptos básicos de compresión sin pérdida, sentando las bases para los resultados sobre codificación de variables geométricas, y las aplicaciones de estas a la compresión sin pérdida de imágenes a color, que serán presentadas en los capítulos subsiguientes.

2.1. Definiciones Básicas

Definición 2.1.1 (*Alfabeto*)**.** Un *alfabeto* es un conjunto numerable $\mathcal{A}$, cuyos elementos son conocidos como *símbolos*.

Definición 2.1.2 (*Palabra*, *String*)**.** Dado un alfabeto $\mathcal{A}$, una *palabra* es una secuencia finita $x_1 x_2 \dots x_n$ de elementos de $\mathcal{A}$. Frecuentemente se utiliza también *string* como sinónimo del término palabra. El entero n se conoce como el *largo* de la palabra y escribimos $\ell(x_1 \dots x_n) \triangleq n$.

Se denota mediante $\mathcal{A}^n$ al conjunto de palabras en $\mathcal{A}$ que tienen largo n, y mediante $\mathcal{A}^*$ al conjunto de todas las palabras sobre el alfabeto $\mathcal{A}$.

Finalmente, denotamos mediante ϵ a la *palabra vacía*, de largo 0.

Definición 2.1.3 (*Concatenación*)**.** Dado un alfabeto $\mathcal{A}$, el conjunto $\mathcal{A}^*$ se puede tornar en un monoide bajo la operación binaria $\bowtie : \mathcal{A}^* \times \mathcal{A}^* \to \mathcal{A}^*$, conocida como *concatenación*, que se define mediante

$$(x_1 \dots x_n) \bowtie (y_1 \dots y_m) \triangleq x_1 \dots x_n y_1 \dots y_m \quad .$$

Es directo verificar que esta operación es asociativa, y que ϵ es el neutro.

Definición 2.1.4 (*Código*,*Palabra de Código*)**.** Dados alfabetos $\mathcal{A}$ y $\mathcal{B}$, un *código* es una función $c\colon \mathcal{A} \to \mathcal{B}^*$. Para cada símbolo $s \in \mathcal{A}$, se dice que $c(s) \in \mathcal{B}^*$ es la *palabra de código* asociada a s.

En el caso de que c sea inyectivo, hay una biyección entre $\mathcal{A}$ y $C \triangleq c(\mathcal{A})$. Entonces al conjunto imagen $C \subset \mathcal{B}^*$ se lo conoce tambiń como el *código*, ya que en lo que sigue nos interesan las frecuencias con las que ocurren las palabras de $\mathcal{A}$, y no sus valores particulares, y queremos entonces, dada una secuencia de probabilidades asociarles $C \subset \mathcal{B}^*$.

Queremos codificar *tiras* de símbolos sobre un alfabeto $\mathcal{A}$, de manera que sea posible *decodificar* la stream original, y de modo que la codificación sea corta en cuanto a longitud. Es así que, no estamos interesados en códigos $c\colon \mathcal{A} \to \mathcal{B}^*$ que no sean inyectivos, y además es significativo extender el código $c\colon \mathcal{A} \to \mathcal{B}^*$ a $c\colon \mathcal{A}^* \to \mathcal{B}^*$, pero de manera que siga siendo inyectivo.

Definición 2.1.5 (*Concatenación de Código*)**.** Dado un código $c\colon \mathcal{A} \to \mathcal{B}^*$, se llama *concatenación* de c al código $c\colon \mathcal{A}^* \to \mathcal{B}^*$ definido por

$$c(x_1 \dots x_n) \triangleq c(x_1) \bowtie \dots \bowtie c(x_n) \in \mathcal{B}^* \,,$$

para cada palabra $x_1 \dots x_n \in \mathcal{A}^*$. Observar en particular para $n = 0$ que $c(\epsilon) = \epsilon$, ya que los productos son vacíos.

Frecuentemente utilizamos el alfabeto binario $\mathcal{B} = \{0, 1\}$ y por lo tanto lo denotamos con una letra especial $\mathbb{B}$. Este alfabeto resulta conveniente, no solo por ser simple, sino que también porque es el usado por lo general en la práctica ya que la memoria de las computadoras funciona en base a *bits*, que es el nombre que utilizamos para los símbolos de $\mathbb{B}$.

Definición 2.1.6 (*Alfabeto b–ario*, *Código b–ario*)**.** Dado un alfabeto finito $\mathcal{B}$ decimos que este es *b–ario* cuando $|\mathcal{B}| = b$, por ejemplo, el alfabeto binario $\mathbb{B}$ es 2–ario. De manera similar, decimos que un código $c : \mathcal{A} \to \mathbb{B}^*$ es *b–ario* si $|\mathcal{B}| = b$.

Una pregunta natural es si tener $c\colon \mathcal{A} \to \mathcal{B}^*$ inyectiva es suficiente para que su concatenación sea inyectiva, sin embargo, no hay que buscar mucho para encontrar que esto es falso;

Ejemplo 2.1.1. Sean $\mathcal{A} = \{x, y, z\}$ y $c : \mathcal{A} \to \mathbb{B}^*$ definida por $c(x) = 01$, $c(y) = 10$, $c(z) = 0$. Nótese entonces que

$$c(xz) = c(x)c(z) = (01) \bowtie (0) = 010 = (0) \bowtie (10) = c(z)c(y) = c(zy) \,,$$

y así la extensión no es inyectiva.

En cualquier caso, parece intuitivo y es práctico, leer de izquierda a derecha en $\mathbb{B}^*$ hasta encontrar una palabra de código, decodificar esta, y seguir con el resto de la string de entrada. Sin embargo, como el ejemplo anterior demuestra, esto nos haría decodificar 010 como cb, incluso cuando la palabra original podría haber sido ac. El problema es que $c(x)c(z) = c(xz) = c(zy) = c(z)c(y)$, y, al ser c inyectiva $c(x) \neq c(z)$, una de las dos debe ser más corta, en este caso $c(x) = c(z)1$.

En este caso se dice que $c(z)$ es prefijo de $c(x)$. Más generalmente, si $w = vq$ para algún $q \in \mathcal{B}^*$, decimos que v es prefijo de w. Esto sugiere que la siguiente definición será significativa

Definición 2.1.7 (*Código libre de prefijos, o de prefijos*)**.** Un código $c : \mathcal{A} \to \mathcal{B}^*$ se dice *libre de prefijos* (o *de prefijos*) si y sólo si tener dos símbolos $x, y \in \mathcal{A}$ con $c(x)$ prefijo de $c(y)$, implica tener $x = y$.

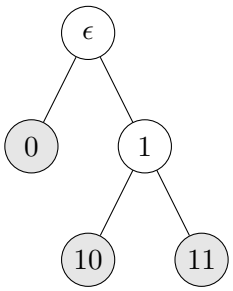


Figura 2.1.1: Representación del código $C = \{0, 10, 11\} \subset \mathbb{B}^*$.

Proposición 2.1.1. *Si $c : \mathcal{A} \to \mathcal{B}^*$ es un código libre de prefijos, su concatenación $c\colon \mathcal{A}^* \to \mathcal{B}^*$ es inyectiva.*

Demostración. Consideremos dos strings con igual codificación $c(x_1 \dots x_n) = c(y_1 \dots y_m)$. Se tiene entonces que

$$c(x_1) \dots c(x_n) = c(y_1) \dots c(y_m)\,,$$

lo cual implica que $c(x_1)$ y $c(y_1)$ deben ser una prefijo de la otra, y, por lo tanto, $x_1 = y_1$ ya que nuestro código es libre de prefijos. Concluímos también que

$$c(x_2) \dots c(x_n) = c(y_2) \dots c(y_m)\,,$$

y, razonando inductivamente en $\min\{n, m\}$, se sigue el resultado. □

Definición 2.1.8 (*Código unívocamente decodificable*)**.** Un código $c\colon \mathcal{A} \to \mathcal{B}^*$ es unívocamente decodificable (UD), si su concatenación $c\colon \mathcal{A}^* \to \mathcal{B}^*$ es inyectiva.

Ya probamos que un código libre de prefijos es unívocamente decodificable, sin embargo, no es cierto que todo código UD sea libre de prefijos. Pero, de cualquier modo, mostramos en la siguiente sección que no perdemos nada en cuanto a economía de la codificación, si utilizamos los códigos libres de prefijo en lugar de los más generales UD.

Ejemplo 2.1.2. Notamos que el código $C = \{0, 01\} \subset \mathbb{B}^*$ no es libre de prefijos, sin embargo es UD. En efecto, podemos reconocer si la primer palabra de código es 0 o 01 según el segundo símbolo sea 0 o 1 respectivamente.

Definición 2.1.9 (*Árbol b–ario*)**.** Un árbol b–ario T es un árbol (V, E), con un nodo distinguido r llamado *raíz*, en el cual cada nodo tiene a lo sumo b *hijos*.

Un nodo v es *hijo* de un nodo w si y sólo si v y w son adyacentes, y la distancia de v a r es mayor que la distancia de w a r. Finalmente, dado un nodo $w \in V$, denominamos *descendientes* de w a los nodos conformados por w, los hijos de w, los hijos de los hijos de w, etc. En otros términos, v es descendiente de w si y sólo si el camino de v a r (que es único) incluye a w, y es v es hijo de w, si w es el nodo inmediatamente siguiente a v en dicho camino.

La distancia de un nodo $v \in V$ a la raíz se conoce como su *profundidad*, o, el *nivel* en que v se encuentra.

Definición 2.1.10 (*Árbol asociado a un Código libre de prefijos*)**.** Dado un código b–ario $C \subset \mathcal{B}^*$, libre de prefijos, asociamos a este un árbol b–ario con raíz y etiquetas (labels) utilizando las siguientes reglas:

a) La raíz corresponde a la string vacía ϵ.

b) Dado un nodo con string correspondiente s, sus hijos corresponden a distintas strings (no necesariamente todas) de $\{s \bowtie x : x \in \mathcal{B}\}$. Esto se construye de forma inductiva según la distancia a la raíz.

c) Las hojas corresponden a las palabras de código de C.

d) Cada nodo del árbol tiene al menos una hoja entre sus descendientes (posiblemente el mismo nodo).

Nótese que estas reglas describen únicamente códigos libres de prefijo. En efecto, la string correspondiente a un nodo v es prefijo de la string correspondiente a otro nodo w solamente si el v nodo aparece en el camino entre la raíz y w, en cuyo caso el primer nodo v no era una hoja. Observar también que dado un código b–ario y libre de prefijos C, las reglas dadas definen un único árbol, y viceversa, dado un árbol b–ario, este define un código único.

Esta definición se encuentra ilustrada en la Figura 2.1.1, observar que, en la representación gráfica del árbol, los hijos se encuentran ordenados de izquierda a derecha según el símbolo $x \in \mathcal{B}$ que se concatena al final.

Finalmente, decimos que un árbol b–ario es *completo* si y sólo si cada nodo o es una hoja o tiene **exactamente** b hijos. Por ejemplo, el árbol 2–ario de la Figura 2.1.1 es completo, pero el árbol asociado a $C = \{0, 10\}$ no lo es.

2.2. Desigualdad de Kraft

Dado un código b–ario $C \subset \mathcal{B}^*$ definimos el *perfil* de C como la sucesión $\{n_l\}_{l\geq 0} \subset \mathbb{Z}_{\geq 0}$ donde n_l corresponde al número de palabras de largo l en C.

A lo largo de esta sección consideramos un código b–ario C, finito, y denotamos

$$\ell_{\text{máx}} \triangleq \text{máx}\{l \in \mathbb{Z}_{\geq 0} : n_l > 0\}\,,$$

la máxima longitud asociada a una palabra de código. Nuestro objetivo encontrar condiciones que relacionen el supuesto perfil con la posibilidad de construir códigos libres de prefijos para dicho perfil.

Definición 2.2.1 (*Número de Kraft-McMillan*)**.** El número de Kraft-McMillan asociada a una sucesión $\{n_l\}_{l\geq 0} \subset \mathbb{Z}_{\geq 0}$ y una base b es

$$K \triangleq \sum_{l\geq 0} \frac{n_l}{b^l}\,. \tag{2.2.1}$$

Otra forma de escribir K, cuando el perfil corresponde a un código b–ario C, es

$$K = \sum_{c\in C} b^{-\ell(c)}, \tag{2.2.2}$$

donde la igualdad en (2.2.2) sigue de agrupar los sumandos según $\ell(c)$.

Teorema 2.2.1 (*Suficiencia*)**.** *Dados $n_1, \ldots, n_{\ell_{\text{máx}}}$ y una base $b < \infty$, si tenemos $K \leq 1$ existe un código b–ario, libre de prefijo $C \subset \mathcal{B}^*$, con el perfil dado.*

Demostración. La prueba es constructiva, es decir, nos dará un algoritmo para construir C. Comenzamos notando que $K \leq 1$ implica que $\frac{n_1}{b^1} \leq 1$, es decir $n_1 \leq b^1$, y por lo tanto podemos fijar n_1 hojas en el primer nivel luego de la raíz, sobrando $b - n_1$ lugares para usar como nodos internos.

En general, una vez que hemos terminado con las palabras de longitudes menores que l, hay $n_i b^{l-i}$ palabras de longtiud l que tienen alguna de nuestras palabras de longitud i como prefijo, por lo cual tenemos exactamente $g_l = b^l - b^{l-1} \times n_1 - b^{l-2} \times n_2 - \ldots - b \times n_{l-1}$ lugares disponibles para nodos a profundidad l.

Obtenemos entonces

$$\frac{g_l}{b^l} = 1 - \frac{n_1}{b^1} - \ldots - \frac{n_{l-1}}{b^{l-1}} \geq \frac{n_l}{b^l},$$

donde la desigualdad es consecuencia de que $K \leq 1$, y así $g_l \geq n_l$ y podemos ubicar las palabras de largo l. □

Observación 2.2.1. Observar en la prueba anterior que el árbol resulta completo si y sólo si $g_l = n_l$ para algún l, y que esto es equivalente a tener $K = 1$.

No es difícil de ver que $K \leq 1$ es una condición necesaria para la existencia de un código libre de prefijos, pero el siguiente teorema prueba algo más general.

Teorema 2.2.2 (*Necesidad*)**.** *Los parámetros $n_1, \ldots, n_{\ell_{\text{máx}}}$ del perfil de un código b–ario, unívocamente decodificable $C \subset \mathcal{B}^*$ un código, cumplen $K \leq 1$.*

Demostración. Para esto consideramos la función generatriz $A(x) = \sum_{l=1}^{\ell_{\text{máx}}} n_l x^l$, notamos entonces que el coeficiente de x^l en $(A(x))^N$ corresponde al número de formas de generar palabras de largo l, concatenando N palabras de C.

Al ser C unívocamente decodificable, tenemos necesariamente que el coeficiente de x^l en $(A(x))^N$ no puede exceder b^l, ya que de otro modo alguna palabra de largo l tendría dos formas distintas de escribirse como concatenación de N palabras de código.

Escribimos ahora $(A(x))^N = \sum_{l=1}^{N \cdot \ell_{\text{máx}}} a_l x^l$ ya que no se pueden generar palabras de largo mayor que $N \cdot \ell_{\text{máx}}$, siendo que la longitud máxima de las palabras de C es M, en otras palabras porque $\deg A(x) = \ell_{\text{máx}}$. Vimos arriba que $a_l \leq b^l$, y por lo tanto, observando que $K = A(1/b)$, tenemos

$$K^N = (A(1/b))^N = \sum_{l=1}^{N \cdot \ell_{\text{máx}}} \frac{a_l}{b^l} \quad \leq \quad N \cdot \ell_{\text{máx}}\,, \tag{2.2.3}$$

válido para $N \geq 1$.

Supongamos ahora que $K > 1$. Podemos escribir $K = 1 + \delta$ con $\delta > 0$, y tenemos entonces que

$$K^N = (1+\delta)^N = \sum_{k \geq 0} \binom{N}{k} \cdot \delta^k \geq \binom{N}{2} \cdot \delta^2 = N \cdot \frac{N-1}{2} \cdot \delta^2\,. \tag{2.2.4}$$

Como $\dfrac{N-1}{2} \cdot \delta^2 > \ell_{\text{máx}}$ para todo N suficientemente grande, se sigue por (2.2.4) que $K^N > N \cdot \ell_{\text{máx}}$ para todo N suficientemente grande, en contradicción con (2.2.3). Concluímos entonces que se debe cumplir $K \leq 1$. □

En otras palabras, siempre que tenemos un código UD, podemos construir un código libre de prefijos con los mismos parámetros. Como para la codificación económica lo que nos va a interesar es únicamente los parámetros, y no las palabras en sí, esto quiere decir que nos podemos concentrar en estudiar únicamente códigos libres de prefijo.

A la desigualdad $K \leq 1$ se la conoce por el nombre de **desigualdad de Kraft**.

2.3. Codificación de símbolo aleatorios

Usualmente los datos de entrada se modelan como un proceso estocásticamente. Para comenzar, supongamos que cada símbolo del alfabeto de entrada $\mathcal{A}$ aparece con una cierta frecuencia relativa; esto es, tenemos una distrubción probabilidad $\mathbf{p}$ sobre sus símbolos.

Nuestro problema de interés entonces, es minimizar la *longitud media*

$$L(c) \triangleq \sum_{a \in \mathcal{A}} \ell(c(a)) \cdot p(a), \tag{2.3.1}$$

sobre los códigos unívocamente decodificables $c\colon \mathcal{A} \to \mathcal{B}^*$.

Podemos escribir este problema como un problema de programación entera, donde las variables corresponden a las longitudes ℓ_a para $a \in \mathcal{A}$, de la siguiente manera:

$$\begin{aligned} \text{mín} \quad & \sum_{a \in \mathcal{A}} \ell_a \cdot p(a) \\ \text{sujeto a} \quad & K = \sum_{a \in \mathcal{A}} b^{-\ell_a} \leq 1, \\ & \ell_a \in \mathbb{Z}_{\geq 0}, \forall a \in \mathcal{A}. \end{aligned}$$

Esto se justifica de la siguiente manera:

- El teorema 2.2.1 implica que dada una solución a este problema de programación entera, podemos encontrar un código libre de prefijos c tal que $\ell_a = \ell(c(a))$ para cada $a \in A$.
- Recíprocamente, el teorema 2.2.2 implica que las longitudes $\ell_a = \ell(c(a))$ de un código UD para $\mathcal{A}$ deben satisfacer $K \leq 1$, y por lo tanto resultan solución factible.
- Es importante notar que en este caso la justificación de la existencia del mínimo es sencilla, dado que estamos asumiendo por ahora que $|\mathcal{A}| < \infty$.

Definición 2.3.1 (*Fuente con probabilidad* $\mathbf{p}$)**.** Sea $\mathcal{A}$ un alfabeto y $\mathbf{p}$ una distribución sobre $\mathcal{A}$, decimos entonces que $(\mathcal{A}, \mathbf{p})$ es una fuente con distribución de probabilidad $\mathbf{p}$.

En el caso de que las variables aleatorias $X_1, X_2, \ldots$ que asociamos a nuestros datos de entrada sean idénticamente distribuídas con $(\mathcal{A}, \mathbf{p})$, observamos que si $c\colon \mathcal{A} \to \mathcal{B}^*$ tenemos

$$\mathbb{E}[\ell(c(X_1 \ldots X_N))] = N \times L(c),$$

subrayando entonces, nuevamente, la importancia de minimizar $L(c)$.

2.4. La Entropía

La denominada *entropía* correspondiente a una distribución $\mathbf{p} = (p_1, \ldots, p_n)$ surgirá naturalmente de acotar inferiormente la longitud media L, para cualquier código para esta fuente, mediante una relajación de la reestricción $\ell_i \in \mathbb{Z}_{\geq 0}$. Si quitamos la reestricción de que ℓ_i han de ser enteros positivos tomando en su lugar $\ell_i \in \mathbb{R}_{\geq 0} = [0, \infty)$, notamos que tener $K < 1$ nos permitiría reducir un poco por ejemplo ℓ_1, reduciendo también L sin violar la reestricción $K \leq 1$.

Es así que consideramos el siguiente problema de minimización:

$$\begin{aligned} \text{mín} \quad & \sum_{i=1}^{n} \ell_i \cdot p_i \\ \text{sujeto a} \quad & K = \sum_{i=1}^{n} b^{-\ell_i} = 1, \\ & \ell_i \in \mathbb{R}_{\geq 0}, \forall i \in \{1, \ldots, n\}. \end{aligned}$$

Para minimizar buscamos los multiplicadores de Lagrange; sea $J = \sum_i p_i \ell_i + \lambda \sum_i b^{-\ell_i}$, derivando obtenemos

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda \log(b) b^{-\ell_i},$$

e igualando a 0 tenemos $b^{-\ell_i} = \dfrac{p_i}{\lambda \cdot \log(b)}$. Al ser $\sum_i b^{-\ell_i} = 1$ debemos tener $\lambda = 1/\log(b)$ y así $b^{-\ell_i} = p_i$ i.e. $\ell_i = \log_b(1/p_i)$ que nos daría una longitud $\sum_i p_i \log_b(1/p_i)$.

Definición 2.4.1 (*Entropía*)**.** Dada una variable aleatoria discreta X, o, equivalentemente, su distribución $\mathbf{p} = (p_1, \ldots, p_n, \ldots)$, definimos la *entropía* de X como

$$H_b(\mathbf{p}) = \sum_i p_i \log_b(1/p_i),$$

y también utilizamos la notación $H_b(X)$. Un par de notas al respecto:

- Se utiliza la convención de que $0 \times \infty = 0$, que es usual en la Teoría de la Medida. Así, si $p_i = 0$, tomamos el término $p_i \log_b(1/p_i) = 0$. Observar que esta elección es razonable en que es la condición que nos da continuidad, ya que $\lim_{p \to 0^+} p \log(p) = 0$.
- El cambio de base b corresponde al cambio de base de los logaritmos, y corresponde a un *cambio de unidad*, ya que $H_b(X)$ tendrá unidades de longitud en símbolos de un alfabeto de tamaño b. En caso de que la base b no se especifique, se entiende que $b = 2$ y su unidad correspondiente es el *bit*.
- Observar que $H(X) \geq 0$.

Lema 2.4.1. *Para $x > 0$ tenemos*

$$\log(x) \leq x - 1$$

con igualdad si y solamente si $x = 1$.

Demostración. Sea $f(x) = x - \log(x)$, derivando encontramos $f'(x) = 1 - 1/x$. Ahora, el signo de la derivada resulta negativo para $x < 0$, positivo para $x > 0$ y 0 para $x = 1$. Se sigue entonces que $f(x)$ tiene un mínimo absoluto en $x = 1$ y es $f(1) = 1$. □

Lema 2.4.2. *Sean $\{p_1, \ldots, p_n\}$ y $\{q_1, \ldots, q_n\}$ distribuciones de probabilidad sobre $\{1, \ldots, n\}$. Luego*

$$\sum_{i=1}^{n} p_i \log(1/p_i) \leq \sum_{i=1}^{n} p_i \log(1/q_i)$$

con igualdad si y solo si $p_i = q_i$ para todo $i \in \{1, \ldots, n\}$.

Demostración. El resultado es equivalente a

$$0 \geq \sum_{i=1}^{n} p_i \cdot \Big(\log(1/p_i) - \log(1/q_i) \Big) = \sum_{i=1}^{n} p_i \log\left(q_i/p_i\right)$$

para probar esto último notamos que $\log\left(q_i/p_i\right) \leq q_i/p_i - 1$ por el lema 2.4.1 y así

$$\sum_{i=1}^{n} p_i \log\left(q_i/p_i\right) \leq \sum_{i=1}^{n} p_i \cdot (q_i/p_i - 1) = \sum_{i=1}^{n} q_i - \sum_{i=1}^{n} p_i = 1 - 1 = 0$$

y el caso de igualdad se sigue del caso de igualdad en el lema 2.4.1. □

Corolario 2.4.1. *Dada una distribución* $\mathbf{p} = (p_1, \ldots, p_n)$ *tenemos*

$$H_b(\mathbf{p}) \leq \log_b(n)$$

con igualdad si y solo si $\mathbf{p}$ *es uniforme.*

Demostración. En el lema 2.4.2 tomamos $q_i = 1/n$ para cada $i = 1, \ldots, n$. □

Teorema 2.4.1 (*Teorema de la codificación de fuente*)**.** *El largo medio de un código* $C \subset \mathcal{B}$ *b–ario y libre prefijos para una variable aleatoria discreta* X *satisface*

$$L(C) \geq H_b(X)\,. \tag{2.4.1}$$

Demostración. Sea K el número de Kraft-McMillan correspondiente a C, sabemos que $K \leq 1$. Dada nuestra distribución $\mathbf{p}$, definimos otra distribución $\mathbf{q}$ mediante $q_i = b^{-\ell_i}/K$, para aplicar el lema 2.4.2 obteniendo

$$\begin{aligned} H_b(\mathbf{p}) = \sum_{i=1}^{n} p_i \log_b(1/p_i) &\leq \sum_{i=1}^{n} p_i \log_b(1/q_i) \\ &= \sum_{i=1}^{n} p_i \log_b\left(K b^{\ell_i}\right) \\ &= \sum_{i=1}^{n} p_i \log_b(K) + \sum_{i=1}^{n} p_i \ell_i \\ &= \log_b(K) + \sum_{i=1}^{n} p_i \ell_i \\ &\leq \sum_{i=1}^{n} p_i \ell_i = L(C)\,, \end{aligned}$$

donde la última desigualdad se sigue de que $\log_b(K) \leq 0$, dado que $K \leq 1$. □

Definición 2.4.2 (*Redundancia*)**.** Dada una fuente $(\mathcal{A}, \mathbf{p})$ y un código C para esta, decimos que

$$L(C) - H\left(\mathbf{p}\right)$$

es la *redundancia* del código.

Observación 2.4.1. Sería deseable obtener parámetros $\{\ell_i\}_{i=1}^{n}$ tales que se pueda alcanzar la igualdad con la entropía. Para esto debemos tener $p_i = q_i$ para cada $i \in \{1, \ldots, n\}$ i.e. $p_i = b^{-\ell_i}/K$, y además $\log_b(K) = 0$, esto es $K = 1$. Por lo tanto, necesitamos tener $K = 1$ y $p_i = b^{-\ell_i}$ para cada i. Esto último es una condición bastante reestrictiva, ya que por lo general las probabilidades no son necesariamente de la forma $\{b^0, b^{-1}, b^{-2}, \ldots\}$.

Observación 2.4.2 (*Código de Shannon-Fano*)**.** De cualquier modo, se pueden elegir los parámetros $\ell_i \in \mathbb{Z}_{\geq 0}$ de modo que no estemos muy lejos de la igualdad:

Tomando el menor entero mayor o igual a $\log_b(1/p_i)$, que denotamos $\ell_i = \lceil \log_b(1/p_i) \rceil$, obtenemos de inmediato que $p_i \geq b^{-\ell_i}$ y así

$$1 = \sum_{i=1}^{n} p_i \geq \sum_{i=1}^{n} b^{-\ell_i} = K\,,$$

y por el teorema 2.2.1 podemos construir un código libre de prefijo para estos parámetros.

Corolario 2.4.2. *Dada una distribución* $\mathbf{p} = (p_1, \ldots, p_n)$*, existe un código* C*, b-ario, libre de prefijos, que satisface*

$$H_b(\mathbf{p}) \leq L(C) < H_b(\mathbf{p}) + 1\,.$$

Demostración. Por el teorema 2.4.1 ya sabemos que se cumple la primera desigualdad. Consideramos ahora el Código de Shannon-Fano de la observación 2.4.2. Se cumple $\ell_i = \lceil \log_b(1/p_i) \rceil < \log_b(1/p_i) + 1$, y así $L(C) = \sum_{i=1}^{n} \ell_i p_i < \sum_{i=1}^{n} (1 + \log_b(1/p_i)) p_i = 1 + H_b(\mathbf{p})$. □

El corolario 2.4.2 establece la existencia de códigos cuya redundancia es menor a un bit. Si en lugar de codificar símbolos del alfabeto original $\mathcal{A}$ codificamos bloques de N símbolos (i.e. símbolos de $\mathcal{A}^N$), la redundancia por símbolo de $\mathcal{A}$ queda acotada por $1/N$ y puede hacerse arbitrariamente pequeña aumentando N. Esto se explica en detalle en la sección 2.6.

2.5. Codificación de Huffman

En esta sección se presenta un algoritmo para, dada una distribución $\mathbf{p} = (p_1, \ldots, p_n)$ sobre un alfabeto finito, encontrar un código libre de prefijo, binario (i.e. sobre $\mathbb{B} = \{0, 1\}$) **óptimo** en el sentido de que tenga longitud media mínima. Este algoritmo fue publicado por primera vez por Huffman [6], y se basa en la metodología *greedy*.

Comenzamos observando algunas propiedades que deben satisfacer los códigos óptimos binarios, que en última instancia mostramos, no son necesariamente ciertas en el caso de otras bases.

Lema 2.5.1. *Supongamos que tenemos una distribución* $\mathbf{p} = (p_1, \ldots, p_n)$ *con* $p_i > 0$ *para todo* $i \in \{1, \ldots, n\}$.

I) *Todo código óptimo debe satisfacer* $\ell_i \geq \ell_j$ *si* $p_i < p_j$.

II) *Para todo código libre de prefijos óptimo binario, las palabras de longitud máxima vienen de a pares de hermanas* $x0$ *y* $x1$*. De hecho, más generalmente, un código binario óptimo, libre de prefijos, debe corresponder a un árbol completo.*

III) *Podemos encontrar un código binario que es libre de prefijos y óptimo, tal que los dos símbolos de menor probabilidad tengan palabras de código que difieran en el último bit.*

Demostración. I) De cumplirse $\ell_i < \ell_j$ cuando $p_i < p_j$, podemos cambiar las palabras de código correspondientes a i y a j, para obtener un cambio de longitud media

$$\Delta L = (p_i \ell_j + p_j \ell_i) - (p_i \ell_i + p_j \ell_j) = (p_i - p_j) \cdot (\ell_j - \ell_i) < 0\,,$$

con lo cual el primer código no puede ser óptimo.

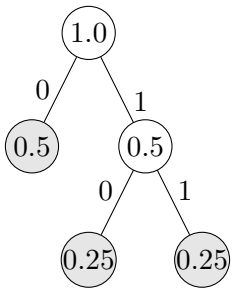


Figura 2.5.1: Aplicación de Huffman sobre la fuente $\mathbf{p} = (0.5, 0.25, 0.25)$.

II) Simplemente, si un nodo correspondiente a una palabra de longitud máxima no tiene hermano, podemos eliminar su último bit, y el código sigue siendo libre de prefijo. En particular, los nodos de profundidad máxima deben tener una palabra de código como hermano. Notar que el árbol no es necesariamente completo si la base es $b > 2$, ya que podríamos tener un solo hermano, y ya no podríamos eliminar el último símbolo.

III) Reordenar los nodos que están en un mismo nivel del árbol no afecta la longitud media. Por el item II sabemos que en el nivel más profundo debe haber por lo menos 2 palabras de código (si $n > 1$), y por el item I sabemos que dos de estas deben corresponder a los símbolos menos probables, podemos reordenarlos de manera que sean hermanas las palabras de código correspondientes. □

El lema 2.5.1 sugiere el siguiente procedimiento para encontrar un código de prefijos óptimo: tomamos los dos símbolos menos probables, digamos i y j, y hacemos un merge de estos, formando un nuevo nodo $[i, j]$ con probabilidad asociada $p_i + p_j$. Ahora repetimos el procedimiento con la distribución

$$\mathbf{p}^* = (p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_{j-1}, p_{j+1}, \ldots, p_n, p_i + p_j)$$

que tiene $n - 1$ símbolos. Cuando llegamos a tener un solo símbolo, podemos asignarle a este directamente la palabra vacía ϵ. Finalmente, la secuencia de merges hechos indica, en orden inverso, la palabra a asignarle a cada símbolo. Dicho de otra manera, si C^*_{n-1} es el código que resulta de aplicar el algoritmo a la fuente $\mathbf{p}^*$, definimos C^*_n mediante

$$C^*_n(k) = \begin{cases} C^*_{n-1}(k), & \text{si } k \notin \{i, j\}\,; \\ C^*_{n-1}([i, j]) \bowtie 0, & \text{si } k = i\,; \\ C^*_{n-1}([i, j]) \bowtie 1, & \text{si } k = j\,. \end{cases}$$

A este algoritmo se lo conoce como la *codificación de Huffman*.

Teorema 2.5.1. *Dada una fuente finita $(\mathcal{A}, \mathbf{p})$, el algoritmo de Huffman produce un código binario, libre de prefijos, óptimo para la fuente.*

Demostración. Como siempre, comenzamos asociando a nuestros $n = |\mathcal{A}|$ símbolos los enteros del 1 al n, sin pérdida de generalidad.

La prueba será inductiva, es claro el resultado si $n = 1$, ya que nuestro código óptimo es sencillamente $C^*_0 = \{\epsilon\}$ y la longitud correspondiente es 0. Supongamos entonces verdadero el resultado para toda fuente con estrictamente menos de n símbolos, procedemos a probar el resultado para una fuente arbitraria de n símbolos.

Podemos asumir, sin pérdida de generalidad que los símbolos de menor probabilidad son $n-1$ y n. Luego, el código producido por Huffman C^*_{n-1} es un código de prefijos óptimo para la fuente $\mathbf{p}^* = (p_1, \ldots, p_{n-2}, p_{n-1} + p_n)$. Notamos además que

$$L\left(C^*_n\right) - L\left(C^*_{n-1}\right) = p_{n-1} + p_n\,, \tag{2.5.1}$$

ya que a los símbolos $n-1$ y n se les agrega un bit.

Ahora, sea C un código libre de prefijo óptimo para la fuente $\mathbf{p}$, tal que las dos palabras de código correspondientes a dos símbolos $n-1$ y n de menor probabilidad difieren en el último bit, como vimos en el lema 2.5.1. Nos construimos un código C' para la fuente $\mathbf{p}^*$ mediante

$$C'(k) = \begin{cases} C(k), & \text{si } k \neq [n-1, n] \\ x, & \text{si } k = [n-1, n], \end{cases}$$

donde x es el prefijo común de las palabras hermanas $C(n-1)$ y $C(n)$. Observamos también que

$$L\left(C\right) - L\left(C'\right) = p_{n-1} + p_n\,, \tag{2.5.2}$$

porque $n-1$ y n son hermanas en C. Como por hipótesis inductiva $L\left(C'\right) \geq L\left(C^*_{n-1}\right)$, se sigue por (2.5.1) y (2.5.2) que

$$L(C) \geq L\left(C^*_n\right)\,,$$

pero, como C era óptimo para $\mathbf{p}$ se cumple también la otra desigualdad y obtenemos que $L(C) = L\left(C^*_n\right)$, concluyendo que C^*_n es también óptimo. □

2.6. Codificación en bloque

Volviendo la mirada a nuestros datos de entrada $X_1, X_2, \ldots$, resulta importante tener resultados que relacionen la entropía de una tira con las entropías individuales, este será entonces el tema de esta sección.

Dado que la base $b = 2$ aparece muy frecuentemente, de ahora en adelante utilizamos $\lg$ para denotar el logaritmo en base 2.

2.6.1. Entropía conjunta y condicional

Definición 2.6.1 (*Entropía conjunta*)**.** Dado un par de variables aleatorias discretas (X, Y), X tomando valores sobre un alfabeto $\mathcal{A}$ e Y tomando valores sobre un alfabeto $\mathcal{B}$, con distribución conjunta $p(x, y) = \Pr\left(X = x, Y = y\right)$, se define

$$H(X, Y) = \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} p(x, y) \times \lg\left(1/p(x, y)\right)$$

o, escrito de otro modo $H(X, Y) = -\mathbb{E}[\lg(p(X, Y))]$.

Notar que $H(X, Y)$ no es otra cosa que la entropía usual de la definición 2.4.1, aplicada a la variable discreta (X, Y) con alfabeto $\mathcal{A} \times \mathcal{B}$.

Observación 2.6.1. Observar que

$$H((X,Y),Z) = H(X,(Y,Z))$$

y por lo tanto denotamos mediante $H(X,Y,Z)$ a este número, que corresponde de nuevo a la entropía de (X,Y,Z). Esto se generaliza para cualquier número de variables aleatorias discretas.

Definición 2.6.2 (*Entropía condicional*)**.** Dado un par de variables aleatorias discretas (X,Y), X tomando valores sobre un alfabeto $\mathcal{A}$ e Y tomando valores sobre un alfabeto $\mathcal{B}$ se define

$$H(X|Y) = -\mathbb{E}[\lg(p(X|Y)]$$

donde definimos $p(x|y) = \Pr(X = x|Y = y)$ para $x \in \mathcal{A}$ e $y \in \mathcal{B}$, con $\Pr(Y = y) > 0$. Observar que la función $p(x|y)$ no está definida sobre un conjunto de probabilidad 0, pero esto no es un problema en lo que a valor esperado concierne.

O, escrito de otro modo, $H(X|Y) = \mathbb{E}_Y[H(X_Y)]$ donde, dado $y \in \mathcal{B}$ tenemos que X_y es la reestricción de X a $\{Y = y\}$, con las correspondientes probabilidades condicionadas a $Y = y$. De nuevo, X_y tiene sentido siempre que $\{Y = y\}$ tenga probabilidad positiva.

Proposición 2.6.1. *Sea $\mathbf{p} = (p_{i,j})_{(i,j)\in\mathcal{A}\times\mathcal{B}}$ una probabilidad conjunta, con distribuciones marginales $\mathbf{p}'$ y $\mathbf{p}''$. Tenemos entonces*

$$H(\mathbf{p}) \leq H(\mathbf{p}') + H(\mathbf{p}'')$$

con igualdad si y solo si $p_{i,j} = p'_i p''_j$ para cada i,j, esto es, hay independencia entre las marginales.

Demostración. En el lema 2.4.2 tomamos $\mathbf{q} = \left(p'_i p''_j\right)_{(i,j)\in\mathcal{A}\times\mathcal{B}}$, que es en efecto una distribución. Luego

$$H(\mathbf{p}) = \sum_{i,j} p_{i,j} \lg(1/p_{i,j}) \leq \sum_{i,j} p_{i,j} \lg\left(\frac{1}{p'_i p''_j}\right) = H(\mathbf{p}') + H(\mathbf{p}'')$$

con igualdad si y solo si $p_{i,j} = p'_i p''_j$ para cada i,j. □

Corolario 2.6.1. *Dado un par de variables aleatorias discretas (X,Y), X tomando valores sobre un alfabeto $\mathcal{A}$ e Y tomando valores sobre un alfabeto $\mathcal{B}$ tenemos*

$$H(X,Y) \leq H(X) + H(Y)$$

con igualdad si y solo si X e Y son independientes.

Proposición 2.6.2 (*Regla de la cadena*)**.** *Dadas X e Y variables aleatorias discretas, tenemos*

$$H(X,Y) = H(X) + H(Y|X)$$

Demostración. Se sigue de $\Pr(X = x)\cdot\Pr(Y = y|X = x) = \Pr(X = x, Y = y)$, y $\lg(ab) = \lg(a) + \lg(b)$. □

Corolario 2.6.2. *En el contexto de la proposición anterior tenemos $H(Y|X) \leq H(Y)$, con igualdad si y solo si X e Y son variables independientes.*

Demostración. Por el corolario 2.6.1 tenemos que $H(X,Y) \leq H(X) + H(Y)$ con igualdad si y solo si X e Y son independientes, y por la proposición anterior $H(X,Y) = X(X) + H(Y|X)$. □

También se cumple la regla de la cadena cuando condicionamos sobre una variable aleatoria:

Proposición 2.6.3. *Dadas X, Y y Z variables aleatorias discretas, tenemos*

$$H(Y,Z|X) = H(Y|X) + H(Z|X,Y)$$

Demostración. Aplicando la proposición 2.6.2 tenemos

$$H(Y,Z|X) = H(Y,Z,X) - H(X)\,, \tag{2.6.1}$$

y también

$$H(Y|X) = H(Y,X) - H(X)\,. \tag{2.6.2}$$

Combinando, entonces, tenemos

$$H(Y,Z|X) - H(Y|X) = H(Y,Z,X) - H(Y,X)\,. \tag{2.6.3}$$

Aquí podemos ver que $H(Y,Z,X) = H(Z,X,Y) = H(Z,(X,Y))$, y, por la proposición 2.6.2 aplicada viendo el par (X,Y) como una variable, tenemos:

$$\begin{aligned} H(Y,Z,X) &= H(Z,(X,Y)) && (2.6.4)\\ &= H(X,Y) + H(Z|X,Y)\,, && (2.6.5)\end{aligned}$$

esto, junto con (2.6.3) implica

$$H(Y,Z|X) - H(Y|X) = H(Z|X,Y)\,. \tag{2.6.6}$$

En este punto estamos hechos, ya que (2.6.6) se puede reescribir como $H(Y,Z|X) = H(Y|X) + H(Z|X,Y)$. □

Proposición 2.6.4. *Dadas X, Y y Z variables aleatorias discretas, tomando valores en $\mathcal{A}, \mathcal{B}$ y $\mathcal{C}$ respectivamente, tenemos*

$$H(Y,Z|X) \leq H(Y|X) + H(Z|X)$$

Demostración. Como $H(Y,Z|X) = \mathbb{E}_X[H(Y_X, Z_X)]$ y tenemos $H(Y_X,Z_X) \leq H(Y_X) + H(Z_X)$, por el corolario 2.6.1, se sigue que

$$H(Y,Z|X) \leq \mathbb{E}_X[H(Y_X) + H(Z_X)] = \mathbb{E}_X[H(Y_X)] + \mathbb{E}_X[H(Z_X)] = H(Y|X) + H(Z|X)$$

utilizando la linealidad del valor esperado. Notar que la igualdad ocurre sii Y_x y Z_x son independientes para cada $x \in \mathcal{A}$ tal que $\Pr(X = x) > 0$, esto es, que Y y Z son condicionalmente independientes dado X. □

Observar que este argumento se generaliza directamente para más variables. Como en el caso no condicionado, tenemos también la siguiente desigualdad:

Corolario 2.6.3. *Dadas X, Y y Z variables aleatorias discretas, tenemos*

$$H(Z|X,Y) \leq H(Z|X)$$

Demostración. Por la proposición 2.6.3 y la proposición 2.6.4 tenemos $H(Y|X) + H(Z|X,Y) = H(Y,Z|X) \leq H(Y|X) + H(Z|X)$. □

Aplicando primero la proposición 2.6.2 y luego la proposición 2.6.3 reiteradamente, tenemos una versión general de la proposición 2.6.2:

Proposición 2.6.5 (*Regla de la cadena*)**.** *Dadas $X_1, \ldots, X_n$ variables aleatorias discretas, tenemos*

$$\begin{aligned} H(X_1, \ldots, X_n) &= H(X_1) + H(X_2, \ldots, X_n | X_1) \\ &= H(X_1) + H(X_2|X_1) + H(X_3, \ldots, X_n | X_1, X_2) \\ &\vdots \\ &= H(X_1) + H(X_2|X_1) + \ldots + H(X_n | X_1, \ldots, X_{n-1}) \end{aligned}$$

Nuevamente, por el corolario 2.6.2 tenemos

Corolario 2.6.4. *Dadas $X_1, \ldots, X_n$ variables aleatorias discretas, tenemos*

$$H(X_1, \ldots, X_n) \leq H(X_1) + \ldots + H(X_n)$$

con igualdad si y solo si las variables son independientes.

Ejemplo 2.6.1. Para ilustrar los conceptos en esta sección, probamos un caso del denominado Lema de Shearer [9].

Supongamos que n puntos en $\mathbb{R}^3$ tienen exactamente n_x proyecciones distintas sobre el plano YZ, n_y proyecciones sobre el plano XZ y n_z proyecciones sobre el plano XY. Se cumple

$$n^2 \leq n_x \cdot n_y \cdot n_z$$

Demostración. Comenzamos observando que podemos, equivalentemente, probar

$$2 \lg n \leq \lg n_x + \lg n_y + \lg n_z \,. \tag{2.6.7}$$

Elegimos un punto $P = (X, Y, Z)$ de manera uniforme de entre nuestos n puntos, de forma que el corolario 2.4.1 implica $H(X, Y, Z) = \lg n$. En este punto, para que esta representación de $\lg n$ nos sea de utilidad, debemos relacionar los términos del lado derecho de (2.6.7) con la variable P. Observamos, entonces, que (X, Y) es la proyección sobre el plano XY, y análogamente con (Y, Z) y (X, Z). Del corolario 2.4.1 se sigue que $H(X, Y) \leq \lg n_z$, $H(X, Z) \leq \lg n_y$, $H(Y, Z) \leq \lg n_x$, y, por lo tanto, si podemos demostrar que

$$2H(X, Y, Z) \leq H(X, Y) + H(X, Z) + H(Y, Z) \,, \tag{2.6.8}$$

estamos hechos.

Por la proposición 2.6.5 se sigue que

$$\begin{aligned} H(Y, Z) &= H(Y) + H(Z|Y) \\ H(X, Z) &= H(X) + H(Z|X) \\ H(X, Y) &= H(X) + H(Y|X) \,, \end{aligned}$$

y así, agrupando términos convenientemente, tenemos

$$H(X, Y) + H(X, Z) + H(Y, Z) = 2H(X) + \Big(H(Y) + H(Y|X)\Big) + \Big(H(Z|X) + H(Z|Y)\Big) \,.$$

Esta última ecuación tiene un parecido importante con $2H(X,Y,Z)$, en efecto, por la regla de la cadena:

$$2H(X,Y,Z) = 2H(X) + 2H(Y|X) + 2H(Z|X,Y)\,.$$

Para relacionarlos, aplicando el corolario 2.6.2 tenemos $H(Y|X) \leq H(Y)$, y por el corolario 2.6.3 obtenemos también $H(Z|X,Y) \leq H(Z|Y)$ y $H(Z|X,Y) \leq H(Z|X)$, implicando entonces

$$\begin{aligned} 2H(X,Y,Z) &= 2H(X) + 2H(Y|X) + 2H(Z|X,Y) \\ &= 2H(X) + \Big(H(Y|X) + H(Y|X)\Big) + \Big(H(Z|X,Y) + H(Z|X,Y)\Big) \\ &\leq 2H(X) + \Big(H(Y) + H(Y|X)\Big) + \Big(H(Z|X) + H(Z|Y)\Big) \\ &= H(X,Y) + H(X,Z) + H(Z,Y)\,, \end{aligned}$$

probando (2.6.8) y, por lo tanto, el resultado. ◊

2.6.2. Codificación de bloque

Si nuestra **fuente es sin memoria**, es decir, nuestro proceso estocástico $X_1, X_2, \ldots$ sobre $\mathcal{A}$, satisface que las variables aleatorias son independientes e identicamente distribuidas, por el corolario 2.6.4 tenemos $H(X_{i+1}, \ldots, X_{i+n}) = n \times H(\mathbf{p})$ donde $\mathbf{p}$ es la distribución común.

Por lo tanto, el código óptimo para los bloques de a n símbolos, que tienen todos la misma distribución $\mathbf{p}^n$, tiene longitud promedio L_n que satisface

$$n \times H(\mathbf{p}) \leq L_n < n \times H(\mathbf{p}) + 1$$

por el corolario 2.4.2. Y así, la **longitud normalizada** por símbolo de $\mathcal{A}$ (en lugar de símbolos de $\mathcal{A}^n$), que es L_n/n, satisface

$$H(\mathbf{p}) \leq \frac{L_n}{n} < H(\mathbf{p}) + 1/n$$

y así podemos acercarnos arbitrariamente a la entropía, tomando bloques suficientemente grandes.

Observación 2.6.2. Dada una fuente sin memoria, si $r|n$ debemos tener $\frac{L_n}{n} \leq \frac{L_r}{r}$.

Demostración. Notamos que para codificar n símbolos, podemos partir n en $\frac{n}{r}$ bloques consecutivos de r símbolos, cada uno con la misma distribución (de otro modo escribir L_r no tendría sentido), y codificar cada bloque utilizando el código óptimo para bloques de largo r. Luego tenemos

$$L_n \leq \frac{n}{r} \times L_r\,,$$

ya que la distribución de cada bloque de largo r es la misma. ◊

Ejemplo 2.6.2. Por ejemplo, al codificar un par (X,Y) de variables aleatorias independientes e idénticamente distribuídas (*iid*), la longitud promedio no aumenta si codificamos el par conjuntamente en lugar de cada entrada por separado.

Considerar una fuente sin memoria resulta un modelo muy limitado. Por ejemplo, en el caso de la codificación de texto, relaciones como *dada una consontante, es más probable que la siguiente letra sea una vocal*, no son tomadas en cuenta. Una suposición que sí es razonable sin embargo, es que el proceso sea estacionario,

Definición 2.6.3 (*Proceso estacionario*)**.** Un proceso estocástico $\mathcal{X} = (X_1, X_2, \ldots)$ con variables tomando valores en un alfabeto $\mathcal{A}$ se dice estacionario si y solo si $(X_{i_1}, X_{i_2}, \ldots, X_{i_m})$ tiene la misma distribución que $(X_{i_1+\tau}, X_{i_2+\tau}, \ldots, X_{i_m+\tau})$ para todo $m \in \mathbb{Z}_{>0}$, y cualquier elección de índices $1 \leq i_1 < i_2 < \ldots < i_m$ y $\tau \geq 0$.

Definición 2.6.4 (*Tasa entropía de un proceso estocástico*)**.** Dado un proceso $\mathcal{X} = (X_1, X_2, \ldots)$ definimos su entropía mediante

$$H(\mathcal{X}) \triangleq \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n)$$

cuando este límite existe.

En algunos libros como [2] y [4], esta cantidad también es conocida como la *entropía* del proceso, no obstante, es razonable llamarle *tasa* ya que corresponde a la cantidad de bits por símbolo con la que aumenta la entropía, al codificar un símbolo más.

Ya vimos que en caso que $X_1, X_2, \ldots$, sean independientes e idénticamente distribuídas (*iid*), tenemos $H(X_1, \ldots, X_n) = n \times H(\mathbf{p})$, y luego $H(\mathcal{X}) = H(X_1) = H(\mathbf{p})$. De existir el límite, como vimos, el corolario 2.4.2 nos permite, tomando bloques, aproximar arbitrariamente $H(\mathcal{X})$ con la longitud media normalizada. La definición 2.6.4 es la que aparece en [1], también se podría definir $H(\mathcal{X}) = \text{ínf}_{n\in\mathbb{Z}_{>0}} \frac{1}{n} H(X_1, \ldots, X_n)$ como se puede ver en [2], cuya existencia es evidente en general. En [2] también se prueba que dicho ínfimo coincide con el límite en el caso de las fuentes estacionarias.

Proposición 2.6.6. *Para un proceso estacionario $\mathcal{X} = (X_1, X_2, \ldots)$, se tiene que $H(X_n|X_1, \ldots, X_{n-1})$ es no-creceiente con n, y por lo tanto $H'(\mathcal{X}) \triangleq \lim_n H(X_n|X_1, \ldots, X_{n-1})$ existe.*

Demostración. Sea $a_n = H(X_n|X_1, \ldots, X_{n-1}) \geq 0$. Observamos que por el corolario 2.6.3 tenemos

$$a_{n+1} = H(X_{n+1}|X_1, \ldots, X_{n-1}, X_n) \leq H(X_{n+1}|X_2, \ldots, X_{n-1}, X_n) ,$$

y, como el proceso es estacionario, tenemos $H(X_{n+1}|X_2, \ldots, X_{n-1}, X_n) = H(X_n|X_1, \ldots, X_{n-1}, X_n)$, así pues $a_{n+1} \leq a_n$. Por lo tanto la secuencia $\{a_n\}_{n=1}^{\infty}$ es no-creciente y no-negativa, y tiene límite. □

Observar que $H(X_1, \ldots, X_{n+1}) - H(X_1, \ldots, X_n) = H(X_{n+1}|X_1, \ldots, X_n)$, y así $H'(\mathcal{X})$ corresponde al límite de las diferencias de primer orden de $a_n = H(X_1, \ldots, X_n)$. Por este motivo, a $H'(\mathcal{X})$ frecuentemente se lo conoce también como *tasa de entropía*. Ésto no da lugar a confusión en el contexto de los procesos estacionarios, como lo prueba el siguiente corolario

Corolario 2.6.5. *Para un proceso estacionario $\mathcal{X} = (X_1, X_2, \ldots)$, se tiene que $H(\mathcal{X})$ está definido y $H(\mathcal{X}) = H'(\mathcal{X})$.*

Demostración. Notamos que por la regla de la cadena:

$$\frac{H(X_1, \ldots, X_n)}{n} = \frac{1}{n} \sum_{i=1}^{n} H(X_i|X_1, \ldots, X_{i-1})$$

pero $\frac{1}{n} \sum_{i=1}^{n} H(X_i|X_1, \ldots, X_{i-1}) \to H'(\mathcal{X})$, considerando que $H(X_i|X_1, \ldots, X_{i-1}) \to H'(\mathcal{X})$. □

Notar que el corolario 2.6.5 sugiere una manera codificar un proceso estacionaro con un largo de código normalizado que se aproxima a la tasa de entropía. En efecto, $H(X_1)$ es la longitud ideal si codificamos el símbolo i con $\lg(1/p(i))$ bits. Seguidamente, una vez que codificamos X_1, el decodificador lo puede decodificar y conocer su valor, por lo tanto a la hora de codificar X_2, podemos aprovechar que conocemos X_1 y codificar idealmente j con $\lg(1/p(j|i))$ bits, para una longtiud promedio $H(X_2|X_1)$ y así sucesivamente.

2.7. Códigos para alfabetos infinitos

El resultado principal de esta sección será la prueba de existencia de códigos óptimos, libres de prefijo, para alfabetos de fuente infinitos (necesariamente numerables), cuando la entropía es finita. En esta sección nuestra variable aleatoria discreta X tomará, sin pérdida de generalidad, valores en $\mathbb{Z}_{>0} = \{1, 2, 3, \ldots\}$ con probabilidades $p_1 \geq p_2 \geq p_3 \geq \ldots$, todas positivas.

2.7.1. Códigos óptimos y la desigualdad de Kraft

Recordamos que para alfabetos finitos probamos el teorema 2.2.1 y teorema 2.2.2, que justificaban nuestra decisión de trabajar únicamente con código libres de prefijo. Para códigos infinitos, no es difícil de ver que estos teoremas se generalizan. En el caso del teorema 2.2.1 no es un problema que K sea la suma de una serie, y en el caso del teorema 2.2.2, podemos considerar que el código correspondiente a las palabras de código que tienen largo menor o igual a $M < \infty$ debe ser UD, y así $\sum_{i=1}^{M} \frac{n_i}{b^i} \leq 1$ para cada $M \in \mathbb{Z}_{>0}$, con lo cual se sigue $K \leq 1$.

Observando cuidadosamente la prueba del teorema 2.2.1, notamos que, para el *caso finito*, $K = 1$ implica que el árbol de prefijos que nos construimos es completo, y recíprocamente, un árbol completo implica la igualdad en la desigualdad de Kraft. En el caso de códigos binarios, sabemos que los códigos óptimos para fuentes de alfabetos finitos cumplen con igualdad la desigualdad de Kraft, ya que por el lema 2.5.1 deben ser árboles completos. Lo mismo no es cierto ya en el caso de códigos b-arios con $b > 2$, por ejemplo si el tamaño del alfabeto es $m \not\equiv 1 (\text{mód}. b - 1)$. En efecto, una simple inducción demuestra que todo árbol b–ario debe cumplir que su número de hojas m satisface $m \equiv 1 (\text{ mód }. b - 1)$.

En cualquier caso, si tenemos un alfabeto finito, tener igualdad en la desigualdad de Kraft resulta equivalente a que el árbol de prefijo correspondiente sea completo como discutimos arriba. Mostramos ahora que esto no es cierto en el caso alfabetos infinitos, ya que existen árboles completos de prefijo, tales que el código correspondiente no alcanzan la igualdad en la desigualdad de Kraft.

Ejemplo 2.7.1. Consideramos el código binario

$$\mathcal{C} = \{c_2 c_3 \ldots c_k 0^{k+1} : k \geq 1, c_i \in T_i \text{ para } i \in \{2, \ldots, k\}\} \tag{2.7.1}$$

donde $T_n = \mathbb{B}^n \setminus \{0^n\}$, que consiste de todas las palabras de largo n excepto $0^n = \underbrace{00 \ldots 00}_{n \text{ veces}}$.

Observar que para $k = 1$ tenemos simplemente $0^2 = 00$. Así pues, si ordenamos por cantidad de unos, el código comienza

$$00, 01000, 10000, 11000, 010010000, 010100000, 011000000, 100010000, 10010000, 10100000, \ldots,$$

y se encuentra ilustrado en la figura 2.7.1.

A continuación probamos que $\mathcal{C}$ es libre de prefijos, genera un árbol completo, pero satisface $K < 1$

Demostración. **El código $\mathcal{C}$ es libre de prefijos:** Supongamos, por absurdo, que tenemos $\alpha \neq \beta$ palabras de $\mathcal{C}$ con $\alpha = \beta q$ para algún $q \in \mathbb{B}^* \setminus \{\epsilon\}$ Escribimos $\alpha = \alpha_2 \alpha_3 \ldots \alpha_k 0^{k+1}$ para un cierto $k \geq 1$ y $\alpha_i \in T_i$, y $\beta = \beta_2 \beta_3 \ldots \beta_j 0^{j+1}$ para un cierto $j \geq 1$ y $\beta_i \in T_i$.

Como $\alpha = \beta q$, con $q \neq \epsilon$, α es más larga que β y por lo tanto $k > j$. Así $\alpha_2 \ldots \alpha_j \alpha_{j+1} \ldots \alpha_k 0^{k+1} = \beta_2 \ldots \beta_j 0^{j+1} q$ pero esto implica, comparando posiciones, que $\alpha_{j+1} = 0^{j+1}$, un absurdo.

El árbol asociado al código $\mathcal{C}$ es completo: Sea $\alpha \in \mathbb{B}^*$ prefijo de una palabra de código $c \in \mathcal{C}$. Mostramos ahora que $\alpha 0$ y $\alpha 1$ son prefijos de alguna palabra de código.

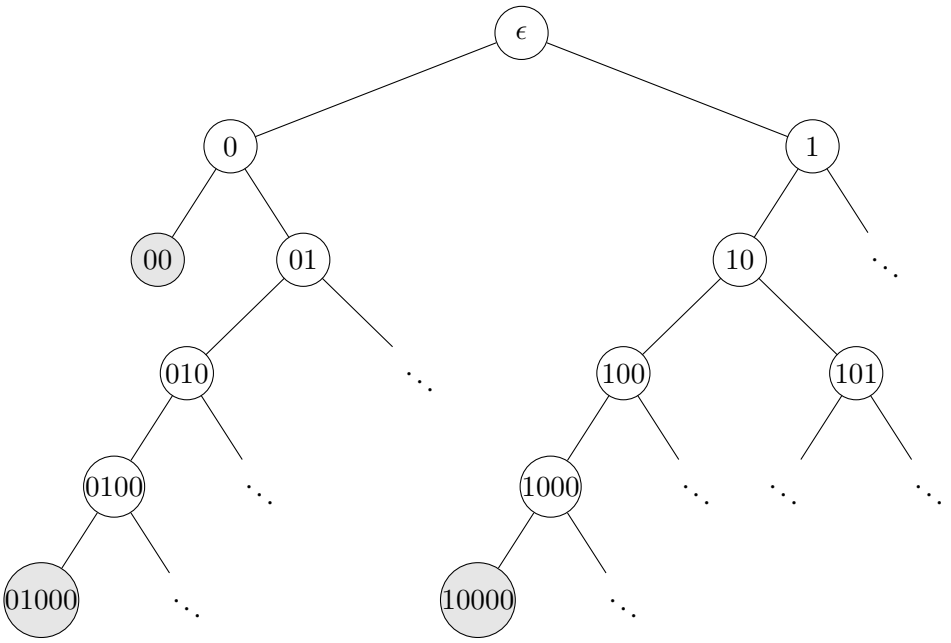


Figura 2.7.1: Representación gráfica de C.

Podemos escribir entonces $c = c_2c_3 \dots c_k 0^{k+1}$ con $c_i \in T_i$, para un cierto $k \geq 1$. Observar que el caso $k = 1$ corresponde únicamente a $c = 00$. Recordamos ahora que α es prefijo de c, y separamos en dos casos

I) **Caso** $\alpha = c_2 \dots c_t s$ con $t < k$ y s prefijo propio de c_{t+1} (posiblemente $s = \epsilon$):
En este caso, es claro que podemos completar $s1$ a una palabra T_{t+1}, y así $\alpha 1$ es prefijo de alguna palabra de código. El caso de $s0$ es similar, pero puede ocurrir que $s = 0^t$, en cuyo caso, si bien no podemos completar $s0$ a una palabra de T_{t+1}, tenemos directamente que $\alpha 0 = c_2 \dots c_t 0^{t+1} \in \mathcal{C}$.

II) **Caso** $\alpha = c_2 \dots c_k 0^t$ para algún $0 \leq t \leq k+1$:
Si $t \neq k+1$, estamos en el caso anterior. En efecto 0^t es prefijo propio de $\tilde{c}_{k+1} = 0^t 1^{k+1-t} \in T_{k+1}$, y α es prefijo de $c_2 \dots c_k \tilde{c}_{k+1} 0^{k+2} \in \mathcal{C}$. Así que resta $t = k+1$, pero en este caso $\alpha = c \in \mathcal{C}$.

No alcanza $K = 1$: A diferencia del caso finito, observamos que tenemos:

$$\sum_{w \in \mathcal{C}} 2^{-\ell(w)} = \frac{1}{4} + \sum_{k=2}^{\infty} \left(\prod_{n=2}^{k} \frac{2^n - 1}{2^n} \right) 2^{-k-1} < \frac{1}{4} + \sum_{k=2}^{\infty} 2^{-k-1} = \frac{1}{2} < 1 \,,$$

concluyendo entonces el ejemplo. ◊

Si bien los códigos binarios completos, libres de prefijo, para un alfabeto infinito, no necesariamente satisfacen con igualdad la desigualdad de Kraft, los que son óptimos para una fuente si cumplen la igualdad, como lo muestra el siguiente teorema.

Teorema 2.7.1. *Sea X una variable aleatoria discreta, con alfabeto infinito, luego cualquier código libre de prefijos b–ario óptimo debe satisfacer $K = 1$.*

Demostración. En efecto, supongamos que los largos de palabras correspondientes a nuestro código

de prefijo óptimo son $l_1 \leq l_2 \leq l_3 \ldots$ y que tenemos

$$K = \sum_{k=1}^{\infty} \frac{1}{b^{l_k}} < 1\,.$$

Sea entonces $\delta = 1 - \sum_{k=1}^{\infty} b^{-l_k} > 0$. Como necesariamente $b^{-l_k} \to 0$, existe $N \in \mathbb{Z}_{>0}$ tal que $b^{-l_k} < \delta$ si $k \geq N$. Sea $j \in \mathbb{Z}_{>0}$ con $l_j > l_N$, y definimos $\hat{l}_i = l_i$ para $i \neq j$, y $\hat{l}_j = l_N$. Se sigue entonces que

$$\sum_{k=1}^{\infty} b^{-\hat{l}_k} = \sum_{k=1}^{\infty} b^{-l_k} - b^{-l_j} + b^{-l_N} < \sum_{k=1}^{\infty} b^{-l_k} + \delta = 1\,.$$

Luego, por el teorema 2.2.1, existe un código de prefijo b-ario para esos parámetros. Sin embargo el largo promedio de nuestro nuevo código de prefijo es estrictamente menor que el de nuestro código original, contradiciendo su optimalidad. □

Nótese que en el teorema anterior se asume la existencia de un código óptimo, esto era evidente en el caso de alfabetos finitos, pero no es tan simple en el caso infinito. En la próxima sección probamos que tener $H(X) < \infty$ es una condición suficiente para la existencia de un código óptimo.

2.7.2. Prueba de existencia

La estrategia para la prueba consiste en aplicar el algoritmo de Huffman a una sucesión de fuentes finitas sobre alfabetos crecientes, y mostrar que una subsucesión de los códigos resultantes convergen en el sentido que definimos a continuación, a un código de prefijos, óptimo para la fuente infinita.

Definición 2.7.1 (*Convergencia de una sucesión de códigos*)**.** Decimos que una sucesión de códigos b-arios $C_1, C_2, \ldots$ para el alfabeto $\mathbb{Z}_{>0}$ *converge* a un código C si y solo si cada para cada i se tiene que $C_k(i)$ es constante para todo k suficientemente grande.

La definición anterior puede interpretarse está inspirada por el concepto de unión creciente. Ejemplo de esta relación es el siguiente lema, que utilizaremos en la prueba de existencia

Lema 2.7.1. *Consideramos una sucesión de códigos b-arios $C_1, C_2, \ldots$ para el alfabeto $\mathbb{Z}_{>0}$, que convergen a un código C. Si el código C_k reestringido a $\{1, \ldots, k\}$ es libre de prefijos para cada $k \geq 1$, entonces C es libre de prefijos.*

Demostración. Por absurdo, si C no fuera libre de prefijos existen enteros positivos $i \neq j$, tales que $C(i)$ es prefijo de $C(j)$. Por la definición de convergencia, existe un $N \geq \max\{i, j\}$ tal que $C_k(i) = C(i)$ y $C_k(j) = C(j)$ para todo $k \geq N$. Esto último es un absurdo, ya que por hipótesis C_N reestringido a $\{1, \ldots, N\}$ era libre de prefijos. □

Volviendo a la prueba del teorema de existencia, consideramos $X^{(n)}$ que toma valores en $\{1, \ldots, n\}$, con la distribución $\mathbf{p}^{(n)} = \left(p_1^{(n)}, \ldots, p_n^{(n)}\right)$ donde $p_i^{(n)} = \frac{p_i}{S_n}$, y $S_n = \sum_{i=1}^{n} p_i$ es la constante de normalización. Sea c el código resultante de aplicar Huffman a $X^{(n)}$. Decimos entonces que $C_n \colon \mathbb{Z}_{>0} \to \mathbb{B}^*$ definido mediante

$$C_n(k) = \begin{cases} c(k), & \text{si } k \leq n \\ \epsilon, & \text{si } k > n, \end{cases} \tag{2.7.2}$$

es el *código de Huffman truncado* n–ésimo para X.

Teorema 2.7.2. *Sea X una variable aleatoria discreta que, sin pérdida de generalidad, toma valores en $\mathbb{Z}_{>0}$. Supongamos que $H(X) < \infty$, y sea $b > 1$, luego tenemos:*

(a) Existe una subsucesión de códigos b–arios de Huffman truncados para X, tal que converge a un código libre de prefijos que es óptimo para X.

(b) El largo promedio de los códigos truncados para X converge al largo promedio óptimo.

Demostración. La prueba de (a) se basará en el famoso *argumento diagonal* de Cantor y el principio del palomar.

Sea C_n el n-ésimo código b-ario de Huffman truncado para X, y sean

$$\{l_1^{(n)}, \ldots, l_n^{(n)}, 0, \ldots\}$$

las correspondientes longitudes.

Por el corolario 2.4.2 sabemos que $\sum_{k=1}^{n} l_k^{(n)} p_k^{(n)} \leq H\left(X^{(n)}\right) + 1$ y, por definición, tenemos

$$H\left(X^{(n)}\right) = -\sum_{k=1}^{n} p_k^{(n)} \lg\left(p_k^{(n)}\right) = -\frac{1}{S_n} \sum_{k=1}^{n} p_k \lg\left(p_k\right) + \lg S_n \,,$$

es decir

$$H\left(X^{(n)}\right) = \frac{1}{S_n} H(X) + \lg S_n \,,$$

donde $S_n = p_1 + \ldots + p_n \to 1$. Luego se sigue que para todo n suficientemente grande se tiene

$$H\left(X^{(n)}\right) \leq H(X) + 1 \,,$$

y, por lo tanto

$$l_i^{(n)} p_i \leq \sum_{k=1}^{\infty} l_k^{(n)} p_k \leq \left(H\left(X^{(n)}\right) + 1\right) S_n \leq H(X) + 2 \,.$$

Así pues, para todo n suficientemente grande se tiene:

$$l_i^{(n)} \leq \frac{H(X) + 2}{p_i} < \infty, \forall i \in \mathbb{Z}_{>0}$$

Se sigue que si $N = \left\lfloor \frac{H(X) + 2}{p_i} \right\rfloor$, donde $\lfloor \cdot \rfloor$ denota la parte entera, se cumple $\{C_n(i)\}_{n=1}^{\infty} \subset \mathcal{B}^0 \cup \ldots \mathcal{B}^N$, donde $\mathcal{B}$ es nuestro alfabeto b-ario de salida. Como $|\mathcal{B}| = b < \infty$ esto implica que $\{C_n(i)\}_n$ solo puede tomar un número finito de palabras de código. Siendo esta una secuencia infinita, debe tener una subsucesión constante, i.e. hay una palabra que aparece infinitas veces en $\{C_n(i)\}_n$.

Sea entonces $n_k^{(1)}$ sucesión creciente tal que $C_{n_k^{(1)}}(1)$ es constante en k. Repitiendo ahora, sea $n_k^{(2)}$ subsucesión de $n_k^{(1)}$ tal que $C_{n_k^{(2)}}(2)$ es constante en k, y así sucesivamente. Considerando la sucesión diagonal $m_k = n_k^{(k)}$ obtenemos que $C_{m_k}(r)$ es constante para $k \geq r$. Por definición C_{m_k} es convergente, y denotamos por $\hat{C}$ su código límite.

Observar que $m_k \geq k$, entonces, por el lema 2.7.1, aplicado a $C_{m_1}, C_{m_2}, \ldots$, sabemos que $\hat{C}$ es libre de prefijos. Para terminar con *(a)*, probaremos que en efecto $\hat{C}$ es óptimo para X. Sean $\{\lambda_1, \lambda_2, \ldots\}$

las longitudes correspondientes a $\hat{C}$, y sean $\{\alpha_1, \alpha_2, \ldots\}$ las longitudes correspondientes a otro código para X. Por la optimalidad de C_n para $X^{(n)}$, tenemos:

$$\begin{aligned}\sum_{k=1}^{n} l_k^{(n)} p_k^{(n)} &\leq \sum_{k=1}^{n} \alpha_k p_k^{(n)} \\ &= S_n^{-1} \sum_{k=1}^{n} \alpha_k p_k \\ &\leq S_n^{-1} \sum_{k=1}^{\infty} \alpha_k p_k \,. \end{aligned} \tag{2.7.3}$$

Sea $i \in \mathbb{Z}_{>0}$ arbitrario, como $\lambda_k = l_k^{(m_i)}$ para $k = 1, \ldots, i$, tenemos

$$S_{m_i}^{-1} \cdot \sum_{k=1}^{i} \lambda_k p_k = \sum_{k=1}^{i} \lambda_k p_k^{(m_i)} = \sum_{k=1}^{i} l_k^{(m_i)} p_k^{(m_i)} \,.$$

Al ser $m_i \geq i$, esto implica que

$$S_{m_i}^{-1} \cdot \sum_{k=1}^{i} \lambda_k p_k = \sum_{k=1}^{i} l_k^{(m_i)} p_k^{(m_i)} \leq \sum_{k=1}^{m_i} l_k^{(m_i)} p_k^{(m_i)} \,, \tag{2.7.4}$$

y por (2.7.3) con $n = m_i$ se sigue

$$S_{m_i}^{-1} \cdot \sum_{k=1}^{i} \lambda_k p_k \leq \sum_{k=1}^{m_i} l_k^{(m_i)} p_k^{(m_i)} \leq S_{m_i}^{-1} \sum_{k=1}^{\infty} \alpha_k p_k \,,$$

es decir $\sum_{k=1}^{i} \lambda_k p_k \leq \sum_{k=1}^{\infty} \alpha_k p_k$, y como i es arbitrario, tomando $i \to \infty$ se sigue que $\hat{C}$ es óptimo.

Finalmente probaremos *(b)*. La estrategia para esta parte consiste en acotar $\sum_{k=1}^{n} l_k^{(n)} p_k^{(n)}$, por arriba y por abajo, con sucesiones que tiendan a la longitud óptima $\sum_{k=1}^{\infty} \lambda_k p_k$.

Para acotar por arriba, aplicando (2.7.3) con $\alpha_k = \lambda_k$ tenemos

$$\sum_{k=1}^{n} l_k^{(n)} p_k^{(n)} \leq S_n^{-1} \sum_{k=1}^{\infty} \lambda_k p_k \,, \tag{2.7.5}$$

Ahora procedemos a acotar por abajo. Si $i \in \mathbb{Z}_{>0}$ es arbitrario y n satisface $n \geq m_i$, tenemos por (2.7.4)

$$\sum_{k=1}^{i} \lambda_k p_k^{(m_i)} \leq \sum_{k=1}^{m_i} l_k^{(m_i)} p_k^{(m_i)} \leq \sum_{k=1}^{m_i} l_k^{(n)} p_k^{(m_i)} \,, \tag{2.7.6}$$

donde la igualdad última desigualdad se sigue de que C_{m_i} es óptimo cuando nos reestringimos a $\{1, \ldots, m_i\}$.
Como $n \geq m_i$ tenemos $\sum_{k=1}^{m_i} l_k^{(n)} p_k^{(m_i)} \leq \sum_{k=1}^{n} l_k^{(n)} p_k^{(m_i)}$, y junto con (2.7.6) esto implica

$$\sum_{k=1}^{i} \lambda_k p_k^{(m_i)} \leq \sum_{k=1}^{n} l_k^{(n)} p_k^{(m_i)} = \frac{S_n}{S_{m_i}} \sum_{k=1}^{n} l_k^{(n)} p_k^{(n)} \,,$$

es decir

$$S_n^{-1} \sum_{k=1}^{i} \lambda_k p_k \leq \sum_{k=1}^{n} l_k^{(n)} p_k^{(n)} \,, \tag{2.7.7}$$

para $i \geq 1$ arbitrario y $n \geq m_i$. Tomamos aquí $i_n = \max\{i \in \mathbb{Z}_{>0} \colon m_i \leq n\}$, por (2.7.5) y (2.7.7) tenemos entonces

$$S_n^{-1} \sum_{k=1}^{i_n} \lambda_k p_k \leq \sum_{k=1}^{n} l_k^{(n)} p_k^{(n)} \leq S_n^{-1} \sum_{k=1}^{\infty} \lambda_k p_k \,, \tag{2.7.8}$$

y estamos hechos. En efecto como m_i es creciente, se sigue por definición que $i_n \to \infty$ cuando $n \to \infty$, y, por otro lado, tenemos $S_n \to 1$, concluyendo que ambos extremos de (2.7.8) tienden a $\sum_{k=1}^{\infty} \lambda_k p_k$. □

2.8. Comentarios finales del capítulo

Probamos recién que $H(X) < \infty$ implica la existencia de códigos libres de prefijo óptimos, pero, a pesar de que podemos aproximar su largo promedio, no tenemos todavía forma de construirlos, ya que la prueba recién vista es no constructiva. En efecto, en la práctica no hay forma de determinar una sucesión $\left(n_k^{(1)}\right)_{k=1}^{\infty}$ tal que $C_{n_k^{(1)}}(1)$ sea constante, ya que requeriría determinar una palabra de código que aparezca infinitas veces en $(C_1(1), C_2(1), \ldots)$, y esto en principio no puede saberse mirando únicamente una cantidad finita de elementos.

Existen, no obstante, construcciones de códigos de prefijos óptimos cuando las colas $\sum_{n \geq k+1} p_n$ cumplen ciertas propiedades, este es el enfoque en [8, 10]. Dichas construcciones preceden la prueba de existencia, que apareció por primera vez en [7], y su optimalidad es probada utilizando la técnica introducida en [12], que se basa en la codificación de Huffman explicada en este capítulo. En nuestro caso estaremos interesados en un caso más particular de distribución $\mathbf{p}$, que es la geométrica, y realizaremos un estudio detallado de códigos óptimos para estas, utilizando también la técnica introducida en [12].

El teorema 2.7.2 no solo sirve para justificar la búsqueda de códigos óptimos, sino que también en algunos casos justificará razonamientos del tipo *cualquier código de prefijos óptimo debe tener un cierto tipo de estructura en sus niveles más altos*, y trabajar con las aproximaciones, en lugar del código óptimo en sí.

Finalmente, como vimos en el ejemplo 2.6.2, si codificamos pares de variables iid, en lugar de cada una por separado, existen posibilidades de ganancias y acercarnos más a la entropía. Concretamente, el corolario 2.4.2 nos indica que el *bit extra* ahora se divide entre 2 símbolos en lugar de 1. Así pues, cuando estudiemos la codificación de variables geométricas, comenzaremos viendo códigos óptimos para una variable aislada [11], y luego para pares de variables geométricas.

CAPÍTULO 3

Codificación de variables aleatorias geométricamente distruídas

3.1. Códigos de Golomb

Sean $q \in (0,1)$ y X una variable aleatoria, que toma valores en $\mathbb{Z}_{\geq 0} = \{0, 1, 2, \ldots\}$, con

$$\Pr(X = n) = (1-q)\, q^n \,.$$

Es decir, su fuente correspondiente $(\mathcal{A}, \mathbf{p})$ es

$$\mathcal{A} = \mathbb{Z}_{\geq 0}, \quad p(n) = (1-q)\, q^n \,, \tag{3.1.1}$$

y denotamos mediante $ODGD(q)$ a esta distribución.

Esta X se puede interpretar representa el número de intentos (p.e. tiradas de una moneda), independientes, hasta obtener un evento favorable (p.e. obtener cara), que tiene probabilidad $p = 1 - q$ de ocurrir en cada intento.

Aplicando la definición de entropía tenemos

$$H(X) = -(1-q) \cdot \sum_{n=0}^{\infty} q^n \left(n \cdot \lg(q) + \lg(1-q)\right) \,,$$

recordando ahora $\sum_{n=0}^{\infty} q^n = \dfrac{1}{1-q}$ y $\sum_{n=0}^{\infty} n \cdot q^n = \dfrac{q}{(1-q)^2}$, se tiene

$$H(X) = \frac{h(q)}{1-q} \,, \tag{3.1.2}$$

donde $h(q) \triangleq H(\{q, 1-q\}) = q \lg(1/q) + (1-q) \lg\left(\frac{1}{1-q}\right)$ es la denominada *entropía binaria*.

Como $H(X) < \infty$, por el teorema 2.7.2 se sigue que existe un código binario libre de prefijo óptimo para X. Resulta interesante el problema de caracterizar explícitamente este código, no solo por su elegancia, sino también por su utilidad en la compresión sin pérdida de imágenes de tono de gris.

Aclaración: En lo sucesivo utilizamos la palabra código para decir código libre de prefijos, y los códigos tienen alfbabeto de salida binario $\mathbb{B} = \{0, 1\}$.

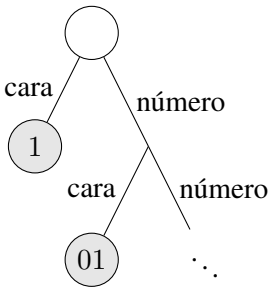


Figura 3.1.1: Codificación para $q = 1/2$.

3.1.1. Moneda justa

Primero veamos el caso en que $q = 1/2$, es decir, tenemos una moneda justa (no cargada). Mediante un simple cálculo utilizando (3.1.2) obtenemos

$$H(X) = \frac{h(1/2)}{1 - 1/2} = 2\,.$$

Pensemos ahora un código libre de prefijo óptimo para X. Tiramos la moneda una vez. Que salga cara resulta tan probable como que salga número, por simetría entonces, es razonable codificar con la misma cantidad de símbolos la ocurrencia de una cara como la ocurrencia de un número. Escribimos entonces un 1 de ocurrir cara, y un 0 de ocurrir número y continuamos hasta obtener cara (ver la figura 3.1.1). La longitud esperada con dicho código es

$$L = \sum_{n \geq 0} (n+1) \cdot q^n \cdot (1-q) = \frac{1}{1-q} = 2\,,$$

y, por lo tanto, resulta óptimo ya que alcanza la entropía.

En resumen, para $q = 1/2$ tenemos lo que se denomina una codificación unaria; para codificar $X = n$ escribimos n ceros seguidos por un uno.

3.1.2. Caso general

Motivación y estrategia

Para $q \neq 1/2$, cada vez que estamos parados en un nodo interno, la probabilidad de ir a izquierda o derecha ya no es la misma. Inspirados entonces por la idea de balancear las probabilidades del árbol, probamos tomar de a varias tiradas a la vez.

Consideramos $b \in \mathbb{Z}_{>0}$ tal que

$$\Pr(X < b) \approx \Pr(X \geq b)\,,$$

donde el significado preciso de esto se dara en breve. Nótese que en el caso $q = 1/2$ tomamos $b = 1$, y miramos de a *un* evento. Ahora lo que codificaremos primero es si $X < b$ o $X \geq b$. Si $X \geq b$, entonces repetimos el proceso con los *siguientes* b pasos, esto es $b \leq X < 2b$ o $X \geq 2b$, y así sucesivamente.

El árbol resultante se visualiza en la Figura 3.1.2 y es relativamente balanceado. Sin embargo, esto no resulta suficiente para codificar X. Una vez que conocemos $m \in \mathbb{Z}_{\geq 0}$ tal que

$$b \cdot m \leq X < b \cdot (m+1)\,,$$

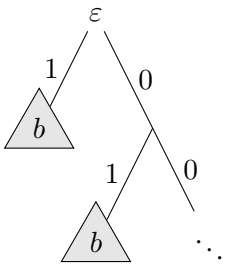


Figura 3.1.2: Estrategia general.

debemos codificar la diferencia $X - b \cdot m$. Observar que m es el cociente de dividir X entre b y $r = X - b \cdot m$ es el resto.

Así pues, el resto de dividir el resultado del experimento X lo codificamos en binario (utilizando $\lfloor \log b \rfloor$ bits o $\lceil \log b \rceil$ bits[a]), tras codificar el cociente en unario.

Elección de b

Notamos que

$$\Pr(X < b) = \sum_{n<b} q^n \cdot (1-q) = 1 - q^b \,,$$

por la famosa suma geométrica. Como $\Pr(X < b) + \Pr(X \geq b) = 1$, nos gustaría tener $1 - q^b = 1/2$, sin embargo esto rara permite tener $b \in \mathbb{Z}_{>0}$.

No obstante, observamos que $\Pr(X < b) = 1 - q^b$ y $\Pr(X \geq b) = q^b$, donde la primera probabilidad crece hacia 1 al aumentar $b \in \mathbb{Z}_{>0}$ y la segunda decrece hacia 0. Existe por lo tanto un *punto de cruce*, es decir, un $b \in \mathbb{Z}_{>0}$ tal que $q^b \geq 1 - q^b$ pero $q^{b+1} < 1 - q^{b+1}$.

Por razones que resultan más claras luego, tomamos el *punto de cruce* de $\Pr(X < b) = 1 - q^b$ y $\Pr(X > b) = q^{b+1}$. Tomamos entonces el menor $b \in \mathbb{Z}_{>0}$ tal que $\Pr(X > b) \leq \Pr(X < b)$, dicho de otro modo, tomamos b tal que

$$q^{b+1} + q^b \leq 1 < q^b + q^{b-1} \,. \tag{3.1.3}$$

El hecho de tomar el punto de corte con las desigualdades estrictas $X < b$ y $X > b$ puede resultar un poco arbitrario, pero las desigualdades (3.1.3) resultan importantes en la prueba de optimalidad.

Prueba de optimalidad

La técnica que utilizaremos para probar la optimalidad fue publicada por primera vez por Gallager y van Voorhis [12]. La idea consiste en agrupar conjuntos infinitos de símbolos de manera adecuada, formando fuentes finitas especiales, que en [12] se denominan `fuente reducidas` finita, que coincidan en cierta medida con la original para luego aplicar el algoritmo de Huffman sobre estas fuentes finitas. La prueba de optimalidad de los códigos de Golomb ejemplifica esta técnica, que también utilizaremos posteriormente en el caso de los códigos bidimensionales.

Para este caso, nuestra fuente reducida m-ésima, también denominada fuente m-reducida, es la siguiente:

[a] Esto se explica posteriormente, cuando veamos los códigos óptimos para la distribución cuasi-uniforme.

- Los símbolos son $\{0, 1, \ldots, m-1, m, \ldots, m+b-1\}$, y los denominaremos símbolos reducidos para distinguirlos de los de la fuente original.
- Denotamos mediante $\mathbf{p}_m$ su vector de probabilidades. Definimos $p_m(i) = p(i) = q^i(1-q)$ para $i \in \{0, \ldots, m-1\}$, mientras que las probabilidades de los demás símbolos agrupan las probabilidades $\{p(i) : i \geq m\}$ de la siguiente forma

$$p_m(i+m) = \Pr\left(X > m,\ X \equiv i+m\,(\text{mód}.b)\right) = \sum_{k=0}^{\infty} q^{i+m+b\cdot k}(1-q) = \frac{q^{i+m}\cdot(1-q)}{1-q^b}\,,$$

para $0 \leq i \leq b-1$. Notar que estas corresponden a sucesos disjuntos, cuya unión es $X \geq m$.

Lema 3.1.1. *Los símbolos reducidos de menor probabilidad en la fuente m-reducida son $m-1$ y $m+b-1$.*

Demostración. Notamos que $p_m(i)$ es claramente decreciente en $i \in \{0, \ldots, m-1\}$, y también, que $p_m(i)$ es decreciente en $i \in \{m, \ldots, m+b-1\}$. Luego para probar que los de menor probabilidad son $m-1$ y $m+b-1$, basta probar que $p_m(m-1) \leq p_m(m+b-2)$ y que $p_m(m+b-1) \leq p_m(m-2)$.

En efecto, por (3.1.3) se tiene $1 < q^b + q^{b-1}$ y luego $1 < \dfrac{q^{b-1}}{1-q^b}$, por lo tanto

$$p_m(m-1) = q^{m-1}(1-q) < \frac{q^{m-1+b-1}(1-q)}{1-q^b} = p_m(m+b-2)\,.$$

De manera similar, utilizando $q^b + q^{b+1} \leq 1$, se tiene la otra desigualdad. □

Ahora, con este resultado en mente, resulta más claro lo que sucede al aplicar Huffman. Nótese que la desigualdad (3.1.3) fue fundamental. A continuación caracterizamos entonces el código óptimo para la fuente m-reducida.

Definición 3.1.1 (*Códigos de Golomb*)**.** Dado $k \in \mathbb{Z}_{>0}$, definimos el código de Golomb G_k como el código unario $G_1(n) = 0^n 1$ para $k = 1$, y

$$G_k(n) = T_k\left(n \bmod .k\right) \bowtie G_1\left(\left\lfloor \frac{n}{k} \right\rfloor\right)$$

en otro caso. Aquí T_k es un código óptimo para una fuente finita con los siguientes símbolos y probabilidades

$$\hat{\mathcal{A}}_k = \{i : 0 \leq i < k\}, \quad p_i = q^i \times \frac{1-q}{1-q^k}\,, \tag{3.1.4}$$

donde $q \in (0,1)$ es cualquier número real que satisface (3.1.3) con $b = k$. La codificación T_k no depende del valor de q elegido en dichas condiciones, lo cual se demuestra en la proposición 3.1.1 más adelante.

Dado $m \geq 0$, consideramos el código $G_{b,m}\colon \{0, \ldots, m+b-1\} \to \mathbb{B}^*$ definido por

$$G_{b,m}(i) = \begin{cases} G_b(i), & \text{si } i < m\,; \\ x \cdot 0, & \text{si } i \geq m \text{ donde } G_b(i-b) = x \cdot 1\,, \end{cases}$$

el siguiente lema prueba que $G_{b,m}$ es óptimo para la fuente m–reducida.

Lema 3.1.2. *Consideramos $m \geq 0$. El código $G_{b,m}$ es óptimo para la fuente m–reducida.*

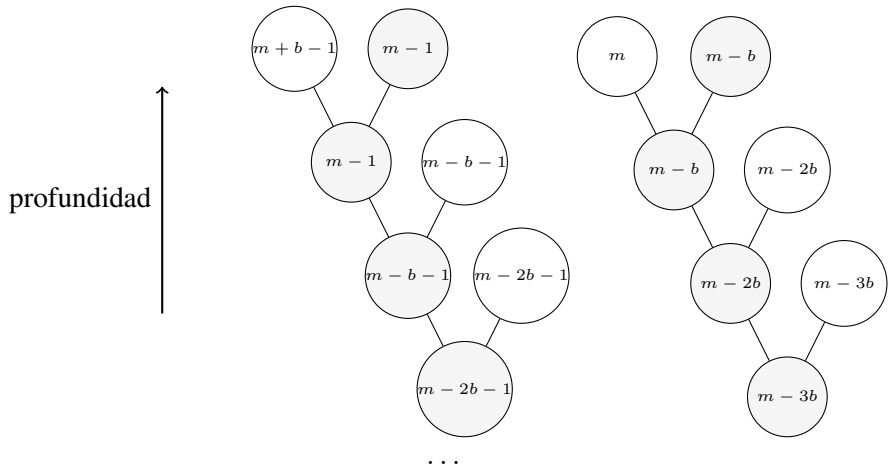


Figura 3.1.3: Ejemplo de los árboles que se forman para los restos $\{b-1, 0\}$ en módulo b. El nivel más alto corresponde a la fuente reducida m–ésima, el siguiente de arriba hacia abajo corresponde a la fuente $m-b$–reducida, y así sucesivamente. Se encuentra marcado con color el camino reverso que se sigue para codificar $m-1$ y para codificar $m-b$.

Demostración. Empezando con la fuente m-reducida, al aplicar Huffman podemos primero hacer un merge de los símbolos $m-1$ y $m+b-1$, para obtener uno de probabilidad

$$q^{m-1}(1-q) + \sum_{k=0}^{\infty} q^{m+b\cdot k+b-1}(1-q) = \sum_{k=0}^{\infty} q^{m-1+b\cdot k}(1-q) = p_{m-1}(m-1)\,,$$

que pasaremos a llamar simplemente $m-1$ en la fuentes subsiguientes, es importante sin embargo recordar los merges realizados.

Notamos que lo que tenemos en este punto es en realidad la fuente $(m-1)$–reducida. Nuevamente por el lema 3.1.1 podemos tomar, para el siguiente merge , a $m-2$ y $m+b-2$ para generar un símbolo de probabilidad $p_{m-2}(m-2)$ en el lugar de estos.

Este proceso se repite, hasta llegar a la fuente 0–reducida:

$$S^{\star} = \left\{ \frac{q^0}{1-q^b} \cdot (1-q), \ldots, \frac{q^{b-1}}{1-q^b} \cdot (1-q) \right\}.$$

Notar que los merges realizados corresponden a la codificación unaria del cociente, como se encuentra ilustrado en la figura 3.1.3. Los símbolos en $S^{\star}$, ahora, corresponden a los restos módulo b, y se codifican por el código T_b de la definición de los códigos de Golomb. □

Teorema 3.1.1. *Sea $q \in (0,1)$ y b el único entero positivo que satisface (3.1.3). Si X tiene distribución $OSGD(q)$, tenemos que G_b, definido según la definición 3.1.1, es un código de prefijo óptimo para X.*

Demostración. Consideremos $m \geq 0$. Sean $L(G_{b,m})$ la longitud promedio del código $G_{b,m}$, $L(G_b)$ la longitud promedio del código de Golomb G_b, y $L(C)$ la longitud de un código C arbitrario para nuestra distribución geométrica que satisface $L(C) \leq L(G_b)$. Nuestro objetivo es probar que $L(C) = L(G_b)$, probando entonces que $L(G_b)$ es mínimo.

Afirmamos, ahora, que se cumple

$$L(G_{b,m}) \leq L(C)\,, \tag{3.1.5}$$

para cada $m \geq 0$. En efecto, si $\lambda_k = \ell(C(k))$ escribimos

$$L(C) = \sum_{k=0}^{\infty} \lambda_k p_k = \sum_{k=0}^{m-1} \lambda_k p_k + \sum_{i=0}^{b-1} \sum_{k=0}^{\infty} p_{i+m+b\cdot k} \lambda_{i+m+b\cdot k} \, .$$

Definimos $\lambda'_i = \lambda_i$ si $i < m$ y $\lambda'_{i+m} = \min\{\lambda_{i+m+b\cdot k} : k \in \mathbb{Z}_{\geq 0}\}$, el mínimo en la clase, para $i \in \{0, \ldots, b-1\}$. Como nos podemos generar un código libre de prefijos para la fuente m–reducida, a partir de C, considerando la palabra más corta para representar cada clase de la partición $\{\{i\} : i < m\} \cup \{\{i + m + b \cdot k : k \in \mathbb{Z}_{\geq 0}\} : i \geq m\}$:

$$\begin{aligned}
L(G_{b,m}) &\leq \sum_{k=0}^{m} \lambda_k p_k + \sum_{i=1}^{b} p_m(i+m) \ \lambda'_{i+m} && \text{(Optimalidad de } G_{b,m}) \\
&= \sum_{k=0}^{m} \lambda_k p_k + \sum_{i=1}^{b} \sum_{k=0}^{\infty} p_{i+m+b\cdot k} \lambda'_{i+m} && \text{(Definición } p_m(i+m)) \\
&\leq \sum_{k=0}^{m} \lambda_k p_k + \sum_{i=1}^{b} \sum_{k=0}^{\infty} p_{i+m+b\cdot k} \lambda_{i+m+b\cdot k} && \text{(Definición } \lambda'_{i+m+b\cdot k}) \\
&= L(C) \, , && \text{(Definición } \lambda_k)
\end{aligned}$$

probando entonces nuestra afirmación.

Tenemos $\sum_{k=0}^{m-1} \ell_k p_k \leq L(G_{b,m})$ donde ℓ_k es la longitud de $G_{b,m}(k)$, por definición coincide exactamente con la de G_b siempre que $m > k$, y p_k es ya la probabilidad de k en la fuente infinita, que coincide con su probabilidad en la fuente reducida si $m > k$. Por lo tanto tenemos

$$\sum_{k=0}^{m-1} \ell_k p_k \leq L(G_{b,m}) \leq L(C) \leq L(G_b) \, .$$

Como tenemos $\lim_{m\to\infty} \sum_{k=0}^{m-1} \ell_k p_k = L(G_b)$, concluímos que $L(C) = L(G_b)$. □

Para completar la sección, se presente a continuación una caracterización de los códigos de prefijo óptimos para distribuciones cuasi-uniformes, explicando cómo se utilizan entonces para codificar G_k.

Códigos de prefijo óptimos para distribuciones cuasi-uniformes

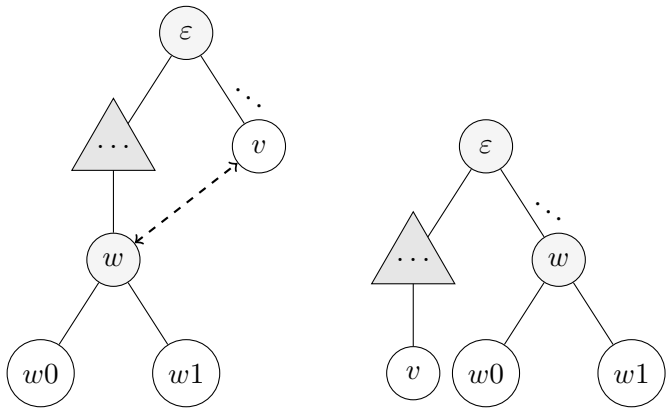
Definición 3.1.2. Decimos que una distribución discreta $\mathbf{p} = (p_1, \ldots, p_m)$ (asumimos $p_1 \geq p_2 \ldots \geq p_m$) es cuasi-uniforme si y sólo si

$$p_{m-1} + p_m > p_1 \, .$$

Observación 3.1.1. La fuente con los siguientes símbolos y probabilidades

$$\hat{\mathcal{A}}_k = \{i : 0 \leq i < k\}, \quad p_i = q^i \times \frac{1-q}{1-q^k} \, , \tag{3.1.6}$$

que aparece en la definición de los códigos de Golomb, es cuasi-uniforme cuando k es el parámetro óptimo para $q \in (0, 1)$.



(a) Antes (b) Después

Figura 3.1.4: Transformación.

En efecto, la suma de las dos probabilidades más pequeñas es

$$p_{k-2} + p_{k-1} = q^{k-2}\,\frac{1-q}{1-q^k} + q^{k-1}\,\frac{1-q}{1-q^k}\,,$$

que, por (3.1.3), es mayor que la probabilidad del símbolo más probable $p_0 = \frac{1-q}{1-q^k}$.

Proposición 3.1.1. *Sea X una variable aleatoria con distribución cuasi-uniforme $\{p_1, \ldots, p_m\}$ ($p_i \geq p_{i+1}$). El código óptimo para esta fuente está dado por $2^{d+1} - m$ palabras de longitud $d = \lfloor \lg(m) \rfloor$ y $2m - 2^{d+1}$ palabras de longitud $d + 1$.*

Demostración. Sea $\mathcal{C}$ un código de prefijo óptimo. Recordamos primero que, por el lema 2.5.1, las palabras de longitud máxima vienen de a pares que difieren en el último bit.

A continuación demostramos que no pueden haber dos hojas en el árbol de prefijo correspondiente cuyas profundidades difieran en más de uno. Sean $w0, w1 \in \mathcal{C}$ las palabras de longitud máxima (y probabilidad mínima sin pérdida de generalidad, por el lema 2.5.1), y $v \in \mathcal{C}$ de profundidad mínima (y de probabilidad máxima sin pérdida de generalidad). Asumimos, por absurdo, que $\ell(w0) - \ell(v) \geq 2$, esto es, que $\ell(w) - \ell(v) \geq 1$.

Realizamos ahora la transformación que se detalla en la figura 3.1.4, intercambiando los nodos v y w (junto con sus hijos). El cambio de longitud media es

$$\Delta L = \Big((\ell(v)+1)\cdot(p_{m-1}+p_m) + \ell(w)\cdot p_1\Big) - \Big(\ell(v)\cdot p_1 + (\ell(w)+1)\cdot(p_{m-1}+p_m)\Big)\,,$$

que, simplificando, resulta

$$\Delta L = (\ell(v) - \ell(w))\cdot(p_{m-1} + p_m - p_1) < 0\,,$$

lo cual es absurdo ya que nuestro código $\mathcal{C}$ era óptimo.

Sea d la profundidad mínima. Acabamos de probar que todos los nodos están a profundidad, o bien d o $d+1$. Notamos que, al ser nuestro árbol óptimo, ha de contener un árbol completo de profundidad d, pero no uno de $d+1$ (de otro modo no tendríamos nodos a profundidad d), luego $2^d \leq m < 2^{d+1}$ y así $d = \lfloor \lg(m) \rfloor$. También, sabemos que la cantidad de nodos a profundidad $d+1$ es par, sea esta 2β.

Notamos que $2^d + \beta = m$ y así $\beta = m - 2^d$, ya que, borrando uno de cada uno de los pares de nodos a profundidad $d+1$ y subiendo el restante un nivel debemos tener un árbol completo de profundidad

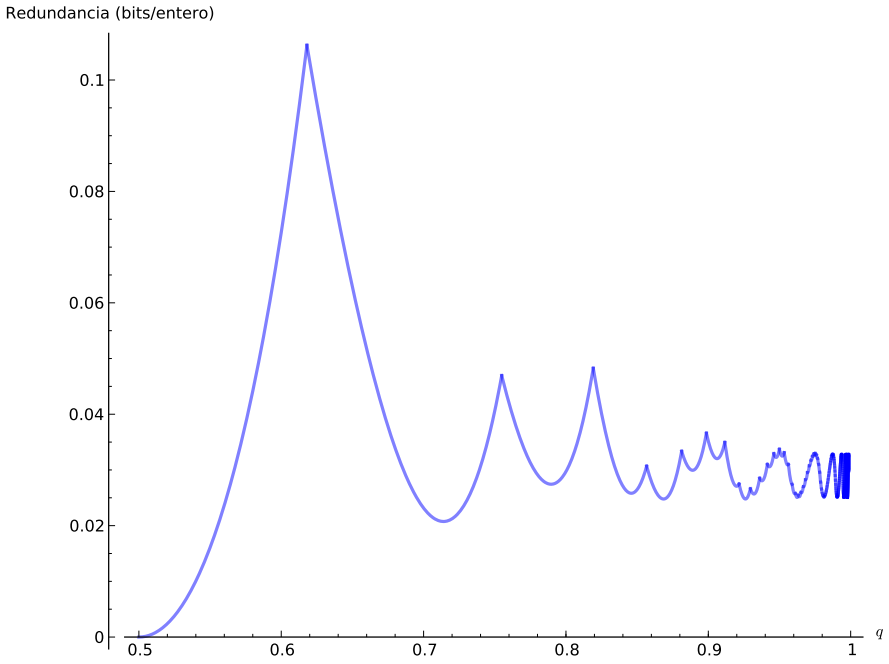


Figura 3.1.5: Redundancia de los códigos de Golomb óptimos para $q \in [0.5, 1]$.

d. Se sigue de inmediato entonces que el número de palabras a profundidad d ha de ser $m - 2\beta = 2^{d+1} - m$. □

Observación 3.1.2. Si $m = 2^k$ para algún k, entonces $d = k$ y todos los nodos están a profundidad k. Luego en este caso el código se corresponde a todas las palabras binarias de longitud k. Como corolario resulta entonces que el código de Golomb es especialmente sencillo de codificar/decodificar si b es una potencia de 2. A estos códigos de Golomb se les llama *códigos de Rice*.

Observación 3.1.3. Si tenemos $p_1 = p_m + p_{m-1}$, el resultado sigue siendo cierto. Si bien en este caso no se produce un absurdo, la transformación de la figura 3.1.4, aplicada mientras el árbol tenga hojas que difieran en más de un nivel, nos lleva a un árbol de dos niveles, sin aumentar la longitud promedio.

3.1.3. Redundancia de los códigos de Golomb

Recordamos que la redundancia indica cuánto difiere la longitud promedio de la longitud promedio ideal $H(X)$. Recordamos primero que la entropía de una variable aleatoria geométrica, con $\Pr(X = n) = (1-q)\,q^n$, es $H(X) = \dfrac{h(q)}{1-q}$. Por otra parte, la longitud promedio de Golomb G_k para q es

$$L_q(G_k) = d + 1 + \frac{q^{2^{d+1}-k}}{1-q^k}$$

donde $d = \lfloor \lg(k) \rfloor$. Esto se sigue de calcular

$$L_q(G_k) = (1-q) \times \sum_{n=0}^{\infty} \sum_{r=0}^{k-1} (n + 1 + f(r)) \cdot q^{n \cdot k + r}$$

donde $f(r)$ es d si $r < 2^{d+1} - k$, y $d+1$ en otro caso, de acuerdo a la proposición 3.1.1. En la figura 3.1.5 se muestra la redundancia del código óptimo (obteniendo el parámetro por medio de (3.1.3)).

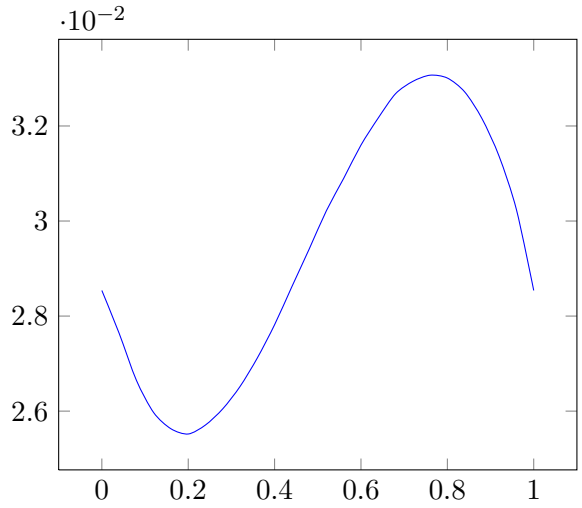


Figura 3.1.6: $R(x) = 1 - x + 4 \times 2^{-2^{1-x}} - C$

Comportamiento límite

Es interesante observar en la figura 3.1.5 que la redundancia por un lado parece volverse más *regular* cuando $q \to 1$, y además parece seguir oscilando en un cierto rango. A continuación, entonces, trataremos de explicar este comportamiento. Recordamos que el parámetro óptimo k es el que satisface $q^{k+1} + q^k \leq 1 < q^k + q^{k-1}$.

Proposición 3.1.2. *La redundancia del código de Golomb óptimo satisface*

$$R_G(q) = 1 - \{\lg(k)\} + 4 \times 2^{-2^{1-\{\lg(k)\}}} - C + o(1) \tag{3.1.7}$$

cuando $q \to 1^-$. Aquí $k \in \mathbb{Z}$ es el único entero tal que $q^{k+1}+q^k \leq 1 < q^k+q^{k-1}$, $C = \lg(e)+\lg\lg(e)$ y $\{x\}$ representa la parte fraccional de x.

Idea de la prueba. Notar que $2q^{k+1} \leq 1 < 2q^{k-1}$, y así tenemos

$$\left| -\frac{\log(2)}{k} - \log(q) \right| \leq \tfrac{1}{k} \times \log(1/q) \leq \frac{\log(2)}{k(k-1)}$$

es decir $\log(q) = -\frac{\log(2)}{k} + O(1/k^2)$. Utilizando esto, que $q^k = 1/2 + o(1)$ y $\lim_{x\to 0} \log(1+x) = 0$, y las fórmulas para la longitud promedio y la entropía, se sigue el resultado. □

El resultado anterior mencionado en el paper original de Gallager y van Voorhis [12]. En [13, Theorem 3], se puede ver una prueba para el caso $q = 2^{-1/\ell}$, junto con resultados asintóticos para otros códigos.

Observación 3.1.4. Como tenemos $\lg(k) = \lg\log 2 - \lg\log(1/q) + o(1)$, y $R(x) = 1 - x + 4 \times 2^{-2^{1-x}} - C$ es continua si identificamos 0 con 1, podemos sustituir $\lg\log 2 - \lg\log(1/q)$ por $\lg(k)$ en la fórmula de la redundancia R_G.

Observar que $\log(1/q) = 1/q - 1 + o(1) = 1 - q + o(1)$ cuando $q \to 1^-$, por lo tanto, si graficamos la redundancia de los códigos de Golomb en escala logarítmica cuando $q \to 1$, aparece la onda que se muestra en la figura 3.1.6, repetida periódicamente.

Finalmente, los máximos y mínimos de las oscilaciones ocurren son aproximadamente 0.0327 y 0.0251 respectivamente. Los puntos en los que estos ocurren se pueden caracterizar explícitamente en función de la función W de Lambert como se muestra en [13, Section 5].

Ejemplo 3.1.1 (*Golomb revisitado*)**.** Ahora que tenemos calculado el largo promedio para G_k, podemos justificar la elección del parámetro k, para un $q \in (0,1)$ dado. Para ver esto, notamos que

$$L_q(G_{k+1}) - L_q(G_k) = d_{k+1} - d_k + \frac{q^{2^{d_{k+1}+1}-k-1}}{1-q^{k+1}} - \frac{q^{2^{d_k+1}-k}}{1-q^k}\,,$$

donde $d_s = \lfloor \lg(s) \rfloor$. Aquí, si $k+1$ no es potencia de 2, se sigue que $d_{k+1} = d_k$ y luego

$$L_q(G_{k+1}) - L_q(G_k) = q^{2^{d_k+1}-k-1} \times \left(\frac{1}{1-q^{k+1}} - \frac{q}{1-q^k} \right)\,,$$

mientras que $d_{k+1} = r = d_k + 1$ si $k+1 = 2^r$, y luego

$$L_q(G_{k+1}) - L_q(G_k) = 1 + \frac{q^{k+1}}{1-q^{k+1}} - \frac{q}{1-q^k} = \frac{1}{1-q^{k+1}} - \frac{q}{1-q^k}\,.$$

Por lo tanto, en cualquier caso, el signo depende únicamente del signo de

$$\frac{1}{1-q^{k+1}} - \frac{q}{1-q^k} = (1-q^{k+1})^{-1}(1-q^k)^{-1} \times \left(1 - q^k - q + q^{k+2}\right)\,.$$

Aquí $1 - q^k - q + q^{k+2} = (1-q) \times \left(1 - q^k - q^{k+1}\right)$, y como $q \in (0,1)$ concluímos que

$$L_q(G_{k+1}) \geq L_q(G_k) \iff 1 \geq q^k + q^{k+1}\,.$$

Como $q^k + q^{k+1}$ decrece con k, concluímos que el menor $k \geq 1$ tal que $1 \geq q^k + q^{k+1}$ nos da el código de Golomb con menor longitud promedio para el parámetro $q \in (0,1)$ dado.

Observación 3.1.5 (*Códigos de Rice*)**.** Una alternativa menos compleja en cuanto a implementación es utilizar los códigos de Rice $R_k \triangleq G_{2^k}$. Si bien no son los códigos óptimos para nuestra distribución en general, se utilizan en algoritmos de compresión de imágenes como los presentados en [14, 15] debido a que su utilización permite simplificar tanto la codificación como la estimación de los parámetros.

Por un análisis similar, podemos ver que el parámetro óptimo para $q \in (0,1)$ es el menor $k \geq 0$ que satisface

$$q^{2^k} \leq \phi^{-1}\,,$$

donde $\phi = \frac{1+\sqrt{5}}{2}$.

En la figura 3.1.7 se muestra la redundancia de los códigos de Rice óptimos, en función de $q \in (0,1)$. Nótese que la redundancia en bits coincide exactamente con la de los códigos de Golomb G_k generales cuando q es cercano a $1/2$, pero el comportamiento cuando $q \to 1$ es significativamente distinto.

Proposición 3.1.3. *La redundancia del código de Rice óptimo satisface*

$$R_R(q) = 1 - x + \frac{1}{1-\phi^{-2^{1-x}}} - C + o(1) \tag{3.1.8}$$

cuando $q \to 1^-$. *Aquí* $x = \{\lg\log\phi - \lg\log(1/q)\}$, $C = \lg(e) - \lg\log\phi$ *y* $\{z\}$ *representa la parte fraccional de* z.

Demostración. Sea $k \geq 0$ el menor entero tal que $q^{2^k} \leq \phi^{-1}$. Luego $\phi^{-1} < q^{2^{k-1}}$ y así

$$k - 1 < \lg\log\phi - \lg\log(1/q) \leq k\,.$$

Concluímos entonces que

$$\lg\log\phi - \lg\log(1/q) = k - 1 + x\,, \tag{3.1.9}$$

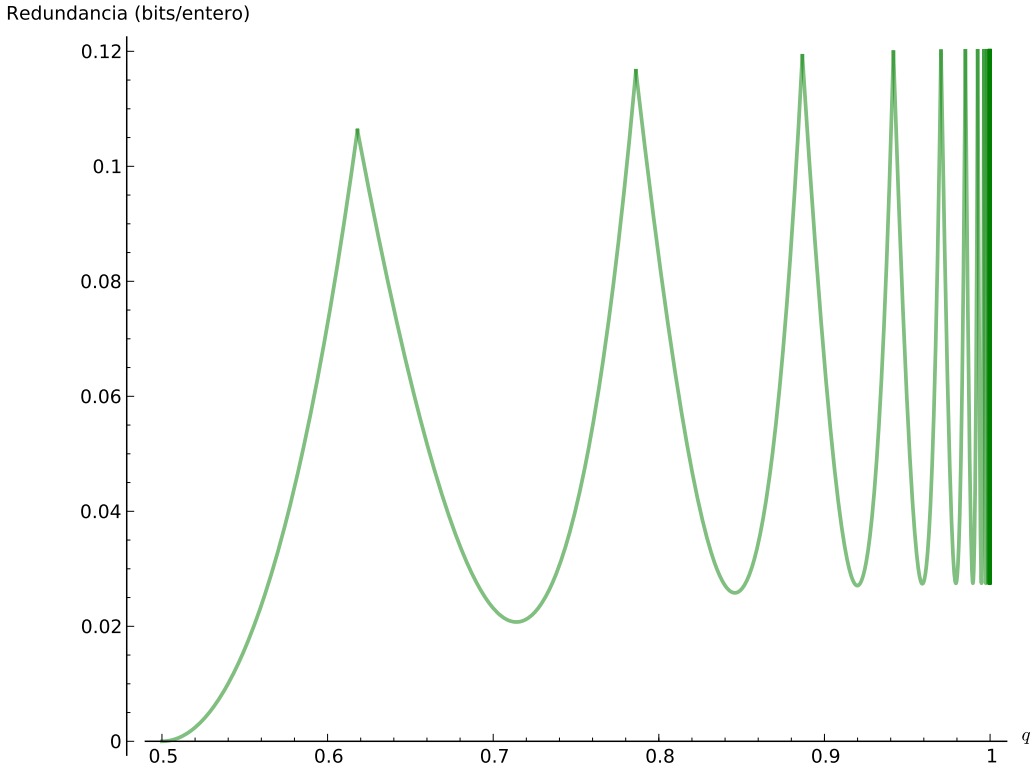


Figura 3.1.7: Redundancia de los códigos de Rice óptimos para $q \in [0.5, 1]$.

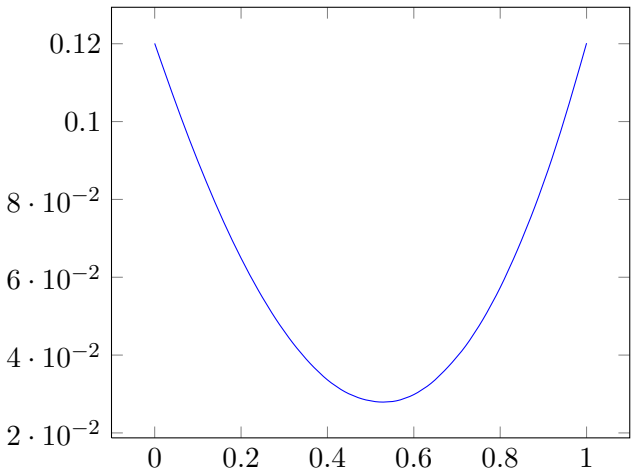


Figura 3.1.8: $R(x) = 1 - x + \frac{1}{1-\phi^{-2^{1-x}}} - C$

para un cierto $x \in (0,1]$ que depende de q. Notar que $x = \{\lg\log\phi - \lg\log(1/q)\}$, **excepto** cuando $\lg\log\phi - \lg\log(1/q)$ es entero, en cuyo caso la parte fraccional es 0 en lugar del 1 que deberíamos tener para que $\lg\log\phi - \lg\log(1/q) = k - 1 + x$ sea cierto. Como el resto de la derivación depende únicamente de (3.1.9), se puede luego considerar $x = \{\lg\log\phi - \lg\log(1/q)\}$ de todos modos, ya que el resultado final (3.1.8) se puede verificar que genera el mismo valor para $x = 0$ que para $x = 1$, utilizando que $\phi^2 = \phi + 1$.

Como vimos para la longitud media, tenemos

$$R(q) = L_q(G_{2^k}) - H(X) = k + \frac{1}{1-q^{2^k}} + \frac{q\lg(q) + (1-q)\lg(1-q)}{1-q}$$

Para completar la prueba notamos que

1. Tenemos
$$q^{2^k} = \phi^{-2^k \cdot \log(1/q)/\log(\phi)} = \phi^{-2^{k+\lg\log(1/q) - \lg\log(\phi)}} = \phi^{-2^{1-x}}$$

2. Tenemos
$$\frac{q\lg(q)}{1-q} = \frac{q\log(q)}{\log(2)(1-q)} = -\frac{1}{\log(2)} + o(1) = -\lg(e) + o(1)$$
dado que $\log(q) = \log(1+(q-1)) = (q-1) + O((q-1)^2)$ cuando $q \to 1$.

3. Finalmente
$$k + \lg(1-q) = 1 - x + \lg\log\phi - \lg\log(1/q) + \lg(1-q)$$
y aquí
$$\lg\log(1/q) = \lg(1-q) + \lg\left(\frac{\log(1/q)}{1-q}\right) = \lg(1-q) + o(1)\,. \quad \square$$

Recordamos que en el caso de la redundancia asintótica de Golomb G_k tomabamos la parte fraccional de $\lg(k) = \lg\log 2 - \lg\log(1/q) + o(1)$ mientras que en los de Rice tomamos la parte entera de $\lg\log\phi - \lg\log(1/q)$. Podemos entonces poner ambas funciones asintóticas (eliminando los términos que tienden a 0) en función de $x = \{\lg\log(1/q)\} \in [0,1]$ para tener

$$R_{\text{Golomb}}(x) = 1 - \{\lg\log 2 - x\} + 4 \times 2^{-2^{1-\{\lg\log 2 - x\}}} - C_{\text{Golomb}} \tag{3.1.10}$$

$$R_{\text{Rice}}(x) = 1 - \{\lg\log\phi - x\} + \left(1 - \phi^{-2^{1-\{\lg\log\phi - x\}}}\right)^{-1} - C_{\text{Rice}}\,, \tag{3.1.11}$$

donde $C_{\text{Golomb}} = \lg(e) + \lg\lg(e)$ y $C_{\text{Rice}} = \lg(e) - \lg\log(\phi)$.

Esta situación se encuentra ilustrada en la figura 3.1.9. Calculando la diferencia máxima entre las redundancias límites $R_{\text{Golomb}}(x)$ y $R_{\text{Rice}}(x)$, mediante una discusión por casos según la parte fraccional, obtenemos que esta es $5\phi - 8 \approx 0.09017$, que se alcanza para $x = 2 + \lg\log\phi$. Esta observación aparece en [18, Theorem 2 ($\Delta_{\text{lím}}$)], pero en el contexto de las distribuciones geométricas a *dos lados*.

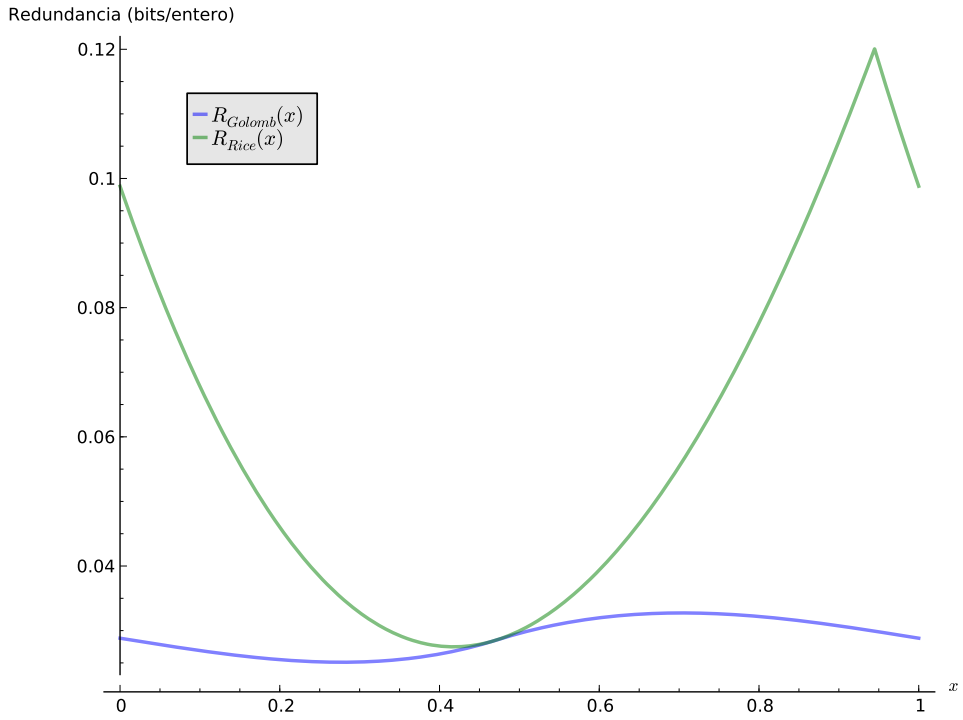


Figura 3.1.9: Las redundancias asintóticas de los códigos de Golomb (3.1.10) y de Rice (3.1.11) óptimos, en función de $x = \{\lg\log(1/q)\}$.

3.2. Códigos para pares de variables aleatorias con distribución geométrica

Si bien tenemos caracterizados los códigos óptimos para la distribución geométrica, cuando se codifican varias variables consecutivamente, se puede reducir la redundancia por símbolo mediante una codificación en bloque. Resulta entonces de interés estudiar lo que sucede cuando nuestra variable aleaoria es (X, Y) con X e Y geométricas iid.

Sin embargo, el panorama en este caso no resulta tan favorable como en el anterior, y, caracterizar una subfamilia de códigos óptimos resulta un problema considerablemente complicado.

3.2.1. Definiciones previas

En esta sección tratamos con una fuente $(\mathcal{A}, \mathbf{p})$ con

$$\mathcal{A} = \{(i,j) : i,j \geq 0\} \quad \text{y} \quad p(i,j) = (1-q)^2 q^{i+j} \ \text{ para cada } (i,j) \in \mathcal{A}\,, \tag{3.2.1}$$

a la que nos referimos en lo sucesivo como $TDGD(q)$, por *two-dimensional geometric distribution*.

Notar que $p(i,j)$ depende únicamente de la *firma* $s = i + j$ del símbolo $(i,j) \in \mathcal{A}$. Por lo tanto, si tenemos un árbol de código para $TDGD(q)$, permutar las hojas correspondientes a símbolos (i,j) con igual $s = i + j$ no cambia la longitud promedio. También decimos que s es la *firma* de la hoja o de la palabra de código que representa a (i,j) en el código.

Para una firma definimos su peso mediante $w(s) = q^s$, y más generalmente, para un árbol T definimos $w(T) \triangleq \sum_{x \text{ hoja}} q^{s(x)}$ donde $s(x)$ denota la firma de la hoja x. Para un nodo v definimos también $w(v) \triangleq w(T_v)$, donde T_v denota el subárbol de su descendencia.

Observar que la probabilidad de un símbolo (i, j) se obtiene multiplicando el peso de su firma por el factor $(1 - q)^2$. Definimos

$$\mathcal{L}_q(T) \triangleq \sum_{x \text{ hoja}} \ell_T(x) \cdot q^{s(x)} \,,$$

donde $\ell_T(x)$ es la profundidad de x en el árbol T, o, dicho de otro modo, el largo de la palabra de código representada por x.

Recordamos que identificamos los códigos con los árboles que los representan y, cuando hablamos del *nivel* ℓ, nos referimos a los nodos a profundidad ℓ en el código C dado. Dado un árbol T, lo que nos interesa realmente es el llamado *perfil* del árbol, esto es la sucesión $\{n_\ell^T\}_{\ell \geq 0}$ donde n_ℓ^T nos indica cuántas hojas hay en el nivel ℓ, ya que podemos permutar los nodos en un nivel dado (internos y hojas) sin afectar la longitud promedio final.

Observación 3.2.1. Observamos como siempre que, en un árbol óptimo, si un nivel ℓ tiene una hoja de firma s, no puede ser que un nivel $\ell' > \ell$ tenga hojas de firma $s' < s$, por la parte 1 del lema 2.5.1. Por lo tanto, el máximo nivel que ocupan las hojas de firma s es necesariamente menor o igual al menor nivel que ocupan las hojas de firma $s + 1$.

Definición 3.2.1 (*Gap*)**.** Un *gap* en un árbol $\mathcal{T}$ es un conjunto no vacío de niveles consecutivos sin hojas, y tal que el nivel inmediatamente arriba del conjunto, y el inmediatamente debajo de este, tienen al menos una hoja. El *tamaño del gap* es el número de niveles que contiene.

3.2.2. Singularidad de los códigos bidimensionales

Los códigos de Golomb, particionan el intervalo $[0, 1]$ en una familia numerable de intervalos, cada uno con su correspondiente código óptimo. Es razonable entonces preguntarse si la situación es similar en el caso bidimensional. Como veremos, la respuesta resulta negativa. En efecto, el objetivo de esta sección es probar que si $\mathcal{T}_q$ es óptimo para $TDGD(q)$ entonces no es óptimo para $TDGD(q_1)$ con $q_1 \neq q$.

Teorema 3.2.1. *Ningún árbol puede ser simultáneamente óptimo para dos distribuciones $TDGD(q)$ y $TDGD(q_1)$ con $q \neq q_1$.*

Para probar este teorema necesitaremos primero una serie de lemas auxiliares.

Lema 3.2.1. *Las hojas con firma s se encuentran en a lo sumo dos niveles de $\mathcal{T}_q$, y estos deben ser consecutivos.*

Demostración. Por absurdo, supongamos que tenemos hojas con firma s en dos niveles no consecutivos y sean entonces d_0 y d_1 los niveles mínimo y máximo en los que se encuentra s. Por nuestra suposición tenemos $d_1 - d_0 \geq 2$. Vamos a realizar ahora una transformación sobre $\mathcal{T}_q$. Tomemos dos hojas con firma s, una en el nivel d_0 y la otra en el nivel d_1, y en lugar de la hoja de nivel d_0, coloquemos allí un nodo interno que tenga como hijos las hojas que hemos tomado. Finalmente, tomemos una hoja s' de un nivel estrictamente mayor que d_1 y coloquémosla en lugar que ocupaba nuestra hoja de nivel d_1. Sea $\mathcal{T}_q'$ el árbol resultante, tenemos entonces:

$$\mathcal{L}_q\left(\mathcal{T}_q'\right) = \mathcal{L}_q\left(\mathcal{T}_q\right) + q^s\left(d_0 - d_1 + 2\right) - q^{s'}\delta \,,$$

donde $\delta > 0$ es el cambio de altura para la hoja s'. Nótese entonces que $\mathcal{L}_q\left(\mathcal{T}_q'\right) < \mathcal{L}_q\left(\mathcal{T}_q\right)$, lo cual es absurdo. □

Corolario 3.2.1. *De darse un gap en el árbol $\mathcal{T}_q$, si s es la mayor firma antes de iniciar el gap, la menor firma luego del gap debe ser necesariamente $s+1$.*

Demostración. Observar que, como $\mathcal{T}_q$ es óptimo, tenemos que s se encuentra necesariamente en el nivel anterior al gap, y $s+1$ en el nivel inmediatamente posterior al gap. □

Lema 3.2.2. *Sea $k = 1 + \left\lfloor \lg\left(q^{-1}\right) \right\rfloor$. Luego, para todo s suficientemente grande se tiene que el tamaño g de un gap entre hojas de la firma s y $s+1$ en $\mathcal{T}_q$, cumple $g \leq k-1$.*

Demostración. El caso $q = 1/2$ se sigue que en este caso todas las probabilidades son de la forma 2^{-i}, y por la observación 2.4.1 podemos derivar las longitudes **necesarias** para alcanzar la entropía a partir de las probabilidades. El perfil óptimo es, por tanto, único en este caso y se puede ver que corresponde a $G_1 \bowtie G_1$, que no tiene gaps.

Consieramos ahora $q > \frac{1}{2}$. En este caso se tiene que $k = 1$ y por lo tanto debemos probar que no pueden existir gaps a partir de un cierto nivel. Asumamos entonces que existe un gap entre el nivel d con firmas s y el nivel d' con firmas $s+1$ y $d' - d \geq 2$. Reorganizando niveles de ser necesario, podemos asumir que existe en $\mathcal{T}_q$ un nodo interno v a profundidad $d' - 1 \geq d+1$ tal que tiene al menos dos hojas con firma $s+1$ en su subárbol (no necesariamente en el mismo nivel). Luego

$$w(v) \geq 2q^{s+1} > q^s\,,$$

con lo cual intercambiar el nodo interno v y una hoja s en el nivel d hace decrecer el largo promedio del código, contradiciendo la optimalidad de $\mathcal{T}_q$.

Queda entonces el caso $q < \frac{1}{2}$. Sea $s \geq 2^k - 2$, y supongamos que existe un gap g entre las firmas s en el nivel d y las $s+1$ en el nivel $d+g+1$. Como $(s+1)+1 \geq 2^k$, sin pérdida de generalidad, reordenando niveles de ser necesario, podemos asumir que existe un nodo interno v en el nivel $d+g+1-k$ que contiene en su subárbol 2^k hojas con firma $s+1$ (entre los niveles $d+g+1$ y $d+g+2$). Luego, por definición de k, tenemos

$$w(v) \geq 2^k q^{s+1} > q^s\,.$$

Esto implica necesariamente que $d+g+1-k < d$ si $\mathcal{T}_q$ ha de ser óptimo, ya que en otro caso intercambiando v y una hoja s del nivel d haríamos decrecer el largo promedio. □

Por el lema 3.2.1 tenemos que las $s+1$ hojas de firma s se encuentran en 1 o 2 niveles consecutivos. Se sigue entonces que hay un nivel que tiene al menos la mitad de las hojas con firma s.

Definición 3.2.2. Denotamos mediante $\mathcal{D}(s)$ a la profundidad del nivel de $\mathcal{T}_q$ que contiene al menos la mitad de las hojas de firma s (con alguna regla de elección fija para el caso de empates).

Lema 3.2.3. *Sean $s \in \mathbb{Z}_{\geq 0}$ y $\ell \geq 2$, tales que $s \geq 2^{\ell-1} - 1$ y que existe $s' > s$ con $\mathcal{D}(s') = \mathcal{D}(s) + \ell$. En dicho caso, $\mathcal{T}_q$ cumple*

$$\frac{\ell - 2}{\lg q^{-1}} \leq s' - s\,. \tag{3.2.2}$$

Demostración. Como $s' + 1 \geq 2^{\ell-1}$, se sigue que en el nivel $\mathcal{D}(s')$ hay por lo menos $2^{\ell-2}$ hojas de firma s'. Sin pérdida de generalidad, existe entonces un árbol completo con altura $\ell - 2$ tal que tiene exactamente dichas hojas. Evidentemente su raíz debe estar en el nivel $\mathcal{D}(s) + 2$. Tomemos ahora una hoja s del nivel $\mathcal{D}(s)$, y pongamos en su lugar un nodo interno que tenga como hijos a la hoja original a su izquierda, y al árbol completo anterior a su derecha.

Sea $\mathcal{T}_q'$ el árbol resultante. Tenemos entonces

$$\mathcal{L}_q\left(\mathcal{T}_q'\right) - \mathcal{L}_q\left(\mathcal{T}_q\right) = q^s - 2^{\ell-2}q^{s'}\,. \tag{3.2.3}$$

dado que la hoja s aumentó su profundidad en 1, mientras que las hojas s' bajaron un nivel, ya que su raíz pasó del nivel $\mathcal{D}(s)+2$ al $\mathcal{D}(s)+1$.

Como $\mathcal{T}_q$ es óptimo, tenemos que

$$\mathcal{L}_q\left(\mathcal{T}_q'\right) - \mathcal{L}_q\left(\mathcal{T}_q\right) \geq 0\,,$$

y luego por (3.2.3) se deduce la desigualdad (3.2.2). □

Lema 3.2.4. *Sean s una firma y $\ell \geq 2$, tales que $s \geq 2^{\ell+2}-1$ y $\mathcal{D}(s') = \mathcal{D}(s)+\ell$ para una cierta firma $s' > s$. Luego en $\mathcal{T}_q$ tenemos*

$$s' - s \leq \frac{\ell+1}{\lg q^{-1}}\,. \tag{3.2.4}$$

Demostración. Notamos análogamente que en la prueba del lema anterior, que en el nivel $\mathcal{D}(s')$ hay al menos $2^{\ell+1}$ hojas con firma s'. Podemos suponer entonces que estas $2^{\ell+1}$ hojas, son hojas de un árbol completo, de altura $\ell+1$, y por lo tanto, con raíz en el nivel $\mathcal{D}(s')-(\ell+1)$, que es igual a $\mathcal{D}(s)-1$.

Intercambiamos ahora el nodo raíz del árbol completo con una hoja de firma s en el nivel $\mathcal{D}(s)$. Sea $\mathcal{T}_q'$ el árbol resultante, luego tenemos:

$$\mathcal{L}_q\left(\mathcal{T}_q'\right) - \mathcal{L}_q\left(\mathcal{T}_q\right) = -q^s + 2^{\ell+1}q^{s'}\,, \tag{3.2.5}$$

ya que la hoja con firma s decrece su profundidad en 1, mientras que las hojas del árbol completo aumentan su profundidad en 1.

Como $\mathcal{T}_q$ es óptimo, tenemos que

$$\mathcal{L}_q\left(\mathcal{T}_q'\right) - \mathcal{L}_q\left(\mathcal{T}_q\right) \geq 0\,,$$

y esto, junto con (3.2.5), implica la desigualdad (3.2.4). □

Puede no ser obvio a primera vista que existen ℓ, s y s' en las condiciones de los lemas anteriores, dado que hay una cierta circularidad. El siguiente lema prueba que existen $\ell \in \mathbb{Z}_{>0}$, arbitrariamente grandes, para los cuales es posible encontrar tales s y s'.

Lema 3.2.5. *Sea $\mathcal{T}_q$ árbol óptimo para $TDGD(q)$. Dado $\ell_0 \in \mathbb{Z}_{>0}$, existen firmas s y s', con $s' > s$, tales que la diferencia $\ell = \mathcal{D}(s') - \mathcal{D}(s)$ satisface $\ell \geq \ell_0$ y $s \geq 2^{\ell+2}-1$.*

Demostración. Sea $g_q = \left\lfloor \lg q^{-1} \right\rfloor$, y consideramos $s \geq 2^{\ell_0+g_q+4}$. Vamos a probar que existe un $\ell \in \{\ell_0, \ldots, \ell_0+g_q+2\}$ tal que $\mathcal{D}(s)+\ell$ es el nivel principal $\mathcal{D}(s')$ para una cierta firma s'. Observar que esto prueba el resultado, ya que $s > 2^{\ell_0+g_q+4}-1 \geq 2^{\ell+2}-1$.

Eventualmente tomando s aún más grande, podemos suponer que de los niveles

$$\mathcal{D}(s)+\ell_0+1, \ldots, \mathcal{D}(s)+\ell_0+g_q+1\,,$$

al menos uno debe contener hojas, ya que de otro modo tendríamos un gap de tamaño mayor que $g_q = \left\lfloor \lg q^{-1} \right\rfloor$, contradiciendo el lema 3.2.2. Sea $\tilde{\ell}$ dicho nivel, y sea s' la firma de una hoja arbitraria del nivel $\tilde{\ell}$.

Por el lema 3.2.1 sabemos que el nivel principal $\mathcal{D}(s')$ debe ser $\tilde{\ell}-1$, $\tilde{\ell}$ o $\tilde{\ell}+1$. En conclusión tenemos que $\mathcal{D}(s')$ es uno de los niveles

$$\mathcal{D}(s)+\ell_0, \ldots, \mathcal{D}(s)+\ell_0+g_q+2\,,$$

probando el resultado. □

Prueba del teorema 3.2.1. Sin pérdida de generalidad podemos suponer que $q < q_1$. Demostramos que si $\mathcal{T}_q$ es un árbol óptimo para $TDGD(q)$, no puede ser también óptimo para $TDGD(q_1)$, probando entonces el teorema.

Consideremos $s' > s$, con

$$\ell \triangleq \mathcal{D}\left(s'\right) - \mathcal{D}\left(s\right) \geq 2, \quad s \geq 2^{\ell+2} - 1\,. \tag{3.2.6}$$

Aplicamos la misma transformación que en el lema 3.2.3 sobre un árbol $\mathcal{T}_q$ para obtener

$$\Delta = \mathcal{L}_{q_1}\left(\mathcal{T}'_q\right) - \mathcal{L}_{q_1}\left(\mathcal{T}_q\right) = q_1^s - 2^{\ell-2} q_1^{s'} = q_1^s \times \left(1 - 2^{\ell-2} q_1^{s'-s}\right)\,.$$

Ahora, aplicando el lema 3.2.4 para acotar el término de más a la derecha obtenemos

$$\Delta \leq q_1^s \times \left(1 - 2^{\ell-2} q_1^{\frac{\ell+1}{\lg q^{-1}}}\right)\,, \tag{3.2.7}$$

y notamos que Δ es negativo si $\ell - 2 + \dfrac{\ell+1}{\lg q^{-1}} \lg q_1 > 0$, o, equivalente,

$$\ell - 2 - \frac{\ell+1}{\lg q^{-1}} \lg q_1^{-1} > 0\,. \tag{3.2.8}$$

Como $q_1 > q$ tenemos que $0 < \lg q_1^{-1} < \lg q^{-1}$, y la expresión (3.2.8) tiende a ∞ cuando $\ell \to \infty$. Existe entonces $M > 0$ tal que $\Delta < 0$ para todo $\ell \geq M$, que por el lema 3.2.5 sabemos que esta condición se puede satisfacer junto con (3.2.6). Con dicha selección de s y s', se tiene que (3.2.7) implica $\mathcal{L}_{q_1}\left(\mathcal{T}'_q\right) < \mathcal{L}_{q_1}\left(\mathcal{T}_q\right)$, probando que $\mathcal{T}_q$ no es óptimo para $TDGD(q_1)$. □

En consecuencia existe una cantidad no numerable de árboles óptimos distintos, sin embargo, en la práctica no podemos esperar más que trabajar con una familia numerable de estos. Por lo tanto, nos restringimos a estudiar familias numerables, convenientemente elegidas.

3.2.3. Códigos óptimos para $q = 2^{-1/k}$

Comenzamos por la familia de parámetros $q \in \{2^{-1/k} : k \in \mathbb{Z}_{>0}\}$ que cubren parte del intervalo $[1/2, 1]$. Para mostrar las ventajas de elegir esta familia, mostramos primero brevemente una prueba para la Geométrica unidimensional en el caso $q = 2^{-1/k}$.

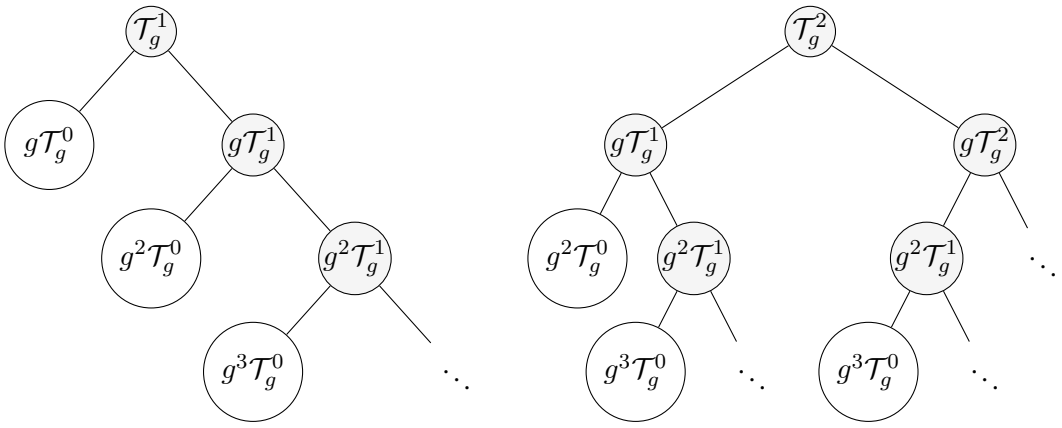
Definición 3.2.3. Denotamos mediante $\mathcal{T}_g^0$ el árbol trivial con una hoja de peso 1. Para $n \geq 0$ definimos inductivamente

$$\mathcal{T}_g^{n+1} \triangleq [g\mathcal{T}_g^n, g\mathcal{T}_g^{n+1}]\,,$$

donde $[\mathcal{T}_1, \mathcal{T}_2]$ denota el árbol que tiene a $\mathcal{T}_1$ y $\mathcal{T}_2$ como subárboles, izquierdo y derecho de su raíz, respectivamente, y $g\mathcal{T}$ denota el árbol que resulta de multiplicar los pesos de las hojas de $\mathcal{T}$ por g.

Esta definición se encuentra ilustrada en la figura 3.2.1. Observar que $\mathcal{T}_g^1$ corresponde al árbol de la codificación unaria.

En un abuso de notación también utilizaremos $\mathcal{T}_g^n$ para denotar la suma formal (función generatriz) de los pesos de sus hojas (que corresponde también a la suma en caso de que $|g| < 1$). Observar que esta suma formal indica, exactamente, cuál es el multiset de firmas correspondiente al árbol.



(a) Ilustración de $\mathcal{T}_g^1$. (b) Ilustración de $\mathcal{T}_g^2$.

Figura 3.2.1: Ejemplos de árboles definidos según la definición 3.2.3.

Observación 3.2.2. Observamos que la definición $\mathcal{T}_g^{n+1} = [g\mathcal{T}_g^n, g\mathcal{T}_g^{n+1}]$ implica, en cuanto a la suma de los pesos que

$$\mathcal{T}_g^{n+1} = g\mathcal{T}_g^n + g\mathcal{T}_g^{n+1}\,.$$

Se sigue entonces que

$$\mathcal{T}_g^{n+1} = \left(\frac{g}{1-g}\right)\mathcal{T}_g^n\,,$$

para cada $n \geq 0$. Como $\mathcal{T}_g^0 = 1$, se sigue que

$$\mathcal{T}_g^n = \left(\frac{g}{1-g}\right)^n\,, \tag{3.2.9}$$

para cada $n \in \mathbb{Z}_{\geq 0}$.

Observación 3.2.3. En particular, si en (3.2.9) tomamos $g = q^k$ y $q = 2^{-1/k}$, tenemos $\mathcal{T}_{q^k}^n = 1$ para cada $n \geq 0$. Esta observación resulta fundamental para las pruebas de optimalidad que siguen.

Ejemplo 3.2.1 (*Golomb para* $q = 2^{-1/k}$)**.** Para ejemplificar la forma en la que se prueba el resultado para los códigos bidimensionales, comenzamos por ver cómo funcionaría la misma prueba, pero para el caso unidimensional. La idea consiste en seguir el esquema de [12], pero con algunas modificaciones importantes.

La fuente reducida:

Consideramos el alfabeto $W_s = \mathcal{H}_s \cup \mathcal{F}_s$ formado por los árboles

$$\mathcal{H}_s = \{q^i\mathcal{T}_{q^k}^0 : i < s\}, \quad \mathcal{F}_s = \bigcup_{i=0}^{k-1}\{q^{i+s}\mathcal{T}_{q^k}^0, q^{i+s}\mathcal{T}_{q^k}^1\}$$

para $s \geq k$. A cada árbol le asignamos una probabilidad igual a su peso multiplicado por el factor de normalización $1-q$. Esto define en efecto una distribución de probabilidad dado que la suma de los

pesos de los árboles del alfabeto satisface

$$\begin{aligned}\sum_{i=0}^{s-1} q^i \mathcal{T}_{q^k}^0 + \sum_{i=0}^{k-1} \left(q^{i+s}\mathcal{T}_{q^k}^0 + q^{i+s}\mathcal{T}_{q^k}^1 \right) &= \sum_{i=0}^{s-1} q^i + \sum_{i=0}^{k-1} \left(q^{i+s} + q^{i+s} \cdot \frac{q^k}{1-q^k} \right) \\ &= \frac{1-q^s}{1-q} + q^s \times \left(\frac{1-q^k}{1-q} + \frac{q^k}{1-q} \right) \\ &= \frac{1-q^s}{1-q} + \frac{q^s}{1-q} \\ &= \frac{1}{1-q}\,,\end{aligned}$$

y, por lo tanto, el factor de normalización es $(1-q)$. Así, las probabilidades de los símbolos de $\mathcal{H}_s$ coinciden con la de los s símbolos más probables de la fuente original (3.1.1), y los podemos identificar. La cadena de igualdades anterior es también una igualdad formal, y que, por lo tanto, si pensamos en los árboles como el multiset de sus hojas, W_s corresponde a una partición del multiset de firmas $\mathbb{Z}_{\geq 0}$ ya que $\sum_{k\geq 0} q^k = (1-q)^{-1} \in \mathbb{Q}[\![q]\!]$. Observar también que a cada elemento de la partición le estamos asignando la suma de la probabilidades de sus elementos (hojas, si los pensamos como árboles).

Una vez que tenemos un árbol para la fuente reducida, esto induce también un árbol para la fuente original (3.1.1). En efecto, al código para la fuente reducida le concatenamos el código inducido por el árbol de los respectivos elementos de la partición, obteniendo un código libre de prefijos para $\mathbb{Z}_{\geq 0}$.

El unfolding (o merging) de la fuente reducida:

Notamos aquí que el peso del símbolo virtual $q^{i+s}\mathcal{T}_{q^k}^1$ es q^{i+s} por la observación 3.2.3. Observamos entonces que los dos elementos menos probables son exactamente $q^{i+s}\mathcal{T}_{q^k}^0$ y $q^{i+s}\mathcal{T}_{q^k}^1$ para $i = k-1$. Luego, el primer merge al aplicar el algoritmo de Huffman es de $q^{i+s}\mathcal{T}_{q^k}^0$ con $q^{i+s}\mathcal{T}_{q^k}^1$ para $i = k-1$. Escribimos el símbolo resultante como $[q^{i+s}\mathcal{T}_{q^k}^0,\, q^{i+s}\mathcal{T}_{q^k}^1]$, recordando entonces el merge recién efectuado. Este árbol resultante se puede escribir como

$$q^{i+s-k}[q^k\mathcal{T}_{q^k}^0, q^k\mathcal{T}_{q^k}^1]\,,$$

que coincide exactamente con $q^{i+s-k}\mathcal{T}_{q^k}^1$, de peso q^{i+s-k}.

Siguiendo, se realizan los merges

$$q^{i+s-k}\mathcal{T}_{q^k}^1 = [q^{i+s}\mathcal{T}_{q^k}^0,\, q^{i+s}\mathcal{T}_{q^k}^1]$$

para $i = k-2, \ldots, 0$, en orden decreciente de i. Afirmamos que la fuente que resulta luego de realizar estos merges es exactamente W_{s-k}. En efecto, esto se sigue de que

$$\mathcal{H}_s = \mathcal{H}_{s-k} \cup \bigcup_{i=0}^{k-1} \{q^{i+(s-k)}\mathcal{T}_{q^k}^0\}\,,$$

y que los k merges que acabamos de realizar producen $\bigcup_{i=0}^{k-1} \{q^{i+(s-k)}\mathcal{T}_{q^k}^1\}$.

Consideramos ahora, por conveniencia, que $s = m \cdot k$, luego en m pasos llevamos W_s a

$$W_0 = \mathcal{F}_0 = \bigcup_{i=0}^{k-1} \{q^i\mathcal{T}_{q^k}^0, q^i\mathcal{T}_{q^k}^1\}\,.$$

Observar aquí que, de hecho, podemos aplicar el mismo argumento de arriba una última vez para producir

$$W_{-k} = \{q^{i-k}\mathcal{T}^1_{q^k} : i \in \{0, \dots, k-1\}\}\,.$$

Factorizando $q^{-k}\mathcal{T}^1_{q^k}$ tenemos la fuente $\{q^0, \dots, q^{k-1}\}$, que es cuasi-uniforme ya que

$$q^{k-2} + q^{k-1} = \frac{q^{-2} + q^{-1}}{2} > \frac{q^0 + q^0}{2} = q^0.$$

Notar que nuestros árboles finales mantienen estructuralmente los merges realizados al aplicar Huffman. Por lo tanto, este 'factorizar' se corresponde a la concatenación de códigos y observar que $q^{-k}\mathcal{T}^1_{q^k}$ corresponde a G_1. Es decir, el código inducido para la fuente original $(\mathcal{A}, \mathbf{p})$, al expandir los árboles de la fuente reducida, es exactamente G_k, para cada $s \geq k$.

Conclusión:

Concluimos que el código inducido para la fuente original $(\mathcal{A}, \mathbf{p})$ es justamente G_k, y que el código de la fuente reducida de hecho coincide con G_k en la longitud de los primeros $|\mathcal{H}_s| = \binom{s}{2}$ símbolos más probables. En efecto, como el peso de los símbolos de $\mathcal{H}_s$ es estrictamente menor que el de cualquier árbol de $\mathcal{F}_s$, las raíces de los árboles de $\mathcal{F}_s$ se encuentran a mayor (o igual) profundidad que los símbolos de $\mathcal{H}_s$. Esto deja determinadas las longitudes de estos símbolos.
Finalmente, por un argumento límite, idéntico al que vimos en el teorema 3.1.1, concluimos que G_k es óptimo para la fuente original, tomando $s \to \infty$. $\lozenge$

¿Cómo se podría haber predicho que había que usar esa fuente reducida? Notar que el hecho de que, cuando expandimos los símbolos virtuales como árboles, tengamos el código límite que buscamos, sugiere que lo que se hace en realidad es lo contrario: tomamos el código que queremos probar es el límite, elegimos subárboles para un buen `unfolding` y los declaramos como símbolos virtuales.

Esta misma idea es la que se aplica en la demostración del siguiente teorema, que establece la optimalidad de una familia de códigos para $TDGD(q)$. Para esto denotamos mediante T_k a un código óptimo arbitrario para la fuente finita con los siguientes símbolos y pesos

$$\hat{\mathcal{A}}_k = \{(i,j) : 0 \leq i, j < k\}, \quad w(i,j) = q^{i+j}\,, \tag{3.2.10}$$

donde $q = 2^{-1/k}$. Resulta importante observar que los pesos de los elementos de $\hat{\mathcal{A}}_k$ corresponden exactamente al multiset

$$\{\underbrace{q^0}_{1\text{ veces}}, \underbrace{q^1}_{2\text{ veces}}, \underbrace{q^{k-2}}_{3\text{ veces}}, \dots, \underbrace{q^{k-1}}_{k\text{ veces}}, \underbrace{q^k}_{k-1\text{ veces}}, \dots, \underbrace{q^{2k-3}}_{2\text{ veces}}, \underbrace{q^{2k-2}}_{1\text{ veces}}\}\,. \tag{3.2.11}$$

Finalmente, a un código T_k óptimo para la fuente finita definida por (3.2.10) lo denominamos *top-code*.

Teorema 3.2.2. *Un código óptimo de prefijo C_k para $TDGD(q)$ con $q = 2^{-1/k}$, $k \geq 1$ es*

$$C_k(i,j) = T_k(i \bmod k, j \bmod k) \bowtie G_1\left(\left\lfloor \tfrac{i}{k} \right\rfloor\right) \bowtie G_1\left(\left\lfloor \tfrac{j}{k} \right\rfloor\right) \tag{3.2.12}$$

donde G_1 indica el código unario, $\bowtie$ la concatenación, y T_k denota un top-code arbitrario.

Demostración. Definimos una fuente reducida con el alfabeto $W_s = \mathcal{H}_s \cup \mathcal{F}_s$ donde

$$\mathcal{H}_s = \{\underbrace{q^i\mathcal{T}^0_{q^k}}_{i+1\text{ veces}} : 0 \leq i < s\}, \quad \mathcal{F}_s = \bigcup_{i=0}^{k-1} \{\underbrace{q^{s+i}\mathcal{T}^2_{q^k}}_{k\text{ veces}}, \underbrace{q^{s+i}\mathcal{T}^1_{q^k}}_{k+s+i+1\text{ veces}}, \underbrace{q^{s+i}\mathcal{T}^0_{q^k}}_{s+i+1\text{ veces}}\}\,.$$

Las respectivas probabilidades quedan definidas por el peso del símbolo correspondiente, a menos de un factor de normalización. Es directo verificar que W_s, al expandir los árboles, particiona los símbolos del multiset $\mathcal{A}$ de la definición de la fuente bidimensional, de modo que, a cada hoja le corresponde un par (i, j) tal que q^{i+j} coincide con el peso de la hoja. Así pues, el factor de normalización es $(1-q)^2$ y $\mathcal{H}_s$ corresponde directamente, como en el caso unidimensional del ejemplo 3.2.1, a los $|\mathcal{H}_s|$ símbolos más probables de $(\mathcal{A}, \mathbf{p})$.

Procedemos a aplicar el algoritmo de Huffman, de manera similar al ejemplo 3.2.1. Los símbolos de menor peso de W_s son exactamente los del multiset

$$\{\underbrace{q^{s+i}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{s+i}\mathcal{T}^1_{q^k}}_{k+s+i+1 \text{ veces}}, \underbrace{q^{s+i}\mathcal{T}^0_{q^k}}_{s+i+1 \text{ veces}}\} \tag{3.2.13}$$

para $i = k-1$. Podemos elegir primero los $s+i+1$ símbolos $q^{s+i}\mathcal{T}^0_{q^k}$ de (3.2.13) y hacer merge , par a par, con $s+i+1$ símbolos $q^{s+i}\mathcal{T}^1_{q^k}$ de (3.2.13), formando $s+i+1$ símbolos de peso $q^{s+i-k+1}$, que definimos como el árbol $[q^{s+i}\mathcal{T}^0_{q^k}, q^{s+i}\mathcal{T}^1_{q^k}]$, a fin de recordar los merges realizados. Podemos escribir este símbolo como $q^{s+i-k+1}\mathcal{T}^1_{q^k}$. A los k símbolos $q^{s+i}\mathcal{T}^1_{q^k}$ de (3.2.13), les hacemos merge , par a par, con los restantes k símbolos $q^{s+i}\mathcal{T}^2_{q^k}$ de (3.2.13) que no fueron apareados, para generar, análogamente, k símbolos de peso q^{s+i-k} que definimos como $q^{s+i-k}\mathcal{T}^2_{q^k}$.

Una vez realizadas estas transformaciones, el multiset de (3.2.13) para $i = k-1$ se transforma en

$$\{\underbrace{q^{s+i-k}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{s+i-k}\mathcal{T}^1_{q^k}}_{s+i+1 \text{ veces}}\}\,. \tag{3.2.14}$$

Este proceso sigue con $i = k-2$, $i = k-3$, ..., $i = 0$, en dicho orden, transformando W_s en W_{s-k}. En efecto, esto último se sigue de observar que

$$\mathcal{H}_s = \mathcal{H}_{s-k} \cup \bigcup_{i=0}^{k-1} \{\underbrace{q^{s+i-k}\mathcal{T}^0_{q^k}}_{s+i-k+1 \text{ veces}}\}\,,$$

y esto, junto con los multisets de (3.2.14) para $i = 0, \ldots, k-1$ forma W_{s-k}.

Por conveniencia tomamos $s = m \cdot k$, transformamos $S_m = W_s$ en $S_{m-1} = W_{s-k}$, y de hecho podemos aplicarlo $t+1$ veces para obtener

$$S_{-1} = \bigcup_{i=0}^{k-1} \{\underbrace{q^{i-k}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{i-k}\mathcal{T}^1_{q^k}}_{i+1 \text{ veces}}\}\,. \tag{3.2.15}$$

Dado $t \in \{0, \ldots, k\}$ definimos

$$\mathcal{S}_t \triangleq \bigcup_{i=1}^{t} \{\underbrace{q^{-k-i}\mathcal{T}^2_{q^k}}_{k+1-i \text{ veces}}\} \cup \mathcal{Y}_t \cup \bigcup_{i=0}^{k-1-t} \{\underbrace{q^{i-k}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{i-k}\mathcal{T}^1_{q^k}}_{i+1 \text{ veces}}\}\,, \tag{3.2.16}$$

donde $\mathcal{Y}_t \triangleq \mathcal{P}_t \cup \Big\{ \underbrace{q^{-t}\mathcal{T}^2_{q^k}}_{\varepsilon_{k,k-t} \text{ veces}} \Big\}$ y

$$\begin{aligned}\mathcal{P}_t \triangleq \Bigg\{ & \underbrace{[q^{-t}\mathcal{T}^2_{q^k}, q^{-t}\mathcal{T}^2_{q^k}]}_{\left(t-1-\varepsilon_{k,k-t+1}-\varepsilon_{k,k-t}\right)/2 \text{ veces}}, \underbrace{[q^{-t}\mathcal{T}^2_{q^k}, q^{1-t}\mathcal{T}^2_{q^k}]}_{\varepsilon_{k,k-t+1} \text{ veces}}, \dots \\ & \dots, \underbrace{[q^{-k+i}\mathcal{T}^2_{q^k}, q^{-k+i}\mathcal{T}^2_{q^k}]}_{\left(k-1-i-\varepsilon_{k,i+1}-\varepsilon_{k,i}\right)/2 \text{ veces}}, \underbrace{[q^{-k+i}\mathcal{T}^2_{q^k}, q^{-k+i+1}\mathcal{T}^2_{q^k}]}_{\varepsilon_{k,i+1} \text{ veces}}, \dots \\ & \dots, \underbrace{[q^{-1}\mathcal{T}^2_{q^k}, q^{-1}\mathcal{T}^2_{q^k}]}_{\left(0-\varepsilon_{k,k}-\varepsilon_{k,k-1}\right)/2=0 \text{ veces}}, \underbrace{[q^{-1}\mathcal{T}^2_{q^k}, q^{0}\mathcal{T}^2_{q^k}]}_{\varepsilon_{k,k}=0 \text{ veces}} \Bigg\},\end{aligned}$$

con $\varepsilon_{k,i}$ igual a 1 si $k - i \not\equiv 0, 1 (\text{mód}.4)$ y 0 en otro caso.

Afirmamos que $\mathcal{S}_0 = S_{-1}$ y que para todo t, con $0 \le t \le k$, existe una secuencia de pasos válidos del algoritmo de Huffman que transforman $\mathcal{S}_t$ en $\mathcal{S}_{t+1}$ si $0 \le t < k$.

En efecto, para $t = 0$ tenemos que $\mathcal{P}_t = \emptyset$, y por lo tanto $\mathcal{Y}_t = \emptyset$, pues $\varepsilon_{k,k} = 0$. Luego de (3.2.16) se sigue que $\mathcal{S}_t = S_{-1}$, como afirmamos. Ahora, dado $\mathcal{S}_t$ con $t < k$, mostramos cómo pasar a $\mathcal{S}_{t+1}$ mediante merges de Huffman válidos.

Los símbolos de menor peso de $\bigcup_{i=0}^{k-1-t} \{\underbrace{q^{i-k}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{i-k}\mathcal{T}^1_{q^k}}_{i+1 \text{ veces}}\}$ son los que tienen i máximo, estos son, exactamente $\{\underbrace{q^{-1-t}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{-1-t}\mathcal{T}^1_{q^k}}_{k-t \text{ veces}}\}$. Como los símbolos de $\mathcal{P}_t$ tienen peso estrictamente mayor que $q^0 + q^0 = q^{-k}$, y los símbolos de la unión de más a la izquierda en (3.2.16) tienen peso mayor que q^{-k}, concluímos que, si $\varepsilon_{k,k-t} = 1$, el símbolo de menor peso en $\mathcal{S}_t$ es el $q^{-t}\mathcal{T}^2_{q^k}$ proveniente de $\mathcal{Y}_t$, y los siguientes símbolos de menor peso son $\{\underbrace{q^{-1-t}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{-1-t}\mathcal{T}^1_{q^k}}_{k-t \text{ veces}}\}$. En este caso, el primer merge será de $q^{-t}\mathcal{T}^2_{q^k}$ con uno de los símbolos $q^{-1-t}\mathcal{T}^2_{q^k}$, generando el elemento $\underbrace{[q^{-1-t}\mathcal{T}^2_{q^k}, q^{-t}\mathcal{T}^2_{q^k}]}_{\varepsilon_{k,k-t} \text{ veces}}$ de $\mathcal{P}_{t+1}$. Quedan entonces $k - \varepsilon_{k,k-t}$ símbolos $q^{-1-t}\mathcal{T}^2_{q^k}$ disponibles. Esto es válido incluso cuando $\varepsilon_{k,k-t} = 0$.

Observamos ahora que $k - t \le k - \varepsilon_{k,k-t}$. Esto es obvio si $t \ge 1$, mientras que para $t = 0$ tenemos $\varepsilon_{k,k-0} = 0$. Por lo tanto, seguidamente hacemos merge de $q^{-1-t}\mathcal{T}^2_{q^k}$ con $q^{-1-t}\mathcal{T}^1_{q^k}$, exactamente $k - t$ veces, generando entonces $k - t$ símbolos $q^{-k-(1+t)}\mathcal{T}^2_{q^k} = [q^{-1-t}\mathcal{T}^2_{q^k}, q^{-1-t}\mathcal{T}^1_{q^k}]$. Este corresponde al término para $i = t + 1$, nuevo, en la unión de más la izquierda en (3.2.16).

Quedan disponibles $(t - \varepsilon_{k,k-t}) - (k - t) = t - \varepsilon_{k,k-t}$ símbolos $q^{-1-t}\mathcal{T}^2_{q^k}$, que son, en este punto, exactamente los de menor peso. Los siguientes merge siguiendo el algoritmo de Huffman generan entonces

$$\left\lfloor \frac{t - \varepsilon_{k,k-t}}{2} \right\rfloor = \frac{t - \varepsilon_{k,k-t} - \varepsilon_{k,k-t-1}}{2}$$

pares $[q^{-1-t}\mathcal{T}^2_{q^k}, q^{-1-t}\mathcal{T}^2_{q^k}]$, sobrando exactamente $\varepsilon_{k,k-t-1}$ símbolos $q^{-1-t}\mathcal{T}^2_{q^k}$.

Esto completa $\mathcal{Y}_{t+1}$ y la derivación de $\mathcal{S}_{t+1}$ a partir de $\mathcal{S}_t$, utlizando merges de Huffman válidos. Tenemos probada la afirmación (3.2.16).

Tomando $t = k$ en (3.2.16) tenemos:

$$\mathcal{S}_k = \{ \underbrace{q^{-2k}\mathcal{T}^2_{q^k}}_{1 \text{ veces}}, \underbrace{q^{-2k+1}\mathcal{T}^2_{q^k}}_{2 \text{ veces}}, \ldots, \underbrace{q^{-k-1}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \\ \underbrace{[q^{-2}\mathcal{T}^2_{q^k}, q^{-3}\mathcal{T}^2_{q^k}]}_{1 \text{ veces}}, \underbrace{[q^{-3}\mathcal{T}^2_{q^k}, q^{-4}\mathcal{T}^2_{q^k}]}_{1 \text{ veces}}, \underbrace{[q^{-4}\mathcal{T}^2_{q^k}, q^{-4}\mathcal{T}^2_{q^k}]}_{1 \text{ veces}}, \underbrace{[q^{-5}\mathcal{T}^2_{q^k}, q^{-5}\mathcal{T}^2_{q^k}]}_{2 \text{ veces}}, \ldots \},$$

donde aparece un extra $q^{-k}\mathcal{T}^2_{q^k}$ cuando $\varepsilon_{k,0} = 1$.

Podemos revertir todos los merge que están entre paréntesis rectos, y obtenemos entonces

$$S^\star = \{ \underbrace{q^{-2k}\mathcal{T}^2_{q^k}}_{1 \text{ veces}}, \underbrace{q^{-2k+1}\mathcal{T}^2_{q^k}}_{2 \text{ veces}}, \underbrace{q^{-2k+2}\mathcal{T}^2_{q^k}}_{3 \text{ veces}}, \ldots \\ \ldots, \underbrace{q^{-k-1}\mathcal{T}^2_{q^k}}_{k \text{ veces}}, \underbrace{q^{-k}\mathcal{T}^k_{q}}_{k-1 \text{ veces}}, \ldots, \underbrace{q^{-3}\mathcal{T}^2_{q^k}}_{2 \text{ veces}}, \underbrace{q^{-2}\mathcal{T}^2_{q^k}}_{1 \text{ veces}} \},$$

nuevamente, factorizando aquí el árbol $q^{-2k}\mathcal{T}^2_{q^k}$, obtenemos $\hat{\mathcal{A}}_k$, mientras que $q^{-2k}\mathcal{T}^2_{q^k}$ corresponde a $G_1 \bowtie G_1$. □

Como no resulta necesario distinguir entre elementos (i, j) de igual firma $i + j$ en $\hat{\mathcal{A}}_k$, para alivianar notación denotamos mediante q^i a un símbolo de $\hat{\mathcal{A}}_k$ de firma i (equivalentemente, de peso q^i), que lo pensamos también como árbol que consiste solamente en dicha raíz. Observar entonces que esto es equivalente a $q^i\mathcal{T}^0_\cdot$, pero, como no tratamos con los árboles $\mathcal{T}_\cdot$ en lo que sigue, omitimos $\mathcal{T}^0_\cdot$ en la notación.

Definición 3.2.4. Dado un árbol T denotamos mediante f_T la diferencia máxima entre 2 niveles que contienen hojas, o, dicho de otro modo, si C es un código asociado

$$f_T = \max\{\ell(c) - \ell(c') : c, c' \in C\}$$

Volviendo a la prueba del teorema 3.2.2, podemos en $\hat{\mathcal{A}}_k$ realizar los mismos merge que relacionan $S^\star$ y $\mathcal{S}_k$, obteniendo

$$\hat{A}' = \{ \underbrace{q^0}_{1 \text{ veces}}, \underbrace{q^1}_{2 \text{ veces}}, \ldots, \underbrace{q^{k-1}}_{k \text{ veces}}, \underbrace{[q^{2k-2}, q^{2k-3}]}_{1 \text{ veces}}, \underbrace{[q^{2k-3}, q^{2k-4}]}_{1 \text{ veces}}, \underbrace{[q^{2k-4}, q^{2k-4}]}_{1 \text{ veces}}, \underbrace{[q^{2k-5}, q^{2k-5}]}_{2 \text{ veces}}, \ldots \}, \tag{3.2.17}$$

eventualmente pudiendo quedar un solo q^k, a lo sumo, sin par[b]. Esta fuente intermedia resulta de importancia como lo demuestra la siguiente proposición.

Proposición 3.2.1. *Existe un árbol óptimo para $\hat{\mathcal{A}}_k$ que tiene $f_T \leq 2$.*

Demostración. Aquí notamos que el símbolo de menor peso en (3.2.17) es o q^k o q^{k-1}, según quede un q^k sin par o no, en cualquier caso, $q^0 < q^k + q^{k-1} = \frac{1}{2} \times \left(1 + q^{-1}\right)$, con lo cual la fuente modificada es cuasi-uniforme y se sigue el resultado por la proposición 3.1.1. □

Observación 3.2.4. Los merges transforman $\hat{\mathcal{A}}_k$ en $\hat{A}'$ son necesarios por el lema 2.5.1 si reordenamos nodos en los distintos niveles de T_k. En efecto, q^{2k-2} y q^{2k-3} son exactamente las hojas de menor peso, y por lo tanto en cualquier caso deben estar en el nivel más profundo y, reordenando podemos asumir que son hermanas. Una vez hecho esto, se repite el argumento tomando el merge $[q^{2k-2}, q^{2k-3}]$ como un nodo con peso $q^{2k-2} + q^{2k-3} > 2q^{2k-2} = q^{k-2} > q^k$.

[b] Esto ocurre en caso de que $1 + 2 + \ldots + (k-1) = \dfrac{k \cdot (k-1)}{2}$ sea impar.

El resultado anterior es de importancia práctica, en pocas palabras nos dice que el código T_k se puede describir con unos pocos parámetros, ya que tiene a lo sumo 3 niveles con hojas. De hecho, ordenando por pesos en $\hat{A}'$ y aplicando la proposición 3.1.1, podemos obtener cuántos nodos hay a cada profundidad y con esto construir un código libre de prefijo como en la prueba de la desigualdad de Kraft.

Ordenar $\hat{A}'$ de acuerdo a pesos no resulta complicado ya que

$$q^{k-i} = 2q^{2k-i} < w\left([q^{2k-i}, q^{2k-i-1}]\right) < 2q^{2k-i-1} = q^{k-i-1}$$

y

$$w\left([q^{2k-i}, q^{2k-i}]\right) = 2q^{2k-i} = q^{k-i}\,.$$

La siguiente proposición caracteriza los niveles en los cuales los códigos T_k pueden tener hojas. Para esto definimos primero

$$M = \lfloor \lg(Q) \rfloor\,, \quad Q = \left\lceil \frac{k\cdot(k-1)}{4} \right\rceil + \frac{k\cdot(k+1)}{2}\,, \tag{3.2.18}$$

que se utilizan en lo sucesivo.

Proposición 3.2.2. *Los árboles óptimos para $\hat{\mathcal{A}}_k$ pueden tener hojas únicamente en los niveles M, $M+1$ y $M+2$.También, existe al menos un árbol óptimo que tiene al menos una hoja en el nivel M.*

Demostración. Realizamos nuevamente merges para obtener $\hat{A}'$. Como se mencionó en la observación 3.2.4, estos merges son necesarios, a menos de reordenamiento de nodos en los niveles del árbol óptimo para $\hat{\mathcal{A}}_k$, y por lo tanto no modifican el perfil del árbol.

Afirmamos que $|\hat{A}'| = Q$. En efecto $\left\{ \underbrace{q^0}_{1 \text{ veces}}, \underbrace{q^1}_{2 \text{ veces}}, \ldots, \underbrace{q^{k-1}}_{k \text{ veces}} \right\}$ contiene $1+2+\ldots+k = \frac{k\cdot(k+1)}{2}$ elementos, mientras que los restantes surgen de aparear los elementos de $\left\{ \underbrace{q^k}_{k-1 \text{ veces}}, \underbrace{q^{k+1}}_{k-2 \text{ veces}}, \ldots, \underbrace{q^{2k-2}}_{1 \text{ veces}} \right\}$, que son $\frac{k\cdot(k-1)}{2}$, más un posible extra q^k que quede sin par. Estos cuentan entonces por $\left\lceil \frac{k(k-1)/2}{2} \right\rceil$ elementos.

Ahora, por la proposición 3.1.1, se sigue que el código óptimo para $\hat{A}'$ tiene todas las hojas a profundidad M o $M+1$, y por lo tanto $\hat{\mathcal{A}}_k$ tiene todos sus símbolos a profundidades M, $M+1$ o $M+2$.

El número de elementos n_M a profundidad M corresponde con el número de símbolos q^i con $i < k$ que están en el nivel M de $\hat{A}'$. Análogamente n_{M+1} corresponderá al número de árboles con dos hojas[c] de $\hat{A}'$ que queden a profundidad M más los elementos q^i con $i < k$ que queden a profundidad $M+1$, y n_{M+2} corresponderá exclusivamente al número de árboles con dos hojas de n_{M+2} que queden a profundidad $M+1$ en el árbol para $\hat{A}'$.

Como q^0 siempre es de los elementos de mayor peso en $\hat{A}'$, podemos ordenar los símbolos de ese peso de manera que siempre quede a profundidad M. □

En consecuencia un perfil óptimo para T_k queda determinado por el trío (n_M, n_{M+1}, n_{M+2}), en vista de que $n_i = 0$ si $i \neq M, M+1, M+2$. Denominamos entonces *perfil* de T_k a la terna (n_M, n_{M+1}, n_{M+2}).

[c] Recordar que q^i denota el árbol $q^i\mathcal{T}_{q^k}^0$ que tiene una sola hoja. Los árboles con dos hojas, en este caso, son entonces los de la forma $[q^i, q^j]$.

k	M	n_M	n_{M+1}	n_{M+2}
1	0	1	0	0
2	2	4	0	0
3	3	7	2	0
4	3	1	13	2
5	4	7	18	0
6	4	1	25	10
7	5	15	34	0
8	5	5	49	10
9	5	0	47	34
10	6	29	69	2

Tabla 3.1: Lista de algunos perfiles óptimos para T_k.

Observación 3.2.5. No es cierto que exista un único perfil de árbol óptimo. Por ejemplo, para $k = 9$ hay un árbol óptimo con $(n_M, n_{M+1}, n_{M+2}) = (0, 47, 34)$ y otro con $(n_M, n_{M+1}, n_{M+2}) = (1, 44, 36)$.

Notar que $36 = (9 \cdot 8)/2$ y que, por lo tanto el 36 en realidad corresponde exactamente a todos los símbolos 'apareados' en $\hat{A}'$. Uno de estos símbolos es $[q^k, q^k]$ que tiene peso q^0, y luego se puede intercambiar este de lugar con el símbolo de mayor peso q^0 (que deberá estar lo más arriba posible necesariamente en el caso $(n_M, n_{M+1}, n_{M+2}) = (1, 44, 36)$).

Corolario 3.2.2. *Se tiene* $n_{M+2} \leq \frac{k \cdot (k-1)}{2}$

En el apéndice A.1 presentamos un script para calcular perfiles, implementado de modo que solo utilice suma y resta de enteros, así como *bit-wise and*. Su tiempo de ejecución es claramente $O(k)$. En la tabla 3.1 se detallan algunos perfiles óptimos, obtenidos utilizando la script mencionada. Notar que estos perfiles no son los únicos posibles en todos los casos, como se discutió arriba. Los demás perfiles se obtienen según cómo se decida reordenar los símbolos $[q^{k+i}, q^{k+i}]$ y q^i, que tienen igual peso. De tomar $q^i \prec [q^{k+i}, q^{k+i}]$ en nuestro orden, obtenemos n_{M+2} mínimo, ya que colocamos a menor profundidad los elementos mayores según el orden, minimizando así también la varianza en la longitud.

Las ganancias en cuanto a redundancia con relación a utilizar los códigos unidimensionales de Golomb G_k y los de Rice G_{2^k} se pueden visualizar en la figura 3.2.2.

Determinar la longitud media no es tan sencillo en este caso como en el de los códigos de Golomb, dado que no es trivial determinar el perfil correspondiente a un $q = 2^{-1/k}$ dado. En [17, Theorem 4] se caracterizan más generalmente perfiles óptimos a partir de las raíces de un polinomio cuadrático. Si bien este análisis no es necesario desde el punto de vista estrictamente práctico, permite obtener fórmulas bastante directas para obtener la longitud promedio de los códigos bidimensionales, para luego tener resultados asintóticos para la redundancia de los códigos bidimensionales [17, Corollary 6].

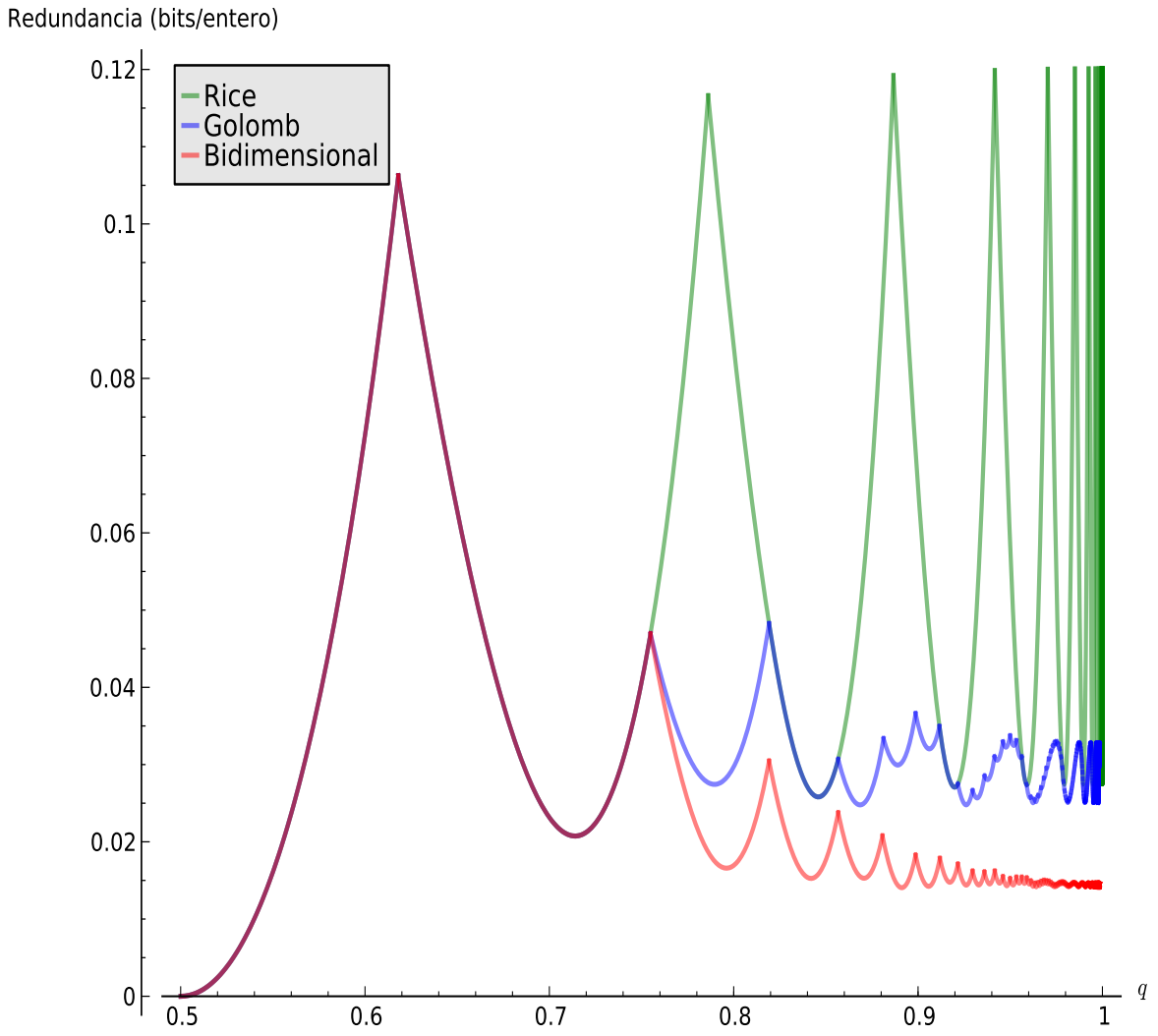


Figura 3.2.2: Redundancia (en bits/entero) de los códigos de Rice, Golomb y bidimensionales óptimos para $q \in [1/2, 1]$.

3.2.4. Perfiles óptimos para T_k

En esta sección presentamos un análisis más detallado de los perfiles óptimos para T_k, obteniendo un algoritmo eficiente para calcularlos.

Observación 3.2.6. Para caracterizar un perfil óptimo (n_M, n_{M+1}, n_{M+2}) para T_k, basta calcular n_M.

Demostración. Comenzamos observando que

$$n_M + n_{M+1} + n_{M+2} = |\hat{\mathcal{A}}_k| = k^2\,, \tag{3.2.19}$$

dado que $\hat{\mathcal{A}}_k$ tiene k^2 elementos.

Por el teorema 2.7.1 debemos tener $K = 1$, donde K denota el número de Kraft-McMillan (ver definición 2.2.1). Por definición de K tenemos

$$n_M 2^{-M} + n_{M+1} 2^{-M-1} + n_{M+2} 2^{-M-2} = 1\,. \tag{3.2.20}$$

Resolviendo el sistema determinado por las ecuaciones (3.2.19) y (3.2.20) en función de n_M tenemos que

$$n_{M+1} = 2^{M+2} - k^2 - 3\,n_M\,, \quad n_{M+2} = 2\,k^2 + 2\,n_M - 2^{M+2}\,, \tag{3.2.21}$$

mostrando que el valor de n_M determina el perfil. □

Para calcular n_M, recordamos que n_M corresponde al número de símbolos q^i, con $0 \le i < k$, en el nivel M del árbol correspondiente para $\hat{A}'$. Esto se observó en la prueba de proposición 3.2.2. La fuente $\hat{A}'$ es cuasi-uniforme, tiene exactamente $2^{M+1} - Q$ hojas en el nivel M según la proposición 3.1.1, y podemos seleccionar estas como las $2^{M+1} - Q$ símbolos más probables de $\hat{A}'$, según algún orden total que respete $a \prec b$ cuando $w(a) < w(b)$.

Si ordenamos los símbolos de $\hat{A}'$ mediante la regla $q^i \prec [q^{k+i}, q^{k+i}]$, y $a \prec b$ si $w(a) < w(b)$, obtenemos, por lo que vimos en la prueba del teorema 3.2.2 (factorizando $q^{-2k}\mathcal{T}^2_{q^k}$ de $\mathcal{S}_k$) que

$$\begin{aligned}\hat{A}' = \Bigg\{ &\underbrace{[q^k, q^k]}_{(k-1-\varepsilon_{k,1}-\varepsilon_{k,0})/2 \text{ veces}}\,, \underbrace{q^0}_{1 \text{ veces}}, \underbrace{[q^k, q^{k+1}]}_{\varepsilon_{k,1} \text{ veces}}, \underbrace{[q^{k+1}, q^{k+1}]}_{(k-2-\varepsilon_{k,2}-\varepsilon_{k,1})/2 \text{ veces}}\,, \underbrace{q^1}_{2 \text{ veces}}, \underbrace{[q^{k+1}, q^{k+2}]}_{\varepsilon_{k,2} \text{ veces}}, \dots \\ &\dots, \underbrace{[q^{k+i}, q^{k+i}]}_{(k-1-i-\varepsilon_{k,i+1}-\varepsilon_{k,i})/2 \text{ veces}}\,, \underbrace{q^i}_{i+1 \text{ veces}}, \underbrace{[q^{k+i}, q^{k+i+1}]}_{\varepsilon_{k,i+1} \text{ veces}}, \dots \\ &\dots, \underbrace{[q^{2k-1}, q^{2k-1}]}_{(0-\varepsilon_{k,k}-\varepsilon_{k,k-1})/2=0 \text{ veces}}\,, \underbrace{q^{k-1}}_{k \text{ veces}}, \underbrace{[q^{2k-1}, q^{2k}]}_{\varepsilon_{k,k}=0 \text{ veces}} \Bigg\} \cup \Big\{ \underbrace{q^k}_{\varepsilon_{k,0} \text{ veces}} \Big\}\,,\end{aligned}$$

donde $\varepsilon_{k,i} = 1$ si $k - i \not\equiv 0, 1 (\text{mód}.4)$ y es 0 en otro caso, y los símbolos de $\hat{A}'$ están en orden decreciente según $\prec$.

Observar que los elementos de $\hat{A}' \setminus \Big\{ \underbrace{q^k}_{\varepsilon_{k,0} \text{ veces}} \Big\}$ pueden agruparse en ternas de la forma

$$\Upsilon_i \triangleq \Bigg\{ \underbrace{[q^{k+i}, q^{k+i}]}_{(k-i-1-\varepsilon_{k,i+1}-\varepsilon_{k,i})/2 \text{ veces}}\,, \underbrace{q^i}_{i+1 \text{ veces}}, \underbrace{[q^{k+i}, q^{k+i+1}]}_{\varepsilon_{k,i+1} \text{ veces}} \Bigg\}$$

tiene exactamente $\dfrac{k+1+i+\varepsilon_{k,i+1}-\varepsilon_{k,i}}{2}$ elementos.

Dado $-1 \leq j < k$ definimos

$$\mathcal{R}_j \triangleq \bigcup_{i=0}^{j} \Upsilon_i \,,$$

que tiene cardinal

$$|\mathcal{R}_j| = \sum_{i=0}^{j} \left(\frac{k+1+i+\varepsilon_{k,i+1}-\varepsilon_{k,i}}{2} \right) = \frac{(1+k)(1+j)}{2} + \frac{j(j+1)}{4} + \frac{\varepsilon_{k,j+1}-\varepsilon_{k,0}}{2} \,.$$

Elegimos ahora

$$j \triangleq \text{máx}\{j' \in \{-1,0,\ldots,k-1\} : \left|\mathcal{R}_{j'}\right| \leq 2^{M+1} - Q\} \,, \tag{3.2.22}$$

y escribimos $\mathcal{R}_k \triangleq \{ \underbrace{q^k}_{\varepsilon_{k,0} \text{ veces}} \}$.

Observamos entonces que los elementos de $\mathcal{R}_j$ están en el nivel M de nuestro árbol cuasi-uniforme para $\hat{A}'$ (recordar la proposición 3.1.1), y que las restantes hojas de dicho nivel provienen de $\mathcal{R}_{j+1} \setminus \mathcal{R}_j$. Notar que $\mathcal{R}_{j+1} \setminus \mathcal{R}_j$ no tiene más de k elementos, por lo tanto:

$$2^{M+1} - Q = \frac{(1+k)(1+j)}{2} + \frac{j(j+1)}{4} + \frac{\varepsilon_{k,j+1}-\varepsilon_{k,0}}{2} + \delta \tag{3.2.23}$$

con $0 \leq \delta \leq k$. Los elementos de la forma q^i con $i < k$ en $\mathcal{R}_j$ forman parte del nivel M en T_k, de hecho, analizando los δ elementos más probables de $\{ \underbrace{q^k}_{\varepsilon_{k,0} \text{ veces}} \} \cup (\mathcal{R}_{j+1} \setminus \mathcal{R}_j)$ según el orden $\prec$ concluimos que

$$n_M = \frac{(j+2)(j+1)}{2} + r \,, \tag{3.2.24}$$

donde

$$r = \text{mín}\left\{j+2, \delta - \text{máx}\left\{0, \text{mín}\left\{\delta, \left(k-j-2-\varepsilon_{k,j+2}-\varepsilon_{k,j+1}\right)/2\right\}\right\}\right\} . \tag{3.2.25}$$

Observar que los elementos de $\mathcal{R}_j$ están en el nivel M para todo árbol cuasi-uniforme para $\hat{A}'$, ya que las reordenaciones de los elementos de igual peso solo afectan internamente el orden dentro de cada trío Υ_i. Lo que si puede variar según el orden es r. Por ejemplo, si ordenamos $[q^{k+i}, q^{k+i}] \prec q^i$, tenemos una fórmula más sencilla $r = \text{mín}\{j+2, \delta\}$, que corresponde maximizar r. Esta última elección de orden, como se mecinonó antes, maximiza n_M y n_{M+2}, maximizando la varianza de la longitud. Finalmente, notamos que cualquier r entre el definido por (3.2.25) y $\text{mín}\{j+2, \delta\}$, inclusive, define mediante (3.2.24) un perfil óptimo (esto se sigue de colocar, en nuestra ordenación de mayor a menor, una fracción símbolos que consisten en un árbol de la forma $[q^{k+i}, q^{k+i}]$ antes de los que consisten en una hoja de peso q^k). Resumimos lo expuesto en la siguiente proposición

Proposición 3.2.3. *Los perfiles óptimos (n_M, n_{M+1}, n_{M+2}) para T_k corresponden exactamente (es decir, todo perfil óptimo se describe de esta manera, y recíprocamente, definir un perfil de esta manera da lugar a un perfil óptimo), mediante la relación (3.2.21), a*

$$n_M = \frac{(j+2)(j+1)}{2} + r \,,$$

donde

$$j = \text{máx}\{j' \in \{-1,0,\ldots,k-1\} : \xi(j') \leq 2^{M+1} - Q\} \,, \quad r_{\text{mín}} \leq r \leq r_{\text{máx}} \,.$$

Aquí

$$\xi(j) \triangleq \frac{(1+k)(1+j)}{2} + \frac{j(j+1)}{4} + \frac{\varepsilon_{k,j+1} - \varepsilon_{k,0}}{2}$$

y

$$r_{\text{mín}} \triangleq \text{mín}\left\{j+2, \delta - \text{máx}\left\{0, \text{mín}\left\{\delta, \left(k-j-2-\varepsilon_{k,j+2}-\varepsilon_{k,j+1}\right)/2\right\}\right\}\right\}, \quad r_{\text{máx}} \triangleq \text{mín}\{j+2, \delta\},$$

donde $\delta \triangleq 2^{M+1} - Q - \xi(j)$ *y* $\varepsilon_{k,i} = \mathbf{1}_{k-i\equiv 2,3(\text{ mód }4)}$.

Este resultado nos permite calcular perfiles óptimos rápidamente, o aproximando $\xi(j)$ por un polinomio cuadrático, o notando que $\xi(j)$ es creciente en j y utilizando búsqueda binaria. Esta última opción se encuentra implementada en el apéndice A.2.

Es importante recordar que $r = O(k)$ para todos los perfiles óptimos. Esto implica, como vemos en la siguiente sección, que podemos hallar resultados asintóticos para la redundancia, que resultan independientes de los perfiles elegidos para cada k.

3.2.5. Resultados asintóticos

Finalmente, caracterizamos el comportamiento asintótico de la redundancia. Observamos primero que la longitud media cuando trabajamos con $TDGD(q)$ es

$$L_q(C_k) = \frac{2}{1-q^k} + \frac{(1-q)^2}{(1-q^k)^2} \sum_{0\le a,b<k} \ell\left(T_k(a,b)\right) q^{a+b}. \tag{3.2.26}$$

Denotamos mediante W_M, W_{M+1} y W_{M+2} el peso total normalizado para los símbolos de $\hat{\mathcal{A}}_k$ a los que T_k asigna longiudes M, $M+1$ y $M+2$ respectivamente. Por definición

$$\left(\tfrac{1-q}{1-q^k}\right)^2 \sum_{0\le a,b<k} \ell\left(T_k(a,b)\right) q^{a+b} = M\cdot W_M + (M+1)\cdot W_{M+1} + (M+2)\cdot W_{M+2} \tag{3.2.27}$$

$$= M + 1 + W_{M+2} - W_M, \tag{3.2.28}$$

y, por lo tanto, basta calcular W_M y W_{M+2}.

Lema 3.2.6. *El parámetro j definido en (3.2.22) para T_k satisface*

$$j/k = 2\sqrt{\lambda_k - \tfrac{1}{2}} - 1 + O(1/k), \tag{3.2.29}$$

a medida que $k\to\infty$, donde $\lambda_k \triangleq \dfrac{2^{M+1}}{k^2}$.

Demostración. Dividiendo ambos lados de (3.2.23) entre k^2 tenemos que λ_k satisface, como $Q = \frac{3}{4}k^2 + O(k)$, que:

$$\lambda_k - \tfrac{3}{4} = \frac{(j/k)}{2} + \frac{(j/k)^2}{4} + O(1/k)$$

cuando $k\to\infty$, esto es $4\lambda_k - 2 = (1+j/k)^2 + O(1/k)$.

Tenemos entonces

$$(1+j/k)^2 = 4\lambda_k - 2 + O(1/k), \tag{3.2.30}$$

y queremos obtener un resultado asintótloszico para j/k. Para todo k suficientemente grande el resto en el lado derecho de (3.2.30) tiene valor absoluto menor que $1/2$, y luego $4\lambda_k - 2 \geq (1 + j/k)^2 - 1/2 \geq 1/2$.

Como la derivada de $x \mapsto \sqrt{x}$ está acotada cuando $x \geq 1/2$, se sigue de (3.2.30), por el teorema del valor medio para derivadas, que

$$1 + j/k = \sqrt{4\lambda_k - 2} + O(1/k)\,,$$

para todo k suficientemente grande, probando el resultado. □

Lema 3.2.7. *Si $q = z^{1/k}$ para cierto $z \in (0,1)$, cuando $k \to \infty$ tenemos*

$$W_M = \frac{z^\alpha\ (\alpha\ \log(z) - 1) + 1}{(1-z)^2} + O_z(1/k)\,, \tag{3.2.31}$$

y

$$W_{M+2} = \frac{z^2 + z^{1+\alpha}\ ((1-\alpha)\ \log(1/z) - 1)}{(1-z)^2} + O_z(1/k)\,, \tag{3.2.32}$$

donde $\alpha \triangleq 2\sqrt{\lambda_k - \frac{1}{2}} - 1$, $\lambda_k \triangleq \frac{2^{M+1}}{k^2}$, y O_z indica que la constante depende de z.

Demostración. Dado j como lo elegimos para (3.2.23), observamos que los elementos de $\{\underbrace{q^i}_{i+1 \text{ veces}} : i \leq j\}$ forman parte del nivel M de T_k. Por otro lado, observamos que r en (3.2.24), es necesariamiente $O(k)$, y por lo tanto, la suma de los pesos de estos elementos es $O(k)$ ya que $q^i \leq 1$ para cada $i \geq 0$.

Obtenemos entonces que

$$W_M = \left(\frac{1-q}{1-q^k}\right)^2 \times \left(\sum_{i=0}^{j}(i+1)q^i + O(k)\right),$$

donde $\left(\frac{1-q}{1-q^k}\right)^2$ es la constante de normalización.

Afirmamos entonces que

$$W_M = \frac{q^{j+1}\ (j\,(q-1) + q - 2) + 1}{(1-q^k)^2} + O_z(1/k)\,. \tag{3.2.33}$$

En efecto, tenemos que el primer sumando se sigue del hecho de que

$$\sum_{i=0}^{j}(i+1)q^i = \frac{q^{j+1}\ (j\,(q-1) + q - 2) + 1}{(1-q)^2}\,.$$

Finalmente, el sumando $O_z(1/k)$ surge de que $(1-q)^2 = (1 - z^{1/k})^2 = (\log(z)/k)^2 + O_z(1/k^3)$, y que $1 - q^k = 1 - z$, que es constante en z.

Ahora por el lema 3.2.6 tenemos $j = \alpha\, k + O(1)$. Como $1 - q = \frac{1}{k}\log(1/z) + O_z(1/k^2)$ tenemos que

$$j(q-1) = \alpha\ \log(z) + O_z(1/k)\,.$$

Aplicando el lema 3.2.6 nuevamente, obtenemos $q^j = z^{j/k} = z^{\alpha + O(1/k)} = z^\alpha + O_z(1/k)$. Por lo tanto, volviendo a (3.2.33) obtenemos

$$W_M = \frac{z^\alpha\ (\alpha\ \log(z) - 1) + 1}{(1-z)^2} + O_z(1/k)\,.$$

De manera similar para el nivel $M+2$, los elementos que corresponden al nivel $M+2$ de T_k corresponden a un subconjunto de los símbolos de dos hojas de $\mathcal{R}_k \setminus \mathcal{R}_j$ a menos de un excedente de $O(k)$ elementos.

Por lo tanto, si $j' = k-2-j$, podemos escribir

$$\begin{aligned}\frac{(1-q^k)^2}{(1-q)^2} W_{M+2} &= \sum_{i=0}^{j'-1} (i+1) q^{2k-2-i} + O(k) \\ &= \frac{q^{2\,k-1-j'} \left(q^{j'+1} + (-j'-1)\; q + j' \right)}{(1-q)^2} + O(k)\,,\end{aligned}$$

y el resto es análogo al caso de W_M. □

Observación 3.2.7. Observamos que la constante en la cota de los restos O_z del lema 3.2.7 puede elegirse de modo que no dependa de z, si consideramos $z \in [a,b] \subset (0,1)$, en cuyo caso depende de a y b.

Corolario 3.2.3. *Tenemos*

$$L_{z^{1/k}}(C_k) = \frac{2}{1-z} + M + 1 + \frac{z^\alpha \log(z)}{(1-z)^2} \Big((\alpha-1)z - \alpha \Big) + \frac{z^\alpha - z - 1}{1-z} + O_z(1/k)\,. \quad (3.2.34)$$

Demostración. Notar que

$$L_q(C_k) = \frac{2}{1-q^k} + M + 1 + W_{M+2} - W_M\,,$$

en consecuencia de (3.2.26) y (3.2.28). El resultado se sigue, ahora, de sustituir W_M y W_{M+2} por sus expresiones asintóticas correspondientes según el lema 3.2.7. □

Ahora, si tomamos $z = 1/2$ en particular, tenemos $q = 2^{-1/k}$ que corresponde al punto en el que C_k es óptimo. Simplificando (3.2.34) para dicho caso, encontramos

$$L_q(C_k) = M + 2 + 2^{1-\alpha}\; (\log(2)\; (1+\alpha) + 1) + O(1/k)\,.$$

Observar que, por definición de λ_k, tenemos $\lg \lambda_k = M + 1 - 2\lg k$ y así $M + 2 = 1 + \lg \lambda_k + 2\lg k$, obteniendo entonces

$$L_q(C_k) = 1 + \lg \lambda_k + 2\lg k + 2^{1-\alpha}\; (\log(2)\; (1+\alpha) + 1) + O(1/k)\,. \quad (3.2.35)$$

Por otro lado, la entropía de una variable aleatoria con distribución $TDGD(q)$ es

$$H(q) = -\frac{q \lg q}{1-q} - \lg(1-q) = \lg(e) + \lg\lg(e) + \lg k + o(1)$$

así, por (3.2.35), tenemos

$$R(k) = \tfrac{1}{2} L(C_k) - H(q) = \tfrac{1}{2} \left(1 + \lg \lambda_k \right) + 2^{1-2\sqrt{\lambda_k - 1/2}} \left(1 + \frac{2}{\lg(e)} \sqrt{\lambda_k - 1/2} \right) - C + o(1)$$

cuando $k \to \infty$, donde $C = \lg(e) + \lg\lg(e)$.

Hemos probado entonces la siguiente proposición, que aparece en [17].

Proposición 3.2.4. *Sea $\lambda_k = 2^{M+1}/k^2$ donde M se define como en la proposición 3.2.2. A medida que $k \to \infty$ tenemos que la redundancia (por entero) de C_k en $q = 2^{-1/k}$ es*

$$R(k) = \tfrac{1}{2}\left(1 + \lg \lambda_k\right) + 2^{1-2\sqrt{\lambda_k - \frac{1}{2}}}\left(1 + \frac{2}{\lg(e)}\sqrt{\lambda_k - \tfrac{1}{2}}\right) - C + o(1)\,, \tag{3.2.36}$$

donde $C = \lg(e) + \lg\lg(e)$.

Observación 3.2.8. Se tiene que $M = \lfloor \lg Q \rfloor$ satisface $M \leq \lg(Q) < M + 1$. Como $Q = \left\lceil \frac{k(k-1)}{4} \right\rceil + \frac{k(k+1)}{2} = \frac{3}{4}k^2 + O(k)$, se sigue que $\lg(Q) = \lg(\frac{3}{4}k^2) + o(1)$ y

$$\lg\left(\tfrac{3}{4}k^2\right) - 1 + o(1) < M \leq \lg\left(\tfrac{3}{4}k^2\right) + o(1)\,.$$

Por lo tanto $\lambda_k = \dfrac{2^{M+1}}{k^2} = 2^{M+1-\lg(k^2)}$ satisface, como $\lg(3/4) + o(1) < M + 1 - \lg(k^2) \leq \lg(3/2) + o(1)$, que

$$\frac{3}{4} \lesssim \lambda_k \lesssim \frac{3}{2}\,, \tag{3.2.37}$$

donde $\lesssim$ indica desigualdad a menos de términos asintóticamente despreciables. Para k grande, λ_k recorre el intervalo desde $3/2$ a $3/4$ a medida que crece k, hasta que M pasa al siguiente entero, volviendo entonces a tener $\lambda_k \approx 3/2$ y reiterándose el ciclo.

Observación 3.2.9. Finalmente, si bien tenemos caracterizada la redundancia asintótica sobre el conjunto $\{2^{-1/k} : k \in \mathbb{Z}_{>0}\}$, uno podría preguntarse por el resto de los puntos del intervalo $[1/2, 1]$. Se puede verificar [17] que la redundancia de C_k no varía más de $O(k^{-1})$ cuando $2^{-1/k} \leq q \leq 2^{-1/(k+1)}$, y por lo tanto tenemos caracterizado el comportamiento asintótico.

3.2.6. Códigos para el intervalo $[0, 1/2]$

En este caso se conocen códigos óptimos para los pares de variables geométricas con $q = 2^{-k}$, pero resultan bastante más complejos de describir. Esto aparece explicado en detalle en [17]. Sin embargo, es interesante que, cuando $q \to 0$, los códigos óptimos convergen a un código límite $C_{-\infty}$ (esto no es particular de $d = 2$).

Antes de definir el código límite $\mathcal{C}^d_{\text{lím}}$ para cuando tratamos con una d-upla de variables aleatorias geométricas iid, comenzamos recordando la proposición 3.1.1 que caracteriza los códigos de prefijo óptimos para una distribución cuasi-uniforme. Observar que los códigos de prefijo óptimos para una distribución cuasi-uniforme sobre un conjunto de m elementos queda caracterizados por tener $2^{d+1} - m$ palabras de largo $d = \lfloor \lg(m) \rfloor$, y $2m - 2^{d+1}$ palabras de largo $d + 1$. Este perfil depende únicamente de m, la cantidad de elementos, y no de la distribución cuasi-uniforme particular. Es por esto que podemos hablar de 'el árbol cuasi-uniforme' con m hojas, para referirnos a un árbol con dicho perfil, donde despeciamos las permutaciones de nodos en un mismo nivel.

Definimos $\mathcal{Q}^d_k$ de la siguiente manera: comenzando por el árbol cuasi-uniforme con $1 + \binom{k+d-1}{k}$ hojas, sustituímos una de las hojas más profundas por $\mathcal{Q}^d_{k+1}$, y a las restantes les asociamos hojas de peso q^k. Finalmente, definimos $\mathcal{C}^d_{\text{lím}}$ mediante $\mathcal{C}^d_{\text{lím}} \triangleq \mathcal{Q}^d_0$. Esta definición se encuentra ilustrada en la figura 3.2.3.

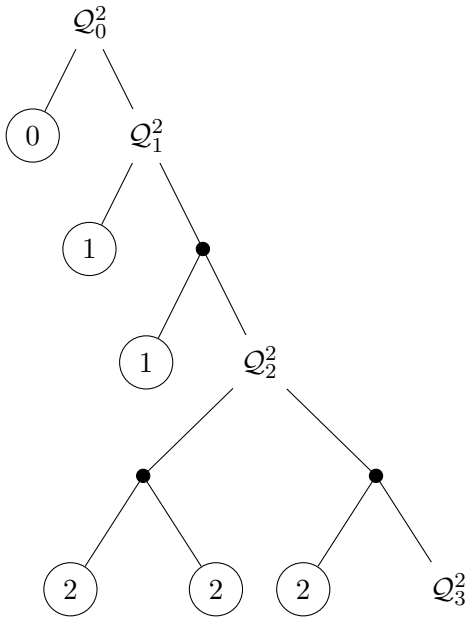


Figura 3.2.3: Primeros niveles del código límite $\mathcal{C}^d_{\text{lím}}$ para $d = 2$.

Antes de poder enunciar precisamente nuestro teorema de convergencia, debemos definir precisamente lo que queremos decir con 'convergencia'. Cuando probamos la existencia de los códigos óptimos para fuentes con entropía finita (teorema 2.7.2), definimos la convergencia de secuencias de códigos, sin embargo, en el caso que sigue tenemos una convergencia un poco más fuerte a medida que $q \to 0$.

Sea $\mathcal{C}^d_q$ un código de prefijos óptimo, arbitrario, para codificar una d-upla de variables aleatorias geométricas $(X_1, \ldots, X_d)$ con distribución $ODGD(q)$. Lo que probamos a continuación es que, dado un nivel $\ell \geq 0$ arbitrario, existe $q_\ell > 0$ tal que si $q \in (0, q_\ell)$ entonces los códigos de prefijo óptimos para $(X_1, \ldots, X_d)$ coinciden con $\mathcal{C}^d_{\text{lím}}$ en sus primeros ℓ niveles, a menos de reordenación de los nodos en cada nivel. A esta convergencia la denominamos *uniforme*.

Teorema 3.2.3. *Sea $d \in \mathbb{Z}_{>0}$ fijo, y sean $X_1, \ldots, X_d$ variables iid, con distribución $ODGD(q)$. Los códigos de prefijo óptimos para $(X_1, \ldots, X_d)$ convergen a $\mathcal{C}^d_{\text{lím}}$ uniformemente cuando $q \to 0$.*

Observación 3.2.10. Notar en particular que en el caso $d = 1$ se tiene que $\mathcal{C}^1_{\text{lím}}$ coincide exactamente con G_1, que es el código óptimo para el subintervalo $[0, 1/2]$.

Lema 3.2.8. *Supongamos tenemos un alfabeto infinito, $\mathbb{Z}_{>0}$ sin pérdida de generalidad, con probabildiades $p_1 = p_2 = \ldots = p_n = p$, y $\sum_{i>n} p_i < p$. Cualquier código libre de prefijo óptimo para esta fuente debe estar conformado por un árbol cuasi-uniforme con $n+1$ hojas, en el que n hojas están asociadas a los primeros n símbolos, y la hoja restante, que debe ser del nivel más profundo, es sustituída por un subárbol que contiene únicamente hojas asociadas a $n+1, n+2 \ldots$.*

Demostración. Es claro que si todas las hojas correspondientes a $n+1, n+2, \ldots$ están en un subárbol todos juntos, sin involucrar ningún nodo $\{1, \ldots, n\}$, se sigue el resultado dado que la fuente $\{p_1, \ldots, p_n, \sum_{i>n} p_i\}$ es claramente cuasi-uniforme. Para verificar que esto último debe ser el caso, consideramos $\mathcal{T}_1, \ldots, \mathcal{T}_m$, subárboles[d] de nuestro código óptimo, disjuntos, tales que sus hojas no corresponden a ningún símbolo de $[n] \triangleq \{1, \ldots, n\}$,y que sus respectivos subárboles hermanos sí tienen alguna hoja asociada a un símbolo de $[n]$.

A partir de una cierta profundidad las hojas solo pueden corresponder a nodos de $\mathbb{Z}_{>0} \setminus [n]$, ya que el conjunto $[n]$ es finito. Esto implica que en efecto un tal conjunto ordenado $\mathcal{T}_1, \ldots, \mathcal{T}_m$ ha de ser finito, y, aún más, tiene su cardinal acotado. Por otro lado, es claro que existe al menos un conjunto ordenado $\mathcal{T}_1, \ldots, \mathcal{T}_m$ en tales condiciones, en efecto, comenzando por una hoja de $\mathbb{Z}_{>0} \setminus [n]$, subimos en el árbol encontrarnos en un nodo v que tenga alguna hoja de $[n]$ en su descendencia, y tomamos el hijo de v que visitamos anteriormente. El árbol definido por dicha raíz cumple con las condiciones pedidas.

Consideramos entonces $\mathcal{T}_1, \ldots, \mathcal{T}_m$ con $m \geq 1$ máximo. Observar que, bajo estas condiciones, todas las hojas asociadas a un símbolo de $\{n+1, n+2, \ldots\}$, deben aparecer en $\mathcal{T}_1, \ldots, \mathcal{T}_m$. En efecto, si v_0 es una hoja que no se encuentra en ninguno de los subárboles $\mathcal{T}_1, \ldots, \mathcal{T}_m$, al ser estos subárboles, el nodo padre v_1 de v_0, no puede encontrarse en $\mathcal{T}_1, \ldots, \mathcal{T}_m$, y así sucesivamente, el padre v_2 de v_1, etc. Como v_0 está asociada a un símbolo de $\{n+1, n+2, \ldots\}$, existe un primer $i \geq 1$ tal que v_i tiene alguna hoja asociada a un símbolo de $[n]$ en uno de sus subárboles (en el peor caso v_i es la raíz del árbol del código). Observamos entonces el subárbol $\mathcal{T}$ con raíz v_{i-1} cumple con nuestras condiciones, y podemos considerar $\mathcal{T}, \mathcal{T}_1, \ldots, \mathcal{T}_m$, contradiciendo la maximalidad de m.

Si $m = 1$, estamos en el caso mencionado en el primer párrafo, así que asumimos que $m \geq 2$. Denotamos $w_i = w(\mathcal{T}_i)$. Observar que los pesos satisfacen $w_1 + \ldots + w_m = \sum_{i>n} p_i < p$.

Como nuestro árbol es óptimo para la fuente infinita, en particular, tomando $\mathcal{T}_i$ como hojas, el árbol finito resultante debe resultar también óptimo para la fuente finita con probabilidades agregadas

$$\{p_1, \ldots, p_n, w_1, \ldots, w_m\}.$$

Se sigue entonces que cada nodo de peso $w_i < p = p_j$ para $j \leq n$, se encuentra a profundidad mayor o igual que la de los asociados a símbolos de probabilidad p. También, al ser óptimo, sabemos que debe ser subárbol, y que los elementos de mayor profundidad vienen de a pares de hermanos, uno de estos podemos suponer es $\mathcal{T}_1$, pero su hermano no puede ser otro elemento $\mathcal{T}_j$ ya que esto contradice su definición. Por lo tanto, un par de mayor profundidad, sin pérdidad de generalidad es $(\mathcal{T}_1, v_1)$ con $v_1 \leq n$ (estamos identificando la hoja con el símbolo que representa). Pero nosotros sabemos que v_1 se encuentra a profundidad menor o igual que $\mathcal{T}_2$, por lo tanto $\mathcal{T}_2$ está también en profundidad máxima, y

[d] En este contexto, un subárbol es un poco más fuerte que ser simplemente un subgrafo del árbol. Concretamente, queremos que si un nodo forma parte del subárbol, toda su descendencia también lo haga.

debe tener un hermano $v_2 \leq n$, $v_2 \neq v_1$, por el mismo argumento. Luego, tenemos pares de hermanos distintos $(\mathcal{T}_1, v_1)$, $(\mathcal{T}_2, v_2)$ a profundidad máxima.

Para completar ahora la prueba, notamos que podemos considerar elevar v_2 un nivel, y colocar $[\mathcal{T}_1, \mathcal{T}_2]$ en lugar de $\mathcal{T}_1$ obteniendo una variación de longitud media:

$$\Delta L = -p_2 + (w_1 + w_2) = -p + w_1 + w_2 < 0\,,$$

ya que $w_1 + w_2 \leq w_1 + \ldots + w_m < p$, reduciendo la longitud promedio, un absurdo. □

Prueba. *(del teorema 3.2.3).* Notamos que tenemos una fuente con exactamente $\binom{k+d-1}{k}$ símbolos de probabilidad $(1-q)^d \times q^k$ para cada $k \in \mathbb{Z}_{\geq 0}$, dado que formalmente, por el Binomio Generalizado, tenemos

$$\sum_{i_1,\ldots,i_d \geq 0} q^{i_1+\ldots+i_d} = \frac{1}{(1-q)^d} = \sum_{k=0}^{\infty} \binom{k+d-1}{k} q^k\,.$$

En particular, tenemos exactamente uno con probabilidad $(1-q)^d$, que es la máxima posible. Cuando $q \to 0$, notamos que si q es suficientemente pequeño, entonces $1-(1-q)^d < (1-q)^d$ y luego por el lema 3.2.8 se sigue que cualquier árbol óptimo ha de estar formado por un árbol cuasi-uniforme con una hoja de probabilidad $(1-q)^d$, y otra que acumula el resto de los símbolos.
Una vez que sabemos esto, podemos mirar el subárbol que acumula el resto de las probabilidades aisladamente, ya que este aislado debe ser óptimo también, una vez que renormalizamos las probabilidades.

Una vez que hemos finalizado con los nodos de firma $k-1$, suponiendo $q < q_{k-1}$ para cierto $q_{k-1} \in (0,1)$, tenemos los nodos de firma mayor o igual a k en un subárbol único. Luego de normalizar por $w(\mathcal{Q}_k^d) = \sum_{n \geq k} \binom{n+d-1}{n} q^n$, los nodos de firma k tienen pesos

$$\frac{q^k}{w(\mathcal{Q}_k^d)} = \binom{k+d-1}{k}^{-1} + o(1)\,,$$

mientras que el resto de las firmas mayores que k acumuladmente son

$$\frac{w(\mathcal{Q}_{k+1}^d)}{w(\mathcal{Q}_k^d)} = o(1)$$

cuando $q \to 0$, luego para todo q suficientemente pequeño tenemos las condiciones del lema 3.2.8, y podemos repetir el argumento.

Concretamente podemos ver que

$$\frac{w(\mathcal{Q}_{k+1}^d)}{w(\mathcal{Q}_k^d)} = \frac{\sum_{n \geq k+1} \binom{n+d-1}{n} q^n}{\sum_{n \geq k} \binom{n+d-1}{n} q^n} = \frac{k+d}{k+1} \times q + O(q^2),$$

ya que $\binom{k+d}{k+1} = \frac{k+d}{k+1} \times \binom{k+d-1}{k}$. □

Observación 3.2.11. Observar que, por el lema 3.2.8, si $w(\mathcal{Q}_{k+1}^d) < q^k$, esto es

$$q \times \left\{ \binom{k+d}{k+1} + \binom{k+1+d}{k+2} \cdot q + \ldots \right\} < 1\,,$$

entonces cualquier árbol óptimo coincide con $\mathcal{C}_{\text{lím}}^d$ en los primeros $k+1$ árboles cuasi-uniformes (notar que la expresión a la izquierda crece con k).

Ejemplo 3.2.2. En particular, si $d = 1$, la observación 3.2.11 nos dice que si

$$\frac{q}{1-q} = q \times \{1 + q + \ldots\} = q \times \left\{ \binom{k+1}{k+1} + \binom{k+1+1}{k+2} \cdot q + \ldots \right\} < 1\,,$$

en otras palabras $q < 1/2$, nuestro árbol es igual a $\mathcal{C}^1_{\text{lím}}$, dado que la desigualdad resultante no depende de k. Derivamos entonces, nuevamente, que $\mathcal{C}^1_{\text{lím}} = G_1$ es óptimo en $[0, 1/2)$.

Ejemplo 3.2.3. Para el caso $d = 2$ tenemos

$$\binom{k+2}{k+1} + \binom{k+3}{k+2} \cdot q + \ldots = \sum_{j=0}^{\infty} (k+2+j) \cdot q^j = \frac{k+2}{1-q} + \frac{q}{(1-q)^2}$$

y luego encontrar los $q \in (0,1)$ que satisfacen la desigualdad de la observación 3.2.11 corresponden a los $q \in (0,1)$ tales que $q \times \left\{ \frac{k+2}{1-q} + \frac{q}{(1-q)^2} \right\} < 1$. Se sigue de resolver una ecuación cuadrática, para obtener que si

$$\begin{aligned} q &< \frac{k+4-\sqrt{(k+2)^2+4}}{2(k+2)} \\ &= \frac{(k+4)^2 - (k+2)^2 - 4}{2(k+2)\left(k+4+\sqrt{(k+2)^2+4}\right)} \\ &= \frac{2}{k+4+\sqrt{(k+2)^2+4}}\,, \end{aligned}$$

cualquier código óptimo coincide con $\mathcal{C}^2_{\text{lím}}$ en los primeros $k+1$ árboles cuasi-uniformes (estos corresponden a las firmas $s \in \{0, 1, \ldots, k\}$). Relajando la desigualdad, podemos escribirlo de manera más elegante como

$$q \leq \frac{1}{k+4}\,.$$

En efecto porque $k + 4 + \sqrt{(k+2)^2+4} < k + 4 + (k+3) = 2k + 7 < 2k + 8$ y así

$$q \leq \frac{1}{k+4} = \frac{2}{2k+8} < \frac{2}{k+4+\sqrt{(k+2)^2+4}}$$

y se satisface $q \times \left\{ \frac{k+2}{1-q} + \frac{q}{(1-q)^2} \right\} < 1$.

Observación 3.2.12. Observar también que, si $w(\mathcal{Q}^d_{k+1}) > q^k > w(\mathcal{Q}^d_k)$ y $\binom{k+d-1}{k} + 1$ no es una potencia de 2, entonces todo árbol óptimo debe diferir de $\mathcal{C}^d_{\text{lím}}$ en el nivel correspondiente a $\mathcal{Q}^d_k$.

Observación 3.2.13. Supongamos que tenemos una fuente infinita con distribución $p_1 \geq p_2 \geq \ldots$, y existe $(m_k)_{k=0}^{\infty} \subset \mathbb{Z}_{\geq 0}$ creciente, tal que $m_0 = 0$ y $p_{m_k} > \sum_{i>m_k} p_i$ para cada $k \geq 1$. Un argumento análogo al del lema 3.2.8 nos permite construir un código libre de prefijo óptimo, siempre que la *entropía sea finita* (para tener garantizada la existencia de un tal óptimo). En efecto, primero aplicamos el algoritmo de Huffman a la fuente con alfabeto $\{1, \ldots, m_1, *\}$ y probabilidades respectivas $\{p_1, \ldots, p_{m_1}, \sum_{i>m_1} p_i\}$, obteniendo un cierto código $C^H_1 : \{1, \ldots, m_1, *\} \to \mathbb{B}^*$. De manera análoga, aplicamos el algoritmo Huffman sobre la fuente con el alfabeto $\{m_1+1, \ldots, m_2, *\}$ y probabilidades respectivas $\{p_{m_1+1}, \ldots, p_{m_2}, \sum_{i>m_2} p_i\}$, obteniendo un código $C^H_2 : \{m_1+1, \ldots, m_2, *\} \to \mathbb{B}^*$, y así sucesivamente. Definimos entonces $C : \{1, 2, \ldots\} \to \mathbb{B}^*$ mediante

$$C(i) = C^H_1(*) \bowtie \ldots C^H_k(*) \bowtie C^H_{k+1}(i)$$

si $m_k < i \leq m_{k+1}$. El código C resulta óptimo para la fuente original, si su entropía es finita.

Ver [8] y [10] por construcciones similares a la de esta observación, pero con hipótesis más débiles. Las pruebas en dichas referencias están basadas en la técnica de Gallager y van Voorhis [12], y preceden la prueba de existencia que apareció por primera vez en [7].

Finalmente, concluímos esta sección estudiando la longitud promedio del código límite $\mathcal{C}_{\text{lím}} = \mathcal{C}^2_{\text{lím}}$, para dimensión $d = 2$, discutiendo cuándo es que resulta más conveniente utilizar $\mathcal{C}_{\text{lím}}$ que C_1, y la diferencia entre utilizar el código límite y los códigos óptimos para cada $q \in (0, 1/2)$.

La siguiente proposición nos proporciona una forma eficiente de calcular la longitud promedio de $\mathcal{C}_{\text{lím}}$. Este resultado aparece probado originalmente en [17, Corollary 5].

Proposición 3.2.5. *La longitud promedio del código $\mathcal{C}_{\text{lím}}$ para la distribución $TDGD(q)$ es*

$$L_q\left(\mathcal{C}_{\text{lím}}\right) = 1 + \frac{1}{1-q} \sum_{t\geq 0} q^{2^t} \left(2^t(1-q) + 2\right).$$

Demostración. Indexando según $2^t \leq s + 1 < 2^{t+1}$:

$$\mathcal{L}_q\left(\mathcal{C}_{\text{lím}}\right) = (1-q)^2 \times \sum_{t\geq 0} \sum_{s=2^t-1}^{2^{t+1}-2} q^s D(t,s)\,,$$

donde $D(t,s) = t\cdot(s+1) + (2(s+2) - 2^{t+1} - 1) + (s+1)\cdot\sum_{0\leq k<s}\lceil \lg(k+2)\rceil$, los primeros dos términos se corresponden a la suma de las profunidades de las hojas dentro del árbol cuasi-uniforme para los hojas de firma s, y el resto es lo que se suma en el camino para llegar a dicho árbol multiplicado por $(s+1)$, el número de hojas de firma s.

Notamos que $2^{t'} \leq s' + 1 < 2^{t'+1} \Leftrightarrow 2^{t'} + 1 \leq s' + 2 \leq 2^{t'+1} \Leftrightarrow \lceil \lg(s'+2)\rceil = t' + 1$ con lo cual $\sum_{0\leq k<s}\lceil \lg(k+2)\rceil = (t+1)\cdot(s-2^t) + \sum_{t'=0}^{t} t'\cdot\left(2^{t'+1} - 1 - 2^{t'} + 1\right)$ y calculando esto

$$\sum_{0\leq k<s} \lceil \lg(k+2)\rceil = (t+1)\cdot(s+1-2^t) + (t-1)\ 2^t + 1.$$

Concluímos entonces que $D(t,s) = (-2\,s-4)\ 2^t + \left(s^2+3\,s+2\right)\ t + s^2 + 5\,s + 5$. Sumando, haciendo uso de la identidad $\sum_{k=0}^{m-1} k^r x^k = (x\partial_x)^r\left(\frac{1-x^m}{1-x}\right)$, tenemos

$$\mathcal{L}_q\left(\mathcal{C}_{\text{lím}}\right) = (1-q)^2 \times \sum_{t\geq 0}\left(q^{2^{t+1}-1}A(t) + q^{2^t-1}B(t)\right)\,,$$

donde escribimos

$$A(t) = \frac{4\,(q-1)^2 2^{2\,t}t + 2(q-1)\,(q-2-(q+1)t)2^t + 2\,qt + 2\,q + 1 - q^2}{(q-1)^3}\,,$$
$$B(t) = \frac{\left(q^2 - \left(q^2 - 2\,q + 1\right)t - 2\,q + 1\right)2^{2\,t} - \left(q^2 - \left(q^2-1\right)t - 2\,q + 1\right)2^t + q^2 - 2\,qt - 2\,q - 1}{(q-1)^3}\,.$$

Las funciones $A(t)$ y $B(t)$ se obtienen agrupando términos a partir de

$$\sum_{s=2^t-1}^{2^{t+1}-2} q^s D(t,s) = \sum_{s=0}^{2^{t+1}-2} q^s D(t,s) - \sum_{s=0}^{2^t-2} q^s D(t,s)\,,$$

los términos con $q^{2^{t+1}-1}$ que forman $A(t)$ provienen de $\sum_{s=0}^{2^{t+1}-2} q^s D(t,s)$, mientras que los de q^{2^t-1} provienen de $\sum_{s=0}^{2^t-2} q^s D(t,s)$.

Aquí notando que $B(0) = \dfrac{1-q^2+2q}{(1-q)^3}$ y $A(t-1)+B(t) = q\dfrac{2^t-2^tq+2}{(1-q)^3}$, obtenemos

$$\begin{aligned} L_q\left(\mathcal{C}_{\text{lím}}\right) &= (1-q)^2 \times \left\{ B(0) + \sum_{t\geq 1} q^{2^t-1}\left(B(t)+A(t-1)\right)\right\} \\ &= (1-q)^2 \times \left\{ \frac{1-q^2+2q}{(1-q)^3} + \sum_{t\geq 1} q^{2^t}\frac{2^t-2^tq+2}{(1-q)^3}\right\} \\ &= 1 + \frac{1}{1-q}\sum_{t\geq 0} q^{2^t}\cdot\left(2^t(1-q)+2\right). \end{aligned}$$

□

Observación 3.2.14 (Aproximación de la longitud media $L_q\left(\mathcal{C}_{\text{lím}}\right)$ mediante sumas parciales)**.** La serie obtenida en la proposición 3.2.5 converge muy rápidamente debido a la doble exponencial q^{2^t}. Como en este caso $q \in (0, 1/2)$, tenemos

$$\begin{aligned} \left| L_q\left(\mathcal{C}_{\text{lím}}\right) - \left\{ 1 + \frac{1}{1-q}\sum_{t=0}^{N} q^{2^t}\cdot\left(2^t(1-q)+2\right)\right\}\right| &< \frac{1}{1-1/2}\times\sum_{t=N+1}^{\infty} 2^{-2^t}\cdot\left(2^t+2\right) \\ &\leq 2\times\left\{\sum_{t=2^{N+1}-N-1}^{\infty} 2^{-t} + \sum_{t=2^{N+1}}^{\infty} 2^{-t}\cdot 2\right\} \\ &= 2\times\left\{2^{N+2-2^{N+1}} + 2^{2-2^{N+1}}\right\}, \qquad (*) \end{aligned}$$

donde utilizamos que $t - 2^t$ es estrictamente decreciente en $\mathbb{Z}_{>0}$. Evaluando $(*)$ para $N = 4$, por ejemplo, observamos que el error luego de sumar 5 términos ($t = 0, \ldots, 4$) es menor que 3.17×10^{-8} bits/par.

En la figura 3.2.4 presentamos una gráfica aproximada de la redundancia del código límite bidimensional $\mathcal{C}_{\text{lím}}$, comparado con el código bidimensional óptimo para $q = 2^{-1}$, que corresponde a la concatenación de códigos de Golomb G_1. Aproximando $L_q(\mathcal{C}_{\text{lím}})$ considerando solamente 5 términos de la serie, podemos ver que recién para $q \lesssim 0.33715$ resulta más conveniente utilizar $\mathcal{C}_{\text{lím}}$ en lugar de C_1 definido en (3.2.12). Como es explicado en [17, Section IV], en la práctica no hay grandes pérdidas por utilizar el código límite en lugar de los códigos bidimensionales óptimos para $q = 2^{-k}$, de hecho, la diferencia en cuanto a redundancia es prácticamente despreciable para $q < 1/4$.

Observación 3.2.15. El hecho de que no sea mucho peor utilizar el código límite que los códigos óptimos puede verse como una consecuencia del ejemplo 3.2.3 y una modificación de la proposición 3.2.5. La longitud esperada de $\mathcal{C}^2_{\text{lím}}$ que proviene de los árboles cuasi-uniformes de firmas $t \geq 2^\ell - 1$ es, por

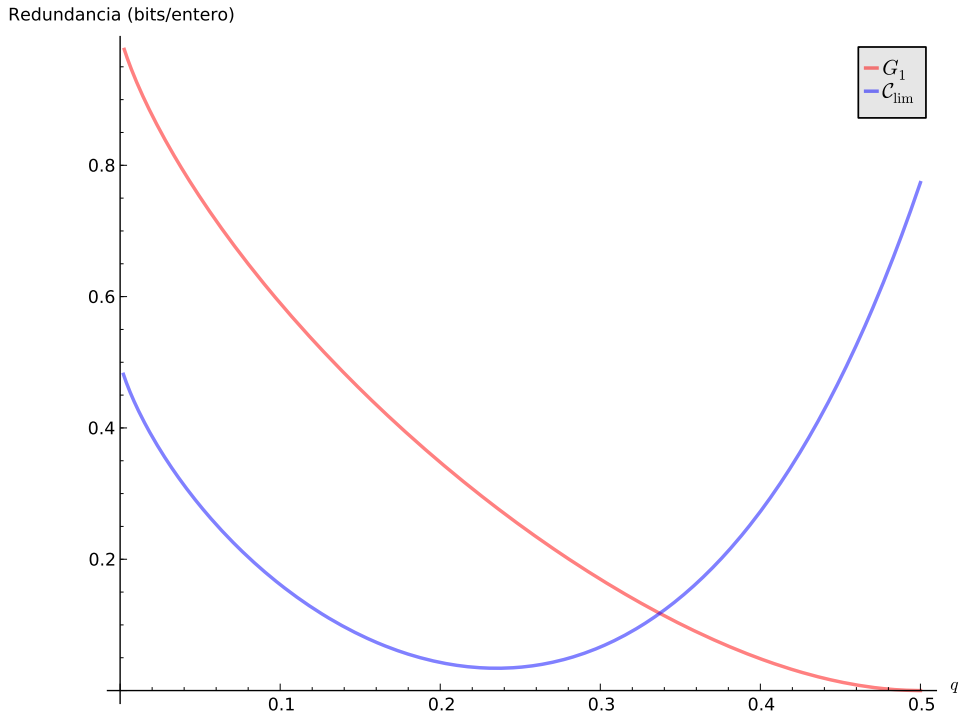


Figura 3.2.4: Redundancia normalizada por entero para $q \in [0, 1/2]$.

el mismo argumento que en la proposición 3.2.5:

$$\begin{aligned}
\Lambda_\ell(q) &= (1-q)^2 \times \sum_{t\geq \ell} \sum_{s=2^t-1}^{2^{t+1}-2} q^s D(t,s) \\
&= (1-q)^2 \times \sum_{t\geq \ell} \left(q^{2^{t+1}-1} A(t) + q^{2^t-1} B(t) \right) \\
&= (1-q)^2 \times \left\{ q^{2^\ell-1} B(\ell) + \sum_{t\geq \ell+1} q^{2^t-1} \left(B(t) + A(t-1) \right) \right\} \\
&= (1-q)^2 \times \left\{ q^{2^\ell-1} B(\ell) + \sum_{t\geq \ell+1} q^{2^t} \frac{2^t - 2^t q + 2}{(1-q)^3} \right\},
\end{aligned}$$

lo cual nos permite acotar $\Lambda_\ell(q)$, procediendo como en la observación anterior.

Por el ejemplo 3.2.3 sabemos que si $q \leq \dfrac{1}{2^\ell + 2}$, entonces las hojas de firmas no mayores a $2^\ell - 2$ tienen igual longitud en $\mathcal{C}^2_{\text{lím}}$ que en cualquier código óptimo de prefijos para $TDGD(q)$. Por ejemplo, tomando $\ell = 3$, si $q \leq \dfrac{1}{10}$, tenemos que, para cualquier código libre de prefijos C óptimo para $TDGD(q)$, se cumple

$$\begin{aligned}
0 &\leq L_q\left(\mathcal{C}^2_{\text{lím}}\right) - L_q(C) \\
&\leq \Lambda_3(q) \\
&\leq \Lambda_3(0.1) < 1.51 \times 10^{-5} \text{ bits/par}. \qquad (\Lambda_3(q) \text{ es no-decreciente en } [0, 1/2])
\end{aligned}$$

A pesar de que esta estimación es un poco grosera, demuestra que para $q \leq 0.1$ la diferencia entre usar el código límite y el código óptimo para $TDGD(q)$ es despreciable con respecto a la redundancia de $\mathcal{C}^2_{\text{lím}}$, que tiene un mínimo de aproximadamente 0.068 bits/par. $\Diamond$

3.3. Dimensiones mayores

Es razonable ahora preguntarse si tenemos un análogo del teorema 3.2.2 para dimensiones mayores a 2. De acuerdo a [17] esto es cierto, aunque solo presenta la prueba para el caso $d = 2$ por simplicidad. A continuación mostramos cómo haríamos para el caso $d = 3$. Nuestra función generatriz a cubrir es $1/(1-q)^3 = \sum_{i=0}^{\infty} \binom{i+2}{i} q^i$.

Definimos

$$\mathcal{F}_s := \bigcup_{i=0}^{k-1} \{ \underbrace{q^{i+s}\mathcal{T}^0_{q^k}}_{\binom{s+i+2}{2} \text{ veces}}, \underbrace{q^{i+s}\mathcal{T}^1_{q^k}}_{\binom{k+s+i+2}{2} \text{ veces}}, \underbrace{q^{i+s}\mathcal{T}^2_{q^k}}_{k^2+\binom{k+s+i+2}{2}-\binom{s+i+2}{2} \text{ veces}}, \underbrace{q^{i+s}\mathcal{T}^3_{q^k}}_{k^2 \text{ veces}} \}$$

y $W_s = \mathcal{H}_s \cup F_s$ donde $\mathcal{H}_s = \left\{ \underbrace{q^i}_{\binom{i+2}{2} \text{ veces}} \mathcal{T}^0_{q^k} : i < s \right\}$.

¿Por qué elegir esta fuente?

- Primero, para poder tener árboles $\mathcal{T}^3_{q^k}$ al final, necesitamos tener alguno en $\mathcal{F}_s$. La forma más sencilla ahora de que tengan un buen *unfolding* es entonces seguir el esquema de los demás casos y buscar $\mathcal{F}_s$ de la forma

$$\mathcal{F}_s = \bigcup_{i=0}^{k-1} \{ \underbrace{q^{i+s}\mathcal{T}^0_{q^k}}_{f_0(i+s) \text{ veces}}, \underbrace{q^{i+s}\mathcal{T}^1_{q^k}}_{f_1(i+s) \text{ veces}}, \underbrace{q^{i+s}\mathcal{T}^2_{q^k}}_{f_2(i+s) \text{ veces}}, \underbrace{q^{i+s}\mathcal{T}^3_{q^k}}_{f_3(i+s) \text{ veces}} \}.$$

- Si queremos transformar W_s en W_{s-k}, las hojas $q^{i+s-k}\mathcal{T}^0_{q^k}$ de W_{s-k} deben provenir de $\mathcal{H}_s$. Hay exactamente $\binom{s+i-k+2}{2}$ hojas $q^{i+s-k}\mathcal{T}^0_{q^k}$ en $\mathcal{H}_s$. Por lo tanto, vamos a pedir que el número de hojas $q^{i+s}\mathcal{T}^0_{q^k}$ sea exactamente

$$f_0(s+i) = \binom{s+i+2}{2}.$$

- Como para un buen *unfolding* queremos que todas las hojas $q^{i+s}\mathcal{T}^0_{q^k}$ desaparezcan, buscamos que todas estas puedan hacer merge con árboles $q^{i+s}\mathcal{T}^1_{q^k}$, formando los árboles $q^{i+s-k}\mathcal{T}^1_{q^k}$ de W_{s-k}. Así pues, queremos $f_1(s+i-k) = f_0(s+i)$ y podemos tomar

$$f_1(s+i) = \binom{s+k+i+2}{2}.$$

- Por el mismo motivo, los restantes $q^{i+s}\mathcal{T}^1_{q^k}$ tienen que hacer merge con $q^{i+s}\mathcal{T}^2_{q^k}$, formando los árboles $q^{i+s-k}\mathcal{T}^2_{q^k}$ del siguiente W_{s-k}. Así pues, pedimos $f_2(s+i-k) = f_1(s+i) - f_0(s+i) \geq 0$ y podemos tomar

$$f_2(s+i) = f_1(s+i+k) - f_0(s+i+k) = k^2 + \binom{k+s+i+2}{2} - \binom{s+i+2}{2}.$$

- Finalmente, los restantes $q^{i+s}\mathcal{T}^2_{q^k}$ tienen que hacer merge con $q^{i+s}\mathcal{T}^3_{q^k}$, formando los árboles $q^{i+s-k}\mathcal{T}^3_{q^k}$ del siguiente nivel. Por lo tanto $f_3(s+i-k) = f_2(s+i) - f_1(s+i)$ y así tomamos

$$f_3(s+i) = f_2(s+i+k) - f_1(s+i+k) = k^2\,,$$

que es una constante, como se debería esperar dado que no pueden sobrar árboles $q^{i+s}\mathcal{T}^3_{q^k}$.

Por construcción, si $s \geq k$, aplicando el algoritmo de Huffman podemos llevar W_s a W_{s-k}. Observar que no verificamos que cada firma i aparezca el número correcto de veces $\binom{i+2}{i}$ al expandir cada árbol en W_s (esto es, que los árboles particionen las firmas). En efecto, como en realidad W_s lo podemos obtener a partir de $W_{s+k\times t}$ para cada $t \geq 0$, una firma s' dada debe aparecer el número adecuado de veces, ya que para t suficientemente grande se tiene $s + k \times t > s'$ y allí $q^{s'}$ solo puede aparecer en $\mathcal{H}_{s+k\times t}$ y así aparece exactamente $\binom{s'+2}{2}$ veces.

Considerando inicialmente $s = t \times k$, y $S_t = \mathcal{F}_{t\times k}$ podemos llevar S_t a

$$S_0 = \bigcup_{i=0}^{k-1} \{ \underbrace{q^i\mathcal{T}^0_{q^k}}_{\binom{i+2}{2}\text{ veces}}, \underbrace{q^i\mathcal{T}^1_{q^k}}_{\binom{k+i+2}{2}\text{ veces}}, \underbrace{q^i\mathcal{T}^2_{q^k}}_{k^2+\binom{k+i+2}{2}-\binom{i+2}{2}\text{ veces}}, \underbrace{q^i\mathcal{T}^3_{q^k}}_{k^2\text{ veces}} \}.$$

Aplicando el algoritmo de Huffman nuevamente obtenemos

$$S_{-1} = \bigcup_{i=0}^{k-1} \left\{ \underbrace{q^{i-k}\mathcal{T}^1_{q^k}}_{\binom{i+2}{2}\text{ veces}}, \underbrace{q^{i-k}\mathcal{T}^2_{q^k}}_{k^2+\binom{i+2}{2}-\binom{i-k+2}{2}\text{ veces}}, \underbrace{q^{i-k}\mathcal{T}^3_{q^k}}_{k^2\text{ veces}} \right\}.$$

Aquí notamos que para $i < k$ tenemos :

$$k^2 \geq k^2 - \binom{i-k+2}{2} = k^2 - \frac{(i-k+2)\cdot(i-k+1)}{2} \geq 0$$

en efecto, el mínimo se da para $i = 0$ si $k > 2$.

Afirmamos entonces que podemos aplicar el algoritmo Huffman de nuevo, obteniendo

$$S_{-2} = \bigcup_{i=0}^{k-1} \left\{ \underbrace{q^{i-2k}\mathcal{T}^2_{q^k}}_{\binom{i+2}{2}\text{ veces}}, \underbrace{q^{i-2k}\mathcal{T}^3_{q^k}}_{k^2-\binom{i-k+2}{2}\text{ veces}}, \underbrace{q^{i-k}\mathcal{T}^3_{q^k}}_{\binom{i-k+2}{2}\text{ veces}} \right\}.$$

En efecto, comenzando por $i = k-1$, al aplicar el algoritmo de Huffman comenzamos haciendo merge de $\binom{i+2}{i}$ símbolos $q^{i-k}\,\mathcal{T}^1_{q^k}$ con otros tantos símbolos $q^{i-k}\,\mathcal{T}^2_{q^k}$, generando $\binom{i+2}{i}$ símbolos que

denotamos $q^{i-2k}\,\mathcal{T}^1_{q^k}$. Seguidamente, realizamos merge de $k^2-\binom{i-k+2}{2}$ símbolos $q^{i-k}\,\mathcal{T}^2_{q^k}$ con otros tantos $q^{i-k}\,\mathcal{T}^3_{q^k}$, generando $k^2-\binom{i-k+2}{2}$ símbolos que denotamos $q^{i-2k}\,\mathcal{T}^3_{q^k}$. En este punto aún quedan $\binom{i-k+2}{2}$ símbolos $q^{i-k}\mathcal{T}^3_{q^k}$ excedentes.

Como $\binom{i-k+2}{2}=\binom{k-1-i}{2}$, que decrece al aumentar i, los $\binom{i-k+2}{2}$ símbolos $q^{i-k}\mathcal{T}^3_{q^k}$ excedentes se pueden aparear con los que serían excedentes para el siguiente i (en orden decreciente) $q^{i-1-k}\mathcal{T}^3_{q^k}$. Una vez hecho esto, podemos repetir el argumento con el siguiente i en orden, ya que $[q^{i-k}\mathcal{T}^3_{q^k}, q^{i-1-k}\mathcal{T}^3_{q^k}]$ tiene peso $q^{i-k}+q^{i-1-k}>2q^{i-k}=q^{i-2k}$. Al finalizar con cada i, **deshacemos** dichos merge entre elementos 'excedentes' obteniendo entonces S_{-2}.

Seguidamente, aplicando el algoritmo de Huffman de nuevo mediante un argumento similar (que es cada vez menos sencillo a medida aumenta dimensión), arrivamos a

$$S_{-3}=\bigcup_{i=0}^{k-1}\left\{\underbrace{q^{i-3k}\mathcal{T}^3_{q^k}}_{\binom{i+2}{2}\text{ veces}},\ \underbrace{q^{i-2k}\mathcal{T}^3_{q^k}}_{k^2-\binom{i-k+2}{2}-\binom{i+2}{2}\text{ veces}},\ \underbrace{q^{i-k}\mathcal{T}^3_{q^k}}_{\binom{i-k+2}{2}\text{ veces}}\right\},$$

de donde, factorizando $q^{-3k}\mathcal{T}^3_{q^k}$, llegamos a

$$S^\star=\bigcup_{i=0}^{k-1}\left\{\underbrace{q^{i}}_{\binom{i+2}{2}\text{ veces}},\ \underbrace{q^{i+k}}_{k^2-\binom{i-k+2}{2}-\binom{i+2}{2}\text{ veces}},\ \underbrace{q^{i+2k}}_{\binom{i-k+2}{2}\text{ veces}}\right\}.$$

Observar que $k^2-\binom{i-k+2}{2}-\binom{i+2}{2}\geq 0$. Notando que

$$\binom{i+k+2}{2}-3\binom{i+k+2-k}{2}=k^2-\binom{i-k+2}{2}-\binom{i+2}{2}$$

y que

$$\binom{i+2k+2}{2}-3\binom{i+2k+2-k}{2}+3\binom{i+2k+2-k}{2}=\binom{i-k+2}{2},$$

vemos que esta distrbución es justamente la que queríamos. En efecto esto se sigue de expandir $(1-q^k)^3$ en la igualdad formal:

$$\sum_{0\leq a,b,c<k}q^{a+b+c}=\left(\frac{1-q^k}{1-q}\right)^3=(1-q^k)^3\times\sum_{i=0}^{\infty}\binom{i+2}{i}q^i\,.$$

Ejemplo 3.3.1 (Caso $k=2$)**.** Tenemos $S_{-1}=\bigcup_{i=0}^{1}\left\{\underbrace{q^{i-2}\mathcal{T}^1_{q^2}}_{\binom{i+2}{2}\text{ veces}},\ \underbrace{q^{i-2}\mathcal{T}^2_{q^2}}_{4+\binom{i+2}{2}\text{ veces}},\underbrace{q^{i-2}\mathcal{T}^3_{q^2}}_{4\text{ veces}}\right\}$ y aplicando

el algoritmo de Huffman obtenemos

$$\left\{ \underbrace{q^{-4}\mathcal{T}^3_{q^2}}_{4\text{ veces}}, \underbrace{q^{-3}\mathcal{T}^2_{q^2}}_{3\text{ veces}}, \underbrace{q^{-3}\mathcal{T}^3_{q^2}}_{4\text{ veces}}, \underbrace{q^{-4}\mathcal{T}^2_{q^k}}_{1\text{ veces}} \right\} .$$

Aplicamos el algoritmo Huffman una vez más, llegando a

$$S' = \left\{ \underbrace{q^{-5}\mathcal{T}^3_{q^2}}_{3\text{ veces}}, [q^{-3}\mathcal{T}^3_{q^2}, q^{-4}\mathcal{T}^3_{q^2}], q^{-6}\mathcal{T}^3_{q^2}, \underbrace{q^{-4}\mathcal{T}^3_{q^2}}_{2\text{ veces}} \right\} .$$

Ahora deshaciendo el merge que está entre paréntesis rectos obtenemos

$$S^\star = \left\{ \underbrace{q^{-5}\mathcal{T}^3_{q^2}}_{3\text{ veces}}, q^{-3}\mathcal{T}^3_{q^2}, q^{-6}\mathcal{T}^3_{q^2}, \underbrace{q^{-4}\mathcal{T}^3_{q^2}}_{3\text{ veces}} \right\}$$

de donde, finalmente, factorizando $q^{-3k}\mathcal{T}^3_{q^k}$ obtenemos la fuente $\hat{\mathcal{A}}_k = \{q^0, q^1, q^1, q^1, q^2, q^2, q^2, q^3\}$ que corresponde a los pesos de $\{(a,b,c) : 0 \le a,b,c < 2\}$. Por lo tanto, en este caso, se tiene un resultado análogo al bidimensional. $\Diamond$

Teorema 3.3.1. *Un código óptimo de prefijo $C^{(3)}_k$ para la fuente $(\mathcal{A}, \mathbf{p})$ con*

$$\mathcal{A} = \{(a,b,c) : 0 \le a,b,c\}, \quad w(a,b,c) = q^{a+b+c}\,, \tag{3.3.1}$$

donde $q = 2^{-1/k}$, $k \ge 1$ es

$$C^{(3)}_k(x,y,z) = T^{(3)}_k(x \bmod k, y \bmod k, z \bmod .k) \bowtie G_1\left(\left\lfloor \tfrac{x}{k} \right\rfloor\right) \bowtie G_1\left(\left\lfloor \tfrac{y}{k} \right\rfloor\right) \bowtie G_1\left(\left\lfloor \tfrac{z}{k} \right\rfloor\right)\,, \tag{3.3.2}$$

donde G_1 indica el código unario, $\bowtie$ la concatenación, y $T^{(3)}_k$ es cualquier código óptimo para una fuente finita con los siguientes símbolos y pesos

$$\hat{\mathcal{A}}_k = \{(a,b,c) : 0 \le a,b,c < k\}, \quad w(a,b,c) = q^{a+b+c}\,. \tag{3.3.3}$$

Parte II

Aspectos Prácticos

CAPÍTULO 4

Principios del Funcionamiento del Algoritmo

En este capítulo mostramos la utilidad de los códigos de Golomb en el contexto de la codificación sin pérdida de imágenes, mediante algoritmos de baja complejidad. Nos centramos en el algoritmo conocido como LOCO-I[a] [15, 16], mostrando cómo incorporar los códigos bidimensionales cuando tratamos de codificar imágenes a color, manteniendo en lo posible una complejidad comparable a la del original. Los resultados experimentales obteniendos se presentan, y discuten, luego en el capítulo 5.

4.1. Modelado probabilístico y Codificación

Comenzamos modelando las imágenes, o archivos más en general, como una realización $\mathbf{x} = (x_1, \ldots, x_n)$ de una variable aleatoria $\mathcal{X} = (X_1, \ldots, X_n)$, donde cada variable X_i toma valores en un cierto alfabeto finito $\mathcal{A}$.

Idealmente, quisiéramos codificar $\mathcal{X}$ utilizando aproximadamente $H(\mathcal{X})$ bits en media, o, recordando la regla de la cadena

$$H(\mathcal{X}) = \sum_{i=1}^{n} H(X_i|X_1, \ldots, X_{i-1}),$$

y, podríamos codificar x_i utilizando en promedio $H(X_i|X_1, \ldots, X_{i-1})$ bits aproximadamente.

Una posible aproximación a este problema consiste en estimar secuencialmente las distribuciones de probabilidad condicionadas $p(\cdot|x_1, \ldots, x_{i-1})$ para cada $i \in \{1, \ldots, n\}$ y codificar x_i con un código óptimo para dicha distribución. Sin embargo, ningún *contexto* condicionantes $x_1 \ldots x_{i-1}$ aparece dos veces, y resultaría imposible realizar esta estimación.

No obstante, en muchos casos es razonable suponer que la distribución de X_i depende únicamente de una cantidad finita k de las muestras previas, dando entonces un modelo de memoria finita

$$\Pr(X_{i+k} = x|X_1 = x_1, \ldots, X_{i+k-1} = x_{i+k-1}) = \Pr(X_{i+k} = x|X_i = x_i, \ldots, X_{i+k-1} = x_{i+k-1}),$$

[a] *LOw COmplexity LOssless COmpression for Images*.

para cada $i \in \{1, \ldots, n-k\}$. Si asumimos además que

$$\Pr(X_{i+k} = x | X_i = x_i, \ldots, X_{i+k-1} = x_{i+k-1}) = \Pr(X_{k+1} = x | X_1 = x_i, \ldots, X_k = x_{i+k-1}),$$

para cada $i \in \{1, \ldots, n-k\}$, tenemos que las probabilidades de transición no dependen del tiempo, y podemos tomar estadísticas por cada estado $x_i \ldots x_{i+k-1}$ y aproximar las probabilidades condicionales.

Finalmente, más en general, consideramos una función $f : \mathcal{A}^* \to Z$ tal que $f(x_1 \ldots x_i) \in Z$ sea suficiente para determinar $p(\cdot|x_1, \ldots, x_i)$ y que $|Z|$ sea lo suficientemente pequeño de modo que la cantidad de veces que se repite cada valor $f(x_1 \ldots x_i)$ con $i \in \{1, \ldots, n\}$ permita en general estimar las probabilidades condicionales. Para ejemplificar, en el caso de una imagen podríamos elegir f de manera que nos dé únicamente información sobre el valor de los píxeles más cercanos geométricamente al punto actual. A los elementos de Z se los denomina *contextos* o *clases condicionantes*.

La codificación secuencial se puede separar en dos partes: el modelador y el codificador. En cada instante i el modelador realiza una estimación de $p(\cdot|x_1 \ldots x_{i-1})$ basada en los datos previos $x_1 \ldots x_{i-1}$. Esta estimación de la distribución es luego utilizada por el codificador para realizar la codificación como considere adecuado. Esta misma información también la tiene disponible el decodificador, ya que la compresión se asume sin pérdida y, de esta manera, el decodificador puede des-hacer los pasos realizados por el codificador y obtener x_i.

La separación entre modelado y codificación puede realizarse mediante la denominada *codificación aritmética* [4], que permite trabajar con la asignación de probabilidad $\hat{p}(x_i|x_1, \ldots, x_{i-1})$ dictada por el modelo, de forma que la longitud de código total para la secuencia completa $\mathbf{x} = (x_1, \ldots, x_n)$ sea aproximadamente $-\lg(\hat{p}(\mathbf{x}))$ bits, donde $\hat{p}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \hat{p}(x_i|x_1, \ldots, x_{i-1})$.

No obstante, la codificación aritmética resulta relativamente compleja. El algoritmo LOCO-I, mediante la elección de un modelo particular para $p(\cdot|f(x_1 \ldots x_{i-1}))$ nos permite reducir significativamente esta complejidad, tanto del modelado como de la codificación, a costo de un modesto incremento en longitud [16].

Finalmente, es importante mencionar que el número de parámetros libres K de un modelo probabilístico paramétrico, de ser muy grande, puede llegar a resultar contraproducente en cuanto a longitud de código. Esta limitación fundamental es analizada de forma precisa por Rissanen en [19, Theorem 1], demostrando un largo de código medio extra por encima de $H(\mathcal{X})$ de al menos $(K \lg n)/2$ bits, para todos los parámetros, a menos de un conjunto cuyo volumen tiende a 0 con n, de un modelo paramétrico con un espacio de parámetros compacto. Este costo se puede interpretar como una manifestación de la denominada 'dilución de contextos' que ocurre cuando los datos estadísticos limitados deben ser divididos entre una excesiva cantidad de contextos [16].

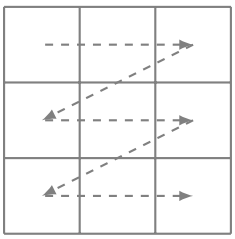


Figura 4.2.1: Orden raster-scan.

4.2. El algoritmo LOCO-I para la codificación de imágenes de tono de gris

Imágenes digitales

Una *imagen* es un conjunto de matrices bidimensionales de enteros. A cada uno de dichos enteros se los conoce como *muestra*, y a cada array se lo denomina *componente*. A las entradas de correspondientes a una posición (i, j) dada las denominamos conjuntamente *píxel* (i, j).

En el caso de las imágenes de tono de gris, hay única componente que corresponde a la intensidad de luz. Por el contrario, en el caso de las imágenes a color tenemos varios componentes, por ejemplo, en el caso del espacio de colores RGB tenemos tres componentes, uno para la intensidad del rojo R, uno para la intensidad del verde G, y uno para la intesidad del azul B.

Estas intensidades lumínicas, que, teóricamente serían contínuas, deben ser discretizadas para poderlas guardar en memoria. En la práctica, sin embargo, asumiremos que cada muestra tiene un número de bits β considerable, típicamente 8 bits, como es el caso en las imágenes utilizadas para las pruebas. Escribimos una imagen de tono de gris como $\mathbf{x} = (x_1, \ldots, x_n)$, donde los x_i indican las intensidades de cada píxel, en algún orden preestablecido dado. Por ejemplo, si representamos la imagen como $(a_{i,j})_{(i,j)\in[l]\times[m]}$, podemos considerar el orden $x_1 = a_{1,1}, x_2 = a_{1,2}, \ldots, x_m = a_{1,m}, x_{m+1} = a_{2,1}, \ldots$. A este orden, que se encuentra ilustrado en la figura 4.2.1, se le denomina *raster-scan*, y es el que asumiremos en lo sucesivo.

LOCO-I es un algoritmo de compresión sin péridad de imágenes de tono de gris, desarrollado por investigadores en los laboratorios Hewlett-Packard, y luego estandarizado a JPEG-LS [16]. Su énfasis se encuentra en mantener una baja complejidad, manteniendo el alto nivel en cuanto al radio de compresión, siendo comparables con los resultados obtenidos por algoritmos más complejos.

Para guiarnos en la descripción de LOCO-I, notamos que este último sigue los siguientes elementos para la asignación de probabilidad, que son usuales en los esquemas de compresión sin pérdida. [16]

1. Se estima el valor de x_i según los bits ya codificados $x_1, \ldots, x_{i-1}$. Sea $\hat{x}_i$ dicha estimación.
2. Se calcula $f(x_1 \ldots x_{i-1})$, el contexto en el que ocurre x_i. Aquí f corresponde a la función descrita en la sección anterior.
3. Un modelo probabilístico paramétrico para el residuo de predicción (o señal de error) $\epsilon_i \triangleq x_i - \hat{x}_i$, condicionado al contexto del item anterior.

LOCO-I utiliza, para el item 1, un predictor relativamente simple, con alguna capacidad de detección de bordes denominado MED.

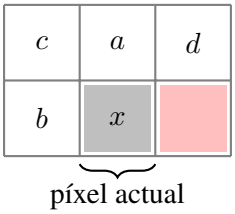


Figura 4.2.2: Contexto causal utilizado para el predictor y el modelado de contexto en LOCO-I. Los pixeles a, b, c y d se utilizan para predecir y para determinar el contexto de codificación del píxel x.

4.2.1. Predictor

En pos de mantener la simplicidad, LOCO-I [16, sección 2.1] considera un predictor fijo, basado en el contexto causal que se muestra en la figura 4.2.2.

El enfoque consiste en determinar primero, de forma sencilla, si existe un borde horizontal o vertical. De no detectarse una borde, se procede a estimar la intensidad del píxel actual x asumiendo una cierta 'suavidad' en las intensidades.

Concretamente el predictor se define de la siguiente manera

$$\hat{x} \triangleq \begin{cases} \text{mín}\{a, b\}, & \text{si } c \geq \text{máx}\{a, b\}\,, \\ \text{máx}\{a, b\}, & \text{si } c \leq \text{mín}\{a, b\}\,, \\ a + b - c, & \text{en otro caso}\,. \end{cases} \tag{4.2.1}$$

Este predictor se conoce como 'median edge detector' (MED), ya que $\hat{x}$ corresponde exactamente a la mediana de a, b y $a + b - c$.

Observar que por simetría de $\hat{x}$ respecto de a y b, podemos asumir $a \leq b$. Dicha simetría corresponde a una simetría del cuadrado que contiene a a, b, c y x en la figura 4.2.2. En caso de que $c \geq b$ el predictor selecciona a. La interpretación de esto es que, si $c \geq b$, es razonable pensar que debe corresponder bastante seguido a un borde vertical a la izquierda del píxel x. De manera similar, se elige b cuando $c \leq a$ y podemos pensar que esto corresponde a un borde horizontal por encima del píxel actual.

Finalmente, el caso restante es el caso 'suave', y se puede interpretar como una interpolación de orden 2, como mostramos a continuación. Imaginemos la imagen digital como una grilla de puntos muestras de una imagen continua, g, tomadas regularmente a distancia $h > 0$ entre sí. Suponemos entonces que, si el píxel actual x corresponde a la posición (i, j), el píxel a corresponde a $(i, j - h)$, b a $(i - h, j)$ y c a $(i - h, j - h)$. Notamos entonces que

$$a + b - c = g(i, j - h) + g(i - h, j) - g(i - h, j - h) = g(i, j) + O(h^2)\,,$$

si g es dos veces diferenciable (con derivada segunda continua).

Obsérvese que el predictor (4.2.1) no utiliza el píxel d de la figura 4.2.2. Este sí se utiliza, sin embargo, en el modelado de contexto.

4.2.2. Modelado de contextos

Distribución de los residuos

Es una observación ampliamente aceptada [25] que, en una imagen continua, el residuo de predicción $\epsilon_i = x_i - \hat{x}_i$, sujeto al contexto, se ajusta bien a una *distribución geométrica a dos lados* (*two-sided geometric distribution TSGD*). Esta distribución se define como sigue

Definición 4.2.1 (*TSGD*)**.** Decimos que una variable aleatoria X, que toma valores en $\mathbb{Z}$, tiene distribución $TSGD(\theta, \mu)$ para ciertos $\theta \in (0, 1)$ y $\mu \in \mathbb{R}$ si y solo si

$$\Pr(X = k) = C \times \theta^{|k+\mu|}, \tag{4.2.2}$$

donde $C > 0$ es la constante de normalización.

El parámetro μ es denominado el *offset* de la distribución, ya que mide cuánto hay que trasladar el eje Ox para que la distribución esté centrada en 0.

Esta distribución está fuertemente relacionada con la distribución geométrica como mostramos más adelante, lo cual permite utilizar los códigos de Golomb que vimos en el capítulo 3, para tener rutinas muy sencillas de codificación y decodificación. Observar también que el número de parámetros por contexto es bastante pequeño, ya que tenemos dos parámetros libres por contexto. Es por esto que este modelo resulta atractivo para la codificación de baja complejidad.

Estrictamente hablando, como en la práctica tenemos un alfabeto finito con $\alpha = 2^\beta$ elementos, los residuos de predicción ϵ solo pueden tomar valores en el rango $-(\alpha - 1) \leq \epsilon \leq (\alpha - 1)$. Más aún, dado el valor de $\hat{x}$, que tanto el codificador como el decodificador conocen, el residuo ϵ puede tomar únicamente alguno de los α valores del conjunto $\{x - \hat{x} : 0 \leq x \leq \alpha\}$. Con esto, podemos reducir el residuo ϵ al rango entre $-\alpha/2$ y $\alpha/2$ tomando módulo α.

Si pensamos en $\epsilon \in \{-(\alpha - 1), \ldots, 0, \ldots, (\alpha - 1)\}$ como bien ajustado a una geométrica centrada en 0 (ajustando primero ϵ con un término adaptativo de la predicción, que intenta eliminar la parte entera del 'offset'), la operación de reducir ϵ en módulo α al conjunto $\{-\alpha/2, \ldots, 0, \ldots, \alpha/2 - 1\}$ corresponde a unificar las colas $\{\alpha/2, \ldots, \alpha - 1\}$ y $\{-(\alpha - 1), \ldots, -\alpha/2 - 1\}$ con las partes centrales $\{-\alpha/2, \ldots, 0\}$ y $\{0, \ldots, \alpha/2 - 1\}$ respectivamente. De cualquier modo, el efecto de unificar las colas es pequeño por lo general, por lo cual consideramos en general que ϵ reducido módulo α se sigue comportando como una distribuciíon TSGD.

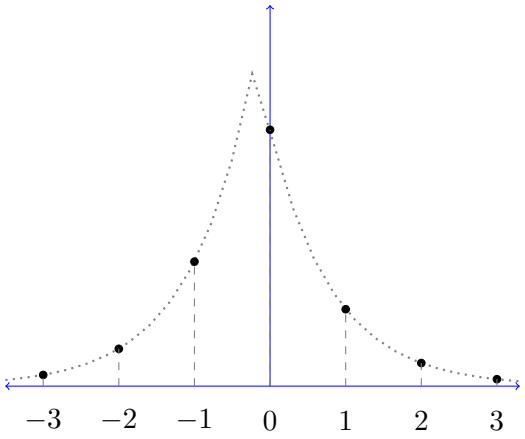


Figura 4.2.3: Distribución $TSGD(\theta, \mu)$.

Determinación de los contextos

Resta todavía definir la función f que asigna al píxel actual x_i su contexto, de acuerdo a la secuencia de píxeles pasados $x_1 \dots x_{i-1}$.

Es importante elegir f de modo que se asocie razonablemente bien al comportamiento estadístico de los errores de predicción. En este sentido, el algoritmo LOCO-I considera los gradientes en el área del píxel actual. Dada la situación de la figura 4.2.2, consideramos entonces $g_1 = d-b$, $g_2 = b-c$ y $g_3 = c-a$. Observar que esto nos permite tratar uniformemente situaciones en las que tenemos simplemente una traslación de las intensidades, sumando una constante. En efecto, esto es posible porque esperamos que el error de predicción para estos casos sea similar, y esté casi centrado en 0. Hasta el momento, tenemos $(2\alpha-1)^3$ posibles tríos, o contextos, posibles. Sin embargo, $(2\alpha-1)^3$ puede llegar a ser excesivamente grande para ser práctico. Por ejemplo, para $\alpha = 256$ tendríamos $(2\alpha-1)^3$ del orden de 1.3×10^8, y es posible que en una imagen, de 1000×1000 digamos, no encontremos dos contextos iguales. En efecto, una pequeña variación en cualquiera de los gradientes distingue los contextos, si bien estamos suponiendo que las imágenes tienen una cierta 'continuidad', que esperamos de las imágenes naturales.

Para explotar la continuidad de las imágenes naturales, cada uno de los gradientes g_1, g_2, g_3 es cuantizado a un valor entero que identificamos con una 'región' de una partición dada de $\{-(\alpha-1), \dots, \alpha-1\}$, cuyas partes corresponden a intervalos. En principio estas regiones se deben aprender adaptativamente, pero, para mantener la simplicidad, LOCO-I opta por mantenerlas fijas. Por simetría, se toma una región centrada en 0, y luego, por cada región $[r_1, r_2]$ tenemos otra $[-r_2, -r_1]$. Escribimos entonces $2R+1$ para el número de regiones, donde $R \in \mathbb{Z}_{\geq 0}$, y tenemos $(2R+1)^3$ contextos posibles. A las distintas regiones les asignamos números enteros de modo que la función $g_i \mapsto q_i$ resultante sea impar.

Denotamos los contextos mediante una 3-upla (q_1, q_2, q_3), donde q_i indica el resultado de cuantizar el gradiente g_i. Considerando que, por simetría, es razonable asumir que

$$\Pr(\epsilon_i = \Delta | q_1 = r, q_2 = s, q_3 = t) \approx \Pr(\epsilon_i = -\Delta | q_1 = -r, q_2 = -s, q_3 = -t) , \tag{4.2.3}$$

podemos reducir aún más el número de contextos a $\left((2R+1)^3 + 1\right)/2$, manteniendo juntas las estadísticas asociadas a los contextos (q_1, q_2, q_3) y $(-q_1, -q_2, -q_3)$, teniendo cuidado con el cambio de signo.

El estándar JPEG-LS [16] utiliza $R = 4$ y las regiones

$$\{0\}, \pm\{1,2\}, \pm\{3,4,5,6\}, \pm\{7,8,\dots,20\}, \pm\{e : e \geq 21\} .$$

Sin embargo, como es explicado en [16], esto depende del tamaño de las imágenes con las que estemos tratando. En nuestro caso, estas regiones tendrán cambios cuando tratemos con los canales $R-G$ y $B-G$ imágenes a color, que se estudian para nuestro juego concreto de imágenes.

Cancelación del sesgo

Las predicciones, sujetas al contexto, muestran por lo general un offset $\mu \in \mathbb{R}$, que resulta de que el predictor es fijo. A continuación mostramos cómo corregir este offset mediante un entero, dando lugar a una distribución corregida que se encuentra relativamente centrada en 0, que mostraremos luego cómo codificar.

Escribimos $\mu = R - s$ donde $R \in \mathbb{Z}$ y $s \in [0, 1)$, dicho de otra forma, $R = \lceil \mu \rceil$. Observamos entonces que podemos escribir la distribución de los residuos de predicción como

$$P_{(\theta,s)}(\epsilon) = C(\theta, s)\, \theta^{|\epsilon - R + s|} ,$$

donde $C(\theta, s) = \dfrac{1-\theta}{\theta^{1-s}+\theta^{s}}$ es la constante de normalización.

Una forma sencilla de estimar R en un determinado contexto, es considerar la suma D de los errores de predicción, el número N de ocurrencias del contexto actua y calclar la estimación como $\hat{R} = \lceil D/N \rceil$. Sin embargo, esta estimación es bastante susceptible a la aparición de 'outliers', es decir, residuos que resulten bastante mayores de lo usual, que puedan afectar el resultado en el futuro por un tiempo considerable.

El estimador usado en LOCO-I evita este problema y, adicionalmente, su implementación es muy eficiente. Comenzamos escribiendo

$$D = N \cdot C' + B' \,,$$

donde B' satisface $-N < B' \leq 0$. Es posible implementar esto manteniendo B' y C' y ajustándolos en cada ocurrencia del contexto, sin embargo, tenemos el problema de los 'outliers' antes mencionado. La solución adoptada, utiliza dos registros C y B, que aproximan C' y B' respectivamente, de forma que C no cambia en más de una unidad por cada ocurrencia del contexto, de modo de evitar un impacto demasiado fuerte de eventuales outliers. La actualización de estos registros se realiza según el pseudocódigo presentado en el Algoritmo 4.2.1 a continuación.

Algoritmo 4.2.1 Rutina de la cancelación del sesgo.

```
procedure BIASCOMPUTATION(ε)
    B = B + ε
    N = N + 1
    if B ≤ −N then
        C = C − 1
        B = B + N
        if B ≤ −N then
            B = −N + 1
        end if
    else if B > 0 then
        C = C + 1
        B = B − N
        if B > 0 then
            B = 0
        end if
    end if
end procedure
```

Observar que $-B/N$ resulta una estimación de s, y C una aproximación de R.

4.2.3. Codificación

En este punto, una vez que restamos la aproximación estática $\hat{x}$ dada por (4.2.1) y la aproximación adaptativa que aproxima el techo del 'offset' μ de la distribución TSGD, tenemos el residuo ϵ que se modela con una TSGD con $\theta \in (0,1)$ y un 'offset' d que está en el intervalo $[0,1)$.

Observación 4.2.1. Dada una variable aleatoria X con distribución TSGD con parḿetros $\theta \in (0,1)$ y $d \in [0,1)$, tenemos

$$H(X) = \frac{h(\theta)}{1-\theta} + h(\rho)\,, \tag{4.2.4}$$

donde $\rho \triangleq \dfrac{\theta^d}{\theta^{1-d}+\theta^d}$. En particular, X tiene entropía finita y por lo tanto, como consecuencia del teorema 2.7.2, sabemos que existen códigos de prefijo óptimos para codificarla.

Una forma directa de ver la relación con las variables geométricas es la siguiente.

Observación 4.2.2. Sea X una variable aleatoria con distribución $TSGD(\theta, d)$ con $d \in [0,1]$, entonces

$$\Pr(X \in \{n, -n-1\}) = (1-\theta)\,\theta^n\,, \tag{4.2.5}$$

independientemente de d. Esto se sigue directamente de hacer el cálculo con la definición de TSGD. En principio esto sugiere codificar una realización ϵ de la siguiente manera

1. Encontrar $n \geq 0$ tal que $\epsilon \in \{n, -n-1\}$. Codificar n utilizando el código de Golomb óptimo para el parámetro θ.
2. Adjuntar un bit al final para indicar si se trata de n o $-n-1$.

De hecho, observar que si N es la variable aleatoria que representa el entero no-negativo tal que $X \in \{N, -N-1\}$ y S es la variable aleatoria que indica si X es no-negativo o negativo, entonces especificar X es equivalente a especificar (N,S). Por la regla de la cadena

$$H(X) = H(N) + H(S|N)\,.$$

Como N es una variable aleatoria con distribución geométrica de parámetro θ, tenemos inmediatamente que

$$H(N) = \frac{h(\theta)}{1-\theta}\,,$$

mientras que S, dado N, es una variable aleatoria que toma dos valores, con probabilidades ρ y $1-\rho$ respectivamente, se sigue entonces que

$$H(S|N) = h(\rho)\,,$$

dando entonces una intepretación directa de (4.2.4).

Sin embargo, este procedimiento para codificar X es subóptimo. En otras palabras, se puede codificar (N,S) conjuntamente, de una manera más inteligente. Observar que, al adjuntar el bit al final, si $d \neq 1/2$ tenemos que $\rho \neq 1/2$, y por lo tanto el paso 2. ya tiene una cierta redundancia.

Los códigos de prefijo óptimos para las distribuciones TSGD fueron caracterizados en [18]. La codificación involucra 4 regiones en el espacio de pares de parámetros $(\theta, d) \in [0,1] \times [0, 1/2]$, cada una con una codificación distinta. Consideramos $0 \leq d \leq 1/2$ porque, de otro modo, con una simetría $x \mapsto -1-x$ de la recta real, logramos tener $0 \leq d \leq 1/2$.

Observar que bajo esta hipótesis, la secuencia $0, -1, +1, -2, +2, \ldots$ está ordenada en orden decreciente de probabilidad. Una idea popular [23] es entonces suponer que el índice de dicha secuencia se

aproxima bien a una variable geométrica, y utilizar un código de Golomb unidimensional. Esta idea se encuentra ilustrada en la figura 4.2.4, donde el mapa $M(n)$ se define mediante $M(n) = 2n - \mathbf{1}_{<0}(n)$, que mapea los elementos de $\mathbb{Z}$ en el índice de la secuencia ordenada por probabilidad. De hecho, esta codificación corresponde a la de una de las 4 regiones caracterizadas en [18] para los códigos óptimos.

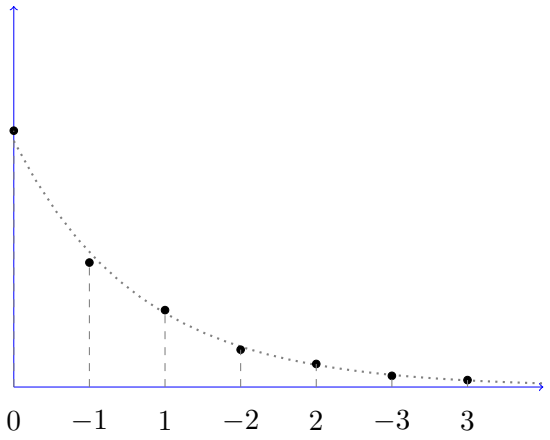


Figura 4.2.4: Aplicando el mapa $x \mapsto M(x)$ a la distribución TSGD de parámetros (θ, d), contra su aproximación $(1 - \sqrt{\theta}) \times \theta^{x/2}$.

Más concretamente, las siguientes observaciones muestran la relación entre $M(X)$ y una variable de distribución geométrica.

Observación 4.2.3. Si una variable X tiene distribución $TSGD(\theta, d)$ con $d \in [0, 1/2]$, entonces

$$\Pr(M(X) = n) = C(\theta, d) \times \theta^{|\mu(n)+d|},$$

donde $\mu(n) \triangleq (-1)^n \left\lceil \frac{n}{2} \right\rceil$ es la inversa del mapa M y $C(\theta, d) = \dfrac{1-\theta}{\theta^d + \theta^{1-d}}$. Como $|\mu(n) + d| \approx n/2$ a medida que crece n, aproximar $M(X)$ como una variable aleatoria geométrica de parámetro $\theta^{1/2}$ es una aproximación razonable (ver figura 4.2.4).

Observación 4.2.4. De hecho, si $d = 1/4$ tenemos que una variable aleatoria con distribución $TSGD(\theta, d)$ se transforma en una variable geométrica de parámetro exactamente $\theta^{1/2}$ al aplicarle M.

Demostración. En efecto, tenemos

$$\Pr\left(M(X) = 2n\right) = C(\theta, d) \times \theta^{n+1/4}, \quad \Pr\left(M(X) = 2n+1\right) = C(\theta, d) \times \theta^{n+1-1/4}.$$

Como

$$n + 1/4 = (2n)/2 + 1/4, \quad n + 1 - 1/4 = (2n+1)/2 + 1/4,$$

se sigue que

$$\Pr\left(M(X) = n\right) = C(\theta, d)\, \theta^{1/4} \times \theta^{n/2},$$

para todo $n \in \mathbb{Z}_{\geq 0}$. ◊

Volviendo a LOCO-I, como los códigos óptimos resultan significativamente más complejos, en la práctica LOCO-I utiliza la subfamilia de los códigos GPO2 (Golomb-Power-Of-2), también conocidos como códigos de Rice [14], considerando entonces utilizar la subfamilia

$$\mathcal{G} = \left\{G_{2^k}\left(M(\cdot)\right) : k \geq 0\right\} \cup \left\{G_1\left(M'(\cdot)\right)\right\}, \tag{4.2.6}$$

donde $M'(x) \triangleq M(-1-x)$. Observar que no consideramos $G_{2^k}\left(M'(\cdot)\right)$ para $k \geq 1$ porque este asigna las mismas longitudes a x que a $-(x+1)$, y así resulta tener la misma longitud promedio que $G_{2^k}\left(M(\cdot)\right)$. Esto último nos indica que aplicar el mapa $x \mapsto 1-x$ cuando $1/2 < d \leq 1$, solo repercute en la codificación de Rice posterior si se va a aplicar G_1. Por lo tanto, la familia $\mathcal{G}$ involucra todos los códigos de Rice que podemos necesitar utilizar, incluso cuando $d \in (1/2, 1)$.

La selección de los códigos de $\mathcal{G}$ óptimos para un par $(\theta, d) \in (0,1) \times (0,1)$ dado se realiza a partir de

$$S \triangleq \frac{\theta}{1-\theta}, \quad \rho \triangleq \frac{\theta^d}{\theta^{1-d} + \theta^d}, \tag{4.2.7}$$

mediante el siguiente lema que citamos de [18, Lemma 4]

Lema 4.2.1. *Consideramos una variable aleatoria discreta X, que toma valores en $\mathbb{Z}$ con una distribución TSGD con parámetros (θ, d) con $\theta \in (0,1)$, $d \in [0,1]$. Sean S y ρ definidos por (4.2.7), y $\phi \triangleq \frac{1+\sqrt{5}}{2}$. Las siguientes reglas de selección de código minimizan la longitud esperada sobre la familia $\mathcal{G}$.*

a) Si $S \leq \phi$, comparar S, ρ y $1-\rho$. Si S es el mayor, elegir $G_2(M(\cdot))$. En otro caso, si ρ es el mayor, elegir $G_1(M(\cdot))$, y en otro caso elegir $G_1(M'(\cdot))$.

b) Si $S > \phi$, elegir $G_{2^{r+1}}\left(M(\cdot)\right)$, con $r \geq 1$ que cumple

$$\frac{1}{\phi^{2^{-r+1}} - 1} < S \leq \frac{1}{\phi^{2^{-r}} - 1}\,.$$

Observar la relación de este lema con la observación 3.1.5. En efecto, intuitivamente, si $d \in [0, 1/2]$ y suponemos que $M(X)$ se comporta de manera similar a una distribución geométrica de parámetro $\sqrt{\theta}$, entonces deberíamos elegir G_{2^r} con $r \geq 0$ el menor entero no-negativo tal que tal que $\left(\sqrt{\theta}\right)^{2^r} \leq \phi^{-1}$. En otras palabras

$$\theta^{2^{r-1}} \leq \phi^{-1} < \theta^{2^{r-2}}\,.$$

Observamos ahora que $\dfrac{1}{\phi^{2^{-r+1}} - 1} = \dfrac{\phi^{-2^{-r+1}}}{1 - \phi^{-2^{-r+1}}}$, y que $x \mapsto \dfrac{x}{1-x}$ es creciente, por lo tanto la desigualdad del item b) es equivalente a $\phi^{-2^{-r+1}} < \theta \leq \phi^{-2^{-r}}$, esto es

$$\theta^{2^r} \leq \phi^{-1} < \theta^{2^{r-1}}\,,$$

y por lo tanto deberíamos utilizar G_{r+1}, tal como asegura el lema 4.2.1.

En la práctica utilizamos estimaciones de S y ρ. Para esto, llevamos contadores por cada contexto de

1. El número de veces que ha aparecido el contexto hasta ahora t.
2. La suma
$$S_t \triangleq \sum_{i=1}^{t} \left(|\epsilon_i| - \mathbf{1}_{<0}(\epsilon_i)\right),$$
donde $\epsilon_1, \ldots, \epsilon_t$ indican los residuos de predicción correspondientes a nuestro contexto.
3. La suma
$$N_t \triangleq \sum_{i=1}^{t} \mathbf{1}_{<0}(\epsilon_i)\,,$$
que utilizamos para estimar $\mathbb{E}[\mathbf{1}_{<0}(\epsilon)]$ mediante $\frac{1}{t}\, N_t$.

Las estimadores son entonces $\hat{S}_t = S_t/t$ y $\hat{\rho}_t = 1 - N_t/t$. Notar en efecto que

$$S = \mathbb{E}[|X| - \mathbf{1}_{<0}(X)], \quad \rho = \mathbb{E}[1 - \mathbf{1}_{<0}(X)]\,. \tag{4.2.8}$$

Es observado también en [18, Theorem 2, Corollary 1] que en el peor caso la pérdida relativa por utiliar la subfamilia $\mathcal{G}$ no excede el $4.7\,\%$ con respecto a la longitud media ideal, y en el límite $\theta \to 1$ tiende a 0.

El algoritmo LOCO-I, sin embargo, en su énfasis por la simplicidad, no utiliza la regla del lema 4.2.1 tal cual descrita. Para esto observamos [16] que

$$S(k) \triangleq \frac{1}{\phi^{2^{-k+1}} - 1} = \frac{2^{k-1}}{\log \phi} - \frac{1}{2} + o(1)\,, \tag{4.2.9}$$

y que el resto que tiende a 0 es en realidad una función decreciente en k, con un máximo de $\phi + 1/2 - 1/\log(\phi) \approx 0.04$ en $k = 1$. Aquí $\dfrac{1}{\log \phi} \approx 2.078$, y así podemos considerar la aproximación $S(k) \approx 2^k - 1/2 + 1/8$, sin grandes pérdidas.

Es así que, si definimos $S'_t \triangleq S_t + (t/2) - (t/8)$, podemos considerar la siguiente simplicación de la regla del lema 4.2.1 para codificar ϵ_{t+1}.

1. Si $S'_t \leq 2t$, comparar S_t, N_t y $t - N_t$. Si S_t es el mayor, elegir $G_2(M(\cdot))$. Si $t - N_t$ es el mayor, elegir $G_1(M(\cdot))$, y en otro caso $G_1(M'(\cdot))$.
2. Si $S'_t > 2t$, elegir $G_{r+1}(M(\cdot))$ con $r \geq 1$ tal que $t2^r \leq S'_t < t2^{r+1}$.

Observar que $S'_t \leq 2t$ si y solo si $S_t \leq (2 - 1/2 + 1/8)t$ y que $2 - 1/2 + 1/8 = 1.625 \approx \phi$. Por otro lado, tenemos que $\frac{1}{t}S_t \approx S$ y que $\frac{1}{t}(t - N_t) \approx \rho$ y $\frac{1}{t}N_t \approx 1 - \rho$, de acuerdo a nuestros estimadores.

Simplificación Finalmente, la forma estandarizada de LOCO-I en JPEG-LS simplifica esta regla aún más notando que es razonable esperar que $N_t \approx t/2 - t/8$, y luego aproximamos $S_t + N_t \approx S'_t$. La cantidad $S_t + N_t$ se guarda en un registro A. Seguidamente, calculamos $k = \min\{r \geq 0 : 2^r N \geq A\}$ donde N es el registro que guarda el número t de ocurrencias previas.

Si $k > 1$, estamos en el caso del item 2, dada nuestra estimación $A \approx S'_t$. En dicho caso codificamos ϵ_{t+1} según $G_k(M(\cdot))$.

Por otro lado, si $k = 1$, esto implica que $N < A$. Ahora, como estamos suponiendo $N_t \approx t/2 = A/2$, esto nos indica que $t < S_t + t/2$, esto es $t/2 < S_t$. Como $N_t \approx t/2$ y $t - N_t \approx t/2$, la regla del item 1 indica que debemos utilizar $G_2(M(\cdot))$. Finalmente, si $k = 0$, observamos que $-B/N < 1/2$ es una aproximación de $\Pr(\epsilon < 0) < 1/2$. Así $2B > -N$ es una aproximación de la comparación $N_t < t - N_t$ (en su forma $2N_t < t$) muy razonable. Así, seleccionamos $G_1(M(\cdot))$ cuando $2B > -N$, y $G_1(M'(\cdot))$ en otro caso.

El parámetro k se halla entonces mediante la siguiente línea de código [16] en el lenguaje C:

```
for (k = 0; (N << k) < A; k++);
```

4.2.4. Otros detalles

Resets

Si bien es razonable pensar que localmente las estadísticas por contexto varían levemente, en una imagen relativamente grande pueden llegar a ocurrir diferencias importantes entre las estadísticas de las distintas regiones. Es por este motivo que cuando el número de estadísticas obtenidos alcanza cierto umbral (por defecto 64), dividiemos las variables N, A y B entre 2, de forma de mantener las estimaciones, pero restándole peso al pasado.

Run Mode

Codificar símbolo por símbolo tiene la limitación de que debemos producir al menos un bit por cada píxel. En caso de que la región sea extremadamente suave, y la entropía sea significativamente menor que 1 bit/entero, esta limitación se torna muy importante. Usualmente este clase de inconvenientes se resuelve utilizando codificación en bloque. LOCO-I/JPEG-LS utiliza una idea similar, cuando el contexto $[q_1, q_2, q_3]$ actual es $[0, 0, 0]$, el codificador entra en *run-mode*. Una vez en run-mode, el codificador codifica, a groso modo, el número de muestras iguales a la actual x que hay, hasta que difieren (en cuyo caso se codifica la diferencia $\varepsilon = x - b$) o hay un final de línea, saliendo entonces del *run-mode*.

4.3. Imágenes a color

En esta sección explicamos las diferencias con respecto a LOCO-I que utilizamos para codificar las imágenes a color, y cómo aparecen los códigos para distribuciones geométricas bidimensionales en dicho contexto. Presentamos también distintas variantes posibles, y una regla simplificada para la codificación de los códigos bidimensionales, explicando sus fundamentos. Posteriormente, en el capítulo siguiente se presentan los resultados experimentales, comparando con los resultados correspondientes para una implementación de LOCO-I.

Recordamos que las imágenes a color son imágenes con más de un componente. En nuestro caso nos concentraremos en el espacio de color RGB, es decir, tres componentes, uno para indicar la intensidad del rojo R, uno para el verde G, y uno para el azul B.

4.3.1. Decorrelación

Una primera idea para codificar la secuencia $((R_t, G_t, B_t))_{t=1}^n$ en orden, puede ser juntar las estadísticas de los componentes, ya que estos suelen tener comportamientos similares. Si bien esto trae ventajas a la hora de tener más muestras para cada contexto, no se obtiene una mejora tan grande como cuando consideramos 'decorrelacionar' los componentes mediante la transformación reversible $(R, G, B) \mapsto (R - G, G, B - G)$.

Es razonable suponer aquí que los compontentes $R-G$ y $B-G$ tienen errores residuales de predicción independientes, y que por lo general también sus parámetros resultan similares. Es aquí donde esperamos utilizar los códigos bidimensionales. Así pues, el componente G se codifica mediante LOCO-I, mientras que $R - G$ y $B - G$ se codifican mediante los códigos bidimensionales, de tener parámetros similares, y con LOCO-I en otro caso. Esto es explicado más adelante en este capítulo.

Finalmente, si bien $R - G$ y $B - G$ se puede pensar son mucho más 'suaves' que G, no es cierto que estén del todo decorrelacionados. Por ejemplo, suelen compartir bastantes de los bordes.

4.3.2. Codificación

El algoritmo LOCO-I hace uso de la familia $\mathcal{G}$ por sus ventajas a la hora de codificar y por la simplicidad de las reglas de adaptación resultantes. En lo que respecta a codificación, esto se da porque las operaciones de resto y cociente entre una potencia de dos se pueden realizar muy eficientemente con operaciones lógicas elementales.

En [17, Section VI] se menciona que los códigos con parámetro $k = 2^r$ deberían tener ventajas similares en el caso bidimensional, pero no proceden a desarrollar el tema ya que lo consideran similar al de [18, Section IV].

Comenzamos entonces motrando una regla simplificada para codificar pares de variables aleatorias geométricas iid, y luego mostramos cómo usar esto cuando tenemos pares de variables aleatorias iid con distribución TSGD.

Regla simplificada para los códigos bidimensionales

Dado (X, Y) con distribución $TDGD(q)$, sabemos que C_{2^k} definido en (3.2.2) es óptimo si $q = 2^{-1/2^k}$. Observar que para cada k, no tiene porque haber un único perfil de T_{2^k}, así, la definición de

C_{2^k} depende en realidad de una elección arbitraria de perfil para el 'topcode' T_{2^k}.

Notar que $\Delta_k(q) = L_q(C_{2^{k+1}}) - L_q(C_{2^k})$ satisface $\Delta_k\left(2^{-1/2^k}\right) > 0$ y $\Delta_k\left(2^{-1/2^{k+1}}\right) < 0$ por el teorema 3.2.1. Existe por lo tanto al menos un cero para $\Delta_k(q)$ en el intervalo $I_k \triangleq (2^{-1/2^k}, 2^{-1/2^{k+1}})$, ya que $\Delta_k(q)$ es continua.

Es natural pensar que en general $\Delta_k(q)$ debe ser decreciente, y por lo tanto tener un único 0 en todo $(0,1)$. Sin embargo, la prueba de esto se ve complicada por la complejidad de las fórmulas correspondientes.

Nos contentaremos entonces con caracterizar un punto $r_k \in I_k$ tal que $\Delta_k(r_k) \approx 0$, y lo tomaremos como punto de transición, o corte entre C_{2^k} y $C_{2^{k+1}}$. Es decir, para los puntos de I_k menores que r_k utilizamos C_{2^k} para codificar, y a su derecha utilizamos $C_{2^{k+1}}$.

En el caso unidimensional, los códigos de Rice tienen puntos de transición de la forma $z^{1/2^k}$ con $z = \phi^{-1}$, como se explica en la observación 3.1.5. Es razonable preguntarnos entonces si tenemos una situación similar en este caso, posiblemente de forma aproximada, para un cierto z. Observar también que parte del atractivo de esta idea es que podemos lograr que $z^{1/2^k} \in (2^{-1/2^k}, 2^{-1/2^{k+1}})$ para cada $k \geq 0$ de una forma bastante natural, eligiendo simplemente $z \in (1/2, 1/\sqrt{2})$ fijo.

Puntos de cruce ideales Inspirados por esta idea, consideramos $z \in (0,1)$ y

$$f(z) \triangleq \lim_{k\to\infty} \left(L_{z^{1/2^k}}\left(C_{2^{k+1}}\right) - L_{z^{1/2^k}}\left(C_{2^k}\right) \right). \tag{4.3.1}$$

El cálculo de (4.3.1) se encuentra detallado en el apéndice B.1, mostramos aquí el resultado.

Proposición 4.3.1. *El límite puntual de* (4.3.1) *para* $z \in (0,1)$ *existe y es*

$$f(z) = g(z^2) - g(z) + 2\,, \tag{4.3.2}$$

donde

$$g(z) \triangleq \frac{2}{1-z} + \frac{z^\alpha \log(z)}{(1-z)^2}\Big((\alpha-1)z - \alpha\Big) + \frac{z^\alpha - z - 1}{1-z}, \quad \alpha = \sqrt{2} - 1\,. \tag{4.3.3}$$

Observación 4.3.1. La definición de $f(z)$ en (4.3.1) implica que $f(2^{-1}) \leq 0$ y $f(2^{-1/2}) \geq 0$, ya que C_{2^k} es óptimo para $2^{-1/2^k}$ y $C_{2^{k+1}}$ es óptimo para $2^{-1/2^{k+1}}$. Al ser $f(z)$ continua, esto implica que existe un cero, por lo menos, en el intervalo $[1/2, 1/\sqrt{2}]$.

La derivada de $f(z)$ es una función relativamente compleja, y por lo tanto, hallar el máximo de $f'(z)$ analíticamente parece estar fuera de alcance. Sin embargo, es posible estimar el máximo de $f'(z)$ numéricamente para obtener que $f'(z) \leq -1$ para cada $z \in (0,1)$. Como consecuencia esto implicaría que $f(z)$ es estrictamente decreceinte en $(0,1)$ y tiene un único cero $z_* \in (0,1)$.

Los puntos de transición de C_{2^k} a $C_{2^{k+1}}$ cuando $k \to \infty$, quedan determinados por $z_*^{1/2^k}$. Concretamente, si $(s_k)_{k=0}^\infty$ es una secuencia de puntos de corte, es decir $L_{s_k}\left(C_{2^k}\right) = L_{s_k}\left(C_{2^{k+1}}\right)$ y $s_k \in [2^{-1/2^k}, 2^{-1/2^{k+1}}]$ para cada k, entonces $s_k^{2^k} \to z_*$. [b] Observar que esto implica que $2^k \times |z_*^{1/2^k} - s_k| \to$

[b] La idea es la siguiente: De otro modo tendríamos una subsucesión $\left(s_{k_n}^{2^{k_n}}\right)_{n=0}^\infty$ convergente a algún $\tilde{z} \in [1/2, 1/\sqrt{2}] \setminus \{z_*\}$. Como entonces $s_{k_n} = \tilde{z}^{1/2^{k_n}} + o(1/2^{k_n})$, se sigue que

$$L_{\tilde{z}^{1/2^{k_n}}}\left(C_{2^{k_n+1}}\right) - L_{\tilde{z}^{1/2^{k_n}}}\left(C_{2^{k_n}}\right) = L_{s_{k_n}}\left(C_{2^{k_n+1}}\right) - L_{s_{k_n}}\left(C_{2^{k_n}}\right) + o(1) = o(1)\,,$$

por lo tanto $f(\tilde{z}) = 0$. Como estamos suponiendo que z_* es el único cero de $f(z)$, tenemos un absurdo.

0, lo cual muestra la importancia de $z_*^{1/2^k}$, ya que la longitud del intervalo $[2^{-1/2^k}, 2^{-1/2^{k+1}}]$ es del orden de $1/2^k$. En el apéndice B.1 se explica como también, si asumimos $f'(z) \leq -1$ como sugieren los resultados numéricos, tenemos aún más que $2^k \times |z_*^{1/2^k} - s_k| = O(1/2^k)$. Como consecuencia, la precisión de aproximar s_k con $z_*^{1/2^k}$ aumenta muy rápidamente.

El número z_* es a priori demasiado complicado como para tener una fórmula cerrada. Numéricamente, este es aproximadamente $z_* \approx 0.6066177$, y podemos aproximarlo mediante $z_* \approx e^{-1/2}$.

Simplificación ¿Por qué elegimos una aproximación que involucra a e? Para responder esta pregunta, observar que

$$z^{1/2^k} = 1 + \frac{\log(z)}{2^k} + O_z\left(4^{-k}\right),$$

donde O_z indica que la constante depende de z. Notar entonces que $\left(\frac{1}{\sqrt{e}}\right)^{1/2^k} = 1 - \frac{1}{2^{k+1}} + O\left(4^{-k}\right)$ y se sigue que la la penalización límite por utilizar $r_k \triangleq 1 - \frac{1}{2^{k+1}}$ como puntos de corte es $f\left(e^{-1/2}\right) \approx 0.0005741$ bits/par, ya que tomar $z^{1/2^k} + o(1/2^k)$ en lugar de $z^{1/2^k}$ no altera el resultado del límite (4.3.1).

Es así que podemos considerar utilizar C_{2^k} en el intervalo $(1 - \frac{1}{2^k}, 1 - \frac{1}{2^{k+1}}]$, siempre que k sea suficientemente grande. Observar que C_{2^k} es óptimo para $2^{-1/2^k} \approx 1 - \frac{\log(2)}{2^k}$ y en efecto $1 - \frac{\log(2)}{2^k}$ forma parte de nuestro intervalopara C_{2^k}, ya que $1/2 < \log(2) < 1$. Observar, no obstante, que esta aproximación no es muy buena cuando k es cercano a 0, pero que mejora rápidamente.

Redundancia límite A continuación mostramos el comportamiento límite de la redundancia tomando como regla que si $q \in (1 - \frac{1}{2^k}, 1 - \frac{1}{2^{k+1}}]$ elegimos C_{2^k}.

Calculamos entonces

$$\tilde{R}_2(z) \triangleq \lim_{k\to\infty} \frac{1}{2} L_{z^{1/2^k}}(C_{2^k}) - H\left(z^{1/2^k}\right) \tag{4.3.4}$$

con $z \in [e^{-1}, e^{-1/2})$. Esto porque $z^{1/2^k} = 1 + \frac{\log(z)}{2^k} + O_z\left(4^{-k}\right)$, y, por lo tanto, tenemos que $z^{1/2^k} \approx 1 - \frac{1}{2^k}$ para $z = e^{-1}$ y $z^{1/2^k} \approx 1 - \frac{1}{2^{k+1}}$ para $z = e^{-1/2}$, justificando así la elección del intervalo.

Proposición 4.3.2. *El límite* (4.3.4) *existe y es*

$$\tilde{R}_2(z) = \tfrac{1}{2}\, g(z) + \lg\log(1/z) - \lg(e)\,, \tag{4.3.5}$$

donde $g(z)$ es la definida función definida en (4.3.3).

Es sencillo obtener el resultado análogo para el caso unidimensional

Proposición 4.3.3. *El límite*

$$\tilde{R}_1(z) \triangleq \lim_{k\to\infty} L_{z^{1/2^k}}(G_{2^k}) - H(z^{1/2^k})\,, \tag{4.3.6}$$

existe y es

$$\tilde{R}_1(z) = \frac{1}{1-z} + \lg\log(1/z) - \lg(e)\,. \tag{4.3.7}$$

Esto nos permite comparar el utilizar el código bidimensional con el utilizar el código unidimensional, cuando usamos la regla de utilizar C_{2^k} y G_{2^k} en el intervalo $[e^{-1}, e^{-1/2}]$. Esta situación se encuentra ilustrada en la figura figura 4.3.1.

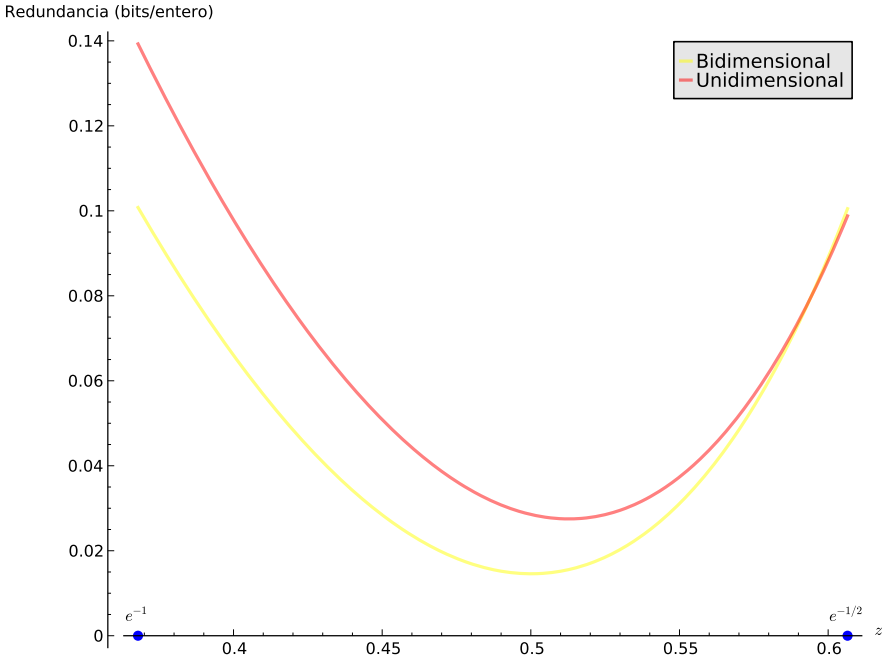


Figura 4.3.1: Las funciones $\tilde{R}_1(z)$ y $\tilde{R}_2(z)$ en función de $z \in [e^{-1}, e^{-1/2}]$.

Observar que el máximo de $\tilde{R}_1(z)$ es algo mayor al de la redundancia asintótica de los códigos de Rice óptimos. Esto se debe a que estamos tomando $z \in [e^{-1}, e^{-1/2}]$ en lugar de $z \in [\phi^{-2}, \phi^{-1}]$, que corresponde a la elección óptima del parámetro del código de Rice; esto es, utilizar G_{2^k} cuando

$$q = z^{1/2^k} \in [\phi^{-1/2^{k-1}}, \phi^{-1/2^k}].$$

Eliminación del uso de tablas Una vez que sabemos con cuál código C_{2^k} vamos a realizar la codificación, aún necesitamos tener un perfil para el top-code correspondiente. Para el caso unidimensional, el top-code es muy sencillo y queda determinado directamente por el parámetro, sin embargo, en el caso bidimensional no es tan sencillo calcular los perfiles. Una posibilidad es guardar una pequeña tabla con los valores de n_M para los k útiles, que, como los parámetros son potencias de dos, resulta bastante pequeña.

Sin embargo, esto no es del todo necesario a medida que $k \to \infty$. El $j \in \mathbb{Z}_{\geq 0}$ definido en (3.2.22) satisface

$$n_M = \frac{(j+2)\,(j+1)}{2} + r\,,$$

con $0 \leq r \leq 2^k$, y en nuestro caso, según el lema 3.2.6 tenemos, como se calcula en el Apéndice B.1, que

$$j/2^k = \alpha + O(1/2^k)\,,$$

donde $\alpha = \sqrt{2} - 1$. Cuando se trata de la redundancia asintótica, los términos que no se encuentren en j^2 en la ecuación para $j/2^k$, y r en la ecuación para n_M, se pueden omitir sin afectar los resultados. Es así que es posible utilizar una aproximación sencilla de α, por ejemplo $\sqrt{2} - 1 \approx \beta = 53/2^7$, para luego tomar $j_k \approx \beta\, 2^k$ y $n_M \approx \frac{j_k\,(j_k - 1)}{2}$. El deterioro asintótico por utilizar β en lugar de α se puede

ver tomando $\alpha \mapsto \beta$, $k \mapsto 2^k$ y $M \mapsto 2k-1$ en el corolario 3.2.3. En comparación con utilizar los códigos ideales, con esta regla modificada, estamos utilizando a lo sumo 1.6×10^{-4} bits/par extras, a medida que $k \to \infty$. En la práctica mantenemos una tabla con n_M para $k \leq 4$, y utilizamos esta regla aproxmada para $k \geq 5$.

Codificación de pares de TSGD iid

Consideramos dos variables aleatorias iid X e Y con distribución TSGD de parámetros $(\theta, d) \in (0,1) \times [0, 1/2]$. Recordar que podemos suponer que $d \in [0, 1/2]$ luego de aplicar la simetría $x \mapsto -1-x$, en caso de tener $1/2 \leq d \leq 1$. De manera similar a como se realizó en LOCO-I, consideramos una familia reducida de códigos

$$\mathcal{G}_2 = \{\Gamma_k : k \geq 0\} \,, \tag{4.3.8}$$

donde $\Gamma_k \triangleq C_{2^k}\left(M(\cdot), M(\cdot)\right)$, que es el análogo bidimensional.

Esta definición requiere un poco más de cuidado que en el caso de la distribución $TDGD$ como lo evidencia el siguiente lema, cuya prueba se presenta en el apéndice B.2.

Lema 4.3.1. *Consideramos $k \in \mathbb{Z}_{>0}$. Sea T_{2^k} el 'top-code' correspondiente a C_{2^k} que utilizamos para definir el código Γ_k. Sea $L_{\theta,d}\left(\Gamma_k\right)$ la longitud media de utilizar Γ_k para codificar un par de variables aleatorias (X, Y) iid con distribución TSGD de parámetros $(\theta, d) \in (0,1) \times [0, 1/2]$. Tenemos entonces*

$$L_{\theta,d}\left(\Gamma_k\right) = \frac{2}{1-\theta^{2^{k-1}}} + \sum_{0 \leq i,j < 2^k} \ell_{i,j}\, p_i\, p_j \,, \tag{4.3.9}$$

donde $\ell_{i,j} \triangleq \ell\left(T_{2^k}((i \bmod 2^k, j \bmod 2^k))\right)$ y $p_i \triangleq \Pr(M(X) \equiv i \bmod 2^k)$.

Además

$$p_i = \frac{C(\theta,d)}{1-\theta^{2^{k-1}}} \cdot \theta^{\lfloor i/2 \rfloor + \omega_d(i)} \,, \tag{4.3.10}$$

donde

$$\omega_d(i) = \begin{cases} \theta^d, & \textit{si } i \textit{ es par}\,; \\ \theta^{1-d}, & \textit{si } i \textit{ es impar}\,. \end{cases}$$

Observación 4.3.2. Observar que si $s = i + j$, entonces $p_i\, p_j = \left(\frac{C(\theta,d)}{1-\theta^{2^{k-1}}}\right)^2 \cdot \theta^{s/2+\beta_d(i,j)}$ donde

$$\beta_d(i,j) = \begin{cases} 1/2, & \text{si } i \not\equiv j (\text{mód}.2)\,; \\ 2d, & \text{si } i \text{ y } j \text{ son pares}\,; \\ 1-2d, & \text{si } i \text{ y } j \text{ son impares}\,. \end{cases}$$

Para elegir T_{2^k}, nos gustaría que este reduzca, en lo posible, la longitud media $\sum_{0 \leq i,j < 2^k} \ell_{i,j}\, p_i\, p_j$ en (4.3.9). Observar que esto corresponde a cambiar los pesos del alfabeto de entrada $\hat{\mathcal{A}}_{2^k}$ del código T_{2^k}, definidos en (3.2.2), por $w(i,j) = \theta^{(i+j)/2+\beta_d(i,j)}$.

En este caso, la firma s no determina la probabilidad totalmente, como ocurre en el caso de la fuente que determina al 'topcode' T_{2^k}. Es así que, no resulta indistinta la forma en que ordenamos las hojas de una firma s dada en T_{2^k}.

Concretamente, si s es impar, tener $i + j = s$ implica que i y j son incongruentes módulo 2, y por lo tanto $\beta_d(i,j) = 1/2$. Así pues, si s es impar, todas las hojas con dicha firma tienen igual probabilidad. En cambio, si s es par, tener $i + j = s$ implica que i y j son, o ambos pares, o ambos impares. En

dicho caso hay que discutir según $2d \geq 1-2d$, es decir, según si $d \geq 1/4$ o no. Si $d \geq 1/4$ colocamos los pares (i,j) con $i+j=s$ que tengan i,j impares lo más arriba posible en el árbol, y en otro caso colocamos los que tengan i,j pares lo más arriba posible.

De cualquier manera, la inicidencia de la elección de un T_{2^k} parece ser pequeña. De hecho, los resultados asintóticos que probamos a continuación son independientes de dicha elección. En la práctica realizamos la elección arbitraria de ordenar los pares $(i,j) \in \{0,\dots,2^k-1\} \times \{0,\dots,2^k-1\}$ con $i+j=s$ de forma creciente en i.

Tenemos un resultado asintótico para la selección de los códigos C_{2^k} cuando tenemos una geométrica bidimensional, el siguiente lema, cuya prueba se encuentra en el apéndice B.2, muestra que este resultado se extiende a los códigos de la familia $\mathcal{G}_2$.

Lema 4.3.2. *Sea $L_{\theta,d}(\Gamma_k)$ la longitud media de utilizar Γ_k para codificar un par de variables aleatorias iid con distribución TSGD de parámetros (θ,d). Tenemos, para todo $d_1,d_2 \in [0,1/2]$ que*

$$\left| L_{\theta,d_1}(\Gamma_k) - L_{\theta,d_2}(\Gamma_k) \right| \leq 4|d_1-d_2| \times \theta^{-2^k} \frac{\log(\theta^{-1})}{\theta^2}\,. \tag{4.3.11}$$

Intuitivamente, esto nos dice que la dependencia de d desaparece a medida que $\theta \to 1$, si dejamos fijo k. Sin embargo, nosotros estamos interesados en el caso en que k varía con θ, ya que debemos hallar los puntos de transición de Γ_k a Γ_{k+1}.

No obstante, observamos también que si $\theta^{2^k} \geq C > 0$, el lema 4.3.2 sigue mostrando que la dependencia de d desaparece. Como en particular para el caso $d=1/4$ tenemos que $M(X)$ y $M(Y)$ son geométricas de parámetro $\sqrt{\theta}$, el lema 4.3.2 implica que tenemos también el mismo resultado asintótico

Proposición 4.3.4. *Dados $z \in (0,1)$ y $d \in [0,1/2]$, tenemos*

$$f(\sqrt{z}) = \lim_{k\to\infty} L_{z^{1/2^k},d}(\Gamma_{k+1}) - L_{z^{1/2^k},d}(\Gamma_k)\,, \tag{4.3.12}$$

donde $f(z)$ es la función definida por (4.3.1), que se calcula como se explica en la proposición 4.3.1.

Por lo tanto consideramos e^{-1} como una buena aproximación de la raíz de $z \mapsto f(\sqrt{z})$. Nuevamente, la penalización asintótica por usar estos puntos de cruce es $f(e^{-1/2})$.

La regla simplificada será, por lo tanto, utilizar Γ_{k+1} cuando $\theta \in (e^{-1/2^k}, e^{-1/2^{k+1}}]$, y Γ_0 si $\theta \in [0,e^{-1}]$. Los resultados de utilizar esta regla, comparados con los puntos de cruce ideales, donde es $[l_k,r_k]$ el intervalo ideal para Γ_k, se encuentran ilustrados en la tabla 4.1, redondeados a tres cifras decimales. Estos resultados se encuentran también graficados en la figura 4.3.3, comparar esto con la figura 4.3.2 que corresponde a los puntos de corte ideales.

Observando la tabla 4.1, la precisión parece ser peor en el de transición de Γ_0 a Γ_1. En este caso no es difícil realizar los cálculos explícitos, para obtener que el punto óptimo de cruce r satisface

$$r = \frac{1}{2}\left(1 - r^{2-2d}\right)\,. \tag{4.3.13}$$

Notar entonces que r varía en $[1/3, \sqrt{2}-1]$, alcanzando su máximo $r=\sqrt{2}-1$ para $d=0$, y su mínimo $r=1/3$ para $d=1/2$. En cualquier caso, como $\sqrt{2}-1 \approx 0.414$, vemos que $1/e$ no es tan mala aproximación de r en general.

El punto r de transición de Γ_0 a Γ_1 se puede obtener exactamente comparando

$$\rho = \frac{\theta^d}{\theta^d + \theta^{1-d}}, \quad \text{y} \quad S = \frac{\theta}{1-\theta}\,.$$

k	l_k	r_k	$e^{-1/2^k}$	k	l_k	r_k	$e^{-1/2^k}$	k	l_k	r_k	$e^{-1/2^k}$
0	0.000	0.414	0.368	0	0.000	0.389	0.368	0	0.000	0.333	0.368
1	0.414	0.604	0.607	1	0.389	0.609	0.607	1	0.333	0.615	0.607
2	0.604	0.781	0.779	2	0.609	0.780	0.779	2	0.615	0.779	0.779
3	0.781	0.883	0.882	3	0.780	0.883	0.882	3	0.779	0.883	0.882
4	0.883	0.939	0.939	4	0.883	0.939	0.939	4	0.883	0.939	0.939
5	0.939	0.969	0.969	5	0.939	0.969	0.969	5	0.939	0.969	0.969

Tabla 4.1: Resultados para $d = 0$, $d = 0.2$ y $d = 0.5$.

Esto se sigue de notar C_{2^0} y C_{2^1} corresponden directamente a la concatenación de los códigos unidimensionales correspondientes, y luego aplicar el lema 4.2.1.

Como en el caso de los pares de geométricas, observamos que $e^{-1/2^k} = 1 - \frac{1}{2^k} + \frac{1}{2\cdot 4^k} + O(1/8^k)$, lo cual nos permite simplificar aún más el resultado, como se discute a continuación.

Relación con la regla que utiliza JPEG-LS Si consideramos la aproximación $e^{-1/2^k} \approx 1 - \frac{1}{2^k}$, obtenemos que debemos utilizar Γ_k, con $k \geq 1$, siempre y cuando

$$1 - \frac{1}{2^{k-1}} < \theta \leq 1 - \frac{1}{2^k}\,, \tag{4.3.14}$$

o, equivalentemente, si $S = \dfrac{\theta}{1-\theta}$, como

$$2^{k-1} - 1 < S \leq 2^k - 1\,, \tag{4.3.15}$$

que corresponde, a menos de un término despreciable cuando $\theta \to 1$, a la regla de codificación que utiliza JPEG-LS para el caso unidimensional.

Esto asegura entonces que la regla que utiliza JPEG-LS es buena para este caso también, siempre que k sea relativamente grande. Sin embargo, elegir k como en (4.3.14), o su equivalente (4.3.15), es claramente malo cuando $\theta \in (0, 1/3)$, ya que elige Γ_1 cuando deberíamos en realidad elegir Γ_0. Es más, incluso, si arreglaramos el intervalo correspondiente a Γ_0, el de Γ_1 seguiría abarcando menos de lo que debe.

Dicho esto, si en (4.3.15) cambiamos los términos -1 a ambos lados por $-1/2$ (por el $1/2$ que aproxima N_t/t), tenemos aproximadamente la regla simplificada de JPEG-LS. Observar que esto corresponde, inviertiendo el mapa $\theta \mapsto S = \dfrac{\theta}{1-\theta}$, a elegir Γ_k siempre que

$$1 - \frac{2}{2^k + 1} < \theta \leq \frac{2^k - 1/2}{1 + (2^k - 1/2)} = \frac{2^{k+1} - 1}{2^{k+1} + 1} = 1 - \frac{2}{2^{k+1} + 1}\,,$$

que es una aproximación inceríblemente ajustada para nuestro caso, incluso para $k = 0, 1, 2$. En efecto, obtenemos aproximadamente $0.0, 0.333, 0.6, 0.777, 0.882, 0.939, \ldots$

Variante de la regla de selección. Finalmente, para una versión última versión, consideramos utilizar $e^{-1/2^k} = 1 - \frac{1}{2^k} + \frac{1}{2\cdot 4^k} + O(1/8^k)$ para los puntos de corte. En otras palabras, utilizamos Γ_{k+1} cuando

$$\theta \in \left(1 - \frac{1}{2^k} + \frac{1}{2\cdot 4^k}, 1 - \frac{1}{2^{k+1}} + \frac{1}{2\cdot 4^{k+1}}\right],$$

y Γ_0 en caso de que $\theta \leq 1 - \frac{1}{2^0} + \frac{1}{2\cdot 4^0} = \frac{1}{2}$. Esta resulta una buena aproximación para los bordes derechos de Γ_{k+1}, sin embargo $1/2$ sigue siendo una mala aproximación del punto de transición de Γ_0 a Γ_1, y así consideramos aproximarlo por separado, mediante $1/e \approx 3/8$. En la figura 4.3.2 se ilustra la redundancia si utilizamos el mejor código posible de la familia $\mathcal{G}$, mientras que en la figura 4.3.3 se ilustra el resultado de utilizar esta regla de selección. En limpio, tenemos entonces

Algoritmo 4.3.1 Regla de selección RS_2.

if $\theta \leq \frac{3}{8}$ **then**
  Codificar utilizando Γ_0
else
  $k \leftarrow$ menor entero no-negativo tal que $\theta \leq 1 - \dfrac{1}{2^{k+1}} + \dfrac{1}{2\cdot 4^{k+1}}$.
  Codificar utilizando Γ_{k+1}
end if

En la práctica estimamos θ mediante

$$\hat{\theta} = \frac{A - U}{A - U + N}\,.$$

Aquí $N = t$ el número de ocurrencias del contexto dado, $A = S_t + N_t = \sum_{i=1}^{t} |\epsilon_i|$ la suma de los valores absolutos de los residuos en el contexto dado, y $U = N_t = \sum_{i=1}^{t} \mathbf{1}_{<0}(\epsilon_t)$.

Observar que todas estas operaciones se pueden implementar sencillamente con adiciones y operaciones con bits. En efecto, si escribimos $s \triangleq A - U + N$ y $r \triangleq A - U$, esta regla simplifica a la siguiente

a) Si $2s + s \geq 2^3\, r$, utilizar Γ_0.

b) En otro caso, utilizar Γ_k donde $k \geq 1$ es el **menor** entero positivo tal que

$$2^{2k+1}\, N + s \geq 2^{k+1}\, s\,.$$

La búsqueda del item b se puede implementar como sigue

```
for (k = 1;  (N << (2*k+1)) + s < (s<<(k+1)); k++);
```

Para referirnos a esta regla de selección en el capítulo 5 la denotamos mediante RS_2, entendiendo también que RS_1 corresponde a la regla de selección de parámetro de LOCO-I.

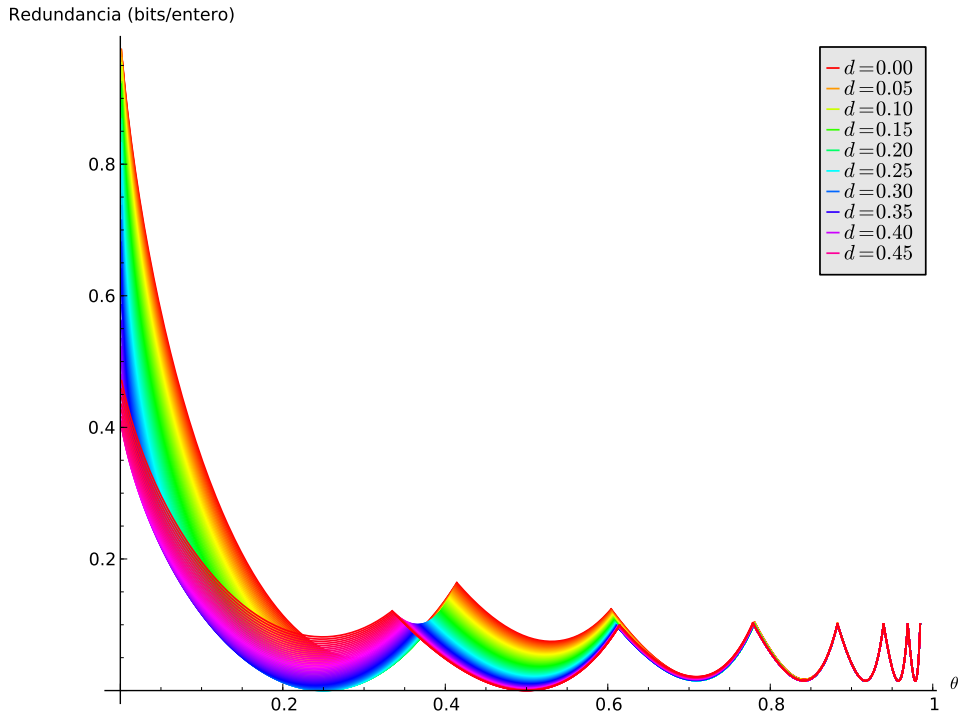


Figura 4.3.2: Redundancia de la codificación de (X, Y), eligiendo para cada q el código de $\mathcal{G}_2$ que minimza la longitud media. El valor de d se indica con color, que varía de forma continua cuando d varía en $[0, 1/2]$.

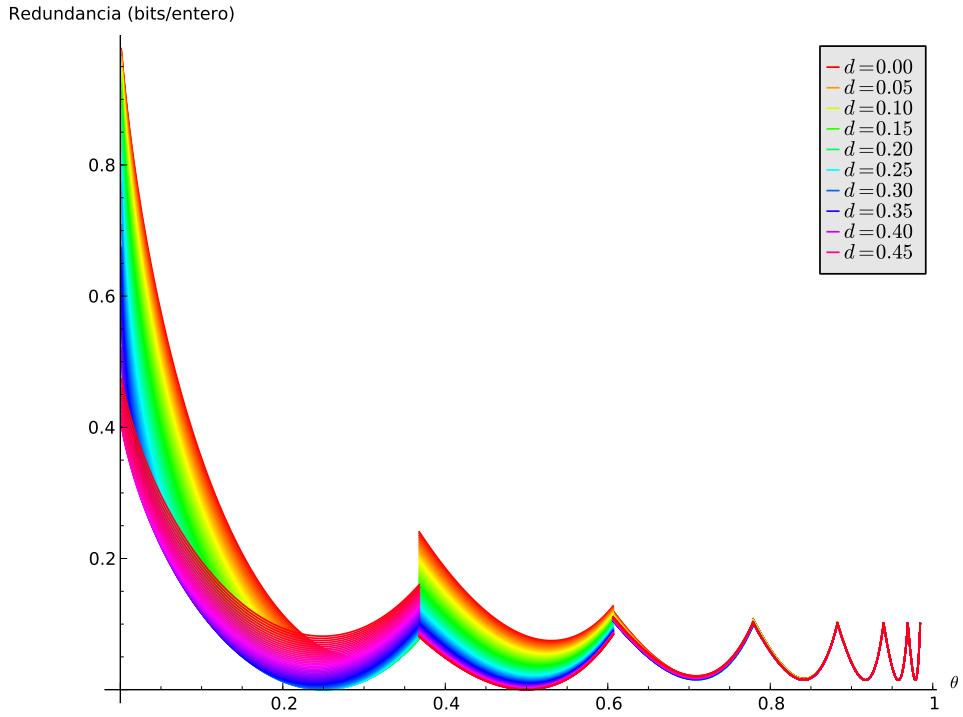


Figura 4.3.3: Redundancia de la codificación de (X, Y), eligiendo para cada q el código de $\mathcal{G}_2$ según la regla simplificada de estimar los puntos de cruce con $e^{-1/2^k}$. El valor de d se indica con color, que varía de forma continua cuando d varía en $[0, 1/2]$.

Comentarios finales

Asumir que ambos residuos de predicción tienen exactamente la misma distribución puede parecer una hipótesis bastante reestrictiva. Así pues, contemplamos brevemente el caso en el que el parámetro θ es distinto para ambos componentes del par (X, Y). Para no extendernos excesivamente, olvidamos la dependencia con $d \in [0, 1/2]$, y nos consideramos un par (X, Y) de variables aleatorias discretas e independientes, con distribuciones $ODGD(q_1)$ y $ODGD(q_2)$ respectivamente, donde en realidad X e Y corresponderían a $M(X)$ y $M(Y)$ respectivamente si $d = 1/4$.

Supongamos que elegimos $k \in \mathbb{Z}_{\geq 0}$ de forma que $q_1, q_2 \in (e^{-1/2^k}, e^{-1/2^{k+1}})$, ¿será que codificar (X, Y) mediante $C_{2^k}(X, Y)$ es mejor que utilizar $G_{2^k}(X) \bowtie G_{2^k}(Y)$? Esta pregunta ya la tratamos cuando $q_1 = q_2$ (ver la figura 4.3.1), sin embargo, en la práctica puede ser más razonable asumir que $q_1 \neq q_2$. El siguiente resultado muestra que, asintóticamente, es mejor utilizar C_{2^k}.

Proposición 4.3.5. *Sean $z, u \in [e^{-1}, e^{-1/2}]$, con $z \neq u$. Tenemos que*

$$\tau(z, u) \triangleq \lim_{k \to \infty} \left(L_{z^{1/2^k}, u^{1/2^k}}(C_{2^k}) - L_{z^{1/2^k}}(G_{2^k}) - L_{u^{1/2^k}}(G_{2^k}) \right), \tag{4.3.16}$$

existe y satisface

$$\tau(z, u) = \frac{1}{(1-z)(1-u)} \times \left(\frac{(z^\alpha - z)u \log(u) - (u^\alpha - u) z \log(z)}{\log(z) - \log(u)} - \frac{(z^\alpha - 1)\log(u) - (u^\alpha - 1)\log(z)}{\log(z) - \log(u)} \right), \tag{4.3.17}$$

donde $\alpha \triangleq \sqrt{2} - 1$. De hecho tenemos

$$L_{z^{1/2^k}, u^{1/2^k}}(C_{2^k}) - L_{z^{1/2^k}}(G_{2^k}) - L_{u^{1/2^k}}(G_{2^k}) = \tau(z, u) + O(1/2^k).$$

La prueba de esta proposición se basa en probar un resultado análogo al corolario 3.2.3, pero con dos variables. Como la prueba es muy similar, se omite aquí. Esta función $\tau(z, u)$ tiene un máximo igual a aproximadamente 3.4×10^{-3} bits/par, y por lo tanto, asintóticamente, los casos en los que utilizar $C_{2^k}(\cdot, \cdot)$ en lugar de $G_{2^k}(\cdot) \bowtie G_{2^k}(\cdot)$ resulta en un deterioro, son despreciables. La función τ se encuentra graficada en la figura 4.3.4, donde se puede apreciar que τ resulta en general negativa. Al interpretar la figura 4.3.4, recordar que la unidad correspondiente es 'bits/par'. En la figura 4.3.5 se pueden apreciar correspondientes resultados para casos concretos de k.

Esto motiva la regla que se utiliza en última instancia, que es utilizar el código bidimensional cuando los k estimados mediante la regla 4.3.2 para cada componente coinciden, y codificar utilizando C_{2^k} para el parámetro común k. Si el contexto aparece por primera vez, o los parámetros k calculados para cada componente no coinciden, utilizamos los códigos unidimensionales para cada componente (actualizando la estadística conjunta en el medio). Los contextos para ambos canales se actualizan una vez codificado el resultado utilizando C_{2^k}. Esto último en principio puede llegar a correr con la desventaja de que, si los codificamos en serie, podemos hacer uso de la información de la primera muestra para codificar la segunda, ya que estamos llevando las estadísticas conjuntamente.

Recordar que también tenemos el códigos $\mathcal{C}^2_{\text{lím}}$ para el intervalo $[0, 1/2]$. Una regla posible entonces, podría ser utilizar este código para cuando ambos parámetros estimados $\hat{q}_1, \hat{q}_2$ son menores que $1/4$. Sin embargo, por lo general, los casos de baja entropía se espera que sean mejor manejados por una codificación por corridas, es decir, el run-mode en LOCO-I. En conclusión, si bien está implementado el código límite también, no es de particular interés su utilización. Es por esto que la implementación de LOCO-I que utilizamos omite el run-mode, y al codificar con los códigos bidimensionales omitimos la utilización del código límite.

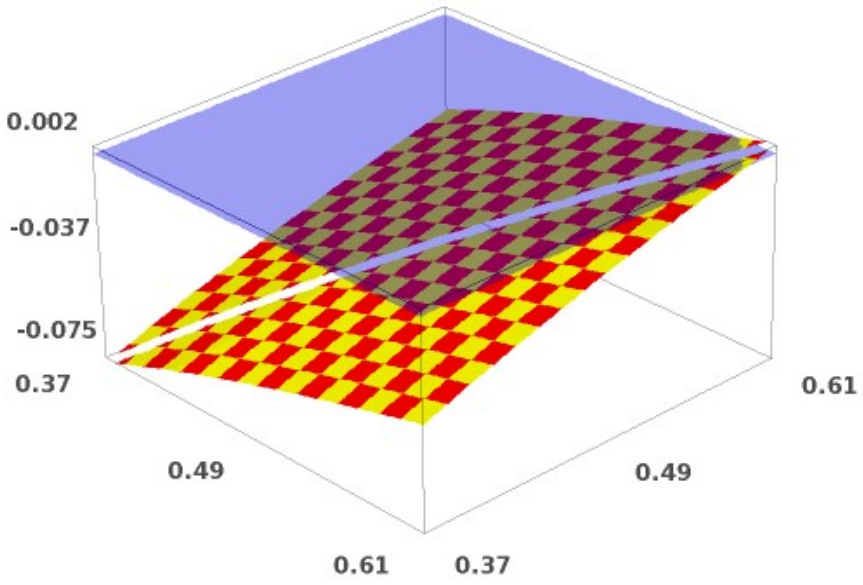


Figura 4.3.4: La función $\tau(x, y)$ de la proposición 4.3.5 para $(x, y) \in [1/e, 1/\sqrt{e}] \times [1/e, 1/\sqrt{e}]$. El plano $z = 0$ se encuentra marcado en azul para poder distinguir el signo de $\tau(x, y)$.

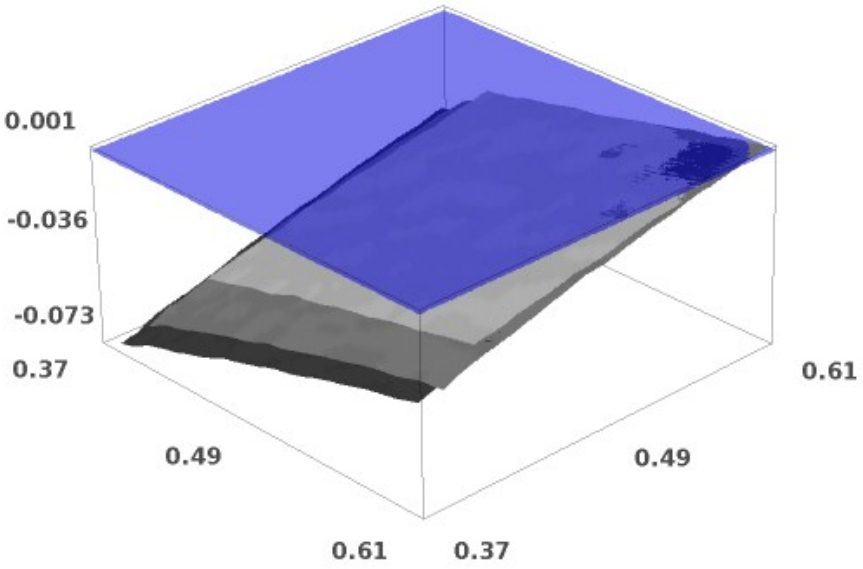


Figura 4.3.5: Estimaciones empíricas para $k = 2, 3, 4$ (intensidad de luz decrece con k). Entre los puntos empíricos se ha interpolado linealmente.

CAPÍTULO 5

Resultados experimentales

En este capítulo estudiamos los resultados experimentales obtenidos aplicando LOCO-I junto con códigos bidimensionales sobre un conjunto de 24 imágenes[a]. Estas imágenes se encuentran representadas en el espacio RGB, con muestras de precisión $\beta = 8$ bits y sus dimensiones son 768×512 en todos los casos.

Resumen de la codificación. El procedimiento que utilizamos comienza, en primer lugar, por comprimir el canal G utilizando únicamente los códigos unidimensionales (como LOCO-I, pero sin 'run-mode'). Luego, utilizando estadísticas separadas, con las regiones de cuantización de los contextos levemente distintas (porque $R - G$ y $B - G$ no se comportan de igual forma que G), codificamos $(R - G, B - G)$. Para codificar $(R - G, B - G)$ para un píxel dado, comenzamos por calcular los parámetros k_1 y k_2 correspondientes a cada componente, utilizando alguna de las reglas de selección. Si $k_1 \neq k_2$, codificamos utilizando los códigos unidimensionales, mientras que si $k_1 = k_2$, codificamos los residuos de predicción (ajustados primero de forma que el offset de cada uno satisfaga $d \in [0, 1/2]$) utilizando $\Gamma_{k_1} = \Gamma_{k_2}$, donde recordamos que se define $\Gamma_k(\cdot, \cdot) \triangleq C_{2^k}(M(\cdot), M(\cdot))$.

Actualización de estadísticas. Cuando codificamos un par de intensidades $(R - G, B - G)$ de un píxel dado mediante códigos unidimensionales, codificamos primero $R - G$ y luego $B - G$ separadamente, dejando entonces la posibilidad de actualizar las estadísticas luego de codificar $R-G$. Como las estadísticas de $R-G$ y $B-G$ se acumulan conjuntamente, tras codificar el componente $R-G$ del par tenemos información más actualizada respecto al contexto correspondiente a $R - G$. Si el componente $B-G$ del par ocurre en el mismo contexto, la actualización de estadísticas en el 'medio' del par puede resultar benficiosa ya que tenemos estadísticas más frescas.

Regla de selección de código. Recordamos que para la selección del código bidimensional de la familia $\mathcal{G}_2$ definida en (4.3.8) tenemos definidas dos reglas: la regla de selección del parámetro de los códigos Golomb-potencia-de-2 que utiliza LOCO-I que denotamos RS_1, y la regla RS_2 definida en el

[a] Las imágenes de la 'Kodak Lossless True Color Image Suite'.

pseudo-código 4.3.1, donde θ se estima mediante $\frac{A-U}{A-U+N}$ (ver la sección 4.3.2 por detalles). Dada la similitud de los puntos de cruce dados por RS_2 con los puntos de corte que produce RS_1 bajo la suposición de que $N_t/t \approx 1/2$, se probó también utilizar RS_2 para la selección de los unidimensionales, obteniendo resultados ligeramente mejores que con RS_1. Es por esto que en los experimentos, a la hora de comparar, o utilizamos únicamente la regla RS_1 para ambas selecciones, o únicamente RS_2 para ambas selecciones.

5.1. Resultados

Comenzamos permitiendo la actualización de estadísticas en el medio del par, para el caso unidimensional. Los resultados obtenidos utilizando las reglas RS_2 y RS_1 se pueden visualizar en las tablas 5.1 y 5.2 respectivamente. Aquí las columnas $\mathcal{U}$ y $\mathcal{B}$ indica los resultados en bytes de comprimir utilizando únicamente los códigos unidimensionales y con el agregado los códigos bidimensionales, respectivamente. Finalmente, la columna $\Delta_{rel}(\%)$ se calcula mediante $\Delta_{rel}(\%) = 100\,\frac{\mathcal{B}-\mathcal{U}}{\mathcal{U}}$ y representa la diferencia relativa porcentual entre $\mathcal{B}$ y $\mathcal{U}$.

Observando las tablas 5.1 y 5.2 notamos que:

- La diferencia entre los largos de código que representan las columnas $\mathcal{U}$ y $\mathcal{B}$ es prácticamente despreciable.
- Más aún, esta diferencia ínfima parece ser en contra de la utilización de códigos bidimensionales.
- El método de selección RS_2 parece acentuar la ventaja de los códigos unidimensionales.

Con respecto al último punto, en parte es razonable que el método de selección RS_2 favorezca más a los códigos unidimensionales cuando los parámetros estimados son pequeños. En efecto, recordamos que la regla RS_2 elige $k \in \mathbb{Z}_{>0}$ cuando este es el menor entero positivo tal que

$$\theta \leq 1 - \frac{1}{2^k} + \frac{1}{2 \cdot 4^k}\,,$$

donde θ es el parámetro de la distribución $TSGD(\theta, d)$ estimado. Aquí $1 - \frac{1}{2^k} + \frac{1}{2\cdot 4^k}$ es una sobre-estimación de $e^{-1/2^k}$. Como en el caso de los códigos de Rice los puntos de corte son de la forma $\phi^{-1/2^k}$ con $\phi = \frac{1+\sqrt{5}}{2}$ y $e^{-1/2} < \phi^{-1}$, este cambio es positivo. Más aún porque tomamos el punto de corte de $G_1(M(\cdot))$ con $G_2(M(\cdot))$ como $3/8$, una excelente aproximación de ϕ^{-2}.

Existen ventajas similares para los códigos bidimensionales ya que los puntos de cruce ideales quedan determinados por z_* que es levemente mayor que $e^{-1/2}$. Sin embargo para k pequeño la estimación es menos ajustada, por ejemplo, $1 - \frac{1}{2^0} + \frac{1}{2\cdot 4^0} = 0.625$ parece sobre-estimar demasiado los puntos de cruce de Γ_1 a Γ_2 como se puede apreciar en la figura 4.3.2 para el caso en el que los componentes son iid. Sin embargo, esta clase de inconvenientes desaparecen a medida que k aumenta, mientras que en el caso de los códigos unidimensionales la estimación se vuelve relativamente menos ajustada que para los bidimensionales.

¿Por qué estos resultados ? En principio parece contra-intuitivo que las diferencias encontradas, por más despreciable que sean, sugieran un peor desempeño por parte de la variante que utiliza los códigos bidimensionales. Recordamos entonces que los códigos unidimensionales están en realidad actualizando estadísticas luego de codificar el primer componente del par $(R - G, B - G)$

Realizamos experimentos posponiendo la actualización de la estadística al usar los códigos unidimensionales. Esto es, codificamos $R - G$, pero no actualizamos la estadística hasta no haber codificado

Imagen	$\mathcal{U}$	$\mathcal{B}$	$\Delta_{rel}(\%)$	Imagen	$\mathcal{U}$ (bytes)	$\mathcal{B}$ (bytes)	$\Delta_{rel}(\%)$
1	504482	504594	0.022	13	592872	592927	0.011
2	448093	448157	0.014	14	497106	497160	0.012
3	382579	382701	0.032	15	434492	434613	0.032
4	454706	454774	0.015	16	417725	417815	0.020
5	541627	541682	0.010	17	435949	435984	0.006
6	469204	469268	0.014	18	550230	550226	−0.001
7	410514	410638	0.030	19	474138	474180	0.010
8	543586	543679	0.017	20	400247	400233	−0.003
9	429669	429754	0.020	21	478713	478754	0.010
10	435152	435225	0.017	22	512182	512221	0.009
11	454265	454351	0.019	23	417238	417327	0.020
12	406357	406479	0.030	24	501490	501486	−0.001

Tabla 5.1: Resultados para RS_2 en los bidimensionales, y actualización de estadísticas en el medio del par al utilizar los códigos unidimensionales.

Imagen	$\mathcal{U}$ (bytes)	$\mathcal{B}$ (bytes)	$\Delta_{rel}(\%)$	Imagen	$\mathcal{U}$ (bytes)	$\mathcal{B}$ (bytes)	$\Delta_{rel}(\%)$
1	505483	505574	0.018	13	593570	593607	0.007
2	448757	448828	0.016	14	497501	497549	0.011
3	383176	383308	0.034	15	434802	434926	0.032
4	455018	455067	0.011	16	418364	418437	0.016
5	542305	542342	0.007	17	436408	436476	0.013
6	469876	469936	0.013	18	550703	550664	−0.008
7	411072	411179	0.026	19	474510	474577	0.016
8	544229	544292	0.012	20	400625	400595	−0.006
9	430510	430594	0.020	21	479518	479554	0.008
10	435971	436033	0.014	22	512832	512842	0.002
11	454976	455063	0.019	23	417923	418002	0.017
12	407222	407363	0.035	24	502128	502093	−0.009

Tabla 5.2: Resultados para RS_1 en los bidimensionales, y actualización de estadísticas en el medio del par al utilizar los códigos unidimensionales.

Imagen	$\mathcal{U}$ (bytes)	$\mathcal{B}$ (bytes)	$\Delta_{rel}(\%)$	Imagen	$\mathcal{U}$ (bytes)	$\mathcal{B}$ (bytes)	$\Delta_{rel}(\%)$
1	504592	504594	0.000	13	592950	592927	−0.005
2	448152	448157	0.001	14	497184	497160	−0.005
3	382703	382701	−0.001	15	434646	434613	−0.009
4	454798	454774	−0.005	16	417825	417815	−0.002
5	541716	541682	−0.006	17	436010	435984	−0.005
6	469263	469268	0.001	18	550299	550226	−0.016
7	410627	410638	0.003	19	474207	474180	−0.007
8	543681	543679	−0.000	20	400283	400233	−0.009
9	429781	429754	−0.006	21	478753	478754	0.000
10	435238	435225	−0.003	22	512252	512221	−0.007
11	454353	454351	−0.000	23	417362	417327	−0.008
12	406499	406479	−0.005	24	501619	501486	−0.033

Tabla 5.3: Resultados para RS_2 en los bidimensionales, sin actualización de estadísticas en el medio del par al utilizar los códigos unidimensionales.

Imagen	$\mathcal{U}$ (bytes)	$\mathcal{B}$ (bytes)	$\Delta_{rel}(\%)$	Imagen	$\mathcal{U}$ (bytes)	$\mathcal{B}$ (bytes)	$\Delta_{rel}(\%)$
1	505607	505574	−0.007	13	593635	593607	−0.006
2	448839	448828	−0.002	14	497594	497549	−0.010
3	383315	383308	−0.002	15	434971	434926	−0.012
4	455117	455067	−0.011	16	418470	418437	−0.007
5	542386	542342	−0.008	17	436516	436476	−0.007
6	469939	469936	−0.001	18	550754	550664	−0.019
7	411204	411179	−0.006	19	474598	474577	−0.005
8	544310	544292	−0.003	20	400664	400595	−0.013
9	430639	430594	−0.010	21	479563	479554	−0.002
10	436065	436033	−0.007	22	512900	512842	−0.013
11	455083	455063	−0.004	23	418050	418002	−0.011
12	407400	407363	−0.009	24	502257	502093	−0.040

Tabla 5.4: Resultados para RS_1 en los bidimensionales, sin actualización de estadísticas en el medio del par al utilizar los códigos unidimensionales.

$B - G$ para el mismo píxel. Los resultados correspondientes se muestran en las tablas 5.3 y 5.4, mostrando ahora lo que parece ser una débil ventaja pero favor de los códigos bidimensionales.

De todos modos, observando las figuras 4.3.1 y 4.3.2, uno esperaría una diferencia algo mayor. Para investigar este asunto estudiamos las siguientes interrogantes:

- ¿Cuán seguido estamos utilizando los códigos bidimensionales ?
- ¿Cuán comúnes son los parámetros k que favorecen $C_{2^k}(\cdot)$ sobre $G_{2^k}(\cdot) \bowtie G_{2^k}(\cdot)$?
- Teóricamente, ¿qué deberíamos esperar?

En las secciones subsiguientes estudiamos estos asuntos.

5.2. Distribución de los parámetros estimados

Si bien la selección del parámetro k de los códigos afectan la frecuencia con que son utilizados en última instancia los códigos bidimensionales, los resultados experimentales muestran que esto no varía muy fuertemente. Es por esto que fijamos un caso de los cuatro presentados en la sección anterior, entendiendo que los resultados y el análisis son fundamentalmente los mismos para los demás casos. El caso elegido se conforma utilizando la regla de selección RS_2, y no actualizando las estadísticas en el medio del par al utilizar los códigos unidimensionales para codificar $(R-G, B-G)$.

Frecuencia con la que ambos parámetros k son iguales. Comenzamos con la pregunta de qué tan seguido estamos utilizando la familia de códigos bidimensionales $\mathcal{G}_2$. En la tabla 5.5 se muestra, para cada imagen de nuestro conjunto de muestra, la proporción de píxeles para los cuales se utiliza algún código de la familia $\mathcal{G}_2$ para codificar los residuos de $R-G$ y $B-G$. Observar que, generalmente, para al menos para la mitad de los píxeles se ha utilizado algún código de $\mathcal{G}_2$.

Imagen	Prop. C_{2^k}	Imagen	Prop. C_{2^k}	Imagen	Prop. C_{2^k}	Imagen	Prop. C_{2^k}
1	0.593	7	0.572	13	0.649	19	0.630
2	0.529	8	0.608	14	0.506	20	0.493
3	0.609	9	0.561	15	0.541	21	0.567
4	0.496	10	0.551	16	0.594	22	0.601
5	0.558	11	0.638	17	0.569	23	0.550
6	0.555	12	0.583	18	0.661	24	0.531

Tabla 5.5: Proporción de píxeles para los cuales se ha utilizado algún código de $\mathcal{G}_2$.

Distribución de los parámetros k. Si bien los códigos bidimensionales parecen utilizarse frecuentemente, como muestra la tabla 5.6 los parámetros que se utilizan son prácticamente siempre $k=0$ y $k=1$. Recordamos que para $k=0$ y $k=1$ tenemos que $C_{2^k}(\cdot,\cdot)$ es exactamente $G_{2^k}(\cdot) \bowtie G_{2^k}(\cdot)$, y por lo tanto utilizar los códigos bidimensionales no representa una ganancia frente a la utilización de los códigos unidimensionales.

Para $k=0$ es bastante probable que se esté en un 'run' de LOCO-I (ver 4.2.4), por lo tanto, si bien habría ganancias aquí por utilizar el código límite $\mathcal{C}_{\text{lím}}$ para $q \to 0$, no tendría sentido analizarlo, ya que se entiende que el 'run-mode' cubre este caso más eficientemente.

Uno podría pensar que, quizás, siendo que la tabla 5.6 muestra que los parámetros usados para los códigos bidimensionales han sido casi siempre $k=0$ y $k=1$, solo estemos en las hipótesis para utilizar los códigos bidimensionales cuando estamos en una zona de baja entropía, es decir una zona muy precedible. Sin embargo, analizando los parámetros de Golomb-Potencia-de-2 para los canales $R-G$ y $B-G$ completos (esto es, no solamente cuando sus parámetros coinciden), observamos que para este conjunto de imágenes los parámetros también resultan en general $k=0$ y $k=1$.

Recordamos entonces que a menor k, menor es el parámetro θ y más acertada es el resultado de la predicción mediante el predicor fijo, más el término adaptativo. En este conjunto de imágenes parece ser, en principio, que los canales $R-G$ y $B-G$ resultan ser relativamente predecibles, y esto deja poco margen para aprovechar los códigos bidimensionales.

Para imágenes algo menos naturales, como la figura 5.2.1 de dimensiones 253×254, que tiene una cierta discretización (las intensidades toman únicamente unos pocos valores) y ruido, se obtienen por lo

Imagen	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
01	0.391	0.608	0.001	0.000	0.000	0.000	0.000	0.000
02	0.389	0.609	0.002	0.000	0.000	0.000	0.000	0.000
03	0.758	0.225	0.007	0.008	0.002	0.000	0.000	0.000
04	0.514	0.473	0.013	0.001	0.000	0.000	0.000	0.000
05	0.209	0.713	0.060	0.015	0.004	0.000	0.000	0.000
06	0.414	0.580	0.004	0.001	0.000	0.000	0.000	0.000
07	0.591	0.395	0.013	0.000	0.000	0.000	0.000	0.000
08	0.146	0.818	0.034	0.002	0.000	0.000	0.000	0.000
09	0.527	0.468	0.005	0.001	0.000	0.000	0.000	0.000
10	0.487	0.503	0.007	0.002	0.000	0.000	0.000	0.000
11	0.468	0.527	0.004	0.000	0.000	0.000	0.000	0.000
12	0.684	0.291	0.020	0.004	0.000	0.000	0.000	0.000
13	0.073	0.897	0.022	0.006	0.002	0.000	0.000	0.000
14	0.518	0.450	0.021	0.010	0.000	0.000	0.000	0.000
15	0.552	0.394	0.046	0.007	0.001	0.000	0.000	0.000
16	0.696	0.302	0.001	0.000	0.000	0.000	0.000	0.000
17	0.525	0.470	0.003	0.001	0.001	0.000	0.000	0.000
18	0.079	0.855	0.058	0.005	0.002	0.000	0.000	0.000
19	0.284	0.713	0.002	0.000	0.000	0.000	0.000	0.000
20	0.380	0.590	0.029	0.002	0.000	0.000	0.000	0.000
21	0.325	0.669	0.005	0.002	0.000	0.000	0.000	0.000
22	0.126	0.827	0.044	0.003	0.000	0.000	0.000	0.000
23	0.453	0.526	0.020	0.001	0.000	0.000	0.000	0.000
24	0.425	0.528	0.022	0.006	0.018	0.001	0.000	0.000

Tabla 5.6: Frencuencia relativa con la que cada código Γ_k es utilizado para el conjunto de 24 imágenes.

general mayores parámetros k para los códigos bidimensionales utilizados, como se puede apreciar en la tabla 5.7. La diferencia entre utilizar los códigos unidimensionales y bidimensionales se muestra en la tabla 5.8, donde AI denota que los códigos unidimensionales realizan la actualización de estadísticas luego de codificar el primer componente del par. Observar que en este caso existen ganancias incluso permitiendo la actualización en el medio para los unidimensionales, y que la ganancia parece ser mayor para la selección RS_2 que para RS_1.

Figura 5.2.1: Imagen de prueba en la cual los errores de predicción para $R - G$ y $B - G$ suelen ser grandes.

$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
0.0	0.0	0.0	0.0	0.012	0.137	0.850	0.0

Tabla 5.7: Frencuencia relativa con la que cada código Γ_k es utilizado en la figura 5.2.1.

Modo	$\mathcal{U}$	$\mathcal{B}$	$\Delta_{rel}(\%)$
(AI, RS_1)	172223	172177	-0.027
(SAI, RS_1)	172246	172177	-0.040
(AI, RS_2)	172179	172116	-0.037
(SAI, RS_2)	172194	172116	-0.045

Tabla 5.8: Resultados para la imagen de la figura 5.2.1. Aquí AI denota que hay actualización intermedia, y SIN que no.

Modo	$\mathcal{U}$	$\mathcal{B}$	$\Delta_{rel}(\%)$
(AI, RS_1)	163862	163750	-0.068
(SAI, RS_1)	163875	163750	-0.076
(AI, RS_2)	163806	163672	-0.082
(SAI, RS_2)	163830	163672	-0.096

Tabla 5.9: Resultados para la imagen de la figura 5.2.1, pero codificando los canales (R, B) en lugar de $(R - G, B - G)$.

La ganancia por utilizar los códigos bidimensionales pueden parecer relativamente pequeñas, sin embargo, el tamaño del archivo original, sin comprimir, es 192786 bytes, y el comprimido no tanto menos. Es decir, en la columna $\Delta_{rel}(\%)$ estamos normalizando por una cantidad que es relativamente bastante mayor que para el conjunto de 24 imágenes, ya que estas resultan ser más compresibles por este método. En el caso de la imagen de la figura 5.2.1, de hecho, cambiando $R - G$ y $B - G$ por R y B, y usando la misma familia de contextos que para G, se obtienen mejores resultados, que se visualizan en la tabla 5.9.

5.3. Comparación con estimaciones teóricas

En esta sección mostramos que los resultados obtenidos son en cierta manera esperados, incluso en una situación idealizada y con estimaciones algo groseras. Vamos a estimar las mejoras utilizando las siguientes consideraciones/simplificaciones:

- Suponemos que cada componente de los pares $(\epsilon_{R-G}, \epsilon_{B-G})$ es independiente y tiene distribución $TSGD$ con $d = 1/4$ (de esta forma al aplicar M tenemos una variable geométrica).
- Para los parámetros $k = 0$ y $k = 1$, ya sabemos que utilizar los códigos bidimensionales no presenta mejoras con respecto a los códigos unidimensionales, así que podemos ignorar los píxeles en los que tenemos $k = 0$ y $k = 1$.
- Para $k > 1$, como se muestra en la figura 4.3.5, la diferencia entre utilizar los códigos bidimensionales y los unidimensionales se va aproximando al límite $\tau(z, u)$ de (4.3.16), que aparece en la figura figura 4.3.4. A modo de simplificación consideramos el mínimo de $\tau(z, u)$, que es aproximadamente -0.075 bits/par, como estimación para la reducción en longitud de código por usar los códigos bidimensionales. Suponemos entonces que la ganancia al utilizar C_{2^k}, con $k > 1$, para codificar un par es $\lambda = 0.075$ bits, que por lo general esperamos que sea una sobre-estimación.

Denotamos mediante η la proporción de píxeles para los cuales se utiliza algún código bidimensional C_{2^k} con $k > 1$, y mediante $N \times M$ las dimensiones de la imagen. La mejora que estimamos es $\lambda\,\eta \cdot N \times M$ bits.

Con los datos que tenemos podemos calcular este número para las imágenes de nuestra colección de 24 imágenes. Por ejemplo, para la Imagen 1 tenemos $\eta \approx 0,0006$, que se obtiene multiplicando la suma de las frecuencias de los códigos Γ_k con $k > 1$ con la frecuencia con la que los códigos de $\mathcal{G}_2$ son utilizados. Luego la ganancia estimada es 17.7 bits $\approx$ 2 bytes, que es de dimensiones cercanas a las diferencias obtenidas en las tablas 5.3 y 5.4, si bien en el segundo caso la diferencia relativa es grande.

Es importante observar que, para $k > 1$, se tiene que la longitud para codificar un par mediante C_{2^k} es necesariamente mayor que $2 \cdot k + 1 \geq 5$ bits[b], y por lo tanto una mejora de aproximadamente 0.075 bits por par no llega a representar más de un $1.5\,\%$ del largo de $R - G$ y $B - G$ codificados conjuntamente. Considerando también la longitud de codificar G, la reducción en longitud de la secuencia comprimida resulta relativamente aún menor. Por lo tanto, la mejora en cuanto a longitud de código es necesariamente modesta, y se ve fuertemente limitada por lo acertado que resulta el predictor para el caso de $R - G$ y $B - G$.

5.4. Conclusiones

La utilización de códigos bidimensionales para pares de variables aleatorias geométricas en principio conllevan una reducción en la redundancia con respecto a los códigos de Golomb, sin alterar en gran modo la complejidad de la codificación. Los códigos de Golomb son utilizados en algoritmos de compresión de imágenes sin pérdida como LOCO-I [16] para codificar los residuos de predicción, que son bien modelados por una distribución geométrica a dos lados. Implementamos entonces modificaciones a LOCO-I agregando los códigos bidimensionales para codificar imágenes a color en el espacio RGB, codificando primero G por LOCO-I y luego los canales $(R - G, B - G)$ incorporando los códigos bidimensionales.

Los resultados empíricos obtenidos, comparados con los obtenidos utilizando únicamente los códigos unidimensionales, no resultan muy positivos. Para nuestro conjunto inicial de 24 imágenes a color, la diferencia en la longitud final de salida entre utilizar los códigos bidimensionales y los unidimensionales es casi despreciable, y no en favor de la utilización códigos bidimensionales. Para este conjunto

[b] Esto se sigue de que $M = 2k - 1$ para $k > 1$, como se muestra en el apéndice B.1, y de que se suman por lo menos dos bits de codificar los cocientes en unario.

de imágenes, la actualización de estadísticas en medio del par $(R-G, B-G)$ cuando utilizamos los códigos unidimensionales parece redundar en mayores ganancias. Como vimos, para estas imágenes un primer problema que impide las mejoras en longitud de los códigos bidimensionales es que los pares $(R-G, B-G)$ para un píxel dado su contexto resultan en general tener baja entropía, dando lugar a parámetros $k = 0$ y $k = 1$, para los cuales utilizar los códigos unidimensionales y bidimensionales es lo mismo. Para imágenes algo menos naturales y predecibles, vimos un ejemplo en el que incluso admitiendo la actualización de estadísticas en el medio del par, los códigos bidimensionales tienen un mejor desempeño, sin embargo, la diferencia es aún marginal.

Finalmente, mostramos que estos resultados son en cierto modo razonables, estimando a groso modo las ganancias. Las ganancias por la utilización de los códigos bidimensionales en lugar de los códigos unidimensionales son modestas, y en nuestro conjunto de imágenes se encuentran limitadas por el buen desempeño del predictor para los canales $R-G$ y $B-G$.

CAPÍTULO 6

Conclusión

Las distribuciones geométricas a dos lados resultan un modelo ajustado y simple para los residuos de predicción de la intensidad de un píxel basado en sus vecinos anteriores, y han sido utilizadas con éxito en el algoritmo LOCO-I/JPEG-LS [15, 16], donde los residuos se codifican mediante una subfamilia de los códigos Golomb que estudiamos en el capítulo 3. Los códigos de Golomb son óptimos para las distribuciones geométricas unidimensionales, como demostramos utilizando una técnica desarrollada en [12], que hace uso el algoritmo de Huffman que explicamos en el capítulo 2. Los códigos de Golomb tienen una codificación sencilla, especialmente para los códigos con parámetro potencia de dos, que son aprovechados por LOCO-I/JPEG-LS.

Como la codificación en bloques reduce en principio la longitud de código, en el capítulo 3 abordamos el estudio de las distribuciones geométricas bidimensionales, estudiando una familia en concreto de parámetros de la forma $q = 2^{-1/k}$. Esta familia de parámetros permite ciertas facilidades a la hora de trabajar con los pesos de las hojas para aplicar la técnica introducida en [12] como aparece en [17], llevando a la prueba de optimalidad de una familia de códigos similar a los de Golomb. La utilización de estos códigos bidimensionales ofrece una leve reducción en el largo del código por entero que intentamos explotar para la compresión de imágenes a color.

En el capítulo 4 explicamos los principios de funcionamiento del algoritmo LOCO-I/JPEG-LS para compresión de imágenes de tono de gris, que luego modificamos incorporando códigos bidimensionales para codificar imágenes a color en el espacio RGB. Utilizando los resultados asintóticos que probamos en el capítulo 3, derivamos reglas de selección para los códigos bidimensionales con parámetros potencia de dos, y consideramos entonces su utilización para la codificación de distribuciones geométricas a dos lados independientes. Esto resulta en reglas sencillas para codificar pares de variables aleatorias con distribuciones geométricas a dos lados independientes mediante la utilización de los códigos bidimensionales potencia de dos, que agregamos al algoritmo LOCO-I/JPEG-LS.

Los experimentos con el algoritmo resultante, que estudiamos en el capítulo 5, muestran en general una diferencia marginal con la utilización de los códigos unidimensionales. Mostramos entonces que el hecho de que el predictor resulte muy acertado para los canales $R - G$ y $B - G$, para los cuales utilizamos los códigos bidimensionales, impide obtener mejoras más significativas.

Apéndices

APÉNDICE A

Cálculo de perfiles para T_k

```
import math

def profile(k) :

  M = int(math.ceil(k*(k-1)/4.0)+((k*(k+1))/2))
  d = int(math.floor(math.log(M,2)))
  level_m = 2**(d+1) - M # vacantes en el nivel M
  ans = [0,0,0] # n_M , n_{M+1} y n_{M+2}

  for i in xrange(k) :
    # total de elementos de la forma q^(k+i)
    tot = k-i-1
    # 4 | (k-i-1) o 4 | (k-i) si y solo si (k-i-1)(k-i) / 2 es par
    # i.e. los elementos de firma mayor a k + i se pueden aparear perfectamente
    # en caso de no suceder esto, se necesita agregar un elemento de la forma q^(k+i)
    # para que vaya junto con estos, por lo tanto, en ese caso, restamos uno del total
    if (k-i)&3 != 1 and ((k-i)&3) != 0 :  tot -= 1
    # numero de pares [q^(k+i),q^(k+i)], observar que tienen peso q^i.
    r = tot/2
    ans[1] += 2 * min(level_m,r) # se agregan los pares al nivel M + 1
    if (r > level_m) :
      ans[2] += 2 * (r-level_m) # si se exceden, se agregan los pares al nivel M + 1
    # ajustamos el numero de lugares restantes en el nivel M
    level_m -= r
```

```
    level_m = max(0,level_m)

    # agregamos elementos de firma i al nivel M
    ans[0] += min(level_m,i+1)
    if (i + 1 > level_m) :
      ans[1] += i+1-level_m # si se exceden, se agregan al nivel M + 1
    # ajustamos el numero de lugares restantes en el nivel M
    level_m -= i + 1
    level_m = max(0,level_m)

    # notar que 2 q^(k+i+1) = q^(i+1) < w([q^(k+i),q^(k+i+1)]) < 2 q^(k+i) = q^i
    # por eso esto se checkea luego del peso i si hay algun par [q^(k+i),q^(k+i+1)]
    # el razonamiento es analogo a cuando restamos un elemento de la forma q^(k+i)
    if ((k-i)&3) != 2 and ((k-i)&3) != 1 :
      if level_m >= 1 :
        ans[1] += 2
        level_m -= 1
      else :
       ans[2] += 2

  # queda una palabra de firma k sola al final?
  if ((k*(k-1))&3) == 2 :
    if level_m >= 1 :
      ans[0] += 1
    else :
      ans[1] += 1

  return ans

for k in xrange(1,101) :
  print profile(k)
```

Código A.1: Código Python para calcular perfiles para códigos bidimensionales.

```
import math

def epsilon(k,i) :
    if (k-i)&3 != 1 and ((k-i)&3) != 0 :
        return 1
    return 0

def profile(k) :
    Q = (k*(k-1) + 2*epsilon(k,0))/4+((k*(k+1))/2)
    M = len(bin(Q))-3 # parte entera de lg(Q)
    level_m = 2**(M+1) - Q # vacantes en el nivel M
    l = -1
```

```
13      h = k
14      while h - l > 1 :
15          m = (h+l) / 2
16          if (2 * (k + 1) * (m+1) + m * (m+1) + 2*(-epsilon(k,0)  +
    epsilon(k,m+1)) ) >= 4 * level_m :
17            h = m
18          else :
19            l = m
20      level_m -= (2 * (k + 1) * (l+1) + l * (l+1) + 2*(-epsilon(k,0)
    + epsilon(k,l+1)) ) / 4
21      j = min(l+2,level_m - max(0,min(level_m,(k-l-2-epsilon(k,l+2)-
    epsilon(k,l+1))/2)))
22      c = ( (l+2) * (l+1) ) / 2 + j - 2**(M+1) + k*k
23      return [2**(M+1) - k*k+c,2*k*k-(2**(M+1))-3*c,2*c]
24
25  for k in xrange(1,101) :
26      print profile(k)
```

Código A.2: Versión significativamente mejorada.

APÉNDICE B

Pruebas de la parte práctica

B.1. Existencia de los límites

Prueba. *(de la proposición 4.3.1).* Calculamos

$$f(z) \triangleq \lim_{k\to\infty} \left(L_{z^{1/2^k}}\left(C_{2^{k+1}}\right) - L_{z^{1/2^k}}\left(C_{2^k}\right) \right).$$

Notamos que M y Q definidos en la proposición 3.2.2, para el caso $k \mapsto 2^k$, satisfacen

$$Q = 2^{k-1}(2^k+1) + 2^{k-2}(2^k-1) = 2^{2k-1} + 2^{2k-2} + 2^{k-2}\,,$$

si $k \geq 2$ y luego

$$M = 2k-1\,.$$

Como consecuencia λ_{2^k} definido en el lema 3.2.6, satisface $\lambda_{2^k} = 1$. Por lo tanto

$$\alpha = 2\sqrt{\lambda_{2^k} - 1/2} - 1 = \sqrt{2} - 1\,,$$

en el lema 3.2.6.

Ahora tenemos todos los ingredientes para aplicar el corolario 3.2.3.

Por el corolario 3.2.3, aplicado primero a $C_{2^{k+1}}$ con z^2 en lugar de z tenemos

$$L_{z^{1/2^k}}(C_{2^{k+1}}) = \frac{2}{1-z^2} + 2k + 2 + \frac{z^{2\alpha}\,\log(z^2)}{(1-z^2)^2}\Big((\alpha-1)z^2 - \alpha\Big) + \frac{z^{2\alpha} - z^2 - 1}{1-z^2} + O_z(1/2^k)\,,$$

mientras que, una aplicación directa con C_{2^k} implica

$$L_{z^{1/2^k}}(C_{2^k}) = \frac{2}{1-z} + 2k + \frac{z^{\alpha}\,\log(z)}{(1-z)^2}\Big((\alpha-1)z - \alpha\Big) + \frac{z^{\alpha} - z - 1}{1-z} + O_z(1/2^k)\,.$$

Restando desaparece el único término que varía con k (y no tiende a 0), ya que $2k+1-(2k-1)=2$. Obtenemos directamente

$$L_{z^{1/2^k}}(C_{2^{k+1}}) - L_{z^{1/2^k}}(C_{2^k}) = f(z) + O_z(1/2^k)\,,$$

probando la existencia del límite puntual.

Si definimos

$$g(z) = \frac{2}{1-z} + \frac{z^\alpha \log(z)}{(1-z)^2}\Big((\alpha-1)z-\alpha\Big) + \frac{z^\alpha - z - 1}{1-z}\,,$$

observamos que $f(z) = g(z^2) - g(z) + 2$. □

Observación B.1.1. Esto nos dice bastante más, sin embargo. Por ejemplo, si reducimos z al intervalo $[1/e, 1/\sqrt{e}]$, que es el que interesa después de todo, tenemos, como se mencionó en la observación 3.2.7 que

$$L_{z^{1/2^k}}(C_{2^{k+1}}) - L_{z^{1/2^k}}(C_{2^k}) = f(z) + O(1/2^k)\,, \tag{B.1.1}$$

para todo $z \in [1/e, 1/\sqrt{e}]$. Es decir, tenemos una convergencia uniforme bastante rápida.

Observación B.1.2. Tomando $z = z_*$, la raíz de $f(z)$, tenemos

$$L_{z_*^{1/2^k}}(C_{2^{k+1}}) - L_{z_*^{1/2^k}}(C_{2^k}) = O(1/2^k)\,.$$

¿Qué tan cerca está z_* de un cero $z_k = s_k^{2^k}$ de $f_k(z) = L_{z^{1/2^k}}(C_{2^{k+1}}) - L_{z^{1/2^k}}(C_{2^k})$? Observamos que más en general, la observación 3.2.7 es una convergencia uniforme en compactos, pero no solo en $(0,1)$, también en una bola de z_* en $\mathbb{C}$, suficientemente pequeña (para tener $0 < a \le |z| \le b < 1$ en todo punto, y evitar el branch point en 0). Luego tenemos que las derivadas $f_k'(z)$ tienden a $f'(z)$ uniformemente en compactos.

Por otro lado, por el teorema del valor medio

$$O(1/2^k) = f_k(z_*) = f_k(z_*) - f_k(z_k) = (z_* - z_k) f_k'(\theta_k)\,,$$

donde $\theta_k \in [1/2, 1/\sqrt{2}]$ y de hecho $\theta_k \to z_*$ cuando $k \to \infty$, ya que vimos que $z_k \to z_*$. Como f_k' tiende a f' uniformemente en compactos, en particular, $|f_k'(\theta_k)| \ge |f'(z_*)| - \varepsilon > 0$, para todo k suficientemente grande, ya que estamos asumiendo que $f'(z_*) \le -1 < 0$.

Por lo tanto

$$O(1/2^k) = |z_* - z_k|\,,$$

en otras palabras, los puntos de cruce concretos tienden exponencialmente hacia el ideal. Se sigue también

$$O(1/2^k) = 2^k \times |z_*^{1/2^k} - s_k|\,,$$

mostrando la utilidad de utilizar z_*, ya que $1 - s_k$ tiende a 0 como $1/2^k$.

Prueba. *(de la proposición 4.3.2).* El análisis es similar al de la prueba de la proposición 4.3.1, pero observando que $H(z^{1/2^k}) = k + \lg(e) - \lg\log(1/z) + O_z(1/k)$. □

Observación B.1.3. De hecho, podríamos más generalmente, hallar una expresión asintótica para

$$\tfrac{1}{2} L_{z^{1/k}}(C_k) - H(z^k)\,. \tag{B.1.2}$$

Esta depende del parámetro α, como en particular la proposición 3.2.4 con $z = 1/2$.

Observación B.1.4. Sin embargo, no es posible hacer una generalización análoga para el caso de

$$L_{z^{1/k}}\left(C_{k+1}\right) - L_{z^{1/k}}\left(C_k\right) . \tag{B.1.3}$$

Intuitivamente lo que ocurre es que variar z aisladamente (sin relacionarlo con k) no permite distinguir, asintóticamente, entre $z^{1/k}$ y $z^{1/(k+1)}$. En concreto, tendremos que la diferencia en (B.1.3) tenderá a 0 para cada $z \in (0,1)$.

B.2. Pruebas sobre TSGD

Comenzamos recordando que si X es una variable aleatoria discreta, con distribución $TSGD(\theta, d)$, entonces

$$\Pr\left(M(X) = n\right) = C(\theta,d) \times \theta^{\left|(-1)^n \lceil n/2 \rceil + d\right|},$$

para cada $n \in \mathbb{Z}_{\geq 0}$, donde $C(\theta,d) \triangleq \dfrac{1-\theta}{\theta^d + \theta^{1-d}}$.

Observamos entonces que si codificamos $(M(X), M(Y))$, donde X e Y son variables independientes, y con distribución $TSGD(\theta,d)$, mediante C_{2^k} tenemos una longitud esperada $L_{\theta,d}\left(\Gamma_k\right)$ que es igual a

$$L_{\theta,d}\left(\Gamma_k\right) = \sum_{i,j\geq 0} \ell\left(C_{2^k}((i,j))\right) \Pr\left(M(X) = i \text{ y } M(Y) = j\right) .$$

Escribiendo $C_{2^k}((i,j)) = T_{2^k}((i \bmod 2^k, j \bmod 2^k)) \bowtie G_1(\lfloor i/2^k \rfloor) \bowtie G_1(\lfloor j/2^k \rfloor)$ vemos que

$$\begin{aligned} L_{\theta,d}\left(\Gamma_k\right) = &\sum_{i,j\geq 0} \Big(\lfloor i/2^k \rfloor + \lfloor j/2^k \rfloor + 2\Big) \Pr\left(M(X) = i \text{ y } M(Y) = j\right) \\ &+ \sum_{i,j\geq 0} \ell\left(T_{2^k}((i \bmod 2^k, j \bmod 2^k))\right) \Pr\left(M(X) = i \text{ y } M(Y) = j\right) , \end{aligned}$$

que lo podemos escribir, como $\Pr\left(M(X) = i \text{ y } M(Y) = j\right) = \Pr(M(X) = i)\ \Pr(M(Y) = j)$, de la siguiente manera

$$\begin{aligned} L_{\theta,d}\left(\Gamma_k\right) = &\ 2 + 2\sum_{i\geq 0} \lfloor i/2^k \rfloor\ \Pr\left(M(X) = i\right) \\ &+ \sum_{i,j\geq 0} \ell\left(T_{2^k}((i \bmod 2^k, j \bmod 2^k))\right) \Pr(M(X) = i)\ \Pr(M(Y) = j) . \end{aligned}$$

Aquí calcularemos primero $A_k = \sum_{i\geq 0} \lfloor i/2^k \rfloor\ \Pr\left(M(X) = i\right)$. Para esto, partimos la suma según la división entera por 2^k, obteniendo

$$\begin{aligned} A_k &= \sum_{q=0}^{\infty} \sum_{r=0}^{2^k-1} q\ \Pr\Big(M(X) = 2^k q + r\Big) \\ &= \sum_{q=0}^{\infty} q \sum_{r=0}^{2^k-1} \Pr\Big(M(X) = 2^k q + r\Big) . \end{aligned}$$

Para calcular $\sum_{r=0}^{2^k-1} \Pr\left(M(X) = 2^k q + r\right)$ sepramos en 2 sumas, según la paridad del resto r

$$\begin{aligned}\sum_{r=0}^{2^k-1} \Pr\left(M(X) = 2^k q + r\right) &= \sum_{r=0}^{2^{k-1}-1} \Pr\left(M(X) = 2^k q + 2r\right) + \sum_{r=0}^{2^{k-1}-1} \Pr\left(M(X) = 2^k q + 2r + 1\right) \\ &= \sum_{r=0}^{2^{k-1}-1} C(\theta, d)\, \theta^{2^{k-1}q+r+d} + \sum_{r=0}^{2^{k-1}-1} C(\theta, d)\, \theta^{2^{k-1}q+r+1-d}\end{aligned}$$

donde utilizamos que $d \in [0, 1/2]$, al calcular las probabilidades en la segunda suma. Primero factorizando, y luego sumando las geométricas en r, deducimos

$$\begin{aligned}\sum_{r=0}^{2^k-1} \Pr\left(M(X) = 2^k q + r\right) &= C(\theta, d) \cdot \theta^{2^{k-1}q} \cdot \left(\theta^d + \theta^{1-d}\right) \times \left(\sum_{r=0}^{2^{k-1}-1} \theta^r\right) \\ &= C(\theta, d) \cdot \theta^{2^{k-1}q} \cdot \left(\theta^d + \theta^{1-d}\right) \times \frac{1 - \theta^{2^{k-1}}}{1 - \theta} \\ &= \theta^{2^{k-1}q} \times \left(1 - \theta^{2^{k-1}}\right).\end{aligned}$$

Volviendo a A_k, tenemos

$$\begin{aligned}A_k &= \sum_{q=0}^{\infty} q \sum_{r=0}^{2^k-1} \Pr\left(M(X) = 2^k q + r\right) \\ &= \sum_{q=0}^{\infty} q \theta^{2^{k-1}q} \times \left(1 - \theta^{2^{k-1}}\right) \\ &= \frac{\theta^{2^{k-1}}}{\left(1 - \theta^{2^{k-1}}\right)^2} \times \left(1 - \theta^{2^{k-1}}\right) \\ &= \frac{\theta^{2^{k-1}}}{1 - \theta^{2^{k-1}}}.\end{aligned}$$

Por lo tanto

$$L_{\theta,d}(\Gamma_k) = \frac{2}{1 - \theta^{2^{k-1}}} + \sum_{i,j \geq 0} \ell\left(T_{2^k}((i \bmod 2^k, j \bmod 2^k))\right) \Pr(M(X) = i)\, \Pr(M(Y) = j), \tag{B.2.1}$$

siempre que $k \geq 1$. Agrupando términos según $i \bmod 2^k$ y $j \bmod 2^k$ tenemos

$$L_{\theta,d}(\Gamma_k) = \frac{2}{1 - \theta^{2^{k-1}}} + \sum_{0 \leq i,j < 2^k} \ell\left(T_{2^k}((i \bmod 2^k, j \bmod 2^k))\right) \Pr(M(X) \equiv i \bmod 2^k)\, \Pr(M(X) \equiv j \bmod 2^k). \tag{B.2.2}$$

Aquí observamos, por un argumento similar, que

$$\Pr(M(X) \equiv i \bmod 2^k) = \frac{C(\theta, d)}{1 - \theta^{2^{k-1}}} \cdot \theta^{\lfloor i/2 \rfloor + \omega_d(i)}, \tag{B.2.3}$$

donde

$$\omega_d(i) = \begin{cases} \theta^d, & \text{si } i \text{ es par}; \\ \theta^{1-d}, & \text{si } i \text{ es impar}. \end{cases}$$

Prueba. *(del lema 4.3.2).* Por (B.2.2) y (B.2.3) observamos que

$$L_{\theta,d_1}(\Gamma_k) - L_{\theta,d_2}(\Gamma_k) = \left(1-\theta^{2^{k-1}}\right)^{-2} \sum_{0\le i,j<2^k} \ell_{i,j}\theta^{\lfloor\frac{i}{2}\rfloor+\lfloor\frac{j}{2}\rfloor}\left(C(\theta,d_1)^2\theta^{\omega_{d_1}(i)+\omega_{d_1}(j)} - C(\theta,d_2)^2\theta^{\omega_{d_2}(i)+\omega_{d_2}(j)}\right),$$

donde $\ell_{i,j} \triangleq \ell\left(T_{2^k}((i \bmod 2^k, j \bmod 2^k))\right) - M$ con M definido en la proposición 3.2.2, siendo $M = 2k-1$ para este caso particular, como vimos en el apéndice B.1.

Por definición de $C(\theta,d)$ reescribimos lo de arriba como

$$L_{\theta,d_1}(\Gamma_k) - L_{\theta,d_2}(\Gamma_k) = \left(\frac{1-\theta}{1-\theta^{2^{k-1}}}\right)^2 \sum_{0\le i,j<2^k} \ell_{i,j}\theta^{\lfloor\frac{i}{2}\rfloor+\lfloor\frac{j}{2}\rfloor}\left(\frac{\theta^{\omega_{d_1}(i)+\omega_{d_1}(j)}}{\left(\theta^{d_1}+\theta^{1-d_1}\right)^2} - \frac{\theta^{\omega_{d_2}(i)+\omega_{d_2}(j)}}{\left(\theta^{d_2}+\theta^{1-d_2}\right)^2}\right). \tag{B.2.4}$$

Ahora discutimos según las paridades de i y j, para poder acotar el factor

$$h_{i,j}(\theta) \triangleq \left(\frac{\theta^{\omega_{d_1}(i)+\omega_{d_1}(j)}}{\left(\theta^{d_1}+\theta^{1-d_1}\right)^2} - \frac{\theta^{\omega_{d_2}(i)+\omega_{d_2}(j)}}{\left(\theta^{d_2}+\theta^{1-d_2}\right)^2}\right). \tag{B.2.5}$$

a) Caso $i \not\equiv j (\bmod .2)$. En este caso notamos que $\omega_d(i)+\omega_d(j) = 1$ para todo d, y tenemos entonces

$$h_{i,j}(\theta) = \frac{\theta}{\left(\theta^{d_1}+\theta^{1-d_1}\right)^2} - \frac{\theta}{\left(\theta^{d_2}+\theta^{1-d_2}\right)^2}.$$

Observamos ahora que la derivada de $A(d) = \frac{\theta}{\left(\theta^d+\theta^{1-d}\right)^2}$ respecto a d es

$$\partial_d A = -\frac{\left(2\,\theta^{4\,d+1} - 2\,\theta^{2\,d+2}\right)\log(\theta)}{3\,\theta^{4\,d+1} + 3\,\theta^{2\,d+2} + \theta^{6\,d} + \theta^3},$$

que se puede acotar, como $0 \le d \le 1/2$ y así $6d \le 4d+1 \le 2d+2 \le 3$, mediante

$$|\partial_d A| \le \frac{2\,\theta^{4\,d+1}\log(1/\theta)}{3\,\theta^3 + 3\,\theta^3 + \theta^3 + \theta^3} = \frac{\theta^{4d-2}}{4}\log(1/\theta).$$

Ahora, por el teorema del valor medio tenemos

$$h_{i,j}(\theta) = (d_1-d_2)\,\partial_d A|_{d=d'}$$

para un cierto d' entre d_1 y d_2. Entonces

$$\left|h_{i,j}(\theta)\right| \le |d_1-d_2| \times \frac{\theta^{4d'-2}}{4}\log(1/\theta),$$

y, al ser $0 \le d'$, concluímos que

$$\left|h_{i,j}(\theta)\right| \le |d_1-d_2| \times \frac{\log(1/\theta)}{4\theta^2}. \tag{B.2.6}$$

b) Caso $i \equiv j (\bmod .2)$. En este caso notamos que $\omega_d(i)+\omega_d(j) = 2d$ si ambos i y j son pares, y $\omega_d(i)+\omega_d(j) = 2-2d$ en caso que ambos sean impares. Como podemos pasar de un caso al otro con la transformación $(d_1,d_2) \mapsto (1-d_1, 1-d_2)$ aplicada al lado derecho de (B.2.5), nos basta con calcular la cota en uno de los casos, y luego aplicarle la transformación a $(d_1,d_2) \mapsto (1-d_1,1-d_2)$ a la cota, que, en última instancia, resulta invariante por dicha transformación.
Suponemos entonces que $\omega_d(i)+\omega_d(j) = 2d$. Realizando un procedimiento análogo al del item anterior, pero con $A(d) = \left(\frac{\theta^d}{\theta^d+\theta^{1-d}}\right)^2$, obtenemos

$$\left|h_{i,j}(\theta)\right| \le |d_1-d_2| \times \frac{\log(1/\theta)}{2\theta^2}. \tag{B.2.7}$$

Por (B.2.6) y (B.2.7), concluímos que tenemos

$$\left|h_{i,j}(\theta)\right| \leq |d_1 - d_2| \times \frac{\log(1/\theta)}{2\theta^2}\,, \tag{B.2.8}$$

para todo $(i,j) \in \{0,\ldots,2^k-1\} \times \{0,\ldots,2^k-1\}$.

Volviendo a (B.2.4), utilizando la desigualdad triangular, junto con $\ell_{i,j} \leq 2$ y $\theta \in (0,1)$, tenemos

$$\left|L_{\theta,d_1}(\Gamma_k) - L_{\theta,d_2}(\Gamma_k)\right| \leq \left(\frac{1-\theta}{1-\theta^{2^{k-1}}}\right)^2 \sum_{0\leq i,j<2^k} 2|h_{i,j}(\theta)|\,,$$

utilizando (B.2.8) obtenemos

$$\left|L_{\theta,d_1}(\Gamma_k) - L_{\theta,d_2}(\Gamma_k)\right| \leq |d_1 - d_2| \times \left(\frac{1-\theta}{1-\theta^{2^{k-1}}}\right)^2 2^{2k}\frac{\log(1/\theta)}{\theta^2}\,. \tag{B.2.9}$$

Observar ahora que $\dfrac{1-\theta^{2^{k-1}}}{1-\theta} = 1+\theta+\ldots+\theta^{2^{k-1}-1}$. Se sigue entonces que $\dfrac{1-\theta^{2^{k-1}}}{1-\theta} \geq 2^{k-1} \times \theta^{2^{k-1}}$, y, por lo tanto

$$\left(\frac{1-\theta}{1-\theta^{2^{k-1}}}\right)^2 \leq 4\theta^{-2^k} \times 2^{-2k}\,.$$

Usando esto en (B.2.9), concluímos que

$$\left|L_{\theta,d_1}(\Gamma_k) - L_{\theta,d_2}(\Gamma_k)\right| \leq 4\,|d_1 - d_2| \times \theta^{-2^k}\,\frac{\log(1/\theta)}{\theta^2}\,,$$

que prueba nuestro lema. □

Bibliografía

[1] COVER, Thomas M., THOMAS, Joy A. *Elements of Information Theory*. 2da ed. New Jersey: Wiley–Interscience, 2006. ISBN-13: 978-0-471-24195-9

[2] BIGGS, Norman L. *Codes: An Introduction to Information Communication and Cryptography*. 1ra ed. London: Springer, 2008. ISBN-13: 978-1-848-00272-2

[3] ASH, Robert B. *Information Theory*. 1ra ed. New York: Dover Publications, 1965. ISBN-13: 978-0-486-66521-4

[4] RISSANEN, Jorma. *Information and Complexity in Statistical Modeling*. 1ra ed. New York: Springer, 2007. ISBN-13: 978-0-387-36610-4.

[5] HOGGARD, S.G. *Mathematics of Digital Images: Creation, Compression, Restoration, Recognition*. 1ra ed. New York: Cambridge University Press, 2006. ISBN-13: 978-0-521-78029-2

[6] HUFFMAN, D.A. *A method for the construction of minimum redundancy codes*. Proceedings of the IRE, September 1952, vol 40, no 9, pp. 1098–1101.

[7] LINDER, T., TAROKH, V., ZEGER, K. *Existence of optimal prefix codes for infinite source alphabets*. IEEE Transactions on Information Theory. 1997, vol 43, no 6, pp. 2026–2028.

[8] HUMBLET, P.A. *Optimal source coding for a class of integer alphabets*. IEEE Transactions on Information Theory. 1978, vol 24, no 1, pp. 110–112.

[9] RADHAKRISHNAN, J. *Entropy and Counting*. IIT Kharagpur, Golden Jubilee Volume on Computational Mathematics, Modelling and Algorithms. New Delhi: Narosa Publishers, 2001.

[10] KATO, A., HAN, T.S., NAGAOKA, H. *Huffman coding with an infinite alphabet*. IEEE Transactions on Information Theory. 1996, vol 42, no 3, pp. 977–984.

[11] GOLOMB, S.W. *Run length encodings*. IEEE Transactions on Information Theory. 1966, vol 12, no 3, pp. 399–401.

[12] GALLAGER, R.G., VAN VOORHIS, D.C. *Optimal source codes for geometrically distributed integer alphabets*. IEEE Transactions on Information Theory. 1975, vol 21, no 2, pp. 228–230.

[13] SZPANKOWSKI, W. *Asymptotic Average Redundancy of Huffman (and Other) Block Codes*. IEEE Transactions on Information Theory. 2000, vol 46, no 7, pp 2434–2443.

[14] HOWARD, P., VITTER, J. *Fast and efficient lossless image compression*. Data Compression Conference. Snowbird, UT: IEEE, March 1993. pp. 351–360.

[15] WEINBERGER, M.J., SEROUSSI, G., SAPIRO, G. *LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm*, Data Compression Conference. Snowbird, UT: IEEE, 1996. pp. 140–149.

[16] WEINBERGER, M.J., SEROUSSI, G., SAPIRO, G. *The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS*. IEEE Transactions on Information Theory. 2000, vol 9, no 8, pp. 1309–1324.

[17] BASSINO, F., CLÉMENT, J., SEROUSSI, G., VIOLA, V. *Optimal Prefix Codes for Pairs of Geometrically Distributed Random Variables*. IEEE Transactions on Information Theory. 2013, vol 59, no 4, pp. 2375–2395.

[18] MERHAV, N., SEROUSSI, G., WEINBERGER, M.J. *Optimal Prefix Codes for Sources with Two-Sided Geometric Distributions*. IEEE Transactions on Information Theory. 2000, vol 46, no 1, pp. 121–135.

[19] RISSANEN, J. *Universal coding, information, prediction, and estimation*. IEEE Transactions on Information Theory. 1984, vol 30, no 4, pp. 629–636.

[20] RISSANEN, J., LANGDON, G.G. *Arithmetic Coding*. IBM Journal of Research and Development. 1979, vol 23, no 2, 146–162.

[21] KNUTH, D.E. *Dynamic Huffman Coding*. Journal of Algorithms. 1985, vol 6, no 2, pp. 163–180.

[22] CARPENTIERI, B., WEINBERGER, M.J., SEROUSSI, G. *Lossless Compression of Continuous-Tone Images*. Proceedings of the IEEE. 2000, vol 88, no 11, pp. 1797–1809.

[23] RICE, R.F. *Some practical universal noiseless coding techniques - parts I and III*. Pasadena, CA: Jet Propulsion Laboratory, Mar. 1979 and Nov. 1991. JPL-79-22 and JPL-91-3.

[24] ABRAHAMS, J. *Huffman-Type Codes for Infinite Source Distributions*. Data Compression Conference. Snowbird, UT: IEEE, 1994. pp. 83–89.

[25] LIMB, J.O., NETRAVALI, A. *Picture coding: A review*. Proceedings of the IEEE. 1980, vol 68, no 3, pp. 366–406.