

Tema 1: Bases de la Compresión Multimedia

Rafael Molina

Depto. de Ciencias de la Computación
e Inteligencia Artificial
Universidad de Granada

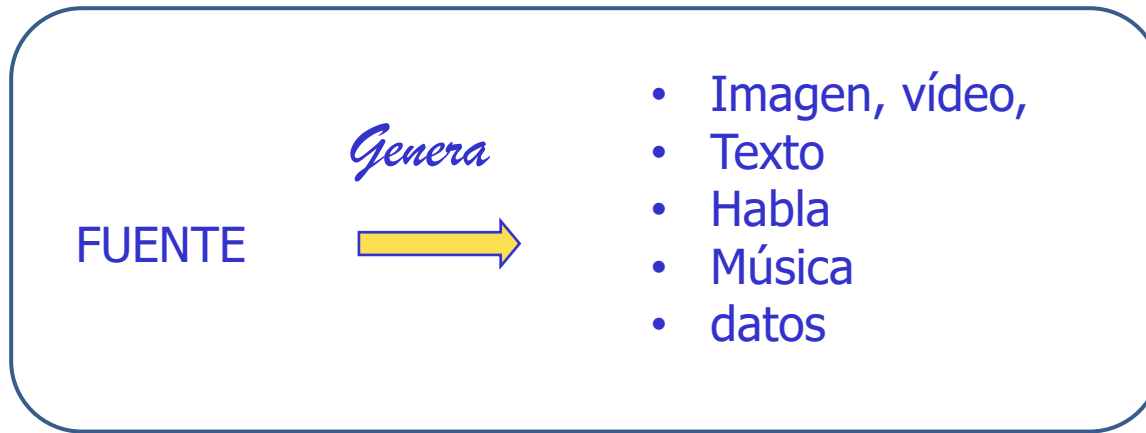
Contenidos

- I. Objetivos del tema
- II. ¿Cómo medimos la información?
- III. Breve introducción a la Teoría de la Información (Entropía)
 - I. Algunos ejemplos de cálculo de la entropía
- IV. Modelos para la compresión
 - I. Modelos probabilísticos simples
 - II. Modelos de Markov
 - III. Modelo de fuente compuesta
- V. Más sobre modelización
- VI. Codificación
 - Códigos de longitud fija y longitud variable
 - Códigos decodificables de modo único
 - Test para decodificación única
 - Códigos prefijo
 - Decodificación de una secuencia de palabras en código prefijo
 - Desigualdad de Kraft-McMillan
- VII. Consideraciones finales
- VIII. Bibliografía

I. Objetivos del tema

Estudiaremos las bases teóricas que nos permiten abordar la compresión de datos multimedia

II. ¿Cómo medimos la información?



Una fuente es un lugar donde se genera información:

Puede ser continua, discreta, multidimensional, multimodal,

- Una fuente X es una “caja” que produce símbolos
- Cada valor de X es la realización de una variable aleatoria x_i que toma valores de un conjunto M de posibilidades (símbolos)

$$A = \{a_1, a_2, \dots, a_M\}$$

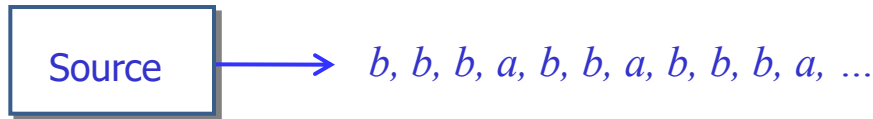
A recibe el nombre de **alfabeto**, y sus elementos son **símbolos** o **letras**

- Cada símbolo tiene una probabilidad de ocurrir
- El conjunto de símbolos ordenados producidos por la fuente X forma un **proceso estocástico**: $x_1, x_2, \dots, x_{n-1}, x_n, x_{n+1}, x_{n+2}, \dots$

Ejemplos

- Una fuente binaria $A = \{a, b\}$: $P(a)=P(X=a)=p_a$, $P(b)=P(X=b)=p_b$

Ejemplo: $p_a=0.2$, $p_b=0.8$



- Una imagen de niveles de gris

$A = \{0, 1, \dots, 255\}$:

$P(a_i) = 1/256$



- La luz de un semáforo

Nota: hay un problema subyacente importante en todos los ejemplos: ¿de dónde sacamos estas probabilidades?

Rafael Molina



$p_R=0.50$



$p_Y=0.09$



$p_G=0.40$

roto

$p_B=0.01$

Bases de la compresión multimedia

...GYRGYR...GYR...

- El objetivo de la codificación de una fuente es reducir el **número de dígitos binarios (bits)** usados para representar una señal
- **¿Cuál es la cantidad mínima de bits necesaria para describir una señal sin pérdida de información?**

» La Teoría de la Información proporciona el marco matemático que permite responder a ésta y a otras preguntas.

» La Teoría de la Información es una rama de las matemáticas relacionada con la cuantificación de la información.

» La piedra angular que estableció la disciplina fue la publicación del trabajo de Claude E. Shannon "A Mathematical Theory of Communication" (Bell System Technical Journal 1948).



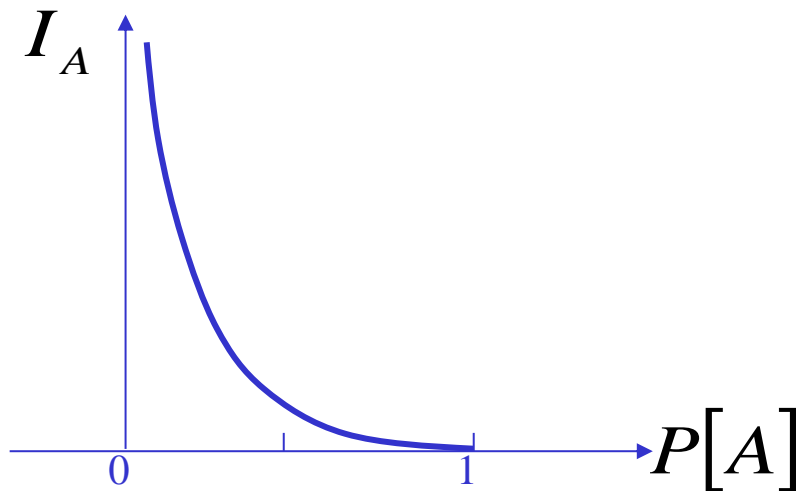
(1916-2001)

- Dado un suceso aleatorio que ocurre con una cierta probabilidad, ¿cómo definimos la información asociada con el suceso?
- Intuitivamente, la información asociada con un suceso debe cumplir las siguientes condiciones :
 - Debería ser positiva,
 - La probabilidad de un suceso debería estar relacionada con la información contenida en el suceso. Además cuánto más probable sea un evento menos información tendrá,
 - Dos sucesos independientes deberían contener la misma cantidad de información cuando aparecen juntos o separados.

- Consideremos un suceso A que ocurre con probabilidad $P(A)$
- Una medida que cumple las condiciones anteriores es la **auto-información** asociada a A definida mediante

$$I(A) = \log \frac{1}{P(A)} = -\log P(A)$$

El logaritmo será siempre entendido en base dos.



Veamos las propiedades de la auto-información (y si coinciden con nuestra intuición)

1. Recordemos que $\log(1)=0$ y la función $-\log(x)$ logaritmo crece cuando x se mueve de uno a cero. Por tanto, **a mayor probabilidad de un suceso menor auto-información asociada.**
2. De Sherlock Holmes: el ladrido de un perro durante un robo es un suceso muy probable y por tanto contiene poca información, sin embargo el que un perro no ladre durante un robo es muy poco probable y contiene mucha información.
3. Propiedad de la auto-información: La auto-información de dos sucesos A y B que son independientes es la suma las auto-informaciones de A y B .

Demostración:

$$i(AB) = \log \frac{1}{P(AB)} = -\log P(AB) \text{ [indep.]}$$
$$= -\log P(A) - \log P(B) = i(A) + i(B)$$

La cantidad de información depende de la base del logaritmo. Si la base del logaritmo es 2 la unidad es bits, si es e la unidad es nats y si es 10 la unidad se llama hartleys.

Ejemplo:

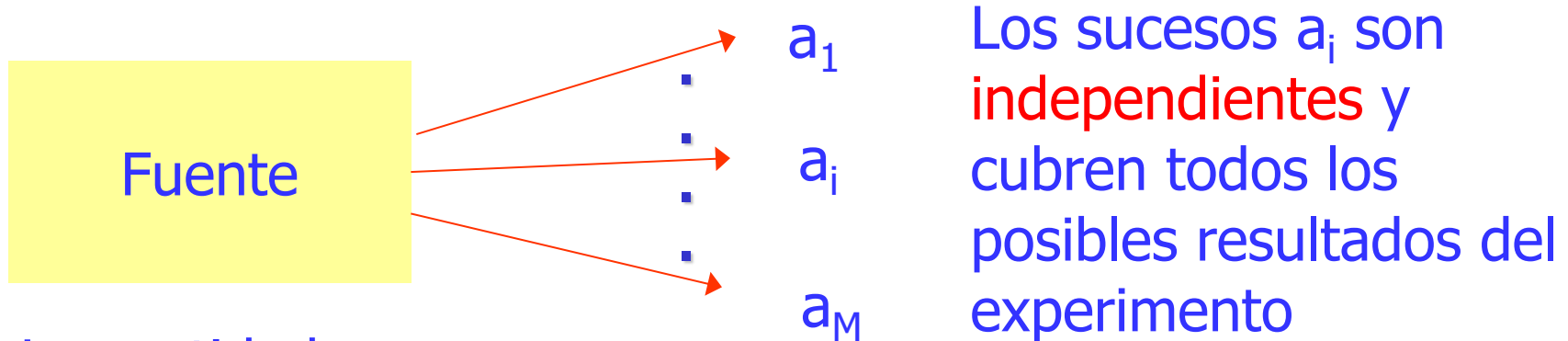
Sean cara (C) y cruz (X) los posibles resultados de lanzar una moneda. Si la moneda no está sesgada tendremos

$$P(C) = P(X) = \frac{1}{2} \quad \text{e} \quad i(C) = i(X) = 1 \text{ bit}$$

Si la moneda está cargada la información asociada a cada suceso es diferente. Por ejemplo, si

$$P(C) = \frac{1}{8}, \quad P(X) = \frac{7}{8} \quad i(C) = 3 \text{ bits} \quad i(X) = 0.193 \text{ bits}$$

III. Breve introducción a la Teoría de la Información (Entropía)



La cantidad

$$H = \sum_{k=1}^M P(a_k) i(a_k) = - \sum_{k=1}^M P(a_k) \log P(a_k)$$

Recibe el nombre de **entropía de primer orden** asociada al experimento (fuente)

X va a denotar una variable aleatoria que observa el suceso a_i con probabilidad $P(a_i)$. Usaremos $H(X)$ o sólo H si está claro el contexto.

Para cualquier variable aleatoria (fuente) X con M valores posibles

- La entropía está acotada: $0 \leq H(X) \leq \log_2 M$

- Si todos los sucesos son igualmente probables:

de donde deducimos, mira el primer resultado, que la entropía es máxima cuando todos los sucesos tienen la misma probabilidad. Máxima incertidumbre.

$$H(X) = \log_2 M$$

- Si X es determinista:

$$H(X) = 0$$

en otras palabras, no hay incertidumbre

¿Sabrías probar estos resultados?

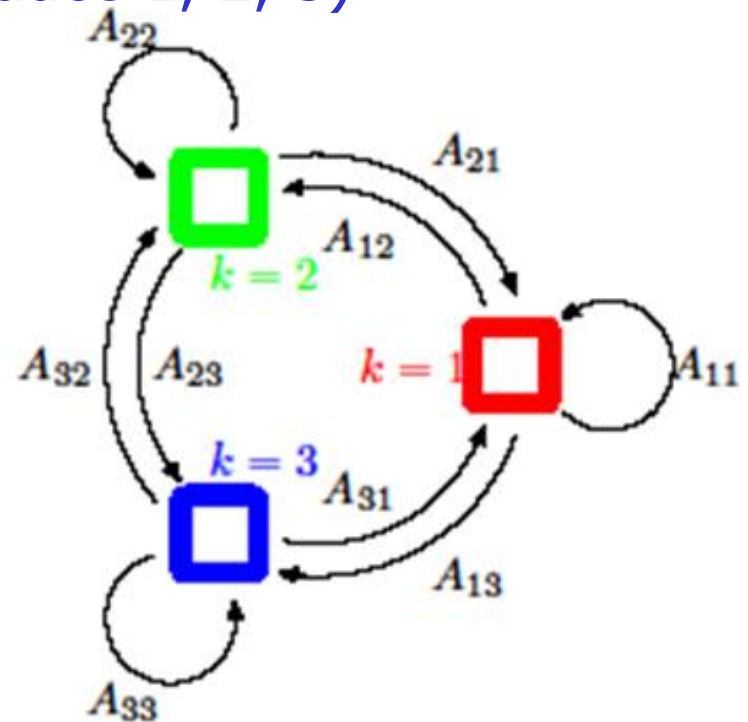
Veamos distribuciones en las que las variables están relacionadas. Con un ejemplo

Un ejemplo de modelo de estados discreto. La variable X_1 puede tomar tres valores (estar en tres estados 1, 2, 3)

$$\Pr(X_1=e)=1/3 \quad e=1,2,3$$

y las probabilidades condicionadas

$$\Pr(X_i=k|X_{i-1}=j)=A_{jk} \quad \sum_{k=1}^3 A_{jk} = 1$$



Piensa en generar secuencias infinitas de símbolos con estas probabilidades

Dada una fuente S que tiene un alfabeto $\mathbf{A}=\{a_1, a_2, \dots, a_M\}$ que genera una sucesión $\{x_1, x_2, \dots\}$ de símbolos del alfabeto su **razón de entropía** viene dada por

$$h(\mathbf{S}) = \lim_{n \rightarrow \infty} \frac{1}{n} G_n \quad (1)$$

donde

Observa: nos dará el número medio de bits por símbolo (letra)

$$G_n = - \sum_{x_1=1} \dots \sum_{x_n} P(x_1, \dots, x_n) \log P(x_1, \dots, x_n)$$

y $\{x_1, \dots, x_n\}$ es una sucesión de longitud n de la fuente S .

Shannon probó que si una fuente produce un conjunto de símbolos (letras) que tienen unas determinadas probabilidades:

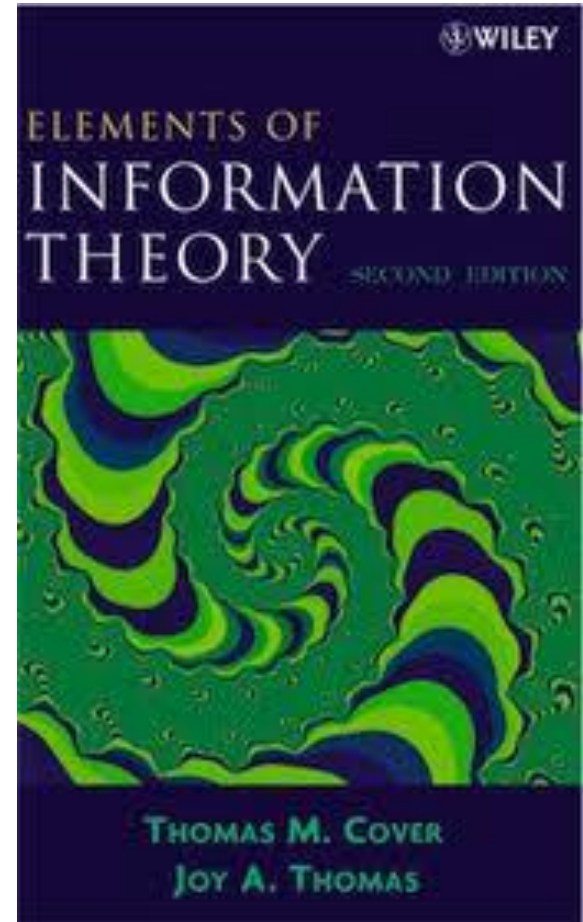
La entropía es el número medio de símbolos binarios necesarios para codificar la salida de la fuente.

Además probó también que:

lo mejor que un esquema de compresión sin pérdida puede hacer es codificar la salida de una fuente (experimento) con un número medio de bits igual a la entropía de la fuente.

Es muy interesante estudiar las propiedades de la entropía (razón de entropía) y analizar su convergencia.

Este estudio no lo haremos aquí. Si estás interesado en aprender más, lee los capítulos 4 y 5 del libro de Cover y Thomas.



Supongamos que las variables X_i son independientes e idénticamente distribuidas (iid) y notemos su distribución común mediante X . Puede probarse fácilmente que

$$G_n = -n \sum_{i=1}^M P(a_i) \log P(a_i)$$

y la ecuación para la razón de entropía se convierte en

$$h(\mathbf{S}) = - \sum_{i=1}^M P(a_i) \log P(a_i) \quad (2)$$

En general las entropías definidas en (1) y (2) no coinciden por lo que (2), como hemos indicado, recibe el nombre de *entropía de primer orden* y (1) *razón de entropía*.

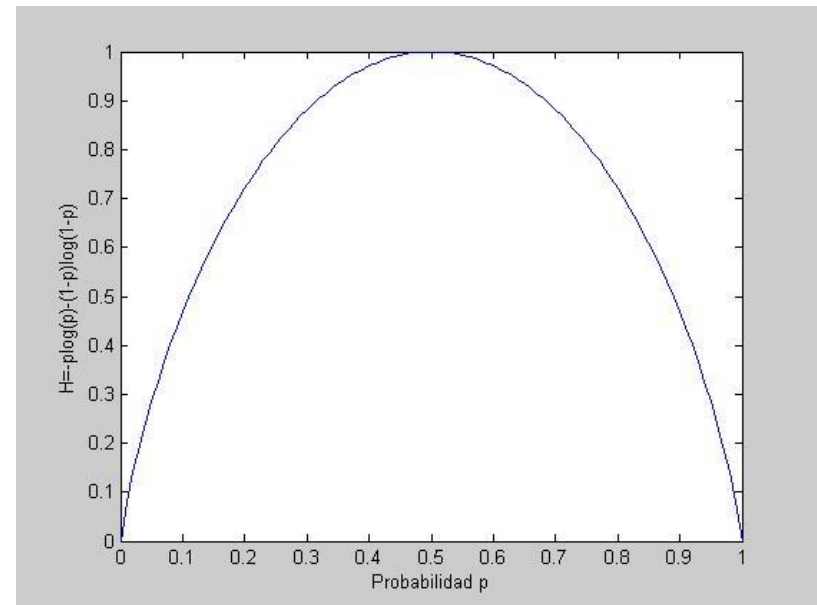
Entendamos un poco más el teorema de Shannon.

Consideremos una fuente binaria con sucesos a_1 y a_2 que cumplen

$$P(a_1)=p \text{ y } P(a_2)=1-p \quad 0 \leq p \leq 1$$

Su entropía vale $H(p)=-p\log(p)-(1-p)\log(1-p)$ (ver gráfico)

1. Una codificación que le asigne un dígito binario (por ejemplo el cero) a a_1 y el otro dígito binario (el uno) a a_2 , sólo será óptima si $p=1/2$.
2. Si $p \neq 1/2$, la entropía, $H(p)$, es siempre menor que uno y el teorema de Shannon nos dice que podemos hacerlo mejor que codificar a_1 y a_2 con un bit cada uno (al menos, teóricamente)



III.1 Algunos ejemplos de cálculo de entropía

Supongamos que tenemos una fuente cuyas realizaciones son independientes con tres posibles resultados a_1 , a_2 y a_3 con

$$P(A_1) = \frac{1}{2} \quad P(A_2) = P(A_3) = \frac{1}{4}$$

La entropía de esta fuente vale

$$H = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 = 1.5 \text{ bits}$$

Un esquema de codificación que alcanza esta cota inferior es:

$A_1 \longrightarrow 0$

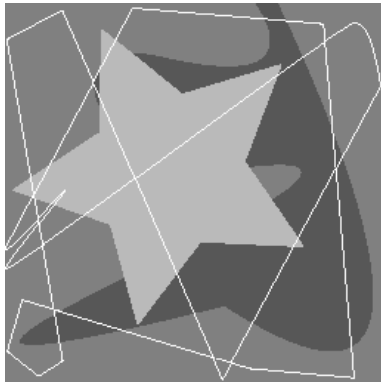
$A_2 \longrightarrow 10$

$A_3 \longrightarrow 11$

Sin embargo, si la fuente genera los símbolos $A_2A_2A_2A_2$ es obvio que para esta secuencia en particular podríamos haber elegido otra codificación mejor. El teorema de Shannon habla de comportamiento medio.

Es muy importante entender bien la cota inferior que supone la entropía

Podemos construir un modelo de fuente basado en la frecuencia relativa de cada uno de los niveles de gris en la imagen. Es decir, el histograma de la imagen suponiendo que las realizaciones son independientes



Nivel de gris	probabilidad
87	0.25
128	0.47
186	0.25
255	0.03
Otros	0

La entropía de la fuente sería

$$h(M_2) = H(M_2) = -[0.25 \log 0.25 + 0.47 \log 0.47 + 0.25 \log 0.25 + 0.03 \log 0.03]$$

$$h(M_2) = H(M_2) = 1.6614 \text{ bits / símbolo}$$

Fíjate que todavía no hemos abordado preguntas muy importante: ¿Existe algún código (asignación de secuencias de 0 y 1 a los niveles de gris) que alcance esta entropía?, ¿Podemos hacer algo con los datos (símbolos) para reducir esta entropía?



512x512

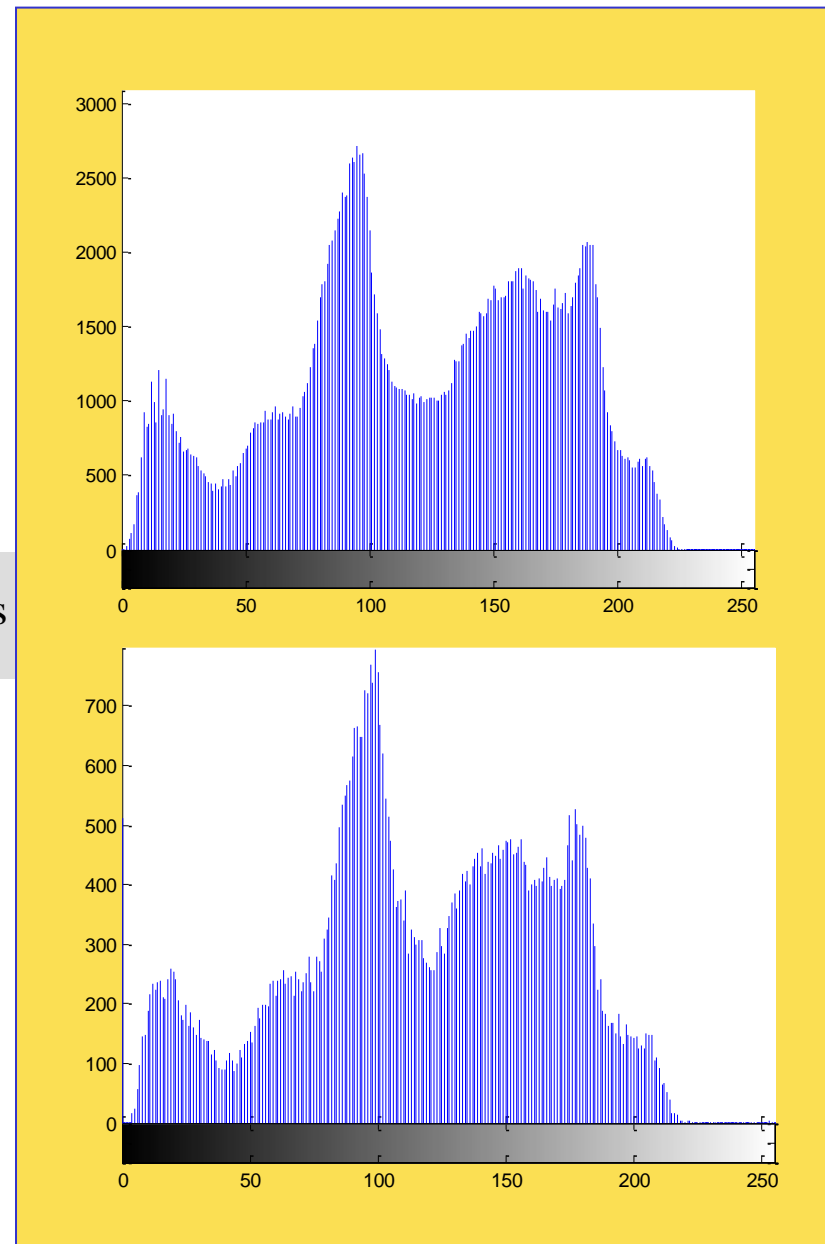
$H = 7.5925$ bits/pixel



256x256

$H = 7.5597$ bits/pixel

$$H = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} \text{ bits}$$





512x512

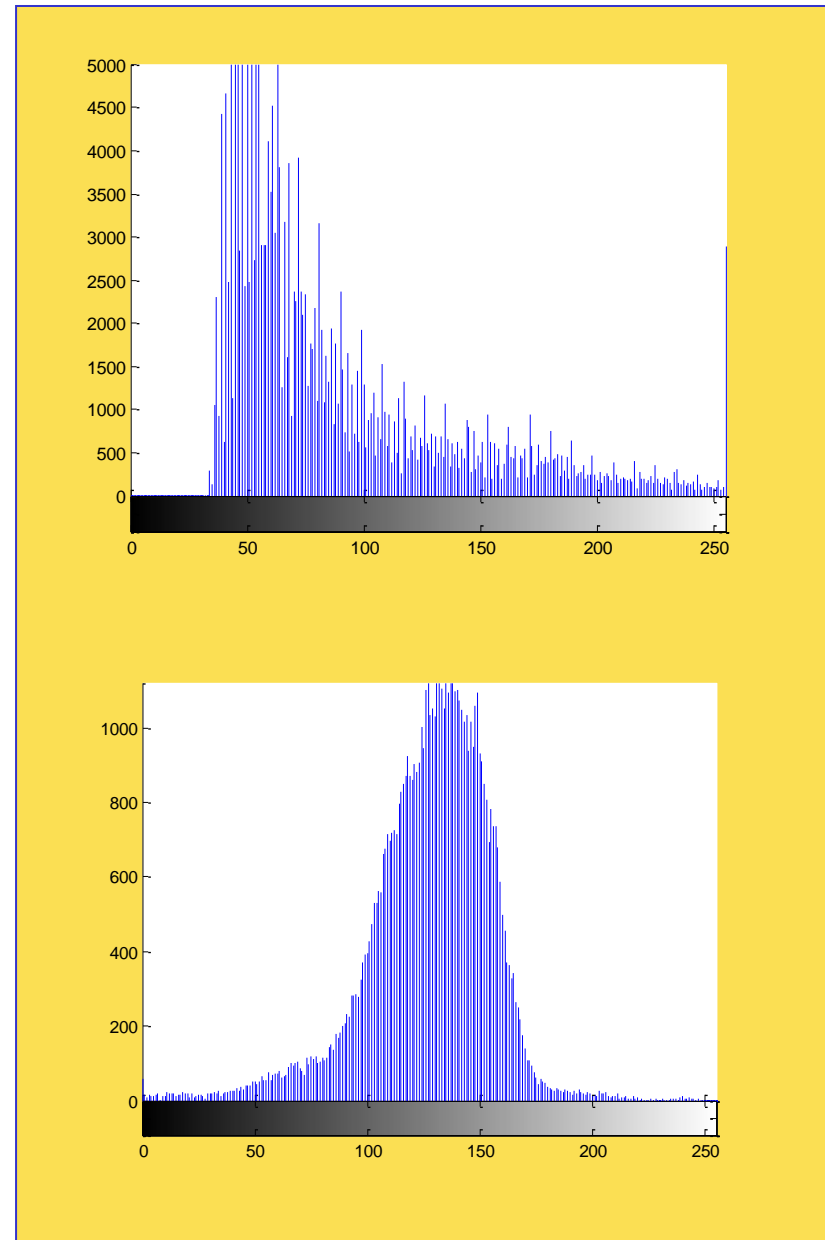
$H = 6.7893$ bits/pixel

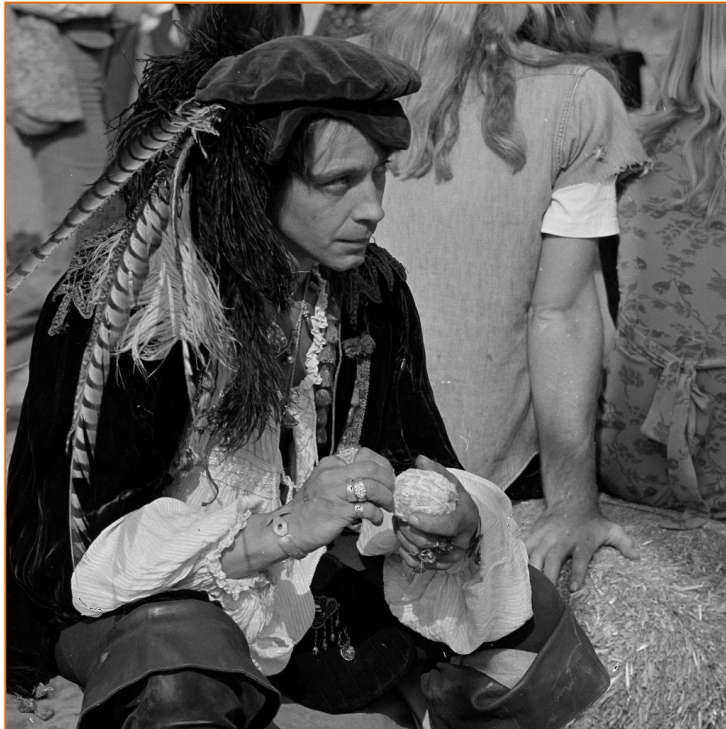


256x256

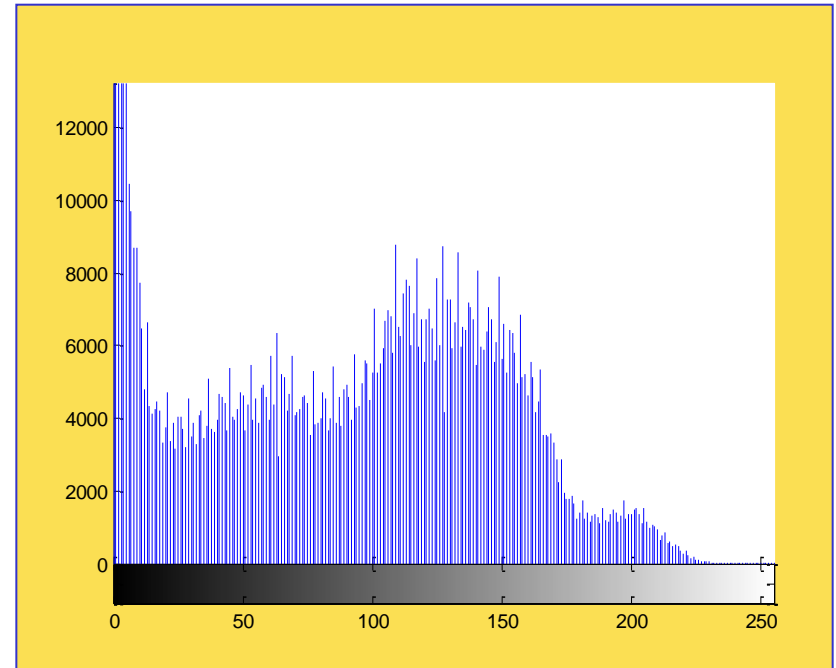
$H = 6.7093$ bits/pixel

$$H = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} \text{ bits}$$





1024x1024



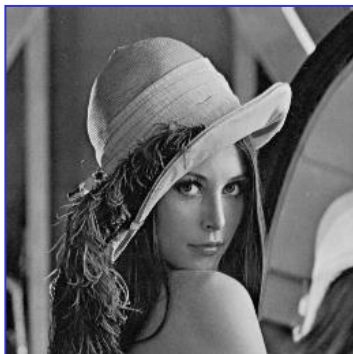
$H = 7.5237$ bits/pixel

$$H = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} \text{ bits}$$



512x512

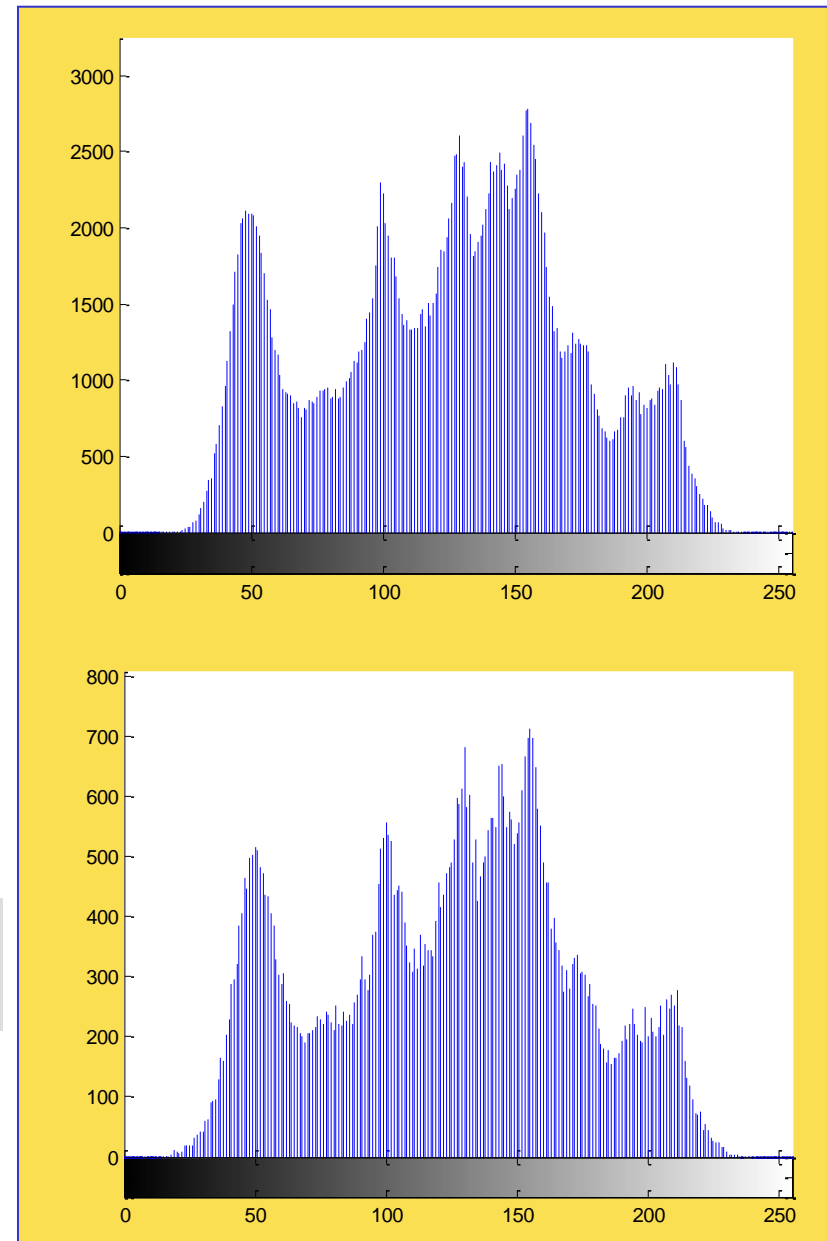
$H = 7.4474$ bits/pixel



256x256

$H = 7.4453$ bits/pixel

$$H = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} \text{ bits}$$





64x64

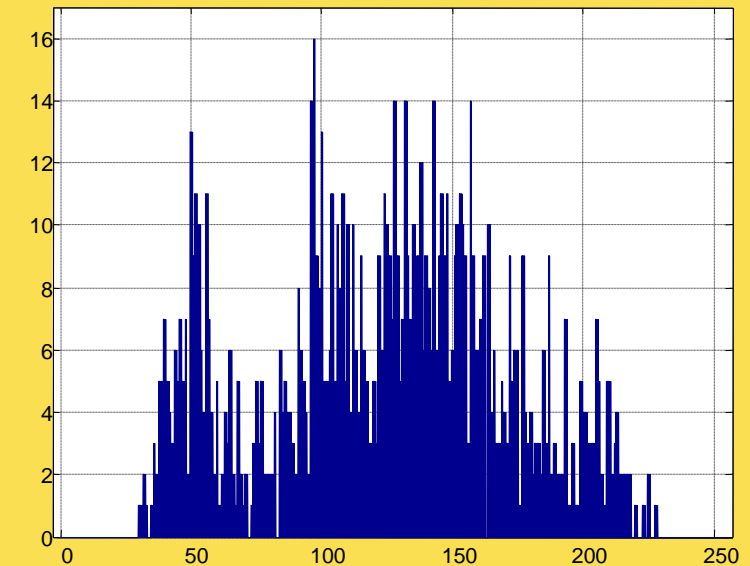
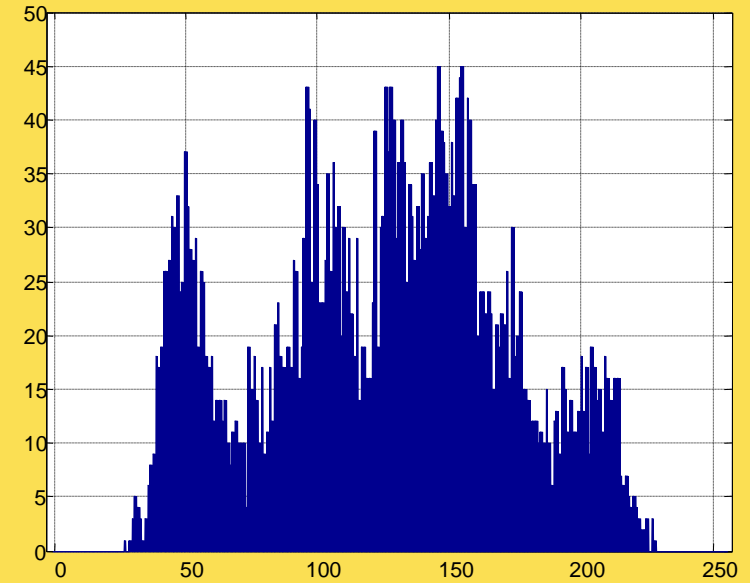
$H = 7.4103$ bits/pixel

$$H = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} \text{ bits}$$



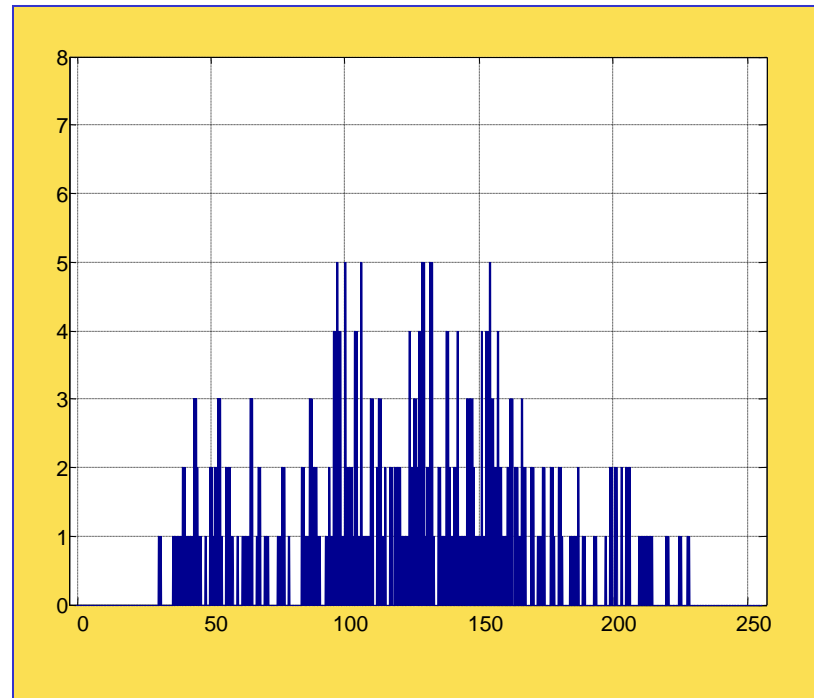
32x32

$H = 7.277$ bits/pixel





16x16



$H = 6.88$ bits/pixel

$$H = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} \text{ bits}$$



Tamaño	Entropía
256*256	7.2074
128*128	7.1956
64*64	7.1740
32*32	7.0357
16*16	6.6282
8*8	5.6014
4*4	3.8750
2*2	2



Tamaño	Entropía
256*256	7.4918
128*128	7.4659
64*64	7.4455
32*32	7.3669
16*16	6.9288
8*8	5.7561
4*4	4
2*2	2



Tamaño	Entropía
256*256	7.5931
128*128	7.5873
64*64	7.5556
32*32	7.4488
16*16	6.8918
8*8	5.6972
4*4	4
2*2	2



Tamaño	Entropía
256*256	4.0750
128*128	4.0685
64*64	3.9898
32*32	3.9306
16*16	3.6793
8*8	3.2296
4*4	2.7806
2*2	1.5000

¿Cómo crees que aparecen este tipo de imágenes?

¿Por qué al disminuir el tamaño disminuye la entropía?

¿Se te ocurriría algún ejemplo en el que al disminuir el tamaño no bajase la entropía?

IV Modelos para la compresión

IV.1 Modelos probabilísticos simples

El modelo probabilístico más simple que podemos usar es suponer que cada letra del alfabeto que se genera es independiente de las demás y que todas tienen la misma probabilidad. Es el llamado modelo de ignorancia que normalmente no representa bien la fuente y en el que, por desgracia, no podemos alcanzar mucha compresión.

Supongamos que M es el número de letras del alfabeto. Si una fuente S sigue el modelo de ignorancia su entropía vale:

$$H(\mathbf{S}) = -\sum_{i=1}^M \frac{1}{M} \log \frac{1}{M} = \log M$$

El modelo probabilístico siguiente en complicación **mantiene** la hipótesis de independencia, pero **elimina la asignación de la misma probabilidad a todas las letras**. Así pues, ahora tenemos para $\mathbf{A}=\{a_1,\dots,a_M\}$ el modelo de probabilidades $\mathbf{P}=\{P(a_1),\dots,P(a_M)\}$ siendo uno la suma de las probabilidades. Su entropía vale:

$$H(\mathbf{S}) = -\sum_{i=1}^M P(a_i) \log P(a_i)$$

¿Cómo sacamos estas probabilidades?

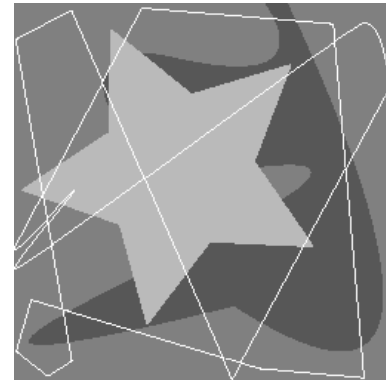
Veámoslo a continuación

- Para estimar la entropía o información contenida en una imagen, necesitamos un modelo de generación de la imagen
- **Modelo 1:** Las imágenes son producidas por una fuente que genera en cada posición valores independientes en $\{0,1,2,\dots,255\}$ todos con la misma probabilidad. Todos con la misma probabilidad.

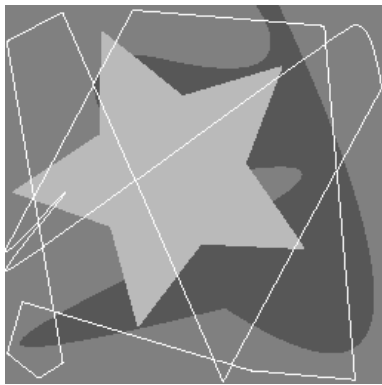
La entropía de la fuente sería

$$h(M_1) = H(M_1) = -\sum_{i=0}^{255} \frac{1}{256} \log\left(\frac{1}{256}\right)$$

$$h(M_1) = -\log(2^{-8}) = 8 \text{ bits} / \text{símbolo}$$



Podemos construir un modelo de fuente basado en la frecuencia relativa de cada uno de los niveles de gris en la imagen. Es decir, el histograma de la imagen suponiendo que las realizaciones son independientes. Veamos un ejemplo.



Nivel de gris	probabilidad
87	0.25
128	0.47
186	0.25
255	0.03
Otros	0

La entropía de la fuente sería

$$h(M_2) = H(M_2) = -[0.25 \log 0.25 + 0.47 \log 0.47 + 0.25 \log 0.25 + 0.03 \log 0.03]$$

$$h(M_2) = H(M_2) = 1.6614 \text{ bits / símbolo}$$

De nuevo: Fíjate que todavía no hemos abordado preguntas muy importante: ¿Existe algún código (asignación de secuencias de 0 y 1 a los niveles de gris) que alcance esta entropía?, ¿Podemos hacer algo con los datos (símbolos) para reducir esta entropía?

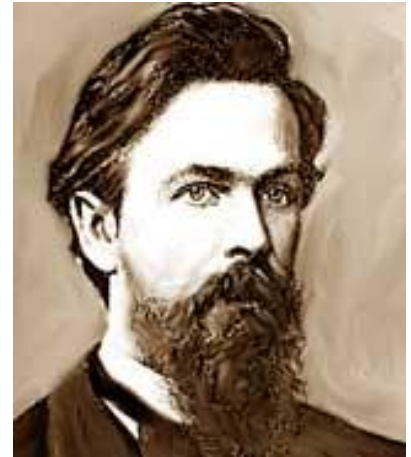
Si la hipótesis de independencia no es correcta, tenemos que utilizar otros modelos alternativos.

Ejemplos de estos modelos probabilísticos “más complicados” son los modelos de Markov que veremos a continuación

IV.2 Modelos de Markov

La forma más usada para representar dependencia entre datos es el uso de las cadenas de Markov.

Los modelos de Markov utilizados en compresión sin pérdida son las llamadas **cadenas de Markov en tiempo discreto**.



Andrei Andreivich Markov
(1856-1922)

Una sucesión de observaciones $\{x_n\}$ se dice que sigue un modelo de Markov de orden k-ésimo si para todo n

$$P(x_n | x_{n-1}, \dots, x_{n-k}) = P(x_n | x_{n-1}, \dots, x_{n-k}, \dots)$$

Es decir, el conocimiento de los k símbolos anteriores es equivalente al conocimiento de todo el pasado. Los valores tomados por el conjunto $\{x_1, \dots, x_k\}$ reciben el nombre de **estado del proceso**.

El modelo de Markov más usado es el de primer orden en el que se tiene

$$P(x_n | x_{n-1}) = P(x_n | x_{n-1}, x_{n-2}, \dots)$$

Podemos desarrollar diferentes modelos de primer orden dependiendo de la forma de dependencia entre las muestras.

Por ejemplo, podríamos utilizar el modelo lineal siguiente

$$x_n = x_{n-1} + \varepsilon_n$$

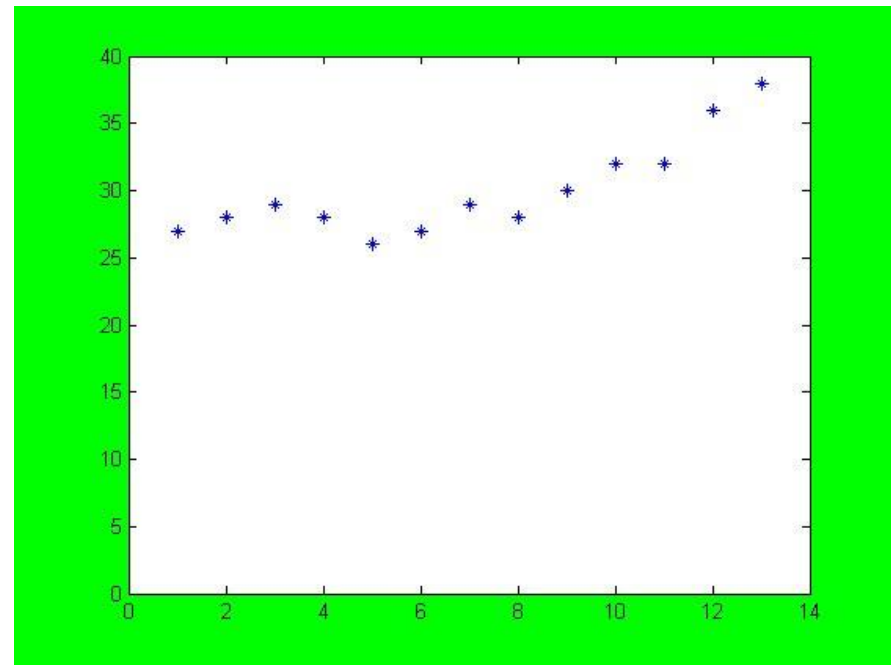
Donde ε_n es ruido (normalmente blanco). Este modelo se utiliza frecuentemente en la codificación de imágenes y voz. Veamos un ejemplo.

Ejemplo IV.2.1

Consideremos la secuencia $(x_1, x_2, \dots$

27 28 29 28 26 27 29 28 30 32 34 36
38

La sucesión no parece seguir una ley sencilla como en el ejemplo anterior.



Consideremos el modelo $x_{n+1} = x_n + d_{n+1} \quad n = 1, 2, \dots$

Podemos transmitir o almacenar x_1 y todas las diferencias, es decir:

27 1 1 -1 -2 1 2 -1 2 2 2 2 2

Discutiremos este tipo de técnicas que reciben el nombre de esquemas de codificación predictiva en el tema 6 para compresión sin pérdida y con posterioridad para compresión con pérdida.

Supongamos que tenemos un texto en Blanco (B) y Negro (N). Pensamos que la aparición de un píxel blanco o negro en la siguiente observación depende de que hayamos observado un píxel blanco o negro en la observación actual.

Del texto estimamos las probabilidades $P(B)$, $P(N)$, $P(B|N)$, $P(N|N)$, $P(B|B)$ y $P(N|B)$. **¿Cómo lo hacemos?**

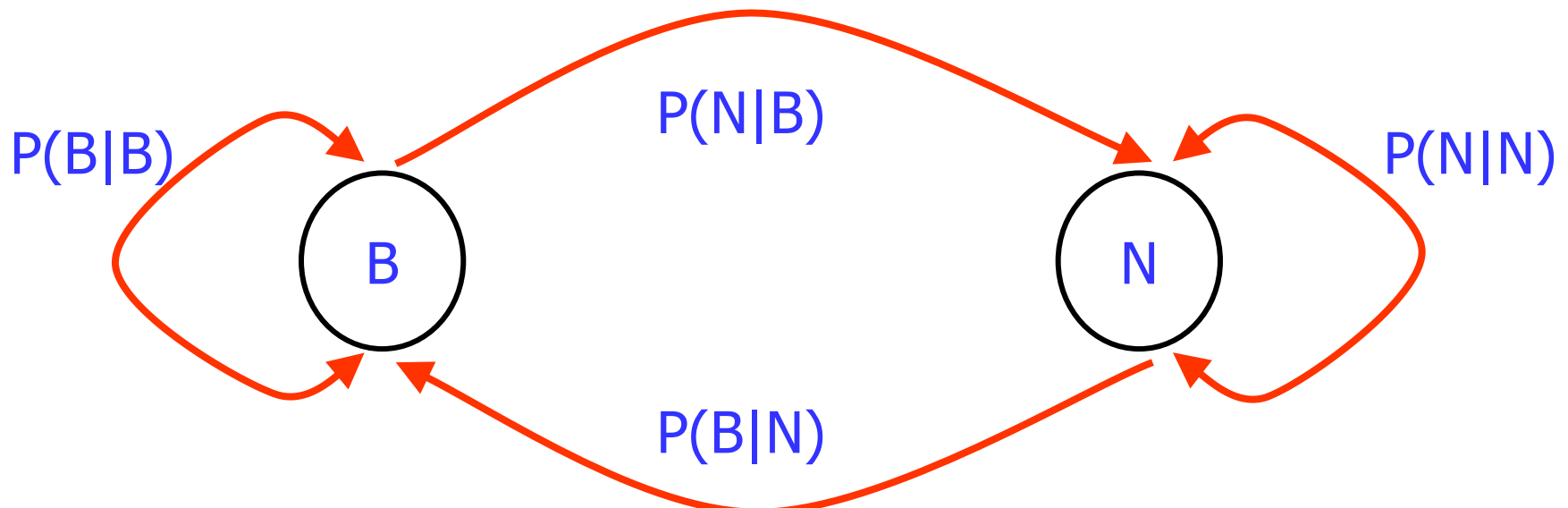
Una vez observada la letra N, $P(B|N)$ y $P(N|N)$ cumplen

$$P(B|N) + P(N|N) = 1$$

Una vez observada la letra B, $P(B|B)$, $P(N|B)$ cumplen

$$P(B|B) + P(N|B) = 1$$

Gráficamente:



Ejemplo IV.2.2

Supongamos que las probabilidades de nuestra fuente **S** son con

$$P(B) = 0.8, \quad P(N) = 0.2$$

$$P(B | B) = 0.99, P(N | B) = 0.01 \quad P(B | N) = 0.3, P(N | N) = 0.7$$

Queremos codificar a partir del segundo término una sucesión de Bs y Ns. Para el primero, obviamente, usamos como límite inferior la entropía correspondiente a las probabilidades $P(B)=0.8$ y $P(N)=0.2$.

Si suponemos independencia y usamos, desde el segundo término en adelante, $P(B)=0.8$, $P(N)=0.2$ tendremos

$$H(\mathbf{S}) = -0.8 \log 0.8 - 0.2 \log 0.2 = 0.206 \text{ bits}$$

También podemos basarnos en la entropía de las distribuciones condicionadas a los dos casos posibles: ha salido una N

$$\begin{aligned} H(\mathbf{S}_N) &= -P(N | N) \log(P(N | N)) - P(B | N) \log(P(B | N)) \\ &= -0.3 \log 0.3 - 0.1 \log 0.7 = 0.881 \text{ bits} \end{aligned}$$

o ha salido una B

$$\begin{aligned} H(\mathbf{S}_B) &= -P(N | B) \log(P(N | B)) - P(B | B) \log(P(B | B)) \\ &= -0.01 \log 0.01 - 0.99 \log 0.99 = 0.081 \text{ bits} \end{aligned}$$

Por tanto, para la codificación a partir del segundo símbolo tendríamos como cota inferior

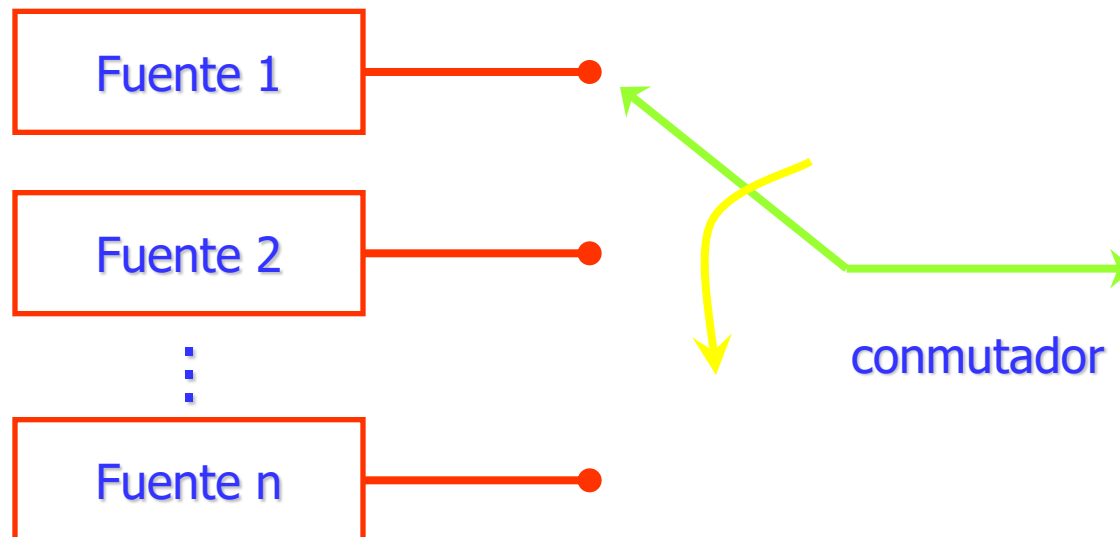
$$\begin{aligned} H(\mathbf{S}) &= P(B)H(\mathbf{S}_B) + P(N)H(\mathbf{S}_N) \\ &= 0.2 \times 0.881 + 0.8 \times 0.081 = 0.107 \text{ bits} \end{aligned}$$

Casi la mitad de la obtenida cuando se supone independencia.

IV.3 Modelo de fuente compuesta

Los modelos de fuente compuesta se pueden representar como un conjunto de fuentes individuales \mathbf{S}_i , en el que cada fuente \mathbf{S}_i tiene su modelo \mathbf{M}_i y un conmutador que selecciona la fuente \mathbf{S}_i con probabilidad \mathbf{P}_i .

El modelo es muy rico y puede usarse para describir procesos muy complicados.



V. Más sobre modelización

Uno de los aspectos más importantes de la compresión es la caracterización (modelización) de los datos a comprimir.

Cualquier algoritmo de compresión podría dividirse en dos fases:

Modelización (**HACER ALGO** del tema anterior) donde extraemos información sobre la redundancia en los datos y describimos la redundancia como un modelo y

Codificación de la descripción del modelo y de los datos en el modelo.

Ejemplo V.1

Consideremos la sentencia siguiente:

abbarayaranbarraybranbfarbfaaarbaway

donde b denota espacio en blanco. Podemos usar tres bits por símbolo para codificarla. También podemos usar la siguiente tabla para codificarla con longitud variable:

a	1
<u>b</u>	001
b	01100
f	0100
n	0111
r	000
w	01101
y	0101

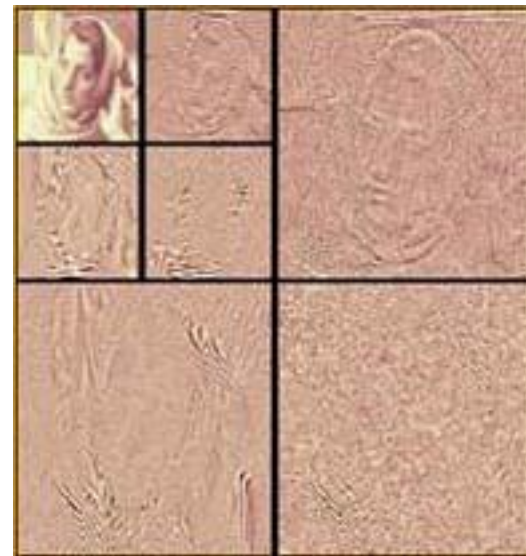
Si usásemos estos códigos, la secuencia será codificada utilizando 106 bits. Puesto que tenemos 41 símbolos el modelo utiliza en media 2.58 bits por símbolo. El factor de compresión es $3:2.56=1.16:1$. Estos modelos se basan en la redundancia estadística.

Usando texto hay palabras que se repiten frecuentemente, podemos construir una lista con ellas y representarlas por su posición en la lista. Estamos ante los esquemas de compresión basados en diccionarios.

Por último, en determinadas situaciones (muchas) será más conveniente descomponer los datos en un conjunto de componentes, podemos estudiar cada componente separadamente y usar un modelo para cada una de las componentes. Aparecerán los métodos basados en transformadas: wavelets, transformada coseno discreta, etc



Imagen



Transformación wavelet

VI. Codificación

VI.1 Códigos de Longitud Fija y Longitud Variable

- Codificación (para nosotros) = asignación de secuencias binarias a las letras de un alfabeto.
- El conjunto de secuencias binarias se llama **código**. Los miembros individuales del conjunto se llaman **palabras del código**.
- **El número medio de bit por letra (símbolo) se llama tasa o razón (rate).**

$$\text{Tasa} = \sum_{a \in \text{letras del alfabeto}} l_a p_a$$

Probabilidad de la letra

Longitud de la representación de la letra

Las palabras del código pueden tener una longitud fija y se habla entonces de **códigos de longitud fija**. Ejemplos:

- Código ASCII
- Asignación de un byte (8 bits) a los posibles 256 valores de una imagen de niveles de gris
- Si tenemos M símbolos necesitamos $\lceil \log_2 M \rceil$ bits para una codificación de longitud fija

Códigos de Longitud variable (VLC)

- **Motivación:** la intuición de que se puede alcanzar mayor compresión si asignamos a símbolos más (menos) probables palabras del código más (menos) cortas
- Un código de longitud variable c asigna a cada símbolo a_j en la fuente del alfabeto A una **palabra de código** $c(a_j)$. El número de bits en $c(a_j)$ recibe el nombre de **longitud** $l(a_j)$ de $c(a_j)$.
- Palabras de código sucesivas en un VLC se transmiten como una secuencia de dígitos binarios sin separación entre palabras (no comas ni espacios). El decodificador, dado un punto de comienzo, debe determinar donde están las fronteras entre las palabras del código (parsing)

Códigos de Longitud variable (VLC)

Ejemplo: para el alfabeto $A = \{a, b, c\}$, un posible VLC es:

$$\begin{array}{ll} c(a) = 0 & l(a) = 1 \\ c(b) = 10 & l(b) = 2 \\ c(c) = 11 & l(c) = 2 \end{array}$$

La secuencia *cab* es codificada como ***11010***

- Códigos de Longitud fija
 - El principal problema de los códigos de longitud fija es su **ineficiencia**
- Los VLC
 - Usan un número distinto de dígitos binarios para representar cada símbolo
 - Asignan palabras de código más cortas a los símbolos más frecuentes
 - Asignan palabras de código más largas a los símbolos menos frecuentes
 - Producen una representación más eficiente

VI.2 Códigos decodificables de modo único

Consideremos el ejemplo siguiente (ver tabla): nuestro alfabeto tiene cuatro letras con las probabilidades que se muestran y pensamos en asignarles los siguientes códigos

La entropía de esta fuente es 1.75 bits/símbolo.

Letra	Probabilidad	Código 1	Código 2	Código 3	Código 4
a_1	0.5	0	0	0	0
a_2	0.25	0	1	10	01
a_3	0.125	1	00	110	011
a_4	0.125	10	11	111	0111
Longitud media		1.125	1.25	1.75	1.875

Letra	Probabilidad	Código 1
a_1	0.5	0
a_2	0.25	0
a_3	0.125	1
a_4	0.125	10
Longitud media		1.125

El primer código parece el mejor en cuanto a la longitud media.

Sin embargo los símbolos a_1 y a_2 han sido asignados a la misma palabra.

Cuando leemos un cero no sabemos de qué símbolo viene

Necesitamos trabajar con códigos que asignan **palabras distintas a símbolos distintos**. Son los llamados **códigos singulares**

Letra	Probabilidad	Código 2
a_1	0.5	0
a_2	0.25	1
a_3	0.125	00
a_4	0.125	11
Longitud media		1.25

Este código no parece tener el problema de la ambigüedad.

Sin embargo si tenemos la secuencia de bits 110 su decodificación no es única.

Nos gustaría que la secuencia tuviera una **decodificación única** por el decodificador.

Letra	Probabilidad	Código 3
a_1	0.5	0
a_2	0.25	10
a_3	0.125	110
a_4	0.125	111
Longitud media		1.75

La regla de decodificación de este código es simple:

Acumular dígitos hasta que encontremos un cero o encontremos tres unos consecutivos.

Este tipo de **códigos** recibe el nombre de **instantáneos**. El codificador sabe cuando la palabra del código está completa.

Letra	Probabilidad	Código 4
a_1	0.5	0
a_2	0.25	01
a_3	0.125	011
a_4	0.125	0111
Longitud media		1.875

La regla de decodificación de este código es muy simple:

La decodificación consistirá en acumular dígitos hasta que encontremos un cero. El bit anterior al cero es el último de la anterior palabra.

Este código **no es instantáneo**, aunque casi cumple la condición de instantaneidad. La condición de instantaneidad es interesante pero **no es obligatoria** como veremos a continuación.

Letra	Código 4
a_1	0
a_2	01
a_3	11

Estudiemos un poco más la instantaneidad de los códigos. Consideremos la codificación proporcionada por la tabla a la izquierda y decodifiquemos la secuencia **011111111111111111**

Decodificando empezando por a_1 (0) obtendríamos:

0	11	11	11	11	11	11	11	11	1
a_1	a_3	a_3	a_3	a_3	a_3	a_3	a_3	a_3	No válido

Empezando por a_2 obtendríamos la decodificación válida:

01	11	11	11	11	11	11	11	11
a_2	a_3	a_3	a_3	a_3	a_3	a_3	a_3	a_3

Obviamente el código no es instantáneo, pero probaremos después que todas las secuencias de palabras de este código (no sólo ésta) son decodificables de modo único.

Letra	Código 4
a_1	0
a_2	01
a_3	10

Consideremos la codificación proporcionada por la tabla de la izquierda y decodifiquemos la secuencia **01010101010101010**

Decodificando empezando por a_1 (0) obtendríamos la codificación válida:

0	10	10	10	10	10	10	10	10
a_1	a_3	a_3	a_3	a_3	a_3	a_3	a_3	a_3

Empezando por a_2 obtendríamos la decodificación también válida:

01	01	01	01	01	01	01	01	0
a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_1

Por tanto este código no produce secuencias decodificables de modo único.

VI.3 Test para decodificación única

Definición: Supongamos que tenemos dos palabras, a y b , de un código binario donde a tiene k bits y b tiene n bits con $k < n$.

Si los k primeros bits de b coinciden con a , diremos que a es un prefijo de b .

Los restantes $n-k$ bits de b reciben el nombre de sufijo restante.

Ejemplo: si $a=010$ y $b=01011$ entonces a es prefijo de b y el sufijo restante es 11

Test sobre decodificación única

1. Construir una lista con todas las palabras del código,
2. Examinar todos los pares de palabras del código para ver si una palabra es prefijo de otra,
3. Cada vez que encontremos una palabra prefijo de otra, añadir el sufijo restante a la lista antes construida,
4. Repetir el proceso usando esta lista mayor,
5. Continuar el proceso hasta que ocurra uno de los siguientes sucesos
 1. Obtenemos un sufijo restante que es una palabra del código,
 2. No hay más sufijos restantes nuevos.

En el primer caso el código no será decodificable de modo único y en el segundo sí.

Ejemplo VI.1

Consideremos el código $\mathcal{C}=\{0,01,11\}$ y la lista inicial $\mathcal{L}=\{0,01,11\}$.

La palabra 0 es prefijo de 01 y no hay ningún otro par de palabras que cumplan esta condición. Generamos la nueva lista

$$\mathcal{L}=\{0,01,11,1\}.$$

En ella, el código sufijo añadido no es una palabra del código original.

Repetimos el proceso con \mathcal{L} pero vemos que no añadimos ningún sufijo restante nuevo.

El código es decodificable de modo único.

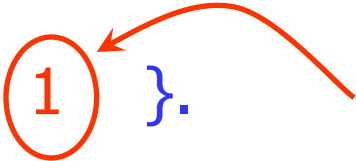
Ejemplo VI.2

Consideremos el código $\mathbf{C}=\{0,01,10\}$ y la lista $\mathbf{L}=\{0,01,10\}$

La palabra 0 es prefijo de 01 y no hay ningún otro par de palabras que cumplan esta condición. Generamos la nueva lista

$$\mathbf{L}=\{0,01,10, \textcircled{1}\}.$$

añadido



En ella, el código sufijo añadido no es una palabra del código original.

Repetimos el proceso con \mathbf{L} y vemos que 1 es prefijo del 10 y el sufijo restante 0 es una palabra del código original.

El código no es decodificable de modo único.

VI.4 Códigos prefijo

Código prefijo: código en el que ninguna palabra del código es prefijo de otra palabra del código.

Para comprobar si un código es prefijo ¿qué hacemos?:

1. Dibujamos un árbol binario para el código (la rama izquierda se utilizará para el cero y la derecha para el uno).
2. En un código prefijo las palabras del código estarán asociadas sólo a nodos externos (hojas).

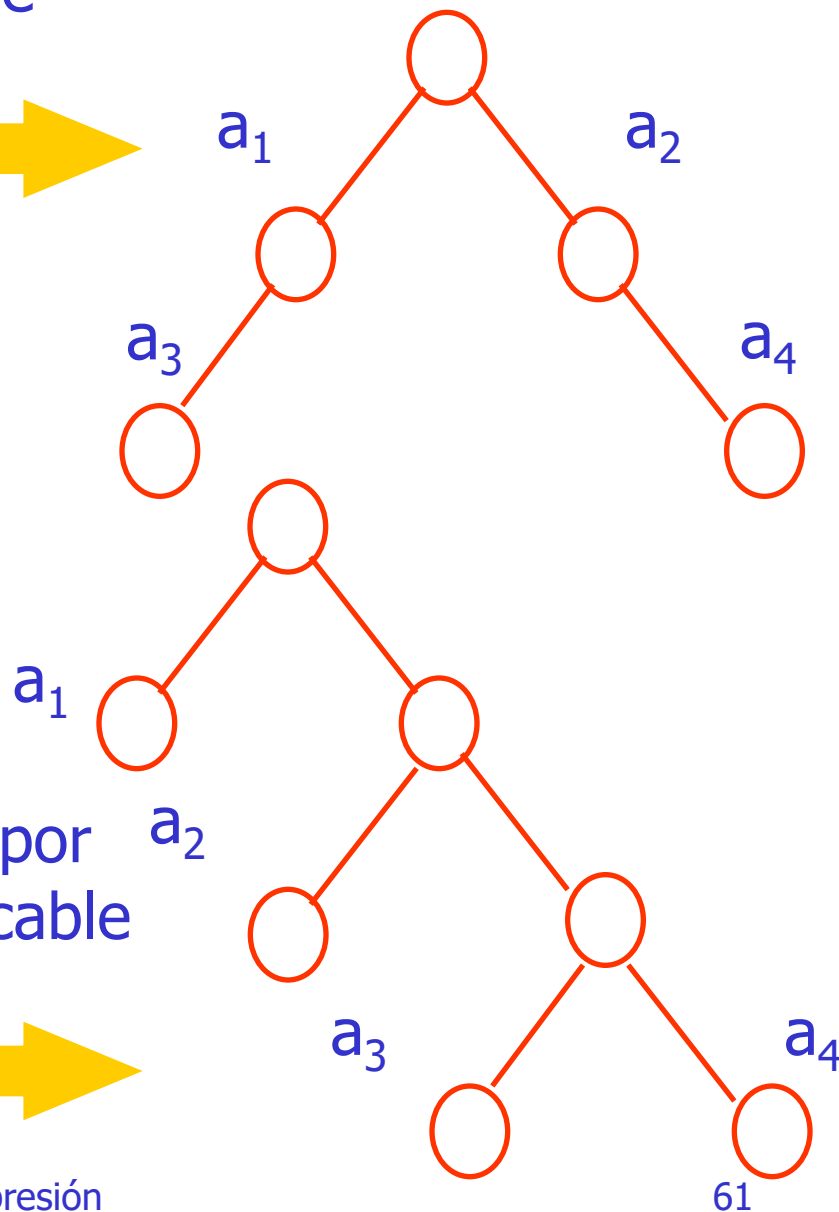
Observa que por construcción los códigos prefijos son decodificables de modo único.

Letra	Código 2
a_1	0
a_2	1
a_3	00
a_4	11

No es prefijo, ni
decodificable de
modo único



Árbol binario



Letra	Código 3
a_1	0
a_2	10
a_3	110
a_4	111

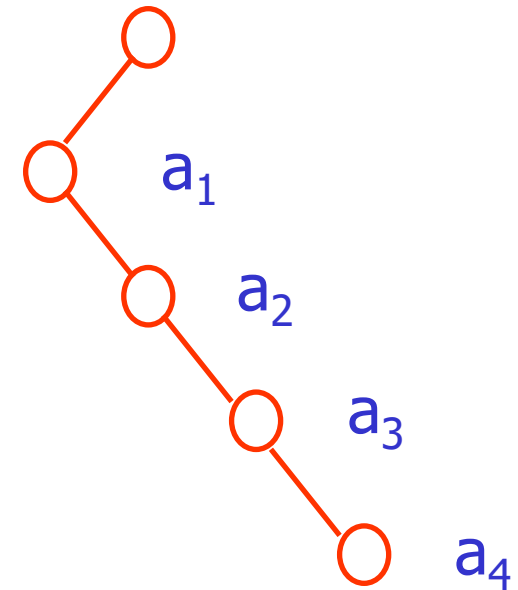
Sí es prefijo y, por
tanto, decodificable
de modo único



Árbol binario

Letra	Código 4
a_1	0
a_2	01
a_3	011
a_4	0111

No es prefijo, pero
sí decodificable de
modo único



Puede probarse fácilmente que este código cumple la condición de decodificación única. Ser código prefijo es una condición suficiente para ser decodificable de modo único pero no es necesaria.

VI.5 Decodificación de una secuencia de palabras en código prefijo

Para decodificar una secuencia codificada usando un código prefijo sólo tenemos que ejecutar el siguiente algoritmo:

Repetir

- Comenzar en el nodo raíz del árbol

- repetir

 - if leer bit = 1 entonces ir derecha

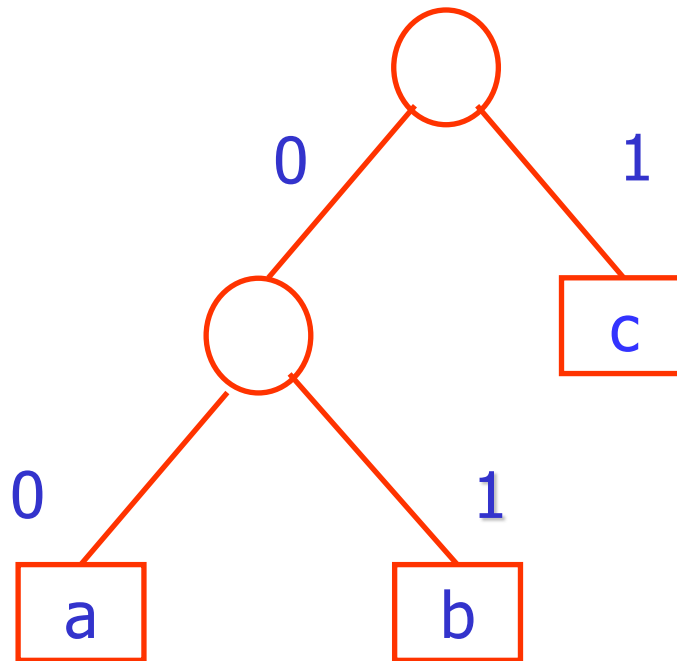
 - else ir izquierda

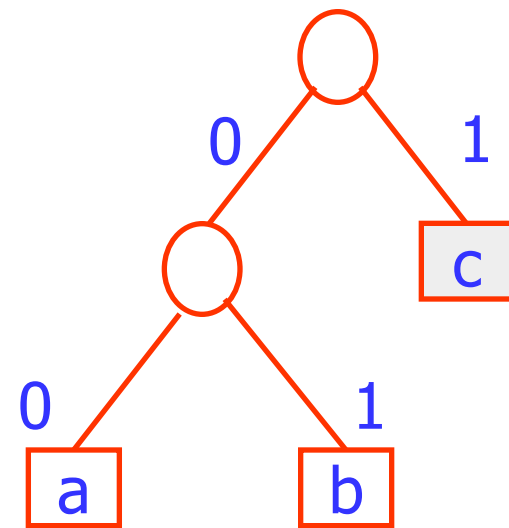
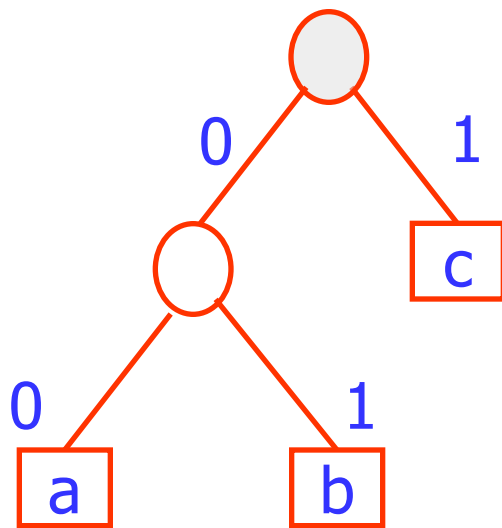
- hasta que el nodo es una hoja

- mostrar la letra asociada a la palabra del código obtenida

Hasta final de la secuencia

Decodifiquemos la secuencia 110001 para el siguiente árbol que corresponde a un código prefijo.





110001

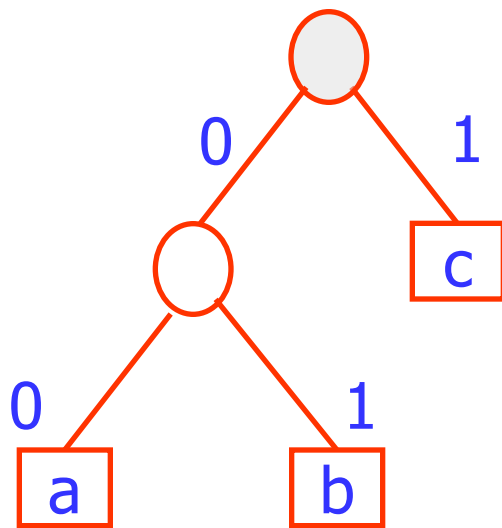


110001

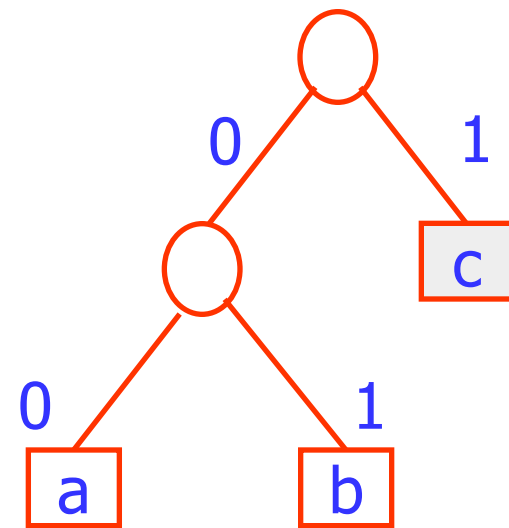
Decodificación actual:C

Indica nodo que vamos recorriendo.

Dígito en rojo, el bit que vamos recorriendo

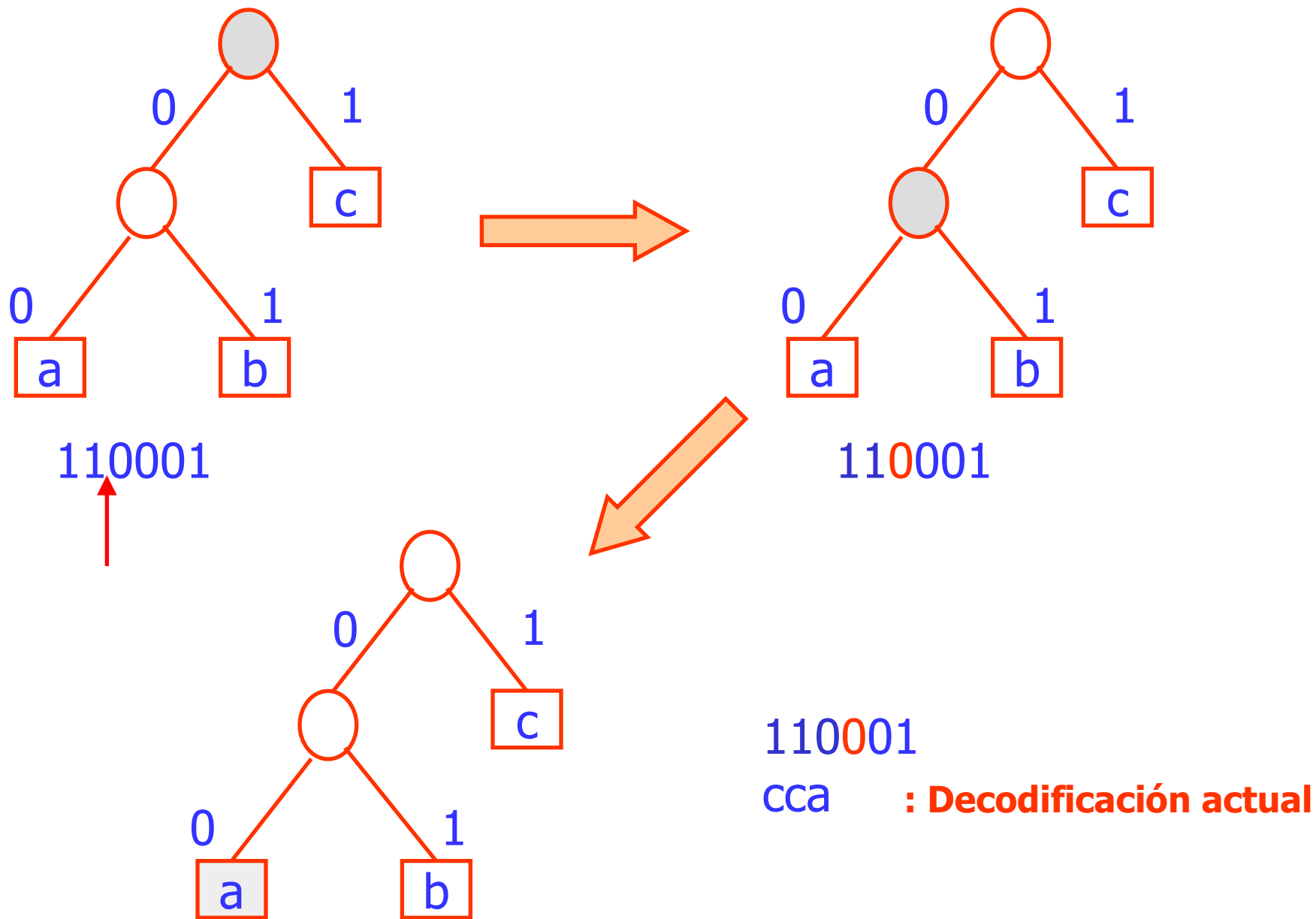


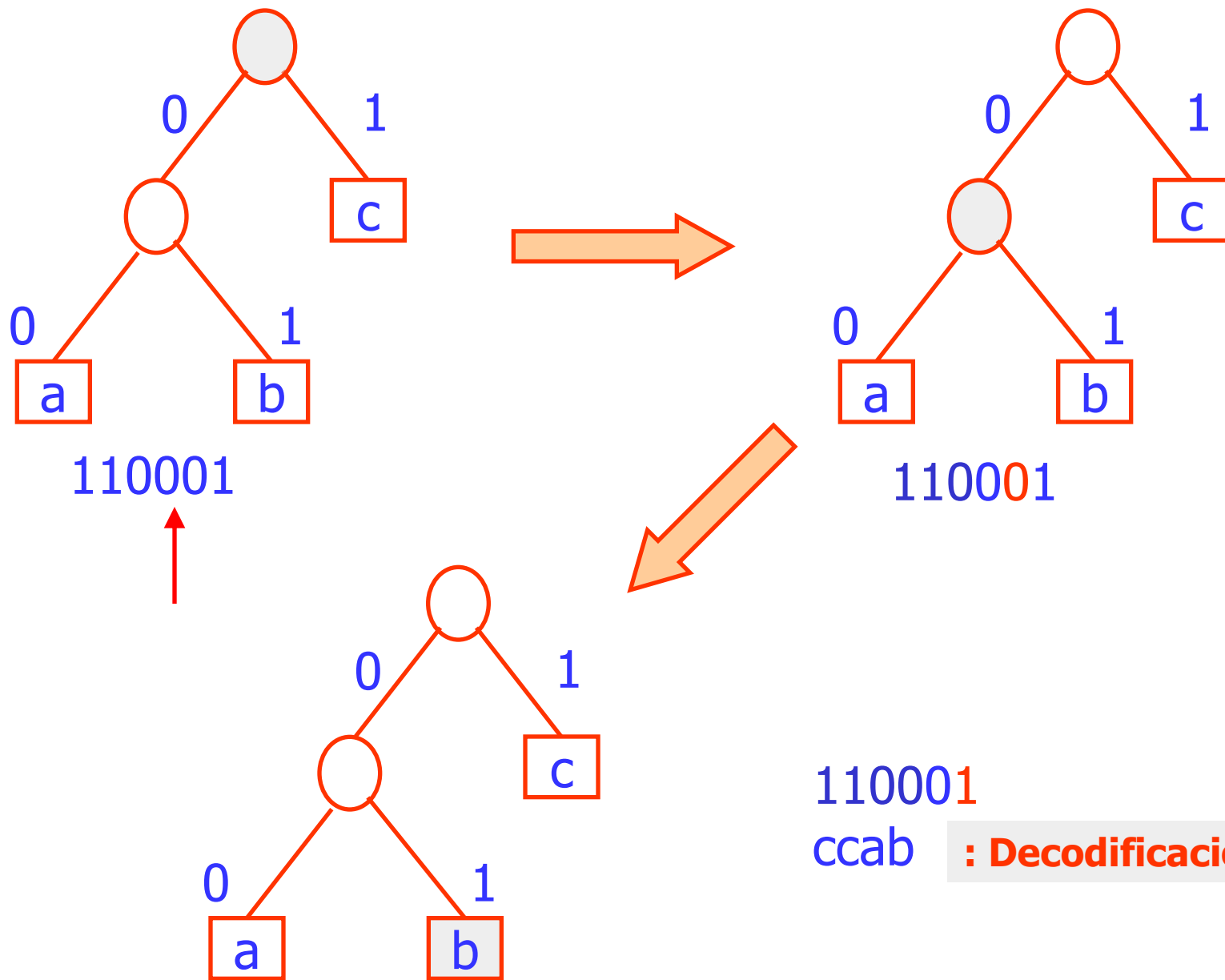
110001



110001

Decodificación actual: CC





110001
ccab : Decodificación actual

VI.6 Desigualdad de Kraft-McMillan

¿Podríamos encontrar palabras más cortas si no usásemos sólo códigos prefijo?

Utilizando los siguientes teoremas, puede probarse que para cualquier código no prefijo decodificable de modo único, podemos encontrar siempre un código prefijo con la misma longitud de palabras.

Teorema (desigualdad de Kraft-McMillan). Sea c un código decodificable de modo único con N palabras de longitudes l_1, l_2, \dots, l_N entonces

$$K(C) = \sum_{i=1}^N 2^{-l_i} \leq 1$$

Teorema. Dado un conjunto de enteros l_1, l_2, \dots, l_N que cumplen

$$\sum_{i=1}^N 2^{-l_i} \leq 1$$

Podemos encontrar siempre un código prefijo con longitudes de palabra l_1, l_2, \dots, l_N

VII Consideraciones finales

Volvamos, para terminar, a la entropía.

¿Qué ocurre si queremos codificar una secuencia dada y no conocemos las probabilidades de las letras del alfabeto?.

Consideremos la secuencia

1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10

si estimamos la probabilidad de ocurrencia de cada letra mediante su frecuencia en la secuencia obtenemos

$$P(1) = P(6) = P(7) = P(10) = \frac{1}{16}$$

$$P(2) = P(3) = P(4) = P(5) = P(8) = P(9) = \frac{2}{16}$$

Usando estas probabilidades, la entropía de la fuente sería

$$H = -\sum_{i=1}^{10} P(i) \log P(i) = 3,25 \text{ bits}$$

Puede probarse que lo mejor que podemos hacer para codificar esta secuencia es utilizar $16 \times 3,25$ bits supuesto que las 16 observaciones vienen de variables independientes idénticamente distribuidas.

Es decir, no vamos a encontrar un código decodificable de modo único que asigne independientemente palabras de código a los números del 1 al 10 que utilice menos de $16 \times 3,25$ bits.

Consideremos de nuevo la secuencia

1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10

ahora, para eliminar la correlación entre los datos calculamos la diferencia entre cada valor y el anterior. Obtenemos entonces la secuencia (además del uno inicial)

1 1 -1 1 1 1 -1 1 1 1 1 1 -1 1 1

de la que calculamos

$$P(1) = \frac{12}{15} \quad P(-1) = \frac{3}{15} \quad H(S) = 0,70 \text{ bits}$$

y podríamos codificar la secuencia desde el segundo elemento en el mejor de los casos, bajo las hipótesis que sabemos, con $15 \times 0,70$ bits.

Observemos que en este caso tendríamos que codificar la secuencia

1 1 -1 1 1 1 -1 1 1 1 1 1 -1 1 1

y también enviar el primer valor, un uno y el modelo para la secuencia, es decir,

$$x_n = x_{n-1} + r_n \quad n = 2, \dots, 16 \quad x_1 = 1$$

donde x_n es el n -ésimo elemento de la secuencia original y r_n es el n -ésimo elemento de la secuencia de residuos.

Consideremos ahora la siguiente secuencia

1 2 1 2 3 3 3 3 1 2 3 3 3 3 1 2 3 3 1 2

Si miramos un símbolo cada vez tendremos

$$P(1) = P(2) = \frac{1}{4} \quad P(3) = \frac{1}{2}$$

$$H(S) = 1.5 \text{ bits/símbolo}$$

Necesitaríamos $20 \times 1.5 = 30$ bits para representar la secuencia. Sin embargo, si consideramos dos símbolos consecutivos tendremos

$$P(1\ 2) = P(3\ 3) = \frac{1}{2}$$

$$H(S) = 1 \text{ bit/símbolo}$$

Necesitaríamos por tanto, con este modelo, $10 \times 1 = 10$ bits para representar la secuencia.

1. Obviamente cuanto mayor sea el tamaño de los bloques de letras del alfabeto más podemos, en principio, mejorar la compresión.
2. Sin embargo, hay un límite de tipo práctico en esta aproximación. ¿Cuál?

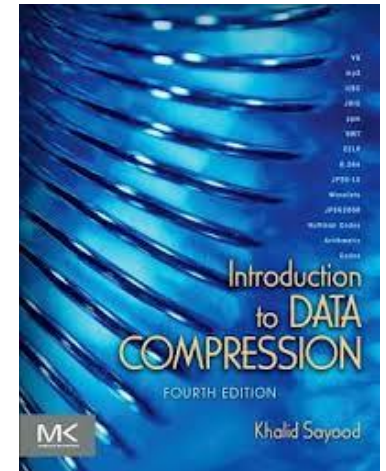
En resumen es necesario realizar una buena modelización de nuestra fuente. Es bueno, muchas veces, **hacer algo** (transformar) con los datos antes de iniciar el proceso de compresión

VII. Ejercicios Teórico-Prácticos

Ver relación de problemas

VIII Bibliografía

K. Sayood, "Introduction to Data Compression", Morgan and Kaufmann, 2012.



Apuntes de Prof. Luis Torres, Codificación de Contenidos Audiovisuales, Escuela Técnica Superior de Ingeniería de Telecomunicación de Barcelona, Universidad Politécnica de Catalunya